Using Refined Least Square Image Matching to Improve 3D Reconstruction under Gaussian Blur and Motion Blur images

Research Thesis

Presented in Partial Fulfillment of the Requirements for

Graduation with Honor Distinction

By

Xuyang Han

Undergraduate program in Environmental Engineering

The Ohio State University

2017

Abstract

Image matching, defined as the process of finding corresponding points across multiple images, is a critical step for accurate measurement of 3D objects. Least Square Image Matching (LSM), as a classic point matching method, is known for its high accuracy over the competing approaches (up to 1/10th of pixel-size). However, the major drawback of Least Square Image Matching is its high requirement on algorithm convergence, which demands good image texture and initial solution. In addition, it deals with reasonable image distortions, while being limited by occluded area where part of the area around the points is not visible in all images and high computational resources. For this reason, the state-of-the-art procedure tends to use rather less accurate, but faster and more robust point matching approaches. But these approaches, which currently are widely used in the 3D reconstruction software, has not been examined carefully on their reliability and accuracy for images taken under various suboptimal acquisition conditions (e.g. image motion blur). Such data, if being the only source and impossible for recollection, such as those from emergent disasters or historical sources, demand for much more reliable and accurate methods for reconstructing 3D information.

In this research, the classical LSM matching approach has been revisited and the LSM robustness has been improved by modification applying bound constraints optimization and good window size selection. Bound constraints are set based on the previous study. The bound in each iteration procedure can constrain the search for a solution to a

reasonable region and prevent the parameters for affine deformations and the illuminations from straying from the correct values. The window size will be adjusted to be large enough to include strong features which has enough intensity variation for reliable matching, but small enough to avoid severe occlusion problems. The modified LSM algorithm has been implemented on disparity map, and the convergence has been verified.

Furthermore, how photogrammetric 3D reconstruction performs using imageries under suboptimal scenario has been evaluated. 3D reconstruction using various blurred images with different extent of blurriness is experimented using Apero photogrammetry software. Such blurriness includes Gaussian blur (often resulting from camera defocusing) and directional and unidirectional motion blur (often resulting from unmatched shuttle speed to motion speed or platform oscillation (windy conditions)). 3D reconstruction based on these blurry images which are frequently observed in small UAVs surveying will degrade the quality and accuracy of 3D modeling. To carry out a controlled analysis, the blurred images used for experiment are synthetically generated using different blurring functions. The accuracy of the bundle adjustment and 3D dense reconstruction using corresponding points from state-of-art point detectors (e.g. SIFT) and corresponding points refined by LSM is compared. The robustness of the state-of-the-art point detectors and the refinement capability of LSM under different blurriness conditions is concluded.

Acknowledgements

My deepest gratitude to my advisor, Dr. Rongjun Qin, who has consistently and diligently supported me to be a better student and researcher. Without his guidance, it would be impossible to learn so many things in Photogrammetry and Computer Vision field. I would like to express my gratitude for providing me the incredibly valuable opportunity to learn computer engineering for future research. But more than his guidance in my work, I want to thank him for his ability to constantly push me out of my comfort zone and continue the never-ending process of learning. Gratitude to my committee member, Dr. Alper Yilmaz for being such a great teacher. I would like to thank Singapore-ETH Center, Future Cities Lab to provide the NUS dataset for this research. I also appreciate my family who supported my education abroad in the U.S. Finally, gratitude to all my friends who provided their supports and feedbacks.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

1.1 Background:

Photogrammetry is the science of making measurements from photographs, especially for recovering the exact positions of surface points. Three-dimensional reconstruction, as the core of the photogrammetry technology, is a process to convert multiple two-dimensional stereo images of an object back to a three-dimensional model. Finding the corresponding pixel on two images of the same physical region, defined as image matching, is a crucial step in 3D reconstruction. The accuracy of image matching will directly influence the accuracy of 3D reconstruction, thus affecting the measurement accuracy and further analysis.

Least Square image matching is a very classic image matching method, known for the high accuracy, which can be up to 1/10th pixel. The concept of Least Square is to find an approximate regression solution of overdetermined systems ensuring the overall solution minimizes the sum of squares of the residuals in the results of every single equation. After LSM concept was created and developed, it was firstly applied in Photogrammetry measurement by Förstner (1982). Least Square concept in image matching is to minimize the gray level differences between the template and the matching windows whereby the position and the shape of the matching window are parameters to be determined in adjustment process. Since the noises are non-known for all pixels, the system is nonlinear and needs to be solved by iterative refinement. In each iteration, the system is approximated by Taylor linearization, and optimized to a new solution until convergence.

Even the LSM has high accuracy, instability is its major drawback. LSM requires good image texture and initial solution for algorithm convergence. In addition, unreasonable image distortions and occlusion problems may lead to diverged iterations if the extent is too high. Even though some refinement approach such as Adaptive Window adjustment (Kanade, Okutomi, 1991) and bound constrained optimization (Hu, 2016) was researched, today's 3D reconstruction software do not use highly accurate Least Squares Matching approach but use less accurate but more robust matching method.

Unmanned aerial systems (UAS) and unmanned aerial vehicles (UAV) are playing increasingly important roles in this century due to its versatile functionalities of surveying. Researchers are very interested in these flexible and low-cost plat-form which reduces data acquisition time and accessible to dangerous environment monitoring scenes. However, motion blur images data frequently occurs in small UAVs surveying and the quality and accuracy of 3D modeling can be degraded. In addition, the state of the art method, which are widely used nowadays may not exam the accuracy for these images taken under various suboptimal acquisition conditions.

1.2 Research Significance

Through this research, the improved accuracy in image matching, which is a crucial issue, can make progress and enable better performance in a broad range of applications: digital infrastructure/heritage recording, precise engineering measurement, Computational Fluid Dynamics (CFD) Simulation, land boundary mapping and terrain topography. Furthermore, academic achievements from this research can provide references for research communities. Since LSM is a conventional and widely researched approach for image matching, a good result overcoming the common drawbacks may contribute important knowledge to the research field.

1.3 Research Objective

In performing this research project, three goals have been pursued. Firstly, revisiting the classic LSM matching approach and improving the LSM robustness by modification applying bound constraints optimization and using good window size. Secondly, evaluate the sift point accuracy using Gaussian blur and motion blur images data based on 3D reconstruction result based on corresponding points generated by Apero photogrammetry software. Thirdly, using refined LSM to optimize the 3D reconstruction accuracy under these blur images, and comparing the accuracy of the bundle adjustment and 3D dense reconstruction using corresponding points from state-of-art point detectors (e.g. SIFT) and corresponding points refined by LSM, then evaluate if the accuracy is increased.

Chapter 2: Methodology

2.1 Cross Correlation (CC)

Cross correlation is an area-based matching process to look for a best match of certain template within a larger size image background. It exhaustively searches every single position within the background (M pixels in total) and find the position where it has the maximum similarity. Cross correlation only consider the plain translation. Other parameters such as affine transformation, brightness and contrast are ignored by far. The similarity, ρ, at the certain position, (u,v), is calculated by the following Equation [1]:

$$\rho_{g_1 g_2}(u,v) = \frac{\sigma_{g_1 g_2}(u,v)}{\sigma_{g_1}(u,v) * \sigma_{g_2}} \qquad [1]$$

where:

1. σ_{g_2}represent standard deviation of intensity values of template g2:

$$\sigma_{g_2}^2 = \frac{1}{M-1} * \left[\sum_{m=1}^{M} g_2^2(i_m, j_m) - \frac{1}{M} * \left(\sum_{m=1}^{M} g_2(i_m, j_m)\right)^2\right] \qquad [2]$$

2. $\sigma_{g_1}(u,v)$ represent the standard deviation of intensity values of query image g1 in the area of template g2 at current offset [u, v]:

$$\sigma_{g_1}^2(u,v) = \frac{1}{M-1} * \left[\sum_{m=1}^{M} g_1^2(i_m - u, j_m - v) - \frac{1}{M} * \left(\sum_{m=1}^{M} g_1(i_m - u, j_m - v)\right)^2\right] \qquad [3]$$

3. $\sigma_{g_1 g_2}(u,v)$ represent the covariance between intensity of g1 and g2 in the area of template g2 at current offset [u, v]:

$$\sigma_{g_1g_2}(\mathrm{u},\mathrm{v}) = \frac{1}{\mathrm{M}-1} * [\sum_{m=1}^{M} g_1(i_m - u, j_m - v) g_2(i_m, j_m) - \frac{1}{M} * \sum_{m=1}^{M} g_1(i_m - u, j_m - v) * \sum_{m=1}^{M} g_2(i_m, j_m)] \qquad [4]$$

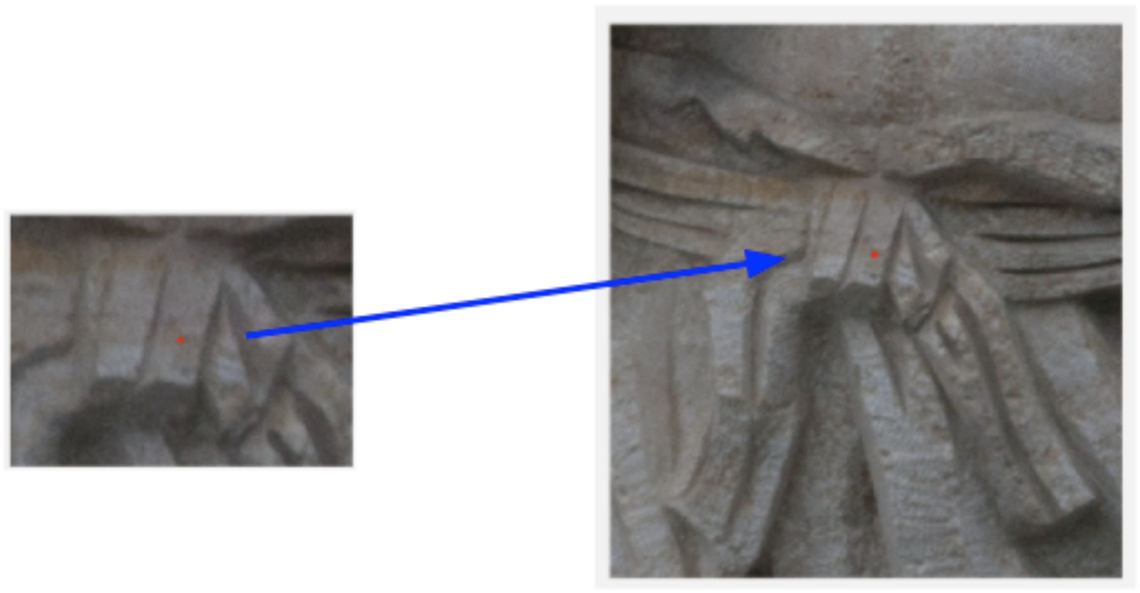

Figure 1. Example of Area-based Point Matching using Cross Correlation

Figure 1 shows an example of Area-based Point Matching using Cross Correlation. After searching line by line for every single position within the background, the similarity of intensities in each pixel is calculated using Equation [1]. 3D surface of the similarities is represented in the following **Figure 2**. The peak in the surface indicates the position with the maximum similarity position and the coordinates of the position will be used as initial solution for further LSM matching.

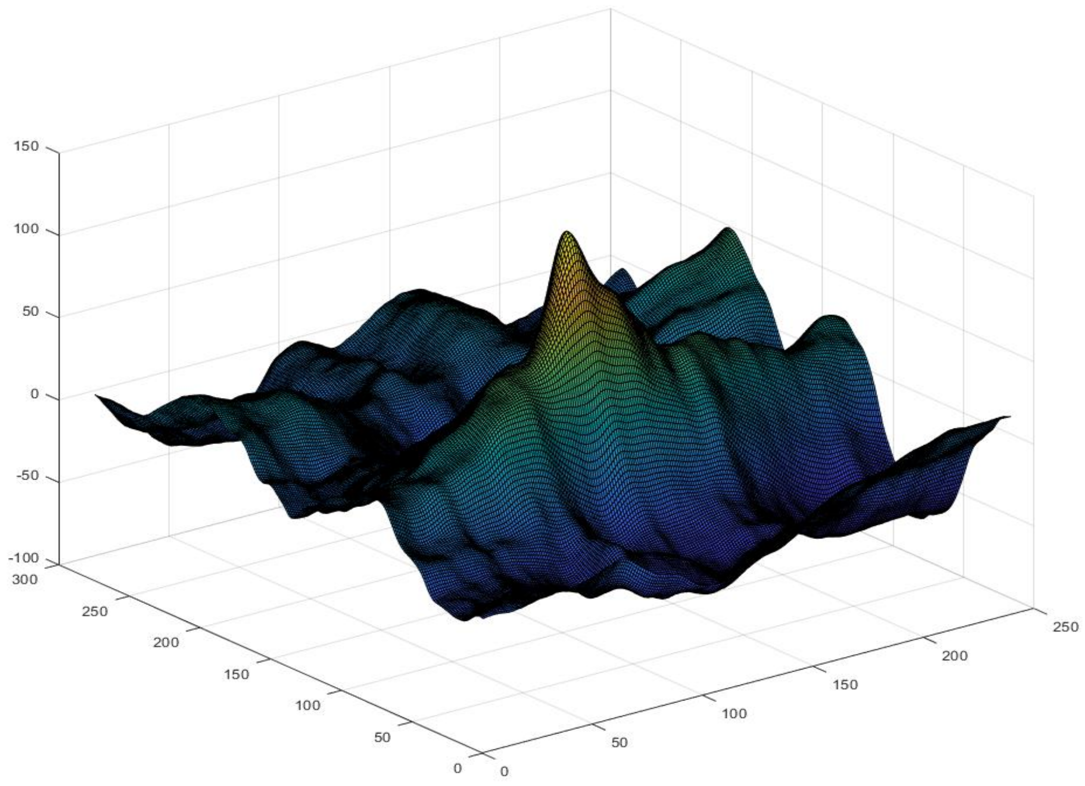

Figure 2. Similarity Distribution among All pixels

The major drawback of Cross Correlation process is that the algorithm will not return valid matching in repetitive pattern texture area. Since more than one feature is very similar to the template, it will be difficult to distinguish it as the right match. **Figure 3** is a case in point. Furthermore, since the affine transformation is not considered in cross correlation, higher than 20 degrees' rotation will dramatically decrease the similarity, thus no valid matching will be obtained.

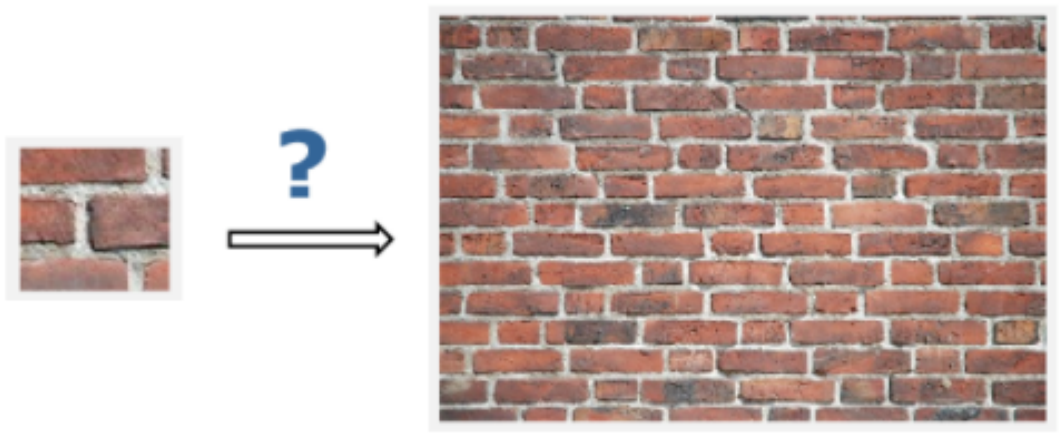

Figure 3. Repeating patterns such as on a brick wall can provide incredible difficulties when trying to match the template in the background

In this research, Cross Correlation was used as the first step of LSM to determine a good initial solution. Repetitive features and high rotation transformation has been avoided particularly. Using the result from the cross correlation, further iterations will consider more factors such as brightness, contrast and affine transformation, and the more accurate position will be obtained based on the result of LSM.

2.2 Least Squares Matching (LSM) Fundamentals

Least Square Matching is a generalization of Cross Correlation but consider more factors supporting arbitrary geometric and radiometric transformations. Instead of exhaustively search every possibility, an initial guess is used to go through iterations. A good initial solution is required for LSM and a simple way to find one is the Cross Correlation.

For the image pixels g1(x, y) and g2(x', y') of two images in a square window (normally 31*31 as used), the fundamental idea of LSM is to determine a set of parameters to satisfy the following observation condition:

$$x' = a_1 * x + a_2 * y + a_3 \qquad [5]$$

$$y' = b_1 * x + b_2 * y + b_3 \qquad [6]$$

$$g_2(x, y) = k_1 g_1(x', y') + k_2 \qquad [7]$$

Where:

1. a1 through b3 are used to describe the affine deformations using 6 unknown parameters shown in Equation [5] & [6]
2. k1 and k2 control the contrast and brightness differences respectively, shown in Equation [7]
3. the random noise, which is assumed to be zero when perfectly matched, shown in Equation [7]
4. There are a total of 8 unknown parameters in LSM in Eq. [5], [6] and [7] as

$$x = [a_1, a_2, a_3, b_1, b_2, b_3, k_1, k_2]^T \qquad [8]$$

The initial solution is set by default. a1, b2 and k1 are set as one. a2, b1 and k2 are set as zero. a3 and b3 are determined by CC in the previous step. Those default values in the initial solution showing no rotation and radiometric transformations:

$$x_0 = [1, 0, a_3, 0, 1, b_3, 1, 0] \qquad [9]$$

After considering those factors, the values of eight parameters will be updated to the optimal solution. However, the solution cannot be acquired directly since it is a nonlinear function. Therefore, using Taylor linearization, it is transformed to a linear one as illustrated in Equation [11]:

$$F(x, y) = g_2(x, y) - k_1 g_1(x', y') - k_2 \qquad [10]$$

$$F = F_0 + \frac{\partial F}{\partial a_1} * (\Delta a_1) + \frac{\partial F}{\partial a_2} * (\Delta a_2) + \frac{\partial F}{\partial a_3} * (\Delta a_3) + \frac{\partial F}{\partial b_1} * (\Delta b_1) + \frac{\partial F}{\partial b_2} * (\Delta b_2) +$$

$$\frac{\partial F}{\partial b_3} * (\Delta b_3) + \frac{\partial F}{\partial k_1} * (\Delta k_1) + \frac{\partial F}{\partial k_2} * (\Delta k_2) \qquad [11]$$

where:

$$\frac{\partial F}{\partial a_1} = -k_1 * \frac{\partial g_1}{\partial a_1} = -k_1 * \frac{\partial g_1}{\partial x'} * \frac{\partial x'}{\partial a_1} = -k_1 * g_{1_x} * x \qquad [12]$$

$$\frac{\partial F}{\partial a_2} = -k_1 * \frac{\partial g_1}{\partial a_2} = -k_1 * \frac{\partial g_1}{\partial x'} * \frac{\partial x'}{\partial a_2} = -k_1 * g_{1_x} * y \qquad [13]$$

$$\frac{\partial F}{\partial a_3} = -k_1 * \frac{\partial g_1}{\partial a_3} = -k_1 * \frac{\partial g_1}{\partial x'} * \frac{\partial x'}{\partial a_1} = -k_1 * g_{1_x} \qquad [14]$$

$$\frac{\partial F}{\partial b_1} = -k_1 * \frac{\partial g_1}{\partial b_1} = -k_1 * \frac{\partial g_1}{\partial y'} * \frac{\partial y'}{\partial b_1} = -k_1 * g_{1_y} * x \qquad [15]$$

$$\frac{\partial F}{\partial b_2} = -k_1 * \frac{\partial g_1}{\partial b_2} = -k_1 * \frac{\partial g_1}{\partial y'} * \frac{\partial y'}{\partial b_2} = -k_1 * g_{1_y} * y \qquad [16]$$

$$\frac{\partial F}{\partial b_3} = -k_1 * \frac{\partial g_1}{\partial b_3} = -k_1 * \frac{\partial g_1}{\partial y'} * \frac{\partial y'}{\partial b_3} = -k_1 * g_{1_y} \qquad [17]$$

$$\frac{\partial F}{\partial k_1} = -g_1(x', y') \qquad [18]$$

$$\frac{\partial F}{\partial k_2} = -1 \qquad [19]$$

In practice, g_{1_x} and g_{1_y} can be computed as follows using gray values from the background image:

$$g_{1_x} = [g_1(x'+1, y') - g_1(x'-1, y')]/2 \qquad [20]$$

$$g_{1_y} = [g_1(x', y'+1) - g_1(x', y'-1)]/2 \qquad [21]$$

As normal, 31*31 size template is used. Then 961 linearization equations like Eq. [11] can be rearranged to the following form:

$$J * \Delta \mathrm{x} = \Delta \mathrm{O} \qquad [22]$$

where:

1. J is Jacobian matrix that records the partial derivatives with respect to each unknown parameter.
2. Δx is the incremental vector.
3. ΔO is the residual vector showing the signed difference in that pixel, which is $[g_2(x,y) - k_1 g_1(x', y') - k_2]_0$

After expanding the nonlinear form with a first order Taylor series and assigning appropriate initial values for the parameter x_0, the increment vector can be solved:

$$\Delta x = (J^T * J)^{-1} * (J^T * \Delta O) \qquad [23]$$

The procedure will iterate until it reaches a certain termination criterion, such as all elements in Δx vector converge to zero. In that case, x vector will converge to the optimum solution.

2.3 Template Window Size Selection

A key problem in area-based matching by computing least sum of squared differences lies in selecting an appropriate window size, especially when dealing with plain texture. The window size must be large enough to include enough strong intensity variation for reliable matching. However, the template size must be small enough to avoid the effects of projective distortion such as occlusion. If the window is too small and does not cover enough strong features, it gives a poor matching solution, because the parameters convergence is bad. But, on the other hand, if the window is too large and covers too much strong features which are not just affine transformation across each images, the convergence solution will not represent correct matching. Because larger template which contains more and stronger features, will match them as a whole. Without assigning any weight to different pixels, the center of the template is just one plain pixel comparing to other strong features, thus the accuracy will not be guaranteed using larger template size.

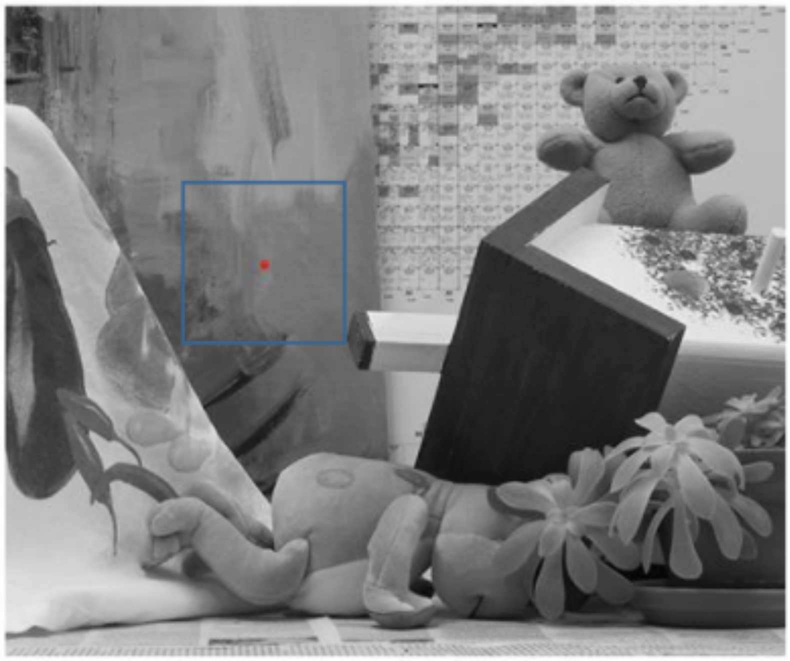

Figure 4. Example of wrong template size which contains not enough strong feature for area-based matching

For this reason, a window size must be selected wisely depending on characteristics of different data sets such as local variations of intensity and disparity. In this research, different template sizes have been tried to determine the best fit for the data sets used for evaluation. As a result, choosing the correct template window size is the foundation of a good matching.

2.4 Bound Constraint

As discussed above, every iteration a new parameter vector will be updated and be used for the initial solution for the next iteration until convergence. However, in the iteration procedure, the parameters for affine deformations and the illuminations may stray from the correct values or even exceed the areas of the whole background. These phenomenon occurs in point matching with bad initial guess, plain texture, occlusion problems, and other unreasonable distortion of the images.

In the standard photogrammetry surveying, even in extreme conditions, variations of more than 15 degrees' rotation are very rare in experience. In order to give an intuitive understanding of the extents of the affine transformation, the transformation with corresponding skewness is illustrated in the following **Figure 5**:

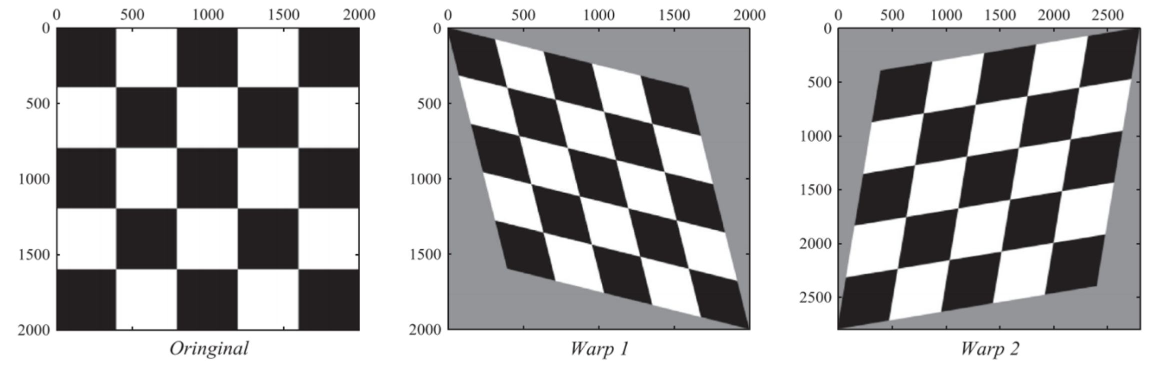

Figure 5. Example of Affine transformation with corresponding skewness

Figure 5 Illustrates two different affine transformations. The left one is the original image. The four parameters controlling affine transformation in the middle image is:

$$A = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

The four parameters controlling affine transformation in the right image is:

$$A = \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} = \begin{bmatrix} 1.2 & -0.2 \\ -0.2 & 1.2 \end{bmatrix}$$

The skewness is approximately 15 degrees for both cases. In addition, for the unknown parameters a3 and b3, which account for translations in x and y directions, the boundaries can also be gauged before optimization based on the accuracy of initial guess. No matter the initial solution is gained from Cross Correlation discussed above, or from initial disparity map and sift corresponding points that will be discussed below, more than 5-pixel distance away from the ground truth is very rare as well. Furthermore, the contrast and brightness should be also within a reasonable range. It can be noted that a contrast value that varies in the region from 0.5 to 2 and a brightness value in the range from -50 to 50 will be sufficient to address the illumination differences.

Consequently, all eight unknown parameters have powerful physical significances and thus should be constrained to reasonable regions. Therefore, lower and upper boundaries to the unknown parameters are proposed in the following Equation as the follows:

$$1 - \delta_1 < a_1 < 1 + \delta_1, 1 - \delta_1 < b_2 < 1 + \delta_1$$

$$-\delta_2 < a_2 < \delta_2, -\delta_2 < b_1 < \delta_2$$

$$-\delta_3 < a_3 < \delta_3, -\delta_3 < b_3 < \delta_3$$

$$\delta_4 < k_1 < 1/\delta_4, -\delta_5 < k_2 < \delta_5$$

where:

1. δ_1 and δ_2 control the degree of the affine transformation, and 0.2 will provide ample space for potential deformation in real data sets application.
2. δ_3 controls the translation from the initial corresponding position, and 5 pixels will be large enough considering the accuracy of the initial solution.
3. δ_4 and δ_5 control the illumination difference, and values of values of $\delta_4 = 0.5$ and $\delta_5 = 50$ are used.

2.5 Three-Dimension Reconstruction Evaluation using Blur Images

3D reconstruction using blur images are very common in practical. Gaussian blur images, defined as images with Gaussian distributed noises, often result from camera defocusing unintentionally on an unrelated object from the surveying interest subject. Motion blur images often result from the camera moving while the shutter is open. Those blur images are commonly generated using small unmanned aerial vehicles (UAV) surveying especially when researchers are increasingly interested in these flexible and low-cost plat-form to reduce data acquisition time and increase the accessibility. Furthermore, sometimes the data sets cannot be re-obtained and the blur image data are the only data to analyze such as when dealing with historical data sets or disaster data sets. Therefore, finding a way to overcome the degradation of 3D modeling reconstruction quality based on these blur images is crucial especially when today's widely used state of the art method may not exam the accuracy for these images taken under various suboptimal acquisition conditions.

In this research, 3D reconstructions using nine data sets has been evaluated. The original clear images data set is 25 images taken by small UAVs covering the main campus of National University of Singapore. Four Gaussian blur images data sets with different standard deviation of 4, 5, 7 and 8 are artificially generated using the original clear image data sets. Examples are as shown in **Figure 6**. Four motion blur images data sets using different kernel size of 15*15, 31*31, 37*37 and 41*41 are synthetically generated. Examples are as shown in **Figure 7**.

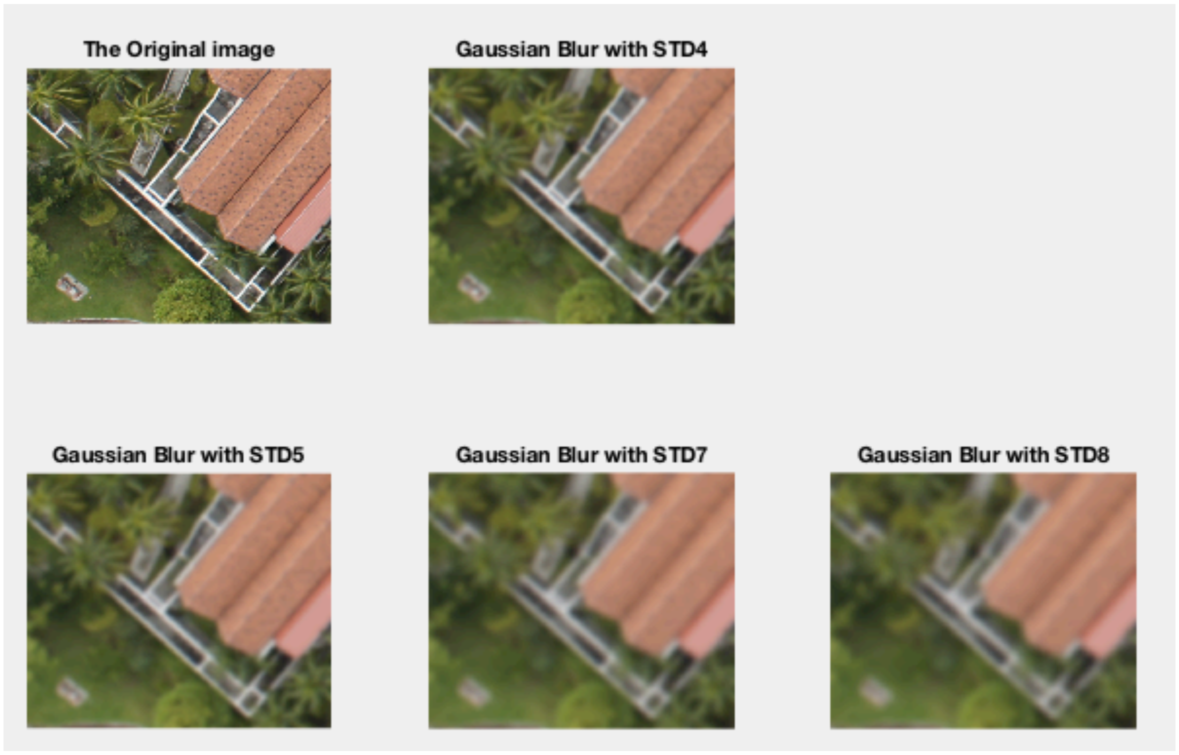

Figure 6. Example of different degrees Gaussian blur images

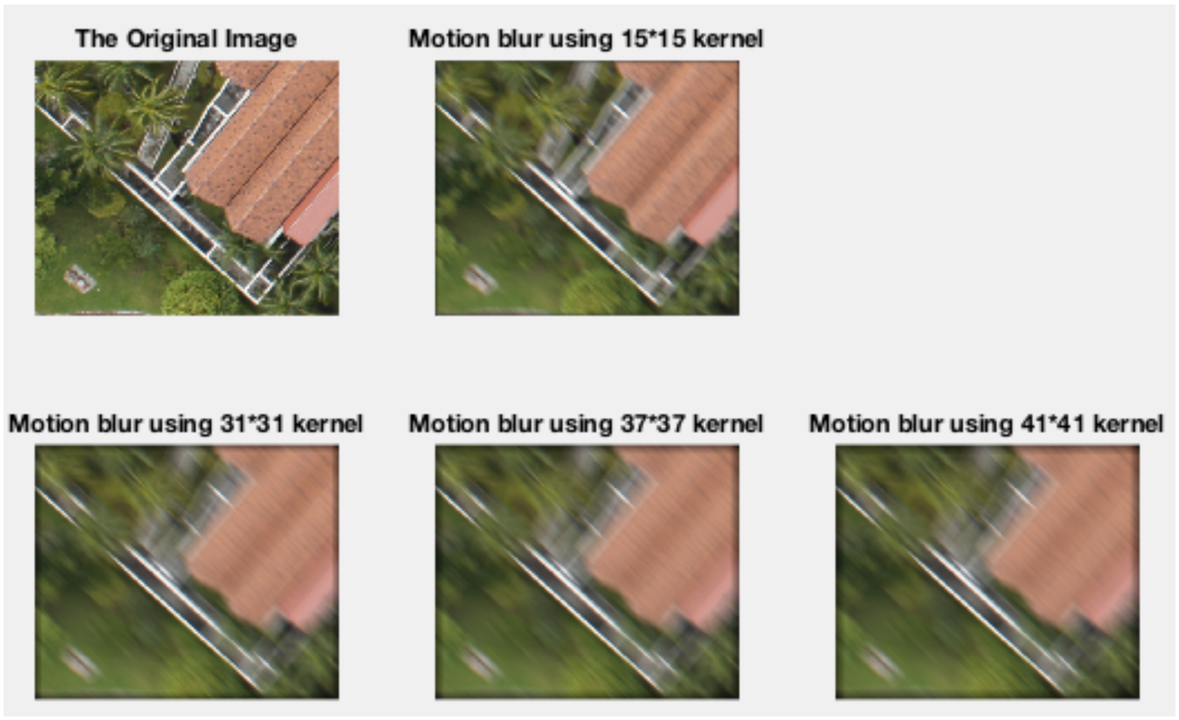
The Original Image
Motion blur using 15*15 kernel
Motion blur using 31*31 kernel
Motion blur using 37*37 kernel
Motion blur using 41*41 kernel

Figure 7. Example of different degrees motion blur images

3D reconstructions are experimented using Apero photogrammetry software and MSP software based on all those nine images data sets. Aerpo is an orientation computation software using a set of images. Firstly, SIFT tie points across all images will be extracted in the first step. Then, after running the free-network triangulation, 5 ground control points (GCPs) are marked in every images. Those five GCPs information stays the same for different data sets. The ground control points and their coordinates are used to geo-referencing to calibrate the map. Then GCP based optimization is run and the orientation information will be obtained. MSP is an operational-ready multi-stereo dense image matching software that Dr.Qin has developed for DSM generation and true ortho-photo generation from frame camera images (e.g. UAV, aerial and oblique images). Running the file generated by Apero through MSP, the generated cloud points and digital surface model will be used for further analysis. An example is shown in **Figure 8.** Different elevations are represented in different colors. Darker bluish color represents lower elevation while lighter reddish color represents higher elevation.

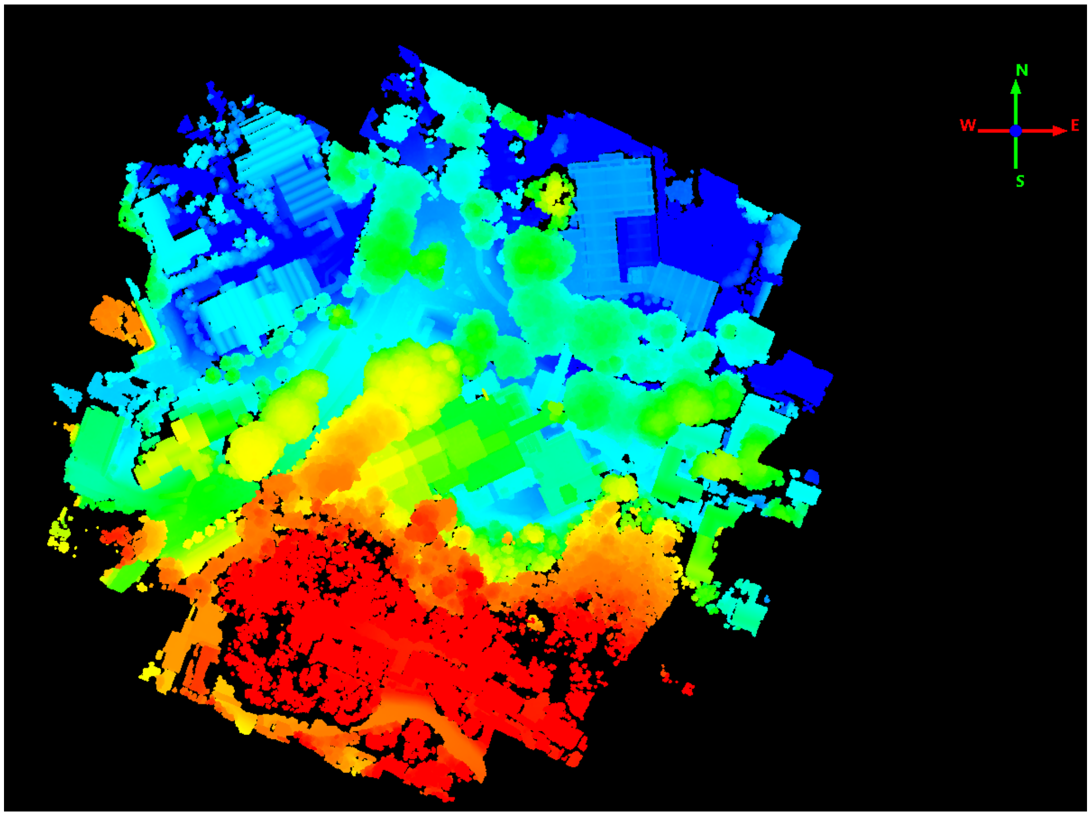

Figure 8. Example of Cloud Points generated by Apero and MSP

After eight digital surface models are generated based on blur images, the 3D reconstruction accuracies are evaluated by comparing to the DSM generated from original clear images. By comparing the depth accuracy of every single pixel to the ground truth, which is assumed to be the DSM from clear images, and calculate the sum of absolute value of the difference between generated DMS and ground truth, the trend of

3D reconstruction accuracy will be observed and that will be used as the criteria for examining the influence of Gaussian and motion blur on 3D reconstruction accuracy.

As mentioned above, SIFT tie points generation is the first step in Apero photogrammetry software. The accuracy of these corresponding points across all images is crucial for all other steps in DSM generation. Refined Least Squares Image Matching, using bound constrain optimization and optimum template size, will be used to updating all corresponding points gained by sift on blur images. Then using the same procedures to generate new digital surface models and compare those models with the previous results. Whether the 3D reconstruction accuracy can be enhanced using LSM optimized corresponding points will be determined.

Chapter 3: Result

3.1 Template Size Selection

As discussed through the whole thesis, template size selection is the foundation to get accurate area-based point matching. The template cannot be too small not to contain enough high variance features to get convergence matching result. However, larger template size will always get better matching result is a misunderstanding as well. When the template is too large to contain way too much strong features, those features matching will reduce the weight of the pixel in the center of the template, thus the accuracy will not be guaranteed. Furthermore, occlusion problems will be more severe when the template size is enlarged. In real images data sets, occlusion problems are very difficult to be avoided. When taking photo of an object in two different angles, the displacement of the background behind the object is impossible to fix. So in that case, when matching an object using template matching and including the edge of the object, the displaced backgrounds are included in the template as well. Since different texture will never be matched well, larger template size will not be always return better matching. An example of the point matching of data sets covering main campus of National University of Singapore using different template size is shown below.

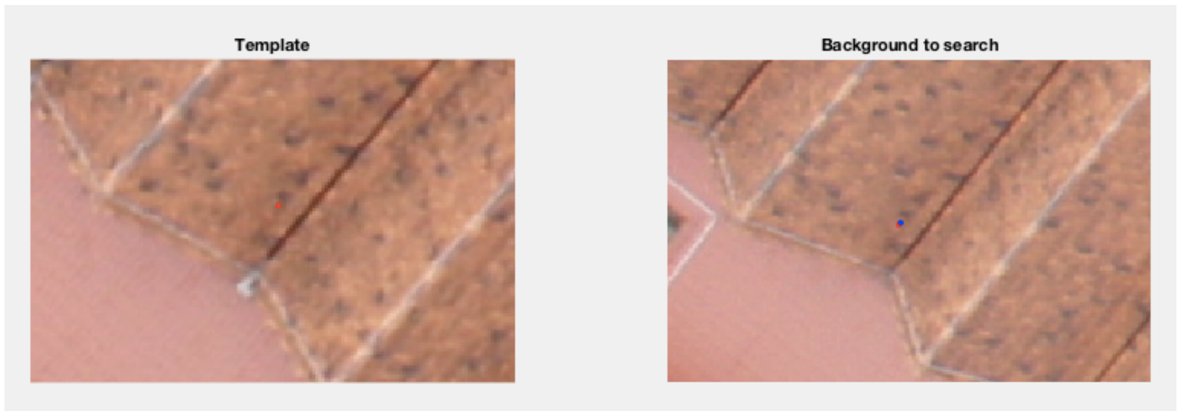

Figure 9. Example of Point Matching using 31*31 pixels' template, where red point is the matched point and blue point is the ground truth

As shown in **Figure 9**, the red point in the template (left image) is matched with the red point in the background (right image) by LSM in Gaussian blur images with standard deviation of 8 using 31*31 pixels' template. The blue point in the background (right image) is the ground truth, which is gained by SIFT using clear images.

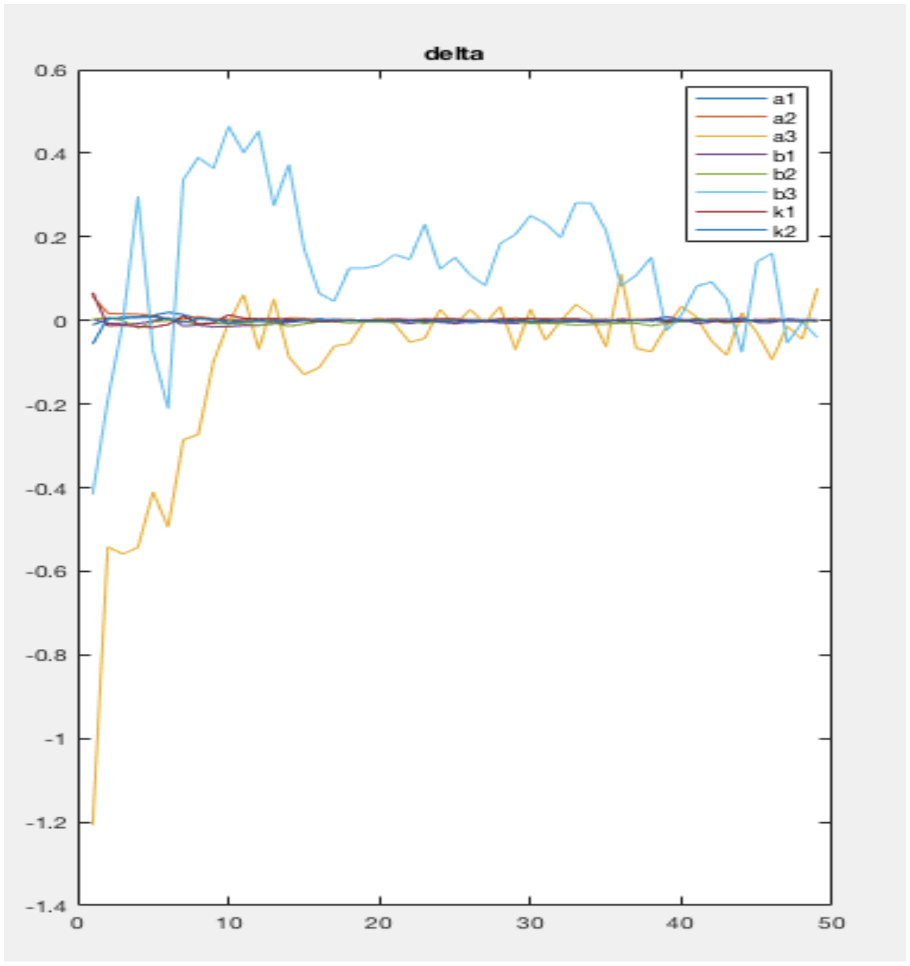

Figure 10. Parameter Adjustment Values of the Point Matching Example Using 31*31 Pixels' Template

As shown in **Figure 10**, almost all parameters converge to zero except a3 and b3, which control translation in X and Y directions. These two parameters keep pivoting around the ground truth but will never converge because the template size is too small to contain enough distinctive features. Even though no fully convergence is observed in this case, the distance from the matched point to the ground truth is just 1.68 pixels.

The following example shows how the matching result is changed when using larger template size of 131*131 pixels' template.

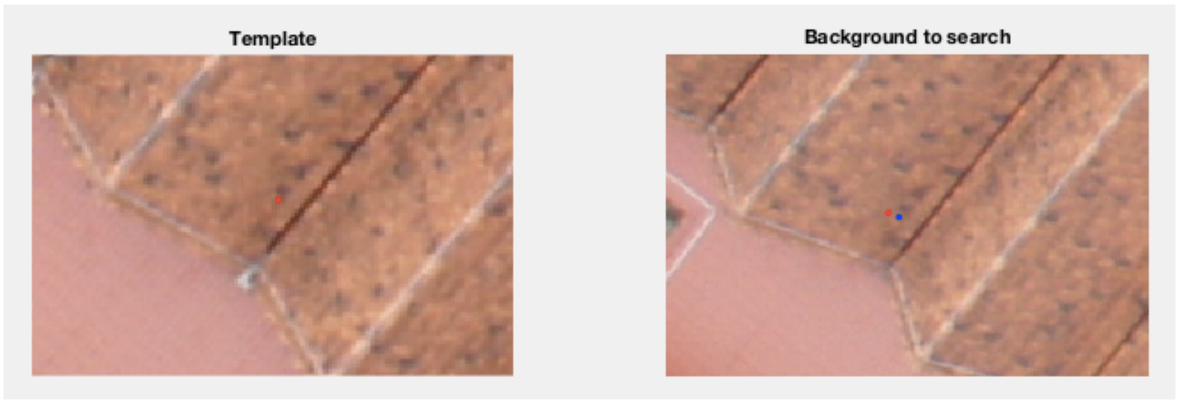

Figure 11. Example of Point Matching using 131*131 pixels' template, where red point is the matched point and blue point is the ground truth

As shown in **Figure 11**, the red point in the template (left image) is matched with the red point in the background (right image) by LSM in Gaussian blur images with standard deviation of 8 using 131*131 pixels' template. The blue point in the background (right image) is the ground truth, which is gained by SIFT using clear images.

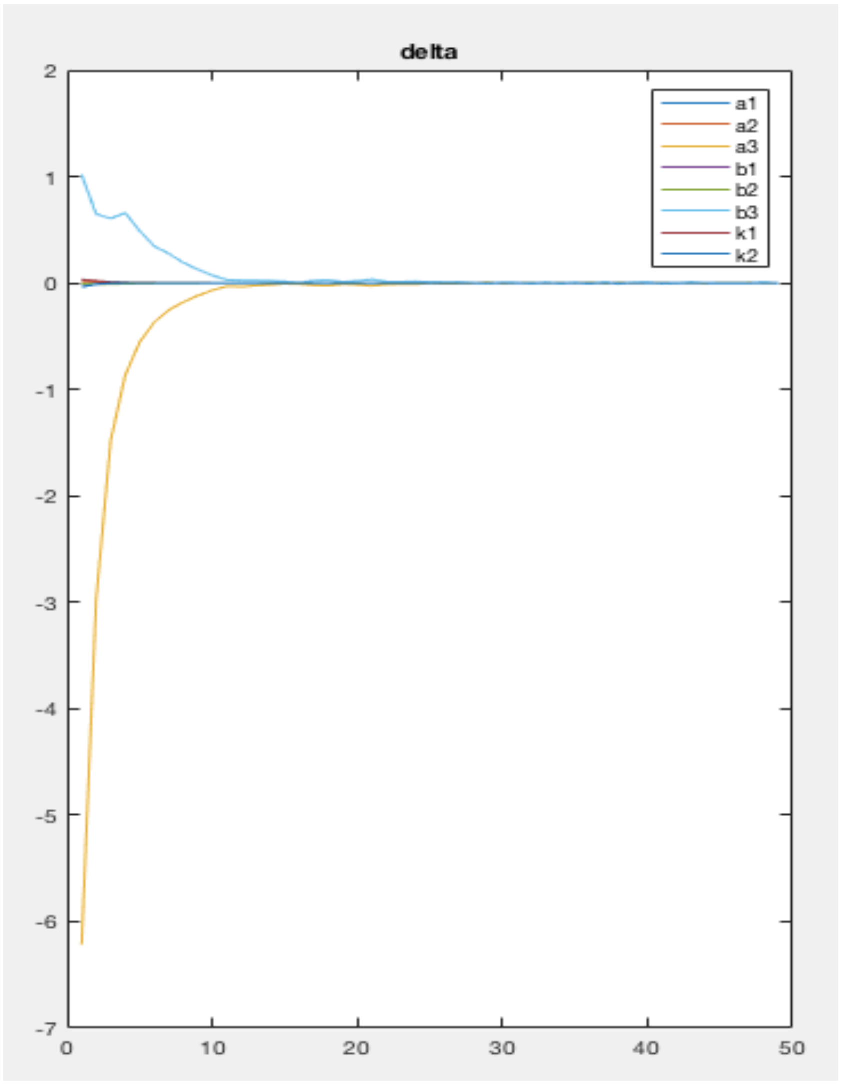

Figure 12. Parameters Adjustment Values of the Point Matching Example Using 31*31 Pixels' Template

As shown in **Figure 12**, all parameters converge to zero because the template size is large enough to contain more strong features. However, good convergence will not guarantee accuracy. The distance from the matched point to the ground truth is 5.78 pixels in this case, which is larger than using smaller template size.

In order to find the optimum template size for the Singapore data set and to reduce bias from any single case, 10 points in the image data sets are randomly selected for point matching using 5 different template sizes. The distances of the matched from the ground truth of all the cases are shown in the below ***Table 1***:

Table 1*. distance from ground truth of 10 random point matching cases with 5 different template sizes*

Template size / Point number	21*21	31*31	71*71	91*91	131*131
1	2.19	1.68	1.09	4.59	5.77
2	2.41	1.48	1.26	0.95	2.17
3	1.74	2.25	2.43	2.47	5.36
4	2.67	2.86	2.29	7.64	7.32
5	3.21	2.22	2.65	0.96	3.83
6	1.86	1.46	3.24	3.11	2.39
7	2.39	2.19	3.05	2.33	1.8
8	2.62	2.43	0.99	1.46	3.6
9	1.16	1.81	0.9	1.99	0.43
10	2.32	1.58	1.65	2	2.54
average	2.257	1.996	1.955	2.75	3.521
standard deviation	0.5374392989	0.4413887176	0.8390262213	1.921728389	1.9888914

As recorded in ***Table 1***, the average of the errors and the standard deviation of the errors have been calculated. The average error gets smaller when increasing the template size at the beginning, then gets larger when keep enlarging the template size. Average error using 31*31 pixels' template and 71*71 pixels' template are the smallest, then the standard deviation is compared within these two cases. Since using 71*71 pixels' template will return some unreasonable matching result, which are marked in red color in the table, the standard deviation is larger. Besides this reason, larger template will reduce the algorithm speed dramatically. Overall, 31*31 pixels' template is selected for the images data sets covering the main campus of National University of Singapore.

3.2 Bound Optimization

As discussed above, Least Square algorithm updates 8 parameters until converging to optimal solution through iterations. Each iteration, a new solution will be updated based on the initial solution. However, divergence problems occur when doing point matching in plain texture areas. The parameters for affine deformations and the illuminations may stray from the correct values or even exceed the areas of the whole background. The bound introduced will be constrain the parameters within reasonable range. Based on the discussion in methodology, values for bound have been set properly and ample space for the deformation has been ensured. An example of bound functionality has been shown in the following figures.

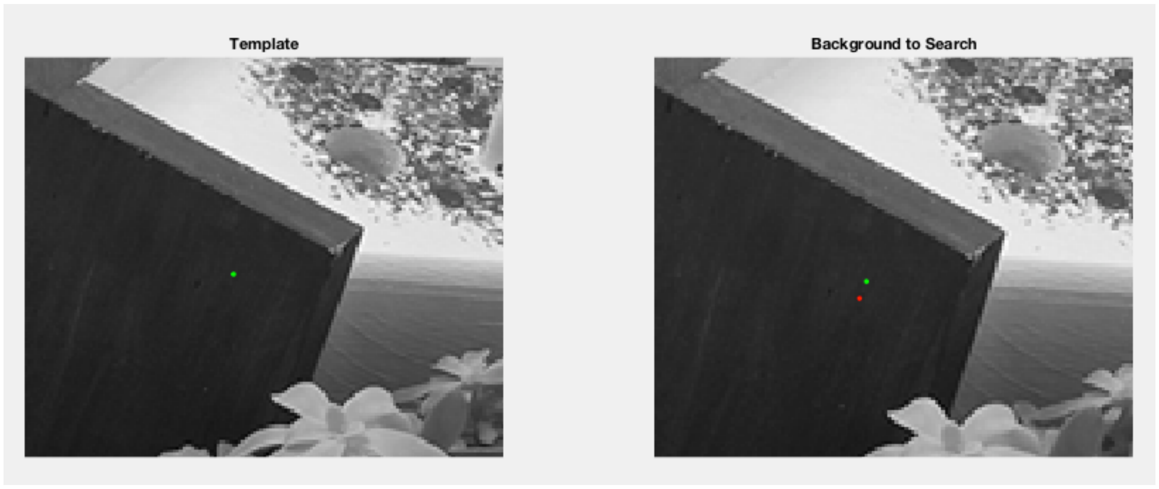

Figure 13. Example of Point Matching without bound optimization, where green point is the ground truth and red point is the matched point

As shown in **Figure 13**, the green point in the template (left image) is matched with the red point in the background (right image) by LSM without bound optimization. The green point in the background (right image) is the ground truth, which is gained by disparity map provided by Stereo - Middlebury Computer Vision open sources.

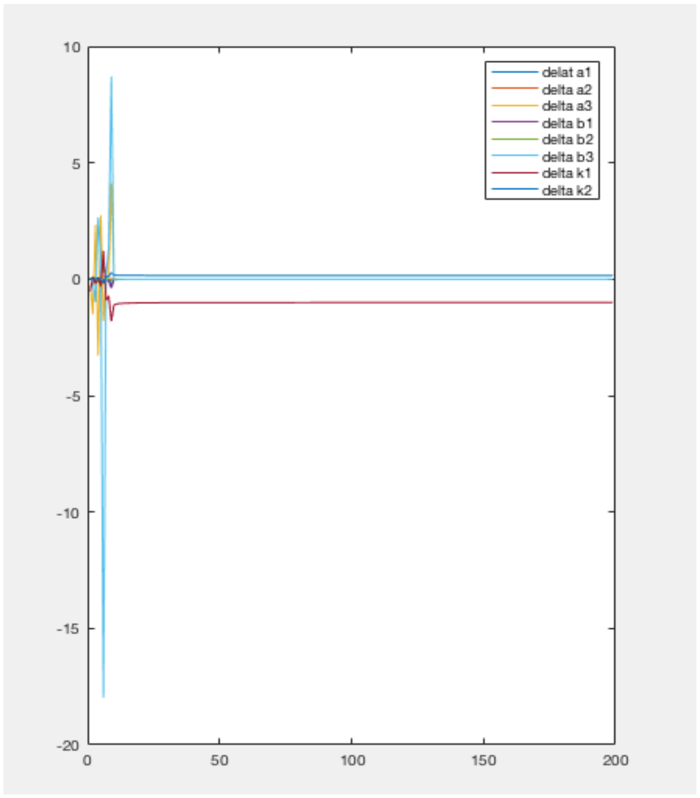

Figure 14. Parameters Adjustment Values of the Point Matching Example without using bound optimization

As shown in **Figure 14**, a typical divergence occurs in this matching. After unreasonable parameters which are controlling translation are returned in two iterations, the algorithm stops to be functional anymore. After those two peak values are returned, the parameters will never converge to zero anymore. The distance from the matched point to the ground truth is 5.76 pixels in this case.

The following example shows how the matching result is refined when using bound constrain optimization method.

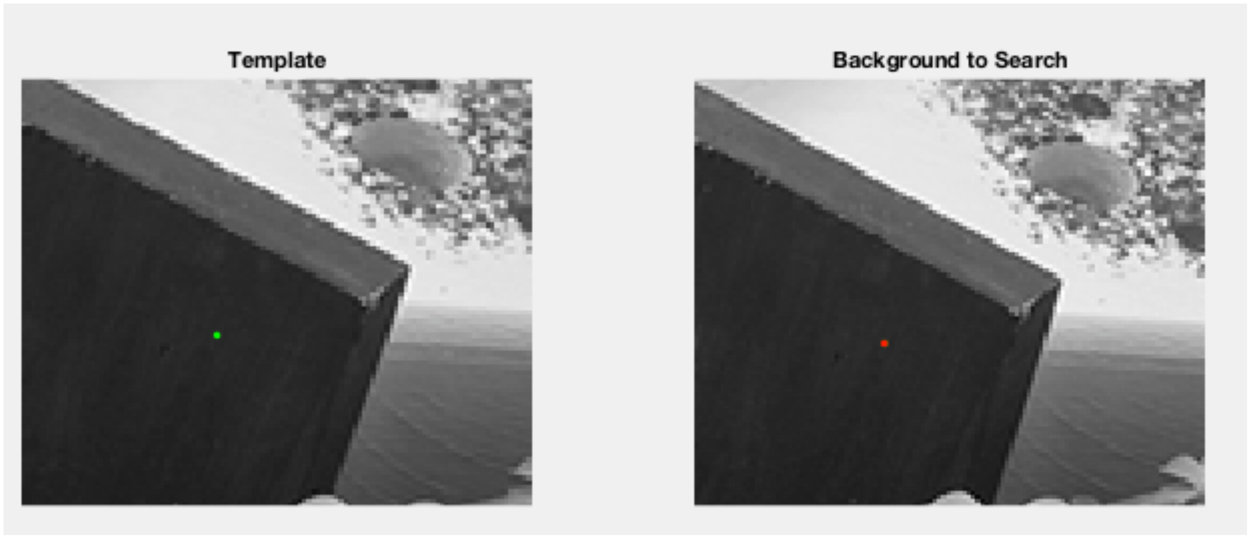

Figure 15. Example of Point Matching with bound optimization, where matched point is overlapped with the ground truth

As shown in **Figure 15**, the green point in the template (left image) is matched with the red point in the background (right image) by LSM without bound optimization. The green point in the background (right image) is the ground truth, which is gained by disparity map provided by Stereo - Middlebury Computer Vision open sources. The green point is unseen in the background because the ground truth point is totally overlapped with the matched point.

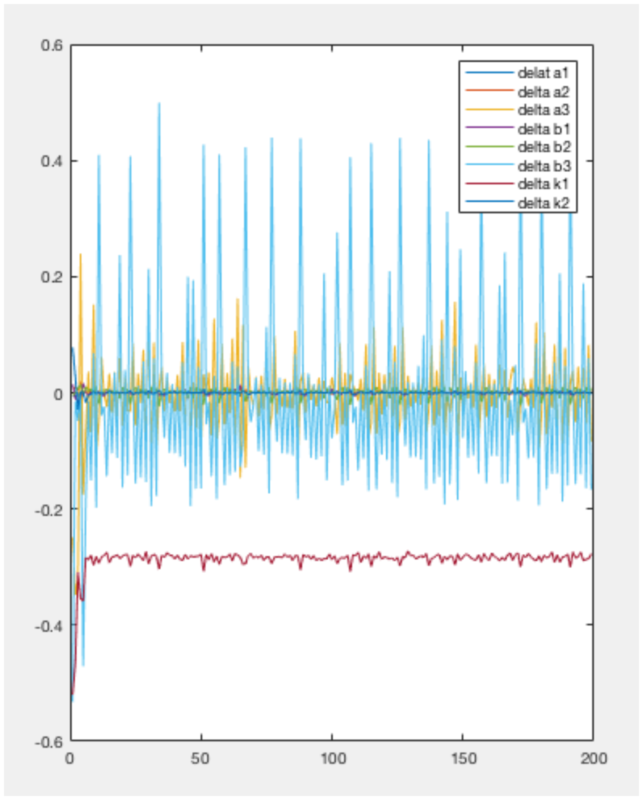

Figure 16. Parameters Adjustment Values of the Point Matching Example using bound optimization

As shown in **Figure 16**, no convergence can be observed in this matching neither. However, in practice this is true when dealing with plain texture area-based point matching. In addition, from the figure, it can be noticed that the k1 parameter controlling illumination stays away from the correct value for every iteration. This phenomenon shows the bounds are working to prevent the parameters go out of reasonable regions. Even every iteration, an unreasonable solution is returned, it has been ignored and that's the reason why the red curve is not converging to zero. a3 and b3 are pivoting back and forth around the ground truth. The distance from the matched point to the ground truth is 0.22 pixels in this case, which is much smaller than without using the bound.

In order to determine whether using bound is helpful on increasing point matching accuracy for the Singapore data set and to reduce bias from any single case, 10 points in the image data sets are randomly selected for point matching with and without using bound constrain optimization. The distances of the matched from the ground truth of all the cases are shown in the below ***Table 2***:

Table 2. *comparison of distances from ground truth of 10 random point matching cases with and without using bound constrain optimization*

points	with bound	without bound
1	1.31	1.68
2	1.48	1.48
3	2.24	2.25
4	2.24	2.87
5	2.52	2.23
6	1.46	1.46
7	2.34	2.19
8	2.43	2.43
9	1.71	1.81

10	1.58	1.58
average	1.931	1.998
standard deviation	0.4639312689	0.4678746271

As recorded in ***Table 2***, the average of the errors and the standard deviation of the errors have been calculated. Both the average error and standard deviation are smaller when using the bound by comparison. Overall, bound optimization is selected for the images data sets covering the main campus of National University of Singapore.

3.3 Three-dimensional Reconstruction Improvement Using Refined LSM

As discussed through the whole thesis, the main objective is to used refined Least Squares Image Matching method to improve 3D reconstruction using blur images. In order to achieve the purpose, two evaluations have been done: evaluating the influence of different degrees of Gaussian and motion blur images on 3D reconstruction accuracy and whether refined LSM will decrease the error when replacing all corresponding points from the sift.

After running four Gaussian blur images data sets with different standard deviations in Apero photogrammetry software and MSP software, four digital surface models (DSM) are generated. Every single pixel in these four DSMs are compared with the ground truth, which is the DSM generated from the clear images. After align different DSM models into the same position, the averages of the absolute difference of all pixels are calculated for four data sets. The result is plotted in the following **Figure 17**.

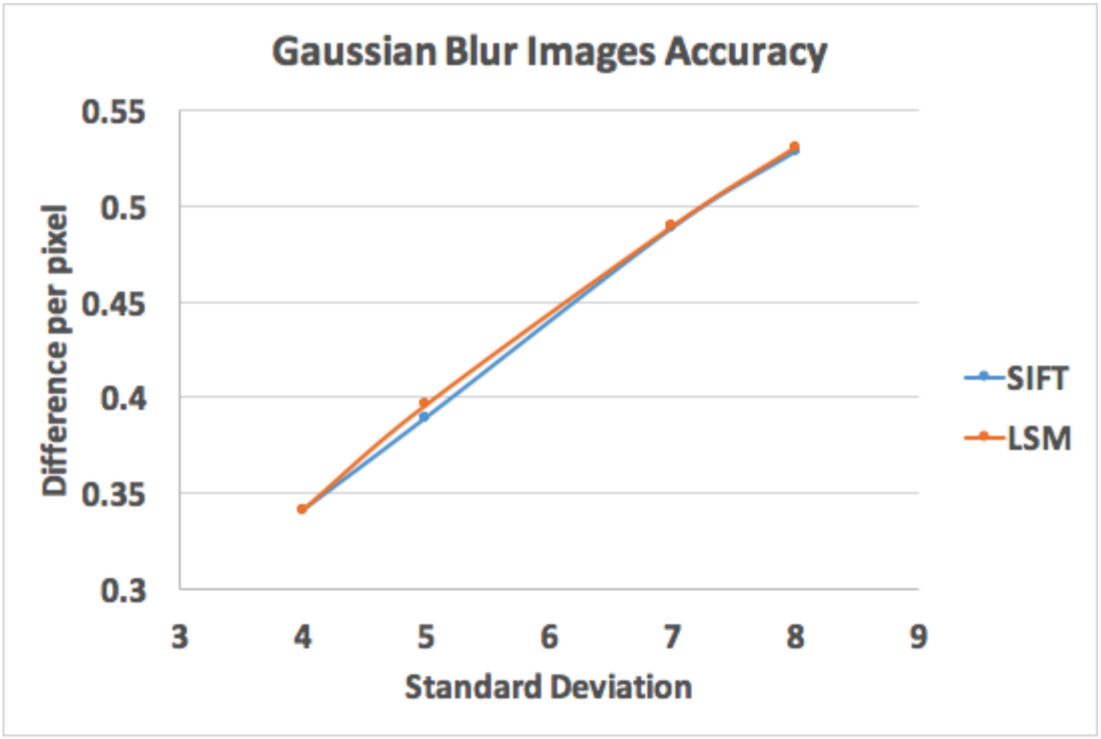

Figure 17. Average error per pixel in DSM of different degrees of Gaussian blur

As shown in **Figure 17**, the blue curve represents the accuracy of DSM using corresponding points from SIFT. It can be observed that the average error increases (or the accuracy decreases) when the blurriness is increased.

Then refined LSM method has been introduced with bound and right template size selection. Just after the first step running the Apero photogrammetry software, which is generating SIFT tie points across all images, refined LSM has been applied to replace all these corresponding points. Then the updated corresponding points are used for later steps to generate new digital surface models. Following the same procedures, the DSM are compared to the ground truth. The new average errors are represented as red curve shown in **Figure 17**. However, for every single data sets, the accuracy of DSM using refined LSM is decreased (or average error is larger). This can conclude that accuracy of 3D reconstruction using Gaussian blur images cannot be optimized by least squares images matching method.

Using the same procedures, four digital surface models using motion blur images with different kernel sizes are generated. Those DSM are compared to the ground truth and the average errors have been calculated. The result is plotted in the following **Figure 18**.

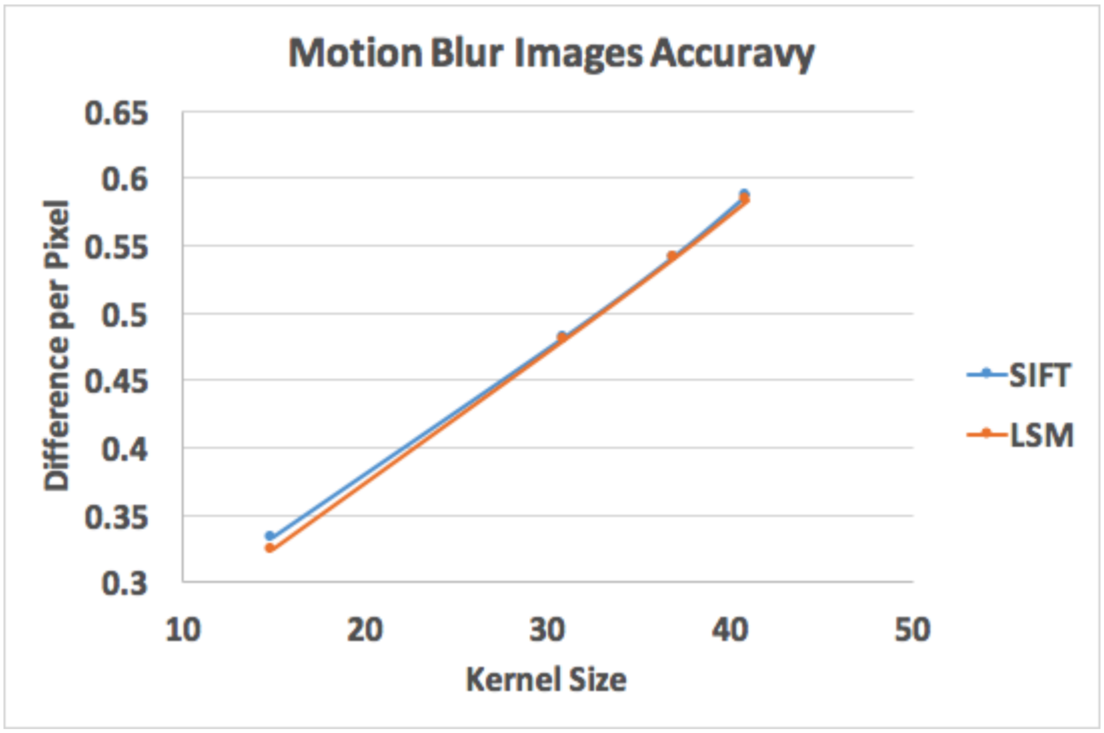

Figure 18. Average error per pixel in DSM of different degrees of Motion blur

As shown in **Figure 18**, the blue curve represents the accuracy of DSM using corresponding points from SIFT. It can be observed that the average error increases (or the accuracy decreases) when the motion blurriness is increased, which is the same to the Gaussian blur images trend.

Then following the same procedures of Gaussian blur image analysis, refined LSM has been applied to replace all corresponding points generated from SIFT. Then new digital surface models are generated to compare to the ground truth. The new average errors are represented as red curve shown in **Figure 18**. It can be observed that, for every single data sets, the accuracy of DSM using refined LSM is increased (or average error is smaller), even though it is slightly. This can conclude that accuracy of 3D reconstruction using motion blur images can be optimized by least squares images matching method.

Chapter 4: Conclusion

4.1 Summary

In conclusion, the classical LSM matching approach has been revisited and the LSM robustness has been improved by modification applying bound constraints optimization and good window size selection. Parameter bounds, constraining the search for a solution to a reasonable region and preventing the parameters for affine deformations and the illuminations from straying from the correct values, will increase the accuracy of LSM matching, even though convergence will not be ensured. A good window size selection is crucial for good point matching. Too small and too large templates are both not optimum for matching accuracy. 31*31 pixels' template is recommended as typical template size. Even though better convergence can be achieve using larger template size, when considering accuracy and calculation speed, smaller template size is more effective.

Furthermore, how photogrammetric 3D reconstruction performs using imageries under suboptimal scenario has been evaluated. 3D reconstruction using Gaussian and motion blurred images with different degrees of blurriness is experimented using Apero photogrammetry software. The accuracy of the bundle adjustment and 3D dense reconstruction using corresponding points from state-of-art point detectors (e.g. SIFT) and corresponding points refined by LSM is compared. Based on the result obtained in the research, SIFT still has high accuracy when used on Gaussian blur images, and LSM modification is not able to enhance the 3D reconstruction accuracy. SIFT has relatively lower accuracy when used on motion blur images, and the matching can be optimized by refined LSM.

4.2 Future work

In this research, classic least square image matching has been refined to increase the robustness using bound constrain optimization and correct template size selection. But there are still spaces for the refined LSM to be more accurate. In this research, fixed bound and fixed template size is utilized for every single point in all data sets. Even though before LSM matching, bias has been reduced by randomly selecting ten points for determining the template size and whether bound optimization would be effective in improving accuracy, bias still exists because different pixels have different local variations of intensity and disparity. In the future research, adaptive method can be introduced to template size and bound setting. The criteria for adjusting the values is the core of future research. As discussed in this thesis, convergence will not always provide good matching result. So the criteria to change the size of template and bound limitation should be other than convergence. If adaptive method can be introduced to every pixel, the bias of evaluation will be dramatically reduced and the matching accuracy will be enhanced.

In addition, the blur images data sets used for 3D reconstruction evaluation are artificially generated. Gaussian blur images data sets are generated by using Gaussian filter. But in the real life, the noises of Gaussian blur images do not perfectly follow Gaussian distribution. Motion blur images data sets are generated by using motion kernel. But in real life, the motion blur is not perfectly linear to a single direction. So in the future research, more complex Gaussian blur and motion blur images data sets can be utilized for the evaluation on refined LSM.

References

[1] Boykov et al., Fast Approximate Energy Minimization via Graph Cuts, International Conference on Computer Vision, September 1999.

[2] D. Scharstein and R. Szeliski. Stereo matching with nonlinear diffusion. International Journal of Computer Vision, 28(2):155-174, July 1998

[3] Gruen, A., & Akca, D. (January 01, 2005). Least squares 3D surface and curve matching. Isprs Journal of Photogrammetry and Remote Sensing, 59, 3, 151-174.

[4] Hassaballah, M., Abdelmgeid, A. A., & Alshazly, H. A. (January 01, 2016). Image Features Detection, Description and Matching.

[5] Hu, H., Ding, Y., Zhu, Q., Wu, B., Xie, L., & Chen, M. (August 01, 2016). Stable least squares matching for oblique images using bound constrained optimization and a robust loss function. Isprs Journal of Photogrammetry and Remote Sensing, 118, 53-67.

[6] ICIAR (Conference), Campilho, A., & Kamel, M. (2012). Image analysis and recognition: 9th International Conference, ICIAR 2012, Aveiro, Portugal, June 25-27, 2012. Proceedings. Berlin: Springer.

[7] Rongjun Qin PhD, "Multi-View Stereo Dense Matching – A Software Package for Dense Point Cloud and DSM Generation from Multi-View Stereo Images." u.osu.edu/qin.324/msp/.

[8] T. Kanade and M. Okutomi, A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment,, Proc. International Conference on Robotics and Automation, 1991.

[9] Wang, M., Zhang, B., & Ning, X. (January 01, 2005). Automatic change detection based on least square image matching [6045-111]. Proceedings- Spie the International Society for Optical Engineering, 6045, 604534.

Appendix

Matlab Source Codes

CrossCorrelation.m

CrossCorrelation.m is the function to find the corresponding position of certain template within the background. The user is responsible for entering the template and the background, and the translation in X and Y direction and the matched image will be returned.

```
function [YTranslation,XTranslation,CoeffMap,ImageCCMathed] =
CrossCorrelation(RawTemplate,RawBackground)

% transform color image to gray image
% do not need it if using gray image
GrayTemplate = rgb2gray(RawTemplate);
% to avoid 255 saturation
Template = im2double(GrayTemplate);

% do the same opration to the background
GrayBackground = rgb2gray(RawBackground);
Background = im2double(GrayBackground);

% get size of template and background
% Cross Correlation Function works only at Background larger than Template
[TemplateY,TemplateX] = size(Template);
[BackgroundY,BackgroundX] = size(Background);

% Calculate the standard deviation of template
TemplatePixelNum = TemplateY * TemplateX;
TemplateSumOfSqr = sum(sum(Template.^2));
TemplateSum = sum(sum(Template));
TemplateStdDeviation = sqrt(TemplateSumOfSqr - TemplateSum^2/TemplatePixelNum) /
(TemplatePixelNum - 1);



for i = 1 : BackgroundY - TemplateY
```

```
  for j = 1 : BackgroundX - TemplateX
     % Calculate standard deviation of background in the area of template at current
offset
     BackgroundSumOfSqr = sum(sum(Background(i : i+TemplateY-1 , j : j+TemplateX-
1) .^2));
     BackgroundSum = sum(sum(Background(i : i+TemplateY-1 , j : j+TemplateX-1)));
     BackgroundStdDeviation = sqrt((BackgroundSumOfSqr -
BackgroundSum^2/TemplatePixelNum) / (TemplatePixelNum - 1));

     % Calculate covariance between template and background in the area of template at
current position
     SumOfMultiply = sum(sum( Background(i : i+TemplateY-1 , j : j+TemplateX-1) .*
Template));
     Covariance = (SumOfMultiply - BackgroundSum * TemplateSum/TemplatePixelNum)
/ (TemplatePixelNum - 1);

     % calculate cross correlation coefficient over all possible locations
     CoeffMap(i,j) = Covariance / (BackgroundStdDeviation * TemplateStdDeviation);
  end
end
%find the max value
[MaxCoeff,MaxCoeffIndx] = max(CoeffMap(:));
[YTranslation,XTranslation] = ind2sub(size(CoeffMap),MaxCoeffIndx);

XList = [1:TemplateX];
XIndexList = repmat(XList,TemplateY,1);% x coordinates
XOldIndex = XIndexList(:);% x coordinates in index
Xnew = XOldIndex + XTranslation;% get new x coordinate

YList = [1:TemplateY]';
YIndexList = repmat(YList,1,TemplateX);% y coordinates
YOldIndex = YIndexList(:);% y coordinates in index form
Ynew = YOldIndex + YTranslation;% get new y coordinate


% convert row and column subscripts to index
IndxNew = Ynew + (Xnew-1) * BackgroundY;

% get new intensity values
IndxNew2 = reshape(IndxNew,size(Template));
ImageCCMathed = Background(IndxNew2);% convert index to subscripts
end
```

LSMMatching.m

LSMMatching.m the function to find the corresponding point of center of the template within the background. The user is responsible for entering the template and the background, and initial coordinate solution in the background. The new position of the corresponding point in the background will be returned. The delta values used for ensuring convergence will be returned as well. Bound optimization algorithm is included.

```
function [xNew, yNew,  delta, ImageLSMMatched, Template, a1,a2,a3,b1,b2,b3,k1,k2] =
LSMMatching(xCor, yCor, xInitialGuess, yInitialGuess, RawTemplate, RawBackground)

% transform color image to gray image
% do not need it if using gray image
GrayTemplate = rgb2gray(RawTemplate);
% to avoid 255 saturation
Template = im2double(GrayTemplate);
windowSize = 31;
w = (windowSize + 1)/2;
rect=[xCor-w yCor-w windowSize-1 windowSize-1];
template=imcrop(Template,rect);

% do the same opration to the background
GrayBackground = rgb2gray(RawBackground);
Background = im2double(GrayBackground);


% get size of template and background
% Cross Correlation Function works only at Background larger than Template
[TemplateY,TemplateX] = size(template);
[BackgroundY,BackgroundX] = size(Background);

a1 = 1;
a2 = 0;
a3 = xInitialGuess - w;
v = a3;
b1 = 0;
b2 = 1;
b3 = yInitialGuess - w;
```

```
u = b3;
k1 = 1;
k2 = 0;%initial guess

XList = 1:TemplateX;
XIndexList = repmat(XList,TemplateY,1);% x coordinates
XOldIndex = XIndexList(:);% x coordinates in index

YList = [1:TemplateY]';
YIndexList = repmat(YList,1,TemplateX);% y coordinates
YOldIndex = YIndexList(:);% y coordinates in index form

k=1;% num of iteration
while k<50
   % new x and y coordinates
   Xnew = round(a1*XOldIndex + a2*YOldIndex + a3);
   Ynew = round(b1*XOldIndex + b2*YOldIndex + b3);


   % convert row and column subscripts to index
   Indnew = Ynew + (Xnew-1) * BackgroundY;



   Inew2 = k1 * Background(Indnew) + k2;% get new intensity values

   % convert index to subscripts
   ImageLSMMatched = reshape(Inew2,size(template));
   xGradient = gradient(ImageLSMMatched);
   xGradientIndx = xGradient(:);% x gradient

   yGradient = gradient(ImageLSMMatched')';
   yGradientIndx = yGradient(:);% y gradient

   % partial derivative of eight parameters
   J1 = - k1 * xGradientIndx .* XOldIndex;% dF/da1
   J2 = - k1 * xGradientIndx .* YOldIndex;% dF/da2
   J3 = - k1 * xGradientIndx;% dF/da3
   J4 = - k1 * yGradientIndx .* XOldIndex;% dF/db1
   J5 = - k1 * yGradientIndx .* YOldIndex;% dF/db2
   J6 = - k1 * yGradientIndx;% dF/db3
   J7 = - ImageLSMMatched(:);% dF/dk1
   J8 = - ones(1,TemplateX * TemplateY)';% dF/dk2
   J = [J1,J2,J3,J4,J5,J6,J7,J8];% Jacobian Matrix
   N = J'*J;% Normal Matrix
   b = template - ImageLSMMatched;
   O = -b(:);

   B = J'*O;% B matrix

   delta(:,k)=N\B; %delta values on parameters

% bound restriction optimizationdelta=N\B;
   aa = a1 + delta(1,k);
   if aa > 0.5 && aa < 1.5
      a1 = aa;
   end

   ab = a2 + delta(2,k);
   if ab > -0.5 && ab < 0.5
      a2 = ab;
   end

   ac = a3 + delta(3,k);
   if ac > v-40 && ac < v+40
      a3 = ac;
   end

   ba = b1 + delta(4,k);
   if ba > -0.5 && ba < 0.5
      b1 = ba;
   end

   bb = b2 + delta(5,k);
   if bb > 0.5 && bb < 1.5
      b2 = bb;
   end

   bc = b3 + delta(6,k);
   if bc > u-40 && bc < u+40
      b3 = bc;
   end

   ka = k1 + delta(7,k);
```

```
    if ka > 0.5 && ka < 1/0.5
        k1 = ka;
    end

    kb = k2 + delta(8,k);
    if kb > -10 && kb < 10
        k2 = kb;
    end

k = k + 1;% count the iteration number
end

Xnew = round(a1*XOldIndex + a2*YOldIndex + a3);
Ynew = round(b1*XOldIndex + b2*YOldIndex + b3);
% new x and y coordinates


% convert row and column subscripts to index
Indnew = Ynew + (Xnew-1) * BackgroundY;


Inew2 = k1 * Background(Indnew) + k2;% get new intensity values

ImageLSMMathed = reshape(Inew2,size(template));


xNew = a1*w + a2*w + a3;
yNew = b1*w + b2*w + b3;

end
```

GaussianBlur.m

GaussianBlur.m is the function to artificially generate Gaussian blur images with a typical standard deviation set by the developer.

```
% blue the image

for j = 0:24
   % read the old images
   if j <= 7
   imgNameTemplate =
sprintf('/Users/hxy/Downloads/imgs/8%d_swimming_pool_S_DSC0010%d.jpg',j+10,j+2);
   RawTemplate = imread(imgNameTemplate);
   % set the degree of the blur
   blurImage = imgaussfilt(RawTemplate,6);
   % generate the new blur images
   NewimgName =
sprintf('/Users/hxy/Downloads/imgs/Gaussian6_8%d_swimming_pool_S_DSC0010%d.tif',
j+10,j+2);
   imwrite(blurImage, NewimgName);
   else
      imgNameTemplate =
sprintf('/Users/hxy/Downloads/imgs/8%d_swimming_pool_S_DSC001%d.jpg',j+10,j+2);
   RawTemplate = imread(imgNameTemplate);
   % set the degree of the blur
   blurImage = imgaussfilt(RawTemplate,6);
   % generate the new blur images
   NewimgName =
sprintf('/Users/hxy/Downloads/imgs/Gaussian6_8%d_swimming_pool_S_DSC001%d.tif',j
+10,j+2);
   imwrite(blurImage, NewimgName);

   end
end
```

MotionBlur.m

MotionBlur.m is the function to artificially generate motion blur images with a typical sized kernel set by the developer.

```
motionmask = [1 0 0 0 0
            0 1 0 0 0
            0 0 1 0 0
            0 0 0 1 0
            0 0 0 0 1
            ]/5;
% Motion blur the image

for j = 0:24
   % read the old images
   if j <= 7
   imgNameTemplate = sprintf('8%d_swimming_pool_S_DSC0010%d.jpg',j+10,j+2);

   RawTemplate = imread(imgNameTemplate);
   % set the degree of the blur
   ImotionBlur = imfilter(RawTemplate,motionmask);
   % generate the new blur images
   NewimgName =
sprintf('MotionBlurDia5_8%d_swimming_pool_S_DSC0010%d.jpg',j+10,j+2);
   imwrite(ImotionBlur, NewimgName);
   else

   imgNameTemplate = sprintf('8%d_swimming_pool_S_DSC001%d.jpg',j+10,j+2);

   RawTemplate = imread(imgNameTemplate);
   % set the degree of the blur
   ImotionBlur = imfilter(RawTemplate,motionmask);
   % generate the new blur images
   NewimgName =
sprintf('MotionBlurDia5_8%d_swimming_pool_S_DSC001%d.jpg',j+10,j+2);
   imwrite(ImotionBlur, NewimgName);
   end

end
```

----

textRead.m

textRead.m is the function to read all pairs of corresponding points and call the function of least square matching to replace all those points. These original points are the initial guess for the LSMMatching.m.

```
clc
clear
% to read a text file
for j = 0:24 % j represent the folder name number
   if j <= 7
      imgNameTemplate =
sprintf('/Users/hxy/Downloads/imgs/Gaussian6_8%d_swimming_pool_S_DSC0010%d.tif',
j+10,j+2);
   elseif j <= 99
      imgNameTemplate =
sprintf('/Users/hxy/Downloads/imgs/Gaussian6_8%d_swimming_pool_S_DSC001%d.tif',j
+10,j+2);
   end
   j
   for i = 0:24
      i
      % write the text file name containing pts info
      % write the image name
      % write the new file name after LSM optimization
      if i <= 7
         imgNameBackground =
sprintf('/Users/hxy/Downloads/imgs/Gaussian6_8%d_swimming_pool_S_DSC0010%d.tif',
i+10,i+2);
      elseif i <= 99
         imgNameBackground =
sprintf('/Users/hxy/Downloads/imgs/Gaussian6_8%d_swimming_pool_S_DSC001%d.tif',i
+10,i+2);
      end

      if i <= 9 && j <= 9
         ptsFileName =
sprintf('/Users/hxy/Downloads/g6/Before_LSM/Homol/PastisIMG_0%d.tif/IMG_0%d.tif.t
xt',j,i);
         newPtsFileName =
sprintf('/Users/hxy/Downloads/g6/After_LSM/Homol/PastisIMG_0%d.tif/IMG_0%d.tif.txt
',j,i);
    elseif i <= 9 && j > 9
        ptsFileName =
sprintf('/Users/hxy/Downloads/g6/Before_LSM/Homol/PastisIMG_%d.tif/IMG_0%d.tif.txt
',j,i);
        newPtsFileName =
sprintf('/Users/hxy/Downloads/g6/After_LSM/Homol/PastisIMG_%d.tif/IMG_0%d.tif.txt',j
,i);
    elseif i > 9 && j <= 9
        ptsFileName =
sprintf('/Users/hxy/Downloads/g6/Before_LSM/Homol/PastisIMG_0%d.tif/IMG_%d.tif.txt
',j,i);
        newPtsFileName =
sprintf('/Users/hxy/Downloads/g6/After_LSM/Homol/PastisIMG_0%d.tif/IMG_%d.tif.txt',j
,i);
    elseif i > 9 && j > 9
        ptsFileName =
sprintf('/Users/hxy/Downloads/g6/Before_LSM/Homol/PastisIMG_%d.tif/IMG_%d.tif.txt',
j,i);
        newPtsFileName =
sprintf('/Users/hxy/Downloads/g6/After_LSM/Homol/PastisIMG_%d.tif/IMG_%d.tif.txt',j,i
);
    end


    %check if file exists
    if exist(ptsFileName, 'file') == 2
        ptsMatrix = textread(ptsFileName);

    % check how many lines should the algorithm read
    [numOfPts,b]= size(ptsMatrix);

    % read the template image and background image
    template = imread(imgNameTemplate);
    background = imread(imgNameBackground);
    % read everyline of pts
    count = 1;
    while count <= numOfPts
        xCor = ptsMatrix(count,1);
        yCor = ptsMatrix(count,2);

        xInitialGuess = ptsMatrix(count,3);
        yInitialGuess = ptsMatrix(count,4);
```

```
          % generate the new coordinate
          % call the function
         if xCor > 200 && yCor > 200 && xInitialGuess > 200 && yInitialGuess > 200 && xCor
< 4400 && xInitialGuess < 4400 && yCor < 2850 && yInitialGuess < 2850

             [xNew, yNew] = LSMMatching(xCor, yCor, xInitialGuess, yInitialGuess, template,
background);
          % write the new coordinate into new matrix
             if abs(xNew - xInitialGuess) < 3 && abs(yNew - yInitialGuess) < 3
                NewptsMatrix(count,1) = xCor;
                NewptsMatrix(count,2) = yCor;
                NewptsMatrix(count,3) = xNew;
                NewptsMatrix(count,4) = yNew;

             else
                NewptsMatrix(count,1) = xCor;
                NewptsMatrix(count,2) = yCor;
                NewptsMatrix(count,3) = xInitialGuess;
                NewptsMatrix(count,4) = yInitialGuess;
             end
          else
             NewptsMatrix(count,1) = xCor;
                NewptsMatrix(count,2) = yCor;
                NewptsMatrix(count,3) = xInitialGuess;
                NewptsMatrix(count,4) = yInitialGuess;
          end
          count = count + 1;
       end

       fileID = fopen(newPtsFileName, 'wt+');
       count = 1;
       while count <= numOfPts
           fprintf(fileID,'%f %f %f %f\n',NewptsMatrix(count,:));
          count = count + 1;
       end

       fclose(fileID);

       end
    end



end
```