##### UNIQUE IDENTIFICATION USING TONGUE PRINT

##### A PROJECT REPORT

###### ***Submitted by***

##### Saketh P(10bce0495)

**Anne Mary Joy(10bce0460)**

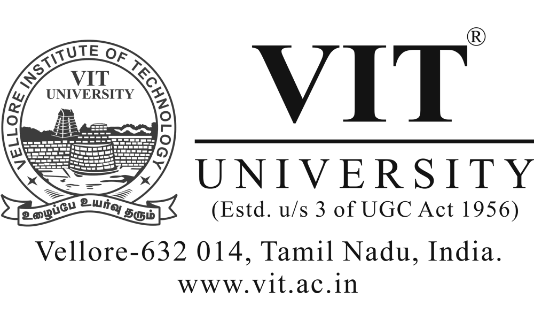
***in partial fulfillment for the award of the degree***

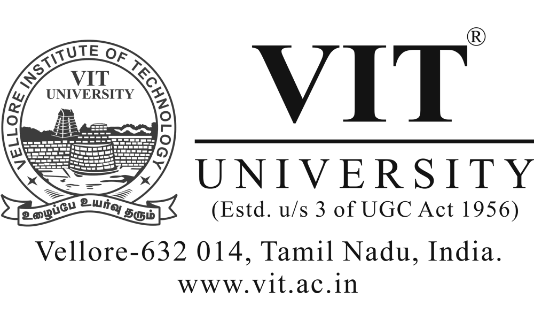
***of***

##### B.TECH

**in**

Computer Science And Engineering

****

****

**School of Computing Science & Engineering**

**BONAFIDE CERTIFICATE**

This is to certify that the project report entitled “**Unique Identification Using Tongue Print”**submitted by **Anne Mary Joy(10BCE0460)**and **Saketh P (10BCE0495)** to Vellore Institute of Technology, Vellore in partial fulfilment of the requirement for the award of the degree of B.TECH in COMPUTER SCIENCE & ENGINEERING is a record of bonafidefor Project Report undertaken by her under my supervision. The report fulfils the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

**SUPERVISOR**

Date: Date:

**Internal Examiner (s) External Examiner (s)**

**Acknowledgement**

First and foremost, we would like to express our sincere gratitude and respect to  
our project guide,**Prof. Vijayarajan.V** without whose advice, guidance, supervision and patience,this project would not have been possible.

Besides,wethank**Dr.G.Vishwanathan**, President/Chancellor, VIT University, **Dr. Margret Announcia,** Director of SCSE and **Prof.Anand** M, Program Manager of SCSE for giving us the opportunity to undertake this project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO.** | **TITLE** | **PAGE NO.** |
|  |  |  |
| **1.** | **ABSTRACT** | 4 |
| 2. | OBJECTIVE | 4 |
| 3. | LIST OF FIGURES/SNAPSHOTS | 4 |
| 4. | INTRODUCTION | 5 |
| 5. | STEPS INVOLVED | 5 |
| 6. | LITERATURE SURVEY | 6 |
| 7. | METHODOLOGY/DESIGN | 7 |
|  | 7.1 DATA GATHERING | 7 |
|  | 7.2IMAGE PREPROCESSING | 7 |
|  | 7.3 IMAGE RESIZE | 7 |
|  | 7.4 APPLICATION OF 2D DCT | 8 |
|  | 7.5 IMAGE RESHAPE AND FEATURE VECTOR | 9 |
|  | 7.6 NEURAL NETWORK | 10 |
|  | 7.6.1SELF ORGANIZING MAP | 10 |
| 8 | IMPLEMENTATION | 12 |
| 9 | RESULT | 21 |
| 10 | CONCLUSION AND FURTHER WORK | 21 |
| 11 | REFERENCES | 22 |

**1 ABSTRACT**

Forensic and biometric systems need uniqueness in identifying relations, and tongues provide that stature of uniqueness being one of the four vividly used biometric authentication analyzers of the human body in security and medical systems.This report describes the work completed for the mini project, design of a recognition system using tongue prints. This project includes a broad literature survey about some of the existing methods for recognition namely fingerprints and retina scan. The developed algorithm for the system formulates an image-based approach, using the Two-Dimensional Discrete Cosine Transform (2D-DCT) for image compression and the Self-Organizing Map (SOM) Neural Network for recognition purpose, simulated in MATLAB.

**2 OBJECTIVE**

The aim of this paper is to design a system wherein the identity of an individual can be ascertained using his tongue print.In fields requiring security and those related to medicine,this plays a useful role as the tongue of certain diseased persons vary greatly from those of normal healthy ones.Similarly, we use various methods to get the images of a required resolution, extract it into a more feasible image using many different techniques in MATLAB such as (sharpness, after morphological open) and various data.

We then obtain a vector matrix for a single feature extraction method like DCT and apply various feature extractions and form a feature Vector matrix.

We then use neural networks for recognition purpose.

**3 LIST OF FIGURES/SNAPSHOTS**

* Steps involved in performing the activity.
* Image Preprocessing
* Image Resize
* Image Reshape and feature vector creation
* Feature Vector
* Neural Network Chart
* Segmentation to 4 parts and saving in current directory
* Application of 2D DCT and feature vector creation

**4 INTRODUCTION**

The programming language used to design and implement the recognition system recognition code is MATLAB. The reason for using MATLAB in this project is due to its Neural Network and Image Processing toolbox that helped to obtain an efficient code.

**5 Steps involved**

5.1) Data Gathering

5.2) Image Preprocessing in in paint and MS office picture manager

5.3) Image Resize in MATLAB

5.4) Apply 2D DCT

5.5)Image reshape and Feature Vector creation

5.6) Neural network training

Neural Network training

Image reshape and feature vector creation

Image resize in MATLAB

Apply 2D DCT

Image preprocessing in paint and MS office picture manager

Take images and form database

**6 LITERATURE SURVEY**

Human beings use textures or scales, perception, various patterns to classify the image of any being (be it animals, pictures on the screen) and an entire set of data entities.

In security systems and in information security, human beings mainly use two major types of biometric recognition which are Fingerprint scan recognition and retina scan display, their uses being the facts that a fingerprint scan gives a very high accuracy, can be used in PC systems and is the most economic measure of using a biometric system, and is one of the most developed systems of biometric method, Now very less information is developed and is present on a tongue print which is safe and secure and can be coherently used even in medical applications.

Both finger and tongue print require a small storage space, so the picture size in the database is not going to be an overhead. An additional feature being that less reduction is needed in a tongue print compared to a finger print scan, which requires you to have a template or reduce the size of it more and more as you go on. A major disadvantage of a finger print scan is avoided in case of a tongue print because here chemicals do not disrupt the texture of it. The reason of it being dry may lead to mistakes with dirty fingers.

Now, in ocular identification technologies like Retinal or an Iris scan, all the details are related to the eye which changes differently based on different emotions and fields which are more sensitive compared to something as robust as a tongue.

* Both of them require a camera for their functioning while iris scan is performed mainly through scanning than training.Speedy results are produced which make an individual’s identity of the subject is verified very quickly.

One difference that gives a low point is that it giving you less accurate results that can be developed in the future.

**7 METHODOLOGY/DESIGN  
  
7.1 Data Gathering**

Images of tongues of different subjects were taken under uniform light conditions and light backgrounds with a digital camera for the training database. The images were then transferred from the digital camera to the computer. Figure below shows some of the images taken.



**7.2 Image Pre-processing in Paint/MS office Picture Manager**

This step hastwo phases:

7.2.1) Adjusting image levels, contrast, brightness and colors

7.2.2) Image resizing from 1200 x 1600 pixels to 512 x 512 pixels

7.3 Image resize

The images were resized to *512 x 512 pixels* by reducing the image size using Paint and MS office Picture Manager.

**7.4 Apply 2D DCT**

After all images were resized to 8 x 8 pixels and saved, the next step was to compress them by applying the 2D DCT. When the 2D DCT is applied with a mask, high-coefficients are in the image discarded. Then the 2D IDCT is applied to regenerate the compressed image, which is blurred due to loss of quality and also smaller in size.

To find a technique to apply the 2D DCT to an image, the MATLAB help was searched in the Image Processing Toolbox. A program for 2D DCT image compression in MATLAB help was found with its source code.

The mask used was

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

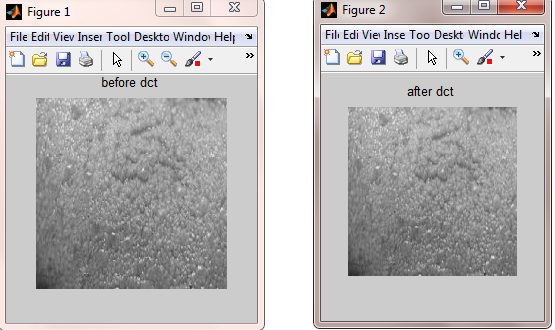
0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

After the DCT program found in MATLAB help was modified according to the DCT options and the masking host was set to 8 coefficients and all images were compressed using the DCT. The newly compressed images were saved under a different filename. Figure below shows the program used for DCT image compression.

All the images were run through the program and were compressed. Each input image and the image to written (saved) was changed and the program was run one by one, for all images which were to be compressed.

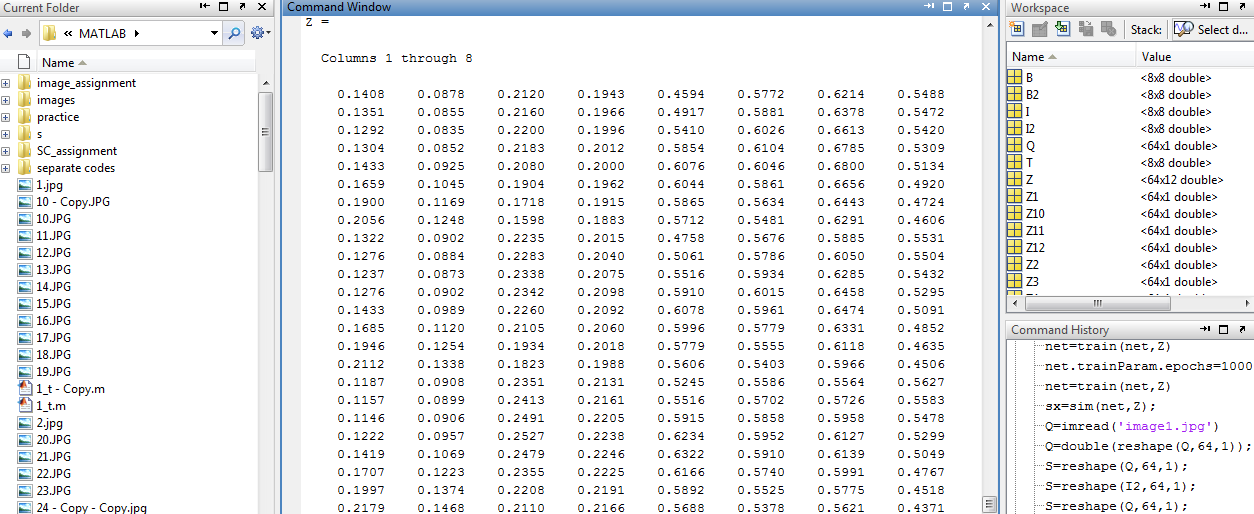


**7.5 Image reshape and Feature Vector creation**

For the image data to be input into the neural network, it should follow the form of only one column, despite the number of rows. Currently, all the resized and DCT compressed images are in the form of 8 x 8 pixels.

Hence the image data needed to be reshaped from an 8 x 8 matrix to a *64 x 1 array* for it to be used both for the input and training database of the neural network. For reshaping the MATLAB *reshape* command is used.All image data was then reshaped to a 64 x 1 array.

*Feature vector*



**7.6 Neural Network**

***7.6.1 Self-Organizing Maps***

SOMs were found to be efficient for image data management and proved to be an accurate closest matching technique of untrained input images with trained database of images.

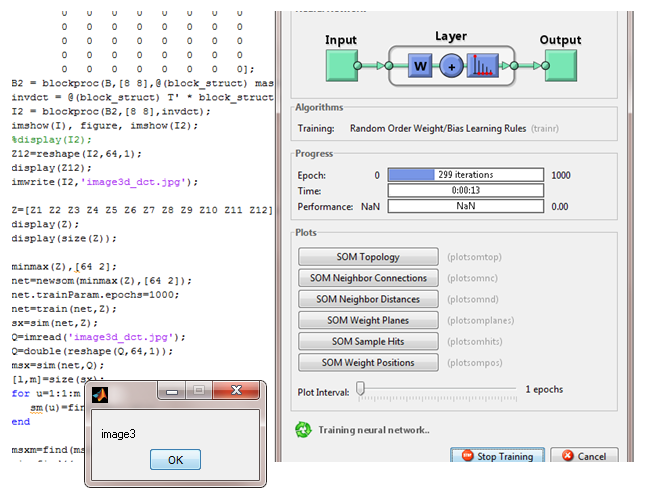
For the design of SOM, a set of 12 tongue print data of 3 different subjects were used for the training database which was loaded into MATLAB. Figure shows image data loaded into MATLAB into vector Z.

A SOM was then created with the MATLAB command *newsom.* Parameters for the SOM network were selected to be a minimum and maximum point for each row on vector *P*; training database. There were 64 minimum and 64 maximum points selected altogether. Figure shows the SOM minimum and maximum points for the training database in a 64 x 2 array. After the SOM neural network was created, it was trained for 1000 epochs. Figure shows the training of the SOM neural network for 1000 epochs.

After the SOM neural network was trained and simulated for the 12 images in the training database, the SOM neural network was then simulated for the single input image. Figure shows loading the input image into MATLAB and simulating it for the SOM neural network.

The image in the training database which is the closest match by the SOM neural network for the input image is found by finding the minimum absolute deviation.

After the closest matched training database images are found, they are then classified. Classification in MATLAB was done through simple *if* and *else* statements. So, if the number of training database images change, the number of if and else statements would also change.



**8 IMPLEMENTATION**

*Code for image preprocessing(to improve sharpness,removingnoise,etc)*

closeall;

clearall;

l=imread('image\_three.jpg');

I= rgb2gray(l);

h=fspecial('unsharp');

l2=imfilter(I,h);

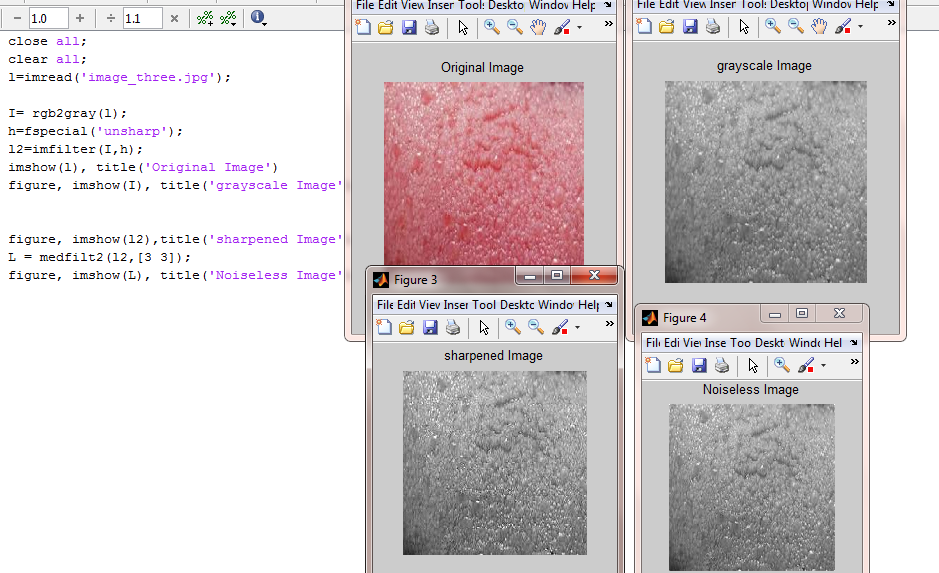
imshow(l), title('Original Image')

figure, imshow(I), title('grayscale Image');

figure, imshow(l2),title('sharpened Image');

L = medfilt2(l2,[3 3]);

figure, imshow(L), title('Noiseless Image');



*To split image in 4 parts and save in current directory*

clc;

clearall;

myImage = imread('image1.jpg');

image\_red = myImage(:,:,1);

[m,n] = size(image\_red);

image\_top\_left = myImage(1:m/2,1:n/2,:);

image\_top\_right = myImage(1:m/2,m/2:m,:);

image\_bottom\_left = myImage(m/2:m,1:n/2,:);

image\_bottom\_right = myImage(m/2+1:m,n/2+1:n,:);

figure,imshow(image\_top\_left),title('top left corner');

figure,imshow(image\_top\_right),title('top right corner');

figure,imshow(image\_bottom\_left),title('bottom left corner');

figure,imshow(image\_bottom\_right),title('bottom right corner');

imwrite(image\_top\_left,'im1a.jpg');

imwrite(image\_top\_right,'im1b.jpg');

imwrite(image\_bottom\_left,'im1c.jpg');

imwrite(image\_bottom\_right,'im1d.jpg');

imag1a=imresize(image\_top\_left,[8,8],'nearest');

imwrite(imag1a,'image1a.jpg');

imag1b=imresize(image\_top\_right,[8,8],'nearest');

imwrite(imag1b,'image1b.jpg');

imag1c=imresize(image\_bottom\_left,[8,8],'nearest');

imwrite(imag1c,'image1c.jpg');

imag1d=imresize(image\_bottom\_right,[8,8],'nearest');

imwrite(imag1d,'image1d.jpg');

myImage = imread('image2.jpg');

image\_red = myImage(:,:,1);

[m,n] = size(image\_red);

image\_top\_left = myImage(1:m/2,1:n/2,:);

image\_top\_right = myImage(1:m/2,m/2:m,:);

image\_bottom\_left = myImage(m/2:m,1:n/2,:);

image\_bottom\_right = myImage(m/2+1:m,n/2+1:n,:);

figure,imshow(image\_top\_left),title('top left corner');

figure,imshow(image\_top\_right),title('top right corner');

figure,imshow(image\_bottom\_left),title('bottom left corner');

figure,imshow(image\_bottom\_right),title('bottom right corner');

imwrite(image\_top\_left,'im2a.jpg');

imwrite(image\_top\_right,'im2b.jpg');

imwrite(image\_bottom\_left,'im2c.jpg');

imwrite(image\_bottom\_right,'im2d.jpg');

imag2a=imresize(image\_top\_left,[8,8],'nearest');

imwrite(imag2a,'image2a.jpg');

imag2b=imresize(image\_top\_right,[8,8],'nearest');

imwrite(imag2b,'image2b.jpg');

imag2c=imresize(image\_bottom\_left,[8,8],'nearest');

imwrite(imag2c,'image2c.jpg');

imag2d=imresize(image\_bottom\_right,[8,8],'nearest');

imwrite(imag2d,'image2d.jpg');

myImage = imread('image3.jpg');

image\_red = myImage(:,:,1);

[m,n] = size(image\_red);

image\_top\_left = myImage(1:m/2,1:n/2,:);

image\_top\_right = myImage(1:m/2,m/2:m,:);

image\_bottom\_left = myImage(m/2:m,1:n/2,:);

image\_bottom\_right = myImage(m/2+1:m,n/2+1:n,:);

figure,imshow(image\_top\_left),title('top left corner');

figure,imshow(image\_top\_right),title('top right corner');

figure,imshow(image\_bottom\_left),title('bottom left corner');

figure,imshow(image\_bottom\_right),title('bottom right corner');

imwrite(image\_top\_left,'im3a.jpg');

imwrite(image\_top\_right,'im3b.jpg');

imwrite(image\_bottom\_left,'im3c.jpg');

imwrite(image\_bottom\_right,'im3d.jpg');

imag3a=imresize(image\_top\_left,[8,8],'nearest');

imwrite(imag3a,'image3a.jpg');

imag3b=imresize(image\_top\_right,[8,8],'nearest');

imwrite(imag3b,'image3b.jpg');

imag3c=imresize(image\_bottom\_left,[8,8],'nearest');

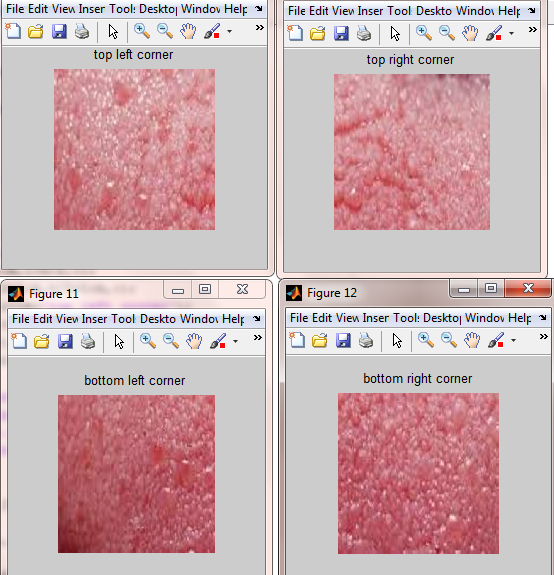
imwrite(imag3c,'image3c.jpg');

imag3d=imresize(image\_bottom\_right,[8,8],'nearest');

imwrite(imag3d,'image3d.jpg');



Original image



*Code for application of 2D-DCT and feature vector extraction*

clc;

clearall;

I = imread('image1a.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z1=reshape(I2,64,1);

display(Z1);

imwrite(I2,'image1a\_dct.jpg');

I = imread('image1b.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z2=reshape(I2,64,1);

display(Z2);

imwrite(I2,'image1b\_dct.jpg');

I = imread('image1c.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z3=reshape(I2,64,1);

display(Z3);

imwrite(I2,'image1c\_dct.jpg');

I = imread('image1d.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z4=reshape(I2,64,1);

display(Z4);

imwrite(I2,'image1d\_dct.jpg');

I = imread('image2a.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z5=reshape(I2,64,1);

display(Z5);

imwrite(I2,'image2a\_dct.jpg');

I = imread('image2b.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z6=reshape(I2,64,1);

display(Z6);

imwrite(I2,'image2b\_dct.jpg');

I = imread('image2c.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z7=reshape(I2,64,1);

display(Z7);

imwrite(I2,'image2c\_dct.jpg');

I = imread('image2d.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z8=reshape(I2,64,1);

display(Z8);

imwrite(I2,'image2d\_dct.jpg');

I = imread('image3a.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z9=reshape(I2,64,1);

display(Z9);

imwrite(I2,'image3a\_dct.jpg');

I = imread('image3b.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z10=reshape(I2,64,1);

display(Z10);

imwrite(I2,'image3b\_dct.jpg');

I = imread('image3c.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z11=reshape(I2,64,1);

display(Z11);

imwrite(I2,'image3c\_dct.jpg');

I = imread('image3d.jpg');

I = rgb2gray(I);

I = im2double(I);

T = dctmtx(8);

dct = @(block\_struct) T \* block\_struct.data \* T';

B = blockproc(I,[8 8],dct);

mask = [1 1 1 1 0 0 0 0

1 1 1 0 0 0 0 0

1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0];

B2 = blockproc(B,[8 8],@(block\_struct) mask .\* block\_struct.data);

invdct = @(block\_struct) T' \* block\_struct.data \* T;

I2 = blockproc(B2,[8 8],invdct);

imshow(I), figure, imshow(I2);

%display(I2);

Z12=reshape(I2,64,1);

display(Z12);

imwrite(I2,'image3d\_dct.jpg');

Z=[Z1 Z2 Z3 Z4 Z5 Z6 Z7 Z8 Z9 Z10 Z11 Z12];

display(Z);

display(size(Z));

minmax(Z),[64 2];

net=newsom(minmax(Z),[64 2]);

net.trainParam.epochs=1000;

net=train(net,Z);

sx=sim(net,Z);

Q=imread('image3d\_dct.jpg');

Q=double(reshape(Q,64,1));

msx=sim(net,Q);

[l,m]=size(sx);

for u=1:1:m

sm(u)=find(sx(:,u));

end

msxm=find(msx);

pic=find((sm-msxm)==min(abs((sm-msxm))));

display(pic);

if(pic>0)&(pic<=4)

msgbox('image1');

elseif (pic>4) & (pic<=8)

msgbox('image2');

elseif (pic>8) & (pic<=12)

msgbox('image3');

end

end

end

**9 RESULT**

The SOM neural network was trained and simulated for the 12 images in the training database, the SOM neural network was then simulated for the single input image. The image in the training database which is the closest match by the SOM neural network for the input image is found by finding the minimum absolute deviation.

*Classification:-*

After the closest matched training database images are found, they are then classified. Classification in MATLAB was done through simple *if* and *else* statements. So, if the number of training database images change, the number of if and else statements would also change. Classification of the subject found in the image training database with respect to the input image.

**10 CONCLUSION AND FURTHER WORK**

In this project,we managed to identify an input sample tongue print making use of the fact that every individual has a unique tongue print.Here,we have taken samples of tongue prints,split them and used the individual parts for training the neural network so that a test input tongue print can be checked.In this way we can identify the individual if his details are present in the database.

As future work,this system may be implemented in the medical field in case of diseases where symptoms are observed on the tongue. Training of neural networks can be done using such infected tongues as inputs to ‘classify’ the disease. Further, when faced with an event of unknown samples of similar infected tongue images,the trained neural network can be obtained.

11**REFERENCES**

We obtained our knowledge and information from the following sources.

[1]ZHI LIU, JING-QI YAN, DAVID ZHANG, QUN-LIN TANG-A tongue print image database for recognition-Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, Hong Kong, 19-22 August 2007

[2]Pankaj H Chandankhede-Soft Computing Based Texture Classification with

MATLAB Tool-International Journal of Soft Computing and Engineering

(IJSCE) ISSN: 2231-2307, Volume-2, Issue-2, May 2012

[3]JyotiRajharia, P.C. Gupta-A New and Effective Approach for Fingerprint Recognition by using Feed Forward Back Propagation Neural Network-International Journal of Computer Applications (0975 – 8887) Volume 52– No.10, August 2012.

[4]Matlab Product HELP

[5]Mathworks.in(for theory concepts)