Topic G - Automatic Video Segmentation - 15 Jan. 2024

Elyas Benyamina
ENS Paris-Saclay
elyas.benyamina@ens-paris-saclay.fr

Oussama Zekri
ENS Paris-Saclay
oussama.zekri@ens-paris-saclay.fr

Abstract

The objective of video motion segmentation is to identify and isolate one or more prominent objects within a scene by examining their motion patterns. A novel method introduced for this task is SAM-PT [6]. It operates by utilizing positive queries (points on the object) and negative queries (points on the background) within a frame.

While SAM-PT currently relies on user input (via clicking on frames to provide positive/negative ground truths) or a mask, we proposed a method based on point tracker methods, that automate this selection process for single-class videos.

1. Introduction

1.1. SAM-PT method

The SAM-PT article [6], an extension of SAM [4], presents a method for tracking an object over the course of a video, enabling dynamic segmentation.

This method requires the selection of points belonging to the object and points belonging to the background over one or more frames of the video.

This paper is based on Meta's Co-Tracker [3] and SAM [4] tools. Co-Tracker is an opensource method that tracks the trajectory of a given point (selected on a frame) throughout a video. SAM is a model for segmenting an image.

1.2. Project goal

This method requires the intervention of a user who knows in advance which object he wishes to track and must select points accordingly.

The chosen topic proposes to implement methods to skip this step. The aim is to have a tool that automatically detects the object to be tracked and provides points belonging to this object and to the background to SAM-PT.

Several approaches are possible to detect such points, and there are two main types of approach.

The first is to analyze and compare the trajectories (obtained using tools such as Co-Tracker [3]) of different points in the video. The second is to adopt an image segmentation approach (using one or a few frames), to detect and select different objects. We have concentrated mainly on the first type of approach.

Note also that our codes are available on our Github Repository for this project.

2. Methods

2.1. Foundations of our method

2.1.1 Overview

As a first step, we decided to implement a naive version of SAM-PT automation. Consider a video containing M frames, counted from $m = 0$ to $m = M - 1$. This first version consists of three stages :

1. First, K points are chosen at random from the frame $m = 0$ of the video.
2. Next, we use CoTracker [3] on all points $k \in \{0, \ldots, K - 1\}$ to determine the trajectory these points will take during the video.
3. Finally, using these trajectories and a metric we will detail in section 2.1.2, we classify the points, i.e. assign them a positive or negative value.

It seems quite natural that the greater the number of randomly chosen points K, the greater the chance of detecting a *small object*.

2.1.2 A metric on trajectories: normalized speed

The *decision metric* we use for this first approach is quite simple. It is a treshold-based metric.

Let us denote z_i^k be the position of point $k \in \{0, \ldots, K - 1\}$ at frame $i \in \{0, \ldots, M - 1\}$.

The velocity of point k at frame i, denoted v_i^k (see Figure 1), can be introduced by the following relationship.

$$v_i^k := \frac{\|z_{i+1}^k - z_i^k\|}{i + 1 - i} = \frac{\|z_{i+1}^k - z_i^k\|}{1}$$

Our idea is to sum up the values of these velocities at point k, over the whole video. We obtain $v^k = \sum_{i=0}^{M-1} v_i^k$.

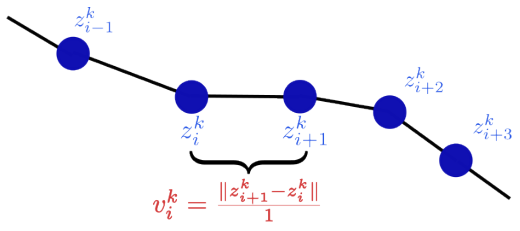

Figure 1. In blue, the successive positions of point k from frame $i-1$ to frame $i+3$. In red, the way we compute the velocity of point k at frame i.

Then normalize this quantity for all K points. This gives us a score $V^k = \dfrac{v^k}{\max\{v^k\}} \in [0,1]$ for each initial point $k \in \{0,\dots,K-1\}$.

Finally, we give ourselves a treshold $T := \alpha \dfrac{\min\{V^k\}}{\max\{V^k\}}$, with $\alpha \geq 0$. If V^k does not exceed the threshold, we will classify the k point as positive. Otherwise, we will classify it as negative.

This idea comes from the fact that a point k belonging to the object follows the camera's movements, and will therefore have a lower score V^k than a point outside the object.

A typical value for the threshold is to take $\alpha = 2$, so that $T = 2\dfrac{\min\{V^k\}}{\max\{V^k\}}$.

This approach lays the foundations for our method, but the following subsections aim to improve the point selection strategy (section 2.2) and the point classification strategy (section 2.3).

2.2. Point selection strategy improvements

A first area of improvement for our method is to better select the zones where we randomly generate our points.

To do this, we use our first approach on a large number of points K_1, with a treshold higher than usual (but not too high, so as not to mark all points as positive). See Figure 2.

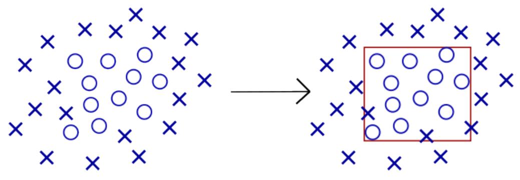

Figure 2. On the left, a first rough classification, even if it means increasing the treshold, to target movement zones. On the right, we repeat the random point selection operation, but these points are sampled in the red zone, which is an zone of interest.

Next, we create a new zone, encompassing all points marked as positive. We will then repeat our random selection of points (with a number K_2 of points), but limiting ourselves exclusively to this zone.

2.3. Classification of trajectories using neural network

Some trajectories can be difficult to classify using conventional approaches. The idea of using a neural network is that it could learn complex relationships between a point's trajectory and its class (whether it belongs to the object or the background).

We implemented a classical feed-forward neural network as well as an LTSM neural network. We focused on LSTM neural networks because the network takes temporal trajectory information as input, and LSTM networks are used to process temporal sequences.

We have provided as input various quantities relative to a point M of one of the frames we wish to classify.

1. Coordonnées: $(x_t, y_t)_t$
2. Vitesse : $(v_{x,t}, v_{x,t})_t$
3. Vitesse normalisée : $\left(\dfrac{v_{x,t}}{\|v_x\|_\infty}, \dfrac{v_{y,t}}{\|v_y\|_\infty}\right)_t$

We'll come back to the training and testing phases in more detail, as well as the architecture of the neural network used.

3. Experiments and Results

3.1. Our method with the Section 2.2 improvements

DAVIS 2016 gives a single-object VOS benchmark across 20 diverse sequences. We report results on the DAVIS 2016 validation subset in Table 1, for our method with the improvement of section 2.2, and the one of SAM-PT [6].

DAVIS 2016 Validation [5]			
	J&F-Mean	J-Mean	F-Mean
Our Method, with $(K_1, K_2) = (150, 4)$	**53.1**	**52.4**	**54.8**
SAM-PT [6]	84.3	84.9	83.7

Table 1. Quantitative results on the DAVIS 2016 validation set for semi-supervised VOS. Our method has bad scores due to the fact that some videos from DAVIS 2016 have a camera too unstable, which does not help to automatically gives the good query points to SAM-PT.

The low score of our method is explained by the presence of some videos in the dataset.

Indeed, for some videos (those with a fairly stable camera), the result looks very good, as can be seen in Figure 3. On the other hand, for videos like the one in Figure 4, where the camera moves a lot, our normalized speed metric is very confused and detects the object poorly.

For a discussion on how to improve these scores, see Section 4.2.

3.2. Attempts to use neural networks

3.2.1 Neural network structure

The neural network finally selected consists of an LSTM layer (*hidden_dim* = 20, *layer_dim*= 3) followed by a linear layer before applying a sigmoid function. We used an Adam optimizer with a learning rate of 0.001.

Note that we have also implemented a classic feedforward network with no obvious results (accuracy of 0.55 at best when trained and tested on points from the same videos).

3.2.2 Training phase

We chose images from the DAVIS 2016 dataset on which to train the chosen neural network. We selected two different video sets.

1. **Dataset 1:** "hike", "car-turn", "horsejump-low", "motocross-bumps", "stroller", "surf", "mallard-fly"
2. **Dataset 2:** "horsejump-low"

We prepare the dataset as follows. **1st step:** Random selection of n points belonging to the object to be classified and n points belonging to the background using the ground truth provided in the DAVIS dataset [5]. **2nd step:** Use cotracker to obtain the trajectories of the points. **3rd step:** Format the dataset and calculate choice quantities, such as speed if required. **4th step:** Separate the training and validation sets.

Then, at the output of the neural network, we have a sigmoid to obtain a score between 0 and 1, and we use the Binary Cross Entropy loss function.

3.2.3 Testing phase and results

We tested the networks obtained after training for the following two options, using point velocities on both axes as input.

1. **Dataset 3:** "hike", "car-turn", "horsejump-low", "motocross-bumps", "stroller", "surf", "mallard-fly" (obtained from same videos used to obtain Dataset 1)
2. **Dataset 4:** "horsejump-high"

We obtained the following accuracy results.

Table 2. Best accuracies for different training and testing data

Training option / Testing option	Option 1	Option 2
Option 3	0.92	not tested
Option 4	0.52	0.64

3.2.4 Comments on the results

The score of 0.92 obtained when training and testing the dataset on points from the same videos shows that there are no errors in our code.

The most interesting result is obtained when we try to classify the trajectories of "horsejump-high" points after training on "horsejump-high". An accuracy of 0.64 is obtained, showing that the neural network can learn to classify dots when trained on videos presenting a similar movement.

See Section A.2 for training loss and validation accuracy plot obtained during the training phase.

When the neural network is trained on several videos, the accuracy obtained drops to 0.52, which is very close to random classification. So it's better to classify trajectories independently. We could try to increase the training dataset by adding videos similar to the one we want to test.

4. Conclusion

4.1. General conclusion

The results obtained are interesting, especially in "good" cases such as that shown in Figure 3: a result is obtained whose limits are given by the original SAM-PT [6] implementation, and this **automatically**.

For more complicated videos, our method still needs to be improved to reach the scores of the manual SAM-PT [6] from Table 1.

4.2. Areas of improvement

We ran out of time to implement the second family of approaches mentioned in the introduction.

To drastically improve the results in Table 1, we would need to use a tool that allows us to determine whether we have a "good" (as in Figure 3) or "bad" (as in Figure 4) video.

One possibility, which would perhaps be too resource-intensive, would be to use Detectron2 [1] or a segmentation tool like YOLO [2] on a frame before detecting the objects in the frame and thus having points of interest. The advantage of this method is that it lends itself well to multiclass classification.

We can also adapt our approaches to multiclass classification by, for example, having a neural network with its last layer adapted to multiclass classification.

References

[1] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, and Kaiming He. Detectron. https://github.com/facebookresearch/detectron, 2018. 3

[2] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. Ultralytics YOLO, 2023. 3

[3] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together, 2023. 1

[4] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollar, and Ross

Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4015–4026, 2023. 1

[5] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alexander Sorkine-Hornung, and Luc Van Gool. The 2017 davis challenge on video object segmentation. *arXiv:1704.00675*, 2017. 2, 3, 5

[6] Frano Rajič, Lei Ke, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. Segment anything meets point tracking, 2023. 1, 2, 3

A. Appendix

A.1. Qualitative results on our automatic SAM-PT method

Here are some qualitative results of our method with the Section 2.2 improvements, on some videos from the DAVIS 2016 [5] dataset.

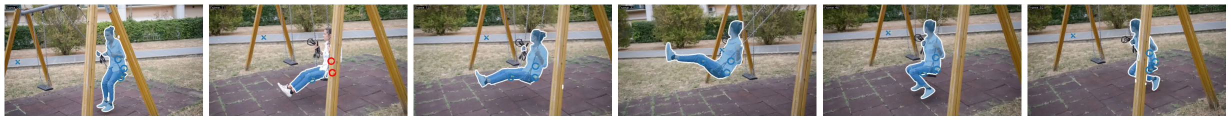

Figure 3. Method applied on the *swing* video from DAVIS 2016 [5]. Good case for our automatic SAM-PT method : Not many distracting camera movements

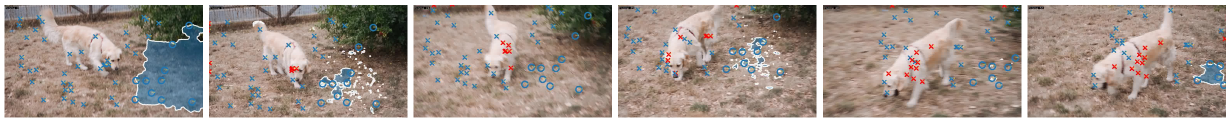

Figure 4. Method applied on the *dog* video from DAVIS 2016 [5]. Bad case for our automatic SAM-PT method : Too much camera movement, our trajectory metric is confused.

A.2. Qualitative results on our neural network implementation

Here are some qualitative results of our Section 2.3. See Section 3.2.2 for details on training.

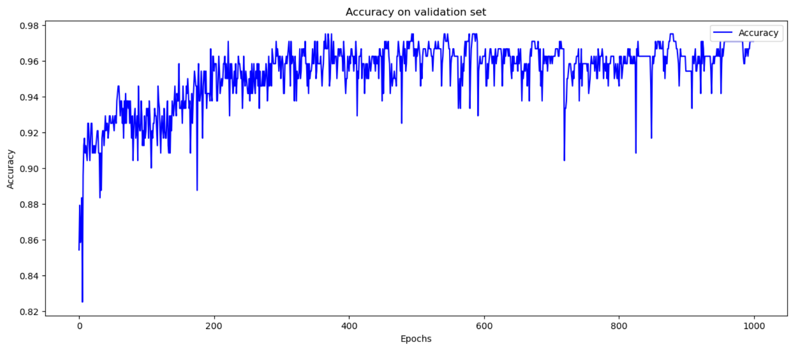

Figure 5. Accuracy on validation set for the neural network implementation

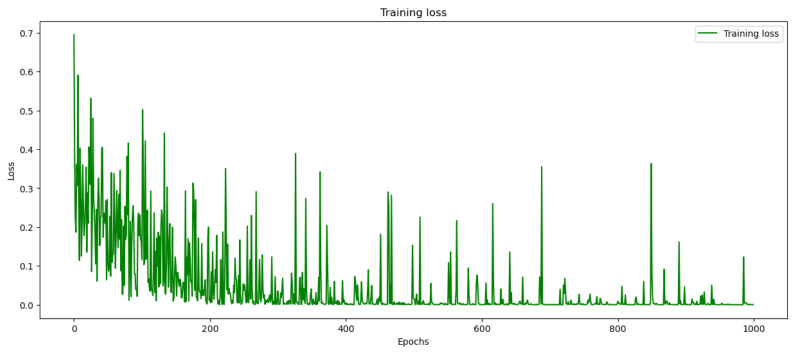

Figure 6. Training loss on validation set for the neural network implementation