Università degli Studi di Roma "Sapienza"
Advanced Machine Learning
Final Project Report

5 June 2020

# Human Instance Segmentation

Authors:
Michele Cernigliaro, Fabio Montello, Francesco Russo

Keywords: Image segmentation, OSVOS (One Shot Video Object Segmentation), Surveillance, Python, pytorch

Abstract

In this report we provide a description of our Human Instance Segmentation model for sequences recorded by surveillance cameras. In our work we are trying to segment specific persons along a video sequence, given their trajectory annotations. A demo is available at our github repo.

## Contents

# 1 Introduction

## 1.1 The goal: Human Instance Segmentation

In this project we are going to develop a model performing image segmentation of persons in video sequences recorded by surveillance cameras. In order to achieve this, we are going to use as a starting point some existing neural network architectures, which have state-of-the-art level of accuracy and performance in their native domain of application, and we will adapt these models to our particular context and field of interest. The fundamental base for our work is OSVOS [1], a model for segmentation of objects in video sequences using a specific approach based on one-shot fine tuning and temporal independence of different frames in a video sequence.

## 1.2 The foundations: OSVOS (One Shot Video Object Segmentation)

The model we are going to develop in our work is based on an existing architecture called **OSVOS** (One Shot Video Object Segmentation). The expression "One Shot" effectively summarizes the nature of this model. OSVOS is, indeed, composed of 3 networks (**Base Network, Parent Network, Test Network**) performing increasingly accurate separation between foreground and background until reaching the desired image segmentation. The Base and Parent Network are pre-trained, while the Test Network is fine tuned on the first frame (hence the "One Shot") of the particular video sequence on which we want to perform the segmentation, allowing the fine tuning of the model on a particular object or subject. Although this setting is the main usage configuration proposed in [1], using only one frame, in particular the first frame of the video sequence, is not a strict requirement, which means we can afford a certain degree of flexibility in the fine tuning phase, regarding the frames we use to train the Test Network. We can now explore in greater depth the structure of OSVOS.

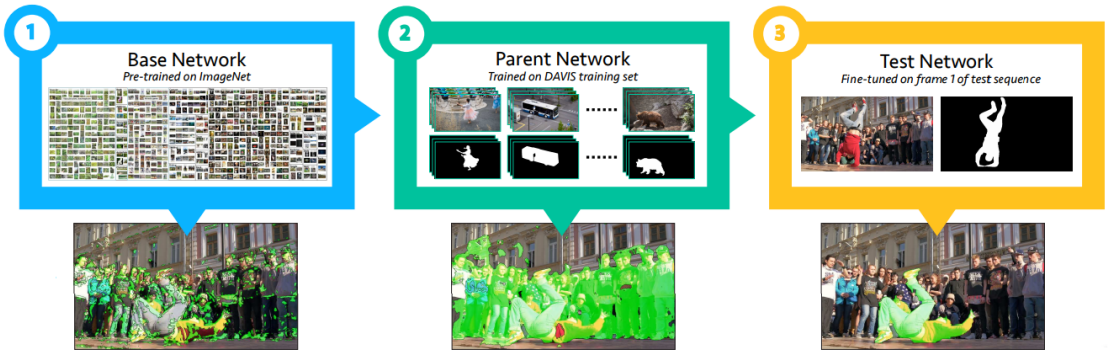


Figure 1: OSVOS models, source: [1]

1. **The Base Network**. The Base Network is the starting block of the OSVOS model, and performs a very raw separation between background and foreground. It is a Convolutive Neural Network, pre-trained on ImageNet for the task of image labeling, and the resulting model is able to capture some general features of an image, but is very far from an actual identification of background and foreground, or from the desired segmentation.

2. **The Parent Network**. The Parent Network performs a more accurate separation between background and foreground. It is pre-trained on DAVIS, starting from the Base Network, and the resulting model is able to perform a noticeably improved distinction, but it's still not able to focus on the particular object of the segmentation.

3. **The Test Network**. The Test Network performs the final and most accurate segmentation of the desired object or subject, carrying the necessary level of fine-tuning. The training of the Test Network is executed, in the standard configuration, using the first frame of the video sequence on which we want to perform the segmentation, although it is possible to use more frames. The time spent on this operation affects the accuracy of the resulting trained network.

### 1.3 The case study: OSVOS for surveillance videos

Once the Test Network is trained, the model can perform image segmentation independently on all the other frames in the sequence. The time required for this operation is fixed: in the results reported in [1], OSVOS takes 102 ms to process each 480p frame in a video sequence. This temporal independence feature is extremely useful in the context of segmentation for surveillance videos, since we might easily have to deal with occlusions, or put together sequences from different cameras, with possible interruptions when switching from a camera to another. In this framework, working with temporal consistency, as in most models for segmentation in videos, would be practically unfeasible.

On the other hand, OSVOS, in its standard configuration, is not accurate enough to perform the desired segmentation of a single individual, when multiple subjects are present in a sequence. Using more frames for fine tuning is a suitable solution.

## 2 Dataset creation: binary masks

Our main focus was on ETH and UCY video sequences, widely spread in trajectory forecasting challenges such as **Trajnet**. As we already introduced in section 1.3, given only one annotated input image from our datasets OSVOS is not able to focus on a specific person for his whole walk inside the video sequence, confusing his mask along with the other person's ones. Thus we needed more binary masks images to train it at its best. Ideally this is not an issue since in the paper it is clearly stated that it is possible to train the Test Network using more images. The downside effect is in term of training time.

## 2.1 Semi-supervised training

Initially we took one person from a video sequence, we manually created binary masks for i.e. 10 frames, and then we tried to fine tune OSVOS.

The next question we asked ourselves was how to create in **semi-supervised** way the training data? Namely, given a person (identified by a certain ID), how could we identify him within a certain number of frames and mask those frames automatically, in order to feed them to the Test network?

At this points, **trajectory annotations** have a relevant role. From ETH and UCY sequences we have trajectory annotations which give us information about which person is where every 10 frames. In practice such annotations are stored in .txt files with the following structure:

| frame | pID | x | y |

Where frame is the frame number for which there is also the position in world coordinates of a person with a certain pID (person Identifier). Our idea then was the following:

1. Given a person with a pID
2. Take the frames annotated in the trajectory annotation dataset
3. Then for each i-th "annotated" frame $f_i^{PID}$
   - 3.a Detect persons inside that frame
   - 3.b Associate with a certain criteria the closest bounding box to the (x,y) coordinates, otherwise do not consider $f_i^{PID}$.
   - 3.c Perform the masking of that person inside $f_i^{PID}$

For the point 3.a we used a **Mask RCNN**[3] in order to get bounding boxes and also the masking (point 3.c). We incorporated in our pipeline a resnet 50 model pretrained on COCO dataset and available to download with the trained weights from torchvision. With this network we were able to predict simultaneously the bounding box for each element in the image, the actual label of the content of the box and the masking of the element. These three pieces of information, when combined, are enough to discriminate between people and other elements in the image and furthermore retrieve the best box containing the person we are interested in, along with a proposed mask which will be useful later on.

Regarding the point 3.b, the criteria was the following: our algorithm has to:

- find the bbox with the smallest distance from (x,y) and
- it cannot exceed a certain threshold (should not be i.e. 40 pixels far away from (x,y))

The conversion of (x,y) from world coordinates to pixel and viceversa is performed using *homography transformations* (whenever the homography matrix is available). For ETH and UCY homography matrices are available in the github repo of [6].

Once we find the valid bounding boxes, we can proceed with the masking (phase 3.c). We decided to vary this procedure and to let the user decide among two alternatives:

1. **Background Separation algorithm** (BGS library in python)
2. **Mask R-CNN**

The latter case is straightforward, we already have the masks: when we used the Mask R-CNN in point 3.a we stored also the annotations. Hence in point 3.c we already have for each valid frame its binary mask.

In the former case, we used one of background algorithms contained in the *BGS*[4] python library in order to perform background separation (in short words background/foreground distinction, assigning black pixels to the background and white ones to foreground). Obtained in each frame all the person's white pixels representations, we just keep the white ones inside the bounding box of the person. In figure 2 you can some results of the two procedures.

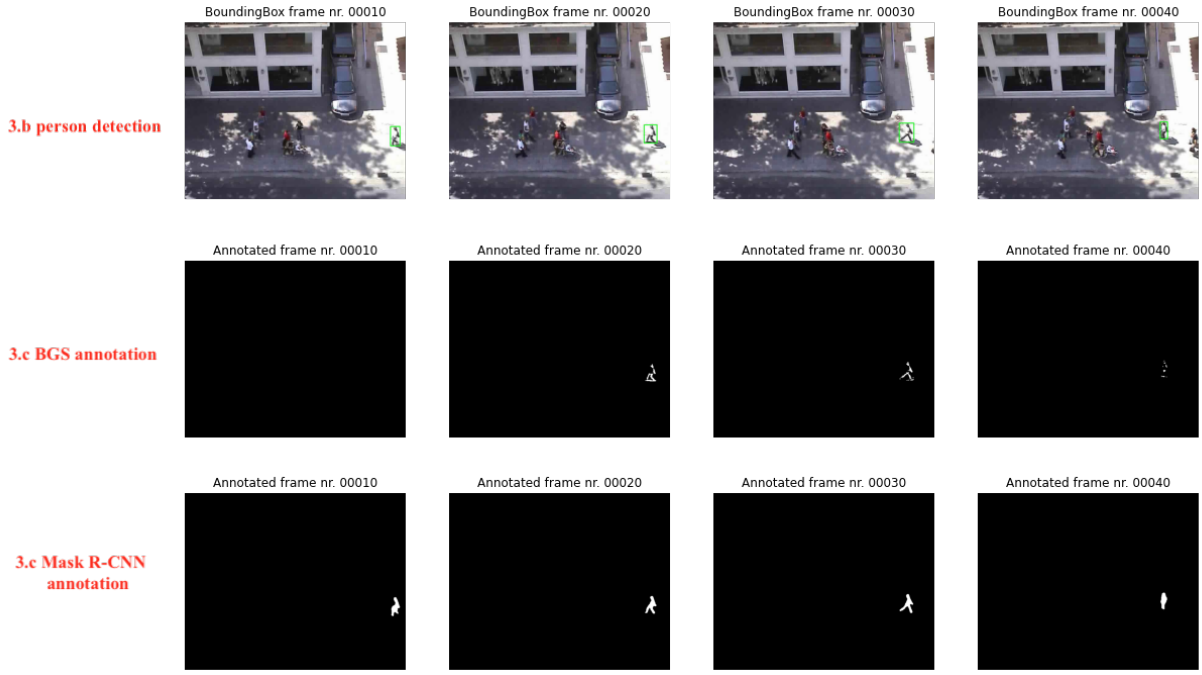


Figure 2: Example of binary masking steps: 3.b (top) finding the closest bounding box and then 3.c using either BGS (center) or Mask R-CNN (bottom) for the final ground truth binary masks

# 3 Fine tuning and testing

Obtained the annotations, we are ready to train OSVOS. OSVOS runs separately for each person. We created a pytorch dataset class and we can iterate over the segmented images. The fine tuning could be quite long (even using GPUs) and it depends on how many annotated images have been created and on how many epochs we choose to train it.

As optimization method we used the stochastic gradient descent with momentum with learning rate $lr = 1 \cdot 10^{-8}$, $momentum = 0.9$ and weight decay of 0.0002 as suggested by the paper.

After the training, the weigths of the last model related to the person $pID$ are stored as tensorboard log files. As example (using *crowds_zara02* video sequence, we can see in figure 3 the training loss curves (both using BGS and Mask R-CNN) for the person with $pID = 14$, with the model trained over 70 epochs.

t

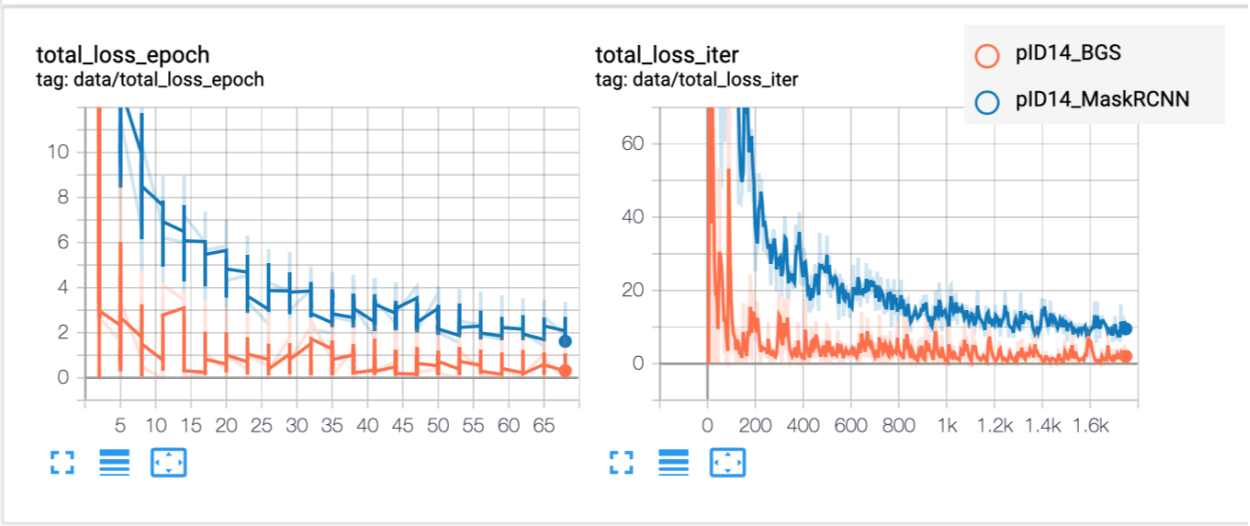


Figure 3: OSVOS Test network training loss curves for pID14 in crowds_zara02 using both preprocessing methods (BGS in orange, MaskR-CNN in blue)

# 4 End-to-end pipeline in Pytorch

One of the goals of our project was to have an end-to-end pipline, where an user is able to upload his own dataset, with the proper data annotation as seen above, and retrieve from this the segmented tracking of the person interested in. In order to do so, we tried to simplify as much as possible the set of instructions needed by the user, by wrapping them in macro functions and python files. We also added everywhere cases control, in order to give the final user the possibility to define at his/her choice some parameters and hyperparameters with the purpose of better tuning the data preprocessing and the model, to best fit the input dataset. Thanks to this approach, we give the possibility to run our version of tracing OSVOS within 25 lines of code (if we don't count the processing on the images in order to overlap the masks with the original image).

Given a video sequence (frames in .jpg format) and a trajectory dataset, our pipeline is mainly composed of two steps:

1. Given a specific person within the frame (with a certain pID), create ground truth annotations (binary masks in .png format) and JPEG images, through either BGS or the mask retrieved from the Mask R-CNN, according to the user preferences. This

procedure can be done only if there are trajectory annotations. Homography matrices are highly reccomended in order to have precise coordinates conversion. But the model is able to run even with only trajectory annotations (through proportion approximations).

2. Perform OSVOS online training with the obtained ground truth images and the original frame image. Then perform the instance segmentation for the whole length of the person's frames.

It is nice to observe that the two tasks are separated. You can perform OSVOS with any other dataset following the second data setup section. Our work is suitable for the UCY and ETH video sequences datasets, but can be used with any other dataset, as long as they respect the constraints mentioned above.

## 4.1 Final result and Demo

The Demo is available on GitHub at this link. We provided also a jupyter notebok to show how to run the code.

## 4.2 Conclusions and further developements

Video Object Segmentation is not a trivial task. We had some troubles trying to make OSVOS perform good enough to be satisfied with the result. We believe that the main issue is given by a domain shift from DAVIS to ETH/UCY video sequences, where the background/foreground ratio is drastically lower than in the former dataset. OSVOS seems to not being able to focus on a single person, on the contrary, it generalizes and it ends up recognising more persons, expecially when they occlude each other, but also when they are in different places in the image.

For each person, the fine-tuning asks a lot of annotated images and a lot of epochs to converge and give us reasonable results. We have to compromise between annotation quality and training time, and this trade-off varies from person to person, also within the same dataset.

Unfortunately, a part from the training loss, at the moment we don't have other quantitative results in our hands in order to compare the two masking algorithms that we performed in point 3.b.

Surprisingly faster R-CNN and Mask R-CNN are able to detect pretty well bounding boxes and person masks in dataset such as crowds_zara02, in a reasonable time. Perhaps algorithms which enforce temporal consistency such as MaskTrack R-CNN and/or an implementation which includes Re-Identification models along with BGS or masking techniques could led to better results in terms of annotations and performances.

As for another approach, we could re-think the parent model, shift its objective to a multi-class one and try to train it with dataset which segment different human instances, this in order to let OSVOS being able to discriminate among different person within the same frame.

# References

[1] S. Caelles, K.K. Maninis, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool - One-Shot Video Object Segmentation, Computer Vision and Pattern Recognition (CVPR), 2017.

[2] K.K. Maninis, S. Caelles, Y.Chen, J. Pont-Tuset, L. Leal-Taixé, D. Cremers, and L. Van Gool - Video Object Segmentation Without Temporal Information, Transactions of Pattern Analysis and Machine Intelligence (T-PAMI), 2018.

[3] Kaiming He and Georgia Gkioxari and Piotr Doll 'ar and Ross B. Girshick - Mask R-CNN, 2017

[4] Sobral, Andrews and Bouwmans, Thierry - BGS Library: A Library Framework for Algorithm's Evaluation in Foreground/Background Segmentation, 2014

[5] Sadeghian, Amir and Kosaraju, Vineet and Gupta, Agrim and Savarese, Silvio and Alahi, Alexandre - TrajNet: Towards a Benchmark for Human Trajectory Prediction, 2018

[6] Huynh, Manh, and Gita Alaghband. "Trajectory Prediction by Coupling Scene-LSTM with Human Movement LSTM." arXiv preprint arXiv:1908.08908 (2019). To appear in ISVC 2019