International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:4, 2009

Robust Face Recognition Using Eigen Faces and Karhunen-Loeve Algorithm

Parvinder S. Sandhu, Iqbaldeep Kaur, Amit Verma, Prateek Gupta

***Abstract*—**The current research paper is an implementation of Eigen Faces and Karhunen-Loeve Algorithm for face recognition. The designed program works in a manner where a unique identification number is given to each face under trial. These faces are kept in a database from where any particular face can be matched and found out of the available test faces. The Karhunen –Loeve Algorithm has been implemented to find out the appropriate right face (with same features) with respect to given input image as test data image having unique identification number. The procedure involves usage of Eigen faces for the recognition of faces.

***Keywords*—**Eigen Faces, Karhunen-Loeve Algorithm, Face Recognition.

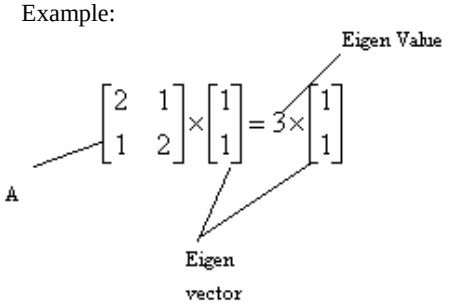
I. Introduction

FACE recognition system based on Eigen Faces Method and Karhunen-Loeve algorithm. Eigen values and Eigenvectors are [1] defined as If v is a nonzero vector and λ is a number such that

$$Av = \lambda v \tag{1}$$

Then, v (Equation 1) is said to be an eigenvector of A with eigen value λ.

Example:



Eigenvectors of Covariance Matrix The eigenvectors v_i of $A.A$ are

Parvinder S. Sandhu is working as Professor with the Rayat & Bahra Institute Of Engineering & Bio-Technology, Mohali- India.

Amit Verma and Iqbaldeep Kaur are Assistant Professor with the Rayat & Bahra Institute Of Engineering & Bio-Technology, Mohali, India, .E-Mail:eramitverma@rediffmail.com, er_iqbaldeel@yahoo.com.

Prateek Gupta is Lecturer (ECE Deptt.) at the Rayat & Bahra Institute Of Engineering & Bio-Technology, Mohali.

$$\underset{v_1}{\begin{bmatrix}0.3923\\0.5060\\0.7681\end{bmatrix}} \quad \underset{v_2}{\begin{bmatrix}0.9087\\-0.0835\\-0.4091\end{bmatrix}} \quad \underset{v_3}{\begin{bmatrix}0.1429\\-0.8585\\0.4926\end{bmatrix}}$$

Consider the eigenvectors [2] v_i of ATA such that

$$A_T A v_i = \mu_i v_i \tag{2}$$

Pre multiplying both sides by A, we have

$$AA_T (Av_i) = (\mu_i Av_i) \tag{3}$$

Eigenvectors of Covariance Matrix (Equation 2 and 3)

$$\mu_i = Av_i \tag{4}$$

$$\underset{u_1}{\begin{bmatrix}-0.2621\\-0.2621\\-0.6527\\-0.1137\\-0.5589\\0.6015\\0.4895\\0\\0.6379\end{bmatrix}} \underset{u_2}{\begin{bmatrix}0.3256\\0.3256\\-3.3773\\0.9922\\-1.0076\\-0.5080\\2.3100\\0\\-1.6434\end{bmatrix}} \underset{u_3}{\begin{bmatrix}-1.3511\\-1.3511\\2.2735\\1.0014\\-6.0561\\-5.4206\\0.6517\\0\\1.7008\end{bmatrix}}$$

$$u = \underset{u_1}{\begin{bmatrix}-0.2621\\-0.2621\\-0.6527\\-0.1137\\-0.5589\\0.6015\\0.4895\\0\\0.6379\end{bmatrix}} \underset{u_2}{\begin{bmatrix}0.3256\\0.3256\\-3.3773\\0.9922\\-1.0076\\-0.5080\\2.3100\\0\\-1.6434\end{bmatrix}} \underset{u_3}{\begin{bmatrix}-1.3511\\-1.3511\\2.2735\\1.0014\\-6.0561\\-5.4206\\0.6517\\0\\1.7008\end{bmatrix}}$$

Here, u_i resemble facial images which look ghostly, hence called Eigenfaces A face image can be projected into this face space by $pk = UT(xk – m)$ where k=1,…,m

The test image x is projected into the face space to obtain a vector p (Equation 5)

$$p = UT(x - m) \quad (5)$$

The distance of p (Equation 6) to each face class is defined by

$$\epsilon k2 = \| p - pk \|2; k = 1, \ldots m \quad (6)$$

A distance threshold [2] Θ_c, (Equation 7) is half the largest distance between any two face images:

$$\theta_c = 1/2 \max_{j,k} \{ \| p_j - p_k \| \}; j, k = 1, \ldots m \quad (7)$$

Find the distance ϵ (Equation 8) between the original image x and its reconstructed image from the eigen face space, *xf*,

$$\epsilon 2 = \| x - xf \| 2, \text{where} \quad xf = U * x + m \quad (8)$$

The Recognition [2] process begins with the conditions,

$$\text{if } \epsilon ns \geq \Theta_c$$

Then, input image is not a face image;

$$\text{if } \epsilon < \theta_c \text{ AND } \epsilon k \geq \theta_c \text{ for all } k \quad (9)$$

Then, input image contains an unknown face;

$$\text{If } \epsilon < \theta_c \text{ AND } \epsilon k^* = \min k\{ \epsilon k \} < \theta_c \quad (10)$$

then input image contains the face of individual k*

II. Eigen Faces

The Eigen face method for human face recognition is remarkably clean and simple. The basic concept behind the Eigen face method is information reduction. When one evaluates even a small image, there is an incredible amount of information present. From all the possible things that could be represented in a given image, pictures of things that look like faces clearly represent a small portion of this image space. Because of this, we seek a method to break down pictures that will be better equipped to represent face images rather than images in general. To do this, one should generate 'base-faces' and then represent any image being analyzed by the system as a linear combination of these base faces. Once the base faces have been chosen we have essentially reduced the complexity of the problem from one of image analysis to a standard classification problem. Each face that we wish to classify can be projected into face-space and then analyzed as a vector. A k-nearest-neighbor approach, a neural network or even a simply Euclidian distance measure can be used for classification. The technique can be broken down into the Generate the Eigen faces, Project training data into face-space to be used with a predetermined classification method and evaluation of a projected test element by projecting it into face space and comparing to training data.

III. Generation Eigen Faces

Before any work can be done to generate the Eigen faces, sample faces are needed. These images will be used as examples of what an image in face-space looks like. These images do not necessarily need to be images of the people the system will later be used to identify (though it can help); however the image should represent variations one would expect to see in the data on which the system is expected to be used, such as head tilt/angle, a variety of shading conditions etc. Ideally these images should contain pictures of faces at close to the same scale, although this can be accomplished through preprocessing if necessary. It is required that all of the images being used in the system, both sample and test images, be of the same size. The resulting Eigen faces will also be of this same size once they have been calculated.

It should be noted that it is assumed that all images being dealt with are grayscale images, with pixel intensity values ranging from 0 to 255. Suppose, there are *K* images in our data set. Each sample image will be referred to as X_i where n indicates that we are dealing with i^{th} sample image ($1 <= i <= K$). Each X_i is a column vector. Generally images are thought of as pixels, each having (*x*, *y*) coordinates with (0, 0) being at the upper left corner (or one could think of an image as a matrix with y rows and *x* columns). Converting this to a column form is a matter of convenience, it can be done in either column or row major form, so long as it is done consistently for all sample images it will not affect the outcome. The size of the resulting X_i column vector will depend on the size of the sample images. If the sample images are *x* pixels across and y pixels tall, the column vector will be of size *(x*y)* x 1. These original image sizes must be remembered if one wishes to view the resulting Eigen faces, or projections of test images into face-space. This is to allow a normal image to be constructed from a column vector of image pixels. Let X^- be the mean of all X_i ($1 <= i <= K$). This is the step to calculate an average face of the database. If one were to reinterpret the vector as a normal image, it would appear as one might expect, as shown in Fig. 1.

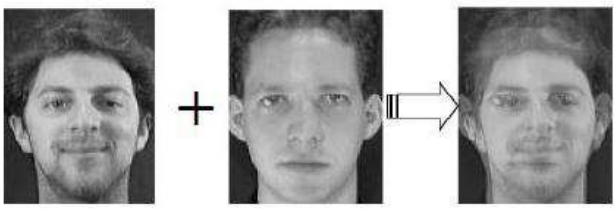


Fig. 1 Addition of Faces

The next step is to calculate difference faces U_i such that $U_i = X_i - X^-$ (where X^- is mean) and form a matrix *U*, such that $U = [U_1\ U_2 \ldots U_K]$. Our goal now is to generate the eigenfaces which is done by calculating the eigenvectors of the covariance matrix UU^T .This cannot be done directly as the size of UU^T is *(x*y)*(x*y)* which is very large. Clearly, doing these calculations on a resulting matrix of this size is going to be taxing on all but the most specialized, advance hardware. To avoid this problem, a trick from linear algebra is applied. The eigenvectors of the UU^T matrix can actually be found by

considering linear combinations of the eigenvectors of the U^TU matrix. This is extremely usefully when one realizes that the size of the U^TU matrix is K x K. For practically all real world situations $K << (x*y)$. The eigenvectors w_j of this matrix can be readily found through the following formula:

$$W_j = \frac{\sum_{l=1}^{k} U_l E_{l_j}}{\sqrt{\lambda_j}} \tag{11}$$

Where, E_{lj} is the *lth* value of the *jth* eigenvector of U^TU and λ_j is the corresponding Eigen value of w_j and E_j. The linear algebra part of this trick is given below: Let the eigenvectors of U^TU be E_j (1 <= j <= K) and the corresponding Eigen values be λ_j . Hence, it can be written:

$$U^tU\ E_j = \lambda_j E_j \tag{12}$$

Multiplying both the sides by U,

$$U*U^TU\ E_i = \lambda_j\ UE_j \tag{13}$$

Thus, w_j = UE_j is the *jth* eigenvector of UU^T with corresponding eigenvalue λ_j. The fact that the eigenvalues for the UU^T and U^TU are the same (though if we were going to calculate all of the Eigen values of the UU^T matrix, we could get more values, the eigenvectors of the U^TU only represent the most important subset of the Eigen values of the UU^T matrix).

IV. Karhunen–Loeve (KL) transform

The Karhunen–Loeve (KL) transform is a preferred method for approximating a set of vectors or images by a low dimensional subspace. The method provides the optimal subspace, spanned by the KL basis, which minimizes the MSE between the given set of vectors and their projections on the subspace. The KL transform has found many applications in traditional fields such as statistics and communication. In computer vision, it was used for a variety of tasks such as face recognition, object recognition, motion estimation, visual learning, and object tracking. Typical computer vision applications calculate the KL basis of hundreds or thousands images, each of size (Width Height) in the range of 10 K to 1 M. The basis is partial and typically includes only a few dozens vectors or less. Calculating the KL basis for images of size requires roughly operations. In many applications, this large computational demands may be prohibitive. In fact, for such applications, even the memory requirement may exceed the resources of common computers. Several attempts were already made to reduce the computational effort by using efficient image coding. The Karhunen-Loeve expansion is a powerful spectral technique for the analysis and synthesis of dynamical systems. It consists in decomposing a spatial correlation matrix, which can be obtained through numerical or physical experiments. The decomposition produces orthogonal Eigen functions or proper orthogonal modes, and Eigen values that provide a measurement of how much energy is contained in each mode.

It makes a principal component analysis that is a mathematical way of determining that linear transformation of a sample of points in L-dimensional space which exhibits the properties of the sample most clearly along the coordinate axes. Along the new axes, the sample variance are extremes and uncorrelated, see Fig. 2. Using a cutoff on the spread along each axis, a sample may be reduced in its dimensionality. This way it can be used to transform independent coordinates into significant and independent ones.

The KLC transform is a reversible linear transform that exploit the statistical property of a vector representation. The basic function of KLC transform are orthogonal Eigen vector of the covariance matrix of a data set. A KL transform optimally decor relates the input data. After a KL transform most of the energy of the transform coefficient is concentrated within the first few components. This is compact energy property of KL transform.(As from Fig. 2)

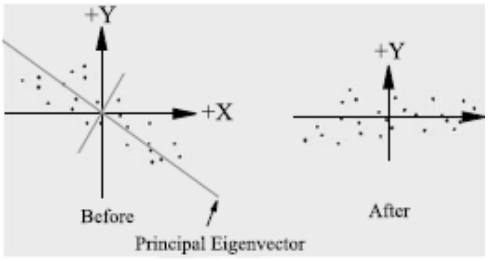


Fig. 2 Larhunen-Loeve (KL) Map

V. Karhunen–Loeve (KL) implementation

The seismic traces *xi(t)* correspond to the rows of the named data matrix *xn*m* , *n* is the number of traces in the gather and *m* the number of gathers. The zero-lag covariance matrix £ **nn* is

$$£*nm \quad xn*m \quad x'n*m \tag{14}$$

This expression can be decomposed as

$$£*nm \quad Vn*p \quad An*mV'n*p \tag{15}$$

In equation 2 the columns of the matrix *Vn*p* are the eigenvectors £*n*n* and *An*n* is a diagonal matrix with the eigen values on the diagonal. These eigen values are ordered in descending sequence along the principal diagonal of the matrix *An*n*, *V*npt* is the corresponding transpose matrix. The principal component of the data can be written by:

$$Mp*m \quad Xn*mV'n*p \tag{16}$$

If we form the matrix *M'p*m* by selecting the upper *m* rows of the matrix *Mp*m*, *m* is related with the order of the filter or principal row number, placing zeros in the remaining *n-m* rows, *n* is related with the Cut of the Filter, the Ground roll is assumed to be represented by the matrix product between the eigenvectors matrix *Vn*p* and the matrix *M'p*m*, that include

the principal components in agreement with the following expression:

$$X'p*m \quad Vn*mM'p*m \tag{17}$$

Finally, the Ground roll $X'n*m$ is subtracted from the data matrix $Xn*m$, obtaining the gather without Ground roll

$$X_{On*m} \; X_{n*m} \; X_{n*m} \tag{18}$$

Where, X_o is a $n*m$ matrix that represents the filtered seismic gather. As mentioned before $Xn*m$ is the input data matrix and $X'n*m$ is a matrix that represents the Ground roll extracted from the input data.

VI. Present Implementation

In the system (designed) functions by projecting face images onto a feature space that spans the significant variations among known face images. The significant features are known as "eigenfaces"[4] because they are the eigenvectors (principal components) of the set of faces.

Face images must be collected into sets: every set (called "class") should include a number of images for each person (entity), with some variations in expression and in the lighting. When a new input image is read and added to the training database, the number of class is required. Otherwise, a new input image can be processed and confronted with all classes present in database. We choose a number of eigenvectors M' equal to the number of classes. Before start image processing first select input image. This image can be successively added to database (training) or, if a database is already present, matched with known faces.

First, select an input image clicking on "Select image". Then add this image to database .If one choose to add image to database, a positive integer (face ID) is required. This positive integer is a progressive number which identifies a person (entity) (each person (entity) corresponds to a class). For example: run the GUI delete previous database add "image1.jpg" to database the ID has to be 1 since image1 is the first entity which is adding to database add "image1.jpg" to database the ID has to be 1 since you have already added a image1 image to database as first entity add "image2" to database the ID has to be 2 since image2 is the second one entity adding to database. Add "image3" to database the ID has to be 3 since image3 is the third one entity you are adding to database add "image2" to database the ID has to be 2 once again since we have already added image2 to database and so on!

The recognition[16] gives as results the ID of nearest entity present in database. For example if you select image "image2" the ID given SHOULD be 2 "it should be" because errors are possible.

VII. Main Functions and Working

The main functions [6]-[8] of the algorithm and functional description (As from Table I) are as given below.

TABLE I. Function Description

Sr.No	Function	Discription
1.	Select image:	Read the input image
2.	Add selected image to database	The input image is added to database and will be used for training
3.	Database Info	Show information's about the images present in database. Images must have the same size. If this is not true you have to resize them.
4.	Face Recognition [9-10]	Face matching. The selected input image is processed
5.	Delete Database	Remove database from the current directory
6.	Visualization tool	Show information's
7.	Exit	Quit the procedure

First, the recognition [3] and[11-13] system is run in the GUI (as shown in the Fig. 3).Then input image is selected by pressing the select image icon in the GUI add the image in the database (Fig. 4) an given a unique number called ID number (positive).

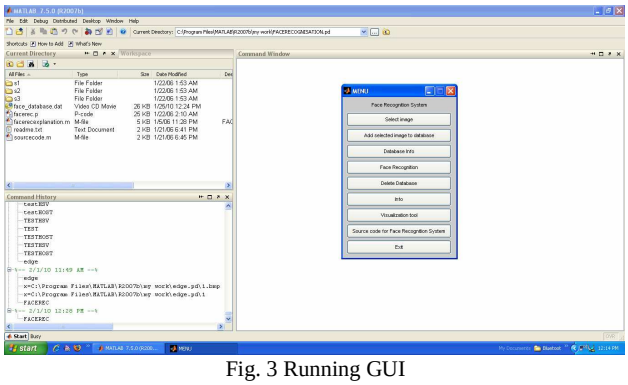


Fig. 3 Running GUI

We can add as many faces and correspondingly can assign the ID number for each image (As shown in Fig. 5) so that each image is represented by the Unique ID number[14-16]. We can perform face recognition[5] (As shown in Fig. 6) as

we select a particular face which was previously added in the data base algorithm search for the near by face as per work flow (As shown in Fig. 7) with reference to class [6]-[8] and distance from the face space.

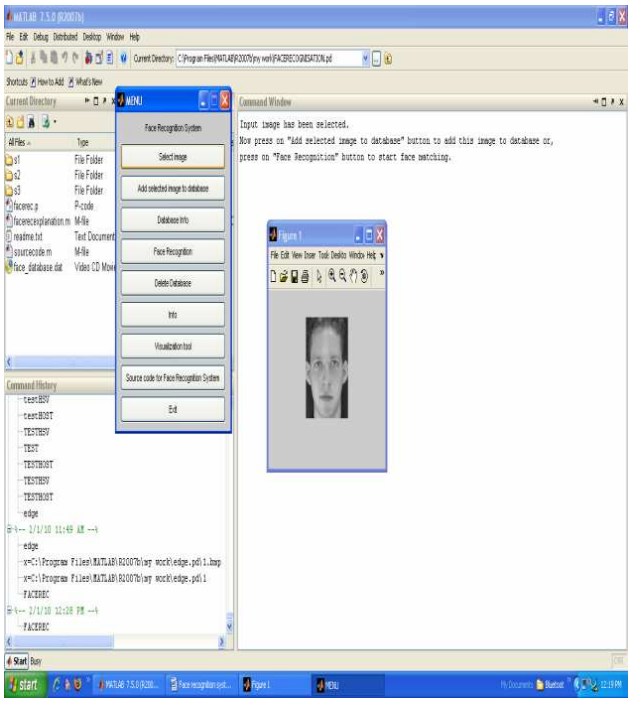


Fig. 4. Selecting Input Image

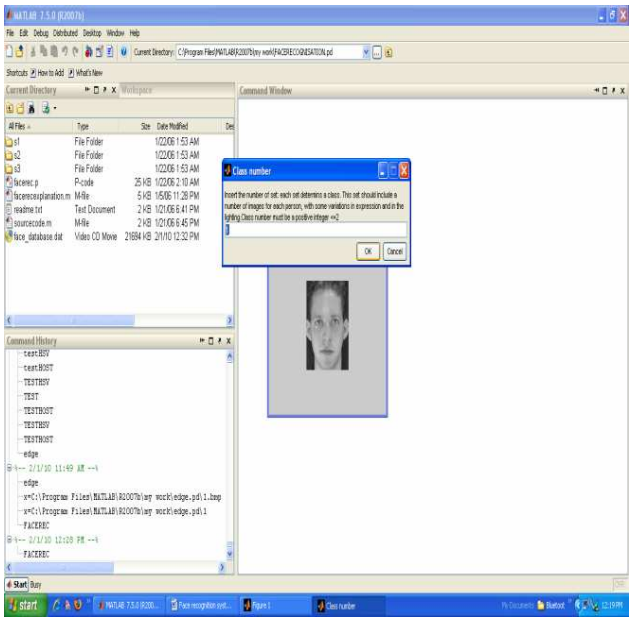


Fig. 5 Assigning ID Number

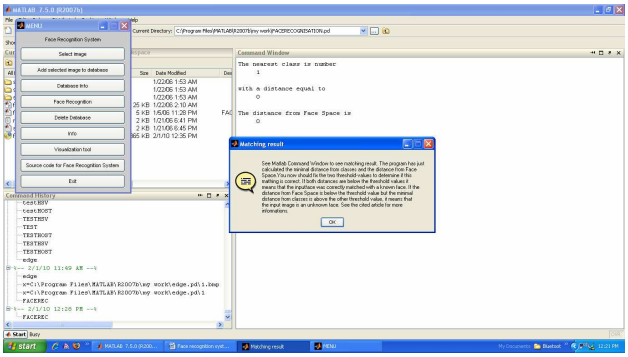


Fig. 6 Performing Face Recognition

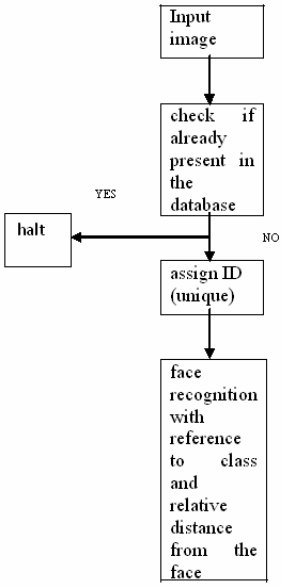


Fig. 7 Work Flow of Algorithm

VIII. Scope for Optimization in Face Recognition

The face is the commonly used biometric characteristics for person recognition. The most popular approaches to face recognition are based on shape of facial attributes, such as eyes, eyebrows, nose, lips, chin and the relationships of these attributes. Recognizing people by their facial features (or vectors) is the oldest identification mechanism of all, going back at least to our early primate ancestors. But there can be various factors affecting recognition of faces as it involves human behavior, hence distinct characteristics resulting in different images of the same face. Below listed are some factors that may be improved upon in order to achieve better face recognition[16-17]. These are illustrated by the shown images as follows.

- Limitations of Eigen faces Approach.
- Variations in lighting conditions
- Different lighting conditions for enrolment and query.
- Bright (As from Fig. 6) light causing image saturation.

Fig. 8 Effect on images due to Lighting

- Differences in pose – Head orientation
- 2D feature distances appear to distort.
- Expression -Change in feature location and shape.

REFERENCES

[1] Turk, M., Pentland, A., "Eigenfaces for recognition", J. Cognitive Neuroscience, 3 (1991), pp. 71–86.
[2] Belhumeur, P., Hespanha, J., Kriegman, D., "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection", IEEE Transactions on Pattern Analysis and Machine Intelligence, 19 (1997) , pp. 711–720.
[3] Tolba, El-Baz, El-Harby, "Face Recognition – A Literature Review", International Journal of Signal Processing, vol. 2, no. 2, 2005, pp. 88-103.
[4] M. Turk and A. Pentland, "Eigenfaces For Recognition", Journal of Cognitive Neuroscience, vol. 3, 1991, pp.71-86.
[5] Turk, Pentland, "Face Recognition Using Eigen Faces", In Proceedings of Computer Vision and Pattern Recognition, IEEE, June 1991, pp. 586-591.
[6] http://scien.stanford.edu/class/ee368/projects2001/dropbox/project16/
[7] http://www.irc.atr.jp/%7Emlyons/pub_pdf/fg98-1.pdf
[8] http://www.kasrl.org/jaffe.html
[9] S. Zeki, "The Visual Image in Mind and Brain", Scient. American, 1992, pp. 43-50.
[10] R. Baron, "Mechanisms of human facial recognition", Int. J. Man-Machine Studies, Vol. 15, 1981, pp. 137-178.
[11] C. Liu, "Statistical And Evolutionary Approaches For Face Recognition", Ph.D. Dissertation, Geo. Mason Uni, 1999.
[12] Y. Cui and J. Weng, "A Learning-Based Prediction-and-Verification Segmentation Scheme for Hand Sign Image Sequance", IEEE Trans. on Patt. Ana. and Mach. Intell., vol. 21, 1999, pp. 798-804.
[13] S. J. McKenna, S. Gong and Y. Raja, "Modelling Facial Color and Identity with Gaussian Mixtures", Patt. Recog, vol. 31(12), 1998, pp. 1883-1892.
[14] M. Turk and A. Pentland, "Eigenfaces for Recognition", Journal of Cognitive Neuroscience, vol. 3(1), 1991, pp. 71-86.
[15] T. McInery and D. Terzopoulos, "Deformable Models In Medical Image Analysis: A Survey", Medical Image Analysis, vol. 1, 1996, pp. 91-108.
[16] W.N. Martin and J.K. Aggarwal, "Dynamic Scene Analysis: A Survey", Computer Vision, Graphics and Image Process, vol. 7, 1978, pp. 356-374.
[17] J. K. Aggarwal and N. Nandhakumar, "On The Computation Of Motion From Sequences Of Images", Proc. IEEE, Vol. 76, 1988, pp. 917-935.