Digital processing for large pixel array dToF systems

Jia Shi

Delft University of Technology

Digital processing for large pixel array dToF systems

Master's Thesis

To fulfill the requirements for the degree of
Master of Science Electrical Engineering, Track: Microelectronics
at Delft University of Technology under the supervision of
Georgi Gaydadjiev (Quantum & Computer Engineering, University of TUDelft)

Jia Shi (5218845)

May 23, 2022

Contents

List of Acronyms

AR	Augmented Reality
ToF	Time-of-Flight
dToF	direct Time-of-Flight
SPAD	Single-Photon Avalanche Diode
RTL	Register-Transfer Level
SRAM	Static Random Access Memory
ASIC	Application-Specific Integrated Circuit
FOV	Field-Of-View
iToF	indirect Time-of-Flight
SNR	Signal-to-Noise Ratio
FWHM	Full Width at Half-Maximum
PiFH	Partitioned inter-Frame Histogram
FiFH	Folded inter-Frame Histogram
SiFH	Shifted inter-Frame Histogram
CH	Coarse Histogram
FH	Fine Histogram
PDE	Photon Detection Efficiency
CIE	International Commission on Illumination
AFOV	Angle of Field-Of-View
DCR	Dark Count Rate
PD	Peak Detection
PH	Partial Histogramming
IC	Integrated Circuit
IP	Intellectual Property
DOE	Diffraction Optics Element

Acknowledgments

This thesis could not finish without much support from various people. Georgi Gaydadjiev, as my thesis advisor, I really appreciate your guidance and patience for the last nine months. For me, a good teacher is a person who truly thinks from the student's side, and you are definitely a spectacular teacher. You truly cared about my thesis and future, and I will not forget that. Further, I'd like to thank kezhengMa, who taught me to think like an IC designer and helped me with his knowledge on the subject. I appreciated his help and his opinions on the subject.

I want to thank Ting Gong for his patience and supportive supervision of the dToF system, also his witty working attitude. Also, AoBa, BingXue, and other people helped me from the company side.

Thanks to GuliuxinJin, RanKong, and other friends, who listened to me and supported me.

Last, a big shout out to my parents, MingliangShi and YukunZhao, as my guide on the road of life, the companion in life forever. You are such wonderful parents who give me infinite love and support me in everything.

Abstract

Over the last decade, the recognition of the potential value of augmented reality (AR) and other human-machine interfaces has been growing. These applications are all based on depth sensing technologies. Among various depth sensing technologies, the Time-of-Flight (ToF) approach is emerging as a widely applicable method because it has the potential of reaching much longer distances at higher speed and accuracy, is natively suitable for mobile phones or AR.

One obvious problem is that the dToF system mainly targets the automotive industry or 3D imaging in close range, which means different power, accuracy, and fewer area requirements than mobile phone or AR applications. The application of the dToF system in the mobile phone or AR industry needs to be tested. The data volume is another problem of dToF. Usually, in the read out part, the dToF time-to-digital converter (TDC)'s timestamps will all be saved into a histogram. The peak value of the histogram is the detected result. However, in a large pixels scenario, the area cost will be too much for mobile phone or AR applications if the whole histogram for all pixels is saved. Hence, an algorithm that can save histogram partially or find out the peak value without saving histogram is needed.

This thesis proposed two novel algorithms for the dToF system's read out part and tested three other algorithms' functions in mobile phone or AR applications. With the lambertian model and probabilistic theory, a module of the dToF system is built using MATLAB to generate a testing dataset. Besides, introductions to depth-sensing technologies, single-photon avalanche diode (SPAD) sensors, and dToF systems will be given before the core chapters of the thesis.

1 Introduction

The increasing number of digital automotive applications leads to a steadily growing demand for precise depth sensing technology. Depth sensing enables automotive systems to better understand its surroundings, and are based on various technologies including ultrasound, microwave, and optical signals. Direct Time-of-Flight (dToF), one of the most suitable techniques for far range applications [1], is emerging as a promising technology, because of the potential of reaching much longer distances at higher speed and accuracy.

DToF system is mainly used in self-driving or driver-assisting applications [1], however, is is also used in the mobile phones but only in iPhone 12Pro and higher. The implementation details of commercial dToF systems, e.g., the one used in the iPhone, such as the the pixel array size and the algorithms used are unknown. The power for driving the laser in mobile phone systems is limited due to the area cost. Since the sensors that work at the same time will separate the reflected laser power equally, the number of pixels, also referred as the size of the working sensor array, is also limited.

The read out algorithm of the dToF system is barely being focused on. Only a few papers are reporting on read out algorithms [2, 3, 4, 5], they all focused on only limited sizes of the pixel arrays, e.g., 64×64 or 32×32, also have a different average laser power per pixel. It is necessary to investigate if the small pixel algorithms can be scaled to a 200×160 array for the system in this thesis and if there is an algorithm that could cost less area.

1.1 Problem statement and research questions

Problem statement

There are mainly two problems with current dToF system's read out algorithms.

Firstly, it is a novel technology with no publicly available information. It is possible that this 3D image technology cannot be used on a mobile phone at all. A larger pixel array means higher resolution on the image, and low resolution cannot meet the requirement for the mobile phone or AR application. Though a small resolution image could be enhanced in software, it still needs more effort and enhancing time. Considering the price and area cost of the single-photon avalanche diode (SPAD) sensors, also the requirement for image precision, a 200x160 sized pixel array is needed by most mobile phone companies. At present, only Sonny and Apple can maturely take advantage of this technology to realize their commercial products, but these companies never disclose information related to the implementation of any specific details, we do not even know their array's size. Currently,

most papers related to dToF merely focus on limited pixel size, while only a few of them pay close attention to large pixel arrays. Overall, the resources associated with dToF's large pixel array processing are very limited, making the project highly innovative.

Secondly, massive amount of time-to-digital converter (TDC)'s timestamps demanding large on-chip memory capacities. A large pixel array implies masively parallel real-time measurements, which lead to a large volume of data. Consider a 200×160-pixel array, 30,000 measurements per frame and 10-bit TDC. This scenario will generate approximately 2.5GBits of data per frame. Hence an area-efficient algorithm is required for on-chip data processing.

Research questions

This thesis focuses on the following problems:

1. Can algorithms used in dToF systems with small dimensions, e.g., 32×32 or 64×64 pixels, be scaled to a 200×160 array?
2. Are there ways to process TDC's timestamps without saving all timestamps into on-chip memory?
3. Is there a feasible hardware implementation of large pixel array dToF algorithms able to meet the specific requirements of mobile phone applications?

1.2 Thesis contributions

The main contributions of this thesis can be summarized as follows.

1. A MATLAB module of the dToF system, based on the Lambertian model and probabilistic theory.
2. An advanced dToF read out algorithm that is able to process 200*160 pixel array at 30 frames per second and has low memory usage, e.g., less than 3 MB.
3. Register-transfer level (RTL) implementation of the read out algorithms for application-specific integrated circuit (ASIC) implementation.
4. Careful analysis of 5 different read out algorithms.
5. Pointed out the future research directions of mobile phones or AR's large pixel array dtof systems.

1.3 Thesis outline

The rest of the thesis is organized as follows.

Chapter 2: Background Literature This chapter introduces the relevant background of depth-sensing technologies, also presents some details about the single-photon avalanche diode (SPAD) sensor.

Chapter 3: Operation and modeling of dToF system In this chapter, based on the Lambertian model and probabilistic theory, a mobile phone targeted dToF sensor architecture is built and analyzed. Based on the output timestamp from this module's TDC, 5 read out algorithms are introduced and compared, based on the MATLAB module.

Chapter 4: Readout Algorithm's RTL implementation With the diagnoses in Chapter 3, 4 of the 5 algorithms are selected and implemented in RTL.

Chapter 5: Evaluation This chapter present the evaluation result of the selected 4 algorithms, including area and accuracy.

Chapter 6: Conclusion The conclusion of this thesis will be drawn. The main contributions are listed, together with recommendations and prospects for future work.

2 Background and related work

This chapter describes and explores the concepts and approaches to depth sensing technologies, especially dToF. This chapter functions as a general introduction to the dToF system and its read out algorithms.

2.1 Depth sensing technologies

The classification of the major depth sensing technologies is shown in Fig.2.1. Depth sensing technology falls under the broad definition of technologies that enable devices and machines to sense their surroundings, including ultrasound, microwave and optical techniques. Both microwave and ultrasound techniques have severe problems in applications: Microwave techniques affect birds and aircraft, and ultrasound technique's propagation speed varies in different environments. Hence the fundamental and applied research are all based on optical sensing. Optical sensing technology are capable of estimating the position of target in its field-of-view (FOV) and detection range [6]. The important part of optical sensing technology is the trade-off in terms of detection range, FOV, weight, and the size of the system [7].

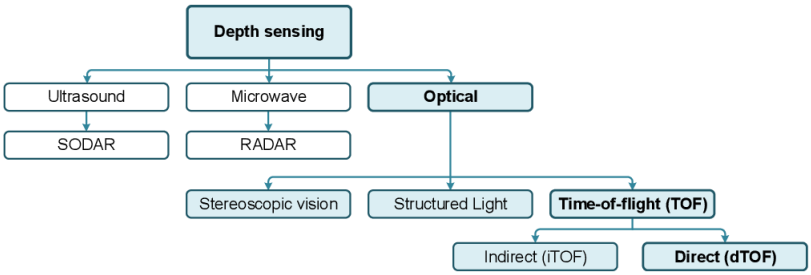


Figure 2.1: Depth sensing technologies [8]

ToF systems are mainly used for optical sensing, classified into direct (dToF) and indirect (iToF) systems. iToF uses intensity-modulated light (typically continuous wave), and measures the phase delay between a continuous sinusoidal light-wave signal and its back-reflected echo. Differently from iToF, dToF directly measures the round-trip time of a light pulse (the time from a short light pulse is emitted until its back-reflected echo is received). This thesis focus on dToF systems. dToF systems are the most suitable technique for far range applications, because the measurement range of the dToF system is only limited by the available optical power budget. On top of that, the work condition of dToF is constant, which means it is more robust [9].

2.1.1 Direct Time-of-Flight (dToF) imaging applications

The dToF system could be applied in many different fields. This section briefly introduces some of the current applications.

Automotive Driving

Automotive driving applications are the main driver for dToF technologies because of high frame rates, which means they are suitable for fast (higher than standard video-rate) 3D ranging[10, 11]. DToF could use a higher optical peak power to overcome high illumination noise, making it the most suitable technique for far-range applications. It is widely used in automotive systems[1].

Augmented Reality (AR)

A potential application of dToF is related to AR[12]. Typically, AR uses software-based depth estimation, making it impossible to use regular RGB cameras or other technologies that require high power consumption and considerable read out elements. Compared to a regular camera, dToF's read out algorithm is less complicated, which means it is possible that a higher frame rate could be realized with a more negligible power consumption and area cost.

Mobile phone integration

In 2020, Apple installed dToF system on the iPhone 12 Pro, applying dToF technology to the mobile phone for the first time, making the mobile phone's camera function more accurate with lower power consumption[13]. Implementing 3D dToF in mobile devices is expected to spark the next wave of killer consumer applications, for example, interior styling and photo-realistic reconstruction.

2.2 Single-Photon Avalanche Diodes (SPADs) based dToF systems

Several physical principles and technologies enable the detection capabilities of range sensors. DToF uses light waves in a typical range to determine the distance between the ToF sensor and the target.

The process of a typical dToF system is shown in Fig. 2.2. One detection in the dToF system has several acquisitions[14], depending on the frame rate and repetition rate. In each acquisition, the pulsed laser sends a laser pulse with special width to the target object, through the transmitter optics. The laser pulse will be reflected by the target object, across the receiver optics, then trigger the SPAD. During one acquisition, 0 to 2 samples will be sent into the read out block by TDC, the number of samples depends on the power of the reflected laser. All the output signal timestamps of TDC during one detection will be saved into one histogram.

The output timestamps of the dToF system follow the Gaussian model [15], the peak of the Gaussian module is the result value. Hence the timestamp that we received the most, also the bin number of histogram's peak is the detected result. The read out part will find out the peak in the histogram and calculate the target distance due to its bin number. The depth data calculated by the read out part will be sent to a mobile phone or AR processor.

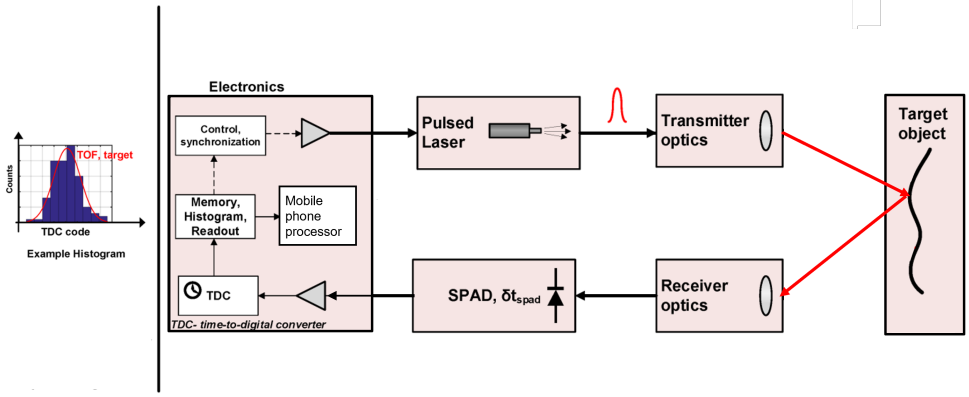


Figure 2.2: The process of a typical dToF system [16]

There are two most common approaches for the dToF system: scanning and flash. Scanning is the traditional method for dToF systems [17]. It is a rotating mechanical scanning dToF system consisting of a laser, scanning motor, receiving sensor (SPAD in dToF case), electronics elements (like TDCs), and the data processor [16].

The systems for flash sensors are much simpler as they do not require mechanical redirection in scanning mode. However, these systems come with an extra cost in electronics complexity, such as more TDCs [18]. The main difference between scanning mode and flash mode is that all the array of SPADs in flash mode can capture depth simultaneously, which means a higher data throughput. However, the achievable optical power of each SPAD in a flash mode dToF is smaller since more SPADs separate the overall optical power. A nominal feasible optical power also means a smaller signal-to-noise ratio (SNR), contributing to the same power of background noise [16].

2.2.1 Single-Photon Avalanche Diodes (SPADs) Sensor

Single-photon avalanche diodes (SPADs) are essential for the dToF image sensor. SPADs can detect a single photon with high time-of-arrival resolution. To be more

specific, SPADs are a kind of particular avalanche photon diode that is biased above the breakdown voltage. In this case, the avalanche effect can be triggered by a single photon, and the diode will break down. [19, 20].

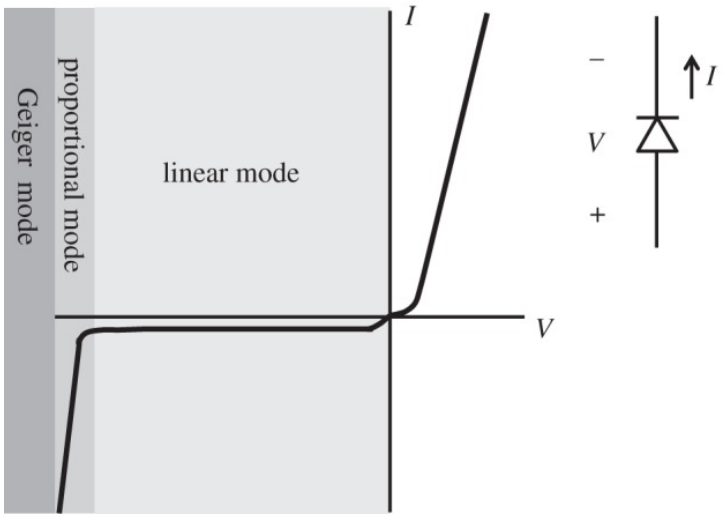


Figure 2.3: The I-V characteristics of a diode. The SPADs operate slightly above the breakdoown voltage, hence could be easily triggered by a single photon [21]

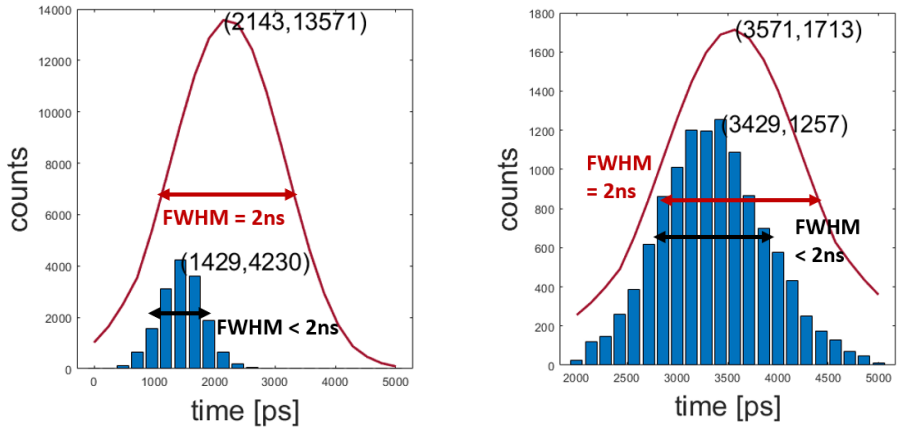
Fig. 2.3 shows the I-V characteristic of a diode. SPAD operates slightly above the breakdown, where the optical gain could quickly change into virtually infinite. When a single photon is detected, a SPAD outputs an analog voltage pulse that can directly trigger the time-to-digital converter (TDC) to save the current timestamp [22].

2.2.2 Distortion on Time-To-Digital converter's timestamps

The data distortion of the dToF system is mainly because of the pile-up of TDC. Pile-up is the phenomenon that after the first or first two photons is detected, the additional photons within the same signal detection period will be lost [23]. The reason for this phenomenon is that in each signal period, TDC has its dead time on signal detection. Pile-up causes distortion of the signal shape and loss in the number of detected events [24].

Fig. 2.4 shows the ideal number of counts at each timestamp and the histogram of actual received data because of the pile-up. Fig. 2.4a and Fig. 2.4b operates with 0.2m and 0.5m target distance separately. Since the laser generates all the signal timestamps, the full width at half-maximum (FWHM) of the resulting histogram

should be the same as the FWHM of the laser. However, that's not the case in reality because of the pile-up. Fig. 2.4a shows a severe distortion than Fig. 2.4b since more timestamps are neglected, because of more signal timestamps.



(a) (b)

Figure 2.4: This figure shows the distortion of pile-up. The red curve in the ideal number of counts that we should receive for each pixel. The blue histogram is the actual histogram that built with timestamps from TDC. (a) shows the figure when the target distance is 0.2m. In (b), the target distance is 0.5m. Both histograms show the distortion in the total number of counts and shape of the curve (a much smaller FWHM for the histogram). The pile-up problem is more severe in 0.2m than in 0.5m because 0.2m has more signals more timestamps are neglected during TDC's operation.

2.2.3 Read out algorithms for SPAD-based dToF system

Efficient read out algorithms for the SPAD-Based dToF system is the main focus of this thesis. Traditionally, SPAD sensors are used for the detection of small arrays. A classical dToF system uses a non-optimized histogram building as the read out part [25, 26, 2]. All timestamps for each pixel will be built into an individual histogram, and the bin holding the peak value indicates the detected result. However, the main benefit of SPAD sensors, also the focus of this thesis, is the potential to design a solution for large pixel arrays. In the case of large pixel arrays, the situation of the read out algorithm is different. Large pixel arrays will generate massive number of output timestamps, which leads to a enormous memory footprint, in other words, prohibitive area cost, as mentioned in Chap. 1.1.

Since the dToF technology is still in its infancy, only a few papers focused on read out algorithms, and all of them still tried to find out the result by histogramming. Especially, Apple used a time-gated scanning technique for the read out part [27]. This time-gated scanning technique is called Partitioned inter-Frame Histogram (***PiFH***). ***PiFH*** stores only a part of the complete histogram at a time, decreasing the area cost with a huge cost on calculating time. Assuming the position of the bin with the largest value of the full-pictured histogram is P_T. For a histogram with N_p bins totally, only N_b bins need to be saved. The index of the partial histogram is N, the position of the bin with the most significant value of the partial histogram is P_C, and the partitions are not overlapping, then P_T could be calculated using:

$$P_T = 2^{N_b}(N-1) + P_C \tag{2.1}$$

Another approach intended to present the histogram with fewer bits is called Folded inter-Frame Histogram (***FiFH***) [3]. ***FiFH*** intended to realize a whole range histogram using two smaller histograms. The second histogram represents the first significant N_b bits of the pixel value, and the second histogram represents the most significant $N_p - N_b$ bits. In [4], the bits of the pixel value is separated into three parts. Hence the peak of the first histogram will be founded in three steps, called the ***three steps*** algorithm.

FiFH may have uncertainty errors when the ToF gaussian bell is centered at multiples of 2^{N_b} bins. Overlaps between the multiple histograms in detection can reduce this problem with the cost of the area. Assuming the position of the bin with the largest value in the histogram with the most and least significant bit is P_{MSB} and P_{LSB} separately. Then P_T could be calculated using:

$$P_T = 2^{N_b}(P_{MSB} - 1) + P_{LSB} \tag{2.2}$$

The newest approach to dToF histogramming saves area using the character of Gaussian bell, called shifted inter-Frame histogram [5]. ***SiFH*** algorithm is based on the fact that the ToF information is concentrated in a gaussian bell, and the noise of the ToF system is evenly distributed. Hence, only saving the center of gaussian bell part is sufficient to find out the histogram's peak value. To keep only 2^{Nb} bins that are centered, all the timestamps have to be shifted into the N_b bit gaussian bell area, using a band-pass filter (filters in RTL). The details about this algorithm is introduced as follow:

Two histograms are saved with two measurements in shifted inter-Frame histograms (***SiFH***). The first one is called coarse histogram (CH), with a bin number of 2^{N_b}. CH will be built with the result of the first measurement. The second one, called fine histogram (FH), will be made with the second measurement result. CH

finds the peak position roughly, then FH zoom in the peak position of the first histogram and finds a more accurate peak result. The number of bits of the CH and FH is both 2^{N_b}, but FH has a smaller gap between each bin to realize an accuracy with 2^{N_p} bins. The N_b is calculated using

$$N_b = [N_p/2] + 1 \tag{2.3}$$

CH and FH can share the same part of the memory since they have the same number of bins.

Before mapping the second N_b-bit histogram (FH), the incoming pixels have to pass through the filter to extract the input data near the peak.

$$TH_- < pixel\ value \leq TH_+ \tag{2.4}$$

The two thresholds is depend on the result of CH and N_b, N_p. Assume the peak of CH is P_{CH}:

$$TH_- = 2^{N_p - N_b} P_{CH} - SB\ ;\ TH_+ = 2^{N_p - N_b} P_{CH} + SB \tag{2.5}$$

SB is the number of side-band bins.

$$SB = 2^{N_b - 1} \tag{2.6}$$

To map the 2^{N_p} bits inputs into a 2^{N_b} bit histogram, a shifting value Δ is used to shift the inputs:

$$\Delta = TH_- = 2^{N_p - N_b} P_{CH} - SB \tag{2.7}$$

After the peak position of FH (P_{FH}) is found, it will shifted back (shifted upwards) to compute the real result:

$$real\ result = P_{FH} + \Delta \tag{2.8}$$

FiFH and ***SiFH*** are mainly used in automotive applications, either have massive power consumption or are used in small pixel scenarios, like 64×64 and 32×32 pixels. The performance of these algorithms in this thesis's application is unknown and still needs to be tested.

2.2.4 Improvement methods

Several methods are introduced to improve the accuracy of dToF systems. The improvement methods are mainly of two kinds: improving the SNR of TDC's timestamps and increasing the number of received input timestamps. Coincidence window is a methodology to improve the SNR of TDC's timestamps. Binning and time-gating are techniques to increase the number of input timestamps.

Binning

When the target object is far away from the pulsed laser, due to a weaker reflected laser pulse, only a small amount of timestamps could be detected from TDC. Binning methodology is used to increase the number of detected signal timestamps. Pixel-binning is a usual ability for image sensors [28], combining changes of adjacent pixels in a block into a super-pixel during the read out process or while the generation of pixel's data [29]. The process of pixel-binning is shown in Fig.2.5. This process could produce an SNR improvement equal to the binning factor [30], which is $B * B$.

In the dToF system, binning is mostly done in the analog part before the data process [31].

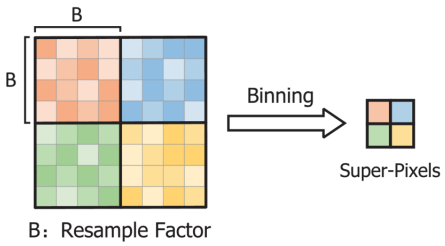


Figure 2.5: The pixel-binning process of image sensors. In this picture, the binning factor B = 4. The four blocks, each with $B * B$ pixels, are binned into four super-pixels. All the timestamps in each block will save the same data, also get same result. [32]

Time-Gating

When the background light is sufficient, the effective triggering rate of SPAD is low [33]. The time-gating technology improves this situation by shortening the sensitive period of the SPAD, increasing detection depth when used in long-distance optical imaging [34]. The technique can be implemented by gating schemes [35].

Coincidence Window

Based on the fact that two nearby SPADs have similar result, a coincidence window could be added to reduce the noise due to background light. The coincidence window can also help mitigate pile-up distortion since it could avoid unnecessary TDC samples [24]. Pile-up is a phenomenon that occurs due to the loss of photons after the first or first two photon within one same signal period is already detected. Pile-up distorts the signal shape and also reduces the number of efficient signal events [24].

For a system with coincidence window technology, one SPAD's one signal event is regarded as effective only if the nearby SPADs have a signal event that happens within the time of coincidence window [16].

Overall, the improvement methods except binning will not influence the resolution of the output image and are possible to be implemented in this thesis's system.

3 Operation and modeling of dToF system

Understanding the working flow of the whole system is essential for read out algorithm design. This chapter built up a module of dToF system based on lambertian model and probabilistic theory, then tested the function of 5 read out algorithms in MATLAB.

3.1 Overview

As mentioned in Chap. 2.2, there are two modes of dToF systems, called flashing mode and scanning mode. This thesis focus on implementing a flashing mode dToF system. The condition used is listed as follows:

- **Laser:** $P_{laser(avg)} = 20W$; $F_{laser} = 100kHz$; $\lambda = 905nm$; $FWHM = 2ns$;
- **System:** AFOV = 0.48°; $T_{int} = 45ns$;
- **Lens:** A_{lens} = 1.16×10^{-6} m^2; $\eta_{lens} = 0.7$;
- **Sensor:** PDE = 0.2; number of SPAD sensors: 160×200
- **Frame rate:** 30fps;
- **Noise:** DCR = 10^4Hz; background noise $\leq$ 50klux;

$P_{laser(avg)}$ is the average peak power of dToF system's laser, F_{laser} is the laser frequency of dToF system. AFOV is the angle of field-of-view. A_{lens} is the surface area that is covered by the lens. η_{lens} is the lens efficiency. PDE means photon detection efficiency and DCR means dark count rate.

3.2 Photon event generation module

Though this thesis focus on the read out algorithm part, the optical module of the dToF system is needed for testing data generation. In this section, the number of effective avalanche events, also the number of timestamps will be calculated based on the energy of both signal and noise.

3.2.1 Signal event

Assuming no laser energy is wasted in out-of-view regions, the total number of signal events can be calculated based on the reflected energy per pixel. The target object of one single SPAD pixel is a single point. A single point has an area that approaches zero, hence could be considered a small Lambertian surface. A

Lambertian surface for reflection is a surface that appears uniformly bright from all directions of view and reflects the entire incident light. [36].

The physics module of the dToF system is shown in Fig. 3.1. The laser shoots a particular wave to the Lambertian target through the diffuser. The reflected wave passes through the lens and gets collected by the sensor.

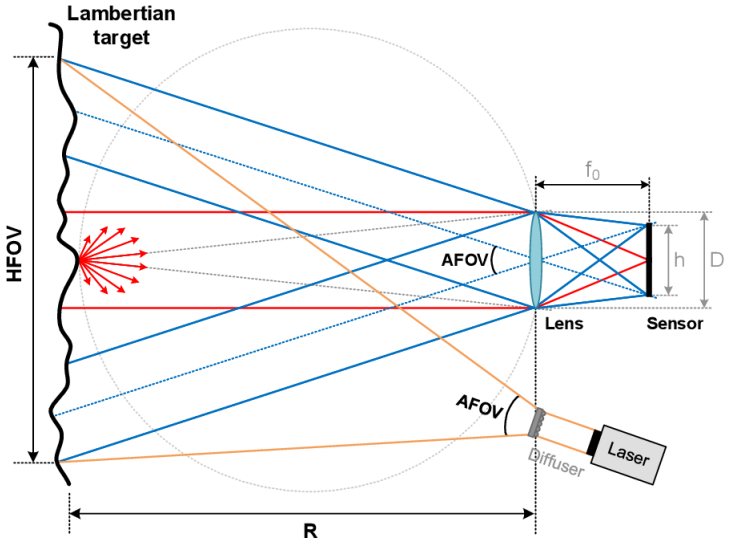


Figure 3.1: Module of dToF system, using a Lambetian reflectance as target. [8]

While passing to the sensor through the lens, the optical power will be concentrated and projected. The scenograph of lens and sensor is shown in Fig. 3.2, A_{sensor}, A_{LENS} and A_{lens_proj} are the area of the sensor, lens, and lens projection area separately. The ratio between A_{sensor} and A_{lens_proj} could be easily calculated using the geometric relationship, the result if $2/\pi$. The whole sensor and some power will lose while passing through the lens due to the lens efficiency η_{lens} of the lens itself, the equation of optical power per pixel is shown in Eq.3.1. The reflected laser power will be divided by all the SPADs, so the P_{lens} has to be divided by M, M is the number of SPADs. The target for each SPAD is exceptionally close to each other, hence assuming the power received for each SPAD is the same.

$$P_{pixel} = P_{lens} \cdot \eta_{lens} \cdot \frac{2}{M \cdot \pi} \tag{3.1}$$

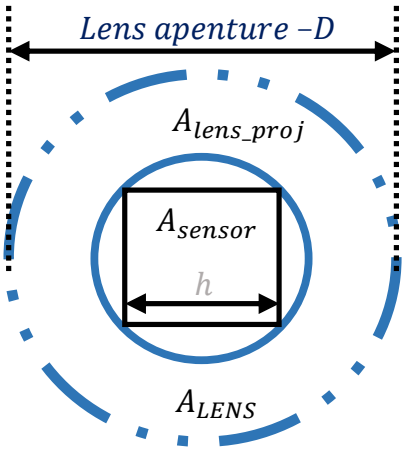


Figure 3.2: The scenograph of lens and sensor. The large outside circle is the cutting plane of lens and the square is the sensor. The area inside the small circle the area of lens projection. [8]

The energy that is collected at the lends corresponds to the area ratio at aperture D. The sphere surface with diameter R. The reflected laser power from the target P_{target} depends on the source power $P_{source}(R)$ and the reflectivity of the object, ρ_{target}. The power source, $P_{source}(R)$, is a function of the distance. Overall, the P_{lens} could be calculated using Eq.3.2.

$$P_{lens} = P_{source}(R) \cdot \rho_{target} \cdot \left(\frac{D}{2 \cdot R}\right)^2 \tag{3.2}$$

By combining Eq.3.1 and Eq.3.2, following equation could be obtained:

$$P_{pixel} = P_{source} \cdot \rho_{target} \cdot \left(\frac{D}{2 \cdot R}\right)^2 \cdot \eta_{lens} \cdot \frac{2}{M \cdot \pi} \tag{3.3}$$

Eq.3.3 is the average power that recieves by each SPAD. Hence it could be integrated over a certain time, to calculate the energy per laser pulse (E_{pixel_pulse}), using:

$$E_{pixel\_pulse} = P_{pixel} \cdot \frac{1}{F_{laser}} \tag{3.4}$$

Thus, the number of photons that generated per laser pulse N_{p_pulse}, could be calculated as:

$$N_{p\_pulse} = \frac{E_{pixel\_pulse}}{E_{photon}} = P_{pixel} \cdot \frac{\lambda}{F_{laser} \cdot h \cdot v} \tag{3.5}$$

where λ is the light source wavelength that laser used, h is the Planck's constant ($6.62607004 \times 10^{34}\ m^2\ kg/s$), and v is the speed of light ($2.998\times 10^{8}\ m/s$).

As mentioned in Chap. 2.2.1, if a SPAD is triggered successfully, it convert photons into electric current. The photon detection efficiency (PDE) defines the probability that a photon trigger a SPAD successfully. Hence the number of signal events that trigger SPAD successflly per laser pulse per pixel is:

$$N_{signal\_events\_pulse} = N_{p\_pulse} \cdot PDE = P_{pixel} \cdot \frac{\lambda}{F_{laser} \cdot h \cdot v} \cdot PDE \tag{3.6}$$

.

As mentioned in Chap. 2.2, the timestamps of triggered signal events follows the Gaussian model. With $N_{signal\_events\_pulse}$ and the full width at half-maximum (FWHM) of laser, a model that estimate the number of triggered signal events per timestamps could be generated.

3.2.2 Noise event

Fig. 3.3 is the figure of solar irradiance curves, published by The International Commission on Illumination (CIE), the world authority on radiometric and photometric nomenclature and standards, for the first time. From the solar irradiance curve at the earth's surface, the power of sunlight at different wavelengths could be estimated. Since the read about this picture is a trivial part of this thesis, the detail about reading is not presented. The maximum the energy of outdoor sunlight is 50klux. And the maximum energy of indoor sunlight is 14klux.

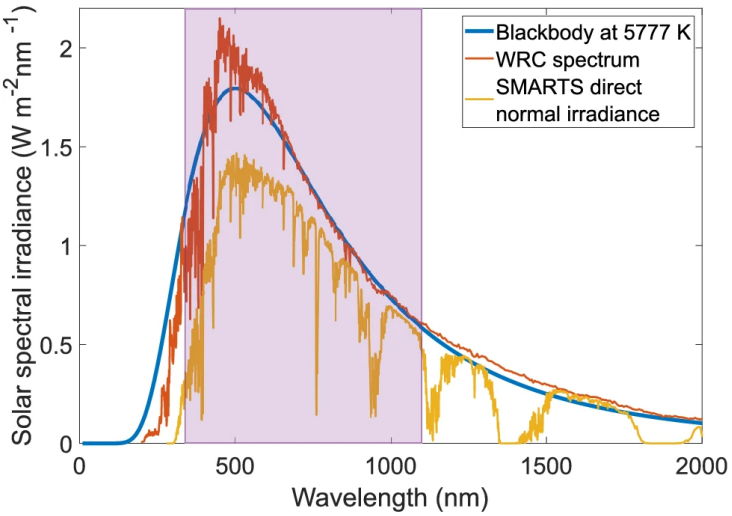


Figure 3.3: Solar irradiance curves at top of difference places. The "SMARTS direct" one is the solar irradiance curve at top of earth surface [37]

Since the chance that sun irradiance directly irradiated onto the sensor is very rare, the reflected light is the main affection that we should consider. Hence assuming that no direct sunlight could reach the sensor lens. [8] and [15] also used this assumption. Fig. 3.4 shows the transmission of noise events from the target to the sensor. It could be easily seen that the transformation of noise is the same as signal events, except they have different power. Due to Eq.3.6, we could refer that the number of effective noise events per second per pixel is:

$$N_{noise\_events\_pulse} = P_{pixel\_noise} \cdot \frac{\lambda}{F_{laser} \cdot h \cdot v} \cdot PDE \tag{3.7}$$

In which, P_{pixel_noise} is the noise power that could be received by each SPAD, calculted similar as Eq.3.3:

$$P_{pixel\_noise} = P_{sun} \cdot \rho_{target} \cdot \left(\frac{D}{2 \cdot R}\right)^2 \cdot \eta_{lens} \cdot \frac{2}{M \cdot \pi} \tag{3.8}$$

P_{sun} is the sunlight power that reach target, could be calculated by integrating the sunlight energy over a special spectrum bandwidth (depends on the laser frequency), AFOV is the angle of FOV:

$$P_{sun} = \zeta \cdot (2 \cdot R \cdot tan(AFOV/2))^2 \cdot P_{sun\_integrated} \tag{3.9}$$

In which ζ means the efficiency term of indirect rays reflected on the floor. Since the indirect rays from the floor are small, it is neglected in this thesis's MATLAB module.

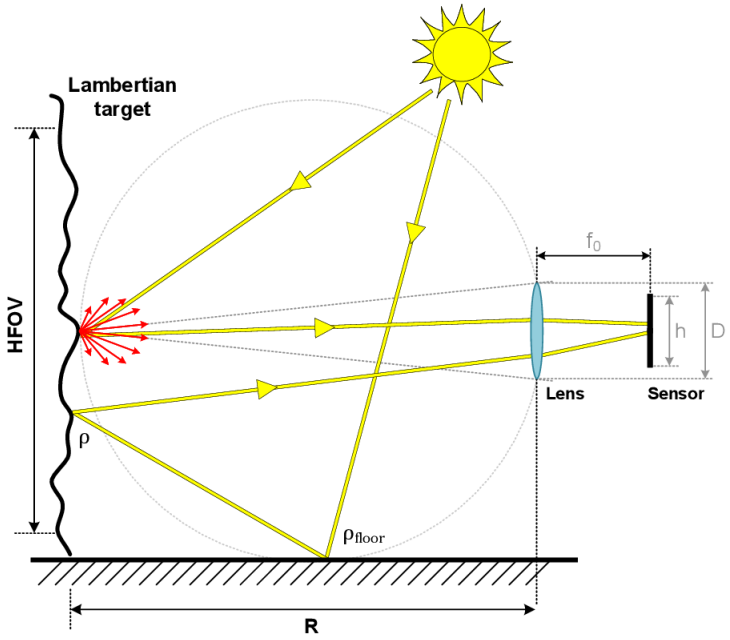


Figure 3.4: How does the noise events being transferred to the sensor. [8]

Differently from the signal, the noise is evenly distributed, and the overall noise value depends on the integration period T_{int}. Typically, the integration window is the largest timestamp that dToF could reach, corresponding to the largest distance that the dToF system could detect. That is also what has been assumpted in this thesis.

On top of the noise from sunlight, there are also noise from the internal SPAD. These noise are reported as dark count rate (DCR) by analoge designers, should be added in Eq.3.9:

$$N_{noise\_events\_pulse} = T_{int} \cdot (P_{pixel\_noise} \cdot \frac{\lambda}{h \cdot v} \cdot PDE + DCR) \tag{3.10}$$

The R in P_{pixel_noise} cancels each other, hence the number of noise events is not influeced by the target distance.

If the number of occurrences per interval time follows Poisson distribution, the length of time between occurrences follows exponential distribution[38]. Since the noise events follow the Poisson distribution, the length of time between each

noise event follows the exponential distribution. The length of time between each noise event could be generated with the definition of the exponential distribution, by adding them together, the noise timestamps could be easily generated.

3.3 Read out algorithms

This thesis introduces a novel read out algorithm that can find out the most frequent timestamp without histogram building, called ***ezPointer***. The extension of ***ezPointer*** is called ***hisPointer***. The other 3 algorithms are the algorithms that introduced by other papers, and their functionality in my system will be verified.

3.3.1 Overview

As mentioned in Chap. 2.2.3, the timestamps from TDC is a train of pulses corresponding to the arrival time of individual photons. Normally, these timestamps are presented as a histogram. Since the timestamps follows Poisson distribution, there is a distinguishable peak centered around the target location, the histogram peak's bin number will be the output result. Fig. 3.5 shows the histogram of all timestamps from TDC during one frame (detection), the target distance of this picture is 1m. In all the examples below, the timestamps from TDC have 10 bits, writen as Q<9:0>.

Assuming the resulting bin number of the algorithm is B_{max}, T_{res} is the resolution of TDC, corresbonding to 1 bin in histogram. FWHM represents the FWHM of laser power.

$$\textbf{detected result} = T_{res} \times B_{max} - FWHM \tag{3.11}$$

In the example shown in Fig. 3.5, the detected timestamp is 8.4ns, slightly smaller than the ideal timestamp, 8.6ns. Pile-up distorts the peak position of timestamps, leading to this difference. However, as long as the peak position remains at the same position during multiple detection, this difference could be offset using a look-up table. The detail about how to calculate relative precision will be discussed in Chap. 3.4.

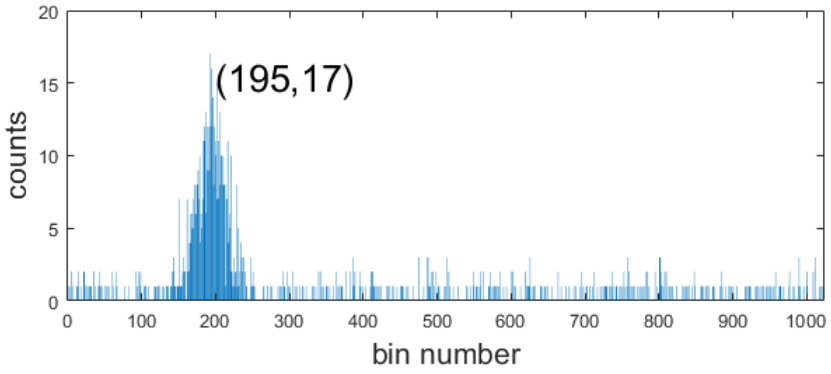


Figure 3.5: the histogram of all timestamps from TDC during one frame. The bin with maximum counts is bin 195, which has 17 counts. Bin number 195 means the range of the timestamp in (194,195] is counted in this bin. The target distance of this example is 1m in a 14klux environment. The examples in this thesis are all in this scenario if not specially mentioned.

3.3.2 The pointer algorithm

The pointer algorithm is one original read out algorithm without histogram building, proposed by this thesis. In order to distinguish between pointer algorithm and the pointer used in pointer algorithm, the pointer algorithm is called ***ezPointer*** in the following part of this thesis. The ***ezPointer*** algorithm takes all the timestamp's center of mass as the final output. The relevant principles are explained as follows.

Consider the condition without background noise. Based on the fact that the laser pulse and overall system timing noise are Gaussian distributed (in time), the detected result (the peak of the laser pulse) is at the center of mass or mean of all input samples in one frame.

Now consider the condition with background noise. As mentioned in Chap. 3.2.2, the typical background noise is uniformly distributed (in time). So if the time period before the detected result and after the detected result is the same, the detected result is the center of mass or mean of all input samples in one frame.

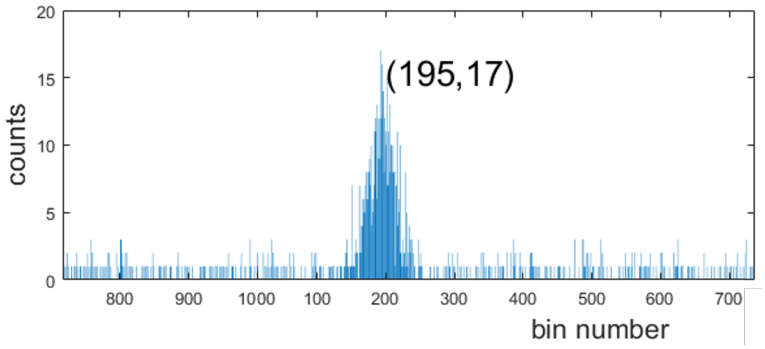


Figure 3.6: the histogram of all timestamps from TDC during one frame (after cut-down and replace conduction). The bins on the right are cut and replaced before bin 0 to keep the bins are equally seperated by the peak of histogram.

However, from the histogram of all timestamps in one detection, shown in Fig. 3.5, the number of bins on two sides of the histogram's peak (detected result) is not necessarily equal. To keep two time periods in the two sides of the detected result the same, a cut-down and reorganization is done. The side with a more extended period will cut half of the exceeded period and place it into the side with a shorter time. The final histogram after reorganizing is shown in Fig. 3.6.

The replaced window has two parts of the same length, separated by the histogram's peak. However, for the ***ezPointer*** algorithm, no actual histogram is built. Hence we cannot finish the reorganizing process at the beginning. The cut-down and replace conduction is a real-time process that happens gradually. The ***ezPointer*** algorithm is a real-time process, which means the pointer moves each time a new input time frame arrives. The left window and right window are introduced to judge where the pointer should move. If the input time frame locates in the right window, the pointer will move right. If the input time frame locates in the left window, the pointer will move left. Otherwise, it will not move. When the pointer moves, the left, and right windows move correspondingly. The boundary of the right window could be computed as:

$$\begin{cases} P_{pointer} < \textbf{inputData} \leq min(P_{max}, P_{pointer} + W_{half}) & , for\ all \\ P_{min} \leq \textbf{inputData} \leq P_{pointer} + W_{half} - P_{max} & , when\ P_{pointer} + W_{half} > P_{max} \end{cases} \tag{3.12}$$

P_{min} and P_{max} are the minimum and maximum possible value of input timestamps. Besides the right window and pointer position, the other part of input timestamp range is the left window. Based on the formula, it is noted that both the left

window and right window are not necessarily one consecutive piece, it could be two pieces, but the left window and right window are always same sized.

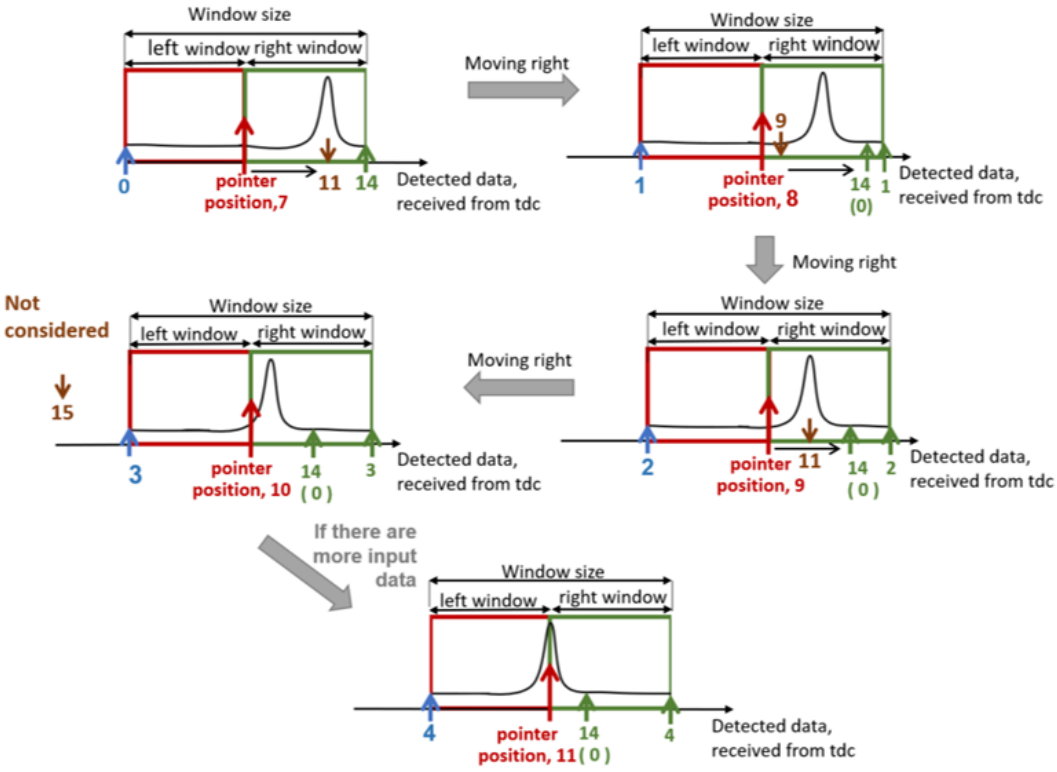


Figure 3.7: An example of the process of pointer algorithm: The pointer will eventually stay at the peak of the histogram if there are a sufficient number of input data.

To better explain the progress of the pointer algorithm, an example is shown in Fig. 3.7. The x-axis of the curve is the time frame received from tdc. The y-axis of the curve represents the number of data received at this time frame during this detection. Therefore, the pointer is at the center of mass if the area below the curve is evenly divided.

$P_{pointer}$ is the current location of the pointer, W_{half} is half of the pointer's window size, also half the efficient timestamps' range. The pointer window size is the same as the range of timestamps that tdc sends. Initially, the $P_{pointer}$ locates at W_{half},which is $(P_{max} + P_{min})/2$, in the middle of P_{min} and P_{max}. In this example, the possibility of receiving a time frame that is larger than W_{half} is apparently larger than the possibility that was receiving a time frame that smaller than W_{half}. Thus,

the pointer should move right to divide the area below the curve equally.

In the example shown in Fig. 3.7, the pointer locates at seven at the beginning. The first arriving data is 11, found in the right window. Hence the pointer will move to 8, and the range of the left and the right window will also change accordingly. The second arriving data is 9, also located in the right window. The pointer will keep moving right into 9. The third arriving data is 15, outside of all the windows, so it's not considered. If there are a sufficient amount of input timestamps, the pointer will remain at the center of mass of all the timestamps eventually.

One potential problem of this algorithm is that only a limited number of samples could be detected in each acquisition. If the number of samples is insufficient to move the pointer to the center of mass, the result will always be wrong. One potential solution is adding the step size. Normally, the step size of ***ezPointer*** is 1 bin, it could be added, with the cost of precision (the highest precision for ***ezPointer*** is the step size).

Another potential problem is that when the system faces severe pile-up distortion, the signal events do not follow a symmetrical gaussian bell, the detected result of ***ezPointer*** may not be the peak of signal events. However, this problem could be reduced by building a look-up table and speculating the actual peak of signal events since the center of mass could also represent a different target distance.

3.3.3 The inter-Frame Histograms (iFH) algorithm

As introduced in Chap. 2.2.3, the newest approach to dToF histogramming is ***SiFH*** and ***FiFH***. In these two algorithms, the timestamps in one frame are evenly separated into two measurements to build up the two histograms accordingly. The only difference between ***SiFH*** and ***FiFH*** is that they used different bits to build up the two histograms. If changing the relationship between Np and Nb in Eq. 2.3 to $Np = 2Nb$, the ***SiFH*** algorithm will be ***FiFH*** algorithm. To be more specific, assuming Np is 10, then a full-range histogram for timestamps has $2^{10} = 1024$ bits. In this scenario, the first histogram in ***SiFH*** will have 64 bins. Each bin represents 16 bins in a full-range histogram. The first histogram in ***FiFH*** will have 32 bins. Each bin represents 32 bins in a full-range histogram. The first histogram building is called the peak detection (PD) part, and the second histogram building is called the partial histogramming part (PH).

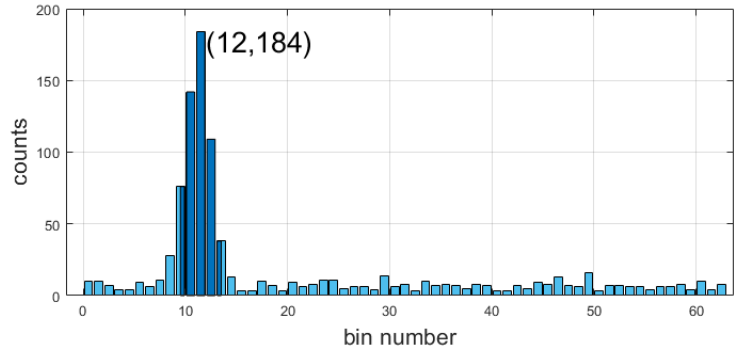
Till now, ***iFH*** algorithms have been used for small distance dToF systems only. This thesis is intended to figure out if the ***iFH*** algorithm is suitable for mobile phone or AR applications.

FiFH, mentioned in Chap. 2.2.3, may face a larger relative precision when the resulting timestamp locates at the border, with low SNR. However, this may not be the case in this thesis's application for three reasons:

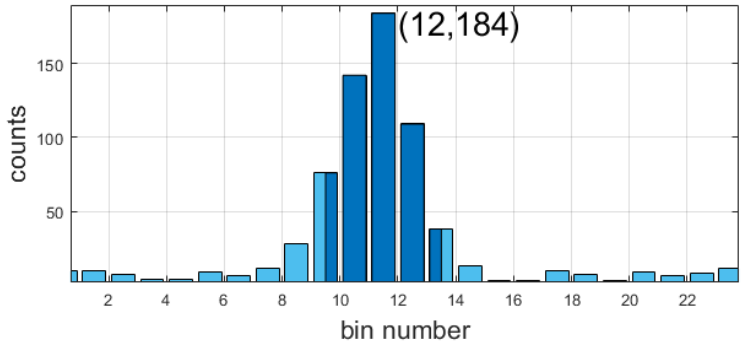
1. The target distance of [5] is smaller than 1m, with a 1cm distance resolution. In this thesis's scenario, the distance resolution at 0.2m is 20cm, and the distortion faced by [5] may not even be a problem.
2. Work [5] have only $160\mu W$ power per pixel. In the situation of this thesis, the average power is $625\mu W$, leading to a higher SNR.
3. Work [5]'s Nb is only 3, leading to an only eight bins histogram. In the design of this thesis, if we want to present 1024 bins using two histograms with the same number of bins, the bin number of the histogram will be 32, much more accurate than [5]. Overall, because of the large difference in the environment, it is possible to get a different result on FiFH than its original result.

The reason [5] used only eight bins for the first histogram is that it wants to use signaled clearing mechanism, with which the bins of the histogram and the peak value of each pixel have to be saved separately in two SRAMs. A small first histogram is chosen to save the area of peak detection (PD) part. That's also the reason that the later introduced ***three steps*** algorithm has a small bin number on the first histogram. The details about signaled clearing mechanism will be introduced in Chap. 4.4.2. In ***iFH***, as mentioned in Chap. 2.2.3, the timestamps of one frame will be separated into two parts.

The details below take the ***SiFH*** method as an example. The first one is called peak detection (PD) histogram, with a bin number of 2^{N_b}, shown in Fig. 3.8. The red square is the filter generated with the first detection result based on Eq.2.4.



(a)



(b)

Figure 3.8: (a) shows the coarse histogram built with the first measurement result. During the second measurement, only the timestamps inside the darker part will be saved. Four bins are dark colored in total. The timestamps inside the range (9.5,13.5] will be saved. (b) zoomed in on the peak part of (a) to present how the two bins beside the histogram's peak are equally separated into two parts.

The second one, called fine histogram (FH), will be built with the second measurement result. CH finds the peak position roughly, then FH zoom in the peak position of the first histogram and finds a more accurate peak result. The number of bits of the CH and FH is both 2^{N_b}, but FH have a higher resolution for each bin to realize an accuracy with 2^{N_p}. Because of the same bin number, CH and FH can share the same part of the memory. The FH is shown in Fig. 3.9.

Since the detail of ***SiFH*** algorithm is already introduced in Chap. 2.2.3, it will not be explained in detail here.

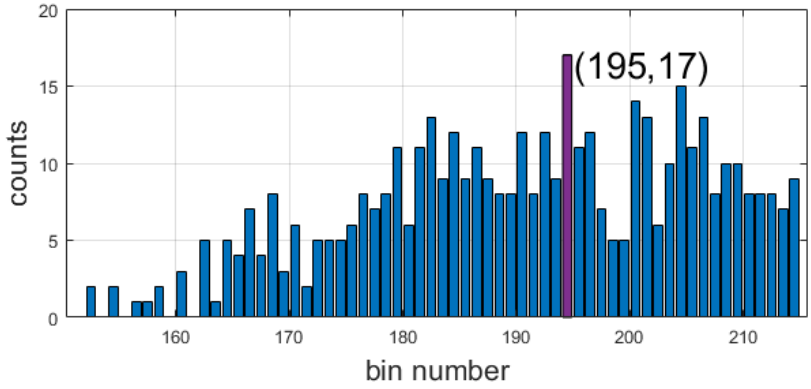


Figure 3.9: This picture shows the fine histogram that was built with the result of the second measurement. Bin numbered 195 is the peak of this histogram, meaning the most frequently received timestamp during the second measurement is represented in range (194,195]. A peak value of 195 will be used to calculate the final detected distance.

3.3.4 The three steps algorithm

The peak detection method that based on three histograms, introduced in [4], is already mentioned in Chap.2.2.3. In this thesis, the method in [4] will be called "***three steps***".

The timestamps during the first measurement will be separated into three parts in the ***three steps*** algorithm. In each step, the searching resolution of a special range will improve. The range will be determined based on the peak during the last step.

The progress of the ***three steps*** algorithm is shown in Fig. 3.10. Same as ***SiFH***, the ***three steps*** algorithm also has two parts. peak detection (PD) will vaguely find out the peak position of the histogram and PH will build up a small histogram with higher precision. Partial Histogramming (PH) in ***three steps*** algorithm is according to the FH part in ***iFH*** algorithm.

For the PD part, firstly, only the most-significant bits Q<9:7> will be considered, that's the region T1 in Fig. 3.10. In the second step, a filter that same as the first histogram peak's time range (T1) will be used, Q<6:4> is considered. Finally, a histogram will be built based on Q<3:1> of the third step. The reason that Q<3:1> instead of Q<3:0> is used is that there are only 3 bits for the PD's memory.

For the partial histogramming part, with the timestamps of the second mea-

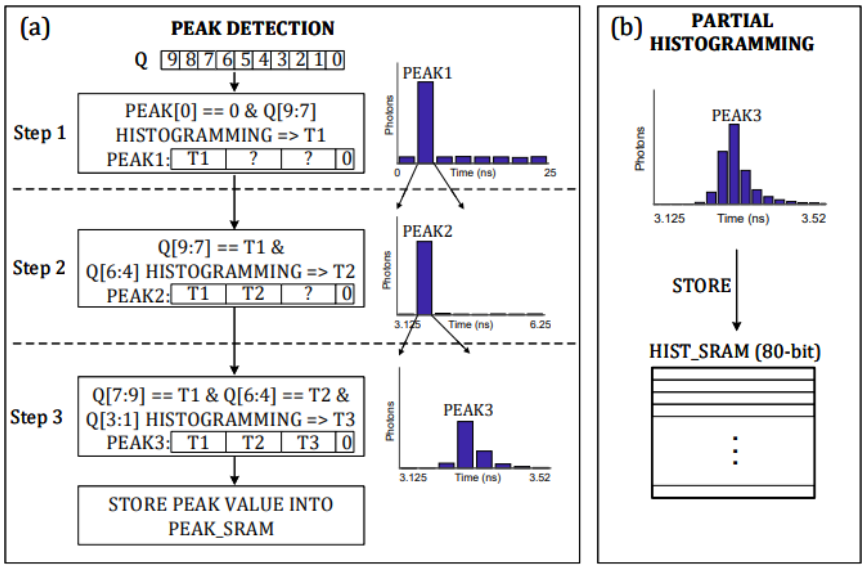


Figure 3.10: The progress of "three steps" algorithm. (a) peak detection with a 3-step successive estimation of histogram peak. (b) the partial histogram that generated during step 3, also the histogram that save into SRAM [14]

surement, a histogram that has 16 bins is built based on T3.

Since the timestamps are in-depth at 10-bits and by each step of PD, the full TDC range is divided into 8 sections. In the third step of PH, each section is combined with 2 bins in whole range histogram. In PH, each bin represents 1 bin in whole range histogram only.

3.3.5 The pointer algorithm with histogram (version 1)

The drawback of the algorithm ***ezPointer*** is that the result of it depends on the acquisition number. In reality, the algorithm's input will be a limited amount of samples, making the detected noise not strictly evenly distributed. Also, if the pointer step is fixed as 1, the limited events number may result in stopping before the pointer moving to the center of mass.

One intuitive method is to use a histogram to decide the original peak position roughly, called ***hisPointer (version 1)***. Using the rough peak position as the original pointer position, the pointer should be swinging near the peak position and return a more accurate result than the algorithm ***ezPointer***.

No filter is used in this algorithm. Hence the only difference between ***his-***

Pointer (version 1) and ***ezPointer*** is the initial position of the pointer. However, the result of ***hisPointer (version 1)*** is almost as same as ***ezPointer***. This phenomenon means that the number of timestamps in this system is sufficient. ***HisPointer (version 2)*** is introduced to further improve the performance of the pointer algorithm.

3.3.6 The pointer algorithm with histogram (version 2)

HisPointer (version 2) added a filter during the second measurement. An intuitive thinking is to use a filter that is the same as ***SiFH*** or ***FiFH***. However, in reality, this will lead to an unstable result. Fig. 3.11a shows the remaining timestamps in the second measurement when the filter of ***SiFH*** is used. Apparently, the signal timestamps are not all in the window; hence the Gaussian peak is not symmetrical anymore. Since the pointer algorithm is based on the theory that the signal timestamps are symmetrical and the peak value is the center of mass, the pointer algorithm does not work anymore.

To solve this unsymmetrical Gaussian peak problem, a larger size filter window has to be used. Because the signal timestamps are all generated by the laser, the FWHM of timestamps should be the same as the FWHM of the laser. The FWHM of timestamps should be smaller due to pile-up distortion. Hence the number of bins that save signal timestamps should be a maximum of $2 * FWHM/44ps \approx 91$. As mentioned in Chap. 3.3.1, one bin in the full-range histogram with 1024 bins represents 44ps. So, in theory, extending the window on both left and right sides for $91/2 \approx 46$ bins could reduce this too-narrow window problem. The new SB that used is:

$$SB = 2^{N_b-1} + 46 \tag{3.13}$$

Fig. 3.11b shows the remaining timestamps when the larger filter is used. The Gaussian peak is symmetrical again. The example in Fig. 3.11a and Fig. 3.11b used the result for 40×80 pixel, to show the result more clear. The actual result is shown in Fig. 3.11c and Fig. 3.11d. All these pictures are in the situation of 14klux with a 1m distance target.

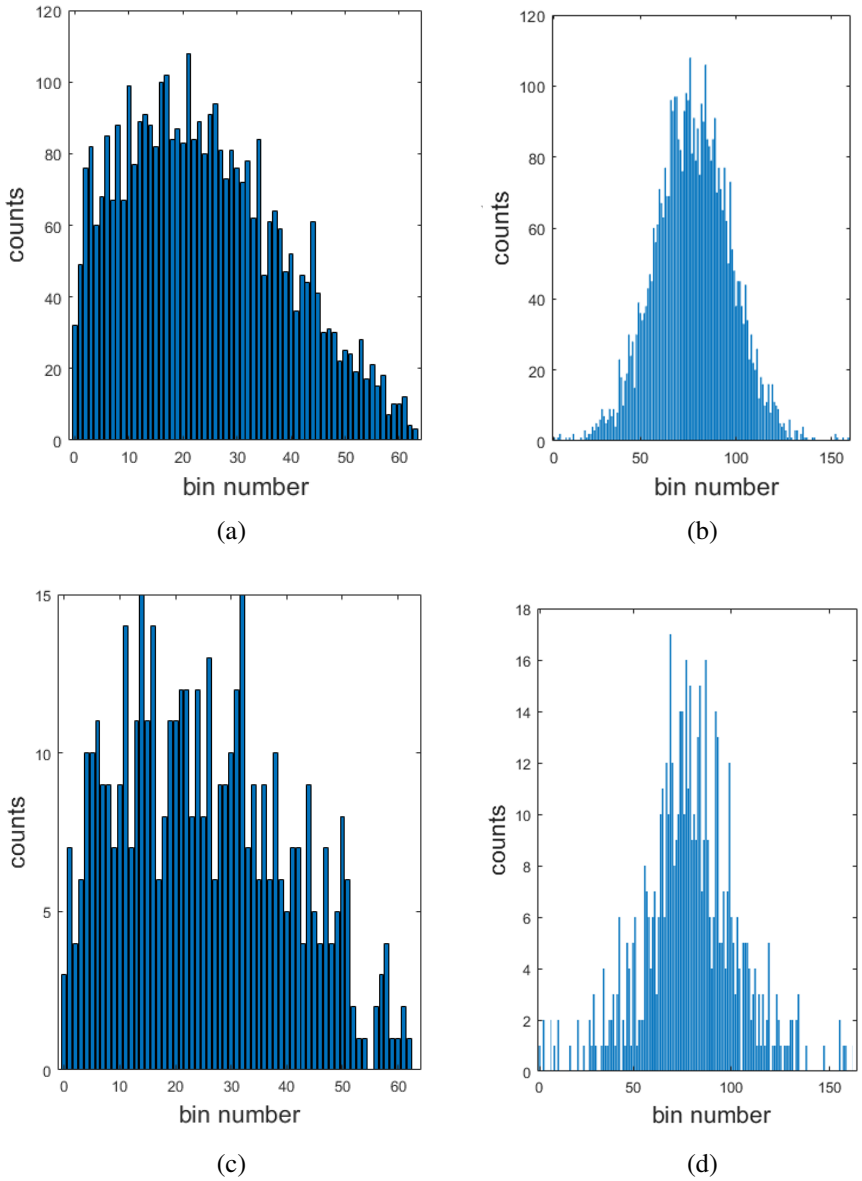


Figure 3.11: The remaining timestamps in the second measurement after filtering. : (a) and (c) are the remaining timestamps after filtering with ***SiFH***'s filter, result for 40×80 pixels and 200×100 separately. Apparently, the gaussian bell inside this filter window is not symmetrical anymore. (b) and (d) are the remaining timestamps after filtering with ***hisPointer***. The whole gaussian bell is all captured in the window. Hence the center of mass of these timestamps is still the peak of the histogram.

All the ***hisPointer*** algorithms that are mentioned in the rest of this paper mean ***hisPointer (version 2)*** .

3.4 Algorithm evaluation

From the content in this chapter, it can be seen that the model of Time-of-Flight imaging is very complicated. Many equations are listed, and several parameters need to be optimized to reach the best work environment for the designed system. Hence a MATLAB model is used to simulate the performance with different parameters and algorithms. The evaluated error is the absolute error of the detected result, calculated using:

$$\textbf{absolute error} = \frac{|\textbf{target position} - \textbf{detected position}|}{\textbf{target position}} \tag{3.14}$$

In Integrated Circuit (IC) design, different algorithms always have different offsets. This offset could be reduced using a look-up table, as long as the algorithm's result for different inputs is always unequal. The standard deviation is calculated using the absolute error, simulated 1000 times per distance.

Normally, there are three criteria to evaluate the performance of processing elements: power consumption, area cost, and algorithm accuracy [39]. It is impossible to get any accurate estimations on power in the MATLAB evaluation part. Even in RTL simulation, the simulated power is always inaccurate. Accuracy is the only result that we could get with the MATLAB module. The area is also considered, using roughly estimation.

As long as the result of the read out part is a function, not a mapping, the difference between the ideal position and detected position could be fixed using a look-up table in the host part, the mobile phone's processor, for example. Hence, we could use the standard deviation of absolute error as our relative precision and evaluate this situation. In the later part of this thesis, all the relative precision mentioned means the standard deviation of absolute error. The linearity of the module will be verified during the simulation.

From the customers of this design, the mobile phone companies, the requirement on relative precision is less than 10% for the distance smaller than 1m and less than 3% for the distance from 1m to 3m. The customer also hopes for a 3% for distance from 3m to 5m, but it is not necessary.

In the following simulation, no improvement methods like binning or time-gating are used.

The relative precision when the background noise is 50klux

As mentioned in Chap. 3.2.2, 50klux is the largest background noise in outdoor application. Apparently, from Fig. 3.12, all the algorithms do not reach the 3% requirement in 50klux. Hence this system cannot be used outdoor.

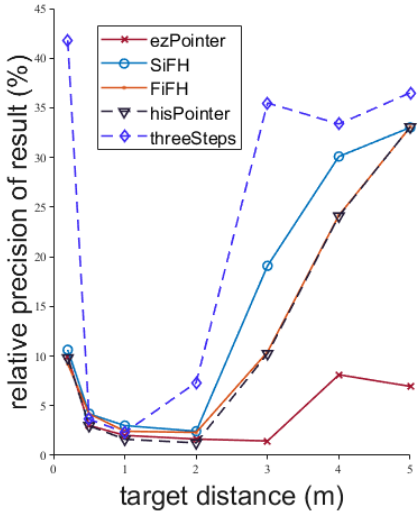
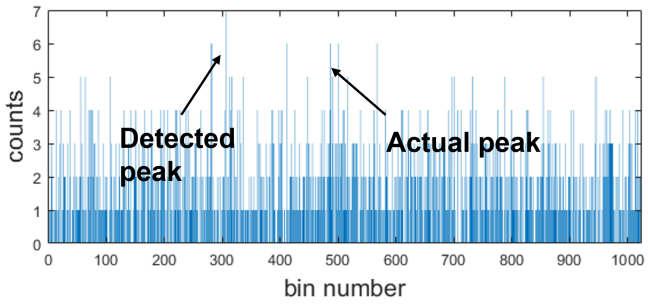
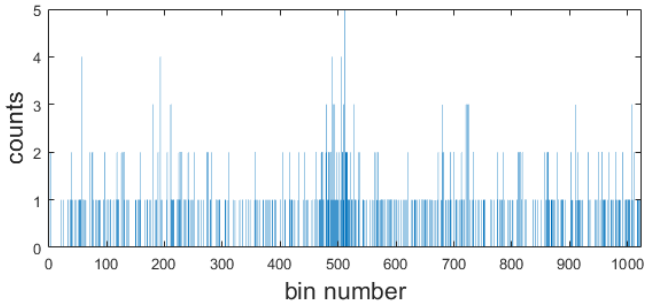


Figure 3.12: The relative precision of different algorithms in 50klux: All the algorithms could not work properly at 3m distance, hence this system cannot be used in 50klux.

The low relative precision for a large distance target in 50klux is due to the low SNR. The SNR of 50klux is higher than 14klux because the background noise does not influence the number of signal events. Fig. 3.13a shows the 1024 bins histogram of all timestamps during one measurement, in 50klux, 3m target distance. To compare, the 1024 bins histogram of all timestamps in 14klux with the same distance is shown in Fig. 3.13b. Typically, the coincidence window method could be used to reduce the noise problem, increasing SNR. However, the coincidence window also reduces the number of signal events and has no improvements in results in this situation. Both “binning” and ”time-gating” methods can hardly improve the system’s SNR, hence not used in this situation. Also, the ”binning“ method will decrease the number of result pixels, which do not meet the 200×160 large pixel array requirement of this thesis.



(a)



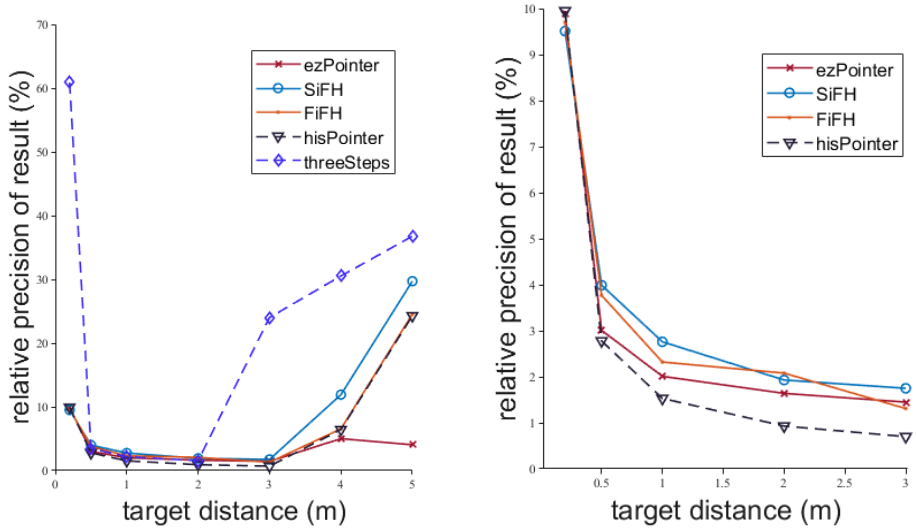
(b)

Figure 3.13: The 1024 bins histogram of all timestamps, in 50klux and 14klux senario, both 3m target distance: (a) shows the 1024 bins histogram of all timestamps during one measurement, with 50klux background noise and 3m target distance. Because of low SNR, the actual peak cannot be distinguished from the noise. A fake peak is detected. (b) is the histogram with 14klux background noise and 3m target distance.

Next, the system will be run in 14klux, the worst case scenario for indoor applications, to see if it can work properly in indoor application.

The relative precision when the background noise is 14klux

Fig. 3.14a shows the relative precision of different algorithms at 14klux. Apparently, all the algorithms except the ***threeStep*** algorithm reach the requirement from 0.2m to 3.5m. When the target distance is larger tham 3.5m, none of the algorithms could work with a relative precision smaller than 3%. Fig. 3.14b shows the curve of other algorithms except the ***threeStep*** algorithm, ranging from 0.2m to 3m. the ***hisPointer*** algorithms shows superior result than the other algorithms.



(a) (b)

Figure 3.14: The relative precision of different algorithms in 14klux: (a) shows the relative precision of all five algorithms. The ***threeStep*** algorithm reach a huge relative precision when the target distance is larger than 2m, do not meet the requirement. (b) shows the relative precision of four algorithms that reach customer's requirement.

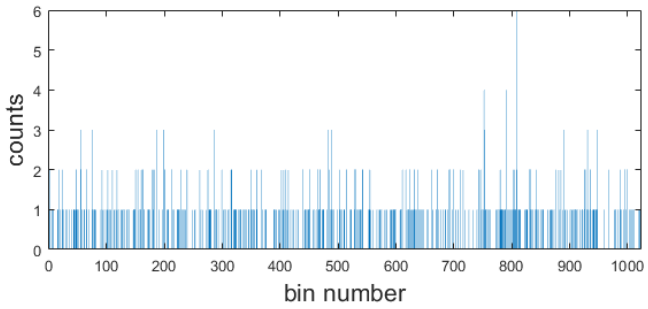
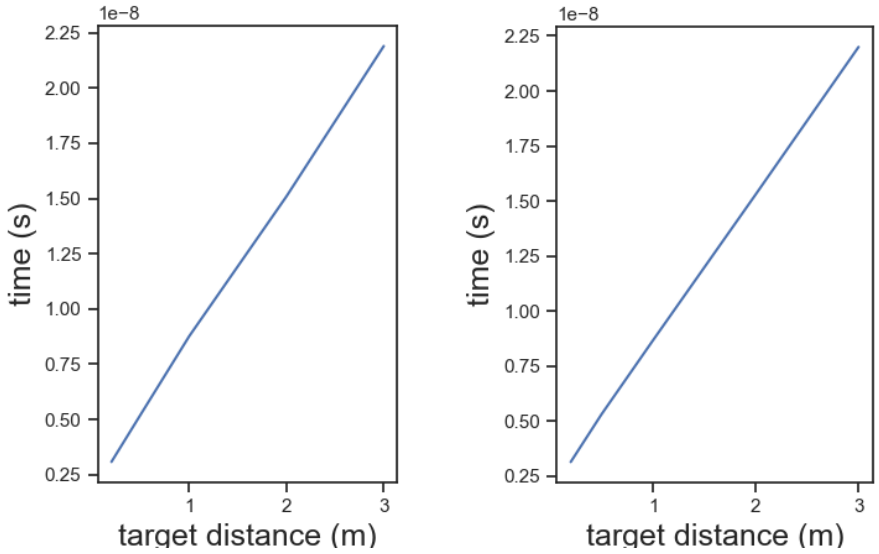


Figure 3.15: The 1024 bins histogram of all timestamps, 14klux scenario, 5m target distance. Compared to Fig. 3.13b, the number of signal events for 5m is so narrow that it cannot present the guassian distribution anymore.

From Eq. 3.3 and Eq. 3.6, the number of signal events is inversely proportional to the target distance. As mentioned in Chap. 3.2.2, the number of noise events is not influenced by the target distance. Hence more minor signal events directly decrease the timestamp's SNR. A much higher SNR lead to the high relative precision of results in long target distance. Fig. 3.15 shows the 1024 bins histogram of all timestamps, 14klux senario, 5m target distance. Compare to Fig. 3.13b, the number of signal events is apparently smaller. Though the gaussian bell still could be founded, the position of peak bin fluctuate because of limited signal number. The ***ezPointer*** algorithm works better than the other 4 algorithms because it finds the center of mass of whole histogram instead of a single peak.

Fig. 3.16 shows the relationship between the ***FiFH*** and ***hisPointer*** algorithm's target distance and result. The result means the timestamp that is received most time in seconds. The detected distance range is from 0.2m to 3m. It could be easily observed that they are linear. Hence the method of using standard deviation to calculate relative precision is right.



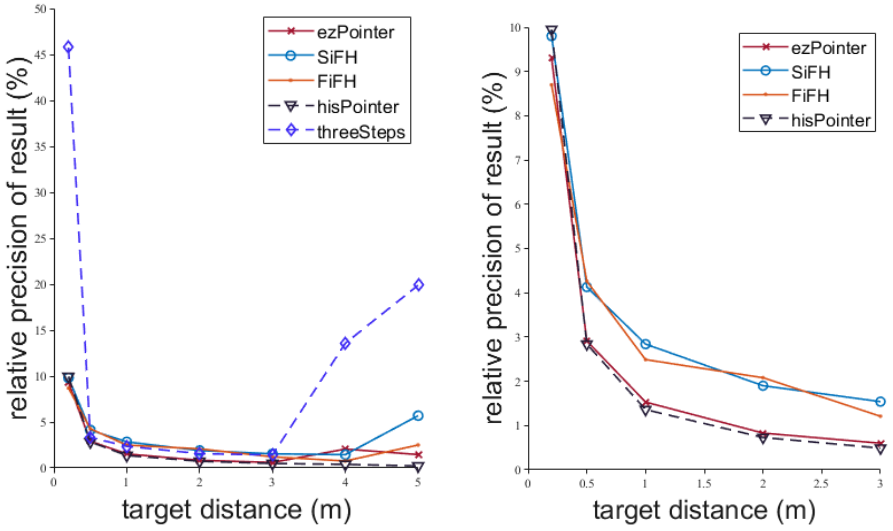
(a) (b)

Figure 3.16: The relationship between target distance and result (in second) in 14klux: (a) ***FiFH***. (b) ***hisPointer***.

Sufficient working in light below 14klux means sufficient to work indoor. After testing, the ***ezPointer*** algorithm shows the best tolerance on background noise, kept working in 25klux. The ***SiFH***, ***FiFH*** and ***hisPointer*** algorithm could reach our requirement until 21klux.

The relative precision when the background noise is 2klux

The typically indoor background noise is 0-2klux. For the system with 2klux background noise, the relative precision is shown in Fig. 3.17. Though the relative precision still gets large after 3m, all the algorithms except the ***threeStep*** algorithm all have a less than 3% relative precision until 5m. ***hisPointer*** shows an extraordinary stable result.



(a) (b)

Figure 3.17: The relative precision of different algorithms in 2klux: (a) shows the relative precision of all five algorithms. The ***threeStep*** algorithm reach a huge relative precision when the target distance is larger than 3m, hence cannot measure the distance that larger than 3m. (b) shows the relative precision of four algorithms that reach customer's requirement.

The above algorithms except ***three steps*** algorithm are sequenced based on their area cost and accuracy. The algorithms that have similar results are rated using same number. The ***ezPointer*** and ***hisPointer*** algorithm shows superior results because high rank in both area and accuracy. All the algorithms except the ***three steps*** algorithm are implemented in register-transfer level.

Algorithm	***ezPointer***	***SiFH***	***FiFH***	***hisPointer***
area cost	1	3	2	2
accuracy	2	3	3	1

Table 1: Evaluated result for different algorithms: Area are listed from smallest to largest and accuracy are listed from highest to lowest, in 14klux

4 Readout Algorithm's Register-Transfer Level (RTL) implementation

In Chap. 3.4, ***iFH*** (***SiFH*** and ***FiFH***), ***ezPointer*** and ***hisPointer*** algorithms were chosen because having the potential to be implemented in mobile phone or AR applications. In this chapter, these 4 algorithms are implemented in RTL.

4.1 Design environment

This section introduces the simulation environment of RTL implementation and also calculates the frequency of this thesis's design.

4.1.1 Simulation environment

This thesis targets on 40nm integrated circuit (IC). Cadence irun utility (version 15.20) is used for the module's functional simulation and the intellectual property (IP) core used for SRAM is the "SiWare Dual Port High-Density Leakage Control SRAM 512K Sync" from the synopsis company, generated by Embed-it! (R) Integrator.

4.1.2 Clock frequency calculation

The standard frame rate of dToF is 30 frames/second. We also use 30 frames/s as our target frame rate, that leads to the time for one frame of about 33ms. Assuming 33,333 acquisitions leads to 33ms/33333 $\approx 1\mu s$ for one acquisition. The detail about the relationship between frame, acquisition, and data of each pixel is shown in Fig. 4.1a and Fig. 4.1b. In the example shown in Fig. 4.1b, there are two pixels, each have two acquisitions. In each acquisition, each pixel receives two data. pixel0 data0 acq0 means the first data of the first pixel detected during the first acquisition.

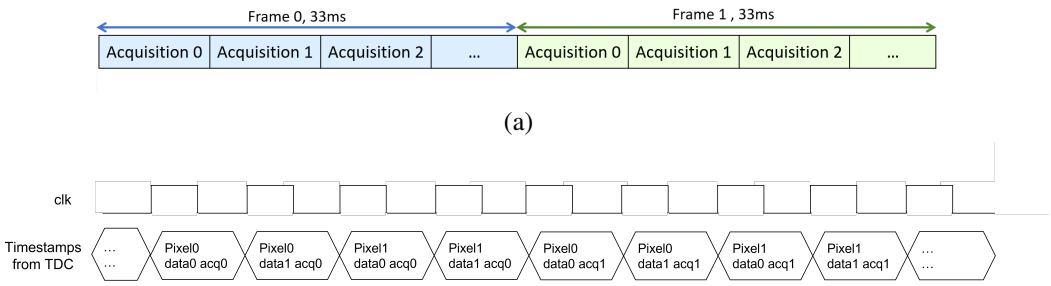


Figure 4.1: The relationship between frame, acquisition and data of each pixel: (a) shows the relationship between frame and acquisition. The standard frame rate of dToF is 30 frame/s. In each frame, there will be n acquisitions. (b) shows how data of different pixels are sent to the digital processing part. pixel0 data0 acq0 means the first data of the first pixel detected during the first acquisition.

The size of our SPAD sensor is 200×160. Since the result of each pixel is shifted in line from TDC, the number of pixels in serial could be 200 or 160. To reduce energy consumption as much as possible, the lowest possible frequency is sought. One hundred sixty pixels in serial cost less time in timestamps reading, leading to a lower frequency. One hundred sixty pixels in serial means 1 μs / (160*2), about 3.125ns one clk, leading to a 320MHz clock frequency. To make sure the system works in different scenarios, the chosen frequency is higher than the calculated limitation, 330MHz in this case.

4.2 Design variables

The variables used in this design are listed as follow:

- ***Nb*** : the bit precision of first histogram.
- ***Np*** : the bit precision of second histogram.
- ***INPUT_MAX*** : the max value of input rough data. $\boldsymbol{INPUT\_MAX} = \sum_{n=1}^{Np} 2^n$
- ***PIXEL_NUM*** : Overall pixel number in sensor.
- ***SRAM_NUM***: the number of pixels that work in parallel.
- ***PIXEL_NUM_PER_RAM*** : number of pixels that save in one SRAM. $\boldsymbol{PIXEL\_NUM\_PER\_RAM} = \boldsymbol{PIXEL\_NUM}/\boldsymbol{SRAM\_NUM}$

- ***BIN_NUM_PER_HIS*** : number of bins per histogram.
  $\boldsymbol{BIN\_NUM\_PER\_HIS} = 2^{\boldsymbol{Np}}$
- ***peakMax*** : number of bits per bin.
- ***pointerSize*** : number of bits that cost by pointer.
- ***ACQ_NUM*** : number of acquisitions.

The actual parameter for this design is listed in Tab.2. To better explain the process of algorithm, a simplified example for each algorithm is given in the following part of this chapter. The input timestamps that used for example is listed in Tab. 4.

Table 2: Actual parameter

parameter	***Nb***	***Np***	***ACQ_NUM***	***PIXEL_NUM***	***SRAM_NUM***	***pointerSize***
Value	6 or 5	10	33,333	200*160	200	10

parameter	***PIXEL_NUM_PER_RAM***	***BIN_NUM_PER_HIS***	***peakMax***
Value	160	64 or 32	10

Data and parameters used in example

Tab. 3 shows the data and parameters used in execution examples.

Table 3: Example parameter for ***SiFH*** and ***ezPointer*** design

parameter	***Nb***	***Np***	***ACQ_NUM***	***PIXEL_NUM***	***SRAM_NUM***	***pointerSize***
Value	3	4	2	1	1	4

parameter	***PIXEL_NUM_PER_RAM***	***BIN_NUM_PER_HIS***	***peakMax***
Value	1	8	3

4.3 The pointer algorithm

The pointer algorithm is the original algorithm from this thesis. No histogram will be built during the process of the pointer algorithm.

4.3.1 RTL design block diagram

Fig. 4.2 shows the schematic of the pointer algorithm. The pointer algorithm has three important parts except for the controller: the window calculator, the pointer calculator, and the result reader. The state machine diagram of the pointer algorithm is shown in Fig. 4.3. There are four states in total.

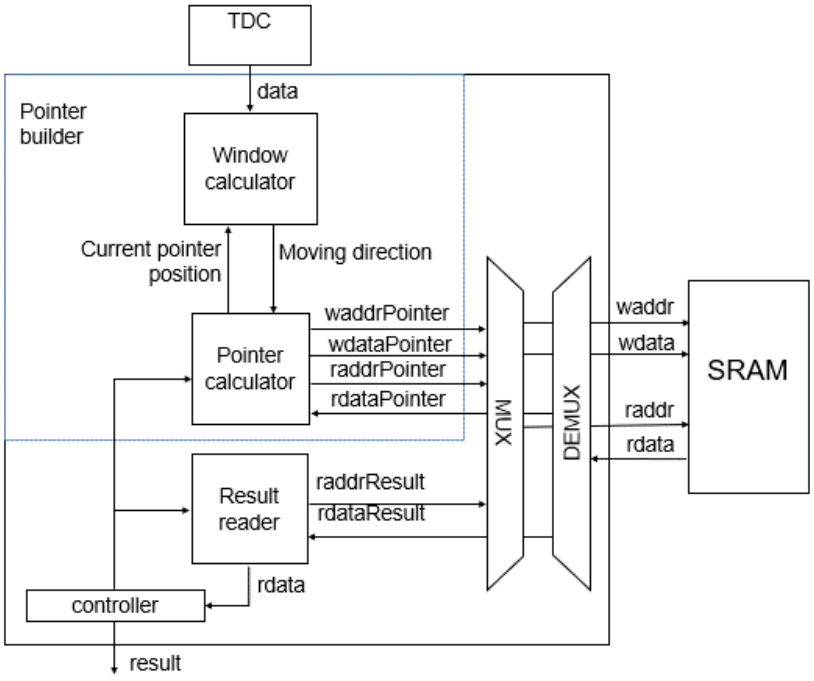


Figure 4.2: The schematic of pointer algorithm

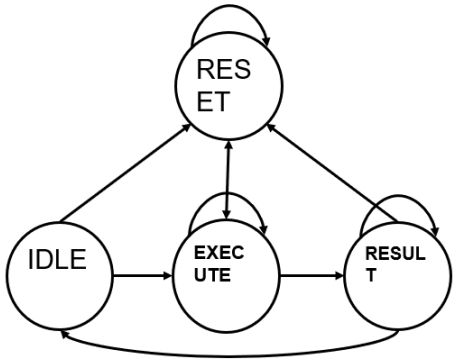


Figure 4.3: The state machine diagram of pointer algorithm

- ***IDLE*** : All the processing blocks are not used.
- ***EXECUTE*** : Calculate the result of each pixel using pointer algorithm and save into SRAM.
- ***RESULT***: This state means all the calculation is finished, the measurement's result will be output pixel-by-pixel in serial.
- ***RESET***: Reset the SRAM, reset all the addresses one by one, one per clock cycle.

The timing diagram of pointer builder is shown in Fig. 4.4. The write address are the addresses that corresponding to each pixel and the write data is the result that saved in SRAM's address. p0_0 and p0_1 means the pointer position after the first and second timestamp of pixel0 are read. The details about SRAM's data changing and structuring is shown in Chap. 4.3.2.

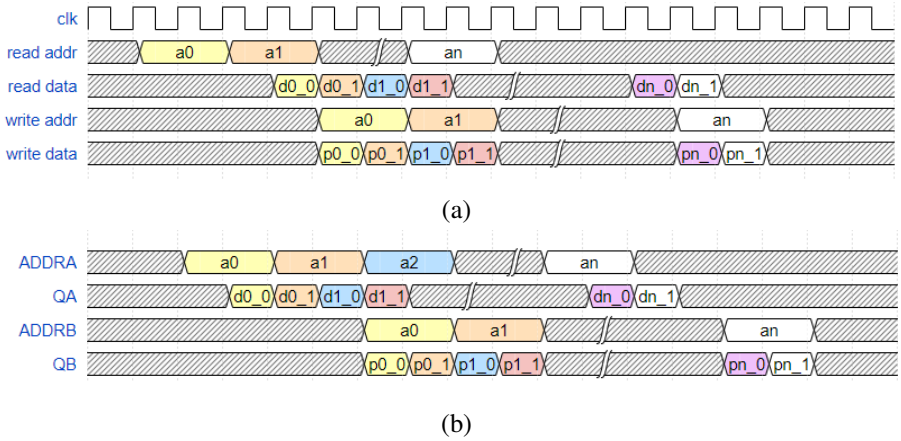


Figure 4.4: The timing diagram of pointer builder: (a) shows the data sequence inside pointer builder. (b) shows the according to data sequence in SRAM. Port ADDRA and QA correspond to read addr and read data separately, port ADDRB and QB correspond to write addr and write data seperately.

4.3.2 SRAM structure and the pointer algorithm's execution example

Only the pointer position itself needs to be saved during the pointer calculation. Only one SRAM is used, the address of SRAM according to the pointer positions of pointer algorithm, as shown in Fig. 4.5. In Fig. 4.5, **PIXEL_NUM_PER_RAM** pixels are working in serial, hence SRAM has **PIXEL_NUM_PER_RAM** addresses. An example of how the pointer positions in SRAM are changed is shown in Fig. 4.6.

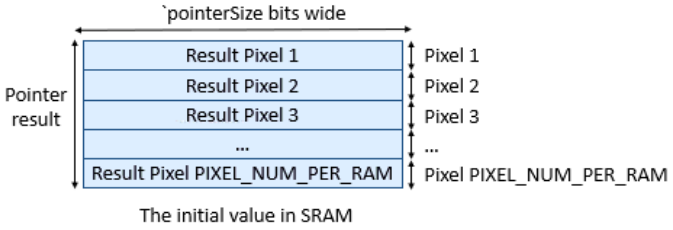


Figure 4.5: The structure of data in a single SRAM: assuming **PIXEL_NUM_PER_RAM** pixels are working in serial

A brief calculation of SRAM size is shown below. All the size memory blocks (SRAM and registers) are in bits during the calculation. Assuming the size of SRAM is $S_{pointer\_signle}$ bits. The area of *SRAM* could be calculated with the following equation:

$$S_{pointer\_signle} = \textbf{\textit{pointerSize}} * \textbf{\textit{PIXEL\_NUM\_PER\_RAM}} \tag{4.1}$$

Using the parameter shown in table 2, we can get the following result:

$$S_{pointer} = \textbf{\textit{pointerSize}} * \textbf{\textit{PIXEL\_NUM\_PER\_RAM}} * \textbf{\textit{SRAM\_NUM}} = 39.06KB$$

Fig. 4.6 shows the SRAM data process during the execution of ***ezPointer*** algorithm. In this example, the timestamps of other pixels are neglected, only the processing of pixel 1 is considered. In each clock signal, the pointer builder read the data in SRAM, according to the current pixel's corresponding address, then calculate a new pointer position. At the same time, the pointer builder write the new pointer position of last pixel. The calculation of pointer position are mentioned in Chap. 3.3.2.

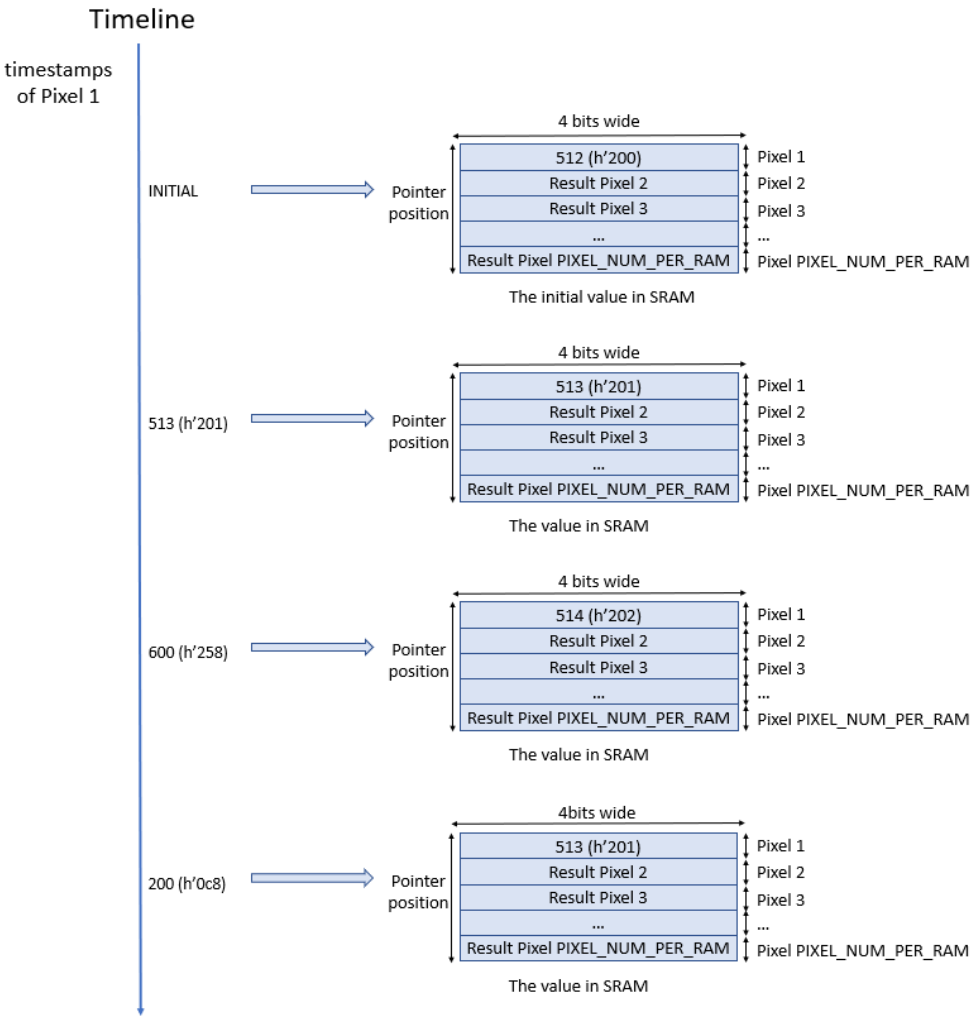


Figure 4.6: A presentation about the data process inside SRAM with pointer algorithm: Assuming the initial pointer position is 512. Based on the Eq. 3.12 If the timestamps given by TDC locates in the right window, the corresponding pointer position in the SRAM will be increased by one. If the timestamps given by TDC locates in the left window, the corresponding pointer position in the SRAM will be decreased by one. Otherwise, it remains unchanged. After TDC completes sending all timestamps, the pointer position stored in SRAM is the final output result for each pixel separately.

4.4 The inter-Frame Histograms (iFH) algorithm

As introduced in Chap. 2.2.3, the newest approach to dToF histogramming by other paper is ***FiFH*** and ***SiFH***. The only difference between ***FiFH*** and ***SiFH*** is that they used different bits to build up the two histograms. If change the relationship between Np and Nb in Eq. 2.3 to $Np = 2Nb$, the ***SiFH*** algorithm will be ***FiFH*** algorithm.

4.4.1 iFH block diagram

The schematic of the ***iFH*** algorithm is shown in Fig. 4.7. It shows the architecture of a single execution block, in the end, there will be ***SRAM_NUM*** execution blocks. There are mainly 4 processing blocks: data filter, histogram builder, peak detector and result output part, all controlled by the controller.

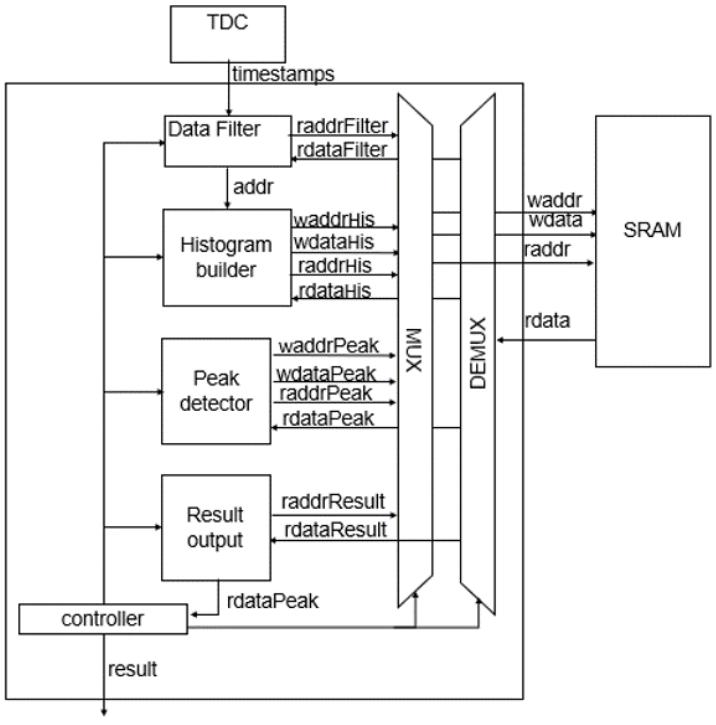


Figure 4.7: The schematic of ***iFH*** algorithm. It shows the architecture of a single execution block, in the end there will be ***SRAM_NUM*** execution blocks.

Controller

Controller controls the state of the execution block. The state machine diagram of controller is shown in Fig. 4.8. There are 8 states:

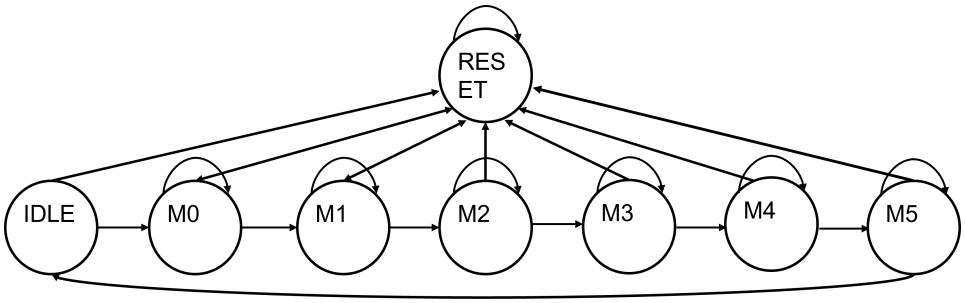


Figure 4.8: The state machine diagram of controller

- ***IDLE*** : All the processing blocks are not used.
- ***M0(BUILD_HIS1)*** : Build the first histogram. The histogram builder block and data filter block are used in this state. The data filter block is used in all histogram building states. The original filter of the data filter block keeps the highest N_b bits. The histogram builder reads the 'addr' from the data filter and adds the counts saved in the SRAM's corresponding address.
- ***M1(FIND_PEAK_HIS1)***: Find the address with the max value that is saved in SRAM (also the bin with max counts in the first histogram). All the addresses in SRAM will be read, one per clock cycle.
- ***M2(CAL_FILTER)*** : Calculate the filter of second histogram, based on the result that founded in ***FIND_PEAK_HIS1***.
- ***M3(BUILD_HIS2)***: Build the second histogram, similar as building the first histogram.
- ***M4(FIND_PEAK_HIS2)***: Find the address with the max value that is saved in SRAM (also the bin with max counts in the second histogram). All the addresses in SRAM will be read, one per clock cycle. This state generates the final result with these two detections.
- ***M5(GENERATE_RESULT)***: This state means all the calculation is finished, these two measurement's result will be output pixel-by-pixel in serial.

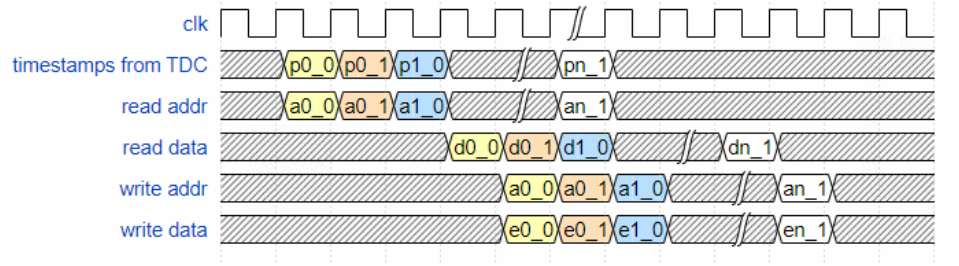
- ***RESET***: RESET the SRAM, reset all the addresses one by one, one per clock cycle.

Data filter

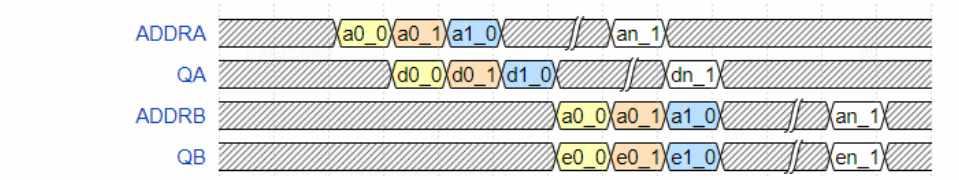
Calculated in state M2, working in all states except IDLE and RESET. The original filter of the data filter block keeps the highest N_b bits of each input timestamps. After the finding out of the first histogram's peak value, the data filter will read TH- from SRAM, based on which a new filter will be built.

Histogram builder

This block works in states M0 and M3. In each rising edge, the histogram builder writes the new counts of the last pixel into the accordingly address in SRAM. At the same time, the histogram builder reads the counts of the current pixel from SRAM. The pictures of the histogram builder's timing diagram are shown in Fig. 4.9. "pn_0" and "pn_1" means the first and second timestamps for pixel n that TDC detects. "a" presents the following reading address of SRAM and "d" presents the data that read from SRAM. "e" presents the data that write into SRAM. For each pixel, "e = d + 1".



(a)



(b)

Figure 4.9: The timing diagram of histogram builder: (a) shows the data sequence inside histogram builder. The four signal exept timestamps for TDC and clk are the signals that used to interact with SRAM. (b) shows the according to data sequence in SRAM. Port ADDRA and QA correspond to read addr and read data separately, port ADDRB and QB correspond to write addr and write data seperately.

Peak detector

This block works in states M1 and M4. The peak detector reads data in one address from SRAM in sequence in each rising edge. Inside the peak detector, there is a register called ***maxValue***, saving the current max value of data that is read from SRAM. Once the number that is saved is ***maxValue*** is refreshed, the peak detector will write the according to bin number into the SRAM as the histogram peak's bin number. The original number that saved in ***maxValue*** is 0. The pictures about peak detector's timing diagram is shown in Fig. 4.10. a0_0 to an_n are the address of SRAM, from 0 to $\boldsymbol{PIXEL\_NUM\_PER\_RAM} * \boldsymbol{BIN\_NUM\_PER\_HIS} - 1$. In the example that shown in Fig. 4.10, $dn\_n > d0\_2 > d0\_0 > d0\_1$.

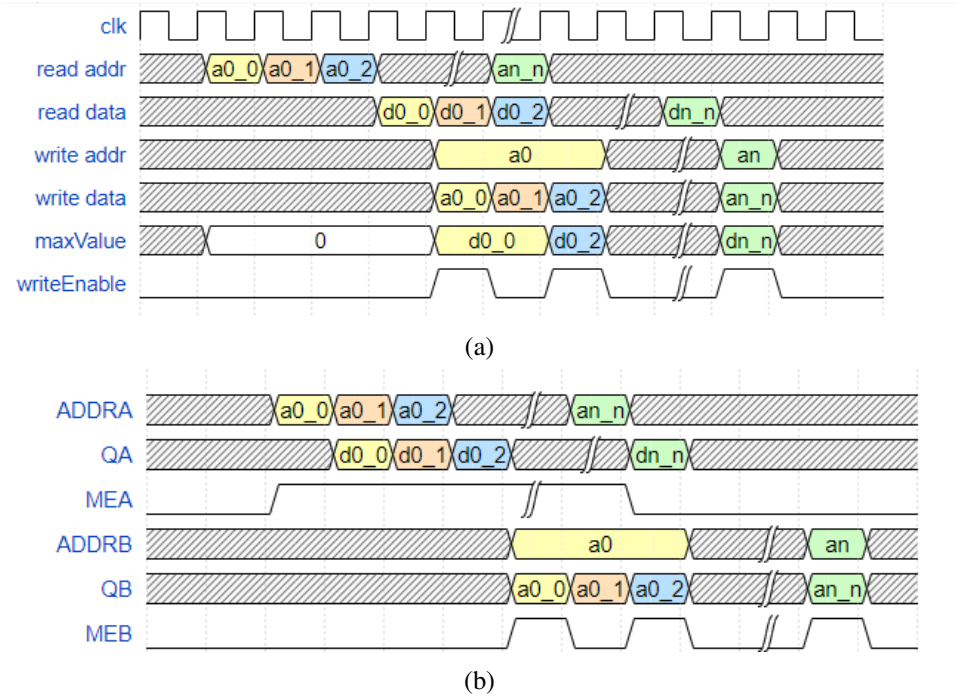


Figure 4.10: The timing diagram of peak detector: (a) shows the data sequence inside peak detector. The writeEnable signal change into 1 when the current read data is larger than the data that saved in ***maxValue*** register. (b) shows the according to data sequence in SRAM. Port ADDRA and QA correspond to read addr and read data separately, port ADDRB and QB correspond to write addr and write data seperately.

Result output part

This block works in state M5. It reads the saved result in SRAM and outputs them one by one.

4.4.2 SRAM structure

A single SRAM is used to save the histogram built during the execution. Each address in SRAM is corresponding to a histogram's bin. The detail about how SRAM works during the operation will be shown in Chap. 4.4.3. The structure of SRAM is shown in Fig. 4.11.

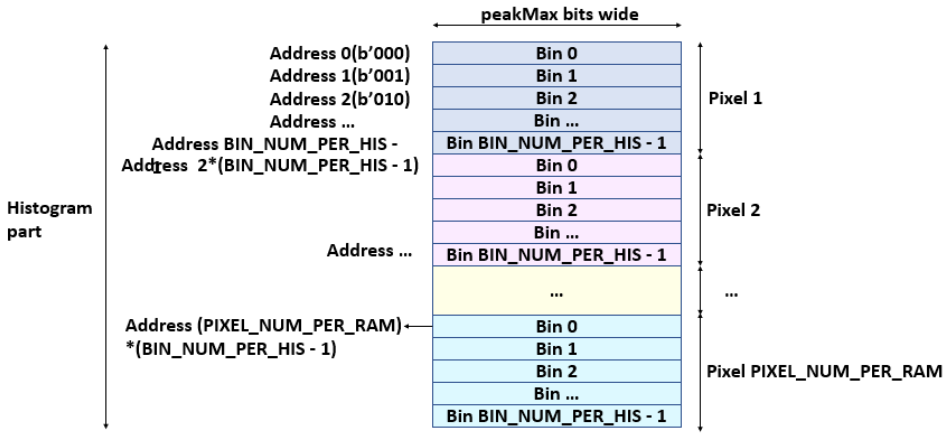


Figure 4.11: The memory location in SRAM. Each address (Bins) counts the number of times that this address (Bins) has been read.

Most SRAM modules do not have a reset signal. There are two different clearing approaches. One is straightforward: clear one address in one clk. SRAM has ***PIXEL_NUM_PER_RAM*** * ***BIN_NUM_PER_HIS*** addresses, hence this clearing mechanism introduces a latency ***PIXEL_NUM_PER_RAM*** * ***BIN_NUM_PER_HIS*** clks.

Another method is the signaled clearing mechanism, which does not involve latency because the histogram bins are initialized on the fly when needed. In this method, a special state status register is needed. The status of each bin in SRAM is presented by one bit in a status register. Suppose that the current 'addr' appears for the first time during the accumulation of a histogram, the 1 bit state bit for this address will be 0. In this case, the data in this address will be written as 1. Otherwise, the data in this address will be read, add 1, then write back. The drawback of this method is that a ***PIXEL_NUM_PER_RAM*** * ***BIN_NUM_PER_HIS*** bits register is needed, which is 200*160 bits typically, cost too much area. To solve this large register problem, work [40] introduced a latch-based reset block to save the area of the state status register. However, a latch is unstable and difficult to test, and it is hard to produce this idea as a product. On top of that, if the peak result is on-fly, it cannot reuse the histogram building's SRAM, causing extra area. Since my project targets a design that could be mass-produced, this latching method is not implemented. The straightforward reset method is chosen.

A brief calculation of ***SiFH***'s SRAM size is shown below. Since ***FiFH*** has half number of bins compared to ***SiFH***, ***FiFH***'s estimated SRAM size will be half on ***SiFH***. All the size memory blocks (SRAM and registers) are in bits during the calculation. Assuming the size of a single SRAM is S_{single} bits. The area of a single

SRAM could be calculated with the following equation:

$$S_{single} = \textbf{\textit{peakMax}} * \textbf{\textit{BIN\_NUM\_PER\_HIS}} * \textbf{\textit{PIXEL\_NUM\_PER\_RAM}} \quad (4.2)$$

Using the parameter shown in Tab. 2, we can get the size of all SRAMs (S_{SiFH}):

$$S_{SiFH} = S_{single} * \textbf{\textit{SRAM\_NUM}} = 2.44MB$$

The SRAM size of ***FiFH*** is half of ***SiFH*** because the ***BIN_NUM_PER_HIS*** is halfed.

From the calculation, the bits of SRAM for ***SiFH*** is 64 times the bits of SRAM for ***ezPointer***. However, in reality, this ratio will be smaller since the area of SRAM have not only the memory elements, but also the decoding part.

4.4.3 The iFH algorithm's execution example

A block diagram that describe ***SiFH***'s data flow brefly is shown in Fig. 4.12 . The overall design follows the timeline, the first and second histograms are built based on the first and second measurements separately. The timestamps from TDC range from 0 to 2^{N_p} - 1. The upper bound of this range presents that this data is not detected successfully and should be neglected. The data filter of the first histogram keep the first Nb bits of timestamps as 'addr'. 'addr' corresbounds to the bin number of first histogram, which indicates the address of SRAM that save histogram's bin counts. When there are no new timestamps, the peak detector outputs the value based on which the second histogram's filter is being built.

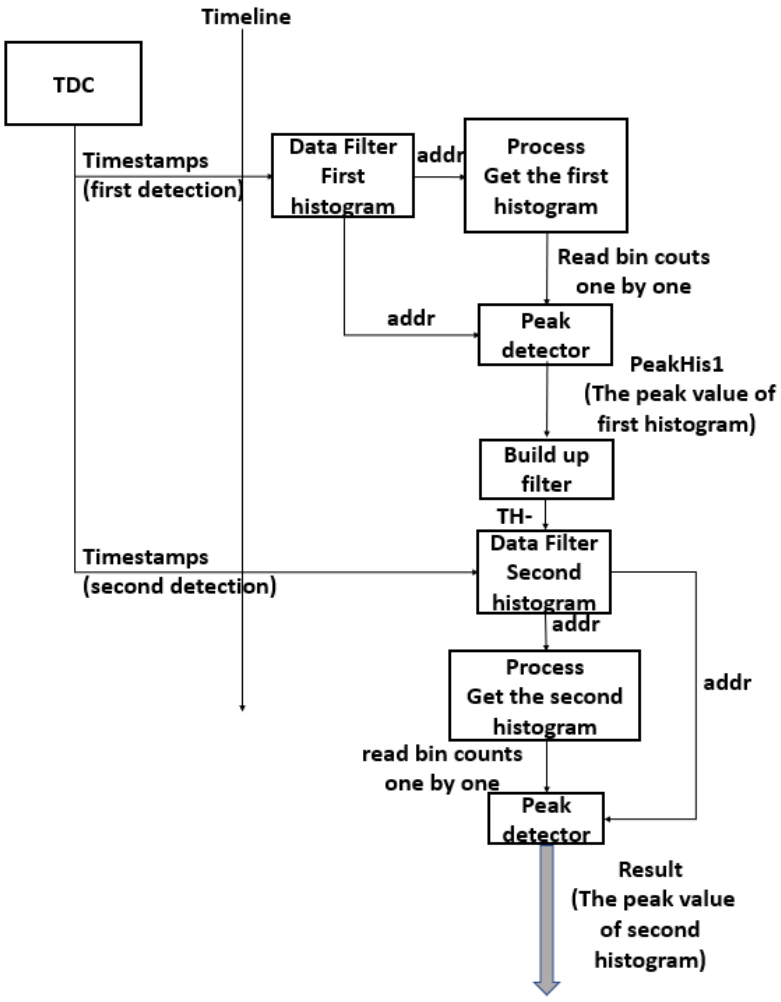


Figure 4.12: The block diagram of implementation without step-by-step methodology: The peak value will be find until all the input of one single measurement is being processed. 'TH-' is the lower bound of the second histogram's filter, calculated using the peak value of the first histogram. The upper bound of filter will be calculated accordingly.

An example about how the ***SiFH*** algorithm works is shown in Fig. 4.13. The result will be saved in the first address of SRAM after the histogram of all pixels is built. The data used in this example are listed in Tab. 4. In the example, 15(b'1111) means invalid detection, will be deleted directly.

The peak result of first measurement is 5(b'101), according to Eq. 2.6, the

Table 4: Example timestamps for ***SiFH*** design

measurement 1	Pixel 1 Acq1 data 1	Pixel 1 Acq1 data 2	Pixel 1 Acq2 data 3	Pixel 1 Acq2 data 4
Input(rough data)	11(b'1011)	9(b'1001)	11(b'1011)	15(b'1111)

measurement 2	Pixel 1 Acq1 data 1	Pixel 1 Acq1 data 2	Pixel 1 Acq2 data 1	Pixel 1 Acq2 data 2
Input(rough data)	10(b'1010)	10(b'1010)	15(b'1111)	11(b'1011)

according filter range from 7 to 14, 14 means invalid input. The calculation details are shown as below:

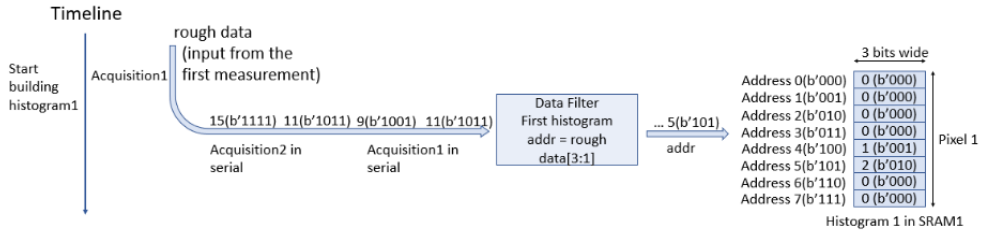
$$SB = 2^{N_b-1} = 4$$

$$TH_- = 2^{N_p-N_b}P_{CH} + SB = 2*5+4 = 14$$

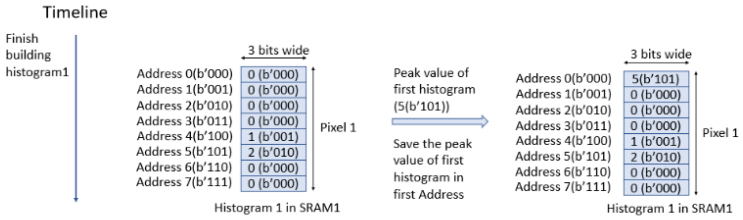
$$TH_+ = 2^{N_p-N_b}P_{CH} - SB = 2*5-4 = 6$$

$$6 = TH_- < pixel\ value \leq TH_+ = 14$$

The process of the second histogram is shown in Fig. 4.14. The whole SRAM is cleared before building the second histogram. Still, 15(b'1111) means invalid detection and will be directly deleted.

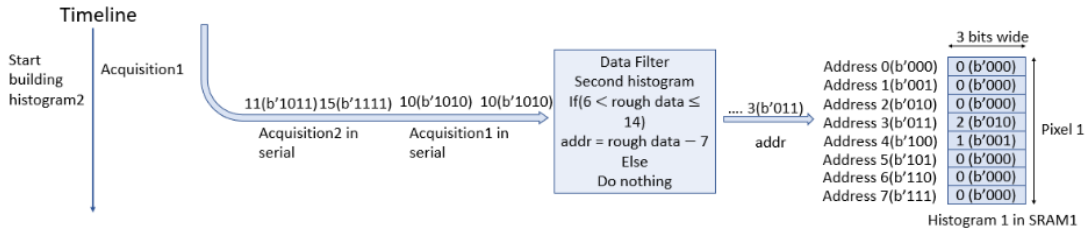


(a)

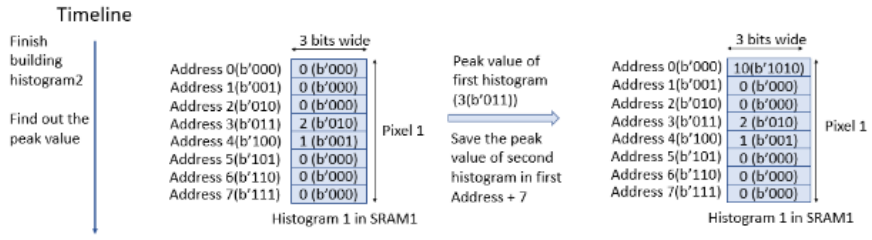


(b)

Figure 4.13: Presentation about the data process of the first measurement in step-by-step implementation: (a) shows the detail when the data of the first measurement is processed. SRAM is reseted before all the operations. No other operation occurs before all the input data in the first measurement are counted in SRAM (b) shows the detail when the address of the max value saved in SRAM is saved in SRAM's Address 0.



(a)



(b)

Figure 4.14: Presentation about the data process of the second measurement in step-by-step implementation: (a) shows the detail when the data of the second measurement is processed. (b) shows the detail when the address of the max value saved in SRAM is saved. (a) and (b) in the second measurement is the same as what that done in the first measurement, except a new filter for input data is used.

4.5 The pointer algorithm with histogram (hisPointer)

The ***hisPointer*** algorithm combined the method of ***iFH*** and ***ezPointer*** algorithm. As ***iFH***, the ***hisPointer*** algorithm need two measurements. The first measurement will build up the data filter for the second measurement. For ***hisPointer*** algorithm, the timestamps in second measurement will be processed by the ***ezPointer*** algorithm.

4.5.1 hisPointer block diagram

The schematic of ***hisPointer*** algorithm is shown in Fig. 4.15. The state machine diagram of ***hisPointer*** algorithm's controller is the similar as ***iFH***'s controller, shown in Fig. 4.8. The M3 and M4 state is changed from building up the second histogram to the ***ezPointer*** algorithm.

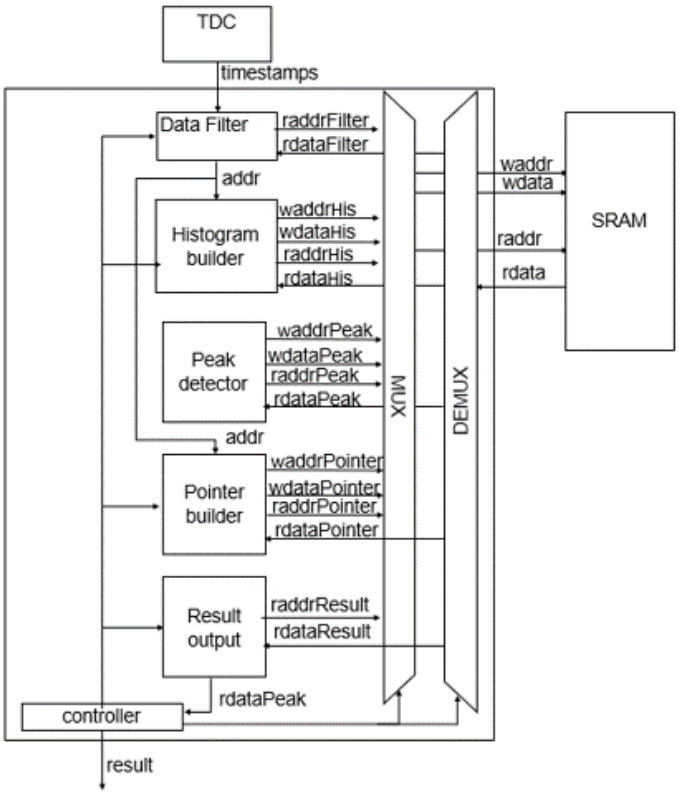


Figure 4.15: The schematic of hisPointer algorithm.

4.5.2 Expected area difference between Folded inter-Frame Histograms and the pointer algorithm with histogram

The SRAM usage of hisPointer algorithm is the same as the ***FiFH*** algorithm. The only difference is that an extra **pointer builder** part is implemented. In ***hisPointer*** algorithm, the peak detector and histogram builder block will not be used twice. Compared to the ***FiFH*** algorithm, a slightly higher area cost is expected.

5 Evaluation

In this chapter, the four algorithms implemented in RTL will be evaluated.

5.1 Tools and technologies

This thesis targets on 40nm integrated circuit (IC). Synopsis's design compiler version L-2016.03-SP3 is used to simulate timing, area and do the constrain test. The intellectual property (IP) core used for SRAM is the "SiWare Dual Port High-Density Leakage Control SRAM 512K Sync" provided by Synopsis, generated by Embed-it! (R) Integrator.

5.2 Performance criteria

Based on the simulation, the latency of a single detection for all implemented algorithms is 1.06×10^7 to 1.08×10^7 clock cycles per measurement, which is around 32.4ms for our 330MHz clock frequency. For a 30 frame/s system, the maximum latency for each measurement is about 33ms, longer than the latency reached by all algorithms.

Since all the algorithms can easily reach the latency requirement, latency is not one of the criteria of our design. The power cost is not one of the criteria of this design due to two reasons: The first one is that we can only try to push the energy cost as small as possible, and the simulation result in any compiler on power consumption is not accurate. On top of that, the SRAM for all algorithms uses the same technology currently. Hence their power consumption could be estimated based on the size of SRAM. The lowest power of SRAM and how the power consumption should be covered in mobile phones or AR systems is a complicated topic for future work.

Overall, we chose area and accuracy as our criteria to evaluate the design's performance.

5.3 Comparison

For the four algorithms that mentioned in this thesis, ***ezPointer*** and ***hisPointer*** are the original methods introduced by this thesis, ***SiFH*** and ***FiFH*** are the algorithms that introduced by other paper.

Tab. 5 shows algorithm's area simulation result. The ***ezPointer*** algorithm has apparently smaller area than the other three algorithms that based on histogtam building. The ***ezPointer*** algorithm even only cost less than 10% of ***SiFH***'s area.

Table 5: Simulation result on different algorithm's area cost

algorithm	***ezPointer***	***SiFH***	***FiFH***	***hisPointer***
area (mm^2)	1.15	15.46	11.46	11.48

Overall, ***ezPointer*** shows apparently superior performance because of much smaller area cost and second nice accuracy. For the typical and area important design, ***ezPointer*** is no doubt the best solution. For the design where relative precision has the highest priority, ***hiPointer*** is the best choice because the best relative precision and a similar area cost as ***FiFH***. Because of the limited area in mobile phones and AR applications, ***ezPointer*** is chosen as the best algorithm in this thesis's system.

6 Conclusions

This chapter concludes this thesis work. It addressed the thesis's research questions and gave some insights about future developments based on the thesis result.

6.1 Addressing the research questions

The research questions that listed in Chap. 1.1 were:

1. Can algorithms used in dToF systems with small dimensions, e.g., 32×32 or 64×64 pixels, be scaled to a 200×160 array?
2. Are there ways to process TDC's timestamps without saving all timestamps into on-chip memory?
3. Is there a feasible hardware implementation of large pixel array dToF algorithms able to meet the specific requirements of mobile phone applications?

Can algorithms used in dToF systems with small dimensions, e.g., 32×32 or 64×64 pixels, be scaled to a 200×160 array?

Both ***FiFH*** and ***SiFH*** meet the requirements on relative precision for indoor application. However, they will require silicon area of 11.46 and 15.46 mm^2 respectively. In industry's production, the area influence the cost directly. Such a huge cost cannot be tolerate. On top of that, ***FiFH*** and ***SiFH*** have lower relative precision than ***ezPointer*** algorithm, which requires only $1.15mm^2$ of silicon area.

Are there ways to process TDC's timestamps without saving all timestamps into on-chip memory?

Yes, the ***ezPointer*** algorithm is able to do this, in addition it has a superior area costs.

Is there a feasible hardware implementation of large pixel array dToF algorithms able to meet the specific requirements of mobile phone applications?

The ***ezPointer*** algorithm that was introduced in this thesis is sufficient for AR or mobile phone applications, also superior in area cost and relative precision. However, it still cannot meet our requirements at 50klux, which is the worst-case scenario for outdoor applications. Hence the 200*160 pixels dToF system could only be used in indoor applications.

6.2 Future work

Since the large pixel array system cannot be used in outdoor applications, it is necessary to test the maximum number of pixels in the array that could work at 50klux. While thorough research and simulation with a large pixel array were done, this study was beyond the scope of this thesis.

As mentioned in Chap. 3.4, the reason for high relative precision is low SNR, which could be weakened by increasing the power that each SPAD could capture. Since the application has a limited area, it is impossible to increase the laser power drastically.

There are two solutions to increase the laser power. One is decreasing the number of SPAD sensors that work simultaneously. Hence the power that is averaged for each SPAD is more considerable. The other one is using diffraction optics element (DOE) technology. DOE is flat, window-like optical components that are used to shape laser light [41]. Typically, the SPADs will be separated into x parts. Each part has one SPAD that will receive all the reflected waves in this part. Using DOE, the reflected wave of one part could all be gathered at one SPAD, hence increasing the SNR. With more timestamps and an unchanged noise rate, the SNR of the system will be increased, with the cost of pixel number.

After testing, if we reduced the size of the array by a factor of ten, we were able to make the design work at 50klux. However, this introduced another problem: how can we convert the small pixel array's outputs into the dimensions of the large pixel array, e.g., 200×160? Machine learning techniques, could be a possible solution.

Moreover, since the number of the pixels in the array is no longer 200×160, all the algorithms in this thesis will have to be reevaluated.

Overall, there are mainly three challenges with future directions:

1. Investigate the maximum number of simultaneously working sensors that can meet the AR or mobile phone application requirements in outdoor scenarios with up to 50klux;
2. Compare the algorithms used in this work for a small pixel array sizes working at 50klux;
3. Propose machine learning methods to enhance the results of small dimension arrays to a pixel array with 200×160 dimensions.

Another part of future work is the improvement of the ***ezPointer*** algorithm. A decimal pointer could be used to get higher precision. Also, we could try to output a confidence ratio for the pointer algorithm, then increase the available data presented by the pointer algorithm.

Bibliography

[1] M. Beer, O. M. Schrey, J. F. Haase, J. Ruskowski, W. Brockherde, B. J. Hosticka, and R. Kokozinski, "SPAD-based flash LiDAR sensor with high ambient light rejection for automotive applications," in *Quantum Sensing and Nano Electronics and Photonics XV*, M. Razeghi, G. J. Brown, J. S. Lewis, and G. Leo, Eds., vol. 10540, International Society for Optics and Photonics. SPIE, 2018, pp. 320 – 327. [Online]. Available: https://doi.org/10.1117/12.2286879

[2] C. Niclass, M. Soga, H. Matsubara, M. Ogawa, and M. Kagami, "A 0.18µm cmos soc for a 100m-range 10fps 200×96-pixel time-of-flight depth sensor," in *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, 2013, pp. 488–489.

[3] S. Lindner, C. Zhang, I. M. Antolovic, M. Wolf, and E. Charbon, "A 252 × 144 spad pixel flash lidar with 1728 dual-clock 48.8 ps tdcs, integrated histogramming and 14.9-to-1 compression in 180nm cmos technology," in *2018 IEEE Symposium on VLSI Circuits*, 2018, pp. 69–70.

[4] C. Zhang, S. Lindner, I. M. Antolović, J. Mata Pavia, M. Wolf, and E. Charbon, "A 30-frames/s, 252 × 144 spad flash lidar with 1728 dual-clock 48.8-ps tdcs, and pixel-wise integrated histogramming," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 4, pp. 1137–1151, 2019.

[5] I. Vornicu, A. Darie, R. Carmona-Galán, and Rodríguez-Vázquez, "Compact real-time inter-frame histogram builder for 15-bits high-speed tof-imagers based on single-photon detection," *IEEE Sensors Journal*, vol. 19, no. 6, pp. 2181–2190, 2019.

[6] A. Carrio, J. Tordesillas, S. Vemprala, S. Saripalli, P. Campoy, and J. P. How, "Onboard detection and localization of drones using depth maps," *IEEE Access*, vol. 8, pp. 30 480–30 490, 2020.

[7] A. Carrio, S. Vemprala, A. Ripoll, S. Saripalli, and P. Campoy, "Drone detection using depth maps," 2018. [Online]. Available: https://arxiv.org/abs/1808.00259

[8] A. Ronchini Ximenes, "Modular time-of-flight image sensor for light detection and ranging a digital approach to lidar," Ph.D. dissertation, Delft University of Technology, 2019.

[9] R. Horaud, M. E. Hansard, G. Evangelidis, and C. Ménier, "An overview of depth cameras and range scanners based on time-of-flight technologies," *CoRR*, vol. abs/2012.06772, 2020. [Online]. Available: https://arxiv.org/abs/2012.06772

[10] D. Bronzi, Y. Zou, F. Villa, S. Tisa, A. Tosi, and F. Zappa, "Automotive three-dimensional vision through a single-photon counting spad camera," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 3, pp. 782–795, 2016.

[11] I. Maksymova, C. Steger, and N. Druml, "Review of lidar sensor data acquisition and compression for automotive applications," *Proceedings*, vol. 2, no. 13, 2018. [Online]. Available: https://www.mdpi.com/2504-3900/2/13/852

[12] A. R. Ximenes, P. Padmanabhan, M.-J. Lee, Y. Yamashita, D. N. Yaung, and E. Charbon, "A 256×256 45/65nm 3d-stacked spad-based direct tof image sensor for lidar applications with optical polar modulation for up to 18.6db interference suppression," in *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*, 2018, pp. 96–98.

[13] Apple Inc., Cupertino. [Online]. Available: https://www.apple.com/

[14] C. Zhang, "Cmos spad sensors for 3d time-of-flight imaging, lidar and ultra-high speed cameras," 2019.

[15] A. Tontini, L. Gasparini, and M. Perenzoni, "Numerical model of spad-based direct time-of-flight flash lidar cmos image sensors," *Sensors*, vol. 20, no. 18, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/18/5203

[16] P. Padmanabhan, C. Zhang, and E. Charbon, "Modeling and analysis of a direct time-of-flight sensor architecture for lidar applications," *Sensors*, vol. 19, no. 24, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/24/5464

[17] R. Thakur, "Scanning lidar in advanced driver assistance systems and beyond: Building a road map for next-generation lidar technology," *IEEE Consumer Electronics Magazine*, vol. 5, no. 3, pp. 48–54, 2016.

[18] S. Lindner, C. Zhang, I. M. Antolovic, M. Wolf, and E. Charbon, "A 252 × 144 spad pixel flash lidar with 1728 dual-clock 48.8 ps tdcs, integrated histogramming and 14.9-to-1 compression in 180nm cmos technology," in *2018 IEEE Symposium on VLSI Circuits*, 2018, pp. 69–70.

[19] B. Aull, A. Loomis, D. Young, R. M. Heinrichs, B. Felton, P. J. Daniels, and D. J. Landers, "Geiger-mode avalanche photodiodes for three-dimensional imaging," 2002.

[20] R. Horaud, M. Hansard, G. Evangelidis, and C. Menier, "An overview of depth cameras and range scanners based on time-of-flight technologies," 12 2020.

[21] E. Charbon, "Single-photon imaging in complementary metal oxide semiconductor processes," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 372, no. 2012, p. 20130100, 2014.

[22] R. Horaud, M. Hansard, G. Evangelidis, and C. Ménier, "An overview of depth cameras and range scanners based on time-of-flight technologies," *Machine vision and applications*, vol. 27, no. 7, pp. 1005–1020, 2016.

[23] A. Gupta, A. Ingle, and M. Gupta, "Asynchronous single-photon 3d imaging," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 7909–7918.

[24] W. Becker, *Advanced time-correlated single photon counting techniques*. Springer Science & Business Media, 2005, vol. 81.

[25] N. A. W. Dutton, S. Gnecchi, L. Parmesan, A. J. Holmes, B. Rae, L. A. Grant, and R. K. Henderson, "11.5 a time-correlated single-photon-counting sensor with 14gs/s histogramming time-to-digital converter," in *2015 IEEE International Solid-State Circuits Conference - (ISSCC) Digest of Technical Papers*, 2015, pp. 1–3.

[26] A. T. Erdogan, R. Walker, N. Finlayson, N. Krstajic, G. O. S. Williams, and R. K. Henderson, "A 16.5 giga events/s 1024 × 8 spad line sensor with per-pixel zoomable 50ps-6.4ns/bin histogramming tdc," in *2017 Symposium on VLSI Circuits*, 2017, pp. C292–C293.

[27] A. K. Sharma, A. Laflaquière, G. A. Agranov, G. Rosenblum, and S. Mandai, "Spad array with gated histogram construction," Apr. 14 2020, uS Patent 10,620,300.

[28] H. Li, H. Zhang, X. Guo, and G. Hu, "Image restoration after pixel binning in image sensors," *Tsinghua Science and Technology*, vol. 14, no. 4, pp. 541–545, 2009.

[29] Z. Zhou, B. Pain, and E. Fossum, "Frame-transfer cmos active pixel sensor with pixel binning," *IEEE Transactions on Electron Devices*, vol. 44, no. 10, pp. 1764–1768, 1997.

[30] T. Aach, U. W. Schiebel, and G. Spekowius, "Digital image acquisition and processing in medical x-ray imaging," *Journal of Electronic Imaging*, vol. 8, no. 1, pp. 7 – 22, 1999. [Online]. Available: https://doi.org/10.1117/1.482680

[31] S. W. Hutchings, N. Johnston, I. Gyongy, T. Al Abbas, N. A. W. Dutton, M. Tyler, S. Chan, J. Leach, and R. K. Henderson, "A reconfigurable 3-d-stacked spad imager with in-pixel histogramming for flash lidar or high-speed time-of-flight imaging," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 11, pp. 2947–2956, 2019.

[32] Z.-W. Pan, H.-L. Shen, C. Li, S.-J. Chen, and J. H. Xin, "Fast multispectral imaging by spatial pixel-binning and spectral unmixing," *IEEE Transactions on Image Processing*, vol. 25, no. 8, pp. 3612–3625, 2016.

[33] A. Gallivanoni, I. Rech, and M. Ghioni, "Progress in quenching circuits for single photon avalanche diodes," *IEEE Transactions on Nuclear Science*, vol. 57, no. 6, pp. 3815–3826, 2010.

[34] A. Puszka, L. D. Sieno, A. D. Mora, A. Pifferi, D. Contini, G. Boso, A. Tosi, L. Hervé, A. Planat-Chrétien, A. Koenig, and J.-M. Dinten, "Time-resolved diffuse optical tomography using fast-gated single-photon avalanche diodes," *Biomed. Opt. Express*, vol. 4, no. 8, pp. 1351–1365, Aug 2013.

[35] W. Jiang, Y. Chalich, R. Scott, and M. J. Deen, "Time-gated and multi-junction spads in standard 65 nm cmos technology," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 12 092–12 103, 2021.

[36] J. H. Lambert, *Photometria*, 1760.

[37] B. V. Ricketti, E. M. Gauger, and A. Fedrizzi, "The coherence time of sunlight in the context of natural and artificial light-harvesting," *Scientific Reports*, vol. 12, no. 1, pp. 1–9, 2022.

[38] J. C. Cooper, "The poisson and exponential distributions," *Mathematical Spectrum*, vol. 37, no. 3, pp. 123–125, 2005.

[39] B. Li, L. Xia, P. Gu, Y. Wang, and H. Yang, "Merging the interface: Power, area and accuracy co-optimization for rram crossbar-based mixed-signal computing system," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6.

[40] I. Vornicu, A. Darie, R. Carmona-Galán, and Rodríguez-Vázquez, "Tof estimation based on compressed real-time histogram builder for spad image sensors," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2019, pp. 1–4.

[41] H. P. Herzig, *Micro-optics: elements, systems and applications*. CRC Press, 1997.