MODEL-BASED METRICS FOR ASSESSING COMPLETENESS AND ACCURACY FOR 3D IMAGE-BASED RECONSTRUCTION METHODS

BY

THAPANAPONG RUKKANCHANUNT

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Advisor:

Assistant Professor Mani Golparvar-Fard

Abstract

Structure from Motion (SfM) is a technique for inferring the underlying 3D geometry of the real world from a set of unordered images using a pinhole camera model. Due to a wider availability of cameras, the availability of commodity smart phones and tablets equipped with cameras, and enhanced computational power, more attention is now being given to leverage visual data such as still images and video streams for the automated processes such as mobile augmented reality and engineering performance monitoring applications. Discuss significant research progress related to leveraging rich visual data for 3D image-based point cloud modeling purposes, today there is no way to provide feedback to users who are taken these imagery on the best location and viewpoints for cameras. To address current limitations, this thesis introduces 3D image-based modeling incrementation -- a variant of incremental SfM -- where no prior knowledge of the new image/cameras is given. The state-of-the-art online algorithm for SfM only focus on processing video streams in which the images are ordered and sequence is known as a priori. In this thesis, we use a heuristic approach to avoid the need for pair-wise image matching which is a bottleneck for incremental SfM processes. We also discuss how we measure the completeness of the 3D point cloud without the knowledge of the ground truth. We propose a general framework for feedback systems where the location and the pose of the new camera that can enhance the completeness of the 3D image-based point cloud is predicted. We demonstrate that this framework can assist users of SfM methods to automatically improve the completeness of the reconstructed 3D point cloud models form the real world scenery. We validate our algorithm on both small and large dataset as well as indoor and outdoor scenes using daily photologs collected from two ongoing construction sites. The perceived benefits and practical significance of the proposed method are also discussed in detail.

ACKNOWLEDGEMENT

I would like to express my deep gratitute to Professor Mani Golparvar-Fard, my thesis advisor, for his patient guidance, enthusiastic encouragement, and useful critiques of this research work. His knowledge on the subject matter and his vision on the interesting problems that motivate me to continue on this line of research are very much appreciated.

I would also like to extend my thanks to Changchang Wu for his feedbacks on technical concepts and issues and Kathleen Tuite for her help in explaining the concept behind her project.

Finally, I wish to thank my parents for their support and encouragement throughout my educational career.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Structure from motion (SfM) has been introduced in many engineering fields outside computer vision community [3]. The ability to reconstruct a 3D model from an unordered set of photos without specialized equipment proves to be useful in enhancing automated model-based assessment processes for engineering applications. An example is the process of Quality Assessment and Quality Control (QA/QC) in construction industry where the owners, clients, and engineering firms can leverage the as-built 3D point cloud models, compare with expected construction performance -- typically in form of semantically rich 3D CAD models known as Building Information Models (BIM) -- and visually inspect the progress and quality of the construction without being physically at the site [32, 13, 12, 11]. For the purpose of model-based assessment, a number of SfM methods has been developed and tested on several scenarios for verifying engineering metrics; e.g., [28, 10]. Those software provide accuracy and efficient for the given tasks. However, none of them address the rubric for assessing the completeness of the 3D image-based models.

Many publicly available SfM software packages such as Bundler [26] and VisualSFM [34] provide easy to use interfaces for 3D image-based reconstruction purposes. These solutions together with other relevant software, leverage a linear time algorithm to incrementally construct the 3D image-based point cloud models. However, they fail to provide an interface -- with proper feedback on completeness of the produced 3D point cloud models -- to add a new photo to the scene after the 3D point cloud has been completely constructed. This issue has been addressed in recent works such as the PhotoCity project [30] in form of a gaming environment, where the introduced method provides a complete structure from motion framework as well as incremental interface. Despite informative feedbacks on the poorly reconstructed areas, such methods do not provide feedback on the location and orientation of the new camera. This is important as in today's application of SfM for engineering practices, often the produced 3D point cloud models from the initially collected photo collections are not complete. Thus, the users has to go back to the site and capture new photos to enhance the completeness of the produced 3D image-based point cloud

models. In term of completeness of an already reconstructed 3D point cloud model, there is no automated method that can decide whether or not the resulting 3D model is complete and/or provide feedback to the user about possible new location and camera poses that can enhance the density of the reconstructed models. The term "completeness" is also used differently in various contexts. For example, [25] introduce a metric for completeness for comparing the performance of multi-view stereo algorithms. Completeness evaluation in [20] compares the vertex in mesh model which is related to our definition. However, both of these methods require specialized tools to generate the ground truth model for comparison purposes. To address such limitations, Tuite et al.[31] introduced the PhotoCity project. The method does not directly address the completeness of the point cloud models but it provides a feedback to the users regarding each area of the point cloud that requires more photographs. Here, the completeness of the area is measured by the density of the points around that area. Different from previous work, we leverage strong priors -- e.g., Building Information Model (BIM) and schedule of a construction project -- to define a rough structure of the scene and provide feedback to the user on the best locations for capturing new photographs. Based on such prior, we provide visual feedback to the users -- e.g., construction field personnel -- on the best "locations" and "viewpoints" for taking new photographs.

To address current limitations, this thesis builds on two recent works: (1) The newly developed image-based 3D reconstruction method of [4] which provides flexibility in several steps of an SfM pipeline; and (2) the PhotoCity project [30], along with linear-time structure from motion algorithm. It particularly two key methods that could be useful as part of both image-based reconstruction and localization systems: First, we introduce a new incremental image-based 3D reconstruction algorithm which will be used when new image are added to the collection of existing photos. Second, the metric for measuring completeness and also the visual interface for the user feedback are introduced. The feedback interface provides the location and viewpoint of the camera whose image can fulfill the requirement on the completeness of the 3D image-based point cloud model. The rest of the thesis is organized as follows: Section 2 provides related work on SfM algorithms. Section 3 gives a detailed information on the developed image-based 3D reconstruction method, the algorithm for incrementation module and the metric for model completeness evaluation. The experiments and conclusions are discussed in Section 4 and Section 5 respectively.

Chapter 2

Related Work

The early incremental SfM algorithms begin with two-view stereo where the initial point cloud is constructed using the feature matching of the initial pair of images. The initial point cloud is expanded by repeatedly adding matched images, triangulating new keypoints, and optimizing the constructed point cloud. However, this type of procedure induces an $O(n^4)$ complexity and is very slow when dealing with large number of images. Bundler [26] -- as an example of these methods -- also suffers a long wait time, typically several hours to a day, for generating a single 3D point cloud. This issue can be addressed in different parts of the algorithm.

Feature detection and extraction can be slow for real-time localization. The initial implementation of Scale Invariant Feature Transforms (SIFT) detector [19] is computationally expensive and is very time consuming. It takes roughly 30 seconds to process a single 8 Megapixels image on a 16 cores workstation. The OpenCV (Open Computer Vision Library) implementation of SIFT makes some improvement in term of speed and can process 8 Megapixels image in few seconds. SiftGPU implemented by Wu [33] improves the speed of SIFT detector by exploiting the architecture of GPU. The shader program is used for this purpose. As a result, it can extract keypoints from 8 Megapixels image in only 1 second. The only limitation for SiftGPU is GPU memory where the size of the image is too big to fit in. For mobile application, the physical memory is even more limited so DAISY descriptors [7] are introduced to overcome the limitation. DAISY still uses histogram of gradients similar to SIFT but it uses a Gaussian weighting and circularly symmetrical key as oppose to image pyramid.

Pair-wise matching is very expensive and a lot of matchings will end up with no matched keypoints. To achieve a faster speed, approximation algorithm can be utilized. Instead of pair-wise matching, Agarwal et al.[1] identify a small number of related image using vocabulary tree recongition [23]. The features are then matched with approximate nearest neighbor algorithm. The experiment shows that two-folded approximation (approximate pairing and approximate matching) algorithm can still provide enough matches for SfM and generates a dense point cloud. GPS tags

are also useful in grouping the images in same vicinity [8] but the tags may not be available in some devices. Wu [34] describes a fast preemptive feature matching. Features of each image are sorted in decreasing scale order. For each pair, only match the first h (= 100) features. If the number of matches is large enough, the matching will resume with the rest of the features. Therefore, most computation time is spent on potential pairs.

In term of image-based localization, the general framework is to compute 2D-3D correspondences between an image and a point cloud followed by pose estimation. The 2D-3D correspondences can be easily computed using any matching algorithm. The early pose estimation for this type of localization is perspective-three-point approach where 3 points are used compute camera orientation and translation in world reference. The algorithm returns four possible solutions which can be disambiguated by the fourth points. Kneip et al.[15] proposes a close-form solution to this problem. To handle outliers produced by the matching, the same approach can be extended using RANSAC loop. The most efficient solution to this problem is ePnP introduced by Lepetit et al.[16] as it is non-iterative and has linear complexity in the number of 2D-3D correspondences. This problem becomes much more harder in robotic field when the real-time localization on a video feed is required. Lim et al.[18] uses a fast keypoint tracking to avoid excessive feature extraction. This results in a very fast localization even on portable devices with low computational power.

In evaluating the completeness of the model, there is not much progress in this area. When geometry of the scene is assumed to be unknown, the evaluation will be based on the density of the point cloud. Kathleen et al.[31] build a competitive game where the players help growing the existing 3D point cloud to cover the entire building. The completeness measurement for this work is identifying which area of the building has not been constructed. This is done by calculating the density at each point. Low density area will be reported as incompleteness and the players need to take pictures that this area is visible. Because the geolocation is not embedded in the point cloud, the point cloud is projected onto Google map and manually aligned. The players should be able to tell from the overlay which part of the scene needs to be photographed. Even though the point cloud is projected onto the Google map, no information from Google map is utilized. Projection onto 2D map will lose its 3D geometry of the scene.

When the geometry of the constructed scene is known, the evaluation will revolve around aligning the ground truth with the constructed point cloud. Seitz et al.[25] gives taxonomy to categorize the multi-view stereo reconstruction algorithm. The byproduct of this categorization is the evaluation methodology. The ground truth is uniform point cloud and the completeness of the reconstructed point cloud is measured by whether the point in ground truth is covered by the point in the reconstructed point cloud, using distance cut-off. Merrell et al.[20] gives an evaluation of large scale scene by comparing it with a mesh model as a ground truth. Both methodologies require the ground truth to be point cloud generated by scanning technique which is not a common rep-

resentation of 3D model such as CAD (Computer-aided Design) and BIM (Building Information Modeling) formats. For a large open scene, the scanning technique will take time to generate the ground truth.

Chapter 3

Method

Our method will be similar to the recently developed image-based 3D reconstruction in [5] and [9, 12] which is capable of performing SfM from unordered collections of photos and/or structured sequences of videos, and also localizes new images to already constructed point cloud models. In this section, we will discuss the current methods used in each component of image-based 3D reconstruction method. Then, we will give an algorithm for model incrementation where new image will be added to the current model. Finally, we propose a new feedback method and a user interface, along with our new metric for evaluation of the completeness of the developed 3D point cloud models.

3.1 The Image-based Reconstruction Method

For our experiment, we leverage the core modules of image-based reconstruction model similar to [5]. The developed prototype follows the general incremental SfM guidelines similar to other existing SfM software. First, from an unordered collection of images, feature keypoints are detected for each image, independently. Then, the keypoints are matched for each pair of images. The similar keypoint presents across multiple images will form a track. After that, initial pair of images is chosen and the initial point cloud is generated. Then, the rest of the images, one at a time, is localized followed by non-linear optimization. We prefer this project to existing SfM software due to parallel computation and GPU exploitation. In this section, we will give an overview of this method.

3.1.1 Camera Parameters

Suppose we have a 3D point $P = [x\ y\ z\ 1]^T$ in homogeneous coordinates, then following pinhole camera model, we can use a camera matrix to map P to 2D feature point $p = [u\ v\ 1]^T$

which is its corresponding 2D location on the image. The camera perspective projection matrix can be decomposed into K and $[R\ t]$. K is the intrinsic camera matrix, while R and t are the extrinsic parameters. K follows the following form:

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where f is the focal length and (c_x, c_y) is the camera center. f can be extracted from EXIF tag of the image (usually fixed across all images as many of the image collections in our case are captured using the same camera and under the same conditions). (c_x, c_y) is estimated to be (width/2,height/2). R is a 3x3 rotation matrix and t is a 1x3 translation vector. The complete camera equal is given by $w \cdot p = K[R\,t]P$ where w is a scalar factor due to uncertainty in scale of the real world.

3.1.2 Feature Detection and Extraction

The developed image-based 3D reconstruction method, offers a variety of feature detectors such as Scale Invariant Feature Transforms (SIFT) [19] and Speeded-up Robust Features (SURF) [6]. SIFT detector creates an image pyramid and filters each layer with Difference of Gaussian (DoG). SURF, in contrast, creates a stack of pyramid without downsampling (for higher level) and uses box filter approximation of second-order Gaussian partial derivatives. SURF is faster than SIFT due to its implementation and GPU usage.

Once we detect a feature in each image, we need to encode the information for keypoints so that they can be compared in the matching step. The developed image-based 3D reconstruction system offers 4 type of feature descriptor extractions: SIFT and Binary Robust Invariant Scalable Keypoints (BRISK) [17] are CPU-based implement while SURF and Fast REtinA Keypoint (FREAK) [2] are GPU-based implementation. FREAK uses retinal sampling patterns to compare image intensities and produces a cascade of binary strings. BRISK assembles a bit-string descriptor from intensity comparisons retrieved by dedicated sampling of each keypoint neighborhood. Both FREAK and BRISK utilize small disk space compared to SIFT and SURF and only require hamming distance for descriptor comparison as oppose to Euclidean distance. The color is also extracted for visualization purposes. The color of the 3D points will be the average color of all corresponding detected keypoints across all images that see that 3D point.

We can integrate the detector and extractor to suit our need. The common combinations are SIFT-SIFT, SIFT-SURF, SURF-FREAK and SURF-BRISK. In our experiment, we only use SIFT-SIFT as our initial experiments shows that this combination -- while not the fastest -- can provide the most reliable point cloud models. Figure 1 shows the detected features of one image.

Figure 1: SIFT detector returns 11919 keypoints on a 2144 x 1424 image (SD9 data set). Small dots indicate the center of keypoints while their color shows response level of keypoints

3.1.3 Feature Matching and Forming Tracks

The matching module used in the image-based 3D reconstruction method, is the fast approximation nearest neighbors (FANN) [21]. Depending on the feature descriptor of choice, FANN library performs differently. In the case of FREAK and BRISK, hierarchical clustering tree is used. For SIFT and SURF, randomized k-d tree searching algorithm is used. Regardless of the method, the inconsistent matches can be filtered by using a distance ratio test. The matching can be further improved by using fundamental matrix and homography matrix. The fundamental matrix enforces the epipolar geometry and controls the maximum distance from a matching keypoint to the corresponding epipolar line; while the homography matrix controls how far the matching points can be when projecting one image to another. The matching step usually produces several inconsistent pairs, and thus both fundamental and homography matrices are estimated using a RANSAC (RANdom SAmple Consensus) loop with normalized Direct Linear Transform [14] to detect the most consistent transformations and also filter the false positives. The thresholding equations are the followings:

$$||x_i^T F_{i,j} x_j|| \geq \delta_F = \max(w_i, h_i, w_j, h_j) \times 0.006 \tag{1}$$

Figure 2: (Top) FANN matcher returns 685 matchings. Although it is robust to viewpoint change, some bad matches such as the lower-right keypoint of the right image still persist. (Bottom) Tracker groups similar matched keypoints across four images. Bad matches are further filtered.

$$||x_i^T - H_{i,j}x_j|| \geq \delta_H = \max(w_i, h_i, w_j, h_j) \times 0.004 \tag{2}$$

where x_i and x_j are homogenous coordinates of the matched keypoints in image i and j, $F_{i,j}$ is the fundamental matrix and $H_{i,j}$ is the homography matrix between image i and j, and (w_i, h_i) and (w_j, h_j) are the width and height of image i and j. The homography score (H-score) is then calculated by the percentage of matches satisfying equation (2). It is important to note that H matrix estimation uses the inliers from F matrix calculation.

The tracking module is simply grouping the matching keypoints from multiple images. The track length is the number of images that contain similar keypoints. The minimum track length can be specified and used for filtering out short tracks. By grouping the keypoints together, we can further remove bad keypoints by comparing it the rest of the keypoints of the same track. This is called a track ratio-test. Figure 2 shows the result of matching one pair of images and tracking the matched keypoints across multiple images.

3.1.4 Structure from Motion Algorithm

The SfM algorithm estimates a set of camera parameters, such as focal length, rotation matrix, and translation vector for each image along with and a 3D location for each reconstructed point. Similar to the Bundler and VisualSFM, an incremental approach is used to recover few cameras at a time. The small difference is that the SfM method used in this work is that it stores the feature de-

scriptor for each 3D point. This will come into play when we want to do direct 2D-to-3D matching in image-based localization. Algorithm 1 shows the overall procedure of this algorithm.

Algorithm 1: SfM Algorithm

Input: Tracks with feature descriptors

Output: Camera parameters and 3D points

```
Select initial image pair
while there exists unregistered images do
    Pick a new unregistered image
    Estimate 3D camera pose
    Run Local Bundle Adjustment
    Triangulate new 3D points
    Perform Cheirality test on new 3D points
    Run Global Bundle Adjustment
    Remove erroneous 3D points
end
```

The first step is to select initial image pair and apply Nister's five-point algorithm [22] to recover the camera parameters. The initial pair should have a sufficiently large number of matches and, at the same time, have a long separation distance between the cameras. This is to prevent solution close to local minima in optimization process. These requirement can be met by selecting the pair with the lowest H-score. In image-based 3D reconstruction experiments, H-score should be greater than 0.25 whilte the number of matches should be more than 200. After pose estimation, the initial 3D point cloud is created through triangulation. This point cloud is further optimized by invoking Bundle Adjustment optimization, which minimizes the overall mean re-projection error. The optimization package used in this step will be discussed in the next section.

The next step is an iterative process to register one camera at a time. The criteria for picking new camera is by the number keypoints that already have their 3D points correspondence. The new camera will then be calibrated by using the already triangulated 3D points. This can be done using PnP (Perspective-n-Point) camera estimation method with RANSAC and Levenberg-Marquardt optimization [14]. Upon success, the local bundle adjustment is called. In this step, the camera parameter of the new image will be optimized while fixing the camera parameters of registered images. After that, the new 3D points are triangulated. Cheirality test, which is a depth test, will ensure that the new 3D points are indeed in front of the camera. We also remove points with high reprojection error afterward because very little number of high-error 3D points can destroy the entire shape of the 3D point cloud during Global Bundle Adjustment step. The erroneous 3D

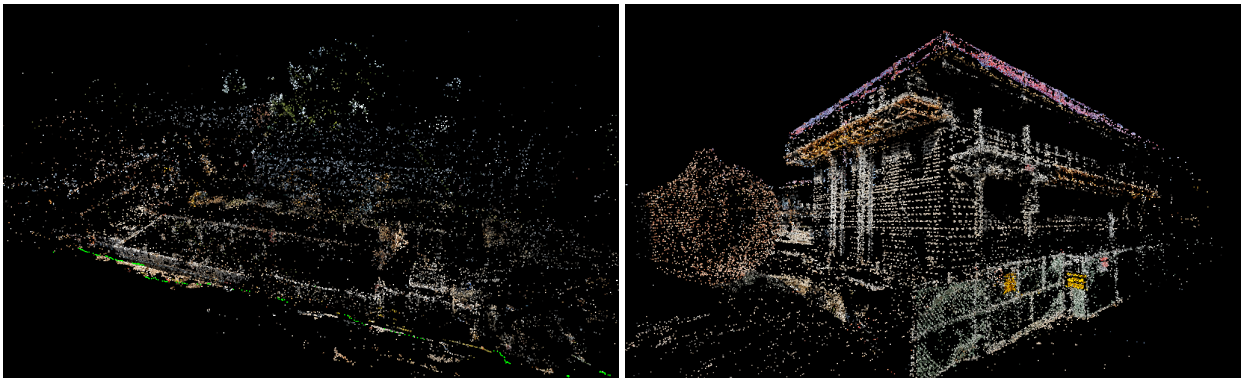


Figure 3: (Left) The point cloud generated from RH4 data (Right) The point cloud generated from SD9 data.

points are to be removed as part of the standard routine to avoid noise and outliers. Examples of point cloud are shown in Figure 3.

3.1.5 Bundle Adjustment

Bundle adjustment is an optimization problem that refines a visual reconstruction to achieve optimal 3D structure and camera parameter estimates. Mathematically, given a set of image 2D feature locations and corresponding 3D world coordinates, the goal is to minimize the reprojection error [29]. The error is often regarded as an L_2 norm of the difference between the 2D feature locations and the projected points from 3D onto the image plane. This type of problem is also called non-linear least squares problem.

If x is a vector of parameters (3D points, axis rotation vector, translation vector, focal length) and $f_{i,j}(x)$ is the residual error for projecting point i onto image plane j, we wish to solve

$$x^* = \underset{x}{\operatorname{argmin}} \sum_{i,j} ||f_{i,j}(x)||^2 \tag{3}$$

The algorithm of choice for this type of problem is the Levenberg-Marquardt (LM) algorithm [24]. This algorithm iteratively solves the regularized linear approximation of equation (3) in the form

$$\delta^* = \underset{\delta}{\operatorname{argmin}} ||J(x)\delta + f(x)||^2 + \lambda||D(x)\delta||^2 \tag{4}$$

where $J(x)$ is the Jacobian of $f(x)$ and $D(x)$ is the square root of the diagonal matrix $J(x)^T J(x)$. λ is a non-negative parameter that control the strength of regularization (sometimes called Lagrange multiplier).

There are several optimization package that offer a competitive implementation of LM algo-

rithm. The solver of choice for developed image-based 3D reconstruction is Multicore Bundle Adjustment package [35] (referred to as PBA). This package exploits the GPU architecture which offers a fast computation for Jacobian matrix in multiple GPU threads. It solves the optimization problem involving 14,000 cameras, 4,500,000 points and 30,000,000 measurements (projected 2D points) in only 2 minutes. This nice property will help us in toward real-time localization module and model incrementation.

There are two modes for this method: The motion-only mode is used in Local Bundle Adjustment step while full mode is used in Global Bundle Adjustment step.

3.1.6 Localization Module

Localization algorithm in the developed image-based 3D reconstruction, estimates a camera pose of the new image (most likely not part of the images used to construct 3D point cloud). The approach is similar to when the new image camera is added in line 4 and 5 of Algorithm 1. Algorithm 2 shows the steps in detail.

Algorithm 2: Localization Algorithm

Input: 3D point cloud and new image

Output: Camera parameters for new image

1 Extract keypoints from the new image
2 Compute matching with the 3D points
3 Estimate 3D camera pose using Direct Linear Transform (DLT)
4 Run Local Bundle Adjustment

We begin the process by extracting keypoints from the new image. It is important to use the same detector type and descriptor type as when we construct the point cloud. This will allow matching to work properly. The matching in this algorithm compares the 2D keypoints descriptors from the new image and 3D point descriptors from point cloud. To guarantee a one-to-one mapping, only one descriptor will be stored for each 3D point even if there are multiple keypoints corresponding to it. The heuristic algorithm for deciding the descriptor representation of 3D point is explained in Algorithm 3. The camera calibration algorithm is then performed by solving the Direct Linear Transformation (DLT) equation followed by a Levenberg-Marquardt optimization [14]. This model-based camera calibration results in 6-DOF (degrees-of-freedom) localization in 3D space and thus gives high localization accuracy despite possible variation in the position and orientation of the user within the reconstructed scene.

Algorithm 3: Average Descriptor Algorithm

Input: 3D point and its 2D corresponding image points

Output: Representative descriptor

```
for each 2D corresponding image point do
    if descriptor is SIFT or SURF then
        Sum Euclidean distance to all other descriptors
    else if descriptor is BRISK or FREAK then
        Sum Hamming distance to all other descriptors.
    end
end
Select the descriptor with the minimum sum as a representative descriptor
```

3.2 Model Incrementation

When new image is introduced to the scene, we can compute the matching between this image and the rest of the collection and proceed through the normal incremental structure from motion. However, as the number of images in the collection grows, this process will be slow and inefficient. Most images will have zero matching. Therefore, we need more efficient algorithm.

Algorithm 4: Model Incrementation

Input: 3D point cloud and new image

Output: Incremented 3D point cloud

```
Localize new image to retrieve camera parameter
Select a set of images C from existing images whose number of matched 3D points is greater than k
Match new image against images in C
Triangulate new 3D points
Run Bundle Adjustment on new 3D points and filter erroneous points
```

The algorithm for model incrementation is shown in Algorithm 4. In the first step, we localize new image with respect to the input 3D point cloud. The procedure for this step is exactly the same as Algorithm 2. Because we have a set of matched 3D points from the first step, we can use the visibility of those points to find images that have a high probability of producing sufficient

number of matches against new image. For fast computation, we store track information (all 2D corresponding keypoints and image ID) for each 3D point. Therefore, we only need to compute matches against the images whose 3D points are matched to new image. The rest of the steps is exactly the same as step 6 through 9 in Algorithm 1. It is important to note that only new 3D points whose 2D correspondences have not been triangulated will be added to existing point cloud, to avoid redundancy.

There is one parameters in this algorithm. The minimum number of visible points k determines how connected the new image and the old image are. The choice of k will depend on the density of the points but it can also be chosen as to limit the number of images that will be matched against new image. This is to speed up the process since matching is the bottleneck of the entire process.

3.3 Completeness of Point Cloud

The completeness of the point cloud defines how complete the point cloud is with respect to the real object or scene it represents. To be able to measure such completeness without a ground truth, we need to make assumptions.

1. The objects in the scene are closed and their boundary is continuous.
2. The density of the feature keypoints is uniform on the boundary.

Therefore, we can measure the completeness of the area around a particular point P by the **density** of the points around P with radius r. Because of the uniformity assumption, the area around every points on the surface must have the same density so the surface that is not fully reconstructed will have low density of point cloud. The closeness and continuity of the boundary will guarantee that the low density area can be extended further. The reason that we want to measure the density around the point instead of pre-defined region is we don't know the geometry of the scene. With this definition, we can derive an algorithm for finding incomplete area. For simplicity, we will call incomplete area a flag.

There are 4 parameters in Algorithm 5. r_1 is the radius of the sphere around point P. The choice of r_1 should depend on size of the size of the point cloud (such as diameter) as we want to avoid sphere covering the entire object. r_2 is minimum distance between two flags. This requirement will reduce the number of the flags and avoid redundancy. Normally, $r_2 > 2 \cdot r_1$ so the area that two flags covers does not overlap. The choice of r_2 should also depend on the size of the point cloud. d_{min} is the minimum density required to be a flag. The choice of d_{min} will depend on the feature response on the surface. For textureless surface, d_{min} will be much lower. However, we do not know the nature of the scene so we can use heuristic approach and pick d_{min} based on the

percentile of the overall density. c is the cutoff parameter to avoid flagging isolated points that are less likely part of the objects in the scene. The choice of c will depend the density around noises which may be hard to determine. One heuristic is to set c to some fraction of d_{min}.

Algorithm 5: Point Cloud Flagging

Input: 3D point cloud

Output: Flags

```
for each 3D point P do
    Compute the number of points n_p around point P
    with distance less than r_1
    if c < n_p < d_min then
        Flag P
    end
end
Sort flagged points by their density n_p
for each flagged points F in sorted order do
    if There is another flagged point within r_2 radius
    then
        Unflag F
    end
end
```

As the model grow, the density of flags will exceed d_{min}. In this case, we can unflag them and find new flags. The new flags will come from new points using the same algorithm as in Algorithm 5.

3.4 Feedback on Location and Viewpoint of New Cameras

It is very informative to know which area of the point cloud is incomplete. It is even more useful to be able to suggest where new images should be taken in order to complete that area. This information can be sent to an automated robot that can fly around the scene and take photos for us. Before we explain the algorithm, let us look at how the camera locations are arranged. For a closed object, the camera location should be at some distance away from the surface. To capture a complete object, the camera locations should form a similar shape as the object itself. When we have an incomplete shape, finding camera location is equivalent to completing incomplete shape, which is not trivial. Instead, we assume the smoothness of the object surface (first derivative continuous) and try to extend the current camera location. We decide to use Bezier curve to model the camera

location path. Bezier curve is parametric curve that is shaped by 4 control points. Because of that, we are able to interpolate or subdivide the curve for the section that we want. We can also try to extrapolate the curve to reach the incomplete area but this may not be good for sharp turn or corner of the building. To solve this issue, we introduce one additional control point for each flag. This control point will form a ray (same as surface normal) to the flag that is orthogonal to the surface at the flag. The distance δ between the control point and the flag can be estimated by the projected ray ρ from the closest camera onto surface normal η, which is shown in Figure 4. Hence, the suggested camera location is a Bezier curve extending from the current camera location path to control point.

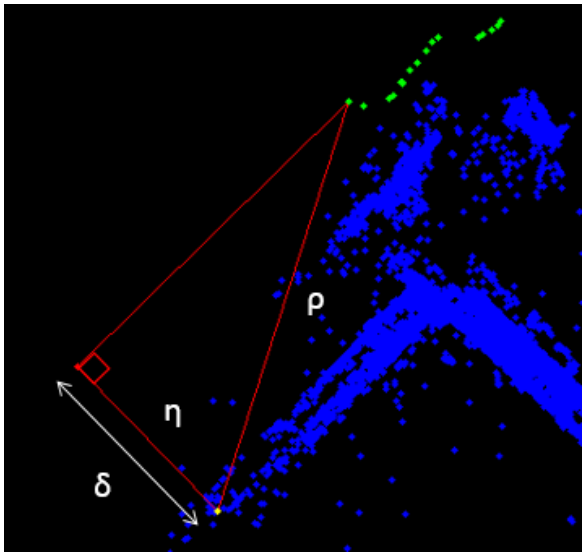


Figure 4: The ray ρ from closest camera to the flag is projected onto surface normal η. Its length is the distance between the flag and control point

Algorithm 6: Up Vector Selection

Input: Camera Rotation Matrices

Output: Rotation Matrix

1 $r_{g1} = \text{min eigenvector}(\sum_k r_{k0} r_{k0}^T)$

2 $r_{g0} = N((\sum_k r_{k2}) \times r_{g1})$

3 $r_{g2} = r_{g0} \times r_{g1}$

Estimating camera location for 3D point cloud is difficult because the camera path may not cross the surface normal at the incomplete area. To simplify the problem, we choose to project the point cloud into 2D space. However, the point cloud may not align well with the real world coordinate so we need to transform it so that one dimension can be ignored. We use up vector selection algorithm mentioned in [27] to transform the point cloud. The outline of up vector selection is shown in Algorithm 6 where r_{ki} is the (i+1)-th row of the k-th camera matrix and r_{gi} is the (i+1)-th column

of the desired rotation matrix. This up vector algorithm will be useful later when visualizing the point cloud on a portable device. The projected 2D points will only be used in estimating surface normal and control point. The rest of the algorithm will use complete set of 3D points. Algorithm 7 describe the high-level idea of the entire process.

Algorithm 7: Camera Location Suggestion

Input: 3D point cloud and flags

Output: control points of Bezier curve for each flag

1 **for** *each flag* F **do**
2 Form a covariance matrix Σ from the 3D points near F
3 Compute surface normal at F by choosing the smallest eigenvector of Σ
4 A control point cpt is the closest point on the surface normal ray to the closest camera location
5 Compute 4 control points for Bezier curve that goes through cpt and the rest of cameras by solving a least linear square problem
6 Perform De Casteljau's subdivision to obtain the last segment of the curve
7 **end**

Chapter 4

Experimental Results

4.1 Experimental Setup and Datasets

In our experiment, we validate the applicability of the proposed method for image data collected from two construction projects: Student Dining (SD) and Residential Hall (RH) programs buildings on campus of the University of Illinois at Urbana-Champaign. We used five different visual data sets captured from these two construction projects; RH4, RH6, RH7, SD9 and SD10. RH3, RH4, RH6 and RH7 are taken from the same construction site but different lighting condition and date while SD9 is the outside and SD10 is the inside of the same building. We use NIKON D300 camera to take all of the images. This allows us to subsample the images spatially for other experiments. With an exception of SD10, the image spatial resolution is 2144 x 1424 pixels.

Figure 5: (Top) Sample images from RH4 (Bottom) Sample images from SD9

The quantitative information of each data set is summarized in Table 1. Sample images from RH4 and SD9 are shown in Figure 5. The overlap of image on the full point cloud -- one for each construction project -- is shown in Figure 6. Finally, Figure 7 shows the camera frames (shown

both location and viewpoint) for each of the construction sites.

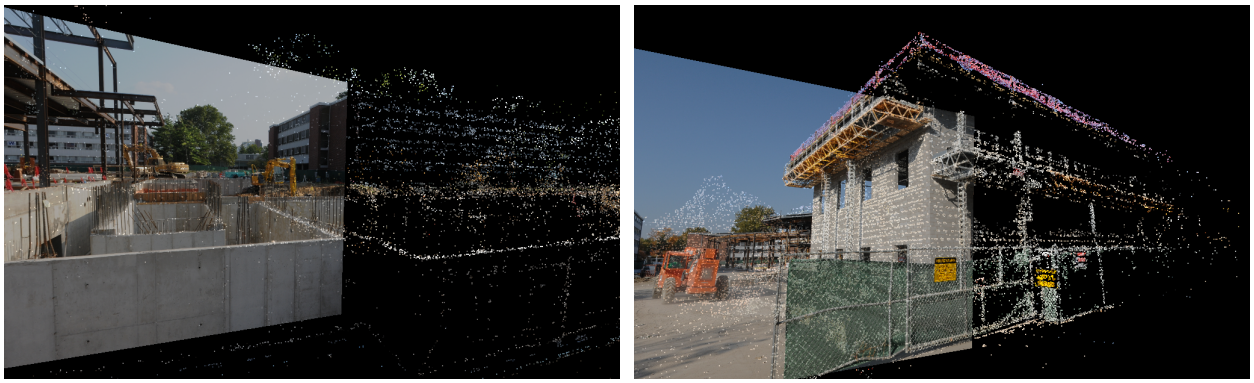


Figure 6: (Left) RH4 image-based 3D point cloud model with overlay image (Right) SD9 image-based 3D point cloud model with overlay image

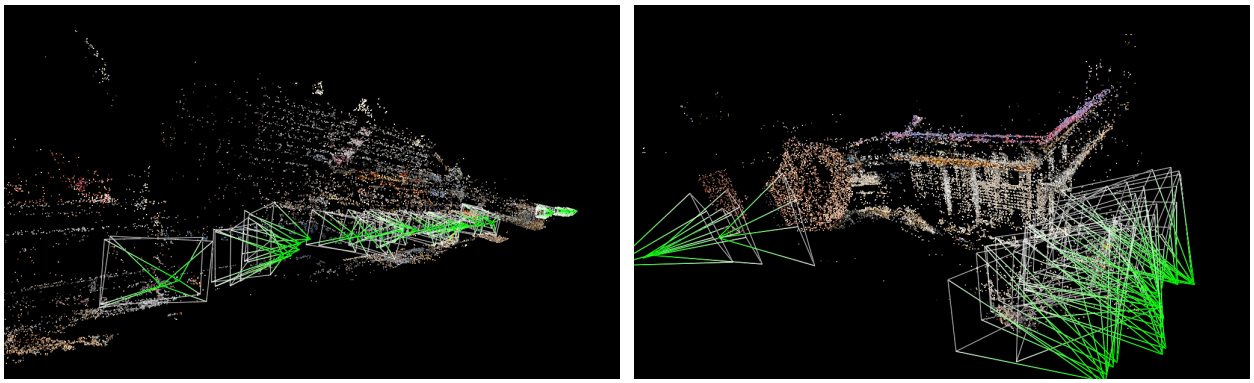


Figure 7: (Left) RH7 image-based 3D point cloud model with camera frames visualized as frustra (Right) SD9 image-based 3D point cloud model with camera frames visualized as frustra

4.2 Model Incrementation and Completeness Measurement

First, we will look at the reconstructed 3D point cloud model for each data set. We then select between 20% - 50% of the data set for initial reconstruction (we call it a seed for simplicity). The result is shown in Figure 8 and Figure 9. For SD9 data set, 25 out of 51 images are registered. Similarly, 56 out of 147 images are registered for RH7 data set. Finally, 117 out of 123 images are registered for SD10 data set.

Figure 8b shows that RH4 Seed is missing the area to the right of current point cloud compared to Figure 8a. Figure 8d is missing the interior of the front structure and the right part of that structure. Figure 8f is missing the detail beyond the wall to the right. For SD data set, Figure 9b is

Data Set	Spatial Resolution	Illumination and Weather Conditions	Number of images
RH4	2144 x 1424	Sunny	160
RH6	2144 x 1424	Cloudy	114
RH7	2144 x 1424	Weak Sunlight	147
SD9	4288 x 2848	Outdoor	51
SD10	2144 x 1424	Indoor	123

Table 1: Quantitative summary of the data set used in our experiments

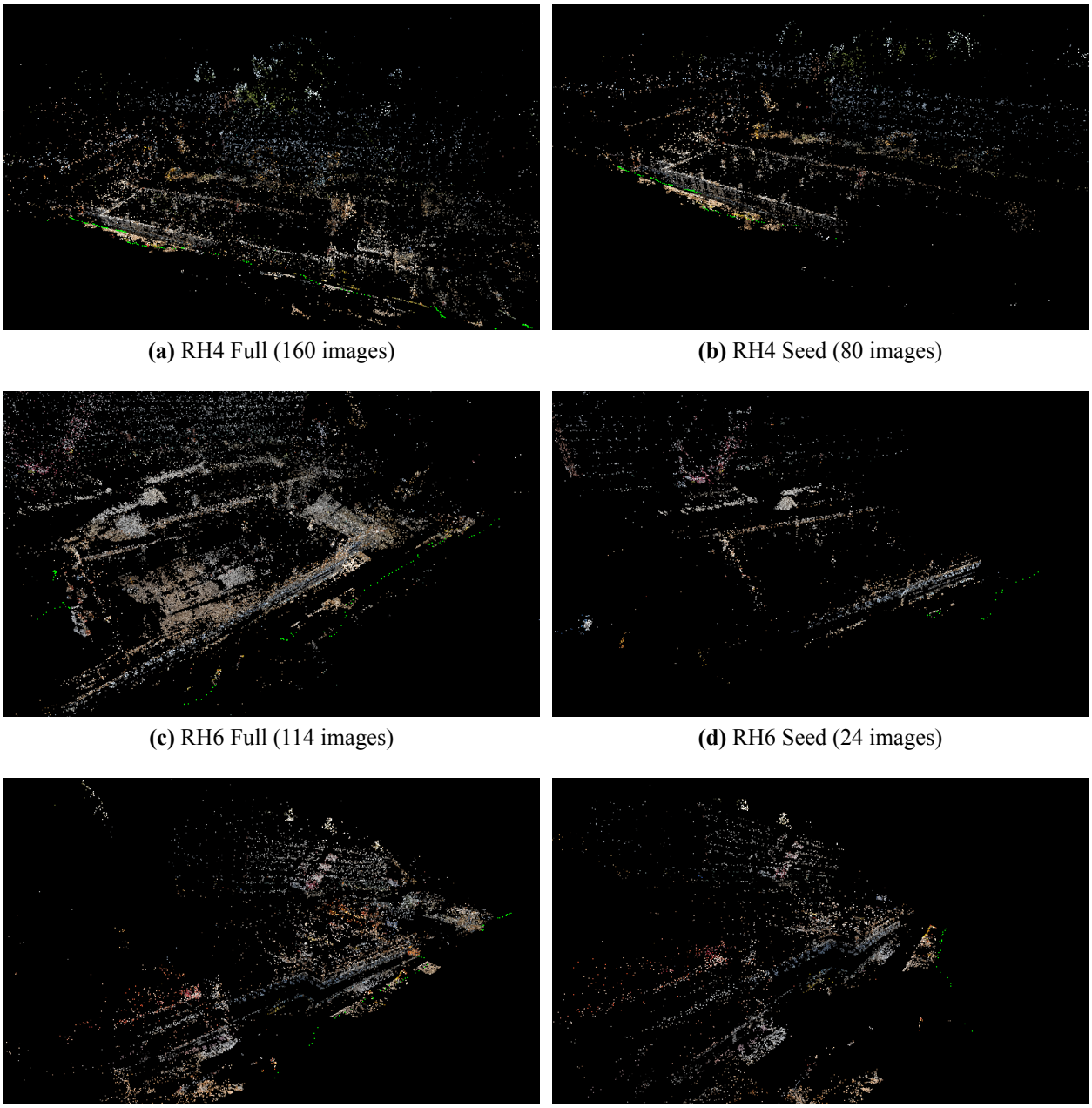
missing the extension of the building on both ends while Figure 9d does not reach the left part of the interior.

Next, we will look at the initial flags on each of the seed. Table 2 shows the parameters for each data set. We set r_1 to be one eighth of the smallest dimension diameter (not greater than 1) and $r_2 = 2 \cdot r_1$. d_{min} is the 3rd percentile of density distribution while $c = 0.2 \cdot d_{\text{min}}$. In Figure 12 and 10, the initial flags are shown along side with their normalized density plot. The density plot is a visualization showing the number of points (n_p in Algorithm 5) around a particular point. For the density plot shown in the right column of Figure 12 and 10, red spectrum indicates high density while green color indicates low density. You can see the main structure building in each data set has several high density points and the density slowly drops as we go further away from the center. In Figure 12f, the area between two high density point clusters has low density points so two flags are generated to account for this information shown in Figure 12e.

Figure 12a and 12c shows the result of flagging algorithm on RH4 and RH6 Seed point cloud. In both examples, the location of 5 closest flags to the camera cluster happens to be on the isolated noises due to excessive amount of background noise affecting density distribution. For RH7 Seed point cloud, Figure 12e shows that 2 out of 12 closest flags are on the building structure. Due to the distance between camera being large, r_1 and r_2 are allowed to be larger than normal. Figure 10a shows the result of flagging algorithm on SD9 Seed point cloud. 6 flags are generated for which two are on either end of the wall. In the case of SD10, the scene is indoor so parts of the floor and ceiling are reconstruct. Therefore, the flagging on the projected point cloud shown in Figure 10c

Data Set	d_{min}	c (cutoff)	r_1	r_2
RH4	90.53	18.106	1	2
RH6	22	4.4	1	2
RH7	20	4	2.5	5
SD9	215.34	43.0680	0.9402	1.8804
SD10	22	4.4	0.8968	1.7936

Table 2: Parameters used for each data set



(a) RH4 Full (160 images)

(b) RH4 Seed (80 images)

(c) RH6 Full (114 images)

(d) RH6 Seed (24 images)

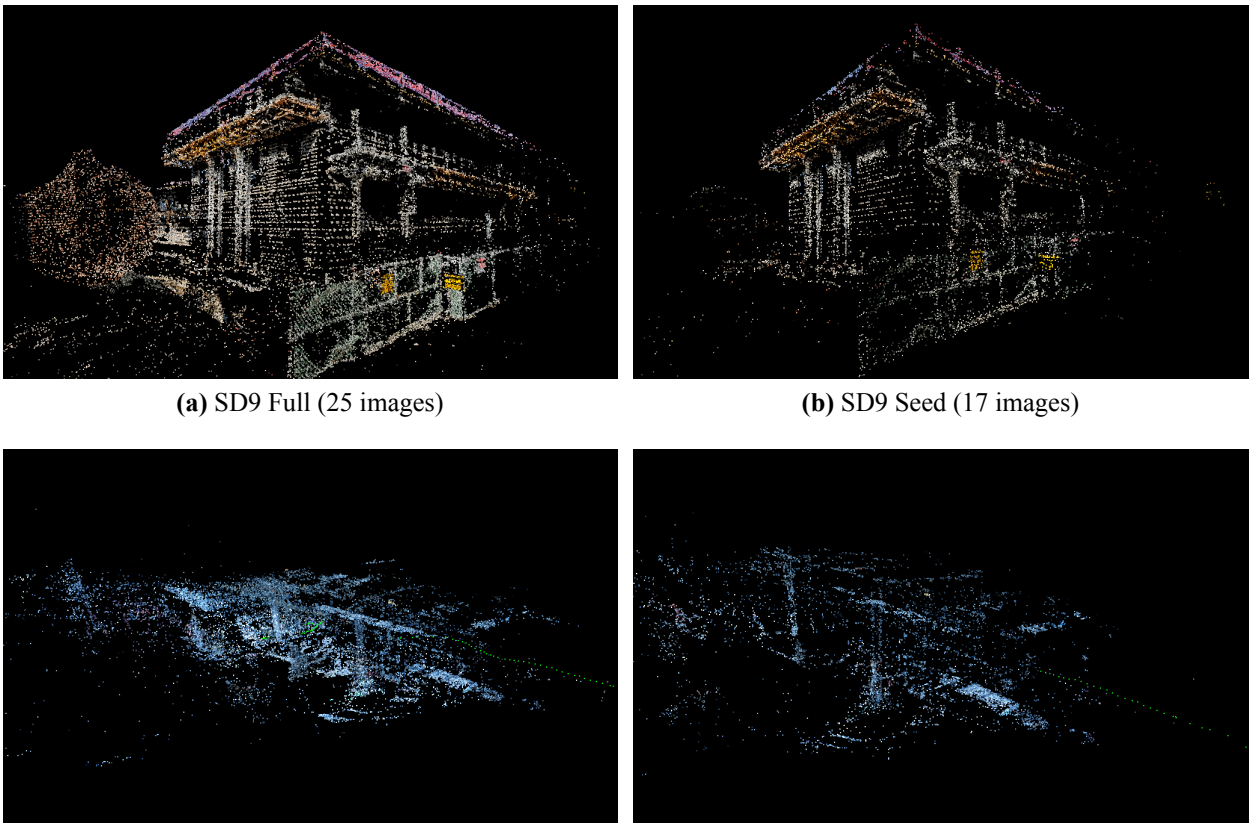
(e) RH7 Full (56 images)

(f) RH7 Seed (20 images)

Figure 8: Image-based 3D point cloud models of different RH data sets. For each set, the number of images used to produce the point cloud model is indicated at the bottom of the subfigure

looks messy compared to the other outdoor point cloud. 2 flags have a very close control point due to the surface normal being parallel to the camera locations curve.

We will use SD9 data set as an example to show how Model Incrementation will affect the flags. Figure 11 shows the development of the point cloud after 4 images are added one at a time. For simplicity, one flag that is affected throughout the entire process will be color with pink (as oppose



(a) SD9 Full (25 images)

(b) SD9 Seed (17 images)

(c) SD10 Full (117 images)

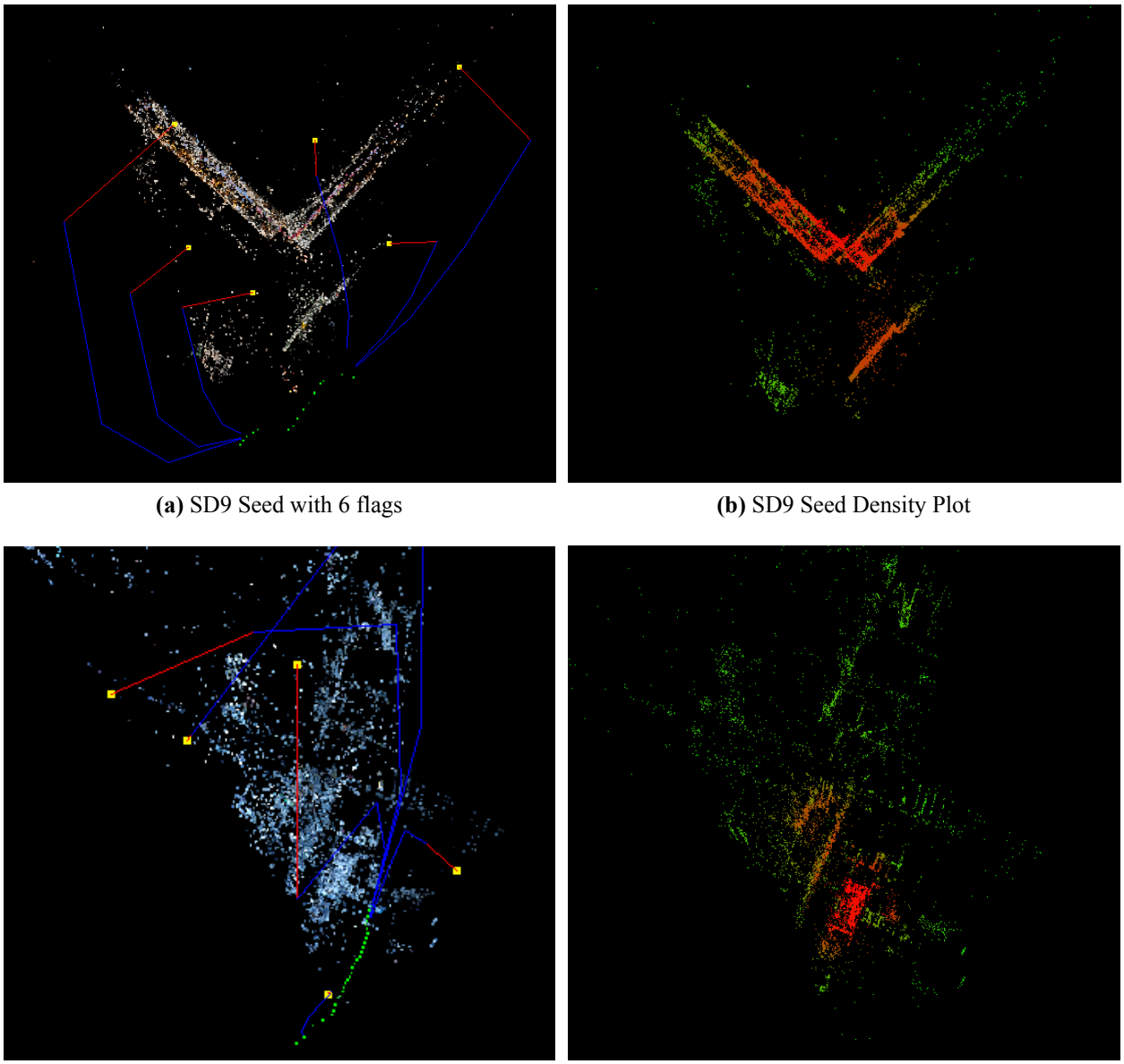
(d) SD10 Seed (30 images)

Figure 9: Image-based 3D point cloud models of different SD data sets. For each set, the number of images used to produce the point cloud model is indicated at the bottom of the subfigure

to yellow). Figure 11b shows that the new camera position is at the upper-right-most green dot. This results in a denser point cloud along the right wall. However, the density does not exceed d_{min} so the flag does not disappear. In Figure 11c, the wall is extended further but the the density at the flag is slightly below the d_{min}. You can clearly see that, at each step, the location of new cameras being added follow the line to the control point of the flag. After adding 4th image, the density at the flag surpass d_{min} so the flag is removed in Figure 11d. Upon removal, new flag (orange color) is being generated and is placed at the further end of that wall.

For statistical results, we consider two quantitative values. First, the success ratio is measured by the ratio between the number of actual new images that are being added to the point cloud and the number of expected images that should be able to localized and added to the point cloud. The high ratio will indicates that the localization performs well and is robust to noise during pose estimation step. Figure 13 shows the success ratio on all data set. SD data is dominating in term of success ratio while RH6 and RH4 suffers a low end of spectrum. There is no direct correlation between the success ratio and the number of images. Second, the match ratio is measured by the

ratio between the number of images that are actually being matched in step 3 of Algorithm 4 and the number of images in the current point cloud. The low ratio will induce less number of matching pair, resulting in the faster model incrementation. In Figure 14, you can see that RH4 which has the highest number of images has a very low ratio. In fact, the ratio is low for all RH data set compared to SD data set. This demonstrates that for a large point cloud, only a handful of images is needed for the matching step and there is no need to match new image against all registered images.



(a) SD9 Seed with 6 flags

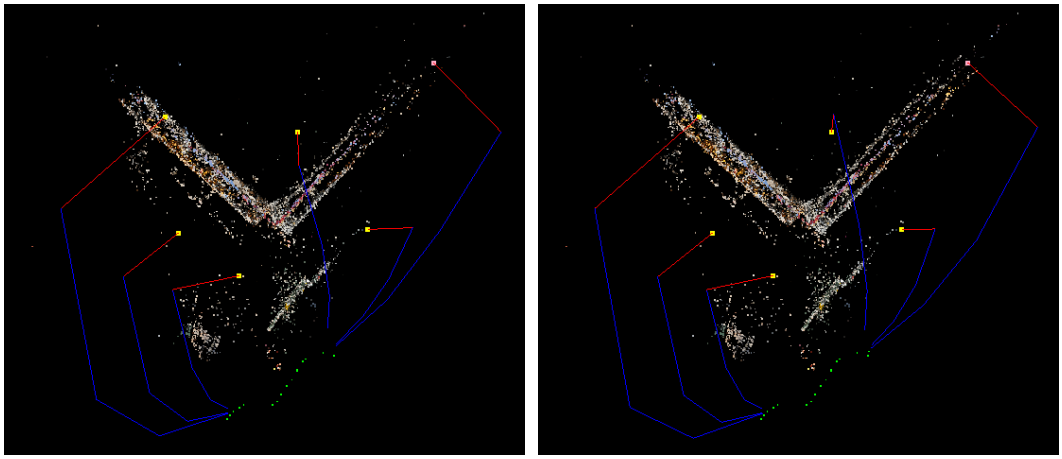
(b) SD9 Seed Density Plot

(c) SD10 Seed with 5 closest flags

(d) SD10 Seed Density Plot

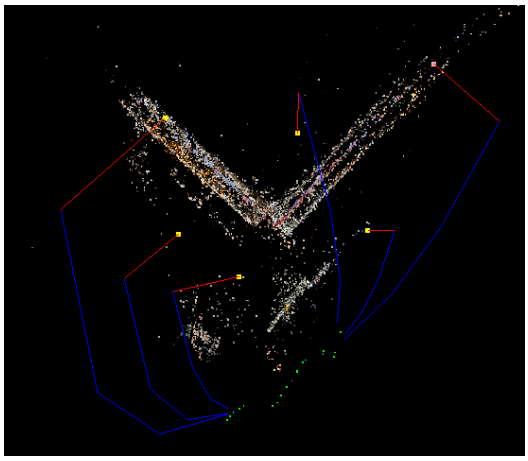
Figure 10: SD Point cloud with initial flags from top view. Yellow dot is the location of flag while blue line indicates the control point path for Bezier curve. Red line is the surface normal at the flag and the green dots are camera locations.

It is important to note that success ratio graph and match ratio graph follows the same trend. The match ratio also indicates how far each camera is with respect to each other because if the camera is clustered, the number of points that is visible by many cameras should be high. Points with high visibility from several images will be optimized better due to more information. Therefore, they improve the localization step and result in a higher chance of registering new images.

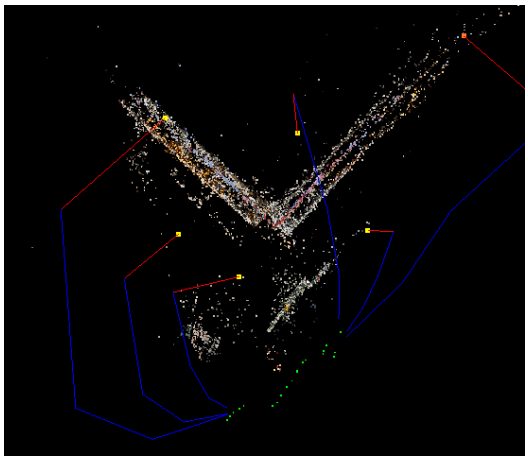


(a) SD9 Seed Point Cloud

(b) SD9 Seed + 1 Image

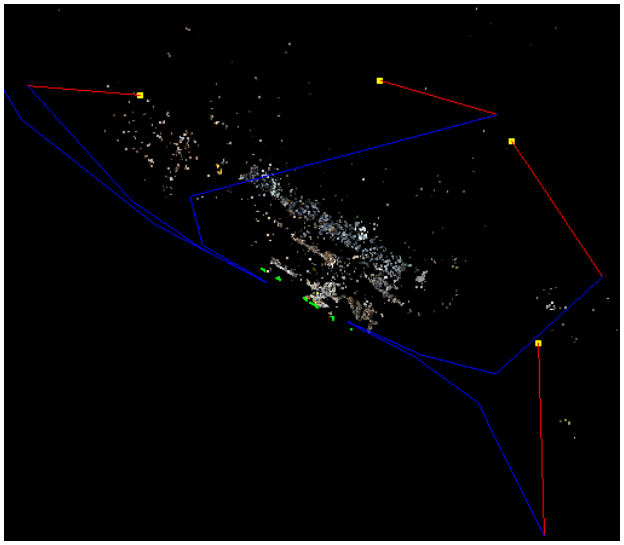


(c) SD9 Seed + 2 Images

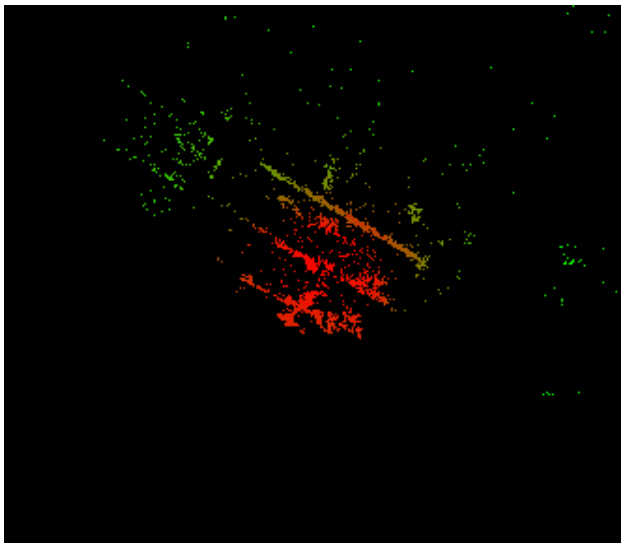


(d) SD9 Seed + 4 Images

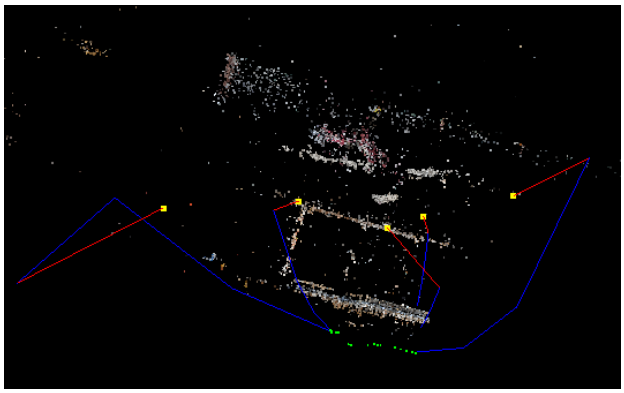
Figure 11: The development of the point cloud on SD9 data set. The order is from left to right and top to bottom respectively. Pink dot represents the flag of interest



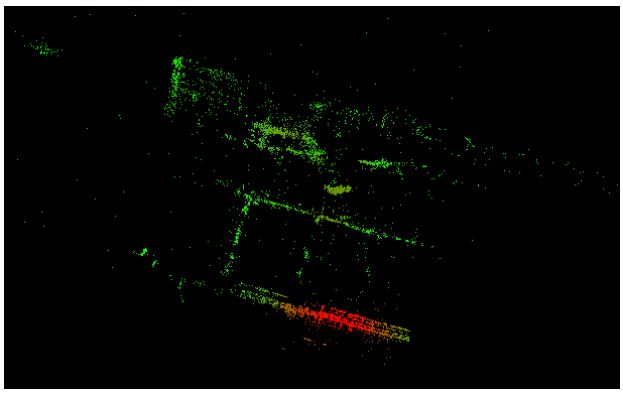
(a) RH4 Seed with 5 closest flags



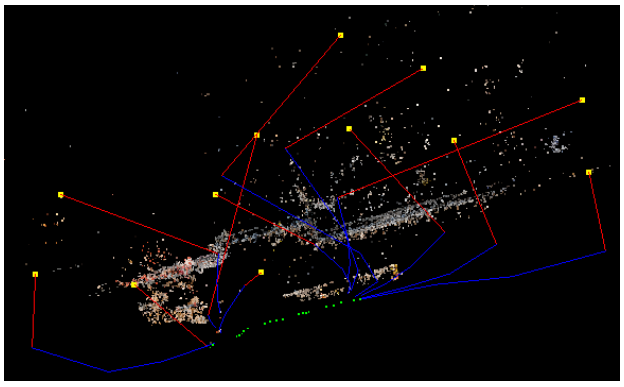
(b) RH4 Seed Density Plot



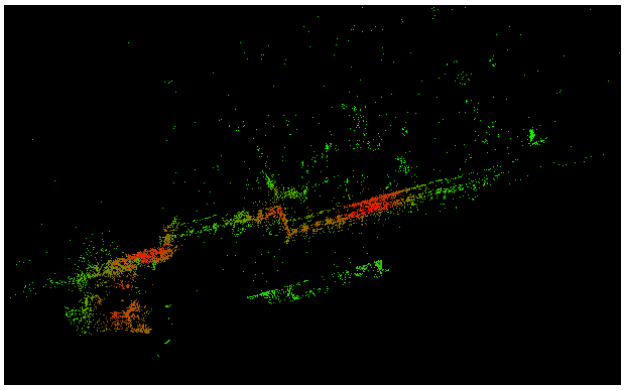
(c) RH6 Seed with 5 closest flags



(d) RH6 Seed Density Plot



(e) RH7 Seed with 12 closest flags



(f) RH7 Seed Density Plot

Figure 12: RH Point cloud with initial flags from top view. Yellow dot is the location of flag while blue line indicates the control points path for Bezier curve. Red line is the surface normal at the flag and the green dots are camera locations.

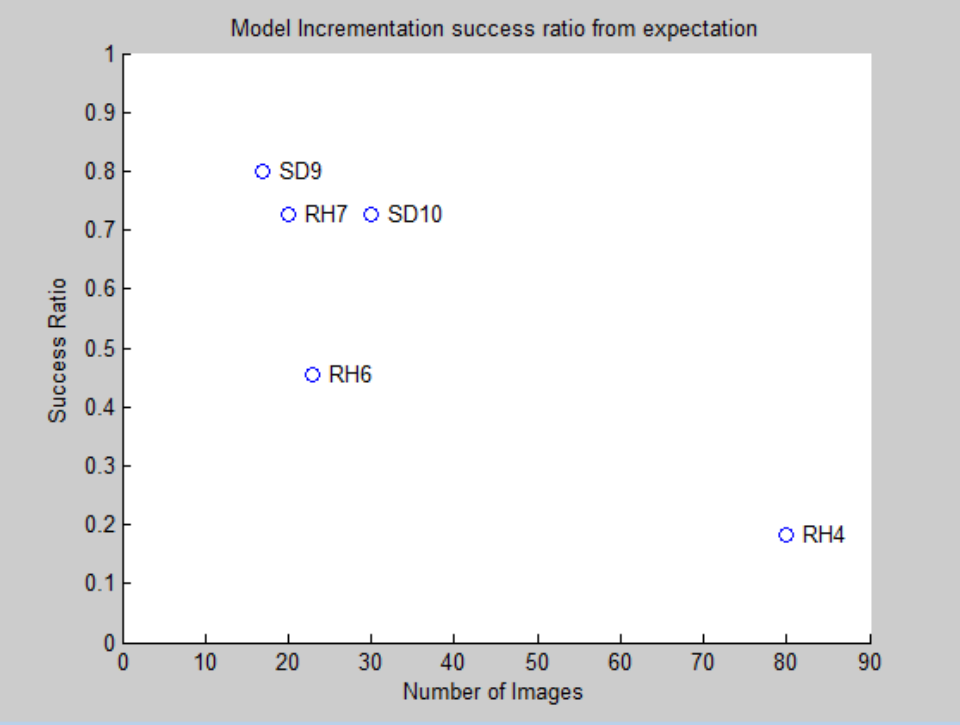


Figure 13: Success Ratio Plot

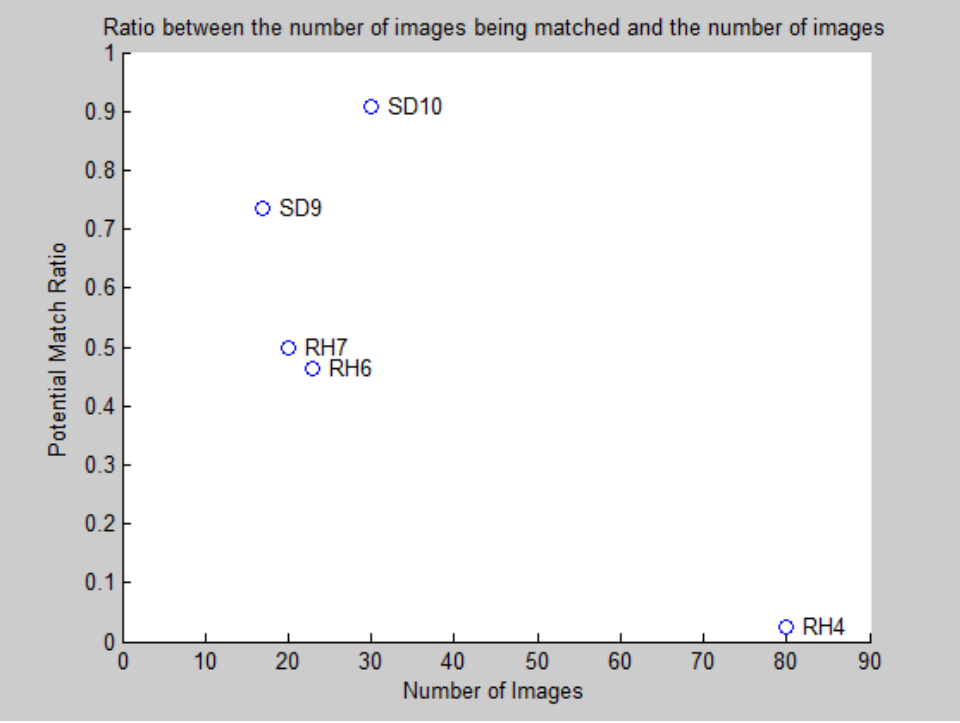


Figure 14: Match Ratio Plot

We also try to put each piece mentioned in the method together and create an automatic system that can grow the point cloud and at the same time provide users with feedback at each incremental step. First, the users will upload a set of photos that will become a seed to the server. Once the seed point cloud is generated, the users will be prompt with alignment tool. Typically, when creating a seed, the users will also provide latitude and longitude of the scene of interest. Therefore, the alignment tool will project point cloud onto 2D google map. The users will be able to rotate, translate and scale the point cloud to fit the shape of the building in the map. Figure 16 shows an interface for point cloud alignment with respect to the google map. Once the transformation is saved, the user will be able to view the point cloud in both 2D and 3D side by side. Because our image-based reconstruction module has server-client support, the users can choose to upload the photo for which it will be localized and added to the point cloud if applicable. The system will response with feedback after the execution is over. The feedback will be in the form of flags and suggested camera locations. Figure 15 show the visualization part of the system. The yellow dot indicates the flag while the blue Bezier curves are the suggested location.

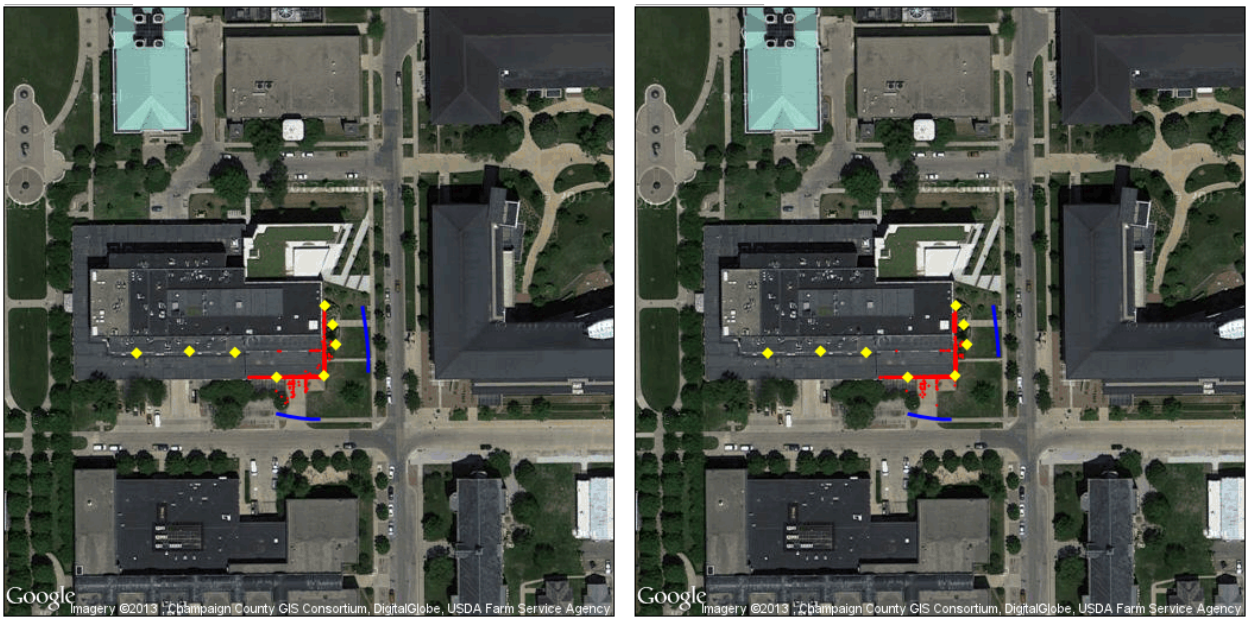


Figure 15: (Left) Simple visualization interface for full system (Right) After adding new image, the right blue Bezier curve is shrink.

This system can be extended to portable device where the users can take a picture on the actual site and receive feedback on the go. This convenient tool will be useful for completing the point cloud without going back and forth between the scene and the workstation to run the SfM software, which could take several hours for hundreds of images. This feedback system will guide the users to take a photo that will improves the completeness of the model.

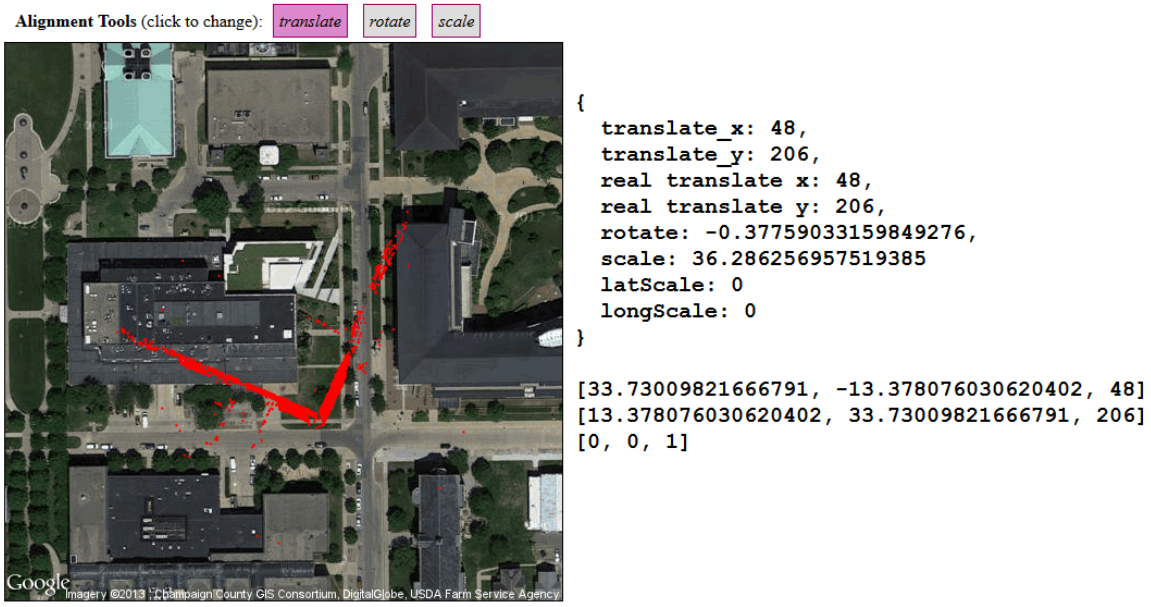


Figure 16: Interface for point cloud alignment

4.3 Discussion on Results

Based on experimental results shown in the previous section, we demonstrate the robustness of our image-based reconstruction method for generating 3D point clouds from data sets of construction sites. Because of the building of the construction site is not fully built, most areas of interest are either hallow or open. The surrounding objects and buildings are visible and our method reconstructs those extra areas. Examples are shown in Figure 8c and 8e where the building behind the construction site is reconstructed. This result comes in two-fold. The positive fold is that the method is able to capture the entire scene without loss of information while the negative side is that we did not want area behind construction site to be part of flagging process.

The choice for selecting images for a seed follows from the part that is well constructed in the full model. By selecting a set of images that capture the main part of the scene, we are able to produce a good seed compared as shown in Figure 8 and 9. For some data set whose all images are not registered, we do the reverse. We start with picking a set of images for seed and see if the point cloud is acceptable. After that, we use the same initial pair for the full reconstruction. This technique improves the camera registration -- from 33 to 56 for RH7 data set.

During the flagging process, the parameters such as d_{min} and r_1 have to be changed for each data. We try the automatic approach where d_{min} is the 3rd percentile of overall density and it works out for all but RH7. This is due to the amount of points in the nearby building area dominating the low percentile of the density. Thus, we need to provide a minimum value -- in this case, we pick

20 -- to ensure that the area we are interested in will be flagged. Similarly for r_1, in all RH dataset, the strategy for finding the diameter of the most condense area along one dimension does not work out well. The building behind the construction site is reconstructed so the height of the building becomes the smallest dimension. In this case, we set up a maximum value for r_1 to be 1 in RH4 and RH6 and 2.5 in RH7.

In Figure 13, you can see that the success rate is below 50% for RH4 and RH6. In Model Incrementation, there are two steps that the process can fail. First, the localization step may not have enough 2D-3D correspondences so the pose estimation step cannot be performed. This may cause by not having enough overlap between image and the point cloud or having too many repeated pattern resulting in several false matches. Second, once new camera is calibrated, the new track is unable to produce a 3D point. This is largely due to a dramatic viewpoint change which results in a low number of matches. The issue can be broken down further to feature detection and the nature of the scene -- such as a pile of sand.

Chapter 5

Conclusion and Future Work

In this thesis, a heuristic algorithm for SfM model incrementation is proposed that can significantly reduce the feature matching time for large 3D image-based point cloud model. Our heuristic approach uses the visibility information of the points that is already available in the existing 3D image-based point cloud model. We demonstrate that the number of images that need to be matched is a lot smaller than the number of images representing the point cloud model and our system completeness one run of Model Incrementation algorithm in about 10 seconds, including feature extraction and matching.

As an extension to SfM model incrementation, we discuss the criteria for completeness of the 3D point cloud model. We show that under simple assumptions we are able to locate the area which is not complete. Furthermore, we propose an algorithm to suggest the next camera location based on the current camera location trajectory. We show that new image whose camera lies on the suggested Bezier curve is able to produce a point cloud that will complete the corresponding area.

The initial experimental results show promise of the proposed method for assessing completeness and accuracy of 3D image-based reconstruction methods. The short runtime of our system enables us to run this system in its current client-server architecture on commodity smart devices and thus provide feedback to users on new locations and orientations for acquiring imagery data.

In our future work, we will investigate how we can incorporate the information from the 3D model generated from CAD drawing tool -- e.g., semantically rich CAD models such as Building Information Models integrated with temporal information (schedule) -- to the definition of completeness. This can relax several assumptions introduced in this work and can enable various platforms to use for model-based performance assessments in engineering applications.

Bibliography

[1] S. Agarwal, N. Snavely, I. Simon, S.M. Seitz, and R. Szeliski. Building rome in a day, 2009.

[2] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint, 2012.

[3] M. Allen, H. Regenbrecht, and M. Abbott. Smart-phone augmented reality for public participation in urban planning, 2010.

[4] H. Bae, M. Golparvar-Fard, and J. White. Enhanced hd^{4}ar (hybrid 4-dimensional augmented reality) for ubiquitous context-aware aec/fm applications, 2012.

[5] H. Bae, M. Golparvar-Fard, and J. White. High-precision vision-based mobile augmented reality system for context-aware architectural, engineering, construction and facility management (aec/fm) applications. *Visualization in Engineering*, 1(3), 2013.

[6] H. Bay, A. Ess, T. Tuytelaars, and L.V. Gool. Speeded-up robust features (surf). *Computer Visioin and Image Understanding*, 110(3):346--359, 2008.

[7] W.C. Chen, Y. Xiong, J. Gao, N. Gelfand, and R. Grzeszczuk. Efficient extraction of robust image features on mobile devices, 2007.

[8] J. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, E. Dunn, Clipp B., S. Lazebnik, and M. Pollefeys. *Computer Vision-ECCV 2010*, pages 368--381. Springer Berlin Heidelberg, 2010.

[9] M. Golparvar-Fard, A. Dimitrov, and F. Pena-Mora. D4ar-4 dimensional augmented reality - tools for automated remote progress tracking and support of decision-enabling tasks in the AEC/FM industry. In *The 6th Int. Conf. on Innovations in AEC Special Session - Transformative machine vision for AEC, State College, PA*, pages 1--10, jun. 2010.

[10] M. Golparvar-Fard, B. Ghadimi, K.S. Saidi, G.S. Cheok, M. Franaszek, and R.R. Lipman. Image-based 3d mapping of rebar location for automated assessment of safe drilling areas prior to placing embedments in concrete bridge decks. In *Construction Research Congress 2012@ sConstruction Challenges in a Flat World*, pages 960--970. ASCE, 2012.

[11] M. Golparvar-Fard, F. Peña-Mora, C.A. Arboleda, and S. Lee. Visualization of construction progress monitoring with 4d simulation model overlaid on time-lapsed photographs. *Journal of Computing in Civil Engineering*, 23(6):391--404, 2009.

[12] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese. Integrated sequential as-built and as-planned representation with tools in support of decision-making tasks in the aec/fm industry. *Journal of Construction Engineering and Management*, 137(12):1099--1116, 2011.

[13] M. Golparvar-Fard, F. Pena-Mora, and S. Savarese. Monitoring changes of 3d building elements from unordered photo collections. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 249 --256, nov. 2011.

[14] R.I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[15] L. Kneip, D. Scaramuzza, and R. Siegwart. A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, pages 2969--2976. IEEE Computer Society, 2011.

[16] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o(n) solution to the pnp problem. *Int. J. Comput. Vision*, 81(2):155--166, 2009.

[17] S. Leutenegger, M. Chli, and R.Y. Siegwart. Brisk: Binary robust invariant scalable keypoints, 2011.

[18] H. Lim, S.N. Sinha, M.F. Cohen, and M. Uyttendaele. Real-time image-based 6-dof localization in large-scale environments, 2012.

[19] D.G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91--110, 2004.

[20] P. Merrell, P. Mordohai, J.-M. Frahm, and M. Pollefeys. Evaluation of large scale scene reconstruction. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1--8, October 2007.

[21] M. Muja and D.G. Lowe. Fast matching of binary features, 2012.

[22] D. Nistér. An efficient solution to the five-point relative pose problem. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):756--777, 2004.

[23] D. Nistér and H. Stewénius. Scalable recognition with a vocabulary tree. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2*, CVPR '06, pages 2161--2168. IEEE Computer Society, 2006.

[24] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[25] S.M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms, June 2006.

[26] N. Snavely, S.M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189--210, 2007.

[27] R. Szeliski. Image alignment and stitching: A tutorial. *Found. Trends. Comput. Graph. Vis.*, 2(1):50--51, 2006.

[28] Matthew M Torok, Mani Golparvar-Fard, and Kevin B Kochersberger. Image-based automated 3d crack detection for post-disaster building assessment. *Journal of Computing in Civil Engineering*, 2013.

[29] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment - a modern synthesis. In *Proceedings of the International Workshop on Vision Algorithms: Theory and Practice*, ICCV '99, pages 298--372, 2000.

[30] K. Tuite, N. Snavely, D. Hsiao, A.M. Smith, and Z. Popović. Reconstructing the world in 3d: bringing games with a purpose outdoors, 2010.

[31] Kathleen Tuite, Noah Snavely, Dun-yu Hsiao, Nadine Tabing, and Zoran Popovic. Photocity: Training experts at large-scale image acquisition through a competitive game. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 1383--1392. ACM, 2011.

[32] C. Woodward, M. Hakkarainen, O. Korkalo, T. Kantonen, M. Aittala, K. Rainio, and K. Kähkönen. Mixed reality for mobile construction site visualization and communication, 2010.

[33] C. Wu. SiftGPU: A GPU implementation of scale invariant feature transform (SIFT). http://cs.unc.edu/∼ccwu/siftgpu, 2007.

[34] C. Wu. Towards linear-time incremental structure from motion, June 2013.

[35] C. Wu, S. Agarwal, B. Curless, and S.M. Seitz. Multicore bundle adjustment, 2011.