# PROJECT PLAN
# Image Compression based on Non-Parametric Sampling in Noisy Environments

**Group 19G01: Kishan Narotam (717 931) & Nitesh Nana (720 064)**

*School of Electrical & Information Engineering, University of the Witwatersrand, Private Bag 3, 2050, Johannesburg, South Africa*

**Abstract:**

**Key words:** DCT, Decoding, Encoding, Holes, Image Compression

## 1. INTRODUCTION

A project plan is a formal document that forms a guideline of the project that will ensure the overall success and completion of the project [1]. Planning the project involves looking at three fundamental aspects of the project:

- Scope: what are the objectives of the project and how it will be completed
- Cost: how much money is allocated, budgeted and spent over the course of the project
- Time: the time taken to execute the project to a point of completion [2].

In the report that follows, a project plan is created for the telecommunications project of *image compression based on non-parametric sampling in noisy environments*. Firstly, the project specifications are defined, followed by a brief look into existing models. Subsequently, the proposed strategy for the project is done with a cost and time management which is discussed in order for the project to be completed on time within a specified budget.

## 2. PROJECT SPECIFICATIONS

The aim of the project is to create a robust scheme for determining multiple holes that may be received in an image and appropriately fill these holes. in addition, the image may be subject to random burst errors which need to be identified and corrected.

An image will be chosen for compression utilizing a compression technique and creating holes in the image and after transmitting, filling of those holes will result in the original image. The holes will be created manually making use of current compression algorithms (such as fractal compression for example). Once the holes are created, the image will be transmitted via a simple channel that will introduce errors randomly. The image will be received at the receiver, and knowing where the holes are present, will fill them accordingly using a different algorithm. The receiver will also have to deal with random errors that may have occurred from the channel. The received image with the filled holes will be reconstructed and presented as the final image which should coincide with the original image.

## 3. EXISTING MODELS

Image compression is an application of data compression where an image file is encoded with a few bits with the overall goal of reducing the size of the image file compared to that of the original [3]. There are two main techniques of image compression:

- Lossless
- Lossy

### *3.1 Lossless*

Lossless compression allows the original form of data to be reproduced, thus meaning that the original data from the file before compression is preserved [4]. Some of the current lossless compression techniques include:

- Run-Length encoding
- Huffman encoding
- Shannon-Fano encoding
- Arithmetic
- Dictionary based.

### *3.2 Lossy*

Lossy compression removes some of the data from the original file, resulting in an overall reduction of the size of the file [4]. The non-useful parts of the data that is not noticeable is removed, thus reducing the overall quality of the data and file. Some of the current lossy compression techniques include:

- Lossy predictive
- Vector Quantization
- Transform Coding
- Block transform
- DCT/DWT
- JPEG.

## 4. PROPOSED STRATEGY

The proposed solution will be implemented in MATLAB and can be broken down into two sides:

- Encoder side
- Decoder side.

In conjunction with this, a simple channel will be created and random errors will be introduced after the image is encoded. Figure 1 shows the basic framework and block diagram of the proposed solution that will be implemented.

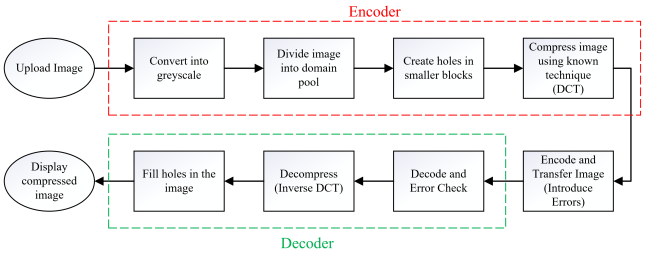


Figure 1 : Block diagram of the proposed strategy

### 4.1 *Step 1: Loading/Reading an Image into MATLAB*

Firstly, an image is loaded into MATLAB, and a PNG or BMP image is specifically chosen with the dimensions $M \times N$, where $M$ and $N$ are divisible by 8. A PNG or BMP image is used as a JPEG image, is already a compressed image via the DCT image compression technique. Figure 2 shows an example of the image that can be used.


Figure 2 : $128 \times 128$ image that will be loaded

### 4.2 *Step 2: Convert the image to greyscale*

The loaded image is converted to greyscale, and the reason for this is because an image that has been imported into MATLAB creates a 3-dimensional array where the first two elements of the array represent the dimensions of the image and the third dimension is the colour map. Converting the image to a greyscale one, creates a 2-dimensional array, which are simply the dimensions of the image, where each array value correlates to the specific integer value of that specific pixel. Figure 3 shows an example of the outcome of converting the image to greyscale.


Figure 3 : Image after it has been greyscaled in MATLAB

### 4.3 *Step 3: Divide the image into the domain pool*

The image is now divided into smaller $8 \times 8$ blocks, creating the domain pool. Once the image has been divided into smaller blocks, we index each block from the top left starting from 1 and increment each index by one, moving left-to-right, top-to-bottom. The reason for creating such an index is so that the receiver will be able to determine which smaller block in the domain pool contains holes so that the receiver can fill these holes. Figure 4 shows an example of the loaded image having a domain pool of 256 blocks.

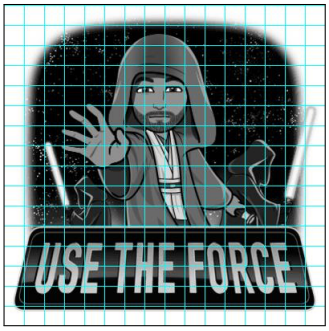


Figure 4 : Smaller $8 \times 8$ blocks creating the domain pool

### 4.4 *Step 4: Create holes in the smaller blocks*

With the domain pool created, holes are created by starting at the center square within the $8 \times 8$ block. Starting off with the center $2 \times 2$ square, the average value of those 4 pixels are calculated. If the average value is within $\pm 5$ *or* 10 of each pixel value, a hole can be created in those 4 pixels. A larger square in the same $8 \times 8$ block is checked with a size of $4 \times 4$. The average of these pixels are calculated and as before compared to each value of the center square. This can continue until we reach a center square size of $6 \times 6$, which we will define as the largest hole that can be made. Figure

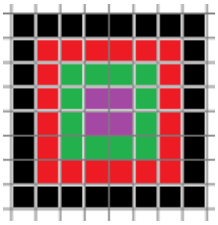

Figure 5 : How a $8 \times 8$ block (black square) will be checked for hole creation starting with the purple square.

### *4.5 Index block if the hole has been created*

If a hole has been created in a specific block in the domain pool, a one-dimensional array will be created with the value of the block index. This array will be transmitted with the image itself, so that when received by the receiver, it will have a list of the blocks in the domain pool that contain a hole.

### *4.6 Step 5: Compress image using a known technique (DCT)*

The DCT compression technique will be utilized for the compression of the image. DCT will be performed on each block in the domain pool, and DCT is done on $8 \times 8$ matrix, which is the reason the domain pool is made up of multiple $8 \times 8$ blocks and that an image where the dimensions are divisible by 8 is chosen. The image is now compressed and can be transmitted into the channel.

### *4.7 Step 6: Encode and Transfer image*

The image is encoded as a binary message, and when transmitted through the channel, errors are randomly introduced. The data that will be sent through the channel will be a bit stream and the errors introduced will be based off of a byte with a probability of $10^{-3}$ where the byte can then be randomized. Alternatively, each bit can be random selected to be prone to an error and the bit value flipped.

### *4.8 Step 7: Decode and Error Check*

Identification of the location of the errors is required. First, identifying the errors are required, and if detected must be corrected and then the image can be decoded. The image will be converted from a bit stream into its 2-dimensional matrix.

### *4.9 Step 8: Decompress (Inverse DCT)*

Since DCT compression was used, we now perform the inverse DCT to obtain the compressed image and its respective values in the 2-dimensional matrix.

### *4.10 Step 9: Fill holes in the image*

With the one-dimensional array sent across the channel, the blocks within the domain pool that contain the holes can be identified. Starting at outer pixels, the average pixel value of the square is calculated, and that will fill the inner pixels where the hole was created.

## 5. COST MANAGEMENT

The nature of this project is software based, and thus the total costs are minimal to none. Licences for the MATLAB software is already provided as a student licence by the university, and personal computers or laptops will be used.

## 6. TIME MANAGEMENT

The Project window runs from 15 July 2019 until 29 August 2019. The proposed timeline of events and duration of these events can be seen in Figure 7. Based on the planned approximated time that is allocated to each task, this indicates that keeping to a tight schedule is required.

## 7. CONCLUSION

## REFERENCES


[1] Techopedia; *What is a Project Plan? - Definition from Techopedia*; `https://www.techopedia.com/definition/24775/project-plan`; Last Accessed: 02/07/2019

[2] Heerkins, G R; *Project Management*; McGraw-Hill; New York, NY, United States; 1st Edition, 2001

[3] Wei-Yi Wei; *An Introduction to Image Compression*; Graduate Institute of Communication Engineering; National Taiwan University; Taipei, Taiwan, ROC

[4] Chapman, C; *Everything You Need to Know About Image Compression — The JotForm Blog*; `https://www.jotform.com/blog/everything-you-need-to-know-about-image-compression/`; Last Accessed: 02/07/2019

| Task Name | Duration | Start Date | Finish Date |
|---|---|---|---|
| **Project Image Compression** | **45 days** | **Mon 19-07-15** | **Fri 19-09-13** |
| **Initiating** | **5 days** | **Mon 19-07-15** | **Fri 19-07-19** |
| **Research** | **5 days** | **Mon 19-07-15** | **Fri 19-07-19** |
| Literature Review | 5 days | Mon 19-07-15 | Fri 19-07-19 |
| **Execution** | **28 days** | **Mon 19-07-22** | **Wed 19-08-28** |
| **Prototype** | **7 days** | **Mon 19-07-22** | **Tue 19-07-30** |
| Design hole-implementing algorithm | 7 days | Mon 19-07-22 | Tue 19-07-30 |
| Set up channel | 4 days | Mon 19-07-22 | Thu 19-07-25 |
| **Test Prototype** | **3 days** | **Wed 19-07-31** | **Fri 19-08-02** |
| Test algorithm | 3 days | Wed 19-07-31 | Fri 19-08-02 |
| Test channel | 3 days | Fri 19-07-26 | Tue 19-07-30 |
| **Final Build** | **18 days** | **Mon 19-08-05** | **Wed 19-08-28** |
| Introduce errors | 3 days | Mon 19-08-05 | Wed 19-08-07 |
| Handle errors | 6 days | Thu 19-08-08 | Thu 19-08-15 |
| Test error handling | 3 days | Fri 19-08-16 | Tue 19-08-20 |
| Transmit image with errors and error correction | 3 days | Wed 19-08-21 | Fri 19-08-23 |
| Test various cases | 2 days | Fri 19-08-23 | Mon 19-08-26 |
| Debugging | 3 days | Mon 19-08-26 | Wed 19-08-28 |
| **Monitoring & Control** | **34 days** | **Mon 19-07-15** | **Thu 19-08-29** |
| Progress Reports | 34 days | Mon 19-07-15 | Thu 19-08-29 |
| **Closing** | **12 days** | **Thu 19-08-29** | **Fri 19-09-13** |
| Final Project Report | 7 days | Thu 19-08-29 | Fri 19-09-06 |
| Project Presentation | 5 days | Mon 19-09-09 | Fri 19-09-13 |
| Open day (Build Deadline) | 1 day | Thu 19-08-29 | Thu 19-08-29 |

Figure 6 : Table showing the tasks and time allocated in completing the project

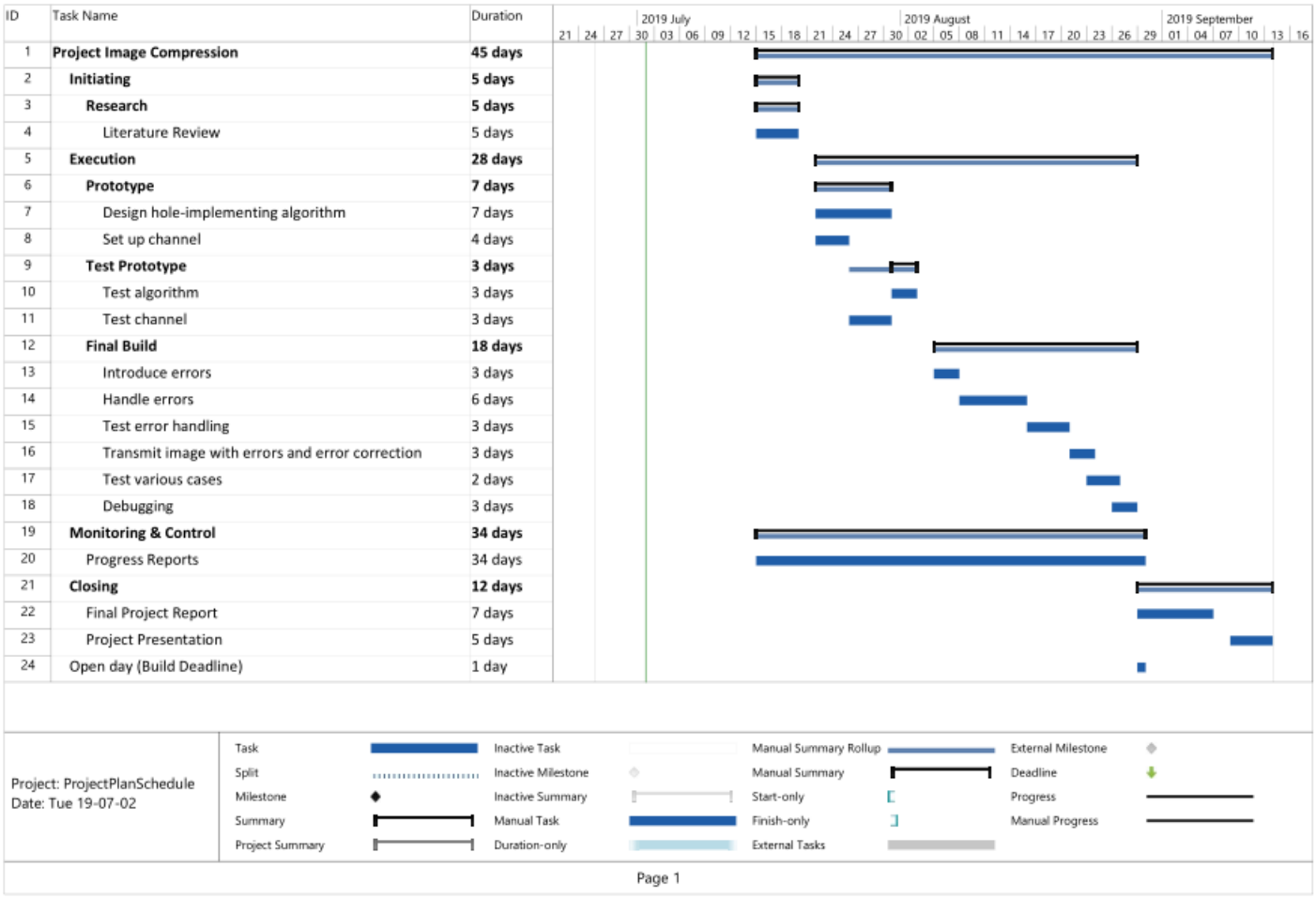


Figure 7 : Gantt chart showing the tasks and time allocated in completing the project