Characterization of CMOS Image Sensor

Master of Science Thesis

For the degree of Master of Science in Microelectronics at Delft University of Technology

Utsav Jain
July 21,2016

Faculty of Electrical Engineering, Mathematics and Computer Science · Delft University of Technology

Delft University of Technology
Department of
Electrical Engineering

The undersigned hereby certify that they have read and recommend to the Faculty of Electrical Engineering, Mathematics and Computer Science for acceptance a thesis entitled
Characterization of CMOS Image Sensor

by

Utsav Jain

in partial fulfilment of the requirements for the degree of
Master of Science Microelectronics

Dated: July 21,2016

Supervisor(s):

prof. dr. ir. Albert J.P. Theuwissen

M.S.EE Dirk Uwaerts

Reader(s):

prof. dr. ir. Albert J.P. Theuwissen

prof. dr. ir. Andre Bossche

prof. ing. H.W.(Henk) van Zeijil

Acknowledgment

This master thesis project marks the end of my M.Sc. journey with lots of high and some lows. This experience was full of knowledge, uncertainty, life lessons and cherishing moments. I always wanted to pursue master degree and with the support of my family, friends, colleagues and professors, I accomplished my dream. I would like to thank all individuals who have accompanied me during this journey. Thank you all, who contributed in many ways to make this thesis possible and an unforgettable experience for me.

First and foremost, I would like to give my sincere thanks to my daily supervisor Mr. Dirk Uwaerts, System Development Manager at Caeleste for his supervision of my thesis. This work would not have been possible without guidance, patient supervising and encouragement. I would also like to give my sincere thanks to Prof. Albert J.P. Theuwissen, who introduced me to the world of image sensor. It was his unique way of teaching and enthusiasm that developed interest in me for image sensors.

Next, I would like to thanks Bart Dierickx and Patrick Henckes of Caeleste CVBA to give me the opportunity to do my master thesis project at their company. I would like to thanks all the test system team to help me to perform the measurements and helping me in building measurement setups and learning the software environment. A special thanks to Alexander Klekachev and Walter Verbruggen for their day to day help and support, we have spent quality time together with all the discussions about ways to improve measurement procedure and to upgrade hardware and software tools. I would also like to thank design team of Caeleste for their help in making me understand CMOS image sensor design and characteristics and for providing constant feedback on measurement results. I would like to express my gratitude to Bert Luyssaert who was always there for explaining any doubt regarding measurements and optical aspect of CMOS image sensor.

Next, I would like thanks my friends, Mansi Shah, Shreyans Jain and Nitant Shinde for making me laugh in moments of stress and motivating me to work harder on my thesis. A special thanks to my friend Sri Harsha Achante for late night studies, performing simulations and completing course assignments. These memories and the friendship is invaluable.

Lastly, I would like to thank my parents and my brother for all that they have done for me. Only with their support I could take the courage to study across the globe. Thank you for your continuous love, care and encouragements throughout my life.

Abstract

CMOS image sensors comprises of two process: designing and measurement/testing. CMOS image sensors are designed with certain characteristic performance and it is important to measure these characteristics accurately. CMOS image sensor convert light information into digital information which can be reproduced in form of an image. Conventional 4T pixel with pinned photodiode is a popular choice for designing image sensor; with certain modification in the pixel architecture better characteristic performance can be achieved with trade-offs.

Quantum efficiency, linearity, full-well capacity, conversion gain, noise, non-uniformity, dark current, modulation transfer function and image lag are main characterises of CMOS image sensor. Quantum efficiency defines the efficiency of the image sensor and ideally it should be 100 percent i.e. 1electron-hole pair for every photon incident with linear photo response. Higher full-well capacity means more number of electrons that can be generated which results in higher dynamic range and better signal to noise ratio. Conversion gain tells the efficiency of signal processing unit, higher the better. Noise sets the dynamic range of the image sensor by defining the lower limit of signal level, continuous advances have been made to reduce the noise and some image sensor can achieve noise level of $1e^-$. Modulation transfer function defines the spatial resolution of the image sensor, which is also the measure of optical crosstalk of the image sensor. Another characteristic which is more of an advantage over CCD image sensor is image lag which is also known as memory effect; defines the image sensors ability to transfer charge from photodiode to floating diffusion. These characteristic parameters define the CMOS image sensor and having a standard measurement procedure for computing this characteristic is necessary.

This report presents the standard measurement procedure to characterize CMOS image sensor. This project is an internal project for Caeleste CVBA, Mechelen, Belgium, hence the measurement procedures are more specific to Caeleste requirement and follows EMVA 1288 standards. Measurement procedure includes all the details to evaluate the characteristic parameter: measurement background, block diagram, procedure and data processing are some of key elements of the procedures. We at Caeleste performed different methods to compute a characteristic parameter accurately and precisely. Also software library and hardware tools were updated for improving measurement accuracy and speed.

Table of Contents

List of Figures

List of Tables

1. Chapter 1

Introduction

Standard and Accurate measurement procedure are key to a successful image sensor design. Importance and need of standard measurement procedure for CMOS image sensor cannot be stressed more. Measurement of any opto-electrical system is an art and it should be precise and accurate. With the ever-growing technology, many technical advances and improvements have been reported in CMOS image sensor, so it becomes challenging to accurately determine the performance parameters, hence it is important to have a standard procedure to characterize the sensor.

The main motivation for my thesis was to get deep understanding of CMOS Image sensor, right from fabrication process to test and characterize the imager. As I first learned about CMOS image sensor, I was fascinated and curious to learn more and decided to do my master thesis in CMOS image senor. During my academic curriculum I did research about CMOS image sensor to get more insight about them and wrote couple of report about it. The thesis project combines my motivation and challenges involved for accurate measurement, so the objective of my thesis work is to standardize measurement procedure for characterization of CMOS Image Sensor. This work presents standard procedure for measurement of characteristic parameters of CMOS image sensor.

As a part of internal project at Caeleste CVBA, Mechelen, Belgium, we performed various measurements and testing to improve and upgrade the measurement procedure for key characteristic parameters that includes quantum efficiency (QE) and responsivity, noise, non-uniformity, dark current, linearity, full well capacity, conversion gain, modulation transfer function (MTF) and image lag of CMOS image sensor, also upgrading software library and hardware to tools for fast and accurate measurements.

The project started with characterizing the LED Light source that is used for performing measurement on CMOS image sensor, as light source is one of the most important device used during measurements of imager. Stability, spatial homogeneity and spectral response are key parameters of light source and knowledge about these parameters assisted in accurate measurements on CMOS image sensor.

This project also addresses the influence and limitations of instruments, environmental condition and interfacing devices on measurement results. The second last chapter of this report consist of standard measurement procedures for all the characteristic parameters.

Thesis Organization

This thesis report consists 6 chapters. The first chapter gives an introduction about this project which illustrates the objective and motivation for this project. Then Chapter 2 gives necessary background information for this project which includes basic overview of CMOS image sensor, the working principle of pinned photodiode and working and timing operation of conventional 4T pixel architecture. Chapter 3 elaborate on importance of LED light source and its characterization and draws conclusion about present light source which will be helpful in building new improved light source at Caeleste. It is followed by Chapter 4 which explains all the characteristic parameters for which the measurements are performed and how they define the performance of the CMOS image sensor. Then Chapter 5 includes all the measurement

procedure that we at Caeleste performed and standardized for characterizing CMOS image sensor that Caeleste design. Chapter 6 contains conclusions of the thesis and the future work.

2. Chapter 2

Overview of CMOS Image Sensor

This chapter gives a brief introduction of the CMOS image sensor and 4T CMOS active pixel sensor which is followed by brief explanation about pinned photodiode and its structure in Section 2.1. Then section 2.2 discusses 4T APS architecture in detail and explains it's working. Section 2.2.1 introduces two electronic shutter mode for CMOS image sensor: Global shutter and Rolling shutter mode.

2.1. Background of CMOS image sensor

CMOS Image Sensor (CIS) are semiconductor device used for making digital camera. They detect information in form of light or any other electromagnetic radiation and create image that represents the information. CMOS image sensors consist of integrated circuits that sense the information and convert it into equivalent current or voltage which is later converted into digital data.

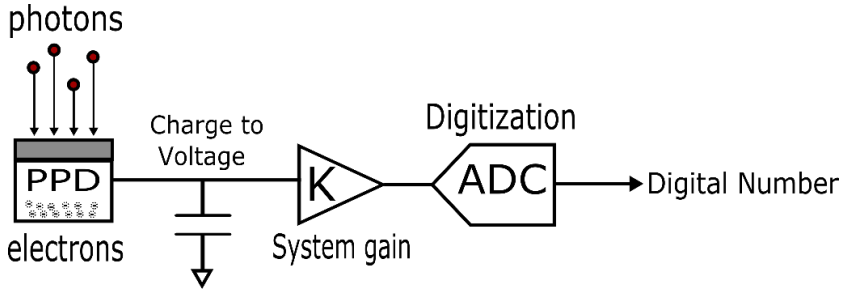


Figure 2.1-1- Physical model of a camera.

In 1967, Weckler proposed the operation of charge integration on a photon-sensing p-n junction which was treated as the fundamental principle of CMOS image sensor [1]. This charge integration technology is still being used in the CMOS image sensors. Shortly, in 1968, Weckler and Dyck proposed the first passive pixel image sensor [2]. In 1968, Peter Noble described the CMOS active pixel image sensor and this invention laid the foundation for modern CMOS image sensors [3]. Yet one had to wait until the 1990s solving the limitations of CMOS technology for active pixel image sensors to develop rapidly [4].

In modern days CMOS image sensor have over taken CCD's in most of the fields. CMOS image sensor as an integrated technology offers wide range of functionality, fast read out, low power consumption, low cost and some better characteristics parameters. Although CCDs had excellent imaging performance, their fabrication processes are dedicated to make photo sensing elements instead of transistors and hence it is difficult to implement good performance transistors using CCD fabrication processes. Therefore, it is very challenging to integrate circuitry blocks on a CCD chip. However, if the similar imaging performance can be achieved using CMOS imagers, it is even possible to implement all the required functionality blocks together with the sensor, i.e. a camera-on-a-chip, which may significantly improve the sensor performance and lower the cost. In 1995, the first successful high-performance CMOS image sensor was demonstrated by JPL [5]. It included on-chip timing, control, correlated double sampling, and fixed pattern noise suppression circuitries.

Active pixel sensors are state of the art implementation of CMOS image sensor, they are integrated circuit consisting of an array of pixels and signal processing unit. They are discussed in detail in the following section. This document addresses important characteristic parameter of a CMOS image sensor and test methodology to compute and evaluate them. Ideally CMOS image sensor should have 100% Quantum efficiency, high frame rate, low noise, linear response, no optical cross talk and no image lag [6].

2.2.Pinned Photodiode

Photodiode is a semiconductor device that converts light into current. Photodiodes works on the principle of photoelectric effect which states that when a photon with sufficient energy is incident on a photodiode it creates electron-hole pair. So these free carriers can be swept with an electric field which results in current in the diode. Pinned photodiode is a variation in photodetector structure with large depletion region, that is currently used in almost all CCD's and CMOS image sensor due to its low noise, low dark current and high quantum efficiency [7].

Pinned photodiode (PPD) has p+/n/p regions with shallow P+ implant in N type diffusion layer over a P-type epitaxial substrate layer refer Figure 2.2-1. Pinning, refers to fermi level pinning or pinning to a certain voltage level, or also forcing or preventing of fermi level/voltage from moving in energy space [8] [9]. A PPD is designed to have the collection region which deplete out when reset. As the PPD depletes it becomes disconnected from the readout circuit and will drain all charge out of the collection region (accomplishing complete charge transfer). The major effect is that the diode can have an exact "empty" state, and allows therefor to do "correlated double sampling" (CDS) in a simple way, cancelling the KTC noise. Also the p+ pinning layer decreases dark current by preventing interface to be depleted and also by absorbing the carriers due to surface generation and preventing them to reach the depletion region. When you design the depletion of the PPD to deplete at a certain voltage you are pinning that PPD to that voltage.

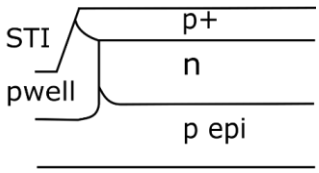


Figure 2.2-1- Cross section of PPD structure. Fundamental Characteristics of a Pinned Photodiode CMOS Pixel [10].

2.3.4T Active Pixel Sensor Architecture

Modern CMOS image sensor uses 4T active pixel sensor system architecture for photo sensing and read out. Earlier CMOS image sensor used to have just photodiode in the pixel for photo charge collection and then came the passive pixel sensor with a photodiode and a switch for row select in the pixel with a column amplifier, the main advantage of PPS was small pixel size but the slow column read out and large noise resulted in design of APS [11]. APS consist of a photodiode, a switch and a pixel level amplifier which result in fast read out and low noise.

There are two common architectures for active pixel sensor 3T and 4T pixel, the difference seems to be of a single transistor but there is a significant difference in their performances. The difference will be discussed in the following section which describes design and working of 4T a pixel. Figure 2.3-1 shows schematic diagram of a 4T pixel. It consists of a p+/n/p pinned

photodiode, a transfer gate TG to transfer charge from photodiode to floating diffusion node FD of capacitance C_{FD} to store charge from the photodiode, with a reset transistor to reset the FD node after every read out, a source follower to isolate sense node from the column bus capacitance and a row select switch. The difference between the 3T pixel and 4T pixel is the transfer gate, which separates photodiode from floating diffusion node or storage node. Hence 4T pixel enables signal buffering that allows to perform integrate while read operation: read out the signal of previous frame while integrating for next frame, which will improve read-out speed and SNR. Also unlike 3T pixel, 4T pixel enables implementation of correlated double sampling CDS which is a useful technique to eliminate reset noise and will be discussed during 4T pixel operation. The other advantage of transfer gate is that it prevents crosstalk between neighboring pixel which also mitigate blooming (overflow of charge to neighboring pixel when pixel saturates) which is a major disadvantage in CCD's [12].

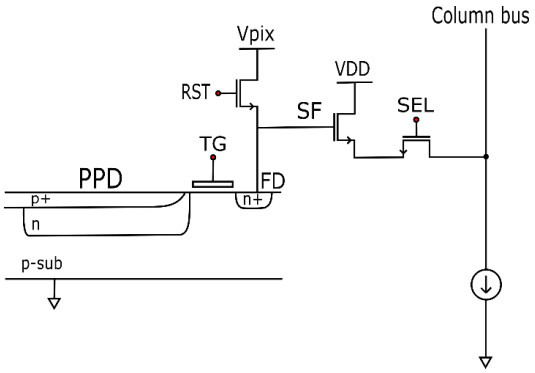


Figure 2.3-1- Schematic of CMOS 4T APS with PPD.

The operation and timing of 4T APS is shown in Figure 2.3-2. First step starts with integration period during which photodiode generate charge according to incident photons and at the same time FD node is reset so that new read out is not influenced by any charges from previous read out, which also gives the reset signal. Now the transfer gate is turned ON and all the charges accumulated in photodiode are transferred to floating diffusion node and then signal is sampled [13] [14] [15] [16].

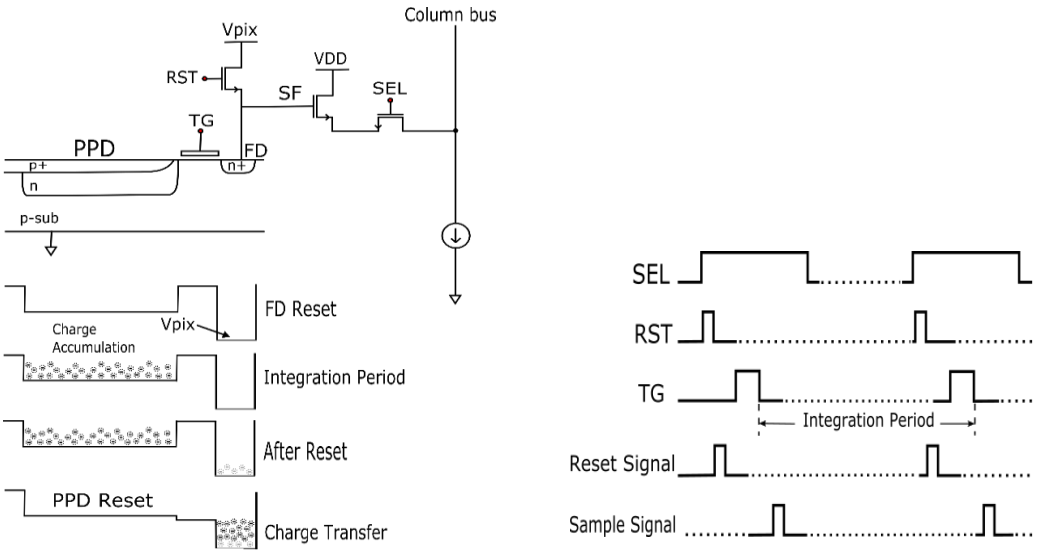


Figure 2.3-2- (a) Operation of 4T APS, (b) Timing diagram of signals.

There are certain sampling techniques used during read out of signal to improve the performance of the image sensor. In CDS technique the pixel is read out twice, once for the reset signal and once for the sample signal and the difference between the two values is the signal value. This technique has quite some advantage in performance of CMOS image sensor, not only it eliminates reset/KTC noise but also suppress 1/f noise (only if it slows that is correlated) [17] [2.17]. The timing diagram in Figure 2.3-2-(b) shows the CDS read out technique in which reset signal is read out and then after transfer of charge the sample signal is readout. The other sampling technique is correlated multiple sampling; in which both reset and signal levels of pixel outputs are sampled for multiple times and summed up, and the difference of the average of the two levels is taken as signal value. This technique helps in reducing thermal and RTS noise refer [18] [2.18] but increases read noise.

2.3.1.Electronic shutter modes

CMOS image sensor can operate in two types of electronic shutter modes namely global shutter mode and rolling shutter mode. In rolling shutter mode pixels are addressed row by row i.e. pixels of one row are exposed/reset simultaneously and then of the next row and so on. This mode of operation causes motion artifact in images, like moving blades of a fan or a helicopter, this problem can be solved in global shutter mode. In global shutter mode all the pixels of the imager are exposed/reset at the same time and the charge from each row is stored and later all the charges are read out row by row. Figure 2.3-3 shows the timing diagram of both the modes of operation [19] [2.19].

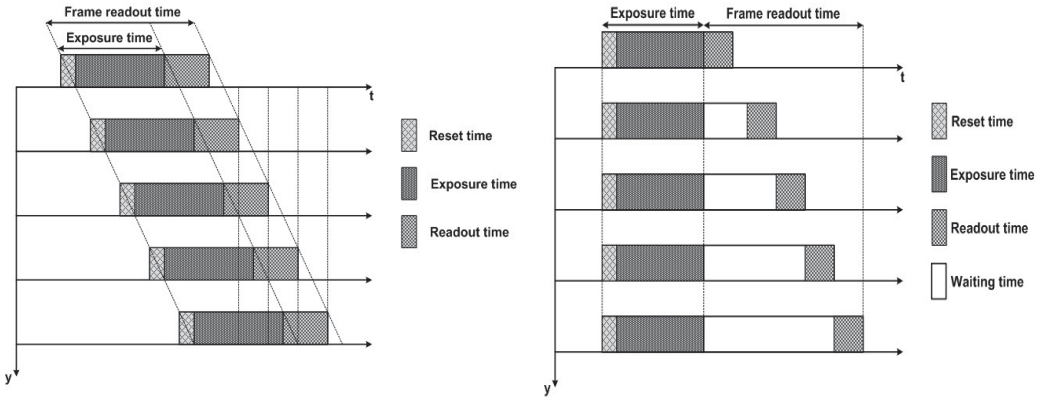


Figure 2.3-3- Working principle of (a) Rolling shutter mode and (b) Global shutter mode [19].

There can be many variations in the pixel architecture depending on the requirement and specification, with the current technology it is possible to design digital pixel sensor which include pixel level ADC. There are various trade-offs between fill factor, read out speed and size of the pixel that leads to the choice for pixel architecture.

3. Chapter 3

LED Light Source

Light emitting diodes (LEDs) offer a number of advantages over conventional light sources, including reduced power consumption, better spectral purity, and longer life time and lower cost. With the rapid development of LED industry during the past decades, LEDs have become popular in an increasing amount of applications and are considered as key replacements for conventional light sources.

The LED-array light sources are used for image sensor characterization; it is essential to know their specifications. LED light source key characteristics are spatial uniformity, temporal and thermal stability and spectral response and this information will help while characterizing CMOS image sensor. Caeleste uses its in-house customized LED light source; the project started with characterizing the LED light source and the measurement results helped in designing improved LED light source. It is very important to have a stable and uniform light source as it directly affects the measurement results of CMOS image sensor. This report present measurement procedure and results that were performed to characterize the LED light source.

3.1.Stability

Light source should emit constant optical power with constant Intensity. There should be no variation in light intensity with time and temperature i.e. light source should have temporal and thermal stability. LED's takes some time to get stable get thermally stable; so it is important to know how much time LED light source takes to get stable. Additionally, LED light source should not have any hysteresis effect.

1. Measured photocurrent of reference Hamamatsu photodiode which can be converted into light intensity for 20 min with a step of 10 seconds at a distance of 20cm between LED light source and photodiode. The intensity of LED source can be set by supply current.
2. The measured photocurrent (A) can be converted into light intensity (W/cm^2) by using conversion factor for a specified light source wavelength available from standard datasheet.
3. Simultaneously measured the temperature of the light source using pt-1000.

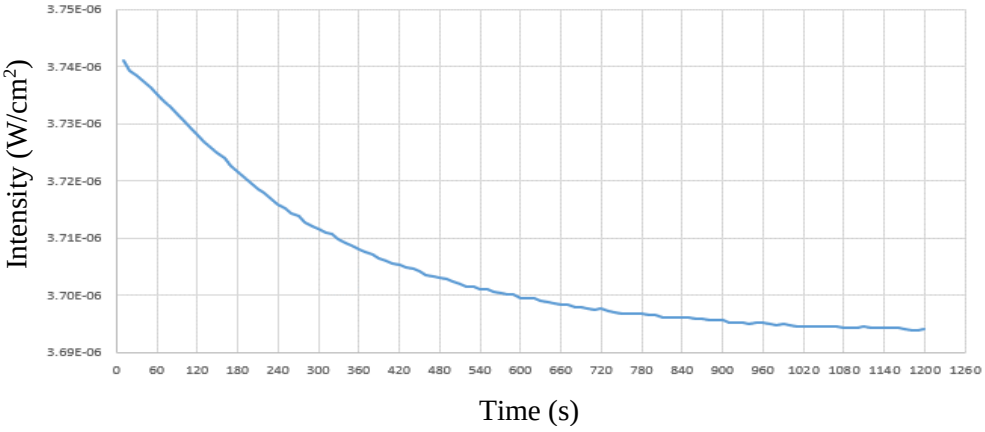


Figure 3.1-1- Intensity of LED light source in function of time.

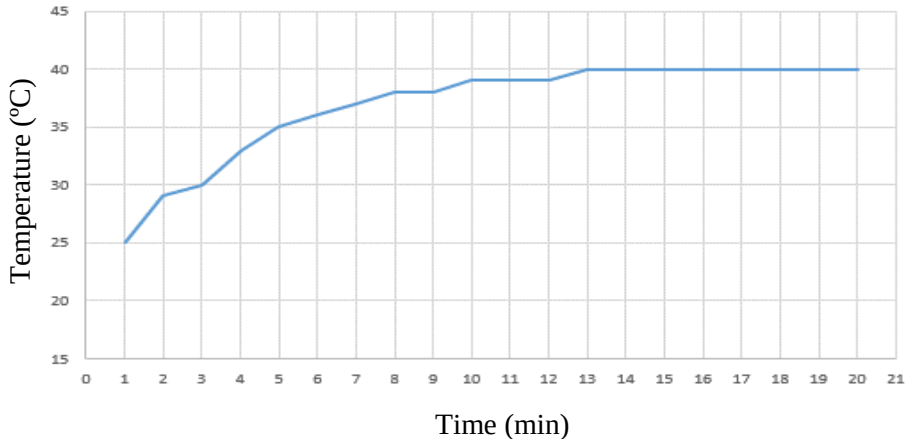


Figure 3.1-2- Temperature of light source in function of time.

3.2.Spectral Response

It is defined by the spectral response of the LED's. Generally, LED's manufacturers provide this information. For characteristic measurement of quantum efficiency, it is important to know the spectral response of the light source.

1. Replaced the lamp of the monochromator with the LED light source.
2. Measured the photocurrent by sweeping the wavelength of the monochromator with a step of 1nm.
3. This measurement was repeated for Red, Blue and Green light source.

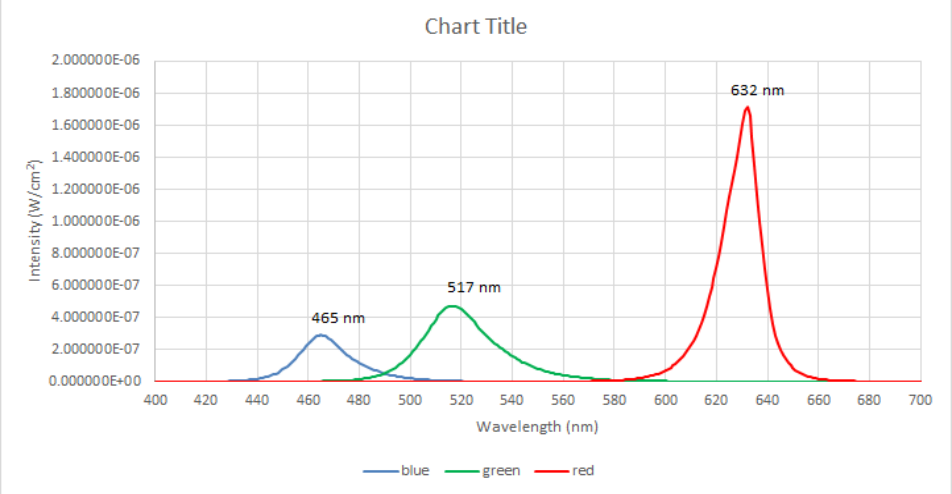


Figure 3.2-1- Spectral response of Blue, Green and Red LED light source.

Table 3.2-1- Spectral specification of LED light source.

LED Light Source	Peak Wavelength	Full Width Half Maximum
Red	632nm	16nm
Green	517nm	30nm
Blue	465nm	22nm

3.3.Spatial uniformity

It is one of the most important characteristic of any light source. LED light source should be spatially uniform i.e. equal level of intensity in space at equal distance from the source.

1. Measured spatial photocurrent of reference Hamamatsu photodiode by scanning the LED light source using precision stepper motor, with a step size of 0.5mm.
2. A 2×2cm^2 3D plot of spatial light intensity is generated for the light field.
3. The measurement was performed at different distance at 10cm, 20cm and 30cm between light source and photodiode.

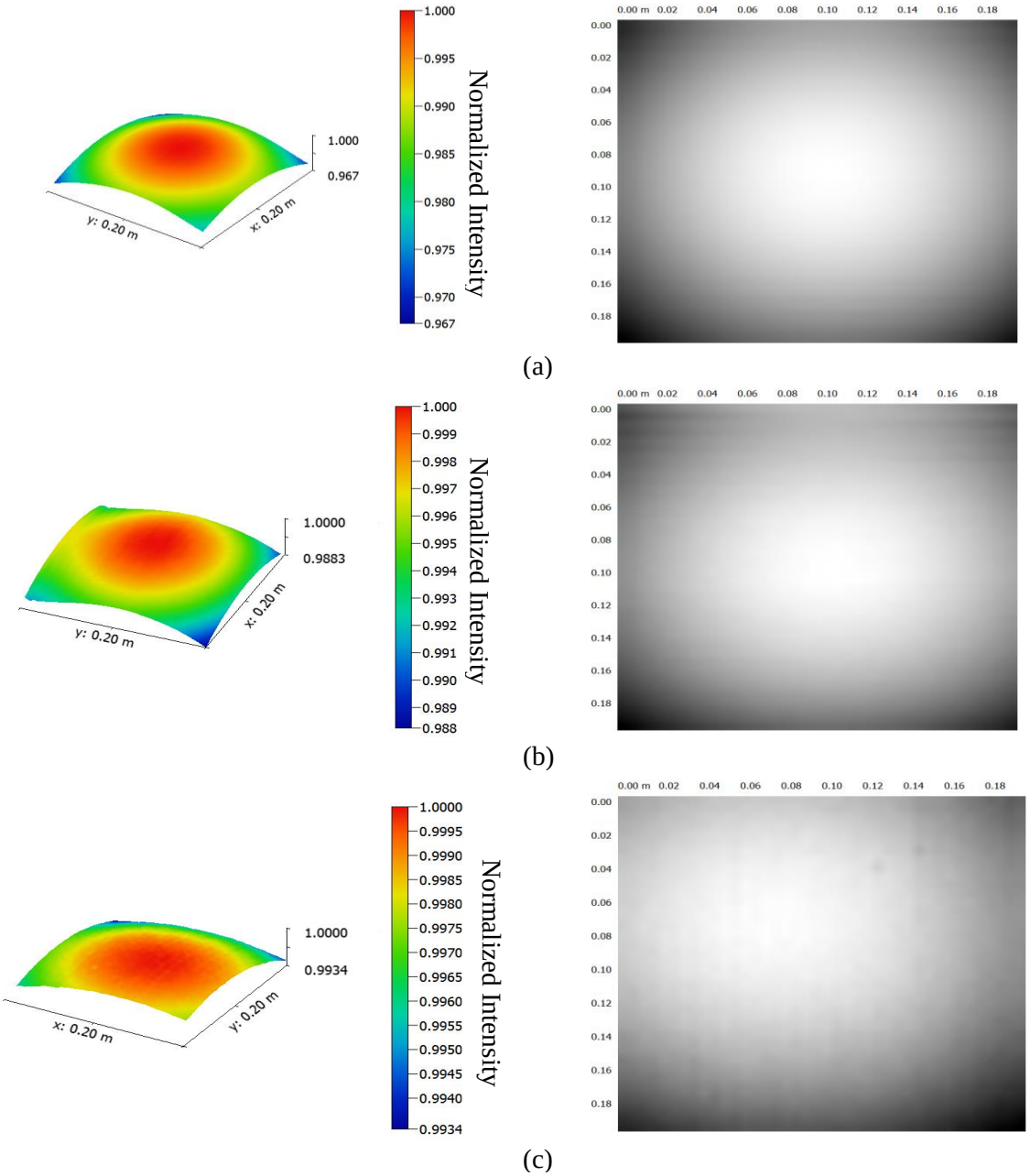


Figure 3.3-1-3D and 2D plot of Spatial intensity of the LED light source at (a) 10cm, (b) 20cm and (c) 30cm distance between monochromator output and photodiode.

3.4.Conclusions

From the measurement results it can be concluded that light source requires about 10 minutes with intensity variation of 1nW/cm^2 to get stable which is in agreement to time taken by light source to get thermally stable. Also from spatial uniformity results it seems that light source gets more uniform as the distance from the light source to photodiode is increased and for 30cm distance the variation is less than 1% for 2×2cm^2 of area which is comparable to size of the image sensors at Caeleste. These measurements where performed at earlier stage and based on the result Caeleste designed an improved light source with optical feedback and PID controller, yielding much better accuracy and stability.

4. Chapter 4

Characteristic Parameter of CMOS Image Sensor

4.1.Quantum Efficiency and Spectral Response

4.1.1.Definition

Quantum efficiency (QE) is one of the most important characteristics of any electro-optical device including the CMOS image sensor. QE is the ratio of average number of electrons generated in the pixel (μ_e) to the average number of impinging photons (μ_p) on the pixel during exposure time.

$$QE(\lambda) = \frac{\mu_e}{\mu_p} \qquad (4.1\text{-}1)$$

When a photon is absorbed by PPD, it generates free carriers (electrons-holes) and these free carriers contributes to the conductivity of the material and the phenomena is known as photoelectric effect. In general, PPD have two modes of operation: Photovoltaic mode, where the electron-hole pair is converted to electron current by the built-in electric field and Photo resistive mode where the free carrier(s) increase the overall conductivity of the material and QE quantifies the relation between photons absorbed and free carriers generated.

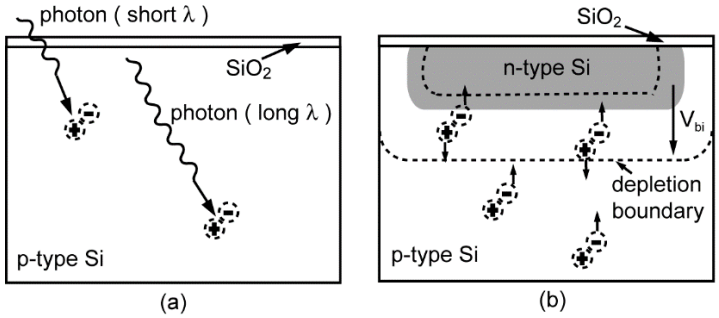


Figure 4.1-1-(a) Electron-hole generations by photons with different wavelength, (b) Photon-generated carriers by a p-n junction/photodiode.

In the field of CMOS image sensors one typically considers total quantum efficiency including fill factor i.e. QE that is referred to the total area occupied by an image sensor single pixel (not only the light sensitive area). QE is expressed in units of percent or simply a value from zero to one.

Ideally for image sensor QE should be as 100 percent i.e. 1 electron-hole pair for every impinging photon; but in reality the photoelectric phenomena in PPD is not perfect there are some limitations associated to it. The first limitation is the loss of impinging photon which can be due to various optical phenomena like photons reflection of the surface or photons absorption by the layers above the depletion region and hence the impinging photons does not reach the photosensitive area of the pixel and thus QE of the system decreases. Second limitation is the inefficiency of the photodiode to collect all the electron-hole pair generated by the impinging photons, this inefficiency is result of free carrier's generation outside the depletion region of the PPD. As the absorption of impinging photons depends upon absorption coefficient of silicon which depends on its wavelength, so photons with longer wavelength will penetrate deep inside the PPD and the free carrier generated by these photons can be outside

the depletion region and hence it will be difficult to collect those carriers' and this will result in lower QE.

Another important characteristic of image sensor is called Spectral Response (SR) or Spectral Sensitivity and it determines how much photocurrent is generated by the image sensor per impinging photon of given energy and it is expressed in units of A/W. Both QE and Spectral response of a photodiode depends on the wavelength of impinging photons hence the term spectral [20].

$$\mathrm{SR}\ [\mathrm{A/W}] = \frac{\mathrm{QE}\cdot\lambda\cdot q}{hc} \qquad (4.1\text{-}2)$$

Where,
λ: Wavelength of impinging photon [nm]
q: electron charge = $1.602*10^{-19}$ [C]
h: Planck's constant = $6.626*10^{-34}$ [J·s]
c: speed of light= $2.99792*10^{10}$ [cm/s]

4.1.2.Measured value versus Theoretical value

Nearly every CMOS image sensor today is fabricated using Silicon material, therefore spectral properties of the image sensor are governed by Silicon and the spectral response of PPD is defined by spectral response of Silicon, hence QE and SR are fabrication process dependent. With the information on absorption coefficient of silicon, thickness of silicon and wavelength of the impinging photons, designer can estimate the QE value of the image senor at designing stage. QE is often measured including fill factor (FF) and designer know how much light (photons) will reach pixel depending on pixel design, so while designing QE value can be estimated [21] [4.2].

Also there are standard set of results based on various test and measurement performed for different pixels to evaluate QE and SR. So if technology, fabrication material and thickness of the material is known, QE of the image sensor can be estimated.

The other way to crosscheck the accuracy of measurement result is to verify the result from alternative method by computing QE from CVF (Charge to voltage factor) data available for the image sensor. This method is described in the measurement procedure.

4.1.3.Fringing effect or Etaloning effect

In measurement results; QE and SR suffer from fringing effect as shown in Figure 4.1-2, this is due to the optical phenomena that occurs within different layers of PPD.

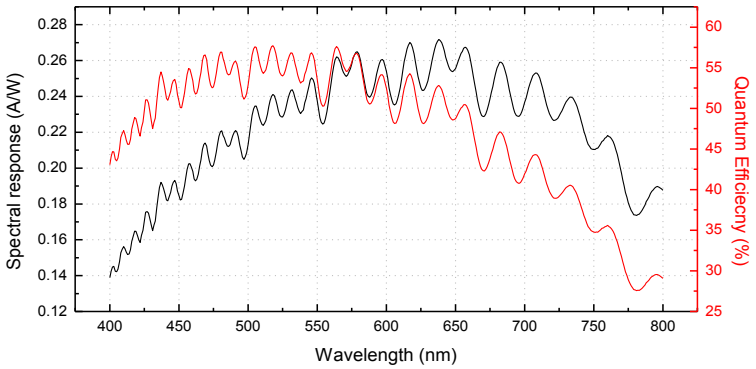


Figure 4.1-2- Fringing effect in QE and SR measurement result.

When a photon with a certain wavelength hits a PPD, it transverse through SiO_2 layer or epitaxy layer before reaching depletion region. Spectral fringing is result of the interference pattern created by photons multiple refection back and forth within this layer which is due to difference in reflective and transmission ability of this layers and the layers behaves as etalons; they don't allow 100% transfer of photons [22] [4.3]. For FSI, fringing is significant in oxide layer and for BSI sensor it is due to epitaxy layer. The Figure 4.1-3 describes the etaloning phenomena for light transmitting through two surface separated by air, similar phenomena occurs when impinging photons transmit through different layers in the pixel [23].

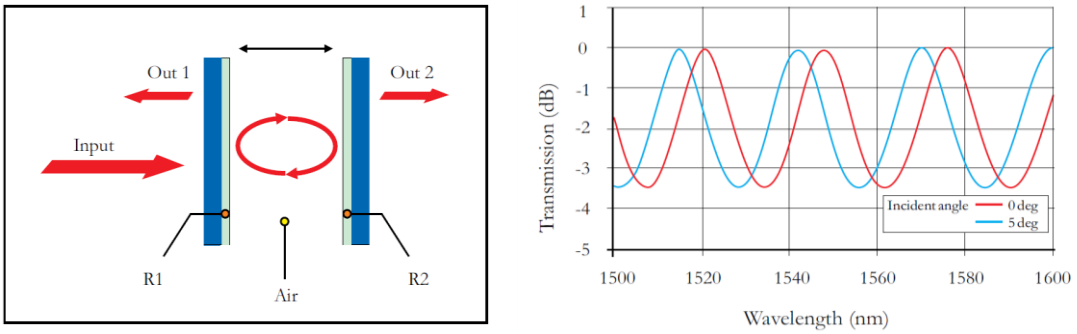


Figure 4.1-3- Principle of etalons and its transmission curve [24].

4.2.Photo Response Curve

4.2.1.Definition

Photo response of an image sensor is the measure of sensor ability to convert optical incident power (number of impinging photons on the sensor for a given integration time) into electrical signal (gain of the system multiplied by the number of electron generated). It is a function of wavelength.

Photo response curve gives a good insight for many characteristic parameters of CMOS image sensor like CVF from known QE value, Non-linearity, Saturation voltage and full well capacity of the image sensor. Photo response is generally performed for specific spectral wavelength (mostly at peak response wavelength).

4.2.2.Non-Linearity

Ideally CMOS image sensor should have a linear behaviour i.e. it should respond linearly with incident light (photons) but due to nonlinear devices in the pixel and in the signal processing unit, the sensor deviates from linear response. The major source of non-linearity in the image sensor comes from the source follower MOSFET in the 4T pixel design, as it is used as trans-impedance amplifier and its gain depends on the source resistance which induces non-linearity [25]. Other transistors are just used as a switch and thus do not contribute much to non-linearity of the image sensor. Other sources of non-linearity are [26]:

1. Image lag.
2. The non-linearity of the photodiode or of the floating diffusion.
3. Non-linearity's in the further downstream analog processing and multiplexing.

Non linearity can be classified into Integration non-linearity (INL) and Differential non-linearity (DNL). INL is the measure of maximum deviation or error from the ideal response

and DNL quantifies deviation of two consecutive values corresponding to ideal values. In case of image sensor only INL is calculated as there are no DAC used in imager for signal processing. So photo response tells the response of the sensor for incident optical power and ideally it should be linear and INL is evaluated from the actual response to the ideal response of the sensor [27].

$$\mathrm{NL}[\%] = \frac{\mathrm{E_{max}}}{\mathrm{FS}} \times 100 \qquad (4.2\text{-}1)$$

4.2.3.Full well capacity (FWC)

FWC defines/tells the charge generating/storing capacity of the pixel and the state when pixel reaches its FWC is called as saturation. Image sensors absorb photons and generate free carriers (electrons-hole), depending on the way PPD is designed. Usually a PPD has a wide depletion region where all the free carriers are collected and the amount of charge that can be collected by PPD depletion region defines the FWC. The other way to define FWC is the capacity of floating diffusion to store charges and of a good pixel design both the quantity should be equal. So according to design there is a limit to number of free carriers that can be generated in PPD and it should be equal to amount of charge that can be stored on the floating diffusion.

FWC can be determined by photo response of the image sensor, but first it is important to define the full well capacity. For the measurements at Caeleste, FWC is defined w.r.t saturation level of the image sensor; so it is defined as the point of intersection best line fit for 10% and 90% saturation level and best line fit.

4.2.4.Saturation (V_{sat})

A pixel is said to be saturated when it reaches its FWC for incident optical power and the corresponding output voltage is called as saturation voltage.

$$\mathrm{FWC} = \mathrm{V_{sat}} \cdot \mathrm{C_{eff}} \qquad (4.2\text{-}2)$$

4.2.5. Charge to voltage factor (CVF)

Charge to voltage factor also known as conversion gain is the conversion factor that tells how much output voltage (V_{signal}) is generated corresponding to number of electrons (μ_e) generated by the pixel. It is defined either at floating diffusion or at the output of the image sensor. The conversion gain is one of the most important parameters of a CMOS imager pixel. The linearity and uniformity of the pixel response, light sensitivity, and the pixel noise are all influenced by its value and distribution.

$$\mathrm{CVF}\ [\mu\mathrm{V/e^-}] = \frac{\mathrm{V_{signal}}}{\mu_e} \qquad (4.2\text{-}3)$$

It can be calculated from photo response curve; as the curve gives the relation between output voltage and incident optical power (i.e. the number of photons). The number of photons can be converted into number of electrons from the QE value of the pixel at that wavelength; which results in relation between the output voltage and the electron (charge) i.e. the CVF. The CVF can be classified as internal and external conversion factor. Internal conversion factor take path gain into the consideration which allows Caeleste to compare different pixel design, independent of the surrounding circuit and external CVF is the property of the chip, which is important for their customers.

An alternative method used for determining CVF is mean-variance curve; it is based on basic law of physics. The method employs the fact that the photon shot noise (PSN) level is proportional to the square root of the signal level [28].

$$\sigma_n = \sqrt{q.V_{signal}} \quad (4.2\text{-}4)$$

Where σ_n is the photon shot noise and V_{signal} is the output signal and q is the photo charge. Hence CVF can be determined as slope of curve between Noise (Variance) and mean signal.

4.3.Modulation Transfer Function

4.3.1.Definition

MTF (modulation transfer function) is the image sensor's ability to transfer contrast at particular spatial frequency. It is a direct measure of senor image quality and resolution. Ideal response of a pixel is an impulse of certain width defined by pixel pitch, but pixel suffers from optical cross talk i.e. pixel shares its information with neighbour pixels. Hence this results in more or less a Gaussian response rather than an impulse response. This optical cross talk can be quantified by MTF which is basically FFT of the pixel response. Note that, although the term is "optical" crosstalk, the underlying mechanisms are both optical and electrical in nature [29].

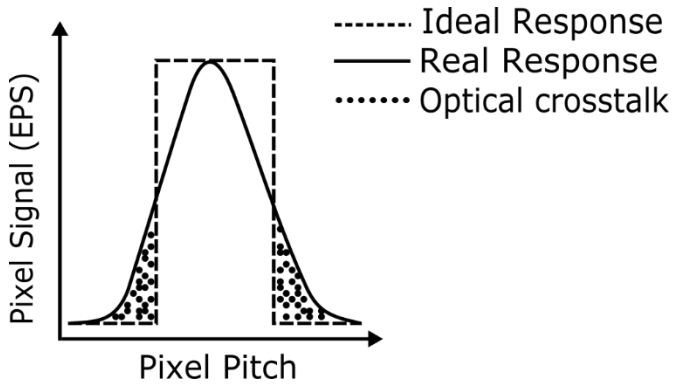


Figure 4.3-1- Optical cross talk in a single pixel.

To compute frequency response, the pixel can be exposed to alternating black/white lines pattern with a line thickness equal to pixel pitch. The special situation where the black/white spatial line frequency corresponds exactly to 2 times pixel pitches is called the "Nyquist frequency" (f_N). The MTF at Nyquist frequency is often used as an optical crosstalk measure [30].

4.3.2.Source of cross talk

1. Lateral diffusion when incident light (photons) bounces around inside the chip.
2. Optical crosstalk due to different optical phenomena like diffraction, refraction, scattering, interference and reflection of light (photons).
3. Electrical crosstalk results from photo-generated carriers having the possibility to move to neighboring charge accumulation sites (pixel).
4. Other physical limitation like the limit of the main (cameras) lens or angle of incidence of incident light also contribute to crosstalk.

4.3.3.Slanted Edge method

There are several methods to measure the MTF of the image sensor like Sine target method, knife edge method but this method suffers from drawbacks of long computation time and also need large number of images respectively. But the slanted edge method is fast and requires just 1 image to compute MTF. The method is based on ISO 12233 standard; consists in imaging an edge onto the detector, slightly tilted with regard to the rows (or the columns). So a vertically oriented edge allows obtaining the horizontal Spatial Frequency Response (SFR) of the sensor. In that case, the response of each line gives a different edge spread function (ESF) due to different phase and the ESF data is used to compute the MTF [29].

4.3.4.Correct Image and Lens Setting

Images that are captured suffer from non-uniformities and offset. So to measure MTF accurately, captured images need to be corrected first and to remove this artifact flat-filed correction is applied.

$$C = \frac{(\text{Edge}-\text{Dark})\cdot m}{(\text{Light}-\text{Dark})} \qquad (4.3\text{-}1)$$

Where,
C: Corrected image
Edge: Raw image of target edge.
Dark: Dark frame.
Light: Flat-field light image.
m: Average value of (Light-Dark)

It is also important that the slanted edge is located at the center of the image and the center of lens; the image contrast and resolution are typically optimal at the center of the image, and due to imperfect lens; deteriorate toward the edges of the field-of-view. Another important point to consider is that while capturing images using Lens, the image should be perfectly focused because an unfocused image degrades the MTF because of aliasing and can lead to phase-reversal (i.e. black and white segment get reversed). Basically any optical aberration degrades MTF of the system [31] [32].

4.4.Noise and Non-uniformity

4.4.1.Definition

Every integrated circuit suffer from noise and so do CMOS image sensor. Image sensor convert light information into electrical signal and while processing information CMOS image sensor suffers from two types of noise, temporal noise and spatial noise. Noise is any fluctuation in signal value and noise waveform can be conventionally modelled as random process. Mathematically noise power can be estimated by computing variance of the signal. Therefore, variance of the fluctuation in signal gives the mean signal and thus mean number of electrons [33].

Before measuring noise, it is important to know what type of noise comes from which part of the sensor. Below in the Figure 4.4-1 a CMOS image sensor based on standard 4T pixel architecture is shown and it describes what kind of noise originate from specific part of the pixel. This Figure 4.4-1 only indicates major source of noise, there are other components as well which contribute to the total noise of the system.

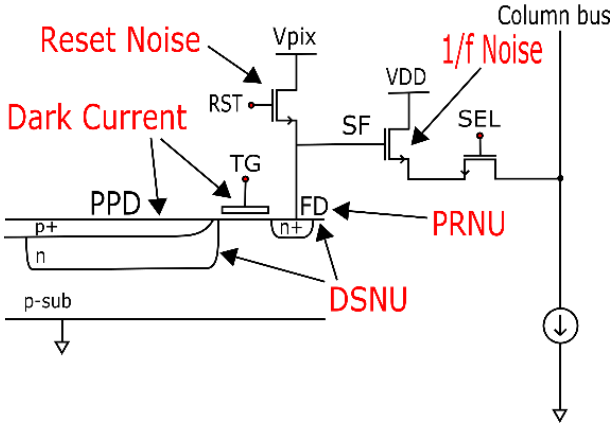


Figure 4.4-1- Major Noise sources in a 4T pixel PPD.

4.4.2.Temporal Noise

Temporal Noise is variation in pixel response for constant illumination over a period of time. It can be evaluated by computing the standard deviation of pixel response for every pixel through number of frames taken over period of time for specific illumination and then taking mean of all the pixel values of the image sensor. Temporal noise consists of following noise components:

1. Johnson noise of all resistive components (Thermal noise).
2. 1/f noise of the MOS transistors (mainly from the source-follower in CMOS pixels).
3. KTC noise (if not cancelled by CDS).
4. Quantization noise of the ADC.
5. EMI and other interference.
6. Timing jitter.
7. Any other noise contribution of extra electronics on-chip or off-chip.

To estimate the noise in the image sensor, it is important to model noise mathematically. The most often used random variable in noise analysis is Gaussian random variable. It is used to describe the magnitude distribution of a large variety of noise sources, including thermal noise, shot noise and 1/f noise. If noise signal is a Gaussian random variable, then the white noise process is a white Gaussian noise (WGN). Two of the most important noise processes in integrated circuits, thermal and shot noise, are modelled as WGN processes [33].

4.4.2.1.Photon shot noise

It is the noise due the statistical variation on generated/excited electron-hole pairs due to random arrival of impinging photons under illumination and it obeys a Poisson statistic. Therefore, magnitude of the photon shot noise (σ_n) equals to the square root of mean signal (V_{signal}) [28]:

$$\sigma_n = \sqrt{q \cdot V_{signal}} \tag{4.4-1}$$

It is possible to compute conversion gain/CVF from this relation -

$$CVF = \frac{\sigma_n^2}{V_{signal} \cdot A} \tag{4.4-2}$$

Where, A is the voltage gain of the signal processing unit.

It is the most fundamental noise of any photonic system as it comes from the basic law of physics and not from the image sensor design. Its square root dependency on illumination level is utilized to characterize the image sensor. The conversion gain derived from this relation is very accurate as it is not influenced by CMOS image sensor design.

4.4.2.2.Dark current shot noise

It is the noise of CMOS image sensor under no illumination. Dark current shot noise is result of thermally random generation of electrons-hole pair in dark i.e. dark current and it depends exponentially on temperature. As it also follows poisons statistics dark current shot noise is given as:

$$\sigma_n = \sqrt{V_{dark}} \tag{4.4-3}$$

4.4.2.3.Read Noise

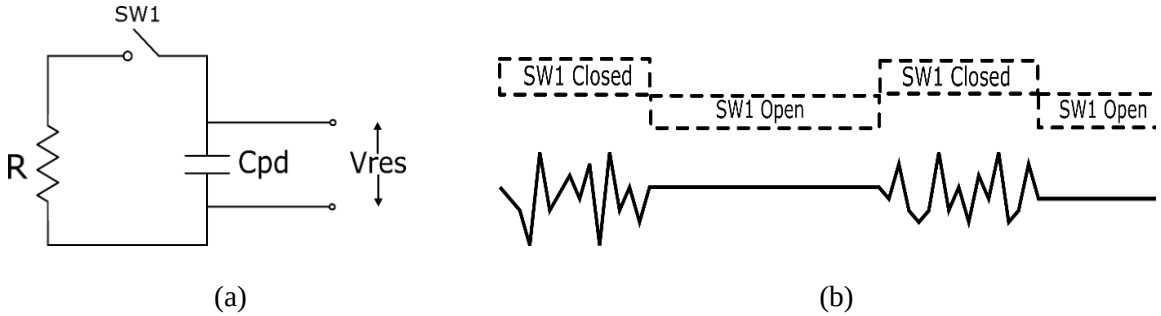
Read noise or temporal noise in dark is the inherent noise of a sensor and is equal to noise level under no illumination. It is result of various noise sources like- KTC noise, ADC noise, temporal row noise and interference pattern. It is measured for very short integration time (t_{int}) to avoid DSNU and is measured in dark to avoid photon shot noise on temperature stable system. It is calculated by evaluating temporal signal variance over series of frame for individual pixel and then by taking the average over all the pixels, which tells noise over image.

4.4.2.4. Reset Noise

It is the thermal noise of the reset switch sampled over capacitor and also known as "KTC noise". It is the noise sampled on the floating diffusion capacitor C_{pd} due to charge redistribution and uncertainty of charge on the capacitor when the reset switch is turned ON/OFF. A reset transistor and capacitor can be modelled as a resistor in series with a switch and a capacitor as shown in Figure 4.4-2 [34]. But in modern CMOS image sensor using CDS technique, the reset noise is cancelled out. Matematically reset noise is given as:

$$V_{res} = \sqrt{\frac{KT}{C_{pd}}} \tag{4.4-4}$$

Where,
V_{res}: Reset noise voltage.
K: Boltzmann constant.
T: Absolute temperature
C_{pd}: Diffusion capacitance or floating node capacitance.



(a) (b)

Figure 4.4-2- (a) nMOS transistor modelled as switch and a resistor in series with floating diffusion capacitor, (b) KTC noise sampled when switch is opened and closed.

4.4.2.5.1/f Noise

1/f noise also known as flicker noise is the noise power density which is inversely proportional to frequency. 1/f noise depends upon MOS technology and with the downscaling of transistors it is becoming dominant source of read noise. The main sources of 1/f noise in CMOS image sensor are the MOS transistors. It is a result of random fluctuations in charge carrier due to random capture and emission of carriers by traps in the gate oxide and also due to fluctuations in mobility carrier. These two effects are correlated and are considered together during noise modelling [35]. There are different models to quantify 1/f noise of which K.K Hung model [36] is most commonly used.

4.4.3.Spatial noise

Spatial noise is the variation of pixel response within a frame i.e. pixel to pixel variations which are steady over time. This are statistical variations in "offset" and "gain" of the pixel over a frame and are fixed at constant illumination therefore referred as fixed pattern and hence fixed pattern noise (FPN). Spatial noise can be evaluated by computing coefficient of variation for a frame.

4.4.3.1. Fixed pattern noise

Fixed pattern noise is the noise dark and is often considered to be as an offset because the variations are fixed for a given pixel at fixed integration time. FPN in dark is result of mismatch in transistors of a single pixel and mismatch of column-level transistor of the image sensor and also due variation in dark currents of the different pixels of the image sensor. Generally, FPN due to mismatch in transistors is cancelled during correlated double sampling (CDS) performed while reading out pixel signal. FPN in dark due to dark current signal is often referred as dark signal non uniformity (DSNU) which represents variation in dark signal of pixels in a pixel array.

4.4.3.2.Photo response non-uniformity

Photo response non uniformity (PRNU) also called as "gain" noise is result of variation in gain value (photo-responsivity) of a pixel in a pixel array and it is proportional to photo illumination. There are several components of PRNU, primarily it is due to imperfection in photodiode during fabrication process. The imperfection in fabrication process can result in different value of diode junction capacitance, which result in variations in depletion width and hence variation in pixel photo response. Also mask misalignment results in different size of photodiodes which result in variation in active region of pixels in a pixel array [25]. It is primarily due to photodiode capacitance variation, photodiode collection volume variations, variation in device gain and capacitance variations.

It is important to understand that while measuring PRNU, FPN in dark (DSNU) is also included. FPN in dark due to transistor mismatch is supposed to be cancelled during CDS and to eliminate FPN due to dark current, either FPN in dark need to be subtracted or measurement should be performed by keeping minimum integration time to make DSNU negligible.

4.5.Dark current

4.5.1.Definition

Dark current is the small leakage current that flows in a photosensitive device under dark condition and in CMOS image sensor it is due to random generation of electrons and holes in PPD. Dark current is a result of many different semiconductor physical phenomena that occurs

in the PPD, these phenomena's will be discussed further in this section. But dark current is not just restricted to PPD, FD and TG also contribute to total dark current of the pixel. Dark current is the major source of noise in CMOS image sensor, it directly affects the signal to noise ratio (SNR) of the image sensor as it defines the lower limit of the signal that can be detected. Dark current itself does not create problem for image sensor as it can be calibrated by performing "Dark frame subtraction" from the signal frame but it is temporal variability and spatial non uniformity of dark current which introduces noise in the system. In high speed cameras the dark current is negligible because of the very short integration time and then read noise dominates.

4.5.2.Mechanism

Dark current is PPD response under no illumination. Its generation depends on fabrication technology and design of the CMOS imager, of which major factors influencing dark current are silicon defect density, the electric field of the photo-sensing element, and operation temperature. There are several components of dark current which are result of different physical mechanism. Following section will discuss all the mechanism briefly.

4.5.2.1.Generation center

Generation center is a physical location which is responsible for random generation of electron and holes that result in dark current. This generation centers are result of impurities, dislocation faults, vacancies, mechanical stress, lattice mismatch, interface states etc., which are side effect (by-product) of fabrication process. The physical size of a generation center is in the order of 1-2 inter-atomic distances.

4.5.2.2.Thermal generation

Thermal generation and recombination is a common phenomenon in any opto-electrical devices. In absence of photons (light) the thermal generation-recombination of carriers is a result of impurities and defect in crystal. In silicon trap assisted generation-recombination (GR) is a dominant phenomenon compare to other generation-recombination mechanism. Due to impurities and defect in crystal, some electrons have energy above fermi level which result in indirect transition of electron in conduction band and thus contribute to dark current. The electrons in transition between bands passes through a state created in the middle of the band gap by an impurity in the lattice. The impurity state can absorb differences in momentum between the carriers, and so this process is the dominant generation and recombination process in silicon and other indirect bandgap materials [37]. Also PPD is usually reverse-biased therefore the minority carrier concentration is lower than the equilibrium concentration hence generation process is dominant over recombination process to re-establish the equilibrium. This complete process can be characterized by Shockley–Read–Hall (SRH) process; hence the rate of electron-hole pair generation inside the depletion region is given as [38]:

$$G = \left[\frac{\sigma_p \sigma_n \vartheta_{th} N_t}{\sigma_n \exp\left(\frac{E_t - E_i}{KT}\right) + \sigma_p \exp\left(\frac{E_i - E_t}{KT}\right)}\right] \qquad (4.5\text{-}1)$$

Where,
σ_n: electron capture cross section,
σ_p: hole capture cross section,
υ_{th}: thermal velocity of either electrons or holes (assuming they are equal),
N_t: density of the generation centers (silicon defects),
E_t: defect energy level,

E_i: intrinsic energy level,
K: Boltzmann's constant and
T: absolute temperature.
The dark current caused by thermal generation in the depletion region is:

$$J_{gen} = \int_0^W q\,G\,dx \approx qGW = \frac{qn_iW}{\tau_g} \quad (4.5\text{-}2)$$

Where,
W: Depletion width,
q: electronic charge,
n_i: intrinsic concentration, and
τ_g: generation life time.
As shown in Eq. (4.5-2), the dark thermal generation current is proportional to the intrinsic concentration n_i. The temperature dependence of n_i is given by [3.3]:

$$n_i = \sqrt{N_c N_V}\exp\left(-\frac{E_g}{2KT}\right) \quad (4.5\text{-}3)$$

where N_c and N_v are the carrier densities and E_g is the energy bandgap. By combining both Eq. (4.5-2) and Eq. (4.5-3), it can be concluded that the temperature dependency of thermal generation current is proportional to the exponential value of a half silicon bandgap.

4.5.2.3.Surface generation

The phenomena behind surface generation is same as of thermal generation, the only difference is the location of the traps, defects or impurities. Surface generation is result of imperfection in Si-SiO_2 interface of PPD, hence the lattice structure becomes non-uniform at the interface which results in traps at the surface. There are different measures taken to reduce surface generation, p+ implant is one of the technique i.e. PPD is not completely depleted a thin p+ layer is left, so if there is any surface generation the carriers will be absorbed in the thin p+ layer itself [39].

4.5.2.4.Tunneling

It is a common phenomenon that occurs in heavily doped p-n junction which result in thin depletion layer. During tunneling process the valance band carriers "penetrates" through the bandgap into the conduction band instead of overcoming the barrier see Figure 4.5-1. This phenomenon may occur in PPD as they are heavily doped and under high electric field carriers can tunnel through p-substrate to n+ depletion region and contribute to dark current of the pixel. As more doping result in more impurities and thus more dark current, therefore PPD are not completely depleted during doping as it increases dark current, and a thin P layer between the Si-SiO_2 keeps the dark current in control by preventing the free carriers to go in depletion region [37].

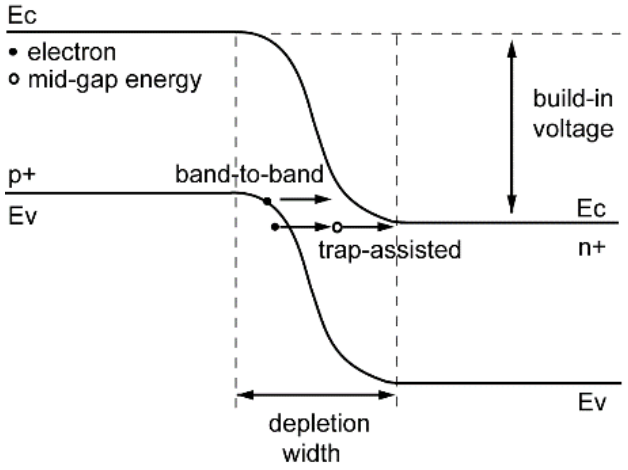


Figure 4.5-1- Energy band diagram of tunnelling process of a heavily doped p-n junction [40].

Avalanche multiplication/Impact ionization is a rare phenomenon and is most unlikely to occur in 4T pixel PPD as the biased applied to the pixel is not strong enough to initiate the phenomena.

4.5.2.5.Diffusion current

It is the current that is a result of difference in concentration level in two regions in a p-n junction diode, so whenever one region is at higher potential than other than there is net movement of carriers from higher concentration to lower concentration and this movement of charge carrier's result in current called as diffusion current. Due to heavy doping concentration in n-layer, the dark current due to diffusion of holes is in negligible. Hence the equation of dark current due to diffusion current of electrons can be given as [38]:

$$J_{diff} = \frac{qD_n n_{p0}}{L_n} = q\sqrt{\frac{D_n}{\tau_n}} \cdot \frac{n_i^2}{N_A}$$

Where,
J_{diff}: Diffusion current due to electrons,
n_{p0}: electron concentration in boundary condition,
D_n: electron diffusion coefficient,
L_n: diffusion length,
τ_n: carrier lifetime,
n_i: intrinsic concentration.
From the temperature dependency of intrinsic concentration n_i, the equation shows the temperature dependency of diffusion current on the exponential value of one silicon bandgap.

4.5.2.6.Dark current from fabrication process

Also most CMOS image sensor uses STI (Shallow Trench Isolation) for decreasing cross talk, but on other hand the oxide layer of STI creates traps and this traps will release in undesired trapping and releasing of carriers that will result in dark current and 1/f noise.

4.6.Image Lag

4.6.1.Definition

Image lag is defined as memory effect of the pixel due to insufficient charge transfer to floating diffusion from PPD. In a 4T pixel, the complete transfer of signal charge from the PPD to the floating diffusion node is critical to the pixel performance in terms of noise and image lag. The potential profile under the transfer gate must be properly tailored to establish the proper barrier height between the photodiode and the floating diffusion node to achieve full charge transfer when the transfer gate is high. The relative position and the geometry of the n-type diffusion layer of the PPD, combined with the shape and size of the transfer gate directly affect key device characteristics such as noise, dark current, and image lag [41]. It is like a trade-off between large full well, low dark current and significant image lag for a 4T pixel designer.

4.6.2.Cause of Image Lag

1. Potential barrier between PPD and TG
   Pinned photodiode can be modeled as lumped RC network, hence there is certain time constant (delay) for charge to deplete through the diode or the potential barrier and if the pulse applied to transfer gate is smaller than the time constant (delay) then it will result in residual charge at photodiode therefore the lag in the sensor.
   Or,
   In a 4T pixel, this is due to the emptying time constant of charge that is limited by the potential barrier. Charge transfer from Pinned photodiode can mathematically corresponds to current through MOSFET in sub-threshold or the forward current through a diode. Hence it requires certain time constant to overcome the barrier and this emptying time constant can then be directly translated into image lag.
2. Insufficient electric filed
   If the applied electric filed to transfer charges is not enough then there will be residual charges in the pixel which will result in image lag.
3. Trapping effect in TG
   It is a common phenomenon in MOSFET, where charge carriers are being trapped at transfer gate interface i.e. in the channel and this traps can capture free carriers and can release then anytime which result in Image lag.
4. Large signal level
   Large signal can also result in image lag as some electrons can fall back in PPD if the applied potential to TG is not enough w.r.t large amount of charge [26].

The amount of carriers not transferred obeys a Poisson distribution. Hence if 100 electrons are not transferred, the uncertainty on that number is 10. This is obvious when considering a transition from light to dark. The noise of the pixel's value is the square root of the remaining charge. One should be aware that it is as well valid for a steady-state situation: although no apparent image lag is visible, in steady state, as the charges lost by the previous frame are compensated by the loss of the next frame, the losses are uncorrelated.

5. Chapter 5

Measurement Procedure Overview

It is important that the reasons for undertaking a measurement are clearly understood so that the measurement procedure can be properly planned. Good planning is vital in order to produce reliable data to time and to cost. The planning process should cover aspects such as the objectives of the measurement, background information, selection of the method, the required level of accuracy and confidence and finally reporting [42]. The following measurement procedures are planned for accurate, robust and fast measurements.

The following section includes measurement procedures for all the characteristic parameters of CMOS Image sensor. Measurements were performed on different samples and prototype sensors; different characteristics parameters are computed using different standard methods. The measurements procedures are more friendly with Caeleste working environment, but can be used to characterize any CMOS Image sensor in general with suitable hardware and software tool. The tests are performed with different sensor settings to get the more accurate and precise results, like for some measurements high gain mode is used and for some measurements low gain mode, some measurements are performed for same integration time and constant light source intensity and some with changing integration time and changing light source intensity.

Measurement procedures include all the basic information about the characteristic parameter, image sensor specification and settings, measurement setup and environmental condition under which measurement procedure is/should be performed. The procedures also suggest alternative methods to compute characteristic parameter and also to cross check the measurement results. Measurement procedures includes pictures and graphs for some standard results that may or may not comply with actual results. All the measurement procedure complies within EMVA Standard 1288 and MTF measurement procedure is based in ISO 12233 test standard. Along with creating measurement procedure, the software library and hardware equipment are also updated for fast and accurate measurements.

5.1.Quantum Efficiency and Spectral Response

5.1.1.Objective

Quantum efficiency (QE) is one of the most important characteristics of any electro-optical device including the CMOS image sensor. QE is the ratio of average number of electrons generated in the pixel to the average number of impinging photons on the pixel during exposure time. This document gives brief introduction on quantum efficiency and spectral response, explains necessary measurement procedures and provides data analysis protocol.

5.1.2.Measurement Background

5.1.2.1.Method Description

In the field of image sensors one typically considers total quantum efficiency: QE that referred to the total area occupied by an image sensor single pixel (not only the light sensitive area). QE expressed in units of percent or simply a value from zero to one. Another important characteristic of a photodiode or image sensor is called Spectral response and it determines how much photocurrent generated by the DUT per impinging photon of given energy. Therefore, it is expressed in units of A/W. Both, QE and Spectral response of a photodiode depend of the wavelength of impinging light. By knowing QE one can derive SR and vice versa. To estimate quantum efficiency first determine the Spectral Response of the DUT. Depending on the project two situations can appear:

1. Image sensor design has special QE test structure intended for QE and SR measurements only.
2. No specialized QE test structure is available within current image sensor project.

5.1.2.2.Specialized QE structure

Dedicated QE structure consists of a set of pixels (referred as QE or ‘real’ pixels) identical to those of the original sensor’s pixel array in terms of photodiode, transfer gate and metallization properties. Additionally, QE structure has a set of guard pixels that prevent charge leakage into the QE pixels from outside. The QE test structure is either a part of the image sensor die (hence, accessed via sensor’s dedicated bonding pads/pins) or designed as a separate die. Figure 5.1-1 shows typical layout of a QE structure: three pads corresponding to substrate (SUB), QE pixels (diode) and the guard pixels and the pixel array consisting of real pixels surrounded by guard pixels. Figure 5.1-2 shows electrical connection of the QE structure. Both, guard and real pixels are biased with external power supply. Ampere meter is placed into QE-pixel net for photocurrent measurements.

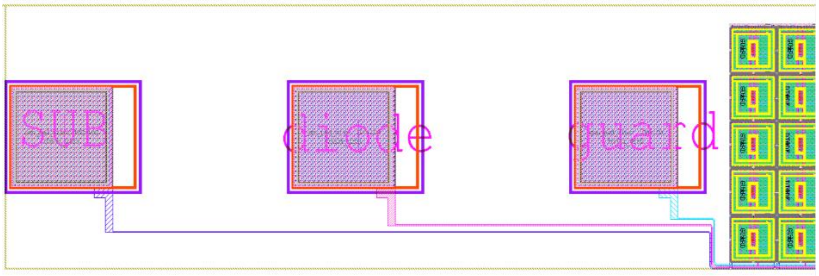


Figure 5.1-1-Layout of QE test structure (full pixel array is not shown).

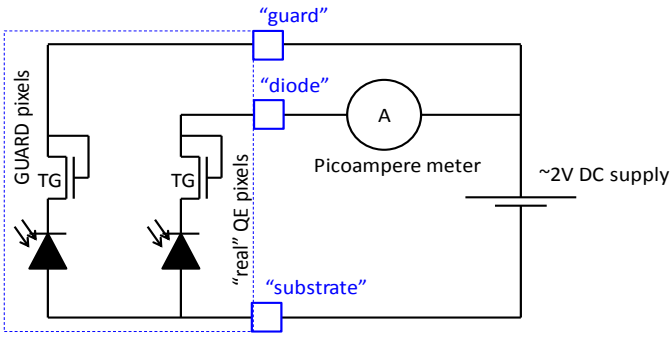


Figure 5.1-2-QE structure connection diagram.

5.1.3.Measurement Setup

5.1.3.1.List of Equipment

From the explained above it is clear that one needs a monochromatic light source, a calibrated light intensity radiometer and high precision electrometer for photocurrent measurements. The setup should be mounted on a rigid optical bench or breadboard in the dark cabinet. Typical list of equipment employed for QE measurements:

1. DUT/QE structure.
2. Reference detector (Hamamatsu Si reference diode).
3. Power supply (CI-0033 TTi_QL355TP_Power_supply).
4. Electrometer (CI-0013 Keithley 6514 electrometer).
5. Monochromator (TMc300).

5.1.3.2.Block Diagram

The test setup for QE measurements is shown in Figure 5.1-3. The setups assume the use of a monochromator for obtaining QE at various wavelengths whose principle of operation is explained in detail in Figure 5.1-3. The light from a broadband light source enters monochromator via its input slit and is then collimated onto a diffraction grating. The latter splits continuous spectrum into separate wavelengths reflected at different angles. Focusing mirror collimates this light onto the output reflector, which together with the exit slit outputs light of one particular wavelength. Light from the output of the monochromator is then projected onto device under test. Both, monochromator and electrometer are computer-controlled. In case the measurement needs to be done only at one wavelength, a nearly monochromatic light source can be used such as LED whose emission spectrum is known. For this procedure, consider the use of monochromator for obtaining QE and SR spectra in this document.

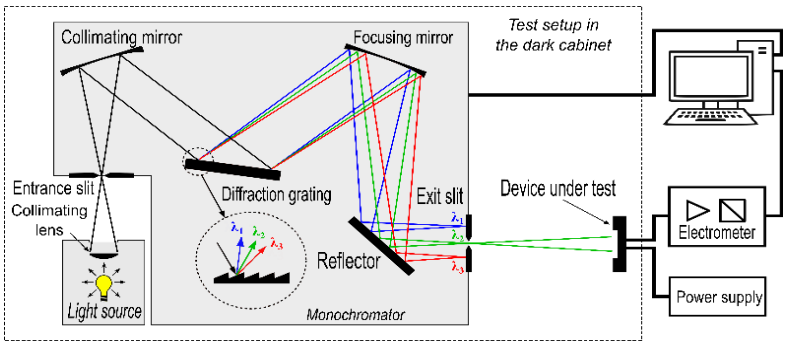


Figure 5.1-3- QE measurement setup schematic.

5.1.3.3.Software Tools

1. Iron Python-based Caeleste software environment.

5.1.4.Measurement Procedure

5.1.4.1.Algorithm

The principle behind QE measurement is that, first measure the current of reference-calibrated detector as a function of wavelength to determines the intensity (standard conversion table provided from photodiode manufacturer) of light incident on the structure i.e. irradiance E (directly tell about the number of impinging photons on the DUT). Then measure the current of DUT structure for same illumination as a function of wavelength to calculate spectral response (which tell us about the number of electron generated) and if CVF is known then the output voltage of the DUT can be converted into number of electrons at the output of DUT. Now calculate QE as the ratio of number of electrons by number of photons.

5.1.4.2.Procedure

1. Use the Bentham Halogen light source with its dedicated stable current source and allow it to reach thermal equilibrium for at least 20 minutes after switching it ON. (Do not switch of the light source until you complete all the measurement.)
2. Select the appropriate slit width at the monochromator exit to get desired bandwidth of light from Figure 5.1-4. (Refer Table for Slit width vs Bandwidth from monochromator user manual).

Grating Groove Density (l/mm)		2400	1200	600	400	300	150	100	75	50
Reciprocal Dispersion (nm/mm)		1.35	2.70	5.40	8.11	10.81	21.62	32.42	43.23	64.85
Slit widths (mm)	Part no. for pair of slits	Bandwidth produced (nm)								
0.05	FS (0.05)	0.07	0.14	0.27	0.41	0.54	1.08	1.62	2.16	3.24
0.1	FS (0.10)	0.14	0.27	0.54	0.81	1.08	2.16	3.24	4.32	6.48
0.2	FS (0.20)	0.27	0.54	1.08	1.62	2.16	4.32	6.48	8.65	12.97
0.37	FS (0.37)	0.50	1.00	2.00	3.00	4.00	8.00	12.00	16.00	23.99
0.4	FS (0.40)	0.54	1.08	2.16	3.24	4.32	8.65	12.97	17.29	25.94
0.5	FS (0.50)	0.68	1.35	2.70	4.05	5.40	10.81	16.21	21.62	32.42
0.56	FS (0.56)	0.76	1.51	3.03	4.54	6.05	12.10	18.16	24.21	36.31
0.74	FS (0.74)	1.00	2.00	4.00	6.00	8.00	16.00	23.99	31.99	47.99
1	FS (1.00)	1.35	2.70	5.40	8.11	10.81	21.62	32.42	43.23	64.85
1.12	FS (1.12)	1.51	3.03	6.05	9.08	12.10	24.21	36.31	48.42	72.63
1.48	FS (1.48)	2.00	4.00	8.00	12.00	16.00	31.99	47.99	63.98	95.97
1.85	FS (1.85)	2.50	5.00	10.00	15.00	19.99	39.99	59.98	79.98	119.97
2	FS (2.00)	2.70	5.40	10.81	16.21	21.62	43.23	64.85	86.46	129.69
2.78	FS (2.78)	3.76	7.51	15.02	22.53	30.05	60.09	90.14	120.18	180.27
3.7	FS (3.70)	5.00	10.00	19.99	29.99	39.99	79.98	119.97	159.96	239.93
4	FS (4.00)	5.40	10.81	21.62	32.42	43.23	86.46	129.69	172.92	259.39
5.56	FS (5.56)	7.51	15.02	30.05	45.07	60.09	120.18	180.27	240.36	360.55
8	FS (8.00)	10.81	21.62	43.23	64.85	86.46	172.92	259.39	345.85	518.77
10	-	13.51	27.02	54.04	81.06	108.08	216.16	324.23	432.31	648.47

Figure 5.1-4 - Slit width and bandwidth configuration [43].

5.1.4.2.1.Measurement from known CVF

1. Place the reference detector under illumination and measure the current at wavelength of interest. Use the reference detector calibration data (in A/W) to calculate irradiance

E in [W/cm^2] from the detector output current [A]. This will result in mean number of impinging photons (μ_p).

2. Record exactly the location of the reference detector (within a few mm in X, Y and Z) and place the DUT to be measured at the same location.
3. Now illuminate the DUT at the same wavelength, capture the image for known integration time (t_{int}), subtract it from ambient or dark image and measure the mean output signal [Volts] of DUT. One can evaluate mean number of electron generated (μ_e) by using mean output signal and CVF.

Note: It is advisable to take same ROI as size of the reference detector to get accurate result.

5.1.4.2.2.Measurement from dedicated QE structure

1. Place the reference detector in the light beam and measure the intensity from the Monochromator at every wavelength of interest. Use the reference detector calibration data (in A/W) to calculate light intensity in [W/cm^2] from the detector output current [A] and subtract ambient light from the measurement.
2. Record exactly the location of the reference detector (within a few mm in X, Y and Z) and place the QE STRUCTURE to be measured at the same location.
3. Perform a wavelength scan with exactly the same parameters (bandwidth, wavelengths etc.) to measure spectral response and subtract ambient light from the measurement. (Step size of the scan should be lower than the FWHM of the light).

5.1.4.3.Replacing reference detector with DUT at same location

1. One method to replace the device and place it at the exact same location is to use a Convex Lens and place it between reference detector and Monochromator.
2. Adjust the position of lens so that one should get a small spot of light (convex lens converges light at focal point) focused on at middle of detector reference detector and now place the DUT and align it so that you get the same small beam of light on at middle DUT.

5.1.4.4.Pay attention

1. The f-number of the light towards the QE structure and the reference detector should be identical.
2. Both QE structure and reference detector should be perfectly homogenously illuminated.
3. Signal level should be well above the dark current for accurate result.

5.1.4.5.Further recommendations

1. If no specific F-number is required the measurement is best done using plain Monochromator output (at F/ 4 diverging) in the dark cabinet, without any additional optics.
2. Direct or diffuse illumination will give different results for QE structure; clearly report the method of illumination. The Bentham reference diode can be used for both.
3. Make sure that your Connection cables are not twisted or rounded as at times communication freezes.

5.1.5.Accuracy of the method

Accuracy depend on the non idealities and assumptions. Following factor may affect accuracy of the result:

1. Unstable light source.

2. Misalignments during replacing reference photodetector with QE structure. Below you can see the deviation in measurement results when setup was assembled and dismantle setup 4 times. Maximum difference in QE is of around 2.68%.

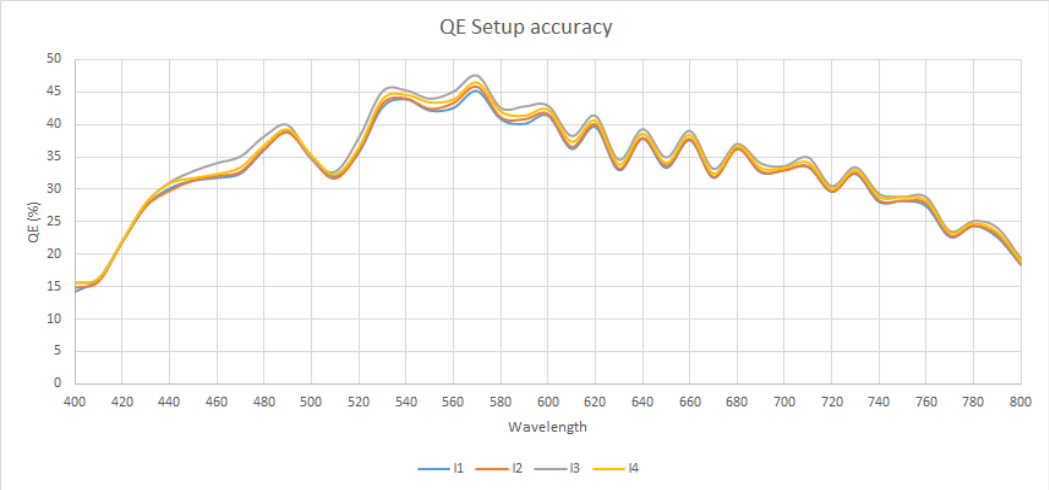


Figure 5.1-5 - QE setup accuracy diagram.

5.1.6.Alignments

Alignment is very important for this measurement. During measurement following points should be kept in mind:

1. Light source should be parallel to DUT and the distance between them should be within light source uniformity and intensity range (Refer monochromator data sheet).
2. QE structure and reference detector should be placed exactly at the same position.

5.1.7.Data Processing

5.1.7.1.Calculating SR and QE

5.1.7.1.1.CVF method

Quantum efficiency (QE) is the ratio of the average number of electrons generated in the pixel (μ_e) to the average number of impinging photons in that pixel (μ_p) and is wavelength (λ) dependent:

$$QE(\lambda) = \frac{\mu_e}{\mu_p} \tag{5.1-1}$$

The mean number of photons μ_p incident over the pixel area A [cm^2] during the integration time t_{int} [s] can be computed from the known irradiance E [W/cm^2] with:

$$\mu_p = \frac{A \cdot t_{int} \cdot E}{hc/\lambda} \tag{5.1-2}$$

Where c and h are the speed of light and Planck's constant respectively.

In addition, mean number of electrons is given by-

$$\mu_e = V_{out}/CVF \tag{5.1-3}$$

Photo induced charge Q_{ph} is the number of electrons μ_e multiplied by the elementary charge q.

$$Q_{ph} = \mu_e \cdot q \tag{5.1-4}$$

Spectral response (SR) expressed in units A/W is the ratio between the average photocurrent (I_{ph}) and irradiance per pixel area (E*A):

$$SR = \frac{I_{ph}}{E \cdot A} = \frac{Q_{ph}/t_{int}}{E \cdot A} = \frac{q(V_{out}/CVF)}{t_{int} \cdot E \cdot A} \tag{5.1-5}$$

Where $I_{ph} = Q_{ph}/t_{int}$ is the pixel photocurrent expressed as a photoinduced charge collected during the integration time t_{int}.

Where,
λ: Wavelength [nm]
q: electron charge = $1.602*10^{-19}$[C]
h: Planck's constant = $6.626*10^{-34}$ [Js]
c: speed of light= $2.99792*10^{10}$ [cm/s]

5.1.7.1.2.QE structure measurement method
Spectral response is ratio of current generated and the power of incident light on the QE structure. Now determine SR by the following formula:

$$SR[A/W] = \frac{I}{E \cdot Ar} \tag{5.1-6}$$

The quantum efficiency can be determined from the spectral response by replacing the power of the light at a particular wavelength with the photon flux for that wavelength. This give:

$$QE[\%] = \frac{SR \cdot hc}{\lambda \cdot q} \times 100 \tag{5.1-7}$$

Where,
I: measured current through "QE structure" [A]
A: Effective area of "QE structure" [cm^2]
P: Intensity of light, which is incident on the sensor [W/cm^2]

5.1.8.Graphs and Figures

Figure 5.1-6 shows a standard plot from one of the Caeleste image sensor. Both QE and SR are plotted in function of wavelength, where QE is expressed in percentage (%), SR in (A/W) and wavelength in (nm).

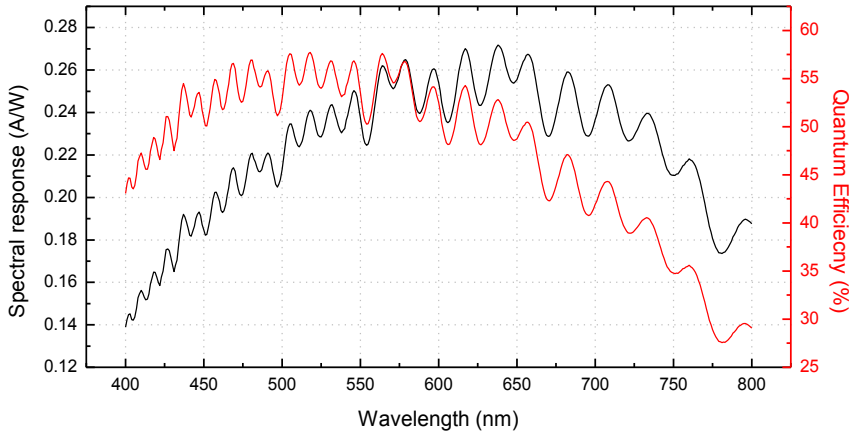


Figure 5.1-6 - Plot of QE and SR in function of wavelength.

5.2.Photo Response

5.2.1.Objective

Photo response of an image sensor is the measure of sensor ability to convert optical incident power (number of photons hitting the sensor for a given integration time) into electrical signal (gain of the system multiplied by the number of electron generated). It is a function of wavelength. The main result from this procedure are photo response curve and charge to voltage factor (CVF). Further from the curve one can determine other parameter like saturation voltage, non-linearity and full well capacity.

5.2.2.Measurement Background

5.2.2.1.Method description

To determine photo response two parameters are required, 1st the incident input optical power, which can be calculated using reference calibrated photodetector for particular wavelength and 2nd is the output electrical signal of the DUT, which is determined by measuring the output signal over a fixed integration time for different illumination of sensor i.e. from saturation to dark. Hence at a given wavelength and for given input optical power one can evaluate photo response.

5.2.3.Measurement Setup

5.2.3.1.Block Diagram

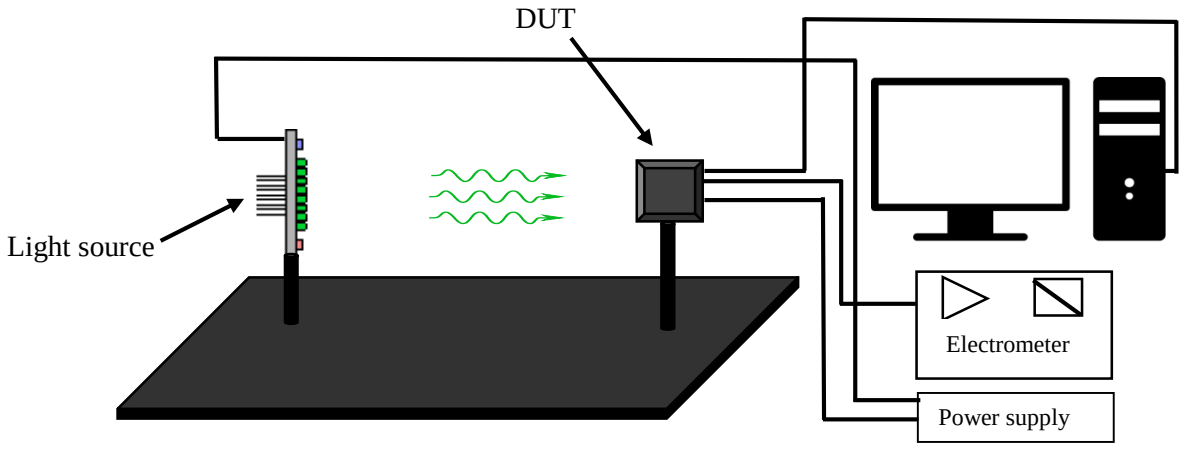


Figure 5.2-1- Setup diagram for PR and CVF measurement.

5.2.3.2.List of Equipment

1. DUT.
2. Reference detector (CI-0046 Hamamatsu Si reference diode H1).
3. Mounting rails.
4. Power supply (CI-0033 TTi_QL355TP_Power_supply).
5. Electrometer (CI-0013 Keithley 6514 electrometer).
6. Caeleste LED light source.
7. Connection cables.

5.2.3.3.Software Tools

1. Iron Python-based Caeleste software environment.

5.2.4.Measurement Procedure

5.2.4.1.Algorithm

To calculate the optical input power from a light source, measure the current of reference-calibrated detector to irradiance E (standard conversion table provided from photodiode manufacturer). Simultaneously measure the output electrical signal of the DUT i.e. the output of ADC for different illuminations (saturation to dark) to obtain photo response curve. Measuring the DUT signal means: grab a sequence of at least 10 images and calculate mean of all the images to obtain final image.

5.2.4.2.Setup

1. Place the light source, DUT and reference detector at an appropriate position on the optical bench.
2. Switch on the light source at maximum power and allow 10 minutes to get it thermally stable.
3. Check that with the given placement deep saturation of the DUT can be reached near maximum light source power such that you can still take couple of measurement after saturation to get better result.
4. Record the position of light source, DUT and reference detector in X, Y and Z to within a few mm. Make sure the parts can be removed from the optical bench and positioned later on the same location.

5.2.4.3.Determine the correction factor for the reference detector

1. In case if you cannot replace the DUT and reference detector at the same location one can use 2nd reference detector to cancel the error caused due to misalignment.
2. Remove the DUT and place a 2nd reference detector at the DUT position.
3. Measure the light intensity at both detector locations and determine the scaling factor. Check that this factor is constant for different light intensities.
4. Remove the 2nd reference detector.

5.2.4.4.Procedure

1. Record the DUT identification and test conditions in the test report header template and place the DUT at its position.
2. Control the LED source current to obtain different steps (smaller the better) in illumination from saturation to dark.
3. At each illumination step acquire series of images (e.g. 10, 20) from the sensor and take mean of the images [volts] and simultaneously measure the irradiance E [W/cm^2] of illumination using reference detector, apply the displacement correction factor to the detector measurement value.
4. Also capture image in dark (μ_{dark}) for offset reference.
5. Plot the mean output signal (μ- μ_{dark}) [Volts/DN] versus the light intensity [W/cm^2] to obtain Photo response curve.

5.2.4.5.Further recommendations

1. Light source / illumination field non-flatness, non-uniformity: For large DUTs it is advisable to use only a window of pixels to calculate the pixel average. This window must have about the same size as the reference detector and be located at the X/Y position of the 2nd reference detector during the determination of the displacement correction factor.

5.2.5.Accuracy of the method

This measurement procedure accuracy depends on how accurately you measure the incident optical power and also on the stability of light source. Also it is important to accurately process the images that are captured to remove any offset and non-uniformities, but this procedure is accurate to obtain non linearity, V_{sat} and full well capacity.

5.2.6.Data Processing

Characterization of Photo response curve give us information of number of useful parameters like full well capacity, CVF, linearity, saturation. One can evaluate all this parameters using data collected by following the above mentioned procedure.

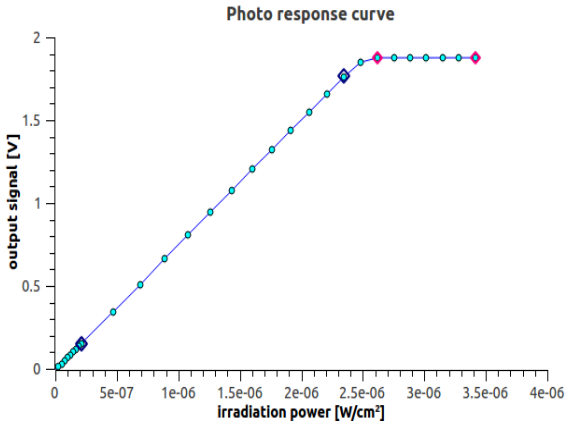


Figure 5.2-2- Photo response curve.

5.2.6.1.Non Linearity

It is the measure of maximum deviation of output signal from an ideal (best line fit) response of the sensor. It is defined within a certain range of illumination intensity for e.g. 10% to 90% of saturation level. In this case the best fit line will be the straight line joining output voltage at 10% and 90% saturation value.

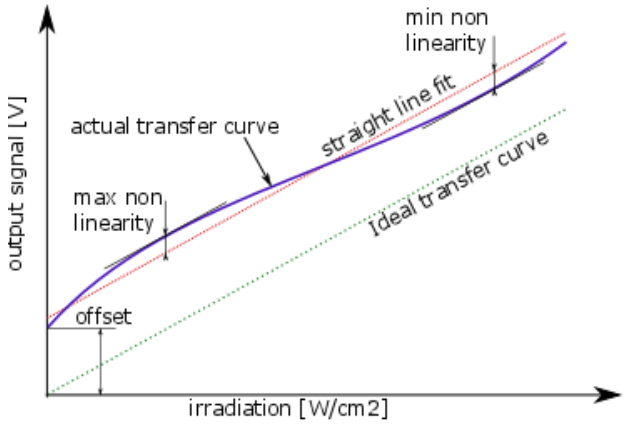


Figure 5.2-3- Non Linearity from Photo response curve.

$$NL[\%] = \frac{E_{max}}{FS} \times 100 \qquad (5.2\text{-}1)$$

Where,
NL = Nonlinearity,

E_{max} = maximum (most positive) error from best-fit straight line [Volts/DN],
FS = full-scale value or maximum output of sensor [Volts/DN].

Note that E_{max} is the maximum deviation in any direction. The full-scale value is the value of the highest measurement taken which is still within the linear performance range. Normally this is set to be as close to the actual full-scale capability of the A/D converter as is practical during camera calibration.

5.2.6.2.Saturation (V_{sat})

It is the point in the photo response curve after which the output of the sensor does not depend upon illumination level i.e. point after which no more charge can be stored in floating diffusing.

5.2.6.3.Full well capacity

It is important how you define the full well capacity. Here full well capacity is referring to saturation level of the sensor. Hence it is given as point of intersection of best line fit (i.e. the line joining the 10% and 90% saturation level point) and best fit line for V_{sat} on photo response curve as shown in the

Figure 5.2-4 below.

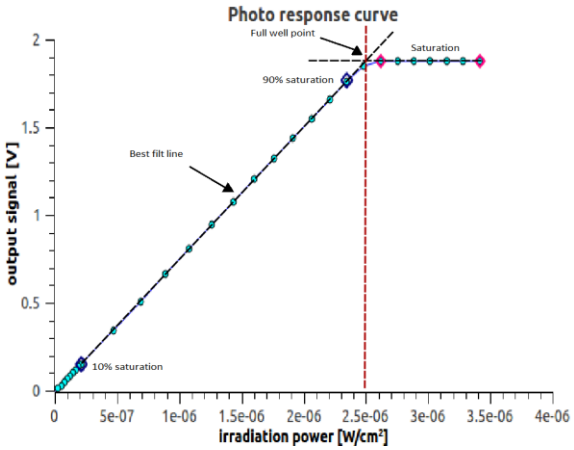


Figure 5.2-4- Photo response curve showing full well capacity.

5.2.6.4.Alternatively, from mean-variance method

It is the point in the noise plot where the temporal noise curve Figure 5.2-5 abruptly drops. This is because the image sensor is in saturation region hence the output signal does not depend on illumination level anymore and therefore average noise goes down.

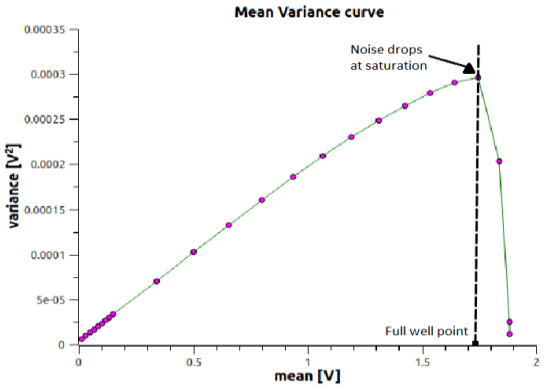


Figure 5.2-5- Full well from Noise curve.

Or,

It is defined as the maximum charge storing capacity of floating diffusion, hence it can be calculated as:

$$FWC = V_{sat} \cdot C_{eff} \quad (5.2\text{-}2)$$

Where,
FWC = Full well capacity (number of electrons),
V_{sat} = Saturation voltage,
C_{eff} = Effective floating diffusion voltage.

5.2.6.5.CVF

Charge to voltage conversion factor tells you voltage equivalent of the number of electrons generated and vice versa. Depending upon the data available there are two different methods to determine CVF:

5.2.6.5.1.First method- (Photo response curve if Quantum efficiency value is known)

Convert the irradiance into number of electrons generated by following calculations:

Firstly, the mean number of photons μ_p absorbed over the pixel area A [cm^2] during the integration time t_{int} [s] can be computed from the known irradiance E [W/cm^2] with:

$$\mu_p = \frac{At_{int}E}{hc/\lambda} \quad (5.2\text{-}3)$$

Where c and h are the speed of light and Planck's constant respectively and λ is the wavelength of incident light.
Now, Quantum efficiency (QE) is the ratio of the average number of electrons generated in the pixel (μ_e) to the average number of photons absorbed in that pixel (μ_p) and therefore one can calculate number of electron generated for known QE:

$$\eta(\lambda) = \frac{\mu_e}{\mu_p} \quad (5.2\text{-}4)$$

Now you can plot V_{out} (mean output signal) versus μ_e (number of electron) to get a CVF plot. And the 1st derivative (slope) of the curve will give CVF [Volts/e^-].

Where,
λ: Wavelength [nm]
q: electron charge = $1.602*10^{-19}$[C]
h: Planck's constant = $6.626*10^{-34}$ [Js]
c: speed of light= $2.99792*10^{10}$ [cm/s]

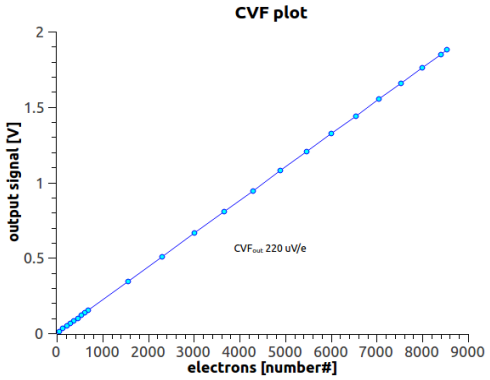


Figure 5.2-6- CVF from Photo response curve if Quantum efficiency value is known.

5.2.6.5.2.Second method (Mean-Variance method or PTC curve method)

The mean-variance method employs the fact that the photon shot noise (PSN) level is proportional to the square root of the signal level. Thus, by measuring signal and its variance one can obtain the CVF. Take following steps to evaluate CVF:

1. From the set of captured image, compute the signal variance (temporal noise) and mean signal for each pixel over the series of images at a given illumination level (however, watch out for non-linearity). The result is the average over all pixels of signal variance (σ^2) [Volts^2/DN^2] and mean signal (μ) [Volts/DN] for that series and μ_{dark} serves as offset reference.
2. Plot (σ^2) and (μ - μ_{dark}) and in the range where (σ^2) has a square root behavior i.e. the linear region, the ratio between (σ^2) and mean (μ - μ_{dark}) is actually the square root of the number of photo electrons and the slope of this curve will result in CVF [Volts/e$^-$].

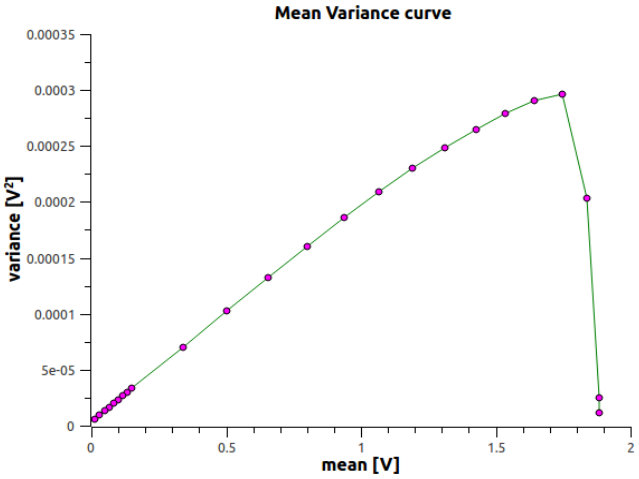


Figure 5.2-7- CVF from PTC curve using mean variance method.

5.2.7.Accuracy of the method

Among the two method described; the photo response curve method seems more accurate as mean-variance method is only accurate for perfectly linear image sensors. So photo response curve is preferred for the CVF measurement.

5.3.Modulation Transfer Function

5.3.1.Objective

Modulation Transfer Function (MTF) is the measure of sensors ability to transfer contrast at a particular spatial resolution from the object to the image. It quantifies the overall imaging performance of a system in terms of resolution and contrast. The following measurement procedure is based on Slanted-Edge method and will result in Edge spread function (ESF), Line spread function (LSF) and MTF of the sensor.

5.3.2.Measurement Background

5.3.2.1.Slanted-Edge method

The slanted-edge method based on ISO 12233 standard consists in imaging an edge onto the detector, slightly tilted with regard to the rows (or the columns). So, a vertically oriented edge allows obtaining the horizontal Spatial Frequency Response (SFR) of the detector. In that case, the response of each line gives a different ESF, due to different phases [29] .

5.3.2.2.Method description

In order to calculate MTF of the system, first determine ESF which is the summation of response of all the lines perpendicular to the slanted edge as response of single line will give an under sample ESF, so by combining multiple lines with a slightly different position of the edge (hence the slant edge) the sampling of the ESF becomes much better. Then calculate the Line spread function (LSF) which is the derivative of ESF. Then by performing Fast Fourier transform (FFT) of LSF will result in Optical Transfer Function (OTF). Finally, the MTF of the sensor is the modulus of the OTF. Figure 5.3-3 shows numerical relationship between all the parameters.

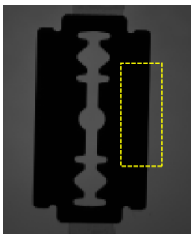


Figure 5.3-1-Slanted edge with ROI.

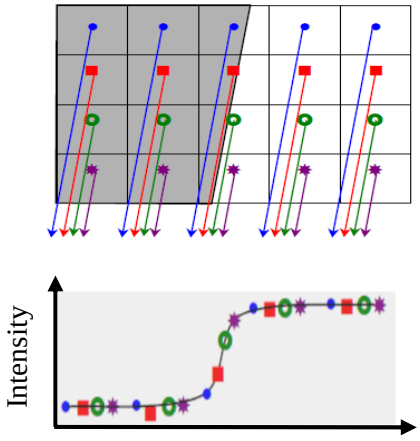


Figure 5.3-2-ESF curve obtained by projecting data from corrected image [29].

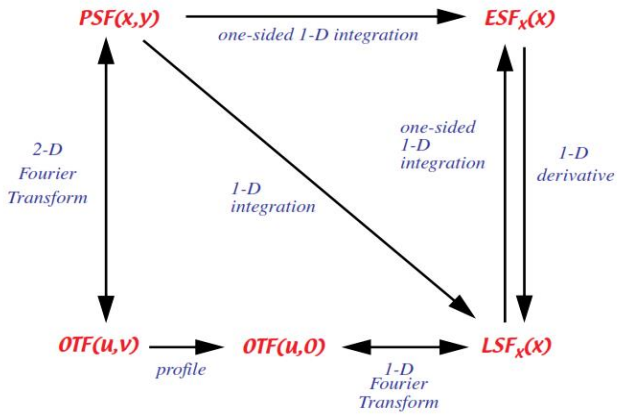


Figure 5.3-3- Relationship between PSF, ESF, LSF and OTF [44] .

5.3.3.Measurement Setup

Build the setup as shown in Figure 5.3-4, mount the calibrated lens on motor stage such that center of lens is focused at the center of slanted edge and use motor stage to change the distance to get the best focused image. Place the object (slanted edge) between sensor and the light source. The setup should be mounted on a rigid optical bench or breadboard in the dark cabinet. Typical list of equipment required for MTF measurements are:

1. DUT.
2. Reference detector (Hamamatsu Si reference diode).
3. Power supply (CI-0033 TTi_QL355TP_Power_supply).
4. Light source.
5. Slanted edge object (razor blade).
6. Flex TC.
7. Calibrated Camera Lens (F = 2.8 and OBJ = 0.3m).
8. Camera link.
9. Thorlabs motor stage.

5.3.3.1.Software Tools

1. Iron Python-based Caeleste software environment.

5.3.3.2.Block Diagram

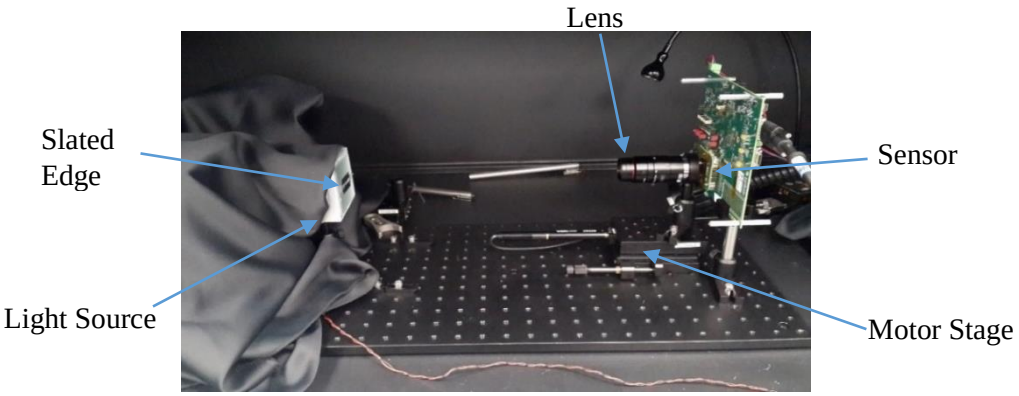


Figure 5.3-4- Setup for MTF measurement inside dark chamber.

5.3.4.Measurement Procedure

5.3.4.1.Algorithm

First correct the images for non-uniformities using flat-field correction. Then find the transition point (position of the edge) i.e. position of the pixels where pixel value crosses 50% of the maximum intensity and determine the response for every transition. Now shift the response for every line on the slanted edge image over each other to get the oversampled transition curve i.e. the ESF. Further LSF is calculated by taking the derivative of the ESF. And finally MTF is calculated as modulus of Fast Fourier Transform of LSF at Nyquist frequency. If needed various mathematical operations can be performed like- linear polyfit on transition position value for small pixel size, interpolation on oversampled ESF and filtering for smoothing of LSF curve.

5.3.4.2.Procedure

1. Build the setup as described in measurement setup to capture the images and place the edge (Target) at distance (OBJ) of 30cm from the DUT.
2. Make sure that captured image does not saturate by setting appropriate light source intensity (50% saturation).
3. Set the correct Lens setting (F = 2.8 and OBJ = 0.3m).
4. Capture three images under same camera setting and environment condition for different distance to get best focused image:
    a. Take a dark image as your reference.
    b. Take a light image as your second reference.
    c. Take an image of the slanted edge (target).
5. Correct the images for dark non-uniformity and for non-uniformities in light of your pixel response using flat-field correction.

5.3.4.3.Pay attention

1. Make sure that the captured images do not saturate.
2. The slanted edge should be within 1º-10º angle as MTF depends on edge angle.
3. Make sure that you have correct lens setting.

5.3.5.Accuracy of method

The thing that can go wrong is the misalignment while removing target (slanted edge) and capturing reference light and dark images. Also stability of light source is a must.

5.3.6.Data Processing

It is the main part of the procedure. To calculate MTF of the sensor perform the following steps:

Note- Following calculation are for vertical edge and for horizontal edge interchange column with row.

5.3.6.1.Correcting image

For the three images (dark, edge, light) taken at every distance perform flat-field correction as following:

1. Both EDGE and LIGHT are corrected for their offset and dark non-uniformities by subtracting DARK reference image, which leads to normalize images with black = 0 and white = 1.

2. The obtained correction (LIGHT-DARK) will be used to create a gain map for each pixel, called correction factor (gain factor).
3. Hence obtained correct image will be (EDGE – DARK) / (LIGHT-DARK).
4. And finally linearly normalize the correct image.

5.3.6.2.Calculate ESF

First determine the pixel position for which pixel value is just > 0.5 and then determine 50% transition point for all the lines (rows) of the edge from following formula -

$$\mathrm{Col_{intercept}} = \mathrm{Threshold_{col}} + \frac{(\mathrm{col_0} - 0.5)}{(\mathrm{col_1} - \mathrm{col_0})} \tag{5.3-1}$$

Where,
$Col_{intercept}$: 50% transition point,
$Threshold_{col}$: column position where pixel value is > 0.5,
col_0: pixel value at threshold column for that line (row),
col_1: pixel value at (threshold column -1) for that line (row).

Note: It is better to take linear poly-fit value for $Col_{intercept}$ if the pixel size is small.

Now obtain ESF plot by determining intensity value for all the transition pixels and taking 10 (can be any number) pixels on either side of it. The data for ESF is highly oversampled and erratic, so should perform linear interpolation over ESF data.

5.3.6.3.Calculate LSF

LSF is the derivative of ESF. So one could employ Three-point derivative method to determine LSF-

$$\mathrm{LSF(x)} = \frac{\mathrm{d}}{\mathrm{dx}}\mathrm{ESF(x)} \tag{5.3-2}$$

$$\mathrm{LSF_i} = \frac{1}{2}\left(\frac{\mathrm{ESF_{i+1}} - \mathrm{ESF_i}}{\mathrm{col_{i+1}} - \mathrm{col_i}} - \frac{\mathrm{ESF_i} - \mathrm{ESF_{i-1}}}{\mathrm{col_i} - \mathrm{col_{i-1}}}\right) \tag{5.3-3}$$

Note: You can apply smoothing filter (Savitzky–Golay filter) for LSF data smoothing. A good setting for the filter can be order (m) = 2nd and window size (nr + nl + 1) = 51.

5.3.6.4.Calculate MTF

MTF is calculate by performing FFT over LSF and taking the modulus of the result.

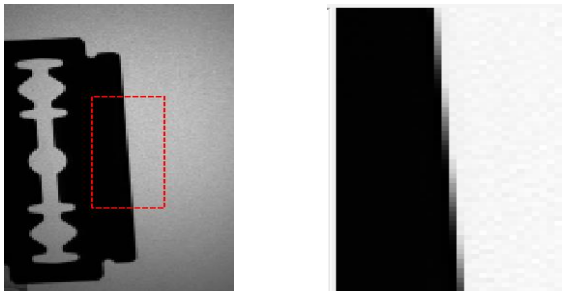
$$\text{At Nyquist frequency,} \quad \mathrm{MTF} = |\mathrm{FFT(LSF)}| \tag{5.3-4}$$

Nyquist frequency is fastest signal that can be reliably sampled, hence Nyquist frequency = 2×pixel pitch.
Also, Nyquist frequency can be determined by calculating perfect MTF from a perfect LSF. A perfect LSF is an ideal impulse of finite width and Nyquist frequency is the frequency at first zero.

This MTF consist of MTF of the lens as well. So the final sensor MTF normalized from 0 to 1 is given as-

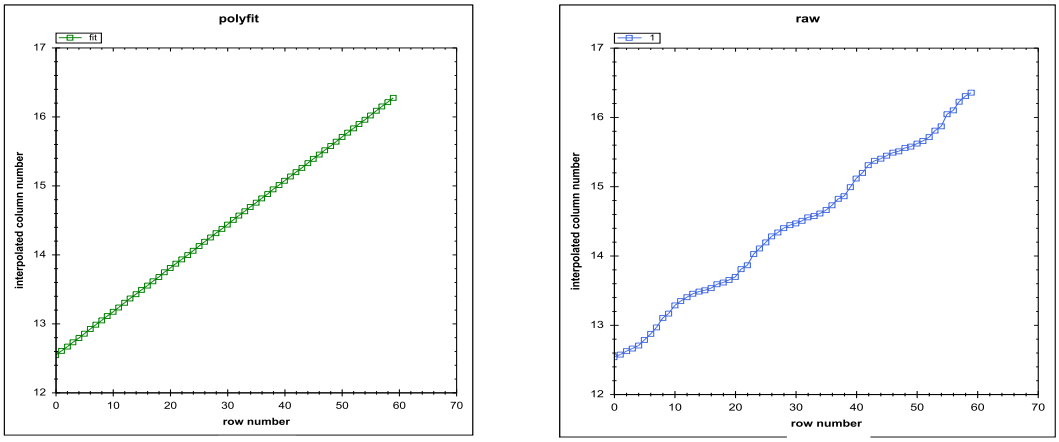
$$\mathrm{MTF_{sensor}} = \frac{\mathrm{MTF_{measured}}}{\mathrm{MTF_{lens}}} \tag{5.3-5}$$

5.3.7.Graphs and Figures



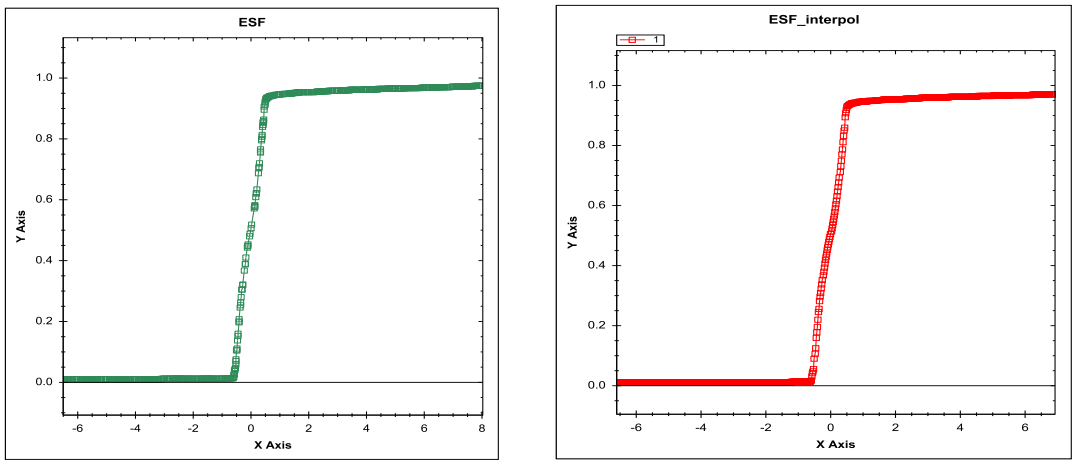
(a) (b)

Figure 5.3-5- (a) Target edge with ROI and (b) Flat-Filed corrected image for computing ESF.



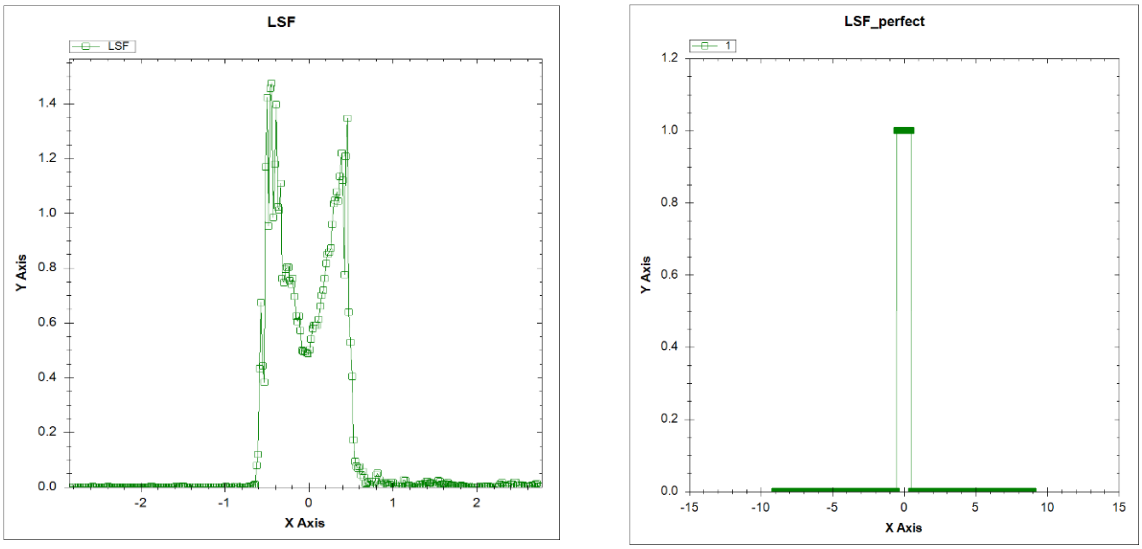
(a) (b)

Figure 5.3-6- Graph between interpolated column number and row number (a) Raw data of column values and (b) Linear polyfit data of column values.



(a) (b)

Figure 5.3-7- Edge spread function of the edge image on y-axis normalized signal and on x-axis pixel number, (a) Using raw data and (b) Interpolated data for pixel number.



(a) (b)

Figure 5.3-8- Line spread function of the corrected edge image, (a) LSF from actual data and (b) LSF of perfect edge.

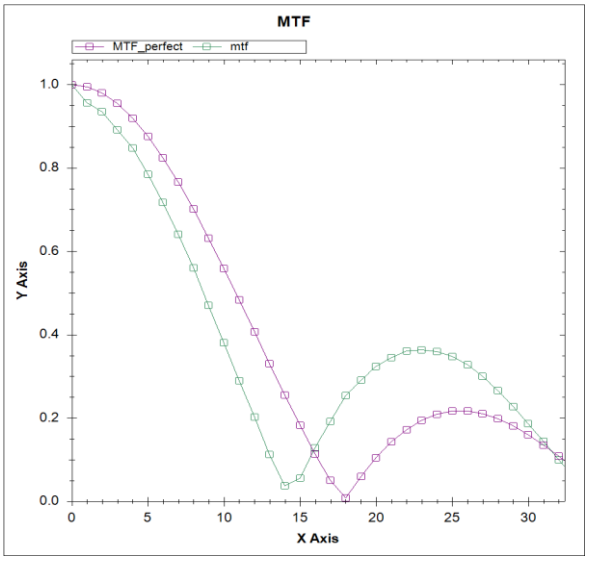


Figure 5.3-9- MTF of the corrected the edge image along with the perfect MTF obtained from perfect LSF.

5.3.8. Three-point derivative method

The derivative of a function is defined as-

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{5.3-6}$$

While h is small enough, we can use a centred difference formula to approximate the derivative:

$$f'(x_i) \approx \frac{f(x_i+h) - f(x_i-h)}{2h} \tag{5.3-7}$$

In practice, Origin treats discrete data by the transform centred difference formula, and calculates the derivatives at point P_i by taking the average of the slopes between the point and its two closest neighbours.

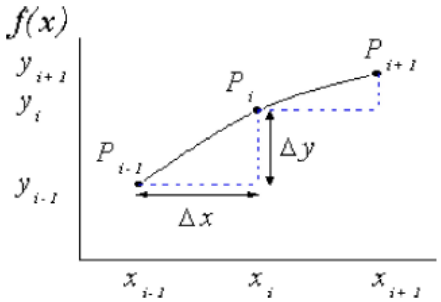


Figure 5.3-10- Three-point derivative method.

The derivative function applied to discrete data points can therefore be written as-

$$f'(x_i) = \frac{1}{2}\left(\frac{y_{i+1} - y_i}{x_{i+1} - x_i} - \frac{y_i - y_{i-1}}{x_i - x_{i-1}}\right) \tag{5.3-8}$$

5.3.9.Code Description
Class Modulation Transfer

Description
A class that measures and computes Modulation Transfer Function. This class computes the best focus position for the Lens and then capture images on and around the focus positon and then correct the images to compute the MTF of the sensor.

Function - position_array()
This function generates a list of all the position on which motor stage will move depending on the parameters passed.

position_array(mid_pos, range_pos, step_pos)

Parameters-
mid_pos <float> - start position in mm,
range_pos <float> - range of motor stage in mm,
step_pos <float> - step size of motor stage in mm.
Returns-
p_array <list> - List of all the positons of motor stage.

Function - focus_area()
This function captures an image and ask user select ROI of area to be focused. Coordinates of ROI are stored in vector which are later used to find focus.

Returns-
roi <vector> - coordinates of focus area.

Function - find_focus()
This function finds the best focus position for the Lens mounted on motor stage depending on the parameters passed, to capture image for computing MTF.

find_focus(self,range_, step)

Parameters-
range_ <float> - range of motor stage in mm,
step <float> - step size of motor stage in mm.
Returns-
focus_position <float> - best focus position.

Function - capture_image()
This function captures images according to specified experiment_type (dark, light and edge) at different positons and saves them in NAS directory corresponding to current project.

capture_image(self,range_, step)

Parameters-
range_ <float> - range of motor stage in mm,
step <float> - step size of motor stage in mm.

Function - correct_horizontal_image()
This function corrects the horizontal edge image by applying flat-field correction, take ROI of the images and normalize, rotate and flip if needed.

Returns-

img_edge <vector> - vector of all the images corrected.

Function - correct_vertical_image()
This function corrects the vertical edge by applying flat-field correction, take ROI of the images and normalize, rotate and flip if needed.

Returns-
img_edge <vector> - vector of all the images corrected.

Function - create_ESF()
Function creates ESF vector and generate ESF plot for all the images in img_edge, also generate plot for column intercept i.e. the transition position.

Returns-
ESF <vector> - 3D vector for intensity and interpolated column values for all the images.

Function- create_ESF()
Function creates LSF vector and generate LSF plot for all the images in img_edge. Also perform filtering for curve smoothing.

Returns-
LSF <vector> - 3D vector for intensity and pixel number for all the images,
LSF_filtered <vector> - 3D vector of filtered LSF values.

Function - create_MTF()
Function creates computes MTF of all the images in img_edge and generate MTF, MTF perfect plot.

Returns-
value <list> - List of MTF values of all the images at Nyquist frequency from LSF,
value_filt <list> - List of MTF values of all the images at Nyquist frequency from filtered LSF,
mtf <list> - List of values FFT of LSF.

5.3.10.Example

```
#creating instance for class
mtf = ModulationTransfer():
#Computes ROI coordinates for finding best focus image and return roi
mtf.focus_area();
#find position for best focused image starting at start_pos for range of 10mm and with step size of 0.1mm and return focus_position
mtf.find_focus(5,0.1);
#capture image starting at focus_position for range of 10mm and with step size of 0.1mm
mtf.capture_image(10,0.1);
#correct horizontal images and return images in vector image_edge
mtf.correct_horizontal_image();
#correct vertical images and return images in vector image_edge
mtf.correct_vertical_image();
#compute ESF from corrected images in image_edge and return ESF vector
mtf.create_ESF();
#compute LSF from ESF vector and return LSF and LSF_filtered vector
mtf.create_LSF();
#compute MTF from LSF vector and return MTF value at Nyquist frequency
mtf.create_MTF();
```

5.4.Read Noise

5.4.1.Objective

This procedure is to perform noise analysis of the sensor. Read Noise is the noise under zero illumination and it is invariant of temperature, integration time. It is result of various noise sources like- KTC noise, ADC noise, and temporal row noise and interference pattern.

5.4.2.Measurement Background

5.4.2.1.Method description

Read noise or temporal noise in dark is the inherent noise of a sensor and is equal to noise level at zero illumination. It is measured for 0ms integration time (t_{int}) to avoid DSNU and DUT is not illuminated otherwise photon shot noise is dominant.

5.4.3.Measurement setup

5.4.3.1.Block Diagram

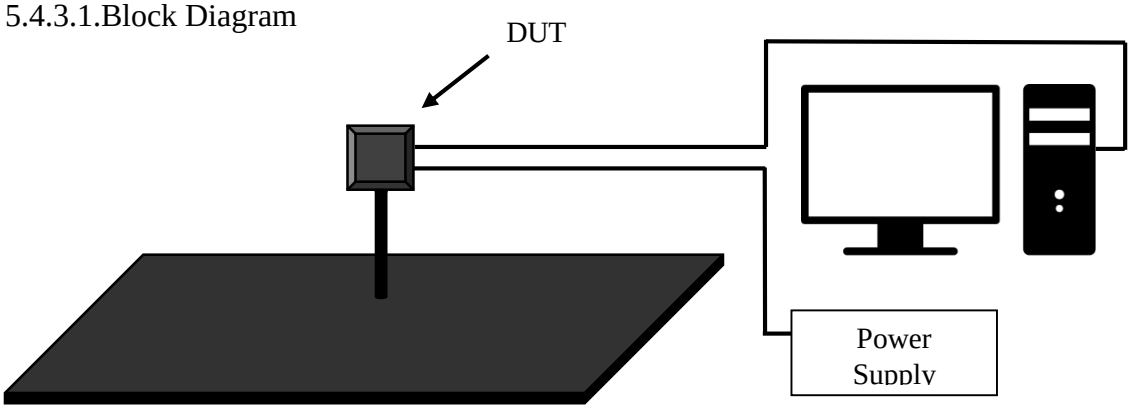


Figure 5.4-1-Setup for Noise measurement.

From the block diagram in Figure 5.4-1build the setup for the measurement. The camera link is used to capture image in dark. The setup should be mounted on a rigid optical bench or breadboard in the dark cabinet. Typical list of equipment employed for this measurements:

1. DUT.
2. Camera link.
3. Power supply (CI-0033 TTi_QL355TP_Power_supply).

5.4.3.2.Software Tools

1. Iron Python-based Caeleste software environment.

5.4.4.Measurement Procedure

5.4.4.1.Algorithm

The idea behind the measurement is to calculate read noise by computing standard deviation $(\sigma)_{rms}$ [Volts/DN] for each pixel over the series of images taken in dark for same t_{int}[s] and taking average of all the pixel's value to get noise over frame.

5.4.4.2.Procedure

1. Built the setup as shown in Figure 5.5-1, place the DUT and connect the peripheral equipment and turn ON the supply.

2. First capture some dummy images (e.g. 20, 30) for system to get stable and then capture certain number of images (e.g. 10, 20).

5.4.4.3.Pay attention

1. Capture as many image for accurate result and by taking their average one can remove all the offset and fixed pattern noise.

5.4.5.Data Processing

Read Noise is random variation in measurement value over a period of time, hence the noise is calculated in dark by evaluating temporal signal variance over series of frame for individual pixel and then by taking the average over all the pixels, which tells noise over image.

$$\text{Variance } (\sigma^2) = \frac{\sum_{j=1}^{K} \frac{\sum_{i=1}^{N} (P_{ij} - M)^2}{N}}{K} \quad (5.4\text{-}1)$$

$$\text{Noise}_{rms}\ [\text{Volts/DN}] = \sqrt{\sigma} \quad (5.4\text{-}2)$$

Where,
P_{ij} = j^{th} pixel value from i^{th} frame,
M = mean value of all the j^{th} Pixels of N frames,
N = Total number of frame acquired at specific illumination,
K = Total number of pixel.

Note: One can divide Noise in Volts by CVF to get Noise in electrons and above calculation can be performed on a row or a column to get row noise and column noise.

5.4.6.Accuracy of the method

This method is very accurate for determining Read noise. The two things that are important for accurate result are:

1. Measurement should be performed under perfectly dark condition and stable temperature so that there is no influence of dark current.
2. Integration time to capture image should be as short as possible so that the influence of DSNU is minimized.

5.4.7.Graphs and Figures

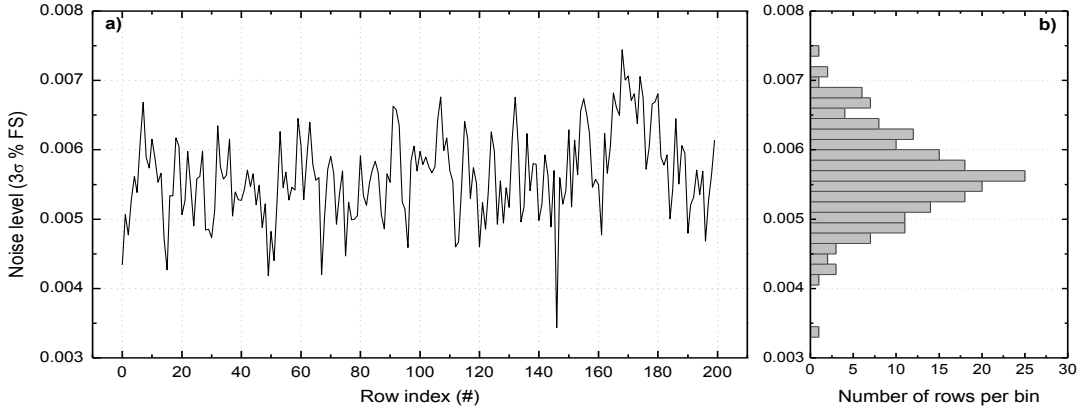


Figure 5.4-2- a) Noise per row and b) Its histogram.

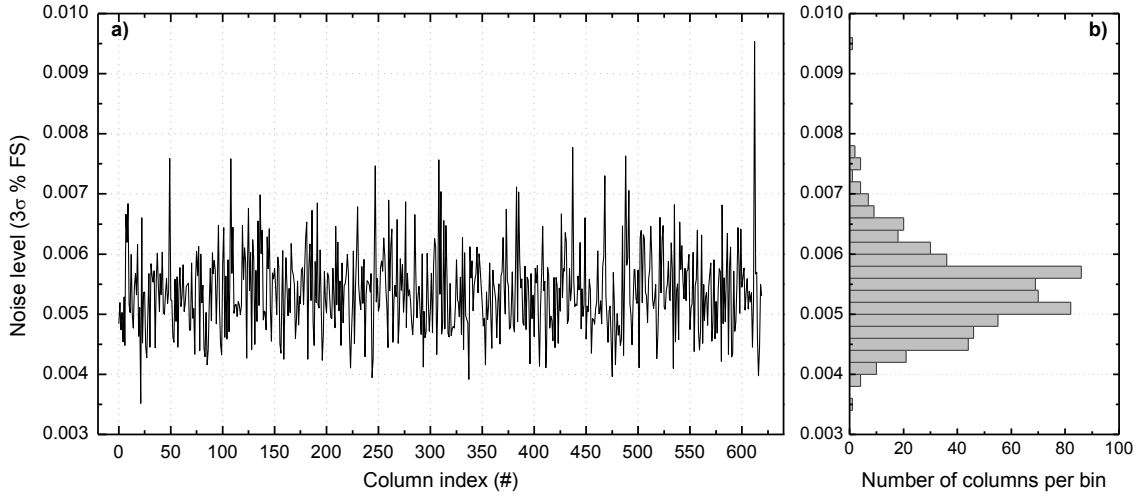


Figure 5.4-3- a) Noise per column and b) its histogram.

5.5.Dark signal and Dark signal non-uniformity

5.5.1.Objective

This procedure is to evaluate Dark signal i.e. dark current that flows through the sensor under no illumination and dark signal non uniformity (DSNU) is the spatial non-uniformity associated with the dark signal.

5.5.2.Measurement Background

5.5.2.1.Method description

Dark signal is a result of dark current of the photodiode and is dependent on temperature and integration time. DSNU are statistical variation on the generation/recombined level in each pixel i.e. the dark signal. Hence by measuring sensor output with respect to time under zero illumination dark signal and DSNU can be evaluated.

5.5.3.Measurement setup

5.5.3.1.Block Diagram

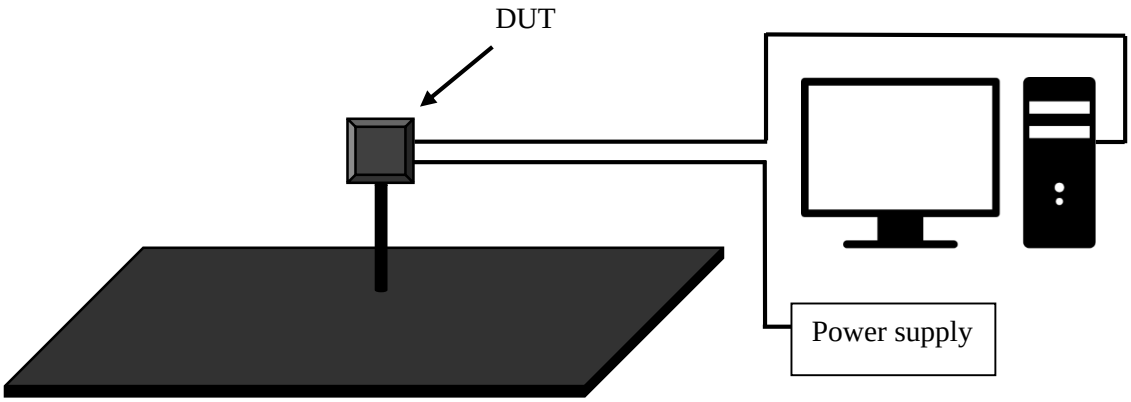


Figure 5.5-1- Setup for DS and DSNU measurement.

From the block diagram Figure 5.5-1 build the setup for the measurement. The camera link is used to capture image in dark. The setup should be mounted on a rigid optical bench or breadboard in the dark cabinet. Typical list of equipment employed for this measurements:

2. DUT.
3. Camera link.
4. Power supply (CI-0033 TTi_QL355TP_Power_supply).

5.5.3.2.Software Tools

1. Iron Python-based Caeleste software environment.

5.5.4.Measurement Procedure

5.5.4.1.Algorithm

The idea is to capture two images one with short t_{int1}[s] and one with long t_{int2}[s] and take their difference to compute dark response and then computing standard deviation $(\sigma)_{rms}$ to determine DSNU.

5.5.4.2.Procedure

1. Built the setup as shown in Figure 5.5-1, place the DUT and connect the peripheral equipment and turn ON the supply.
2. Capture Image 1 with very short t_{int1}.
3. Capture Image 2 with a long t_{int2}, at least long enough to see a significant dark current.
4. Capture a dummy Image 3 for even a longer integration time then t_{int2} just to check that Image 1 and Image 2 are in linear region.

5.5.5.Data Processing

Dark signal (DS) or dark count rate [e^-/sec] is the mean response of (Image 1- Image 2)/ (t_{int1} - t_{int2}). DSNU [V_{rms}] is the standard deviation of this response / (t_{int1} - t_{int2}).
Note: It is believed that I_{dark} must be measured with pixels SELECT off, to avoid hot carrier luminescence.

5.5.6.Accuracy of the method

The accuracy of this method is based on the assumption that the variation in pixel value over series of images is only noise, but there can be other variations, like delay in capturing subsequent images. Also dark signal and DSNU dependent on temperature so measurement should be done on temperature stabilized system.

5.5.7.Graphs and Figures

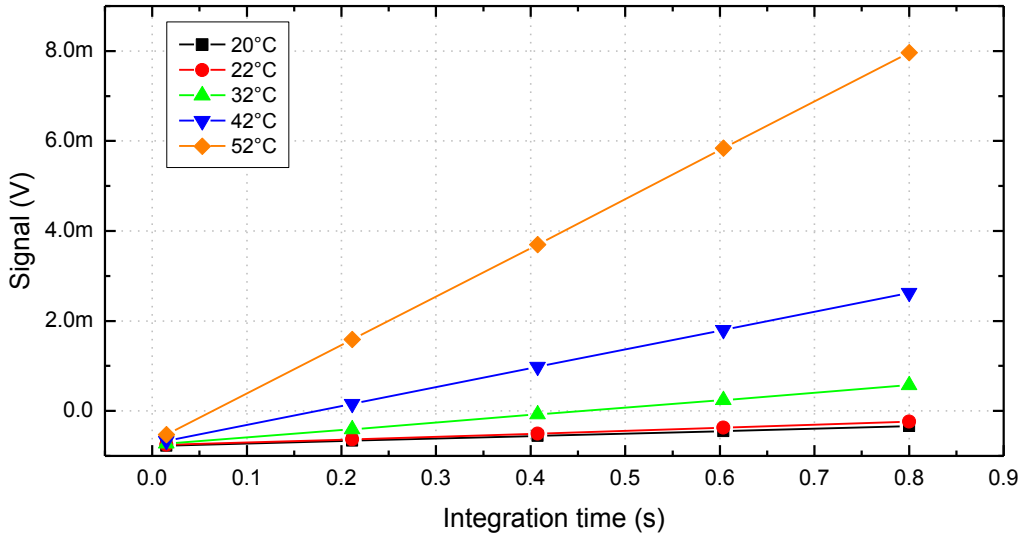


Figure 5.5-2-Signal level in function of Integration time setting at different sensor temperatures.

5.6.FPN and PRNU

5.6.1.Objective

This procedure is to perform non-uniformities analysis of the sensor. The statistical static variation of “offset” Fixed Pattern Noise (FPN) and “gain” Photo Response Non Uniformity (PRNU).

5.6.2.Measurement Background

5.6.2.1.Method description

Fixed pattern noise are the type of spatial non-uniformities within an image, this non uniformities are result of pixel to pixel variation. The two basic non uniformity are “offset” FPN which is also part of DSNU (Dark signal non uniformity) not dependent on temperature and integration time and “gain” PRNU (Photo response non uniformity) is non uniformity under illumination. Hence by observing pixel variation over a frame FPN and PRNU can be evaluated in dark and under illumination respectively.

5.6.3.Measurement setup

5.6.3.1.Block Diagram

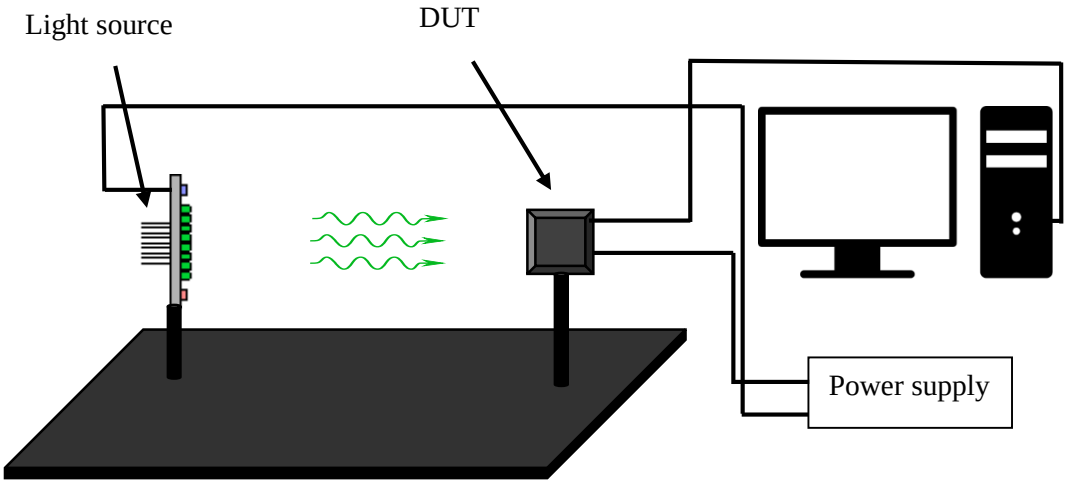


Figure 5.6-1- Setup for FPN and PRNU measurement.

From the block diagram in Figure 5.6-1 build the setup for the measurement. The camera link is used to capture images. The setup should be mounted on a rigid optical bench or breadboard in the dark cabinet. Typical list of equipment employed for this measurements:

5. DUT.
6. Camera link.
7. Power supply (CI-0033 TTi_QL355TP_Power_supply).
8. Caeleste LED light source.

5.6.3.2.Software Tools

1. Iron Python-based Caeleste software environment.

5.6.4.Measurement procedure

5.6.4.1.Algorithm

To calculate FPN and PRNU, one needs to capture certain number of images for three illumination settings dark, light and saturation. **FPN** is the variation in pixel dark signal over a frame and is determined by calculating STDEV of mean image obtained from set of frame taken in dark. **PRNU** as the name suggest is the non-uniformity in spatial variation of pixel value for specific illumination. It can be determined by calculating coefficient of variation of the mean image. We take mean of number of images in order to remove temporal noise. Images are taken for short t_{int} to nullify DSNU.

5.6.4.2.Procedure

1. Place the light source and DUT at an appropriate position on the optical bench such that with the given placement deep saturation of the DUT can be reached near maximum light source power.
2. Capture number of images (e.g. 10, 20) in dark.
3. Now power ON the light source and allow 10 minutes to get it thermally stable.
4. Capture number of images (e.g. 10, 20) for light (50% of saturation level) and for saturation.
5. Calculate mean of all the images captured for dark, light and saturation respectively to obtain resultant images for all three conditions.

Note- Capture images with short t_{int} to keep DSNU negligible.

5.6.5.Data processing

FPN is the static variation of offset in dark signal from pixel to pixel.

$$FPN\ [V_{rms}] = STDDEV\ (dark) \quad (5.6\text{-}1)$$

$$FPN\ [\%] = FPN\ [V_{rms}]/\ (sat- dark) \quad (5.6\text{-}2)$$

PRNU as the name suggest is the non-uniformity in spatial variation of pixel value for specific illumination (nominally at 50% of saturation level). It can be evaluated by calculating coefficient of variation of an image.

$$PRNU = CV = STDEV\ (light\text{-}dark) / (light\text{-}dark) \quad (5.6\text{-}3)$$

Where,
dark: resultant image by taking mean of all the images taken in dark.
light: resultant image by taking mean of all the images taken in light.
sat: resultant image by taking mean of all the images taken in saturation.
STDDEV: Standard deviation.
CV: Coefficient of variation.

5.7.Image Lag

5.7.1.Objective

It is the memory effect of the sensor due to the residual charge in pinned photodiode. The measurement result in [%] of signal level not reaching the final value.

5.7.2.Measurement Background

5.7.2.1.Method Description

The purpose of this test is to estimate the lag of the image sensor. Pinned photodiode can be modeled as lumped RC network, hence there is certain time constant (delay) for charge to deplete through the diode or the potential barrier and if the width of the pulse applied to transfer gate is smaller than the time constant (delay) then it will result in residual charge at photodiode therefore the lag in the sensor.

In a 4T pixel, this is due to the emptying time constant of charge that is limited by the potential barrier. Charge transfer from Pinned photodiode can mathematically corresponds to current through MOSFET in sub-threshold or the forward current through a diode. Hence it requires certain time constant to overcome the barrier and this emptying time constant can then be directly translated into “image lag”.

5.7.3.Measurement Setup

5.7.3.1.Block Diagram

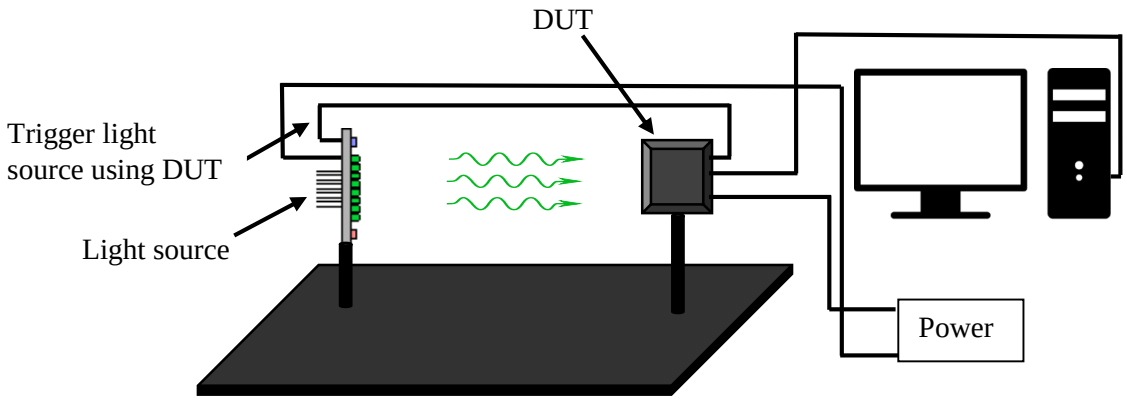


Figure 5.7-1- Setup for Image lag measurement.

5.7.3.2.List of Equipment

9. DUT.
10. Mounting rails.
11. Power supply (CI-0033 TTi_QL355TP_Power_supply).
12. Caeleste LED light source.
13. Connection cables.
14. Cameral ink.

5.7.3.3.Software Tools

1. Iron Python-based Caeleste software environment.

5.7.4.Measurement Procedure

5.7.4.1.Algorithm

The idea behind the measurement is to grab certain (e.g. 10) black frames followed by illuminated frames and vice versa by illuminating the sensor by a light pulse of same duration as t_{int} and synchronized with begin/end of the t_{int}. The frame-to-frame crosstalk (image lag) is then estimated as an average signal level in dark state expressed in percent of signal level in illuminated state.

5.7.4.2.Procedure

1. Built the setup as shown in Figure 5.7-1, place the light source and DUT and connect the peripheral equipment and turn ON the supply.
2. Adjust the illumination level (via current, distance, extra attenuators) so that the signal is (default, and if not, report this) at about 50% of saturation when the light is on. In some case it is of interest to do the measurement also at 10% or even less (1%) of saturation.
3. Generate the sequence using either waveform generator or SeqC and grab 10 frames in dark, followed by 10 illuminated frame.
4. Perform above step for illumination to dark situation.
5. For accurate triggering of light pulse, one should use a trigger pulse from the test board itself.

5.7.4.3.Accuracy of the method

There is hardly anything that can go wrong with this simple procedure, but still there can be some sources of error-

1. Generated sequence is not perfectly aligned with begin/end of the t_{int}.
2. Light source instability can cause delay in switching ON/OFF of light pulse.

5.7.5.Data Processing

Image lag is calculated by computing the decaying response from illumination to dark or vice versa.

$$\text{Lag}\,[\%] = \frac{\mu_{light} - \mu_{dark}}{\mu_{light}} \times 100 \qquad (5.7\text{-}1)$$

Where,
μ_{dark}: mean value of the first dark frame taken just after the light frame.
μ_{light}: mean value of the last light frame.

5.7.6.Graphs and Figures

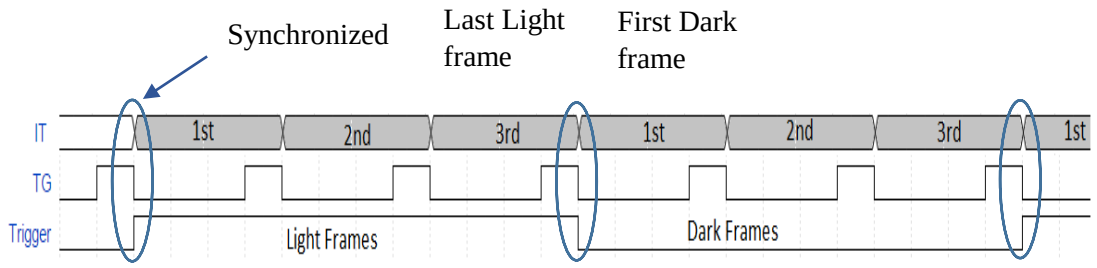


Figure 5.7-2- Timing diagram for Image Lag measurement.

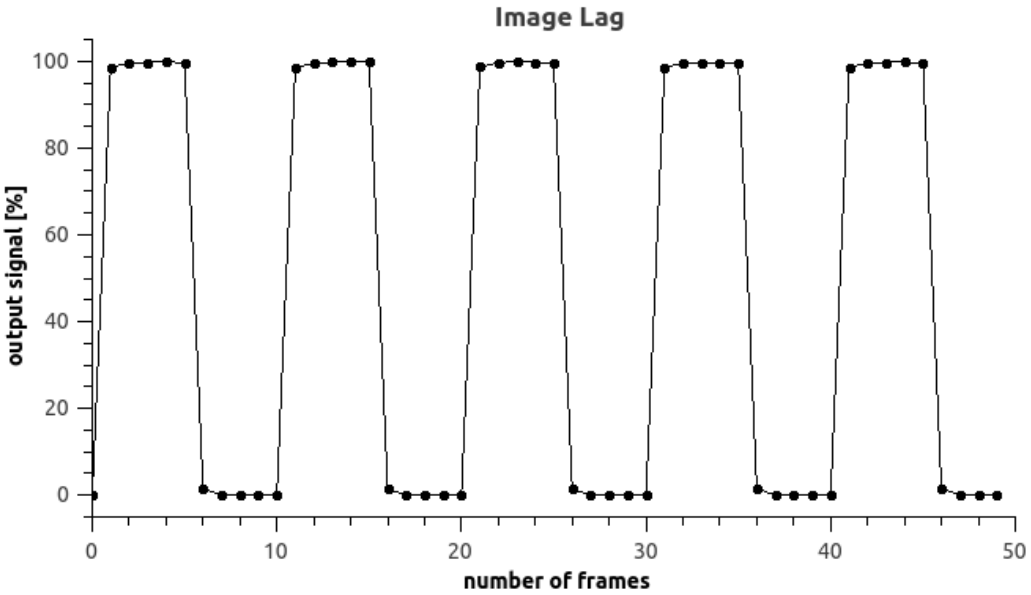


Figure 5.7-3- Image lag for 50 frames with 5 consecutive light and dark frames.

6. Chapter 6

6.1.Summary and Conclusions

This work presents complete standard procedure for characterizing CMOS image sensor. The main objective of this work is to better the measurement procedure, so after rigorous measurements and testing on different image sensor samples standard procedures are created. Also the software library to communicate with the test board and image sensor are updated and new hardware tools were incorporated for accurate and precise measurements. Here are some conclusions drawn on the basis of the measurements performed for characteristic parameters of the CMOS image sensor.

6.1.1. Conclusions

1. First and foremost, it is importance to have a stable and accurate light source, all the characteristic measurement includes light source except dark signal measurement. Therefore, it is important that light source emits light with constant intensity and uniformity over DUT.
2. The characterization of Caeleste LED light source shows that the light source is not very stable and uniform and with improvements like an optical feedback and PID controller may result in a better stability and uniformity of light source. The initial result of improved light source showed significant improvements in stability but complete characterization was not performed to draw firm conclusions.
3. Quantum efficiency is an important characteristic. For measuring quantum efficiency Caeleste uses dedicated test structure which are identical to those of the original sensor's pixel array in terms of photodiode, transfer gate and metallization properties. So this helps in performing accurate measurement as it eliminates large test boards and influence of any extra circuit. For example, QE test structures consist of guard pixel which prevents any charge leakage in the pixel from outside.
4. The other problem with QE measurement is its setup accuracy. As it is important to place the QE test structure exactly at the same location as of reference photodiode to receive same intensity of light for both. But from the measurement result shown in Figure 5.1-5. The current setup result in inaccuracy of 2.68% which is quite a lot, so to solve this problem new hardware tool are ordered for precise alignment of devices under test.
5. The other method to compute quantum efficiency is from the conversion gain value or charge to voltage factor as described in section 5.1.7.1.1 and this method seems less accurate as compared to QE test structure method, as it depends on the accuracy of conversion gain value.
6. Next characteristic measurement is photo response measurement of the image sensor which provides information on linearity, full well capacity and conversion gain value. The accuracy of this method entirely depends on light source stability and how accurately light intensity is measured using reference photodiode, as measurement are performed by changing the light intensity in a controlled way and computing image sensor response.
7. From the two methods to compute conversion gain mentioned in section 5.2.6.5.1 and section 5.2.6.5.2, the photo response curve method seems more accurate as mean-variance method is only accurate for perfectly linear image sensors. So photo response curve is preferred for the CVF measurement at Caeleste. The mean-variance method gives more accurate result for linear image sensors.

8. The important thing to take into account while computing conversion gain from mean-variance method is the higher the number of frames taken to calculate variance of the output signal better is the accuracy of the result. As accuracy of conversion gain depends on the number of samples taken [].
9. Next characteristic measurement is modulation transfer function which is also measure of optical crosstalk. There are different methods to compute MTF of which slanted edge method is more accurate and fast method.
10. This thesis work draws some conclusion about data processing, as MTF measurement involves lots of mathematical computation. The first and foremost is to get oversampled data for edge spread function. The measurements clearly show more the number of sampling point higher is the accuracy of generated ESF.
11. The other interesting thing about data processing while computing edge location; for large pixel size it better to take raw sampled data and for small pixel size it is better to apply linear polyfit on sampled data. It is because for small pixel size there are chances that sampled signal is a result of noise and not the actual data. This phenomenon can be seen in Figure 5.3-6.
12. Similarly, for the edge spread function data, there is a need for interpolation. The oversampled data obtained for the edge spread function is very erratic and inconsistent, so before computing line spread function the oversampled data need to be interpolated first.
13. A generalized conclusion can be drawn for noise measurement; it is important to know which type of noise is being measured. While measuring read noise, which is the inherent noise of the image sensor under no illumination, it should be independent of dark noise and should be measured on temperature stable system. Similarly, while measuring PRNU, the integration time for grabbing a frame should be as small as possible to eliminate DSNU.
14. The other important thing while measuring the signal level is that for accurate measurements the signal level should be significant with respect to noise level.
15. Image lag is not much significant in CMOS image sensor compare to CCD's, but with continuous increase in high speed and large full charge capacity requirements image lag can be a problem in future.

6.2.References

[1] G. P. Weckler, "Operation of p-n junction photodetectors in a photon flux integrating mode," *IEEE Journal of Solid-State Circuits,* vol. 2, pp. 65-73, 1967.

[2] G. Weckler and R. Dyck, "Integrated arrays of silicon photodetectors for image sensing," *IEEE Trans. Electron Devices,* Vols. ED-15, pp. 196-201, 1968.

[3] P. Noble, "Self-scanned silicon image detector arrays," *IEEE Trans. Electron Devices,* Vols. ED-15, pp. 202-209, 1968.

[4] D. Renshaw, P. B. Denyer, G. Wang and M. Lu, "ASIC image sensors," *IEEE International Symposium on Circuits and Systems,* vol. 4, pp. 3038-3041, 1990.

[5] R. Nixon, "128x128 CMOS Photodiode-Type Active Pixel Sensor with On-Chip Timing, Control and Signal Chain Electronics," *Proceeding of SPIE,* vol. 2415, pp. 117-123, 1995.

[6] A. Krymski, D. V. Blerkom, A. Andersson, N. Block, B. Mansoorian and E. R. Fossum, "“A high speed, 500 frames/s, 1024 × 1024 CMOS active pixel sensor," *Symp. VLSI Circuits,* p. 137–138, 1999.

[7] E. R. Fossum Fellow IEEE and D. B. Hondongwa Student Member IEEE, "A Review of the Pinned Photodiode for CCD and CMOS Image Sensors," *IEEE Journal of the Electron Devices Society,* vol. 2, no. 3, 2014.

[8] W. F. Kosonocky and J. E. Carnes, "Basic concepts of charge-coupled devices," vol. 36, p. 566–593, 1975.

[9] T. Yamada, H. Okano and N. Suzuki, "The evaluation of buriedchannel layer in BCCD’s," *IEEE Trans. Electron Devices,* vol. 25, p. 544–546, 1978.

[10] Y. XU, "Fundamental Characteristics of a Pinned Photodiode CMOS Pixel," 2015.

[11] A. Gamal El and H. Eltoukhy, "CMOS Image sensor," *IEEE CIRCUITS & DEVICES MAGAZINE,* MAY/JUNE 2005.

[12] S. Wuu, H. Chien, D. Yaung, C. Tseng, C. Wang, C. Chang and H. Y.K., "A high performance active pixel sensor with 0.18-μm CMOS color imager technology," *IEEE IEDM Tech. Dig.,* p. 555–558, 2001.

[13] T. Inoue, S. Takeuchi and S. Kawahito, "A CMOS Active Pixel Image Sensor with In-pixel CDS for High-Speed Cameras," *Proceedings of SPIE - The International Society for Optical Engineering (Proceedings of SPIE),* Vols. 2 5301A-32, pp. 1-8, 2004.

[14] R. M. Guidash, T. -H. Lee, P. P. K. Lee, D. H. Sackett, C. I. Drowley, M. S. Swenson, L. Arbaugh, R. Hollstein, F. Shapiro and S. Domer, "A 0.6 /spl mu/m CMOS pinned photodiode color imager technology," *Electron Devices Meeting, 1997. IEDM '97. Technical Digest., International,* pp. 927 - 929, 1997.

[15] K. Yonemoto, H. Sumi, R. Suzuki and T. Ueno, "A CMOS image sensor with a FPN-reduction technology and a hole accumulated diode," *Dig. Tech. Papers, ISSCC,* pp. 102-103, 2000.

[16] I. Inoue, H. Nozaki, H. Yamashita, T. Yamaguchi, H. Ishiwata, H. Ihara, R. Miyagawa, H. Miura, N. Nakamura, E. Y. and M. Y., "New LV-BPD(Low Voltage Buried Photo-Diode) for CMOS Imager," *IEDM, Technical Digest,* 1999.

[17] S. K. Mendis, "CMOS active pixel image sensors for highly integrated imaging systems," *IEEE Journal of Solid-State Circuits,* vol. 32, pp. 187-197, 1997.

[18] S. Kawahito, S. Suh, T. Shirei, S. Itoh and S. Aoyama, "Noise Reduction Effects of Column-Parallel Correlated Multiple Sampling and Source-Follower Driving Current Switching for CMOS Image Sensors," 2009.

[19] X. Ge, "The design of a global shutter cmos image sensor in 110nm technology," 2012.

[20] J. Nakamura, in *Image Sensors and Signal Processing for Digital Still Cameras*, 2005, pp. 79-91.

[21] A. Darmont, "Spectral Response of Silicon Image Sensors," Aphesa (www.aphesa.com), 2009.

[22] "Sensitivity of CCD cameras–some key factors to consider," [Online]. Available: http://www.andor.com/learning-academy/sensitivity-of-ccd-cameras-key-factors-to-consider.

[23] M. Green and M. Keevers, "Optical properties of intrinsic silicon at 300 K. Progress in Photovoltaics," vol. 3, pp. 189-192, 1995.

[24] "Photop Technologies, Inc.," 2012. [Online]. Available: http://www.photoptech.com/main/products_gx/Etalon.php.

[25] S. Kar, "MOSFET: Basics, Characteristics, and Characterization," in *High Permittivity Gate Dielectric Materials*, 2013, pp. 47-152.

[26] S. Shafie, S. Kawahito, I. A. Halin and W. Z. W. Hasan, "Non-Linearity in Wide Dynamic Range CMOS Image Sensors Utilizing a Partial Charge Transfer Technique," *Sensors (14248220)*, vol. 9, no. 12, pp. 9452-9467, 2009.

[27] D. Gardner, "Characterizing digital cameras with the photon transfer curve," *Summit imaging.*

[28] J. Janesick, "Scientific CCDs," in *Optical Engineering*, 1987, pp. 692-714.

[29] P. Magnan and M. Estribeau, "Fast MTF measurement of CMOS imagers using ISO 12233 slanted edge methodology," *SPIE Proceedings*, vol. 5251, 2004.

[30] E. Buhr, S. Guenther-Kohfahl and U. Neitzel, "Simple method for modulation transfer function determination of digital imaging detectors from edge images," in *Proc. SPIE 5030, Medical Imaging 2003: Physics of Medical Imaging*, 2003.

[31] O. Corporation, "How to Measure MTF and other Properties of Lenses," 1999. [Online]. Available: http://www.optikos.com/wp-content/uploads/2013/11/How-to-Measure-MTF.pdf.

[32] D. Vany and S. Arthur, Master Optical Techniques (Wiley Series in Pure and Applied Optics), 1981.

[33] H. Tian, "Noise Analysis in CMOS image sensor," 2000.

[34] B. Razavi, Design of Analog CMOS Integrated Circuits, Boston: McGraw Hill, 2001.

[35] A. Scholten and D. Klaassen, "New 1/f noise model in MOS Model 9, level 903," 1998.

[36] K. K. Hung, P. K. Ko, H. C. and Y. C. Cheng, "A unified model for the flicker noise in metal-oxide-semiconductor field-effect transistors," *IEEE Transactions on Electron Devices*, vol. 37, no. 3, 1990.

[37] N. Loukianova al. et, "Leakage Current Modeling of Test Structures for Characterization of Dark Current in CMOS Image Sensors," *IEEE Transactions on Electron Devices,* vol. 50, pp. 77-83, Jan. 2003.

[38] S. Sze, Semiconductor Devices. Physics and Technology, New York: John Wiley & Sons, 2001.

[39] D. A. Neamen, in *Semiconductor Physics and Devices Basic Principles*, The McGraw-Hill Companies, 2012, p. 227.

[40] X. Wang, "Noise in Sub-Micron CMOS Image Sensors," 2008.

[41] E. R. Fossum and D. B. Hondongwa, "A Review of the Pinned Photodiode for CCD and CMOS Image Sensors".

[42] M. Sargent, "National Physics Laboratory," 2011. [Online]. Available: http://www.kayelaby.npl.co.uk/introduction_to_quality_assurance_of_measurements/8_3/8_3.html.

[43] Bentham, "TMc300 Single Monochromator," [Online]. Available: http://www.bentham.co.uk/tmc300.htm.

[44] D. R. A. Schowengerdt, "Image Science and Engineering," 2000.

7. Chapter 7

Appendix

7.1.Python Code for Filtering

```
__author__ = 'utsav'
from TSSF import *
''' Filtering module '''

def savgol_filter(nl, nr, m, data):
    """savgol_filter(no. of points in left, no. of points in right, order, data)
        nl <int> = 'no. of point in left'
        nr <int> = 'no. of point in right'
        m <int> = '2 or 4' #order
            The order of the polynomial used to fit the samples.
            `m` must be less than `nl + nr + 1'.
        data <vector> = data you want to filter
        returns filter data in form of a vector

        np = window_length
            The length of the filter window (i.e. the number of coefficients).
            `window_length` must be an odd positive integer.
        ld(derivative) = 0 for smoothing optional
            The order of the derivative to compute.  This must be a
            nonnegative integer.  The default is 0, which means to filter
            the data without differentiating.
        check if all the given inputs are valid
        ld = 0 , np = nr + nl + 1 , m < nr + nl
        first modify the vector data by adding zero in start and end just to match size then
        generates filter data by performing convolution between data and filter coefficients
    """
    filter_data = Vector()
    #creating filter coefficients
    coeff = coefficients(nl , nr, m);
    i = 0
    j = 0
    filter_data = Vector(data.TotalSize)
    average = 0
    convolution = []
    modify_data = Vector(data.TotalSize + (coeff.TotalSize-1))
    while (i < data.TotalSize):
            #add zero in starting
        for j in range(0,(coeff.TotalSize-1)/2):
            modify_data[j] = 0
            #actual data
        for j in range((coeff.TotalSize-1)/2,(modify_data.TotalSize-          (coeff.TotalSize-
1)/2)):
            modify_data[j] = data[i]
            i = i + 1
            #add zero at the end
```

```
        for j in range((modify_data.TotalSize-(coeff.TotalSize-
            1)/2),modify_data.TotalSize):
            modify_data[j] = 0

    #performing convolution
    for i in range((modify_data.TotalSize - (coeff.TotalSize-1))):
        average = 0
        for j in range(coeff.TotalSize):
            average = average  + coeff[j]*modify_data[i+j]
        convolution.append(average)
    for i in range(len(convolution)):
        filter_data[i] = convolution[i]
    return filter_data

def coefficients(nl , nr, m):
    """ coefficients (no. of point in left, no. of point in right, order)
        nl = 'no. of point in left'
        nr = 'no. of point in right'
        m = '2 or 4' #order: int
            The order of the polynomial used to fit the samples.
            `m` must be less than `window_length`.
        returns coefficients in form of a vector
        np = window_length : int
            The length of the filter window (i.e. the number of
             coefficients).
            `window_length` must be an odd positive integer.
        ld(derivative) = 0 for smoothing optional
            The order of the derivative to compute.  This must be a
            nonnegative integer. The default is 0, which means to filter the data without
             differentiating.
        check if all the given inputs are valid
        ld = 0 , np = nr + nl + 1 , m < nr + nl
    """
    np = nr + nl + 1
    a = Vector(m+1 , m+1)
    d = 1
    #The order of the derivative to compute
    ld = 0
    indx = Vector(m+1)
    b = Vector(m+1)
    n = m+1
    #filter coefficients
    coeff = Vector(np)
    #this loop performs least square method
    for ipj in range(m*2+1):
        if (ipj != 0):
            sum = 0
        else:
            sum = 1
        for k in range(1, nr+1, 1):
            sum = sum + pow(k,ipj)
        for k in range(1, nl+1, 1):
```

```
            sum = sum + pow(-k, ipj)
        mm = min(ipj , (2*m -ipj))
        for imj in range(-mm, mm+1, 2):
            a[(ipj+imj)/2, (ipj-imj)/2] = sum
    #linear equation solution, LU decomposition
    ludcmp(a, m+1, indx, d);
    for j in range(m):
        b[j] = 0.0
        b[ld] = 1.0
    #linear equation solution, back substitution
    lubksb(a, m+1, indx, b);
    for kk in range(np):
        coeff[kk] = 0.0
    #generates filter coefficients
    for k in range(-nl, nr+1, 1):
        sum = b[0]
        fac = 1.0
        for mm in range(m):
            fac *= k
            sum = sum + (b[mm+1]*fac)
        coeff[k+nr] = sum
    return coeff

def ludcmp(a, n, indx, d):
    """linear equation solution, LU decomposition"""
    vv = Vector(1, n)
    Tiny = 1.0e-20
    for i in range(n):
        big = 0.0
        for j in range(n):
            temp = abs(a[i, j])
            if (temp > big):
                big = temp
        if (big == 0.0):
            print 'error'
        vv[i] = 1.0/big
    for j in range(n):
        for i in range(j):
            sum = a[i, j]
            for k in range(i-1):
                sum = sum - a[i, k]*a[k, j]
            a[i, j] = sum
        big = 0.0
        for i in range(j, n, 1):
            sum = a[i, j]
            for k in range(j):
                sum = sum - a[i, k]*a[k, j]
            a[i,j] = sum
            dum = vv[i]*abs(sum)
            if (dum >= big):
                big = dum
                imax = i
```

```
        if (j!= imax):
            for k in range(n):
                dum = a[imax, k]
                a[imax, k] = a[j, k]
                a[j, k] = dum
            d = -(d)
            vv[imax] = vv[j]
        indx[j] = imax
        if (a[j, j] == 0.0):
            a[j, j] = Tiny
        if (j != n):
            dum = 1.0/a[j, j]
            for i in range(j+1, n, 1):
                a[i,j ] *= dum
    return a

def lubksb(a, n, indx, b):
    """linear equation solution, backsubstitution"""
    ii=0
    for i in range(1, n+1, 1):
        ip = indx[i-1]
        sum = b[int(ip)]
        b[int(ip)] = b[i-1]
        if (ii != 0):
            for j in range(ii-1, i-1, 1):
                sum = sum - a[i-1, j]*b[j]
        else:
            if (sum != 0):
                ii = i
        b[i-1] = sum
    for i in range(n-1, -1, -1):
        sum = b[i]
        for j in range(i+1, n, 1):
            sum -= a[i,j ]*b[j]
        b[i] = sum/a[i, i]
    return b
```

7.2.Python Code for MTF measurement

```
"Author- Utsav"
# slanted edge
import os
import sys
import System
import inspect;
import TSSF;
from Toolbox.Utilities import DirectorySwap
from Toolbox.Utilities import filtering
from time import sleep;
from time import time;
from TSSF import UIUtil
import datetime
from TSSF.UIUtil import Plot, PlotRefresh, Ask  # import ironplot as ip
from TSSF import Vector
from TSSF import ImageLab
import glob
import clr, System
import shutil
import random;
from Toolbox import Equipment;
import Toolbox;
from Caeleste.Instruments.Thorlabs import TDC001;

clr.AddReference("System.Windows.Forms");
clr.AddReference("Caeleste.Instruments.Thorlabs");

def position_array(mid_pos, range_pos, step_pos):
    p_array = [mid_pos - range_pos];
    while p_array[-1] <= (mid_pos + range_pos):
        p_array.append(p_array[-1] + step_pos);
    return p_array

class ModulationTransfer():
    """
    Routines needed to perform a modulation transfer measurements
    """
    def __init__(self, intensity=1e-6, tolerance=0.25):
        #Procedure data
        self.img_edge = "";
        self.col_st = 0;
        self.col_end = 0;
        self.row_st = 0;
        self.row_end = 0;
        self._diff = "" ;
        self.focus_position = 0;
        self.start_pos = 2.5;                    #initial position of motor stage
        self.roi = Vector(4);
        self.numpoints = 1000 ;
```

```
    # here we find the product data...
    soft_dir = os.path.join(os.environ['PROJECTDIR'], 'python')
    sys.path.append(soft_dir)
    try:
        from ChipProperties import product;
        self.chip = product();
        self.chip.load_testboard();
        self.project_dir = self.chip.sensorname;
        print "Sensor class initialized. Chip mode is %s" % self.chip.mode;
    except:
        self.chip = '';
        self.project_dir = 'unknown';
        print "Could not load the chip properties. Exiting"
        return;
    self.test_dir = "Modulation Transfer Function";
    self.NAS_dir = os.path.join("\\\\192.168.2.40", "Public", "BulkData");
    self.output_dir = os.path.join(self.NAS_dir, self.project_dir, self.test_dir);
    #Folder for current experiment
    self.image_dir = "";

    # stage initialization
    try:
        self.stagex = TDC001.Connect('83847325');
        print "ThorLabs stage is initialized successfully";
    except:
        print "ThorLabs stage is not initialized!";

    # saving desired light intensity and tolerance settings
    self.lint = intensity;
    self.tol = tolerance;

    # Initializing the equipment
    try:
        self.glsc = Equipment.GLSC();
        print "GLSC initialized successfully. Calling its calibration functions";
        self.glsc.InitInstrument();
        self.glsc.Calibrate();
    except:
        print "Could not initialize the GLSC";
    # ready
    print "MTF method ready";

def focus_area(self):
    """
    place the lens at appropriate position using motor stage so that image is more
    or less focused and position is defined by self.start_pos during class initialization
    """
    self.stagex.MoveAbsolute(self.start_pos);
    self.glsc.set_intensity(False, self.lint, self.tol);
    self.chip.capture_image();
    handle = "Focus Select"
    ImageLab.Plot(self.chip.image[:,:,1], handle);
```

```
    select = Ask("Please select ROI and type ok")
    if select == 'ok':
        area = ImageLab.GetSession(handle)["ROI1"].Area;
        self.roi[0] = area.Left;
        self.roi[1] = self.roi[0] + area.Width;
        self.roi[2] = area.Top;
        self.roi[3] = self.roi[2] + area.Height;
    else:
        print 'Please select ROI'
    return self.roi

def find_focus(self, range_, step):
    """
    Use the thorlabs stages to define the best focal distance.
    Place image with slanted edge to define the focal distance
    select a roi. With median calculate the max stddev
    Input:
        range_  <float> - scanning range in mm
        step    <float> - scanning step in mm
    Return: self.focus_position:
    """
    # grab image
    # get median image
    # select ROI
    # start focus
    # raw check finds max stddev
    self._diff = 0;
    self.focus_position = 0;
    # create position array
    position = position_array(self.start_pos, range_, step);
    # raw measurement
    for pos in position:
        self.stagex.MoveAbsolute(pos);
        print "stage position %4.3f [mm]" % pos;
        self.glsc.set_intensity(False, self.lint, self.tol);
        self.chip.capture_image();
        dummy1 = self.chip.image[:,:,0];
        dummy2 = self.chip.image[:,:,1];
        cds = dummy2 - dummy1;
        difference =
         (cds[int(self.roi[0]):int(self.roi[1]),int(self.roi[2]):int(self.roi[3])].MedianFilter(5) -
         cds[int(self.roi[0]):int(self.roi[1]),int(self.roi[2]):int(self.roi[3])]).stddev();
        if difference > self._diff:
            self.focus_position = pos
            self._diff = difference
    # fine tuning
    position = position_array(self.focus_position, 0.1, 0.05);
    for pos in position:
        self.stagex.MoveAbsolute(pos);
        print "stage position %4.3f [mm]" % pos;
        self.glsc.set_intensity(False, self.lint, self.tol);
        self.chip.capture_image();
```

```
        dummy1 = self.chip.image[:, :, 0];
        dummy2 = self.chip.image[:, :, 1];
        cds = dummy2 - dummy1;
        difference =
         (cds[int(self.roi[0]):int(self.roi[1]),int(self.roi[2]):int(self.roi[3])].MedianFilter(5) -
         cds[int(self.roi[0]):int(self.roi[1]),int(self.roi[2]):int(self.roi[3])]).stddev();
        if difference > self._diff:
            self.focus_position = pos;
            self._diff = difference;
    self.stagex.MoveAbsolute(self.focus_position);
    return self.focus_position;
    # save position data

def capture_images(self, range_, step):
    """
        Scans the distance with a thor The MTF method class.
        You have to make sure this will not go beyond
        Input:
            self.focus_position <float> - current position in mm
            range_ <float> - scanning range in mm
            step    <float> - scanning step in mm
    """
    # initializing the test board
    if self.chip.test_board==None:
        self.chip.load_testboard();
        self.chip.capture_image();
    else:
        self.chip.capture_image();
    if self.chip.test_board==None:
        print "AcquireImages(): Could not initialize the test system!";
        return;

    # Configuring folders for saving the data
    devpartnum = Ask("Enter the device part number please");
    image_type = Ask("Enter the image type (edge, dark or light)");
    experiment_type = Ask("Enter the edge direction (horizontal or vertical)");

    # setting the light off if the image type is 'dark'
    if image_type=='dark':
        self.glsc.light_off();
    devtemp = Ask("Enter the device temperature");

    # folder for current experiment
    self.image_dir = os.path.join(self.output_dir, "PartNum_%s" % devpartnum,
                                                    experiment_type, image_type);
    DirectorySwap(path_to_directory=os.path.join(self.output_dir, "PartNum_%s" %
                            devpartnum, experiment_type), directoryName=image_type);
    os.mkdir(os. path.join(self.image_dir, 'CDSed'));
    pos = [];
    pos = position_array(self.focus_position, range_, step);
    numsteps = len(pos);
    print "scan range from %.2f to %.2f mm with step of %.2f" % (pos[0], pos[-1], step);
```

```
        # allocating the memory;
        reset = Vector(self.chip.width, self.chip.height, self.chip.test_board.no_frames);
        signal = Vector(self.chip.width, self.chip.height, self.chip.test_board.no_frames);
        imgcdsed = Vector(self.chip.width, self.chip.height, self.chip.test_board.no_frames);

        # Running the loop to move motor stage and capture image
        for i in range(numsteps):
            print "Setting new stage position at %.2fmm" % pos[i];
            self.stagex.MoveAbsolute(pos[i]);
            if not(image_type=='dark'):
                # setting the intensity
                self.glsc.set_intensity(False, self.lint, self.tol);
            self.chip.capture_image(); # acquiring the images
            print "Capturing images...";
            # averaging to get rid of temporal noise
            for k in range(self.chip.test_board.no_frames):
                reset[:, :, k] = self.chip.image[:, :, k*2];
                signal[:, :, k] = self.chip.image[:, :, k*2+1];
                imgcdsed[:, :, k] = signal-reset;

            # saving the data
            tmp = reset.mean(2);
            file_name = 'PartNum_%s_%s_%s_reset_%.2fmm.png' % (devpartnum,
experiment_type, image_type, pos[i]);
            tmp.Save(os.path.join(self.image_dir, file_name));
            tmp = signal.mean(2);
            file_name = 'PartNum_%s_%s_%s_signal_%.2fmm.png' % (devpartnum,
experiment_type, image_type, pos[i]);
            tmp.Save(os.path.join(self.image_dir, file_name));
            tmp = imgcdsed.mean(2);
            file_name = 'PartNum_%s_%s_%s_%.2fmm.png' % (devpartnum, experiment_type,
image_type, pos[i]);
            tmp.Save(os.path.join(self.image_dir, 'CDSed', file_name));
        print "Scanning is finished, light source set off";
        self.glsc.light_off();
        print "Setting initial stage position (at %.1fmm)" % self.focus_position;
        self.stagex.MoveAbsolute(self.focus_position);

        # Preparing the log file
        ts = time();
        date_today = datetime.datetime.fromtimestamp(ts).strftime('%Y/%m/%d %H:%M ')
        header = "%s project %s measurement report\n" % (self.chip.sensorname, 'Modulation
Transfer Function');
        header += "Date %s\n" % date_today
        header += "Integration time setting %fs\n" % (float(self.chip.integration_time/1e7));
        header += "Sensor part reference (entered by user): %s\n" % devpartnum;
        header += "Sensor temperature (entered by user) is %sC degrees\n" % devtemp;
        header += "Sensor operating mode: %s\n" % self.chip.mode;
        header += "Sensor gain (capacitor) setting: %i\n" %
                    self.chip.seqc_parameters['gain_value'];
```

```
    #vcmsrc = Ask("Type common-mode voltage source please");
    vcmsrc = 'provided by testboard ADC';
    header += "Common mode voltage source: %s\n" % vcmsrc;
    header += "Number of frames per illumination point %i\n" %
              self.chip.test_board.no_frames;
    header += "Light source intensity setting = %eW/cm2, tolerance = %.2f%%\n" %
              (self.lint, self.tol);

    # adding the data to header
    header += "Distance range from %.2fmm to %.2fmm with step of %.2fmm\n" %
              (pos[0], pos[-1], step);
    header += "Number of steps in this experiment is %i\n" % numsteps;
    header += "Image type (edge, dark, light) is %s\n" % image_type;
    header += "Edge direction (if specified) %s\n" % experiment_type;

    ts = time();
    st = datetime.datetime.fromtimestamp(ts).strftime('%Y%m%d_%H%M_');
    # make sure a timestamp is included in filename
    filename = "MTF_PartNum=%s__%s.txt" % (devpartnum, st);
    f = open(os.path.join(self.image_dir, filename), 'w')
    f.writelines(header);
    f.close();

def correct_horizontal_image(self):
    """
    take ROI of the images and normalize, rotate and flip if needed
    """
    devpartnum = Ask("Enter the device part number please");
    image_type = 'edge';
    experiment_type = 'horizontal';
    # folder for current experiment
    self.image_dir = os.path.join(self.output_dir, "PartNum_%s" % devpartnum,
    experiment_type);
    path1 = os.path.join(self.image_dir, image_type, 'CDSed');
    os.chdir(path1);
    # set ROI
    # load edge image
    # load all the images for different position
    images = glob.glob1(os.getcwd(), "PartNum_%s_horizontal_edge*" % devpartnum);
    dummy_image = Vector.Load(self.images[0]);
    handle = "ROI Select"
    ImageLab.Plot(dummy_image, handle);
    cont = Ask("Please select ROI of at least 21 rows and type ok");
    if cont == 'ok':
        #keep in mind to change ROI number
        area = ImageLab.GetSession(handle)["ROI1"].Area ;
        self.col_st = area.Left;
        self.col_end = self.col_st + area.Width;
        self.row_st = area.Top;
        self.row_end = self.row_st + area.Height;
        dummy = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
        #transition = "check if transition from black to white yes or no?"
```

```
        if dummy[0,0] > dummy[0,(self.row_end - self.row_st)-1] :
            transition = 'no';
        else:
            transition = 'yes';
        # clear previous data
        self.img_edge = " " ;
        for image in images:
            suf_x = image.split('_');
            res = Vector.Load(os.path.join(self.image_dir, 'edge',
          "PartNum_%s_%s_edge_reset_%s" % (devpartnum, experiment_type, suf_x[4])));
            sig = Vector.Load(os.path.join(self.image_dir, 'edge',
          "PartNum_%s_%s_edge_signal_%s" % (devpartnum, experiment_type, suf_x[4])));
            dummy_image = sig – res;
            edge = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
            res = Vector.Load(os.path.join(self.image_dir, 'dark',
          "PartNum_%s_%s_dark_reset_%s" % (devpartnum, experiment_type, suf_x[4])));
            sig = Vector.Load(os.path.join(self.image_dir, 'dark',
          "PartNum_%s_%s_dark_signal_%s" % (devpartnum, experiment_type, suf_x[4])));
            dummy_image = sig – res;
            dark = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
            res = Vector.Load(os.path.join(self.image_dir, 'light',
          "PartNum_%s_%s_light_reset_%s" % (devpartnum, experiment_type, suf_x[4])));
            sig = Vector.Load(os.path.join(self.image_dir, 'light',
          "PartNum_%s_%s_light_signal_%s" % (devpartnum, experiment_type, suf_x[4])));
            dummy_image = sig – res;
            light = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
            edge_correct = (edge - dark)/(light-dark);
            edge_image = (edge_correct - edge_correct.min())/(edge_correct.max()-
                          edge_correct.min());
            if transition == 'no':
                if self.img_edge:
                    edge_image = edge_image.Rotate90CW();
                    self.img_edge = Vector(self.img_edge.Append(edge_image));
                else:
                    edge_image = edge_image.Rotate90CW();
                    self.img_edge = edge_image;
            if transition == 'yes':
                if self.img_edge:
                    edge_image = edge_image.Rotate90CW();
                    edge_image = edge_image.FlipV();
                    self.img_edge = Vector(self.img_edge.Append(edge_image));
                else:
                    edge_image = edge_image.Rotate90CW();
                    edge_image = edge_image.FlipV();
                    self.img_edge = edge_image;
    else:
        print 'ROI not selected'
    return self.img_edge

def correct_vertical_image(self):
    """
    take ROI of the images and normalize, rotate and flip if needed
```

```
"""
devpartnum = Ask("Enter the device part number please");
image_type = 'edge';
experiment_type = 'vertical'
 # folder for current experiment
self.image_dir = os.path.join(self.output_dir, "PartNum_%s" % devpartnum,
experiment_type);
path1 = os.path.join(self.image_dir, image_type ,'CDSed');
os.chdir(path1);
# set ROI
# load edge image
# load all the images for different position
self.images = glob.glob1(os.getcwd(), "PartNum_%s_vertical_edge*" % devpartnum)  ;
dummy_image = Vector.Load(self.images[0]);
handle = "ROI Select"
ImageLab.Plot(dummy_image, handle);
cont = Ask("Please select ROI of atleast 21 column and type ok");
if cont == 'ok':
    #keep in mind to change ROI number
    area = ImageLab.GetSession(handle)["ROI2"].Area;
    self.col_st = area.Left;
    self.col_end = self.col_st + area.Width;
    self.row_st = area.Top;
    self.row_end = self.row_st + area.Height;
    dummy = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
    #transition = "Is transition from black to white yes or no?"
    if dummy[0,0] < dummy[(self.col_end - self.col_st)-1,0] :
        transition = 'yes';
    else:
        transition = 'no';
    # clear previous data
    self.img_edge = " ";
    for image in self.images:
        suf_x = image.split('_');
        res = Vector.Load(os.path.join(self.image_dir, 'edge',
     "PartNum_%s_%s_edge_reset_%s" % (devpartnum, experiment_type, suf_x[4])));
        sig = Vector.Load(os.path.join(self.image_dir, 'edge',
     "PartNum_%s_%s_edge_signal_%s" % (devpartnum, experiment_type, suf_x[4])));
        dummy_image = sig – res;
        edge = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
        res = Vector.Load(os.path.join(self.image_dir, 'dark',
     "PartNum_%s_%s_dark_reset_%s" % (devpartnum, experiment_type, suf_x[4])));
        sig = Vector.Load(os.path.join(self.image_dir, 'dark',
     "PartNum_%s_%s_dark_signal_%s" % (devpartnum, experiment_type, suf_x[4])));
        dummy_image = sig – res;
        dark = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
        res = Vector.Load(os.path.join(self.image_dir, 'light',
     "PartNum_%s_%s_light_reset_%s" % (devpartnum, experiment_type, suf_x[4])));
        sig = Vector.Load(os.path.join(self.image_dir, 'light',
     "PartNum_%s_%s_light_signal_%s" % (devpartnum, experiment_type, suf_x[4])));
        dummy_image = sig – res;
        light = dummy_image[self.col_st:self.col_end, self.row_st:self.row_end];
```

```
            edge_image = (edge - dark)/(light-dark);
            edge_image = (edge_image - edge_image.min())/(edge_image.max()-
                          edge_image.min());
            if transition == 'yes':
               if self.img_edge:
                  self.img_edge = Vector(self.img_edge.Append(edge_image));
               else:
                  self.img_edge = edge_image;
            if transition == 'no':
               if self.img_edge:
                  edge_image = edge_image.FlipV();
                  self.img_edge = Vector(self.img_edge.Append(edge_image));
               else:
                  edge_image = edge_image.FlipV();
                  self.img_edge = edge_image;
      else:
         print 'ROI not selected'
      return self.img_edge;

   def create_ESF(self):
      """
      Calculate the ESF of the image
      return ESF vector:
      """
      # choose what kind of intercepted column values you want
      data = Ask("polyfit(for small pixel) or raw(for large pixel)");
      if data == 'polyfit':
         self.numpoints = 1000;
         self.ESF = Vector(2, self.numpoints, len(self.images));
         for n in range(len(self.images)):
            image = self.img_edge[:, :, n];
            #determining transition and threshold value
            self.col_interp = Vector(image.Height);
            self.polyval = Vector(image.Height);
            y_list = Vector(image.Height);
            x_list = Vector(image.Height);
            for row in range(0, image.Height):
               col = 0
               while image[col, row] < 0.5:                    #check for transition
                  col = col + 1
               threshold_col = col
               col_0=image[threshold_col, row];
               col_1=image[threshold_col-1, row];
               #transition position
               self.col_interp[row]=((threshold_col) +((col_0-0.5)/(col_1-col_0)));
               x_list[row] = row;
               y_list[row] = self.col_interp[row];
            #Polyfit transition position according to row number
            #perform 1st order polyfit
            poly = y_list.PolyFit(x_list, order = 1);
            for i in range(0,x_list.TotalSize):
               self.polyval[i] = poly[1]*x_list[i] + poly[0];
```

```
UIUtil.Plot(x_list,self.polyval,str(n),'polyfit');
plothandle = UIUtil.Plot('polyfit');
plothandle.XAxis.Title.Text = 'row number';
plothandle.YAxis.Title.Text = 'interpolated column number';
#2nd way by taking polyval
#21 pixels for each row
col_integer = Vector(21*image.Height);
col_exact = [];
col_sort = Vector(21*image.Height);
intensity_sort = Vector(21*image.Height);
row = 0;
j = 0;
k = 0;
for row in range(image.Height):
   for j in range(-10,11,1):
      col_integer[k] = self.polyval.Floor()[row] + j;
      exact = col_integer[k] - self.polyval[row];
      col_exact.append(exact);
      k = k + 1;
col_exact.sort();
j = 0;
k = 0;
row = 0;
intensity = [];
for row in range(image.Height):
   for j in range(0, 21):
      intensity.append(image[int(col_integer[k]), row]);
      k = k + 1;
   row = row + 1;
intensity.sort();
for i in range(21*image.Height):
   col_sort[i] = col_exact[i];
   intensity_sort[i] = intensity[i];
UIUtil.Plot(col_sort,intensity_sort, str(n), 'ESF');
#perform interpolation
i = 0;
j = 0;
k = 0;
col_start = math.ceil(col_sort.min()) + 1;
col_end = math.floor(col_sort.max());
step = ((col_end-col_start)/(float(self.numpoints-1)));
self.col_interpol = Vector.FromRange(col_start, step, self.numpoints);
intensity_interpol = Vector(self.col_interpol.TotalSize);
polyval2 = [];
for i in range(self.col_interpol.TotalSize):
   k = 0;
   polyval2 = [];
   dummy_x = Vector(2);
   dummy_y = Vector(2);
   for k in range(col_sort.TotalSize-1):
      if ((self.col_interpol[i] >= col_sort[k]) and (self.col_interpol[i] <=
         col_sort[k+1])):
```

```
                for j in range(2):
                    dummy_y[j] = intensity_sort[k+j];
                    dummy_x[j] = col_sort[k+j];
                    poly2 = dummy_y.PolyFit(dummy_x, order = 1);
                polyval2.append(poly2[1]*self.col_interpol[i] + poly2[0]);
                intensity_interpol[i] = polyval2[0];
        for i in range(self.col_interpol.TotalSize):
            self.ESF[0, i, n] = self.col_interpol[i];
            self.ESF[1, i, n] = intensity_interpol[i];
        UIUtil.Plot(self.col_interpol,intensity_interpol, str(n), 'ESF_interpol');
else:
    self.numpoints = 1000;
    self.ESF = Vector(2,self.numpoints,len(self.images));
    for n in range(len(self.images)):
        image = self.img_edge[:, :, n];
        #determining transition and threshold value
        self.col_interp = Vector(image.Height);
        self.polyval = Vector(image.Height);
        y_list = Vector(image.Height);
        x_list = Vector(image.Height);
        for row in range(0, image.Height):
            col = 0;
            #check for transition
            while image[col, row] < 0.5:
                col = col + 1;
            threshold_col = col;
            col_0=image[threshold_col,row];
            col_1=image[threshold_col-1,row];
            #transition position
            self.col_interp[row]=((threshold_col)+((col_0-0.5)/(col_1-col_0))) ;
            x_list[row] = row;
            y_list[row] = self.col_interp[row];
        #Polyfit transition position according to row number
        #perform 1st order polyfit
        poly = y_list.PolyFit(x_list, order = 1) ;
        for i in range(0, x_list.TotalSize):
            self.polyval[i] = poly[1]*x_list[i] + poly[0];
        UIUtil.Plot(x_list,self.col_interp,str(n),'raw');
        plothandle = UIUtil.Plot('raw');
        plothandle.XAxis.Title.Text = 'row number';
        plothandle.YAxis.Title.Text = 'interpolated column number';

        #2nd way by taking polyval
        col_integer = Vector(21*image.Height); #21 pixels for each row
        col_exact = [];
        col_sort = Vector(21*image.Height);
        intensity_sort = Vector(21*image.Height);
        row = 0;
        j = 0;
        k = 0;
        for row in range(image.Height):
            for j in range(-10, 11, 1):
```

```
            col_integer[k] = self.col_interp.Floor()[row] + j;
            exact = col_integer[k] - self.col_interp[row];
            col_exact.append(exact);
            k = k + 1;
    col_exact.sort();
    j = 0;
    k = 0;
    row = 0;
    intensity = [];
    for row in range(image.Height):
        for j in range(0,21):
            intensity.append(image[int(col_integer[k]),row]);
            k = k + 1;
        row = row + 1;
    intensity.sort();
    for i in range(21*image.Height):
        col_sort[i] = col_exact[i];
        intensity_sort[i] = intensity[i];
    UIUtil.Plot(col_sort,intensity_sort, str(n), 'ESF');

    #perform interpolation
    i = 0;
    j = 0;
    k = 0;
    col_start = math.ceil(col_sort.min()) + 1;
    col_end = math.floor(col_sort.max());
    step = ((col_end-col_start)/(float(self.numpoints-1)));
    self.col_interpol = Vector.FromRange(col_start, step, self.numpoints);
    intensity_interpol = Vector(self.col_interpol.TotalSize);
    polyval2 = [];
    for i in range(self.col_interpol.TotalSize):
        k = 0;
        polyval2 = [];
        dummy_x = Vector(2);
        dummy_y = Vector(2);
        for k in range(col_sort.TotalSize-1):
            if ((self.col_interpol[i] >= col_sort[k]) and (self.col_interpol[i] <=
                col_sort[k+1])):
                for j in range(2):
                    dummy_y[j] = intensity_sort[k+j];
                    dummy_x[j] = col_sort[k+j];
                poly2 = dummy_y.PolyFit(dummy_x, order = 1);
                polyval2.append(poly2[1]*self.col_interpol[i] + poly2[0]);
                intensity_interpol[i] = polyval2[0];
    for i in range(self.col_interpol.TotalSize):
        self.ESF[0, i, n] = self.col_interpol[i];
        self.ESF[1, i, n] = intensity_interpol[i];
    UIUtil.Plot(self.col_interpol,intensity_interpol, str(n), 'ESF_interpol');
plothandle = UIUtil.Plot('ESF');
plothandle.XAxis.Title.Text = 'pixel number';
plothandle.YAxis.Title.Text = 'intensity';
plothandle = UIUtil.Plot('ESF_interpol');
```

```
        plothandle.XAxis.Title.Text = 'pixel number';
        plothandle.YAxis.Title.Text = 'ESF_interpolated';
    return self.ESF;

def create_LSF(self):
    """
    Calculate the LSF of the image from the ESF
    return LSF vector:
    """
    self.LSF_filtered = Vector(2, self.numpoints, len(self.images));
    self.LSF = Vector(2, self.numpoints, len(self.images));
    for n in range(len(self.images)):
        for i in range(1,self.col_interpol.TotalSize-1):
            self.LSF[1, i, n] = 0.5*(((self.ESF[1, i+1, n] - self.ESF[1,i,n])/(self.ESF[0,i+1,n] -
            self.ESF[0, i, n])) + ((self.ESF[1,i,n] - self.ESF[1,i-1,n])/(self.ESF[0,i,n] -
             self.ESF[0, i-1, n])));
        UIUtil.Plot(self.col_interpol,self.LSF[1, :, n], str(n), 'LSF');
        self.LSF_filtered[1, :, n] = filtering.savgol_filter(25, 25, 2, self.LSF[1, :, n]);
         #perform filtering on LSF data using filter coefficient from savgol_filter
        UIUtil.Plot(self.col_interpol,self.LSF_filtered[1, :, n], str(n), 'LSF_filt');
    return self.LSF;
    return self.LSF_filtered;

def create_MTF(self):
    """
    Calculate the MTF of the image from the LSF
    """
    LSF_perfect = Vector(self.col_interpol.TotalSize);
    for i in range(self.col_interpol.TotalSize):
        if ((self.col_interpol[i] > -0.5) and (self.col_interpol[i] < 0.5)):
            LSF_perfect[i] = 1;
    UIUtil.Plot(self.col_interpol, LSF_perfect, '1', 'LSF_perfect');
    mtf_perfect = LSF_perfect.DFT().norm(0);
    mtf_perfect = mtf_perfect/mtf_perfect.max();
    UIUtil.Plot(mtf_perfect, 'MTF_perfect', 'MTF');
    UIUtil.Plot(mtf_perfect, 'MTF_perfect', 'MTF_filt');
    self.value = [];
    self.value_filt = [];
    for n in range(len(self.images)):
        mtf = self.LSF[1, :, n].DFT().norm(0);
        mtf = mtf/mtf.max();
        self.value.append(mtf[9]);          #9 is Nyquist frequency obtained from perfect MTF
        UIUtil.Plot(mtf, str(n), 'MTF');
    for n in range(len(self.images)):
        mtf_filt = self.LSF_filtered[1, :, n].DFT().norm(0);
        mtf_filt = mtf_filt/mtf_filt.max();
        self.value_filt.append(mtf_filt[9]) #9 is Nyquist frequency obtained from perfect MTF
        UIUtil.Plot(mtf_filt, str(n), 'MTF_filt');
    return self.value;
    return mtf;
```

7.3. List of Acronym

ADC	Analog-to-Digital Conversion
APS	Active Pixel Sensor
BSI	Back side illuminated
CCD	Charge-coupled Device
CDS	Correlated Double Sampling
CG	Conversion gain
CVF	Charge to Voltage factor
CMOS	Complementary Metal-Oxide-Semiconductor
DR	Dynamic Range
DSNU	Dark Signal Non-Uniformity
FD	Floating Diffusion
FPN	Fixed Pattern Noise
FSI	Front side illuminated
FWC	Full Well Capacity
GSE	Global shutter efficiency
HDR	High Dynamic Range
MTF	Modulation Transfer Function
NL	Non-Linearity
PID	Proportional Integral Derivative
PPD	Pinned Photodiode
PPS	Passive Pixel Sensor
PRNU	Photo Response Non-Uniformity
PSD	Power Spectral Density
PWL	Piecewise Linear
QE	Quantum efficiency
SNR	Signal to Noise Ratio
SPI	Serial Peripheral Interface
STI	Shallow Trench Isolation
S/H	Sample and Hold