Reconstruction of Small Scale 3-Dimensional Objects From Multiple Images

Dvir Hilu

***Abstract*—This paper aims to discuss the process of creating a tool for generating three dimensional models of small objects using a smart phone and a printable background. Using the phone, multiple images of the object are taken while it is placed in the center of the chessboard pattern background (with the middle portion of the chessboard removed). The set of x-shape corners in the printable background are first detected in the image using a Harris detector, and are filtered to remove false matches and ordered in a way that preserves projective-transformation invariance. The set of sixteen detected image points is then used with a nonlinear least squares algorithm to compute the extrinsic calibration parameters of the camera. Once the projective transformation is defined for each image through the camera calibration, features are matched across images using the Scale-invariant Feature Transform descriptors, and triangulated to three dimensional points using a homogeneous least squares algorithm. This results in a point cloud reconstruction of the objects. Computed results show that the current implementation, while qualitatively satisfactory at times, could be improved in a number of areas. These improvements are discussed at length.**

***Index Terms*—camera calibration, corner extraction, reprojection error, triangulation**

I. Introduction

Three dimensional modeling or reconstruction is an active field of study due to its wide ranging applications [1]. Although there are many existing technologies for 3D scanning, there are concerns when it comes to cost, privacy and confidentially as they usually rely on 3rd party Apps or programs [2]. The ability to perform the task locally significantly reduces the complexity and makes the technology more accessible. This opens the door to a variety of applications such as 3D artwork, hobbyist project prototyping, and more. This paper presents the methodology used in an open source tool for generating 3D models of small objects (a few cm in size) using a smart phone and a printable background.

The general workflow can be broken down to four high level steps: camera calibration, feature matching, 3D point triangulation, and point cloud interpolation. Camera calibration is a necessary step to compute the projective transformation between 3D world points and 2D image points [3]. While the intrinsic calibration parameters are inherent to the camera and can be computed once for a single camera (which was done using the MATLAB Calibration Toolbox for this project), the extrinsic parameters must be calculated dynamically for each image. Section II discusses different approaches to camera calibration and presents a solution to this problem for the application of small object 3D reconstruction.

Feature matching is the process in which unique features of an object are detected and matched across images under a variety of projective transformations. Given that each image point can be resolved into a 3D line (due to image depth ambiguity) [4], a feature matched across a set of images can then be triangulated into a 3D world point. A discussion of an implementation to these steps can be found in section III.

This paper was submitted on December 10, 2020.

D. Hilu is an Engineering Physics undergraduate student at the University of British Columbia, Vancouver, BC, Canada V6T 1Z1 (e-mail: dvirhilu@alumni.ubc.ca).

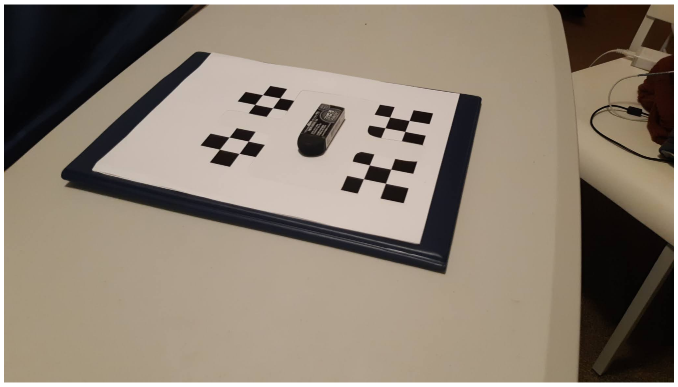


Fig. 1. An example of the typical setup and image for this application, showing a typical object that could be reconstructed by the methods presented in this paper. Only the space between the inner x-corners of the chessboard is reconstructed in 3-dimensions.

The set of all reconstructed features combine to form a point cloud in 3D. The effectiveness of the methods proposed in this paper to reconstruct the point clouds of various objects is discussed in section IV. Finally, the point cloud can be interpolated to generate a solid surface. While various methods were explored [5], [6], this could not be implemented due to project time constraints and should be explored in the future.

II. Computation of Image Projection Matrix

The projective transformation between a 3D world point, $\vec{X} = [X, Y, Z, 1]^T$, and the 2D image point, $\vec{x} = [j, i, 1]^T$, in homogeneous coordinates is

$$K[R \mid \vec{t}]\vec{X} = w\vec{x}. \tag{1}$$

Here, K is the intrinsic camera calibration matrix, w is a scale factor, and $[R \mid \vec{t}]$ is a composite matrix containing a 3x3 rotation matrix R and translation vector $\vec{t}$, describing the extrinsic camera calibration properties [7].

Although the intrinsic camera calibration parameters are constant for a given camera and were found only once using the MATLAB Camera Calibration Toolbox, the extrinsic parameters must be found on a per-image basis.

It is possible to find the extrinsic parameters of the camera from image features matched across more than three views [3], but given the short timeline of the project, a background with defined features (shown in Fig. 1) was used to calculate the geometrical portion of the image projection matrix. The problem of finding the extrinsic parameters is then divided in two: first, 16 chessboard x-corners in the background are to be detected and sorted in a manner that preserves projective-transformation invariance, then, the 3 rotation and 3 translation parameters are calculated from the known correspondence of image and world points.

A. X-Corner Detection

To locate an initial set of x-corner candidates, two commonly used methods are variations of the Hessian corner detector similar to what is proposed by [8], and the Harris corner detector [9]. Although the Hessian corner detector in [8] produces acceptable results at low perspective distortion (i.e. image taken almost at a “bird’s eye view”), this method was far too noisy and resulted in poor corner localization at larger perspective distortions (i.e. images taken more parallel to the x-corner plane).

As a result, the Harris corner detector was explored instead. The Harris corner detector seeks to find how the image changes across various shift directions, as a region in the image containing a corner would experience a large change in all directions. It does so by defining the cost function presented in (2), where M is the structure tensor, k is a controlled constant typically between 0.04 and 0.06, and λ_1 and λ_2 are the eigenvalues of M [9]. Given that the relative sizes of λ_1 and λ_2 describe the dominance of a single direction in the gradient distribution of a window (i.e. if $\lambda_1 \gg \lambda_2$, the gradient distribution is heavily aligned with the eigenvector of λ_1), The following relationship between image features and R can be extracted:

- $R \simeq 0 \rightarrow$ gradients in the window are small and the window thus corresponds to a flat region in the image
- $R \ll 0 \rightarrow$ there is a single dominant direction in the window and the window thus corresponds to an edge
- $R \gg 0 \rightarrow$ there are multiple directions of large gradients in the window and the window thus corresponds to a corner

$$R = \det(M) - k[\text{trace}(M)]^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2 \quad (2)$$

Based on the relationship above, a corner threshold of 10% of the maximum was empirically set for R, creating a boolean mask containing information about which points were above the threshold. The boolean mask contained clusters at corners that needed to be reduced down to a single corner point. This was done by iteratively increasing a window around an above-threshold point until it contained the entire cluster. Once the entire cluster was contained within the window the corner point was found by finding the centroid of the windowed region in the boolean mask.

Another popular set of methods are variations of the SUSAN corner detector as described in [10], but given that the Harris corner detector worked sufficiently well, this group of methods was not experimented with.

B. Elimination of False Corners

The Harris corner detector works well for finding initial candidates, but is too generic and will generate too many false matches. To filter the points and extract the x-corners, the detected corners were filtered using centrosymmetry and distance criteria based on the work of [11].

The heuristic for the centrosymmetry filter is that x-corners are symmetric along a diagonal direction while being antisymmetric along vertical and horizontal direction. We start by taking a circular mask of radius R_c like the one shown in Fig. 2, and defining the average intensities in the slice containing angles between $\frac{k\pi}{4}$ and $\frac{(k+1)\pi}{4}]$ as I_k. The difference in the average intensities of I_k and I_{k+4} should be much smaller when compared with the difference in the average intensities of I_k and I_{k+2} $(k\epsilon[0,7])$. Formalizing this, the imposed filtering conditions are described by (3) and (4), where H is the filtered x-corner boolean mask and p is a controlled parameter between 0 and 1. The reason for the“or” condition and the adjustable threshold is to weaken the requirements of the filter to avoid eliminating true x-corners at higher projective distortions.

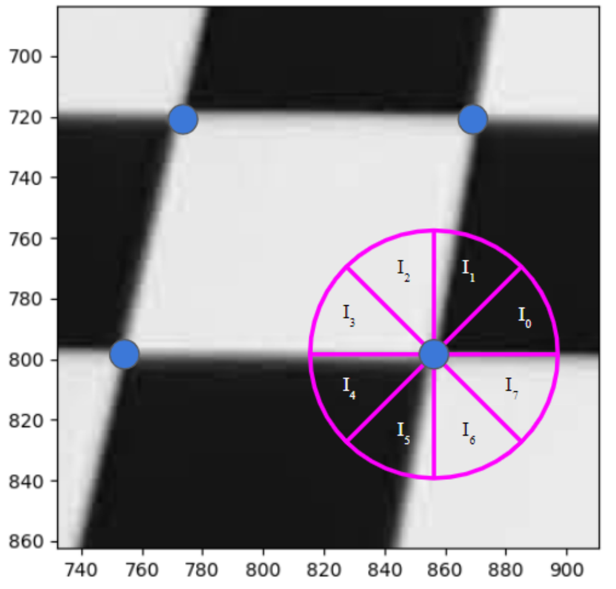


Fig. 2. Visualization of centrosymmetry criterion heuristic. The difference in the average intensities across the diagnoals will be very small compared to the average difference in intensities across the vertical and horizontal directions.

$$D_k = \begin{cases} I_k - I + k + 4 & k = 0,1,2,3 \\ \frac{I_{k-4}+I_k-I_{k-2}-I_{k+2}}{2} & k = 4,5 \end{cases} \quad (3)$$

$$(D_0 < pD_4 \;\&\&\; D_2 < pD_4) \parallel (D_1 < pD_5 \;\&\&\; D_3 < pD_5) \quad (4)$$

Any corners that pass the centrosymmetry test are then filtered using a distance threshold criterion. As shown in Fig. 2, each x-corner has 3 neighbours within a close proximity. Thus, if a point’s third nearest neighbour is located above some distance threshold, d_{TH}, it can be eliminated with high confidence.

The two false x-corner elimination methods described above contain three parameters: R_c, p, and d_{TH}. Although [11] presents a method for adapatively calculating these, the method assumes a full chessboard is present with no object in the middle, and therefore does not perform well in this analysis. Given that the images used for reconstructing the objects are fairly predictable and constrained (i.e. no x-corners hidden or out of frame, relatively consistent sizing of x-corners on frame, etc.), these parameters were found empirically to ensure no false matches occur while maximizing the number of images for which all x-corners were found. For all image sets investigated so far (and presented in this report), all x-corners were found and no false corners were seen in the background. Future work expanding on this project should investigate modifying the adaptive parameter calculation method in [11] to apply to the image set explored in this paper.

C. World Points to Image Points Mapping

Once the x-corners are obtained, it is important to find a projective-transformation-invariant ordering of the points that could then be mapped directly to the known 3D world points of the x-corners. Given that there are 4 “clusters” of 4 x-corners, the sorting is done for each cluster and each corner independently. Each cluster is represented by the outermost corner, which is also deemed as “corner 0” for the specific cluster.

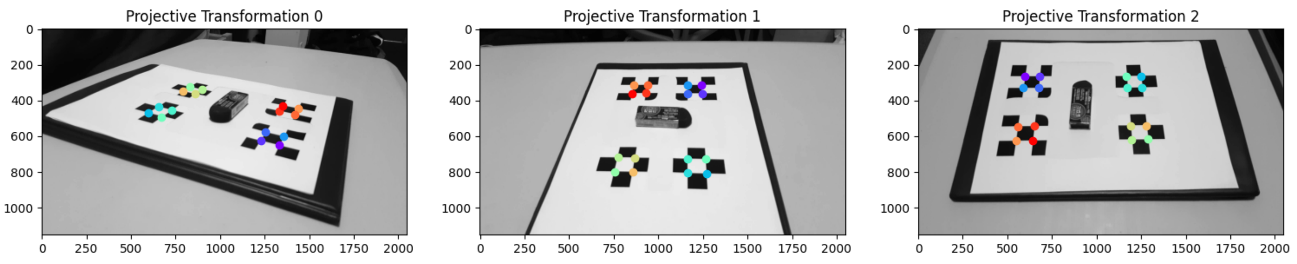

Projective Transformation 0
Projective Transformation 1
Projective Transformation 2
0
200
400
600
800
1000
0
250
500
750
1000
1250
1500
1750
2000

Fig. 3. An example of three different views and the resulting x-corner ordering. The order of the points can be seen in the colour variation, starting from purple. Note that in all 3 images, points with the same colour correspond to the same world point.

To isolate for the correct ordering of the clusters, it is important to first extract "corner 0" in each cluster. The method suggested by [11] finds the four x-corners as the extrema of (x+y) and (x-y) for all detected x-corners where (x,y) are the image coordinates of an x-corner. This method works well in most cases, but when the board is rotated at an angle that is close to 45° (and multiples of 90° after that), the projective transformation can skew the point distribution in a way that a point can appear as two of the four extrema. However, in such a scenario, the four "corner 0" points can be found as the extrema of x and y. Thus, an updated algorithm can be developed for extracting the four "corner 0" points:

1) First, find the four extrema of (x-y) and (x+y) and order them as [min(x+y), min(x-y), max(x+y), max(x-y)]. This sorts the points in a clockwise direction (so the order will be the same regardless of orientation)
2) Find the distance between each pair of neighbouring points in the array. If the distance between any of them is less than d_{TH} (defined in section II-B), the corners must be recalculated
    a) If the corners must be recalculated, find them as the extrema of x and y and order them as follows [min(x), min(y), max(x), max(y)]. Again, this ensures a clockwise order.

Next, to order each cluster, a projective-transformation-invariant "cluster 0" must be determined. This is done by first taking a small step in the direction of the centroid of x-corner points. For two of these points, the colour of the pixel after the step will be black, while for the other two it will be white. After eliminating the two white points, the two black points can be distinguished by finding the distance to the next clockwise "corner 0" point (this ordering was already found in a previous step). Since the two black points are adjacent, one will have a longer distance to the next clockwise "corner 0" than the other. This point is chosen to represent "cluster 0".

The individual points in each cluster can be sorted by finding $\sin(\theta_0)$, where θ_0 is the angle formed by the point, "corner 0", and the x axis. To order the points in a cluster in a clockwise order, the points are sorted in decreasing order of $\sin(\theta_0)$, starting with "corner 0". The resulting point order can be seen under three different projective transformations in Fig. 3.

To obtain better precision on the location of the corner, the squared grayscale centroid method described in [12] was used. The method finds the location of the x-corner (x_0, y_0), by calculating the centroid of the squared intensities, $I(x,y)^2$ in a specified window W, as described in (5). Note that while [12] uses a circular window, this implementation use a square window as it was marginally easier to implement while producing sufficient results. Now that the subpixel locations of the x-corners are found and sorted, a direct mapping between a set of 16 image points and world points for each image is obtained.

$$x_0 = \frac{\sum_W xI(x,y)^2}{\sum_W I(x,y)^2}, \quad y_0 = \frac{\sum_W yI(x,y)^2}{\sum_W I(x,y)^2} \tag{5}$$

D. Computation of Extrinsic Camera Parameters

Recall the projective transformation in (1), containing the unknown 3x4 extrinsic parameters matrix $[R \mid \vec{t}]$. Given that (1) is a linear transformation between image points and world points, and given that the 16 image-world point mappings are sufficient to resolve the 12 matrix coefficients, the transformation can be found using a linear least-squares solution. However, this would produce a poor solution, because while the translation vector $\vec{t}$ has three degrees of freedom (three translation coordinates t_x, t_y, t_z), the 3x3 rotation matrix R only has 3 degrees of freedom (the three Euler angles, α, β, γ). Furtheremore, as shown in (6), the matrix coefficients are nonlinear with respect to those three degrees of freedom. This results both in an inability to solve for the six degrees of freedom system using a linear method, and in an overestimation of the degrees of freedom of the system if using a linear least squares approach to find the coefficients of the matrix independently.

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix} \tag{6}$$

To solve this problem, the six degrees of freedom were instead directly calculated using a nonlinear least squares approach. Here, the vector $\vec{g} = [\alpha, \beta, \gamma, t_x, t_y, t_z]$ represents the full geometry of the camera:

1) An initial guess for the geometry of the camera was set to be $\vec{g} = [0, 0, 0, 0, 0, 30]$. This represents a fully vertical view at a height of about 30cm. This guess is the average typical expected camera position, and was thus the simplest method to achieve good convergence for the calculation. Future work expanding on this project should look at ways to improve on this initial guess (e.g. adjust initial guess based on x-corner image point distribution).
2) To calculate the k^{th} expected image point $\vec{x_{c,k}}$ as a function of $\vec{g}$, we first calculate $\vec{x_{p,k}}$, the projection of the k^{th} world point $\vec{X_{w,k}}$. Using (1) and (6), we can define that
$$\vec{x_{p,k}} = K[\,R(\alpha,\beta,\gamma) \mid \vec{t}\,]\vec{X_{w,k}}. \tag{7}$$
3) Given that $\vec{x_{p,k}}$ is the projection expressed in homogeneous coordinates, we can extract $\vec{x_{c,k}}$ with the simple conversion in

(8).

$$\vec{x_{c,k}} = \frac{1}{x_{p,k,3}} \begin{bmatrix} x_{p,k,1} \\ x_{p,k,2} \end{bmatrix} \tag{8}$$

4) The sum of square error function, $S(\vec{g})$, could then be calculated between the expected image point, $\vec{x_{c,k}}$, and actual image point, $\vec{x_{a,k}}$ as

$$S(\vec{g}) = \sum_{k=0}^{15} |\vec{x_{c,k}} - \vec{x_{a,k}}|^2 \tag{9}$$

5) Finally, a minimizer is used to find the value of $\vec{g}$ that minimizes $S(\vec{g})$. Given that calculating the gradient of S is quite complex, a variety of gradient-less minimization methods provided by Scipy were tested, of which Powel's method was found to be the fastest and most effective [13].

E. Elimination of Poorly Performing Images

To determine the accuracy of the calculation in section II-D, an analysis was done on the projection error when using the calculated projection matrix for each image. To do this, the known world points of the x-corners were projected onto the image and compared to the detected x-corners in the image. The distributions of mean and maximum error (calculated as the distance between the projected and detected x-corner) are presented in Fig. 4. As shown, while most images produce minimal error levels, there are a few that perform poorly and must be eliminated. As a result, any images for which the mean projection error is larger than 10 pixels or the max projection error is larger than 20 pixels were eliminated from the analysis.

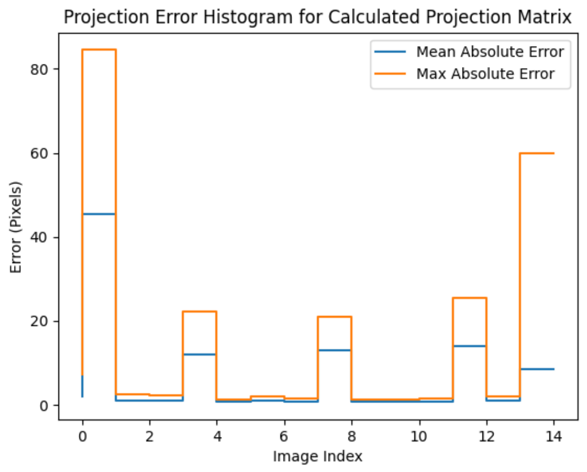


Fig. 4. The mean and maximum absolute projection errors using the computed projection matrix for each image in the eraser image set (an example of an image in the eraser set is given in Fig. 1). For each image, all 16 x-corner world points were projected onto the image using the computed projection matrix, and compared to the detected x-corner image points.

III. Feature Matching and Reconstruction

A. Feature Detection and Matching

Recalling back to (1), now that the projective transformation is calculated, we simply need to find a sufficient number of points across different images. Given that each image point can be used to define a line in 3D space [4], a feature must be matched across at least two images in order to be able to successfully triangulate its 3D position.

The task of matching features between images was performed using the Scale-Invariant Feature Transform (SIFT). SIFT achieves scale invariance by locating features to be the local extrema of the Laplacian of Gaussian of an image across spatial dimensions and a spectrum of Gaussian scales (also known as scale space). This is because an object that increases in scale in an image could then generate the same signature at a larger Gaussian blur scale. After feature detection and some filtering, features are assigned a 128D descriptor containing binned gradient information in the local region of the feature, and thus different image features have fairly distinct SIFT descriptors. Finally, the feature detector achieves rotation invariance by shifting the binned gradient information in the descriptor based on the dominant gradient directions around the feature [14].

Once features are detected in each image, a feature in image A can be matched to a feature in image B by finding the nearest neighbour descriptor (in terms of Euclidean distance) in image B. If image B contains two near neighbours to the feature in image A, this means that two different features in image B constitute the quality match to the feature in image A and the matching is ambiguous. Such situation indicates a high probability of the nearest neighbour being a false match. Thus, if the ratio of nearest neighbour descriptor euclidean distance and second nearest neighbour euclidean distance is greater than 0.8, the match is discarded and it is deemed that the feature in image A has no matching feature in image B [14].

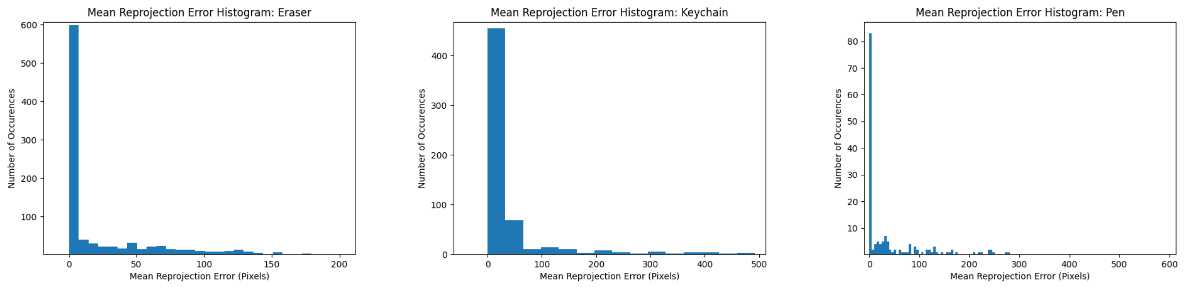
When matching features across images, the SIFT search is only conducted within the inner corners of each x-corner cluster. This reduces the number of analyzed features in the image, and along with the already relatively small number of features detected across images, ensures that brute-force feature matching can be executed relatively quickly. Note that certain object properties like light reflectance could pose a challenge to this algorithm, since it appears on different points of the object in different views, which could result in false matches.

B. 3D Triangulation of Matched Features

Given that the projective transformation equation in (1) is a linear mapping between 3D world points and 2D image points, the 3D points can be reconstructed from the matched feature points via a linear least squares solution provided there are $N \geq 2$ matching features. To improve the accuracy of the solution, [15] suggests a method to convert the inhomogeneous system into a homogeneous system least squares problem that is more robust to error from the homogeneous coordinates scale factor. The scale factor can be eliminated by realizing that the projection of the normalized homogeneous 3D point is a scalar multiple of the image point and thus would yield a 0 cross product, as presented in (10). Hekre, P_k and $\vec{x_k}$ are the projection matrix and feature coordinates for the image containing the kth matching feature, respectively; $P_{k,j}$ is the jth row of projection matrix P_k; and $\vec{X}$ is the triangulated world point. Note that the 3rd equation in the system is linearly dependent and can therefore be eliminated.

$$P_k \begin{bmatrix} wX \\ wY \\ wZ \\ w \end{bmatrix} \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = w \begin{bmatrix} xP_{k,3}\vec{X} - P_{k,1}\vec{X} \\ yP_{k,3}\vec{X} - P_{k,2}\vec{X} \\ xP_{k,2}\vec{X} - yP_{k,1}\vec{X} \end{bmatrix} = \vec{0} \tag{10}$$

This means that each matching feature can generate two homogeneous equations, and having $N \geq 2$ features allows to construct the homogeneous linear system shown in (11). The solution to the system is the eigenvector associated with the smallest eigenvalue of the matrix $B^T B$.



(a) Re-projection error of eraser image set (b) Re-projection error of keychain image set (c) Re-projection error of pen image set

Fig. 5. Mean re-projection error of reconstructed 3D point clouds. The value is calculated by re-projecting the point to all images which contain it as a feature and calculating the average of re-projection error across all such images. Refer to section IV for more information about the objects in each image set.

$$B\vec{X} = \begin{bmatrix} xP_{1,3} - P_{1,1} \\ yP_{1,3} - P_{1,2} \\ xP_{2,3} - P_{2,1} \\ yP_{2,3} - P_{2,2} \\ \vdots \\ xP_{N,3} - P_{N,1} \\ yP_{N,3} - P_{N,2} \end{bmatrix} \vec{X} = \vec{0} \tag{11}$$

C. Re-Projection Error for Elimination of Error-Prone Reconstructions

Once triangulation of points is complete, error prone reconstructions are first eliminated by finding the re-projection error of the reconstructed points in the images they were located in. The mean re-projection error, defined as the average distance between the re-projected point and the features it corresponds to across the various images, is computed for each reconstructed point. Any reconstructed point that was found to have a mean re-projection error of more than 10 pixels was eliminated. The threshold for mean re-projection error was found empirically by examining the quality of the reconstructed point clouds. The re-projection error for the three test cases that will be used in section IV is shown in Fig. 5

D. Elimination of Outliers

While the re-projection error works well as a filter for error-prone reconstructed points that appear across a variety of images, it fails when the reconstructed point is only a feature in two images. This is because two matching image point provide an exact solution to the triangulation problem in many cases, which would therefore result in a re-projection error of 0 even if there is error in the reconstruction. To counter this, we analyze the distribution of points in the point cloud in order to remove any outliers.

We start by eliminating all points for which Z$<$ 0, since Z=0 corresponds to the surface on which the object rests. Then, the z distribution is further filetered by eliminating all points in the distribution that have a Z value higher than the 95th percentile. Then, the X and Y distributions are filtered by removing any points that are 2.5 standard deviations from the mean. All thresholds in this analysis were found empirically by varying the parameters and visually confirming that the distribution of points in the point cloud does not have any clear outliers.

IV. Experiments

To test the methods discussed in sections II and III, a variety of different objects were reconstructed into point clouds, four of which will be discussed in this section. The objects subjected to this experiment are presented in Fig. 6.

In general, the images were taken in a fairly consistent manner, always including the full x-corner board in the picture, and always ensuring that the object does not hide any of the x-corners. Furthermore, the background in all images were relatively featureless, and posed little issues to the image detection. This was done to maximize reconstruction accuracy, but the effects of weakening the conditions and rules posed on the image should be explored in the future to ensure a more robust reconstruction algorithm.

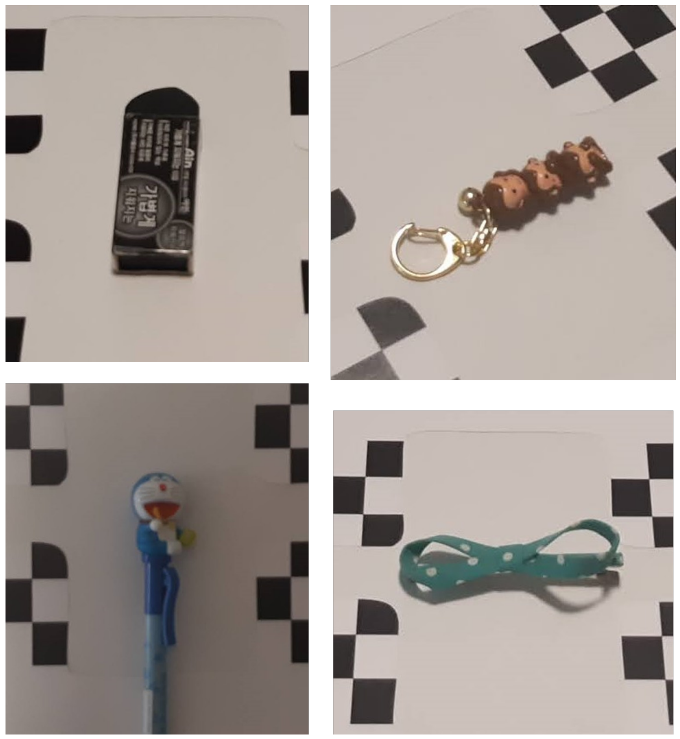


Fig. 6. Objects used for the experiments discussed in section IV. From top left it is an eraser, a keychain with three monkey faces, a decorative pen, and a hair ribbon

A. Eraser Image Set

The eraser image set was the image set first used to develop the method. The object has various advantages, including a simple shape, the fact that it is not lustrous, and the text on it which is favourable for SIFT feature detection.

The point cloud generated for this image set is presented in Fig. 7, along with a drawing presenting an outline of the actual object to make it easier to distinguish the shape. As expected, the top of the eraser, which includes a lot of writing reconstructed quite well. This is due to the fact that SIFT behaves well with features that contain a lot of distinct changes (which are picked up well by the binned gradient descriptor). However, one of the sides and the actual eraser portion are relatively featureless, and cannot be distriguished very well by algorithms that seek high frequency transitions like SIFT.

While it seems that not enough features were detected, there are nearly no noticeable noise points in the image (points that are located far away or seem out of place). This confirms that the error reduction methods presented in sections II-E, III-C, and III-D performed quite well for this image set to remove noise from the point cloud.

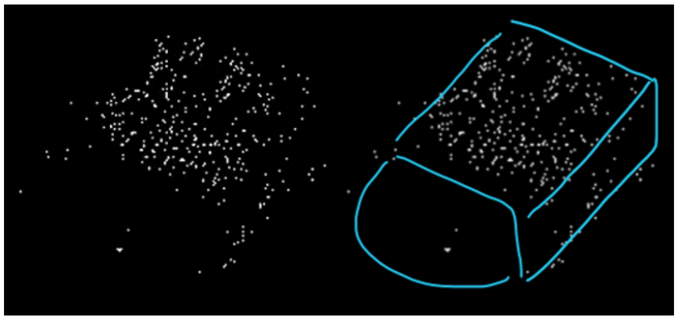


Fig. 7. The reconstructed point cloud for the eraser image set. The top, which is full of distinct text feature reconstructed well. However, sides were all relatively textureless and reconstructed poorly.

B. Keychain Image Set

The keychain was selected as the next reconstruction object because it has several elements that make it more challenging than the eraser. First, the keychain has metallic components which produce significant light reflections (which could be picked up as a noisy feature). The plastic itself is also somewhat lustrous, and produces light reflections on the main object itself. Secondly, it has a much more complex shape than the eraser, which should make reconstruction more challenging. Finally, the keychain includes some repeating features around the faces of the monkey, which could lead to some false image matches.

The point cloud reconstruction of the keychain image set is shown in Fig. 8. As shown, the object once again reconstructed fairly well, especially near the top. Although the sides are once again less densely populated with points as with the eraser. Note that the shape of the metal chain can be roughly made out from the point cloud and contains a relatively low amount of noise points, despite the large amounts of light reflections it exhibited. Once again, a low number of noise points can be seen in the reconstruction, though a small number of noise points can be viewed around the chain.

C. Pen Image Set

The pen was chosen as the next object because it was similar in its light-reflecting ability, while presenting some additional challenges. The pen reaches out of the bounds of the x-corner, which could add some noise when detecting the x-corners. Furthermore, the pen in general is slightly more featureless than the keychain. Additionally, to provide an additional challenge, this image set was taken at a faster pace and with less care, producing images that are in general slightly worse in quality.

The point cloud reconstruction of the pen image set is shown in Fig. 9. As opposed to the other two image sets, the reconstruction

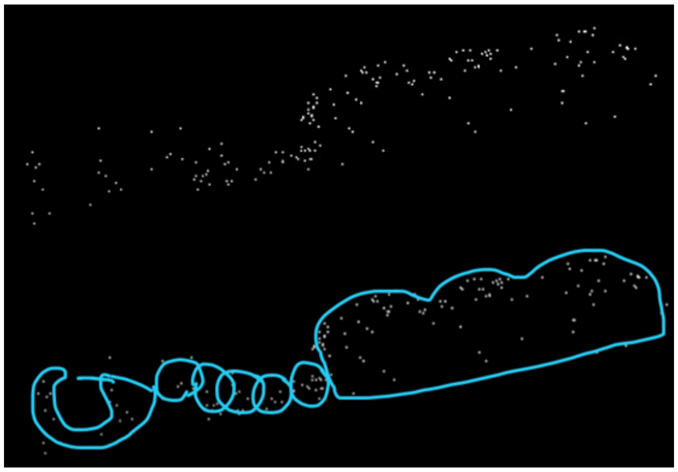


Fig. 8. The reconstructed point cloud for the keychain image set. Once again, the top reconstructed fairly well, while the sides have a much more sparse distribution.

quality of the pen set was extremely poor, with the number of points being too small to properly make out the general shape of the object. This could be due to the fact that images were slightly lower quality, or the fact that SIFT found a relatively small amount of features for this image.

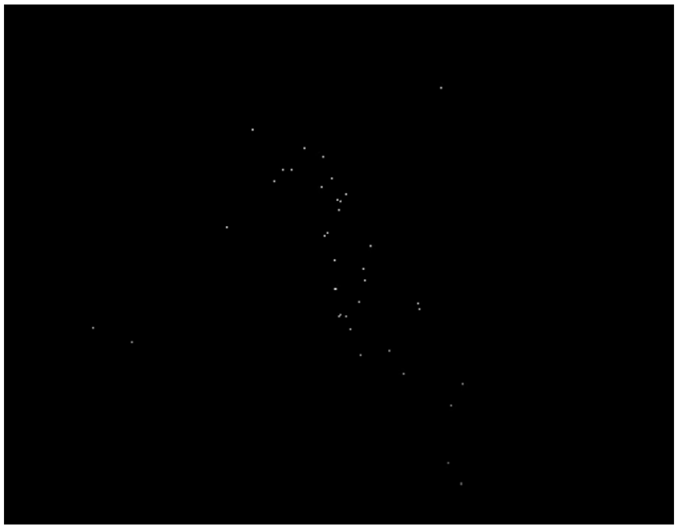


Fig. 9. The reconstructed point cloud for the pen image set. As shown, reconstruction quality is extremely poor and points in the cloud are sparse.

D. Hair Pin Image Set

The last object that was experimented with was the hair pin. This object was chosen specifically because of its complex shape and loops. It also poses an additional challenge in that it contains many features that are symmetric and repetitive, which could make feature matching with SIFT challenging.

The reconstruction of the hair pin set is shown in Fig. 10. While the set of points is definitely too sparse to produce a good interpolation, the general outline of the object can be made out from the point cloud. This is likely a sign that there were not enough feature matches to produce a more dense point cloud, which could be attributed to the repetitive pattern on the object. As with the previous cases, there are very little noisy points in the cloud.

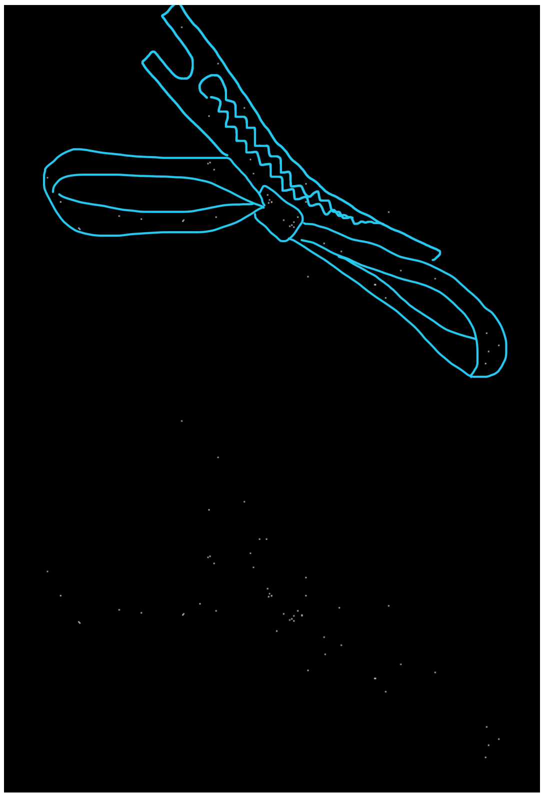


Fig. 10. The reconstructed point cloud for the hair pin image set. Although the points are very sparse, the overall shape of the object could be detected in their distribution.

V. Conclusion

This project explored creating a tool for generating 3D models for small objects using a phone and a printable chessboard pattern background. As presented in section IV, although the general shape of the object is distinguishable in the resulting point cloud, there are a few areas where the point cloud is too sparse to generate a proper reconstruction. Furthermore, the current implementation's quality is highly sensitive to the object chosen and is not robust to handle extreme projective distortions or camera angles.

Table I presents a quantitative summary of the performance of each step in the method. This summary highlights a few issues with the presented reconstruction methods which could have been addressed if more time could be allocated to the project. The large number of images eliminated suggest relatively poor accuracy for the nonlinear least squares projection matrix computation. This could suggest either that the initial guess should be estimated more accurately, or that the method itself is inherently inaccurate. One potential way to improve the initial guess could be by using the distribution of detected x-corners. For example, simply examining which area of the image"cluster 0" is located in should give a rough estimate of the rotation of the camera along the z axis. Furtheremore, relative distances and angles of the point clusters could be used to generate a rough estimate for the tilt angle of the camera, as well as the relative distance from the x-corner board. Even a small reduction in the number of eliminated images could result in a massive boost to the number of feature matches (as indicated by the average features/image field) and therefore improvements to the nonlinear least squares computation of the projection matrix should be explored in future work.

Although the number of features and feature matches for the keychain and eraser are relatively high, there is a significant drop for the pen and hair pin. It is expected that this drop is due to the relatively featureless structure of these objects, and as a result different methods that deal with relatively textureless objects should be explored, similar to the what is discussed in [1]. Furthermore, the low number of reconstructed points generated from the relatively high number of features, as well as the large number of points that are filtered could indicate an issue when matching features between images. One could potentially counter that by constraining the SIFT match search to an expected match region in the image. As shown in Epipolar Geometry, a feature in an image could be localized to a line in another image [4]. As a result, instead of only looking for features that have a short descriptor Euclidean distance, one could alter the requirement to maximize some linear combination of the descriptor distance and the distance to the line expected to contain the feature match.

Additionally, the high number of eliminated points in the point cloud could suggest inadequate accuracy for the homogeneous least squares triangulation method. This is somewhat expected, as the method is not invariant to projective transformations [15]. There are a variety of methods that are invariant to projective transformations described in [15], which also perform better statistically, but many of them do not expand well to more than two views. Nevertheless, presented methods that do expand well should be explored. Another option is to explore the Structures from Motion (SfM) algorithm, which is a popular reconstruction method in Photogammetry [16].

Finally, future work should explore various 3D point cloud interpolation methods in order to turn the point cloud reconstruction into a solid surface 3D model. This could be done using a variety of methods, such as generating a Delaunay triangle mesh [5], or using a Kriging interpolation method, which interpolates a Gaussian between the points [6].

Acknowledgment

This work was supported by Professor Rafeef Garbi, Department of Electrical and Computer Engineering, University of British Columbia.

The following portions of the project were implemented using 3rd party software libraries:

- Intrinsic camera parameter calibration was done using the MATLAB Single Camera Calibrator App
- Undistortion of images, Harris response, and SIFT corner detection were all computed using the OpenCV library for Python
- Powel method minimization was performed using the Scipy module in Python
- The 3D viewer used to interactively view the 3D point clouds was created using OpenGL and Pygame

References

[1] J. Hafeez, A. Hamacher, S. Kwon, and S. Lee, "Performance evaluation of patterns for image-based 3d model reconstruction of textureless objects," in *2017 International Conference on 3D Immersion (IC3D)*, pp. 1–5, 2017.

[2] J. Tan, U. Drolia, R. Martins, R. Gandhi, and P. Narasimhan, "Short paper: Chips: content-based heuristics for improving photo privacy for smartphones," 07 2014.

[3] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[4] R. Hartley and A. Zisserman, "Epipolar geometry and the fundamental matrix," in *Multiple View Geometry in Computer Vision*, ch. 9, pp. 239–259, Cambridge, United Kingdom: Cambridge University Press, 2nd ed., 2003.

[5] Song Dahu and Li Zhongke, "A fast surface reconstruction algorithm based on delaunay," in *2012 International Conference on Computer Science and Information Processing (CSIP)*, pp. 981–984, 2012.

Object	No. of Images	Eliminated Images	Avg. Features/Image	Point Cloud Size	Reprojection Error Eliminations	Outliers Eliminated
eraser	15	5	307.5	948	337	148
keychain	12	4	281	616	248	177
pen	20	14	96.7	163	78	45
hair pin	20	11	160.5	448	340	56

TABLE I

QUANTITATIVE SUMMARY OF THE EFFECTIVENESS OF VARIOUS STEPS IN THE RECONSTRUCTION METHOD

[6] R. W. Parrott, M. R. Stytz, P. Amburn, and D. Robinson, "Towards statistically optimal interpolation for 3d medical imaging," *IEEE Engineering in Medicine and Biology Magazine*, vol. 12, no. 3, pp. 49–59, 1993.

[7] R. Hartley and A. Zisserman, "Camera models," in *Multiple View Geometry in Computer Vision*, ch. 6, pp. 153–177, Cambridge, United Kingdom: Cambridge University Press, 2nd ed., 2003.

[8] D. Chen and G. Zhang, "A new sub-pixel detector for x-corners in camera calibration targets.," in *WSCG*, pp. 97–100, 01 2005.

[9] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, pp. 147–151, 1988.

[10] Yang Xingfang, Huang Yumei, and Gao Feng, "A simple camera calibration method based on sub-pixel corner extraction of the chessboard image," in *2010 IEEE International Conference on Intelligent Computing and Intelligent Systems*, vol. 3, pp. 688–692, 2010.

[11] Y. Liu, S. Liu, Y. Cao, and Z. Wang, "Automatic chessboard corner detection method," *IET Image Processing*, vol. 10, no. 1, pp. 16–23, 2016.

[12] M. Shortis, T. Clarke, and T. Short, "A comparison of some techniques for the subpixel location of discrete target images," *Videometrics III, SPIE*, vol. 2350, 10 1994.

[13] M. J. D. Powell, "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *The Computer Journal*, vol. 7, pp. 155–162, 01 1964.

[14] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, pp. 91–, 11 2004.

[15] R. Hartley and A. Zisserman, "Structure computation," in *Multiple View Geometry in Computer Vision*, ch. 12, pp. 310–323, Cambridge, United Kingdom: Cambridge University Press, 2nd ed., 2003.

[16] M. Smith, J. Carrivick, and D. Quincey, "Structure from motion photogrammetry in physical geography," *Progress in Physical Geography*, vol. 40, 11 2015.