# BUILDING SPARSE 3D REPRESENTATIONS FROM A SET OF CALIBRATED PANORAMIC IMAGES

**Daniel Wojtaszek, Robert Laganière, Hamed Peikari and Mohammad Peikari**

VIVA Research Lab, SITE, University of Ottawa
800 King Edward, Ottawa, Ontario
K1N 6N5 Canada
laganier@site.uottawa.ca

**Commission III/4**

**KEY WORDS:** panorama, structure from motion, 3D reconstruction, image-based modeling

**ABSTRACT:**

Virtual navigation in remote environments can be achieved by building an image-based model made of multiple panoramas gathered from cameras moving around the scene. In such models, it could be useful to acquire knowledge of the 3D structure of the scene. In this paper, we propose a method that constructs a sparse but rich 3D representation of a scene, given a set of calibrated panoramic images. The proposed method is a heuristic search algorithm that, given calibrated panoramic images, finds 3D points that correspond to the surfaces of objects in the scene. The algorithm constructs a set of 3D points by searching for matching edge pixels in pairs of images using the epipolar constraint. Empirical results show that the proposed method performs well at locating 3D points of interest in different scenes.

## 1 INTRODUCTION

A goal of tele-presence applications is to allow someone to visually experience a remote environment such that they can freely navigate through the environment with the impression of "being there". One way to reach this goal is to create an image-based models of the scene composed of a multitude of panoramas captured in the scene of interest. Starting from a user-selected georeferenced panorama, virtual navigation is then achieved by allowing the user to move from one panorama to a neighboring one, thus simulating motion along some path in the scene. Under such a framework, knowledge of the 3D structure of the scene is not a necessary requirement; however extracting 3D information from the scene can be beneficial in many ways: i) it allows to more accurately register the panoramic images one with respect to the other and with respect to maps or other representations of the scene; ii) the image-model can then be augmented with virtual objects or virtual annotation that can be coherently displayed on the different panoramic images; iii) 3D measurement in the scene can be made and non feasible motions can be invalidated (e.g. going through an obstacle); iv) it facilitates the generation of photo-realistic virtual views in order to simulate smooth motion while navigating through the scene, from a finite set of images.

The purpose of this work is, given a sparse set of calibrated panoramic images, to obtain a rich set of 3D points that correspond to the surfaces of the objects in a scene. Towards this goal we have developed a search method that searches for matches using features that appear more frequently in each image than the features used during the calibration procedure. Our method uses a multi-start search methodology which is a variation of the method proposed in (Louchet, 1999).

The rest of this paper is organized as follows: Section 2 gives a brief description of methods that have been developed to estimate the 3D structure of a scene; our proposed heuristic search algorithm is presented in Section 3; the results of testing our proposed algorithm on sets of real calibrated images can be found in Section 4; and finally, our conclusions are given in Section 5.

## 2 STRUCTURE FROM MOTION

The purpose of structure from motion algorithms is to estimate the position and orientation of each image in a set of images, and to estimate the 3D structure of the scene.

Recent work has been done by Snavely et al. (Snavely et al., 2008) on calibrating images of a scene taken from different view points, and in turn estimating the 3D structure of the scene. In both cases, camera calibration is carried out by (1) finding correspondences between pixels among subsets of the images using the scale invariant feature transform (SIFT) (Lowe, 2004), (2) estimating the camera parameters (internal and external) using the epipolar constraint and the RANSAC algorithm, and then (3) using bundle adjustment to optimize these parameters, minimizing the reprojection error over all correspondences. The correspondences constitute a sparse description of the 3D structure of the scene. Goesele et al. (Goesele et al., 2007) then proceed to estimate the complete 3D structure of the scene from these sparse 3D points. Both of these methods were tested using large, densely located sets of non-panoramic images.

An alternative to SIFT, called speeded up robust features (SURF), is proposed by Bay et al. (Bay et al., 2008). SURF claims to be faster to compute and more accurate than SIFT.

Although this calibration method is very effective at estimating the camera parameters, it may result in a set of 3D points that are too sparse to adequately describe the 3D structure of the scene. Figure 1 shows an example of how SURF detected correspondences may not adequately cover the scene.

Pollefeys et al. (Pollefeys et al., 2008) and Cornelis et al. (Cornelis et al., 2008) designed systems that perform 3D reconstruction of urban environments from video sequences. Camera pose estimation is carried out using camera calibration techniques that are similar to the technique summarized above. In order to perform faster and more accurately in urban environments, both systems use simplifying geometric assumptions of the scene to model the objects, such as roads and buildings. The system designed by

Figure 1: An image with green dots drawn on it to show the sparse 3D points found during calibration using SURF.

Cornelis et al. also detects the location of any cars in the scene to improve its visual reconstruction. Lhullier et al. (Lhuillier and Quan, 2005) proposed a quasi-dense approach to 3D surface model acquisition from uncalibrated images. Sparse matching is first applied, then in the neighborhood of the matched points, new matches are found based on a combination of local constraints such as correlation, gradient disparity, and confidence.

The method proposed in (Louchet, 1999) searches for 3D points using the images from two calibrated cameras for the purpose of detecting obstacles in front of a moving robot. They use an evolutionary algorithm in which a set of 3D points is evolved to correspond to the surfaces of objects, and to not be too closely concentrated in one area of the scene. These goals are achieved by assigning a fitness value to each point that depends on (1) the image gradient of the point's projection onto one of the images, (2) the similarity measure between the point's projections onto the two images, and (3) the proximity of this point to other points in 3D. A linear weighted average of a small subset of points from the current generation, along with random mutations, are used to evolve the next generation of points.

## 3 A 2D IMAGE AND 1D DEPTH HEURISTIC SEARCH

This heuristic attempts to find points in the 3D world coordinate frame that correspond to the surfaces of stationary objects that are visible in the scene. The input to this algorithm is comprised of a set of calibrated images such that any point in the world reference frame can be projected onto each of these images with reasonable accuracy. Optionally, in addition to the calibrated images, a set of image points that are known to match in two or more of the input images can also be used to initialize the algorithm.

The algorithm first detects a set of candidate pixels in a reference image $I_r$. In this paper, a candidate pixel is any pixel that lies on an edge since edges are useful features for detecting objects in the scene, and edges occur more frequently than SURF and SIFT features. The location $X$ of a candidate point in the three dimensional world coordinate frame is found by searching for the pixel in a neighboring image $I_n$ that most closely matches the pixel in $I_r$. This search is performed along the corresponding epipolar curve in $I_n$. The coordinates of $X$ are computed using the matching image pixels in $I_r$ and $I_n$. To further test that $X$ is indeed correct, $X$ is projected onto each of the images, except $I_r$ and $I_n$, to see if any of these projections match with the projection of $X$ onto $I_r$.

To carry out the detection of edge pixels, a multi-start search method similar to the flies algorithm (Louchet, 1999) is used. The multi-start search methodology uses a set of agents that each behave according to a set of rules. In this algorithm each agent searches $I_r$ for an edge pixel using a combination of local and line searches, and random jumps.

For each combination of pairs of images $(I_r, I_n)$ in the set of input images, the algorithm proceeds as follows:

1. Each agent randomly chooses a pixel in $I_r$ as its starting point.
2. While the stopping condition (Section 3.3.1) is not satisfied, each agent does the following:
   (a) Search for an edge pixel using the line search (Section 3.2.1). If the line search finds an edge pixel that has already been found by any agent, then go to Step 2e.
   (b) Search along the epipolar curve in $I_n$ for the pixel that best matches the corresponding pixel in $I_r$ (Section 3.1).
   (c) If the match search is successful then check the conditions (Section 3.3) to determine if the 3D point corresponding to this match will be added to the set of good points, and add or discard the match accordingly. If the match is discarded then go to Step 2e.
   (d) Perform a local search (Section 3.2.2) to find the next edge pixel. If the local search is successful then go to Step 2b.
   (e) Change this agent's location in $I_r$ by adding a normally distributed random two dimensional vector to its current position. The normal distribution has a mean of 0 and a standard deviation that is set as a multiple of the desired density of good solution points in $I_r$.
   (f) Go to Step 2a.

### 3.1 Match Search

The search for a pixel in an image $I_n$ that matches a given pixel $p_r$ in the reference image $I_r$ is performed by searching every pixel $p_n$ along the corresponding epipolar curve in $I_n$. This search is performed if and only if $p_r$ is an edge pixel. Let $X_r$ be the three dimensional point corresponding to $p_r$ in the coordinate frame of $I_r$. Each pixel on the epipolar curve in $I_n$ is found by quantising the distance $d_{rX}$ between $X_r$ and the focal point $f_r$ of $I_r$ so that two or more values of $X_r$ do not project onto the same pixel in $I_n$. This contrasts with the method proposed by Louchet (Louchet, 1999), which treats $d_{rX}$ as a continuous value. Since there are many values of $d_{rX}$ that correspond to the same pixel in $I_n$, treating $d_{rX}$ as a continuous value may result in wasted computation time.

The similarity measure $M_{rn}^X$ used in this paper is computed using Equation 1. It is the normalized sum of square differences between the intensities $I_r(p)$ of all pixels $p$ within a square neighborhood $N_r$ of $p_r$, and the corresponding intensities $I_n(p_{\mapsto n}))$, where $p_{\mapsto n}$ is the projection of $p$ onto $I_n$.

$$M_{rn}^X = \frac{\sum_{p \in N_r} (I_r(p) - I_n(p_{\mapsto n}))^2}{\sqrt{\sum_{p \in N_r} I_r(p)^2 \times \sum_{p \in N_r} I_r(p_{\mapsto n}))^2}} \qquad (1)$$

The sum of square differences is used assuming that the images were captured under similar lighting conditions. Projecting pixels in $I_n$ onto $I_r$ in this way reduces the effect of scale differences and image warping on the similarity measure. This assumes that every part of the surface in the scene that is projected onto $N_{rn}$ is the same distance from the focal point of $I_r$. Although this assumption is not generally correct, neighborhood projection still works better than not using it. $N_{rn}$ is centered on $p_r$ and has a size of $21 \times 21$.

All of the following conditions must be true for $p_n$ to be considered as a match for $p_r$. Note that $p_r$ is an edge pixel.

1. $p_n$ must be an edge pixel;
2. $M_{rn}^X < 0.05$;

Finally, in order for $X$ to be deemed a good 3D point, $p_r$ must also match its projection onto at least $k$ other images according to the above criteria. Setting $k = 2$ works fairly well at reducing the number of false matches.

### 3.2 Finding edge pixels

When searching for edge pixels, it is desirable to find a sufficient number of them in an acceptable amount of time. Searching through every pixel in an image will guarantee that the maximum number of edge pixels will be found, but it will likely take an unacceptable amount of time. Therefore, to quickly find edge pixels, we use two search methods within a multi-start methodology.

In this paper the Canny algorithm, as implemented in OpenCV, is used to detect edges in an image. This differs from the method proposed in (Louchet, 1999) in which a Sobel operator is used to compute the image gradient which is then used to compute a fitness value. A consequence of this is that their approach allows points that do not project onto edges to be considered for matching. It should be noted that the values of the thresholds used in the canny algorithm should depend on the input images. A simple method to do this is to choose an input image and, using trial and error, experiment with different values until a satisfactory amount of detail in the edges is achieved.

Our multi-start search method begins by placing each agent randomly in the reference image $I_r$, where each agent then proceeds to search for edge pixels using a random line search (Section 3.2.1) and a local search (Section 3.2.2).

**3.2.1 Random Line Edge Pixel Search** If the agent is not on an edge pixel then, starting at this pixel, search $I_r$ along a straight line in a randomly chosen direction until either an edge pixel is found, or a maximum number of pixels have been searched at which point a new direction is randomly chosen and the search is continued. This search method tends to find edge pixels that are adjacent to large areas that are void of edge pixels.

**3.2.2 Local Edge Pixel Search** Since edge pixels tend to be adjacent to other edge pixels, a local search is performed to try and find an edge pixel at which a match search has not yet been performed. This search proceeds by searching the perimeter of a square neighborhood $N_{ej}$ around the current edge pixel until either a new edge pixel is found, or all the pixels on the perimeter have been searched. $N_{ej}$ is centered on the current edge pixel, and the length of each side is 5 pixels. The size of this square is chosen according to how densely distributed one desires the matches to be in the scene.

### 3.3 Assembling Good 3D Points

Once the match search finds a 3D point $X$ that satisfies all of the match criteria (Section 3.1), it will be added to the set of good 3D points $S$ except if at least one of three conditions concerning its relative quality and proximity to 3D points already in $S$ is true. The process of searching for good 3D points using $(I_r, I_n)$ is then stopped when the detail and quality of $S$ stops improving.

The quality measure of 3D points is used to decide which points to keep in $S$ if they all cannot be kept. The quality measure $Q_X$ of $X$ is computed using Equation 2.

$$Q_X = d_{r,n}/M_{rn}^X \tag{2}$$

$M_{rn}^X$ is the similarity measure between the projection $p_r$ of $X$ onto $I_r$ and the projection $p_n$ of $X$ onto $I_n$, and $d_{r,n}$ is the distance between the focal points of $I_r$ and $I_n$. We make $Q_X$ proportional to $d_{r,n}$ because a larger value of $d_{r,n}$ corresponds to a better resolution of the depth of $X$ from the focal point of $I_r$. It should be noted that $S$ can be built upon using, in turn, different combinations of input image pairs for $I_r$ and $I_n$.

There are two conditions concerning the proximity of $X$ to 3D points already in $S$ that, if true, will prevent $X$ from being added to $S$.

The purpose of the first condition is to ensure that the points in $S$ are not too densely crowded in the scene. Therefore, $X$ is not added to $S$ if:

$$(Q_X < Q_Y)\&(Y \in S)\&(Y_{\mapsto r} \in N_{pr}),$$

where $Y$ is a point in $S$ that projects onto $I_r$ within a square neighborhood $N_{pr}$ of $p_r$. $N_{pr}$ is centered on $p_r$ with the length of each side set to 3 pixels. Note that $N_{pr}$ is smaller than $N_{ej}$. Recall from Section 3.2.2 that $N_{ej}$ is the neighborhood used to search for local edge pixels. Setting the size of $N_{pr}$ this way reduces the frequency of $X$ being discarded because of this condition.

The second condition is based on the idea that a group of adjacent pixels in an image usually corresponds to the same surface in the scene, therefore the distance from $f_r$ to $X$ and from $f_r$ to $Y$ should not be too different if $X$ and $Y$ project onto adjacent pixels in $I_r$. Recall that $f_r$ is the focal point of $I_r$. Therefore, $X$ is not added to $S$ if $Q_X < Q_Y$ and $Y \in S$ and $Y_{\mapsto r} \in N_{dx}$ and $|d(X, f_r) - d(Y, f_r)| > t_{dX}$ where $d(A, B)$ is the distance between the points $A$ and $B$, $N_{dX}$ is a square neighborhood of $p_r$, and $t_{dX}$ is a threshold. $N_{dx}$ is centered on $p_r$ and is the same size as $N_{rn}$. Recall that $N_{rn}$ is the neighborhood used to compute the matching similarity measure (Section 3.1). A good value of $t_{dX}$ is half of the minimum $d_{r,n}$ over all possible pairs of input images $I_r$ and $I_n$. Setting the value of $t_{dX}$ this way compensates for the 3D scale given to the scene when calibrating the images.

The third condition for which $X$ may be rejected is used to ensure that, if there is an upper limit to the number of points in $S$, then only the highest quality points are kept. Therefore, $X$ is not added to $S$ if:

$$(|S| = \lceil S \rceil)\&(Q_X < Q_Y \forall Y \in S),$$

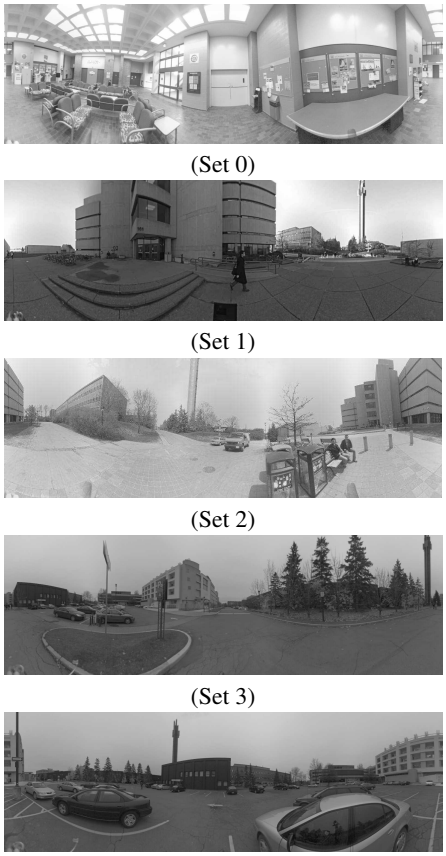
where $|S|$ is the number of points in $S$, and $\lceil S \rceil$ is the upper bound on the number of points in $S$.

When a new point is added to $S$, all points in $S$ that satisfy at least one of the three conditions described above are removed from $S$.

**3.3.1 Stopping Condition** The search for new points to add to $S$ is halted for a given $(I_r, I_n)$ pair when, after a minimum number of iterations that each agent performs while searching for edges, the number of points in $S$ does not increase and the average quality of the points in $S$ does not improve.

## 4 EXPERIMENTS AND RESULTS

We tested our proposed algorithm on a few small sets of calibrated images of various scenes. The experimental setup and results are presented in the following two sections.



(Set 0)

(Set 1)

(Set 2)

(Set 3)

(Set 4)

Figure 2: A sample image from each of the five sets.

### 4.1 Experimental setup

Five sets of five cylindrical panoramic images each were used to test our proposed algorithm. These images are shown in Figure 2. These images were captured using the Ladybug camera and are each $1608x512$ pixels in size. Each set was calibrated using SURF features and descriptors, the RANSAC method, and triangulation.

In our experiments we used 200 agents to search for edges with a minimum 5 of iterations per agent and a maximum number of 3D points allowed in the final solution set to 1000. The algorithm was implemented in C++ using Microsoft Visual Studio 2005, and executed on an Intel Pentium E4600 2.4 GHz processor running the Windows XP operating system.

### 4.2 Results

In this section we will demonstrate how the proposed algorithm performs at finding 3D points corresponding to the surfaces of stationary objects in the scene.

We first used Set 0 to test the proposed algorithm, which needed less than 12 minutes to complete and was able to find more than double the number of 3D points than were found during calibration. Figure 3 shows each image in Set 0 with the projections of the 3D points found using the proposed algorithm drawn on them. These images show how the points are distributed throughout the scene. Set 0 represents a rectangular room in which the measurements of its structure can easily be taken, so we tested the accuracy of these 3D points by estimating the rectangular dimensions of this room using these 3D points and then comparing

Figure 3: Images showing the 3D points found in Set 0 represented as black circles.

Figure 4: An image showing the projections of 3D points corresponding to four walls in the scene.

these estimates to the actual dimensions of the room. We did this by first picking all points that, in their projection onto an image, appear to lie within a predefined region on each of four walls that form a rectangle in the scene.

The regions corresponding to each wall are hand chosen such that: (a) they correspond to a flat wall, (b) they have at least 50 points in them, and (c) the points in each region are distributed so that a good estimate of the best fit plane can be computed. Let $W_f$, $W_b$, $W_l$, and $W_r$ represent the sets of points in the chosen regions corresponding to the front, back, left, and right walls respectively. The projection of these points onto an image is shown in Figure 4. The front wall corresponds to the middle of this image. The best fit plane $R_f$ for the front wall is then estimated by minimizing the sum of squared distances between $R_f$ and each point in $W_f$. The planes $R_b$, $R_l$, and $R_r$ are likewise computed from $W_b$, $W_l$, and $W_r$ respectively.

The angle between the normal vectors of $R_f$ and $R_b$ is $10^\circ$. The angle between the normal vectors of $R_l$ and $R_r$ is $18^\circ$. Since $R_f$ and $R_b$ are not parallel, an estimate of the distance $\tilde{D}_{fb}$ between $R_f$ and $R_b$ is computed as follows.

- let $\bar{G}_{fb}$ = the average distance between $R_b$ and the points in $W_f$.

Table 1: The estimated ($\tilde{D}_{fb}$, $\tilde{D}_{lr}$) and actual ($D_{fb}$, $D_{lr}$) distances between opposing walls in the scene.

| | | | |
|---|---|---|---|
| $\bar{G}_{fb}$ | 4.1615 | $\bar{G}_{lr}$ | 4.0521 |
| $\bar{G}_{bf}$ | 4.4122 | $\bar{G}_{rl}$ | 4.1312 |
| $\tilde{D}_{fb}$ | 4.2869 | $\tilde{D}_{lr}$ | 4.0917 |
| $D_{fb}$ | 12.49 m | $D_{lr}$ | 12.24 m |

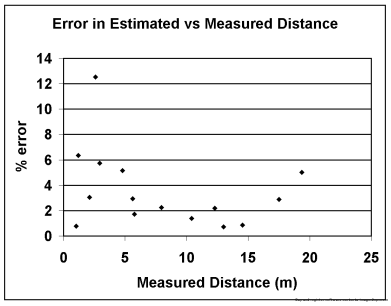



Figure 5: A plot of the error in the estimated distances versus the corresponding measured distances.

- let $\bar{G}_{bf}$ = the average distance between $R_f$ and the points in $W_b$.
- $\tilde{D}_{fb} = (\bar{G}_{fb} + \bar{G}_{bf})/2$

The above procedure is also used to compute $\tilde{D}_{lr}$ using the corresponding planes and sets of points. Table 1 shows that $\bar{G}_{fb}$ and $\bar{G}_{bf}$ are similar in value, as are $\bar{G}_{lr}$ and $\bar{G}_{rl}$. The estimated aspect ratio of the rectangle formed by the four walls, $\tilde{D}_{fb}/\tilde{D}_{lr} = 1.0477$, is very close to the actual aspect ratio, $D_{fb}/D_{lr} = 1.0204$, $D_{fb}$ and $D_{lr}$ being the actual distances (in meters) from the front wall to the back wall and the left wall to the right wall respectively. When computing the scale factor needed to convert the coordinates of the 3D points to meters , using either $\tilde{D}_{fb}/D_{fb} = 0.3432$ or $\tilde{D}_{lr}/D_{lr} = 0.3343$ yields similar results, so the scale factor is approximately $0.34$.

We tested this scale value by choosing 10 detected 3D points in the scene that were not used to estimate the scale, estimating the distance in metres between 15 pairwise combinations of them using the estimated scale factor, and comparing the result with the measured distance. These pairs of points were chosen such that the distances between them can be measured with reasonable accuracy. Figure 5 shows that the estimated distances are mostly within $6\%$ of the measured values, the worst estimate being $12.5\%$ from the measured value.

We then used Sets 1 to 4 to test the proposed algorithm, which needed less than 10 minutes to complete and was able to find more than double the number of 3D points found during calibration for each set. Figures 6 to 10 show the 3D points found in each set of images using the proposed algorithm. Figure 7 shows a 3D rendering of the points found in Set 1, where we can easily see the structure of a building and a tall chimney. These sets of images were captured outside where a number of buildings are visible, so the accuracy of these 3D points is shown by projecting them onto a map of the scene. The 3D points found in a given set are converted to the coordinate system of the map by hand picking three points whose corresponding pixel coordinates can be found on this map, and then using these correspondences to compute the scale, rotation, and translation factors between the world and map coordinate frames. This process was carried out

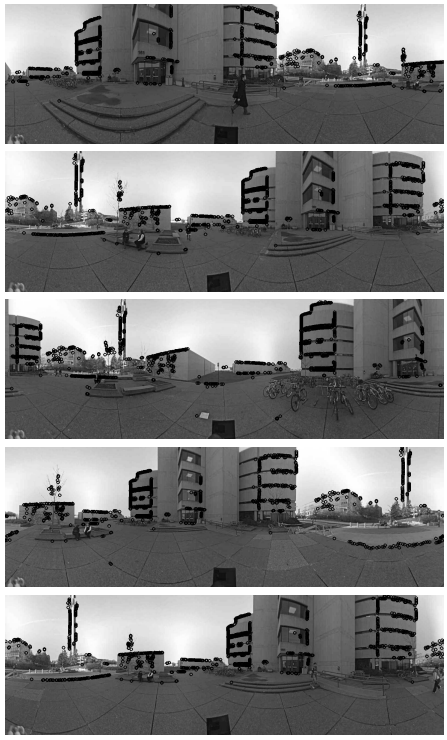


Figure 6: Images showing the 3D points found in Set 1.

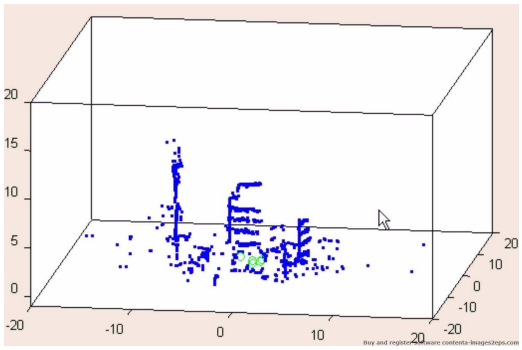


Figure 7: A 3D rendering of the points found in Set 1.

for the 3D points found using each of Sets 1 to 4. All of the converted points, as well as the camera positions, are shown on a map in Figure 11. It should be noted that 3D points that are close to the ground are not shown on the map to more clearly show the points corresponding to objects that are visible in the map. This map shows that each set of 3D points found from a corresponding image set can all be combined to fit fairly well in the same coordinate frame, even though each set of 3D points was found independently of the others.

## 5 CONCLUSION

An heuristic search algorithm is presented that finds 3D points corresponding to the surfaces of stationary objects in the scene represented by a sparse set of panoramic images. The results presented in Section 4 show that the algorithm performs reasonably well at finding 3D points of objects despite the sparsity of each image set and the highly textured areas of the scene. Although the algorithm requires too much processing time for it to be suited for real-time applications, it is not impractical for off line processing, which is acceptable since only stationary objects are of interest.

Figure 8: The 3D points found in one image of Set 2.

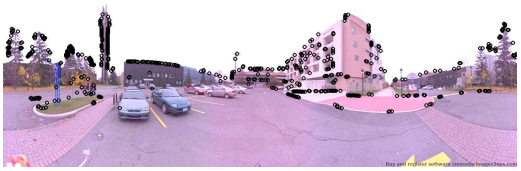


Figure 9: The 3D points found in one image of Set 3.

Future research will focus on speeding up the algorithm, especially since it will be used on larger sets of images. Since a large proportion of the computation time is spent searching the epipolar line for a matching pixel, working towards speeding up this part of the search will likely have the greatest effect on speeding up the algorithm.

## REFERENCES


Bay, H., Ess, A., Tuytelaars, T. and Gool, L. V., 2008. Surf: Speeded up robust features. Computer Vision and Image Understanding 110(3), pp. 346–359.

Cornelis, N., Leibe, B., Cornelis, K. and Gool, L. V., 2008. 3d urban scene modelling integrating recognition. International Journal of Computer Vision 78(2–3), pp. 121–141.

Goesele, M., Snavely, N., Curless, B., Hoppe, H. and Seitz, S. M., 2007. Multi-view stereo for community photo collections. In: Proceedings of the International Conference on Computer Vision, pp. 1–8.

Lhuillier, M. and Quan, L., 2005. A quasi-dense approach to surface reconstruction from uncalibrated images. IEEE Trans. Pattern Anal. Mach. Intell. 27(3), pp. 418–433.

Louchet, J., 1999. From hough to darwin: an individual evolutionary strategy applied to artificial vision. In: Proceedings of Artificial Evolution 99.

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), pp. 91–110.

Pollefeys, M., Nister, D., Frahm, J. M., Akbarzadeh, A., Mordohai, P., Clipp, B., Engels, C., Gallup, D., Kim, S. J., Merrell, P., Salmi, C., Sinha, S., Talton, B., Wang, L., Q.Yang, Stewenius, H., Yang, R., Welch, G. and Towles, H., 2008. Detailed real-time urban 3d reconstruction from video. International Journal of Computer Vision 78(2–3), pp. 143–167.

Snavely, N., Seitz, S. M. and Szeliski, R., 2008. Modeling the world from Internet photo collections. International Journal of Computer Vision 80(2), pp. 189–210.


Figure 10: The 3D points found in one image of Set 4.

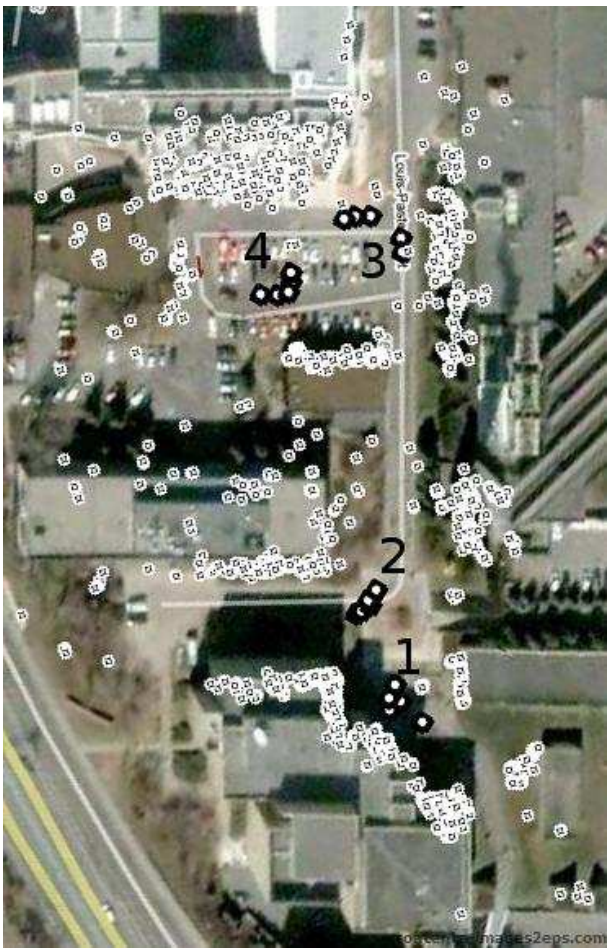



Figure 11: A mapping of the 3D points found in Sets 1 to 4. The positions of the cameras are shown as black squares with white in the middle and the 3D points are shown as white circles with black in the middle.