Segmenting out Generic Objects in Monocular Videos

Jan Hula, David Adamczyk, David Mojzisek, and Vojtech Molek

CE IT4I - IRAFM, University of Ostrava
30. dubna 22, 701 03 Ostrava, Czech Republic
{jan.hula, vojtech.molek, david.adamczyk, david.mojzisek}@osu.cz

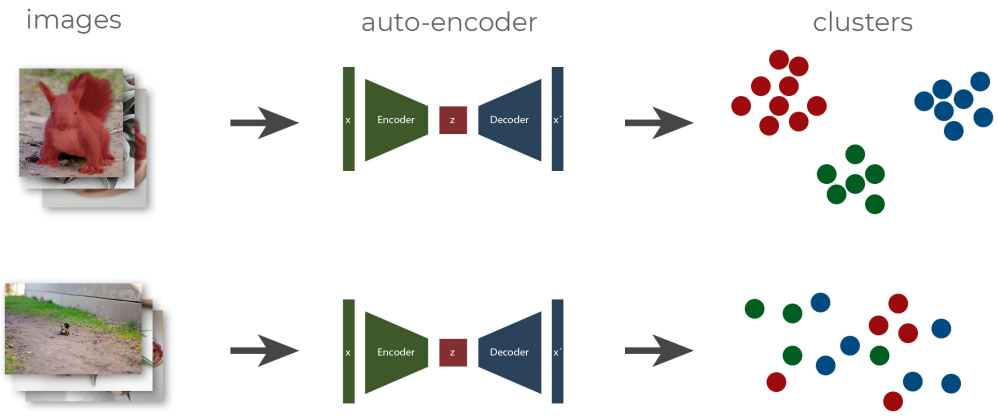

Figure 1: A schema depicting the improvement of clustering based on features from autoencoder trained on segmented objects using our approach vs. the one trained on the original images.

Abstract: We present an approach for generic object detection and segmentation in monocular videos. In this task, we want to segment objects from a background with no prior knowledge about the possible classes of objects which we may encounter. This makes this task much harder than the classical object detection and segmentation, which can be posed as a supervised learning problem. Our approach uses an ensemble of 3 different models which are trained by different objectives and have different failure modes and therefore complement each other. We demonstrate the usefulness of our approach on a custom dataset containing 18 classes of organic objects. Using our method, we were able to recover the classes of objects in a fully unsupervised way.

1 Introduction

Separating generic objects from a background in monocular videos is a challenging task. We believe that this problem is essential to Computer Vision and, as such, gained an unproportionally small amount of attention from the research community. The ability to separate objects from the background would vastly simplify other tasks as it can be viewed as a kind of dimensionality reduction on relevant features.

In image classification, object separation prevents a classifier from learning spurious correlations, which could arise when a certain class is often captured on a particular background. Object separation from a background automatically restricts a classifier to consider only features directly tied to the class label, as opposed to features only correlated with it.

As the generic object separation is a rather nonstandard task and still vaguely defined – it is not clear what should be considered an independent object – we focus on a simplified setting in which a camera captures a single salient object. Our solution is an ensemble of three different models trained for different objectives.

Furthermore, we study the impact of the background removal on the clustering properties of the resulting representations. The representations are obtained by training a neural network in an unsupervised way. We are able to recover the categories of objects in a fully unsupervised way, using our custom dataset containing videos of organic things.

Our main contributions are:

- We present an ensemble model that can separate objects from the background in monocular videos containing one salient object. The ensembled models compensate for each other failure modes.
- We demonstrate the benefits of object separation by comparing the classification accuracy of objects with and without background using clustering.

Section 2 contains related work. Section 3 describes our approach for detecting and segmenting generic objects within monocular videos. Section 4 describes how the detected objects enable unsupervised discovery of object

classes. In section 5 we describe our experiments and the dataset we test our approach on, and finally we provide a conclusion in section 6.

2 Related Work

Generic object separation is a largely an unexplored area, and therefore similar works are scarce. Our approach is most related to DINO method [1] which uses self-supervised learning with Vision Transformers. The authors introduced DINO as a form of self-distillation with no labels. They emphasize that DINO automatically learns an interpretable representation and separates the main object from the background clutter.

Lu et al. introduced an approach called CO-attention Siamese Network (COSNet) [2] for unsupervised video object segmentation. It is based on two ideas. The first is the importance of inherent correlation among video frames, and the second is the global co-attention mechanism responsible for learning motion in short-term temporal segments. The COSNet is trained on pairs of video frames, which increases the learning capacity.

The task of class discovery is marginally related to current self-supervised approaches using large amounts of unlabeled data such as [3, 4] and approaches that try to exploit coherency in the data [5].

Lastly, our approach for class discovery can be seen as a version of clustering with constraints. It has been heavily studied in the past, for example, by [6, 7].

3 Generic Object Detection and Segmentation

This section describes our approach to generic object detection and segmentation. By a generic object we mean an object of an unknown class. We use this term to distinguish it from classical object detection and segmentation, which can deal only with a concrete set of specified classes. Classical object detection and segmentation is much easier because it can be approached as a supervised learning problem on a dataset with annotated bounding boxes and segmentation masks. With generic objects, it is not that straightforward because it is not known in advance what kind of objects we will encounter at the test time.

Moreover, at first it may not be obvious how to define what should be considered as a separate object. One useful definition would be that an object is anything that can move independently from the rest of the environment. In this view, we can understand generic object segmentation as a way to factorise the visual stream into independent components. We need to mention that this definition does not cover all cases in which we would like to detect something as a separate object. Examples include buildings, letters on a sheet of paper, and other "entities" which can not move independently. Nonetheless, we consider this as our working definition because it allows us to make progress in generic object detection and segmentation.

3.1 Ensemble of Models Trained for Different Objectives

Our approach to the problem of generic object detection and segmentation is based on an ensemble of 3 models which are trained by different objectives. Even though each of these models has its own failure modes, together they constitute a robust ensemble. Concretely, we use one model trained for depth map prediction, one model trained for optical flow estimation, and one model trained for tracking of objects. Using the model for depth prediction, we can separate foreground objects based on depth, using the model trained for optical flow estimation, we can separate moving objects, and finally, using the tracker, we can verify the temporal coherency of our predictions. The tracker is initialised with a bounding box obtained from the predictions of the two other models in the frames where these predictions are most consistent. The following paragraphs provide a high-level description of these three models. For a more complete description of these models, see the respective publications.

Depth Prediction For the depth prediction, we use the model introduced by Ranftl et al. [8], available from the author's repository[1]. This transformer-based model predicts a scalar value for each pixel, which represents the distance of the surface captured by that pixel from the camera center.

Optical Flow Estimation For optical estimation, we use the model introduced by Teed et al. [9]. It is also a transformer-based model which requires 2 consecutive frames of video to produce the optical flow field. The optical flow field assigns two scalar values to each pixel. These values represent the pixel displacement on the x and y axes, relative to the previous frame. To obtain one scalar value for each pixel, we take the magnitude of the displacement. We used the implementation of the model with trained weights provided in the authors repository[2].

Object tracking For tracking objects, we use a model called SiamMask [10], a neural network trained as a Siamese architecture which simultaneously performs both visual object tracking and object segmentation in a video. We used the implementation available online[3].

[1] https://github.com/intel-isl/DPT
[2] https://github.com/princeton-vl/RAFT
[3] https://github.com/foolwood/SiamMask

3.2 Segmenting out Known Classes

In our dataset, each video captures a hand holding one object. Using our approach for generic object segmentation which is based on predicted depth maps and optical flow, our model segments out the hand together with the object. We fix this issue by segmenting out hands separately by a model trained specifically for hand segmentation.

To obtain training data for hand segmentation, we downloaded the following 4 datasets: GTEA, HandOverFace, GTEA_GAZE_PLUS, and EgoHands[4][5][6]. The architecture of the model is UNet with timm_regnetty_160 [11] as encoder and softmax2D as activation. The encoder weights were pretrained on ImageNet.

Using freely available datasets for hand segmentation mentioned above, the trained model was not working well on our dataset, probably because of a large distribution shift (most of the images in these public datasets contained hands in front of the face or were captured in the interior).

To mitigate this problem, we used a simple trick to enlarge the training data with images of hands which are similar to the images in our target dataset. Concretely, we captured our hands from a similar viewpoint as in our dataset, and then used the same model for depth prediction to produce depth maps for every 10th frame within the video. Finally, we thresholded the predicted death maps to obtain reliable segmentation masks of hands. In this way, we obtained hundreds of labeled images of hands with minimal effort. After adding this dataset to the other datasets, we obtained an accurate model for hand segmentation.

We use this model to remove hands from the mask predicted by our ensemble. More precisely, we remove the hands from the outputs of the model predicting the depth map and the model predicting the optical flow before we initialise the bounding box for the tracker. In this way we obtain masks only for the object, ignoring the hands.

3.3 Bounding Box Initialization

As mentioned above, we initialize the tracker with a bounding box obtained from the predictions of depth and optical flow maps within each frame. These predictions are two rectangular matrices of the same shape. We rescale the range of values to the interval between 0 and 1 and denote the final matrices by $f_1(x)$ and $f_2(x)$, respectively.

We first choose k frames where the predictions from these two models are most consistent. To measure this consistency, we devise the following heuristic. We first detect edges using a Canny edge detector [12] in both predictions and then measure the overlap of the resulting edges. To account for small deviations of edges in the two predictions, we blur them with a gaussian kernel of the width set to 7px to achieve their overlap if they are close to each other. Then we compute the consistency score c of frame x using the following formula:

$$c(x) = \sum_{i \in Pixels(x)} e^{\varepsilon 1_i + \varepsilon 2_i}, \tag{1}$$

where $\varepsilon 1$ and $\varepsilon 2$ are the two blurred edge maps from the two predictions and i indexes individual pixels. Next, we obtain the aggregated predictions for each pixel i by:

$$y(x)_i = e^{f_1(x)_i + f_2(x)_i} + \frac{1}{|Pixels(x)|} \sum_{j \in Pixels(x)} e^{f_1(x)_j + f_2(x)_j}. \tag{2}$$

We exponentiate the sum of the predictions from the two models because we want these predictions to interact superlinearly. We also subtract the mean of this value taken over all pixels within the image to make the aggregated predictions centered at zero. Therefore, the pixels where no object was predicted will contain negative values.

Once we have the aggregated predictions for the selected frames, we initialize the bounding boxes for the tracker. For this, we again devise a score which captures how well a given bounding box (*bbox*) covers pixels with high values (signifying that an object is present) and at the same time excludes pixels with low values. It has the following form:

$$\text{bboxScore}(bbox, y(x)) = \sum_{i \in Pixels(x)} \text{isInBbox}(i, bbox) \cdot y(x)_i, \tag{3}$$

where the function isInBbox returns -1 if the pixel is not contained in the bounding box and 1 otherwise.

Finally, we optimize the coordinates of the bounding box using CMA-ES [13] which is a derivative-free optimization algorithm used for the optimization of continuous parameters. The optimization tries to find coordinates which maximize this score. At the end of this procedure, we obtain k frames with bounding boxes in each video. The quality of predictions and the resulting bounding box is shown in Figure 2.

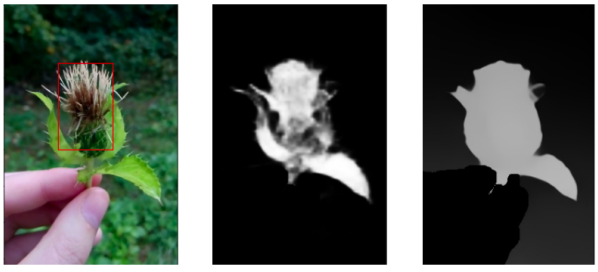

Figure 2: Predictions from the model for optical flow estimation (middle) and depth estimation (right). The initialized bounding box is depicted in the RGB image.

[4] http://cbs.ic.gatech.edu/fpv/
[5] https://www.cl.cam.ac.uk/research/rainbow/emotions/hand.html
[6] http://vision.soic.indiana.edu/projects/egohands/

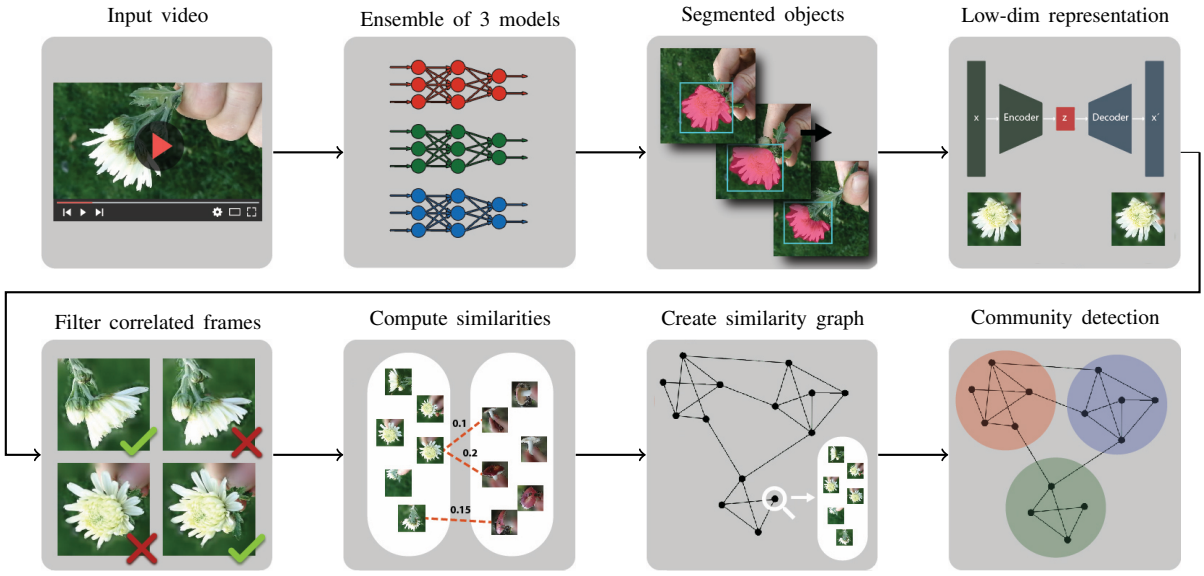

Figure 3: This figure depicts the whole pipeline of our approach. The diagram describes the process of our approach from the first step, where we preprocess the input video for the last step, where we obtain sets of similar objects.

3.4 Verification with the Tracker

Using the chosen frames and their bounding boxes, we initialize the tracker and let it track the object in between the selected frames. The tracker provides another layer of consistency check. Once we have the predictions from the three models (denoted by $f_1(x)$, $f_2(x)$ and $f_3(x)$), we can treat the consistency of these predictions as a certainty of the whole ensemble. To measure this certainty, we again compute the consistency score as we did in the selection of reliable frames in the Equation 1. Using an empirically estimated threshold, we filter out frames with low consistency and for each pixel i in the filtered frames, we aggregate the predictions of the ensemble with the following formula:

$$\text{output}(x)_i = \frac{\min\left(e^{\sum_{j=1}^{3} f_j(x)_i} - 1, e^2 - 1\right)}{e^2 - 1}, \qquad (4)$$

The subtraction of 1 ensures that we get 0 when all three models predict 0. Thresholding and dividing by $e^2 - 1$ insures that we obtain a value close to 1 when at least two models predict values close to 1. Finally, we obtain a bounding box for each frame using the same method as in the bounding box initialization (optimization using CMA-ES), i.e., minimizing the objective in Equation 1.

The whole process can be viewed as certainty propagation. We first select a few frames where the first two models agree on their predictions and from these the tracker propagates the certainty to other frames.

Evaluating The Quality of the Aggregated Predictions Our final task is a discovery of classes of objects within monocular videos. We view the generic object detection and segmentation as an intermediate step towards this goal. Therefore, we evaluate the usefulness of our approach on this target task. That is, we test how well are we able to recover classes of objects from images where the objects were segmented out by our approach. We compare it to the setup where we use the same algorithm for class discovery but where we use the original images with a background. We also mention that we do not require pixel-perfect segmentation masks as our goal is only to focus on the relevant parts of the image, so that the measured similarity between images will mostly reflect the similarity of objects and not of backgrounds. The next section describes our pipeline for the task of class discovery

4 Class Discovery

The algorithm for class discovery was proposed in [14]. Its input is a set of videos, each containing one object and each represented as a sequence of images. The goal is to find clusters of videos based on the similarity between them. Generally, our algorithm works in 3 steps:

1. Measure the similarity between every pair of videos with the method described in Section 4.1.
2. Construct a similarity graph by connecting each video to its five most similar videos.
3. Apply the Louvain community detection algorithm [15] to detect the highly interconnected parts of the graph and consider these as the discovered classes.

The advantage of the Louvain algorithm is that it needs no apriori knowledge of the number of clusters/communities.

The accuracy of our approach is measured in two ways. First, by counting how many times a video was assigned

to an incorrect cluster. Second, whether the algorithm discovered all clusters. It is clear that the final accuracy mostly reflects the measured similarity between individual videos.

4.1 Computing Similarity between a Pair of Videos

Each video is represented by a sequence of images, but to compute the similarity, we ignore the ordering and treat the sequence as a set. The similarity between the two videos is computed in the following four steps:

1. Train an autoencoder using images from all videos to obtain a low-dimensional representation z_i of each image x_i.
2. In each video, select n representative frames which are not correlated, described in 4.2.
3. For each pair of videos, compute all pairwise similarities $d(z_i, z_j)$ with cosine distance.
4. Finally, select the l most similar pairs of images and average their similarities to obtain the final similarity between two videos.

The intuition behind step 4 is that videos of similar objects may contain only a few frames where these objects are captured from the same angle or in the same situation.

4.2 Filtering out Correlated Frames

Step 2 of similarity computation takes n representative frames. If we would simply use all frames from each video, the distribution of the dataset may end up skewed because some parts of a video may be more static than others. These static parts would produce many correlated frames. Therefore, the correlated frames need to be filtered out from a given video. We first test whether the subjective visual similarity of images can be captured by cosine similarity between their low-dimensional representations obtained in step 1 of the similarity computation. As can be seen in Figure 4, it captures the visual similarity well enough.

To extract n uncorrelated frames from each video, we run k-means clustering, where $k = n$, on the low-dimensional representations and take the most similar frame to every centroid of the resulting clusters. This simple heuristic produces uncorrelated images.

To conclude this section, if the low-dimensional representation of individual images obtained by the autoencoder reflects the similarity between the captured objects and not some other irrelevant factors, we may expect to obtain meaningful clusters. Moreover, note the benefit of creating the similarity graph of videos instead of individual images. All images in one video are automatically linked together. If a few images in 2 videos are similar, this similarity is propagated to other frames within the video,

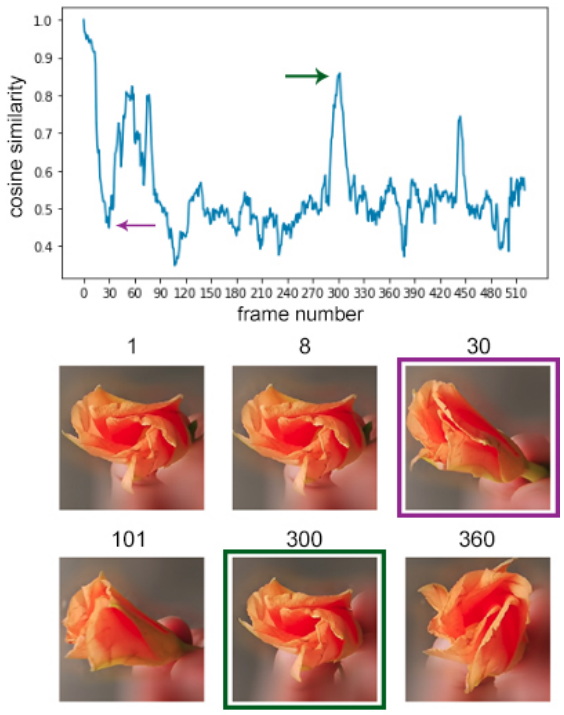

Figure 4: ***Top***: A graph of cosine similarity between the first and other frames in the video. ***Bottom***: Selected frames from the video with their corresponding frame numbers. The two highlighted frames correspond to two arrows in the graph.

Figure 5: Samples from the Organic Objects dataset. Images are cropped and have blurred background.

which would otherwise not be linked based only on the similarity. The video contains a constraint that says that the object cannot change its class in time.

5 Experiments

To test the algorithm for object discovery, we assembled a custom dataset of organic objects. The dataset contains 18 classes of organic objects, some of which are depicted in Figure 5. We have chosen organic objects because they naturally produce large variability between instances. For every class, we capture ten different samples from different viewpoints. The final dataset can be downloaded at the following address – `github.com/Jan21/`

`Organic-objects-dataset`.

Using our ensemble described in Section 3, we segment the object in every frame of each video. Using the resulting bounding boxes and segmentation masks, we crop each image and blur the background to suppress the distinctive features present in the background.

To obtain the low-dimensional representations used to filter out correlated frames and construct the similarity graph, we resize all images to a fixed resolution of 64×64 pixels and train a convolutional autoencoder. The autoencoder has five convolutional layers (16, 32, 64, 128, 256 filters with stride 2) and one fully-connected layer[7] with dimensions $1024 \rightarrow 96$. [8]

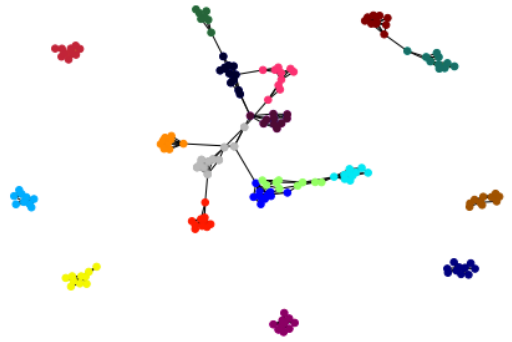

Figure 6: Visualization of detected communities in the Organic Objects dataset with the Louvain method. Nodes are colored according to the component (community) they are assigned to. The method discovered 18 components which belong to 18 different classes.

5.1 Results

After running community detection on the similarity graph, we inspected how many image bundles were assigned to the wrong component. Out of 173 videos, only 5 in the training set were assigned to the wrong component. The result of community detection on the constructed similarity graph is shown in Figure 6. Numerical results and comparison with clustering of non-segmented images are presented in Table 1. From the accuracy and number of discovered classes, it is clear that background removal created a significant accuracy difference of 56.1% and a difference in the number of correctly discovered classes.

6 Conclusion

In this contribution, we present a method for generic object detection and segmentation, which uses an ensemble of three different models trained by three different objectives. To demonstrate the effectiveness of the approach, we have created a custom dataset of organic objects. The dataset was used in our pipeline to remove background, create low-dimensional representations, and perform class discovery and classification. We have shown that background removal significantly increases the accuracy of the classification and the number of correctly discovered classes.

In future work, we plan to optimize our approach for speed, generalize it to work with videos containing multiple objects, and make the class discovery an online process that can discover new classes on-the-fly.

[7] Image is downscaled to 2×2 times 256 filters $\rightarrow$ 1024 input vector to the fully-connected layer.

[8] We also tried to extract representations by using VGG16 which was pre-trained on ImageNet. These representations better discriminated very similar objects (e.g., two types of red flowers).

References

[1] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," *arXiv preprint arXiv:2104.14294*, 2021.

[2] X. Lu, W. Wang, C. Ma, J. Shen, L. Shao, and F. Porikli, "See more, know more: Unsupervised video object segmentation with co-attention siamese networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3623–3632, 2019.

[3] Q. Xie, E. Hovy, M.-T. Luong, and Q. V. Le, "Self-training with noisy student improves imagenet classification," *arXiv preprint arXiv:1911.04252*, 2019.

[4] H. Bagherinezhad, M. Horton, M. Rastegari, and A. Farhadi, "Label refinery: Improving imagenet classification through label progression," *ArXiv*, vol. abs/1805.02641, 2018.

[5] P. Bhattacharjee and S. Das, "Temporal coherency based criteria for predicting video frames using deep multi-stage generative adversarial networks," in *Advances in Neural Information Processing Systems*, pp. 4268–4277, 2017.

[6] I. Davidson and S. Ravi, "Clustering with constraints: Feasibility issues and the k-means algorithm," in *Proceedings of the 2005 SIAM international conference on data mining*, pp. 138–149, SIAM, 2005.

[7] S. Basu, M. Bilenko, A. Banerjee, and R. J. Mooney, "Probabilistic semi-supervised clustering with constraints," *Semi-supervised learning*, pp. 71–98, 2006.

[8] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," *arXiv preprint arXiv:2103.13413*, 2021.

[9] Z. Teed and J. Deng, "Raft: Recurrent all-pairs field transforms for optical flow," in *European Conference on Computer Vision*, pp. 402–419, Springer, 2020.

[10] Q. Wang, L. Zhang, L. Bertinetto, W. Hu, and P. H. Torr, "Fast online object tracking and segmentation: A unifying approach," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1328–1338, 2019.

[11] P. Yakubovskiy, "Segmentation models pytorch." `https://github.com/qubvel/segmentation_models.pytorch`, 2020.

Pre-segmenting the objects	Number of discovered classes	Accuracy
Yes	18	97.1%
No	15	41.0%

Table 1: Comparison of the clustering of videos with and without the background removed. The accuracy is computed by checking how many times a video was assigned to incorrect cluster.

[12] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[13] N. Hansen, "The cma evolution strategy: a comparing review," *Towards a new evolutionary computation*, pp. 75–102, 2006.

[14] J. Hula, "Unsupervised object-aware learning from videos," in *2020 IEEE Third International Conference on Data Stream Mining & Processing (DSMP)*, pp. 237–242, IEEE, 2020.

[15] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre, "Fast unfolding of communities in large networks," *Journal of statistical mechanics: theory and experiment*, vol. 2008, no. 10, p. P10008, 2008.