Report

Complex Systems Project
Detecting human faces

Sergii Voronov 890725-P113
Gothenburg University

June 12, 2012

Contents

Abstract

In this work a new method for face detection is presented. It was tested on still images and its performance is around 93% of correctly identified faces. In this algorithm a skin pixel detector and connected components are used to find the face region. The notion of gradient is exploited to find the face features as eyes, mouth and nostrils.

1 Introduction

During last 20 years humankind made a huge leap in technology and developing of computers. Computers evolved from the big immobile machines, which could occupy a whole floor of the building, to portative devices, but not only size of computers changed. Speed and computational capability increased dramatically. Tasks, that were impossible to solve in the past, are easily solved now. It can even be said that computers become more intelligent, of cause, they cannot think and make decisions like people do and, may be, never will, but at least they can fulfill some tasks independently.

Interaction between computer and human being is a very promising and developing part of computer science. Today it can be seen as many researchers present their products and devices that can interact with human in a natural way. Computers can listen to people, recognize their speech and react according to people's behavior. Face detection is a one of the most important part of that interaction. Face detection is the first step to face recognition, face verification, lip reading, understanding of face expressions, etc.

One can think that face detection is an easy task, because it could be done by human being quite easily. However, it is not the case for the machines(computers). Main problems, that makes computer struggle with this task, are a great variation in scale, orientation, lightning, occlusions and others. In this work a new algorithm for detecting faces is presented. It is supposed to implement the algorithm into a robotic head that should interact with a human being. There are limitations of our algorithm, namely, only one face in the frame is considered and simple background is exploited. Of course, algorithm will be improved in the future and these assumptions will be discarded. Our algorithm consists from several steps. First, the skin pixels are retrieved, then connected components is used to find out where is the face region. Finally, some ideas that will be described below are exploited to find such facial features as eyes and mouth that allows us locate a face.

The remainder of the work is organized as follows. In Section 2 recent advances in face detection area are presented. The implementation of the algorithm is described in Section 3. In Section 4 some results are shown, being obtained using the algorithm. Conclusions and suggestions about the future work are presented in section 5.

2 Recent advances in face detection

During the past decades a significant progress was made by researchers in the field of face detection. One can find many reports and results that originate from knowledge-based methods, template matching methods and appearance-based techniques to boosting-based face detection procedure. In this section a brief overview of the main existing methods is given.

The work by Viola and Jones[2],[3] probably is the most famous and significant work at the beginning of the century. It combines a good speed of the algorithm, about 15 frames per second on Intel PIII 700 MHz processor, with the relatively high accuracy of face detection. There are three cornerstones which form framework of Viola-Jones algorithm, namely, notion of the integral image, cascade tree of classifiers for fast discarding of negative regions, regions that do not contain face features, and learning system AdaBoost to learn classifiers.

A brief description of it will now be given. Integral image $ii(x, y)$ at the point (x, y) is a sum of all pixels that are to the left from x and upper than y, as defined in (1) below. Integral image can be used as the efficient tool for computing a sum of pixels in rectangular areas. On Figure 2.1 can be seen rectangular area $ABCD$ and a sum of pixels inside this area can be computed using (2).

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} ix', y' \quad (1)$$

where $i(x', y')$ is a value of the pixel at point (x, y).

$$ii_{ABCD}(x, y) = ii(D) + ii(A) - ii(B) - ii(C) \quad (2)$$

As can be seen from (2) one only needs four references to compute a sum of the rectangular region. This technique speeds up the computation a lot. Where can the integral image be used? One of the application is to use the integral image computing Haar-like rectangle features. These features can be seen on Figure 2.1a - 2.1f, Viola and Jones are using Haar-like features in classifiers building process.

Cascade structure of classifiers is a key part of the Viola-Jones algorithm. Cascade consists of a set of nodes. Every node contains some number of classifiers which discard negative areas. At first stage one needs few classifiers, because a big number of negative regions should be discarded very fast.

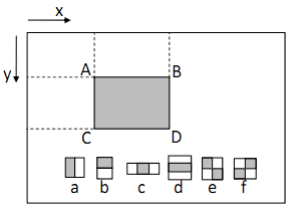

Figure 2.1 Shows rectangular area $ABCD$ and Haar-like features a-f. Picture is taken from [1].

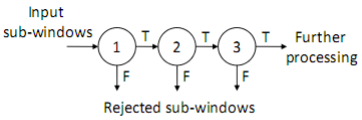

Figure 4. The attentional cascade.

Figure 2.2 Shows cascade tree of classifiers, where 1, 2, 3 denote nodes with classifiers. Picture is taken from [1].

Successive nodes have more and more classifiers as one needs more accuracy to determine negative areas. A one separate classifier can identify negative region with probability around 51% so that for greater probability of rejecting negative areas more classifiers are needed. According to [1] the first five nodes can contain 1, 10, 25, 25, 50 classifiers respectively.

A brief description of AdaBoost algorithm will be given here. Boosting is the algorithm of finding right hypothesis by combining many hepotheses. In AdaBoost a set of training examples as $S = \{(x_i, z_i); i = 1, ..., N\}$ is considered, where x_i belongs to a domain or an instance space X, and z_i belongs to a finite label space Z. In binary classification problems, $Z = \{1, -1\}$, where $z_i = 1$ for positive examples and $z_i = -1$ for negative examples. Then a function $F^T(x) = \sum_{t=1}^{T} f_t(x)$ is built, this function is supposed to predict the label of an input example x, where $F^T(x)$ is a real valued function in the form $F^T : X \to R$. It can be said that AdaBoost learning algorithm finds the best additive base functions $f_t(x)$. That is why, it is assumed that a set of base functions $\{f(x)\}$ is described in a such way that a real feature value $h(x)$ is first extracted from x and $h : X \to R$. For example, Viola and Jones use $h(x)$ as the Haar-like features computed with the integral image, which are mentioned in Figure 2.1 (a-f).

The work of Viola and Jones made a great impact on further progress in

the area of face detection. Many researchers made modifications of Viola-Jones algorithm using it as the basis for their algorithm. However, there are methods that approach the problem of face detection in another way. According to [1] Liu presented a Bayesian discriminating features method for the frontal face detection. He used a multivariate normal distribution to model a face class. He also created a subset of the non-faces areas that lie closest to the face class. Bayesian classifier was a key instrument to make a decision about if the area corresponds to face or non-face area. As only non-face areas closest to the face class were modeled, the most part of the non-faces were discarded during the evaluation.

Another researchers like Romdhani, Later and Ratsch[1] used support vector machines to (SVM) to build face detector. SVMs are known as the maximum margin classifiers, as they simultaneously minimize the empirical classification error and maximize the geometric margin. SVM shows a quite good performance in the machine learning problems, so, researchers started to use them in the face detection problem. However, there is a one disadvantage – slow speed of face detector. During last decade successful attempts were made to speed up the algorithm.

Neural networks were also used to find the face regions. First attempts were made by Rowley and Roth[1]. They created a constrained generative model (CGM). CGM is a fully connected multilayer perceptron with three layers of weights. Some authors proposed the model based on a convolutional neural network. Compared with the traditional neural network, the convolutional neural network derives face features from the training examples automatically.

Many researchers used color-based and model-based tracking. The aim is to retrieve the skin color regions in the case of color-based technique or match regions of the image with a model of the face, for example ellipse. Drawback of these methods is a poor performance when the image contains more then one face and in the case of occlusion. In our algorithm a color-based method is used, because of the assumption that only one face will appear in the frame.

Another interesting approach in face detection is to use the contextual information of the image. Human faces are strongly correlated with the other parts of the body or another objects, at least during some time. The authors [4], [5] believe that one can use this information to facilitate in face detection. The mentioned works are recent, so, some new interesting results are expected in the near future.

However many companies equip digital cameras with face detection software and its performance is quite good, face detection in a completely unconstrained environment remains a very hard task, particularly due to the significant pose and lighting variations. According to [1] modern detectors can achieve about $50-70\%$ detection rate together with about $0.5-3\%$ false positive rate, so, there is a room for the improvement and future work.

3 Face detector. Algorithm

Our face detector consists of four steps. The first step is to classify every pixel from the particular image as a skin pixel or a non-skin pixel. The second step is to find the connected skin regions using connected components and then discard all regions except the largest one. Remember, assumption was made that only one face appears in a frame. When the face region has been found, its height and width can be identified. Using these parameters, a rectangle around the face region can be built. The third step is to find eyes inside the rectangle, the fourth step is to find mouth inside the same rectangle. Below the algorithm is presented in more detail.

3.1 Skin pixel classification

Many color spaces were used in skin detection procedure, namely, HSV, YCrCb, YIQ , CIELAB and others. And it was shown that such color spaces as HSV and YCrCb are good for detecting skin pixels. For example, in [7] researchers have shown that the skin pixels form clusters in HSV space that allow us to separate them from non-skin pixels. The same is true for YCrCb space. It should be noted that all colors of skin are meant when speaking about skin clasters. So, the skin detector will work not only for Caucasian skin, but for the other types of skin too.

In our algorithm YCrCb space is exploited. A brief description of it will now be given. YCbCr representation defines color in terms of a luma component Y and two components Cb and Cr. Luma component represents the brightness of the image and the other components represent the colors, $Cb = b - y$ and $Cr = r - y$, where $r,\ b$ are the channels of the image – red, blue respectively, and y is the brightness component. The fact was mentioned before that skin pixels form clusters in YCrCb space. So, rule can be defined for identifying skin pixels in the following way

$$C_1 \leq Cb \leq C_2$$

$$C_3 \leq Cr \leq C_4$$

If a pixel satisfies two above inequalities, it is treated as a skin pixel, otherwise as a non-skin pixel. It was found by researchers that appropriate values for constants $C_1,\ C_2,\ C_3,\ C_4$ are $C_1 = 77$, $C_2 = 127$, $C_3 = 133$, $C_4 = 173$. On Figure 4.1 - 4.3 can be seen a result of the skin pixel detection. The first image is an original image and the others are the images after processing.

3.2 Connected components

Considering an image after the skin detecting procedure, the skin or non-skin pixels are only available, there is no information if the pixels form some region or not. And if they do, what the shape of the region is, it can be hand, face or some animal. We need additional method that will help us understand a connectivity of the pixels. First step is to binarise the current image. A threshold $T_1 = 50$ is used to binarise the image. After binarising the image, the connected components is exploited to understand the connectivity between pixels. Connected component labeling is used in computer vision to detect connected regions in the binary digital images, however, it can be generalized to color images. There are two versions of this algorithm, namely, 4-connected and 8-connected. 8-connected version is used. On Figure 3.1 the concept of 8-connectivity is demonstrated. Here it is seen the current pixel (red with a dot) and four neighbors (pixels with dots) that form 8-connectivity. Here two-pass connected components is implemented. Relatively simple to implement and understand, the two-pass algorithm iterates through a 2-dimensional image. The algorithm makes two passes over the image: one pass to record equivalences and assign temporary labels and the second to replace each temporary label by the label of its equivalence class. Connectivity check is carried out by checking the labels of the pixels (at the beginning is 0) that are north-east, north, north-west and west of the current pixel (as can be seen on Figure 3.1). If neighbors have labels 0 and the current pixel is foreground, it receives increased by one current maximum label value. If the neighbors pixels have non-zero values of label, the pixel is assigned with the least label value among neighbors and make a connection between the other labels, that will allow us to build equivalent classes of labels. Once the initial labeling and equivalence recording is completed, the second pass merely replaces each pixel label with its equivalent representative element.

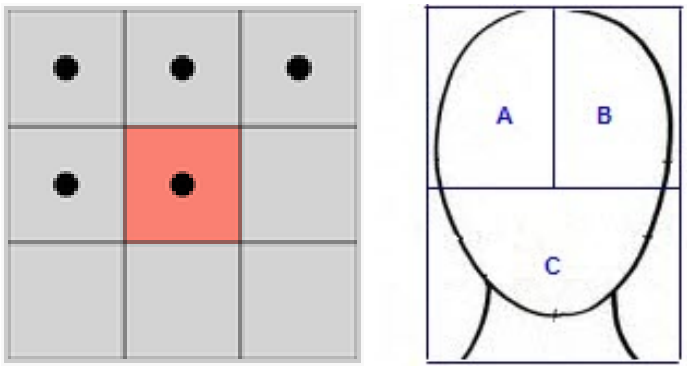

Figure 3.1 Demonstrating the concept of 8-connectivity (left figure). Shows three searching area A, B and C. These areas are used to find face features such as eyes and mouth (right figure).

3.3 Finding eyes and mouth

When the face region was found in the previous two steps, now eyes and mouth can be found. Height and width can be determined using the information about face area. A rectangle around the face region will be drawn as it is shown on Figure 3.1 using those two parameters. As it is known there are proportions between faces features such as position of nose and eyes. If only face without neck was determined and bigger side of rectangle (height in our case) was divided by three equal parts, one can state that eyes are in the first part and mouth is in the third part. The problem is a skin pixel detector will find not only face, but whole head if, for example, men is bald or color of person's hair is close to the color of skin, or will detect head together with neck. That is why a bigger side of the rectangle, that contains the face region, is divided by two equal parts. Upper part contains eyes and lower part contains mouth. Moreover one can say that the left part of the upper rectangle contains a right eye and the right part - contains a left eye, if divided by two equal rectangles. So, now there are three searching areas A, B, C as stated on Figure 3.1. Original rectangle was divided by three rectangles to reduce computational time. Of course, there is a possibility that an eye will not appear in one of the searching areas if head is tilted. The extra rectangle was added to the areas A and B. For example, if the area A is considered, a rectangle with the width of 10 pixels was added to the right side of A. In a similar way a rectangle can be added to B area. It turned out that 10 pixels is enough the second eye not to appear in one area and the first eye not to go out of the current area. Mentioned above is valid for up to 45 degree head tilting.

A procedure used to find eyes will be described below. Searching area A and the right eye are used to demonstrate the procedure. The same is true for the left eye. Many researchers utilize template matching to find an eye. For example in [7] they used stochastic optimization algorithm and template matching to find an eye. On Figure 3.2 one can find template images authors used in their algorithm. Our idea is to use a square with the length of side a instead of template and in every pixel belonging to this square measure a gradient. Illustration of this idea one can find on Figure 3.2. Position with largest gradient which points outwards of square is the position of an eye. Assume that the eyes are found. Using the y-coordinates of the eyes, head tilting can be determined, while the x-coordinates of the eyes help us determine a width of the mouth. We build a straight line that goes through two centers of the eyes and then shift this line in C area. All pixels in C area are scanned using mentioned line to find the mouth. Moreover we added a second line which is lifted up by the distance between the nose and mouth to the previous line. Second line is used to find nostrils. One can find formulae describing two lines below

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \quad (1)$$

$$y = y_1 + d + \frac{y_2 - y_1}{x_2 - x_1}(x - x_1) \quad (2)$$

where (x_1, y_1) and (x_2, y_2) are the coordinates of the starting and ending points, d – distance between the mouth and nose. Formula (2) corresponds to a line used to find nostrils, x varies in interval $x \in [\frac{x_2+x_1}{2} - r, \frac{x_2+x_1}{2} - \frac{r}{2}] \cup [\frac{x_2+x_1}{2} + \frac{r}{2}, \frac{x_2+x_1}{2} + r]$, where r is the distance between nostrils. When scanning through all pixels of C area, a gradient is measured in every point of lines (1) and (2). Position with the largest gradient which points downwards correspond to the position of the mouth. Why was the downward direction of the gradient chosen? If one choose the upward direction there is a risk of detecting the chin instead of the mouth, because regions under the chin are darker in most of cases.

When the eyes and mouth have been found, one can draw a square around the face. It is known that half of the distance between mouth and eyes give us approximate y-coordinate of the center of the face. Half of the horizontal distance between eyes give us approximate x-coordinate of the center of the face. Using knowledge about the center of the face and width of the face region, one can draw a square around the face with the length of side as

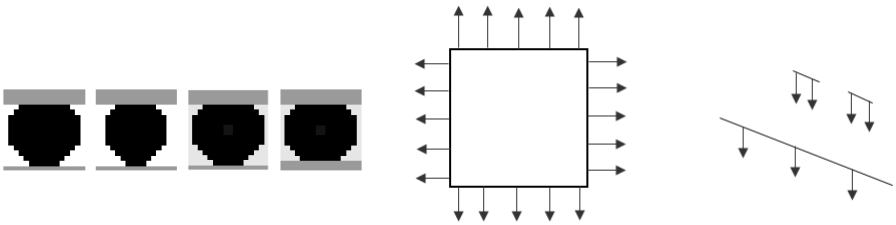

Figure 3.2 Template images of the iris used in [7](left figure). Gradient method used for detecting an eye and mouth. Middle picture – gradient method for an eye detection, right picture – gradient method for detection of a mouth and nostrils.

width of the face region.

During the discussion of the algorithm three parameters used to find the eyes and mouth were mentioned, namely, length of the side of the eye square a, distance between the mouth and nose d and distance between nostrils r. One can expect that these parameters depend and vary on different scales of the face. To deal with this problem the following idea was proposed. Parameters were adjusted for the one particular distance between face and webcam. As the measure of the distance between face and webcam one can exploit fraction of the skin pixel in the whole image. Adjusted parameters and fraction of the skin pixels are etalon values now. For an arbitrary distance we count fraction of the skin pixels, divide obtained number by the etalon fraction of the skin pixels, obtained coefficient is multiplied by the etalon values of the parameters. New values of the parameters can be used to find the eyes and mouth.

Figure 4.1 Original image. Photo by author.

Algorithm

1. Using skin detector retrieve skin pixels.
2. Make image binary with the threshold $T = 50$ and using connected components algorithm find connected areas of foreground pixels.
3. Find the largest connected area of pixels and discard other areas.
4. Find height and width of face area, build rectangle around it and partition it in three searching areas A, B and C.
5. In area A and B for all pixel compute gradient of square with direction outwards as described above and determine position of right and left eyes.
6. In area C for all pixels compute gradient of two lines as described above to find mouth and nares.
7. Determine center of face and draw square around face.

4 Results

In this section some results are presented. Face was detected on still images, in future there is an intention to implement this algorithm in video stream. On Figure 4.1 - 4.3 one can find the results of the skin pixel extraction and connected components. Original image is on the first photo. Second photo shows us the result of the skin pixel extraction and the last one shows the result of the connected components. On Figure 4.4 - 4.11 the results of the face detection are shown.

Figure 4.2 Result of pixel extraction.

Figure 4.3 Result of connected components algorithm.

Figure 4.4 Original image.

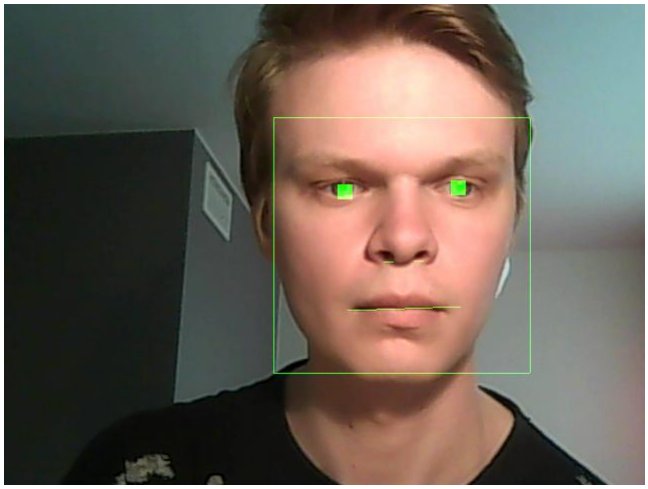

Figure 4.5 Result of face detection. Parameters $a = 15$ pixels, $d = 47$ pixels and $r = 20$ pixels.

Figure 4.6 Original image.

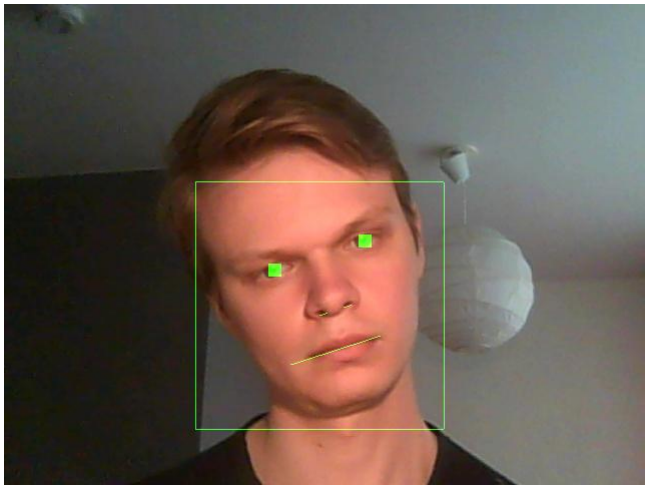

Figure 4.7 Result of face detection. Parameters $a = 10$ pixels, $d = 35$ pixels and $r = 14$ pixels.

Figure 4.8 Original image.

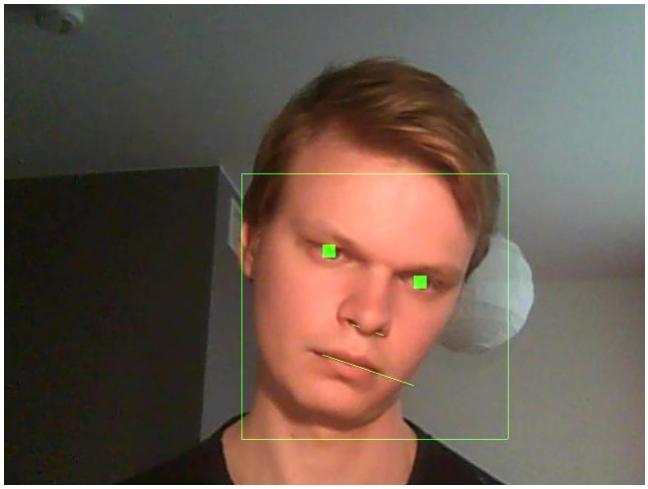

Figure 4.9 Result of face detection. Parameters $a = 10$ pixels, $d = 35$ pixels and $r = 14$ pixels.

Figure 4.10 Original image.

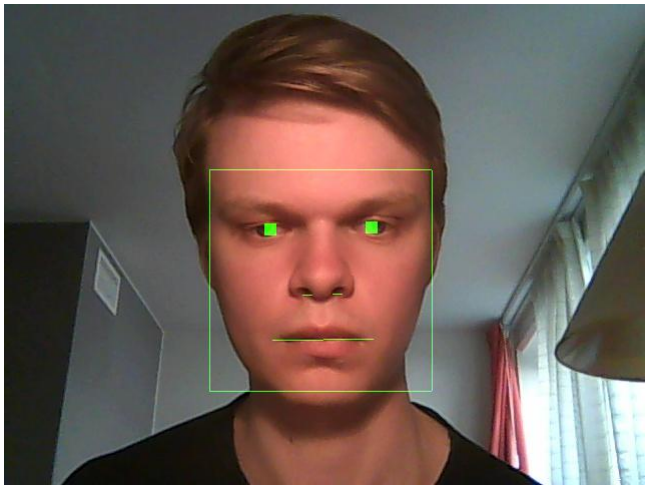

Figure 4.11 Result of face detection. Parameters $a = 13$ pixels, $d = 44$ pixels and $r = 18$ pixels.

5 Conclusion and future work

The conclusion will be given below. First, we talked about major directions in the face detection area. The aim was to show what a great work was done during last decades. However, it was mentioned that there are no perfect algorithm so far, so, the work continues. Then we presented our face detector algorithm. What is the performance of it? The performance is around 93 % of correctly identified faces on still image, 24 in our case. Of course, one should have more larger data base to give more reliable assessment. We do not claim that the algorithm is good, because it should be tested in video stream, and only after this we can talk about how good it is.

As the future work one could do the following. One can try to improve color-based model, try different color spaces or different approaches in building color-based model. There are some works that can help us with this. For tracking faces, one can add to the existing algorithm contextual awaring algorithm. This is a recent approach and seems it gives good results. There is a price for good results. And this is a speed. One should check if it is acceptable to use our algorithm together with contextual awaring algorithm.

References

[1] Cha Zhang, Zhengyou Zhang *A Survey of Recent Advances in Face Detection.* Technical Report, Microsoft Research, 2010.

[2] Paul Viola, Michael Jones *Robust Real-time Object Detection.* Second International Workshop on Statistical and Computational Vision, Vancouver, 2001.

[3] Paul Viola, Michael Jones *Robust Real-Time Face Detection.* International Journal of Computer Vision 57(2), 137-154, 2001.

[4] Liang Wang, Tieniu Tan, Weiming Hu *Face Tracking Using Motion-Guided Dynamic Template Matching.* The 5th Asian Conference on Computer Vision, Melbourne, 2002.

[5] Ming Yang, Ying Wu, Gang Hua *Context-aware Visual Tracking.* IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 0, No. 0, 2008.

[6] Carolina Galleguillos, Serge Belongie *Context based object categorization: A critical survey.* Computer Vision and Image Understanding, 2010.

[7] Takuya Akashi, Yuji Wakasa, Kanya Tanaka *Using Genetic Algorithm for Eye Detection and Tracking in Video Sequence.* Systemics, Cybernetics and Informatics, Vol. 5, No. 2, 72-78, 2007.