Specimen 'A': Title Sheet
(IMAGE PROCESSING USING GRAPHS)

Project report submitted in partial fulfillment of the requirement for the award of the degree of

Bachelor of Technology in Electronics & Communication Engineering

By

PRAVEEN RAJ V (1510110274)

Under supervision

of

Prof. Vijay Kumar Chakka

DEPARTMENT OF ELECTRICAL ENGINEERING

SCHOOL OF ENGINEERING, SHIV NADAR UNIVERSITY

Gautam Buddh Nagar, U.P., India 201314

(May 2019)

Specimen “B” – Candidate Declaration

I hereby declare that the thesis entitled “**Image Processing Using Graphs**” submitted for the B.Tech Degree program. This thesis was written in my own words. I have adequately cited and referenced the original sources.

(Signature)

(Praveen Raj V)

(1510110274)

Date: ___________

Specimen "C" CERTIFICATE

It is certified that the work contained in the B. Tech project report titled "**Image Processing Using Graphs**" by "Praveen Raj V" has been carried out under my / our supervision.

Signature of Supervisor(s)
Prof. Vijay Kumar Chakka
Department of Electrical Engineering
School of Engineering
Shiv Nadar University

May, 2019

ACKNOWLEDGEMENT

I wish to express a deep sense of gratitude to my guide Dr. Vijay Kumar Chakka, Professor, Department of Electrical Engineering, School of Engineering for his valuable guidance and continuous encouragement and guidance throughout the project period, which made all my endeavors successful.

I am greatly indebted to Dr. Madhur D Upadhayay, Assistant Professor, Advisor of Undergraduate studies (ECE), for the support and encouragement that he extended towards the project work.

I also thank the respective lab in-charges and staffs for their continuous support during this period.

INDEX

LIST OF FIGURES:

ABSTRACT:

Image processing is a term used to define the operations performed on an image to modify the image or extract the information from the image. One of the several fast growing and in deman field in image processing is the detection/classification/identification of an image. It finds great use in areas like face recognition, pattern recognition, etc. Over the years several techniques and methods have been developed to perform these operations in this field. Each of them have been found to produce several advantages and disadvantages.

Graphs are generic data representation forms that are useful for describing the geometric structures of data domains in numerous applications. In recent years, there have been researches going on to determine whether graphs can be used as a unified representation for image analysis and processing.

In this project, we aim to study the feasibility of graphs for such image processing and classification. We also explore the ways to classify images based on their underlying graph structure. To move forward in this regard, we first study in depth about graphs. One of the most important challenges in this whole process lies in the part where a graph has to be developed/learned from a given image. We study one of such graph learning methods where a signal is received sample by sample and its underlying graph structure is learnt. The same is applied on images and their underlying graph structures are used for image processing.

1. INTRODUCTION:

1.1. Image processing:

In technical context, an image can be defined as a distributed amplitude of colors. Images can be two dimensional or three dimensional. It can also be classified into two types namely, still images and moving images.

Image processing is the set of operations which are done on an image to enhance its quality or extract certain useful information from it. It includes enhancing/decreasing the quality of the image, processing an input image to obtain an output image, zooming in/out an image, etc. Image processing plays a vital role in medical imaging, satellite imaging, entertainment devices, etc.

1.2. Graph signal processing:

Graphs are generic data representation forms that are useful for describing the geometric structures of data domains in numerous applications [1]. A graph is basically a structure consisting vertices and edges. An edge connects two vertices. Each edge contains or is define by its own edge weight which usually depicts the relation between the two vertices it connects. Graphs are highly useful in signal processing, computer graphics, etc.

In our work, we aim to learn the underlying graph of a signal and also predict the upcoming values of a signal using a graph filter, which will be discussed later. This will be used later, when we consider an image in place of a signal.

2. PARAMETERS OF A GRAPH

2.1. Adjacency matrix:

A graph is usually defined by its adjacency matrix. Adjacency matrix helps us in concluding several properties of a graph. It is a matrix which conveys the connectivity of each node with other nodes in the graph and also the weights associated with the edges connecting them in case of a weighted adjacency matrix.

2.2. Laplacian matrix:

A Laplacian matrix is related to the adjacency matrix as follows.

$L = \mathrm{diag}(A\mathbf{1}) - A$; Where **1** is an all-one vector [2]

Laplacian matrix play an important role in graph signal processing. The eigenvalues and eigenvectors of the matrix helps us in various operations involving a graph.

2. LEARNING THE GRAPH UNDERLYING A SEQUENTIAL DATA

3.1. Graph model:

Initially the graph is assumed to be a weighted undirected graph without any self-loops i.e. a node is never connected to itself. Based on these, the following properties of the graph's adjacency matrix A is derived, (i). Symmetry ($A = A^T$) (ii). Non-negativity (i.e. no element of the adjacency matrix A is negative) (iii). No self-loops i.e. $a_{ii} = 0$ where a_{ij} are the elements of the adjacency matrix A.

Since it is known that the Laplacian matrix is given by L = D-W, where D is the degree matrix, the following properties of L are derived [2].

Symmetry: $L = L^T$

Non-positive off-diagonal elements: $l_{ij} \leq 0$; ¥ $i \neq j$

Positive definite

Nullspace: $L\frac{1}{\sqrt{N}}\mathbb{1} = 0$

3.2. Signal Model:

It is assumed that the observed signal is of discrete samples of a continuous time graph process s(t) which evolves according to the differential equation, [2]

$$s'(t) = -L^* s(t) + p(t)$$

Where L^* denotes the Laplacian matrix of the underlying graph and p(t) denotes the perturbations observed as a function of time. The variable p(t) can either be viewed as an outside force, which influences the evolution of the signal, or some internal events that subsequently diffuse over the graph [2]. When the above differential equation is solved we get

$$s(t) = e^{-tL^*} s(0) + \int_0^t e^{-(t-u)L^*} p(u) du$$

We know the evolution of graph signal at the initially at some t_0. The samples of s(t) are observed at t_i where $t_i = t_0 + i.T$. Here, 'i' denotes the ith sample and 'T' denotes the sampling period.

A recursive relationship is observed between adjacent samples which can be denoted as,

$$s(t_i) = e^{-TL^*} s(t_{i-1}) + \int_{t_i - T}^{t_i} e^{-(t_i-u)L^*} p(u) du$$

The relationship between $s(t_i)$ and $s(t_{i-1})$ only depends on L^* and on the perturbations p(t) between t_i and t_{i-1} [2].

Now the whole system is moved to a discrete domain by representing the equation as

$S_i = e^{-TL^*}.s_{i-1} + p_i$ where p_i is a random stochastic variable.

The prime objective of this algorithm to be developed is to estimate L^* from the signal as soon as a sample is observed. The above model fails to provide a computationally less expensive model when graphs of large size are encountered. Therefore, an equivalent linear model is developed. In the above equation, e^{-TL^*} is observed to be non-linear which leads to high complexity, considering the fact that L^* is a matrix. Therefore another variable W^* is defined as

$$W^* \triangleq e^{-TL^*}$$

which results in the following equation,

$S_i = W^*.s_{i-1} + p_i$ which is linear in nature.

This makes us derive at conclusion that $L^* = -\frac{[\ln(W^*)]}{T}$.

$$W^* = e^{-TL^*}$$

Expanding the above equation, we get an infinite series in terms of L^*. Meanwhile we also introduce the eigen-decomposition of L^*.

$$L^* = V\mathbb{I}V^T$$

where V is the matrix containing the eigenvectors column wise and $\mathbb{I}$ is the diagonal matrix containing the eigenvalues. Therefore, expanding W^*, we get,

$$W^* = Ve^{-T\mathbb{I}}V^T$$

where $e^{-T\mathbb{I}}$ = diag$\{e^{-T\mathbb{I}\text{k}(L^*)}\}$

Set of conditions are implied on W^* to preserve the properties of L^* [2]. They are,

(i). Symmetry

(ii). Non-negative elements

(iii). Spectral bound

(iv). Stochastic

A matrix is called stochastic when either each of its columns or rows add up to one.

3.3. Graph signal evolution:

An assumption is made that the graph defined by A and L is connected [2]. This makes us infer that the eigen-structure of $L^* = V\mathbb{I}V^T$ is

$$V = \left[\begin{matrix} \frac{1}{\sqrt{N}}\mathbb{1} & \overline{V} \end{matrix} \right]$$

$$\mathbb{I}_L = \begin{matrix} 0 & 0 \\ 0 & \mathbb{I}L \end{matrix}$$

Therefore, the matrix W^* also has a similar structure.

$$V = \left[\begin{matrix} \frac{1}{\sqrt{N}}\mathbb{1} & \overline{V} \end{matrix} \right]$$

$$\mathbb{I}_w = \begin{matrix} 1 & 0 \\ 0 & \mathbb{I}w \end{matrix} \qquad \text{(eigenvalue at zero becomes one)}$$

The mean across the graph of the signal at time ‘i’ is given by $S_{c,i} = \left(\frac{1}{N}\right).(I)\hat{}T.S_i$ where I is an all-one matrix. Subtracting this mean would yield the centered graph signal [2] i.e.

$$\overline{s}_i \triangleq s_i - s_{c,i} = \left(I - \frac{1}{N}\mathbb{1}\mathbb{1}^{\mathsf{T}}\right) s_i$$

Since the mean is independent of W, it contains no information about the graph. Now we can write the equation of the centered graph signal as

$$\overline{s}_i = \overline{W}^{\star}\overline{s}_{i-1} + \overline{p}_i$$

$\overline{W^*}$ can be defined as, $\overline{W^*} = W^* - \left(\frac{1}{N}\right)\mathbf{11^T}$

The eigen-decomposition of this new $\overline{W^*}$ can be derived from W^* as follows,

$$V = \left[\begin{array}{cc} \frac{1}{\sqrt{N}}\mathbb{1} & \overline{V} \end{array}\right], \quad \Lambda_{\overline{W}} = \left[\begin{array}{cc} 0 & 0 \\ 0 & \Lambda_W \end{array}\right]$$

3.4. Graph learning:

To learn the underlying graph of the online signal, an optimization problem is defined with the cost function associated with it. A stochastic gradient descent algorithm based on Laplacian Mean Square (LMS) strategy is formulated which helps us in learning $\overline{W^*}$ with respect to every online signal received [2].

$$\overline{W^*}(\mathrm{i}) = \overline{W^*}(i-1) + u\left(\overline{s(i)} - \overline{W^*}(i-1) * \overline{s(i-1)}\right) * s(i-1)^{\mathrm{T}}$$

The choice of the step size is crucial to the algorithm. A small step size would lead to a slow convergence of the algorithm and a large step size might result in the algorithm not converging at the point of minimum error. In this case, we take the step size as the minimum eigenvalue of the autocorrelation matrix of the input signal. Initially, smaller blocks i.e. 5, 30, 60,100 of the signal was considered and the resulting step size and final error of the obtained graph structure was observed. The error observed seemed to reduce as the block size increased, though with minimal amount. Therefore, the whole signal was considered to determine the step size.

3.5. Imposition of constraints:

It is important to preserve the properties of W^* seen above, to arrive at a valid graph structure. Therefore we include some projections on the above LMS algorithm that the resulting $\overline{W^*}$ would preserve the properties of W^*. Since $W^* = \overline{W^*} + \left(\frac{1}{N}\right)\mathbf{11^T}$, the properties of W^* can be imposed on $\overline{W^*}$ through the update relation as follows,

1).Non-negative elements : $\overline{W^*}_{ij} = \overline{w}_{ij}$, if $\overline{w}_{ij}$ >=0 ; -(1/N) , otherwise

2).Symmetricity: $\overline{W^*} = \left(\frac{1}{2}\right)(\overline{W^*} + \overline{W^*}^T)$

3).Null: $\overline{W^*} = \overline{W^*} - \left(\frac{1}{N}\right)\overline{W^*}\mathbf{11^T}$

4).Spectral bound: $\overline{W^*} = V\pi_t V^T$ where π_t is the eigenvalue matrix of $\overline{W^*}$ with imposed conditions that an eigenvalue is changed to zero if its less than zero and to one if its more than one [2].

Including the spectral bound condition in the update relation, increases the complexity of the algorithm several times [2]. Therefore, two types of algorithm are taken into account. Initially, the first three projections are imposed on the update relation and the same is named as Type 1. On top of all these projections, the fourth condition i.e. the spectral bound condition is imposed and it is named as Type 2.

To implement the above graph learning algorithm and to compare the above two types of the update relation, we take the Minnesota graph and the signal defined on it as an example. The Minnesota graph contains 2642 nodes with a sample of the signal defined on each of them. We take the signal as an online signal i.e. we take the samples of the signal one by one. The code for learning the underlying graph of this signal is included in Appendix 1.

The results of the implementation shows that the above does a commendable job is learning the graph of the signal. The final error observed between the recovered graph and the original graph was 0.000066203. Also, the Mean Square Error (MSE) between the original graph and the recovered graph in every iteration of the update relation was observed and plotted against the number of ieterations and the below graph was obtained. Here, y-axis represents the MSE and x-axis represents the number of iterations.

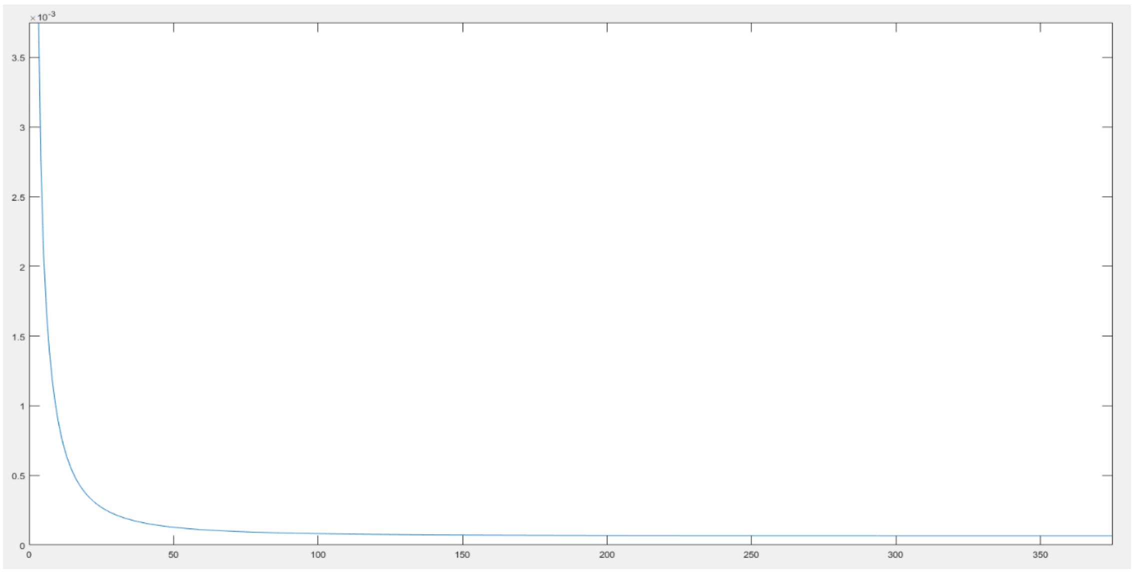


Fig.1. MSE vs Number of iterations for Type1

It is observed that an almost accurate graph (error = 0.001) is obtained much earlier i.e. at

Around 280 iterations. The above graph was observed for Type 1 algorithm. Type 2 gave a similar MSE with a similar final error. The plot of MSE vs Number of iterations for Type 2 is given below.

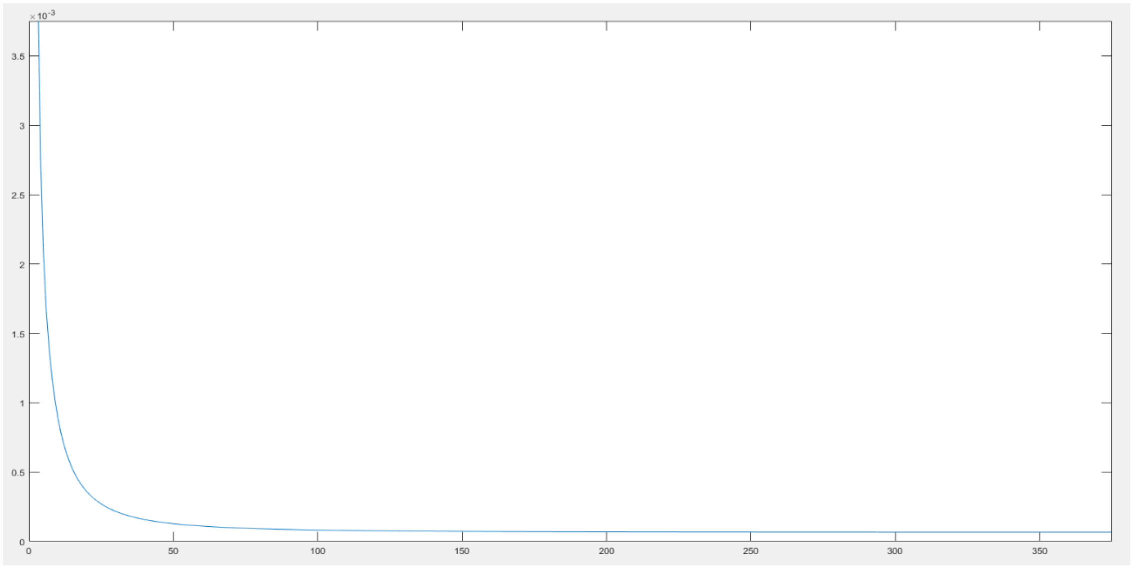


Fig.2. MSE vs Number of iterations for Type2

But the time taken to execute Type 2 was much higher than Type 1 because of the complexity of the spectral bound condition. Therefore, it was inferred that using Type 1 algorithm is more efficient than using Type 2 which leads us to use Type 1 in subsequent operations.

4. IMPLEMENTING THE GRAPH LEARNING ALGORITHM ON IMAGES

An RGB image is an image in which each pixel contains three information i.e. the intensity levels of the colors red, green and blue. An RGB image can be converted into a grayscale image for operational simplicity. A grayscale image is an image in which each pixel contains the level of intensity which ranges from 0 to 1. A value of 0 represents black and 1 represents white. In the subsequent processes in this section, we use the grayscale version of the 20x20 version of the below RGB image which is originally of the size 439x270.

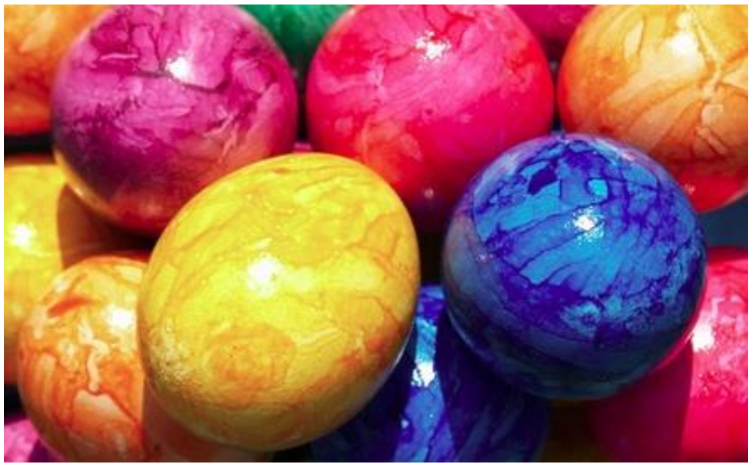
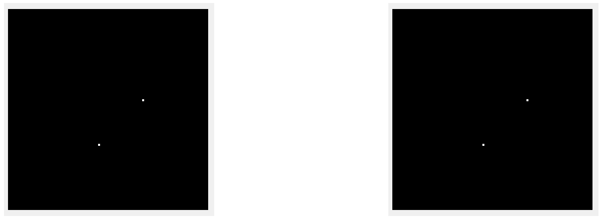


Fig.3. A random RGB image

In subsequent implementations, the adjacency matrix of a graph will be represented by a black and white image where black represents zeros and white represents ones.

4.1. Graphs underlying row and column wise representation of images:

The image is usually a matrix which represents a pixel by pixel information of the image.
The graph learning algorithm in the above section was implemented on the above image, first by taking the row wise elements of the image as an online signal and then column wise. The resulting adjacency matrix of both the graphs were found to be similar. Therefore, it was inferred that both row wise and column wise representation of the image would result in the same graph structure. The adjacency matrices found in the two cases are given below.



(i) (ii).

Fig.4. (i).row wise (ii). Column wise

4.2. Graphs of downsized versions of an image:

The 48x48 image was downsized to a 10x10 image and then, a 5x5 image and the resulting graph structures of all the three versions of the image were compared.

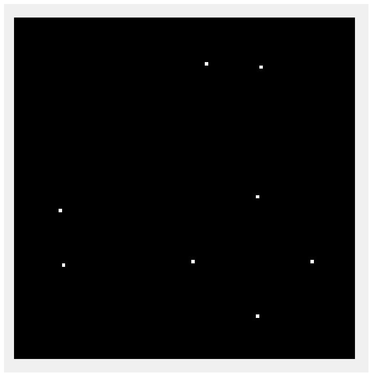
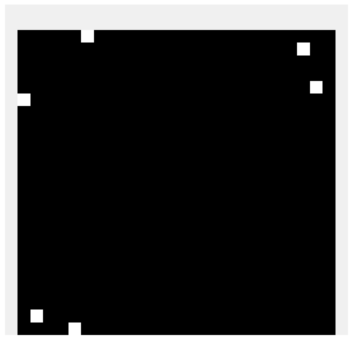



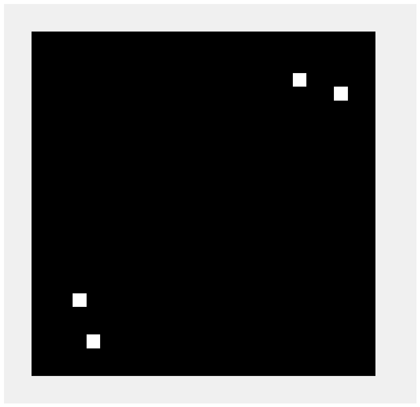
Fig.5. Adjacency matrices of 10x10 and 5x5 versions of the image respectively

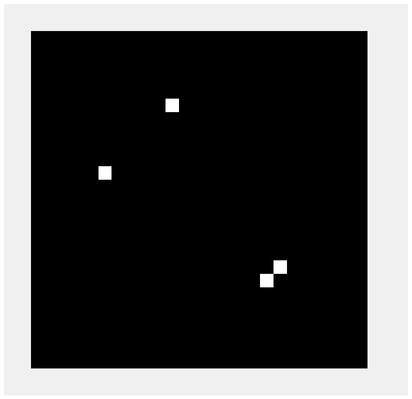
5. CLASSIFICATION OF IMAGES BASED ON THEIR GRAPHS

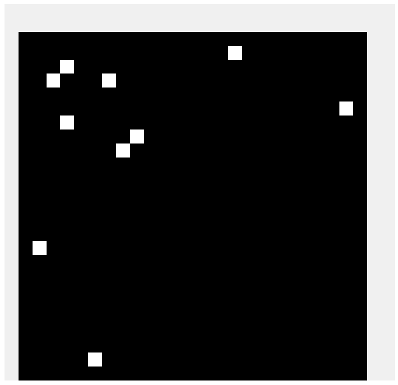
Identifying and classifying images is a crucial topic of interest these days. From the above implementations, it can be inferred that the graph underlying an image can be learnt and it is unique as the image. The graph underlying an image can be used as a means of classifying images. For example, out of several images, an image containing a human can be identified by comparing the graph underlying those images, with a predefined graph which represents an image of a human. This phenomenon can be used in various identification/classification applications.

The corresponding dataset was taken from the INRIA dataset [3] which contains human and non-human images. Some of these images were taken and used for the implementation of classification by graphs.

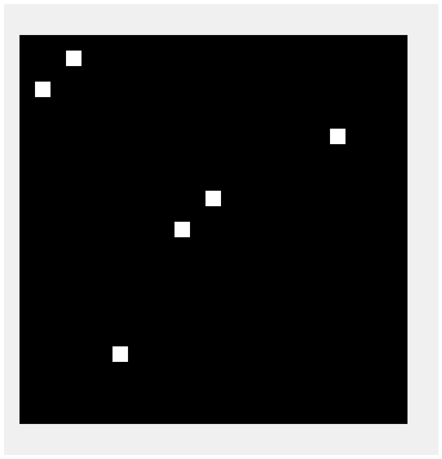
Initially three images of size 5x5 were considered. They are images of a human, a bicycle and a building. Let the images be represented as I1, I2 and I3. The corresponding graphs underlying these images were learnt using the above algorithm. Let the graphs underlying I1, I2 and I3 be represented as G1, G2 and G3. Below are the three images considered and the respective adjacency matrices of their underlying graphs.







After this, some images where each of them had either a human, a bicycle or a building were considered. The graphs of these images were learnt and then compared with the previously learnt graph i.e. G1, G2 and G3. For example the below image of size 5x5 containing a human was considered and its underlying graph was computed.



Let the image be ‘It’ and the underlying graph be ‘Gt’. Now, Gt is being compared with G1, G2 and G3 through the Mean Square Error observed between them.

W.R.T	G1	G2	G3
MSE	0.0224	0.0256	0.0320

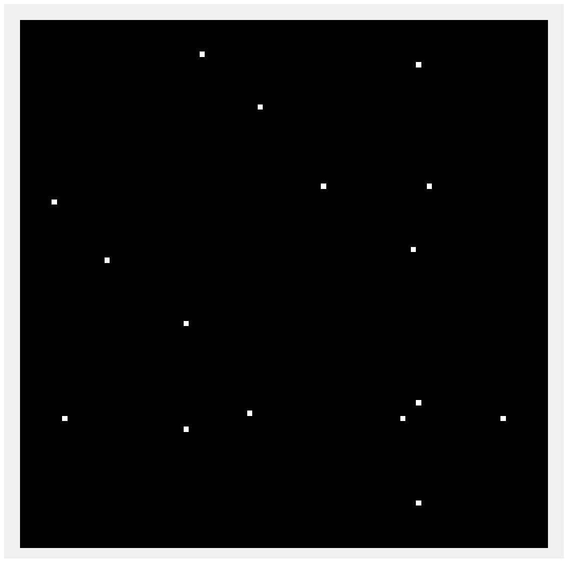
Table 1

It was observed that the error observed between Gt and G1 was the lowest among all the three. This makes us conclude that G1 is the closest graph among G1, G2 and G3 to Gt. We infer from the above result that, Gt i.e. It is an image containing a human. Likewise, images of bicycles and buildings were taken and the system identified them, though inconsistently.

6. LEARNING GRAPHS UNDERLYING DIFFERENT PATCHES OF AN IMAGE

Usually, an image of size XxY can be divided into patches of size AxB where A is a factor of X and B is a factor of Y. This leads us to divide an image into patches of particular size and find the underlying graphs of those patches. This would provide us insights into finding the most important part/patch of an image when an operation such as classification is performed on the image.

For the above purpose, we take the below image of size 16x16 and divide it into patches of size 4x4. The adjacency matrix of the graph observed for the 16x16 image is also given.



Below are the respective adjacency matrices of the four patches of the above image.

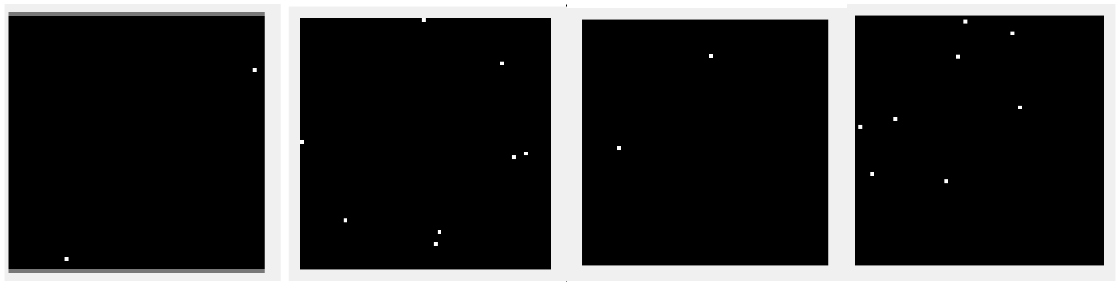


Fig.6. Adjacency matrices of the four patches of the above image.

7. SUMMARY:

In the above project the following were done.

1. Implementing an algorithm to learn the graph underlying an online signal. The error observed between the recovered graph and the original graph was assessed.
2. The above algorithm was further used to learn the graph underlying an image and it's downsized versions.
3. The acquired knowledge and expertise in learning the graph underlying an image from the above works was further used to classify images based on their graph structure.
4. The same classification algorithm was used to identify the patch containing a human face in an image containing a human face along with other details.

8. FUTURE WORK:

1. In continuation to the above image classification problem, the use of the "first non-zero eigenvector" of the graph can be explored.
2. Similarity between graphs can be computed by various methods other than the naive one implemented here

9. REFERENCES:

1]. David I Shuman, Sunil K. Narang, Pascal Frossard, Antonio Ortega, Pierre Vandergheynst, *“The emerging field of signal processing on graphs”*. IEEE signal processing magazine, May 2013.

2]. Stefan Vlaski, Hermina P. Mareti´c, Roula Nassif, Pascal Frossard and Ali H. Sayed*, “Online graph learning from sequential data”*. Institute of Electrical Engineering, EPFL, Lausanne, Switzerland.

3]. J. Shukla, V. K. Chakka, A. S. Reddy and M. Gopal, "*Graph fourier transform based descriptor for gesture classification*," 2017 Fourth International Conference on Image Information Processing *(ICIIP)*, Shimla, 2017.

10. APPENDIX:

FUNCTION TO LEARN THE GRAPH UNDERLYING A SIGNAL/IMAGE:

```
function [adj,L4,wr1] = learnundgraph(I1);

T=1;

%adj = learngraph(I1);
[n1,m1] = size(I1);
N1 = n1*m1;
p1=1;
sr1 = zeros(N1,1);
sc1 = zeros(N1,1);
for i1=1:1:n1              % row wise
    for j1=1:1:m1
    sr1(p1,1) = I1(i1,j1);    % row wise
    p1 = p1+1;
    end
end
```

lms for sr1

```
[Nr1,nr1] = size(sr1);
Sr1 = (eye(Nr1) - ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1))))*sr1;
Rr1 = Sr1*transpose(Sr1);
[eigvecr1,eigvalr1] = eig(Rr1);
ur1 = -(min(min(eigvalr1)));
Wr1 = rand(Nr1,Nr1);
sxr1 = zeros(Nr1,1);
qr1=0;
err1 = zeros(Nr1,1);
for z2=2:1:Nr1

    sxr1(z2-1,1) = sr1(z2-1,1);
    S = (eye(Nr1) - ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1))))*sxr1;  %Si-1
    sxr1(z2,1) = sr1(z2,1);
    S1 = (eye(Nr1) - ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1))))*sxr1;  %Si

    Wr1 = Wr1-ur1*(S1-(Wr1*S))*transpose(S);

            %imposing constraints         %%ux
            Wr1 = Wr1 - ((1/Nr1)*Wr1*ones(Nr1,1)*transpose(ones(Nr1,1)));%nulL
            Wr1 = (1/2)*(Wr1+transpose(Wr1)); %symmetricity
            %[eigvecW,eigvalW] = eig(Wr1);      %spectral bound
            %for i3=1:1:Nr1
            %    if eigvalW(i3,i3)<0
            %        eigvalW(i3,i3) = 0;
            %    end
            %    if eigvalW(i3,i3)>1
            %        eigvalW(i3,i3) = 1;
            %    end
            %end
            %Wr1 = eigvecW*eigvalW*transpose(eigvecW);
            for i2=1:1:Nr1                           %ele
                for j2=1:1:Nr1
                    if Wr1(i2,j2)<(-(1/Nr1))
                        Wr1(i2,j2)=(-(1/Nr1));
                    end
                end
            end

            wr1 = Wr1 + ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1)));

end


[eigc,eiglw] = eig(wr1);
T=1;
[eig1,eig2] = eig(wr1);
[N,~] = size(wr1);
tt = eig2;
for i=1:1:N
    for j=1:1:N
        if i~=j
            tt(i,j)=1;
        end
    end
end
y = log(tt)*(-1/T);
L3 = eig1*y*transpose(eig1);
L4=L3;
for i=1:1:N
for j=1:1:N
L4(i,j)=round(L4(i,j));
if i~=j
if L4(i,j)~=-1   %~=-1
L4(i,j)=0;
end
end
end
end

adj = zeros(N);
for p=1:1:N
    for q=1:1:N
        if p==q
            adj(p,q)=0;
        else
            adj(p,q) = -1*round(L4(p,q));  %round(L4(p,q)
        end
    end
end
```

Published with MATLAB® R2015b

PROGRAM TO LEARN THE GRAPH UNDERLYING A GRAPH SIGNAL AND COMPARE IT WITH THE ORIGINAL GRAPH:

```
clc
clearvars
close all
ip = load('inputSignal.mat');
[N,n] = size(ip.inputSignal);
load('Lab3_minnesota.mat');
L = diag(Problem.A*ones(2642,1))-Problem.A;   %Laplacian Matrix
[eigVec,eigVal] = eig(L); % Eigenvalue and Eigenvector computation
Q = L*((1/sqrt(N))*ones(N)); %constraint_check
eigVal1 = eigVal;
eigVal1(1,1)=0;  %equation_22
T =1; %time period
lambdaw =diag(diag(exp(-T*eigVal))); %eigenvalue of w
w = eigVec*lambdaw*transpose(eigVec);  %eigendecomposition of w_equation (17)

[eig1,eig2] = eig(w);
N=2642;

T=1;


[n1,m1] = size(ip.inputSignal);
N1 = n1*m1;
p1=1;
sr1 = zeros(N1,1);
for i1=1:1:n1              % row wise
    for j1=1:1:m1
    sr1(p1,1) = ip.inputSignal(i1,j1);    % row wise
    p1 = p1+1;
    end
end
```

lms for sr1

```
[Nr1,nr1] = size(sr1);
Sr1 = (eye(Nr1) - ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1))))*sr1;
Rr1 = Sr1*transpose(Sr1);
[eigvecr1,eigvalr1] = eig(Rr1);
ur1 = -(min(min(eigvalr1)));
Wr1 = rand(Nr1,Nr1);
sxr1 = zeros(Nr1,1);
qr1=0;
err1 = zeros(Nr1,1);
mse = zeros(Nr1-1,1);
for z2=2:1:Nr1
```

```
    sxr1(z2-1,1) = sr1(z2-1,1);
    S = (eye(Nr1) - ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1))))*sxr1;  %Si-1
    sxr1(z2,1) = sr1(z2,1);
    S1 = (eye(Nr1) - ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1))))*sxr1;  %Si

    Wr1 = Wr1-ur1*(S1-(Wr1*S))*transpose(S);

            %imposing constraints         %%ux
            Wr1 = Wr1 - ((1/Nr1)*Wr1*ones(Nr1,1)*transpose(ones(Nr1,1)));%nulL
            %[eigvecW,eigvalW] = eig(Wr1);      %spectral bound
            %for i3=1:1:Nr1
            %    if eigvalW(i3,i3)<0
            %        eigvalW(i3,i3) = 0;
            %    end
            %    if eigvalW(i3,i3)>1
            %        eigvalW(i3,i3) = 1;
            %    end
            %end
            %Wr1 = eigvecW*eigvalW*transpose(eigvecW);
            for i2=1:1:Nr1                              %ele
                for j2=1:1:Nr1
                    if Wr1(i2,j2)<(-(1/Nr1))
                        Wr1(i2,j2)=(-(1/Nr1));
                    end
                end
            end
            Wr1 = (1/2)*(Wr1+transpose(Wr1)); %symmetricity

            wr1 = Wr1 + ((1/Nr1)*ones(Nr1,1)*transpose(ones(Nr1,1)));
            mse(z2-1,1) = sum(sum(abs(wr1 - w)).^2)/numel(w);

end
 figure
 plot(mse);

[eigc,eiglw] = eig(wr1);
T=1;
[eig1,eig2] = eig(wr1);
[N11,~] = size(wr1);
tt = eig2;
for i=1:1:N11
    for j=1:1:N11
        if i~=j
            tt(i,j)=1;
        end
    end
end
y = log(tt)*(-1/T);
L3 = eig1*y*transpose(eig1);
L4=L3;
for i=1:1:N11
for j=1:1:N11
L4(i,j)=round(L4(i,j));
if i~=j
if L4(i,j)~=-1   %~=-1
```

```
L4(i,j)=0;
end
end
end
end

adj = zeros(N11);
for p=1:1:N11
    for q=1:1:N11
        if p==q
            adj(p,q)=0;
        else
            adj(p,q) = -1*round(L4(p,q));  %round(L4(p,q)
        end
    end
end

%figure
%imshow(adj);
%figure
%imshow(Problem.A);
pq=1;
for i=1:1:2642
    for j=1:1:2642
        if adj(i,j)==1
            a(pq,1)=i;b(pq,1)=j;
            pq=pq+1;
        end
    end
end
```

Published with MATLAB® R2015b

PROGRAM TO CLASSIFY IMAGES AS HUMAN/NON HUMAN USING GRAPHS:

```
clc
clear all
close all
```

```
dim = 5;
img1 = imread('crop_000001a_60x60.png');
img1 = imresize(img1,(1/60)*dim);
img1 = im2double(img1);
I1 = rgb2gray(img1);
img2 = imread('bike_016_60x60.png');
img2 = imresize(img2,(1/60)*dim);
img2 = im2double(img2);
I2 = rgb2gray(img2);
img3 = imread('00000264a_60x60.png');
img3 = imresize(img3,(1/60)*dim);
img3 = im2double(img3);
```

```
I3 = rgb2gray(img3);
%
imgt = imread('crop_000023b_60x60.png');
imgt = imresize(imgt,(1/60)*dim);
imgt = im2double(imgt);
It = rgb2gray(imgt);
```

```
[adj1,L1,w1] = learnundgraph(I1);
[adj2,L2,w2] = learnundgraph(I2);
[adj3,L3,w3] = learnundgraph(I3);
[adjt,Lt,wt] = learnundgraph(It);
```

```
L1 = real(L1);
L2 = real(L2);
L3 = real(L3);
Lt = real(Lt);
[n,~] = size(L1);
for i=1:1:n
    L1(i,i) = -1*(sum(L1(:,i))-L1(i,i));
    L2(i,i) = -1*(sum(L2(:,i))-L2(i,i));
    L3(i,i) = -1*(sum(L3(:,i))-L3(i,i));
    Lt(i,i) = -1*(sum(Lt(:,i))-Lt(i,i));
end
```

```
            MSEt_1 = sum(sum(abs(adj1 - adjt) .^ 2))/numel(adj1);
            %MSE22_1 = sum(sum((adj1 - adjt) .^ 2))/numel(adj1);
            MSEt_2 = sum(sum(abs(adj2 - adjt) .^ 2))/numel(adj2);
            MSEt_3 = sum(sum(abs(adj3 - adjt) .^ 2))/numel(adj3);
            error = [MSEt_1,MSEt_2,MSEt_3];

            m = min(error);%,MSEt_3);

            if MSEt_1==MSEt_2 && MSEt_2==MSEt_3
                class='null';
            end
            if m==MSEt_1
                class='human';
            else
                class='non human';
            end
```

```
%classeig = classusingeig(adj1,adj2,adj3,adjt);
```

```
d1 = det(adj1);
d2 = det(adj2);
d3 = det(adj3);
d4 = det(adjt);

figure
imshow(adj1);
figure
imshow(adj2);
figure
imshow(adj3);
```

```
figure
imshow(adjt);
```

Published with MATLAB® R2015b