Track, Check, Repeat: An EM Approach to Unsupervised Tracking

Anonymous CVPR 2021 submission

Paper ID 7264

Abstract

We propose an unsupervised method for 3D object segmentation and motion/content disentanglement of unlabelled RGB-D video streams. We isolate the independently-moving foreground under an arbitrary moving camera, split it into trackable components, and track each component in 3D, across partial and full occlusions. The method works by iteratively learning 2D and 3D identity detectors, objectness detectors, bottom-up 2D and 3D motion estimators, and a motion prior; together these modules form a high-precision tracker. Learning happens in an expectation-maximization framework, 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. This iterative process gradually expands recall, until the entire video is explained by tracklets. The constraint of ensemble agreement helps combat contamination of the pseudo-labels, and standard data augmentation techniques help the modules generalize to yet-unlabelled data. We compare against existing state-of-the-art unsupervised object discovery and tracking methods, using challenging real-world videos from the KITTI benchmark, and show strong improvements.

1. Introduction

Humans can detect moving objects and delineate their approximate extent in 3D from a single and possibly moving viewpoint, and without ever been supplied 3D boxes or 3D segmentation masks as supervision, whether in their lifespans or their evolutionary history. How does this remarkable ability develop without supervision? Neuroscience and psychology literature points to a variety of perceptual grouping cues, which makes some regions look more "object-like" than others. These types of objectness cues have long been known in computer vision literature; why has this domain knowledge not yet led to powerful self-supervised object detectors?

Work on integrating perceptual grouping cues into computer vision models stretches back decades, and is still likely serves as inspiration for many of the design decisions in modern computer vision architectures related to attention, metric learning, two-stream architectures, and so on. Much of the current work on object recognition and tracking is fully-supervised, and relies on vast pools of human-provided annotations. On the unsupervised side, a variety of deep learning-based methods have been proposed, which hinge on reconstruction objectives and object-centric or scene-centric bottlenecks in the architecture [5]. 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 [13].

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 suggest 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 train a self-supervisedly train a learning-based module to mimic and outperform the traditional method. This is a kind of knowledge distillation [19], from traditional models to deep ones.

Our optimization algorithm is essentially expectation maximization. After identifying the successful outcomes of a traditional object discovery method, we optimize the parameters of a learning-based method to make those successes more likely. We then use the combination of the traditional and learned methods to obtain new high-confidence estimates, and repeat. Our method outperforms not only the traditional methods, which only work under specific conditions, but also the deep methods, which currently only work in toy environments. We demonstrate success in a popular synthetic environment where recent deep models have been successful, and also on the real-world KITTI urban scenes benchmark, where the existing 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 simple handcrafted detectors and trackers, we can train a robust

learning-based method to detect and track objects in a target domain, without requiring any human annotations.

2. Related Work

Object discovery Many recent works have proposed deep neural networks for object discovery. These models typically have an object-centric bottleneck, and are tasked with a reconstruction objective. MONet [5], Slot attention [29], IODINE [17] , SCALOR [22], AIR [11], and AlignNet [8] fall under this category. These methods have been demonstrated successful in a variety of simple domains, but have not yet been tested on real-world videos. In this paper we use the publicly available code for a subset of these models and evaluate whether they are still able to perform well under the complexities of real-world imagery. Compared to our method, these baselines do not make use of depth, but in the supplementary file we show that concatenating depth to their input stream provides very little impact on performance.

Ensemble methods Using an ensemble is a well-known way to improve the performance of a machine learning 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 [9]. Ensembling is also the key idea behind knowledge distillation, where knowledge gets transferred from cumbersome models to simpler ones [19, 35]. The typical modern setup is to make up the ensemble out of multiple copies of a neural network, which are trained from different random initializations and therefore have slightly different parameters. In our case, the ensemble is more diverse: it is made up of components which solve different tasks, but which 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.

Never ending learning We take inspiration from the methods described in "never ending learning" [6]. We set up a version of this in an expectation-maximization framework, 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.

Structure-from-Motion/SLAM Earlier works on structure from motion (SfM) methods [40, 7] set the ambitious goal of extracting 3D scene pointclouds and camera 3D trajectories from 2D pixel trajectories, exploiting the reduced rank of the 2D trajectory matrix under rigid motions. Unfortunately, these methods were often confined to very simple videos, due to their difficulty handling camera motion degeneracies, non rigid object motion and frequent occlusions that caused 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 in real time, and assume a calibrated setup, i.e., that the focal length of the camera is known [37, 24]. Our model exploits the rare successes of these methods to learn about the static vs. moving part of the scene.

Moving object estimation Most motion segmentation methods cluster 2D optical flow vectors to segment moving objects. While earlier approaches attempted motion segmentation completely unsupervised by integrating motion information over time through 2D pixel flow trajectories [3, 32], recent works focus on learning to segment 2D objects in videos, supervised by annotated video benchmarks [2, 20, 34, 26, 12, 25].

3. Track, Check, Repeat

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 salient objects in the video.

We treat this data as a test set, in the sense that we do not use any annotations. In the current literature, most machine learning methods have a training phase and a test phase, where the model is "frozen" when the test phase arrives. Our method instead attempts to optimize its parameters for the test domain, using "free" supervision that it automatically generates (without any human intervention).

The optimization operates in "rounds". The first round leverages optical flow and cycle-consistency constraints, to discover moving objects in the videos. The most-confident object proposals are upgraded into pseudolabels for training appearance-based object detectors. 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.

3.1. Optical flow

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

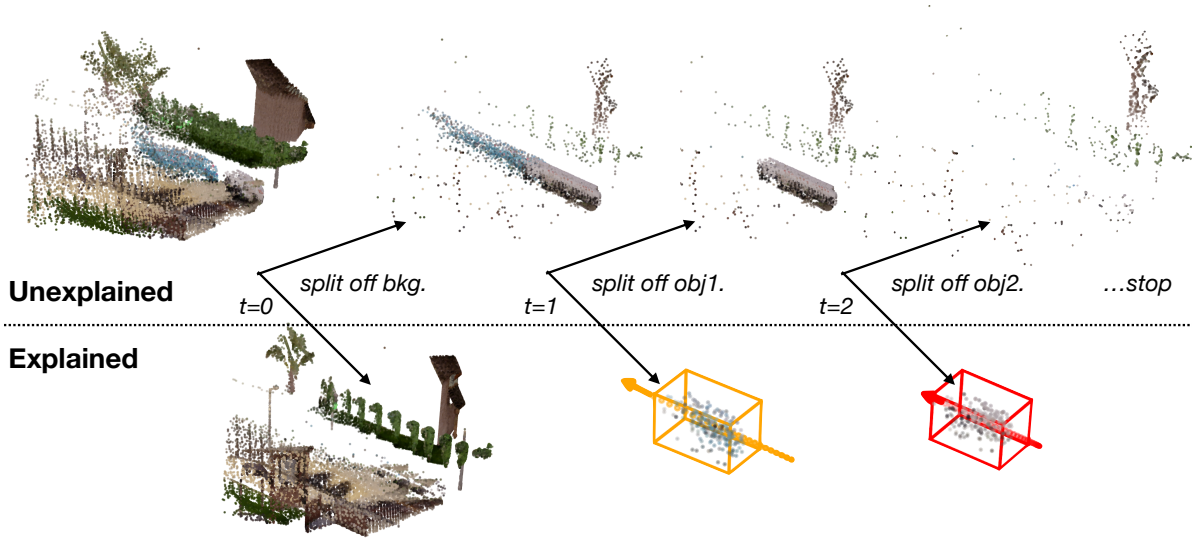

Figure 1. **Incrementally explaining an RGB-D video.** As input, our method consumes an RGB-D video, or alternatively, a set of RGB-D videos from a single test domain. This data is considered to be entirely "unexplained" at the start. This sequence is then "explained" in stages, which proceed from the most prominent moving components, which "pop out" due to motion saliency against the background, down to the less-prominent components which need to be discovered from appearance cues. Optimization proceeds in an expectation-maximization framework, where the parts of the scene that are already "explained" become pseudolabels for the next stage, which optimizes the parameters to explain more confidently with more modules, and generalize to the unexplained regions of the data.

use an off-the-shelf pre-trained convolutional optical flow network [39]. We note that optical flow can be learned entirely unsupervised [42].

We finetune this model with three self-supervision techniques. First, we use traditional edge-aware motion smoothness and brightness constancy [42]:

$$\mathcal{L}(\mathbf{w}; I_t, I_{t+1}) = \ell_{\text{photometric}}\left(\mathbf{w}; I_t, I_{t+1}\right) + \lambda \ell_{\text{smoothness}}\left(\mathbf{w}\right), \tag{1}$$

where $\{I_t, I_{t+1}\}$ are temporarily consecutive images, $\mathbf{w} \in \mathbb{R}^{H\times W\times 2}$ is the optical flow, and λ is the parameter that weighs the relative importance of the smoothness. The photometric loss is computed by

$$\ell_{\text{photometric}}\left(\mathbf{w}; I_t, I_{t+1}\right) = \sum_{i,j} \rho_D\left(I_t - I_{t+1}(\mathbf{p}+\mathbf{w})\right), \tag{2}$$

where ρ_D is the robust generalized Charbonnier penalty function $\rho_(x) = (x^2+\epsilon^2)^\alpha$ to mitigate the effects of outliers. And the smoothness loss is computed by

$$\begin{aligned}\ell_{\text{smoothess}}\left(\mathbf{w}\right) = \\ \sum_j^H \sum_i^W \left[\rho_S\left(u_{i,j} - u_{i+1,j}\right) + \rho_S\left(u_{i,j} - u_{i,j+1}\right)\right. \\ \left. +\rho_S\left(v_{i,j} - v_{i+1,j}\right) + \rho_S\left(v_{i,j} - v_{i,j+1}\right)\right],\end{aligned} \tag{3}$$

where $u_{i,j}, v_{i,j}$ are horizontal and vertical components at (i,j) of $\mathbf{w}$, respectively, and $\rho_s(\cdot)$ is the spatial smoothness penalty function realized by the generalized Charbonnier function.

Second, we make a copy of the model and freeze it, to act as a "teacher" for another copy. From flows estimated by the teacher, we compute a mask indicating which flows are forward-backward consistent [41, 33, 38]. We first generate the forward flow $\mathbf{w}_{t\to t+1}$ and the backward flow $\mathbf{w}_{t+1\to t}$. The reversed forward flow is computed by

$$\hat{\mathbf{w}}_{t\to t+1} = \mathbf{w}_{t+1\to t}\left(\mathbf{p} + \mathbf{w}_{t\to t+1}(\mathbf{p})\right). \tag{4}$$

A pixel $\mathbf{p}$ is considered to be forward-backward consistent if it satisfies

$$\left|\mathbf{w}_{t\to t+1} + \hat{\mathbf{w}}_{t\to t+1}\right|^2 < \alpha_1\left(\left|\mathbf{w}_{t\to t+1}\right|^2 + \left|\hat{\mathbf{w}}_{t\to t+1}\right|^2\right) + \alpha_2, \tag{5}$$

where we use $\alpha_1 = 0.05$ and $\alpha_2 = 0.5$ in our experiments.

This is a well-known "check" for optical flow – if flow is not cycle-consistent, it is not likely to be correct. We supervise the student model to mimic the teacher model at the pixels that are cycle-consistent. This helps ensure that the student model does not get worse than the teacher during optimization.

The third technique, inspired by the recent SelFlow method [28], is to apply random synthetic occlusions on the "student" model's input. Synthetic augmentations on the student, while self-supervising via the teacher, forces the student model to learn a more robust flow estimator than the teacher.

3.2. 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. Pixels moving independently are a strong cue for objectness.

We begin by "upgrading" the (dense) 2D flow fields into a sparse 3D pointcloud flow. To do this, we project the two pointclouds that correspond to the image pair, to obtain the nearest pixel coordinate of each 3D point. We then simply gather the flows whose startpoint and endpoint both have a corresponding 3D point, and use the delta of these points as the 3D pointcloud flow.

We then use RANSAC to estimate the rigid motion that explains the maximal number of 3D point flows. RANSAC is intended to be robust to outliers, but the answer returned is often catastrophically wrong.

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 the rigid motion twice: 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 displacement:

$$XYZ'_0 = RT_{10}^{bw} RT_{01}^{fw} (XYZ_0) \tag{6}$$

$$err = \max_n(||XYZ'_0 - XYZ_0||), \tag{7}$$

where RT_{01}^{fw} denotes the rotation and translation computed from forward flow, which carries the pointcloud from timestep 0 to timestep 1, and RT_{10}^{bw} is the backward counterpart.

If the maximum displacement across the entire pointcloud is below a threshold (set to 0.25 meters), then treat the estimate as "correct". In practice we find that this occurs about 10% of the time in the KITTI dataset.

On these successful runs, we use the egomotion to create another 3D flow field (in addition to the one produced by the flow module), 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.

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

3.3. 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 a moving object. 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 from of the "moving" type even when they are stationary.

We use a 50-layer ResNet [18] with a feature pyramid neck [27] as the architecture, and train the last layer with a logistic loss against sparse pseudo-ground-truth:

$$\mathcal{L}^{\text{seg}} = \sum \hat{m} \log(1 + \exp(-\hat{s} \cdot s)), \tag{8}$$

where m is a mask indicating where the supervision is valid. We experimented with and without ImageNet pretraining for the ResNet, and found that the pretrained version converges more quickly but is not very different in performance.

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.

3.4. 3D object detection

This module takes as input a voxelized colorized pointcloud (computed from the RGB-D and intrinsics), and estimates oriented 3D bounding boxes of objects.

We use 3D U-Net-style convolutional encoder [36], and a CenterNet-style detection head [10]. 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). For implementation details we refer the reader to the supplementary file, and the original 2D CenterNet paper [10].

In training this module, we find that color augmentations have little effect, but randomized pointcloud orientation (when creating the voxelized input) is critical for learning an even distribution over possible object orientations. Additionally, we create partial occlusion augmentations, by randomly erasing an 20×20 area around the object center, while ensuring the target is not fully occluded.

3.5. Short-term tracking

To relocate a detected object over short time periods, we use the classic technique of cross correlation with a rigid template [30]. We do this using the features provided by the

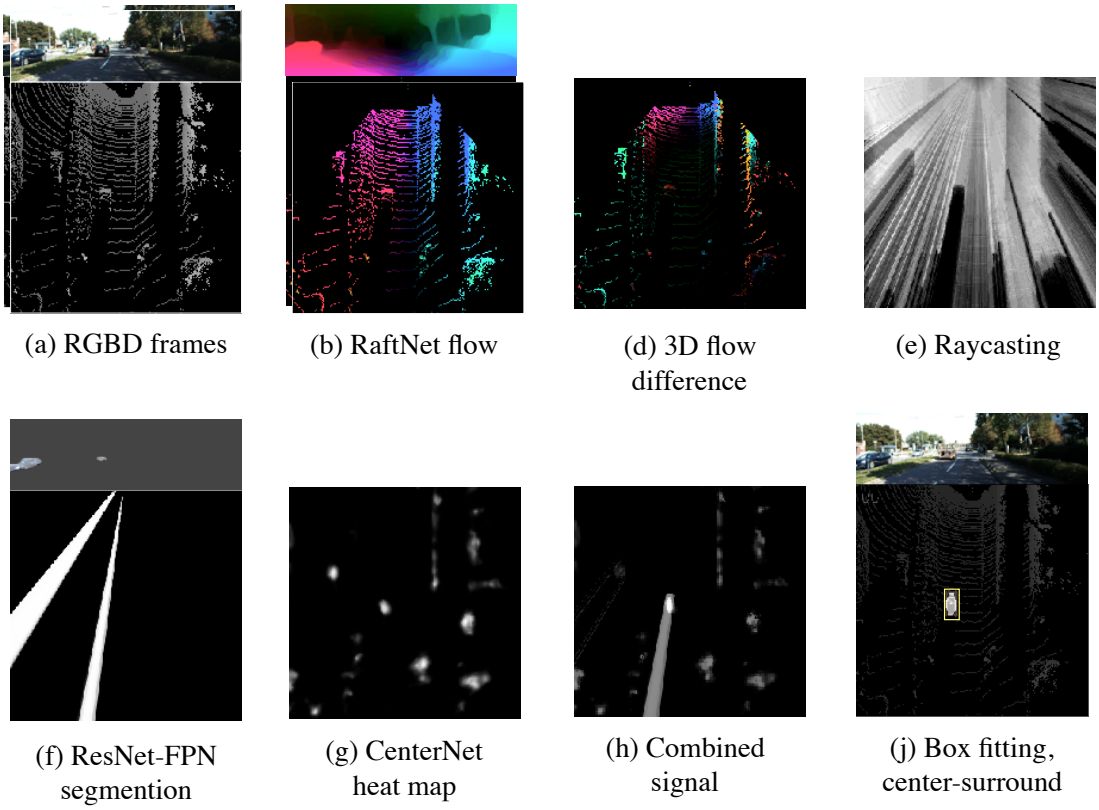

Figure 2. **Expectation (E) step of our unsupervised object discovery pipeline.** Given an input video (a), we have multiple sources of evidence, including flow difference (c), visibility ray casting (e), 2D segmentation (f) and object-ness score from the previous round (g). Each of them can be error-prone, but ensembling them gives us high confidence object labels (j).

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. 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.

3.6. Long-term tracking, with trajectory libraries

To track the objects, we need to associate the detection of the same objects through time. We use a trajectory library as a reference, to check the if the association is affordable, and to link detections across long temporal gaps (e.g., across occlusions).

To build the trajectory library, we first use nearest neighbour to link the detection from temporally consecutive frames. The trajectories are initialized by the detection in the first frame. Then we iterate through frames to extend the trajectories. We assume that the objects move with constant velocity. Therefore, we can approximate the location of the object in the next frame $t+1$. The detection that is close to the approximation and has similar size in frame $t+1$ will be appended to the trajectory. The trajectories that last through the whole video will be included in the trajectory library since they are non-occluded and thus have high confidence to show the possible paths of objects.

During inference, we will first find the trajectories with the approach to build the trajectory library and mark the detections that have been associated to trajectories as explained. Notice that only those surviving through the whole video will be found. To also track the occluded objects, we use the trajectory library to link the unexplained detections. Given two unexplained detections D_{ts}, D_{te} from two different frames, where ts and te are the frame ids, we try to find the trajectories that have the same movement as the one from D_{ts} to D_{te} within $te - ts$ frames. Then we evaluate the possible trajectories in the library by computing the likelihood given the visual evidence. We generate the cost volumn by adding the centernet objectness heatmap, the unproject 2D segmentation heatmap and the invisible

map, where the centernet objectness heatmap and the unproject 2D segmentation heatmap provide the confidence that *visible* objects are on the trajectory and the invisible map shows where can have *invisible* objects that cannot be found by visual modules. The cost through the trajectory in the cost volumn is used as the likelihood. If there exist trajectories in the library that have high likelihood, we will say D_{ts} and D_{te} are likely to belong to the same trajectory, thus linking them as a new trajectory. Notice that the detections from other frames that are close to this trajectory will also be explained.

3.7. 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 [31], 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 [21]: (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]:

$$cs(\theta) = \chi^2(h(\text{cen}_\theta(\Delta XYZ)), h(\text{surr}_\theta(\Delta XYZ))), \quad (9)$$

where θ denotes the region being evaluated, cen_θ and surr_θ select points within and surrounding the region, h computes a histogram, and ΔXYZ denotes the 3D motion field.

When the trained objectness detectors are available (i.e., on rounds after the first), we convert the rigid motion field into a heatmap with $exp(-\lambda||\Delta XYZ||)$, and add this heatmap to the ones produced by the 2D and 3D objectness estimators, and then proceed with thresholding, connected components, and box fitting as normal. The only difference is that we use a threshold that demands multiple modules to agree: since each module produces confidences in $[0, 1]$, setting the threshold to any value above 2 effectively enforces this constraint.

4. Experiments

We test our model in the following datasets:

1. Synthetic RGB-D videos of tabletop scene rendered with CATER [16]. CATER is a built upon CLEVR [23], and it focuses on testing the model's ability to do long term temporal reasoning. We modified the simulator so that it can generate pointclouds, but leave all other rendering parameters untouched. The max number of object is set to 10 to make the scene as complex as possible. Videos are captured by 6 virtual camera set static around the scene.

2. Real RGB-D videos of urban scenes, from the KITTI dataset [15]. This data was collected with a sensor platform mounted on a moving vehicle, with a human driver navigating through a variety of areas in Germany. The data provides multiple RGB images; we use the "left" color camera. The dataset provides depth in the form of LiDAR sweeps synced to the RGB images. We use the "tracking" subset of KITTI, which includes 3D object labels, and approximate (but relatively inaccurate) egomotion provided by an inertial measurement unit.

In all datasets, we test on 50-frame videos. We evaluate all models on their ability to discover objects in 3D. For our model and the traditional baseline, we additionally evaluate tracking performance.

4.1. Baselines

We evaluate the following unsupervised object discovery baselines. We also tested the baselines for unsupervised object tracking, but their performance is poor on both datasets, so we include the results in supplementary materials.

- **Object-Centric Learning with Slot Attention [29].** The Slot Attention model trains an autoencoder with image reconstruction loss. An iterative attention-based update mechanism is applied when constructing the slots, which act as a representational bottleneck.

- **MONet [5].** MONet applies an recurrent attention mechanism that tries to explain the scene part by part. A VAE is also trained to reconstruct the image. Since there is no official code released, we re-implemented using PyTorch.

- **Discontinuities Tracing [4].** This method first extract dense point trajectories using optical flow. Video segmentation is then performed by detecting discontinuities of embedding density. We also tested a more naive approach **Spectral Clustering** [14] which the Discontinuities Tracing improved upon.

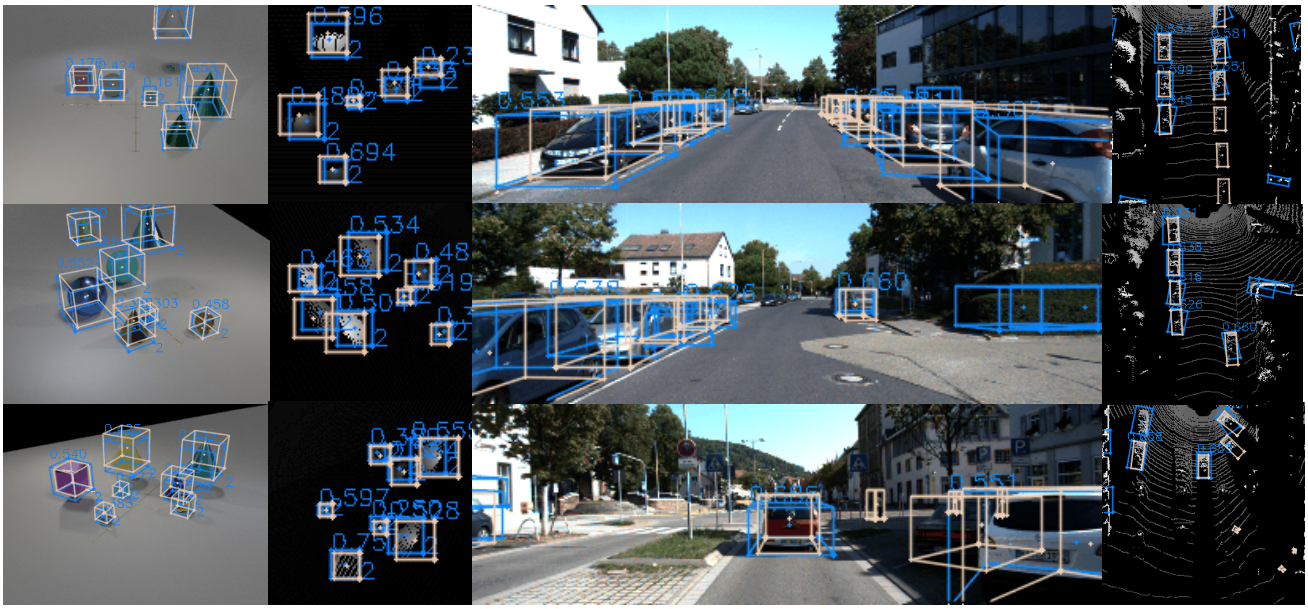

Figure 3. **3D object detections in CATER (left) and KITTI (right).** Ground-truth boxes are shown in beige and detection results are shown in blue. IoU scores are marked alongside each boxes. Results are shown in RGB and bird's-eye view.

Table 1. Comparison of ours method to baselines performance on Object Discovery task. Results are reported as mAP at 7 different IoU thresholds. Ours method works best in all the matrices reported. 2D means perspective view and BEV means bird's-eye view.

Method	Dataset	mAP@X						
		0.1	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**
Slot Attention [29]	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 [5]	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 [4]	CATER (2D)	0.18	0.08	0.04	0.03	0.01	0	0
	KITTI (2D)	0.01	0	0	0	0	0	0
Discontinuity Aware Clustering [14]	CATER (2D)	0.17	0.08	0.04	0.02	0.01	0.01	0
	KITTI (2D)	0.01	0	0	0	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**

4.2. Quantitative Results

Object Discovery Our main evaluation is in Table 1, where we evaluate the object proposal accuracy in mean Average Precision (mAP) at different IoU threshold, on both CATER and KITTI dataset. The numbers are collected in Bird's-eye view (BEV) and in RGB (2D). We find that our model produces the most accurate bounding boxes, in nearly all metrics. Adding another round of EM improves the numbers most of the time. Interestingly, the baselines methods that achieves state-of-the-art results in synthetic datasets have near zero accuracy in KITTI, probably because their unsupervised learning objectives are not power enough to produce meaningful decomposition of complex scenes.

Object Tracking The object tracking accuracy (in IoU) is shown in Table 4. We provide the model with ground-truth bounding box at frame0. No ego-motion information are used during testing. Our methods can maintain a high IoU over a long time horizon. The IoU at frame19 is 0.34

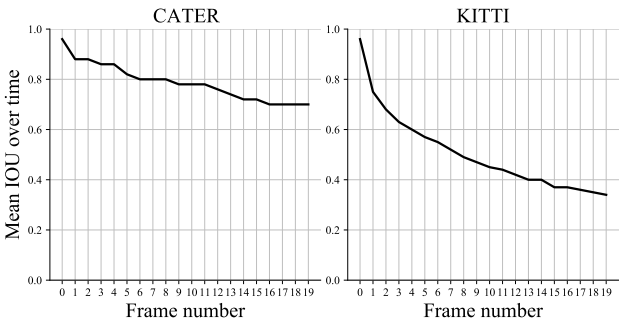

Figure 4. **Object Tracking IOU in KITTI and CATER.**

Table 2. Ablations of the trajectory library, in CATER.

Method	Accuracy	Precision
Ours, with nearest neighbour	0.53	0.94
. . . and trajectory library	0.64	0.91

in KITTI and 0.7 in CATER.

Ablation on the trajectory library The ablation study on the trajectory library is shown in Table 2. The accuracy is the ratio of the ground-true trajectories that are found in predictions. The precision is the ratio of the predicted trajectories that successfully link the detections of the same objects. With the trajectory library, we improve the accuracy from 53% to 64%, while no big drop happens in precision measurements. It shows that our method can successfully track the objects through occlusion without blindly linking unexplained detections.

We present the remainder of the quantitative results in the supplementary, showing that the tracking performance of our model outperforms the baseline, and showing that ablating components of our model decreases performance.

4.3. Qualitative Results

For object discovery, We show object proposals of our model in CATER and KITTI in 3. 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 object tracklets in KITTI in Figure 5, but we encourage the reader to inspect the supplementary video for clearer visualizations of the tracking performance.

Figure 5. **3D object tracking in KITTI.** Ground-truth boxes are shown in beige and detection results are shown in blue. IoU scores are marked alongside each boxes. Each column shows timesteps from a video. Objects frequently undergo partial occlusions and truncation.

4.4. Limitations

The proposed method has three main limitations. Firstly, our work assumes access to RGB-D data with accurate depth, which is not affordable yet for many domains that our method has potential application on. Secondly, our model architecture requires a lot of GPU memory, due to its third spatial dimension. This severely limits either the resolution and speed. Finally, it is unclear for us how to mine for negatives. Right now we use a small region around each pseudo label as "negative", but it leaves us open to false positives in far-away stuff like bushes and trees. Sparsifying our feature grid, or using points instead of voxels, are clear areas for future work.

5. Conclusion

We propose an unsupervised method that learns by segmenting motions of unlabelled RGB-D video streams. Learning and improvement is achieved in an expectation-maximization framework. In the expectation step we take the high confidence agreement among individual modules to create pseodo labels, and in the maximization step we retrain the modules. Standard data augmentation techniques are used to improve the generalization ability of the model. We showed that our method achieves state-of-the-art object discovery and tracking accuracy on two challenging datasets, and we further show that our method can address occlusions. Our approach opens new avenues for learning object models from videos in arbitrary environments, without requiring explicit object supervision. Extending our method to handle deformable and articulating motion is a useful avenue for future work.

References

[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, 6

[2] 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

[3] 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] 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. 6, 7

[5] 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. 1, 2, 6, 7

[6] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam R. Hruschka, and Tom M. Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, AAAI'10, page 1306–1313. AAAI Press, 2010. 2

[7] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. *ICCV*, 1995. 2

[8] 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

[9] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000. 2

[10] 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. 4

[11] 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. 2

[12] Katerina Fragkiadaki, Pablo Arbelaez, Panna Felsen, and Jitendra Malik. Learning to segment moving objects in videos. In *CVPR*, June 2015. 2

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

[14] 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. 6, 7

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 6

[16] Rohit Girdhar and Deva Ramanan. Cater: A diagnostic dataset for compositional actions and temporal reasoning. *arXiv preprint arXiv:1910.04744*, 2019. 6

[17] Klaus Greff, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. *arXiv preprint arXiv:1903.00450*, 2019. 2

[18] 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. 4

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 1, 2

[20] 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

[21] 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. 6

[22] 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. 2

[23] 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. 6

[24] C. Kerl, J. Sturm, and D. Cremers. Dense visual SLAM for RGB-D cameras. In *IROS*, 2013. 2

[25] Anna Khoreva, Rodrigo Benenson, Eddy Ilg, Thomas Brox, and Bernt Schiele. Lucid data dreaming for object tracking. In *CVPR Workshops*, 2017. 2

[26] Yu Li, Zhuoran Shen, and Ying Shan. Fast video object segmentation using the global context module, 2020. 2

[27] 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. 4

[28] 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

[29] 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. 2, 6, 7

[30] Lain Matthews, Takahiro Ishikawa, and Simon Baker. The template update problem. *IEEE transactions on pattern analysis and machine intelligence*, 26(6):810–815, 2004. 4

[31] 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. 6

[32] P. Ochs and T. Brox. Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*, 2011. 2

[33] 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

[34] 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

[35] 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

[36] 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. 4

[37] T. Schöps, J. Engel, and D. Cremers. Semi-dense visual odometry for AR on a smartphone. In *ISMAR*, 2014. 2

[38] Ishwar K Sethi and Ramesh Jain. Finding trajectories of feature points in a monocular image sequence. *IEEE Transactions on pattern analysis and machine intelligence*, (1):56–73, 1987. 3

[39] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020. 3

[40] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: A factorization method. *Int. J. Comput. Vision*, 9(2):137–154, Nov. 1992. 2

[41] 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

[42] 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