Volume 2, No. 3, May-June 2011
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

A Face Recognition implementation Using The Simple PCA LDA Algorithm

Sukhvinder Singh*
Mtech CSE (4th sem)
Sri Sai College Of Engg. & Tech.,
Jammu University ,Pathankot
sukhaish@gmail.com

Meenakshi Sharma
HOD CSE
Sri Sai College Of Engg. & Tech.,
Jammu University,Pathanko
Mss.s.c.e.t@gmail.com

Dr. N Suresh Rao
HOD CSE
Sri Sai College Of Engg. & Tech.,
Jammu University,Pathanko

Abstract: The objective is to recognise and identify faces, not previously presented to or in some way processed by the system. There are many datasets involved in this project. Some of them are the ORL, MIT database which consisting of a large set of images of different people. The database has many variations in pose, scale, facial expression and details. Some of the images are used for training the system and some for testing. The test set is not involved in any part of training or configuration of the system, except for the weighted committees as described in a section later on.
Keywords: Face recognition, PCA,Symbols

INTRODUCTION

The purpose of face Detection algorithm is to examine a set of images and try to find the exact match of a given image. An advanced system would be a neural network face Detection algorithm. The system examines small windows of the image in order to calculate the distances of given points. That would be done from any algorithm but in a system where we use neural networks the system arbitrates between multiple networks in order to improve performance over a single network.

The goal of this ongoing project is to formulate paradigms for detection and Detection[1] of human faces, and especially develop an algorithm, which is going to have high performance in complex backgrounds. One of the applications would be towards adding face-oriented queries to our image database project.

The fundamental principle, which we are exploiting for our face Detection algorithm, is Principal Component Analysis. Thought the algorithm is much simpler. One of the aims is to run tests in order to compare the algorithm with two PCA algorithm and also show that the calculation between two given point with the ARENA algorithm is efficient as much as if we had use the Euclidean distance. The algorithm used for the face Detection, from the project is known as ARENA. Similar to several other approaches to face Detection and identification, which use Principal Component Analysis (PCA) as pre-processing, dimensionality reduction and feature extraction, of the input images. One of the main parts of the project is a neural network. The use of a neural network makes the algorithm perform better.

In chapters two and three we are going to analyse the background of the project. A literature background of face Detection and neural networks discussing also the methods that were used for the project. In the next chapter, four there is a description of the datasets that where used in order to test and train the algorithm and the neural network, the implementation of which is in chapter six.After that, in chapter [2]seven there is a detailed analysis of the outputs we get from the programs and a comparison of the ARENA algorithm with other methods that have been used for face Detection, the theory of which is analysed in chapter five. Finally in chapter eight there is a discussion about the work that had been done and further improvement that could be done.

II. FACE DETECTION

Face Detection is a part of a wide area of pattern Detection technology. Detection and especially face Detection covers a range of activities from many walks of life. Face Detection is something that humans are particularly good at and science and technology have brought many similar tasks to us. Face Detection in general and the Detection of moving people in natural scenes in particular, require a set of visual tasks to be performed robustly. That process includes mainly three-task acquisition, normalisation and Detection. By the term acquisition we mean the detection and tracking of face-like image patches in a dynamic scene. Normalisation is the segmentation, alignment and normalisation of the face images[3], and finally Detection that is the representation and modelling of face images as identities, and the association of novel face images with known models.

III. NEURAL NETWORK

A neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing element or nodes. Neural network architecture is inspired by the architecture of biological nervous systems, which use many simple

processing elements operating in parallel to obtain high computation rates .

Neural networks are a form of microprocessor computer system with simple processing elements, a high degree of interconnection, simple scalar messages and adaptive interaction between elements .

The neural networks resemble the brain mainly in two respects

- Knowledge is acquired by the network through a learning process
- Interneuron connection strengths known as synaptic weights are used to store the knowledge.

That means to construct a machine that is able to think. Somehow, not really known yet, the brain is capable to think and perform some operations and computations, much faster sometimes from a computer even the "memory" is much less. How the brain is managed to do that is a hardware parallelism. The computing elements are arranged so that very many of them are working on a problem at the same time. Since there is a huge number of neurones, somehow the weak computing powers of these many slow elements are combined together to form a powerful result.

A Neural Network is an interconnected assembly of simple processing elements, units or nodes, whose functionality is loosely based on the animal neuron. The processing ability of the network is stored in the inter-unit connection strengths, or weights, obtained by a process of adaptation to, or learning from, a set of training patterns.

IV. How Artificial Neural networks Work.

Artificial neural networks can modify their behaviour in the response to their environment. This factor, more than any other, is responsible for the interest they[4] have received. Shown a set of inputs, which perhaps have specific desired output, they self adjust to produce consistent responses. A wide variety of training algorithms has been developed for that reason. Each of the algorithms has it's own strength and weaknesses.

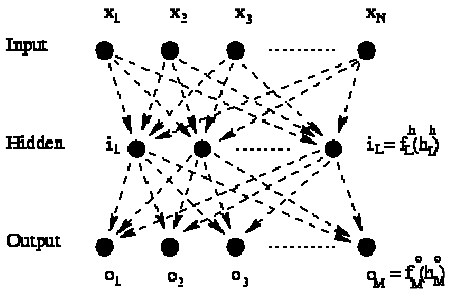
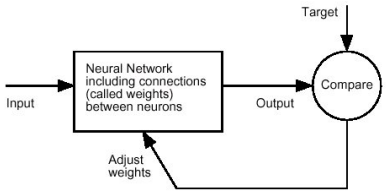


Figure.1 How neural network operate

A. *Feedforward Network*

Feedforward networks often have[5] one or more hidden layers of sigmoid neurons followed by an output layer of linear neurons. Multiple layers of neurons with non-linear transfer functions allow the network to learn non-linear and linear relationships between input and output vectors. The linear output layer lets the network produce values outside the



range –1 to +1. On the other hand, if it is desirable to constrain the outputs of a network then the output layer should use a sigmoid transfer function such as logsig.

Figure.2 Feedforward neural network

Furthermore in the case of multiple-layer networks we use the number of the layers to determine the superscript on the weight matrices. The appropriate notation is used in the two-layer tansig or purelin network shown next.

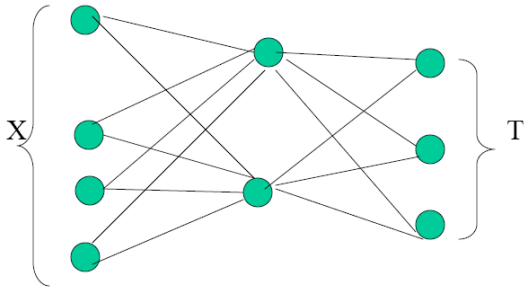


Figure.3 Neural Network

A feed forward network can be used as a general function approximation. It can approximate any function with a finite number of discontinuities, arbitrarily well, given sufficient neurons in the hidden layer.

In order to create a feedforward neuron network we have to follow a specific procedure. The first step in training a feedforward network is to create the network object. Then we have to initialise the weights and the bias. Then the network is ready to be trained. A feedforward network takes as said, before an object as input and returns[7] a network object with all weights and biases initialised. There is a more detailed analysis of the network and the process we follow in the coding part of the report.

a)Database Design

The databases, which are used for the project, are standard databases of the University of Surrey, Olivetti-Oracle Research Lab and FERET. Thought is possible to test the algorithm with other databases as well.

The databases consist of more than 400 images, each. All the databases contain images of different people, but in sets. That means that there are a number of images of the same people in each of them. Though each image is different from each other.

[6]For example, in the ORL database we have ten different images of each of 40 distinct subjects. For some people, the images were taken at different times, with different lighting, where we might have facial expressions, with open or closed eyes, where the people are smiling or not and facial details, glasses or with out no glasses. Many images of a person can be acquired in a few seconds. Given sufficient data, it becomes possible to model class-conditional structure, i.e. to estimate probability densities for each person.

Apart from that, in all the databases images were taken against a dark homogeneous background with the subjects in an upright, frontal position, but also we have images with more complex backgrounds. On of the most important aspects of the databases is the variation of the pose. There is a limitation of ±20° at the posing angle. If the person's pose, in the image is more than then is nearly impossible to be detected from mainly any of the existing face Detection algorithms. Thought in the databases we have posing angle variations but with in the limits.

The files of the images that are used are in TIFF format, and can conveniently be viewed on UNIX, TM systems using the xv program. Most of the images have size of 92x112 pixels, with 256 grey levels per pixel.

b) Test and Train sets

The data sets that have been used for the particular project are divided to two sub sets. The first is the training set that contains the images that were used in order to train the algorithm and the neural network. Training sets are used from the two training programs, arntrn and nntrn. Samples of the set can be found in appendix II. The other set is the other subset is the testing database, which contains different images than the training set but of the same people. [8]

In MATLAB we use the commands imread and imresize in order to read the images and reduce the resolution. More detailed description of the commands and their properties is given in the code implementation chapter.

c) Algorithms for face Detection

As mentioned in the introduction but also in other parts of the report, there are many algorithms that can be used for face Detection. Most of them are based on the same techniques and methods. Some of the most popular are Principal component analysis and the use of eigenfaces.

B. Principal Component Analysis

On the field of face Detection most of the common methods employ Principal Component Analysis. Principal Component Analysis is based on the Karhunen-Loeve (K-L), or Hostelling Transform, which is the optimal linear method for[9] reducing redundancy, in the least mean squared reconstruction error sense. 1. PCA became popular for face Detection with the success of eigenfaces.

The idea of principal component analysis is based on the identification of linear transformation of the co-ordinates of a system. "The three axes of the new co-ordinate system coincide with the directions of the three largest spreads of the point distributions."

In the new co-ordinate system that we have now the data is uncorrected with the data we had in the first co-ordinate system. [2]

For face Detection, given dataset of N training images, we create N d-dimensional vectors, where each pixel is a unique dimension. The principal components of this set of vectors is computed in order to obtain a d x m projection matrix, W. The image of the i^{th} vector may be represented as weights:

$$\vec{\theta i} = (\theta i1, \theta i2, \ldots, \theta im)^T \qquad (1)$$

Such that

$$\vec{xi} = \vec{\mu} + W\vec{\theta} \qquad (2)$$

Approximates the original image where μ is the mean, of the χ_i and the reconstruction is perfect when m = d. P1

As mentioned before the ARENA algorithm is going to be tested and its performance is going to be compared with other algorithms. For the comparison we are going to use two different PCA algorithms. The first algorithm[11] is computing and storing the weight of vectors for each person's image in the training set, so the actual training data is not necessary. In the second algorithm each weight of each image is stored individually, is a memory-based algorithm. For that we need more storing space but the performance is better.

In order to implement the Principal component analysis in MATLAB we simply have to use the command prepca. The syntax of the command is

ptrans,transMat = prepca(P,min_frac)

Prepca pre-processes the network input training set by applying a principal component analysis. This analysis transforms the input data so that the elements of the input vector set will be uncorrected. In addition, the size of the input vectors may be reduced by retaining[10] only those components, which contribute more than a specified fraction (min_frac) of the total variation in the data set.

Prepca takes these inputs the matrix of centred input (column) vectors, the minimum fraction variance component to keep and as result returns the transformed data set and the transformation matrix.

a) Algorithm

Principal component analysis uses singular value decomposition to compute the principal components. A matrix whose rows consist of the eigenvectors of the input covariance matrix multiplies the input vectors. This produces transformed input vectors whose components are uncorrected and ordered according to the magnitude of their variance.

Those components, which contribute only a small amount to the total variance in the data set, are eliminated. It is assumed that the input data set has already been normalised so that it has a zero mean.

In our test we are going to use two different "versions' of PCA. In the first one the centroid of the weight vectors for each person's images in the training set is computed and stored. On the other hand in PCA-2 a memory based variant

ofPCA, each of the weight vectors in individually computed and stored.

C. Eigenfaces

Human face Detection is a very difficult and practical problem in the field of pattern Detection. On the foundation of the analysis of the present methods on human face Detection, [12]a new technique of image feature extraction is presented. And combined with the artificial neural network, a new method on human face Detection is brought up. By extraction the sample pattern's algebraic feature, the human face image's eigenvalues, the neural network classifier is trained for Detection. The Kohonen network we adopted can adaptively modify its bottom up weights in the course of learning. Experimental results show that this method not only utilises the feature aspect of eigenvalues but also has the learning ability of neural network. It has better discriminate ability compared with the nearest classifier. The method this paper focused on has wide application area. The adaptive neural network classifier can be used in other tasks of pattern Detection.

In order to calculate the eigenfaces and eigenvalues in MATLAB we have to use the command eig. The syntax of the command is

d = eig(A)

V,D = eig(A)

V,D = eig(A,'nobalance')

d = eig(A,B)

V,D = eig(A,B)

d = eig(A) returns a vector of the eigenvalues of matrix A. V,D = eig(A) produces matrices of eigenvalues (D) and eigenvectors (V) of[13] matrix A, so that A*V = V*D. Matrix D is the canonical form of A, a diagonal matrix with A's eigenvalues on the main diagonal. Matrix V is the modal matrix, its columns are the eigenvectors of A. The eigenvectors are scaled so that the norm of each is 1.0. Then we use W,D = eig(A'); W = W' in order to compute the left eigenvectors, which satisfy W*A = D*W.

V,D = eig(A,'nobalance') finds eigenvalues and eigenvectors without a preliminary balancing step. Ordinarily, balancing improves the conditioning of the input matrix, enabling more accurate computation of the eigenvectors and eigenvalues. However, if a matrix contains small elements that are really due to round-off error, balancing may scale them up to make them as significant as the other elements of the original matrix, leading to incorrect eigenvectors. We can use the no balance option in this event.

d = eig(A,B) returns a vector containing the generalised eigenvalues, if A and B are square matrices. V,D = eig(A,B) produces a diagonal matrix D of generalised eigenvalues and a full matrix V whose columns are the corresponding eigenvectors so that A*V = B*V*D. The eigenvectors are scaled so that the norm of each is 1.0.

D. Euclidean distance

One of the ideas on which face Detection is based is the distance measures, between to points. The problem of finding the distance between two or more point of a set is defined as the Euclidean distance. The Euclidean distance is usually referred to the closest distance between two or more points. So we can define the Euclidean distance d_{ij} between points x x_{ik} and x_{jk} as :

$$d_{ij} = \sum_{k=1}^{p} (x_{ik} - x_{jk})^2 \quad (3)$$

V. IMPLEMENTATION

The first component of our system is a filter that receives as input a 20x20 pixel region of the image, and generates an output ranging from 1 to -1, signifying the presence or absence of a face, respectively. To detect faces anywhere in the input, the filter is applied at every location in the image. To detect faces larger than the window size, the input image is repeatedly reduced in size (by subsampling), and the filter is applied at each size. This filter must have some invariance to position and scale. The amount of invariance determines the number of scales and positions at which it must be applied. For the work presented here, we apply the filter at every pixel position in the image, and scale the image down by a factor of 1.2 for each step in the pyramid. The filtering algorithm is shown in . First, a preprocessing step, adapted from , is applied to a window of the image. The window is then passed through a neural network, which decides whether the window contains a face. The preprocessing first attempts to equalize the intensity values in across the window. We fit a function which varies linearly across the window to the intensity values in an oval region inside the window. Pixels outside the oval may represent the background, so those intensity values are ignored in computing the lighting variation across the face. The linear function will approximate the overall brightness of each part of the window, and can be subtracted from the window to compensate for a variety of lighting conditions. Then histogram equalization is performed, which non-linearly maps the intensity values to expand the range of intensities in the window. The histogram is computed for pixels inside an oval region in the window. This compensates for differences in camera input gains, as well as improving contrast in some cases.The preprocessed window is then passed through a neural network. Although the figure shows a single hidden unit for each subregion of the input, these units can be replicated. For the experiments which are described later, we use networks with two and three sets of these hidden units. Similar input connection patterns are commonly used in speech and character recognition tasks .The network has a single, real-valued output, which indicates whether or not the window contains a face. The network has some invariance to position and scale, which results in multiple boxes around some faces. To train the [14]neural network used in stage one to serve as an accurate filter, a large number of face and nonface images are needed. Nearly 1050 face examples were gathered fromface databases at CMU, Harvard2, and from the World Wide Web. The images contained faces of various sizes, orientations, positions, and intensities. The eyes, tip of nose, and corners and center of the mouth of each face were labelled manually. These points were used to normalize each face to the same scale, orientation, and position, as follows:

1. Initialize _ F, a vector which will be the average positions of each labelled feature over all the faces, with the feature locations in the first face F1.
2. 2. The feature coordinates in _ F are rotated, translated, and scaled, so that the average locations of the eyes will appear at predetermined locations in a 20x20 pixel window.
3. For each face i, compute the best rotation, translation, and scaling to align the face's features Fi with the average feature locations _ F. Such transformations can be written as a linear function of their parameters. Thus, we can write a system of linear equations mapping the features from Fi to _ F.
4. 4. Update _ F by averaging the aligned feature locations F0 i for each face i.
5. 5. Go to step 2.

The alignment algorithm converges within five iterations, yielding for each face a function which maps that face to a 20x20 pixel window. Fifteen face examples are generated for the training set from each original image, by randomly rotating the images (about their center points) up to 10_, scaling between 90% and 110%, translating up to half a pixel, and mirroring. Each 20x20 window in the set is then preprocessed (by applying lighting correction and histogram equalization). A few example images are shown in Fig. 4. The randomization gives the filter invariance to translations of less than a pixel and scalings of 20%. Larger changes in translation and scale are dealt with by applying the filter at every pixel position in an image pyramid, in which the images are scaled by factors of 1.2. Practically any image can serve as a nonface example because the space of nonface images is much larger than the space of face images. However, collecting a "representative" set of nonfaces Rowley, Baluja, and Kanade: Neural Network-Based Face Detection (PAMI, January 1998) 4 is difficult. Instead of collecting the images before training is started, the images are collected during training, in the following manner:

1. Create an initial set of nonface images by generating 1000 random images. Apply the preprocessing steps to each of these images.

2. Train a neural network to produce an output of 1 for the face examples, and -1 for the nonface examples. The training algorithm is standard error backpropogationwith momentum . On the first iteration of this loop, the network's weights are initialized randomly. After the first iteration, we use the weights computed by training in the previous iteration as the starting point.

3. Run the system on an image of scenery *which contains no faces*. Collect subimages in which the network incorrectly identifies a face (an output activation > 0).

4. Select up to 250 of these subimages at random, apply the preprocessing steps, and add them into the training set as negative examples. Go to step 2.

Stage Two: Merging Overlapping Detections and Arbitration

The raw output from a single network will contain a number of false detections. In this section, we present two strategies to improve the reliability of the detector: merging overlapping detections from a single network and arbitrating among multiple networks.

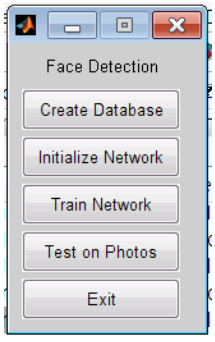
Merging Overlapping Detections

Most faces are detected at multiple nearby positions or scales, while false detections

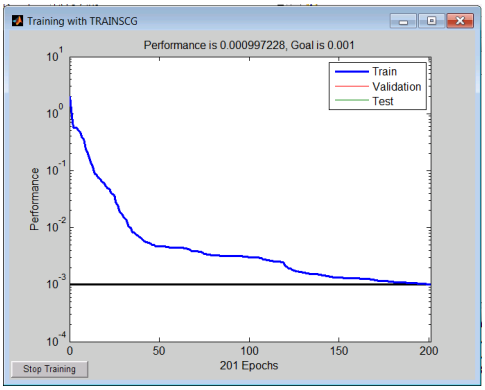
often occur with less consistency. This observation leads to a heuristic which can eliminate many false detections. For each location and scale, the number of detections within a specified neighborhood of that location can be counted. If the number is above a threshold, then that location is classified as a face. The centroid of the nearby detections defines the location of the detection result, thereby collapsing multiple detections. In the experiments section, this heuristic will be referred to as "thresholding". If a particular location is correctly identified as a face, then all other detection locations which overlap it are likely to be errors, and can therefore be eliminated. Based on the above heuristic regarding nearby detections, we preserve the location with the higher number of detections within Rowley, Baluja, and Kanade: Neural Network-Based Face Detection a small neighborhood, and eliminate locations with fewer detections. In the discussion of the experiments, this heuristic is called "overlap elimination".

Each detection at a particular location and scale is marked in an image pyramid, labelled the "output" pyramid. Then, each location in the pyramid is replaced by the number of detections in a specified neighborhood of that location. This has the effect of "spreading out" the detections. A threshold is applied to these values, and the centroids (in both position and scale) of all above threshold regions are computed. All detections contributing to a centroid are collapsed down to a single point. Each centroid is then examined in order, starting fromthe ones which had the highest number of detections within the specified neighborhood. If any other centroid locations represent a face overlapping with the current centroid, they are removed from the output pyramid. All remaining centroid locations constitute the final detection result. In the face detection work described in , similar observations about the nature of the outputs were made, resulting in the development of heuristics similar to those described above.

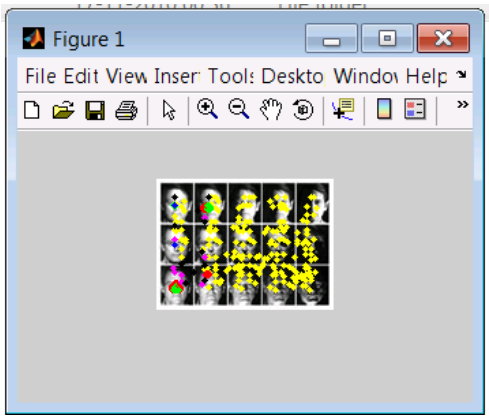
V. RESULTS



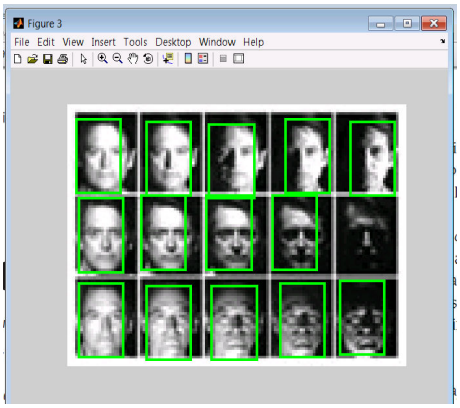
Main Page



Training



Recognising Images



Recognised Faces

VI. REFERENCES

[1] "Rafael C. Gonzalez and Richard E. Woods ", Digital Image Processing, 2nd Edition, , Prentice Hall, 2002.

[2] Korean standard: http://www.cwi.nl/~dik/english/codes/stand.html#ksc

[3] "Complete discrete 2-d Gabor transforms by neural networks for image analysis and compression", J.D. Daugman, IEEE Trans. Acoustics, Speech, and Signal Processing, 36:1169-1179, 1988.

[4] "An evaluation of the two-dimensional gabor filter model of simple receptive fields in cat striate cortex.", J.P. Jones and L.A. Palmer. , J. Neurophysiol., 58(6):1233-1258, 1987.

[5] "A tutorial on Principal Components Analysis", Lindsay I Smith, February 26, 2002.

[6] "Face Detection using fisherface algorithm and elastic graph matching",
Hyuyng-Ji Lee, Wan-Su Lee, and Jae-Ho Chung, Dept. of Electronic Engr., Inha Univ., Inchon 402-751, Korea

[7] "A Comparative Study On Color Edge Detection", Andreas Koschan, Proceedings 2nd Asian Conference on Computer Vision ACCV´95, Singapore, 5-8 December 1995, Vol. III, pp. 574-578.

[8] "Chinese Optical Character Detection for Information Extraction from Video Images", Wing Hang Cheung, Ka Fai Pang, Michael R. Lyu, Kam Wing Ng, Irwin King , Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong

[9] "Text Localization, Enhancement and Binarization in Multimedia Documents", [1]Christian Wolf, [1]Jean-Michel Jolion, [2]Francoise Chassaing, [1]Lab. Reconnaissance de Formes et Vision, [2]France T´el´ecom R&D.

[10] "Data Mining", Course CSC5180 Web Site , The Chinese University of Hong Kong – The Department of Computer Science and Engineering : http://www.cse.cuhk.edu.hk/~lwchan/teaching/csc5180.html

[11] "Spatial Mapping in the Primate Sensory Projection: Analytic Structure and Relevance to Perception", Biological Cybernetics, vol. 25, pp. 181.194, 1977.

[12] "Gray-Scale Character Detection by Gabor Jets Projection", Hiroshi Yoshimura, Minoru Etoh, Kenji Kondo and Naokazu Yokoya, Graduate School of Information Science, Nara Institute of Science and Technology Multimedia Laboratories, NTT DoCoMo, Inc

[13] "Field. Relations between the statistics of natural images and the response properties of cortical cells", David J. Field, Optical Society of America A, 4(12):2379-2394, 1987.

[14] "Statistics of Natural Images: Scaling in the wood", Daniel L. Ruderman & William Bialek, Physical Review Letters, 73(6):814-817, 1994.