Face Tracking in Video

Hamidreza Khazaei and Pegah Tootoonchi Afshar
Stanford University
350 Serra Mall
Stanford, CA 94305, USA

I. Introduction

Object tracking is a hot area of research, and has many practical applications. These applications include navigation, security, robotics, vehicular communications, and etc. It has also found applications in biology, where people are studying cells and mitosis. One of the biggest applications of object tracking is being able to track a face.

In a face tracking application, Boosting and cascading detectors have gained great popularity due to the their efficiency in selecting features for faces. However these detectors suffer from high miss-classification rates. In addition they depend on the orientation of the face. They require that the face be in a full frontal position. If there is any deviations from this position (the face is tilted by 45 degrees, or the face turns to a profile position) the AdaBoost fails to detect a face.

We will implement a face tracking algorithm, and compare it to other well known tracking algorithm presented in the literature. Specifically we will be enhancing the AdaBoost algorithm, and comparing the resulting system to one that uses AdaBoost only. There are three components to our algorithm: image representation, appearance model, and motion model. Image representation is done using Haar like features. These features are calculated efficiently using integral image representation, which allows the computation of these features to be done in constant time. Once the features have been attained the face detection is performed using two classifiers, the AdaBoost algorithm creates a classifier by combining a set of weak classifiers. The motion model tracks the aspect ratio, and the center of the face that it has detected.

The paper is organized as follows. Section II will provide the necessary background and present a detailed description of our model, and will provide the necessary background information. Section IV will present the results. Section V will serve as a conclusion, and will discuss possible future works.

II. Background and System Model

A. Haar features

The features used to represent the pictures are the haar features that were originally proposed by Papageorgiou et al. [1]. There are three types of rectangular features that can be computed. For example a two-rectangular feature can be computed by the difference between the sum of two rectangular blocks. Viola-Jones proposed a fast method for computing these features making the features an attractive option for face detection [2]. Their method, which is called integral image and can be calculated with one pass over the picture, uses an intermediate array to store a running sum of pixel above and to the left of the point x,y:

$$ii(x,y) = \sum_{x' \leq x, y' \leq y} i'(x',y'), \tag{1}$$

where $ii(x,y)$ is the integral image, and $i'(x,y)$ is the original image. Now using this representation for the image any rectangular sum, and thus haar features, can be computed efficiently. For example the rectangular region D in figure 1 can be computed by $ii(4) - ii(3) - ii(2) + ii(1)$.

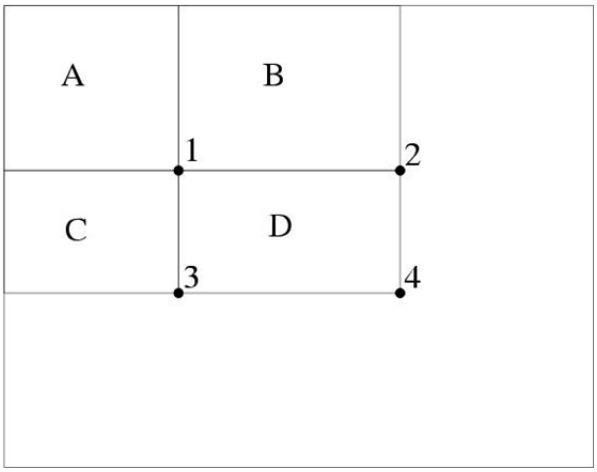

Fig. 1. Demonstration of the way integral image is used to compute the sum of pixels in a rectangle.

B. AdaBoost

The number of Haar-like features in each window is much more than the number of pixels, and it is not possible to efficiently classify the image using all the features. However, an effective classifiers can be created by using only a small subset of these features. The algorithm used in this work is the AdaBoost [3] algorithm which extracts the essential features to create a well developed classifier. The AdaBoost algorithm creates a classifier by combining a set of "weak classifiers", which are trained on only one feature.

We can explain the algorithm in two steps. First, each weak classifier optimizes its performance, i.e. minimizes the classification error over the training set. In this application, a weak classifier has a feature (f_i), threshold (θ_i) and a parity (p_i). It tries to find the optimal threshold that correctly classifies the maximum possible number of examples. The parity is used to determine the direction of inequality sign.

$$h_j(x) = \begin{cases} x & \text{if } p_j f_j \leq p_j\ \theta_j \\ 0 & \text{otherwise} \end{cases}$$

After the first round of learning, a weight is assigned to each training example (in the first round they were all equally weighted). In the next step, a strong classifier is constructed by combining well-performed weak classifiers. It is shown that the training error of the strong classifier approaches zero exponentially in the number of rounds [3]. The AdaBoost algorithm is summarized below.

- Given training examples (x_i,y_i) where $y_i = 0, 1$ for negetive and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negative and positive examples respectively.
- for t = 1,...,T
  - Normalize the weights $w_{t,i} = \frac{w_{t,i}}{\sum_j w_{t,j}}$
  - Train each classifier h_j corresponding to feature f_j. The weighted error with respect to w_t is $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$
  - Choose the classifier h_t with the lowest error ϵ_t.
  - Update the weights, $w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$, where $e_i = 0$ if example is classified correctly and 1, otherwise; And $\beta = \frac{\epsilon_t}{1-\epsilon_t}$
- the final strong classifier is

$$h(x) = \begin{cases} 1 & \sum_t \alpha_t h_t(x) \geq \frac{1}{2} \sum_t \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

C. Background-Subtraction

Given a frame sequence from a fixed camera, detecting all the foreground objects is based on the difference between the current frame and the stored (and frequently updated) image of the background:

if $|Frame_{ij} \quad Background_{ij}| \geq Threshould_{ij}$
$Frame_{ij}$ *is foreground*
else
$Frame_{ij}$ *is background*

The threshold was chosen proportional to the variance of pixel values of the current frame (for a more complex model each pixel could have a different threshold based on its own gradual changes). After detecting the foreground objects, the background image is updated as follows:

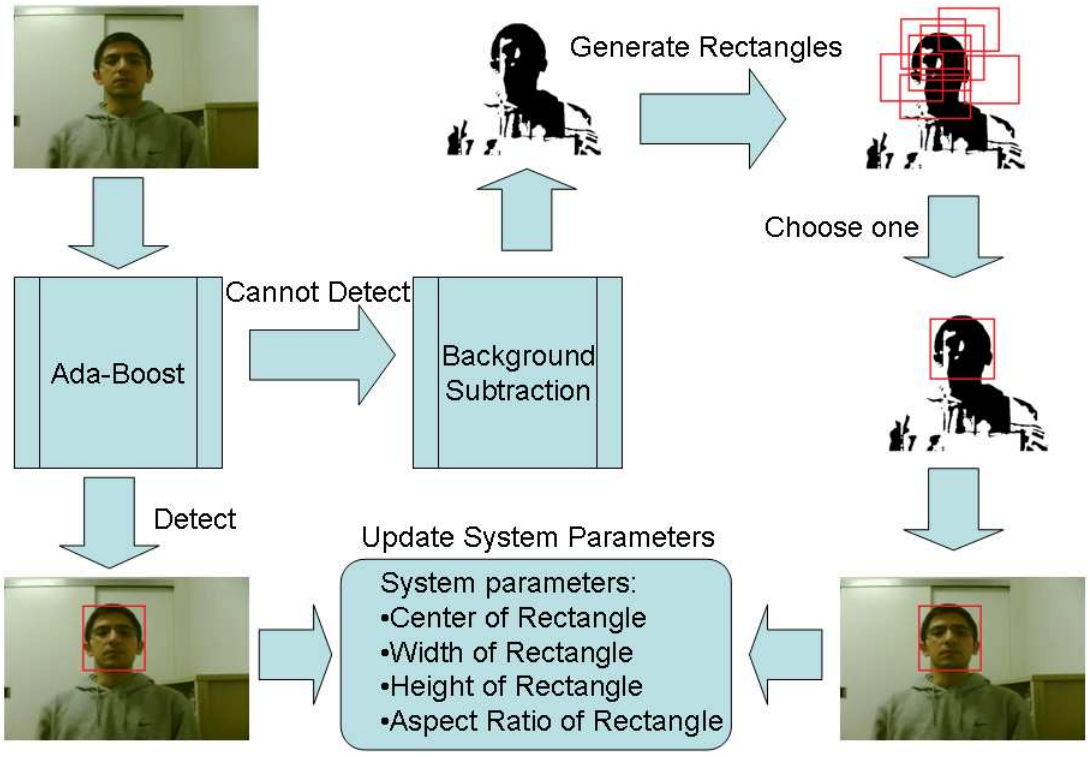

Fig. 2. Overview of the face tracking system.

if $Frame_{ij}$ *is detected as background*

$Background_{ij} = \alpha\ Frame_{ij} + 1\text{-}\alpha\ Background_{ij}$

else

$Background_{ij} = Background_{ij}$

Where α is the updating rate which was chosen to be a small value. Furthermore in order to eliminate the noise from the subtracted foreground image we used the common low pass filters in image processing called dilation and erosion filters. Dilation, in general, increases the size of objects and erosion causes objects to shrink in size. The amount and the way that they affect the objects depend on their structuring elements. To explain them simply, dilation take each binary object pixel (value 1) and set all background pixels (value 0) that are connected to it (determined by the structuring element) to value 1; but erosion takes each binary object pixel (value 1) which is connected to background pixels and set its value to 0. [5]

D. Overall Model

In order to display the results the face tracking system draws a rectangle around the region that it has recognized to be a face. The system will maintain a set of parameters, which are the center, width, height and the aspect ratio of the rectangle from the previous frame and the current frame.

The system Initializes when the AdaBoost classifier detects the first instance of a face, and initialize all the parameters of the system accordingly. In the next frame the AdaBoost classifier will try to draw a rectangle around the position of the face. If this classifier could not find a face or if the rectangle's parameters are significantly different from the previous rectangle, then the background-subtraction algorithm that was describe in section II-C will process the image, and remove the

background. The system will then draw a number of random rectangles, $N_{rectangles}$, near the vicinity of the previous center. Within each rectangle the sum of the pixels is calculated, and the rectangle with the maximum sum is selected. The parameters are then updated using this rectangle, so they can be used in the next frame. An overview of the entire system can be seen in figureřeffig:SystemModel.

III. Experiment setup

A 480 x 720 video, with 1803 frames was used to test the system. As described in section II-D when the parameters of the current rectangle are significantly different from the parameters of the previous rectangle, the background subtraction algorithm is used. Two criterions were used to determine whether the previous rectangle and the current rectangle are different. First, the difference between the centers of the rectangles were calculated, and if the difference was greater than ten percent of the video width (720) in the x-direction, and ten percent of the video height (480) in the y direction, then the two rectangles were classified as being significantly different. Secondly, if the aspect ratio of the two rectangles differ by more than ten percent, then the two rectangles were classified as being significantly different. The threshold γ that was described in section II-C was set to be sixty percent of the variance of the picture. This threshold was selected to take into consideration the change in lighting of each frame. And finally, the $N_{rectangles}$ parameter described in section II-D was set to ten.

IV. Results

The face tracking system was tested using the experimental setup described above, and was compared to a face tracking system that only uses an Ada-Boost classifier to track faces. The results are summarized in table I. As it can be seen our face tracking system out performed the Ada-Boost face tracking system. We have managed to reduce the errors from 726 frames to 193 frames which is a significant number. It must be noted that the results do not make a distinction between no detection and a false positive detection. In other words we consider a missed frame as either a false positive or no detection.

TABLE I
TABLE COMPARING THE ADABOOST ALGORITHM, AND THE ALGORITHM DESCRIBES BY THIS PAPER

	Number of frames missed (total of 1803 frames).	Percent missed
AdaBoost only Algorithm	726	40.3%
Proposed Algorithm	193	10.7%

V. Conclusions

The face tracking algorithm proposed by this paper does very well, and it was shown that it can enhance performance significantly. The only problem with this algorithm is that it is slow, which means face tracking cannot be done in real-time. Currently, the face tracking system is implemented in Mat-lab, and reading in a frame and processing it takes a long time. In order to increase the speed of the system the algorithm can be implemented in C++, where reading and writing files can be done with ease. Another modification that can be made, is to reduce the number of times the AdaBoost classifier is used to detect the face. Instead of having the AdaBoost classifier look at each frame, the algorithm could be modified so that it calls the AdaBoost classifier every ten or twenty frames. These implementation changes could be laid out in the future to increase the speed of the face tracking algorithm.

References

[1] C. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection", International Conference on Computer Vision, 1998.

[2] P. Viola and M. Jones, "Robust Real-time Object Detection", Second International Workshop on Statistical and Computational Theories of Vision - Modeling, Learning, Computing, and Sampling, 2001.

[3] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In Computational Learning Theory: Eurocolt 95, pages 2337. Springer-Verlag, 1995

[4] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, ”Detecting moving objects, ghosts and shadows in video streams”, IEEE Trans. on Patt. Anal. and Machine Intell., vol. 25, no. 10, Oct. 2003, pp. 1337-1342.

[5] ”Image Processing Fundamentals”, Delft University of Technology.