Object Recognition and Computer Vision
Final Project Report
Topic G - Automatic Video Segmentation

Gauthier Multari
École Normale Supérieure Paris-Saclay
gauthier.multari@ens-paris-saclay.fr

Lilian Hunout
École Normale Supérieure Paris-Saclay
lilian.hunout@ens-paris-saclay.fr

Abstract

Object segmentation and tracking in videos is crucial for a wide range of applications, such as robotics, video editing and video compression. Despite the advancements of deep neural networks in this field, manual labeling of data is still a requirement for achieving high accuracy. Recent work on training image segmentation models suggests that point-based annotation coupled with mask editing tools offers a scalable solution for labeling large datasets. Rajič et al. proposed a new approach called "Segment Anything Meets Point Tracking (SAM-PT)" [6]. Given user-defined points on the object to segment and on the background in a frame, a point-tracking method is used to propagate the information to the next frames of the video. The goal of this project is to overcome the need of a user input in order to fully automate the process.

1. Related work

1.1. Model

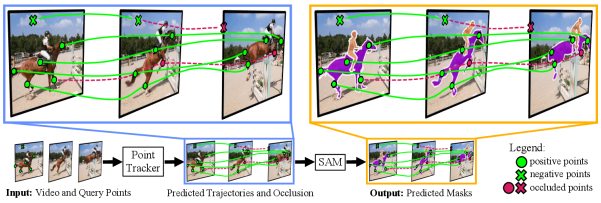

Figure 1. Segment Anything Meets Point Tracking (SAM-PT) model

SAM-PT is a point-centric method that uses sparse point propagation for interactive video segmentation, enabling easier interaction and faster annotation. SAM-PT extends SAM with long-term point trackers to work efficiently on zero-plane videos. SAM-PT considers user clicks as "query points" that point to either the target object (positive points) or non-target segments (negative points). Points are tracked throughout the video using point trackers that propagate query points to all video frames, producing trajectory predictions and occlusion scores. SAM is then asked to indicate the non-occluded points in the trajectories in order to independently generate a segmentation mask for each video frame. The propagated points can be further modified for precise segmentation and tracking.

1.2. Problem definition

Video segmentation is a problem that actually has two underlying sub-problems: the ability to segment objects on an image and the ability to track objects of interest along a video sequence using various tracking methods.

Many methods currently exist in the state of the art. These can be semi-supervised, unsupervised or interactive, as is the case with SAM-PT which requires user interaction in order to choose the object points to track. This is the task we want to automate.

We will not have to carry out training on the segmentation and tracking models because the object of our work is to give the SAM-PT model the coordinates of the points of the objects to be tracked as proposed in its architecture. We will be interested in scripts allowing us to find points of interest and annotate them in a file with the expected form.

1.3. Datasets

The datasets that will be used for this project are DAVIS 2016 and 2017 [4, 5]. The 2016 version contains 50 videos in which a single instance is annotated. The 2017 version contains 150 videos with multiple instances annotated.

1.4. Setup

For our experiments, we mainly used a GeForce RTX 4080 Laptop GPU. We did not use Google Cloud Platform credits because our project required interaction via a graphical interface which was not available (cf. Part 3.1). We therefore had to use our own machine and install all the CUDA libraries in order to benefit from hardware acceleration.

2. Experiments and results

2.1. Reproduction of paper results

The first experiment performed we planned to do was to reproduce the SAM-PT results on the DAVIS 2016 and 2017 datasets using the provided pre-trained checkpoints (cf. Table 1). We were interested in the J&F scores on the DAVIS 2016 and 2017 datasets. We used the same weights and the same parameters and got results close to those announced. The difference could be due to a change in the code announced on the GitHub repository.

Method	Propagation	DAVIS 2016	DAVIS 2017
SAM-PT (paper)	Points Prompting	84.3	79.4
SAM-PT (ours)	Points Prompting	83.9	79.2

Table 1. Comparative analysis of semi-supervised Video Object Segmentation methods

The first idea proposed to solve the problem defined in Part 1.2 was to initialize the point queries randomly in the first frame of the video, run the point tracking method a first time on the whole video to recover the trajectories and use those to classify the foreground and background points. After further consideration, we concluded that this method on its own could be associated to do a heavy down-sampling of the video, requiring a high amount a point to get accurate masks, thus being very computationally heavy. We decided to focus on approaches that take the whole images into account. We will therefore subsequently present 3 methods used to achieve self-labeling of objects.

2.2. Detection with background subtraction

The background subtraction-based detection method using the OpenCV implementation of "mixture of Gaussians" [1] relies on several steps to identify moving objects in an image sequence, assuming a static camera and relatively slow object movements.

First, the background model is initialized to capture the static scene without movement from the initial image (cf. Figure 2). Subsequently, this model is updated with each new frame of the sequence, allowing the system to adapt to slow changes in the scene over time. Background subtraction is done by obtaining a foreground mask from the updated background model. This foreground mask isolates pixels that differ significantly from the background model, thereby representing potential areas of moving objects (cf. Figure 3). To obtain a proper motion mask, morphological operations are performed to remove noise and imperfections from the foreground mask. Afterwards, the initial motion detections are extracted from the proper motion mask. To improve accuracy, non-maximum suppression is performed to remove redundant detections and keep only the most important motion areas (cf. Figure 4).

Figure 2. Initial image from static camera scene

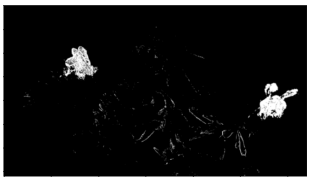

Figure 3. MOG foreground mask after 5 frames

Finally, the last step consists of generating n positive points (green) and m negative points (blue) by generating the points at random coordinates in the foreground and the background of the image, for each object detected (cf. Figure 5), and saving this in the expected format. This will also be done in the other methods (cf. Parts 2.3 & 2.4).

Figure 4. Final motion mask

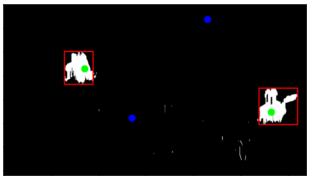

Figure 5. Tracking points query for SAM-PT

We assumed a static camera at the start of the video because this method is not robust in the opposite case, which represents an important limitation (cf. Figure 7).

Figure 6. Initial image from non-static camera scene

Figure 7. MOG foreground mask after 5 frames

2.3. Detection with Dense Optical Flow

In order to have a method robust to camera movement, we opted for another method, the computation of dense optical flow. It is one of the main technique used in computer vision for motion estimation.

The first working experiment was done using the Gunnar Farneback's algorithm [3]. The algorithm formulates the optical flow as a Taylor series expansion of the image intensity function. By employing polynomial approximations, Farneback's method characterizes the spatial variations in intensity across the images, allowing for the derivation of dense motion vectors. This approach involves processing the image at different scales, starting from a coarse level and progressively refining the flow estimation at finer

scales. This strategy enables the algorithm to capture both global and local motion information, making it robust in scenarios with complex motion pattern. For our testings, the OpenCV implementation of this algorithm was used.

Once the optical flow vector field is computed we can use the magnitude and angles of the vectors in order to detect the moving objects in the image (cf. Figure 9). To do so we compute a variable motion threshold based on the prior knowledge of the camera position.

Figure 8. Initial image

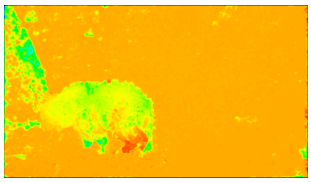

Figure 9. Log of dense optical flow magnitude

We then apply this threshold to the field (cf. Figure 10), binarize the results to get a mask and post process it using morphological operations to remove small moving parts. Finally we detect the objects and perform non-maximal suppression in order to keep only the most relevant objects (cf. Figure 11).

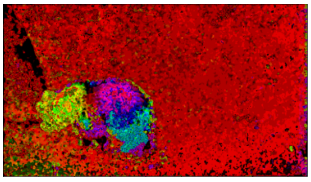

Figure 10. Dense optical flow

Figure 11. Non-max suppressed bounding boxes

This method yields much better results than the one of Part 2.2, especially with a moving camera. But it is far from being sufficient, it struggles in many instances whether regarding the quality of the optical flow or the detection of the moving objects. The threshold method is too simple to efficiently discriminate the vectors.

2.4. Improved detection with optical flow

Our final experiment's goal was two-fold. The first idea was to improve the quality of the results of the optical flow. The second to better discriminate the moving parts from the static part.

For the first goal, we decided to use the method presented in the paper Recurrent All-Pairs Field Transforms (RAFT) [7]. The core of RAFT lies in its recurrent update mechanism. Unlike traditional optical flow methods that rely on a single forward pass, RAFT iteratively refines the initial flow field using a gated recurrent unit (cf. Figure 12). This recurrent update process effectively captures complex motion patterns and temporal inconsistencies, leading to more accurate flow estimates.

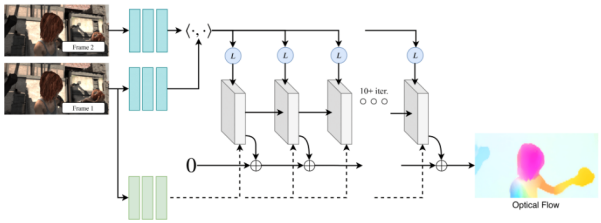

Figure 12. Recurrent All-Pairs Field Transforms for Optical Flow

RAFT's all-pairs correlations provide a comprehensive representation of the spatial relationships between pixels in both the reference and target frames. To capture these relationships, RAFT constructs multi-scale 4D correlation volumes by taking pairwise element-wise multiplication of the feature maps extracted from the input images. These correlation volumes encode rich contextual information that is crucial for accurate flow estimation. An example for the output of this method can be seen on Figure 14. We notice a clear improvement in comparison to Figure 10 generated with the previous method.

Figure 13. Initial image

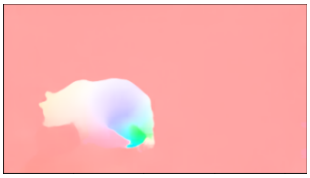

Figure 14. Estimated flow

Regarding the discrimination of the moving parts of the video, we looked for a method that could model the background flow in order to get a more accurate threshold. The chosen method is the one proposed by Zhang et al.. The idea is to consider the background optical flow field as a quadratic function of the point coordinates. Let u the horizontal and v the vertical flow vectors, F_b the background optical Flow, F_f the foreground optical Flow and $F = F_b + F_f$ the optical flow, H the matrix modelling the background flow and N the number of points to sample. We define $P = \begin{bmatrix} x & y \end{bmatrix}_{N\times 2}$ the matrix holding the locations, $Y = \begin{bmatrix} x+u & y+v \end{bmatrix}_{N\times 2}$ the matrix holding the projected pixel locations from frame t to $t+1$ and $X = \begin{bmatrix} x^2 & xy & y^2 & x & y & 1 \end{bmatrix}_{N\times 6}$ the quadratic function of the locations. We want to solve $min||Y - XH||^2$ with $H = (X^T X)^{-1} X^T Y$.

To do so, we first sub-sample the image using Quasi-random sampling to get a sparse representation of the image. This method divides the image into grid squares out of

which a single pixel is sampled from. In order to select the best model we then use the constrained RANSAC algorithm presented in [8]. We re-implemented it from scratch given that the authors did not provide any code. The pseudo-code is written in Algorithm 1.

Algorithm 1 Constrained RANSAC algorithm

procedure GET_BACKGROUND_MODEL_RANSAC $(F, P, X, N, n\_iters, min\_inliers, threshold)$
  for i in n_iter **do**
    $Y_s = P_s + F_s$
    $X_s =$ get_poly_from_samples(P_s)
    $H_s = \left(X_s^T X_s\right)^{-1} X_s^T Y_s$
    $F_b = XH_s - P$
    inliers $= MSE(F - F_b) <$ threshold
    if len(inliers) $>$ min_inliers **then**
      $Y_i = P_i + F_i$
      $H_i = \left(X_i^T X_i\right)^{-1} X_i^T Y_i$
      $F_i = X_i H_i - P_i$
      $MSE_i = MSE\left(F - F_i\right)$
      if $MSE_i <$ best_mse **then**
        best_mse $= MSE_i$
        $H = H_1$
      end if
    end if
  end for
  return H
end procedure

The authors stated that the algorithm gave the best results when using 50 iterations, so that is the value we used for our testing. An example of result can be found on Figure 16.

Figure 15. Mask generated with RANSAC

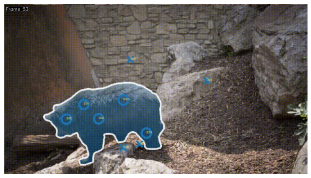

Figure 16. Segmentation result using the mask

This method lead to a major improvement in the quality of the generated masks and this translated to an improvement in the quality of the segmentation results.

It still has several limitations. The first one is that it fails to detect the correct object when the camera is moving on the first frames and the object is far from the camera. In these situations, if there is an object closer to the camera, it will be detected as the moving object. Instances where the camera moves more than in the DAVIS dataset would also lead to poor performances for this method.

3. Discussion

3.1. Problems encountered

During the implementation of our development environment, several obstacles emerged, generating significant challenges. Firstly, the lack of a local GPU for a team member meant that a virtual machine had to be set up on Google Cloud Platform to benefit from the necessary performance. However, once inside the virtual machine, issues with some dynamic libraries complicated the process, hindering the proper functioning of our environment. In addition, the absence of a graphical user interface in the virtual machine limited our ability to study certain aspects interactively, making certain tasks more complex. Sharing tasks during pair programming also posed challenges, with the inherent difficulty in distributing work efficiently. Installing local dependencies, including Detectron2 and CUDA versioning, was also a source of complexity. Additionally, resolving FFMPEG-related errors was another major hurdle in our development journey, highlighting the need for robust solutions to ensure a smooth workflow. These issues required a methodical approach and close collaboration to overcome them and ensure the success of our project. Finally, another major issue we faced was the calculation of the J&F metric. Several days have been spent trying to debug the authors code, given that the results given by the script were always 1, for every image. We decided to skip this part at first in order to focus on the implementations of the methods. In the final days of the project, we managed to compute the metrics by using a virtual machine with a Linux installation. Our guess is that the "davis2017" library used to compute the metrics requires some Linux dependant components. Unfortunately, given the time restrictions of this we did not manage to fully automate the process in order get quantitative results on our methods.

3.2. Limitations and extensions

The first extension to our work would be to completely automate the processus by adding our methods to the initial SAM-PT repository and therefore get quantitative results to compare them. The randomness of the point initialization is an important matter that we did not address during this project. We could imagine smarter point initialization than a random sampling conditioned by the mask, like a method that would add point to the extremities of the mask. We could also preprocess the first frames of the videos using semantic segmentation model to filter the detection mask, but this would not be trivial to implement. Another option would be to use a depth prediction model (e.g DinoV2 [2]) to generate the mask. Finally, we think that the best approach would be to build a model that take into account both the dense optical flow and the point tracking. This would allow for more information and probably better results.

References

[1] Thierry Bouwmans, Fida El Baf, and Bertrand Vachon. Background modeling using mixture of gaussians for foreground detection - a survey. *Recent Patents on Computer Science*, 1: 219–237, 2008. 2

[2] Alexander Kolesnikov Elad Hoffer, Danijar Hafner. Dinov2: Learning robust visual features without supervision, 2023. 4

[3] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian conference on Image analysis (SCIA'03)*, pages 363–370, Berlin, Heidelberg, 2003. Springer-Verlag. 2

[4] 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. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 724–732, 2016. 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. 1

[6] Frano Rajič, Lei Ke, Yu-Wing Tai, Chi-Keung Tang, Martin Danelljan, and Fisher Yu. Segment anything meets point tracking. *arXiv:2307.01197*, 2023. 1

[7] Shuyang Sun, Jiarong Yang, Xinbo Wang, and Kai Yu. Raft: Recurrent all-pairs field transforms for optical flow. In *Proceedings of the 28th International Conference on Computer Vision (ICCV)*, pages 7171–7180, 2020. 3

[8] Li Zhang, Zhe Liu, Yiping Zhang, and Zhiyong Zhang. An efficient optical flow based motion detection method for non-stationary scenes. *Pattern Recognition Letters*, 124:60–69, 2020. 3, 4