Eidgenössische Technische Hochschule Zürich
Computer Vision Laboratory

Video Object Segmentation by Tracking One Point

---

by Alberto Montes Gómez

Advisors: Dr. Jordi Pont-Tuset, Kevis Maninis, Sergi Caelles
Director: Prof. Luc Van Gool
Zürich, April 2018

Abstract

Acknowledgements

In the first place, I would like to thank Dr. Jordi Pont-Tuset for allowing me to do my master thesis with the segmentation group at Computer Vision Lab. In extension I would like to thank Kevis Maninis and Sergi Caelles for their advisory work. I am very gratefully to them all for their patiente and guidance during this Master Thesis. They also have teached me how to do research in the Computer Vision field.

From ETH Zürich, I would like to thank Prof. Luc Van Gool for giving me the opportunity to perform research in the Computer Vision Lab. In addition I would like to thank all the professors I had from the courses taken, from which I have learned a lot.

From UPC Barcelona, I would like to mention Prof. Xavi Giró-i-Nieto, Amaia Salvador and Santiago Pascual from the Image Processing Group. Was with them, when I was performing my Bachelor Thesis, that I was introduced to Computer Vision and Image Processing and they supported me to pursuing my studies in this field in ETH Zürich.

I would also like to thank my girlfriend Judith Bergadà for all his support given to me during the development of my studies and this thesis and pushing me forward.

Finally, I would like to thank my family for their constant support and motivate me during my studies.

Contents

1 Introduction

During the last decade Computer Vision has grown in popularity as a tool to perform tasks as humans do. Moreover, with the introduction of deep learning techniques, which try to mimic how the human brain works, the field has experienced an explosion on speed and performance which has fostered a lot of research in the topic. Furthermore, many products currently already avaiable use these techniques in products such as image and video processing software, video classification at video portals or language translation. Thus, this is an exciting field to work nowadays as new results can be around the corner and can lead us to great new products.

This thesis focuses on the problem of video instance segmentation. We refer to segmentation as the partition of pixels from a video in different classes. These classes will be different instances appearing in the video plus the background. This is an important task that has been tried to be solved in the Computer Vision field during the lasts years and can be a core block for numerous applications like video editing and post-processing, video retrieval activity recognition and much more.

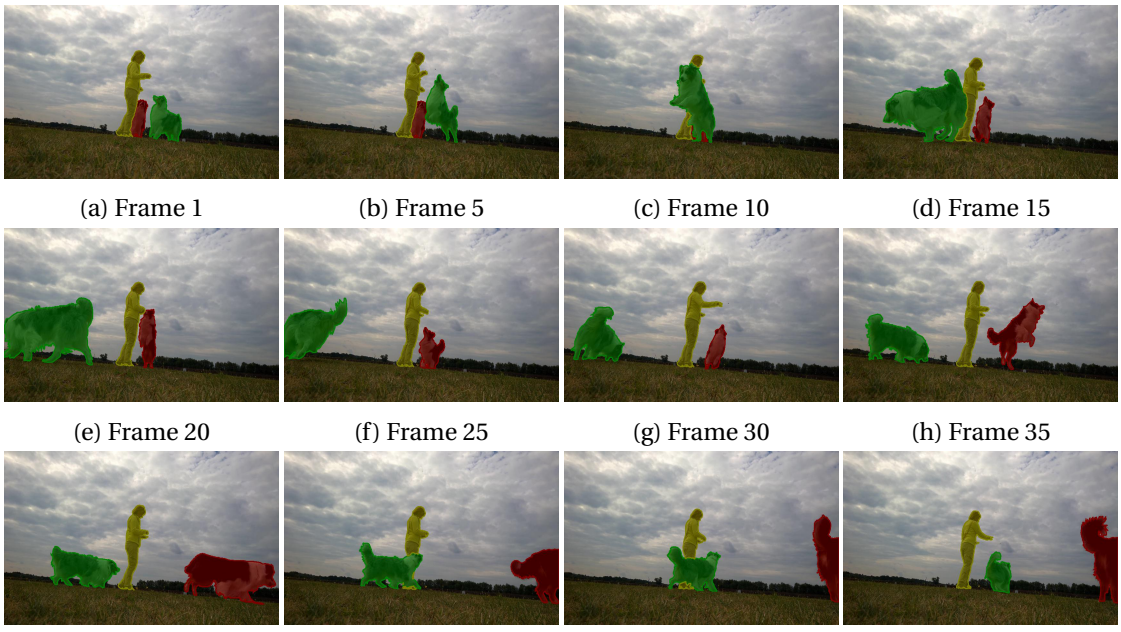
In previous works, there has been mainly two different approaches to tackle this problem, unsupervised or semi-supervised. In the unsupervised methods, the model tries to infer which instances or objects are the protagonists during the whole video and then output a segmentation mask for each instance. On the other hand, in the semi-supervised methods, the instances that we want to segment are given to the model in the form of the mask in the first frame. Thus, the model has already information about which instances must be segmented and can propagate this initial mask through all the sequence to output a prediction.

Our proposed method during this thesis tries to solve the problem of video instance segmentation from a new perspective. We call it a "weakly semi-supervised" approach which instead of using the whole mask of the instance to segment the whole video, a single point per instance is given. This approach present some benefits when applied to real user applications as points annotations are more easy and fast to performn than full mask segmentations.

Our proposed method performs two tasks and fuses them to obtain a final prediction: point tracking and instance segmentation from point. With point tracking, we are able to keep track at every frame of a point per each instance. In addition to this, a model has been trained to learn an embedding representation that can be used, given a point and its label, to label the rest of the pixels with a similarity computation. A combination of this two tasks gives us a method that is capable of performing the task of weakly semi-supervised video instance segmentation.

In order to train and evaluate our method, a dataset is required. To evaluate our method, we have used the the DAVIS [PPTM+16, PTPC+17] dataset which provides instance segmentation

masks for multiple video sequences. This dataset is well know in the research community for being a benchmark to evaluate video segmentation models. In Figure 1.1 we can see a sequence and its mask annotation overlaid to it for each instance in the video.



(a) Frame 1 (b) Frame 5 (c) Frame 10 (d) Frame 15

(e) Frame 20 (f) Frame 25 (g) Frame 30 (h) Frame 35

(i) Frame 40 (j) Frame 45 (k) Frame 50 (l) Frame 55

Figure 1.1: *Dogs Jump* annotated video sequence with all the instance masks overlaid in different colors.

In addition, the PASCAL [EVGW+10] has been used. It provides segmentation masks for different object instances in images. This dataset has helped us to train the embedding model to be able to obtain a good pixel representation that know the difference between two instances of the same class.

The document is structured in the following way. First, in Chapter 2, the current state of the art techniques in image and video instance segmentation are reviewed in order to have an idea of which are the advantages and drawbacks of the other methods. Then, in Chapter 3, our proposed method is defeined and explained in detail. After that, in Chapter 4, the different experiments to test our method are detailed in addition to its results. Finally, in Chapter 5, the conclusions about the benefits and drawbacks of our proposed method are explained.

2 State of the Art

In this chapter, we summarize the most relevant work on instance segmentation. First, we focus on tracking implementations and later, we talk about the state of the art methods for instance segmentation and some implementations applied to videos.

2.1 Tracking

Tracking implementations are mainly focus on video object tracking. This consists in taking the bounding box sorrounding an object and make bounding box predictions throught the video that contains the object to track.

One state of the art implementation is MDNet [NH16] . It proposes a Convolutional Neural Network with shared layers and multiple branches of domain-specific layers. The architecture overview is showed at Figure 2.1. Each branch is responsible for binary classification to identify the target in each specific video. This domain-specific layers are updated online and the online tracking is performed by evaluating the candidate windows randomly sampled around the previous target state. This showed state-of-the art results on tracking benchmarks and was the winner of the VOT Challenge [KML+16] in 2016.

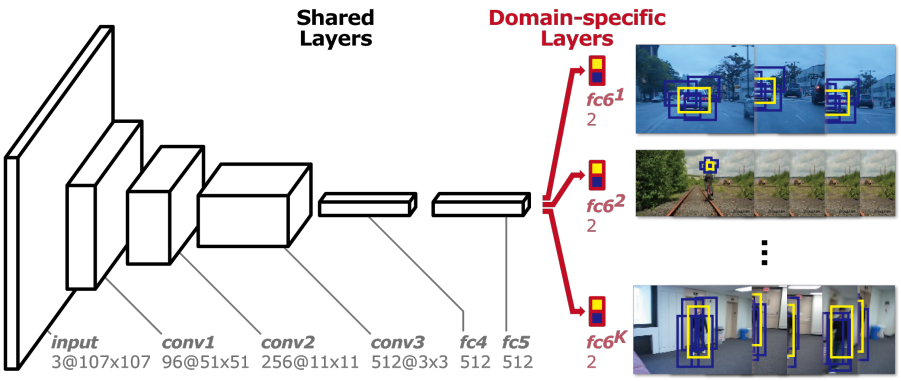


Figure 2.1: MDNet [NH16] architecture.

Unfortunetly there are not implementations that focus on keypoints tracking which is the idea behind this thesis.

2.2 Instance Segmentation

Instance detection and segmentation is one of the most challenging tasks in Computer Vision right now. In this section we explore some of the works that tackle instance segmentation in different domains like image and video.

Mask R-CNN [HGDG17] This work come from a series of works that started with object detection on images using Convolutional Neural Networks. Afterwards, they iterate the architecture to make it differentiable, being able to train it end-to-end. The last iteration consisted on predicting not only the bounding box of the detected object but its mask. The idea behind this method consists on a CNN that with a forward pass extract the features at the last convolutional layer. Then a set of bounding box proposals are generated and the features in each proposed bounding box are used in one branch to predict a objectiveness score and bounding box regressions. With this branch a set of bounding box are obtained and score predicted. With this values, a second branch is responsible to predict a mask from each proposal. The final prediction use maximum supression to reduce the proposals to the final prediction masks. This approach show state of the art results on multiple datasets at the cost of using a very large amount (in the order of 1000) proposals per a single image. Some qualitative results can be found on Figure 2.2.

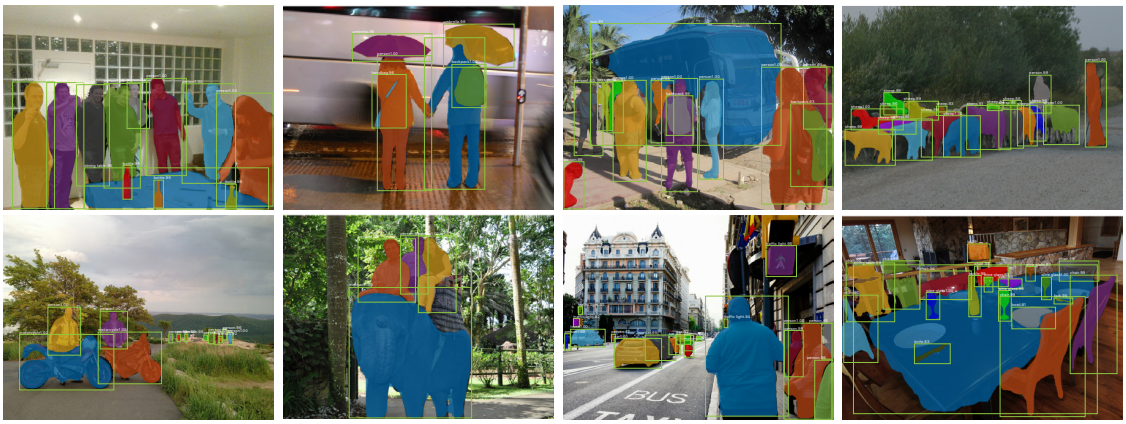


Figure 2.2: Mask R-CNN [HGDG17] results.

Deep Metric Learning [FWR+17] This works also focus in instance object segmentation on images. They use a Convolutional Neural Network to predict an embedding representation per each pixel. This network is trained using metric learning, regressing how likely two pixels are to belong to the same object. Then they propose the use of seed points and its pariwise similarity to predict the instances and its masks on an image once the embedding for each pixels is computed. Some results of the segmentations can be found on Figure 2.3.

Figure 2.3: Deep Metric Learning [FWR+17] instance segmentation results.

OSVOS [CMPT+17] and OnAVOS [VL17] This works focus on instance segmentation on video. Their main approach consists on train a parent network that given a frame is able to predict the mask of the foreground (all object instances in the image) and then finetune a network for each video. This finetune is performed given the mask of the first frame (this methods tackle semi-supervised video object segmentation) and then the trained network for each video is

used to predict the masks for the rest of the frames. The OnAVOS [VL17] work in this last step add an online training. It makes predictions of the frames and after each frame prediction, the network is online trained with the new predicted mask. This two implementations present the state of the art results on the DAVIS [PPTM+16, PTPC+17] dataset.

MaskRNN [HHS17] This work proposes a recurrent neural network approach which fuses in each frame of the video the output of two deep nets for each objects instance. The first net is a binary segmentation network providing a mask and the second one is a localization network providing a bounding box. Thanks to the recurrent component and the localization component, this method is able to take advantage of long-term temporal structures of the video as well as rejecting outliers. This method also provide competitive results with the previous works of OSVOS [CMPT+17] and OnAVOS [VL17] .

3 Methods

In this chapter we are going to explain the different methods used during this thesis. First, we are going to explain the backbone architecture used during the whole thesis. Then, we are going to explain how we performed a model to track points. The last section will explain how metric learning works and how we have use it.

3.1 Backbone Architecture

The objective of this thesis has been to obtain a deep learning model which is capable of, given a point per instance, being able to predict its segmentation through all the videos. As it is usual on the literature, the best deep learning models for computer vision tasks (and more precisely in segmentation) are the ones that are fully convolutional neural networks. This models perform convolution operations over the input which is usually an image.

During the development of this thesis, we have used as a backbone architecture the DeepLab [CPK+18] model for semantic segmentation. This model is fully convolutional and it is based on the ResNet [HZRS16] architecture which was used for image classification. The DeepLab [CPK+18] model what does is to keep the convolutional layers from ResNet [HZRS16] and add at the end some deconvolutional operations to obtain as an output a mask instead of an object classifier. Doing this allows to be the model fully convolutional and free the constrain about the input images size's.

The ResNet architecture is based on layers with residual connections. In Figure 3.1 is shown how each bottleneck layer block is built. Each layer applies some convolutional operations to the input and then adds this result to the original input as a residual.

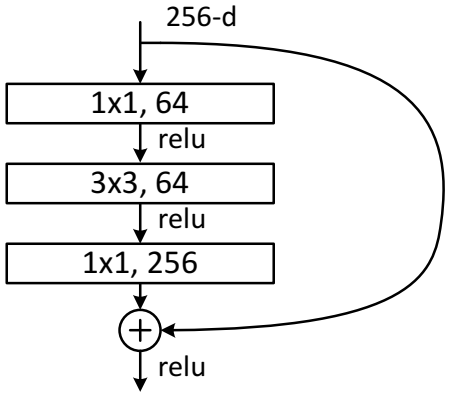


Figure 3.1: ResNet [HZRS16] residual block architecture.

For a more detailed description of the whole architecture of the DeepLab [CPK+18] convo-

lutional neural network see Table 3.1, where two versions used are described: ResNet50 and ResNet101. The architecture takes as input images of size 512×512 and outputs the input size downsampled by 8 both in height and width. Also, note that the number of output channels is 2048, so this output can be used directly, or a header can be plugged to reduce the dimensionality (in the case of classification, reduce the dimensionality to the number of classes). Because of these, this architecture is very versatile and easy to use with images.

layer name	output size	50-layer	101-layer
conv1	256×256	7×7, 64, stride 2	
conv2_x	128×128	3×3 max pool, stride 2	
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	64×64	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$
conv4_x	64×64	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$
conv5_x	64×64	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$

Table 3.1: Architectures for DeepLab [CPK+18] using residual blocks. Building blocks are shown in brackets (see also Figure 3.1), with the numbers of blocks stacked.

3.2 Tracking

In order to track points through a video sequence, one possible approach is to regress the point position in an image using a heatmap. The point representation is inspired by [NYD16] which used heatmaps to predict keypoints positions. In our experiments, we are going to use the DeepLab ResNet architecture described in Section 3.1 plus an additional module, a PSP [ZSQ+17] module that will reduce the output to a single channel mask to represent the heatmap using a pyramid scene parsing. This will allow us to regress a heatmap representation of a point using the Mean Squared Error loss function.

The heatmap is built using the coordinate of a point $p = (p_x, p_y)$ and computing a guassian around this point. In Equation 3.1 is shown how to build a heatmap $\mathscr{H}$ which allows some customization with the parameter σ which is the spreadness of the sigma. Note that the values of the heatmap are bounded between $[0, 1]$. To observe a graphical example, in Figure 3.2 there is a point annotated on an image and the resulting heatmap.

$$\mathscr{H}(x, y) = \exp\left[-4\log(2)\frac{(x-p_x)^2 + (y-p_y)^2}{\sigma^2}\right] \tag{3.1}$$

The strategy to train a model that tracks a point given in the first frame will be the same used in OSVOS [CMPT+17] . We finetune a model pretrained with Visual Object Classes over the first

Figure 3.2: Point representation. **Left**. Image with the annotated points (image resolution 854 × 480 pixels). **Right**. Heatmap to represent the point with $\sigma = 32$ pixels.

frame of each sequence. Adding some data augmentation to enrich the training, we will test each sequence models with the rest of the frames and extract the maximum of the predicted heatmap as tracked object.

3.3 Metric Learning with Triplet Loss

Another approach to track a point in a video sequence could consist on training a model which outputs an embedding for each pixel that can be meaningful to obtain instance segmentation. In order to obtain a good embedding, metric learning has been used which consists on the task of learning a distance function over objects, which in our case are embeddings.

One way to apply metric learning is using a Triplet Loss which was first introduced in [BRPM16] . One famous implementation of this Triplet Loss was used in FaceNet [SKP15] where they train a model to embed face images and predict similarity between them.

The objective of the Triplet Loss consists on learning a distance difference between triplets of points. The triplet consists on three points that are called: anchor, positive and negative. It must fill the condition that the label of the anchor and the positive point are the same, while the label of the negative is different. The objective of the Triplet Loss is push the distance of the anchor and positive points to be closer than the distance between the anchor and negative by some margin α. In Figure 3.3 a diagram about the learning procedure is shown.

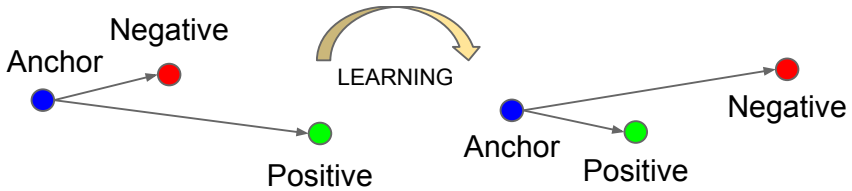


Figure 3.3: Graphical representation of Triplet Loss learning. Figure extracted from [SKP15].

In our escenario, we use DeepLab [CPK+18] explained in detail in Section 3.1. This has is pretrained for semantic segmentation, and we use the full network except the last classification part which predicts the pixels class. By default, this model given an image of size $H \times W$ output a features map of size $H' \times W' = \frac{H}{8} \times \frac{W}{8}$ with 2048 dimensions. A convolutional head can be plugged to reduce this dimensionality to a lower dimension D which may lead to a better embedding.

Before giving the equations of the Triple Loss, lets define some notation. Be $\mathscr{I} \in \mathbb{R}^{H \times W}$ the input image of size $H \times W$. The output size of the model will be $H' \times W' = \frac{H}{8} \times \frac{W}{8}$ and the

output embedding will be $\mathscr{X} = \{x_i\}^{H' \times W'}$ where $x_i \in \mathbb{R}^D$ is the i-th output pixel embedding. Let be the Convolutional Neural Network be the embedding model $f(\mathscr{I};\Theta)$ which computes the embedding from a single image:

$$\mathscr{X} = f(\mathscr{I};\Theta) \tag{3.2}$$

When training the model, there is available the ground truth mask of one object instance in the image. When the image is forwarded through the model and the embeddings computed, N triplets of pixels are sampled. For each triplet, two pixels belonging to the mask are sampled and assigned as anchor and positive pixels (x_a and x_p respectively). Finally, a pixel not belonging to the mask is sampled and assigned as negative pixel x_n.

$$\mathscr{L}_{triplet} = \sum_{i}^{N} \max\left(d(x_a^{(i)}, x_p^{(i)}) - d(x_a^{(i)}, x_n^{(i)}) + \alpha, 0\right) \tag{3.3}$$

The α term on the Triplet Loss is the margin used to enforce the minimum difference between the distance of positive pairs and negative pairs. d is the distance function, which in our case the l^2 norm is used. So Equation 3.3 will end up as follows:

$$\mathscr{L}_{triplet} = \sum_{i}^{N} \max\left(\|x_a^{(i)} - x_p^{(i)}\|_2 - \|x_a^{(i)} - x_n^{(i)}\|_2 + \alpha, 0\right) \tag{3.4}$$

This learned embedding can be very useful for example to implement a second approach for tracking. A similarity between pixel's embedding in the next frame and the embedding of the point to track in order to obtain its location on the next frame.

3.4 From Embedding to Segmentation

Once we have learned a good embedding for each pixel of the image, the embedding can not give us directly the segmentation of every instance. As our approach will consists on tracking a point for every instance in the video sequence, an already annotated pixel will be provided. This pixel, we are going to call it keypoint and it has an associated embedding $x_k \in \mathbb{R}^D$. This point can come from the ground truth, from tracking or from another frame annotation.

In order to obtain the segmentation for multiple instances in an images given a keypoint for each instance, we can compute the distance of every pixel embedding against all keypoint's embeddings. Then, every pixel will be assigned to the label of the closest keypoint embedding only if this distance is lower than a threshold γ, otherwise will be assign to background. All of these can be formulated mathematically taking into account and image embedding $\mathscr{X} = {x_i}^{H \times W}$ with N instances and each instance with a keypoint embedding $x_k^{(n)}$. Every keypoint will have a label $m_k^{(n)} \in [1, N]$ and label 0 will belong to the background. The distance map for every image will be:

$$\mathscr{D}_i = min_n\left(d(x_i, x_k^{(n)})\right) \tag{3.5}$$

Then the prediction label can be computed as follows:

$$\mathcal{M}_i = \begin{cases} \arg\min_n d(x_i, x_k^{(n)}) & \mathcal{D}_i \leq \gamma \\ 0 & \mathcal{D}_i > \gamma \end{cases} \tag{3.6}$$

All this steps can be visualized in Figure 3.4 where the ground truth mask and a keypoint per each dog instance, the distance map of the embeddings against the keypoint embedding and finally the predicted mask are displayed.

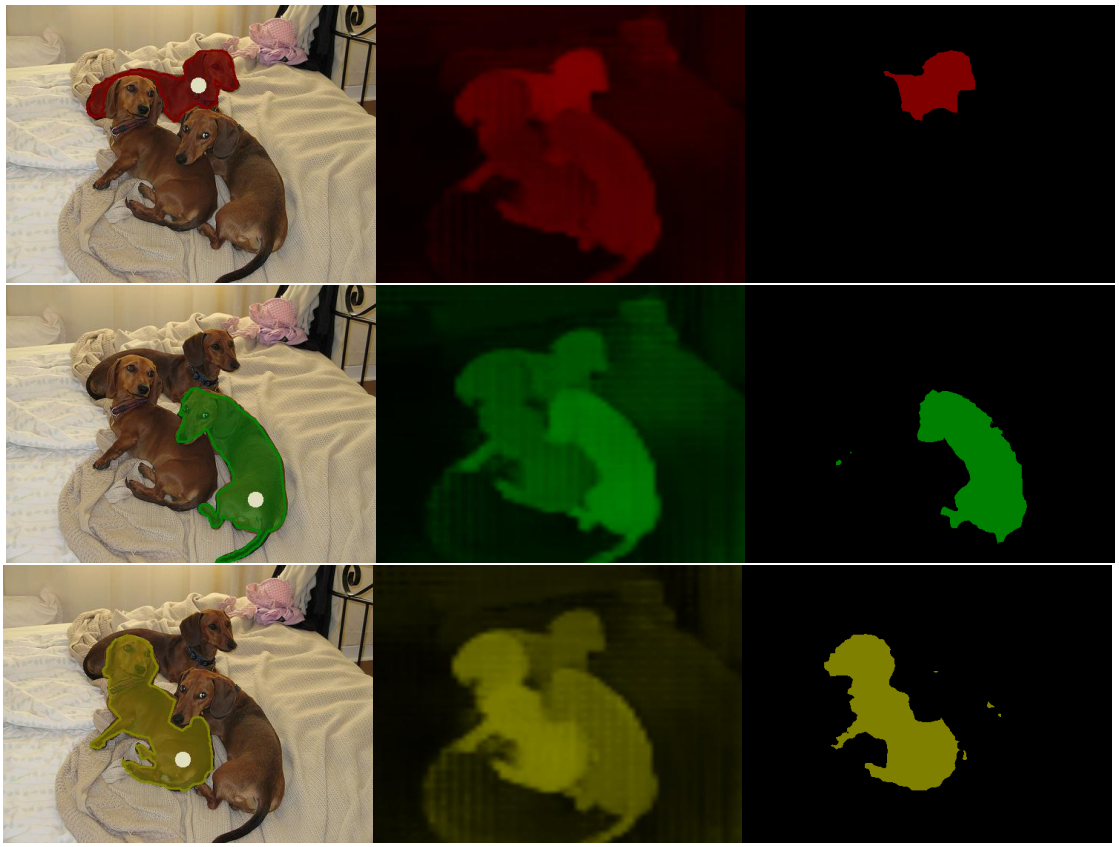


Figure 3.4: Example of instance segmentation procedure given a point per instance. **Left**: Ground truth annotation of the instance and keypoint given. **Middle**: Distance map of the every pixel's embedding versus keypoint embedding (higher intensity means lower distance, dark means higher distance). **Right**: Predicted mask once applied the thresholding.

4 Experiments and Results

In this chapter is going to be explained the setup required to run the experiments, the datasets used during this experiments and its numerical and qualitative results.

4.1 Setup

4.1.1 Workstation

The main developing tool has been a laptop which has been used for a huge variaty of tasks: write code, debug, check papers and documentation, write notes, etc. In order to train and make prediction with deep learning models, an intensive computational resources are required.

This is why in addition, access to a GPU cluster has been provided in order to train models using GPUs. The Computer Vision Lab provide access to the BIWI cluster which has been used to store datasets and models and also as a computational resource. The BIWI cluster consists on 70 computational nodes (CPU only) with a total of 556 processors, 8328 cores and 13.3TB of RAM memory. In addition, there are the GPU nodes which consists in 75 GPUs with 12GB of GPU memory each one.

4.1.2 Software

All the code development has been done using the Python3 language. Python is commonly extended in computer vision and deep learning applications because all the available libraries that provides computational frameworks and also because its easiness developing.

The main library used to develop deep learning models has been PyTorch [PGC+17] . This library provides high-level features for tensor computation and deep neural networks. It is a wrapper in Python which under the hood is written in C allowing fast performance and strong GPU acceleration. This library is usually used in the research community because its flexibility, fast code prototyping and easily debugging capabilities.

4.2 Datasets

During the development of this thesis, two segmentation datasets have been used: DAVIS [PPTM+16, PTPC+17] and PASCAL [EVGW+10] . DAVIS [PPTM+16, PTPC+17] was used to evaluate the instance segmentation in videos while the PASCAL [EVGW+10] was used as additional data, as it provides a huge set of instance annotations on images.

4.2.1 DAVIS Dataset

The DAVIS Dataset [PPTM+16, PTPC+17] consists in a Densely Annotated Video Segmentation dataset. It provides a curated densely annotations for object instances in video sequences. There are two versions of the dataset: DAVIS 2016 [PPTM+16] and DAVIS 2017 [PTPC+17] . The 2016 version provides foreground/background annotations while the 2017 provides annotations for multiple objects and instances in the foreground. During the work done on this thesis, DAVIS 2017 [PTPC+17] version has been used. Some examples of the annotations are shown in Figure 4.1 and information about the dataset is also given in Table 4.1.

DAVIS 2017	train	val	test-dev	test-challenge	**Total**
Number of sequences	60	30	30	30	**50**
Number of frames	4219	2023	2037	2180	**10459**
Mean number of frames per sequence	70.3	67.4	67.9	72.7	**69.7**
Number of objects	138	59	89	90	**376**
Mean number of objects per sequence	2.30	1.97	2.97	3.00	**2.51**

Table 4.1: Size of the DAVIS 2017 data splits: number of sequences, frames and annotated objects.

Figure 4.1: First frame annotation for all the sequences in validation subset at DAVIS 2017 [PTPC+17] .

The DAVIS dataset owners also organize a challenge to evaluate the performance of different segmentation method. This challenge have two branches which are the following:

Semi-supervised Consists on generate a prediction given only the ground truth mask of the first frame. This gives information about which is the object intended to be segmented.

Unsupervised As the name specifies, no information is given about the object that must be segmented. To solve this, the segmentation methods rely only on the frames to infer the foreground and background.

As our goal consists on weakly semi-supervised instance segmentation using tracked points, some additional annotation was required. To obtain more data, which in this case was point trajectories inside each sequence, a manual annotation by the Segmentation Group at the Computer Vision Lab was performed. The annotation was done using an existing tool developed inside the group by Dr. Jordi Pont-Tuset. The annotation consisted on fixed points of the object along all the frames and also its visibility. An example of a fixed point in an object will be an animal's eye, a person point in the head or the center of a wheel.

4.2.2 PASCAL Dataset

PASCAL Dataset [EVGW+10] is a dataset used to benchmark vision object category recognition, detections and segmentation. Consist on images containing 20 visual object classes and provides multiple annotations for different computer vision tasks: detection and segmentation. The segmentation annotations consist on masks over instances belonging to the 20 classes, which lead with images with multiple instance annotations.

During this thesis, the segmentation annotations from PASCAL VOC 2012 have been used. The information about the number of instances per split can be found on Table 4.2 and some samples of the dataset at Figure 4.2 are shown.

PASCAL VOC 2012	train	val	**Total**
Number of images	1464	1449	**2913**
Number of instances	3507	3427	**6934**
Mean number of instances per image	2.40	2.37	**2.38**

Table 4.2: Size of the PASCAL VOC 2012 [EVGW+10] data splits: number of images and annotated objects.

4.3 Experiments

To test our proposed method we divide the experimentation in three steps. The first step consists in test the tracking of the points. The second one will consist on instance segmentation, once given a point belonging to an instance. The last step consists in evaluate on video sequences the weakly semi-supervised method for video instance segmentation.

4.3.1 Tracking

In order to perform tracking of single points in video sequences we have performed two approaches. The first approach is inspired by OSVOS [CMPT+17] where a model is finetuned for each sequence with the information of the first frame available. In our case the point representation as a heatmap is being used and tested against the DAVIS 2017 validation subset.

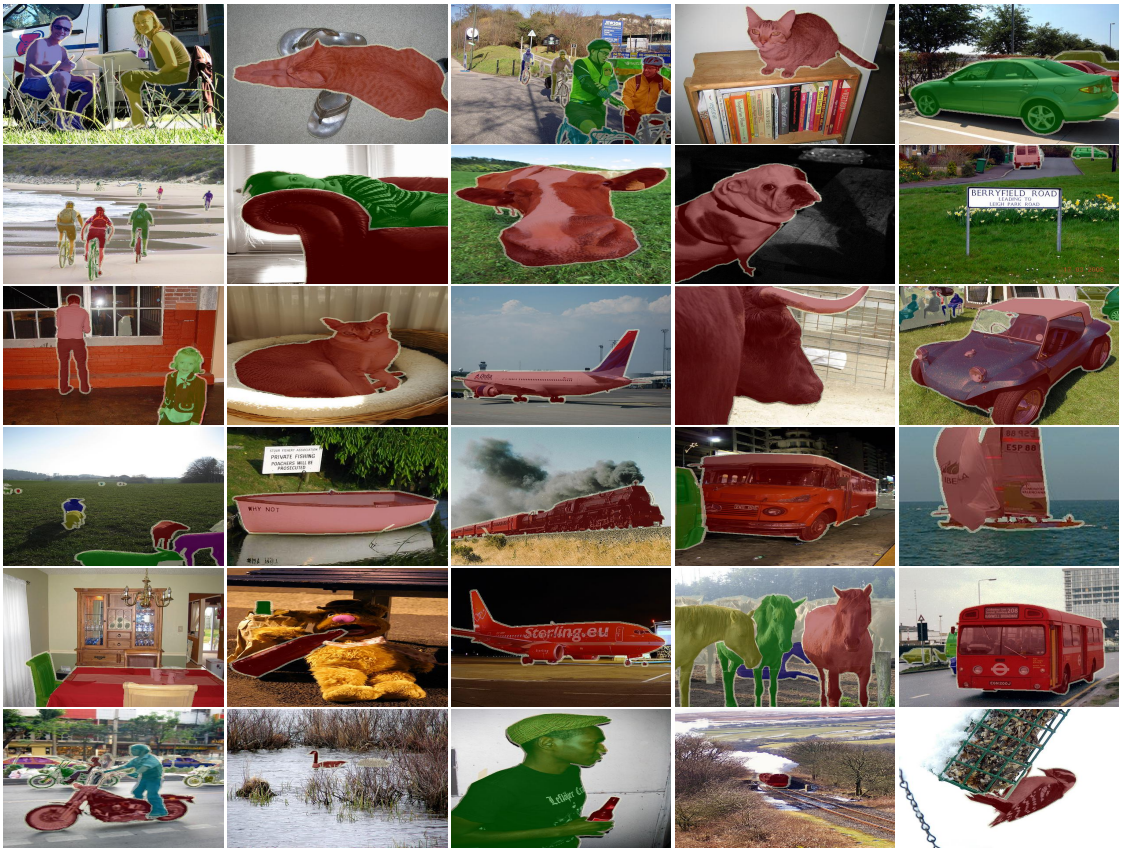


Figure 4.2: Example of images and annotations from PASCAL VOC 2012 Dataset [EVGW+10] .

Before presenting the results some metric should be defined. For point tracking PCK metric [ST13] can be used as it has been widely used in pose estimation to evaluate keypoint position prediction. This metric plots for a range of distance in pixels in the original resolution of the image (480p in our case), the detection rate. The detection rate can be seen as the percentage of samples that the prediction is as furthest the distance in pixels.

For the following experiments a ResNet50 backbone architecture has been used with a PSP module [ZSQ+17] plug at the end. No major difference was observed using ResNet101 architecture so ResNet50 was mainly used because of its lower training time as it has less parameters to train. The image data was fed to the model at the resolution of 512×896 pixels. As training parameters, a SGD optimizer was used with a learning rate of 10^{-6}, momentum 0.9 and weight decay 0.0005. The training for each sequence consisted on 3000 iterations with batch size 2.

To compare our results, we set two baselines to compare with. The first one consist on a simple baseline where the predicted point at each frame of the sequence is the same as the point given at the first frame. The second baseline consisted on running the MDNet [NH16] tracker on the smallest bounding box that fit the ground truth mask of the sequence. The testing was performed comparing the center of the predicted bounding box against the ground truth bounding box.

We tested two variations of the model. The first one using a single channel prediction trained with a heatmap of $\sigma = 40$ pixels. The second variation consisted in a multiscale heatmap. In

the experiment we trained using three channels with a heatmap in each one of σ equals to 24, 48 and 96 respectively. The prediction of the point at all time was performed computing the position of the maximum value. In the case of the last configuration, all the three channels were summed along the channel dimension before predicting the point position. The results are plotted on Figure 4.3.

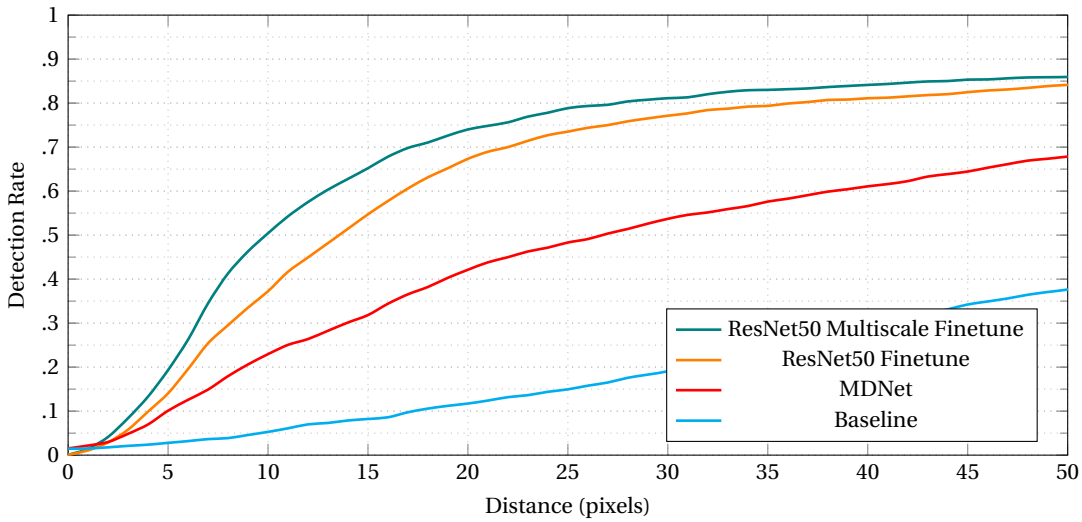


Figure 4.3: Precision performance for point regression.

In addition to this metric, in order to see if our model has been able to learn the representation of the object to track, we compute a coverage factor. This coverage factor is defined as the accuracy of the model to predict a tracked point inside the ground truth mask of the object to track. These results are showed at Table 4.3.

Method	Coverage
Baseline	0.428
MDNET	0.851
ResNet50	0.912
ResNet50 Multiscale	**0.940**

Table 4.3: Coverage for different methods that regress the heatmap.

This experiments presents good results in localization precision and coverage (at least we know that if the tracked point is not precise is consistent with the object to track). The big setback is regarding the training model. We need to train a model per each sequence which may lead to a big prediction time for new sequences (for the order of 10-30 minutes per sequence).

For this main reason and in order to obtain a single model to be not dependent on the sequence to track, we try using unsupervised learning via metric learning. This is the second approach we tried. Given the ResNet model, we tested this architecture with a pre-trained weights, performing some training with the Triplet Loss and expanding this architecture with a convolutional head to reduce the embedding dimensions.

The training was performed using Adagrad optimizer with learning rate 0.001 for 100 epochs over PASCAL [EVGW+10] dataset with batch size 8. The Triplet Loss margin was set to $\alpha = 0.7$

and 64 triplets were sampled per image. It was tested with the original embedding dimension $D = 2048$, and then adding a convolutional head to reduce its dimensionality. $D = 128$ was the dimensionality that presented better results. For this experiments the backbone architecture used was ResNet101 as was the only one available with pretrained weights on semantic segmentation.

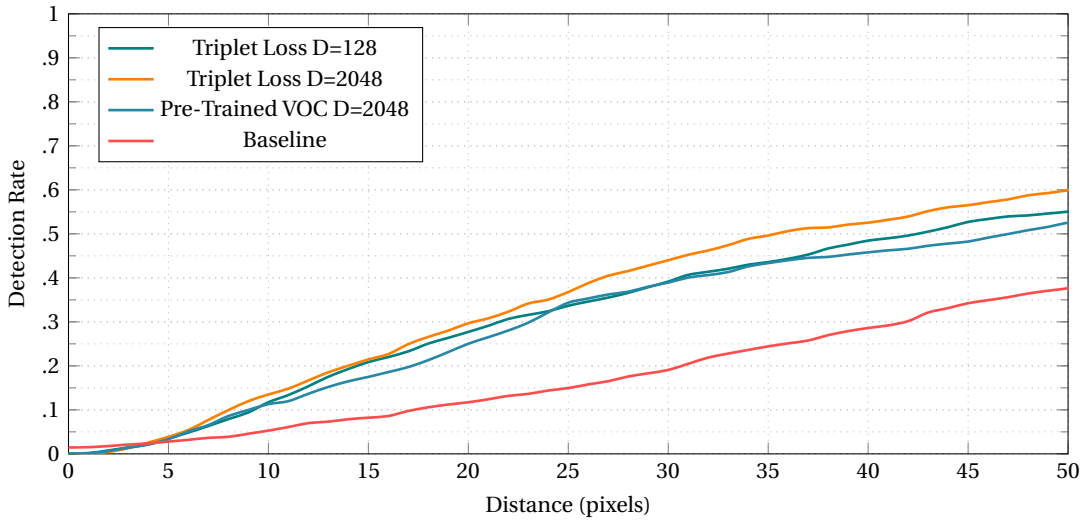


Figure 4.4: Precision performance for pixel representation tracking.

As can be seen on Figure 4.4 the localization precision is really poor, even in comparison with the previous experiments. In contrast, the coverage results available in Table 4.4 increase as long as training is being performed and the dimensionality is being reduced achieving in the last configuration almost the same result with a dedicated network for each sequence. Note that this evaluation is performed on DAVIS [PTPC+17] validation while the trained model has not seen any information from this dataset as it has been trained with the PASCAL [EVGW+10] dataset.

Method	Coverage
Baseline	0.428
Pre-Trained VOC $D = 2048$	0.620
Triplet Loss $D = 2048$	0.750
Triplet Loss $D = 128$	**0.911**

Table 4.4: Coverage for different tracking methods trained with metric learning.

From the last model trained using the Triplet Loss at $D = 128$, we can conclude that the embeddings obtained for each pixel are representative for the instance the pixels belongs to, but not much from the specific point of the object. Even though the tracking performance is not good enough, we think that the good coverage and the good embedding quality can make this architecture valid for both tracking and segmentation using a single model which is sequence independent. That is the reason why we move forward with this model on the coming experiments.

4.3.2 Instance Segmentation

We wanted to test the model trained using the Triplet Loss applied to instance segmentation. Our purpose is to obtain the instance segmentation given a point per instance. To test the instance segmentation, the PASCAL [EVGW+10] dataset has been used. The point used to test the segmentation has been chosen computing the distance to the ground truth mask contour and take the coordinate which is furthest. This point will have an embedding representation that is descriptive enough for the instance to be segmented.

As the Triplet Loss is a unsupervised training no metric is used to check the training in addition to the validation loss. Nevertheless, a qualitative visualization can be performed to check the quality of the embeddings obtained. To visualize it, for each image the embeddings have been computed and then a PCA decomposition has been computed over the embeddings of all pixels. The decomposition is perform to reduce the dimensionality into 3 dimensions and then scale the values of this 3 dimensions to obtain a RGB image. In Figure 4.5 are shown some examples of PASCAL [EVGW+10] validation images. On this projection, in most of the cases we can observe how different color tones show differences between instances even for the same class.

The next step is to evaluate the instance segmentation given a point per instance. As there exists a one-to-one matching between the ground truth masks and the predicted masks, the evaluation metric used will be the Mean Intersection over Union (mIoU). This is computed as follows, being $\mathscr{P}$ the set of pixels which the prediction mask is 1 and $\mathscr{G}$ the set of pixels which are annotated to 1:

$$mIoU = \frac{|\mathscr{P} \cap \mathscr{G}|}{|\mathscr{P} \cup \mathscr{G}|} \tag{4.1}$$

The segmentation given a point is computed as it is specified in Section 3.4. Given a point, we obtain the embedding of the pixel x_k. The next step is compute a distance metric between this pixel embedding x_k and the rest of the pixels in the image $\mathscr{X}$. We have observed that the distance function (between euclidean distance and cosine distance which are commonly used for embedding similarity) do not present any difference on the results. Because of this in all future steps euclidean distance will be used.

Then, given a distance map, a simple thresholding operation will be able to return the mask for each instance. Some examples of this procedure are shown at Figure 4.6 where the image and the ground truth are show at the left column. On the middle column are shown the distance maps for different instances in the image and in the right column there are the predicted masks after thresholding the distance maps. For this example, the threshold used has been $\gamma = 0.5$.

We have observed that the value of the margin α used at training with the metric loss have impact on the performance of the segmentation. In order to test the results for different margin values α, we have build the Table 4.5. It can be observed how the best value to train the model and obtain a good segmentation is $\alpha = .7$.

Margin α	.5	**.7**	.8	1.0	1.2	1.5	2.0
$mIoU$	.6351	**.6464**	.6385	.5620	.5825	.4834	.3977

Table 4.5: Mean intersection over union against the Triplet Loss margin α used to train (ResNet101 with a convolution head and $D = 128$).

4.3.3 Video Instance Segmentation

The last experiment is related with fusing both previous approaches. We have tested the performance of the instance segmentation on the video dataset DAVIS [PTPC+17] 2017 over the sequences at the validation subset. Out approach is going to be compared against the two available methods tested against the DAVIS [PTPC+17] 2017 dataset which have made their predictions publicly available. These are OnAVOS [VL17] and OSVOS [CMPT+17] . We compare this two methods against different variations of ours.

The first comparison is against the segmentation using perfect keypoints (keypoints obtained from the ground truth mask) which in this case the segmentation method is being tested. The second approach is segment each frame of the sequence using the embedding of the keypoint given in the first frame. The third and last configuration is find at each frame the pixel which has higher similarity to the annotated pixel in the first frame. Then for each frame, with this found keypoint tracked from the first frame, segment the frame as explained in Section 3.4. The results in terms of mIoU are shown in Table 4.6.

Method	mIoU
OnAVOS [VL17]	0.616
OSVOS [CMPT+17]	0.566
Perfect Keypoints	0.610
First Frame Keypoint	0.426
First Frame Keypoint tracked	0.432

Table 4.6: Comparison on overall performance for different methods and our different approaches.

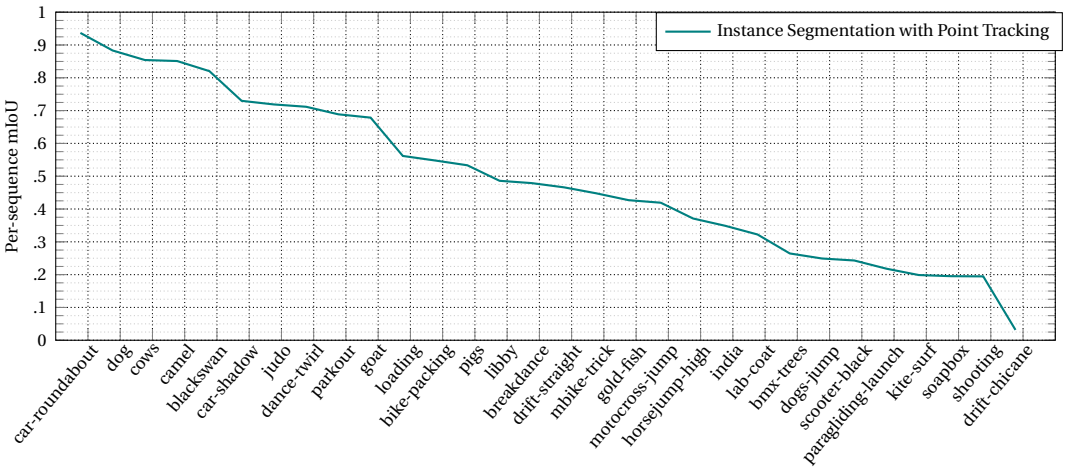


Figure 4.7: Per sequence results on DAVIS 2017 validation.

As can be seen on the Table 4.6, the segmentation taking into consideration that we are computing it from the best pixel, leads to performance compared with the state of the art. In contrast, when only the information from the first frame is used, the performance drops drastically. Nevertheless, the tracking operation seems to give a little boost in comparison with only use the first frame embedding.

To dig a little bit more on the results, the performance per sequence when tracking the keypoints is shown at Figure 4.7. Here can be observe the big disparity of results between sequences. It seems that the biggest problems are present on sequences where the object on the first frame start being very distant and small. During all the experimentation we have experienced some difficulties with embeddings of small objects and also embeddings very close to the contour so very thick objects (e.g. bicycles) present very bad performance. In addition, on Figure 4.8 are plotted the prediction of the mask for different sequences.

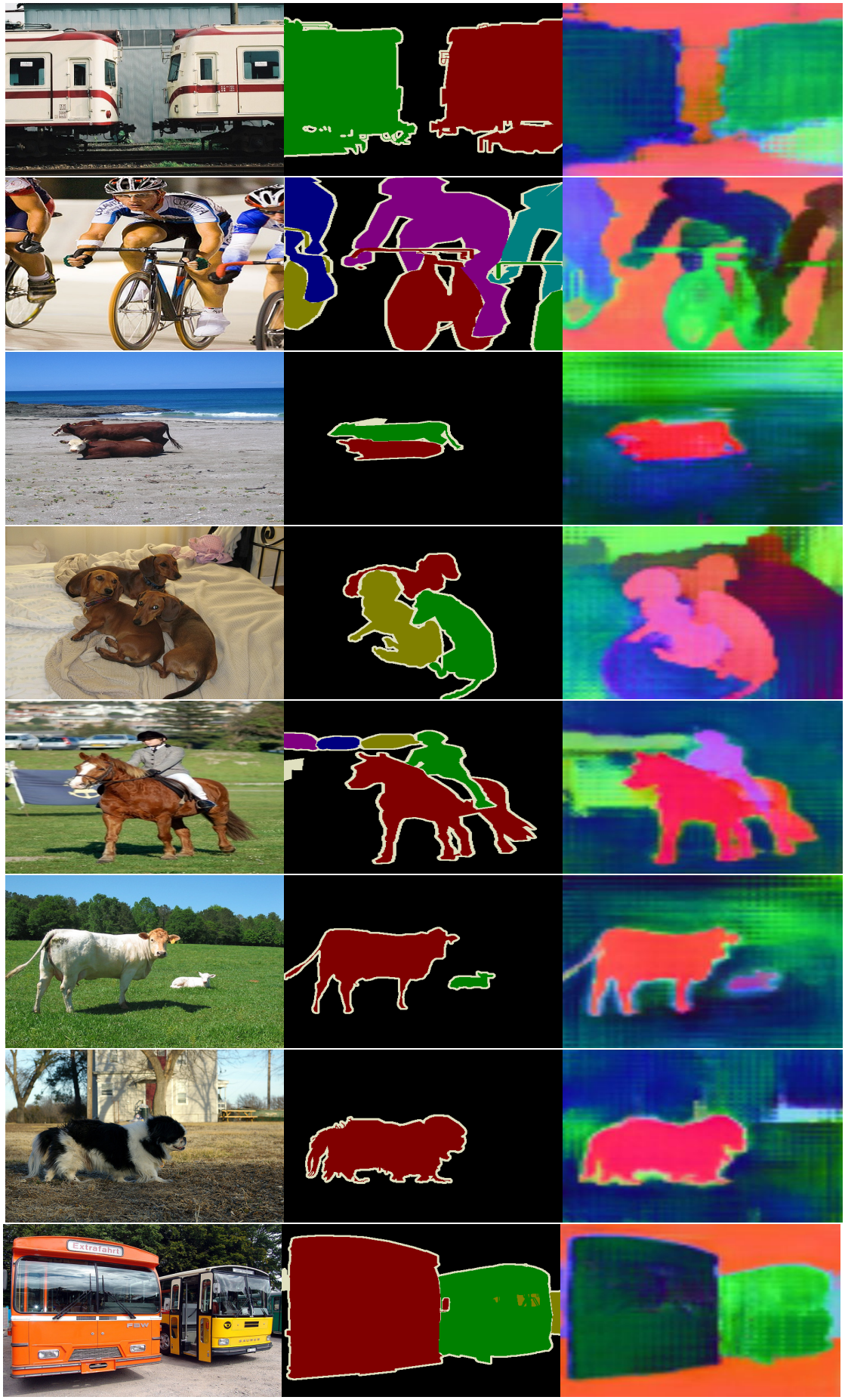


Figure 4.5: PASCAL [EVGW+10] validation images which embeddings have projected to RGB using PCA.

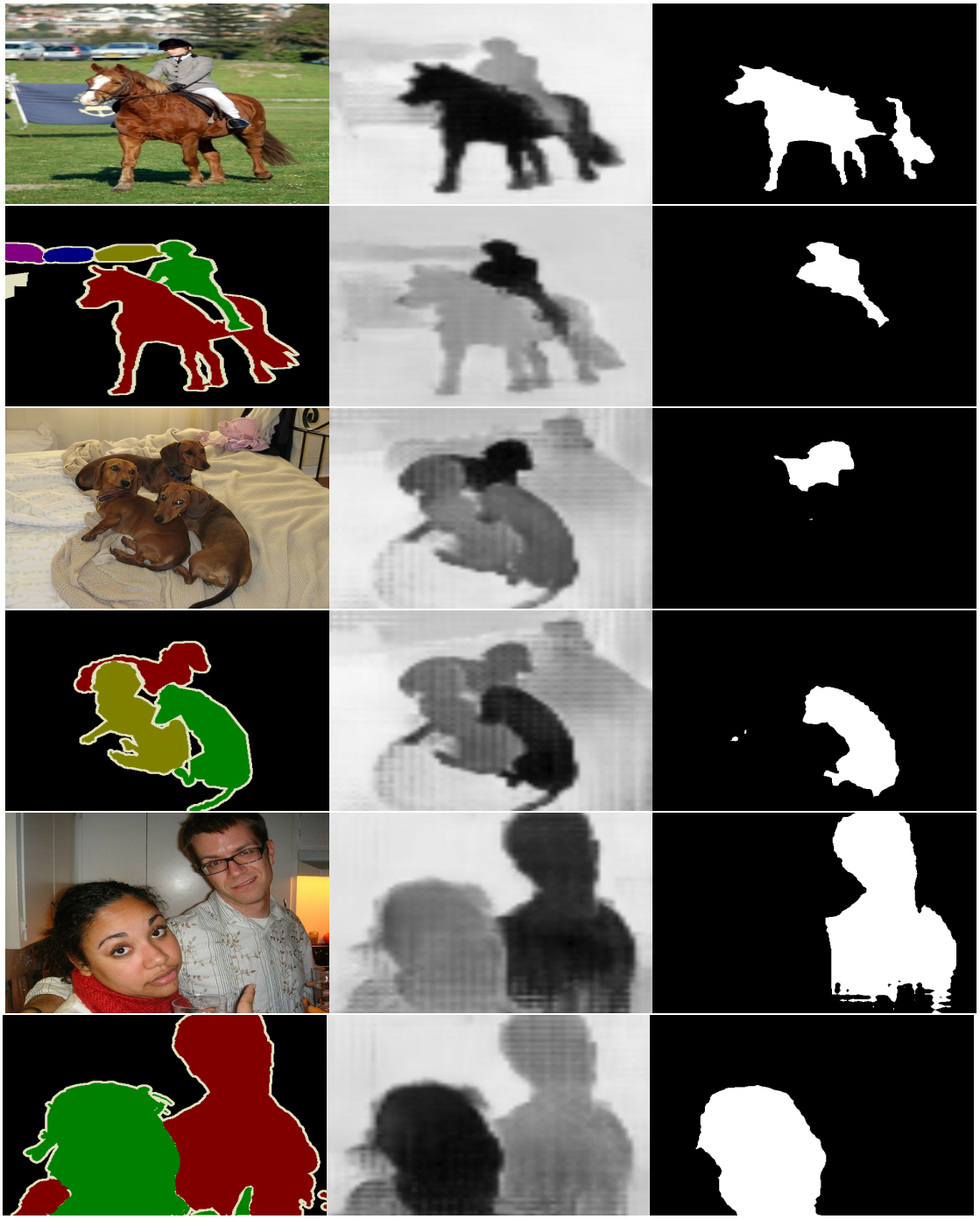


Figure 4.6: Instance segmentation procedure computing the distance map

Figure 4.8: Mask prediction results on some sequences in DAVIS [PTPC+17] 2017 validation set.

5 Conclusion and Future Work

We present a single model that is capable of segmenting instances given a point in the first frame.

The tracking performance is poor and present some flaws with occlusions and small objects.

The segmentation through embedding similarity is promising. The learned embedding seem to represent very well object and differ them between instances.

New experiments that can be done from here as future work.

Bibliography

[BRPM16] Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk, *Learning local feature descriptors with triplets and shallow convolutional neural networks.*, BMVC, vol. 1, 2016, p. 3. 9

[CMPT+17] Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool, *One-shot video object segmentation*, CVPR 2017, IEEE, 2017. 4, 5, 8, 15, 20

[CPK+18] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille, *Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs*, IEEE transactions on pattern analysis and machine intelligence **40** (2018), no. 4, 834–848. 7, 8, 9

[EVGW+10] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The pascal visual object classes (voc) challenge*, International Journal of Computer Vision **88** (2010), no. 2, 303–338. 2, 13, 15, 16, 17, 18, 19, 22

[FWR+17] Alireza Fathi, Zbigniew Wojna, Vivek Rathod, Peng Wang, Hyun Oh Song, Sergio Guadarrama, and Kevin P Murphy, *Semantic instance segmentation via deep metric learning*, arXiv preprint arXiv:1703.10277 (2017). 4

[HGDG17] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, *Mask r-cnn*, Computer Vision (ICCV), 2017 IEEE International Conference on, IEEE, 2017, pp. 2980–2988. 4

[HHS17] Yuan-Ting Hu, Jia-Bin Huang, and Alexander Schwing, *Maskrnn: Instance level video object segmentation*, Advances in Neural Information Processing Systems 30 (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), Curran Associates, Inc., 2017, pp. 325–334. 5

[HZRS16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778. 7

[KML+16] Matej Kristan, Jiri Matas, Aleš Leonardis, Tomas Vojir, Roman Pflugfelder, Gustavo Fernandez, Georg Nebehay, Fatih Porikli, and Luka Čehovin, *A novel performance evaluation methodology for single-target trackers*, IEEE Transactions on Pattern Analysis and Machine Intelligence **38** (2016), no. 11, 2137–2155. 3

[NH16] Hyeonseob Nam and Bohyung Han, *Learning multi-domain convolutional neural networks for visual tracking*, Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, IEEE, 2016, pp. 4293–4302. 3, 16

[NYD16] Alejandro Newell, Kaiyu Yang, and Jia Deng, *Stacked hourglass networks for human pose estimation*, European Conference on Computer Vision, Springer, 2016, pp. 483–499. 8

[PGC+17] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, *Automatic differentiation in pytorch*, NIPS-W, 2017. 13

[PPTM+16] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, *A benchmark dataset and evaluation methodology for video object segmentation*, Computer Vision and Pattern Recognition, 2016. 1, 5, 13, 14

[PTPC+17] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool, *The 2017 DAVIS Challenge on Video Object Segmentation*. 1, 5, 13, 14, 18, 20, 24

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin, *Facenet: A unified embedding for face recognition and clustering*, Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 815–823. 9

[ST13] Ben Sapp and Ben Taskar, *Modec: Multimodal decomposable models for human pose estimation*, Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, IEEE, 2013, pp. 3674–3681. 16

[VL17] Paul Voigtlaender and Bastian Leibe, *Online adaptation of convolutional neural networks for video object segmentation*, BMVC, 2017. 4, 5, 20

[ZSQ+17] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia, *Pyramid scene parsing network*, IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), 2017, pp. 2881–2890. 8, 16