IMAGE PROCESSING ALGORITHMS FOR REALIZING A SEAMLESS MULTI-PROJECTION SCREEN

IMAGE PROCESSING ALGORITHMS FOR REALIZING A SEAMLESS MULTI-PROJECTION SCREEN

BY

XIUXIAN YE, B.Eng.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2020) McMaster University
(electrical & computer engineering) Hamilton, Ontario, Canada

TITLE: Image Processing Algorithms for Realizing a Seamless Multi-projection Screen

AUTHOR: Xiuxian Ye
B.Eng. (Measuring and Control Technology with High Precision Instrument),
McMaster University, Hamilton, Canada

SUPERVISOR: Jun Chen

NUMBER OF PAGES: ix, 63

Abstract

Nowadays, screens are very common in our daily life. There are several different kinds of screens, LCD, LED, OLED, ULED and so on. LCD screens can display high-resolution pictures while LED has advantages of low energy consumption and wider color range. This project has two goals. The first one is to achieve a seamless display screen which consists of 9 LED backlit LCD boards. The second goal is to improve image quality, which is enabled by the combination of LED and LCD. There are two main problems that need to be solved in this project. The first problem is brightness correction. Because of the projection method and the distance between lights and final screen, there are different kinds of overlapping situations and distinct lines on screen. The other one is the combination of LED and LCD. The algorithms need to be developed to ensure that RGB LEDs and LCD panels display the same picture and to address some problems caused by the LCD module.

Acknowledgements

My deepest gratitude goes first and foremost to Dr. Jun Chen, my supervisor, for his constant encouragement and guidance. He give me many instructive advice and useful suggestions when I was stuck by some difficult questions. I am deeply grateful for his patience and illuminating instruction. Secondly, I would like to express my gratitude to Professor Adrian Kitai. He always explains patiently to me when I am not familiar with the hardware part and internal circuit structure. Without his help, I can not figure out problems about the internal structure of the LCD board and find reasons for failing to present better results. Finally, I would like to thanks to my parents. They give great courage to face difficulties and address them. I am indebted to my parents for their continuous support and encouragement.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background of research project

Display technologies have witnessed a rapid development recently. LCD, LED and OLED are among the most commonly used ones. LCD is a kind of screen that uses liquid crystal to control light transmittance. From the perspective of the structure of the liquid crystal display, the LCD liquid crystal panel is composed of two parallel glass plates with a thickness of about 1 mm. And two glass plates are separated by liquid crystal materials. Because the liquid crystal material itself does not emit light, we use LED as its backlight in the project.

LCD can display gentle color without glare, which can make viewers feel more comfortable. And it also has an advantage of high-resolution. However, LCD screen is also restrained by its narrow color range compared with LED's. During the research, it is found that there are many obvious flaws at high resolution. So it is necessary to introduce LED lights as a remedy. This part will be explained in detail in the third part. Since one of the import goal of this project is to minimize power consumption, LED should play a role of increasing brightness while we reducing brightness of LCD. The reason of it is LED has a big advantage of low-energy-consumption.

1.2 Introduction of project

The internal structure of the system model is shown below. It chooses LED as the backlight of LCD module. As shown in Figure 1.1, the whole LCD board is composed of many small panels. Each LCD panel has 4 RGB LEDs located at 4 corners of it.

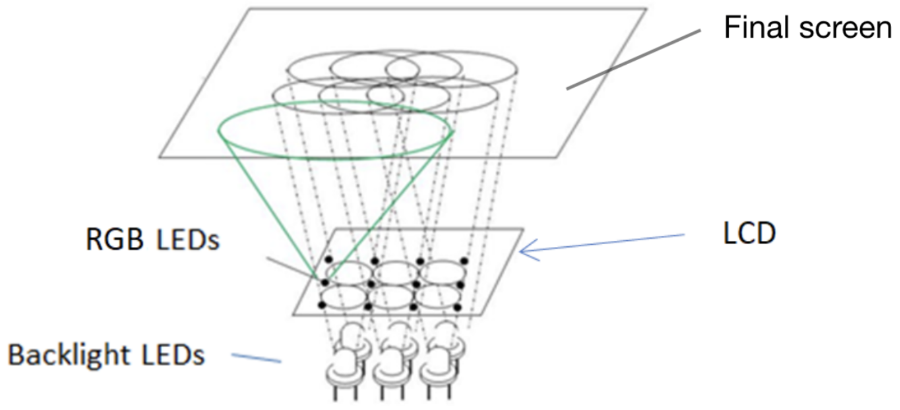


Figure 1.1: System diagram of project

As said before , LCD board consists of multiple square panels. And this is also the key of principle of the tiled video wall. By increasing the number of LCD board, the size of projection screens can become larger and larger. Figure 1.1 demonstrates there is a final screen set at the front of LCD module. Screens specifically designed for use in video walls always have gaps between display areas. So the distance between final screen and LCD board can address the problem of gaps of LCD panels. And the arrangement of each small panel is shown in the Figure 1.2 below.

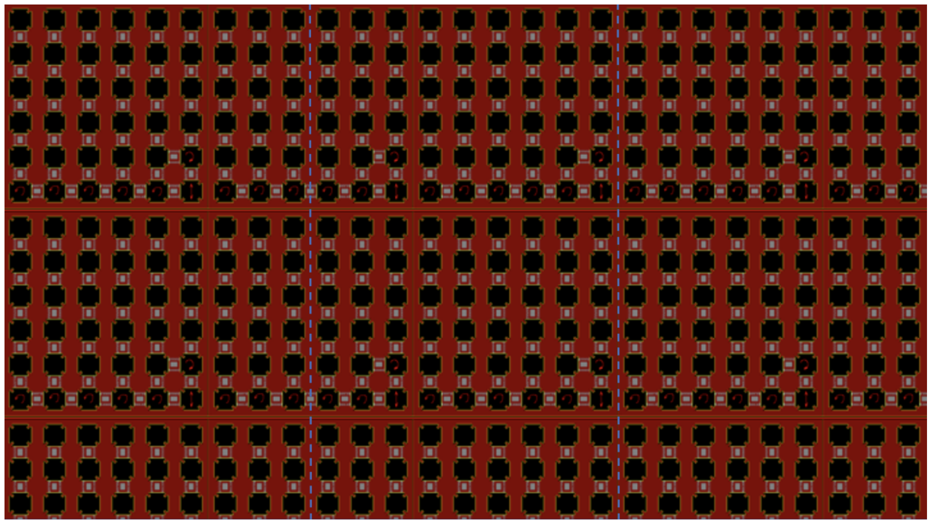


Figure 1.2: Arrangement of each LCD panel

1.3 Statement of main problems

This project takes several steps to completion, which includes, among others, the design of image processing algorithms for LCD panels and the treatment of LED lights. Figure 1.3 demonstrates the procedure of the whole project. The general idea of LCD image processing is inspired by (GUO, 2020).

First of all , we need to determine how many pixels every LCD panel has and calculate the distance between the LCD board and the final screen. Next, it is time to cut the original picture and arrange them in order. And then, we need to generate a new image that has same arrangement as small LCD panels' on LCD board. The most important step of image process on LCD is the correction of brightness. And this is also the first problem which must be solved during the project.

Another part is the process of LED. Only if we determine the coordinates of every LED on final picture, we can capture LED spots on this image and get RGB data inside light spots. After doing these steps, every RGB LED can be judged whether

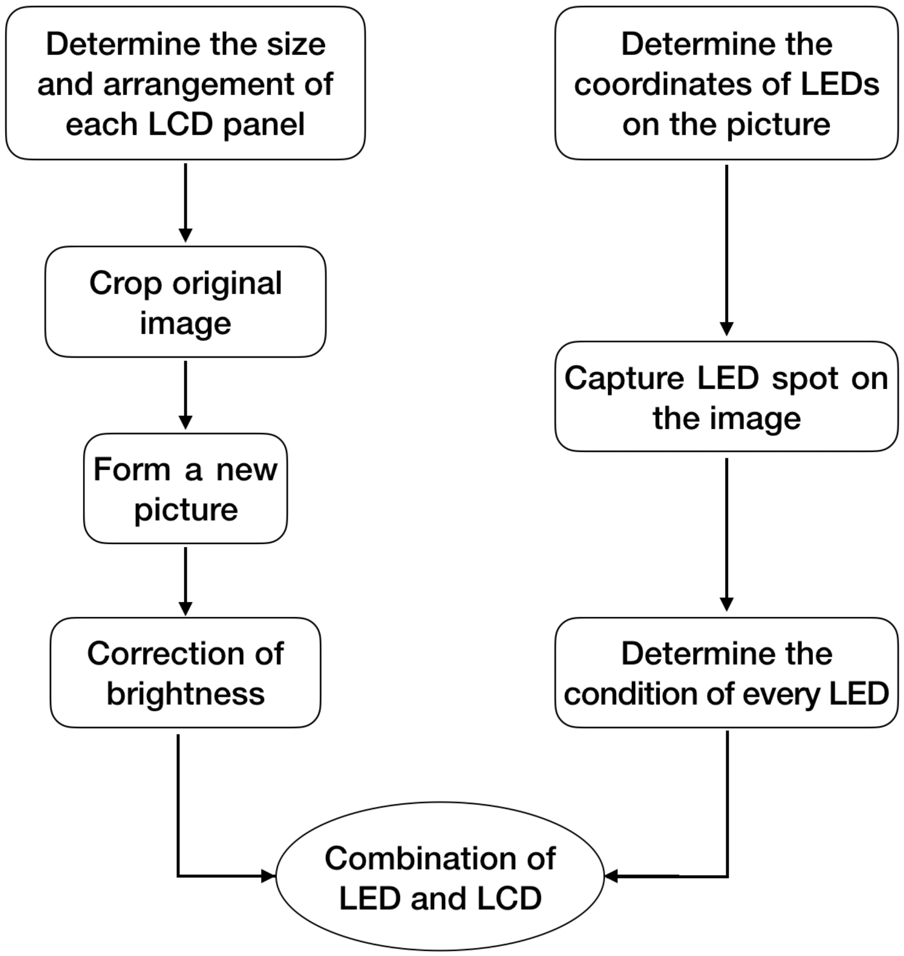

Determine the size and arrangement of each LCD panel
Crop original image
Form a new picture
Correction of brightness
Determine the coordinates of LEDs on the picture
Capture LED spot on the image
Determine the condition of every LED
Combination of LED and LCD

Figure 1.3: The project flow diagram

it should be lit. Finally, there comes the second main problem, the combination of LCD and LED.

1.3.1 First problem

As said before, the first problem is the correction of brightness (Majumder *et al.*, 2000). One essential purpose is to eliminate gaps between these LCD panels, which could be achieved by setting distance of LCD board and final screen. By increasing the quantity of LCD panels, the size of display can become infinite. And this is also one major advantage of this project. But then again, this kind of set will cause some problems on brightness of picture shown. Because of the distance between LCD board and final screen, pictures displayed by every LCD panels will be amplified on final screen. And pictures from adjacent LCD panels will overlap on display screen. Figure 1.4 shows the result when overlapping images are displayed by final screen (Prentašić *et al.*, 2010).

According to this image, we can find that some parts of picture are dark and some parts are bright. The bright part means that this place has been overlapped several times by pictures from different panels. On the contrary, some parts of final screen, which only have one image on it will look darker. If we change the distance between LCD board and final screen, or the gap between adjacent LCD panels, the result of overlapping will change obviously.

1.3.2 Remaining problems

We also need to resolve various issues arising from the combination of LDC and LEDs. Firstly, the the value of each LED should be determined. Because LED spot is much bigger than that of LCD, the color of the LED should not be limited to one point to determine. It is important to combine the pixel value of the area around the LED and the size of range depends on the spot size.

Secondly, according to the meaning of adding front LEDs, we are supposed to set boundaries to selectively turn on LED lights. The rule of it also depends on LCD pixel value within the illumination range of the light spot.

After turning on the LEDs in front of LCD panels, there are two main problems that we should think about. On the final glass screen, the image should have smooth brightness. However, if we add LEDs directly without adjusting the brightness of LCD image, the brightness of the whole final image will look very uneven. Besides, there will be errors between the measurement results and the actual position of the LED. The content of the LED light may not correspond to the part of the LCD picture.

The above problems need to be solved by constantly adjusting parameters through algorithms.

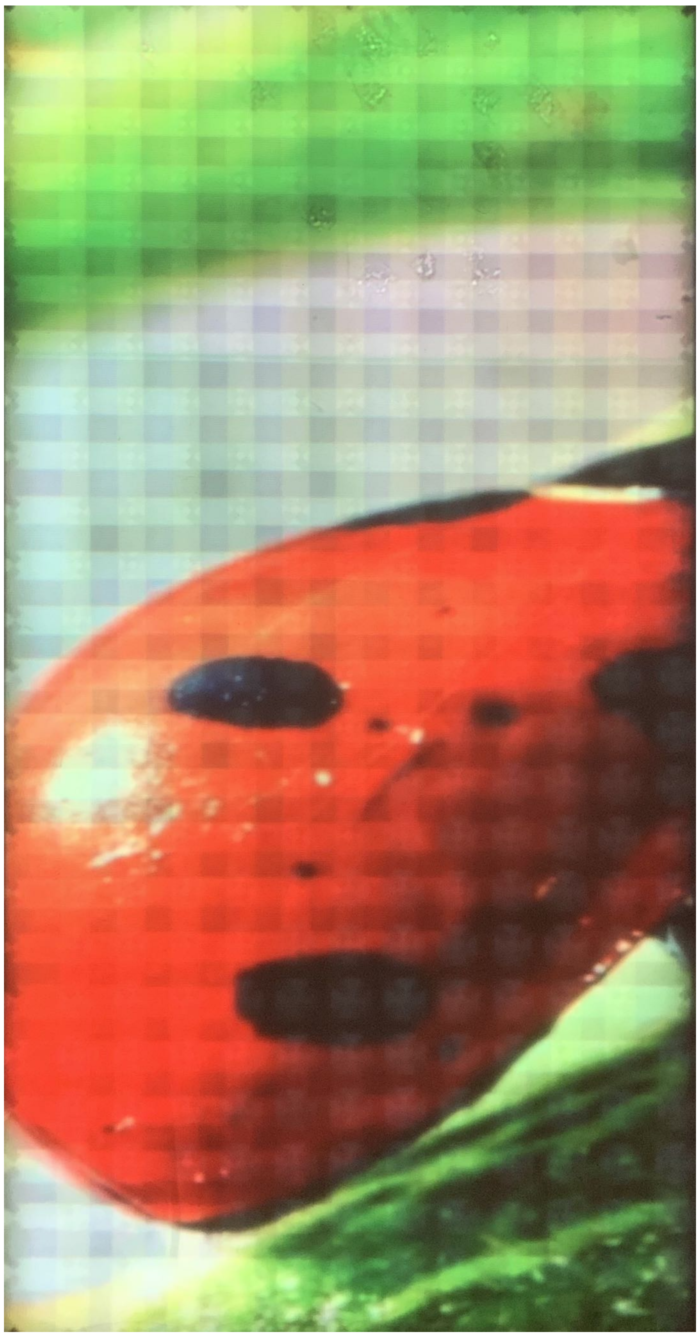


Figure 1.4: Result of overlapping

Chapter 2

Brightness correction

2.1 Data preparation

Different structure and arrangement of LCD panels and LED will cause different result. In order to confirm the case of image overlapping. We need to prepare data of instruments firstly. Every LCD panel is in a square shape and there is a gap between contiguous panel. After measuring the length of each panel and distance of gap, we can figure out the size of LCD panel in pixel. And only if we get data in pixel, we can match the original picture with LCD board, since they are all calculated by pixels. The relationship between distance and pixels is $Distance*Resolution = size\ in\ pixel$.

The distance between LCD panel and final screen determines the magnification of image, and it influences the situation of overlapping as well. After knowing the size of the object and image, we can calculate the data of magnification (Figure 2.5) by $Magnification = \frac{image\ size}{object\ size}$. The object size means the size of LCD panel and image size means the size of image displayed on final screen.

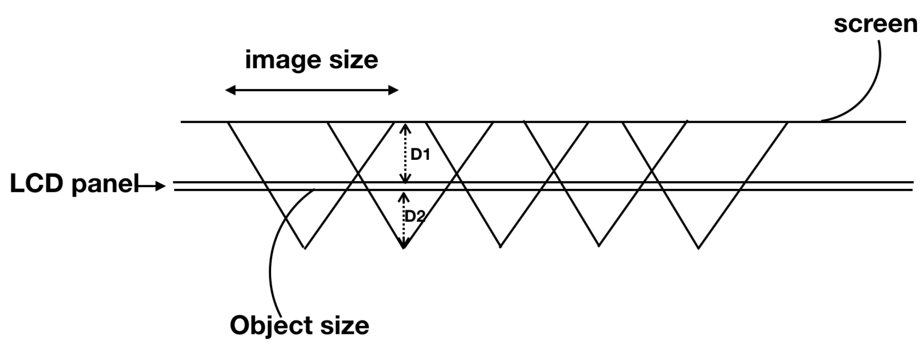


Figure 2.5: Calculation of magnification

2.2 Crop images

Every LCD panel will show one part of original image, and different parts will display on same final screen. Our objective is to make different parts of images be stitched into a complete picture on the final screen even if they will overlap on it.

First of all, we need to determine the unique part of every LCD panel. The unique part means that this part of image will not be overlapped by other pictures displayed by other LCD panels. If each LCD panel only shows its unique part, then those parts will just connect seamlessly on screen theoretically. However, because of the deficiency in hardware, there are obvious lines between different parts of images as shown in Figure 2.6. And that is why each LCD panel should show different images with overlapping part. Overlapping not only can address the gap between LCD boards, but also can make images look like a complete picture without distinct lines and point.

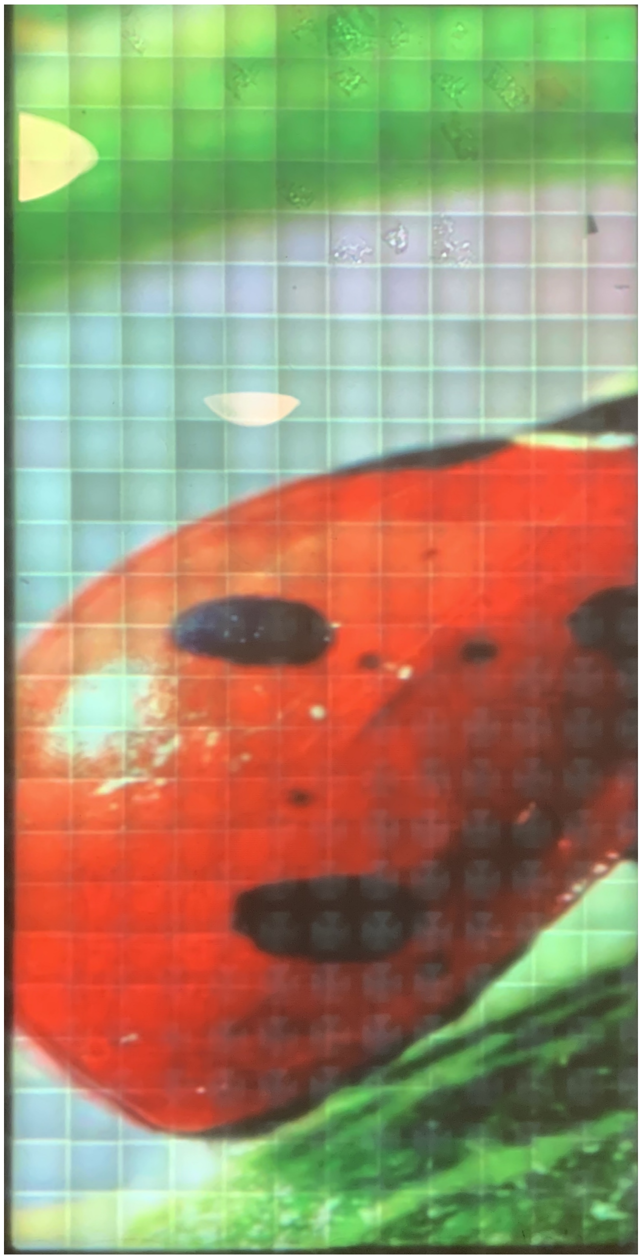


Figure 2.6: Non-overlapping display results

So every LCD panel must has its unique part and overlapping part, which is illustrated in Figure 2.7 below. And an image displayed by the LCD panel also has its unique part and overlapping part. As it said before, unique parts of the images

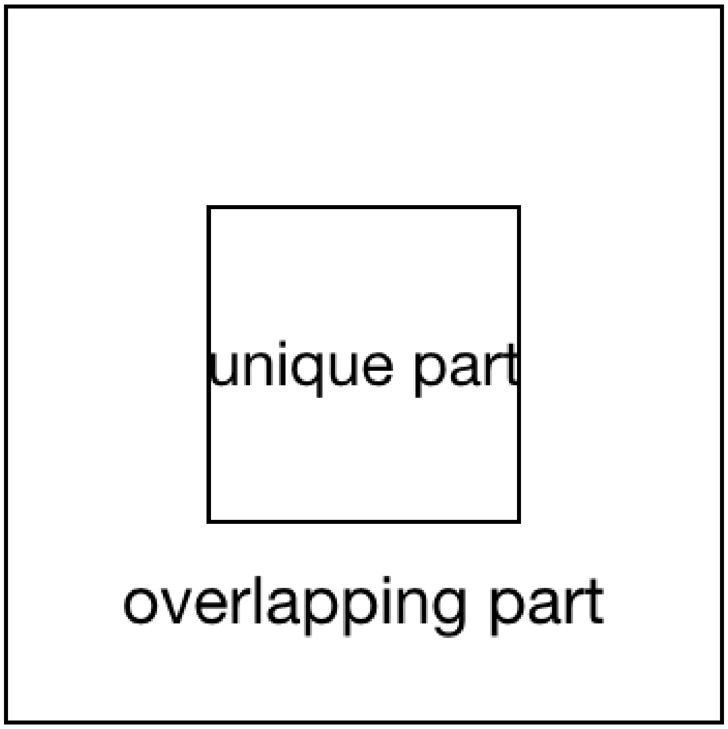


Figure 2.7: Image displayed by the LCD panel

from different LCD panels can just form a complete image seamlessly in theory. So the size of unique part of every LCD panel can be determined by this idea. We can divide the final screen evenly according to the distribution of LCD panels on LCD board. In other words, dividing final screen is dividing original picture. It is because that original image will be showed on final screen. And the amount of pixels of original image and pixels of the picture on final screen is same, including their RGB data of every pixel. As for overlapping part, it is the rest part of the LCD panel.

After getting the size of unique part and LCD panel, the segmentation of original image can be started. Each image which is cropped on original image has same size of the LCD panel. And the stride of every division is determined by the size of unique size. The diagram below(Figure 2.8) shows the way of segmentation of image.

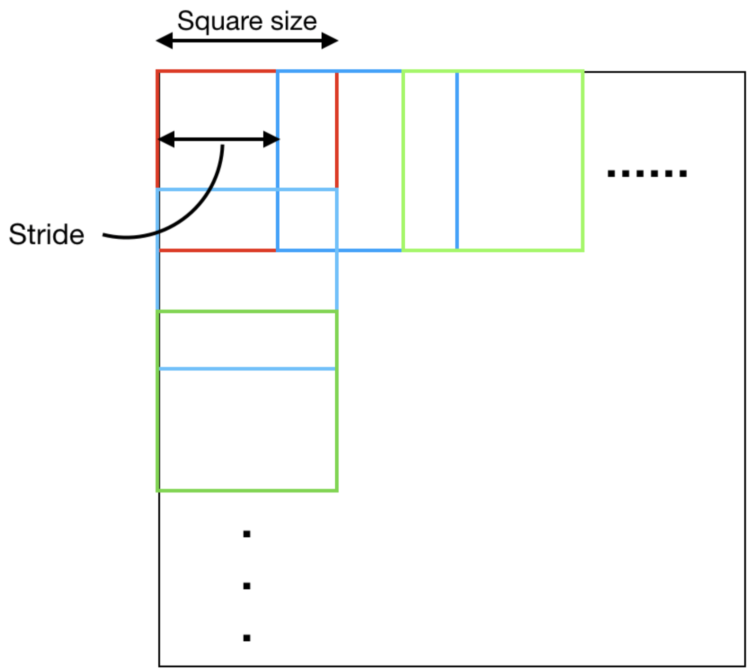


Figure 2.8: Segmentation of the original image

Since the shape of LCD panel is square, the method of cropping in the horizontal direction is same as in vertical direction. And the arrangement of cropped images is same as the arrangement of LCD panels on LCD board.

After finishing the step of segmentation of image, the next is to form a new image which looks like a whole LCD board. The gaps between different LCD panels can not display any pictures. So the same location on the new image should be black, and only the location of LCD panels on new image can have its RGB data. The picture below (Figure 2.16) is one new image that shows the same arrangement of LCD panels.

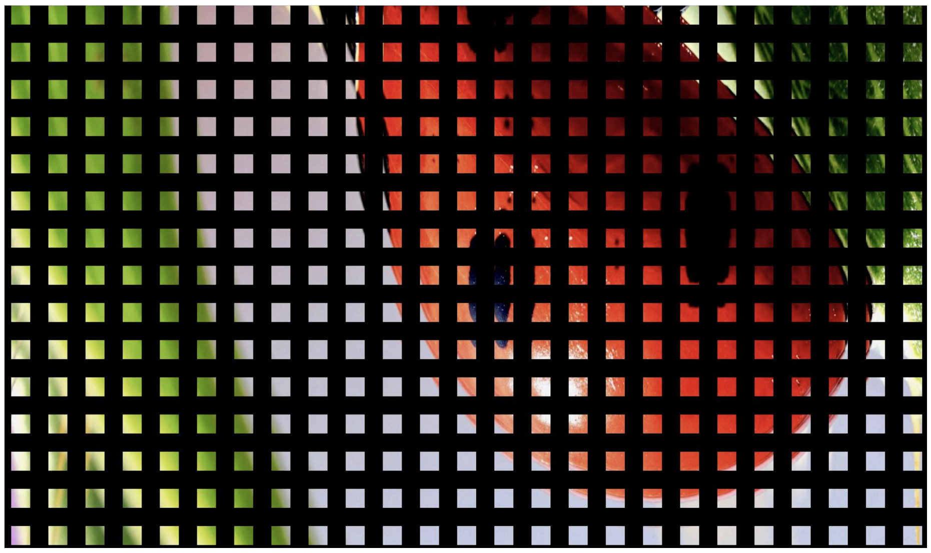


Figure 2.9: New image

2.3 Correction of brightness

2.3.1 Correct the brightness directly

Adjacent images on the new image has same RGB data on their overlapping part. And same images overlapped on one place will increase the brightness on this position. When displaying the new image on final screen, unique part will be darker than overlapping part. So it is important to correct the brightness on each LCD panel (Sajadi *et al.*, 2009). Before starting the correction of brightness, analyzing the situation of overlapping on each LCD panel is very important.

On the final screen, there are 8 other images around each image from each LCD panel. And for every image, there are 3 levels of brightness. From the picture below(Figure 2.10), the central part is the unique part which has the first level of brightness. As for 4 corners of the image from one LCD panel, it is overlapped by 4 times and has the 4th level of brightness. And the rest parts of the image are only overlapped by twice.

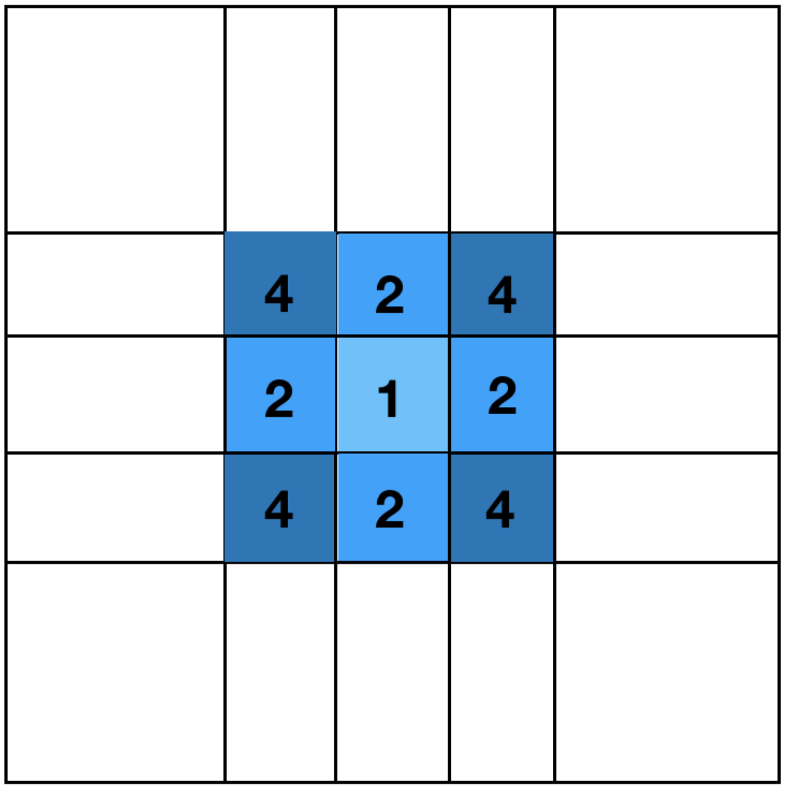
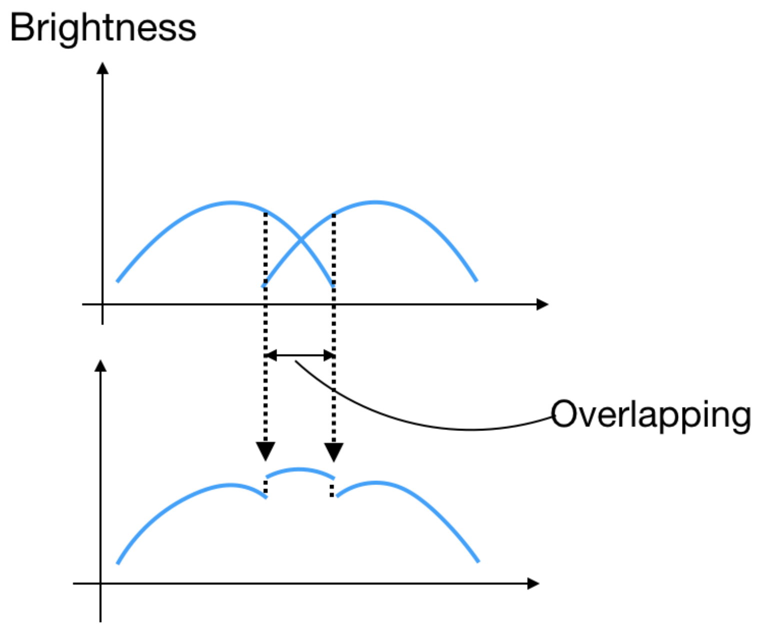


Figure 2.10: Different levels of brightness

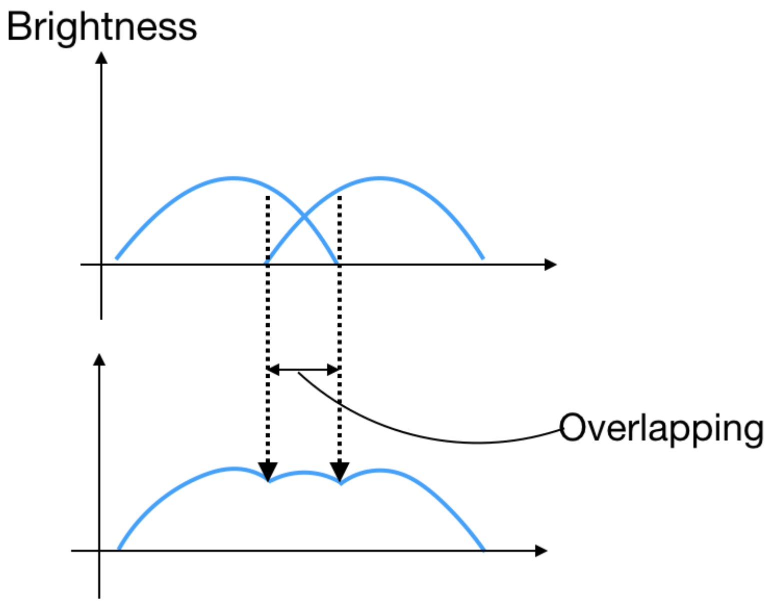
Theoretically, the number of overlaps is equal to the multiple of brightness. First try is to reduce the brightness of the corresponding multiple on this image before it displayed on final screen. However, there are some problems when trying to correct brightness directly. LCD panel is a kind of hardware, so it sometimes has bad contact in the actual situation. If we correct the brightness just by its multiple, there will be some distinct lines on the final display image. And this will cause bad experience for users.

From the picture(Figure 2.11a), we can find the reason of appearance of obvious lines. If each image has same brightness, there will be double brightness of overlapping part. And it makes the brightness of overlapping part much more different and obvious while comparing with the non-overlapping part.

But if we make the brightness of a picture drop smoothly from the middle, there will be some difference. It can be seen from the Figure 2.11b, the brightness of overlapping part is very close to the brightness of nearby none-overlapping part. This change can weaken the lack of hardware and the make lines not too distinct.



(a) Uniform brightness



(b) Brightness drops smoothly

Figure 2.11: Variation of brightness after overlapping

2.3.2 Simulation of the brightness

As it said before, It is better to choose the brightness of each image decrease smoothly from the central part. So the change of the brightness on each image is like a hill, the middle part is the brightest, and the marginal part is the darkest. We can choose cosine function to simulate this kind of mountain, and adjust a little to satisfy the hill we want.

Figure 3.10 below shows the simulation of variation of brightness on each image which is displayed by a small LCD panel. Figure 2.13 illustrates the brightness of some special point on the image. X,Y coordinates represent the location at an image, and Z means the brightness at corresponding location. It is also showed that the brightness of edge is very low, but not reduce to 0.

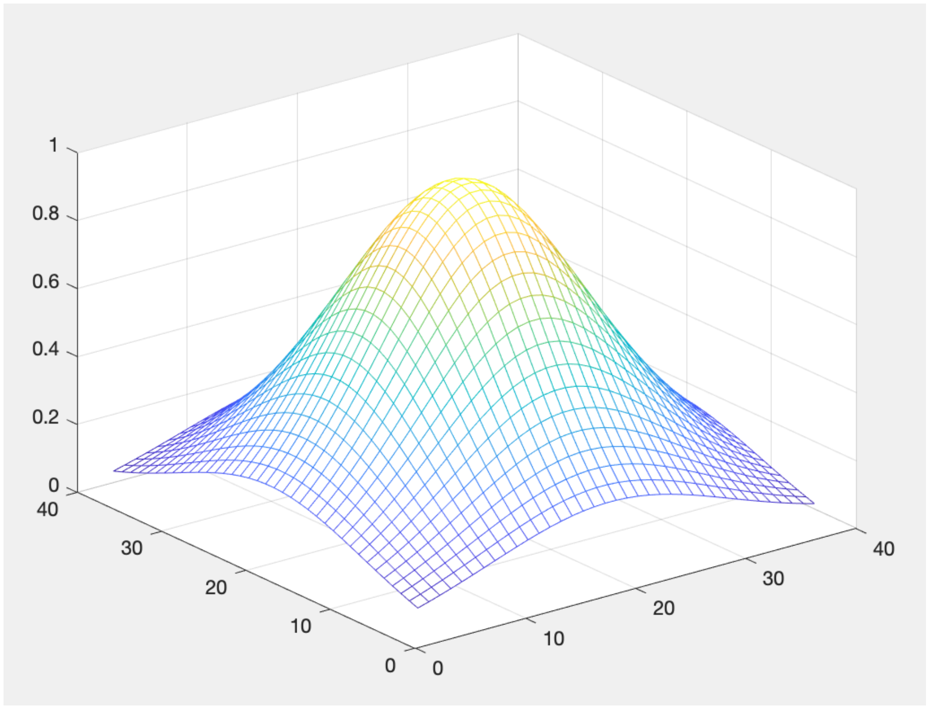


Figure 2.12: Simulation of brightness on one image

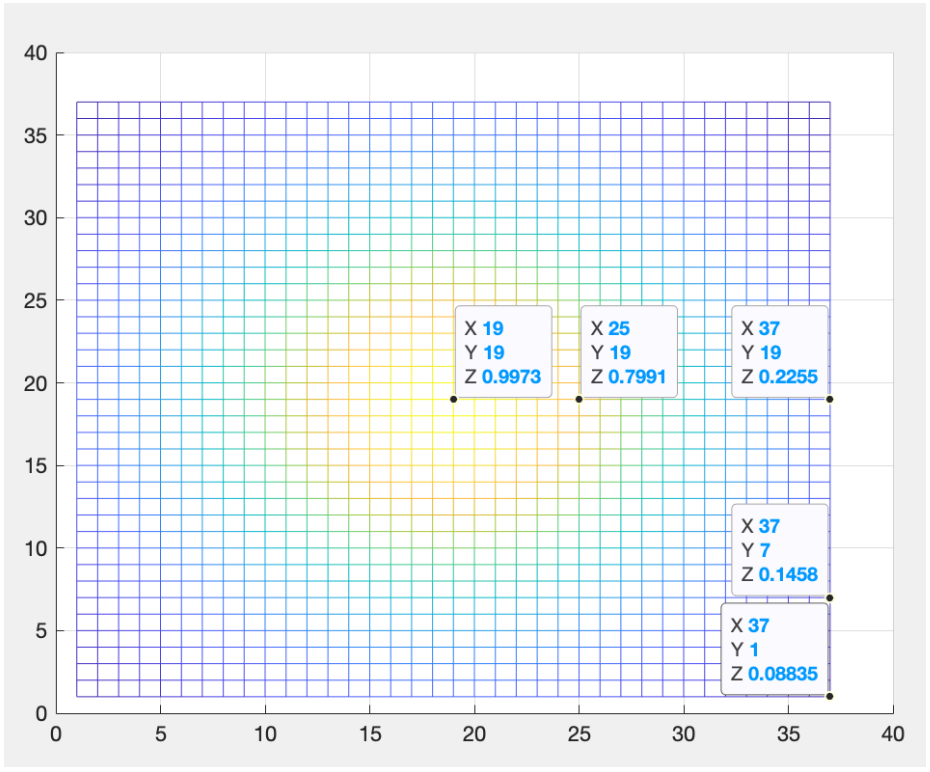


Figure 2.13: Brightness of each image

The way to simulate the brightness of one image as a hill need several steps in algorithm. First of all, getting the distance of every point and the central point of the image. And then, put the distance into the arctan function to get angle in equation 2.3.1. There is a parameter C which can be justified. When C is increased, the angle value decreased, which will cause the value of brightness becomes larger, which is shown in equation 2.3.2. In this way, we can make the slope of the hill becomes gentler since the brightness of edge becomes larger while the maximum of brightness unchanged.

$$angle = arctan(\frac{distance}{C}) \tag{2.3.1}$$

$$bright = cos(angle) \tag{2.3.2}$$

As it mentioned above, one image is overlapped by 8 adjacent images from nearby LCD panels. So the brightness of them also should be increased while overlapping. We need to choose the most representative situation to simulate the brightness of every images. So it is time to simulate the overlapping of 9 of same hills, and they have the same way as pictures overlap. However, there are some faults when we getting the simulation of 9 of hills, which is not we expect.

From the diagram(Figure 2.14) below, the brightness does not change smoothly and some parts even decrease vertically. This kind of condition can not solve the problem of distinct lines well. It is better than reducing the brightness directly, but not the best way. The reason of that is because the brightness of edge did not reduce to 0. It is showed in picture(Figure 2.13), even at the corner, the brightness is 0.08835 rather than 0. This is because that the value of arctan function can not be $\frac{\pi}{2}$. And the value of the brightness will not be 0 in cosine function.

The picture below(Figure 2.15)demonstrates the difference of variation of the brightness in overlapping part. In the Figure 2.15b, the edge of an image from an LCD panel is dark, which makes the brightness change continuously. Comparing with the picture(Figure 2.15a), the new one can better solve the appearance of faults in the figure(Figure 2.14). And there are several ways to justify the brightness of marginal part of each image. The function of brightness is a kind of continuous function. In order to maintain its continuity, we could multiply it by another continuous function and let the edge of image become dark. In my project, I multiple the original function of brightness by three different continuous function to get a better result. And it is necessary to make three continuous function be able to form a complete continuous function.

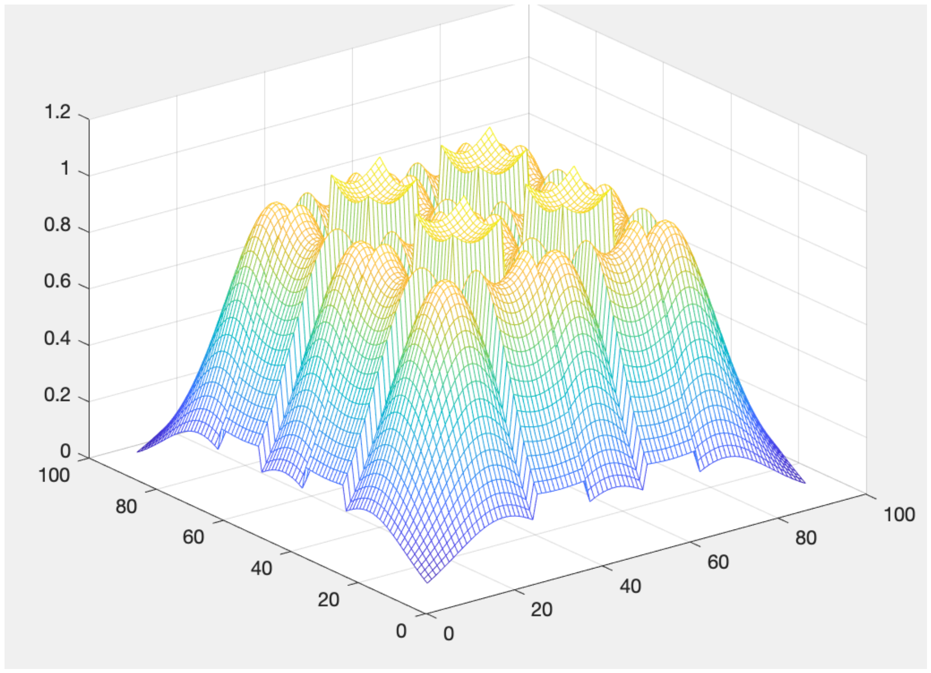
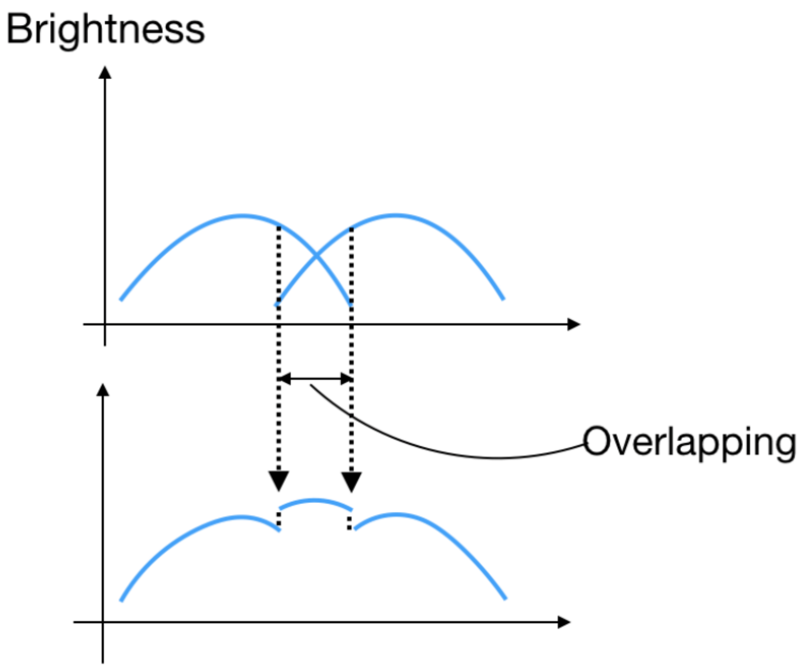
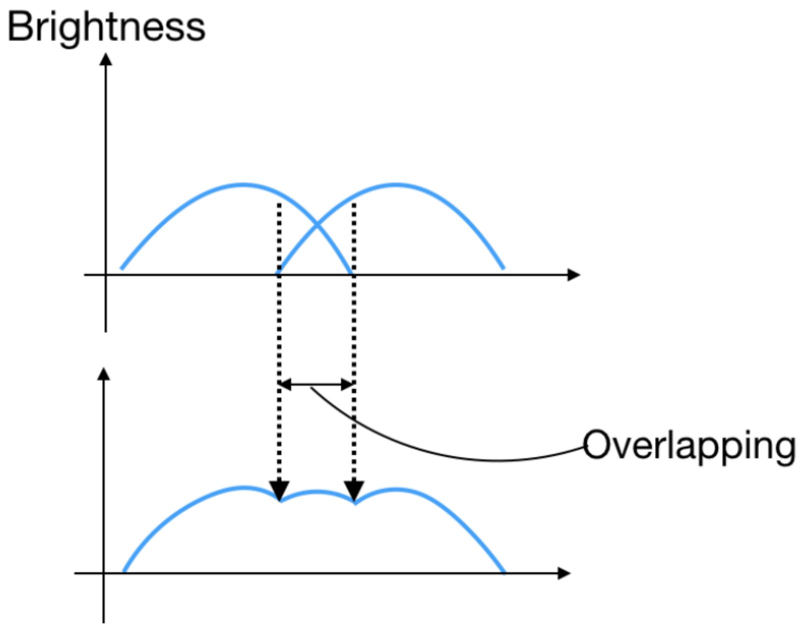


Figure 2.14: Brightness distribution of 9 images



(a) Uniform brightness



(b) Brightness drops smoothly to 0

Figure 2.15: Two kinds of variation of the brightness

With a slight modification of the hill of brightness, the distribution of the brightness of one picture is showed in Figure 2.16. Compared with the previous distribution, we can find the edge part of the image is almost dark. From the right part of the Figure 2.16, the brightness of the corner of image from LCD is very close to 0. And this kind of result is good enough to solve the problem of faults of distribution of brightness.

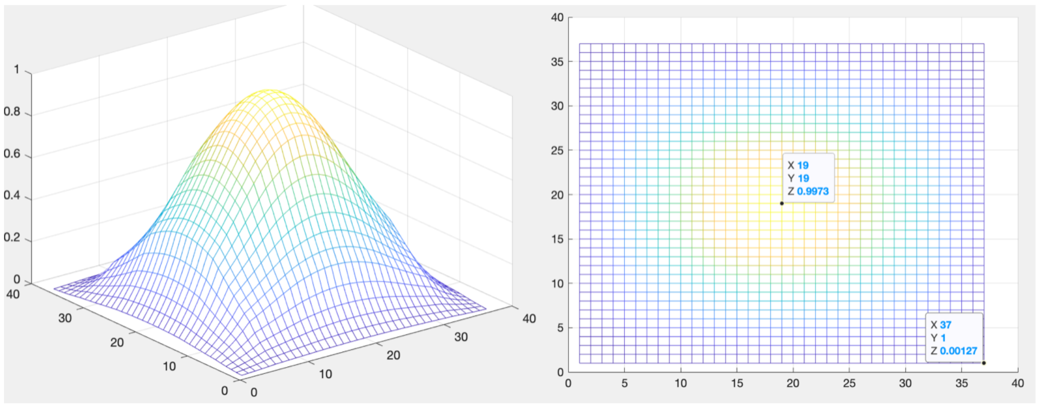


Figure 2.16: New distribution of brightness of one image

By using the new hill of brightness of one image, we add 9 of same hills together. And the overlapping way is same as that of the pictures. It can be seen that the result in Figure 2.17 is perceptually better than that in Figure 2.14. The adjusted distribution of brightness looks like rolling mountains and there are not any faults in it, which can help make up for the problems caused by hardware defects.

2.3.3 Normalization of the brightness

In order to correct the brightness of overlapping part, we need to cut out the middle piece of 9 of hills, which has the same size of just one hill. Figure 2.10 demonstrates that one image from one LCD panel is overlapped by adjacent 8 images. For every

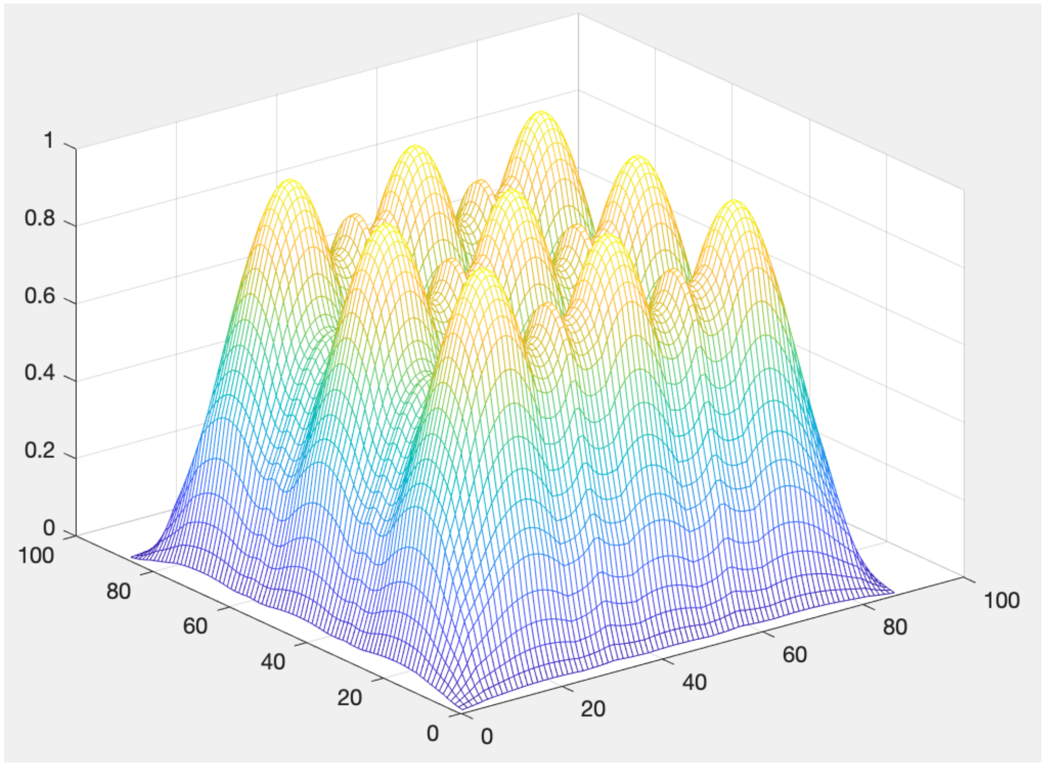


Figure 2.17: New distribution of brightness of 9 images

picture from each panel, they are all overlapped like that. So the center section is the most representative part which represent the distribution of the brightness of each image after overlapping.

Figure 2.18 below illustrates which part we select. The yellow color means the brightest parts, and we can see there are 9 yellow round spots since there are 9 images be overlapped. The square we select contains one yellow round spot, which matches what mentioned above.

Since there are different levels of brightness at the selecting part, we choose the minimum of them, which will be uniform brightness of the entire image. And then we need to set a scale that is very significant in normalization of brightness. Scale and lcd_map_mid both are matrixes. The index of them means the location at the image,

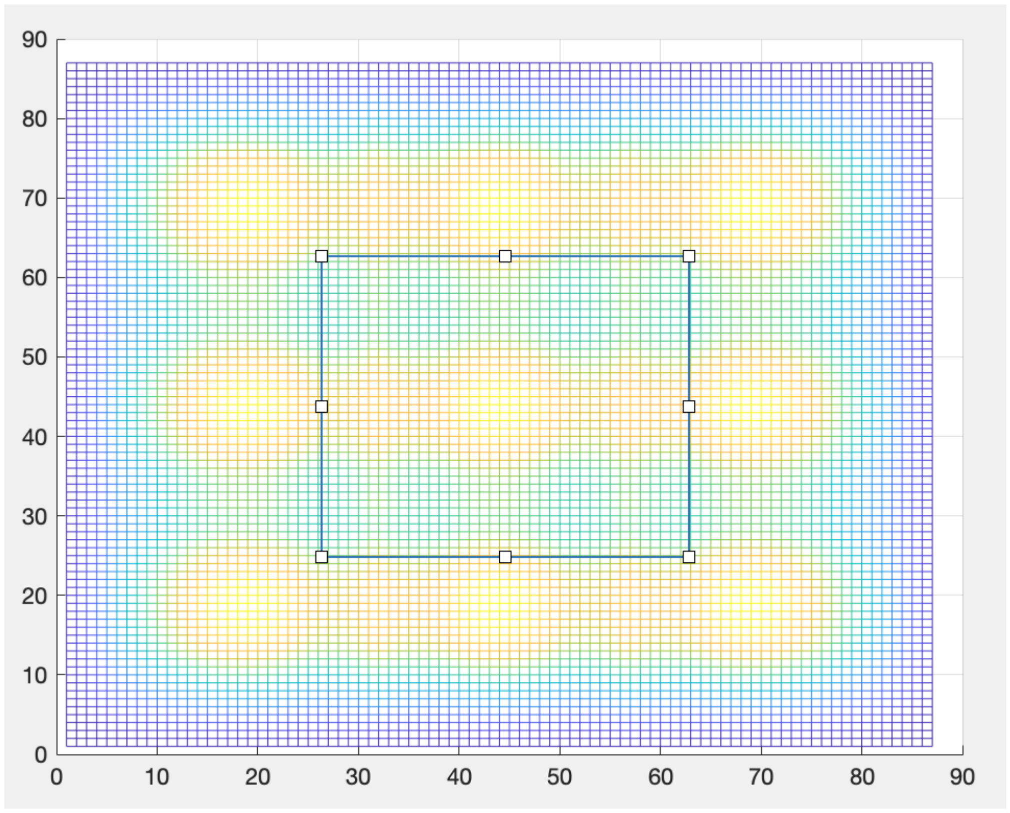


Figure 2.18: Selecting the central part of distribution of brightness of 9 images

and the value of them represent the value of brightness at that location. Therefore, every pixel of the image has its scale value which is the minimum value divided by the brightness of corresponding position ($Scale = \frac{min(lcd\_map\_mid)}{lcd\_map\_mid}$).

After obtaining the scale, multiple it with the brightness of one image which has not been overlapped. As a result, we correct the brightness of each image before it displayed by each LCD panel. And then, we need to re-arrange the processed 9 pictures in the same way as before. Figure 2.19 below shows the distribution of brightness of 9 images after normalization. It is obvious that mountains are flattened while comparing with the Figure 2.17.

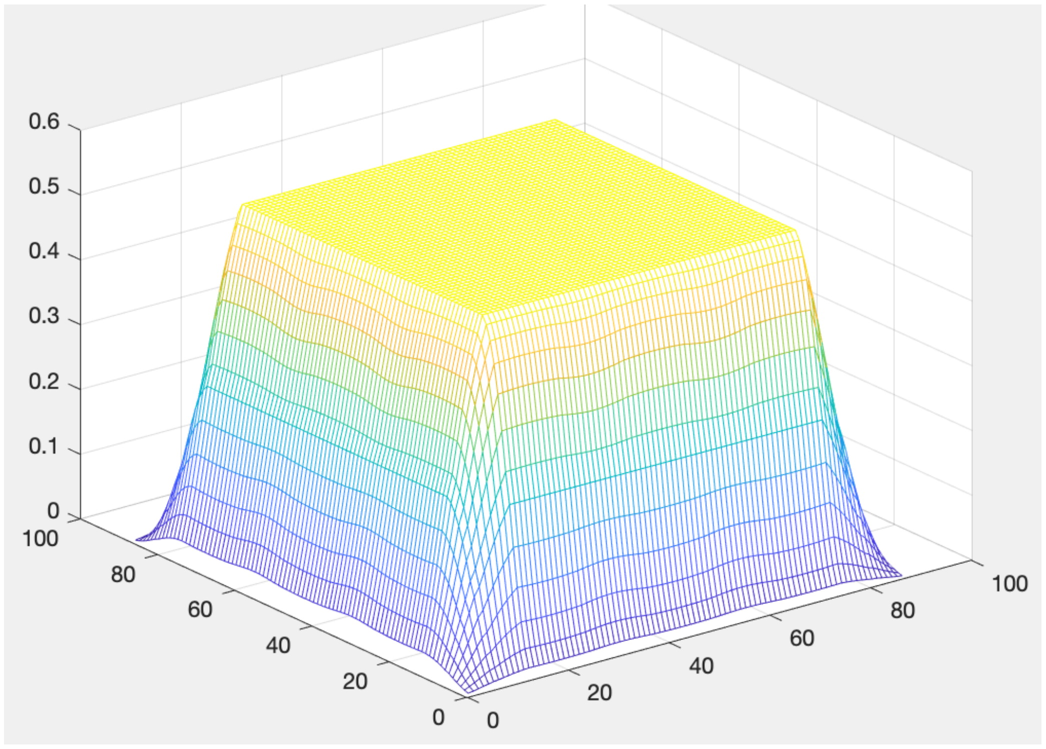


Figure 2.19: Distribution of brightness of 9 images after normalization

Besides, in the Figure 2.20, we can find the brightness of central part has the same value which is 0.5266. However, the marginal part of whole image is still darker than middle part, and the brightness decreases smoothly from the highest platform. This is because we only simulate the overlap of 9 images here. The marginal part has not overlapped enough times, so they can not get the same value of brightness of platform.

In fact, If we simulate overlap of more pictures, the size of platform will be larger. The application of this kind of method can make the entire screen achieve uniform brightness even if different parts overlap different times.

The principle of this method is a kind of linear normalization. For each image before overlapping, the value of scale is different at different position of it. And different images at the same position will increase the brightness. The value of brightness of those images are a, b, c...... in the equation 2.3.3 below. When we multiple the scale by the value of brightness of every image and overlap them as the arrangement before, the sum of brightness at one position will be the minimum of the brightness of the whole image. Their relationship is showed in equation 2.3.3 and 2.3.4.

$$Scale = \frac{min}{(a+b+c+\ldots.)} \tag{2.3.3}$$

$$scale * a + scale * b + scale * c + \ldots = min \tag{2.3.4}$$

As a result, the actual screen is able to display a better image with higher quality.

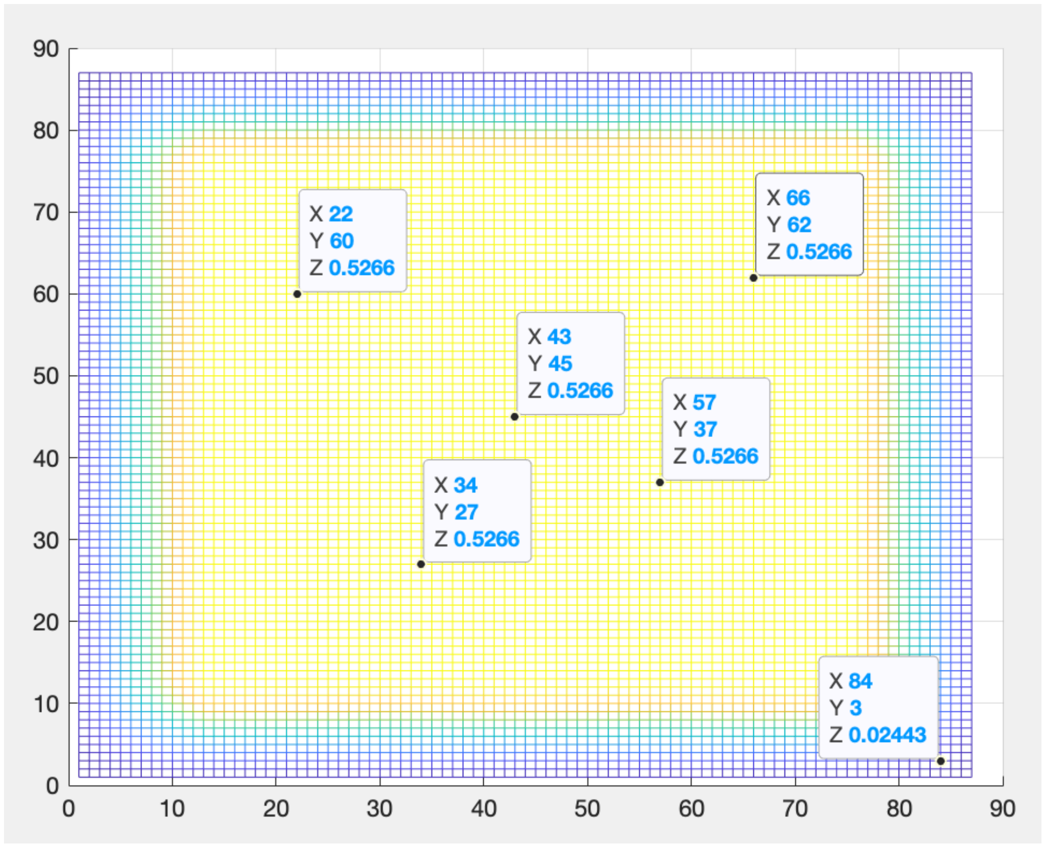


Figure 2.20: Top view of the normalization of brightness

2.4 Gamma correction

Gamma is very important in the digital imaging system. It represents the relationship of digital of RGB and the real illumination from LCD board (Lee and Chen, 2005). Gamma correction always have other names, like gamma encoding, gamma compressing, which have the same principle (Xiao *et al.*, 2011).

In fact, our eyes are more sensitive to the dark tones than bright tones. For example, when the number of RGB data increases twice, we will not perceive more than twice the light for the dark part. Figure 2.21 demonstrates the relationship of them, and the value of gamma there is 1/2.2. Since the sensitivity of human eye to light is a gamma function which is a kind of convex relationship.

And for the pixel In an image like JPG or JPEG, we only use 8 bits to store the RGB data which can have 256 kinds of color. However, because of the relationship between the actual value of brightness and it observed by human eye, we need to change the way of storing the value of brightness in computers. If we encode the 0.5 brightness into 0.5 pixels, this will mean we use 128 colors to represent the dark and bright areas. Actually, using so many kinds of bright areas is a waste of storage compared to dark areas. In human eyes, 0.5 brightness corresponds 0.18 value of it digitally. This means we should encode 0.18 brightness into 0.5 pixels. In this way, storage space can be fully utilized.

On the other hand, the gamma encoded gradient distributes the tones roughly evenly across the entire range (”perceptually uniform”). And this kind of encoding is also called gamma correction.

After talking about gamma encoding, there is a word called display gamma. In fact, in TV and graphics monitors, the electron beam generated by the picture tube and the brightness of the image generated by it does not change linearly with the input voltage of the picture tube.Compared with the input voltage, the electron flow changes according to an exponential curve. This shows that the signal in the dark area is darker than the actual situation, and the bright area is higher than the actual situation. The relationship of recorded pixel value and display luminance is also a gamma function, and the value of gamma here is greater than 1.

For lots of monitors today, the industry of them has converged on a standard display gamma of 2.2 which is just the reciprocal of that of encoding gamma and adjust the contrast setting. The overall display gamma is actually comprised of some native monitors like LCD gamma and some display monitors that have gamma correction applied inside. However, each effect of them always depends on the type of display device. For LCD monitors, an overall display gamma which is 2.2 always needs to correct substantially and the consistency of it is much less than that of CRT. Therefore, LCD requires something called look-up table (LUT) in oder to ensure the intended value of display gamma can describe input value.

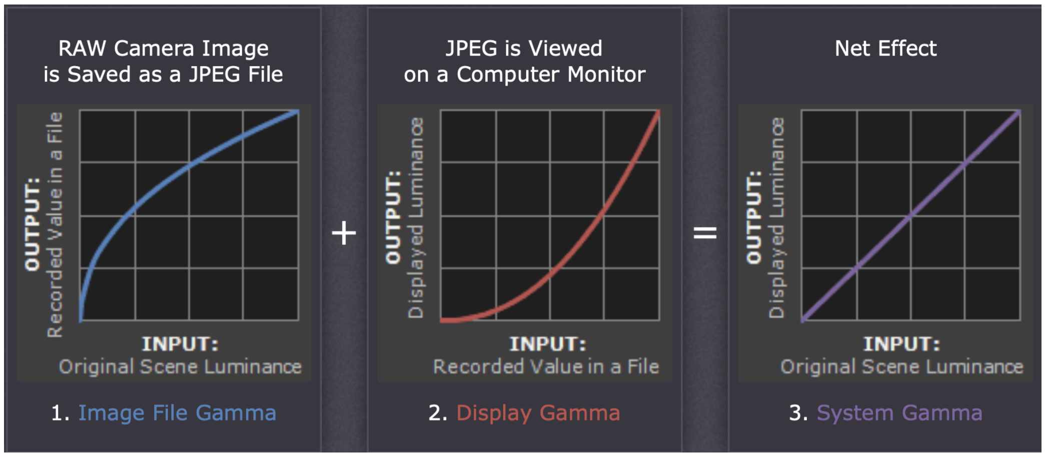


Figure 2.21: Combination of image gamma and display gamma

After the combination of encoding gamma and display gamma, we can find the result from Figure 2.21. The system gamma here represents the net effect of all gamma values including encoding gamma (Rahman *et al.*, 2016) and display gamma. These two kinds of gamma applies on the image, and ideally make the relationship of input and output becomes close to a straight line(gamma equals to 1.0).

As it is mentioned above, there are two kinds of overlapping for each image. We need to pay attention to the gamma curve for each contributor when correcting the brightness of overlapping parts.

$$V_{out} = C * V_{in}^{\gamma} \tag{2.4.1}$$

$$V_{out} = C * \left(BC^{\frac{1}{\gamma}} * V_{in}\right)^{\gamma} \tag{2.4.2}$$

$$V_{out} = C * BC * V_{in}^{\gamma} \tag{2.4.3}$$

Equation(2.4.1) shows the relationship of recorded value and displayed luminance, which corresponds to display gamma. V_{in} and V_{out} are the input and output matrix here, and constant C is 255 since V_{in} is normalized image which would be 0 to 1. Besides, BC in the equation 2.4.2 and equation 2.4.3 represent the scale which is mentioned above. We need to apply gamma correction to BC before pictures displayed by monitors. By this way, the brightness of images we see will multiple to BC instead of being influenced by display gamma of device.

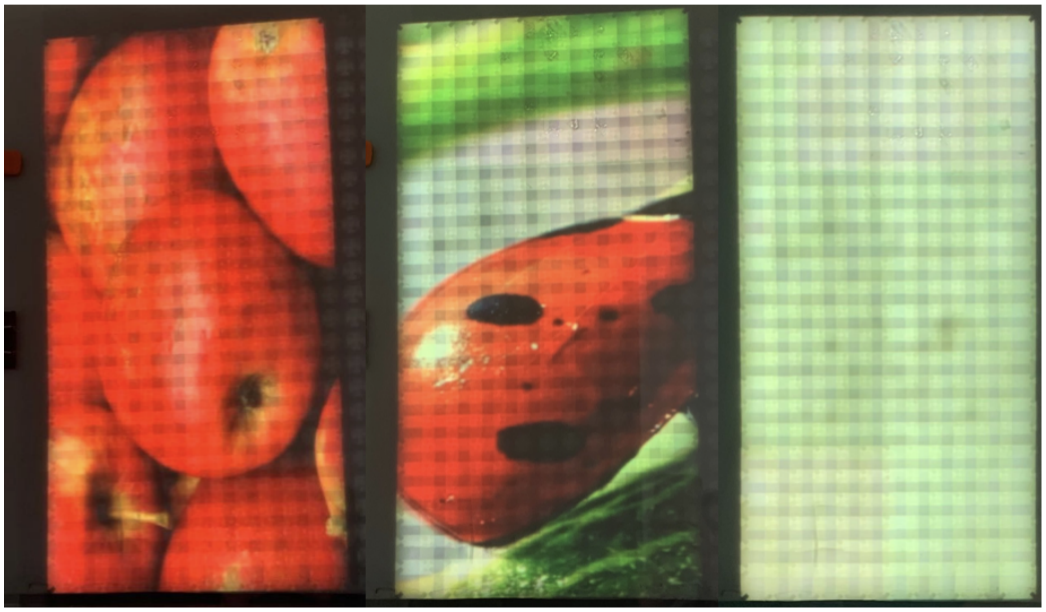


Figure 2.22: Images before normalization and gamma correction

It can be seen from Figure 2.22 and Figure 2.23 that the pictures displayed after brightness correction are better than pictures before brightness adjustment. The brightness of them becomes more uniform and the obvious lines which appear in the junction of images has been blurred. However, because of the defect of hardware, the vertical stripes can still be observed on the final screen even if images has been corrected on brightness.There is some small misalignment between the front LEDs

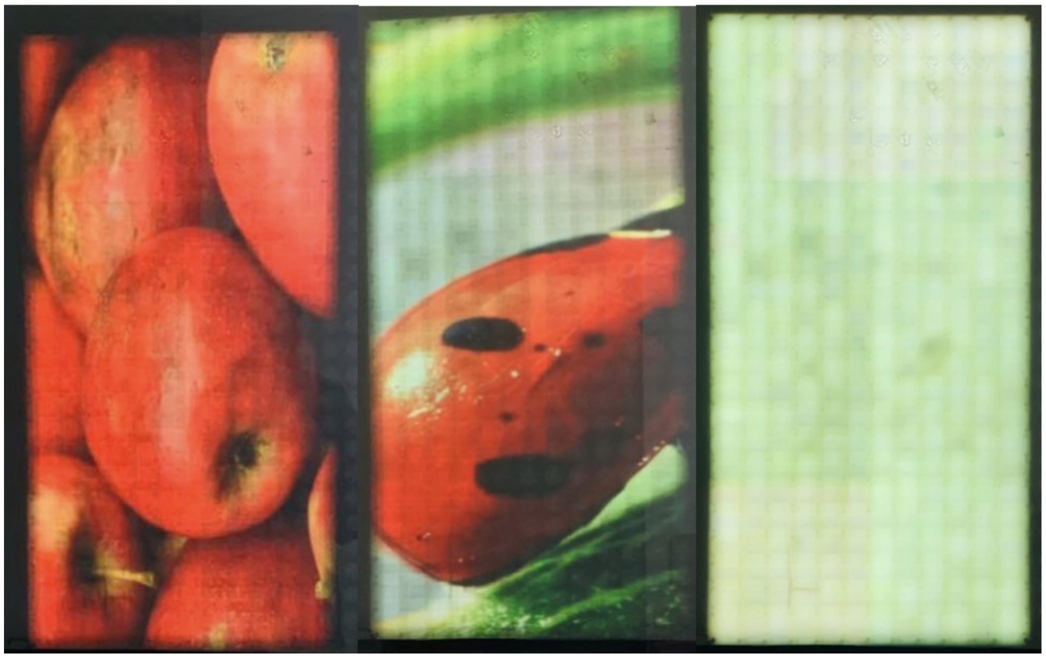


Figure 2.23: Images showed after normalization and gamma correction

and the back LEDs. Therefore, mechanical tolerances in the LED alignments between front and backlights are the cause of vertical stripes. However, this kind of defect can be eliminated when producing the final displayed screen.

Chapter 3

Combination of LCD and LED

Adding RGB LEDs is very necessary in this project, since there are some defects of LCD display screen. Even if we process the LCD images in advance, there still are some vertical dark lines and they are unable to achieve uniform brightness under ideal conditions. This is because that there still is some difference between theoretical simulation and real situation.

Not only is there an error in the measurement, but also the positions of back LEDs and front LEDs are misaligned. On the LCD images, it can be discovered that the detail part is not easy to see flaws. Instead, the blemishes in the pure color part are obvious, especially when the picture is all white. However, the characters of LED can just offset these hardware defects.And this is the main reason of combining LCDs with LEDs.

LED spot is bigger and the brightness of it fades smoother, which can blur the lines and uneven brightness on the LCD image. Besides, the range of LED is wider than LCD's so that the color showed by LED is closer to the real picture. So we add LEDs in front of LCD panels and each LED located near the 4 corners of each LCD panel. The main idea of combination is using LEDs to dim the noticeable lines in the pure color part of LCD image while keep details of it. In this way, we can reduce defects and maintain the high resolution of LCDs.

There are four steps of combining LCD and LED together. Firstly, we need to determine the position of LEDs in the LCD image. Secondly, the color of each LED should be determined since the diameter of the light spot of LED is almost 53 millimeters. Next, we are supposed to delight some of the LEDs to achieve the best effect rather than lighting all the LEDs. So we need to confirm which LEDs should be turned on. Finally, we need to make sure that the LED light can coincide with the LCD image very well.

3.1 Confirm the coordinates of LEDs

As said above, LEDs located on the 4 corners of each LCD panel. Even if the light spot of LED will be expand on the final glass screen, the position of the center of the light spot on the LCD panel is same as that on the glass screen.

From Figure 3.1 below, the black square means one LCD panel and there are 4 LEDs located at 4 corners of it. On this picture, we can find that the distance of every 2 LEDs are 10 mm which is half of the distance between the leftmost sides of the two LCD panels. As for the red dotted square in the figure, it represents every LCD image displayed by each LCD panel, and the size of this LCD image is 25 by 25 pixels on the final screen.

When two LCD images displayed by LCD panels just not overlap, the size of each image is 25 by 25 pixels. And the length of square with red dotted edge is also 20 millimeters, which is same as the distance of two leftmost sides of the two LCD panels.

This is because that the red square in this size just can divide the entire LCD board equally.

In this way, we will know the pixel value that the measured length represents on the final screen. In other words, the pixel distance of LEDs on the final image can be determined, which is shown in equation 3.1.1.

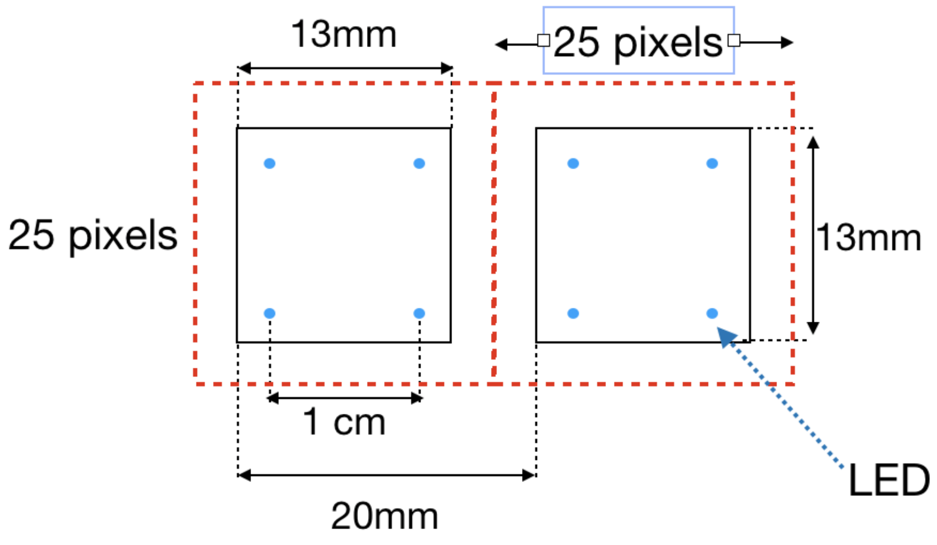


Figure 3.1: Position of LEDs on the LCD board and LCD images

$$\frac{Distance\ in\ pixel}{measured\ distance} = \frac{25\ pixels}{20mm} \tag{3.1.1}$$

However, there is one thing we need to pay attention to. In actual operation, each LCD panel will not only display 25 by 25 pixels image since it can not make full use of each panel and will also cause obvious black lines.

If we need to determine the coordinate of every LED in pixel. We are supposed to know the pixel distance between the first LED and the very edge of the picture on final screen, since we set the pixel value of very edge of the picture to 0.

From the equation of distance in pixel and measured distance, we could calculate that each LED and the nearest red square is 12.5/2 pixels apart. And if the LCD panel displays a bigger image such as 37 by 37 pixels, this image will expand on the final screen and has a larger size than that of red square.

Since the center of the bigger square image on final screen is same as the center of the red square, the leftmost distance of the two squares are (37-25)/2 which is 6 pixels. After calculating the distance of every two LEDs and the distance between the first LED and the very edge of the final image showed on glass screen, the coordinates of every LED can be determined eventually.

Algorithm 1 Determination of the coordinates of LDEs

```
n=2;
m=1;
for j = 1:n*24                    % n*24 LED lights per row
    for i = 1:m*24                % m*24 LED lights per column
        x2 = round((6 + 6.25 + (j-1)*12.5));
        y2 = round((6 + 6.25 + (i-1)*12.5));% coordinate of
            each  LED
```

3.2 Determination of the color of each LED

3.2.1 Mean of the pixel value inside the range

In fact, there will be a complete LCD image on the display screen finally. The image consists of plenty of pixels and we assume each pixel in the image is in row i and column j. The point in the upper left corner of the picture is set to be the starting point where i and j are 0. As for the LEDs, the light spot of each of them influences a circular range of LCD image rather than only a point, the pixel value of each LED should be determined by its surrounding LCD pixel values.

That means we are supposed to get all the pixel values in the range by searching their index of row and column.

Firstly, we need to measure the diameter of LED spot on the final screen and multiply it by the resolution of the glass screen to get the pixel length of the spot. As mentioned above, the coordinate of every LED on the LCD image can be determined by the method told before. Combining the two, the value of each pixel inside the spot range can be confirmed, which is the crucial information to determine the color of LEDs.

However, the circular range is not efficient enough in terms of the algorithm because the discrete pixels in the image are arranged like a matrix and the calculation speed of the mean of square is faster than that of circle. That's why, in my algorithm, I used a square instead of a circle to represent the spot range.

In the Figure 3.2, the red point means one LED light and there is a blue square which is the circumscribed square of the blue circle. This kind of blue square is called color box and the the blue circle is called color circle here.

After that, we need to calculate the average value of RGB in the range. Since there are three channels for RGB per pixel, we need to calculate the mean of pixel value separately. In other words, one picture is like a three-dimensional matrix and the third dimension contains RGB data respectively. For the red channel, the pixel values in the color box are arranged like a two-dimensional matrix and the value

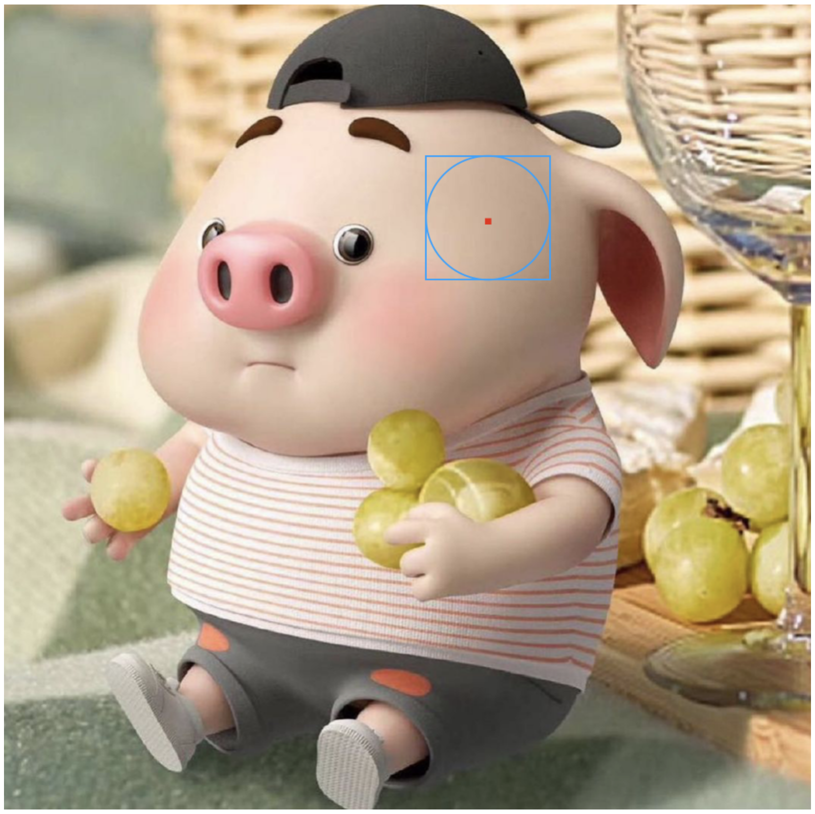


Figure 3.2: The range of LED on the LCD screen

for each pixel located in row i and column j is called r_{ij}. Therefore, we just need to calculate the mean of this two-dimensional matrix, which is calledR_{mean} in the equation 3.2.1. And the same goes for the blue and green channels, which is shown in equation 3.2.3 and equation 3.2.2.

$$R_{mean} = \frac{1}{m * n} \sum_{i=1}^{n} \sum_{j=1}^{m} r_{ij} \tag{3.2.1}$$

$$G_{mean} = \frac{1}{m * n} \sum_{i=1}^{n} \sum_{j=1}^{m} g_{ij} \tag{3.2.2}$$

$$B_{mean} = \frac{1}{m * n} \sum_{i=1}^{n} \sum_{j=1}^{m} b_{ij} \tag{3.2.3}$$

3.2.2 Simulation of display gamma

After confirming the mean of pixel value inside the color box, we proceed to consider the display gamma of LEDs and LCDs.

In fact, there is no gamma distortion for RGB LEDs, which means that it is a linear relationship between the brightness change in the data and the brightness change by our human eyes. Nevertheless, the color value of LEDs comes from the LCD image, and there is a gamma distortion for the LCD image. In order to combine LCD and LED appropriately, it is necessary to simulate the display gamma for every RGB LED.

$$A_{LED} = 255 * \left(\frac{A_{LCD}}{255}\right) .\hat{}\ (2.2) \tag{3.2.4}$$

In equation 3.2.4 above, A_{LCD} means the three-dimensional matrix of the LCD image and A_{LED} represents another three-dimensional matrix of LEDs after gamma simulation. It should be noted that normalization is required before gamma simulation since the range of the gamma curve is 0 to 1.

3.2.3 Color matching of LEDs and LCD

When we got the average value and simulated the gamma distortion, the LED light did not show the desired effect. Figure 3.3 below illustrates the big difference of original images and the displayed LED images while the LCD pictures are on the left half and the LED pictures are on the other half. Since the resolution of the LED is relatively low, it is understandable that the resulting pictures are blurry.

However, compared to the original pictures, the LED pictures look pinkish overall, which means that the color difference between LCD and LED lights affects the overall color of the picture.

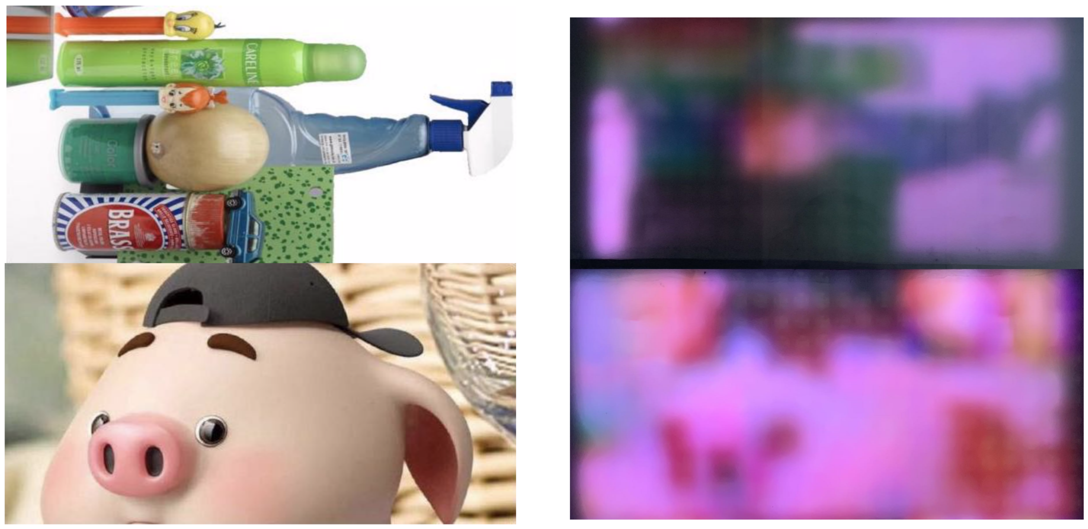


Figure 3.3: Original LCD images and displayed LED images

There are several reasons for this phenomenon, one is the difference of the color range between LCD and LED and another is the disparity of chromaticity coordinate systems of them.

Figure 3.4 demonstrates the color gamut possible with CRT, LCD, LED, and laser-based displays.The available color for several display types is indicated as a set of triangles. Each triangle has 3 vertices that represent R,G,B points (Chellappan *et al.*, 2010). Notice that the vertices of LED triangle are more saturated than the vertices of the LCD triangle. And this can explain the difficulty in color matching.

On other words, for LEDs and LCDs, each color can be decomposed into three primary colors of RGB in different proportions. In Figure 3.5, the color light C to be measured can be decomposed into the three primary colors R, G, and B in the

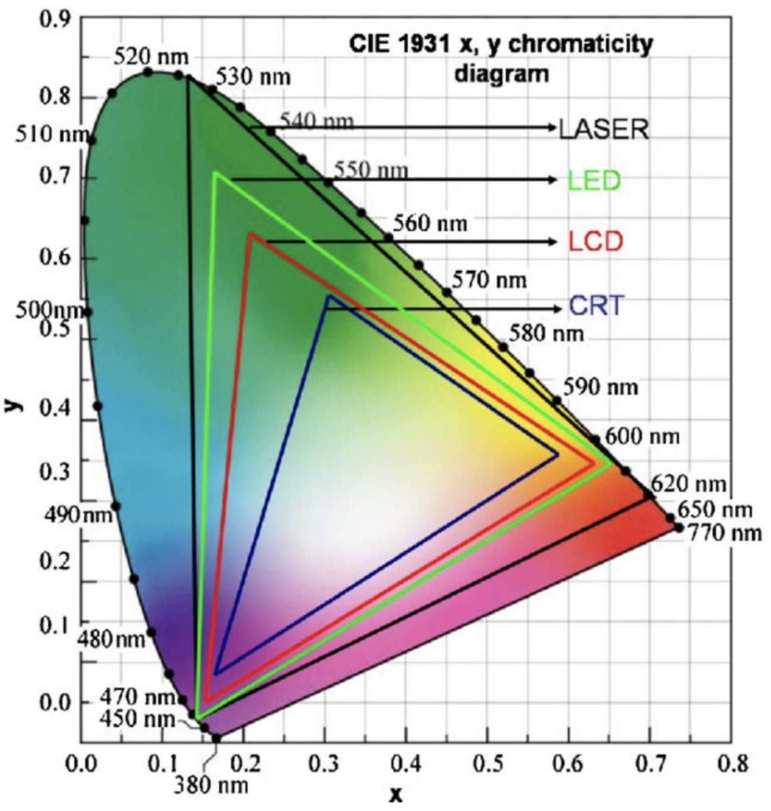


Figure 3.4: 1931 CIE xyY chromaticity diagram

chromaticity coordinate system of the LCD, which are represented by black lines in the picture. And the blue lines mean the chromaticity coordinate system of the LED while its three coordinates of R,G and B are shown as R2, G2 and B2 in this figure.

For the same monochromatic light C, it has different expressions in the two coordinate systems which is shown in equation 3.2.5 and equation 3.2.6. If we assume that the RGB primary colors in the LCD coordinate system are the correct colors, then we are supposed to find the correspondence between the three primary colors of the LED and the three primary colors of the LCD.

$$C = \overline{r}(R) + \overline{g}(G) + \overline{b}(B) \tag{3.2.5}$$

$$C = \overline{r2}(R2) + \overline{g2}(G2) + \overline{b2}(B2) \tag{3.2.6}$$

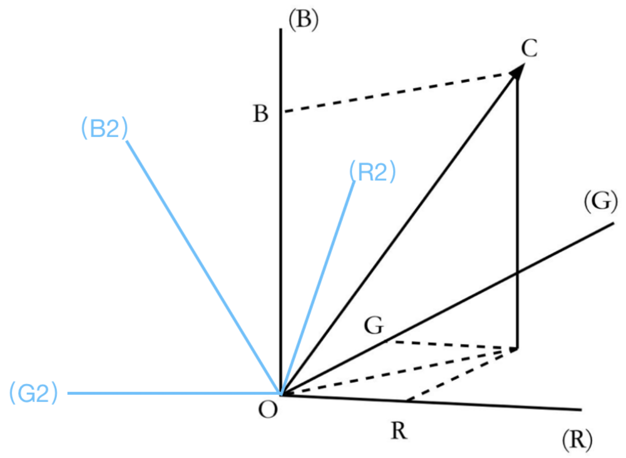


Figure 3.5: Chromaticity coordinate systems of the LED and the LCD

Because of the limitations of the experimental equipment, here we can only compare and adjust with the naked eye. Firstly, We display an all-red picture on the LCD while the RGB data of it is (255,0,0). Afterwards, we input the same RGB data to the LED light. Certainly, there is a big difference between the two kinds of red color. So we need to add some green or blue to make the red color looks same as that on the LCD screen.

And when there is almost no difference between the two colors, we record the new RGB data at this time. These new R,G and B colors are a1,a2, and a3 respectively. In the same way, we can also get the blue and green adjustment parameters which are able to convert two coordinate systems.

The basic principle of this method is to decompose first and reassemble. For each kind of color, it is mixed by a specific ratio of RGB of the LCD, which are expressed by Rmean, Gmean and Bmean in the equations below. In order to make the LED show the same color, we need to mix same ration of Rmean, Gmean and Bmean by the LED.

By the equation 3.2.7, 3.2.8 and 3.2.9 below, we should replace Rmean, Gmean and Bmean by R2, G2, and B2, since they are the three primary color of the LEDs.

$$R2 = a_1 * R_{mean} + b_1 * G_{mean} + c_1 * B_{mean} \tag{3.2.7}$$

$$G2 = a_2 * R_{mean} + b_2 * G_{mean} + c_2 * B_{mean} \tag{3.2.8}$$

$$B2 = a_3 * R_{mean} + b_3 * G_{mean} + c_3 * B_{mean} \tag{3.2.9}$$

After the determination of those 9 conversion parameters, we input the RGB value of LCD and convert them into the RGB value of LED to delight the RGB LEDs. Figure 3.6 below shows the LED images after adjustment. It can be observed that their color is closer to the original images than that of the LED images which inputs the mean of pixel value directly.

Besides, the phenomenon of the overall reddish color has also weakened significantly.

In the process of inputting RGB data to the LED lights, there is one more point that needs attention. In order to reduce costs, this project chooses to use TLC59711 to control the LED lights. In fact, the TLC board supposed gray resolution of 16-bits

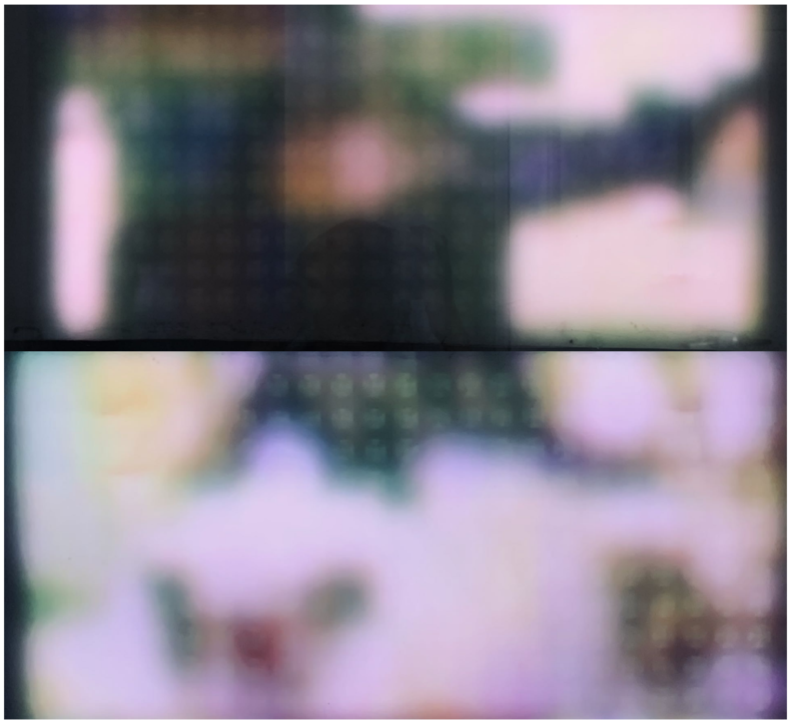


Figure 3.6: LED images after adjustment

while the LCD images we used has a radiometric resolution of 8-bits. This means 65535 is full brightness for LED light rather than 255. So what we need to do is just scale the image value by a scaling factor.

At the beginning, we choose to set the scaling factor to 256, since it means simply shifting the 8-bits value to the left by 8-bits. However, doing so will have some unsatisfactory conditions. For example, the RGB data of (0,3,0) makes the corresponding point quite dark in the LCD image.

And if there are such similar RGB data in a range, then this range is all black to the naked eye. However, if we multiply the RGB data by 256 and input it to the TLC to delight LED, the RGB data will become (0,768,0) which makes the LED light green. This means that this kind of scaling factor is too big and we are not supposed to shift the 8-bits value by 8-bits simply.

In order to find a much smaller scaling factor, there are several steps. We need to put the LCD brightness to 255 white firstly, and then try to match this brightness with the RGB LEDs. Once the brightness of LED and LCD are matched we will have a usable scaling factor.

3.3 Determine which LED should be turned on

As mentioned above, if an image has many details on it, the lines on the picture due to overlap will be almost invisible. For instance, there is a LCD picture of church windows below, which is the leftmost one in Figure 3.7. Even if there is no brightness correction on the overlapping part, it is not easy to see any flaw in this detailed picture. Its reason can be explained by the two pictures on the right. In Figure 3.7,

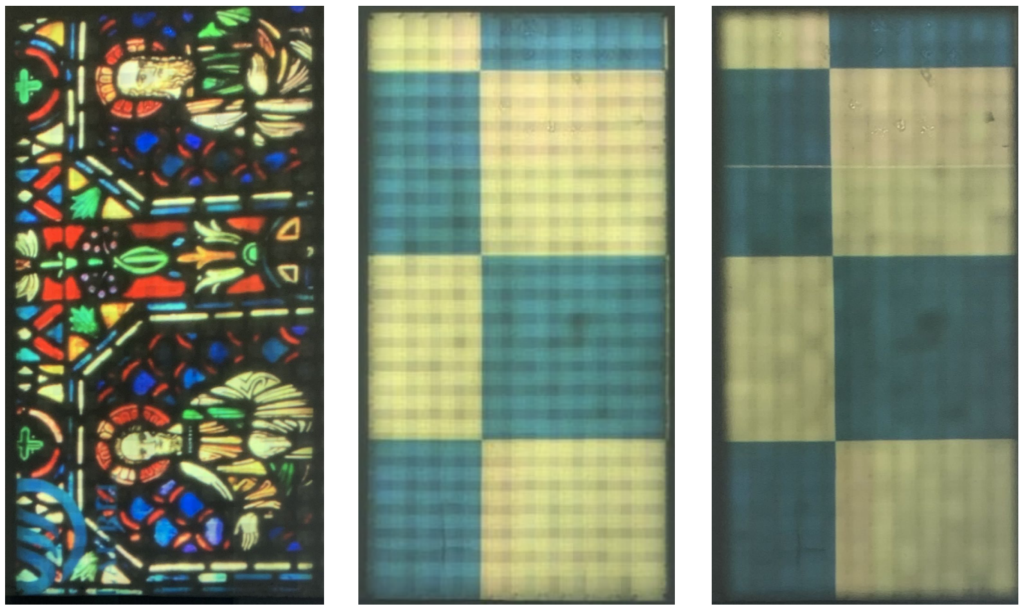


Figure 3.7: LCD images without LED

the middle picture is a blue checker board before brightness adjustment. We can find that there are quite obvious lines on it. And the white lines are more conspicuous than the blue lines. It is because the white part is brighter than the blue part. For example, if the RGB value of one image is (1,1,1), then the maximum brightness of the overlapping part is only (4,4,4) which is not much different from non-overlapping. However, if the pixel value of another image is (50,50,50), then the overlapping part will become (200,200,200) which makes the difference between light and dark suddenly becomes very big. Once we apply brightness correction on the overlapping part, the lines on the brighter part will be more obvious than that on the darker part, which has been shown on the right of Figure 3.7.

And that is why the detailed part will rarely show flaws such as the horizontal and vertical black lines. In one detailed part, it has many small areas with different colors and different brightness. And this make each black line has different part with different brightness, which can not be seen as a line in the naked eye.

However, an ordinary picture has not only detailed parts, but also evenly colored parts. Therefore, we will see noticeable lines on the detailed parts instead of the parts with uniform color. And this phenomenon determines which LED lights need to be fully turned on. We are supposed to add LED light on some evenly colored parts in order to blur blemishes and attenuate light or even turn off the LED in the uniform color in order to preserve the high resolution of the LCD picture.

And here we use variance to judge whether the color of an area is uniform or monochromatic. Inside the color box, we calculate the variance of the three color channels separately, and then, add them together to get a new variance.

$$R_{var} = \frac{1}{m*n}\sum_{i=1}^{m}\sum_{j=1}^{n}(r_{ij} - R_{mean})^2 \tag{3.3.1}$$

$$G_{var} = \frac{1}{m*n}\sum_{i=1}^{m}\sum_{j=1}^{n}(g_{ij} - G_{mean})^2 \tag{3.3.2}$$

$$B_{var} = \frac{1}{m*n}\sum_{i=1}^{m}\sum_{j=1}^{n}(b_{ij} - B_{mean})^2 \tag{3.3.3}$$

$$Var_{new} = R_{var} + G_{var} + B_{var} \tag{3.3.4}$$

In the equation 3.3.4 above, R_{var}, G_{var} and B_{var} respectively represent the variance of the three channels of RGB. And the average value of the pixels inside the color box is expressed by R_{mean}, G_{mean} and B_{mean} in equation 3.3.1, equation 3.3.2 and equation 3.3.3.

For each point located at row i and column j inside the color box, it should minus the mean of whole range firstly. Then, the mean of the sum of the squares will get the variance of each channel. After getting the new variance which is the sum of three of channel variance, it is time to set a threshold to determine whether to fully turn on the LED. If the value of the new variance is less than the threshold, then we input the mean of color box to the LED to turn on it. Note that the value of mean here is the adjusted average which has been matched by LCD and LED color.

On the other hand, when the value of the new variance is greater than the threshold, we choose to reduce the brightness of this LED rather than turning it off directly.

There are several choice to delight the LED with low brightness when it should not be fully lit. In this project, we input the minimum value in a color box and divide its brightness by 2. For this step, we could choose to not use the previous color box since we used the value of a specific point instead of the overall value. Hence, we could appropriately narrow the color box. And the specific reduction scale needs to be judged and adjusted according to the results of the experiment.

There are several advantages to light up the LED lights through the above methods. The biggest one is to blur lines that shouldn't appear and reduce blemishes.And by adding the LEDs on the LCD, the overall brightness of the picture has increased. Besides, because we choose to reduce the brightness of LED which should not be turned on theoretically, the brightness and color transitions in the final picture will be smoother.

Figure 3.8 shows an example of selectively turning on the LEDs by the aforementioned method.

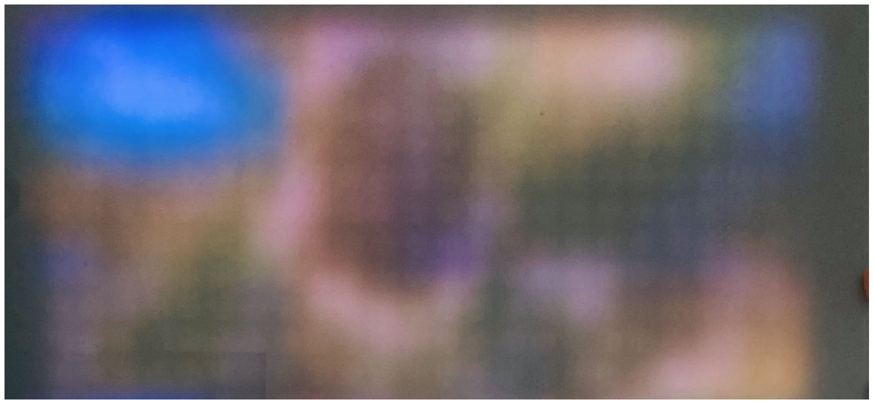


Figure 3.8: the Image of selectively turning on the LEDs

Algorithm 2 Generation of the LDE image

```
for j = 1:n*24
```

```
    for i = 1:m*24
        x2 = round((6 + 6.25 + (j-1)*12.5));
        y2 = round((6 + 6.25 + (i-1)*12.5));
        temp = image(y2-radius2:y2+radius2,x2-radius2:x2+
           radius2,:);
        R = temp(:,:,1);
        G = temp(:,:,2);
        B = temp(:,:,3);
        R_mean = round(mean(R(:)));
        G_mean = round(mean(G(:)));
        B_mean = round(mean(B(:)));
        deviation = std2(G) + std2(R) + std2(B);
        if deviation < threshold %determine which LED should
           be turned on
            R2 = round((R_mean*80 + G_mean*60 + B_mean*2)
               /255);
            G2 = round((R_mean*8 + G_mean*195 + B_mean*0)
               /255);
            B2 = round((R_mean*0 + G_mean*0 + B_mean*85)/255)
               ; %color matching of LED and LCD
            led_image(i, j, 1) = R_mean;
            led_image(i, j, 2) = G_mean;
            led_image(i, j, 3) = B_mean; % determine the
               color of each LED
        else
            R_min = double(temp4(:,:,1));
            G_min = double(temp4(:,:,2));
           B_min = double(temp4(:,:,3));
           R_min = (R_min*80 + G_min*60 + B_min*2)/255;
           G_min = (R_min*8 + G_min*195 + B_min*0)/255;
           B_min = (R_min*0 + G_min*0 + B_min*85)/255; %color
               matching of LED and LCD
           R_min = round(min(R_min(:))/denominator);
           G_min = round(min(G_min(:))/denominator);
           B_min = round(min(B_min(:))/denominator);
           led_image(i-2, j-2, 1) = R_min;
           led_image(i-2, j-2, 2) = G_min;
           led_image(i-2, j-2, 3) = B_min;
        end
    end
```

3.4 Adjusting LCD picture according to the LEDs

After selectively turning on several LEDs for one image, we could start to combine LCD image and LEDs together. However, there will be some undesirable phenomena if we combine them simply. The brightness of the entire picture will look uneven, and some areas will be significantly brighter, making it look a little abrupt. This is because that turning on the LED lights fully in some areas, coupled with the brightness of the LCD picture itself, will make this area much brighter than other areas.

And that is why we need to adjust LCD images according to the LEDs. We are supposed to darken the brightness of the LCD picture at the position where the LDE light is fully turned on.

Firstly, assuming that only one LED light is on, we need to simulate its brightness change on the LCD picture. The light spot illuminated by the LED light is a smoothly darkened spot from the center to the edge. Its brightness changes like a small hill. In order to simulate this kind of hill, we need to draw a square range firstly. Its center is the coordinates of the LED light, and the length is twice the radius of the LED light spot. In fact, the length here can be set appropriately larger than the diameter according to the actual situation.

Next, depending on the coordinates of each pixel in the square range, calculate the distance between each point and the center. Since we aim to simulate the appearance of the light spot, we need to filter out the pixels inside the square outside the spot, which is the white part in the box in Figure 3.9. And the way to achieve this filtering is to divide the distance by the radius to achieve normalization. If the result is greater than 1, then we directly set the pixel value of this point to 0, because the distance from the point in the circle to the center of the circle will not be greater than the radius. In terms of the remaining point, we will use a cosine function to express the normalized distance of them.

$$r_{norm} = \frac{r}{radius} \tag{3.4.1}$$

$$I = a * cos^b(r_{norm} * \frac{\pi}{2}) \tag{3.4.2}$$

Equation 3.4.1 and equation 3.4.2 above show the way of normalization and how

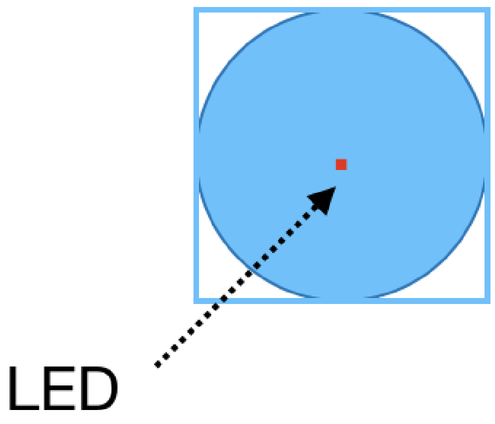


Figure 3.9: Assumed LED spot range and square range

to use cosine function to mimic the brightness distribution of single front led. In equation 3.4.2, we set the normalized distance r_{mean} as the input of the cosine function, and then, power this function and multiply it by a constant to simulate the brightness of the LED at that point, which is represented by I in the equation.

As for a and b in the equation, they are parameters to adjust the brightness, which I will explain in detail below.

Figure 3.10 and Figure 3.11 show the result of the simulation of the brightness of one LED spot. And the parameters a and b are used to respectively adjust the height and slope of the hillside in Figure 3.10. Increasing the values of these two will make the slope higher and steeper.

Algorithm 3 Simulation of the brightness of one LED

```
radius = 36;%diameter of the subtracted circle.
```

```
[x,y]=meshgrid(-radius:radius,-radius:radius);
led_mask=(x.^2+y.^2).^(1/2);   %calculate the distance to
   the center.
led_mask = led_mask/(radius+0.5);
led_mask(led_mask>1)=1; %the points whos distance to the
   center are more than 1 are set to 1.
led_mask = cos(led_mask*pi/2);%led_mask could be used to
   control the size and the brightness of the subtracted
   circle.
led_mask =0.15*led_mask.^4;%led_msak mimic the brightness
   distribution of single front led.
surf(x,y,led_mask);
```

As said above, we should compare the variance and the threshold firstly. And if we decide to turn on the LED lights completely, then add a hillside of brightness to the all black picture at that point. In this way, we can simulate the brightness of the picture after lighting up multiple LEDs on a completely black picture, which is called *add_mask* here. Figure 3.12 below, the left part of it illustrates the *add_mask*. It can be found that the more LED lights are turned on, the greater the superimposed brightness of the area.

Next, since we need to reduce the brightness of the corresponding LCD picture at the position where the LED is turned on, the next step is to reverse the simulated LED brightness. We are supposed to subtract the brightness of each pixel on *add_mask* by 1. And because some points on the LCD picture will be illuminated by multiple

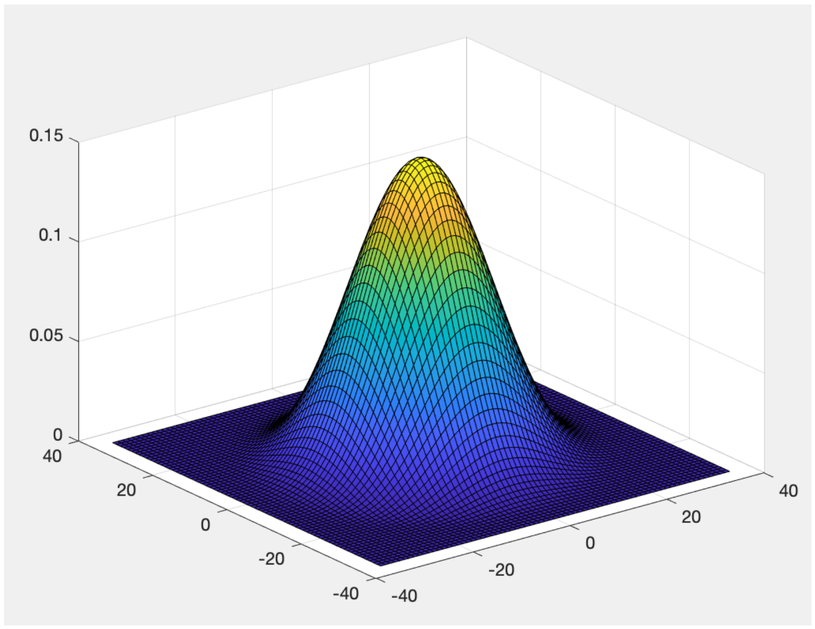


Figure 3.10: Simulation of LED light spot

surrounding LEDs at the same time, the position of the point will be reduced in brightness many times so that its brightness will appear negative. Therefore, we also need to directly set the negative number to 0 to generate a new *add_mask*. The right part of Figure 3.12 shows the new *add_mask*, which looks like the *add_mask* is upside down.

By the way, since there is gamma distortion on LCD, we need to apply gamma correction on the new *add_mask*, too. Finally, multiplying the obtained new *add_mask* by the original LCD picture will successfully get a subtracted LCD image which is demonstrated by Figure 3.13 below.

From Figure 3.13, we can find that there are several areas with obvious darkening in the picture, such as the sky and petals. This is because that several LED lights will be fully turned on in these areas and we need to reduce the brightness of the LCD in the corresponding position in advance.

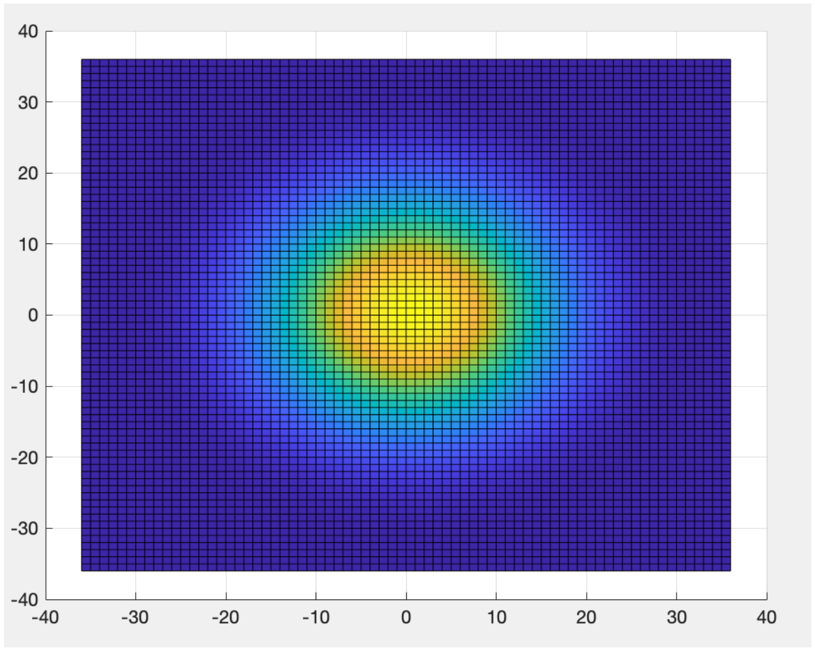


Figure 3.11: Top view of the brightness map of LED

After knowing how to adjust the LCD picture, the next step is how to adjust the parameters to make the picture look uniform in brightness. In this project, I choose to use a white LCD picture as the background of the LED and only turn on one LED which is white color, too. By the method mentioned above, we could generate a subtracted white image. Combining this subtracted LCD image with the white LED, we can adjust the parameters more conveniently.

We can make subtracted part of LCD image darker by slightly increasing the value of parameter a. Besides, changing the the radius of the assumed LED spot range (Figure 3.9) can help us adjust the size of the subtracted area so that the LCD picture and LED are more perfectly combined.

Figure 3.14 is a result of the combination of LEDs and LCD image. This sunflower picture not only has details, but also has consistent colors on some parts of it, which

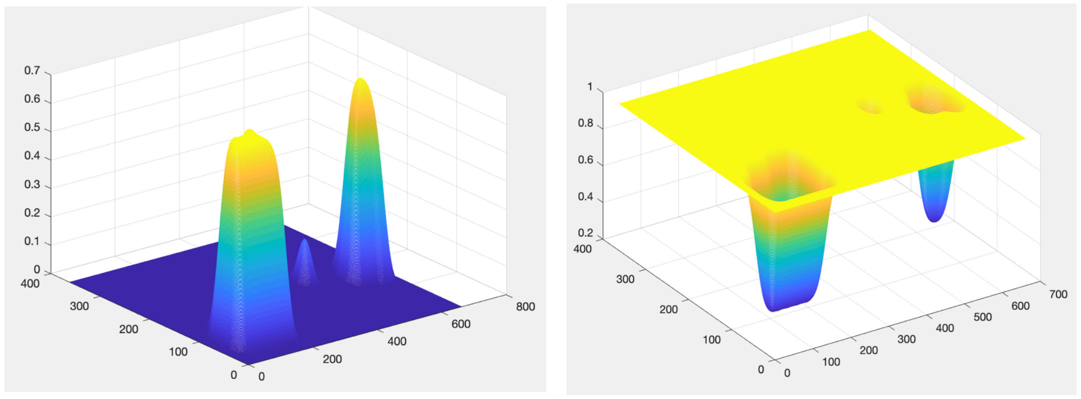


Figure 3.12: Brightness map of LEDs which are fully turned on(left one) ; The reversed brightness map(right one)

is a good example to represent normal pictures. We can find that the combination of LED and LCD is very good. Their positions just coincide after adjusting to the appropriate parameters, and the brightness looks very uniform and smooth.

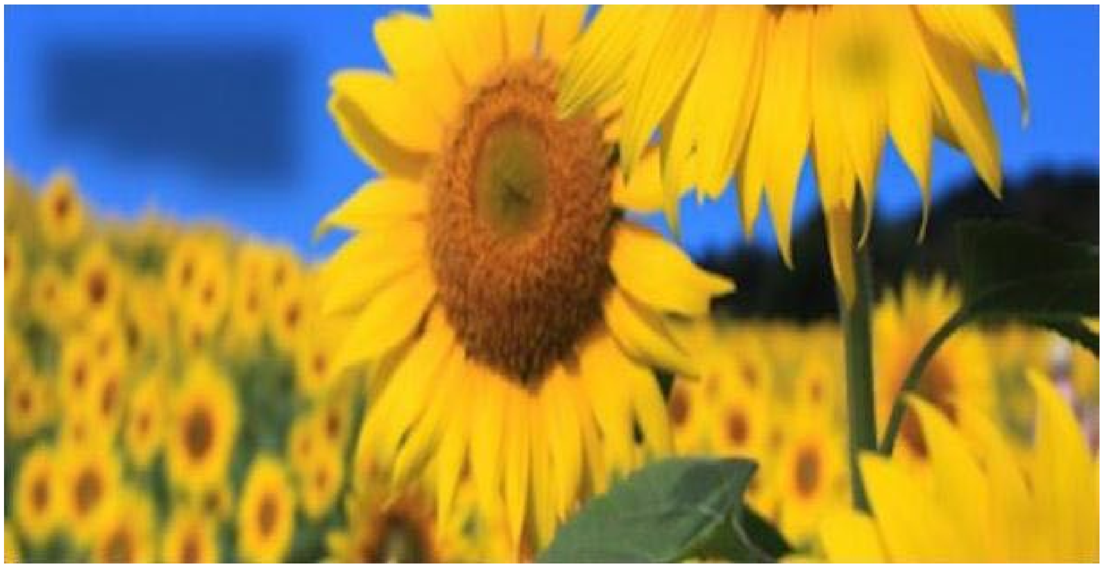


Figure 3.13: Subtracted LCD image

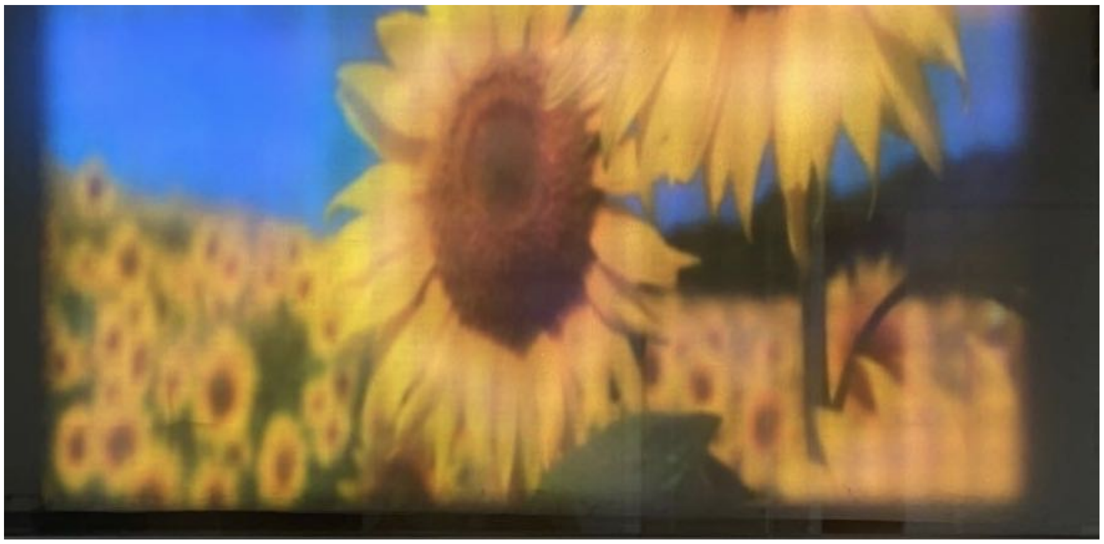


Figure 3.14: Combination of LCD and LEDs

Chapter 4

Conclusion

Since we have resolved the problem of overlapping, we could achieve a seamless video wall. There will be no gaps between tiled LCD boards, which helps viewers have a better viewing experience and feel more comfortable. In this way, if we add the number of LCD panels, the screen will become as big as you want without any gaps.

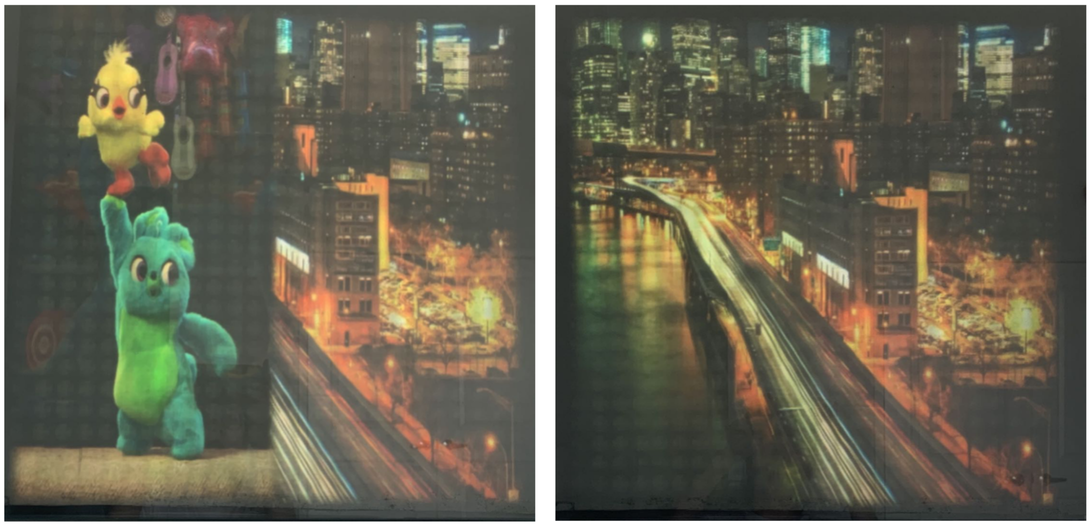


Figure 4.1: the Results of two LCD boards

For example, Figure 4.1 shows the results of two LCD boards. Compared with the effect of the previous LCD panel, the display interface has become larger, and there is no gap between the two boards. They can show the same picture or different pictures together.

What's more, there are many advantages of the combination of LCD and LEDs. First of all, since the power consumption of LED is much smaller than that of LCD, using LED lights to increase the brightness of the picture will be more energy efficient.

Secondly, this combination can improve the quality of images. As shown by Figure 4.2 and Figure 4.3, we can find that the addition of LED makes the picture look more vivid, the brightness become more uniform, and it can cover up the defects on the LCD hardware. It can be seen from Figure 4.2 that even if we have processed the brightness of the projected image, there still are some obvious vertical stripes and some white grid pattern due to the weakness of the hardware (Baker, 2013). But if we add RGB LED in front of them, the picture quality is greatly improved.

Finally, the LCD panels and the TLC which controls LED have low cost, and this would greatly reduce costs after making products in the future.

As said before, the general steps of image processing is similar to that of the article(GUO, 2020). Compared with the image processing in the article(GUO, 2020), there are some difference between this project and the project in article(GUO, 2020). We made some progress based on the previous one(GUO, 2020). First of all, we find the specific reason for the horizontal and vertical dark lines in the final image, which is explained in Figure 2.15 and has not been mentioned in other articles. Secondly, there is a color difference between LCD and LEDs in this project. Therefore, we try to find the reasons of it and transform the RGB data expressed by LCD coordinate system to the RGB data expressed by LED system. What's more, since the low cost of TLC59711, we choose to use it to control LED lights and solved the problems caused by it, such as the determination of scaling factor to convert 8-bits deep image to 16-bits deep image. The reasons and solutions for this part have been explained in detail in the section3.2.3. Finally, in this project, we found the solutions of some problems caused by hardware, such as the appearance of vertical stripes and white grids in Figure1.4 and Figure2.23.

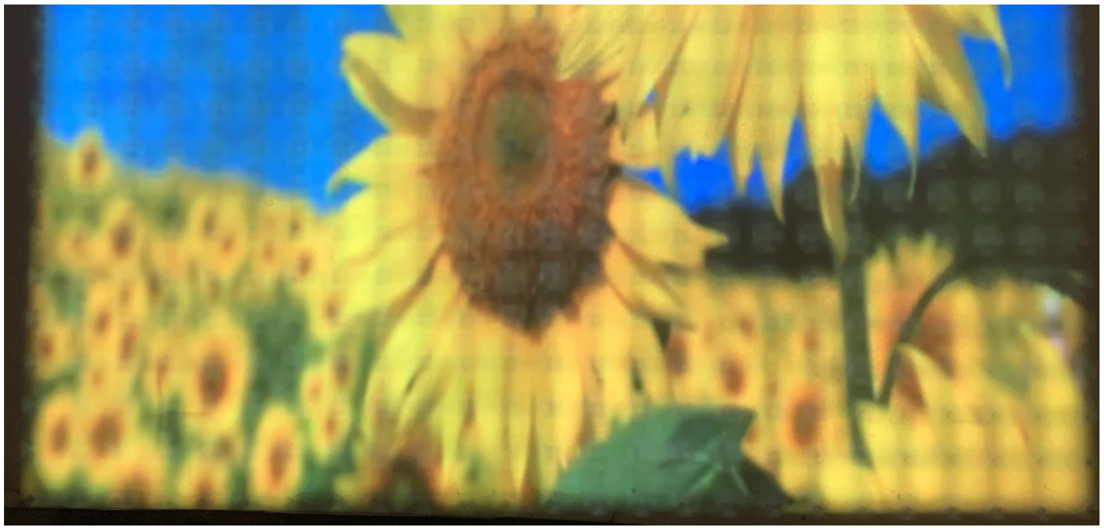


Figure 4.2: LCD image without LEDs

However, there still are something that should be improved. For example, the vertical stripes on the image. Even added with LEDs, the picture still does not achieve very perfect results because of some small misalignment between the front LEDs and the back LEDs. It is necessary to develop a future backlighting LED design to avoid this problem. Besides, in the process of matching the color of LCD and LEDs, we only use my eye to adjust them, which can not achieve ideal color matching. The best way is using use professional color detection equipment to form a more formal color lookup table between LCD and LED. And the last one is that

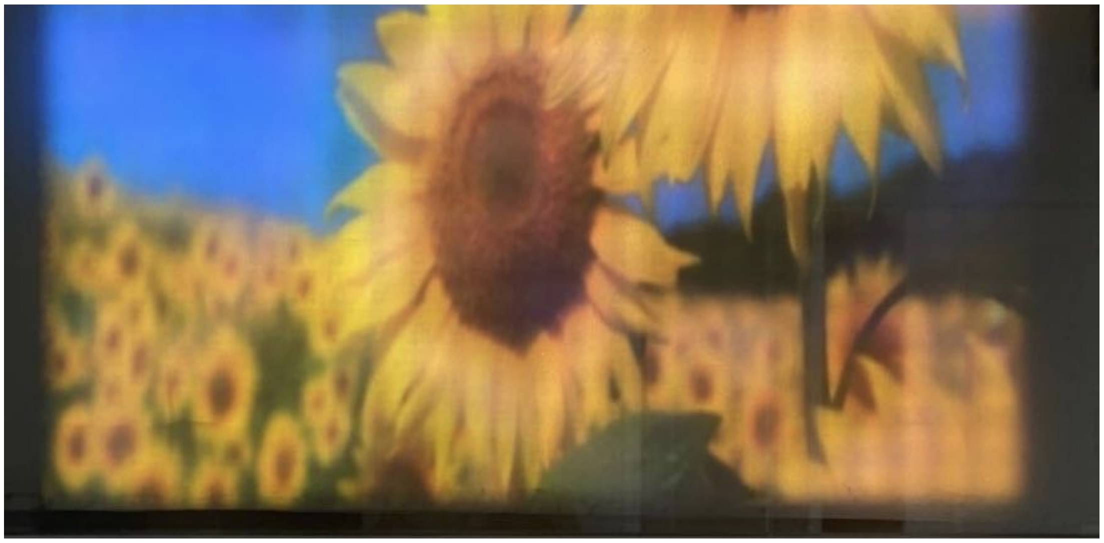


Figure 4.3: Combination of LCD and LED

we can try to develop from projecting pictures to playing videos. This aspect relates to the speed of each image processing (Kameyama and Miura, 2018). For example, we could improve the speed of reading images through multiple cores.

Bibliography

Baker, S. (2013). Panel technologies. *TNT Central, December*, page 34.

Chellappan, K. V., Erden, E., and Urey, H. (2010). Laser-based displays: a review. *Applied optics*, **49**(25), F79–F98.

GUO, D. (2020). Image processing algorithms for a tiled multi-projection screen.

Kameyama, S. and Miura, Y. (2018). Research for high speed image processing programming method on combined environment of cuda and opencv. In *2018 IEEE International Conference on Consumer Electronics-Taiwan (ICCE-TW)*, pages 1–2. IEEE.

Lee, P.-M. and Chen, H.-Y. (2005). Adjustable gamma correction circuit for tft lcd. In *2005 IEEE International Symposium on Circuits and Systems*, pages 780–783. IEEE.

Majumder, A., He, Z., Towles, H., and Welch, G. (2000). Achieving color uniformity across multi-projector displays. In *Proceedings Visualization 2000. VIS 2000 (Cat. No. 00CH37145)*, pages 117–124. IEEE.

Prentašić, P., Mikša, N., Hei, A., Subašić, M., and Lončarić, S. (2010). A method for projector brightness calibration for tiled displays. In *The 33rd International Convention MIPRO*, pages 1379–1383. IEEE.

Rahman, S., Rahman, M. M., Abdullah-Al-Wadud, M., Al-Quaderi, G. D., and Shoyaib, M. (2016). An adaptive gamma correction for image enhancement. *EURASIP Journal on Image and Video Processing*, **2016**(1), 1–13.

Sajadi, B., Lazarov, M., Gopi, M., and Majumder, A. (2009). Color seamlessness in multi-projector displays using constrained gamut morphing. *IEEE Transactions on Visualization and Computer Graphics*, **15**(6), 1317–1326.

Xiao, K., Fu, C., Karatzas, D., and Wuerger, S. (2011). Visual gamma correction for lcd displays. *Displays*, **32**(1), 17–23.