Theory:

Let us denote the **i**th image vector be $\mathbf{x}_i$ after reordering. The conventional MACE filter is better formulated in the frequency domain. The discrete Fourier transform (DFT) of the column vector $\mathbf{x}_i$ is denoted by $\mathbf{X}_i$ and we define the training image data matrix **X** as:

$$X = [X_1, X_2, \dots \dots ., X_N],$$

Where the size of X is ***d X N*** and ***N*** is the number of training image. Let the vector **h** be the filter in the space domain and represent by **H** its Fourier transform vector. We are interested in the correlation between the input image and the filter. The correlation of the **i**th image sequence $\boldsymbol{x_i(n)}$ with the filter sequence $\boldsymbol{h(n)}$ can be written as:

$$g_i(n) = x_i(n) \otimes h(n)$$

By Parseval's theorem, the correlation energy of the ith image can be written as a quadratic form:

$$E_i = \mathbf{H}^H \mathbf{D}_i \mathbf{H}$$

where $\mathbf{D}_i$ is a diagonal matrix of size $\boldsymbol{d \times d}$ whose diagonal elements are the magnitude squared of the associated element of $\mathbf{X}_i$, i.e the power spectrum of $\boldsymbol{x_i(n)}$ and the superscript ***H*** denotes the Hermitian transpose i.e. transpose of conjugate. The objective of the MACE filter is to minimize the average correlation energy over the image class while simultaneously satisfying an intensity constraint at the origin for each image. The value of the correlation at the origin can be written as:

$$g_i(0) = \mathbf{X}^H \mathbf{H} = c_i$$

for $\boldsymbol{i = 1, 2, \cdots\cdots, N}$ training images, where $\boldsymbol{c_i}$ is the user specified output correlation value at the origin for the **i**th image. Then the average energy over all training images is expressed as:

$$E_{avg} = \mathbf{H}^H \mathbf{D} \mathbf{H}$$

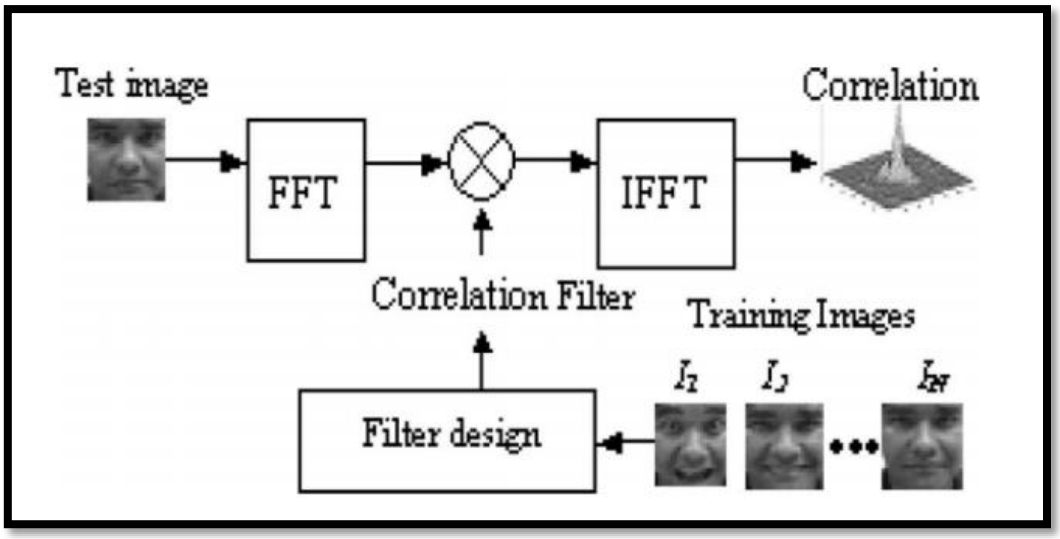
where

$$D = \frac{1}{N} \sum_{i=1}^{N} D_i$$

The MACE design problem is to minimize $\boldsymbol{E_{avg}}$ while satisfying the constraint, $\mathbf{X}^H\mathbf{H} = \mathbf{c}$, where $\mathbf{c} = [c_1, c_2, \cdots, c_N]$ is an ***N*** dimensional vector. This optimization problem can be solved using Lagrange multipliers, and the solution is:

$$\mathbf{H} = \mathbf{D}^{-1}\mathbf{X}(\mathbf{X}^H\mathbf{D}^{-1}\mathbf{X})^{-1}\mathbf{c}$$

It is clear that the spatial filter **h** can be obtained from **H** by an inverse DFT. Once **h** is determined, we apply an appropriate threshold to the output correlation plane and decide whether the test image belongs to the class of the template or not.

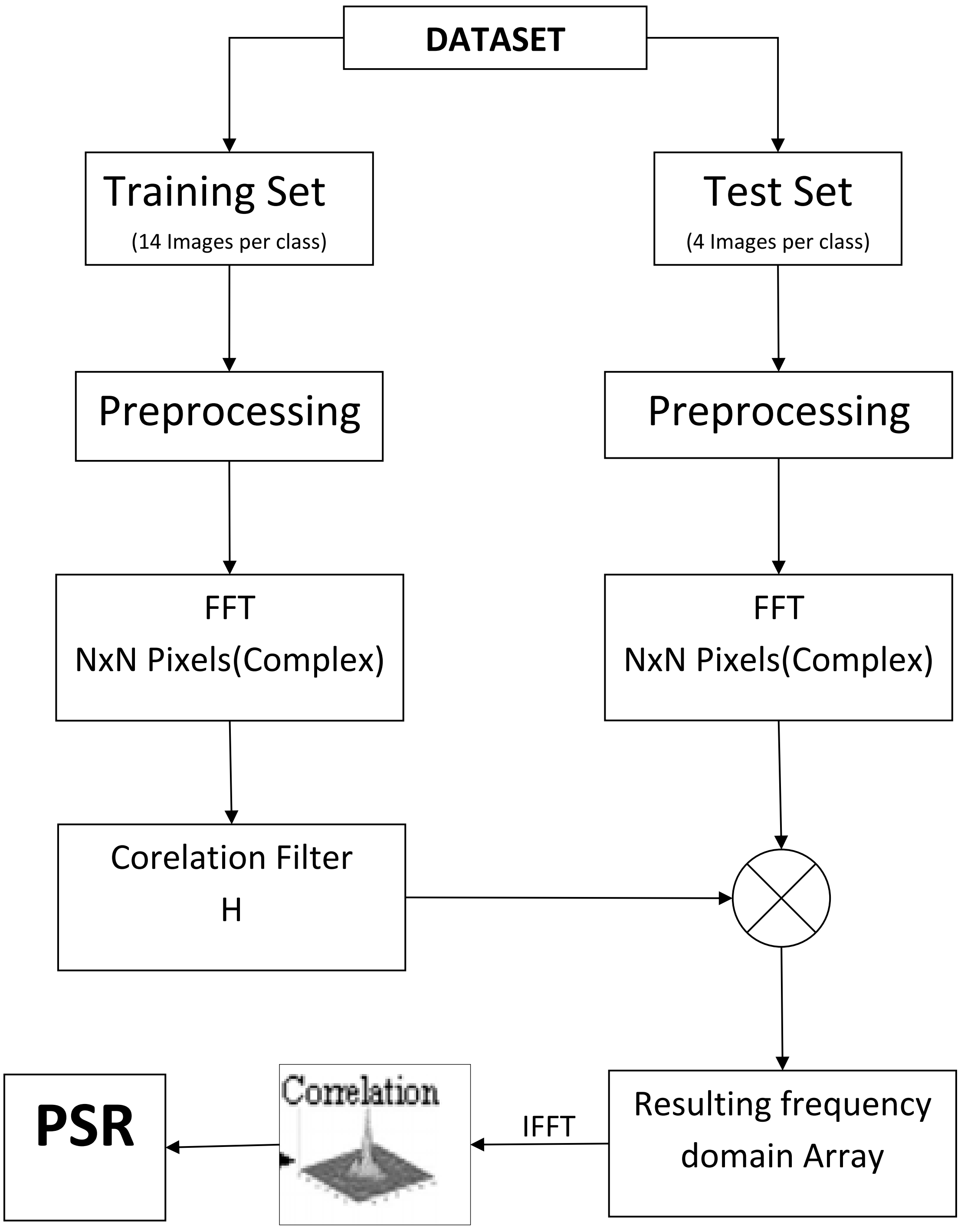


For each subject in this illumination subset, a MACE filter was designed based on images numbered and the resulting correlation filter was tested against all test images of each of the 5 subjects in the database. We expect the correlation output to exhibit sharp, high peaks for the authentic and no such peaks for the impostors. We quantify the peak sharpness by the peak-to-side lobe ratio (PSR) defined as

$$PSR = \frac{Peak - mean}{std}$$

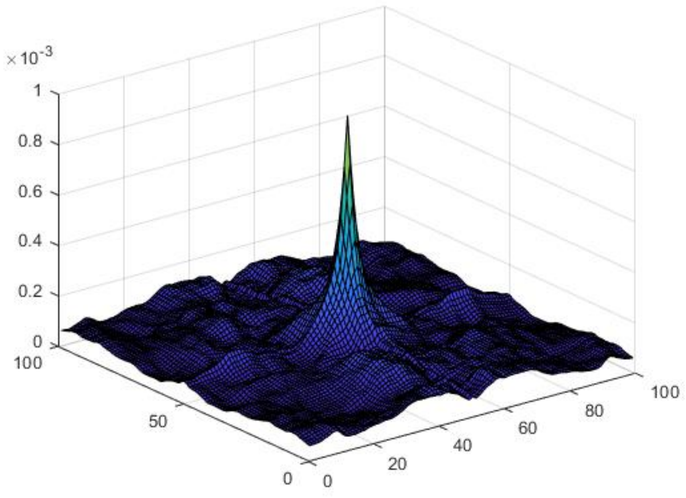
Where **peak** is the largest value in the correlation output and **mean** and **std** are the average value and the standard deviation of the correlation outputs. PSR is designed to measure the relative height of the correlation peak to the background and is observed to be not too sensitive to the sizes of these regions.

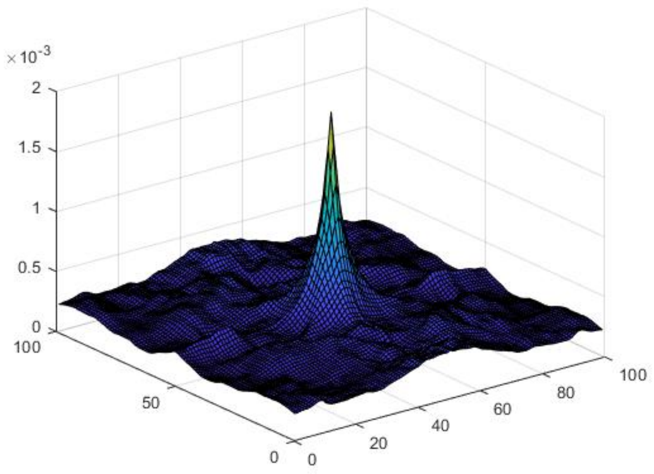
Flowchart:

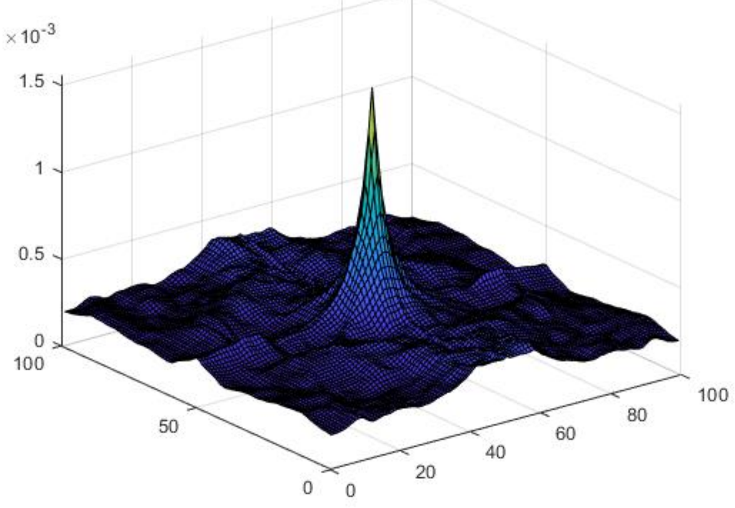

DATASET
Training Set
(14 Images per class)
Test Set
(4 Images per class)
Preprocessing
Preprocessing
FFT
NxN Pixels(Complex)
FFT
NxN Pixels(Complex)
Corelation Filter
H
Resulting frequency domain Array
IFFT
Correlation
PSR

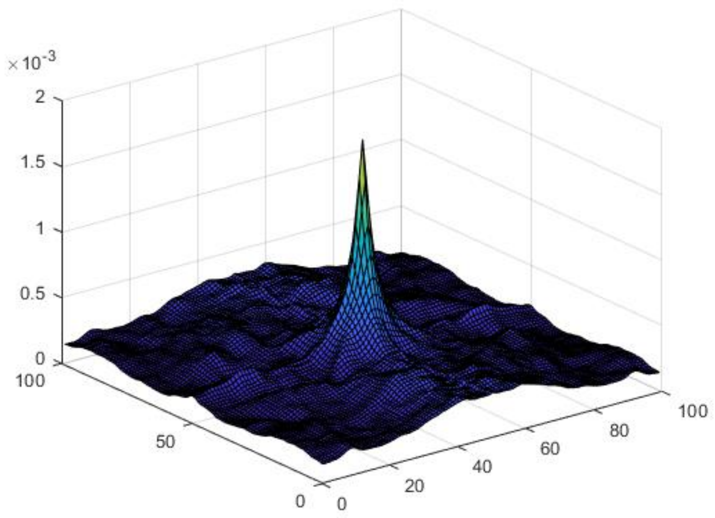
Result:

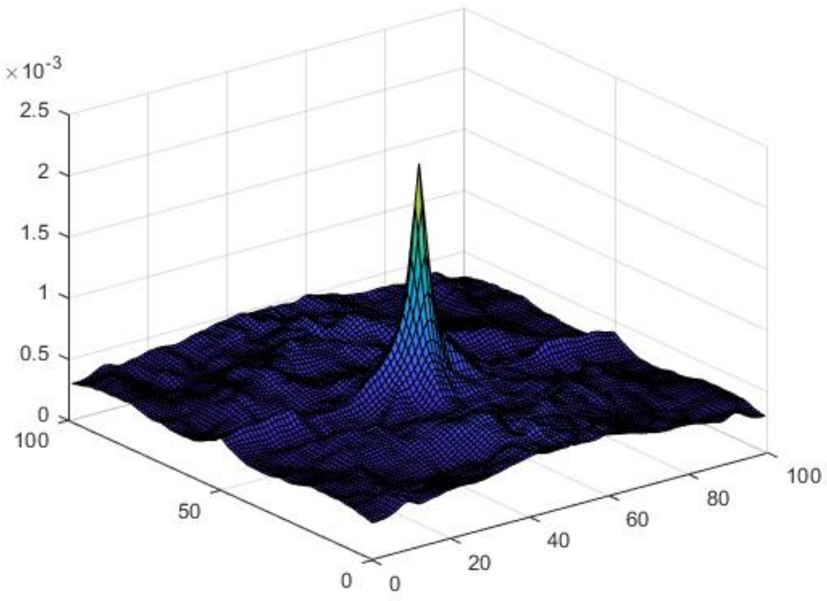
To test our designed MACE filters, we will use them to compute PSR values and will then compare the results with PSR values obtained with MACE filters. We get following Correlation graph of test image from different class filters:











Conclusion:

- We implemented successfully the MACE filter using MATLAB.
- We tested it on test images and it was giving good accuracy.
- We tested for 5 classes image, each having 14 image, which sums 70 images. We assigned 70% images for training and remaining for testing.

APPENDIX

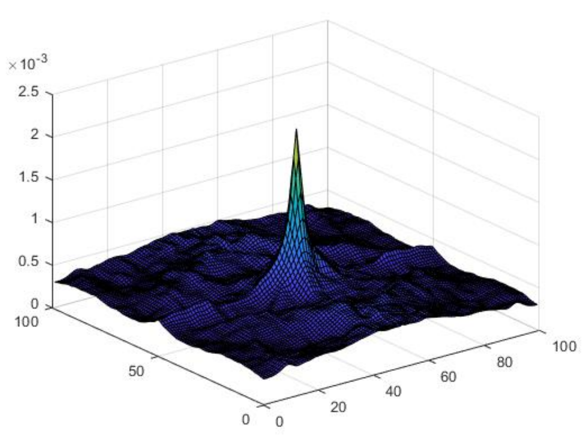
AVERAGE CORRELATION ENERGY (MACE) FILTER

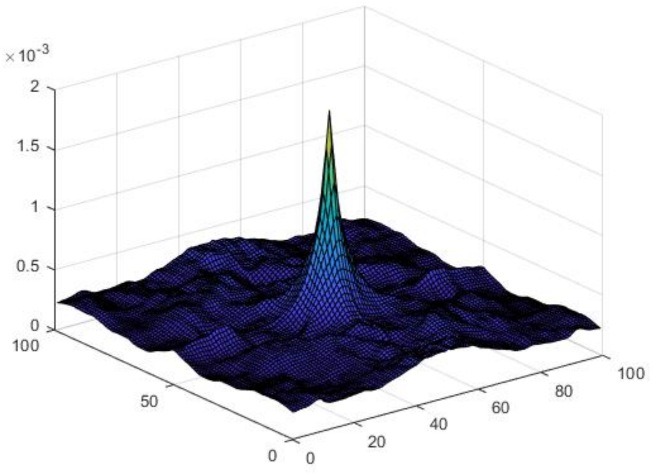
```
clc;
close all;
clear all;
warning off;
dd=100;

path1='findataset\imgcatg\ClassA';
path2='findataset\imgcatg\ClassB';
path3='findataset\imgcatg\ClassC';
path4='findataset\imgcatg\ClassD';
path5='findataset\imgcatg\ClassE';

test=imread('p1.tif');
J = imresize(test, [dd dd]);
J1 = abs(fftshift(fft2(J)));

H1=filt(path1);
R=J1.*H1;
[xxx yyy]=find(max(max(R))==R);
Region=R(xxx-10:yyy+9,xxx-10:yyy+9);
Region(8:12,8:12)=zeros(5);
R1=Region;
peak_value=max(max(abs(R1)));  %For computing PEAK from all elements
mean_value=mean(R1,'all');  %For mean of all elements
std_dev=std2(R1);
PSR(1)=(peak_value-mean_value)/std_dev;

H2=filt(path2);
S=J1.*H2;
[xxx yyy]=find(max(max(S))==S);
Region=S(xxx-10:yyy+9,xxx-10:yyy+9);
Region(8:12,8:12)=zeros(5);
S1=Region;
peak_value=max(max(abs(S1)));  %For computing PEAK from all elements
mean_value=mean(S1,'all');  %For mean of all elements
std_dev=std2(S1);
PSR(2)=(peak_value-mean_value)/std_dev;

H3=filt(path3);
T=J1.*H3;
[xxx yyy]=find(max(max(T))==T);
Region=T(xxx-10:yyy+9,xxx-10:yyy+9);
Region(8:12,8:12)=zeros(5);
T1=Region;
peak_value=max(max(abs(T1)));  %For computing PEAK from all elements
mean_value=mean(T1,'all');  %For mean of all elements
std_dev=std2(T1);
PSR(3)=(peak_value-mean_value)/std_dev;

H4=filt(path4);
U=J1.*H4;
```

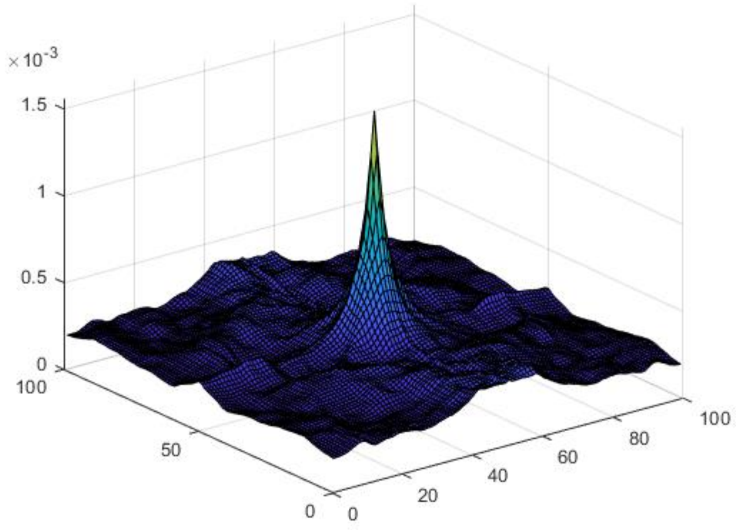
```
[xxx yyy]=find(max(max(U))==U);
Region=U(xxx-10:yyy+9,xxx-10:yyy+9);
Region(8:12,8:12)=zeros(5);
U1=Region;
peak_value=max(max(abs(U1)));  %For computing PEAK from all elements
mean_value=mean(U1,'all');  %For mean of all elements
std_dev=std2(U1);
PSR(4)=(peak_value-mean_value)/std_dev;

H5=filt(path5);
V=J1.*H5;
[xxx yyy]=find(max(max(V))==V);
Region=V(xxx-10:yyy+9,xxx-10:yyy+9);
Region(8:12,8:12)=zeros(5);
V1=Region;
peak_value=max(max(abs(V1)));  %For computing PEAK from all elements
mean_value=mean(V1,'all');  %For mean of all elements
std_dev=std2(V1);
PSR(5)=(peak_value-mean_value)/std_dev;

maximum = max(PSR);
class=find(PSR==maximum);
figure(1);
surf(abs(ifftshift(ifft2(R))));
figure(2);
surf(abs(ifftshift(ifft2(S))));
figure(3);
surf(abs(ifftshift(ifft2(T))));
figure(4);
surf(abs(ifftshift(ifft2(U))));
figure(5);
surf(abs(ifftshift(ifft2(V))));
str1 = "Pic belongs to class-> ";
str2 = int2str(class);
str=append(str1,str2)
msgbox(str,'Success')



function H = filt(path)
filenames=dir(fullfile(path,'*.tif'));
noi=numel(filenames);   %number of images
dd=100;       %required diamention of pics

%For matrix X  "X is FFT of training input images in column vectors"
N=noi-4;     %No. of training images
for nn = 1:N
    f=fullfile(path, filenames(nn).name);
    our_images=imread(f);

    [m n] = size(our_images) ;
    J = imresize(our_images, [dd dd]);
    K=reshape(J,[],1);
    ZZZ=fftshift(fft(double(K)));
    X(:,nn)=fftshift(ZZZ);
end

tbl=size(X);
```

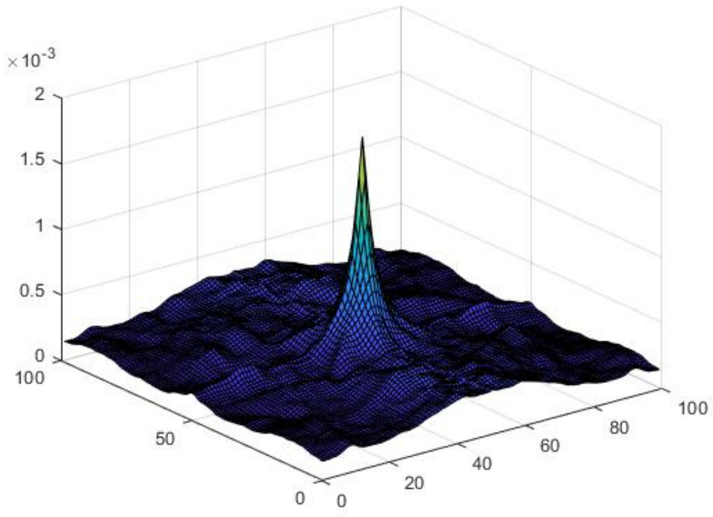
```
d=tbl(1,1);
%For diag mat Di of dxd, whose diagonal elements are "mag of squared
%associated element Xi" i.e. power spectrum of input images.
D = diag(mean(abs(X).^2,2));
u=ones(N,1);
h = inv(D)*X*(inv((ctranspose(X))*inv(D)*X))*u;
H = abs(reshape(h, size(J)));
end
```

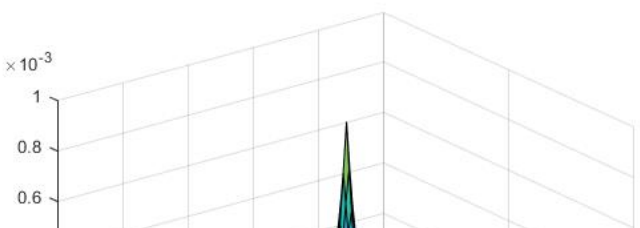
```
str = 

    "Pic belongs to class-> 2"
```

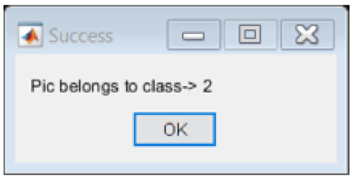












```
Pic belongs to class-> 2
```

Published with MATLAB® R2020b

References

1. K. Jeong and J. C. Principe, "The Correntropy Mace Filter for Image Recognition," 2006 16th IEEE Signal Processing Society Workshop on Machine Learning for Signal Processing, Arlington, VA, 2006, pp. 9-14, doi: 10.1109/MLSP.2006.275513.
2. Jeong, Kyu-Hwa, et al. "The correntropy MACE filter." Pattern Recognition 42.5 (2009): 871-885.
3. Savvides, Marios, BVK Vijaya Kumar, and Pradeep Khosla. "Face verification using correlation filters." 3rd IEEE Automatic Identification Advanced Technologies (2002): 56-61.