웹캠과 적외선 LED를 이용한 카메라 광통신 시스템 개발

Development of an Optical Camera Communication System

Based on a Webcam and Infrared LEDs

2019년 2월 25일

조선대학교 대학원

전자공학과

송 범

웹캠과 적외선 LED를 이용한 카메라 광통신 시스템 개발

지도교수 이 충 규

이 논문을 전자공학 석사학위신청 논문으로 제출함

2018년 12월

조선대학교 대학원

전자공학과

송 범

송 범의 석사학위논문을 인준함

위원장 조선대학교 교 수 황 석 승 (인)

위 원 조선대학교 부교수 오 순 수 (인)

위 원 조선대학교 교 수 이 충 규 (인)

2018년 11월

조선대학교 대학원

Contents

List of Figures

List of Tables

초록

웹캠과 적외선 LED 를 이용한 카메라 광통신 시스템 개발

송 범

지도교수: 이충규

조선대학교 일반대학원 전자공학과

통신 채널의 단점을 해결하기 위해 최근 몇 년간 무선 광통신이 널리 보급되어왔다. 본 논문에서는 USB 웹 카메라와 적외선 LED 를 이용한 실시간 광학 카메라 통신 시스템을 소개한다. 본 시스템은 자동으로 광원을 찾고 데이터를 실시간으로 송·수신 및 출력 할 수 있다. 본 시스템의 송신기는 저전력 적외선 LED 배열로 구성되어 있고, 수신기는 USB 웹 카메라로 구성되어있다. 수신기 역할을 하는 USB 웹 카메라는 컴퓨터와 연결되어 있고, 컴퓨터는 이미지 처리 및 데이터 추출 작업을 수행한다. 실험 결과에 따르면 본 시스템의 통신 속도는 초당 10 자의 문자를 전송할 수 있고, 5m 거리에서 평균 1.08 % 오차율을 갖는다.

Chapter 1. Introduction

After nearly a hundred years of development, RF communication technology has made a great progress. Because there are more and more wireless communication devices we use, the available wireless communication frequency channels are getting crowded. Radio-frequency interference between channels makes the quality of communication worse, but also the design difficulty of wireless devices increases. In order to solve this problem, researchers have turned their attention to optical wireless communication systems. Optical wireless communication systems use light as a carrier for transmitting message signals, which have certain advantages over conventional wireless communication systems.

Thanks to the strong directivity of light's propagation, it can be easily blocked. That means it is more secure than a RF communication system. on the other hand, because different frequencies are used with conventional wireless communication systems, it is possible to avoid the interference with other electronic devices. Finally, due to very wide light bandwidth, optical frequency division multiplexing can be realized, and the transmission capacity can be increased.

Visible light communication system (VLC), as a kind of optical wireless communication system, is regarded as a promising technology in the future, and has made great progress in the past century. It uses the visible light as a communication carrier and high-speed photodiodes (PDs) or solar cells is used as the signal detection devices. Because the response time of the photodiode usually a sub-ns [1], the system can communicate at high data rate over several Gb/s, in the latest study, some VLC systems can communicate at data speeds of dozens of Gbps[2, 3]. Some technical features of VLC system were fully described concerning the PHY and MAC layers in IEEE 802.15.7 in 2011[4].

Over the past decade, due to the development of mature complementary metal-oxide-semiconductor (CMOS) technologies, the built-in cameras of smartphones have achieved high resolution and high speed than ever before. Because of its low price and power consumption, the CMOS image sensor has been used as an image sensor for built-in cameras. It provides a new possibility for optical communication systems. Researchers have proposed to use CMOS image sensor as a receiver for optical communication system, which is called optical camera communication (OCC) system. The OCC system is a revised version of VLC and they share the same optical channel and the difference is that it uses the image sensor to capture light signals and some researchers believed that OCC system is much closer to the market than PD-based VLC system [5] Some researchers have also analyzed the possible applications of OCC systems in the 5G era [6]. In the OCC system, the transmitter can be a LED/LD, a screen or other light sources. Meanwhile, digital camera, USB webcam, or smartphone camera can be used as the optical receiver. After receiving the light signal, it can be stored as one frame of image. Compared with the photodiode-based VLC system, the OCC system can record not only frequency and brightness information, but also location information of the light source in the image. By processing image frames, we can easily locate and separate different light sources and signals. OCC technology inherits VLC by using an embedded camera without modifying hardware [7], but the performance of camera sensor can limit the performance of OCC system.

1.1 CMOS and Shutter Mode

Nowadays most camera use CCD or CMOS image sensor, and CMOS is more widely used because of its lower price and lower power consumption. CMOS image sensor is a two-dimensional array of light sensors and a received image is built row by row. In other words, the data is read in units of rows. The recording process of each row in one frame can be divided into four steps:

- Reset
- Exposure

- Charge transfer
- Data readout

Similar to traditional cameras, the exposure time in CMOS image sensor is controlled by a shutter mechanism called electronic shutter. According to the way of shutter operation, we can divide them into two shutter modes: rolling shutter and global shutter.

Rolling shutter and global shutter modes are referred to as readout modes in which CMOS image sensor is operated. Figure 1-1(a) describes the image buildup process using the rolling shutter mode. In the rolling shutter mode, each individual row is typically able to begin the next frame once completing the previous frame’s readout. Because there is a delay extant at the beginning of the exposure between row and row, if the light source`s flicker frequency is higher than the frame number of the camera, there will be black/white stripes on the screen, as shown in Figure 1-1(b). By analyzing the distribution of stripes in the image, we can get multiple bits of information in one frame. So, the data rate can higher than the refresh rate of image sensor. On the other hand, because of the limitation of optical resolution, communication distance of system using rolling shutter is usually within a very short distance. Some basic principles of using rolling shutters can be seen in Trong-Hop Do and Myungsik Yoo’s paper [8].

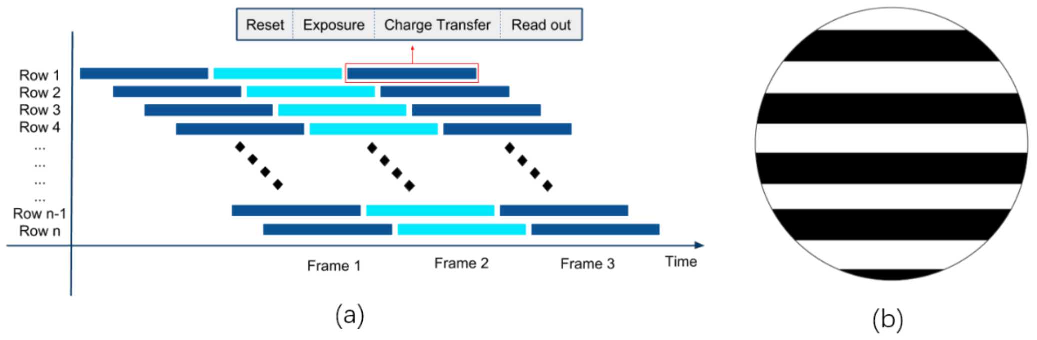


Figure 1-1. Rolling shutter (a). Diagram demonstrating the rolling shutter. (b). The stripe appeared on the image when the light source flickers at high frequencies.

Figure 1-2 describes the image buildup process of the global shutter mode. In the global shutter mode, all rows will exposure (be exposed) at the same time and have the same exposure time. Compared with the former, system which use global shutter mode always has a disadvantage in data rate. Because the sampling rate of the light source will not exceed the refresh rate of the CMOS image sensor. Because we are concerned about the overall brightness of the light source, rather than the internal light intensity distribution, it can be regarded as a bright spot on the screen. For this reason, with the same brightness and size of light source, the global shutter system can achieve a further communication distance.

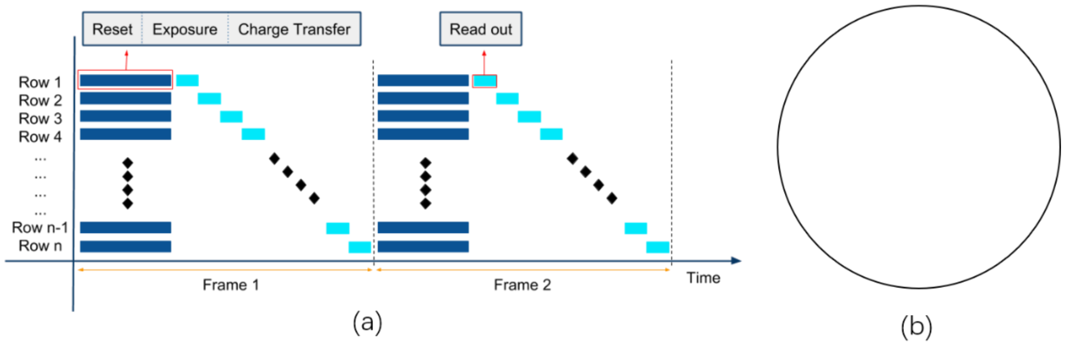


Figure 1-2. Global shutter (a). Diagram demonstrating the global shutter. (b). The image presented when the light source flashes at high frequencies.

1.2 Transmitter (LED/Display)

LED (Light-emitting diode) have grown fast over the past two decades. Compare with traditional light source, LED has many advantages, such as lifetime, energy efficiency and size. Nowadays there are more and more scenes of using LED as the light source in daily life. Meanwhile, LED is not only an environmentally friendly alternative to traditional light sources, but also a perfect optical signal source. Because the LED can change its brightness at a very fast frequency. After proper modulation, the LED can serve as a signal source to transmit information. In traditional VLC systems which using photon detectors as receiver, LEDs have been widely used as signal sources for optical signals [9, 10]. Due to the simple structure, the size of one LED can be made quite small. Also, thanks to the extremely low power consumption, we can use a small power supply to drive many LEDs simultaneously. In many practical usage scenarios, many LED can be integrated into one LED array and become a large light-emitting unit to obtain higher light intensity. Such as car lights and various traffic lights on the road. Meanwhile, to improve the bandwidth and signal-to-noise ratio, some study teams have proposed to use LED arrays as signal sources. Because the state of the individual LEDs in the array can be easily controlled, we can make different LEDs of the array stay in different states at the same time. while these LEDs can be treated as multiple transmitters that can enable visible light MIMO communication [11]. For these systems, the signal is not only related to the brightness of LED, but also to the spatial coordinates of LED lights.

Nowadays, most optical camera communication (OCC) systems use LED as the light source but are different in the specific forms. With respect to those systems which use monochrome LED as a light source, the ON or OFF states of an LED is used to represent the binary data stream. There are also systems that use the wavelength (or equivalently, color) division multiplexing, which uses the wavelength of each color in the RGB LED to enhance the communication capacity.

Besides the LED, the display screen is also a viable option for a communication light source. In a system that uses display screen as a light source, the modulating signal is represented by the color and intensity of parts of the pattern on the screen. Compared with the use of LED, because the display screen can achieve a more detailed display than an array of LEDs, each frame can be filled with more information. However, most commercial screens have a maximum refresh rate of 60 Hz, which will limit the bandwidth throughput of the system. Moreover, due to the limited resolution of the camera, systems using the display as the signal source usually have the disadvantage of communication distance.

Also, laser emitter is an option for optical signal sources. Compared with the former two, the use of laser as a signal transmitter optical communication system can achieve a longer communication distance. Some study team has already demonstrated a VLC link which has a communicate distance of more than 72m, this kind of system can used to build a communication link between buildings [12]. In general, systems using a combination of Laser-PD can achieve longer distance and higher speed communication [13 - 15]. Since the OCC system is different from the receiver of the VLC system using the PD as a receiver, most OCC systems do not use a laser emitter as an optical source.

1.3 Input/output and Modulation Techniques

Multiple input multiple output (MIMO) technology is an important direction for communication system development. Considering the limitation of camera sensor refresh rate (~ 60 Hz), to improve the system throughput, we hope to transmit more information in one frame. As with RF system, MIMO technology can increase the capacity of OCC system. There are many ways to realize multiplexing, while the OCC system often adopts color shift OOK (CSK) modulation and spatial coordinate modulation.

The former (CSK) uses the wavelength (color) information of the optical signal to identify the input of the signal. Thanks to the RGB LED, we have more choices on the wavelength of the signal source. The basic idea of CSK modulation is to modulate different signal sources by different color combinations and to demodulate the received signals according to the wavelengths at the receiving end. So that multiple sources can transmit data at the same time, for they occupy different channels respectively. A demonstration of 3 channels CSK modulation in shown in Figure 1-3.

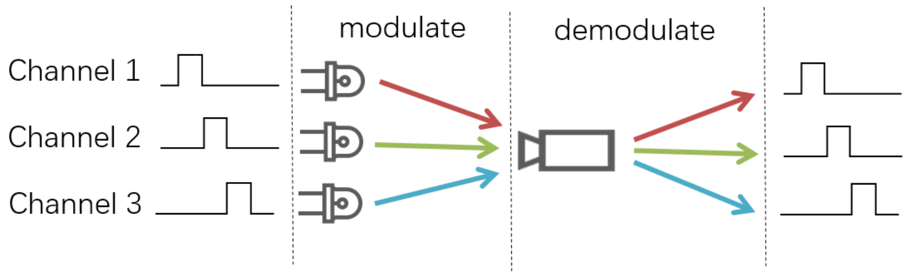


Figure 1-3. A demonstration of 3 channels CSK modulation

The latter (spatial coordinate modulation) uses the different location information of the signal source to distinguish different inputs. Because the image reflects the two-dimensional distribution of light information in space. Because the image reflects the two-dimensional distribution of light information in space, it also records the spatial distribution of light intensity while recording light intensity and wavelength information. The signal source and its spatial coordinates are combined to identify each signal source by coordinates, so as to achieve simultaneous multi-channel input. A demonstration of 4 channels spatial coordinate modulation in shown in Figure 1-4

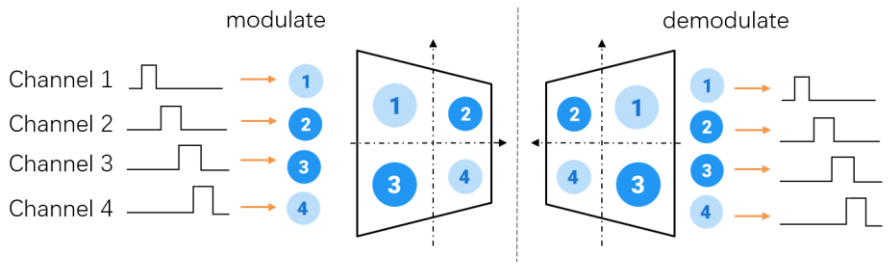


Figure 1-4. A demonstration of 4 channels spatial coordinate modulation

1.4 Source Positioning Algorithm

Because OCC system entails more directional transmission compared to VLC, source positioning algorithm is important point we need care about. For systems which need to be operated under rolling shutter mode, the light source is usually close to the receiver and the SNR is quite high. Furthermore, the relative position of light source and receiver is usually fixed. So, there is no need for a strong light source positioning capability. Consider of that, some indoor communication system chooses to locate the signal source by user manually [16].

For systems that need to work on high mobility, source positioning algorithms are critical. Because most of the time the light source is just a dot on the screen, and it moves constantly. Some OCC system use RGB LED as the light source, so they can locate the light source using color detection [17]. In some situations, image need to be processed grayed, so color detection does not work in all situations.

For dynamic positioning of the signal source, some study reported a locate system based on specific hardware. I. Takai and their team proposed a system using optical communication image sensor to locate all light source and the correct signal source is then filtered through the software [18]. Some research teams are trying to use software to complete the location. G. K. H. Pang and H. H. S. Liu use thresholding process to reduce noise and locate the LED panel use the row and column projections [19]. H. C. N. Premachandra, et al. found the transmitter by compare the similarities and differences between frames and frames [20]. While S. Arai and their team employ the block matching algorithm, which is a way of finding a corresponding position between two successive frames, for the proposed method [21]. This algorithm will divide the captured image into a number of small domains (blocks) and determines if the LED array is present or absent using the block matching.

1.5 Examples of OCC Systems

With respect to OCC system, many pioneers have proposed various ideas and practices. Although their structures and purpose are quite different, they can be classified into two categories according to the readout mode of the camera: rolling shutter and global shutter modes.

Rolling shutter-based OCC

Systems that use rolling shutter mode usually have higher data rate but shorter communication distances. At the same time, this method has lower lens requirements for the camera, and the LED-Embedded Camera combination has become a wide choice for people. C. Danakis and his team use a mobile phone`s camera to achieve a high data rate faster than the camera`s frame rate [22]. Data that needs to be transmitted will be packaged into blocks and then encoded by Manchester code. After that data will be transmitted using a surface illuminated by LED. The change of light intensity in the surface can reflect different state of data. Using a built-in camera, the mobile phones run in the rolling shutter mode to capture the change of state. Eventually, the rapidly changing of data state can lead to black and white stripes on the screen. By analyzing these stripes, the transmitted data can be recovered. The communication bandwidth of this system can reach 3.1 kbps, but the communication distance is only tens of centimeters. The same principle can also be seen in some other papers [23].

Because the data is presented by stripe image of the system using rolling shutter mode, the quality of the stripe image will affect the performance of the system directly. C. Chow and his team suggest processing the stripe image to improving bit error rate (BER) [24]. Same as C. Danakis, they use a built-in camera of mobile phone as the receiver, but the transmitter has no reflective surface, and light signal from the light source will be directly received by the image sensor. After receiving the signals, they first use a second-order polynomial fitting to mitigate the “blooming effect”, which can lead to a different width of the stripes. Then use histogram equalization and Sobel filter to process column matrix grayscale values, enhance signal performance. After those processing, the gray contrast increased from 35 to 175, resulting in a higher signal-to-noise ratio. The net data rate was 0.896 kbps and the communication distance are 25 cm in free-space.

T. Nguyen et al. discussed the suitable frequency range for OCC using rolling shutter camera by measuring the practical response of normal camera [25]. The frame structure of the existing asynchronous decoding data is compared, and a new frame structure is proposed. Because most of the OCC systems are unidirectional communication systems, in order to solve data loss during the camera exposure interval, they proposed the concept of a super frame. First, the data is packed with the asynchronous bit and the start frame as a data sub-frame. Then, system will repeat this data sub-frame several times and packs it into a super frame. There is a proportional relationship between the number of sub-frames in a super frame and the refresh rate of the camera. With the aim of improving the data rate without reducing the brightness contrast, different run-length limited (RLL) encoding schemes are analyzed. In the achieved system, when the LED's pulse rate is 2 kHz, the throughput of entire link can reach 1.9 kbps while the data rate is up to 1.5 kbps.

Since the stripes on the screen when using the scrolling shutter are due to the time difference between the exposures of each part of the camera, the distribution of the stripes actually reflects the frequency information of the light source. If the frequencies used in the system are not the same, we can use the frequency characteristics to distinguish the light source currently used.

J. H. Kim proposed an indoor location system using OCC system operated under rolling shutter mode [26]. In the test environment, they set different flicker frequencies for LED light sources in different rooms. The receiver is a smartphone with specific software installed, and the software will analyze the frames received. After analyzing the results, they found that when the frequency of the light source was within the range of 100~2000 Hz, the number of stripes on the screen was proportional to the frequency. The exact number is related to the equipment, and the overall range is between 1 and 63. On the other hand, they also analyzed the relationship between distance and detection rate. If we use the detection accuracy of 20 cm as the baseline, when the distance is 80 cm, the detection rate is reduced to less than 70 %. In fact, the detection rate is related to the frequency of the light source, and the high frequency light source will lose more detection rate than the low-frequency light source at the same distance. In order to ensure the efficiency of the system, they limit the detection distance to 20-70 cm, while the frequency of the light source is limited to 100~1800 Hz. There are many papers that use the same principle to achieve indoor positioning [27].

Global shutter-based OCC

Compared to the systems mentioned above that use a rolling shutter, the system using the global shutter has a better performance in terms of communication distance. Global shutter mode often used in the system which needs to process image of each frame.

P. Luo et al. proposed a non-flicking OCC system which using two monochrome LED lamps with an alternative modulation scheme of undersampled phase shift ON-OFF keying (UPSOOK) as the signal source [28]. In the UPSOOK system, the transmitter can indicate three different states by changing the blinking frequency and the pulse phase of the LED light source. Similar methods include Undersampled frequency shift on-off keying (UFSOOK) [29], undersampled differential phase shift on-off keying (UDPSOOK) [30] and undersampled QAM subcarrier modulation (UQAMSM) [31]. If we use the light intensity received by the receiver when the LED is on continuously as the baseline. When the LED flashes at high frequencies, the intensity of the light signal received by the receiver will be 0.5. Since the LED is only turned on for half the time during the receiver sampling/exposure. This is the first state, called "HALF ON". The other two states correspond to “0” and “1” respectively. In these two states, the system will drive the LED flicker with a square wave of the same frequency, but the phase between them will be 180 degrees out of phase. Compared with "HALF ON", the blinking frequency is much lower at this time to ensure that the status of the LED will not change or only change once during sampling. The three states can have 9 combinations and using eight of them can establish a mapping relationship to all 3-digit binary numbers. In that case we can get 3 bits of data in one frame, and the data rate is 3 times of the camera's frame rate. The experiment results show that the proposed camera communication system can achieve 150 bps error-free communications for a range up to 12 m. In another paper, the team also demonstrated how to use RGB-LEDs as a light source and use different colors to communicate on different channels. The experimental results show that the proposed system using a single commercially available RGB LED and a standard 50-frame/s camera is able to achieve a data rate of 150 bits/s over a range of up to 60 m [32]. Similarly, they also discussed the application of UPSOOK in v2v communication [33].

Some research groups choose color-shift keying modulation to map the different combination of bits into different colors.   System usually has only one light source which made of RGB-LED as the transmitter. By adjusting the light power of red, green, and blue in the light source, we can get different colors of light, and they correspond to different binary data respectively. Since the range of light power of each color is 0~255, it can even be used to realize CDMA modulation by selecting the appropriate offset. In the system proposed by S. Chen and C. Chow, 4 different colors are used to correspond to all 2-digit binary number [34]. Using CDMA to modulate the light power of red, green, blue, it allows 2 users to access simultaneously.

Like the RF communication, we can establish a MIMO system by dividing the frequency range to create different channels. Because different frequencies of light will have different colors, we can divide channels by color of the light. In this scenario, there will be several light sources as the transmitters, different color channels are chosen to avoid inter-channel interference.

Those systems can communicate over long distances, but the communication rate is not fast. K. Liang, C. W. Chow, and Y. Liu proposed an OCC system use a CMOS camera running at rolling shutter mode as the receiver, and a block of RGB LED as the transmitter [35]. They use 3 color channels to transmit data at the same time, with a 30-fps refresh rate camera, the data rate of their system can reach 2.88 kbps.

In addition to the LED, display is also a viable option and many scholars have studied this aspect [36 - 39]. For example, Y. Zeng et al. proposed a VLC system based on mobile devices, data were converted to screen code before transmission and the display screen were used as transmitter [40]. Using this system, they can reach 80 Mbps theoretical in a short distance.

N. Trang, N. T. Le, and Y. M. Jang proposed a screen-to-camera based optical camera communication [41]. They used an LCD monitor made up of 256 RGB-LEDs and use RGB value to drive the LEDs simultaneously. With a screen refresh rate of 20, their communication rate can reach 15 kbps. Screen to camera system can transmit data at a very high rate, but usually the transmission is very close, since the camera's optical resolution limits the pattern recognition.

Chapter 2. Proposed System

The starting point for the designing this communication system is for the communication between vehicles and vehicles(V2V). Because the V2V communication system has the characteristics of short communication distance and high density, people have great expectations for the application of optical communication in this field. The use of optical communication systems can avoid already crowded radio frequencies. And because of the directivity of the optical link, it is easy to remove interference from other directions.

Our design goal is to build an economical, lower energy consumption, real-time OCC system with high reliability. Even if the source location is uncertain, positioning can be performed automatically and as little as possible of any signal loss of the frame. Also, this system should not disturb or harm the driver. To sum up, the purpose and requirements of this system are:

1. Use optical communication link
2. Lower energy consumption
3. Mobility
4. Display results in real time
5. Can locate signal source automatically
6. Have communication capability in short-medium distance
7. Harmless to human

Based on the above considerations, I designed an optical camera communication system using infrared rays. This system consists of two parts, transmitter and receiver, as shown in Figure 2-1.

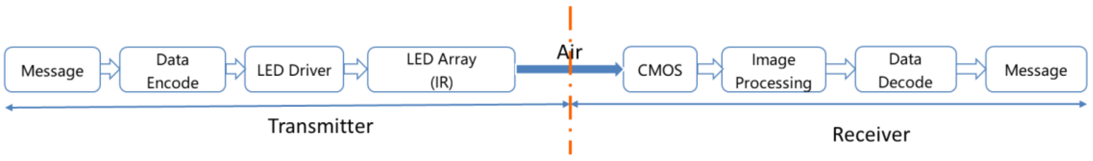


Figure 2-1. Block diagram of system

2.1 Transmitter

The transmitter is responsible for processing the message to be sent and presenting it to the receiver through the LED array. Because the infrared light is outside the visible light range, that is, humans cannot see the infrared light, so it does not interfere with the driver. on the other hand, because infrared light can help the camera during imaging, most of the Digital cameras we use can detect infrared signal. Because it is economical, low energy consumption and has a long service life, IR-LED had been widely used in wireless communication in the last decade. With the same consideration, infrared LED is a good choice for this communication system.

Because the refresh rate of the camera is usually between 30 and 60 Hz, in OCC systems, it is common to choose to transmit multiple bits of data in one frame to increase the transmission rate. To increase the data transfer speed as much as possible, we will use LED array to transmit the data. Also, the LED array will also help the receiver to locate the signal source. For these two purposes, LED array is divided into two major components: Position_LED and Data_LED, and there are 2 separate spaces between different areas to separate them. As the name saying, LEDs in the Position_LED will help the receiver to locate the LED array. They will keep lighting on when the transmitter is transmitting data. Data is presented at the area of Data_LED. Before sent to LED array, message will be encoded by a specific coding protocol in pre-processing progress. Different data will make the Data_LED shows different characteristics. Figure 2-2 is a demonstration when the data is “01011010” with 8 LEDs forms the Data_LED, P1~P8 indicate the position of the detection points.

In the array, the position of each LED needs to be accurately fixed under the specification, so that we can calculate the position of each point in the Data_LED by a simple mathematical operation after knowing the position of the center point of the Position_LED on both sides.

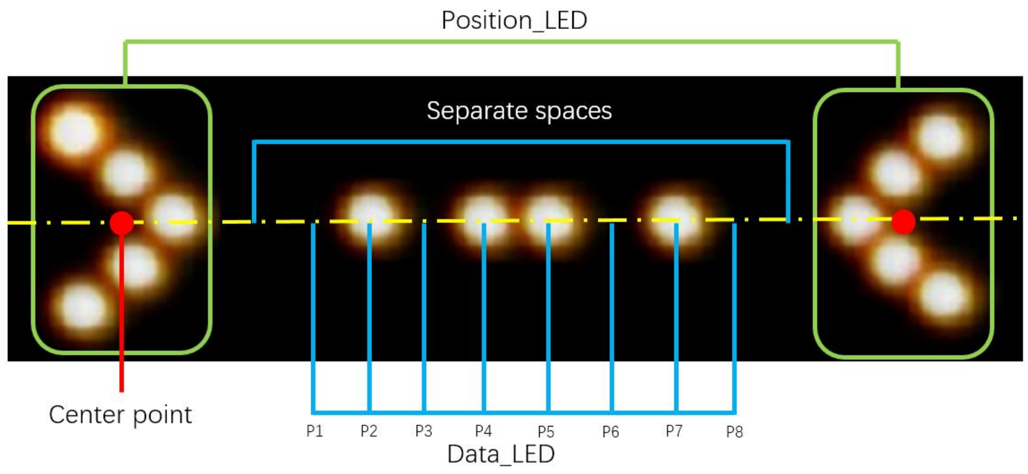


Figure 2-2. LED array of "01011010"

2.2 Receiver

Receiver will be responsible for data receiving and extracting. In this system, the signal sent by the transmitter will be received as an image, and image processing is an important step for the receiver.

Thanks to the development of CMOS, the web camera has achieved a great development in the past decade, we can get a high refresh rate camera at a very low price nowadays. For example, ELP-USBFHD01M, a middle level web camera on the market can generally achieve a refresh rate of 60 frames at 720p resolution. Its quality and economical can help us to set up an OCC system which can be used widely. Also, the CMOS web camera can detect infrared rays. So, the web camera is an ideal choice for the signal receiver.

After the camera captures the image, the software will process the image to extract the data in the image. As a consideration of time and my ability, I decided to use the OpenCV library to process the image. OpenCV is a mature computer version library which originally developed by Intel. This library is cross-platform and free for user under open-source BSD license. Under its help, we can use the existing implementation methods to achieve recognition and image processing, without having to start from scratch. What's more, it has a fully support of Python. We will use OpenCV to complete the image processing part, including camera access, target identification and data readout.

Unlike ordinary signal processing, image processing requires a lot of computational work, which means that image processing will take a long time. At the same time, because the processing time of each frame is different, if we just processed the incoming frame in order, system may miss some frame or lost synchronization. In order to achieve real-time communication and speed up data transmission, in the software system, the multi-process system structure will be used to process the frames in parallel. At the same time, the multi-process system can also control the processing time of each frame to prevent the system from being stuck on a certain frame.

2.3 Experimental System

In order to verify the feasibility of the system, I built an experimental system. In this experimental system, we will transfer eight bits of data at a time, which means that the Data_LED component will be set with eight IR-LEDs. We will try to transmit text message over a distance through this OCC system, and analyze the performance of this system based on the results.

Chapter 3. Software

Software is an important part of the system. In the transmitter, the processing of the input text and the driving of the LED will be controlled by a Raspberry Pi. Software system flow chart of transmitter in shown in Figure 3-1(a). After system receives input message, system will split them into separate characters. System will only transmit one character in one frame. After that, system will wait for a signal from synchronization control component to transmit the next character. In the next step, system will convert the character into an 8-bit binary number according to the protocol. Here we may call it the “eigenvalue” of this character. The Raspberry Pi will drive the LEDs through its GPIO port and put the LEDs in different states according to the eigenvalue.

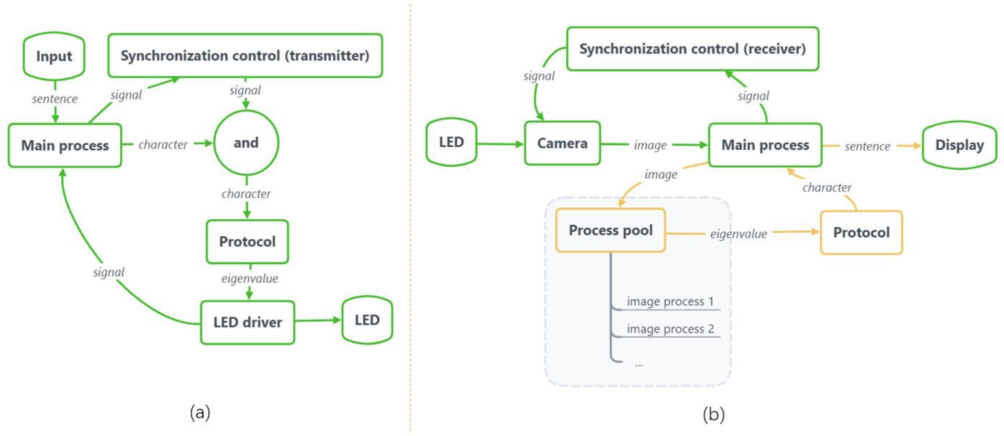


Figure 3-1. Software System Flow Chart

The control of the camera and the processing of the images are implemented through a script written in Python. Compared with the transmitter the software of the receiver is more complicated. Here are four important components, image processing, process pool, synchronization control and protocols.

After the camera captures an image, it will remain standby until the synchronization control gives it a signal to execute the next capture. The main process receives the image, it will perform preliminary processing, attach a current timestamp, and transfer the data to the process pool. The process pool will be responsible for process creation, exit and timeout detection. Each frame captured from the camera will be assigned a separate process for processing, and the system will implement timeout detection through the control of the process. Whenever a process returns a result or times out, the system will convert the result to a corresponding character according to the protocol. Then main process stores the returned characters, sorts them according to the timestamp, and print the results on the screen. The system flow chart at the receiver is shown in Figure 3-1(b).

3.1 Image Processing

Image processing is an important action at the receiver, because the signal sent by the LED array of the transmitter is received and stored by the receiver in the form of image. By analyzing and process image, we can restore the signal that the sender wants to send. The whole process of image processing is shown below (Figure 3-2).

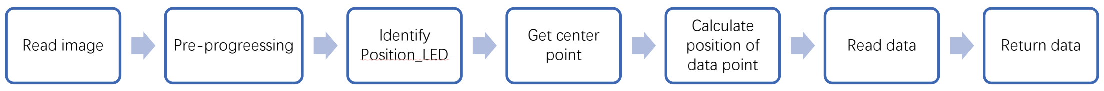


Figure 3-2. Flow chart of image process

After one process was created, it will receive one frame of image. After pre-processing, system will try to identify and locate Position_LED first. After some correction work, we can get the coordinates of Position_LED of the left and right parts. Next step is calculating the detect point of Data_LED according to the coordination we got from the former work. The system will evaluate the value of this point based on the brightness value of the detect point. This entire image processing program will be assigned a separate process to avoid the overall delay of the system due to possible timeout.

Pre-progressing

Because the received image may have defects such as noise and uneven brightness, and the presence of these defects may affect the result of target detection algorithm. So before proceeding to the next step, we will eliminate these defects through pre-processing work.

Conversion between different color space

Color space is the way color is organized. In combination with physical device profiling, it allows for reproducible representations of color, in both analog and digital representations. Because of the different application scenarios, different color spaces will be used. The commonly used color spaces are:

- Describes what kind of light needs to be emitted: RGB, RGBA
- Describes what kind of light needs to be reflected: CMYK
- Describe the feelings of human vision: HSV, HSL
- For commercial use: Munsell color system, PMS, NCS

Because the hardware and evaluation criteria for different color spaces are different, they cover different color ranges.

Different from color image, greyscale image is one in which the value of each pixel is a single sample representing only an amount of light, that is, it carries only intensity information. Images of this sort, also known as black-and-white or gray monochrome, are composed exclusively of shades of gray. The contrast ranges from black at the weakest intensity to white at the strongest [42].

Compare with the image in RGB color space, which has 3 channels, grayscale image only has 1 channel to store the intensity value of the light. For scenes that do not need to care about the color information of the image, using grayscale images can save a lot of computation than using color images. Thanks to the fewer channels and the ability which can reflect the morphological features of objects in the image, grayscale images are widely used in target detection.

In this OCC system, what we need to pay attention to is the space coordinates and brightness values of the LEDs in the LED array. Since no color shift OOK modulation is involved, the actual color of the LED is not our consideration. For the default output of most camera which we are using is in RGB color space, in order to reduce the calculation load, we can reduce the number of channels by converting the received image to a grayscale image in the first step. In OpenCV, to converts an input image from one color space to another can be achieved by the cvtColor() function. It supports a lot of conversion types, and the conversions we always use is:

- RGB/BGR ↔ GRAY:
- RGB ↔ HSV
- RGB ↔ HLS
- RGB ↔ BGR

What we need is to convert RGB image to grayscale image and this part of code will be like this:

```
gray = cv2.cvtColor(frame, cv2.COLOR_RGB2GRAY)
```

`frame` is original image and `gray` is the output image. `cv2.COLOR_RGB2GRAY` is the color conversion code which mean convert from RGB color space to grayscale image, the conversion algorithm is:

$$Y = 0.299\,R + 0.587\,G + 0.114\,B$$

R, G, B are the values of red channel, green channel and blue channel respectively. Y is the converted value.

Thresholding

In the actual experimental environment, due to environmental interference and equipment constraints, we can hardly get a perfect image (like Figure 2-2) with a simple camera. For example, in practical applications, camera only have limited exposure time, which make camera cannot catch all the light. In addition, because the optical radiation of LED has strong directivity (for example, the half angle of SI5312-H is only 8 degrees), if there is a certain deviation between the directions of the LED in the array, the eventual brightness will be very different. As shown in Figure 3-3. Two examples of uneven brightness., the right side is darker than the left side in both images. Such uneven brightness may lead to a large deviation of the positioning result of the Position_LED part. In order to solve this problem, we can perform threshold processing on the image.

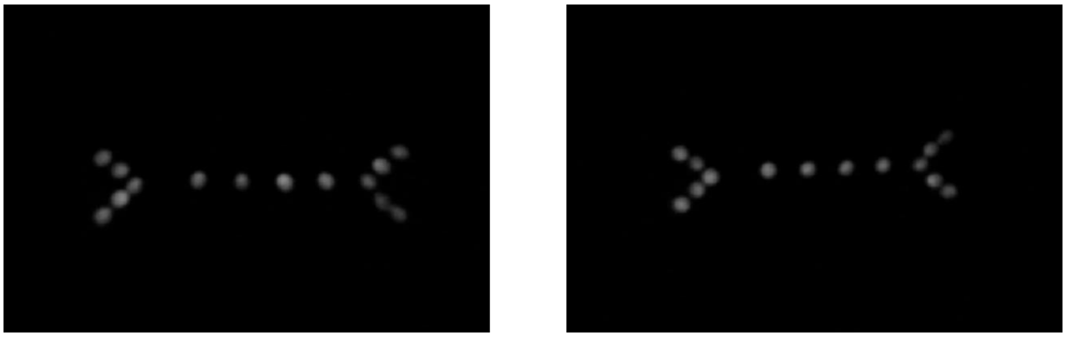


Figure 3-3. Two examples of uneven brightness.

Thresholding is a commonly used method in grayscale image processing. It will filter the pixels of an image by comparing with a threshold. Thresholding can be used to remove noise from images or enhance image contrast. According to the different threshold setting methods, there are three kinds of thresholding in OpenCV: simple thresholding, adaptive thresholding and Otsu's thresholding.

As the name implies, simple thresholding will simply compare pixel values with a specified, fixed threshold. The next step will be determined by the threshold type. In OpenCV, simple thresholding has 5 different styles of threshold: THRESH_BINARY, THRESH_BINARY_INV, THRESH_TRUNC, THRESH_TOZERO, THRESH_TOZERO_INV. Corresponding signal of each threshold style is shown in Figure 3-4. The blue solid line represents the waveform, and the green dotted line is the given threshold.

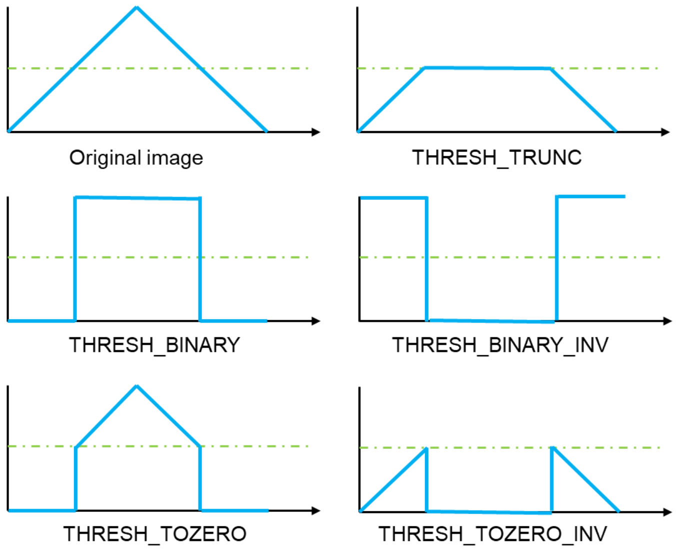
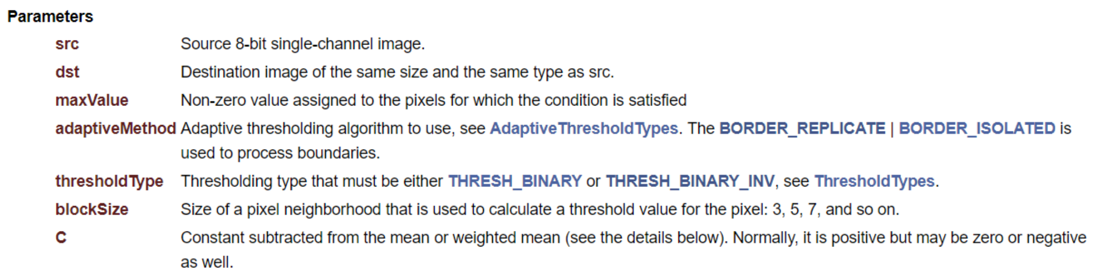


Figure 3-4. Corresponding signal corresponding to different threshold methods.

Because the threshold of the simple threshold method needs to be specified before running, and the whole picture will use one same threshold, which makes the selection of the threshold of the simple thresholding will directly affect the processing effect. If the value is too low, there will be a lot of background noise, and if the value is too high, there is a risk of data loss.

In adapting thresholding, the system will use an algorithm to calculate the threshold for a small region of the image instead of using a given global threshold. The API of adapting thresholding in OpenCV for Python is shown below:

```
    dst = cv.adaptiveThreshold( src, maxValue, adaptiveMethod, thresholdType,
blockSize, C[,dst])
```



Parameters

src	Source 8-bit single-channel image.
dst	Destination image of the same size and the same type as src.
maxValue	Non-zero value assigned to the pixels for which the condition is satisfied
adaptiveMethod	Adaptive thresholding algorithm to use, see AdaptiveThresholdTypes. The **BORDER_REPLICATE** | **BORDER_ISOLATED** is used to process boundaries.
thresholdType	Thresholding type that must be either **THRESH_BINARY** or **THRESH_BINARY_INV**, see ThresholdTypes.
blockSize	Size of a pixel neighborhood that is used to calculate a threshold value for the pixel: 3, 5, 7, and so on.
C	Constant subtracted from the mean or weighted mean (see the details below). Normally, it is positive but may be zero or negative as well.

Figure 3-5. Parameters of adapting thresholding

OpenCV provide 2 kind of adaptive method for adapting thresholding: ADAPTIVE_THRESH_MEAN_C and ADAPTIVE_THRESH_GAUSSIAN_C, and there are some differences in how these two methods are calculated. The description of these 2 adaptive methods from the official document is shown below:

ADAPTIVE_THRESH_MEAN_C:
the threshold value T(x,y) is a mean of the blockSize×blockSize neighborhood of (x,y) minus C
ADAPTIVE_THRESH_GAUSSIAN_C:
the threshold value T(x,y) is a weighted sum (cross-correlation with a Gaussian window) of the blockSize×blockSize neighborhood of (x,y) minus C, The default sigma (standard deviation) is used for the specified blockSize.

According to the description, ADAPTIVE_THRESH_MEAN_C will calculate threshold through getting the mean value of neighborhood area, while ADAPTIVE_THRESH_GAUSSIAN_C will get the threshold by calculating weighted sum of neighborhood values where weights are a gaussian window.

In order to show the different processing effects of those threshold methods, we can compare them through a comparison which is given in OpenCV’s official website (Figure 3-6).

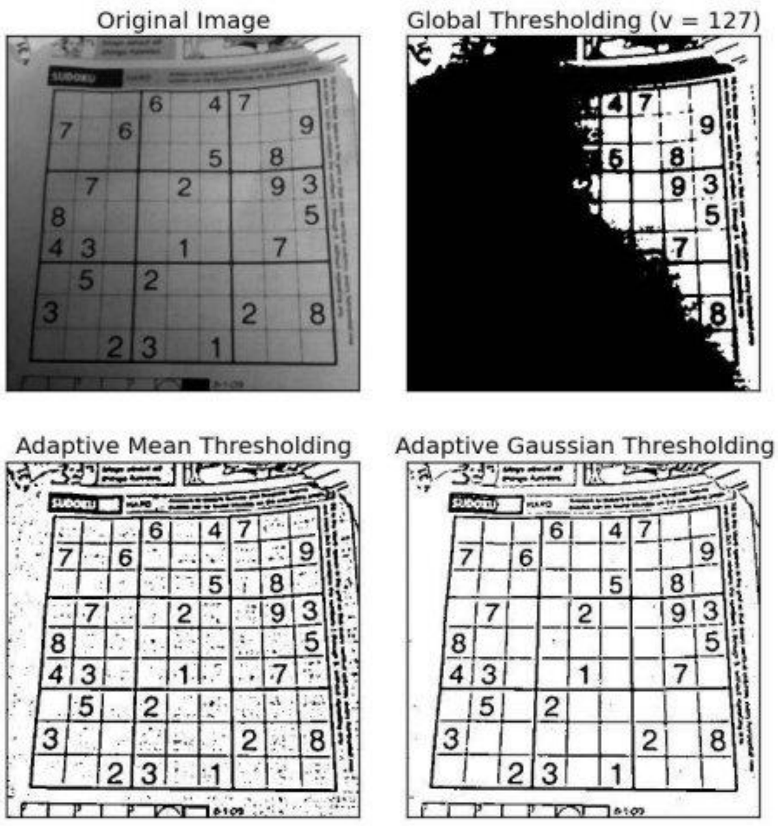

Original Image
Global Thresholding (v = 127)
Adaptive Mean Thresholding
Adaptive Gaussian Thresholding

Figure 3-6. Comparison of different thresholding in OpenCV[43]

As we can see in the above diagram, the original image has the defect of uneven brightness. Because a high threshold is selected, the method of using the global thresholding loses a lot of information we need. Meanwhile, Since the adaptive thresholding uses an adaptive threshold based on the brightness of the image during the binarization process, the final result achieves a certain degree of brightness balance. if we compare these two methods, the adaptive gaussian threshold have better noise reduction effect.

As for Otsu's binarization, its main idea is to calculate a threshold based on the histogram, and then use this threshold to binarize the image, it's like a simple threshold method with adaptive global thresholds. This method has a good performance with bimodal image. In order to verify its performance, we conducted several experiments. But the result (Figure 3-7) is not pretty. Because the histogram of this image has only one peak, this algorithm performs poorly, even worse than the simple thresholding in this case.

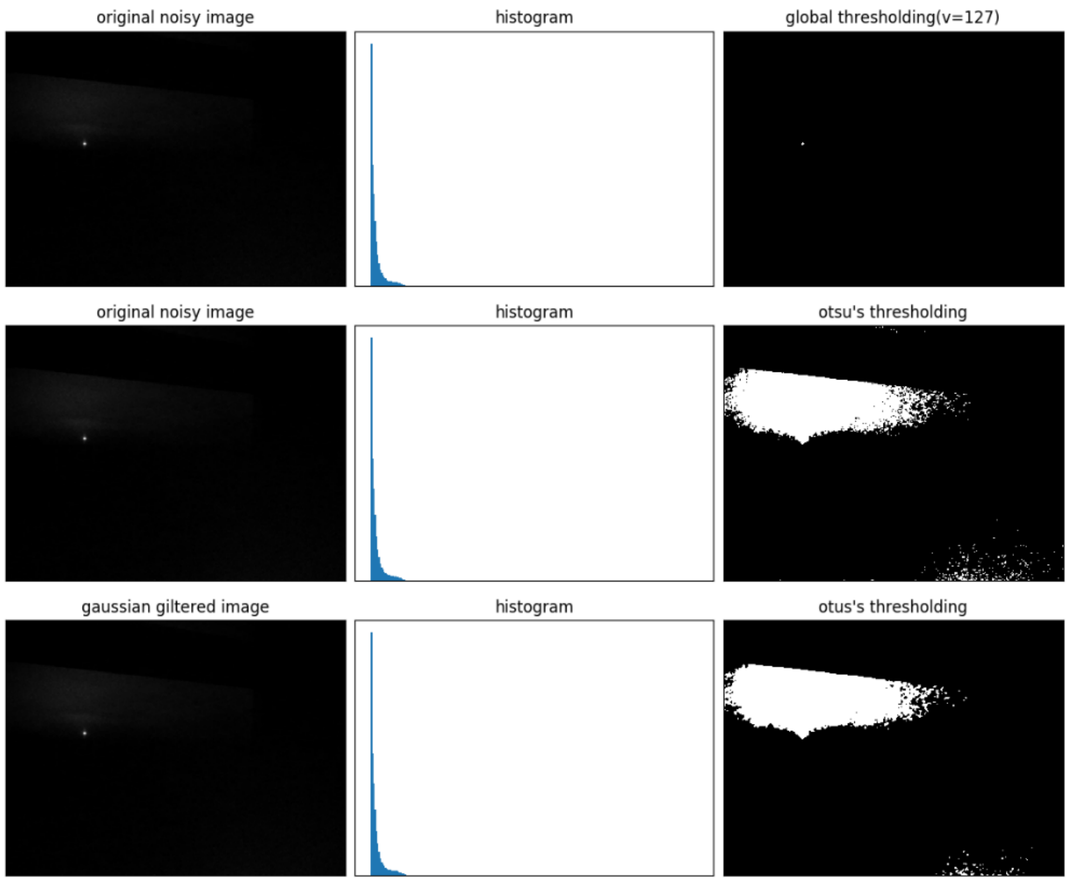


Figure 3-7. Result of Otsu's thresholding

So, in the pre-processing, we will first use the fixed threshold method eliminate low brightness noise by selecting a lower threshold. After that, the adaptive gaussian threshold will be used to balance the brightness and do a further noise reduction. In addition, we will smooth the image before performing the brightness balance, which is helpful in noise reduction. The flow chart in the pre-processing stage is shown below (Figure 3-8):

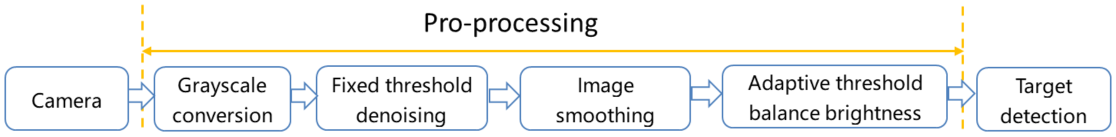


Figure 3-8. Flow chart of pre-processing

After pre-processing, we will get a grayscale image with 8-bit color depth, which has been processed of noise reduction and brightness balance, Figure 3-9 shows the pre-progressed Figure 3-3, and you can see that the brightness is balanced.

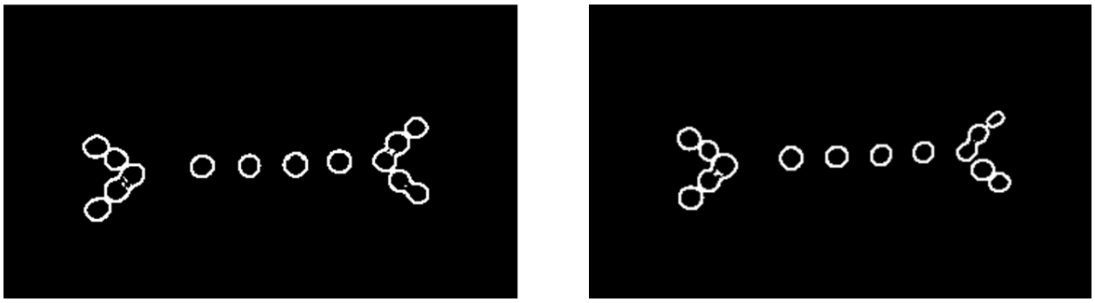


Figure 3-9. After pre-progress, the brightness on both sides is the same.

Target Detection Algorithm

In our system, the Position_LEDs are distributed symmetrically on both sides of the LED array, they are designed to identify the location of LED array. So, in the target detection, our main goal is to locate and return the coordinates of the center points on both sides of right / left Position_LED, while the center point is marked in red in Figure 2-2.

Template

Before start to identifying, the system needs to know what we are looking for. In other words, we need to prepare a template of target. In the target detection of our system, the target is the

Position_LEDs, so we need to find a suitable template for them first. For our system, the form of the template needs to have the following characteristics:

1.	With obvious feature	-- easy to detect
2.	Simple	-- low complexity
3.	Anti-noise	-- ability to keep shapes with noise
4.	Orientation	-- to distinguish between right and left

With these considerations, we chose a pattern like the Angle brackets “> <”, as shown in Figure 3-10 [44]. They are asymmetrical, simple enough to be clearly represented by just five LEDs; oriented, we can easily distinguish their left and right sides; The shape is stable, and the features are obvious.

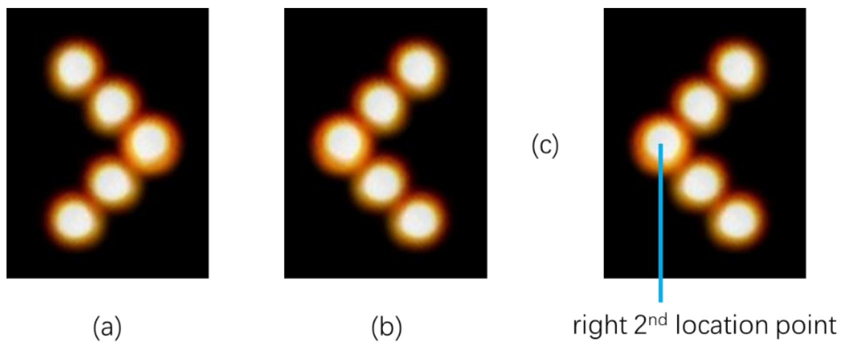


Figure 3-10. Left template(a) right template(b) and the second location point(c)

We will identify the left and right parts of the Position_LED separately.

Feature Detection?

When it comes to target detection, feature detection is often used. The basic idea of feature detection is to find the feature points in the template and the target image respectively, and then compare the feature points to identify the position of the template in the target image. According to the feature detection method used, there may be a slight difference in the feature used to detect, but most of them using corner detection, because the corners of a region will have maximum variation when region is moved. In OpenCV, there are several feature detection algorithms can be used:

- Harris Corner Detection
- Shi-Tomasi Corner Detector & Good Features to Track
- SIFT (Scale-Invariant Feature Transform)
- SURF (Speeded-Up Robust Features)
- FAST Algorithm for Corner Detection
- BRIEF (Binary Robust Independent Elementary Features)
- ORB (Oriented FAST and Rotated BRIEF)

They have their own advantages and disadvantages and what we need is the algorithm which is fast, reliable and low cost of system. For these reasons the final choice is ORB, which is fast (faster than SIFT) and relatively stable (has rotation invariance, which FAST doesn`t). And it`s free to use (SIFT and SURF is patented).

ORB (Oriented FAST and Rotated BRIEF) was brought up by E. Rublee, V. Rabaud, K. Konolige and G. Bradski in 2011 [45]. It's a computationally-efficient replacement to SIFT that has similar matching performance, is less affected by image noise, and is capable of being used for real-time performance. ORB algorithm proposed a feature based on FAST keypoint detector and BRIEF descriptor but made some modifications to enhance the performance. For FAST features do not have an orientation component, the author proposed a method to measure the orientation by using intensity centroid, and it's called oFAST. Since BRIEF has a poor performance when image is rotated, they introduce a new descriptor rBRIEF to improve that defect. According to the paper, ORB is much faster than SURF and SIFT and ORB descriptor works better than SURF. They even made a real-time feature tracker that can run on a smart phone which only have a basic hardware configuration.

Everything looks nice on the paper, but if it is not suitable for us, we need further experimentation. In order to verify his performance, we used different combinations to conduct experiments. Before we start our experiment, template and target image will be assigned first.

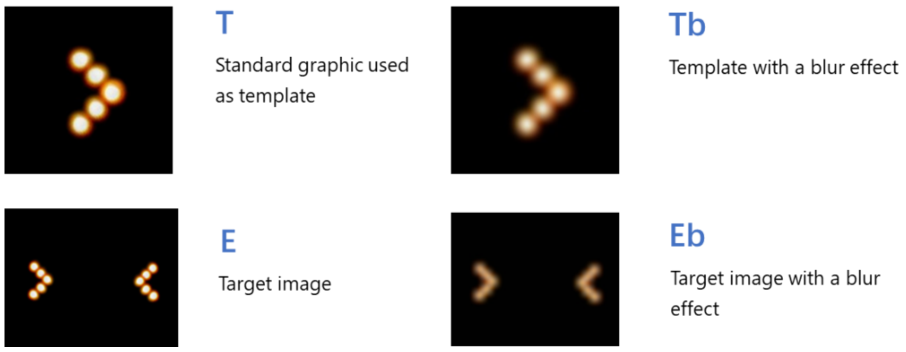

T
Standard graphic used as template
Tb
Template with a blur effect
E
Target image
Eb
Target image with a blur effect

Figure 3-11. Templates and target images

To verify the ORB, we will use 2 templates (T&Tb) and 2 target images (E&EB), as shown in Figure 3-11, and use the ORB algorithm to try to find the template part of the target image. It should be noted that this template is not simply intercepted from the target image, the highlights in these pictures are arranged manually, so there will be a small deviation between the position of the highlights between the picture and the picture, in order to imitate the possible deviation of the real LED array. There will be three sets of experiments, which are ORB (basic), ORB+RANSAC and ORB + GaussianBlur + RANSAC. And the matching results will be evaluated in 3 ways:

1. location of matching point (right side/left side/both side)
2. Number of correctly matched keypoint (a lot/very little)
3. Distortion (high/low)

ORB (basic)

First experiment is using only ORB to match the target and template, But the result does not look very good. In the four sets of matches, the positions of the matching points are distributed on both sides, that is to say, there are many mismatched points; as can be seen from 2, 4, the algorithm performs poorly in the face of fuzzy targets; There is a crossover of the line, which means that they all have distortion in the matching results.

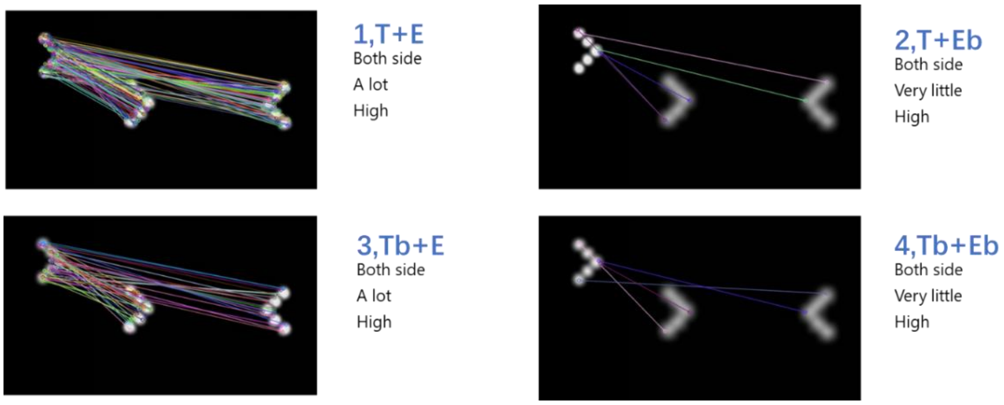


Figure 3-12. Matching result of ORB (basic)

ORB + RANSAC

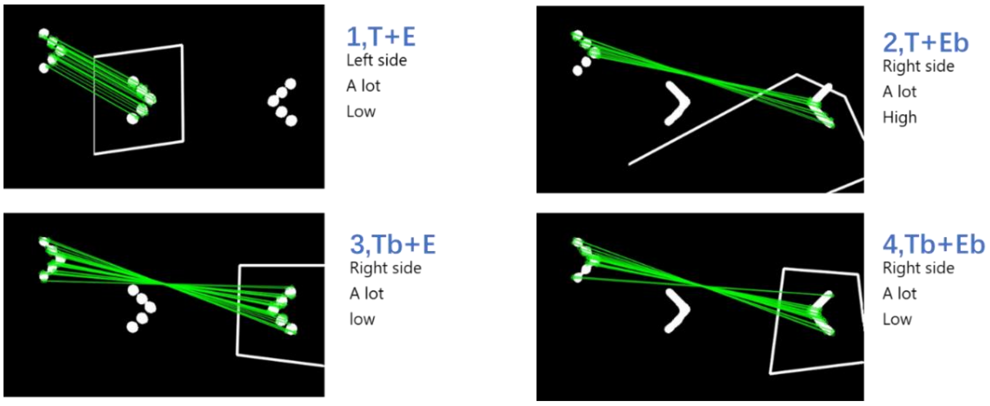


Figure 3-13. Matching result of ORB + RANSAC

RANSAC (Random Sample Consensus) is a robust estimation algorithm which can be used to get the relatively high reliability results. By using RANSAC we have improved matching quality, at least all the key points are on the same side. Also cv.findHomography() function can return transformation matrix. For matching results, it is better than only using ORB. But in addition to 1st image, the matching result is still not ideal.

ORB + GaussianBlur + RANSAC

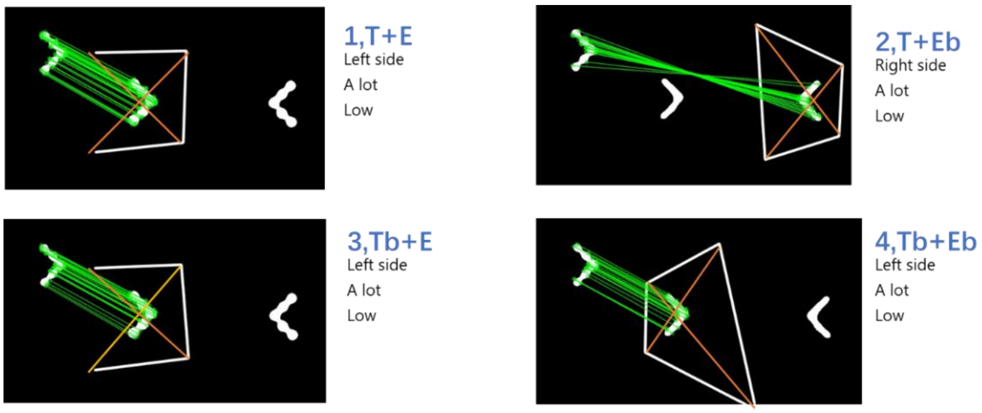


Figure 3-14. Matching result of ORB + GaussianBlur + RANSAC

As can be seen from the above three experiments, the ORB feature detection method does not show the high performance we expect, there is always a mistake in the process of matching, which is what we are trying to avoid. For this result, I think it may be because our templates and target images are too simple to extract enough keypoints with strong correlation. Or maybe just because we didn't find the correct way to use it. So even though ORB algorithm looks perfect, it seems that it is not the best one for our system.

Template matching

Template Matching is a method for searching and finding the location of a template image in a larger image. Its core logic is to slide the template on the target image, and then use a specific comparison method to calculate the similarity between the template image and the target image at this point. It returns a grayscale image, where each pixel 's value is the similarity of this position, says how much does the neighborhood of that pixel match with the template. This process is similar to two-dimensional convolution, and different comparison methods will result in different results. In OpenCV-Python this method is provided by a function cv2.matchTemplate(), and there are 6 comparison methods can be used, their formula is listed in Table 1.

Table 1. Comparison method and their formulas offered in matchTemplate(), where T stand for template and I stand for target image. The return value will be stored as R.

TM_SQDIFF	$\boldsymbol{R(x,y) = \sum_{x',y'} \left(T(x',y') - I(x+x',y+y')\right)^2}$
TM_SQDIFF_NORMED	$R(x,y) = \frac{\sum_{x',y'} \left(T(x',y') - I(x+x',y+y')\right)^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} I(x+x',y+y')^2}}$
TM_CCORR	$R(x,y) = \sum_{x',y'} \left(T(x',y') \cdot I(x+x',y+y')\right)$
TM_CCORR_NORMED	$\mathrm{R(x,y)} = \frac{\sum_{x',y'} \left(\mathrm{T}(x',y') \cdot \mathrm{I}(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} \mathrm{T}(x',y')^2 \cdot \sum_{x',y'} \mathrm{I}(x+x',y+y')^2}}$
TM_CCOEFF	$R(x,y) = \sum_{x',y'} \left(T'(x',y') \cdot I'(x+x',y+y')\right)$ Where, $T'(x',y') = T(x',y') - 1/(w \cdot h) \cdot \sum_{x'',y''} T(x'',y'')$ $I'(x+x',y+y') = I(x+x',y+y') - 1/(w \cdot h) \cdot \sum_{x'',y''} I(x+x'',y+y'')$

TM_CCOEFF_NORMED	$R(x,y) = \frac{\sum_{x',y'}\left(T'(x',y') \cdot I'(x+x',y+y')\right)}{\sqrt{\sum_{x',y'} T'(x',y')^2 \cdot \sum_{x',y'} I'(x+x',y+y')^2}}$

According to the formulas in the above (Table 1), we can see that the 6 comparison methods can be divided into 3 groups. TM_SQDIFF choose to calculate the squared difference between the target image and the template image at each pixel, and then summing them up. As for TM_CCORR, it will calculate the sum of correlation between pixels of the target image and the template image. The last one, TM_CCOEFF, will uses correlation coefficients to calculate the sum of correlation. At the same time, they all have a normalized version, which end with “NORMED”. Because the formula is only a theoretical explanation, and cannot show the processing effect intuitively, we will determine which method we use through a set of comparisons. Comparison result, template and target image is shown in Figure 3-15.

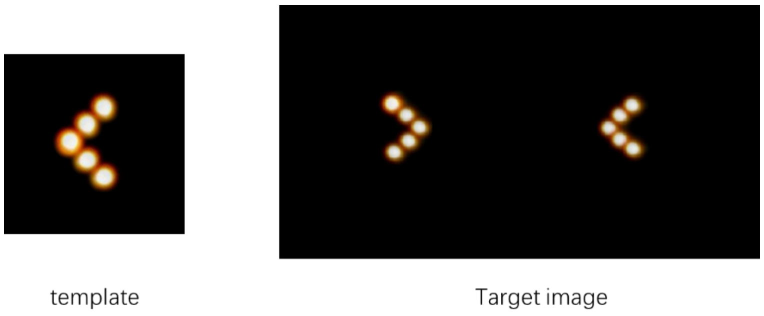
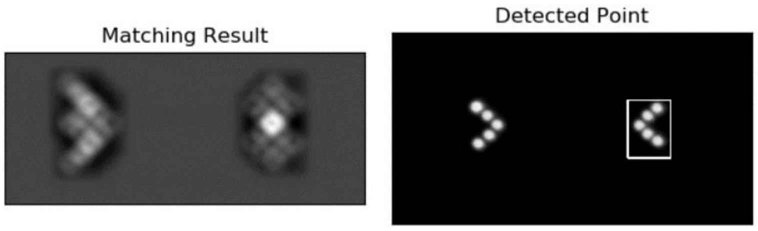
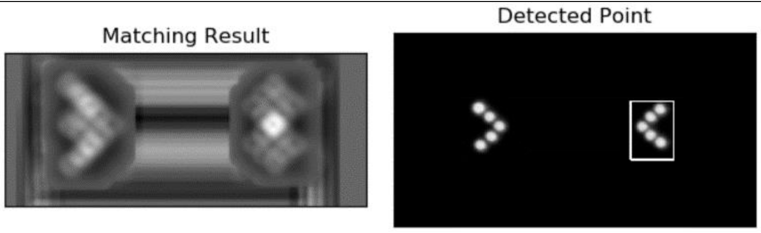
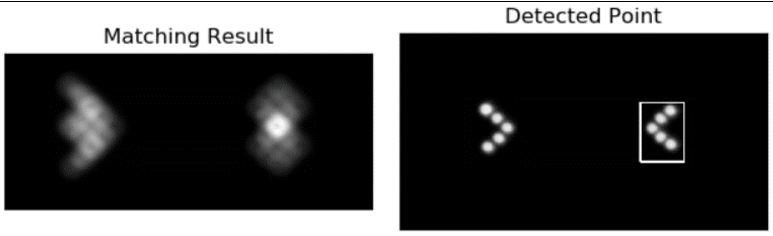
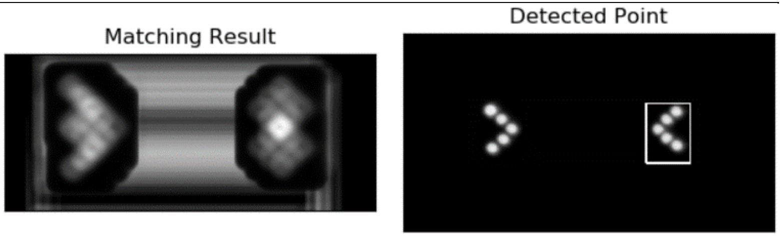
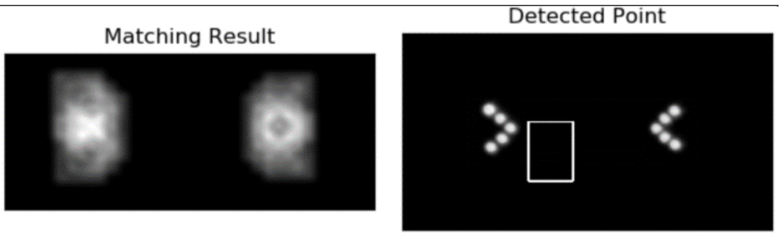
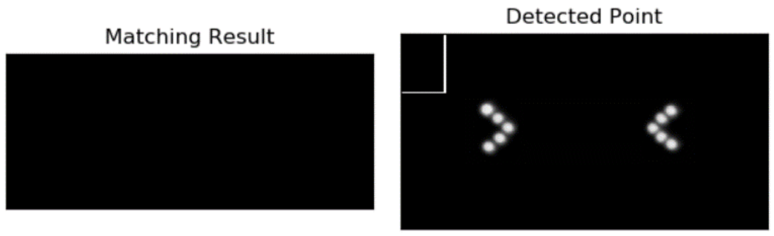


Figure 3-15. Template and target image

Table 2. Comparison of matching method

TM_CCOEFF	
TM_CCOEFF_NORMED	
TM_CCORR	
TM_CCORR_NORMED	
TM_SQDIFF	

TM_SQDIFF_NORMED	

From the matching results, the first four methods match the correct results. From the matching result, the TM_CCORR method seems to be more stable, because the highlights on the matching result are concentrated near the patterned position. Therefore, we selected the CV_TM_CCORR method.

Correcting

The disadvantages of template matching

As we can see, template matching is a very basic and direct algorithm, it finds out what we need by comparing the different parts of the template and the target image. As a result, template matching is not scaling invariant nor is it rotation invariant. In order to find out how much they can affect the matching result, I conducted a comparative test. The target patterns in the test has 3 variables:

- Size: size ratio of the template (0.1~2.0)
- Orientation: rotation degree (-45°~45°)

It can be seen from the results (Figure 3-16), for our template image, the scaling of the size has the greatest impact on the matching results, especially when the target image is smaller than the template image. In addition, the tilt of the image causes a slight shift in the position of the center point (small point in Figure 3-16), which will also have adverse consequences for subsequent positioning process. In order to get the results correctly, we need to take steps to correct these errors.

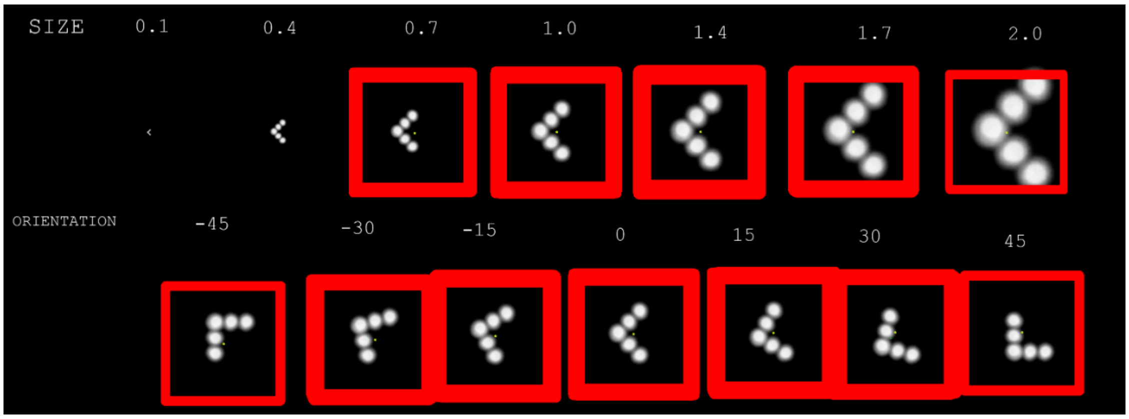


Figure 3-16. Size & Orientation

Scale

When the size of the template image and the target pattern are different, especially when the target pattern is smaller than the template image, the matching may fail, so we need to adjust the template image to the appropriate size before matching. But what is the appropriate size? The answer may already be told. If we look at the first line of Figure 3-16, you will find that the two areas in the middle have the better matching result, with scales of 1.0 and 1.4. So, we can know that the target pattern is preferably 1~1.4 times larger than the template image, and the matching result is better. In other words, the template image is preferably slightly smaller than the target pattern. And here comes another question, how to know the size of the target pattern?

In the current system, we know that only 14 LED lights will be lit at the same time in the LED array. Each of the right and left part of Position_LED have 5 LEDs while the left 4 LEDs belong to Data_LED. For template images, it is made up of 5 virtual LED. If we assume that all LEDs have the same brightness and no background noise, then when the size of the template image is the same as the size of the target pattern, the brightness of the template should be slightly higher than 1/3 of the overall brightness of the target image. Based on this assumption, we will establish a formula to adjust the size of the template graphic.

We already know that after pre-processing, what we get is the image after noise reduction, balance brightness, and binarization. Because there are only two values of 0 or 1 in the binary image, the luminance value can be calculated directly by calculating the sum of the matrix elements. Next, we will use a perfect picture (no noise, equal brightness of LEDs) as the target picture, and directly intercept the left half of Positio_LED as a template to see how much the brightness ratio between them is.

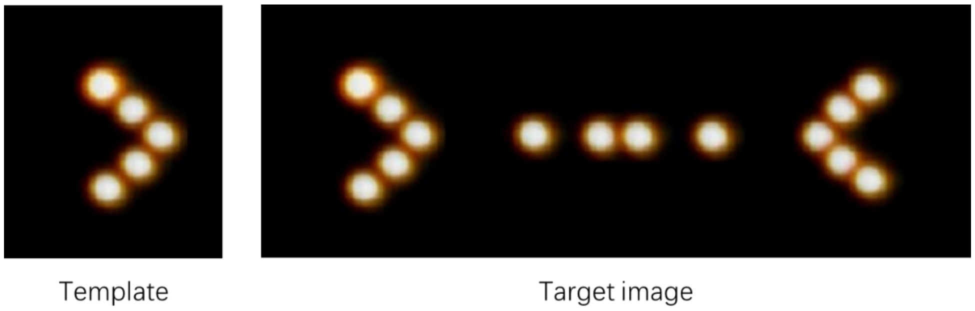


Figure 3-17. Template and target image we used

After calculation, the brightness value of the target image is 1125060, and the brightness value of the template is 418200. The brightness of the template image is equivalent to 0.372 times of the target image. Similar to our guess, but with the use of perfect image.

Now let's imagine a difference in the size of the template image and the target pattern, because when the size of the image changes, the area of the pattern in the figure actually expands by a square. Therefore, if you do not consider the preprocessing process, the brightness between the two should match the relationship:

$$\xi_{template} = \sqrt{B_{image}/\left(B_{template} * 2.8\right)} \tag{1}$$

In order to verify this formula, we carried out a set of experiments. In the experiment, we scaled the target image which shown in Figure 3-17 by setting index as 0.5, 0.6…1.4, 1.5, and analyzed the relationship between the scaling value and the brightness ratio. After finishing the data, we got a chart (Figure 3-18) and a fitting formula (2).

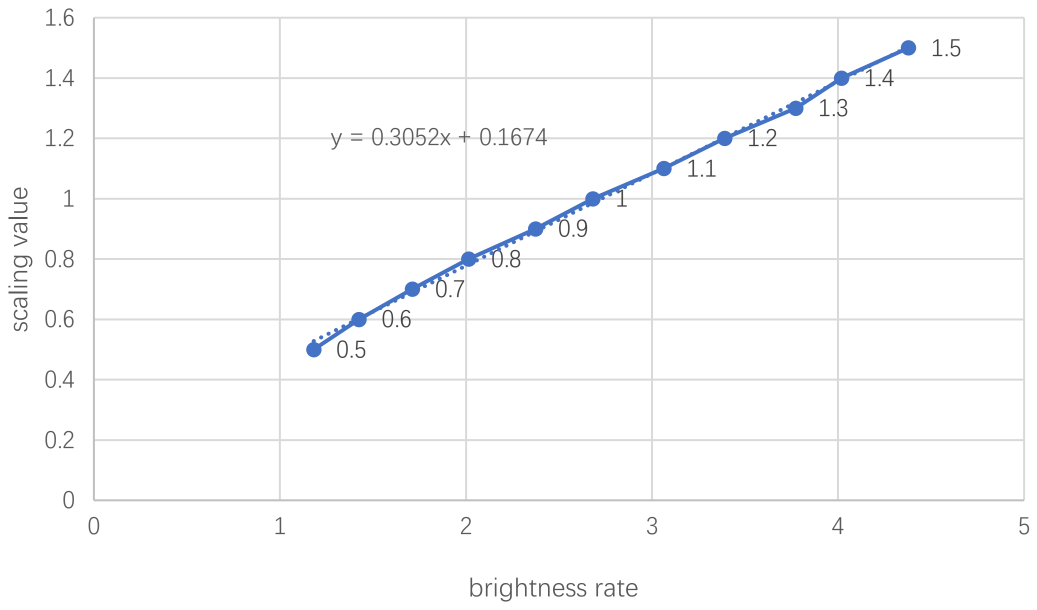


Figure 3-18. Scaling value & Brightness rate

$$\xi_{template} = 0.3052\left(B_{image}/B_{template}\right) + 0.1674 \qquad (2)$$

To be honest, I didn't expect the fitting equation to be a linear equation at first, so it was a bit of a surprise to see it. After analysis, I found that the problem occurred in the pre-processing stage because of the use of adaptive threshold method. Because in the adaptive threshold method, the threshold is determined by the average of the brightness of pixels within a certain range (decided by blocksize). So, in the case where the LED is imaged as a solid circle, the processed image will be hollow. Because the brightness of the pixels near the center of the circle is all high, the average threshold near the center point will rise to a high level, in the end only a few prominent bright spots are left. The larger the size of the pattern, the higher the proportion of the hollow. Therefore, when the binarized image is used to obtain the brightness of the original image, the obtained brightness value becomes smaller; otherwise, when the pattern size is small, the obtained brightness value will be larger than it should be.

Tilt and Secondary positioning

Because the basic principle of this function is to calculate the similarities and differences of the two images, the final matching result can be biased when the pattern in the target image is tilted, as shown in Figure 3-20. In order to get a higher matching precision, we introduced the secondary positioning. As the name suggests, it will locate the target for a second time on the basis of initial positioning results, and the target which we want to locate in this step is called the second location point. We choose the corner of the Angle brackets as the second location point (Figure 3-10 (c)), because it should theoretically be coaxial with Data_LED, get its position will be helpful for the next step. On the other hand, this point only has very small deviations in the tilted image, making it easier to accurately determined than other points.

In the secondary positioning, we first calculate the approximate position of the second location point from the known left and right location points by using the proportional relation existing in the LED array. If the LED is closely aligned and the LED diameter is treated as Unit 1, the distance between the left center point and the right center point of Position_LED is approximately 13. Create a one-dimensional coordinate axis between left location point and right location point, using the midpoint as the origin of the axis, the coordinates of the left and right second location points are approximately -5.5 and 5.5 respectively, we can use that to calculate the approximate position of the second positioning point. However, the results may be biased rather than the exact location of the second location point in the actual image. In this step, we will scale the coordinates of the approximate position in the vertical direction to reduce the vertical drift (in this experimental system, we chose 0.8 as the scale factor), and then use the precise positioning to determine the exact location of this point in the image.

In this step, we will look for the target in a smaller scope. Because of the change of scale, the shape of the second location point can be approximately treated as a circle, and the goal of this step is to find a point as close to the center of the circle as possible. To achieve this goal, we will set up a detection array to assess the fitness and determine the direction of the next adjustment by analyzing the distribution of the values in the array. After several iterations, we will approach the real position. The detection array is a 3 by 3 square array made up of 9 detection points. The calculated approximate location of the second positioning point will be the location of first detection array's center point. The sides' length of the array is an important parameter need to consider. If the selection is too small, the target may be missed; If the sides are too large, there will be detection points outside the target when fully fitting. In this experimental system, we choose 0.6 as the side length, and the diagonal length is about 0.85. So, when the test results are fully fit, all the detection points can be included within the target.

After obtaining the location of each detection point in the detection array, the brightness value of each point will be measured with a threshold. If the brightness value exceeds the threshold, the value of that point is 1. If not, it's 0. Next, we will calculate the sum of the detection arrays. If sum is greater than our expectations (7 is used in this experimental system), we can think that the center point at this time is close enough to the center of the target, and we can stop iterating and return the coordinates of the current center point; if it does not meet the requirements, we will adjust the position of the center point. First, we need to compare the sum of the first row and the third row, the normalized result will be used as a horizontal adjustment vector's normalized coefficient. Correspondingly, we can compare the first column with the third column to get the vertical adjustment vector's normalized coefficient. The combination of the vertical and the horizontal adjustment vectors is the direction vector we need. Now that we have the direction, the next step is to determine the step length of the adjustment. The step length is the same as the side length of the inspection array. If it is too long or too short, the system performance will be affected. If the step size is too large, it may cause the result to oscillate, or even the result will not converge, leading to the failure of positioning. Conversely, too short steps slow down the convergence of the system and extend the positioning time. To balance stability and speed, we use a variable adjustment step size.

Because our goal is to make the center of the detection array as close to the center of the circle as possible, we can use its value as an adjustment parameter for the step size. If its value is 0, it means that it is still outside the scope of the target circle. We use a larger adjustment stride to reduce the time required for positioning. If its value is 1, it means that it has already entered the target circle. At this time, a smaller adjustment step is used for more precise adjustment. At the same time, the module of the normalized horizontal adjustment vector and the vertical adjustment vector can also adjust the step size to a certain extent. Because in some cases they are not integers. Combine the adjustment step and direction vector, we can get the adjustment vector needed. In our experimental system, we use the following equation to calculate the adjustment vector:

$$adjustment\ vector = \begin{cases} 0.3 * \left(x * \vec{X} + y * \vec{Y}\right),\ value_{centerpoint} = 1 \\ 0.6 * \left(x * \vec{X} + y * \vec{Y}\right),\ value_{centerpoint} = 0 \end{cases}$$

x, y is the normalized coefficient
$\vec{X}$,$\vec{Y}$ is the unit vector of the x-axis and y-axis

After we get the adjustment vector, we can use it to adjust the coordinates of the center point, and then enter the next iteration. In the next iteration, a new detection array will be created with the updated coordinates of the center point.

Because of defects in imaging or other reasons, we may never get a perfect location. Also, there are only nine detection points, the accuracy of the detection is not very high. In some cases, the iterative process can become a dead loop. To avoid this, we need to set the maximum number of iterations for an iteration. Therefore, the precise positioning will exit the iteration in two situations:

1. The value of the detection array meets the expectation
2. The number of iterations reaches the maximum number.

The Figure 3-19 is an example of an accurate positioning. The green dot means that the value of this point is 1, and the blue dot is 0. The red dot is the center point of the detection array. After two iterations, it reached the expected value.

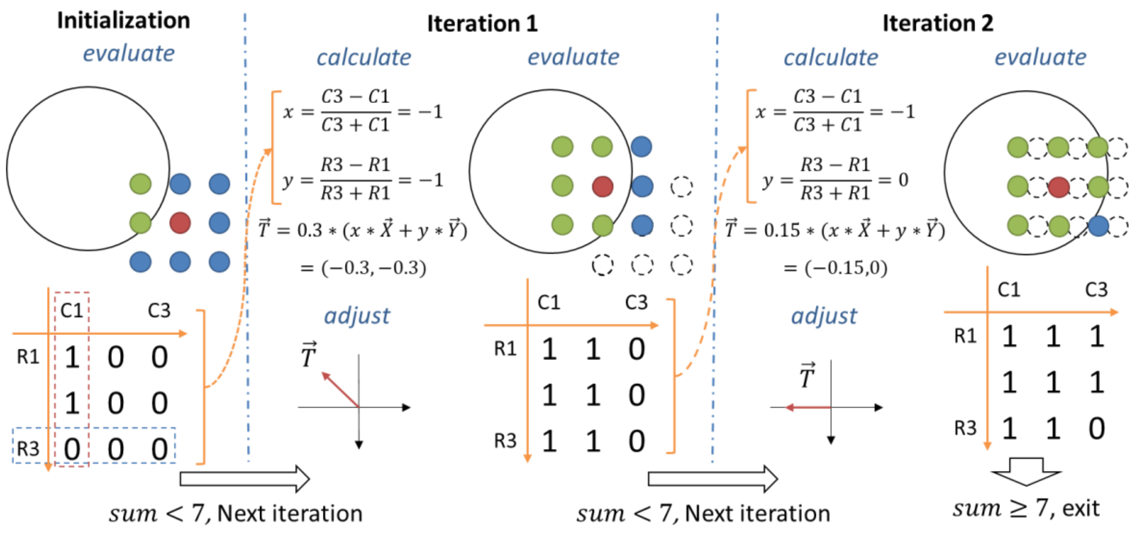


Figure 3-19. The process of precise positioning.

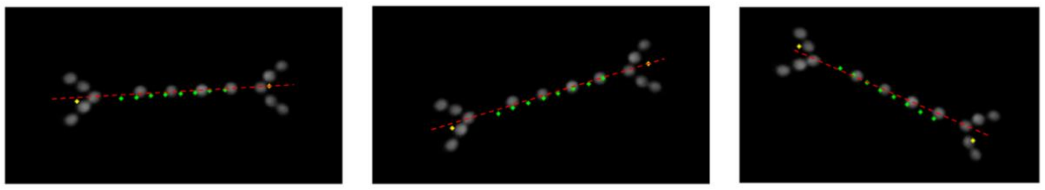


Figure 3-20. After initial positioning, location point of Position_LED(yellow) and Data_LED(green) have some deviation from the central axis (red)

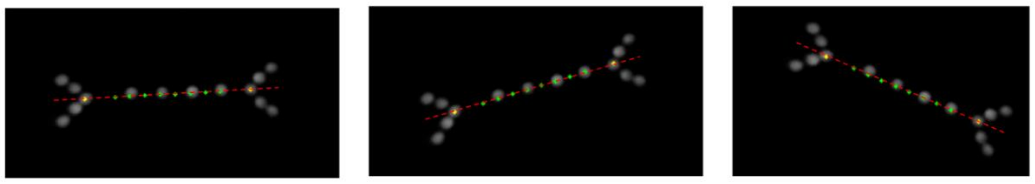


Figure 3-21. After the secondary positioning, the deviation is well corrected.

Figure 3-20 is the result of initial positioning. As can be seen from the image, the center point (yellow point) of the left and right Position_LEDs and the coordinates (green points) of the Data_LED calculated on this basis are shifted to different degrees with respect to the symmetry axis (red dotted line). Because of the coordinates of each point after calculation, there is no small drift with the real position. After sampling the LED of the Data_LED part, the three sets of experiments all get the wrong result.

Position result After secondary positioning is shown in Figure 3-21. Be aware of that, the yellow point marks the position of the second location point rather than the location point of Position_LED. As we can see from the image, the LED of the second location points and the Data_LED parts are distributed on the symmetry axis. After sampling the LED of the Data_LED part, we can get the correct result we want. Hence, we can say that this method indeed effectively corrects the deviation of the initial positioning results and achieve higher-precision positioning of the target points.

After getting the exact position of the second location point of both sides, we will reestablish a one-dimensional number axis between two points and take the midpoint as the origin. So, all points on Data_LED should be passed by this number axis. We can calculate the coordinates of each points according to Figure 2-2, results are summarized in the table below (Table 3).

Table 3. Coordinate of each detection point on number axis

2nd location point(L)	P1	P2	P3	P4	P5	P6	P7	P8	2nd location point(R)
-5.5	-3.5	-2.5	-1.5	-0.5	0.5	1.5	2.5	3.5	5.5

Combine this proportional relation and the coordinates of the left/right second location point, we can find the location of each point in Data_LED. Then compare the brightness value of this point with a threshold. If it exceeds the threshold, it is 1, and if it does not exceed it, it is 0. The result will be summarized and stored as "P1P2P3…P8". Because in this experiment, Data_LED is composed of

8 LEDs. After all the comparisons are completed, we will get an 8-bit binary code, like”01100101”. Because this code corresponds to a particular piece of information, we might as well call it the eigenvalue.

3.2 Protocol & Synchronization Control

Protocol

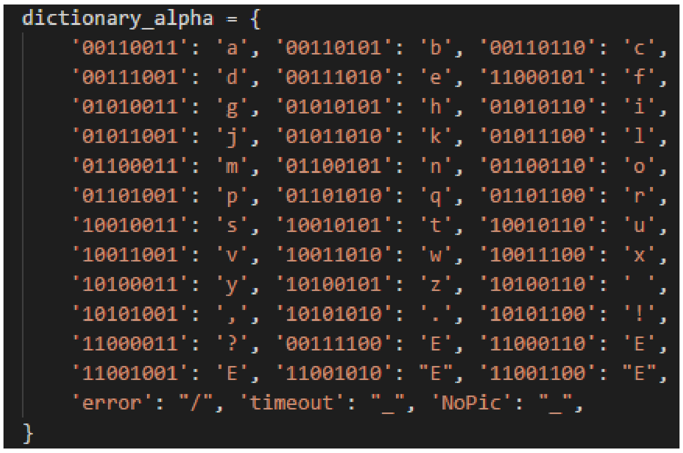
The role of the protocol is to translate the eigenvalues into the results we need, and error detection should also be done during this process. In this experimental system, eigenvalue only has 8-bit space to put binary numbers, which can make up 256 different numbers. But we can't assign them all, and then we can't tell if there's an error in the communication process. We decided to light up 4 LEDs each time to pass information, means that there will only be 4 “1” in the eigenvalue. If the number of "1" in the result exceeds four, it means that there is an error in the transmission of this frame signal, we will return “error” to the main process. In addition, we should pay attention to the possible errors caused by a single strong light source. For example, the system itself does not have an output, but when a large point of light appears in the range of Data_LED, and crosses multiple detection points at once, it can be read by the system incorrectly. So, the eigenvalues like "01111000" also need to be excluded. In this experimental system, 8 detection points will be divided into two parts. P1~P4 is the first half, P5~P8 is the second half. Each part will have only 2 “1” each time, together there will be 4 “1” in the eigenvalue. So, a valid eigenvalue should meet the conditions:

$$P1 + P2 + \cdots + P8 = 4$$

$$and$$

$$P1 + P2 + P3 + P4 = P5 + P6 + P7 + P8 = 2$$

With this configuration, we finally have a total of 36 different eigenvalues. Because our purpose is to pass textual information, English letters will occupy 26 of them. In addition to "00111100", there are also 5 eigenvalues to be assigned to commonly used space, comma, period, exclamation mark, and question mark, as shown in Figure 3-22



```
dictionary_alpha = {
    '00110011': 'a', '00110101': 'b', '00110110': 'c',
    '00111001': 'd', '00111010': 'e', '11000101': 'f',
    '01010011': 'g', '01010101': 'h', '01010110': 'i',
    '01011001': 'j', '01011010': 'k', '01011100': 'l',
    '01100011': 'm', '01100101': 'n', '01100110': 'o',
    '01101001': 'p', '01101010': 'q', '01101100': 'r',
    '10010011': 's', '10010101': 't', '10010110': 'u',
    '10011001': 'v', '10011010': 'w', '10011100': 'x',
    '10100011': 'y', '10100101': 'z', '10100110': ' ',
    '10101001': ',', '10101010': '.', '10101100': '!',
    '11000011': '?', '00111100': 'E', '11000110': 'E',
    '11001001': 'E', '11001010': "E", '11001100': "E",
    'error': "/", 'timeout': "_", 'NoPic': "_",
}
```

Figure 3-22. Alphabet dictionary for our system

Synchronization Control

Synchronous control has always been the focus of communication systems, and it is also true in OCC system. Our goal is to establish a communication system that can run continuously in real time. Synchronous control should start before the signal is transmitted and keep running on the background.

In this experimental system, we will synchronization with timestamp detection. After obtaining the timestamp, the system performs mathematical operations on the timestamp. If the result satisfies the condition, go to the next step, otherwise it will continue to operate on the next timestamp.

We set the communication rate to 10 frames per second, so when the cent second in the time-stamp is 0, we will give the camera a signal to start recording. At the transmitter, the system will refresh a few cent seconds in advance because we needed to avoid refreshing in the camera exposure interval. The detailed explanation will be shown in the Figure 3-23.

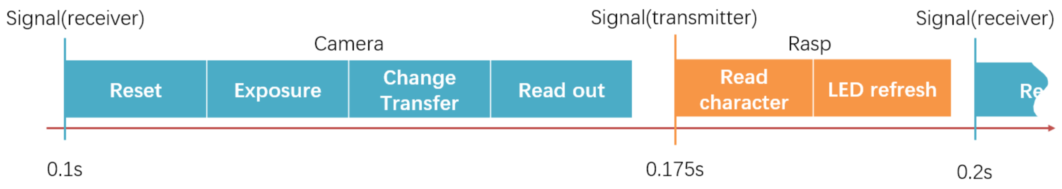


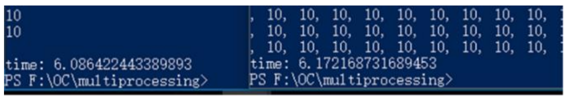
Figure 3-23. Synchronization control

3.3 Multiple Processing

Image process will spend a lot of time and there is not so much time for the system. For example, if the refresh rate of camera is 30fps, which means we have to finish each processing in 0.033s. And if the image is large or our equipment only have a limited computing power, we will miss some flame. To avoid this problem, we need to "make" some time by using multi-progress system.

And the question is, does it save time?

To get the answer, I ran a test using multi-progress system. In this test, system will loop the image process for certain number of times and record the total time. Equipment is my laptop with an i5-2430m. result is shown in Figure 3-24 and Figure 3-25, where the output is a single-process system for a single column 10, and the multi-process system for outputting multiple columns of 10

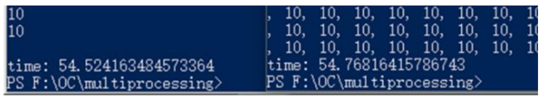


```
10
10

time: 6.086422443389893
PS F:\OC\multiprocessing>
```

```
, 10, 10, 10, 10, 10, 10, 10, 10, 10,
, 10, 10, 10, 10, 10, 10, 10, 10, 10,
, 10, 10, 10, 10, 10, 10, 10, 10, 10,
time: 6.172168731689453
PS F:\OC\multiprocessing>
```

100 times



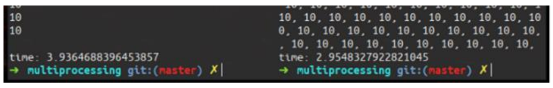
```
10
10

time: 54.524163484573364
PS F:\OC\multiprocessing>
```

```
, 10, 10, 10, 10, 10, 10, 10,
, 10, 10, 10, 10, 10, 10, 10,
, 10, 10, 10, 10, 10, 10, 10,
time: 54.76816415786743
PS F:\OC\multiprocessing>
```

1000 times

Figure 3-24. Under Windows10, two system have a close performance.

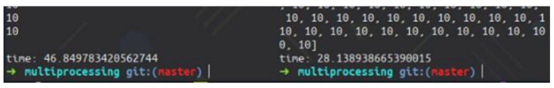


```
10
10

time: 3.9364688396453857
→ multiprocessing git:(master) ✗
```

```
10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10
0, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10,
time: 2.9548327922821045
→ multiprocessing git:(master) ✗
```

100 times



```
10
10

time: 46.849783420562744
→ multiprocessing git:(master)
```

```
 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 1
10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10
0, 10]
time: 28.138938665390015
→ multiprocessing git:(master)
```

1000 times

Figure 3-25. Under Linux system (Arch Linux)

As the result show, it doesn't seem to make much difference to use multiple processes on Windows. While on Linux systems, using multiple processes does save a lot of time. From Figure 3-25, we can see that system uses multi-progress is slightly faster than single-process system when loop 100 times and much faster when loop number is 1000. For Windows system, to create or manage a lot of processes it is a heavy work, and Linux has a better performance in multi-progress. So, this result is not surprising. In general, if we use Linux platform, multi-progress system can help us save much time. If we use windows, there is almost no difference between single process and multi-progress system

Chapter 4. Hardware

The hardware part is the basis of the whole system implementation, so the choice of this part will also affect the performance of the whole system.

4.1 LED Array

LED array is key component of the transmitter, 14 LEDs will be composed together to work as a signal source. For various reasons, infrared LEDs will be used in our system. The model we chose was SI5312-H, which was designed as a remote-control light source, so it matched our needs. It emits a wave peak near 950 nm, a power half angle of plus or minus 8 degrees, and a power dissipation of 145 mW [46]. The driver of the LED array will be completed by a Raspberry Pi 3b. Raspberry Pi is the most popular single-board computer in the world, and its own GPIO can control many electronic devices. At the same time, because it runs the Linux system itself, it is very easy to remotely control it. Because it is outdoors, because it runs outdoors, it will be powered by charging power (Figure 4-1(a)).

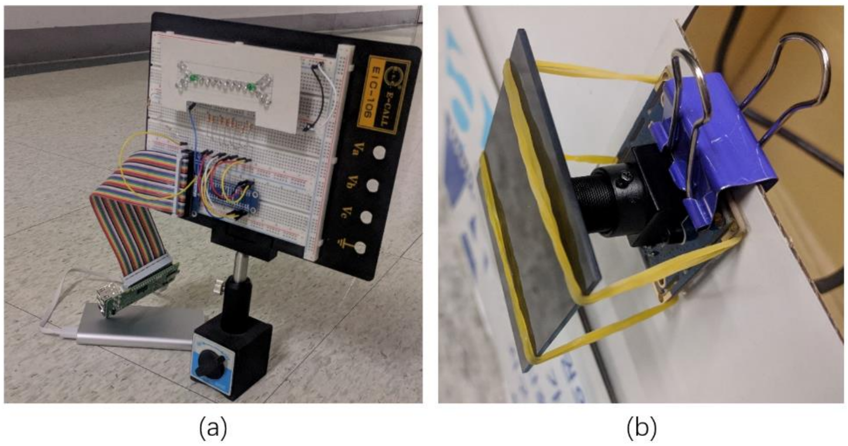


Figure 4-1. An overview of hardware of transmitter and receiver

4.2 Optical Filter

In order to shield the light wave of visible wavelength, we use an infrared optical filter, model FGL780S, made by THORLABS. From the data we got from the official website [47], the filter can filter out light below 780 nm, basically including all the visible light bands. Its transmission curve is shown below (Figure 4-2)

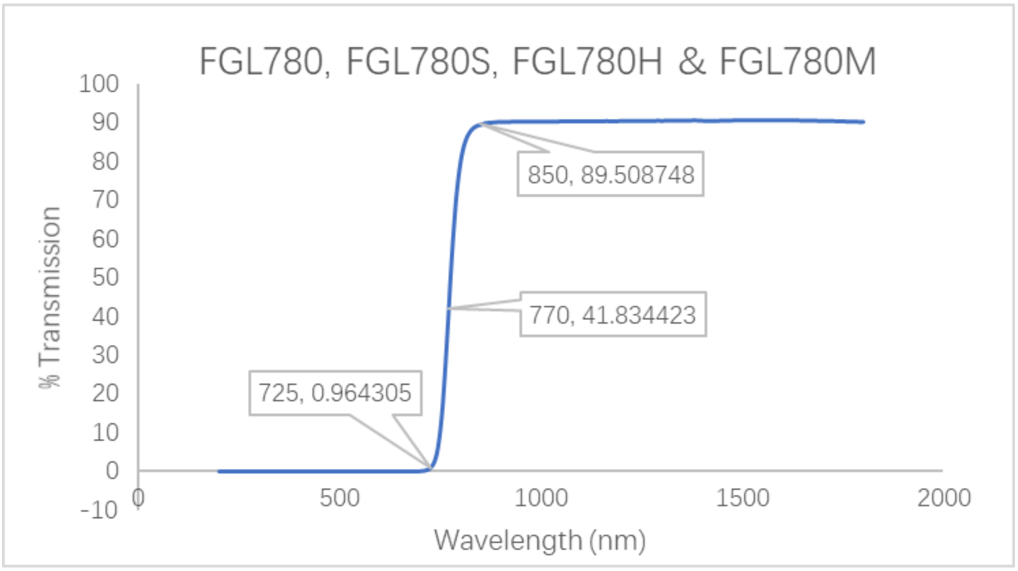


Figure 4-2. The transmission curve of FGL780S

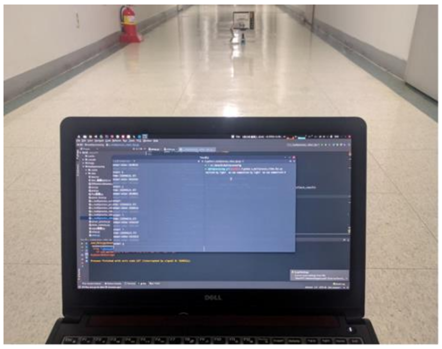
4.3 Camera

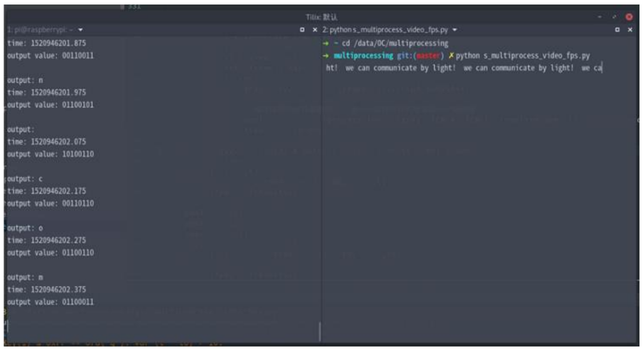
As a main component of receiver, camera will be used to detect light signal. In this system we use a combination of USB web camera (USBFHD01M) and an optical lens (16mm focal length,20° viewing angle). Also, an optical filter will be used to reduce the interference caused by ambient light. USBFHD01M is a web camera with UVC support, which means you can control it by OpenCV directly and no need to install some other drivers. It uses Omnivision OV2710 as the image sensor and can stream in MJPEG or YUY2 formats. Most importantly, it can stream images in 60 frames at

1280×720 resolution. This camera is driven by a PC and all image processing and data processing will be completed by the PC program. Hardware setting of camera is shown in Figure 4-1(b).

Chapter 5. Experiment Results & Analysis

We did this experiment in the corridor (shown in Figure 5-1). The distance between transmitter and receiver is 5m, and the message we try to transmit is a sentence consist 30 characters (space is included) " we can communicate by light! ". transmission speed is 10 letters per second and the transmitter will keep trying until we make it stop. We run the experiments for 5 times, each time system will keep running for 10 seconds. So, every time we finish the experiment, we will receive 1000 letters. We will analyze the operation of the system by analyzing these letters. Transmission errors include two situations: missing frames and error frame. Missing frames means that the system have missed a frame, which makes the information we receive incomplete. The wrong frame indicates that the image has been processed without the correct result.





```
time: 1520946201.875
output value: 00110011

output: n
time: 1520946201.975
output value: 01100101

output:
time: 1520946202.075
output value: 10100110

output: c
time: 1520946202.175
output value: 00110110

output: o
time: 1520946202.275
output value: 01100110

output: m
time: 1520946202.375
output value: 01100011
```

```
→ ~ cd /data/OC/multiprocessing
→ multiprocessing git:(master) ✗ python s_multiprocess_video_fps.py
 ht!  we can communicate by light!  we can communicate by light!  we ca
```

Figure 5-1. Experiment environment and output in the terminal.

5.1 Result

As the result shown in Table 4, the number of missing frames was not significant, with an average of 3.2 times. The number of occurrences of erroneous frames fluctuated significantly, with an average of 7.6 frames. The final total error is 10.8 per thousand frames. In conclusion, we can get the whole sentence at the most time and the average error rate is about 1%.

Table 4. Result

Experiment	1	2	3	4	5	Average
Missing Frames	3	4	3	2	4	**3.2**
Error Frames	0	14	10	10	4	**7.6**
Error Rate	**0.3%**	**1.8%**	**1.3%**	**1.2%**	**0.8%**	**1.08%**

5.2 Analysis

Before analyzing the received results, we need to analyze the sent information first. Before analyzing the received results, we need to analyze the information sent. In this experimental system, the message we send is a sentence " we can communicate by light! ". The characters and numbers it contained are as follows (Table 5):

Character	space	c	n	m	a	t	e	i	w	y	h	b	u	g	!	o	l	SUM
Times	6	3	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	30

Table 5. Characters in the sentence, different background colors represent different occurrences

Suppose the occurrence of the error is random. If the number of occurrences of a letter is higher, the number of errors should be higher when the number of errors is counted. For this idea, we divided the characters into three groups: “spaces” is the only one which appeared six times, marked with red; characters that appeared 2 to 3 times were grouped and marked with yellow; The characters appeared only once are grouped together and are identified by white background, as shown in Table 5. Next, we will analyze the missing and error frames from both the characters and the distribution in the timeline.

Characters

The statistical results of the number of occurrences of characters in lost frames and error frames are as follows (Table 6):

Missing Frame Statistics

Character	Times
space	3
g	3
e	2
y	2
u	1
b	1
w	1
l	1
a	1
i	1
SUM	**16**

Error Frame Statistics

Character	Times	Eigenvalue (“□”:1, “ _ ”:0)
t	7	□ _ _ □ _ □ _ □
m	6	_ □ □ _ _ _ □ □
e	6	_ _ □ □ □ _ □ _
g	4	_ □ _ □ _ _ □ □
space	3	□ _ □ _ _ □ □ _
y	3	□ _ □ _ _ _ □ □
n	2	_ □ □ _ _ □ _ □
b	2	_ _ □ □ _ □ _ □
a	2	_ _ □ □ _ _ □ □
w	1	□ _ _ □ □ _ □ _
c	1	_ _ □ □ _ □ □ _
i	1	_ □ _ □ _ □ □ _
SUM	**38**	

Table 6. Statistics result on the number of occurrences of characters in missing frames and error frames

Missing Frame

As can be seen from Table 4, the number of lost frames in the five sets of experiments is relatively stable, about 3‰. For characters, we can see that in our experimental results (Missing Frame Statistics), 10 characters have been missed. Because the number of missing frames is not very large, only 16 of the 5,000 characters we tested were missed; at the same time, the number of times each character was lost was not much different. And similar to the previous assumptions, because of the large number, "space" is lost the most. So there seems to be no strong connection between the lost frames and the categories of characters, or, there are no cases where certain characters are particularly easy to be missed.

Error Frame

Because the number of error frames is over twice that of the former, we seem to be able to see some problems. Unexpectedly, "space" is not the most wrong this time. If you consider the number of characters in our data, the most common characters that are wrong are "g", "t", "m", "e". For such a result, we have such speculation for the reason:

- Hardware problem (LED problem)
- Algorithm problem (less recognition of a certain pattern)

Since both of these problems are related to the eigenvalue, we will analyze the eigenvalue next. We add the eigenvalues of the error frame and we can get the following results (Table 7, Table 8).

	P1	P2	P3	P4	P5	P6	P7	P8
SUM	14	13	25	24	7	14	29	26

Table 7. The number of occurrences of each detect point in the Data_LED in the error frames

	P1	P2	P3	P4	P5	P6	P7	P8
rate	1.17	1.30	1.19	1.41	1.40	0.70	1.26	2.17

Table 8. Considering the total number of uses，occurrences of each detect point in the Data_LED in the error frames

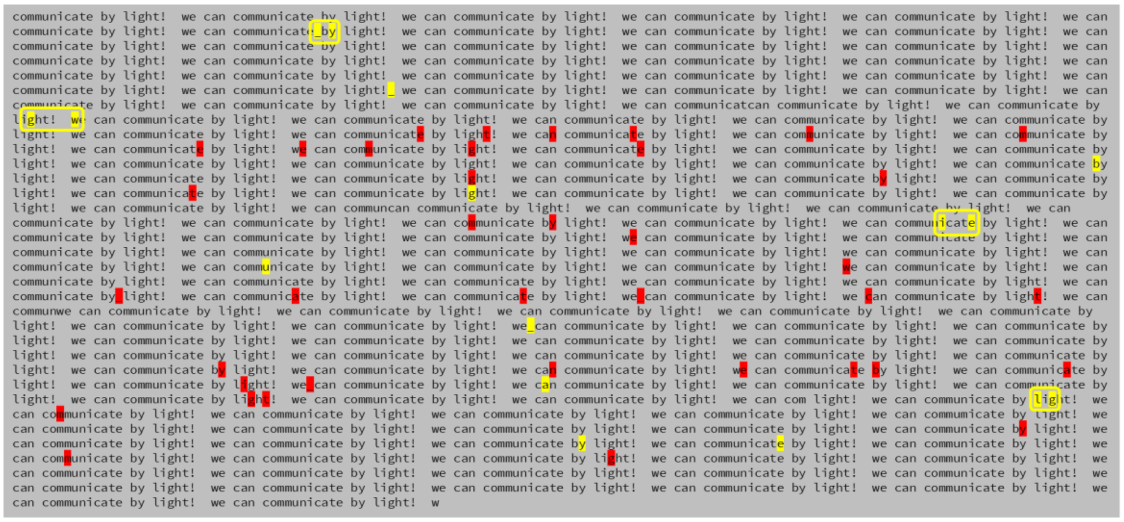
Table 8 is the result of dividing the value of Table 7 by the number of occurrences of the character itself. As we have done before, we divide them into three groups based on the number of occurrences. It can be seen from Figure 8 that if the value of P4, P5, and P8 in the eigenvalue of the character is 1, there is a higher risk of getting the error frame. In fact, I am little surprised by this result of P4 and P5. Because the principle of our positioning is to locate the two ends first, and then to locate the middle, in this case, in fact, the middle position should be the most stable position. For P8, I think it is more likely to be a problem in the LED array, resulting in a higher error rate when using the P8 position. In fact, it is theoretically possible to use LEDs in both P1 and P8 positions to cause positioning errors because they are closer to Position_LED; this is why I have to set a space between the two ends of the Data_LED.

	P1	P2	P3	P4	P5	P6	P7	P8
rate	0.44	0.25	0.36	0.38	0.33	0.33	0.43	

Table 9. The result of removing all characters using P8 position and the character "e"

If we removing all characters using P8 position and the character "e", we can see P3-P6 have a similar error rate, while P1 has the highest error rate and P7 is the second. Someone may ask why we need to eliminate the character "e". While that should blame the eigenvalue of "e": _ _ □□□ _ □ _ (00111010), which has three "1" positions at the same time are adjacent. Because there are three LEDs in our template image to form a straight line, when three "1" neighbors appear, it may cause interference to the positioning and eventually lead to recognition failure. And in fact, the system has made many mistakes in the recognition of the character "e". As can be seen from the results, using the detection on both sides is actually more likely to cause recognition errors.

Distribution in the Timeline



```
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate_by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!_ we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicatcan communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communcan communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can
communicate by_light!  we can communicate by light!  we can communicate by light!  we_can communicate by light!  we can communicate by light!  we can
communwe can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we_can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we_can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by
light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can com light!  we can communicate by light!  we
can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we
can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we
can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we
can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we
can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we
can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we can communicate by light!  we
can communicate by light!  we can communicate by light!  w
```

Figure 5-2. The total received data set is arranged according to the receiving time.

In order to analyze the time and concentration of missing frames and error frames, we arrange all the data according to the receiving time. The final result can be seen in the Figure 5-2, while the missing frame is marked with yellow, and the error frame is marked with red.

Missing Frame

As can be seen from the figure, the distribution of missing frames is more random, but there is a tendency to appear intensively (identified by a yellow rectangular frame in the figure). So, my guess is that the missing frames may be due to systemic reasons. Because image processing needs to occupy a lot of computing resources, the processing time of each frame is actually different. In order to avoid waiting for the entire system due to the long processing time of a certain frame of image processing, we set up a multi-process system to process multiple frames of images in parallel. This is equivalent to an insurance measure to ensure that the system can receive all frames from the transmitter.

But there are always times when you are unlucky. If all processes in the process pool exceed the specified processing time (0.1s for 10fps), then the system has no extra resources to process the new frame. So, when the time is up to receive the next frame, the sync control module will not send a signal to the camera. This leads to the occurrence of missing frames. If a process is stuck in a certain frame for a long time, the probability of the system missing frames increases. If we can find a faster, more resource-efficient target location algorithm, we can reduce the number of lost frames to a large extent. Also, reducing the timeout detection time of a single process can be another solution, but this will increase the number of frames that process timeouts.

Error Frame

The distribution of error frames is equally random, but the distribution is different in different experiments. Because of this, I feel that the error frame is more related to the environment and hardware, so the results of different experiments will have such a big difference.

What Can We Research?

To protocol:

- Using P1 and P8 in Data_LED may result in higher detection failure rates
- 3 LEDs adjacent to each other may result in higher detection failure rates

To system design:

- Missing frame is a systematic reason, but can be further balanced

To hardware:

- There may be a problem with the P8 LED in this experiment.

Future Work

- Modify Protocol
- Find a suitable threshold for timeout detection
- Modify hardware
- Find a faster, more resource-efficient target location algorithm

Chapter 6. Conclusion

Starting from an optical wireless communication system which can be used for V2V communication, an optical camera communication system that can automatically locate the light source and output data real-time has been developed. This system has used infrared LEDs and a webcam as transmitters and receiver and is capable of communicating at a rate of 10 letters per second with a distance of 5 meters. According to the experimental results, the bit error rate of the system is about 1.08%.

This result is achieved under limited hardware conditions, and the software has some areas for improvement. For example, the match template algorithm used requires a lot of computing resources. It is believed that if there are improvements to the software and hardware, there is still much room for improvement in this system. According to the results analysis, future work will be concentrated in the following points:

- Modify Protocol
- Find a suitable threshold for timeout detection
- Modify hardware
- Find a faster, more resource-efficient target location algorithm

References

[1] A. O. Goushcha, B. Tabbert, “On response time of semiconductor photodiodes,” *Optical Engineering,* vol. 56, no. 9, Sep 2017.

[2] A. T. Hussein and J. M. H. Elmirghani, "10 Gbps mobile visible light communication system employing angle diversity, imaging receivers, and relay nodes," in *IEEE/OSA Journal of Optical Communications and Networking*, vol. 7, no. 8, pp. 718-735, August 2015.

[3] A. T. Hussein, M. T. Alresheedi and J. M. H. Elmirghani, "25 Gbps mobile visible light communication system employing fast adaptation techniques," *2016 18th International Conference on Transparent Optical Networks (ICTON)*, Trento, 2016, pp. 1-7.

[4] IEEE Standard for Local and Metropolitan Area Networks--Part 15.7: Short-Range Wireless Optical Communication Using Visible Light," in *IEEE Std 802.15.7-2011*, vol., no., pp.1-309, 6 Sept. 2011

[5] R. D. Roberts, “Kick-starting the VLC market via optical camera communications,” in *Proc. International Conference and Exhibition on Visible Light Communications*, Yokohama, Japan, Oct. 2015.

[6] T. Nguyen, A. Islam, T. Hossan and Y. M. Jang, "Current Status and Performance Analysis of Optical Camera Communication Technologies for 5G Networks," in *IEEE Access*, vol. 5, pp. 4574-4594, 2017.

[7] Le, Nam Tuan, M. A. Hossain, and Y. M. Jang. "A survey of design and implementation for optical camera communication," *Signal Processing: Image Communication*, vol. 53, pp. 95-109, April 2017.

[8] Do, Trong-Hop and Myungsik Yoo. “Performance Analysis of Visible Light Communication Using CMOS Sensors,” *Sensors (Basel, Switzerland)* vol. 16,3 pp. 309, 29 Feb. 2016.

[9] Y. Wang, R. Li, Y. Wang and Z. Zhang, "3.25-Gbps visible light communication system based on single carrier frequency domain equalization utilizing an RGB LED," *OFC 2014*, San Francisco, CA, 2014, pp. 1-3.

[10] M. Y. Abualhoul, O. Shagdar and F. Nashashibi, "Visible Light inter-vehicle Communication for platooning of autonomous vehicles," *2016 IEEE Intelligent Vehicles Symposium (IV)*, Gothenburg, 2016, pp. 508-513.

[11] P. H. Pathak, X. Feng, P. Hu and P. Mohapatra, "Visible Light Communication, Networking, and Sensing: A Survey, Potential and Challenges," in *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2047-2077, Fourthquarter 2015.

[12] B. Fahs, M. Romanowicz and M. M. Hella, "A Gbps Building-to-Building VLC Link Using Standard CMOS Avalanche Photodiodes," in *IEEE Photonics Journal*, vol. 9, no. 6, pp. 1-9, Dec. 2017, Art no. 7907709.

[13] H. Le Minh *et al*., "A 1.25-Gb/s Indoor Cellular Optical Wireless Communications Demonstrator," in *IEEE Photonics Technology Letters*, vol. 22, no. 21, pp. 1598-1600, Nov.1, 2010.

[14] Urabe, Hideki, et al. "High data rate ground-to-train free-space optical communication system." *Optical Engineering* vol. 51, no. 3, Mar. 2012.

[15] C. Chang *et al.*, "A 100-Gb/s Multiple-Input Multiple-Output Visible Laser Light Communication System," in *Journal of Lightwave Technology*, vol. 32, no. 24, pp. 4723-4729, 15 Dec.15, 2014.

[16] Z. Ong, V. P. Rachim and W. Chung, "Novel Electromagnetic-Interference-Free Indoor Environment Monitoring System by Mobile Camera-Image-Sensor-Based VLC," in *IEEE Photonics Journal*, vol. 9, no. 5, pp. 1-11, Oct. 2017, Art no. 7907111.

[17] Z. Ong and W. Chung, "Long Range VLC Temperature Monitoring System Using CMOS of Mobile Device Camera," in *IEEE Sensors Journal*, vol. 16, no. 6, pp. 1508-1509, Mar.15, 2016.

[18] I. Takai, T. Harada, M. Andoh, K. Yasutomi, K. Kagawa and S. Kawahito, "Optical Vehicle-to-Vehicle Communication System Using LED Transmitter and Camera Receiver," in *IEEE Photonics Journal*, vol. 6, no. 5, pp. 1-14, Oct. 2014, Art no. 7902513.

[19] G. K. H. Pang and H. H. S. Liu, "LED location beacon system based on processing of digital images," in *IEEE Transactions on Intelligent Transportation Systems*, vol. 2, no. 3, pp. 135-150, Sept. 2001.

[20] H. C. N. Premachandra *et al.*, "High-speed-camera image processing based LED traffic light detection for road-to-vehicle visible light communication," *2010 IEEE Intelligent Vehicles Symposium*, San Diego, CA, 2010, pp. 793-798.

[21] S. Arai, Y. Shiraki, T. Yamazato, H. Okada, T. Fujii and T. Yendo, "Multiple LED arrays acquisition for image-sensor-based I2V-VLC using block matching," *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*, Las Vegas, NV, 2014, pp. 605-610.

[22] C. Danakis, M. Afgani, G. Povey, I. Underwood and H. Haas, "Using a CMOS camera sensor for visible light communication," *2012 IEEE Globecom Workshops*, Anaheim, CA, 2012, pp. 1244-1248.

[23] C. W. Chow, C. Y. Chen, & S. H. Chen, "Visible light communication using mobile-phone camera with data rate higher than frame rate. " *Optics express*, vol. *23*, no. 20, pp. 26080-26085.

[24] C. Chow, C. Chen and S. Chen, "Enhancement of Signal Performance in LED Visible Light Communications Using Mobile Phone Camera," in *IEEE Photonics Journal*, vol. 7, no. 5, pp. 1-7, Oct. 2015, Art no. 7903607.

[25] T. Nguyen, Chang Hyun Hong, Nam Tuan Le and Y. M. Jang, "High-speed asynchronous Optical Camera Communication using LED and rolling shutter camera," *2015 Seventh International Conference on Ubiquitous and Future Networks*, Sapporo, 2015, pp. 214-219.

[26] J. H. Kim, S. Y. Kang, and S. T. Lee. "VLC-based location data transferal for smart devices." *Optical Switching & Networking*, vol. 23, no. 3, pp. 250-258, Jan.3, 2016.

[27] Y. Li, Z. Ghassemlooy, X. Tang, B. Lin and Y. Zhang, "A VLC Smartphone Camera Based Indoor Positioning System," in *IEEE Photonics Technology Letters*, vol. 30, no. 13, pp. 1171-1174, 1 July1, 2018.

[28] P. Luo, Z. Ghassemlooy, H. Le Minh, X. Tang and H. Tsai, "Undersampled phase shift ON-OFF keying for camera communication," *2014 Sixth International Conference on Wireless Communications and Signal Processing (WCSP)*, Hefei, 2014, pp. 1-6.

[29] R. D. Roberts, "Undersampled frequency shift ON-OFF keying (UFSOOK) for camera communications (CamCom)," *2013 22nd Wireless and Optical Communication Conference*, Chongqing, 2013, pp. 645-648.

[30] N. Liu, J. Cheng and J. F. Holzman, "Undersampled differential phase shift on-off keying for optical camera communications with phase error detection," *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, Valencia, 2017, pp. 916-921.

[31] P. Luo *et al*., "Experimental Demonstration of a 1024-QAM Optical Camera Communication System," in *IEEE Photonics Technology Letters*, vol. 28, no. 2, pp. 139-142, 15 Jan.15, 2016.

[32] P. Luo *et al*., "Experimental Demonstration of RGB LED-Based Optical Camera Communications," in *IEEE Photonics Journal*, vol. 7, no. 5, pp. 1-12, Oct. 2015, Art no. 7904212.

[33] S. Vitek, J. Libich, P. Luo, et al, "Influence of Camera Setting on Vehicle-to-Vehicle VLC Employing Undersampled Phase Shift On-Off Keying." *Radioengineering*, vol. 26, no. 4, pp. 946-953, 2017.

[34] S. Chen and C. Chow, "Color-Shift Keying and Code-Division Multiple-Access Transmission for RGB-LED Visible Light Communications Using Mobile Phone Camera," in *IEEE Photonics Journal*, vol. 6, no. 6, pp. 1-6, Dec. 2014, Art no. 7904106.

[35] K. Liang, C. W. Chow, and Y. Liu, "RGB visible light communication using mobile-phone camera and multi-input multi-output, " *Optics express*, vol. 24, no. 9, pp. 9383-9388, 2016.

[36] T. Hao, R. Zhou, and G. Xing, "COBRA: color barcode streaming for smartphone systems." *Proceedings of the 10th international conference on Mobile systems, applications, and services*, pp. 85-98, 2012.

[37] W. Hu, G. Hao, and Q. Pu, "Lightsync: Unsynchronized visual communication over screen-camera links," *Proceedings of the 19th annual international conference on Mobile computing & networking*, pp 15-26, 2013.

[38] A. Wang, S. Ma, C. Hu, et al, "Enhancing reliability to boost the throughput over screen-camera links," *Proceedings of the 20th annual international conference on Mobile computing and networking*, pp. 41-52, 2014.

[39] Q. Wang, M. Zhou, K. Ren, T. Lei, J. Li and Z. Wang, "Rain Bar: Robust Application-Driven Visual Communication Using Color Barcodes," *2015 IEEE 35th International Conference on Distributed Computing Systems*, Columbus, OH, 2015, pp. 537-546.

[40] Y. Zeng *et al.*, "Visible light communication system for mobile device," *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN),* Shanghai, 2014, pp. 26-28.

[41] N. Trang, T. L. Nam and M.J. Yeong, "Practical design of Screen-to-Camera based Optical Camera Communication," *2015 International Conference on Information Networking (ICOIN)*, Cambodia, 2015, pp. 369-374.

[42] Johnson, Stephen (2006). Stephen Johnson on Digital Photography. O'Reilly. ISBN 0-596-52370-X.

[43] https://docs.opencv.org/3.4.3/ada_threshold.jpg

[44] F. Song, H. An, C. G. Lee,"LED Array Positioning and Correction in UAV-Ground Communication," in Proc. *The 22nd International Computer Science and Engineering Conference (ICSEC),* Chiang Mai, Thailand, Nov. 2018.

[45] E. Rublee, V. Rabaud, K. Konolige and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," *2011 International Conference on Computer Vision*, Barcelona, 2011, pp. 2564-2571.

[46]https://www.auk.co.kr/Common/DownloadFile?option=inline&option2=productPdfByPartNo&partNo=SI5312-H

[47] https://www.thorlabs.com/images/popupimages/FGL780.xlsx

Abstract

Development of an Optical Camera Communication System Based on a Webcam and Infrared LEDs

Fan Song
Advisor: Prof. Chung Ghiu Lee, Ph.D.
Department of Electronic Engineering,
Graduate School of Chosun University

In order to solve the shortage of communication channels, wireless optical communication has been widely developed in recent years. In this thesis, starting from an optical wireless communication system which can be used for V2V (Vehicle to vehicle) communication, a real-time optical camera communication system using USB web camera and IR-LED is described.

This OCC system can automatically locate the light source and output data real-time. A LED array was built by low power IR LEDs used as the transmitter to transmit information, meanwhile, a USB web-camera is used as the receiver. The web-camera is connected to the computer and the computer will be responsible for the rest of the image processing. In the image processing process, the system will complete noise reduction, target positioning, positioning correction, data extraction. In order to avoid waiting for the entire system due to the long processing time of a certain frame of image processing, a multi-process system which can process multiple frames of images in parallel is used.

According to the experimental results, the system can communicate at a distance of 5 meters, transmitting ten characters per second, and the average bit error rate is 1.08%.

Acknowledgment

The completion of this thesis is due to steadfast support and encouragement from many people around me.

First and foremost, I would like to express my gratitude to my advisor, Professor Chung Ghiu Lee, whose patient guidance, valuable suggestions and constant encouragement allowed me to successfully complete my thesis. His academic literacy and professionalism inspire me both in academic study and daily life. His magnanimous nature with his students makes working with him very easy and beneficial.

I would also like to express my sincere gratitude towards my friends and co-workers who helped me both in study and daily life: Jiseong Jeong, Dongil Jeong and my other lab friends. I am grateful for the care they have taken over the years.

Studying abroad is not a simple thing, and without help of many people, I could not continue to the present. Thanks to all the friends I know here, your help and support is to me like a warm coat in the cold winter.

Last but not least, I would like to express my special thanks to my family, whose care and support motivates me to continue forward and become a better person today and tomorrow

SONG FAN