Face recognition and tracking

Lluís-Pere de las Heras Caballero   Ahmed Mounir Gad
Mónica Piñol Naranjo

March 17, 2010

1 Introduction

- Simple Tracker

$$\begin{tikzcd}
\boxed{Segmentation} \arrow[r] & \boxed{Detection} \arrow[r] & \boxed{Representation} \arrow[r] & \boxed{Tracking}
\end{tikzcd}$$

- Our Tracker

$$\begin{tikzcd}
\boxed{Segmentation} \arrow[r] \arrow[d] & \boxed{Detection} \arrow[r] \arrow[d] & \boxed{Representation} \arrow[r] & \boxed{Tracking} \\
\boxed{Grey - World} \arrow[d] & \boxed{FaceDetection} \arrow[d] & & \\
\boxed{Eigenbackground} & \boxed{Facerecognition} & &
\end{tikzcd}$$

In this project our main focus is to build a single and multiple target tracker. We extend this approach to build a useful real-life application to that tracker which is a face recognizer and tracker.

Our project first starts the normal tracking process which is segmentation and only determining the useful parts of the scene. We use these useful parts of the scene to detect the faces and correspond these faces to the interesting faces in the scene. We afterwards calculate the velocity of the person having this face and we keep tracking until the person leaves the scene or our tracker fails.

2 Segmentation

The process of segmentation is to separate the background to foreground.

- The Background subtraction use three parameters: **gamma**,**tau** and **radius**.

$$B_{i+1} = \gamma * F_i + (1 - \gamma) * B_i \tag{1}$$

  - Gamma is the learning rate, usually is 0,05.
  - Tau is the different between frame to background.

  - Radius is the parameter to delete the little blobs.

- The *background subtraction* is not robustness to change illumination, for this we implement *Grey-World* because is invariant to illuminant changes. The Grey-World method try to eliminate the grey in the image, for this reason move the minimum and the maximum values.

$$(R'G'B') = \begin{pmatrix} \frac{R_{grey}}{R_m(I)} & 0 & 0 \\ 0 & \frac{G_{grey}}{G_m(I)} & 0 \\ 0 & 0 & \frac{B_{grey}}{B_m(I)} \end{pmatrix} * \begin{pmatrix} R \\ G \\ B \end{pmatrix} \tag{2}$$

- The *selectivity* is the other method that use the simple different between foreground with background but the selectivity a prior knows that pixel is or isn't foreground.

$$\begin{cases} B_{i+1}(x,y) = \alpha * F_t(x,y) + (1-\alpha) * B_t(x,y) & if F_t(x,y) is Background \\ B_{i+1}(x,y) = B_t(x,y) & if F_t(x,y) is Foreground \end{cases} \tag{3}$$

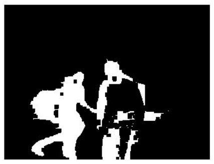
- The *eigenbackground* method use the Principal Component Analisys (PCA) to reduce the dimensionality. Also, the first M eigenvectors are the eigenbackground.

  eigenbackground = eigenvectors(:,1:M)
  Foreground = I - I'
  I' = Image Projection and Reconstruction (Background)

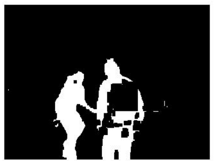
3 Detection

Now the segmenter has finished its work and we have a number of blobs that correspond to the changes in the background. Some of these changes are definitely the objects we are interested in. This step in the tracker mainly focuses on detecting the important parts of these background changes.

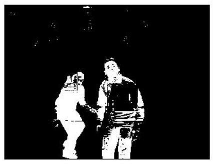
The basic detector provided in our framework is based on simple blob detection. It just returns each set of connected pixels as an object. We basically have two questions to investigate about. The first is how the basic



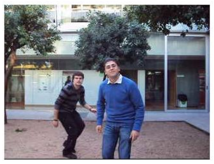
(a) Background Subtraction



(b) Selectivity



(c) Eigenbackground



(d) Frame 300

Figure 1: Different segmentation with the same frame

tracker provided behave in the presence of noisy detection. The second is how the tracker behaves in the case of no detection. The answers to these two questions are illustrated in the tracking section.

We did several optimizations to the simple detector provided. The first was to detect faces and keep tracking those detected faces. We thought of an interesting application that may use this technique. We added a face recognizer that only detects faces of people of interest to the tracker. We provide a trained Support Vector Machine(SVM) classifier to our tracker. We train it using labeled faces detected from a video for 5 persons. In total 364 faces for 5 people were used for training and all taken from one single video. The detector then will only detect the people of interest in any video and the tracker will only track those people.

In case a match was found, the detector will return the bounding box of the face that made that match. We thought as an improvement we may try to detect the whole body of the person who made that match. However, plenty of noise coming from the background made this process hard and time consuming. We stopped at this step only satisfied with detecting the face's bounding box and passing it to the representer.

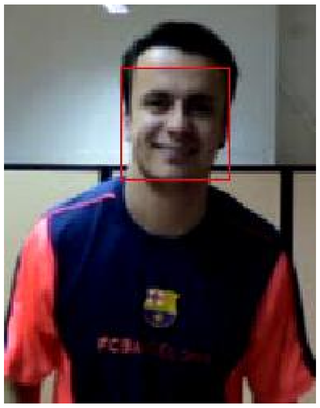


Figure 2: Face detected and recognized as Lluis

4 Representation

The representer takes the results of the detection module and computes a *representation* of each recognized object to be tracked. In the basic tracker, the representer extracts the BoundingBox and size of each blob and only keeps the biggest one as the single object being tracked.

The requirements in this module was to not only extract the bounding box and the size, but also determine the velocity and a local histogram for the object being tracked.

The velocity was calculated based on the location of the *centroid* of the object being tracked as follows:

$$velocity = (centroid - old\_centroid) * fps \tag{4}$$

where *fps* is the frames per second in the video. Therefore, in this case we have to keep the centroid of the previous object being tracked.

For the histogram, we calculated the color histogram for the object being tracked.

According to our detection methodology. Our new representer now has to solve the correspondance problem between the detected objects in different frames. There are two cases that we have to consider.

Case 1, the detector already recognized the person. In this case we don't have to do anything. The correspondance probelm is already solved by the detector.

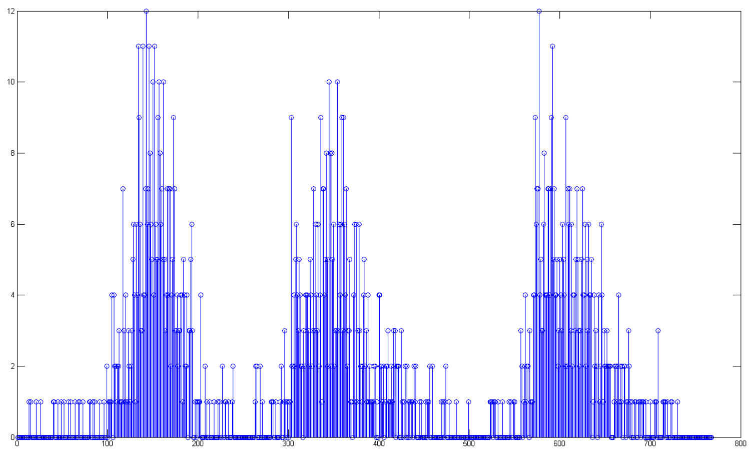


Figure 3: RGB color histogram concatenated together on the same axis

Case 2 is when the detector fails to detect the face of the person. However, this person is still in the scene. We then have to find a way to correspond the blobs in the scene with the people being tracked. We do this by getting the minimum euclidean distance between the previously represented blobs and the newely detected unrecognized blobs. We also make use of the prediction coming from the tracker as we try to find the minimum distance between the unrecognized blobs coming from the detector. In our case the usage of the color histogram will be redundant. The colors of the faces won't be of a high discriminative power for different blobs in the scene. However, If we have implemented the person's tracker. In this case it would had been an advantage to also make use of the colors histogram. Clothes of the people can be discriminative for the detected persons.

5 Tracking

In this section we will discuss the main module of the project which is the tracker. A simple Kalman filter tracker has been provided in the tracking framework. It uses results of the representer to track the position and the extent of the object being tracked. The Kalman filter works by estimating an unobservable state which is updated in time with a linear state update and additive Gaussian noise. It uses the technique described in [1].

The basic Kalman filter provided in the framework estimates the position and the extent of the target. We also included the velocity and did the necessary modifications. Now the initializations for the Kalman filter is done as shown in the following snippet:

```
Tracker.H          = eye(6);         % System model
Tracker.Q          = 0.1 * eye(6);   % System noise
traker_state       = eye(6);         % Measurement model
traker_state(1,3)  = 1;
traker_state(2,4)  = 1;
Tracker.F          = traker_state;
Tracker.R          = 5 * eye(6);     % Measurement noise
Tracker.innovation = 0;
Tracker.track      = @multiple_kalman_step2;
```

As we see we have to change the values of the update function F as well. It's no longer the identity matrix. Also we had to change the representation of the measurements of the state variable to tolerate these changes.

We have to discuss 3 main issues in the tracker. The first issue is the effect of changing the noise parameters on the behaviour of the tracker. The second is the effect of having noise detections on the tracker. The third which was previously refered to in the representer is the effect of getting no readings from the detection on the tracker.

Concerning the effect of the noise parameters changings. We observed that

About the effect of noisy detections,

Finally, when the detector fails to detect anything

6 Conclusion

Tracking of visual phenomena is a very hard and frustrating task. It depends on many parameters that are very hard to be managed or correctly handled. The background subtractor is hardly going to be perfect. The lighting conditions and the quality of the camera play an important role. The detector especially when detecting specific thing like a face can hardly be robust. Also when attempting to recognize and depend on classification then the problem becomes more and more complex. According to these problems associating the measurements with the object being tracked can still be very hectic. Finally, all of these factors affect the tracker making the tracking problem very hard.

References

[1] Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2001.