GRAPH SIGNAL PROCESSING: FREQUENCY ANALYSIS

Sresthavadhani,Niteesh

November 2020

1 Introduction

Signal Processing by itself is a very interesting and deep domain. But Standard signal processing techniques are often restricted to problems which have well known signals as inputs and outputs. We know that in our traditional Signal processing we will be able to analyse the graphs in two domains,**time** and **frequency**,hence making many complex problems in time domain a very simple math in frequency domain.

Even though Standard Signal Processing is used in many domains, few domains and major day to day problems remain untouched by it.Here is where search for alternative tools for analysis and processing began. But as mentioned above, the flexibility and ease of use of standard signal processing can't be matched by any other technique. Hence people came up with a way of solving this problem by using **Graph Signal Processing**. After this mode of signal processing was discovered we observed that our standard signal processing is also a subset of Graph Signal Processing using correct notations and conversions.

As we know signals are the object(The one on which analysis is performed) in standard Signal Processing, **Graphs signals** are the analogous objects in Graph Signal Processing. So our main task is to convert our problem into a **Graph** which in turn is converted into a Graph Signal in complex domain.

In the upcoming sections we will look at the basics of Graph Signal Processing and techniques used in it along with few possible applications.

2 Discrete Graph Signal Processing

This is a more general way of optimizing our day to day life problems. Traditional Signal Processing is a part of **Graph Signal Processing**. In Graph signal processing we take a graph signal and analyse it in the frequency domain by performing **Graph Fourier Transform** and also filtering out the signal according to our needs.
In this section We discuss all the properties of Graph signals and ways of analysing them.

2.1 Graph Signals and their Properties

Signal processing on graphs is concerned with the analysis and processing of data-sets in which data elements can be connected to each other with some relational property.
This relation is given through a graph $\mathbf{G} = (\nu, A)$
where $\nu = \{\nu_0, ..., \nu_{N-1}\}$ are the set of nodes of the graph and A is defined as the weighted Adjacency matrix which has the weights between two nodes. The element $A_{n.m} \in \mathcal{C}$ gives the weight between the nodes $\nu_n and\ \nu_m$.
If there exists $A_{n.m}$ for any $n, m \neq 0 \in \mathcal{R}$ then we can say that the nodes ν_n and ν_m are neighbours to each other.

Using this Graph Signal we refer to the dataset of graph signal as a map

$$\mathbf{s} : \mathcal{V} \to \mathcal{C}\nu_n \to s_n$$

Here each dataset s_n is a complex number. We can represent this graph signal as a vector

$$s = [s_0\ s_1.....s_{N-1}]^T \in \mathcal{C}^N$$

2.2 Graph Filters

A graph filter is defined as a system that takes a signal s, processes it and then produces another signal $\tilde{s} = \mathrm{H(s)}$ as output. This system is known as a **Graph Shift** which replaces the signal value at that particular node with the linear combination of signal values of the neighbouring nodes.

$$\widetilde{s_n} = \sum_{m \in \mathcal{N}_n} \mathcal{A}_{n,m} * s_m$$

We can also infer this as

$$\tilde{s} = [\tilde{s_0}.....\tilde{s_{N-1}}]^T = \mathcal{A} * s.$$

This graph shift is the basic building block in $\mathbf{DSP_G}$.
Generally graph filters are polynomials in the Adjacency matrix $\mathcal{A}$.

$$h(\mathcal{A}) = h_0 I + h_1 \mathcal{A} + ... + h_L \mathcal{A}^L$$

So the output for any input signal to the Graph filter is

$$\tilde{s} = \mathcal{H}(s) = h(\mathcal{A}) * s.$$

Now we can define a normalised graph shift matrix as

$$\mathcal{A}^{norm} = \frac{\mathcal{A}}{|\lambda_{max}|}$$

Where λ_{max} *denotes the eigen value of matrix* $\mathcal{A}$ *with largest magnitude. The use of this normalised matrix over the actual matrix is that it prevents excessive scaling of the shifted signal.*

2.3 Graph Fourier Transform

According to standard **Fourier Transform** we convert a time based signal into its frequency domain and then analyse it calling it the **Fourier Basis** of the signal. In $\mathbf{DSP_G}$ *we define what is called a* **Graph Fourier Transform** *which corresponds to Jordan basis of the adjacency matrix* $\mathcal{A}$.
To understand the graph fourier transform we need to go through what is called **Jordan Decomposition**.

2.3.1 Jordan Decomposition

Let us consider an arbitary matrix A $\in \mathcal{C}^{N\times N}$ *which has* $M <=$ N *distinct eigenvalues* $\lambda_0, ...\lambda_{M-1}$. Each of these eigenvalue (λ_m) *has* D_m *corresponding eigenvectors* $v_{m,o}, ...v_{m,D_m-1}$. *where* $0 < m < M$. *Each of these eigenvector* $(v_{m,d})$ *where* $0 < d < D_M - 1$ *satisfy the condition*

$$(\mathcal{A} - \lambda_m \mathcal{I}_N)v_{m,d} = 0.$$

Now each of these D_M *eigenvectors has* $R_{m,d} >= 1$ *generalised eigenvectors* $v_{m,d,0}, ...v_{m,d,R_{m,d}-1}$ *which is called* **Jordan chain**. *Each of these generalised eigenvector* $v_{m,d,r}$ *for* $0 < r < R_{m,d}$ *satisfies the condition*

$$(\mathcal{A} - \lambda_m \mathcal{I}_N)v_{m,d,r} = v_{m,d,r-1}.$$

All the eigenvectors and the corresponding generalised eigenvectors are linearly independent .
Now for each eigenvector $(v_{m,d})$ *and its* **Jordan Chain** *of size* $R_{m,d}$ *we define a* **Jordan Block Matrix** *of size* $R_{m,d} \times R_{m,d}$.

$$J_{R_{m,d}}(\lambda_m) = \begin{bmatrix} \lambda_m & 1 & & & \\ & \lambda_m & . & & \\ & & . & . & \\ & & & . & 1 \\ & & & & \lambda_m \end{bmatrix} \in \mathcal{C}^{R_{m,d}\times R_{m,d}}$$

Now each of these eigenvalues(λ_m) *has* D_m *such Jordan blocks. For each of this eigenvector we collect its Jordan chain into a* $N \times R_{m,d}$*matrix.*

$$\mathcal{V}_{m,d} = \begin{bmatrix} v_{m,d,0} & . & . & . & v_{m,d,R_{m,d-1}} \end{bmatrix}$$

Now we concatenate all $\mathcal{V}_{m,d}$ *blocks into one block matrix*

$$\mathcal{V} = \begin{bmatrix} \mathcal{V}_{0,0} & . & . & . & \mathcal{V}_{M-1,D_{M-1}} \end{bmatrix}$$

Now we define the **Jordan Decomposition** of a matrix A as

$$\mathcal{A} = \mathcal{V}\mathcal{J}\mathcal{V}^{-1}$$

where J is defined as

$$\mathrm{J} = \begin{bmatrix} J_{R_{0,0}}(\lambda_0) & & & & \\ & J_{R_{1,1}}(\lambda_1) & & & \\ & & . & & \\ & & & . & \\ & & & & . \\ & & & & J_{R_{M-1,M-1}}(\lambda_{M-1}) \end{bmatrix}$$

All the eigenvectors and generalised eigenvectos of A are called the **Jordan Basis** of A.

2.3.2 Fourier Transform

For the adjacency matrix A we have the eigenvalues $\lambda_0, \lambda_1, \lambda_2, ...\lambda_{M-1}$ these are known as the **Graph Frequencies** and the generalised eigenvectors for a given eigenvalue are called the **Frequency Components** corresponding to the M^{th} *frequency*.
From the Jordan Decomposition we have

$$\mathcal{A} = \mathcal{V}\mathcal{J}\mathcal{V}^{-1}$$

Hence, the **Graph Fourier Transform** of a signal s is given by

$$\hat{s} = \mathcal{F}s$$

where $\mathrm{F} = \mathrm{V}^{-1}$ *is called the* **Graph Fourier Transform Matrix**
The inverse graph fourier transform is given by

$$s = \mathcal{F}\hat{s}$$

2.4 Total Variation

This is a fundamental concept for many applications of **DSP** like signal regularization,image compression etc...

In classical **DSP** the **Total Variation** of a discrete signal is defined as the sum of magnitudes of differences between two consecutive signal samples.

So we define the **Total Variation** as

$$TV(s) = \sum_{n} |s_n - sn - 1|$$

The Total Variation compares a signal s to its shifted version, if the difference is small then we say that the signal Variation is less. So we can represent this mathematically as,

$$TV(s) = ||s - \mathcal{A}^{norm} s||$$

Here $\mathcal{A}^{norm}$ is the normalized adjacency matrix for the signal s.
If we define the gradient of graph signal as $\nabla_n(s)$ which is given by

$$\nabla_n(s) = s_n - \sum_{m \in \mathcal{N}_n} \mathcal{A}^{norm}_{n,m} s_m$$

The local variation of the signal at vertex v_n is the magnitude $|\nabla_n(s)|$. Now the total variation is give by the p-Dirichlet form

$$S_p(s) = \frac{1}{p} \sum_{n=0}^{N-1} |\nabla_n(s)|^p$$

for p = 1;

$$\begin{aligned} S_1(s) &= \sum_{n=0}^{N-1} |\nabla_n(s)| \\ &= \sum_{n=0}^{N-1} |s_n - \sum_{m \in \mathcal{N}_n} \mathcal{A}^{norm}_{n,m} s_m| \\ &= ||s - \mathcal{A}^{norm} s|| \end{aligned}$$

Total Variation of the graph fourier basis is given by taking the Jordan basis of the adjacency matrix A. Let us consider an eigenvalue of A and let $v = v_0, v_1, ...v_{R-1}$ be the eigenvectors corresponding to the eigenvalue so we have the total variation as

$$TV_G(v) = |1 - \frac{\lambda}{|\lambda_{max}|}|||v||$$

This total variation of a normalized proper eigenvector is a real number between 0 and 2.

3 Frequency Analysis

As we have mentioned several times when handling with large data, frequency always comes to our rescue, Many complex math are way lot easier in frequency domain hence we perform our major analysis in frequency domain. In our report we are considering two sub-parts of Frequency Analysis
1.Frequency Response
2.Graph Filter and their Designing

3.1 Frequency Response

As we have discussed above the graph filters H(.) which take input a signal s and return the signal H(s). Generally this H(.) is a polynomial of $\mathcal{A}$. And from the **Graph Filters** we have

$$\tilde{s} = h(\mathcal{A})s.$$

Let us take $\mathcal{A}$ as the Jordan Decomposition $\mathcal{V}\mathcal{J}\mathcal{V}^{-1}$ then we have

$$\tilde{s} = h(\mathcal{A})s = \mathcal{F}^{-1}h(\mathcal{J})\mathcal{F}s \implies \mathcal{F}\tilde{s} = h(\mathcal{J})\hat{s}$$

Here $h(\mathcal{J})$ is given by

$$h(\mathcal{J}) = \begin{bmatrix} h(J_{r_{0,0}}(\lambda_0)) & & & & \\ & h(J_{r_{1,1}}(\lambda_1)) & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & \cdot \\ & & & & h(J_{r_{M-1,M-1}}(\lambda_{M-1})) \end{bmatrix}$$

This matrix is called the **Graph Frequency Response** of the filter h(A) and is denoted as

$$h(\hat{\mathcal{A}}) = h(\mathcal{J})$$

3.2 Filtering and Filter Design

Similar to traditional Signal Processing, the graph signals when processed with graph filters, show change in frequency content in conjunction to the frequency response of the filter.

Graph filters can be divided into three categories:

1) LOW-PASS FILTER
2) HIGH-PASS FILTER
3) BAND-PASS FILTER

The type of filter is decided by the frequency response produced by the filters.

As we have seen in the previous section our filter is characterised by the matrix h(A).
As mentioned in the paper We also showcase our graph filter using a diagonalizable adjacency matrix for simplicity. The below considered is an Ideal Graph filter.

$$\mathcal{F} * \tilde{s} = \begin{bmatrix} h(\lambda_0) & & & & & \\ & h(\lambda_1) & & & & \\ & & \cdot & & & \\ & & & \cdot & & \\ & & & & \cdot & \\ & & & & & h(\lambda_{M-1}) \end{bmatrix} * \hat{s} \implies \begin{bmatrix} h(\lambda_0) * \hat{s}_0 \\ h(\lambda_1) * \hat{s}_1 \\ h(\lambda_2) * \hat{s}_2 \\ \cdot \\ \cdot \\ \cdot \\ h(\lambda_{M-1}) * \hat{s}_{M-1} \end{bmatrix}$$

Now we construct our $h(\lambda_m)$ in such a way that it tends to 0.
Now considering an example for low pass filter : here we take another matrix $g(\mathcal{A})$ and consider the cut-off frequency to beλ_{cut}. We define the frequency response $h(\mathcal{A})$ That is

$$h(\lambda_m) = \begin{cases} 1 & \lambda_m \geq \lambda_{cut} \\ 0 & \lambda_m \leq \lambda_{cut} \end{cases}$$

Now we do the same for a high pass filter
considering a high pass of the form $g(\lambda_m) = 1 - h(\lambda_m)$

$$g(\lambda_m) = \begin{cases} 0 & \lambda_m \geq \lambda_{cut} \\ 1 & \lambda_m \leq \lambda_{cut} \end{cases}$$

4 Applications

4.1 Image Compression Using Reduction in Colour Space

One of a good applications we could think of when we went through the paper was Image Compression. On further research in Image Compression

we found out there were two situations which could be optimised i.e. 2 situations were Compression could be done using Graph Signal Processing.

Firstly, All of the Image data captured around us is spread in various colourspaces namely RGB, sRGB, Adobe RGB, CMYK etc... each concerning to a specific field. So when these many different colour spaces are used compatibility in different setups become difficult. This is where we can use Graph Signal Processing. When we have hundreds of files sorting and finding out the variations in them plays a vital role. Sorting in time domain can be a lot of hard work ,But frequency domain completely changes the game.

And here is where graph signal processing has its hand, we convert all our colour data into graph signals in complex domain.This conversion can be done and mapped to our need. This complex domain graph signals of components in present colour space are now given as an input to a system where our output is graph signals of components in our required colour space.

Now when we start converting to simpler colour spaces as an immediate by product our image is way more simpler to work on and also takes up less space because it can only capture data to a specific clarity, rather than focusing on the clarity and shooting up our size of the image file.

Let us now consider an example of conversion of a image from CMYK colour space to RGB colour space.
CMYK has 4 components - Cyan,Magenta,Yellow,Black

For each pixel in our image we have 4 values be noted which are the intensities of the 4 components i.e Cyan,Magenta,Yellow and Black.Our next work is to map individual values of the components to their respective graph signals outputting a matrix of tuple of 4 graph signals (or) (n x n x 4) 3-D matrix where n are the no of pixels in the width and breadth if the image respectively.

Each cell of the matrix contains the graph signal corresponding to its specific component in the form $a + ib$ where the real part corresponds to the numbering of the part cell in the 3-D matrix and imaginary part gives the intensity of the corresponding component of the colour space.

The input graph signals are of the form

$$s^x_{i,j} = a^x_{i,j} + i * b^x_{i,j} \ where \ x = C, M, Y, K.$$

Now we this matrix as input to a system with transfer function p which outputs a (n x n x 3) 3-D matrix which on applying Inverse Graph Fourier Transform and reduction of $\hat{s}$ to s back again produces our compressed image.

Here p outputs the graph signals in RGB colour space using the conversions mentioned below but in frequency domain.

- Red = 255 × (1 - Cyan ÷ 100) × (1 - Black ÷ 100)
- Green = 255 × (1 - Magenta ÷ 100) × (1 - Black ÷ 100)
- Blue = 255 × (1 - Yellow ÷ 100) × (1 - Black ÷ 100)

The output graph signals are of the form

$$s'^{x'}_{i,j} = a'^{x'}_{i,j} + i * b'^{x'}_{i,j} \ where \ x' = R, G, B.$$

4.2 Image Compression using variation between pixels in neighborhood

Now we look at compression from a different perspective, Out of the millions of pixels present in an image we start by considering many small neighbourhoods, where we compare the values of the graph signal a node with its neighbouring nodes and calculate the local variation and find out the the values which values repeat many times using the frequency analysis.

We can do this by using a high pass-filter to cut off the places where there is maximum repetition of a single value. We can also set a threshold ϵ where the image quality is compromised considering the fact that our major motive of image compression is to reduce the size of the file rather than preserving the quality of it. Now we give this value as the value for all of them because the value of $'\epsilon'$ is small such that the quality of image is not hindered.

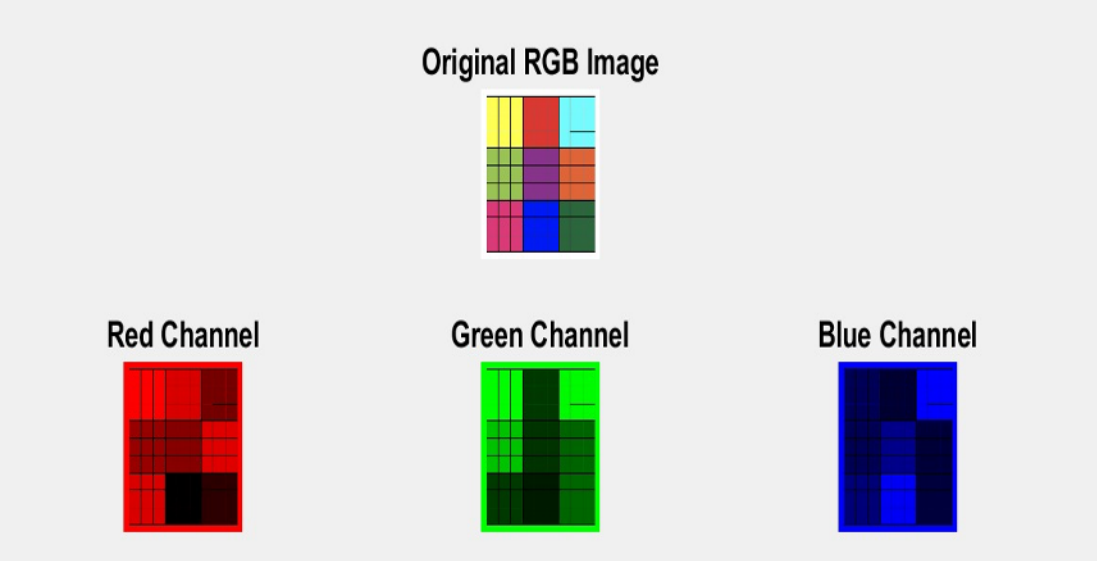
Considering RGB colour space we get three Graph signals, One each for Red,Green and Blue now we take neighbourhood of $'\epsilon'$ pixels in each of the

3 components and calculate our output matrix whose cells are shifted to normalised values in each neighbourhood as mentioned above. Before if we had to store values for 'n' pixels now we only have to store it for n^2/ϵ^2 number of pixels which is a very large number in storing hundreds and thousands of files.

In this method we also have a flexibility of choosing $'\epsilon'$ to our need depending on our preference of image quality (or) file size.

The three graph signals are now subjected to Graph Fourier Transform and we now observe the outputs of each Red, Green and Blue graph signals individually.

an example of the three graph signal outputs is shown below:



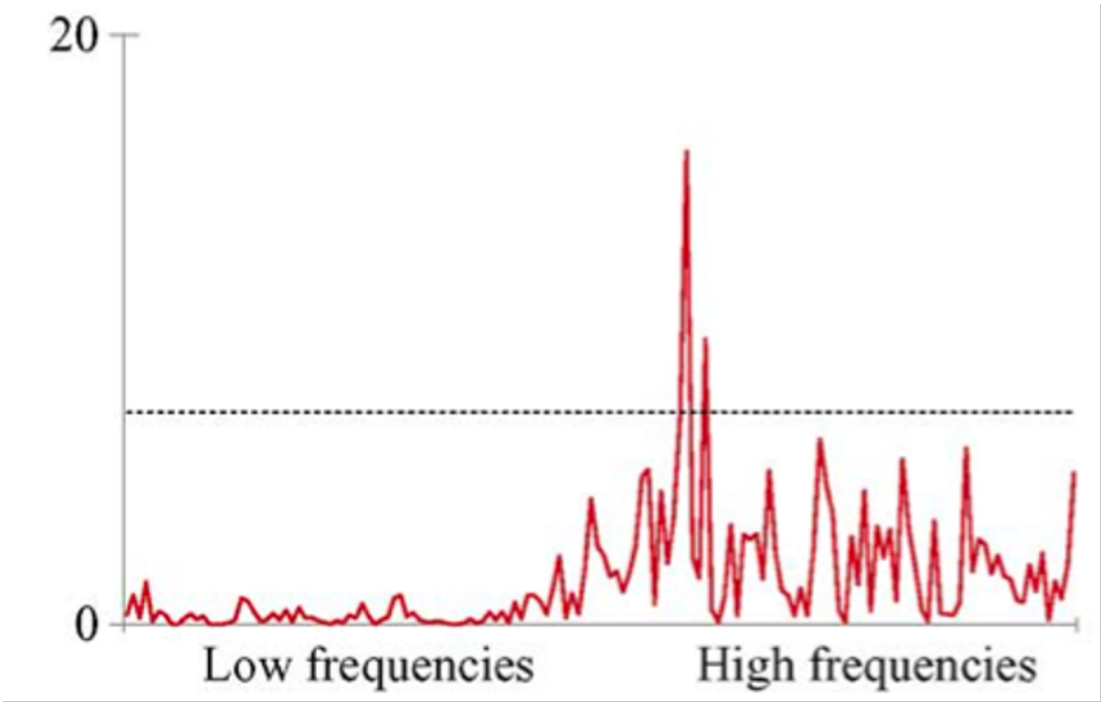
source: `https://www.ginifab.com/feeds/pms/cmyk_to_rgb.php`

We do our neighbourhood analysis on these outputs by considering a small neighbourhood around the pixel and matching all the values to a common colour which are in the specific threshold preset by us.

This basically implies that small variations in the same colour are neglected and the whole neighbourhood is now considered as a single pixel in

the view of efficiency of storage. As mentioned above now the no of pixels is reduced by a factor of ϵ^2 So our new shifted graph signals are now obtained in each neighbourhood. Where the Inverse graph Fourier Transform and inverse mapping to R,G,B values from the S graph signal together give the new R,G,B values for the reduced image.

The frequency graph which is to be filtered is something as shown below:



Here the high frequency components are the over repeated values which are closely spaced on the real line and now we normalise these values to a single value for compression.

5 Conclusion

In this report we discussed how **Discrete time Graph Signal Processing** is done to Graph Signals. We also looked up at the Fourier Transform of Graph signals and derived expressions for them. We discussed about the Filtering of these signals in the frequency domain.

We have taken up an application of Image compression using reduction in colour space and using variation between pixels in neighbourhood.Even though the results are not yet mathematically supported most of the application part is very much used in present day image compression, which educates us on how wide the applications of **Graph Signal Processing** are and how well they can be implemented.

6 References

1.https://ieeexplore.ieee.org/abstract/document/6808520

2.A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph Fourier transform," in Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., 2013, pp. 6167–6170

3.https://en.wikipedia.org/wiki/Manifold#:~:text=In%20mathematics%2C%20a%20manifold%20is%20a%20topological%20space,homeomorphic%20to%20the%20Euclidean%20space%20of%20dimension%20n

4.https://en.wikipedia.org/wiki/Riemannian_geometry

5.https://www.ginifab.com/feeds/pms/cmyk_to_rgb.php

6.https://in.mathworks.com/matlabcentral/answers/91036-how-do-i-split-a-color-imag

7.https://en.wikipedia.org/wiki/RGB_color_model

8.https://scihub.scihubtw.tw/https://www.researchgate.net/publication/317662348_The_use_of_Graph_Fourier_Transform_in_image_processing_a_new_solution_to_classical_problems?channel=doi&linkId=5947c1c3aca27242cda7e194&showFulltext=true

9.https://www.youtube.com/watch?v=3gjJDuCAEQQ

10.https://arxiv.org/pdf/1712.06393.pdf