Face recognition and tracking

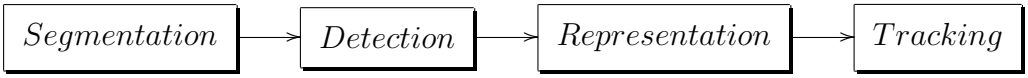
Lluís-Pere de las Heras Caballero Ahmed Mounir Gad
Mónica Piñol Naranjo

March 18, 2010

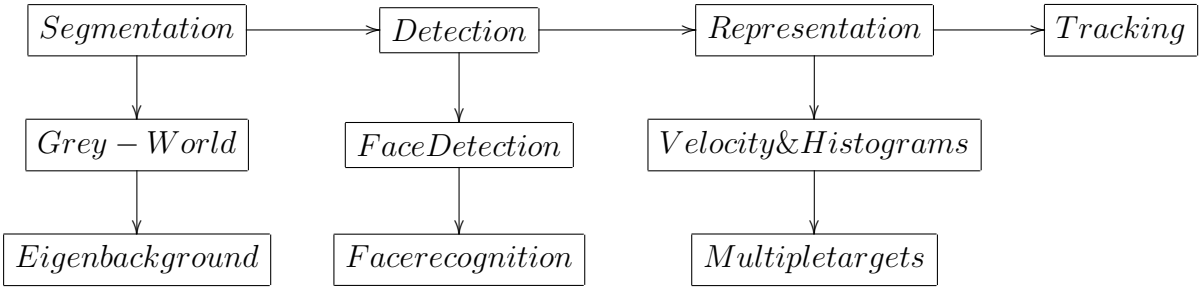
Contents

1 Introduction

- Simple Tracker



- Our Tracker



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.

The first step of our tracking process is **Segmentation**, which is the responsible of separate the foreground from the background . We use the foreground of the scene to detect the faces and correspond them to the ones which we are interested in. Afterwards, we calculate the velocity of the person having his/her face and we keep tracking until the person leaves the scene or our tracker fails.

2 Segmentation

The aim of **Segmentation** is to separate the foreground from the background.

- 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, thus we implement *Grey-World* which is an invariant to illuminant changes method. 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 other method that uses the simple difference of foreground and background. It uses the prior knowledge weather that pixel is or isn't foreground.

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

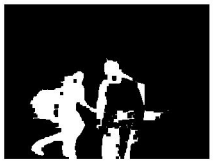
- The *eigenbackground* method use the Principal Component Analisys (PCA) to reduce the dimensionality. Also, the first M eigenvectors are the eigenbackground.

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

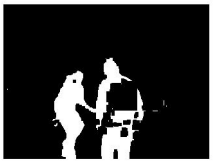
3 Detection

Up to here, 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.

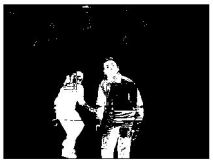
The basic detector provided in our framework is based on simple blob detection. It just returns each set of connected pixels as an object. We



(a) Background Subtraction



(b) Selectivity



(c) Eigenbackground

(d) Frame 300

Figure 1: Different segmentation with the same frame

basically have two questions to investigate about. The first is how the basic 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. It was used the Viola and Jones AdaBoost Face-Detector[2]. 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

On the other hand, the **Detector** may not be able to detect and recognize any face in a certain frame. In this case, the **Detector** return all those blobs that their *bounding-box* is larger than appropriate threshold. This is done to delete too small noisy blobs and just being interested in those ones that could be a person. The representer will be the responsible to assure if these blobs are belonging to people who was tracked in the last frame.

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 bounding-box 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.

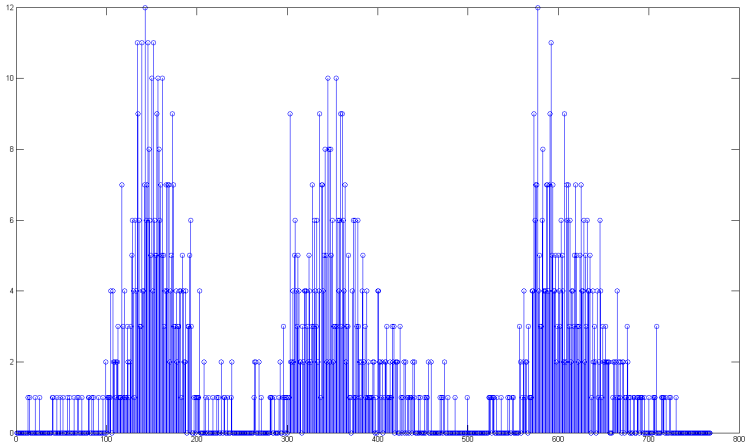


Figure 3: RGB color histogram concatenated together on the same axis

According to our detection methodology. Our new **Representer** now has to solve the correspondence 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 correspondence problem is already solved by the detector.

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 - with a label -. We do this by getting the minimum euclidean distance between the previously prediction of the tracker for a label and the newly detected unrecognised blobs. If the closest blob is nearer than an appropriate threshold we assume that this is the new blob for the certain label. In our case the usage of the color histogram will be redundant. The colours 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 instead of a face tracker. In this case it would had been an advantage to also make use of the color 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;
```

Our new measurements obtained from the **Representer** are compounded by the centroid of the object, its velocity and the blob height and width as follows:

$$\mathrm{m}_k = [x_k, y_k, \dot{x}_k, \dot{y}_k, h_k, w_k] \tag{5}$$

- $\mathrm{x}_t and y_t are the position in both x and y coordinates. \dot{x}_t and \dot{y}_t are the velocities in each component.$
- $\mathrm{h}_t and w_t are the height and the with of the bounding - box of the object.$

Now, since we take into account the velocity as a measurement in Kalman filtering, we also have changed the transition model of the filter. F is no longer an identity matrix, but adds the velocity in each position compound.

The new composition of the measurements and the transition model makes Kalman filter be able to predict also the size of the new blob. This makes a lot of sense since it can be useful to assume precisely which is the corresponding blob in the next frame. Moreover, in order to improve more the tracker, we played with the internal parameters of the Kalman filter. Those are the *System Noise* **K.Q** and the *Measurement Noise* **K.R**.

Regarding the *System Noise K.Q* of the filtering, we can observe that giving it higher values the tracker follows faster the *Representer* blob. This is because it makes increase the *Kalman Gain* which is the believing in *innovation* of the filter. The *innovation* is the difference between the new real measurement of the Representer and the predicted - a priori - state. So, the new prediction - a posteriori - which is the prediction from the state before plus the Kalman trust in the innovation, gives more closely measurement approximation to the real one. This could be very useful whether the measurements obtained are very trusty. Otherwise, if the detection - for any reason - is not able to obtain correct blobs, the tracker will follow fastly its wrong measurements. In those cases, if the *System Noise* is small, we can smoothly follow the object.

On the other hand, the *Measurement Noise K.R* with higher values the *Kalman Prediction* trust less the measurements. *K.R* is added to calculate the state covariance which its invert is used to obtain the *Kalman Gain* - lower in this case -. As we have said before, the *Kalman Gain* is the degree of believe on the *innovation*. With low values, the measurements are less trusted, so the tracking follows smoothly the prediction.

One of the highest problems we have in the tracking step is caused by the **Detector**. We are trying to track faces, but the faces that are found by the

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. Fi-

nally, 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.