

# RICE SORTER

*The final capstone report submitted in fulfilment of the requirement for the award of the degree of*

Bachelor of Engineering

in

Electronics and Communication Engineering

Submitted By

LAKSHAY AGGARWAL (101506091)

PULKIT BAHL (101506131)

RAGHAV SHARMA (101506132)

RAHUL GARG (101506133)

RAJAN BANSAL (101506135)

SAKET PANCHORI (101506145)

Under Supervision of

**Dr. Vinay Kumar**

Assistant Professor

Electronics and Communication Engineering Department



THAPAR INSTITUTE  
OF ENGINEERING & TECHNOLOGY  
(Deemed to be University)

Department of Electronics and Communication Engineering

THAPAR INSTITUTE OF ENGINEERING & TECHNOLOGY, PATIALA, PUNJAB

December, 2018

## CERTIFICATE

This is to certify that the project report on, “*Rice Sorter*” being submitted by **Mr. Lakshay Aggarwal, Mr. Pulkit Bahl, Mr. Raghav Sharma, Mr. Rahul Garg, Mr. Rajan Bansal and Mr. Saket Panchori** to our mentor **Dr. Vinay Kumar**, Thapar Institute of Engineering and Technology, Patiala for the fulfillment of the course requirement of **Capstone (UEC797)** is a bonafide record of work carried out by us in conformity with the rules and regulations of the institute. The results presented in this report have not been submitted, in part or full, to any other University or Institute for the award of any degree or diploma.

Lakshay Aggarwal

(101506091)

Dated:14/12/2018

Pulkit Bahl

(101506131)

Raghav Sharma

(101506132)

Rahul Garg

(101506133)

Rajan Bansal

(101506135)

Saket Panchori

(101506145)

## **ACKNOWLEDGEMENT**

We would like to convey our gratefulness to the Department of Electronics and Communication Engineering at Thapar Institute of Engineering & Technology, Patiala for providing such a good platform for learning and letting us create the capstone project to transform our theoretical concepts into the practical project.

A special thanks to our mentor Dr. Vinay Kumar for helping and supporting us through all the ups and down alongside the project. We would like to thank our head of the department Dr. Alpana Agarwal for providing us all the relevant resources and important information regarding all the difficulty faced during the project.

## **ABSTRACT**

This study explores the pattern to separate out rice grains according to the sizes. Due to system complexity and huge research in the field of signal processing, the principle and implementation of a system of rice sorter is proposed. The system refers to automatically sorting the heterochromatic particles out of the batch of raw rice, removing such impurities in this process improves the quality of the rice. Improvement in food safety due to the removal of foreign materials is the primary aim of the proposed study. This study leads to a simple and hassle-free method for sorting out rice grains which will prove to be a boost for market. This whole concept is based on MATLAB software including various concepts of image processing. The method revolves around the scanning of rice and then implementation of sorting techniques in MATLAB software. Major issues for the system are to free noise from the scanned images which is overcome by passing the scanned image through a number of filters. This system proved to be accurate for small sample space, although for large sample space its accuracy was decreased by 14.08 %. Further for large sample space, numerous iterations are to be made for improved results.

## TABLE OF CONTENTS

Sr. No.	Name of Chapter	Page No.
	<i>Abstract</i>	(iv)
	<i>Chapter 1</i> Introduction.....	1
	1.1 Purpose.....	1
	1.2 History and current industrial applications.....	1
	1.3 Use of MATLAB software.....	2
	1.4 Quality of raw rice .....	2
	1.5 Potential of Food industry .....	2
	<i>Chapter 2</i> Literature Survey.....	3
	<i>Chapter 3</i> Flow Chart.....	7
	3.1 Explanation of the Flowchart.....	8
	<i>Chapter 4</i> Project Design.....	9
	4.1 Specifications.....	9
	4.2 Noise factor.....	15
	4.3 Method of cleaning.....	16
	4.4 Area estimation for counting purpose.....	16
	4.5 Border counting .....	17
	4.6 Analysis of data.....	21
	4.7 Sorting Algorithm.....	21
	4.8 Code.....	22
	4.9 Courses Referred.....	27
	4.10 IEEE Standards Used.....	28
	<i>Chapter 5</i> Results.....	29
	5.1 Observations and Results.....	29
	5.2 Efficiency obtained.....	32
	<i>Chapter 6</i> Outcomes and Prospective Learning.....	33
	6.1 Outcomes.....	33

6.2 Prospective Learning.....	33
6.3 Student Outcomes.....	34
6.3.1 Outcomes achieved till the end of present semester...	34
6.3.2 Learning outcomes from capstone project.....	37
<i>Chapter 7</i> Project Timeline.....	38
7.1 Time Schedule for the Project.....	38
References.....	41

## LIST OF FIGURES

<b>Sr. No.</b>	<b>Figure Details</b>	<b>Page no.</b>
<i>Figure 3.1</i>	Overall working of project	7
<i>Figure 4.1.1</i>	Image showing coarse grain	10
<i>Figure 4.1.2</i>	Image showing fine grain	10
<i>Figure 4.1.3</i>	Image at medium contrast	11
<i>Figure 4.1.4</i>	Image at low contrast	11
<i>Figure 4.1.5</i>	Image at high contrast	11
<i>Figure 4.1.6</i>	Image before circular filter	12
<i>Figure 4.1.7</i>	Image after circular filter	12
<i>Figure 4.1.8</i>	Output1	13
<i>Figure 4.1.9</i>	Output2	13
<i>Figure 4.1.10</i>	Output3	13
<i>Figure 4.1.11</i>	Input image for HLS	13
<i>Figure 4.1.12</i>	Output1 HLS	14
<i>Figure 4.1.13</i>	Output2 HLS	14
<i>Figure 4.1.14</i>	Output3 HLS	14
<i>Figure 4.4.1</i>	Image for area estimation	16
<i>Figure 4.5.1</i>	Linked rice	18
<i>Figure 4.5.2</i>	Grayscale Image	19
<i>Figure 4.5.3</i>	White borders	19
<i>Figure 5.1.1</i>	Input Image 1	29
<i>Figure 5.1.2</i>	MATLAB Output 1	29
<i>Figure 5.1.3</i>	Input Image 2	30
<i>Figure 5.1.4</i>	MATLAB Output 2	30
<i>Figure 5.1.5</i>	Input Image 3	31
<i>Figure 5.1.6</i>	MATLAB Output 3	31
<i>Figure 7.1.1</i>	Gantt chart for Lakshay Aggarwal	38
<i>Figure 7.1.2</i>	Gantt chart for Pulkit Bahl	38
<i>Figure 7.1.3</i>	Gantt chart for Raghav Sharma	39
<i>Figure 7.1.4</i>	Gantt chart for Rahul Garg	39
<i>Figure 7.1.5</i>	Gantt chart for Rajan Bansal	39
<i>Figure 7.1.6</i>	Gantt chart for Saket Panchori	40
<i>Figure 7.1.7</i>	Gantt chart for team	40

# **CHAPTER 1 INTRODUCTION**

## **1.1 PURPOSE**

The purpose of this project is to prepare an efficient algorithm to sorting of rice grains according to the color differences in raw rice. The raw rice available in the market consists of anomalies in its texture, size, shape etc. This algorithm generally focuses on the technique of sorting on the basis of the size. This system focused on the improvement in the quality of the rice available to the customers in the market. Although there exist some conventional methods for this task but automation is the need of the time. This system refers to automatically sorting the heterochromatic particles out of batch of raw rice. On removal of such impurities quality of rice will be improved which will further improve the food safety.

## **1.2 HISTORY AND CURRENT INDUSTRIAL APPLICATIONS**

One of the first applications of digital images was in the newspaper industry, when pictures were first sent by submarine cable between London and New York. Introduction of the Bart lane cable picture transmission system in the early 1920s reduced the time required to transport a picture across the Atlantic from more than a week to less than three hours. Specialized printing equipment coded pictures for cable transmission and then reconstructed them at the receiving end” (Gonzalez, Woods) [2]. Digital Image Processing (DIP) was developed during the 1960's with Jet Propulsion Laboratory, MIT, Bell Labs, UM Maryland and other universities at the cutting edge. The most progressive work was used to advance the analysis of satellite imagery, photo enhancement, medical imagery, wire photo standards conversation while the cost of computer processing was still relatively high. With the rise of faster, cheaper, smaller micro-chips; real time processing gave birth to television standards conversation. DIP has become the most prolific and cheapest processing method used throughout industry. There are many typical problems of DIP; among those are: compression; reduce redundancy in the image data in order to store or transmit more efficiently; pattern Recognition; identifying common attributes in an image or group of images; reconstruction; predicting possibilities of a damaged image; and Morphology; Identifying structure in an image.



### **1.3 USE OF MATLAB SOFTWARE**

This project is based on a multi-paradigm numerical computing environment i.e. MATLAB. It is a proprietary programming language developed by MathWorks. MATLAB allows a wide range of tasks as well as functions to be implemented. It includes matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces and interfacing with programs written in other languages, including C, C++, C#, Java, Fortran and Python. Basically, the image that we are providing is convert into a matrix form on uploading it to software. Conversion to matrix form makes it easy to manipulate the data and operate it to provide efficient output.

### **1.4 QUALITY OF RAW RICE**

Rice available in the market contains anomalies in its texture, size, color, shape etc. Although at the time of harvesting the major anomalies are removed but the left is difficult to handle. A lot of manpower is wasted in this task for which automation is needed. Especially in the Indian markets there are various types of rice available that includes long grain, regular long grain white rice, brown whole grain, basmati, jasmine , japonica etc. Generally, these rice differs in terms of their sizes. This is the main reason for the process to be concentrated on the sorting on the basis of size.

### **1.5 POTENTIAL OF FOOD INDUSTRY**

Food industry especially the Indian food industry is dependent on the labor up to a higher extent. This reduces the efficiency of the industry. There is a high demand of automation in this industry. This is the main reason of our project to be focused on this type of industry. The sorting of rice provides a boost to this industry. Sorting removes the anomalies in the rice while improves the quality of rice. Due to improvement in the quality of rice, better prices are introduced. This technique sorts the rice according to size.

## CHAPTER 2 LITERATURE SURVEY

- **“IMAGE PROCESSING IN FREQUENCY DOMAIN USING MATLAB: A STUDY FOR BEGINNERS” by Vinay Kumar, Manas Nanda**

Vinay Kumar and Manas Nanda have introduced some basic concepts for image processing for beginners. The basic concept underlying in their Filter Design is that using simple 2-D FFT and IFFT approach we can restrict the frequencies according to our choice. It depends entirely on the user to mold the filter specifications and allow a particular range of frequencies from the origin to pass through and observe the result obtained as an image. Also, the side squares approach helps us to find a solution where minor (sometimes imperceptible) loss of fidelity is acceptable to achieve a substantial reduction in bit rate and some of the finer details in the image can be sacrificed for the sake of saving a little more bandwidth or storage space.

- **“Automated Defect Recognition Method by Using Digital Image Processing” by Sangwook Lee, Ph.D. Texas Tech University Lubbock, Texas**

Sangwook Lee proposed an idea about Automated defect recognition method using Digital Image Processing. The image defect recognition method was developed by making pairwise comparisons and calculating eigenvalues which were chosen as a key feature to distinguish defective images from non-defective images. This was realized by taking the following three stages: image acquisition, image processing, and data analysis. In the image acquisition stage, bridge painting digital images were acquired and prepared to generate two types of data sets: defective and non-defective. In the image processing stage, a pair-wise comparison was performed to generate eigenvalues. Some limitations this research work identified need to be addressed. Digital image processing is an effective tool to assess external conditions of a facility. However, there is a limitation to examine internal conditions. In case internal conditions of a structure are in question, additional technology should be considered. Also, this research work was performed to propose a generic methodology to detect bridge coating defects and present testing results. In order to put this technique into practice, more comprehensive field testing is required. It would be better to work with DOT personnel to obtain a right to access more steel bridges and take more field images. By doing this, the validity of this methodology can be enhanced.

- **“On Detection of Median Filtering in Digital Images” by Matthias Kirchner and Jessica Fridrich**

Matthias Kirchner and Jessica Fridrich investigated the detection of median filtering in digital images. In the broader framework of digital image forensics, median filtering is a contribution to the problem of determining the general processing history of digital images. While the application of ‘classical’ image processing primitives for denoising, sharpening, or contrast enhancement does typically not harm the authentic value of an image, it is still of high interest to learn as much as possible about what exactly has happened to an image and to make informed decisions based on this knowledge. Such knowledge is desirable not only in forensics but also in steg analysis and watermarking. For uncompressed images, the analysis of the so-called streaking artifacts in median filtered images has proven to be a reliable measure for discriminating between filtered and unfiltered images. In the case of median pre-compression (i.e., median filtering of already JPEG compressed images), the variation of the eigen-values between different original images is reduced to some extent. After the second compression, the SPAM features are still able to detect median filtering reliably. In fact, a low pre-compression quality can even increase the detector’s performance. As to the limitations, JPEG compression does a good job in obfuscating the actual type of smoothing applied to the image before compression. While being generally well-detectable with the SPAM features, experiments showed that, contrary to the analysis of streaking artifacts in uncompressed images, it is not possible to distinguish between the median filter and other smoothers. While this could also be turned into an advantage by considering the SPAM features as a general-purpose smoothing detector, alternative or additional features should be explored that allow to track down further particularities of the median filter.

- **“FPGA: An Efficient and Promising Platform for Real-Time Image Processing Applications” by Sparsh Mittal, Saket Gupta and S. Dasgupta**

Sparsh Mittal, Saket Gupta and S. Dasgupta provided us with relative information. FPGA (Field Programmable Gate Array) consists of an array of uncommitted elements that can be programmed or interconnected (or configured) according to a user’s specification in a virtually limitless number of ways. Being reprogrammable and easily upgradable, an FPGA offers a compromise between the flexibility of general-purpose processors and the hardware-based speed of ASICs. With a multibillion-dollar market per year, increases in FPGA speeds and capacities have followed or exceeded Moore’s law for the last several years. They conducted a survey which demonstrated the outstanding features of FPGAs which make

them seem very promising choice for the researchers in the field. FPGAs are great fits for video and image processing applications, such as broadcast infrastructure, medical imaging, HD videoconferencing, video surveillance, and military imaging. The greater future potential lies in including FPGAs on-chip with the main processor, giving the benefit of general-purpose acceleration without the communication bottleneck created by placing the FPGA in a co-processor.

- **“Sphericity, shape factor, and convexity measurement of coarse aggregate for concrete using digital image processing” by C.F. Mora, A.K.H Kwan**

The DIP method has been applied to measure the flakiness ratio, elongation ratio, sphericity, shape factor, convexity ratio and fullness ratio of aggregate particles. Although some of these shape parameters are dependent on the thickness of the particles, which is not directly obtainable by DIP, the method previously developed by the authors of supplementing the DIP results by the weight of the aggregate sample can be used to estimate the thickness of the particles to allow measurement of shape parameters dependent on thickness. The shape parameter of an aggregate sample may be taken either as the arithmetic mean or weighted mean value of the shape parameters of the individual particles. While evaluation of arithmetic mean is straightforward, evaluation of weighted mean involves the volume of the particles, which again is not directly obtainable by DIP. Nevertheless, using the method previously developed by the authors for estimating the volume of the particles, the weighted mean value may be determined. It is found that the weighted mean values of the shape parameters generally correlate better to the packing density of the particles than the arithmetic mean values. Correlation of the various shape parameters measured by DIP to the traditional measure of angularity reveals that the flakiness ratio, sphericity, shape factor, convexity ratio and fullness ratio have fairly high correlation with the traditional measure of angularity. Among them, only the convexity ratio and fullness ratio, which are closely related to roundness and angularity, may be taken as measures of angularity. However, as the sharpness of the corners is not considered in their measurement, the convexity ratio and fullness ratio cannot be treated as complete measures of angularity. For a complete picture of angularity, additional measurement on the sharpness of the corners is required. Lastly, it is advocated that the present concept of angularity number should be abandoned. The angularity number as measured by the traditional method is, in reality, a measure of the packing density of the aggregate particles, which may be dependent also on other shape attributes apart from angularity.

- **“Particle shape analysis of coarse aggregate using digital image processing” by A.K.H. Kwan C.F. Mora H.C. Chan**

A method of analyzing the flakiness and elongation of coarse aggregates using DIP technique is developed. Although the thickness and volume of the particles are not measured, this DIP method is capable of producing the mean thickness/breadth ratio of the aggregate and shape measurement results in terms of mass fractions. For the purpose of verification, the shape measurement results obtained by the proposed DIP method have been compared to those obtained by traditional mechanical sieving and manual gauging. Strong correlation between the parameter (mean thickness-to-breadth ratio) derived by the DIP method and the flakiness index measured by the traditional method has been achieved. On the other hand, the elongation indexes obtained by the DIP method agree very closely with those by the traditional method. After such comparison, however, it is suggested that we need not follow the existing definitions of flakiness and elongation indexes, which are suffering from several shortcomings as discussed previously. The parameter  $\lambda$ , which is easily obtainable by the DIP method, can replace the flakiness index. For replacing the elongation index, a new weighted mean elongation ratio, which is believed to be a better measure of elongation, is developed.

- **“Particle size distribution analysis of coarse aggregate using digital image processing” by C.F. Mora A.K.H. Kwan H.C. Chan**

A method of analyzing the particle size distribution of coarse aggregates using DIP technique is developed. Unlike other DIP methods, which yield only results in terms of area gradation, the method developed herein has the capability of converting the area gradation to mass gradation, provided all particles of the aggregate sample are from the same source. During the process of converting area gradation to mass gradation, a shape parameter  $\lambda$ , which may be taken as a measure of flakiness, is generated and hence flakiness is also measured. Good agreement is achieved when the DIP results are compared to those by conventional mechanical sieving. Apart from being fast, convenient, and versatile, the DIP method has the major advantage that it yields a lot more information than the mechanical sieving method. In fact, from the same captured image of the aggregate, detailed shape analysis of the particles may also be carried out. However, the DIP method has the difficulty that the particles must be carefully spread out to avoid touching or overlapping. There is also the limitation that the particles must not appear to be too small in the pixel image or otherwise their pixel representations would be inaccurate.

## CHAPTER 3 FLOW CHART

In this chapter, we will be discussing the following things

1. Signal flow in the entire project system
2. Processing of the signal at each block.
3. Inputs given to each block
4. Outputs from each block

Each arrow represents an activity flow and the block represents the process

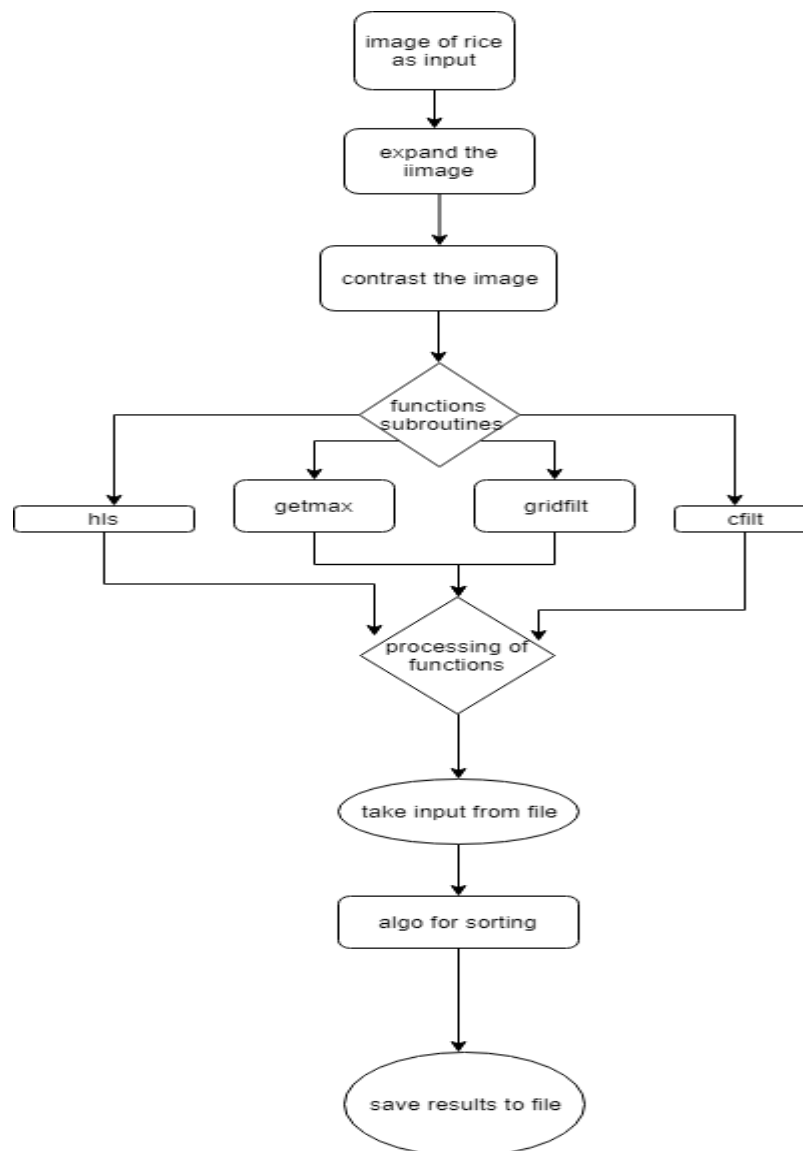


Figure 3.1 Overall working of the project

### 3.1 EXPLANATION OF THE FLOWCHART

The first 5 lines of the program establish the function and use MATLAB image processing workshops inherent algorithms for taking in an image, gray-scaling it, and changing it in a way to make it usable in MATLAB, subroutines that will be explain in further detail below,

The first is ‘expand’ which adds the black border around the image necessary for processing it with our grid and circular filters.

The second is ‘contrast’ which takes the image and processes it into black and white.

The third, ‘getmax’, looks at the image, given the radius size we’ve designated ( $r=5$ ), and counts how many pixels qualify within an  $2r+1$  size grid, for later comparison.

Next, the ‘cfilt’ subroutine runs the image through our circular filter using the given radius size.

‘HLS’, as the name implies, runs the HLS counter once on the image after running through the circular filter, for an initial count.

The ‘gridfilt’ subroutine runs the image through, in this case, 2 cycles of grid filtering. The algorithm then gets a new count on each cycle and, bearing the result doesn’t vary too far from the running average, adds it into consideration for a final average. The average is then calculated and, as you can see, there is an option to show the result in either exact or rounded results, for aesthetic reasons, depending on what result is desired by simply uncommenting the “total...” line.

The ‘gridfilt’ subroutine takes in the original image, it’s size, the desired grid size for the filter to use, and the circular filter radius. The radius is only taken into account so that the program is not run on anything but the original image, excluding the black border, which is based on that radius size. A second matrix is created exactly the same as the given one, then the algorithm looks pixel by pixel at the original image. For those pixels, it looks at the surround area using the given grid size, and counts the number of white pixels. If the grid doesn’t meet the requirements (here we look for all white), then the pixel is turned black. We are using two matrices/images in order to make changes without disturbing the original image, which would affect the algorithm as it runs.

## CHAPTER 4 PROJECT DESIGN

In this chapter, we go through the design procedure that will be followed in the project and the reasoning as to why we used that design technique.

### 4.1 SPECIFICATIONS

#### **Direct Contrast Filter**

We have used to select the first filter for RGB to Grey conversion. MATLAB program can be used to convert the images in the grayscale format. Basically, there are two reasons for using this logic First reason is there is a three-dimensional matrix of RGB which decide the color of the image, now we are grey scaling image in order to solve our problem.

Second reason when there will be two colors in the end on which we have to process the rice count that is we are having just two colors which can be distinguished easily and will be further discussed.

One notable thing we have to make our camera in apposition such that we have not having any shadow of our image so that there is no wrong image or shadow of camera on it.

Also, there must be apt light in the room so that there is no mismatching /blurring of the images, if the light in the room is more than there will be more contrast in the image hence the result will be spoilt.

Now talking about the threshold values for our image, if our value is 0 then pixel is said to be black and the if value is near to 255 then the pixel is black.

Once we had a value, a simple algorithm was enough to go through every pixel in the image and change the value to 0 (black) if it was below the given threshold, and 255 (white) if it was at or above that same threshold, effectively turning the gray-scale image into a black and white image, though it was technically still in gray-scale. This was far from perfect, however, since there were often points in the background that were not rice, yet were above the threshold. This irrelevant data still needed to be removed. This is where the grid and circular filters came into play.

#### **Grid Filter**

The grid filter is the first filter we created to take advantage of the black-and-white filtered images we were now capable of producing. The basic idea of the grid filter is to go through



the image pixel by pixel and examine a grid or matrix surrounding that pixel to attempt to determine if it is internal or external to a piece of rice. However, if we want to examine *every* pixel in the image, we need be able to examine a grid around the corner of the image. This poses an obvious problem, since there is nothing up and to the left of the upper-left corner's pixel, for example. To get around this, we included a series of

commands that created a new “empty” zero matrix that was  $\frac{(S-1)}{2}$  longer on the top, bottom, left, and right of the image, where  $s$  is the length of one side of the grid we intend to use.

Next, we reinsert the original matrix/image into the new one, centered inside of it. This results in our original image with a black border drawn around the outside of it large enough to allow us to run the algorithm over the entirety of the original image.

Once we have a bordered image, the grid filter algorithm runs, starting at what was the original upper-left corner of the image, and goes pixel by pixel along the first row, then each subsequent row, until it reaches the bottom-right corner of the image. If it finds a white pixel, it looks at the grid surrounding it, then turns it black if there are any black pixels in that grid, otherwise leaving it white. This has the effect of leaving the “core” of the rice, while getting rid of any excess data. While very effective, it could sometimes be too destructive, splitting a rice into multiple rice, particularly if the rice has any black pixels within it. This brings up again the importance of quality images, and an accurate filter contrast. Below is an example of the filter used on an early image:



Fig 4.1.1 Image showing coarse grain



Fig 4.1.2 image showing fined grain

### **Strel Filter**

This filter will be discussed little here, as it is not our own work. We mention this MathWorks created filter because we used it as a temporary measure during the middle portion of the project, where our own filters seemingly weren't able to filter out enough excess data for our counting algorithms to run, and we needed a way to test them. The filter essentially dilated and erodes the image in such a manner as to filter out the "background", although even in this industry grade filter, the size of the filter used and consistent images were key to a good filtered image. It was more effective than our direct contrast filter by itself, and can choose its own contrast threshold.

This filter fell out of use when we finally had images that were of high quality, and could use our own filters exclusively. Nonetheless, it did allow us to work in parallel on both our filters and our counting algorithms to accomplish more within the given three months allowed for this project, buying invaluable time. Its effect is shown below:



Fig 4.1.3 Image at low contrast

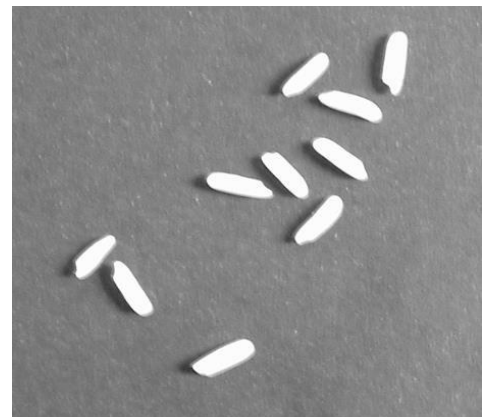


Fig 4.1.4 Image at low contrast



Fig 4.1.5 Image at high contrast

### **Circular Filter**

This is our final individual filter. It functions under the same principle as the Grid Filter, but looks only at a circular area within each grid, and allows for a few black pixels to be included, preventing us from “splitting” the rice in half, as the grid filter sometimes did, as well as better fitting the rice, then a square filter, due to the inherent shape of the rice. This prevented excessive destruction for the most part, giving much better images. The circular filter, used on its own, gave the below result:



Fig 4.1.6 Image before circular filter



Fig 4.1.7 Image after circular filter

As you can see in these image leaves more the rice intact than did the Grid filter, however, you can also see that a few spots of irrelevant data qualify to remain in the image. Thus is why our final filter involves the use of all three of our created filters, first gray-scaling and using the direct contrast filter, then using the Circular filter to get the above image, then using the grid filter on a much lower level than it has been used previously, getting rid of those last few pieces of erroneous data that would reap havoc in our counting algorithms, the final result of which can be seen at the top of the next page:



Fig 4.1.8 Output 1



Fig 4.1.9 Output 2

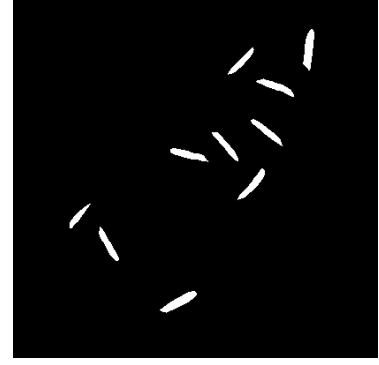


Fig 4.1.10 Output 3

As arduous as all of this cleaning and filtering was, it paled in comparison to the other half of our ultimate goal, the counting algorithms.

### **Horizontal Layered Scanning (HLS)**

The bright idea that saved the day is Horizontal Layered Scanning, named after its methodology and abbreviated HLS hereafter (if IBM or Scientific Atlanta can do it, so can we!). The basic idea is to look at each row as a whole, tracking how many rices we pass through, and comparing it with the row before. When the algorithm notes that it is counting less rice than before, it makes the assumption that it has completed passing through a grain of rice. For example, let's look at the rice shown in the image below:



Fig 4.1.11 Input Image for HLS

Feel free to refer to the image on the top of the next page as we move along. As we start moving down the image there is nothing but black emptiness, so it reads  $[0, 0]$ , meaning 0 rice in the row, 0 rice completed. Once it reaches the row where the upper-right rice grain is locating, it reads  $[1, 0]$ , meaning one rice in the row, and none completed, the same when it reaches the second rice grain on the left  $[2, 0]$ . Before it reaches the final rice grain however it first detects the end of one rice grain, counting  $[1,1]$ ; one rice grain in row, one completed.

A few rows later it finishes the second rice grain [2, 0]. This continues through the final rice grain, with us first counting the grain in the row [1, 2], then it's completion [0, 3]. Once the algorithm completes the last row in the image, it outputs the final count of three rice grains. This method is highly efficient and effective, particularly when compared to the overwhelming complexity of the border programs. The inherent flaw with HLS comes into play when you have one rice grain complete in a row *immediately* before a new rice starts in the next. In its current state, the program would see it as the same rice, and the count of rice would drop by one in comparison to the actual number of rice grains in the image. We need to compensate for this somehow. As the problem lies with the rice positioning, and we are free to change the size grain itself, as our solution we use several iterations of the grid destructive filter so that the rice which cause the problem are figuratively “pulled apart” as shown in the images below:

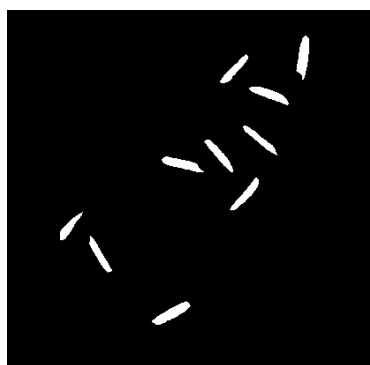


Fig 4.1.12 Output1 HLS



Fig 4.1.13 Output2 HLS



Fig 4.1.14 Output3 HLS

We then take an average of the counts provided to get a more accurate count, removing any numbers that highly deviate from the rest as a precaution. This can happen in a case where an image is too thoroughly destroyed, and rice grains is split into several large dots, which would each then be counted as its own rice, artificially increasing the count.

A few ideas of alternate methods to improve efficiency come to mind. The first is to devise a method of tracking the rice “centers”. In this case it means the center horizontally in a row, for each rice. If done properly, it would allow the algorithm to discern when a rice is continuing, or if it is, in reality, a pair of rice, one ending just before the other begins. Let's take a look at the resulting accuracy using this method with both the industry grade filter by MathWorks and our own circular filter.

## 4.2 NOISE FACTOR

We discovered very early on that one of our most critical tasks would be to take consistent photos in lighting, distance from target objects, and zoom level on the camera being used. This is absolutely critical as all three of the counting algorithms, and two of the three cleaning methods require that the objects be of a consistent size throughout the series of photos. The only inherent problem at first seemed to be the shadows, as you can see a bit on the right-hand image above. We compensated for this by taking two bright (and very hot, I might add) bendable light sources and focused them in the sides. This worked well for the first month or so, until we needed to photograph larger numbers of rice. We discovered that the structure simply wasn't high enough for us to get them all in the image without bundling them all together. On top of that, the images we did take showed varying brightness as will be discussed in the cleaning section, partially due to the fact that light was coming from the sides, and the more rice we had, the more shadows we blocked out some of that light, creating internal shadows.

The second method was a bit cruder, but was more effective for our purposes. We simply took a thick string used for architecture design and tied it tightly to the top of two chairs in the dining room, laying our background paper and rice underneath it on the table and resting the camera against the string to take the photos. This allowed for ample light and no shadows. Multiple photos had to be taken to compensate for human error, as movement would blur the images, and roughly one in three images would be clear enough to submit into the program. The only difference was in that we had to change our background color.

Back when we first used the Lego construction to take images, we thought to try different background colors to see if one would be more effective. We tried hues of black, green, blue, red, orange, and pink. The most effective, as we suspected from watching behind-the-scenes movie creation on DVDs, was a dark green. The black seemed to reflect back too much of the light from a flash or high-level light source. We did, however, discover that once we changed our method of photography into the dining room, we needed to change to black. The green background was still effective, but the black was now more effective than the green. We can only contemplate that it has something to do with natural versus artificial lighting on the paper we were using at the time.

### 4.3METHODS OF CLEANING

For cleaning of the image, the following filters were used: -

1. Direct Contrast filter
2. Grid filter
3. Strel filter
4. Circular filter

### 4.4AREA ESTIMATION FOR COUNTING PURPOSE

The basic idea behind this counting technique is to approximate the number of rice present in an image by utilizing the area that a single rice grain takes up in that image {noted by the variable ( $R_s$ )}. We also need to know how much area of the image is rice; in other words, how many pixels in the image have a value corresponding to a rice grain {noted by the variable ( $P_r$ )}. Once the single rice grain area is determined with the units of pixels; it is simple division to approximate how many rice grains are present in the image {noted by the variable ( $N_r$ )}.  $P_r / R_s = N_r$  (1)

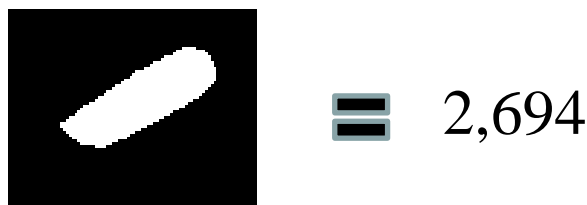


Fig 4.4.1 Image for area estimation

There are many intricacies involved during translation from theory into the Matrix Laboratory (MATLAB) construct. First, we must devise a method to count ( $R_s$ ); the number of pixels associated with a single rice grain. Few grains are identical so we need to find out an average pixel value for many rice grains. To do so we analyze many pictures of individual rice grains. All of the images were cleaned and gray scaled with a threshold to produce a binary image with values of 0 and 255; the latter is the pixel value associated with a grain and 0 is the pixel value of the background. This reduces the task to counting all pixels with the value of 255. This program is not complicated; refer to the algorithms section at the end.

The result is a constant of  $R_s = 2,694$  average pixels in one rice grain. The same program can be reused to count  $P_r$  for any image with any number of rice grains present. We then slightly manipulate the output by inserting formula (1) to get the Area Estimation ( $N_r$ ) of how many rice grains are present in an image.

Unfortunately, there are a few drawbacks to using Area Estimation. The first is obvious; it is an estimate. The only true counting takes place at the pixel level on a micro scale. The average of that information is then used to formulate the number of rice on a macro scale. The result is seldom ever an integer number of rice. Another negative attribute is the inability to use destructive filtering to clean the image. The destructive filter drastically changes the value for the average pixel number of an individual rice grain ( $R_s$ ). The amount of destruction varies for each picture thus the  $R_s$  constant is misleading when used in conjunction with destructive filtering. Furthermore; the image is best cleaned with a destructive filter otherwise background clutter is not all removed. The gray scaled binary manipulation reduces the clutter but all inconsistencies above the threshold remain. The Area Estimation method then counts all of the surviving inconsistencies as a pixel associated with a rice grain; hence inflating the  $P_r$  value and consequently the final rice count ( $N_r$ ) is also inflated. The only way to curtail this inflation is to sample the average clutter per area of background ( $C$ ), multiply by the total background area in the image, and then subtract that from the total number of pixels associated with rice ( $P_r$ ) before dividing by the single rice constant  $R_s$ . This is an inferior counting method with many flaws but it does have some appeal. The advantages will become obvious upon comparison with the other more complex counting methods and algorithms. Area Estimation is easily understood and implemented. Understanding the program and how it works allows for quick trouble shooting. Also; the more complex the program the longer it takes to run; as evident in our Border counting technique.

## **4.5 BORDER COUNTING**

Now we've reached to point at which new previously unknown heights of stress were discovered. The basic idea of our border counting algorithm was to define the rice borders and then utilize those borders in counting the rice. Like most tasks, we discovered, it was far simpler in theory than in practice.



The first task at hand, as usual was to have a clean photo. However, as any border counting algorithm would have little bearing on the actual size of the rice grains, we have much more freedom with respect to how roughly we can clean the image. Therefore, full use of destructive filters is perfectly permissible. The most important thing is to minimize irrelevant data, and minimize the number of times the subroutines are run unnecessarily.

The second task is to define the borders themselves. Originally, we tried to create an algorithm that would follow the border with the rice grains intact, then run a secondary routine to empty the interior. However, we started running into problems where two rice grains were intact. Due to the fact that our original algorithm scanned the “cone” immediately ahead of it before spreading out it passed straight through, creating major issues, as shown below:

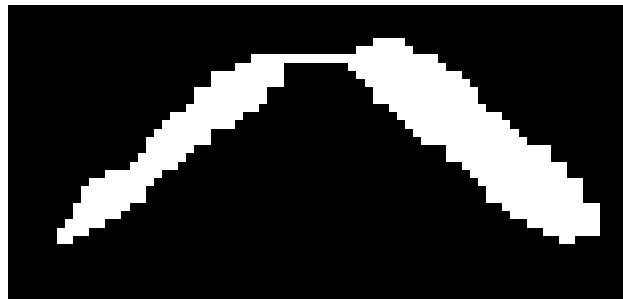


Fig 4.5.1 Linked rice

In effect, this caused the algorithm to “see” and then draw borders around (and into) the same rice, multiple times. One particular example repeated within the same rice over one hundred times! Obviously, we still have some work to do.

After numerous attempts at the previous method, a second approach surfaced. Instead of following the border first, why not find a way to empty out the interiors first, giving the program a “track” to follow, preventing it from digging straight through. The first attempt was far more complex than it needed to be. It involved searching row by row and when it hit a white pixel, it cleared out all but the last white pixel that occurred consecutively in that row. It worked, but eventually a simpler method was devised. If we simply pass through each row and tag any white pixel next to a black one with a different number, say 100, instead of the usual 255 (the value for white in a grayscale image, also the maximum value), then do the same for each column, it makes our job much simpler.

All we need to do then is change any value that isn't "tagged" at 100 to 0, changing it black, then change all "tagged" pixels to 255, effectively removing the rice interiors, and raising the borders in white, as shown below:



Fig 4.5.2 Grayscale image

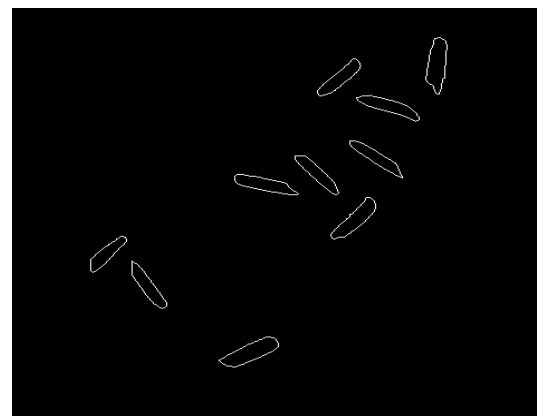


Fig 4.5.3 White borders

Okay, we have our borders selected, now we have to find a way to follow our predefined granular "rail". This particular part of the project caused us both great angsts, and hit our wallets as well. At first it didn't seem too bad, we used a cropped image of a single rice, shrunk down for debugging purposes. After two attempts that ended up in a total dumping of all progress and restarting from scratch, we had one that seemed to work, following a single rice border on our example. In essence, it passing row by row until it hit the first white pixel in the image, then marked it by changed the value to 102, then ran the border following subroutine, changing each border pixel, as it found them, to values of 101, so that the algorithm itself could recognize where it had already been, proceeding until it tracked back around to the pixel marked as 102, then returning to the original program, which would continue searching for any border. As they are normally set at 100, the program would pass right over any border that had already been followed and marked as 101 or 102. When it reached the bottom of the image, it would output the number of rices counted. We even had the border program count the length of the borders, so that we could filter out any irrelevant data that had been missed in cleaning, as they would show very small border lengths.

However, even once we altered the algorithm to follow twisting borders, such as the image of two rice touching that we saw a few pages ago, the program would lock up Citrix, the school-provided internet MATLAB program, on a regular basis. This, in and of itself, would cause problems, because it would halt progress on the project until we

could get it running again, sometimes for more than a week. It had even come to the point that we had to buy a copy, as well as the image processing package, so we can work on it at home, or face dire academic consequences out of sheer lack of time. It took over a month, and four *complete* rewrites of the program before we discovered that it wasn't our loops that were the problem. We, along with teachers and fellow programmers we spoke to, had been assuming that with all of our "if" "for" and particularly "while" loops, that we simply had created an infinite loop. It took a typo to discover the true problem. A missing statement, causing a single rice grain border to count as 246 rice, forced us to look deeper, because it was working and, once fixed, it locked the computer again.

What we discovered was this: when the program ran, even on just a single grain of rice, it had to go through an average of four or five loops before it moved to the next pixel. When we used the old white pixel counting algorithm from our Area Estimation days to count the number of pixels in that defined border, we realized that an *unmodified, full size* rice grain from our images could have well in excess of 2,500 pixels along its border. Our program hasn't been running infinitely, just seemingly so. It was trying to run at least 10,000 loops *per grain of rice*! Just to confirm this, we tried several much smaller images, and discovered that once we used an image beyond about 200 by 200 pixels, the algorithm simply became overwhelmed. Now if we compare this to a full-size image containing one hundred rice, that brings us in excess of one million loops that it has to run through, not to mention the countless non-applicable "if" statements it had to compare to in earlier versions of the algorithm.

At this point comes a very hard decision, continue and try to bring efficiency out of an algorithm that is grossly over-running itself, with no idea as to how to smooth out the wrinkles, or the painful choice of scrapping in excess of a month's work and start with a fresh idea. We chose the latter. Even so, it was a week or two before we came up with a fresh idea.

The negative side of this border approach is painfully obvious. The difficulty in tracking the border is a monumental task that seemed so simple at first, made even more so by our fledgling knowledge in MATLAB. On the positive side, it allows for easy secondary filtering, once the borders are being counted, by tracking border length as each border is counted. Also, it should be noted that through all the difficulties, new knowledge and understanding of MATLAB was born. This, I believe, ultimately led to our success.

## **4.6 ANALYSIS OF DATA**

Taking an overall average of the data gives a 85.92% accuracy using our algorithms. It wasn't until the very end that we were able to produce a filter that, used in tandem with our grid filter and contrast cleaning method, gave comparable results, independent of all third-party algorithms.

Some trends also show within in both sets of data. The most notable of these is the slow, but steady drop in accuracy as the number of rice increases. This is, of course, due to the inherent flaw of HLS as stated previously. Given more time, we would understandably like to expand our statistical database, including far more images per sample size, and test into the thousands, instead of stopping at one hundred rice, something that would have taken quite a while, given we counted all rice by hand for every single image.

## **4.7 SORTING ALGORITHM**

We have made block /partitions of pixels in a image. We have to just check whether the rice is of the standard size or not, this is our main idea for performing this action in that loop. If the rice is above the threshold or just merely equal to the threshold then its output is termed as the fair rice and if the rice doesn't suffice the threshold size criterion then it is termed as a false/small rice which cannot be accommodated in the rest of full-length rice.

The code working in this loop is as follows: firstly, the per block pixel dimension of the image, this task was performed by the trial and error approximately there were pixels in a block of image. Now if there are some scattering of black particles in the image then this will be removed by the earlier mentioned filters. The erratic rice image sample is depicted using the 0 in the output. If per block size of the image is checked again then this result remains the same because the operation performed by us are of fixed threshold values. The time constraint in this problem is to perform the calculation in the certain time stipulation such that output must be computed by code in less than time as compared to the time of fall of the rice on the holes in the bottom of the machine. Hence the algorithm for computations of the rice size must be less than the time of fall, this condition is main condition before all condition has to be met in order to get the job done.

## 4.8 CODE

```
function finalct(p)
jpegFiles = dir('*.jpeg');
numfiles = length(jpegFiles);
mydata = cell(1, numfiles);
r=5;
filename = 'testfile.xlsx';
for k = 1:numfiles
    I = imread(jpegFiles(k).name);
    I=rgb2gray(I);
    I=double(I);
    I=expand(I,r);
    I=contrast(I,206);
    [m n]=size(I);
    maxct=getmax(r);
    I=cfilt(I,m,n,r,maxct);
    count=HLS(I,m,n);
    average=count;
    cycle=1;
    White_pix=0;
    Black_pix=0;
    ocunter = 1;
    a = [0 0 0 0 0 0 0 0];
    kkkk=1;
    kkxx=200;
    tttt=1;
    for zz=1:9
        if kkxx<1964
            for j=kkkk:kkxx
                for i=1:m
                    if I(i,j)==255
                        White_pix=White_pix+1;
                    else
                        Black_pix=Black_pix+1;
                    end
                end
            end
        end
    end
end
```

```

        end
    end
    tttt=j;
end
end
kkkk=tttt;
kkxx=tttt+200;
if(White_pix < 3000)
a(ocunter) =0;
else
    a(ocunter) = 1;
end
White_pix;
Black_pix;
White_pix=0;
    Black_pix=0;
ocunter=ocunter+1;
end
% White_pix
% Black_pix
ssssu=jpegFiles(k).name
sw=cellstr(ssssu);
a
Y = sprintf('A%d', k);
xlswrite(filename,a,'sheet1',sprintf('A%d',k));
xlswrite(filename,sw,'sheet1',sprintf('J%d',k));
end
for grid=3:2:5
X=I;
X=gridfilt(I,grid,m,n,r);
count=HLS(X,m,n);
if cycle==1
average=average+count;
cycle=cycle+1;

```

```

end
if (cycle>1)&&(abs((average/cycle)-count)<=((average/cycle)/3))
    average=average+count;
    cycle=cycle+1;
end
end
average=average/cycle;
% total=round(average);
sprintf('The image contains %d rice',average)
function B=gridfilt(A,grid,m,n,r)
B=A;
g=(grid-1)/2;
for x=1+r:m-r
    for y=1+r:n-r
        gridct=0;
        for u=-g:1:g
            for v=-g:1:g
                if (A(x+u,y+v)==255)
                    gridct=gridct+1;
                end
            end
        end
        if gridct<(grid^2)
            B(x,y)=0;
        end
    end
end
end
function ct=HLS(A,m,n)
ct=0;
previous=0;
current=0;
for x=1:m
    current=0;
    for y=1:n
        if (A(x,y)==255)&&(A(x,y-1)==0)

```

```

current=current+1;
end
    if (A(x,y)==255)&&(A(x,y+1)==0)
        current=current+1;
    end
end
current=current/2;
if current<previous
    ct=ct+(previous-current);
end
    previous=current;
end
function B=cfilt(A,m,n,r,maxct)
B=A;
for i=1+r:m-r
    for j=1+r:n-r
        if A(i,j)==255
            ct=getct(A,i,j,r);
            if ct>=(maxct*0.8)
                B(i,j)=255;
            else
                B(i,j)=0;
            end
        end
    end
end
end
function count=getct(X,i,j,r)
count=0;
for x=-r:1:r
    for y=-r:1:r
        if round(sqrt((x^2)+(y^2)))<=r
            if X(i+x,j+y)==255
                count=count+1;
            end
        end
    end
end

```



```

        end
    end
end
end
function count=getmax(r)
count=0;
for a=-r:1:r
    for b=-r:1:r
        if round(sqrt((a^2)+(b^2)))<=r
            count=count+1;
        end
    end
end
end
function B=expand(A,r)
[m n]=size(A);
B=zeros(m+2*r,n+2*r);
for i=1:m
    for j=1:n
        B(i+r,j+r)=A(i,j);
    end
end
end
function B=contrast(A,cv)
[m n]=size(A);
B=A;
for i=1:m
    for j=1:n
        if A(i,j)>=cv
            B(i,j)=255;
        else
            B(i,j)=0;
        end
    end
end
end
end

```

## 4.9 COURSES REFERRED

### ☐ SIGNAL AND SYSTEM (UEC 404)

Studying signal and system we understood and learned the basic concepts of MATLAB which will be the fundamental for our project implementation. We have learnt the basics of the MATLAB and its basic functions. Scope of MATLAB is very vast and this was again polished in our next course.

### ☐ DIGITAL SYSTEM PROCESSING (UEC 404)

In our project we have to segregate the pixels of rice and its background i.e. we have to count black and white pixels in the image and then estimate the no of faulty rice and then segregate them. For it we have used the concepts of MATLAB learned in the above-mentioned course.

### ☐ EMBEDDED SYSTEM (UEC 502)

In this course we have learned embedded c which is useful for our project. As MATLAB coding is closely related to embedded C so we have used the concepts learned in this course. This course taught us to link hardware to software, this will further be used in our upcoming days to Link our software to hardware of rice sorter machine.

### ☐ IMAGING PROCESSING (UCS 773)

This subject is basically the backbone of our project we tried to explore the various the facets of image processing in order to get our job done. Image processing concepts are used to start our project, e.g. conversion of RGB to grey code, building a matrix of pixels in order to have a count of number of pixels in the data and also removing any scattering of waste images and converting them into a uniform image and then counting the number of rice in the image.

### ☐ INNOVATION AND ENTREPREURSHIP (UTA 012)

As we know that this sort of machines is not available in India, main manufacturers are Korea and china and placed around 20 lakhs INR. But can have the potential for manufacturing and tying up with some Indian firm and sell at less prices. Hence this project gives us immense opportunity for entrepreneurship.

### ☐ PROFFESIONAL COMMUNICATION (UHU 003)

In order to present our idea, we have to have the skills of professional communication. We are reacquired with the skills that we require to present our idea in lucid yet attractive way so that anyone relishes it.

## **4.10 IEEE STANDARDS USED**

IEEE standards that have been used in the project

- 4.10.1 610.4-1990 - IEEE Standard Glossary of Image Processing and Pattern Recognition Terminology.
- 4.10.2 208-1995 - IEEE Standard on Video Techniques: Measurement of Resolution of Camera Systems, 1993 Techniques.
- 4.10.3 1858-2016 - IEEE Standard for Camera Phone Image Quality

## CHAPTER 5 RESULTS

### 5.1 Observations and Results

#### Sample Data Set and Output No.- 1

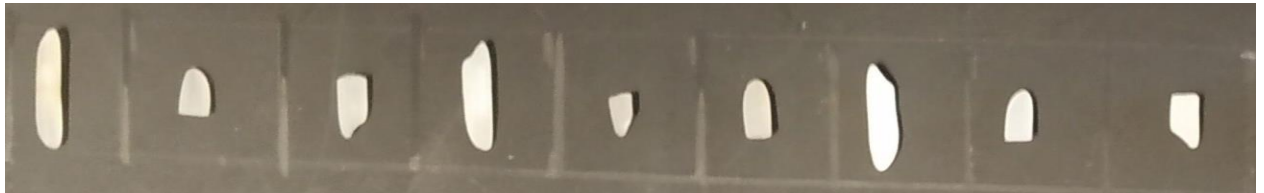
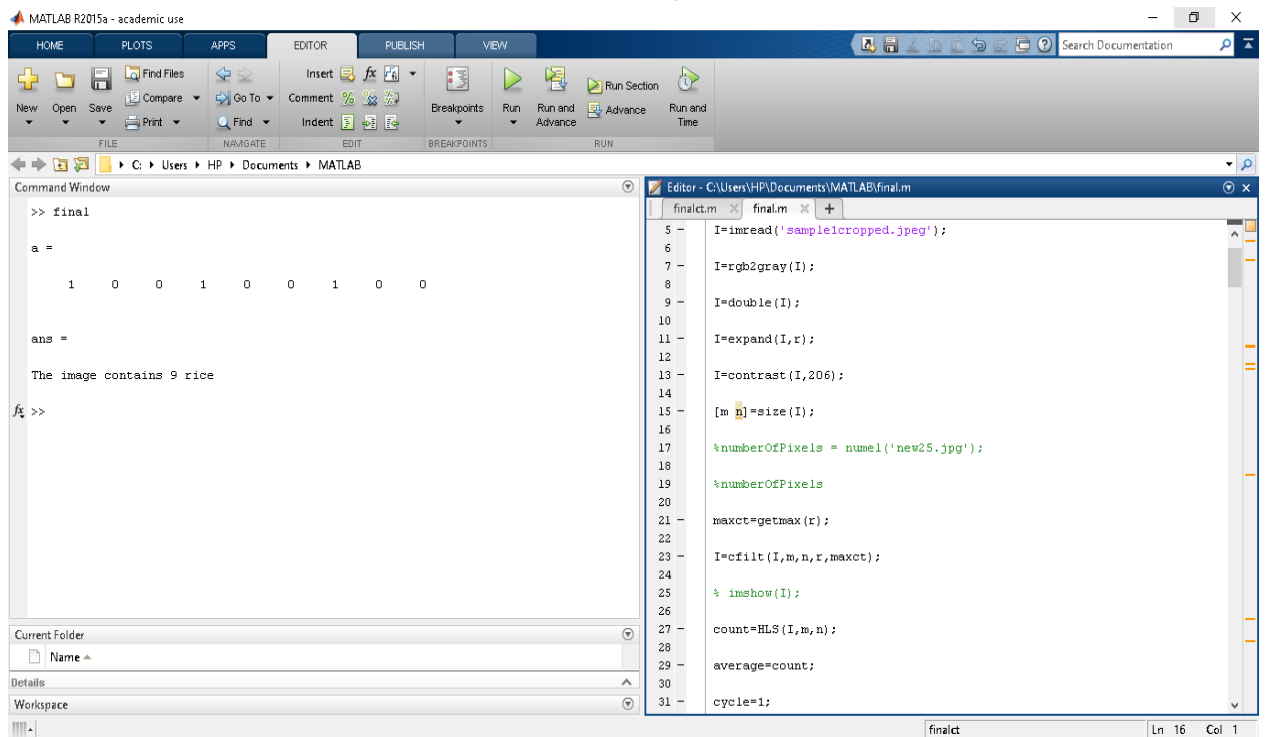
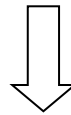


Fig. 5.1.1 Input Image 1



```
Command Window
>> final
a =
    1     0     0     1     0     0     1     0     0
ans =
The image contains 9 rice
fx >>

Editor - C:\Users\HP\Documents\MATLAB\final.m
5 - I=imread('sample1cropped.jpeg');
6 -
7 - I=rgb2gray(I);
8 -
9 - I=double(I);
10 -
11 - I=expand(I,r);
12 -
13 - I=contrast(I,206);
14 -
15 - [m n]=size(I);
16 -
17 - %numberOfPixels = numel('new25.jpg');
18 -
19 - %numberOfPixels
20 -
21 - maxct=getmax(r);
22 -
23 - I=cfilt(I,m,n,r,maxct);
24 -
25 - % imshow(I);
26 -
27 - count=HLS(I,m,n);
28 -
29 - average=count;
30 -
31 - cycle=1;
```

Fig. 5.1.2 MATLAB Output 1

## Sample Image and Output No.- 2

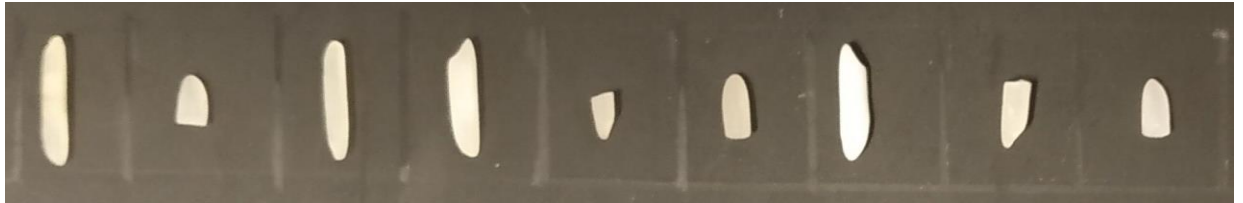
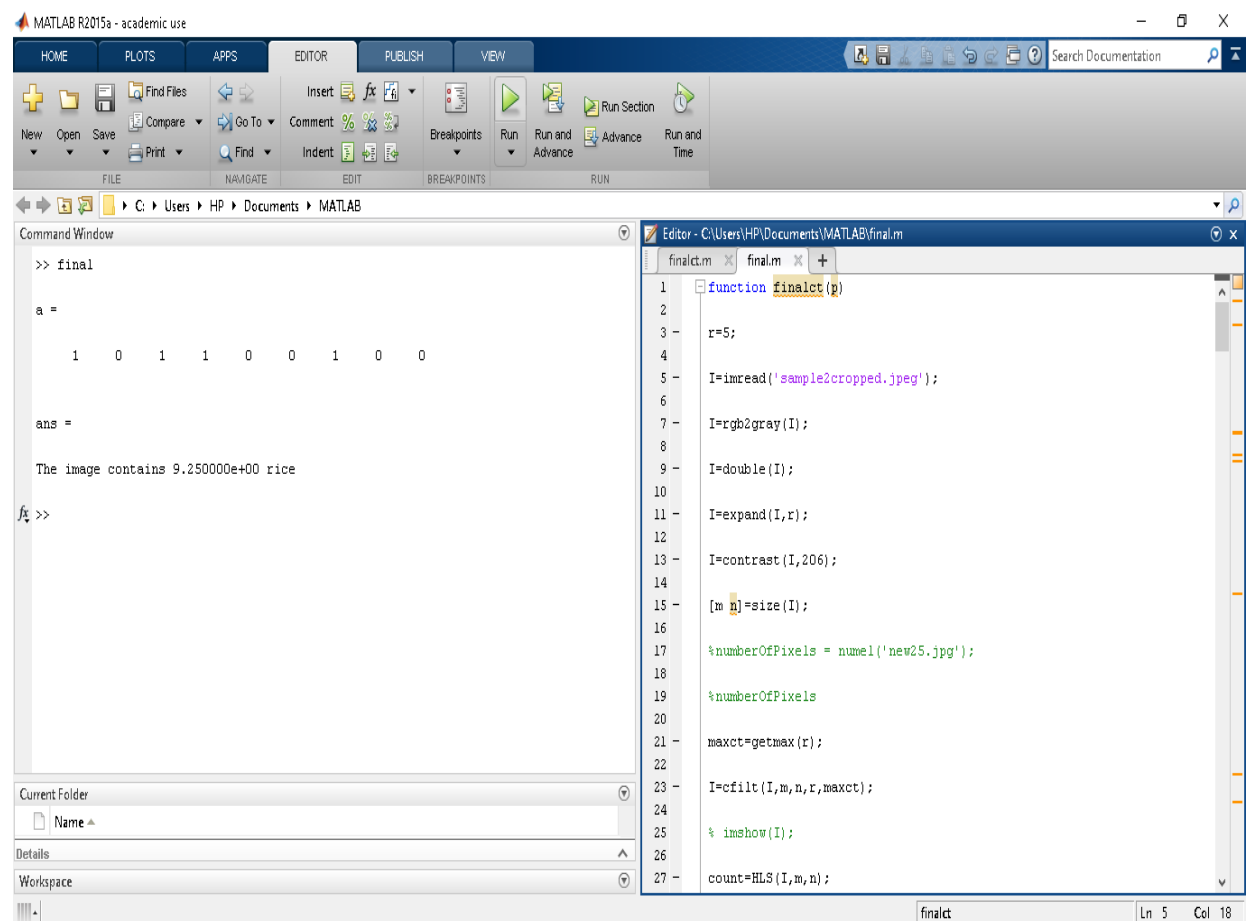
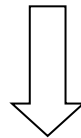


Fig. 5.1.3 Input Image 2



```

>> final

a =

     1     0     1     1     0     0     1     0     0

ans =

The image contains 9.250000e+00 rice

fx >>

function finalct(p)
1
2
3   r=5;
4
5   I=imread('sample2cropped.jpeg');
6
7   I=rgb2gray(I);
8
9   I=double(I);
10
11  I=expand(I,r);
12
13  I=contrast(I,206);
14
15  [m n]=size(I);
16
17  %numberOfPixels = numel('new25.jpg');
18
19  %numberOfPixels
20
21  maxct=getmax(r);
22
23  I=cfilt(I,m,n,r,maxct);
24
25  % imshow(I);
26
27  count=HLS(I,m,n);

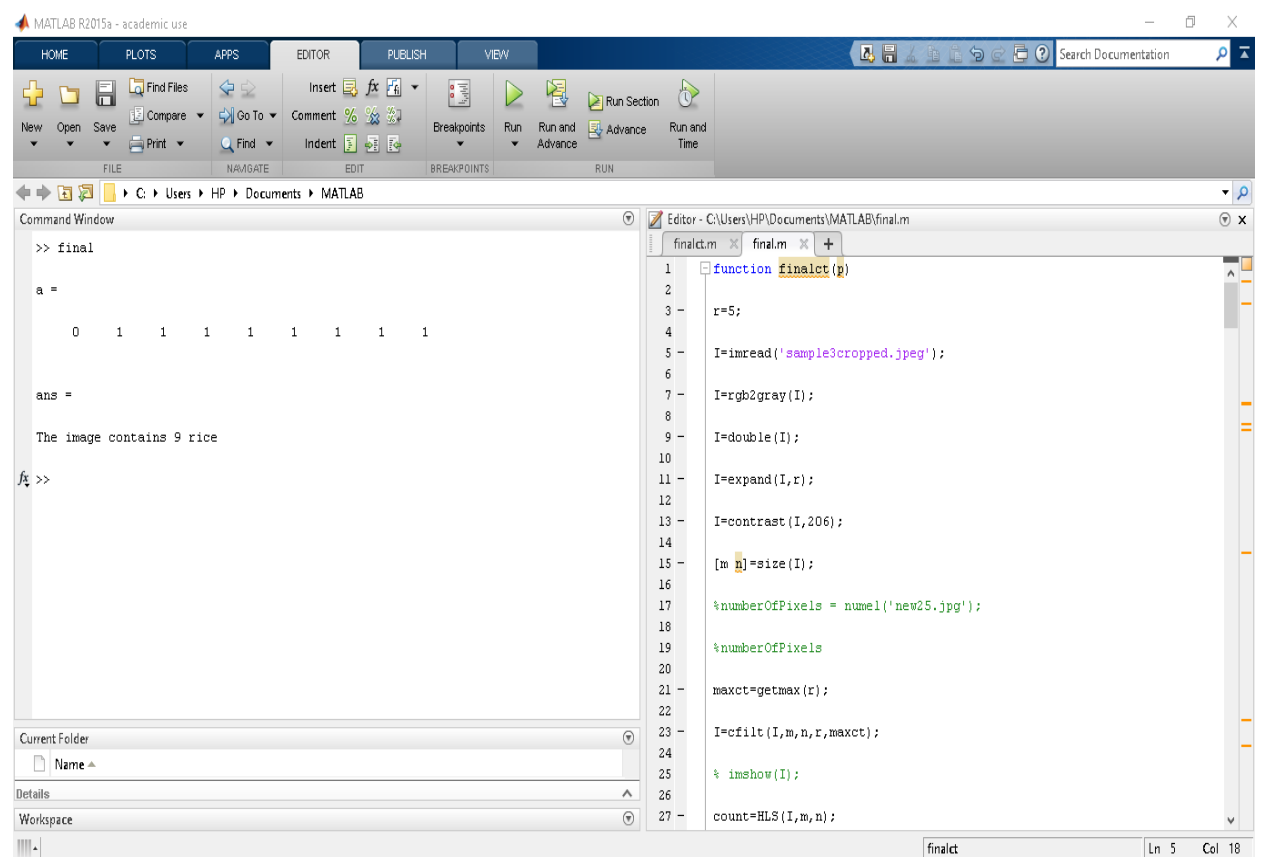
```

Fig. 5.1.4 MATLAB Output 2

### Sample Image and Output No.-3



Fig. 5.1.5 Input Image 3



```

>> final

a =

    0    1    1    1    1    1    1    1    1

ans =

The image contains 9 rice

f> >>

function finalct(p)
1
2
3 r=5;
4
5 I=imread('sample3cropped.jpeg');
6
7 I=rgb2gray(I);
8
9 I=double(I);
10
11 I=expand(I,r);
12
13 I=contrast(I,206);
14
15 [m n]=size(I);
16
17 %numberOfPixels = numel('new25.jpg');
18
19 %numberOfPixels
20
21 maxct=getmax(r);
22
23 I=cfilt(I,m,n,r,maxct);
24
25 % imshow(I);
26
27 count=HLS(I,m,n);

```

Fig. 5.1.6 MATLAB Output 3

## 5.2 EFFICIENCY

$$\text{Efficiency} = \frac{\text{Total no. of rice} - \text{No. of rice having error}}{\text{Total no. of rice}} \times 100 \%$$

Total Sample data of 15 pictures were observed i.e.

	A	B	C	D	E	F	G	H	I	J	K
1	1	0	0	1	0	0	1	0	0	sample1cropped.jpeg	
2	1	0	1	1	0	0	1	0	0	sample2cropped.jpeg	
3	0	1	1	1	1	1	1	1	1	sample3cropped.jpeg	
4	1	1	0	1	1	1	0	0	1	samplecropped10.jpeg	
5	1	1	1	1	0	0	1	0	1	samplecropped11.jpeg	
6	1	0	1	1	1	1	0	0	1	samplecropped12.jpeg	
7	1	1	1	1	0	1	1	1	0	samplecropped13.jpeg	
8	1	0	1	1	1	0	1	0	1	samplecropped14.jpeg	
9	1	1	1	0	0	1	1	1	0	samplecropped15.jpeg	
10	1	1	0	1	1	0	0	1	0	samplecropped4.jpeg	
11	0	0	0	0	1	0	1	0	1	samplecropped5.jpeg	
12	1	1	1	0	1	0	1	1	0	samplecropped6.jpeg	
13	1	0	1	1	1	0	1	1	1	samplecropped7.jpeg	
14	1	1	0	1	1	0	0	0	1	samplecropped8.jpeg	
15	1	1	0	1	1	0	0	0	1	samplecropped9.jpeg	

Fig. 5.2.1 Results of 15 sample images stored in MS Excel

$$\text{Efficiency} = \frac{(15 \times 9) - 19}{(15 \times 9)} \times 100 = 85.92 \%$$

## **CHAPTER 6 OUTCOMES AND PROSPECTIVE LEARNING**

### **6.1 OUTCOMES**

Outcome of this project is that mainly related to the agro processing industry which will provide impetus to lacunae of this industry in our country. The mentioned technology is manufactured in the countries like Korea and China, but they export this whole finished machine at a very high price.

But if we are able to build this technology, we will be making prices of machine can be dropped in our country also.

Now we are having advancement in algorithm also so that it can be used in other fields like sorting of pulses, lentils, finished ultrafine products and plastic products. There is a tremendous ray of hope ahead in this technology with no shortfalls to have.

**We would like to conclude the major outcomes of this project as follows:**

1. learning the new concepts of the MATLAB.
2. learning about the rice types and sizes.
3. Analyzing and testing the result on the rice samples of varying range.
4. It enhanced our algorithm writing skills.
5. We got an insight into agro processing industry.

### **6.2 PROSPECTIVE LEARNING**

There is tremendous probability of enhancement of this technology and taking it into the MSME sector so that we as a major producer of raw materials can become successful exporters of the finished products.

This study explores the pattern to separate out rice grains according to the color differences in raw rice arising from anomalies on the basis of texture, size, shape etc.

Due to system complexity and huge research in the field of signal processing, the principle and implementation of a system of rice sorter is proposed.

The system refers to automatically sorting the heterochromatic particles out of the batch of raw rice, removing such impurities in this process improves the quality of the rice. Improvement in food safety due to the removal of foreign materials is the primary aim of the proposed study.



This sector has a huge potential so as our product which can transform the sector in our country to take it into new heights.

There are various processes through which we have to follow, first the faulty rice is selected and this has lot of complications in it.

We have to make a border around the rice and mark its region, then we have to calculate its area in order to get idea about the discrepancy in the rice.

The scattering in the image is removed by applying the filters as discussed above chapters.

Now we have to spot the rice specifically and segregate it from the heap.

The structure of code can be improved using opening and closing approach of this figure.

Opening means enlarging the images in order to have the insights of image in a better way remove the shortfalls of images.

Next comes the closing which implies that we have to take our image back to its original size.

After we will evaluate our image so that we can have better calculations.

## 6.3 STUDENT OUTCOMES

### 6.3.1 Outcomes achieved till the end of present semester:

C. an ability to design a system, component, or process to meet desired needs within realistic constraints such as economic, environmental, social, political, ethical, health and safety, manufacturability, and sustainability.	
	<u>Yes/No</u>
C1. Analyze needs to produce problem definition for electronics and communication systems.	YES
C2. Carries out design process to satisfy project requirement for electronics and communication systems	YES
C3. Can work within realistic constraints in realizing systems.	YES
C4. Can build prototypes that meet design specifications.	YES

D. an ability to function on multidisciplinary teams.	
D1. Shares responsibility and information schedule with others in team	YES
D2. Participates in the development and selection of ideas.	YES
G. an ability to communicate effectively.	
G1. Produce a variety of documents such as laboratory or project reports using appropriate formats and grammar with discipline specific conventions including citations.	YES
G2. Deliver well organized, logical oral presentation, including good explanations when questioned.	YES
H. the broad education necessary to understand the impact of engineering solutions in a global, economic, environmental, and societal context.	
H1. Aware of societal and global changes that engineering innovations may cause.	YES
H2. Examines economics tradeoffs in engineering systems.	NO
H3. Evaluates engineering solutions that consider environmental factors.	YES
I. a recognition of the need for, and an ability to engage in life-long learning.	
I1. Able to use resources to learn new devices and systems, not taught in class.	YES

I2. Ability to list sources for continuing education opportunities.	YES
I3. Recognizes the need to accept personal responsibility for learning and of the importance of lifelong learning.	YES
K. an ability to use the techniques, skills, and modern engineering tools necessary for engineering practice.	
K1. Able to operate engineering equipment	YES
K2. Able to program engineering devices.	YES
K3. Able to use electronic devices, circuits and systems modelling softwares for engineering applications	YES
K4. Able to analyze engineering problems using software tools	YES

### 6.3.2 Learning outcomes from this capstone project:

Developing new/multidisciplinary technical skills.	5
Using professional and technical terminology appropriately.	5
Effectively utilizing and troubleshooting a tool for development of a technical solution.	4
Analyzing or visualizing data to create information.	4
Creating technical report with usage of international standards.	5
Acquiring and evaluating information.	5

## CHAPTER 7 PROJECT TIMELINE

### 7.1 TIME SCHEDULE FOR THE PROJECT

The project was finalized after numerous iterations in the project. After checking feasibility and various other aspects including efficiency of the system, the team came with the final version of the project.

Team as well as individual efforts put through the time are demonstrated in the following Gantt charts:

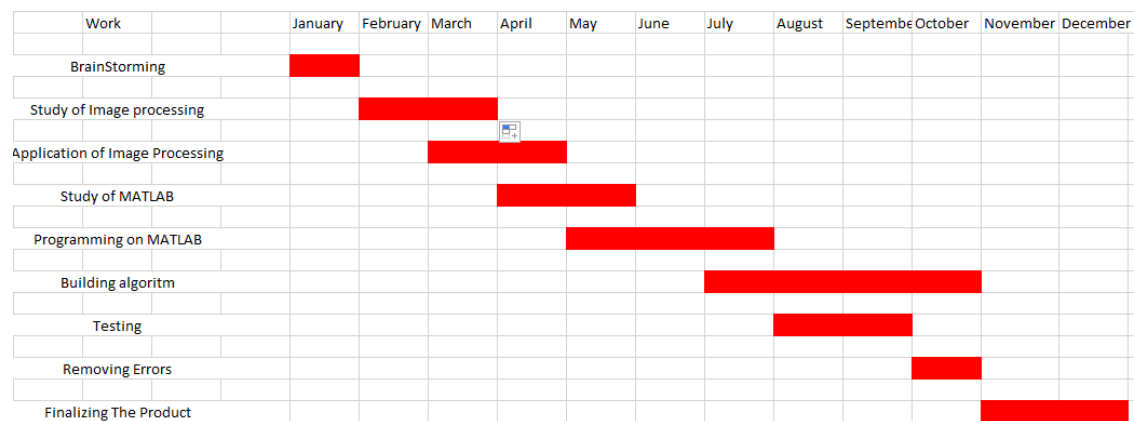


Figure 7.1.1 Gantt chart for Lakshay Aggarwal

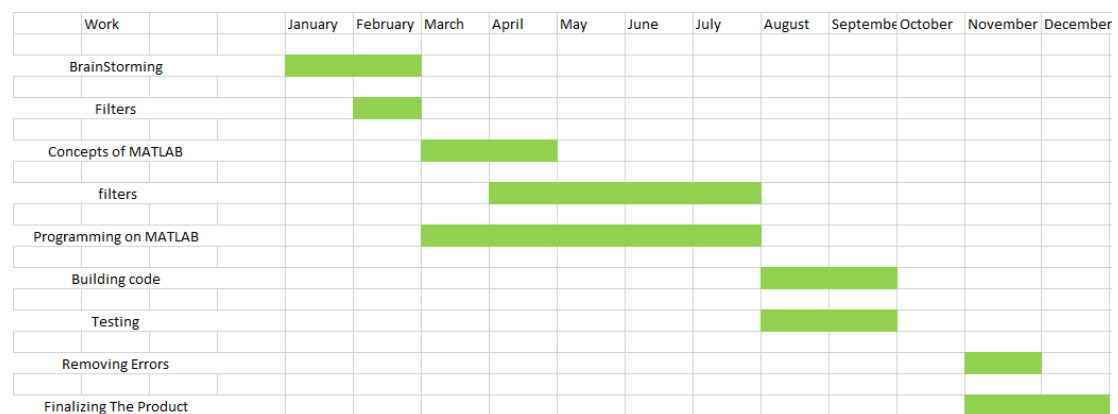


Figure 7.1.2 Gantt chart for Pulkit Bahl

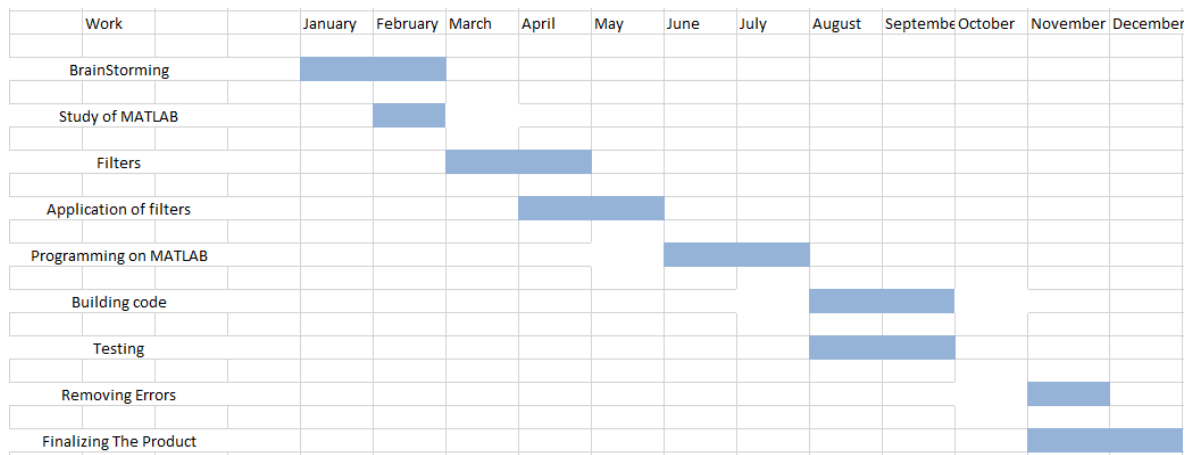


Figure 7.1.3 Gantt chart for Raghav Sharma

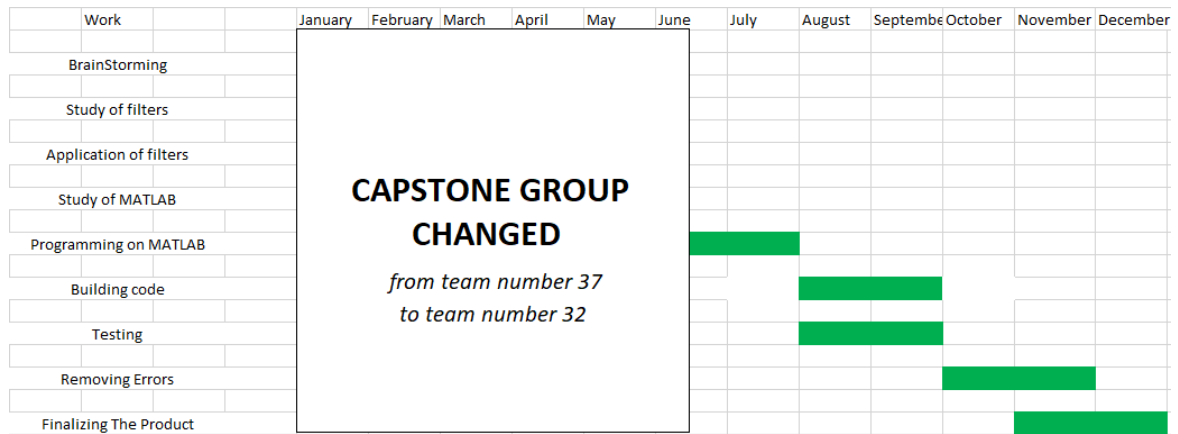


Figure 7.1.4 Gantt chart for Rahul Garg

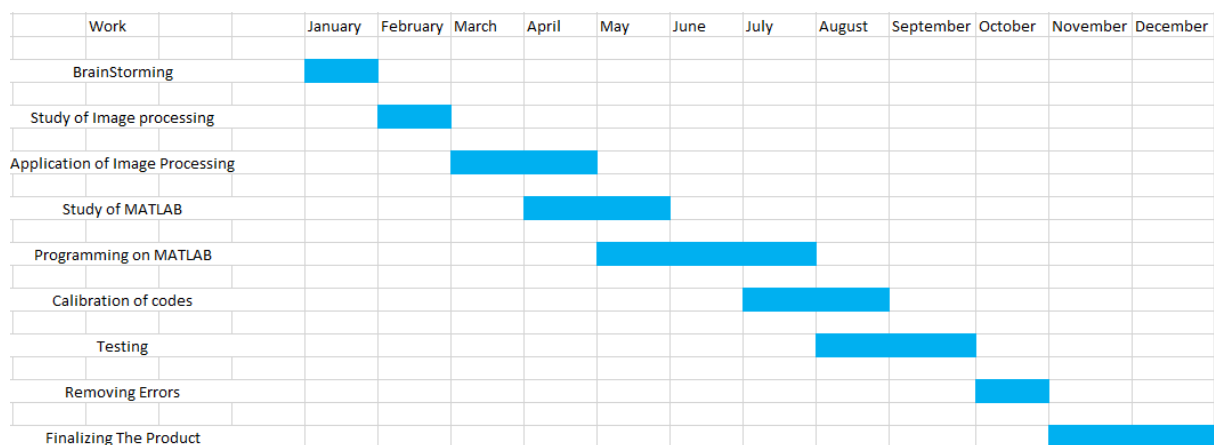


Figure 7.1.5 Gantt chart for Rajan Bansal

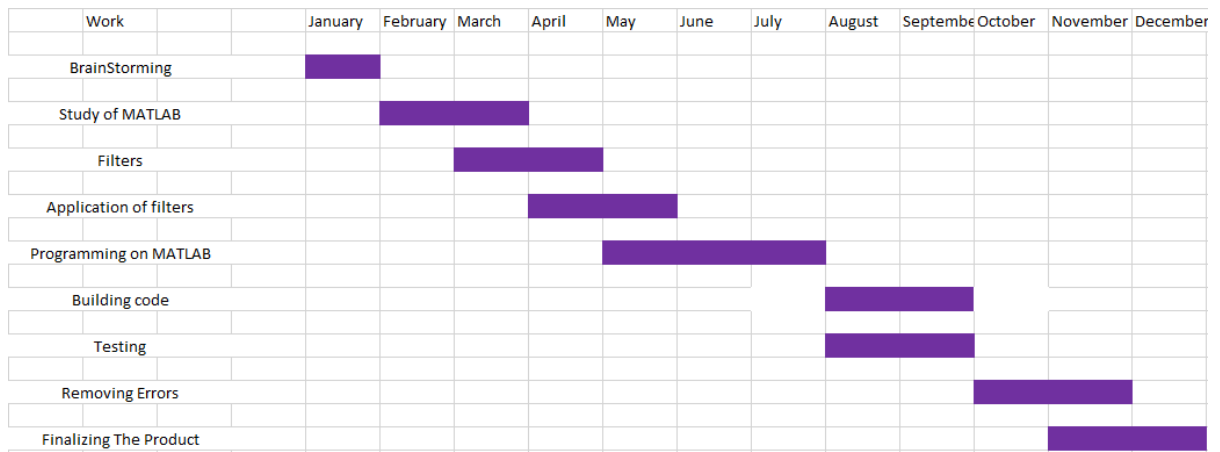


Figure 7.1.6 Gantt chart for Saket Panchori

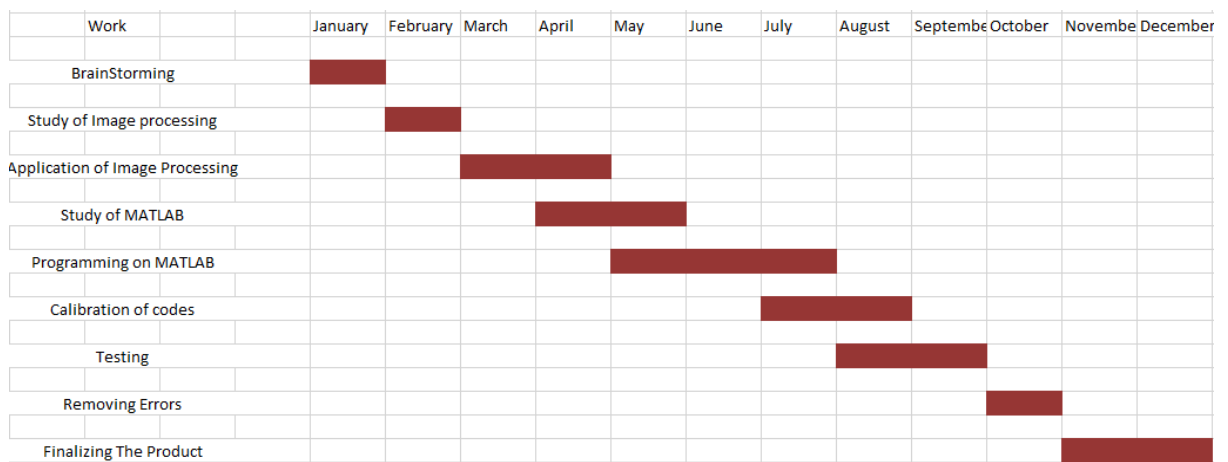


Figure 7.1.7 Gantt chart for our team

## REFERENCES

1. "IMAGE PROCESSING IN FREQUENCY DOMAIN USING MATLAB: A STUDY FOR BEGINNERS" by Vinay Kumar, Manas Nanda
2. "Automated Defect Recognition Method by Using Digital Image Processing" by Sangwook Lee, Ph.D. Texas Tech University Lubbock, Texas
3. "On Detection of Median Filtering in Digital Images" by Matthias Kirchner and Jessica Fridrich
4. "FPGA: An Efficient And Promising Platform For Real-Time Image Processing Applications" by Sparsh Mittal, Saket Gupta and S. Dasgupta
5. "Sphericity, shape factor, and convexity measurement of coarse aggregate for concrete using digital image processing" by C.F. Mora, A.K.H Kwan
6. <http://www.wikipedia.org>
7. <https://in.mathworks.com>
8. "Particle shape analysis of coarse aggregate using digital image processing "by A.K.H. Kwan CF Mora H.C Chan
9. "Particle size distribution analysis of coarse aggregate using digital processing" by C.F. Mora A.K.H. Kwan H.C. Chan