Development of an Optimised Photogrammetry Protocol

Jill Ueng, CID: 01119321, MEng Final Year Project

Supervisors: Dr. Huai-Ti Lin, Dr. Maria Jose Fernandez Jauregui

***Abstract*—Photogrammetry is a cost-effective and widely accessible technique used to generate accurate 3D digital models of real-world surfaces through interpretation of 2D data, such as images. Inaccurate models reconstructed using photogrammetry are often due to the usage of low quality image datasets. This project aims to optimise photogrammetry for scientific applications by providing known uncertainties and outline useful assistive software to minimise inaccuracies. Varying different parameters on image conditions was found to have a noticeable effect on model accuracy. A range of optimal conditions was determined by characterising reconstruction error, while specific parameter values for individual contexts can be found by examining feature matching success rates in the early stages of the photogrammetry pipeline.**

I. Introduction

THE process behind photogrammetry can be divided into three main sections: image acquisition, sparse point cloud generation, and dense point cloud generation. A point cloud refers to a dataset of points in 3D space. A sparse point cloud consists of data points forming a preliminary scene reconstruction based off corresponding features in the images, along with estimates of the camera poses, while a dense point cloud consists of more points corresponding to the scene geometry (1). Understanding the steps behind image acquisition and sparse point cloud generation are crucial towards the development of this project; thus, they will be delved into in great detail here. As the developed desktop application is not intended to generate the complete 3D model by itself, the generation of the dense point cloud will only be briefly touched upon.

The first step towards acquiring a 3D model from a real life object or environment is to acquire a set of images. Images may be taken using two methods, the first of which involves keeping the subject of interest static and physically moving the camera around it to take images from every angle possible. The second method places the subject on a turntable, against a neutral (e.g. solid white) background. The camera position is kept the same as images of the subject, covering various angles, are taken by rotating the turntable (2). To increase the accuracy of the feature extracting matching algorithms, the images should be taken under bright and even illumination conditions with the whole subject in focus at all times, with high visibility of physical details. There should be high visual overlap between consecutively taken images (3). A static background is preferred over a dynamic one, and the subject should be contrasted against the background.

A. Structure-from-Motion

Structure-from-Motion (SfM) is a widely used technique in photogrammetry used to generate a sparse point cloud given a set of images. SfM first performs feature extraction, then simultaneously computes feature matching and camera calibration (4). Feature extraction may be performed using Scale Invariant Feature Transform (SIFT), a highly robust algorithm well established in the field of computer vision. The features extracted using SIFT are invariant to changes in scale, orientation, and illumination in the images, meaning a feature in one image can be successfully matched to features in another images, even if any of those parameters change (5). The four main computational steps involved are as follows:

1) Scale-space extrema detection: Detects potential interest points invariant to scale and orientation using a difference-of-Gaussian function.
2) Keypoint localization: Model fitted at each point to determine location and scale, selected as a keypoint based on stability measurements.
3) Orientation assignment: Assigned to each keypoint location based on local image gradient directions, providing invariance to changes in orientation, scale, and location, as for then onwards any future operations are done relative to the established conditions.
4) Keypoint descriptor: The local image gradients around each keypoint are measured then transformed into a representation robust to shape distortion and illumination changes.

In order to detect scale-invariant extrema as potential keypoints, each image must first be represented in scale space, allowing SIFT to search across as many possible scales. Each image will be extended to a set of octaves, where a single octave contains some number of scales. The first octave holds images of the same dimension as the original, the second holds images downsampled by a factor of 2 from the previous octave, and so forth. The first scale of an octave is the image in question without any filter over it, the second scale will be the image with Gaussian blur applied, the next scales then contain increasingly blurred images. Equations 1 and 2 show the operation used to generate the different scales, where convolution of the Gaussian operator (G) and the image (I) produces a blurred image (L), The pixel coordinates are represented by (x, y), and indicates the scale (blur) parameter.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

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

Within each octave, consecutive scales are subtracted from each other to generate the Difference-of-Gaussian images, which are an approximation of the Laplacian of Gaussian images, as illustrated in Figure 1 (5). To determine the extrema, SIFT iterates through every pixel in each octave and compares it to its 26 neighbouring pixels in the same scale, the one above, and the one below. If the pixel is greater than or smaller than all of its neighbours, then it is marked as a keypoint. As the true local maximum or minimum tends to be between pixels, Taylor expansion around the marked pixel is used to find the location of the true subpixel extrema.

To perform keypoint localisation, keypoints that are in low contrast to their surroundings or lie along an edge must first be filtered out due to their instability. Low contrast keypoints can easily filtered out by setting a threshold for the minimum value in the difference in intensity between the keypoints and its surroundings. At the remaining keypoints, two gradients perpendicular to each other are calculated and the ratio between the two is checked using the Hessian matrix. If the ratio is greater than a certain value, the keypoint must lie along an edge and is then rejected (5).

Orientation is then assigned to the remaining keypoints by calculating the gradients and their magnitudes around each one using Equations 3 and 4, then assigning the most prominent surrounding orientation (gradient) to each one. The size of the region used to calculate the gradients is directly proportional to the scale the keypoint is located on (5).

$$m(x,y)=\sqrt{(L(x+1,y)-L(x-1,y))^2+(L(x,y+1)-L(x,y-1))^2} \quad (3)$$

$$\theta(x,y) = tan^{-1}\frac{L(x,y+1)-L(x,y-1)}{L(x+1,y)-L(x-1,y} \quad (4)$$

Finally, a feature vector containing 128 values will be generated for each of the remaining keypoints. A 16x16 window of in-between pixels is formed around each keypoint, which is then split into sixteen 4x4 windows. A histogram of 8 bins corresponding to a different range of angles is generated from each 4x4. Gradient orientations from each 4x4 are placed into their corresponding bins. 128 values are then extracted and normalised from each bin across the entire 16x16 window. The keypoint is made rotation independent by subtracting its own orientation value from each value in the feature vector, and illumination independent by setting a threshold of 0.2 for each value in the feature vector. The value of 0.2 was determined empirically by Lowe through experimentation using a set of images with differing illumination conditions for the same subject (5). The culmination of these four major steps results in a SIFT feature, which mathematically comprises of its location (x, y), scale (relevant size of the keypoint, related to the scale it was first detected on), orientation, and the 128-value vector (5).

After features are extracted and stored in a new database, pairs of images are tested for scene overlap. New images are then matched by individually comparing features from the new images to those already in the database. This is done based on Euclidean distance of their feature vectors, and computed through fast near-neighbour algorithms (5). The output of this step consists of a set of potential image pairs with high scene overlap and their corresponding putative feature matches. Outlying matches from this stage are filtered out by iteratively applying geometric constraints upon the putative matches, such as through epipolar geometry mapping. False matches are discarded through filtering by random sample consensus (4).

The final stage in the generation of the sparse point cloud consists of calibrating the camera poses and the incremental preliminary scene reconstruction, that is, populating the point cloud with matched features. The sparse point cloud model is initialised using a pair of verified images. New images are registered to the model one at a time based off correspondence with already registered images, increasing the number of points in the model by triangulation. This step outputs camera pose estimates along with the location of features keypoints in 3D space. To refine the pose estimates and 3D position of observable points, a non-linear technique known as bundle adjustment, which minimises the reprojection error, is used. The equation for bundle adjustment is shown in Equation 5. The sparse point cloud at this point is complete (4).

$$E = \sum_j \rho_j \|\pi((P)_c, \mathbf{X}_k) - x_j\|_2^2 \quad (5)$$

The generation of the dense point cloud uses a technique known as Multi-view Stereo (MVS). MVS takes the camera pose estimates from the sparse point cloud model, computes the depth and normal information for every pixel in the image set and fuses that information to produce the dense point cloud (3). A triangular mesh can be created by using a surface reconstruction algorithm such as the Screened Poisson Surface Reconstruction (3). The meshed 3D model is then ready for post-processing.

B. Accuracy Evaluation of Photogrammetric Reconstruction

The success of a photogrammetric reconstruction may be judged qualitatively by eye; however, for scientific applications, it is crucial to determine and define objective uncertainties. The image acquisition stage of photogrammetry offers the highest degree of control over the final 3D output, so it is there this project attempts to determine the conditions that will lead to the greatest model accuracy.

Applying image processing filters over a set of thermal images, such as channel-based local histogram equalisation and high pass filters have been found to feature matching accuracy (6). While this provides a guideline for desirable image conditions, as the experiment was performed on a pair of low spatial and radiometric resolution thermal images, without the generation of a 3D model, it does not provide evidence towards the success of a photogrammetric reconstruction. Similarly, a image post-processing pipeline consisting of exposure equalisation, denoising, and adaptive median filtering has been shown to increase the number of features extracted and matched while decreasing the bundle adjustment reprojection error (7). Acquiring images through pre-processing such as the use of polarising filters and High Dynamic Range tone mapping will also increase the number of feature matches (6). But though these experiments provides a useful guideline, as

the success of the reconstruction was evaluated using values inferred from the photogrammetric algorithm, it does not define an empirical performance metric. One such empirical performance metric was devised by examining reconstruction artefacts using a nearest neighbour distance measure, however, this method is only able to quantify local errors (7).

This project proposes a novel approach to quantifying the success of photogrammetric reconstruction through obtaining ground truth validation. By characterising reconstruction error as a result of varying image conditions, the accuracy of the model can be directly verified. The range of optimal parameters for a given photogrammetry subject can be then be determined through an automated process.

II. Methodology

A. Ground Truth Validation

Image sets of a calibration object were acquired using an automated image capture camera rig, which consisted of an industrial standard machine vision camera and a servomotor-operated turntable, allowing for precise control over angular position. Diffuse illumination was provided by bouncing the light from a LED bicycle torch off a white paper backdrop encasing the automated rig on three sides. The calibration object used was a 3D-printed cuboid of known dimensions, placed in the centre of the turntable. Images of the cuboid were automatically captured by the rig after performing a rotation at an angle set by the user. Six initial datasets of the cuboid were obtained by varying the angle between consecutive camera views or images, as demonstrated in Fig. 1.

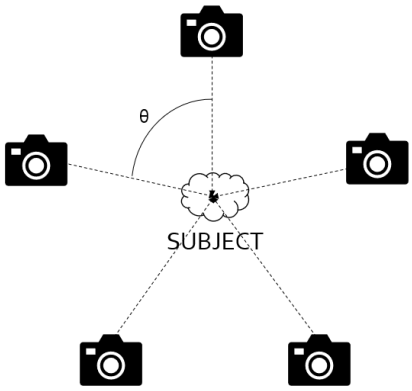

Fig. 1: A basic photogrammetry setup. θ indicates the angle between consecutive camera views (or images).

The following image filters and processing operations, chosen by their ability to modulate conditions that affect model quality, were then applied onto one of the six initial datasets:

1) Low pass convolution filter, or Gaussian Blur, for denoising, varying kernel size
2) High pass convolution filter, or deconvolution, for sharpening/detail enhancement, varying kernel size
3) Downsampling image resolution, varying the downsampling factor
4) Contrast-limited adaptive histogram equalisation (CLAHE), varying the clip limit

Photogrammetric reconstruction was performed using Meshroom, an open-source software. The optimal parameter value was determined by that which resulted in the lowest measured reconstruction error. Reconstruction error here is defined in Equation 6.

$$\%E = \frac{distance_{known} - distance_{measured}}{distance_{known}} * 100 \quad (6)$$

Various output values from the SfM pipeline, corresponding to the first three consecutive images used to initialise the sparse point cloud, were also recorded to search for potential correlation: the mean features extracted, the mean putative feature matches, and the mean geometric feature matches.

To demonstrate the feasibility of using ground truth data for a given photogrammetry rig to set the optimal image parameters, the same filters and processing operations, with values determined from the cuboid experiments, were performed on an image dataset of a different subject, a dragonfly specimen. This image dataset was obtained using the same photogrammetry rig described above. Two distinct markers placed on the dragonfly specimen provided the dimension to measure the reconstruction error. The reconstruction error of the unfiltered image dataset was compared to that of image datasets with different filters applied.

B. Real Time Feature Matching Application

A real-time feature matching application was written in Python with extensive usage of the OpenCV library. The base function of this feature matching application is to guide the user through the image acquisition stage of the photogrammetry pipeline, through the indication of a sufficient number of feature matches between consecutive views of the subject to be reconstructed. The threshold for a sufficient number of matches was to be determined through ground truth validation. Additional functions to be implemented are as follows:

1) Complete automation of image acquisition by:
    a) Connection to the photogrammetry rig used for this project; servomotor set to rotate continuously rather than in discrete steps
    b) Automatically capture images at the given match percentage threshold (as the subject rotates with the servomotor, the number of matches will decrease given the differing camera views of the subject)
    c) Automatic termination once a full rotation is complete
    d) Automatically determine and apply the optimal image parameter values after capturing an initial pair of consecutive images
2) Real time control over all camera settings, such as exposure, gain, and white balance
3) Real time indication of poor image conditions, such as uneven illumination, motion blur, and inappropriate depth of field for the subject
4) Option for manual post-processing of captured images

The full script for the implemented real-time feature tracking application can be found in *Appendix B*.

III. Results

A 3D printed cuboid with known x, y, and z dimensions was used as a calibration object. Through preliminary trials, the reconstructed model was noted to be slightly skewed, that is, lengths in the same dimension were not parallel. Therefore, two lengths in each dimension were considered while obtaining ground truth validation, as labeled in Fig. 2 As the units of the reconstructed model were arbitrarily set by Meshroom, all of the measured lengths were normalised to y_1 in order to make comparisons using real world units.

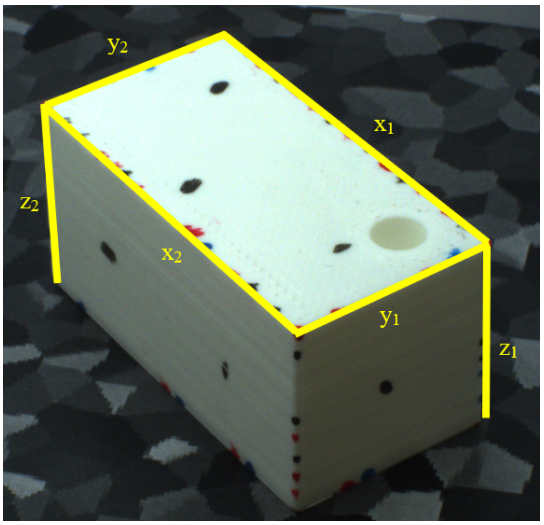

Fig. 2: 3D printed cuboid labelled with the lengths to be measured in the reconstructed model.

A. Angle between Consecutive Views

In general, the reconstruction error increases as the angle between consecutive camera views increases, as shown in 3. Angles of 3° , 6° , 9° , 12° , 15° , and 18° were used. Past a certain angle threshold, found to be in the range of 18-24°, photogrammetry software fails to reconstruct a 3D model entirely.

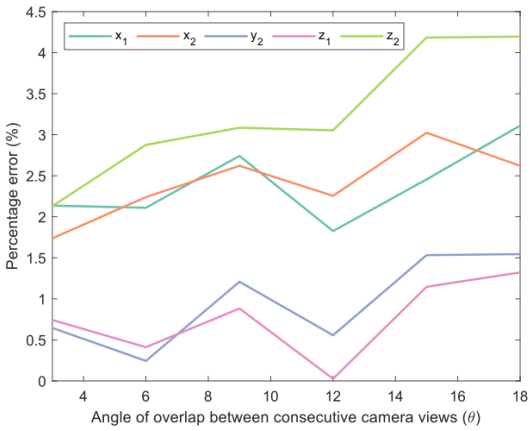

Fig. 3: Reconstruction error when varying the angle of overlap.

B. Downsampling Image Resolution

Downsampling image resolution decreases reconstruction error across all dimensions until the resolution of the image set reaches 70% of the original resolution (2048 x 1536) and from there a general trend of increased reconstruction error can be observed, as in Fig. 4.

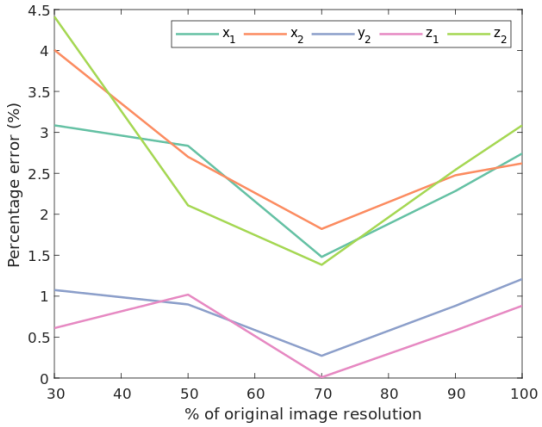

Fig. 4: Reconstruction error with downsampled image resolution.

The mean extracted features of each of the first three consecutive images from each dataset in this section were recorded, along with the percentage of features used for matching out of all extracted features, the percentage of inlying matches out of those (i.e. the percentage of geometrically verified putative matches). The percentage of features matched appears to increases with downsampling, while the percentage of inliers increases in general with a peak, and the mean extracted features of the three initial consecutive images decreases with downsampling, as shown in Fig. 5 and Fig. 6.

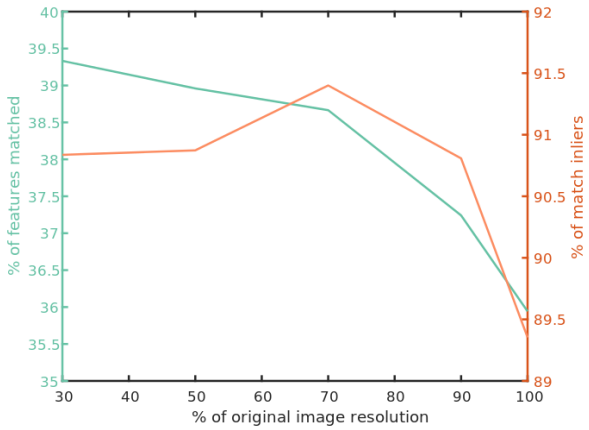

Fig. 5: Percentage of feature matches and match inliers with downsampled image resolution.

C. Image Convolution Filters

Increasing levels of Gaussian blur were applied onto the 9° image set from the previous step using a low pass convolution

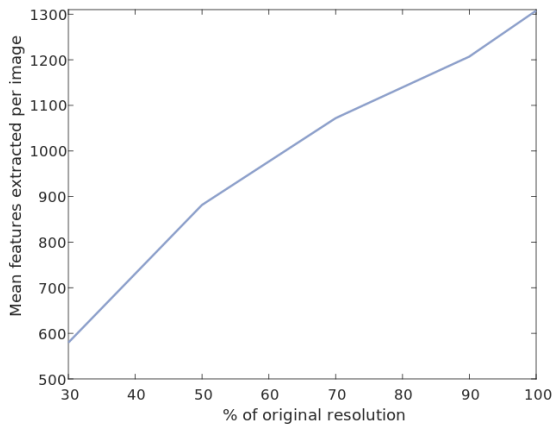

Fig. 6: Mean extracted features per image with downsampled image resolution

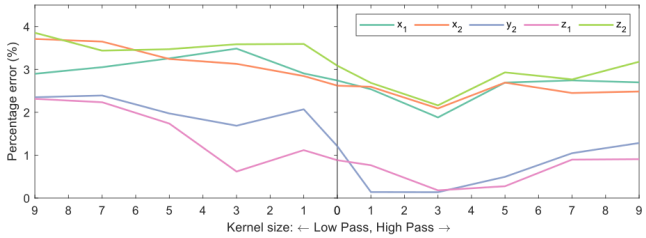

Fig. 7: Reconstruction error with applied image convolution filters.

filter. The kernel size of the convolution filter increases the size of the filter, and thus the perceived strength of the effect. Similarly, a sharpening or detail enhancement effect was applied onto the image set using a high pass convolution filter, while varying kernel size. Kernel sizes of 1, 3, 5, 7, and 9 were used. As shown in 7, increasing the kernel size of the low pass filter does not improve the reconstruction accuracy nor significantly increase the error, while using a kernel size of 3 for the high pass filter noticeably lowers the reconstruction error. The reconstruction error again increases when the kernel size of the high pass filter exceeds 3.

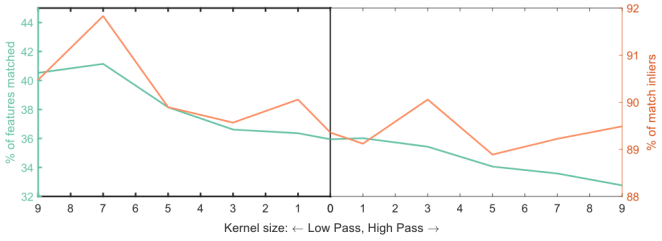

Fig. 8: Percentage of feature matches and match inliers with applied convolution filters

Fig. 8 shows that while the percentage of matched features and the percentage of inliers both decrease as the filter crosses over from low pass to high pass (i.e. blurred to sharpened), the percentage of inliers has multiple local maxima. Conversely, the mean extracted features for each image increases steadily

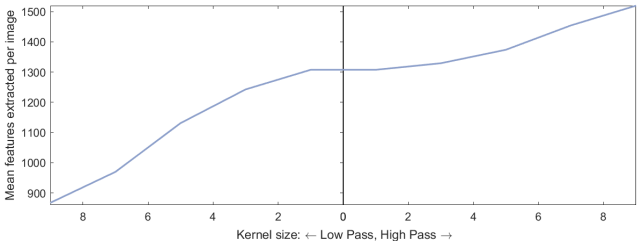

Fig. 9: Mean extracted features per image with applied image convolution filters.

across the same spectrum (i.e. from blurred to sharpened), demonstrated in Fig. 9.

D. Contrast-Limited Adaptive Histogram Equalisation

Contrast-limited adaptive histogram equalisation (CLAHE) is an image processing technique used to improve local contrast and enhance edges. The level of contrast can be adjusted by varying the clip limit, which is a parameter that prevents contrast amplification in homogeneous areas of an image (ref). Increasing the clip limit increases the perceived change in contrast (i.e. detail amplification in near-homogeneous areas).

In the image sets of the cuboid, the cuboid was already in stark contrast to its surroundings, one of the main factors influencing the quality of a reconstruction. The cuboid itself is a plain, homogeneous object with details hidden on its faces in the images, thus applying CLAHE on these image sets would only amplify noise in the homogeneous regions of the cuboid, decreasing model quality. Therefore, a dragonfly specimen was chosen to be the subject for the application of CLAHE, due to the many small, protruding details present in its anatomy. Intuitively, a dragonfly model would benefit from CLAHE as improving contrast would help prevent the subject from blending into the background.

The distances between the outer edges and the inner edges of two white markers on the dragonfly, shown in Fig. 10 were measured with calipers. The units of the measurements on the reconstructed model were normalised to the outer distance, and the inner distance was used to quantify the reconstruction error.

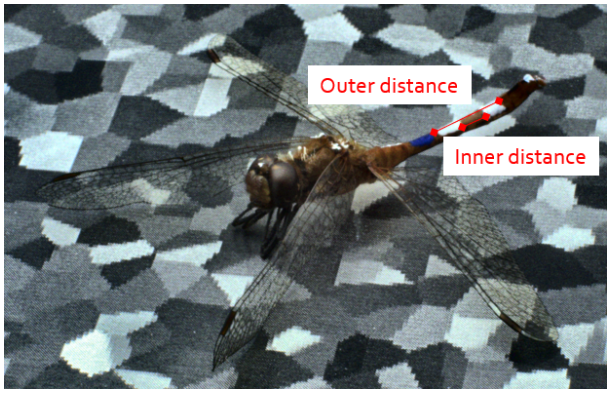

Fig. 10: Dragonfly specimen with two markers a known distance apart.

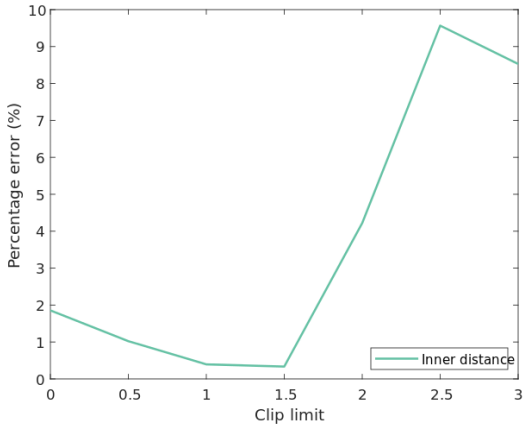

Fig. 11: Reconstruction error with adjusted contrast.

Fig. 11 shows the reconstruction error for the dragonfly subject with CLAHE applied. The reconstruction error decreases until it reaches a minimum at clip limit = 1.5, and from there it greatly increases.

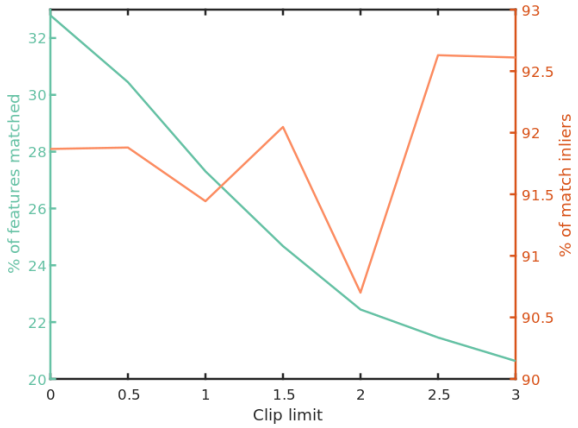

Fig. 12: Percentage of feature matches and match inliers with adjusted contrast.

Again, Fig. 12 demonstrates that the percentage of features matched decreases with clip limit, while the percentage of inliers has multiple local peaks. Fig. 13 shows that the mean extracted features increases with clip limit, as an improvement in contrast results in amplification of details in homogenous regions, leading to a higher number of extracted features. This is confirmed as the SIFT algorithm discards keypoints in low contrast areas due to their instability.

E. Proof of Concept

To demonstrate that the parameter values corresponding to the lowest reconstruction error of the cuboid can also be applied, with similar success, to different subjects given the same photogrammetry rig, various filters were applied onto the image set of the dragonfly specimen using values determined from the cuboid experiments.

Table I shows the reconstruction error for the dragonfly specimen under different image conditions. The base error

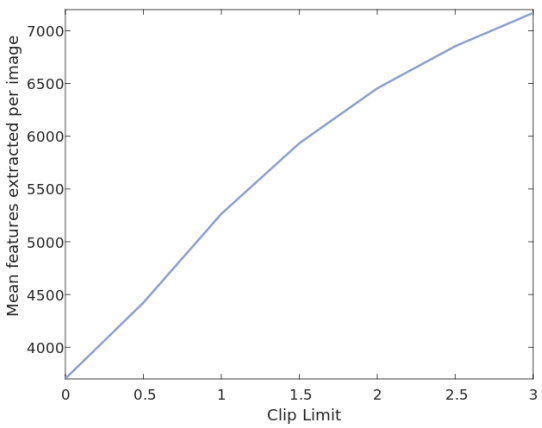

Fig. 13: Mean extracted features per image with adjusted contrast.

Version	% Error
Unfiltered	1.85%
Sharpening, Downsampling	1.28%
All Filters	1.02%

TABLE I: Reconstruction error for the dragonfly image set with different post-processed image adjustments.

from the unfiltered image set has decreased upon downsampling the image resolution to 70% of its original resolution, and applying a high pass filter with a kernel size of 3. The reconstruction error has further increased with the application of CLAHE with a clip limit of 1.5.

IV. Discussion

A. Reconstruction Error

The trend in the reconstruction error with a varying angle between consecutive views is likely due to an increased angle leading to fewer matches between consecutive views (as different features on the subject will be displayed), thus increasing the error (see *Appendix A* for a visualisation of feature matching between consecutive views with varying angles).

In a similar manner, the trend in the reconstruction error with a varying resolution downsampling factor can be explains as follows: too high of an image resolution results in the SIFT algorithm, which searches at a pixel-by-pixel scale, detecting noise or insignificant changes in the image as valid feature keypoints, while too low of a resolution greatly decreases the number of extracted features along with the precision of the location of potential keypoints (see Fig. 6). At both ends of this spectrum reconstruction error will be high.

Sharpening, or using a high pass convolution filter with a large kernel size, along with applying CLAHE with a high clip limit, will both increase noise in an image and thus reconstruction error. Blurring, or using a low pass convolution filter with a large kernel size will significantly lower the number of extracted features (see Fig. 9), which will increase reconstruction error.

It is worthwhile to note that for all of the varying parameters, the reconstruction error for x_1, x_2, and z_2, is consistently higher than that of y_2, and z_2. Upon examining the pair of images used to initialise the reconstruction, it is evident that the visibility of individual details in the images strongly affect the accuracy of their reconstruction. As Structure-from-Motion generates the sparse point cloud by incrementally evaluating images, the quality of the initial pair is crucial to reconstruction accuracy. In the initial pair of images, y_2 and z_2 lie in stark contrast to the background, while z_2 is barely visible and x_2 lies on a low contrast boundary (see *(*Appendix A)).

For three of the varied image condition parameters, the resolution size, the level of blur/sharpness, and the level of contrast, there is clearly a parameter value leading to the lowest reconstruction error. To reiterate, these values are the following:

1) Resolution: 70%, where the original resolution is 2048 x 1536
2) Sharpening: high pass filter kernel size = 3
3) Contrast: CLAHE clip limit = 1.5

These parameter values have been shown to lower reconstruction error for two subjects, where the images were initially captured under identical conditions, using the same photogrammetry rig.

However, as discussed earlier, applying the same CLAHE filters (used on the dragonfly image sets) on the cuboid image sets would not lower reconstruction error, but rather increase it to the point where reconstruction fails entirely. It must also be noted that while the reconstruction error for the dragonfly specimen decreased with the application of filters using parameter values from the cuboid experiments, the resultant reconstruction error (Table I) is still higher than that obtained when only applying the CLAHE filter (Fig. 11). This suggests that while error characterisation through ground truth validation can determine a range of optimal image conditions, specific parameter values require further inspection of individual subject cases.

B. Feature Matching Output

Data from the feature matching stage of SfM, when considered alongside the obtained ground truth data, suggests that it is indeed possible to determine the optimal parameter values for a given subject in the early stages of the SfM pipeline. For all parameters, one of the peaks on Figures 5, 8, and 12, representing the highest percentage of inlying matches corresponds to a quantitative compromise between the percentage of matched features and the mean extracted features of the three initial images in the dataset (see Figures 6, 9, and 13). The parameter value this peak exactly at the same value that gives the lowest reconstruction error in each case. Thus it is likely that the optimal parameter values may be determined solely from examining the extracted and matched features from the first few images captured of a given subject. This may be implemented in the proposed feature matching application by applying filters with varying parameters and selecting the value corresponding to the highest percentage of inlying matches and a mean or median value for the mean extracted features and percentage of matched features across the initial captured images.

V. Conclusion

The range of optimal image conditions under the same photogrammetric rig are defined by the parameter values corresponding to the lowest reconstruction error. Specific values for different subjects can be determined by examining the feature matching output just from capturing at least a pair of initial images. A simple assistive software is useful for acquiring the highest quality and lowest quantity of images.

Error determination through ground truth validation can be further refined by performing multiple trials of the same calibration object, and different calibration objects. Multiple trials (taking measurements of the 3D model) is important in this case as this step is done manually, and thus may be prone to small, random variations. Using different calibration objects will help determine whether or not there exists an universal range of optimal image conditions.

Furthermore, refining error determination in photogrammetry will also allow for a fully automated photogrammtry pipeline, where the optimal conditions can be determined from a single image and 3D models of the highest accuracy can generated.

Acknowledgments

I would like to express sincere gratitude to Dr. Huai-Ti Lin and Dr. Maria Jose Fernandez Jauregui for their support and guidance throughout the course of my research. My appreciation also goes to Dr. Sam Fabian, for providing me with crucial technical assistance when needed.

References

1) Dharmapuri, S., Tully, M. (2018). Evolution of Point Cloud, Lidar Magazine. Accessed 19 December 2018, ¡https://lidarmag.com/2018/07/16/evolution-of-point-cloud¿.
2) Autodesk ReCap, (2016). Autodesk Re-Make How to Take Photos for Photogrammetry. [online video] Available at: ¡https://www.youtube.com/watch?v=D7Torjkfec4¿. Accessed 3 December 2018.
3) Schoenberger, J. L., (2018). Tutorial COLMAP 3.6 documentation. COLMAP. Accessed 11 November 2018, ¡https://colmap.github.io/tutorial.html¿.
4) Schoenberger, J. L., Frahm, J. (2016). Structure-from-Motion Revisited. Conference on the Computer Vision and Pattern Recogntion. Conference paper.
5) Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision. 60(2). 91-110.
6) Akcay, O. and Avsar, E. (2017). THE EFFECT OF IMAGE ENHANCEMENT METHODS DURING FEATURE DETECTION AND MATCHING OF THERMAL IMAGES. ISPRS - International Archives of the

Photogrammetry, Remote Sensing and Spatial Information Sciences, XLII-1/W1, pp.575-578.

7) Ballabeni, A., Apollonio, F., Gaiani, M. and Remondino, F. (2015). ADVANCES IN IMAGE PRE-PROCESSING TO IMPROVE AUTOMATED 3D RECONSTRUCTION. ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-5/W4, pp.315-323.

APPENDIX A
ADDITIONAL GROUND TRUTH VALIDATION DATA

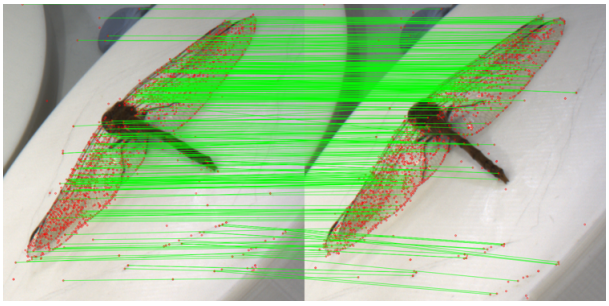

Fig. 14: Feature matching between consecutive views with a narrow angle.

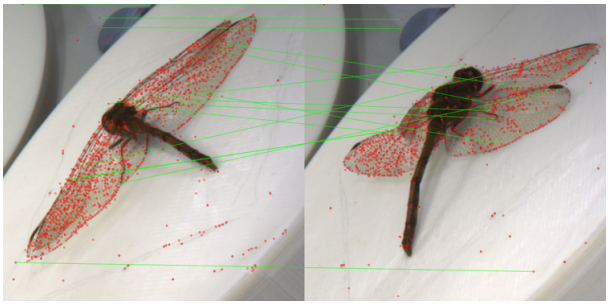

Fig. 15: Feature matching between consecutive views with a wide angle.

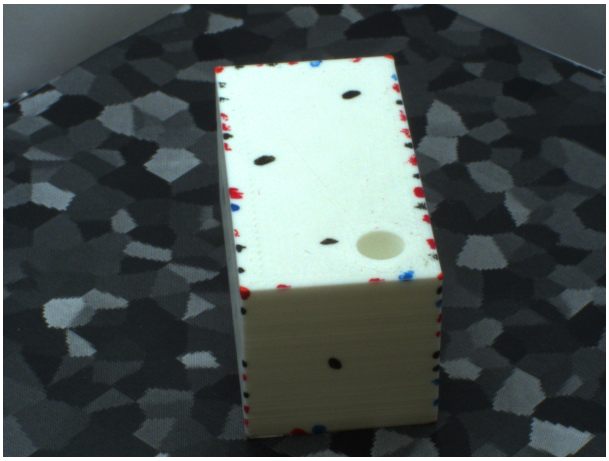

Fig. 16: Pair of images used to initialise reconstruction.

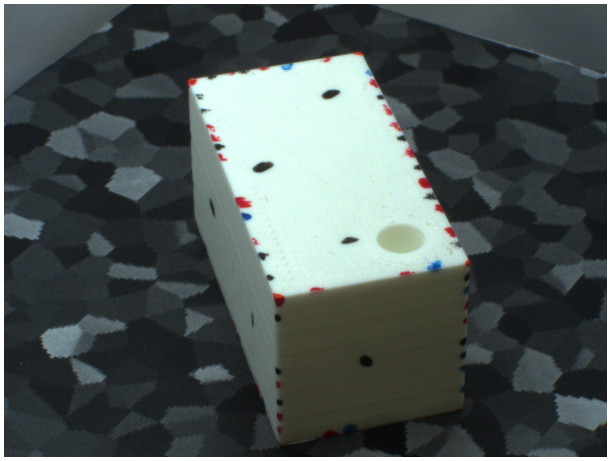

Fig. 17: Pair of images used to initialise reconstruction.

APPENDIX B
REAL TIME FEATURE MATCHING APPLICATION SCRIPT

The Python script for the feature matching application has been appended below:

```
import cv2
import os


def sift_detector(new_image, image_template):

    image1 = cv2.cvtColor(new_image, cv2.COLOR_BGR2GRAY)
    image2 = image_template

    sift = cv2.xfeatures2d.SIFT_create()

    keypoints1, descriptors1 = sift.detectAndCompute(image1, None)
    keypoints2, descriptors2 = sift.detectAndCompute(image2, None)

    FLANN_INDEX_KDTREE = 0
    index_params = dict(algorithm=FLANN_INDEX_KDTREE, trees=3)
    search_params = dict(checks=100)

    flann = cv2.FlannBasedMatcher(index_params, search_params)

    matches = flann.knnMatch(descriptors1, descriptors2, k=2)

    good_matches = []
    for m, n in matches:
        if m.distance < 0.7 * n.distance:
            good_matches.append(m)
    return len(good_matches), keypoints1


path = "C:\\Users\\Jill Ueng\\Documents\\MEng Final Year Project\\Feature Tracking\\Demo"
cap = cv2.VideoCapture(0)
cap.set(3, 480)
cap.set(4, 360)
count = 0
init = 0
saved = 0.0
check = 0

while True:
    ret, frame = cap.read()
    height, width = frame.shape[:2]
    top_left_x = int(width / 3)
    top_left_y = int((height / 2) + (height / 4))
    bottom_right_x = int((width / 3) * 2)
    bottom_right_y = int((height / 2) - (height / 4))
    frame = cv2.flip(frame, 1)

    cv2.rectangle(frame, (top_left_x, top_left_y), (bottom_right_x,bottom_right_y), (170, 170,
    cropped = frame[bottom_right_y:top_left_y, top_left_x:bottom_right_x]
    cv2.imshow('Object Detector using SIFT', frame)

    if init == 0:  # re/INITIALISE
        if cv2.waitKey(1) == 9:  # KEY: ENTER
            img_path = path + "\\img%d.jpg" % count
            cv2.imwrite(img_path, cropped)
            image_template = cv2.imread(img_path)

            if os.path.isfile(img_path):
                print('Image %d captured.' % count)
                count = count + 1
```

```
            init = 1
            check = 0

        elif cv2.waitKey(1) == 27:  # KEY: ESC
            print('Exiting (1) ~~!!!')
            for file in os.listdir(path):
                os.unlink(os.path.join(path, file))
            break

    elif init == 1:
        features = sift_detector(cropped, image_template)
        matches = features[0]
        kp = features[1]

        cv2.putText(frame, "MATCHES: %d" % matches, (157, 325), cv2.FONT_HERSHEY_COMPLEX_SMALL
        cv2.drawKeypoints(cv2.flip(cropped, 1), kp, frame[bottom_right_y:top_left_y, top_left_x
                          flags=cv2.DRAW_MATCHES_FLAGS_DRAW_OVER_OUTIMG)

        if check == 0:
            saved = float(matches)  # 100% of matched points
            check = 1

        if not saved == 0:
            cv2.putText(frame, "%.2f %%" % ((matches/saved)*100), (375, 325), cv2.FONT_HERSHEY_
            if 0.3*saved <= matches <= 0.5*saved:
                cv2.rectangle(frame, (top_left_x, top_left_y), (bottom_right_x, bottom_right_y
                cv2.putText(frame, 'PRESS ENTER', (145, 60), cv2.FONT_HERSHEY_COMPLEX_SMALL, 2

        cv2.imshow('Object Detector using SIFT', frame)
        if cv2.waitKey(1) == 9:
            init = 0
        elif cv2.waitKey(1) == 27:  # KEY: ESC
            # for file in os.listdir(path):
            #     os.unlink(os.path.join(path, file))
            print("Exiting (2) ~~!!!")
            break

cap.release()
cv2.destroyAllWindows()
```