ISSN:2229-6093

Nandita Sethi et al, Int. J. Comp. Tech. Appl., Vol 2 (5), 1432-1438

Robust Face Detection and Tracking Using Pyramidal Lucas Kanade Tracker Algorithm

Nandita Sethi
M.Tech (CSE)
Department of Computer Science and Engineering, Haryana College of Technology and Management, Kaithal, India
nandita.sethi21@yahoo.com

Alankrita Aggarwal
(Senior Assistant Professor)
Department of Computer Science and Engineering, Haryana College of Technology and Management, Kaithal, India
alankrita.agg@gmail.com

Abstract

In this paper, we present a face detection and tracking algorithm in real time camera input environment. The entire face tracking algorithm is divided into two modules. The first module is face detection and second is face tracking. To detect the face in the image, Haar based algorithm is used. On the face image, Shi and Thomasi algorithm is used to extract feature points and Pyramidal Lucas-Kanade algorithm is used to track those detected features. Results on the real time indicate that the proposed algorithm can accurately extract facial features points. The algorithm is applied on the real time camera input and under real time environmental conditions.

Key Words: Face detection, face tracking, optical flow method, Pyramidal lucas kanade algorithm.

1. Introduction

Facial Features tracking is a fundamental problem in computer vision due to its wide range of applications in psychological facial expression analysis and human computer interfaces. Recent advances in face video processing and compression have made face-to face communication be practical in real world applications. And after decades, robust and realistic real time face tracking still poses a big challenge. The difficulty lies in a number of issues including the real time face feature tracking under a variety of imaging conditions (e.g., skin color, pose change, self-occlusion and multiple non-rigid features deformation). In this paper, we concentrate our work on facial feature tracking. To detect the face in the image, we have used a face detector based on the Haar-like features [9]. This face detector is fast and robust to any illumination condition. For feature point extraction, we have used Shi and Tomasi method [4]. In order to track the facial feature points, Pyramidal Lucas-Kanade Feature Tracker algorithm [8] is used. Pyramidal Lucas Kanade algorithm [8] is the powerful optical flow algorithm used in tracking. It tracks starting from highest level of an image pyramid and working down to lower levels. Tracking over image pyramids allows large motions to be caught by local windows.

2. Proposed algorithm

The algorithm used to track the face is given below:

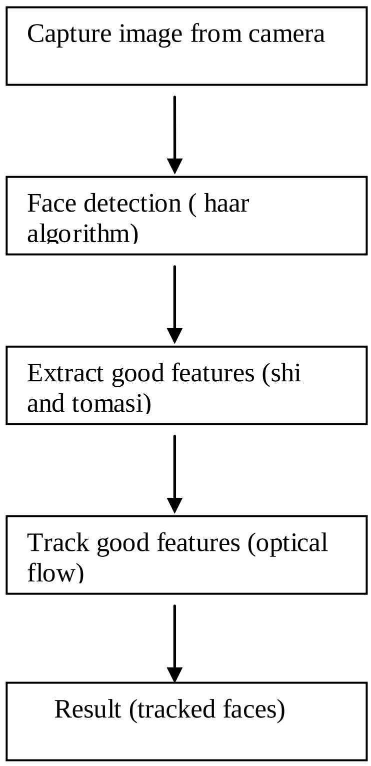


Figure 1. Proposed algorithm to track the face

The following sections will provide the detail of each step. The algorithm is implemented in C language using Opencv Image library.

ISSN:2229-6093

3. Face detection

For face detection, we have used Viola & Jones's face detector based on the Haar-like features [9]. In [9], Paul Viola and Michael Jones, describes a visual object detection framework that is capable of processing images extremely rapidly while achieving high detection rates. There are three key contributions. The first contribution is a new a technique for computing a rich set of image features using the integral image. The second is a learning algorithm, based on AdaBoost, which selects a small number of critical visual features and yields extremely efficient classifiers [5]. The third contribution is a method for combining classifiers in a "cascade" which allows background regions of the image to be quickly discarded while spending more computation on promising object-like regions.

3.1. Feature selection

The main purpose of using features rather than the pixels directly is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. Also the the feature-based system operates much faster than a pixel-based system [9]. The simple features used are reminiscent of Haar basis functions which have been used by Papageorgiou et al. [6]. Examples of the features used can be seen in Figure 2. The features consist of a number of rectangles that are equal in size and horizontally or vertically adjacent. The value of a two-rectangle feature (A and B in Figure 2) is calculated as the difference between the sum of pixels within the two rectangular regions of the feature. In a three-rectangle feature (C in Figure 2) the sum of pixels in the two outside rectangles is subtracted from the sum of the pixels in the centre rectangle. In a four-rectangle feature (D in Figure 2) the feature value is the difference in sum of pixels between diagonal pairs of rectangles.

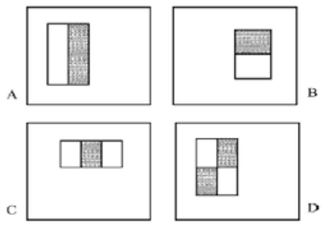


Figure 2. Rectangle features shown relative to the enclosing detection window.

The sum of the pixels which lie within the white rectangles are subtracted from the sum of pixels in the grey rectangles. Two-rectangle features are shown in (A) and (B). Figure (C) shows a three-rectangle feature, and (D) a four-rectangle feature.

3.2. Integral image

Rectangular features can be computed very rapidly using an intermediate representation for the image called the integral image [9]. The integral image at location (x, y) contains the sum of the pixels above and to the left of (x, y), inclusive:

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

where i(x, y) is the original image and ii(x, y) is the integral image .

Using the following pair of recurrences:

$$s(x, y) = s(x,y-1) + i(x, y) \qquad (1)$$

$$ii(x, y) = ii(x-1,y) + s(x, y) \qquad (2)$$

where s(x, y) is the cumulative row sum, s (x, -1) = 0 and ii (-1, y) = 0 the integral image can be computed in one pass over the original image [9].

Using the integral image any rectangular sum can be computed in four array references (Figure 3). Clearly the difference between two rectangular sums can be computed in eight references. Since the two rectangle features defined above involve adjacent rectangular sums they can be computed in six array references, eight in the case of the three-rectangle features, and nine for four-rectangle features.

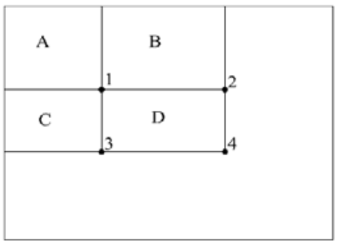


Figure 3. Sum of pixel values within "D".

The Above Figure 3 showing that the sum of the pixels within rectangle D can be computed with four array references. The value of the integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A + B, at location 3 is A + C, and at location 4 is A + B + C + D. The sum within D can be computed as 4 + 1 - (2 + 3).

3.3. Learning classification functions

Given a feature set and a training set of positive and negative sample images, any number of machine learning approaches could be used to learn a classification function. A variant of AdaBoost is used both to select the features and to train the classifier [5]. In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple (sometimes called weak) learning algorithm. Recall that there are over 117,000 rectangle features associated with each image 24×24 sub-window, a number far larger than the number of pixels. Even though each feature can be computed very efficiently, computing the complete set is prohibitively expensive. The main challenge is to find a very small number of these features that can be combined to form an effective classifier. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples.

3.4. Cascade of classifiers

The overall form of the detection process is that of a degenerate decision tree, what we call a "cascade"[2] (see Figure 4). A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates. A positive result from the second classifier triggers a third classifier, and so on. A negative outcome at any point leads to the immediate rejection of the sub-window.

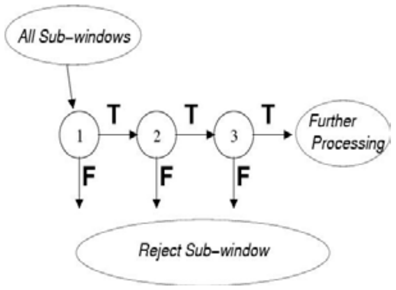


Figure 4. Schematic depiction of a the detection cascade.

A series of classifiers are applied to every sub window. The initial classifier eliminates a large number of negative examples with very little processing. Subsequent layers eliminate additional negatives but require additional computation. After several stages of processing the number of sub windows have been reduced radically. Further processing can take any form such as additional stages of the cascade or an alternative detection system.

4. Facial feature extraction

For facial feature points extraction we have used Shi and Tomasi method [4]. This method is based on the general assumption that the luminance intensity does not change for image acquisition. To select interest points, a neighbourhood *N* of *nxn* pixels is selected around each pixel in the image. The derivatives *Dx* and *Dy* are calculated with a Sobel operator for all pixels in the block *N*. For each pixel the minimum eigenvalue λ is calculated for matrix *A* where

$$A = \begin{bmatrix} \sum D_{x_{i,j}}^2 & \sum D_{x_{i,j}} \sum D_{y_{i,j}} \\ \sum D_{x_{i,j}} \sum D_{y_{i,j}} & \sum D_{y_{i,j}}^2 \end{bmatrix}$$

and Σ is performed over the neighborhood of *N*. The pixels with the highest values of λ are then selected by thresholding.

The next step is rejecting the corners with the minimal Eigen value less than some threshold. Finally, a test is made, all the found corners are distanced enough from one another by getting two strongest features and checking that the distance between the points is satisfactory. If not, the point is rejected. For further details see [4].

5. Motion detection and tracking

In the field of computer vision motion detection has a relevant importance. By using information contained in a stand image, we can obtain a lot of information about what we are observing, but there is no way to automatically infer what is going to happen in the immediate future. On the other hand a sequence of images provide information about movement of depicted objects. There's a plenty of techniques to recognize movement in a sequence, some based on feature and pattern recognition, some other based just on pixels, regardless what they mean for a human being. Examples are Block Matching analysis and Optical Flow estimation methods etc.

5.1. Motion and motion field

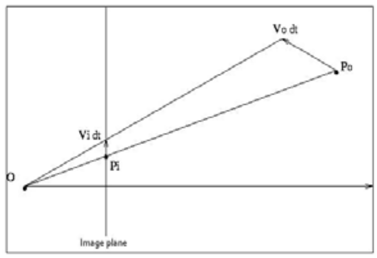


Figure 5. Velocity projection over image surface

When an object is moving in space in front of a camera there's a corresponding movement on the image surface. Give the point P0 and its projection Pi, by knowing its velocity V0, we can find out the vector Vi representing its movement in the image (figure 5). Given a moving rigid body we can build a motion field by computing all the Vi motion vectors. The motion field is a way to map the movement in a 3D space, on a 2D image taken on camera: for this reason, since we loose a dimension, we cannot exactly compute motion field, but just approximate it.

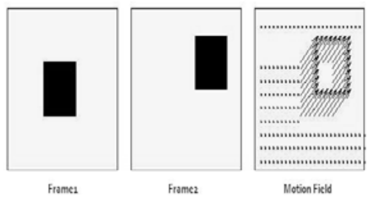


Figure 6. Example of motion flow

ISSN:2229-6093

5.2. Optical flow

Optical flow is a phenomenon we deal with every day. It is the apparently visual motion of standing objects as we move through the world. When we are moving the entire world around us seems to move in the opposite way: the faster we move, the faster it does. This motion can also tell us how close we are to the different object we see. The closer they are, the faster their movement will appear. There is also a relationship between the magnitude of their motion and the angle between the direction of our movement and their relative position to us: if we are moving toward an object, it will appear to stand still, but it will become larger as we get closer (this phenomena is also called FOE, focus of expansion); rather, if the object we are looking at is beside us, it will appear moving faster. In computer vision there are a lot of optical flow estimation techniques applied in fields as behaviour recognition or video surveillance; though they are "blinder" than pattern recognition based methods, there are fast enough implementations that allow us to build soft real time applications.

6. Optical flow tracking

Optical flow is the apparent motion of image brightness. Let I(x,y,t) be the image brightness that changes in time to provide an image sequence. Two main assumptions can be made [12]:
1. Brightness $I(x,y,\ t)$ smoothly depends on coordinates x, y in greater part of the image.
2. Brightness of every point of a moving or static object does not change in time.
Let some object in the image, or some point of an object, move and after time dt the object displacement is (dx,dy). Using taylor series for brightness I(x,y,t), gives the following:

$$I(x+dx,y+dy,t+dt)=I(x,y,t)+\frac{\delta I}{\delta x}dx+\frac{\delta I}{\delta y}dy+\frac{\delta I}{\delta t}dt+\ldots$$

where “…” are higher order terms.
Then, according to assumption 2,

$$I(x+dx,y+dy,t+dt)=I(x,y,t)$$

and

$$\frac{\delta I}{\delta x}dx+\frac{\delta I}{\delta y}dy+\frac{\delta I}{\delta t}dt+\ldots=0$$

Divide by dt and denote

$$I_x u + I_y v + I_t = 0$$

The above equation is called optical flow constraint equation where $u=\frac{dx}{dt}$ $\quad v=\frac{dy}{dt}$

are components of optical flow field in x and y coordinates and the derivatives of I are denoted by subscripts.
The optical flow constraint equation can be rewritten as

$$I_x u + I_y v = -I_t$$

We have one equation but two variables, that means we need some other constraints. For this reason optical flow sometimes doesn't correspond to the motion field. This is the so called aperture problem and we can understand it better by watching at figure. 7, since the cylinder is rotating, if we consider just the black bars, it would be impossible to determine whether they're moving horizontally (as they do), or vertically, as detected by optical flow. It is impossible to determine the real movement unless the end of the bars become visible.

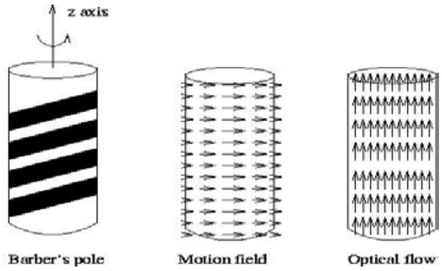


Figure 7. An example when optical flow is different from motion field

6.1. Aperture problem

One problem we do have to worry about, however, is that we are only able to measure the component of optical flow that is in the direction of the intensity gradient. We are unable to measure the component tangential to the intensity gradient. This problem is illustrated in figure 8, and further developed below.

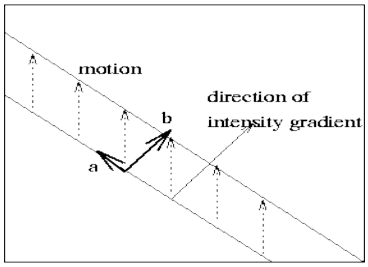


Figure 8. The aperture problem

Since,

$$I_x u + I_y v + I_t = 0$$

This optical flow constraint equation (which expresses a constraint on the components u and v of the optical flow) can be rewritten as

$$(I_x, I_y)\cdot(u, v) = -I_t.$$

Thus, the component of the image velocity in the direction of the image intensity gradient at the image of a scene point is

$$(u, v) = \frac{-I_t}{\sqrt{I_x^2 + I_y^2}}.$$

We cannot, however, determine the component of the optical flow at right angles to this direction. This ambiguity is known as the aperture problem.

6.2. Optical flow methods

There are different methods that add some constraint to the problem, in order to estimate the optical flow. Some of them are:

1. Block based methods. minimizing sum of squared differences or sum of absolute differences, or maximizing normalized cross-correlation

2. Discrete optimization methods. the whole space is quantized, and every pixel is labelled, such that the corresponding deformation minimizes the distance between the source and the target image. The optimal solution is often computed through min-cut max-flow algorithms or linear programming.

3. Differential methods. The differential methods of optical flow estimation, based on partial spatial and temporal derivatives of the image signal, as following:

- **Lucas kanade method.** dividing image into patches and computing a single optical flow on each of them
- **Horn schunck method.** optimizing a functional based on residuals from the brightness constancy constraint, and a particular regularization term expressing the expected smoothness of the flow field.
- **Buxton buxton method.** based on a model recovered from the motion of edges in image sequences.
- **General variational methods.** range of the extensions or modifications of Horn-schunck, using other data terms and other smoothness terms.

Here we are going to explain lucas kanade method.

6.3. Lucas kanade method

The Lucas Kanade (LK) algorithm [1], as originally proposed in 1981, was an attempt to produce dense results. Yet because the method is easily applied to a subset of the points in the input image, it has become an important sparse technique. The LK algorithm can be applied in a sparse context because it relies only on local information that is derived from some small window surrounding each of the points of interest. The disadvantage of using small local windows in Lucas-Kanade is that large motions can move points outside of the local window and thus become impossible for the algorithm to find. This problem led to development of the "pyramidal" Lucas Kanade algorithm [8], which tracks starting from highest level of an image pyramid (lowest detail) and working down to lower levels (finer detail). Tracking over image pyramids allows large motions to be caught by local windows. The basic idea of the LK algorithm rests on three assumptions:

1. Brightness constancy. A pixel of an object in an image does not change in appearance as it (possibly) moves from frame to frame. For grayscale image, this means we assume that the brightness of a pixel does not change as is tracked from frame to frame.

2. Temporal persistence or small movements. The image motion of a surface patch changes slowly in time. In practice, this means the temporal increments are fast enough relative to the scale of motion in the image that the object does not move much from frame to frame.

3. Spatial coherence. Neighboring points in a scene belong to the same surface, have similar motion, and project to nearby points on the image plane.

6.4. Pyramidal lucas-kanade feature tracker

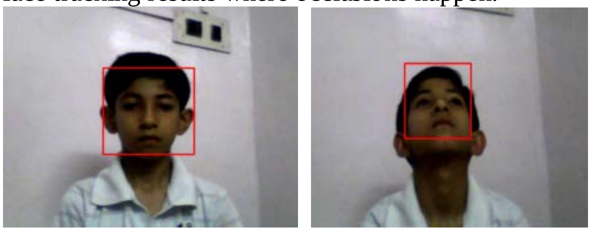
Pyramidal lucas kanade algorithm is the powerful optical flow algorithm used in feature tracking. Consider an image point u = (ux, uy) on the first image I, the goal of feature tracking is to find the location v = u + d in next image J such as I(u) and J(v) are "similar". Displacement vector d is the image velocity at x which also known as optical flow at x [8]. Because of the aperture problem, it is essential to define the notion of similarity in a 2D neighborhood sense. Let ωx and ωy are two integers. then d the vector that minimizes the residual function defined as follows:

$$\epsilon(\mathbf{d}) = \epsilon(d_x, d_y) = \sum_{x=u_x-\omega_x}^{u_x+\omega_x} \sum_{y=u_y-\omega_y}^{u_y+\omega_y} (I(x,y) - J(x+d_x, y+d_y))^2$$

Observe that following that defintion, the similarity function is measured on a image neighborhood of size ($2\omega x$ + 1) x ($2\omega y$ +1). This neighborhood will be also called integration window. Typical values for ωx and ωy are 2,3,4,5,6,7 pixels.

7. Implementation and results

The algorithm is implemented in C language using Opencv Image Library in Visual Studio 6 editor. The operating system is windows XP and machine specification are Pentium-4, 2 GB RAM. The frame per second after processing is 20. The processing time of each frame is 50 milliseconds. The tracking results shown below are captured under occlusions, pose variations, scales, and illumination changes. Figure 9 illustrates face tracking results with different poses and under illumination changes. Figure 10 illustrates face tracking results where occlusions happen.



ISSN:2229-6093
Nandita Sethi et al, Int. J. Comp. Tech. Appl., Vol 2 (5), 1432-1438

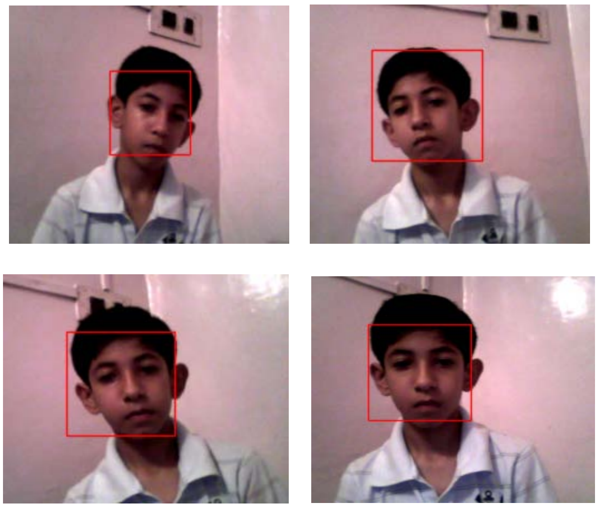


Figure 9. Face tracking results with different poses and under illumination changes

Figure 10. Face tracking results with occlusions

8. Conclusion and future work

In this paper, we have presented a face tracking algorithm in real time camera input environment. To detect the face in the image, we have used a face detector based on the Haar-like features. This face detector is fast and robust to any illumination condition. For feature points extraction, we have used the algorithm of Shi and Tomasi to extract feature points. This method gives good results. To track the facial feature points, Pyramidal Lucas-Kanade Feature Tracker KLT algorithm is used. Using detected points with the algorithm of Shi and Tomasi, we have got good results in video sequence and in real time acquisition. The obtained results indicate that the proposed algorithm can accurately extract facial features points. The future work will include extracting feature points with some conditions to limit the number of feature points in bounding box and choose only the points which describe well the shape of the facial feature. This work will be used for real time facial expression recognition application. Further the work can be extended by detecting faces at different inclinations or slopes as Haar classifier has its own limitations.

9. References

[1] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision,” Proceedings of the DARPA imaging understanding workshop, pp. 121–130, 1981.
[2] J. Quinlan, “Induction of decision trees,” Machine Learning, Journal, vol. 1, no. 1, pp. 81-106, 1986.
[3] M. Lades, J.C. Vorbrüggen, J. Buhmann , J. Lange, C. Malsburg, R. Würtz., and W. Konen, “Distortion invariant object recognition in the dynamic link architecture,” IEEE Trans. Computer, vol. 42, no. 3, pp. 300-310, 1993.
[4] J. Shi and C. Tomasi, “Good Features to Track,” IEEE Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.
[5] Y. Freund and R.E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” in European Conference on Computational Learning Theory, pp. 23–37, 1995.
[6] C. Papageorgiou, M. Oren, and T. Poggio, “A general framework for object detection,” In International Conference on Computer Vision, vol. 21, pp. 555–562, 1998.
[7] K. Sung and T. Poggio, “Example-based learning for view-based face detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, volume 20, pp. 39–51, 1998.
[8] J.Y. Bouguet, “Pyramidal Implementation of the Lucas Kanade Feature Tracker Description of the algorithm,” Intel Corporation, Microprocessor Research Labs, pp. 1-9, 2000.
[9] P. Viola and M. Jones, “Robust Real-time Object Detection,” 2nd international workshop on statistical and computational theories of vision - modeling, learning, computing, and sampling, pp. 1-25, 2001.
[10] E. Hjelmas and B.K. Low, “Face detection: a survey,” Computer vision and image understanding, vol. 83, pp. 236-274, 2001.
[11] M. H Yang, “Detecting faces images, A survey,” IEEE Transations on Pattern Analysis and Machine Inteligence vol. 24, no. 1, pp. 34–58, 2002.
[12] Z. Vamossy, A. Toth, and P. Hirschberg, “PAL Based Localization Using Pyramidal Lucas-Kanade Feature Tracker”, In 2nd Serbian-Hungarian Joint Symposium on Intelligent Systems, Subotica, Serbia and Montenegro, pp. 223-231, 2004.
[13] K.S. Huang, and M.T. Trivedi , “Robust real-time detection, tracking, and pose estimation of faces in video streams,” Proceedings of the 17th International Conference on Pattern Recognition, Vol. 3 pp. 965 - 968 , 2004.
[14] L. Stan and Z. Zhang, “FloatBoost learning and statistical face detection”, IEEE Trans. On Pattern Analysis and Machine Intelligence. vol. 26, no. 9, pp. 1112-1123, 2004.
[15] P. Menezes, J. C. Barreto, and J. Dias, “face tracking based on haar like features and eigen faces,” 5th IFAC Symposium on Intelligent Autonomous Vehicles, Lisbon, Portugal, pp. 1-6, July 5-7, 2004.
[16] J. Barreto, P. Menezes and J. Dias, “Human-robot interaction based on Haar-like features and eigenfaces,” In International Conference on Robotics and Automation, Vol. 2, pp. 1888 – 1893, 2004.
[17] F. Shih and C. Chuang, “Automatic Extraction of Head and Face Boundaries and Facial Feature”, Information Sciences, pp. 117-130, 2004.
[18] J. Yu, B.Bhanu., Y. Xu., A. Roy-Chowdhury, “Super-resolved facial texture under changing pose and

ISSN:2229-6093

illumination", In: Intl. Conf. on Image Processing, pp. 553-556, 2007.
[19] G. Bradski, and A. Kaehler, Learning OpenCV: Computer Vision with the OpenCV Library. O'Reilly Media, Inc, 2008.
[20] J. Sung, T. Kanade, and D. Kim, "Pose robust face tracking by combining active appearance models and cylinder head models", International Journal of Computer Vision, 80, pp. 260-274, 2008.