

Active Vision: Autonomous Aerial Cinematography with Learned Artistic Decision-Making

Rogério Bonatti

CMU-RI-TR-21-16

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:

Sebastian Scherer, CMU RI (chair)
Jessica Hodgins, CMU RI
Oliver Kroemer, CMU RI
Ashish Kapoor, Microsoft Research
Nathan Ratliff, Nvidia

*Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Robotics*

May 2021

© Copyright by Rogerio Bonatti, 2021.
All rights reserved.

Abstract

Aerial cinematography is revolutionizing industries that require live and dynamic camera viewpoints such as entertainment, sports, and security. Fundamentally, it is a tool with immense potential to improve human creativity, expressiveness, and sharing of experiences. However, safely piloting a drone while filming a moving target in the presence of obstacles is immensely taxing, often requiring multiple highly trained human operators to safely control a single vehicle. Our research focus is to build autonomous systems that can empower any individual with the full artistic capabilities of aerial cameras. We develop a system for active vision: in other words, one that not only passively processes the incoming sensor feed, but on the contrary, actively reasons about the cinematographic quality of viewpoints and safely generates sequences of shots. The theory and systems developed in this work can impact video generation for both real-world and simulated environments, such as professional and amateur movie-making, videogames, and virtual reality.

First, we formalize the theory behind the aerial filming problem by incorporating cinematography guidelines into robot motion planning. We describe the problem in terms of its principal cost functions, and develop an efficient trajectory optimization framework for executing arbitrary types of shots while avoiding collisions and occlusions with obstacles.

Second, we propose and develop a system for aerial cinematography *in the wild*. We combine several components into a real-time framework: vision-based target estimation, 3D signed-distance mapping for collision and occlusion avoidance, and trajectory optimization for camera motion. We extensively evaluate our system both in simulation and in field experiments by filming dynamic targets moving through unstructured environments.

Third, we take a step towards learning the intangible art of cinematography. We all know a good clip when we see it - but we cannot yet objectively specify a formula. We propose the use of deep reinforcement learning with a human evaluator in the loop to guide the selection of artistic shots, and show that the learned policies can incorporate intuitive concepts of human aesthetics. Next, we develop novel data-driven framework to enable direct user control of camera positioning parameters in an intuitive learned semantic space (*e.g.* calm, enjoyable, establishing), and show its effectiveness in a series of user studies.

Lastly, we take the first steps towards the concept of multi-camera collaboration for filming. The use of multiple simultaneous viewpoints is necessary when capturing real-world scenes such as sports or social events. In these situations it is difficult to capture the optimal viewpoint at all times employing a single aerial camera, specially because the events cannot be reenacted for additional takes. Here, we design motion planning algorithms for multi-camera cinematography that are able to maximize the quality of multiple video streams simultaneously using limited onboard resources.

Acknowledgements

I would first like to thank my advisor Sebastian Scherer for his guidance and support throughout these incredible years at graduate school. I am inspired by his commitment and passion towards field robotics, which have influenced much of my research so far. Thank you for giving me the freedom to explore collaborations and new ideas across the domains of robotics, machine learning, and computer vision.

I also want to thank my thesis committee members for their invaluable feedback. I would like to thank Ashish Kapoor for giving me the opportunity to do a fantastic internship at Microsoft, and the encouragement to explore topics in multi-modal sensing and representation learning. His insightful questions always made me go back to the fundamentals of the problem. I had the great opportunity to work with Jessica Hodgins at Facebook, and I am immensely thankful for her teachings and feedback on user how to scientifically evaluate subjective human experiences. Her guidance was crucial for the design of my user study methodology, and she helped put the word *artistic* into the title of this thesis. Oliver Kroemer has provided me with indispensable feedback on how to structure my thesis document: thank you for helping me craft my work into a cohesive storyline. I also want to thank Nathan Ratliff for his thoughtful feedback, in addition to his fantastic work on motion planning, which sparked several ideas throughout my research. I also received valuable advice and teachings from several other faculty members at CMU who shaped my thoughts over the years: Simon Lucey, David Held, Yaser Sheikh, Maxim Likhachev, Kris Kitani, Sidd Srinivasa, Sanjiv Singh, Henny Admoni, Alex Waibel, Ralph Vituccio, and Scott Stevens.

I could not have asked for a better house than the Air Lab for the past five years. First and foremost, I owe a special thanks to Sanjiban Choudhury for being such an amazing mentor and friend. Thank you for believing in my potential and encouraging me to set a high bar for myself. Our discussions inspired many new research directions, and your feedback greatly helped me evolve as a researcher. It has been a privilege to collaborate with my incredible lab colleagues, who had such a positive impact on my research: Cherie Ho, Wenshan Wang, Aayush Ahuja, Efe Camci, Mirko Gschwindt, Arthur Buckner, and Daniel Maturana. Thanks Cherie for all the amazing discussions, and Arthur for being the best intern, and doing the impossible in remote times. Thanks to the lab friends who made this journey so much more fun: Azarakhsh Keipour, Mohammadreza Mousaei, Sankalp Arora, Ratnesh Madaan, Rohit Garg, Lucas Nogueira, Jay Patrikar, Brady Moon, Andrew Saba, Yaoyu Hu, Anish Bhattacharya, Vaibhav Viswanathan, Kevin Pluckter, Vishal Dugar, Puru Rastogi, Dong-Ki Kim, Silvio Maeta, Guilherme Pereira, and Geetesh Dubey. Last but not least, I would like to thank Nora Kazour, Sanae Minick, and Suzanne Lyons Muth for all of their help.

I am grateful for all the friends I've made at CMU over the past years: Alex Spitzer, Adithya Murali, Xuning Yang, Anirudh Vemula, Shaurya Shankar, Ankit Bhatia, Pragna Mannam, Paloma Sodhi, Dhruv Saxena, Cara Craig, Shushman Choudhury, Rosario Scalise, Jaskaran Grover, Samuel Clarke, Shohin Mukherjee, Rick Goldstein, Brian Okorn, Jerry Hsiung, Wenhao Luo, Naji Shajarisales, and Thomas Weng. I will always look back with nostalgia at our grad school adventures together, from hikes in the Pittsburgh woods to white water rafting (rock bottom!), world-wide trips, and the late nights studying and/or having fun.

My life at CMU has been significantly more gratifying, interesting and fun, because it has been filled with many smart, enthusiastic, and supportive friends. The list of people who had positive influences on my experience does not fit in these lines, but I am grateful to all of you for sharing this time with me.

I am thankful for all the support from the folks at CMU's Swartz Center for teaching me a great deal about entrepreneurship. Thanks to Dawn Rucker, Dave Mawhinney, Sonya Ford, and Kit Needham for the guidance. Thanks to Vai Viswanathan and Max Henkart for being amazing partners; I learned a great deal about startup development from both of you. And thanks all the Swartz Fellows for the terrific interactions at the Friday seminars and beyond.

I was also extremely lucky to have collaborated and have been mentored by partners in industry and academia. Thanks to Mustafa Mukadam and Jessica for opening the doors for me for a fantastic internship at Facebook and for all the guidance. Thanks to Vibhav Vineet at Microsoft for teaching me so much about computer vision, plus Ashish and Ratnesh for the close collaboration. Thanks to Hyun-Soo Park, Volkan Isler, Kris Kitani, and Oscar Rodrigues Alves for the recent collaborations and ideas.

I would not be at CMU today if it were not for the guidance of mentors who believed in my potential, gave me opportunities, and pushed me to go much further in life than what I could have imagined. Thanks to Fabio Cozman, Marcel Bergerman, Rosiane Pecora, Carlos Pegurier, Cris Fortes, Henrique Takachi, Ephraim Garcia, and Tim DeVoogd for the immense positive impact on my life. This is just the beginning.

Thank you Gabriel Bayomi for being a faithful friend and confidant over the past years - at last our strenuous journey at CMU is done. Also, thanks to all my friends outside of grad school for the continued support over many years: Arthur Gola, Gustavo Woiler, Thomas Kilmar, Joao Marques, Gabriel Francisco, Leonardo Tinoco, Gustavo Majzoub, Paulo Fisch, Lucas Chow, and Asta Li.

Finally, no words can express the feelings of gratitude towards my family. I am immensely grateful for my parents Fernanda and José, who showed me endless love, support and encouragement. I want to thank my siblings Rodolfo and Laura for always being by my side. And to my fiancé Rachel, I leave my utmost appreciation. Thank you for the love and encouragement, even in the toughest times of this journey. Thank you all for always believing in me, and being with me every step of the way.

This work was funded by a Siebel Scholarship, a Microsoft Research Dissertation Grant, a Swartz Entrepreneurial Fellowship, Waibel Education Fund, and Yamaha Motor Co. under grant number A019969. Their support is gratefully acknowledged.

Contents

Abstract	iii
Acknowledgements	iv
List of Figures	ix
List of Tables	xii
1 Introduction	1
1.1 Motivational applications	2
1.2 Challenges of autonomous aerial filming	3
1.3 Contributions	5
2 Background	9
2.1 Taxonomy of a studio movie production	9
2.2 Autonomous filming problem definition	11
2.3 Related Work	13
2.3.1 Virtual cinematography	13
2.3.2 Automatic movie editing	14
2.3.3 Autonomous aerial cinematography	14
2.3.4 Making artistic choices autonomously	15
2.3.5 Online environment mapping	16
2.3.6 Visual target state estimation	16
3 Planning for moving cameras	18
3.1 UAV Trajectory Definition	18
3.2 Defining Cost Functions and a Planner for Aerial Filming	19
3.2.1 Definition of Cost Functions	20
3.2.2 Trajectory Optimization Algorithm	24
3.3 Experiments	25
3.3.1 Simulation experiments	25
3.3.2 Field experiments	26
3.4 Conclusion	28
4 Building a System for Filming in the Wild	30
4.1 System Overview	30
4.1.1 Design Hypotheses	31
4.1.2 System Architecture	31
4.1.3 Hardware	32
4.1.4 Photo-realistic Simulation Platform	33

4.2	Online Environment Mapping	33
4.2.1	LiDAR Registration	34
4.2.2	Occupancy Grid Update	34
4.2.3	Incremental Distance Transform Update	35
4.2.4	Building a Height Map	36
4.3	Visual Actor Localization and Heading Estimation	36
4.3.1	Detection and Tracking	37
4.3.2	Heading Estimation	39
4.3.3	Ray-casting	40
4.3.4	Motion Forecasting	41
4.4	Experimental Results	42
4.4.1	Integrated System Results	42
4.4.2	Visual Actor Localization and Heading Estimation	46
4.4.3	Planner Evaluation	50
4.5	Discussion	52
4.5.1	Lessons Learned	52
4.5.2	Adapting Our Work to Different UAVs and Sensors	53
4.6	Conclusion	53
5	Learning Artistic Decision-Making	55
5.1	Deep Reinforcement Learning Problem Formulation	55
5.2	Reward Definition	58
5.3	Implementation Details	60
5.4	Results	61
5.4.1	Learning an artistic policy	61
5.4.2	User study results	63
5.4.3	Extended results in field experiments	65
5.5	Conclusion and discussion	66
6	Learning semantic camera controls	68
6.1	Introduction	68
6.2	Related Work	69
6.3	Perceptual Experiments	70
6.3.1	Minimal Perceptual Units for Shot Parameters	71
6.3.2	Obtaining Semantic Scores for Videos	72
6.3.3	Building a Semantic Descriptor Space via Crowd-Sourcing	73
6.4	Learning a Semantic Control Space	75
6.4.1	Training Details	76
6.4.2	Model Results	76
6.5	Experimental Validation	76
6.5.1	Semantic Control Space	76
6.5.2	Semantic Shot Classification and Latent Space Analysis	78
6.6	Conclusion and Discussion	79
7	Multi-camera coordination	81
7.1	Introduction	81
7.2	Related Work	82
7.3	Multi-Camera Coordination	83

7.3.1	Centralized Multi-Camera Planning	83
7.3.2	Decentralized UAV Trajectory Optimization and Tracking	87
7.3.3	Live Image Selection	87
7.4	Experimental Results	87
7.4.1	Simulation experiments	87
7.4.2	Real-world results	89
7.5	Conclusion and Discussion	90
8	Learning state representations for aerial navigation	93
8.1	Introduction	94
8.2	Related Work	96
8.3	Approach	97
8.3.1	Definitions and Notations	97
8.3.2	Learning Cross-Modal Representations for Perception	97
8.3.3	Imitation learning for control policy	99
8.4	Results	99
8.4.1	Learning Representations	99
8.4.2	Simulated navigation results	101
8.4.3	Real-World Results	102
8.5	Conclusion and Discussion	105
9	Conclusion	107
9.1	Summary and contributions	107
9.2	Future directions	109
9.2.1	Learning robot and camera motion styles	109
9.2.2	Using dynamic aerial cameras for new applications	110
9.2.3	Learning better state representations	110
9.3	Concluding remarks	111
A	Derivations of planning cost functions	112
A.1	Smoothness cost	112
A.2	Shot quality cost	113
	Bibliography	115

List of Figures

1.1	Applications for dynamic aerial cameras	2
1.2	Aerial cinematography pipeline	3
1.3	Contextual and geometrical computational threads	4
2.1	Movie studio production process	10
2.2	Differences between studio production and our system	11
3.1	Artistic shot parameters	21
3.2	Occlusion cost function	23
3.3	Randomized occlusion environment	25
3.4	Planner behavior with different time horizons	26
3.5	Planning system architecture	27
3.6	Photo-realistic simulator	27
3.7	Comparison of planning with and without occlusion cost	28
3.8	Planning results under different scenarios	29
4.1	Full system architecture	32
4.2	Experimental hardware and drone	33
4.3	Airsim simulation platform	34
4.4	Map with obstacle representations	36
4.5	Building a height map	37
4.6	Vision sub-system	37
4.7	Examples of challenging images for actor detection	38
4.8	Rule of thirds parameters	38
4.9	Example of actor bounding boxes and heading angles	39
4.10	Heading prediction network architecture	41
4.11	Raycasting module for obtaining actor's 3D configuration	41
4.12	Testing facilities	43
4.13	Time lapse of back shot following a runner	43
4.14	Incremental distance transform computation time	44
4.15	Field results with different shot types and environments	45
4.16	Precision recall curve for object detection	47
4.17	Different heading estimation models tested on the sequential data	48
4.18	Heading estimation module using more or less data	49
4.19	Pose and heading estimation results	50
4.20	Planner performance with full knowledge of the map versus online mapping	51
4.21	Planner performance with perfect actor location versus noisy estimates	51

5.1	Example of artistic parameter selection as a sequential decision-making problem	56
5.2	Shot length decrease over the years	57
5.3	Coupling sparse artistic decisions with the motion planner	58
5.4	Shot discount parameter over shot repetition count	59
5.5	Video clips human evaluators during training procedure	60
5.6	Artistic shot selection network architecture	60
5.7	Rendering and height maps of AirSim environments	62
5.8	Drone trajectory during filming in photo-realistic environment	63
5.9	Visualization of the Q-values during testing	64
5.10	Drone trajectories of the highest rated policies	65
5.11	UAV follows an actor in the wild	65
5.12	Field test results using the online artistic shot selection	66
6.1	Proposed framework for semantic camera control	69
6.2	Simulator setup	71
6.3	Dynamic shots	72
6.4	Minimal perceptual deltas	73
6.5	Correlations between descriptors	74
6.6	3D emotional space	75
6.7	Normalized linear coefficients	77
6.8	Emotional model prediction	78
6.9	Likert user study	78
6.10	Latent space of shots and emotions	79
6.11	Real-world testing	79
7.1	Proposed framework for multi-camera control	82
7.2	Multi-camera system overview	84
7.3	Discrete spherical state space	84
7.4	Visualizing cell costs	86
7.5	Greedy planner	86
7.6	Simulations of multi-camera setup	88
7.7	Multi-drone user study	88
7.8	Multi-drone planning time	90
7.9	Multi-drone field experiment	91
7.10	Soccer game filming	91
8.1	Proposed framework for sim-to-real transfer	94
8.2	Control system architecture	97
8.3	Cross-modal VAE architecture	98
8.4	Visualization of latent space	100
8.5	Original simulated images with their respective CM-VAE reconstructions	100
8.6	Visualization of latent space interpolation between two simulated images	101
8.7	Performance of different navigation policies on simulated track	102
8.8	UAV platform.	103
8.9	Visualization of latent space interpolation between two real-world images	103
8.10	Side and top view of: a) Circuit track, and b) S-shape track.	104
8.11	Analysis of a 3-second flight segment	104
8.12	Examples of challenging test environments	105

9.1	Summary of thesis contributions	108
9.2	Angular coverage metric in multi-drone scenario	110

List of Tables

1.1	Videos and source code for thesis chapters	8
2.1	Comparison of related works	15
3.1	Occlusion evaluation in randomized environment	26
3.2	Planner performance under various time horizons	26
4.1	System and payload weights.	33
4.2	Objectives and results for integrated experiments.	43
4.3	System statistics recorded during flight time on onboard computer.	44
4.4	Objectives and results for vision-specific experiments.	46
4.5	Datasets used in HDE study	47
4.6	Semi-Supervised Finetuning results	48
4.7	Objectives and results for detailed motion planner experiments	50
4.8	Performance comparison between ground-truth and online map	50
5.1	Objectives and results for artistic shot selection	61
5.2	Average reward per time step	62
5.3	Average normalized score of video clips between 0 (worst) and 10 (best)	64
6.1	Shot presets	72
6.2	Descriptor clusters	74
7.1	Greedy planner performance	89
8.1	Average and standard errors for encoding gate poses	101
8.2	Policy performance in number of gates traversed	103
8.3	Examples of distance to gates decoded from real images	105

Chapter 1

Introduction

Manually-operated unmanned aerial vehicles (UAVs) are drastically improving efficiency and productivity in a diverse set of industries and economic activities. In particular, tasks that require dynamic camera viewpoints have been most affected, where the development of small scale UAVs has alleviated the need for sophisticated and expensive hardware to manipulate cameras in space. Indeed, the global drone market is expected to grow at 17% year-over-year, reaching almost US\$52B by 2023, of which US\$30B will be in defense expenditures and US\$9B in entertainment [153].

In the movie industry, for instance, drones are changing the way both professional and amateur film-makers can capture shots of actors and landscapes by allowing the composition of aerial viewpoints which are not feasible using traditional ground devices such as hand-held cameras and dollies [204]. Several recent blockbuster movies such as *The Hunger Games* and *Skyfall* made extensive use of drones in their productions, with shots that previously required the use of helicopters being produced at a fraction of the cost [153]. In amateur film-making, drones also became an important tool for supporting human creativity and expressiveness, fundamentally altering the way we share spontaneous experiences and social events. In the sports domain, flying cameras can track fast-moving athletes and accompany dynamic movements [135]. Furthermore, flying cameras show a largely unexplored potential for tracking subjects of interest in defense and security applications [56], which are not possible with static sensors.

The main issue with aerial cameras is that they are extremely difficult to control. Manually-operated UAVs often require multiple expert pilots due to the difficulty of executing all necessary perception and motion planning tasks synchronously. The online and dynamic nature of the filming task poses an enormous decision-making burden over human operators, often leading to crashes. It takes high attention and effort to simultaneously identify the actor(s), predict how the scene is going to evolve, control the UAV, avoid obstacles and reach the desired viewpoints. How can we empower *any* individual with the full potential of aerial cameras?

The goal of this thesis is to augment human creativity and expressiveness with the use of autonomous drones. We wish to empower any individual with the full artistic capabilities of aerial cameras.



Figure 1.1: Active vision applications require aerial cameras to interact with highly dynamic targets. The use of static sensors is not sufficient to follow or control fast moving targets.

1.1 Motivational applications

The astounding recent developments of computer vision and robot autonomy have reached the space of aerial vehicles, causing profound impact on several classes of applications. Better motion planning algorithms and neural-network based image processing allow UAVs need to safely traverse an environment with obstacles while tending for additional objectives. For instance, we works with UAVs inspecting 3D structures [194, 243, 247] and power lines [104, 161], efficiently delivering cargo [44], or exploring unknown environments [205].

Our work, however, develops solutions for a different class of problems, which is relatively less explored in the UAV literature. We focus on UAV motion in dynamic scenes, which involve planning the motion of UAVs relative to moving targets. Figure 1.1 shows examples of important applications where a drone interacts dynamically with targets in the environment. Being able to capture dynamic footages opens doors to a whole new set of applications, from herding animals in agriculture [82] to using UAVs for military and police defense systems.

In particular, the field of entertainment presents numerous use cases for dynamic aerial systems. With UAVs we can capture players practicing sports in amateur and professional settings, record challenging scenes for Hollywood movies, or film social events. We find almost untapped potential for both new research and commercial applications for autonomous filming, enabled in most part by the light-weight, safe, low-cost and agile of UAVs.

However, a whole new set of challenges arises when we consider filming scenes *in the wild* due to the online nature of the problem. Unlike the scripted scenes of a movie studio, in real-world scenarios our flying camera needs to make decisions based on the information it senses, without a strong prior of the future that lies ahead (Fig. 1.2). When filming a mountain biker’s jump in the woods, we don’t have the luxury to record multiple takes of the same scene; the UAV must do its best with the information available at hand. The next section details the main difficulties in developing an autonomous aerial camera.

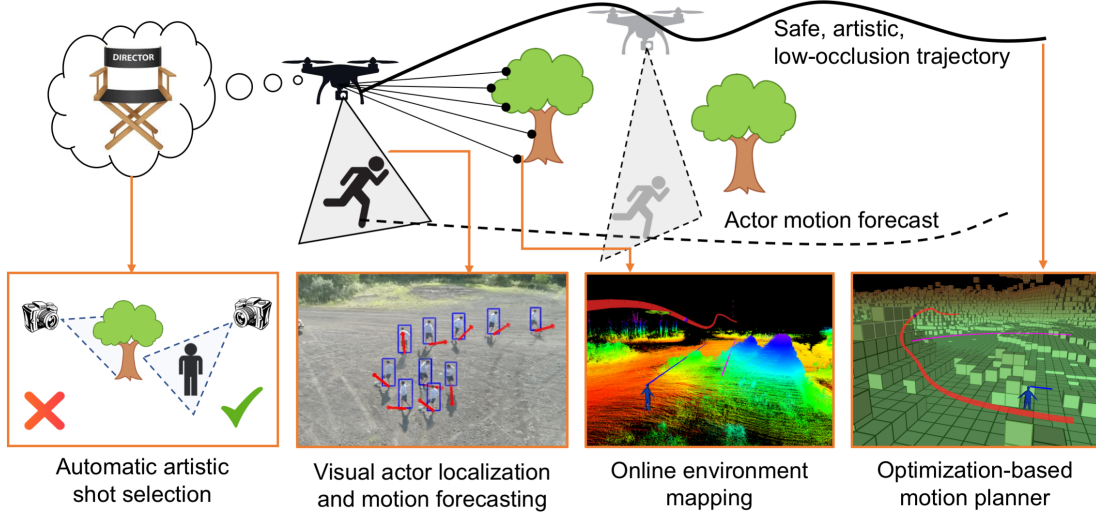


Figure 1.2: Aerial cinematographer pipeline: the UAV visually detects the actor’s motion using a vision-based localization module, maps the environment with an onboard LiDAR sensor, reasons about artistic guidelines, and plans a smooth, collision-free trajectory while avoiding occlusions.

1.2 Challenges of autonomous aerial filming

An aerial camera platform can be categorized as a system for *active vision*, meaning that it has gaze control mechanisms that can actively change the position and orientation of viewpoints in response to environmental stimuli [11, 12, 28]. This paradigm allows us not only to passively process the incoming video feed, but to also acquire data that would otherwise be unseen with the use of passive sensors. We close the loop between sensing and acting: our camera planning objective involves not only going from point A to point B, but rather designing paths while reasoning about the quality of future camera viewpoints.

The distinctive challenge of developing an autonomous aerial cinematography system lies in the need to tightly couple *contextual* and *geometric* processing threads, in real time. Contextual reasoning involves processing camera images to detect the actor and the semantic composition of its surroundings, understanding how the scene is going to evolve, and hallucinating over the most desirable viewpoints to maximize artistic quality. Geometric reasoning is tied to motion planning, and considers the 3D configuration of objects in the environment. We use geometry to evaluate actor occlusions or visibility from particular viewpoints, and whether the UAV can reach it in a safe manner without collisions. Although these two threads differ significantly in terms of sensing modalities, computational representation and computational complexity, both sides play a vital role when addressing the entirety of the autonomous filming problem. In this work we present a complete system that combines both threads in a cohesive and principled manner.


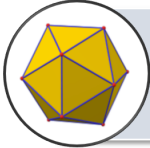
Processing thread	Components	Sensors and representations
 <div>Scene Context</div>	<ul style="list-style-type: none"> • Actor detection • Activity forecasting • Semantic understanding • Artistic reasoning 	<ul style="list-style-type: none"> • Cameras <ul style="list-style-type: none"> • Bounding boxes • Probability maps • Learned artistic models
 <div>Scene Geometry</div>	<ul style="list-style-type: none"> • Obstacle mapping • Motion planning • Collision / occlusion avoidance 	<ul style="list-style-type: none"> • Range sensors (LiDAR) <ul style="list-style-type: none"> • Occupancy maps • Signed distance maps

Figure 1.3: Combining contextual and geometrical processing threads is a major challenge in the aerial filming problem. Both have distinct sensor modalities and computational representations, varying from probability maps to occupancy grids.

In specific, in a typical filming scenario the UAV must overcome several challenges:

Challenge 1: Keep actor visible while staying safe:

In cinematography, the UAV must maintain visibility of the actor for as long as possible while staying safe in a partially known environment. When dealing with dynamic targets, the UAV must anticipate the actor’s motion, forecasting future movements, and reason about collisions and occlusions generated by obstacles in potential trajectories. The challenge of visibility has been explored in previous works in varying degrees of environment complexity [22, 73, 171, 181].

Challenge 2: Operating in unstructured scenarios:

The UAV typically flies in diverse, unstructured environments without prior information. In a standard mission, it follows an actor across varying terrain and obstacle types, such as slopes, mountains, and narrow trails between trees or buildings. The challenge is to maintain an online map that has a high enough resolution to reason about viewpoint occlusions and that updates itself fast enough to keep the vehicle safe.

Challenge 3: Actor pose estimation with challenging visual inputs:

The UAV needs to film a dynamic actor from various angles (*e.g.*, for a frontal or right-side shot), therefore it is critical to accurately localize the actor’s position and orientation in a 3D environment. In realistic use cases, outside the context of research, the use of external sensors such as motion capture systems [171] and GPS tags [22, 110] for pose estimation is highly impractical. A robust system should only rely on visual localization. The challenge is to deal with all possible viewpoints, scales, backgrounds, lighting conditions, and motion blur caused by both the dynamic actor and fast-moving camera.

Challenge 4: Understanding scene context for artistic decision-making:

When filming a movie, the director actively selects camera poses based on the actor’s movement, environment characteristics and their intrinsic artistic values. Humans can make such aesthetic decisions implicitly, based on intuition and knowledge learned over a lifetime of training and observations. Even though demonstrations of good artistic actions are relatively easy to obtain, it is challenging to explicitly define rules for the optimal artistic

choices within a given scene context. Additionally, it is also challenging to find quantitative metrics for evaluating the quality of artistic decisions.

Challenge 5: Translating user commands into robot decision-making:

Generating expressive camera behaviors generally requires non-technical users to edit a large number of unintuitive control parameters when dealing with a robotics system. A major challenge arises when our goal is not to completely replace the human in the loop, but rather to have users provide intuitive high-level controls to the vehicle, which are often given in terms of semantics.

Challenge 6: Making real-time decisions with onboard resources:

Our focus in this work is on unscripted scenarios where shots are decided on the fly. A major challenge we face is to design algorithms for perception, decision-making and control that can run in real-time on a UAV with limited computational resources.

Challenge 7: Learn in simulation, deploy in the wild:

Simulators can be powerful tools for learning models for visuo-motor navigation, such as policies for artistic decision-making. Unlike training directly in the real-world, simulation is cheap, safe, and scalable. The major drawback, however, comes from the differences in appearance between simulated and real data. Successful models require environment representations that are invariant to specificities of simulated visual appearances, and cannot overfit to synthetic data.

1.3 Contributions

There is a rich history of work in autonomous aerial filming that tackles parts of the challenges presented above. For instance, we find several works focused on integrating cinematographic guidelines into motion planning [71, 73, 110, 171]. These works, however rely on perfect actor localization through high-precision RTK GNSS or motion-capture systems, hindering their applicability in real-world scenarios. Additionally, while the majority of work in the area applies safety checks to ensure the UAV does not crash into the actors [98, 110, 171], the environment is not factored in at all, both in terms of collisions with obstacles and occlusions that may block actor visibility. Such simplifications impose restrictions on the diversity of real-life scenarios that these systems can handle. While there are several successful commercial drone products in the market, they too have certain limitations. For example, products such as DJI Mavic [1] can visually track targets, but have obstacle avoidance capabilities that only operate well in low speed and low clutter regimes. Today, the most advanced system on the market for aerial filming is the Skydio R1 [211], with impressive real-time mapping and obstacle avoidance capabilities.

Nonetheless, no platform available today has the ability to reason about artistic principles during the filming process: they require the user to carefully pre-plan the scene to be recorded. Even with the most advanced platforms, the amateur drone operator needs to create a script for the scene that is about to happen, and *a priori* select the most appropriate viewpoints, even if the drone can navigate to those positions autonomously. This proves to be impossible in several real-world use cases where scenes are inherently unscripted, such as sports, social events and news coverage. Furthermore, current operator-drone interfaces such as joysticks connected to cell-phones are cumbersome and unintuitive to access during deployment, specially if the operator is doing the activity herself (*e.g.*, a mountain biker going

down a trail). Our goal, instead, is to free the operator of the burden of decision-making. We want to empower amateur users with the full capabilities of aerial cameras, rendering professional studio-like quality to amateur footages of dynamic scenes. Users should focus on what matters the most: doing the activity itself, and not worry about how to record it.

The thesis revolves around two key ideas. First, we design differentiable objective functions for camera trajectories that can translate the language of the movies into the motion plan of a UAV. We architect our system to compute these objectives efficiently in real time to film a dynamic actor. Second, we apply machine learning to elicit human artistic preferences in selecting shot configurations and sequences. By employing both key ideas, we hypothesize that users are able to produce better quality movies with less effort. Specifically, we offer the following contributions:

- In Chapter 2 we provide an overview and mathematical problem definition of all the components present in the aerial cinematography problem. In addition, we describe a large body of related works, and compare our contributions to the existing literature.
- In Chapter 3 we begin by formalizing the aerial filming motion planning problem following cinematography guidelines for arbitrary types of shots and arbitrary obstacle shapes, while avoiding collisions and occlusions. We propose an efficient optimization-based motion planning method that exploits covariant gradients and Hessians of the objective functions for fast convergence. Reference paper:
 - Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming (ISER 2018) [22]
- In Chapter 4 we develop a system for real-time autonomous aerial cinematography *in the wild*, removing the simplifying assumptions that have been commonly used in previous literature. We propose an incremental signed distance transform algorithm for large-scale real-time environment mapping using a range sensor, *e.g.*, LiDAR (Section 4.2). In addition, we introduce a method for visually localizing the actor’s position, orientation, and forecasting their future trajectory in the world coordinate frame. We present a novel semi-supervised approach that uses temporal continuity in sequential data for the heading direction estimation problem (Section 4.3). Finally, we extensively evaluate our system both in simulation and in field experiments by filming dynamic targets moving through unstructured environments. Reference papers:
 - Towards a Robust Aerial Cinematography Platform (IROS 2019) [17]
 - Improved Generalization of Heading Direction Estimation (ICRA 2019) [236]
- In Chapter 5 we propose a deep reinforcement learning method that incorporates human aesthetic preferences for artistic reasoning. The model acts as an autonomous movie director, making high-level shot selection decisions considering the current scene context to select the next camera viewpoints. We offer extensive quantitative and qualitative performance evaluations and user studies of the system in simulation and field tests. Reference papers:
 - Autonomous aerial cinematography in unstructured environments with learned artistic decision-making (Journal of Field Robotics 2019) [19]
 - Can a Robot Become a Movie Director? Learning Artistic Principles for Aerial Cinematography (IROS 2019) [85]

- In Chapter 6 we create a data-driven framework that enables editing of these complex camera positioning parameters in an intuitive semantic space (*e.g.* calm, enjoyable, establishing). First, we generate a database of video clips with a diverse range of shots in a photo-realistic simulator, and use hundreds of participants in a crowd-sourcing framework to obtain scores for a set of semantic descriptors for each clip. Next, we learn a generative model that can map a set of desired semantic video descriptors into low-level camera trajectory parameters. We evaluate our system by demonstrating that the model successfully generates shots that are rated by participants as having the expected degrees of expression for each descriptor in simulated and real-world experiments. Reference paper:
 - Batteries, camera, action! Learning a semantic control space for expressive robot cinematography (ICRA 2021) [16]
- In Chapter 7 we extend the drone-filming problem towards a multi-camera setting. We develop a real-time multi-UAV coordination system that is capable of recording dynamic targets while maximizing shot diversity and avoiding collisions and mutual visibility between cameras. We validate our approach in multiple cluttered environments of a photo-realistic simulator, and deploy the system using two UAVs in real-world experiments. Reference paper:
 - Do You See What I See? Coordinating Multiple Aerial Cameras for Robot Cinematography (ICRA 2021) [29]
- In Chapter 8 we briefly step away from the cinematography topic, and explore efficient representations for UAV navigation within the context of drone racing. We propose the use of cross-modal variational auto-encoders as a means to generate rich latent spaces for deep navigation policies. We use imitation learning to train motion policies in simulation, and directly deploy the networks in challenging real-world scenarios. Reference paper:
 - Learning Visuomotor Policies for Aerial Navigation Using Cross-Modal Representations (IROS 2020) [18]
- Finally, in Chapter 9 we offer the concluding remarks connecting all multiple works along with and potential future research directions.

Together, these contributions form a comprehensive system for autonomous aerial cinematography. It is important to note that the algorithmic contributions from this thesis can impact applications beyond context-specific tasks in cinematography. Just within the field of aerial robotics, the concept of *active vision* can be used for a diverse range of activities such as data collection, aerial surveillance, and animal monitoring [24]. Other robotics applications such as autonomous vehicles and manipulators can also greatly benefit from many of the ideas presented here. For instance, we can improve robot performance by taking into account scene visibility and occlusion avoidance inside of the motion planning pipeline. In addition, the data-driven framework for altering robot behavior using semantic controls (discussed mostly in Chapter 6) can be applied towards uncovering features that perform different robot motion styles, which is important when operating near and around humans.

In Table 1.1 we provide links to the overview videos of each chapter, and open-source implementations where applicable. The overview videos are intended to be viewed before the reader dives into each chapter.

Table 1.1: Videos and source code for thesis chapters

Chapter 3: Planning for moving cameras	
Video	https://youtu.be/QX73nBBwd28
Chapter 4: Building a Complete System for Aerial Cinematography	
Platform video:	https://youtu.be/ZE9MnCVmumc
Vision module video:	https://youtu.be/-UVSXSxtKN4
Chapter 5: Learning Artistic Decision-Making	
Journal video:	https://youtu.be/ookhHnqmlaU
Paper video:	https://youtu.be/qmVw6mfyEmw
Chapter 6: Learning semantic camera controls	
Paper webpage:	https://sites.google.com/view/robotcam
Video:	https://youtu.be/6WX2yEUE9_k
Chapter 7: Multi-camera coordination	
Video:	https://youtu.be/m2R3anv2ADE
Chapter 8: Learning state representations for aerial navigation	
Video:	https://youtu.be/VKc3A5H1UU8
Source code:	https://github.com/microsoft/AirSim-Drone-Racing-VAE-Imitation

Chapter 2

Background

In this chapter we introduce the preliminary definitions and terminology that will setup the infrastructure for the remaining discussions in this thesis. First, in Section 2.1 we lay out the phases present in the production of movies in professional studios. Understanding the structure and components from Hollywood-style movies help us identify which functionalities we want or don't want to emulate with our autonomous platform. Next, in Section 2.2, we formally define what we mean by the cinematography planning problem for a an aerial robot. Finally, in Section 2.3 we present a broad collection of related works from various fields that help us better contextualize our problem and contributions.

2.1 Taxonomy of a studio movie production

The process of producing a professional movie entails a complex and coordinated effort from dozens of artists and technical workers such as actors, producers, directors, cinematographers, makeup artists, and technicians [215]. Based on the cinematography literature [5, 26, 215] we lay the basic language on movie-production concepts that will be used throughout the document:

- *Synopsis*: is a short summary of the movie story, usually around 30 pages;
- *Script*: is a detailed version of the movie story that includes dialogs, storylines and background descriptions. Around 90-120 pages long, where each page equates to roughly one minute of the final movie;
- *Scene*: specific unit of action that is spatially and temporally continuous. It is defined by lighting conditions, actor movement, camera position, and camera angles;
- *Shot*: is the basic unit of a movie, captured in a single uninterrupted camera run;
- *Take*: is a specific version of a shot. A movie production may have more or less takes depending on schedule and budget constraints;
- *Sequence*: has a broad definition, being a group of scenes coupled by a common theme or idea;
- *Shooting script*: is a document prepared by the director containing all technical information necessary for producing the movie. Is similar to a storyboard;

- *Director*: is the person in charge of the translation of the script into images and sounds. He or she leads the day-to-day shooting operations, and sometimes their reach can also extend to scripting, casting and editing;
- *Cinematographer*: is the person responsible for planning and executing camera angles and positioning.

We can break down the long production process into three main phases: pre-production, production and post-production. As seen in Figure 2.1, each phase encompasses numerous tasks:



Figure 2.1: Movie studio production process involves three phases: pre-production, where the script and actors are defined; production, where the movie is actually shot; and post-production, where editing happens.

The first phase, among other tasks, involves writing the scripts, financing the production, and defining the shooting locations. The production phase is where the actual shooting takes place, and is responsible for most of the costs of the venture. The post-production phase takes care of editing all of the recorded content, adding visual effects, soundtracks and dubbing.

The main motivation behind this thesis is to allow amateur camera users to produce high-quality content without the direct support of a studio production crew and its costly resources. Our focus is on the types of scenarios typically encountered by aerial camera users filming unscripted action scenes involving dynamic actors in natural environments. These can range from a person practicing martial arts in a park to a runner or mountain-biker going down a trail surrounded by trees.

The scenes we consider in this work are mostly non-scripted and spontaneous. The camera user does not necessarily know where the actor is going to, and which elements will make up the filming set in terms of background and obstacles. Therefore, our system fundamentally cannot deal with most of the tasks pertaining to the pre-production phase. Our focus is mostly on developing online solutions for automating tasks relative to the movie production itself. Differently than a professional production, our the typical use cases we consider here do not allow for multiple takes of the same scene, nor do we count with multiple cameras from which we can do editing *a posteriori*. Arguably, our system needs to write the movie script on-the-fly, making the best use of the information collected up until that moment. Figure 2.2 shows analogies between concepts from traditional studio filming and our system.

Given these fundamental differences, while translating the rules of filming and editing from the domain of studios to autonomous drones we do not necessarily find a 1 : 1 mapping


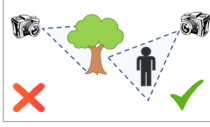

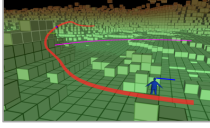

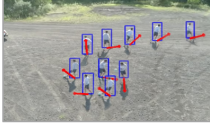

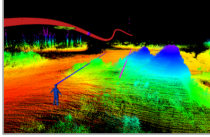
Movie studio concepts		Our related autonomous system	Differences
 <div>Storyboard writing Movie director Editing / cutting</div>	<ul style="list-style-type: none"> Automatic shot selection 		<p>We operate on unscripted scenarios, writing the movie on-the fly, and cutting between scenes by selecting viewpoints</p>
 <div>Camera placement Cinematographer</div>	<ul style="list-style-type: none"> Motion planning 		<p>Camera placement cannot be planned ahead of time, and adapts to the scene online</p>
 <div>Actor movements Assistant director</div>	<ul style="list-style-type: none"> Visual actor localization 		<p>Actor movements are spontaneous and do not follow a script: they are inferred rather than planned</p>
 <div>Movie set construction</div>	<ul style="list-style-type: none"> Environment mapping 		<p>The movie set must be sensed on-the-fly, and is not specified ahead of time</p>

Figure 2.2: Differences between studio production and our system.

between concepts. Although some rules transfer trivially between domains, we also investigate strategies which are unique to our domain. For instance, aerial cameras allow us to seamlessly switch between new vantage points, creating shot compositions and strategies that have not been possible with the use of ground equipment, or that required multiple cameras placed strategically coupled with editing later.

2.2 Autonomous filming problem definition

Following the ideas presented in this thesis' introduction, we provide a terse description for the motion planning pipeline for a mobile robot. The mobile robot, equipped with a range limited sensor, is moving in an environment. At a given planning cycle, the motion planning module is invoked to plan a dynamically feasible and collision-free path that follows a moving target, while obeying cinematographic guidelines defined by a scene director. This path is then sent to the control system of the robot which takes a step along the path. The robot receives measurements from its sensors, which it uses to update its belief about the world in terms of its own state, target motion and presence of obstacles. This cycle is repeated. Later, we dedicate Chapter 4 towards an in-depth discussion of a pipeline for operating in real-world conditions.

Let $\xi_q(t) : [0, t_f] \rightarrow \mathbb{R}^3 \times SO(2)$ be the trajectory of the UAV as a mapping from time to a position and heading, i.e., $\xi_q(t) = \{x_q(t), y_q(t), z_q(t), \psi_q(t)\}$. Analogously, let $\xi_a(t) : [0, t_f] \rightarrow \mathbb{R}^3 \times SO(2)$ be the trajectory of the actor in space: $\xi_a(t) = \{x_a(t), y_a(t), z_a(t), \psi_a(t)\}$. In our work, a instantaneous measurement of the actor state $S_a : \mathbb{R}^3 \times SO(2)$ is obtained using onboard sensors (monocular camera and LiDAR, as seen in Section 4.3), but external sensors and motion capture systems could also be employed (Section 2.3). Measurements S_a are continuously fed into a prediction module that computes ξ_a (Section 4.3).

The UAV also needs to store a representation of obstacles in the environment. Let grid $\mathcal{G} : \mathbb{R}^3 \rightarrow [0, 1]$ be a voxel occupancy grid that maps every point in space to a probability of occupancy. Let $\mathcal{M} : \mathbb{R}^3 \rightarrow \mathbb{R}$ be the signed distance value of a point to the nearest obstacle border. Positive signs are used to represent the distance for points in free space to the closest obstacle, and negative signs for the distance of points from occupied regions towards the closest free cell. We follow a conservative approach and assume that unknown/unmapped regions as being occupied. During flight the UAV senses the environment with an onboard LiDAR, and updates grid \mathcal{G} with its measurements. Next, it updates the signed distance map \mathcal{M} (more details at Section 4.2).

We can generically formulate a motion planning problem that aims to minimize a particular cost function $J(\xi_q)$ for cinematography. Given the vast body of literature within the context of movie-making and photography [5, 26, 215], we identify four major axis of constraints that an active aerial moving camera should satisfy. These major cost functions should provide a measure of jerkiness of motion, vehicle safety, environmental occlusion of the actor's image, and shot quality (artistic quality of viewpoints). These cost functions depend on various factors such as the environment configuration (\mathcal{G} and \mathcal{M}), the actor's motion forecast (ξ_a), and a set of artistic parameters, which we now generically define as Ω_{art} . All quantities are sensed on-the-fly, and the changing nature of environment and ξ_a demands re-planning at a high frequencies so that the UAV can adapt to the new information.

Here we briefly touch upon the four components of the cost function $J(\xi_q)$ (refer to Chapter 3 for details and mathematical expressions):

1. *Smoothness* $J_{\text{smooth}}(\xi_q)$: Penalizes jerky motions that may lead to camera blur and unstable flight. Only depends on the shape of the UAV trajectory;
2. *Safety* $J_{\text{obs}}(\xi_q, \mathcal{M})$: Penalizes proximity to obstacles and collisions that are unsafe for the UAV. Depends on the current UAV trajectory and obstacle field;
3. *Occlusion* $J_{\text{occ}}(\xi_q, \xi_a, \mathcal{M})$: Penalizes occlusion of the actor by obstacles in the environment, which is generally an undesired situation during a movie. Depends on the current UAV and actor trajectories, and map;
4. *Shot quality* $J_{\text{shot}}(\xi_q, \xi_a, \Omega_{\text{art}})$: Penalizes poor viewpoint angles and scales that deviate from the desired artistic guidelines, given by the set of artistic parameters Ω_{art} .

In its simplest form, we can express $J(\xi_q)$ as a linear composition of each individual cost, weighted by scalars λ_i . Our objective is find an optimal UAV trajectory ξ_q^* that minimizes $J(\xi_q)$, subject to the initial boundary constraints $\xi_q(0)$. The solution ξ_q^* is then tracked by the UAV:

$$\begin{aligned} \xi_q^* &= \arg \min_{\xi_q} J(\xi_q) , \\ \text{s.t.} \quad &\begin{cases} \xi_q(0) = p_0 = \{x_0, y_0, z_0, \psi_0\} \\ J(\xi_q) = \begin{bmatrix} 1 & \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix} \begin{bmatrix} J_{\text{smooth}}(\xi_q) \\ J_{\text{obs}}(\xi_q, \mathcal{M}) \\ J_{\text{occ}}(\xi_q, \xi_a, \mathcal{M}) \\ J_{\text{shot}}(\xi_q, \xi_a, \Omega_{\text{art}}) \end{bmatrix} \end{cases} \end{aligned} \quad (2.1)$$

We describe in Chapter 3 how the optimization from Equation 2.1 is solved.

The parameters Ω_{art} of the shot quality term J_{shot} are usually specified by a user prior to takeoff. They can remain constant throughout flight, or change dynamically, either by user’s choice or algorithmically. For instance, based on the terrain characteristics and the type of motion the user expects the actor to do, they may initially specify a frontal shot, closely following the actor from ahead, and later perform a circular shot while keeping a greater distance from the target.

A dynamically changing Ω_{art} leads to a new challenge: the UAV must make choices that maximize the artistic value of the incoming visual feed. As explained further in Section 5, choices of artistic parameters at time t affect not only the immediate images recorded by the UAV, but also influence the states and situations that the UAV encounters at later time steps. Therefore, we frame the selection of Ω_{art} as a sequential decision-making process.

Let video $v_t = \{I_1, I_2, \dots, I_k\}$ be a sequence of k observed images captured by the UAV during time step t between consecutive decisions over the artistic parameters. Let $R_{\text{art}}(v_t)$ be the user’s implicit evaluation reward based on the observed the video segment v_t . The user’s choice of an optimal artistic parameter sequence $\{\Omega_1^*, \Omega_2^*, \dots, \Omega_n^*\}$ can be interpreted as an optimization of the following form:

$$\{\Omega_1^*, \Omega_2^*, \dots, \Omega_n^*\} = \arg \max_{\{\Omega_1, \Omega_2, \dots, \Omega_n\}} \sum_t R_{\text{art}}(v_t) \quad (2.2)$$

The optimization from Equation 2.2 is usually left up to the UAV operator’s experience and intuition. In Section 5, we detail a novel method for implicitly learning the selection of artistic parameters depending on the scene’s context.

2.3 Related Work

Our work exploits synergies at the confluence of several domains of research to develop an aerial cinematography platform that can follow dynamic targets in unstructured environments using onboard sensors and computing power. In this section we briefly describe related works in different areas that come together under the problem definition described in Section 2.2.

First, we look at works that develop fundamental ideas for automating camera movement in virtual environments, away from the limitations of physics and robotics systems. Second, we delve into prior works on aerial cinematography, and describe their main achievements and limitations. Next, we touch the domain of learning subjective cost functions, looking at works that attempt to rank the quality of images and movements using human feedback. Finally, we describe the extensive literature of visual estimation of object poses, focused on the context of fast and low-cost inference. We also provide additional background materials in each of the next chapters which are useful for understanding the proposed problem formulations and algorithms.

2.3.1 Virtual cinematography

Camera control in virtual cinematography has been extensively examined by the computer graphics community as reviewed by [48]. These methods typically reason about the utility

of a viewpoint in isolation, follow artistic principles and composition rules [5, 26] and employ either optimization-based approaches to find good viewpoints or reactive approaches to track the virtual actor. The focus is typically on through-the-lens control where a virtual camera is manipulated while maintaining focus on certain image features [63, 81, 145, 146]. However, virtual cinematography is free of several real-world limitations such as robot physics constraints and assumes full knowledge of the environment.

Several works analyze the choice of which viewpoint to employ for a particular situation. For example, in [63], the researchers use an A* planner to move a virtual camera in pre-computed indoor simulation scenarios to avoid collisions with obstacles in 2D.

2.3.2 Automatic movie editing

We also find works such as [141] that post-processes videos of a scene taken from different angles by automatically labeling features of different views. The approach uses high-level user-specified rules which exploit the labels to automatically select the optimal sequence of viewpoints for the final movie. In addition, [239] help editors by defining a formal language of editing patterns for movies.

[4] shows work where an algorithm automatically selects the best camera for display within a set of social cameras, mounted on the participants of an activity involving multiple participants. They combine the concept of joint social attention together with cinematographic guidelines to select when and where to cut the scenes. [69] take a semi-automated approach towards scene selection within the context of 3-rd party perspective sports footage. The idea of video summarization is also present in various lines of work, such as [221] for pre-recorded videos, [86] for finding the best viewpoints in 360° footages, and in [109] for producing high-speed versions of long videos, also known as hyperlapses.

2.3.3 Autonomous aerial cinematography

There is a rich history of work in autonomous aerial filming. For instance, several works focus on following user-specified artistic guidelines [71, 73, 110, 171] but often rely on perfect actor localization through a high-precision RTK GPS or a motion-capture system. Additionally, although the majority of work in the area deals with collisions between UAV and actors [98, 110, 171], they do not factor in the environment for safety considerations. While there are several successful commercial products, they too have certain limitations such as operating in low speed and low clutter regimes (e.g. DJI Mavic [61]) or relatively short planning horizons (e.g. Skydio R1 [211]). Even our previous work [22], despite handling environmental occlusions and collisions, assumes a prior elevation map and uses GPS to localize the actor. Such simplifications impose restrictions on the diversity of scenarios that the system can handle.

Several contributions on aerial cinematography focus on keyframe navigation. [77, 79, 112, 195, 241] provide user interface tools to re-time and connect static aerial viewpoints to provide smooth and dynamically feasible trajectories, as well as a visually pleasing images. [138] use key-frames defined on the image itself instead of world coordinates.

Other works focus on tracking dynamic targets, and employ a diverse set of techniques for actor localization and navigation. For example, [98, 99] detect the skeleton of targets from visual input, while others approaches rely on off-board actor localization methods from either

motion-capture systems or GPS sensors [22, 71, 73, 110, 171]. These approaches have varying levels of complexity: [22, 73] can avoid obstacles and occlusions with the environment and with actors, while other approaches only handle collisions and occlusions caused by actors. In addition, in our latest work [17] we made two important improvements on top of [22] by including visual actor localization and online environment mapping.

Furthermore, different systems present significant differences in onboard versus off-board computation. We summarize and compare contributions from past works in Table 2.1. It is important to notice that none of the previously published approaches provides a complete solution to the generic aerial cinematography problem using only onboard resources.

Table 2.1: Comparison of dynamic aerial cinematography systems. Notes: ^{*1} only avoid collisions with the actors, but disregard other obstacles in the environment, ^{*2} only avoid occlusions caused by other actors in the environment, but disregard other obstacles, ^{*3} localizes the actor visually only for control of the camera, but use GPS to obtain the actor’s position in global coordinates for planning, ^{*4} define artistic selection as the viewpoint that maximizes the projection of the actor on the image.

	Related works						Our works		
	[71]	[110]	[171]	[73]	[99]	[98]	[22]	[17]	[19]
Online planning	✓	✓	✓	✓	✓	✓	✓	✓	✓
Obstacle avoidance	×	^{*1}	^{*1}	✓	^{*1}	^{*1}	✓	✓	✓
Occlusion avoidance	×	×	^{*2}	✓	×	×	✓	✓	✓
Onboard processing	×	×	×	×	✓	✓	✓	✓	✓
Actor localization	×	×	×	×	✓	✓	^{*3}	✓	✓
Online mapping	×	×	×	×	×	×	×	✓	✓
Shot selection	×	×	×	×	×	^{*4}	×	×	✓

2.3.4 Making artistic choices autonomously

A common theme behind all the work presented so far is that a user must always specify which kind of output they expect from the system in terms of artistic behavior. This behavior is generally expressed in terms of the set of parameters Ω_{art} , and relates to different shot types, camera angles and angular speeds, type of actor framing, etc. If one wishes to autonomously specify artistic choices, two main points are needed: a proper definition of a metric for artistic quality of a scene, and a decision-making agent which takes actions that maximize this quality metric, as explained in Equation 2.2.

Several works explore the idea of learning a beauty or artistic quality metric directly from data. [119] learns a measure for the quality of *selfies*; [67] learn how to generate professional landscape photographs; [76] learn how to transfer image styles from paintings to photographs.

On the action generation side, we find works that have exploited deep reinforcement learning [169] to train models that follow human-specified behaviors. Closest to our work, [47] learns behaviors for which hand-crafted rewards are hard to specify, but which humans find easy to evaluate.

Our work, as described in Chapter 5, brings together ideas from all the aforementioned areas to create a generative model for shot type selection in aerial filming drones which maximizes an artistic quality metric.

2.3.5 Online environment mapping

Dealing with imperfect representations of the world becomes a bottleneck for viewpoint optimization in physical environments. As the world is sensed online, it is usually incrementally mapped using voxel occupancy maps [226]. To evaluate a viewpoint, methods typically raycast on such maps, which can be very expensive [39, 102]. Recent advances in mapping have led to better representations that can incrementally compute the *truncated signed distance field (TSDF)* [126, 175], i.e. return the distance and gradient to nearest object surface for a query. TSDFs are a suitable abstraction layer for planning approaches and have already been used to efficiently compute collision-free trajectories for UAVs [51, 179].

2.3.6 Visual target state estimation

Accurate object state estimation with monocular cameras is critical for many robot applications, including autonomous aerial filming. Two key problems in target state estimation include detecting objects and their inferring their orientation.

Deep learning based techniques have achieved remarkable progress in the area of 2D object detection, such as YOLO (You Only Look Once) [189], SSD (Single Shot Detector) [151] and Faster R-CNN method [190]. These methods use convolutional neural networks (CNNs) for bounding box regression and category classification. They require powerful GPUs, and cannot achieve real-time performance when deployed to the onboard platform. Another problem with off-the-shelf models trained on open datasets is that they do not generalize well to the aerial filming scenario due to mismatches in data distribution due to angles, lighting, distances to actor and motion blur. Later in Section 4.3 we present our approach for obtaining a real-time object detector for our application.

Another key problem in the actor state estimation for aerial cinematography is estimating the heading direction of objects in the scene. Heading direction estimation (HDE) has been widely studied especially in the context of humans and cars as target objects. There have been approaches that attach sensors including inertial sensors and GPS to the target object to obtain the object’s [58, 148][234] heading direction. While these sensors provide reliable and accurate estimation, it is highly undesirable for the target actor to carry these extra sensors. Thus, we primarily focus on vision-based approaches for our work that don’t require the actor to carry any additional equipment.

In the context of Heading Direction Estimation using visual input, there have been approaches based on classical machine learning techniques. Based on a probabilistic framework, [68] present a joint pedestrian head and body orientation estimation method, in which they design a HOG based linear SVM pedestrian model. Learning features directly from data rather than handcrafting them has proven more successful, especially in the domain of computer vision. Deep learning based approaches have been successfully applied to the area of 2D pose estimation [34, 230] which is a related problem. However, the 3D heading direction cannot be trivially recovered from 2D points because the keypoint’s depth remains undefined and ambiguous. Also, these approaches are primarily focused on humans and don’t address other objects including cars.

There are fewer large scale datasets for 3D pose estimation [80, 101, 152, 185] and the existing ones generalize poorly to our aerial filming task, again due to mismatch in the data distribution. Thus, we look for approaches that can be applied in limited labeled data

setting. The limited dataset constraint is common in many robotics applications, where the cost of acquiring and labeling data is high. Semi-supervised learning (SSL) is an active research area in this domain. However, most of the existing SSL works are primarily focused on classification problems [54, 95, 186, 238], which assume that different classes are separated by a low-density area and easy to separate in high dimensional space. This assumption is not directly applicable to regression problems.

In the context of cinematography, temporal continuity can be leveraged to formulate a semi-supervised regression problem. [170] developed one of the first approaches to exploit temporal continuity in the context of deep convolutional neural networks. The authors use video temporal continuity over the unlabeled data as a pseudo-supervisory signal and demonstrate that this additional signal can improve object recognition in videos from the COIL-100 dataset [174]. There are other works that learn feature representations by exploiting temporal continuity [83, 214, 216, 237, 250]. [250] included the video temporal constraints in an autoencoder framework and learn invariant features across frames. [237] designed a Siamese-triplet network which can be trained in an unsupervised manner with a large amount of video data, and showed that the unsupervised visual representation can achieve competitive performance on various tasks, compared to its ImageNet-supervised counterpart. Inspired by these approaches, our recent work [236] aims to improve the learning of a regression model from a small labeled dataset by leveraging unlabeled video sequences to enforce temporally smooth output predictions.

After the target’s location and heading direction is estimated on the image plane, we can project it onto the world coordinates and use different methods to estimate the actor’s future motion. Motion forecast methods can range from filtering methods such as Kalman filters and extended Kalman filters [226], which are based solely on the actor’s dynamics, to more complex methods that take into account environmental features as well. As an example of the latter, [232] use traditional motion planner with handcrafted cost functions for navigation among obstacles, and [246] use deep inverse reinforcement learning to predict the future trajectory distribution vehicles among obstacles.

Chapter 3

Planning for moving cameras

Motion planning problems in robotics typically involve moving the vehicle’s base (in the case of mobile robots) or an end-effector (in the case of manipulators) from point A to point B. While doing so two constraints are enforced. First, the calculated motions must be dynamically feasible for the robot’s controller, as there’s no utility in a plan that intrinsically cannot be followed for being too aggressive. Second, the robot must remain safe, out of collisions with the environment and with itself. The problem of cinematography, however, presents a unique set of challenges for the planning system.

When planning for the motion of moving cameras, the objective function needs to include additional constraints to calculate trajectories for the UAV because we are filming a moving actor. In addition to staying safe, we need to ensure target visibility among obstacles. And most importantly, given an infinity of possible viewpoints to a target, we must ensure that the configurations we select are as close as possible to the artistic guidelines desired for that scene. For now we assume that guidelines are given to us as a parameter, however, later in Chapter 5 we investigate learning artistic decision-making on-the-fly.

This chapter’s objective is to define a theoretical framework for planning the motion of physical cameras in space. First, we detail the definition of trajectories and cost functions, and describe an efficient optimization-based algorithm for finding the best locally optimal camera trajectories. Next, we describe implementation details and validate our algorithms in a simulation and real-world experiments. The supplementary video shows more details of footages captured by our system: <https://youtu.be/QX73nBBwd28>.

3.1 UAV Trajectory Definition

Recall Section 2.2, where we defined $\xi_q(t) : [0, t_f] \rightarrow \mathbb{R}^3 \times SO(2)$ as the UAV trajectory, where $\xi_q(t) = \{x_q(t), y_q(t), z_q(t), \psi_q(t)\}$. Analogously, we define a trajectory for the actor as $\xi_a(t) : [0, t_f] \rightarrow \mathbb{R}^3 \times SO(2)$, where $\xi_a(t) = \{x_a(t), y_a(t), z_a(t), \psi_a(t)\}$. For now, we assume that ξ_a is given to us by a motion estimation module, and ξ_q is the variable we wish to estimate.

We assume that our UAV counts with a gimbal controller that can orient the camera independently of the UAV’s body motion. Therefore, we can purposefully decouple the UAV body’s heading ψ_q from the main motion planning algorithm. We set $\psi_q(t)$ to always

point the UAV's front towards the actor's current location, along the orientation connecting $\xi_q(t) \rightarrow \xi_a(t)$, as seen in Equation 3.1.

$$\psi_q(t) = \text{atan2}(y_a(t) - y_q(t), x_a(t) - x_q(t)) \quad (3.1)$$

This assumption significantly reduces the complexity of the planning problem by removing four degrees of freedom: three from the camera's orientation and one from the UAV's heading. In practice, this design choice also significantly improves filming performance because the camera can be controlled directly from image feedback, without the accumulation of cascading errors from the raycasting and motion estimation modules (as described in more detail in Chapter 4).

Now, let $\xi_q(t)$ represent the UAV's trajectory in a continuous time-parametrized form, and let ξ_q represent the same trajectory in a finite discrete form, with total time length t_f . Let point p_0 represent the contour conditions where the trajectory begins. ξ_q contains a total of $n - 1$ variables of the form p_i , where $i = 1, 2, \dots, n - 1$, as shown in Equation 3.2.

$$\xi_q = \begin{bmatrix} p_1 \\ p_2 \\ \vdots \\ p_{n-1} \end{bmatrix} = \begin{bmatrix} p_{1x} & p_{1y} & p_{1z} \\ p_{2x} & p_{2y} & p_{2z} \\ \vdots & \vdots & \vdots \\ p_{n-1\ x} & p_{n-1\ y} & p_{n-1\ z} \end{bmatrix} \quad (3.2)$$

3.2 Defining Cost Functions and a Planner for Aerial Filming

As explained in Section 2.2, in a generic aerial filming framework we want trajectories which are smooth (J_{smooth}), capture high quality viewpoints (J_{shot}), avoid occlusions (J_{occ}) and keep the UAV safe (J_{obs}). Each objective can then be encoded in a separate cost function, and the motion planner's objective is to find the trajectory that minimizes the overall cost, subject to the initial condition constraints. We assume that the total cost is a linear combination of individual cost functions. For the sake of completeness, we repeat Equation 2.1 below as Equation 3.3:

$$\begin{aligned} \xi_q^* &= \arg \min_{\xi_q} J(\xi_q) , \\ \text{s.t.} \quad &\begin{cases} \xi_q(0) = p_0 = \{x_0, y_0, z_0, \psi_0\} \\ J(\xi_q) = \begin{bmatrix} 1 & \lambda_1 & \lambda_2 & \lambda_3 \end{bmatrix} \begin{bmatrix} J_{\text{smooth}}(\xi_q) \\ J_{\text{obs}}(\xi_q, \mathcal{M}) \\ J_{\text{occ}}(\xi_q, \xi_a, \mathcal{M}) \\ J_{\text{shot}}(\xi_q, \xi_a, \Omega_{\text{art}}) \end{bmatrix} \end{cases} \end{aligned} \quad (3.3)$$

Different UAV applications can influence the choice of motion planning algorithms and cost function formulations. The main motivation is that different types of planners can exploit specific properties and guarantees of the cost functions. For example, sampling-based planners [64, 117, 134] or search-based planners [2, 139] should ideally use fast-to-compute costs so that many different states can be explored during search in high-dimensional state spaces. Other categories of planners, based on trajectory optimization [187, 206], usually

require cost functions to be differentiable to the first or higher orders. We also find hybrid methods that make judicious use of optimization combined with search or sampling [45, 156].

Our choices of cost functions and planning algorithm are dictated by two main observations:

- First, we consider the fact that good camera positioning requires the UAV to reason over relatively long time horizons, and not act reactively purely based on the current image. Reactive operations are, for example, the avoidance of small local obstacles such as branches or light posts. However, while filming we need to consider how larger obstacles such as trees, buildings, and terrain elevations may affect image generation, which can only happen if we reason over horizons in the order of magnitude of around 10 seconds. In practice, longer time horizons are limited by the actor motion forecast module, which decreases rapidly with time.
- Second, we need to take into account that following dynamic targets requires a high planning frequency. The actor is constantly changing direction and velocity, and the environment map is continuously evolving based on sensor readings. Our motion plan needs to quickly evolve based on new updates.

Our observations present conflicting objectives: on the one hand we need a computationally heavy planner for long time horizons, and on the other hand it needs to re-plan fast to keep up with the changes in unscripted scenes. To solve this dilemma we chose a motion planner based on local trajectory optimization. Optimizations are fast and reason over a smooth continuous space of trajectories. In addition, higher-order information from the cost functions can help accelerate solution convergence, and plans can be incrementally updated across planning cycles. The downside from using local optimization is the absence of global optimality in our solutions. Within the context of filming, however, locally optimal solutions are almost always of acceptable quality, and a fast planner with a longer horizon is more advantageous than a slow globally optimal one.

A popular optimization-based approach that addresses the aerial filming requisites is to cast the problem as an unconstrained cost optimization, and apply covariant gradient descent (such as CHOMP, from [188, 251]). These quasi-Newton methods require that some of the objectives have analytic Hessians that are easy to invert and that are well-conditioned. With the use of first and second order information about the problem, such methods exhibit fast convergence while being stable and computationally inexpensive. The use of such quasi-Newton methods requires a set of differentiable cost functions for each objective, which we detail next.

3.2.1 Definition of Cost Functions

a) Smoothness

A smoothness metric ensures that the output trajectory can be properly executed by the UAV. In addition, penalizing jerkiness in trajectories can have a significant impact on video quality. We measure smoothness as the cumulative sum of n -th order derivatives of the

trajectory, following the rationale of [187]. Let D be a discrete difference operator. The smoothness cost is:

$$J_{\text{smooth}}(\xi_q(t)) = \frac{1}{t_f} \frac{1}{2} \int_0^{t_f} \sum_{d=1}^{d_{\max}} \alpha_n (D^d \xi_q(t))^2 dt, \quad (3.4)$$

where α_n is a weight for different orders, and d_{\max} is the number of orders. In practice, we penalize derivatives up to the third order, setting $\alpha_n = 1, d_{\max} = 3$.

Appendix A.1 expands upon this cost function and reformulates it in matrix form using auxiliary matrices A_{smooth} , b_{smooth} , and c_{smooth} . We state the cost, gradient and Hessian for completeness:

$$\begin{aligned} J_{\text{smooth}}(\xi_q) &= \frac{1}{2(n-1)} \text{Tr}(\xi_q^T A_{\text{smooth}} \xi_q + 2\xi_q^T b_{\text{smooth}} + c_{\text{smooth}}) \\ \nabla J_{\text{smooth}}(\xi_q) &= \frac{1}{(n-1)} (A_{\text{smooth}} \xi_q + b_{\text{smooth}}) \\ \nabla^2 J_{\text{smooth}}(\xi_q) &= \frac{1}{(n-1)} A_{\text{smooth}} \end{aligned} \quad (3.5)$$

b) Shot quality

First, we analytically define the the artistic shot parameters. Based on cinematography literature [5, 26], we select a minimal set of parameters that compose most of the shots possible for single-actor, single-camera scenarios. We define Ω_{art} as a set of three parameters: $\Omega_{\text{art}} = \{\rho, \psi_{\text{rel}}, \phi_{\text{rel}}\}$ (Fig. 3.1), where:

- ρ is the shot scale, which can be mapped to the distance between actor and camera
- ψ_{rel} is the relative yaw angle between actor and camera
- ϕ_{rel} is the relative tilt angle between the actor's current height plane and the camera

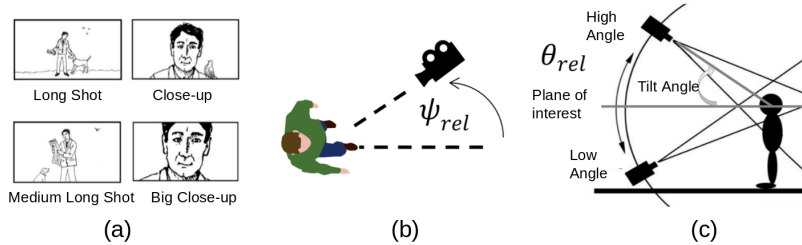


Figure 3.1: Shot parameters Ω_{art} for shot quality cost function, adapted from [26]: a) shot scale ρ corresponds to the size of the projection of the actor on the screen; b) line of action angle $\psi_{\text{rel}} \in [0, 2\pi]$; c) tilt angle $\theta_{\text{rel}} \in [-\pi, \pi]$.

Given a set Ω_{art} , we can now define a desired cinematographic trajectory $\xi_{\text{shot}}(t)$:

$$\xi_{\text{shot}}(t) = \xi_a(t) + \rho \begin{bmatrix} \cos(\psi_a + \psi_{\text{rel}}) \sin(\theta_{\text{rel}}) \\ \sin(\psi_a + \psi_{\text{rel}}) \cos(\theta_{\text{rel}}) \\ \cos(\theta_{\text{rel}}) \end{bmatrix} \quad (3.6)$$

The desired artistic trajectory is not necessarily achievable by the UAV. It may be in direct collision with obstacles, or depending on the actor’s forecast, it may be dynamically infeasible for the drone. Therefore, instead of imposing a hard constraint for achieving the desired positions, we define a shot quality metric. We use the L^2 norm to define the shot quality cost function as the distance between the current camera trajectory and the desired cinematography path:

$$J_{\text{shot}}(\xi_q, \xi_a) = \frac{1}{t_f} \frac{1}{2} \int_0^{t_f} \|\xi_q(t) - \xi_{\text{shot}}(\xi_a(t))\|^2 dt \quad (3.7)$$

Appendix A.2 expands upon this cost function and reformulates it in matrix form using auxiliary matrices A_{shot} , b_{shot} , and c_{shot} . Again, we state the cost, gradient and Hessian for completeness:

$$\begin{aligned} J_{\text{shot}}(\xi_q, \xi_a) &= \frac{1}{2(n-1)} \text{Tr}(\xi_q^T A_{\text{shot}} \xi_q + 2\xi_q^T b_{\text{shot}} + c_{\text{shot}}) \\ \nabla J_{\text{shot}}(\xi_q) &= \frac{1}{(n-1)} (A_{\text{shot}} \xi_q + b_{\text{shot}}) \\ \nabla^2 J_{\text{shot}}(\xi_q) &= \frac{1}{(n-1)} A_{\text{shot}} \end{aligned} \quad (3.8)$$

We note that although the artistic parameters of the shot quality cost described in this work are defined for single-actor single-camera scenarios, the extension of J_{shot} to multi-actor scenarios is trivial. It can be achieved by defining an artistic guideline ξ_{shot} using multi-actor parameters such as the angles with respect to the line of action [26], or geometric center of the targets. We detail more possible extensions of our work in Section 4.6.

c) Obstacle Avoidance

The obstacle avoidance cost function keeps the vehicle safe, out of collisions with the environment, and away from the actor as well. For its computation, we assume that we have access to an occupancy grid \mathcal{G} that maps all obstacles in the environment. Using \mathcal{G} we can calculate a truncated signed distance (TSDF) map $\mathcal{M} : \mathbb{R}^3 \rightarrow \mathbb{R}$, that contains the distance and direction of any point to the nearest obstacle. Section 4.2 describes more details on how we can calculate a map on-the-fly, but for now we can assume that we have a complete map of the environment.

First, for any point p in space, we adopt the obstacle avoidance function from [251]. This function linearly penalizes the intersection with obstacles, and decays quadratically with distance, up to a threshold ϵ_{obs} :

$$c(p) = \begin{cases} -\mathcal{M}(p) + \frac{1}{2}\epsilon_{\text{obs}} & \mathcal{M}(p) < 0 \\ \frac{1}{2\epsilon_{\text{obs}}}(\mathcal{M}(p) - \epsilon_{\text{obs}})^2 & 0 < \mathcal{M}(p) \leq \epsilon_{\text{obs}} \\ 0 & \text{otherwise} \end{cases} \quad (3.9)$$

Similarly to [251], now define a safety cost function for the entire UAV trajectory:

$$J_{\text{obs}}(\xi_q, \mathcal{M}) = \int_{t=0}^{t_f} c(\xi_q(t)) \left| \frac{d}{dt} \xi_q(t) \right| dt \quad (3.10)$$

We can differentiate J_{obs} with respect to a point at time t_i to obtain the cost gradient (note that $\hat{v} = \frac{v}{|v|}$ denotes a normalized vector):

$$\begin{aligned} \nabla J_{\text{obs}}(\xi_q(t_i), \mathcal{M}) &= \nabla J_{\text{obs}}(p_i, \mathcal{M}) = |\dot{p}_i| \left[\left(I - \hat{p}_i \hat{p}_i^T \right) \nabla c(p_i) - c(p_i) \kappa \right], \\ \text{where: } \kappa &= \frac{1}{|\dot{p}_i|^2} (I - \hat{p}_i \hat{p}_i^T) \ddot{p}_i \end{aligned} \quad (3.11)$$

In practice, we use discrete derivatives to calculate \mathcal{M} , the velocities \dot{p}_i , and accelerations \ddot{p}_i .

d) Occlusion Avoidance

Even though the concept of occlusion is binary, *i.e.*, we either have or don't have visibility of the actor, a major contribution of our work [22] was to define a differentiable cost that expresses a viewpoint's occlusion intensity among arbitrary obstacle shapes. The fundamental idea behind this cost is the following: given the actor's trajectory forecast (ξ_a) and the current camera path (ξ_q), it measures along how much obstacle blockage the light rays that connect both paths would have to go through. For illustration purposes, Figure 3.2 shows the concept of occlusion for motion in a 2D environment, even though our problem operates with occlusions in a 3D setting.

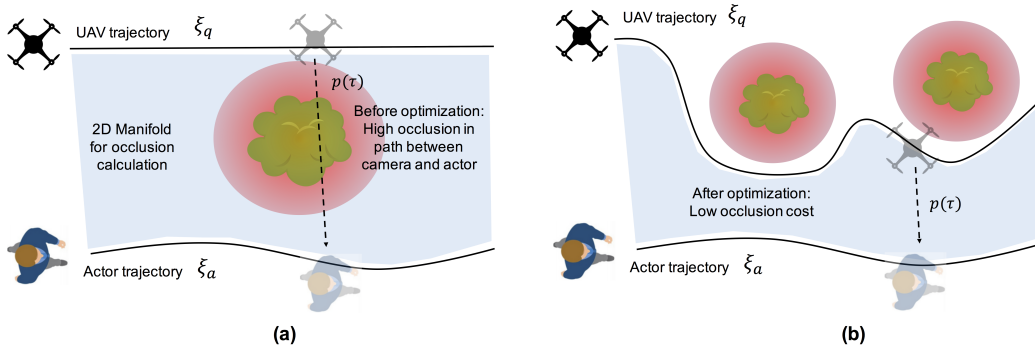


Figure 3.2: Occlusion cost representation. (a) For every pair of UAV-actor position, we integrate the penalization c on the signed distance field over a 2D manifold. (b) After optimization, occlusion gradients pull the UAV trajectory towards regions with full visibility of the actor.

Mathematically, we define occlusion as the integral of the TSDT cost c over a 2D manifold connecting both trajectories ξ_q and ξ_a . The manifold is built by connecting each UAV-actor position pair at time t using the parametrized path $p(\tau)$, where $p(\tau = 0) = \xi_q(t)$ and $p(\tau = 1) = \xi_a(t)$:

$$J_{\text{occ}}(\xi_q, \xi_a, \mathcal{M}) = \int_{t=0}^{t_f} \left(\int_{\tau=0}^1 c(p(\tau)) \left| \frac{d}{d\tau} p(\tau) \right| d\tau \right) \left| \frac{d}{dt} \xi_q(t) \right| dt \quad (3.12)$$

We can derive the functional gradient with respect to a point p_i at time t_i , resulting in:

$$\begin{aligned} \nabla J_{\text{occ}}(\xi_q, \xi_a, \mathcal{M})(t_i) = \int_{\tau=0}^1 \nabla c(p(\tau)) |L| |\dot{q}| \left[I - (\hat{q} + \tau(\frac{\dot{a}}{|\dot{q}|} - \hat{q})) \hat{q}^T \right] \\ - c(p(\tau)) |\dot{q}| \left[\hat{L}^T + \frac{\hat{L}^T \dot{L} \hat{q}^T}{|\dot{q}|} + |L| \kappa^T \right] d\tau, \end{aligned} \quad (3.13)$$

where:

$$q = \xi_q(t_i), \quad a = \xi_a(t_i), \quad p(\tau) = (1-\tau)q + \tau a, \quad L = a - q, \quad \hat{v} = \frac{v}{|v|}, \quad \kappa = \frac{1}{|\dot{q}|^2} (I - \hat{q} \hat{q}^T) \ddot{q} \quad (3.14)$$

Intuitively, the term multiplying $\nabla c(p(\tau))$ is related to variations of the signed distance gradient in space, with the rest of the term acting as a lever to deform the trajectory. The term $c(p(\tau))$ is linked to changes in path length between camera and actor.

3.2.2 Trajectory Optimization Algorithm

Our objective is to minimize the total cost function $J(\xi_q)$ (3.3). We do so by employing covariant gradient descent, using the gradients of the cost function $\nabla J(\xi_q)$, and a analytic approximation of the Hessian $\nabla^2 J(\xi_q) \approx (A_{\text{smooth}} + \lambda_3 A_{\text{shot}})$:

$$\xi_q^+ = \xi_q - \frac{1}{\eta} (A_{\text{smooth}} + \lambda_1 A_{\text{shot}})^{-1} \nabla J(\xi_q) \quad (3.15)$$

In the optimization context, $\nabla^2 J(\xi_q)$ acts as a metric to guide the solution towards a direction of steepest descent on the functional cost. This step is repeated until convergence. We follow two conventional stopping criteria for descent algorithms based on current cost landscape curvature and relative cost decrease [27], and limit the maximum number of iterations. We use the current trajectory as initialization for the next planning problem, appending a segment with the same curvature as the final points of the trajectory for the additional points until the end of the time horizon. This step ensures that our previous solution can be re-used in the following planning problem, greatly increasing convergence to a point of local minimum in the cost landscape.

Note in Algorithm 1 one of the main advantages of the CHOMP algorithm [187]: we only perform the Hessian matrix inversion once, outside of the main optimization loop, rendering good convergence rates [22]. By fine-tuning hyper-parameters such as trajectory discretization level, trajectory time horizon length, optimization convergence thresholds, and relative weights between costs, we can achieve a re-planning frequency of approximately 5Hz on the onboard hardware, considering a 10s horizon. These are adequate parameters for safe

and non-myopic operations in our environments, but significantly higher frequencies can be achieved with the same underlying algorithm depending on application-specific parameters.

Algorithm 1: Optimize (ξ_q)

```

1  $M_{inv} \leftarrow (A_{smooth} + \lambda_3 A_{shot})^{-1} \triangleright$  Hessian inversion happens only once
2 for  $i = 0, 1, \dots, i_{max}$  do
3   if  $(\nabla J(\xi_{qi})^T M_{inv} \nabla J(\xi_{qi}))^2 / 2 < \epsilon_0$  or  $(J(\xi_{qi}) - J(\xi_{qi-1})) < \epsilon_1$  then
4     return  $\xi_{qi}$ 
5   end
6    $\xi_{qi+1} = \xi_{qi} - \frac{1}{\eta} M_{inv} \nabla J(\xi_{qi})$ 
7 end
8 return  $\xi_{qi}$ 

```

The resulting trajectory from the most recent plan is appended to the UAV’s trajectory tracker, which uses a PD controller to send velocity commands to the aircraft’s internal controller.

3.3 Experiments

We validate the cost functions and optimization-based planner developed for aerial cinematography in simulated and real-world environments.

3.3.1 Simulation experiments

a) Quantitative role of the occlusion cost function:

We evaluate our planning algorithm on environments with increasing levels of randomized clutter, as seen in Figure 3.3. Table 3.1 summarizes the planner performance in different environments in terms of actor visibility and the average distance to the artistically desired trajectory. By using the occlusion cost function, we improve actor visibility by over 10% in comparison with pure obstacle avoidance in environments with 40 random spheres; however, the trade-off is an increase in the average distance to the desired artistic trajectory.

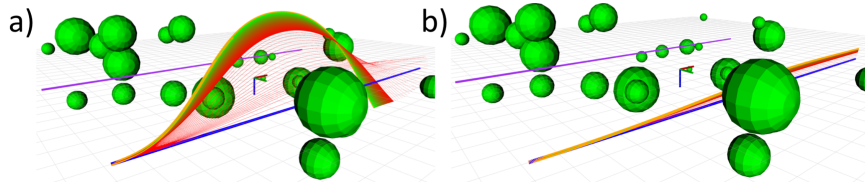


Figure 3.3: Randomized environment with obstacles to evaluate planner robustness. a) Solution including occlusion cost function, and b) Pure obstacle avoidance.

b) Advantage of longer planning horizons:

We evaluate the overall system behavior when using different planning time horizons between 1 and 20 seconds, as seen in Table 3.2. Short horizons reason myopically about the environment, and cannot render robust and safe behavior in complex scenes, thus increasing the normalized cost per time length of the resulting trajectory. Figure 3.4 displays the

Table 3.1: Evaluation of motion planner performance in the randomized environment from Fig 3.3. Statistics computed using 100 random configurations for each environment complexity level.

Success metric	Cost functions	Num. of spheres in environment		
		1	20	40
Actor visibility along trajectory	$J_{\text{occ}} + J_{\text{obs}}$	$99.4 \pm 2.2\%$	$94.2 \pm 7.3\%$	$86.9 \pm 9.3\%$
	J_{obs}	$98.8 \pm 3.0\%$	$87.1 \pm 8.5\%$	$75.3 \pm 11.8\%$
Avg. dist. to ξ_{shot} , in m	$J_{\text{occ}} + J_{\text{obs}}$	0.4 ± 0.4	6.2 ± 11.2	10.7 ± 13.2
	J_{obs}	0.05 ± 0.1	0.3 ± 0.2	0.5 ± 0.3

qualitative difference between trajectories, keeping all variables except planning horizon constant.

Table 3.2: Performance of motion planner with varying planning time horizons for the environment shown in Fig 3.4. Longer planning horizons allow better reasoning for safety and occlusion avoidance, lowering the normalized planning cost. However, longer horizons naturally increase planning computing time.

Planning horizon length [s]:	1.0	5.0	10.0	20.0
Normalized trajectory cost [J/t_f]	0.0334	0.0041	0.0028	0.0016
Computing time [ms]	0.0117	0.0131	0.0214	0.0343

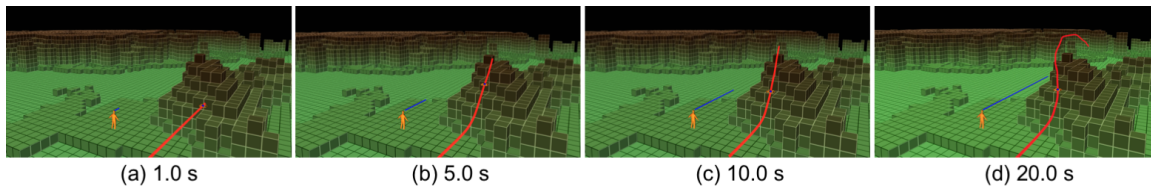


Figure 3.4: Planner behavior with different time horizons of 1 (a), 5 (b), 10 (c), and 20 (d) seconds for same actor trajectory and environment. The shortest time horizon of 1s is not sufficient for the planner to find a trajectory that avoids the mound, and the vehicle gets stuck in a bad local minimum of solutions. Longer horizons let the UAV plan more intelligent trajectories, reasoning about obstacle shapes long before the UAV reaches those positions.

3.3.2 Field experiments

We use a DJI M210 drone model, coupled with a NVIDIA TX2 computer for both the vision and planning pipelines (Figure 3.5). A ROS [197] infrastructure handles node communication.

For this set of outdoors experiments we use a conventional GPS antenna for localizing the actor’s position in the environment. This choice results in high noise for the actor motion prediction step. Therefore, we decided to decouple the motion of the drone and the camera. The camera is mounted on a 3-axis independent gimbal and can place the actor on the correct

screen position by visual detection, despite errors in the drone’s position. The camera’s roll is stabilized, and the vision pipeline actively controls its pitch and yaw.

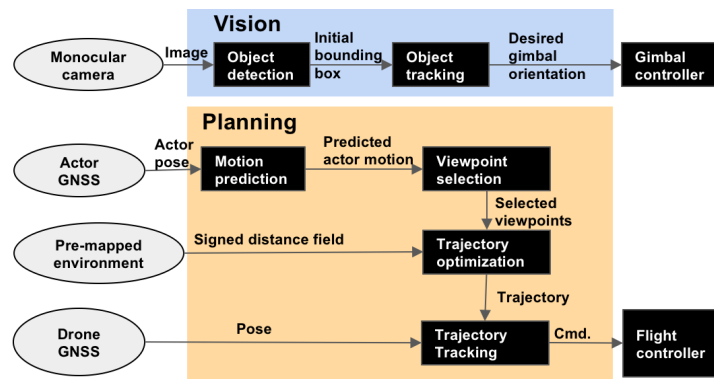


Figure 3.5: System architecture. The vision subsystem controls the camera orientation using only the monocular image, independently of the planning subsystem. Planning uses the drone’s and actor’s current location, and the environment to generate trajectories for the flight controller.

In the vision pipeline, we detect the actor using Faster-RCNN [190] with MobileNets [96] for feature extraction and Kernelized Correlation Filter (KCF) [91] for tracking, controlling the camera gimbal to keep the actor within the desired screen position. The actor wears a Pixhawk PX4 module on a hat that sends his pose to the onboard computer via radio communication. A linear Kalman filter uses these pose estimates to infer his velocity, which is used to predict her trajectory (ξ_a) for the next 10 s. Using a point cloud map of the test site we pre-compute a TSDF map of the region of interest. Re-planning happens at 5 Hz with a 10 s horizon. To simulate the full pipeline and to decide on the relative weights between each cost function (Eq. 3.3), we built a ROS wrapper to test our software in a photo-realistic environment [207] (Fig. 3.6).

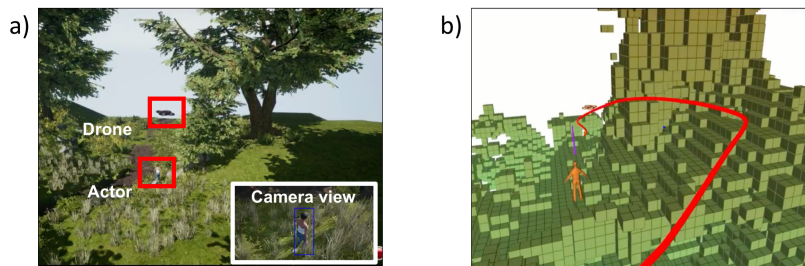


Figure 3.6: Photo-realistic simulator used to test the system. Third and first-person renderings shown on the left, and occupancy map with drone trajectory shown on the right.

a) Qualitative role of the occlusion cost function:

For this experiment, unlike the tests with visual actor localization, we detect the actor using a ground-truth GPS tag. We set up the motion planner to calculate the UAV paths with and without the occlusion cost function, keeping all other scenario variables equal. As seen in Figure 3.7, our proposed occlusion cost significantly improves the aesthetics of the resulting

image, keeping the actor visibility. In addition to aesthetics, maintaining actor visibility is a vital feature of our architecture, allowing vision-based actor localization.

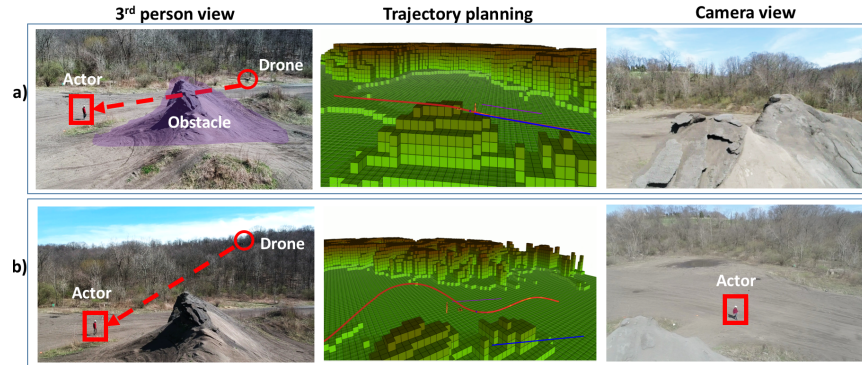


Figure 3.7: Comparison of planning a) without and b) with occlusion cost function. The occlusion cost function significantly improves the quality of the camera image in comparison with pure obstacle avoidance, for same shot type.

b) Algorithm robustness:

We evaluated our algorithm performing different types of static and dynamic shots, following different types of actors: humans, cars and bicycles at different speeds and motion types. In total, we collected over 1.25 hours of flight time while re-planning and avoided obstacles and/or occlusion 65 times. The maximum velocity achieved during the tests was of 7.5 m/s. Figure 3.8 summarizes the most representative shots, which are also shown in the supplementary video.

3.4 Conclusion

In this chapter we proposed and validated a motion planning algorithm and system for autonomous aerial cinematography. Our system is able to execute aesthetically pleasing shots, calculating trajectories that balance motion smoothness, occlusion, and cinematography guidelines. Our experimental results show the algorithm's robustness and speed in a real-life scenarios outdoors, even under noisy actor predictions. In addition, we show that the occlusion cost function we introduced significantly improves the quality of the resulting image, and works for arbitrary obstacle shapes.

There are many key aspects that still need to be solved for us to create a fully autonomous filming system. In terms of actor localization, instead of relying on a GPS antenna on the actor's head, we should be able to use only visual inputs to identify the target's position, with no need for GPS. In addition, we can incorporate an online-mapping module to the vehicle to allow us to film in unknown regions of space. Chapter 4 introduces both of these improvements into our system. Furthermore, instead of relying on artistic parameters determined before flight by a director, in Chapter 5 we investigate techniques for automatically selecting the best shots to be executed for a particular scenario, taking into account motion cues of the actor and obstacle configurations. Later in Chapter 7 we also detail the adaption of the current algorithm to multi-drone configurations by considering additional cost functions such as inter-drone sight and artistic coordination.

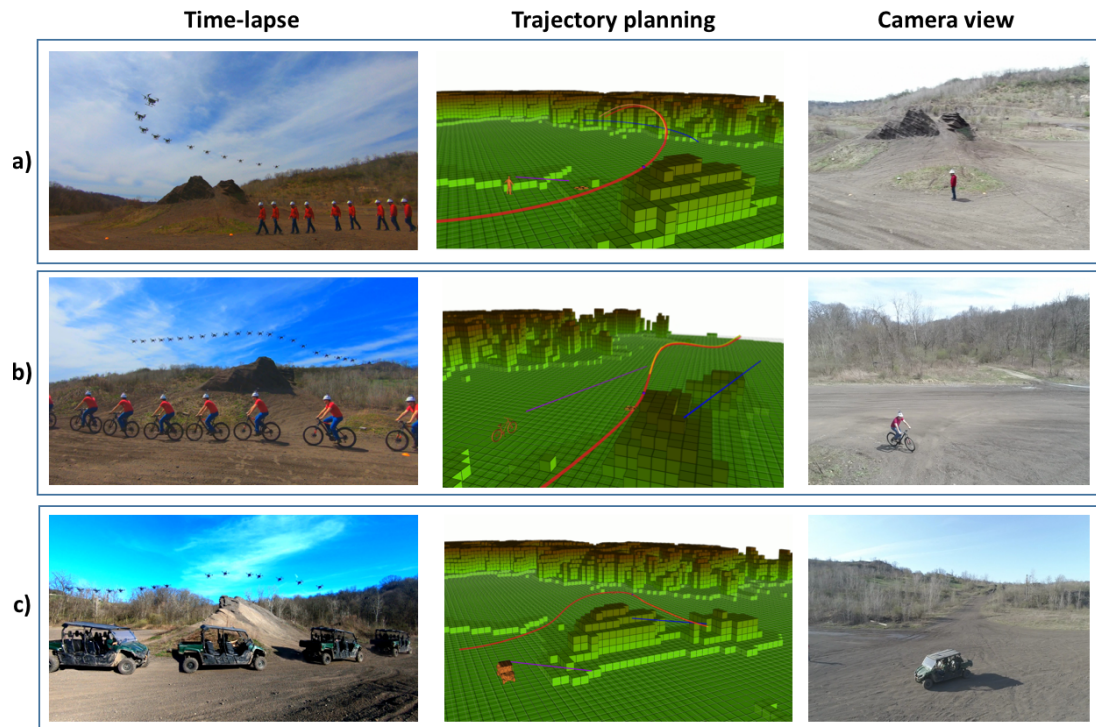


Figure 3.8: Results: a) Circling shot around person, b) Side shot following biker, c) Side shot following vehicle. The planned trajectory (red) avoids colliding with and being occluded by the mountain, while remaining smooth even under high actor motion prediction noise. The actor's motion forecast is in purple, and the desired artistic shot is in blue.

Chapter 4

Building a System for Filming in the Wild

In the previous chapter we formalized the motion planning problem for aerial cameras. Even though we validated our theoretical results in simulated scenarios and field experiments, our tests contained simplifying assumptions that restrict system deployment *in the wild*.

For example, our previous experiments assumed that a complete map of the environment was available to the motion planning system. In practice, a UAV should be able to operate among unstructured object configurations which are unknown prior to takeoff. In addition, the experiments in Chapter 3 counted with a GPS tag and a compass attached to the actor’s head for target localization and motion estimation. Even though we were able to detect the position of the actor on the screen space for gimbal positioning, we could not estimate the actor’s position and orientation in 3D just using monocular images.

This chapter’s objective is to describe the development of an aerial filming system that can be deployed in real-world conditions. First, we describe our main design hypothesis and requirements, followed by the system’s architecture. Next, we provide details on the mapping and actor visual localization modules, and show how they differ from modules found in standard UAV applications. Finally, we validate our system deployment with extensive field experiments and report on the main lessons learned in our tests.

The supplementary videos show more details of footages captured by our system <https://youtu.be/ZE9MnCVmumc>, and of the computer-vision module: <https://youtu.be/-UVSXSxtKN4>.

4.1 System Overview

In this section we detail the design hypotheses that influenced the system architecture layout (Subsec. 4.1.2), as well as our hardware (Subsec. 4.1.3) and simulation (Subsec. 4.1.4) platforms.

4.1.1 Design Hypotheses

Given the application challenges (Subsec. 1.2) and problem definition (Sec. 2.2), we formalize three key hypotheses to guide the layout of the system architecture for the task of autonomous aerial cinematography *in the wild*. These hypotheses serve as high-level principles for our choice of sub-systems, sensors and hardware. We evaluate the hypotheses later in Section 4.4, where we present our simulation and field experiments.

Hyp. 1 *Onboard sensors provide sufficient information for good cinematography performance.*
This is a fundamental assumption and a necessary condition for development of real-world aerial cinematography systems that do not rely on ground-truth data from off-board sensors. We hypothesize our system can deal with noisy measurements and extract necessary actor and obstacle information for visual actor localization, mapping and planning.

Hyp. 2 *Decoupling gimbal control from motion planning improves real-time performance and robustness to noisy actor measurements.*

We assume that an independent 3-DOF camera pose controller can compensate for noisy actor measurements. We expect advantages in two sub-systems: (i) the motion planner can operate faster and with a longer time horizon due to the reduced trajectory state space, and (ii) visual tracking will be more precise because the controller uses direct image feedback instead of a noisy estimate of the actor’s location. We use a gimballed camera with 3-DOF control, which is a reasonable requirement given today’s UAV and camera technology.

Hyp. 3 *Analogous to the role of a movie director, a sub-system for deciding artistic intent should only provide high-level guidelines for camera positioning, but not interfere directly on low-level controls.*

We hypothesize that a hierarchical structure to guide artistic filming behavior employing high-level commands is preferable to an end-to-end low-level visio-motor policy because: (i) it’s easier to ensure overall system safety and stability by relying on more established motion planning techniques, and (ii) it’s more data-efficient and easier to train a high-level decision-making agent than an end-to-end low-level policy.

Note: We dedicate Chapter 5 for describing the process of *learning* artistic decisions. In this chapter we assume that high-level artistic parameters are still chosen *a priori* by a human director.

4.1.2 System Architecture

Taking into account the design hypotheses, we outline the software architecture in Figure 4.1. The system consists of 4 main modules: Vision, Mapping, Planning and Artistic Shot Selection. The four modules run in parallel, taking in camera, LiDAR and GPS inputs to output gimbal and flight controller commands for the UAV platform.

Vision: The module takes in monocular images to compute a predicted actor trajectory for the Shot Selection and Planning Module. Following *Hyp. 2*, the vision module also controls the camera gimbal independently of the planning module. Details described in Section 4.3.

Mapping: The module registers the accumulated LIDAR point cloud and outputs different environment representations: obstacle height map for raycasting and shot selection, and truncated signed distance transform (TSDF) map for the motion planner. Details described in Section 4.2.

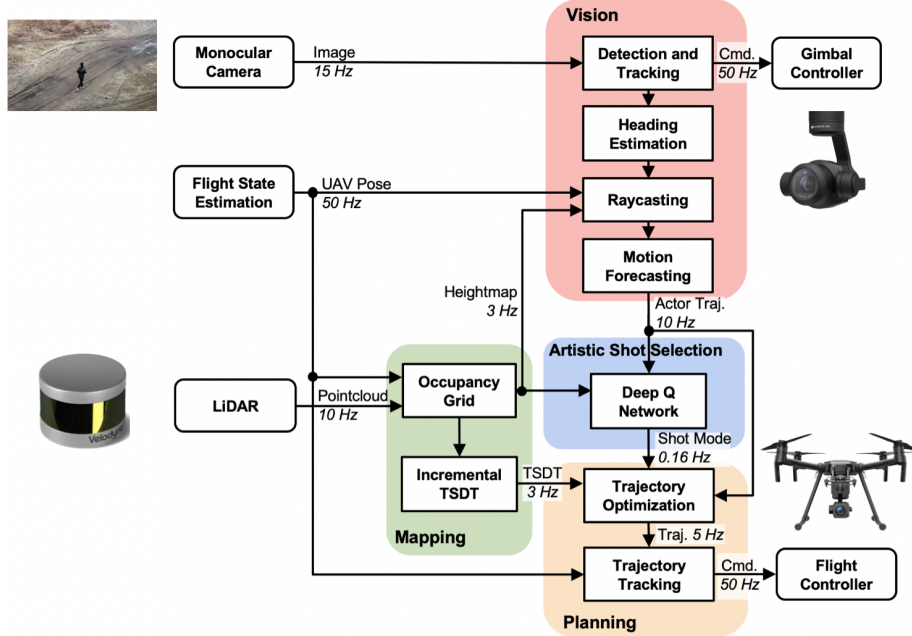


Figure 4.1: The system consists of 4 main modules running in parallel: Vision, Mapping, Planning and Artistic Shot Selection. The system takes in visual, LiDAR and GPS inputs to output gimbal and flight controller commands.

Artistic Shot Selection: Following *Hyp. 3* the module acts as an *artistic movie director* and defines high-level inputs for the motion planner defining the most aesthetic shot type (left, right, front, back) for a given scene context, composed of actor trajectory and obstacle locations. Although we present the shot selection module in this chapter’s architecture, its details and experiments are described in Chapter 5. For now, we assume that shot parameters are chosen by a human director.

Planning: The planning module takes in the predicted actor trajectory, TSDT map, and the desired artistic shot mode to compute a trajectory that balances safety, smoothness, shot quality and occlusion avoidance. Using the UAV pose estimate, the module outputs velocity commands for the UAV to track the computed trajectory. Details of the planning system were previously described in Chapter 3.

4.1.3 Hardware

We chose as our base platform the DJI M210 quadcopter, shown in Figure 4.2. The UAV fuses GPS, IMU and compass for state estimation, which can be accessed via DJI’s SDK. The M210 has a maximum payload capacity of 2.30 kg ¹, which limits our choice of batteries and onboard computers and sensors.

Our payload is composed of (weights are summarized in Table 4.1):

- DJI TB50 Batteries, with maximum flight time of 13 minutes at full payload;
- DJI Zenmuse X4S gimbaled camera, whose 3-axis gimbal can be controlled independently of the UAV’s motion with angular precision of $\pm 0.01^\circ$, and counts with a

¹ <https://www.dji.com/products/compare-m200-series>

vibration-dampening structure. The camera records high-resolution videos, up to 4K at 60 FPS;

- NVIDIA Jetson TX2 with Astro carrier board. 8 GB of RAM, 6 CPU cores and 256 GPU cores for onboard computation;
- Velodyne Puck VLP-16 Lite LiDAR, with $\pm 15^\circ$ vertical field of view and 100 m max range.

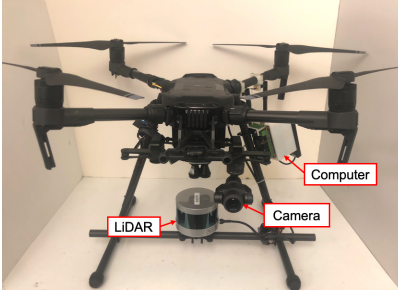


Figure 4.2: System hardware: DJI M210 drone equipped with Nvidia TX2 computer, Velodyne VLP-16 Puck Lite LiDAR and Zenmuse X4S camera gimbal.

Table 4.1: System and payload weights.

Component	Weight (kg)
DJI M210	2.80
DJI Zenmuse X4S	0.25
DJI TB 50 Batteries $\times 2$	1.04
NVIDIA TX2 w/ carrier board	0.30
VLP-16 Lite	0.59
Structure Modifications	0.63
Cables and connectors	0.28
Total:	$5.89 \leq 6.14$ (maximum takeoff weight)

4.1.4 Photo-realistic Simulation Platform

We use the Microsoft AirSim simulation platform [207] to test our framework and to collect training data for the shot selection module, as explained in detail later in Chapter 5. Airsim offers a high-fidelity visual and physical simulation for quadrotors and actors (such as humans and cars), as shown in Figure 4.3. We built a custom ROS [184] interface for the simulator, so that our system can switch between the simulation and the real drone seamlessly. All nodes from the system architecture are written in C++ and Python languages, and communicate using the ROS framework.

4.2 Online Environment Mapping

As explained in Chapter 3, the motion planner requires signed distance values \mathcal{M} to solve the optimization problem that results in the final UAV trajectory. The main role of the mapping subsystem described here is to register LiDAR points from the onboard sensor, update the occupancy grid \mathcal{G} , and incrementally update the signed distance \mathcal{M} .

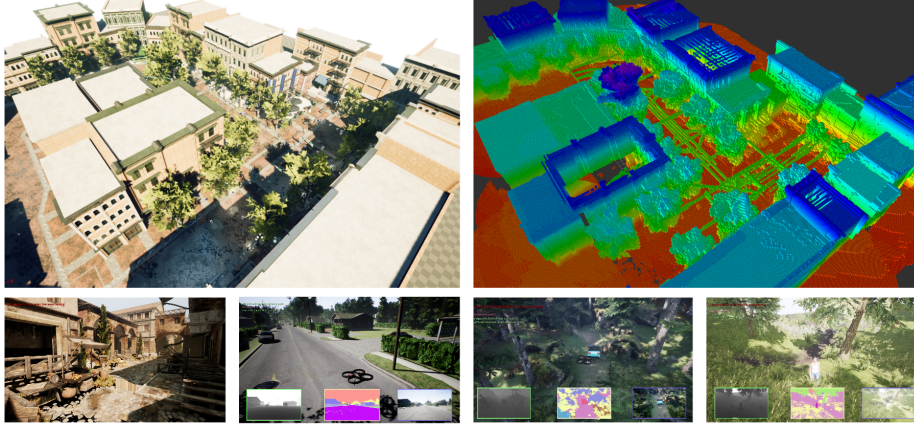


Figure 4.3: Simulation platform: Airsim combines a physics engine with photo-realistic renderings of various environments and actors (human and vehicles). Here we show the urban and forest environments we used for testing our framework. We build a point cloud and an occupancy map in the simulation. The simulation provides ground truth data for the actor’s pose, which is used to evaluate the performance of the vision pipeline.

4.2.1 LiDAR Registration

During our filming operation, we receive approximately 300,000 points per second from the laser sensor mounted at the bottom of the aircraft. We register the points in the world coordinate system using a rigid body transform between the sensor and the aircraft plus the UAV’s state estimation, which fuses GPS, barometer, internal IMUs and accelerometers. For each point we also store the corresponding sensor coordinate, which is used for the occupancy grid update.

LiDAR points can be categorized either as hits, which represent successful laser returns within the maximum range of 100 m, or as misses, which represent returns that are either non-existent, beyond the maximum range, or below a minimum sensor range. We filter all expected misses caused by reflections from the aircraft’s own structure. Finally, we probabilistically update all voxels from \mathcal{G} between the sensor and its LiDAR returns, as described in Subsection 4.2.2.

4.2.2 Occupancy Grid Update

The mapping subsystem holds a rectangular grid that stores the likelihood that any cell in space is occupied. In this work we use a grid size of $250 \times 250 \times 100$ m, with 1 m square voxels that store an 8-bit integer value between 0 – 255 as the occupancy probability, where 0 is the limit for a fully free cell, and 255 is the limit for a fully occupied cell. All cells are initialized as *unknown*, with value of 127.

Algorithm 2 covers the grid update process. The inputs to the algorithm are the sensor position p_{sensor} , the LiDAR point p_{point} , and a flag `is_hit` that indicates whether the point is a hit or miss. The endpoint voxel of a hit will be updated with log-odds value l_{occ} , and all cells in between sensor and endpoint will be updated by subtracting value l_{free} . We assume that all misses are returned as points at the maximum sensor range, and in this case only the cells between endpoint and sensor are updated l_{free} .

As seen in Algorithm 2, all voxel state changes to *occupied* or *free* are stored in lists $V_{\text{occ}}^{\text{change}}$ and $V_{\text{free}}^{\text{change}}$. State changes are used for the signed distance update, as explained in Subsection 4.2.3.

Algorithm 2: Update \mathcal{G} ($p_{\text{sensor}}, p_{\text{point}}, \text{is_hit}$)

```

1 Initialize  $V_{\text{occ}}^{\text{change}}, V_{\text{free}}^{\text{change}} \triangleright$  list of changed voxels
2 Initialize  $l_{\text{free}}, l_{\text{occ}} \triangleright$  log-odds probabilistic updates
3 for each voxel  $v$  between  $p_{\text{sensor}}$  and  $p_{\text{point}}$  do
4    $v \leftarrow v - l_{\text{free}}$ ;
5   if  $v$  was occupied or unknown and now is free then
6     Append( $v, V_{\text{free}}^{\text{change}}$ );
7     for each unknown neighbor  $v_{\text{unk}}$  of  $v$  do
8       Append( $v_{\text{unk}}, V_{\text{occ}}^{\text{change}}$ )
9     end
10  end
11  if  $v$  is the endpoint and  $\text{is\_hit}$  is true then
12     $v \leftarrow v + l_{\text{occ}}$ ;
13    if  $v$  was free or unknown and now is occupied then
14      Append( $v, V_{\text{occ}}^{\text{change}}$ )
15    end
16  end
17 end
18 return  $V_{\text{occ}}^{\text{change}}, V_{\text{free}}^{\text{change}}$ 

```

4.2.3 Incremental Distance Transform Update

We use the list of voxel state changes as input to an algorithm, modified from [51], that calculates an incremental truncated signed distance transform (iTSDT), stored in \mathcal{M} . The original algorithm described by [51] initializes all voxels in \mathcal{M} as free, and as voxel changes arrive in sets $V_{\text{occ}}^{\text{change}}$ and $V_{\text{free}}^{\text{change}}$, it incrementally updates the distance of each free voxel to the closest occupied voxel using an efficient wavefront expansion technique within some limit (therefore truncated).

Our problem, however, requires a *signed* version of the DT, where the *inside* and *outside* of obstacles must be identified and given opposite signs (details of this requirement are given in the description of the occlusion cost function detailed in Chapter 3). The concept of regions inside and outside of obstacles cannot be captured by the original algorithm, which provides only a iTDT (with no sign). Therefore, we introduced two important modifications:

Using the borders of obstacles. The original algorithm uses only the occupied cells of \mathcal{G} , which are incrementally pushed into \mathcal{M} using set $V_{\text{occ}}^{\text{change}}$. We, instead, define the concept of *obstacle border* cells, and push them incrementally as $V_{\text{occ}}^{\text{change}}$.

Let v_{border} be an obstacle border voxel, and V_{border} be the set of all border voxels in the environment. We define v_{border} as any voxel that is either a direct hit from the LiDAR (lines 13 – 15 of Alg. 2), or as any *unknown* voxel that is a neighbor of a *free* voxel (lines 5 – 9 of Alg. 2). In other words, the set V_{border} will represent all cells that separate the known free

space from unknown space in the map, whether this unknown space is part of cells inside an obstacle or cells that are actually free but just have not yet been cleared by the LiDAR.

By incrementally pushing V_{occ}^{change} and V_{free}^{change} into \mathcal{M} , its data structure will maintain the current set of border cells V_{border} . By using the same algorithm described in [51] but now with this distinct type of data input, we can obtain the distance of any voxel in \mathcal{M} to the closest obstacle border. One more step is required to obtain the sign of this distance.

Querying \mathcal{G} for the sign. The data structure of \mathcal{M} only stores the distance of each cell to the nearest obstacle border. Therefore we query the value of \mathcal{G} to attribute the sign of the iTSDT, marking free voxels as positive, and unknown or occupied voxels as negative (Figure 4.4).

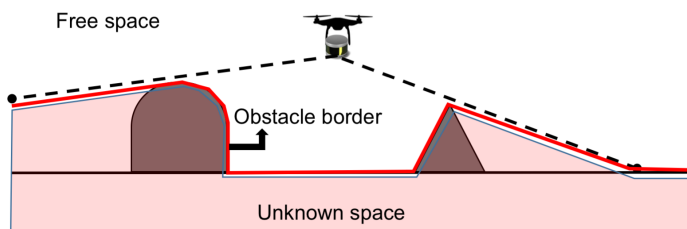


Figure 4.4: Diagram with our obstacle representation. Unknown voxels (red shade) are either inside of obstacles, or in zones occluded from the sensor’s field of view. The red line displays the obstacle border, which is at the interface between LiDAR hits and free space, and at the interface between unknown and free space. In this conceptual figure we assume the sensor to be omnidirectional, so the ground and obstacles below aircraft were captured as hits; however in practice the sensor has field of view limitations.

4.2.4 Building a Height Map

Despite keeping a full 3D map structure as the representation used for planning (Chapter 3), we also incrementally build a height map of the environment that is used for both the raycasting procedure when finding the actor position in world coordinates (Section 4.3), and for the online artistic choice selection procedure (Chapter 5).

The height map is a 2D array where the value of each cell corresponds to a moving average of the height of the LiDAR hits that arrive in each position. All cells are initialized with $0m$ of height, relative to the world coordinate frame, which is taken from the UAV’s takeoff position. An example height map is shown in Figure 4.5.

4.3 Visual Actor Localization and Heading Estimation

The vision module is responsible for two critical roles in the system: to estimate the actor’s future trajectory and to control the camera gimbal to keep the actor within the frame. Figure 4.6 details the four main sub-modules: actor detection and tracking, heading direction angle estimation, global position ray-casting, and finally a filtering stage for trajectory forecasting. Next, we detail each sub-module.

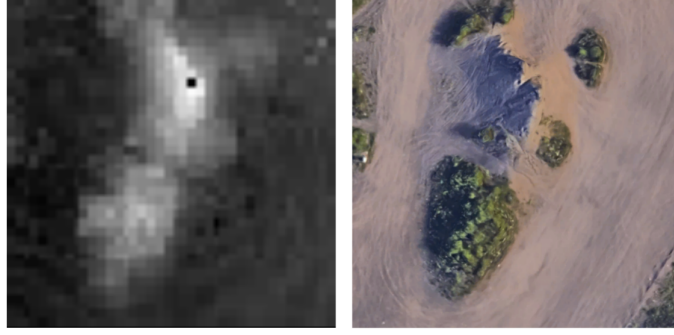


Figure 4.5: The left figure shows the height map accumulated over flight over a small mountain. Color scale goes from 0 (−10m) to 255 (+10m), where the zero reference of 127 is taken at the UAV’s initial takeoff height. The right figure is the top-down view of the mountain of the same place.

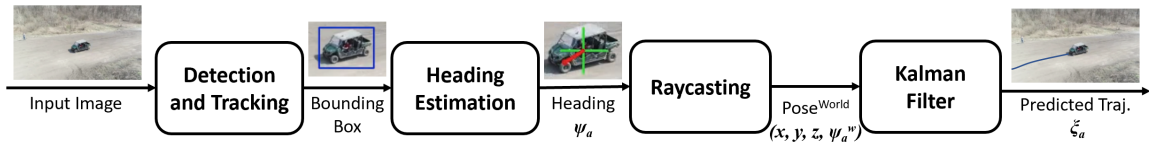


Figure 4.6: Vision sub-system. We detect and track the actor’s bounding box, estimate its heading, and project its pose to world coordinates. A Kalman Filter predicts the actor’s forecasted trajectory ξ_a .

4.3.1 Detection and Tracking

As we discussed in Section 8.2, the state-of-the-art object detection methods require large computational resources, which are not available on our onboard platform, and do not perform well in our scenario due to the data distribution mismatch. Therefore, we develop two solutions: first, we build a custom network structure and train it on both the open and context-specific datasets in order to improve speed and accuracy; second, we combine the object detector with a real-time tracker for stable performance.

The deep learning based object detectors are composed of a feature extractor followed by a classifier or regressor. Different feature extractors could be used in each detector to balance efficiency and accuracy. Since the onboard embedded GPU is less powerful, we can only afford feature extractor with relatively fewer layers. We compare several lightweight publicly available trained models for people detection and car detection.

Due to good real-time inference speed and low memory usage, we combine the MobileNet [96] for feature extraction and the Faster-RCNN [190] architecture. Our feature extractor consists of 11 depth-wise convolutional modules, which contains 22 convolutional layers. Following the Faster-RCNN structure, the extracted feature then goes to a two-stage detector, namely a region proposal stage and a bounding box regression and classification stage. While the size of the original Faster-RCNN architecture with VGG is 548 MB, our custom network’s is 39 MB, with average inference time of 300 ms.

The distribution of images in the aerial filming task differs significantly from the usual images found in open accessible datasets, due to highly variable relative yaw and tilt angles

to the actors, large distances, varying lighting conditions and heavy motion blur. Figure 4.7 displays examples of challenging situations faced in the aerial cinematography problem. Therefore, we trained our network with images from two sources: a custom dataset of 120 challenging images collected from field experiments, and images from the COCO [144] dataset, in a 1:10 ratio. We limited the detection categories only to person, car, bicycle, and motorcycle, which are object types that commonly appear as actors in aerial filming.

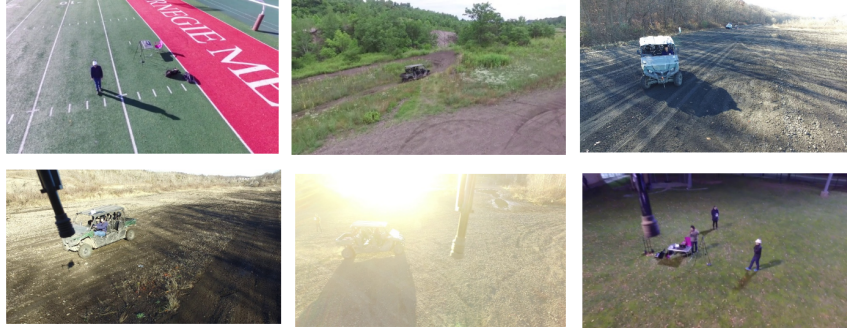


Figure 4.7: Examples of challenging images for actor detection in the aerial filming task. Large relative tilt angles to the ground, variable lighting, large distance to actor and heavy motion blur make bounding box detection harder than in images from open datasets.

The detection module receives the main camera’s monocular image as inputs, and outputs a bounding box. We use this initial bounding box to initialize a template tracking process, and re-initialize detection whenever the tracker’s confidence falls below acceptable limits. We adopt this approach, as opposed to detecting the actor in every image frame, because detection is a computationally heavy process, and the high rate of template tracking provides more stable measurements for subsequent calculations. We use Kernelized Correlation Filters [91] to track the template over the next incoming frames.

As mentioned in Section 4.1, we actively control camera gimbal independently of the UAV’s motion to maintain visibility of the target. We use a PD controller to frame the actor on the desired screen position, following the commanded artistic principles from the operator. Typically, the operator centers the target on the middle of the image space, or uses visual composition rules such as the rule of thirds [26], as seen on Figure 4.8.



Figure 4.8: Desired screen position of the actor projection, defined by parameters: $sp_x, sp_y \in [0, 1]$. Typically the user uses one of the thirds of the screen to set the actor’s position, or centers the actor on the frame [26].

4.3.2 Heading Estimation

When filming a moving actor, heading direction estimation (HDE) plays a central role in motion planning. Using the actor’s heading information, the UAV can position itself within the desired shot type determined by the user’s artistic objectives, *e.g.*: back, front, left and right side shots, or within any other desired relative yaw angle.

Estimating the heading of people and objects is also an active research problem in many other applications, such as pedestrian collision risk analysis [227], human-robot interaction [233] and activity forecasting [125]. Similar to challenges in bounding box detection, models obtained in other datasets do not easily generalize to the aerial filming task, due to a mismatch in the types of images from datasets to our application. In addition, when the trained model is deployed on the UAV, errors are compounded because the HDE relies on an imperfect object detection module, increasing the mismatch [80, 193].

No current dataset satisfies our needs for aerial HDE, creating the need for us to create a custom labeled dataset for our application. As most deep learning approaches, training a network is a data-intensive process, and manually labeling a large enough dataset for conventional supervised learning is a laborious and expensive task. The process is further complicated as multiple actor types such as people, cars and bicycles can appear in footages.

These constraints motivated us to formulate a novel semi-supervised algorithm for the HDE problem [236]. To drastically reduce the quantity of labeled data, we leverage temporal continuity in video sequences as an unsupervised signal to regularize the model and achieve better generalization. We apply the semi-supervised algorithm in both training and testing phases, drastically increasing inference performance, and show that by leveraging unlabeled sequences, the amount of labeled data required can be significantly reduced.

Defining the loss for temporal continuity

We define the pose of the actor as a vector $[x, y, z, \psi_a^w]$ on the ground surface. In order to estimate the actor’s heading direction in the world frame ψ_a^w , we first predict the actor’s heading ψ_a in the image frame, as shown in Figure 4.9. Once ψ_a is estimated, we project this direction onto the world frame coordinates using the camera’s intrinsic and extrinsic matrices.



Figure 4.9: Example of actor bounding boxes and their respective heading angles ψ_a in image space. Given the images, our objective is to predict the heading direction, shown as red arrows.

The HDE module outputs the estimated heading angle ψ_a in image space. Since ψ_a is ambiguously defined at the frontier between $-\pi$ and π , we define the inference as a regression problem that outputs two continuous values: $[\cos(\psi_a), \sin(\psi_a)]$. This avoids model instabilities during training and inference.

We assume access to a relatively small labeled dataset $D = \{(x_i, y_i)\}_{i=0}^n$, where x_i denotes input image, and $y_i = [\cos(\psi_i), \sin(\psi_i)]$ denotes the angle label. In addition, we assume access to a large unlabeled sequential dataset $U = \{q_j\}_{j=0}^m$, where $q_j = \{x_0, x_1, \dots, x_t\}$ is a sequence of temporally-continuous image data.

The HDE module’s main objective is to approximate a function $y = f(x)$, that minimizes the regression loss on the labeled data $\sum_{(x,y) \in D} L_l(x_l, y_l) = \sum_{(x,y) \in D} \|y_i - f(x_i)\|^2$. One intuitive way to leverage unlabeled data is to add a constraint that the output of the model should not have large discrepancies over a consecutive input sequence. Therefore, we train the model to jointly minimize the labeled loss L_l and some continuity loss L_u . We minimize the combined loss:

$$L_{tot} = \min \sum_{(x_l, y_l) \in L} L_l(x_l, y_l) + \lambda \sum_{q_u \in U} L_u(q_u) \quad (4.1)$$

We define the unsupervised loss using the idea that samples closer in time should have smaller differences in angles than samples further away in time. A similar continuity loss is also used by [237] when training an unsupervised feature extractor:

$$L_u(q_u) = \sum_{x_1, x_2, x_3} \max[0, D(x_1, x_2; f) - D(x_1, x_3; f)], \quad (4.2)$$

where: $D(x_1, x_2; f) = \|f(x_1) - f(x_2)\|_2$,
and: $x_1, x_2, x_3 \in q_u$

Network structure

For lower memory usage and faster inference time in the onboard computer, we design a compact CNN architecture based on MobileNet [96]. The input to the network is a cropped image of the target’s bounding box, outputted by the detection and tracking modules. The cropped image is padded to a square shape and resized to 192 x 192 pixels. After the 10 group-wise and point-wise convolutional blocks from the original MobileNet architecture, we add another convolutional layer and a fully connected layer that output two values representing the cosine and sine values of the angle. Figure 4.10 illustrates the architecture.

During each training iteration, one shuffled batch of labeled data and one sequence of unlabeled data are passed through the network. The labeled loss and unlabeled losses are computed and backpropagated through the network.

Cross-dataset semi-supervised fine-tuning

Due to data distribution mismatch between the aerial cinematography task and open datasets, we train our network on a combination of images from both sources. Later in Subsection 4.4.2 we evaluate the impact of fine-tuning the training process with unsupervised videos from our application.

4.3.3 Ray-casting

The ray-casting module converts the detection/tracking and HDE results from image space to coordinates and heading in the world frame $[x, y, z, \psi_a^w]$. Given the actor’s bounding

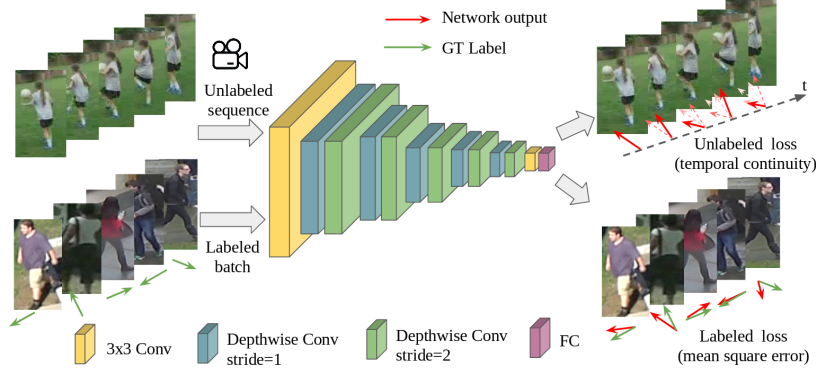


Figure 4.10: Our network architecture for predicting the actor’s heading direction. We use a Mobilenet-based feature extractor followed by a convolutional layer and a fully connected layer to regress to angular values. The network is trained using both labeled and unsupervised losses.

box, we project its center-bottom point onto a height map of the terrain, provided by the mapping module. The intersection of this line with the height map provides the $[x, y, z]$ location of the actor.

Assuming that the camera gimbal’s roll angle is fixed at zero degrees by the active gimbal controller, we can directly obtain the actor’s heading direction on the world frame ψ_a^w by transforming the heading ψ from the image space with the camera’s extrinsic matrix in world coordinates.

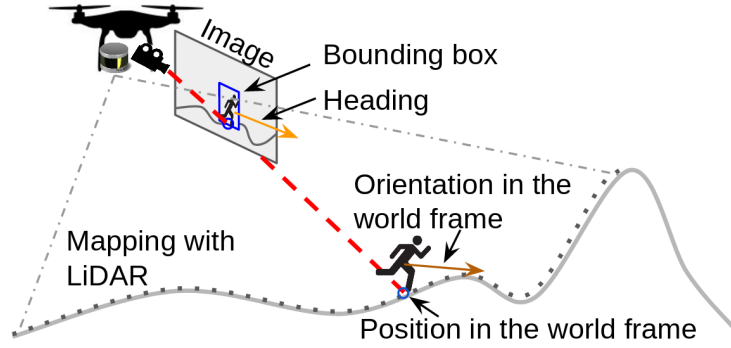


Figure 4.11: Raycasting module uses the actor’s bounding box, estimated heading angle, environment height map and camera matrices to obtain pose of actor in the world frame $[x, y, z, \psi_a^w]$.

4.3.4 Motion Forecasting

Given a sequence of actor poses in the world coordinates, we estimate the actor’s future trajectory based on motion models. The motion planner later uses the forecast to plan non-myopically over long time horizons.

We use two different motion models depending on the actor types. For people, we apply a linear Kalman filter with a two-dimensional motion model. Since a person’s movement direction can change drastically, we use no kinematic constraints applied to the motion

model, and just assume constant velocity. We assume no control inputs for state $[x, y, \dot{x}, \dot{y}]$ in the prediction step, and use the next measurement of $[x, y, z]$ in the correction step. When forecasting the motion of cars and bicycles we apply an extended Kalman filter with a kinematic bicycle model. For both cases we use a 10 s horizon for prediction.

4.4 Experimental Results

In this section we detail integrated experimental results, followed by detailed results on each subsystem.

4.4.1 Integrated System Results

We conducted a series of field trials to evaluate our integrated system in real-life conditions. We used a large open facility named Gascola in Penn Hills, PA, located about 20 min east of Pittsburgh, PA. The facility has a diverse set of obstacle types and terrain types such as several off-road trails, large mounds of dirt, trees, and uneven terrain. Figure 4.12 depicts the test site and shows the different areas where the UAV flew during experiments. We summarize the test’s objectives and results in Table 4.2, and indicate which results explain our initial hypotheses from Subsection 4.1.1.

Figure 4.15 summarizes our experiments conducted with fixed shot types. We employ a variety of shot types and actors, while operating in a wide range of unstructured environments such as open fields, in proximity to a large mound of dirt, on narrow trails between trees and on slopes. In addition, Figure 4.13 provides a detailed time lapse of how the planner’s trajectory output evolves during flight through a narrow trail between trees.

We also summarize our integrated system’s runtime performance statistics in Table 4.3, and discuss details of the online mapping performance in Figure 4.14. Videos of the system in action can be found at <https://youtu.be/oookhHnqmlaU>.

From the field experiments, we verify that our system achieved all system-level objectives in terms of safely and robustly executing a diverse set of aerial shots with different actors and environments. Our data also confirms the questions raised to validate our hypotheses: onboard sensors and computing power sufficed to provide smooth plans, producing artistically appealing images. Next we present detailed results on the individual sub-systems of the aircraft.

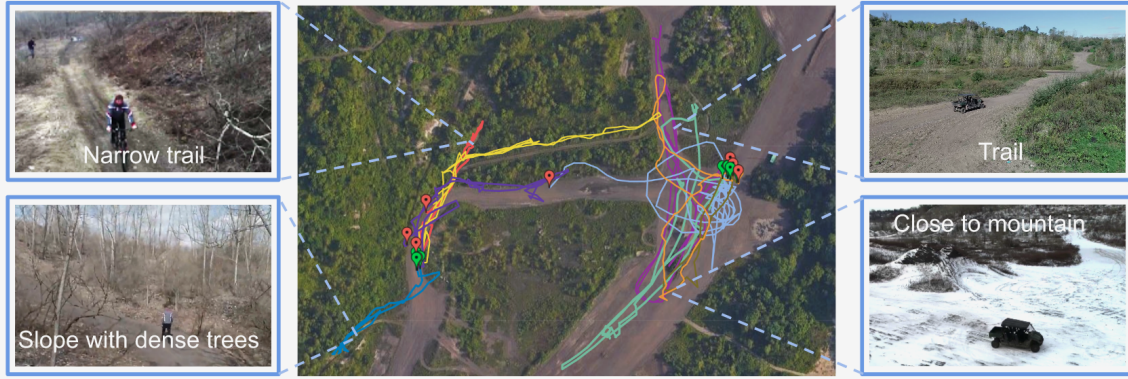


Figure 4.12: Testing facility. The middle figure shows a top-down satellite view of testing terrain, overlaid with UAV positions from different trials. We accumulated over 2h of flight time, and a total distance of almost 6km. The side figures show the diverse types of terrains in our experiments. The figures also depict different actors and different seasons.

Table 4.2: Objectives and results for integrated experiments.

Objectives	Results
Stay safe among unstructured obstacles sensed online (addresses Hyp.1)	Avoided all obstacles successfully, including trees, dirt mound, slopes, posts. See Figs. 4.15-4.13.
Avoid occlusions among any obstacle shape (addresses Hyp.1)	Planner maintained actor visibility. See Fig. 4.15. More results in Subsec 4.4.3.
Process data fully onboard (addresses Hyp.1)	Data processed solely on onboard computer. Table 4.3 and Fig. 4.14 show system statistics.
Operate with different types of actors at different speeds (addresses Hyps.2-3)	Person, car, bikes shot at high-speed chases. See Figs. 4.15-5.12.
Execute different shot types (addresses Hyp.3)	Successful recording of back, right, front, circling shots (Fig. 4.15). Smooth shot transitions (Fig. 5.12).

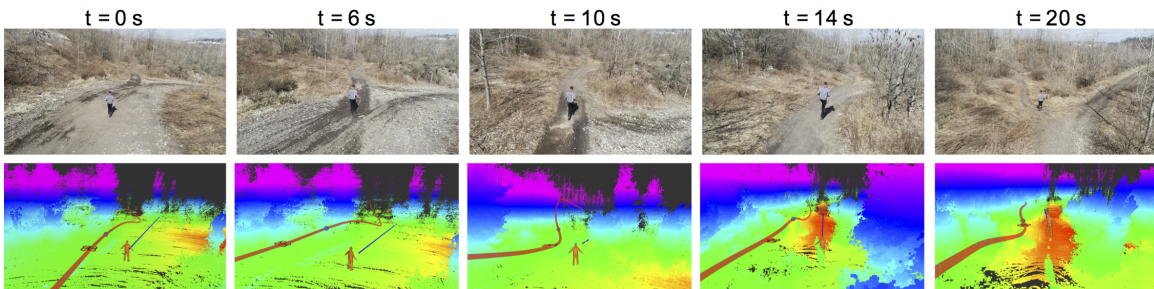


Figure 4.13: Detailed time lapse of back shot following a runner in a narrow trail with trees. As the UAV approaches the trees at $t = 6s$, the trajectory bends to keep the vehicle safe, maintain target visibility, and follow the terrain's downwards inclination.

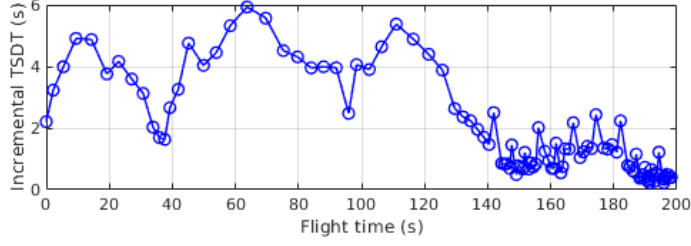


Figure 4.14: Incremental distance transform computation time over flight time. The first operations take significantly more time because of our map initialization scheme where all cells are initially considered as unknown instead of free, causing the first laser scans to update a significantly larger number of voxels than later scans. During calculation the planner is not blocked: it can still access TSDT values from the latest version of \mathcal{M} .

Table 4.3: System statistics recorded during flight time on onboard computer.

System	Module	CPU Thread (%)	RAM (MB)	Runtime (ms)	Target freq. (Hz)
Vision	Detection	57	2160	145	15
	Tracking	24	25	14.4	
	Heading	24	1768	13.9	
	KF	8	80	0.207	
Mapping	Grid	22	48	36.8	10
	TSDF	91	810	100-6000	
	LiDAR	24	9	100	
Planning	Planner	98	789	198	5
Controls	DJI SDK	89	40	NA	50

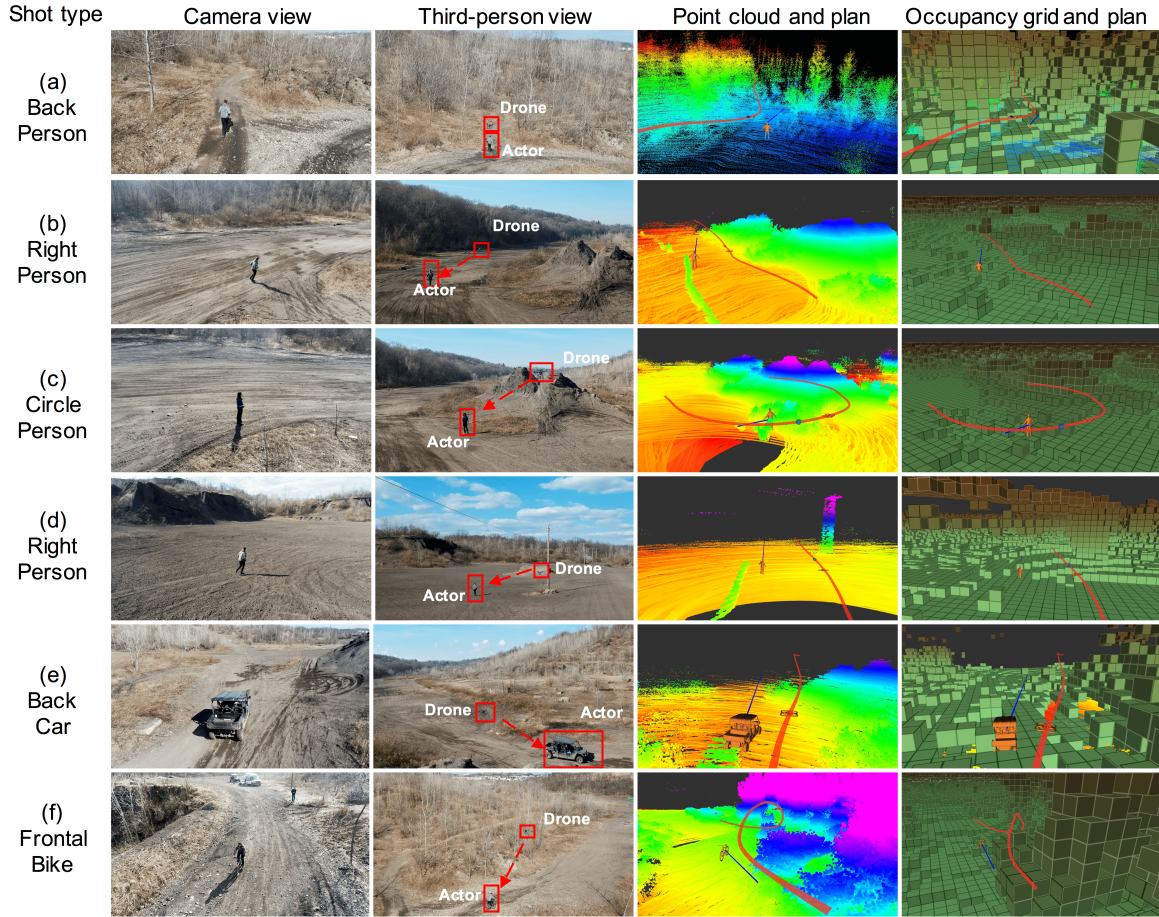


Figure 4.15: Field results with different fixed shot types in multiple environment types, following different actor types. The UAV trajectory (red) tracks the actor's forecasted motion (blue), and stays safe while avoiding occlusions from obstacles. We display accumulated point clouds of LiDAR hits and the occupancy grid: a) Back shot following runner in narrow tree trail; b) Right side shot following a runner close to dirt mound; c) Circular shot on person close to dirt mound; d) Right side shot below the 3D structure of an electric wire. Note that LiDAR registration is noisy close to the pole in row due to large electromagnetic interference with the UAV's compass; e) Right side shot following car close to dirt mound; f) Frontal shot on biker going downhill on a trail with tall trees.

4.4.2 Visual Actor Localization and Heading Estimation

Here we detail dataset collection, training and testing of the different subcomponents of the vision module. We summarize the vision-specific test’s objectives and results in Table 4.4.

Table 4.4: Objectives and results for vision-specific experiments.

Objectives	Results
Compare object detection network architectures	Faster-RCNN showed significantly better performance than SSD architecture (Fig. 4.16).
Compare supervised and semi-supervised methods for heading estimation	Semi-supervised method has smoother output and higher accuracy (Table 4.6 and Fig. 4.17)
Analyze the amount of labeled data needed for semi-supervised training	Loss increased by less than $\sim 8\%$ when we trained the model with 1/10 of labeled data (Fig. 4.18)
Validate integrated 3D pose estimator using image projections	Error of less than 1.7m in estimated actor path length over a 40m long ground-truth actor trajectory (Fig. 4.19).

Object detection network

Dataset collection. We trained the network on the COCO dataset [144], and fine-tuned it with a custom aerial filming dataset. To test, we manually labeled 120 images collected from our aerial filming experiments, with bounding box over people and cars.

Training procedure: We trained and compared two architectures: one based on Faster-RCNN, another on SSD. As mentioned in Section 4.3, we simplify feature extraction with MobileNet-based structure to improve efficiency. First we train both structures on the COCO dataset. While the testing performance is good on the COCO testing dataset, the performance shows a significant drop, when tested on our aerial filming data. The network has a low recall rate (lower than 33%) due to big angle of view, distant target, and motion blur. To address the generalization problem, we augmented the training data by adding blur, resizing and cropping images, and modifying colors. After training on a mixture of COCO dataset [144] and our own custom dataset as described in Section 4.3, Figure 4.16 show the recall-precision curve of the two networks when tested on our filming testing data. The SSD-based network has difficulties detecting small objects, an important need for aerial filming. Therefore, we use Faster-RCNN-based network in our experiments and set the detection threshold to precision=0.9, as shown with the green arrow in Figure 4.16.

Heading direction estimation (HDE) network

Dataset collection: We collected a large number of image sequences from various sources. For the person HDE, we used two surveillance datasets: VIRAT [177] and DukeMCMC [193], and one action classification dataset: UCF101 [212]. We manually labeled 453 images in the UCF101 dataset as ground-truth HDE. As for the surveillance datasets, we adopted a semi-automatic labeling approach where we first detected the actor in every frame, then computed the ground-truth heading direction based on the derivative of the subject’s position over a sequence of consecutive time frames. For the car HDE we used two surveillance datasets, VIRAT and Ko-PER [219], in addition to one driving dataset, PKU-POSS [235]. Table 4.5 summarizes our data.

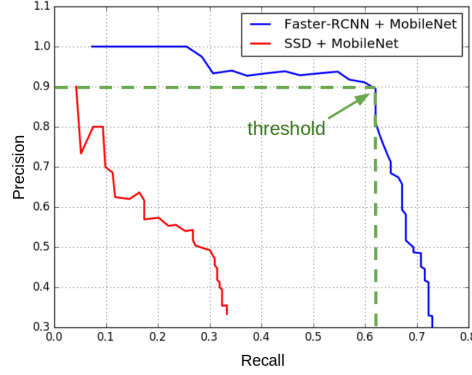


Figure 4.16: Precision recall curve for object detection. The results are tested on the filming testing data, which contains many challenging cases.

Table 4.5: Datasets used in HDE study. *MT denotes labeling by motion tracking, HL denotes hand labels.

Dataset	Target	GT	No. Seqs	No. Imgs
VIRAT	Car/Person	MT*	650	69680
UCF101	Person	HL(453)*	940	118027
DukeMCMT	Person	MT*	4336	274313
Ko-PER	Car	✓	12	18277
PKU-POSS	Car	✓	-	28973

Training the network:

We first train the HDE network using only labeled data from the datasets shown in Table 4.5. Rows 1-3 of Table 4.6 display the results. Then, we fine-tune the model with unlabeled data to improve generalization.

We collected 50 videos, each contains approximately 500 sequential images. For each video, we manually labeled 6 images. The HDE model is finetuned with both labeled loss and continuity loss, same as the training process on the open accessible datasets. We qualitatively and quantitatively show the results of HDE using semi-supervised finetuning in Figure 4.17 and Table 4.6. The experiment verifies our model could generalize well to our drone filming task, with a average angle error of 0.359 rad. Compared to the pure supervised learning, utilizing unlabeled data improves generalization and results in more robust and stable performance.

Baseline comparisons: We compare our HDE approach against two baselines. The first baseline Vanilla-CNN is a simple CNN inspired by [43]. The second baseline CNN-GRU implicitly learns temporal continuity using a GRU network inspired by [150]. One drawback for this model is that although it models the temporal continuity implicitly, it needs large number of labeled sequential data for training, which is very expensive to obtain.

We employ three metrics for quantitative evaluation: 1) Mean square error (MSE) between the output $(\cos \theta, \sin \theta)$ and the ground truth $(\cos \hat{\theta}, \sin \hat{\theta})$. 2) Angular difference (AngleDiff) between the output and the ground truth. 3) Accuracy obtained by counting the percentage

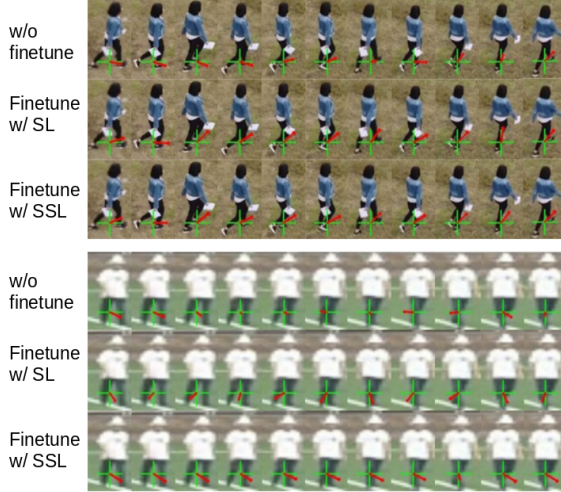


Figure 4.17: Three models are tested on the sequential data. Two testing sequences are shown in this figure. The top row of each testing sequence shows the results that directly employ the model trained on other open accessible datasets to the aerial filming task. It generalizes poorly due to the distribution difference. The middle row and bottom row show the results after finetuning the model on the filming data with and without continuity loss, respectively. The model using continuity loss for finetuning (bottom row) outputs more accurate and smooth results.

of correct outputs, which satisfies $\text{AngleDiff} < \pi/8$. We use the third metric, which allows small error, to alleviate the ambiguity in labeling human heading direction.

Vanilla-CNN [43] and CNN-GRU [150] baselines trained on open datasets don’t transfer well to drone filming dataset with accuracy below 30%. Our SSL based model trained on open datasets achieves 48.7% accuracy. By finetuning on labelled samples of drone filming, we improve this to 68.1%. Best performance is achieved by finetuning on labelled and unlabeled sequences of the drone filming data with accuracy of 72.2% (Table 4.6).

Table 4.6: Semi-Supervised Finetuning results

Method	MSE loss	AngleDiff (rad)	Accuracy (%)
Vanilla-CNN w/o finetune	0.53	1.12	26.67
CNN-GRU w/o finetune	0.5	1.05	29.33
SSL w/o finetune	0.245	0.649	48.7
SL w/ finetune	0.146	0.370	68.1
SSL w/ finetune	0.113	0.359	72.2

Reduction in required labeled data using semi-supervised learning: Following Section 4.3, we show how semi-supervised learning can significantly decrease the number of labeled data required for the HDE task.

In this experiment, we train the HDE network on the DukeMCMT dataset, which consists of 274k labeled images from 8 different surveillance cameras. We use the data from 7 cameras for training, and one for testing (about 50k). Figure 4.18 compares result from the proposed semi-supervised method with a supervised method using different number of labeled data.

We verify that by utilizing unsupervised loss, the model generalizes better to the validation data than the one with purely supervised loss.

As mentioned, in practice, we only use 50 unlabeled image sequences, each containing approximately 500 sequential images, and manually labeled 300 of those images. We achieve comparable performance with purely supervised learning methods, which require more labeled data.

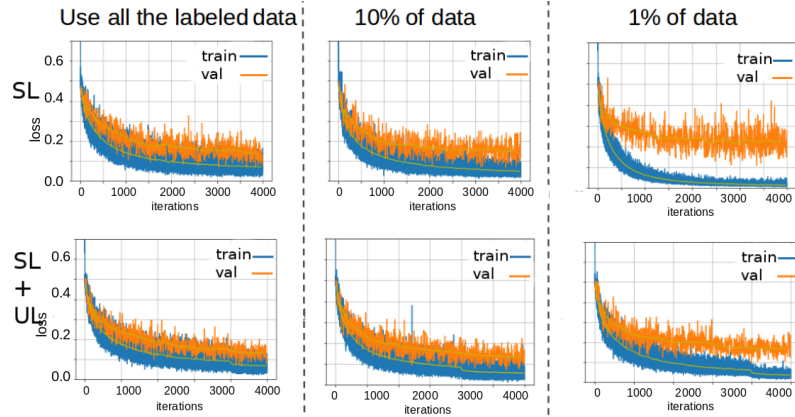


Figure 4.18: The top row shows training and validation loss for supervised learning using different number of labeled data. The validation performance drops from 0.13 to 0.22, when decreasing the number of labeled data from 100% to 1%. The bottom row shows results with semi-supervised learning. The validation losses are 0.13, 0.14 and 0.17 respectively for 100%, 10% and 1% labeled data.

3D pose estimation

Based on the detected bounding box and the actor’s heading direction in 2D image space, we use ray-casting method to calculate the 3D pose of the actor, given the online occupancy map and the camera pose. We assume the actor is in a upward pose, in which case the pose is simplified as (x, y, z, ψ_a^w) , which represents the position and orientation in the world frame.

We validate the precision of our 3D pose estimation in two field experiments where the drone hovers and visually tracks the actor. First, the actor walks between two points along a straight line, and we compare the estimated and ground truth path lengths. Second, the actor walks in a circle at the center of a football field, and we compute the errors in estimated position and heading direction. Figure 4.19 shows our estimation error is less than 5.7%.

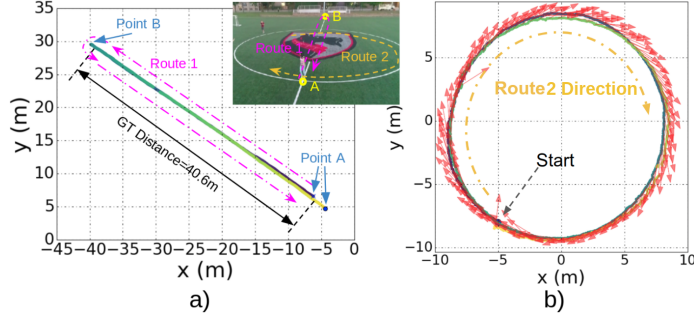


Figure 4.19: Pose and heading estimation results. a) Actor walks on a straight line from points A-B-A. Ground-truth trajectory length is 40.6m, while the estimated motion length is 42.3m. b) The actor walks along a circle. Ground-truth diameter is 18.3m, while the estimated diameter from ray casting is 18.7m. Heading estimation appears tangential to the ground circle.

4.4.3 Planner Evaluation

Next we present detailed results on different aspects of the UAV’s motion planner. Table 4.7 summarizes the experiments’ objectives and results.

Table 4.7: Objectives and results for detailed motion planner experiments

Objectives	Results
Performance comparison between online vs. ground-truth map	Similar path quality, with increase in planning time. See Fig. 4.20 and Table 4.8.
Performance comparison between noisy actor forecast vs. ground-truth actor positioning	Similar path quality: smoothness cost handles noisy inputs. See Fig. 4.21.
Confirm ability to operate in full 3D environments	Can fly under 3D obstacles, not only height maps. See Fig. 4.15-d.

Ground-truth obstacle map vs. online map: We compare average planning costs between results from a real-life test where the planner operated while mapping the environment in real time with planning results with the same actor trajectory but with full knowledge of the map beforehand. Results are averaged over 140 s of flight and approximately 700 planning problems. Table 4.8 shows a small increase in average planning costs with online map, and Fig 4.20 shows that qualitatively both trajectories differ minimally. The planning time, however, doubles in the online mapping case due to mainly two factors: extra load on CPU from other system modules, and delays introduced by accessing the map that is constantly being updated. Nevertheless, computation time is low enough such that the planning module can still operate at the target frequency of 5 Hz.

Table 4.8: Performance comparison between ground-truth and online map

Planning condition	Avg. plan time(ms)	Avg. cost	Median cost
Ground-truth map	32.1	0.1022	0.0603
Online map	69.0	0.1102	0.0825

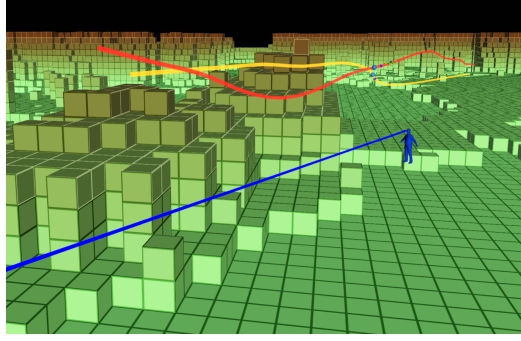


Figure 4.20: Performance comparisons between planning with full knowledge of the map (yellow) versus with online mapping (red), displayed over ground truth map grid. Online map trajectory is less smooth due to imperfect LiDAR registration and new obstacle discoveries as flight progresses.

Ground-truth actor pose versus noisy estimate: We compare the performance between simulated flights where the planner has full knowledge of the actor’s position versus artificially noisy estimates with 1m amplitude of random noise. The qualitative comparison with the actor’s ground-truth trajectory shows close proximity of both final trajectories, as seen in Fig 4.21.

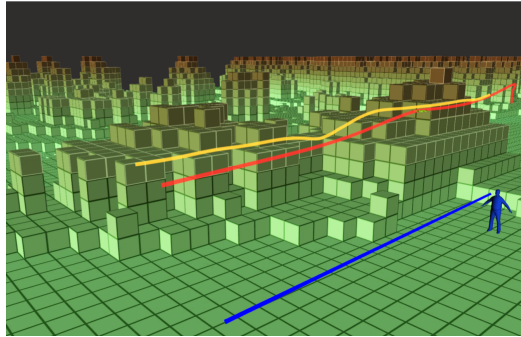


Figure 4.21: Performance comparison between planning with perfect ground truth of actor’s location (red) versus noisy actor estimate with artificial noise of 1m amplitude (yellow). The planner is able to handle noisy actor localization well due to smoothness cost terms, with final trajectory similar to ground-truth case.

Operation on unstructured 3D map: As seen in Figure 4.15d, our current system is able to map and avoid unstructured obstacles in 3D environments such as wires and poles. This capability is a significant improvement over previous work that only deals with ellipsoidal obstacle representations [98, 110, 171], or a height map assumption [22].

These detailed results allow us to draw insights into the planner performance under different conditions: it can operate smoothly, in full 3D maps, even under noise of real-time environment mapping and noisy actor inputs.

4.5 Discussion

In this section we discuss lessons learned throughout the development of our work, and also present comments on how our methods can be used with different types of sensors and UAV platforms.

4.5.1 Lessons Learned

During development of our autonomous aerial cinematographer platform we learned several lessons and gained insights into problem specificities, which we summarize below. We expect these lessons to not only be useful to researchers in the field of aerial vehicles, but also to generalize to other related areas.

Cascading errors can destabilize the robot: Estimation errors in a module get amplified if used downstream in decision making. For example, we learned that jerky UAV movements lead to mis-registration of camera pose, which leads to poor actor detection. This in turn leads to poor actor prediction, which can be off by meters. Unlike previous works that operate under highly precise motion capture systems, in real scenarios we observe that controlling the camera orientation towards the position estimates causes the robot to completely lose the actor. To mitigate this effect, we chose to decouple motion planning, which uses the actor world projection estimates, from camera control, which uses only object detections on the current image as feedback. To validate the quality of both threads independently, we performed statistical performance evaluations, as seen throughout Section 4.4.

Long-range sensors are beneficial to planning performance: The planner relies on the online map. Slow online map updates slow down the planner. This typically happens when the robot moves near large pockets of unknown areas, which triggers large updates for the TSDF. We learned that a relatively long range LiDAR sensor maps out a significantly larger area. Hence almost always, the area near the robot is mapped out fully and the planner does not have to wait for map updates to enter an unknown region. While our system used a relatively large map of $250 \times 250 \times 100\text{m}$, significantly faster mapping-planning frequencies could be achieved with the use of a smaller local map. Such change would likely be necessary with the use of shorter-range sensors like stereo pairs, which are common in commercial aircrafts due to their reduced price.

Semi-supervised methods can improve learning generalization: While deep learning methods are ubiquitous in computer vision, they rely on massive amounts of labeled data due to the complexity of the model class. Moreover, these models do not generalize to varying data distributions. We learned that one can reduce sample complexity by enforcing regularization / additional structure. In our case, we enforce temporal continuity on the network output. We show that a combination of *small labeled* dataset for supervisory loss and a *large unlabeled* dataset for temporal continuity loss is enough to solve the problem. Exploring other regularization techniques such as consistency between different sensory modalities is also an interesting area to be investigated in the future.

Height estimate using IMU and barometer is not enough for long operations: During extended vehicle operations (over 5 – 10 minutes), we learned that the UAV’s height calculated by fusing IMU and barometer data drifts significantly, especially after large vertical maneuvers. Inaccurate height estimates degrade pointcloud registration, thereby

degrading the overall system performance. In the future, we plan to use LiDAR or visual SLAM to provide more accurate 3D localization.

Real-world noise reduces transferability of the artistic policy trained in simulation: The noise present in real-world testing conditions, in particular for the map registration and actor localization, affected the results generated by the policy that was trained purely in simulation using ground-truth data. Shot selection in simulation highly prioritized viewpoints that drew the UAV away from tall obstacles, while in deployment we observed that the drone avoided proximity to tall objects with a significantly smaller frequency. In the future we will consider artificially adding noise to simulations for better transferability, or training the artistic policy with a combination of simulated and real-life data.

4.5.2 Adapting Our Work to Different UAVs and Sensors

In our experiments we employed a long-range LiDAR sensor for mapping the environment, and a DJI M210 UAV as the base platform. Even though we used relatively standard robotics development platform and sensor, researchers and developers who work on problems similar to aerial cinematography may face different constraints in terms of payload capacity, vehicle size, sensor modalities and costs. We argue that our problem formulation can be easily extended to other contexts.

First, we argue that the LiDAR sensor used in the online mapping module (Section 4.2) can be replaced by other sensors. Stereo cameras and depth sensors, for example, can be light-weight and significantly cheaper alternatives. The incoming hits for the mapping pipeline can then be acquired by using each pixel from a depth image, or each match from the stereo pair. The main advantage of LiDAR is the relatively long range, in the order of hundreds of meters. When using lighter sensors, the developer needs to take into account the new sensor range in order to keep the UAV safe. They must consider the expected vehicle speed and the scale of obstacles in the environment, so that the planner can reason about obstacles far from the UAV’s current position.

In addition, our system architecture is platform-agnostic. Our methods can easily be adapted to smaller or potentially cheaper platforms. To do so, one only needs to care for the software interface between the trajectory controller and the aircraft’s internal attitude or velocity controller.

In the future, we hope to see our architecture extended to other UAV types: our framework is not constrained to uniquely multi-rotors. With the appropriate changes in the motion planner’s trajectory parametrization and cost functions, our pipeline can also be employed by fixed-wing or hybrid aircrafts. More generally, despite the lower path dimensionality, even ground robots can employ the same methodology for visually tracking dynamic targets.

4.6 Conclusion

In this work we presented a system for robust autonomous aerial cinematography in unknown, unstructured environments while following dynamic actors in unscripted scenes. Current approaches do not address all theoretical and practical challenges present in real-life operation conditions; instead, they rely on simplifying assumptions such as requiring ground truth actor position, using prior maps of the environment, or only following one shot type specified

before flight. Our system revolves around the key idea of framing the filming task as an efficient cost optimization problem, which allows trajectories with long time horizons to be computed in real time, even under sensor noise.

We developed a system with four modules that work in parallel. (1) A vision-based actor localization module with motion prediction. (2) A real-time incremental mapping algorithm using a long-range LiDAR sensor. (3) A real-time optimization-based motion planner that exploits covariant gradients to efficiently calculate safe trajectories with long time horizons while balancing artistic objectives and occlusion avoidance for arbitrary obstacle shapes. (4) Finally, a deep reinforcement learning policy for artistic viewpoint selection, whose details are presented in Chapter 5.

We offered extensive detailed experiments to evaluate the robustness and real-time performance of our system both in simulation and real-life scenarios. These experiments occur among a variety of terrain types, obstacle shapes, obstacle complexities, actor trajectories, actor types (i.e., people, cars, bikes) and shot types.

Based on our results, we identify several key directions for possible future work. One clear direction is the extension of our theory to multi-drone, multi-actor scenarios. This improvement can be achieved by the addition of new cost functions that penalize inter-drone collisions, inter-drone sight, and a metric for multi-actor coverage. In addition, multi-actor scenarios require a slight modification in the definition of artistic parameters that define the desired artistic shot for our motion planner. We discuss some of these ideas in Chapters 7. Another interesting direction to follow lies among the reconstruction of dynamic scenes. While systems such as the CMU PanOptic Studio [108] can precisely reconstruct scenes volumetrically in indoor and static scenarios, to our knowledge, no current system offers good volumetric reconstruction of dynamic scenes in natural environments in real-life conditions.

Chapter 5

Learning Artistic Decision-Making

The system developed thus far takes on the role of the cinematographer, taking care of camera positioning with the assumption that all shot parameters had been previously specified either prior to take-off, or by remote access from a user during motion. When we compare such approach to a professional movie-making pipeline one key element is missing: the director. As exposed in Section 2.1, the director is the agent who specifies the shooting script, defining the technical parameters for the best viewpoints to tell a story.

In this section we introduce a novel method for the automatic selection of artistic shot parameters, emulating the role of a movie director. Using fixed shot type parameters Ω_{art} renders undesirable results during operation, since the UAV does not adapt to different configurations of obstacles in the environment. For instance, while following a runner in a narrow trail among trees the UAV should ideally stay ahead or behind the actor, and not attempt to do a risky side shot among obstacles. In addition, when filming in obstacle-free scenarios the UAV should avoid maintaining the same shot type for extended periods of time, as such strategy generally bores the viewer. Here we design an algorithm to train policies for adapting the selection shot types depending on the context of each scene.

Another contribution we offer in this chapter is the evaluation of our learned artistic policies. We conduct user studies to qualitatively and quantitatively evaluate the aesthetic preferences of users towards our proposed approach and baseline methods for selecting shot types. The supplementary videos show more details of footages captured by our system <https://youtu.be/ookhHnqmlaU>, and of the artistic shot selection module: <https://youtu.be/qmVw6mfyEmw>.

5.1 Deep Reinforcement Learning Problem Formulation

As introduced in Section 5.1, the choice of artistic parameters is a time-dependent sequential decision-making problem. Decisions taken at the current time step influence the quality of choices in future states. Figure 5.1 exemplifies the sequential nature of the problem.

We define the problem as a Contextual Markov Decision Process (C-MDP) [132], and use an RL algorithm to find an optimal shot selection policy. Our goal is to learn a policy $\pi_{\theta}(a_t|c_t)$, parametrized by θ , that allows an agent to choose action a_t , given scene context c_t , to select among a discrete set of UAV artistic parameters Ω_{art} . Our action set is defined as four

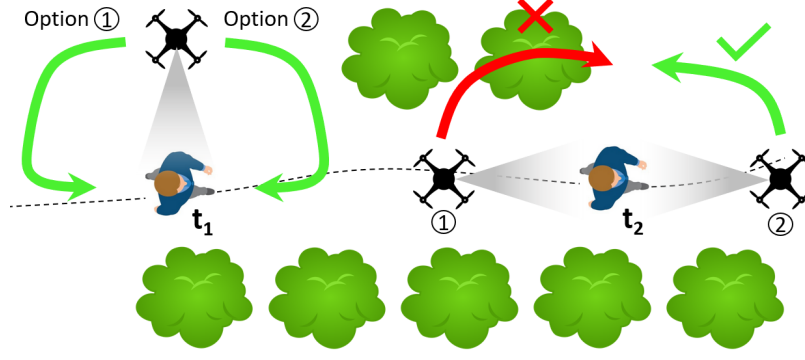


Figure 5.1: Example of artistic parameter selection as a sequential decision-making problem. The choice of frontal or back shots at time step t_1 influence the quality of left side shot choice at time step t_2 .

discrete values of Ω relative to left, right, back, and frontal shots. These shot types define the relative yaw angle ϕ_{rel} , which is fed into the UAV’s motion planner, as explained in Chapter 3.

We define state c_t as the scene context, which is an observation drawn from the current MDP state s_t . The true state of the MDP is not directly observable because, to maintain the Markovian assumption, it encodes a diverse set of information such as: the UAV’s full state and future trajectory, the actor’s true state and future trajectory, the full history of shot types executed in past choices, a set of images that the UAV’s camera has recorded so far, ground-truth obstacle map, environmental properties such as lighting and wind conditions, etc. Therefore, our definition of context c_t can be seen as a lower dimensional compression of s_t , given by a concatenation of following three elements:

1. *Height map*: a local 2.5D map containing the elevation of the obstacles near the actor;
2. *Current shot type*: four discrete values corresponding to the current relative position of the UAV with respect to the actor;
3. *Current shot count*: number of time steps the current shot type has been executed consecutively.

We assume that states evolve according to the system dynamics: $s_{t+1} \sim p(s_t, a_t)$. Finally, we define the artistic reward $R_{\text{art}}(v_t)$ where $v_t(s_t, a_t) = \{I_1, I_2, \dots, I_k\}$ is the video taken after the UAV executed action a_t at state s_t . Our objective is to find the parameters of the optimal policy, which maximizes the expected cumulative reward:

$$\theta^* = \arg \max_{\theta} \mathbb{E} \left[\sum_{t=1}^T R_{\text{art}}(v_t) \right], \quad (5.1)$$

where the expectation accounts for all randomness in the model and the policy. A major challenge for solving Equation 5.1 is the difficulty of explicitly modeling the state transition function $p(s_t, a_t)$. This function is dependent on variables such as the quadrotor and actor dynamics, the obstacle set, the motion planner’s implicit behavior, the quadrotor and camera gimbal controllers, and the disturbances in the environment. In practice, we cannot derive

an explicit model for the transition probabilities of the MDP. Therefore, we use a model-free method for the RL problem, using an action-value function $Q(c_t, a_t)$ to compute the artistic value of taking action a_t given the current context c_t :

$$Q(c_t, a_t) = \sum_{t'=t}^T \mathbb{E}_{\pi_\theta} [R_{\text{art}}(v(c_{t'}, a_{t'})) | c_t, a_t] \quad (5.2)$$

The large size and complexity of the state space for our application motivates us to use a deep neural network with parameters θ to approximate Q : $Q(c_t, a_t) \approx f(c_t, a_t, \theta)$ [168, 222].

One fundamental question we should also ask is: how often should we change shot types? In our problem, the choices of artistic parameters do not happen at the fastest frequency at which the network can operate. Rather, our choice is influenced by physical constraints (how fast can the drone switch positions) and movie-making techniques (what creates most enjoyment to the viewer). By analyzing references on the history of movie-making [53, 62, 163, 231], we find that the average shot length (ASL) in movies has fallen drastically over the years, going from about 12 seconds in 1930 to 2.5 seconds today [167] (Fig. 5.2). We also find a large variation in average shot duration between different directors: Micheal Bay (director of *Transformers*) has an ASL of 3.0 seconds, while others such as Steven Spielberg and Alfred Hitchcock have ASLs of 6.5 and 9.1 seconds respectively.

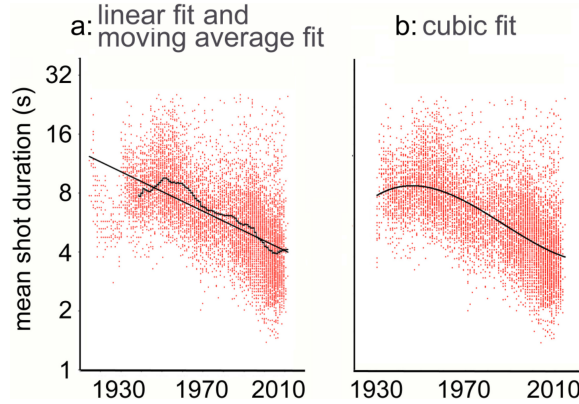


Figure 5.2: Significant shot length decrease over the years, from 1930 to 2010 [52]

After consideration of typical shot length values and taking into consideration the dynamics of our vehicle, we fixed the time step length between actions to be of 6 seconds. Under the lenses of a movie production, we can view these artistic actions as analogous to the role of the movie director. Between actions, the motion of the vehicle is guided by the low-level motion planner presented in Chapter 3, which is analogous to the role of the cinematographer in a studio. The artistic parameters Ω_{art} are fixed for that time duration.

Before moving on to the definition of the reward function, we would like to trace a parallel between our choice of coupling reinforcement learning with a low-level planner and the concepts of hierarchical reinforcement learning and option learning [217, 224]. In the RL literature options can be seen as macro-actions that allow our system to reason over multi-step actions. In between the selection of options we can have another policy, planner or controller acting on the robot [224]. This is exactly what happens in our system, with the artistic decisions taking the role of options, and the motion planner making decisions at

a higher frequency (Fig. 5.3). Without loss of generality, we will continue to refer to our artistic decisions as actions instead of options for the remainder of this chapter.

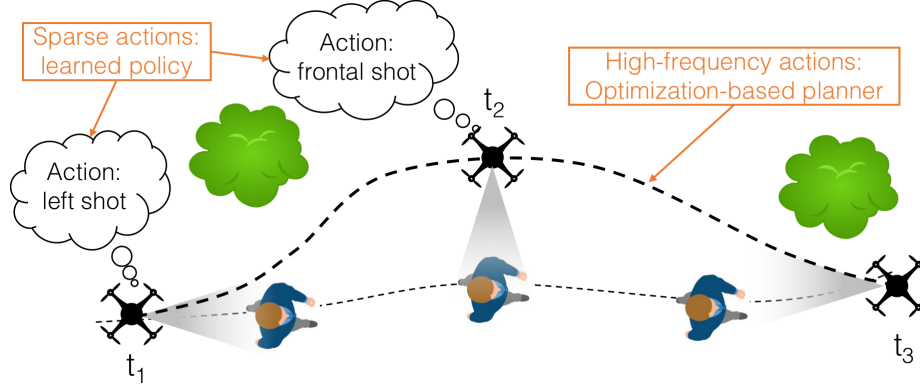


Figure 5.3: Coupling low-frequency sparse artistic decisions with the high-frequency motion planner. We can view our reinforcement learning framework as options [224], where in between decisions a optimization-based planner calculates the best trajectory to follow, considering a smaller time scale.

5.2 Reward Definition

Now we define the artistic reward function R_{art} . At a high level, we define the following basic desired aesthetical criteria for an incoming shot sequence:

- Keep the actor within camera view for as much time as possible;
- Maintain the tilt viewing angle θ_t within certain bounds; neither too low nor too high above the actor;
- Vary the relative yaw viewing angle over time, in order to show different sides of the actor and backgrounds. Constant changes keep the video clip interesting. However, too frequent changes don't leave the viewer enough time to get a good overview of the scene;
- Keep the drone safe, since collisions at a minimum destabilize the UAV, and usually cause complete loss of actor visibility due to a crash.

While these basic criteria represent essential aesthetical rules, they cannot account for all possible aesthetical requirements. The evaluation of a movie clip can be highly subjective, and depend on the context of the scene and background of the evaluator. Therefore, in this work we compare two different approaches for obtaining numerical reward values. In the first approach we hand-craft an arithmetical reward function R_{art} , which follows the basic aesthetics requirements outlined above. In addition, we explore an alternative approach for obtaining R_{art} directly from human supervision. Next, we describe both methods.

Hand-crafted reward:

The reward calculation from each control time step involves the analysis and evaluation of each frame of the video clip. Since our system operates with steps that last 6s, the reward value depends on the evaluation of 180 frames, given that images arrive at 30Hz. We define R_{frame} as the sub-reward relative to each frame, and compute it using the following metrics:

Shot angle: $R_{\text{frame}}^{\text{shot}}$ considers the current UAV tilt angle θ_{rel} in comparison with an optimal value $\theta_{\text{opt}} = 15^\circ$ and an accepted tolerance $\theta_{\text{tol}} = \pm 10^\circ$ around it¹. The shot angle sub-reward decays linearly and symmetrically between 1.0 and 0.0 from θ_{opt} to the tolerance bounds. Out of the bounds, we assign a negative penalty of $R_{\text{frame}}^{\text{shot}} = -0.5$.

Actor's presence ratio: considers the screen space occupied by the actor's body. We set two bounds $pr_{\text{min}} = 0.05$ and $pr_{\text{max}} = 0.10$ based on a desired long-shot scale, actor size of 1.8m, and the camera's intrinsic matrix. If the presence ratio lies within the bounds, we set the value of $R_{\text{frame}} = R_{\text{frame}}^{\text{shot}}$. Otherwise, this parameter indicates that the current frame contains very low aesthetics value, with the actor practically out of the screen or occupying an exorbitant proportion of it. In that case, we set a punishment $R_{\text{frame}} = -0.5$.

We average the resulting R_{frame} over all frames present in one control step to obtain an intermediate reward $R_{\text{step}} = \frac{1}{N} \sum_{i=1}^N R_{\text{frame}, i}$. Next, we consider the interaction between consecutive control steps to discount R_{step} using a third metric: shot type duration.

Shot type duration: considers the duration of the current shot type, given by the count of steps c in which the same action was selected sequentially. We use the heuristic that the ideal shot type has a length 12s, or $c_{\text{opt}} = 2$ time steps², and define a variable discount parameter α_c , as seen in Fig. 5.4. High repetition counts are penalized quadratically to maintain the viewers interested in the video clip.

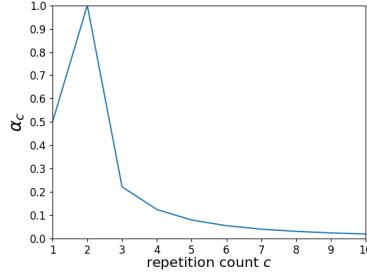


Figure 5.4: Values of the variable discount parameter α_c over shot repetition counts c .

Eq. (5.3) shows how we obtain the final artistic reward R_{art} for the current movie clip. If R_{step} is positive, α_c serves as a discount factor, with the aim of guiding the learner towards the optimal shot repetition count. In the case of negative R_{step} , we multiply the reward by the inverse α_c , with the objective of accentuating the penalization, and to incentivize the policy to quickly recover from executing bad shot types repetitively.

$$R_{\text{art}} = \begin{cases} R_{\text{step}} \cdot \alpha_c, & \text{if } R_{\text{step}} \geq 0 \\ \frac{R_{\text{step}}}{\alpha_c}, & \text{otherwise.} \end{cases} \quad (5.3)$$

In the eventual case of a UAV collision during the control step we override the reward calculation procedure to only output a negative reward of $R_{\text{art}} = -1.0$.

¹The optimal value and bounds were determined by using standard shot parameters for aerial cinematography.

²This heuristics choice was based on informal tests with shot switching frequencies.

Human supervision reward:

We also explore a reward scheme for video segments based solely on human evaluation. We create an interface (Fig. 5.5) in which the user gives an aesthetics score between 1 (worst) and 5 (best) to the video generated by the previous shot selection action. The score is then linearly mapped to a reward R_{art} between -0.5 and 1.0 to update the shot selection policy in the RL algorithm. In the case of a crash during the control step, we override the user’s feedback with a penalization of $R_{\text{art}} = -1.0$.

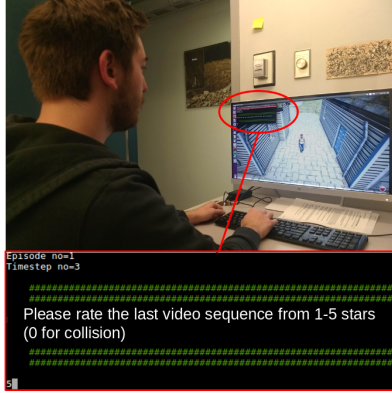


Figure 5.5: Interface for human evaluators in the DQN training procedure.

5.3 Implementation Details

Our DQN architecture is composed of different linear layers which combine the state inputs, as seen in Figure 5.6. We use ReLU functions after each layer except for the last, and use the Adam optimizer [124] with Huber loss [100] for creating the gradients. We use an experience replay (ER) buffer for training the network, such as the one described by [169].

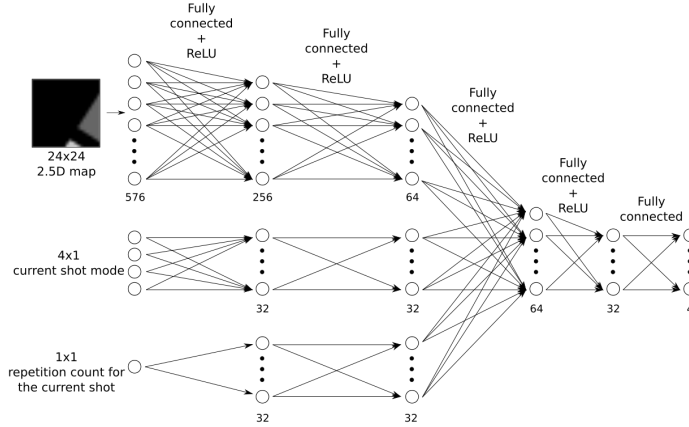


Figure 5.6: DQN architecture for artistic shot selection.

5.4 Results

Next we present detailed results on training and testing the artistic shot selection module, as well as experiments that provide insights to understand which artistic concepts this subsystem is learning. Table 5.1 summarizes our test objectives and results.

Table 5.1: Objectives and results for artistic shot selection

Objectives	Results
Compare policy generalizability to different environments	Learned policies generalized well to new unseen environments. Table 5.2 compares performances.
Analyze role of specific environment context in policy behavior	Policy learned to actively avoid potential occlusions and switch often to keep video interesting (Figs. 5.8-5.9)
Evaluate policies against baselines using real human aesthetics in user study	Our policy outperformed constant shot types or random actions (Table 5.3 and Fig. 5.10)
Transfer policy learned in simulation to real-life environments	We deployed the policy in additional field experiments (Fig. 5.11, Fig. 5.12).

We trained our agent exclusively in simulation, using the Microsoft AirSim release (see Section 4.1.4). We organize our environments in three categories:

- *BlockWorld*: It is generated from a height map, and the actor walks on a path with alternating blocks on the left and right sides. Blocks have varying heights and lengths (Figure 5.7a).
- *BigMap*: It is generated from a height map, and significantly more complex. It is separated into three zones: one that resembles the BlockWorld environment, a second zone with alternating pillars, and third zone with different shapes of mound-like structures (Figure 5.7b).
- *Neighborhood*: Unlike the two previous height maps, this environment is a photo-realistic rendering of a suburban residential area. The actor walks among structures like streets, houses, bushes, trees, and cars (Figure 5.7c).

5.4.1 Learning an artistic policy

Hand-crafted reward: Using the hand-crafted reward definition from Section 5.2, we train a total of six policies in different environments. For all policies except *Neighborhood 1 roam*, the actor walks along a pre-defined path. We define each episode as a concatenation of 5 consecutive time steps, each with a duration of 6 seconds, and we train each policy between 300 to 2000 episodes, depending on the complexity of the environment:

- *BlockWorld 1 and 2*: Trained in different randomized BlockWorld environments, BW1 and BW2, for 300 episodes.
- *BigMap*: Trained in a randomized BigMap environment, BM, for 1500 episodes.
- *Neighborhood 1 and 2*: Trained in different sections of the Neighborhood environment, NH1 and NH2. Trained for 500 episodes.

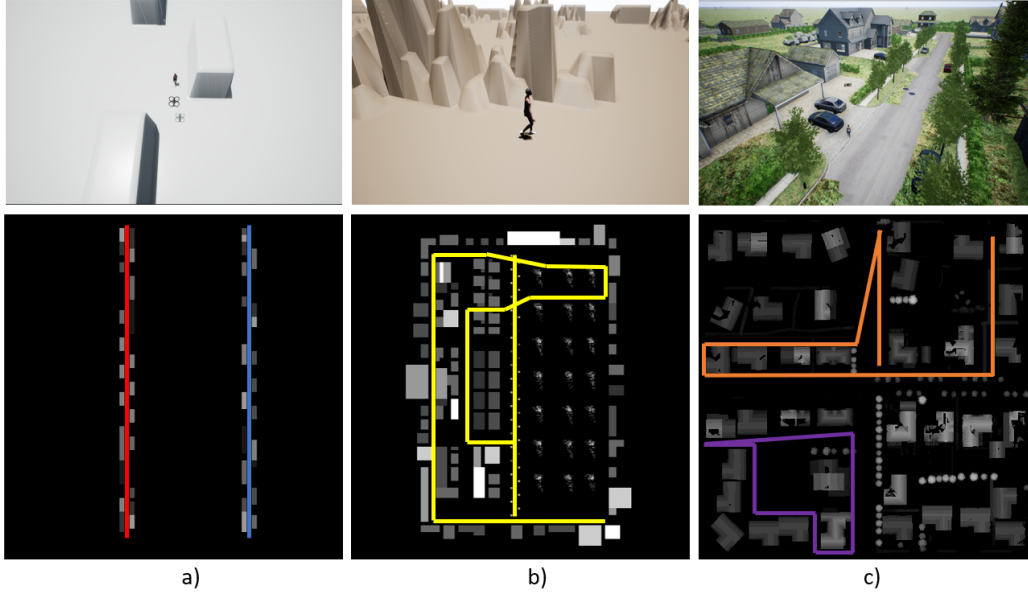


Figure 5.7: Rendering and height maps of AirSim environments with training routes. a) BlockWorld 1 (red), BlockWorld 2 (blue) b) Bigmap (yellow) c) Neighborhood 1 (orange), Neighborhood 2 (purple).

- *Neighborhood roam*: Trained in the entirety of the Neighborhood environment NH, with actor walking in random motion, for 2000 episodes.

We test all policies in all environments to evaluate generalizability. Table 5.2 summarizes the quantitative results. As expected, all policies perform better than random choice, and we achieve highest testing rewards in the same environments the policies were trained in. We also verify that the best generalization performance occurs when policies are trained and tested on the same environment category. It is interesting to note that policies trained on the Neighborhood environments tested on BigMap perform significantly better than those trained on BlockWorld, likely due to the simple geometry of the BlockWorld obstacles.

Table 5.2: Average reward per time step. As expected, policies have the highest rewards when trained and tested on the same environment (bold diagonal). The second-best policy for each environment is underlined and italicized.

Test Env.	BW1	BW2	BM	NH1	NH2
Policy					
BlockWorld 1	0.3444	<u><i>0.3581</i></u>	0.3635	0.3622	0.2985
BlockWorld 2	<u><i>0.3316</i></u>	0.3718	0.3918	0.4147	0.3673
BigMap	0.2178	0.2506	0.5052	0.5142	0.5760
Neighborhood 1	0.1822	0.1916	0.4311	0.5398	<u><i>0.5882</i></u>
Neighborhood 2	0.0813	0.1228	<u><i>0.4488</i></u>	0.4988	0.5897
Neighborhood 1 roam	0.1748	0.1779	0.4394	<u><i>0.5221</i></u>	0.5546
Random choice	-0.0061	0.0616	0.1944	0.2417	0.2047

Figures 5.8 and 5.10 show examples of trajectories generated with trained policies. In addition, Figure 5.9 shows a heatmap with different actions, providing insights into the learned policy’s behavior. Qualitatively, we observe that the learned behavior matches the intended goals:

- Keeps the actor in view by avoiding drastic shot mode switches between opposite positions such as left and right or front and back, which often cause visual loss of the actor;
- Switches shot types regularly to keep the viewer’s interest;
- Avoids flying above high obstacles to keep the shot angle within desirable limits;
- Avoids high obstacles to maintain aircraft safety.

Human-generated rewards: Using the interface described in Section 5.2, we use human aesthetics evaluations as rewards to train two new policies in the BlockWorld and Neighborhood environments for 300 and 500 episodes respectively. Comparisons between both reward schemes are presented next.

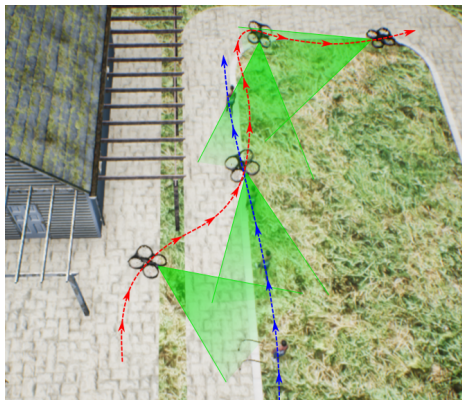


Figure 5.8: Time lapse of drone trajectory during filming in photo-realistic environment. Since the left hand side is occupied, the drone switches from left to front and then right shot.

5.4.2 User study results

We asked ten participants to watch 30-seconds video clips taken from four different policies in five different scenes. Each participant ranked clips from most to least visually pleasing, and wrote open-ended comments on each clip. We chose two scenes from the BlockWorld and three from the Neighborhood environments, and compared the highest-performing handcrafted reward policy for each environment against the human-generated reward policy. In addition, we included a constant back shot policy and a random action policy for comparison.

Table 5.3 summarizes the user study results, and Figure 5.10 shows the best-rated drone trajectories for each scene. As seen in Table 5.3, the trained policies using both reward schemes have higher ratings in all scenes than the random or constant back shot policies.

On average, the hand-crafted rewards trained policies which were ranked better than those trained with human-defined rewards, although participants’ opinions were the opposite in some cases. We also summarize the participant’s comments on the clips:

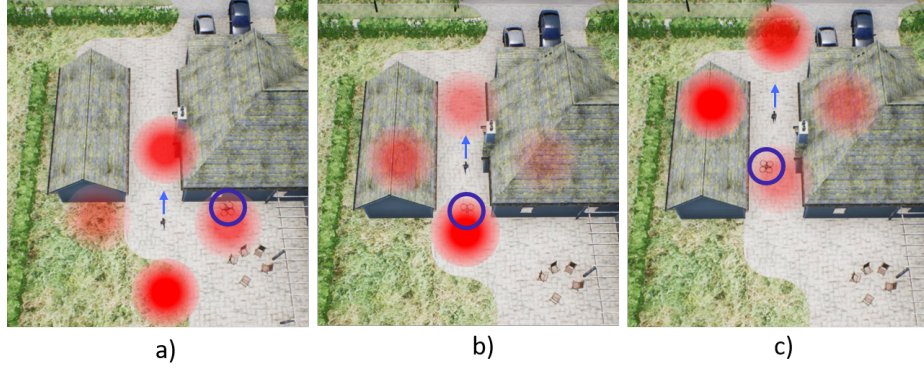


Figure 5.9: Visualization of the Q-values of the DQN during testing corresponding to the 4 shot types. The more opaque a circle is, the higher is the action’s Q-value. The drone position before each decision is indicated by a blue ring around the drone. The drone starts on the right side of the actor and switches to back shot mode to traverse a narrow passage (a) where it stays in the actor’s back for one time step (b). Finally it decides to switch to a left side shot once the obstacles are passed (c).

- All participants criticize the back shot policy as ‘boring’ or ‘unexciting’;
- All participants mention that the random policy loses view of the actor too often;
- The most common aesthetics complaint is due to loss of actor visibility;
- Participants often complained about too little camera movement when only one shot type is used for the entire 30s clip. They also complained about camera movements being visually unpleasing when the shot type changes at every time step (every 6s);
- Participants frequently mention that they like to see an overview of the surrounding environment, and not only viewpoints with no scene context where the actor’s back-ground is just a building or wall. Clips where the UAV provides multiple multiple viewpoints were positively marked;
- The human reward policy was often described as the most exciting one, while the handcrafted reward policy was described as very smooth.

Table 5.3: Average normalized score of video clips between 0 (worst) and 10 (best)

	Average	Scene 1	Scene 2	Scene 3	Scene 4	Scene 5
Hand-crafted reward	8.2	10.0	5.3	9.3	7.7	8.7
Human reward	7.1	5.0	9.0	6.0	7.7	8.0
Back shot	3.8	4.0	4.7	4.3	4.0	2.0
Random	0.9	1.0	1.0	0.3	0.7	1.3

The main perceived difference between the handcrafted reward policy and the human reward policy was the consistency of switching shots. While the former tries to switch shots every 2 times steps (12s) if not disturbed by an obstacle, the latter is more irregular in timing. From the user studies it is evident that a more regular period that is neither too short nor too long improves aesthetic scores.

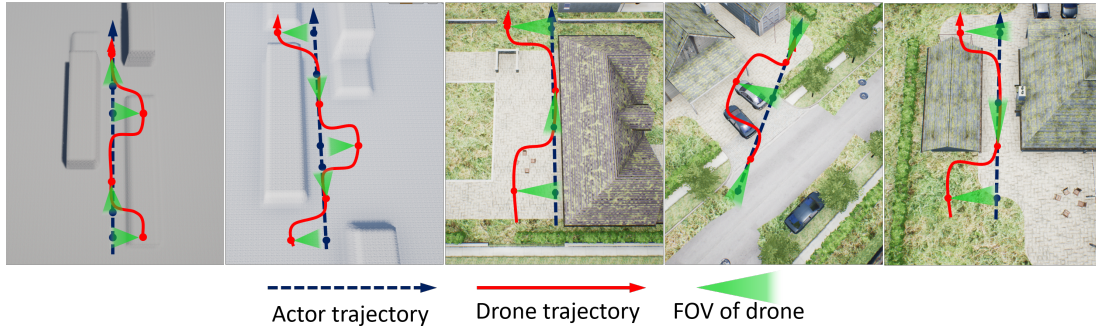


Figure 5.10: Drone trajectories of the highest rated policies in the user study in scenes 1 to 5.

5.4.3 Extended results in field experiments

In addition to the simulated results, we tested our trained policies in real-life settings. We filmed scenes using three distinct policies: one trained with handcrafted rewards on the *BigMap* environment, a second fixed back shot policy and a third random policy. First, in Figure 5.11 we show shot selection results that operate using a pre-mapped environment. In addition, in Figure 5.12 we show results where we employ the online mapping module to build a height map for the DQN on-the-fly.

Similar to the simulation results, the random policy results in constant loss of actor due to drastic position changes and close proximity to obstacles. For example, a random action of right shot will cause the UAV to climb much above the actor if it is too close to a large mound. The back shot policy, although stable in vehicle behavior, results in visually unappealing movies. Finally, our trained policy provides a middle ground, resulting in periodic changes in UAV viewpoint, while providing stable visual tracking.

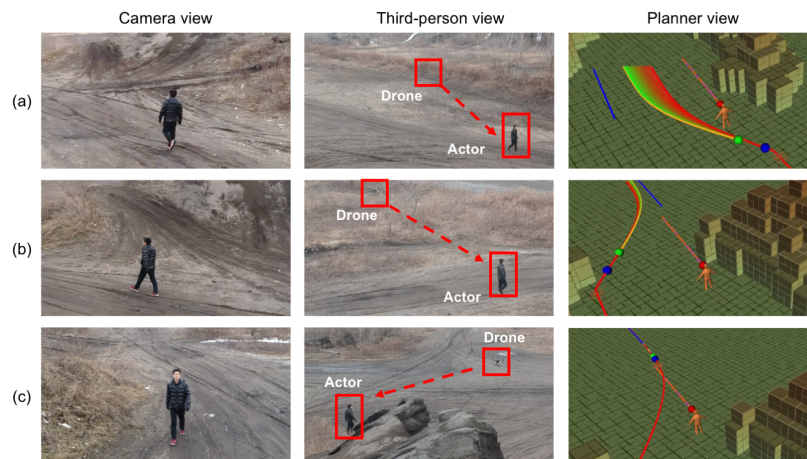


Figure 5.11: UAV follows an actor while switching shot types autonomously with our trained policy: a) UAV starts at the back of the actor; b) The mound's presence on the right side of the actor leads to a left shot selection; c) UAV switches to a frontal shot in the open area.

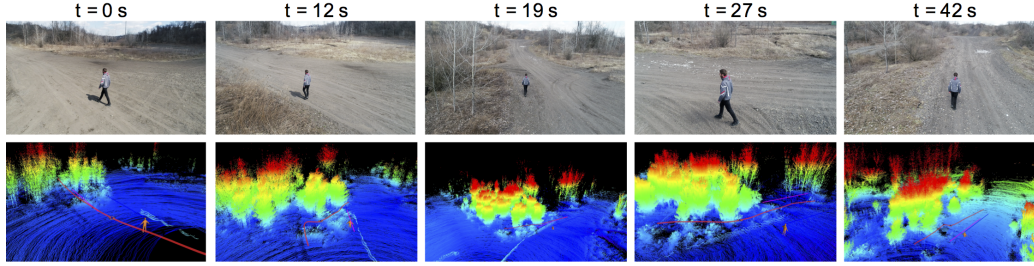


Figure 5.12: Field test results using the online artistic shot selection module. UAV trajectory is shown in red, actor motion forecast in blue, and desired shot reference in pink. The UAV initially does a left side shot at $t = 0\text{s}$ in the open field, but as a line of trees appear, it switches to a back shot at $t = 12\text{s}$. When an opening appears among the tree lines, the UAV selects a left side shot again at $t = 27\text{s}$, and when the clearance ends, the module selects a back shot again.

5.5 Conclusion and discussion

In this work, we presented a fully autonomous drone director that follows a moving actor while making intelligent decisions about the shot type in real time. These decisions are based on previous experience, gained during training via deep RL with a hand-crafted as well as a human-generated reward function. Our approach works robustly in realistic simulation environments as well as in real world tests on a physical drone and successfully generalizes to previously unseen environments. The decisions about the shot direction acknowledge the environment around the actor and follow cinematographic principles such as occlusion avoidance, flat shot angles and frequent camera angle switches. Our user study confirms that our trained policies satisfy the human sense for aesthetics and offers insight into possible future improvements of the algorithm.

Our work opens up a broad field that remains largely unexplored in the robotics research community, and we identify several open research questions for future extensions of this work:

- Instead of learning a reward function for artistic choices that is general across participants, can we learn individual preferences from users?
- While training such systems, how can we properly deal with the non-stationary nature of the human-provided aesthetic reward function? For example, if ratings vary from 0 to 5, what a user rates as a 5 during the initial phases of training when most choices are fairly random will arguably be a worse behavior than what the same user will rate as a 5 after hours of training.
- Instead of using human supervision for the reward value directly, can we achieve a more efficient training scheme using different techniques? For example, we can use the human feedback to iteratively learn an approximator for the reward function, and remove the human from the main RL training loop. As an example, in [47] the authors use humans to compare the quality of two actions at a time for approximating the reward function.

It is important to note that the initial goal of this chapter, that of learning artistic decision-making, is likely an objective that cannot ever be fully accomplished. The very definition of

artistic quality or beauty is intrinsically a subjective quantity, which can vary drastically depending on the human evaluator. As the popular proverb says, *beauty is in the eye of the beholder*. Mathematically, the approximation we make in this work is that the *average* of camera behaviors judged as being good by a series of human evaluators during the training process will also be good during execution. Next, our user studies prove that on *average* our hypothesis is true. However, we can define the problem of learning artistic behaviors with an infinity of training and evaluation criteria, tending more or less to personalization to a specific group of users or a particular individual.

In addition, we envision further research in learning the artistic reasoning behind human choices in several contexts beyond the one of movie productions. More broadly, as robotics evolves, autonomous agents are required to operate in a large variety of tasks in proximity to humans, where success is in great part measured by the ability of the robot to execute *aesthetic* and *human-like* behaviors. We identify important related areas to cinematography, such as autonomous driving and human-robot interaction, where fine nuances of human behavior modeling are important for the development of autonomous agents.

Chapter 6

Learning semantic camera controls

Despite great advancements in autonomous flight technology, generating expressive camera behaviors is still a challenge and requires non-technical users to edit a large number of unintuitive control parameters. So far in Chapter 5 we described an automated system that can directly output the most appropriate shot types capable of maximizing a average artistic reward for the film. In contrast, in this chapter our goal is not to replace the human in the loop; rather, we provide an intuitive interface that allows humans to control high-level scene parameters. We develop a data-driven framework that enables editing of the complex camera positioning parameters in a semantic space (*e.g.* calm, enjoyable, establishing).

First, we generate a database of video clips with a diverse range of shots in a photo-realistic simulator, and use hundreds of participants in a crowd-sourcing framework to obtain scores for a set of semantic descriptors for each clip. Next, we analyze correlations between descriptors and build a semantic control space based on cinematography guidelines and human perception studies. Finally, we learn a generative model that can map a set of desired semantic video descriptors into low-level camera trajectory parameters. We evaluate our system by demonstrating that our model successfully generates shots that are rated by participants as having the expected degrees of expression for each descriptor. We also show that our models generalize to different scenes in both simulation and real-world experiments. The supplementary video shows more details of the system in action: https://youtu.be/6WX2yEUE9_k.

6.1 Introduction

As discussed in the previous chapters, aerial vehicles have become an important tool for supporting human creativity and expressiveness, fundamentally altering the way both professional and amateur users produce media content for movies, sports and virtual / augmented reality. Much of the impact of aerial cameras stems from their capability to compose aerial and dynamic viewpoints which are infeasible using traditional devices such as hand-held cameras and dollies [204]. In addition, recent developments both in industry [61, 211] and academia [20, 98, 105] now allow drones to detect, track and follow objects of interest autonomously while maintaining safety in cluttered environments.

However, a major limitation of today’s cinematography platforms is the difficult and unintuitive interface for camera control. There are infinite ways in which one can capture

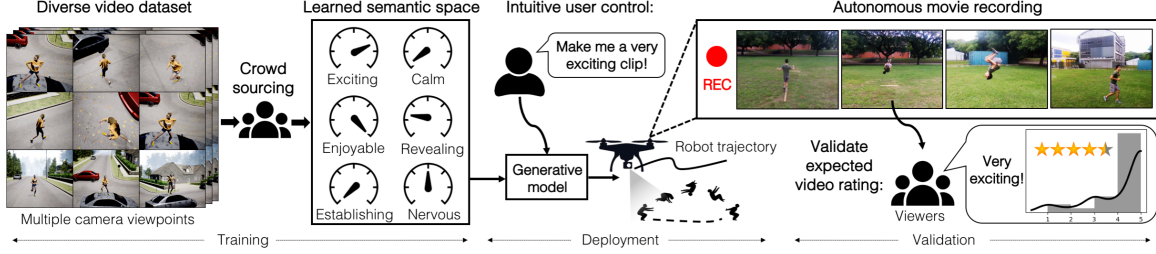


Figure 6.1: Our framework uses a diverse set of video clips to learn a semantic descriptor space from crowdsourced ratings. During deployment, a user can intuitively manipulate the robot camera motion with descriptors instead of tuning low-level robot positioning parameters.

a scene. Even for identical scenes, each camera path provides a different visualization of the story [182], causing distinct impressions on the viewers. Users are not able to directly control the emotional expression of the final footage. Instead, they need to carefully tune the camera’s position, velocities and angles in order to achieve a desired expression. An added challenge is the complex interaction between camera parameters, which produces a combinatorial complexity that drives the viewer’s perception. This task is cumbersome to users not only due to the large parameter search space, but also because it requires intimate knowledge of cinematographic rules [5, 26, 215], which can take years to master.

Our work aims to fill in the gap in existing controls for autonomous cameras. As seen in Fig. 6.1, our framework enables users to determine the desired shot types based on intuitive and perceptually meaningful parameters (*e.g.* exciting, enjoyable, establishing). We employ a photo-realistic simulator to generate a database of video clips with a diverse set of shot types, and use crowd-sourced perceptual experiments to obtain semantic scores for each video. We then learn the two-way mapping between low-level shot parameters and semantic descriptors. We take inspiration from previous works that created semantic control spaces for domains such as cloth animations [209], object and body shapes [218, 244], and robot motion [59]. Our contributions are threefold:

- 1) Perceptual experiments:** First, we conduct a series of experiments to determine the minimal perceptually valid step sizes for different shot parameters. We then build a dataset of 200 videos using variations along these units, and use an interactive crowd-sourcing platform to obtain numerical scores for different subjective semantic descriptors;
- 2) Semantic control space:** Using the perceptual scores, we learn a regression model that provides an intuitive semantic control space to re-parameterize the desired video expression into low-level shot parameters to guide the aerial camera;
- 3) Experimental validation:** We validate the learned models in a series of experiments in simulation and real-world tests. We show that shots generated from the semantic space are rated by participants as having the expected degrees of expression for each attribute, and that the model generalizes to different actors, activities, and background compositions.

6.2 Related Work

Autonomous aerial cinematography: We find multiple lines of work on the use of drones for cinematography. For instance, [195] and [171] use optimization methods for navigation on or close to pre-defined trajectories segments. We also find works that use

pre-selected high-level cinematographic guidelines such as distances and angles relative to actors [5, 26] as an input for trajectory optimization in unscripted scenes among obstacles [17, 20]. Alternatively, [99] control the camera based on image projection features.

Autonomous artistic reasoning: Autonomous reasoning about how camera movements can provide artistic value to videos has been a topic of interest in multiple contexts. For instance, [41] and [42] train policies to control pan-tilt-zoom cameras in basketball and soccer games by imitating human demonstrations and using video features. Specifically on drone cinematography, [85] use reinforcement learning to train a policy that switches between four basic shot types to maximize human-provided rewards. We also find works that aim to imitate different camera motion styles. For example, [9] use trajectory optimization so that a flying camera can imitate the idiosyncrasies of hand-held camera motions. [97] use supervised learning to predict camera actions for different shot types, and [106] build a latent space of shot types out of movie examples, which is coupled with a generative model.

Sentiment analysis: Unlike the previous works which were focused on *imitating* a particular behavior, sentiment analysis focuses on *understanding* how humans perceive and interpret visual content [32]. Most video sentiment models are based on psychology literature and use a combination of the Valence-Arousal-Dominance state model [164] with semantic descriptors grouped by similarities [200]. Several authors explore the relationship between image features and sentiments [13, 33, 160], and train models for movie recommendation, summarization, and information retrieval [33, 89].

Learning a perceptual control space: Our approach is motivated by previous works that edit processes using semantic or context-specific attributes, as found in the fields of garment simulations [209], 3D shapes [218, 244], and images [137]. Closest to our domain in robotics, [59] developed a data-driven interface for semantic design of expressive robot motion. All these past works share a common approach: they first build a dataset connecting instances to their intuitive semantic labels (usually via crowd-sourcing), and then use this data to learn a generative model mapping the semantic descriptors back to unintuitive low-level parameters.

Our work is the first to develop a perceptual control space for designing camera trajectories. We allow users without any domain expertise in aerial vehicles nor cinematography to capture expressive camera behaviors. Unlike previous works that imitate styles, our system does not require the user to search for specific examples of videos they want to indirectly emulate. As we detail next, we show that we can generate any footage by directly selecting its desired semantic descriptors.

6.3 Perceptual Experiments

Our overarching goal is to learn a perceptually meaningful space for aerial camera control. We focus our study on single-actor shots, and assume the existence of sparse sets of obstacles such as the ones found in suburban environments (*e.g.* trees, telephone poles, traffic signs). All simulated data is recorded using a drone in the photo-realistic environment AirSim [207] (Fig. 6.2a) coupled with a custom ROS interface [197]. We limit our clips to 15 second segments, which is a reasonable unit of length for individual shots [53, 163]. As seen in the supplementary video, the main scene consists of an animated character running down a street, avoiding a car parked on the road, and finally jumping on top of another vehicle and

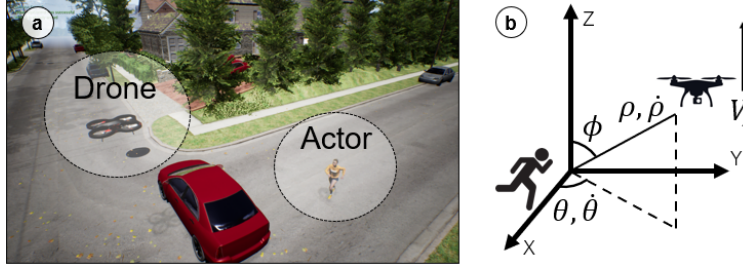


Figure 6.2: Experimental setup: a) Photo-realistic simulator with an animated character and aerial vehicle; b) Shot parameters that define the vehicle’s position relative to the actor, in spherical coordinates.

dancing. We chose this actor motion because it contains both static and dynamic segments, and causes a relatively neutral emotional impression on the viewer.

We employ the base motion planner from [17], which uses a trajectory optimization method to avoid collisions and occlusions with the environment. Within this framework, a shot is parameterized by the positions and velocities of the drone relative to the actor. We define a shot using spherical coordinates (Fig. 6.2b): $\Omega_{shot} = [\rho, \dot{\rho}, \theta, \dot{\theta}, \phi, V_z]^T \in \mathbb{R}^6$, where ρ and $\dot{\rho}$ are the distance and velocity towards the actor, θ and $\dot{\theta}$ are the horizontal angle and angular velocity, ϕ is the tilt angle and V_z the vehicle’s vertical speed.

6.3.1 Minimal Perceptual Units for Shot Parameters

Our approach requires us to build a mapping between shot parameters and semantic descriptors. To do so, we must sample the manifold containing all possible shot variations. Within the cinematography literature [5, 26] we find canonical sets of values for parameters like the distance to the actor ρ (close-up, medium, and long shots) and tilt angles (variations every 45°). However, a naive approach for sampling the remaining parameters results in a prohibitively large state space given that some of its parameters such as angle rates are virtually unbounded. In addition, parameter steps that are too small may render imperceptible changes in the final video, resulting in redundant samples.

To address these issues, we learn a minimally perceptible unit of measure along each dimension in the shot parameter space. A major challenge arises because this is a high-dimensional space, and metrics are not globally consistent (*e.g.* variations in angular velocity will be more perceptible when the camera is closer to the actor).

In order to make our experiments tractable we adopt the same simplifying assumptions of [209], and evaluate local axis-aligned perceptual steps around a set of meaningful shot preset configurations. In Table 6.1 we define 6 shot presets based on common aerial shot types: the static shots (*Follow 0/1*) maintain a constant relative position between camera and actor, and Fig. 6.3 depicts the dynamic shots.

For each parameter $\phi, \dot{\rho}, \dot{\theta}, V_z$, we sample approximately 10 linearly distributed variations around its preset value over the maximum range. We only allow changes that kept the shot within its original type. Table 6.1 highlights the variable parameters in bold. We create a total of 84 videos, and for each variation we perform a two-sided t-test analyzing if the

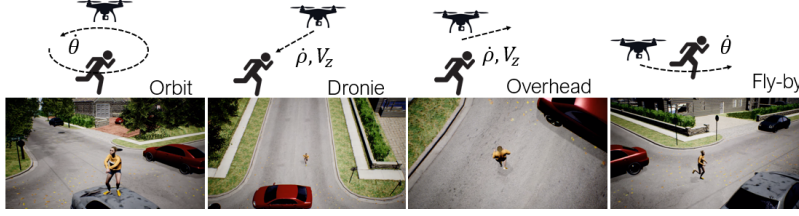


Figure 6.3: Examples of dynamic shot presets used in the study.

Table 6.1: Shot presets for perceptual units study. We only vary the parameters in bold for each preset.

	ρ	θ	ϕ	$\dot{\rho}$	$\dot{\theta}$	V_z
Unit	[m]	[°]	[°]	[m/s]	[°/s]	[°/s]
Preset						
Follow 0	8	0	20	0	0	0
Follow 1	8	135	20	0	0	0
Orbit	5	0	20	0	20	0
Dronie	25	0	45	-0.5	0	-0.5
Overhead	8	180	85	0	0	0
Fly-by	15	150	20	0	-8	0

resulting video is perceived as the same or different from the preset video (with $p = 0.05$ significance).

For this first web survey (WS1) we recruited over 200 participants using Amazon Mechanical Turk (MTurk) [3]. This research received a waiver from our Institutional Review Board, and participants were compensated for their time. After being approved on a short qualifying task, each participant viewed a total of 12 pairs of videos, one being the preset and the other being either a variation or the preset against itself. Videos were played asynchronously three times (only one video played at a time), and after watching at least once, participants answered: “*Is the camera perspective the same or different in the two clips shown?*” Each clip was compared 30 times against its preset.

Figure 6.4 displays our results showing the minimal perceptual units around each preset and parameter. As expected, we find that not all noticeable variations are symmetrical around the presets, and deltas are only locally consistent. For example, a change larger than $\Delta\phi = 5^\circ$ in tilt is noticeable for *overhead* shots, while a larger delta $\Delta\phi = 10^\circ$ is required for *follow* shots, where the preset angle is lower. Humans are surprisingly good at noticing variations in angular speed: deltas larger than $\Delta\dot{\theta} = 2.5^\circ$ were already noticeable.

6.3.2 Obtaining Semantic Scores for Videos

Once we are able to generate a diverse dataset of videos with the minimal perceptual units of each shot parameter, the next step is to devise a method to obtain numerical scores for the subjective semantic descriptors representing each clip (our choice of descriptors is detailed in Section 6.3.3). To do so, we use *TrueSkill* [93], which is a relative ranking algorithm based on pairwise comparisons, similar to the Elo chess-player rating algorithm [65]. Pairwise comparisons have been shown to be more consistent and render more accurate results than

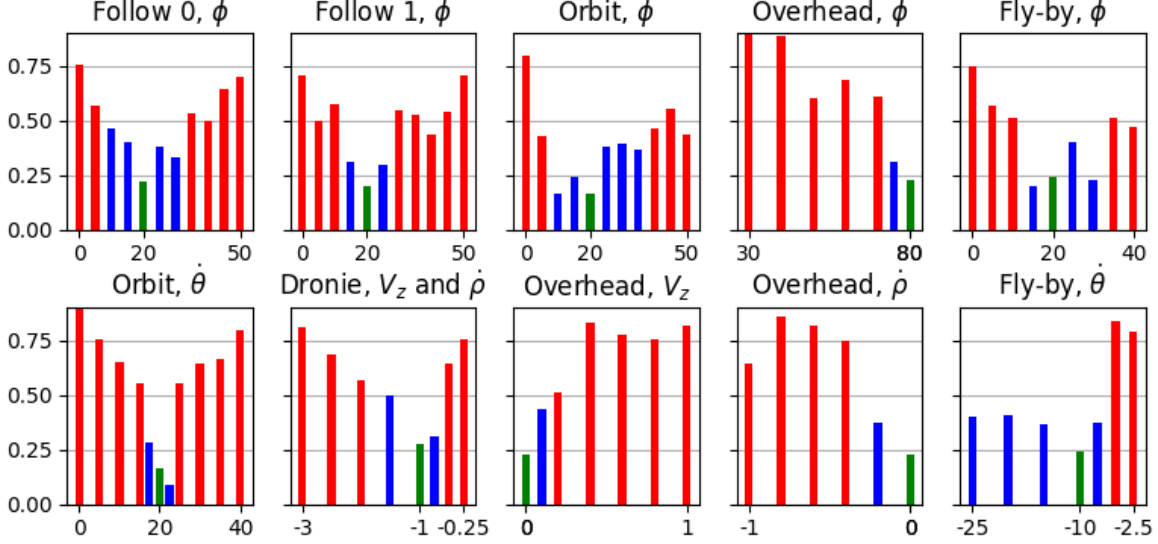


Figure 6.4: Minimal perceptual deltas for each parameter. The preset value is shown in green, red indicates statistically significant perceptual variations ($p = 0.05$), and blue are the insignificant variations. Heights display the percentage of videos rated as being different than their preset.

absolute scales for subjective scores [13, 166, 201, 242]. We model the score for each semantic descriptor d_i as a Gaussian distribution $d_i = \mathcal{N}(\mu_i, \sigma_i)$, and update both mean and variance of each pair of samples after each comparison.

6.3.3 Building a Semantic Descriptor Space via Crowd-Sourcing

We initially compiled a list of 15 semantic descriptors (Table 6.2) which are commonly used to refer to subjective qualities of images and videos in the cinematography and psychology literature [5, 13, 26, 32, 33, 201]. Since our scenes do not include dialogs nor soundtracks, we do not include descriptors which are uniquely associate with audio features.

A camera control space consisting of 15 descriptors is still not an intuitive interface for non-expert users. Therefore, in order to reduce the cognitive load on the operator, our first semantic study targeted reducing the space to a smaller number of descriptors. To do so, we generated a database of 50 distinct 15-second clips using each shot preset’s minimal perceptual units. We randomly sampled parameter variations for all axis simultaneously, and allowed positive and negative deltas using multiples of $\{0, 0.5, 1.0, 1.5, 2.0\}$ from the perceptual units.

To obtain semantic scores for each shot, we designed a second web survey (WS2) on MTurk. This time, each user was shown one pair of clips at a time, which played synchronously three times. After watching the videos at least once, users answered five questions of the type “Which video is more ___?”, where ___ denotes different descriptors. Each user watched a total of 12 video pairs, and passed a qualifying task before the survey. We processed answers from a new set of 200 users using *TrueSkill* to obtain a descriptor vector $d \in \mathbb{R}^{15}$ containing the relative scores of each semantic descriptor for each clip. We analyzed the similarity between all pairs of descriptors using correlation coefficients, as seen in Figure 6.5.

Table 6.2: 7 Clusters of the 15 original descriptors shown within brackets [], and divided by emotion axis and direction. We place no descriptor as representing negative valence, but low values of *interesting* and *enjoyable* can span this emotion.

Axis	Arousal	Valence	Dominance
Positive	[Exciting , Surprising, Rushed, Dynamic]	[Interesting], [Enjoyable]	[Establishing], [Revealing]
Negative	[Calm , Slow, Predictable, Boring, Serene, Static]	NA	[Nervous]

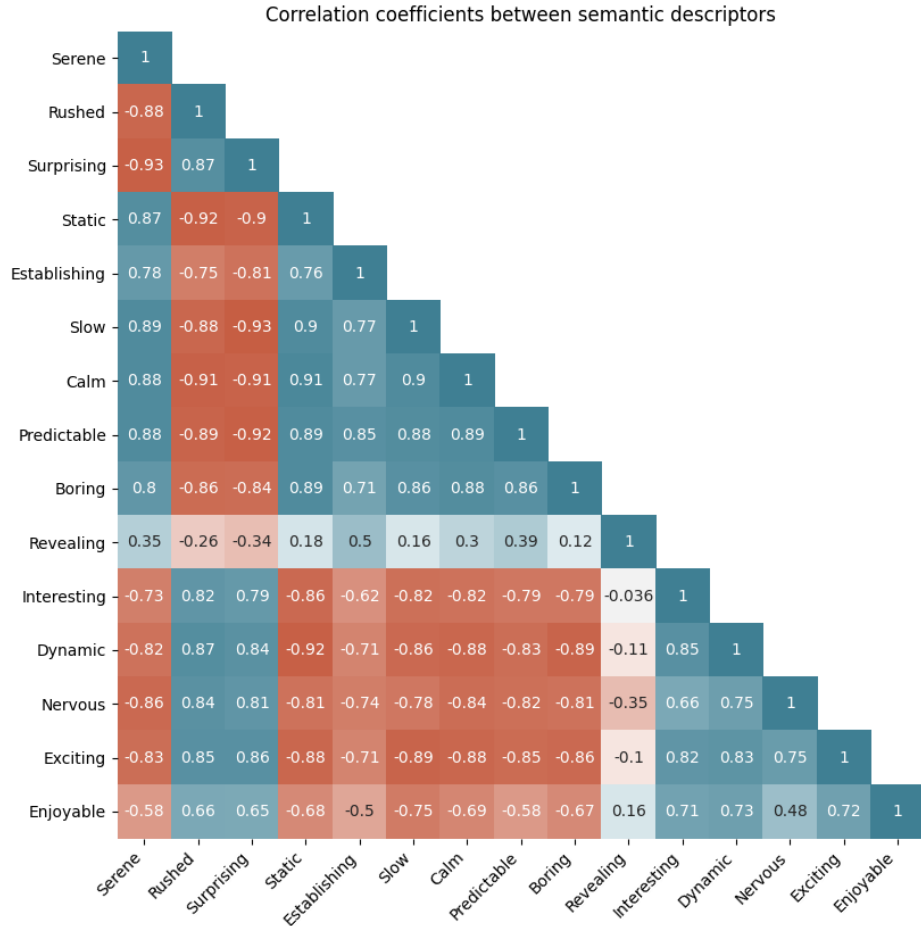


Figure 6.5: Correlations between a set of 15 semantic descriptors within a diverse dataset of 50 videos. Data processed from 200 user surveys.

We found that some groups of descriptors present large positive and negative correlations with one another. For instance, videos labeled as being very *calm* also present high scores for *slow*, and low scores for *dynamic*. We use the Affinity Propagation algorithm [70] to cluster groups of descriptors with high correlation. As seen in Table 6.2, we reduce the dimensionality of the descriptor space by building a total of 7 groups of descriptors, with one descriptor representing the group for subsequent studies. The choice of final number of clusters is ambiguous, and any value between 1 and 15 clusters would have been possible by varying the *preference value* in the Affinity Propagation algorithm. Our choice of 7 clusters

attended two criteria: (i) it presents a small enough number of parameters that a user can interact within an interface, and (ii) the resulting 7 clusters can be grouped to span all axes of the Valence-Arousal-Dominance [164] space, which is widely used in affective content analysis [13].

Next, we generated the final dataset for learning the mapping between semantics and shot parameters, now using a larger sample of 200 randomly generated videos sampled around the preset values. Analogously to WS2, we deployed a third perceptual rating survey (WS3) to compute a 7-dimensional descriptor vector for each video ($d \in \mathbb{R}^7$) using answers from a new set of 500 participants who passed a qualification test. In order to verify our initial assumption that the resulting clusters are able to span the full space of 3 emotions axis, we again calculate a correlation matrix between descriptors. This time we also artificially create negated scores by mirroring each descriptor around its mean value, and then use a multidimensional scaling method [133] to find the best 3D coordinates that fit our data, treating correlation scores as distances between points. Figure 6.6 shows that our experimental data in fact is able to span a 3D affective space, as seen by our fitted Valence-Arousal-Dominance basis vectors. The vectors are not fully orthogonal, indicating correlations among emotion axes, which is an expected result based on psychology literature [201].

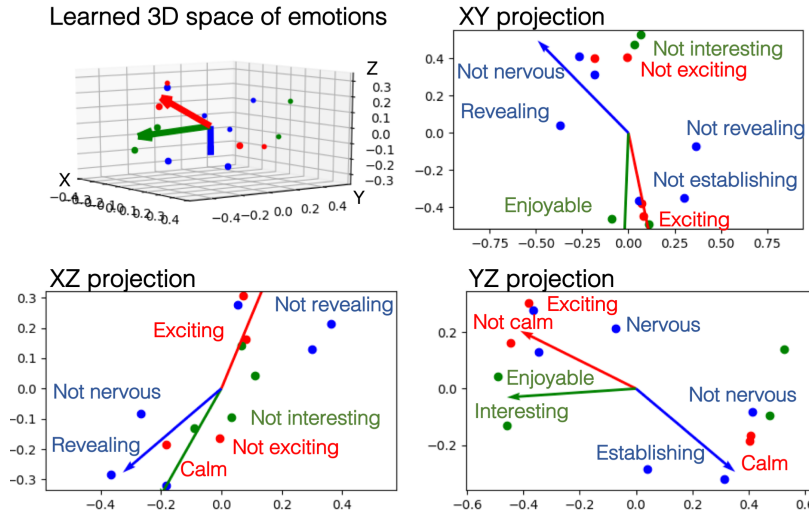


Figure 6.6: Learned 3D space of emotions computed from a survey with 200 videos and 500 participants. We display 7 semantic descriptors and their negated values (*e.g.* Calm, Not calm), and fit a emotion basis vector along each major emotion axis: Arousal (red), Valence (green) and Dominance (blue).

6.4 Learning a Semantic Control Space

Given the dataset of shot parameters $\Omega_{shot} \in \mathbb{R}^6$ and semantic descriptors $d \in \mathbb{R}^7$, we wish to learn a function $f : d \rightarrow \Omega_{shot}$ that provides a mapping from a set of desired descriptors to the set of shot parameters that produces a clip with such emotional impression on the viewer (D2P model). We are also interested in the inverse question: given a shot, can we infer its emotional expression as a mapping $f^{-1} : \Omega_{shot} \rightarrow d$ (P2D model)?

6.4.1 Training Details

We explored two approaches to learn such functions: linear regression (LR) and deep neural networks (DNN). We employ Lasso regression [228] for our linear model, as it includes an additional loss to reduce the $L1$ norm of coefficients. We tested different DNN architectures, and after a series of tests using 5-fold cross validation on our dataset, we settled with a fully connected network with 3 hidden layers containing 32, 16, and 8 neurons respectively. We augment the shot parameter vector with 5 additional variables corresponding to a one-hot encoding of the shot type (follow, orbit, drone, overhead, fly-by), which is used to process the model’s output into one of the canonical drone shot types. All values are normalized to the $[-1, 1]$ range for training.

6.4.2 Model Results

Both LR and DNN presented very similar performance on the D2P model using a dataset of 200 videos, with $R_{LR}^2 = 0.19$ and $R_{DNN}^2 = 0.22$. Given this similarity and the fact that LR allows us to interpret the values of its coefficients more easily, the rest of our analysis here uses linear models. We note, however, that we trained both models with a relatively small quantity of examples due to the high cost of obtaining labeled data. If given more data, a DNN model would very likely achieve superior performance.

Figure 6.7a shows the normalized coefficients for the D2P model. We notice intuitive relationships between parameters which were learned entirely from user evaluations. For instance, *establishing* videos are linked to larger distances ρ to the actor, matching the cinematographic concept of establishing shots. Other strong relationships we identify are of *calm* videos having smaller angular velocities $\dot{\theta}$, and enjoyable clips tending to veer towards lower tilt angles ϕ , away from overhead shots.

We also investigate the P2D i.e. inverse model, which maps shot parameters to descriptor values. Figure 6.7b shows the relationships between variables. The distance between camera and actor causes the largest impact on viewer perception. Other shot parameters have a sparse influence on descriptors. For instance, larger tilt angles ϕ produce less revealing and less interesting videos, but have less influence on how calm a video is.

6.5 Experimental Validation

6.5.1 Semantic Control Space

We validated the quality of the learned semantic control space in a series of user experiments. Using the D2P model, we generated a set of videos per semantic descriptor by linearly sampling increasingly low to high desired expression values over the range of $[-2, 2]$ standard deviations. The choice of values for the descriptor vector is not entirely arbitrary, as several of the attributes present positive and negative correlations. For instance, if the user is only interested about the value of a single descriptor and sets a high score for *exciting*, the *calm* score must naturally go down. Therefore, we treat the descriptor vector d as a multivariate normally distributed variable $d = [d_1, d_2]^T$, where $d_2 \in \mathbb{R}^n$ are the n user-specified scores. We then calculate the remaining scores d_1 using the means and covariances between groups of variables: $d_1 = \mu_1 + \Sigma_{12}\Sigma_{22}^{-1}(d_2 - \mu_2)$.

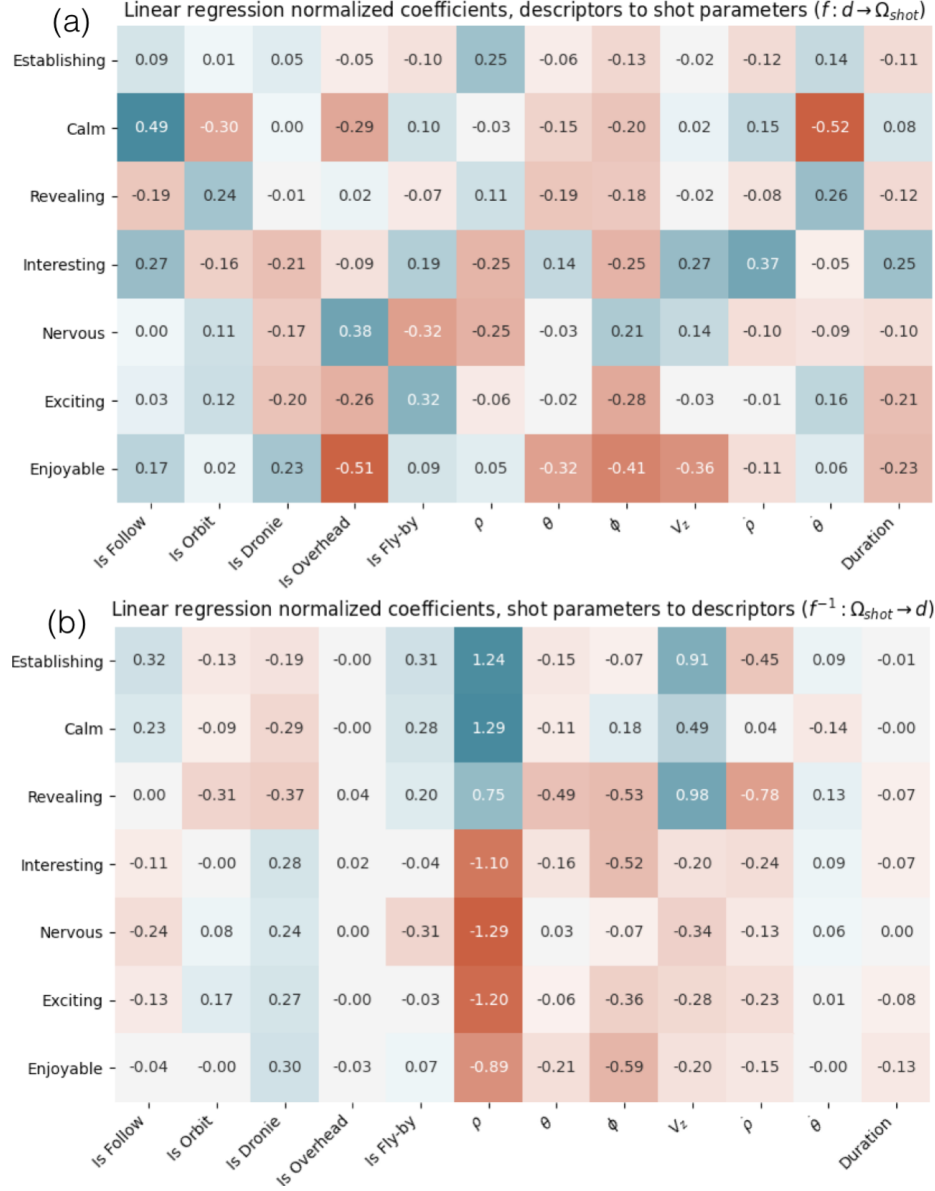


Figure 6.7: Normalized linear model coefficients: a) mapping from semantic descriptors to shot parameters (D2P model), and b) mapping from shot parameters to semantic descriptors (P2D model).

In contrast to our previous experiments, here we created a new survey on MTurk (WS4) where users viewed each clip individually, and scored them using a 5-point Likert scale (1 being *Not ___ at all*, and 5 being *Extremely ___*). We averaged scores from 15 users per clip. An absolute scale is preferred for this study because our goal is to understand how average users perceive the resulting clips, and not only to create a relative video ranking.

As seen in Figure 6.9, we test our model in two simulation environments using 42 videos (6 per descriptor axis). The first environment is identical to the one employed for learning the models, and the second one tests how well the D2P model generalizes to a new actor type

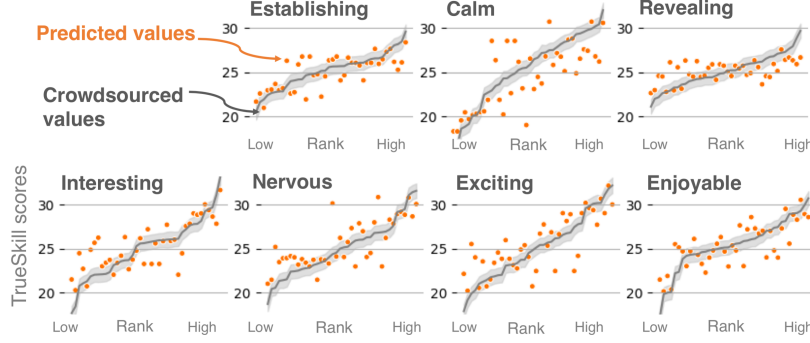


Figure 6.8: Comparison of semantic scores predicted from shot parameters using the learned model (orange) against the original crowdsourced values (gray). Shaded gray area displays TrueSkill score uncertainty. Videos are ordered in ascending order of scores.

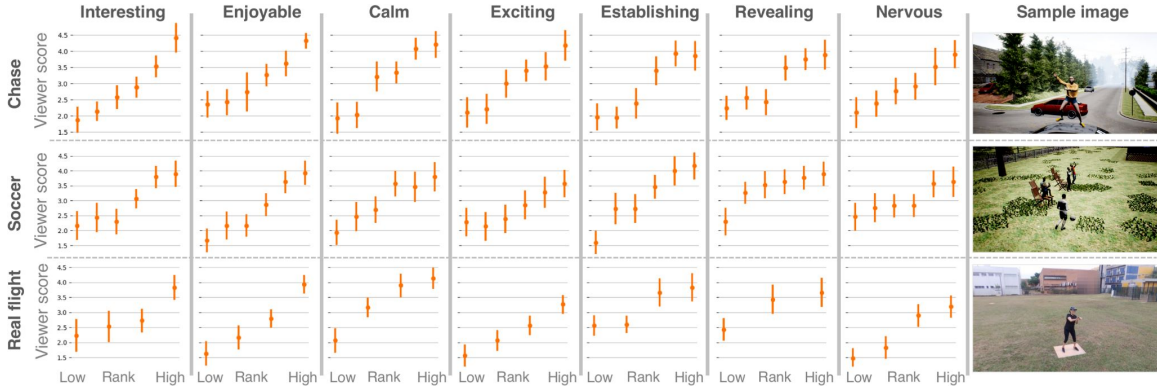


Figure 6.9: User study results validating generative shot model for different simulated and real-world scenes. For each descriptor we generated videos ranging from low to high desired scores, and averaged the perception score from 15 users on a 5-point absolute Likert scale. Perceived scores match the desired ascending ranking.

(soccer player dribbling a ball), scene background (grass field with an audience cheering), and shot duration (20s, up from 15s earlier).

In addition, we also validate our model in real-world experiments, collecting a total of 28 clips (4 per descriptor axis). We use a Parrot Bebop 2 drone and control it with the real-time autonomy stack developed by [20] to visually localize the actor and plan smooth aircraft motions. We collected additional shots (Fig. 6.11) shown in the supplementary video.

Figure 6.9 shows that our D2P model is able to successfully generate shots that are rated by participants as having the expected degrees of expression for each descriptor. Furthermore, the model generalizes well to other simulated scenes and to real-world footages, which strongly suggests that our semantic control space is not overly attached to specific features of the training environment nor to a single set of actor motions.

6.5.2 Semantic Shot Classification and Latent Space Analysis

We also validate the quality of the P2D model on a held-out test set with 20% of the 200 videos. Fig. 6.8 shows the quality of the model’s predictions per emotion ($R^2 = 0.69$).

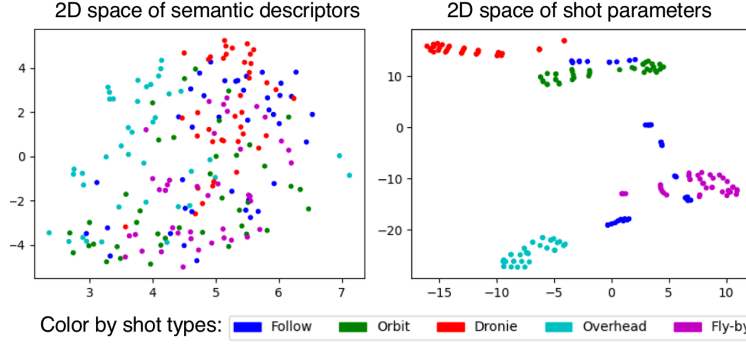


Figure 6.10: Reduction of semantic descriptor space and shot parameter space into 2D compressions using T-SNE. Colors indicate shot types.



Figure 6.11: Examples of shots taken with our autonomous drone setup: 1) Dance moves; 2) Parkour; 3) Running scenes. The full sequences are shown in the supplementary video.

Figure 6.10 provides further insights into the mappings between shot parameters and descriptors. We use T-SNE [159] to visualize a 2D projection of higher-dimensional spaces, and color the data by shot types. Based on the 2D visualization and on the different R^2 score values, we argue that learning a model that maps shot parameters to descriptors is an easier task, as it requires learning a surjective function (many-to-one). The inverse mapping is harder since one point in the descriptor space can correspond to multiple set of shot parameters (one-to-many).

6.6 Conclusion and Discussion

In this paper we present a semantic control space for aerial cinematography that allows users to guide camera motions using intuitive controls instead of selecting low-level shot parameters. We use crowdsourcing with hundreds of users to learn the mapping between shot parameters and semantic descriptors in a diverse dataset of 200 videos. We validate our semantic generative shot model in multiple experiments, and show that it successfully generalizes to new scenes in simulation and in real-world experiments with an aerial robot. Additionally, we provide insights into the space of emotions our model can represent, and investigate the relationship between shot parameters and descriptors to understand what our model is effectively learning.

Our framework targets non-technical users, and can generate shot parameters directly from a semantic vector. However, expert users can easily adapt it to gain more control over the model’s outcome. For example, one can learn separate generative models for individual shot types and gain more control over the system’s inputs / outputs.

We find multiple directions for future work in the area. For instance, we are interested in employing a larger set of parameters to control the shots, such as lens zoom [183], and potentially even soundtracks. In addition, we would like to extend our framework to accept environment features into the generative model. Currently, we also allow the existence of sparse sets of obstacles in the environment which are avoided by the vehicle’s underlying motion planner, but are not explicitly accounted for in the learning framework. Another limitation we face is regarding the shot time horizon: currently our algorithm generates a single shot at a time. However, we find great potential in developing algorithms that can reason over longer durations, and infer emotional expression over sequences of shots. Furthermore, one of the major difficulties faced in this work was building a diverse dataset with good-quality semantic labels. We see great potential in works such as [106], which extract shot parameters directly from pre-existing footage, and we plan to investigate methods for automatic extraction of semantic labels.

Chapter 7

Multi-camera coordination

Professional movie productions require the use of multiple cameras simultaneously to record different viewpoints of the same scene, which are edited into the final footage either in real time or in post-production. Such extreme motion coordination is particularly hard for unscripted action scenes, which are a common use case of aerial cameras.

In this chapter we extend the cinematography theory developed so far into a real-time multi-UAV coordination system that is capable of recording dynamic targets while maximizing shot diversity and avoiding collisions and mutual visibility between cameras. We validate our approach in multiple cluttered environments of a photo-realistic simulator, and deploy the system using two UAVs in real-world experiments. We show that our coordination scheme has low computational cost and takes only 1.17 ms on average to plan for a team of 3 UAVs over a 10 s time horizon. More results can be found on the supplementary video: <https://youtu.be/m2R3anv2ADE>.

7.1 Introduction

Flying cameras are revolutionizing industries that require live and dynamic camera viewpoints such as entertainment, sports, and security. The use of unmanned aerial vehicles (UAVs) has several advantages in comparison with traditional techniques such as dollies and cranes in terms of cost, efficiency and safety [55, 204]. Furthermore, recent works both in industry [61, 211] and academia [20, 98, 105] now allow UAVs to track targets autonomously in cluttered environments.

High-quality cinematographic productions, however, require the composition of multiple viewpoints to form the final footage [5, 26], which can be edited *on-the-fly* in the case of live sports events, or more commonly in post-production. Camera arrangement is a complex problem that requires the production crew to reason about what is going to happen in the scene in terms of actor movements, increase viewpoint diversity, obey cinematographic guidelines (*e.g.* the 180° rule, staying on a single side of the scene), and avoid mutual visibility between cameras. The problem becomes even more challenging in the case of unscripted scenes in sports or journalistic coverage, where cameras need to be re-arranged more frequently while being subjected to velocity and acceleration constraints, and the event cannot be re-enacted in the case of mistakes.

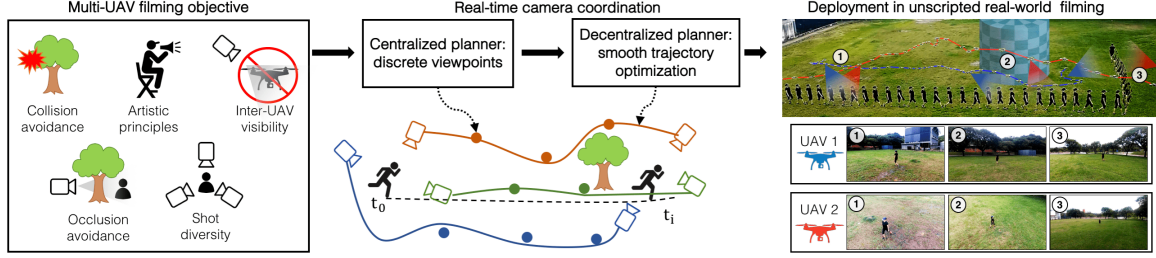


Figure 7.1: Our framework uses a set of objectives for multi-UAV cinematography to coordinate all cameras trajectories with a fast centralized viewpoint planner coupled with a decentralized trajectory optimization algorithm. We deploy the system in real-world settings.

Despite the large recent progress in single-UAV filming solutions, multi-UAV systems are still scarce. Existing approaches focus mainly on coordinating vehicles to follow predefined shot types [172] or pre-planned filming missions [37, 162], restricting the use of this technology to scripted applications. We also find work on maximizing actor coverage [203], partially addressing the shot diversity objective, but this approach is restricted to 2D reasoning and static targets. None of the existing systems can autonomously coordinate multiple cameras in dynamic and unscripted applications.

As seen in Figure 7.1, our work aims to tackle the full multi-UAV coordination problem, targeting cinematography in unscripted and dynamic scenarios. We propose a system that can generate diverse and artistic shots in real-time, empowering operators with full artistic capabilities of aerial cameras. Our main contributions are:

- 1) **Problem formulation:** We formalize the multi-UAV coordination problem in term of its principal cost functions. We start with a single-UAV formulation [21] that considers trajectory smoothness, obstacle avoidance and environmental occlusions, and add new cost terms to maximize shot diversity, avoid mutual visibility between cameras, and to respect high-level user-specified cinematography guidelines;
- 2) **Camera coordination:** We propose a greedy framework for multi-UAV coordination that can run in real-time in dynamic scenes. First, a fast centralized planner computes a set of desired viewpoints for each camera, and a decentralized planning system computes the final trajectory for each UAV based on an occupancy map of the environment and the predicted actor trajectory;
- 3) **Experimental validation:** We validate our approach in multiple environments using a photo-realistic simulator, and deploy the system in real-world experiments with two UAVs.

7.2 Related Work

Single-Drone Cinematography: There is vast body of academic literature and consumer products on the topic of single-drone cinematography. For instance, products such as the DJI Mavic [1] and Skydio R1 [211] can detect and track targets autonomously. In addition, works such as [23, 72, 74, 78, 111, 173, 196] can follow user-specific artistic guidelines during motion. We also find works that try to automate the artistic decision-making guidelines as well. For instance, [21, 84] use deep reinforcement learning to train a deep policy to automatically select the best shot types for a given scene, conditioned on the current actor

actions and obstacle field. Also, [98] uses the actor’s skeleton configuration to automatically position the camera, maximizing the body’s projection on the image.

Multi-Robot Systems: There is a rich selection of work on multi-robot systems, ranging from safety and controls [157, 158], planning [107, 118, 149], target localization [14, 40, 66], exploration [49, 50], robot swarm task planning [38], and even theatrical performances [35, 36]. We also find important theoretical work on multi-sensor coordination using efficient greedy methods [130, 194] that enjoy bounds on sub-optimality when compared to the full exploration of the search space.

Multi-Drone Cinematography: In the field of UAV cinematography we find a few pioneer works that employ multiple vehicles. For instance, [172] proposes an optimization-based algorithm to coordinate multiple UAVs while accounting for inter-drone collisions and mutual visibility, and requires user-defined paths as guidelines for the motion of each drone. The work of [37] takes on additional constraints into a greedy optimization, and maximizes target visibility over time using a team of drones subject to limited battery life. [203] simplifies the actor coverage problem to a 2D space, maximizing target visibility. Also focusing on multi-UAV coverage, [248] optimizes multiple flying cameras trajectories for efficient 3D reconstruction. In the context of filming outdoor events with dynamic targets, [162] provides an overview of cinematography principles that can be used for filming with multiple drones.

7.3 Multi-Camera Coordination

We now detail the methods we use for camera coordination in the multi-UAV system. As displayed in Equation 2.1, our overall objective function involves the minimization of 6 sub-objectives, which often conflict with one another. Our goal is to formulate an algorithm that works in real-time, in unscripted scenes, and that can deal with a state space which grows exponentially in complexity with the number of UAVs. Given this challenge, we prefer to find fast solutions which are only locally optimal as opposed to globally optimal trajectories that take a long time to compute.

To address the time complexity issue, we break down our method into three main subsystems that operate together. First, a centralized motion planner (Sec. 7.3.1) coordinates desired positions for all cameras simultaneously. Next, a decentralized motion planner network (Sec. 7.3.2) computes the final trajectories for each specific UAV. Finally, an image selection module (Sec. 7.3.3) chooses the best live image to be displayed out of all cameras. Alternatively, the final image selection can be manually performed in post-production. Fig. 7.2 depicts the system diagram.

7.3.1 Centralized Multi-Camera Planning

Our centralized planning system parametrizes trajectories as waypoints, i.e., $\xi \in \mathbb{R}^{T \times 3}$, where T is the number of time steps. We assume that the heading dimension $\psi(t)$ is set to always point the drone from $\xi_{qi}(t)$ towards the actor in $\xi_a(t)$, which can be achieved independently of the aircraft’s translation by rotating the UAV’s body and camera gimbal.

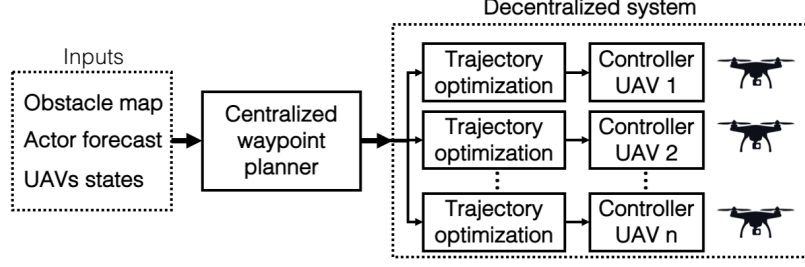


Figure 7.2: System overview: a centralized waypoint planner computes discrete camera positions. Next, a decentralized planning system optimizes smooth trajectories for each UAV.

We parametrize the state-space of all possible camera positions using spherical coordinates $\{\rho, \theta, \phi\}$ centered on the actor's location:

$$\xi_{qi}(t) = \xi_a(t) + \rho \begin{bmatrix} \cos(\psi_a + \theta) \sin(\phi) \\ \sin(\psi_a + \theta) \cos(\phi) \\ \cos(\phi) \end{bmatrix} \quad (7.1)$$

Space discretization: We define a discrete state-space lattice S that contains all possible camera positions distributed as $|S| = 576$ points in a half-sphere above ground. Based on cinematographic guidelines [5, 26] we equally divide the yaw coordinates θ into 16 values between $[0, 2\pi]$, the tilt angles ϕ into 6 values between $[0, \pi/2]$, and the distance to actor within the set $\rho \subseteq \{2, 3, 4, 5, 6, 7\}$, ranging from close-up to long shots. In addition, we discretize the trajectory's time into 5 steps equally spaced every 2 seconds, forming a 10-second planning time horizon.

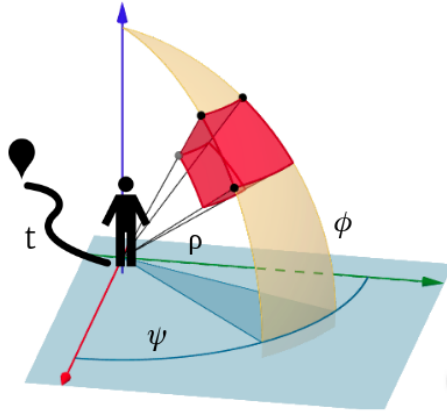


Figure 7.3: Discrete spherical state space centered on actor.

Cost functions: Next we mathematically formulate the cost functions that optimized by the centralized camera planner for the set of UAV trajectories Ξ :

i) Shot diversity cost: The shot diversity cost prevents the planner to allocate the n camera positions close to each other up until a maximum distance d_{max}^{div} :

$$J_{\text{div}}(\Xi) = \sum_{\tau=1}^T \sum_{i=1}^n \sum_{j=1}^n D(\xi_{qi}(\tau), \xi_{qj}(\tau)),$$

$$\text{where } D(p_1, p_2) = \begin{cases} 1 & \text{if } d < d_{\min}^{\text{div}} \\ \frac{d}{d_{\max}^{\text{div}} - d_{\min}^{\text{div}}} & \text{if } d_{\min}^{\text{div}} < d < d_{\max}^{\text{div}} \\ 0 & \text{if } d > d_{\max}^{\text{div}} \end{cases} \quad (7.2)$$

ii) *Inter-drone collision cost:* Analogously to J_{div} in Eq. 7.2, we define the inter-drone collision cost as a measure of distance between UAVs. The user can, however, choose a different cost weight λ and safety distance d_{\max}^{col} to penalize collisions.

iii) *Inter-drone visibility:* We express the inter-drone visibility cost as a binary measure $V(p_i, p_j) \subseteq \{0, 1\}$ of whether point j is visible from position i . We model the visible area as a cone in space using the camera's diagonal field-of-view angle, and position its center line towards the actor's position:

$$J_{\text{vis}}(\Xi) = \sum_{\tau=1}^T \sum_{i=1}^n \sum_{j=1}^n V(\xi_{qi}(\tau), \xi_{qj}(\tau)) \quad (7.3)$$

We pre-compute all mutual visibility values into a look-up table to reduce planning times, which is possible in our discrete state-space model.

iv) *Obstacle and occlusion avoidance:* In order to keep all UAVs safe and to maintain visibility of the actor at all times, we must reason about the role of obstacles in the environment. First, we transform the environment's occupancy grid into a time-dependent spherical domain centered around the actor $\mathcal{G} \rightarrow \mathcal{G}_s^t \in [0, 1]$, as shown in Fig. 7.4a. We then compute the obstacle avoidance cost by summing the occupancy likelihood of all cells within a radius r_{\max} of each UAV. We also calculate the occlusion cost as a measure of occupancy along a line $l_i(\tau) = \tau \xi_{qi}(t) + (1 - \tau) \xi_a(t)$ between UAV and actor:

$$J_{\text{obs}}(\Xi) = \sum_{\tau=1}^T \sum_{i=1}^n \int_0^{r_{\max}} \mathcal{G}_s^\tau(\xi_{qi}(\tau)) d(\text{volume})$$

$$J_{\text{occ}}(\Xi) = \sum_{\tau=1}^T \sum_{i=1}^n \int_0^1 \mathcal{G}_s^\tau(l_i(\tau)) d\tau \quad (7.4)$$

v) *Cinematography guidelines as cost prior:* We also allow operators to manually specify a cost prior J_{cine} for each cell, in case they wish to follow specific cinematographic guidelines. For example, Fig. 7.4b shows an example where we define overhead shots as undesired.

Greedy optimization: The centralized planning problem for multi-camera optimization over multiple time steps proves to be NP-hard, similarly to other optimal sensor placement works [7, 129, 131]. In order to make the computation tractable in real-time and avoid a combinatorial problem, we develop a greedy optimization approach that computes the optimal trajectory ξ_{qi}^{greedy} for each UAV sequentially, fixing all previously calculated trajectories $\{\xi_{q1}^{\text{greedy}}, \dots, \xi_{qi-1}^{\text{greedy}}\}$.

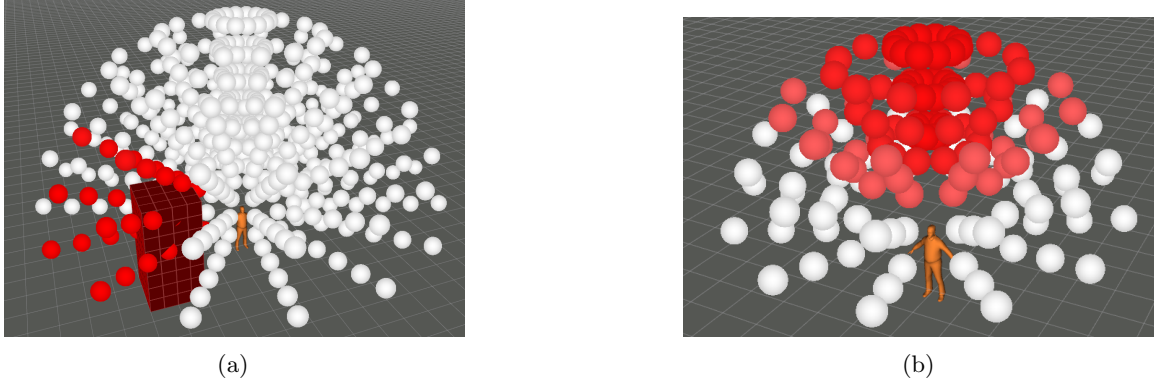


Figure 7.4: (a) Visualization of occupancy and occlusion avoidance costs in the cells of the spherical grid \mathcal{G}_s^t . (b) Example of a cinematography guideline prior that penalizes overhead shots. Red spheres represent regions of high cost.

At each stage i we employ a backward induction dynamic programming algorithm [15, 140, 223] to find the optimal cost-to-go for each state $s \subseteq S$ at all time steps, analogously to a value iteration algorithm. To do so, we build a cost map $C : S \rightarrow \mathbb{R}^{|S|}$ that contains the cost of all states, and a cost-to-go map $V : S \rightarrow \mathbb{R}^{|S|}$. In order to make transitions between cells dynamically feasible for the real vehicle, we only allow expansions to neighboring cells in the spherical grid. Since we operate in a discrete state-space with a small branching factor and deterministic transitions, a single backwards pass quickly yields the optimal solution. Finally, we build the full trajectory $\xi_{qi}^{*greedy}$ by selecting neighboring cells with the least cost-to-go at consecutive time steps, starting at the UAV's initial position S_i^0 . We update the cost manifold after each drone is added, since inter-visibility, shot diversity, and collision costs are recalculated. Algorithm 3 details the process:

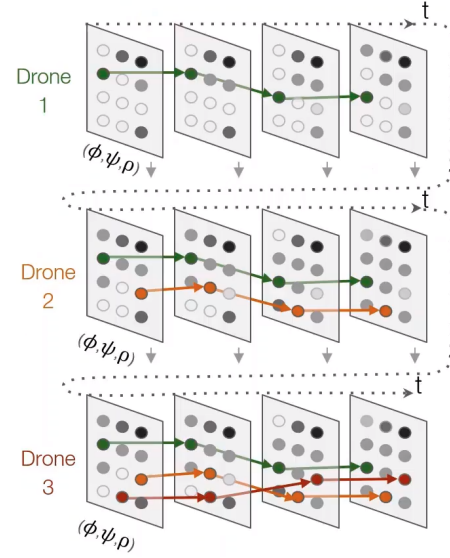


Figure 7.5: Greedy planner.

Algorithm 3: Compute greedy traj. set $\Xi = \{\xi_{q1}, \dots, \xi_{qn}\}$

```

1 Initialize  $\Xi_{greedy} = \{\}$   $\triangleright$  empty set
2 for each UAV  $i$  do
3    $C \leftarrow J(S, \Xi_{greedy})$   $\triangleright$  update entire cost map
4    $V \leftarrow \text{BackwardInduction}(C)$   $\triangleright$  update cost-to-go
5    $\xi_{qi}^{*greedy} \leftarrow \text{OptimalPath}(V, S_i^0)$ ;
6   Append  $\xi_{qi}^{*greedy}$  to  $\Xi_{greedy}$ ;
7 end
8 return  $\Xi_{greedy}$ 

```

7.3.2 Decentralized UAV Trajectory Optimization and Tracking

After calculating the greedy-optimal set of UAV trajectories Ξ_{greedy} using the centralized planner, we post-process each output using a decentralized planning system. Our objective here is to obtain smoother individual trajectories at a finer time discretization. While the original waypoints were spaced every 2 seconds over a 10-second horizon, here we achieve finer resolutions with 0.5 s granularity in local planning, and 0.02 s for trajectory tracking. We employ the single-UAV local planner described in [21], which uses covariant gradient descent to produce locally optimal trajectories while considering the costs of smoothness, obstacle and occlusions avoidance, and a desired artistic trajectory, which we define as $\xi_{q_i}^{\text{greedy}}$. In addition, each local planner receives the expected waypoints of all remaining vehicles, and avoids positioning its trajectory within 1m of other UAVs. We run the local planner at 5 Hz, and use a PID controller at 50 Hz.

7.3.3 Live Image Selection

Even though our system produces multiple image streams, most filming applications display a single camera to the viewer at a time. The final movie can be produced with manual post-processing, or in the case of live streams it requires a human to make selections while viewing all videos simultaneously. Here, we propose a method for automatic selection of live images based on a subset of our cost functions. We compute a image quality score Q_i for each camera stream, calculated as a weighted sum of the inter-drone visibility cost J_{vis} and of the prior cinematographic guidelines J_{cine} . We select the current camera with the highest score, and once a camera is chosen we exponentially decay its score to foster viewpoint diversity, and gradually return it to the original value once another image is chosen. Based on cinematography literature we set minimum and maximum limits for shot lengths of 3 and 8 seconds respectively, which are reasonable units of length for individual action shots [53, 163].

7.4 Experimental Results

Here we detail the simulated and real-world experiments that validate our multi-UAV cinematography system. Additional visualizations are shown in the supplementary video.

7.4.1 Simulation experiments

Experimental setup: We record all simulated data using a drone in a photo-realistic environment, AirSim [208], coupled with a custom ROS interface [198]. As seen in the supplementary video, our simulated scenes consists of an animated character walking around a suburban environment, surrounded by obstacles such as trees, buildings, and posts.

E1) Viewpoint diversity validation: This first experiment’s objective was to validate the assumption of the relation between shot diversity and artistic quality by quantifying the benefits of the employing multiple UAVs for aerial cinematography as opposed to the use of a single camera. To do so, we generated a set of 5 videos with 12 seconds of length of an actor walking around a park. The first clip featured a single-camera back shot for its entire duration, while all other clips alternated between the back shot and a secondary viewpoint every 3 seconds. We chose the secondary viewpoints to be either 45°, 90°, 135°, or 180° for each of the other four videos respectively.

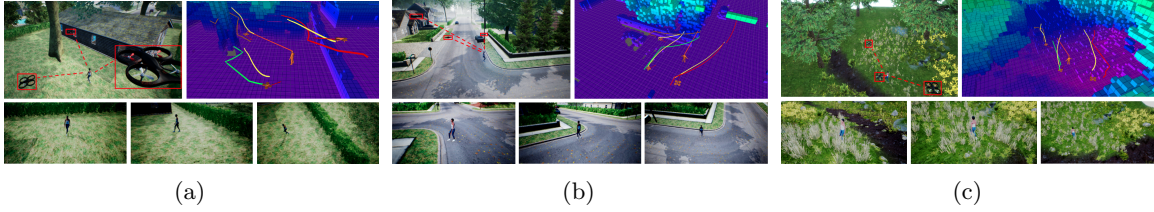


Figure 7.6: Diverse range of simulated environments where we tested our multi-UAV system: a) narrow gaps, b) suburban environment with sparse tall obstacles, and c) dense foliage.

For this survey we recruited 15 participants using Amazon Mechanical Turk (MTurk) [3], who were compensated for their time. After being approved on a short qualifying task, each participant viewed a total of 12 pairs of videos, one being the constant back shot clip, and the other being one of the multi-viewpoint clips. Videos were played synchronously three times, and after watching at least once participants answered: “Which video is more enjoyable?” Each clip was compared 15 times against the baseline single-camera shot.

Figure 7.7 displays the survey results, where the height of each bar represents the percentage of users that rated the multi-UAV clip as being more enjoyable than the single-UAV clip. As expected, we found that an overwhelming majority of users prefer to watch a scene with viewpoint diversity.

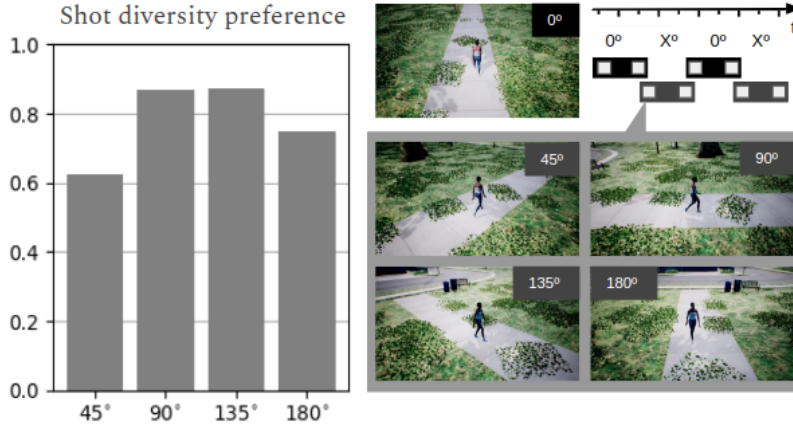


Figure 7.7: User study to evaluate viewer preference for videos with diverse viewpoints. We ask users to compare how enjoyable a single back shot view is against clips with additional viewpoints. Vertical bar shows percentage of users that prefer the video with a secondary viewpoint, where each bar corresponds to an angle value.

E2) Robustness across environments: We tested our approach in a diverse set of photo-realistic simulations, ranging from open to cluttered environments, and with groups of UAVs containing 2 to 5 vehicles. As shown in Figure 7.6, our approach is able to successfully coordinate the teams of UAVs in a wide spectrum of scenarios. For instance, in 7.6a the actor passes through a narrow gap (2m) that the UAVs need to avoid, and in 7.6b she walks next to an extensive line of trees (1m apart), which forces the UAVs to move to the right side of the actor. In addition, in 7.6c we see the UAVs navigating within dense and unstructured foliage tunnels.

The tests indicate that our system is able to adapt to a diverse range of environments and actor motions. Furthermore, we verify that the final clips follow our objective functions and present diverse viewpoints with little inter-drone visibility.

E3) Scaling performance: We conducted a set of experiments to quantify our centralized planner’s performance as it scales to larger teams of UAVs and larger state-spaces. This analysis is important because keeping low computational costs is vital for real-time performance.

First, we measured the planning time and computer resources consumption for finer discretizations of the state space (Table 7.1). We note that memory usage grows proportionally to the square of the number of cells in the state space because we employ a precomputed lookup table for the inter-drone visibility and shot diversity costs. However, our implementation offers large benefits in terms of computing time and CPU usage, which grow linearly.

State space (n_ψ, n_ϕ, n_ρ)	Computed states	Planning time for 3 drones [ms]	CPU[%] (1 core)	Memory use [MB]
(3,3,8)	360	0.17+-0.03	22	35
(16,6,6)	2880	1.17+-0.18	25	37.5
(24,9,9)	9720	4.65+-0.47	28	64
(32,12,12)	23040	16.70+-1.26	34	197
(40,15,15)	45000	19.56+-0.83	38	655
(48,18,18)	77760	51.71+-2.04	52	1840
(52,21,21)	114660	116.28+-6.98	75	4693
(64,24,24)	184320	228.01+-8.85	100	10150

Table 7.1: Performance of the greedy planner for different discretizations of the State space. We use 3 UAVs with a 5 time-steps horizon, and re-plan at 5 Hz.

The discretization we use for most of our experiments ($n_\psi=16, n_\phi=6$ and $n_\rho=6$) presents a great trade-off, with a reasonable space discretization and a low computation time of 1.17ms, while only consuming 37.5 MB of RAM. As a comparison, we implemented an optimal brute-force planner and tested it using the same spatial discretization and number of drones. Using only 2 time steps, the non-greedy planner took 16.4 seconds to find the optimal solution. The planning time jumps to 2h:30min when we consider 3 time steps. These results show the importance of adopting a greedy strategy to solve a NP-hard problem in real-time.

We also evaluated how the planning time increase with a larger number of drones and time steps. As expected, our greedy solution has a linear growth of complexity (Fig. 7.8).

7.4.2 Real-world results

Experimental setup: For the real-world experiments we used two unmodified Parrot Bebop 2 UAVs with an integrated electronic gimbal, a WiFi router and a standard desktop PC (Intel i7-8700 CPU and Nvidia GTX1060 GPU). All communication between the drones and PC were handled via a ROS interface with the Bebop SKD. We transmitted images from the UAVs to the base station for the actor’s visual detection and trajectory forecasting, and sent velocity commands back to the UAV after planning.

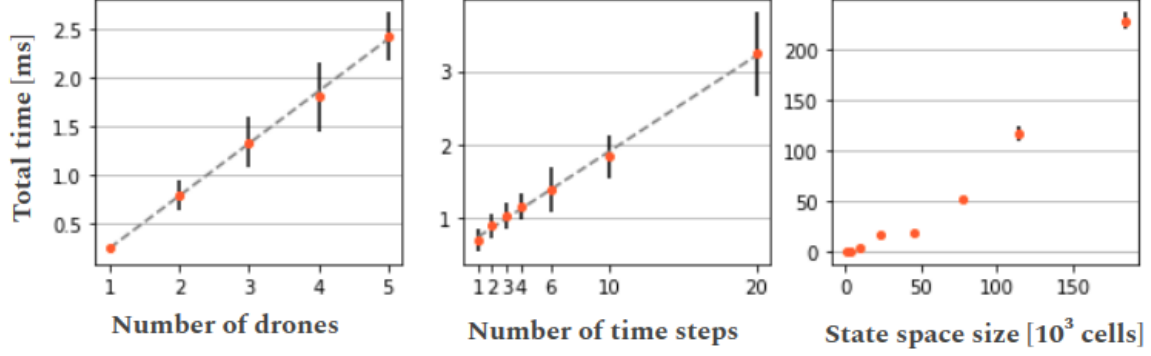


Figure 7.8: Time required by the high-level planner to compute the trajectories of the drones: (a) given the number of vehicles, (b) given the number of time steps computed and (c) given the number of states in the artistic domain ($n_\psi \cdot n_\phi \cdot n_\rho \cdot n_t$)

E4) Flight among obstacle: We tested the system by recording a moving actor with 2 drones in an environment with a virtual obstacle. Figure 7.9 depicts how the high-level central planner interacts with the local trajectory optimization method to calculate the final UAV paths. *Drone 1* is optimized first, and takes a trajectory close to, but sufficiently far from the obstacle. *Drone 2* is optimized next by the central planner. When it reaches the vicinity of the virtual obstacle, the planner brings its ideal path to a higher elevation, above the first drone’s path, in order to avoid colliding with the obstacle while keeping the actor visible and avoiding visibility of *drone 1*. The central planner’s output for drones 1 and 2 are colored in blue and red respectively. The UAV’s respective local planners receive the discrete high-level path and optimize it, creating a smooth trajectory output. After *drone 2* reaches the left side of the actor, it switches to a slightly tilted and distant shot due to the shot diversity cost.

E5) Dynamic actor: We evaluated the system’s capability to track and record an actor performing dynamic movements with abrupt motion changes, such as a person playing soccer. As seen in the supplementary video, both drones were able to safely keep the actor in frame through the entire test while exploring diverse viewpoints with low inter-drone visibility.

7.5 Conclusion and Discussion

In this paper we present a system for real-time coordination of aerial cameras for autonomous cinematography in dynamic and unscripted scenarios. First, we formalize the multi-UAV filming problem in terms of its main objectives: maximizing shot diversity, avoiding inter-vehicle visibility and obeying high-level cinematographic guidelines. Next, we develop a two-step approach for calculating the trajectory of each UAV, based on an efficient centralized greedy planner for viewpoint selection coupled with a decentralized trajectory optimizer to calculate smooth trajectories. We validate our system in multiple simulated and real-world experiments, and show that it can successfully control a team of UAVs. Additionally, we provide insights into how our methods can scale for larger numbers of UAVs in terms of planning time and memory.

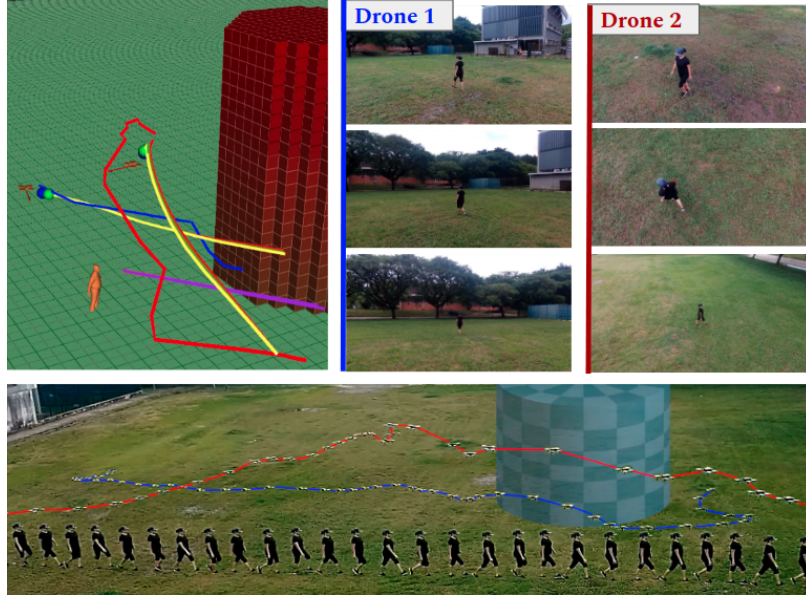


Figure 7.9: Real-life flight among a simulated obstacle. On the left, the outputted sequence waypoints for each drone (red and blue) guide the locally optimized paths (yellow) towards a region free from obstacles and free of inter-drone visibility. Diverse shots can be seen during flight.



Figure 7.10: Real-life flight following a highly dynamic actor playing soccer. The drones are able to keep up with abrupt motion motion changes while filming the actor.

We note that the nature of trajectories generated with our system is highly dependent on the combination of relative weights between costs. Different applications may require distinct weighing schemes. For instance, for a journalistic coverage the operator would likely increase the penalty on inter-drone visibility, while this weight could be negligible for footage generated in a 3D motion reconstruction application.

We are interested in extending this research in multiple future directions. Despite using a decentralized trajectory optimization framework, our experiments in this paper were performed on a single desktop PC, and commands were then transmitted to each UAV individually in real time. We are currently working on a new multi-UAV system with onboard processors, where one master drone will run the fast centralized planner and then communicate with all other UAVs via a mesh network so that each drone can optimize its

own trajectories. In addition, we plan to employ our system in novel applications such as 3D reconstruction of scenes in the wild. Even though multi-view systems are already well developed for indoors environments [108], creating such systems for capturing images in natural environments is still an open research field.

Chapter 8

Learning state representations for aerial navigation

The chapters presented thus far focused on motion planning and learning techniques for the application of aerial cinematography. In this chapter we develop theories for general aerial navigation, within the context of drone racing. We focus on learning state representations that can be easily transferred from simulation domains to real-world tasks.

The main motivation behind this work is that despite enormous advancements in the recent years, machines are still a long way from robustly solving open-world perception-control tasks, such as first-person view (FPV) aerial navigation. While recent works in end-to-end Machine Learning, especially Imitation Learning and Reinforcement appear promising, they are constrained by the need of large amounts of difficult-to-collect labeled real-world data. Simulated data, on the other hand, is easy to generate, but generally does not render safe behaviors in diverse real-life scenarios.

In this chapter we propose a novel method for learning robust visuomotor policies for real-world deployment which can be trained purely with simulated data. We develop rich state representations that combine supervised and unsupervised environment data. Our approach takes a cross-modal perspective, where separate modalities correspond to the raw camera data and the system states relevant to the task, such as the relative pose of gates to the drone in the case of drone racing. We feed both data modalities into a novel factored architecture, which learns a joint low-dimensional embedding via Variational Auto Encoders. This compact representation is then fed into a control policy, which we trained using imitation learning with expert trajectories in a simulator.

We analyze the rich latent spaces learned with our proposed representations, and show that the use of our cross-modal architecture significantly improves control policy performance as compared to end-to-end learning or purely unsupervised feature extractors. We also present real-world results for drone navigation through gates in different track configurations and environmental conditions. Our proposed method, which runs fully onboard, can successfully generalize the learned representations and policies across simulation and reality, significantly outperforming baseline approaches. The supplementary video shows examples of policy deployment: <https://youtu.be/VKc3A5H1UU8>, and open-sourced is at: <https://github.com/microsoft/AirSim-Drone-Racing-VAE-Imitation>.

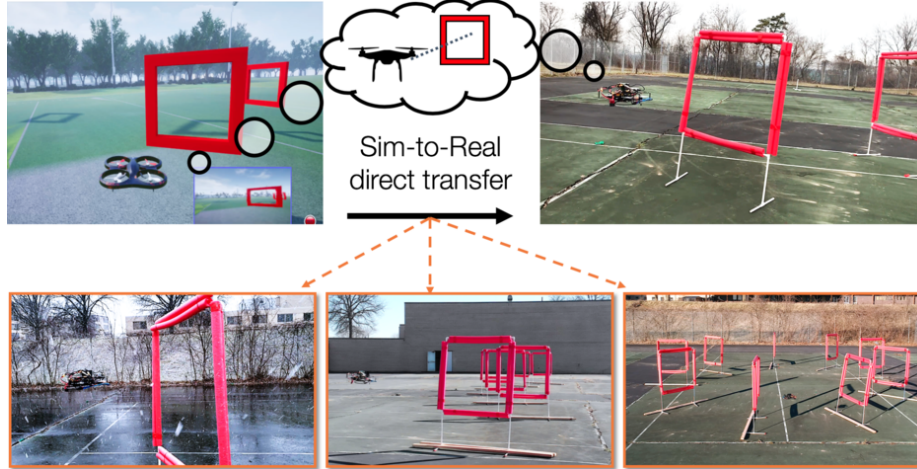


Figure 8.1: The proposed framework uses simulations to learn a rich low-dimensional state representation using multiple data modalities. This latent vector is used to learn a control policy which directly transfers to real-world environments. We successfully deploy the system under various track shapes and weather conditions, ranging from sunny days to strong snow and wind.

8.1 Introduction

Aerial navigation of drones using first-person view (FPV) images is an exemplary feat of the human mind. Expert pilots are able to plan and control a quadrotor with high agility using a potentially noisy monocular camera feed, without comprising safety. We are interested in exploring the question of what would it take to build autonomous systems that achieve similar performance levels.

One of the biggest challenges in the navigation task is the high dimensional nature and drastic variability of the input image data. Successfully solving the task requires a representation that is invariant to visual appearance and robust to the differences between simulation and reality. Collecting labeled real-world data to minimize the sim-to-real gap, albeit possible, requires intensive effort and specialized equipment for gathering ground-truth labels [120,121]. Attempting to solve the task solely with real-world data is also challenging due to poor sample efficiency of end-to-end methods, and often leads to policies that are unable to deal with large perceptual variability [116,225,249]. Additionally, end-to-end training in the real world is expensive and dangerous, especially in early phases of training when policies are highly prone to errors and collisions.

Sim-to-real transfer learning methods aim to partially alleviate these challenges by training policies in a synthetic environment and then deploying them in the real-world [75,202,229]. Domain randomization uses a large collection of object appearances and shapes, assuming that any real-life features encountered will be represented within a subset of the database. Simulations can be used to generate large amounts of synthetic data under a wide variety of conditions [154,240].

In this work, we introduce cross-modal learning for generating representations which are robust to the simulation-reality gap, and do not overfit to specificities of the simulated data. In particular, the need to explain multiple modes of information during the training

phase aids in implicit regularization of the representation, leading to an effective transfer of models trained in simulation into the real world. We use high-fidelity simulators [207] to generate both labeled and unlabeled modalities of simulated data. Furthermore, our proposed framework can also use unlabeled real-world data in the training phase, thereby allowing us to incorporate real-life unlabeled traces into the state representation. Fig. 8.1 depicts the overall concept, showing a single perception module shared for simulated and real autonomous navigation. Our unmanned aerial vehicle (UAV) uses FPV images to extract a low-dimensional representation, which is then used as the input to a control policy network to determine the next control actions. We successfully test the system operating in challenging conditions, many of which were previously unseen by the simulator, such as snow and strong winds.

Our proposed framework entails learning a cross-modal representation for state encoding. The first data modality considers the raw unlabeled sensor input (FPV images), while the second directly characterizes state information directly relevant for the desired application. In the case of drone racing, our labels correspond to the relative pose of the next gate defined in the drone’s frame. We learn a low-dimensional latent representation by extending the Cross-Modal Variational Auto Encoder (CM-VAE) framework from [213], which uses an encoder-decoder pair for each data modality, while constricting all inputs and outputs to and from a single latent space. Consequently, we can naturally incorporate both labeled and unlabeled data modalities into the training process of the latent variable. We then use imitation learning to train a control policy that maps latent variables into velocity commands for the UAV. The representation uses only raw images during deployment, without access to the ground-truth gate poses. While closest to our work is the recent line of research on autonomous drone racing [121, 154], we would like to note that our objective here is not to engineer a system that necessarily finishes the race the fastest. Unlike the prior work, we do not assume prior knowledge during deployment in terms of an accurate dynamics model coupled with optimal trajectory generation methods. Instead, our goal is to learn visuomotor policies operating on learned representations that can be transferred from simulation to reality. Thus, the methods presented in this paper are not directly comparable to [121, 154].

While in this paper we specifically focus on the problem of aerial navigation in a drone racing setting, the proposed techniques are general and can be applied to other perception-control tasks in robotics. Our key contributions are:

- We present a cross-modal framework for learning latent state representations for navigation policies that use unsupervised and supervised data, and interpret the bi-modal properties of the latent space encoding;
- We provide simulation experiments comparing variants of the cross-modal framework with baseline feature extractors such as variational auto-encoders (VAEs), gate pose regression, and end-to-end learning;
- We provide multiple real-world navigation experiments and performance evaluations of our control policies. We show that our proposed representation allows for sim-to-real deployment of models learned purely in simulation, achieving over one kilometer of cumulative autonomous flight through obstacles.

8.2 Related Work

Navigation policies Classically, navigation policies rely on state estimation modules that use either visual-inertial based odometry [57] or simultaneous localization and mapping [220]. These techniques can present high drift and noise in typical field conditions, impacting the quality of both the robot localization and the map representation used for planning. Therefore, trajectory optimization based algorithms [17, 165, 178] can result in crashes and unsafe robot behaviors. Against these effects, [199] learn a collision avoidance policy in dense forests using only monocular cameras, and [155] learn a steering function for aerial vehicles in unstructured urban environments using driving datasets for supervision.

Recently, [87, 88, 116, 143] explore learning separate networks for the environment representation and controls, instead of the end-to-end paradigm. The goal of an intermediate representations is to extract a low-dimensional space which summarizes the key geometrical properties of the environment, while being invariant to textures, shapes, visual artifacts. Such intermediate representations mean that behavior cloning or reinforcement learning methods have a smaller search space [143] and more sample efficiency.

Learning representations for vision Variational Autoencoder (VAE) based approaches have been shown to be effective in extracting low-dimensional representation from image data [30, 60, 94, 123]. Recently, VAEs have been leveraged to extract representations from multiple modalities [10, 147, 176, 213]. Relevant to our work, [213] propose a cross-modal VAE to learn a latent space that jointly encodes different data modalities (images and 3D keypoints associated with hand joints) for a image to hand pose estimation problem.

Drone Racing We find different problem definitions in the context of autonomous drone racing. [121, 154] focus on scenarios with dynamic gates by decoupling perception and control. They learn to regress to a desired position using monocular images with a CNN, and plan and track a minimum jerk trajectory using classical methods [165, 192]. [154] utilize domain randomization for effective simulation to reality transfer of learned policies.

Gate poses are assumed as *a priori* unknown in [113–115]. [113] use depth information and a guidance control law for navigation. [114, 115] use a neural network for gate detection on the image. A limitation of the guidance law approach is that the gate must be in view at all times, and it does not take into account gate relative angles during the approach.

[120] formulate drone racing as flight through a predefined ordered set of gates. They initialize gate locations with a strong prior via manual demonstration flights. The belief over each gate location is then updated online by using a Kalman Filter over the gate pose predictions from a neural network.

In our work we take inspirations from the fields of policy and representation learning to present a method that combines unsupervised and supervised simulated data to train a single latent space on which a control policy acts. The bi-modal nature of the latent space implicitly regularizes the representation model, allowing for policy generalization across the simulation-reality gap.

8.3 Approach

This work addresses the problem of robust autonomous navigation through a set of gates with unknown locations. Our approach is composed of two steps: first, learning a latent state representation, and second, learning a control policy operating on this latent representation (Fig. 8.2). The first component receives monocular camera images as input and encodes the relative pose of the next visible gate along with background features into a low-dimensional latent representation. This latent representation is then fed into a control network, which outputs a velocity command, later translated into actuator commands by the UAV’s flight controller.

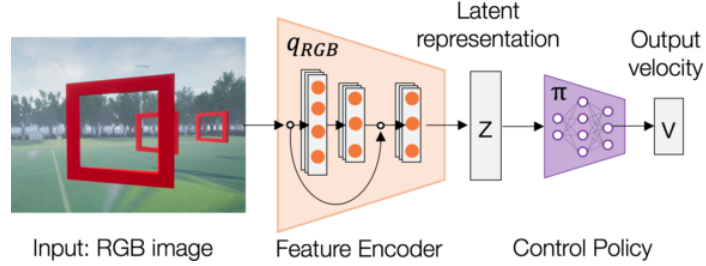


Figure 8.2: Control system architecture. The input image is encoded into a latent representation of the environment. A control policy acts on the lower-dimensional embedding to output the desired robot velocity commands.

8.3.1 Definitions and Notations

Let W define the world frame, B the body frame, and G_i the frame of the target gate. Let E define the full environment geometry and object categories. Assuming that all gates are upright, let $y_i = [r, \theta, \phi, \psi]$ define the relative spherical coordinates and yaw gate frame G_i , in the B frame.

We define $q_{RGB}(I_t) \rightarrow \mathbb{R}^N$ to be an encoder function that maps the current image I_t to a latent compressed vector z_t of size N . Let $\pi(z_t) \rightarrow \mathbb{R}^4$ be a control policy that maps the current encoded state to a body velocity command $v_B = [v_x, v_y, v_z, v_\psi]$, corresponding to linear and yaw velocities.

Let π^* be an expert control policy. Our objective is to find the optimal model parameters Θ^* and Φ^* that minimize the expectation of distance D between our control policy and the expert, taken over observed states s . Note that the expert policy operates with full knowledge of the environment E , while our policy only has access to the observation $q_{RGB}(I_t)$:

$$\Theta^*, \Phi^* = \arg \min_{\Theta, \Phi} \mathbb{E}_s \left[D \left(\pi^*(E), \pi^\Phi(q_{RGB}^\Theta(I)) \right) \right] \quad (8.1)$$

8.3.2 Learning Cross-Modal Representations for Perception

The goal of the perception module is to extract all pertinent information for the current task from the current state of the environment E and UAV. Several approaches exist for feature extraction, from fully supervised to fully unsupervised methods, as mentioned in Section 8.2.

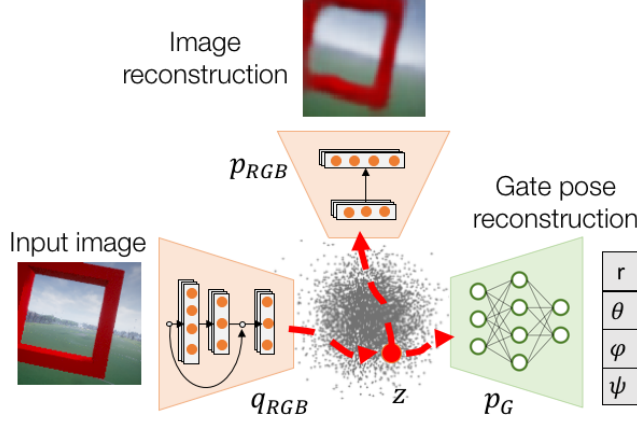


Figure 8.3: Cross-modal VAE architecture. Each data sample is encoded into a single latent space that can be decoded back into images, or transformed into another data modality such as the poses of gates relative to the UAV.

An effective dimensionality reduction technique should be smooth, continuous and consistent [213], and in our application, also be robust to differences in visual information across both simulated and real images. To achieve these objectives we build on the architecture developed by [213], who use a cross-modal variant of variational auto-encoders (CM-VAE) to train a single latent space using multiple sources of data representation. The core hypothesis behind our choice of encoding architecture is that, by combining different data modalities into one latent vector, we can induce a regularization effect to prevents overfitting to one particular data distribution. As shown in Section 8.4.3, this becomes an important feature when we transfer a model learned with simulated data to the real world, where appearances, textures and shapes can vary significantly.

CM-VAE derivation and architecture: The cross-modal architecture works by processing a data sample x , which can come from different modalities, into the same latent space location (Fig. 8.3). In robotics, common data modalities found are RGB or depth images, LiDAR or stereo pointclouds, or 3D poses of objects in the environment. In the context of drone racing, we define data modalities as RGB images and the relative pose of the next gate to the current aircraft frame, *i.e.*, $x_{RGB} = I_t$ and $x_G = y_i = [r, \theta, \phi, \psi]$. The input RGB data is processed by encoder q_{RGB} into a normal distribution $\mathcal{N}(\mu_t, \sigma_t^2)$ from which z_t is sampled. Either data modality can be recovered from the latent space using decoders p_{RGB} and p_G .

In the standard definition of VAEs, the objective is to optimize the variational lower bound on the log-likelihood of the data [60, 191]. In [213], this loss is re-derived to account for probabilities across data modalities x_i and x_j , resulting in the new lower bound shown in Eq. 8.2:

$$\mathbb{E}_{z \sim q(z|x_i)} [\log p(x_j|z)] - D_{KL}(q(z|x_i)||p(z)) \quad (8.2)$$

We use the Dronet [155] architecture for encoder p_{RGB} , which is equivalent to an 8-layer Resnet [90]. We choose a small network, with about 300K parameters, for its low onboard inference time. For the image decoder q_{RGB} we use six transpose convolutional layers, and for the gate decoder p_G we use two dense layers.

Training procedure: We follow the training procedure outlined in Algorithm 1 of [213], considering three losses: (i) MSE loss between actual and reconstructed images (I_t, \hat{I}_t), (ii) MSE loss for gate pose reconstruction (y_i, \hat{y}_i), and (iii) Kullback-Leibler (KL) divergence loss for each sample. During training, for each unsupervised data sample we update networks q_{RGB} , p_{RGB} , and for each supervised sample we update both image encoder q_{RGB} and gate pose decoder p_G with the gradients.

Imposing constraints on the latent space: Following recent work in disentangled representations [94, 123], we compare two architectures for the latent space structure. Our goal is to improve performance of the control policy and interpretability of results. In first architecture, z_{unc} stands for the unconstrained version of the latent space, where: $\hat{y}_i = p_G(z_{unc})$ and $\hat{I}_t = p_{RGB}(z_{unc})$. For second architecture, instead of a single gate pose decoder p_G , we employ 4 independent decoders for each gate pose component, using the first 4 elements of z_{con} . As human designers, we know that these features are independent (*e.g.*, the distance between gate and drone should have no effect on the gate’s orientation). Therefore, we apply the following constraints to z_{con} : $\hat{r} = p_r(z_{con}^{[0]})$, $\hat{\theta} = p_\theta(z_{con}^{[1]})$, $\hat{\psi} = p_\psi(z_{con}^{[2]})$, $\hat{\phi} = p_\phi(z_{con}^{[3]})$. The image reconstruction step still relies on the full latent variable: $\hat{I}_t = p_{RGB}(z_{con})$.

8.3.3 Imitation learning for control policy

Expert trajectory planner and tracker: To generate expert data ($\pi^*(E)$) we use a minimum jerk trajectory planner following the work of [31, 165, 192] considering a horizon of one gate into the future, and track it using a pure-pursuit path tracking controller. We generate a dataset of monocular RGB images with their corresponding controller velocity commands.

Imitation learning algorithm: We use behavior cloning (BC), a variant of supervised learning [180], to train the control policy $\pi(q(I))$ when minimizing Equation 8.1. We freeze all encoder weights when training the control policy. In total we train 5 policies for the simulation experiments: BC_{con} and BC_{unc} , which operate on z_{con} and z_{unc} respectively as features, BC_{img} , which uses a pure unsupervised image reconstruction VAE for features, BC_{reg} , which uses a purely supervised regressor from image to gate pose as features, and finally BC_{full} , which uses a full end-to-end mapping from images to velocities, without an explicit latent feature vector. We train an additional policy BC_{real} for the physical experiments, using unsupervised real-life images along the CM-VAE architecture, as further detailed in Section 8.4.3.

To provide a fair comparison between policy classes, we design all architectures to have the same size and structure. BC policies learned on top of representations are quite small, with only 3 dense layers and roughly 6K neurons. The end-to-end BC_{full} layout is equivalent to the combination of the Dronet encoder plus BC policy from the other cases, but initially all parameters are untrained.

8.4 Results

8.4.1 Learning Representations

Our first set of experiments aims to valuate the latent representations from three different architectures: (i) q_{reg} , for direct regression from $I_t \rightarrow z_{reg} = [r, \theta, \phi, \psi] \in \mathbb{R}^4$, (ii) q_{unc} , for

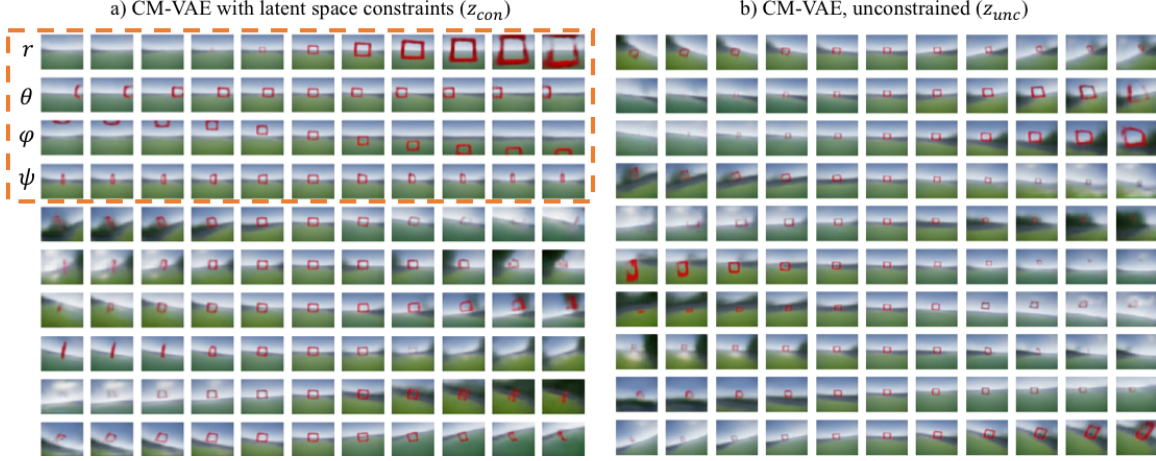


Figure 8.4: Visualization of latent space from a) constrained and b) unconstrained cross-modal representations. The constraints on latent space force the disentanglement of the first four variables of z_{con} to encode the relative gate pose, condition that is also observed in the image modality.

the CM-VAE using RGB and pose modalities without constraints: $I_t \rightarrow z_{unc} \in \mathbb{R}^{10}$, and (iii) q_{con} , for the CM-VAE using RGB and pose modalities with constraints: $I_t \rightarrow z_{con} \in \mathbb{R}^{10}$.

We generated 300K pairs of 64×64 images along with their corresponding ground-truth relative gate poses using the AirSim simulator [207]. We randomly sample the distance to gate, aircraft orientation, and gate yaw. 80% of the data was used to train the network, and 20% was used for validation.

Fig. 8.4 displays images decoded over various regions of the latent spaces z_{con} and z_{unc} . Each row corresponds to variations in z values in one of the 10 latent dimensions. We verify that the latent variables can encode relevant information about the gate poses and background information. In addition, the constrained architecture indeed learned to associate the first four dimensions of z_{con} to affect the size, the horizontal offset, the vertical offset and the yaw of the visible gate.

Fig. 8.5 depicts examples of input images (left) and their corresponding reconstructions (right). We verify that the reconstruction captures the essence of the scene, preserving both the background and gate pose.

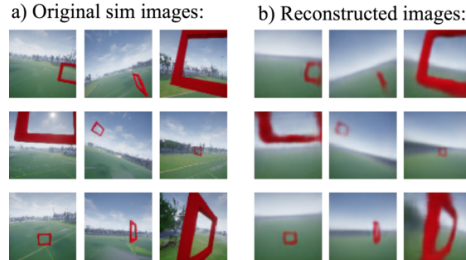


Figure 8.5: Comparison between original simulated images with their respective CM-VAE reconstructions. Reconstructed images are blurrier than the original, but overall gate and background features can be well represented.

The smoothness of the latent space manifold with respect to the gate poses and image outputs is a desirable property (*i.e.*, similar latent vectors correspond to similar gate poses). Intuitively, our single cross-modal latent space should lead to such smooth latent space representation, and our next analysis confirms that such properties emerge automatically. In Fig. 8.6 we show the decoded outputs of a latent space interpolation between the encoded vectors from two very different simulated images. Both images and their decoded poses are smoothly reconstructed along this manifold.

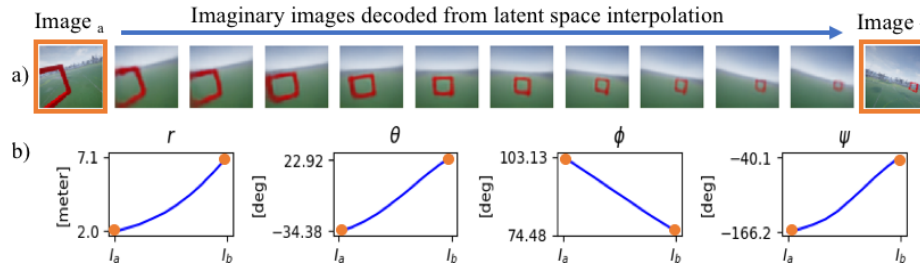


Figure 8.6: Visualization of latent space interpolation between two simulated images. Smooth interpolation can be perceived in both image and gate pose data modalities. Even background features such as the ground’s tilt are smoothly captured.

Additionally, we quantitatively evaluate the predictions of the three architectures that can recover the gate poses from the images, as shown in Table 8.1. When trained for the same number of epochs, q_{reg} , q_{unc} , and q_{con} achieve roughly the same error in prediction. The cross-modal latent space can encode gate pose information slightly better than direct regression, likely due to the additional unsupervised gradient information. Small variations can also be attributed to the training regime using stochastic gradient descent.

Table 8.1: Average and standard errors for encoding gate poses

q	Radius r [m]	Azimuth θ [°]	Polar ϕ [°]	Yaw ψ [°]
q_{reg}	0.41 ± 0.013	2.4 ± 0.14	2.5 ± 0.14	11 ± 0.67
q_{unc}	0.42 ± 0.024	2.3 ± 0.23	2.1 ± 0.23	9.7 ± 0.75
q_{con}	0.39 ± 0.023	2.6 ± 0.23	2.3 ± 0.25	10 ± 0.75

8.4.2 Simulated navigation results

Our next set of experiments evaluates control policies learned over five different types of feature extractors. As described in Section 8.3.3, we train behavior cloning policies on top of the CM-VAE latent spaces (BC_{con} , BC_{unc}), a direct gate pose regressor (BC_{reg}), vanilla VAE image reconstruction features (BC_{img}), and finally full end-to-end training (BC_{full}).

For data collection we generated a nominal circular track with 50m of length, over which we placed 8 gates with randomized position offsets in XYZ changing at every drone traversal. We collected 17.5K images with their corresponding expert velocity actions while varying the position offset level from 0-3m. 80%, or 14K datapoints, were used to train the behavior cloning policies, and the remainder were used for validation.

We evaluate our proposed framework under controlled simulation conditions analogous to data collection. Similarly to previous literature [120, 121, 154], we define a success metric of

100% as the UAV traversing all gates in 3 consecutive laps. For each data point we average results over 10 trials in different randomized tracks. Figure 8.7 shows the performance of different control policies which were trained using different latent representations, under increasing random position offset amplitudes. At a condition of zero noise added to the gates, most methods, except for the latent representation that uses pure image compression, can perfectly traverse all gates. As the track difficulty increases, end-to-end behavior cloning performance drops significantly, while methods that use latent representations degrade slower. At a noise level of 3 m over a track with 8 m of nominal radius the proposed cross-modal representation BC_{con} can still achieve approximately 40% success rate, 5X more than end-to-end learning. We invite the reader to watch the supplementary video for more details.

The three architectures that implicitly or explicitly encode gate positions (BC_{con} , BC_{unc} , BC_{reg}) perform significantly better than the baselines. This behavior likely spans from the small pixel-wise footprint of gates on the total image, which makes it harder for the vanilla VAE architecture or end-to-end learning to effectively capture the relative object poses. However, even though the regression features have a relatively good performance in simulation, policy success degrades when exposed to real-world images, as detailed in Subsection 8.4.3.

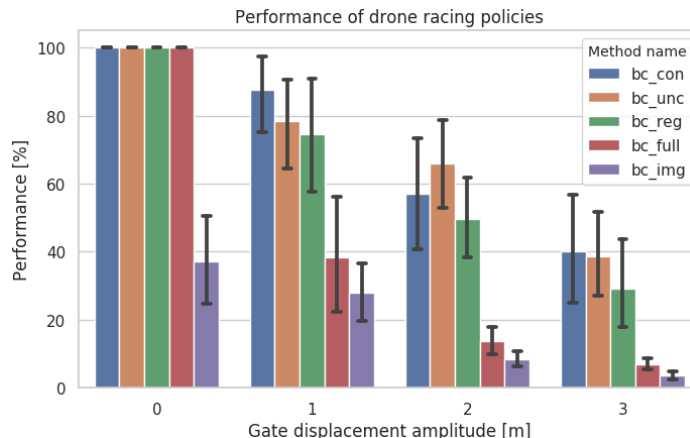


Figure 8.7: Performance of different navigation policies on simulated track

8.4.3 Real-World Results

We also validate the ability of the different visuomotor policies to transfer from simulation to real-world deployment. Our platform is a modified kit¹, as shown in Figure 8.8. All processing is done fully onboard with a Nvidia TX2 computer, with 6 CPU cores and an integrated GPU. An off-the-shelf Intel T265 Tracking Camera provides odometry, and image processing uses the Tensorflow framework. The image sensor is a USB camera with 83° horizontal FOV, and we downsize the original images to dimension 128 × 72.

First we evaluate how the CM-VAE module, which was learned only with simulated data, performs with real-world images as inputs. We only focus on z_{con} given that it presented the best overall performance in simulation experiments. Fig. 8.9 shows that the latent space

¹<https://www.getfpv.com/student-competition-5-bundle.html>

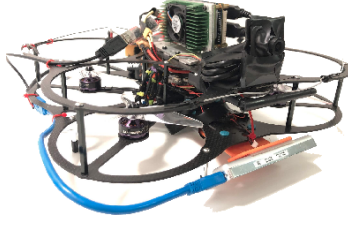


Figure 8.8: UAV platform.

encoding remains smooth and consistent. We train this model with a new simulation dataset composed of 100K images with size 128×72 and FOV equal to our hardware USB camera.

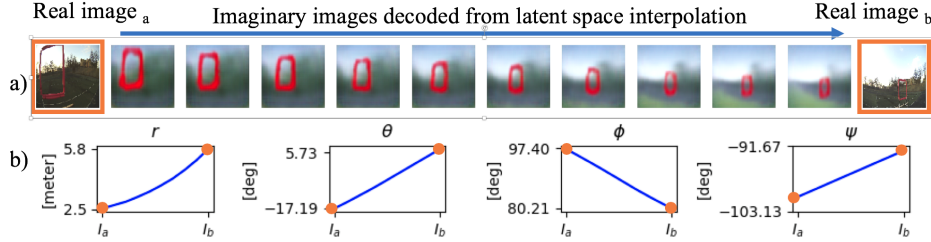


Figure 8.9: Visualization of smooth latent space interpolation between two real-world images. The ground-truth and predicted distances between camera and gate for images A and B were (2.0, 6.0) and (2.5, 5.8) meters respectively.

To show the capabilities of our approach on a physical platform, we test the system on a S-shaped track with 8 gates and 45m of length, and on a circular track with 8 gates and 40m of length, as shown in Fig 8.10. We compare three policies: BC_{con} , BC_{reg} , and BC_{real} . To train this last control policy, BC_{real} , we use a CM-VAE trained using not only the 100K images from simulation, but also additional 20K unsupervised real-world images. Our goal with this policy is to compare if the use of unsupervised real data can help in the extraction of better features for navigation.

We display results from both tracks on Table 8.2. The navigation policy using CM-VAE features trained purely in simulation significantly outperforms the baseline policies in both tracks, achieving over $3\times$ the performance of BC_{reg} in the S-track. The performance gap is even larger on the circuit track, with BC_{con} achieving a maximum of 26 gates traversed before any collision occurs. It is important to note that some of the circuit trials occurred among wind gusts of up to 20km/h, a fact that further shows that our policies learned in simulation can operate in physical environments.

Table 8.2: Policy performance in number of gates traversed

	S-Track [12 trials]		Circuit [6 trials]	
	Mean	Max	Mean	Max
BC_{con}	7.8	8	14.3	26
BC_{real}	5.0	7	3.1	5
BC_{reg}	2.3	5	1.2	2

We also investigate the cause of the performance drop in BC_{reg} when transferred to real-world data. Table 8.3 shows the ground-truth and predicted gate distances for different input

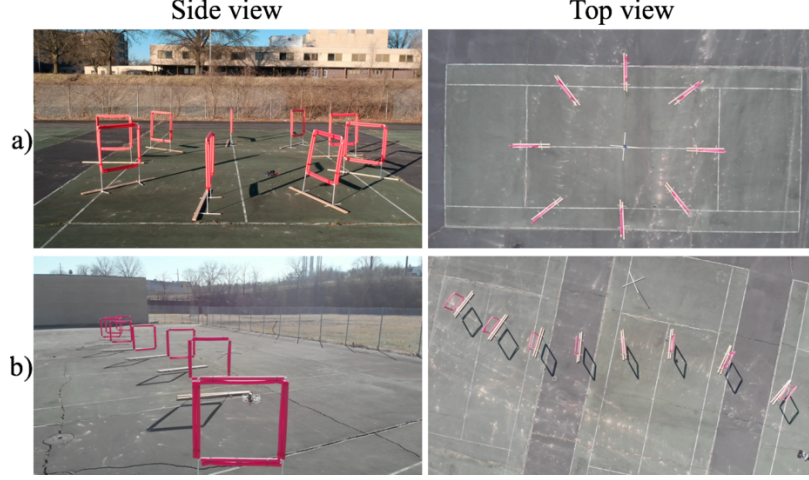


Figure 8.10: Side and top view of: a) Circuit track, and b) S-shape track.

images. The CM-VAE, despite being trained purely on simulation, can still decode reasonable values for the gate distances. Direct regression, however, presents larger errors. In addition, Figure 8.11 displays the accumulated gate poses as decoded from both representations during 3s of a real flight test. The regression poses are noticeably noisier and farther from the gate’s true location.

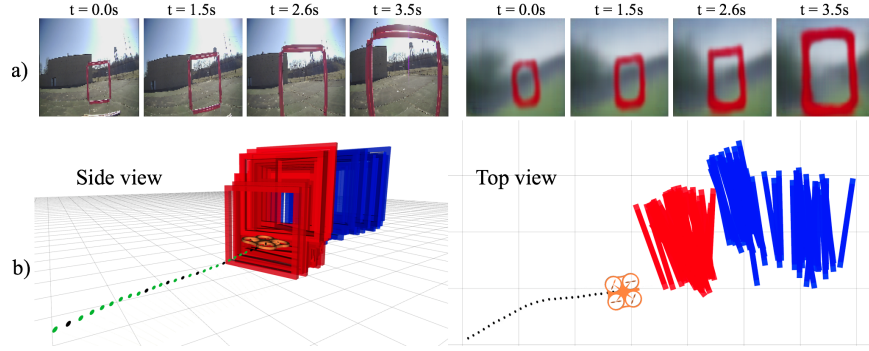






Figure 8.11: Analysis of a 3-second flight segment. a) Input images and their corresponding images decoded by the CM-VAE; b) Time history of gate center poses decoded from the CM-VAE (red) and regression (blue). The regression representation has significantly higher offset and noise from the true gate pose, which explains its poor flight performance.

In the experiments thus far we deployed the learned policies on physical environments that roughly resemble the visual appearances of the simulation dataset. There, all images were generated in a grass field with blue skies, and trees in the background. To verify policy and representation robustness to extreme visual changes, we perform additional tests in more challenging scenarios. Fig. 8.1 shows examples of successful test cases: Fig. 8.1a) indoors where the floor is blue with red stripes, and Fig. 8.1b-c) among heavy snow. We invite the reader to visualize these experiments in the video attachment.

Table 8.3: Examples of distance to gates decoded from real images

Image				
Ground-truth [m]	2	4	6	8
CM-VAE [m]	2.16	3.78	6.10	8.77
Regression [m]	4.67	5.50	6.68	9.13

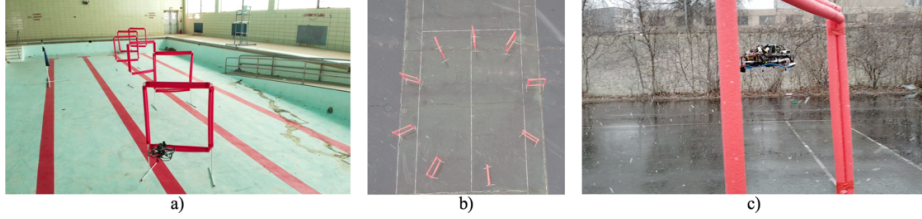


Figure 8.12: Examples of challenging test environments: a) Indoors, with blue floor with red stripes, and b-c) among heavy snowfall.

8.5 Conclusion and Discussion

In this work we present a framework to solve perception-control tasks that uses simulation-only data to learn rich representations of the environment, which can then be employed by visuomotor policies in real-world aerial navigation tasks. At the heart of our approach is a cross-modal Variational Auto-Encoder framework that jointly encodes raw sensor data and useful task-specific state information into a latent representation to be leveraged by a control policy. We provide detailed simulation and real-world experiments that highlight the effectiveness of our framework on the task of FPV drone navigation. Our results show that the use of cross-modal representations significantly improves the real-world performance of control policies in comparison with several baselines such as gate pose regression, unsupervised representations, and end-to-end approaches.

The main finding we can infer from our experiments is that by introducing multiple data modalities into the feature extraction step, we can avoid overfitting to specific characteristics of the incoming data. For example, even though the sizes of the square gates were the same in simulation and physical experiments, their width, color, and even intrinsic camera parameters are not an exact match. The multiple streams of information that the CM-VAE encounters regularize its model, leading to better generalization among appearance changes.

From our experiments in Fig. 8.7 we can also infer that features trained for unsupervised image reconstruction can serve as important cues describing the UAV’s current state for the control policy, on top of the explicit human-defined supervised parameters. For example, by using background features such as the line of horizon, a pilot may infer the UAV’s current roll angle, which influences the commanded velocities. This remark can serve as an additional motivator for the use of a semi-supervised features extraction module, since it is difficult to hand-define all relevant features for a particular control task. Another advantage of the CM-VAE architecture is that it can allow the robot operator to gain insights onto the decisions made by the networks. For instance, a human can interpret the decoded gate pose and decoded images in real time and stop the vehicle if perception seems to be abnormal.

Interestingly, BC_{real} did not outperform BC_{con} in our real-world experiments, as we originally expected. However, it was still better than BC_{reg} . We suspect that the drop in performance happens because the dataset used for imitation learning only contained images from simulation, and there is distribution shift in comparison with images used for training the representation. As future work we envision using adversarial techniques such as [245] for lowering the distance in latent space between similar scenes encountered in sim and real examples.

Additional future work includes extensions of the cross-modal semi-supervised feature extraction framework to other robotics tasks, considering the use of multiple unsupervised data modalities that span beyond images. We believe that applications such as autonomous driving and robotic manipulation present perception and control scenarios analogous to the aerial navigation task, where multiple modalities of simulated data can be cheaply obtained.

Chapter 9

Conclusion

This thesis argues for the proposition that in order for us to augment human creativity to its fullest, we need to develop *active cameras*, which are perception systems that can actively change their position and orientation in response to environmental stimuli. The proposed approach revolves around the unification of the perception and motion planning pipelines in robotics systems, taking contextual and geometrical features into account for planning algorithms. Machine learning can play an important role in this process, and this thesis shows various examples of how we can learn a mapping between subjective human-defined artistic objectives and cost functions that drive robot behavior.

Our main thesis hypothesis is that users are able to produce better quality movies with less effort by employing active vision methods coupled with machine learning algorithms. We begin this chapter by summarizing the arguments, the algorithmic contributions and the findings in Section 9.1 that back up our original hypothesis. Next, we present future avenues for research in Section 9.2 that build on the novel connections and observations that this thesis makes. Finally, we offer some concluding remarks in Section 9.3.

9.1 Summary and contributions

Section 1.2 exposed the fundamental challenges pertaining to this thesis:

- Keep actor visible while staying safe;
- Operate in unstructured scenarios;
- Estimate actor pose with challenging visual inputs;
- Understand the scene context for artistic decision-making;
- Translate user commands into robot decision-making;
- Make real-time decisions with onboard resources;
- Learn in simulation, deploy in the wild.

We now summarize the key contributions in the thesis, represented in Figure 9.1:

Chapter 3, Planning for cinematography: We defined the fundamental theoretical framework for planning the motion of physical and virtual cameras in space. We created

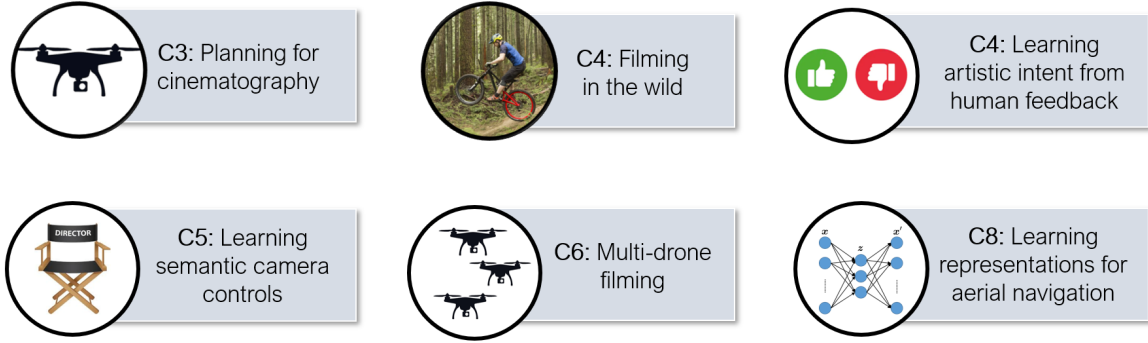


Figure 9.1: Summary of thesis contributions, along with the corresponding chapter numbers.

definitions for trajectories and cost functions, and proposed an efficient algorithm calculating camera trajectories. We showed that these algorithms work in complex simulated and real-world experiments.

Chapter 4, Filming in the wild: We developed an aerial filming system that can be deployed in more challenging real-world conditions, including online environment mapping and visual actor localization. We showed with extensive field experiments that the system is able to sustain good performance even with the removal of simplifying operational assumptions.

Chapter 5, Learning artistic intent from human feedback: We introduced a method for automatic selection of artistic shot parameters, emulating the role of a movie director. We trained the artistic policy directly using human feedback among various scenes, using deep reinforcement learning. We showed qualitatively and quantitatively the proposed approach maintains the aesthetic quality of the autonomous footage.

Chapter 6, Learning semantic camera controls: We examined a different variation of the camera control problem, where our goal was not to replace eliminate the human user, but rather to provide an intuitive interface for high-level parameter control. We developed a data-driven framework that enables editing of complex camera positioning parameters in a semantic space as opposed to low-level robot motion. We evaluated the system by demonstrating that our model successfully generated shots that are rated by participants as having the expected degrees of expression for multiple semantic descriptors.

Chapter 7, Multi-drone filming: We extended the cinematography theory developed for a single UAV towards a real-time multi-UAV coordination framework capable of recording dynamic targets while maximizing shot diversity and avoiding collisions and mutual visibility between cameras. We validated the proposed approach in multiple cluttered environments of a photo-realistic simulator, and deployed the system using two UAVs in real-world experiments.

Chapter 8, Learning representations for aerial navigation: In this chapter we stepped away from the problem of cinematography, and looked at more general theories for general aerial navigation. We developed state representations that can be easily transferred from simulation domains to real-world tasks. To do so, we proposed a method for learning robust visuomotor policies using cross-modal visual representations, and we showed that the use of

our architecture significantly improves control policy performance as compared to end-to-end learning or purely unsupervised feature extractors.

9.2 Future directions

The intersection of vehicle perception and intentional motion planning is rich, and contains several open problems. We survey and discuss a few promising directions for future research in the area.

9.2.1 Learning robot and camera motion styles

Our high-level goal in this work is to produce tools that improve human creativity, expressiveness, and sharing of experiences. However, when we consider the automation of the movie-making process, a major bottleneck comes from the fact that even though we all know a good clip when we see it, we cannot yet objectively specify a formula to generate one. This thesis takes the first steps towards learning artistic decision-making, but we envision a broad spectrum of open research areas when it comes to learn particular movie-making *styles*. So what exactly is a style? In his book *Filming Directing Fundamentals* [182], Proferes defines the concepts of style and design within the context of filming:

[...] there is a subtle difference between style and design, but the difference is worth recognizing. Style can be defined as an approach to the visualization of a story, while design can be defined as a plan. ‘Plan’ has a clear and unambiguous meaning, and the methodology for such planning is introduced in this book. But what is meant by ‘approach’? It has a vague connotation in this context, and yet I feel it is the fullest and most inclusive definition of how a director really works with style, especially one that is personal. The roots of style can be vague, nebulous, tenuous, hazy—as opposed to design, which connotes concreteness, clarity, and intelligibility. Original styles often come from the imagination that rests in the artist’s unconscious. A personal style is not something that can be ‘taught’. An original personal style, when revealed, becomes food for fodder and can then be incorporated by future directors in their design.

Within the field of dance, we find the work of Laban [136], which defines a minimal set of features of body motion styles. Knight [127] employed this feature space within the context of robotic motion to generate distinct styles for robot expression (*e.g.* happy/sad, confident/shy, rushed/lackadaisical). In [128], Knight mapped the Laban efforts of time, space, weight and flow towards trajectory parameters like velocities, accelerations, jerks and time delays. The field of cinematography requires a analogous approach, but with two important differences. First, we are not directly evaluating the motion of the aircraft, but the resulting video generated by the camera. And second, we need to define a novel feature space for filming styles considering several contextual elements beyond just the camera’s trajectory.

A few pioneer works started exploring the idea of imitating styles directly from footage data, such as [97, 106], but so far these methods only consider features related to the camera positioning relative to the actor over time. There is much work to be done still when considering the positioning and classes of objects and backgrounds in the scenery, lighting conditions, and longer-term history of camera actions [8, 25, 122, 182].

9.2.2 Using dynamic aerial cameras for new applications

Once the base technology for the coordination of aerial cameras exists, we can unlock a huge potential of new research applications that can be built on top of a dynamic image collection system. For instance, we can close the key gap in current data collection technology: there are no methods that can accurately measure large-scale dynamic group activities of people and animals in the wild. New research projects can emerge to tackle how teams UAVs can detect, track, predict, follow, coordinate static or dynamic subjects with the purpose reconstructing meshes and behaviors in space and time.

We find similarities between the actor reconstruction problem and coverage metrics within the context of exploration. For example, in [49, 50], Corah developed a greedy exploration scheme for multi-robot teams taking advantage of monotonicity and submodularity property of the objective function, which tries to reduce the entropy of the map. Under such objective conditions, greedy placement of robot positions has bounded sub-optimality guarantees [130, 210].

When filming a target we can define the *quality* of a viewpoint according to geometry, maximizing visual coverage from distinct points (Figure 9.2). This metric is submodular [194], and can be optimized using greedy approaches by computing the best next camera location for each new sensor. However, as pointed by Arora [6], the problem still remains challenging in terms of maintaining vehicle safety, obeying dynamics constraints and low onboard processing power. A possible method we can try to speed up computations is to learn to imitate an oracle utility function which gives us the best next viewpoints, following the works of Hepp [92] and Choudhury [46].

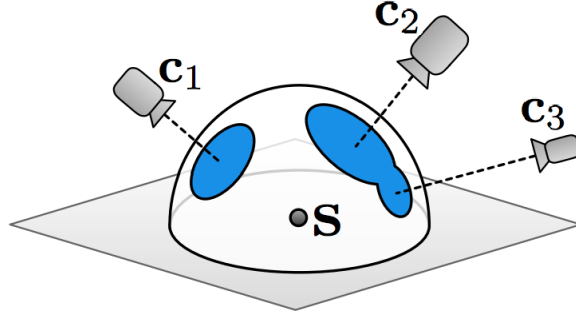


Figure 9.2: Solid angle coverage metric for multi-drone scenario. The summation of solid angles between cameras determines the quality of coverage. Note that camera c_3 offers a small advantage in comparison to camera c_2 , as their solid angles overlap. Figure extracted from [194].

9.2.3 Learning better state representations

In order to learn effective policies for navigation and interaction with the environment, mobile robots need to encode reliable representations of the environment and scene context. There are several important unsolved problems related to representation learning that one can tackle. For instance, we envision the extension of the cross-modal semi-supervised feature extraction framework presented in Chapter 8 towards other robotics tasks, considering the use of multiple unsupervised data modalities that span beyond images. We believe that applications such as autonomous driving and robotic manipulation present perception

and control scenarios analogous to the aerial navigation task, where multiple modalities of simulated data can be cheaply obtained.

Additionally, one investigate training more complex control policies on top of the learned latent representations, such as with the use of reinforcement learning (RL). To do such, we can obtain more accurate state representations by applying recurrent dynamics models to the features extracted with the CM-VAE, following techniques discussed in [87]. Within the context of RL, we can compare different architectures that employ auxiliary unsupervised tasks, such as the work of [103].

Another interesting and new line of work in multi-modal representation learning comes from the compensation of faulty data modalities, such as the work of [142]. We can measure the deviation of each modality from its expected value, conditioned on all sensor measurements, and detect corrupted inputs that come from sensor failure or occlusion.

9.3 Concluding remarks

We believe that the frameworks described in this thesis significantly extend the efficiency, applicability, and general usability of automated aerial cameras. However, many important issues remain as open research areas, as outlined in this chapter.

Ultimately, we hope that our methods can be used to expand the capabilities of real-world robotics systems in multiple other applications, spanning beyond the context of cinematography. Just within the field of aerial robotics, several activities related to data collection can benefit from the idea of *active vision* discussed in this document, from 3D scanning of structures to aerial surveillance and animal monitoring. We can also see how other robotics applications such as autonomous vehicles and manipulators can greatly benefit from many of the ideas presented here. For instance, by taking into account scene visibility and occlusion avoidance inside of the motion planning pipeline (Chapter 3), we can design robots that don't just passively capture environmental data, but on the contrary, can act conditioned on the current sensor feed. In addition, as robots start to operate closer to humans, the importance of generating legible and human-like behaviors grows, as these systems need to inspire trust on the users. Concepts such as those presented in Chapter 6 are an important first step in the direction of helping us learn how to generate robot trajectories that are associated with specific semantic or emotional descriptors. Although natural for humans, explicitly defining autonomous behaviors and styles is still a major challenge in robotics research today.

Appendix A

Derivations of planning cost functions

A.1 Smoothness cost

We define the smoothness cost function in its continuous form as:

$$J_{\text{smooth}}(\xi_q(t)) = \frac{1}{t_f} \frac{1}{2} \int_0^{t_f} \sum_{d=1}^{d_{\max}} \alpha_d (D^d \xi_q(t))^2 dt, \quad (\text{A.1})$$

We can discretize Equation 3.4 to compute the smoothness for ξ_q :

$$J_{\text{smooth}}(\xi_q) = \frac{1}{(n-1)} \frac{1}{2} \sum_{t=1}^{n-1} \left[\alpha_0 \left| \frac{p_t - p_{t-1}}{\Delta t} \right|^2 + \alpha_1 \left| \frac{\dot{p}_t - \dot{p}_{t-1}}{\Delta t} \right|^2 + \alpha_2 \left| \frac{\ddot{p}_t - \ddot{p}_{t-1}}{\Delta t} \right|^2 + \dots \right] \quad (\text{A.2})$$

To simplify Equation A.2, we define: $\Delta t = \frac{t_f}{n-1}$, a finite differentiation operator K , and an auxiliary matrix e for the contour conditions:

$$K_{(n-1) \times (n-1)} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 & 0 \\ -1 & 1 & 0 & \cdots & 0 & 0 & 0 \\ 0 & -1 & 1 & \cdots & 0 & 0 & 0 \\ & & & \ddots & & & \\ 0 & 0 & 0 & 0 \cdots & 0 & -1 & 1 \end{bmatrix} \quad e_{(n-1) \times 3} = \begin{bmatrix} -p_{0x} & -p_{0y} & -p_{0z} \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \end{bmatrix} \quad (\text{A.3})$$

By manipulating the terms K , e , and Δt we obtain the auxiliary terms:

$$\begin{aligned} K_0 &= \frac{K}{\Delta t}, & e_0 &= \frac{e}{\Delta t} \\ K_1 &= \frac{K^2}{\Delta t^2}, & e_1 &= \frac{Ke}{\Delta t^2} + \frac{\dot{e}}{\Delta t^2} \\ K_2 &= \frac{K^3}{\Delta t^3}, & e_2 &= \frac{K^2 e}{\Delta t^3} + \frac{K\dot{e}}{\Delta t^3} + \frac{\ddot{e}}{\Delta t^3} \end{aligned} \quad (\text{A.4})$$

$$\begin{aligned} A_0 &= K_0^T K_0, & b_0 &= K_0^T e_0, & c_0 &= e_0^T e_0 \\ A_1 &= K_1^T K_1, & b_1 &= K_1^T e_1, & c_1 &= e_1^T e_1 \\ A_2 &= K_2^T K_1, & b_2 &= K_2^T e_2, & c_2 &= e_2^T e_2 \end{aligned} \quad (\text{A.5})$$

Finally, we can analytically write the smoothness cost as a quadratic objective:

$$\begin{aligned} J_{\text{smooth}}(\xi_q) &= \frac{1}{2(n-1)} \text{Tr}(\xi_q^T A_{\text{smooth}} \xi_q + 2\xi_q^T b_{\text{smooth}} + c_{\text{smooth}}) \\ \text{where: } A_{\text{smooth}} &= \alpha_0 A_0 + \alpha_1 A_1 + \dots \\ b_{\text{smooth}} &= \alpha_0 b_0 + \alpha_1 b_1 + \dots \\ c_{\text{smooth}} &= \alpha_0 c_0 + \alpha_1 c_1 + \dots \end{aligned} \quad (\text{A.6})$$

Since $J_{\text{smooth}}(\xi_q)$ is quadratic, we find analytic expressions for its gradient and Hessian. Note that the Hessian expression does not depend on the current trajectory, which is a property used serve to speed up the optimization algorithm described in Subsection 3.2.2:

$$\begin{aligned} \nabla J_{\text{smooth}}(\xi_q) &= \frac{1}{(n-1)} (A_{\text{smooth}} \xi_q + b_{\text{smooth}}) \\ \nabla^2 J_{\text{smooth}}(\xi_q) &= \frac{1}{(n-1)} A_{\text{smooth}} \end{aligned} \quad (\text{A.7})$$

A.2 Shot quality cost

We find the continuous formulation of the shot quality cost function as:

$$J_{\text{shot}}(\xi_q, \xi_a) = \frac{1}{t_f} \frac{1}{2} \int_0^{t_f} \|\xi_q(t) - \xi_{\text{shot}}(\xi_a(t))\|^2 dt \quad (\text{A.8})$$

We can calculate J_{shot} in the discrete form:

$$J_{\text{shot}}(\xi_q, \xi_a) = \frac{1}{(n-1)} \frac{1}{2} \sum_{t=1}^n |p_t - p_{\text{t shot}}|^2 \quad (\text{A.9})$$

By defining auxiliary matrices, we can also define a quadratic expression:

$$J_{\text{shot}}(\xi_q, \xi_a) = \frac{1}{2(n-1)} \text{Tr}(\xi_q^T A_{\text{shot}} \xi_q + 2\xi_q^T b_{\text{shot}} + c_{\text{shot}}), \quad (\text{A.10})$$

where:

$$K_{\text{shot}} = -I_{(n-1) \times (n-1)}$$

$$e_{\text{shot}} = \xi_{\text{shot}} = \begin{bmatrix} p_{1x \text{ shot}} & p_{1y \text{ shot}} & p_{1z \text{ shot}} \\ p_{1x \text{ shot}} & p_{1y \text{ shot}} & p_{1z \text{ shot}} \\ \vdots & \vdots & \vdots \\ p_{(n-1)x \text{ shot}} & p_{(n-1)y \text{ shot}} & p_{(n-1)z \text{ shot}} \end{bmatrix}, \quad (\text{A.11})$$

and:

$$A_{\text{shot}} = K_{\text{shot}}^T K_{\text{shot}}, \quad b_{\text{shot}} = K^T e_{\text{shot}}, \quad c_{\text{shot}} = e_{\text{shot}}^T e_{\text{shot}} \quad (\text{A.12})$$

We can again find analytic expressions for the shot quality gradient and Hessian, which is independent from the current trajectory:

$$\nabla J_{\text{shot}}(\xi_q) = \frac{1}{(n-1)} (A_{\text{shot}} \xi_q + b_{\text{shot}})$$

$$\nabla^2 J_{\text{shot}}(\xi_q) = \frac{1}{(n-1)} A_{\text{shot}} \quad (\text{A.13})$$

Bibliography

- [1] Dji mavic. <https://www.dji.com/br/mavic>, August 2020.
- [2] Sandip Aine, Siddharth Swaminathan, Venkatraman Narayanan, Victor Hwang, and Maxim Likhachev. Multi-heuristic a. *The International Journal of Robotics Research*, 35(1-3):224–243, 2016.
- [3] Amazon. Amazon mechanical turk.
- [4] Ido Arev, Hyun Soo Park, Yaser Sheikh, Jessica Hodgins, and Ariel Shamir. Automatic editing of footage from multiple social cameras. *ACM Transactions on Graphics (TOG)*, 33(4):1–11, 2014.
- [5] Daniel Arijon. Grammar of the film language. 1976.
- [6] Sankalp Arora. *Safe Data Gathering in Physical Spaces*. PhD thesis, Carnegie Mellon University, 2018.
- [7] Sankalp Arora and Sebastian Scherer. Randomized algorithm for informative path planning with budget constraints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4997–5004. IEEE, 2017.
- [8] Steven Ascher and Edward Pincus. *The filmmaker’s handbook: A comprehensive guide for the digital age*. Penguin, 2007.
- [9] Amirsaman Ashtari, Stefan Stevšić, Tobias Nägeli, Jean-Charles Bazin, and Otmar Hilliges. Capturing subjective first-person view shots with drones for automated cinematography. *ACM Transactions on Graphics (TOG)*, 39(5):1–14, 2020.
- [10] Yusuf Aytar, Lluís Castrejon, Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Cross-modal scene networks. *IEEE transactions on pattern analysis and machine intelligence*, 40(10):2303–2314, 2017.
- [11] Ruzena Bajcsy. Active perception. *Proceedings of the IEEE*, 76(8):966–1005, 1988.
- [12] Dana H Ballard. Animate vision. *Artificial intelligence*, 48(1):57–86, 1991.
- [13] Yoann Baveye, Emmanuel Dellandrea, Christel Chamaret, and Liming Chen. Liris-accede: A video database for affective content analysis. *IEEE Transactions on Affective Computing*, 6(1):43–55, 2015.
- [14] Haluk Bayram, Nikolaos Stefas, Kazim Selim Engin, and Volkan Isler. Tracking wildlife with multiple uavs: System design, safety and field experiments. In *2017 International symposium on multi-robot and multi-agent systems (MRS)*, pages 97–103. IEEE, 2017.

- [15] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.
- [16] Rogerio Bonatti, Arthur Buckner, Sebastian Scherer, Mustafa Mukadam, and Jessica Hodgins. Batteries, camera, action! learning a semantic control space for expressive robot cinematography. *ICRA 2021*, 2021.
- [17] Rogerio Bonatti, Cherie Ho, Wenshan Wang, Sanjiban Choudhury, and Sebastian Scherer. Towards a robust aerial cinematography platform: Localizing and tracking moving targets in unstructured environments. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [18] Rogerio Bonatti, Ratnesh Madaan, Vibhav Vineet, Sebastian Scherer, and Ashish Kapoor. Learning visuomotor policies for aerial navigation using cross-modal representations. *IROS 2020*, 2020.
- [19] Rogerio Bonatti, Wenshan Wang, Cherie Ho, Aayush Ahuja, Mirko Gschwindt, Efe Camci, Erdal Kayacan, Sanjiban Choudhury, and Sebastian Scherer. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *Journal of Field Robotics*, 2019.
- [20] Rogerio Bonatti, Wenshan Wang, Cherie Ho, Aayush Ahuja, Mirko Gschwindt, Efe Camci, Erdal Kayacan, Sanjiban Choudhury, and Sebastian Scherer. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *Journal of Field Robotics*, 2020.
- [21] Rogerio Bonatti, Wenshan Wang, Cherie Ho, Aayush Ahuja, Mirko Gschwindt, Efe Camci, Erdal Kayacan, Sanjiban Choudhury, and Sebastian Scherer. Autonomous aerial cinematography in unstructured environments with learned artistic decision-making. *Journal of Field Robotics*, 37(4):606–641, January 2020.
- [22] Rogerio Bonatti, Yanfu Zhang, Sanjiban Choudhury, Wenshan Wang, and Sebastian Scherer. Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming. *International Symposium on Experimental Robotics*, 2018.
- [23] Rogerio Bonatti, Yanfu Zhang, Sanjiban Choudhury, Wenshan Wang, and Sebastian Scherer. Autonomous drone cinematographer: Using artistic principles to create smooth, safe, occlusion-free trajectories for aerial filming, 2018.
- [24] Elizabeth Bondi, Debadeepta Dey, Ashish Kapoor, Jim Piavis, Shital Shah, Fei Fang, Bistra Dilkina, Robert Hannaford, Arvind Iyer, Lucas Joppa, et al. Airsim-w: A simulation environment for wildlife conservation with uavs. In *Proceedings of the 1st ACM SIGCAS Conference on Computing and Sustainable Societies*, pages 1–12, 2018.
- [25] David Bordwell, Kristin Thompson, and Jeff Smith. *Film art: An introduction*, volume 7. McGraw-Hill New York, 1993.
- [26] Christopher J Bowen and Roy Thompson. *Grammar of the Shot*. Taylor & Francis, 2013.
- [27] Stephen Boyd and Lieven Vandenbergh. *Convex optimization*. Cambridge university press, 2004.

- [28] Cynthia Breazeal, Aaron Edsinger, Paul Fitzpatrick, and Brian Scassellati. Active vision for sociable robots. *IEEE Transactions on systems, man, and cybernetics-part A: Systems and Humans*, 31(5):443–453, 2001.
- [29] Arthur Buckner, Rogerio Bonatti, and Sebastian Scherer. Do you see what i see? coordinating multiple aerial cameras for robot cinematography. *ICRA 2021*, 2021.
- [30] Christopher P. Burgess, Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *CoRR*, 2019.
- [31] Michael Burri, Helen Oleynikova, , Markus W. Achtelik, and Roland Siegwart. Real-time visual-inertial mapping, re-localization and planning onboard mavs in unknown environments. In *Intelligent Robots and Systems (IROS 2015), 2015 IEEE/RSJ International Conference on*, September 2015.
- [32] Erik Cambria and Amir Hussain. Sentic computing. *Cognitive Computation*, 7(2):183–185, 2015.
- [33] Luca Canini, Sergio Benini, and Riccardo Leonardi. Affective analysis on patterns of shot types in movies. In *2011 7th International Symposium on Image and Signal Processing and Analysis (ISPA)*, pages 253–258. IEEE, 2011.
- [34] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017.
- [35] Ellen A Cappel, Arjav Desai, Matthew Collins, and Nathan Michael. Online planning for human–multi-robot interactive theatrical performance. *Autonomous Robots*, 42(8):1771–1786, 2018.
- [36] Ellen A Cappel, Arjav Desai, and Nathan Michael. Robust coordinated aerial deployments for theatrical applications given online user interaction via behavior composition. In *Distributed Autonomous Robotic Systems*, pages 665–678. Springer, 2018.
- [37] Luis-Evaristo Caraballo, Ángel Montes-Romero, José-Miguel Díaz-Báñez, Jesús Capitán, Arturo Torres-González, and Aníbal Ollero. Autonomous planning for multiple aerial cinematographers, 2020.
- [38] Meghan Chandarana, Wenhao Luo, Michael Lewis, Katia Sycara, and Sebastian Scherer. Decentralized method for sub-swarm deployment and rejoining. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1209–1214. IEEE, 2018.
- [39] Benjamin Charrow, Gregory Kahn, Sachin Patil, Sikang Liu, Ken Goldberg, Pieter Abbeel, Nathan Michael, and Vijay Kumar. Information-theoretic planning with trajectory optimization for dense 3d mapping. In *Robotics: Science and Systems*, volume 11. Rome, 2015.
- [40] Benjamin Charrow, Vijay Kumar, and Nathan Michael. Approximate representations for multi-robot control policies that maximize mutual information. *Autonomous Robots*, 37(4):383–400, 2014.
- [41] Jianhui Chen and Peter Carr. Mimicking human camera operators. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 215–222. IEEE, 2015.

- [42] Jianhui Chen and James J Little. Where should cameras look at soccer games: Improving smoothness using the overlapped hidden markov model. *Computer Vision and Image Understanding*, 159:59–73, 2017.
- [43] Jinyoung Choi, Beom-Jin Lee, and Byoung-Tak Zhang. Human body orientation estimation using convolutional neural network. *arXiv preprint arXiv:1609.01984*, 2016.
- [44] Sanjiban Choudhury, Vishal Dugar, Silvio Maeta, Brian MacAllister, Sankalp Arora, Daniel Althoff, and Sebastian Scherer. High performance and safe flight of full-scale helicopters from takeoff to landing with an ensemble of planners. *Journal of Field Robotics*, 36(8):1275–1332, 2019.
- [45] Sanjiban Choudhury, Jonathan D Gammell, Timothy D Barfoot, Siddhartha S Srinivasa, and Sebastian Scherer. Regionally accelerated batch informed trees (rabit*): A framework to integrate local information into optimal path planning. In IEEE, editor, *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4207–4214. IEEE, May 2016.
- [46] Sanjiban Choudhury, Ashish Kapoor, Gireeja Ranade, Sebastian Scherer, and Debadepta Dey. Adaptive information gathering via imitation learning. *arXiv preprint arXiv:1705.07834*, 2017.
- [47] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.
- [48] Marc Christie, Patrick Olivier, and Jean-Marie Normand. Camera control in computer graphics. In *Computer Graphics Forum*, volume 27, pages 2197–2218. Wiley Online Library, 2008.
- [49] Micah Corah and Nathan Michael. Efficient online multi-robot exploration via distributed sequential greedy assignment. In *Robotics: Science and Systems*, volume 13, 2017.
- [50] Micah Corah and Nathan Michael. Distributed matroid-constrained submodular maximization for multi-robot exploration: Theory and practice. *Autonomous Robots*, 43(2):485–501, 2019.
- [51] Hugh Cover, Sanjiban Choudhury, Sebastian Scherer, and Sanjiv Singh. Sparse tangential network (spartan): Motion planning for micro aerial vehicles. In *2013 IEEE International Conference on Robotics and Automation*, pages 2820–2825. IEEE, May 2013.
- [52] James E Cutting and Ayse Candan. Shot durations, shot classes, and the increased pace of popular movies, 2015.
- [53] James E Cutting, Jordan E DeLong, and Christine E Nothelfer. Attention and the evolution of hollywood film. *Psychological science*, 21(3):432–439, 2010.
- [54] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Ruslan R Salakhutdinov. Good semi-supervised learning that requires a bad gan. In *Advances in Neural Information Processing Systems*, pages 6513–6523, 2017.

- [55] Mara De-Miguel-Molina. *Ethics and Civil Drones: European Policies and Proposals for the Industry*. 2018.
- [56] María De-Miguel-Molina. *Ethics and Civil Drones: European Policies and Proposals for the Industry*. Springer, 2018.
- [57] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018.
- [58] Zhian Deng, Weijian Si, Zhiyu Qu, Xin Liu, and Zhenyu Na. Heading estimation fusing inertial sensors and landmarks for indoor navigation using a smartphone in the pocket. *EURASIP Journal on Wireless Communications and Networking*, 2017(1):160, 2017.
- [59] Ruta Desai, Fraser Anderson, Justin Matejka, Stelian Coros, James McCann, George Fitzmaurice, and Tovi Grossman. Geppetto: Enabling semantic design of expressive robot behaviors. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2019.
- [60] Max Welling Diederik Kingma. Autoencoding variational bayes. In *ICLR*, 2014.
- [61] DJI. Dji mavic, <https://www.dji.com/mavic>, 2018.
- [62] Jeremy Douglass. Patterns across 1100 feature films, 1900-2008.
- [63] Steven M Drucker and David Zeltzer. Intelligent camera control in a virtual environment. In *Graphics Interface*, pages 190–190. Citeseer, 1994.
- [64] Mohamed Elbanhawi and Milan Simic. Sampling-based robot motion planning: A review. *IEEE Access*, (2):56–77, 2014.
- [65] Arpad E Elo. *The rating of chessplayers, past and present*. Arco Pub., 1978.
- [66] Selim Engin and Volkan Isler. Active localization of multiple targets using noisy relative measurements. *arXiv preprint arXiv:2002.09850*, 2020.
- [67] Hui Fang and Meng Zhang. Creatism: A deep-learning photographer capable of creating professional work. *arXiv preprint arXiv:1707.03491*, 2017.
- [68] Fabian Flohr, Madalin Dumitru-Guzu, Julian FP Kooij, and Dariu M Gavrilă. A probabilistic framework for joint pedestrian head and body orientation estimation. *IEEE Transactions on Intelligent Transportation Systems*, 16(4):1872–1882, 2015.
- [69] Eric Foote, Peter Carr, Patrick Lucey, Yaser Sheikh, and Iain Matthews. One-man-band: A touch screen interface for producing live multi-camera sports broadcasts. In *Proceedings of the 21st ACM international conference on Multimedia*, pages 163–172, 2013.
- [70] Brendan J Frey and Delbert Dueck. Clustering by passing messages between data points. *science*, 315(5814):972–976, 2007.
- [71] Quentin Galvane, Julien Fleureau, Francois-Louis Tariolle, and Philippe Guillo-tel. Automated cinematography with unmanned aerial vehicles. *arXiv preprint arXiv:1712.04353*, pages 23–30, 2017.

- [72] Quentin Galvane, Julien Fleureau, Francois-Louis Tariolle, and Philippe Guillotel. Automated cinematography with unmanned aerial vehicles. *arXiv preprint arXiv:1712.04353*, 2017.
- [73] Quentin Galvane, Christophe Lino, Marc Christie, Julien Fleureau, Fabien Servant, Fran Tariolle, Philippe Guillotel, et al. Directing cinematographic drones. *ACM Transactions on Graphics (TOG)*, 37(3):34, 2018.
- [74] Quentin Galvane, Christophe Lino, Marc Christie, Julien Fleureau, Fabien Servant, François-louis Tariolle, and Philippe Guillotel. Directing cinematographic drones. *ACM Transactions on Graphics (TOG)*, 37(3):1–18, 2018.
- [75] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.
- [76] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2414–2423, 2016.
- [77] Christoph Gebhardt, Benjamin Hepp, Tobias Nägeli, Stefan Stevšić, and Otmar Hilliges. Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2508–2519. ACM, 2016.
- [78] Christoph Gebhardt, Benjamin Hepp, Tobias Nageli, Stefan Stevšić, and Otmar Hilliges. Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, pages 2508–2519, 2016.
- [79] Christoph Gebhardt, Stefan Stevšić, and Otmar Hilliges. Optimizing for aesthetically pleasing qadrotor camera motion. *ACM Transactions on Graphics (TOG)*, 37(4):90, 2018.
- [80] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013.
- [81] Michael Gleicher and Andrew Witkin. Through-the-lens camera control. In *ACM SIGGRAPH Computer Graphics*, volume 26, pages 331–340. ACM, 1992.
- [82] Michael S Gordon, James Robert Kozloski, Ashish Kundu, and Clifford A Pickover. Specialized contextual drones for animal virtual fences and herding, February 1 2018. US Patent App. 15/223,351.
- [83] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proceedings of the IEEE international conference on computer vision*, pages 4086–4093, 2015.
- [84] Mirko Gschwindt, Efe Camci, Rogerio Bonatti, Wenshan Wang, Erdal Kayacan, and Sebastian Scherer. Can a robot become a movie director? learning artistic principles for aerial cinematography. *arXiv preprint arXiv:1904.02579*, 2019.

- [85] Mirko Gschwindt, Efe Camci, Rogerio Bonatti, Wenshan Wang, and Sebastian Scherer. Can a robot become a movie director? learning artistic principles for aerial cinematography. *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.
- [86] Michael Gygli, Helmut Grabner, and Luc Van Gool. Video summarization by learning submodular mixtures of objectives. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3090–3098, 2015.
- [87] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.
- [88] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.
- [89] Alan Hanjalic and Li-Qun Xu. Affective video content representation and modeling. *IEEE transactions on multimedia*, 7(1):143–154, 2005.
- [90] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [91] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. High-speed tracking with kernelized correlation filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(3):583–596, 2015.
- [92] Benjamin Hepp, Debadeepta Dey, Sudipta N Sinha, Ashish Kapoor, Neel Joshi, and Otmar Hilliges. Learn-to-score: Efficient 3d scene exploration by predicting view utility. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 437–452, 2018.
- [93] Ralf Herbrich, Tom Minka, and Thore Graepel. Trueskill™: a bayesian skill rating system. In *Advances in neural information processing systems*, pages 569–576, 2007.
- [94] Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France*, 2017.
- [95] Elad Hoffer and Nir Ailon. Semi-supervised deep learning by metric embedding. *arXiv preprint arXiv:1611.01449*, 2016.
- [96] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [97] Chong Huang, Yuanjie Dang, Peng Chen, Xin Yang, et al. One-shot imitation filming of human motion videos. *arXiv preprint arXiv:1912.10609*, 2019.
- [98] Chong Huang, Fei Gao, Jie Pan, Zhenyu Yang, Weihao Qiu, Peng Chen, Xin Yang, Shaojie Shen, and Kwang-Ting Tim Cheng. Act: An autonomous drone cinematogra-

- phy system for action scenes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7039–7046. IEEE, 2018.
- [99] Chong Huang, Zhenyu Yang, Yan Kong, Peng Chen, Xin Yang, and Kwang-Ting Tim Cheng. Through-the-lens drone filming. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4692–4699. IEEE, 2018.
 - [100] Peter J Huber et al. Robust estimation of a location parameter. *The annals of mathematical statistics*, 35(1):73–101, 1964.
 - [101] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, July 2014.
 - [102] Stefan Isler, Reza Sabzevari, Jeffrey Delmerico, and Davide Scaramuzza. An information gain formulation for active volumetric 3d reconstruction. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3477–3484. IEEE, 2016.
 - [103] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.
 - [104] Robert Jenssen, Davide Roverso, et al. Automatic autonomous vision-based power line inspection: A review of current status and the potential role of deep learning. *International Journal of Electrical Power & Energy Systems*, 99:107–120, 2018.
 - [105] Boseong Felipe Jeon, Dongseok Shim, and H Jin Kim. Detection-aware trajectory generation for a drone cinematographer. *arXiv preprint arXiv:2009.01565*, 2020.
 - [106] Hongda Jiang, Bin Wang, Xi Wang, Marc Christie, and Baoquan Chen. Example-driven virtual cinematography by learning camera behaviors. *ACM Transactions on Graphics (TOG)*, 39(4):45–1, 2020.
 - [107] Yuqian Jiang, Harel Yedidsion, Shiqi Zhang, Guni Sharon, and Peter Stone. Multi-robot planning with conflicts and synergies. *Autonomous Robots*, 43(8):2011–2032, 2019.
 - [108] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3334–3342, 2015.
 - [109] Neel Joshi, Wolf Kienzle, Mike Toelle, Matt Uyttendaele, and Michael F Cohen. Real-time hyperlapse creation via optimal frame selection. *ACM Transactions on Graphics (TOG)*, 34(4):1–9, 2015.
 - [110] Niels Joubert, Dan B Goldman, Floraine Berthouzoz, Mike Roberts, James A Landay, Pat Hanrahan, et al. Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles. *arXiv preprint arXiv:1610.01691*, 2016.

- [111] Niels Joubert, Dan B Goldman, Floraine Berthouzoz, Mike Roberts, James A Landay, Pat Hanrahan, et al. Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles. *arXiv preprint arXiv:1610.01691*, 2016.
- [112] Niels Joubert, Mike Roberts, Anh Truong, Floraine Berthouzoz, and Pat Hanrahan. An interactive tool for designing quadrotor camera shots. *ACM Transactions on Graphics (TOG)*, 34(6):238, 2015.
- [113] Sunggoo Jung, Sungwook Cho, Dasol Lee, Hanseob Lee, and David Hyunchul Shim. A direct visual servoing-based framework for the 2016 iros autonomous drone racing challenge. *Journal of Field Robotics*, 35(1):146–166, 2018.
- [114] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robotics and Automation Letters*, 3(3):2539–2544, 2018.
- [115] Sunggoo Jung, Hanseob Lee, Sunyou Hwang, and David Hyunchul Shim. Real time embedded system framework for autonomous drone racing using deep learning techniques. In *2018 AIAA Information Systems-AIAA Infotech Aerospace*, page 2138. 2018.
- [116] Katie Kang, Suneel Belkhale, Gregory Kahn, Pieter Abbeel, and Sergey Levine. Generalization through simulation: Integrating simulated and real data into deep reinforcement learning for vision-based autonomous flight. *arXiv preprint arXiv:1902.03701*, 2019.
- [117] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.
- [118] Nikhil Karnad and Volkan Isler. Modeling human motion patterns for multi-robot planning. In *2012 IEEE International Conference on Robotics and Automation*, pages 3161–3166. IEEE, 2012.
- [119] Andrej Karpathy. What a deep neural network thinks about your #selfie, andrej karpathy blog. karpathy. github. io, 2015.
- [120] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 690–696. IEEE, 2019.
- [121] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: Learning agile flight in dynamic environments. *arXiv preprint arXiv:1806.08548*, 2018.
- [122] Christopher Kenworthy. *Master Shots Vol 1: 100 Advanced Camera Techniques to Get an Expensive Look on Your Low-Budget Movie*, volume 100. Michael Wiese Productions, 2011.
- [123] Hyunjik Kim and Andriy Mnih. Disentangling by factorising. *arXiv preprint arXiv:1802.05983*, 2018.
- [124] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

- [125] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European Conference on Computer Vision*, pages 201–214. Springer, 2012.
- [126] Matthew Klingensmith, Ivan Dryanovski, Siddhartha Srinivasa, and Jizhong Xiao. Chisel: Real time large scale 3d reconstruction onboard a mobile device using spatially hashed signed distance fields. In *Robotics: Science and Systems*, volume 4, 2015.
- [127] Heather Knight. Expressive motion for low degree-of-freedom robots. 2016.
- [128] Heather Knight and Reid Simmons. Laban head-motions convey robot state: A call for robot body language. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 2881–2888. IEEE, 2016.
- [129] Andreas Krause, Carlos Guestrin, Anupam Gupta, and Jon Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the 5th international conference on Information processing in sensor networks*, pages 2–10, 2006.
- [130] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.
- [131] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(Feb):235–284, 2008.
- [132] Akshay Krishnamurthy, Alekh Agarwal, and John Langford. Contextual-mdps for pacreinforcement learning with rich observations. *arXiv preprint arXiv:1602.02722*, 2016.
- [133] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.
- [134] JJ Kuffner and SM LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.
- [135] Laura La Bella. *Drones and Entertainment*. The Rosen Publishing Group, Inc, 2016.
- [136] Rudolph Laban. Modern educational dance. revised by l. Ullmann. *London: MacDonald and Evans. (First published 1948)*, 1963.
- [137] Pierre-Yves Laffont, Zhile Ren, Xiaofeng Tao, Chao Qian, and James Hays. Transient attributes for high-level understanding and editing of outdoor scenes. *ACM Transactions on graphics (TOG)*, 33(4):1–11, 2014.
- [138] Ziquan Lan, Mohit Shridhar, David Hsu, and Shengdong Zhao. Xpose: Reinventing user interaction with flying cameras. In *Robotics: Science and Systems*, 2017.
- [139] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at <http://planning.cs.uiuc.edu/>.
- [140] Steven M LaValle. *Planning algorithms*. Cambridge university press, 2006.

- [141] Mackenzie Leake, Abe Davis, Anh Truong, and Maneesh Agrawala. Computational video editing for dialogue-driven scenes. *ACM Trans. Graph.*, 36(4):130–1, 2017.
- [142] Michelle A Lee, Matthew Tan, Yuke Zhu, and Jeannette Bohg. Detect, reject, correct: Crossmodal compensation of corrupted sensors. *arXiv preprint arXiv:2012.00201*, 2020.
- [143] Timothée Lesort, Natalia Díaz-Rodríguez, Jean-Francois Goudou, and David Filliat. State representation learning for control: An overview. *Neural Networks*, 108:379–392, 2018.
- [144] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.
- [145] Christophe Lino and Marc Christie. Intuitive and efficient camera control with the toric space. *ACM Transactions on Graphics (TOG)*, 34(4):82, 2015.
- [146] Christophe Lino, Marc Christie, Roberto Ranon, and William Bares. The director’s lens: an intelligent assistant for virtual cinematography. In *Proceedings of the 19th ACM international conference on Multimedia*, pages 323–332. ACM, 2011.
- [147] Venice Erin Liong, Jiwen Lu, Yap-Peng Tan, and Jie Zhou. Cross-modal deep variational hashing. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 4097–4105. IEEE, 2017.
- [148] Donghui Liu, Ling Pei, Jiuchao Qian, Lin Wang, Peilin Liu, Zhenjiang Dong, Siyuan Xie, and Wei Wei. A novel heading estimation algorithm for pedestrian using a smartphone without attitude constraints. In *Ubiquitous Positioning, Indoor Navigation and Location Based Services (UPINLBS), 2016 Fourth International Conference on*, pages 29–37. IEEE, 2016.
- [149] Lantao Liu and Nathan Michael. An mdp-based approximation method for goal constrained multi-mav planning under action uncertainty. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 56–62. IEEE, 2016.
- [150] Peiye Liu, Wu Liu, and Huadong Ma. Weighted sequence loss based spatial-temporal deep learning framework for human body orientation estimation. In *Multimedia and Expo (ICME), 2017 IEEE International Conference on*, pages 97–102. IEEE, 2017.
- [151] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [152] Wu Liu, Yongdong Zhang, Sheng Tang, Jinhui Tang, Richang Hong, and Jintao Li. Accurate estimation of human body orientation from rgb-d sensors. *IEEE Transactions on Cybernetics*, 43(5):1442–1452, 2013.
- [153] BCC Research LLC. Drone technology and global markets. *IAS104B*, 2018.
- [154] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *arXiv preprint arXiv:1905.09727*, 2019.

- [155] Antonio Loquercio, Ana I Maqueda, Carlos R Del-Blanco, and Davide Scaramuzza. Dronet: Learning to fly by driving. *IEEE Robotics and Automation Letters*, 3(2):1088–1095, 2018.
- [156] Ryan Luna, Ioan A Șucan, Mark Moll, and Lydia E Kavraki. Anytime solution optimization for sampling-based motion planning. In *2013 IEEE international conference on robotics and automation*, pages 5068–5074. IEEE, 2013.
- [157] Wenhao Luo, Nilanjan Chakraborty, and Katia Sycara. Distributed dynamic priority assignment and motion planning for multiple mobile robots with kinodynamic constraints. In *2016 American Control Conference (ACC)*, pages 148–154. IEEE, 2016.
- [158] Wenhao Luo and Ashish Kapoor. Multi-robot collision avoidance under uncertainty with probabilistic safety barrier certificates. *arXiv preprint arXiv:1912.09957*, 2019.
- [159] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [160] Jana Machajdik and Allan Hanbury. Affective image classification using features inspired by psychology and art theory. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 83–92, 2010.
- [161] Ratnesh Madaan, Daniel Maturana, and Sebastian Scherer. Wire detection using synthetic data and dilated convolutional networks for unmanned aerial vehicles. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3487–3494. IEEE, 2017.
- [162] I. Mademlis, V. Mygdalis, N. Nikolaidis, M. Montagnuolo, F. Negro, A. Messina, and I. Pitas. High-level multiple-uav cinematography tools for covering outdoor events. *IEEE Transactions on Broadcasting*, 65(3):627–635, 2019.
- [163] Bhushan Mahadani and Bhushan Mahadani. How film shot length decrease in last century, Feb 2015.
- [164] Albert Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in temperament. *Current Psychology*, 14(4):261–292, 1996.
- [165] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2520–2525. IEEE, 2011.
- [166] Angeliki Metallinou and Shrikanth Narayanan. Annotation and processing of continuous emotional attributes: Challenges and opportunities. In *2013 10th IEEE international conference and workshops on automatic face and gesture recognition (FG)*, pages 1–8. IEEE, 2013.
- [167] Greg Miller. Data from a century of cinema reveals how movies have evolved, Jun 2017.
- [168] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

- [169] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- [170] Hossein Mobahi, Ronan Collobert, and Jason Weston. Deep learning from temporal coherence in video. In *Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09*, pages 737–744, New York, NY, USA, 2009. ACM, ACM.
- [171] Tobias Nageli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, 36(4):132, 2017.
- [172] Tobias Nageli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics*, 2017.
- [173] Tobias Nageli, Lukas Meier, Alexander Domahidi, Javier Alonso-Mora, and Otmar Hilliges. Real-time planning for automated multi-view drone cinematography. *ACM Transactions on Graphics (TOG)*, 36(4):1–10, 2017.
- [174] Sameer A Nene, Shree K Nayar, Hiroshi Murase, et al. Columbia object image library (coil-20). 1996.
- [175] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J Davison, Pushmeet Kohi, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on*, pages 127–136. IEEE, 2011.
- [176] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.
- [177] Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, JK Aggarwal, Hyungtae Lee, Larry Davis, et al. A large-scale benchmark dataset for event recognition in surveillance video. In *Computer vision and pattern recognition (CVPR), 2011 IEEE conference on*, pages 3153–3160. IEEE, 2011.
- [178] Helen Oleynikova, Michael Burri, Zachary Taylor, Juan Nieto, Roland Siegwart, and Enric Galceran. Continuous-time trajectory optimization for online uav replanning. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5332–5339. IEEE, 2016.
- [179] Helen Oleynikova, Zachary Taylor, Marius Fehr, Juan Nieto, and Roland Siegwart. Voxblox: Building 3d signed distance fields for planning. *arXiv*, pages arXiv–1611, 2016.
- [180] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J Andrew Bagnell, Pieter Abbeel, Jan Peters, et al. An algorithmic perspective on imitation learning. *Foundations and Trends® in Robotics*, 7(1-2):1–179, 2018.

- [181] Bryan Penin, Paolo Robuffo Giordano, and François Chaumette. Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions. *IEEE Robotics and Automation Letters*, 3(4):3725–3732, 2018.
- [182] Nicholas T Proferes. *Film Directing Fundamentals: see your film before shooting*. Taylor & Francis, 2017.
- [183] Pablo Pueyo, Eric Cristofalo, Eduardo Montijano, and Mac Schwager. Cinemairsim: A camera-realistic robotics simulator for cinematographic purposes. *arXiv preprint arXiv:2003.07664*, 2020.
- [184] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.
- [185] Rahul Raman, Pankaj Kumar Sa, Banshidhar Majhi, and Sambit Bakshi. Direction estimation for pedestrian monitoring system in smart cities: an hmm based approach. *IEEE Access*, 4:5788–5808, 2016.
- [186] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554, 2015.
- [187] Nathan Ratliff, Matthew Zucker, J Andrew Bagnell, and Siddhartha Srinivasa. CHOMP: Gradient optimization techniques for efficient motion planning. In *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*, pages 489–494. IEEE, 2009.
- [188] Nathan D Ratliff, David Silver, and J Andrew Bagnell. Learning to search: Functional gradient techniques for imitation learning. *Autonomous Robots*, 27(1):25–53, 2009.
- [189] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.
- [190] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.
- [191] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic back-propagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.
- [192] Charles Richter, Adam Bry, and Nicholas Roy. Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments. In *Robotics Research*, pages 649–666. Springer, 2016.
- [193] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European Conference on Computer Vision workshop on Benchmarking Multi-Target Tracking*, 2016.
- [194] Mike Roberts, Debadeepta Dey, Anh Truong, Sudipta Sinha, Shital Shah, Ashish Kapoor, Pat Hanrahan, and Neel Joshi. Submodular trajectory optimization for aerial

- 3d scanning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5324–5333, 2017.
- [195] Mike Roberts and Pat Hanrahan. Generating dynamically feasible trajectories for quadrotor cameras. *ACM Transactions on Graphics (TOG)*, 35(4):61, 2016.
 - [196] Mike Roberts and Pat Hanrahan. Generating dynamically feasible trajectories for quadrotor cameras. *ACM Transactions on Graphics (SIGGRAPH 2016)*, 35(4), 2016.
 - [197] ROS. Robot operating system (ros), 2018.
 - [198] ROS. Robot operating system (ros), 2018.
 - [199] Stephane Ross, Narek Melik-Barkhudarov, Kumar Shaurya Shankar, Andreas Wendel, Debadeepta Dey, J. Andrew (Drew) Bagnell, and Martial Hebert . Learning monocular reactive uav control in cluttered natural environments. In *IEEE International Conference on Robotics and Automation*. IEEE, March 2013.
 - [200] James A Russell. A circumplex model of affect. *Journal of personality and social psychology*, 39(6):1161, 1980.
 - [201] Philip A Russell and Colin D Gray. Ranking or rating? some data and their implications for the measurement of evaluative response. *British journal of Psychology*, 85(1):79–92, 1994.
 - [202] Fereshteh Sadeghi and Sergey Levine. (CAD)²RL: Real single-image flight without a single real image. *arXiv preprint arXiv:1611.04201*, 2016.
 - [203] Ahmed Saeed, Ahmed Abdelkader, Mouhyemen Khan, Azin Neishaboori, Khaled A Harras, and Amr Mahmoud Salem Mohamed. On realistic target coverage by autonomous drones. *ACM Transactions on Sensor Networks*, 2019.
 - [204] Virginia Santamarina-Campos and Marival Segarra-Oña. Introduction to drones and technology applied to the creative industry. airt project: An overview of the main results and actions. In *Drones and the Creative Industry*, pages 1–17. Springer, 2018.
 - [205] Davide Scaramuzza, Michael C Achtelik, Lefteris Doitsidis, Fraundorfer Friedrich, Elias Kosmatopoulos, Agostino Martinelli, Markus W Achtelik, Margarita Chli, Savvas Chatzichristofis, Laurent Kneip, et al. Vision-controlled micro flying robots: from system design to autonomous navigation and mapping in gps-denied environments. *IEEE Robotics & Automation Magazine*, 21(3):26–40, 2014.
 - [206] John Schulman, Jonathan Ho, Alex Lee, Ibrahim Awwal, Henry Bradlow, and Pieter Abbeel. Finding locally optimal, collision-free trajectories with sequential convex optimization. In *Robotics: Science and Systems*, volume 9, pages 1–10. Citeseer, 2013.
 - [207] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.
 - [208] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and service robotics*, pages 621–635. Springer, 2018.

- [209] Leonid Sigal, Moshe Mahler, Spencer Diaz, Kyna McIntosh, Elizabeth Carter, Timothy Richards, and Jessica Hodgins. A perceptual control space for garment simulation. *ACM Transactions on Graphics (TOG)*, 34(4):1–10, 2015.
- [210] Amarjeet Singh, Andreas Krause, Carlos Guestrin, and William J Kaiser. Efficient informative sensing using multiple robots. *Journal of Artificial Intelligence Research*, 34:707–755, 2009.
- [211] Skydio. Skydio r1 self-flying camera, <https://www.skydio.com/technology/>, 2018.
- [212] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.
- [213] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–98, 2018.
- [214] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.
- [215] Robert Henry. Stanley. *The movie idiom: film as a popular art form*. Waveland Press, Inc., 2011.
- [216] David Stavens and Sebastian Thrun. Unsupervised learning of invariant features using video. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1649–1656. IEEE, 2010.
- [217] Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pages 212–223. Springer, 2002.
- [218] Stephan Streuber, M Alejandra Quiros-Ramirez, Matthew Q Hill, Carina A Hahn, Silvia Zuffi, Alice O’Toole, and Michael J Black. Body talk: Crowdshaping realistic 3d avatars with words. *ACM Transactions on Graphics (TOG)*, 35(4):1–14, 2016.
- [219] Elias Strigel, Daniel Meissner, Florian Seeliger, Benjamin Wilking, and Klaus Dietmayer. The ko-per intersection laserscanner and video dataset. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1900–1901. IEEE, 2014.
- [220] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of rgb-d slam systems. In *IROS*, pages 573–580. IEEE, 2012.
- [221] Yu-Chuan Su, Dinesh Jayaraman, and Kristen Grauman. Pano2vid: Automatic cinematography for watching 360 videos. In *Asian Conference on Computer Vision*, pages 154–171. Springer, 2016.
- [222] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [223] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

- [224] Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.
- [225] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 31–36. IEEE, 2017.
- [226] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.
- [227] Renran Tian, Lingxi Li, Kai Yang, Stanley Chien, Yaobin Chen, and Rini Sherony. Estimation of the vehicle-pedestrian encounter/conflict risk on the road based on tasi 110-car naturalistic driving data collection. In *Intelligent Vehicles Symposium Proceedings, 2014 IEEE*, pages 623–629. IEEE, 2014.
- [228] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.
- [229] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 23–30. IEEE, 2017.
- [230] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1653–1660, 2014.
- [231] Yuri Tsivian. Cinemetrics, part of the humanities’ cyberinfrastructure. 2009.
- [232] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, MN Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.
- [233] Marynel Vázquez, Aaron Steinfeld, and Scott E Hudson. Parallel detection of conversational groups of free-standing people and tracking of their lower-body orientation. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 3010–3017. IEEE, 2015.
- [234] Felipe Patiño Vista, Deok-Jin Lee, and Kil To Chong. Design of an ekf-ci based sensor fusion for robust heading estimation of marine vehicle. *International Journal of Precision Engineering and Manufacturing*, 16(2):403–407, 2015.
- [235] Chao Wang, Yongkun Fang, Huijing Zhao, Chunzhao Guo, Seiichi Mita, and Hongbin Zha. Probabilistic inference for occluded and multiview on-road vehicle detection. *IEEE Transactions on Intelligent Transportation Systems*, 17(1):215–229, 2016.
- [236] Wenshan Wang, Aayush Ahuja, Yanfu Zhang, Rogerio Bonatti, and Sebastian Scherer. Improved generalization of heading direction estimation for aerial filming using semi-supervised regression. *Robotics and Automation (ICRA), 2019 IEEE International Conference on*, 2019.

- [237] Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2794–2802. IEEE Computer Society, 2015.
- [238] Jason Weston, Frédéric Ratle, Hossein Mobahi, and Ronan Collobert. Deep learning via semi-supervised embedding. In *Neural Networks: Tricks of the Trade*, pages 639–655. Springer, 2012.
- [239] Hui-Yin Wu, Francesca Palù, Roberto Ranon, and Marc Christie. Thinking like a director: Film editing patterns for virtual cinematographic storytelling. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 14(4):81, 2018.
- [240] Yi Wu, Yuxin Wu, Georgia Gkioxari, and Yuandong Tian. Building generalizable agents with a realistic and rich 3d environment. *arXiv preprint arXiv:1801.02209*, 2018.
- [241] Ke Xie, Hao Yang, Shengqiu Huang, Dani Lischinski, Marc Christie, Kai Xu, Minglun Gong, Daniel Cohen-Or, and Hui Huang. Creating and chaining camera moves for quadrotor videography. *ACM Transactions on Graphics*, 37:14, 2018.
- [242] Yi-Hsuan Yang and Homer H Chen. Ranking-based emotion recognition for music organization and retrieval. *IEEE Transactions on audio, speech, and language processing*, 19(4):762–774, 2010.
- [243] Luke Yoder and Sebastian Scherer. Autonomous exploration for infrastructure modeling with a micro aerial vehicle. In *Field and service robotics*, pages 427–440. Springer, 2016.
- [244] Mehmet Ersin Yumer, Siddhartha Chaudhuri, Jessica K Hodgins, and Levent Burak Kara. Semantic shape editing using deformation handles. *ACM Transactions on Graphics (TOG)*, 34(4):1–12, 2015.
- [245] Fangyi Zhang, Jürgen Leitner, Zongyuan Ge, Michael Milford, and Peter Corke. Adversarial discriminative sim-to-real transfer of visuo-motor policies. *The International Journal of Robotics Research*, 38(10-11):1229–1245, 2019.
- [246] Yanfu Zhang, Wenshan Wang, Rogerio Bonatti, Daniel Maturana, and Sebastian Scherer. Integrating kinematics and environment context into deep inverse reinforcement learning for predicting off-road vehicle trajectories. In *Conference on Robot Learning*, pages 894–905, 2018.
- [247] Weikun Zhen, Yaoyu Hu, Jingfeng Liu, and Sebastian Scherer. A joint optimization approach of lidar-camera fusion for accurate dense 3-d reconstructions. *IEEE Robotics and Automation Letters*, 4(4):3585–3592, 2019.
- [248] Xiaocui Zheng, Fei Wang, and Zhanghua Li. A multi-uav cooperative route planning methodology for 3d fine-resolution building model reconstruction. *ISPRS journal of photogrammetry and remote sensing*, 146:483–494, 2018.
- [249] Jun-Yan Zhu, Zhoutong Zhang, Chengkai Zhang, Jiajun Wu, Antonio Torralba, Joshua B. Tenenbaum, and William T. Freeman. Visual object networks: Image generation with disentangled 3d representation. *CoRR*, abs/1812.02725, 2018.

- [250] Will Zou, Shenghuo Zhu, Kai Yu, and Andrew Y. Ng. Deep learning of invariant features via simulated fixations in video. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 3203–3211. Curran Associates, Inc., 2012.
- [251] Matt Zucker, Nathan Ratliff, Anca D Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher M Dellin, J Andrew Bagnell, and Siddhartha S Srinivasa. CHOMP: Covariant hamiltonian optimization for motion planning. *The International Journal of Robotics Research*, 32(9-10):1164–1193, 2013.