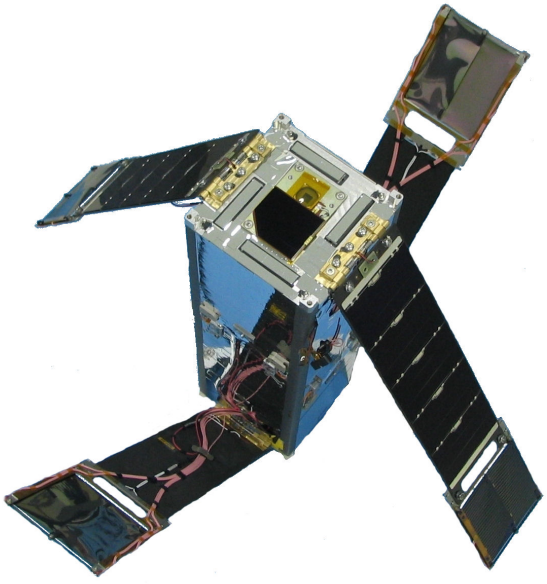
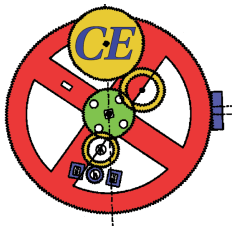
Delft University of Technology
Master's Thesis in Embedded Systems

Computational Imaging for Earth Surveillance

Pranav Sailesh Mani





TU Delft Delft University of Technology

Computational Imaging for Earth Surveillance

Master's Thesis in Embedded Systems

Computer Engineering Section
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, 2628 CD Delft, The Netherlands

Pranav Sailesh Mani
p.s.mani@student.tudelft.nl

12th November 2017

Pranav Sailesh Mani
Pranav Sailesh Mani (p.s.mani@student.tudelft.nl)
Title
Computational Imaging for Earth Surveillance
MSc presentation
12th November 2017

Graduation Committee

Dr. ir. J.S.S.M. Wong	Computer Engineering, Delft University of Technology
Dr. ir. A.J. van Genderen	Computer Engineering, Delft University of Technology
Dr. ir. J.M. Kuiper	Aerospace Engineering, Delft University of Technology

Abstract

TODO ABSTRACT

Preface

This document contains the work that I have been doing for the past eight months. These months just flew by and I enjoyed working in this multi-disciplinary project. Firstly, I am thankful to the almighty for providing me the strength to overcome various challenges I faced over the past two years of my master program which has been highly demanding. I have a long list of people whom I would like to thank and without whose support this project could not have been completed. I would like to first thank my parents for providing me the financial support for coming to the Netherlands and doing my master studies at TU Delft. Without their encouragement and support, I would not be where I am now. Next, I would like to thank my master program co-ordinator and my supervisor, Arjan van Genderen who accepted to supervise me in this multi-disciplinary project. I would like to thank Delfi Space Program Manager, Mr. Jasper Bouwmeester, who first introduced me to my second supervisor Hans Kuiper. Hans provided me the support and was instrumental in understanding the requirements for designing the imager. Next, I would like to thank the Paul Urbach who introduced me to the Optica Research Group which is where I did all the experiments. I am extremely thankful to Phd students, Yifeng Shao and Sander Kojninberg. Yifeng was also my daily supervisor and helped me with understanding the theoretical physics, simulation of imaging algorithms and also on how to work with optical instruments. He continuously encouraged me to overcome challenges I faced along the way. I could not have asked for a better supervisor. I would also like to thank Sander for many discussions that very highly insightful and helped me to cross hurdles that I faced. On the whole, the Optica Research Group was a very nice place to work with very sociable people. Lastly, I would like to thank my friends in Delft for their support and encouraging words in stressful times.

Delft, The Netherlands
12th November 2017

Contents

Chapter 1

Introduction and Problem Statement

The history of cameras go back to 13th century when Aristotle first noticed how light passing through a small hole in a darkened room produced an image of the sun on the wall. Throughout the centuries, the basic design of cameras have been continuously changing with different versions of the 'camera obscura'. 'Camera Obscura' is a phenomenon that occurs when when a scene is projected onto a pinhole and the image of that scene is formed on the surface opposite to that of a pinhole.

In a pinhole camera, light passes through the pinhole and forms an image on the sensor/image plane.As the size of the pinhole increased, the quality of image formed on the plane decreased and as the pinhole size became smaller, lesser light was allowed which resulted in decreased field of view. With the development of science and due to the limitations of the pinhole, lenses were introduced to increase the size of the aperture, the sharpness of the image and the light throughput. As humanity progressed with the rapid pace in technology, we were able to capture images and store them on a film. With the digital explosion in early 1990s, the thin films were replaced by Charged Couple Devices(CCD). Then came the cameras based on Complementary Metal Oxide Semiconductors(CMOS). CCD and CMOS sensors reduced the size of cameras considerably and it was possible to develop low cost cameras in a large number. However, cameras have retained the lens throughout the years. Cameras are used for various applications and one such application is the space exploration domain.

1.1 Context

The Delfi program focuses on development of CubeSats at Delft University of Technology, to provide hands-on education for students and also provide technology demonstrations for the Dutch Space Industry. The satellites that

have been launched under the Delfi program are as follows:

- Delfi-C3 : Delfi-C3 is the first university class satellite and also the first nano-satellite from the Netherlands. It was launched on April 28, 2008 from India. Delfi-C3 is a full mission success and is still operational[2].

- Delfi-n3Xt : The Delfi-n3Xt(pronounced as Delfi-Next) is the second set of Cubesats developed as successor to Delfi-C3. It was launched from Russia on November 21, 2013. It was able to log data for three months and perform all foreseen technology demonstration experiments.

Delfi-PQ programme is a sub-programme of the Delfi Space programme that aims at developing extremely small but highly capable PocketQube satellites. PocketQubes are an order of magnitude smaller than the well known CubeSat standard which formed the basis of previous Delfi satellite projects. The dimensions of a PocketQube satellite would be 50mm * 50mm * 178mm and their volume would be approximately eight times smaller than CubeSats. One of the advanced payload that would be part of the Delfi-PQ would be an imager/camera that consumes extremely low power and would fit into the dimensions specified by the Delfi-PQ team. The thickness of the camera should be less than 6mm thick.

1.2 Problem Statement

In order to reduce the size of a camera, it would be necessary to remove the lens from the camera as the thinnest lens based mobile camera is 5mm thick. The primary focus of a lens would be to focus light from distant objects onto the CMOS sensor. Light from distant objects reach the sensor even without the lens except that the light is incoherent and the CMOS sensor would not be able to form the object properly without a lens. However, the lens could also be replaced by coded apertures. Coded Apertures have been used in the late 20th century to image X-Ray sources of light. Lensless coded aperture cameras can be as small as $100\mu m$ thick. By using lensless cameras, we could potentially reduce the form-facto multiple times to suit the requirements of Delfi-PQ. Apart from this, it would be the first attempt to use a lens-less camera for use in satellites to image astronomical objects in the visible light spectrum. In the case of lensed imaging, the image of the scene is directly obtained on the sensor. That does not happen in the case of lens-less imaging. In lensless imaging, one would need to computationally reconstruct the object scene using various computational methods. This is one of the trade-off that we need to sacrifice in the case of lensless imaging. This thesis would address the following research question: **Is it possible to design "lensless coded" aperture cameras with a small form-**

factor(thickness $<$ 10mm) using COTS(commercial off-the-shelf) components that can be used in U-class Spacecrafts ?

This question can be broken down into the following sub-questions:

- What would be the CMOS sensor that can be used for the camera?
- How do we design the hardware and software for such a camera that can be used in Delf-PQ satellite?
- What would be the field-of-view and spatial resolution of the lensless camera?
- What would be the computational algorithm that would be used in such a lens-less camera? How do we experimentally prove the concept of lens-less imaging?

Based on the above set of questions, the following goals have been determined for the project:

- Perform a survey of previously used CMOS sensors in various CubSat missions around the world. Choose a CMOS sensor that could be used for lensless imaging.
- Design the hardware and software prototype using a commercially available micro-controller platform.
- Design an experimental setup for determining the field-of-view and and spatial resolution of the lens-less camera.
- Design an experimental setup to prove the concept of lensless imaging using existing computational methods.

1.3 Assumptions

The thesis makes the following assumptions:

- The imaging system needs to use a COTS micro-controller platform that could be easily integrated with on-board computer module on the Delfi-PQ. This assumption is fair as the previous Delfi satellites use commercially available micro-controller platforms for use.
- The second assumption is that it would be possible to stream down the data from the satellite to the earth and perform computations in the ground station and perform computational reconstructions. Lens-less Imaging requires computational reconstruction and it would be a computationally complex task to perform complex fourier computations on-board satellite. Apart from this, various computational techniques

can be added to perform better reconstructions at a later stage. This assumption is valid since the previous Delfi satellites periodically send back data to the earth. This assumption allows a greater flexibility while designing the system.

- It is also assumed that there is no loss of image data in the transmission between the satellite and ground-station and that the image is sent as stored by the imaging module.

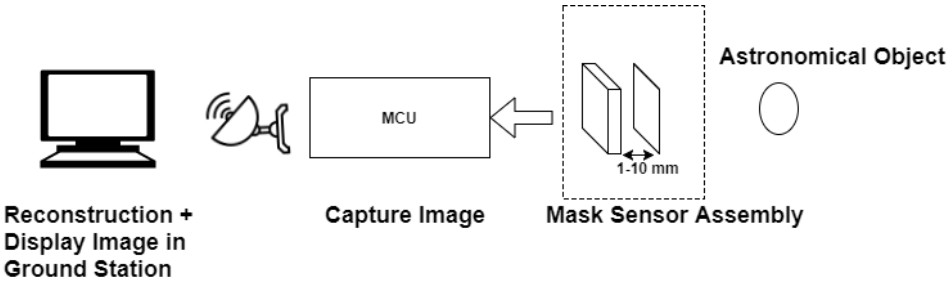


Figure 1.1: Assumed System Architecture

Chapter 2

Background and selection criteria

In this chapter, a state-of-the art study will be presented that could assist in design of the lensless imager with specifications mentioned in the previous chapter.

2.1 Camera Computational Pipeline

In order to design a lensless imaging system, we must first look at the computational imaging pipeline of existing cameras. Since the lensless camera basically uses computation to reconstruct images, it is important to understand the computational pipeline of existing camera systems and make necessary modification in the design of the existing pipeline to suit the system. The computational imaging pipeline of existing camera systems is shown figure 2.1[26]. As shown in figure 2.1, there are five main components

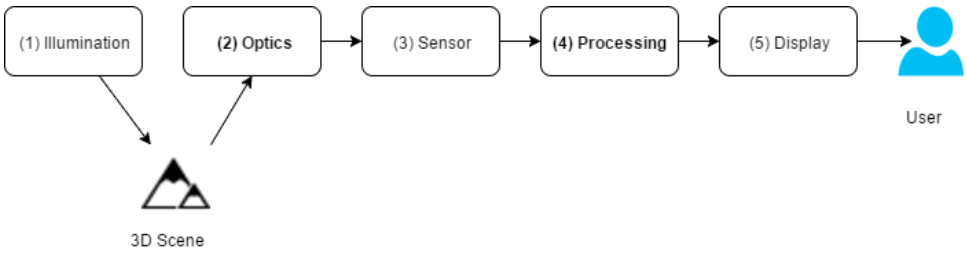


Figure 2.1: Computational Pipeline of Existing cameras

that can be controlled computationally in existing systems. Illumination of the scene can be controlled to produce an enhanced picture. Optics could be controlled to limit the amount of light entering the scene and thereby controlling the image produced on the sensor. The sensor can also computationally modify the date it receives to de-noise, adjust the blackness/white

in an image. Post-processing can also be done on the image produced by the sensor to improve the image produced by the sensor. Finally, a display can also be modified computationally to produce certain effects on the user. And of course, the user can control any of these components to produce the effect he desires. But in the case of the lensless imaging system, we would be modifying the optics and the processing components of the pipeline to reduce the size of the camera.The components to be modified are are darkened in figure 2.1.

2.2 Satellite Imaging Architectures

Since the camera is going to be capturing pictures of the earth, it would be required to study the existing imaging architectures currently being used in satellites and how the design of the lensless camera would fit into the existing imaging architectures. We will first look into the terminology commonly used in space instrumentation. As the imager is carried along the orbit of the earth, it images a strip on the surface of the earth. The width of the strip is called the 'swath'. The direction along which the satellite moves or images is called the 'along-track' direction and the direction perpendicular to it is called the cross-track direction[6]. Figure 2.2 describes these terminology and some other terms as well. Three major types of scanning architectures

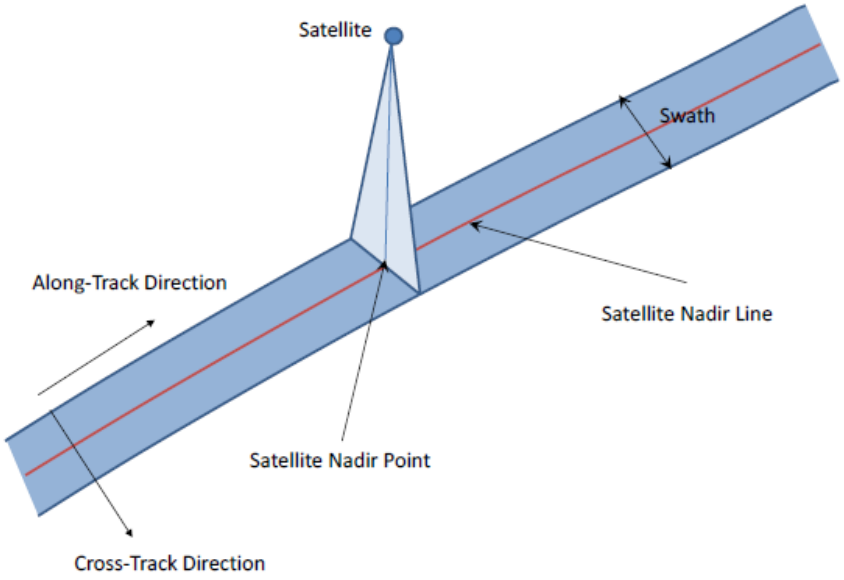


Figure 2.2: Various Imaging Terms[6]

are employed in space instruments, namely:

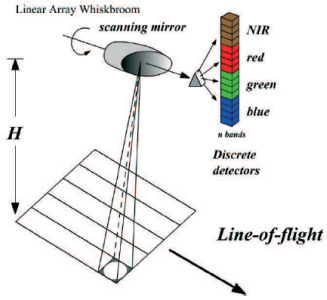
- Whiskbroom Line Scanner: In this type of scanning architecture, a detector element detects it's instantaneous field of view which is projected onto a pixel element. In this scanning type the surface of the earth is scanned in lines. A scanning mirror would project a very small area of the earth onto the single pixel element. The scanning mirror would then rotate to project the next element of the line onto the next pixel. Depending on the motion of the satellite, the next line of the detector is scanned and projected on to the next line on the surface of the earth. An advantage of this type of detector is that it would be possible to obtain a very large field of view. However, it also comes with disadvantage that a very high sampling frequency is required to get decent resolutions. Typically, an earth observation satellite would move at 6.5 km per second. In order to get a resolution of 100 meters per pixel, it would be required to sample atleast 65 lines per second. For a swath of 1000 pixels it would be required to sample at 65000 elements per second. Apart from this, there is very limited time for each detector element which would result in low spatial resolution[32]. Another main disadvantage is that mechanical components would be required to project different parts of the surface of the earth on to the detector element. This type of scanner is also called as along-track scanner. Mathematically, the measurement of the detector element (X, Y) can be described using

  $$(X, Y) = f(t_x, t_y)$$

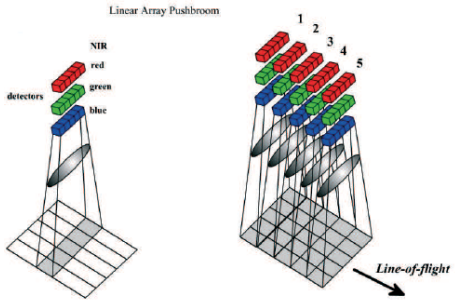
  where t_x and t_y is the time at which the image is captured in the corresponding location

- Pushbroom Line Scanner : In this type of architecture, the orbital motion of the sensor is used to image the swath instead of using a mirror as in the case of whiskbroom scanner. The field of view in the cross-track direction is imaged by the corresponding line detector array. Successive lines are imaged and sampled by the multiplexer as the sensor moves across the surface. The time between sampling two successive lines can be the time it takes for the satellite to move that distance. The most commonly used detector for a pushbroom scanner is Charge Coupled Devices(CCD). One of the main advantages of this type of scanner is that it requires no moving parts. Due to this, it is possible to obtain very high scanning rates(¡ 1μ second). This also leads to lower noise in the received signal[32]. The disadvantage is that large number of detectors are required to image a large piece of area. In addition to this, it requires an optical arrangement that could obtain a wide field of view. Mathematically, the measurement of the detector element (X, Y) can be described using

  $$(X, Y) = f(x), f(t_y)$$



(a) WhiskBroom Scanning



(b) Pushbroom Scanning

Figure 2.3: Push Broom and Whisk Broom Scanning

where $f(x)$ represents the sensor output and $f(t_y)$ represents the time at which the subsequent rows are imaged.

- Staring Array : Staring arrays use 2-d CCD/CMOS detectors to capture an entire area on the surface of the earth. These are also called as framing cameras. This provides speed-up and step-and-stare mechanism is employed wherein observations are made intermittently after a certain number of steps in the cross-track direction. The advantage is that moderate field of view optics is only required in this case[32]. Mathematically, the measurement of the detector element (X, Y) can be described using

$$(X, Y) = f(x), f(y)$$

where $f(x)$ and $f(y)$ represents the sensor output.

2.3 Spatial Resolution and Field of View

Two important factors that come into account when developing cameras for space applications are spatial resolution and field of view. These factors determine the performance of the camera and it is important to know how

these cameras will perform when designed.

2.3.1 Spatial Resolution

Spatial resolution in remote sensing refers to the smallest area represented by the pixel element of the detector. The impact of different spatial resolutions on imaging a house object is shown in Figure 2.4. For example, if an house on earth measures 30*30 m, a camera with a spatial resolution of 30 m cannot image the house completely and the pixel is going to be the average color of that particular area which is 30m. If the spatial resolution is reduced to 5m, then we can differentiate between different components of the house. An even smaller spatial resolution of 1m will enable us to see the finer details of the house components.

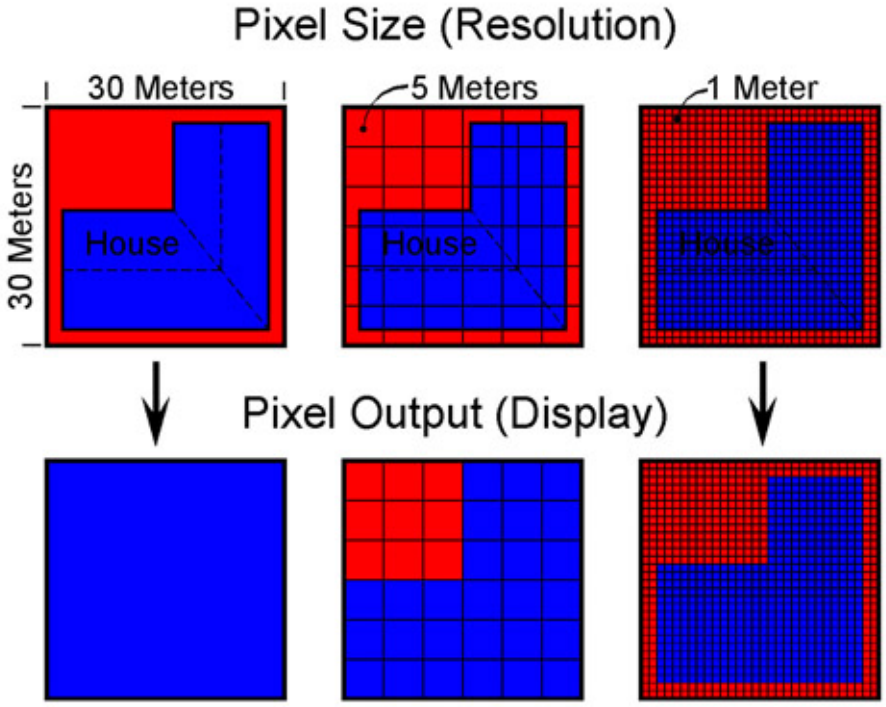


Figure 2.4: Picture Quality variation according to different spatial resolutions[21]

In remote sensing, a system with high spatial resolution is one which has resolution of 0.41-4 meters per pixel, a system with low spatial resolution has resolution ranging from 30-1000 meters per pixel[21]. The spatial resolution of a system can be determined by the equation equation 2.1.

$$d_e = \frac{d_p}{d_t} * d_h \tag{2.1}$$

where d_e indicates the spatial resolution, d_p indicates the individual pixel size of detector element, d_h indicates the height of the instrument from earth and d_t indicates the thickness of the camera or the distance of the lens/mask from the sensor.

The equation 2.1 is obtained by using the simple triangular rules and the relation can be obtained by using Figure 2.5.

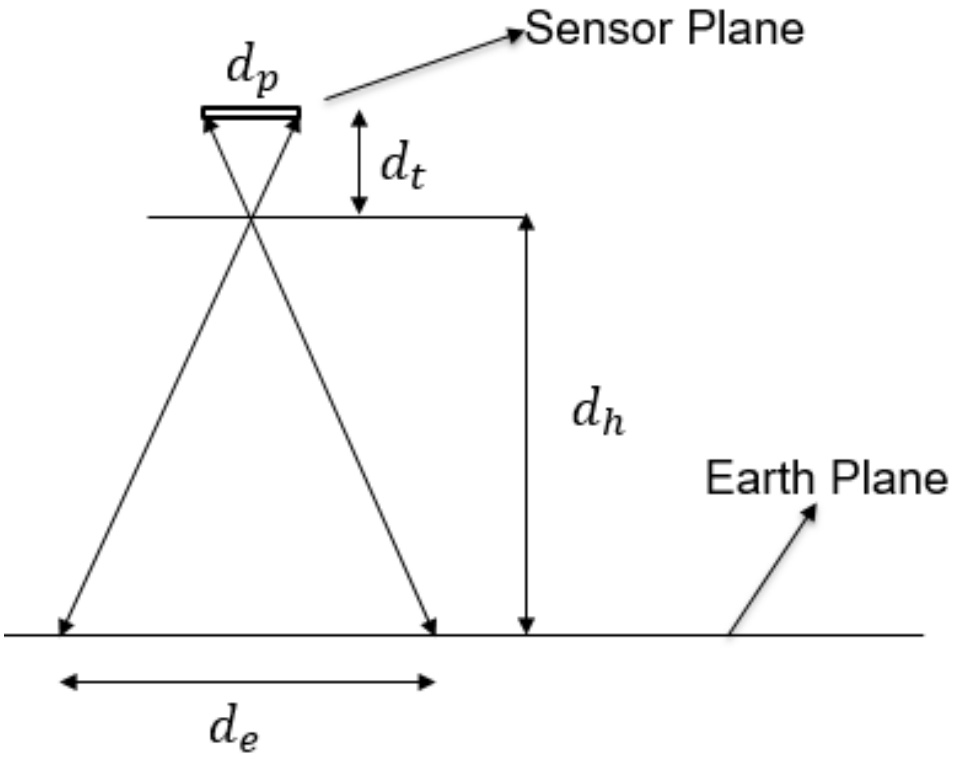


Figure 2.5: Spatial Resolution Calculation

2.3.2 Field of View

The field of view is the amount of area of the surface of the earth that can be imaged by the sensor. In a lens-based system, the field of view is determined by the chief ray angle of the lens in use. In a lensless sytem, the field of view is determined by the acceptance angle of the CMOS/CCD sensor in use. The field of view can be determined by the equation 2.2.

$$d_e = 2 * d_h * tan(\theta_{cra}) \tag{2.2}$$

where d_e is the area that can be imaged by the sensor, d_h determined the height of the instrument from the surface of the earth and θ_{cra} determines

the acceptance cone or the acceptance angle of the sensor elements. This equation can be determined by using simple triangular laws and this is shown in Figure 2.6. Sensors used by the landsat program(program by NASA for

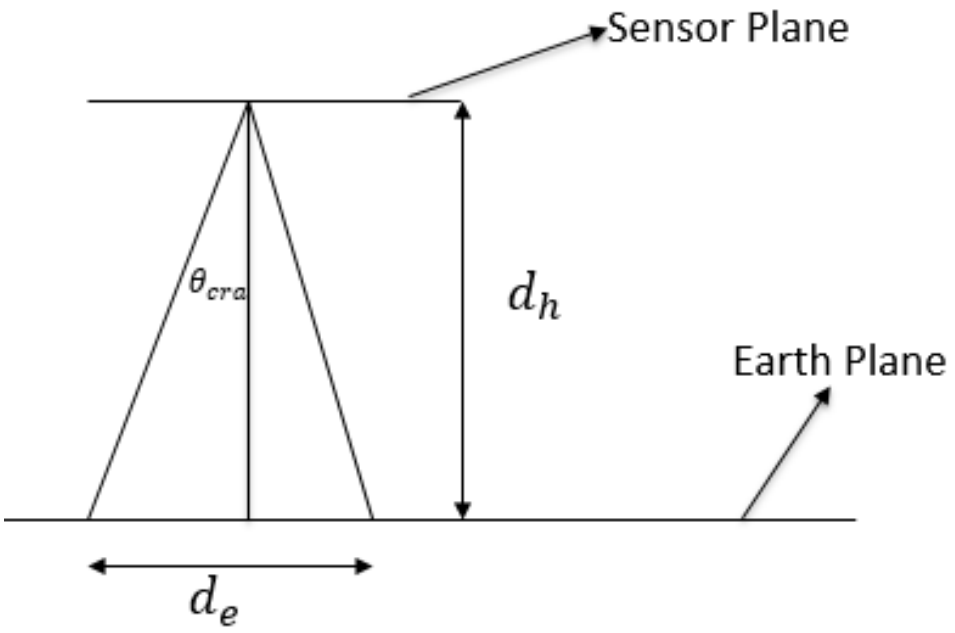


Figure 2.6: Field of View Calculation

earth observation satellites) have a field of view of 185*185 kilometres at a height of 705 kilometres from the surface of the earth[9].

2.4 Trade-off Analysis

2.4.1 Camera Sensor

The camera sensor is the core of the Delfi-PQ Imager. The performance of a camera is mainly limited by the image sensor that it uses[28]. The camera sensor can be off two types namely, CCD(charge coupled device) or CMOS(Complimentary Metal Oxide Semiconductor). Both the types of sensors have their own advantages and disadvantages. To understand the challenges that each type of sensor poses, we must understand how the sensors are designed.

These CMOS sensors can be categorized under the starring array type of imaging architectures that we mentioned previously. The following factors have been chosen to make a trade-off between the different CMOS sensors:

1. Resolution : When rating a camera, the first thing that comes to the mind is the resolution of the camera. The resolution of a camera is directly dependent on the number of pixels in the image sensor of the camera.

2. Power Consumption : In the design of the PQ-Camera, the most important factor is the power consumption of the entire imager. The majority of the power consumption by the imager is dependent on the power consumption of the CMOS sensor.

3. Availability : Even though there are innumerable number of CMOS sensors in the world, availability of CMOS sensors is quite low when it comes to small-scale. Many CMOS manufacturers require large scale orders.

4. Quantum Efficiency(QE) : Quantum Efficiency is the measure of efficiency of the camera sensor to convert incoming photons into electrons. The ratio of electrons generated during the digitization process to photons is called quantum efficiency.

5. Pixel Size : Pixel size is the size of each pixel unit in the CMOS camera. It is also an important factor considering that the signal produced by the CMOS sensor depends on the pixel size as well.

$$Signal = LightDensity * (PixelSize)^2 * QE$$

6. Electronic Interface : The electronic interface that can be used to retreive data from the CMOS sensor also plays an important role. Since the project uses a low-power microcontroller that has limited communication capabilities, it would be wise to chose an interface that is supported by the microcontroller. Recently available chips use LVDS/MIPI interface to send data. These interfaces are not supported by the microcontroller that is being used as an on-board computer. The on-board computer uses an I2C based interface and that the electronic interface would be an important factor as it would reduce the complexity of the system and also reduce the power consumption by removing the addidional circuitry necessary for interfacing with the onboard computer.

7. Dynamic Range : Dynamic Range and SNR are used interchangeably in CMOS sensors. The only difference is that dynamic range considers only the temporal dark noise while SNR includes the root mean square of the shot noise as well.

8. Shutter Type : Camera sensors use different types of shutters namely, global shutter and rolling shutter. Global shutter reduces the distortions due to fast moving artefacts while increasing the dark current.

Rolling shutter has more distortions in the case of imaging moving artefacts, but also has lesser dark noise compared to global shutter.

9. Voltage Level : Voltage level also has to be taken into account while choosing the sensor because if the CMOS sensor needs a voltage level higher than that of the main satellite bus voltage, then additional circuitry has to be introduced to step up the voltage level which in turn increases the overall system power.

10. Operating Temperature : Operating temperature is an important factor to take into account when choosing an imaging sensor. Since the camera is going to operate in space, it is better if the CMOS sensor has a higher operating range of temperature.

11. Overall Size and Weight : As the imager has to fit within specific dimensions, the overall size and weight of the CMOS sensor also needs to be taken into account.

12. Frame Rate: Even though, it is not required to have a camera sensor that is capable of high frame rates, it is an added advantage and higher frame rate camera could help in imaging larger areas of the earth if required.

13. Price: While there are no specific cost constraints in the project, price has also been taken into account.

In [33], a survey of camera modules for a CubeSat space Mission has already been carried. However, we also consider image sensors(not same as camera modules) as we are fundamentally changing the design of a camera.

The following candidates have been chosen for analysis. These candidates are chosen based on [33] and also on the latest CMOS sensors available on the market.

(a) IDS UI- 1646LE USB 1.3MP

(b) C3188A

(c) PC67XC-2 CCD

(d) MicroCam TTL

(e) PB-MV40

(f) Omnivision OV7670

(g) Sony ICX285AL

(h) Omnivision OV5642

(i) Omnivision OV2740

(j) MCM20027

Table 2.1: Comparison of Different Image Sensor Candidates

Factors \ Candidates	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)	(j)
Optical Parameters										
Resolution	++	+	+	-	++	+	++	++	++	++
Pixel Size	+	++	X	++	++	+	++	+	+	++
Shutter type	-	-	X	-	-	+	+	+	+	-
Frame Rate	+	+	+	+	+++	+	+	+	+	+
Electrical and other parameters										
Power Consumption	- -	- -	- - -	++	-	++	-	- - -	- -	- -
Availability	-	+	- - -	- - -	- - -	++	- -	++	- -	- - -
Electronic Interface	+	+	- - -	++	- -	++	- -	++	-	+
DR and SNR	X	+	X	++	+	++	+	++	+	+
Voltage	+	+	- - -	++	+	++	+	++	-	+
Operating Temperature	+	+	+	+	+	+	+	+	+	+
Overall Size and Weight	+	+	+	+	+	+	+	+	+	+
Price	-	+	X	X	- - -	++	X	+	X	X
Points	2	8	-8	8	0	18	5	13	2	4

2.4.2 Compression Algorithms

Data Compression plays a very important role when it comes to space missions. It is a very important aspect in the system design of a lensless camera for space application as the image that needs to be sent down to earth needs to be compressed as much as possible. Various surveys[34][29][23] have been conducted previously in-terms of which compression algorithm would best preserve the data and provide maximum compression at the same-time. Compression algorithms can be divided into two types namely lossy and lossless. Lossless compression algorithms are algorithms in which we can reconstruct the original image without any loss in data. Typical examples of lossless algorithms include Portable Network Graphics Format(PNG), Bitmap Format(BMP), TIFF(Tagged Image File Format). Lossy compression algorithms are compression algorithms in which we cannot completely reconstruct the original image. However, lossy compression factors offer a very high compression ratio compared to loss-less algorithms. Lossy Compression

algorithms include DCT(Discreet Cosine Transform), JPEG(Joint Photographic Experts Group), SPIHT(Set Partitioning in Hierarchical Trees)[29]. A survey conducted for European Student Moon Orbiter Mission[29] has reviewed various possible compression algorithms that could be used for lunar imaging. The compression algorithms were evaluated on the basis of Mean Square Error(MSE), Peak Signal to Noise Ratio(PSNR), Normalized Cross-Correlation(NK), Averaged Difference(AD), Structural Content(SC), Maximal Difference and Normalized Absolute Error(NAED). The reviewed compression formats were BMP, CGM, GIF, JPEG, PNG, TIFF, WebP. Loss-less compression generally does not offer a compression ratio of more than 2 due to the entropy present in real-life artefacts. Since space missions offer a limited communication link-speed, the compression ratio needs to be more than 2. This cannot be offered by lossless compression and hence we need to go for lossy compression formats. It can be seen from table [**?**] that JPEG and SPIHT perform better compared to DCT. It can be seen that JPEG offers better performance in-terms of Mean Squared Error. However, SPIHT offers a higher compression ratio.

Table 2.2: Ranking of Different Compression Algorithms

Factors / Method	MSE	PSNR	AD	Compression Ratio	Implementation Needed
JPEG	1	1	1	2	No
DCT(With Zip)	2	2	1	3	No
SPIHT	3	1	1	1	Yes

JPEG algorithm is chosen for implementation since it offers equivalent performance to SPIHT and JPEG compression engine is present in most of the commercially available CMOS/CCD sensors.

The JPEG compression algorithm is comprised of the following steps.

- The first step is to prepare the image for compression. The picture in the RGB color format is converted to YCbCr colorpace. The YCbCr model has three channels namely, Luminance(indicated by Y), Chroma Blue(indicated by Cb) and Chroma Red(indicated by Cr). The reason for this is that the human eye is more receptive to luminance than chrominance. The Cb and the Cr channels are then down-sampled which results in loss of data. Since the maximum amount of data is in the luminance part, it is not downsampled and the channel data is retained without any down sampling.

- The output image is then split into blocks of size 8*8. It is divided into blocks of 8*8 because there will be less variation in this region of

the image and this can be utilized to compress the image as much as possible.

- After this, the Discreet cosine transform is applied on these blocks of data to obtain the DCT coefficients for different frequencies. The idea behind the DCT is that by knowing the coefficients of different frequency components, it would be possible to reconstruct the image block. This is illustrated in Figure 2.8.

$$S_{i,j} = \frac{1}{4} * C_i * C_j \sum_{x=0}^{x=7} \sum_{y=0}^{y=7} P_{x,y} * cos(\frac{(2y+1) * i * \pi}{16}) * cos(\frac{(2x+1) * i * \pi}{16}) \tag{2.3}$$

  where $P_{x,y}$ is the pixel value at the location x, y and C_i and C_j can be calculated using the equation 2.4.

$$C_i = \begin{cases} \frac{1}{\sqrt{2}} & \text{if } i = 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.4}$$

- The next step is the lossy part of the JPEG compression which involves quantization of the data. In this process the the coefficients obtained are quantized using a 8*8 quantization matrix. The quantization process is illustrated by equation 2.5.

$$B_{j,k} = \frac{S_{i,j}}{Q_{j,k}} \tag{2.5}$$

  This quantization matrix varies form sensor to sensor depending on the manufacturer. However, the matrix is such that the DC component would be retained and the higher frequency components are rounded off to zero. This quantization factor also controls the quality of compression and higher the value of quantization table, lower the value of compression. If you multiply the quantization by a large quantization factor, you would lose the higher frequency components and would only retain the DC component(given by $S_{0,0}$) and lower frequency components. Many CMOS sensors allow the modification of quantization scaling factor.

- After this the coefficient data is converted from 8*8 to a 64*1 vector and the huffman encoding scheme is used to encode the data. This does not lead to any loss in data and the process is completely lossless.

The entire JPEG compression process is shown in Figure 2.7. The process is followed in the reverse order in-order to obtain back the original RGB image and is supported by all image viewing software.

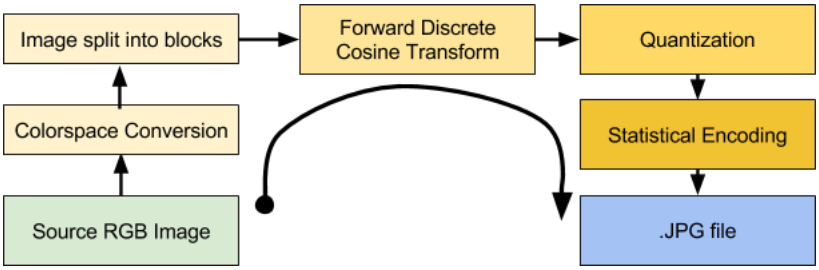


Figure 2.7: Steps involved in JPEG Compression[4]

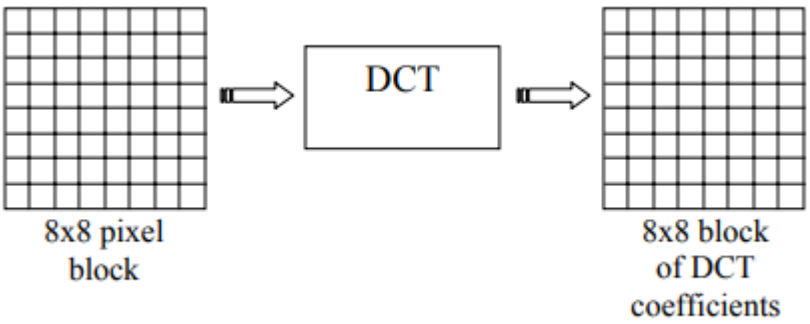


Figure 2.8: DCT involved in JPEG[39]

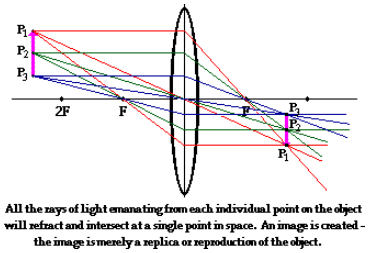
2.4.3 Masks and Reconstruction Algorithms

The main aim of designing a lensless camera is that it would help reduce the size of the camera considerably. In a lensed camera, light emanating from multiple-points in the object would intersect to form the image of the object. This is illustrated in Figure 2.9a. the focal length of the lens-involed primarily determines the thickness of the camera(See Figure 2.9b). In a lensless imaging system, the lens is replaced by means of a mask(series of pinholes arranged in a periodic manner). This is illustrated in Figure 2.11.

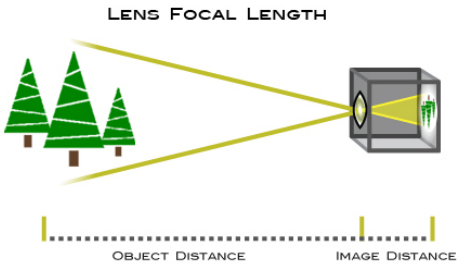
The simplest lensless imaging system is the pinhole camera. However, since the quality of the image depends on the size of the pinhole, that restricts the amount of light that can enter the imaging system. Lenses were introduced to focus the light from distant objects onto a film or a sensor. In the absence of a lens, the sensor would record the average intensity of the light entering it. This can also bee seen in the experiments which are described in the upcoming chapters.

In 1960s, Fernimore and Canon introduced Fresnel Zone Plates for a "large aperture, high resolution camera with neither refracting or reflecting components"[20]. The fresnel zone plate is defined such that the radius of the n^{th} zone is given by

$$r_n = r_1\sqrt{n} \tag{2.6}$$



(a) Image Formation in a lens-based camera[5]



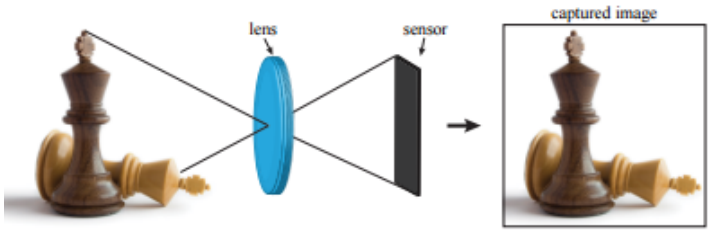
(b) Image Distance dictated by focal length [12]

Figure 2.9: Lens-based Camera

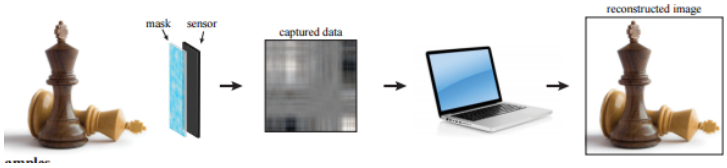
An example of Fresnel Zone plate is shown in Figure 2.11a. Fresnel Zone Plates can be used in the places of lenses to produce images because FZPs produce images at multiple higher order foci depending on the type of fresnel plate used. The fresnel zone plates have a transmission of exactly 50 percent. It is always desirable to have an optical system with ideal system point spread function(SPSF). The point spread function describes the response of an optical system to a point source. It can also be called as the impulse response of the optical system. The ideal point spread function of an optical system would be a dirac-delta function. However, in-order to obtain such response with a fresnel zone plate, it would have to be infinite. So, different forms of coded-apertures were developed to overcome the limitations associated with fresnel zone plates.

Coded aperture cameras extend the pinhole camera concept replacing the single aperture with a mask containing multiple apertures. The first developed coded aperture cameras were used in imaging X-Ray sources due to the difficulty involved in focussing light from X-Ray sources[20]. Since a single pinhole limits the amount of light imaged by the sensing element, it was replaced by many holes, called the aperture so that overlapping images are formed on the film. The recorded image will have no similarity with the source and an digital processing is required to reconstruct the source image or the object. The recorded image is mathematically modelled as a collection of overlapping shadows as described by the following equation[19][20]

$$y = \phi * x + e \tag{2.7}$$



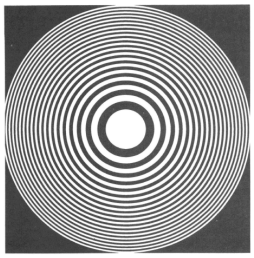
(a) Conventional Lensed Imaging



(b) Lensless Imaging

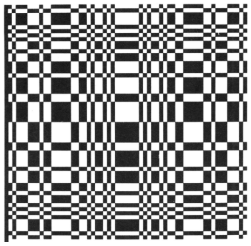
Figure 2.10: Difference between lensed and lensless imaging methods[19]

where y represents the image formed on the sensor, ϕ represents the mask pattern, x represents the irradiance vector or the object and e represents the noise. The $*$ operator represents the convolution operation between the mask an the object. The coded aperture increases the flux that falls on the detector and this leads to an increase in the SNR. The SNR can be as large $\sqrt{N}$, where N represents the number of holes in the aperture[20]. The increased SNR comes at the cost of computational decoding for the image. We can obtain the actual object image x using the inverse of the mask function. An ideal point spread optical function would produce a dirac-delta response $\phi * \phi^{-1}$. In 1968, Dicke and Ables[20] introduced random arrays as am alternative coded aperture imaging method. The mask consists of randomly positioned holes in an opaque surface. An example of a random array is shown in Figure 2.11b. The total open area may or may not be equal to the amount of opaqueness. The image produced by random arrays are decoded digitally using auto-correlation analysis of the encoded images[20]. Like Fresnel Zone Plates, the random array also exhibits an ideal response when the size of the array is infinite which is practically impossible. A new-class of apertures called uniformly redundant arrays(URAs) were developed over the disadvantages associated with random arrays. The uniformly random arrays exhibit a pattern that follows the equation 2.9[27]. The URA have a dimension of r by s with $r - s = 2$. The quation 2.9 takes $I = mod_r i$



(a) A twenty ring Fresnel Zone Plate

(b) A 60*60 Random Array



(c) A 61*59 Uniformly redundant array

Figure 2.11: Masks developed initally for lensless imaging[20]

and $J = mod_r j$(i and j represent the array index position in x and y).

$$A(I,J) = \begin{cases} 0 & \text{if } I = 0 \\ 1 & \text{if } J = 0 \ \& \ I \neq 0 \\ 1 & \text{if } C_r(I)C_s(J) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

where

$$C_r(I) = \begin{cases} 1 & \text{if there exists an integer } x \text{ such that } 1 < x < r \text{ such that } I = mod_r x^2 \\ -1 & \text{otherwise} \end{cases} \tag{2.9}$$

The matrix used for deconvolution is given by the equation 2.10. This kind of array produces an ideal point spread function. The point spread

function of different masks discussed above is shown in Figure 2.12.

$$G(I,J) = \begin{cases} 1 & \text{if } A(i,j) = 1 \\ -1 & \text{if } A(i,j) = 0 \end{cases} \tag{2.10}$$

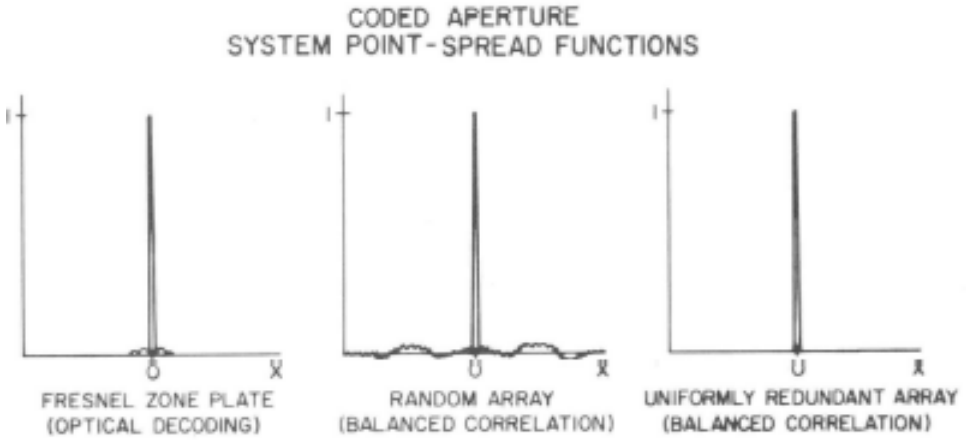


Figure 2.12: Point Spread function of different masks[27]

It can be seen that the Uniformly redundant arrays would offer the best performance and would produce the ideal point spread function. The direct inversion by deconvolution with inverse works when the object is against a dark background with negligible diffraction effects(for example, in the X-Ray Spectrum). However, in the later studies[22], it can be seen that equation 2.7 no longer holds in the visible light spectrum involving extended-object scenes and direct fourier/deconvolution based reconstruction methods would fail. This is also observed in the simulation studies described in the upcoming chapter. In order to solve the problems associated with reconstructing extended object scenes in the visible light spectrum, a new class of separable Doubly Toeplitz mask was developed by Michael et.al [22]. These classes of masks retain their properties even in the presence of diffraction and extended-object scene based scenarios. The doubly toeplitz-masks are expressed as a product of two independent vectors $A(i)$ and $B(j)$.

$$M(i,j) = A(i)B(j) \tag{2.11}$$

If the mask can be expressed in the form of two independent vectors, then the equation 2.7 can be re-written in the form:

$$I = M_A O M_B^T \tag{2.12}$$

The matrices M_A and M_B are toeplitz meaning they follow the form:

$$M = \begin{bmatrix} A1 & A2 & \cdots & AN & 0 & 0 & \cdots & 0 \\ 0 & A1 & A2 & \cdots & AN & 0 & \cdots & 0 \\ \vdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \end{bmatrix}$$

Figure 2.13: An example of Doubly-Toeplitz Mask[22]

Previous studies[22][19][13] have proven that it would be possible to perform lensless imaging in the visible light spectrum for real-life object scenes. The detailed solution and modelling of the computational algorithm would be discussed in the next chapter. This kind of mask would be ideal for our application. There is also another class of masks with hexagonal shapes[30][25] which have been used for coded aperture imaging. However, these have not been studied since they offer no specific advantage compared to other masks discussed above.

Chapter 3

Simulation of Reconstruction Algorithm

This chapter will describe how the system can be mathematically modelled and how the mask for the lensless imager was designed. As mentioned in the previous chapter the system can be modelled as

$$y = \phi * x + e; \tag{3.1}$$

Ignoring the noise and converting the equation to fourier domain, the equation 3.1 can be re-written as

$$F(y) = F(\phi)F(x) \tag{3.2}$$

$$F(x) = \frac{F(y)}{F(\phi)} \tag{3.3}$$

$$x = F^{-1}(\frac{F(y)}{F(\phi)}) \tag{3.4}$$

Equation 3.4 is the simplest possible computational inversion of the scene from the sensor. This method has also been used in [22]. As mentioned in the previous chapter, there are two types of masks that can be used for the purpose of encoding the scene onto the mask, namely separable and non-separable mask. MATLAB has been used for the purpose of simulating the algorithms. In this chapter, we would simulate two types of mask patterns, namely separable and non-separable mask patterns. A separable mask pattern is one in which the mask matrix ϕ can be expressed in-terms of two sub- matrices:

$$\phi = \phi_L * \phi_R \tag{3.5}$$

Both ϕ_L and ϕ_R are doubly-toeplitz mask. The mathematical model is also changed as described by the equation 2.12. A non-separable mask is one which does not follow this property and follows equation 3.1.

3.1 Simulation of a non-separable mask

The non-separable mask simulation was carried out as shown in Figure 3.3. Due to the inherent ill-posed mathematics involved in imaging extended scenes, a regularization term is added to the inversion process and we get equation 3.6. This kind of regularization is called Tikhonov regularization and it regularizes the inversion and controls the effects of noise[22]. The same regularization is also used in previous studies[22].

$$x = F^{-1}(\frac{F(y)}{F(\phi) + k}) \tag{3.6}$$

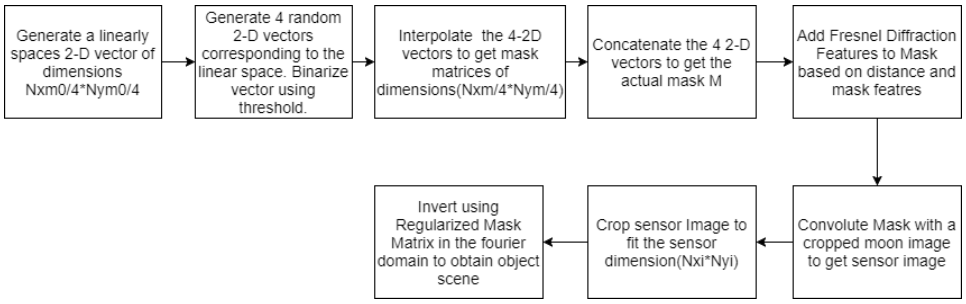


Figure 3.1: Simulation-Flow non-separable mask

In order to start with the mathematical modelling process and to imitate the sensor data and reconstruction, a reference image is needed. For that, it was decided to use the full moon image captured by the Apollo 11 space craft[3]. Since the satellite is going to be pointing towards astronomical objects like the earth, moon, it was decided to crop out a portion of the full moon image as the main purpose of the satellite would be to image specific portions of earth. The simulation is done under the assumption that the camera is enclosed in a box-like structure and light from specific region of the earth/moon would reach it and the sensor size is finite. The image was converted to gray and scaled down form 0 to 1 and is displayed in the `bone` colormap format available on MATLAB as th colormap would display the minute variations in the reconstructed image. This is illustrated in Figure 3.5. The simulation is performed with and without diffraction and analysed. The mask used for simulation is shown in Figure 3.3. The simulation parameters are shown in Table 3.1. The reconstruction error is given by the equation 3.7. O_{guess} indicates the reconstruction and O represents the original object. Fresnel diffraction features are added to the mask since the distance between the mask and the sensor lies in the fresnel diffraction region and the fresnel number for the mask is less than 1. The fresnel diffraction modelling is described in the appendix section of the report.Diffraction

Figure 3.2: The first image is the orginal image of the moon as taken by Apollo 11 spacecraft. The second image indicates the cropped region and the third image indicates the region that is used for the simulation.

Table 3.1: Simulation Parameters

Pixel Size	$2.2\mu m$ * $2.2\mu m$
Sensor Size	512 * 512 pixels
Mask Size	1024 * 1024 pixels
Mask Sensor Distance	5 millimeters
`Nxm0, Nym0`	256, 256

causes the mask to become non-binary.

$$PSNR = 20log(\frac{N * max(O)}{MSE}) \tag{3.7}$$

where

$$MSE = \frac{1}{mn}\sum_{i=1}^{i=N}\sum_{j=1}^{j=N}[O(i,j) - O_{guess}(i,j)]^2 \tag{3.8}$$

The reconstructions obtained using equation 3.6 is shown in Figure 3.6. It can be seen from the figure that we have successful reconstructions using equation 3.6. The value of k was set to 0.01. However, when the diffracted mask is modelled for the sensor image, the reconstruction fails. This is the reason why we need to go for the separable mask as this mask is not resistant to diffraction effects.

3.2 Simulation of a separable mask

As we saw in the previous section, a non-separable mask cannot be used for our application in the visible light region as the visible light spectrum contains significant diffraction effects. We can improve the solution since a

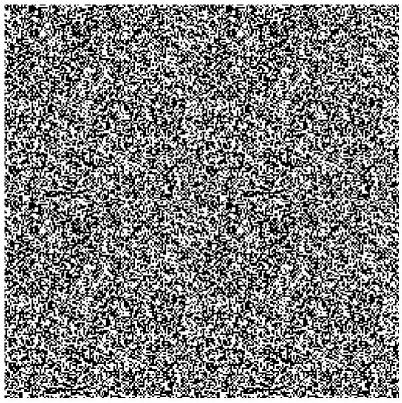


Figure 3.3: Non-separable mask used for simulation

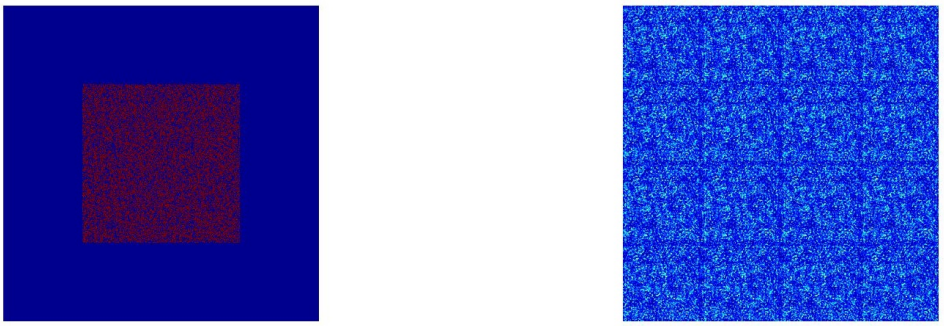


Figure 3.4: The mask image on the left indicates undiffracted mask and the mask on the right indicates the diffracted non-separable mask

separable mask follows the equation 2.12. The same improved solution has been used in [22]. We can express 2.12 as shown in equation 3.9.

$$M_A^T I s M_B = (M_A^T M_A) O (M_B^T M_B)^T \tag{3.9}$$

We multiply the left and right system matrices to obtain square symmetric matrices. The problem is that they are not invertible, due to the presence of zero eigenvalues. So, the parameters α_A and α_B to remove the zero eigenvalues and make them invertible. The final solution will be given by the equation 3.10.

$$O_{Guess} = (M_A^T M_A + \alpha_A^2 1_{R_O})^{-1} M_A^T I M_B (M_B^T M_B + \alpha_B^2 1_{C_O})^{-1} \tag{3.10}$$

One more advantage of using the separable mask is the reduced amount of computation needed compared to the non-separable mask. The number of computations for a non-separable mask will be given by the equation 3.11. For a megapixel image, this would translate to 10^{18}.

$$N_{operations} \propto (r_O * c_O)^3 \tag{3.11}$$

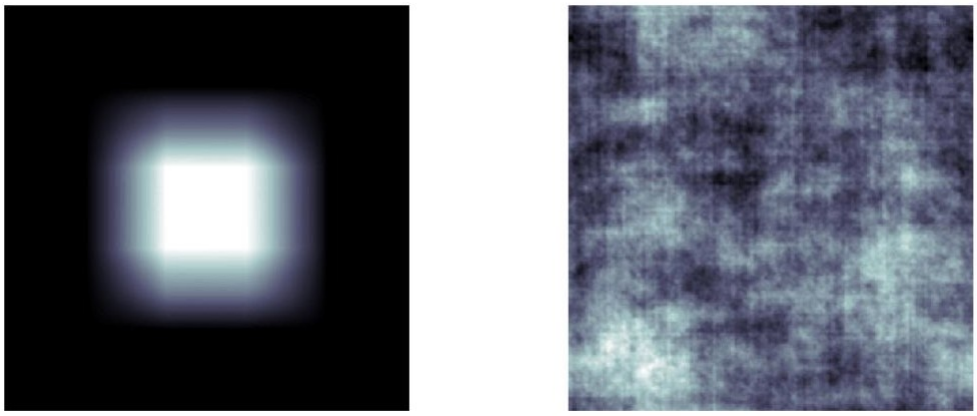


Figure 3.5: The first image indicates the sensor image if the sensor plane was infinite. The second image indicates the cropped sensor image that would be formed on an actual finite sensor size(cropped).

Table 3.2: Table indicating the rows and columns used for simulation

Matrix	Rows	Columns
Mask Size	r_M	c_M
Image on Sensor	r_I	c_I
Object Area contributing	$r_O = r_M + r_I - 1$	$c_O = c_M + c_I - 1$
Left Toeplitz Mask	r_I	r_O
Right Toeplitz matrix	c_I	c_O

For a separable mask however, the number of computation is drastically reduced because the size of the left and right system matrices is reduced to $(R_I * R_O)$ and $(C_I * C_O)$ from multiplying the entire object matrix of size $r_O * c_O$. This would come to around $2 * 10^9$ operations.This can also be observed in the simulation and it takes an extremely long amount of time to complete for a non-separable matrix. The size of different matrices used in simulation is shown in Table **??**.

$$N_{operations} \propto [(r_I * r_O)^3 + (c_I * c_O)^3] \tag{3.12}$$

The separable mask simulation was carried out as shown in Figure 3.7. It is possible to generate different types of Toeplitz mask by simply changing the simulation variable `Nxm0`. These masks follow the same property and offer the same amount of reconstruction in simulation. By changing `Nxm0`, the base vector that is used for interpolating to et the bigger mask changes.

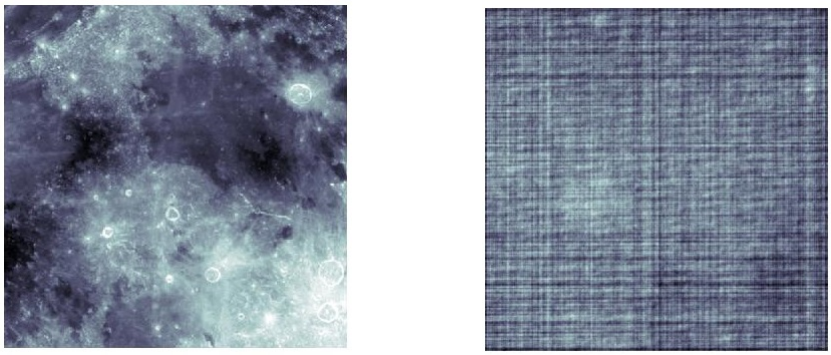


Figure 3.6: the figure shows the reconstruction with and without fresnel diffraction effects for a non-separable mask using equation 3.6. We are able to obtain PSNR = 57.85 without diffraction effects. However, the reconstruction with the mask fails when diffraction effects are incorporated into the mask.

This lead to different masks begin generated. The shape of the mask is more simpler when we use a smaller `Nxm0` as a very small vector is used for interpolation to get a bigger mask. The transmittance of the masks vary between 20 to 25 percent. The different masks obtained by changing `Nxm0` is shown in Figure 3.8. By changing the `Nxm0`, we can basically change the feature size of the mask. A lower `Nxm0` generates a mask where the subsequent binary value comes at a greater distance than the one with a larger `Nxm0`. It can be seen from figure 3.7 that there as an additional step involved in the simulation process of separable masks which is the rank-1 estimation of diffracted matrices. In this step, we perform Singular Value Decomposition(SVD) on the diffracted matrices to obtain M_x and M_y. Now let us see what is singular value decomposition. Singular value decomposition is a mathematical tool that would enable us to express any matrix of the form $M_{\times n}$ in the form given by equation 3.13[35].

$$M_{m\times n} = U_{m\times r} S_{r\times r} (V_{n\times r})^T \tag{3.13}$$

where r represents the rank of the matrix M , m and n denote the size of the matrix respectively. We can estimate the rank-1 estimation of a matrix by taking the first row and column of U and V and multiplying them as shown in equation .

$$M_1 = U_1 S_1 (V_1)^T \tag{3.14}$$

One of the main advantages of using a doubly toeplitz mask is that it is decomposable in to a single rank matrix with and without the effects of diffraction. The other rank-components are negligible even in the presence of diffraction. This is an important property that must be kept in mind which

will be used for various experimental purposes. In the simulation, we use SVD on the diffracted mask, take the rank-1 estimate of of the mask. The U_1 and V_1 will represent the left and right system matrix of the refracted mask. The obtained single dimensional U_1 and V_1 of the doubly toeplitz mask will be converted into toeplitz matrices and inverted using equation 3.10. The decomposition of the diffracted and non-diffracted separable mask into a 1-rank matrix(with other components negligible) was also verified in the simulations. The difference between diffracted and non-diffracted separable mask is shown in Figure 3.9.

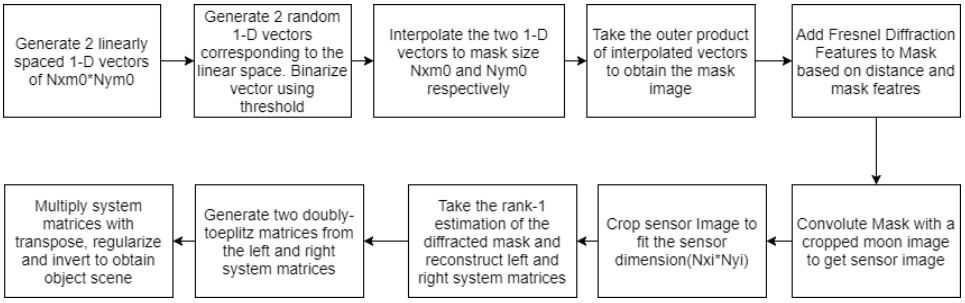
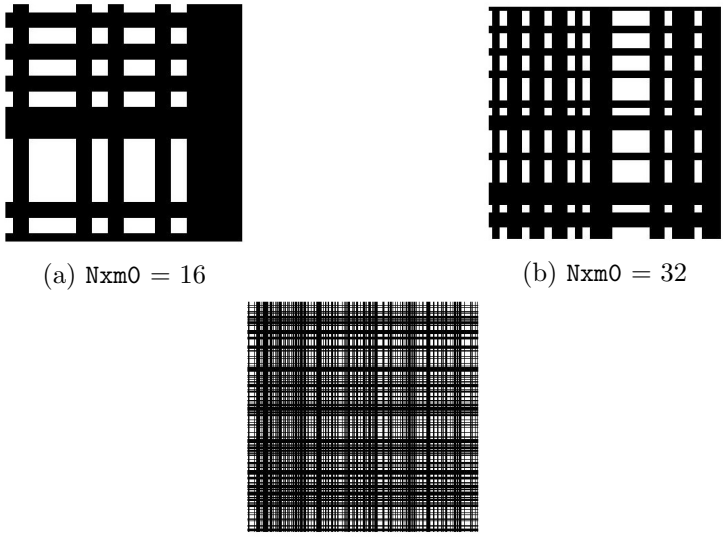


Figure 3.7: Simulation-Flow separable mask



(a) `Nxm0` = 16

(b) `Nxm0` = 32

(c) `Nxm0` = 256

Figure 3.8: Different separable masks generated with different `Nxm0`

It can be seen from Figure 3.10 that even in the presence of diffraction it would be possible to obtain reconstructions using equation 3.10 by using a separable mask unlike a non-separable mask which fails to provide any reconstruction in the presence of diffraction effects. It was seen that the

Figure 3.9: The mask image on the left indicates undiffracted mask and the mask on the right indicates the diffracted separable mask

separable mask would work best for our application and we can safely assume that we can use these masks to obtain reconstructions for extended object imaging in the visible light spectrum as the diffraction effects have also been taken into account.

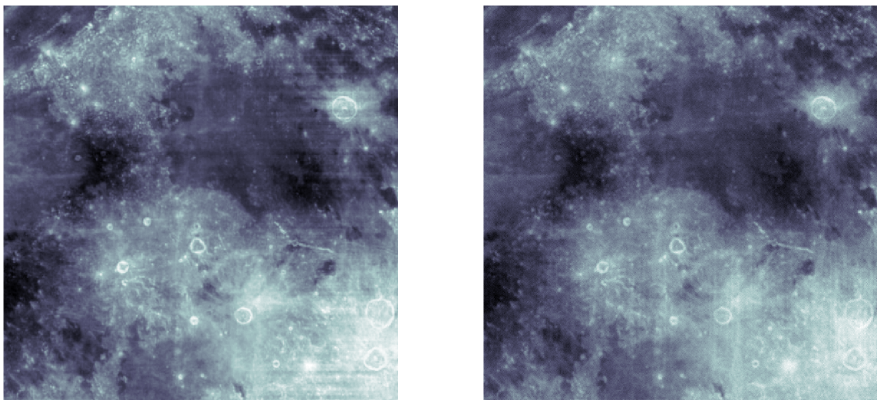


Figure 3.10: The figure shows the reconstruction with and without fresnel diffraction effects for a separable mask using equation 3.10. We are able to obtain PSNR = 57.85 without diffraction effects. It can be seen that we can obtain reconstructions even in the presence of diffraction effects with PSNR = 57.93

Chapter 4

Implementation

In this chapter, we will be discuss about the implementation of hardware and software of the camera that would be used for subsequent experiments. The main specifications of the chosen cameras are shown in Table 4.1. These specifications are taken from the sensor data sheet and the camera manual provided by the vendors of the camera[16][17][36][38].

4.1 Embedded Software of Camera

One of the main reasons behind choosing the OV2640 and OV5642 CMOS sensors is that they already have a ready electronic interface that can be used to interface with standard 8-bit/16-bit microcontrollers. Arducam is an open source camera that comes along with open-source hardware and software that is needed to capture images using the CMOS sensor. One of the core components in every camera made by Arducam is that there is a component called Arduchip[15]. It is a basically an Altera MAXII CPLD

Table 4.1: Key Specifications Specifications of sensors Chosen

Specification	OV2640	OV5642
Voltage Level	3.3/5V	3.3/5V
Frame Buffer Size	384 KB	8MB
Active Pixel Array Size	1600×1200	2592×1944
Camera Dimensions	34mm × 24mm	34mm ×24mm
Sensor Dimensions	5725 μm × 6285 μm	6945 μm × 6695 μm
Pixel Size	2.2 μm × 2.2 μm	1.4μm × 1.4 μm
Weight	20 grams	20 grams
Operating Temperatures	-10 to 55 degree celsius	-10 to 55 degree celsius
Electronic Interfaces Needed	SPI, I2C	SPI, I2C
Dynamic Range	50 dB	68 dB

EPM240 processor that facilitates DMA memory transfer between the camera memory module and components such as microcontroller and thereby helping to reduce the development time for special applications such as space where we the only computational resource that would be available are low power computing platforms such as microcontrollers with only synchronous serial interfaces such as I2C, SPI, etc.

However, using the camera comes with its own advantages and disadvantages. The main advantage behind using this platform is that the platform has open-source libraries that could be used to interface with ATMEGA328P, an 8-bit microcontroller. In space-missions, it would not be possible to send high-powered microprocessors, and microcontroller is used as an on-board computer. Arducam has standard software libraries that can be used to interface with Arduino making the cumbersome and lengthy job of writing an interface software to a CMOS sensor way more easier. A disadvantage of the hardware module is the on-board memory that it has to capture an image. The OV2640 Arducam mini camera module can capture upto 1600×1200 resolution images with or without any form of compression. However due to the limitation of the on-board OV2640 FIFO memory AL422B it would be possible to capture only compressed images and not full resolution RAW images. The AL422B on-board FIFO has only 384KB of memory and that is not enough to obtain a full-resolution RAW image. One of the other disadvantages is that custom code needs to be written to obtain various controls that we need for our camera. We have to write our own camera control software if we need to control factors such as exposure time, ISO, etc. as the default software uses automatic exposure control to enhance the image quality. The camera module architecture is shown in Figure 4.1. The OV5642 offers a greater flexibility in terms of memory as it has a larger FIFO Buffer(8 MB).

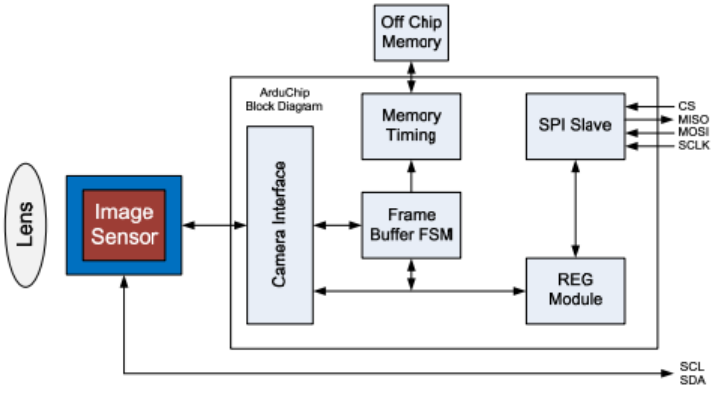


Figure 4.1: Camera Architecture of Arducam Mini OV2640 Camera Module

The system for experiments is as shown in Figure 4.2. The Arduino is

connected to the camera module through an I2C interface. Using the I2c interface it possible to set registers that control the functioning of the camera such as the output format, digital signal processing, etc. SPI interface is used to transfer the image data from the camera module to the Arduino. The Arduino upon receiving the image data either writes it to an SD card or sends it to the software on the PC through the USB connection. The software flow is shown in Figure 4.3. We use the same software flow for both the CMOS sensors as the APIs are designed for use with many sensor models. The same set of libraries can be used for different CMOS sensors by simply adding some preprocessor directives to a `memorysaver.h` header file provided in the open-source library. To see the images on the PC, the vendor has provided a host PC software(not-open source) and firmware which uses USB-UART to transfer images from the Arduino to PC. The firmware could be modified to ignore certain commands coming from the PC and using this we were able to modify the camera settings to our advantage.

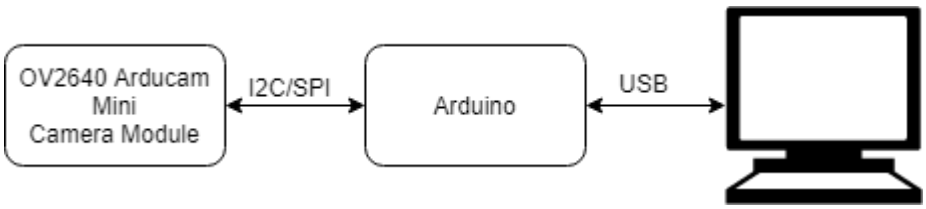


Figure 4.2: Implementation Setup

One of the important factors that need to be controlled in camera is the exposure time. The Arducam libraries provide APIs for 10 different exposure levels for the OV5642 sensor. However, no such APIs are available for OV2640. Since OV2640 consumes lower power, it was decided to start the experiments with this camera. So, we need to write custom software to control the exposure level of this camera. The arducam APIs used for the software are shown in Table 4.3. The details of more APIs can be found in the software application notes[18]. The default camera settings were used with only modifications to exposure in the case of OV2640 module. The CMOS sensor can be directly accessed using `wrSensorReg16_8, wrSensorReg8_8` functions which provide I2C bus access to the CMOS sensor.

4.1.1 Exposure Control of OV2640

In order to do experiments, it was required to control the exposure of the camera. In the default driver that was provided by the vendor, the exposure was automatically set using the Automatic Exposure Control (AEC) feature in the sensor. So, a modification was needed in the driver software. Fortunately, the driver is open source and the there were libraries that could assist in setting the on-board registers through the I2C interface on the Ar-

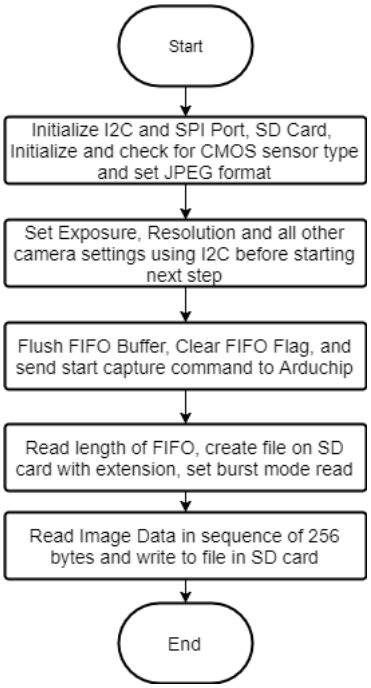


Figure 4.3: Software Flow used for image acquisition from CMOS sensors

duino. First, let us have a look at how exposure control works in an OV2640 camera. All rolling shutter image sensors including OV2640 exposure the sensor one-line at a time i.e. pixels in the same line are exposed at the same time and different pixels in different lines are exposed at a different time. So, the minimum exposure time would be one line time and the maximum exposure time would be the frame time. This is illustrated in Figure 4.4. By default the pixel clock is set at 36MHz. We can calculate the minimum line time using the following equation:

$$MinimumExposureTime = 1/PixelClock * PixelClockesperline$$

As shown in Figure 4.5, one line consists of 1922 pixel clocks(1600 for pixel data and 322 clocks of horizontal blanking). So the minimum exposure time would be 53.39μseconds and the maximum exposure time would be the frame time(multiply line time by 1200 + 44 lines of vertical blanking) which would be 66.63ms[8]. In order to control the exposure of the camera it is necessary to modify registers of address 4, 10, 13, 45. So, these registers were modified according the the required exposure time value. The driver software on the Arduino was modified to obtain different exposure times.

Table 4.2: Details of Main APIs used

API	Function
`InitCAM`	Initialize Camera Module
`set_format`	JPEG an BMP output formats can be selected
`clear_fifo_flag` `read_fifo_length` `set_fifo_burst` `flush_fifo`	These functions are used to control the FIFO buffer, read the size of the image and to reset them when needed
`write_reg`	This function is used to write values to the specified register address on the Arduchip.
`wrSensorReg16_8` `wrSensorReg8_8`	These functions are used to set camera registers using I2C. The first function is used for 16-bit register addresses and the second function is used for 8-bit addresses
`OV2640_set_JPEG_size` `OV5642_set_JPEG_size`	These functions are used to set the resolution of the output image
`OV5642_set_Exposure_level`	This functions are used to set the exposure level of the OV5642 CMOS sensor

4.2 Power Consumption of Sensors

One important aspect that also needs to be taken into account is the power consumption of the camera module(including the sensor, memory, voltage regulators, etc). Since, we have two camera modules, OV2640(low resolution, low power) and OV5642(high resolution, high power). The power consumption profile for OV2640 was provided by the manufacturer, however, the same is not available for OV5642. So, it was decided to perform power consumption measurements for the sensors manually. In order to find the amount of power consumed, we connect a 1 ω resistor in series with the power line connecting the microcontroller and CMOS image sensor. The voltage measured across the resistor would be the current consumed by the camera sensors. The camera was operated in two modes: one in continuous "on" mode and the second mode was "on only when use". The second mode was activated clearing `GPIO_PWDN_MASK` when the camera is in use and

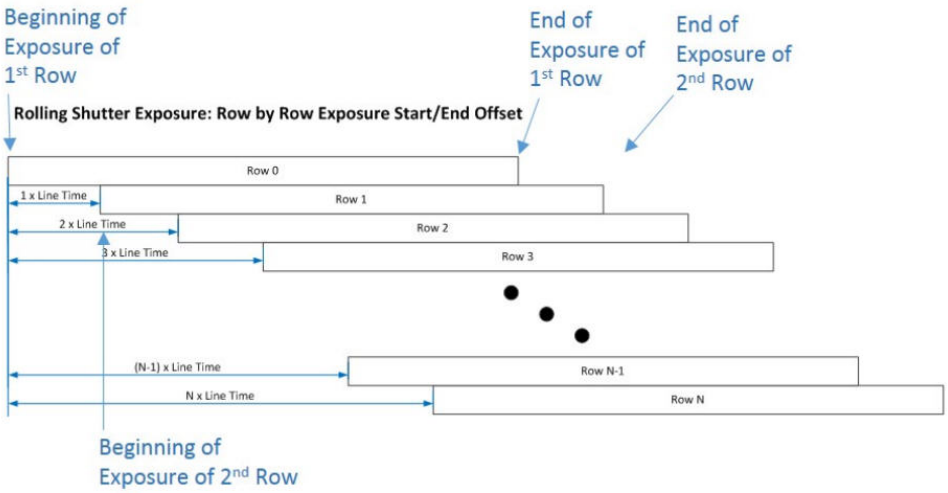


Figure 4.4: Rolling Shutter Operation on OV2640[8]

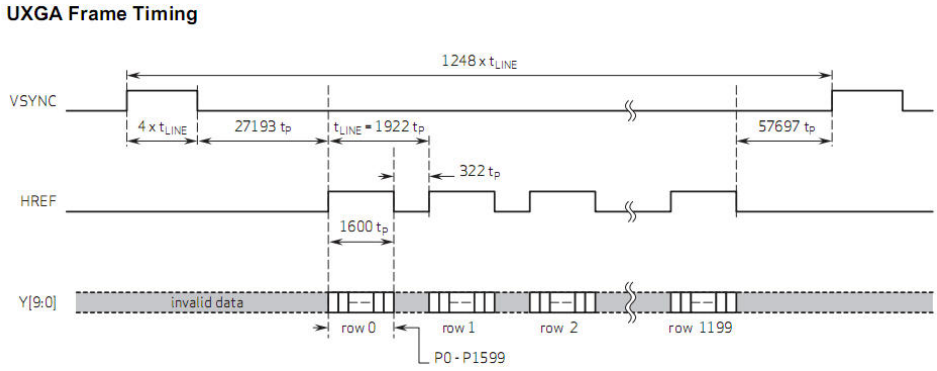
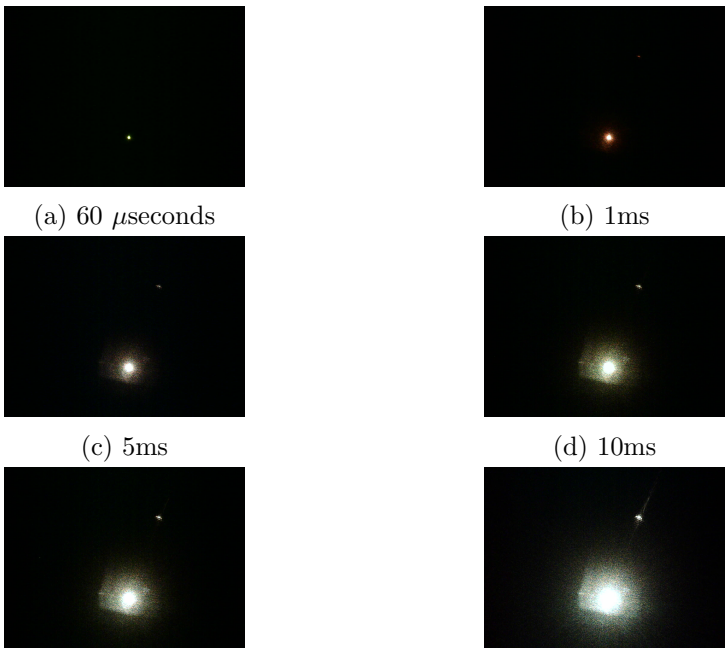


Figure 4.5: Shutter Timing Diagram of OV2640[8]

setting it once again once the camera is done taking the pictures and then saving it onto the memory card. The current consumption graph is shown in Figure 4.7.

It can be seen from Figure 4.7 that we can reduce power consumption by switching on the camera only when in use. The camera is operated in 5V mode. The current consumed by the camera is averaged and the measured power consumption is shown in Table 4.3. As expected, the higher resolution camera OV5642 consumes more power than OV2640. Since, OV2640 is consumes lower power and is also of lower resolution(which implies lesser size), we proceed to continue with the next set of experiments using OV2640.



(a) 60 μseconds (b) 1ms

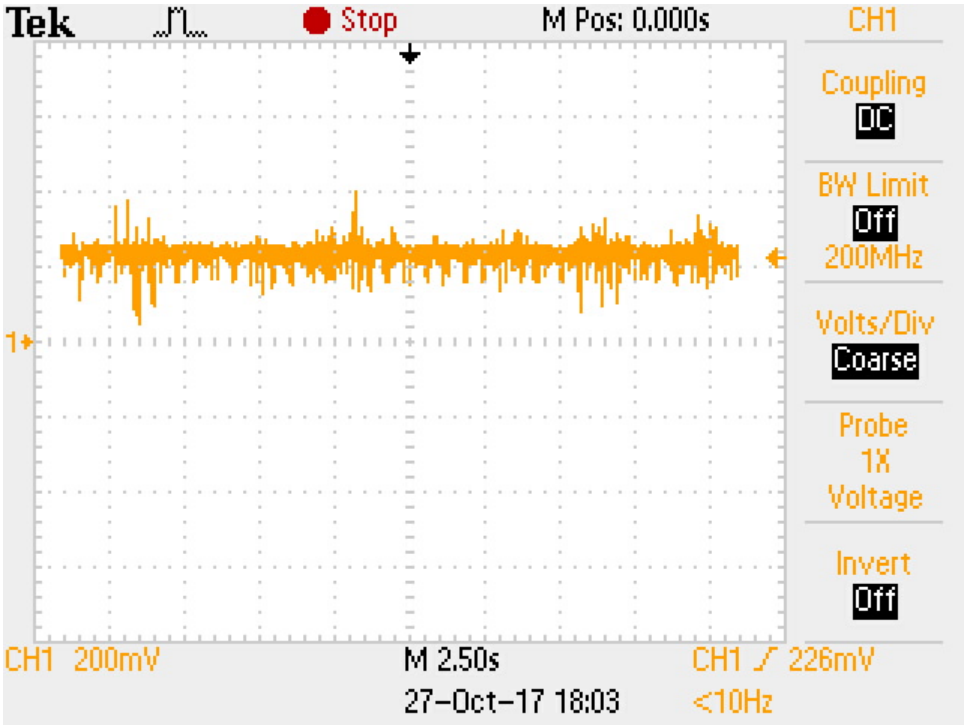
(c) 5ms (d) 10ms

(e) 20ms (f) 60ms

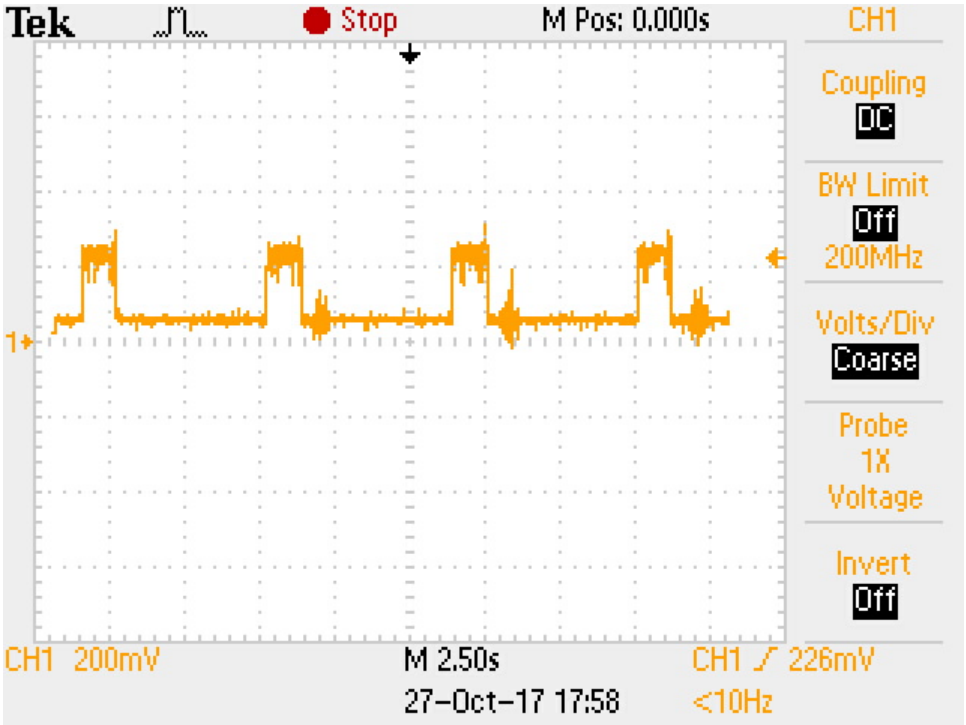
Figure 4.6: Images of a laser beam caught in different exposure times(with lens)

Table 4.3: measured power consumption of Cameras only when "on"

Sensor	Power
OV2640	5V/93mA
OV5642	5V/233mA



(a) Voltage across resistor when continuosly on



(b) Voltage across resistor when using `GPIO_PWDN_MASK`

Figure 4.7: This figure shows the difference between keeping the OV5642 sensor module continuously on and using the `GPIO_PWDN_MASK`

Chapter 5

Experiment to determine Acceptance Angle of CMOS sensor

5.1 Testing of Acceptance Cone of Sensor

5.1.1 Experimental Setup

A study conducted previously[41] has employed cat's eye reflectometer method using four spherical lenses to accurately measure the chief ray angle of a CMOS sensor upto 0.65 degrees[41]. Apart from this, Armstring Optical another method for measuring the chief ray angle or the acceptance angle[1] is proposed. However, our work employs a simpler method to measure the acceptance cone of a CMOS sensor without use of any lenses. Our method uses a collimated laser beam(650 nm), a pinhole for finding the accurate center of axis and a rotational stage(to measure angular rotation) to find the response of the sensor to light at different angles and in-turn the acceptance cone of the sensor. Figure 5.1 shows the experimental setup that was made to measure the acceptance cone of the sensor. A collimated laser beam was reflected off a mirror and it is passed through a pinhole. The pinhole was used because a diverged wave did not provide much information about the distribution of energy of the signal. Hence, a pinhole was introduced to focus the beam onto a reduced area on the sensor. The difference in measurements with and without pinhole is shown in Figure 5.2.

The pinhole cannot be used to measure the acceptance cone of the sensor because the waveform from the pinhole is a combination of different plane waves of different frequencies. Hence, the measurement was done without a pinhole. In order to measure the acceptance cone of the sensor, it is necessary to identify a portion of the image that can be taken as a reference to plot the variation in the signal measurement. It was assumed that the

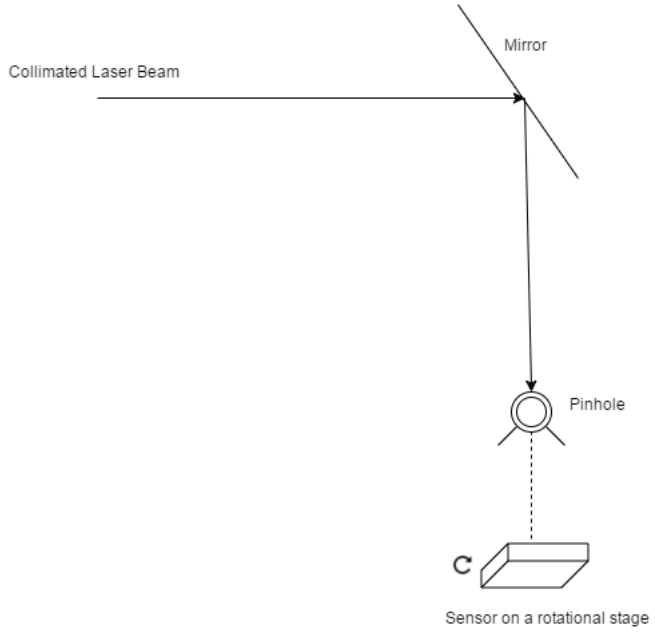


Figure 5.1: Experimental Setup for detection of acceptance cone

(a) Image without pinhole (b) Image with pinhole

Figure 5.2: Images of a laser beam caught with and without pinhole

maximum point in the image can be taken as the signal. Since there was too much variation in the intensities measured, it was decided that we would average the 10 images in each angular position to reduce the signal noise. The maximum value in the image was taken and normalized and the measurements were repeated from -45 to +45 for 10 sets. The same image readings were used for finding out the response of different color channels(RGB). This experiment took a period of two weeks to complete. The exposure value was chosen such that the output does not saturate(does not reach 255) for any value in the image(See Figure 5.3). The maximum value in the image versus the angle is plotted. The ideal exposure value would be the one in which the the signal does not saturate and there is still some room for additional signals. This is also indicated in Figure 5.3.

The obtained response curves for each channel are shown in Figure 5.4, 5.5, 5.6. The normalized signal values are plotted with the standard deviation

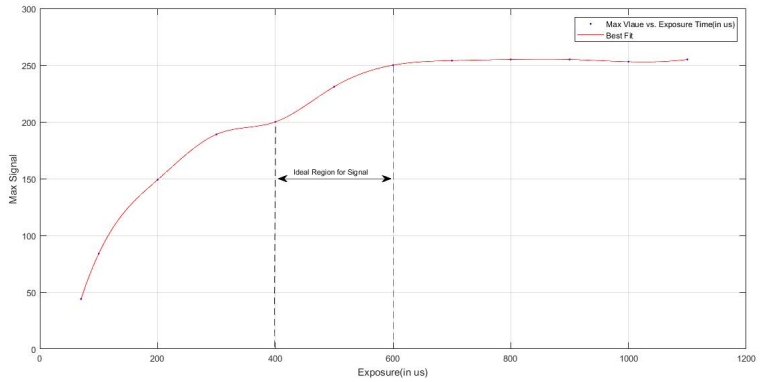


Figure 5.3: Response for different Exposure values in microseconds

obtained for each angle are indicated with the use of an error bar. The best obtained curve fit is also plotted.

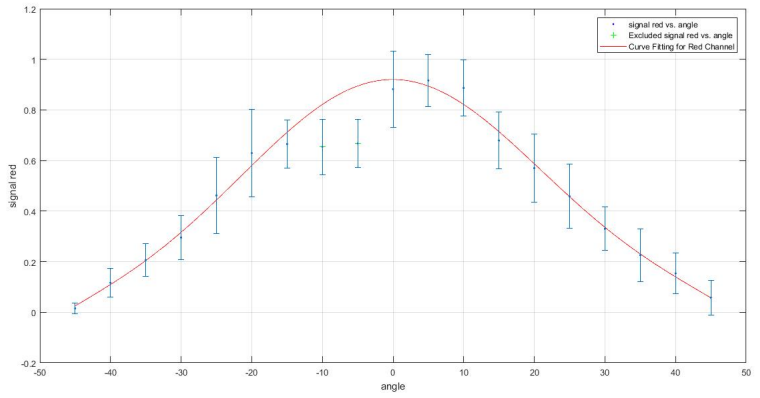


Figure 5.4: Response for Red Channel in Initial Experiment

The initial observations from the results obtained are:

- There seems to be a very large standard deviation for each angle.
- There seems to very sharp outliers at -5 and -10 degrees which is an highly unexpected behaviour.

These two observations point to experimental error. The error in the experimental results could be due to the following reasons:

- **Error introduced due to translation of the rotational stage :** In order to make sure that the beam always hits the stage is translated if the beam does not hit due to translation of the sensor away from the

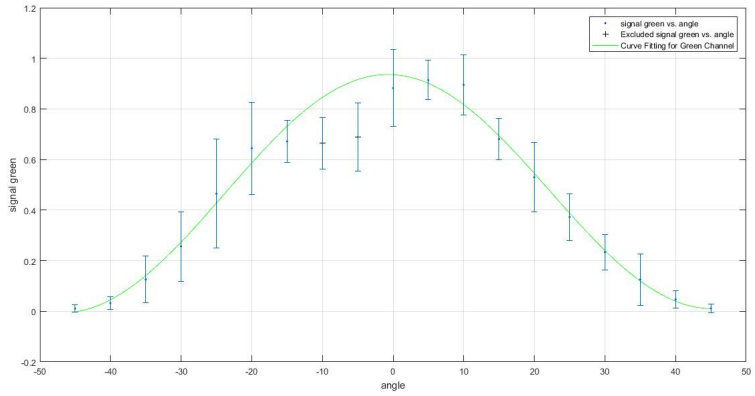


Figure 5.5: Response for Green Channel in Initial Experiment

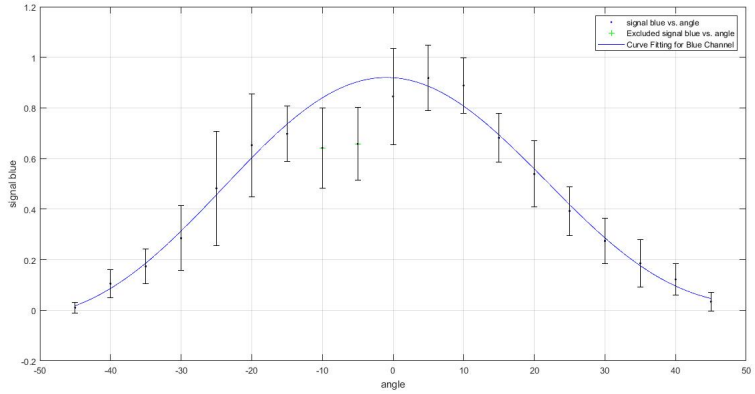


Figure 5.6: Response for Blue Channel in Initial Experiment

beam. This introduces an additional angular error which is not taken into account in the initial experiment. So, the stage was calibrated using the pinhole. If the sensor is exactly at the center of rotation of the rotational stage, the laser beam must always hit the same position of the sensor no matter what the rotational angle may be. The position of the sensor on the rotational stage was calibrated such that the signal(image with pinhole) obtained always remained at the center of the sensor.

- **Improper Reference Signal from Image :** At times, it was observed that the maximum point in the image occurs at different and unexpected points that are outside the beam. This also leads to different parts of the signal being measured each time. The outliers in the results could also mean that. An example average image in which

the maximum is obtained at the end of the beam is shown in Figure 5.7. The point from where the signal is obtained is marked using a yellow star in the Figure 5.7.

Figure 5.7: Improper Reference Point for Signal

- **Large Variation on intensity of output image :** It was noticed that there was a huge variation in intensity for subsequent measurements at the very subsequent instants. This variation in the output is the cause of the large standard deviation in the result. Initially, I thought that this could be due to a relatively low exposure time(as lower exposure could lead to more output noise) or due to the variation of intensity of laser beam. Increasing the exposure time did give lower variations but the problem was traced to the automatic gain control in the OV2640 image sensor. The amplifier gain of each pixel was adjusted automatically and this led to different intensities in subsequent readings. After this feature was turned off by modifying the driver software using I2C, it seemed that the variation in the output was no longer there.

- **Unexpected Colors in the output image :** An unexpected feature in the image is seen in each measurement. The unexpected feature is the presence of green and blue colors of the laser beam. Since the laser beam is red with almost constant wavelength in the red visible light region, the output of green is unexpected. This was very strange and upon studying the sensor, it seemed that the sensor had an automatic white balance(AWB) feature of OV2640. This white balance feature assumes a "gray" world(wherein the average of all colors in the world

is gray)[37] which is not true in our case. The difference in the output is with and without AWB is shown in Figure 5.8. Even after this adjustment, there seems to be a slight tinge of green at negative angles where the beam seems to "dim" out. Even after trying out a variation of different settings this green could not be eliminated and could be due to Black Level Calibration in the sensor. The function of Black Level Calibration (BLC) is to product accurate color in the dark area of picture. There is no mention of how to turn off this feature in the data sheet or the application notes of this sensor. The vendor of the camera did not provide any information on how to turn off this feature. Hence, it was assumed that the sensor does not make any modifications to the red channel in angles where the signal "dims" out.

(a) Image with AWB (b) Image without AWB

Figure 5.8: Images of a laser beam caught with and without pinhole

5.1.2 Improved experiment

As mentioned in the previous experiment, the stage was calibrated such that the sensor remains at the center of rotation of the rotational stage. The reference signal was chosen such that the same point of the beam is always measured. In order to make sure that the same point of the beam is measured with and without pinhole, the coordinate of the central diffraction pattern is stored for different angular positions for -45 to +45 and the reference signal is taken from this point. In order to detect the central region in the output, a function called `imfindcircles` is used[7]. This function finds circles in an image using using circular Hough transform. The readings for a specific angle are averaged and a logarithmic function is applied which is then passed to the `imfindcircles` function to detect the brightest possible circle with a radius of 3 pixels(This value was tuned such that the central portion is detected with 100 percent accuracy). The detection process using this this method provided 100 percent accuracy for detection of the center of the central fringe pattern coordinate. This is illustrated in Figure 5.9. The Automatic gain controls and Automatic White balance features of the CMOS sensors were turned off by using suitable register settings mentioned in [36]. The Arduino was programmed to set the register values using I2C.

The exposure was set at 500μs based on the exposure graph mentioned in the previous section.

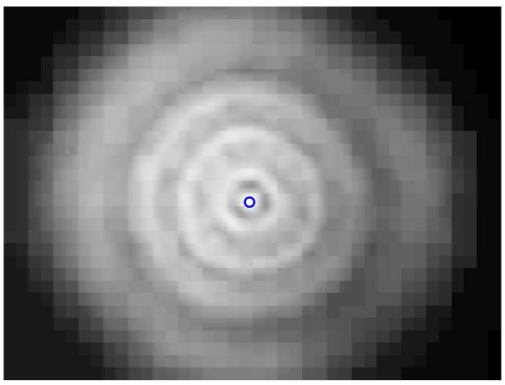


Figure 5.9: Coordinate Detection for central region signal detection

In the first step of the experiment, the rotational stage is rotated from -45 to +45 degrees with pinhole, and the coordinates of the central diffraction pattern is stored in a variable. In the second step of the experiment, the pinhole is removed and the signal is taken for different angle from the coordinates mentioned in the previous step. After this, the signal is taken with different offset(from the central diffraction pattern) in the X-direction to make sure that all the pixels behave in the same manner. The graphs for 100 different pixel positions from the central position is shown in Figure 5.10. From this graph,it can be seen that all the pixels in the same line exhibit the same behaviour with a slight shift in angular position peak. In figure 5.11, the response for different pixels in the Y-direction is shown. It can be seen that there is a wider variation in the curve peak position. It can also be seen that all the pixels follow the same pattern with a slight shift in the peaks.

The peak positions for different pixel offset positions in the X and Y direction is shown in 5.12. It can be seen that pixels in the same neighborhood have the same peak angular positions. The shift in peak is seen as we move across the detector. This can be attributed to the non-uniformity in the laser beam that is used for measuring the response of the pixels.

The experiment is repeated for 10 times and the response is plotted only for the red channel as we have only red frequency light hitting the sensor beam. The final result after solving the problems mentioned in the previous section is shown in Figure 5.13. There is less overlap in the subsequent angles and no points for excluded in the curve fitting process. The maximum

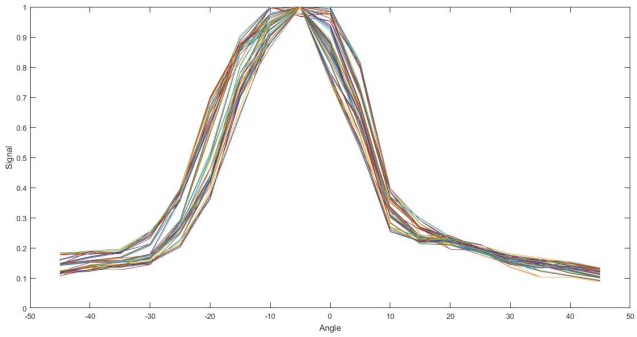


Figure 5.10: Response for 100 different offset pixel positions from central diffraction pattern in X-direction

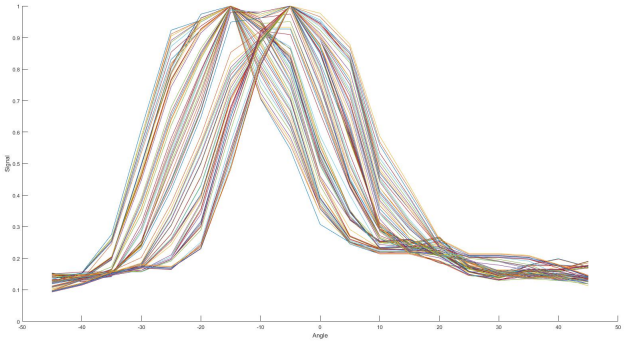


Figure 5.11: Response for 100 different offset pixel positions from central diffraction pattern in Y-direction

standard deviation has reduced from 16 percent for angular position -20 to 6 percent for angular position 0. The peak of the curve is shifted towards -5 degrees because the maximum position of the laser beam occurs at -5 degree position of the rotational beam. This data needs to be incorporated into the simulation to see the effect of the acceptance cone of the sensor on the image reconstruction. This will be discussed in the subsequent section.

5.2 Adding to Simulation

The curve data that was obtained in the previous experiment was incorporated into the previously obtained simulation results. In order to incorporate the data, we need to generate a 2-dimensional matrix that would simulate the behavior of the acceptance cone effect on the re-construction. Since the behaviour will be in both the x and y directions, the 1-d curve was converted into 2-d by multiplying itself with it's transpose. This would generate a

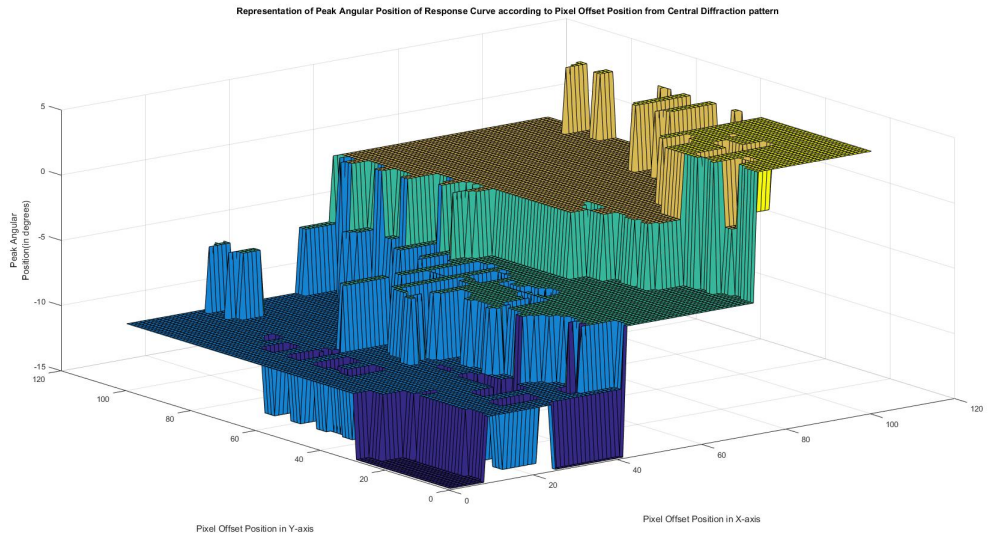


Figure 5.12: Peak Angular Positions for different Pixels

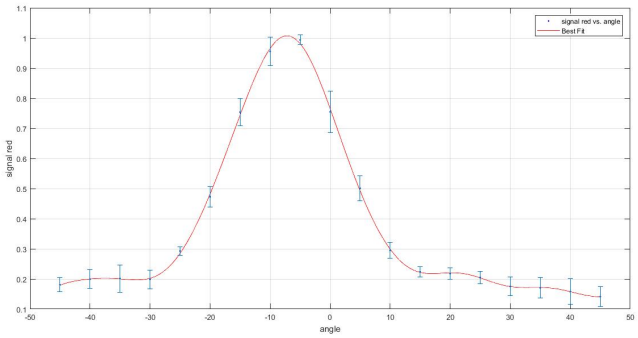


Figure 5.13: Red Channel Response for Different Data Sets

circular symmetric matrix that would generate the effect of the acceptance cone in both the x and y directions. This is shown in Figure 5.14. This matrix was multiplied with the original image to simulate the effect of angle acceptance on the image reconstruction. This was tried with the simulation results that simulate the effects of diffraction to simulate as close to a real world. The source image is shown in Figure 5.15. The reconstruction effects simulated after reconstruction is shown in Figure 5.16. It can be seen that most of the source image is reconstructed with slight loss in information only with diffraction. Once we incorporate the effects of acceptance cone into the simulation, it can be seen that only a portion of the original image can be reconstructed which is indicated by the rectangular box in Figure 5.17. This

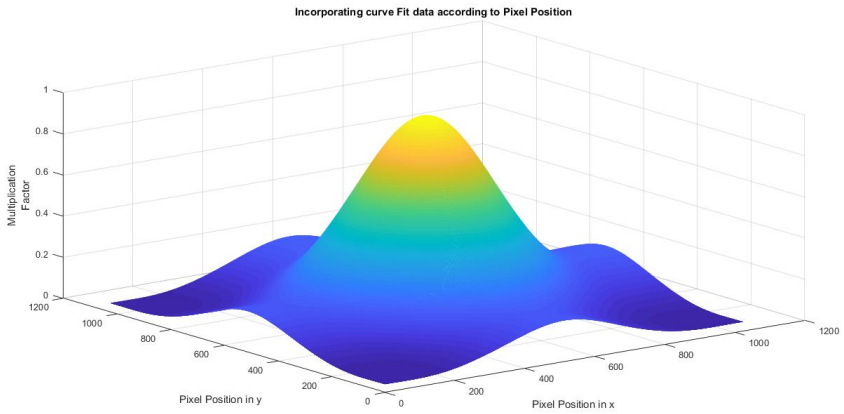


Figure 5.14: Fitted Curve Data into simulation

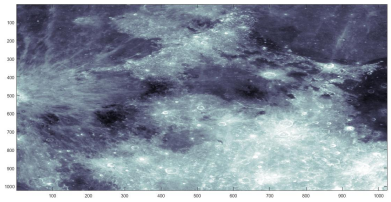


Figure 5.15: Source Image Cut from Moon

would be the actual field of view in the real world. The actual area of the sensor which can be used is drastically reduced by approximately 50 percent. In order to find the angular field of view, the positions of the rectangular box must be correlated with the fitted matrix curve discussed. On doing this, it can be found that the area in the rectangular box corresponds to an angle from -21.7 to +21.7.

A lens based system would have a larger field of view depending on the lens used. The OV2640 camera module by default comes with a 1.4 inch lens with a field Horizontal and Vertical Field of View of 70×63.7[16][14]. The

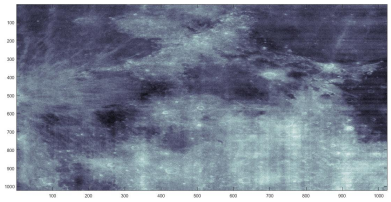


Figure 5.16: Reconstruction with Diffracted Mask

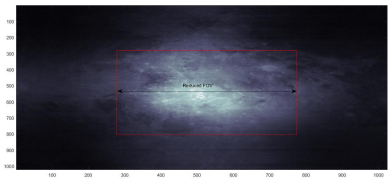


Figure 5.17: Reconstruction with Diffracted Mask and Curve Data

lensless system would have horizontal and vertical field of view of 43.4×43.4. This leads to a field of view reduction of 38 percent and 31.8 percent in the horizontal and vertical directions. The effective pixel are that can be used by the camera is reduced to 48 percent meaning that only 48 percent of the total number of active pixels can be used for effective imaging. This is the trade-off that we need to face when we reduce the size of the camera by removing the lenses.

5.3 Actual Field of View and Spatial Resolution Calculations

In this section, we will be discussing about how we can calculate the field of view of the camera from the experiment and perform spatial resolution calculation of the camera we can expect when we use the lensless camera in practice. The spatial resolution can be calculated using equation 2.1. The calculated spatial resolutions based on mask-sensor distance and the height of the satellite from the earth is shown in Figure 5.18. It can be seen from the graph that as we increase the mask-sensor distance, we can attain better spatial resolutions that can represent the ground data better. the best spatial resolution is obtained when the mask-sensor distance is the greatest(i.e 10 mm) and when the satellite is as close to the earth(350 kilometres). The spatial resolution at a height of 350 kilometres would be 7.7 metres per pixel for a mask-sensor distance of 10 mm. A mask-sensor distance of 5 mm would provide 15.4 meters per pixel at the same height. The figure 5.18 is calculated for a pixel size of $2.2 \times 2.2 \mu m$. The other sensor OV5642 can offer a better spatial resolution as it's pixel size is smaller $1.4 \times 1.4 \mu m$. The spatial resolution values will be scaled by the ratio of the pixel sizes.

The field of view of the camera can be calculated using the acceptance angle calculated from integrating the experimental and simulation results. The sensor OV2640 has an experimentally verified acceptance angle of 21.7 degrees. The field of view can be calculated using equation 2.2. The obtained calculation results are shown in Figure 5.19. It can be seen that maximum field of view can be obtained when the satellite is at a height of 800 kilometres

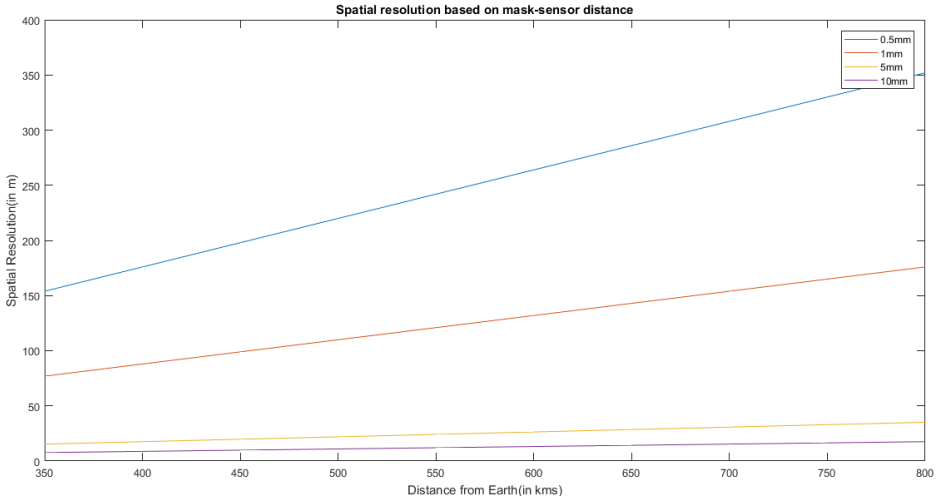


Figure 5.18: Relation between Spatial resolution and mask-sensor distance for sensor OV2640

from the surface of the earth. However, the spatial resolution as mentioned earlier will be lower at higher altitudes. A maximum field of view of 636 kilometres can be obtained. The sensor OV5642 has a chief ray angle of 23.6 degrees as mentioned in it's data sheet[38]. However, this claim has not been experimentally verified.

5.4 Spatial Light Modulators

A spatial light modulator (SLM) is an object that imposes some form of spatially varying modulation on a beam of light[11].SLMs can be controlled by computer controlled software and it would be possible to generate patterns on the SLM that could modulate phase or the intensity of the beam or both simultaneously. An advantage of using an SLM over designing a lithographic mask is that it would be possible to test out different designs of masks quickly in order to find out an optimal mask configuration that would be suitable to our setup. In our design of the lensless imager, it must be possible to block and allow light in a certain binary pattern. A transmissive SLM would suit the purpose of simulating different mask patterns. For this purpose, a holoeye LC2012 SLM was used

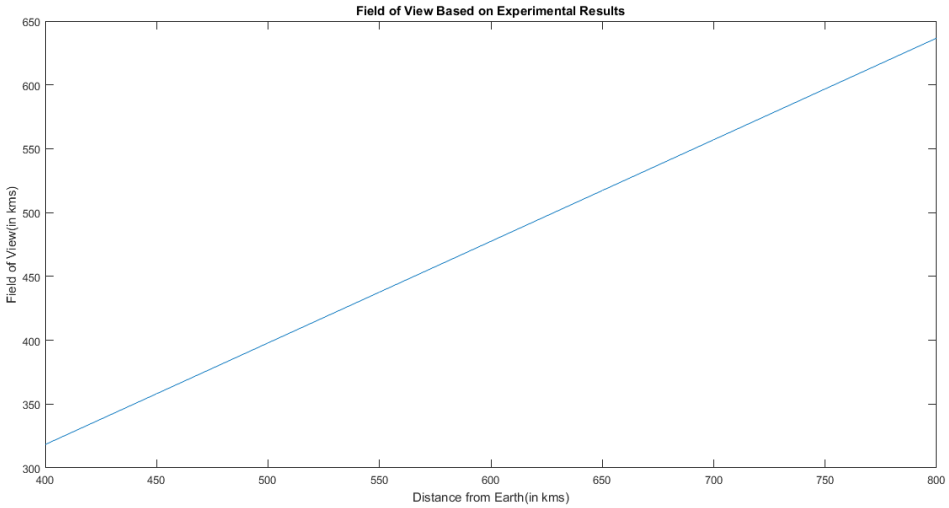


Figure 5.19: Field of View Calculation based on Experimental Results

Figure 5.20: HoloEye LC2012 SLM

Chapter 6

Experimentations with SLM

6.1 Initial Experiments to image mask

The experimental setup consists of a coherent laser beam, two mirrors, a spatial filter, a collimator lens, LC2012 SLM and finally an OV2640 CMOS sensor. The experimetal setup is shown in Figure 6.1. The collimator lens and the spatial filter make the beam parallel and remove unwanted random noise in the laser beam.

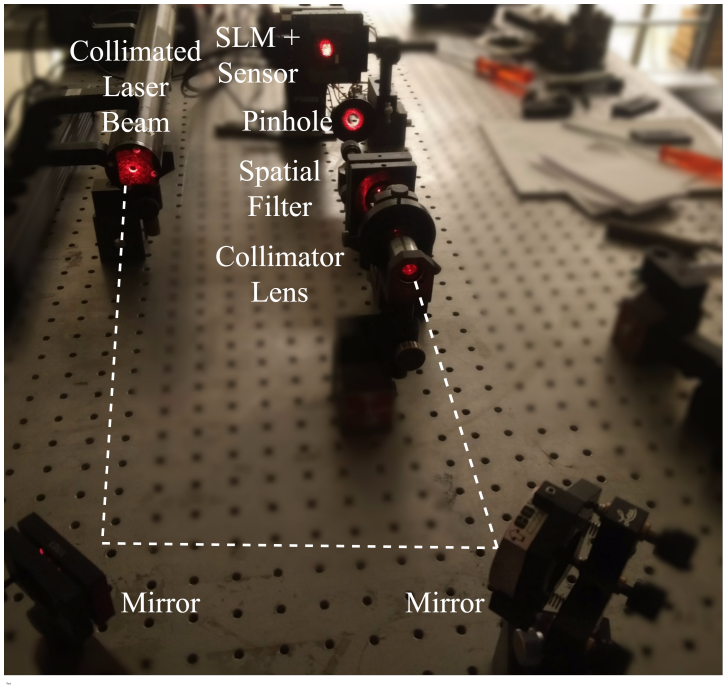


Figure 6.1: Experimental Setup to image mask

I started off the experiment with the setup mentioned in the previous sec-

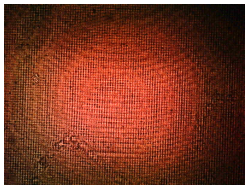
tion. However, when trying to take an image of the mask with the CMOS sensor, I faced multiple challenges along the way. The first challenge came in the form of the memory limitations of the camera module. The OV2640 camera module has only 384KB of memory which can store compressed JPEG images when imaging with lens. The memory limitation of the camera does not come into effect when imaging with lens as JPEG encoding is naturally designed for reducing size of real life artifacts and object scenes with a very high compression ratio. However, when using the same camera module, for lensless imaging, the size of the image easily exceeds the size of the FIFO buffer and the host software provided by the vendor is unable to read out the images. In order to overcome this challenge, I thought of increasing the extent of compression(with a trade-off of loss in quality) to image of the mask. This can be done by changing the `QSC` register(quantization scale factor) [] of the CMOS sensor. The default value of the quantization scale factor is `0C`(hex value). This value was increased to `2F` by setting the registers using I2C as mentioned for the previous acceptance cone experiments. The size of the image was successfully reduced and the host software was able to read out images. After this the sensor was placed in front of the SLM and the experiments were continued. I started experimenting with the default masks provided by the SLM vendor before continuing onto the custom mask designed using simulations. The mask that was used was a binary axicon mask and binary mask which divides the SLM into two different halves.The figure of the mask is shown in Figure 6.2. The white portion of the horizontal mask is not visible.

After overcoming this challenge I faced the second challenge wherein, the mask was not visible and the central part of the sensor had saturated(whited out). This can be seen in Figure 6.3. It can be seen from Figure 6.3 that the binary axicon mask is not at all visible due to the saturation of the CMOS sensor in the central region. The horizontal mask is imaged as a vertical mask by the CMOS sensor as the SLM is oriented vertically against the CMOS sensor. Even in the horizontal mask, the central portion has saturated.

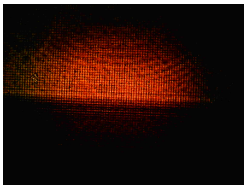
(a) Binary Axicon Mask (b) Horizontal dividable mask

Figure 6.2: Binary Masks used for Test

In order to solve these problems, multiple changes were made to the way the experiment was conducted. To avoid saturation, the exposure time of the CMOS sensor was reduced to the lease possible value(70 μseconds).



(a) Binary Axicon Mask imaged by CMOS



(b) Horizontal dividable mask imaged by CMOS

Figure 6.3: Binary Masks used for Test

Since, this did not affect the image of the mask formed on the sensor, it was decided to change the brightness and contrast levels of the SLM itself. The transmissive SLM acts as a screen and thus it has it's own brightness and contrast levels that affect the amount of light reaching the sensor. So, I decided to study the effect of the brightness and contrast on the image sensor. Also, two n.d filters were added to the setup that cut the intensity of light reaching the sensor by 75 percent. The SLM was placed approximately at a distance of 1 cm(10 mm) from the mask for these experiments.

6.2 Effect of SLM brightness and contrast on CMOS sensor

The first set of experiments was conducted to see the effect of brightness and contrast of the SLM on the image formed by the CMOS sensor. For this I thought of illuminating the CMOS sensor using the laser beam and then study the output image from the CMOS sensor for different gray levels on the CMOS sensor. Three gray levels(0,128,255) were chosen and the brightness/contrast levels were varied to see which brightness/contrast levels best represent the amount of light propotional to the grayness levels of the SLM. The brightness/contrast levels could be varied from 0 to 64 on the SLM. All the readings were normalized with respect to 255 to see how the intensity drops with respect to grayness levels. It can be seen from figures 6.4,6.5,6.6 that there is a certain level of drop in intensity signal irrespective of the brightness levels. However, it is only in some gray levels that the drop is significant and noticeable(¿25 percent reduction in intensity). The attenuation factors(ratio of reduction in intensity with respect to gray level 0) is plotted in tables 6.1,6.2 that brightness levels and contrast levels close to 60 provide maximum attenuation in signals for graylevels 128(around 28 percent reduction) and gray levels 255(around 88 percent reduction) gray levels. It was decided to set the brightness and contrast values to 63 and study how the grayness level of the SLM affects the amount of light passing through the SLM.

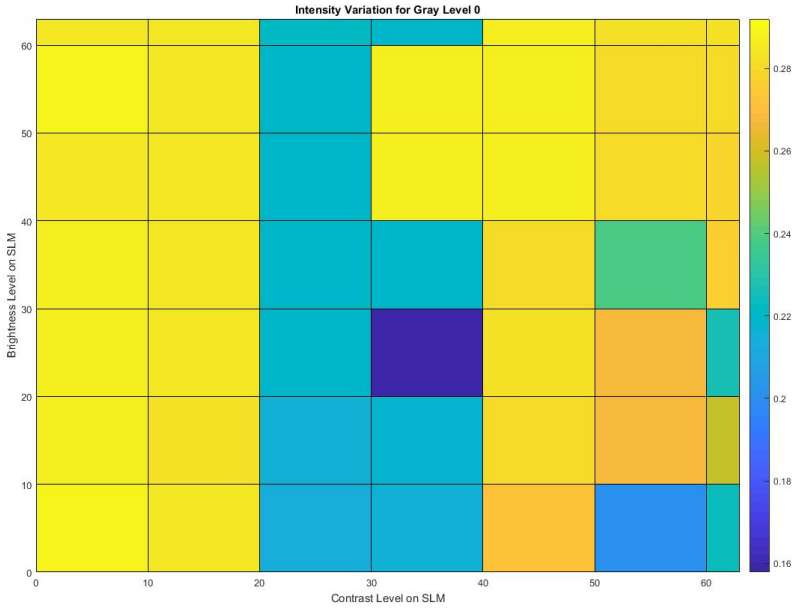


Figure 6.4: Effect of brightness and contrast on intensity for grayscale 0

C / B	0	10	20	30	40	50	60	63
0	0.9462	0.9669	1.2788	1.2455	0.9471	1.1301	0.8830	0.8875
10	0.9537	0.9792	1.2639	1.2326	0.9206	0.8663	0.7717	0.6853
20	0.9618	0.9717	1.2542	1.7043	0.9146	0.8726	0.8941	0.7327
30	0.9622	0.9712	1.2600	1.2353	0.9266	0.9862	0.7416	0.8852
40	0.9731	0.9689	1.2625	0.9466	0.9127	0.8347	0.7390	0.7205
50	0.9593	0.9696	1.2597	0.9499	0.9145	0.8351	0.74525	0.7149
60	0.9745	0.9634	1.2467	1.2331	0.9172	0.8288	0.7433	0.7159
63	0.9745	1.2669	1.233	1.2286	0.9121	0.8318	0.7394	0.7221

Table 6.1: Attenuation Factors of gray level 128 with respect to gray level 0 for different brightness and contrast levels

6.3 Effect of gray levels on CMOS sensor

From the experimental results mentioned in the previous section, it can be seen that the maximum brightness and contrast levels of the SLM provide

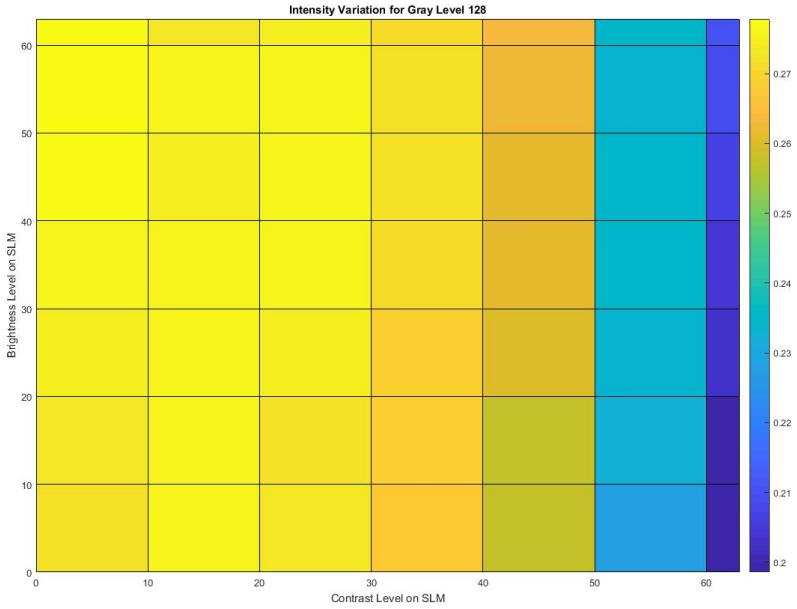


Figure 6.5: Effect of brightness and contrast on intensity for grayscale 128

B \ C	0	10	20	30	40	50	60	63
0	0.9834	0.9941	1.3247	1.2975	0.9602	1.0882	0.81991	0.8124
10	0.9971	1.0161	1.2988	1.2161	0.8096	0.5931	0.4336	0.3780
20	1.011	0.9831	1.2162	1.5084	0.5896	0.3460	0.2686	0.2113
30	0.9966	0.9647	1.1133	0.8718	0.3898	0.2200	0.16253	0.1946
40	0.9841	0.9138	0.9454	0.4592	0.2239	0.1554	0.1627	0.14317
50	0.9350	0.7867	0.6462	0.2665	0.1593	0.1420	0.1296	0.1356
60	0.8626	0.6265	0.3937	0.2237	0.1418	0.1345	0.1327	0.1291
63	0.8188	0.7389	0.3334	0.2104	0.1375	0.1323	0.1385	0.1293

Table 6.2: Attenuation Factors of gray level 255 with respect to gray level 0 for different brightness and contrast levels

the maximum attenuation in signals. So, these values were chosen and the grayness levels of the SLM was varied from 0 to 255 and study how the grayness levels affect the amount of light passing through the SLM. The entire SLM screen is set to this grayness level so that the entire sensor is modulated with the same intensity of light. The mean of the entire output image of the CMOS sensor is taken for plotting the signal and it is normalized with respect to the maximum reading and the graph is plotted. This is shown in figure 6.7. As it can be seen in the graph, the lowest point corresponds to

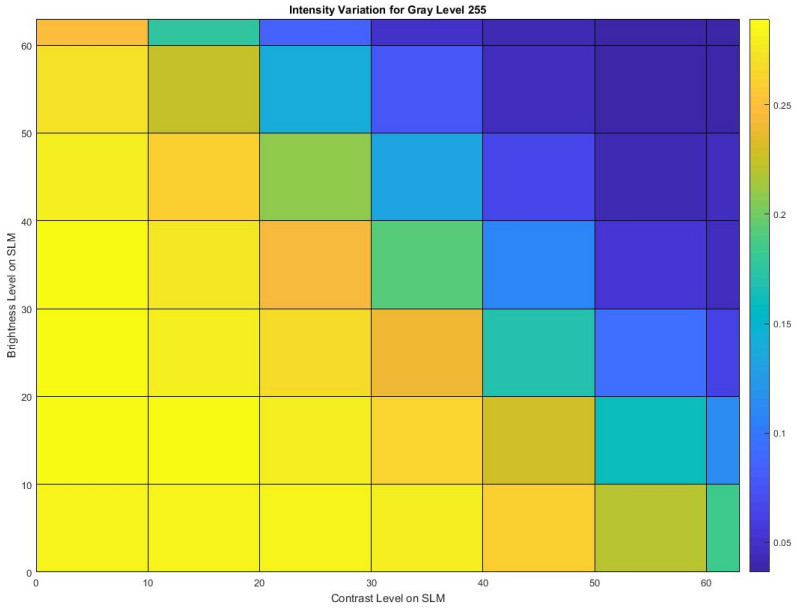


Figure 6.6: Effect of brightness and contrast on intensity for grayscale 255

an intensity reduction of approximately 88 percent. One important observation is that it is not possible to create a completely binary mask which blocks light as the maximum intensity reduction is only 88 percent. This will have an effect on the image reconstruction as it is assumed that the mask is completely binary.

6.4 Imaging of Separable Mask using CMOS sensor

In the previous section, we discussed how the SLM brightness and contrast affect the mask visibility on the CMOS sensor. In this section, we keep the mask brightness and contrast at maximum possible setting and image the mask. A random separable mask is generated for the resolution of the SLM. As mentioned in the previos sections, a random mask is one in which the row and column of the matrix can be expressed in the form:

$$M = M_x * {M_y}^T$$

Like in the simulations, a linearly spaced random vector is generated for a specific length(`Nxm0`) and interpolated to the SLM screen length(1024 X 768). Two vectors are generated(keeping `Nxm0` and `Nym0` constant) and the outer product of the vectors is taken to produce a mask that is same as the resolution of the SLM. In order to make sure that the mask preserves the

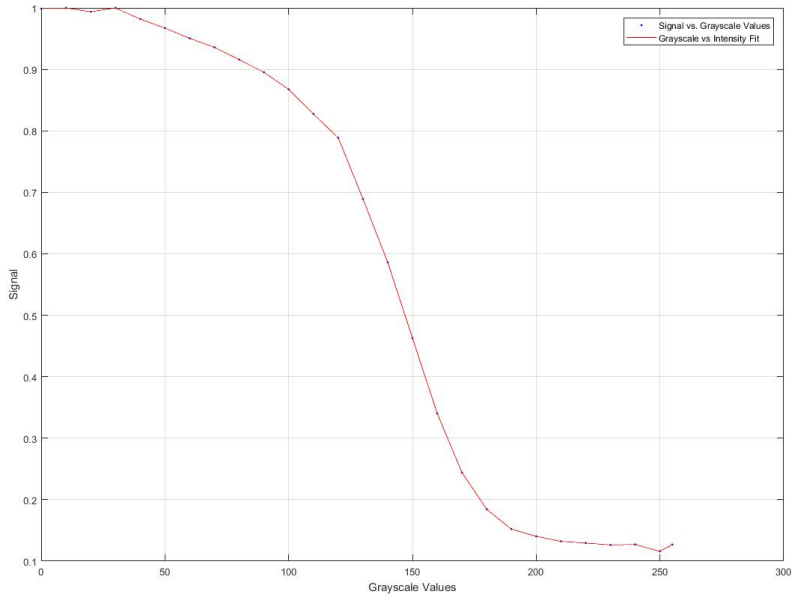


Figure 6.7: Grayscale vs intensity of light received by the CMOS sensor.

property of separability, we need to perform singular value decomposition on the output image from the CMOS sensor.

Since the SLM is larger that the CMOS sensor, the sensor only images a particular portion of mask and not the complete mask itself. The mask can decomposed into a singular matrix value no matter which part of the mask is imaged by the sensor. This was also verified in the MATLAB simulations. An example of doubly toeplitz mask generated is shown in figure 6.8. The region marked in red will also exhibit only one singular value on performing SVD. SVD forms the basis of the mask imaging experiment. We image the mask using OV2640 and see whether the mask still exhibits the same property when imaged by the CMOS sensor. The image of the doubly toeplitz mask imaged by OV2640 is shown in figure 6.9. As can be seen from the figure 6.9, the OV2640 produces a very poor image of the mask. This can be attributed to the poor quality of compression. The quantization factor increased during the previous experiments in-order to reduce the size of the image has reduced the quality of the output image. The change in the default quantization matrix and the modified quantization matrix that has been used for compression is shown below. The default quantization matrix has been modified so that the size of the image is within 384 KB. Without modifying the quantization scale factor, it would not be possible to obtain any image using a mask-sensor setup using OV2640. From the modified Q-Table, we can see that the higher frequency components will be eliminated

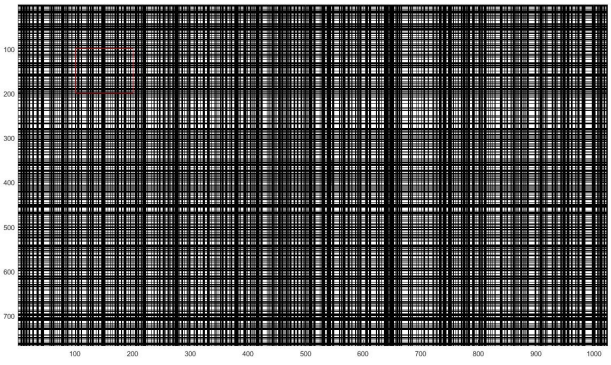


Figure 6.8: Doubly Toeplitz mask

and the lower frequency components will be retained. The Q-tables were obtained using the `djpeg` program[24] on the output image sensors running on Linux.

$$Q - Default = \begin{bmatrix} 12 & 8 & 8 & 12 & 18 & 30 & 38 & 46 \\ 9 & 9 & 11 & 14 & 20 & 44 & 45 & 41 \\ 11 & 10 & 12 & 18 & 30 & 43 & 52 & 42 \\ 11 & 13 & 17 & 22 & 38 & 65 & 60 & 47 \\ 14 & 17 & 28 & 42 & 51 & 82 & 77 & 58 \\ 18 & 26 & 41 & 48 & 61 & 78 & 85 & 69 \\ 37 & 48 & 59 & 65 & 77 & 91 & 90 & 76 \\ 54 & 69 & 71 & 74 & 84 & 75 & 77 & 74 \end{bmatrix}$$

$$Q - Modified = \begin{bmatrix} 47 & 32 & 29 & 47 & 71 & 118 & 150 & 179 \\ 35 & 35 & 41 & 56 & 76 & 170 & 176 & 162 \\ 41 & 38 & 47 & 71 & 118 & 167 & 203 & 165 \\ 41 & 50 & 65 & 85 & 150 & 255 & 235 & 182 \\ 53 & 65 & 109 & 165 & 200 & 255 & 255 & 226 \\ 71 & 103 & 162 & 188 & 238 & 255 & 255 & 255 \\ 144 & 188 & 229 & 255 & 255 & 255 & 255 & 255 \\ 212 & 255 & 255 & 255 & 255 & 255 & 255 & 255 \end{bmatrix}$$

Based on the mask image, we can be sure that it would be impossible to obtain proper reconstructions. So, it was decided to try whether other CMOS sensors would produce better quality pictures that would enable proper reconstruction. Based on the trade-off chosen previously, OV5642 sensor was also bought in-case of any failure. Since, OV2640 and OV5642 follow the same architecture and use the same open-source libraries, it was decided to check the image quality on OV5642. The OV5642 has an advantage that it has a bigger active sensor array(2592*1944) and comes with a

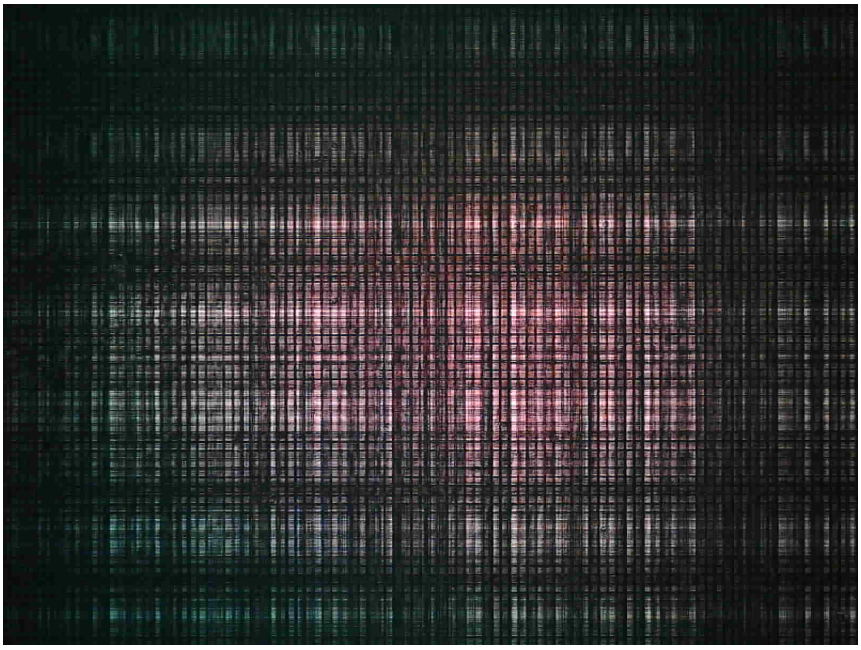


Figure 6.9: Doubly Toeplitz mask imaged by OV2640 after modification of Q-Table

larger FIFO buffer(8MB). In order to position these sensor on the stage, a script was written that would calculate the singular values in real time. The tilt of the sensor was adjusted such that the ratio of the first component to the second component is maximum. It can be seen from figure 6.10 that OV5642 produces a better image of the mask. In order to compare performance of OV5642, another web camera(Microsoft Lifecam HD 3000) was dis-assembled and the mask was imaged(See figure 6.11). The difference in redness is due to the saturation level adjustment on LifeCam. A lot of methods were tried to increase the ratio of the first to the second singular value. Firstly, I tried to capture the image without any mask and capture the noise. I subtracted the noise from the mask image. However, this did not lead to any change in the singular component values. I then tried median filtering the output to reduce the gaussian noise effects and it increased the singular value ratio with all the CMOS sensors. This is shown in table 6.3. Table 6.3 was plotted from the average of images in a particular data set using the same masks. The red channel data and the grayscale image data from the mask image are used for SVD. It can been from the table that OV2640 provides a better singular value ratio than OV5642. However, the highest singular component does not at all represent the mask that is programmed onto the SLM. So, OV2640 does not perform equivalent to OV5642 despite the higher ratio of singular component.

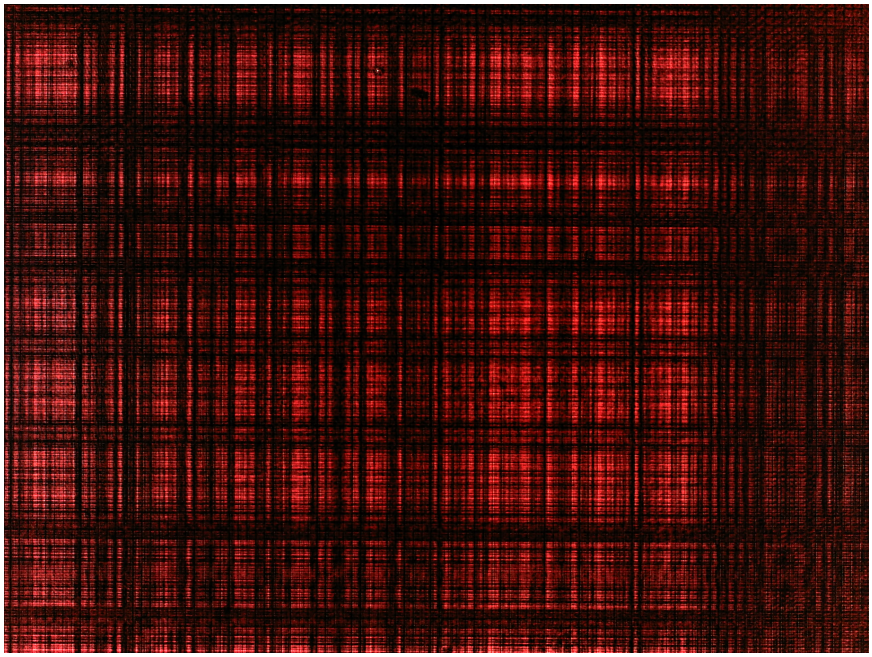


Figure 6.10: Doubly Toeplitz mask imaged by OV5642

Technique / Sensor	SVD(Red)	SVD(Red) with filtering	SVD(Gray)	SVD(Gray) with filtering
OV2640	7.833	11.16	8.05	10.49
OV5642	8.74	10.84	7.04	10.78
LifeCam	16.82	21.57	20.89	26.75

Table 6.3: Performance of Different CMOS sensors(The values indicate the ratio of the first SVD to the second SVD value)

The same results as in the simulation(only one singular value) could not be replicated because the mask that is produced by the SLM is not completely binary as observed in the previous experiment with SLM. If you observe the images from the sensor closely you can see that the black regions allow some portion of the light to pass through. This results in the image matrix becoming non-binary which in-turn results in secondary SVD components. One way to solve this would be to use a threshold and make the mask image binary but this would result in loss of diffraction and object data. Apart from this factor, another reason why other SVD components are observed is due to the presence of dead-pixel regions in the output data. The values of these pixels over-ride the values of the mask. These effects do not

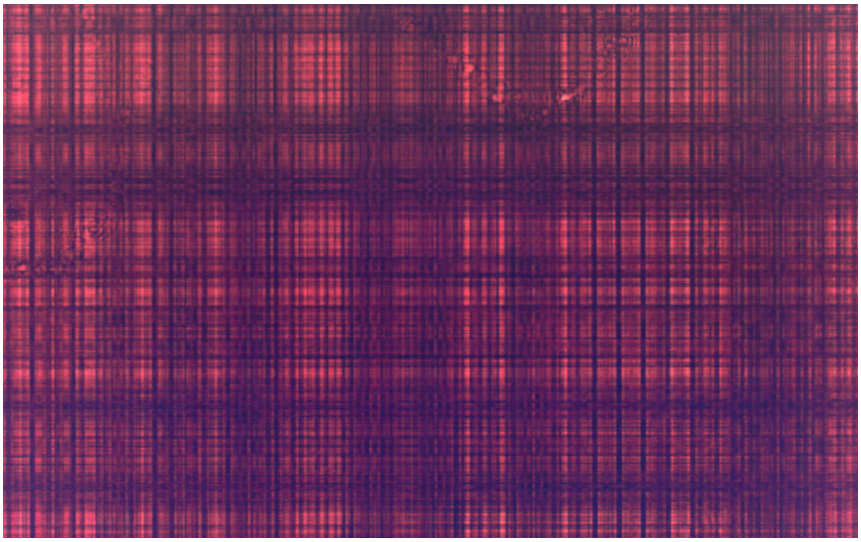


Figure 6.11: Doubly Toeplitz mask imaged by Microsoft Lifecam HD 3000

play a major role in lens based imaging but form a very important role in lensless imaging. Exposure plays a very important role in decomposability of the matrix. I first thought of increasing the exposure in-order to make the difference between the black and white more prominent (which in-turn increases separability). This idea seems to have worked out. The ratio of singular value to the exposure time is shown in Figure 6.12. The mask becomes separable as the exposure time is increased. However, this comes with a disadvantage. LifeCam has a very limited controllable range of exposure time and the sensor starts quickly saturating at around 2ms. When the sensor starts saturating, it leads to loss of data which in turn will have an effect on reconstruction. The best decomposition that I could get was using the LifeCam HD-3000 with exposure adjusted(without saturation) based on figure 6.12. The decomposition of mask into different components is shown in figure 6.13.

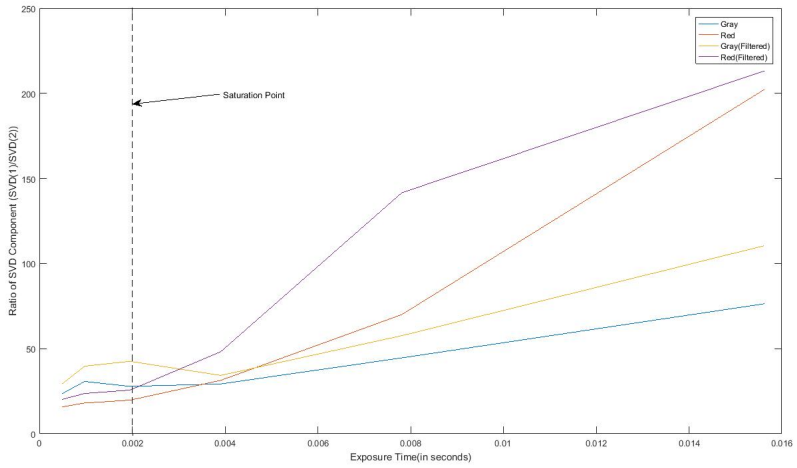


Figure 6.12: Impact of Exposure Time on Singular Values

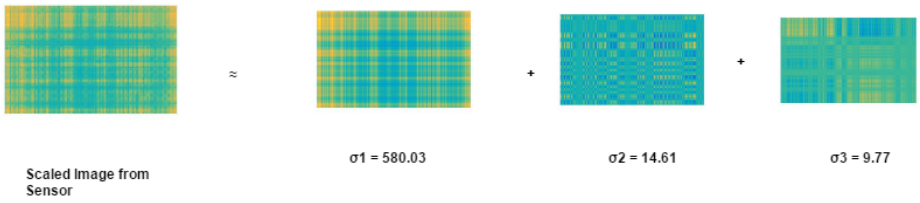


Figure 6.13: Decomposition of best data set(Ratio of first to second component = 39.70)

Chapter 7

Estimation of System Matrices

This chapter will describe the strategy for obtaining the system matrices(M_x and M_y) of the mask produced by the SLM in front of the sensor. The reader may have a question that why do we need to estimate the M_x and M_y when we already know the mask pattern that we project onto the SLM screen. The problem lies with the uncertainity that we have when we use a mask pattern using the SLM instead of fabricating the mask. The simulations assume that there is one-one mapping of the mask-sensor pixels and that they have the same pixel pitch. However, from the manual of the SLM[31] we can see that the SLM has a pixel pitch of 36 μm and the sensor OV5642 has a pixel pitch of 1.4 μm. The sensor OV2640 cannot be used as of now as we saw in the previous chapter. Due to the differences in the pixel pitch, the effects of compression(which have not been taken into account in the simulations) and ambient noise effects in real world applications, we need to manually estimate the system matrices by projecting known patterns to obtain perfect reconstructions. The experimental setup consists of an LCD screen placed at 20 cm from the SLM-sensor setup. The LCD will be used to project objects and patterns in front of the sensor. The experimental setup is shown in Figure 7.1. Initially, I tried projecting a complete blank white pattern on the LCD but the sensor recorded a blank image. I expected that the sensor would give me an image that would represent the mask itself. I tried to move the mask-sensor closer to the LCD as I thought that it might be due to not enough light reaching the sensor, but it also made no difference.The reasoning behind this is due to vignetting of light by the pattern. I tried projecting an horizontal dividable pattern on the LCD to see if there is any variation in intensity across the sensor, but there was no difference in intensity across the output image. Then, I decided to project a circular pattern, to see if there is any difference in the readings from the CMOS sensor. I projected two patterns, a white circle in a black background

and a black circle in a white. On seeing the outputs through the naked eye, I found that there was not much difference. To my surprise, on subtracting the two output images, there were slight patterns seen on the difference image. So, I decided to develop a strategy that would enable me to estimate the system matrices.

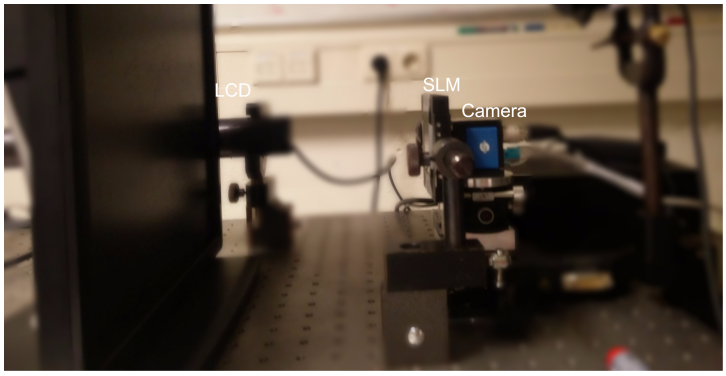


Figure 7.1: Setup for Intial Experiments

7.1 Strategy

The following precautions are taken to make sure that we can accurately estimate the system matrices.

- Even thought the sensor is measuring only one portion of the LCD, during the initial experiments, it was observed that the entire LCD is contributing to the image on the sensor. There was a noticeable variation in intensity when I tried to block the light from the LCD portion not facing the sensor. So, I decided to block the light from the surrounding pattern using black cardboards.

- I use the same idea from [13] to determine the system matrices. If the scene in front of the mask-sensor arrangement is separable, then the sensor output must also be separable. If the scene is of the form ab^T, then the image formed on the sensor will be of the form:

$$I = (M_x a)(M_y b)^T \tag{7.1}$$

If the scene can be represented in the form of $a1^T$, then the scene formed on the sensor will be,

$$Y = (M_x a)(M_y)^T \tag{7.2}$$

On performing singular value decomposition on the scene and approximating the rank-1 matrix(represented in the form $Y_k = u_k v^T$) from

the output image we must be able to estimate the left system matrix using the equation:

$$M_x = u_k a^{-1} \tag{7.3}$$

- For the right system matrix, we can use patterns in the form of $1a^T$, and estimate M_y. In the coming sections, we would look at what kind of pattern can be projected on the screen to accurately estimate the right and left system matrices.

7.1.1 Identity Basis

In order to estimate the system matrices, we can use the idea described previously, but we need to decide what patterns need to be used and hoe we can mathematically simulate and verify our idea. We use a invertible base matrix to generate different patterns that can be projected on the LCD screen. The basis matrix pattern can any pattern that would be optically feasible and invertible.

$$B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{n1} & b_{n2} & \dots & b_{nn} \end{bmatrix}_{n \times n} \tag{7.4}$$

For estimating the left system matrix we need to extract the columns from the basis matrix that we use and multiply it with a row vector of 1s as shown by equation

$$1_{row} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}_{1 \times n} \tag{7.5}$$

$$b_i = \begin{bmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{in} \end{bmatrix}_{n \times 1} \tag{7.6}$$

$$pattern_i = \begin{bmatrix} b_{i1} \\ b_{i2} \\ \vdots \\ b_{in} \end{bmatrix}_{n \times 1} * \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}_{1 \times n} \tag{7.7}$$

The image produced by $pattern_i$ on the sensor can be decomposed into $u_i v^T$ using the rank-1 estimation obtained from singular value decomposition(SVD). We need to accumulate the sensor output from each b_i multiplied by the row-one vector and find the corresponding u_i. These u_i need to be put together to form an $n * n$ matrix. The accumulated u_i values correspond to the left system matrix combined with the basis matrix B. We can then invert it to obtain the actual system matrix as shown in equation 7.8.

$$M_x = \begin{bmatrix} u_1 & \dots & u_i & \dots & u_n \end{bmatrix}_{n \times n} * B^{-1} \tag{7.8}$$

Similarly, two obtain the right system matrix using patterns obtained from multiplying the rows from the basis matrix and columns of 1s as illustrated by equation 7.9. The accumulated v_i obtained from singular value decomposition can be inverted using the inverse of the basis matrix as done for the left system matrix. The procedure for obtaining the system matrices is illustrated in Figure 7.2.

$$pattern_j = \begin{bmatrix} b_{j1} & b_{j2} & \dots & b_{jn} \end{bmatrix}_{1\times n} * \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}_{n\times 1} \tag{7.9}$$

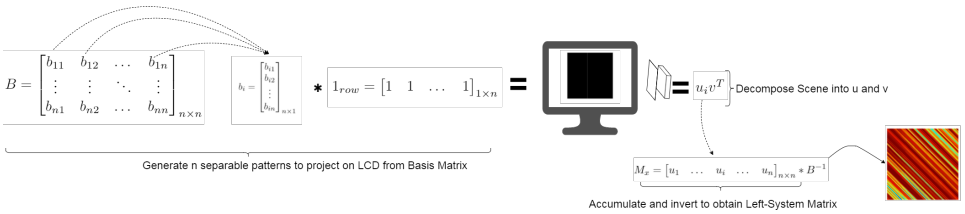


Figure 7.2: Procedure for obtaining the left system matrix

The simplest basis matrix that could be inverted is the identity matrix. The patterns produced by the identity matrix will be a shifting row of ones(white) in a black background for the left system matrix and a shifting column of one in a black background for the right system matrix. This is shown in Figure 7.3.

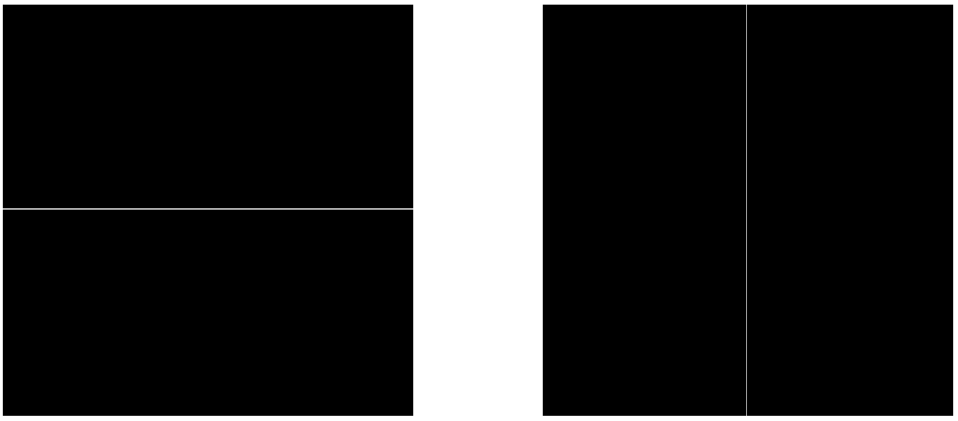


Figure 7.3: Pattern on LCD that would be generated with left and right system matrix

This entire procedure was simulated on MATLAB like in the previous section and reconstructions obtained using the procedure is compared. The

reconstructions obtained by simulating the LCD patterns and obtaining the system matrix is shown in Figure 7.4. It can be seen from the figure that the reconstruction obtained using the procedure mentioned leads to a poorer reconstruction than the one which we obtained using the known diffracted mask pattern. The system matrix obtained by using the procedure explained

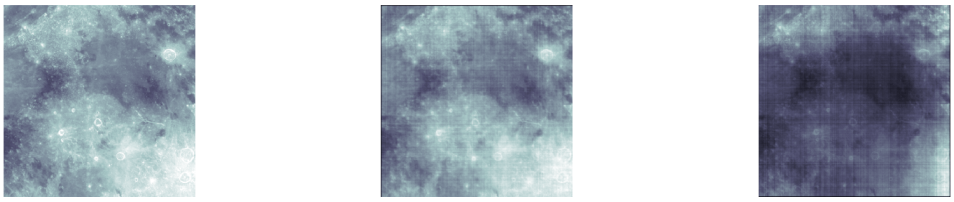


Figure 7.4: This figure shows the differences in reconstruction. The figure on the left most corner is the original object, the middle image represents the inversion obtained using the known diffracted mask pattern, and the right image indicates the reconstruction obtained using simulated LCD patterns and obtaining system matrix. It can bee seen that there is an extreme loss of detail when we try to manually obtain the system matrix using the procedure stated.

above does not provide accurate reconstructions. This problem was traced to the accuracy of the `svd` function in MATLAB which is used for computing the left and the right vectors when analyzing the scene separability. One more issue with using the identity matrix is that this basis matrix generates a very less amount of light. Projecting these patterns on the LCD did not provide any noticeable change on the sensor in the experimental setup. So, it was decided to use some other basis matrix for estimating the system matrices. The inaccuracy in estimating the left and right vectors is resolved mathematically as explained in the next section.

7.1.2 Hadamard Basis

In this section we will first discuss an alternative way of obtaining the left and the right system vectors instead of using `svd` function in MATLAB as it does not provide accurate estimation of a separable scene. Consider that a scene of the form $a1^T$ is imaged by the sensor with the mask. The image on the sensor can be written in the form given by equation 7.2. It can be re-written as shown in equation 7.10.

$$Y(x, y) = \phi_1(x)\phi_2(y) \tag{7.10}$$

The term $\phi_1(X)$ and $\phi_2(Y)$ represent $(M_x a)$ and (M_y) of equation 7.2respectively. The integral of equation 7.10 can be represented by equation 7.11. When we project a scene of the form $a1^T$, equation 7.11 can be re-written as equation 7.12 as a constant 1 vector is the right vector that is

used to generate the pattern.

$$\int Y(x,y) \propto \phi_1(x) \int \phi_2(y)dy \tag{7.11}$$

$$\int Y(x,y) \propto \phi_1(x)k \tag{7.12}$$

When we sum over the sensor output image in the y-direction while projecting patterns of the form $a1^T$, we will be only left with the left system matrix multiplied by a constant. Similarly, when we sum over the sensor output image in the x-direction while projecting patterns of the form $1a^T$, we will be only left with the right system matrix multiplied by a constant. This also reduces the computational load instead instead of using MATLAB function `svd` which is highly computationally intensive. Since, we solved the problem of estimating the left and right system matrices by using a simple summation, let us now focus on using a better basis matrix for estimating the system matrices.

The identity matrix provides a very less amount of light since it produces only one row of emitting light. So, I decided to test some striped separable patterns and see if we are able to receive any response on the sensor. It was observed that striped patterns are able to produce some siginificant sensor response on estimating the left and right system matrix using SVD. So, I decided to look for matrices that can produce striped patterns on the screen.

Figure 7.5: This figure indicates the response of the sensor. The right most image is put on the LCD screen. The middle portion indicates the output image from the sensor. The third indicates the rank-1 estimate of the scene obtained using SVD

In order to accurately estimate the system matrices, we need to project multiple striped patterns. We want a matrix that has binary values(1s and 0s) so that we can translate them into color and then work on reconstruction of it. One such pattern is a hadamard pattern which is widely used in compressed image sensing[10][13]. This basis matrix is also used in previous studies is the hadamard matrix H whose inverse is of the form given by equation 7.13. The size of the hadamard matrix n is always in the powers of 2.

$$H^{-1} = \frac{1}{n} H^T \tag{7.13}$$

However, one disadvantage of hadamard matrix is that it has negative one values which is optically not feasible. However, we can follow some mathematical tricks to make it optically feasible. Consider a 2×2 hadamard matrix shown in equation 7.14. Let us take the keep the matrix and also take the negative of the matrix.

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{7.14}$$

$$H_2^+ = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \qquad H_2^- = \begin{bmatrix} -1 & -1 \\ -1 & 1 \end{bmatrix} \tag{7.15}$$

In order to make the mask optically feasible, let us set the negative entries of equation 7.15 to zero.

$$H_2^+ = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \qquad H_2^- = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.16}$$

We can see from equation 7.16 that

$$H_2 = H_2^+ - H_2^- 1 \tag{7.17}$$

By projecting binary patterns of the form given by equation 7.16, we can make an optically realizable hadamard pattern. However, a disadvantage is that we need to project two kinds of patterns(one for H^+ and one for H^-) and subtract the sensor images to get the reading for the original hadamard pattern. An example of vertical and horizontal hadamard patterns is shown in Figure 7.6.

An example of reconstruction using the hadamrd patterns is shown in Figure 7.7. It can be seen from Figure 7.7 that estimating the system matrix manually provides a lot better results and preserves a lot more detail than using the known diffracted mask pattern.

We can see that both the basis matrices need measurements from $N \times N$ calibration patterns to reconstruct a $N \times N$ resolution image. We can generate $N \times N$ hadamard patterns and estimate the accurate system matrix that can provide extremely good reconstruction results as seen in the simulation. This scheme of estimating the is also extremely resistant to noise which was simulated by adding random noise equivalent to the signal on the LCD screen. However, it is extremely sensitive to rotation angles. The LCD must be exactly parallel to the sensor in-order for this scheme to work. In the actual this can be achieved by manually aligning the screen while computing the SVD components. the perfect alignment would give us the highest value of the singular component ratio. The process must be repeated as mentioned in the previous chapter where the sensor was aligned with the SLM using singular value decomposition ratios.

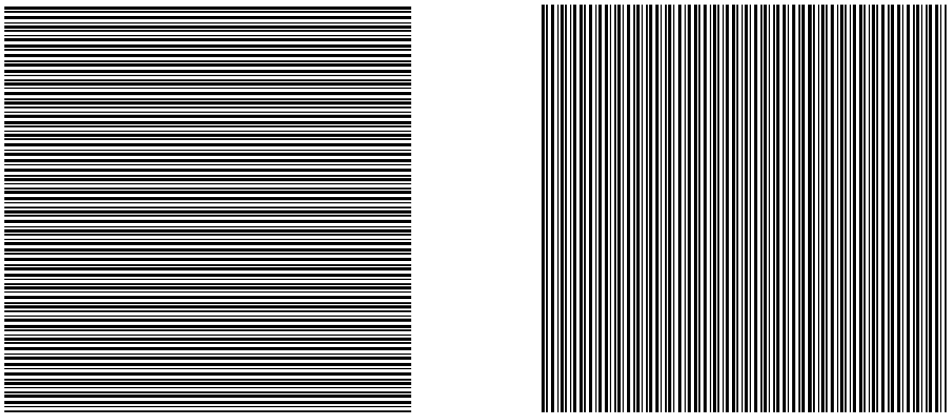


Figure 7.6: An example of vertical and horizontal hadamard pattern(H^+) that can be used for system matrix estimation. Note that the same procedure as mentioned previously holds. Only an additional pattern needs to be incorporated to make the hadamard pattern optically feasible

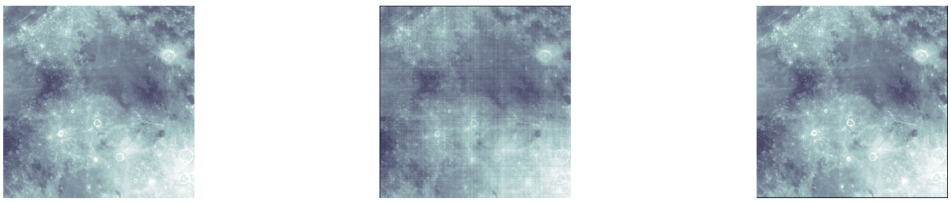


Figure 7.7: This figure indicates the reconstructions obtained using the hadamard matrix as the basis matrix and after summing to obtain the separable scene components. The left most image indicates the original object, the middle indicates the reconstruction obtained with known system matrix and the right indicates the reconstruction with the system matrices obtained by projecting hadamard patterns. It can be seen that the effects of noise are no longer visible and the reconstruction is perfect.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

TODO CONCLUSIONS

8.2 Future Work

TODO FUTURE WORK

Bibliography

[1] Chief ray angle. http://armstrongoptical.co.uk/wp-content/uploads/2013/12/Chief-Ray-Angle.pdf. Accessed: 2017-08-17.

[2] Delfi-c3 project. http://www.delfispace.nl/delfi-c3. Accessed: 2017-05-20.

[3] Full moon photographed from apollo 11 spacecraft. https://www.nasa.gov/mission_pages/apollo/40th/images/apollo_image_25.html. Accessed: 2017-03-29.

[4] How jpg works. https://medium.freecodecamp.org/how-jpg-works-a4dbd2316f35. Accessed: 2017-10-25.

[5] Image formation revisited. http://www.physicsclassroom.com/class/refrn/Lesson-5/Image-Formation-Revisited. Accessed: 2017-11-2.

[6] Imaging geometries. http://www.its.caltech.edu/~ee157/lecture_note/Imaging%20geometries.pdf. Accessed: 2017-05-20.

[7] imfindcircles. https://nl.mathworks.com/help/images/ref/imfindcircles.html. Accessed: 2017-08-17.

[8] Manual exposure for ov2640. http://www.arducam.com/manual-exposure-ov2640/. Accessed: 2017-06-28.

[9] Resolutions of remote sensing. http://www.edc.uri.edu/nrs/classes/NRS409/RS/Lectures/HowRemoteSensonWork.pdf. Accessed: 2017-05-20.

[10] A russian dolls ordering of the hadamard basis for compressive single-pixel imaging. https://www.nature.com/articles/s41598-017-03725-6. Accessed: 2017-11-09.

[11] Spatial light modulators. https://en.wikipedia.org/wiki/Spatial_light_modulator. Accessed: 2017-06-29.

[12] Understanding focal length. https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-focal-length.html#. Accessed: 2017-11-2.

[13] Flatcam: Thin, lensless cameras using coded aperture and computation. *IEEE Transactions on Computational Imaging*, PP(99):1–1, 2016.

[14] Arducam. Arducam lenses. http://www.arducam.com/lenses/.

[15] Arducam. Arduchip Registers. http://www.arducam.com/arduchip-registers/#more-119.

[16] Arducam. *2MP SPI Camera User Guide*, 2 2015. Rev 1.0.

[17] Arducam. *5MP SPI Camera User Guide*, 10 2016. Rev 1.0.

[18] Arducam. *SPI Camera Software Application Note*, 10 2016. Rev 2.0.

[19] V. Boominathan, J. K. Adams, M. S. Asif, B. W. Avants, J. T. Robinson, R. G. Baraniuk, A. C. Sankaranarayanan, and A. Veeraraghavan. Lensless imaging: A computational renaissance. *IEEE Signal Processing Magazine*, 33(5):23–35, Sept 2016.

[20] T. M. Cannon and E. E. Fenimore. Coded aperture imaging: Many holes make light work. *Optical Engineering*, 19(3):193283–193283–, 1980.
[21] Satellite Imaging Corporation. Characterization of satellite remote sensing systems. `https://www.satimagingcorp.com/services/resources/characterization-of-satellite-remote-sensing-systems/`. Accessed: 2017-05-20.
[22] Michael J. DeWeert and Brian P. Farm. Lensless coded-aperture imaging with separable doubly-toeplitz masks. *Optical Engineering*, 54(2):023102, 2015.
[23] K. Dezhgosha, A. K. Sylla, and E. Ngouyassa. Lossless and lossy image compression algorithms for on-board processing in spacecrafts. In *Aerospace and Electronics Conference, 1994. NAECON 1994., Proceedings of the IEEE 1994 National*, pages 416–423 vol.1, May 1994.
[24] die.net. djpeg(1) - Linux man page. `https://linux.die.net/man/1/djpeg`.
[25] Jie Ding, Mohammad Noshad, and Vahid Tarokh. Complementary lattice arrays for coded aperture imaging. *J. Opt. Soc. Am. A*, 33(5):863–881, May 2016.
[26] Irfan Essa. Computational Photography Udacity Course. `https://classroom.udacity.com/courses/ud955`.
[27] E. E. Fenimore and T. M. Cannon. Coded aperture imaging with uniformly redundant arrays. *Appl. Opt.*, 17(3):337–347, Feb 1978.
[28] A. El Gamal and H. Eltoukhy. Cmos image sensors. *IEEE Circuits and Devices Magazine*, 21(3):6–20, May 2005.
[29] M. Gostennik, D. Ganik, and I. Kramberger. Survey of image compression algorithms for esmo mission. In *2011 18th International Conference on Systems, Signals and Image Processing*, pages 1–4, June 2011.
[30] M H. Finger and T A. Prince. Hexagonal uniformly redundant arrays for coded-aperture imaging. 3, 09 1985.
[31] Holoeye. *LC2012 Device Operating Instructions/SLM Software Instructions*, 10 2016. V 3.1.
[32] Ir. A. Kamp. Instruments and their constituents. Space Instrumentation Engineering Lecture Notes, 2017.
[33] K. Khurshid, R. Mahmood, and Q. ul Islam. A survey of camera modules for cubesats - design of imaging payload of icube-1. In *2013 6th International Conference on Recent Advances in Space Technologies (RAST)*, pages 875–879, June 2013.
[34] C. Lambert-Nebout and G. Moury. A survey of on-board image compression for cnes space missions. In *Geoscience and Remote Sensing Symposium, 1999. IGARSS '99 Proceedings. IEEE 1999 International*, volume 4, pages 2032–2034 vol.4, 1999.
[35] Wolfram Mathworld. Singular value decomposition. `http://mathworld.wolfram.com/SingularValueDecomposition.html`.
[36] Omnivision Technologies. *Preliminary Data Sheet*, 2 2006. V 1.6.
[37] Omnivision Technologies. *Software Application Notes*, 1 2008. V 1.04.
[38] Omnivision Technologies. *Data Sheet Preliminary Specifications*, 1 2009. V 1.1.
[39] University of California San Diego. *Lecture Note 5, Multimedia Systems*, 2003.
[40] D.G. Voelz. *Computational Fourier Optics: A MATLAB Tutorial*. SPIE Press, 2011.
[41] Ho-Soon Yang, Yong-Geol Jo, Gyu-Ug Kim, and Yun-Woo Lee. Measurement of the chief ray angle of mobile phone camera. *Optical Review*, 18(5):403, Sep 2011.

Appendix

Fresnel Propagation Model

In this section, we will briefly discuss the fresnel propagation model used for obtaining the effects of diffraction. The model is based on [40] and the MATLAB model developed at Optica Group was used. When modelling the diffraction effect, we have a source plane and a observation plane. In our case, the source plane is the mask and the observation plane would be the sensor. This is shown in Figure 1. The diffraction effects lie in the fresnel

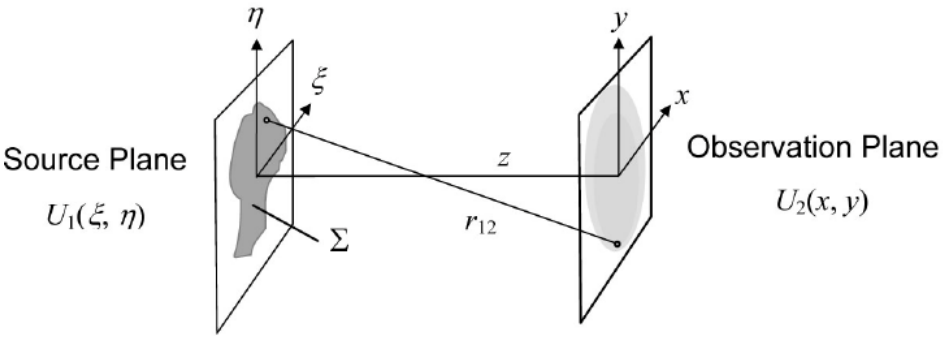


Figure 1: Source Plane and Observation Plane

region since the fresnel number wpuld be less than one. In this case, the diffraction effects can be given by the equation 1.

$$U_2(x,y) = \frac{e^{jkz}}{j\lambda z} \int\int U_1(\xi,\eta) exp(\frac{jk}{2z}[(x-\xi)^2 + (y-\eta)^2])d\xi d\eta \qquad (1)$$

This expression can also be written in the form of equation 2.

$$U_2(x,y) = F^{-1}(F(U_1(x,y)F(h(x,y))) \qquad (2)$$

where

$$h(x,y) = \frac{e^{jkz}}{j\lambda z} exp(\frac{jk}{2z}(x^2+y^2)) \qquad (3)$$

Appendix A Camera Dimensions

This section shows the dimension of the camera module as provided by the camera vendor.

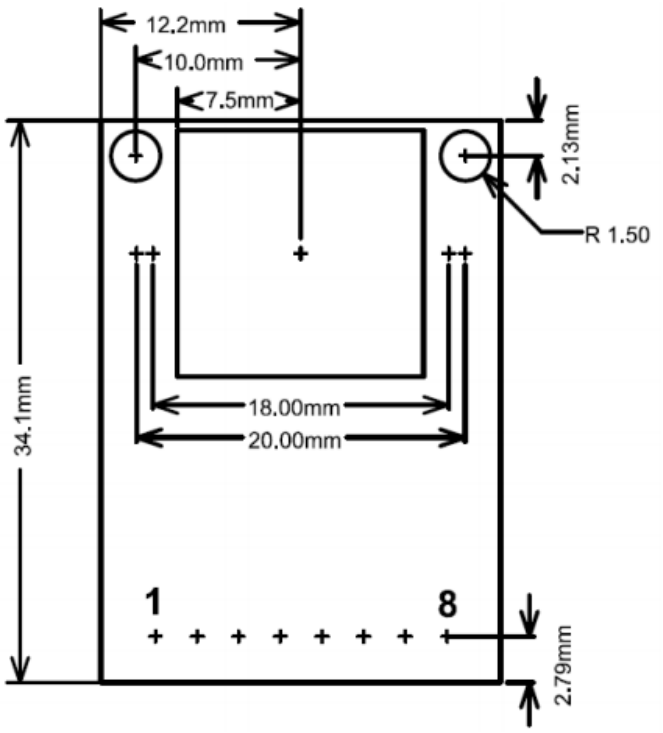


Figure 2: Dimensions of both OV5642 and OV2640 camera modules