Abstract

This project is mainly to investigate the aligned algorithm for the high dynamic range image. The mainly discovering two main aligned algorithm which is suitable for the environment of different exposure image in the process of HDR composition. The two main algorithms are MTB (Medium Threshold Bitmap) algorithm and SIFT (Scale Invariant Feature Transform) algorithm. Through the investigation and study, MTB algorism is stable in the different exposure image. It can calculate the offset between two unaligned images. However, the original MTB algorithm does not consider the rotational alignment. Hence, this project adds the rotational alignment for the MTB algorithm, making it more suitable for Ordinary digital cameras. In additional, experiments are carrying out to discover the SIFT algorithm, making some improvement for the original SIFT algorithm which let it stable in the different exposure environment.

- The improvement of rotational alignment for MTB algorithm see page 30-34
- MTB + SIFT algorithm for the improvement of SIFT algorithm see page 41-42
- The experiment of the rotational alignment in MTB algorithm see page 43-47
- The experiment of the MTB+ SIFT algorithm for alignment. See page 48-56

Content

1 Introduction:

Traditional digital images, which have a color depth of 8 bits, can only express the depth level 256, i.e. a limited range of image contrast. High dynamic range (HDR) images can express changes in brightness changes in a wide range of images. Image pixel values (generally expressed as the decimal floating point) are proportional to the brightness value of the actual scene and correspond to the actual point. It is possible to better record the scene, bright area and optical properties of the dark zone and increase its minutiae. HDR images are widely used in augmented reality, image-based rendering, modeling, light, aerial photography, film and television special effects and so on.

HDR images can be taken using an appropriate camera directly; however these types of cameras are expensive and difficult to popular in the public. Hence an alternative method is to use a common digital camera to shoot the same scene several times with different exposure levels, fitting the light response curve of the camera to obtain the pixel value of the mapping relationship between exposures and then fuse different scenes into one HDR image. This means that when we take multiple exposed photographs, there are lots of different exposed pixels in the same object. We need to find the properly exposed pixel, and eliminate the underexposed and overexposed pixels. Therefore, in the final HDR image, the very dark and very bright pixels will be ignored.

However when we shoot a picture with an ordinary digital camera, it is inevitable that there will be jitter or tilt, etc., This in turn produces a small shift between images or rotation, resulting in a blurred synthetic HDR image, and therefore it needs to be alignment before fitting image registration. Furthermore, we also need to consider lots of other factors, such as camera response curves, and ghost and lens flare removal.

The classic multi-exposure on the HDR image, synthesis includes three basic processes: response curve calibration, HDR synthesis and alignment and HDR display. Now a day, algorithms which are used to capture HDR images are highly advanced. However, there are limits to the algorithms suitable for the methods of taking different exposures.

Hence, the main aim of my project is to find an appropriate algorithm for the alignment, and to evaluate its efficiency. In addition, we aim learn how to get camera response curves and use common digital camera to take different exposure images.

2 Backgrounds:

Traditional film imaging process is based on the theory of photochemical. When taking a photo with camera, light through the camera gets to the photosensitive crystal halogenations silver. This causes the change of optical density in the film. The greater exposure caused the smaller optical density which is a nonlinear relationship. Then after scanning and digitalization, it will be converted into digital images. Now widely used of digital cameras are using the image sensor (usually CCD and CMOS) which receives signals from the image sensor is proportional to the discrete light-sensing cells in the exposure of thousands of pixels. After that, this information converted into digital signal. Finally, thought the nonlinear operation by the microprocessor, it converts the digital signal into the operation standard as BMP, JPEG storage formats, and stored in a physical medium as TIFF.[4](see Figure 2.1 below)

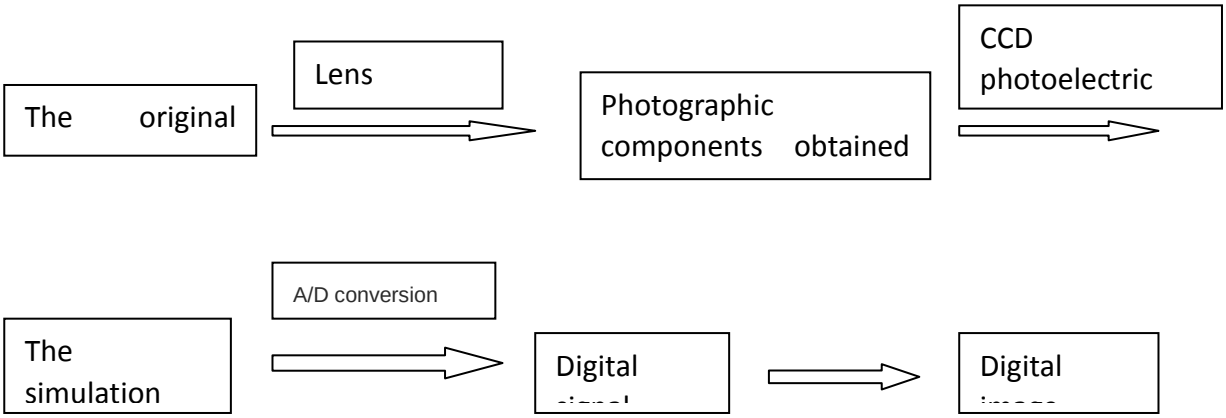


Figure 2.1 Pipe line of digital imaging[3]

2.1 The dynamic range of digital images

In a real world scene, the value of luminance is distributed in a wide range. We define the L as the values of luminance and a former luminance unit is CD/M2. Lmax and Lmin are minimum and maximum brightness values. The ratio of the maximum and minimum values is the dynamic range in this scene.[6] We can use a symbol (A) to indicate this dynamic range:

$$A = Lmax / Lmin$$

In direct sunlight, the brightness of the scene can reach 10^5CD/M2, while the darkness of shadows area just about 10^{-3}CD/M2. This means that it is a huge dynamic range in the real word. (SEE figure 2.2)People's visual system for the brightness values are from a range of 10^{-2} CD/M2 to 10^5CD/M2. However, Human Visual System can be adapting to strong light. Generally speaking, people in the same situation can achieve the brightness of the dynamic range to approximately 10,000:1.

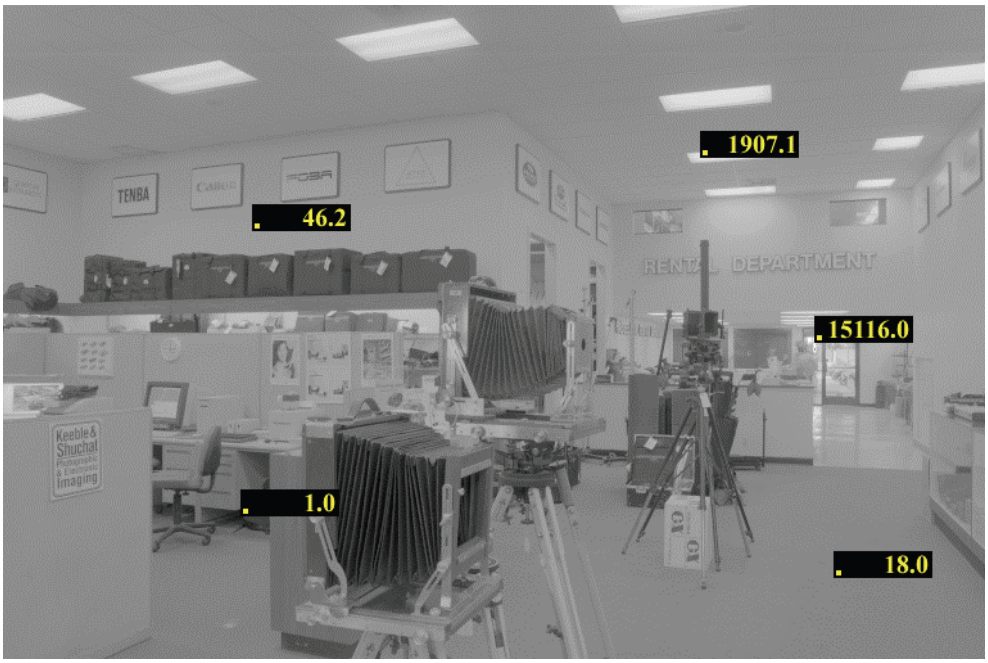


Figure 2.2 the dynamic range of real word. The maximum luminance is 15116.0- , the minimum is 1.0

Traditional digital image is limited in expressing the real word dynamic range. For example, it uses 8bit to store tricolor (red, green, and blue). Hence the range of this bitmap can just be indicated to 255:1. This causes the detail in the brightness and darkness environment lost. Therefore we need a new format to store a higher range of image.

2.2 High dynamic range images.

Traditional digital image is limited in the channels with 256 levels which are not enough to show the dynamic range of real word. High dynamic range images (HDRI) is a real scene that can change the range of brightness [5].It can better express the feature of darkness and brightness area. The range of High dynamic range image usually

reaches hundreds of millions; each color channel needs more bits in the High dynamic range image. It needs to indicate the eye visible range from 10^{-2} CD/M2 to 10^{5}CD/M2 which often use 16bit half precision "or" 32-bit floating-point to represent the range of pixel. Some applications of high dynamic range pixel can use 10-12bit to indicate the luminance, and 8bit for the chromaticity which will not bring any visible quantization error.

There are three types of methods to catch the HDR image.[5] The first kind is based on the physical model of illumination lighting which is the most important method for the composition of HDR image. The second is the use of various exposure images which are in the common dynamic range (LDRI) to calculate the actual brightness and composite the HDR images. The third type is the use of special hardware equipment to shoot HDR image directly. The first kind is a process of manual-synthetic which cannot handle the natural images. The third one also needs the professional equipment and it is very expressive. Hence the most common way is the use of different exposure to composite the image. The figure (2.3) show the different exposure image which are used for the HDR composition. Their brightness is in image sequences from the dark to the light. In these pictures, each image shows good detail in different area in different situation. In order to get such a picture, we can control the exposure time. There are two main methods to control the exposure time: control and adjust exposure camera aperture size. Note that the more exposure in filming, scenes must motionless; otherwise, even small jitter will also cause fuzzy synthesis HDR.

Figure 2.3 Multi-exposure images.

High dynamic range is very widely use in application domain of images, such as: based on illumination technology of image and augmented reality with high dynamic range images of real or lighting of computer-generated object can obtain vivid effect, based on the modeling and rendering image and image synthesis using high dynamic range images can avoid non-normal exposure to graphics, additionally, high dynamic range images are widely applied in movie stunt, human visual system simulation, satellite remote sensing image, motion blur and so on.

High dynamic range image has very wide application domain. For example: the lighting technology based on image and augmented reality with high dynamic range images to lighting of real or computer generated object which can obtain natural effects. Besides that it can be use in the Computer Game, Movie stunt, and human visual system simulation.

Physical light rendering. It can be called the first application of a HDR image. In order to achieve precise rendering effects of light, it is necessary to store some beyond human

The satellite remote sensing. The satellite remote sensing image contains more information than observer from the eye.

Digital imaging. At present more and more digital camera maker began to seek support for HDR photographs. Professional camera now has appeared on the market, which can directly take HDR image. Through the HDR image, people can enjoy more lifelike visual experience.

Digital cinema. Digital cinema is one of important applications for HDR image.

Virtual reality. Many web sites require high speed in image transmission. Therefore, the image needs to do the lossy compression before transmission. There is a good solution for this problem when using the HDR image compression.

2.3 Research status

Due to the good prospect in the high dynamic range images, large quantities of scholars are attracted to research in this aspect. There are three main research directions in the high dynamic range images.

High dynamic range image acquisition
High dynamic range images of storage
High dynamic range of image shows, namely visualization

2.3.1 The generation technology of high dynamic range images

In the area of capturing the dynamic range images, there are special devices used to obtain this image. However, the prices of these devices are very expensive which make

it hard be popular in our common life. Research in this area tends to use the ordinary digital camera equipment to capture the high dynamic range images. Figure 2.4 shows through different exposure to composite a high dynamic range images.

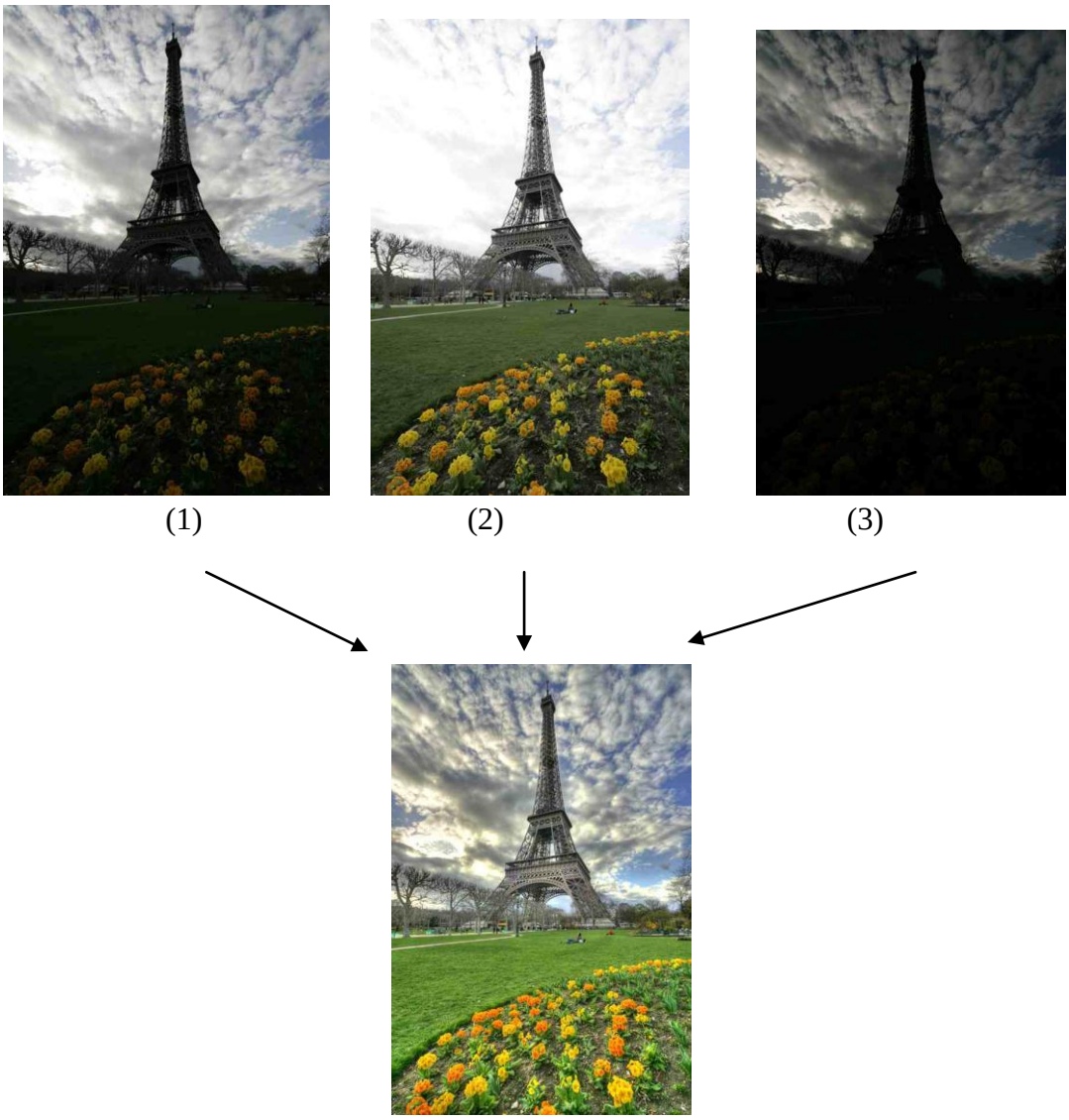


Figure 2.4 Difference exposure image composite to a HDR image.[7]

Image (a) is the moderate exposure image which is we normally take from the digital camera.
According to the people's vision, the brightness of the region retains in balance in the scene. However the cloud in the sky is the highlight areas which are over exposure and the darkness area cannot show enough detail. Image (b) is overexposed image, it get more detail for the darkness area. But it losses information for the brightness area. Image (c) is the underexposed images which get the information of Highlight areas. It is better to performance the detail of sky.

Debevce [1]proposed a method which using the same scene to different image exposure

to composite a HDR image. This method is based on the principle of capture the image. In the process of creating an image, the CCD uses a nonlinear response function for mapping on the pixels. However, we cannot get this response function in the camera's manual. So the key is how to restore synthesis of the nonlinear response function. In 1995, Mann[10] had given a method for restore the camera response function. After that, Debevece[1] presented a method to solve the target function for recovering the response function. Besides that Robertson [11]puts forward a method to solve the probability. Therefore there are lots of methods to recover the response function now.

2.3.2 High dynamic range image storage technology

The other very important aspect for the HDR image is how to stores it. Different from the traditional 24 bits of RGB images, high dynamic range image storage is more complicated. It needs to consider the question how to reduce the disk space and express high dynamic range image.[8] There has been some file formats to store HDR image which have been developed (shown Figure 2.5 below). The main three of formats for the HDR image are HDR format, TIFF format, and EXR format.

Format	Encoding(s)	Compression	Metadata	Support/Licensing
HDR	RGBE	Run-length	Calibration, color space,	Open source software (*Radiance*)
	XYZE	Run-length	+user-defined	Quick implementation
TIFF	IEEE RGB	None	Calibration, color space,	Public domain library (*libtiff*)
	LogLuv24	None	+registered, +user-defined	
	LogLuv32	Run-length		
EXR	Half RGB	Wavelet, ZIP	Calibration, color space, +windowing, +user-defined	Open source library (*OpenEXR*)

Figure 2.5 The main format for the HDR image.[9]

HDR format images (HDR, PIC, rad) is also called illumination format (radiance clusters, it contains a), a period of document image size of string, and through the run-length coding processing image data. For each pixel, it generally uses RGBE coding (below Figure2.6) 32 bits.[9]

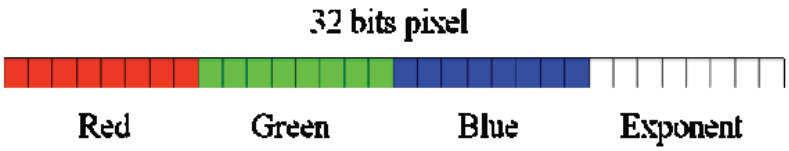


Figure2.6 HDR format with RGB encoding

RGBE coding indicates each component(R E G B) can be transformed from the real values according to high dynamic rule

$$E = \lceil \log_2(\max(R_w, G_w, B_w)) + 128 \rceil$$

$$R_M = \left\lfloor \frac{256 R_w}{2^{E-128}} \right\rfloor$$

$$G_M = \left\lfloor \frac{256 G_w}{2^{E-128}} \right\rfloor$$

$$B_M = \left\lfloor \frac{256 B_w}{2^{E-128}} \right\rfloor$$

RM GM and BM indicate the high dynamic RGB. This means that the range of dynamic range image can reach more than 70 levels.

(Open extended range format, openEXR) is developed by a studio in Unite States. It has the advantage of open source high precision, high expansibility and nondestructive compression. It can be used in film and television animation. [9]Each openEXR channels are adopted S5E10 (a symbol, five indices, 10 digit, 16 bits), thus each pixel uses 48 bits of code, which can indicate maximum brightness value of 65, 504,

2.3.3 High dynamic range images visualization technology

Through the introduction, we know ahead of dynamic range for natural scene is about 10,000,000:1. Our eye visual system can observed this high dynamic range in the scene. It can used special devices to display high dynamic range images which give people the true feeling of natural scene. For example, Ledda[12] uses virtual reality system LEEP optical instruments which had been used for NASA to build a simple set of experimental device. It can be directly observed by high dynamic range in the real

world. Besides that, Seetzen uses color LCD mixed with LED to design a display panel for the high dynamic range of image display.[13]

However the most conventional display equipment can support a relatively low dynamic range of display output: traditional display devices generally have two orders of magnitude of dynamic range. The equipment includes cathode ray tubes (CRTS), liquid crystal displays (LCD) and the projector. These entire devices will loss brightness detail when they display the high dynamic range images. Therefore, how to display high dynamic range images in low dynamic range of display devices on the optimization has become a more and more important. So new technologies have been created to display HDR image in the normal device which is called tone-mapping.

Figure 2.7 indicates the process of scene mapping to traditional display equipment.

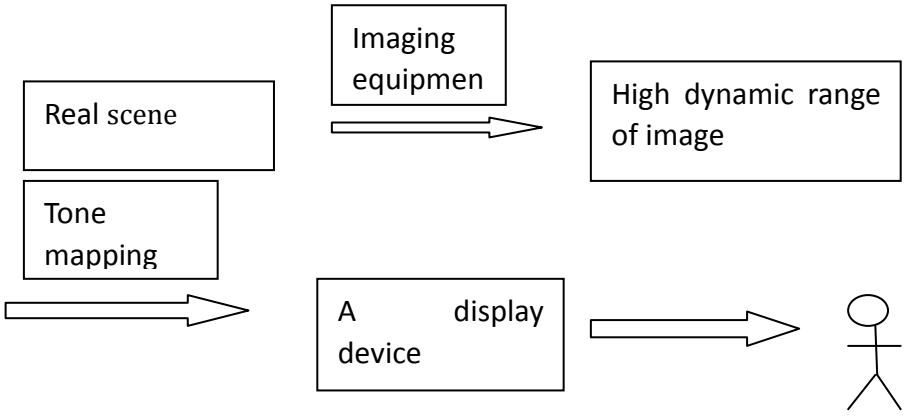


Figure 2.7 the process of HDR image tone-mapping[14]

There are two main aspects in currently work for the Levels reconstruction. Firstly, one research in creating better reconstruction algorithm, making the high dynamic range image perform better in detail after the reconstruction, to reduce the loss of information of contrast and bright color. On the other hand, let the reconstruction of HDR image integrate into application in daily life environment.

2.4 The same scene of different image acquisition exposure

Before we do anything for fuse the HDR image, we need to take different exposure images which are taken by the common camera devices. We will discuss different methods for taking exposure images.

Natural real scene brightness (luminance) is a high dynamic range, while the traditional digital image is limited by access method (such as film or CCD) and information storage format that can only save a limited range of brightness information. Great scene brightness (such as sun light) or small (such as dark corners) in the region are usually difficult to capture the details. In practice, we can overexposure and underexposed to obtain details of these regions. However, the expense of these methods will lose other brightness area of information. Therefore, in order to cover a complete the dynamic range of the real-world scenarios, we need a series of different exposure images.

Debevec and Malik [1] proposed an algorithm to use multiple images with different exposure to synthesize for high dynamic range image. The research contributed greatly to the development of high dynamic range images and applications.

To achieve a high dynamic range image, we need to take a series of the same scene with different exposure photos. Here are two main features of capture this photo:

1 Must be the same scene.
2 Have different exposure.

Through the control of exposure to each image can make them with different exposure. Generally speaking, the control of exposure has three mainly methods: set a shutter speed, adjust aperture size and use neutral filter. The following will introduce three methods with their advantages and disadvantages.

2.4.1 Shutter speed

The shutter speed are usually described by the time seconds (s). It represents the length of exposure and the length of time for light into the camera. Usually all different types of digital camera shutter speed are completely different. The shutter speed between a class, usually knock each level of exposure to change, For example: 1/6, 1/8, 1, 2, 4 seconds. Two adjacent differences between the amounts of light. 2 seconds a shutter speed of light is twice as 1 SEC. The shutter speed directly relates to the exposure time. Hence in the long exposure, noise in image will appear, and hands shaking can cause blurring (use the tripod for fixing device can solve this problem). Currently, the medium price above the digital camera has a function of manually setting a shutter speed. Therefore, setting a shutter speed to control the exposure is the most commonly used and convenient method.

2.4.2 Neuter filter

Filter is a kind of optical devices which has different wavelengths light with the selective absorption. It is consist of mirrors and filters, setting in front of camera or camera lens. Neuter filter can absorb different wavelengths light, reducing the light of object which is suitable for the different exposure. Besides, it will not affect to the color contrast. It makes us gain greater flexibility in adjusting aperture and exposure time. Filter can stack use, which makes it can get a great range of exposure change image. However, the use of filter will cause the migration image. Otherwise, it will make the image with deviation.

2.4.3 The aperture size

The aperture is used to control the light through a lens, enter the light within the fuselage of photographic equipment, it is usually in lens. We used to say aperture size F values. F values are the lens of the camera lens diameter of focal length and diameter ratio. From this we can know about what to achieve the same aperture F value, the caliber of the lens aperture long-focal-length is larger than a short-focal-length lens aperture. Moreover, F is the aperture value of the coefficient of relative aperture. It is not the physical aperture diameter, relating to the distance of the physical aperture and lens aperture to the light-sensitive device (film or CCD or CMOS). When the aperture constant physical aperture, the lens center and light-sensitive devices are farther away, instead of F number is smaller. Whereas the lens center and closer the light-sensitive devices, light-sensitive device through the aperture to reach the higher optical density, F number is greater. The full aperture value range as follows: F1,F1.4,F2,F2.8,F5.6,F8,F11,F16,F22,F32,F44,F64. It has greater aperture, when the value of F is smaller.

In practice, we can use shutter speed and aperture size with together to control the exposure. Beyond the range of shutter speed settings, change the aperture size to get the same effect. But changing the aperture size will change the image of the depth of field (depth of field is the actual focus area around the focus can be clear) and do not have a strong repeatability. Only through the aperture to control the size of the change in exposure, it is very limit in changes of exposure range.

To make a conclusion that the method of change shutter speed is easy to control and would not impose an impact on image quality, while the aperture size of the change will affect the image depth of field, filter use is not convenient enough and can cause color deviation. And with the basic middle-price digital camera has a shutter speed

and aperture size of the manual setting function, while the filter requires a separate purchase. So practice, changing the shutter speed is usually the main adjustment aperture size and the use of neutral filters supplement.

2.5 HDR ALIGNMENT REALIZING

When we use the common digital camera to capture the different exposure image, two main problems will happen:
1 There will have tiny camera movement during in the fixed-point shooting, leading the synthetic high dynamic range image will become blurred (blurry). Even we use the tripod for fixed, slight movement or rotation still is caused by unequal force during we press the shutter or the uneven ground.
2. At the time of capturing the image, if the objects that in the scene are moved then make the final synthesis of high dynamic range images appear Phantom (ghost). This often happens when shooting outdoors, such as moving people, clouds and the trees by the wind.

2.5.1 Related algorithm

Tomaszewska proposed matching algorithm which based on SIFT feature[15]. SIFT algorithm is more efficient extraction of images similar to the corresponding feature points of scale-invariant, but its large amount of calculation, not suitable for real-time image registration.

Greg Ward[9] has proposed a fast matching algorithm which based on binary image: the value of binary image registration (Median Threshold Bitmap, MTB). In the algorithm, according to their median or mean gray value, it let different exposure image become image binaryzation.
It can produce a stable exposure time of image. After that, in order to achieve registration, we can search from the image pyramid-based method, determining the horizontal direction translation between two different exposure images. It is a fast method of algorithm, because it only base on the bit computing. However, its shortcomings are obvious. The disadvantage is mainly in the MTB alignment. It only searches the shift amount in the X, Y direction. Otherwise, it matches on other direction. Such as rotate in the image.

2.5.2 MTB algorithm

When the image is essential to planar objects or targets far away from the lens, the object's 3D shape can be ignored. In other words, the image registration between the relationships with each other exposed image can be simply treated as X, Y plane translation or rotation. Greg Ward (in 2003) raised MTB (median threshold bitmap) method [9] that is applicable to this occasion. MTB algorithm calculates the respective middle values in multi-exposure image, changing it into binary value. After that, it using the pyramid search method which can get the direction of Horizontal and vertical offset. However, MTB methods ignore the information of three-dimensional shape, leading to significant and ghosting when a large and close target has been alignment and fuse.

The basic thought of this algorithm is get one image from different exposed image. It has been proposed to choose the proper exposure image. After that, this image compare with other image, getting their relative offset. The image is shifted by the offset to complete alignment.

2.5.3 Based on 3d registration: SIFT algorithm

In order to accurately know the relationship between match image and reference image
We need to find the target motion parameter model. Image motion can be divided into several categories: translation, rotation, scaling, and affine and projection.

For the reference image pixel coordinates (x, y), matching the corresponding image pixel coordinates can be expressed as

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \sim \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

~ means Equal proportion, Cij for the unknowns .When considering two-dimensional plane movement, C33 = 1. Transformation matrix of the parameters to take a special form .It can be translation, rotation, scaling, and affine and projection transformation matrix.

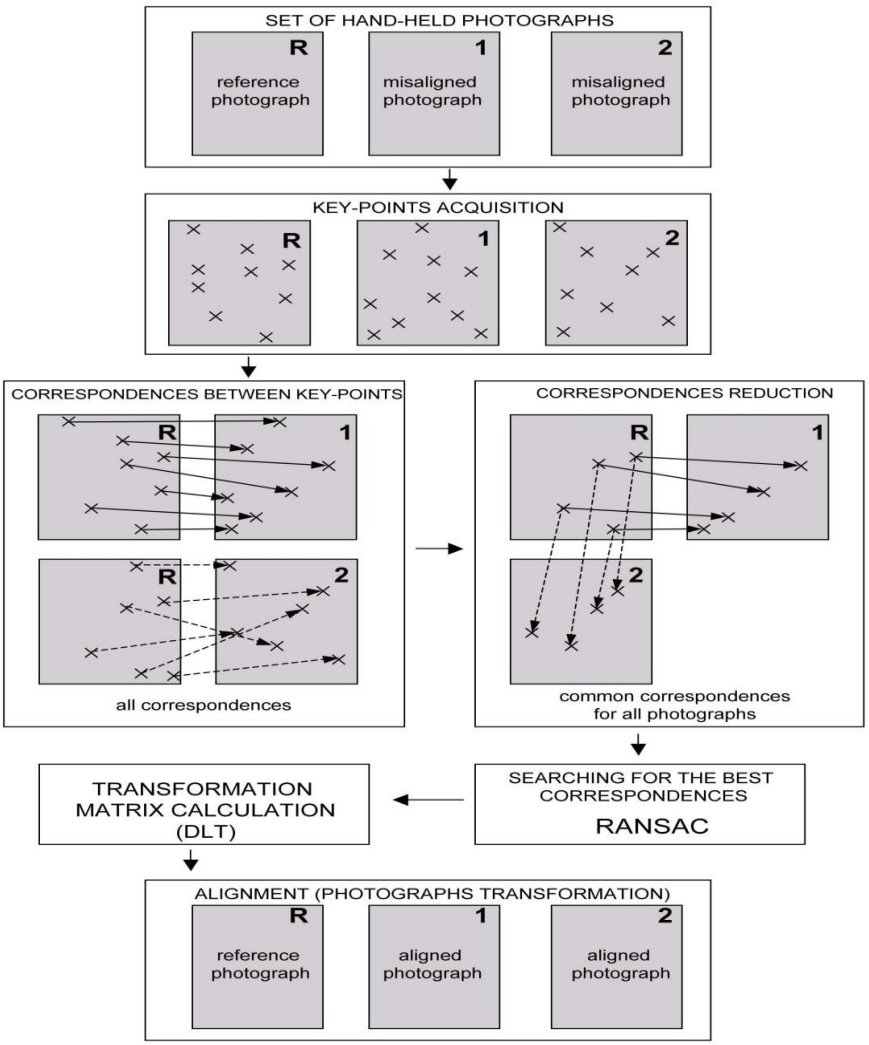


Figure 1: The SIFT algorithm of the image alignment technique (take from ref[15])

SIFT methods is different to the pixel gray level of MTB methods. It is base on feature image registration algorithm which extract the feature point for the matching, using the corresponding feature point position information to solve motion parameters. Tomaszewska proposed SIFT Registration [15] The SIFT algorithm for positioning to be registered images and reference images in the scale invariant feature. This method is effective when the image have the moderate exposure time. Otherwise, when the exposure time is too large or too-hour. SIFT algorithm will increase on extract errors feature point. The corresponding feature points also increased false matches. This will cause significant errors in the estimated parameters which lead to fail in registration. In addition, SIFT the large amount of calculation, not suitable for real-time system.

2.5.4 SURF algorithm

SURF (Speeded Up Robust Features) published by the Herbert Bay and others in ECCV2006 .[16]
There is a great improvement at the speed and robustness compare with SIFT features. It has been widely used in target identification and tracking. SURF algorithm gets the interest points of the location and scale information through calculate the Hessian matrix determinant. Hessian matrix defined as follows:

$$H(x,\sigma) = \begin{pmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{pmatrix}$$

We can identify the interest points which are the maximum point in the Hessian matrix determinant in scale space and image space.
Besides SURF algorithm, there are several other feature extraction algorithm. According to Herbert [15] is shown, the feature extraction method in SURF algorithm is 5 times faster than the way Hessian-laplace, 12 times faster than the way Harris-Laplace, about 3 times faster than the DoG.

SURF feature descriptor for the image feature point with the angle, scale, rotation and illumination invariant change. These properties of SURF let it suitable for the image registration problem. When the feature point position is confirmed, neighbor search method, access to reference map and matching graph corresponding feature point pairs

SURF registration model shown in Figure 2. First of all, the group is calculated and the corresponding match image feature points, and then we use the minimum square method for calculating the value of sport in the transformation matrix (1) of the various parameters. Finally, the transformation matrix C will be matched to the reference map

$$X' = CX$$

$$C = \begin{pmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & 1 \end{pmatrix}$$

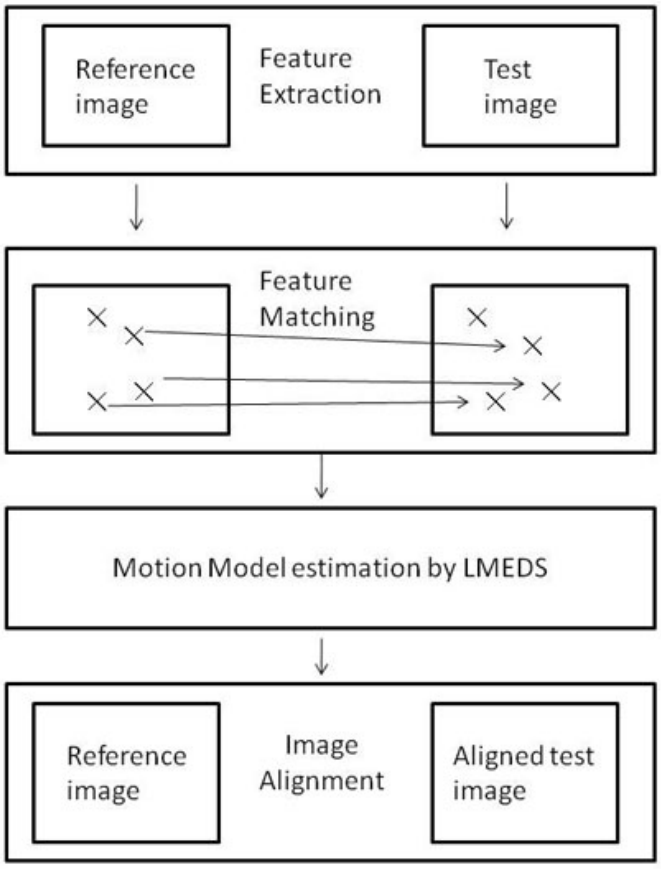


Figure 2 SURF registration model make from Ref[16]

We note SURF characteristics of light and color changes in a robust, so the algorithm for different exposure of the image registration can be achieved very good results. In addition, 3×3 transformation matrix include the image shift, rotation, perspective, parallax, etc. it is more precise than the MTB algorithm which just consider the amount of horizontal and vertical translation.

Transformation matrix C have a total of eight unknown parameters, the matching feature is generally far more than the number of unknowns parameters. Generally, we use least squares estimation, calculating all the point involves in the transformation matrix. But in practice, there are a certain error in the process of extraction and matching. Hence we can
use a good anti-noise level resistance to the minimum value method (Least Median of Squares, LMEDS).

To make a conclusion that all this algorithm can do the alignment for different exposure images. Each has it is feature. We can choose appropriate algorithm according to our requirement

2.6 Camera response function

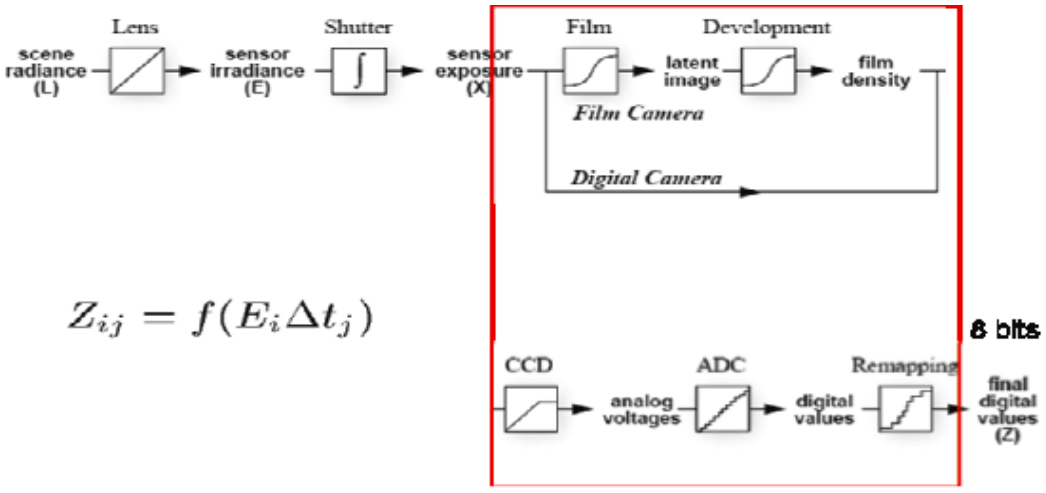
After we finish the alignment we can carry on the HDR image synthesis. Now, we will discuss the basic principle of synthesis.

2.6.1 HDR exposure synthesis principle

When we took a scene, digital image is a bright degrees 2d matrix. These values are not the real value of brightness scene. For example, the image of two points of pixels is double difference, but the actual scene might not observe twice. In actual fact, the scene with the image of the luminance values between pixels exist some unknown nonlinear mapping relation. The mapping relationship cannot found in any instruction of the camera. Because it is created in the process of combination of some nonlinear mapping results.

The following figure shows the camera captures the scene in the schematics (see from Figure 3):
Known condition is the ultimate source of all channels 8bits that image; our aim is to restore the actual scene of the exposure X. There are a linear relationship between the original image and the actual scenes. It covers the internal processing of a series of nonlinear camera processing. We call it the camera's response curve (Camera Response Curve). By pieces of the original image to calibrate the response curve, you can recover the actual scene illumination, the ultimate get HDR image.

Figure 3: HDR Camera Respond Curve make form REF[15]



As shown in figure 2.1 imaging process, the exposure (x) is proportional to the sensor irradiance (E) and value of shutter speed (T).

$$X = E\Delta T;$$

Subsequently, a nonlinear function is applied to exposure (X), the function is called "characteristic curve. For the characteristic curve, when the exposed time close to 0, high exposure can produce saturation phenomenon.

$$Z_{ij} = f(X) = f(Ei\ Tj)$$

2.6.2 The camera calibration algorithm of response curve

By LDR figure synthetic HDR figure need to know, if the camera response curve is given, not to need some algorithm is presented. At present more classic to basically have two kinds of methods: (1) Debevec and Malik algorithm [1] (2) Nayar algorithm. These two methods have different features.

2.6.2.1 Debevec and Malik algorithm

Debevec Malik[1] and use [15] a group of image, precise exposure that can be more precise result. This algorithm for response function not only strictly limit, consecutive. Debevee by solving linear equations to get the response function of the camera, this function in a gray values and the corresponding 256 levels of scene of table describing exposure value

Form the function which we get from previous chapter:

$$Z_{ij} = f(X) = f(Ei\ Tj)$$

i represent the I pixel in the image , j represent which j image we take. E means the illumination, T means the time of exposure.

The algorithm input a series of images points (how we will be talking behind these points), the exposure time Tj (j means the j image we take) has already been recorded. To assume that the scene is stationary and made a series of image in very short time intervals, namely the image of light has not changed. Then we can use Ei to indicate the irradiance of i pixel in this picture.

After that, we assume that f is monotonically increasing, and then it is reversible, Hence we can get a new expression:

$$f^{-1}(Z_{ij}) = E_i\Delta T_j$$

We get the logarithm from this expression

$$\ln f^{-1}(Z_{ij}) = \ln E_i + \ln \Delta T_j$$

And we define the h = $\ln f^{-1}$ we get a new expression:

$$g(Z_{ij}) = \ln E_i + \ln \Delta T_j$$

In these equations, i can get the range of the number of pixels. The range of j is the image of different exposure the number of sheets. Z_{ij}And ΔT_j are known, but g and Ei is unknown.
These two parameters need to solve by us
In traditional digital image pixel value range is limited: from 0 to 255. So we do not need to obtain the specific expression of F, obtaining the value of 256 F (Z). We use integer $Z_{min}$and Z_{max} to indicate the largest and smallest pixel value, N said that the number of pixels in the image, M show the number of sheets of image. The problem can be expressed as the method of least squares objective function:

$$o = \sum_{i=1}^{N}\sum_{j=1}^{M}[\ g(Z_{ij}) - \ln E_i + \ln \Delta T_j]^2 \ + \mu \sum_{z=Z_{min}-1}^{Z_{max}-1} F'(z)^2$$

The front part of the equation for solving the curve, but, $\mu \sum_{z=Z_{min}-1}^{Z_{max}-1} F'(z)^2$part is used for smoothing the curve, and $F'(z) \ = F(z-1) - 2F(z) + F(z+1)$ .Variable μ used to control the smooth. Debevee use single value decomposition (sVD, singular uedecomposition) method to solve the linear equations.

Besides that, we need to introduce additional conditions, which lead to the pixel in the middle of and Z_{min} has the unit exposure

$$g(Z_{mid}) = 0 \ and \ Z_{mid} = 1/2(Z_{max} + Z_{min})$$

In addition, we address one of feature in responses curve: the pixel value close to the limit (ie, Z_{min}, and Zmax) of the region, the curve will become steeper, not smooth. Considering the curve in order to obtain as much as possible to meet this feature, the algorithm weight function $w(z)$, this is the following simple function:

$$w(z) = \begin{cases} z - Z_{min}, & z \le \frac{1}{2} Z_{mid} \\ Z_{max} - z, & z > \frac{1}{2} Z_{mid} \end{cases}$$

From this we can get a new expression:

$$o = \sum_{i=1}^{N}\sum_{j=1}^{M}[\ g(Z_{ij}) - \ln E_i + \ln \Delta T_j]^2 \ + \mu \sum_{z=Z_{min}-1}^{Z_{max}-1} [w(z)F'(z)]^2$$

We can obtain the light response curve by solving the equations. However, this method is suitable for accurate exposure of the known rate and image noise is relatively small.

2.6.2.2 Mitsunaga&Nayar algorithm

Debevec and Malik ‘s [1] algorithm for calculate the CRF TO get a 256 element mapping table which will mapping to the logarithm of exposure. However, this algorithm needs to know the specific image exposure time information. The small handheld devices such as consumer digital camera, etc, to know exposure time or aperture size is very difficult. Even we know the exposure time, because of the hardware conditions and other factors will also introduce measurement error. Nayar [17]proposed a new algorithm using the elastic parameters of model that we only need to know the time exposure adjacent exposure image, without the need to ratio know the exact time exposure can give CRF.

First, CRF said to Z = f (I), Z for grayscale, I for the scene illumination. While $I = f^{-1}(Z) = g(Z)$, assuming a polynomial g says:

$$\varepsilon = \sum_{i=1}^{q-1}\sum_{j=1}^{p}[\sum_{n=0}^{N} c_n Z_{ij}^n - R_{i,j+1}\sum_{n=0}^{N} c_n Z_{i,j+1}]^2$$

It only needs to determine the polynomial coefficients of each value of Ci determined function g. Mitsunaga and Nayar ‘s error function is constructed as follows:

$$\varepsilon = \sum_{i=1}^{q-1}\sum_{j=1}^{p}[\sum_{n=0}^{N} c_n Z_{ij}^n - R_{i,j+1}\sum_{n=0}^{N} c_n Z_{i,j+1}]^2 \qquad \textbf{(2-10)}$$

q for different parameter specifies the number of exposure, p for image pixels points.

We can use simple partial derivative method to calculate the minimum value. For each Ci partial derivative for 0.

$$\frac{\partial \varepsilon}{\partial c_i} = 0$$

2.6.3 HDR compositional method

After we finish the light of the response curve fitting, then we can finish by image pixels to the scene of the conversion. A series of source image can through to the corresponding linear curve (divided by exposure), and add up all the image sequences which was linearized can get high dynamic range images.
Using this formula can be obtained corresponding pixel values of illumination, restore function of p image to calculate the pixels intensity of illumination:

$$\ln E_i = \frac{\sum_{j=1}^{p}(g(Z_{ij}) - \ln \Delta T_j)}{p}$$

Yet we know that in each of the final image exposure, there must be in the following three cases: (1) underexposed point (2) overexposed point (3) appropriate exposure. While (1) (2) contains no accurate information (usually by noise, etc.), it should be place in synthesis. Namely, not all points can be reliably in the synthesis of high dynamic image. In order to solve this problem, we can point to the corresponding compensation a weights, Hence we get another weight function:

$$\ln E_i = \frac{\sum_{j=1}^{p} W(Z_{ij})(g(Z_{ij}) - \ln \Delta T_j)}{p \sum_{j=1}^{p} W(Z_{ij})}$$

□(□_□□) is the weights

Currently there are many Settings weights method. Mann and Picard once put forward the response function of the derivative as weights, its reason is the reliability of the pixel camera and the changes of light is on the sensitivity, Debevec and Malik[1] and puts forward a simple hat shapes triangle weights based on gray, near the center of the assumption is more reliable pixels.

Through the weighting function, we let the pixels which close to the middle of exposure response function have higher weight. Thus, saturated pixels were ignored. It make a make a better synthesis results Using multiple images to calculate the final illumination values can be generated to reduce noise, reducing the image quality problems caused by shadow.

This chapter focuses on how to use the same scene of different exposed image to make high dynamic range images. It describes the two important steps in the methods of synthesis. It address how to recover response curve and get the scene luminance values according to the response curve

2.7 Tone-mapping

HDR image has high dynamic range. Although it can display the real scene, the hardware such as LCD, CRT cannot have a good performance for the HDR image. In order to show high dynamic image on the low-end display device, we must adopt a kind of algorithm to compress the dynamic range which limited to adapt by the device. This image compression algorithm will adjust brightness as tonal mapping. Currently scholars have proposed many tone-mapping algorithms. It can be divided into several categories: global mapping algorithm, local mapping algorithm, frequency-domain mapping algorithm, gradient mapping algorithm.

1) Global mapping algorithm: it uses the uniform mapping curve f(x). This mapping curve suitable for all pixels in an image. It compresses the dynamic range into the required range. Global mapping algorithm is simple, fast, but sometimes not ideal effect.

2) Local mapping algorithm: it based on different areas of pixels use different mapping curve. Scholars have proposed many kind of this algorithm. Local mapping algorithm can often get better results than global mapping.

3) Frequency-domain mapping algorithm: it convert image information into frequency domain, and then compress the dynamic range.

4) Gradient mapping algorithm: it compress the range on the gradient domain

Various mapping algorithm in reserve a HDR image information on, can show different visual effect LDR image. But the tone mapping roots show and purpose is closest to the eye image of high quality image.

3 Algorithms

3.1 Overview of the Program

This project has been developed using the C++ environment in Microsoft Visual C++ 2008 Express Edition and OpenCV(Open Source Computer Vision Library) version 2.1. In this project, Opencv and C++ have been used to implement all of the different alignment algorithms.

Our application is divided into three parts: MTB alignment algorithm and improvement, SIFT alignment algorithm and improvement, and HDR image Composition Tone-mapping.

Each part has its own function and is independent of the others. This means that each part carries out its own work which is not related to the other part.

3.2 MTB algorithm:

In our application we defined an MTB class for the MTB algorithm. The main structure of this class is shown below:

Class	**MTB.cpp/MTB.h**

Methods	void GetExpShift(images& refImage, images& shiftedImage, int shift_bits, int shift_ret[2]);
	void GetExpRotate(images& refImage, images& shiftedImage,double rotate_bit,double *rotate_angle);
	void ImageShift(images& Input, int x_shift, int y_shift, images& Output);
	void ImageRotate(images& Input,images& input_2,images& Output,double angle);
	
	void ImageShrink(images& img, images& img_ret);
	void ComputeImage(images& Input, images& MTB_image, images& exclusion_image);
	void ImageXOR(images& Input1,images& Input2, images& Output);
	void ImageAND(images& Input1,images& Input2, images& Output);
	int ImageTotal(images& Input);
	void ImageRotateXOR(images& Input1,images& Input2, images&

```
Output);
int ImageRotateTotal(images& Input);
void Aligment_horizontal(images& input,int shift_ret[2],int num);
void Aligment_rotation(images& input_1,images& input_2,double
rotate_angle,int num);
void Aligment(images* img_set,int num);
```

Table 3.1 MTB algorithm's main class.

3.2.1 The Median Threshold Bitmap

To correct the photo image of the shift, the most direct method is to compare the edges of the objects in the image. However the general edge detection algorithm is very sensitive to exposure photos. The same edge of an object has different levels of exposure between larger photos sequences may result in a very large difference,

Hence in our application, we used MTB algorithm for the alignment.

Firstly, we needed to change the colour image into a gray image:

For any point in the scene, there is a fixed ratio of the points which are brighter and darker than this point. Hence, for any fixed point in two different exposure images, there is also a fixed ratio between the bright and dark point. Therefore, if the image is gray, we can calculate its histogram. We divided the number of all the pixels into equal or approximately equal parts based on the size of pixel value. This calculates the boundary for the value *M*. Hence the binarization of the image is:

$$v_I = \begin{cases} 0 \, v < M \\ I, v \geq m \end{cases}$$

V_1 represents the binary after the pixels and *V* represents the pixels in the original image. If the image is in colour, we need to convert it to grayscale again in binary value, or directly use a channel of weight as the gray value and binary value. The results of the edge detection are hugely different between two different exposure images, but the binarization results are basically the same, indicating that the value of the binary can be applied to different exposure of the image registration.

But if the image is loaded with noise, it will lead to errors in the next step; therefore the noise should be reduced. Errors which are introduced by noise are generated in the vicinity of the value. For example, the value of pixel that is supposed to be slightly less than (or greater than or equal to) the *M* value should have been assigned to 0 (or 1) in the binary. Because of the noise interference, the pixels are expressed in values slightly greater than or equal to (or less than) the pixel value. Hence, it has been wrongly assigned to 1 (or 0) in the binary. On the other hand, pixels which are further away from the *M* pixel have smaller noise. Therefore, we can effectively reduce the error by only removing the noise around the *M* pixel. We can only change the pixel values in the range of around ± 4 pixel values 0 (black), other pixels unchanged.

$$V_2 = \begin{cases} 0 & |V - M| \leq 4 \\ V_1, & |V - M| > 4 \end{cases}$$

In this calculation, V_2 represents the binary image pixels which have had the noise reduced, V_1 represents the original binary image in pixels, *V* represents the original image in pixels, *M* represent the median value of noise removal, near to the accuracy of the results.

Hence in our application, we have used methods to calculate the threshold bitmap and exclusion bitmap:

```
void ComputeImage( images& Input, images& MTB_image, images& exclusion_image);
```

Figure 3.1 below shows how we used two different exposure images to get the MTB bitmap and exclusion bitmap. The exposure time of Image 1 is 1/125 s. For the Image 2 exposure time is 1/8 s.

Image1

MTB bitmap1

Exclusion bitmap 1

Image2

MTB bitmap2

Exclusion bitmap 2

Figure 3.1 MTB bitmap and exclusion bitmap

3.2. 2 Translational Alignment

After the binary image, the boundary contours are clearly demonstrated. We can get the best matching position between two different exposed images when we operate the binary image and constantly shift to the images and compare with the reference image.

We wrote a method for achieving shift offset in our program.

```
void GetExpShift(images& refImage, images& shiftedImage, int shift_bits, int shift_ret[2]);
```

However, for high-resolution images, the direct translation using the above method was more likely to take a lot of time, For example, if we assume that two non-point shots of the image relative offset (20, 15) (unit pixel), then even if we know the relative offset direction, at least to the second, by moving images and comparing the value of $20 \times 15 = 300$ times, taking into account the image resolution, the computation will be quite enormous. This shows that this method cannot be successfully applied to high-resolution, multiple images, particularly in real-time rendering systems, as rendering time is difficult to meet the system requirements. To reduce the computational time and improve processing speed, we can take this following strategy:

1) We can use the multi-scale ideas which make an image pyramid sequence for two binary images. The pyramid shows the image at each level with the image height and width reduced by half, each image is reduced by a quarter of the size of the previous image in a pyramid series that is decided by the height and width of the small value of the original image, requested the end of the image have a certain level differentiable

   In our program, we divided the image into a four level of pyramid which reduces the image by one quarter the size of the original image (see Figure 3.2)

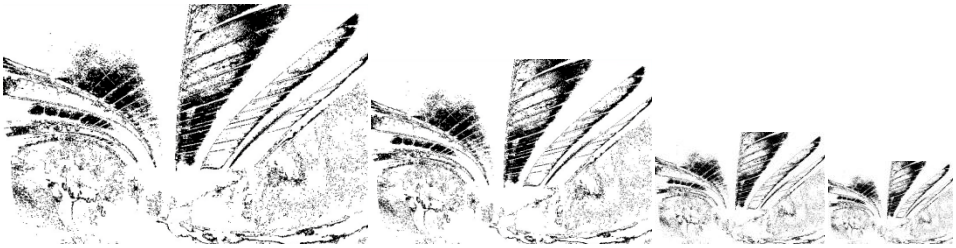


Figure 3.2 Four-level pyramid for the medium threshold bitmap

After we obtained the image pyramid, started to compare the last stage of the image.

We registered the image which moves horizontally and vertically in the range of ± 1 pixel (including the alignment: up, down, left, right, upper left, lower left, upper right and the lower right position), Next, we carried out XOR operation on the image. If the corresponding pixel was inconsistent (value of 1 after XOR operation), we can assumed that there is a difference. All the differences were added (to the sum of 1) and when we got the location of the minimum of sum number, we assigned it to the best image registration position. Then we returned the offset (a, b). When we went back to a higher level of the image, we used the same method to find the best matching position within the range of (2 a ±1, 2 b ± 1) pixels. This step was repeated until we returned to the top of the image pyramid and calculated the final offset.

This method is simple and reduces the number of shifts and comparisons. For example, for the registration of two 1024 ×1024 resolution images, the image size of the last stage was 32 × 32, the pyramid of the series for the 6 level, was moved and compared 6 × 9 = 54 times, and the image resolution was reduced in turn, which offset the maximum registration of 2^6 - 1 = 63 pixels.

2) This is because the integer operations are a lot faster than normal operations. We can divide the binary image data into group of 32, and convert it into 32 bits of integers. This will largely improve operations, or translation. In addition, we can operate 32 pixels at the same time.

3.2.3 Improvement of MTB Algorithm for Rotational Alignment

The original MTB algorithm does not support rotational alignment because the MTB algorithm is specially designed for the ordinary digital camera. However, when we use this camera to capture an image with different exposures, it will produce some rotational errors. If this rotational offset is not adjusted before the composition, then the final HDR image will be blurred. Hence, I improve the MTB algorithm to support the rotational alignment.

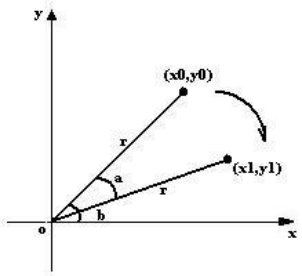
In our program, we extended the MTB algorithm class adding some functions to handle rotational alignment .The main structure of this algorithm is shown in the table below:

Function	Purpose
void GetExpRotate(images& refImage, images& shiftedImage,double rotate_bit,double *rotate_angle);	Get the rotational angle for the alignment
void ImageRotate(images& Input,images& input_2,images& Output,double angle);	Rotate an angle for the image
void Aligment_rotation(images& input_1,images& input_2,double rotate_angle,int num);	Rotational alignment for two images

Table 3.1 The main function for the rotational alignment

3.2.3.1 Image Rotation

The formula for a coordinate to rotate clockwise through an angle to a new point is shown in the diagram below:



$$x_1 = r\cos(b-a) = r\cos b\cos a + r\sin b\sin a = x_0\cos a + y_0\sin a;$$
$$y_1 = r\sin(b-a) = r\sin b\cos a - r\cos b\sin a = -x_0\sin a + y_0\cos a;$$

We define $P_0(x_0, y_0)$ as the rotation of an angle to a new point P(x1, y1) where r is the distance between the end point and the original point and b is the included angle for the r and x-axis.

It is better to do the matrix computation using OPENCV. Hence the rotational formula can indicate the matrix computation:

$$[x_1 \quad y_1 \quad 1] = [x_0 \quad y_0 \quad 1]\begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

In our program we designed a function to carry out the image rotational. It is based on the OPENCV cMAT operation

```
void ImageRotate(images& Input,images& input_2,images& Output,double angle);
```

Generally speaking the centre of the rotational image is the center of this image. This means that we need to rotate all the pixels in the image at the same angle. (Figure 3.3)

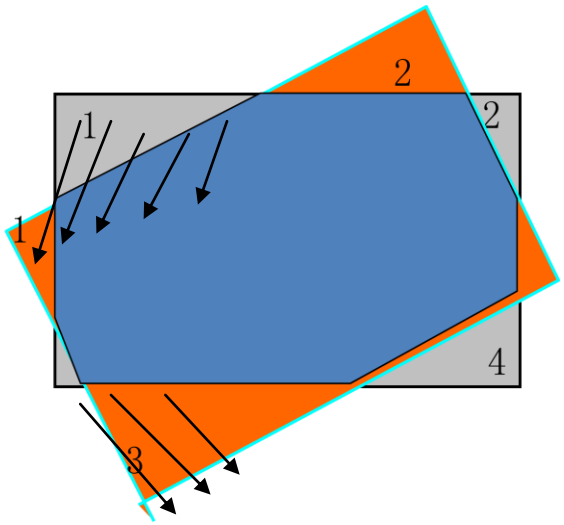


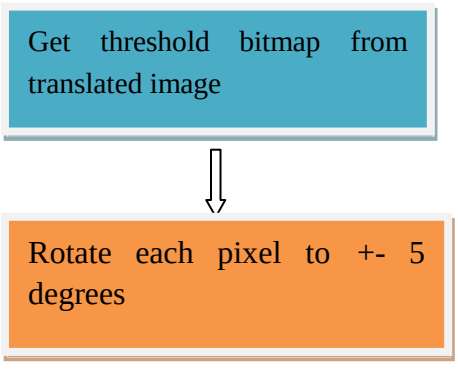
Figure 3.3 Rotating pixels in the program

3.2.3.2 Process of Rotational Alignment

We designed a function to calculate the rotational offset between two images. Here is the definition in our program:

```
void GetExpRotate(images& refImage, images& shiftedImage,double rotate_bit,double *rotate_angle);
```

Figures 3.4 shows the main structure for the rotational alignment.



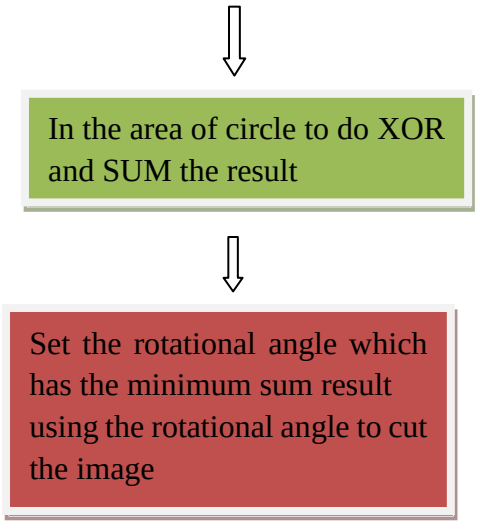


Figure3.4 The main process of Rotational alignment

As I mentioned previously, we carried out the translational alignment to get the translational offset. After that, we needed to carry out the rational alignment. Firstly, we needed to use the threshold bitmap which we got from the process of translational alignment. From this, we got a new pair of images. Moreover we needed to set the rotational center in the middle of these new images. After that, we needed to define the scope of rotation. For each rotation of an image, we used the XOR operation and sum of the result with the base image which simulates to the translation alignment. However, this time, the XOR operation did not calculate the entire pixels in the image. We just needed to use the rotational center as the center of a circle. The smaller of length or width are defined as the diameter of the circle. The entire XOR operation is calculated in this area of the circle.(Figure 3.4)

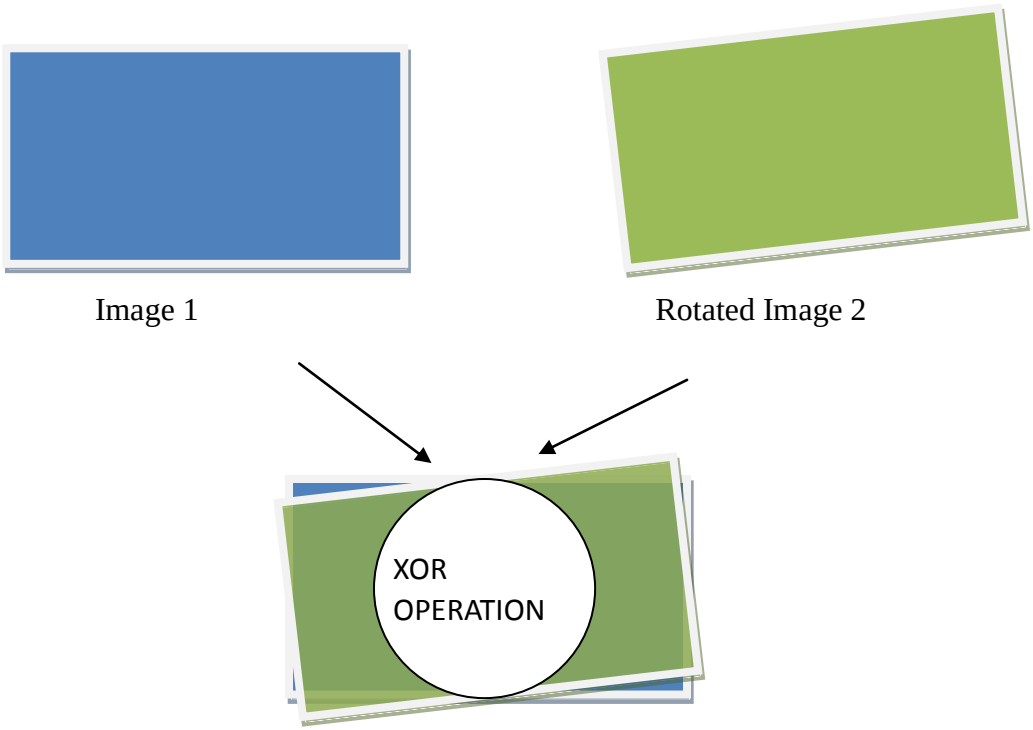


Figure 3.5 XOR operations in rotational alignment

After that, we need to find the minimum value for the sum operation which is the best location of the rotational angle. Once we calculated the rotational angle, we used it to cut the image and retain its public part.

Translation offset and angle offset will have, to some extent, coupling between the rotations.

If we rotate by an angle which is too large, the result may be shift registration which is not accurate. Hence, we need to alternate the rotation and translation.

Translation → rotation → re-translation → rotation →…….

This is performed until the offset angle reaches 0. As the translation of the registration results may not be accurate, the image should not be cut in the translation of offset, when we get the value of offset translation. However we also needed to cut the image for the rotation of offset

3.3 SIFT Algorithm

3.3.1 Overview of SIFT Algorithm

SIFT (Scale Invariant Feature Transform) was proposed by David G.Lowe[35] in 2004. It is based on scale space, scaling, keeping invariant feature for rotating and affine transformation

SIFT algorithms can be divided into four main steps:
(1) Construct scale space
(2) Locate feature point
(3) Determine the direction feature points
(4) Generate SIFT feature vector

According to these steps, our program was constructed to perform a different function. The main function for the SIFT algorithm is shown on the table below:

Class	**Sift.h/sift.cpp**

Function	int _sift_features(IplImage* img,CvSeq* feat, int intvls,double sigma, double contr_thr, int curv_thr,int img_dbl, int descr_width, int descr_hist_bins);
	
	IplImage* create_init_img(IplImage* img, int img_dbl, double sigma);
	
	IplImage*** build_gauss_pyr(IplImage* base, int octvs,int intvls, double sigma);
	
	IplImage*** build_dog_pyr(IplImage*** gauss_pyr, int octvs, int intvls);
	
	CvSeq* scale_space_extrema(IplImage*** dog_pyr, int octvs, int intvls,double contr_thr, int curv_thr,CvMemStorage* storage);
	
	void calc_feature_scales(CvSeq* features, double sigma, int intvls);
	
	void calc_feature_oris(CvSeq* features, IplImage*** gauss_pyr);
	
	void compute_descriptors(CvSeq* features, IplImage*** gauss_pyr, int d, int n);

Table 3.2 construction of SIFT algorithm

3.3.2 Construction the SIFT algorithm

3.3.2.1 Construction of Scale Space

The conception of scale space first appeared in the computer vision field, and was used to simulate scale features of various image data. Koendetink[19] uses the diffusion equation to describe the process of scale space, proving that Gaussian kernel is the uniform transformational kernel in the process of scale space. Lindebrg[20] and Babaud[21] provided further proof that Gaussian kernel is the uniform linear kernel. Therefore, the main idea of the theory of scale space is: transforming the original image through the Gaussian kernel. It can calculate the sequence of scale space for the images in the multi-scale. And then we can use this sequence to extract the feature points in the scale space. The two-dimensional Gaussian kernel is expressed as:

$$G(x, y.\sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$$

In the 2D image I (x, y), l. (x, y, σ) (respect different scales of scale space) can get through calculate the convolution of Gaussian kernel G (x, y, σ) and image I(x, y):

$$L(x, y, \sigma) = G(x, y, \sigma) \times I(x, y)$$

In the formula above, L indicate that scale space. (x, y) is the point in the image and σ is the factor for smaller scales which controls the smoothness of an image. The smaller the value of σ is, the less smooth the image will be.

In our program, we use the Gaussian convolution to initialise an image which has been integrated into a function:

```
IplImage* create_init_img( IplImage* img, int img_dbl, double sigma );
```

Here is a test image which exhibits Gaussian blur (Figure 3.6):

Figure 3.6 Image of Gaussian blur

After that, the main task was to establish the Gaussian pyramid and DOG pyramid (Difference of Gaussian). We needed to detect the extreme value point in the DOG pyramid and get the position of feature point.

We designed two main functions to build the map of the Gaussian pyramid and DOG pyramid

IplImage* build_gauss_pyr(IplImage* base, int octvs,int intvls, double sigma);**
IplImage* build_dog_pyr(IplImage*** gauss_pyr, int octvs, int intvls);**

(1) Construction of the Gaussian pyramid

In order to obtain the feature points in different scale space, we need to calculate the convolution of Gaussian kernel G (x, y, σ) in a different scale factor. From this, we can constitute the Gaussian pyramid. Each Gaussian pyramid have n order. However, we first chose an order of four to experiment with. In each order we have S layers of image (Normally we just get five levels of layers). Finally we achieved the Gaussian pyramid as shown in Figure 3.7

(2) The DOG pyramid

DOG means that we need to subtract two adjacent scale space functions. If we use the function D (x, y, σ) to indicate these, the formula show below:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \times I(x, y)$$
$$= L(x, y, k\sigma) - L(x, y, \sigma)$$

By subtracting two adjacent scale spaces of the Gaussian pyramid .we can calculate the DOG pyramid. Each order of the Gaussian pyramid and the DOG pyramid has the same scale factor.

We can see this in the figure below

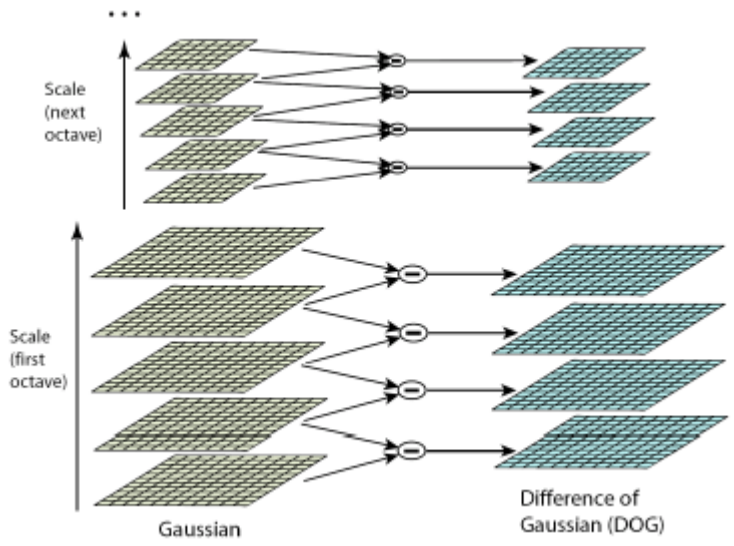


Figure 3.7 Difference of Gaussian (DOG)[22]

(3) Detection of feature points in DOG space

After the establishment of the DOG pyramid scale space, the point in the middle layer needs to be compared with the other point in the same layer. In addition, it also needs to be compared to all 9 points in the upper and lower layers. Finally we found the local maximum and local minimum in the DOG space. This means that we need to find the feature point which is the maximum point or minimum point in the scale space, recorded as a valid feature point.

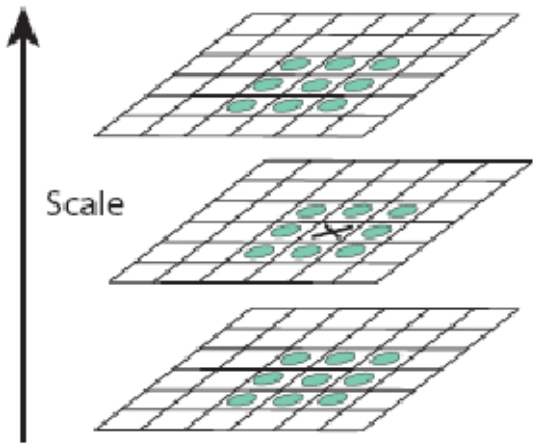


Figure 3.8 Detection of feature points [22]

3.3.2.2 Location of the Position of Feature Point:

Feature points were found in the process of detecting the DOG scale space, which is not stable enough. Hence we needed to remove some of the instability of the feature points. In this process we needed to remove the low comparison point and edge point.

Hence, we designed two main functions to select these feature points

```
void calc_feature_scales( CvSeq* features, double sigma, int intvls );
void calc_feature_oris( CvSeq* features, IplImage*** gauss_pyr );
```

To estimate the low comparison point, we can use the formula below:

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

After that, according to the *x* and **D**, we need to calculate the offset $\hat{x}$ using the formula below:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial x}$$

In order to strengthen and improve the stability of the matching anti-noise ability, and to remove low-contrast, unstable edge points and features points, we used this formula

$$D(X_{\max}) = D + \frac{1}{2} \frac{\partial D^T}{\partial X}$$

Calculate *D (X max)*, if the *D (X)* >= 0.03. We needed to keep this feature point
After that, we needed to remove the unstable edge response, we using the following formula:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

H represents Hessian matrices.:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

We needed to calculate *Tr* (trace) and *Det* (determinant). α is the maximum range feature and β is the rang lower than α. [18]

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

Finally we defined the *R* as 10, and calculate the formula. If $\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$,we keep this feature point, otherwise we remove it.

3.3.2.3 Determination of the Direction of feature points

Because the neighborhood pixels' gradient distribution is different for each feature point, through calculation, operator direction parameters have better rotation invariants.

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1), L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}(L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y))$$

M(x,y) indicate gradient strength in (x,y) and θ(x,y) is the direction of this pixel (x,y). Here we used orientation histogram to determine the direction of the feature points. Firstly, we calculated the position of all the pixels within the size of the surrounding

block. Besides that we defined weight of each pixel. If the pixel is more closes to the feature point, it has a larger weight. Finally, we calculated the total weight, and gave the direction of the largest weight to the feature point.

3.3.2.4 Description of feature points

After we get the direction of the feature points, we needed to define how to describe them. Firstly, we needed to rotate the block of the feature point to fix the direction of the feature point. Secondly, we divided the block into to 4x4 sub-blocks. Finally, we counted the direction for each sub-block.

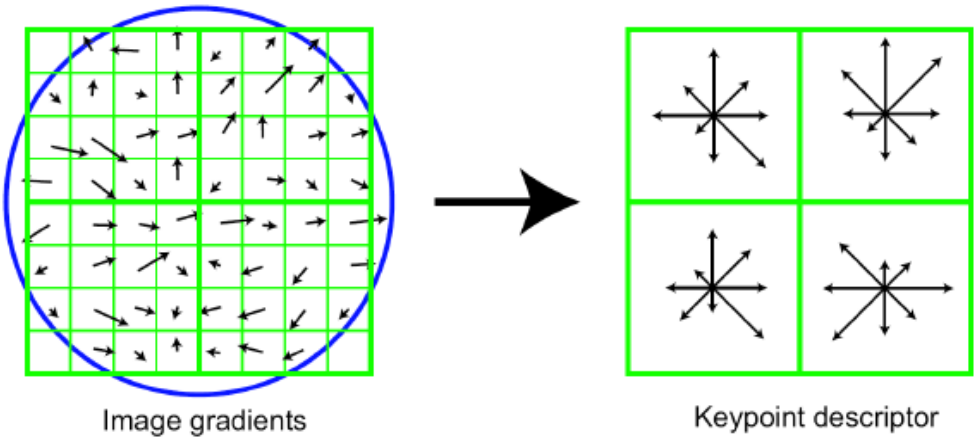
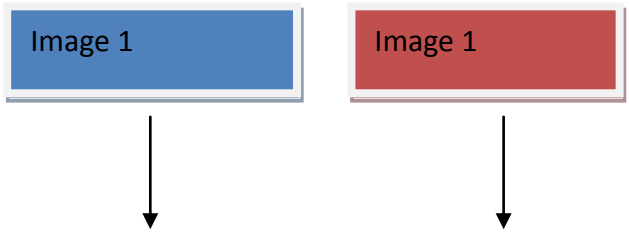
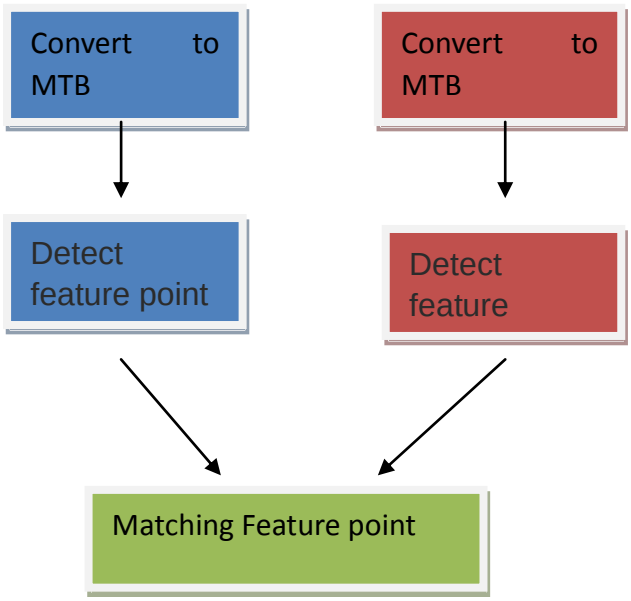


Figure 3.9 The process of describing the feature points [22]

3.3.3 Improvement of the SIFT Algorithm

The original SIFT algorithm cannot locate the feature points, when the images have been taken with hugely different exposure times. Hence, we use the MTB algorithm to handle the image before locating the feature points.





However, when we tested this new MTB+SIFT algorithm, we found that it was not stable when used with a different range of exposures. It can get good results for the conditions of over-exposed or under-exposed images. But it will lose the feature points in the normal exposed image. Hence, we had to find a new solution. Firstly we used the ordinary SIFT to locate the feature point. Next, we used the improved MTB +SIFT algorithm to detect the feature points again. Finally, we merged the feature points which we identified form these two algorithms, and deleted any repeated points.

4 Experiment

4.1 MTB Algorithm

For the MTB algorithm, our experiment can be separated into two parts. Firstly, we tested two images with different exposures time for horizontal alignment. As you can see in the Figure 4.1 below, we took two images to test horizontal alignment. The horizontal offset for this group is x =2 and y = 1. After we aligned the images, we found that the aligned image is close to Image A with regard to horizontal alignment.

Image A

Image B

Aligned Image

Figure 4.1 Horizontal alignment

Secondly, we used the improved MTB algorithm to test two images for rotational alignment. As you can see in Figure 4.2, Image B has an obvious rotational angle. The Table 4.1 below shows the test data for rotational alignment. We analyzed the different points in these two images and we found the minimum different points when the image was rotated 5 degrees. This means that there is a great similarity when the Image B

rotates 5 degree. After rotational alignment, the images showed a great improvement in rotational accuracy.

Image A image B

Aligned Image

Figure 4.2 Rotational alignment

Rotational Angle	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**
Amount of Different Point	28451	31089	33581	36189	38936	36087	45352	48485	51350	54150	56818

Table 4.1 The analysis of different points

After we tested the feasibility of horizontal alignment, we needed to carry out some experiments on the composition of HDR image. As you can see in Figure 4.3, there are sequences of images which have not been aligned. You can also see the final result in the compositional HDR image which has apparent overlap.

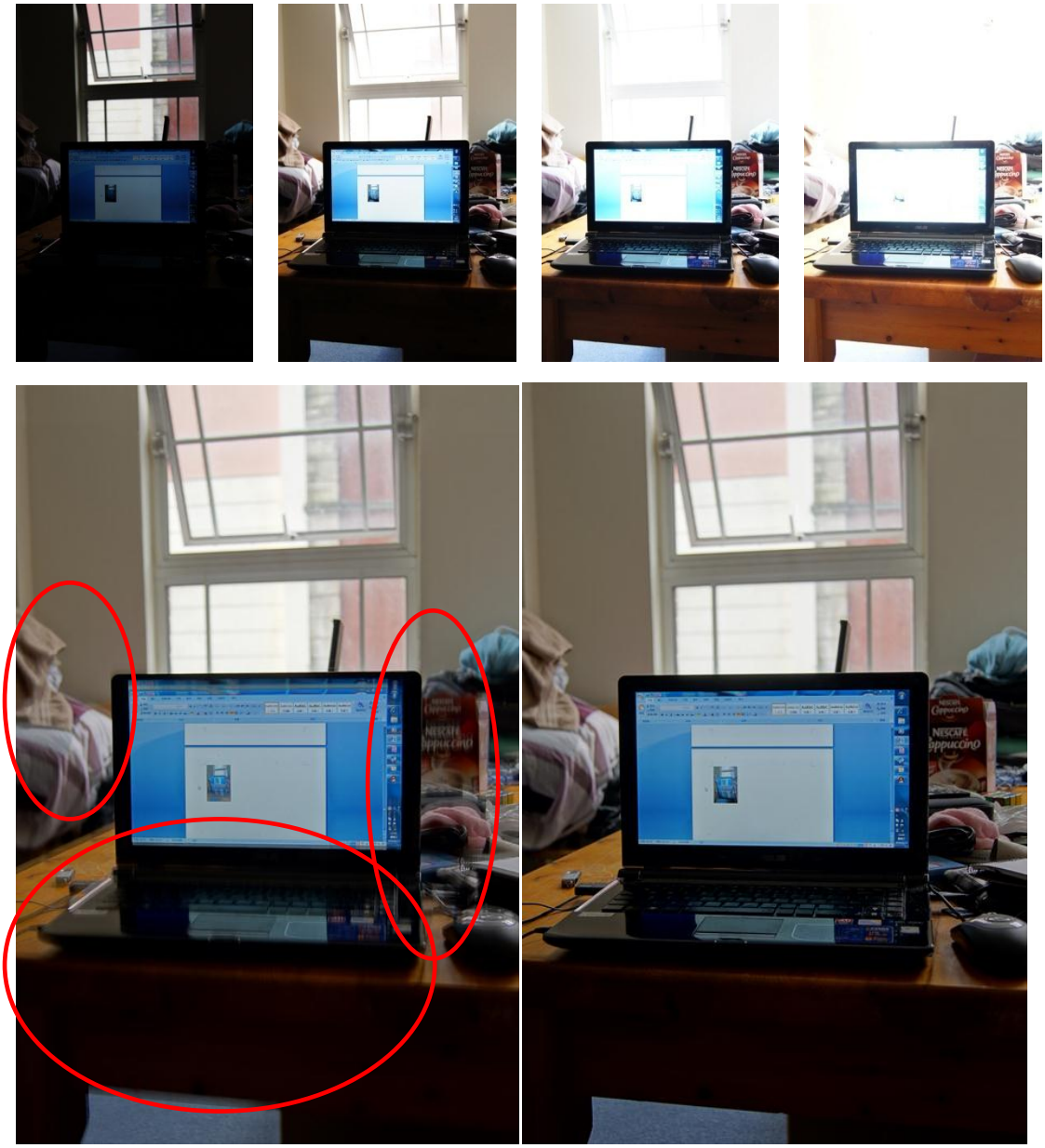
Figure 4.3 Composition of HDR image from the upper sequence of the image

Next, we used the normal exposure image for the standard image and got a sequence for the aligned image. (See Figure 4.4) .Finally, we observed from the HDR image that overlap was eliminated.

Image A horizontal offset (8,3) rotational angle(1) Image B horizontal offset (-31 -6) rotational angle 0

Figure 4.4 Composition of HDR image from the aligned image.

In addition, we tried another sequence of exposed image which results in a better effect in the HDR image. (See Figure 4.5). In this situation, we tried to use the original MTB algorithm and the improved MTB algorithm for the rotational alignment. The image on the left in Figure 4.6 shows the horizontal alignment for the MTB algorithm. We can found that fuzzy pixels in the HDR image which had not been aligned. The red circle indicates the position of fuzzy pixels. This means that the rotational offset affects the final result of HDR image. The image on the right shows the efficacy of using the rotational an horizontal alignment which improved the definition of the compositional image.



NOT ALIGNED HDR IMAGE ALIGNED HDR IMAGE

Figure 4.5 Comparison of the aligned HDR image.

To draw a conclusion on the improved of MTB algorithm, we noted it can increase the definition of the compositional HDR image which reduces the fuzzy image. In addition, there are improvements in the accuracy of the alignment when compared to the image that uses horizontal alignment.

4.2 SIFT Algorithm

Firstly we tested the SIFT algorithm for a sequence of images taken with different exposure times:

As shown in Figure 4.6 below. The exposure time for the image is 1/25s and the image is under-exposed.

Figure 4.6

We tested it using the original SIFT algorithm and MTB + SIFT algorithm to detect the feature points. Figure 4.7 used the original SIFT algorithm. As you can see from the image, it only identified 18 feature points in the under-exposed environment. This means that in the under-exposed images, the SIFT algorithm can do well in detect feature point, but it loses picture information in this environment.

Figure 4.7 SIFT algorithm (18 feature points)

Then, we used the MTB algorithm (Figure 4.8) to detect the feature points in the SIFT algorithm. There were huge improvements in the number of detected feature points, the number increased to 428 point. (Figure 4.9)

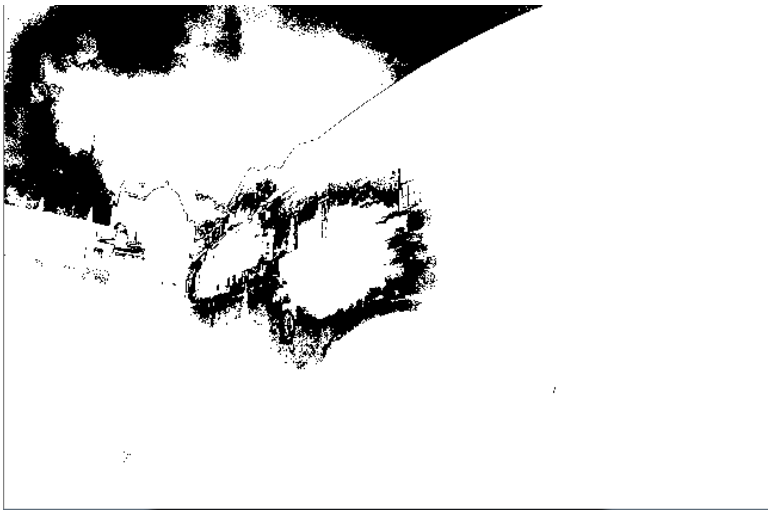


Figure 4.8 Median Threshold Bitmap

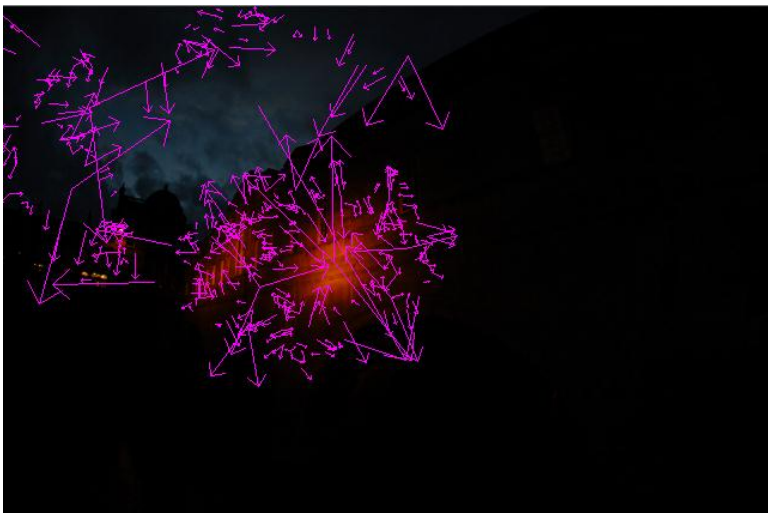


Figure 4.9 MTB + SIFT algorithm (428 feature points)

We also tested an image which is over-exposed. Figure 4.10 below shows an image which has 1/8s exposure time. Image (A) was tested using the normal SIFT algorithm to get the feature points and 1045 points were identified. We used the MTB + SIFT algorithm to detect the feature points in Image (B) and the number increased to 1232 feature points.

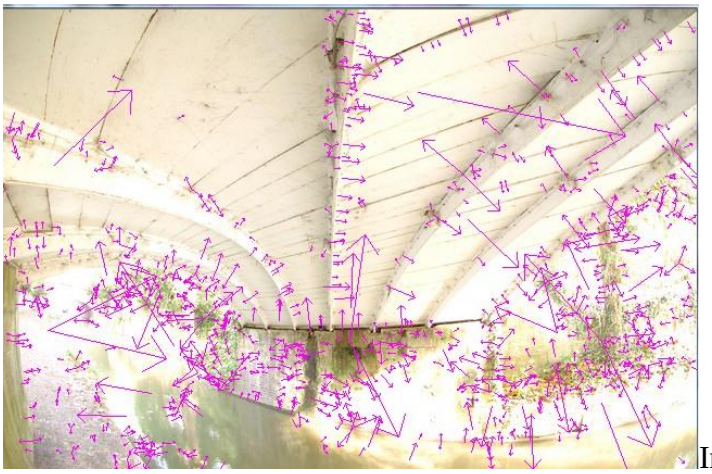


Image (A)

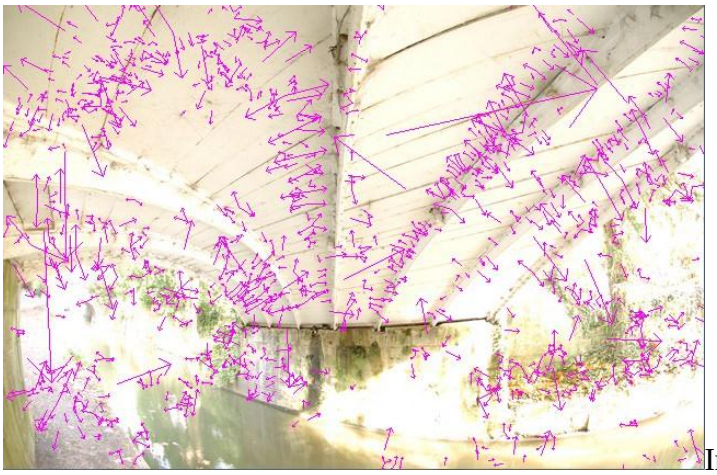


Image (B)

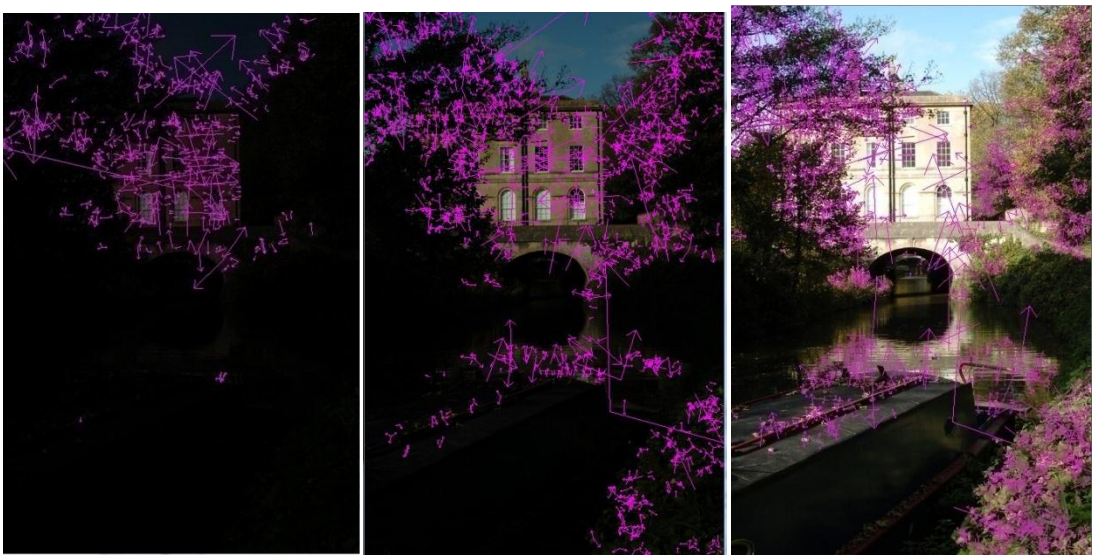
Figure 4.10 The tested for Overexposed images

Hence the experimental results show that, we can use the MTB algorithm to handle the image before using the SIFT algorithm to detect the feature point. It is more stable when used on images which are overexposed and underexposed. It is more suitable for aligning the images in the process of forming a combinative HDR image.

Next we tested these two algorithms on images with longer exposure time. In order to find out whether the MTB+SIFT algorithm is suitable for all conditions.

Figure 4.11 shows a sequence of an image that was exposed for different lengths of time. Group 1 show the images that were subjected to the SIFT algorithm. Group 2 shows the images that were subjected to the MTB + SIFT algorithm. The result is shown in the Table 4.2

Group(1):Image a: 1/2000s (77 feature point) Image b: 1/1000(865 feature point) Image c 1/200s(2322 feature point)



Group(2) :Image a: 1/2000s (662 feature point) Image b: 1/1000(1862 feature point) Image c 1/200s(2075 feature point)

Figure 4.11 Different exposure image sequences subjected to SIFT and MTB+SIFT algorithm

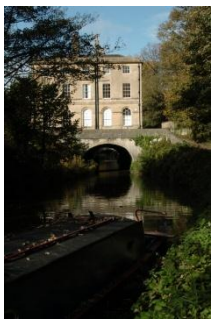
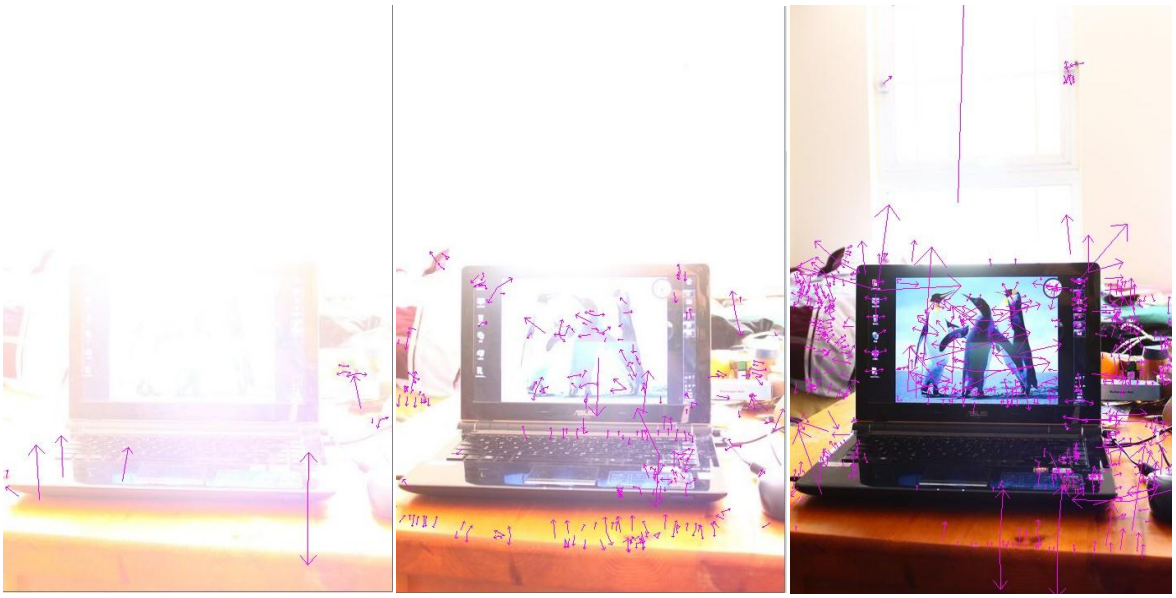
			
SIFT	**77**	**865**	**2322**
MTB+SIFT	**662**	**1862**	**2075**

Table 4.2 Detected feature point in the different exposure image

From the results of upper table, we made a new discovery. The MTB + SIFT algorithm is not stable for different exposures. When the image approaches normal exposure, the detected precision will be reduced. Hence we carried out experiments on a wider range of exposed image.(See Figure 4.12)



Group A Image a 1.3s (17) Image b 1/2 (291) Image c 1/6s (737)

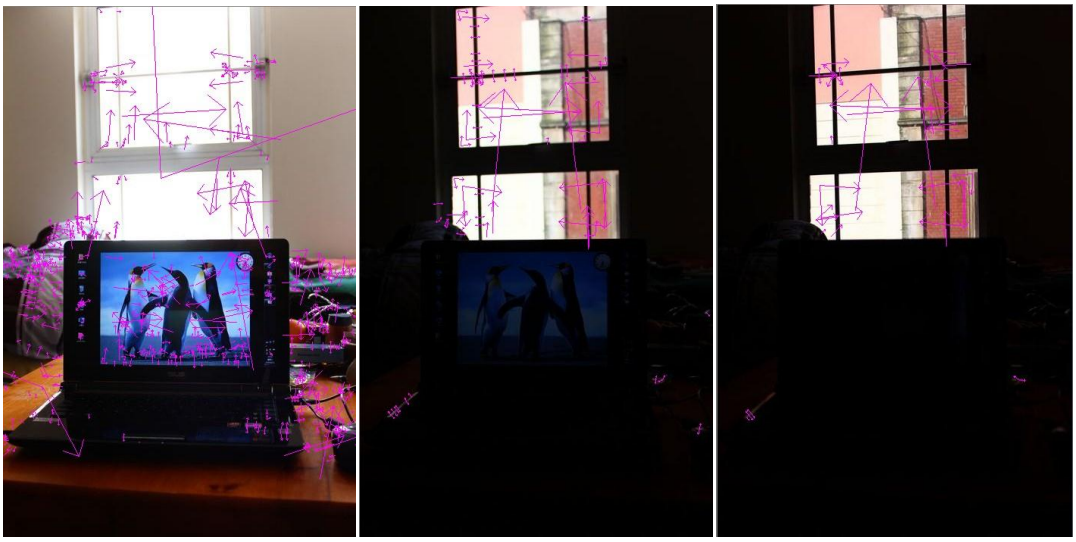


Image d 1/60s （746） Image e 1/1600(117) Image f 1/4000(48)

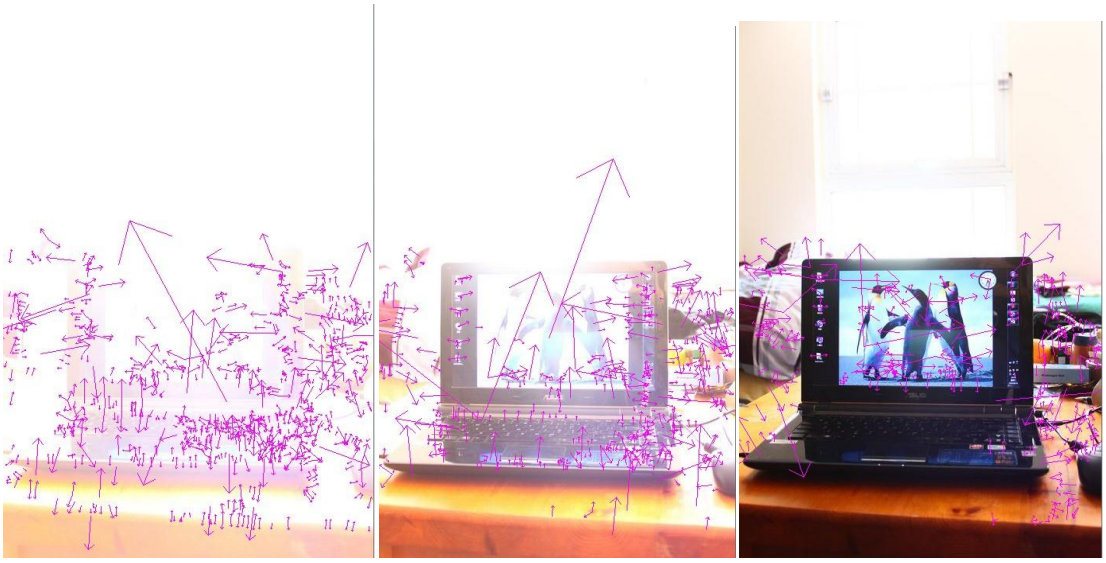


Image a (946) Image b(544) Image c(405)

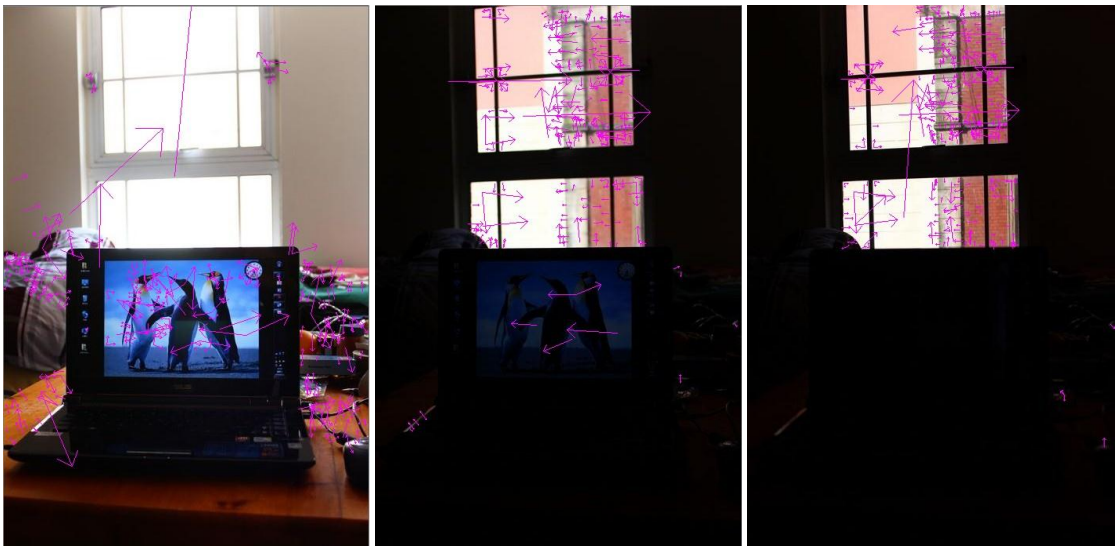


Image d(354) Image e(276) Image f(325)

Figure 4.12 Widely range of exposure image for two algorithms

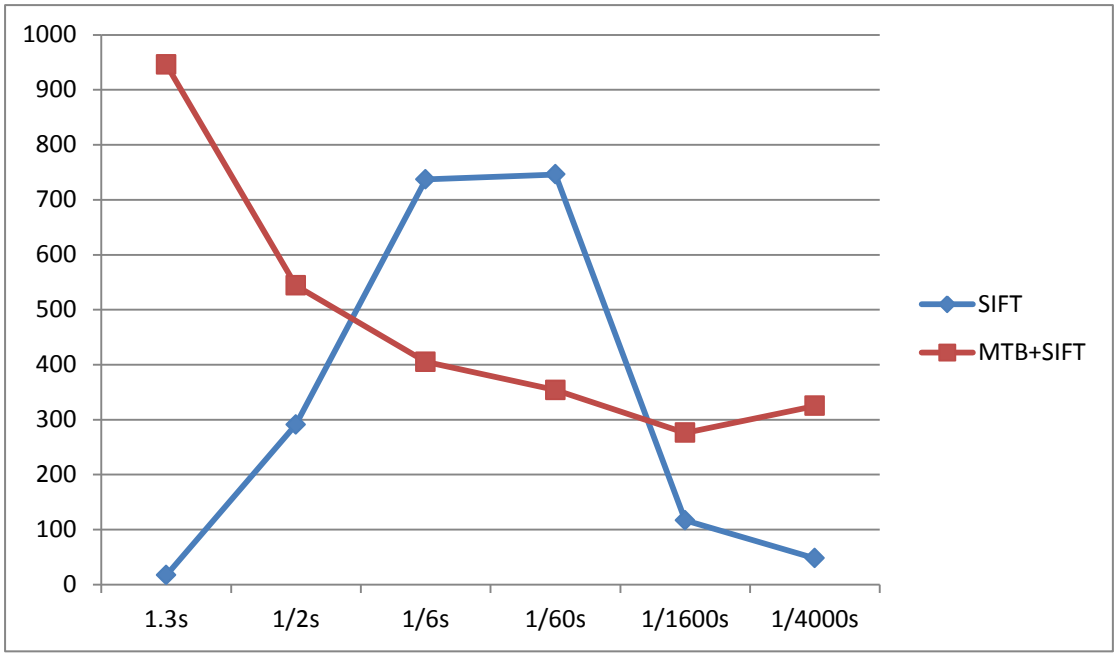


Table 4.3 Detected feature point for SIFT and MTB+SIFT algorithm

As Table 4.3 shows the MTB+SIFT algorithm is more suitable for conditions of over-exposure and under-exposure. There is a trend whereby the accuracy is reduced where the image is closer to normal exposure. For this situation, we carried out the following analysis:

Here is the normal image which was taken using normal exposure. Image B is the MTB bitmap.

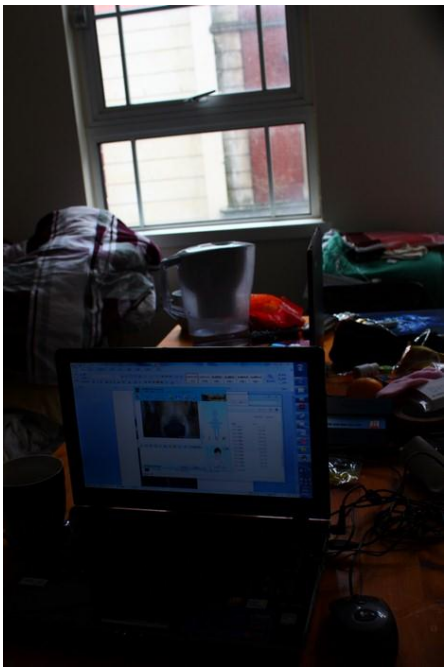
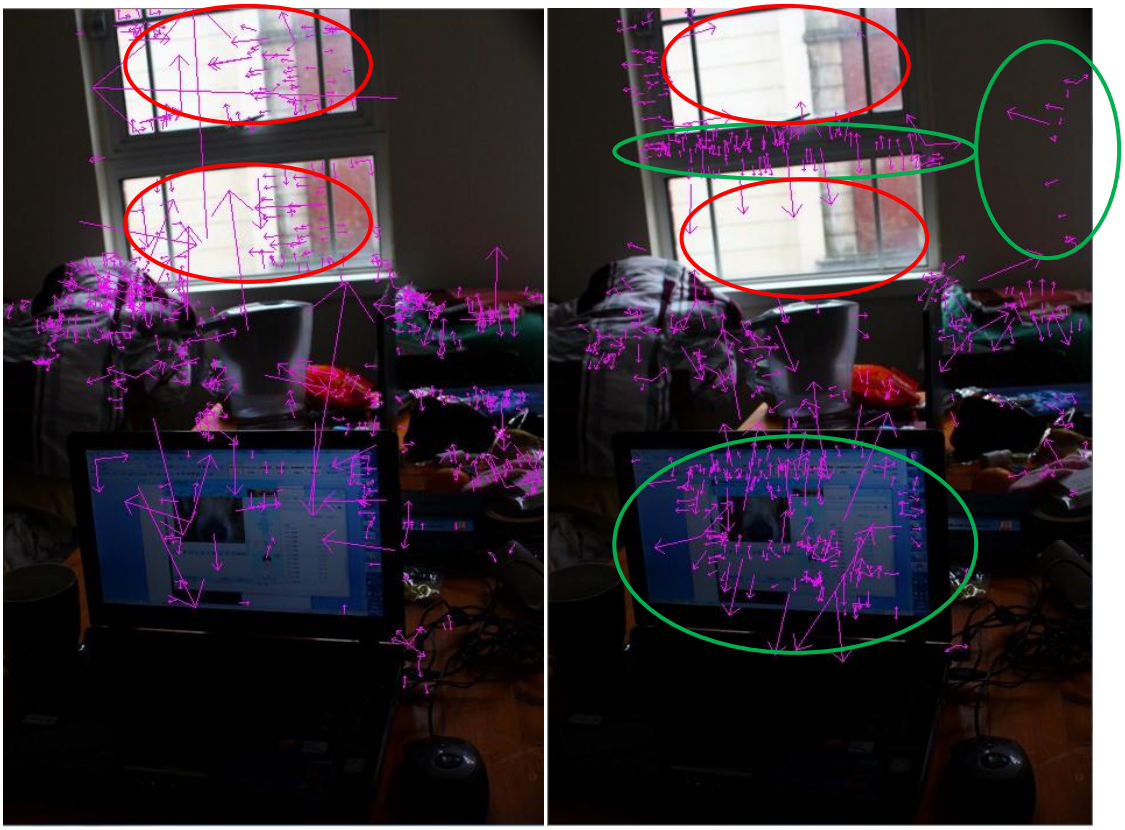


Image A

Image B

From the Image B, we can discern that about the MTB bitmap is more likely to describe the areas of darkness and brightness. In the area which is distinctive to comparative color aberration , the algorithm cannot make a good description. As you can see in the Figure 4.13, the red cycles show the areas of feature points which cannot be detected using the MTB+SIFT algorithm. And the green cycles show that there was an improvement in number of detected feature points when the MTB+SIFT algorithm was used. The green circles show the pixels which are located in areas of darkness or brightness. However when there is huge contrast, a MTB bitmap will lose information . Hence it cannot detect the feature points in the red circles..



SIFT (710 feature point) MTB+SIFT (678 feature point)

Figure 4.13 Analyze of the two algorithms

Therefore, for all condition of exposed image, we use these two algorithms together to detect the feature points and then deleted any repeated feature points. The results are shown in the Figure 4.14 below which shows that 1287 feature points were detected.

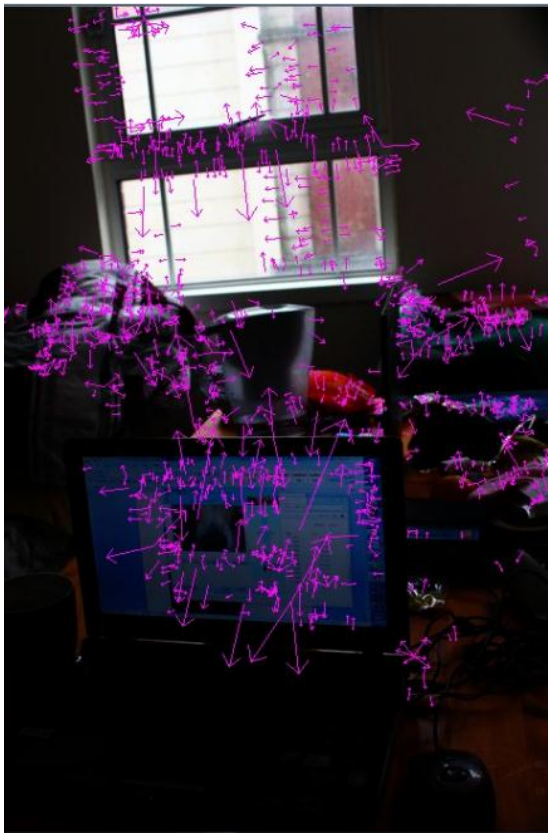


Figure 4.14 improved MTB+SIFT algorithm

5 Evaluation

Our project aimed to analyse two main HDR image alignment algorithms based on sequences of different exposure images. Furthermore, our project used OPENCV as an auxiliary tool to build these two algorithms in the environment of C++. Over the course of the project, we also made some improvements to these two algorithms to increase their accuracy in aligning different exposure images. A summary of these two algorithms will follow.

5.1 Evaluation of the MTB Algorithm

Figure 5.1 Compare of the final result

There are no strict standards when evaluating aligned algorithms. However, we can use the final result of the compositional HDR image to evaluate the efficiency of the algorithm. The image on the left in Figure 5.1 shows the compositional HDR image which did not use the MTB alignment algorithm. There is apparent pixel overlap. This means that the shaking that occurs in the process of capturing the image leads to the displacement of the image. After we used the MTB algorithm to align the image, the definition of the image greatly improved.

Moreover, in the process of testing the original MTB algorithm, we found that carrying out horizontal alignment of the different images did not display enough accuracy for HDR image composition. As we mention before, when we capture different exposure images using an popularization of digital camera, there will be some rotational offset in the image. This offset cannot be ignoring in the process of forming a composite HDR image. Hence we improved the MTB algorithm to support rotational alignment.

Through the analysis of the relevant data, we found that the accurate rate of alignment was greatly enhanced. The compositional HDR image is clearer than the image that used the original MTB algorithm.

However, the accuracy of rotational alignment also needed to be improved, because when the rotational angle is below to 1degree, the algorithm cannot detect rotational offset. Even if we improve the accuracy of rotation to detect the offset at every 0.1 degree; it will only increase the time of detection. (See Table 5.1) Therefore, we also need to improve the efficacy of this algorithm.

Resolution of Image	Time for horizontal Alignment	Time for rotational Alignment(range of 1 degree)	Time for rotational Alignment(range of 0.1 degree)
640*480	2s	4s	10s
800*600	5s	9s	22s
1024*768	10s	16s	35s

Table 5.1 the comparison of original MTB and improvement of MTB algorithm in the time of detection.

5.2 Evaluation of the SIFT Algorithm

In this project, we tried to build the SIFT algorithm in the C++ environment. After that, we try to test the efficacy of the SIFT algorithm on the different exposure images. This research into the SIFT algorithm showed that it did not perform well in the over-exposed and under-exposed environments. In such environments, the algorithm cannot detect the feature points in the aligned image. Therefore we needed to make some improvements to the SIFT algorithm in order to make it suitable for multi-exposure image.

From the research we carried out on the MTB algorithm, we found the medium threshold bitmap performed well when displaying the information in over-exposed and under-exposed environments. Hence we tried to use the MTB bitmap and SIFT algorithm in combination to detect the feature points.

However we found that the SIFT +MTB algorithm presented linear relation with the SIFT algorithm in the process of the experiment. As Figure 5.2 below shows, the improved algorithm can detect more feature points in conditions of over-exposure and under-exposure. But when the image is close to normal exposure, the accuracy of detection is slightly decreased. The original algorithm performs better on images with normal-exposure. Hence, we needed to use both of these algorithms together to handle the different-exposure images and select the feature points. The repeated feature points were deleted. Finally we used the selected feature points for the alignment.

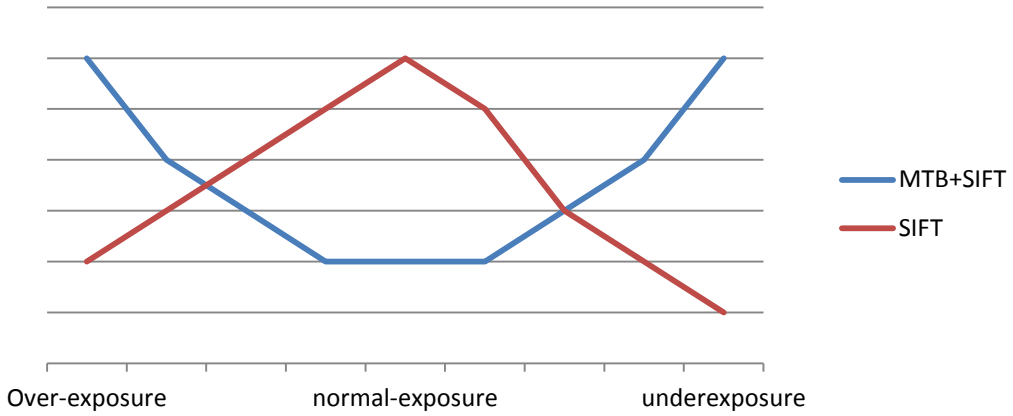


Figure 5.2 The trend of feature points detected with MTB+SIFT algorithm and the original SIFT algorithm in different exposure images

Finally, the methods used to improve the SIFT algorithm which used the MTB+SIFT and ordinary SIFT algorithm together to select the feature points hugely improved the detection of feature points in the different exposure images. However, this means that it took twice the time to detect feature points, thereby reducing the efficacy of the process of image alignment.

6 Conclusion and Feature Work

The main aim of our project was to explore suitable algorithms for the composition of a HDR image. From the research we carried out, we found two main HDR alignment algorithms: MTB and SIFT algorisms. Based on results of our tests on these two algorisms, we made some improvements to make them suitable for different exposure images. For the MTB algorithm, we improved it to support rotational alignment. After improving the rotational alignment, the quality of final compositional HDR image was greatly improved. This means that the MTB algorithm is suitable for the hand capture of different images .On the other hands, we analyzed the SIFT algorithm and found that it does not perform well in the conditions of limited exposure. When the images were captured in the conditions of over-exposure and under-exposure, the SIFT algorithm was unable to detect the feature points which meant that it cannot be used for HDR alignment. Basic on the research of MTB algorithm, we found that the medium threshold bitmap is stable when used on different exposure images. Hence, we added the MTB algorithm as an adjuvant process before the SIFT algorithm was applied. As a result, the accuracy of feature point detection was has a greatly improved.

When carrying out feature work on the HDR alignment, we hoped to increase the efficacy of alignment algorithm. For the MTB algorithm, we hoped to make some further modification to the select offset which increases the speed of the algorithm. On the other hand, the accuracy of feature point detection is more important in these algorithms. Hence, we need to carry out further experiments on the matching feature points. This will help the positioning of the image to be carried out more accurately and rapidly.

Bibliography

[1] Debevec P, Malik J. Recovering High Dynamic RangeRadiance Maps from Photographs[C]//Proceedings ofthe 24th annual conference on Computer graphics and interactive techniques(0-89791-896-7),1997

[2]P.Debevee.Image-basedlighting[J].IEEEComPuterGraPhiesandAPPlications，2002

[3] Craig L. Keast; Charles G. Sodini, CCD/CMOS process for integrated image acquisition and early vision signal processing

[4] The Science of Photography - Digital Image Processing, Ted's photographic , http://www.ted.photographer.org.uk/photoscience_digital.htm

[5] Reinhard E, Pattanaik S, Greg Ward, Debevec P. High Dynamic Range Imaging: Acquisition, Display, and Image-based Lighting[M]. San Francisco: Morgan Kaufmann Publishers,2005.

[6] Norman Koren, Making fine prints in your digital darkroom Tonal quality and dynamic range in digital cameras, http://www.normankoren.com/digital_tonality.html

[7] Photomatix,Create stunning photographs of high-contrast scenes http://www.hdrsoft.com/

[8] Jon Meyer,The Future of Digital Imaging - High Dynamic Range Photography, http://www.cybergrain.com/tech/hdr/ Feb 2004

[9] Greg Ward, High Dynamic Range Image Encodings ,Anyhere Software. http://www.anyhere.com/gward/hdrenc/hdr_encodings.html

[10] Mann，R.W.Pieard.Being'undigital，with digital cameras: Extending dynamie Range bt combining different exposure Pictures[C].Proceedings of Imaging Scienee and Technology 46th annual eonferenee，1995，422一428.

[11] M.A.Robertson，5.Borman，R.L.Stevenson.Dynamic range improvement Through multiple exposures[Cl.Proceedings of IEEE Intemational Conference on Image Processing，1999，

[12]P.Ledda， G.W.Larson， AChalmers.A wide field， high dynamic range, stereographic viewer[C]. Proceedings of the 1st international conference on Computer graphies an dinteractive techniques in Australasia and SouthEastAsia，2003.

[13] H.Seetzen，L.A.Whitehead , G.W.Larson. A HighDynamic Range Display Using Low and High Resolution Modulators[C].Proeeedings of the society for Information display Intemationa l SymPosium,2003

[14] N.Goodnight，R.Wang，C.Woolley ec al. Interactive time-dependent tone mapping using Programmable graphics hardware[C].Proceedings of 14th Eurographies Workshop on Rendering，2003

[15] Tomaszewska A, Mantiuk R. Image Registration for Multi-exposure High Dynamic Range Image Acquisition[C]//WSCG 2007, Full Papers Proceedings Iand II, 2007,

[16] Bay H. SURF: Speeded Up Robust Features[J]. ComputerVision and Image Understanding, San Diego: Academic Press Inc Elsevier Science ,2008

[17] M·Grossberg and Nayar Determining the Camera Response from Images: What Knowable?[J].IEEE Transaetionson Patten Analysis and Maehine Intelligence，2003.

[18]Faugeras O,Robert L.What can two images tell us about the third one[C].Proceedings of theEurope Conference on Computer Vision,Sweden,1994.

[19]Koenderink J.The structure of Images[J].Biological Cybenetics,1984,50:363-396.

[20]Lindeberg T.Scale-Space for discrete Signals[J].IEEE Transactions PAMI,1980, 207:187-217.

[21]Babaud J,Witkin A P,Baudin Metal.Uniqueness of Gaussian kernel for scale-space filtering[J].IEEE Transactions on Pattern Analysis and Machine Intelligence,1996, 8(1):26-33.

[22] D. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91-110, 2004.

[23] ZHANG W.X ZHOU B F An HDR brightness correction algorithm for panorama image Marbella, Spain 2004.

[24] M. Robertson, S. Borman, and R. Stevenson.Dynamic range improvement through multiple exposures. In *Proceedings of the 1999 International Conference on Image Processing (ICIP-99*, Los Alamitos, CA, October24–28 1999.

[25] Won-ho Cho, Ki-Sang Hong, “Extending Dynamic Range of Two Color Images under DifferentExposures,” *ICPR (4) 2004*, 2003.

Appendices: Source code

MTB.h

```
void MTB::ComputeImage( images& Input, images& MTB_image, images& exclusion_image)// create the MTB image
void ImageRotate(images& Input,images& input_2,images& Output,double angle);//rotate the image
void ImageRotateXOR(images& Input1,images& Input2, images& Output);// do XOR operation between two image
int ImageRotateTotal(images& Input);// calculate the different offset in the image
void GetExpRotate(images& refImage, images& shiftedImage,double rotate_bit,double *rotate_angle);//get the rotational offset
void GetExpShift(images& refImage, images& shiftedImage, int shift_bits, int shift_ret[2]);
//get the shift offset
```

MTB.cpp

```
void MTB::ComputeImage( images& Input, images& MTB_image, images& exclusion_image)
{

    int mid_threshold;
    unsigned char* data_input = (unsigned char*) (Input.GetImage()->imageData);
    int step = Input.GetImage()->widthStep/sizeof(unsigned char);

    int sum = 0;
    for (int i = 0; i < Input.GetHeight(); i++)
    {
        for (int j = 0; j < Input.GetWidth(); j++)
        {
            sum += data_input[i*step +j];
        }
    }

    mid_threshold = sum /(Input.GetHeight()*Input.GetWidth());

    if(mid_threshold > 250)
    {
        mid_threshold = 250;
    }
    if(mid_threshold < 15)
    {
        mid_threshold = 12;
    }
```

```
    unsigned char* data_MTB = (unsigned char*) (MTB_image.GetImage()->imageData);
    unsigned char* data_EXU = (unsigned char*) (exclusion_image.GetImage()->imageData);

    for(int i = 0; i < MTB_image.GetHeight(); i++)
    {
        for (int j = 0; j < MTB_image.GetWidth(); j++)
        {
            if (data_input[i*step + j] <= mid_threshold)
            {
                data_MTB[i*step + j] = 0;

            }
            else
            {
                data_MTB[i*step + j] = 255;

            }
            if (data_input[i*step + j] >= mid_threshold-4  && data_input[i*step + j] <=
mid_threshold+4)
            {
                data_EXU[i*step + j] = 0;
            }
            else
            {
                data_EXU[i*step + j] = 255;
            }
        }

    }

    if(mid_threshold < 0)
    {


        CvMat* MTB = cvCreateMat(MTB_image.GetHeight(),MTB_image.GetWidth(), CV_8UC1);
        CvMat* MTB_erode = cvCreateMat(MTB_image.GetHeight(),MTB_image.GetWidth(),CV_8UC1);
        CvMat* MTB_close = cvCreateMat(MTB_image.GetHeight(),MTB_image.GetWidth(),CV_8UC1);
        unsigned char *p = (unsigned char*)cvPtr2D(MTB, 0, 0);
        for(int i = 0; i < MTB_image.GetHeight(); i++)
        {
            for (int j = 0; j < MTB_image.GetWidth(); j++)
            {
```

```
                *p = data_MTB[i*step + j];
                p++;
            }
        }
        cvDilate(MTB,MTB_erode,NULL,2);
        cvErode(MTB_erode,MTB_close);
        p = (unsigned char*)cvPtr2D(MTB_close, 0, 0);
        int sum = 0;
        for(int i = 0; i < MTB_image.GetHeight(); i++)
        {
            for (int j = 0; j < MTB_image.GetWidth(); j++)
            {
                data_MTB[i*step + j] = *p;
                sum += *p;
                p++;
            }
        }
        int midthreshold = sum/( MTB_image.GetHeight()* MTB_image.GetWidth());
        //ofstream outfile("e:\\mtb.txt");
        for(int i = 0; i < MTB_image.GetHeight(); i++)
        {
            for (int j = 0; j < MTB_image.GetWidth(); j++)
            {
                if (data_MTB[i*step + j] <= midthreshold)
                {
                    data_MTB[i*step + j] = 0;
                    //outfile<<(int)data_MTB[i*step + j]<<'\t';
                }
                else
                {
                    data_MTB[i*step + j] = 255;
                }
            }

        }
    }
    return;

}

void MTB::ImageRotate(images& Input,images& input_2,images& Output,double angle)
{
    double anglerad  = (CV_PI* (angle/180)) ;
    IplImage* dst = cvCloneImage( Input.GetImage() );
```

```
     float m[6];
    CvMat M = cvMat( 2, 3, CV_32F, m );
     CvPoint2D32f center;
     center.x=float (Input.GetImage()->width/2.0+0.5);//float (Img_tmp->width/2.0+0.5);
     center.y=float (Input.GetImage()->height/2.0+0.5);//float (Img_tmp->height/2.0+0.5);
   cv2DRotationMatrix( center, angle,1, &M);

   cvWarpAffine( Input.GetImage(), dst, &M,CV_INTER_LINEAR,cvScalarAll(0) );

   Output.CloneImage(dst,dst->width,dst->height);
}

void MTB::GetExpRotate(images &refImage, images &shiftedImage, double rotate_bit, double*
rotate_angle)
{
          int min_err;
          images tb1(refImage.GetWidth(),refImage.GetHeight(), 1);
          images tb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(), 1);

          images eb1(refImage.GetWidth(),refImage.GetHeight(), 1);
          images eb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(), 1);

          ComputeImage(refImage, tb1, eb1);
          ComputeImage(shiftedImage, tb2, eb2);

          min_err = refImage.GetHeight() * refImage.GetWidth();
      for(double i = -rotate_bit ; i <= rotate_bit; i+= 0.5)
      {
               int err;
               images rotated_tb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(),1);
               images rotated_eb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(),1);

               images diff_b(shiftedImage.GetWidth(),shiftedImage.GetHeight(),1);

               ImageRotate(tb2,tb1,rotated_tb2,i);
               ImageRotate(eb2,eb1,rotated_eb2,i);

               ImageRotateXOR(tb1,rotated_tb2,diff_b);
               ImageAND(diff_b,eb1,diff_b);
               ImageAND(diff_b,rotated_eb2,diff_b);

               err = ImageRotateTotal(diff_b);
                    cout << err << " #####" << i << endl;
```

```
            if(err < min_err)
            {
                *rotate_angle = i ;
                min_err = err ;
                //test
            }
        }
}
void MTB::GetExpShift(images &refImage, images &shiftedImage, int shift_bits, int shift_ret[])
{
        int min_err;
        int cur_shift[2];
        images tb1(refImage.GetWidth(),refImage.GetHeight(), 1);
        images tb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(), 1);
        images eb1(refImage.GetWidth(),refImage.GetHeight(), 1);
        images eb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(), 1);
        if(shift_bits > 0)
        {
            images sml_img1(refImage.GetWidth()/2,refImage.GetHeight()/2, 1);
            images sml_img2(shiftedImage.GetWidth()/2,shiftedImage.GetHeight()/2, 1);
            ImageShrink(refImage,sml_img1);
            ImageShrink(shiftedImage, sml_img2);
            GetExpShift(sml_img1, sml_img2, shift_bits-1, cur_shift);
            cur_shift[0] *= 2;
            cur_shift[1] *= 2;
        }
        else
        {
            cur_shift[0] = 0;
            cur_shift[1] = 0;
        }

        ComputeImage(refImage, tb1, eb1);
        ComputeImage(shiftedImage, tb2, eb2);


        min_err = refImage.GetHeight() * refImage.GetWidth();

        for(int i = -1 ; i <= 1 ;i++)
        {
            for(int j = -1 ; j<= 1 ; j++)
            {
                int xs = cur_shift[0] + i;
                int ys = cur_shift[1] + j;
```

```
                    images shifted_tb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(),1);
                    images shifted_eb2(shiftedImage.GetWidth(),shiftedImage.GetHeight(),1);
                    images diff_b(shiftedImage.GetWidth(),shiftedImage.GetHeight(),1);
                    int err;

                    ImageShift(tb2,xs,ys,shifted_tb2);
                    ImageShift(eb2,xs,ys,shifted_eb2);

                    ImageXOR(tb1,shifted_tb2,diff_b);
                    ImageAND(diff_b,eb1,diff_b);
                    ImageAND(diff_b,shifted_eb2,diff_b);

                    err = ImageTotal(diff_b);

                    if(err < min_err)
                    {
                        shift_ret[0] = xs;
                        shift_ret[1] = ys;
                        min_err = err;
                    }
                }
            }
}
```

Sift.h

```
int _sift_features( IplImage* img,CvSeq* feat, int intvls,
                            double sigma, double contr_thr, int curv_thr,
                            int img_dbl, int descr_width, int descr_hist_bins );
```

sift.cpp

```
int sift::_sift_features(IplImage *img,CvSeq* feat, int intvls, double sigma, double contr_thr,
int curv_thr, int img_dbl, int descr_width, int descr_hist_bins)
{
    IplImage* init_img;
    IplImage*** gauss_pyr, *** dog_pyr;
    CvMemStorage* storage;
    CvSeq* features;
    int octvs, i, n = 0;

    //The gaussian blur image
    init_img = create_init_img( img, img_dbl, sigma );
    //Obtain the gaussian convolution
    octvs = log( (double)MIN( init_img->width, init_img->height ) ) / log(2.0) - 2;
```

```
    //Establish gaussian pyramid
    gauss_pyr = build_gauss_pyr( init_img, octvs, intvls, sigma );
    //DOG establish pyramid.
    dog_pyr = build_dog_pyr( gauss_pyr, octvs, intvls );

    storage = cvCreateMemStorage( 0 );
    features = scale_space_extrema( dog_pyr, octvs, intvls, contr_thr,
        curv_thr, storage );

    calc_feature_scales( features, sigma, intvls );

    if( img_dbl )
        adjust_for_img_dbl( features );
    calc_feature_oris( features, gauss_pyr );
    compute_descriptors( features, gauss_pyr, descr_width, descr_hist_bins );
    n = features->total;
    *feat = *features;
    return n;
}
```