Non-rigid Multi-object Tracking

Azzolin Steve, Destro Matteo

University of Trento
Department of Information Engineering and Computer Science (DISI)

1 Introduction

Target visual tracking is one of the most important topics in computer vision, with many applications ranging from surveillance to augmented reality. Over the years, many algorithms have been proposed regarding this topic, the majority of which address the problem with rectangular bounding boxes, thus disregarding the precise shape of the target being tracked. Video inpainting[1] is a foremost example of the benefits of non-rigid tracking, i.e. isolating from the scene only the exact shape of a target object to be either enhanced or removed.

In this project we faced the problem of non-rigid tracking, first by a review of common methods for tracking, then with a research of current algorithms for non-rigid tracking, and finally by proposing our solutions. Another important aspect that we addressed is the applicability of our solutions to multi-object tracking applications, something not widely discussed in the literature regarding non-rigid tracking algorithms.

2 Related work

In the following sections we review some preparatory methods that we used in our proposals later discussed in Section 3.

2.1 Novelty detection

The goal of Novelty detection is to find anomalies, or outliers. Anomalies are samples that greatly differ from the distribution of normal samples and, in many cases, they negatively affect generalization performances of Machine Learning models. Anomaly detection can be performed with several techniques, for example *Dimensionality reduction*. The core idea is to reduce the dimensionality of the dataset and then restore the original dimension. The goal is to preserve only relevant features in the forward step while forgetting the irrelevant details of the data. After the back step, the data will not be exactly reconstructed, but an error will be introduced between the original full-dimensional data and the full-dimensional reconstructed dataset. This reconstruction error is assigned to each sample as an anomaly score. The greater the reconstruction error is, the greater the probability that the sample is an anomaly. This procedure works thanks to the fact that the dimensionality reduction algorithm will have the largest reconstruction error on those samples that are hardest to model, or in other words, those that occur the least often and are the most anomalous [1]. Suitable dimensionality reduction algorithms are, among the others, PCA and Autoencoders.

As will be discussed in Section 3.4, we used Novelty detection to detect potential outlier pixels.

2.2 Feature detection and matching

Feature detection algorithms are used to extract from an image a set of key-points, each represented by a feature vector, that can be used to match points in different images. There are various different algorithms available to extract these features, but during the implementation of our model we focused on the most widely used: SIFT [2] and ORB [3].

Feature matching is the task of establishing correspondences between two images of the same object. A common approach to image matching consists of detecting a set of interest points from two or more images, computing a descriptor vector for each point, and then linking them together. A simple and common algorithm is the Brute-force matcher.

An alternative to feature matching is given by Optical flow, the pattern of apparent motion of objects in a scene. Whereas it is not directly tied to feature matching, it can still be used to compute the new positions of a set of predetermined key-points in the next frame, obtaining a result similar to feature matching. Since we were interested only on computing the optical flow for a limited set of points, we decided to adopt an algorithm for sparse optical flow estimation, in particular the Lucas–Kanade method [4].

2.3 Rigid tracking

Rigid trackers are a family of algorithms used for tracking arbitrary moving objects. They are called ”rigid” because they work on rectangular bounding boxes and they do not take into account any possible deformation of the tracked object.

Our main goal was to extend these methods, in order to implement a non-rigid tracker able to work on deformable polygonal masks. The general idea was to exploit these algorithms to limit the execution of our model to the bounding boxes returned by these trackers, i.e. to implement multi-object tracking and to reduce the computational cost of our non-rigid tracker.

Since our project was based on a previous work from another group, that already conducted some research regarding the most suitable rigid tracker, we decided to use their findings and adopt the same algorithm they chose: the CSRT tracker [5].

[1] https://nbei.github.io/video-inpainting.html

3 Solutions

In this section we present our proposals, all implemented with Python and the OpenCV library. First, we tried the basic approach of background subtraction, for its simplicity. Then we looked at the literature and we implemented the highly non-rigid object tracking proposed by Lin-Pun [6], but the results were not satisfactory. After that, we tried three new approaches: Optical Flow and Convex Hull (OFCH), Pixel Classification (PC) and GrabCut. In the end, to improve the performances of these algorithms, we implemented some tricks, that will be discussed in the respective sections. A summary of the performances of these three proposals can be found in Section 4.

3.1 Background subtraction

Background subtraction is a common technique for generating a motion mask in a static camera setting. It computes the difference between the current frame and a background model that contains the static part of the scene. To keep the background model accurate, it can be updated to reflect any possible change in the scene's background, e.g. due to weather or illumination changes. There are many algorithms that implement background subtraction, each with different techniques to update the background model, e.g. based on Gaussian mixture models or KNN.

Since the class of problems we initially addressed (segmentation of players in basketball matches) provided a static camera, this model gave a good baseline. However, there are still many issues with this approach, the first of which is given by the constraint of having a static camera, which is not always possible in many real use-cases. Moreover if an object with a large flat surface remains in the scene for a certain amount of time, and an initial model for the background is not available, the algorithm may pick it up as background even if the object is moving, since it is unable to detect intensity changes in the flat area. Another issue is caused by non-static objects in the background, e.g. leaves or digital billboard ads, which get mapped as foreground objects.

3.2 Lin-Pun highly non-rigid object tracking

Lin-Pun non-rigid tracker [6] is based on the concept of superpixels, i.e. the over-segmentation of an image into regions by considering similarity measures. The motivation is to obtain regions that represent meaningful descriptors with far less data than the original image (an example of superpixelation is presented in Figure (1g)). The algorithm trains a discriminative online model on HSV pyramidal histograms extracted from superpixels belonging to the object being tracked, provided by a human annotator at the first frame. Then, at every frame, an object proposal [7] algorithm is used to propose target candidates in the form of contiguous superpixels. Finally, candidates are ranked based on both the probability estimated by the discriminative online model, and on motion weights that represent the displacement w.r.t to the target object predicted at the previous frame.

Despite the results of this algorithm being promising, we were not able to reproduce them in our implementation. Moreover, it does not account for multi-tracking, something that we wanted to include in our project.

3.3 Optical Flow and Convex Hull (OFCH)

As already mentioned in Section 2.2, optical flow is the pattern of apparent motion of objects in a scene. One of the first approaches we tested was based on exploiting this information to track a predefined set of key-points belonging to the foreground.

Initially, the user selects an approximated polygonal mask of the object he wants to track. Then, our model extracts an initial set of points of interest from the selected mask using the ORB feature detector (see Section 2.2). The positions of these points are then tracked in the next frame using the Lucas–Kanade method for sparse optical flow [4]. By doing this, we obtain a set of points that most likely still belong to the foreground object. Then, we compute the convex hull of these points to generate an approximated foreground mask. Since the resulting mask is just a convex polygon that do not follow the object's edges very well, we try to refine it by computing a superpixel segmentation and merging it with the convex hull. We assign a superpixel to the final mask if the fraction of its area covered by the convex hull is over a predefined threshold. The results however were still very approximated and subject to noise due to the optical flow not tracking correctly some of the points between the frames.

3.4 Pixel Classification (PC)

The next approach that we tested is based on the paradigm of supervised learning, i.e training a probabilistic classifier in order to distinguish between pixels belonging to the target object and pixels belonging to the background. This approach is suitable in all the contexts in which the color distribution of the target object is clearly distinguishable from the background. The first sketch of the algorithm is presented in Algorithm 1.

Algorithm 1: Pixel Classification (PC) algorithm

while isLast $\neq True$ **do**
  isLast , frame $\leftarrow$ input.read()
  X $\leftarrow$ extractFeatures($p_{i,j}$)$\forall p_{i,j} \in frame$
  probs $\leftarrow$ model.predictProbability(X)
  mask $\leftarrow$ probs.reshape(frame.shape)
  mask $\leftarrow \begin{cases} 255 & \text{if probs}_{i,j} \geq 0.5 \\ 0 & \text{otherways} \end{cases}$
  output(mask) `// save and/or show`
end

The model is trained on the first frame, in which the user is asked for a manual selection of foreground pixels, as depicted in Figure (1b). In our experiments, the model consisted either of Random Forest [8] or K-Nearest Neighbors. The function

extractFeatures is of key importance, since extracting good features would allow the model to learn meaningful patterns. We experimented with different combinations (more on this in Section 4), and at the end the one achieving the best results was to extract the HSV pixels' value from a 6/8-neighborhood around each pixel. However, many more experiment may be conducted, with more advanced feature extraction mechanisms.

To enable the tracking of multiple targets, we included a standard rigid tracker which has the aim of restricting the area in which the discriminative model will be applied. In this way, the tracker will follow the targets independently, while, inside each rectangular bounding box, our model will discriminate between target pixels and background pixels. Actually, in order to fit as better as possible the characteristics of the different tracked objects, an independent model is trained for every target. As already mentioned, the positive samples (target pixels) are provided by the user in the first frame via a polygonal selection tool, while negative samples (background samples) are extracted in the area between the polygonal user selection, and the minimum rectangle that includes the shape of the target. To reduce the impact of the rigid tracker cutting some parts of the target being tracked, we decided to enlarge the bounding box by a constant factor before classifying the pixels. An example of per-frame prediction is depicted in Figure (1c).

At this point, we further improved the algorithm with the following tricks:

- *Superpixels*: As can be noted in Figure (1c), the shape of the soldier is correctly guessed by the algorithm, but it lacks uniformity, opening to the possibility of having multiple holes inside the shape. To reduce this phenomenon, we adopted a similar approach of Section 3.2: Superpixelation. Algorithm (1) remains quite unchanged, except for the fact that the final mask is obtained by averaging the pixels' probability on every superpixel, applying the threshold on superpixels instead of single pixels. In this way we smooth out some noise in the predictions, getting more uniform shapes. The result is presented in Figure (1d).

- *Novelty detection*: To reduce the impact of background objects and colors not included in the training of the discriminative model, we applied PCA for novelty detection, which allows estimating the degree by which a feature vector of a pixel can be considered an outlier. Since they are outliers, they may fool our model, resulting in incorrect probability estimates. So, we reduced the probability estimate for outlier pixels, proportionally to the degree they are considered outliers. Note however that this approach showed conflicting results, since the target object itself might expose unseen colors.

- *Prior probability*: So far, the algorithm is basically applied independently frame-by-frame, disregarding completely the information about motion. To incorporate also this information, we decided to estimate a sort of prior probability, to be used in combination with the prediction of the discriminative model introduced before. To every superpixel a positive prior is assigned depending on whether it contains a point that matched with the previous frame (to encourage the superpixel to be assigned to the foreground), or otherways a negative prior (so that the discriminative model must be quite confident of its predictions about this superpixel). The matching of the points is implemented as described in Section 2.2, and an example of matching between consecutive frames is depicted in Figure (1e). Figure (1f) illustrates the prior map for the second frame of the sequence.

- *Multiple models*: The last trick is about asking the user to label multiple frames, over the duration of the video, instead of only the first one. With this possibility, we could train multiple models over time, combining the predictions of the two closest models in time. Moreover, this enabled also the re-initialization of the rigid tracker, which further reduces the possibility of cutting out some part of the target. In our experiments, we set 3 re-initializations for every target.

3.5 GrabCut

GrabCut is an image segmentation algorithm based on graph cuts [9]. It accepts as input a mask specifying for every pixel if it belongs surely or not either to the background or to the foreground.

Initially, the user selects a polygonal mask containing the object he wants to track. Then, a rectangular bounding box of the polygon is computed. Since GrabCut accepts a mask specifying which pixels belong to the foreground or background, all the points inside the box are marked as "probably foreground", while the points outside as "sure background". To improve the accuracy of the GrabCut segmentation, a set of key-points are computed both inside and outside the selected polygon and are marked respectively as "sure foreground" and "sure background".

By doing this, the algorithm is able to obtain an accurate mask of the foreground object without needing an accurate polygonal selection of the object's contours. This initialization however can be used only on the first frame, where the user's selection is available. In the following frames we need a way to determine the bounding box and the key-points without a polygonal mask. To do so, we use the bounding box returned by a rigid tracking algorithm, in particular CSRT, and we try to track the key-points selected in the first frame into the following frames, by using either a sparse optical flow [4] or a feature matching algorithm (that in our experiments performed better), as seen in Section 2.2. The resulting mask is then given to the GrabCut algorithm, that computes the foreground segmentation and returns a binary mask of the result. To reduce noise and merge contiguous areas, both morphological opening and closing are applied sequentially.

Since the tracking of the key-points is not perfect and it is susceptible to occlusions, some points are lost as the video goes on and the segmentation becomes less and less accurate, and in some cases the object is lost entirely. To overcome this

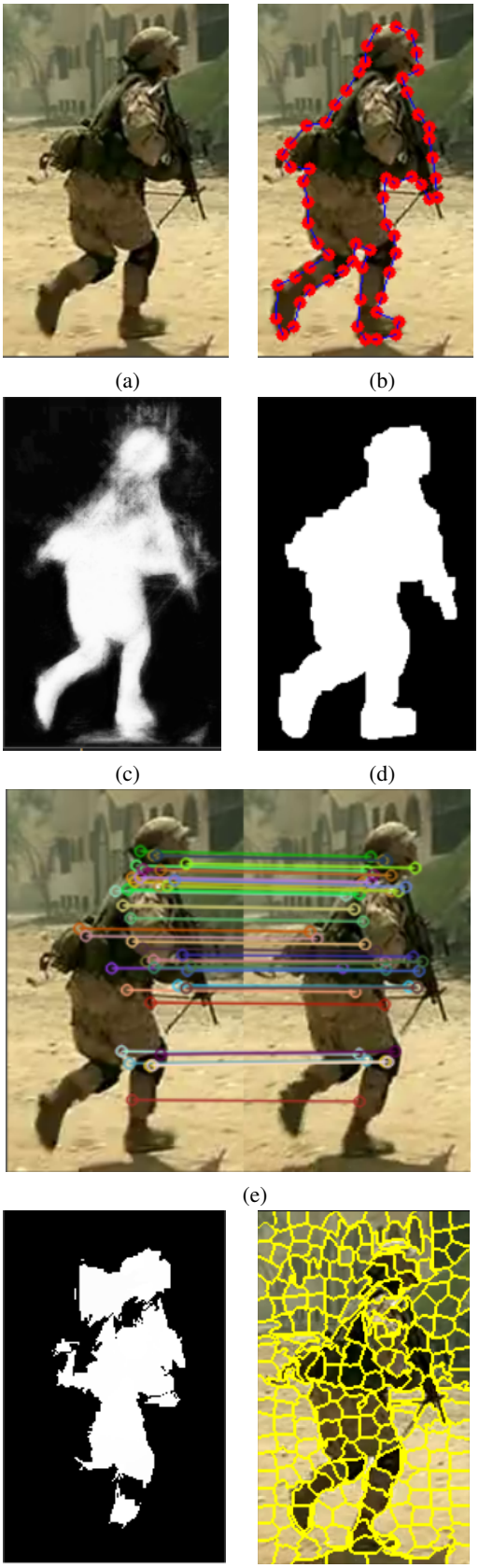

Figure 1: (a) Cutted first frame of soldier sequence (b) Target selection tool (c) Probability map pixel-by-pixel (d) Probability map averaged over superpixels (e) Example of SIFT feature matching between consecutive frames (f) Prior probability estimated from feature matching, clipped in 0-255 for plotting purposes (g) SLIC superpixels

issue, we implemented an on-line re-initialization of the model. When the total size of the resulting mask is below a predefined fraction of the mask's size selected in the first frame, the user is asked for a new polygonal selection on the current frame and the model is re-initialized as explained above. This allowed us to obtain good results with minimal human involvement.

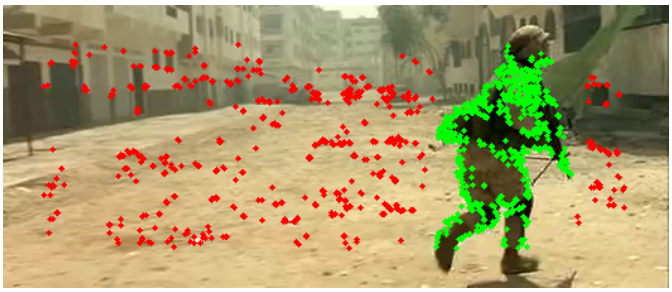

Figure 2: Example of the features extracted by the ORB algorithm. The green points represents foreground features while the red points are background features

Figure 3: Example of the GrabCut segmentation results, based on the extracted features shown in Figure 2

4 Results

In this section we are going to describe the performance of our proposals w.r.t. a labelled dataset, the SegTrack2[2] dataset. This dataset contains several short sequences, along with a ground truth consisting of a binary mask depicting the target object for each frame. Table (1) shows the resulting performances, computed through the common "intersection over union" segmentation metric.

The availability of ground truth allowed us to set-up a hyper-parameter tuning for our solutions, in particular for the Pixel Classification approach since it is based on several parameters. We tested the same algorithm, initialized with different parameters, over a subset of sequences, in order to estimate empirically the best set of parameters. For an effective hyper-parameter exploration, the runtime for every sequence should be pretty low, to allow a fairly complete exploration of the parameter space. To reduce it (with the vanilla Python implementation it was pretty high), we exploited the JIT Numba[3] compiler, that brought vast improvements. Figure (4) illustrates the

[2] Available at https://web.engr.oregonstate.edu/ lif/SegTrack2/dataset.html
[3] https://numba.pydata.org/

Sequence	OFCH	PC	GrabCut
Parachute	0.60	0.77	0.79
Monkey	0.61	0.74	0.78
Soldier	0.58	0.74	0.75
Worm	0.35	0.72	0.73
Frog	0.32	0.76	0.61
Girl	0.52	0.76	0.68

Table 1: Intersection over union score of our solutions for some sequences of the SegTrack2 dataset

variability of the score across all hyper-parameters tested on the six sequences.

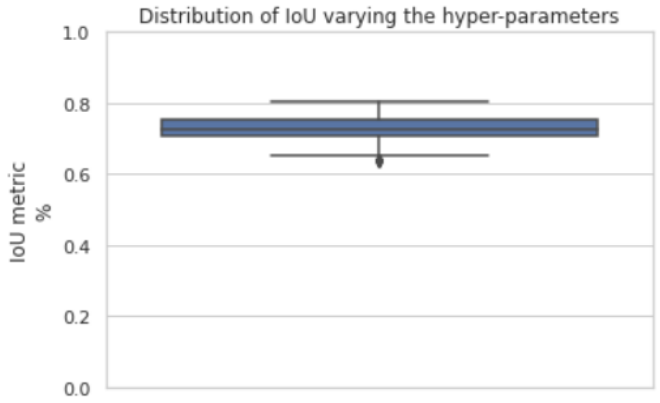

Figure 4: Boxplot of the scores obtained during the benchmark on six sequences, for Pixel Classification

5 Conclusions

Overall, Pixel Classification and GrabCut are achieving the best performance among our solutions. The results were quite promising, given the fact that many more extensions can be applied in future works. For example, instead of enlarging by a fixed constant factor the rectangular bounding box provided by rigid trackers, one might develop a dynamic enlargement of it, based on the relative position of the predicted mask. This would allow to further reduce the problem of the rigid tracker cutting out some parts of the target.

Although there is still room for improvement, we were able to combine non-rigid tracking with multi-object tracking, something that is not always taken into consideration by other approaches.

References

[1] A. Patel, *Hands-On Unsupervised Learning Using Python: How to Build Applied Machine Learning Solutions from Unlabeled Data*. O'Reilly Media, 2019.

[2] D. G. Lowe, "Object recognition from local scale-invariant features," vol. 2, 1999.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," 2011.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *International Joint Conference on Artificial Intelligence*, p. 674–679, 1981.

[5] A. Lukežič, T. Vojíř, L. Čehovin Zajc, J. Matas, and M. Kristan, "Discriminative correlation filter tracker with channel and spatial reliability," *International Journal of Computer Vision*, p. 671–688, 2018.

[6] G. H. C. Lin, C. Pun, "Highly non-rigid video object tracking using segment-based object candidates," *Multimedia Tools and Applications*, 2017.

[7] S. Manen, M. Guillaumin, and L. Gool, "Prime object proposals with randomized prim's algorithm," *2013 IEEE International Conference on Computer Vision*, pp. 2536–2543, 2013.

[8] L. Breiman, "Random forests," *Machine Learning 45*, pp. 5–32, 2001.

[9] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," *Association for Computing Machinery*, p. 309–314, 2004.