Design and implementation of the mean-shift technique to detect and track persons

---

Short Project

REPORT

Computer Vision

Fàtima El Baghdadi

David Pérez Ruiz

Master's degree in Automatic Control and Robotics

December 2019

Barcelona, Spain

Contents

List of Figures

Abstract

Nowadays there are many tracking applications for things, people, objects, etc with different objectives. In this work, we are interested in understanding how the tracking mechanisms and the theory behind this process work for a very particular tracking algorithm: the Mean Shift, introduced by Comaniciu et al [2]. This method has been explained in class and in this work we will deal with it in depth. We will apply it for persons, but also for objects. The algorithm will be implemented in real time on already recorded videos to test its functionality and performance. The first step will be to define what the Mean Shift algorithm is, we will explain the theory behind this method and then we will show the our implementation's results by displaying several frames of the video with the algorithm applied to assess its performance. The code lies with the zip in which document is contained.

1 Mean Shift algorithm explained

1.1 The Mean Shift

To implement the mean shift method in tracking objects or people it is important to have a conceptual idea of the steps to follow. First at all, we are dealing with videos so we will be dealing with the different frames of the video sequence. The general idea of this method is to locate the initial point that we want to follow in the window that we choose in the first frame, and obtain its color values (or grayscale value in case it is not RGB). Once the mentioned characteristics are located, we consider the subsequent frames with a window in the initial position and calculate the average position of the pixels that have the same color as the ones that we have initially defined as objectives. This process is repeated in all frames. However, if a point that has the desired color range can't be found, we maintain the previous center and reapply the procedure to calculate the next center in the next frame. The average, as we have said, is done with the pixels of the same color as the objective and the weights involved are calculated with a kernel. The kernel gives more weight to the pixels that are near the center of the window from which those that are further. This is done in order to track close points and avoid moving too much the window, which could result in tracking another undesired target. Then, the average is made with the (normalized) weights obtained from the kernel and the frame points that have the same color as the initial target. This weighted mean can be seen in equation 1. There are three kernel types that can be used, as mentioned in [2], which we show in formula and graphically in figure 1. These are called Epanechnikov's Kernel, Gaussian Kernel, and the Uniform Kernel. We have chosen the Epanechnikov kernel since it is the one used in the original paper [2].

$$\mathbf{y} = \frac{\sum_{i=1}^{n \in W} K(\mathbf{x}_i) * (\mathbf{y}_0 - \mathbf{w} + \mathbf{x}_i)}{\sum_{i=1}^{n \in W} K(\mathbf{x}_i)}, \tag{1}$$

Where $\mathbf{y}$ is the next position in the global coordinates of the image. $K(\mathbf{x}_i)$ is the kernel function evaluated at $\mathbf{x}_i$, which is the position relative to the kernel window at every point i of said window. $\mathbf{y}_0$ is the last position found by the mean-shift procedure in the last iteration. $\mathbf{w}$ is the window's half size in both the x and y positions, and is a fixed constant defined by the user. It defines how big the window of search must be, and can take arbitrary values. It chosen poorly, it can hinder the program's performance. Finally, $b(\mathbf{x}_i)$ is the bin value of a point's position $\mathbf{x}_i$. Note that the summation is done only for all points i that belong to the window W.

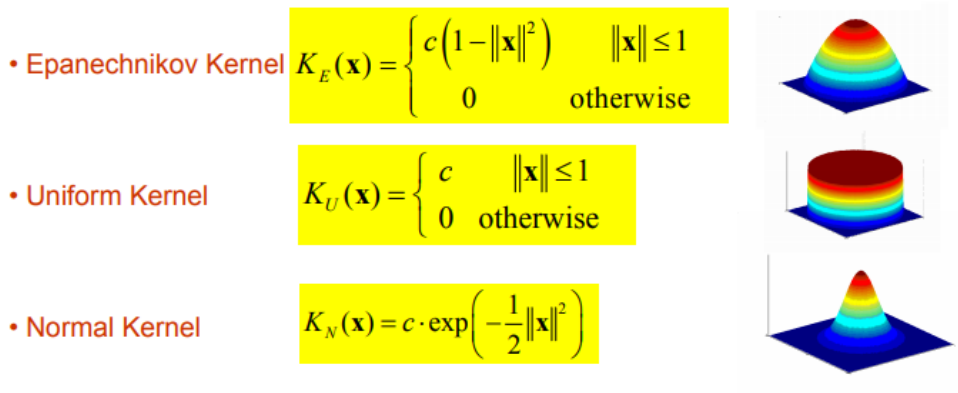

Figure 1: Types of kernels that are most used. Note that $\mathbf{x}$ is normalized. Image extracted from [1].

1.2 The algorithm implementation

Given a video input, the user has to select the position in the initial frame the object/person that needs to be tracked (the detection problem is not tackled in this work), along with a window dimension coherent with the pixel dimensions of the object/person that needs to be tracked. After this initial step, the next frame of the video is loaded and the kernel is applied to the window whose dimensions where specified earlier, and the mean shift formula (see equation 1) is used. This can yield two options: the new object/person position y_1 is a number, which is rounded, so the next frame can be loaded and the process is repeated taking the new window whose center is y_1; or the new object/person position y_1 is not a number (NaN, since y_1's value went to infinite) and the next frame is loaded taking the same window as last iteration. This last disjunctive is important to avoid the loose of tracking induced by spontaneous partial or total occlusions. The flow of the algorithm that has just been explained can be observed in the diagram of figure 2. It has to be noted that the procedure for tracking both grayscale and RGB videos is similar. The difference is that in grayscale it is treated as a single channel, while RGB it is treated as 3 channels at the same time. As a matter of fact, in the algorithm we have implemented a method to detect if the image is RGB or grayscale in order to apply the proper method. For testing purposes, there is also the option to convert an RGB image to grayscale before beginning the algorithm.

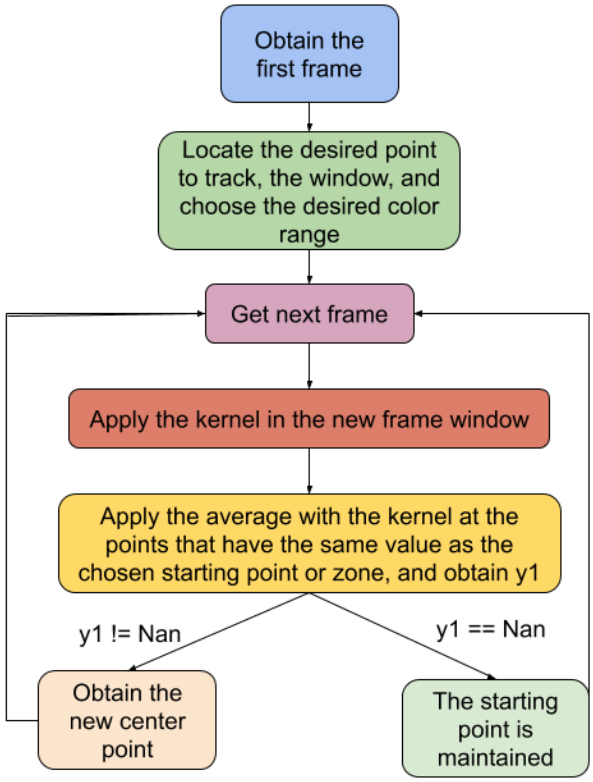

Figure 2: Diagram of the algorithm that we have implemented.

The results obtained using the explained algorithm are shown in the next section.

2 Results and discussion

In this section the algorithm performance is assessed with several example videos. The videos chosen are:

1. Man walking, being crossed by several people to assess the robustness of the algorithm in front of partial occlusions and confusions in front of similar persons.

2. Athletes running, useful to assess its ability to not get confused among the other persons, specially being the camera so far away. The speed of the athletes is also a challenge for the tracking system to keep up.

2.1 First video: Man walking

Note: This video was taken from a test performed by another author to test their own tracking method, that's why there is a green rectangle. Ours is the red one with a star in the center (the star representing the $\mathbf{y}_1$ value obtained using the mean shift).

First, we show a video of a walking man who crosses paths with different people. The method discussed above is applied first in gryascale and then in RGB. To asses the performance, we show the frames that describe how the person is tracked and if it is confused or not. As it can be seen in figures 3 and 4, it gets confused very easily, which is understandable due to the fact that grayscale provides very little information to distinguish one person form another. It can be seen in images from 5 to 15 that the tracker, given an RGB video, can keep track very well of the initial person. It is particularly robust to people crossing in its way,

specially if their dressing contrasts in colour. Still, it gets confused when persons with the same coloured clothes stepping from behind, as it can be seen in images 16, 17 and 18.

Figure 3: The objetive to track.

Figure 4: The tracker is confused with another person with black clothing.

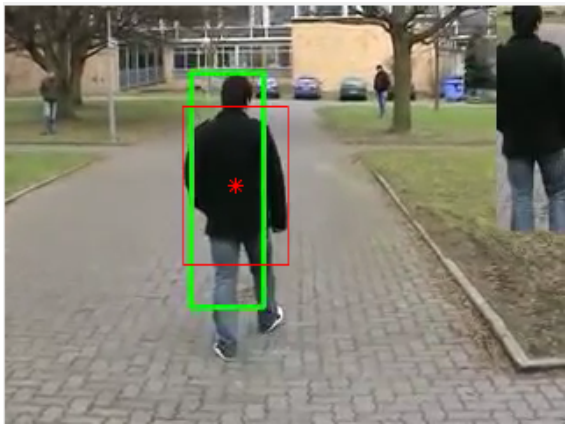

Figure 5: RGB target is tracked properly.

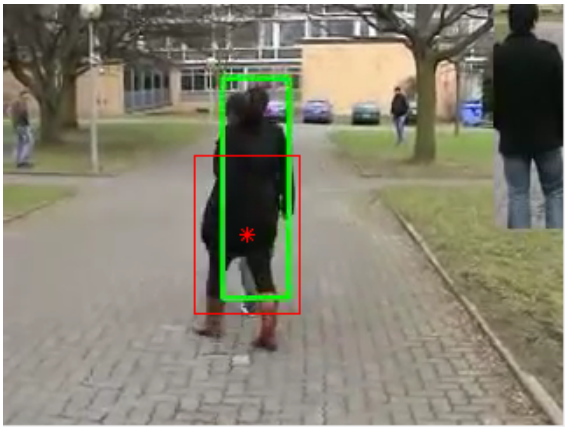

Figure 6: The same woman crosses from behind.

Figure 7: It manages to keep tracking the initial person and not get confused.

Figure 8: Another man crosses from the left side.

Figure 9: Thanks to its distinctive clothes and the fact that it is an RGB video, the tracker does not get confused.

Figure 10: A third person with similar clothes crosses from the side.

Figure 11: Due to the fact that it walks somewhat fast, the tracker manages to stay on target.

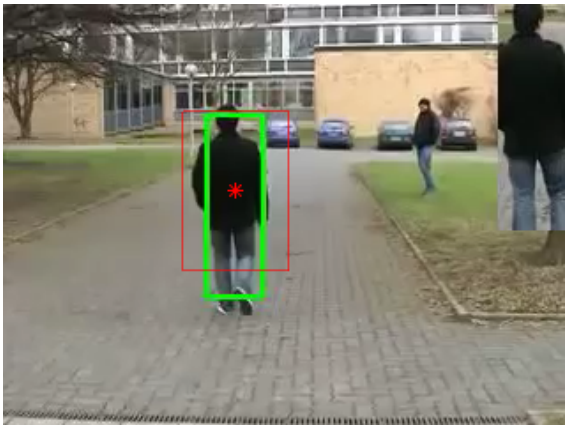

Figure 12: Keep tracking the target person.

Figure 13: Again, another person with a black jacket crosses again.

Figure 14: As in 11, it does not get confused.

Figure 15: Keep tracking the target person.

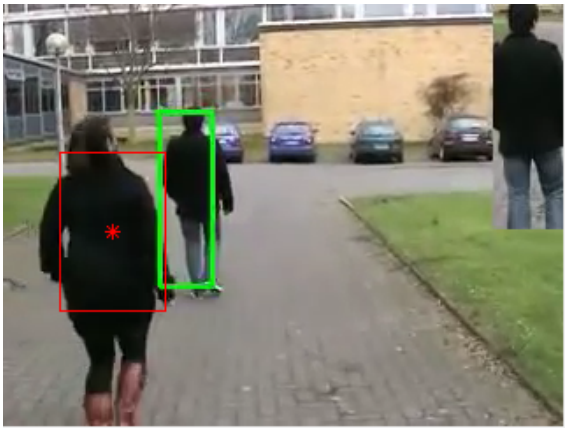

Figure 16: The tracker is confused when the women with black clothes crosses from behind.

Figure 17: The tracker is confused again when the man with black clothes crosses from behind.

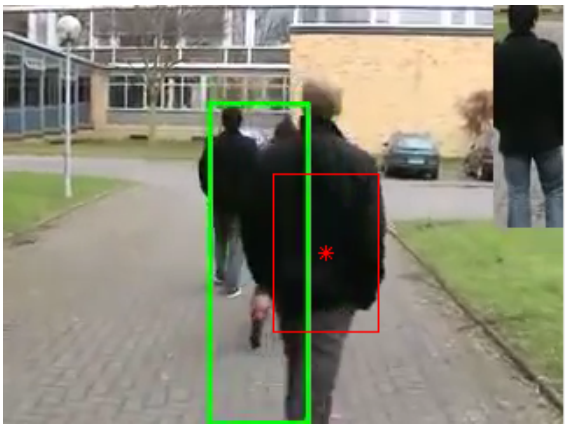

Figure 18: Finally, it follows the man with whom the tracker has been confused.

2.2 Second video: Athletes running

Note: The tracker's position is the red star displayed on top of the image, do not mind the fact that sometimes there is no rectangle. The video of athletes running from a somewhat far distance is ideal to test its robustness against confusions with other persons and its ability to keep up when the target moves at a high speed. It has to be said that the camera that recorded the video follows as do the athletes, so it is a bit lenient on that regard.

In this test, the objective is to follow the african athlete inside the red square in image 19 all throughout the race. As it can bee seen in images 20 and 21, it keeps track of the target when the background is somewhat clear, but is on the verge of getting confused when the athletes are very close together as it can be observed in image 22. Luckily, it maintains track all throughout the race.

Figure 19: The objetive.

Figure 20: Tracking of the target successful at this point.

Figure 21: Keep tracking the target.

Figure 22: It is on the verge of getting confused with the athlete coming from behind.

Figure 23: In this change of scenes (from a camera situated up high to another at ground level) the tracker should theoretically have lost track completely, but since the position of the athlete is (luckily) the same in the x and y coordinates with respect to the last scene, it does not loose track. In this instance, of course, it can stay on target easily being both athletes separated and no occlusions whatsoever.

2.3 Third video: Usain Bolt running

The video of Usain Bolt racing is perhaps the video in which the tracking has performed worst. It does a very good tracking while the camera recording follows the runners and Usain is well away from its competitors, as it can be seen in images 24 and 25. But just when the camera stops turning around so quickly to follow the athletes, their relative speed with respect to the viewer increases. In that moment, the tracker fails to keep up and begins tracking the second runner (see image 26). As the third runner approaches from behind and touches the bounding box, the tracker decides to follow him, which is understandable due to the fact that his color scheme is the same as Usain (see image 27). Just as this third runner moves further and becomes smaller, the tracker gets lost and gets completely lost in the grades.

Figure 24: The tracker is able to follow Usain Bolt and not get confused with the other runners.

Figure 25: Up to this point, it has had no problems tracking.

Figure 26: When the camera slows done it fails to track Usain.

Figure 27: Just as a third runner with the same coloured-clothing as Usain approaches, the tracker decides to follow it.

2.4 Discussion

Results are as expected due to the limitations of the mean shift algorithm. Given an RGB video it is able to keep track of the target selected by the user as long as no other target with the same colours gets on the way of the tracker enough time for it to get confused. That is, if a similar person occludes the tracked objective briefly (like in the walking man video, where a man with same clothing crosses in a fast manner), there tends to be no problem. But, when a person with similar clothing occludes for enough time, the tracker will tend to get confused and follow that new person. The athletes running represents a good example of this: when occlusions are scarce and done by persons with different color schemes the tracking keeps on on point. Usain Bolt racing, on the other hand, presents the challenge of persons moving at a relatively high speed in an occluded environment (athlethes occluding each other), so confusions do occur. It has to be noted that the window size $\mathbf{w}$ for which the mean shift algorithm is being applied does not vary depending on how close the person is, so it will likely misfit the target at some point. As a side note, it is important to note is that our code "almost" runs in real time.

3 Conclusions

The scope of this work is to develop an algorithm to detect and track persons using the Mean Shift procedure. The part of detecting could not be fulfilled due to the complexity that it entails, so it has been substituted by the manual selection of the desired target persons at the first frame by giving as input their x and y positions of the center of mass. The window size $\mathbf{w}$ is also input by the user taking into account the dimensions in pixels of the target person. Said window size is constant all throughout the video, which is another point of contention, since it theoretically should shrink or enlarge depending on how close the person is. Tracking, on the other hand, has given good results using the Mean Shift algorithm. Once it is given a starting point, the tracking tends to perform well even when there are brief partial or total occlusions. It is, however, sensible to fast moving targets or large occlusions that persist in time, since it tends to lead into mistakes, whether by tracking the wrong person or simply getting lost in the video frame. In terms of performance, it falls a bit short: the original author at [2] was able to run the mean-shift algorithm on a 600MHz processor in real time (maintaining around 30 fps[1]), while our implementation runs at around $15 \sim 20$ fps on a modern PC. Clearly there is leeway for code optimization.

References

[1] B. S. Yaron Ukrainitz, "Mean shift theory and applications," University Lecture.

[2] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, vol. 2, June 2000, pp. 142–149 vol.2.

[1] fps stands for frames per second.