Towards Self-supervised Object Discovery and Tracking

Yiming Zuo

CMU-RI-TR-21-28

July 12, 2021

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

Thesis Committee:
Katerina Fragkiadaki, *chair*
Kris M. Kitani
Adam Harley

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

Object discovery and multiple object tracking (MOT) are two highly interrelated tasks that are known to be fundamental problems in computer vision, and are crucial for video understanding. Most existing methods rely on supervised training with human annotations, which is laborious and expensive. In this thesis, we propose a self-supervised method for detecting and tracking moving objects in unlabelled RGB-D videos. The method begins with classic handcrafted techniques for segmenting objects using motion cues: we estimate optical flow and camera motion, and conservatively segment regions that appear to be moving independently of the background. Treating these initial segments as pseudo-labels, we learn an ensemble of appearance-based 2D and 3D detectors, under heavy data augmentation. We use this ensemble to detect new instances of the "moving" type, even if they are not moving, and add these as new pseudo-labels. Our method is an expectation-maximization algorithm, where in the expectation step we fire all modules and look for agreement among them, and in the maximization step we re-train the modules to improve this agreement. The constraint of ensemble agreement helps combat contamination of the generated pseudo-labels (during the E step), and data augmentation helps the modules generalize to yet-unlabelled data (during the M step). We compare against existing unsupervised object discovery and tracking methods, using challenging videos from CATER and KITTI, and show strong improvements over the state-of-the-art.

Acknowledgments

I would like to give my greatest thanks to my thesis committee members, Prof. Katerina Fragkiadaki, Prof. Kris Kitani and Adam Harley. I would like to thank Katerina for all her guidance and support over the past two years. I have learnt a lot from her on how to be a good researcher. I would also like to thank Adam for his generous support. He has been giving me advice from all aspects that helps me grow, from coding style to paper writing, from how to write an good email to how to present my work. I would also like to thank Prof. Kitani for joining my committee and for all the valuable feedback from him.

I want to give special thanks to my lab members, especially Jing Wen and Adam Harley, who has been working on the same research project with me for the last two years. It has been a great time cooperating with them, and I appreciate their help a lot.

I would also like to thank my family members for their support without reservation.

Finally, I would like to thank my friends, both in Pittsburgh and in China, for bringing me so much joy and support during the COVID-19 pandemic.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Object discovery and object tracking are known to be crucial problems in computer vision, and are particularly useful for video understanding. Most existing methods rely on supervised training with human annotations, and gaining dense labels on videos could be laborious and expensive. On the contrary, humans can detect moving objects and delineate their approximate extent [12, 53], without ever having been supplied boxes or segmentation masks as supervision. In the recent computer vision literature, a variety of self-supervised methods [8, 36] are recently proposed to solve the object discovery and tracking problem, which hinge on reconstruction objectives and part-centric, object-centric, or scene-centric bottlenecks in the architecture. These methods are rapidly advancing, but so far only on toy worlds, made up of simple 2D or 3D shapes against simple backgrounds – a far cry from the complexity tackled in older works, based on perceptual grouping (e.g., [17]).

In this thesis, we propose to solve self-supervised object discovery and tracking problem using a variety of perceptual grouping cues, which make some regions look more object-like than others [22]. Work on integrating perceptual grouping cues into computer vision models stretches back decades [50], and still likely serves as inspiration for many of the design decisions in modern computer vision architectures related to attention, segmentation, and tracking.

Classic methods of object discovery, such as center-surround saliency in color or flow [1], are known to be brittle, but they need not be discarded entirely. We propose to mine and exploit the admittedly rare success scenarios of these models,

to bootstrap the learning of something more general. We hypothesize that if the successful vs. unsuccessful runs of the classic algorithms can be readily identified with automatic techniques, then we can self-supervise a learning-based module to mimic and outperform the traditional methods. This is a kind of knowledge distillation [27], from traditional models to deep ones.

We propose an optimization algorithm for learning detectors of moving objects, based on expectation maximization [44]. We begin with a motion-based handcrafted detector, tuned to be very conservative (low recall, high precision). We then convert each object proposal into thousands of training examples for learning-based 2D and 3D detectors, by randomizing properties like color, scale, and orientation. This forces the learned models to generalize, and allows recall to expand. We then use the ensemble of modules to obtain new high-confidence estimates **(E step)**, repeat the optimization **(M step)**, and iterate. Our method outperforms not only the traditional methods, which only work under specific conditions, but also the current learning-based methods, which only work in toy environments. We demonstrate success in a popular synthetic environment where recent deep models have already been deployed (CLEVR/CATER [21, 31]), and also on the real-world urban scenes benchmark (KITTI [20]), where the existing learned models fall flat.

Our main contribution is not in any particular component, but rather in their combination. We demonstrate that by exploiting the successful outcomes of traditional methods for moving object segmentation, we can train a learning-based method to detect and track objects in a target domain, without requiring any annotations.

Chapter 2

Background

2.1 Object discovery

Many recent works have proposed deep neural networks for object discovery from RGB videos. These models typically have an object-centric bottleneck, and are tasked with a reconstruction objective. MONet [8], Slot attention [40], IODINE [23] , SCALOR [30], AIR [16], and AlignNet [13] fall under this category. These methods have been successful in a variety of simple domains, but have not yet been tested on real-world videos. In this thesis we evaluate whether these models are able to perform well under the complexities of real-world imagery.

2.2 Ensemble methods

Using ensembles is a well-known way to improve overall performance of an algorithm. Assuming that each member of the ensemble is prone to different types of errors, the combination of them is likely to make fewer errors than any individual component [14]. Ensembling is also the key idea behind knowledge distillation, where knowledge gets transferred from cumbersome models to simpler ones [27, 48]. A typical modern setup is to make up the ensemble out of multiple copies of a neural network, which are trained from different random initializations or using different partitions of the data [2]. In our case, the ensemble is more diverse: it is made up of components which aim

to solve different tasks, but can still be checked against one another for consistency. For example, we learn a 2D pixel labeller which operates on RGB images, and a 3D object detector which operates on voxelized pointclouds; when the 3D detections are projected into the image, we expect them to land on "object" pixels.

2.3 Structure-from-Motion/SLAM

Early works on structure from motion (SfM) [11, 56] set the ambitious goal of extracting unscaled 3D scene pointclouds and camera trajectories from 2D pixel trajectories, exploiting the reduced rank of the trajectory matrix under rigid motions. Unfortunately, these methods are often confined to very simple videos, due to their difficulty handling camera motion degeneracies, non rigid object motion, or frequent occlusions, which cause 2D trajectories to be short in length. Simultaneous Localization And Mapping (SLAM) methods optimize the camera poses in every frame as well as the 3D coordinates of points in the scene online and often in real time, assuming a calibrated setup (i.e., knowing camera intrinsics) [33, 51]. These methods are sensitive to measurement noise and the difficulties of multi-view correspondence, but produce accurate reconstructions when assumptions on the sensor and scene setup are met. Dynamic objects are typically treated as outliers [32, 60], or are actively detected with the help of optical flow [10] or pre-trained appearance cues [3]. Our method exploits the occasional successes of a flow-based egomotion estimation method as a starting point to learn about the static vs. moving parts of scenes.

2.4 Moving object segmentation

Early approaches attempted to solve motion segmentation completely in an unsupervised manner by integrating motion information over time through 2D pixel trajectories [7, 45]. Recent works instead focus on learning to segment 2D objects in videos, supervised by annotated video benchmarks [5, 19, 28, 37, 47]. Here, we segment a sparse set of objects using motion cues, then learn to segment in 2D and 3D using appearance, without annotations.

2.5 Learning from augmentations

The state of the art approach in self-supervised learning of 1D visual representations (i.e., vectors describing images) relies on training the features to be invariant to random augmentations, such as color jittering and random cropping [9, 26]. Some tracking approaches use data augmentation at test time, to fine-tune the tracker with diverse variations of a specified target [34]. Interestingly, the most important factor seems to be high diversity, rather than realism, even for practical robotics applications [55]. Our work takes inspiration from these methods, to upgrade *self-generated* annotations into data with high diversity through data augmentation.

Chapter 3

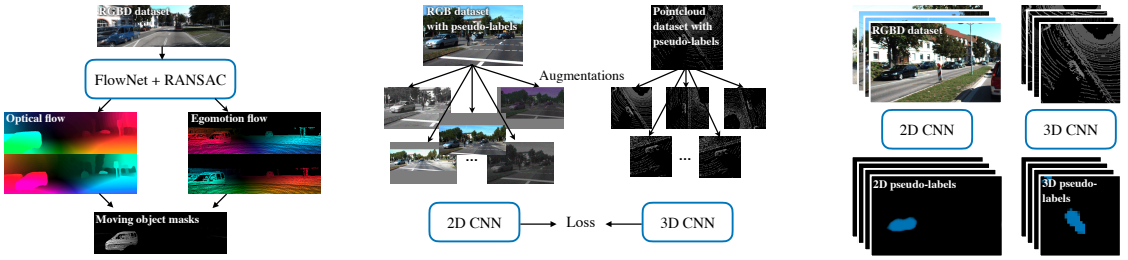
Approach

3.1 Setup and overview

Our method takes as input a video with RGB and depth (either a depthmap or a pointcloud) and camera intrinsics, and produces as output a 3D detector and 3D tracker for objects in the video.

Given an unlabeled input video, the optimization of the model operates in "rounds". The first round leverages optical flow and cycle-consistency constraints to discover a small number of high-confidence, clearly-moving objects in the videos. The most confident object proposals are upgraded into pseudolabels for training appearance-based object detectors, in 2D and 3D. The second round leverages optical flow and the new objectness detectors to find more high-confidence proposals, which again lead to additional training. In the final round we use the detectors as trackers, and use these to generate a library of trajectories, capturing a motion prior for objects in the domain.

A critical piece in each stage is the "check", which decides whether or not to promote an estimate into a pseudolabel for the next round. We now describe each piece of the method, along with its corresponding check.



Input: Unlabelled RGB-D dataset
Method: Find independently-moving segments
Output: Pseudo-labelled dataset
(a) Start (initial E step)

Input: Pseudo-labelled dataset
Method: Apply augmentations; train CNNs
Output: Trained object detectors
(b) M step

Input: Trained object detectors, unlabelled data
Method: Run CNNs; seek ensemble agreement
Output: Pseudo-labelled dataset
(c) E step

Figure 3.1: **An EM approach to unsupervised tracking.** We present an expectation-maximization (EM) method, which takes RGBD videos as input, and produces object detectors and trackers as output. **(a)** We begin with a handcrafted E step, which uses optical flow and egomotion to segment a small number of objects moving independently from the background. **(b)** Next, as an M step, we treat these segmented objects as pseudo-labels, and train 2D and 3D convolutional nets to detect these objects under heavy data augmentation. **(c)** We then use the learned detectors as an ensemble to re-label the data (E step), and loop.

3.2 Optical flow estimation

Optical flow indicates a 2D motion field that corresponds the pixels of a pair of images. We use flow (in combination with other cues) as a signal for objectness, and also as a submodule of egomotion estimation.

We use an off-the-shelf pre-trained convolutional optical flow network [54] to estimate flow. The pre-training involves dense supervision with synthetic frame pairs, but we note that optical flow can be learned unsupervised on real data [39, 61]. In our experiments, we found that the pre-trained model generalized well, and self-supervised finetuning did not improve accuracy further.

For our purposes, it is important to avoid relying on flow vectors which are likely to be incorrect. To do this, we check for forward-backward consistency [46, 52, 59]. We first generate the forward flow $f_{0\to1}$ and the backward flow $f_{1\to0}$ with the flow network, and then warp the backward flow into the coordinates of the first image,

$$\hat{f}_{1\to0} = \text{warp}(f_{1\to0}; f_{0\to1}). \tag{3.1}$$

This uses a bilinear warping function, which takes as arguments an image to warp (in this case $f_{1\to0}$), and a flow field to warp with (in this case $f_{0\to1}$). We then threshold on the size of the discrepancy between flow fields, using a threshold that is sensitive to the flow magnitude:

$$||f_{0\to1} + \hat{f}_{1\to0}||_2 < \alpha_1 \left(||f_{0\to1}||_2 + ||\hat{f}_{1\to0}||_2 \right) + \alpha_2. \tag{3.2}$$

Note that the discrepancy term uses the *addition* of the flow fields rather than the difference, since the flows point in opposite directions. We use $\alpha_1 = 0.01$ and $\alpha_2 = 0.1$ in our experiments.

3.3 Egomotion estimation

Egomotion is the rigid motion of the camera (i.e., transformation of poses) across a pair of frames. Estimating egomotion allows us to better estimate which pixels are moving due to the camera's motion, and which are moving independently of the background. Pixels moving independently are a strong cue for objectness.

We begin by "upgrading" the cycle-consistent 2D flows into a sparse 3D pointcloud flow. To do this, we first obtain sparse 2D depth maps, by projecting the pointclouds into pixel coordinates. We then check each flow vector to see if it starts and ends at pixels with depth measurements. Finally, we use the flows and corresponding depths to un-project the flow into a 3D motion field.

With the sparse 3D motion field, we use RANSAC to estimate the 6-degrees-of-freedom rigid motion that explains the maximal number of point flows. RANSAC is intended to be robust to outliers, but the answer returned is often catastrophically wrong, due either to correspondence errors or moving objects.

The critical third step is to "check" the RANSAC output with a freely-available signal. The inlier count itself is such a signal, but this demands carefully tuning the threshold for inlier counting. Instead, we enforce cycle-consistency, similar to flow. We estimate two rigid motions with RANSAC: once using the forward flow, and once using the backward flow (which delivers an estimate of the inverse transform, or backward egomotion). We then measure the inconsistency of these results, by applying the forward and backward motion to the *same* pointcloud, and measuring

the maximum alignment error:

$$x'_0 = T^{bw}_{1\to 0} T^{fw}_{0\to 1}(x_0) \tag{3.3}$$

$$\text{error} = \max_n(||x'_0 - x_0||), \tag{3.4}$$

where $T^{fw}_{0\to 1}$ denotes the rotation and translation computed from forward flow, which carries the pointcloud from timestep 0 to timestep 1, $T^{bw}_{1\to 0}$ is the backward counterpart, and x_0 denotes the pointcloud from timestep 0.

If the maximum displacement across the entire pointcloud is below a threshold (set to 0.25 meters), then we treat the estimate as "correct". In practice we find that this occurs about 80% of the time in the KITTI dataset.

On these successful runs, we apply the egomotion to the pointcloud to create another "background" 3D flow field (in addition to the "full" flow field produced by upgrading the optical flow to 3D), and we subtract these to obtain the camera-independent motion field. Independently moving objects produce high-magnitude regions in the egomotion-stabilized motion field, which is an excellent cue for objectness. Examples of this are shown in Figure 3.1-a and Figure 3.2-c: note that although real objects are highlighted by this field, some spurious background elements are highlighted also.

In the first "round" of optimization, we proceed directly from this stage to pseudo-label generation. Using the pseudo-labels, we train the parameters of two object detectors, described next.

3.4 2D objectness segmentation

This module takes an RGB image as input, and produces a binary map as output. The intent of the binary map is to estimate the likelihood that a pixel belongs to the "moving object" class. This module transfers knowledge from the motion-based estimators into the domain of appearance, since it learns to mimic pseudolabels that were generated from motion alone. This is an important aspect of the overall model, since it allows us to identify objects of the "moving" type even when they are stationary (e.g., a vehicle parked on the side of the road).

We use a 50-layer ResNet [25] with a feature pyramid [38] as the architecture,

and train the last layer with a logistic loss against sparse pseudo-ground-truth:

$$\mathcal{L}^{\text{seg}} = \sum \hat{m} \log(1 + \exp(-\hat{s} \cdot s)), \tag{3.5}$$

where m is a mask indicating where the supervision is valid. How the mask is generated is detailed in section 3.9. We experimented with and without ImageNet pretraining for the ResNet, and found that the pretrained version converges more quickly but does not perform very differently.

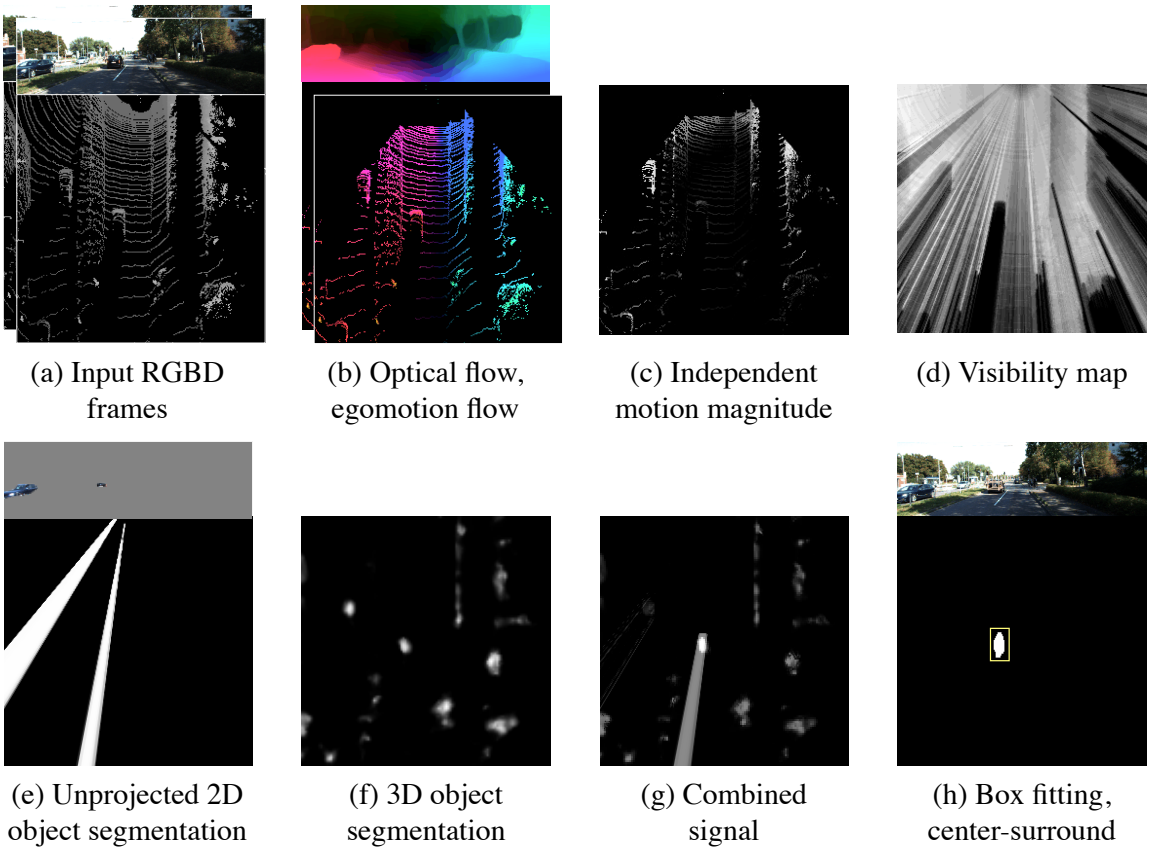
In training this module with sparse labels, it is critical to add heavy augmentations to the input, so that it does not simply memorize a mapping from the happenstance appearance to the sparse objectness labels. We use random color jittering, random translation and scaling, and random synthetic occlusions.

3.5 3D object detection

This module takes as input a voxelized pointcloud (computed from the depth map and intrinsics), and estimates object proposals in 3D. We have experimented with producing oriented 3D bounding boxes, or 3D voxel segmentations, with similar results.

We use a 3D U-Net-style convolutional encoder [49], and a CenterNet-style detection head [15]. The head produces a set of heatmaps, which encode objectness (in 1 channel), 3D size (in 3 channels), 3D subvoxel offset (in 3 channels), and orientation along the vertical axis (encoded as a categorical distribution over 16 channels).

In training this module, we find that randomized translation and orientation (when creating the voxelized input) are critical for learning an even distribution over possible object orientations. Additionally, we apply dropout in the voxel inputs, and we create partial occlusion augmentations by randomly erasing a randomly-sized area near the object pseudo-label in image space, along with the 3D points that project into that area.



(a) Input RGBD frames | (b) Optical flow, egomotion flow | (c) Independent motion magnitude | (d) Visibility map

(e) Unprojected 2D object segmentation | (f) 3D object segmentation | (g) Combined signal | (h) Box fitting, center-surround

Figure 3.2: **Intermediate outputs of the *Expectation* step in our algorithm.** Given an input video (a), we have multiple sources of evidence: using optical flow and egomotion flow (b), we compute independent motion magnitude (c); using the pointcloud we compute visible area (d), using the RGB image we estimate 2D segmentation (e), and using the pointcloud we estimate 3D segmentation (f). Each signal can be error-prone, but combining them (g) gives us high confidence pseudo-labels (h).

3.6 Short-range tracking

To relocate a detected object over short time periods, we use two simple techniques: hungarian matching with IoU scores, and cross correlation with a rigid template [41]. We find that the IoU method is sufficient in CATER, where the motions are relatively slow. In KITTI, due to the fast motions of the objects and the additional camera motion, we find that cross-correlation is more effective. We do this using the features

provided by the backbone of the object detector. We simply create a template by encoding a crop around the object, and then use this template for 3D cross correlation against features produced in nearby frames, extracted using a Siamese network. We find that this is a surprisingly effective tracker despite not handling rotations, likely because the objects do not undergo large rotations under short timescales. To track for longer periods and across occlusions, we make use of motion priors represented in a library of previously-observed motions, described next.

3.7 Long-range tracking, with trajectory libraries

To track objects over longer time periods, we build and use a library of motion trajectories, to act as a motion prior. We build the library out of the successful outcomes of short-range tracker, which typically correspond to "easy" tracking cases, such as close-range objects will full visibility. The key insight here is that a motion prior built from "good visibility" tracklets is just as applicable to "poor visibility tracklets", since visibility is not a factor in objects' motion.

To verify tracklets and upgrade them into library entries, we check if they agree with the per-timestep cues, provided by flow, 2D segmentation, 3D object detection, and a visibility map computed by raycasting on the pointcloud (Fig 3.2). Specifically, we ask that a tracklet (1) obey the flow field, and (2) travel through area that is either object-like or invisible. For flow agreement, we simply project the 3D object motion to 2D and measure the inconsistency with the 2D flow in the projected region. To ensure that the trajectory travels through object-like territory, we create a spatiotemporal volume of objectness/visiblity cues, and trilinearly sample in that volume at each timestep along the trajectory. Each temporal slice of the volume is given by:

$$p = \max(\text{unproj}(s) \cdot o + (1.0 - v), 1), \tag{3.6}$$

where unproj(s) is the 2D segmentation map unprojected to 3D (Figure 3.2-d), o is the 3D heatmap delivered by the object detector (Figure 3.2-f), and v is the visibility map computed through raycasting (Figure 3.2-d). In other words, we require that both the 2D and 3D objectness signals agree on the object's presence, or that the visibility indicates the object is in an occluded area. To evaluate a trajectory's

likelihood, we simply take the mean of its values over the temporal dimension, and we set a stringent threshold (0.99) to prevent erroneous tracklets from entering the library.

Once the library is built, we use it to link detections across partial and full occlusions (where flow-based and correlation-based tracking fails). Specifically, we orient the library to the initial motion of an object, and then evaluate the likelihood of all paths in the library, via the cost volume. This is similar to a recent approach for motion planning for self-driving vehicles [62], but here the set of possible trajectories is generated from data rather than handcrafted.

3.8 Pseudo-label generation

Pseudo-label generation is what takes the model from one round of optimization to the next. The intent is to select the object proposals that are likely to be correct, and treat them as ground truth for training future modules.

We take inspiration from never-ending learning architectures [43], which promote an estimate into a label only if (i) at least one module produces exceedingly-high confidence in the estimate, or (ii) multiple modules have reasonably-high confidence in the estimate.

The 2D and 3D modules directly produce objectness confidences, but the motion cues need to be converted into an objectness cue. Our strategy is inspired by classic literature on motion saliency [29]: (1) compute the magnitude of the egomotion-stabilized 3D motion field, (2) threshold it at a value (to mark regions with motion larger than some speed), (3) find connected components in that binary map (to obtain discrete regions), and (4) evaluate the center-surround saliency of each region. Specifically, we compute histograms of the motion inside the region and in the surrounding shell, compute the chi-square distance between the distributions, and threshold on this value [1]:

$$cs(\theta) = \chi^2(h(\text{cen}_\theta(\Delta x)), h(\text{surr}_\theta(\Delta x))), \tag{3.7}$$

where θ denotes the region being evaluated, cen_θ and surr_θ select points within and surrounding the region, h computes a histogram, and Δx denotes the egomotion-

stabilized 3D motion field.

When the trained objectness detectors are available (i.e., on rounds after the first), we convert the egomotion-stabilized motion field into a heatmap with $\exp(-\lambda||\Delta x||)$, and add this heatmap to the ones produced by the 2D and 3D objectness estimators. We then proceed with thresholding, connected components, and box fitting as normal. The only difference is that we use a threshold of 2 to demand multiple modules to agree.

3.9 Self-supervision details

After converting estimates into pseudolabels, it is important to design the supervision strategy so that the appearance-based objectness modules have the potential to generalize from this data to the yet-unannotated data. This involves (1) heavy augmentations, and (2) converting the pseudolabels into supervision regions:

- **Positive**: regions where we have detected a moving object;
- **Negative**: regions where a moving object is unlikely;
- **Ignore**: regions where the pseudolabels are ambiguous about moving/non-moving.

The "ignore" region is critical: If we simply used the pseudolabels as positives and treated the remainder of the scene as negatives, the modules would effectively be trained to *not* detect objects beyond the annotated ones. The design of the regions is slightly different for the 2D segmentor vs. the 3D object detector.

For 2D objectness segmentation, we need to annotate pixels. To create positive pixels, we shrink the estimated bounding boxes (by 0.1m on each side), then collect pointcloud points that are within those boxes, and project those points into the image. To create negative pixels, we enlarge the boxes (by 2.0m on each side), and collect points that are between the original-sized box and the enlarged box, and project those into the image. Additionally, we project the box itself into the image, and use its outer contour as negative pixels. We leave all other pixels as "ignore", and do not apply loss there. These supervision labels are illustrated in Figure 3.3.

For 3D detection, we annotate voxels on a 3D grid. Since we use a CenterNet-style detector [15], we do not need to annotate anchor boxes as positives or negatives,

but instead create a "target" objectness heatmap (indicating positives and negatives densely), with a small Gaussian at the centroid of each annotated object. At the peak of the Gaussian, we supply the subvoxel offset and orientation information, in the additional channels. As in the original CenterNet, we only supervise offset and orientation at the centroid itself. To avoid penalizing detections in the unannotated part of the scene, we treat all voxels that lie beyond a radius of an annotation as part of the "ignore" region, and do not apply loss there. We set the radius according to the annotation, using a value of $r = 3 \cdot \max(l, h, w)$, where l, h, w denote the length, height, and width of the annotated object.

Within each batch, we evenly balance the loss induced by positive labels and negative labels, to ensure that the model does not learn a frequency bias for either class.

3.10 Connection to EM

The mathematical connection to Expectation-Maximization (EM) is not perfect, because our model is not generative, but the mapping is quite close. Following the terminology of Nigam et al. [44], we have:

E step: ***Use the current classifier to estimate the class membership for each datapoint.*** In our case, use the ensemble of handcrafted and learned modules to estimate the objectness probability for each pixel/point.

M step: ***Re-estimate the classifier, using the estimated class labels.*** In our case, re-optimize the parameters of the learned components of the ensemble, using estimates produced by the ensemble as a whole (i.e., where agreement was reached).

Note that if our classifier were a *single* discriminative model, it would theoretically converge after the first M step; using an *ensemble* of independent modules allows our method to improve over rounds, since it will not converge until all submodules produce the same labelling.

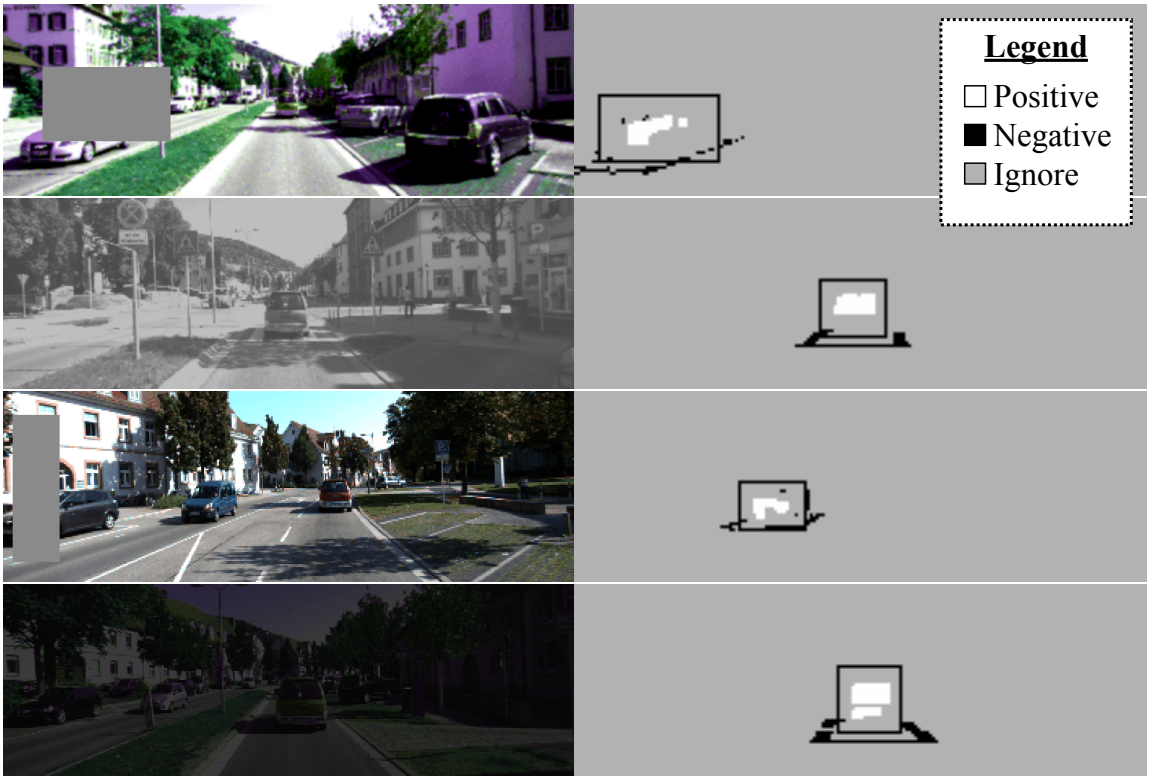


Figure 3.3: **Sparse supervision for 2D objectness segmentation.** Left: RGB images with color and occlusion augmentations. Right: supervision generated from pseudolabels. Supervision is only generated in the region immediately surrounding the discovered objects.

Chapter 4

Experiment Results

4.1 Datasets

We evaluate in the following datasets:

1. Synthetic RGB-D videos of tabletop scenes from CATER [21]. CATER is a built upon CLEVR [31], and it focuses on testing a model's ability to do long-term temporal reasoning. We modified the simulator so that it can generate depth maps in addition to RGB images, but leave all other rendering parameters untouched. The max number of objects is set to 10 to make the scenes as complex as possible. Videos are captured by 6 virtual cameras placed around the scene.
2. Real RGB-D videos of urban scenes, from the KITTI dataset [20]. This data was collected with a sensor platform mounted on a moving vehicle, with a human driver navigating through a variety of road types in Germany. The data provides multiple images per timestep; we use the "left" color camera. The dataset provides depth in the form of LiDAR sweeps synced to the images. We use the "tracking" subset of KITTI, which includes 3D object labels, and approximate (but relatively inaccurate) egomotion.

We evaluate the models on their ability to discover objects in 3D, and track objects over time.

4.2 Baselines

For unsupervised object discovery, we use the following baselines:

- **Object-Centric Learning with Slot Attention** [40]. This model trains an autoencoder with image reconstruction loss, with soft clustering assignments as a representational bottleneck. An iterative attention-based update mechanism is applied when constructing the clusters.
- **MONet** [8]. MONet applies a recurrent attention mechanism that tries to explain the scene part by part, with a VAE. Since there is no official code released, we re-implemented it.
- **Spectral Clustering** [6]. This method first extracts dense point trajectories using optical flow, then performs object segmentation by computing affinities between trajectories and then clustering.
- **Discontinuity-Aware Clustering** [18]. This method also begins with dense point trajectories, but uses density discontinuities in the spectral embeddings to improve the segmentation.

For tracking, we use the following baselines:

- **Supervised Siamese Network** [4]. This is a 3D convolutional net trained as a siamese object tracker. This model produces a feature volume for the object at $t = 0$, and produces feature volume for the scene at each timestep, and then locates the object in the wider scene by doing cross correlation at each step. The model is supervised so that the peak of the correlation heatmap is in the correct place.
- **Tracking by Colorizing** [57]. This model learns features through an image reconstruction task. The goal is to colorize a grayscale image, by indexing into a source color image with feature dot products. We upgrade this model into a 3D tracker by associating the learned features to the 3D pointcloud instead of RGB values, and by doing RANSAC on the point-wise correspondences to estimate rigid object motions [24].

4.3 Training details

We optimize the parameters of all CNNs with the Adam optimizer [35], with an effective batch size of 4. To achieve this on smaller GPUs with memory constraints,

we use a batch size of 1 but accumulate gradients from 4 steps before taking a step with the optimizer.

For the 2D segmentor and the 3D object detector, in the first training round, we use random initialization of the networks, and set the learning rate to $1e-4$. In the second training round, we use the first round's parameters as initialization, and train with a learning rate of $1e-5$. The segmentor converges in approximately 40,000 iterations, while the 3D detector requires approximately 80,000 iterations. For the optical flow module, we initialize with the RAFT model trained on FlyingThings [42]. We have also fine-tuned the RAFT model with standard self-supervision objectives (brightness constancy, smoothness, forward-backward consistency), without significant improvement beyond this strong initialization.

In CATER we use an image resolution of 128×384, and a 3D voxel grid resolution of $128 \times 64 \times 128$, spanning a range of $8 \times 4 \times 8$ in the dataset's units. In KITTI we use an image resolution of 128×416, and a 3D voxel grid resolution of $256 \times 32 \times 256$, spanning a metric range of $32m \times 8m \times 32m$.

4.4 Quantitative results

4.4.1 Object discovery

Our main evaluation is in Table 4.1, where we evaluate the object proposal accuracy in mean Average Precision (mAP) at different IoU thresholds, on both CATER and KITTI. The metrics are collected in a bird's-eye view (BEV) and in 2D projections (2D). Adding additional rounds of EM improves the precision at the higher IoU thresholds. We find that our model outperforms the baselines in nearly all metrics. Please see detailed analysis of the baselines' performance in section 4.7.

4.4.2 Object tracking

Object tracking accuracy (in IoU over time) is shown in Figure 4.1. To evaluate tracking, we initialize the cross-correlation based tracker with the bounding box of the object to track.

As shown in Figure 4.1, the supervised model outperforms the unsupervised

Method	Dataset	mAP@X						
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**
Slot Attention [40]	CATER (2D)	0.63	0.51	0.43	0.34	0.22	0.1	0.05
	KITTI (2D)	0.07	0.03	0.01	0	0	0	0
MONet [8]	CATER (2D)	0.23	0.14	0.12	0.10	0.07	0.03	0.01
	KITTI (2D)	0.03	0.01	0	0	0	0	0
Spectral Clustering [6]	CATER (2D)	0.18	0.08	0.04	0.03	0.01	0	0
	KITTI (2D)	0.08	0.03	0.02	0.02	0.01	0	0
Discontinuity Aware Clustering [18]	CATER (2D)	0.17	0.08	0.04	0.02	0.01	0.01	0
	KITTI (2D)	0.08	0.04	0.03	0.01	0	0	0
Ours (Round1)	CATER (2D)	**0.98**	0.97	**0.97**	0.94	0.86	0.7	**0.36**
	KITTI (2D)	**0.53**	0.39	0.18	0.06	0.03	0.01	0.01
	CATER (BEV)	0.97	0.92	0.75	0.57	0.34	0.06	0
	KITTI (BEV)	**0.46**	**0.42**	0.06	0	0	0	0
Ours (Round2)	CATER (2D)	**0.98**	0.97	0.96	0.94	**0.88**	0.69	0.33
	KITTI (2D)	0.43	**0.40**	**0.39**	0.33	0.30	0.22	0.10
	CATER (BEV)	0.97	0.95	0.84	0.66	**0.46**	0.08	0
	KITTI (BEV)	0.41	0.39	**0.35**	0.31	0.28	0.11	0.02
Ours (Round3)	CATER (2D)	**0.98**	**0.98**	**0.97**	**0.95**	**0.88**	**0.71**	0.34
	KITTI (2D)	0.43	**0.4**	0.37	**0.35**	**0.33**	**0.3**	**0.21**
	CATER (BEV)	**0.98**	**0.97**	**0.9**	**0.76**	**0.46**	**0.1**	**0.02**
	KITTI (BEV)	0.4	0.38	**0.35**	**0.33**	**0.31**	**0.23**	**0.06**

Table 4.1: **Object discovery performance, in CATER and KITTI.** Results are reported as mean average precision (mAP) at several IoU thresholds. Our method works best in all the metrics reported. 2D means perspective view and BEV means bird's-eye view.

ones, especially in CATER (where the data and supervision are perfect), and by a narrower margin in KITTI. Our method can maintain relatively high IoU over long time horizons. The IoU at frame 19 is 0.34 in KITTI and 0.7 in CATER. Our model compares favorably to the 3D-upgraded colorization baseline.

We also input our Round3 KITTI detections to a recent tracking-by-detection method [58], and evaluated with standard multi-object tracking metrics. We obtained sAMOTA: 0.2990; AMOTA: 0.0863; AMOTP: 0.1581. This is encouraging but still far behind the supervised state-of-the-art, which obtains sAMOTA: 0.9328; AMOTA: 0.4543; AMOTP: 0.7741. Our main failure case appears to be missed detections. We also used this tracker to compute alternate results for the IoU-over-time evaluation (cf. Fig. 4.1): this yields a relatively stable line around 0.44 IoU across 20 frames.

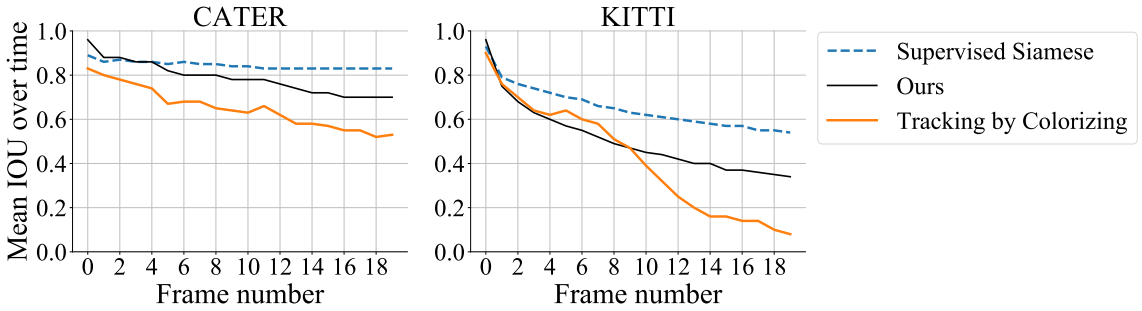


Figure 4.1: **3D object tracking IoU over time, in CATER and KITTI.** Tracking precision necessarily begins near 1.0 because tracking is initialized with a real object box in frame0, and declines over time, more drastically in KITTI than in CATER.

Table 4.2: Ablations of the trajectory library, in CATER.

Method	**Recall**	**Precision**
Ours, with short-range tracker	0.53	0.94
. . . and trajectory library	0.64	0.91

Performance appears upper-bounded by the detector's mean IoU.

4.5 Qualitative results

For object discovery, We show object proposals of our model in CATER and KITTI in Figure 4.2. Ground-truth boxes are shown in beige and proposed boxes are shown in blue. Their IoU are marked near the boxes. Results are shown on RGB image as well as bird's-eye view. The boxes have high recall and high precision overall; it can detect small objects as well as separate the object that are spatially close to each other. In KITTI, there are some false positive results on bushes and trees because of the lack of pseudo-label supervision there. We visualize KITTI object tracking in Figure 4.3.

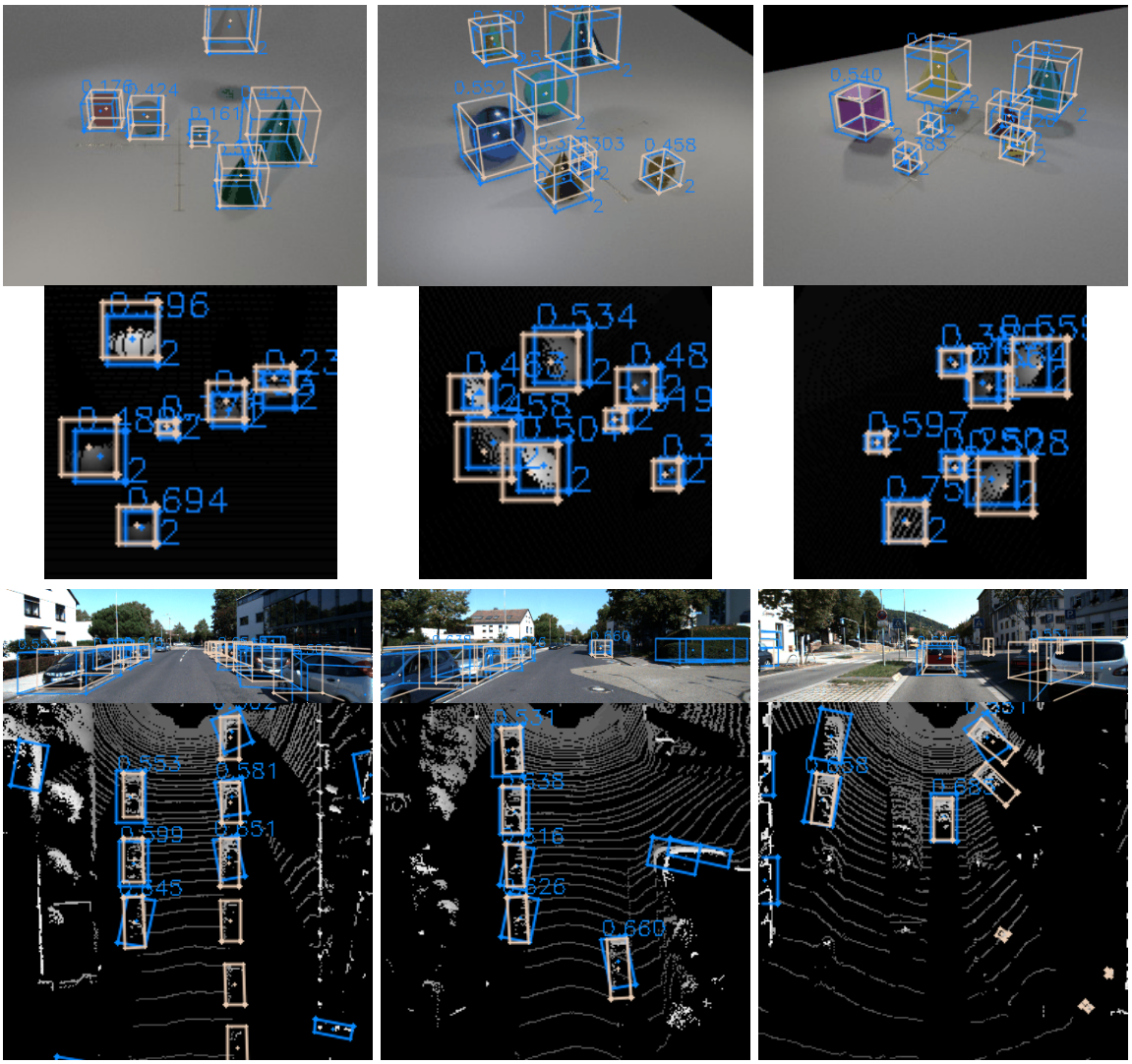


Figure 4.2: **3D object detections in CATER (top) and KITTI (bottom).** Ground-truth boxes are shown in beige and detection results are shown in blue. IoU scores are marked alongside each box. Results are shown in perspective RGB and bird's-eye view.

Figure 4.3: **3D object tracking in KITTI.** IoU scores are marked alongside each estimated box (in blue) across subsampled frames.

4.6 Ablation studies

4.6.1 Ablation on ensemble agreement

Figure 4.4 shows what happens when the ensemble agreement check is dropped: the model gradually begins classifying everything as an object (BEV mAP@.5=0.17 on Round2 instead of 0.28).

4.6.2 Ablation on the trajectory library

Table 4.2 shows an ablation study on the trajectory library. We report "Recall", which we define as the proportion of objects that are successfully tracked by our model from the beginning of the video to the end, where tracking success is defined by an IoU threshold of 0.5. We also report "Precision", which we define as the proportion of tracklets that begin and end on the same object. With the trajectory library, we improve the recall from 53% to 64%, while precision drops slightly from 94% to 91%. Qualitatively we find that the majority of improvement is on partially and fully-occluded objects, where strict appearance-based matching is ambiguous and prone to failure, but where the library is a useful prior.

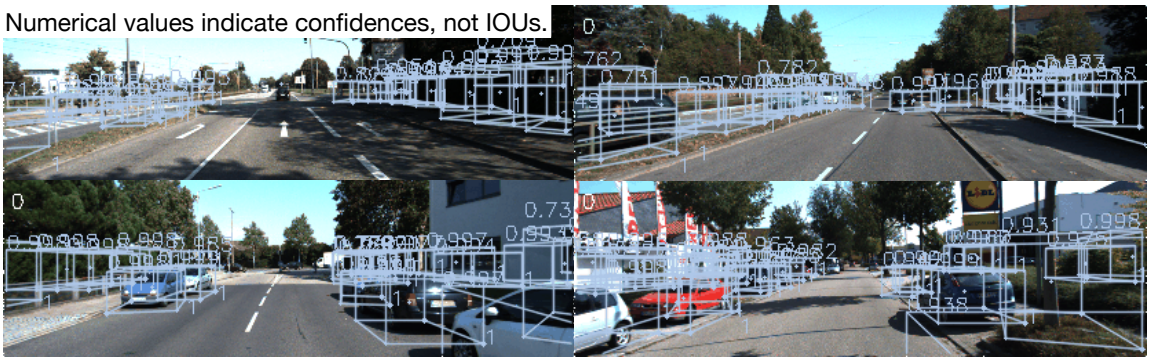


Figure 4.4: **Ablation of ensemble agreement causes divergence.** The model eventually detects objects everywhere.

4.7 Baseline object discovery analysis

Interestingly, the learning-based baselines, which achieve state-of-the-art results in synthetic datasets, have near zero accuracy in KITTI. This is probably because certain assumptions in their design are violated in this data (e.g., a static-camera assumption is violated, and there is relatively little self-similarity within object/background regions compared to CATER). We illustrate this in Figure 4.5 for MONet [8], and in Figure 4.6 for Slot Attention [40]. We find that optimizing these models is considerably more difficult in a real-world dataset such as KITTI, than it is in simple synthetic datasets.

4.8 Limitations

The proposed method has two main limitations. Firstly, our work assumes access to RGB-D data with accurate depth, which excludes the method from application to general videos (e.g., from YouTube). Second, it is unclear how best to mine for negatives (i.e., "not a moving object"). Right now we use a small region around each pseudo label as negative, but it leaves the method prone to false positives in far-away non-objects like bushes and trees.

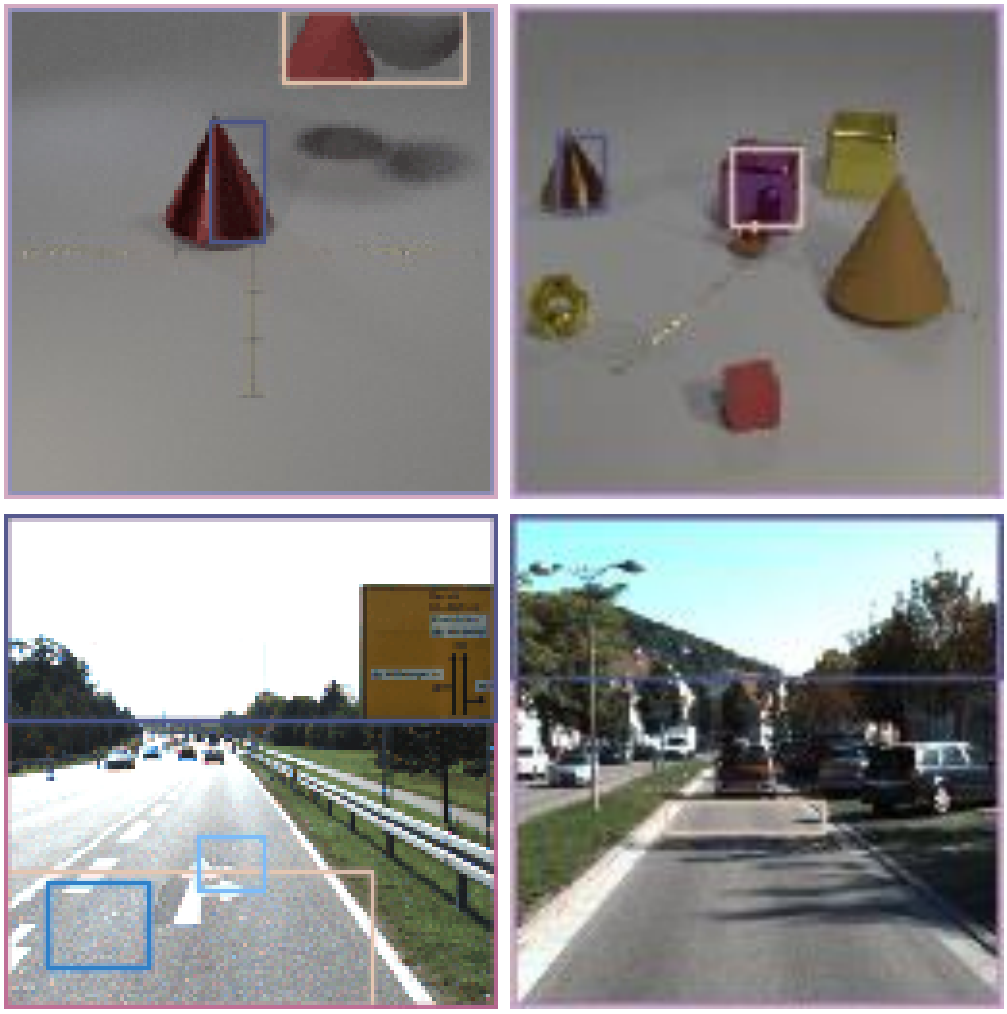


Figure 4.5: **Object discovery results by MONet [8] in CATER (top) and KITTI (bottom).** In CATER the proposed boxes are typically on objects, but in KITTI we find that the model produces boxes for non-object scene elements, such as segments of the road, lane markings, and the sky.

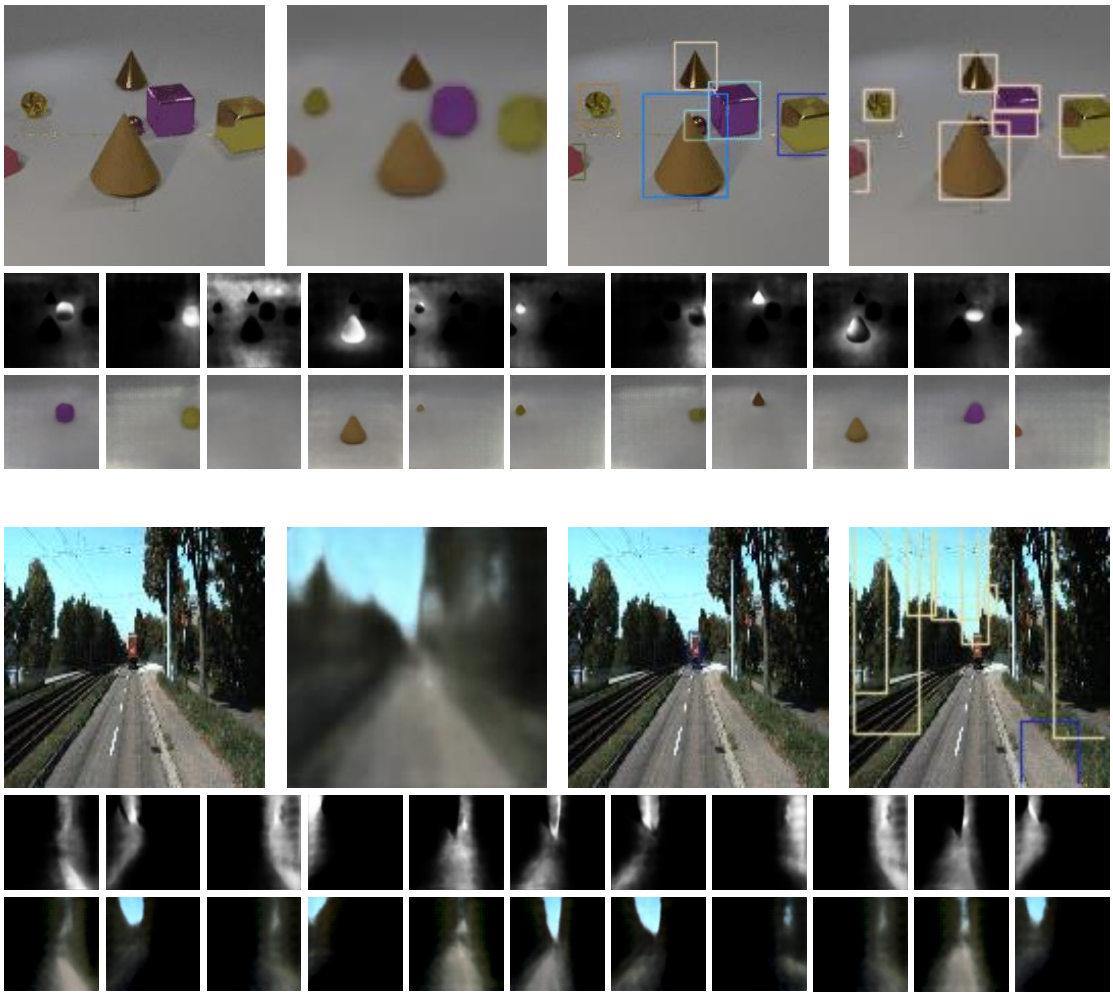


Figure 4.6: **Object discovery results by Slot Attention [40] in CATER (a) and KITTI (b).** For each dataset, the rows are arranged as follows: **1st row (left to right):** RGB ground truth, RGB reconstruction by the model, ground truth bounding boxes, predicted bounding boxes. **2nd row:** The predicted masks for each slot. **3rd row:** RGB reconstruction for each slot.

Chapter 5

Conclusions

This thesis proposes a self-supervised method for discovering and tracking objects in unlabelled RGB-D videos. We begin with a simple handcrafted technique for segmenting independently-moving objects from the background, relying on cycle-consistent flows and RANSAC. We then train an ensemble of 2D and 3D detectors with these segmentations, under heavy data augmentation. We then use these detectors to re-label the dataset more densely, and return to the training step. The ensemble agreement keeps precision of the pseudo-labels high, and the data augmentations allow recall to gradually expand. Our approach opens new avenues for learning object detectors from videos in arbitrary environments, without requiring explicit object supervision.

Bibliography

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2189–2202, 2012. 1, 3.8

[2] Zeyuan Allen-Zhu and Yuanzhi Li. Towards understanding ensemble, knowledge distillation and self-distillation in deep learning. *arXiv preprint arXiv:2012.09816*, 2020. 2.2

[3] Dan Barnes, Will Maddern, Geoffrey Pascoe, and Ingmar Posner. Driven to distraction: Self-supervised distractor learning for robust monocular visual odometry in urban environments. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1894–1900. IEEE, 2018. 2.3

[4] Luca Bertinetto, Jack Valmadre, Joao F Henriques, Andrea Vedaldi, and Philip HS Torr. Fully-convolutional siamese networks for object tracking. In *European conference on computer vision*, pages 850–865. Springer, 2016. 4.2

[5] Goutam Bhat, Felix Järemo Lawin, Martin Danelljan, Andreas Robinson, Michael Felsberg, Luc Van Gool, and Radu Timofte. Learning what to learn for video object segmentation, 2020. 2.4

[6] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In *European conference on computer vision*, pages 282–295. Springer, 2010. 4.2, **??**

[7] Thomas Brox and Jitendra Malik. Object segmentation by long term analysis of point trajectories. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *ECCV*, pages 282–295, 2010. 2.4

[8] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019. (document), 1, 2.1, 4.2, **??**, 4.7, 4.5

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. *arXiv preprint arXiv:2002.05709*, 2020. 2.5

[10] Jiyu Cheng, Yuxiang Sun, and Max Q-H Meng. Improving monocular visual slam in dynamic environments: an optical-flow-based approach. *Advanced Robotics*, 33(12):576–589, 2019. 2.3

[11] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. *ICCV*, 1995. 2.3

[12] Lincoln G Craton. The development of perceptual completion abilities: Infants' perception of stationary, partially occluded objects. *Child Development*, 67(3): 890–904, 1996. 1

[13] Antonia Creswell, Kyriacos Nikiforou, Oriol Vinyals, Andre Saraiva, Rishabh Kabra, Loic Matthey, Chris Burgess, Malcolm Reynolds, Richard Tanburn, Marta Garnelo, et al. Alignnet: Unsupervised entity alignment. *arXiv preprint arXiv:2007.08973*, 2020. 2.1

[14] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000. 2.2

[15] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019. 3.5, 3.9

[16] S. M. Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, koray kavukcuoglu, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 3225–3233. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/52947e0ade57a09e4a1386d08f17b656-Paper.pdf. 2.1

[17] Katerina Fragkiadaki and Jianbo Shi. Detection free tracking: Exploiting motion and topology for segmenting and tracking under entanglement. In *CVPR*, 2011. 1

[18] Katerina Fragkiadaki, Geng Zhang, and Jianbo Shi. Video segmentation by tracing discontinuities in a trajectory embedding. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1846–1853. IEEE, 2012. 4.2, **??**

[19] Katerina Fragkiadaki, Pablo Arbelaez, Panna Felsen, and Jitendra Malik. Learning to segment moving objects in videos. In *CVPR*, June 2015. 2.4

[20] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 1, 2

[21] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. In *ICLR*, 2020. 1, 1

[22] E Bruce Goldstein. Perceiving objects and scenes: the gestalt approach to object perception. *Goldstein EB. Sensation and Perception. 8th ed. Belmont, CA: Wadsworth Cengage Learning*, 2009. 1

[23] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR, 2019. 2.1

[24] Adam W Harley, Shrinidhi K Lakshmikanth, Paul Schydlo, and Katerina Fragkiadaki. Tracking emerges by looking around static scenes, with neural 3D mapping. In *ECCV*, 2020. 4.2

[25] 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. 3.4

[26] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, 2020. 2.5

[27] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2.2

[28] Yuan-Ting Hu, Hong-Shuo Chen, Kexin Hui, Jia-Bin Huang, and Alexander G. Schwing. Sail-vos: Semantic amodal instance level video object segmentation - a synthetic dataset and baselines. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2.4

[29] Laurent Itti and Christof Koch. A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision research*, 40(10-12):1489–1506, 2000. 3.8

[30] Jindong Jiang*, Sepehr Janghorbani*, Gerard De Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*, 2020. URL `https://openreview.net/forum?id=SJxrKgStDH`. 2.1

[31] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning, 2016. 1, 1

[32] Maik Keller, Damien Lefloch, Martin Lambers, Shahram Izadi, Tim Weyrich, and Andreas Kolb. Real-time 3d reconstruction in dynamic scenes using point-based fusion. In *2013 International Conference on 3D Vision-3DV 2013*, pages 1–8. IEEE, 2013. 2.3

[33] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In

IROS, 2013. 2.3

[34] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for object tracking. In *CVPR Workshops*, 2017. 2.5

[35] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4.3

[36] Tejas Kulkarni, Ankush Gupta, Catalin Ionescu, Sebastian Borgeaud, Malcolm Reynolds, Andrew Zisserman, and Volodymyr Mnih. Unsupervised learning of object keypoints for perception and control. In *NeurIPS*, 2019. 1

[37] Yu Li, Zhuoran Shen, and Ying Shan. Fast video object segmentation using the global context module, 2020. 2.4

[38] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 3.4

[39] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu. Selflow: Self-supervised learning of optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4571–4580, 2019. 3.2

[40] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *Advances in Neural Information Processing Systems*, 33, 2020. (document), 2.1, 4.2, **??**, 4.7, 4.6

[41] Lain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6): 810–815, 2004. 3.6

[42] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *CVPR*, 2016. 4.3

[43] Tom Mitchell, William Cohen, Estevam Hruschka, Partha Talukdar, Bishan Yang, Justin Betteridge, Andrew Carlson, Bhavana Dalvi, Matt Gardner, Bryan Kisiel, et al. Never-ending learning. *Communications of the ACM*, 61(5):103–115, 2018. 3.8

[44] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 39(2):103–134, 2000. 1, 3.10

[45] P. Ochs and T. Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*, 2011. 2.4

[46] Pan Pan, Fatih Porikli, and Dan Schonfeld. Recurrent tracking using multifold

consistency. In *Proceedings of the Eleventh IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2009. 3.2

[47] F. Perazzi, A. Khoreva, R. Benenson, B. Schiele, and A. Sorkine-Hornung. Learning video object segmentation from static images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3491–3500, 2017. 2.4

[48] Ilija Radosavovic, Piotr Dollár, Ross Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 2.2

[49] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 3.5

[50] Sudeep Sarkar and Kim L Boyer. Perceptual organization in computer vision: A review and a proposal for a classificatory structure. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(2):382–399, 1993. 1

[51] T. Schöps, J. Engel, and D. Cremers. Semi-dense visual odometry for AR on a smartphone. In *ISMAR*, 2014. 2.3

[52] Ishwar K Sethi and Ramesh Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on pattern analysis and machine intelligence*, 9(1):56–73, 1987. 3.2

[53] Elizabeth S Spelke, J Mehler, M Garrett, and E Walker. Perceptual knowledge of objects in infancy. In NJ: Erlbaum Hillsdale, editor, *Perspectives on mental representation*, chapter 22. Erlbaum, 1982. 1

[54] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision*, pages 402–419. Springer, 2020. 3.2

[55] 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. 2.5

[56] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: A factorization method. *Int. J. Comput. Vision*, 9(2): 137–154, November 1992. ISSN 0920-5691. doi: 10.1007/BF00129684. URL http://dx.doi.org/10.1007/BF00129684. 2.3

[57] Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and

Kevin Murphy. Tracking emerges by colorizing videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 391–408, 2018. 4.2

[58] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. AB3DMOT: A Baseline for 3D Multi-Object Tracking and New Evaluation Metrics. *ECCVW*, 2020. 4.4.2

[59] Hao Wu, Aswin C Sankaranarayanan, and Rama Chellappa. In situ evaluation of tracking algorithms using time reversed chains. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007. 3.2

[60] Heng Yang, Jingnan Shi, and Luca Carlone. TEASER: Fast and certifiable point cloud registration. *IEEE Transactions on Robotics*, 2020. 2.3

[61] Jason J. Yu, Adam W. Harley, and Konstantinos G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *ECCV*, 2016. 3.2

[62] Wenyuan Zeng, Wenjie Luo, Simon Suo, Abbas Sadat, Bin Yang, Sergio Casas, and Raquel Urtasun. End-to-end interpretable neural motion planner. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8660–8669, 2019. 3.7