Unsupervised Learning of the 4D Audio-Visual World from Sparse Unconstrained Real-World Samples

Aayush Bansal

CMU-RI-TR-21-02

January 08, 2021

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Deva K. Ramanan, Co-Chair
Yaser A. Sheikh, Co-Chair
Martial H. Hebert, Carnegie Mellon University
David A. Forsyth, University of Illinois, Urbana Champaign
Alexei A. Efros, University of California, Berkeley

Thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

To Yaser and Deva.

The world knows you for your intellect,
but your kindness and patience far exceeds your intellect.

Abstract

We, humans, can easily observe, explore, and analyze our four-dimensional (4D) audio-visual world. We, however, struggle to share our observation, exploration, and analysis with others. In this thesis, our goal is to learn a computational representation of the 4D audio-visual world that can be: (1) estimated from sparse real-world observations; and (2) explored to create new experiences. We introduce Computational Studio for observing, exploring, and creating the 4D audio-visual world, thereby allowing humans to communicate with other humans and machines effectively without any loss of information. Computational Studio serves as an environment for non-experts to construct and creatively edit the 4D audio-visual world from sparse real-world samples. There are three essential components of the Computational Studio: (1) How can we densely observe the 4D visual world?; (2) How can we communicate the audio-visual world using examples?; and (3) How can we interactively explore the audio-visual world?

The first part introduces capturing, browsing, and reconstructing the 4D visual world from sparse real-world multi-view samples. We bring together insights from classical image-based rendering and neural rendering approaches. Crucial to our work are two components: (1) Fusing information from sparse multi-views to create dense 3D point clouds; and (2) Fusing multi-view information to create new views. Though captured from discrete viewpoints, the proposed formulation allows us to do dense 3D reconstruction and 4D visualization of dynamic events. It also enables us to move around the space-time of the event continuously and facilitate: (1) Freezing the time and exploring 3D space; (2) Freezing the 3D space and moving through time; and (3) Simultaneously changing both time and 3D space. Without any external information, our formulation allows us to get a dense depth map and a foreground-background segmentation, which enables us to efficiently track objects in a video. In turn, these properties allow us to edit the videos and reveal occluded things in a 3D space, provided it is visible in any view.

The second part details the example-based synthesis of the audio-visual world in an unsupervised manner. Example-based audio-visual synthesis allows us to express ourselves easily. In this part, we introduce Recycle-GAN that combines spatial and temporal information via adversarial losses for an unsupervised video retargeting. This representation allows us to translate the contents from one domain to another while preserving the style native to the target domain. E.g., if our goal is to transfer the contents of John Oliver's speech to Stephen Colbert, then the generated content/speech should be in Stephen Colbert's own style. We then extend our work to audio-visual synthesis using Exemplar Autoencoders. Our approach builds on simple autoencoders that project out-of-sample data onto the distribution of the training set. We use Exemplar Autoencoders to learn the voice, stylistic prosody (emotions and ambiance), and visual appearance of a specific target exemplar speech. This work enables us to synthesize a natural voice for speech-impaired

individuals and do a zero-shot multi-lingual translation. Finally, we introduce PixelNN, a semi-parametric model that enables us to generate multiple outputs from a given input and examples.

The third part introduces human-controllable representations that allow a human user to interact with visual data and create new experiences on everyday computational devices. Firstly, we introduce OpenShapes that allows a user to interactively synthesize new images using a paint-brush and a drag-and-drop tool. OpenShapes runs on a single-core CPU to create multiple pictures from a user-generated label map. We then present simple video-specific autoencoders that enable human-controllable video exploration. This exploration includes a wide variety of video-analytic tasks such as (but not limited to) spatial and temporal super-resolution, object removal, video textures, average video exploration, associating various videos, video retargeting, and correspondence estimation within and across videos. Prior work has independently looked at each of these problems and proposed different formulations. We observe that a simple autoencoder trained (from scratch) on multiple frames of a specific video enables one to perform a large variety of video processing and editing tasks without even optimizing for a single task. Finally, we present a framework that allows us to extract a wide range of low-mid-high level semantic and geometric scene cues that could be understood and expressed by both humans and machines.

We follow the concept of exemplar and test-time training for various formulations proposed in this thesis. This unique combination allows us to continually learn the audio-visual world in a streaming manner. The last part of this thesis extends our work on continual and streaming learning of the audio-visual world to learning visual-recognition tasks given a few labeled examples and a (potentially) infinite stream of unlabeled examples. Our approach continually improves a task-specific representation without any task-specific knowledge. We construct a schedule of learning updates that iterates between pre-training on novel segments of the stream, and fine-tuning on the small and fixed labeled dataset. Contrary to popular approaches that use massive computing resources for storing and processing data, streaming learning requires modest computational infrastructure since it naturally breaks up massive datasets into slices that are manageable for processing. Streaming learning can help democratize research and development for scalable and lifelong ML.

Computational Studio is a first step towards unlocking the full degree of creative imagination, which is currently limited to the human mind by the limits of the individual's expressivity and skills. It has the potential to change the way we audio-visually communicate with other humans and machines.

Acknowledgements

Ohana means family.
Family means nobody gets left behind, or forgotten.

— Lilo & Stitch

The universe has been beyond generous to keep me in the company of wonderful human beings. The last ten years have been a roller coaster ride and shaped me in ways I wouldn't have imagined at the beginning of this journey. Carnegie Mellon University (CMU) has been an integral part of this journey. It is one of the few places in the world for a strong interdisciplinary research. I immensely benefited from many conversations with folks in RI, MLD, HCII, ETC, SCS, School of Drama, and College of Fine Arts. I, wholeheartedly, thank everyone I met during this journey – I have strived to learn from every interaction with each one of you. I will try to list everyone here but I sincerely apologize if I missed anyone. Let me begin with my wonderful advisors – Deva Ramanan and Yaser Sheikh. If there was ever a golden moment in my life, it was the 15 minutes of my meeting with Yaser in 2015 when he agreed to be my advisor. Yaser has started his new venture at Facebook during that time and was perhaps not interested in taking a pupil under his tutelage. I, somehow, convinced him for a meeting. At the end of the meeting, Yaser said, "I came to say NO, but now I am tempted to work with you. However, I am busy and cannot singly advise you. Is there someone who would be willing to co-advise?". I mentioned him of my conversation with Deva. Yaser and Deva talked to each other, and in a few minutes – I got an email from them saying, "We are excited to work with you. Would you be interested?". Rest is history. It is hard to imagine life without Deva and Yaser. Despite their busy schedules in the last five years, they have tried to be available whenever I needed even if it meant meeting late in the night or early mornings on a weekend. I cannot thank you enough for your kindness and patience, your efforts on shaping me and not giving up.

I am thankful to Srinivasa Narasimhan for many discussions in the last five years. Thank you for always being available and patiently listening to me! I am always in awe of your high standards of research, and your ability to see things clearly and easily cut through the clutter. I also thank Abhinav Gupta who was my Masters' advisor. Abhinav imbibed in me the confidence and the ability to listen to my own instincts. I am thankful to Martial Hebert, David Forsyth, and Alyosha Efros for serving on my thesis committee despite their busy schedules. Thanks Martial for handpicking me as your teaching assistant for the geometry course. I learnt things in a hard way but you made sure that I learn them anyway. Thanks David for the various conversations and giving me a new perspective in our every discussion. I always look forward for a conversation with you! Thanks Alyosha for being the champion of big visual data, unsupervised learning, exemplar learning, cycle consistency, and nearest neighbors. I also thank you for selflessly promoting my

work. I clearly remember the moment at ECCV'18 in Munich when you dropped by my poster, took pictures, posted it on Facebook, and then told everyone out there about my work. Even more important, I do not have words to express my deep gratitude for giving me an outright and honest feedback – the medicine was sometimes bitter but the patient needed it. I thank Takeo Kanade for taking out time to meet whenever he visited Pittsburgh and having very long conversations ranging from little experimental details to the art of presentation. Your enthusiasm is infectious! I aim to acquire and sustain the level of energy you possess. I am also thankful to other faculty members on the second floor of Smith Hall – Kris Kitani, Fernando de la Torre, Simon Lucey, David Held, Daniel Huber, Ioannis Gkioulekas, Kayvon Fatahalian, Jessica Hodgins, and Jim McCann. You have always been available for the discussion! I am also thankful to other faculty members at CMU and NREC, especially Mike Erdmann, Chris Atkeson, Drew Bagnell, Bob Frederking, Mike Tarr, Alan Black, Jim Herbsleb, Jean Oh, Katerina Fragkiadaki, Nica Ross, Larry Shea, Zeynep Temmel, Heather Kelley, and Jun-Yan Zhu.

I am fortunate to have spent time as an intern at Facebook Reality Labs, Pittsburgh. I learnt a lot from the discussion with a lot of cool researchers there, especially Shugao Ma, Hernan Badino, Jason Saragih, Iain Matthews, Stephen Lombardi, Tomas Simon, and Sahana Vijai. I also made some good friends during this internship, including Chia-Yin Tsai (the coolest person I have ever met) and Shunsuke Saito (who is my Samurai, the great). My internship at Adobe Inc in the amazing San Francisco is the finest experience of my life-time. A big reason for that experience was my mentor, Bryan Russell, who is not just amazing beyond description but also the most humble person in the scientific community I have met so far. During this time, I also got an opportunity to listen to and interact with many excellent computer graphics researchers especially David Salesin and Aaron Hertzmann. Your work on *Image Analogies* has been an inspiration for me. The internship was also fun because brilliant people like Saining Xie, Ardavan Saedi, Anders Oland, Nam-Wook Kim, and Ben Eysenbach were around.

Prior to graduate school, all the burden of carrying me was on Devi Parikh. I am fortunate to have worked closely with her. Beyond everything, I am extremely glad that Devi forcefully sent me to CMU and Pittsburgh. I was reluctant but it is the finest decision of my life in retrospect. Both CMU and Pittsburgh changed me. In the last few years, I have also got an opportunity to interact with many great researchers throughout the world – Maneesh Agrawala, Christian Theobalt, Jean Ponce, Larry Zitnick, Ali Farhadi, Andy Gallagher, Moshe Bar, Cynthia Rudin, Gerard Pons-Mull, Tali Dekel, Mina Cikara, Aude Oliva, Josef Sivic, Henry Fuchs, Hany Farid, Charles Blundell, Greg Mori. Thank you for many wonderful discussions! I am also thankful to Michal Irani for her work on image-specific learning and visual composition has been an inspiration for a lot of my work.

I have been at CMU for quite a long time that many who joined after me have graduated long before I am writing this thesis. The good thing is I got an opportu-

nity to interact with a long list of star students during my time at CMU. Amongst those, thanks to my favorite couples – Sankalp Arora and Prerana Paliwal, Geetesh Dubey and Shreya Kakkar, and Abhinav Shrivastava and Varuni Saxena for celebrating every little milestone of my life! I am grateful to many citizens of Smith Hall and CMU – Aravindh Mahendran, Minh Vo, Hanbyul Joo, Tomas Simon, Gines Hidalgo Martinez, Zhe Cao, Shih-En Wei, Varun Ramakrishna, Xulong Li, Natasha Kholgade, Shu Kong, Peiyun Hu, Martin Li, Ravi Teja Mullapudi, Gengshan Yang, Ruta Desai, Vidya Narayanan, Achal Dave, Yuxiong Wang, Xiaolong Wang, Xinlei Chen, Jacob Walker, David Fouhey, Adhithya Murali, Lerrel Pinto, Gunnar Sigurdsson, Olga Russakovsky, Supreeth Achar, Xiaofang Wang, Chen-Hsuan Lin, Wei-Chu Maa, Namhoon Lee, Jack Valmadre, Rohit Girdhar, Ishan Misra, Haochen Wang, Chao Liu, Xiu Liu, Haroon Idrees, Mariko Isogawa, Shin Usami, Donglai Xiang, Yufei Ye, Allie Del Giorni, Pavel Tokamokov, Arun Srivatsan, Anna Henson, Dinesh Reddy, Rawal Khirodkar, Navyata Sanghvi, Harsh Agarwal, Neelesh Kulkarni, Siva Chaitanya, Vivek Roy, Krishna Kumar, Fanyi Xiao, Anirudh Vishwanathan, Priyam Parashar, De-An Huang, Dhiraj Gandhi, Alok Sharma, Jacob Huh, Gilwoo Lee, Wen Sun, Arun Venkatraman, Kumar Shaurya Shankar, Jiaji Zhou. I am also grateful to wonderful students I collaborated with – ZQ, Kangle, Kevin, Chunhui, Deepthi, Jinye, Xiao, Zixuan, Akash, Julian, Verica, Bhrij, Jerry, Yuan, and Anjali. Here is a warning to the new students at CMU – if you are looking to graduate early then do not be my successor in Smith Hall-201. Both my predecessor, Tomas Simon, and I have spent a luxurious time and were not in a hurry to graduate! Our vibes are perhaps strong at that place.

Many thanks to – Suzanne Muth, Rachel Burcin, Mary Green, Catherine Copetas, Jess Butterbaugh, Stepheny Matvey, Christine Downey, Ann Stetser for making everything smooth and seamless. I am also thankful to Byron Spice for publicising my work. I am also fortunate to work with many movie producers and journalists throughout the world in the last few years. Thanks to – Thierry Ardisson, Naomi Austin and BBC Studios, Mark Schubin, WQED, NBC, PBS, France TV, Full Frontal with Samantha Bee, and many others. I am thankful to friends and mentors outside CMU for many wonderful discussions – Kiran Bhatt, Jia-Bin Huang, Marynel Vazquez, Animesh Garg, Partha Pratim Talukdar, Anand Mishra, Michael Zollhoefer, Sergey Tulyakov, Dushyant Mehta, Ricardo Martin Brualla, Tinghui Zhou, Victor Fragaso, and Bojan Vrcelj. I am also grateful to friends both in the United States and India who helped me collect the data for this thesis – Sam Caloiero, Sofia Caloiero, Zachary Brkovich, Melanie Brkovich, Yulia Zhukoff, Liese Marjorie, Sahana Arun Kumar, Mythreyi Rajan.

Pittsburgh! Pittsburgh is home. I found a family here – my good friend, Aravindho, found it for me! My sincere gratitude and love to Susmita Ghosh, Kunal Ghosh, Santhi Priya, Hari Chandan Mantripragada, Kalpita Chakraborty, Rajib Maji, Anand Srirangam, Riya Sarkar, Sudarshan Narayanan, Sudeshna Ghosh, Namrata S, Salini Konikat, Santhosh Padala, Akshay Bhaskarwar, Shraddha Tul-

siani, Harshal Mahajan, Anjana Sarkar, Meenal Patel, Lavanya Marla, Umashankar Nagarajan. Adding to my Pittsburgh experience is the time spent at the SV Temple and Ramakrishna Vedanta Ashrama. Pittsburgh is home because you are here! You have been extremely gracious in feeding this poor soul both nutritionally and spiritually. I am also thankful to my landlords, Gwenn and Neil Musicante, for allowing me to stay at their place for the last 8 years. It would be wrong to not acknowledge my go-to wonderful restaurants, coffee/tea, and ice-cream places in the Greater Pittsburgh Area – Udipi, Tamarind, Mintt, Taj Mahal, Crepes, People's India, Eat Unique, Bengal Kabab, 61C Caffe, Dobra Tea House, Baskin and Robbins, Cold Stone, amongst many others.

Back in India, my sincere thanks and appreciation to my large-large biological family for a lot of love and sacrifices. I am indebted to my parents, Raksha Bansal and Yashpal Bansal, for instilling the importance of education while growing up and giving me ample freedom to explore my interests. It is because of them I never had to worry about any other thing in life. My maternal grandfather has been my source of inspiration – now and then, I feel like a small child in his lap. His life is a message for me, and I am trying to live his message to become a better human being without any conceit. I am thankful to my elder brother, Manish Bansal, for everything right from holding my hand when I came in to existence, to making me walk literally and figuratively. I am also grateful to my brother's wife, Dikshita Bansal, for patiently listening to me and lavishly feeding me whenever I am around her. My nephew, AikOm, has been an inspiration. A twenty days vacation with him were thought-provoking and led to the development of my recent work on evolving task representations, and strengthened my belief in unsupervised learning.

I am also fortunate to enjoy the love and support of people at my undergraduate institution. I am thankful to MPS Bhatia, Maneesha Gupta, Raj Senani, and Duru Arun Kumar. My acknowledgements would be incomplete without the mention of old friends – Dileep Sharma, Kushal Mahajan, Gaurav Sood, Devesh Satija, Harsha Bornfree, Mayank Lodha, Swarandeep Singh, Ashish Prakash, Akash Gangil, and Mahesh Sharma. While I have always missed important events in your life, you have always taken out time for me. Thank you for the selfless friendship and accepting me to your lives. Life would not be meaningful without you!

Lastly, I am fortunate to be in the company of many spiritually realized souls, and monks in the last few years. I am blessed to be introduced to the *Bhagwad Gita*. The *Upanishads* and the Indian Philosophy have shaped my thoughts. The writings of Veda Vyasa and Adi Shankaracharya have opened a completely new world for me. I pray to the almighty that dwells in every living being, and I pray that I become one with the universe!

Contents

1 Towards Lossless Communication **1**
- 1.1 Computer: The Ultimate Communication Device . . . 5
- 1.2 Learning to Observe, Explore, and Disseminate . . . 9
- 1.3 Contributions: The World is our Computational Studio . . . 13
- 1.4 Part I: Observing the 4D Visual World . . . 18
- 1.5 Part II: Example-based Exploration . . . 19
- 1.6 Part III: Human-Controllable Representations . . . 21
- 1.7 Part IV: Continual and Streaming Learning . . . 23

I Observing the 4D Visual World **25**

2 Dense 4D Visualization **26**
- 2.1 Review of 4D Capture . . . 29
- 2.2 4D Browsing Engine . . . 38
- 2.3 Practical Limitations and Design Decisions . . . 42
- 2.4 Unconstrained Multi-View Dataset . . . 44
- 2.5 Quantitative Analysis . . . 49
- 2.6 Depth Map from Multi-Views . . . 51
- 2.7 Foreground-Background Segmentation . . . 51
- 2.8 Object Tracking . . . 53
- 2.9 Revealing Hidden Components . . . 53
- 2.10 Dense 3D Reconstruction . . . 54
- 2.11 Why Multi-Views? . . . 56
- 2.12 Limitations of Proposed Representation . . . 59

II Example-based Exploration **60**

3 Unsupervised Video Retargeting **61**
- 3.1 Association and Imagination . . . 62
- 3.2 Related Work . . . 64

3.3 Recycle-GAN . . . 66
3.4 Quantitative Analysis . . . 68
3.5 Qualitative Analysis . . . 73
3.6 Limitations: Association beyond Data Distribution . . . 77

4 Unsupervised Audio-Visual Synthesis 80
4.1 Audio-Visual Synthesis . . . 81
4.2 Review of Audio-Visual Synthesis . . . 82
4.3 Exemplar Autoencoders . . . 85
4.4 Implementation Details . . . 88
4.5 Quantitative Analysis . . . 90
4.6 Verification of Reprojection Property . . . 95
4.7 Ablative Analysis . . . 95
4.8 Broader Impacts . . . 98
4.9 Forensic Study . . . 101

5 Semi-Parametric Multi-Modal Image Synthesis 103
5.1 Incomplete Input and Multiple Plausible Outputs . . . 103
5.2 Parametric and Non-Parametric Models . . . 106
5.3 PixelNN: One-to-Many Mappings . . . 107
5.4 Quantitative Analysis . . . 112
5.5 Beyond Synthesis: Understanding CNNs . . . 115
5.6 Input and Output Image Reconstructions . . . 121

III Human-Controllable Representations 126

6 Human-Controllable Image Synthesis 127
6.1 In-the-Wild Image Synthesis . . . 127
6.2 Review: Hierarchical Composition . . . 130
6.3 Shapes and Context . . . 133
6.4 Quantitative Analysis . . . 137
6.5 OpenShapes: User-Controls . . . 140

7 Human-Controllable Video Exploration 148
7.1 Video Analytics . . . 148
7.2 Videos: Opportunities and Conundrums . . . 153
7.3 Video-Specific Autoencoders . . . 155
7.4 Demonstration . . . 161

8 Human Expressable and Understandable Scene Cues 171
8.1 Scene Cues and Constraints . . . 171
8.2 Background . . . 172

8.3 PixelNet . . . 174
8.4 2D Semantic Cues . . . 182
8.5 2.5D Geometric Cues . . . 187
8.6 2.5D Boundary Cues . . . 191
8.7 3D Objects . . . 193

IV Continual and Streaming Learning **204**

9 Streaming Learning with a Few Examples **205**
9.1 Learning in Children vs. Machine . . . 205
9.2 Related Work . . . 209
9.3 Continual Evolution via Streaming Learning . . . 212
9.4 Fine-Grained Image Classification . . . 215
9.5 Pixel Analysis . . . 218
9.6 Extreme-Task Differences . . . 223
9.7 Discussion . . . 225

10 Conclusion & Future Work **226**

Chapter 1

Towards Lossless Communication

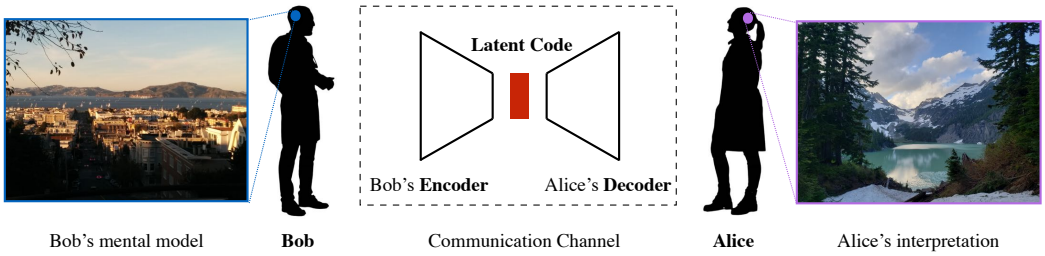


Figure 1.1: **Communication between Alice and Bob**: Bob wants to tell Alice about the place where he grew up, his experiences of spending peaceful evening at that place, the view of mountains, the sea, and beautiful clouds. We refer Bob's experiences as a mental model (shown on left). Bob has an encoder that compresses this rich mental information to a latent code or tiny bits of information consisting of words, expressions, emotions, and body-language. Alice is using her decoder to understand the tiny bits to reconstruct Bob's mental model. **This is a lossy channel of communication**. More often, the reconstruction or interpretation (shown on right) are different from the input.

Consider a dyadic conversation between Alice and Bob as shown in Figure 1.1. Bob wants to take Alice down memory lane and show her the place where he grew up – the lake, the greenery and the mountains, and make her hear the melodies of chirping birds, the dancing trees, and the rustling leaves. How would the place look like on a sunny day? How mesmerizing would it become with clouds and rain? How important were mountains and forests to the beauty of the area? And how gloomy it became when the lake once dried up due to the drought?

Lossy Communication Channel: We refer to Bob's experiences as a mental model [251]. Ideally, Bob would like to transfer this mental model to Alice as is. **Does this happen so easily in a conversation?** What really happens in a standard

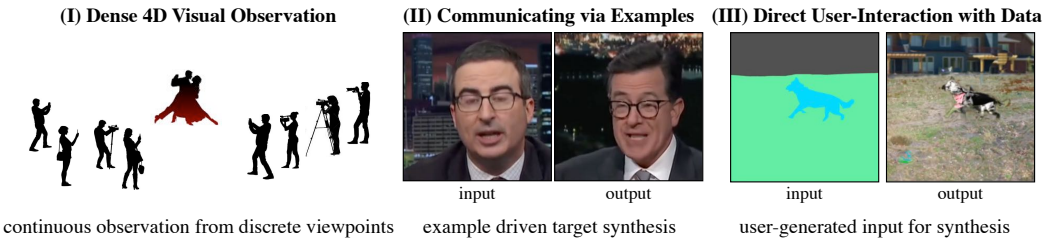


Figure 1.2: Our goal is to learn a computational representation of the 4D audio-visual world that can be: (1) estimated from sparse real-world observations; and (2) explored to create new experiences. We introduce Computational Studio that can continually observe, explore, and create a four-dimensional audio-visual world with minimal manual supervision. There are three essential components of this thesis: (I). **Dense 4D Visual Observation**. We present dense 3D reconstruction and 4D visualization of dynamic events from discrete viewpoints; (II). **Communicating via Examples**. Examples or analogies are what we use in daily life to tell our experiences. Examples allow us to both imagine and express. Shown here is an example of Stephen Colbert acting like John Oliver; (III). **Direct User-Interaction with Data**. Shown here is an example where a user creates an input using a paint brush and a drag-and-drop tool, and synthesized output using our approach.

conversation – Bob has a mental model and has an encoder that compresses this rich information to a latent code or tiny bits of information consisting of words, expressions, emotions, and body-language. Alice is trying to understand the tiny bits to reconstruct Bob's mental model and often leads to a different interpretation. This is a *lossy* channel of communication that leads to massive lose of information. *What if Bob could audio-visually communicate this mental model as he experienced it*?

Computational Studio: In this thesis, our goal is to learn a computational representation of the 4D audio-visual world that can be: (1) estimated from sparse real-world multi-view observations; and (2) explored to create new experiences. We introduce Computational Studio, a framework that can continually observe, explore, and create a four-dimensional audio-visual world with minimal manual supervision. This computational representation allows humans to communicate with other humans and machines effectively without any loss of information. Learning and communicating the 4D audio-visual world requires three things (summarized in Figure 1.2): (1) How can we densely observe the 4D visual world?; (2) How can we communicate our experiences via examples?; (3) How can we interactively create and edit the audio-visual world? These questions are inspired from how we, humans, perceive the world (i.e. inputting audio-visual data from different perspectives) and how we use our experiences to explain that data to others.

Learning and Communicating: In this thesis, we provide a general frame-

(a) a colony of honey bees (b) a herd of elephants (c) a group of dolphins

(d) a group of human beings (e) a group of lions (f) a group of birds

Figure 1.3: **Why Communication? Intelligent Beings Communicate**. Over the years, different studies have observed the communication ability of various species, be it honey bees, elephants, dolphins, lions, birds, and humans. The coordinated efforts of different species shown here could not be possible in the absence of communication. The ability to communicate enables a species to understand their world, and act for their survival and advancement. **The extent of communication within a species largely determine its ability to advance**.

work that enables a computational machinery to learn the 4D audio-visual world in an unsupervised manner from sparse unconstrained real-world samples. There is a continual interplay between learning and communicating the 4D audio-visual world in this work. Communication is not possible without learning. Similarly, the true sign of learning in a computational machinery is when it is able to communicate what it has learnt and interact effectively with other intelligent beings. *A computing machine that can continually learn directly about the world from sparse and unconstrained real-world observations, examples, and interactions with humans can enable a better human-computer communication that can further improve human-human communication*. In this thesis, we bring together insights from computer vision and graphics, machine learning, robotics, human-computer interaction, and psychology. While our work has several applications, we will restrict to the case-study of audio-visual communication for most part of this thesis.

Why Communication? Communication is the hallmark of an intelligent species and quite well-observed in biological beings (Figure 1.3), be it ants, honey bees, elephants, dolphins, cats, dogs, humans, and even plants. The ability to communicate enables a species to understand their world and act for their survival and advancement. The extent of communication within a species largely determine its ability to

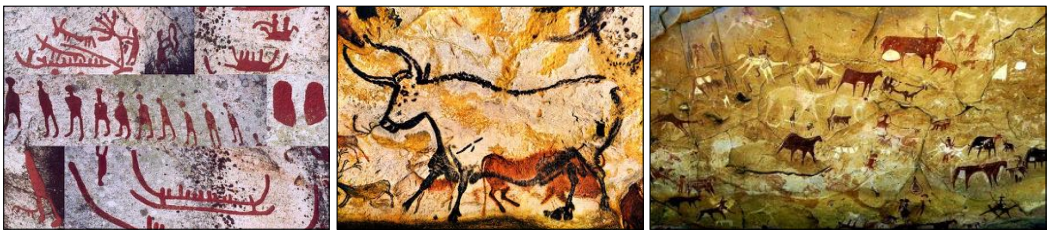
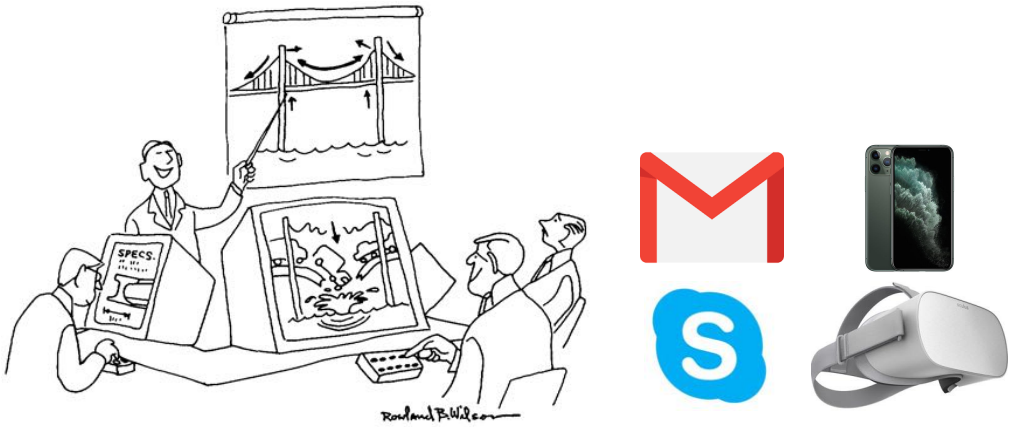


Figure 1.4: **Visual Expression for Communication in Prehistoric Era**: The earliest use of technology in communication dates back to $30,000$ years BCE (prehistoric era) when the early humans would draw on the rocks to communicate.

advance. Communication plays a vital role in human life and largely determines the quality of relationship between individuals and the society. We, humans, have a strong urge to express ourselves, share our thoughts, dreams, and imagination with others. We, however, struggle to share our rich mental imagery, our experiences, our imagination with others. *A computational machinery that understands our world can enable us to express ourselves and our imagination, and lead to better communication than a face-to-face interaction*.

Revisiting Audio-Visual Communication with Computers. In the early times, humans would draw on rocks to communicate as shown in Figure 1.4. The process of drawing was not efficient and restricted only to a selected few. This led to the development of languages over the period of time. Language acts as a codebook of a community that is known by all its member and has name for everything (objects, experiences etc). While it eased the task of communicating about the usual things of the human need, it is not a comprehensive medium to explain our thoughts as the codebooks and our understanding about them are limited. Making matters worse, there is no universal codebook known to all humans. How can we then communicate our thoughts to others without being limited in our abilities to express ourselves due to varying educational backgrounds, language barriers, varying expression and emotions, different cultural and geographical links? Our common audio-visual world that we live in connects us with minimal barriers. Our understanding of the world is fairly standard. For e.g. what we call as "water" in English is called as l'eau in French, wasser in German, paanee in Hindi, jal in Bangla, neer in Sanskrit, and tanneer in Tamil – we all know what it communicates when we see it despite different names. In this thesis, our goal is to use this universal visual language and enable an audio-visual social communication for everyone using their everyday computational devices.



(a) Licklider and Taylor, 1968 (b) Computer: Means of Communication

Figure 1.5: **Beyond Face-to-Face Communication**: (**a**) Licklider and Taylor [251] envisioned computational machinery that could enable better communication between humans than face-to-face interaction. Shown here are different experts talking about the bridge — there is a materials expert on the left, safety expert in the middle, and so on. This means given the *observation* of the bridge, different experts can *explore* and *disseminate* their thoughts. (**b**) We have so far used computing to develop various means of communication, such as mails, messaging, phone calls, video conversation, and virtual reality. These are a proxy of face-to-face communication that aims at encoding words, expressions, emotions, and body language at the source and decoding them reliably at the destination. *None of these devices allow a better communication than a face-to-face interaction*. **Image Credits**: Licklider and Taylor [251] and Rowland Wilson.

1.1 Computer: The Ultimate Communication Device

Face-to-Face interaction is the most potent form of communication. Licklider and Taylor [251] envisioned a computational machinery (with an easy human interaction [409]) can enable a better communication than what is possible with a face-to-face interaction. Figure 1.5-(a) is a sketch illustration of an inter-disciplinary meeting by Rowland Wilson highlighting this vision. Shown here are different experts talking about the bridge — there is a materials expert on the left, safety expert in the middle, and so on. Given the *observation* of the bridge, different experts can *explore* various aspects of the bridge, and can *disseminate* their findings.

(a) a mother teaching her two daughters about an aquatic animal by just pointing the computer to it.

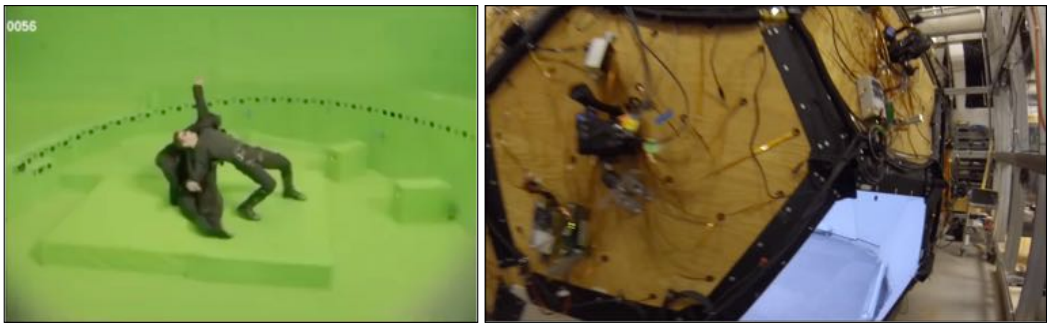
(b) a teacher teaching her third-grade students about various aspects of space-travel.

Figure 1.6: **Computer: An Information Retrieving Machine**. Alan Kay envisioned Dynabook or personal computer that can be used by anyone anywhere as a means to access information. Here are two illustrations of Alan Kay's vision by Glen Keane. **(a)** A mother is teaching her two daughters about an aquatic animal by just pointing the computer to it. **(b)** A teacher is teaching her third-grade students about various aspects of space-travel. The group of students on the right is running a simulation of a Mars landing, while the students on the left are learning about solar sails. This process of learning would not be easy if computers, cameras, Wikipedia, and Google did not exist. This access to information which would have otherwise required days or weeks is now on our finger-tips. We have largely achieved this vision of computing, and— we have a universal access to the information. **Image Credits**: Alan Kay, Bob Stein, and Glen Keane.

1.1.1 Computing for Communication

In the last 50 years, we have developed various means of communication such as mails, messaging, video conversation, or virtual reality headsets (shown in Figure 1.5-(b)). These are, however, means of communication that are trying to build a proxy of face-to-face communication by encoding words, expressions, and emotions at the source, and decoding them reliably at the destination. None of these devices allows us to have a better communication than a simple face-to-face interaction. *The real promise of computing as a means to provide lossless communication is still not realized*.

Storytellers: Computers are largely used by movie producers with ample budgets to tell their stories to everyone. We have come a long way from initial thoughts on a computing machinery [52,251] to the early development of human-machine interfaces [105,409] and availability of massive compute on a hand-held device [223]. Less than a hundred years back, showing airplane dogfight in the movie *Hell's Angels* (Howard Hughes Production) required filming authentic aerial combat scenes.



(a) Bullet-Time Scene — Matrix, 1998 **(b)** Panoptic Studio — Joo et al., 2015

Figure 1.7: **Dense 4D Observation in Studios:** The success in computer graphics is restricted to fixed studio setup that provides properly placed cameras in a controlled environment. **(a)** A popular example is the *bullet-time scene* from *Matrix, 1998* that is captured using 125 cameras placed at 5cm. **(b)** Another example is the CMU Panoptic Studio that houses more than 500 cameras for dense observation. In this work, we consider sparse (10-15 unconstrained cameras) real-world capture in unconstrained environment. There are no green screens in our work.

A fleet of more than 50 airplanes with 137 pilots was assembled for the flying sequences and to pilot the camera planes. With the advances in computing, we have moved well beyond the requirements of physical capture to content creation in the current production. We now have creatures from different realms fighting right in the middle of busy Manhattan in New York City (*Avengers*, 2012; and *Avengers Infinity War*, 2018) to blowing up the Heinz Field in Pittsburgh (*Dark Knight Rises*, 2012) to realistic animal motions with a picturesque background (*The Lion King*, 2019) to showcasing adventures centered around toys (*Toy Story 4*, 2019).

Intensive Infrastructural Requirements: The progress in computer-generated imagery (CGI) and visual effects is restricted to a few places in the world requiring specialized hardware, software, artistic, and technical skills. We have not come to a point where the technology can be used by anyone anywhere. This is primarily due intensive infrastructural requirements in a studio that (1) allows to densely capture the observations from multiple cameras (Figure 1.7); and (2) consists of a team of trained artists that are experts in exploring the audio and visual data, creating new content through a wide variety of tools (Figure 1.8). Our goal is to overcome these infrastructural challenges such that anyone can tell their stories anywhere. Importantly, we are entering into the era of virtual and augmented reality where every user should have the free-form flexibility to observe, analyse, and create the content. Social communication in natural environments is not possible unless everything happens at 90 Hz whereas it takes weeks for a team of artists to create a few seconds of a movie.

Advances and Limitations of Personal Computing: In the last fifty years, we

(a) Intensive Annotations for Supervised Learning

(b) Intensive Human and Computational Resources

Figure 1.8: **Intensive Infrastructural Requirements:** Despite the recent progress in computer vision and machine learning, the problem remains challenging: (**a**) the big revolution in machine learning is owed to **intensive manual annotations** that enabled supervised learning using convolutional neural networks. In this work, we do not have a large amount of data not because we cannot collect it but it does not exist. Additionally, even if we have the data it is not only non-trivial to label the data – it is not even clear as what are the appropriate labels to collect. We, therefore, learn on the data without any manual annotations. More importantly, we go beyond fixed datasets.(**b**) Existing work is primarily restricted to professional movie studios with **a team of trained artists** with ample resources and **intensive computational budget**. Our goal is to democratise content creation such that a non-expert can easily use it on their everyday computational devices (e.g., a mobile phone) in real-time.

have made tremendous progress in personal computers (both hardware and software). This progress has enabled each one of us with the universal access to the information. We have powerful computers [223] in our pocket. Figure 1.6 shows two illustrations of Alan Kay's vision of computers as the information retrieving machine. Shown in Figure 1.6-(a) is a mother teaching her two daughters about an aquatic animal by just pointing the computer to it. This process of learning would not be easy without the progress in computing at every level (the existence of powerful computational power, digital cameras, Wikipedia, Google etc). This access to information which would have otherwise required days or weeks is now on our finger-tips. While this vision of computing is largely achieved, we have not been

able to tap the real potential of *computing for effective communication*. We have so far used computing to develop various **means of communication**, such as mails, messaging, phone calls, video conversation, and virtual reality. These are, however, a proxy of face-to-face communication that aims at encoding words, expressions, emotions, and body language at the source and decoding them reliably at the destination. In its current form, we are primarily the consumers of a certain piece of information. We have a little control on gathering new observation, exploration, and dissemination.

1.2 Learning to Observe, Explore, and Disseminate

Communication has three crucial aspects: Observation, Exploration, and Dissemination. Observation entails extensively seeing or capturing the world such that it can be described without any loss of information. Exploration allows us to investigate and research different aspects of observation. Dissemination, finally, allows us to communicate our observation and exploration with others. In this thesis, we jointly study: (1) observation and dissemination; and (2) exploration and dissemination.

1.2.1 Observation and Dissemination

Imagine watching a tennis game between Roger Federer and Rafael Nadal at Wimbledon – seeing the game as it happened in its entirety is the *Observation*. In plain words, Observation is putting the world under a microscope and seeing every possible details. We live in a four-dimensional (3D space and 1D time) world. Capturing the unconstrained 4D audio-visual world as we see and hear is challenging because we have no large structured microscope that can precisely capture the details.

Prior work on 4D capture [205, 210, 211, 515] has primarily been restricted to studio setups with tens or even hundreds of synchronized cameras (as shown in Figure 1.7). As an example – the *bullet-time* scene in *Matrix, 1998* was created by combining the use of CGI with 120 synchronized cameras and substantial manual efforts. To this day, four hundred hours of video data is uploaded on YouTube every minute. There are more cameras at a place than there are people due to commercial success of high quality hand-held cameras (mobile phones and GoPros). Many public events are easily captured from multiple perspectives by different people (Figure 1.9). Despite this new form of visual data, we are constrained to see what was physically captured. We cannot see the entire space-time extent of the dynamic events. We cannot relive the moment. **In this thesis, we utilize arbitrary mobile phones to densely observe the 4D visual world**. Importantly, there are no constraints on the capture and the events under observation as shown in Figure 1.10. Figure 1.11 shows an illustration of dense 4D observation for a challenging scenario

Figure 1.9: **Public Events Captured from Various Perspectives:** With the onset of mobile phones, many public events are captured from various perspectives by different people. Our goal is to utilize these discrete viewpoints and enable dense 3D reconstruction and 4D visualization of dynamic events.

(more than 30 people in the scene without any constraints) captured from unconstrained viewpoints.

1.2.2 Exploration and Dissemination

When you see the sunset standing on the Seine river in Paris, you can easily imagine the same evening along the Monongahela in Pittsburgh. When you hear something, you can imagine how your partner would have said it. When you think of an event, you can predict different bits of it in your imagination. Humans have a remarkable ability to associate different concepts and think about the visual worlds far beyond what can be seen by a human eye. These capabilities include (and are not limited to) inferring the state of the unobserved, imagining the unknown, and thinking of diverse possibilities about what lies in the future. *Given an observation – how do we explore, interpret, edit, and manipulate it?*

Figure 1.10: **The world is our studio:** The onset of mobile phones and cameras have revolutionized the capture scenario. Each one of us carry a powerful camera. There are more cameras at a place than there are humans around. Many public events are captured by different people from various perspectives. The question is: *how can we utilize the data from discrete cameras and yet be able to move in the space-time of the event continuously?*

Exploration via Examples: Humans can associate different concepts effortlessly. Association and perception are tightly interlinked [27, 172]. However, learning-based approaches [242] are limited to highly supervised setups requiring extensive: (1) domain-specific design to collect data; (2) expert human supervision; or (3) manual annotations. We show two examples of state-of-the-art facial animation with extreme supervision in Figure 1.12. There exist infinitely many problems in facial animation specifically and semantic content translation in general, where it is non-trivial to either capture the data or get manual annotations. The state-of-the-art approaches require extensive: (1) domain-specific setup to collect data [263, 416, 462]; (2) expert human supervision [101, 221, 225]; or (3) manual annotations [222, 449]. Manually associating the examples from two different domains (without a pixel-to-pixel mapping) is an extreme form of manually intensive task and unheard. Synthesizing in-the-wild events require going beyond supervision and curated datasets. **In this thesis, we introduce algorithms that enables humans to explore the data using examples or a scant manual interaction**. We shows an example-based exploration in Figure 1.13 where an *example* is used to drive the *target*. Importantly, we had access to only 10 seconds video (both low-res and black-and-white) of Philippe Petain during World War II. Importantly, our approach does not require any facial information and has been demonstrated for a wide variety of domains (e.g., flowers in Figure 1.16) in this thesis. Learning from sparse and unconstrained real-world examples is also crucial in medical applications. Our work on unsupervised audio-visual synthesis enables us to synthesize natural voice for the speech-impaired.

Beyond Fixed Categories: We need approaches that can automatically understand this sheer diversity appropriating both the visual appearances and motion information from the data without any manual supervision. This is because our vi-

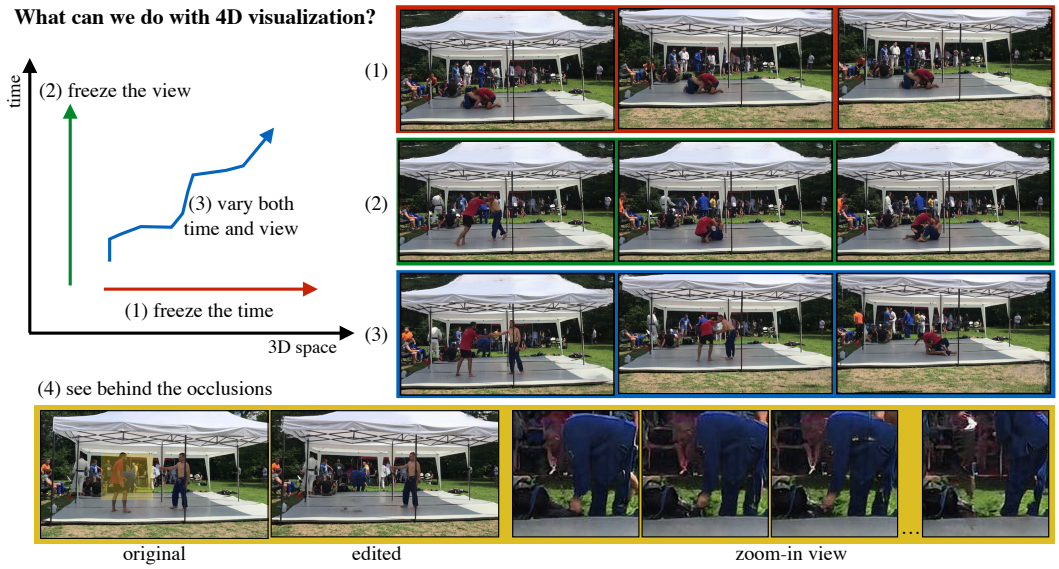


Figure 1.11: **Dense 4D Observation from Discrete Capture:** In this thesis, we present a data-driven approach for 4D space-time visualization of dynamic events from videos captured by multiple hand-held cameras. We can create virtual cameras that facilitate: (1) freezing the time and exploring views (**red**); (2) freezing a view and moving through time (**green**); (3) varying both time and view (**blue**); and (4) seeing behind the occlusions (**yellow**).

sual world is highly diverse, consisting of 8.7M known animal species (Figure 1.14), 390K known plant species, and many human-made objects. There are varied environmental, illumination, and weather conditions, and an infinite number of indescribable things such as human-human and human-object interactions. The sheer diversity of the world, and the fact that it is continually evolving seclude case-by-case approaches in understanding its richness. Physically modeling the wild possibilities in the 4D world is intractable. **In this thesis, we explore data-driven methods that are agnostic of the categories or the content in the videos**.

Human-Controllable Representations: We also provide abilities to a user to explore and analyze the data. This paves the way for a computing machinery to learn directly from scant human interactions on everyday computational devices. Showing an aspect of user-control in Figure 1.15 where a user can create images by using a simple paint-brush and a-drag-and-drop tool. This type of user-control can help a user to put their thoughts together and conceive the idea well. OpenShapes application is running on a single core CPU, enables 150 different background and object categories. It take 10 seconds to generate 5 outputs from a user-input without any system level optimization.

(a) SIREN — Epic Games **(b)** Facial Capture Studio — Facebook Reality Lab

Figure 1.12: **Extreme Supervision for Facial Animation:** We show two examples of highly-supervised setups for facial animation. **(a)** *SIREN, 2019* from Epic Games that was captured in a well constrained environment with expert supervision. We refer the reader to a brief video highlighting the amount of efforts it took to build this demonstration. **(b)** *Facial Capture Studio* at the Facebook Reality Labs, Pittsburgh – this is a multi-camera setup consisting of 40 synchronized cameras that captures 5120×3840 images at 30 frames per second.

1.3 Contributions: The World is our Computational Studio

A computational machinery that can observe, explore, and create a four-dimensional audio-visual world with minimal manual supervision can enable humans to communicate effectively with other humans and machines. This thesis is divided into four parts. We summarize each of them here:

Observing 4D Visual World: The first part of this thesis introduces the work on capturing, browsing, and reconstructing the 4D visual world from sparse real-world multi-view samples. We bring together insights from both classical image-based rendering and neural rendering approaches. Crucial to our work are two components: (1) Fusing information from sparse multi-views to create dense 3D point clouds; and (2) Fusing multi-view information to create new views. Though captured from discrete viewpoints, the proposed formulation allows us to do dense 3D reconstruction and 4D visualization of dynamic events. It also enables us to move around the space-time of the event continuously and facilitate: (1) Freezing the time and exploring 3D space; (2) Freezing the 3D space and moving through time; and (3) simultaneously changing both 3D space and time. Finally, a comprehensive control of 4D visual world allows us to do geometrically consistent content editing. We briefly discuss it in Section 1.4. There is one chapter in this part:

example

output

Figure 1.13: **Example-based Facial Animation:** We show an example of facial translation where we had access to only 10 seconds video (both low-res and black-and-white) of Marshal Philippe Petain during World War II. Our approach retargets the facial animation from example (**left-side**) to the target to create new outputs (**right-side**) in an unsupervised manner. Despite a low-res video of Petain available to us, we get a hi-res output. This approach does not consider any facial information and has been demonstrated for a wide variety of domains (e.g., flowers, birds, robots, clouds, winds, sunrise, and sunset) in this thesis.

1. We introduce a dense 4D visualization of dynamic events captured from unconstrained multi-view videos with large baselines. Most importantly, the proposed approach is agnostic of the video content and works for a large range of videos. An initial version of this work is published at CVPR 2020.

Example-based Exploration: The second part of this thesis details the example-based synthesis of the audio-visual world in an unsupervised manner. Example-based audio-visual synthesis allows us to express ourselves easily. In this part, we introduce Recycle-GAN that combines spatial and temporal information via adversarial losses for unsupervised video retargeting. This allows us to translate the contents from one domain to another while preserving the style native to the target domain. E.g., if our goal is to transfer the contents of John Oliver's speech to Stephen

Figure 1.14: **Diverse Visual Content:** Our visual world is highly diverse, consisting of 8.7M known animal species, 390K known plant species, and many human-made objects. There are varied environmental, illumination, and weather conditions, and an infinite number of indescribable things such as human-human and human-object interactions. Physically modeling the wild possibilities in the 4D world is intractable. *In this thesis, we explore data-driven methods that are agnostic of the content*.

Colbert, then the generated content/speech should be in Stephen Colbert's style. We then extend our work to audio-visual synthesis using Exemplar Autoencoders. Our approach builds on simple autoencoders that project out-of-sample data onto the distribution of the training set. We use Exemplar Autoencoders to learn the voice, stylistic prosody (emotions and ambiance), and visual appearance of a specific target exemplar speech. This work enables us to synthesize a natural voice for speech-impaired individuals and do a zero-shot multi-lingual translation. Finally, we introduce PixelNN, a semi-parametric model that enables the generation of multiple outputs from a given input and examples. We briefly discuss it in Section 1.5. There are three chapters in this part:

1. We introduce an unsupervised video retargeting that transfers the content of one identity to another while preserving the style of the target. This example-based method enables us to explore and synthesize new visual content. An initial version of this work is published at ECCV 2018. This work has also been used by various production houses and news agencies to create content for their documentaries and shows.

2. We introduce a zero-shot unsupervised audio-visual synthesis that allows anyone to input their voice and generate audio-visual stream for the target identity. This work enables us to synthesize a natural voice for speech-impaired individuals and do a zero-shot multi-lingual translation.

3. We introduce a semi-parameteric approach that enables multi-modal image

Figure 1.15: **Human-Controllable Representation for Image Synthesis:** Showing an aspect of user-control where a user can create images by using a simple paint-brush and a-drag-and-drop tool. This type of user-control can help a user to put their thoughts together and conceive the idea well. OpenShapes application is running on a single core CPU, enables 150 different background and object categories. It take 10 seconds to generate 5 outputs from a user-input without any system level optimization.

synthesis without requiring to train a new parametric model. The crucial aspect of this work is that it allows a user to create a desired output by providing an example to an off-the-shelf model. This work is published at ICLR 2018. An extension of this work also shows application in understanding convolutional neural networks.

Human-Controllable Representations: The third part of this thesis introduces human-controllable representations that allows a human-user to interact with visual data and create new experiences on everyday computational devices. Firstly, we introduce OpenShapes that allows a user to interactively synthesize new images using a paint-brush and a drag-and-drop tool. OpenShapes runs on a single-core CPU to generate multiple images from a user-generated label map. We then present simple video-specific autoencoders that enables human-controllable video exploration. This exploration includes a wide variety of analytic tasks such as (but not limited to) spatial and temporal super-resolution, object removal, video textures, average video exploration, video tapestries, and correspondence estimation within

and across videos. Prior work has independently looked at each of these problems and proposed different formulations. In this work, we observe that a simple autoencoder trained (from scratch) on multiple frames of a specific video enables one to perform a large variety of video processing and editing tasks without even optimizing for a single task. Finally, we present a framework that enables us to extract a wide range of low-mid-high level semantic and geometric scene cues that could be understood and expressed by both humans and machines. We briefly describe it in Section 1.6. There are three chapters:

1. We introduce a data-driven approach for interactively synthesizing in-the-wild images from semantic label maps. This work is published at CVPR 2019.

2. We present simple video-specific autoencoders that enables human-controllable video exploration. This includes a wide variety of analytic tasks such as (but not limited to) spatial and temporal super-resolution, object removal, video textures, average video exploration, and correspondence estimation within and across videos.

3. We explore various human understandable and expressable scene cues such as 2D semantic labels, 2.5D surface normal, 2.5D boundary cues, and 3D objects. An initial version of this work is published at CVPR 2016 and an extensive evaluation is done in an extended technical report.

Continual and Streaming Learning: The idea of exemplar and test-time training is pursued throughout this thesis. This combination enables continual learning in a streaming manner. In the last part of this thesis, we extend our work on continual learning of the audio-visual world in a streaming manner to learning visual-recognition tasks from sparse examples. We briefly discuss it in Section 1.7. There is one chapter in this part:

1. We explore semi-supervised learning of deep representations given a few labeled examples of a task and a (potentially) infinite stream of unlabeled examples. Our approach continually evolves task-specific representations by constructing a schedule of learning updates that iterates between pre-training on novel segments of the stream and fine-tuning on a small and fixed labeled dataset. We demonstrate our approach on tasks ranging from fine-grained image classification, medical-image classification, satellite-image classification, to pixel-level analysis. Without any domain knowledge, we improve the performance of these tasks.

Contrary to popular approaches in semi-supervised learning that use massive computing resources for storing and processing data, streaming learning requires modest computational infrastructure since it naturally breaks up massive datasets into slices that are manageable for processing. From this perspective, streaming learning can help democratize research and development for scalable, lifelong ML.

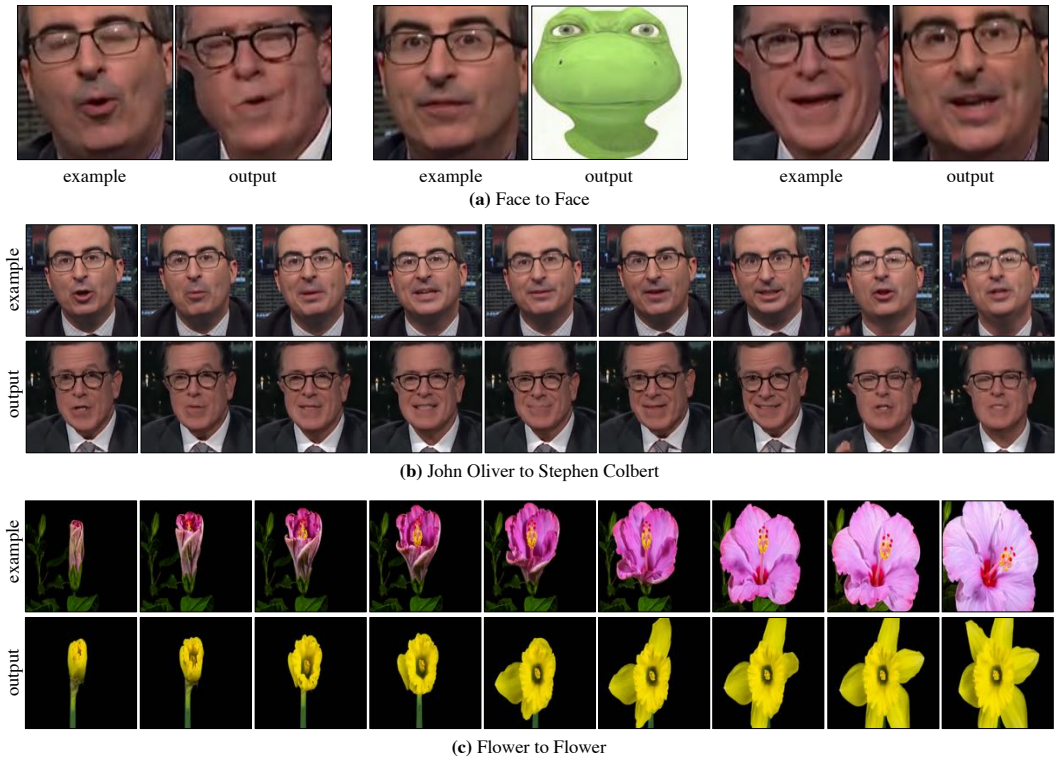
1.4 Part I: Observing the 4D Visual World

Imagine going back in time and revisiting crucial moments of your lives, such as your wedding ceremony, your graduation ceremony, or the first birthday of your child, immersively from any viewpoint. The prospect of building such a *virtual time machine* [352] has become increasingly realizable with the advent of affordable and high-quality smartphone cameras producing extensive collections of social video data. Unfortunately, people do not benefit from this broader set of captures of their social events. When looking back, we are likely to only look at one video or two when potentially hundreds might have been captured from different sources. We present two approaches that leverages all perspectives to enable a more complete exploration of the event:

Dense 4D Visualization: We present a data-driven approach for 4D space-time visualization of dynamic events from videos captured by hand-held multiple cameras. We use self-supervised and scene-specific neural networks to compose static and dynamic aspects of an event. Though captured from discrete viewpoints, this model enables us to move around the space-time of the event continuously. This model allows us to create virtual cameras that facilitate: (1) Freezing the time and exploring 3D space; (2) Freezing the 3D space and moving through time; and (3) Simultaneously changing both time and 3D space. Our formulation also allows us to get dense depth map and a foreground-background segmentation which in turn allows us to track objects in a video. These properties enable us to edit the videos and reveal occluded objects for a given view provided it is visible in any of the other views. Figure 1.11 shows dense 4D visualization of a dynamic event captured from sparse and unconstrained multiple views.

We validate our approach on a wide variety of in-the-wild events (humans and birds) captured using up to 15 mobile cameras. With our approach, the benefits from each extra perspective that is captured leads to a more complete experience. We seek to automatically organize the disparate visual data into a comprehensive four-dimensional environment (3D space and time). The complete control of spatiotemporal aspects not only enables us to see a dynamic event from any perspective but also allows geometrically consistent content editing. Figure 1.11 shows examples of virtual camera views synthesized using our approach for a JiuJitsu event captured from multiple hand-held cameras.

The ability to reconstruct and edit dynamic events captured from free-viewpoint videos, without a strict requirement of sub-frame alignment, opens up avenues for various applications. It enables us to capture dynamic events with drones, robots, hand-held cameras, and yet be able to do the things that require a professional studio. We can achieve an active and denser performance capture system by combining drones, small ground robots, and insect robots with hand-held mobile devices. This setup allows us to carefully visualize the details of human motion in performance, e.g., flowing clothes, hairs, hands, and finger movements. Imagine watching Super



(a) Face to Face

(b) John Oliver to Stephen Colbert

(c) Flower to Flower

Figure 1.16: **Unsupervised Video Retargeting:** We show Recycle-GAN used for facial and flower retargeting. (**a**) We show varied examples of face to face translation using our approach. (**b**) We then show translation from John Oliver to Stephen Colbert. (**c**) Finally, the bottom row shows how a synthesized flower follows the blooming process with the input flower.

Bowl [418] or a Wimbledon game in this new virtualized reality. One does not need to see the event only from stands; they can also stand next to players or even next to the ball to view it. Due to the flexibility of this system, it enables us to capture even the tangible and intangible cultural heritage of endangered tribes and wildlife in their natural habitats.

1.5 Part II: Example-based Exploration

Examples or analogies allows us to express ourselves easily [373]. Humans can associate different examples/concepts effortlessly [27,172]. We present unsupervised approaches that leverages examples to explore and create new visual content.

Unsupervised Video Retargeting : Video retargeting refers to the transfer of sequential content from one domain to another while preserving the style of the target

domain. We introduce a data-driven approach for unsupervised video retargeting that translates content from one domain to another while preserving the style native to a domain. E.g., if we were to transfer the contents of John Oliver's speech to Stephen Colbert, then the generated content/speech should be in Stephen Colbert's style. Our approach combines both spatial and temporal information along with adversarial losses for content translation and style preservation. In this work, we first study the advantages of using spatiotemporal constraints over spatial constraints for effective retargeting. We then demonstrate the proposed approach for the problems where information in both space and time matters, such as face-to-face translation, flower-to-flower, wind and cloud synthesis, sunrise and sunset. We show examples of facial and flower retargeting in Figure 1.16.

Audio-Visual Synthesis: Video translation without audio restricts the creation of realistic content. We present an unsupervised approach that converts the input speech of any individual into audiovisual streams of potentially-infinitely many output speakers. Our approach builds on simple autoencoders that project out-of-sample data onto the distribution of the training set. We use exemplar autoencoders to learn the voice, stylistic prosody (emotions and ambiance), and visual appearance of a specific target exemplar speech. In contrast to existing methods, the proposed approach can be easily extended to an arbitrarily large number of speakers and styles using only 3 minutes of target audio-video data, without requiring *any* training data for the input speaker. To the best of our knowledge, we are the first work to demonstrate audiovisual synthesis from an audio signal. To do so, we learn audiovisual bottleneck representations that capture the structured linguistic content of speech.

Multi-Modal Image Synthesis: We present a semi-parametric approach that synthesizes high-frequency photorealistic images from an "incomplete" signal such as a low-resolution image, a surface normal map, or edges. Current state-of-the-art deep generative models designed for such conditional image synthesis lack two important things: (1) they are unable to generate a large set of diverse outputs, due to the mode collapse problem. (2) they are not interpretable, making it difficult to control the synthesized output. We demonstrate that NN approaches potentially address such limitations, but suffer in accuracy on small datasets. We design a simple pipeline that combines the best of both worlds: the first stage uses a convolutional neural network (CNN) to maps the input to a (overly-smoothed) image, and the second stage uses a pixel-wise nearest neighbor method to map the smoothed output to multiple high-quality, high-frequency outputs in a controllable manner. We demonstrate our approach for various input modalities, and for various domains ranging from human faces to cats-and-dogs to shoes and handbags.

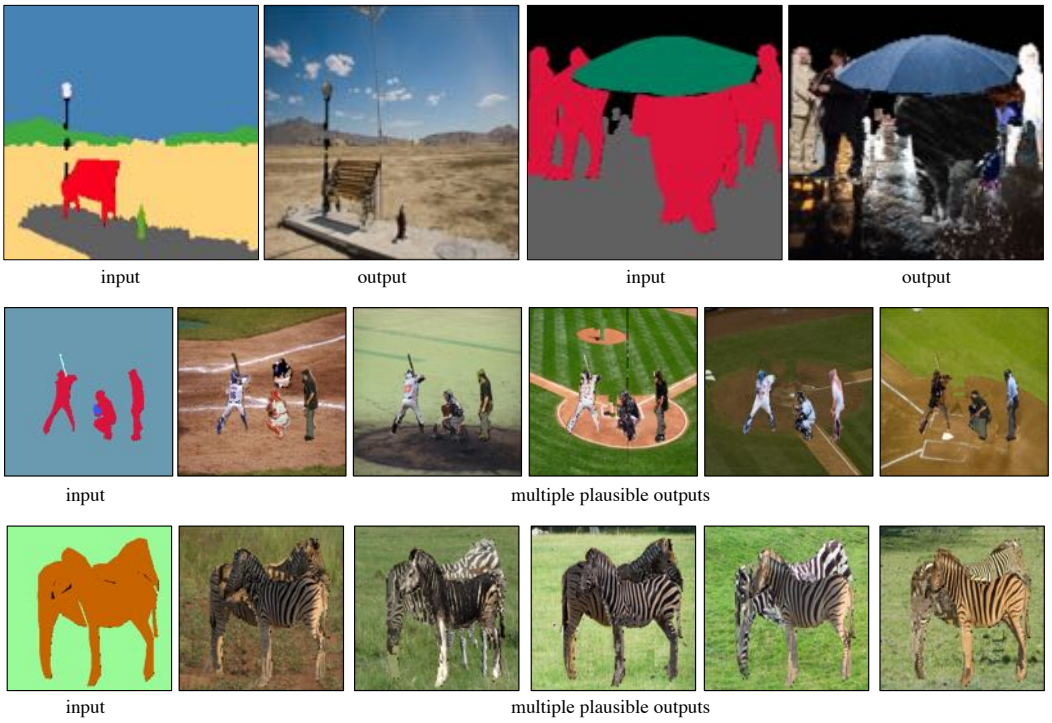


Figure 1.17: **Interactive Image Synthesis:** Our approach synthesizes images from label maps by **non-parametric matching** of shapes, parts, and pixels. We show example results for diverse "in-the-wild" scenes containing large amounts of variation in object composition and deformation. We also show multiple plausible outputs generated from a label map using our approach.

1.6 Part III: Human-Controllable Representations

Interactive visual synthesis allows us to manually create and edit visual experiences. It is important to think about a human user and computational devices when designing content creation applications.

Interactive Image Synthesis: We introduce a data-driven approach for interactively synthesizing in-the-wild images from semantic label maps. Our approach is dramatically different from recent work in this space, in that we make use of no learning. Instead, our approach uses simple but classic tools for matching scene context, shapes, and parts to a stored library of exemplars. Though simple, this approach has several notable advantages over recent work: (1) because nothing is learned, it is not limited to specific training data distributions (such as cityscapes, facades, or faces); (2) it can synthesize arbitrarily high-resolution images, limited only by the resolution of the exemplar library; (3) by appropriately composing shapes and parts, it can generate an exponentially large set of viable candidate out-

put images (that can say, be interactively searched by a user). We present results on the diverse COCO dataset, significantly outperforming learning-based approaches on standard image synthesis metrics. Finally, we explore user-interaction and user-controllability, demonstrating that our system can be used as a platform for user-driven content creation. Figure 1.17 shows images synthesized using OpenShapes, where the input is a label map. OpenShapes enables us to synthesize images from a semantic label map without requiring extensive computational power. It runs on a single-core CPU and generates 5 outputs of 512×512 resolution for a given input in 10 seconds. More importantly, any number of new categories can be added to our system in no time.

Interactive Video Exploration: We present simple video-specific autoencoders that enables human-controllable video exploration. This includes a wide variety of analytic tasks such as (but not limited to) spatial and temporal super-resolution, object removal, video textures, average video exploration, and correspondence within and across videos. Prior work has independently looked at each of these problems and proposed different formulations. In this work, we observe that a simple autoencoder trained (from scratch) on multiple frames of a specific video enables one to perform a large variety of video processing and editing tasks. Our tasks are enabled by two key observations. (1) Latent codes learned by the autoencoder capture spatial and temporal properties of that video and (2) autoencoders can project out-of-sample inputs onto the video-specific manifold. For e.g. (1) interpolating latent codes enables temporal super-resolution and user-controllable video textures; (2) manifold reprojection enables spatial super-resolution, object removal, and denoising without training for any of the tasks. Importantly, a two dimensional visualization of latent codes via principal component analysis acts as a tool for users to both visualize and intuitively control video edits.

Human Understandable and Expressable Scene Cues: We explore various human understandable and expressable scene cues. Firstly, we design principles for general pixel-level prediction problems, from low-level edge detection to mid-level surface normal estimation to high-level semantic segmentation. Convolutional predictors, such as the fully-convolutional network (FCN), have achieved remarkable success by exploiting the spatial redundancy of neighboring pixels through convolutional processing. Though computationally efficient, we point out that such approaches are not statistically efficient during learning precisely because spatial redundancy limits the information learned from neighboring pixels. We demonstrate that stratified sampling of pixels allows one to (1) add diversity during batch updates, speeding up learning; (2) explore complex nonlinear predictors, improving accuracy; and (3) efficiently train state-of-the-art models tabula rasa (i.e., "from scratch") for diverse pixel-labeling tasks. Our single architecture produces state-of-the-art results for semantic segmentation on PASCAL-Context dataset, surface normal estimation on NYUDv2 depth dataset, and edge detection on BSDS. Finally, we introduce an approach that leverages surface normal predictions, along with appearance cues, to

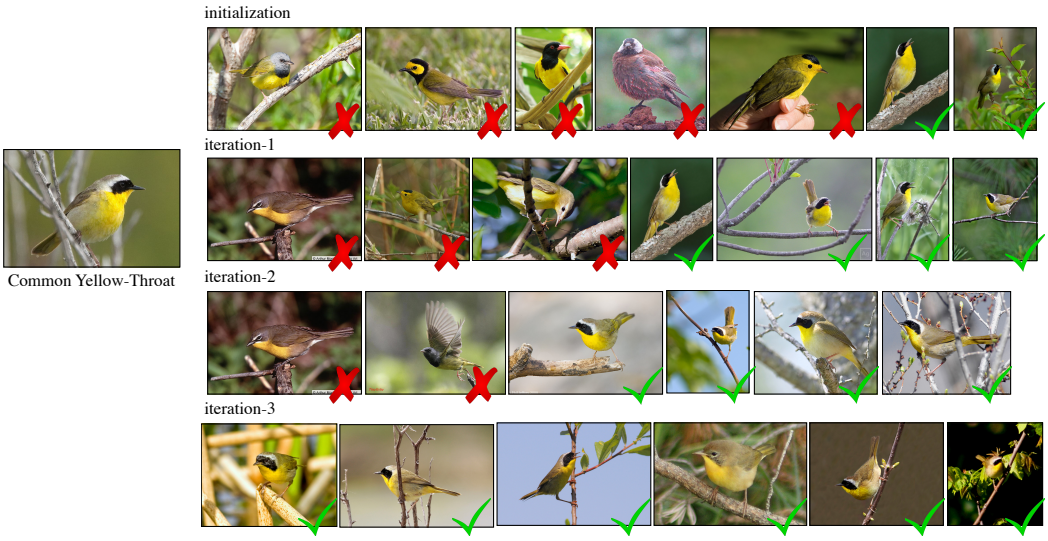


Figure 1.18: **Continual Improvement in Recognizing a Yellow Throat:** We qualitatively show improvement in recognizing a common yellow-throat (shown in left from CUB-200 dataset [463]). At **initialization**, the trained model confuses common yellow-throat with hooded oriole, hooded warbler, wilson rbler, yellow-breasted chat, and other similar looking birds. We get rid of false-positives with every **iteration**. At the the end of the third iteration, there are no more false-positives.

retrieve 3D models for objects depicted in 2D still images from a large CAD object library. We develop a two-stream network over the input image and predicted surface normals that jointly learns pose and style for CAD model retrieval. When using the predicted surface normals, our two-stream network matches prior work using surface normals computed from RGB-D images on the task of pose prediction, and achieves state of the art when using RGB-D input. Our two-stream network allows us to retrieve CAD models that better match the style and pose of a depicted object compared with baseline approaches.

1.7 Part IV: Continual and Streaming Learning

The different formulation proposed in this thesis follows exemplar and test-time training. This unique combination allows to continually learn the audio-visual world in a streaming manner. We extend our work on continual learning of the audio-visual world to learning exemplar visual concepts and visual-recognition tasks.

Streaming Learning with a Few Examples: We explore semi-supervised learning of deep representations given a few labeled examples of a task and a (potentially) infinite stream of unlabeled examples. In this setting, classic approaches that attempt to pseudo-label the entire unlabeled stream may take an exorbitant

amount of time. Our approach continually evolves task-specific representations by constructing a schedule of learning updates that iterates between pre-training on novel segments of the stream and fine-tuning on the small, fixed labeled dataset. Our approach learns progressively more accurate pseudo-labels as the stream is processed by growing the model capacity via larger backbone network architectures (that are pre-trained on successively larger segments in previous iterations). The performance of our continually-evolving models has improved by 27% top-1 accuracy on Flowers-102, and 22% top-1 accuracy on Birds-200 in three iterations. Our insights are also applicable to diverse domains including medical, satellite, and agricultural imagery, where there does not exist a large amount of labeled or unlabeled data. Finally, we also apply our insights to pixel-level problems, specifically surface-normal estimation and semantic segmentation. We improve both surface normal estimation on NYU-v2 depth dataset and semantic segmentation on PASCAL VOC-2012 by $3-7\%$. All these models are trained from scratch without any task-specific tuning or auxiliary knowledge. Figure 1.18 demonstrates the continuous improvement in recognizing a common yellow-throat using unlabeled data.

Part I

Observing the 4D Visual World

Chapter 2

Dense 4D Visualization

Figure 2.1: **The world is our studio:** The onset of mobile phones and cameras have revolutionized the capture scenario. Each one of us carry a powerful camera. There are more cameras at a place than there are humans around. Many public events are captured by different people from various perspectives. The question is: *how can we utilize the data from discrete cameras and yet be able to move in the space-time of the event continuously?*

Imagine going back in time and revisiting crucial moments of your lives, such as your wedding ceremony, your graduation ceremony, or the first birthday of your child, immersively from any viewpoint. The prospect of building such a *virtual time machine* [352] has become increasingly realizable with the advent of affordable and high-quality smartphone cameras producing extensive collections of social video data. Unfortunately, people do not benefit from this broader set of captures of their social events (as shown in Figure 2.1). When looking back, we are likely to only look at one video or two when potentially hundreds might have been captured from different sources. We present a data-driven approach that leverages all perspectives to enable a more complete exploration of the event. With our approach, the benefits from each perspective leads to a more complete experience. We seek to automatically organize the disparate visual data into a comprehensive four-dimensional environment (3D space and time). The complete control of spatiotemporal aspects

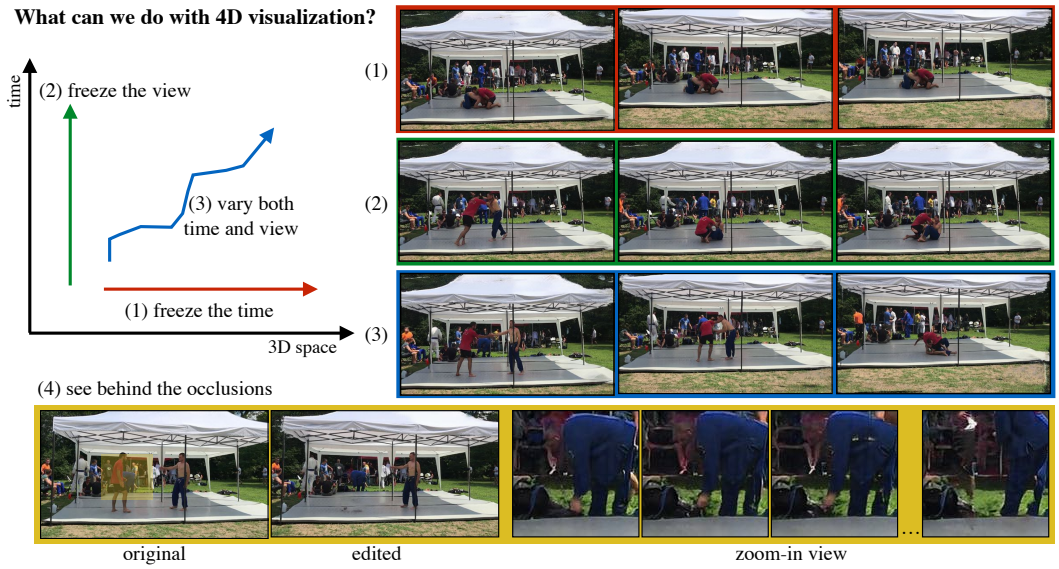


Figure 2.2: **4D Continuous Observation from Discrete Multi-Views:** In this work, we present a data-driven approach for 4D space-time visualization of dynamic events captured by multiple hand-held cameras. We can create virtual cameras that facilitate: (1) Freezing the time and exploring 3D space (**red**); (2) Freezing the 3D space and moving through time (**green**); (3) varying both time and 3D space (**blue**); and (4) seeing behind the occlusions (**yellow**).

allow geometrically consistent content editing. Figure 2.2 shows an example of different virtual camera-views synthesized using our approach for an event captured from multiple views.

Prior work on 4D capture [205,210,211,515] of dynamic events use multi-camera setup in a studio that consists of tens or even hundreds of synchronized cameras and a well-constrained environment. Another body of work use light-field capture devices [2, 148, 247, 309, 451] or small-baseline multi-camera rig [50, 120] for capturing the dynamics. Closely related to our setup is the work [19, 58, 414] that use in-the-wild capture devices for video-based rendering. These approaches are, however, restricted to the content of the videos such as a single human-actor. Recent work [90,91,442,443] use in-the-wild multi-camera systems but sparsely reconstruct the world. In this work, we use an unconstrained and sparse real-world capture of dynamic events: (1) with real-world challenges: no green-screens for background subtraction; (2) arbitrary camera baselines; and (3) no restriction on the captured contents. The required scenario has also been explored by work on 3D reconstruction from internet images [3, 176, 371, 372, 400]. These approaches are primarily restricted to static scenes and ignore the dynamics (showing 3D reconstruction using COLMAP [372] in Figure 2.3-(a)). Recently, Vo et al. [444] combined 3D reconstruc-

(a) COLMAP

(b) COLMAP + Human Modeling

(c) Ours

Figure 2.3: **Comparison to 3D Reconstruction and Human Modeling:** Given a dynamic event captured using 10 phones, we **freeze the time and explore 3D space**. **(a)** We show the outputs of **COLMAP** [371, 372] in the **top-row**. COLMAP treats dynamic information as outliers for 3D reconstruction. **(b)** Vo et al. [444] combine 3D reconstruction [372] and human-modeling [269]. We show the outputs in the **middle-row** and call this **COLMAP+Human Modeling**. These outputs lack *realism*. Additionally, the reconstruction fails for non-Lambertian surfaces (see glass windows), non-textured regions (see umbrellas), and shadows (around humans). **(c)** Our approach (**bottom-row**) can densely synthesize the various static and dynamic components in the scene and looks more realistic.

tion [372] and human-modeling [269] as shown in Figure 2.3-(b). This approach is restricted to specific scenarios where both 3D reconstruction and human-modeling can work. We make no such assumptions and our method achieves a dense 4D visualization of dynamic events. Figure 2.3-(c) contrasts our approach with COLMAP and human-modeling outputs.

In this work, we bring together insights from a large literature on image-based rendering [392] and neural rendering [417]. We pose the problem of 4D visualization from in-the-wild captures within an image-based rendering paradigm utilizing large capacity parametric models. There are three essential components of our approach: (1) Estimating time-specific aspects (instantaneous information)in a

scene for a given 3D space (or camera pose) and a time-instant; (2) Estimating time-averaged (or stationary information) in a scene averaged for a given camera pose; and (3) Fusing the time-specific (instantaneous) and time-averaged (stationary) information via a scene-specific convolutional neural network (CNNs). The time-averaged accumulation helps us to capture the large non-textured regions (e.g., umbrellas in Figure 2.3-(c)) and non-Lambertian surfaces (glass windows in Figure 2.3-(c)). Scene-specific CNNs implicitly fuse the static and dynamic scene components in a self-supervised manner. This data-driven model enables us to extract the nuances and details in a dynamic event. We demonstrate our formulation on various in-the-wild dynamic events captured from multiple mobile phone cameras. These multiple views have arbitrary baselines and unconstrained camera poses.

Explicitly modeling the time-specific (instantaneous) and time-averaged (stationary) components in a scene also allows us to get dense depth information and the foreground-background segmentation. The access to depth map and foreground-background segmentation enables tracking of dynamic objects in the scene. This provides a complete control of static and dynamic components of a scene, and enables user-driven content editing in the videos. In public events, one often encounters random movement obstructing the cameras to capture an event. Traditionally nothing can be done about such spurious content in captured data. The complete 4D control in our system enables a user to remove unwanted occluders and obtain a clearer view of the actual event using multi-view information (provided it is visible in one of the views).

Contributions: (1) We study different multi-view view synthesis approaches (using both geometric modeling and image-based rendering) proposed in the literature via unconstrained multi-view sequences captured from real-world scenarios; (2) We present a simple solution by bringing together insights from various image-based and neural rendering approaches; and (3) We demonstrate applications in user-controlled editing.

2.1 Review of 4D Capture

There is a long history of 4D capture systems [211] to experience immersive virtualized reality [130], especially being able to see from any viewpoint that a viewer wants irrespective of the physical capture systems. Crucially important is a long line of work on 3D geometry [164] and multi-view view synthesis [412] using image-based rendering [392] and neural-rendering [417] approaches. Inspired from Shum and Kang [392], we position different approaches (in Figure 2.4) using an axis of geometry or scene structure – from heavy utilization to no utilization of geometry.

4D Capture in Studios: The ability to capture depth from a small baseline stereo pair via 3D geometry techniques [164] led to the development of video-rate stereo machines [210] mounting six cameras with small baselines. This ability to capture dense depth maps motivated a generation of researchers to develop close stu-

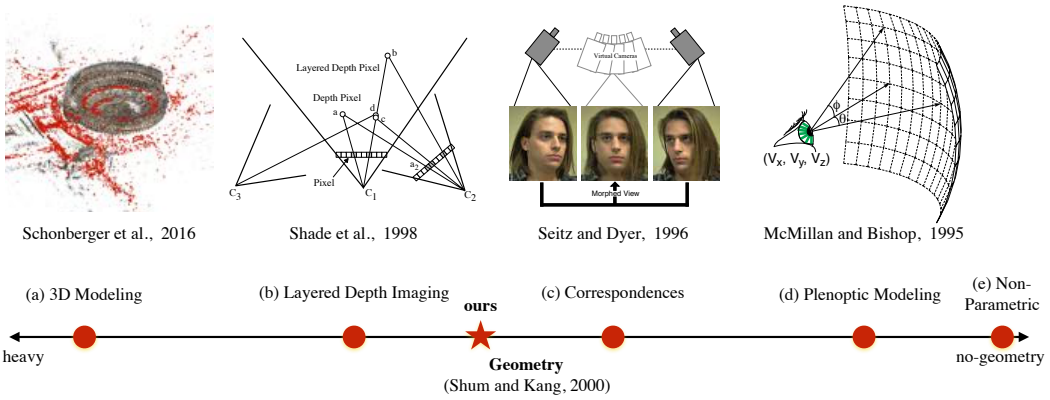


Figure 2.4: **3D Reconstruction and Multi-View View Synthesis:** We study a wide variety of work that has been proposed in the literature for related problems. We use the axis of geometry (following Shum and Kang [392]) to position these approaches. (**a**) On the left side are the approaches based on **3D modeling** [3,371,372,400] that heavily relies on geometry. We observe that traditional 3D modeling does not yield dense and high-fidelity outputs for sparse multi-views. We move from extreme geometric modeling to image-based rendering. Shum and Kang [392] categorize them in three categories: (**b**) **Using geometry by modeling depth cues in the image:** Given depth information for every pixel, we can synthesize new nearby viewpoints by 3D warping techniques [286]. Shade et al. [380] proposed layered depth imaging (LDI) to store not only what is visible in the input image, but also what is behind the visible surfaces to deal with disocclusion artifacts. The idea of layered depth imaging has been revisited recently in multi-plane imaging [293,506]. These approaches assume perfect multi-plane images (MPIs) for composition. However, MPIs for sparse views are not accurate and therefore composition leads to blurry results; (**c**) **Using geometry by establishing pixel-correspondences:** Instead of explicitly modeling the depth, pixel-correspondences between views can also be used for warping pixels to generate new views. This direction has been mostly demonstrated for a two-view or a stereo pair. In this work, we use the insights from this direction of work and establish correspondences between multiple-stereo pairs [482]. The crucial question is *how do we fuse various projections to the target camera view from multiple stereo pairs*? In this work, we use the insights from Shade et al. [380] to synthesize a new view from multiple stereo pairs. Similar to LDI, each pixel in the input image contains a list of depth and color values where the ray from a pixel location intersects with the environment. We position **our approach** in between implicit and explicit geometry; (**d**) Towards the right of correspondences is **Plenoptic Modeling** that describes all of image information from a particular viewing angle. The goal, here, is to capture every possible light ray in the scene and synthesize a new view by composing information from different light rays. Recent approaches following NeRF [294] have revisited this idea. In this work, we observe that NeRF is currently restricted to dense multi-views and good camera parameters. NeRF leads to noisy outputs for sparse camera views; and (**e**) finally, on the right extreme is a **Non-Parametric** method that finds the closest camera view. Given an extremely dense views, we can easily find a nearest camera view to the target. It is, however, different in a sparse scenario.

Figure 2.5: **3D reconstruction from sparse views for a given time-instant:** We study **3D modeling** using COLMAP [371,372] . We observe COLMAP yields sparse outputs when using 13 frames from a given instance of time. On the contrary, it removes the dynamic components when using all the frames.

3D reconstruction + Human modeling

Figure 2.6: **3D reconstruction and Human modeling:** We combine **3D modeling** using COLMAP [371, 372] and human modeling [269, 442]. We use all the frames from videos for a static background reconstruction. Despite this, we get sparse background, missing information for the foreground, and have to limit ourselves to certain types of videos (specifically human events). We use this approach as a baseline method for contrasting our approach as it works best amongst all-known approaches.

(a) Layered Depth: **LLFF**

(b) Multi-View Geometry: **Correspondences + naive fusion**

(c) Plenoptic Modeling: **NeRF**

(d) Non-Parametric: **nearest camera view**

Figure 2.7: **Image-based Rendering (part 1 of 2):** We study the problem using popular image-based rendering methodologies [392] using 12 views to synthesize a new view: **(a) Layered Depth:** We studied **LLFF** [293] that use multi-plane imaging [506] and extended the idea of layered depth imaging [380]. These approaches [293, 506] assume perfect multi-plane imagining for composition. We observe multi-plane imaging for sparse views are not accurate and leads to blurry results; **(b) Correspondences:** We estimated disparity using Yang et al. [482] for multiple stereo pairs. We project the pixels from the known views to the target view using the correspondences. Since we have many stereo pairs, this results in multiple projections of the target camera view from various stereo pairs. We naively fuse the information by taking a per-pixel median from multiple target projections; **(c)** We studied **NeRF** [294] that builds on **Plenoptic Modeling**. NeRF is currently restricted to dense multi-views with good camera parameters. This is not available in our setup and leads to noisy outputs. While (a) and (b) are off-the-shelf approaches, currently NeRF requires 2-3 days to train a model using multi-views for a specific time instant; **(d)** We show the **nearest camera view** (non-parametric) to the target camera view; contd. in Figure 2.8

(e) **Ours** (time-specific component) (f) Original Image

Figure 2.8: **Image-based Rendering (part 2 of 2):** ... contd. from Figure 2.7... **(e)** We also show the output of the first step (estimating time-specific component) of **our approach** that use a multi-view visibility constraint to fuse multiple projections to generate the target view; and **(f)** finally, we contrast with the original image.

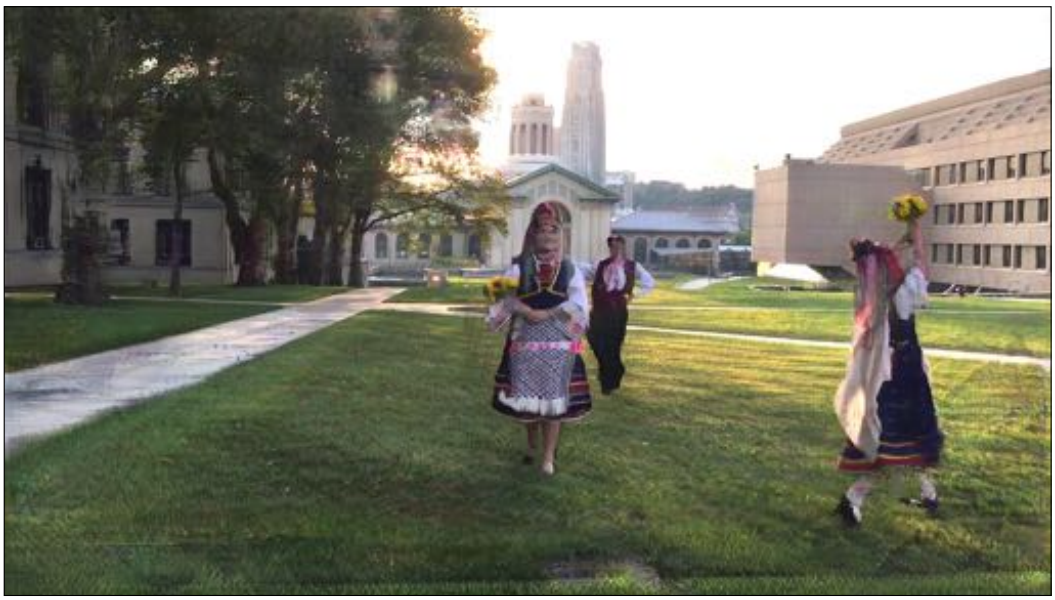


Figure 2.9: **Our approach:** We show a randomly sampled frame for a virtual camera view synthesized using our full pipeline. Our approach is able to capture foreground and background details that is missing in other approaches.

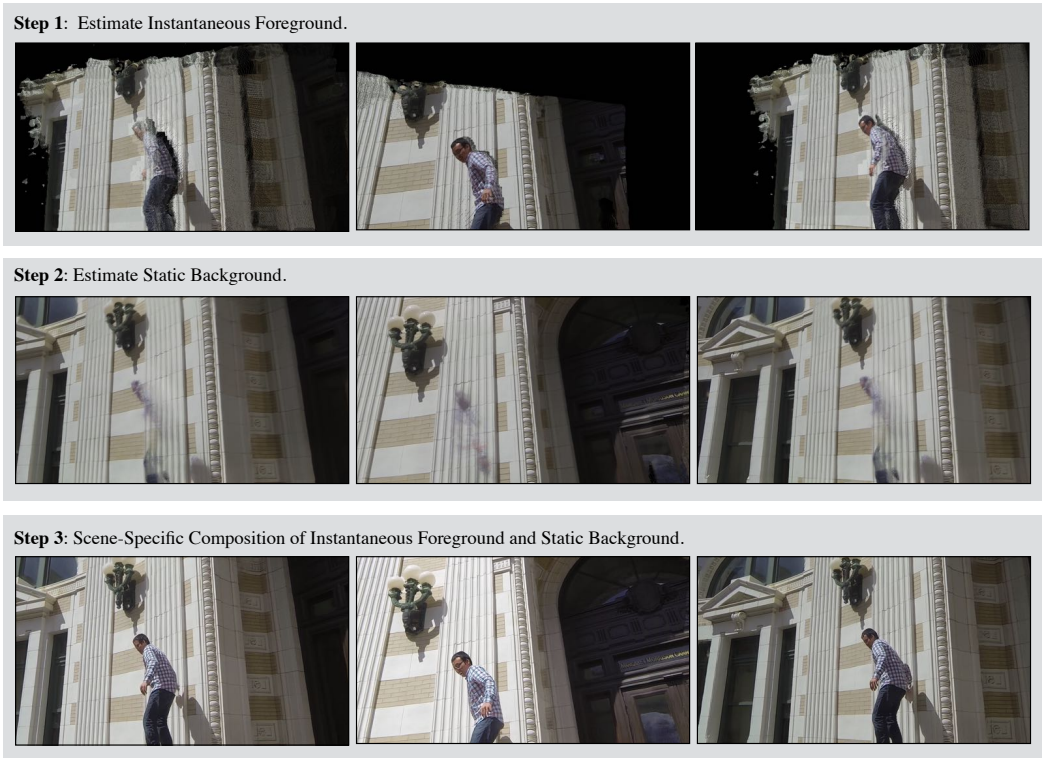


Figure 2.10: **Overview of our formulation:** We pose the problem of 4D visualization of dynamic events captured from multiple cameras as a scene-specific composition of instantaneous foreground (**top**) and static background (**middle**) to generate the final output (**bottom**). The scene-specific composition enables us to capture certain aspects that may otherwise be missing in the inputs, e.g., parts of the human body are missing in the first column and parts of background are missing in first row.

dios [205, 212, 322, 515] that can precisely capture the dynamic events happening within it. A crucial requirement in these studios is the use of synchronized video cameras [212]. This line of research is restricted to a few places in the world with access to proper studios and camera systems.

Beyond Studios and 3D reconstruction: The access to high-quality cameras and large amount of visual data motivated researchers to use in-the-wild data for 3D reconstruction [175, 371, 372, 400] and 4D visualization [19, 58]. Photo tourism [400] and the following work [3, 131, 132, 176, 397] use internet-scale images to reconstruct architectural sites. A hybrid of geometry [164] and image-based rendering [392] approaches have been used to reconstruct 3D scenes from pictures [84, 85]. These approaches have led to the development of immersive 3D browsing. However, the work on 3D reconstruction requires large amount of data for dense 3D reconstruc-

tion. In the absence of dense input views, it leads to sparse outputs (as shown in Figure 2.5). Importantly, 3D reconstruction treats dynamic information as outliers and reconstructs the static components alone. We position this body of work towards the left-side in Figure 2.4.

Capturing Dynamics: Additional cues such as visual hulls [126, 155, 284], or 3D body scans [58, 82], or combination of both [20, 441] are used to capture dynamic aspects (esp. human performances) from multi-view videos. Hasler et al. [165] use markerless method by combining pose estimation and segmentation. Vedula et al. [438] compute scene shape and scene flow for 4D modeling. Ballan et al. [19] model foreground subjects as video-sprites on billboards. However, these methods assume a single actor in multi-view videos. Recent approaches [90, 442] are not restricted by this assumption but does sparse reconstruction. Vo et al. [444] combined 3D reconstruction of static components and human-modeling for dynamics assuming human-only events (Figure 2.6).

View-Synthesis: A new view can be synthesized from a single image [312, 385, 431, 432, 466, 507], videos [249, 326, 356, 427, 472, 487], or multiple views [72, 120, 121, 208, 337, 402, 506]. The number of given views in a formulation provides the flexibility of operations that we could do with the new views. A single input view [385, 431, 432, 466, 507] restricts the amount of deviation of the synthesized views from the original view. Multiple views [356, 402, 487, 506] depending on the amount of distance between the cameras gives a flexibility to synthesize a novel scene. A large body of work [120, 121, 208, 402, 506] has been restricted to short baseline between the cameras. This once again restricts the deviation from the original views. Recent work [356, 487] has used videos from a moving camera to enable diverse view synthesis. However, this formulation is either restricted to static scenes [356] or visible components in a particular view for a given time instant [487]. In this work, our goal is to use arbitrarily spread multi-views that allows us to deviate largely from the original views, consider dynamics in the scene, and also deal with disocclusion.

Image-based Rendering: Multi-view view-synthesis approaches have largely benefitted from a large body of work on image-based rendering [286, 392, 412]. Shum and Kang [392] classified the different image-based rendering in three categories depending on the utilization of geometry to synthesize new views: (1) the approaches that **use geometry via depth modeling** [291, 293, 319, 380, 506]; (2) in the middle are the approaches that **use multi-view geometry via correspondences** [69, 369, 375]; and (3) finally **plenoptic modeling** [2, 148, 247, 287, 294] that use little geometry. In this work, we study these different directions and study their limitations in our setup. Figure 2.4 position these approaches depending on the use of geometry. In this work, we use the insights from the body of work on using multi-view geometry by establishing correspondences between multiple-stereo pairs. We then use insights from layered depth imaging (Shade et al. [380]) to synthesize a new view from the multiple-stereo pairs. We, therefore, position **our approach** in between layered depth imaging and correspondences.

Layered Depth Imaging: Given depth information for every pixel, we can synthesize new nearby viewpoints by 3D warping techniques [286]. Shade et al. [380] proposed layered depth image that stores information at both visible and hidden surfaces in the input image. The idea of layered depth imaging has been recently revisited by recent work [293,506] that use multi-plane imaging. We study LLFF [293] here to synthesize a new view using 12 views. These approaches [293,506] assume perfect multi-plane imagining for composition. We observe multi-plane imaging for sparse views are not accurate and leads to blurry results. Figure 2.7-(a) shows the target view synthesized using LLFF [293].

Correspondences: Instead of explicitly modeling the depth, pixel-correspondences between views can also be used for warping pixels to generate new views. These approaches [69,369,375] were restricted to two-views. This direction has been mostly demonstrated for a two-view or a stereo pair. In this work, we revisited this direction. We estimated disparity using an off-the-shelf disparity estimation module [482] for multiple-stereo pairs. We project the pixels from the known views to the target view using correspondences. This results in multiple target images. The crucial challenge is how to fuse multiple projections. Naively fusing them by taking a per-pixel median leads to lose in information (as shown in Figure 2.7-(b)). Chen and Williams [69] (introduced this direction and) suggested view-independent visible priority but visibility is view dependent when dealing with wide baseline scenarios. In this work, we use the insights from Shade et al. [380] to synthesize a new view using multiple projections. This enable us to deal with the lose in information. Figure 2.8-(e) shows the output of the first step of our approach that estimates time-specific component (instantaneous foreground) from multiple stereo pairs.

Plenoptic Modeling: Plenoptic function [2,51,148,247,287,309] describes the entire image information visible from a particular viewing angle. The goal is to capture every possible light ray in the scene and synthesize a new view by composing information from different light rays. This direction started with the seminal work of Adelson and Bergen [2], followed by work on light-field cameras [50,309,451]. Recently, Neural Radiance Fields (NeRF) [294] revisited this idea and introduce view-synthesis from multiple views. NeRF is, however, currently restricted to dense multi-views with good camera parameters, which is not available in our setup. Figure 2.7-(c) shows a view synthesized using 12 views of a scene from our setup. We observe noisy outputs. We also show the nearest camera view in Figure 2.7-(d).

Neural Rendering: While the image-based rendering approaches allows us to fuse information from multiple views. However, we still do not get the output close to a real image. In this work, we utilize insights from literature on neural rendering [417] using neural networks enabling image synthesis in general [88, 147, 197, 203, 257, 492]. These data-driven approaches using convolutional neural networks [242] have led to impressive results in image synthesis. These results inspired a large body of work [9, 120, 121, 208, 264, 290, 398, 402, 506] on continuous view synthesis for small baseline shifts. Hedman et al. [174] extended this line of

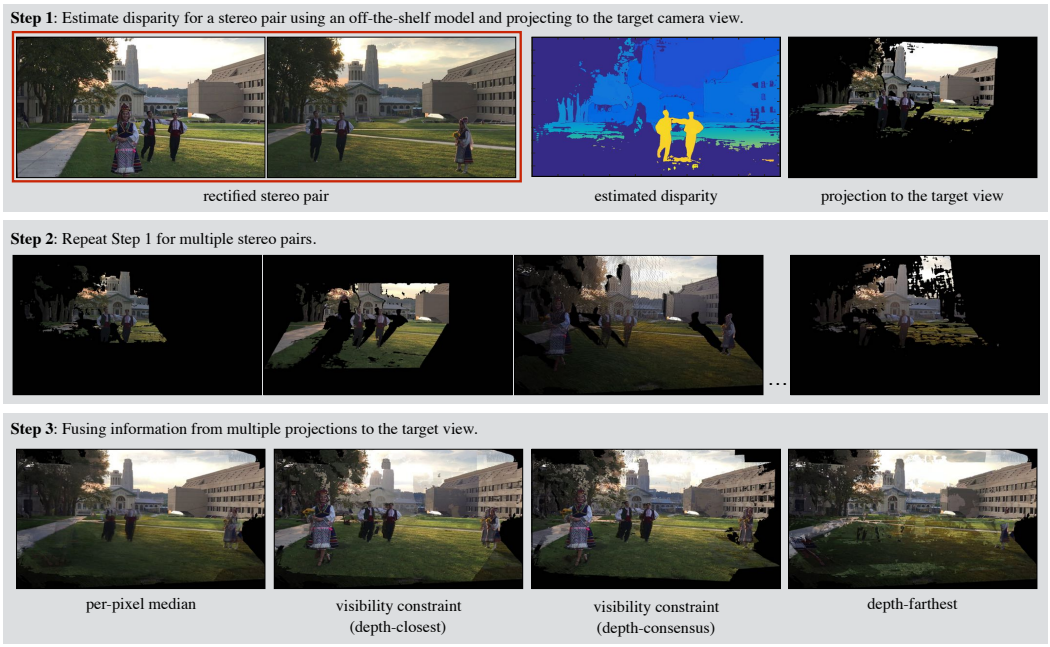


Figure 2.11: **Instantaneous Foreground Estimation:** Given multiple stereo pairs and a target camera view, there are three steps for estimating instantaneous foreground. (1) We estimate disparity for a rectified stereo pair using an off-the-shelf disparity estimation approach [482] and project it to the target camera view using standard 3D geometry [164]. (2) We repeat Step-1 for the $\binom{N}{2}$ stereo pairs. (3) A crucial aspect is to fuse the information from multiple projections to generate a comprehensive instantaneous foreground. A simple per-pixel median over the multiple projections leads to a loss of dynamic information because of the poor 3D estimates (shown in first column). We use the insights from Shade et al. [380] to synthesize a new view using the information from multiple-stereo pairs. Similar to LDI, each pixel in the input image contains a list of depth and color values where the ray from the pixel intersects with the environment. This visibility constraint is similar to painter's algorithm. The visibility constraint builds a per-pixel cost volume of depth. It is natural to have multiple depth values for a given 2D pixel location in the image. For each pixel location, we consider the 3D point corresponding to the *closest depth* to the target camera view. Due to noisy 3D estimates, it often leads to ghosting artifacts (shown in second column). However, we have multiple views and we ensure consistency in the depth values of the closest 3D point for projection to a 2D pixel location by analyzing their visibility. As shown in third column, this *consensus in depth* leads to improved results. Finally, we also show the projection of pixels corresponding to the *farthest points* (fourth column) for a given pixel location to illustrate cost volume of depth.

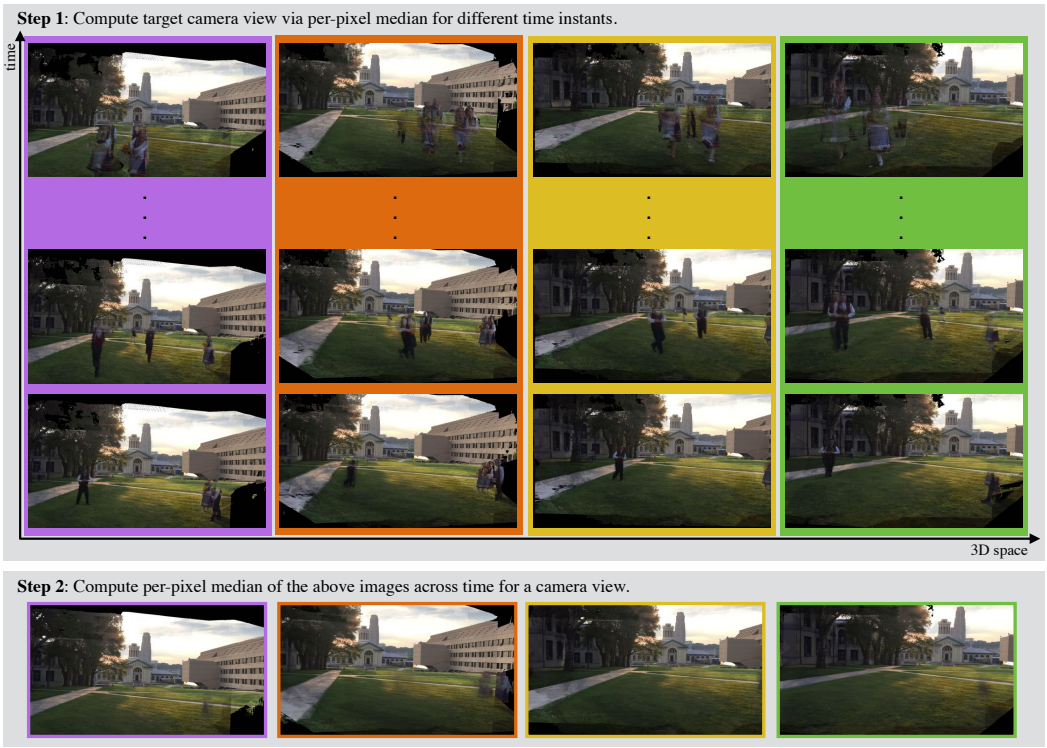


Figure 2.12: **Static Background Estimation:** (1) For a given camera pose and a time instant, we use per-pixel median to compute target camera view. While it looses the dynamic information but yields smooth background. We compute the images for a given camera pose across all time. (2) A per-pixel median of the images over a large temporal window for a target camera pose results in a smoother stationary background.

work to free-viewpoint capture. However, these methods are currently applicable to static scenes only. We combine the insights from CNN-based image synthesis and earlier work on image-based rendering to build a data-driven 4D Browsing Engine that makes minimal assumption about the content of multi-view videos. Figure 2.9 shows a virtual view synthesized using our approach.

2.2 4D Browsing Engine

We are given N camera views with known extrinsic parameters and intrinsic parameters. Our goal is to generate virtual camera view that does not exist in any of these N cameras. We temporally align all cameras using spatiotemporal bundle adjustment [442], after which we assume the video streams are synchronized.

Our method should be robust to possible alignment errors. Figure 2.10 gives an overview of our approach. There are three crucial steps: (1) Estimate instantaneous foreground using classical image-based rendering method as discussed in Section 2.2.1; (2) Estimate static background using simple pixel operation over time as discussed in Section 2.2.2; and (3) Learn a scene-specific composition of instantaneous foreground and static background in a self-supervised manner as discussed in Section 2.2.3. We finally discuss practical details and design decisions in Section 2.3.

2.2.1 Instantaneous Foreground Estimation

The N-camera setup provides us with $\binom{N}{2}$ stereo pairs. We build foreground estimates for a given camera pose and a time using multiple rectified stereo pairs. There are three essential steps for estimating instantaneous foreground (shown in Figure 2.11). First, we begin with estimating disparity using an off-the-shelf disparity estimation module [482]. The knowledge of camera parameters for the stereo pair allows us to project the pixels to a target camera view using standard 3D geometry [164]. Figure 2.11-1 shows an example of projection for a rectified stereo pair. Second, we repeat Step-1 for multiple stereo pairs. This step gives us multiple projections to a target camera view (shown in Figure-2.11-2) and per-pixel depth estimates. The projections from various $\binom{N}{2}$ stereo pairs tends to be noisy due to sparse cameras, large stereo baseline, bad stereo-pairs, or errors in camera-poses. We cannot naively fuse the multiple projections to synthesize a target view in all conditions for a target camera view. As an example, a simple per-pixel median on these multiple projection leads to the loss of dynamic information as shown Figure-2.11-3 (first column). Third, We enforce a multi-view visibility constraint to fuse information from multiple stereo pairs.

Multi-View Visibility Constraint: We use the insights from Shade et al. [380] to synthesize a new view using the information from multiple-stereo pairs. Similar to LDI [380], each pixel in the input image contains a list of depth and color values where the ray from a pixel location intersects with the environment. We build a per-pixel cost volume of depth that allows us to trace the ray of light to its origin for a given camera view. Since we have multiple views, it is natural to have multiple depth values for a given 2D pixel location in the image. For each pixel location, we consider the 3D point corresponding to the closest depth. However, the closest 3D point from the cost volume of depth may not be accurate in our setup. As shown in the second column of Figure-2.11-3, we observe ghosting artifacts and misaligned outputs due to noisy 3D estimates (e.g. see the building in the background).

Fortunately, we can use multiple views to ensure consistency in the depth values of the closest 3D point that we should consider for projection at a given 2D pixel location. While a certain 3D location may not be viewed by all the cameras, it is highly likely that it would be seen in a significant subset of cameras. This means that we will estimate depth of same 3D point in our setup from many stereo pairs.

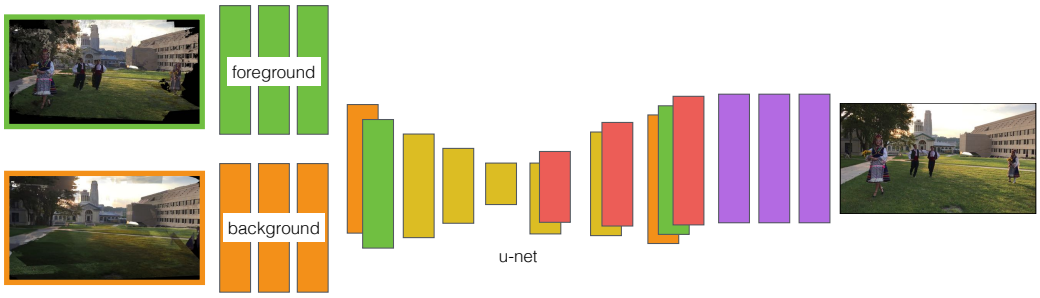


Figure 2.13: **Self-Supervised Composition of Foreground and Background:** Given an input-output paired data created for a held out camera, we train a neural network to learn the composition. We show our neural network architecture that is used to compose the instantaneous foreground ($f_{c,t}$) and static background (b_c). The first part of our network consists of three convolutional layers. We concatenate the outputs of both foreground and background streams. The concatenated output is then fed forward to the standard U-Net architecture [358]. Finally, the output of U-Net is then fed forward to three more convolutional layers that give the final image output.

We use this insight to build a consensus about the closest 3D point by ensuring that same depth value is observed in at least two stereo pairs. We start with the smallest depth value (> 0) at a pixel location in the cost volume and iterate till we find a consensus. We ignore the pixel location (leave it as blank) if it is not observed from a sufficient number of cameras (less than 3). Ideally, the depth values of a 3D point (though computed from different stereo pairs) should be same from the target camera view. For the practical purposes, we define a threshold that can account for noise. As shown in third column of Figure-2.11-3, this leads to improved results. However, there is still missing information and ghosting artifacts in this output to qualify as a real image.

2.2.2 Static Background Estimation

The missing information in the output of Section 2.2.1 is due to the lack of visibility of points across multiple views for a given time instant. We accumulate long-term spatiotemporal information to compute static background for a target camera view. More explicitly, we project the dense stereo depth estimated at every time instance of the temporal window of $[0, t]$ to the target camera position. Figure 2.12-1 shows examples of virtual cameras for various poses and time instants. For a given camera pose and a time instant, we use per-pixel median to compute target camera view. While it looses the dynamic information but yields smooth background.

We compute the images for a given camera pose across all time. A per-pixel median of different views (across time) for a given camera pose results in a smoother

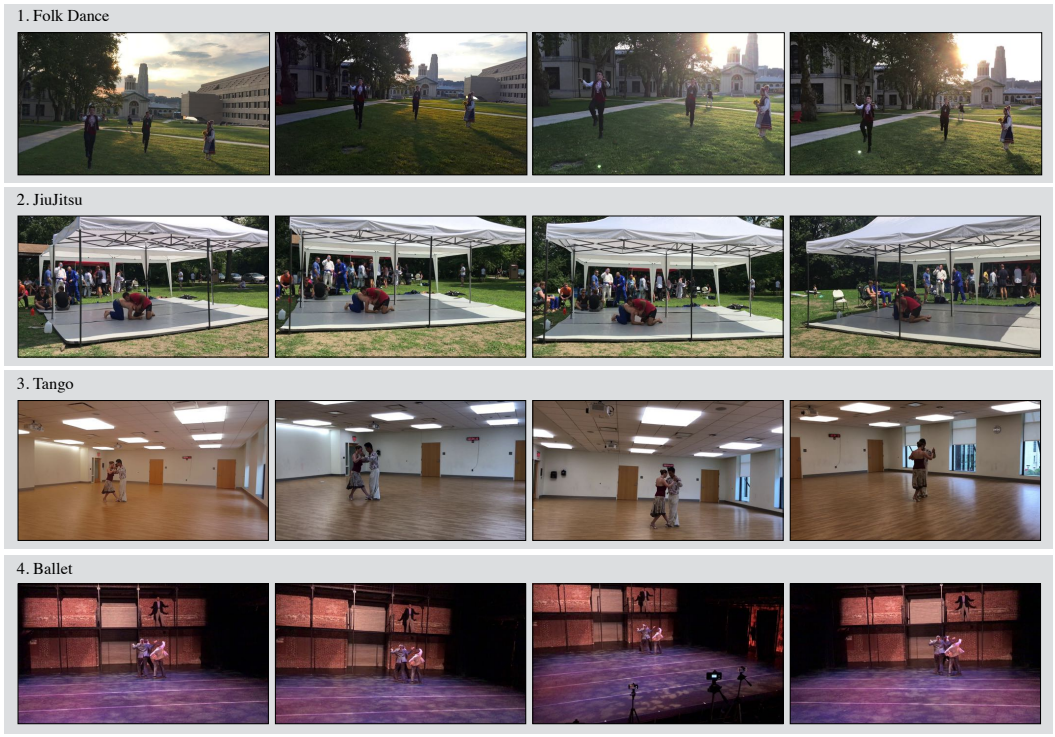


Figure 2.14: **Human Performances**: We captured a wide variety of human performances from multiple cameras. The human performances in these sequences have a wide variety of motion, clothing, human-human interaction, and human-object interaction . These sequences were captured in varying environmental and illumination condition. Shown here are examples from four such sequences to give a sense of extremely challenging setup dealing with a wide and arbitrary camera baseline.

static background. Empirically, such median image computed over large temporal window for a given camera position also contain textureless and non-Lambertian stationary surfaces in a scene (observed consistently in Figure 2.12-2, Figure 2.3, and Figure 2.10). Note that our approach makes no assumption about the camera motion and is applicable to both stationary and moving camera. We now have a pair of complementary signals: (1) sparse and noisy foreground estimates; and (2) dense but static background estimates.

2.2.3 Self-Supervised Composition

We use a data-driven approach to learn the fusion of instantaneous foreground, F, and static background, B, to generate the required target view for given camera parameters. The instantaneous foreground for a camera pose c and time t is notated as $f_{c,t}$. The static background for a camera pose c is notated as b_c. However,

there exists no ground truth or paired data to train such a model in a data-driven manner. We formulate the data-driven composition in a self-supervised manner by reconstructing a known held-out camera view, $i_{c,t}$, from the remaining $N-1$ views. This gives us a paired data $\{((f_{c,t}, b_c,), i_{c,t})\}$ for learning a mapping $G : (F, B) \rightarrow I$. We now have a pixel-to-pixel translation [197] and therefore, we can easily train a convolutional neural network (CNN) specific to a scene. We use two losses for optimization: (1) Reconstruction loss; and (2) Adversarial loss.

Reconstruction Loss: We use standard l_1 reconstruction loss to minimize reconstruction error on the content with paired data samples:

$$\min_G L_r = \sum_{(c,t)} ||i_{c,t} - G(f_{c,t}, b_c)||_1. \tag{2.1}$$

Adversarial Loss: Recent work [147] has shown that learned mapping can be improved by tuning it with a discriminator D that is adversarially trained to distinguish between real samples of $i_{c,t}$ from generated samples $G(f_{c,t}, b_c)$:

$$\min_G \max_D L_{adv}(G, D) = \sum_{(c,t)} \log D(i_{c,t}) + \sum_{(c,t)} \log(1 - D\big(G(f_{c,t}, b_c)\big)\big). \tag{2.2}$$

Network Architecture & Optimization: We use HD-image inputs (1080×1920). The input images are zero-padded making them 1280×2048 in dimensions. We use a modified U-Net architecture in our formulation as shown in Figure 2.13. There are three parts of our neural network architecture. (1) The first part of our network consists of three convolutional layers. We concatenate the outputs of this part for both foreground and background stream. (2) The concatenated output is fed forward to the second part that is a standard U-Net architecture [358]. (3) Finally, we feed the outputs of U-Net through three more convolutions that generates the final image. The batch-size is 1. The number of filter, n_f, in the first conv-layer of our network is 18. The number of filters increase by a factor of 2 with every conv-layer till the bottleneck of U-Net. For data augmentation purposes, we randomly resize the input to upto $1.5\times$ scale factor and randomly sample a crop from it. We use the Adam solver. A model is trained from scratch with a learning rate of 0.0002, and remains constant throughout the training. The model converges quickly in around 10 epochs for a sequence.

2.3 Practical Limitations and Design Decisions

A crucial practical challenges is to hallucinate missing information. In this section, we discuss the design decisions that we made to overcome this challenge.

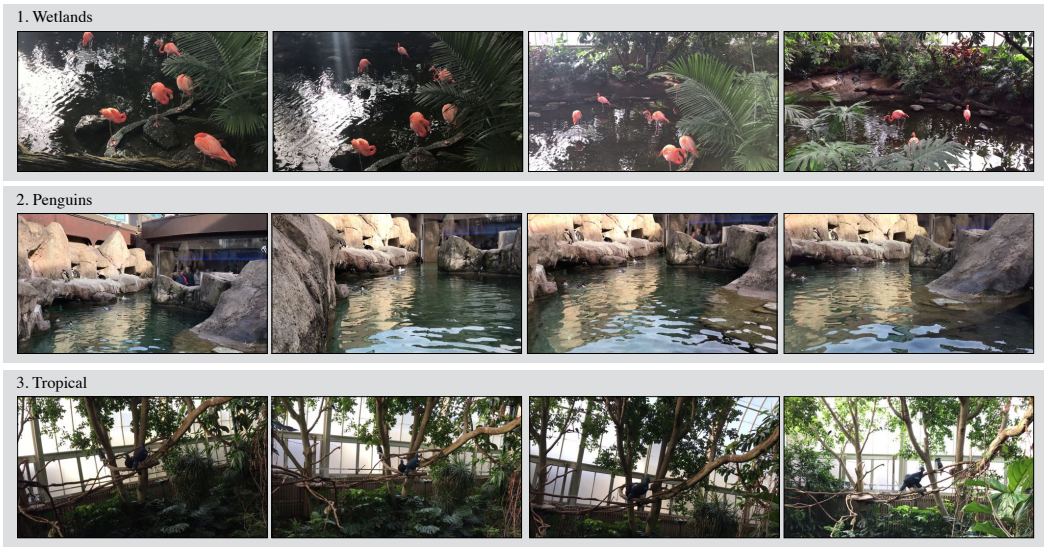


Figure 2.15: **Bird Sequences**: We captured a wide variety of birds at the National Aviary of Pittsburgh. We have no control on the motion of birds, their environment, lighting condition, and dynamism in the background due to human movement in the aviary. Shown here are examples from three such sequences to give a sense of our capture scenario.

2.3.1 Hallucinating Missing Information

We have so far assumed the scenario where sufficient foreground and background information are available that can be composed to generate a comprehensive new view. However, it is possible that certain regions are never captured (e.g. blank pixels in foreground and background estimation in Figure 2.11 and Figure 2.12 respectively). This is possible due to many reasons such as sparse cameras, large stereo baseline, bad stereo-pairs, or errors in camera-poses.

While filling smaller holes is reasonable for a parametric model, filling larger holes lead to temporally inconsistent artifacts. One way to deal with it is to learn a higher capacity model that may learn to fill larger holes. Other than fear of overfitting, the larger model needs prohibitive memory. Training a neural network combining the background and foreground information with HD images already require 30 GB memory of a v100 GPU. In this work, we use a stacked multi-stage CNN to overcome this issue.

Multi-Stage CNN: We use a high capacity model for low-res image generation that learns overall structure, and improve the resolution with multiple stages. We train three models for three different resolutions, namely: (1) low-res (270×480); (2) mid-res (540×960); and (3) hi-res (1080×1960). These models are trained independently and form multiple stages of our formulation. At test time, we use these

models sequentially starting from low-res to mid-res to hi-res outputs. This sequential processing is done when there are large holes in the instantaneous foreground and static background inputs. The output of the low-res model is used to fill the holes to the input of next model in sequence. Here we leverage the fact that we can have a very high capacity model for a low-resolution image and that holes become smaller as we make the image smaller. The overall network architecture remains the same as the high-res model described in Section 2.2.3. We mention the differences for the low-res and mid-res models.

Low-Res Model: The low-res model inputs 270×480 images. As such, it can have more parameters than a mid-res or hi-res model that inputs $4\times$ and $16\times$ higher resolution inputs respectively. We use a $n_f = 32$ for this model. The input images are zero-padded, and make them 320×512 in dimensions.

Mid-Res Model: We use a $n_f = 28$ for this model. The input images are zero-padded, and make them 640×1024 in dimensions.

2.4 Unconstrained Multi-View Dataset

We collected a large number of highly diverse sequences of unrestricted dynamic events consisting of humans and birds. These sequences were captured in different environments and varying activities using up to 15 hand-held mobile phones. We describe a few human sequences in Section 2.4.1 and a few bird sequences in Section 2.4.2. Figure 2.14 and Figure 2.15 show some examples of sequences to give a sense of our setup.

2.4.1 Human Performances

We captured a wide variety of human motion, human-human interaction, human-object interaction, clothing, both indoor and outdoor, under varying environmental and illumination conditions.

Western Folk Dance: We captured sequences of western folk dance performances. This sequence is challenging due to *flowing* dresses worn by performers, open hair, self-occlusions, and illumination conditions. Such a sequence paves the path for explicit parametrization of illumination condition in 4D modeling. Figure 2.14-1 shows one of the two western folk dance sequences that we captured.

Jiu-Jitsu Retreat: Jiu-Jitsu is a type of Brazilian Martial Art. We captured sequences of this sporting event during a summer retreat of the Jiu-Jitsu group. This sequence is an extreme example of unchoreographed dynamic motion from more than 30 people who participated in it. Figure 2.15-2 shows the capture of a JiuJitsu event in the foreground with arbitrary human motion in a picnic in the background.

Tango: We captured sequences of Tango dance in an indoor environment. Both performers wore proper dress for Tango. Self-occlusion between the performers makes it challenging for 4D visualization. Shown in Figure 2.14-3 is an example of

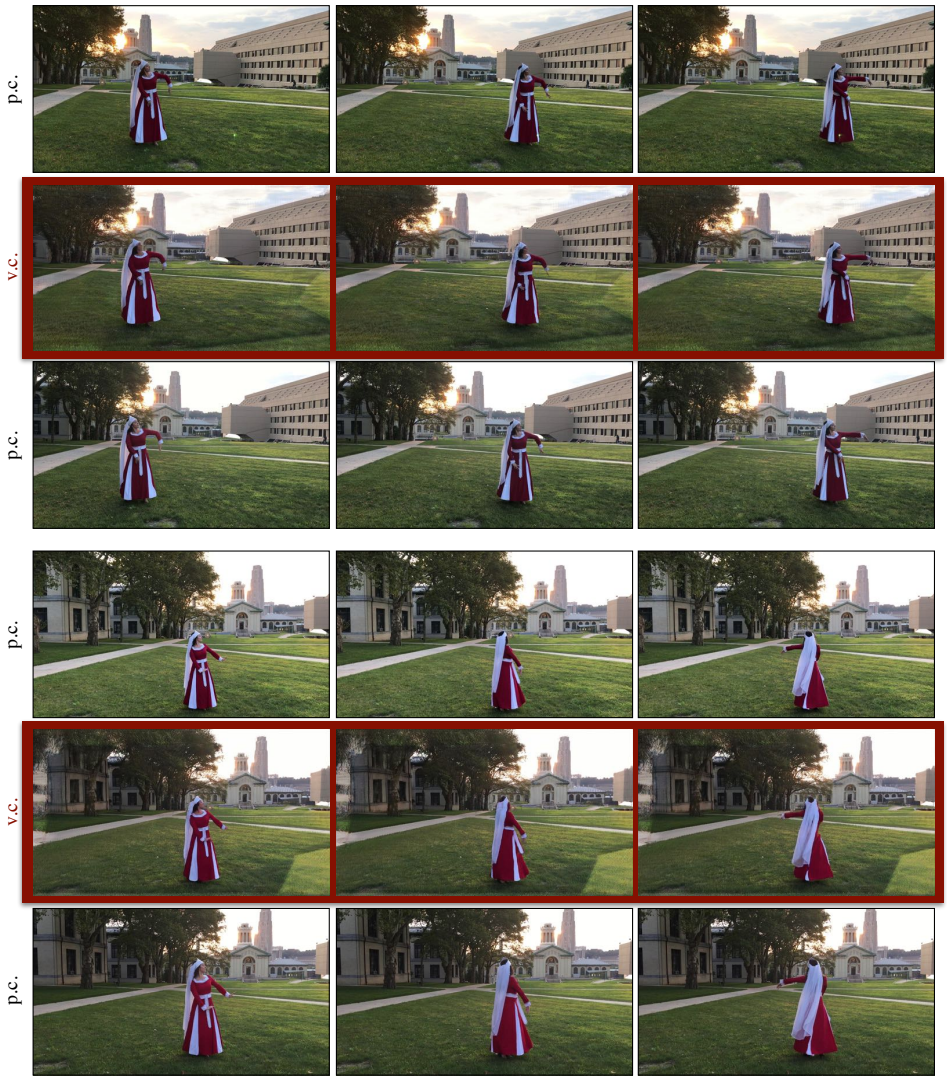


Figure 2.16: **Flowing Dress and Open Hair:** We show two examples of virtual cameras generated for the Western Folk Dance sequence in which the performer is wearing a flowing dress with open hairs. For each virtual camera (v.c.), we show the physical cameras (p.c.) on its sides. Zoom-in for the detailed human motion and the dress in these two examples.

one of the four Tango dance sequences that we captured. Note the reflections on the semi-glossy ground and featureless surroundings.

Performance Dance: We captured many short performance dances including Ballet, and renactments of plays. These sequences were collected inside an auditorium. The illumination, clothing, and motion change drastically in these sequences. Fig-

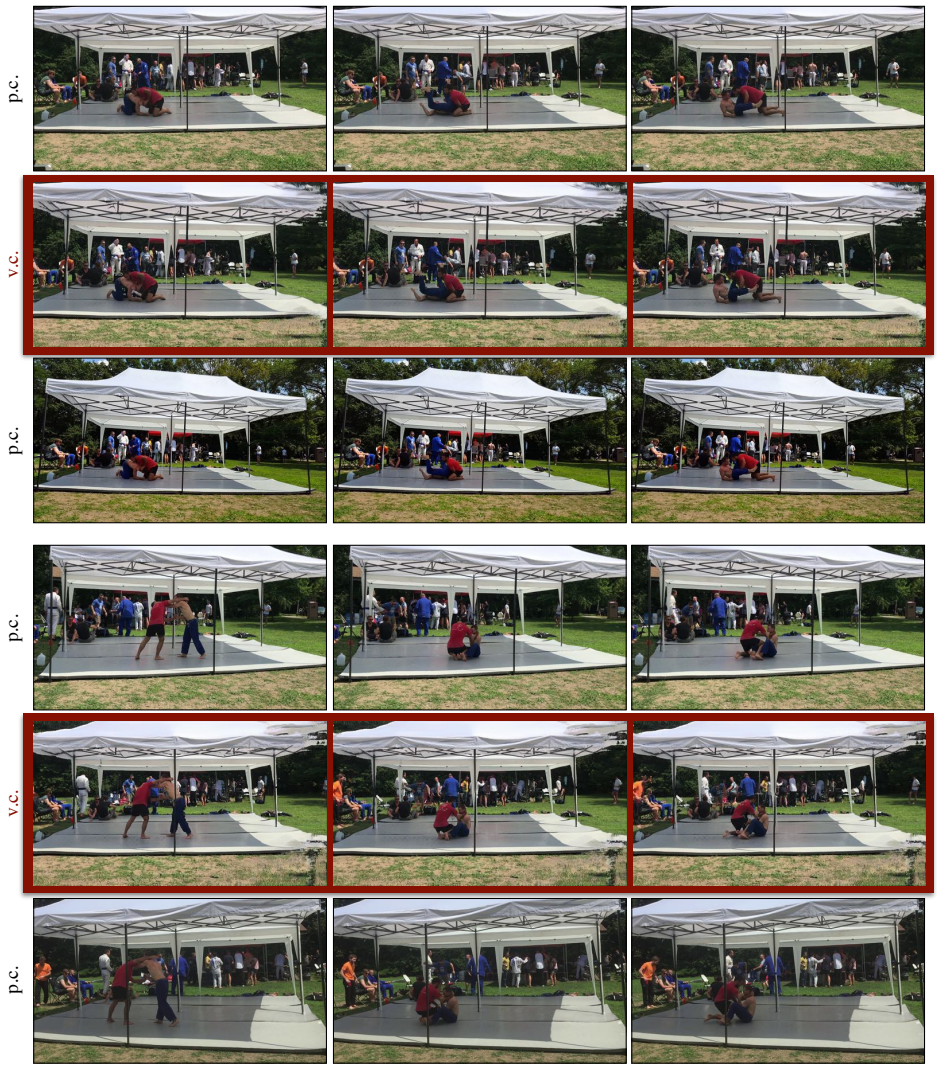


Figure 2.17: **Many People and Unchoreographed Sequence:** We show two examples of virtual cameras generated for Jiu-Jitsu Retreat sequence that is an example of capture with many people and unchoreographed event. For each virtual camera (v.c.), we show the physical cameras (p.c.) on its sides.

ure 2.14-4 shows an example of performance with an extremely wide baseline and challenging illumination.

Sequences from Prior Work: We also used sequences from Vo et al. [444] to properly compare our results with their 3D reconstruction (COLMAP+humans, shown in Figure 2.3).

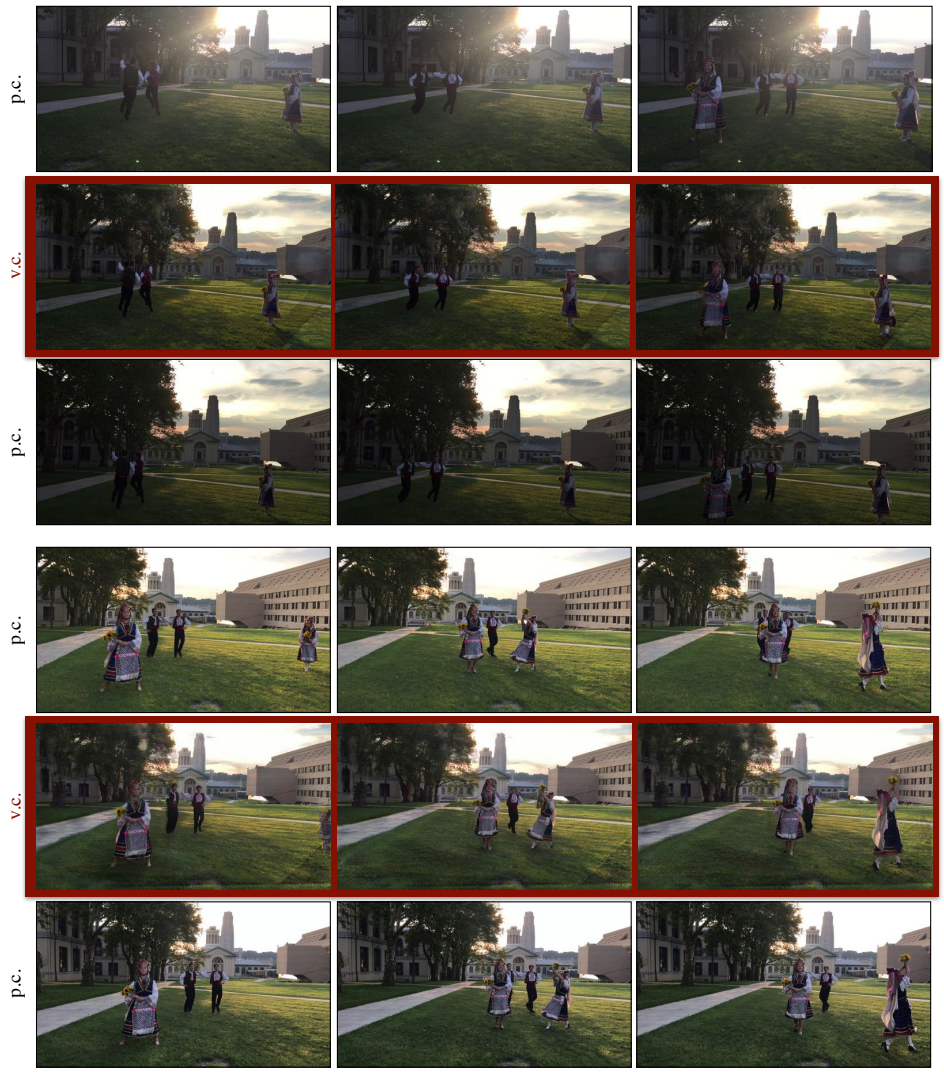


Figure 2.18: **Challenging Illumination and Multiple Dancers:** We show two examples of virtual cameras generated for another Western Folk Dance sequence with challenging illumination condition and self occlusion due to multiple dancers. For each virtual camera (v.c.), we show the physical cameras (p.c.) on its sides.

2.4.2 Bird Sequences

We captured a wide variety of birds at the National Aviary of Pittsburgh. We have no control on the motion of birds, their environment, lighting condition, and dynamism in the background due to human movement in the aviary.

Wetlands: The American Flamingos are the most popular wetlands bird at the Na-

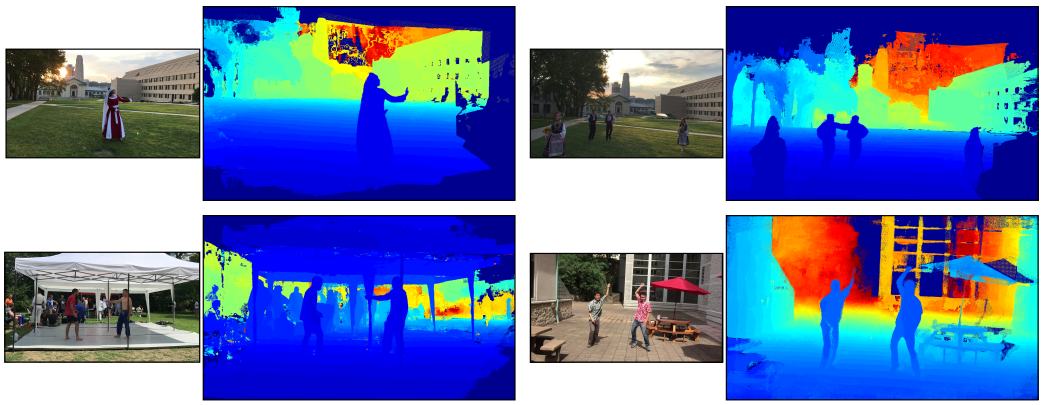
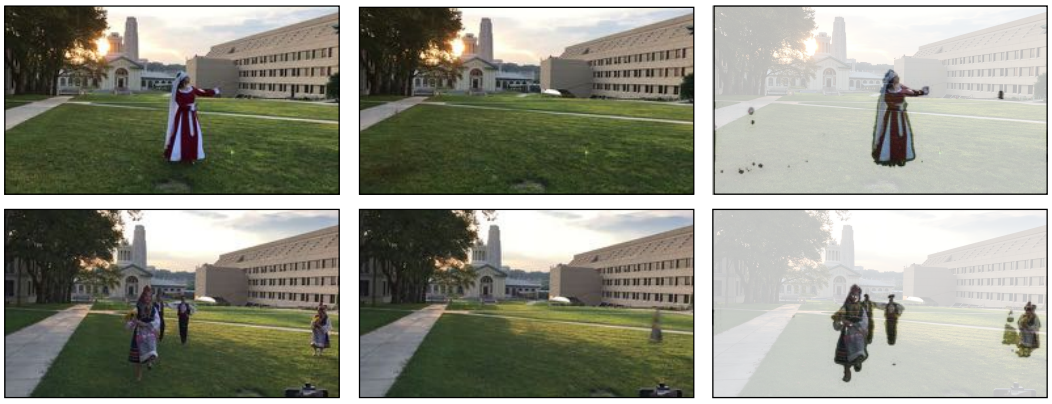


Figure 2.19: **Depth Map:** We show estimated depth map using heat-map (where blue means closer and red means farther) for the images from various sequences. The dark-blue region are the ones where depth could not be reliably estimated. The depth-volume and multi-view visibility constraint enables us to compute a dense depth map for a view. We use depth value of 3D point selected for each pixel location without any post-processing. We are able to capture fine details of human body here as shown in various examples. Best viewed in electronic format.

tional Aviary of Pittsburgh. In this sequence, we captured a free motion of many American Flamingos from multiple cameras. Slowly moving water with reflection of surroundings in which these birds live makes it challenging and appealing (Figure 2.15-1). There may be an occasional sight of Brown Pelicans and Roseate Spoonbills in this sequence.

Penguins: There are around 20 African Penguins at the Penguin Point in National Aviary of Pittsburgh. Penguin Point consists of rocky terrain and a pool for penguins. The arbitrary motion of penguins and reflection in water makes this sequence totally uncontrolled. Figure 2.15-2 show examples of one sequence captured at the Penguin Point. Due to the dynamic background, there is also a frequent movement of humans in this sequence.

Tropical Rain-forests: There are a wide variety of tropical rain forest birds at the National Aviary of Pittsburgh. These include Bubba the Palm Cockatoo, a critically endangered parrot species, Gus and Mrs. Gus the Great Argus Pheasants, a flock of Victoria Crowned Pigeons, Southern Bald Ibis, Guam Rails, Laughing Thrushes, Hyacinth Macaws, and a two-toed Sloth. The dense trees in this section act as a natural occlusion while capturing the birds and makes it exciting for 4D capture. Figure 2.15-3 show examples of sequence that captured Victoria Crowned Pigeons.



(a) original image (b) static background (c) extracted foreground

Figure 2.20: **Foreground-Background Segmentation for Stationary Cameras: (a)** Given a sequence of stationary cameras (showing original image for two sequences), **(b)** we compute static background by computing per-pixel median on the sequence. **(c)** We use both original image and static background in our formulation to extract dynamic foreground. The results here are without any post-processing.

2.5 Quantitative Analysis

We used sequences from Vo et al. [442] to properly compare our results with their results (COLMAP+Humans). Figure 2.3 and Figure 2.10 shows the results of freezing the time and exploring the views for these sequences.
Evaluation: We use a mean-squared error (MSE), PSNR, SSIM, and LPIPS [494] to study the quality of virtual camera views created using our approach. **MSE**: Lower is better. **PSNR**: Higher is better. **SSIM**: Higher is better. **LPIPS**: Lower is better. We use held-out cameras for proper evaluation. We also compute a **FID** score [179], lower the better, to study the quality of sequences where we do not have any ground truth (e.g., freezing the time and exploring views). This criterion contrast the distribution of virtual cameras against the physical cameras.

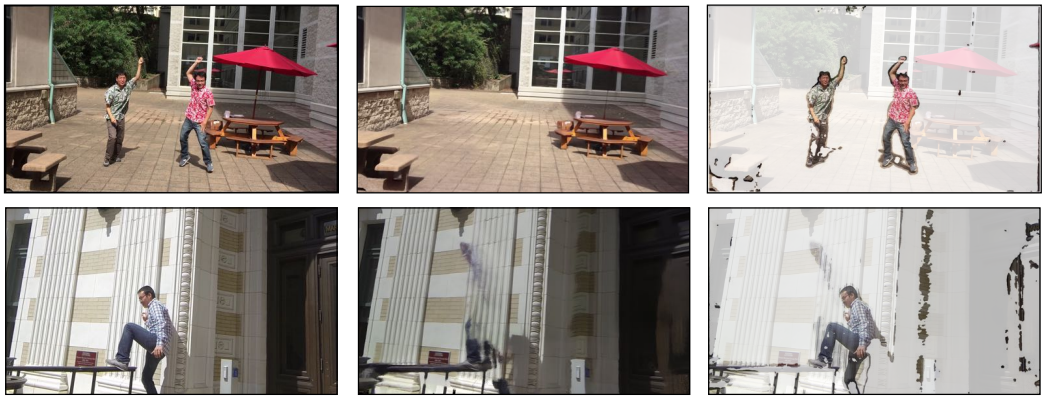
Baselines: To the best of our knowledge, there does not exist a work that has demonstrated dense 4D visualization for in-the-wild dynamic events captured from unconstrained multi-view videos. We, however, study the performance of our approach with: (1) a simple nearest neighbor baseline **N.N.**: We find nearest neighbors of generated sequences using conv-5 features of an ImageNet pre-trained AlexNet model [238]. This feature space helps in finding the images closer in structure. Additionally, there are two kinds of nearest neighbors in our scenario. The first is the nearest camera view at the **same time** instant and the other is the nearest camera view across **all time** instants.; (2) **COLMAP+humans**: We use work from Vo et al [442–444] for these results.; and finally (3) we contrast it with instantenous

Approach	M.S.E	PSNR	SSIM	LPIPS [494]	FID [179]
N.N (same-time)	5577.65 ±927.47	10.72 ±0.73	0.27 ±0.05	0.57 ±0.07	-
N.N (all-time)	4948.64 ±1032.76	11.29 ±1.00	0.31 ±0.04	0.50 ±0.11	-
COLMAP + Humans	3982.26 ±1050.28	12.25 ±1.02	0.33 ±0.08	0.45 ±0.022	190.84
Inst.	5956.25 ±3139.61	11.18 ±2.89	0.42 ±0.17	0.43 ±0.16	100.10
Ours	**714.95 ±364.99**	**20.14 ±2.21**	**0.79 ±0.07**	**0.13 ±0.06**	**21.98**

Table 2.1: **Comparison:** We contrast our approach with: (1). a simple nearest neighbor (**N.N.**) baseline. There are two kinds of nearest neighbors in our scenario. The first is the nearest camera view at the **same time** instant and the other is the nearest camera view across **all time** instants. (2). reconstructed outputs of **COLMAP+Humans**; and finally (3). instantaneous foreground (**Inst**) defined in Section 2.2.1. We use various evaluation crietria to study our approach in comparisons with these three methods: (1). **M.S.E**: We compute a mean-squared error of the generated camera sequences using held-out camera sequences.; (2). **PSNR**: We compute a peak signal-to-noise ratio of the generated sequences against the held out sequences; (3). **SSIM**: We also compute a SSIM in similar manner.; (4). We also use LPIPS [494] to study structural similarity and to avoid any biases due to MSE, PSNR, and SSIM. Lower it is, better it is. Note that all the above four criteria are computed using held-out camera sequences; and finally (5) we compute a **FID**-score [179] to study the quality of generations when a ground-truth is not available for comparisons. Lower it is, better it is.

foreground image (**Inst**) that is defined in Section 2.2.1.

Table 2.1 contrasts our approach with various baselines on held-out cameras for different sequences. We observe significantly better outputs under all the criteria. We provide more qualitative analysis on our project page – http://www.cs.cmu.edu/~aayushb/Open4D/



(a) original image (b) per-frame static background (c) extracted foreground

Figure 2.21: **Foreground-Background Segmentation for Moving Cameras:** (**a**) Given a sequence of moving cameras (showing original image for two sequences), (**b**) we compute per-frame static background. (**c**) We use both original image and per-frame static background in our formulation to extract dynamic foreground. The results here are without any post-processing.

2.6 Depth Map from Multi-Views

Our approach allows us to estimate dense depth map for a scene. We use the depth-volume and visibility constraint to compute dense depth maps. We use depth value of a 3D point selected for each pixel location without any post-processing. Figure 2.19 shows the depth maps for various scenes computed using our approach. We visualize the depth maps using heat-map (where blue means closer and red means farther). The dark-blue region are the ones where depth could not be reliably estimated as these regions did not satisfy the multi-view constraints. We also contrast the depth-map from a single image computed using MiDAS [348] in Figure 2.26 and Figure 2.27. The depth estimated using multi-views is able to better capture the details in the scene, especially the human performer, buildings in the background, and the ground.

2.7 Foreground-Background Segmentation

Our approach allows us to segment foreground moving objects in the scene. This property is possible because we compute: (1) $I_{c,t}$: the view for a given camera pose, c, and a time instant t; and (2) b_c: static background for the camera pose, c, averaged over time. We label each pixel as a foreground (1) or background (0) in the image. A pixel in the image is represented using an $N = 128$ dimensional multi-scale feature descriptor (h). We follow [24] and construct a pixel descriptor using

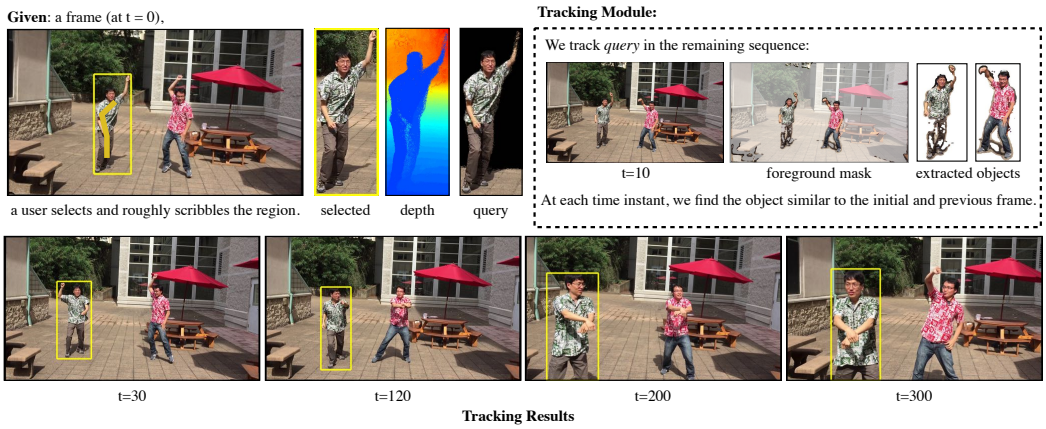


Figure 2.22: **Object Tracking in Videos:** A user selects a region in a given frame and roughly scribbles it. The bounding box provides us with a selected region. We use the depth information and scribbled region to get a range of depth for the object of interest. This allows us to segment the required region at a suitable depth location (*query*). **Tracking Module:** For every new frame in the sequence, we get a foreground mask. This allows us to extract different object proposals (*extracted objects*) in the foreground. At each time instant, we find the proposal most similar to the initial query and the previous best box. The similarity between the query and a proposal is measured via the cosine similarity of their normalized *fc-7* features obtained from a pre-trained AlexNet model [238]. We get a similarity score by computing an average of the cosine similarity between the query and a proposal, and the previous best box and a proposal. We consider the proposal box with maximum similarity score. **Tracking Results:** We show the results of tracking using our approach for varying time instants.

features from conv-$\{1_1, 1_2\}$ of a PixelNet [21] model trained for semantic segmentation using PASCAL Context [302]. The label at a pixel location (x, y) is 1 if the l_1 distance between $h(I(x, y))$ and $h(b(x, y))$ is greater than an *empirically observed* threshold otherwise it is 0:

$$label(x, y) = \begin{cases} 1, & ||h(I(x, y)) - h(b(x, y))||_1 \geq 5 \\ 0, & \text{otherwise} \end{cases}$$

Figure 2.20 and Figure 2.21 shows the static background and segmented foreground masks for a stationary camera and a moving camera respectively.

2.8 Object Tracking

In this work, we ask a user to mark the object of interest in one frame of the sequence. We then track it in the remaining frames. Figure 2.22 shows an example of tracking foreground object in a video. A frame (at time $t = 0$) is shown to a human-user. The user selects the foreground region to be removed by making an appropriate bounding box and roughly scribbling it. The bounding box provides us the selected region and scribbling provides us a rough depth information of the target region, thereby yielding a segmented region (*query*). We track the object in the remaining sequence. At each time instant, we estimate a foreground mask and extract proposals (shown as *extracted objects* in the 2.22). This drastically reduces the search space and allows us to find the object of interest in a frame that is similar to the *query* and the best output in previous frame. The similarity between the query and a proposal is measured via the cosine similarity of their normalized *fc-7* features obtained from a pre-trained AlexNet model [238]. We get a similarity score by computing an average of the cosine similarity between the query and a proposal, and the previous best box and a proposal. We consider the proposal box with maximum similarity score. Figure 2.22-bottom shows the results of tracking the green-shirt performer.

2.9 Revealing Hidden Components

We have a comprehensive 4D control of dynamic events: (1) 3D space as we have appropriate depth information; and (2) temporal aspects of the foreground as we can track a foreground object in the videos. This 4D control allows us to do a geometrically consistent editing of the dynamic events. A user can see behind the occlusions, edit, add, or remove foreground objects in the scene. To accomplish this, a user only needs to mark the required portion in a video. The tracking module (Section 2.8) allows us to find the instances of the object of interest in the remaining frames of video. Our instantaneous foreground estimation module ignores the information at the depth value corresponding to the tracked component. Instead, it considers the second closest region (behind the closest region). The static background is not influenced as it does not contain dynamic information. We use our trained model to compose instantaneous foreground and static background. We show two examples of revealing hidden components in Figure 2.23. In the first example, a user can remove the foreground person by marking on a single frame in video. Our system associates this mask to all the frames in video (shown in Figure 2.22), and edit it to show the region behind the person. We show random frames from an edited video (20 seconds long). In the **middle**-row, a user selects a mask to see the occluded blue-shirt person (behind red shirt person). There is possible because the information was captured by another camera in our multi-view setup. As shown in the **bottom**-row of Figure 2.23, there are ghosting artifacts but one can get a sense

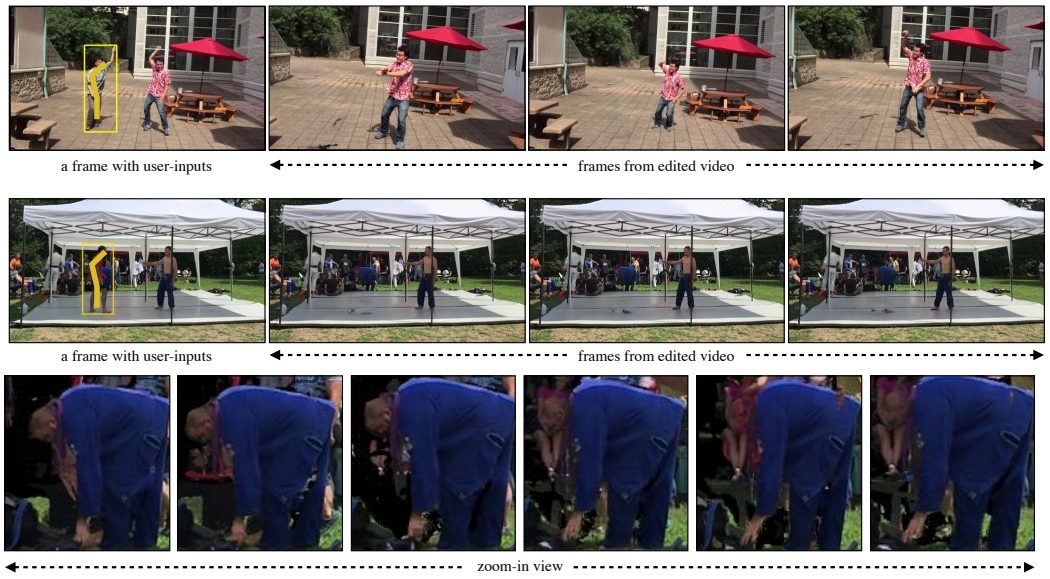


Figure 2.23: **User-Controlled Editing:** We show two examples of user-controlled editing in videos to reveal hidden components. In the **top**-row, a user can remove the foreground person by marking on a single frame in video. The tracking component of our system finds instances of the marked region in the remaining frames of a video (shown in Figure 2.22). We do simple depth and pixel operations to edit the marked region to reveal the region behind the person. We show random frames from an edited video (20 seconds long). In the **middle**-row, a user selects a mask to see the occluded blue-shirt person (behind red shirt person). This is possible since the information was captured by another camera in our multi-view setup. We show a zoom-in view to have a sense of activity of the *disoccluded* blue-shirt person (**bottom**-row). We show frames from 2 seconds of video.

of activity of the *disoccluded* blue-shirt person. We show frames from 2 seconds of video.

2.10 Dense 3D Reconstruction

We use the dense depth volume to get dense 3D point clouds. The depth volume is a rich set of information but it also consists of a substantial amount of noisy 3D points due to sparse cameras, errors in temporal synchronization, estimating camera parameters, or errors in disparity estimation. Our goal is to select good 3D points from this dense depth volume. We use simple multi-view consistency across different views to get good 3D points. One way to do this is to get patch-level similarity across different views in pixel space. However, the matching can be noisy due to the

(a) Dense 3D Reconstruction from COLMAP

(b) Ours (Dense 3D Reconstruction)

Figure 2.24: **Dense 3D Reconstruction:** (**a**) We contrast the dense 3D reconstruction from standard COLMAP [371, 372]. We observe sparse outputs. (**b**) Our approach provides dense 3D reconstruction from 10 views. Crucially, ours is an off-the-shelf method that does not require any scene/time-specific operation. We show multiple views of the 3D point cloud from our approach in Figure 2.25.

difference in illumination and extremely large spatial positioning of different views. We, instead, find consistency of 3D points in depth volume that meet following criteria: (1) we project the depth-volume to a known camera views; (2) for a known camera view, we get an array of depth locations for every pixel location (Shade et al. [380]); (3) for each depth-array, we start from the closest depth value and observe if there are 3D points from other views in its vicinity. In this work, we select a 3D point as a good point if there are 3D points from at least 2 stereo-pairs within a certain threshold. We accumulate good 3D points from all known camera views. Figure 2.24 contrasts the dense 3D point clouds estimated using our approach with standard COLMAP [371, 372]. We also show multiple views using the estimated 3D point cloud from our approach in Figure 2.25.

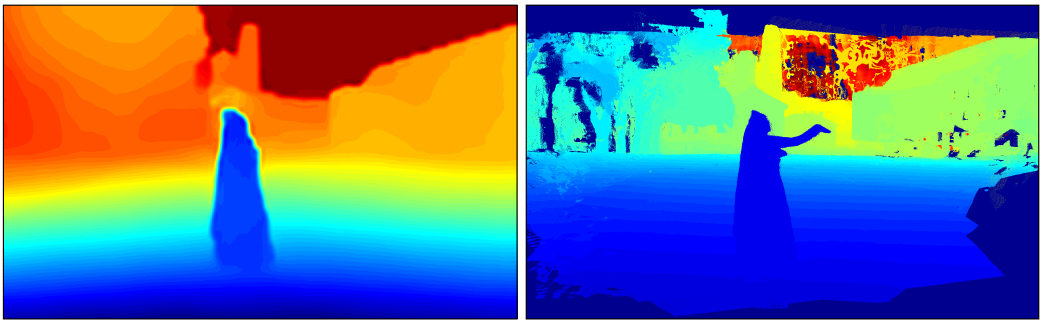
Figure 2.25: We show multiple views using the dense 3D point cloud computed using our approach.

2.11 Why Multi-Views?

Recent approaches have shown view synthesis using a single image [385, 431, 466] or a single video [487]. We present a case study on different scenarios where a single image or a single video might not be sufficient.

Single-Image View Synthesis: We study state-of-the-art for single-image view synthesis [385] to highlight various challenges posed in real-world scenarios. Current approaches work well for generic things like a grass, a road, or non-textured walls etc. They, however, fail for the instances that require exemplar information. We illustrate two challenges in Figure 2.26. (1) **Exemplar information in the scene**. We consider a building or facade with exemplar details instead of being textureless as shown in Figure 2.26. The facade of the building is occluded by the human performer (*close-up*) in the foreground (also showing the *hidden facade*). Synthesizing new views using such a single-image via Shih et al. [385] is not able to capture these exemplar details; (2) **Modeling humans in a scene**. Prior work on single-view view-synthesis [385, 431, 466] either estimate depth or use an off-the-shelf module such as MiDAS [348]. We use the setup of Shih et al. [385] that use MiDAS. Figure 2.26 and Figure 2.27 shows the depth map from a single-image estimated using MiDAS [348]. The depth estimation module lacks detailed 3D information in a single image. Despite trained on a large amount of data, MiDAS is not able to capture

Challenge 1: Unseen Exemplar Information in an Image

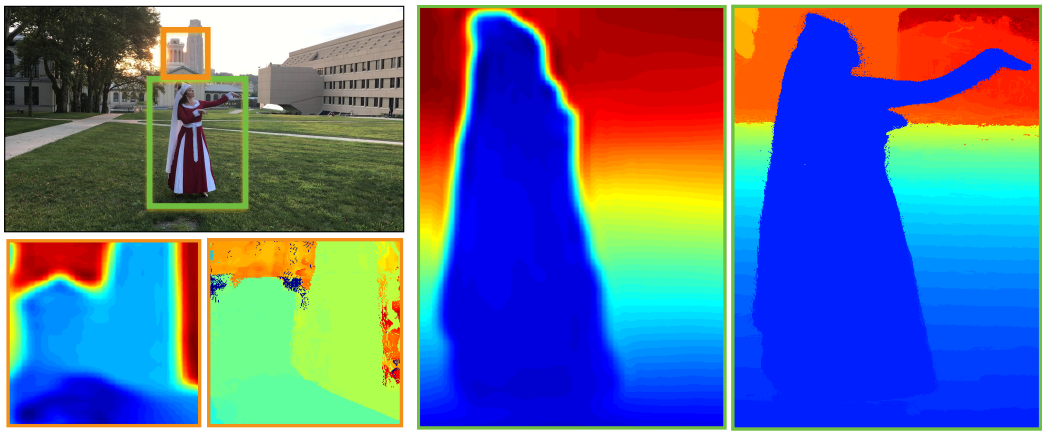


Challenge 2: Depth from a single-view vs. multi-views

Figure 2.26: **Single-Image View Synthesis:** We study state-of-the-art for single-image view synthesis [385] to highlight various challenges posed in real-world scenarios. Current approaches work well for generic things like a grass, a road, or non-textured walls etc. We show two challenges here: **(1)** We consider a building or facade with exemplar details. The facade is hidden by a performer in the foreground. We synthesize new views using Shih et al. [385]. While it is able to capture textureless regions such as grass. They, however, fail for the instances that require **exemplar information** in the scene such as the hidden facade in the above image. Due to the lack of this exemplar information in a single image, Shih et al. [385] generate a *plausible* non-textured output. **(2)** The **depth estimation** from a single-image using MiDAS [348], a current state-of-the-art depth estimation fails to capture the detailed depth information whereas multi-view depth estimation in our approach can capture fine details such as the performer in the image and relative depth between two quite-far away buildings in the background (shown in details in Figure 2.27).

the details belonging to the human performer in the scene. The depth estimation from a single image also fails to capture relative depth between two quite far-away buildings in the background (Figure 2.27). In contrast, the depth from multi-views is able to capture the detailed human body and other components in the scene. Depth estimation is crucial because the errors of depth estimation are propagated to the view synthesis pipeline.

Single-Video View Synthesis: A video captured by a slowly moving camera



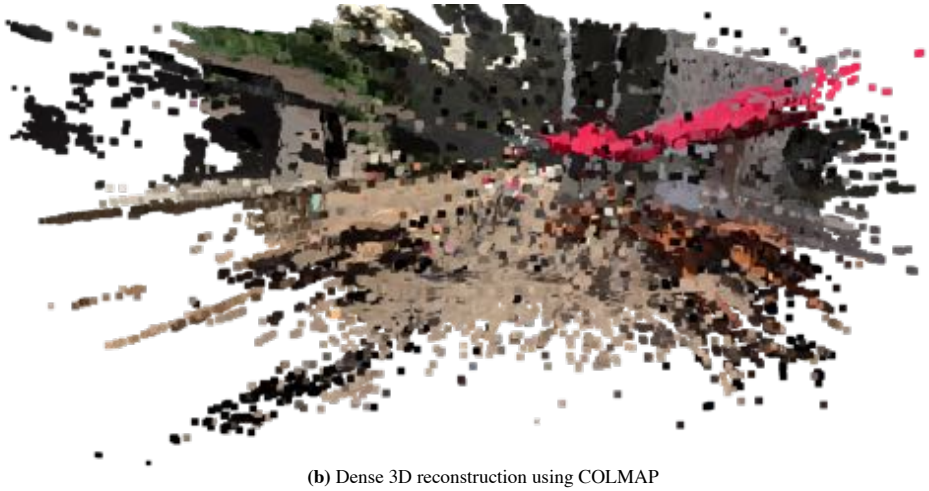
(a) relative depth between two far-away buildings **(b) depth-map of the performer**

Figure 2.27: **Depth from a single-image vs. multi-views:** We contrast the depth estimated from a single image using MiDAS [348] and our approach. We show detailed depth map for: **(a)** relative depth between two quite-far away buildings in the background; and **(b)** the human-performer in the image. The estimated depth from multi-views can capture fine details and relative depth better than single-view depth estimation.

around a dynamic concept is equivalent to dense multi-view view synthesis. Recent approaches have used videos to control space-time of the dynamic events. However, these approaches struggle for various scenarios: (1) **a stationary camera** where it is not possible to get camera poses and the problem reduces to single-view view synthesis; (2) **an unconstrained capture**. Prior work [487] relies on COLMAP for depth estimation to compute a masked foreground and a background. With an unconstrained environment (moving camera and arbitrary events in the scene), this is challenging because 3D reconstruction and depth estimation from COLMAP is quite noisy (as shown in Figure 2.28). The availability of a good depth map is an important assumption in Yoon et al. [487] for the view-synthesis pipeline to operate, which is available in a well-constrained environment; and (3) **seeing dynamic components behind an occlusion** at a given instant of time is challenging from a single-video because it was never captured. The current approaches built on a single-video struggle in a crowded or multi-person scenarios.

Why Multi Views? It is crucial to acknowledge that the different view synthesis approaches (including ours) are interpolating a given data-distribution. The extent of interpolation is currently limited by the information contained in the scene (be it in a single or multi-view images and videos). By design, multi-view videos contain more information and can provide better constraints where other setups struggle.

(a) Random frames (out of 1200 frames) from a video captured by a fast-moving camera



(b) Dense 3D reconstruction using COLMAP

Figure 2.28: **Dense 3D Reconstruction from a Single Video:** We use COLMAP [371, 372] for 3D reconstruction from a single video in an unconstrained environment (arbitrary movement of the camera and the performer). (**a**) We show five random frames from a total of $1,200$ frames in a 2-minutes long video captured at 60fps. (**b**) We use all $1,200$ frames to do a dense 3D reconstruction using COLMAP. We get a noisy 3D reconstruction for this video. The 3D reconstruction from COLMAP is crucial in prior work [487] to get a consistent depth map that is used for view synthesis.

2.12 Limitations of Proposed Representation

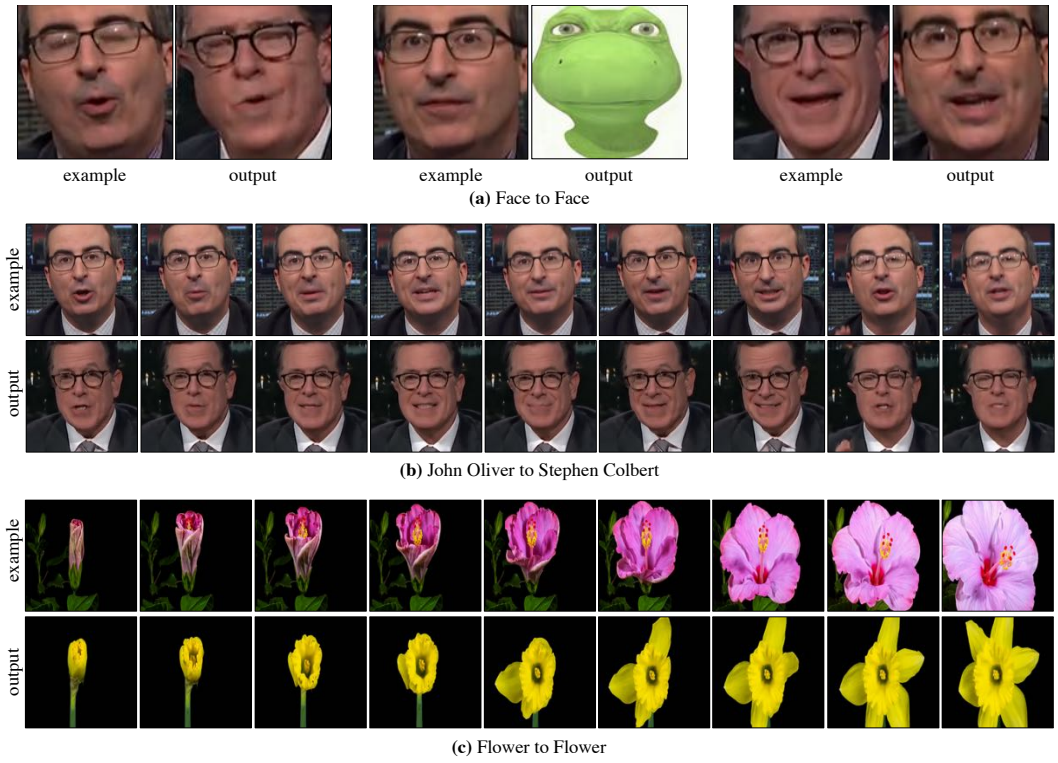
There are many limitations in the current work. We are far-away from the **production quality results** as seen in the bullet-time scene in the movie *Matrix, 1998*. We miss a lot of information on the human face and hands. In a human-event, faces are very important and yet we are missing that information. Our current formulation restricts scalability and development of **user-controllable interfaces** as: (1) we are required to train a new scene-specific model; and (2) explicitly modeling foreground and background requires storing roughly 1TB of meta-data per sequence on the disk. Finally, we observe flickering artifacts in our results. This is primarily due to absence of **temporal constraints** in our formulation that does a per-frame inference. Despite having access to the temporal sequences, we treat them on a per-frame basis.

Part II

Example-based Exploration

Chapter 3

Unsupervised Video Retargeting



(a) Face to Face

(b) John Oliver to Stephen Colbert

(c) Flower to Flower

Figure 3.1: Our approach for video retargeting used for faces and flowers. **a** We show varied examples of face to face translation using our approach. (**b**) We then show translation from John Oliver to Stephen Colbert. (**c**) Finally, the bottom row shows how a synthesized flower follows the blooming process with the input flower. The corresponding videos are available on the project webpage: http://www.cs.cmu.edu/~aayushb/Recycle-GAN/.

3.1 Association and Imagination

Imagine seeing the pink sunset at Cabo de Sao Vicente, our mind can quickly associate it with our previous best at the Orange County and evoke visuals. We, humans, have remarkable abilities to associate different concepts and create visual worlds far beyond what could be seen by a human eye, including inferring the state of the unobserved, imagining the unknown, and thinking about diverse possibilities about what lies in the future. These human powers require minimal instructions and primarily rely on observation and interaction with a dynamic environment. The simple tasks from daily life that are trivial for humans to think and imagine have remained challenging for machine perception and artificial intelligence. The inability to associate and a lack of imagination in machines substantially restricts their applicability. In this work, we present an unsupervised data-driven approach for video retargeting that allows us to: (1) *associate* the contents from two events of the same kind (E.g., two flowers in Figure 3.1-(c)); and (2) create new content (*imagine*) by transferring sequential content from one domain to another while preserving the style of the target domain (E.g., Stephen Colbert *delivering* John Oliver's content in Figure 3.1-(b)). Such a content translation and style preservation task has numerous applications, including human motion and face translation from one person to another, teaching robots from human demonstration, or converting black-and-white videos to color. This work also finds application in creating visual content that is hard to capture or label in real-world settings, E.g., aligning human motion and facial data of two individuals for virtual reality, or labeling night data for a self-driving car. Above all, the notion of content translation and style preservation transcends a pixel-to-pixel operation, into a semantic and human-understandable concepts.

Current approaches for retargeting can be broadly classified into three categories. The first set is designed explicitly for domains such as human faces [54, 419, 420]. While these approaches work well when faces are fully visible, they fail when applied to occluded faces (virtual reality) and lack generalization to other domains. The work on paired image-to-image translation [197] attempts to generalize across domain but requires manual supervision for labeling and alignment. This requirement makes it hard for the use of such approaches as manual alignment or labeling is not possible in many domains. The third category of work attempts unsupervised and unpaired image translation [227, 512]. They enforce a cyclic consistency [504] on unpaired 2D images and learn a transformation from one domain to another. However, unpaired 2D images alone are not sufficient for video retargeting. Firstly, it is not able to pose sufficient constraints on optimization and may lead to a perceptual mode collapse (Figure 3.2-(a)) or get stuck in a bad local minima (Figure 3.2-(b)). The unconstrained optimization makes it hard to generate the required output in the target domain. Secondly, the use of the spatial information alone in 2D images makes it hard to learn the *style* of a particular domain as stylistic information requires temporal knowledge as well.

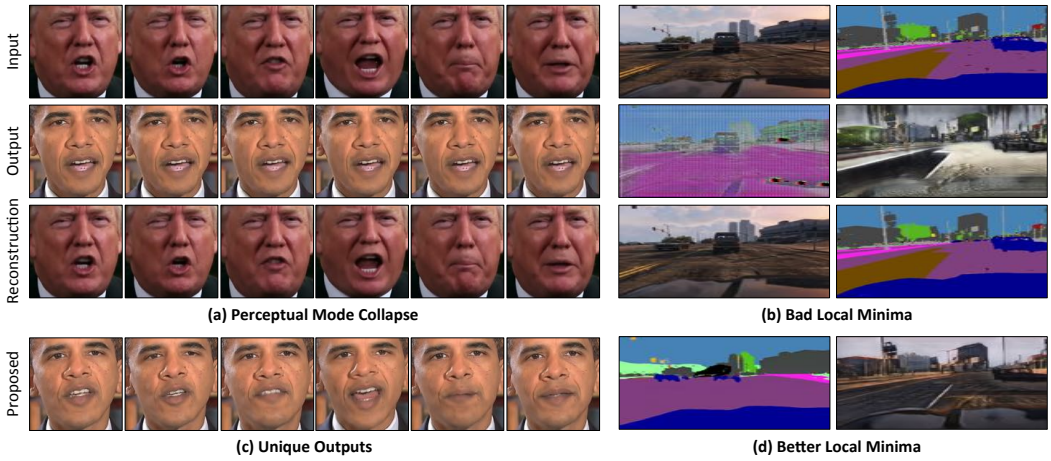


Figure 3.2: **Spatial cycle consistency is not sufficient:** We show two examples illustrating why spatial cycle consistency alone is not sufficient for the optimization. (**a**) We use Cycle-GAN [512] for transferring the contents of Donald Trump to Barack Obama. The first row shows the input of Donald Trump, and the second row shows the output generated. The third row shows the output of reconstruction that takes the second row as input. The second row looks similar despite different inputs, and the third row shows output similar to the first row. On a very close observation, we found that a few pixels in the second row were different (but not perceptually significant), and that was sufficient to get the different reconstruction. We, therefore, termed is as **Perceptual Model Collapse**; (**b**) We show another example for image-to-labels and labels-to-image. While the generator is not able to generate the required output for the given input in both the cases, it is still able to reconstruct the input perfectly. Both examples suggest that the spatial-cyclic-loss is not sufficient to ensure the required output in another domain because the overall optimization reconstructs the input. However, as shown in (c) and (d), we get better outputs with our approach combining the spatial and temporal constraints.

In this work, we make two specific observations: (1) the use of temporal information provides more constraints to the optimization for transforming one domain to the other and helps in reaching a better solution (Figure 3.2-(c)-(d)).; (2) The combined influence of spatial and temporal constraints helps in learning the style characteristic of an identity in a given domain. Figure 3.1-(a) (right) shows an example of capturing subtle details such as a facial dimple on John Oliver's face while smiling. Importantly, temporal information is freely available in videos (available in abundance on the web). Therefore, no manual supervision is required. Figure 3.1 shows different examples for human faces, and flowers using our approach. Without any manual supervision and domain-specific knowledge, our approach learns this *retargeting* from one domain to the other using publicly available video data on

the web from both domains. We refer the reader to our project page for the corresponding videos: http://www.cs.cmu.edu/~aayushb/Recycle-GAN/.
Our contributions : (1) We introduce Recycle-GAN that combines spatial and temporal cues with generative adversarial networks [147] for the problem of video retargeting. We demonstrate the advantages of spatiotemporal constraints over spatial constraints for image-to-labels and labels-to-image in diverse environmental settings. (2) We then present the proposed approach in learning a better association between two domains, and its importance for self-supervised content alignment of the visual data. (3) Inspired by the ever-existing nature of space-time, we qualitatively demonstrate the effectiveness of our approach for various natural processes such as face-to-face translation, flower-to-flower, synthesizing clouds and winds, aligning sunrise and sunset.

3.2 Related Work

A variety of work dealing with image-to-image translation [135, 178, 197, 391, 512] and style translation [47, 127, 183] exists. A large body of work in computer vision and computer graphics is about an image-to-image operation. The initial efforts were on inferring semantic [267], geometric [23, 102], or low-level cues [478]. There is a renewed interest in synthesizing images using data-driven approaches by the introduction of generative adversarial networks [147]. This formulation has been used to generate images from cues such as a low-resolution image [89, 244], class labels [197], and various other input priors [189, 342, 492]. These approaches, however, require an input-output pair to train a model. While it is feasible to label data for a few image-to-image operations, there are numerous tasks for which it is non-trivial to generate input-output pairs for training supervision. Recently, Zhu et al. [512] used the cycle-consistency constraint [504] in an adversarial learning framework to deal with this problem of unpaired data, and demonstrate effective results for various tasks. Cycle-consistency [227, 512] enables many image-to-image translation tasks without any expensive manual labeling. Similar ideas have also found application in learning depth cues in an unsupervised manner [145], machine translation [471], shape correspondences [187], point-wise correspondences [504, 505], or domain adaptation [181].

The variants of Cycle-GAN [512] have been applied to various temporal domains [145, 181]. However, they consider only the spatial information in 2D images and ignore the temporal information for optimization. We observe two major limitations: (1). **Perceptual Mode Collapse:** there are no guarantees that cycle consistency would produce perceptually unique data to the inputs. In Figure 3.2-(a), we use Cycle-GAN [512] for transferring the contents of Donald Trump to Barack Obama. The first row shows the input of Donald Trump, and the second row shows the output generated. The third row shows the output of reconstruction that takes the second row as input. The second row looks similar despite different inputs, and

the third row shows output similar to the first row. On a very close observation, we found that a few pixels in the second row were different (but not perceptually significant), and that was sufficient to get the different reconstruction. We observe a similar behaviour in a different setup (image-to-labels and labels-to-image) shown in Figure 3.2-(b). While the generator is not able to generate the required output for the given input in both the cases, it is still able to reconstruct the input perfectly. In this work, we propose a new formulation that utilizes both spatial and temporal constraints along with the adversarial loss to overcome these challenges. In Figure 3.2-(c, d), we show the outputs generated using the proposed formulation that overcomes perceptual mode collapse. We posit that this is due to more constraints available for an under-constrained optimization; (2). **Tied Spatially to Input:** Zhu et al. [512] use an additional reconstruction loss on the input itself to constrain the optimization. The optimization learns a solution closely tied to the input. This is a reasonable assumption for the problems where only spatial transformation matters, such as transferring texture from horses to zebras, or converting apples to oranges, or pictures to oil paintings. Spatially tying to input is not helpful in our setup for two reasons: (1) the structure of concepts in two domains can be different. E.g, facial structure of two individuals, or two flowers, etc. The additional reconstruction loss does more harm than benefit; and (2) we want to preserve the stylistic information for target domain. We did not use reconstruction loss for these reasons. The spatiotemporal constraints used in our formulation helps us to address these challenges without the reconstruction loss.

The use of GANs [147] and variational auto-encoder [228] have also found a way for synthesizing videos and temporal information. Walker et al. [445] use temporal information to predict future trajectories from a single image. Recent work [168,439,446] used temporal models to predict long term future poses from a single 2D image. MoCoGAN [433] decomposes motion and content to control video generation. Similarly, Temporal GAN [367] employs a temporal generator and an image generator that generates a set of latent variables and image sequences, respectively. This work [367] focuses on predicting future intent from single images at test time or generating videos from a random noise. Concurrently, MoCoGAN [433] shows examples of image-to-video translation using their formulation. Different from these methods, our focus is on a general video-to-video translation where the input video can control the output in a spirit similar to image-to-image translation. To this end, we can generate high-resolution videos of arbitrary length using our approach, whereas prior work [367, 433] generate only 16 frames of 64×64.

Learning Association: Much of computer vision is about learning association, be it learning high-level image classification [362], object relationships [280], or pointwise correspondences [24, 214, 256, 268]. However, there has been relatively little work on learning associations for aligning the content of different videos. In this work, we automatically align visual data without any additional supervision.

Spatial & Temporal Constraints : Spatial and temporal information is known to be

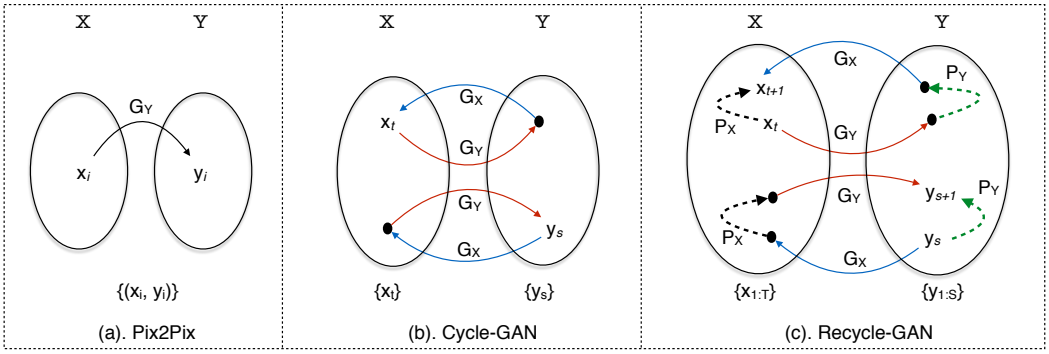


Figure 3.3: We contrast our work with two prominent directions in image-to-image translation. (a) **Pix2Pix** [197]: Paired data is available. A simple function (Eq. 3.1) can be learnt via regression to map $X \rightarrow Y$. (b) **Cycle-GAN** [512]: The data is not paired in this setting. Zhu et al. [512] proposed to use cycle-consistency loss (Eq. 3.3) to deal with the problem of unpaired data. (c) **Recycle-GAN**: The approaches so far have considered independent 2D images only. Suppose we have access to unpaired but *ordered streams* $(x_1, x_2, \ldots, x_t, \ldots)$ and $(y_1, y_2 \ldots, y_s, \ldots)$. We present an approach that combines spatiotemporal constraints (Eq. 3.5). See Section 3.3 for more details.

an integral sensory component that guides human action [137]. There exists a wide literature utilizing these two constraints for various computer vision tasks such as learning better object detectors [296], action recognition [140], etc. In this work, we take a first step to exploit spatiotemporal constraints for video retargeting and unpaired image-to-image translation.

3.3 Recycle-GAN

Assume we wish to learn a mapping $G_Y : X \rightarrow Y$. The classic approach tunes G_Y to minimize reconstruction error on paired data samples $\{(x_i, y_i)\}$ where $x_i \in X$ and $y_i \in Y$:

$$\min_{G_Y} \sum_i ||y_i - G_Y(x_i)||^2. \tag{3.1}$$

Adversarial loss: Recent work [147,197] has shown that one can improve the learned mapping by tuning it with a discriminator D_Y that is adversarially trained to distinguish between real samples of y from generated samples $G_Y(x)$:

$$\min_{G_Y} \max_{D_Y} L_g(G_Y, D_Y) = \sum_s \log D_Y(y_s) + \sum_t \log(1 - D_Y(G_Y(x_t))), \tag{3.2}$$

Importantly, we use a formulation that does *not* require paired data and only requires access to individual samples $\{x_t\}$ and $\{y_s\}$, where different subscripts are used to emphasize the lack of pairing.

Cycle loss: Zhu et al. [512] use cycle consistency [504] to define a reconstruction loss when the pairs are not available. Popularly known as Cycle-GAN (Fig. 3.3-b), the objective can be written as:

$$L_c(G_X, G_Y) = \sum_t ||x_t - G_X(G_Y(x_t))||^2. \tag{3.3}$$

Recurrent loss: We have so far considered the setting when static data is available. Instead, assume that we have access to unpaired but *ordered streams* $(x_1, x_2, \ldots, x_t, \ldots)$ and $(y_1, y_2 \ldots, y_s, \ldots)$. Our motivating application is learning a mapping between two videos from different domains. One option is to ignore the stream indices, and treat the data as an unpaired *and unordered* collection of samples from X and Y (e.g., learn mappings between shuffled video frames). We demonstrate that much better mapping can be learnt by taking advantage of the temporal ordering. To describe our approach, we first introduce a recurrent temporal predictor P_X that is trained to predict future samples in a stream given its past:

$$L_\tau(P_X) = \sum_t ||x_{t+1} - P_X(x_{1:t})||^2, \tag{3.4}$$

where we write $x_{1:t} = (x_1 \ldots x_t)$.

Recycle loss: We use this temporal prediction model to define a new cycle loss across domains and *time* (Fig. 3.3-c) which we refer as a recycle loss:

$$L_r(G_X, G_Y, P_Y) = \sum_t ||x_{t+1} - G_X(P_Y(G_Y(x_{1:t})))||^2, \tag{3.5}$$

where $G_Y(x_{1:t}) = (G_Y(x_1), \ldots, G_Y(x_t))$. Intuitively, the above loss requires *sequences* of frames to map back to themselves. We demonstrate that this is a much richer constraint when learning from unpaired data streams in Figure 3.4 and Figure 3.5.

Recycle-GAN: We now combine the recurrent loss, recycle loss, and adversarial loss into our final Recycle-GAN formulation:

$$\min_{G,P} \max_D L_{rg}(G, P, D) = L_g(G_X, D_X) + L_g(G_Y, D_Y) +$$
$$\lambda_{rx} L_r(G_X, G_Y, P_Y) + \lambda_{ry} L_r(G_Y, G_X, P_X) + \lambda_{\tau x} L_\tau(P_X) + \lambda_{\tau y} L_\tau(P_Y).$$

Inference: At test time, given an input video with frames $\{x_t\}$, we would like to generate an output video. The simplest strategy would be directly using the trained

G_Y to generate a video frame-by-frame $y_t = G_Y(x_t)$. Alternatively, one could use the temporal predictor P_Y to smooth the output:

$$y_t = \frac{G_Y(x_t) + P_Y(G_Y(x_{1:t-1}))}{2},$$

where the linear combination could be replaced with a nonlinear function, possibly learned with the original objective function. However, for simplicity, we produce an output video by simple single-frame generation. This allows our framework to be applied to both videos and single images at test-time, and produces fairer comparison to spatial approach.

Implementation Details: We adopt training details from Cycle-GAN [512] to train our spatial translation model, and Pix2Pix [197] for our temporal prediction model. The generative network consists of two convolution (downsampling with stride-2), six residual blocks, and finally two upsampling convolution (each with a stride 0.5). We use the same network architecture for G_X, and G_Y. The resolution of the images for all the experiments is set to 256×256. The discriminator network is a 70×70 PatchGAN [197,512] that is used to classify a 70×70 image patch if it is real or fake. We set all $\lambda_s = 10$. To implement our temporal predictors P_X and P_Y, we concatenate the last two frames as input to a network whose architecture is identical to U-Net architecture [197,359].

3.4 Quantitative Analysis

We now study the influence of spatiotemporal constraints over spatial-cyclic constraints. Since our crucial technical contribution is the introduction of temporal constraints in learning unpaired image mappings, the natural baseline is Cycle-GAN [512]. Cycle-GAN is a widely adopted approach for exploiting spatial-cyclic consistency alone for an unpaired image translation. We first present quantitative results on domains where ground-truth correspondence between input and output videos are known (e.g., a video where each frame has a semantic label map). Importantly, this correspondence pairing is *not available* to either Cycle-GAN or Recycle-GAN but used only for evaluation. We then present qualitative results on a diverse set of videos with unknown correspondences, including video translations across different human faces and temporally-intricate events found in nature (flowers blooming, sunrise/sunset, time-lapsed weather progressions).

3.4.1 Quantitative Analysis

We use publicly available Viper [355] dataset for image-to-labels and labels-to-image to evaluate our findings. This dataset is collected using computer games with varying realistic content and provides densely annotated pixel-level labels. Out of the 77 different video sequences consisting of diverse environmental conditions, we use 57

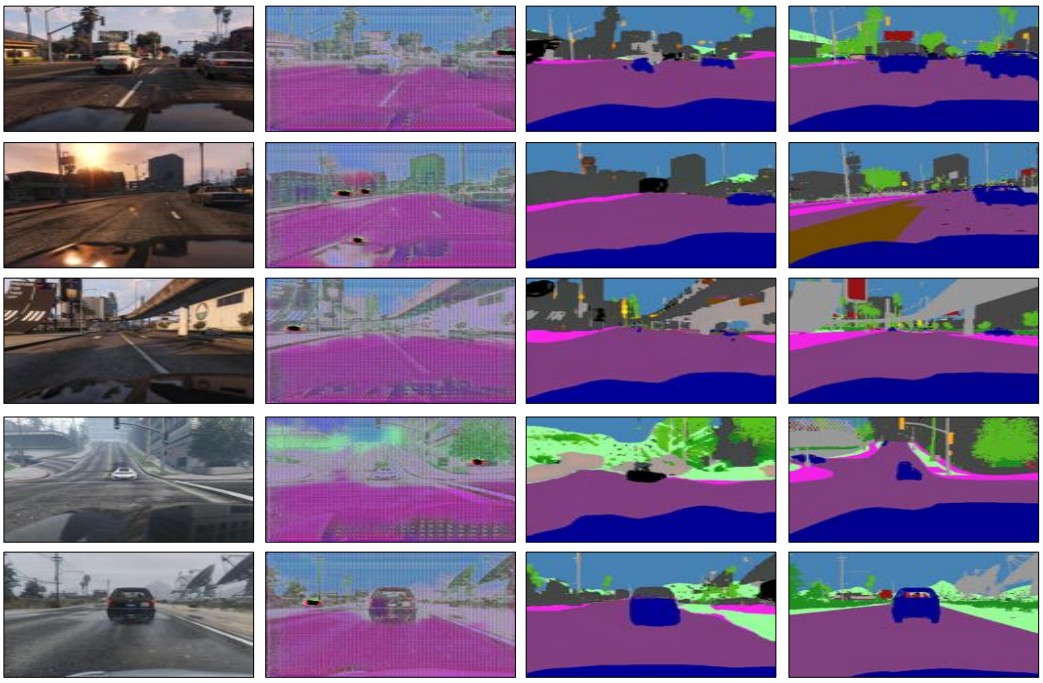


Figure 3.4: We contrast our approach (Recycle-GAN) with Cycle-GAN [512] for predicting semantic labels from images on a held-out data of Viper dataset [355] for various environmental conditions.

sequences for training our model and baseline. We use the held-out 20 sequences for evaluation. The goal for this evaluation is not to achieve the state-of-the-art performance but to compare and understand the advantage of spatiotemporal cyclic consistency over the spatial cyclic consistency [512].

While the prior works [197, 512] have mostly used Cityscapes dataset [80], we could not use it for our evaluation. Primarily the labeled images in Cityscapes are not continuous video sequences, i.e., consecutive frame pairs are quite different from other another. It is not trivial to use a temporal predictor. We used Viper as a proxy for Cityscapes because the task is similar, and that dataset contains dense video annotations. Additionally, a concurrent work [30] on unsupervised video-to-video translation also uses Viper dataset for evaluation. However, they restrict their experiments to a small subset of sequences from the daylight category. In this work, we use all the environmental conditions available in the dataset.

Image to Labels : In this setting, we input an RGB image to the generator that output segmentation label maps. We compute three metrics to compare the output of two approaches: (1). Mean Pixel Accuracy (**MP**); (2). Average Class Accuracy (**AC**); (3). Intersection over Union (**IoU**). We compute these statistics using the ground

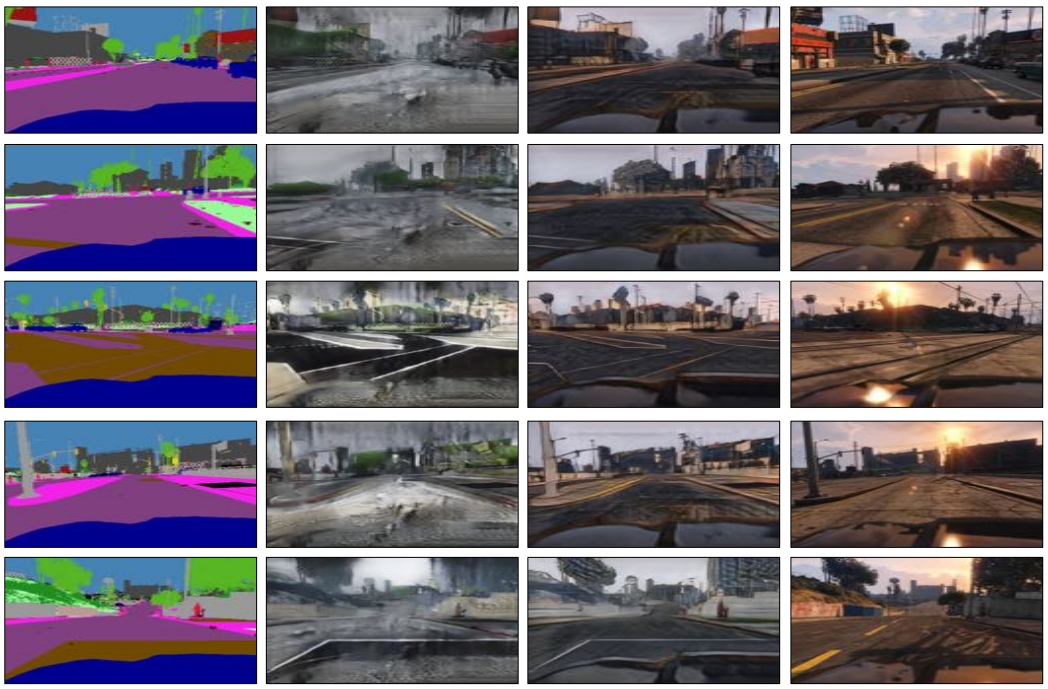


Figure 3.5: We contrast our approach(Recycle-GAN) with Cycle-GAN [512] for the task of generating images from label map on a held-out data of Viper dataset [355].

truth for the held-out sequences from different environmental conditions. Table 3.1 contrast the performance of our approach (Recycle-GAN) with Cycle-GAN. We observe that Recycle-GAN achieves better performance than Cycle-GAN, and combining both losses further improved it. Figure 3.4 qualitatively compares our approach with Cycle-GAN.

Labels to Image : In this setting, we input a segmentation label map to the generator and output an image that is close to a real image. The goal of this evaluation is to compare the quality of output images obtained from both approaches. We follow Pix2Pix [197] for this evaluation. We use the generated images from each of the algorithms with a pre-trained FCN model. We then compute the performance of synthesized images against the real images to compute a normalized FCN-score. Higher performance on this criterion suggests that the generated image is closer to the real images. Table 3.2 compares the performance of our approach with Cycle-GAN. We observe that Recycle-loss does better than Cycle-loss, and combining both the losses led to significantly better outputs. Figure 3.5 qualitatively compares our approach with Cycle-GAN.

In these experiments, we make two observations: (i) Cycle-GAN learns a good translation model within a few initial iterations (seeing only a few examples). This

Criterion	Approach	day	sunset	rain	snow	night	all
MP	Cycle-GAN	35.8	38.9	51.2	31.8	27.4	35.5
	Recycle-GAN	**48.7**	**71.0**	**60.9**	57.1	**45.2**	**56.0**
	Combined	**48.7**	70.0	60.1	**58.9**	33.7	53.7
AC	Cycle-GAN	7.8	6.7	7.4	7.0	4.7	7.1
	Recycle-GAN	11.9	12.2	**10.5**	11.1	**6.5**	11.3
	Combined	**12.6**	**13.2**	10.1	**13.3**	5.9	**12.4**
IoU	Cycle-GAN	4.9	3.9	4.9	4.0	2.2	4.2
	Recycle-GAN	7.9	9.6	7.1	8.2	**4.1**	8.2
	Combined	**8.5**	**13.2**	**10.1**	**9.6**	3.1	**8.9**

Table 3.1: **Image to Labels (Semantic Segmentation):** We use the Viper [355] dataset to evaluate the performance improvement when using spatiotemporal constraints as opposed to only spatial cyclic consistency [512]. We report results using three criteria: (1). Mean Pixel Accuracy (**MP**); (2). Average Class Accuracy (**AC**); and (3). Intersection over union (**IoU**). We observe that our approach achieves better performance than prior work, and combining both leads to further better performance.

Approach	day	sunset	rain	snow	night	all
Cycle-GAN	0.33	0.27	0.39	0.29	0.37	0.30
Recycle-GAN	0.33	0.51	0.37	0.43	0.40	0.39
Combined	**0.42**	**0.61**	**0.45**	**0.54**	**0.49**	**0.48**

Table 3.2: **Normalized FCN score for Labels-to-Image:** We use a pre-trained FCN-style model to evaluate the quality of synthesized images over real images using the Viper [355] dataset. Higher performance on this criteria suggest that the output of a particular approach produces images that look closer to the real images.

model, however, degraded as reconstruction loss started to decrease. We believe that minimizing reconstruction loss alone on input leads it to a bad local minima. Our formulation provided more constraints and led it to a better solution; (ii) CycleGAN learns a better translation model for Cityscapes as opposed to Viper.
Cityscapes consists of images from mostly daylight and agreeable weather. Viper has a vast and varied distribution of different illumination (day, night) and weather conditions (snow, rain). This diversity makes it harder to learn a useful mapping because there are potentially many output images for a labeled input. We find that standard conditional GANs suffer from mode collapse in such scenarios and produce "average" outputs (as pointed by prior works [24]). Our experiments suggest

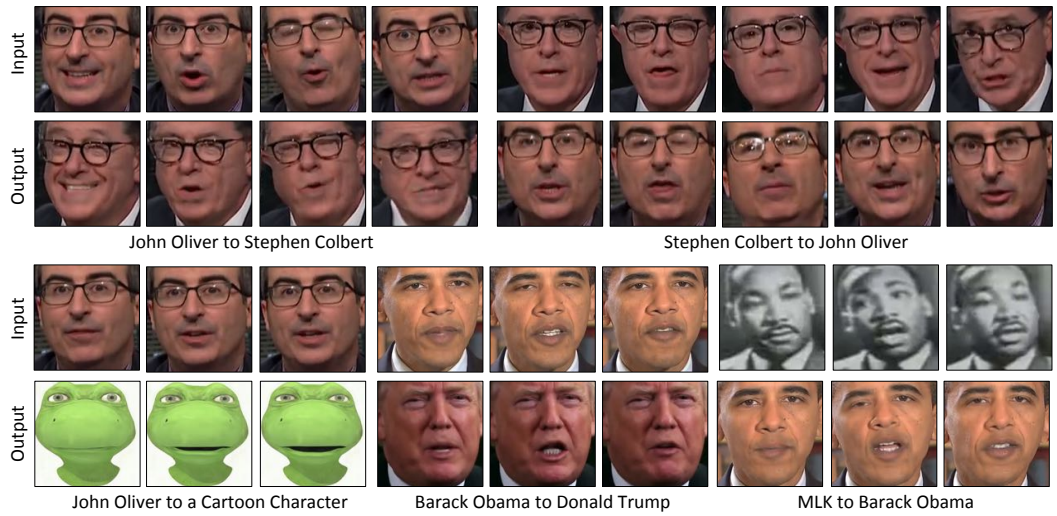


Figure 3.6: **Face to Face:** The top row shows multiple examples of face to face between John Oliver and Stephen Colbert using our approach. The bottom row shows an example of translation from John Oliver to a cartoon character, Barack Obama to Donald Trump, and Martin Luther King Jr. (MLK) to Barack Obama. Without any input alignment or manual supervision, our approach could capture stylistic expressions for these public figures. As an example, John Oliver's dimple while smiling, the shape of mouth characteristic of Donald Trump, and the facial mouth lines and smile of Stephen Colbert.

that having more constraints help ameliorate such challenging translation problems.

3.4.2 Human Studies

We perform human studies on the synthesized output, particularly faces and flowers, following the protocol of MoCoGAN [433], who also evaluated videos. Our analysis consists of three parts: (**1**) We show the synthesized videos individually from both Cycle-GAN and ours to 15 sequestered human subjects. We asked them if it is a real video or a generated video. The subjects misclassified 28.3% times generated videos from our approach as real, and 7.3% times for Cycle-GAN. (**2**) We show the synthesized videos from Cycle-GAN and our approach simultaneously. We then asked them to tell which one looks more natural and realistic. Human subjects chose the videos synthesized from our approach 76% times, 8% times Cycle-GAN, and 16% times they were confused. (**3**) We show the video-to-video translation. This setup is an extension of (2), except we also include input now. We then asked which translation looks more realistic and natural. The human subjects selected our approach 74.7% times, 13.3% times they picked Cycle-GAN, and

"We shall fight on the beaches, we shall fight on the landing grounds .." - Winston Churchill

Figure 3.7: **Winston Churchill in his own words:** One of the popular speeches of Winston Churchill was never visually recorded – "We shall go on to the end. We shall fight in France ... We shall fight on the beaches, we shall fight on the landing grounds..". We were given a short 3 second clip of Winston Churchill that was 180×180 in resolution. Given an actor saying the same speech, we could create the visuals of Winston Churchill saying his own speech. We show a few random frames from synthesized visuals. We refer the reader to the project page for the entire sequence. **Courtesy:** Naomi Austin and BBC Studios, London.

12% times they were confused. From the human study, we observe that combining spatial and temporal constraints lead to better retargeting.

3.5 Qualitative Analysis

3.5.1 Face to Face

We use publicly available videos of various public figures for face to face translation. Figure 3.6 shows an example of face to face translation between John Oliver and Stephen Colbert, Barack Obama to Donald Trump, and Martin Luther King Jr. (MLK) to Barack Obama, and John Oliver to a cartoon character. Without any supervisory signal or manual alignment, our approach learns face-to-face translation. It captures stylistic expression for these personalities, such as the dimple on the face of John Oliver while smiling, the characteristic shape of mouth of Donald Trump, and the mouth lines for Stephen Colbert.

Winston Churchill in his own words: Our approach also allows us to create the visuals of Winston Churchill's speech that were never visually recorded. One example is the popular speech at the beginning of the second world war – 'We shall go on to the end. We shall fight in France ... We shall fight on the beaches, we shall fight on the landing grounds..". We were given a short 3 second clip of Winston Churchill that was 180×180 in resolution (courtesy of BBC Studios, London). Given an actor saying the same speech, we could create the visuals of Winston Churchill saying his own speech. Figure 3.7 show a few random frames from synthesized visuals.

Figure 3.8: **Resurrecting Marshal Philippe Petain:** Our approach used for creating conversation with Marshal Phillippe Petain. We had access to only 10 second video (both low-res and black-and-white) of Philippe Petain during the second world war. Our approach retargets the facial animation from example (**top-row**) to the target to create new outputs (**bottom-row**). **Courtesy:** Thierry Ardisson.

Resurrecting Marshal Philippe Petain: Our approach used for creating **television-shows** where a host is shown to interact with a historical character. A major challenge is to use old video clips of these historical characters and create new content. Such video clips are usually black-and-white and low-res. Importantly, the quality of the video clips vary as we move across time (i.e., the videos in 1950's look completely different from videos in 1980's). Our approach is an example of test-time training that is agnostic of the statistics of the visual content, and therefore allows us to go beyond the limitations of fixed datasets (usually observed in supervised learning). Figure 3.8 shows an example of our approach used for creating conversation with Marshal Phillippe Petain. We had access to only 10 second video (both low-res and black-and-white) of Philippe Petain during the second world war. Our approach can create facial animation of the output similar to the provided example without any annotations. **There are also limitations of using a restricted data distribution** – In the given 10 seconds of Phillippe Petain, he did not blink or closed the eye. As a result, the output **fails to capture closing eyes**.

Don't believe anything you see or hear: In an another example shown in Fig-

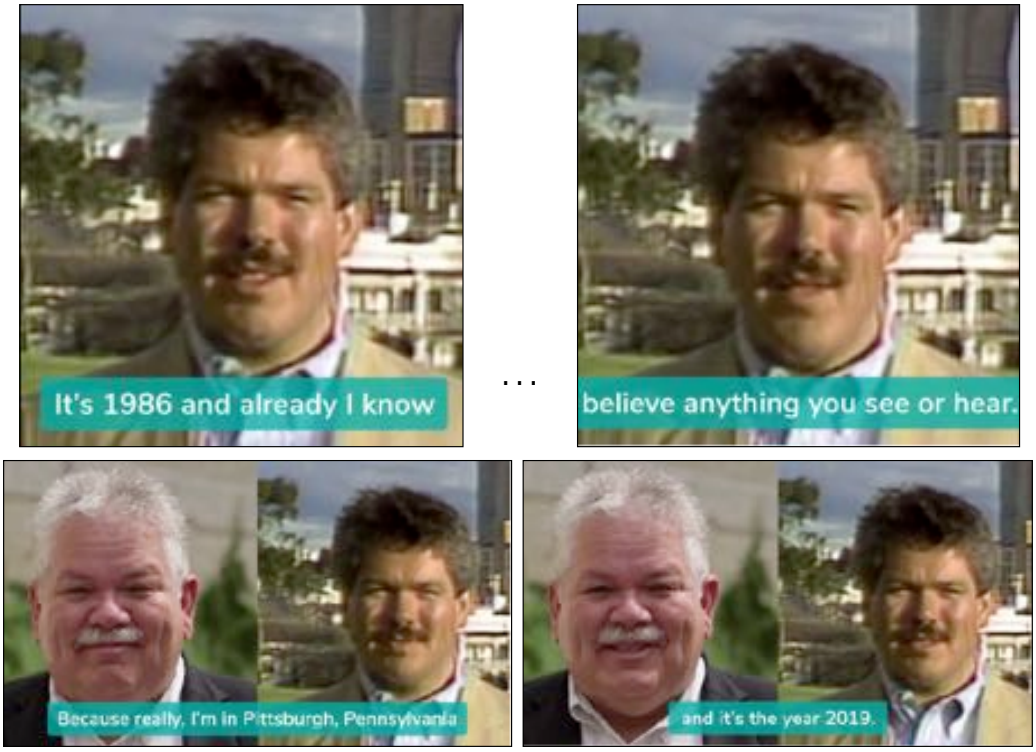


Figure 3.9: **Don't believe anything you see or hear:** Old Rick Sebak (Pittsburgh-based television producer) asked one day if he could make his younger self say something. Shown here is our approach used for creating content given the old video (1986) of Rick Sebak in Australia. **Courtesy:** Rick Sebak and WQED.

ure 3.9, our approach (without any modification) is used for creating new visuals using a short video clip (20 second) of Rick Sebak (a popular American Television producer based in Pittsburgh). The video was captured in Australia in 1986. We show an educational message using this clip (courtesy of WQED, Pittsburgh) – "It's springtime in Australia. It's 1986 and already I know that in 35 or 40 years, it's going to be so easy to manipulate video that it will be wise not to believe anything you see or hear. Because really, I am in Pittsburgh, Pennsylvania and it's the year 2019". While videos from any source are not considered as an evidence in the court of law, they are still important in forming public opinion on social media such as Twitter, Facebook, and YouTube. Every time a viewer see a generated video on these platforms, there needs to be a banner displaying "This is a generated content", so that a viewer does not make any opinions out of it. We discuss it in details in the next chapter and address some of the challenges posed due to audio-visual fakes.

Figure 3.10: **Flower to Flower:** We shows two examples of flower-to-flower translation. Note the smooth transition from Left to Right.

3.5.2 Flower to Flower

We demonstrate our approach for flowers via time-lapse videos. These time-lapses show the blooming of different flowers but without any sync. We use our method to align the content, i.e., both flowers bloom or die together. Figure 3.10 shows the output of our approach to learn an association between the events of different flowers' life.

3.5.3 Clouds & Wind Synthesis

Our approach can synthesize a new video that has the required weather condition such as clouds and wind without the need for recapturing. We use the given video and video data from the target environmental condition as two domains in our experiment. The conditional video and trained translation model generates the required output.

We collected the video data for various wind and cloud conditions, such as a calm day or windy day for this experiment. We convert a calm-day to a windy-day, and a windy-day to a calm-day using our approach. We did not modify the aesthetics of the place in this process. Figure 3.11 shows an example of synthesizing clouds and winds on a windy day when the only information available was a video captured at the same place with a light breeze.

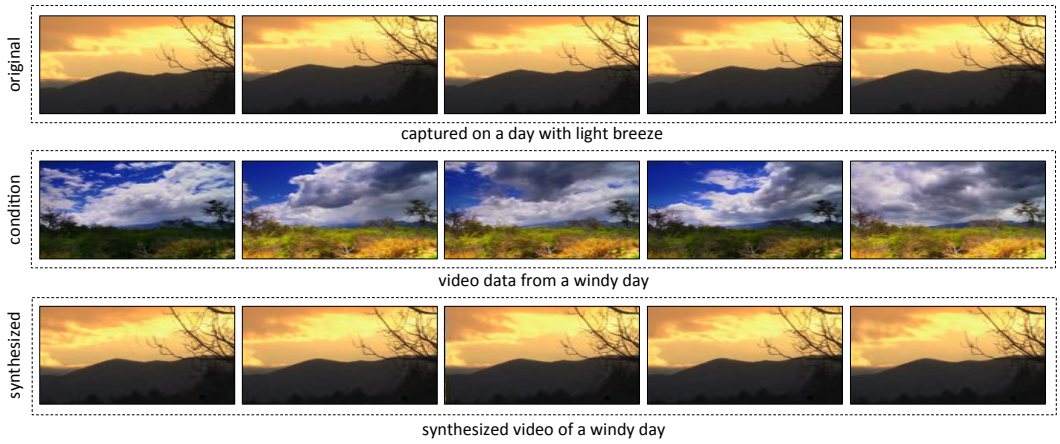


Figure 3.11: **Synthesizing Clouds & Winds:** We use our approach to synthesize clouds and winds. The top row shows example frames of a video captured on a day with light breeze. We condition it on video data from a windy-day video (shown in second row) by learning a transformation between two domains using our approach. The last row shows the output synthesized video with the clouds and trees moving faster (giving a notion of wind blowing). See videos on the project page for better visualization.

3.5.4 Sunrise & Sunset

We show the usefulness of our approach for both video manipulation and content alignment on sunrise and sunset data. This setup is similar to our experiments on clouds and wind synthesis. Figure 3.12 shows an example of synthesizing a sunrise video from an original sunset video by conditioning it on a random sunrise video. We also show examples of the alignment of many sunrise and sunset scenes.
Note: We refer the reader to our project webpage for different videos synthesized using our approach: `http://www.cs.cmu.edu/~aayushb/Recycle-GAN/`.

3.6 Limitations: Association beyond Data Distribution

We show an example of transformation from a real bird to an origami bird to demonstrate a case where our approach failed to learn the association. We extract the real bird data using web videos. We use the origami bird from the synthesis of Kholgade et al. [225]. Figure 3.13 shows the synthesis of the origami bird conditioned on the real bird. While the real bird is sitting, the origami bird stays and attempts to imitate the actions of the real bird. The problem comes when the bird begins to fly. The first frames when the bird starts to fly are elegant. After some time, the origami bird reappears. From an association perspective, the origami bird should not have appeared. Looking back at the training data, we found that the original

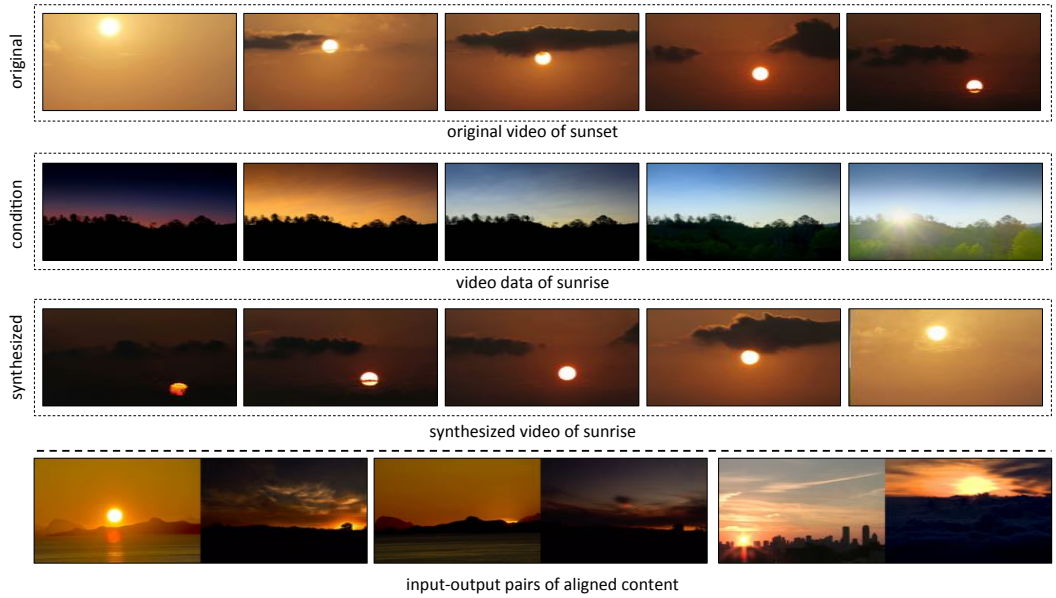


Figure 3.12: **Sunrise & Sunset:** We use our approach to manipulate and align the videos of sunrise and sunset. The top row shows example frames from a sunset video. We condition it on video data of sunrise (shown in second row) by learning a transformation between two domains using our approach. The third row shows example frames of new synthesized video of sunrise. Finally, the last row shows random examples of input-output pair from different sunrise and sunset videos.

origami bird data does not have an example of a frame without the origami bird. Our approach is not able to associate an example when the real bird is no longer visible. We observe that our method could only learn to interpolate over a given data distribution and fails to capture anything drastically beyond it. A possible way to address this problem is by using a lot of training data so that it encapsulates all possible scenarios for a useful interpolation.

3.6.1 Discussion

In this work, we explore the influence of spatiotemporal constraints in learning video retargeting and image translation. Unpaired video/image translation is a challenging task because it is unsupervised, and lacks any correspondences between training samples from the input and output space. We point out that many natural visual signals are inherently spatiotemporal, which provides strong temporal constraints for free. These constraints result in significantly better mappings. We also point out that unpaired and unsupervised video retargeting and image translation is an under-constrained problem. More constraints using auxiliary tasks from

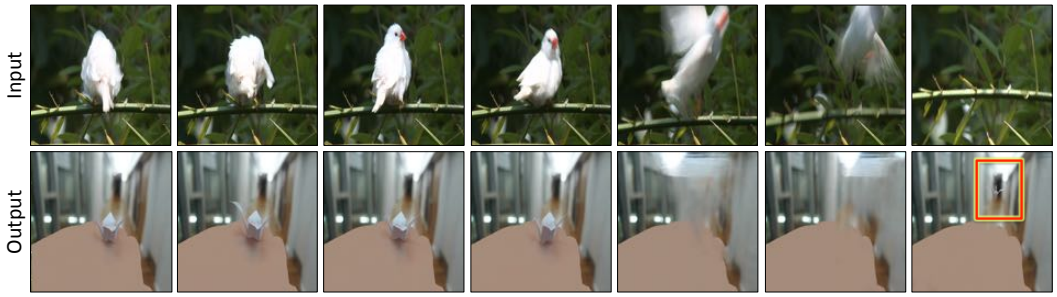


Figure 3.13: **Association beyond Data Distribution:** We present the failure in association/synthesis for our approach using a transformation from a *real* bird to an *origami* bird. While the origami bird (output) is trying to imitate the real bird (input) when it is sitting (Column 1 - 4), and also flies away when the real bird flies (Column 5 - 6). We observe that it reappears after sometime (red bounding box in Column 7) in a flying mode while the real bird didn't exist in the input. Our algorithm is not able to make transition of association when the real bird is completely invisible, and so it generated a random flying origami.

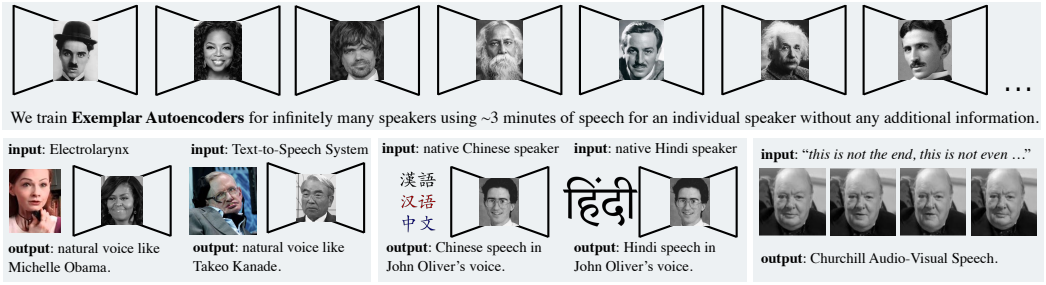
the visual data itself (as used for other vision tasks [288,503]) could help in learning better transformation models.

Recycle-GANs learn both a mapping function and a recurrent temporal predictor. Thus far, our results make use of only the mapping function, to facilitate fair comparisons with previous work. But it is natural to synthesize target videos by making use of both the single-image translation model and the temporal predictor. The style in video retargeting can be incorporated more precisely by using spatiotemporal generative models. These would even allow to modulate the speed of generated output. E.g., Two people may have different ways of content delivery and that one person can take longer than others to say the same thing. A right notion of style should be able to generate even this variation in the time required for delivering speech/content. We believe that better spatiotemporal neural network architecture could attempt this problem.

Chapter 4

Unsupervised Audio-Visual Synthesis

Imagine waking up one day without your voice and not able to express yourself. A survey by NIH (NIDCD) found that approximately 7.5M people in the United States have trouble using their voices. There are estimated $50,000 - 60,000$ *laryngectomees in the US.*



(a) Assistive Tool for Speech Impaired: Zero-Shot Natural Voice Synthesis

(b) Beyond Language Constraints: Zero-Shot Multi-Lingual Translation

(c) Educational/Entertainment Purposes: Zero-Shot Audio-Visual Content Creation

Figure 4.1: We train audio-visual (AV) exemplar autoencoders that capture personalized *in-the-wild* web speech as shown in the **top**-row. We show three representative applications of these Exemplar Autoencoders: (**a**) Our approach enables zero-shot natural voice synthesis from an Electrolarynx or a TTS used by a speech-impaired person; (**b**) Without any knowledge of Chinese and Hindi, our approach can generate Chinese and Hindi speech for John Oliver, an English-speaking late-night show host; and (**c**) We can generate audio-visual content for historical documents that could not be otherwise captured.

4.1 Audio-Visual Synthesis

We present an unsupervised approach to audio-visual synthesis that generates the audio and visual stream of a known target speaker from the speech input of any unknown speaker. Our approach allows the generated data to capture subtle properties of the target speaker including: (1) the scene-context, such as the ambient appearance and acoustics of the environment (e.g., conference room or large public convention); and (2) stylistic prosody of the particular speech (e.g., a "happy" vs. "angry" delivery). Our approach enables a variety of novel applications because it generalizes via a reduced dependence on training data; it is trained on only a few minutes of target speech, and does not require any training on the input speaker. By applying input speech from never-before-encountered languages, our approach enables stylistic multilingual translations. By applying input speech generated by medical devices such as electrolarynxs and text-to-speech (TTS) systems, our approach enables personalized voice generation for voice-impaired individuals [207, 305]. Our work also enables applications in education and entertainment; one can create interactive documentaries about historical figures in their voice, or generate the sound of actors who are no longer able to perform. We highlight such representative applications in Figure 4.1.

Prior work typically independently looks at the problem of audio conversion [74, 216,217,299,341] and video generation from audio signals [77,411,502,509]. Particularly relevant are zero-shot audio translation approaches [74,299,335,341] that learn a *generic* low-dimensional embedding (from a training set) that are designed to be agnostic to speaker identity (Fig. 4.2-a). We will empirically show that such generic embeddings may struggle to capture stylistic details of in-the-wild speech that differs from the training set. Alternatively, one can directly learn an audio translation engine *specialized* to specific input and output speakers, often requiring data of the two speakers either aligned/paired [67, 306, 407, 425] or unaligned/unpaired [75, 111, 209, 215–217, 379]. This requirement restricts such methods to known input speakers at test time (Fig. 4.2-b). In terms of video synthesis from audio input, zero-shot facial synthesis approaches [77, 502, 509] animate the lips but struggle to capture realistic facial characteristics of the entire person. Other approaches [139, 386, 411] restrict themselves to known input speakers at test time and require large amounts of data to train a model in a supervised manner.

Our work combines the zero-shot nature of generic embeddings with the stylistic detail of person-specific translation systems. Simply put, given a target speech with a particular style and ambient environment, we learn an *autoencoder specific to that target speech* (Fig. 4.2-c). We deem our approach "Exemplar Autoencoders". At test time, we demonstrate that one can translate any input speech into the target simply by passing it through the target Exemplar Autoencoder. We demonstrate this property is a consequence of two curious facts, shown in Fig. 4.3: (1) linguistic phonemes tend to cluster quite well in spectrogram space (Fig. 4.3-a); and

(2) autoencoders with sufficiently small bottlenecks act as projection operators that project out-of-sample source data onto the target training distribution, allowing us to preserve the content (words) of the source and the style of the target (Fig. 4.3-c). we jointly synthesize audio-visual (AV) outputs by adding a visual stream to the audio autoencoder. Importantly, our approach is data-efficient and can be trained using 3 minutes of audio-video data of the target speaker and *no training data* for the input speaker. The ability to train Exemplar Autoencoders on small amounts of data is crucial when learning specialized models tailored to particular target data. Table 4.1 contrasts our work with leading approaches in audio conversion [218,341] and audio-to-video synthesis [77, 411].

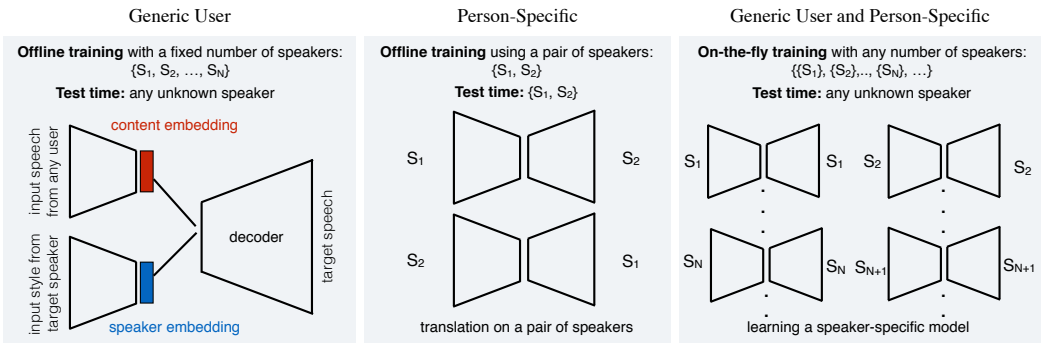
Contributions: (1) We introduce Exemplar Autoencoders, which allow for any input speech to be converted into an arbitrarily-large number of target speakers ("any-to-many" AV synthesis). (2) We move beyond well-curated datasets and work with in-the-wild web audio-video data in this work. We also provide a new CelebAudio dataset for evaluation. (3) Our approach can be used as an off-the-shelf plug and play tool for target-specific voice conversion. We provide an online demo that shows the usefulness of our approach. Finally, to the best of our knowledge, *we are the first to show AV synthesis from an audio signal.*

4.2 Review of Audio-Visual Synthesis

A tremendous interest in audio-video generation for health-care, quality-of-life improvement, educational, and entertainment purposes has influenced a wide variety of work in audio, natural language processing, computer vision, and graphics literature. In this work, we seek to explore a standard representation for a user-controllable "any-to-many" audio-visual synthesis.

Speech Synthesis & Voice Conversion: Earlier works [191,491] in speech synthesis use text inputs to create Text-to-Speech (TTS) systems. Sequence-to-sequence (Seq2seq) structures [410] have led to significant advancements in TTS systems [320, 384, 458]. Recent works [201] have extended these models to incorporate multiple speakers. These approaches enable audio conversion by generating text from the input via a speech-to-text (STT), and use a TTS for target audio. Despite enormous progress in building TTS systems, it is not trivial to embody perfect emotion and prosody due to the limited expressiveness of bare text [333]. In this work, we use the raw speech signal of a target speech to encode the stylistic nuance and subtlety that we wish to synthesize.

Audio-to-Audio Conversion: The problem of audio-to-audio conversion has largely been confined to one-to-one translation, be it using a paired [67,306,407,425] or unpaired [64, 215–217, 379, 424, 498] data setup. Recent works [74, 299, 331, 335, 341] have begun to explore any-to-any translation, where the goal is to generalize to any input and any target speaker. To do so, such approaches learn a speaker-agnostic embedding space that is meant to generalize to never-before-seen identities. Our



(a) zero-shot conversion **(b) one-on-one conversion** **(c) ours — Exemplar Autoencoders**

Figure 4.2: Prior approaches can be classified into two groups: (a) **Zero-shot conversion** that learns a generic low-dimensional embedding from a training set that are designed to be agnostic to speaker identity. We empirically observe that such generic embeddings may struggle to capture stylistic details of in-the-wild speech that differs from the training set. (b) Person-specific **one-on-one conversion** learn a translation engine specialized to specific input and output speakers, restricting them to known input speakers at test time. (c) In this work, we combine the zero-shot nature of generic embeddings with the stylistic detail of person-specific translation systems. Simply put, given a target speech with a particular style and ambient environment, we learn an **autoencoder** specific to that target speech. At test time, one can translate any input speech into the target simply by passing it through the target **Exemplar Autoencoder**.

work is closely inspired by such approaches, but is based on the observation that such generic embeddings tend to be low-dimensional, making it difficult to capture the full range of stylistic prosody and ambient environments that can exist in speech. We can, however, capture these subtle but important aspects via an Exemplar Autoencoder that is trained for a specific target speech exemplar. Moreover, it is not clear how to extend such generic embeddings to audio-visual generation because visual appearance is highly multimodal (people can appear wildly different due to clothing). Exemplar Autoencoders, on the other hand, are straightforward to extend to video because they inherently are tuned to the particular visual mode (clothing) in the target speech.

Audio-Video Synthesis: There is a growing interest [78, 304, 317] to jointly study audio and video for better recognition [224], localizing sound [134, 376], or learning better visual representations [323, 324]. There is a wide literature [36, 48, 110, 129, 151, 416] on synthesizing videos (talking-heads) from an audio signal. Recent approaches [77, 502, 509] have looked at zero-shot facial synthesis from audio signals. These approaches register human faces using facial keypoints so that one can expect a certain facial part at a fixed location. At test time, an image of the target

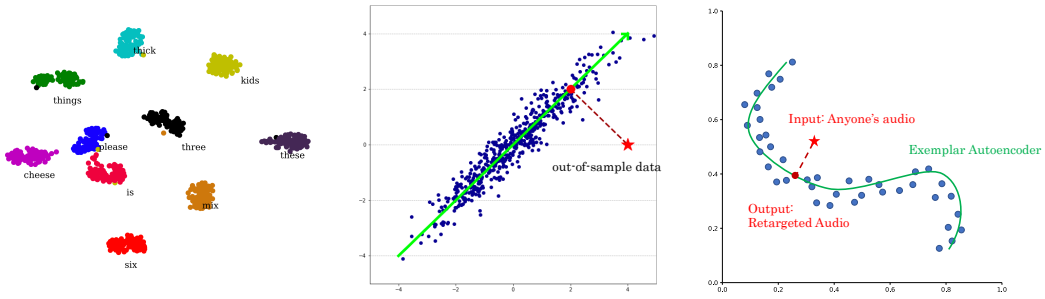
Method	input: audio, **output**: ?	unknown test speaker	speaker-specific model
Auto-VC [341]	A	✓	✗
StarGAN-VC [217]	A	✗	✓
Speech2Vid [77]	V	✓	✗
Synthesizing Obama [411]	V	✗	✓
Ours (Exemplar Autoencoders)	AV	✓	✓

Table 4.1: We contrast our work with leading approaches in audio conversion [217, 341] and audio-to-video generation [77, 411]. Unlike past work, we generate both audio-video output from an audio input (**left**). Zero-shot methods are attractive because they can generalize to unknown target speakers at test time (**middle**). However, in practice, models specialized to particular pairs of known input and target speakers tend to produce more accurate results (**right**). Our method combines the best of both worlds by tuning an Exemplar Autoencoder on-the-fly to the target speaker of interest.

speaker and audio is provided. While these approaches can animate lips of the target speaker, it is still difficult to capture other realistic facial expressions. Other approaches [139,386,411] use large amount of training data and supervision (in the form of keypoints) to learn person-specific synthesis models. While these generate realistic results, they are restricted to known input users at test time. In contrast, our goal is to synthesize audio-visual content (given *any* input audio) in a data-efficient manner (using 2-3 minutes of unsupervised data). Finally, we find our approach complementary to unsupervised video retargeting [22]. We combine our approach with Recycle-GAN, making the resulting system an unsupervised audio-video retargeting engine.

Autoencoders: It is well-known that linear autoencoders (learned with PCA [18]) produce the best reconstruction of any out-of-sample input datapoint, in terms of squared error from the subspace spanned by the training dataset [41]. We empirically verify that this property approximately holds for nonlinear autoencoders (Fig. 4.3). When trained on a target (exemplar) dataset consisting of a particular speech style, we demonstrate that reprojections of out-of-sample inputs tend to preserve the content of the input and the style of the target.

User-Controllable Content Creation: Our design decisions have chiefly been influenced by user-perspective. The use of *many* Exemplar Autoencoders provide flexibility to a user to select the target speaker and a scenario. Not only this, our system can be easily extended in few minutes to new examples using only two-three minutes of audio and a few seconds of video sequence for a new person or scenario. Our work can also be useful in video editing systems directly from audio [36, 129]. We release a live public web-demo with this work that enables anyone to input au-



(a) T-SNE for acoustic features **(b) PCA visualization** **(c) Exemplar Autoencoder**

Figure 4.3: Our insights for Exemplar Autoencoders. (a) **Acoustic features** (MEL spectrograms) of words spoken by 100 different speakers (from the VCTK dataset [437]) easily cluster the words/content irrespective of who said it. (b) **Linear autoencoders** (PCA) *with sufficiently small bottlenecks* project out-of-sample data (the red star) into the linear manifold spanned by the training set. We demonstrate this approximately holds for (c) **nonlinear autoencoders**, allowing them to retarget the content of out-of-sample input into the style of the training set.

dio using a mic and generate audio-video of their favorite person.

4.3 Exemplar Autoencoders

There is an enormous space of stylistic information ranging from prosody, pitch, emotions, and environment. It is challenging for a single *large* model to learn different things. However, many *small* models can easily capture the various nuances. In this work, we seek the problem of any-to-many audio-visual synthesis from an audio input via Exemplar Autoencoders (explicitly trained for a speaker and scenario). In this section, we study why autoencoders should generate retargeted output when applied to a novel input speech (Sec. 4.3.1 and Sec. 4.3.2). We then perform a deep structural dive into Exemplar Autoencoders for audio-video synthesis in Sec. 4.3.3. We begin with defining the speech structure.

Speech contains two types of information: (i) the **content**, or words being said and (ii) **style** information that describes the scene context, person-specific characteristics, and prosody of the speech delivery. It is natural to assume that speech is generated by the following process. First, a style s is drawn from the style space S. Then a content code w is drawn from the content space W. Finally, $x = f(s, w)$ denotes the speech of the content w spoken in style s, where f is the generating function of speech.

4.3.1 Structured Speech Space

In human acoustics, one uses different shapes of their vocal tract[1] to pronounce different words with their voice. Interestingly, different people use similar, or ideally the same, shapes of vocal tract to pronounce the same words. For this reason, in the speech space we can find a built-in structure that the acoustic features of the same words in different styles are very close (also shown in Fig. 4.3-a). Given two styles s_1 and s_2 for example, $f(s_1, w_0)$ is one spoken word in style s_1. Then $f(s_2, w_0)$ should be closer to $f(s_1, w_0)$ than any other word in style s_2. We can formulate this property as follows:

$$\text{Error}(f(s_1, w_0), f(s_2, w_0)) \leq \text{Error}(f(s_1, w_0), f(s_2, w)), \forall w \in W, \quad \text{where} \quad s_1, s_2 \in S.$$

This can be further presented in equation form:

$$f(s_2, w_0) = \arg\min_{x \in M} \text{Error}(x, f(s_1, w_0)), \quad \text{where} \quad M = \{f(s_2, w) : w \in W\}. \quad (4.1)$$

4.3.2 Autoencoders as Projection Operators

We now provide a statistical motivation for the ability of Exemplar Autoencoders to translate content while preserving style.

Given training examples $\{x_i\}$, one learns an encoder E and decoder D so as to minimize reconstruction error:

$$\min_{E,D} \sum_i \text{Error}\Big(x_i, D(E(x_i))\Big).$$

In the linear case (where $E(x) = Ax$, $D(x) = Bx$, and Error = L2), optimal weights are given by the eigenvectors that span the input subspace of data [18].

Given sufficiently small bottlenecks, linear autoencoders project out-of-sample points into the input subspace, so as to minimize the reconstruction error of the output (see Fig. 4.3-(b)). Weights of the autoencoder (eigenvectors) capture dataset-specific *style* common to all samples from that dataset, while the bottleneck activations (projection coefficients) capture sample-specific *content* (properties that capture individual differences between samples). We empirically verify that a similar property holds for nonlinear autoencoders: given sufficiently-small bottlenecks, they *approximately* project out-of-sample data onto the nonlinear *manifold* M spanned by the training set:

$$D(E(\hat{x})) \approx \arg\min_{m \in M} \text{Error}(m, \hat{x}), \text{where} \qquad M = \{D(E(x)) | x \in \mathbb{R}^d\}. \qquad (4.2)$$

[1] The vocal tract is the cavity in human beings where the sound produced at the sound source is filtered. The shape of the vocal tract is mainly determined by the positions and shapes of the tongue, throat and mouth.

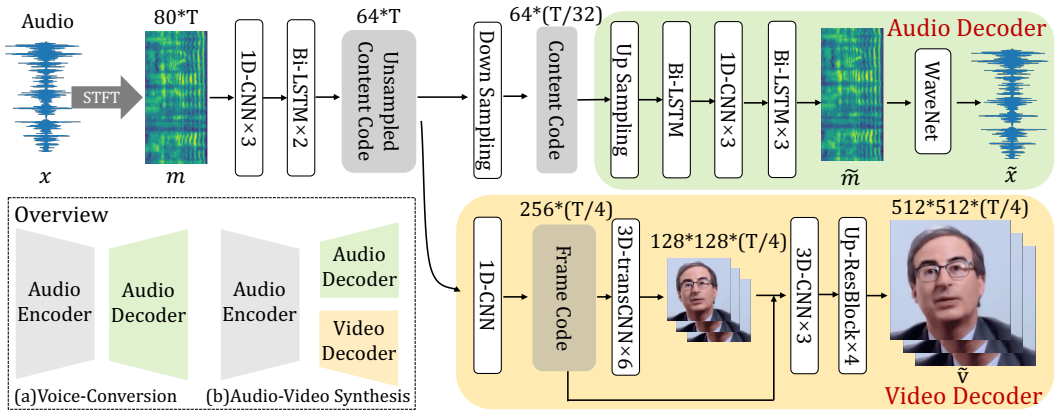


Figure 4.4: **Network Architecture:** (a) The voice-conversion network consists of a content encoder, and an audio decoder (denoted as green). This network serves as a person & attributes-specific auto-encoder at training time, but is able to convert speech from anyone to personalized audio for the target speaker at inference time. (b) The audio-video synthesis network incorporates a video decoder (denoted as yellow) into the voice-conversion system. The video decoder also regards the content encoder as its front-end, but takes the unsampled content code as input due to time alignment. The video architecture is mainly borrowed from StackGAN [342, 492], which synthesizes the video through 2 resolution-based stages.

where the approximation is exact for the linear case. In the nonlinear case, let $M = \{f(s_2, w) : w \in W\}$ be the manifold spanning a particular style s_2. Let $\hat{x} = f(s_1, w_0)$ be a particular word w_0 spoken with any other style s_1. The output $D(E(\hat{x}))$ will be a point on the target manifold M (roughly) closest to $\hat{x}$. From Eq. 4.1 and Fig. 4.3, the closest point on the target manifold tends to be the same word spoken by the target style:

$$D(E(\hat{x})) \approx \arg\min_{t \in M} \text{Error}(t, \hat{x}) = \arg\min_{t \in M} \text{Error}(t, f(s_1, w_0)) \approx f(s_2, w_0). \quad (4.3)$$

At the end of theoretical analysis on Exemplar Autoencoders, we note that nothing in analysis is language-specific, so it still works, in principle, for other languages such as Mandarin Chinese and Hindi. We posit that the "content" w_i can be represented by phonemes, which are shared by different languages. We verify this by showing that retargeting is possible *across* different languages: one can drive John Oliver to speak in Chinese or Hindi.

4.3.3 Stylistic Audio-Visual (AV) Synthesis

We now operationalize our previous analysis into an approach for stylistic AV synthesis. To do so, we learn AV representations with autoencoders tailored to partic-

ular target speech. To enable these representations to be driven by the audio input of any user, we learn in a manner that ensures bottleneck activations capture structured linguistic "word" content: we pre-train an audio-only autoencoder, and then learn a video decoder while finetuning the audio component. This adaptation allows us to train AV models for a target identity with little data (3 minutes of video in contrast to the 14 hours used in [411]).

Audio Autoencoder: Given the target audio stream of interest x, we first convert it to a mel spectogram $m = \mathcal{F}(x)$ using a short-time Fourier Transform [321]. We train an encoder E and decoder D that reconstructs Mel-spectograms $\tilde{m}$, and finally use a WaveNet vocoder V [320] to convert $\tilde{m}$ back to speech $\tilde{x}$. Importantly, we train all components (the encoder E, decoder D, and vocoder V) with a joint reconstruction loss in both frequency and audio space:

$$\text{Error}_{Audio}(x, \tilde{x}) = \mathbb{E}\|m - \tilde{m}\|_1 + L_{WaveNet}(x, \tilde{x}), \tag{4.4}$$

where $L_{WaveNet}$ is the standard cross-entropy loss used to train Wavenet [320]. Fig. 4.4 (top-row) summarizes our network design for audio-only autoencoders.

Video Decoder: Given the trained audio autoencoder $E(x)$ and a target video v, we now train a video decoder that reconstructs the video $\tilde{v}$ from $E(x)$ (see Fig. 4.4).

Specifically, we train using a joint loss:

$$\text{Error}_{AV}(x, v, \tilde{x}, \tilde{v}) = \text{Error}_{Audio}(x, \tilde{x}) + \mathbb{E}\|v - \tilde{v}\|_1 + L_{Adv}(v, \tilde{v}). \tag{4.5}$$

where L_{Adv} is an adversarial loss [147] used to improve video quality. We found it helpful to simultaneously fine-tune the audio autoencoder and so add (4.4) to the overall AV loss.

4.4 Implementation Details

We provide the implementation details of our Exemplar Autoencoder.

4.4.1 Audio Conversion

STFT: The speech data is sampled at 16 kHz. We clip the training speech into clips of 1.6s in length, which correspond to $25,600$-dimensional vectors. We then perform STFT [321] on the raw audio signal with a window size of 800, hop size of 200, and 80 mel-channels. The output of STFT is a complex matrix with size of 80×128. We represent the complex matrix in polar form (magnitude and phase), and only keep the magnitude for next steps.

Encoder: The input to the encoder is the magnitude of 80×128 Mel-spectrogram, which is represented as a 1D 80-channel signal. This input is feed-forward to three layers of 1D convolutional layers with a kernel size of 5, each followed by batch normalization [194] and ReLU activation [238]. The channel of these convolutions

is 512. The stride is one. There is no time down-sampling up till this step. The output is then fed into two layers of bidirectional LSTM [180] layers with both the forward and backward cell dimensions of 32. We then perform a different down-sampling for the forward and backward paths with a factor of 32 following [341]. The result content embedding is a matrix with a size of 64×4.

Audio Decoder: The content embedding is up-sampled to the original time resolution of T. The up-sampled embedding is sequentially input to a 512-channel LSTM layer and three layers of 512-channel 1D convolutional layers with a kernel size of 5. Each step accompanies batch normalization and ReLU activation. Finally, the output is fed into two 1024-channel LSTM layers and a fully connected layer to project into 80 channels. The projection output is regarded as the generated magnitude of Mel-spectrogram $\tilde{m}$.

Vocoder: We use WaveNet as vocoder [320] that acts like an inverse Fourier transform, but merely use frequency magnitudes. It generates speech signal $\tilde{x}$ based on the reconstructed magnitude of Mel-spectrogram $\tilde{m}$.

Training Details: Our model is trained at a learning rate of 0.001 and a batch size of 8. To train a model from scratch, it needs about 30 minutes of the target speaker's speech data and around 10k iterations to converge. Although our main structure is straightforward, the vocoder is usually a large and complicated network, which needs another 50k iterations to train. However, transfer learning can be beneficial in reducing the number of iterations and necessary data for training purposes. When fine-tuning a new speaker's autoencoder from a pre-trained model, we only need about 3 minutes of speech from a new speaker. The entire model, including the vocoder, converges around 10k iterations.

4.4.2 Video Synthesis

Network Architecture: We keep the voice-conversion framework unchanged and enhance it with an additional audio-to-video decoder. In the voice-conversion network, we have a content encoder that extracts content embedding from speech, and an audio decoder that generates audio output from that embedding. To include video synthesis, we add a video decoder which also takes the content embedding as input, but generates video output instead. As shown in Figure 4.4 (bottom-row), we then have an audio-to-audio-video pipeline.

Video Decoder: We borrow architecture of the video decoder from [342, 492]. We adapt this image synthesis network by generating the video frame by frame, as well as replacing the 2D-convolutions with 3D-convolutions to enhance temporal coherence. The video decoder takes the unsampled content codes as input. This step is used to align the time resolution with 20-fps videos in our experiments. We down-sample it with a 1D convolutional layer. This step helps smooth the adjacent video frames. The output is then fed into the synthesis network to get the video result $\tilde{v}$.

VCTK [437]	Zero-Shot	Extra-Data	SCA (%) (Voice Similarity)	MCD (Content Consistency)
StarGAN-VC [217]	✗	✓	69.5	582.1
VQ-VAE [436]	✗	✓	69.9	663.4
Chou et al. [75]	✗	✓	98.9	**406.2**
Blow [379]	✗	✓	87.4	444.3
Auto-VC [341]	✓	✓	98.5	408.8
Ours	✓	✗	**99.6**	420.3

Table 4.2: **Objective Evaluation for Audio Translation:** We do objective evaluation using VCTK dataset that provides paired data. The speaker-classification accuracy (SCA) criterion enables us to study the naturalness of generated audio samples and similarity to the target speaker, where **higher is better**. On the other hand, Mel-Cepstral distortion (MCD) assesses content preservation, where **lower is better**. Our approach achieves competitive performance to prior state-of-the-art without requiring any extra-data and yet be zero-shot. We do a more comprehensive human studies in Table 4.3 using CelebAudio-20 dataset to study the influence of data.

Training Details: We train an audio-visual model based on a pretrained audio model, with a learning rate of 0.001 and a batch size of 8. When fine-tuning from a pre-trained model, the overall model converges around 3k iterations.

4.5 Quantitative Analysis

We now quantitatively evaluate the proposed method for any-to-many audio conversion and audio-video synthesis.

4.5.1 Audio Translation

Dataset: We use the publicly available VCTK dataset [437] and introduce a new CelebAudio dataset for in-the-wild audio translation setup to inspire future work in this direction.

VCTK Dataset: VCTK corpus [437] contains 44 hours of utterances from 109 speakers. Each speaker reads a different set of sentences, except for two paragraphs. While the conversion setting is non-parallel, there exists a small amount of parallel data enables us to conduct objective evaluation.

CelebAudio Dataset: We introduce a new *in-the-wild* dataset for audio translation to validate the effectiveness as well as the robustness of various approaches. This dataset consists of speeches (average 30 minutes) of various public figures collected

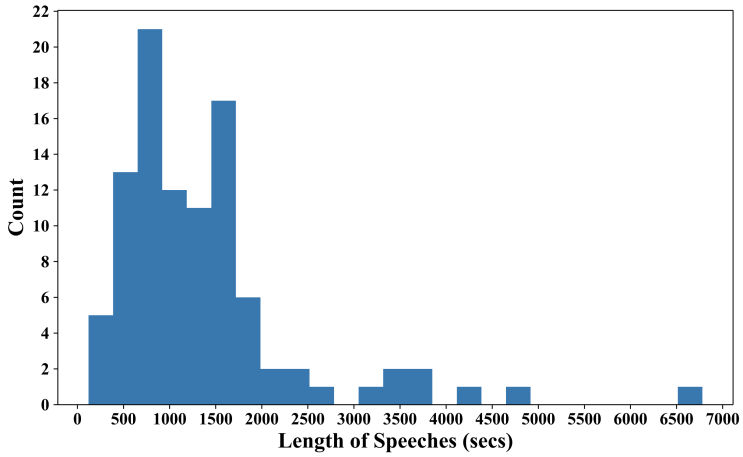


Figure 4.5: **Data Distribution of CelebAudio:** CelebAudio contains 100 different speeches. The average speech length is 23 mins. Each speech ranges from 3 mins (Hillary Clinton) to 113 mins (Neil Degrasse Tyson).

from YouTube. The content of these speeches is entirely different from one another, thereby forcing the future methods to be non-parallel and unsupervised. There are 100 different speeches. We use 20 celebrities for human-studies, and call that subset **CelebAudio-20**. The distribution of speech length is shown in Fig. 4.5

Quantitative Evaluation: We conducted user studies for evaluation on CelebAudio-20 dataset, and do objective evaluation using VCTK dataset.

Speaker Classification Accuracy: We use the speaker-classification accuracy (SCA) criterion to study voice conversion for different approaches. We train a speaker classifier following Serra et al. [379]. We compute the percentage of times a translation is able to classify correctly. **Higher is better**.

Mel-Cepstral Distance: Following prior works [216, 217], we use Mel-Cepstral distortion (MCD) to another objective evaluation criterion to assess content consistency. This metric assesses the distance between the synthesized audio and ground-truth. **Lower is better**.

Human Studies: We extensively conducted human studies on Amazon Mechanical Turk (AMT) using the generated audio samples from CelebAudio-20 dataset. The goal of this study is to (1). assess the quality of generated samples; (2). the ability of an approach to produce voice similar to target speaker; and (3). to ensure the consistency of content during the translation process. We, therefore, conducted our studies in three different phases. In the first phase, we presented an audio sample to a user on AMT and asked: *How natural is this recording?*. The results of this phase enables us to study **naturalness** of generated content. In the next phase, we presented two audio samples (one real, and another generated) to a user and asked: *Are these two audios from the same speaker?*. We instructed users to pay attention to

CelebAudio	Extra Data	Naturalness ↑ MOS	Naturalness ↑ Pref (%)	Voice Similarity ↑ MOS	Voice Similarity ↑ Pref (%)	Content Consistency ↑ MOS	Content Consistency ↑ Pref (%)	Geometric Mean of VS-CC ↑
Auto-VC [341]								
off-the-shelf	-	1.21	0	1.31	0	1.60	0	1.45
fine-tuned	✓	2.35	13.3	1.97	2.0	**4.28**	**48.0**	2.90
scratch	✗	2.28	6.7	1.90	2.0	4.05	18.0	2.77
Ours	✗	**2.78**	**66.7**	**3.32**	**94.0**	4.00	22.0	**3.64**

Table 4.3: **Human Studies for Audio**: We conduct extensive human studies on Amazon Mechanical Turk. We report Mean Opinion Score (MOS) and user preference (percentage of time that method ranked best) for naturalness, target voice similarity (VS), source content consistency (CC), and the geometric mean of VS-CC (since either can be trivially maximized by reporting a target/input sample). Higher the better, on a scale of 1-5. The off-the-shelf Auto-VC model struggles to generalize to CelebAudio, indicating the difficulty of in-the-wild zero-shot conversion. Fine-tuning on CelebAudio dataset significantly improves performance. When restricting Auto-VC to the same training data as our model (scratch), performance drops a small but noticeable amount. Our results strongly outperform Auto-VC for VS-CC, suggesting Exemplar Autoencoders are able to generate speech that maintains source content consistency while being similar to the target voice style.

the voice only and ignore the speech content. This phase allows us to study the notion of **voice-similarity**. Finally, we once again presented two audio samples (one real, and another generated) to a user and asked: *Are these two people saying the same thing?*. Here, we instructed users to pay attention to the content and not voice. The results of this phase allows us to study how much content is preserved during audio translation, i.e., **content-consistency**. The users were asked to rate on a scale of 1-5, i.e., bad to excellent. For all these criteria: **Higher is better**.

We also compute the normalized area under the voice-similarity and content-consistency curve. This criterion is useful to study the joint notion of content and style for audio translation. Given a (source, target) pair, we always have two simple ways to generate the output : (1) output the source audio - 0 in voice similarity (VS) but 5 in content consistency (CC); (2) output a random audio from target speaker - 5 in VS but 0 in CC. However, good results should be in between with the source's content but the target's voice. E.g., a $(2.5, 2.5)$ result should be better than either $(5, 0)$ or $(0, 5)$. The normalized area under VS-CC enables us to study it effectively. **Higher is better**.

We select 10 speakers as target speakers from CelebAudio-20 and randomly choose 5 utterances from the other speakers for test. We then produce $5 \times 10 = 50$ conversions by converting one test utterance to each of the selected 10 speakers' voice. There are a total of 150 HITs for testing naturalness, voice similarity, and content consistency. Each HIT is assigned to 10 users. All the users of AMT were

chosen to have Master Qualification (HIT approval rate more than 98% for more than $1,000$ HITs). We also restricted the users to be from United States to ensure English-speaking audience. Our setup and dataset are available on project page for public-use.

Baselines: We study the various aspects of our methods in contrast with several existing voice conversion systems, such as StarGAN-VC [217], VQ-VAE [436], Blow [379], and Auto-VC [341]. While possible, StarGAN-VC [217], VQ-VAE [436], Chou et al. [75], and Blow [379] does not claim zero-shot voice conversion. Therefore, we train these models on 20 speakers from VCTK dataset and perform traditional voice conversion between speakers within the training set. We observe that our approach outperforms these approaches for voice similarity and yet competitive for content consistency as shown in Table 4.2.

Auto-VC [341]: Auto-VC claims zero-shot conversion. We extensively study two methods via human studies on AMT in Table 4.3. Since Auto-VC claims any-to-any audio conversion, we first use an **off-the-shelf** model[2] for evaluation. We observe poor performance, both quantitatively and qualitatively. We then **fine-tuned** the existing model using the audio data from 20 speakers, thereby making it any-to-many audio translation approach (similar to ours). We observe significant performance improvement in Auto-VC when restricting it to the same set of examples as ours. To make it more similar to ours, we even trained the models from **scratch** using exactly same data and settings as ours. The performance on all three criterion dropped with lesser data. On the other hand, our approach can generate significantly better audio outputs that sounds more like a target speaker while still preserving the original content. Importantly, our models are trained from scratch and does not require hundreds of hours of speech data for training.

4.5.2 Audio-Video Synthesis

Dataset: In addition to augmenting CelebAudio with video clips, we also evaluate on VoxCeleb [78], an audio-visual dataset of short celebrity interview clips from YouTube.

Baselines: We compare to **Speech2Vid** [77], using their publicly available code. This approach requires the face region to be registered before feeding it to a pre-trained model for facial synthesis. We find various failure cases where the human face from in-the-wild videos of VoxCeleb dataset cannot be correctly registered (in contrast with our approach where registration is not required). Additionally, Speech2Vid generates the video of a cropped face region by default. We composite the output on a still background to make it more compelling and comparable to ours. Finally, we also compare to the recent work of **LipGAN** [234]. This approach synthesizes region around lips and paste this generated face crop into the given video. We observe that the paste is not seamless and leads to artifacts es-

[2] Auto-VC inputs a reference audio (roughly 20 seconds long) to obtain a speaker embedding.

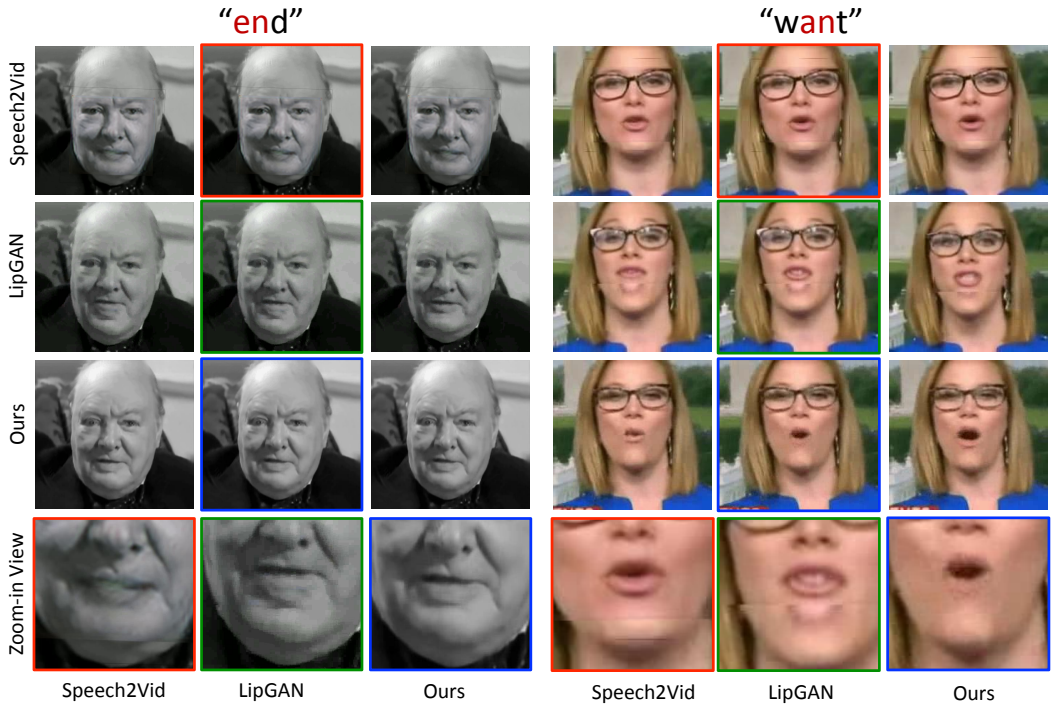


Figure 4.6: **Visual Comparisons of Audio-to-Video Synthesis:** We contrast our approach with Speech2vid [77] (first row) and LipGAN [234] (second row) using their publicly available codes. While Speech2Vid synthesize new sequences by morphing mouth shapes, LipGAN pastes modified lip region on original videos. This leads to artifacts (see zoom-in views at bottom row) when morphed mouth shapes are very different from the original or in case of dynamic facial movements. Our approach (third row), however, generates full face and does not have these artifacts.

pecially when working with in-the-wild web videos. We show comparisons with both Speech2Vid [77] and LipGAN [234] in Figure 4.6. The artifacts from both approaches are visible when used for in-the-wild video examples from CelebAudio and VoxCeleb dataset. However, our approach generates the complete face without these artifacts and captures articulated expressions.

Human Studies: We conducted an AB-Test on AMT: we show a pair of our video output and another method to the user, and ask the user to select which looks better. Each pair is shown to 5 users. Our approach is preferred 87.2% over Speech2Vid [77] and 59.3% over LipGAN [234].

4.6 Verification of Reprojection Property

In Section 4.3, we describe a reprojection property (Eq. 4.6) of our Exemplar Autoencoder: the output of out-of-sample data $\hat{x}$ is the projection to the manifold M spanned by the training set.

$$D(E(\hat{x})) \approx \arg\min_{m \in M} \text{Error}(s, \hat{x}), \text{where} \qquad M = \{D(E(x)) | x \in \mathbb{R}^d\}. \tag{4.6}$$

This is equivalent to two important properties: (a) The output lies in the manifold M. (Eq. 4.7) (b) The output is closer to the input $\hat{x}$ than any other point on manifold M. (Eq. 4.8)

$$D(E(\hat{x})) \in M, \text{where} \qquad M = \{D(E(x)) | x \in \mathbb{R}^d\}. \tag{4.7}$$

$$\text{Error}(D(E(\hat{x})), \hat{x}) \leq \text{Error}(x, \hat{x}), \qquad \forall x \in M. \tag{4.8}$$

To verify these, we conduct an experiment (Table 4.4) on the parallel corpus of VCTK [437]. We randomly sample 2 sets of words (normal and tongue-twister) spoken by 2 speakers (A and B) from VCTK dataset. We train an Exemplar Autoencoder on A's speech, and input B's words (w_B) to get $w_{B \to A}$. To verify Eq. 4.7, we train a speaker-classifier (as Serra et al. [379]) on A and B's speech. As the fifth column in Table 4.4, we report the likelihood that $w_{B \to A}$ is regarded as A's words. To verify Eq. 4.8, we calculate $\text{Error}(D(E(\hat{x})), \hat{x})$ as the second column in Table 4.4, and $\text{Error}(x, \hat{x})$ as the third and fourth columns in Table 4.4. The results show that our conjecture about the reprojection property is reasonable.

4.7 Ablative Analysis

We now conduct controlled experiments to understand various aspects of our formulation.

4.7.1 Exemplar Training

In this work, we posit that it is easier to disentangle style from content when the entire training dataset consists of a single style. To verify this, we train an autoencoder with following setup: (1) speeches from 2 people; (2) speeches from 3 people; (3) 4 speeches with different settings from 1 person. We contrast it with exemplar autoencoder trained using 1 speech from 1 person using A/B testing. Our exemplar results are preferred 71.9% times over (1), 83.3% times over (2), and 90.6% times over (3).

w_B	$\|\|w_{B\to A} - w_B\|\|$	$\min_{w\in s_A} \|\|w - w_B\|\|$	$\text{mean}_{w\in s_A}\|\|w - w_B\|\|$	$\arg\min_{w\in s_A} \|\|w - w_B\|\|$	Likelihood(%)
"please"	17.46	23.11	53.93	"please"	100
"things"	21.59	19.87	50.95	"things"	81.8
"these"	25.22	21.26	82.00	"these"	55.6
"cheese"	21.98	20.34	51.19	"cheese"	100
"three"	24.39	22.13	61.07	"three"	84.1
"mix"	16.19	16.61	55.77	"mix"	100
"six"	20.43	16.75	66.85	"six"	100
"thick"	21.21	18.38	50.16	"thick"	92.3
"kids"	16.98	16.65	48.78	"kids"	100
"is"	16.84	16.25	70.88	"is"	78.0
Average	20.23	19.14	59.16		

Table 4.4: **Verification of the reprojection property:** To verify the reprojection property that Eq. 4.6 describes, we randomly sample tongue-twister words spoken by 2 speakers (A and B) in VCTK dataset. For the autoencoder trained by A, all the words spoken by B are out-of-sample data. Then for each word w_B, we generate $w_{B\to A}$ which is the projection to A's subspace. We need to verify 2 properties - **(a)** The output $w_{B\to A}$ lies in A's subspace; **(b)** The distance between w_B and corresponding word for S_A should be much lesser than the average of distance between w_B and other words for S_A. To verify (a), we train a speaker classification network on speaker A and B, and predict the speaker of $w_{B\to A}$. We report the softmax output to show how much likely $w_{B\to A}$ is to be classified as A's speech (Likelihood in the table). To verify (b), we calculate (1) distance from w_B to $w_{B\to A}$; (2) minimum distance from w_B to any sampled words by A; and (3) average distance from w_B to different words by A. The second column shows distance between transformed word ($w_{B\to A}$) and the original word (w_B). The third column shows minimum distance between words of S_A and w_B. The fourth column shows an average distance between words of S_A and w_B. We also show word corresponding to *min-distance* in fifth column. This also verifies that *min-distance* corresponds to the same word for S_A. This empirically suggests that nonlinear autoencoder behave similar to their linear counterparts (i.e., they minimize the reconstruction error of the out-of-sample input).

4.7.2 Removing Speaker Embedding

We point out that generic speaker embeddings struggles to capture stylistic details of in-the-wild speech. Here we provide the ablation analysis of removing such pretrained embeddings.

We perform new experiments that fine-tune a pretrained Auto-VC for each specific speaker (exemplar training) but still keeps the pre-trained embedding, and find it preforms similarly to the multi-speaker fine-tuning in Table 4.3. Our outputs are still preferred 91.7% times over it.

VCTK [437]	Voice Similarity (SCA / %)	Content Consistency (MCD)
Chou et al. [74]	57.0	491.1
Auto-VC [341]		
with WaveNet	98.5	**408.8**
without WaveNet	96.5	**408.8**
Ours		
with WaveNet	**99.6**	420.3
without WaveNet	**97.0**	420.3

Table 4.5: **Ablation Analysis of WaveNet:** We replace the WaveNet vocoder with Griffin-Lim traditional vocoder, and compare our method with two zero-shot methods: **(1)** AutoVC without Wavenet, and **(2)** Chou et al. [74]. We measure Voice Similarity by speaker-classification accuracy (SCA) criterion, where higher is better. The results show our approach without WaveNet still outperform other zero-shot approaches not using WaveNet. We also report MCD same as Table 4.2 as MCD is only related to Freq. features. For reference, we also list the results of "with Wavenet" experiments.

4.7.3 WaveNet as Vocoder

We adopt a neural-net vocoder to convert Mel-spectrograms back to raw audio signal. We prefer WaveNet [320] to traditional vocoders like Griffin-Lim [152], since Wavenet is one of the state-of-the-art methods. The existence of WaveNet in our structure makes our method end-to-end trainable and can safely be considered a part of exemplar autoencoder (no special adaptation required). While AutoVC [341] and VQ-VAE [436] also use WaveNet, we substantially outperform them on Celeb-Audio dataset.

Here we add the ablation analysis of WaveNet in Table 4.5. We remove the WaveNet vocoder from our approach and AutoVC, and replace it with Griffin-Lim. We also compare with Chou et al. [74], which does not use neural-net vocoders. The results show our model can also generate reasonable outputs with traditional vocoders, and yet outperform other zero-shot approaches not using WaveNet.

4.7.4 Video Synthesis

We study the effectiveness of jointly training a audio-to-audio-video pipeline based on a pre-trained audio model. To further verify the helpfulness of audio bottleneck features and finetuning from audio model in video synthesis, we conduct two ablation experiments on audio-to-video translation, and compare with ours. (a) We contrast training only the audio-to-video translator (first baseline in Table 4.6). (b) We then jointly train audio decoder and video decoder from scratch (second Baseline in Table 4.6). (c) We train an autoencoder of audio, then train video decoder while finetuning the audio part (Ours). From the results in Table 4.6, ours outper-

VoxCeleb	Finetune	Audio Decoder	MSE↓	PSNR↑
Baselines	✗	✗	77.844 ± 15.612	29.304 ± 0.856
	✗	✓	77.139 ± 12.577	29.315 ± 0.701
Ours	✓	✓	$\mathbf{76.401 \pm 12.590}$	$\mathbf{29.616 \pm 0.963}$

Table 4.6: **Ablation Analysis of Video Synthesis:** To verify the effectiveness of a pre-trained audio model and audio decoder, we construct two extra baselines: **(a)** Train only the audio-to-video translator (first baseline). **(b)** Jointly train audio decoder and video decoder from scratch (second baseline). And we compare these two baselines with our method: First train an autoencoder of audio, then train video decoder while finetuning the audio part. From the results, we can see the performance clearly drops without the help of finetuning and audio decoder.

forms (b), which indicates the effectiveness of finetuning from a pre-trained audio model. Finally, (b) also outperforms (a). This implies the effectiveness of audio bottleneck features.

4.8 Broader Impacts

Our work falls in line with a body of work on content generation that retargets video content, often considered in the context of "facial puppeteering". While there exist many applications in entertainment, there also exist many potentials for serious abuse. Past work in this space has included broader impact statements [129, 226], which we build upon here.

Our Setup: Compared to prior work, unique aspects of our setup are (a) the choice of input and output modalities and (b) requirements on training data. In terms of (a), our models take audio as input, and produce audio-visual (AV) output that captures the personalized style of the target individual but the linguistic content (words) of the source input. This factorization of style and content across the source and target audio is a key technical aspect of our work. While past work has discussed broader impacts of visual image/video generation, there is less discussion on responsible practices for audio editing. We begin this discussion below. In terms of (b), our approach is unique in that we require only a few minutes of training on target audio+video, and do not require training on large populations of individuals. This has implications for generalizability and ease-of-use for non-experts, increasing the space of viable target applications. Our target applications include entertainment/education and assistive technologies, each of which is discussed below. We conclude with a discussion of potential abuses and strategies for mitigation.

Assistive Technology: An important application of voice synthesis is voice generation for the speaking impaired. Here, generating speech in an individual's per-

sonalized style can be profoundly important for maintaining a sense of identity[3]. We demonstrate applications in this direction. We have also begun collaborations with clinicians to explore personalized and stylistic speech outputs from physical devices (such as an electrolarynx) that directly sense vocal utterances. Currently, there are considerable data and privacy considerations in acquiring such sensitive patient data. We believe that illustrating results on well-known individuals (such as celebrities) can be used to build trust in clinical collaborators and potential patient volunteers.

Entertainment/Education: Creating high-quality AV content is a labor intensive task, often requiring days/weeks to create minutes of content for production houses. Our work has attracted the attention of production houses that are interested in creating documentaries or narrations of events by historical figures in their own voice (Winston Churchill, John F Kennedy, Nelson Mandela, Martin Luther King Jr). Because our approach can be trained on small amounts of target footage, it can be used for personalized storytelling (itself a potentially creative and therapeutic endeavour) as well as educational settings that have access to less computational/artistic resources but more immediate classroom feedback/retraining.

Abuse: It is crucial to acknowledge that audio-visual retargeting can be used maliciously, including spreading false information for propaganda purposes or pornographic content production. Addressing these abuses requires not only technical solutions, but discussions with social scientists, policy makers, and ethicists to help delineate boundaries of defamation, privacy, copyright, and fair use. We attempt to begin this important discussion here. First, inspired by [129], we introduce a recommended policy for AV content creation using our method. In our setting, retargeting involves two pieces of media; the source individual providing an audio signal input, and target individual who's audio-visual clip will be edited to match the words of the source. Potentials for abuse include plagiarism of the source (e.g., representing someone else's words as your own), and misrepresentation / copyright violations of the target. Hence, we advocate a policy of always citing the source content and always obtaining permission from the target individual. Possible exceptions to the policy, if any, may fall into the category of fair use[4], which typically includes education, commentary, or parody (eg., retargeting a zebra texture on Putin [512]). In all cases, retargeted content should acknowledge itself as edited, either by clearly presenting itself as parody or via a watermark. We plan to include a custom license on released code, detailing this policy. Importantly, there may be violations of this policy. Hence, a major challenge will be identifying such violations and mitigating the harms caused by such violations, discussed further below.

Audio-Visual Forensics: We describe three strategies that attempt to identify misuse and the harms caused by such misuse. We stress that these are not exhaus-

[3] https://news.northeastern.edu/2019/11/18/personalized-text-to-speech-voices-help-people-with-speech-disabilities-maintain-identity-and-social-connection/

[4] https://fairuse.stanford.edu/overview/fair-use/what-is-fair-use/

tive: (a) identifying "fake" content, (b) data anonymization, and (c) technology awareness. (a) As approaches for editing AV content mature, society needs analogous approaches for detecting such "fake" content. Contemporary forensic detectors tend to be data-driven, themselves trained to distinguish original versus synthesized media via a classification task. Such forensic real-vs-fake classifiers exist for both images [285, 360, 448] and audio[5,6]. Because access to code for generating "fake" training examples will be crucial for learning to identify fake content, we commit to making our code freely available. Section 4.9 provides an analysis of audio detection of Exemplar Autoencoder fakes. (b) Another strategy for mitigating abuse is controlling access to private media data. Methods for image anonymization, via face detection and blurring, are widespread and crucial part of contemporary data collection, even mandated via the EU's General Data Protection Regulation (GDPR). In the US, audio restrictions are even more severe because of federal wiretapping regulations that prevent recordings of oral communications without prior consent [404]. Recent approaches for image anonymization make use of generative models that "de-identify" data without degradive blurring by retargeting each face to a generic identity (e.g., make everyone in a dataset look like PersonA) [285]. Our audio retargeting approach can potentially be used for audio de-identification by making everyone in a recording **sound** like PersonA. (c) Finally, we point out that audio recordings are currently used in critical societal settings including legal evidence and biometric voice-recognition (e.g., accessing one's bank account via automated recognition of speech over-the-phone [404]). Our results suggest that such use cases need to be re-evaluated and thoroughly tested in context of stylistic audio synthesis. In other terms, it is crucial for society to understand what information can be reliably deduced from audio, and our approach can be used to empirically explore this question.

Training datasets: Finally, we point out a unique property of our technical approach that differs prior approaches for audio and video content generation. Our exemplar approach reduces the reliance on population-scale training datasets. Facial synthesis models trained on existing large-scale datasets may be dominated by english-speaking celebrities. This may produce higher accuracy on sub-populations with skin tones, genders, and languages over-represented in the training dataset [289]. Exemplar-based learning may exhibit different properties because models are trained on the target exemplar individual. That said, we do find that pre-training on a single individual (but not a population) can speed up convergence of learning on the target individual. Because of the reduced dependency on population-scale training datasets (that may be dominated by English), exemplar models may better generalize across dialects and languages underrepresented in such datasets. We present results for stylized multilingual translations (retargeting Obama to speak

[5] https://www.blog.google/outreach-initiatives/google-news-initiative/advancing-research-fake-audio-detection/

[6] https://www.asvspoof.org/

in Mandarin and Hindi) without ever training on any Mandarin or Hindi speech.

4.9 Forensic Study

In the previous section, we outline both positive and negative outcomes of our research. Compared to prior art, most of the novel outcomes arise from using audio as an additional modality. As such, in this section, we conduct a study which illustrates that Exemplar Autoencoder fakes can be detected with high accuracy by a forensic classifier, particularly when trained on such fakes. We hope our study spurs additional forensic research on detection of manipulated audio-visual content.

Speaker Agnostic Discriminator: We begin by training a real/fake audio classifier on six identities from CelebAudio. The model structure is the same as [379]. The training datasets contains (1) Real speech of six identities: John F Kennedy, Alexei Efros, Nelson Mandela, Oprah Winfrey, Bill Clinton, and Takeo Kanade. Each has 200 sentences. (2) Fake speech: generated by turning each sentence in (1) into a different speaker in those six identities. There are a total of 1200 generated sentences. We test performance under two scenarios: (1) speaker within-the-training set: John F Kennedy, Alexei Efros, Nelson Mandela, Oprah Winfrey, Bill Clinton, and Takeo Kanade; (2) speaker-out-of-the-training set: Barack Obama, Theresa May. Each identity in the test set contains 100 real sentences and 100 fake ones. We ensure test speeches are disjoint from training speeches, even for the same speaker. Table 4.7 shows the classifier performs very well on detecting fake of within-set speakers, and is able to provide a reasonable reference for out-of-sample speakers.

Speaker-Specific Discriminator: We restrict our training set to one identity for specific fake detection. We train four exemplar autoencoders on four different speeches of Barack Obama to get different styles of the same person. The specific fake audio detector is trained on only one style of Obama, and is tested on all the four styles. The training set contains 600 sentences for either real or fake. Each style in the testing set contains 150 sentences for either real or fake. Table 4.7 shows our classifier provides reliable predictions on fake speeches even on out-of-set styles.

Speaker-Agnostic Discriminator	Accuracy(%)	Speaker-Specific Discriminator	Accuracy(%)
speaker within training set	99.0	**style within training set**	99.7
real	98.3	real	99.3
fake	99.7	fake	100
speaker out of training set	86.8	**style out of training set**	99.6
real	99.7	real	99.2
fake	74.0	fake	100

Table 4.7: **Fake Audio Discriminator: (a) Speaker Agnostic Discriminator** detects fake audio while agnostic of the speaker. We train a real/fake audio classifier on six identities from CelebAudio, and test it on both within-set speakers and out-of-set speakers. The training datasets contains (1) Real speech of six identities: John F Kennedy, Alexei Efros, Nelson Mandela, Oprah Winfrey, Bill Clinton, and Takeo Kanade. Each has 200 sentences. (2) Fake speech: generated by turning each sentence in (1) into a different speaker in those six identities. The testing set contains (1) speaker within the training set; (2) speaker out of the training set: Barack Obama, Theresa May. Each identity in the testing set contains 100 real sentences and 100 fake ones. For within-set speakers, results show our model can predict with very low error rate (1%). For out-of-set speakers, our model can still classify real speeches very well, and provide a reasonable prediction for fake detection. (**b**) **Speaker Specific Discriminator** detects fake audio of a specific speaker. We train a real/fake audio classifier on a specific style of Barack Obama, and test it on four styles of Obama (taken from speeches spanning different ambient environments and stylistic deliveries including presidential press conferences and university commencement speechs). The training set contains 1200 sentences evenly split between real and fake. Each style in the testing set contains 300 sentences evenly split as well. Results show our classifier provides reliable predictions on fake speeches even on out-of-sample styles.

Chapter 5

Semi-Parametric Multi-Modal Image Synthesis

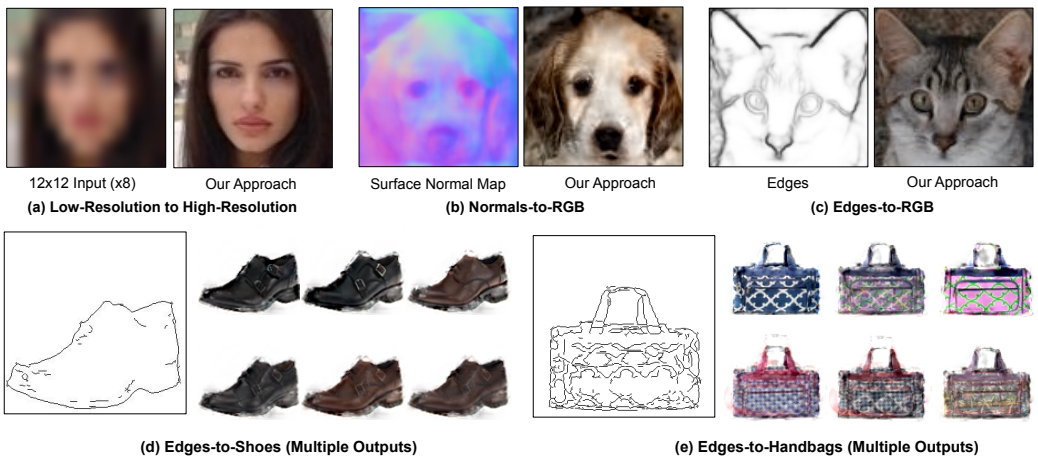


Figure 5.1: Our approach generates photorealistic output for various "incomplete" signals such as a low resolution image, a surface normal map, and edges/boundaries for human faces, cats, dogs, shoes, and handbags. Importantly, our approach can easily generate multiple outputs for a given input which was not possible in previous approaches [197] due to mode-collapse problem. Best viewed in electronic format.

5.1 Incomplete Input and Multiple Plausible Outputs

We consider the task of generating high-resolution photo-realistic images from *incomplete* input such as a low-resolution image, sketches, surface normal map, or

label mask. Such a task has a number of practical applications such as upsampling/colorizing legacy footage, texture synthesis for graphics applications, and semantic image understanding for vision through analysis-by-synthesis. These problems share a common underlying structure: a human/machine is given a signal that is missing considerable details, and the task is to reconstruct plausible details. Consider the edge map of cat in Figure 5.1-c. When we humans look at this edge map, we can easily imagine multiple variations of whiskers, eyes, and stripes that could be viable and pleasing to the eye. Indeed, the task of image synthesis has been well explored, not just for its practical applications but also for its aesthetic appeal.

GANs: Current state-of-the-art approaches rely on generative adversarial networks (GANs) [147], and most relevant to us, *conditional* GANS that generate image conditioned on an input signal [89,197,342]. We argue that there are two prominent limitations to such popular formalisms: (1) First and foremost, humans can imagine *multiple* plausible output images given a incomplete input. We see this rich space of potential outputs as a vital part of the human capacity to imagine and generate. Conditional GANs are in principle able to generate multiple outputs through the injection of noise, but in practice suffer from limited diversity (i.e., mode collapse) (Fig. 5.2). Recent approaches even remove the noise altogether, treating conditional image synthesis as regression problem [68]. (2) Deep networks are still difficult to explain or interpret, making the synthesized output difficult to modify. One implication is that users are not able to *control* the synthesized output. Moreover, the right mechanism for even specifying user constraints (e.g., "generate an cat image that looks like my cat") is unclear. This restricts applicability, particularly for graphics tasks.

Nearest-neighbors: To address these limitations, we appeal to a classic learning architecture that can naturally allow for multiple outputs and user-control: non-parametric models, or nearest-neighbors (NN). Though quite a classic approach [100, 128, 178, 204], it has largely been abandoned in recent history with the advent of deep architectures. Intuitively, NN works by requiring a large training set of pairs of (incomplete inputs, high-quality outputs), and works by simply matching the an incomplete query to the training set and returning the corresponding output. This trivially generalizes to multiple outputs through K-NN and allows for intuitive user control through on-the-fly modification of the training set - e.g., by restricting the training examplars to those that "look like my cat". In practice, there are several limitations in applying NN for conditional image synthesis. The first is a practical lack of training data. The second is a lack of an obvious distance metric. And the last is a computational challenge of scaling search to large training sets.

Approach: To reduce the dependency on training data, we take a *compositional* approach by matching *local* pixels instead of global images. This allows us to synthesize a face by matching "copy-pasting" the eye of one training image, the nose of another, etc. Compositions dramatically increases the representational power of our approach: given that we want to synthesize an image of K pixels using N

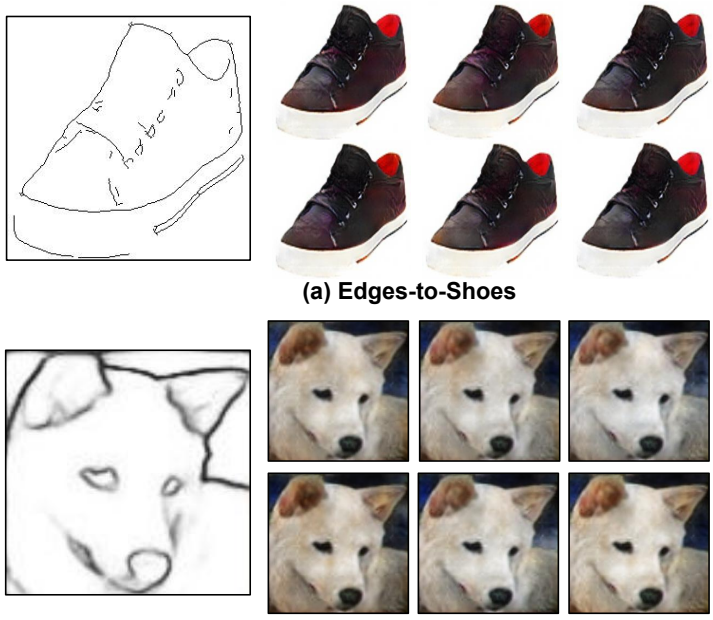


Figure 5.2: **Mode collapse problem for GANs:** We ran pix-to-pix pipeline of Isola et al. [197] 72 times. Despite the random noise set using dropout at test time, we observe similar output generated each time. Here we try to show 6 possible diverse examples of generation for a hand-picked best-looking output from Isola et al. [197].

training images (with K pixels each), we can synthesize an exponential number $(NK)^K$ of compositions, versus a linear number of global matches (N). A significant challenge, however, is defining an appropriate feature *descriptor* for matching pixels in the incomplete input signal. We would like to capture context (such that whisker pixels are matched only to other whiskers) while allowing for compositionality (left-facing whiskers may match to right-facing whiskers). To do so, we make use of deep features, as described below.

Pipeline: Our precise pipeline (Figure 5.3) works in two stages. (1) We first train an initial regressor (CNN) that maps the incomplete input into a single output image. This output image suffers from the aforementioned limitations - it is a single output that will tend to look like a "smoothed" average of all the potential images that could be generated. (2) We then perform nearest-neighbor queries on pixels *from this regressed output*. Importantly, pixels are matched (to regressed outputs from training data) using a multiscale deep descriptor that captures the appropriate level of context. This enjoys the aforementioned benefits - we can efficiently match to an exponential number of training examples in an interpretable and controllable manner. Finally, an interesting byproduct of our approach is the

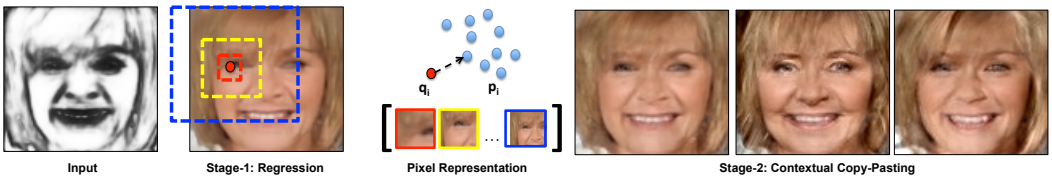


Figure 5.3: **Overview of pipeline:** Our approach is a two-stage pipeline. The first stage directly regresses an image from an incomplete input (using a CNN trained with l_2 loss). This image will tend to look like a "smoothed" average of all the potential images that could be generated. In the second stage, we look for matching pixels in similarly-smoothed training images. Importantly, we match pixels using multiscale descriptors that capture the appropriate levels of context (such that eye pixels tend to match only to eyes). To do so, we make use of off-the-shelf hypercolumn features extracted from a CNN trained for semantic pixel segmentation. By varying the size of the matched set of pixels, we can generate multiple outputs (on the right).

generation of dense, pixel-level correspondences from the training set to the final synthesized outputs.

5.2 Parametric and Non-Parametric Models

Our work is inspired by a large body of work on discriminative and generative models, nearest neighbors architectures, pixel-level tasks, and dense pixel-level correspondences. We provide a broad overview, focusing on those most relevant to our approach.

Synthesis with CNNs: Convolutional Neural Networks (CNNs) have enjoyed great success for various discriminative pixel-level tasks such as segmentation [21, 267], depth and surface normal estimation [21,23,102,103], semantic boundary detection [21,478] etc. Such networks are usually trained using standard losses (such as softmax or l_2 regression) on image-label data pairs. However, such networks do not typically perform well for the *inverse* problem of image synthesis from a (incomplete) label, though exceptions do exist [68]. A major innovation was the introduction of adversarially-trained generative networks (GANs) [147]. This formulation was hugely influential in computer visions, having been applied to various image generation tasks that condition on a low-resolution image [89, 244], segmentation mask [197], surface normal map [454] and other inputs [71, 189, 342, 468, 492, 512]. Most related to us is Isola et al. [197] who propose a general loss function for adversarial learning, applying it to a diverse set of image synthesis tasks.

Interpretability and user-control: Interpreting and explaining the outputs of generative deep networks is an open problem. As a community, we do not have a

clear understanding of what, where, and how outputs are generated. Our work is fundamentally based on *copy-pasting* information via nearest neighbors, which explicitly reveals how each pixel-level output is generated (by in turn revealing where it was copied from). This makes our synthesized outputs quite interpretable. One important consequence is the ability to intuitively edit and control the process of synthesis. Zhu et al. [510] provide a user with controls for editing image such as color, and outline. But instead of using a predefined set of editing operations, we allow a user to have an *arbitrarily*-fine level of control through on-the-fly editing of the exemplar set (E.g., "resynthesize an image using the eye from this image and the nose from that one").

Correspondence: An important byproduct of pixelwise NN is the generation of pixelwise correspondences between the synthesized output and training examples. Establishing such pixel-level correspondence has been one of the core challenges in computer vision [76, 214, 256, 268, 460, 504, 507]. Tappen et al. [415] use SIFT flow [256] to hallucinate details for image super-resolution. Zhou et al. [507] propose a CNN to predict appearance flow that can be used to transfer information from input views to synthesize a new view. Kanazawa et al. [214] generate 3D reconstructions by training a CNN to learn correspondence between object instances. Our work follows from the crucial observation of Long et al. [268], who suggest that features from pre-trained convnets can also be used for pixel-level correspondences. In this work, we make an additional empirical observation: hypercolumn features trained for semantic segmentation learn nuances and details better than one trained for image classification. This finding helped us to establish semantic correspondences between the pixels in query and training images, and enabled us to extract high-frequency information from the training examples to synthesize a new image from a given input.

Nonparametrics: Our work closely follows data-driven approaches that make use of nearest neighbors [100, 128, 167, 204, 353, 390]. Hays and Efros [167] match a query image to 2 million training images for various tasks such as image completion. We make use of dramatically smaller training sets by allowing for compositional matches. Liu et al. [254] propose a two-step pipeline for face hallucination where global constraints capture overall structure, and local constraints produce photorealistic local features. While they focus on the task of facial super-resolution, we address variety of synthesis applications. Final, our compositional approach is inspired by Boiman and Irani [42, 43], who reconstruct a query image via compositions of training examples.

5.3 PixelNN: One-to-Many Mappings

We define the problem of conditional image synthesis as follows: given an input x to be conditioned on (such as an edge map, normal depth map, or low-resolution image), synthesize a high-quality output image(s). To describe our approach, we fo-

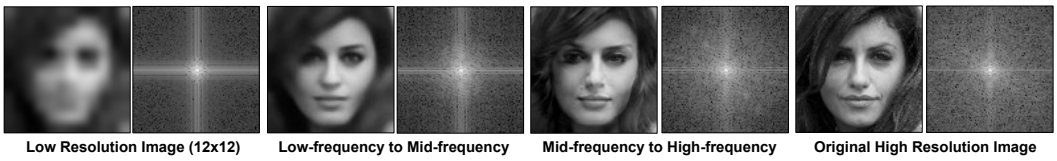


Figure 5.4: **Frequency Analysis:** We show the image and its corresponding Fourier spectrum. Note how the frequency spectrum improve as we move from left to right. The Fourier spectrum of our final output closely matches that of original high resolution image.

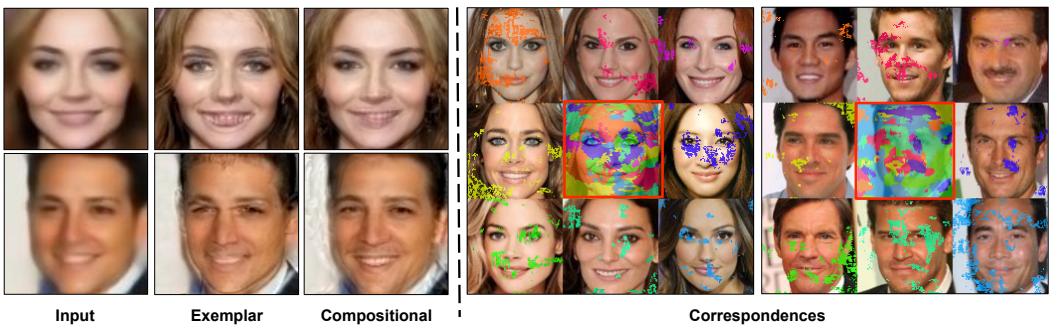


Figure 5.5: **Exemplar vs. Compositional:** Given the output of Stage-1 for low-resolution to high-resolution, we compare an exemplar output with a compositional output. On the right hand side, we show correspondences and how 8 global nearest neighbors have been used to extract different parts to generate output. As an example, the right eye of first image is copied from the bottom-left nearest neighbor; and the nose and mouth of second image is copied from the top-middle nearest neighbor.

cus on illustrative task of image super-resolution, where the input is a low-resolution image. We assume we are given training pairs of input/outputs, written as (x_n, y_n). The simplest approach would be formulating this task as a (nonlinear) regression problem:

$$\min_{w} ||w||^2 + \sum_{n} ||y_n - f(x_n; w)||_{l_2} \tag{5.1}$$

where $f(x_n; w)$ refers to the output of an arbitrary (possibly nonlinear) regressor parameterized with w. In our formulation, we use a fully-convolutional neural net – specifically, PixelNet [21] – as our nonlinear regressor. For our purposes, this regressor could be any trainable black-box mapping function. But crucially, such functions generate *one-to-one* mappings, while our underlying thesis is that conditional image synthesis should generate *many* mappings from an input. By treating

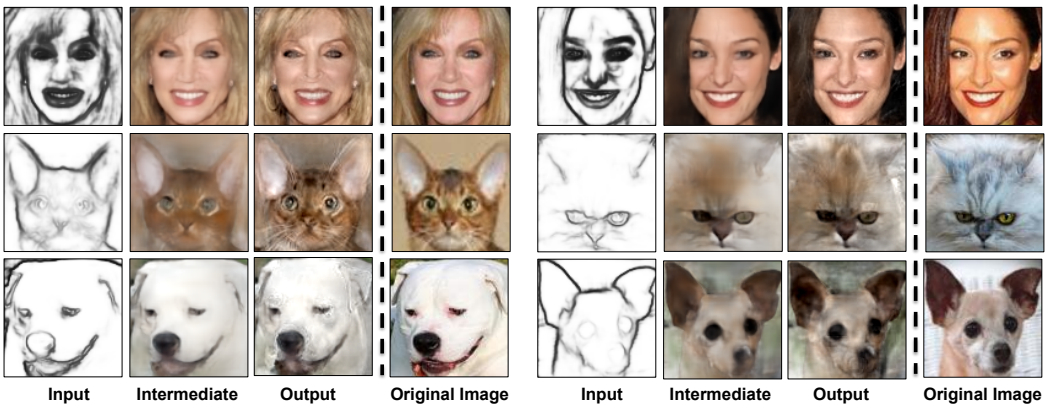


Figure 5.6: **Edges to RGB:** Our approach used for faces, cats, and dogs to generate RGB maps for a given edge map as input. One output was picked from the multiple generations.

synthesis as a regression problem, it is well-known that outputs tend to be over-smoothed [203]. In the context of the image colorization task (where the input is a grayscale image), such outputs tend to desaturated [241, 493].

Frequency analysis: Let us analyze this smoothing a bit further. Predicted outputs $f(x)$ (we drop the dependance on w to simplify notation) are particularly straightforward to analyze in the context of super-resolution (where the conditional input x is a low-resolution image). Given a low-resolution image of a face, there may exist multiple textures (e.g., wrinkles) or subtle shape cues (e.g., of local features such as noses) that could be reasonably generated as output. In practice, this set of outputs tends to be "blurred" into a single output returned by a regressor. This can be readably seen in a frequency analysis of the input, output, and original target image (Fig. 5.4). In general, we see that the regressor generates mid-frequencies fairly well, but fails to return much high-frequency content. We make the operational assumption that a single output suffices for mid-frequency output, but *multiple* outputs are required to capture the space of possible high-frequency textures.

Exemplar Matching: To capture multiple possible outputs, we appeal to a classic non-parametric approaches in computer vision. We note that a simple K-nearest-neighbor (KNN) algorithm has the trivially ability to report back K outputs. However, rather than using a KNN model to return an entire image, we can use it to predict the (multiple possible) high-frequencies missing from $f(x)$:

$$\begin{aligned} Global(x) &= f(x) + \Big(y_k - f(x_k)\Big) \qquad \text{where} \\ k &=_n \text{Dist}\Big(f(x), f(x_n)\Big) \end{aligned} \tag{5.2}$$

where $Dist$ is some distance function measuring similarity between two (mid-frequency)

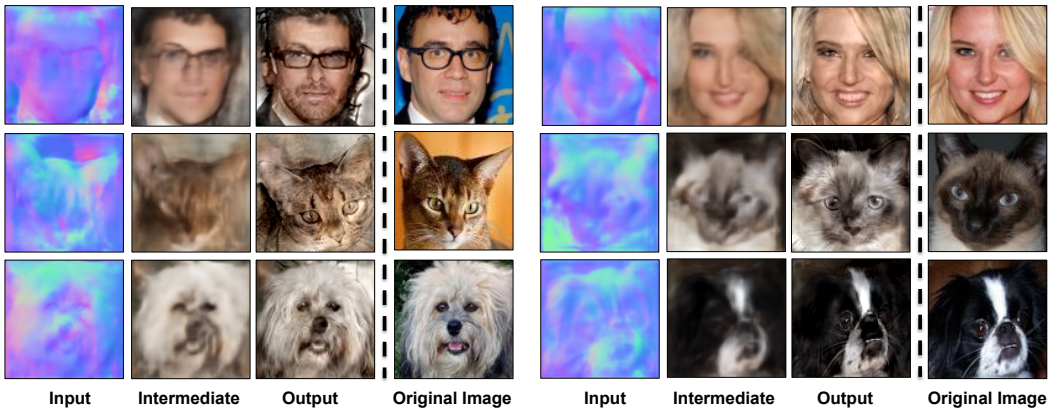


Figure 5.7: **Normals to RGB:** Our approach used for faces, cats, and dogs to generate RGB maps for a given surface normal map as input. One output was picked from multiple generations.

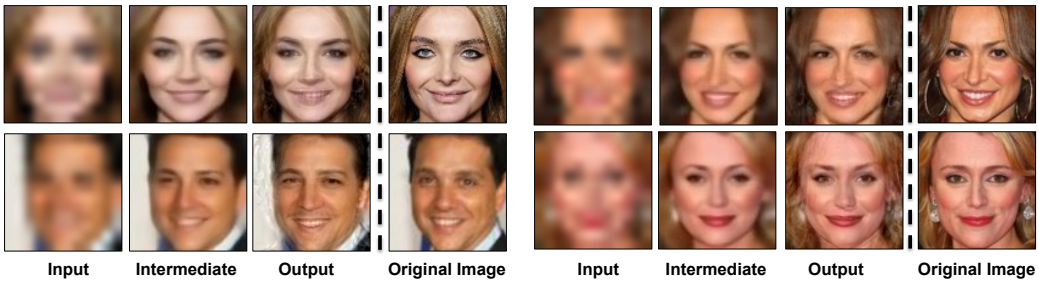


Figure 5.8: **Low-Resolution to High-Resolution:** We used our approach for hallucinating 96×96 images from an input 12×12 low-resolution image. One output was picked from multiple generations.

reconstructions. To generate multiple outputs, one can report back the K best matches from the training set instead of the overall best match.

Compositional Matching: However, the above is limited to report back high frequency images in the training set. As we previously argued, we can synthesize a much larger set of outputs by *copying* and *pasting* (high-frequency) patches from the training set. To allow for such compositional matchings, we simply match individual pixels rather than global images. Writing $f_i(x)$ for the i^{th} pixel in the reconstructed image, the final composed output can be written as:

$$Comp_i(x) = f_i(x) + \Big(y_{jk} - f_j(x_k)\Big) \qquad \text{where}$$
$$(j,k) =_{m,n} \text{Dist}\Big(f_i(x), f_m(x_n)\Big) \tag{5.3}$$

where y_{jk} refers to the output pixel j in training example k.

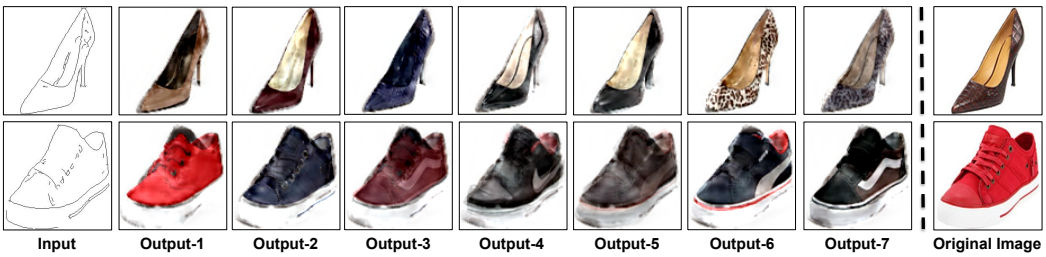


Figure 5.9: **Edges-to-Shoes:** Our approach used to generate multiple outputs of shoes from the edges. We picked seven distinct examples from multiple generations.

Figure 5.10: **Edges-to-Bags:** Our approach used to generate multiple outputs of bags from the edges. We picked seven distinct examples from multiple generations.

Distance functions: A crucial question in non-parametric matching is the choice of distance function. To compare global images, contemporary approaches tend to learn a deep embedding where similarity is preserved [32, 73, 267]. Distance functions for pixels are much more subtle (5.3). In theory, one could also learn a metric for pixel matching, but this requires large-scale training data with dense pixel-level correspondances.

Pixel representations: Suppose we are trying to generate the left corner of an eye. If our distance function takes into account only local information around the corner, we might mistakenly match to the other eye or mouth. If our distance function takes into account only global information, then compositional matching reduces to global (exemplar) matching. Instead, we exploit the insight from previous works that different layers of a deep network tend to capture different amounts of spatial context (due to varying receptive fields) [21, 23, 162, 345, 378]. Hypercolumn descriptors [162] aggregate such information across multiple layers into a highly accurate, *multi-scale* pixel representation (visualized in Fig. 5.3). We construct a pixel descriptor using features from conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ for a PixelNet model trained for semantic segmentation (on PASCAL Context [302]). To measure pixel similarity, we compute cosine distances between two descriptors. We visualize the compositional matches (and associated correspondences) in Figure. 5.5.

Finally, Figure 5.6, Figure 5.7, and Figure 5.8 shows the output of our approach for various input modalities.

Efficient search: We have so far avoided the question of run-time while doing nearest neighbor search in pixel-space. Run-time performance is another reason why generative models are more popular than nearest neighbors. To speed up search, we made some non-linear approximations: Given a reconstructed image $f(x)$, we first (1) find the global K nearest neighbors using *conv-5* features and then (2) search for pixel-wise matches only in a $T \times T$ pixel window around pixel i in this set of K images. In practice, we vary K from $\{1, 2, .., 10\}$ and T from $\{1, 3, 5, 10, 96\}$ and generate 72 candidate outputs for a given input. Because the size of synthesized image is 96×96, our search parameters include both a fully-compositional output $(K = 10, T = 96)$ and a fully global exemplar match $(K = 1, T = 1)$ as candidate outputs. Figure 5.9, Figure 5.10, and Figure 5.11 show examples of multiple outputs generated using our approach by simply varying these parameters.

5.4 Quantitative Analysis

We now present our findings for multiple modalities such as a low-resolution image $(12 \times 12$ image), a surface normal map, and edges/boundaries for domains such as human faces, cats, dogs, handbags, and shoes. We compare our approach both quantitatively and qualitatively with the recent work of Isola et al. [197] that use generative adversarial networks for pixel-to-pixel translation.

Dataset: We conduct experiments for human faces, cats and dogs, shoes, and handbags using various modalities.

Human Faces We use $100,000$ images from the training set of CelebA dataset [261] to train a regression model and do nearest neighbors. We used the subset of test images to evaluate our approach. The images were resized to 96×96 following Gucluturk et al. [154].

Cats and Dogs: We use $3,686$ images of cats and dogs from the Oxford-IIIT Pet dataset [329]. Of these $3,000$ images were used for training, and remaining 686 for evaluation. We used the bounding box annotation made available by Parkhi et al. [329] to extract head of the cats and dogs.

For human faces, and cats and dogs, we used the pre-trained PixelNet [21] to extract surface normal and edge maps. We did not do any post-processing (NMS) to the outputs of edge detection.

Shoes & Handbags: We followed Isola et al. [197] for this setting. $50,000$ training images of shoes were used from [488], and $137,000$ images of Amazon handbags from [510]. The edge maps for this data was computed using HED [478] by Isola et al. [197].

Qualitative Evaluation: Figure 5.12 shows the comparison of our nearest-neighbor based approach (**PixelNN**) with Isola et al. [197] (**Pix-to-Pix**).

Quantitative Evaluation: We quantitatively evaluate our approach to measure if

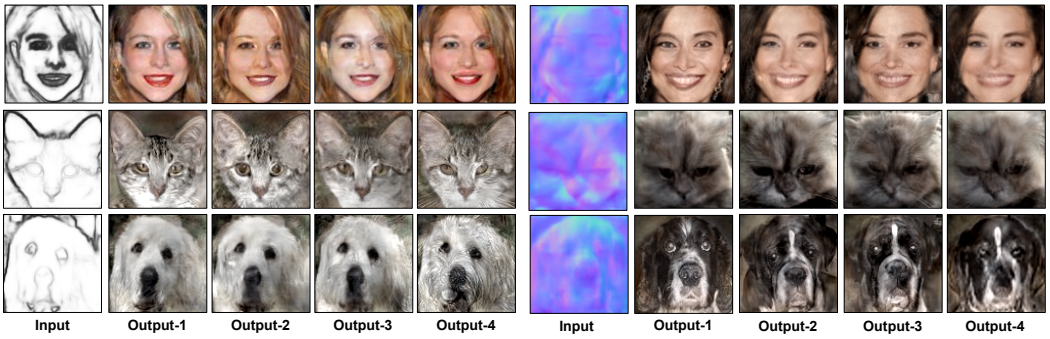


Figure 5.11: **Multiple Outputs for Edges/Normals to RGB:** Our approach used to generate multiple outputs of faces, cats, and dogs from the edges/normals. As an example, note how the subtle details such as eyes, stripes, and whiskers of cat (left) that could not be inferred from the edge map are different in multiple generations.

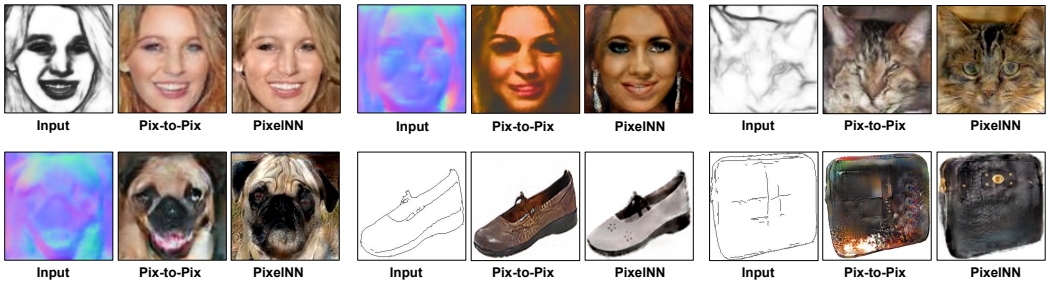


Figure 5.12: Comparison of our approach with Pix-to-Pix [197].

	Mean	Median	RMSE	11.25°	22.5°	30°	AP
Human Faces							
Pix-to-Pix [197]	17.2	14.3	21.0	37.2	74.7	86.8	0.34
Pix-to-Pix [197] (Oracle)	15.8	13.1	19.4	41.9	78.5	89.3	0.34
PixelNN (Rand-1)	12.8	10.4	16.0	54.2	86.6	94.1	0.38
PixelNN (Oracle)	**10.8**	**8.7**	**13.5**	**63.7**	**91.6**	**96.7**	**0.42**
Cats and Dogs							
Pix-to-Pix [197]	14.7	12.8	17.5	42.6	82.5	92.9	0.82
Pix-to-Pix [197] (Oracle)	**13.2**	**11.4**	**15.7**	**49.2**	**87.1**	**95.3**	0.85
PixelNN (Rand-1)	16.6	14.3	19.8	36.8	76.2	88.8	0.80
PixelNN (Oracle)	13.8	11.9	16.6	46.9	84.9	94.1	**0.92**

Table 5.1: **Normals-to-RGB** We compared our approach, PixelNN, with the GAN-based formulation of Isola et al. [197] for human faces, and cats and dogs. We used an off-the-shelf PixelNet model trained for surface normal estimation and edge detection. We use the output from real images as ground truth surface normal and edge map respectively.

	AP	Mean	Median	RMSE	11.25°	22.5°	30°
Human Faces							
Pix-to-Pix [197]	0.35	12.1	9.6	15.5	58.1	88.1	94.7
Pix-to-Pix [197] (Oracle)	0.35	11.5	9.1	14.6	61.1	89.7	95.6
PixelNN (Rand-1)	0.38	13.3	10.6	16.8	52.9	85.0	92.9
PixelNN (Oracle)	**0.41**	**11.3**	**9.0**	**14.4**	**61.6**	**90.0**	**95.7**
Cats and Dogs							
Pix-to-Pix [197]	0.78	18.2	16.0	21.8	32.4	71.0	85.1
Pix-to-Pix [197] (Oracle)	0.81	16.5	14.2	19.8	37.2	76.4	89.0
PixelNN (Rand-1)	0.77	18.9	16.4	22.5	30.3	68.9	83.5
PixelNN (Oracle)	**0.89**	**16.3**	**14.1**	**19.6**	**37.6**	**77.0**	**89.4**

Table 5.2: **Edges-to-RGB:** We compared our approach, PixelNN, with the GAN-based formulation of Isola et al. [197] for human faces, and cats and dogs. We used an off-the-shelf PixelNet model trained for edge detection and surface normal estimation. We use the output from real images as ground truth edges and surface normal map.

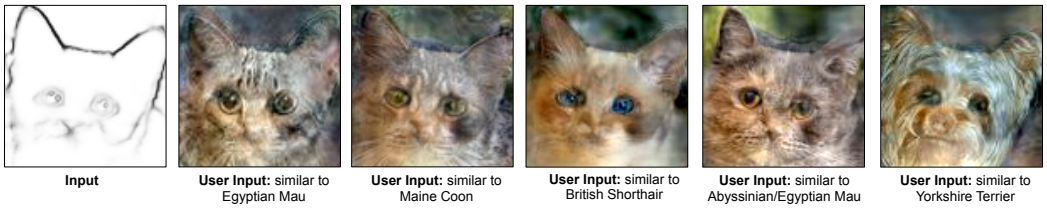


Figure 5.13: **Controllable synthesis:** We generate the output of cats given a user input from a edge map. From the edge map, we do not know what type of cat it is. A user can suggest what kind of the output they would like, and our approach can copy-paste the information.

our generated outputs for human faces, cats and dogs can be used to determine surface normal and edges from an off-the-shelf trained PixelNet [21] model for surface normal estimation and edge detection. The outputs from the real images are considered as ground truth for evaluation as it gives an indication of how far are we from them. Somewhat similar approach is used by Isola et al. [197] to measure their synthesized cityscape outputs and compare against the output using real world images, and Wang and Gupta [454] for object detection evaluation.

We compute six statistics, previously used by [23,102,123,452], over the angular error between the normals from a synthesized image and normals from real image to evaluate the performance – **Mean**, **Median**, **RMSE**, **11.25°**, **22.5°**, and **30°** – The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better. We evaluate the edge detection performance

using average precision (**AP**).

Table 5.1 and Table 5.2 quantitatively shows the performance of our approach with [197]. Our approach generates multiple outputs and we do not have any direct way of ranking the outputs, therefore we show the performance using a random selection from one of 72 outputs, and an oracle selecting the best output. To do a fair comparison, we ran trained models for Pix-to-Pix [197] 72 times and used an oracle for selecting the best output as well. We observe that our approach generates better multiple outputs as performance improves significantly from a random selection to oracle as compared with Isola et al. [197]. Our approach, though based on simple nearest neighbors, achieves result quantitatively and qualitatively competitive (and many times better than) with state-of-the-art models based on GANs and produce outputs close to natural images.

5.4.1 Human-Controllable Synthesis

Finally, NN provides a user with intuitive control over the synthesis process. We explore a simple approach based on *on-the-fly pruning* of the training set. Instead of matching to the entire training library, a user can specify a subset of relevant training examples. Figure 5.13 shows an example of controllable synthesis. A user "instructs" the system to generate an image that looks like a particular cat-breed by either denoting the subset of training examplars (e.g., through a subcategory label), or providing an image that can be used to construct an on-the-fly neighbor set.

5.4.2 Limitations of Semi-Parametric Representation

Failure cases: Our approach mostly fails when there are no suitable nearest neighbors to extract the information from. Figure 5.14 shows some example failure cases of our approach. One way to deal with this problem is to do exhaustive pixel-wise nearest neighbor search but that would increase the run-time to generate the output. We believe that system-level optimization such as Scanner[1], may potentially be useful in improving the run-time performance for pixel-wise nearest neighbors.

5.5 Beyond Synthesis: Understanding CNNs

We extend PixelNN to Compositional Nearest Neighbor (CompNN), a simple approach to visually interpret the representations learned by a CNN for pixel-level tasks. Different from existing visual interpretation methods [39, 122, 237, 277, 318], CompNN reconstructs a query image by retrieving similar image patches from training exemplars and rearranging them spatially to match the query. To do so, CompNN matches image patches using feature embeddings extracted from different layers of the CNN. Because such a reconstruction provides patch correspon-

[1] https://github.com/scanner-research/scanner

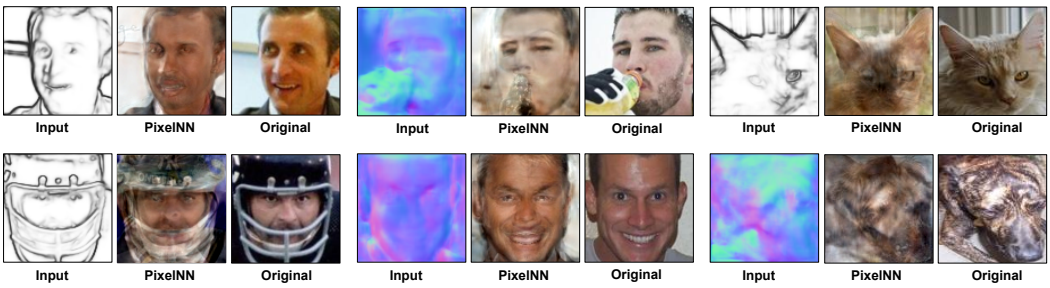


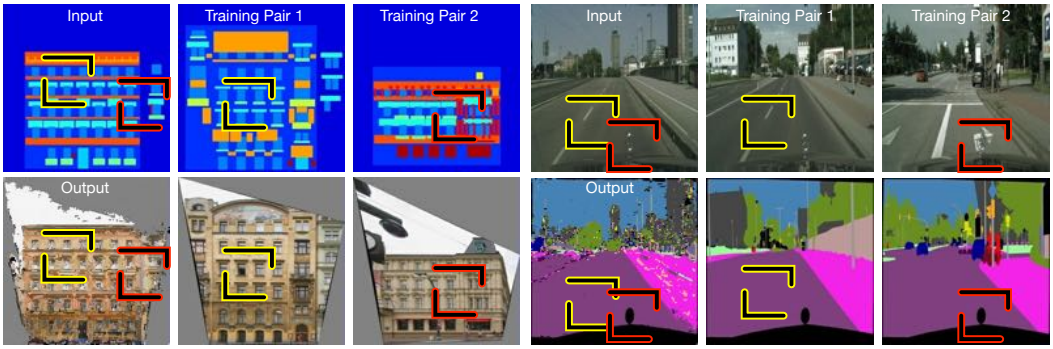
Figure 5.14: **Failure Cases:** We show some failure cases for different input types. Our approach mostly fails when it is not able to find suitable nearest neighbors.

dences between the query image and training exemplars, they can also be used to transfer labels from training exemplars. From this perspective, CompNN also provides a reconstruction of the CNN output. While CompNN can operate by exhaustively searching for the most similar patches, it uses a patch-match-like [28] algorithm that uses patch representations extracted from the learned embeddings by the CNN. This patch-match mechanism allows CompNN to efficiently compute patch correspondences between an input image and images from the training set. CompNN computes these correspondences efficiently because the learned embeddings by the CNN can vary smoothly: an input patch centered at (x, y) with a corresponding patch from a training image centered at (i, j) likely has a neighbor patch centered at $(x + 1, y + 1)$ with a corresponding patch centered at $(i + 1, j + 1)$. This property is crucial for a patch-match-like algorithm since it exploits this property to speed-up the nearest-neighbor search.

In contrast to interpreting the embeddings by retrieving the most similar instance from the training set, composing an image by arranging image patches from a training set enables an exponential range of possible images that can be generated. This is because CompNN has at hand $(NK)^K$ image patches from a training set, where K is the number of patches one can extract from an image and N is the number of training images. Furthermore, patch correspondences are useful because not only they enable the reconstruction of both the input and output images, but also allow a user to understand how a CNN may behave on never-before-seen data, establish semantic correspondences between a pair of images, and generate an output image with different properties by changing the set of images in the training set. This latter feature is useful to understand and possibly correct the implicit bias in a CNN.

5.5.1 CompNN: Compositional Nearest Neighbors

The general idea of Compositional Nearest Neighbors (CompNN) is to establish patch correspondences between the input image and patches from images in the



(a) Labels to Images **(b)** Images to Labels

Figure 5.15: Overview of the steps in Compositional Nearest Neighbors (CompNN) for (**a**) labels to images and (**b**) images to labels. Given an input patch (double-colored-stroked squares) in the top-left image at the top row, CompNN searches in the training set for the most similar patches (top row of training pairs 1 & 2 columns). Then, to reconstruct the output image, CompNN copies the patches from to target images (bottom row of training pairs 1 & 2 columns) and pastes them in the canvas of the output images. CompNN applies the same procedure to reconstruct the input image but copies information from the input training images. Patch correspondences are color coded.

training set. Given these correspondences, CompNN will sample patches from the training set and assemble them to generate a coherent image. Because our focus is to study encoder-decoder architectures that map an image into another image domain, we assume that the training set contains pairs of images that are aligned pixel-wise. For instance, in the segmentation problem, every pixel in the input image is assigned a class label. Thanks to this setting, CompNN can generate images for resembling the CNN's input and output images.

Establishing patch correspondences requires a representation of the patches and a similarity or distance function. Similar to a Nearest Neighbor (NN) approach, CompNN will search for patches in the training set that are the most similar or proximal in the patch representation space. We use the learned embeddings learned by the network to represent patches and a cosine distance to compare these representations. Fig. 5.15 illustrates the overall steps that CompNN performs to compute patch correspondences and generating images.

5.5.2 Extracting Patch Representations from the CNN Embeddings

A key ingredient of CompNN is the patch representation. We extract patch representations from the activation tensors of a particular layer in a CNN with an encoder-decoder architecture. We assume that a CNN for pixel-level tasks or spa-

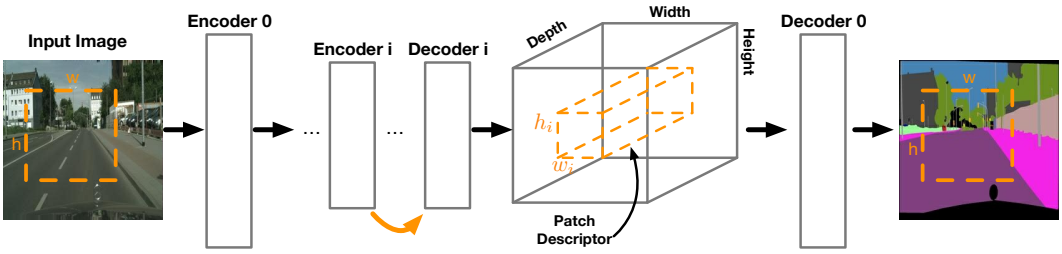


Figure 5.16: We extract patch representations from the activation tensors of a particular layer in the network by grabbing "hyperpatches" (dashed orange rectangular prism). The width w_i and height h_i of these hyperpatches at layer i depend on the filter sizes of layer i and layer type. To extract a patch representation from a decoder layer, we identify the entries in the activation tensor that contribute to the pixels of the patch in the output image. For instance, a 2×2 patch in the output image requires a $2 \times 2 \times d_1$ hyperpatch from the activation tensor that is the input to Decoder 0 layer, assuming that Decoder 0 layer applies transposed-convolution using 2×2 filters. When the hyperpatch belongs to the encoder, then we identify its corresponding decoder layer (orange arrow) and use the same hyperpatch dimensions.

tial predictions learns an embedding for every layer in the network. We consider a layer in this work to involve convolutional layers, batch normalizations, and pooling operations. We represent an input image patch with a sub-tensor or "hyperpatch" with dimensions (h_i, w_i, d_i) from the activation tensor of the i-th layer. Fig. 5.16 illustrates the setting and computation of the patch representation as well as the patch in the input image. If the activation tensor for the layer has dimensions of (H_i, W_i, D_i), then $h_i \leq H_i$, $w_i \leq W_i$, and $d_i = D_i$.

In practice, we first determine the minimal patch size that is constrained by the first encoding layer. For instance, if the first encoder layer convolves with 2×2 filters, then the patch size of that layer is 2×2. Then, we identify the corresponding decoder layer of the first encoder layer and we calculate the hyperpatch dimensions as follows. First, we use the activation tensor which is the input to the identified decoder layer. From this activation tensor, we identify the entries that contribute to the production of the output patch. For instance, the hyperpatch dimension to represent a 2×2 patch in the output image from the last decoder layer is $2 \times 2 \times d_1$, where d_1 is the depth of the input tensor to the last decoder layer that uses transposed convolution with 2×2 filters. Since many architectures downsample by half in their encoder sections, then the patch sizes double in size until reaching the bottleneck of the architecture. For this context, the hyperpatch dimensions for the remaining decoder layers stay the same. For instance, a 4×4 patch corresponding to the second encoder layer uses a $2 \times 2 \times d_2$ hyperpatch from the activation tensor which is the input to the second to last decoder layer.

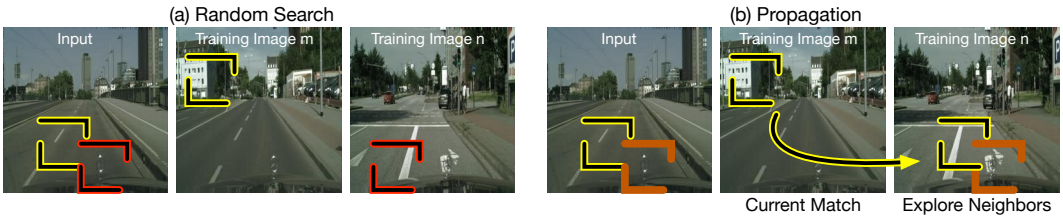


Figure 5.17: HyperPatch-Match steps. (**a**) Random search: Given the query patches (top-left squares), this step selects patches and images from the training set at random; this step is used to initialize the algorithm. (**b**) Propagation: Given the query patch in yellow, propagation examines the correspondence of a neighbor patch (orange square). If the neighbor patch (orange square) in image n produces a best similarity value, then propagation updates the correspondence datum for the yellow patch.

5.5.3 Computing Patch Correspondences

The simplest method to establish patch correspondences is by means of an exhaustive search: given an input patch representation, search for the most similiar or proximal patch representation from the same layer that the input patch representation was extracted. As discussed earlier, we use the cosine distance to measure similarity or proximity. This approach mimics 2D convolution. This is because of two reasons. First, this exhaustive search compares the hyperpatch representing a patch in the input image with all possible hyperpatches from an activation tensor at a particular layer. Second, the cosine distance involves a dot product of normalized hypterpatches, which in turn can be implemented as a convolution operation since it is a linear operation. Although this search is highly parallelizable, its drawback is its $\mathcal{O}(h_i w_i d_i)$ computational cost.

To alleviate this computational cost, we approximate the exhaustive search. Inspired by the Patch-Match [28], we developed HyperPatch-Match, a method that approximates the exhaustive search by exploiting the smooth variation that natural scenes possess. This smooth variation ensures with high probability that an image patch from the input image centered at pixel $q_1 = (x, y)$ with a corresponding patch from the training set centered at $t_1 = (i, j)$ has a neighbor patch (e.g., one centered at $q_2 = (x + 1, y + 1)$) with a corresponding patch in the training set that is a neigbhor of the patch centered at t_1 (e.g., a neighbor centered at $t_2 = (i + 1, j + 1)$). Unlike Patch-Match that computes patch correspondences between two images in the original input image space (e.g., the RGB space), HyperPatch-Match computes correspondences across a set of several training images with their respective hyperpatches.

Similar to Patch-Match, HyperPatch-Match has two main steps: random search and propagation. The random search step randomly selects an image from the train-

ing set and a patch from the selected image. If the selected patch is most similar to the current similar patch found, then HyperPatch-Match updates the correspondence for a given input patch by storing the id of the training image and the center of the patch. On the other hand, the propagation step uses the smooth variation of natural scenes, which is also maintained in the activation tensors as shown experimentally in Section 5.6. Given a query patch from the input image centered at $q = (x, y)$, the propagation step retrieves the current correspondence from a neighbor patch, e.g., $q' = (x+1, y+1) \leftrightarrow t = (i, j, l)$, where (i, j) is the center of the patch from the l-th training image. Then, the propagation step checks if the cosine distance between q and $t' = (i - i, j - 1, l)$ produce a smaller distance than the current one stored for q. If this is the case, the propagation step updates the correspondence datum. HyperPatch-Match initializes the correspondences at random and repeats these steps for several iterations. While this approach empirically shows faster results than an exhaustive search, HyperPatch-Match potentially still needs to check several images from the training set at random. Note that the random step is the one in charge of exploring new images in the training set, while the propagation step only exploits the smooth variation property. Fig. 5.17 illustrates the steps involved in the proposed HyperPatch-Match method.

To accelerate HyperPatch-Match even more, we used one of the activation tensors from the middle layers as a global image descriptor to select the top k most similar images and utilize them as the training set for a given query input image. To select the top k global nearest neighbors, we compare the global image descriptor of the input image with all the global descriptors of each of the images in the training set, and keep the k most similar images. In this way, HyperPatch-Match reduces the set size of the training images to consider while barely affecting its performance.

5.5.4 Interpreting Pixel-Level CNN Embeddings

Similar to other interpretation methods [31, 276, 490, 500], CompNN assumes that each layer of the CNN computes an embedding for the input image. Unlike the existing interpretation methods, CompNN focuses on interpreting the embeddings learned by CNNs devised for spatial predictions or pixel-level tasks. CompNN can be interpreted as an inversion method [277]. This is because CompNN reconstructs the input image given the patch representations extracted from the embeddings learned at every layer. Unlike existing inversion methods that learn functions that recover the input image from a representation, CompNN inverts or reconstructs the input image by exploiting patch correspondences between the input image and images in the training set. While CompNN can reconstruct the input image, it also can generate an output image resembling the output of the CNN. Unlike previous interpretation methods that only focus on understanding representations for image classification, CompNN aims to understand the embeddings that enable the underlying CNN for spatial predictions tasks to synthesize images.

5.6 Input and Output Image Reconstructions

In this section we present qualitative and quantitative experiments that assess the reconstructions for the input and output images of the underlying CNN. For these experiments we used the U-Net-based Pix2Pix [197] network. Note that the layers of generator are named by Encoder 1-7 following with Decoder 7-1.

To get all the patch representations, we used the publicly available[2] pre-trained models for the facades and cityscapes datasets. Because we are interested in interpreting the embeddings that each of the layers of a CNN learn, we extracted all the patch representations for every encoder and decoder layers of the training and validation sets; see Sec. 5.5.2 for details on how to extract patch representations from the activation tensors at each layer.

Given these patch representations, we used HyperPatch-Match to establish correspondences and reconstruct both input and output images, as described in Section 5.5.3. We used the top 16 global nearest neighbors to constrain the set used as the training set for every image in the validation set. To select these global nearest neighbors, we used the whole tensor from the Decoder-7 (bottleneck feature) as the global image descriptor. Also, since HyperPatch-Match is an iterative algorithm, we allowed it to run for 1024 iterations.

The results of the input reconstructions are shown in Fig. 5.18. The first two rows show reconstructions for a labels-to-image task, while the bottom two rows show reconstructions for an image-to-labels task. The left column in this Figure shows the inputs to the network, and the remaining columns show reconstructions computed from three layers belonging to the encoder and decoder parts of the CNN. Given that Pix2Pix uses skips connections, i.e., the output of an encoder layer is concatenated to the output of a decoder layer to form the input of the subsequent decoder layer, Fig. 5.18 shows reconstructions of the corresponding encoder and decoder layers. This means that the activations of Encoder 1 are part of the activations of Decoder 2.

We can observe in Fig. 5.18 that the first layers of the encoder produce a good quality reconstruction of the input, an explanation for this observation is that much of the information to reconstruct the input is still present in the first encoder layer because only a layer of 2D convolutions has been applied. Surprisingly, the last layers of the decoder still produce good quality reconstructions. This can be attributed to the skip connections making both layers (Encoder 1 and Decoder 2) possess the same amount of information to reconstruct a good quality image. On the other hand, the reconstructions from the inner layers tend to maintain the structure of the input image, but the quality of the reconstructions decays as we get closer to the middle of the U-Net architecture.

The results of the output reconstructions are shown in Fig. 5.19. The column in the left shows the output image generated by the underlying CNN. The organiza-

[2]Pix2Pix: https://github.com/affinelayer/pix2pix-tensorflow

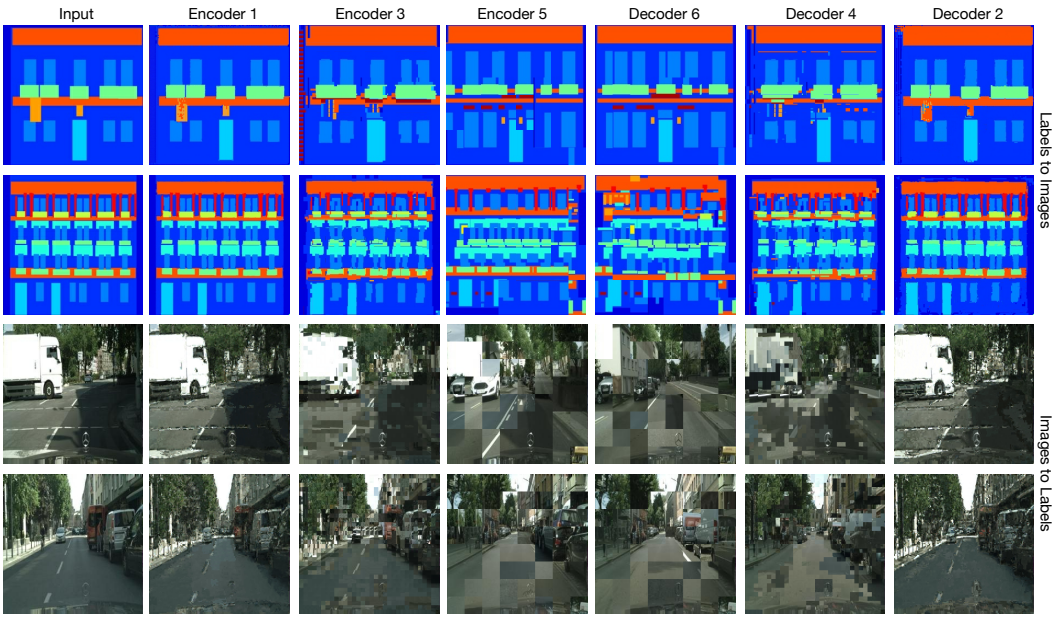


Figure 5.18: Input reconstructions using patch representations from encoder and decoder layers. We can observe that the first layers of the encoder and the last layers of the decoder produce a good quality reconstruction of the input. On the other hand, the reconstructions from the inner layers tend to maintain the structure of the input image, but the quality of the reconstructions decays as we get closer to the middle layers.

tion of the remaining columns is the same as that of the Fig. 5.18. We can observe that the reconstructions from the encoder layers present an overall structure of the output image but lack details that the output image possess. For instance, consider the first row. The reconstructions from the encoder layers preserve the structure of the facade, i.e., the color and location of windows and doors. On the other hand, the reconstructions of the decoder layer possess structural information and more details present in the output of the network. For instance, in the first row, the output image contains a wedge depicting the sky. That part is present in the reconstructions of the decoder layers, but it is not present in the reconstructions of the encoder layers. Despite the fact that we use an approximation method to speed up the nearest neighbor search, we can observe that the reconstruction from the last decoder layer resembles well the image generated by the CNN.

Unlike existing interpretation methods that aim to reconstruct the input image only, CompNN is also capable of synthesizing an output image that resembles the generated image by the CNN. This is an important feature because it allows us to interpret the embeddings in charge of generating the synthesized image. Moreover, these results show that the smoothness variation of natural scenes is still present in

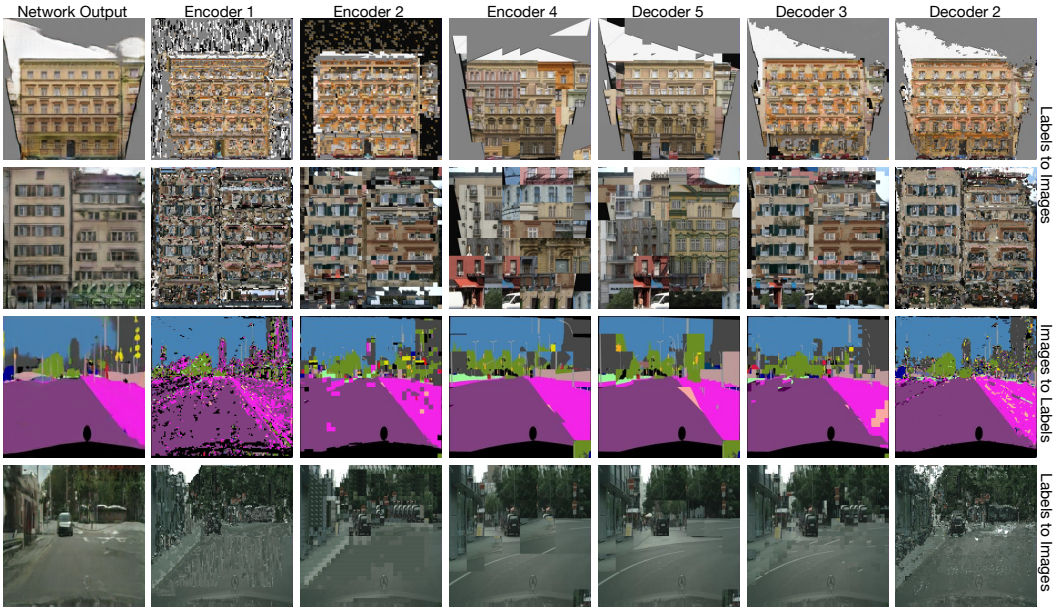


Figure 5.19: Output reconstructions using patch representations from encoder and decoder layers. We can observe that the reconstructions from the encoder layers present an overall structure of the output image but lack details that the output image possess. On the other hand, the reconstructions of the decoder layer possess structural information and more details present in the output of the network.

the activation tensors of both encoder and decoder layers. Thanks to this property, CompNN is able to compute correspondences efficiently using HyperPatch-Match and compose an output image.

To evaluate the reconstructions in a quantitative way, we use the reconstructions from images to labels and assess them by Mean Pixel Accuracy and Mean Intersection over Union (IoU). For this experiment, we consider Pix2Pix on Facades and Cityscapes datasets as well as SegNet [14] on the Camvid dataset for the images-to-labels task. Also, we used HyperPatch-Match to compute correspondences for the Pix2Pix representations, and used an exhaustive search to compute the correspondences for the SegNet representations. The synthesized labeled-images used patch representations from the Decoder 2 layer for the Pix2Pix network, and Decoder 4 or the last layer before the softmax layer for the SegNet network. We trained a SegNet model for the Camvid dataset using a publicly available tensorflow implementation [3].

The results for the output reconstructions are shown in Table 5.3. The rows of the Table show from top to bottom the metrics for the CNN synthesized image (the baseline); GR (Bottleneck), a reconstruction which simply returns the most similar

[3]SegNet Implementation: https://github.com/tkuanlun350/Tensorflow-SegNet

Approach	(*Mean Pixel Accuracy*)			(*Mean IoU*)		
	Facades	Cityscapes	Camvid	Facades	Cityscapes	Camvid
Baseline CNN	0.5105	0.7618	0.7900	0.2218	0.2097	0.4443
GR (Bottleneck)	-0.1963	-0.1488	-0.2200	-0.1437	-0.0735	-0.1981
GR (FC7)	-0.1730	-0.1333	-0.1263	-0.1126	-0.0702	-0.1358
OR (top 1)	**-0.0102**	-0.0545	**-0.0350**	-0.0253	-0.0277	**-0.0720**
OR (top 16)	+0.0324	-0.0218	–	**+0.0214**	+0.0014	–
OR (top 32)	+0.0336	-0.0182	–	+0.0232	**+0.0011**	–
OR (top 64)	+0.0343	**-0.0171**	–	+0.0246	+0.0020	–

Table 5.3: Quantitative evidence that CompNN can reconstruct very similar output images when compared with those of the network. The Table shows metric differences between the output of the CNN (the baseline) and CompNN reconstructions (OR) and global reconstructions (GR), which simply return the most similar output image from the training set. Bold entries indicate the closest reconstruction to the baseline.

Approach	IR (top 1)	IR (top 16)	IR (top 32)	IR (top 64)
Facades	0.723	0.837	0.844	0.846
Cityscapes	0.816	0.894	0.898	0.901

Table 5.4: Mean Pixel Accuracy (MPA) for input reconstructions. We compare the input reconstructions with the original input label images. These results show that there is a good agreement between the input reconstructions and the original input label images.

image in the training set using as global feature the bottleneck activation tensor; GR (FC7), a global reconstructions using the whole activation tensor of the penultimate layer; and output reconstructions (OR) using top 1, 16, 32, and 64 global images as the constrained training set for CompNN and HyperPatch-Match. The result for Camvid dataset which uses exhaustive search is placed in the OR (top 1) row. The entries in the Table show the metric differences between the reconstructions and the baseline, and we show in bold the numbers that are closest to the baseline. We can observe in Table 5.3 that the compositional reconstructions overall tend to be close enough to those of the baseline. This is because the absolute value of the differences is small overall. Moreover, we can observe that considering more images in the training set for HyperPatch-Match tends to improve results; compare Facades and Cityscapes columns.

To evaluate the input reconstructions, we utilized a similar approach used to evaluate the output image reconstructions. However, in this case we compared the input reconstructions with the original input label images only for a Pix2Pix network for labels-to-images task on Facades and Cityscapes datasets. We computed their agreement by means of the Mean Pixel Accuracy (MPA) metric, and we show the results in Table 5.4. Note that MPA compares class labels assigned to every pixel.

We can observe that the MPA is overall high (> 0.7). In particular, we can observe that considering more top k images in the training set for HyperPatch-Match increases the similarity between the reconstruction and the original input image.

Part III

Human-Controllable Representations

Chapter 6

Human-Controllable Image Synthesis

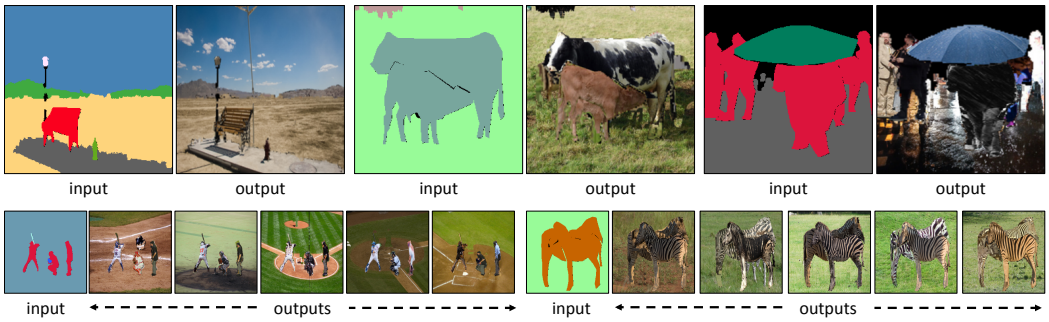


Figure 6.1: Our approach synthesizes images from label maps by **non-parametric matching** of shapes, parts, and pixels. We show example results for diverse "in-the-wild" scenes containing large amounts of variation in object composition and deformation. We also show multiple outputs generated from a label map using our approach in the bottom row.

6.1 In-the-Wild Image Synthesis

We introduce a data-driven approach for interactively synthesizing diverse images from semantic label maps. Specifically, we seek to design a system for *in-the-wild* image synthesis that is controllable and interpretable. While content creation is a compelling task in of itself (a classic goal of computer graphics), image synthesis is also useful for generating data that can be used to train discriminative visual recognition systems [181]. Synthesized data can be used to explore scenarios that are difficult or too dangerous to sample directly (e.g., training an autonomous percep-

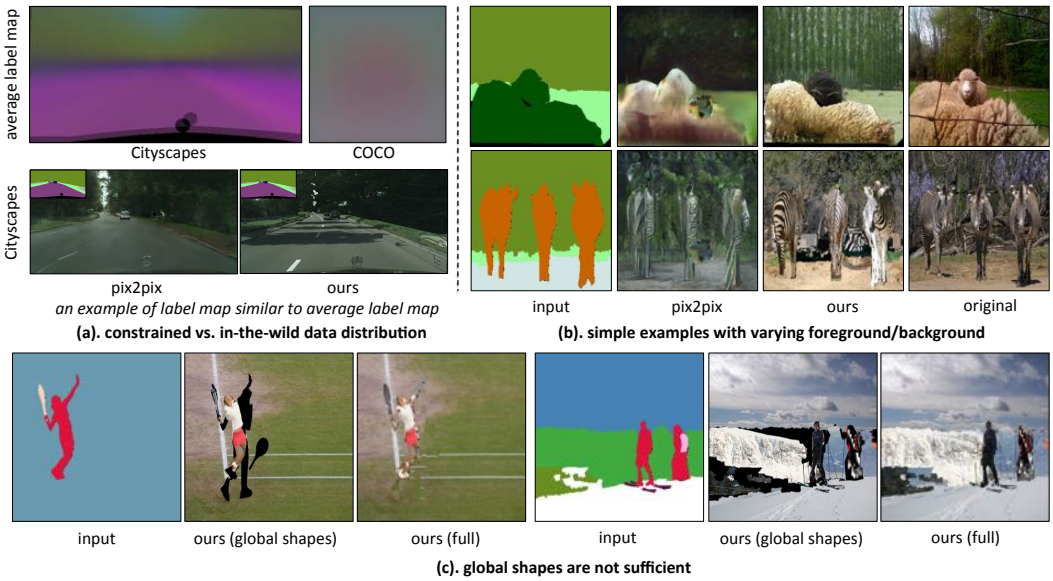


Figure 6.2: **Limitations of current approaches for images synthesis:** (**a**) Current image synthesis models tend to be trained on datasets with somewhat limited diversity, such as Cityscapes [80], faces [399], or facades [434]. For example, the average label mask for Cityscapes [80] reveals redundant structure such as a car hood, road, and foliage. In contrast, the average image for COCO [253] is much less structured, suggesting it is a more varied dataset. (**b**) Indeed, we train state-of-the-art neural architectures [197, 450] on COCO and observe poor convergence (even after a month of training!) resulting in a mode collapsed and averaged outputs. (**c**) In contrast, our simple matching-based approach can synthesize realistic image content by matching to exemplar shapes. To generate high-quality images, we find it crucial to encode scene context and part deformation in the matching process - i.e., matching global shapes alone will produce poor images with missing regions due to shape mismatches.

tion system on unsafe urban scenes [188]). Figure 6.1 shows images synthesized using our approach, where the input is a semantic label map.

Parametric vs Nonparametric: Current approaches for image synthesis and editing can be broadly classified into two categories. The first category uses parametric machine learning models. The current state-of-the-art [22, 68, 197, 450] relies on deep neural networks [242] trained with adversarial losses (GANs) [147] or perceptual losses [203] to create images. These approaches work remarkably well when trained on datasets with somewhat limited diversity, such as Cityscapes [80], faces [399], or facades [434]. It is unclear how to extend such approaches for "in-the-wild" image synthesis or editing: parametric models trained on one data distribution (e.g. Cityscapes) do not seem to generalize to others (e.g. facades), a

problem widely known as dataset bias [426]. The second category of work [98, 99, 178, 240, 364] uses non-parametric nearest neighbors to create content. These approaches have been demonstrated on interactive image editing tasks such as object insertion [240] or scene completion [167]. Though a large inspiration [178] for our own work, such approaches have interestingly fallen out of favor in recent history.

Does more data help? A peculiar property of many parametric synthesis methods is that they do better with *less* data [22, 147, 197, 342, 492, 512]. The culprit seems to be that such methods don't do well on diverse training sets, and in practice larger training sets tend to be diverse. This behavior is in contrast with truly non-parametric methods that do better with *more* data [167]. Figure 6.2-(a)-(b) highlights the differences between limited and diverse datasets, using illustrative examples of Cityscapes [80] and COCO [253]. While parametric methods do well on limited data distributions, they struggle to perform on diverse datasets. Recent works [49, 328] have attempted to overcome this challenge by using enormously large model sizes and massive compute (see concurrent work from Park et al [328] for more discussion on parametric approaches).

Composition by parts: In this work, we make three observations that influence our final approach; (1) humans can imagine *multiple* plausible output images given a particular input label mask. We see this rich space of potential outputs as a vital part of the human capacity to imagine and generate. Most parametric networks tend to formulate synthesis as a one-to-one mapping problem and struggle to provide diverse outputs (a phenomenon also known as mode collapse). Important exceptions include [24, 68, 136, 513] that generated multiple outputs by employing various modifications. (2) Visual scenes are exponentially complex due to many possible compositions of constituent objects and parts. It is tempting to combine both observations and generate multiple outputs by composing scene elements together. But these compositions cannot be arbitrary - one cannot freely swap out a face with a wheel, or place an elephant on a baseball field. Our matching process makes use of implicit *contextual* semantics present in the library of exemplar label masks. (3) Given an exemplar set of sufficient size, nearest neighbor methods may still perform well with simple features that are not learned (e.g., pixel values). We combine our observations to construct an image synthesis system that relies on exemplar shapes and parts using simple pixel features.

Our Contributions: (1) We study the problem of visual content creation and manipulation for in-the-wild settings and observe that reliance on parametric models lead to averaged or mode-collapsed outputs. (2) We present an approach that utilizes shapes and context to generate images consisting of rigid and non-rigid objects in varied backgrounds, and different environmental and illumination conditions. (3) We demonstrate the controllable and interpretable aspects of our approach that enables a user to influence the generation and select examples from many outputs.

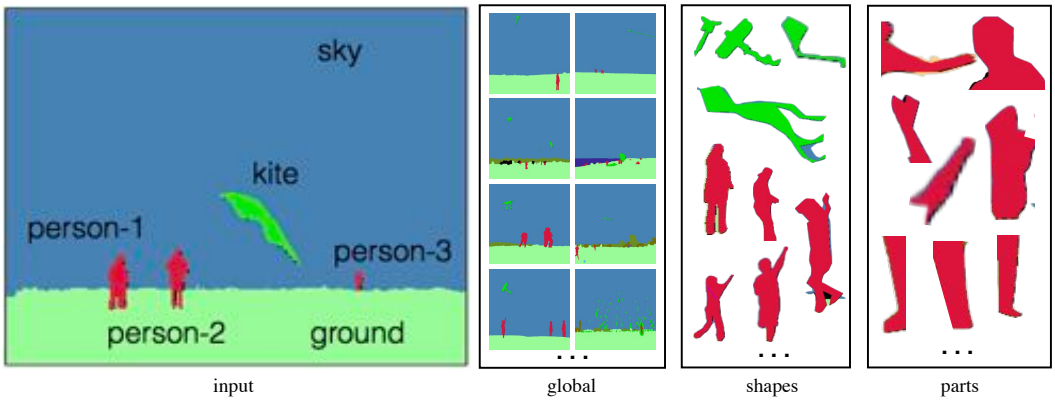


Figure 6.3: **Overview:** Given a label map (left), our goal is to generate multiple plausible images. We make use of hierarchical matching for efficient retrieval of images, shapes, and parts.

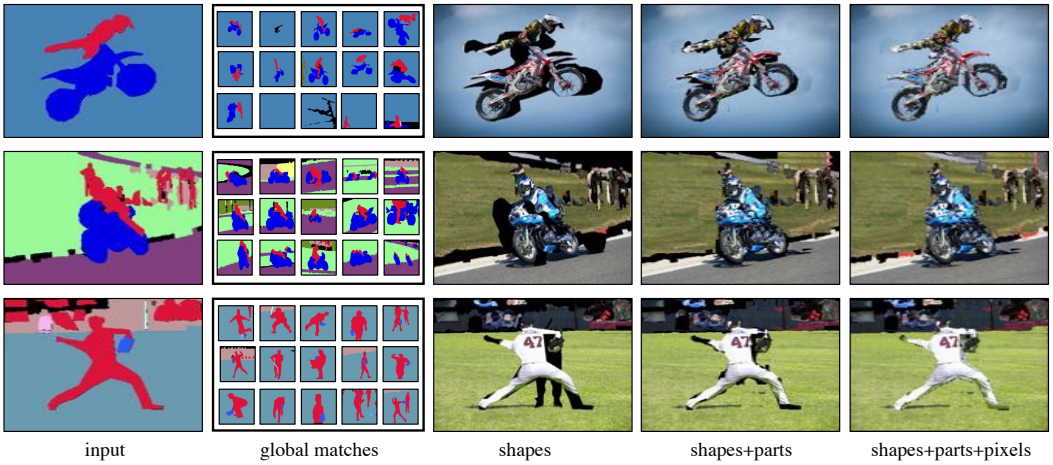


Figure 6.4: **Hierarchical Composition:** Given an input label map (left), we show the outputs of four stages of our non-parametric matching approach. The second column shows the top-15 global matches for the input. The third column shows the output of the composition of global shapes extracted using our shape consistency algorithm. The fourth column shows the improved output by introducing local part consistency to the previous output. Finally, minute pixel-level holes are filled by our pixel-consistency algorithm.

6.2 Review: Hierarchical Composition

Our work combines various ideas on shapes, deformable parts, context, and non-parametric approaches developed in the last two decades. We position each sepa-

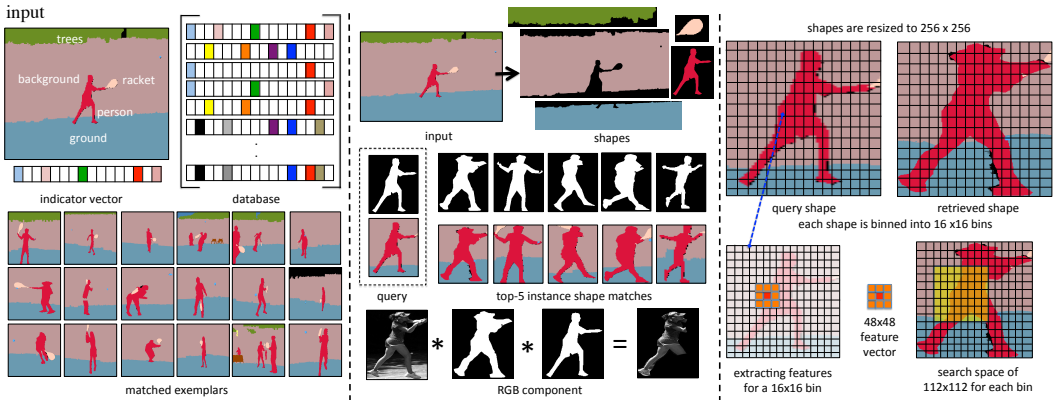


Figure 6.5: **Four stages of non-parametric matching:** (1) **Global Consistency** - Given a semantic label map, we using an indicator vector that tells what categories are present. This indicator vector helps us to quickly find the relevant examples from database. This prunes away 99% of exemplar database to get required shapes. (2) **Shape Consistency** - We extract various shapes from the input label mask, and match query shape to shapes within the exemplar set by using a shape-and-context feature. We show examples of top-5 retrieved shapes for a query shape on left. The image information from the retrieved shapes is then extracted by considering the mask of query shape and retrieved shape; (3) **Part Consistency** - We observe that global shape retrieved in last stage is missing information about the hands and legs of the query shape (human in this case). We define a local shape matching approach that looks in the neighborhood to synthesize parts. The query and top-k shapes are resized to 256×256, and binned into 16×16 bins with each bin being a 16×16 patch. Each patch is represented by label information contained in it, and an additional 8 neighboring patches. This provides contextual information about the surroundings. The parts are looked in an adjacent 112×112 region and the ones with minimum hamming distance is considered. **Pixel Consistency** algorithm follows a similar setup. See section 6.3 for more details.

rately and the specific insights for their particular usage.

Shapes: Shapes [232,281,357] emerge naturally in our world due to its compositional structure. If we had an infinite data-source with all potential shapes for all the objects, then our world could be represented by a linear combination of different shapes [158,357]. In this work, we aim to generate images from semantic and instance label maps as input. Meaningful shapes and contours [33,161] makes for an obvious interpretation for such an input.

Non-parametric approaches: In an unconstrained in-the-wild data distribution consisting of both rigid and non-rigid objects, it becomes non-trivial to model such shapes for a one-to-many mappings. We, therefore, want to leverage the shape

information explicitly from the training data in our formulation by simple *copy-pasting*. Non-parametric methods [98–100,128,178,204] have been used for texture-synthesis [99,100], image super-resolution [128], action recognition [98], or scene completion [167].

Scene Composites: Our work draws similarity to idea of scene composites [196, 339,364,365]. Russell et al [364] use shapes or scene composites to query matches for semantic segmentation using LabelMe dataset [363]. In another work, Russell et al [365] use a similar idea of composites for object discovery. Isola and Liu [196] use this idea of composites for scene parsing and collaging. Recently, Qi et al [339] used shapes in a semi-parametric form to synthesize images from a semantic label map. These different approaches [339,364] on scene composites or shapes are however constrained to rigid and non-deformable objects from a constrained data-distribution such as road-side scenarios from LabelMe [363], or Cityscapes [80]. Our work extends the prior work to non-rigid and deformable shapes from an unconstrained in-the-wild data distribution. Figure 6.2-(c) shows how global shapes are insufficient, and one needs to consider local information about parts and pixels.

Deformable Objects & Parts: The global shape fitting can be reliably estimated for non-deformable objects but local shapes or *parts* [38,113,153] are required when considering non-rigid objects. Prior work on local components [38], regions [153], or parts [113,396,484] has largely been focused on recognition. On the other hand, our work draws insight from ideas on compositional matching [42,109], and we use the parts, components, and regions for synthesizing images. In this work, we relax strict shape matching to do local part generation from various global shapes to do image synthesis. This enables us to consider local information without any explicit part labels.

Context: Context is a natural and powerful tool to put things in perspective [37, 182]. There is a wide literature on the use of context in computer vision community [92,302], and is beyond the scope of this work to illustrate them completely. In this work, the contextual information comes for free with our input, i.e., semantic label map. We use this context at both the global and local level to do better and faster matching of global shapes, parts, and pixels. The contextual information, while itself remaining in background, enables us to do an effective non-parametric matching.

User Control: Multiple works [24, 28, 240, 450, 495, 510] in computer graphics and vision literature have demonstrated the importance of user-controlled image operations. Grab-cut [361] enables user-based segmentation of a given scene. Lalonde et al [240] use a non-parametric approach to insert objects in a given image. Kholgade et al [225] demonstrate a user-controlled 3D object manipulation. In this work, we demonstrate how shapes can be used naturally and intuitively for a user-controllable content creation and manipulation.

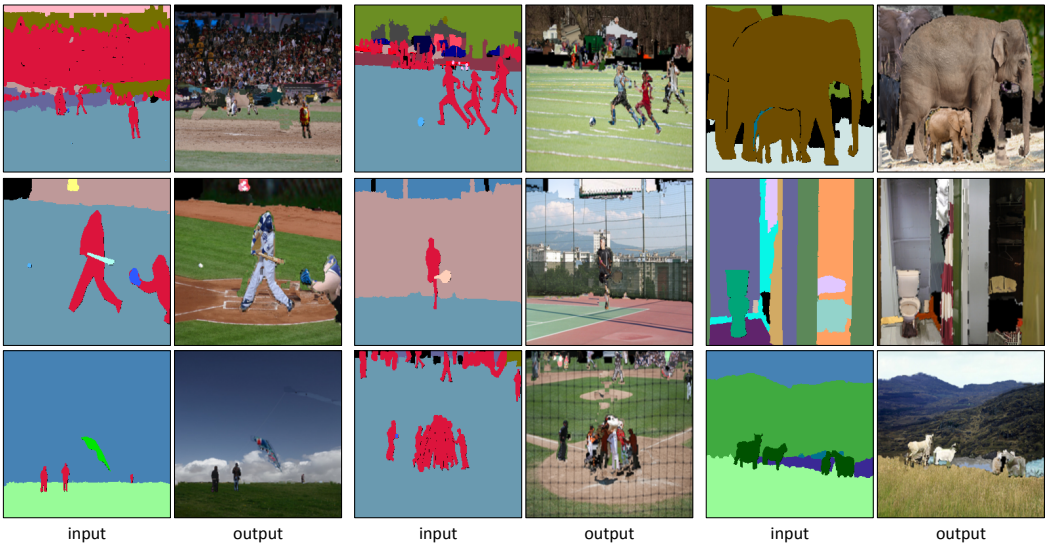


Figure 6.6: **Non-parametric matching:** Our approach generates images from a semantic label map by non-parametric matching of global shapes, local parts, and pixel-consistency. The above examples contain varying background, cluttered environment, varying weather and illumination conditions, and multiple rigid and non-rigid objects in various shapes and forms.

6.3 Shapes and Context

Given a semantic and an instance label map, X, our goal is to synthesize a new image, Y. Our formulation is a hierarchical non-parametric matching ensuring the following stages in order: (1) global scene consistency; (2) instance shape consistency; (3) local part consistency; and finally (4) minute pixel-level consistency. Figure 6.3 gives an overview of our approach.

Global Scene Consistency: In a big data setup with a hundred thousand and million examples, doing nearest neighbors could be a time-consuming process. We make this process faster by using a global indicator vector (or scene context) to prune the list of training examples from which the shapes should be extracted. Only those examples are considered if they fall in one of three categories: (1) their global image has the same labels as input; (2) the labels in input is its subset; (3) the labels in input is its superset. This reduces the search space from a hundred thousand shapes to a few hundreds. We further prune them to top-N images for searching shapes by computing a global coverage and a pixel coverage score.

A **global coverage** score is computed to ensure the top-N label maps in the training set have similar distribution of labels as are in a given query label map. We compute the normalized histogram of labels (both query and training) and compute a l_2 distance between query and training label map. A **pixel coverage** score is com-

puted to ensure we select the images with maximum pixel-to-pixel overlap. This score is computed by aligning a query label map and an example from the training set, followed by the Hamming distance between them. To make it faster, we resize the images to 100×100 and then compute the normalized Hamming distance between the respective labels. We sum both global coverage and pixel coverage scores and choose N images in training set with the lowest scores. This use of global scene context drastically reduces the search space for our non-parametric approach and enables us to synthesize on everyday computing devices (single-core CPUs instead of GPUs). Figure 6.5-(left) shows similar examples retrieved by simple matching of indicator vector.

Instance Shape Consistency: We seek shapes as the first step to define different components in an image. We are given a shape with a binary mask and semantic labels. We make a tight bounding box around it and resize it to 50×50. This allows us to use the shape as a filter to retrieve similar shapes within the exemplar set. We represent the filter using: (1) a simple **mask operator**: the part belonging to the shape is set to 1, and the remaining part is set to -1. This enables us to match shapes with similar boundaries and details; and (2) a **contextual operator**: we use semantic labels for this filter that helps us in getting better matches with a similar context. We convert the filter to a 2500-dimensional vector.

In summary, we have a mask operator, m, where $m[k] \in \{-1, 1\}$ represents the value of the k^{th}-index of a 2500-dimensional vector; and similarly, a contextual operator, v, where $v[k]$ has the associated semantic label. We match a query shape, (m_q, v_q), with a i^{th} shape, (m_i, v_i), in the exemplar set using the scoring function:

$$S_{shape}\Big((m_q, v_q), (m_i, v_i)\Big) = \sum_{k=1}^{2500} \Big(m_q[k].m_i[k] + I(v_q[k] = v_i[k])\Big), \tag{6.1}$$

where I is an indicator function with value 1 when true and zero otherwise. We ignore the shapes from the exemplar set when the ratio of their aspect-ratio to that of query shape is either less than 0.5 or greater than 2. Using fixed size filters and low-res label masks helps us to generate a composite of arbitrarily high resolution without any extra computational cost. The RGB component for an extracted shape is its intersection with the query shape, i.e., only pixels active in both extracted and query shape are considered. Figure 6.5-(middle) shows this part of our algorithm.

Local Part Consistency: Modeling occlusions and deformable aspects of non-rigid objects in the real world are extremely hard. The problem is even aggravated with noisy shape inputs. The insufficient shape data and non-rigid objects in the real world leads to parts and local regions [38]. We seek parts from a few top-scoring global shapes. Importantly, the part information is required when a global shape is not able to capture. We extract the knowledge of parts from the global shapes in a spirit similar to non-parametric texture synthesis [100]. The shape components are resized to 256×256 so that local information can be well searched.

Figure 6.7: **Parametric vs. Non-parametric:** We generate images from an input label map. The second column shows the output of Pix2Pix [197] trained on COCO training set. The third column shows the output of our non-parametric hierarchical matching approach.

We extract a 16×16 patch from the resized global shape template. Local contextual information (similar to HOG [81], or Group-Normalization [469]) is used by considering the neighboring 8 patches. A patch (p) is, therefore, represented by a $256\times9 = 2304$-dimensional vector containing the label information in its region and surroundings, and $p[k]$ means k^{th}-index of this 2304-dimensional vector. Given a query part, p_q, and i^{th} part, p_i, from a selected shape. The parts are scored using:

$$S_{part}(p_q, p_i) = \sum_{k=1}^{2304} I(p_q[k] = p_i[k]). \tag{6.2}$$

Importantly, we do not need to look in a larger window for part matching as we have weakly aligned global shapes. Therefore, we restrict the patches to look in a surrounding 5×5 patch window. This corresponds to the 112×112 pixel window in a resized global shape template. We copy the RGB component from the best matching patch window. Figure 6.5-(right) shows the part of our algorithm to compute part-consistency.

Minute Pixel Consistency: The shapes and parts have accounted for most of the non-parametric image synthesis. However, they do not ensure pixel-level consistency and often end up with minor holes in an image. We enforce a pixel-level consistency in this process to account for the remaining holes in the synthesized

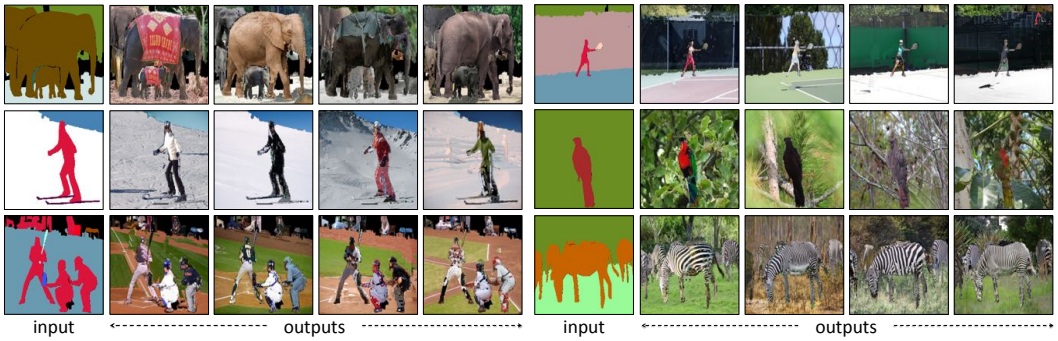


Figure 6.8: **Multiple Outputs:** Our approach can easily generate an exponentially large number of outputs by changing shapes and parts. We show four outputs generated for each label map.

image. This process is similar to our part consistency algorithm, except that it is done on every pixel. Each pixel is represented by a surrounding 11×11 window. We use the criterion in Eq. 6.2 to compute the similarity between two feature vectors. To expedite this matching, we compute features for a low-res input label map (128×128) as pixel consistency is ensured to fill minor holes alone. Finally, we look in the surrounding region of 5×5 from a 128×128 image to fill in the information as global and local consistency have already been accounted for by shape and part consistency.

Hierarchical Composition: We combine the information hierarchically from shapes, parts, and pixels to generate a full image. Figure 6.4 shows the composition starting from an input label map. Firstly, we find relevant examples using the global scene context. We then use the instance shape components to fill the major chunk of the image. The missing information is then filled using the local part consistency. Finally, the minor holes are filled using the pixel-level consistency. The combination of these four stages enables us to efficiently generate an image from an input label mask by simple non-parametric matching.

Qualitative Results: Figure 6.6 shows outputs generated by our approach for varying background, cluttered environment, varying weather and illumination conditions, and multiple rigid and non-rigid objects in various shapes and forms. Figure 6.7 contrasts our approach with Pix2Pix [197].

Multiple-Outputs: A salient aspect of considering shapes and parts in a non-parametric matching provides multiple outputs for free. We can combine various extracted shapes in exponential ways without any extra overhead. We show multiple examples synthesized for a given label map using our approach in Figure 6.8. Generating these multiple outputs is not trivial when using parametric approaches [68, 197], and has been a subject of multiple studies [136, 513].

User Control: We finally demonstrate the applicability of our approach for

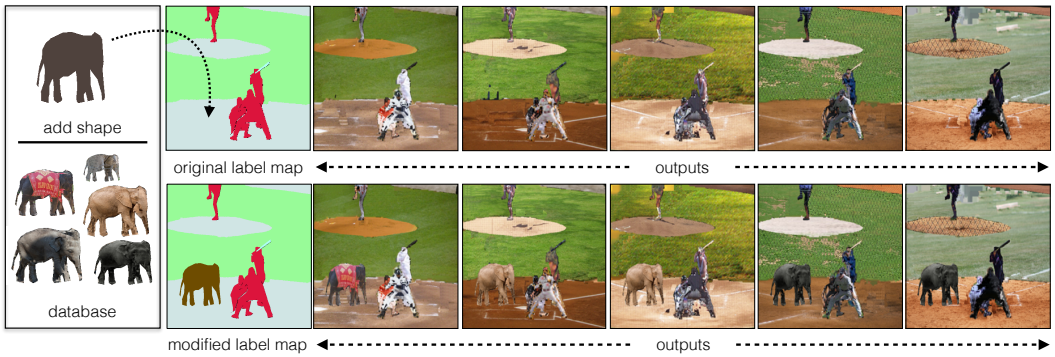


Figure 6.9: **User-Intervention & Manipulation:** The first row shows multiple outputs generated for a given label map. We insert the shape of *elephant* in this input label map to synthesize an *elephant on the baseball field*. The second row shows multiple outputs for the modified label map.

a user-controllable content creation in Figure 6.9. Even though it is rare to see *an elephant on the baseball field*, our method can quickly generate such an example by inserting shapes. More importantly, synthesis and manipulation aspects go hand-in-hand for our approach. A human user can *interpret* and *influence* any stage of synthesis, and can quickly generate a different output by varying a shape. Manipulation naturally emerges in our non-parametric approach without any additional efforts.

6.4 Quantitative Analysis

Dataset: We use semantic and instance label mask from COCO [253] to study the problem of in-the-wild image synthesis and manipulation. This dataset consists of 134 different objects and stuff categories, making it one of the most diverse and varied publicly available datasets. There are $118,287$ images in the training set ($40\times$ more than Cityscapes [80]), and $5,000$ images in the validation set ($100\times$ more than Cityscapes). We use the paired data of labels and images from the training set to extract global shapes and synthesize parts and pixels. The images are synthesized using semantic and instance label masks in the validation set. Our approach does not require any training, and therefore can use the labels and image component from anywhere. For the sake of fair comparison with parametric approaches, we restrict ourselves to COCO training data.

Baselines: To the best of our knowledge, there does not exist a non-parametric approach that has attempted the problem of the in-the-wild image synthesis from label maps on this large scale. We, therefore, compare our approach with parametric approaches: (1). Pix2Pix [197]; and (2). Pix2Pix-HD [450], using their publicly available codes. The complexity, diversity, and size of this dataset makes it a com-

putational challenge for a generative parametric approach to deal with. Training a vanilla Pix2Pix model took 20 days on a single Nvidia Titan-X GPU. With the same computation, we trained a Pix2Pix-HD model for a month but did not observe any convergence. It may be possible that a reasonable Pix2Pix-HD model be trained if we let the training go longer for an extra month or two, or use advanced computational resources. It may also be due to the design of architecture and hyper-parameters specifically suited for Cityscapes, and that efforts are required to tune hyper-parameters to make it work for a large and diverse dataset as COCO. It is, for this reason, we also use Cityscapes to contrast our approach with prior works [68, 197, 339, 450] for the sake of fair comparison. Additionally, we resize our generated outputs to 256×256 just to make a fair comparison with Pix2Pix on COCO. However, we can generate outputs having the same resolution as that of input label masks without any increase in compute.

FID Scores: We compute FID scores [179] using the images generated from different approaches. Lower FID values suggest more realism. Table 6.1 contrast FID scores computed on generated images (COCO) with Pix2Pix and Pix2Pix-HD (resized to 256×256 and 64×64 resolution). Without using any oracle, the top-1 example generated from our approach significantly outperforms the prior work. Additionally, note the performance improvement due to each stage in our hierarchical composition.

Method	#examples	Oracle	FID score (256×256)	FID score (64×64)
Pix2Pix [197]	1	✗	70.43	41.45
Pix2Pix-HD [450]	1	✗	157.13	109.49
Ours (shapes)	1	✗	37.26	23.22
Ours (shapes+parts)	1	✗	32.62	18.02
Ours (shapes+parts+pixels)	1	✗	**31.63**	**16.61**

Table 6.1: **FID Scores on COCO:** We compute FID score [179] to contrast the realism in outputs produced by different approaches. **Lower FID values suggest more realism**. We observe that our approach outperforms prior approaches significantly. We also demonstrate how different stages in our hierarchical composition lead to better outputs.

Mask-RCNN Scores: We use a pre-trained Mask-RCNN [169] to study the quality of synthesis on COCO [253] for Pix2Pix and our approach. This model is trained for 80 object categories of the COCO dataset. While it is trained for instance-segmentation, we use its output and convert it to semantic labels for consistency in evaluation. Our goal is to observe if we can get the same class labels from the synthesized images as one would expect from a real image. We, therefore, run it on original images from the validation set and use these pseudo semantic labels as

ground truth for evaluation. Next, we run it on synthesized images and contrast them with the labels from the original image. To measure the performance, we use three criteria: (1) mean pixel accuracy (PC); (2) mean class accuracy (AC); (3) mean intersection over union (IoU). Higher the score for each of the criteria, better is the quality of synthesis. Table 6.2 contrasts the performance of our approach with Pix2Pix and demonstrates substantially better results. Our performance improves when an oracle is used to select the best from five outputs. Note that top-100 exemplar matches are used for instance-shape matching in this experiment.

Method	#examples	Oracle	PC	AC	IoU
Parametric					
Pix2Pix [197]	1	✗	17.9	8.9	4.9
Non-Parametric					
Ours	1	✗	44.5	31.0	20.9
Ours	5	✓	58.2	41.2	31.4

Table 6.2: **Mask-RCNN Scores on COCO:** We use a pre-trained Mask-RCNN model [169] to study the quality of image synthesis. We run it on synthesized images and contrast them with the labels from the original image. To measure the performance, we use three criteria: (1) mean pixel accuracy (PC); (2) mean class accuracy (AC); (3) mean intersection over union (IoU). **Higher the score for each of the criteria, better is the quality of synthesis.** We outperform Pix2Pix. The performance further improves significantly when an oracle is used to select from five examples.

Human Studies: We did human studies on a randomly selected 500 images. We show the outputs of Pix2Pix, Pix2Pix-HD, and our approach (randomly picked one output from multiple) to human subjects for as much time as they need to make a decision. We asked them to choose one that looks close to a real image. The users were advised to use 'none of these' if all approaches are consistently wrong. 51.2% times user picked an output generated from our method, 7.8% times the outputs from Pix2Pix, and preferred 'none of these' 41% times. The human studies suggest that while our approach is most likable, there are still many situations where our method produced undesirable outputs.

Cityscapes: Table 6.3 contrasts the performance of our approach with prior approaches [68, 197, 339, 450] that have specifically demonstrated on Cityscapes. Except for Pix2Pix, we used publicly available results for this evaluation. Our approach is competitive to prior parametric and semi-parametric approaches with just 25 exemplar matches to extract shapes and parts to compose a new image from semantic labels. The performance improves when using an oracle to select best amongst the 5 generated outputs. Our performance may further improve as we increase the number of global images to do shape and part extraction.

Method	#examples	Oracle	PC	AC	IoU
Parametric					
Pix2Pix [197]	1	✗	72.5	29.5	24.6
CRN [68]	1	✗	49.0	22.5	18.2
Pix2Pix-HD [450]	1	✗	79.0	43.3	37.8
Semi-Parametric					
SIMS [339]	1	✗	68.6	35.1	28.1
Non-Parametric					
Ours (top-25)	1	✗	67.1	38.0	30.5
Ours (top-25)	5	✓	71.3	39.6	32.4

Table 6.3: **PSP-Net Scores on Cityscapes:** We use a pre-trained PSP-Net model [497] to evaluate the quality of synthesized images (1024×2048). This model is trained for semantic segmentation on cityscapes. We run the synthesized images through this model and generate a semantic label map for each image. The semantic label map from the synthesized images is contrasted with the semantic label map from the original image. We compute three statistics for each approach: (1) Mean Pixel Accuracy (PC); (2) Mean Class Accuracy (AC); (3) Mean intersection over union (IoU). For each of these criteria- **higher the score, better is the quality of synthesis**. With just 25 exemplar matches to extract shapes and parts, our non-parametric approach is competitive to the parametric models and semiparametric models.

6.5 OpenShapes: User-Controls

OpenShapes is a web interface (Figure 6.10) that allows a human-user to conceive their thoughts by creating a coarse label map on the canvas (plain white sheet on the left side in Figure 6.10). There are 150 different object-background categories that a user can select from (right side in Figure 6.10). We use paint-brush to fill background categories. The paint-brush is of varying size (*Brush Size* in Figure 6.10). The user selects the paint brush and a category of interest. Figure 6.11 shows a user filling the canvas with *sea* category. Once the canvass is filled, the user calls **Create** on the top-left as shown in Figure 6.12. The system generates 5 different outputs for the input created by the user as shown in Figure 6.13. The user can vary inputs by changing different background categories as shown in Figure 6.14, Figure 6.15, and Figure 6.16. We provide a drag-and-drop tool that enables a user to add objects on the canvas. There are multiple shapes for each object provided in the system (shown in the right side of Figure 6.17). The user selects the desired shape that appears on the left-side of the canvas, and can appropriately position it as shown in Figure 6.18. After calling create, the system generates different outputs. More instances of same object or more objects can be added on the canvas (Figure 6.19). Objects can be resized and appropriately positioned as desired by the user as shown

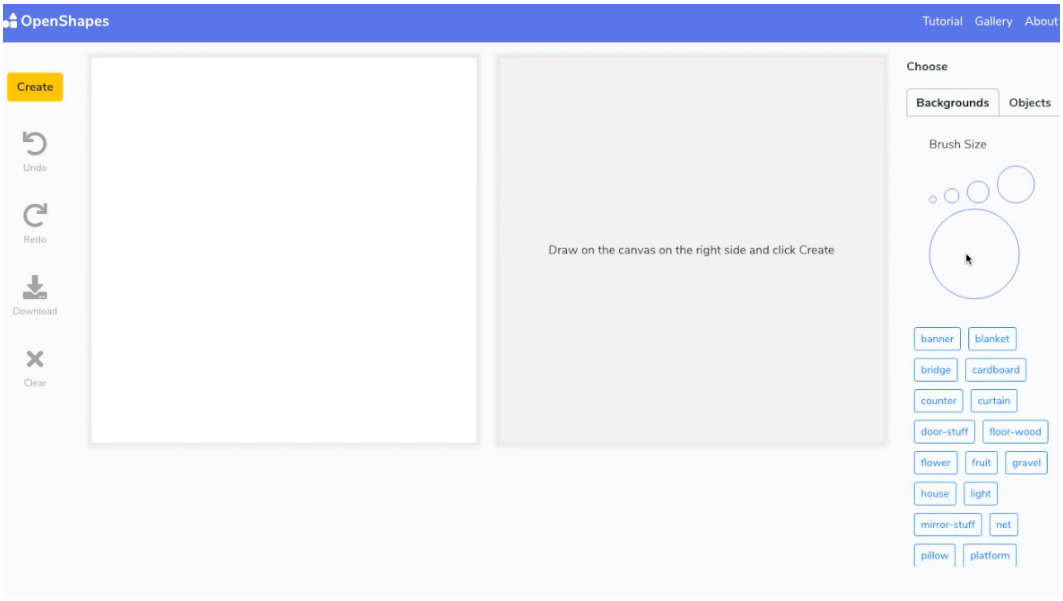


Figure 6.10: **OpenShapes Interface:** OpenShapes is a web interface that allows a human-user to conceive their thoughts by creating a coarse label map on the canvas (plain white sheet on the left side). There are 150 different object-background categories that a user can select from (right side). We use paint-brush to fill background categories. The paint-brush is of varying size (*Brush Size*.

in Figure 6.20. Figure 6.21 shows multiple outputs generated by the system for the user-input. We have other functionalities such as *undo*, *redo*, *download*, and *clear* on the left side. The different examples shown here are for the desktop-version of the application. We also have a user-interface and a version for mobile phones that allows a user to fill the canvas using fingers (Figure 6.22). OpenShapes enables us to synthesize images on everyday computational resources without requiring extensive computational power. It runs on a single-core CPU and generates 5 outputs of 512×512 resolution for a given input in 10 seconds. A user can also add any number of new categories to the system without much manual efforts and without increase in computational resources.

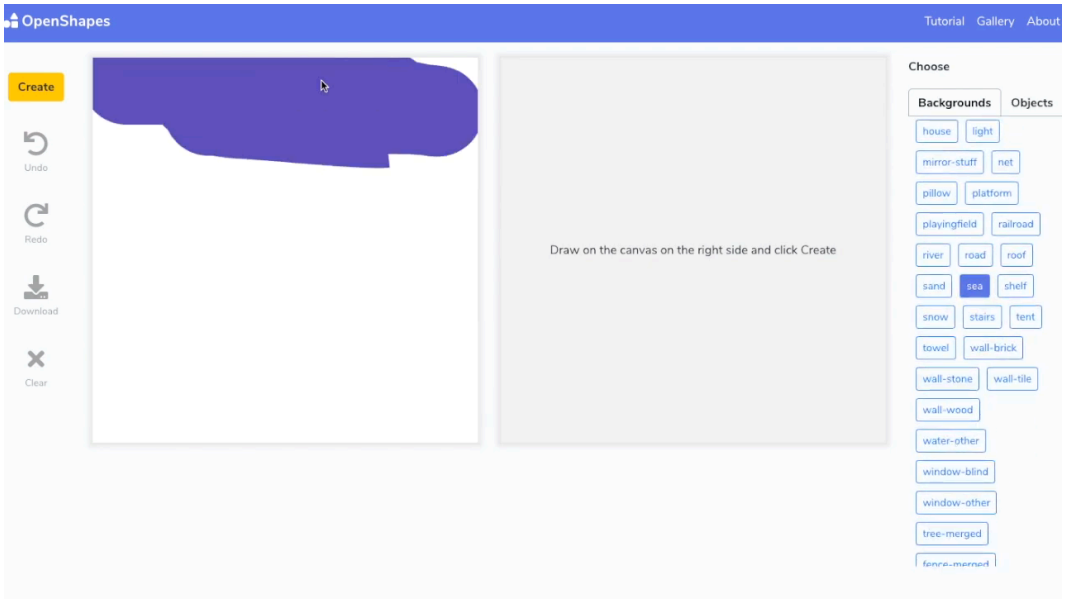


Figure 6.11: **Paint Brush:** The user selects a *background* category (sea in this example) and starts filling the canvas (left-side).

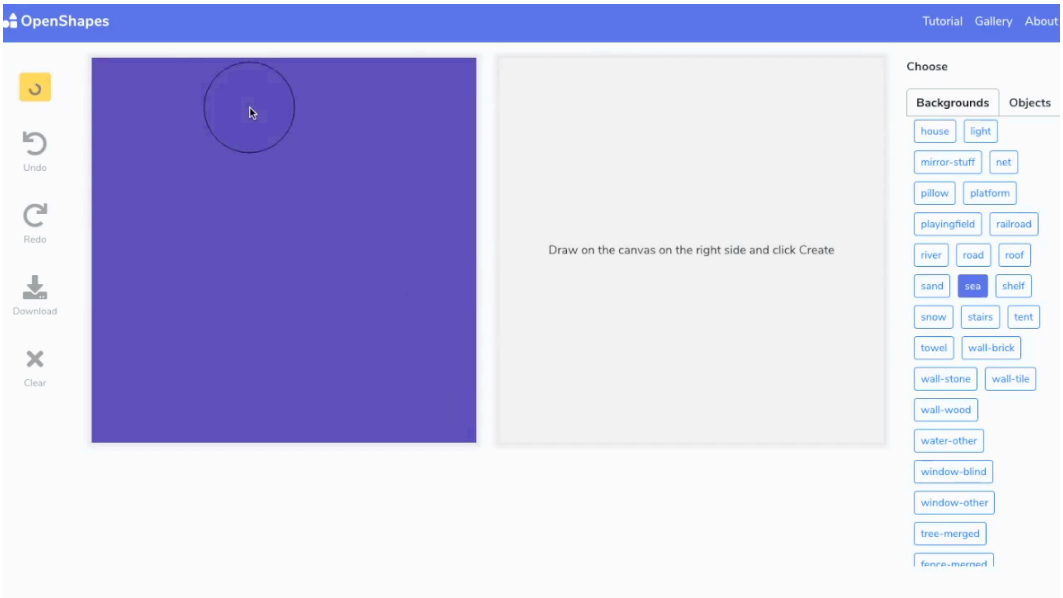


Figure 6.12: **Create:** Once the canvas is *filled*, a user calls *create* on the top-left side of interface.

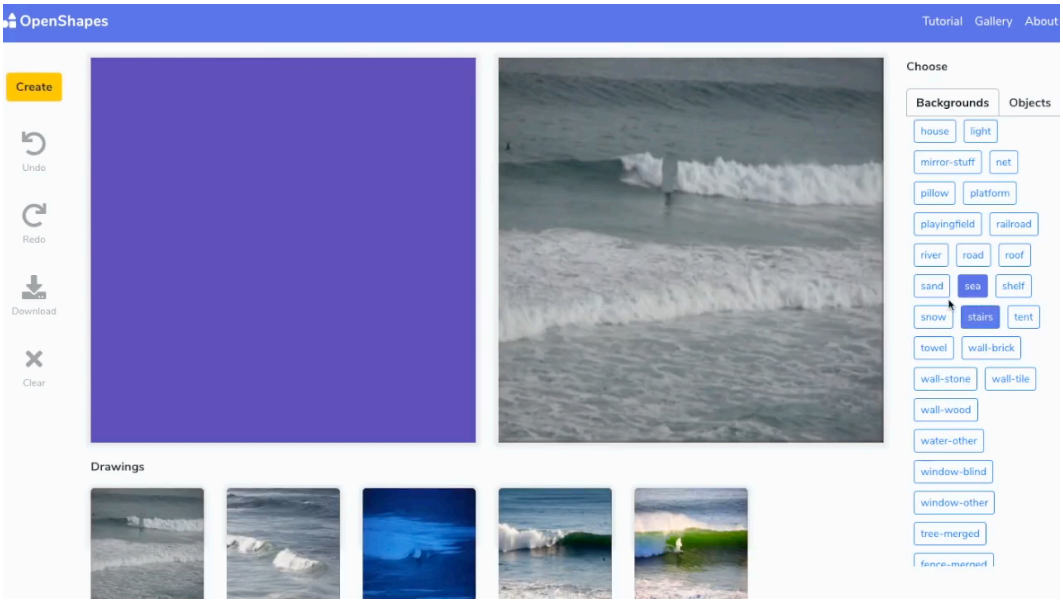


Figure 6.13: **Multiple Outputs:** OpenShapes generate 5 outputs for the given input (shown below). The right side shows the best scoring output.

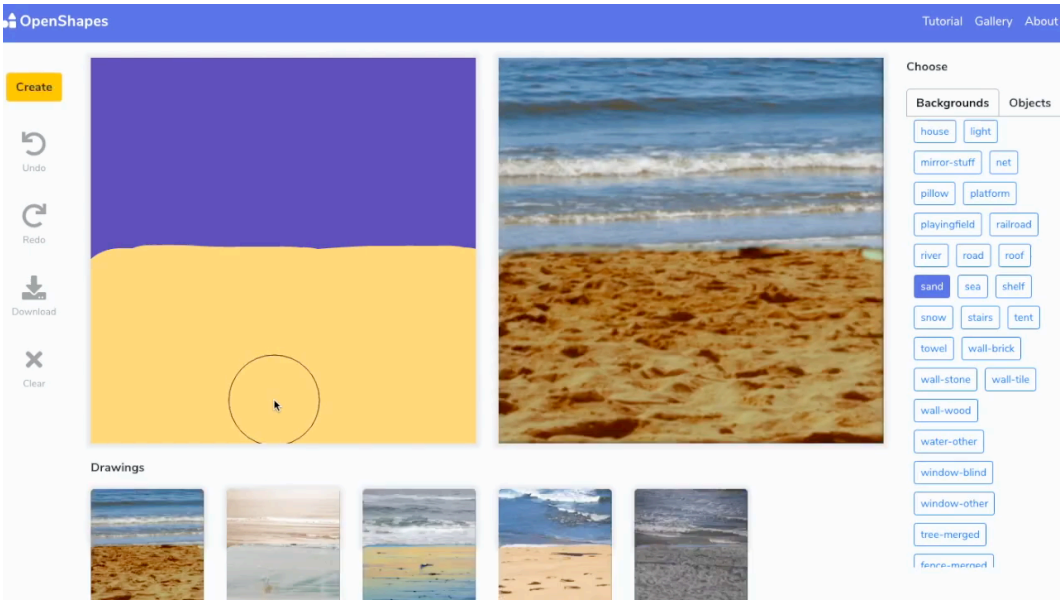


Figure 6.14: **Varying Inputs:** The user can now vary input by using other background categories. E.g., a user adds *sand* in the canvas and call **create** to generate 5 outputs.

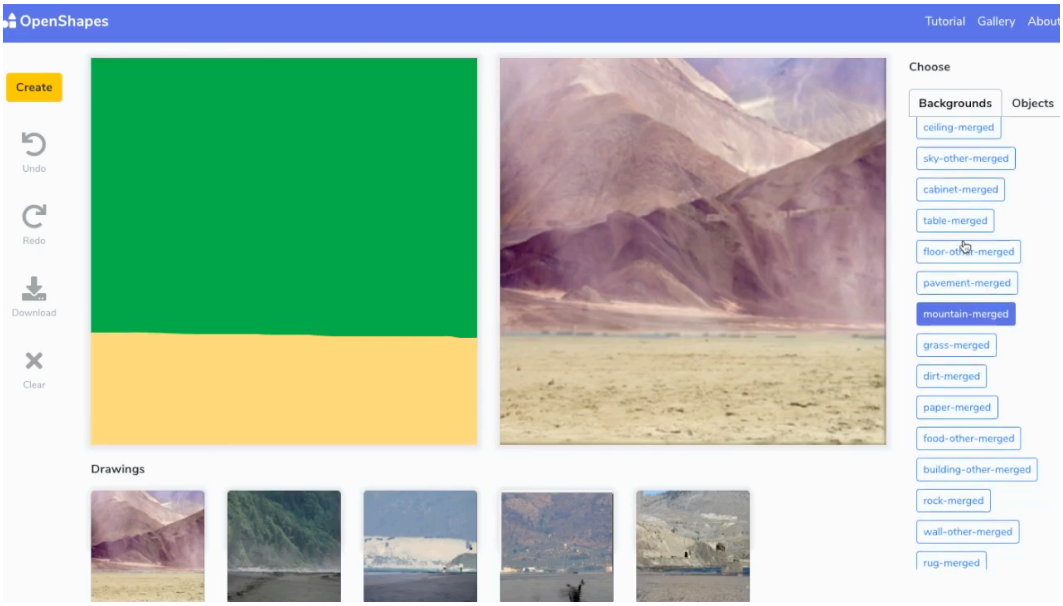


Figure 6.15: **Varying Inputs:** The user changed the *sea* with *mountains*, and call **create** to generate 5 outputs.

Figure 6.16: **Varying Inputs:** The user creates a completely new input by adding *trees* and *grasses*. After pressing **create**, OpenShapes generates 5 outputs.

Figure 6.17: **Drag-and-Drop Tool:** Objects such as dogs, cats etc can be added to the canvas using drag-and-drop tool. OpenShapes provides multiple shapes of objects (shown on the right). The user selects the desired shape that appears on the canvas on the top-left.

Figure 6.18: **Positioning Object and Create:** The user positions object at the desired location and call **create**. The system generates 5 different outputs.

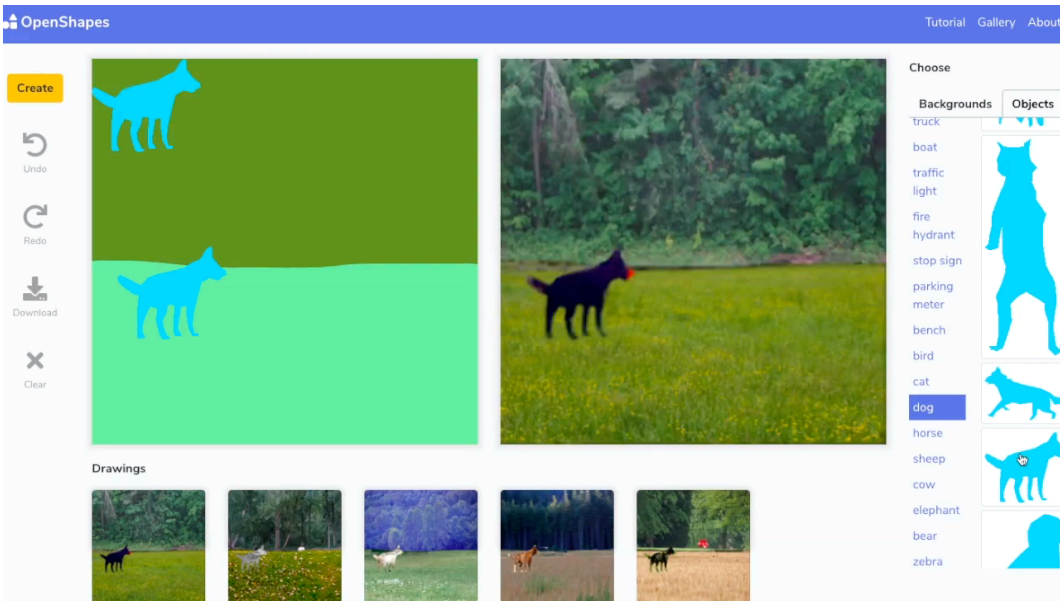


Figure 6.19: **More Instances:** The user can add more instance of the same object and can also add more objects. Shown here is adding the same dog in the scene.

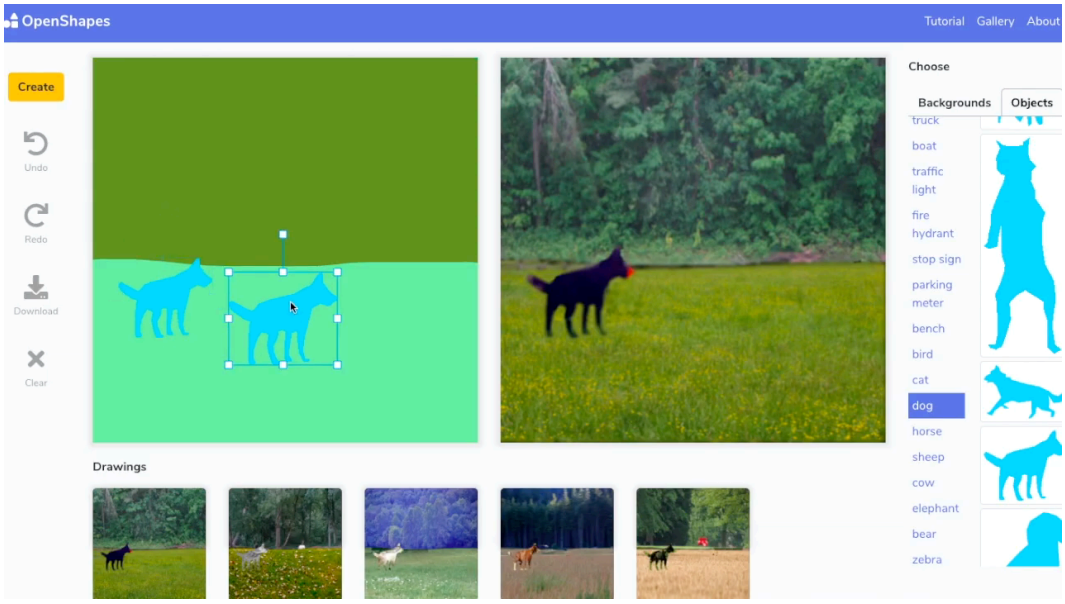


Figure 6.20: **Varying Size of the Object:** The user can resize shape of the object on the canvas.

Figure 6.21: **Output:** After calling **create**, the system generates 5 outputs for the given input.

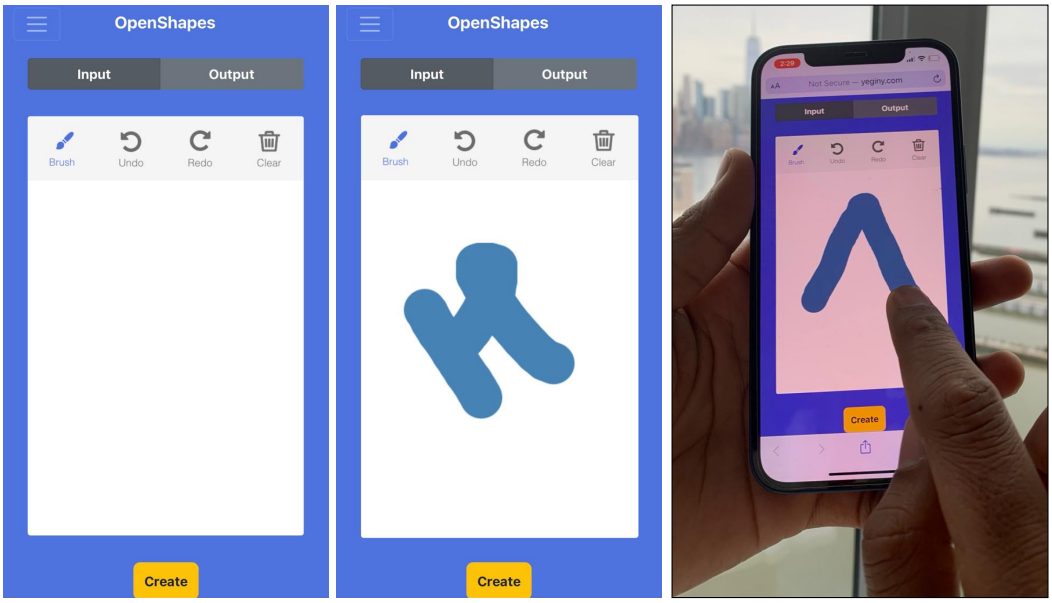


Figure 6.22: **User-Interface for a touch-enabled mobile phone:** We adapted the user-interface to account for the screen space and enable a touch-based interface. As shown here, a user can create the desired input using a finger on their phone.

Chapter 7

Human-Controllable Video Exploration

We present simple video-specific autoencoders that enables human-controllable video exploration. This includes a wide variety of analytic tasks such as (but not limited to) spatial and temporal super-resolution, object removal, video textures, average video exploration, and correspondence estimation within and across videos. Prior work has independently looked at each of these problems and proposed different formulations. In this work, we observe that a simple autoencoder trained (from scratch) on multiple frames of a specific video enables one to perform a large variety of video processing and editing tasks. Our tasks are enabled by two key observations. (1) Latent codes learned by the autoencoder capture spatial and temporal properties of that video; and (2) Autoencoders can project out-of-sample inputs onto the video-specific manifold. For e.g. (1) interpolating latent codes enables temporal super-resolution and user-controllable video textures; (2) manifold reprojection enables spatial super-resolution, object removal, and denoising without training for any of the tasks. Finally, a two dimensional visualization of latent codes via principal component analysis acts as a tool for users to both visualize and intuitively control video edits.

7.1 Video Analytics

In this chapter, we demonstrate that simple video-specific autoencoders learn meaningful representations that enable a multitude of video processing tasks, without being optimized for any specific task. Figure 7.1 shows an example of a video-specific autoencoder, where a simple autoencoder is trained on 1024×1024 resolution frames from an Obama video (300 *un*ordered frames). We use the autoencoder for both temporal and spatial super-resolution in this video, without even optimizing for either task. In this work, we illustrate the effectiveness of video-specific au-

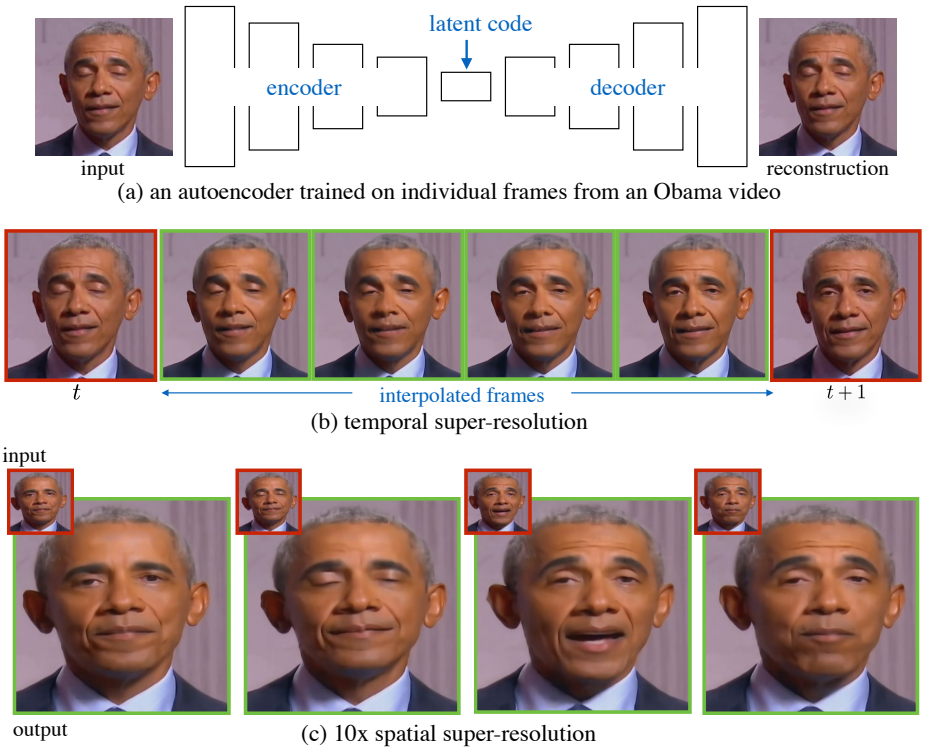
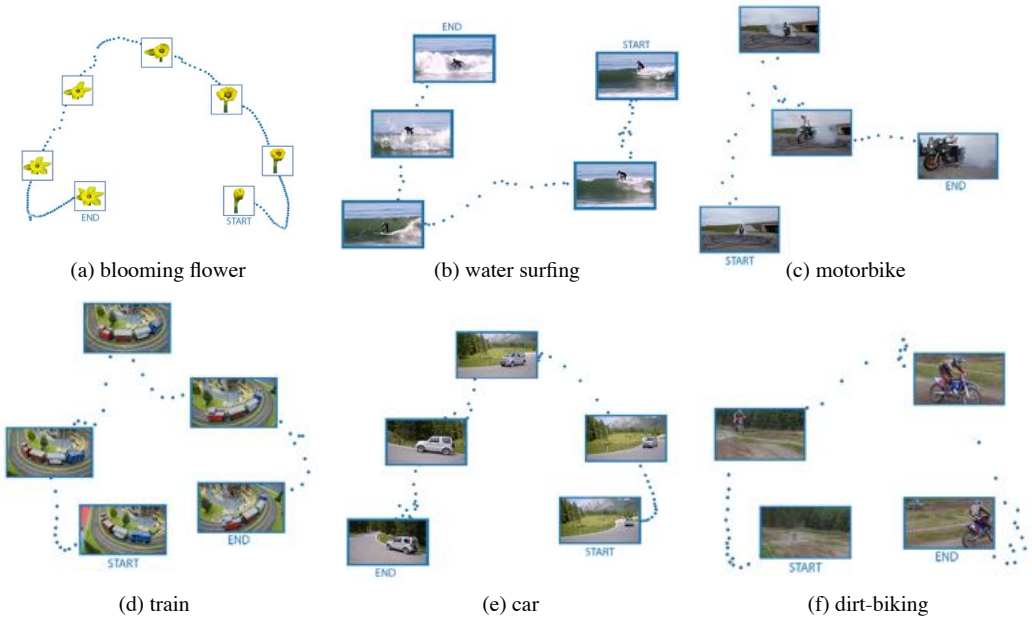


Figure 7.1: **(a) Video-Specific Autoencoders:** We demonstrate a remarkable number of video processing tasks enabled by an exceedingly simple video-specific representation; an image autoencoder trained on frames from a target video (here, 300 unordered 1024×1024 frames). By interpolating the latent codes of adjacent frames (and decoding them), one can perform **(b) temporal super-resolution**. By linearly upsampling low-res 96×96 image frames to 1024×1024 blurry inputs and passing them through the autoencoder, we can "project" such noisy inputs into the high-res-video-specific manifold, resulting in high quality 10X **(c) super-resolution**, even on subsequent video frames not used for training.

toencoders for a wide assortment of video processing tasks, including frame-level correspondence within and across videos, video retargeting, average video exploration, video textures, and object removal. Importantly, we observe that latent codes learned by video-specific autoencoders can provide intuitive controls to a human operator for interactive video exploration and editing.

Task-Specific vs. Video-Specific Modeling: Decades of research on video processing [412] has focused on task-specific models and representations, i.e., if one's goal is to see a video in slow-motion, design models or train representations tuned for temporal super-resolution [202,275]. The same model cannot be used for spatial super-resolution [456], creating infinitely-looping video textures [370], or interac-

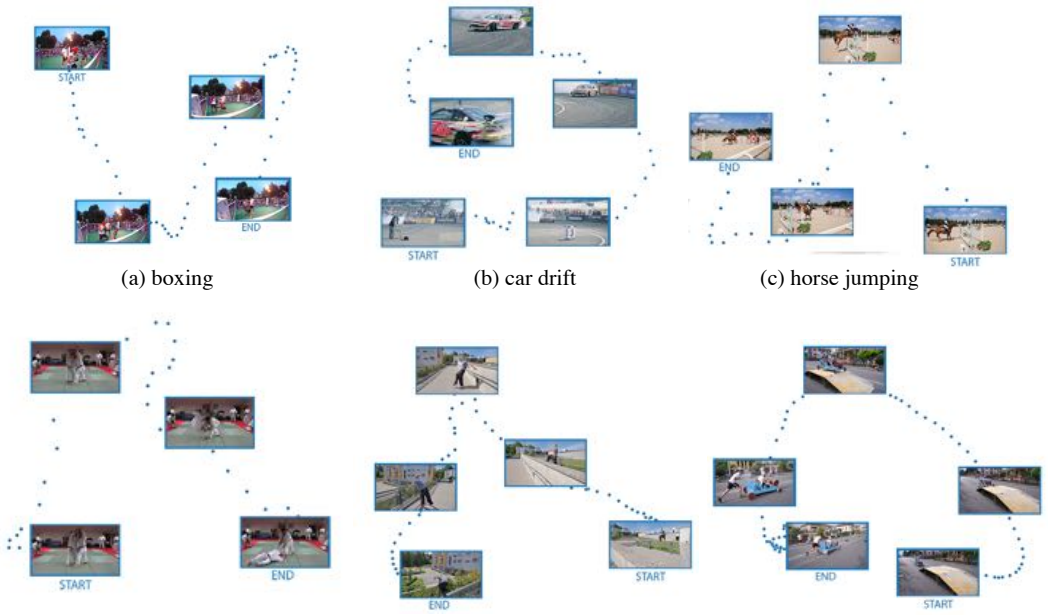


(a) blooming flower (b) water surfing (c) motorbike

(d) train (e) car (f) dirt-biking

Figure 7.2: **Continuous representation without temporal information:** We visualize the latent code learned by a video-specific autoencoder via multidimensional scaling with PCA. We find that video-specific autoencoders learn a temporally **continuous representation** without any explicit temporal information. We show this property for various natural videos such as a blooming flower, a water surfing event, a baseball game, and bowling. This continuous latent representation allows one to slow-down or speed-up a video (through latent code resampling) and enables interactive video exploration.

tive visualization of modes in a video [511]. Our approach dramatically differs from on-going research on video processing because we do not learn task-specific representations. Instead, we learn a *video-specific* representation designed to capture the continuous manifold and latent structure of visual information contained in that video. To do so, we use a remarkably simple and lightweight approach: simply train an *image-based* autoencoder on multiple frames from that video. Our approach is unique in that we require no large-scale training video corpora, which can be notoriously difficult to process and learn from. *Task-agnostic test-time training* on a specific video is beneficial for three reasons: (1) it naturally *adapts* to the spatial and temporal properties of the video; our pipeline works regardless of the video resolution, frame rate, color space, etc.; (2) it does not inherit the limitations of optimization and training data; e.g., models trained for single-frame interpolation [262, 265, 311] are non-trivial to extend to multi-frame interpolation (notable exceptions [202] exist); and (3) it allows human-users to free-form explore and edit



(a) boxing (b) car drift (c) horse jumping

(d) judo (e) parkour (f) soapbox

Figure 7.3: **Continuous representation without temporal information:** We show more examples here for various sporting events such as (a) boxing, (b) car drift, (c) horse jumping, (d) judo, (e) parkour, and (f) soapbox. We observe a continuous representation even though we did not use temporal ordering or explicit temporal information.

videos without regard to a pre-defined application.

Video-Specific Autoencoder: An autoencoder trained using individual frames (without any temporal information) of a specific video via a simple reconstruction loss learns both spatial and temporal aspects of the video (Figure 7.2, Figure 7.3, Figure 7.4, and Figure 7.5). Simple operations on its latent code, encoder, and decoder enables a wide variety of tasks. For e.g. (1) reconstructing the interpolated latent code of the autoencoder enables temporal super-resolution. We can trivially create a new video with any desired frame-rate (Figure 7.1-(b)); (2) passing out-of-sample input images through the autoencoder projects them into the video manifold, enabling spatial super-resolution, object removal, and denoising (Figure 7.1-(b)); (3) a two dimensional visualization of latent codes via principal component analysis (PCA) acts as a tool for a user to quickly explore the contents of a video, and do average video exploration (like average image exploration [511]); (4) a simple nearest neighbor on latent codes combined with the decoder enable us to create arbitrary-length looping videos (also known as video textures); (5) a simple nearest neighbor on latent codes allows one to establish correspondences across frames (6) pixel-level encoder features can be used to enable dense pixel correspondences within a video.

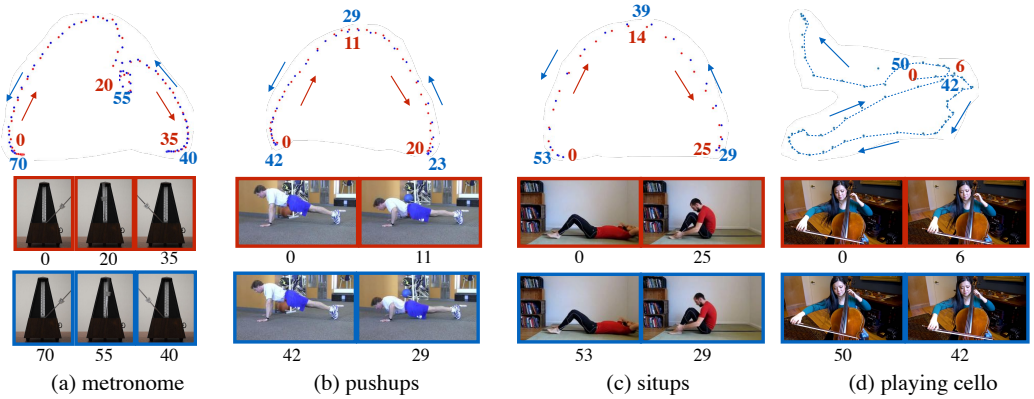
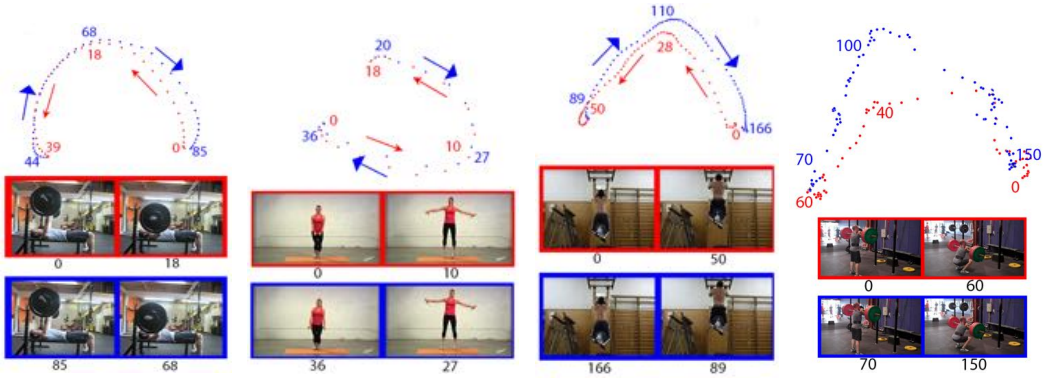


Figure 7.4: **Associating Frames within a Repetitive Video:** We observe that latent spaces are able to learn the repetitive motion without using any temporal information. We show examples of (a) metronome; (b) pushups; and (c) situps. Latent spaces are also tuned to capture video-specific subtleties. (d) We show the example of woman playing cello and observe that the "stretched-out" hand posture in down-bow and up-bow (which are nearby in latent space even when far apart temporally).

We present a number of diverse tasks enabled through simple operations applied to our exceeding simple representation. To the best of our knowledge, we are the first to explore such diverse video processing tasks using a single representation not optimized for any specific task.

Learning from a Single Image: Our approach is closely inspired by extensive research on single image learning [144,292,381,387,388] that aims to learn structure within a single image. These insights have also been used in our work on Exemplar Autoencoders where we learn the characteristic natural voice of a person [87]. Similar to our work on video-specific autoencoders, these approaches [87,381,387] also allow a wide variety of operations on images/audio without optimizing for a specific task. Learning from a single image [381,387], however, is challenging and requires careful encoding of multi-scales, data augmentation strategies, and (adversarial) loss functions. We find learning on video frames to be far simpler, because multiple frames naturally span multiple scales and "augmentations", acting a natural form of regularization. Indeed, we find that simple autoencoder architectures and reconstruction losses suffice.

Contributions: (1) We introduce a simple unsupervised approach for learning video-specific exemplar representations without needing large training data. This representation enables us to do a wide variety of the aforementioned video processing tasks that generally require a dedicated approach. (2) Our approach allows



(a) bench-press (b) jumping-jacks (c) pullups (d) weights

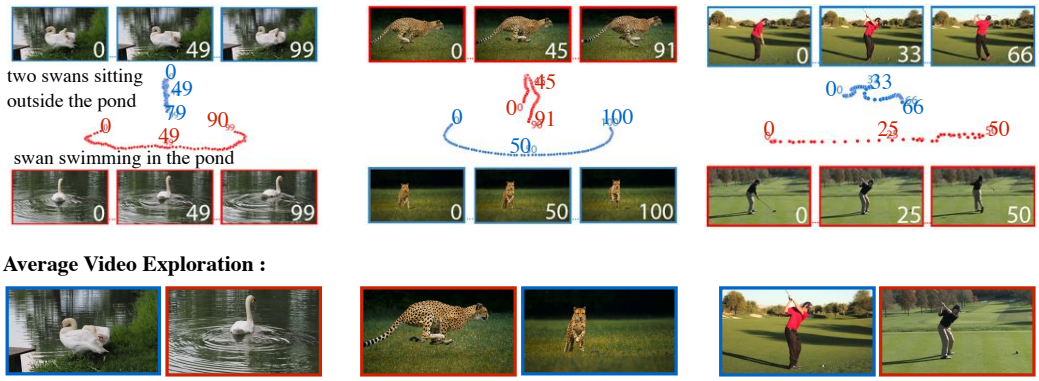
Figure 7.5: **Associating Frames within a Repetitive Video:** We show more examples of human activities: (a) bench-press; (b) jumping-jacks; (c) pullups; and (d) weights. We observe that latent spaces are able to learn the repetitive motion without using any temporal information. Latent spaces are tuned to capture video-specific subtleties and cluster nearby in latent space even when far apart temporally.

for intuitive user interaction, via a low dimensional visualization of latent codes that allow for video exploration and editing; and (3) finally, we extended frame-level (latent-code) representations to per-pixel representations (via "hypercolumn" features extracted from the encoder [162]), enabling dense pixel-level video correspondence.

7.2 Videos: Opportunities and Conundrums

400 hours of video data is uploaded to YouTube every minute. The rich video data opens up enormous opportunities for exploring the vast visual content. However, it also opens up a large number of conundrums because videos contain both spatial and temporal information.

Video Processing: One way to solve this conundrum is to consider every application as an individual problem [412]. There is a large body of work on video processing tasks such as video completion [133], video enhancement [480], video inpainting [61, 185, 479], video editing [44, 308], temporal super-resolution [202, 275, 516], spatial super-resolution [163, 473], space-time super-resolution [382, 383], removing obstructions [260], varying speed of a video [34] or the humans in it [270], video textures [4, 258, 370], finding unintentional events in a video [106], generative modeling for associating two videos [22] discriminative modeling for association [97, 338, 455], associating multi-view videos [26, 442], or pixel-level correspondences in a video [17, 483]. In this work, we take a rather different approach and explore a representation learned for a specific-video that enables all of these tasks.

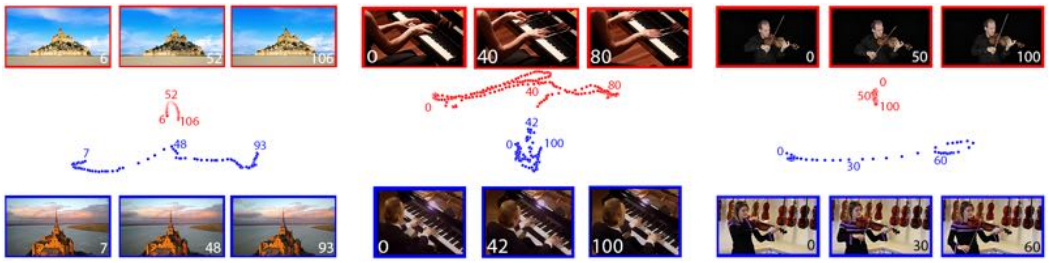


(a) outside pond vs in the pond (b) cheetah in two poses (c) two golf games

Figure 7.6: **Distinct Modes & Average Video Exploration**: Distinct modes emerge when the autoencoder encounters different visual concepts in a video. (a) and (b) show two modes of swan and cheetah respectively from a video. We also show distinct modes learned by a single autoencoder trained on two videos as shown in (c). Here, the autoencoder learns latent spaces able to model both videos, which tend to appear as separate modes. Beneath each example, we show "average image" for each mode in a video.

Learning from a Single Instance: Often there exists repetitive structure in a signal, such as patch-recurrence in an image [15], that could enable us to learn a meaningful representation for that signal without any additional information. There is plethora of work that has explored representation learned using a single image for various tasks [15,144,292,388,435,465]. Recent approaches [381,387] have also explored image-specific representation that enables a wide variety of image editing tasks. We extend these observations to videos. Our goal is to learn a representation for a video without any additional information. Because we have more informative data, we could learn a a simple autoencoder that optimizes the reconstruction of individual video frames. Prior work on video processing [22,133,202] often encodes spatial-temporal information explicitly. In this work, we considered frames from a video as independent images. Despite this, our video-specific autoencoder learns continuous representation as shown in Figure 7.2-7.5.

Human-Controllable Representation: Usually the front-end of an application is designed around the task of interest. For e.g., prior work [16,25,129,511] on user-control are limited to a task. In this work, we hope to provide users with a simple representation that they can easily work with to design new application without much overhead such as simple algebraic operations on the latent codes. This will further reduce the requirement of application-dependent modules. For e.g., work on object removal [133] requires a user to provide extensive object-level mask across the video or use an off-the-shelf segmentation module trained for particular object.



(a) two castles (b) two piano performances (c) two violin performances

Figure 7.7: **Two Videos Separate Out**: Distinct modes emerge when an autoencoder is trained for two videos of a similar concept: (a) two castles; (b) two piano performances; and (c) two violin performances.

In this work, we show that a user can do these operations with minimal work. Figure 7.6 and Figure 7.7 show how a user can explore the contents of a video and explore the average for each mode. This can be further extended to video retargeting shown in Figure 7.8 by playing with latent codes.

7.3 Video-Specific Autoencoders

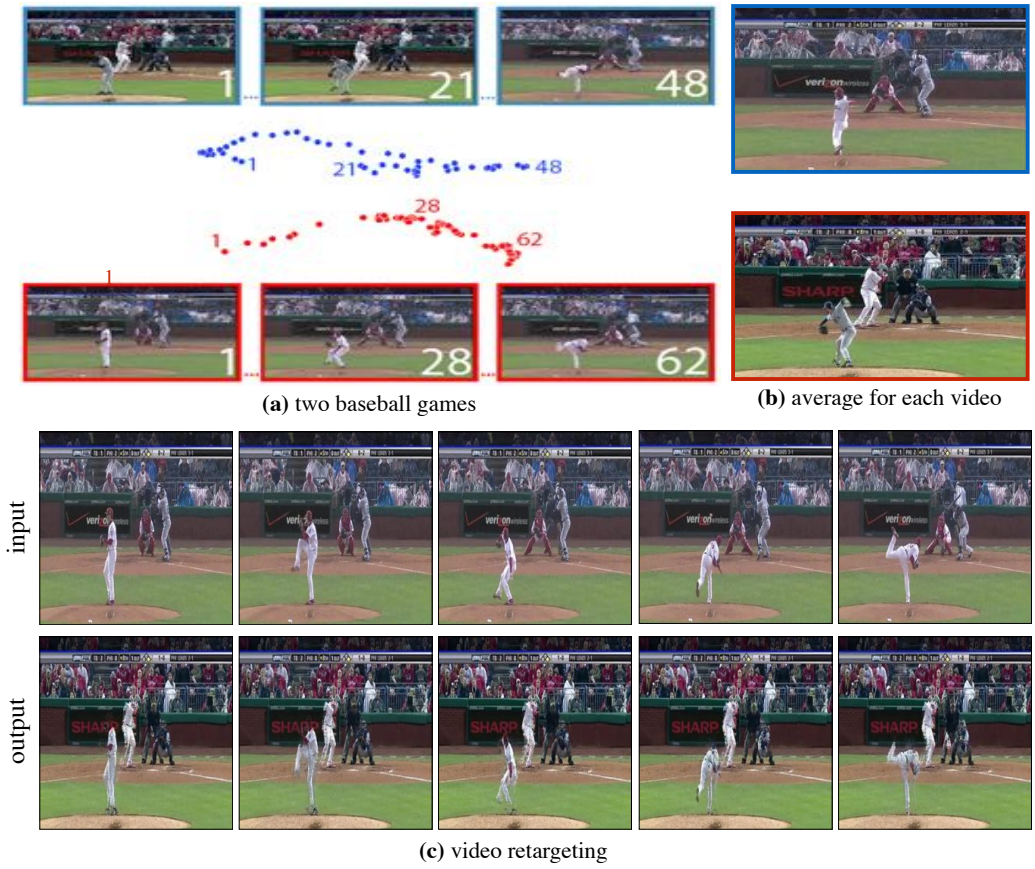
An autoencoder [146] compress the information in a signal via an encoding function. The compressed signal or latent codes are represented using a few bits of information. Given a set of frames from a video $x \in V$, video-specific autoencoders learn to encode each frame into a low-dimensional latent code $f(x)$ that can be decoded (via a function g) back into the high-dimensional input space, so as to minimize the reconstruction error:

$$\min_{f,g} \sum_{x \in V} ||x - g(f(x))||^2 \tag{7.1}$$

The ability to properly compress and reconstruct a video opens up a vast opportunity of things that we can do with it. However, there are two necessary conditions for a video: (1) learning a continuous temporal space such that similar concepts in time clusters well; and (2) learning a continuous spatial space such that similar concepts in 2D image space clusters well. We, first, describe the model used in our analysis in Section 7.3.1. We then study if an autoencoder trained on a single video can capture these properties in Section 7.3.2.

7.3.1 Model

We use a convolutional feed-forward model that inputs an image and reconstructs it. Importantly, we ensure all operations are convolutional, implying that the size of



(a) two baseball games **(b)** average for each video

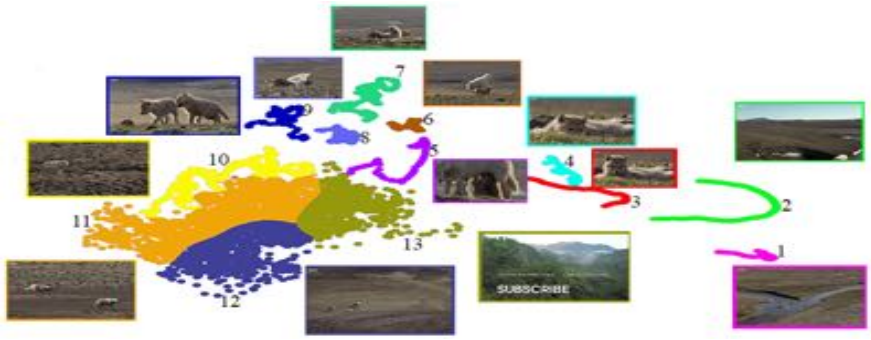
(c) video retargeting

Figure 7.8: **Video Retargeting:** We extend association across videos to do video retargeting. We use the modified latent code from video-1 (**input**) and reconstruct the new frames via decoder. The resulting frames look like video-2 (**output**) but captures the motion of video-1.

the latent code scales the resolution of the video V. To ensure that the latent code contains all the information needed to reconstruct a video frame, we do not use skip connections.

Convolutional Encoder(f): The encoder consists of six 2D convolutional layers. We use 5×5 kernels for first four layers and a stride of 2 that downsamples the input by 0.5 after each convolution. The last two layers have 5×5 kernels without any downsampling. The output of each of these layers is max-pooled in a 2×2 region with a stride of 2. Each conv-layer is followed by batch-normalization [194] and a ReLU activation function [238]. The output of last layer of the encoder is used as a latent representation (or also termed as latent code) in this work.

Convolutional Decoder(g): The decoder inputs the latent code to reconstruct the



(a) a documentary on wolves

(b) a documentary on beetles

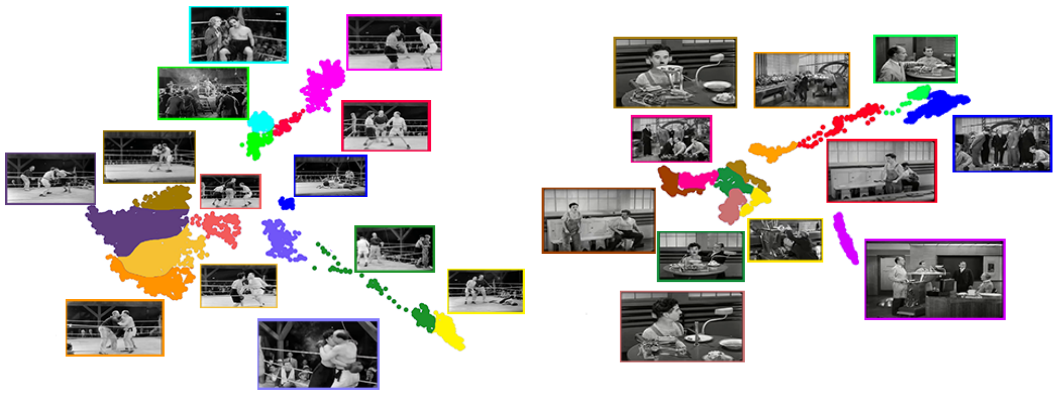
(c) an orchestra performance

Figure 7.9: **Different Concepts in a Long Video Separate Out**: Distinct modes emerge when an autoencoder is trained for long video such as: (a) a documentary on wolves; (b) a documentary on beetles; and (c) an orchestra performance.

output. It consists of eight up-sampling conv-layers with a 4×4 kernels and a stride of 2 that upsamples the input by 2. Each conv-layer is followed by batch-normalization and a ReLu activation function.

Latent Codes ($f(x)$): Given an input image x with shape h×w, the encoder outputs an encoding with the shape $(k * 12) \times \frac{h}{64} \times \frac{w}{64}$, where k is the number of filters in the first layer of encoder. Because of the high dimensionality, we want to find a way to visualize these encodings to potentially see the relations between frames. Using PCA, we can reduce the encoding to a 2-dimensional format to visualize on a 2D graph. For most experiments, we fix $k = 32$, i.e. our autoencoders compress each frame by $32\times$.

Pixel Codes ($f_i(x)$): We use our encoder to extract pixel-level representations following the "hypercolumn" approach [162]. Conceptually, one can resize each of the six convolutional layers of the encoder f back to the original image input size, and then extract out the "hyper"-column of features aligned with pixel i. In prac-



(a) boxing ring **(b)** eating machine

Figure 7.10: **Black and White Charlie Chaplin Videos**: Distinct modes emerge when an autoencoder is trained for old videos of Charlie Chaplin such as: (a) inside the boxing ring ; and (b) eating machine.

tice, one can extract features from the image-level encoder $f(x)$ without resizing through judicious bookkeeping. Our final pixel representation, written as $f_i(x)$ is 2176 dimensional.

Video-Specific Manifold (M): We define the *manifold* of an autoencoder to be the set of all possible output reconstructions obtainable with *any* input:

$$M = \{g(f(x)) : \forall x\} \tag{7.2}$$

where f, g are the "argmin" encoder and decoder learned from (7.1). In our setting, M corresponds to be a video-specific manifold of potential image frame reconstructions. It is well-known that feedforward autoencoders, when properly trained, act as projection operators that project out-of-sample inputs x into the manifold set M. We follow the definition used in [35, 146].

$$||g(f(x)) - x||^2 = \min_{m \in M} ||m - x||^2, \quad \forall x \tag{7.3}$$

$$= \min_{x'} ||g(f(x')) - x||^2, \quad \forall x \tag{7.4}$$

One can build intuition for above by appealing to linear autoencoders, which can be learned with PCA. In this case, the above equations point out the well-known fact that PCA projects out-of-sample inputs into the closet point in the linear subspace spanned by the training data [41]. We make use of this property to "denoise" noisy input images $x \notin V$ into manifold M, where noisy inputs can consist of upsampled / blurry frames.

(a) original (b) reconstructed

Figure 7.11: **Continuous Spatial Imaging:** The reprojection property of autoencoder enables us to learn patch-level statistics in a frame. We show examples of removing the dancer in each frame (red box in **(a)**). Despite discontinuities, the autoencoder generates a continuous spatial image **(b)**.

7.3.2 Properties

We now explore various properties of a video-specific autoencoder to study its applicability for video analytics.

Learning Continuous Temporal Space: We study the space of latent codes $f(x)$ to examine the impact of temporal variation of input frames x_t, since the autoencoder is learned without any explicit temporal input. We visualize the latent code space via multidimensional scaling with PCA [41]. Figure 7.2 and Figure 7.3 shows that a video-specific autoencoder learn a temporally continuous representation without any explicit temporal information for a wide variety of natural videos such as a blooming flower, a water-surfing event, a baseball game, and bowling. The autoencoder trained on the specific video implicitly learns the correlations in the various frames and a continuous temporal space emerges. This property allows us to slow-down or speed-up a video (through latent code resampling).

Repetitive Motion: It is crucial to verify if the property would hold if we provide videos that have repetitive motion. We use various natural videos such as a metronome, cyclic exercises (pushups, situps, squats etc), and playing instruments (cello, violin, piano etc). We observe that latent codes can capture the repetitive motion as shown in Figure 7.4 and Figure 7.5. The cycles emerge for each of these

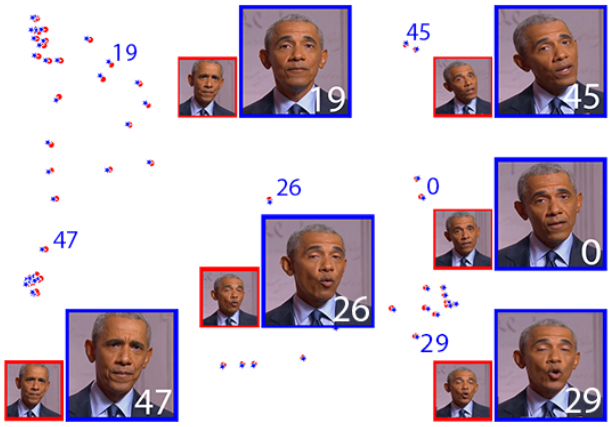


Figure 7.12: **Studying Reprojection Property via Spatial Super-Resolution:** We use the autoencoder trained on Obama examples (Figure 7.1) to study the reprojection property. We observe that similar low-res (96×96) and hi-res inputs (1024×1024) map to the same point in the latent space. This means that we can use low-res inputs once an autoencoder is trained on few hi-res examples and yet be able to get hi-res outputs.

videos. Specifically interesting is the example of a woman playing cello in Figure 7.4-(d). The subtle details such as similar hand position occur at same position though they are far-apart on the time axis of the video. This property enables us do frame-level correspondences and create arbitrary-length videos (also known as video textures).

Non-Continuous Videos: We now study non-continuous videos. Figure 7.6 and Figure 7.7 shows that the latent codes cluster in two distinctive modes when the autoencoder gets a non-continuous video. For e.g., Figure 7.6-(a) a swan swimming in a pond vs. sitting outside the pond with other swan; and 7.6-(b) cheetah in two completely different poses. This property allows us to explore the videos quickly by glancing the modes (or average videos) as shown in Figure 7.16. We further explored this property by explicitly providing frames from two or three videos in Figure 7.6-(c) showing two golf games. Figure 7.7 shows how two videos separate out for different examples. Finally, Figure 7.8-(a) shows distinct modes arising in the space of latent codes for two baseball games. However, each mode in the respective videos still learn a meaningful continuous representation. The latent codes of an autoencoder trained on two or more videos are useful to do frame-level association across the multi-views of an event and across different videos. The association across different videos can be further extended to video retargeting as shown in Figure 7.8-(c).

Long Videos: We extend the above study to long videos (10-15 minutes long)

downloaded from internet. Figure 7.9 consists of (a) a documentary on wolves, (b) a documentary on beetles, and (c) an orchestra performance. We observe different concepts in these videos separate out. Figure 7.10 shows two examples for black and white Charlie Chaplin videos. The quality and statistics of a video does not influence our approach since it is based on test-time training. We observe distinct modes for different sequences despite a large sequence. This property allows to quickly glance the contents of video and explore it easily.

Reprojection onto Manifold: The reprojection property enables the model to map inputs to examples previously-seen during training. This property enables us to do more than $10\times$ spatial super-resolution. Figure 7.1-(c) shows an example where we input a low-res image (96×96) to an autoencoder that was trained on 1024×1024 frames. We study the applicability of the reprojection property of the autoencoder via spatial super-resolution using this example in Figure 7.12. We observe that similar low-res (96×96) and hi-res inputs (1024×1024) map to the same point in the latent space. This means that we can input a 96×96 input and yet be able to get a 1024×1024 output. This property also enables transmission of fewer bits over the network and get hi-res outputs at the reception given the video-specific autoencoder.

Learning Continuous Spatial Imaging: The reprojection property of autoencoder also enables us to learn patch-level statistics in a frame. Shown in Figure 7.11 are the example where we remove the dancers in the frames. Despite discontinuities, the autoencoder generates a continuous spatial image. This property enables us to do object removal and spatial extrapolation in the videos.

7.4 Demonstration

We now demonstrate a few applications using the properties of a video-specific autoencoder.

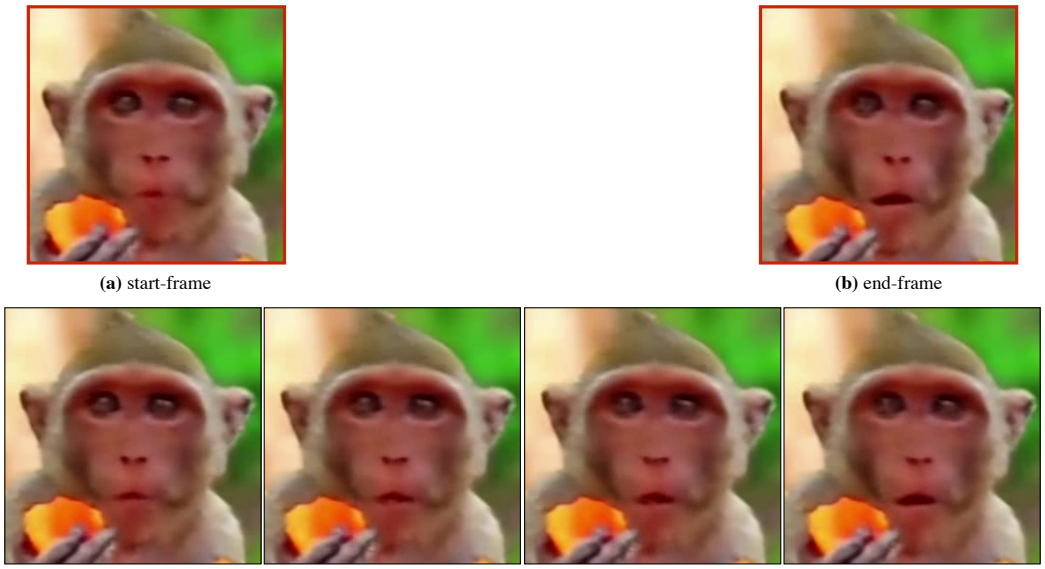
7.4.1 Slowing-down and Speeding-up a Video

We use simple operation on latent-codes to slow-down (known as temporal super-resolution) and speed-up a video.

Temporal super-resolution: Given a frame x_t and x_{t+1}, we insert an arbitrary number of frames between by linearly interpolating their latent codes:

$$g\Big(\alpha f(x_t) + (1-\alpha) f(x_{t+1})\Big), \quad \alpha \in [0,1] \tag{7.5}$$

Figure 7.13 and Figure 7.14 show examples of temporal super-resolution using a video-specific autoencoder trained for a monkey video. Given a start-frame and an end-frame, we insert eight frames between them (showing every other generated frame here).



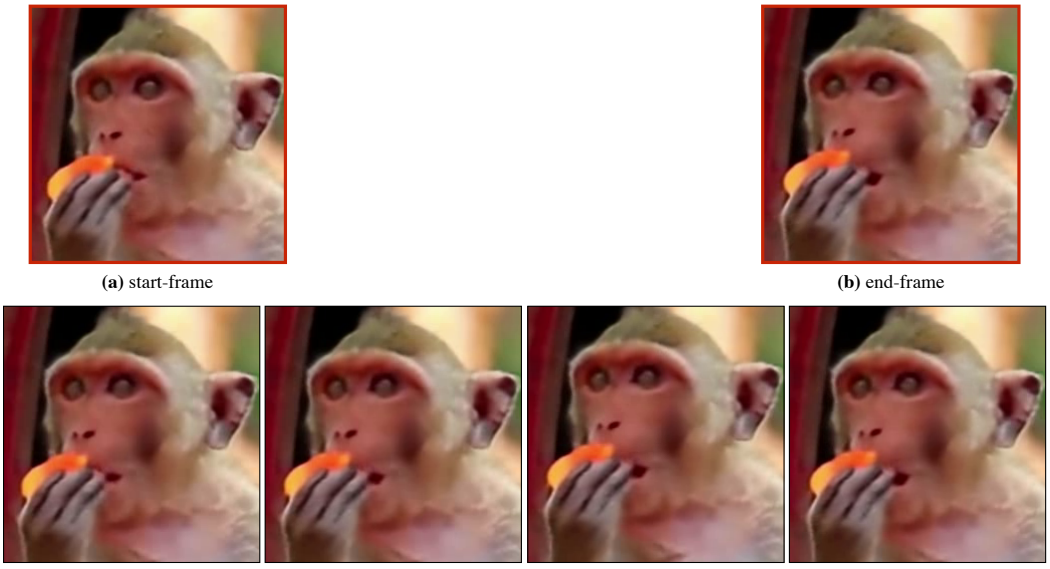
(a) start-frame

(b) end-frame

(c) intermediate frames

Figure 7.13: **Temporal Super-Resolution:** Given (**a**) a start-frame and (**b**) an end-frame, (**c**) we insert frames between them by interpolating the latent codes. We did not optimize for temporal super-resolution. Our approach appropriately captures opening of the mouth in the intermediate frames.

For the quantitative evaluation, we contrasted our approach with an off-the-shelf SuperSlowMo [202] model trained on a large dataset in a supervised manner specifically for the task of temporal super-resolution. We use in-the-wild videos for evaluation and show interpolation between every 4 frames. The original frames are used as a ground-truth for evaluation. We compute PSNR and SSIM scores between the original frames and interpolated frames. **Higher is Better**. Table 7.1 shows the performance of our approach with SuperSlowMo [202]. We achieve competitive results without using an extra data, or temporal information, or any other source of supervision. Our approach, however, requires training on the frames of target video whereas SuperSlowMo does not. In that sense, our approach is not "online" whereas SuperSlowMo is. Note that video-specific autoencoders also enable arbitrary temporal resampling of a video, in contrast to previous methods that are often trained for a fixed resampling factor without being able to generalize to others [262, 265, 311].



(a) start-frame

(b) end-frame

(c) intermediate frames

Figure 7.14: **Temporal Super-Resolution:** Given (**a**) a start-frame and (**b**) an end-frame, (**c**) we insert frames between them by interpolating the latent codes. We did not optimize for temporal super-resolution. Our approach captures subtle details such as mouth opening even though it is not fully visible.

	Extra Data	Temporal Information	Supervision	Online	PSNR↑	SSIM↑
SuperSlowMo [202]	✓	✓	✓	✓	34.167	0.720
Ours	✗	✗	✗	✗	34.306	0.726

Table 7.1: **Temporal Super-Resolution:** We contrast our approach with an off-the-shelf SuperSlowMo [202] model trained on a large dataset in a supervised manner specifically for the task of temporal super-resolution. We use in-the-wild videos for evaluation and show interpolation between every 4 frames. The original frames are used as a ground-truth for evaluation. We compute PSNR and SSIM scores between the original frames and interpolated frames (**Higher is Better**). Our approach achieves competitive performance without using any extra data, temporal information, or extra source of supervision. On the flip side, our approach requires to see multiple frames of a video to learn a video-specific autoencoder. In that sense, it is not "online". However, SuperSlowMo does not require training on the frames of target video.

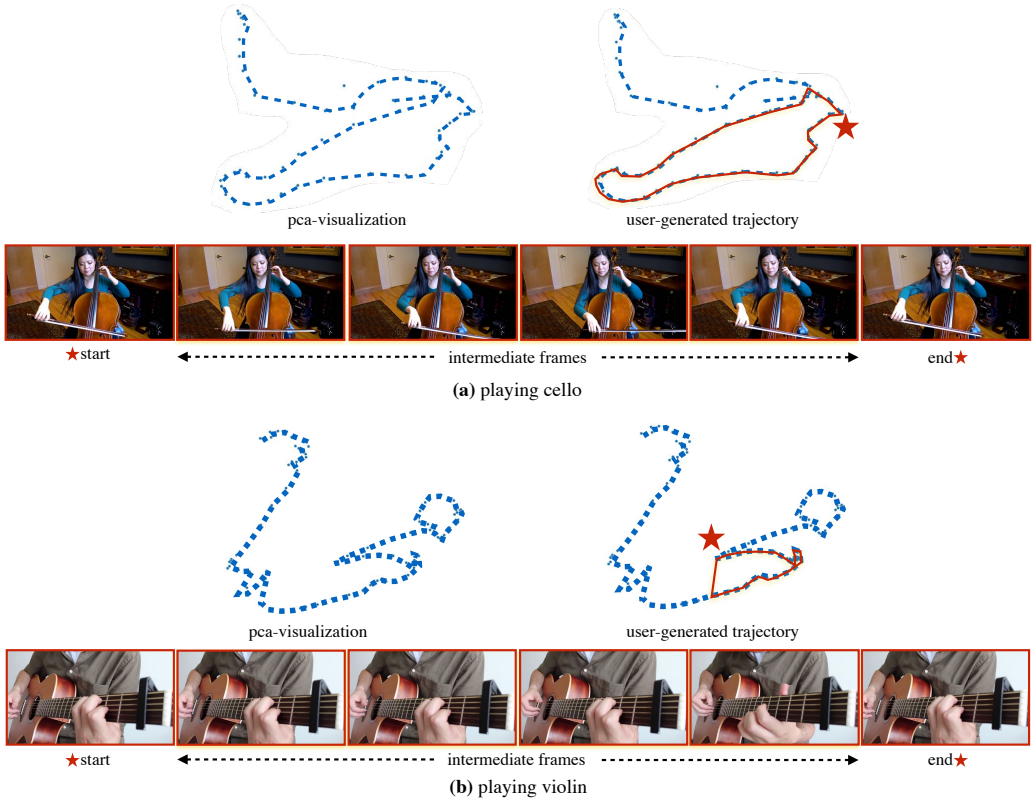


Figure 7.15: **Video Textures:** We can create arbitrary length video (also known as video textures) from an existing short sequence by making continuous loops. Our ability to associate frames within a video using the latent codes enable us to create infinite loops for repetitive motion. We can also create loops between two points which may not overlap but are close-by (as shown in (b)). This is possible due to our ability to interpolate between two points. Importantly, a user can create trajectories they want for various videos using PCA visualization.

7.4.2 Frame-level Association

Corresponding Frames within a Video: By matching frames using a cosine similarity on latent codes $f(x_1), f(x_2)$, we can find corresponding frames. We use this ability to construct infinite loop video textures (discussed in Section 7.4.3).
Corresponding Frames across Videos: Recall that video-specific autoencoders are trained on unordered frames. We exploit this property to learn autoencoders on non-contiguous clips from a longer video, or even two seperate videos V_1 and V_2 by simply pooling together their set of frames $V = V_1 \cup V_2$ during training. This allows us to learn a single latent code representation supporting comparisons across videos:

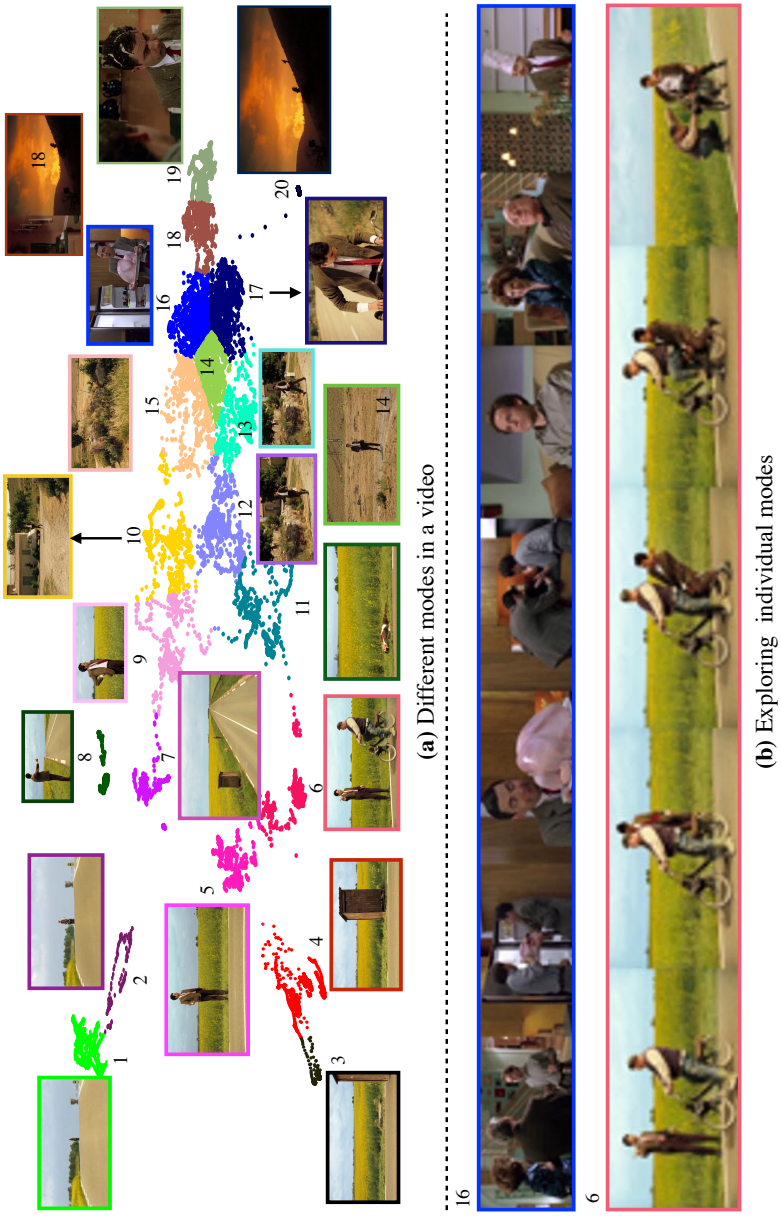


Figure 7.16: **User-Controllable Video Exploration:** (**a**) We embed an 11-minute video of Mr Bean with a video-specific autoencoder. Simple clustering and 2D PCA visualization of the associated latent frame codes allow users to explore visual "modes" in the clip without having to play it forward. (**b**) We can also explore the keyframes of each cluster or mode to get a sense of event happening in it.

Figure 7.17: $10\times$ **Spatial Super-Resolution:** The reprojection property of autoencoder enables us to input (**a**) a low-res input once the autoencoder is trained, and yet yields (**b**) a hi-res output. We did not optimize for spatial super-resolution.

$$\text{Match}(x_2) =_{x_1 \in V_1} \text{dist}(f(x_2), f(x_1)), \quad \forall x_2 \in V_2 \tag{7.6}$$

Video Retargeting: We show the above approach can be used as a simple video retargeting engine by representing each input frame with its residual latent code compared to the average:

$$\begin{aligned} \text{Retarget}(x_1) &= g(f(x_1) - \mu_1 + \mu_2), \quad \text{where} \\ \mu_i &= \frac{1}{|V_i|} \sum_{x \in V_i} f(x), \quad i = 1, 2. \end{aligned} \tag{7.7}$$

Once we represent each source frame by its residual code, we then add this residual code to the average code of the target video (and then decode with g). Figure 7.8 shows the results of both frame-level association when using two different videos and also the results of retargeting from one video to another.

7.4.3 Video Textures

We can create infinite-loop video textures [370] from a short sequence as shown in Figure 7.15. Two important distinctions from prior work: (1) PCA visualization

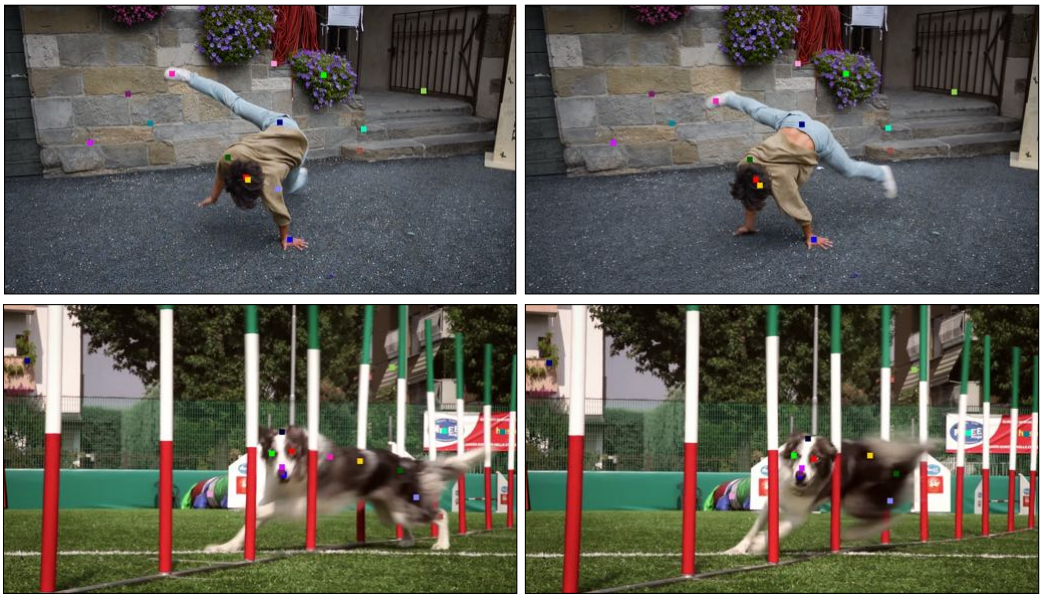
(a) input **(b)** output

Figure 7.18: $10\times$ **Spatial Super-Resolution:** The reprojection property of autoencoder enables us to input **(a)** a low-res input once the autoencoder is trained, and yet yields **(b)** a hi-res output. We did not optimize for spatial super-resolution.

enables a user to define the loops and even paths for the target video; and (2) our ability to interpolate between two points allows us to go beyond the requirements of exact match (shown in Figure 7.15-(b), we connect far-away points to make a loop).

7.4.4 Modes and Average Video Exploration

Video summarization and exploration tools allow users to quickly peruse large amounts of video (e.g., consider an analyst who must process large amounts of surveillance video). We find that our PCA-based 2D visualization allows for users to quickly visualize average summaries using either of two methods depending on the application: (1) selecting median frame from the temporal sequence in a mode; or (2) averaging arbitrary subsets of video frames $V_{\text{subset}} \subseteq V$:



(a) initial-frame **(b)** pixel correspondences

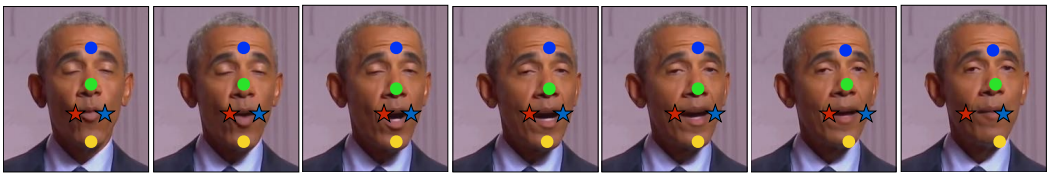
Figure 7.19: **Pixel Correspondences:** (**a**) Given keypoints in a frame, (**b**) we use the pixel codes for each frame using the video-specific autoencoder (Section 7.3) to find them in a different frame. Despite large structural changes, we are able to map the keypoints at the corresponding semantic position. E.g. the pink point on the shoe of the person in top-row, and the purple point towards the lower back of dog in bottom-row.

$$g\left(\frac{1}{|V_{\text{subset}}|}\sum_{x\in V_{\text{subset}}} f(x)\right). \tag{7.8}$$

Figure 7.16-(a) summarizes a 11 minutes long video. Our approach also allows a user to quickly see the contents in a cluster or a mode.

Keyframes for a Cluster: Our approach is drastically simple. We partition the points in the cluster using simple k-means clustering [41]. A keyframe for a cluster is the original data-point closest to the center of a sub-cluster. A user can define how many keyframes are needed. A user can do a continuous temporal-zoom to see further by clicking on a keyframe. This is possible because we can iteratively partition the points in the sub-cluster belonging to the keyframe. Our approach allows the development of a simple interface for users to navigate the latent space and interactively click on frames of interest.

As shown in Figure 7.16-(b), suppose a user selects *Cluster-6*. Our approach provides the keyframes for this cluster. One can glance at the keyframes of Cluster-



(a) Obama video

(b) Freestyle dancing

Figure 7.20: **Pixel Correspondences:** We use the pixel codes for each frame using the video-specific autoencoder (Section 7.3). We do a simple nearest neighbor via cosine-similarity measure using pixel codes to find correspondences in the adjacent frames. We show correspondences for certain points in (a) an Obama video; and (b) a freestyle dancing. We observe that the points map to corresponding locations in the adjacent frames.

6 and get a sense of the activity happening in it, i.e., Mr. Bean is asking for a ride on the road, he gets the ride and is struggling on the moped, and finally something goes wrong with the moped. Similarly, we see activities in *Cluster-16* – three people have joined Mr. Bean for a dinner and are having a fun conversation. Meanwhile, Mr. Bean is trying to cook something.

7.4.5 Spatial Super-Resolution

We use the reprojection property of the autoencoder (discussed in Section 7.3.2) to do spatial super-resolution. Figure 7.17 and Figure 7.18 show various examples of $10\times$ spatial super-resolution. The convolutional autoencoder allows us to use videos of varying resolution. Crucially, we get temporally smooth outputs without using any temporal information.

7.4.6 Dense Pixel Correspondences

We now use our pixel codes (Section 7.3.1) to establish a pixel-wise correspondences in the adjacent video frames. We do simple nearest neighbors using cosine similarity measure to find best match for each pixel. Shown in Figure 7.19, we are given the keypoints for a frame – we find correspondences in a different frame. Despite the substantial structural changes in the two examples, our approach is able to reliably map the keypoints. We also show the results of pixel-correspondences for certain points in Figure 7.20: (a) an Obama facial video; and (b) a freestyle dancing

video. We observe that points map to corresponding locations across the frames. The preliminary results show the potential of video-specific autoencoders for pixel-level correspondences even though it was never trained for it. Future work will help us study this aspect in more details.

7.4.7 Discussion on User-Interface

An autoencoder trained using individual frames of a specific video without any temporal information can be used for a wide variety of tasks in video analytics without even optimizing for any of those tasks. This feat could be possible by careful analysis of spatial and temporal properties of a video-specific autoencoder. We are limited by our imagination in demonstrating different things we could do. However, we hope that an interface based on this simple representation can enable users to easily design new application (not even explored in this work) without much overhead such as simple algebraic operations on the latent codes. We also hope that every video uploaded on web get its autoencoder. It may not take a long time to train a model but it can drastically reduce the amount of resources required for video transmission, exploration, and processing because once trained, the model can be used for fairly large number of applications without much constraints. Future work in this direction may enable faster training of an autoencoder for a new video or fine-tuning an existing representation on a few frames of a video to accomplish this feat.

Chapter 8

Human Expressable and Understandable Scene Cues

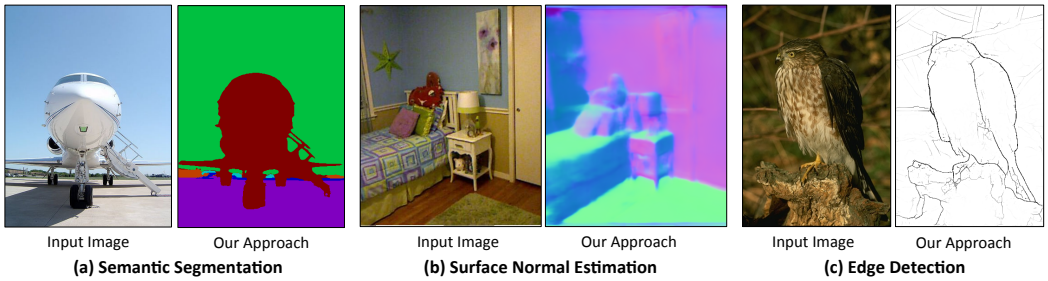


Figure 8.1: **Scene Cues:** Our framework applied to three different pixel prediction problems with minor modification of the architecture (last layer) and training process (epochs). Note how our approach recovers the fine details for segmentation (left), surface normal (middle), and semantic boundaries for edge detection (right).

8.1 Scene Cues and Constraints

A number of computer vision problems can be formulated as a dense pixel-wise prediction problem. These include **low-level** tasks such as edge detection [94,282,478] and optical flow [17,118], **mid-level** tasks such as depth/normal recovery [23,102, 104,368,452], and **high-level** tasks such as keypoint prediction [56,143,347,461], object detection [186], and semantic segmentation [65,112,162,267,301,389]. Though such a formulation is attractive because of its generality, one obvious difficulty is the enormous associated output space. For example, a 100×100 image with 10 discrete class labels per pixel yields an output label space of size 10^5. One strat-

egy is to treat this as a *spatially-invariant label prediction* problem, where one predicts a separate label per pixel using a convolutional architecture. Neural networks with convolutional output predictions, also called Fully Convolutional Networks (FCNs) [65, 267, 283, 334], appear to be a promising architecture in this direction.

But is this the ideal formulation of dense pixel-labeling? While *computationally efficient* for generating predictions at test time, we argue that it is *not statistically efficient* for gradient-based learning. Stochastic gradient descent (SGD) assumes that training data are sampled independently and from an identical distribution () [46]. Indeed, a commonly-used heuristic to ensure approximately samples is random permutation of the training data, which can significantly improve learnability [243]. It is well known that pixels in a given image are highly correlated and not independent [193]. Following this observation, one might be tempted to randomly permute pixels during learning, but this destroys the spatial regularity that convolutional architectures so cleverly exploit! In this work, we explore the tradeoff between statistical and computational efficiency for convolutional learning, and investigate simply *sampling* a modest number of pixels across a small number of images for each SGD batch update, exploiting convolutional processing where possible.

Contributions: (1) We experimentally validate that, thanks to spatial correlations between pixels, just sampling a small number of pixels per image is sufficient for learning. More importantly, sampling allows us to train end-to-end particular nonlinear models not earlier possible, and explore several avenues for improving both the efficiency and performance of FCN-based architectures. (2) In contrast to the vast majority of models that make use of pre-trained networks, we show that pixel-level optimization can be used to train models *tabula rasa*, or "from scratch" with simple random Gaussian initialization. Intuitively, pixel-level labels provide a large amount of supervision compared to image-level labels, given proper accounting of correlations. Without using any extra data, our model outperforms previous unsupervised/self-supervised approaches for semantic segmentation on PASCAL VOC-2012 [108], and is competitive to fine-tuning from pre-trained models for surface normal estimation. (3). Using a single architecture and without much modification in parameters, we show state-of-the-art performance for edge detection on BSDS [12], surface normal estimation on NYUDv2 depth dataset [393], and semantic segmentation on the PASCAL-Context dataset [302].

8.2 Background

In this section, we review related work by making use of a unified notation that will be used to describe our architecture. We address the pixel-wise prediction problem where, given an input image X, we seek to predict outputs Y. For pixel location p, the output can be binary $Y_p \in \{0, 1\}$ (e.g., edge detection), multi-class $Y_p \in \{1, \ldots, K\}$ (e.g., semantic segmentation), or real-valued $Y_p \in \mathbb{R}^N$ (e.g., sur-

face normal prediction). There is rich prior art in modeling this prediction problem using hand-designed features (representative examples include [11, 59, 94, 149, 255, 303, 366, 389, 423, 430, 485]).

Convolutional prediction: We explore *spatially-invariant* predictors $f_{\theta,p}(X)$ that are end-to-end trainable over model parameters θ. The family of fully-convolutional and skip networks [283, 334] are illustrative examples that have been successfully applied to, e.g., edge detection [478] and semantic segmentation [53, 65, 112, 118, 259, 267, 301, 315, 332]. Because such architectures still produce separate predictions for each pixel, numerous approaches have explored post-processing steps that enforce spatial consistency across labels via e.g., bilateral smoothing with fully-connected Gaussian CRFs [65, 235, 499] or bilateral solvers [29], dilated spatial convolutions [489], LSTMs [53], and convolutional pseudo priors [477]. In contrast, our work does *not* make use of such contextual post-processing, in an effort to see how far a pure "pixel-level" architecture can be pushed.

Multiscale features: Higher convolutional layers are typically associated with larger receptive fields that capture high-level global context. Because such features may miss low-level details, numerous approaches have built predictors based on multiscale features extracted from multiple layers of a CNN [88, 102, 104, 112, 332, 452]. Hariharan et al. [162] use the evocative term "hypercolumns" to refer to features extracted from multiple layers that correspond to the same pixel. Let

$$h_p(X) = [c_1(p), c_2(p), \ldots, c_M(p)]$$

denote the multi-scale hypercolumn feature computed for pixel p, where $c_i(p)$ denotes the feature vector of convolutional responses from layer i centered at pixel p (and where we drop the explicit dependance on X to reduce clutter). Prior techniques for up-sampling include shift and stitch [267], converting convolutional filters to dilation operations [65] (inspired by the *algorithme à trous* [273]), and deconvolution/unpooling [118, 267, 315]. We similarly make use of multi-scale features, along with *sparse* on-demand upsampling of filter responses, with the goal of reducing the memory footprints during learning.

Pixel-prediction: One may cast the pixel-wise prediction problem as operating over the hypercolumn features where, for pixel p, the final prediction is given by

$$f_{\theta,p}(X) = g(h_p(X)).$$

We write θ to denote both parameters of the hypercolumn features h and the pixel-wise predictor g. Training involves back-propagating gradients via SGD to update θ. Prior work has explored different designs for h and g. A dominant trend is defining a linear predictor on hypercolumn features, e.g., $g = w \cdot h_p$. FCNs [267] point out that linear prediction can be efficiently implemented in a coarse-to-fine manner by upsampling coarse predictions (with deconvolution) rather than upsampling coarse features. DeepLab [65] incorporates filter dilation and applies similar deconvolution and linear-weighted fusion, in addition to reducing the dimensionality

of the fully-connected layers to reduce memory footprint. ParseNet [259] added spatial context for a layer's responses by average pooling the feature responses, followed by normalization and concatenation. HED [478] output edge predictions from intermediate layers, which are deeply supervised, and fuses the predictions by linear weighting. Importantly, [301] and [112] are noteable exceptions to the linear trend in that *non-linear* predictors g are used. This does pose difficulties during learning - [301] precomputes and stores superpixel feature maps due to memory constraints, and so cannot be trained end-to-end.

Sampling: We demonstrate that sparse sampling of hypercolumn features allows for exploration of highly nonlinear g, which in turn significantly boosts performance. Our insight is inspired by past approaches that use sampling to training networks for surface normal estimation [23] and image colorization [241], though we focus on general design principles by analyzing the impact of sampling for efficiency, accuracy, and *tabula rasa* learning for diverse tasks.

Accelerating SGD: There exists a large literature on accelerating stochastic gradient descent. We refer the reader to [46] for an excellent introduction. Though naturally a sequential algorithm that processes one data example at a time, much recent work focuses on mini-batch methods that can exploit parallelism in GPU architectures [83] or clusters [83]. One general theme is efficient online approximation of second-order methods [45], which can model correlations between input features. Batch normalization [194] computes correlation statistics between samples in a batch, producing noticeable improvements in convergence speed. Our work builds similar insights directly into convolutional networks without explicit second-order statistics.

8.3 PixelNet

This section describes our approach for pixel-wise prediction, making use of the notation introduced in the previous section. We first formalize our pixelwise prediction architecture, and then discuss statistically efficient mini-batch training.

Architecture: As in past work, our architecture makes use of multiscale convolutional features, which we write as a hypercolumn descriptor:

$$h_p = [c_1(p), c_2(p), \ldots, c_M(p)]$$

We learn a nonlinear predictor $f_{\theta,p} = g(h_p)$ implemented as a multi-layer perceptron (MLP) [40] defined over hypercolumn features. We use a MLP, which can be implemented as a series of "fully-connected" layers followed by ReLU activation functions. Importantly, the last layer must be of size K, the number of class labels or real valued outputs being predicted. See Figure 8.2.

Sparse predictions: We now describe an efficient method for generating sparse pixel predictions, which will be used at train-time (for efficient mini-batch gener-

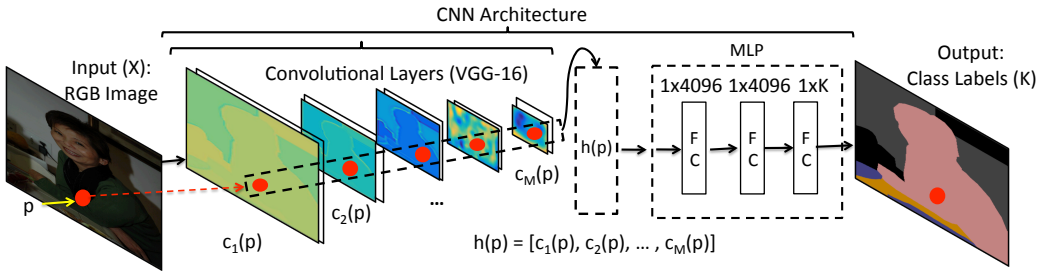


Figure 8.2: **PixelNet:** We input an image to a convolutional neural network, and extract hypercolumn descriptor for a sampled pixel from multiple convolutional layers. The hypercolumn descriptor is then fed to a multi-layer perceptron (MLP) for the non-linear optimization, and the last layer of MLP outputs the required response for the task. See text for more details about the use of network at training/test time.

ation). Assume that we are given an image X and a sparse set of (sampled) pixel locations $P \subset \Omega$, where Ω is the set of all pixel positions.

1. Perform a forward pass to compute dense convolutional responses at all layers $\{c_i(p) : \forall i, p \in \Omega\}$
2. For each sampled pixel $p \in P$, compute its hypercolumn feature h_p *on demand* as follows:
   (a) For each layer i, compute the 4 discrete locations in the feature map c_i closest to p
   (b) Compute $c_i(p)$ via bilinear interpolation
3. Rearrange the sparse of hypercolumn features $\{h_p : p \in P\}$ into a matrix for downstream processing (e.g., MLP classification).

The above pipeline only computes $|P|$ hypercolumn features rather than full dense set of size $|\Omega|$. We experimentally demonstrate that this approach offers an excellent tradeoff between amortized computation (to compute $c_i(p)$) and reduced storage (to compute h_p). Note that our multi-scale sampling layer simply acts as a selection operation, for which a (sub) gradient can easily be defined. This means that backprop can also take advantage of sparse computations for nonlinear MLP layers and convolutional processing for the lower layers.

Mini-batch sampling: At each iteration of SGD training, the true gradient over the model parameters θ is approximated by computing the gradient over a relatively small set of samples from the training set. Approaches based on FCN [267] include features for all pixels from an image in a mini-batch. As nearby pixels in an image are highly correlated [193], sampling them will not hurt learning. To ensure a diverse set of pixels (while still enjoying the amortized benefits of convolutional processing), we use a modest number of pixels ($\sim 2,000$) per image, but sample many

images per batch. Naive computation of dense grid of hypercolumn descriptors takes almost all of the (GPU) memory, while $2,000$ samples takes a small amount using our sparse sampling layer. This allows us to explore more images per batch, significantly increasing sample diversity.

Dense predictions: We now describe an efficient method for generating dense pixel predictions with our network, which will be used at test-time. Dense prediction proceeds by following step (1) from above; and instead of sampling in (2) above, we take all the pixels now. This produces a dense grid of hypercolumn features, which are then (3) processed by pixel-wise MLPs implemented as 1x1 filters (representing each fully-connected layer). The memory intensive portion of this computation is the dense grid of hypercolumn features. This memory footprint is reasonable at test time because a single image can be processed at a time, but at train-time, we would like to train on batches containing many images as possible (to ensure diversity).

8.3.1 Analysis & Generalizability

In this section, we analyze the properties of pixel-level optimization using semantic segmentation and surface normal estimation to understand the design choices for pixel-level architectures. We chose the two varied tasks (classification and regression) for analysis to verify the generalizability of these findings. We use a single-scale 224×224 image as input. We also show *sampling* augmented with careful *batch-normalization* can allow for a model to be trained from scratch (without pre-trained ImageNet model as an initialization) for semantic segmentation and surface normal estimation.

Default network: For most experiments we fine-tune a VGG-16 network [395]. VGG-16 has 13 convolutional layers and three fully-connected (*fc*) layers. The convolutional layers are denoted as $\{1_1, 1_2, 2_1, 2_2, 3_1, 3_2, 3_3, 4_1, 4_2, 4_3, 5_1, 5_2, 5_3\}$. Following [267], we transform the last two *fc* layers to convolutional filters[1], and add them to the set of convolutional features that can be aggregated into our multi-scale hypercolumn descriptor. To avoid confusion with the **fc** layers in our MLP, we will henceforth denote the *fc* layers of VGG-16 as conv-6 and conv-7. We use the following network architecture (unless otherwise specified): we extract hypercolumn features from conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ with on-demand interpolation. We define a MLP over hypercolumn features with 3 fully-connected (fc) layers of size $4,096$ followed by ReLU [238] activations, where the last layer outputs predictions for K classes (with a soft-max/cross-entropy loss) or K outputs with a euclidean loss for regression.

Semantic Segmentation: We use training images from PASCAL VOC-2012 [108] for semantic segmentation, and additional labels collected on 8498 images by Har-

[1]For alignment purposes, we made a small change by adding a spatial padding of 3 cells for the convolutional counterpart of *fc6* since the kernel size is 7×7.

iharan et al. [161]. We used the held-out (non-overlapping) validation set to show most analysis. However, at some places we have used the test set where we wanted to show comparison with previous approaches. We report results using the standard metrics of region intersection over union (**IoU**) averaged over classes (higher is better). We mention it as IoU (V) when using the validation set for evaluation, and IoU (T) when showing on test set.

Surface Normal Estimation: The NYU Depth v2 dataset [393] is used to evaluate the surface normal maps. There are 1449 images, of which 795 are trainval and remaining 654 are used for evaluation. Additionally, there are $220,000$ frames extracted from raw Kinect data. We use the normals of Ladicky et al. [239] and Wang et al. [452], computed from depth data of Kinect, as ground truth for 1449 images and 220K images respectively. We compute six statistics, previously used by [23, 102, 123–125, 452], over the angular error between the predicted normals and depth-based normals to evaluate the performance – **Mean**, **Median**, **RMSE**, **11.25**°, **22.5**°, and **30**° – The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better.

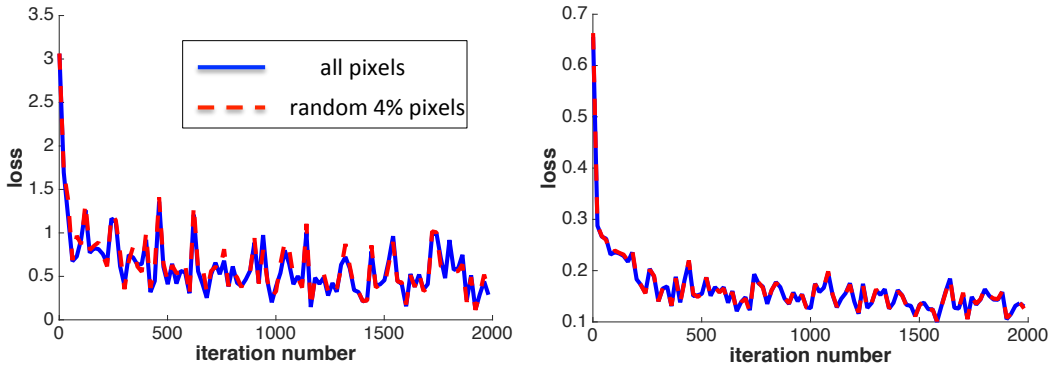
8.3.2 Sampling

We examine how sampling a few pixels from a fixed set of images does not harm convergence. Given a fixed number of (5) images per batch, we find that sampling a small fraction (4%) of the pixels per image does not affect learnability (Figure 8.3 and Table 8.1). This validates our hypothesis that much of the training data for a pixel-level task is correlated within an image, implying that randomly sampling a few pixels is sufficient. Our results are consistent with those reported in Long *et al.* [267], who similarly examine the effect of sampling a fraction (25-50%) of patches per training image.

Long *et al.* [267] also perform an additional experiment where the total number of pixels in a batch is kept constant when comparing different sampling strategies. While this ensures that each batch will contain more diverse pixels, each batch will also process a larger number of images. If there are no significant computational savings due to sampling, additional images will increase wall-clock time and slow convergence. In the next section, we show that adding additional computation after sampling (by replacing a linear classifier with a multi-layer perceptron) fundamentally changes this tradeoff (Table 8.4).

8.3.3 Linear vs. MLP

Most previous approaches have focussed on linear predictors combining the information from different convolutional layers (also called 'skip-connections'). Here we contrast the performance of non-linear models via MLP with corresponding linear models. For this analysis, we use a VGG-16 (pre-trained on ImageNet) as ini-



a. Semantic Segmentation b. Surface Normal Estimation

Figure 8.3: Given a fixed number of (5) images per SGD batch, we analyse convergence properties using all pixels vs. randomly sampling 4% , or $2,000$ pixels for semantic segmentation and surface normal estimation. This experiment suggest that sampling does not hurt convergence.

Method	IoU (V)	Mean	Median	RMSE	11.25°	22.5°	30°
All Pixels	44.4	25.6	19.9	32.5	29.1	54.9	66.8
Random 4% Pixels	44.6	25.7	20.1	32.5	28.9	54.7	66.7

Table 8.1: **Sampling:** We demonstrate that sampling few pixels for each mini-batch yields similar accuracy as using all pixels. The results are computed on models trained for 10 epochs and $10,000$ iterations for semantic segmentation and surface normal estimation, respectively.

tialization and use skip-connections from conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ layers to show the benefits of a non-linear model over a linear model. We randomly sample $2,000$ pixels per image from a set of five 224×224 images per SGD iteration for the optimization.

A major challenge in using skip-connections is how to combine the information as the dynamic range varies across the different layers. The top-row in Table 8.2 shows how the model leads to degenerate outputs for semantic segmentation when 'naively' concatenating features from different convolutional layers in a linear model. Similar observation was made by [259]. To counter this issue, previous work has explored normalization [259], scaling [162], etc. We use batch-normalization [194] for the convolutional layers before concatenating them to properly train a model with a linear predictor. The middle-row in Table 8.2 shows how adding batch-normalization allows us to train a linear model for semantic segmentation, and improve the performance for surface normal estimation. While we have to take care of normalization for linear models, we do not need them while using

Method	IoU (T)	Mean	Median	RMSE	11.25°	22.5°	30°
Linear (no bn)	3.6	24.8	19.4	31.2	28.7	56.4	68.8
Linear (bn)	62.4	22.5	16.1	29.7	37.0	62.8	73.3
MLP	**67.4**	**19.8**	**12.0**	**28.0**	**47.9**	**70.0**	**77.8**

Table 8.2: **Linear vs. MLP:** A multi-layer perceptron over hypercolumn features gives better performance over linear model without requiring normalization/scaling. Note: bn stands for batch-normalization.

a MLP and can naively concatenate features from different layers. The last row in Table 8.2 shows the performance on different tasks when using a MLP. Note that performance of linear model (with batch-normalization) is similar to one obtained by Hypercolum [162] (62.7%), and FCN [267] (62%).

***Deconvolution* vs. *on-demand* compute:** A naive implementation of our approach is to use deconvolution layers to upsample conv-layers, followed by feature concatenation, and mask out the pixel-level outputs. This is similar to the sampling experiment of Long *et al.* [267]. While reasonable for a linear model, naively computing a dense grid of hypercolumn descriptors and processing them with a MLP is impossible if *conv-7* is included in the hypercolumn descriptor (the array dimensions exceed *INT_MAX*). For practical purposes, if we consider skip-connections only from conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ layers at cost of some performance, naive deconvolution would still take more than **12X** memory compared to our approach. Slightly better would be masking the dense grid of hypercolumn descriptors before MLP processing, which is still **8X** more expensive. Most computational savings come from not being required to keep an extra copy of the data required by deconvolution and concatenation operators. Table 8.3 highlights the differences in computational requirements between deconvolution vs. *on-demand* compute (for the more forgiving setting of $\{1_2, 3_3, 5_3\}$-layered hypercolumn features). Clearly, *on-demand* compute requires less resources.

Does statistical diversity matter? We now analyze the influence of statistical diversity on optimization given a fixed computational budget (7GB memory on a NVIDIA TITAN-X). We train a non-linear model using 1 image $\times$ 40,000 pixels per image vs. 5 images $\times$ 2,000 pixels per image. Table 8.4 shows that sampling fewer pixels from more images outperforms more pixels extracted from fewer images. This demonstrates that statistical diversity outweighs the computational savings in convolutional processing when a MLP classifier is used.

8.3.4 Training from scratch

Prevailing methods for training deep models make use of a pre-trained (e.g., ImageNet [362]) model as initialization for fine-tuning for the task at hand. Most

Model	#features	sample (#)	Memory (MB)	Disk Space (MB)	BPS
FCN-32s [267]	4,096	50,176	2,010	518	20.0
FCN-8s [267]	4,864	50,176	2,056	570	19.5
FCN/Deconvolution					
Linear	1,056	50,176	2,267	1,150	6.5
MLP	1,056	50,176	3,914	1,232	1.4
FCN/Sampling					
Linear	1,056	2,000	2,092	1,150	5.5
MLP	1,056	2,000	2,234	1,232	5.1
PixelNet/On-demand					
Linear	1,056	2,000	322	60	43.3
MLP	1,056	2,000	465	144	24.5
MLP (+*conv-7*)	5,152	2,000	1,024	686	8.8

Table 8.3: **Computational Requirements:** We record the number of dimensions for hypercolumn features from conv-$\{1_2, 3_3, 5_3\}$, number of samples (for our model), memory usage, model size on disk, number of mini-batch updates per second (BPS measured by forward/backward passes). We use a single 224×224 image as the input. We compared our network with FCN [267] where a deconvolution layer is used to upsample the result in various settings. Besides FCN-8s and FCN-32s here we first compute the upsampled feature map, and then apply the classifiers for FCN [267]. Clearly from the table, our approach require less computational resources as compared to other settings.

Method	IoU_1 (V)	IoU_2 (V)	Mean	Median	RMSE	11.25°	22.5°	30°
$1 \times 40,000$	7.9	15.5	24.8	19.5	31.6	29.7	56.1	68.5
$5 \times 2,000$	**38.4**	**47.9**	**23.4**	**17.2**	**30.5**	**33.9**	**60.6**	**71.8**

Table 8.4: **Statistical Diversity Matters:** For a given computational budget, using diverse set of pixels from more images shows better performance over more pixels from a few images. IoU_1 and IoU_2 show performance for 10K and 20K iterations of SGD for semantic segmentation. For surface normal estimation, we show performance for 10K iterations of SGD. This suggest that sampling leads to faster convergence.

Initialization	IoU (T)	Mean	Median	RMSE	11.25°	22.5°	30°
ImageNet	67.4	19.8	12.0	28.2	47.9	70.0	77.8
Random	48.7	21.2	13.4	29.6	44.2	66.6	75.1
Geometry	52.4	-	-	-	-	-	-

Table 8.5: **Initialization:** We study the influence of initialization on accuracy. PixelNet can even be trained reasonably well with random Gaussian initialization or starting from a model trained for surface normal prediction (Geometry).

network architectures (including ours) improve in performance with pre-trained models. A major concern is the availability of sufficient data to train deep models for pixel-level prediction problems. However, because our optimization is based on randomly-sampled pixels instead of images, there is potentially more unique data available for SGD to learn a model from a random initialization. We show how *sampling* and *batch-normalization* enables models to be trained from scratch. This enables our networks to be used in problems with limited training data, or where natural image data does not apply (e.g., molecular biology, tissue segmentation etc). We will show that our results also have implications for unsupervised representation learning [5, 93, 241, 298, 330, 493].

Random Initialization: We randomly initialize the parameters of a VGG-16 network from a Gaussian distribution. Training a VGG-16 network architecture is not straight forward, and previously required stage-wise training for the image classification task [395]. It seems daunting to train such a model from scratch for a pixel-level task where we want to learn both coarse and fine information. In our experiments, we found batch normalization to be an effective tool for converging a model trained from scratch.

We train the models for semantic segmentation and surface normal estimation. The middle-row in Table 8.5 shows the performance for semantic segmentation and surface normal estimation trained from scratch. The model trained from scratch for surface normal estimation is within 2-3% of current state-of-the-art performing method. The model for semantic segmentation achieves 48.7% on PASCAL VOC-2012 test set when trained from scratch. To the best of our knowledge, these are the best numbers reported on these two tasks when trained from scratch, and exceeds the performance of other unsupervised/self-supervised approaches [93, 241, 330, 453, 493] that required extra ImageNet data [362].

Self-Supervision via Geometry: We briefly present the performance of models trained from our pixel-level optimization in context of self-supervision. The task of surface normal estimation does not require any human-labels, and is primarily about capturing geometric information. In this section, we explore the applicability of fine-tuning a geometry model (trained from scratch) for more semantic tasks (such as semantic segmentation and object detection). Table 8.5 (last row) and

VOC 2007 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM-v5 [115]	33.2	60.3	10.2	16.1	27.3	54.3	58.2	23.0	20.0	24.1	26.7	12.7	58.1	48.2	43.2	12.0	21.1	36.1	46.0	43.5	33.7
RCNN-Scratch [6]	49.9	60.6	24.7	23.7	20.3	52.5	64.8	32.9	20.4	43.5	34.2	29.9	49.0	60.4	47.5	28.0	42.3	28.6	51.2	50.0	40.7
VGG-16-Scratch [93]	56.1	58.6	23.3	25.7	12.8	57.8	61.2	45.2	21.4	47.1	39.5	35.6	60.1	61.4	44.9	17.3	37.7	33.2	57.9	51.2	42.4
VGG-16-Context-v2 [93]	63.6	64.4	42.0	42.9	18.9	67.9	69.5	65.9	28.2	48.1	58.4	58.5	66.2	64.9	54.1	26.1	43.9	55.9	69.8	50.9	53.0
VGG-16-Geometry	55.9	61.6	29.5	31.1	24.0	66.3	70.6	56.7	32.4	53.2	58.5	49.4	72.1	66.4	53.6	21.8	38.6	55.1	65.4	58.2	51.0
VGG-16-Geom+Seg	62.8	68.7	39.9	37.5	27.4	75.9	73.8	70.3	33.8	57.2	62.7	60.1	72.8	69.5	60.7	22.5	40.8	62.0	70.5	59.2	56.4
VGG-16-ImageNet [141]	73.6	77.9	68.8	56.2	35.0	76.8	78.1	83.1	39.7	73.4	65.6	79.6	81.3	73.3	66.1	30.4	67.2	67.9	77.5	66.5	66.9

Table 8.6: **Evaluation on VOC-2007:** Our model (VGG-16-Geometry) trained on a few indoor scene examples of NYU-depth dataset performs 9% better than scratch, and is competitive with [93] that used images from ImageNet (without labels) to train. Note that we used 110K iterations of SGD to train our model, and it took less than 3 days. [93] required training for around 8 weeks. Finally, we added a minor supervision (VGG-16-Geom+Seg) using the non-overlapping segmentation dataset and improve the performance further by 5%.

Table 8.6 shows the performance of our approach on semantic segmentation and object detection respectively. Note that the NYU depth dataset is a small indoor scene dataset and does not contain most of the categories present in PASCAL VOC dataset. Despite this, it shows 4% (segmentation) and 9% (detection) improvement over naive scratch models. It is best known result for semantic segmentation in an unsupervised/self-supervised manner, and is competitive with the previous unsupervised work [93] on object detection[2] that uses ImageNet (without labels), particularly on indoor scene furniture categories (e.g., chairs, sofa, table, tv, bottle). We posit that geometry is a good cue for unsupervised representation learning as it can learn from a few examples and can even generalize to previously unseen categories. Future work may utilize depth information from videos and use them to train models for surface normal estimation. This can potentially provide knowledge about more general categories. Finally, we add a minor supervision by taking the geometry-based model fine-tuned for segmentation, and further fine-tuning it for object detection. We get an extra 5% boost over the performance.

We now demonstrate the generalizability of PixelNet, and apply (with minor modifications) it to the high-level task of semantic segmentation, mid-level surface normal estimation, and the low-level task of edge detection.

8.4 2D Semantic Cues

Training: For all the experiments we used the publicly available *Caffe* library [200]. All trained models and code will be released. We make use of ImageNet-pretrained

[2]We used a single scale for object detection and use the same parameters as Fast-RCNN except a step-size of 70K, and fine-tuned it for 200K iterations. Doersch et al. [93] reports better results in a recent version by the use of multi-scale detection, and smarter initialization and rescaling.

Model	59-class		33-class	
	AC (%)	IU (%)	AC (%)	IU (%)
FCN-8s [266]	46.5	35.1	67.6	53.5
FCN-8s [267]	50.7	37.8	-	-
DeepLab (v2 [66])	-	37.6	-	-
DeepLab (v2) + CRF [66]	-	39.6	-	-
CRF-RNN [499]	-	39.3	-	-
ConvPP-8 [477]	-	**41.0**	-	-
PixelNet	**51.5**	**41.4**	**69.5**	**56.9**

Table 8.7: **Evaluation on PASCAL-Context [12]:** Most recent approaches [66, 477, 499] except FCN-8s, use spatial context post-processing. We achieve results better than previous approaches without any CRF. CRF post-processing could be applied to any local unary classifier (including our method).

values for all convolutional layers, but train our MLP layers "from scratch" with Gaussian initialization ($\sigma = 10^{-3}$) and dropout [403] ($r = 0.5$). We fix momentum 0.9 and weight decay 0.0005 throughout the fine-tuning process. We use the following update schedule (unless otherwise specified): we tune the network for 80 epochs with a fixed learning rate (10^{-3}), reducing the rate by $10\times$ twice every 8 epochs until we reach 10^{-5}.

Dataset: The PASCAL-Context dataset [12] augments the original sparse set of PASCAL VOC 2010 segmentation annotations [108] (defined for 20 categories) to pixel labels for the whole scene. While this requires more than 400 categories, we followed standard protocol and evaluate on the 59-class and 33-class subsets.

Evaluation Metrics: We report results on the standard metrics of pixel accuracy (AC) and region intersection over union (IU) averaged over classes (higher is better). Both are calculated with DeepLab evaluation tools[3].

Results: Table 8.10 shows performance of our approach compared to previous work. Our approach without CRF does better than previous approaches based on it. Due to space constraints, we show only one example output in Figure 8.4 and compare against FCN-8s [267]. Notice that we capture fine-scale details, such as the leg of birds.

8.4.1 More Analysis

Analysis-1: Dimension of MLP *fc* Layers. We analyze performance as a function of the size of the MLP *fc* layers. We experimented the following dimensions for our *fc* layers: $\{1024, 2048, 4096, 6144\}$. Table 8.8 lists the results. We use 5 images per

[3] https://bitbucket.org/deeplab/deeplab-public/

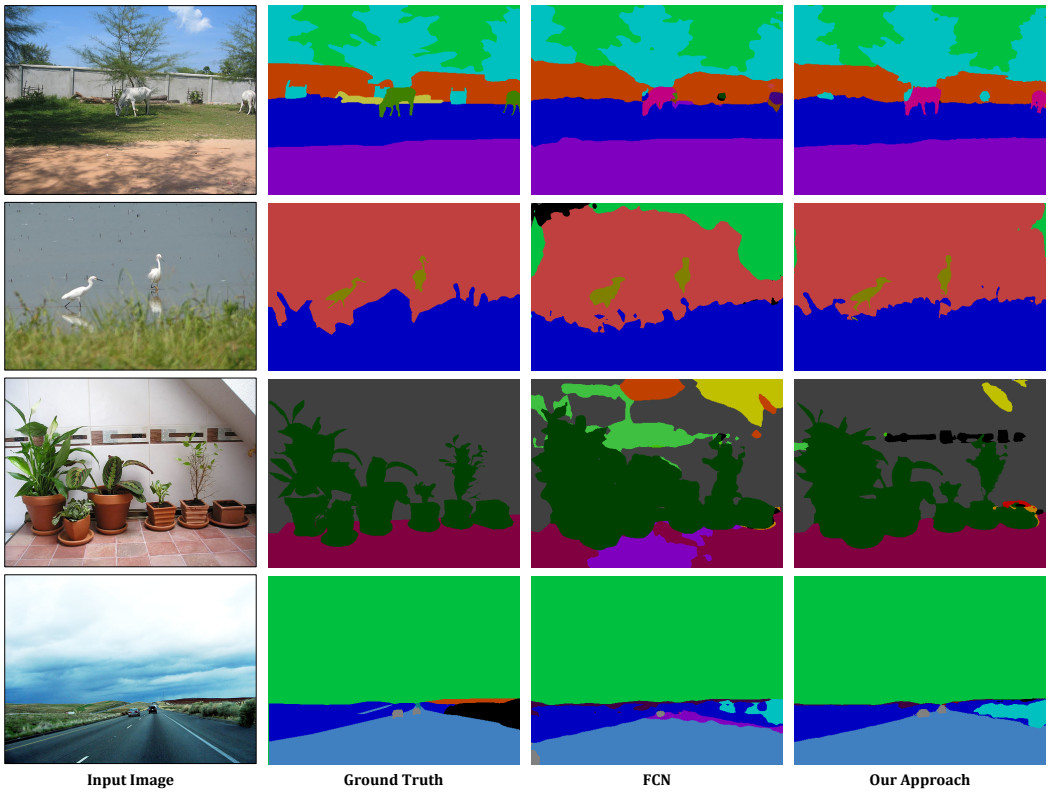


Figure 8.4: **Segmentation results on PASCAL-Context 59-class**. Our approach uses an MLP to integrate information from both lower (1_2) and higher (*conv-7*) layers, which allows us to better capture both global structure (object/scene layout) and fine details (small objects) compared to *FCN-8s*.

SGD batch and sample 2000 pixels per image, and conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ for skip connections to do this analysis. We can see that with more dimensions the network tends to learn better, potentially because it can capture more information (and with drop-out alleviating over-fitting [403]).

Dimension	*AC* (%)	*IU* (%)
1024	41.6	33.2
2048	43.2	34.2
4096	44.0	34.9
6144	44.2	35.1

Table 8.8: **Dimension of MLP fc layers:** We vary the dimension of the MLP *fc* layers on the PASCAL Context 59-class segmentation task from $\{1024, 2048, 4096, 6144\}$. We observe that 4096 is a good trade-off between performance and speed.

Analysis-2: Number of Mini-batch Samples. Continuing from our analysis on statistical diversity, we plot performance as a function of the number of sampled pixels per image. In the first sampling experiment, we fix the batch size to 5 images and sample $\{500, 1000, 2000, 4000\}$ pixels from each image. We use conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ for skip connections for this analysis. The results are shown in Table 8.9. We observe that: 1) even sampling only 500 pixels per image (on average 2% of the $\sim 20,000$ pixels in an image) produces reasonable performance after just 96 epochs. 2) performance is roughly constant as we increase the number of samples.

We also perform experiments where the samples are drawn from the same image. When sampling 2000 pixels from a single image (comparable in size to batch of 500 pixels sampled from 5 images), performance dramatically drops. This phenomena consistently holds for additional pixels (Table 8.9, bottom rows), verifying our central thesis that statistical diversity of samples can trump the computational savings of convolutional processing during learning.

$N \times M$	AC (%)	IU (%)
500×5	43.7	34.8
1000×5	43.8	34.7
2000×5	43.8	34.7
4000×5	43.9	34.9
2000×1	32.6	24.6
10000×1	33.3	25.2

Table 8.9: **Varying SGD mini-batch construction:** We vary the SGD mini-batch construction on the PASCAL Context 59-class segmentation task. $N \times M$ refers to a mini-batch constructed from N pixels sampled from each of M images (a total of N$\times$M pixels sampled for optimization). We see that a small number of pixels per image (500, or 2%) are sufficient for learning. Put in another terms, given a fixed budget of N pixels per mini-batch, performance is maximized when spreading them across a large number of images M. This validates our central thesis that statistical diversity trumps the computational savings of convolutional processing during learning.

Analysis-3: Adding conv-7. While our diagnostics reveal the importance of architecture design and sampling, our best results still do not quite reach the state-of-the-art. For example, a single-scale FCN-32s [267], without any low-level layers, can already achieve 35.1. This suggests that their penultimate *conv-7* layer does capture cues relevant for pixel-level prediction. In practice, we find that simply concatenating *conv-7* significantly improves performance.

Following the same training process, the results of our model with *conv-7* features are shown in Table 8.10. From this we can see that *conv-7* is greatly helping the performance of semantic segmentation. Even with reduced scale, we are able to obtain a similar IU achieved by FCN-8s [267], without any extra modeling of con-

Model	59-class		33-class	
	AC (%)	IU (%)	AC (%)	IU (%)
FCN-8s [266]	46.5	35.1	67.6	53.5
FCN-8s [267]	50.7	37.8	-	-
DeepLab (v2 [66])	-	37.6	-	-
DeepLab (v2) + CRF [66]	-	39.6	-	-
CRF-RNN [499]	-	39.3	-	-
ParseNet [259]	-	40.4	-	-
ConvPP-8 [477]	-	**41.0**	-	-
baseline (conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$)	44.0	34.9	62.5	51.1
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.25,0.5)	46.7	37.1	66.6	54.8
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.5)	47.5	37.4	66.3	54.0
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.5-1.0)	48.1	37.6	67.3	54.5
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (0.5-0.25,0.5,1.0)	**51.5**	**41.4**	**69.5**	**56.9**

Table 8.10: Our final results and baseline comparison on PASCAL-Context. Note that while most recent approaches spatial context post-processing [66,259,477,499], we focus on the FCN [267] per-pixel predictor as most approaches are its descendants. Also, note that we (without any CRF) achieve results better than previous approaches. CRF post-processing could be applied to any local unary classifier (including our method). Here we wanted to compare with other local models for a "pure" analysis.

text [65,259,477,499]. For fair comparison, we also experimented with single scale training with 1) half scale $0.5\times$, and 2) full scale $1.0\times$ images. We use 5 images per SGD batch, and sample 2000 pixels per image. We find the results are better without $0.25\times$ training, reaching 37.4% and 37.6% IU, respectively, even closer to the FCN-8s performance (37.8% IU). For the 33-class setting, we are already doing better with the baseline model plus *conv-7*.

Analysis-4: Multi-scale. All previous experiments process test images at a single scale ($0.25\times$ or $0.5\times$ its original size), whereas most prior work [65,259,267,499] use multiple scales from full-resolution images. A smaller scale allows the model to access more context when making a prediction, but this can hurt performance on small objects. Following past work, we explore test-time averaging of predictions across multiple scales. We tested combinations of $0.25\times$, $0.5\times$ and $1\times$. For efficiency, we just fine-tune the model trained on small scales (right before reducing the learning rate for the first time) with an initial learning rate of 10^{-3} and step size of 8 epochs, and end training after 24 epochs. The results are also reported in Table 8.10. Multi-scale prediction generalizes much better (41.0% IU). Note our pixel-wise predictions do not make use of contextual post-processing (even outperforming some methods that post-processes FCNs to do so [66,499]).

Evaluation on PASCAL VOC-2012 [108]. We use the same settings, and evaluate our approach on PASCAL VOC-2012. Our approach, without any special consideration of parameters for this dataset, achieves mAP of **69.7%**[4]. This is much better than previous approaches, e.g. 62.7% for Hypercolumns [162], 62% for FCN [267], 67% for DeepLab (without CRF) [65] etc. Our performance on VOC-2012 is similar to Mostajabi et al [301] despite the fact we use information from only 6 layers while they used information from all the layers. In addition, they use a rectangular region of 256×256 (called *sub-scene*) around the super-pixels. We posit that fine-tuning (or back-propagating gradients to conv-layers) enables efficient and better learning with even lesser layers, and without extra *sub-scene* information in an end-to-end framework. Finally, the use of super-pixels in [301] inhibit capturing detailed segmentation mask (and rather gives "blobby" output), and it is computationally less-tractable to use their approach for per-pixel optimization as information for each pixel would be required to be stored on disk.

8.5 2.5D Geometric Cues

We evaluate our approach on NYU Depth v2 dataset [393]. There are 795 training images and 654 test images in this dataset. Raw depth videos are also made available by [393]. We use the frames extracted from these videos to train our network for the task of surface normal estimation.

For training and testing we use the surface normals computed from the Kinect depth channel by Ladicky et al. [239] over the NYU trainval and test sets. As their surface normals are not available for the video frames in the training set, we compute normals (from depth data) using the approach of Fouhey et al. [123][5].

We ignore pixels where depth data is not available during training and testing. As shown in [102, 452] data augmentation during training can boost accuracy. We performed minimal data augmentation during training. We performed left-right flipping of the image and color augmentation, similar to [452], over the NYU trainval frames only; we did not perform augmentation over the video frames. This is much less augmentation than prior approaches [102, 452], and we believe we can get additional boost with further augmentation, e.g. by employing the suggestions in [63]. Note that the proposed pixel-level optimization also achieves comparable results training on only the 795 images in the training set of the NYUD2 dataset. This is due to the variability provided by pixels in the image as now each pixel act as a data point.

Figure 8.5 shows qualitative results from our approach. Notice that the back of the sofa in row 1 is correctly captured and the fine details of the desk and chair

[4]Per-class performance is available at http://host.robots.ox.ac.uk:8080/anonymous/PZH9WH.html.

[5]Fouhey et al. [123] used a first-order TGV denoising approach to compute normals from depth data which they used to train their model. We did not use the predicted normals from their approach.

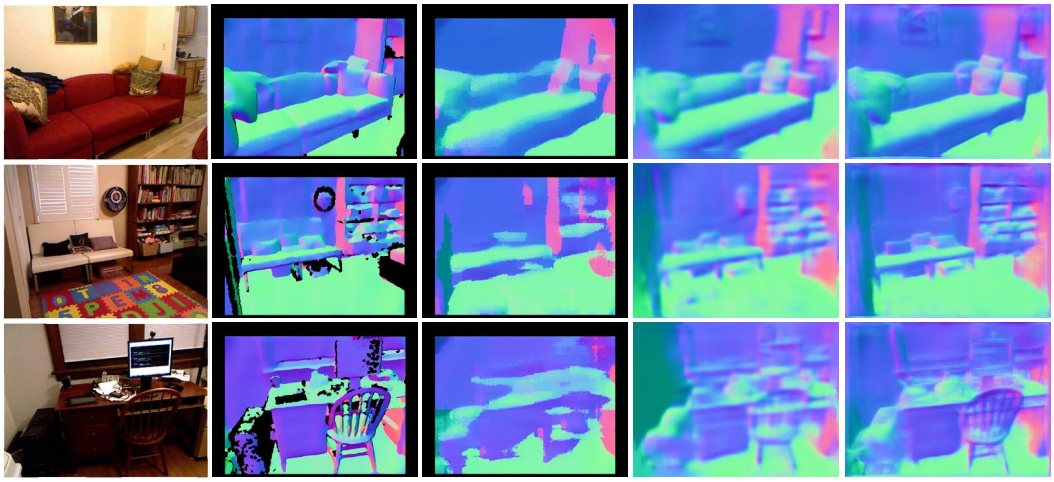


Figure 8.5: **Surface Normal Estimation:** We contrast our approach with prior art [102, 452] using a single 2D image for surface normal estimation. Our approach enables us to capture both global layout and local details in the scene. E.g., cushion on the sofa in the first and second row, and chair in the third row.

in row 3 are more visible in our approach. For quantitative evaluation we use the criteria introduced by Fouhey et al. [123] to compare our approach against prior work [102, 123, 125, 452]. Six statistics are computed over the angular error between the predicted normals and depth-based normals – **Mean**, **Median**, **RMSE**, **11.25**°, **22.5**°, and **30**° – using the normals of Ladicky et al. as ground truth [239]. The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better.

In this work, our focus is to capture more detailed surface normal information from the images. We, therefore, not only evaluate our approach on the entire global scene layout as in [102, 123, 125, 452], but we also introduce an evaluation over objects (chair, sofa, and bed) in indoor scene categories. First we show the performance of our approach on the entire global scene layout and compare it with [102, 123, 125, 452]. We then compare the surface normals for indoor scene furniture categories (chair, sofa, and bed) against [102, 452]. Finally, we perform an ablative analysis to justify our architecture design choices.

Global Scene Layout: Table 8.11 compares our approach with existing work. We present our results both with and without Manhattan-world rectification to fairly compare against previous approaches, as [123, 125, 452] use it and [102] do not. Similar to [123], we rectify our normals using the vanishing point estimates from Hedau et al. [173]. Interestingly, our approach performs worse with Manhattan-

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
Eigen-Fergus [102]	23.7	15.5	-	39.2	62.0	71.1
Fouhey et al. [123]	35.3	31.2	41.4	16.4	36.6	48.2
Ours	**19.8**	**12.0**	**28.2**	**47.9**	**70.0**	**77.8**
Manhattan World						
Wang et al. [452]	26.9	14.8	-	42.0	61.2	68.2
Fouhey et al. [125]	35.2	17.9	49.6	40.5	54.1	58.9
Fouhey et al. [123]	36.3	19.2	50.4	39.2	52.9	57.8
Ours	**23.9**	**11.9**	**35.9**	**48.4**	**66.0**	**72.7**

Table 8.11: **NYUv2 surface normal prediction: Global scene layout**. We compute six statistics over the angular error between the predicted normals and depth-based normals – **Mean**, **Median**, **RMSE**, **11.25°**, **22.5°**, and **30°**. The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better. Our approach significantly outperforms prior art.

world rectification (unlike Fouhey et al. [123]). Our network architecture predicts room layout automatically, and appears to be better than using vanishing point estimates. Though capturing scene layout was not our objective, our work out-performs previous approaches on all evaluation criteria.

Local Object Layout: The existing surface normal literature is focussed towards the scene layout. In this work, we stress the importance of fine details in the scene generally available around objects. We, therefore, evaluated the performance of our approach in the object regions by considering only those pixels which belong to a particular object. Here we show the performance on chair, sofa and bed. Table 8.12 shows comparison of our approach with Wang et al. [452] and Eigen and Fergus [102]. We achieve around **3-10%** better performance than prior art on all statistics for all the objects.

Ablative Analysis: We analyze how different sets of convolutional layers influence the performance of our approach. Table 8.13 shows some of our analysis. We chose a combination of layers from low, mid, and high parts of the VGG network. Clearly from the experiments, we need a combination of different low, mid, high layers to capture rich information present in the image.

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
Chair						
Wang et al. [452]	44.7	35.8	54.9	14.2	34.3	44.3
Eigen-Fergus [102]	38.2	32.5	46.3	14.4	34.9	46.6
Ours	**32.0**	**24.1**	**40.6**	**21.2**	**47.3**	**58.5**
Sofa						
Wang et al. [452]	36.0	27.6	45.4	21.6	42.6	53.1
Eigen-Fergus [102]	27.0	21.3	34.0	25.5	52.4	63.4
Ours	**20.9**	**15.9**	**27.0**	**34.8**	**66.1**	**77.7**
Bed						
Wang et al. [452]	28.6	18.5	38.7	34.0	56.4	65.3
Eigen-Fergus [102]	23.1	16.3	30.8	36.4	62.0	72.6
Ours	**19.6**	**13.4**	**26.9**	**43.5**	**69.3**	**79.3**

Table 8.12: **NYUv2 surface normal prediction: Local object layout**. We isolate local regions in the image and compute six statistics over the angular error between the predicted normals and depth-based normals – **Mean**, **Median**, **RMSE**, **11.25°**, **22.5°**, and **30°**. The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better. Our approach significantly outperforms prior art.

NYUDv2 test	Mean	Median	RMSE	11.25°	22.5°	30°
$\{1_1, 1_2\}$	44.4	42.7	49.3	4.1	16.5	28.2
$\{1_1, 1_2, 3_3\}$	30.2	24.7	37.7	23.1	46.2	58.4
$\{1_1, 1_2, 5_3\}$	22.6	15.3	30.5	39.1	63.4	73.1
$\{1_1, 1_2, 3_3, 5_3\}$	21.3	13.9	29.2	42.3	67.0	76.0
$\{1_2, 3_3, 5_3\}$	21.3	14.0	29.3	42.0	66.7	75.8
$\{1_2, 2_2, 3_3, 4_3, 5_3\}$	20.9	13.6	28.0	43.1	67.9	77.0
$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$	**19.8**	**12.0**	**28.2**	**47.9**	**70.0**	**77.8**

Table 8.13: **NYUv2 surface normal prediction: Ablative Analysis**. We analyze how different sets of convolutional layers influence the performance of our approach. We chose a combination of layers from low, mid, and high parts of the VGG network. Clearly from the experiments, we need a combination of different low, mid, high layers to capture rich information present in the image.

	ODS	OIS	AP
Human [12]	.800	.800	-
SE-Var [95]	.746	.767	.803
OEF [160]	.749	.772	.817
DeepNets [231]	.738	.759	.758
CSCNN [192]	.756	.775	.798
HED [478] (Updated version)	.788	.808	.840
UberNet (1-Task) [233]	.791	.809	**.849**
PixelNet (Uniform)	.767	.786	.800
PixelNet (Biased)	**.795**	**.811**	.830

Table 8.14: **Evaluation on BSDS [12]:** Our approach performs better than previous approaches *specifically* trained for edge detection. Here, we show two results using our architecture: a. Uniform; and b. Biased. For the former, we randomly sample positive and negative examples while we do a biased sampling of 50% positives (from a total 10% positives in edge dataset) and 50% negatives. As shown, biased sampling improves performance for edge detection. Finally, we achieve F-score of 0.8 which is same as humans.

8.6 2.5D Boundary Cues

Dataset: The standard dataset for edge detection is BSDS-500 [12], which consists of 200 training, 100 validation, and 200 testing images. Each image is annotated by $\sim$5 humans to mark out the contours. We use the same augmented data (rotation, flipping, totaling 9600 images without resizing) used to train the state-of-the-art Holistically-nested edge detector (HED) [478]. We report numbers on the testing images. During training, we follow HED and only use positive labels where a consensus (≥ 3 out of 5) of humans agreed.

Training: We use the same baseline network and training strategy defined earlier in Section 8.4. A sigmoid cross-entropy loss is used to determine the whether a pixel is belonging to an edge or not. Due to the highly skewed class distribution, we also normalized the gradients for positives and negatives in each batch (as in [478]). In case of skewed class label distribution, sampling offers the flexibility to let the model focus more on the rare classes.

Results: Table 8.16 shows the performance of PixelNet for edge detection. The last 2 rows in Table 8.16 contrast the performance between uniform and biased sampling. Clearly biased sampling toward positives can help the performance. Qualitatively, we find our network tends to have better results for semantic-contours (around an object), particularly after including *conv-7* features. Figure 8.7 shows some qualitative results comparing our network with the HED model. Interestingly, our model explicitly removed the edges inside the *zebra*, but when the model cannot recognize

	ODS	OIS	AP
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ *Uniform*	.767	.786	.800
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (25%)	.792	.808	.826
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (50%)	.791	.807	.823
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (75%)	.790	.805	.818

Table 8.15: Comparison of different sampling strategies during training. *Top row:* Uniform pixel sampling. *Bottom rows:* Biased sampling of positive examples. We sample a fixed percentage of positive examples (25%,50% and 75%) for each image. Notice a significance difference in performance.

it (its head is out of the picture), it still marks the edges on the black-and-white stripes. Our model appears to be making use of more higher-level information than past work on edge detection.

8.6.1 More Analysis

Baseline. We use the same baseline network that was defined for semantic segmentation, only making use of pre-trained *conv* layers. A sigmoid cross-entropy loss is used to determine the whether a pixel is belonging to an edge or not. Due to the highly skewed class distribution, we also normalized the gradients for positives and negatives in each batch (as in [478]).

Analysis-1: Sampling. Whereas uniform sampling sufficed for semantic segmentation [267], we found the extreme rarity of positive pixels in edge detection required focused sampling of positives. We compare different strategies for sampling a fixed number (2000 pixels per image) training examples in Table 8.15. Two obvious approaches are uniform and balanced sampling with an equal ratio of positives and negatives (shown to be useful for object detection [141]). We tried ratios of 0.25, 0.5 and 0.75, and found that balancing consistently improved performance[6].

Analysis-2: Adding *conv-7*. We previously found that adding features from higher layers is helpful for semantic segmentation. Are such high-level features also helpful for edge detection, generally regarded as a low-level task? To answer this question, we again concatenated *conv-7* features with other *conv* layers $\{ 1_2, 2_2, 3_3, 4_3, 5_3 \}$. Please refer to the results at Table 8.16, using the second sampling strategy. We find it still helps performance a bit, but not as significantly for semantic segmentation (clearly a high-level task). Our final results as a single output classifier are very competitive to the state-of-the-art.

Qualitatively, we find our network tends to have better results for semantic-contours (around an object), particularly after including *conv-7* features. Figure 8.7

[6]Note that simple class balancing [478] in each batch is already used, so the performance gain is *unlikely* from label re-balancing.

	ODS	OIS	AP
Human [12]	.800	.800	-
Canny	.600	.640	.580
Felz-Hutt [114]	.610	.640	.560
gPb-owt-ucm [12]	.726	.757	.696
Sketch Tokens [252]	.727	.746	.780
SCG [474]	.739	.758	.773
PMI [198]	.740	.770	.780
SE-Var [95]	.746	.767	.803
OEF [160]	.749	.772	.817
DeepNets [231]	.738	.759	.758
CSCNN [192]	.756	.775	.798
HED [478]	.782	.804	.833
HED [478] (Updated version)	.790	.808	.811
HED merging [478] (Updated version)	.788	.808	**.840**
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (50%)	.791	.807	.823
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (50%)	**.795**	**.811**	.830
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (25%)-$(0.5\times,1.0\times)$	.792	.808	.826
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (25%)-$(0.5\times,1.0\times)$	**.795**	**.811**	.825
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3\}$ (50%)-$(0.5\times,1.0\times)$	.791	.807	.823
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (50%)-$(0.5\times,1.0\times)$	**.795**	**.811**	.830
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (25%)-$(1.0\times)$	.792	.808	.837
conv-$\{1_2, 2_2, 3_3, 4_3, 5_3, 7\}$ (50%)-$(1.0\times)$	.791	.803	**.840**

Table 8.16: Evaluation on BSDS [12]. Our approach performs better than previous approaches *specifically* trained for edge detection.

shows some qualitative results comparing our network with the HED model. Interestingly, our model explicitly removed the edges inside the *zebra*, but when the model cannot recognize it (its head is out of the picture), it still marks the edges on the black-and-white stripes. Our model appears to be making use of much higher-level information than past work on edge detection.

8.7 3D Objects

Given a selected image region depicting an object of interest, along with a corresponding predicted surface normal map (Section 8.5), we seek to retrieve a 3D model from a large object CAD library matching the style and pose of the depicted object. This is a hard task given the large number of library models and possible

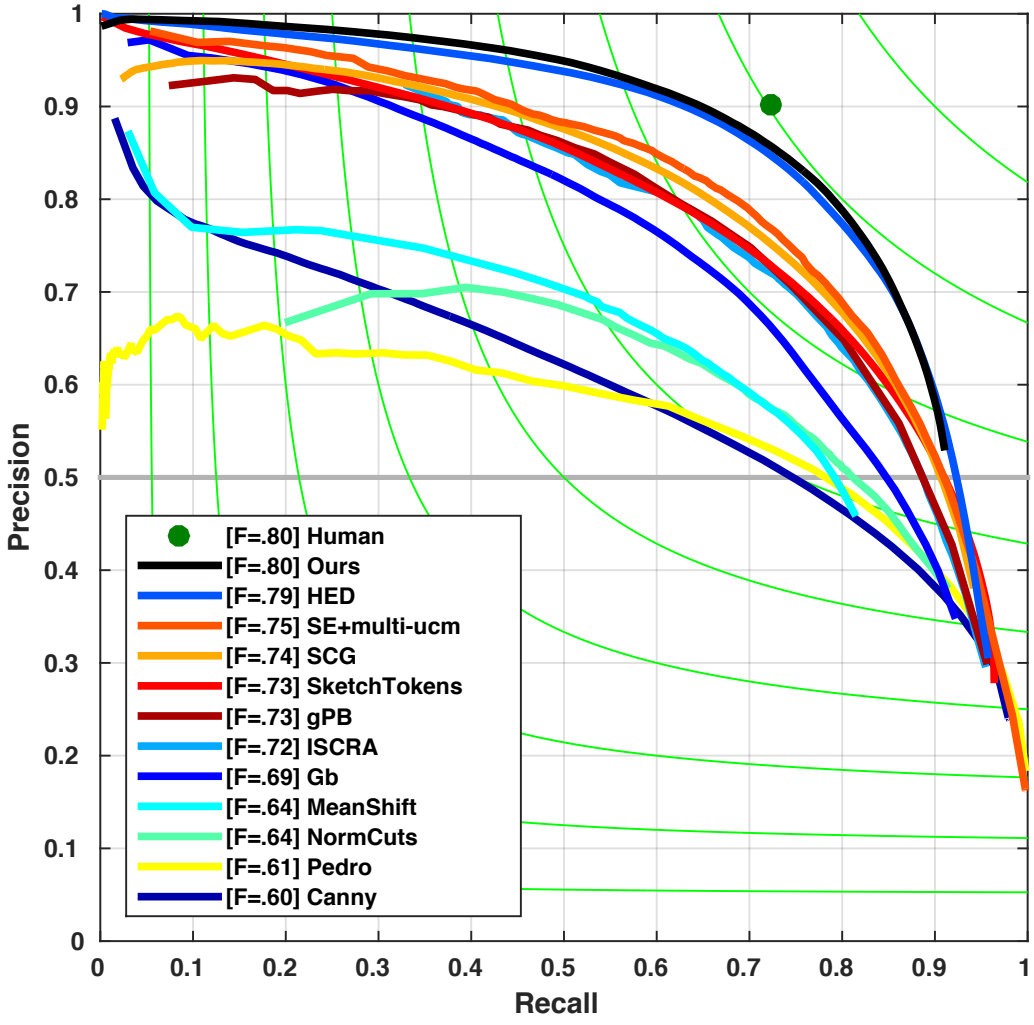


Figure 8.6: Results on BSDS [12]. While our curve is mostly overlapping with HED, our detector focuses on more high-level semantic edges. See qualitative results in Fig.8.7.

viewpoints of the object. While prior work has performed retrieval by matching the image to rendered views of the CAD models [13], we seek to leverage both the image appearance information and the predicted surface normals.

We first propose a two-stream network to estimate the object pose. This two-stream network takes as input both the image appearance I and predicted surface normals $n(I)$, illustrated in Figure 8.9 (left). Each stream of the two stream network is similar in architecture to CaffeNet [238] upto pool5 layer. We also initialize both the streams using pre-trained ImageNet network.

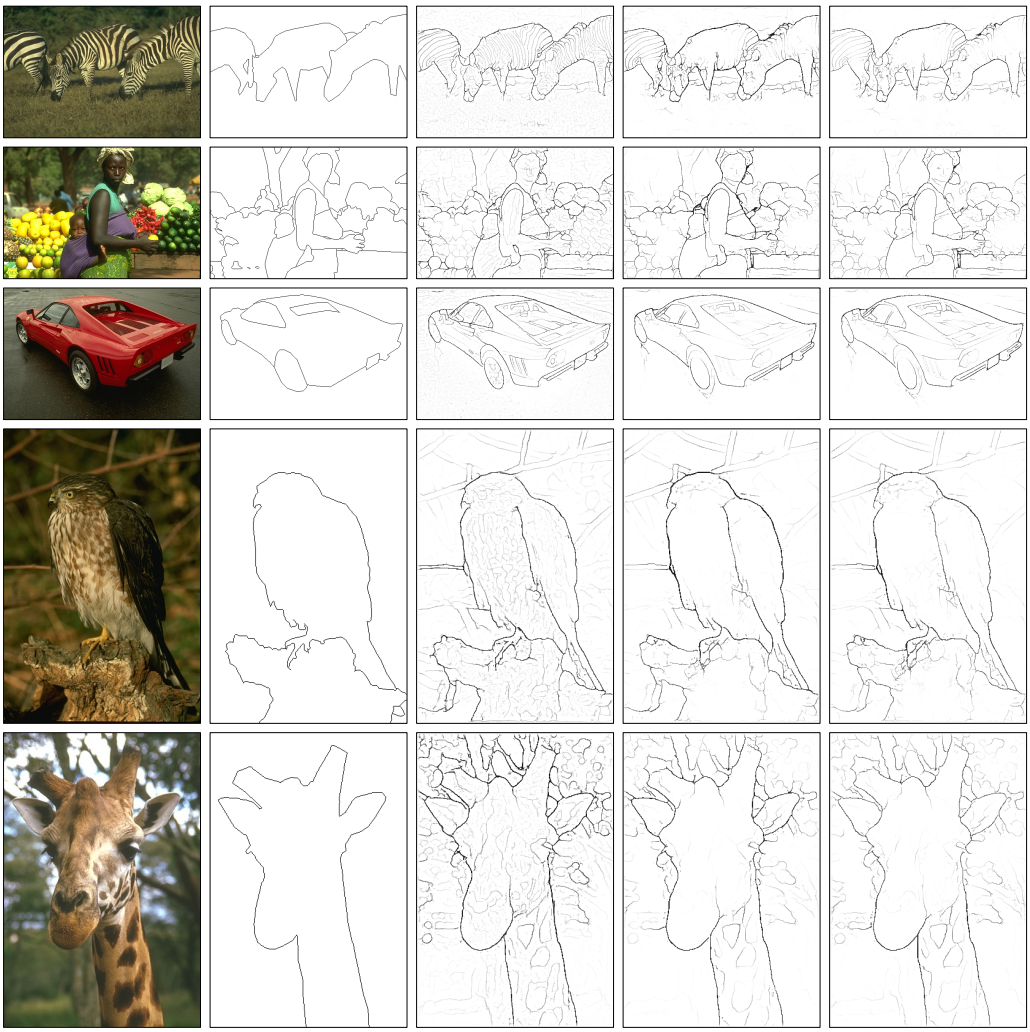


Figure 8.7: Qualitative results for edge detection. Notice that our approach generates more semantic edges for *zebra*, *eagle*, and *giraffe* compared to HED [478]. Best viewed in the electronic version.

Note that for surface normals there is no corresponding pre-trained CNN. Although the CaffeNet model has been trained on images, we have found experimentally (c.f. Section 8.7.1) that it can also represent well surface normals. As the surface normals are not in the same range as natural images, we found that it is important as a pre-processing step to transform them to be in the expected range. The surface normal values range from $[-1, 1]$. We map these scores of surface normals to $[0, 255]$ to bring them in same range as natural images. A mean pixel subtraction is done

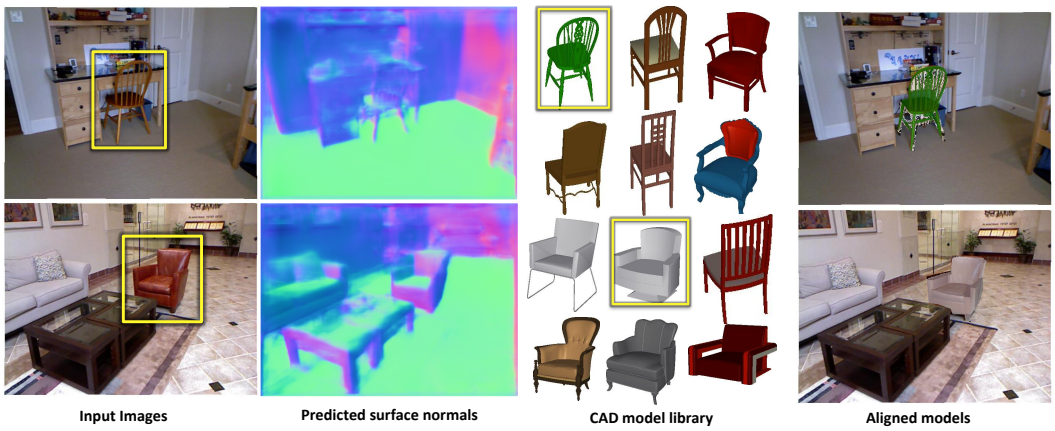


Figure 8.8: **3D Objects:** Given a single 2D image, we predict surface-normals that capture detailed object surfaces. We use the image and predicted surface-normals to retrieve a 3D model from a large library of object CAD models.

before the image is feed-forward to the network. The mean values for n_x, n_y, and n_z are computed using the 381 images in train set of NYUD2.

While one could use the pre-trained networks directly for retrieval, such a representation has not been optimized for retrieving CAD models with similar pose and style. We seek to optimize a network to predict pose and style given training data. For learning pose, we leverage the fact that the CAD models are registered to a canonical view so that viewpoint and surface normals are known for rendered views. We generate a training set of sampled rendered views and surface normal maps $\{(I_i, \hat{n}_i)\}_{i=1}^N$ for viewing angles $\{\phi_i\}_{i=1}^N$ for all CAD models in the library. We generate surface normals for each pixel by ray casting to the model faces, which allows us to compute view-based surface normals $\hat{n}$.

To model pose, we discretize the viewing angles ϕ and cast the problem as one of classifying into one of the discrete poses. We pass the concatenated CaffeNet "pool5" features $\bar{c}(I, \hat{n})$ through a sequence of two fully-connected layers, followed by a softmax layer to yield pose predictions $g(I, \hat{n}; \Theta)$ for model parameters Θ. We optimize a softmax loss over model parameters Θ:

$$\min_{\Theta} - \sum_{i=1}^{N} \phi_i^T \log(g(I_i, \hat{n}_i; \Theta)). \tag{8.1}$$

Note that during training, we back propagate the loss through all the layers of CaffeNet as well. Given a trained pose predictor, at test time we pass in image I and predicted surface normals $n(I)$ to yield pose predictions $g(I, n(I); \Theta)$ from the last fully connected layer. We can also run our network given RGB-D images, where surface normals are derived from the depth channel. We show pose-prediction results

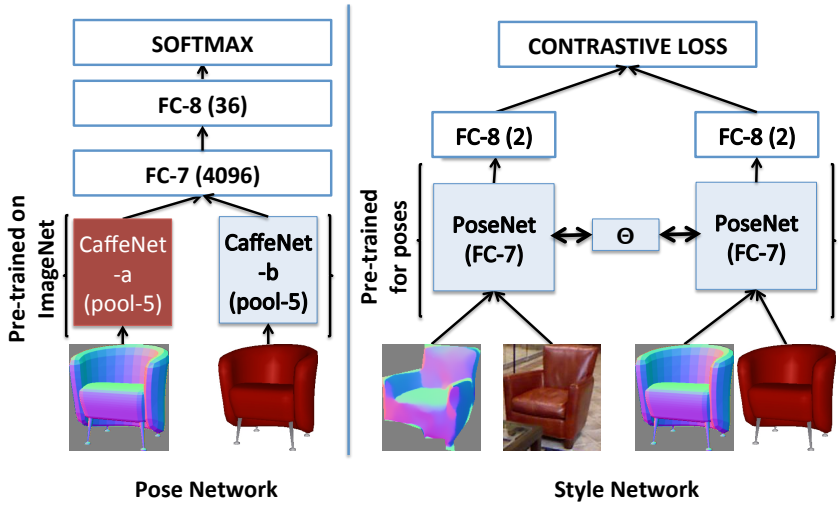


Figure 8.9: Networks for predicting pose (left) and style (right). Our pose network is trained on a set of rendered CAD views and extracted surface normal pairs. During prediction, an image and its predicted surface normals are used to predict the object pose. For the style network, we train on hand-aligned natural image and CAD rendered view pairs. We initialize the style network with the network trained for poses. See text for more details.

for both types of inputs in Section 8.7.1.

Note that a similar network for pose prediction has been proposed for RGB-D input images [159]. There, they train a network from scratch using normals from CAD for training and query using Kinect-based surface normals during prediction. We differ in our use of the pre-trained CaffeNet to represent surface normals and our two-stream network incorporating both surface normal and appearance information. We found that due to the differences in appearance of natural images and rendered views of CAD models, simply concatenating the pool5 CaffeNet features hurt performance. We augmented the data similar to [405] by compositing our rendered views over backgrounds sampled from natural images during training, which improved performance.

From two-stream pose to siamese style network. While the output of the last fully-connected layer used for pose prediction can be used for retrieval, it has not yet been optimized for style. Inspired by [32], we seek to model style given a training set of hand-aligned similar and dissimilar CAD model-image pairs. Towards this goal, we extend our two-stream pose network to siamese two-stream network for this task, illustrated in Figure 8.9 (right). Specifically, let f be the response of the last fully-connected layer of the pose network above. Given similar image-model pairs (f_p, f_q) and dissimilar pairs (f_q, f_n), we seek to optimize the contrastive loss:

$$L(\Theta) = \sum_{(q,p)} L_p(f_q, f_p) + \sum_{(q,n)} L_n(f_q, f_n). \tag{8.2}$$

We use the losses $L_p(f_q, f_p) = ||f_q - f_p||_2$ and $L_n(f_q, f_n) = \max(m - ||f_q - f_n||_2, 0)$, where $m = 1$ is a parameter specifying the margin. As in [32], we optimize the above objective via a Siamese network. Note that we optimize over pose and style, while [32] optimizes over object class and style for the task of product image retrieval.

For optimization, we apply mini-batch SGD in training using the caffe framework. We followed the standard techniques to train a CaffeNet-like architecture, and back-propagate through all layers. The procedure for training and testing are described in the respective experiment section.

8.7.1 Pose Estimation

We evaluated our approach to estimate the pose of a given object. We trained the pose network using CAD models from Princeton ModelNet [470] as training data, and used 1260 models for chair, 526 for sofa, and 196 for bed. For each model, we rendered 144 different views corresponding to 4 elevation and 36 azimuth angles. We designed the network to predict one of the 36 azimuth angles, which we treated as a 36-class classification problem. Note that we trained separate pose networks for the chair, sofa, and bed classes. At test time, we forward propagated the selected region from the image, along with its predicted surface normals, and selected the angle with maximum prediction score. We evaluated our approach using the annotations from Guo and Hoiem [156] where they manually annotated the NYUD2 dataset with aligned 3D CAD models for the categories of interest.

Comparison on NYUD2 val set: Figure 8.10 shows a quantitative evaluation of our approach on the NYUD2 val set. Using the criteria introduced in Gupta et al [159], we plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ (higher is better). We compare our approach with Gupta et al [159] who showed results of pose estimation on the NYUD2 val set for objects with at least 50% valid depth pixels. Note that we trained our skip-network for surface normals using the 381 images of the NYUD2 train set. We clearly out-perform the baseline using RGB-only and RGB-D for chairs and sofas.

Comparison on NYUD2 test set: Unfortunately, we cannot directly compare the approaches of [159] and [325] for pose estimation. While Gupta et al. [159] reported performance on the NYUD2 val set, Papon and Schoeler [325] reported performance on the test set. We evaluated our approach on the val and test sets separately to directly compare against both methods. Figure 8.11 shows the comparison

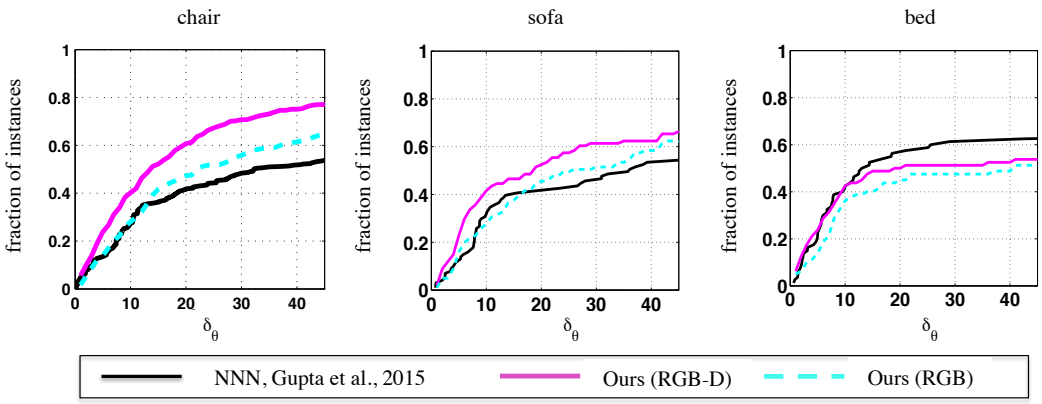


Figure 8.10: Pose prediction on val set. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. Similar to [159] we consider only those objects which have valid depth pixels for more than 50%.

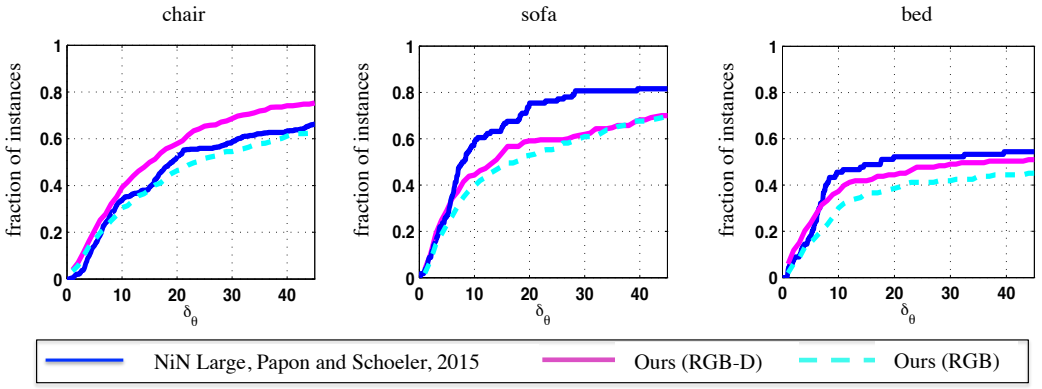


Figure 8.11: Pose prediction on test set. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. Similar to [159] we consider only those objects which have valid depth pixels for more than 50%.

of our approach against Papon and Schoeler [325] on the test set (we trained using the NYUD2 trainval set) and shows that our approach is competitive for both RGB-D and RGB.

Varying predicted surface normals: We analyze how different surface normal prediction algorithms affect the accuracy of predicting object pose. Since no real-world data was used for training our pose estimation network (we only used CAD model rendered views), we can perform this experiment without any bias with re-

spect to the surface normal prediction algorithm. Figure 8.12 shows a comparison of our approach, along with Eigen and Fergus [102] and Wang et al. [452]. Notice that better surface normal prediction results in better object pose predictions.

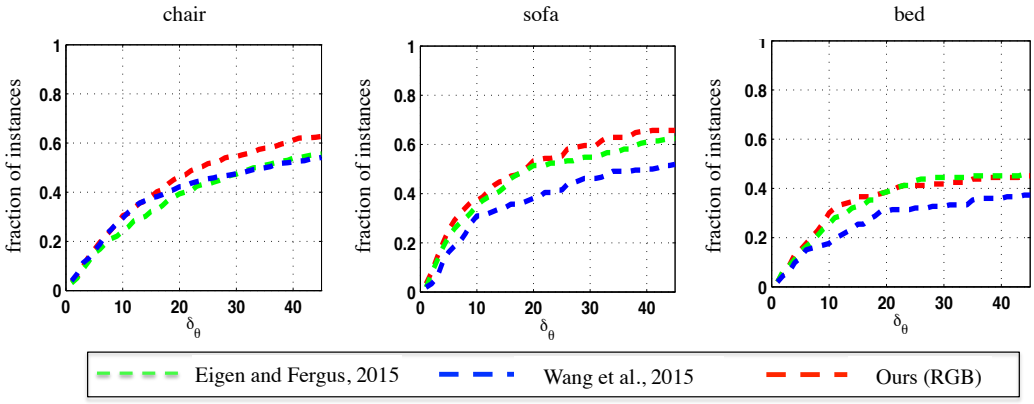


Figure 8.12: Pose prediction for different surface normal predictions. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. Similar to [159] we consider only those objects which have valid depth pixels for more than 50%.

Removing depth constraint in evaluation: So far we have ignored test examples having less than 50% valid depth pixels since prior approaches based on RGB-D data require valid depth for object pose prediction. The predicted normals gives us an added advantage to consider all examples irrespective of available depth information. In this experiment we compare our approach for pose estimation without any depth-based criteria. Figure 8.13 shows the performance of different surface normal approaches on NYUD2 test set for **all test examples**.

Comparison with nearest neighbors: We compare our approach with nearest neighbors using surface normals and CaffeNet pool-5 features to retrieve a 3D model that is similar in pose and style. When using surface normals, we slide the CAD models on the given bounding box of the image to determine the correct location and scale for the model. The CAD model rendered views are resized such that the maximum dimension is 40 pixels. We tried two approaches for scoring the windows for the sliding-window approach - 1) compute dot product; 2) compute the angular error between the two, and then compute the percentage of pixels within 30° angular error (we call this criteria 'Geom'). To penalize smaller windows we compute the product of the scores and overlap (IoU) between the window and original box. This ensures there is no bias towards smaller windows in the sliding-window approach. Finally, we prune similar CAD models within a 20° azimuthal angle.

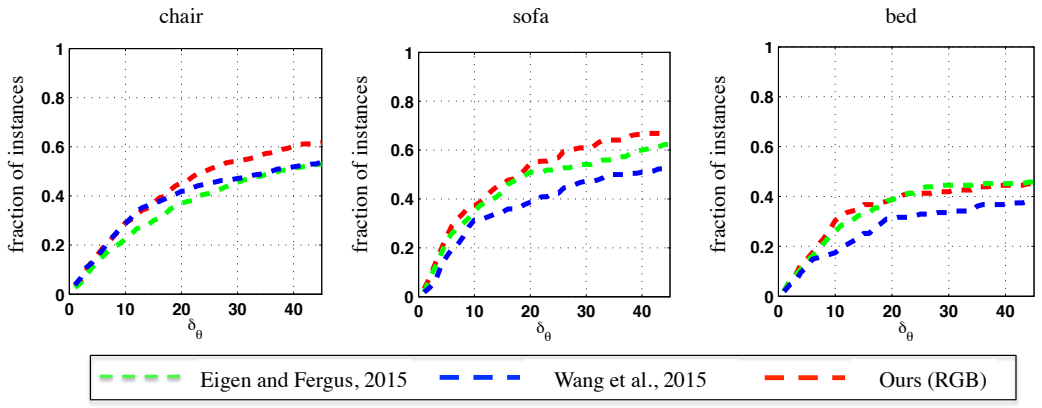


Figure 8.13: Pose prediction for all images irrespective of depth constraint. We plot the fraction of instances with predicted pose angular error less than δ_θ as a function of δ_θ. In this analysis, we consider all the objects irrespective of valid depth data.

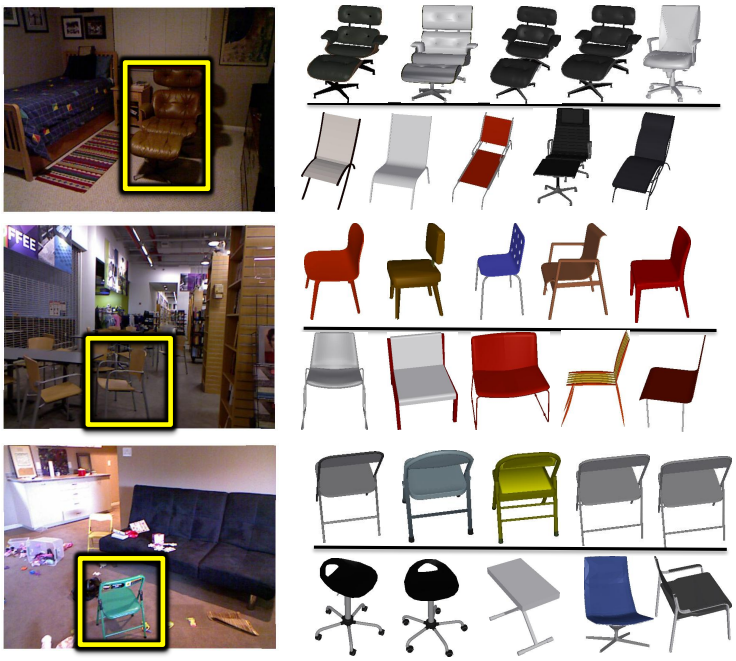


Figure 8.14: For each example, the top row shows CAD models retrieved using fc-7 of Pose Network and the bottom row shows the result of nearest-neighbor retrieval using predicted surface normals.

NYUD2 test	Dot-Product (1 NN)	Dot-Product (K-NN)	Geom (1 NN)	Geom (K-NN)
Random	0.13	0.13	0.13	0.13
Normals (ours)	0.21	0.22	0.31	0.30
Normals (depth)	0.33	0.31	0.41	0.44
App. (pool-5)	0.26	0.28	0.26	0.28
ours+pool-5	0.25	0.28	0.29	0.32
depth+pool-5	0.30	0.33	0.36	0.38

Table 8.17: Area under the fraction of instances versus angular error δ_θ curve. Similar to [159, 325], we consider only those objects which have valid depth pixels for more than 50%. For K-NN we used K = 35. Note that for App. (pool-5), we did not use the 'Geom' criteria but copied the scores of dot product in it. Our PoseNet (RGB) and (RGB-D) achieves 0.43 and 0.55 respectively, which is higher than the proposed nearest neighbor approaches.

To capture appearance information during retrieval, we used the CaffeNet pool-5 features for both CAD models and the given bounding box. We computed the cosine distance between pool-5 features. Both scores from appearance and surface normals were combined into a final score by weighted averaging.

We evaluated on the chairs in the test set. In this experiment, we only considered chairs having more than 50% of valid depth pixels. We report area under the pose prediction curve in Table 8.17. Notice that the 'Geom' criteria outperforms dot product. Also, combining appearance information boosts the performance of predicted normals but hurts the performance of normals from depth. Note that our **PoseNet** trained on just **RGB** is comparable to the results obtained using nearest neighbors for RGB-D.

8.7.2 Style Estimation

We now use our style network to determine the style of objects. To reduce the search space, we use this network to re-rank the top-N output of the CAD models retrieved using the fc-7 feature of the pose network. We evaluate our style network using chairs as chairs span a large variety of styles [13]. To train the model we hand-labeled images in the NYUD2 training set with models having very similar style. To assist with the labeling we used our pose network to retrieve similar CAD models over the NYU training set. For each example we looked at the top-30 retrieved CAD models and manually labeled if a particular CAD model is similar to the input example or not. We used these labels to train our style network using the contrastive loss. Figure 8.15 shows qualitative examples of our re-ranking via the style network.

Figure 8.15: Style re-ranking. For each example the top row shows the top-5 CAD models obtained using our Style Network and the bottom row shows the original retrievals using the Pose Network.

Note that the network is able to boost the ranking of similar examples, e.g., the chairs having wheels in the first and last row have different styles in the initial retrieved examples of the pose network. With the re-ranking, we are able to see chairs with wheels consistently.

Part IV

Continual and Streaming Learning

Chapter 9

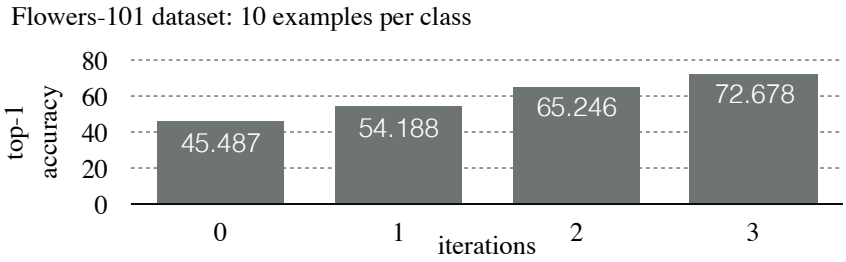
Streaming Learning with a Few Examples

The different formulation proposed in this thesis follows exemplar and test-time training. This unique combination allows to continually learn the audio-visual world in a streaming manner. We extend our work on continual learning of the audio-visual world to learning visual-recognition tasks in a continual and streaming manner. We explore semi-supervised learning of deep representations given a few labeled examples of a task and a (potentially) infinite stream of unlabeled examples. In this setting, classic approaches that attempt to pseudo-label the entire unlabeled stream may take an exorbitant amount of time. Our approach continually evolves task-specific representations by constructing a schedule of learning updates that iterates between pre-training on novel segments of the stream and fine-tuning on the small, fixed labeled dataset. We take inspiration from developmental psychology that explores how children learn.

9.1 Learning in Children vs. Machine

Children can learn about a concept (apple, banana, etc) from a few examples whereas our machines cannot. We, however, forget that the mind of a child is continually evolving from the world around them, even when there is no one teaching them [119]. However, visual recognition models for a machine are trained in a one-stage setting, often from a fixed data distribution. In this work, we take the inspiration from the continual cognitive development of a child (Fig. 9.1). We introduce a simple approach that continually improves a task-specific representation learning on a continuous stream of infinite data without any extra supervision or knowledge. Crucially, our approach learns a meaningful representation for a task given a few labeled examples in a few days of compute. The representation learned using our approach is continually improving – this means the task representation

(a) Children continually improve their knowledge about a concept.



(b) Machines can also improve their knowledge about a concept in this iterative manner.

Figure 9.1: (**a**) We take inspiration from developmental psychology that explores how children learn. Children maybe exposed to a concept (say flowers), play with other things in their environment, and eventually return to the lesson at hand. By interleaving periods of self-supervised play and teacher-supervised learning, they can continually evolve their representations about the world. (b) We take inspiration from this model of continual learning and show how machines can continually learn better representations for various visual understanding tasks. Here, we show an illustrative fine-grained task of classifying flowers [313]. *Iteration*-0 shows the performance of a convolutional neural network trained from scratch using only 10 examples per class. We use millions of unlabeled images as a proxy of exploring the world. We continually improve the performance in each iteration. At the end of the third iteration, the performance on the task improved from 45.49% to 72.68% top-1 accuracy. We did not use any other labeled data or task-specific knowledge to improve performance.

learned in a week will outperform what we have now.

One-Stage vs. Continual Evolution: Prior work focuses on one-stage approaches for learning representations for a task, typically via more labeled data [253, 362, 501], higher capacity parametric models [170, 184, 238, 395], finding better architectures [55, 413, 518], or adding domain-specific knowledge [340, 452]. Recently many approaches have leveraged unlabeled images to learn a good representation using

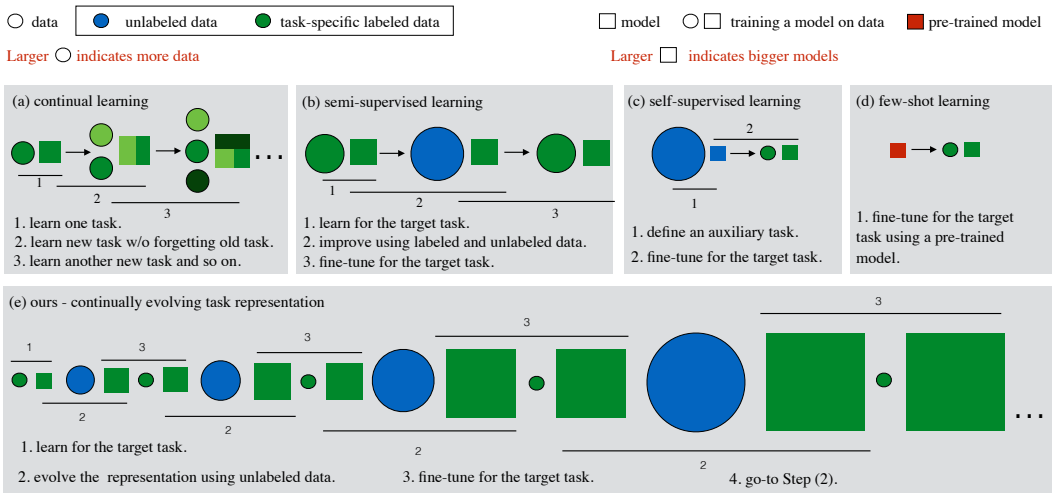


Figure 9.2: Our work combines insights from (a) **continual learning** and (b) **semi-supervised learning**. The goal in continual learning is to continually learn new tasks in a supervised manner without forgetting previous tasks . In this work, we continually improve the representation for a fixed task using an infinite stream of unlabeled data. This is closely related to semi-supervised learning where better representations are learned using large amounts of unlabeled data in a one-shot learning manner. However, semi-supervised learning approaches also consider a large amount of labeled data for the task of interest (shown by big green circle) in conjunction with unlabeled data. In this work, we make no such assumption and can even learn when we have few labeled examples (shown by small green circle). Our approach is also inspired from (c) **self-supervised learning** that uses unlabeled images for learning a generic representation in a one-time learning that can be used for downstream tasks. We, however, use unlabeled images to learn a task-specific representation in a continual manner. Importantly, the underlying representation in our work evolves (both expands and improves) over time. Finally, we are inspired from (d) **few-shot learning** approaches that aims to learn a representation from a few-labeled examples. However, these approaches typically use a model trained on a large amount of labeled data for a related task and then fine-tune for the target task. In this work, we use only few-labeled examples to learn a model from scratch and improve the performance over time.

auxiliary tasks [93, 138, 297, 453, 493]. These approaches learn a one-stage generic representation that can be used as an initialization for other tasks (Fig. 9.2-(c)). This is contrary to human learning [314] where the recognition abilities evolve over years from the perception of simple shapes in infancy to the ability to recognize individual objects through childhood into adolescence. Finally, semi-supervised approaches [344, 481] have also made the use of unlabeled images in learning a

better representation (Fig. 9.2-(b)). While an inspiration for our own work on continual evolving representation, these approaches also learn a one-stage representation (notable exception [475]) and make two solid assumptions: (1) availability of a good initial model to obtain pseudo-labels on the unlabeled images; and (2) a procedure to clean noisy generated labels. Notably [475, 481] shows that they could improve top-1 accuracy on Imagenet [362] by 2% over prior state-of-the-art with a good initial model and selection procedures. In this work, we make no such assumption and demonstrate that we can continually learn a better task-specific representation via Self-Training [374, 486], even when starting with a few labeled examples.

Our Observation: We observe that one can indeed start from a few labeled examples for a task given we have a (potentially infinite) stream of unlabeled examples. We can continually improve a task-representation by constructing a schedule of learning updates that iterates between pre-training on segments of the unlabeled stream and fine-tuning on the small labeled dataset (Fig. 9.2-(e)). We progressively learn more accurate pseudo-labels as the stream is processed. This observation implies that we can learn better mappings using diverse unlabeled examples without any extra supervision or knowledge of the task.

Training deeper models with few examples: An important consequence of our work is that we can now train very deep models from scratch using a few labeled examples. The large capacity models are often prone to overfitting in a low-data regime and usually under-perform [307]. For e.g. a ResNet-50 model [170] trained from scratch for a 200-way fine-grained bird classification [463] using 30 examples-per-class overfits and yields 21.7% top-1 accuracy on a held-out validation set. In a single iteration of our approach, the same model gets 51.5% top-1 accuracy in a day. This is crucial specially when there is a possibility of a better task representations but we could not explore them because of the lack of labeled data. It is also important to mention that because of the lack of labeled data for various tasks, many computer-vision approaches have been restricted to use the models designed for image classification specifically. This observation also implies that we can continually increase the capacity of models over time and achieve even better performance of the required task with the larger capacity models. The continual evolution of fine-grained bird classification improved performance to 66.1% top-1 accuracy in a few more days.

Our Contributions: (1) To the best of our knowledge, we are the first to introduce an approach that continually learn better representations for a task using millions of unlabeled images without any additional assumption and auxiliary information; (2) Our approach enables to train very high capacity models on a few-labeled example per class without any extra knowledge; (3) Finally, we demonstrate our approach on a variety of tasks. The performance of our model improved by 27% top-1 accuracy for Flowers-102 [313] and 22% top-1 accuracy for Birds-200 [463] in three iterations of our approach. Our insights are also applicable to

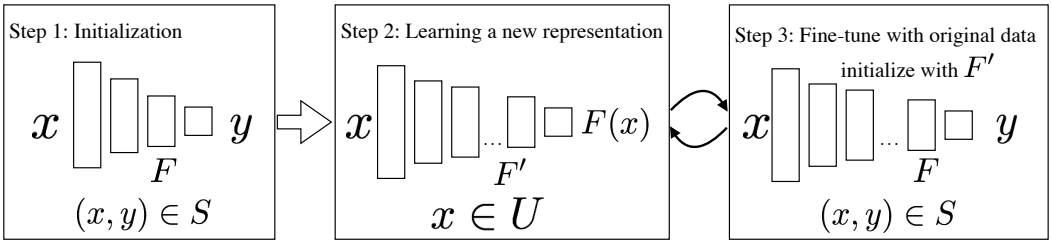


Figure 9.3: **Our Approach:** There are three important steps of our approach. (a) **Step 1:** Initialization– we learn an initial mapping F on $(x, y) \in S$; (b) **Step 2:** Learning a new representation– We use F to learn a new model F' from scratch on sample $x \in U$; and (c) finally, **Step 3:** Fine-tune with original data – we fine-tune F' on S. This becomes our new F. We continually cycle between Step-2 and Step-3. The capacity of model F' increases with every cycle.

medical-image classification, crop-disease classification, and satellite-image classification where we don't have a large amount of labeled and unlabeled dataset. Our insights also hold for pixel-level prediction problems. We improve surface normal estimation on NYU-v2 depth dataset [393] and semantic segmentation on PASCAL VOC-2012 [108] by $3-7\%$. Note all these models are initialized from scratch. We are continually improving the performance of these models with continuous streams of unlabeled data.

9.2 Related Work

Our continually evolving task-specific learning is inspired from the continually improving and expanding human mind [7,8]. Learning a task-specific representation is not new in computer vision. There are substantial efforts in learning a good representation for an individual task using domain-specific knowledge, be it object detection [142], segmentation [267], human keypoints [57], 3D human models [213], depth and surface normal estimation [23, 102], or boundary detection [478]. This has partly been inspired by our understanding of how we perceive the world [190]. Recognition abilities in the human brain is also task dependent and operates at different locations [295]. For e.g., action-related properties in the ventral stream [278], real world scenes and spatial layout/places in PPA and early visual cortex [107,236, 274, 327], shape analysis in LOC [219, 279], and face recognition in the FFA [220] and OFA [166,336,377]. With the onset of data-driven learning, powered by neural networks [242], there have been efforts [93,250,298,453] to learn a generic representation from data that can be used for any task. In this work, we use data to improve task-specific representations without any task-specific or auxiliary knowledge. We demonstrate that we can achieve large performance boosts by continually leveraging unlabeled data without any expertise.

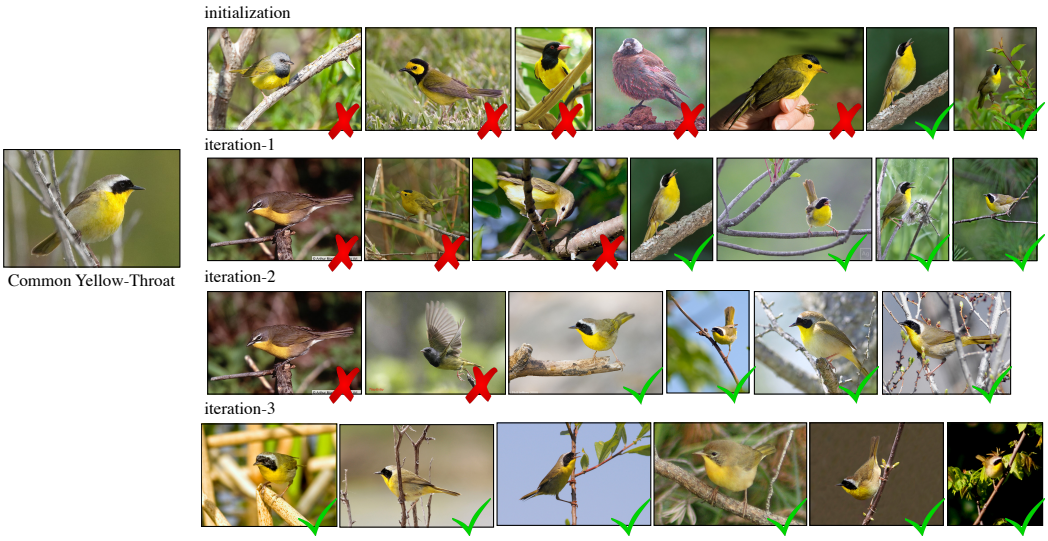


Figure 9.4: **Continual Improvement in Tasks:** We qualitatively show improvement in recognizing a common yellow-throat (shown in left from CUB-200 dataset [463]). At **initialization**, the trained model confuses common yellow-throat with hooded oriole, hooded warbler, wilson rbler, yellow-breasted chat, and other similar looking birds. We get rid of false-positives with every **iteration**. At the the end of the third iteration, there are no more false-positives.

Continual and Iterated Learning: Our work shares inspiration with a large body of work on continual and lifelong learning [394,421,422]. A major goal in this line of work [116,117,349,351,447] has been to continually learn a good representation over a sequence of tasks (Fig. 9.2-(a)) that can be used to adapt to a new task with few-labeled examples without forgetting the earlier tasks [60, 250]. Our goal, however, is also to learn a task-specific representation from a few labeled examples and no extra knowledge. Our work shares insights with iterated learning [229, 230] that suggests evolution of language and emerging compositional structure of human language through the successive re-learning. Recent work [271, 272] has also used these insights in countering language drift and interactive language learning. In this work, we restrict ourselves to visual recognition tasks and show that we continually improve task-specific representations in an iterated learning fashion using infinite stream of unlabeled data.

Learning from Unlabeled or Weakly-Labeled Data: The power of large corpus of unlabeled or weakly-labeled data has been widely explored in semi-supervised learning [10, 62, 195, 310, 343, 344, 346, 496, 514], self-supervised learning [93, 138, 493], or weakly-supervised learning [199, 206, 406, 508]. Closely related to ours is the work on semi-supervised learning [475, 481, 517] that use self-training [96, 459] or pseudo-labels [10, 195, 245] on unlabeled data to learn a better representation

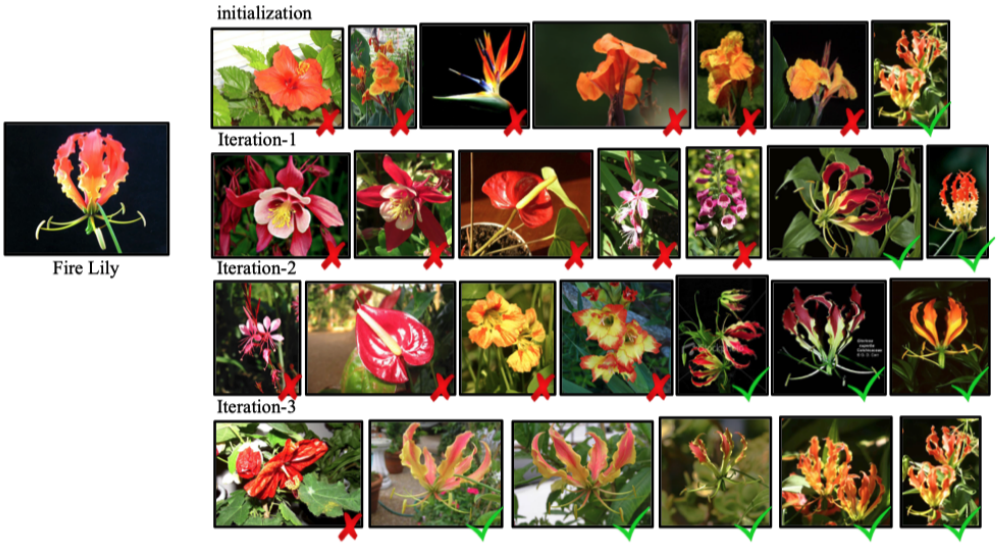


Figure 9.5: **Continual Improvement for Fire Lily:** We qualitatively show improvement in recognizing a fire lily (shown in left from flowers-102 dataset [313]). At **initialization**, the trained model confuses fire lily with tiger lily, bird of paradise, peruvian lily, canna lily, foxglove, and etc. With more iterations, the false positives become fewer.

(Fig. 9.2-(b)). However, these approaches learn a one-time representation for the task. It is also worth mentioning that these approaches generally use a large corpus of labeled examples (e.g., a million labeled images from Imagenet) alongside the unlabeled images. In this work, the task-representations are continually evolving (improving and expanding). We also demonstrate that we can achieve large performance boosts with a few labeled examples (as few as 10 labeled examples per class) with the continuous stream of unlabeled images.

We show that our approach can learn better representations for a wide variety of tasks including image classification, surface normal estimation, and semantic segmentation, without any task-specific knowledge. Our approach also works for medical-image classification, satellite-image classification, and crop-disease classification, even when the models are trained from scratch for such challenging few-shot cross-domain classification tasks.

Our approach differs from self-supervised learning approaches [93, 138, 493] primarily in using the unlabeled data continually v.s. one-time (Fig. 9.2-(c)). Further, these approaches define an auxiliary task for learning from data. While most of these approaches are using the unlabeled images, it is not clear if the performance improvement is due to the task or the images. In this work, we use unlabeled images for the evolution of task-representation without any auxiliary knowledge. Since we do not introduce any auxiliary task in our formulation, we are able to demonstrate

the influence of increase in data size and model size in learning a better representation.

Learning with Limited Labeled Data: A wide variety of work in few-shot learning [350,457,464], meta-learning [354,401,408] aims to learn from few labeled samples. However, these approaches often utilize "pre-trained" knowledge (Fig. 9.2-(d)) from large annotated data from a similar domain (such as ImageNet) [70]. Closely related to ours is the work of Li et al. [248] that also use a large corpus of unlabeled images in a few-shot learning setup with a pre-trained model. While we do show benefits of our approach when initializing using a good model – *we stress that one can learn a good representation even when we do not have a lot of labeled data from any source or a pre-trained model*.

Learning Unusual Stuff from Usual Data: Guo et al. [157] show that meta-learning methods [116, 246, 401, 408, 429, 440] underperform simple finetuning when pre-training on large annotated datasets from similar domains to the few-shot target task. Our work however is both domain- and task-agnostic. For example, we show substantial performance improvement in surface normal estimation [123, 452] on NYU-v2-depth [393] (that is primarily an indoor world dataset collected using a Kinect) via an unlabeled stream of web images. We similarly show that unlabeled internet streams can be used to improve classification accuracy of crop-diseases [300], satellite imagery [177], and medical images [79, 428] with even a modest number of labeled examples (20 examples per class).

9.3 Continual Evolution via Streaming Learning

Our continually-evolving approach can be derived as a continuous extension of semi-supervised learning algorithms. To derive our approach, assume we have access to an optimization routine that minimizes the loss on a supervised data set of labeled examples $(x, y) \in S$:

$$\text{Learn}(\mathcal{H}, S) \leftarrow_{F \in \mathcal{H}} \sum_{(x,y) \in S} \text{loss}(y, F(x)) \tag{9.1}$$

We will explore continually-evolving learning paradigms where the model class $\mathcal{H}$ grows in complexity over time (e.g., deeper models). We assume the gradient-based optimization routine is randomly initialized "from scratch" unless otherwise stated.

Semi-supervised learning: In practice, labeled samples are often limited. Semi-supervised learning assumes one has access to a large amount of unlabeled data $x \in U$. We specifically build on a family of deep semi-supervised approaches that psuedo-label unsupervised data U with a model trained on supervised data S [10, 195, 245]. Since these psuedo-labels will be noisy, it is common to pre-train on this large set, but fine-tune the final model on the pristine supervised set S [481]. Specifically, after learning an initial model F on the supervised set S:

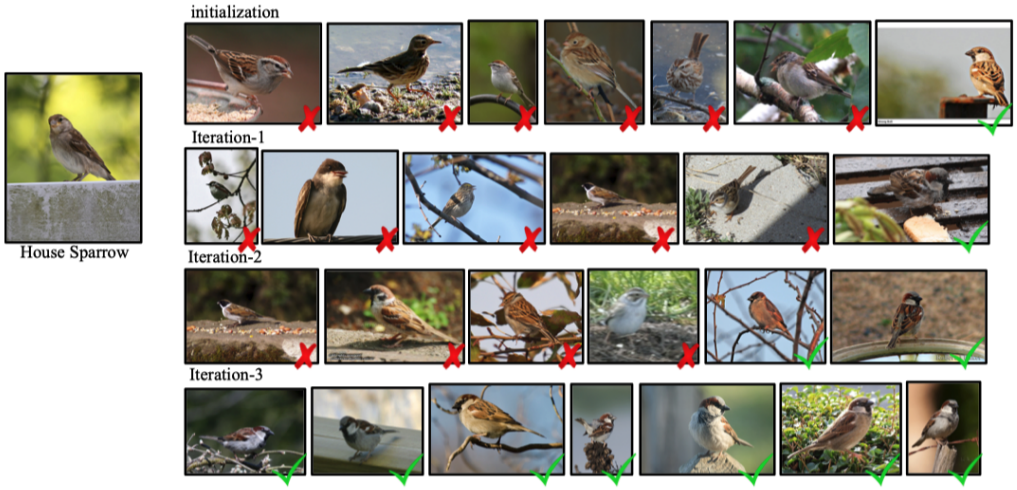


Figure 9.6: **Continual Improvement for House Sparrow:** We qualitatively show improvement in recognizing a house sparrow (shown in left from CUB-200 dataset [463]). At **initialization**, the trained model confuses a house sparrow with savannah sparrow, field sparrow, clay-colored sparrow, tree sparrow, house wren etc. We get rid of false-positives with every **iteration**. With more iterations, the false positives become fewer.

1. Use F to psuedo-label U.
2. Learn a new model F' from random initialization on the pseudo-labelled U.
3. Fine-tune F' on S.

Iterative learning: The above 3 steps can be iterated for improved performance, visually shown in Fig. 9.3. It is natural to ask whether repeated iteration will potentially oscillate or necessarily converge to a stable model and set of pseudo-labels. The above iterative algorithm can be written as an approximate coordinate descent optimization [467] of a latent-variable objective function:

$$\min_{\{z\},F\in\mathcal{H}} \sum_{(x,y)\in S} \text{loss}(y, F(x)) + \sum_{x\in U} \text{loss}(z, F(x)) \tag{9.2}$$

Step 1 optimizes for latent labels $\{z\}$ that minimize the loss, which are obtained by assigning them to the output of model $z := F(x)$ for each unlabeled example x. Step 2 and 3 optimize for F in a two-stage fashion. Under the (admittedly strong) assumption that this two-stage optimization finds the globally optimal F, the above will converge to a fixed point solution. In practice, we do not observe oscillations and find that model accuracy consistently improves.

Algorithm 1: StreamLearning$(S, \{U_t\}_{t=1}^T, \{\mathcal{H}_t\}_{t=1}^T)$

Input : S: Labeled dataset
$\{U_t\}_{t=1}^T$: T slices from unlabeled stream
$\{\mathcal{H}_t\}_{t=1}^T$: T hypothesis classes
Output: F
`// Initialize the model on S`
$F \leftarrow$ Learn$(\mathcal{H}_1, S)$;
for $t \leftarrow 1$ *to* T **do**
  `// Pseudo-label stream slice`
  $U \leftarrow \{(x, F(x)) : x \in U_t\}$;
  `// Pretrain model on` U
  $F' \leftarrow$ Learn$(\mathcal{H}_t, U)$;
  `// Fine-tune model on` S
  $F \leftarrow$ Finetune(F', S);
end

Streaming learning: We point out two important extensions, motivated by the fact that the unsupervised set U can be massively large, or even an infinite stream (e.g., obtained by an online web crawler). In this case, Step 1 may take an exorbitant amount of time to finish labeling on U. Instead, it is convenient to "slice" up U into a streaming collection of unsupervised datasets U_t of manageable (but potentially growing) size, and simply replace U with U_t in Step 1 and 2. One significant benefit of this approach is that as U_t grows in size, we can explore larger and deeper models (since our approach allows us to pre-train on an arbitrarily large dataset U_t). In practice, we train a family of models $\mathcal{H}_t$ of increasing capacity on U_t. Our final continuous algorithm is formalized in Alg. 1.

Experiments: We study our observation on a wide variety of tasks. Our method is both task- and domain-agnostic, as we do not make assumption about either the learning task nor the distributions of the labeled data. We randomly sample from 14M images of ImageNet-21K [86] without ground truth labels as the unlabeled dataset. We extensively present the analysis for image classification in Section 9.4. We then extend the analysis for surface normal estimation and semantic segmentation in Section 9.5 and applicability of our approach for non-internet images such as medical image classification, crop-disease classification, and satellite-image classification in Section 9.6. These experiments are mostly conducted using a machine with 4 GPUs (GeForce RTX 2080).

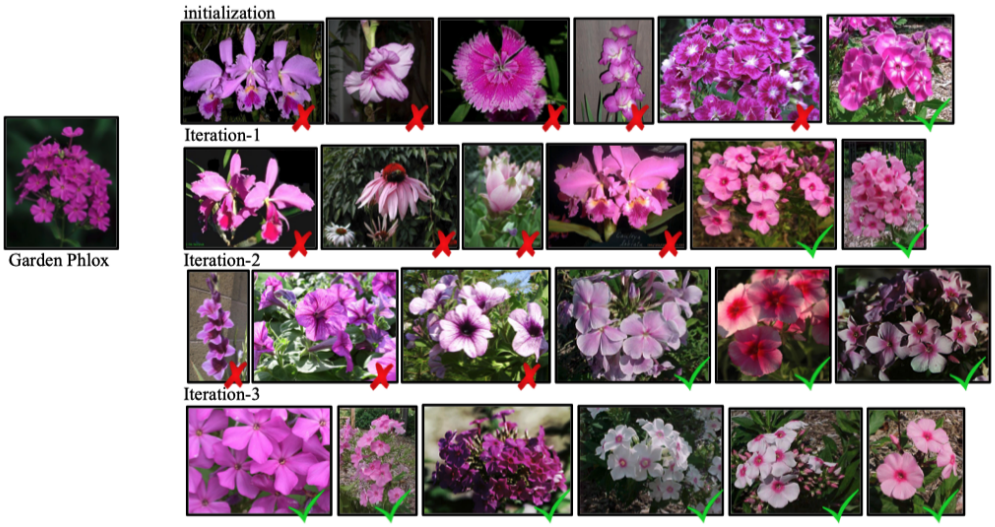


Figure 9.7: **Continual Improvement for Garden Phlox:** We qualitatively show improvement in recognizing a garden phlox (shown in left from flowers-102 dataset [313]). At **initialization**, the trained model confuses garden phlox with sword lily, sweet william, petunia, cape flower etc. With more iterations, the false positives become fewer.

9.4 Fine-Grained Image Classification

We first describe our experimental setup and then study this task using: (1) **Flowers-102** [313] that has 10 labeled examples per class; (2) **CUB-200** [463] that has 30 labeled examples per class; and (3) finally, we have also added analysis on a randomly sampled 20 examples per class from ImageNet-1k [362] (which we termed as **TwentyI-1000**). We use the original validation set [362] for this setup.

Model: We use the ResNet [170] model family as the hypothesis classes in Alg. 1, including ResNet-18, ResNet-34, ResNet-50, ResNext-50, and ResNext-101 [476]. The models are ranked in an increasing order of model complexity. Model weights are randomly generated by He initialization [171] (a random gaussian distribution) unless otherwise specified. We show in Table 9.1 that training deeper neural networks with few labeled examples is non-trivial.

Learning F from the labeled sample S: Given the low-shot training set, we use the cross entropy loss to train the recognition model. We adopt the SGD optimizer with momentum 0.9 and a L2 weight decay of 0.0001. We always start with learning rate 0.1. For **Flowers-102** and **CUB-200**, we decay the learning rate by 0.1 every 100 epochs, and train for a total of 250 epochs. For **TwentyI-1000**, we decay the learning rate by 0.1 every 60 epochs, and train for a total of 150 epochs.

Learning F' from U with pseudo labels: Once we learn F, we use it to gener-

One-stage models trained from scratch			
Model	Flowers-102	CUB-200	TwentyI-1000
Resnet-18	**45.49**	**44.03**	**13.92**
Resnet-34	42.64	**44.17**	**14.23**
Resnet-50	20.82	21.73	12.93
Resnext-50	31.34	28.37	11.87
Resnext-101	34.18	32.31	13.35

Table 9.1: We show performance of various models when trained from scratch. It is non-trivial to train a deep neural network with a few-labeled examples as shown in this analysis. Despite increasing the capacity of the models and training them for longer, we do not observe any performance improvement.

ate labels on a set of randomly sampled images from *ImageNet-21K* dataset to get pseudo-labelled U. Then we randomly initialize a new model F' as we do for F, then apply regular network training on U.

Finetuning F' on labeled sample S: After training F' on the pseudo-labeled U, we finetune F' on the original low-shot training set with the same training procedure and hyper-parameters. We use this finetuned model F' for test set evaluation.

Streaming Schedule and Model Selection: We empirically observe that instead of training on entire unlabeled set U, we can slice up U into a streaming collections U_t. In these experiments, we use three iterations of our approach. We have 1M samples in U_1, 3M samples in U_2, and 7M samples in U_3. We initialize the task using a ResNet-18 model (as shown in Table 9.1, ResNet-18 gets competitive performance and requires less computational resources). We use a ResNext-50 model as F' to train on U_1 and U_2, and a ResNext-101 model to train on U_3. These design decisions are based on empirical and pragmatic observations. For e.g., ResNext-50 slightly underperforms ResNext-101 for both U_1 and U_2. We still use ResNext-50 because it is both faster to train and process unlabeled images. Table 9.2 shows continuous improvement for various image-classification tasks at every iteration when using a few-labeled samples and training a model from scratch. We see similar trends for three different tasks. Figure 9.4 show the improvement in recognizing the common yellow-throat bird. We show more results in Figure 9.5, Figure 9.6, Figure 9.7, Figure 9.8, and Figure 9.9. At initialization, the model confused with similar-looking birds. We get rid of false-positives with every iteration. We are also able to bridge the gap between the popularly used pre-trained model (initialized using 1.2M labeled examples [362]) and a model trained from scratch without any extra knowledge or assumption.

What if we fix the model size in the iterations? We observe that using deeper model could lead to faster improvement of the performance. We perform an ab-

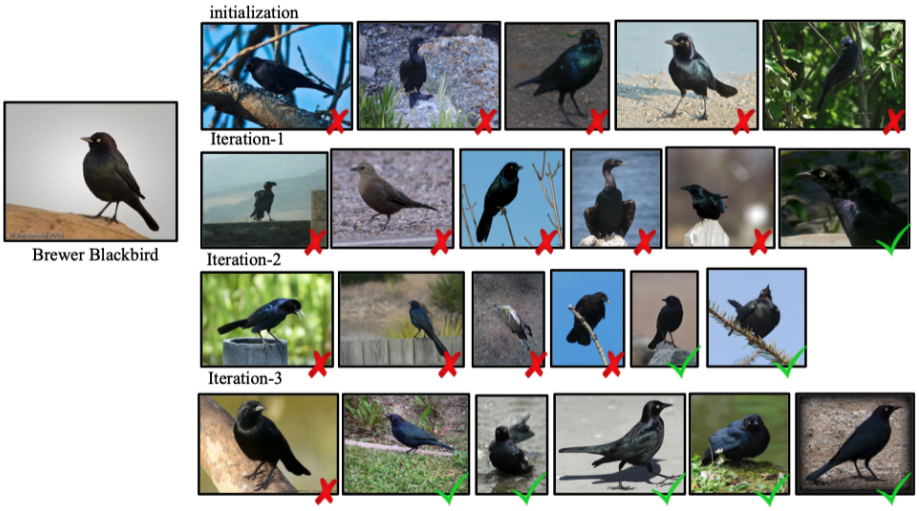


Figure 9.8: **Continual Improvement for Brewer Blackbird:** We qualitatively show improvement in recognizing a brewer blackbird (shown in left from CUB-200 dataset [463]). At **initialization**, the trained model confuses a brewer blackbird with boat tailed grackle, shiny cowbird, rusty blackbird, fish crow, american crow etc. We get rid of false-positives with every **iteration**. With more iterations, the false positives become fewer.

Continually-Improving Image Classification

Task	pre-trained	init	U_1	U_2	U_3	...
Flowers-102	89.12	45.49	54.19	65.25	72.68	...
CUB-200	75.29	44.03	53.73	57.11	66.05	...
TwentyI-1000	77.62	13.92	22.79	24.94	27.27	...

Table 9.2: We continually improve the performance for Flowers-102, CUB-200, and TwentyI-1000, as shown by top-1 accuracy for each iteration. We achieve a large performance improvement for each iteration for all the tasks. This is due to the combination of both increasing unlabeled dataset and model size. Without any supervision, we can bridge the gap between an ImageNet-1k pre-trained model and a model trained from scratch on Flowers-102 and CUB-200 dataset.

lative study by only training a ResNet-18 model using *TwentyI-1000*, as shown in Table 9.3. We could still see the accuracy improving with more unlabeled data, but increasing model capacity turns out to be more effective.

What if we train without iterations? Intuitively, more iterations with our algorithm should lead to an increased performance. We verify this hypothesis by con-

What if we use ResNet-18 for all experiments?

Model	init	U_1	U_2	U_3	...
ResNet-18 only	13.92	19.61	21.22	22.13	...
StreamLearning	13.92	**22.79**	**24.94**	**27.27**	...

Table 9.3: We show that the top-1 validation accuracy on TwentyI-1000 for our **StreamLearning** approach (row 2) for each iteration, which increases the model capacity from ResNet-18 (init) to ResNext-50 (U_1 and U_2) to ResNext-101 (U_3). With **ResNet-18 only** (row 1), the performance gain is much slower.

ducting another ablative study on *TwentyI-1000* experiment. In Table 9.4, we compare the result we got with three iterations (sequentially trained on U_1,U_2,U_3) with that of a single iteration (directly jumping to U_3 while skipping U_1, U_2). We could see that the performance with more iterations is indeed superior.

What if we train without iterations?

Model	init	U_1	U_2	U_3	...
Single Iteration	13.92	–	–	**23.77**	...
StreamLearning	13.92	22.79	24.94	**27.27**	...

Table 9.4: We show that the top-1 validation accuracy on TwentyI-1000 for our **StreamLearning** approach (row 2) for each iteration, which increases the model capacity from ResNet-18 (init) to ResNext-50 (U_1 and U_2) to ResNext-101 (U_3). This result is compared with training with a single iteration on U_3 (also using ResNext-101). As expected, more iterations of our approach on more unlabeled data will lead to an improved performance.

Why do we use ResNext-50 for U_1 and U_2? We show in Table 9.5 that ResNext-50 outperforms ResNet-18 in first iteration to justify the model decision of our stream learning approach. Note that this is not saying ResNext-50 is the best performing model among all possible choices. For instance, ResNext-101 slightly outperforms ResNext-50 (around 1% improvement) on the first two iterations, but we still use ResNext-50 for U_1 and U_2 for pragmatic reasons (faster to train and save more memory). In practice, one can trade off generalization performance and training speed by select the most suitable model size just like what we did in this work.

9.5 Pixel Analysis

We extend our analysis to pixel-level prediction problems. We study surface-normal estimation using NYU-v2 depth dataset [393]. We intentionally chose this task be-

Stream Learning performance on U_1

Model	CUB-200	Flowers-102	TwentyI-1000
ResNet-18	51.35	47.50	19.61
ResNext-50	**53.73**	**54.19**	**22.79**

Table 9.5: We show that the top-1 validation accuracy on all fine-grained classification datasets with our **StreamLearning** approach after the first iteration (U_1 with 1M unlabeled images) training with ResNet-18 or ResNext-50. We can see that ResNext-50 consistently outperforms ResNet-18 across all tasks.

Figure 9.9: **Continual Improvement for Bromelia:** We qualitatively show improvement in recognizing a bromelia (shown in left from flowers-102 dataset [313]). At **initialization**, the trained model confuses bromelia with snapdragon, rose, water lily, lotus, wallflower etc. With more iterations, the false positives become fewer.

cause there is a large domain gap between NYU-v2 depth dataset and internet images of ImageNet-21k.

We follow Bansal et al. [21, 23] (our prior work) for surface normal estimation because: (1) they demonstrate training a reasonable model from scratch; and (2) used the learned representation for downstream tasks. This allows us to do a proper comparison with an established baseline and study the robustness of the models. Finally, it allows us to verify if our approach holds for a different backbone-architecture (VGG-16 [395] in this case).

Evaluation: We use 654 images from the test set of NYU-v2 depth dataset for evaluation. Following [23], we compute six statistics over the angular error between the predicted normals and depth-based normals to evaluate the performance

– **Mean**, **Median**, **RMSE**, **11.25°**, **22.5°**, and **30°** – The first three criteria capture the mean, median, and RMSE of angular error, where lower is better. The last three criteria capture the percentage of pixels within a given angular error, where higher is better.

Table 9.6 contrasts the performance of our approach with Bansal et al [21, 23]. They use a pre-trained ImageNet classification model for initialization. In this work, we initialize a model from random gaussian initialization (also known as scratch). The middle row shows the performance when a model is trained from scratch. We improve this model using a million unlabeled images. The last row shows the performance after one iteration of our approach. We improve by 3-6% without any knowledge of surface normal estimation task. Importantly, we outperform the pre-trained ImageNet initialization. This suggests that we can design better task-specific architecture that are not tied to pre-training with ImageNet labeled dataset. We also contrast our method with Goyal et al. [150] (second-row) that use $100M$ unlabeled images to train a generic representation via jigsaw puzzle [316] using a ResNet-50 model. We significantly outperform them. This means we can learn better task-specific representation with two-orders less unlabeled data [150].

In this experiment, our approach significantly outperform the prior-art using a million unlabeled images in a single iteration. Figure 9.10 contrasts our approach with other models.

Do we improve globally or locally? One may suspect that a model initialized with the weights of pre-trained ImageNet classification model may capture more local information as the pre-training consists of class labels. Table 9.8 contrast the performance of two approaches on indoor scene furniture categories such as chair, sofa, and bed. The performance of our model exceeds prior art for local objects as well. This suggests that we can capture both local and global information quite well without class-specific information.

Can we improve *scratch* by training longer? It is natural to ask if we could improve the performance by training a model from scratch for more iterations. Table 9.7 shows the performance of training the scratch model for longer (until convergence). We observe that we do improve slightly over the model we use. However, this is significantly outperformed by streaming learning.

Is it a robust representation? Bansal et al. [21] has used the model trained for surface-normal as an initialization for the task of semantic segmentation. We study if a better surface normal estimation means better initialization for semantic segmentation. We use the training images from PASCAL VOC-2012 [108] for semantic segmentation, and additional labels collected on 8498 images by [161] for this experiment. We evaluate the performance on the test set that required submission on PASCAL web server [1]. We report results using the standard metrics of region intersection over union (**IoU**) averaged over classes (higher is better).

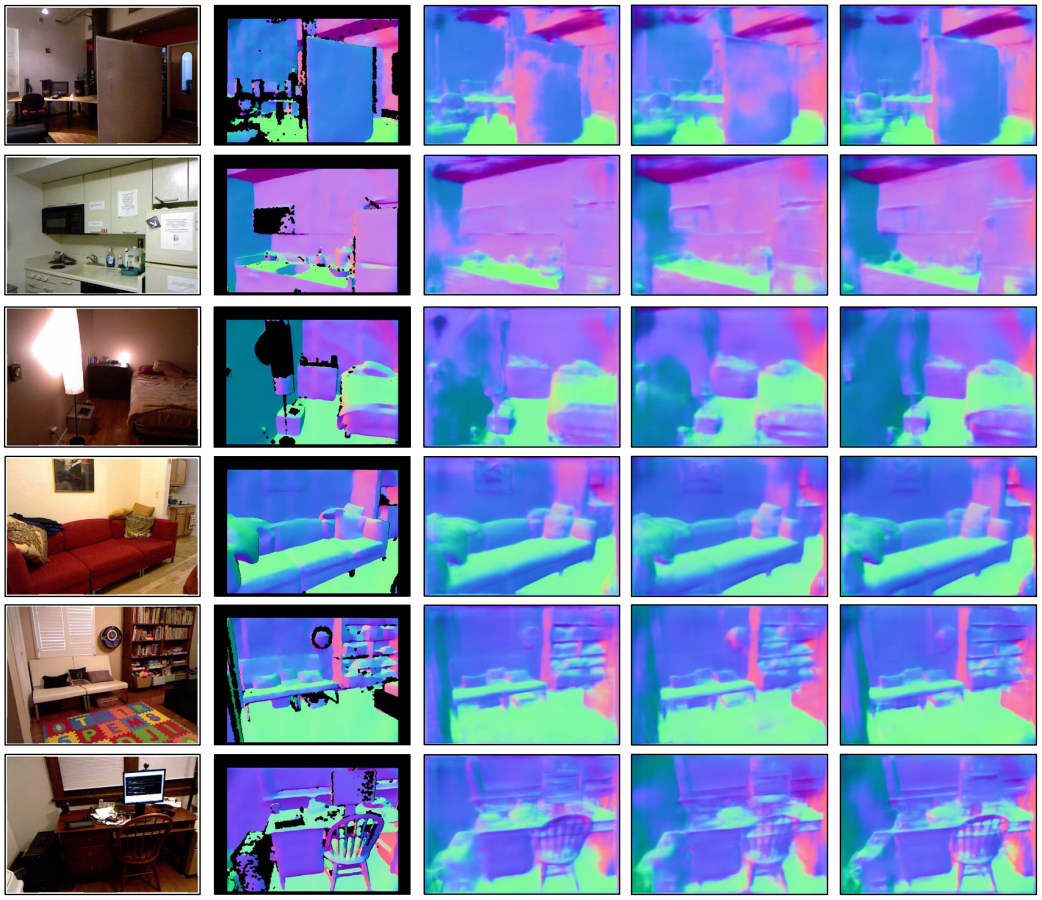
We show our findings in Table 9.9. We contrast the performance of surface-normal model trained from scratch (as in [21]) in the second row with our model in

Surface Normal Estimation (NYU Depthv2)						
Approach	Mean ↓	Median ↓	RMSE ↓	11.25° ↑	22.5° ↑	30° ↑
Bansal et al. [23]	19.8	12.0	28.2	47.9	70.0	77.8
Goyal et al. [150]	22.4	13.1	-	44.6	67.4	75.1
scratch (init)	21.2	13.4	29.6	44.2	66.6	75.1
ours (one-iteration)	**18.7**	**10.8**	**27.2**	**51.3**	**71.9**	**79.3**

Table 9.6: We contrast the performance of our approach with Bansal et al [21, 23], which is the state-of-the-art given our setup. They use a pre-trained ImageNet classification model for initialization. In this work, we initialize a model from random gaussian initialization (also known as scratch). The third row shows the performance of a scratch-initialized model. We improve this model using one million unlabeled images. The last row shows the performance after one iteration of our approach. We improve by 3-6% without any domain-specific knowledge about the surface normal estimation task. Importantly, we outperform the pre-trained ImageNet initialization. We contrast our method with Goyal et al. [150] (second-row), which use $100M$ unlabeled images to train a generic representation via jigsaw puzzle [316] using a ResNet-50 model. Our model trained from scratch competes with their best performing model. This analysis suggests two things: (1) we can design better task-specific architecture that are not tied to pre-training with ImageNet labeled dataset; and (2) we can learn better task-specific representation with two-orders less unlabeled data as compared to [150].

Approach	Mean	Median	RMSE	11.25°	22.5°	30°
pre-trained [23]	19.8	12.0	28.2	47.9	70.0	77.8
init	21.2	13.4	29.6	44.2	66.6	75.1
init (until convergence)	20.4	12.6	28.7	46.3	68.2	76.4
ours (one-iteration)	**18.7**	**10.8**	**27.2**	**51.3**	**71.9**	**79.3**

Table 9.7: **Can we improve scratch by training longer?** It is natural to ask if we can get better performance for training longer, crucially for a model trained from scratch. We observe that one can indeed get a slightly better performance by training for a long time. However, this is significantly outperformed by streaming learning.



(a) 2D Image (b) Kinect (c) Bansal et al. (d) scratch (e) ours

Figure 9.10: **Surface Normal Estimation:** For a given single 2D image (shown in (**a**)), we contrast the performance of various models. Shown in (**c**) are the results from prior work [21, 23] using a model pretrained with *ImageNet-1K* labels; (**d**) shows a model trained from scratch starting from random gaussian initialization; and finally (**e**) shows the result of our StreamLearning approach. The influence of unlabeled data can be gauged by improvements from (**d**) to (**e**). By utilizing diverse unlabeled data, we can get better performance without any additional supervision. For reference, we also show ground truth normals from kinect in (**b**).

Per-Object Surface Normal Estimation (NYU Depthv2)						
	Mean ↓	Median ↓	RMSE ↓	11.25° ↑	22.5° ↑	30° ↑
chair						
Bansal et al. [23]	31.7	24.0	40.2	**21.4**	47.3	58.9
ours (one-iteration)	**31.2**	**23.6**	**39.6**	21.0	**47.9**	**59.8**
sofa						
Bansal et al. [23]	20.6	15.7	26.7	35.5	66.8	78.2
ours (one-iteration)	**20.0**	**15.2**	**26.1**	**37.5**	**67.5**	**79.4**
bed						
Bansal et al. [23]	19.3	13.1	26.6	44.0	70.2	80.0
ours (one-iteration)	**18.4**	**12.3**	**25.5**	**46.5**	**72.7**	**81.7**

Table 9.8: We contrast the performance of our approach with the model fine-tuned using ImageNet (with class labels) on furniture categories, i.e. chair, sofa, and bed. Our approach outperforms prior art without any class information.

the third row. We observe a significant 2% performance improvement. This means better surface normal estimation amounts to a better initialization for semantic segmentation, and that we have a robust representation that can be used for downstream tasks.

Can we improve semantic segmentation further? Can we still improve the performance of a task when we start from a better initialization other than scratch? We contrast the performance of the methods in the third row (init) to the fourth row (improvement in one-iteration). We observe another significant 2.7% improvement in IoU. This conveys that we can indeed apply our insights even when starting from an initialization better than scratch. Finally, we observe that our approach has closed the gap between ImageNet (with class labels) pre-trained model and a self-supervised model to 3.6%. We believe more iterations of our approach would fill the gap.

9.6 Extreme-Task Differences

We now extend our analysis to the setup where the data-distribution is drastically different from the usual world-images. We use: (1) **EuroSat** [177] (satellite imagery) dataset for classifying satellite-captured images into distinct regions; (2) **ISIC2018** [79] (lesion diagnosis) for medical-image classification of skin diseases; and (3) **CropDiseases** [300] dataset which is a crop-disease classification task. We use 20 examples per class for each dataset and train the models from scratch.

Table 9.10 shows the performance for the three different tasks. We did not make

Semantic Segmentation on VOC-2012

	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	bg	**IoU ↑**
scratch-init	62.3	26.8	41.4	34.9	44.8	72.2	59.5	56.0	16.2	49.9	45.0	49.7	53.3	63.6	65.4	26.5	46.9	37.6	57.0	40.4	85.2	**49.3**
normals-init	71.8	29.7	51.8	42.1	47.8	77.9	65.9	59.7	19.7	50.8	45.9	55.0	59.1	68.2	69.3	32.5	54.3	42.1	60.8	43.8	87.6	**54.1**
normalsStream-init	74.4	34.5	60.5	47.3	57.1	74.3	73.1	61.7	22.4	51.4	36.4	52.0	60.9	68.5	69.1	37.6	58.0	34.3	64.3	50.2	90.0	**56.1**
+one-iteration	82.2	35.1	62.0	47.4	62.1	76.6	74.1	62.7	23.9	49.9	47.0	55.5	58.0	74.9	73.9	40.1	56.4	43.6	65.4	52.8	90.9	**58.8**
pre-trained [21]	79.0	33.5	69.4	51.7	66.8	79.3	75.8	72.4	25.1	57.8	52.0	65.8	68.2	71.2	74.0	44.1	63.7	43.4	69.3	56.4	91.1	**62.4**

Table 9.9: The goal of this experiment is to study two things: **(1)** *Can task-specific representations learned on unlabeled streams generalize to other tasks?* This allows us to study the robustness of our learned representations. We consider the target task of semantic segmentation and the source task of surface-normal estimation. Segmentation networks initialized with surface-normal networks already outperform random initialization (row2 vs row1), and further improve by 2% when initialized with stream-trained networks (row3). **(2)** *Can we still further improve the performance of a task when starting from an initialization better than scratch?* We then perform one additional iteration of stream learning (row4 vs row3), resulting in another 2.7% improvement, closing the gap between ImageNet pre-training to 3.6%. We posit that more iterations of our approach may fill the gap.

Task	pre-trained	init	U_1
East-SAT [177]	68.93	70.57	73.58
Lesion [79]	45.43	44.86	50.86
Crop [300]	94.68	87.49	90.86

Table 9.10: **Extreme-Task Differences:** We extend our analysis to tasks that operate on specialized data distributions. We observe significant performance improvement despite using unlabeled streams of internet images. We also achieve performance competitive to the ImageNet-1k pre-trained model (again, trained with a large amount of labels). We use ResNet-18 for all experiments in the table.

any modification to our setup (Section 9.4) and yet achieve significant improvement for each of them. We also show the performance of a pre-trained (using 1.2M labeled examples [362]) model on these datasets. Guo et al. [157] suggested that fine-tuning a pre-trained model generally leads to best performances on these tasks. We observe that a simple random-gaussian initialization works as well despite trained using only a few labeled examples.

Crucially, we use unlabeled Internet images for learning a better representation on classification tasks containing classes that are extremely different to real-world object categories. Still, we see significant improvements.

9.7 Discussion

We present a simple and intuitive approach to semi-supervised learning on (potentially) infinite streams of unlabeled data. Our approach integrates insights from different bodies of work including self-training [96, 459], pseudo-labelling [10, 195, 245], continual/iterated learning [229, 230, 394, 421, 422], and few-shot learning [157, 248]. We demonstrate a number of surprising conclusions: (1) Unlabeled *domain-agnostic* internet streams can be used to significantly improve models for specialized tasks and data domains, including surface normal prediction, semantic segmentation, and few-shot fine-grained image classification spanning diverse domains including medical, satellite, and agricultural imagery. (2) Continual learning on streams can be initialized with very *impoverished models* trained (from scratch) on tens of labeled examples. This is in contrast with much work in semi-supervised learning that requires a good model for initialization. (3) Contrary to popular approaches in semi-supervised learning that make use of massive compute resources for storing and processing data, streaming learning requires *modest computational infrastructure* since it naturally breaks up massive datasets into slices that are manageable for processing. From this perspective, continual learning on streams can help democratize research and development for scalable, lifelong ML.

Chapter 10

Conclusion & Future Work

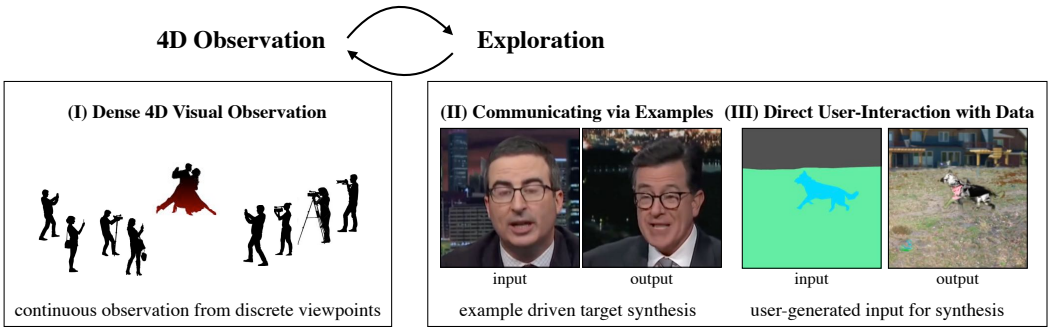


Figure 10.1: **Future Work:** We have, thus far, studied the different components of this thesis independently. The future work will bring both 4D observation and exploration together in a continual and streaming manner. This paradigm shift will allow us to explore sparse and challenging setup. The presence of a human-user would explicitly help us in improving the quality of results. Eventually, this would bring us close to the goal of audio-visual communication where we can communicate effectively with one another and the machines.

In this thesis, we demonstrate that we can indeed learn a computational representation of the 4D audio-visual world that can be: (1) estimated from sparse real-world observations; and (2) explored to create new experiences via examples and direct human interaction.

Future Work: We have, thus far, studied these components independently. The future work will bring both 4D observation and exploration together in a continual and streaming manner (Figure 10.1). A user sees the 4D world, explores it, tells the machine what to do if there is something wrong. The machine learns it, improves further, and then collect more observations. Intuitively, this is how we, humans, learn as well. Our work on continual and streaming learning with few examples

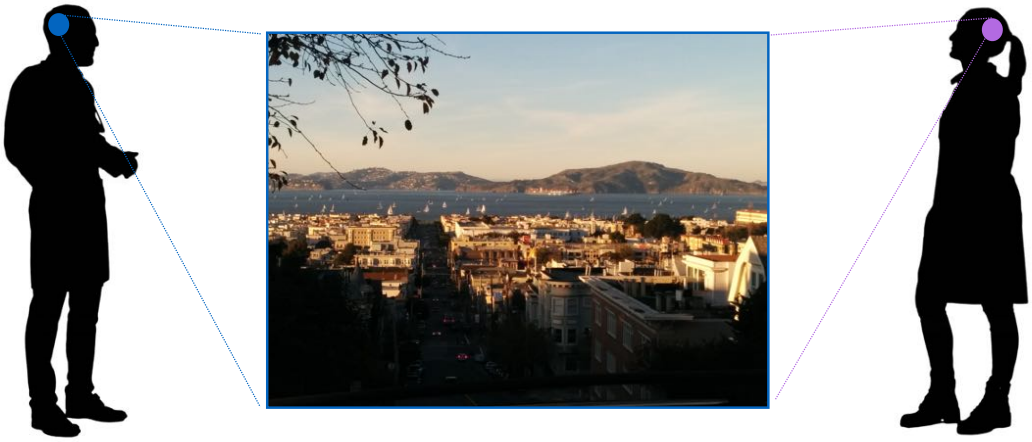


Figure 10.2: **A Lossless Communication Channel:** We envision a lossless communication channel where both Alice and Bob can share their experiences and imagination with one another – thereby enabling them to relive the important moments of their life.

demonstrates that it is indeed possible for a machine to learn in this manner. This paradigm shift will allow us to explore even more sparse and challenging setup. The presence of a human-user would explicitly help us to improve the quality of results. It would also lead to the development of user-interfaces that will allow a better human-machine communication. Most importantly, this would bring us close to the goal of the audio-visual social communication where we can communicate effectively with one another and the machines. Figure 10.2 presents the vision of a lossless communication channel between Alice and Bob that allows them to share their experiences and imagination.

Conclusion: The space of human imagination is much greater than the space or our abilities to physically capture the world. We, humans, have amazing mental powers to reconstruct or imagine the scenarios from sparse signals or tiny bits of information. The fundamental question is how can machines densely reconstruct and generalize from sparse signals. We are currently in the early days of this research. We believe the next decade of research will bring more diverse ideas to solve this problem. With personal computers for communication — *the world will be our personal studio*!

Bibliography

[1] Pascal voc server. https://host.robots.ox.ac.uk:8080//.

[2] E. H. Adelson, J. R. Bergen, et al. *The plenoptic function and the elements of early vision*, volume 2. Vision and Modeling Group, Media Laboratory, Massachusetts Institute of ..., 1991.

[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *ICCV*, 2009.

[4] A. Agarwala, K. C. Zheng, C. Pal, M. Agrawala, M. Cohen, B. Curless, D. Salesin, and R. Szeliski. Panoramic video textures. *ACM Trans. Graph.*, 24(3):821–827, July 2005.

[5] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *ICCV*, 2015.

[6] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*. 2014.

[7] W.-K. Ahn and W. F. Brewer. Psychological studies of explanation—based learning. In *Investigating explanation-based learning*, pages 295–316. Springer, 1993.

[8] W.-K. Ahn, R. J. Mooney, W. F. Brewer, and G. F. DeJong. Schema acquisition from one example: Psychological evidence for explanation-based learning. Technical report, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, 1987.

[9] K.-A. Aliev, A. Sevastopolsky, M. Kolos, D. Ulyanov, and V. Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019.

[10] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *IEEE IJCNN*, 2020.

[11] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *CVPR*. IEEE, 2012.

[12] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 2011.

[13] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. 2014.

[14] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for scene segmentation. *IEEE TPAMI*, 2017.

[15] Y. Bahat and M. Irani. Blind dehazing using internal patch recurrence. In *2016 IEEE International Conference on Computational Photography (ICCP)*, pages 1–9. IEEE, 2016.

[16] Y. Bahat and T. Michaeli. Explorable super resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2716–2725, 2020.

[17] S. Baker, D. Scharstein, J. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *International journal of computer vision*, 92(1):1–31, 2011.

[18] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 1989.

[19] L. Ballan, G. J. Brostow, J. Puwein, and M. Pollefeys. Unstructured video-based rendering: Interactive exploration of casually captured videos. *ACM Trans. Graph.*, 2010.

[20] L. Ballan and G. M. Cortelazzo. Marker-less motion capture of skinned models in a four camera set-up using optical flow and silhouettes. 3DPVT, 2008.

[21] A. Bansal, X. Chen, B. Russell, A. Gupta, and D. Ramanan. PixelNet: Representation of the pixels, by the pixels, and for the pixels. *arXiv:1702.06506*, 2017.

[22] A. Bansal, S. Ma, D. Ramanan, and Y. Sheikh. Recycle-gan: Unsupervised video retargeting. In *ECCV*, 2018.

[23] A. Bansal, B. Russell, and A. Gupta. Marr Revisited: 2D-3D model alignment via surface normal prediction. In *CVPR*, 2016.

[24] A. Bansal, Y. Sheikh, and D. Ramanan. PixelNN: Example-based image synthesis. In *ICLR*, 2018.

[25] A. Bansal, Y. Sheikh, and D. Ramanan. Shapes and context: In-the-wild image synthesis manipulation. In *CVPR*, 2019.

[26] A. Bansal, M. Vo, Y. Sheikh, D. Ramanan, and S. Narasimhan. 4d visualization of dynamic events from unconstrained multi-view videos. In *CVPR*, 2020.

[27] M. Bar. The proactive brain: using analogies and associations to generate predictions. *Trends in cognitive sciences*, 2007.

[28] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 2009.

[29] J. T. Barron and B. Poole. The fast bilateral solver. *CoRR*, abs/1511.03296, 2015.

[30] D. Bashkirova, B. Usman, and K. Saenko. Unsupervised video-to-video translation. *CoRR*, abs/1806.03698, 2018.

[31] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. *CoRR*, 2017.

[32] S. Bell and K. Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Transactions on Graphics*, 2015.

[33] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002.

[34] S. Benaim, A. Ephrat, O. Lang, I. Mosseri, W. T. Freeman, M. Rubinstein, M. Irani, and T. Dekel. Speednet: Learning the speediness in videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9922–9931, 2020.

[35] Y. Bengio, L. Yao, G. Alain, and P. Vincent. Generalized denoising auto-encoders as generative models. *Advances in neural information processing systems*, 26:899–907, 2013.

[36] F. Berthouzoz, W. Li, and M. Agrawala. Tools for placing cuts and transitions in interview video. *ACM Trans. Graph.*, 2012.

[37] I. Biederman. *On the Semantics of a Glance at a Scene*.

[38] I. Biederman. Recognition by components: a theory of human image interpretation. *Pyschological review*, 94:115–147, 1987.

[39] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *IEEE transactions on visualization and computer graphics*, 24(1):152–162, 2018.

[40] C. M. Bishop. *Neural networks for pattern recognition*. Oxford university press, 1995.

[41] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[42] O. Boiman and M. Irani. Similarity by composition. In *NeurIPS*. 2006.

[43] O. Boiman and M. Irani. Detecting irregularities in images and in video. *IJCV*, 2007.

[44] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister. Interactive intrinsic video editing. *ACM Transactions on Graphics* (*Proceedings of SIGGRAPH Asia 2014*), 33(6), 2014.

[45] A. Bordes, L. Bottou, and P. Gallinari. Sgd-qn: Careful quasi-newton stochastic gradient descent. *JMLR*, 10, 2009.

[46] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[47] M. Brand and A. Hertzmann. Style machines. ACM SIGGRAPH. ACM Press/Addison-Wesley Publishing Co., 2000.

[48] C. Bregler, M. Covell, and M. Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 353–360, 1997.

[49] A. Brock, J. Donahue, and K. Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *ICLR*, 2019.

[50] M. Broxton, J. Flynn, R. Overbeck, D. Erickson, P. Hedman, M. Duvall, J. Dourgarian, J. Busch, M. Whalen, and P. Debevec. Immersive light field video with a layered mesh representation. *ACM Transactions on Graphics* (*TOG*), 2020.

[51] C. Buehler, M. Bosse, L. McMillan, S. Gortler, and M. Cohen. Unstructured lumigraph rendering. In *SIGGRAPH*, 2001.

[52] V. Bush and J. Wang. As we may think. *Atlantic Monthly*, 1945.

[53] W. Byeon, T. M. Breuel, F. Raue, and M. Liwicki. Scene labeling with lstm recurrent neural networks. In *CVPR*, 2015.

[54] C. Cao, Q. Hou, and K. Zhou. Displaced dynamic expression regression for real-time facial tracking and animation. *ACM Trans. Graph.*, 2014.

[55] S. Cao, X. Wang, and K. M. Kitani. Learnable embedding space for efficient neural architecture compression. In *ICLR*, 2019.

[56] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[57] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.

[58] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-viewpoint video of human actors. *ACM Trans. Graph.*, 2003.

[59] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*. 2012.

[60] F. M. Castro, M. J. Marín-Jiménez, N. Guil, C. Schmid, and K. Alahari. End-to-end incremental learning. In *ECCV*, 2018.

[61] Y.-L. Chang, Z. Y. Liu, K.-Y. Lee, and W. Hsu. Learnable gated temporal shift module for deep video inpainting. *arXiv preprint arXiv:1907.01131*, 2019.

[62] O. Chapelle, B. Scholkopf, and A. Zien. Semi-supervised learning. *IEEE Trans. NNLS*, 2009.

[63] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *British Machine Vision Conference*, 2014.

[64] L.-W. Chen, H.-Y. Lee, and Y. Tsao. Generative adversarial networks for unpaired voice transformation on impaired speech. *Proc. Interspeech 2019*, 2019.

[65] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *ICLR*, 2015.

[66] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, 2016.

[67] L.-H. Chen, Z.-H. Ling, L.-J. Liu, and L.-R. Dai. Voice conversion using deep neural networks with layer-wise generative training. *IEEE/ACM Transactions on Audio, Speech and Language Processing* (*TASLP*), 2014.

[68] Q. Chen and V. Koltun. Photographic image synthesis with cascaded refinement networks. *arXiv:1707.09405*, 2017.

[69] S. E. Chen and L. Williams. View interpolation for image synthesis. In *SIGGRAPH*, 1993.

[70] W.-Y. Chen, Y.-C. Liu, Z. Kira, Y.-C. F. Wang, and J.-B. Huang. A closer look at few-shot classification. *arXiv:1904.04232*, 2019.

[71] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, 2016.

[72] I. Choi, O. Gallo, A. Troccoli, M. H. Kim, and J. Kautz. Extreme view synthesis. In *CVPR*, 2019.

[73] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *CVPR*, 2005.

[74] J.-c. Chou, C.-c. Yeh, and H.-y. Lee. One-shot voice conversion by separating speaker and content representations with instance normalization. In *INTERSPEECH*, 2019.

[75] J.-C. Chou, C.-C. Yeh, H.-Y. Lee, and L.-S. Lee. Multi-target Voice Conversion without Parallel Data by Adversarially Learning Disentangled Audio Representations. *Proc. Interspeech*, 2018.

[76] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *NeurIPS*, 2016.

[77] J. S. Chung, A. Jamaludin, and A. Zisserman. You said that? *arXiv preprint arXiv:1705.02966*, 2017.

[78] J. S. Chung, A. Nagrani, and A. Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[79] N. Codella, V. Rotemberg, P. Tschandl, M. E. Celebi, S. Dusza, D. Gutman, B. Helba, A. Kalloo, K. Liopyris, M. Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv:1902.03368*, 2019.

[80] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.

[81] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[82] E. De Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. In *ACM SIGGRAPH*. 2008.

[83] J. Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. Large scale distributed deep networks. In *NIPS*, 2012.

[84] P. E. Debevec. *Modeling and Rendering Architecture from Photographs*. PhD thesis, University of California at Berkeley, Computer Science Division, Berkeley CA.

[85] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *ACM Trans. Graph.* ACM, 1996.

[86] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009.

[87] K. Deng, A. Bansal, and D. Ramanan. Unsupervised audiovisual synthesis via exemplar autoencoders. 2021. under review.

[88] E. L. Denton, S. Chintala, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, 2015.

[89] E. L. Denton, S. Chintala, A. Szlam, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. *CoRR*, 2015.

[90] N. Dinesh Reddy, M. Vo, and S. G. Narasimhan. Carfusion: Combining point tracking and part detection for dynamic 3d reconstruction of vehicles. In *CVPR*, 2018.

[91] N. Dinesh Reddy, M. Vo, and S. G. Narasimhan. Occlusion-net: 2d/3d occluded keypoint localization using graph networks. In *CVPR*, 2019.

[92] S. K. Divvala, D. Hoiem, J. H. Hays, A. A. Efros, and M. Hebert. An empirical study of context in object detection. In *CVPR*, 2009.

[93] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *ICCV*, 2015.

[94] P. Dollár and C. Zitnick. Structured forests for fast edge detection. In (*ICCV*, 2013.

[95] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *TPAMI*, 37(8), 2015.

[96] J. Du, E. Grave, B. Gunel, V. Chaudhary, O. Celebi, M. Auli, V. Stoyanov, and A. Conneau. Self-training improves pre-training for natural language understanding. *arXiv:2010.02194*, 2020.

[97] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[98] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing action at a distance. In *ICCV*, 2003.

[99] A. A. Efros and W. T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01. ACM, 2001.

[100] A. A. Efros and T. K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, 1999.

[101] B. Egger, W. A. P. Smith, A. Tewari, S. Wuhrer, M. Zollhoefer, T. Beeler, F. Bernard, T. Bolkart, A. Kortylewski, S. Romdhani, C. Theobalt, V. Blanz, and T. Vetter. 3d morphable face models – past, present and future, 2019.

[102] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *ICCV*, 2015.

[103] D. Eigen, D. Krishnan, and R. Fergus. Restoring an image taken through a window covered with dirt or rain. In *ICCV*, 2013.

[104] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NIPS*, 2014.

[105] D. C. Engelbart and W. K. English. A research center for augmenting human intellect. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 395–410, 1968.

[106] D. Epstein, B. Chen, and C. Vondrick. Oops! predicting unintentional action in video. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[107] R. Epstein and N. Kanwisher. A cortical representation of the local visual environment. *Nature*, 1998.

[108] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes (VOC) Challenge. *IJCV*, 2010.

[109] A. Faktor and M. Irani. Co-segmentation by composition. In *ICCV*, 2013.

[110] B. Fan, L. Wang, F. K. Soong, and L. Xie. Photo-real talking head with deep bidirectional lstm. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4884–4888. IEEE, 2015.

[111] F. Fang, J. Yamagishi, I. Echizen, and J. Lorenzo-Trueba. High-Quality Nonparallel Voice Conversion Based on Cycle-Consistent Adversarial Network. In *IEEE ICASSP*, 2018.

[112] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *TPAMI*, 35(8), 2013.

[113] P. Felzenszwalb, D. McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[114] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *IJCV*, 59(2), 2004.

[115] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9), 2010.

[116] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv:1703.03400*, 2017.

[117] C. Finn, A. Rajeswaran, S. Kakade, and S. Levine. Online meta-learning. *arXiv:1902.08438*, 2019.

[118] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. *arXiv preprint arXiv:1504.06852*, 2015.

[119] J. H. Flavell. *Cognitive development.* prentice-hall, 1977.

[120] J. Flynn, M. Broxton, P. Debevec, M. DuVall, G. Fyffe, R. Overbeck, N. Snavely, and R. Tucker. Deepview: View synthesis with learned gradient descent. In *CVPR*, 2019.

[121] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *CVPR*, 2016.

[122] R. Fong and A. Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. *arXiv preprint arXiv:1801.03454*, 2018.

[123] D. F. Fouhey, A. Gupta, and M. Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013.

[124] D. F. Fouhey, A. Gupta, and M. Hebert. Single image 3D without a single 3D image. In *ICCV*, 2015.

[125] D. F. Fouhey, A. Gupta, and M. Hebert. Unfolding an indoor origami world. In *ECCV*, 2014.

[126] J.-S. Franco and E. Boyer. Fusion of multiview silhouette cues using a space occupancy grid. In *ICCV*, 2005.

[127] W. T. Freeman and J. B. Tenenbaum. Learning bilinear models for two-factor problems in vision. In *CVPR*, 1997.

[128] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-based super-resolution. *IEEE Comput. Graph. Appl.*, 22(2):56–65, March 2002.

[129] O. Fried, A. Tewari, M. Zollhöfer, A. Finkelstein, E. Shechtman, D. B. Goldman, K. Genova, Z. Jin, C. Theobalt, and M. Agrawala. Text-based editing of talking-head video. *ACM Trans. Graph.*, 38(4):68:1–68:14, July 2019.

[130] H. Fuchs, G. Bishop, K. Arthur, L. McMillan, R. Bajcsy, S. Lee, H. Farid, and T. Kanade. Virtual space teleconferencing using a sea of cameras. In *Proc. First International Conference on Medical Robotics and Computer Assisted Surgery*, 1994.

[131] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multiview stereo. In *CVPR*, 2010.

[132] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *TPAMI*, 2009.

[133] C. Gao, A. Saraf, J.-B. Huang, and J. Kopf. Flow-edge guided video completion. In *Proc. European Conference on Computer Vision (ECCV)*, 2020.

[134] R. Gao, R. Feris, and K. Grauman. Learning to separate object sounds by watching unlabeled video. In *ECCV*, 2018.

[135] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *CVOR*, 2016.

[136] A. Ghosh, V. Kulharia, V. P. Namboodiri, P. H. S. Torr, and P. K. Dokania. Multi-agent diverse generative adversarial networks. In *CVPR*, 2018.

[137] J. J. Gibson. The ecological approach to visual perception. 1979.

[138] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. *CoRR*, 2018.

[139] S. Ginosar, A. Bar, G. Kohavi, C. Chan, A. Owens, and J. Malik. Learning individual styles of conversational gesture. In *CVPR*, 2019.

[140] R. Girdhar, D. Ramanan, A. Gupta, J. Sivic, and B. Russell. ActionVLAD: Learning spatio-temporal aggregation for action classification. In *CVPR*, 2017.

[141] R. Girshick. Fast r-cnn. In *ICCV*, 2015.

[142] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

[143] G. Gkioxari, B. Hariharan, R. Girshick, and J. Malik. Using k-poselets for detecting people and localizing their keypoints. 2014.

[144] D. Glasner, S. Bagon, and M. Irani. Super-resolution from a single image. In *2009 IEEE 12th international conference on computer vision*, pages 349–356. IEEE.

[145] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. *CoRR*, 2016.

[146] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[147] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[148] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH*, 1996.

[149] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *ICCV*. IEEE, 2009.

[150] P. Goyal, D. Mahajan, A. Gupta, and I. Misra. Scaling and benchmarking self-supervised visual representation learning. In *ICCV*, 2019.

[151] D. Greenwood, I. Matthews, and S. Laycock. Joint learning of facial expression and head pose from speech. Interspeech, 2018.

[152] D. Griffin and J. Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984.

[153] C. Gu, J. J. Lim, P. Arbeláez, and J. Malik. Recognition using regions. In *CVPR*. IEEE, 2009.

[154] Y. Gucluturk, U. Guclu, R. van Lier, and M. A. J. van Gerven. Convolutional sketch inversion. In *ECCV*, 2016.

[155] J.-Y. Guillemaut, J. Kilner, and A. Hilton. Robust graph-cut scene segmentation and reconstruction for free-viewpoint video of complex dynamic scenes. In *ICCV*, 2009.

[156] R. Guo and D. Hoiem. Support surface prediction in indoor scenes. In *ICCV*, 2013.

[157] Y. Guo, N. C. Codella, L. Karlinsky, J. R. Smith, T. Rosing, and R. Feris. A new benchmark for evaluation of cross-domain few-shot learning. *arXiv:1912.07200*, 2019.

[158] A. Gupta, A. Efros, and M. Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. *ECCV*, 2010.

[159] S. Gupta, P. A. Arbeláez, R. B. Girshick, and J. Malik. Aligning 3D models to RGB-D images of cluttered scenes. In *CVPR*, 2015.

[160] S. Hallman and C. C. Fowlkes. Oriented edge forests for boundary detection. In *CVPR*, 2015.

[161] B. Hariharan, P. Arbelez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.

[162] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015.

[163] M. Haris, G. Shakhnarovich, and N. Ukita. Recurrent back-projection network for video super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3897–3906, 2019.

[164] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[165] N. Hasler, B. Rosenhahn, T. Thormahlen, M. Wand, J. Gall, and H.-P. Seidel. Markerless motion capture with unsynchronized moving cameras. In *CVPR*, 2009.

[166] J. V. Haxby, B. Horwitz, L. G. Ungerleider, J. M. Maisog, P. Pietrini, and C. L. Grady. The functional organization of human extrastriate cortex: a pet-rcbf study of selective attention to faces and locations. *Journal of Neuroscience*, 14(11):6336–6353, 1994.

[167] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics*, 2007.

[168] J. He, A. Lehrmann, J. Marino, G. Mori, and L. Sigal. Probabilistic video generation using holistic attribute control. In *ECCV*, 2018.

[169] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *ICCV*, 2017.

[170] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv:1512.03385*, 2015.

[171] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *ICCV*, 2015.

[172] D. O. Hebb. *The organization of behavior*. 1961.

[173] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009.

[174] P. Hedman, J. Philip, T. Price, J.-M. Frahm, G. Drettakis, and G. Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Trans. Graph.*, 2018.

[175] J. Heinly. *Toward Efficient and Robust Large-Scale Structure-from-Motion Systems*. PhD thesis, The University of North Carolina at Chapel Hill, 2015.

[176] J. Heinly, J. L. Schönberger, E. Dunn, and J.-M. Frahm. Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *CVPR*, 2015.

[177] P. Helber, B. Bischke, A. Dengel, and D. Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[178] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin. Image analogies. In *Proc. of the Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 2001.

[179] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NIPS*, 2017.

[180] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 1997.

[181] J. Hoffman, E. Tzeng, T. Park, J.-Y. Zhu, P. Isola, K. Saenko, A. A. Efros, and T. Darrell. Cycada: Cycle-consistent adversarial domain adaptation. In *ICML*, 2018.

[182] D. Hoiem, A. Efros, and M. Hebert. Putting objects in perspective. In *CVPR*, 2006.

[183] E. Hsu, K. Pulli, and J. Popović. Style translation for human motion. *ACM Trans. Graph.*, July 2005.

[184] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, 2017.

[185] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf. Temporally coherent completion of dynamic video. *ACM Transactions on Graphics (TOG)*, 35(6):196, 2016.

[186] L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[187] Q.-X. Huang and L. Guibas. Consistent shape maps via semidefinite programming. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, SGP '13. Eurographics Association, 2013.

[188] S. Huang and D. Ramanan. Expecting the unexpected: Training detectors for unusual pedestrians with adversarial imposters. In *CVPR*, 2017.

[189] X. Huang, Y. Li, O. Poursaeed, J. E. Hopcroft, and S. J. Belongie. Stacked generative adversarial networks. *CoRR*, 2016.

[190] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106, 1962.

[191] A. J. Hunt and A. W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *IEEE ICASSP*, 1996.

[192] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. *arXiv preprint arXiv:1504.01989*, 2015.

[193] A. Hyvärinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.*, volume 39. Springer Science & Business Media, 2009.

[194] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[195] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum. Label propagation for deep semi-supervised learning. In *CVPR*, 2019.

[196] P. Isola and C. Liu. Scene collaging: Analysis and synthesis of natural images with semantic layers. In *ICCV*, 2013.

[197] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[198] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Crisp boundary detection using pointwise mutual information. In *Computer Vision–ECCV 2014*, pages 799–814. Springer, 2014.

[199] H. Izadinia, B. C. Russell, A. Farhadi, M. D. Hoffman, and A. Hertzmann. Deep classifiers from image tags in the wild. In *Proceedings of the 2015 Workshop on Community-Organized Multimodal Mining: Opportunities for Novel Solutions*. ACM, 2015.

[200] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv:1408.5093*, 2014.

[201] Y. Jia, Y. Zhang, R. Weiss, Q. Wang, J. Shen, F. Ren, P. Nguyen, R. Pang, I. L. Moreno, Y. Wu, et al. Transfer learning from speaker verification to multispeaker text-to-speech synthesis. In *NeurIPS*, 2018.

[202] H. Jiang, D. Sun, V. Jampani, M.-H. Yang, E. Learned-Miller, and J. Kautz. Super slomo: High quality estimation of multiple intermediate frames for video interpolation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9000–9008, 2018.

[203] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, 2016.

[204] M. K. Johnson, K. Dale, S. Avidan, H. Pfister, W. T. Freeman, and W. Matusik. Cg2real: Improving the realism of computer generated images using a large collection of photographs. *IEEE Transactions on Visualization and Computer Graphics*, 2011.

[205] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. S. Godisart, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE TPAMI*, 2017.

[206] A. Joulin, L. van der Maaten, A. Jabri, and N. Vasilache. Learning visual features from large weakly supervised data. In *ECCB*. Springer, 2016.

[207] A. B. Kain, J.-P. Hosom, X. Niu, J. P. Van Santen, M. Fried-Oken, and J. Staehely. Improving the intelligibility of dysarthric speech. *Speech communication*, 2007.

[208] N. K. Kalantari, T.-C. Wang, and R. Ramamoorthi. Learning-based view synthesis for light field cameras. *ACM Trans. Graph.*, 2016.

[209] H. Kameoka, T. Kaneko, K. Tanaka, and N. Hojo. Acvae-vc: Non-parallel many-to-many voice conversion with auxiliary classifier variational autoencoder. *arXiv preprint arXiv:1808.05092*, 2018.

[210] T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka. A stereo machine for video-rate dense depth mapping and its new applications. In *CVPR*, 1996.

[211] T. Kanade and P. Narayanan. Historical perspectives on 4d virtualized reality. In *CVPR Workshops*, 2006.

[212] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE multimedia*, 1997.

[213] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018.

[214] A. Kanazawa, D. W. Jacobs, and M. Chandraker. Warpnet: Weakly supervised matching for single-view reconstruction. *CoRR*, 2016.

[215] T. Kaneko and H. Kameoka. Parallel-data-free voice conversion using cycle-consistent adversarial networks. *arXiv preprint arXiv:1711.11293*, 2017.

[216] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo. Cyclegan-vc2: Improved cyclegan-based non-parallel voice conversion. In *IEEE ICASSP*, 2019.

[217] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo. StarGAN-VC2: Rethinking Conditional Methods for StarGAN-Based Voice Conversion. *Proc. Interspeech*, 2019.

[218] T. Kaneko, H. Kameoka, K. Tanaka, and N. Hojo. Stargan-vc2: Rethinking conditional methods for stargan-based voice conversion. *Proc. Interspeech 2019*, 2019.

[219] N. Kanwisher, M. M. Chun, J. McDermott, and P. J. Ledden. Functional imaging of human visual recognition. *Cognitive Brain Research*, 5:55–67, 1996.

[220] N. Kanwisher, J. McDermott, and M. M. Chun. The fusiform face area: a module in human extrastriate cortex specialized for face perception. *Journal of neuroscience*, 17(11):4302–4311, 1997.

[221] K. Karsch, V. Hedau, D. Forsyth, and D. Hoiem. Rendering synthetic objects into legacy photographs. In *ACM Trans. Graph.* ACM, 2011.

[222] K. Karsch, K. Sunkavalli, S. Hadap, N. Carr, H. Jin, R. Fonte, M. Sittig, and D. Forsyth. Automatic scene inference for 3d object compositing. *ACM Trans. Graph.*

[223] A. Kay. The dynabook - past, present, and future. In *Proceedings of the ACM Conference on The History of Personal Workstations*. ACM, 1986.

[224] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen. Epic-fusion: Audio-visual temporal binding for egocentric action recognition. In *ICCV*, 2019.

[225] N. Kholgade, T. Simon, A. Efros, and Y. Sheikh. 3d object manipulation in a single photograph using stock 3d models. *ACM Trans. Graph.*, 2014.

[226] H. Kim, P. Garrido, A. Tewari, W. Xu, J. Thies, M. Niessner, P. Pérez, C. Richardt, M. Zollhöfer, and C. Theobalt. Deep video portraits. *ACM Transactions on Graphics (TOG)*, 37(4):1–14, 2018.

[227] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim. Learning to discover cross-domain relations with generative adversarial networks. In *ICML*, 2017.

[228] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[229] S. Kirby. Spontaneous evolution of linguistic structure-an iterated learning model of the emergence of regularity and irregularity. *IEEE Transactions on Evolutionary Computation*, 5(2):102–110, 2001.

[230] S. Kirby, T. Griffiths, and K. Smith. Iterated learning and the evolution of language. *Current opinion in neurobiology*, 28:108–114, 2014.

[231] J. J. Kivinen, C. K. Williams, N. Heess, and D. Technologies. Visual boundary prediction: A deep neural prediction network and quality dissection. In *AISTATS*, volume 1, page 9, 2014.

[232] J. J. Koenderink. *Solid Shape*. MIT Press, Cambridge, MA, USA, 1990.

[233] I. Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6129–6138, 2017.

[234] P. KR, R. Mukhopadhyay, J. Philip, A. Jha, V. Namboodiri, and C. Jawahar. Towards automatic face-to-face translation. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1428–1436, 2019.

[235] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011.

[236] D. J. Kravitz, C. S. Peng, and C. I. Baker. Real-world scene representations in high-level visual cortex: it's the spaces more than the places. *Journal of Neuroscience*, 31(20):7322–7333, 2011.

[237] V. Krishnan and D. Ramanan. Tinkering under the hood: Interactive zero-shot learning with net surgery. *arXiv preprint arXiv:1612.04901*, 2016.

[238] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[239] L. Ladicky, B. Zeisl, and M. Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014.

[240] J.-F. Lalonde, D. Hoiem, A. A. Efros, C. Rother, J. Winn, and A. Criminisi. Photo clip art. *ACM Trans. Graph.*, 2007.

[241] G. Larsson, M. Maire, and G. Shakhnarovich. Learning representations for automatic colorization. In *ECCV*, 2016.

[242] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[243] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[244] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. P. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-realistic single image super-resolution using a generative adversarial network. *CoRR*, 2016.

[245] D.-H. Lee. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, 2013.

[246] K. Lee, S. Maji, A. Ravichandran, and S. Soatto. Meta-learning with differentiable convex optimization. In *CVPR*, 2019.

[247] M. Levoy and P. Hanrahan. Light field rendering. In *Annual Conference on Computer Graphics and Interactive Techniques*. ACM, 1996.

[248] X. Li, Q. Sun, Y. Liu, Q. Zhou, S. Zheng, T.-S. Chua, and B. Schiele. Learning to self-train for semi-supervised few-shot classification. In *NeurIPS*, 2019.

[249] Z. Li, S. Niklaus, N. Snavely, and O. Wang. Neural scene flow fields for space-time view synthesis of dynamic scenes. *arXiv preprint arXiv:2011.13084*, 2020.

[250] Z. Li and D. Hoiem. Learning without forgetting. *IEEE TPAMI*, 2017.

[251] J. C. R. Licklider and R. W. Taylor. The computer as a communication device. *Science and Technology*, 1968.

[252] J. Lim, C. Zitnick, and P. Dollár. Sketch tokens: A learned mid-level representation for contour and object detection. In *CVPR*, 2013.

[253] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, 2014.

[254] C. Liu, H. Shum, and W. T. Freeman. Face hallucination: Theory and practice. *IJCV*, 2007.

[255] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2368–2382, 2011.

[256] C. Liu, J. Yuen, and A. Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011.

[257] M.-Y. Liu, X. Huang, J. Yu, T.-C. Wang, and A. Mallya. Generative adversarial networks for image and video synthesis: Algorithms and applications. *arXiv preprint arXiv:2008.02793*, 2020.

[258] S. J. Liu, M. Agrawala, S. DiVerdi, and A. Hertzmann. View-dependent video textures for 360° video. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19. ACM, 2019.

[259] W. Liu, A. Rabinovich, and A. C. Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[260] Y.-L. Liu, W.-S. Lai, M.-H. Yang, Y.-Y. Chuang, and J.-B. Huang. Learning to see through obstructions with layered decomposition. In *CVPR*, 2020.

[261] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *ICCV*, 2015.

[262] Z. Liu, R. A. Yeh, X. Tang, Y. Liu, and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4463–4471, 2017.

[263] S. Lombardi, J. Saragih, T. Simon, and Y. Sheikh. Deep appearance models for face rendering. *ACM Trans. Graph.*, 2018.

[264] S. Lombardi, T. Simon, J. Saragih, G. Schwartz, A. Lehrmann, and Y. Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Trans. Graph.*, 2019.

[265] G. Long, L. Kneip, J. M. Alvarez, H. Li, X. Zhang, and Q. Yu. Learning image matching by simply watching video. In *European Conference on Computer Vision*, pages 434–450. Springer, 2016.

[266] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.

[267] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional models for semantic segmentation. In *CVPR*, 2015.

[268] J. Long, N. Zhang, and T. Darrell. Do convnets learn correspondence? In *NeurIPS*, 2014.

[269] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. Smpl: A skinned multi-person linear model. *TOG*, 2015.

[270] E. Lu, F. Cole, T. Dekel, W. Xie, A. Zisserman, D. Salesin, W. T. Freeman, and M. Rubinstein. Layered neural rendering for retiming people in video, 2020.

[271] Y. Lu, S. Singhal, F. Strub, O. Pietquin, and A. Courville. Countering language drift with seeded iterated learning. *arXiv:2003.12694*, 2020.

[272] Y. Lu, S. Singhal, F. Strub, O. Pietquin, and A. Courville. Supervised seeded iterated learning for interactive language learning. In *Proc. of EMNLP*, 2020.

[273] J. M. M. Holschneider, R. Kronland-Martinet and P. Tchamitchian. A real-time algorithm for signal analysis with the help of the wavelet transform. *In Wavelets, Time-Frequency Methods and Phase Space*, pages 289–297, 1989.

[274] E. A. Maguire, N. Burgess, J. G. Donnett, R. S. Frackowiak, C. D. Frith, and J. O'Keefe. Knowing where and getting there: a human navigation network. *Science*, 280(5365):921–924, 1998.

[275] D. Mahajan, F.-C. Huang, W. Matusik, R. Ramamoorthi, and P. Belhumeur. Moving gradients: a path-based method for plausible image interpolation. *ACM Transactions on Graphics (TOG)*, 28(3):1–11, 2009.

[276] A. Mahendran and A. Vedaldi. Understanding deep image representations by inverting them. In *CVPR*, 2015.

[277] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. *International Journal of Computer Vision*, 120(3):233–255, 2016.

[278] B. Z. Mahon, S. C. Milleville, G. A. Negri, R. I. Rumiati, A. Caramazza, and A. Martin. Action-related properties shape object representations in the ventral stream. *Neuron*, 55(3):507–520, 2007.

[279] R. Malach, J. Reppas, R. Benson, K. Kwong, H. Jiang, W. Kennedy, P. Ledden, T. Brady, B. Rosen, and R. Tootell. Object-related activity revealed by functional magnetic resonance imaging in human occipital cortex. *Proceedings of the National Academy of Sciences*, 92(18):8135–8139, 1995.

[280] T. Malisiewicz and A. A. Efros. Beyond categories: The visual memex model for reasoning about object relationships. In *NeurIPS*, 2009.

[281] D. Marr. Vision: A computational investigation into the human representation and processing of visual information. 1982.

[282] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *TPAMI*, 26(5), 2004.

[283] O. Matan, C. J. Burges, Y. LeCun, and J. S. Denker. Multi-digit recognition using a space displacement neural network. In *NIPS*, pages 488–495, 1991.

[284] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-based visual hulls. In *ACM Trans. Graph.*, 2000.

[285] M. Maximov, I. Elezi, and L. Leal-Taixé. Ciagan: Conditional identity anonymization generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5447–5456, 2020.

[286] L. MCMILLAN. An image-based approach to three-dimensional computer graphics. *Ph. D. Dissertation, UNC Computer Science*, 1999.

[287] L. McMillan and G. Bishop. Plenoptic modeling: An image-based rendering system. In *SIGGRAPH*, 1995.

[288] S. Meister, J. Hur, and S. Roth. UnFlow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI*, 2018.

[289] S. Menon, A. Damian, S. Hu, N. Ravi, and C. Rudin. Pulse: Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2445, 2020.

[290] M. Meshry, D. B. Goldman, S. Khamis, H. Hoppe, R. Pandey, N. Snavely, and R. Martin-Brualla. Neural rerendering in the wild. In *CVPR*, 2019.

[291] A. Meyer and F. Neyret. Interactive volumetric textures. In *Eurographics Workshop on Rendering Techniques*, pages 157–168. Springer, 1998.

[292] T. Michaeli and M. Irani. Blind deblurring using internal patch recurrence. In *European conference on computer vision*, pages 783–798. Springer, 2014.

[293] B. Mildenhall, P. P. Srinivasan, R. Ortiz-Cayon, N. K. Kalantari, R. Ramamoorthi, R. Ng, and A. Kar. Local light field fusion: Practical view synthesis with prescriptive sampling guidelines. *ACM Transactions on Graphics (TOG)*, 2019.

[294] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[295] D. Milner and M. Goodale. *The visual brain in action*, volume 27. OUP Oxford, 2006.

[296] I. Misra, A. Shrivastava, and M. Hebert. Watch and learn: Semi-supervised learning of object detectors from videos. In *CVPR*, 2015.

[297] I. Misra and L. van der Maaten. Self-supervised learning of pretext-invariant representations. In *CVPR*, 2020.

[298] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and Learn: Unsupervised Learning using Temporal Order Verification. In *ECCV*, 2016.

[299] S. H. Mohammadi and T. Kim. One-shot voice conversion with disentangled representations by leveraging phonetic posteriorgrams. *Proc. Interspeech*, 2019.

[300] S. P. Mohanty, D. P. Hughes, and M. Salathé. Using deep learning for image-based plant disease detection. *Frontiers in plant science*, 7:1419, 2016.

[301] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, 2015.

[302] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, 2014.

[303] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *Computer Vision–ECCV 2010*, pages 57–70. Springer, 2010.

[304] A. Nagrani, S. Albanie, and A. Zisserman. Seeing voices and hearing faces: Cross-modal biometric matching. In *CVPR*, 2018.

[305] K. Nakamura, T. Toda, H. Saruwatari, and K. Shikano. Speaking-aid systems using gmm-based voice conversion for electrolaryngeal speech. *Speech Commun.*, 2012.

[306] T. Nakashika, T. Takiguchi, and Y. Ariki. High-order sequence modeling using speaker-dependent recurrent temporal restricted boltzmann machines for voice conversion. In *Proc. Interspeech*, 2014.

[307] A. Newell and J. Deng. How useful is self-supervised pretraining for visual tasks? In *CVPR*, 2020.

[308] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez. Towards fast, generic video inpainting. In *Proceedings of the 10th European Conference on Visual Media Production*, pages 1–8, 2013.

[309] R. Ng, M. Levoy, M. Brédif, G. Duval, M. Horowitz, and P. Hanrahan. *Light field photography with a hand-held plenoptic camera*. PhD thesis, Stanford University, 2005.

[310] K. Nigam, A. K. McCallum, S. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using em. *Machine learning*, 39(2-3):103–134, 2000.

[311] S. Niklaus, L. Mai, and F. Liu. Video frame interpolation via adaptive convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 670–679, 2017.

[312] S. Niklaus, L. Mai, J. Yang, and F. Liu. 3d ken burns effect from a single image. *ACM Transactions on Graphics (TOG)*, 38(6):1–15, 2019.

[313] M.-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[314] M. Nishimura, S. Scherf, and M. Behrmann. Development of object recognition in humans. *F1000 biology reports*, 1, 2009.

[315] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *ICCV*, 2015.

[316] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *ECCV*, 2016.

[317] T.-H. Oh, T. Dekel, C. Kim, I. Mosseri, W. T. Freeman, M. Rubinstein, and W. Matusik. Speech2face: Learning the face behind a voice. In *CVPR*, 2019.

[318] C. Olah, A. Satyanarayan, I. Johnson, S. Carter, L. Schubert, K. Ye, and A. Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

[319] M. M. Oliveira and G. Bishop. Image-based objects. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 191–198, 1999.

[320] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

[321] A. V. Oppenheim, A. S. Willsky, and S. H. Nawab. *Signals & Systems (Second Edition)*. Prentice-Hall, Inc., 1996.

[322] M. Oswald and D. Cremers. A convex relaxation approach to space time multi-view 3d reconstruction. In *ICCVW*, 2013.

[323] A. Owens and A. A. Efros. Audio-visual scene analysis with self-supervised multisensory features. *ECCV*, 2018.

[324] A. Owens, J. Wu, J. H. McDermott, W. T. Freeman, and A. Torralba. Ambient sound provides supervision for visual learning. In *ECCV*, 2016.

[325] J. Papon and M. Schoeler. Semantic pose using deep networks trained on synthetic RGB-D. In *ICCV*, 2015.

[326] K. Park, U. Sinha, J. T. Barron, S. Bouaziz, D. B. Goldman, S. M. Seitz, and R.-M. Brualla. Deformable neural radiance fields. *arXiv preprint arXiv:2011.12948*, 2020.

[327] S. Park, T. Brady, M. Greene, and A. Oliva. Disentangling scene content from its spatial boundary: complementary roles for the ppa and loc in representing real-world scenes. 2011.

[328] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, 2019.

[329] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012.

[330] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016.

[331] D. Paul, Y. Pantazis, and Y. Stylianou. Non-parallel voice conversion using weighted generative adversarial networks. *Proc. Interspeech 2019*, 2019.

[332] P. H. Pinheiro and R. Collobert. Recurrent convolutional neural networks for scene parsing. In *ICML*, 2014.

[333] J. F. Pitrelli, R. Bakis, E. M. Eide, R. Fernandez, W. Hamza, and M. A. Picheny. The ibm expressive text-to-speech synthesis system for american english. *IEEE Transactions on Audio, Speech, and Language Processing*, 2006.

[334] J. C. Platt and R. Wolf. Postal address block location using a convolutional locator network. In *NIPS*, 1993.

[335] A. Polyak and L. Wolf. Attention-based wavenet autoencoder for universal voice conversion. In *IEEE ICASSP*, 2019.

[336] A. Puce, T. Allison, J. C. Gore, and G. McCarthy. Face-sensitive regions in human extrastriate cortex studied by functional mri. *Journal of neurophysiology*, 74(3):1192–1199, 1995.

[337] A. Pumarola, E. Corona, G. Pons-Moll, and F. Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. *arXiv preprint arXiv:2011.13961*, 2020.

[338] S. Purushwalkam, T. Ye, S. Gupta, and A. Gupta. Aligning videos in space and time. *arXiv preprint arXiv:2007.04515*, 2020.

[339] X. Qi, Q. Chen, J. Jia, and V. Koltun. Semi-parametric image synthesis. In *CVPR*, 2018.

[340] X. Qi, R. Liao, Z. Liu, R. Urtasun, and J. Jia. Geonet: Geometric neural network for joint depth and surface normal estimation. In *CVPR*, 2018.

[341] K. Qian, Y. Zhang, S. Chang, X. Yang, and M. Hasegawa-Johnson. Autovc: Zero-shot voice style transfer with only autoencoder loss. In *ICML*, 2019.

[342] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *CoRR*, 2015.

[343] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever. Improving language understanding by generative pre-training, 2018.

[344] I. Radosavovic, P. Dollár, R. Girshick, G. Gkioxari, and K. He. Data distillation: Towards omni-supervised learning. In *CVPR*, 2018.

[345] T. Raiko, H. Valpola, and Y. LeCun. Deep learning made easier by linear transformations in perceptrons. In *AISTATS*, volume 22, pages 924–932, 2012.

[346] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *ICML*, 2007.

[347] D. Ramanan. Learning to parse images of articulated bodies. In *NIPS*. 2007.

[348] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[349] D. Rao, F. Visin, A. Rusu, R. Pascanu, Y. W. Teh, and R. Hadsell. Continual unsupervised representation learning. In *NeurIPS*, 2019.

[350] S. Ravi and H. Larochelle. Optimization as a model for few-shot learning. 2016.

[351] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. icarl: Incremental classifier and representation learning. In *CVPR*, 2017.

[352] R. Reddy. *Teleportation, Time Travel, and Immortality*. Springer New York, 1999.

[353] L. Ren, A. Patrick, A. A. Efros, J. K. Hodgins, and J. M. Rehg. A data-driven approach to quantifying natural human motion. *ACM Trans. Graph.*, 2005.

[354] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel. Meta-learning for semi-supervised few-shot classification. *arXiv:1803.00676*, 2018.

[355] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *ICCV*, 2017.

[356] G. Riegler and V. Koltun. Free view synthesis. In *ECCV*, 2020.

[357] L. Roberts. Machine perception of 3-D solids. In *PhD. Thesis*, 1965.

[358] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015.

[359] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, 2015.

[360] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. FaceForensics++: Learning to detect manipulated facial images. In *International Conference on Computer Vision (ICCV)*, 2019.

[361] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 2004.

[362] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015.

[363] B. Russell, A. Torralba, K. Murphy, and W. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 2008.

[364] B. Russell, A. A. Efros, J. Sivic, W. T. Freeman, and A. Zisserman. Segmenting scenes by matching image composites. In *NIPS*, 2009.

[365] B. Russell, W. T. Freeman, A. A. Efros, J. Sivic, and A. Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In *CVPR*, 2006.

[366] C. Russell, P. Kohli, P. H. Torr, et al. Associative hierarchical crfs for object class image segmentation. In *ICCV*. IEEE, 2009.

[367] M. Saito, E. Matsumoto, and S. Saito. Temporal generative adversarial nets with singular value clipping. In *ICCV*, 2017.

[368] A. Saxena, S. H. Chung, and A. Y. Ng. 3-d depth reconstruction from a single still image. *IJCV*, 76(1), 2008.

[369] D. Scharstein. Stereo vision for view synthesis. In *Proceedings CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 852–858. IEEE, 1996.

[370] A. Schödl, R. Szeliski, D. H. Salesin, and I. Essa. Video textures. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 489–498, 2000.

[371] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *CVPR*, 2016.

[372] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016.

[373] C. D. Schunn and K. Dunbar. Priming, analogy, and awareness in complex reasoning. *Memory & Cognition*, 1996.

[374] H. Scudder. Probability of error of some adaptive pattern-recognition machines. *IEEE Trans. IT*, 1965.

[375] S. M. Seitz and C. R. Dyer. View morphing. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pages 21–30, 1996.

[376] A. Senocak, T.-H. Oh, J. Kim, M.-H. Yang, and I. So Kweon. Learning to localize sound source in visual scenes. In *CVPR*, 2018.

[377] J. Sergent, S. Ohta, and B. MACDONALD. Functional neuroanatomy of face and object processing: a positron emission tomography study. *Brain*, 115(1):15–36, 1992.

[378] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In *CVPR*, 2013.

[379] J. Serrà, S. Pascual, and C. Segura. Blow: a single-scale hyperconditioned flow for non-parallel raw-audio voice conversion. In *NeurIPS*, 2019.

[380] J. Shade, S. Gortler, L.-w. He, and R. Szeliski. Layered depth images. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 231–242, 1998.

[381] T. R. Shaham, T. Dekel, and T. Michaeli. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4570–4580, 2019.

[382] E. Shechtman, Y. Caspi, and M. Irani. Increasing space-time resolution in video. In *European Conference on Computer Vision*, pages 753–768. Springer, 2002.

[383] E. Shechtman, Y. Caspi, and M. Irani. Space-time super-resolution. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(4):531–545, 2005.

[384] J. Shen, R. Pang, R. J. Weiss, M. Schuster, N. Jaitly, Z. Yang, Z. Chen, Y. Zhang, Y. Wang, R. Skerrv-Ryan, et al. Natural tts synthesis by conditioning wavenet on mel spectrogram predictions. In *IEEE ICASSP*, 2018.

[385] M.-L. Shih, S.-Y. Su, J. Kopf, and J.-B. Huang. 3d photography using context-aware layered depth inpainting. In *CVPR*, 2020.

[386] E. Shlizerman, L. Dery, H. Schoen, and I. Kemelmacher-Shlizerman. Audio to body dynamics. In *CVPR*, 2018.

[387] A. Shocher, S. Bagon, P. Isola, and M. Irani. Ingan: Capturing and remapping the" dna" of a natural image. *arXiv preprint arXiv:1812.00231*, 2018.

[388] A. Shocher, N. Cohen, and M. Irani. "zero-shot" super-resolution using deep internal learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3118–3126, 2018.

[389] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost for image understanding: Multi-class object recognition and segmentation by jointly modeling texture, layout, and context. *Int. Journal of Computer Vision (IJCV)*, January 2009.

[390] A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Transaction of Graphics (TOG)*, 2011.

[391] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017.

[392] H. Shum and S. B. Kang. Review of image-based rendering techniques. In *Visual Communications and Image Processing*, 2000.

[393] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[394] D. L. Silver, Q. Yang, and L. Li. Lifelong machine learning systems: Beyond learning algorithms. In *2013 AAAI spring symposium series*, 2013.

[395] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[396] S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012.

[397] S. N. Sinha, D. Steedly, R. Szeliski, M. Agrawala, and M. Pollefeys. Interactive 3d architectural modeling from unordered photo collections. In *ACM Trans. Graph.* ACM, 2008.

[398] V. Sitzmann, M. Zollhöfer, and G. Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Neurips*, 2019.

[399] B. M. Smith, L. Zhang, J. Brandt, Z. L. Lin, and J. Yang. Exemplar-based face parsing. In *CVPR*, 2013.

[400] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3d. *ACM Trans. Graph.*, 2006.

[401] J. Snell, K. Swersky, and R. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[402] P. P. Srinivasan, R. Tucker, J. T. Barron, R. Ramamoorthi, R. Ng, and N. Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *CVPR*, 2019.

[403] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, 15(1), 2014.

[404] G. M. Stevens and C. Doyle. *Privacy: An overview of federal statutes governing wiretapping and electronic eavesdropping*. Lulu. com, 2011.

[405] H. Su, C. Qi, Y. Li, and L. Guibas. Render for CNN: Viewpoint Estimation in Images Using CNNs Trained with Rendered 3D Model Views. In *ICCV*, 2015.

[406] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[407] L. Sun, S. Kang, K. Li, and H. Meng. Voice conversion using deep bidirectional long short-term memory based recurrent neural networks. In *IEEE ICASSP*, 2015.

[408] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales. Learning to compare: Relation network for few-shot learning. In *CVPR*, 2018.

[409] I. E. Sutherland. Sketchpad: A man-machine graphical communication system. In *Proceedings of the Spring Joint Computer Conference*, AFIPS '63 (Spring). ACM, 1963.

[410] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *NIPS*, 2014.

[411] S. Suwajanakorn, S. M. Seitz, and I. Kemelmacher-Shlizerman. Synthesizing obama: Learning lip sync from audio. *ACM Trans. Graph.*, 2017.

[412] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[413] M. Tan and Q. V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv:1905.11946*, 2019.

[414] A. Taneja, L. Ballan, and M. Pollefeys. Modeling dynamic scenes recorded with freely moving cameras. In *ACCV*, 2010.

[415] M. F. Tappen and C. Liu. A bayesian approach to alignment-based image hallucination. In *ECCV*, 2012.

[416] S. Taylor, T. Kim, Y. Yue, M. Mahler, J. Krahe, A. G. Rodriguez, J. Hodgins, and I. Matthews. A deep learning approach for generalized speech animation. *ACM Trans. Graph.*, 2017.

[417] A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, et al. State of the art on neural rendering. *Eurographics*, 2020.

[418] The-New-York-Times. Turning the super bowl into a game of pixels. January 2001.

[419] J. Thies, M. Zollhofer, M. Niessner, L. Valgaerts, M. Stamminger, and C. Theobalt. Real-time expression transfer for facial reenactment. *ACM Trans. Graph.*, 2015.

[420] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Niessner. Face2face: Real-time face capture and reenactment of rgb videos. In *CVPR*, 2016.

[421] S. Thrun. Is learning the n-th thing any easier than learning the first? In *NeurIPS*, 1996.

[422] S. Thrun. Lifelong learning algorithms. In *Learning to learn*, pages 181–209. Springer, 1998.

[423] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *Computer Vision–ECCV 2010*, pages 352–365. Springer, 2010.

[424] P. L. Tobing, Y.-C. Wu, T. Hayashi, K. Kobayashi, and T. Toda. Non-parallel voice conversion with cyclic variational autoencoder. *Proc. Interspeech 2019*, 2019.

[425] T. Toda, A. W. Black, and K. Tokuda. Voice conversion based on maximum-likelihood estimation of spectral parameter trajectory. *IEEE Transactions on Audio, Speech, and Language Processing*, 2007.

[426] A. Torralba and A. A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011.

[427] E. Tretschk, A. Tewari, V. Golyanik, M. Zollhöfer, C. Lassner, and C. Theobalt. Non-rigid neural radiance fields: Reconstruction and novel view synthesis of a deforming scene from monocular video. *arXiv preprint arXiv:2012.12247*, 2020.

[428] P. Tschandl, C. Rosendahl, and H. Kittler. The ham10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions. *Scientific data*, 2018.

[429] H.-Y. Tseng, H.-Y. Lee, J.-B. Huang, and M.-H. Yang. Cross-domain few-shot classification via learned feature-wise transformation. *arXiv:2001.08735*, 2020.

[430] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3d brain image segmentation. *TPAMI*, 32(10), 2010.

[431] R. Tucker and N. Snavely. Single-view view synthesis with multiplane images. In *CVPR*, 2020.

[432] S. Tulsiani, R. Tucker, and N. Snavely. Layer-structured 3d scene inference via view synthesis. In *ECCV*, 2018.

[433] S. Tulyakov, M.-Y. Liu, X. Yang, and J. Kautz. Mocogan: Decomposing motion and content for video generation. In *CVPR*, 2018.

[434] R. Tyleček and R. Šára. Spatial pattern templates for recognition of objects with regular structure. In *Proc. German Conference on Pattern Recognition (GCPR)*, Saarbrucken, Germany, 2013.

[435] D. Ulyanov, A. Vedaldi, and V. Lempitsky. Deep image prior. *International Journal of Computer Vision*, 2020.

[436] A. van den Oord, O. Vinyals, et al. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315, 2017.

[437] C. Veaux, J. Yamagishi, and K. Macdonald. Superseded - CSTR VCTK Corpus: English multi-speaker corpus for CSTR voice cloning toolkit. 2016.

[438] S. Vedula, S. Baker, and T. Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Trans. Graph.*, 2005.

[439] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. *arXiv:1704.05831*, 2017.

[440] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, et al. Matching networks for one shot learning. In *NeurIPS*, 2016.

[441] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. In *ACM SIGGRAPH 2008*. 2008.

[442] M. Vo, S. G. Narasimhan, and Y. Sheikh. Spatiotemporal bundle adjustment for dynamic 3d reconstruction. In *CVPR*, 2016.

[443] M. Vo, E. Yumer, K. Sunkavalli, S. Hadap, Y. Sheikh, and S. Narasimhan. Automatic adaptation of person association for multiview tracking in group activities. *arXiv preprint arXiv:1805.08717*, 2018.

[444] M. P. Vo, Y. A. Sheikh, and S. G. Narasimhan. Spatiotemporal bundle adjustment for dynamic 3d human reconstruction in the wild. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[445] J. Walker, C. Doersch, A. Gupta, and M. Hebert. An uncertain future: Forecasting from variational autoencoders. In *ECCV*, 2016.

[446] J. Walker, K. Marino, A. Gupta, and M. Hebert. The pose knows: Video forecasting by generating pose futures. In *ICCV*, 2017.

[447] M. Wallingford, A. Kusupati, K. Alizadeh-Vahid, A. Walsman, A. Kembhavi, and A. Farhadi. In the wild: From ml models to pragmatic ml systems. *arXiv:2007.02519*, 2020.

[448] S.-Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros. Cnn-generated images are surprisingly easy to spot...for now. In *CVPR*, 2020.

[449] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, G. Liu, A. Tao, J. Kautz, and B. Catanzaro. Video-to-video synthesis. In *NeurIPS*, 2018.

[450] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *arXiv preprint arXiv:1711.11585*, 2017.

[451] T.-C. Wang, J.-Y. Zhu, N. K. Kalantari, A. A. Efros, and R. Ramamoorthi. Light field video capture using a learning-based hybrid imaging system. *ACM Transactions on Graphics (TOG)*, 2017.

[452] X. Wang, D. Fouhey, and A. Gupta. Designing deep networks for surface normal estimation. In *CVPR*, 2015.

[453] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *ICCV*, 2015.

[454] X. Wang and A. Gupta. Generative image modeling using style and structure adversarial networks. In *ECCV*, 2016.

[455] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2566–2576, 2019.

[456] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[457] Y.-X. Wang, R. Girshick, M. Hebert, and B. Hariharan. Low-shot learning from imaginary data. In *CVPR*, 2018.

[458] Y. Wang, R. Skerry-Ryan, D. Stanton, Y. Wu, R. J. Weiss, N. Jaitly, Z. Yang, Y. Xiao, Z. Chen, S. Bengio, et al. Tacotron: Towards end-to-end speech synthesis. *Proc. Interspeech*, 2017.

[459] C. Wei, K. Shen, Y. Chen, and T. Ma. Theoretical analysis of self-training with deep networks on unlabeled data. *arXiv:2010.03622*, 2020.

[460] L. Wei, Q. Huang, D. Ceylan, E. Vouga, and H. Li. Dense human body correspondences using convolutional networks. *CoRR*, 2015.

[461] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016.

[462] S.-E. Wei, J. Saragih, T. Simon, A. W. Harley, S. Lombardi, M. Perdoch, A. Hypes, D. Wang, H. Badino, and Y. Sheikh. Vr facial animation via multiview image translation. *ACM Transactions on Graphics (TOG)*, 2019.

[463] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[464] D. Wertheimer and B. Hariharan. Few-shot learning with localization in realistic settings. In *CVPR*, 2019.

[465] Y. Wexler, E. Shechtman, and M. Irani. Space-time completion of video. *IEEE Transactions on pattern analysis and machine intelligence*, 29(3):463–476, 2007.

[466] O. Wiles, G. Gkioxari, R. Szeliski, and J. Johnson. Synsin: End-to-end view synthesis from a single image. In *CVPR*, 2020.

[467] S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

[468] J. Wu, C. Zhang, T. Xue, W. T. Freeman, and J. B. Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NeurIPS*, 2016.

[469] Y. Wu and K. He. Group normalization. In *ECCV*, 2018.

[470] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3D ShapeNets: A deep representation for volumetric shape modeling. 2015.

[471] Y. Xia, D. He, T. Qin, L. Wang, N. Yu, T. Liu, and W. Ma. Dual learning for machine translation. *CoRR*, 2016.

[472] W. Xian, J.-B. Huang, J. Kopf, and C. Kim. Space-time neural irradiance fields for free-viewpoint video. *arXiv preprint arXiv:2011.12950*, 2020.

[473] X. Xiang, Y. Tian, Y. Zhang, Y. Fu, J. P. Allebach, and C. Xu. Zooming slow-mo: Fast and accurate one-stage space-time video super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3370–3379, 2020.

[474] R. Xiaofeng and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012.

[475] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le. Self-training with noisy student improves imagenet classification. In *CVPR*, 2020.

[476] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv:1611.05431*, 2016.

[477] S. Xie, X. Huang, and Z. Tu. Convolutional pseudo-prior for structured labeling. *arXiv preprint arXiv:1511.07409*, 2015.

[478] S. Xie and Z. Tu. Holistically-nested edge detection. In *ICCV*, 2015.

[479] R. Xu, X. Li, B. Zhou, and C. C. Loy. Deep flow-guided video inpainting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[480] T. Xue, B. Chen, J. Wu, D. Wei, and W. T. Freeman. Video enhancement with task-oriented flow. *International Journal of Computer Vision (IJCV)*, 127(8):1106–1125, 2019.

[481] I. Z. Yalniz, H. Jégou, K. Chen, M. Paluri, and D. Mahajan. Billion-scale semi-supervised learning for image classification. *arXiv:1905.00546*, 2019.

[482] G. Yang, J. Manela, M. Happold, and D. Ramanan. Hierarchical deep stereo matching on high-resolution images. In *CVPR*, 2019.

[483] G. Yang and D. Ramanan. Volumetric correspondence networks for optical flow. In *Advances in Neural Information Processing Systems*, pages 793–803, 2019.

[484] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Trans. Pattern Anal. Mach. Intell.*

[485] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*. IEEE, 2012.

[486] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *33rd annual meeting of the association for computational linguistics*, pages 189–196, 1995.

[487] J. S. Yoon, K. Kim, O. Gallo, H. S. Park, and J. Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *CVPR*, 2020.

[488] A. Yu and K. Grauman. Fine-Grained Visual Comparisons with Local Learning. In *CVPR*, 2014.

[489] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016.

[490] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *ECCV*, 2014.

[491] H. Zen, K. Tokuda, and A. W. Black. Statistical parametric speech synthesis. *Speech Communication*, 2009.

[492] H. Zhang, T. Xu, and H. Li. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *ICCV*, 2017.

[493] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. *ECCV*, 2016.

[494] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[495] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros. Real-time user-guided image colorization with learned deep priors. *ACM Trans. Graph.*, 2017.

[496] Y. Zhang, K. Lee, and H. Lee. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. In *ICML*, 2016.

[497] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[498] S. Zhao, T. H. Nguyen, H. Wang, and B. Ma. Fast learning for non-parallel many-to-many voice conversion with residual star generative adversarial networks. *Proc. Interspeech 2019*, 2019.

[499] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *ICCV*, 2015.

[500] B. Zhou, A. Khosla, À. Lapedriza, A. Oliva, and A. Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2014.

[501] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE TPAMI*, 2017.

[502] H. Zhou, Y. Liu, Z. Liu, P. Luo, and X. Wang. Talking face generation by adversarially disentangled audio-visual representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9299–9306, 2019.

[503] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017.

[504] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In *CVPR*, 2016.

[505] T. Zhou, Y. J. Lee, S. X. Yu, and A. A. Efros. Flowweb: Joint image set alignment by weaving consistent, pixel-wise correspondences. *CVPR*, 2015.

[506] T. Zhou, R. Tucker, J. Flynn, G. Fyffe, and N. Snavely. Stereo magnification: learning view synthesis using multiplane images. *ACM Trans. Graph.*, 2018.

[507] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *ECCV*, 2016.

[508] Z.-H. Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 5(1):44–53, 2018.

[509] H. Zhu, A. Zheng, H. Huang, and R. He. High-resolution talking face generation via mutual information approximation. *arXiv preprint arXiv:1812.06589*, 2018.

[510] J.-Y. Zhu, P. Krähenbühl, E. Shechtman, and A. A. Efros. Generative visual manipulation on the natural image manifold. In *ECCV*, 2016.

[511] J.-Y. Zhu, Y. J. Lee, and A. A. Efros. Averageexplorer: Interactive exploration and alignment of visual data collections. *ACM Transactions on Graphics (SIGGRAPH 2014)*, 33(4), 2014.

[512] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, 2017.

[513] J.-Y. Zhu, R. Zhang, D. Pathak, T. Darrell, A. A. Efros, O. Wang, and E. Shechtman. Toward multimodal image-to-image translation. In *NIPS*, 2017.

[514] X. J. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[515] C. L. Zitnick, S. B. Kang, M. Uyttendaele, S. Winder, and R. Szeliski. High-quality video view interpolation using a layered representation. *ACM Trans. Graph.*, 2004.

[516] X. T. Y. L. Ziwei Liu, Raymond Yeh and A. Agarwala. Video frame synthesis using deep voxel flow. In *Proceedings of International Conference on Computer Vision (ICCV)*, October 2017.

[517] B. Zoph, G. Ghiasi, T.-Y. Lin, Y. Cui, H. Liu, E. D. Cubuk, and Q. V. Le. Rethinking pre-training and self-training. *arXiv:2006.06882*, 2020.

[518] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, 2018.