GPU-FRIENDLY MULTI-VIEW STEREO FOR OUTDOOR PLANAR SCENE RECONSTRUCTION

Hyojin Kim[1], Quinn Hunter[1], Mark Duchaineau[2], Kenneth Joy[1] and Nelson Max[1]
[1]*Department of Computer Science, University of California, Davis, CA, U.S.A.*
[2]*Lawrence Livermore National Laboratory, Livermore, CA, U.S.A.*
{mrhkim, qthunter}@ucdavis.edu, duchaineau1@llnl.gov, {joy, max}@cs.ucdavis.edu

Keywords: Multi-view Stereo, 3D Reconstruction, Planar Reconstruction, GPGPU.

Abstract: This paper presents a new multi-view stereo approach that reconstructs aerial or outdoor scenes in both a planar and a point representation. One of the key features is to integrate two heterogeneous schemes for planar and non-planar reconstruction, given a color segmentation where each segment is classified as either planar or non-planar. In planar reconstruction, an optimal plane for each segment is chosen among possible plane candidates by comparing the remapped reference segment region with multiple target images in parallel on a GPU. In point reconstruction for non-planar objects, remapped pixel descriptors along an epipolar line pair are efficiently matched on a GPU. Our method also detects and discards incorrect segment planes and outliers that have a large 3D discontinuity with the neighboring segment planes. Several aerial and outdoor scene reconstruction results with quantitative analyses are provided.

1 INTRODUCTION

Traditionally, multi-view reconstruction methods begin 3D scene recovery from a set of images with dense point matching, using pixel neighborhood color similarity or feature descriptors. Outliers due to matching error are discarded or adjusted via local smoothing or global energy minimization. A disparity map derived from the set of matched pairs is converted into a 3D depth map via triangulation. Color-segmentation-based algorithms perform a similar procedure, with each segment reconstructed as a plane (or a smooth surface) by least-squares or RANSAC plane fitting, given initially matched points. (Tao and Sawhney, 2000; Hong and Chen, 2004; Taguchi et al., 2008; Bleyer et al., 2010).

In textureless regions or wide-baseline stereo, however, plane fitting may not be successful due to numerous outliers in the initial matching. T-junctions (i.e., incorrectly matched apparent endpoints on partially occluded edges) may cause critical errors in the plane fitting. A more robust planar reconstruction is desirable that can handle such cases. The basic assumption that each segment is planar may also not be applicable to a variety of object surfaces in outdoor scenes. Additionally, over-segmentation, preferred in most algorithms so that fewer segments contain regions in more than one plane, may not be practical for large planar segment regions.

In this paper, we propose a new GPU-based multi-view stereo algorithm for aerial/outdoor urban scenes that integrates planar and point reconstruction methods. Based on the segment properties obtained from color segmentation, each segment is reconstructed either as a plane or as a set of points. This method also detects and discards incorrect segment planes and outliers that have a large 3D discontinuity towards neighboring segment planes or points.

Our proposed algorithm is parallelized onto a GPU where multiple GPU threads are efficiently utilized for higher performance. In the planar reconstruction, multiple plane-finding energy computations are performed in parallel on a GPU to reduce computation time. In the point reconstruction, all pixels in a descriptor form are remapped with respect to camera geometry to improve the efficiency of the data access on a GPU. We provide several aerial/outdoor urban scene reconstruction results, with quantitative analyses to evaluate the GPU speedup of our algorithm.

This paper is organized as follows: Section 2 discusses previous work, together with our contribution. Section 3 gives an overview of the proposed algorithm. Sections 4 and 5 describe the planar and point reconstruction method, respectively. Section 6 provides experimental results with discussion, followed by a conclusion in section 7.

2 RELATED WORK AND CONTRIBUTION

Robust Planar Reconstruction. A key feature in our work is our robust planar reconstruction method using color segmentation. Most segmentation-based algorithms rely on initial matched pairs in each segment, as discussed earlier. However in a scene containing T-junctions or large parallax motions, the initial matched pairs may contain numerous outliers, which can cause incorrect plane fits.

Our approach is more robust and overcomes many potential problems such as T-junctions due to a wide-baseline, since we find an optimal plane by directly remapping each segment, using the homographies from the candidate plane, onto all visible target images. One related approach is Patch-based Multi-View Stereo (PMVS2) (Furukawa and Ponce, 2009) that uses small patches (e.g., 5×5) on which to fit planes. PMVS2 also relies on initial matched pairs (e.g., Harris corner with LoG operators) so the algorithm may lead to incorrect patches. When PMVS2 fails to expand a patch's size, holes or gaps may also occur.

In addition, most segmentation-based stereo algorithms use a disparity map between a stereo-rectified image pair. This is not practical in multi-view cases where we find a plane that satisfies multiple images in which the segment is visible. Moreover, in multi-view cases with widely different camera positions and directions, camera poses cannot be manipulated to prevent large distortions after stereo-rectification.

Another related approach is Iterative Plane Fitting (Habbecke and Kobbelt, 2006) by computing a plane that approximates part of the scene. However, this method requires manual plane initialization. In addition, since it relies on intensity differences between reference and comparison images, it may not be sufficient to find the correct plane in wide-baseline stereo. Our method, on the other hand, improves planar reconstruction by exploiting several matching constraints. The Iterative Plane Fitting method does not detect occlusion, whereas our method simultaneously detects occluded planar segments in the reconstruction process.

Plane Sweeping is another approach to reconstruct planar scenes. Gallup et al. (Gallup et al., 2007) proposed a real-time plane sweeping stereo method for outdoor scenes. However, their method requires initial sparse correspondences and uses intensity differences only, which has limitations in wide-baseline stereo, as mentioned earlier.

In addition, our method use a simple segment graph to filter out plane outliers, that is, any segment that has a large 3D discontinuity between adjacent plane segments is filtered out. Some papers such as (Taguchi et al., 2008) use similar techniques that optimize segment planes. However, their optimization again uses a simple energy function based on a disparity map between a stereo image pair, which is not applicable to multi-view cases. We believe it is more practical and reliable to directly look at planes in 3D instead of a disparity map in an image domain.

Hybrid Reconstruction with Segmentation. We also seek an integration of two heterogeneous methods for aerial and outdoor urban scene reconstruction. Our algorithm uses color segmentation to obtain segmented regions to be categorized by potential 3D geometry. We define a simple planarity criterion by segment color and size so that each segment, treated as either planar or non-planar, is reconstructed as a plane or a set of points, respectively.

GPU-friendliness. For a CPU implementation, our planar reconstruction could potentially be slower than other matching approaches when a segment is too large, since for every pixel in the segment, bilinear interpolation at its reprojected target position is required for every target view. Likewise, the descriptor comparison for point-to-point matches can be very expensive when a scene demands many per-pixel matches. Our algorithm is portable to GPU systems, where it produces matches in a fraction of the time of a CPU version, using the texture mapping hardware in the GPU to do the bilinear interpolation for the remapping.

3 RECONSTRUCTION OVERVIEW

Our reconstruction process begins with a color segmentation, in particular, mean-shift color segmentation (Comaniciu and Meer, 2002). Once segmentation is completed, any tiny segment inside a larger segment that has a similar color is merged with it. Sparse feature-point matching using SURF (Bay et al., 2008), followed by triangulation, is additionally performed to determine an approximate 3D bounding box of the scene. This bounding box is later used to limit the extent of plane candidates in the planar reconstruction and to filter outliers in the point reconstruction.

During the reconstruction process, each segment is classified as either planar or non-planar. However, it is extremely difficult to automatically recog-

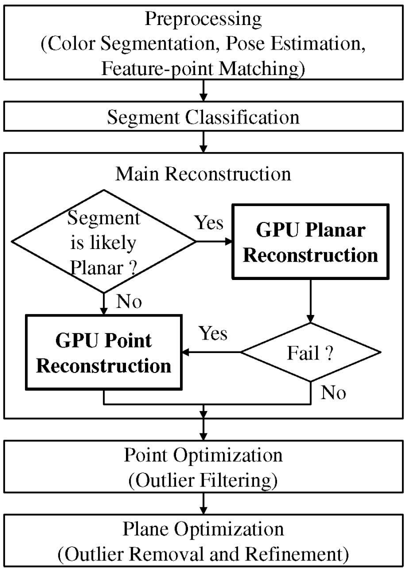

Figure 1: Reconstruction process overview.

nize whether or not a segment is planar without explicit prior knowledge. Instead, we define a simple planarity criterion by segment size and color. For outdoor urban scenes, most building walls and roof, roads, and other artificial structures consist of planes, where our planar reconstruction (Section 4) is used.

In the case of small segments (even ones as parts of larger planes), our point reconstruction method (Section 5) is used. Since small planar objects are more difficult to fit by a plane due to higher ambiguity in matching projections, we treat them as a set of points to be reconstructed. If the segment color is close to green when examining it in both RGB and HSV format, the point reconstruction is also used, as this is likely a tree, bush, or similar non-planar object. Also, the point reconstruction is applied to any segment where the planar reconstruction fails. A semantic classification approach with more intricate object definitions could be used for a more robust classification. Figure 1 gives an overview of our reconstruction process.

4 PLANAR RECONSTRUCTION

An initial plane for a given segment is found among all possible planes by sampling the epipolar 3D rays of three corners of the segment, taking all planes through one sample on each ray, and choosing the best. More optimal planes are then searched in a hierarchical coarse-to-fine manner. This whole search narrows down the set of candidate planes for faster computation. A later filtering step gets rid of any outliers by checking adjacent segment planes for geometric consistency.

4.1 Model

Let I_r and I_i denote, respectively, the reference image and i-th image in the set $\mathbf{I}$ of target images. For each segment $S \in I_r$, we find an optimal plane P that satisfies several constraints described below among I_r and the $I_i \in \mathbf{I}$ in which S is visible. (Visibility is described in the “visibility detection” section below.) In other words, we look for a plane P that has a good intensity/color matching with as many target images as possible. These constraints are defined as an energy function to be minimized.

Color Matching. The first constraint determines if a plane segment is matched to other images in which it is visible by its color (or intensity in gray-scale images). An optimal segment plane P should have a lower color difference with its homography-projected region in a target image. The color energy function, based on the root mean square difference, is defined as

$$E_{color}(P) = \sqrt{\frac{\sum_{I_i \in \mathbf{I}} V(i) C_i(P)}{\sum_{I_i \in \mathbf{I}} V(i) N_i(P)}} \tag{1}$$

where $C_i(P)$ is the sum of squared color differences between reference segment S and target image I_i, and $N_i(P)$ is the number of pixels in the segment visible in target image I_i. $V(i)$ determines whether or not S is visible in target image I_i, and is described later in equation (8). $C_i(P)$ and $N_i(P)$ are defined as

$$C_i(P) = \sum_{(x,y) \in S^*} W_i(H_i(x,y)) M_i(x,y,H_i(x,y)) \tag{2}$$

$$N_i(P) = \sum_{(x,y) \in S^*} W_i(H_i(x,y)) \tag{3}$$

where S^* is the axis-aligned bounding box of the segment S and $H_i(x,y)$ is the image of (x,y) under the homography between I_r and I_i. The boundary clipping function W_i indicates whether or not a pixel is within target image I_i, and M_i measures the squared difference between the reference color of a pixel (x,y) and the bilinear interpolation of the target color at the homography mapped position (x',y') of the pixel in a target image I_i. The definitions of W_i and M_i are

$$W_i(x',y') = \begin{cases} 1 & \text{if } (x',y') \in I_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$M_i(x,y,x',y') = |I_r(x,y) - I_i(x',y'))|^2 \tag{5}$$

Boundary Matching. The second constraint determines if boundary pixels from the reference segment are likely to be remapped into target boundary pixels. We define the energy function as

$$E_{bound}(P) = NN_{Sb} - \sum_{I_i \in \mathbf{I}} \sum_{(x,y)\in S_b} B_i(H_i(x,y)) \qquad (6)$$

where N is the total number of target images, S_b is the set of the boundary pixels in the reference segment, and N_{Sb} is the size of S_b. $B_i(x',y')$ is 1 if (x',y') is on or near boundary pixels (within 3 pixels in our experiments) in a target image I_i, and 0 otherwise.

Visibility Detection. The visibility of S under the homography from a candidate plane P to a target image I_i is computed by checking the plane normal (back-facing or not), the boundary clipping, and the occlusion. We define the energy function as

$$E_{visibility}(P) = N - \sum_{I_i \in \mathbf{I}} V(i) \qquad (7)$$

$$V(i) = \begin{cases} 0 & \text{if } P \text{ is back-facing in } I_i \\ 0 & \text{if } \frac{N_i(P)}{N_{S^*}} < \delta \\ 0 & \text{if } \frac{N_i'(P)}{N_S} < \gamma \\ 1 & \text{otherwise} \end{cases} \qquad (8)$$

where N_{S^*} is the number of pixels within the segment's bounding box. N_i' indicates the number of good matching pixels in segment S and N_S is the number of pixels in S. The first constraint determines if this plane is back-facing or not by checking the normal and the camera direction. The second one indicates how much of the mapped $H_i(S)$ is clipped away by the boundary rectangle of image I_i. In our experiments, δ is approximately .5.

The third constraint measures the occlusion of a segment by counting the number $N_i'(P)$ of good matches. A good matching pixel is a pixel that has a small color difference (e.g., $<$ 20 for a range of 0 to 255) between the reference segment and the target image. If the color difference is greater, we assume that the 3D surface point projecting to the pixel is occluded in image I_i. We want to choose a plane that generates more visible pixels, which prevents us from choosing a slanted plane that may have a low color difference. In our experiments, γ is approximately .7.

4.2 Initial Matching and Refinement

The initial matching process is as follows. First, three corner pixels of each segment are selected, and their camera rays (epipolar lines) in 3D are computed. Each camera ray has a certain range, determined from the precomputed 3D bounding box. Each camera ray range is discretized into K samples and choosing one sample on each ray results in K^3 possible planes. Given each possible plane, we compute its homography and then check the energy functions defined above by reprojecting the segment onto target views, as shown in Figure 2.

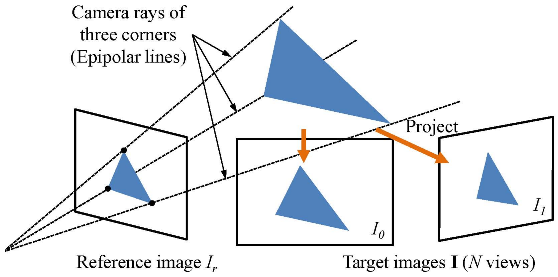
Camera rays of three corners (Epipolar lines)
Project
I_0
I_1
Reference image I_r
Target images **I** (N views)

Figure 2: Initial segment plane matching.

The candidate planes are sorted by $E_{visibility}(P)$ in equation (7). Among the planes that are visible in the most target images, that is, that have the smallest $E_{visibility}(P)$, we choose a plane that has the minimum energy $E_{color}(P)$ in equation (1) and also has at most twice the minimum energy $E_{bound}(P)$ in equation (6). Once an initial plane is chosen, we again search the camera rays in a hierarchical manner, by sampling the rays in smaller intervals near the points defining the optimal plane P, to obtain a better plane. Such a coarse-to-fine matching increases the accuracy with less computation time than a complete high resolution search.

4.3 Filtering

Further steps filter out any outliers from the initial plane matching. First, we filter out any bad plane by examining the score from equations (1) and (7). If the plane's visibility energy $E_{visibility}(P)$ is close to N, which indicates the segment is occluded in most target views, it is removed. If the plane's color energy $E_{color}(P)$ is more than 20, it is also filtered out. We perform another filtering for the remaining planes, using a simple segment graph where each node represents a segment and each edge represents a connectivity between adjacent segments. By examining each segment's neighboring segments, any segment plane that has no adjacent planes in 3D (i.e., a segment has a large 3D discontinuity with all adjacent segments in the reference image) is also discarded. Those discarded segments are again matched as individual points in the following point reconstruction.

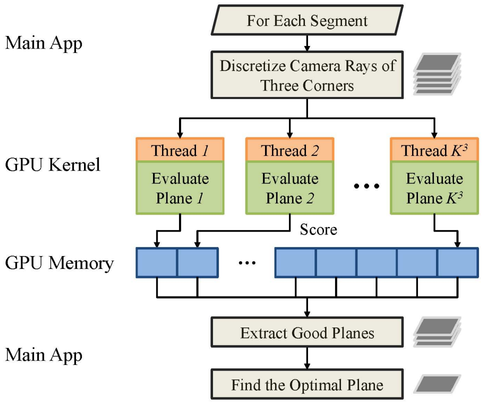

Figure 3: GPU thread utilization for plane matching.

4.4 GPU Implementation

To achieve higher performance, the energy score evaluation for each plane candidate is independently executed in parallel, as illustrated in Figure 3. In general, the number of GPU threads is equivalent to the number of candidate planes. Candidate planes that are invisible in any target view, however, are discarded prior to the GPU computation for better workload balance in all GPU threads.

Initially, all required data such as images, camera poses and segments are copied into the GPU memory so that they can be reused during the entire plane matching process. All input images are loaded into a single 3D texture (but used as an array of 2D textures) to use built-in bilinear texture interpolation in the GPU. Each thread computes energy functions for a certain candidate plane. Then all computed results are simultaneously written to the global memory. For the memory access coalescing, all allocated data (e.g., plane coefficients and energy results) in the global memory are properly aligned.

Once all GPU threads compute the energy of all the candidate planes, the result is copied back to the CPU memory. We then choose several good planes that have the minimum energy $E_{visibility}(P)$. Among the good planes, an optimal plane is chosen by the criterion described in the last paragraph of section 4.2.

5 POINT RECONSTRUCTION

Segments not categorized as planes are reconstructed as a set of 3D points. The goal is to achieve an efficient dense reconstruction at each non-planar segment to better represent the complex geometry of the non-planar surface.

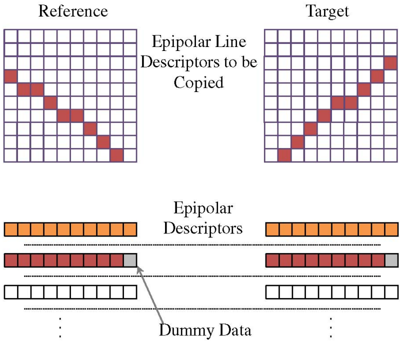

Figure 4: Epipolar descriptor arrangement.

Unless in the form of a stereo-rectified image pair, dense matching tends to require lots of non-localized data access. This is exacerbated when using multidimensional descriptors. Efficient processing on a GPU, with its limited memory size, demands an efficient manner of accessing pixel descriptor data to solve as many pixels as possible using the least amount of data. Our method seeks to leverage epipolar geometry in non-rectified images to increase the efficiency of the point correspondence calculation.

The point reconstruction uses dense DAISY descriptors (Tola et al., 2010) to match as many pixels as possible in non-planar segments. The DAISY descriptors are formed from appending the descriptors at a given pixel across the color channels. That is, each pixel (x,y) stores a set of channel-appended descriptor vectors $D(x,y)$, each of which is a descriptor vector of a given color channel:

$$D(x,y) = [D_r(x,y) : D_g(x,y) : D_b(x,y)] \qquad (9)$$

For a more computationally efficient data structure, we use the epipolar geometry to rearrange the appended descriptors (hereafter referred to as simply descriptors) into a linear array in the GPU memory, which localizes data access to the descriptors.

For a given pixel p_r in a reference image I_r, there is an epiolar line l_r in a target image I_i. Similarly, a pixel p_i along l_r corresponds to an epipolar line l_i in reference image I_r. Epipolar geometry then stipulates that if any pixel p_i on l_r has a match, it must appear in l_i and vice versa. We develop the rearrangement of descriptor data using this property.

5.1 Epipolar Data Localization

For the given reference image I_r, we select a pixel p_r from the set $\mathbf{U}$ of pixels from all segments to be reconstructed as points, and find the epipolar line l_r in target image I_i. We also calculate the epipolar line l_i for image I_r from a point on l_r. We record the epipolar line pair $L = (l_r, l_i)$, and also record the pixels that l_i

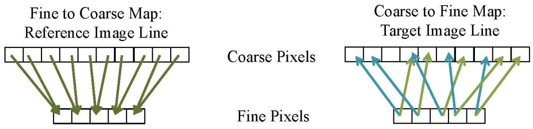

Figure 5: Coarse-to-fine pixel mappings.

covers in **U**. This process is repeated until all pixels in **U** are covered.

The epipolar line pairs are sampled such that the number of samples remains constant. We sample given epipolar line pair L in a fashion such that every integer width and height in its path is covered at least once.

This produces a line with a maximum sample count equal to the larger of the width or the height of the image. For lines with less than this maximum amount of samples, their remaining points are given dummy sample values, as shown in Figure 4.

We rearrange the DAISY descriptor data to fit the epipolar line pairs such that adjacent memory locations contain the descriptors of adjacent pixels on the same epipolar line. We give the dummy pixels added to maintain consistent line sizes descriptors that are distant numerically from what a descriptor could reasonably be.

5.2 Coarse-to-fine Mapping

For a set of sampled epipolar lines in a given image, we also calculate the lines as they appear in a down-sampled version of the original image. Descriptors created from the down-sampled version are then rearranged in the epipolar fashion of the previous section.

During this process, we save a mapping of every fine-level pixel in a reference image epipolar line to its best representative sample in the down-sampled version for the reference image, as shown in Figure 5. For a target image, we create mappings from every coarse-level pixel to a range of pixels on the fine-level epipolar line. This range should contain the points on the fine level that a pixel at the coarse level could map to.

5.3 Point-to-point Matching

Once descriptors are arranged at both coarse and fine levels, we match pixels from a reference line to its target line by minimizing the euclidean distance of the associated descriptor vectors.

The candidate set of matched pixels at the fine level for each reference pixel is modified by the coarse-to-fine mappings. For a given reference pixel and line pair, we first use the fine-to-coarse mappings and match the coarse version of the reference pixel to the line of coarse-level candidate pixels. The coarse-to-fine mapping of this coarse matched pixel gives a subset of fine-level candidate pixels to be matched. Figure 6 illustrates the coarse-to-fine descriptor search.

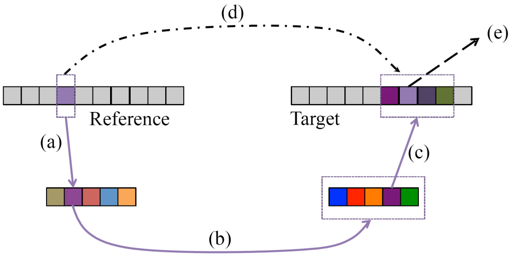

Figure 6: Coarse-to-fine search trajectory. (a) Map to coarse. (b) Match coarse to coarse descriptors. (c) Fine subset from coarse match. (d) Match fine in fine subset. (e) Write out fine match.

After matching the pixels of a reference line l_i, the descriptor matches are translated back to the actual image plane locations of the pixels and triangulated to find the 3D points. In the matching process, we use cross-checking to get rid of outliers (e.g., occlusion), that is, given a matched point in the target image, we apply the matching method in reverse to find its matched point in the reference image. Then the original reference point is compared with the matched point to see if both are the same or similar within a threshold. 3D points outside the bounding box are also filtered out.

5.4 GPU Implementation

For the GPU implementation, we load the descriptor data at the coarse and fine levels and the mappings between them into GPU memory for a set of epipolar line pairs. Because of each epipolar line pair has its own data independent set of Epipolar Descriptor lines (see Figure 4), we arrange GPU thread-blocks around an individual pair of epipolar lines. Each thread within one of these thread blocks attempts to solve at least one pixel or more, depending on the maximum number of threads allowable per thread block.

Threads in a block attempt to operate on adjacent reference descriptors on the line and, after solving for their pixel, refer to another descriptor in memory down the reference epipolar line to solve another pixel if necessary. The dummy pixel descriptors are processed exactly the same as the valid descriptors, as the dummy descriptors are distant enough to never be a closest match. Memory access for each thread becomes more predictable as a result of the data and thread arrangement.

6 EXPERIMENTS AND DISCUSSION

In this section, we evaluate the effectiveness of our algorithm with aerial/outdoor image sets. Our application was written in C/C++ with CUDA, OpenCL and OpenGL. All experiments were performed on an NVIDIA GeForce 9800 GT with 512 MB video memory, or an ATI Radeon HD 4850 with 512 MB video memory (on Intel Core I50 2.66GHz).

Datasets. Figure 7 shows sample input images used for our experiments. We used two sets of aerial photographs of urban scenes; Carlos (Kique) Romero's aerial images of Stockton, California, USA (Romero, 2009), and Brett Wayne's aerial images of Walnut Creek, California, USA (Wayne, 2007). We also used three outdoor image sets that have been acquired in our lab: coffee shack, building 1, and building 2. For camera poses, we used Bundler (Snavely et al., 2008), given known intrinsic camera parameters. To evaluate the accuracy of our planar reconstruction using ground-truth information, we also generated two synthetic scenes based on a simple OpenGL polygonal model with slightly different textures, and then captured images from several camera views.

Reconstruction Results. Figure 8 shows how our two reconstruction methods are integrated. The left image is one of the reference images of the building scene. The middle image shows a mask image for the reference image, generated by using our classification based on the segment size and color. The white regions are reconstructed by our planar reconstruction whereas black regions are reconstructed by our point reconstruction. Thus two results are effectively integrated, as shown in the right image.

Figure 9 shows our reconstruction results for the two aerial scenes and the three outdoor scenes. As shown in the aerial scene reconstruction, our planar reconstruction recovers roads, building roofs, and other large planes whereas our point reconstruction recovers trees, bushes, small building walls, and other small objects. We then render the reconstructed point cloud using Mesh Lab (Cignoni and Ranzuglia, 2011). For the Stockton data, although most roads are extremely large segments where many pixels in the segments are out of image windows in target images, many roads are reconstructed properly. The outdoor scene reconstruction results also look quite dense. In particular, the coffee shack and building 1 show that our reconstruction can work properly for textureless regions. In the coffee shack data, most of the roof and walls are correctly reconstructed although they contain textureless regions.

Table 1: Accuracy of the reconstructed synthetic scenes between our planar reconstruction and PMVS2. Error indicates distance between the reconstructed and the ground-truth surface.

	Error	Our Method	PMVS2
Synthetic Scene 1	Min.	0.000	0.036
	Max.	5.954	27.581
	Mean	1.857	1.965
Synthetic Scene 2	Min.	0.000	0.057
	Max.	29.589	18.900
	Mean	2.446	2.234

Comparison with PMVS2. Figure 10 compares our results and PMVS2 results using several scenes. Because of the larger plane regions (one segment is a large plane region), our result looks more dense and detailed without many holes or gaps, compared to the results of PMVS2, which attempts to solve small piecewise planes with no regard to expected geometry. For instance, our method produces more dense reconstructions in the synthetic scenes which contain T-junctions with large textureless planar regions. In the PMVS2 results, on the other hand, there are incomplete surface patches, probably due to mismatched keypoints or patch extension failure.

Table 1 gives a quantitative evaluation between our planar reconstruction and PMVS2 that measures the accuracy of the reconstructed results of the two synthetic scenes, compared to the ground-truth information. Although our results look more dense than PMVS2 result as shown in Figure 10, both results show similar accuracy.

The separation of reconstruction methods according to expected scene geometry frees us to use the more distinct pixel descriptors to solve non-planar segments such as the bushes and trees in Figure 10, which may not be well modeled as small planes. In general, segment categorization by predicted geometry prevents us from having to optimize a single reconstruction method that may not be appropriate for all scene geometries and textures.

GPU Speedup. We also evaluated the performance gain of our GPU-based implementation. For this analysis, we also implemented a CPU version without parallelization. Figure 11 shows performance differences between the CPU version and our GPU version in the planar reconstruction of the Stockton dataset. The speedup of our GPU-based planar reconstruction is 8.7× on average (Min.: 0.4×, Max.: 39×).

The GPU speedup of our point reconstruction is

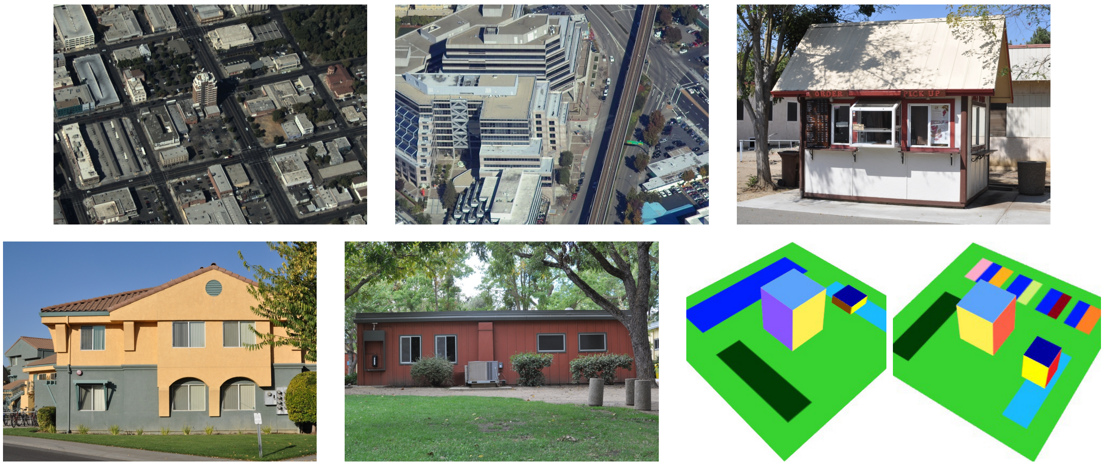

Figure 7: Input images for our experiments. From left to right and top to bottom, *Stockton (CA, USA)*, *Walnut Creek (CA, USA)*, *coffee shack*, *building 1*, *building 2*, *synthetic scene 1* and *synthetic scene 2*.

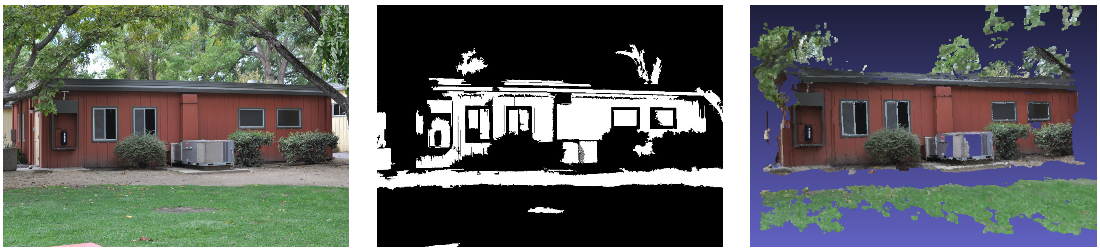

Figure 8: Reconstructed results integrated from our planar- and point-reconstruction methods, rendered by Mesh Lab. One of the reference images *(left)*, the masked image *(middle)*, the reconstructed scene *(right)*.

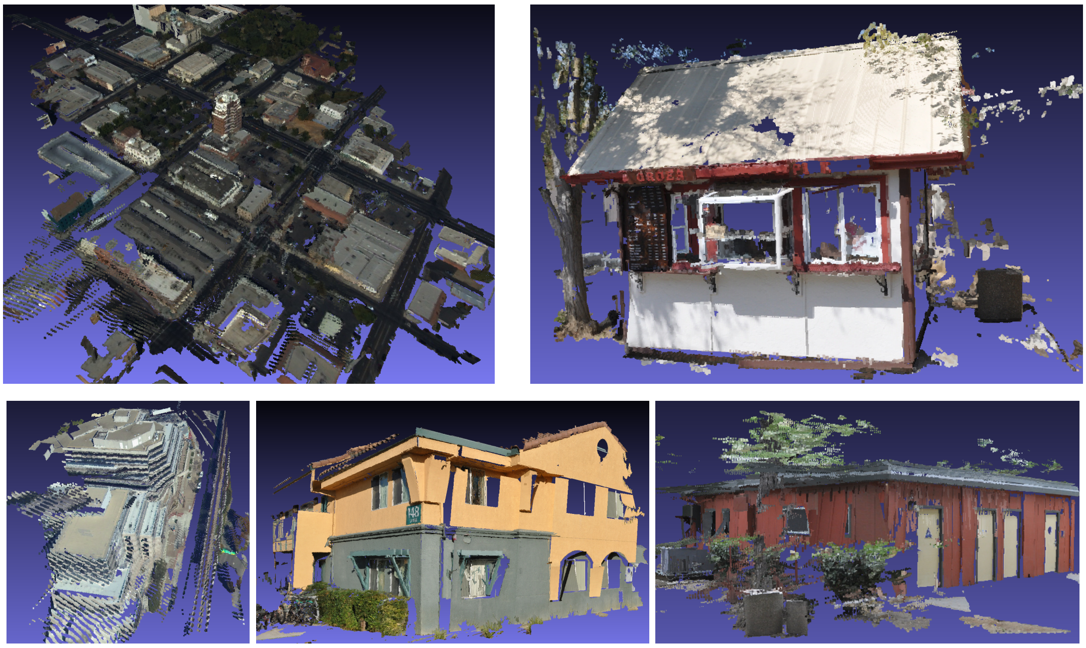

Figure 9: Reconstructed results, rendered by Mesh Lab. From left to right and top to bottom, Stockton (CA, USA) *(left top)*, Coffee shack *(right top)*, Walnut Creek (CA, USA) *(left bottom)*, building 1 *(middle bottom)* and building 2 *(right bottom)*.

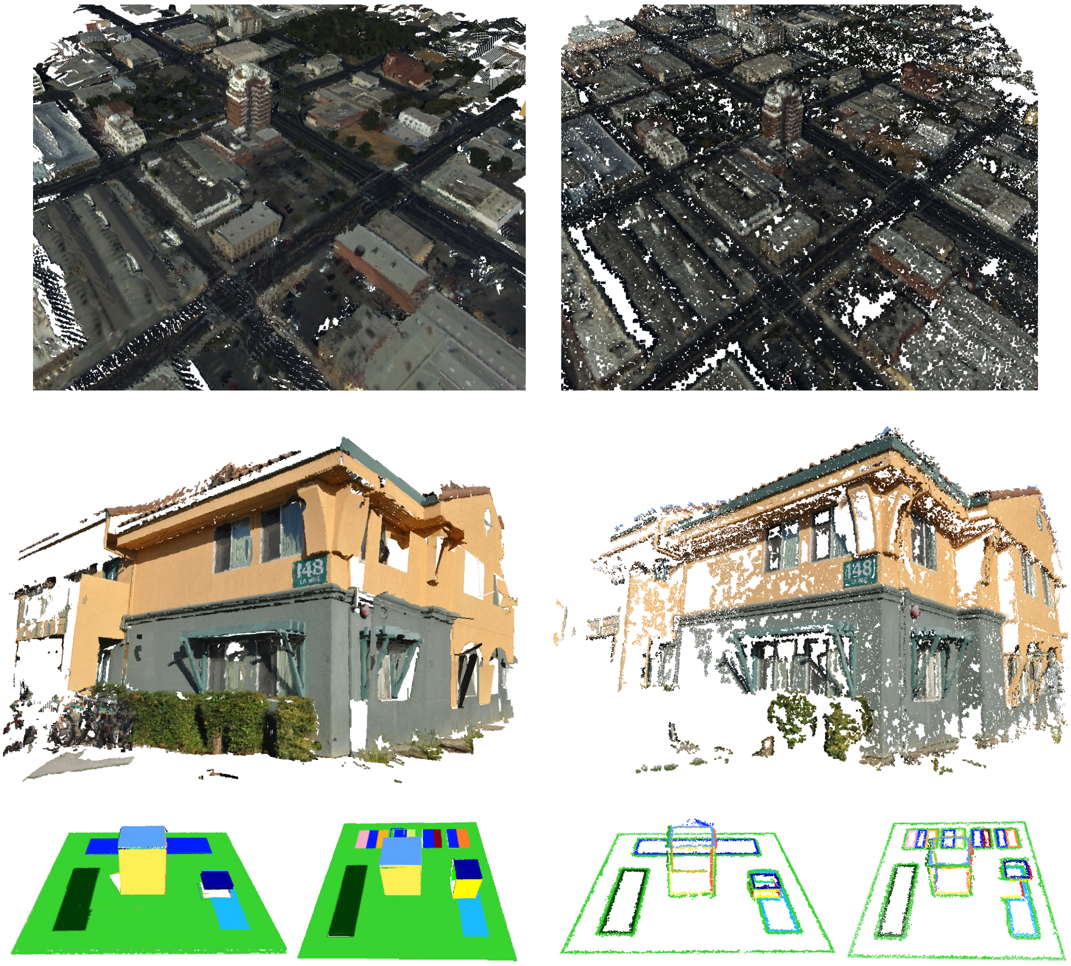

Figure 10: Comparison with PMVS2. Our reconstructed results *(left column)*, and results from PMVS2 *(right column)*.

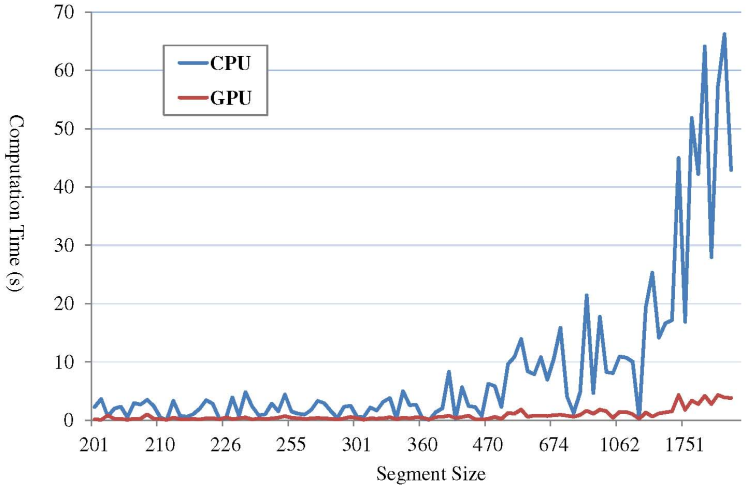

Figure 11: Performance comparison between the CPU- and the GPU-version in our planar reconstruction. In the graph, the x-axis represents segment size (# of pixels) and the y-axis represents computation time (in seconds).

Table 2: Epipolar matching execution time (in seconds) in our point reconstruction.

Data Set	CPU	GPU	Speedup
Stockton	5.279	0.824	6.41×
Walnut Creek	5.276	0.823	6.41×
Coffee shack	6.026	1.612	3.74×

dependent on image resolution and the maximum threads per block, with larger resolutions requiring each thread to solve more points. Table 2 gives average time measurements to solve 5 epipolar line pairs. The Stockton and Walnut Creek scenes required 4 pixels to be solved per thread, while the coffee shack scene required 8 pixels to be solved per thread.

7 CONCLUSIONS

We described a hybrid reconstruction algorithm using GPU parallelism that reconstructs aerial or outdoor urban scenes in a point or in a planar representation. The reconstruction process recovers a scene that contains both planar objects (e.g., building roofs, roads) and non-planar objects (e.g., trees), classified by the segment color and size. Our algorithm is also useful for a scene containing large planar regions. Both reconstructions are efficiently performed in parallel on a GPU.

For future work, one can optimize the planar reconstruction using a Markov random field with a global energy minimization to reduce plane error between adjacent plane segments. For more robust classification between planar and non-planar objects, one can apply a sophisticated semantic scheme by looking up the object identification among a large number of training objects and/or by analyzing the texture pattern.

ACKNOWLEDGEMENTS

This work was performed in part under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

REFERENCES

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

Bleyer, M., Rother, C., and Kohli, P. (2010). Surface stereo with soft segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1570–1577.

Cignoni, P. and Ranzuglia, G. (2011). Mesh lab. Website. available at http://meshlab.sourceforge.net.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:603–619.

Furukawa, Y. and Ponce, J. (2009). Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:1362–1376.

Gallup, D., Frahm, J.-M., Mordohai, P., Yang, Q., and Pollefeys, M. (2007). Real-time plane-sweeping stereo with multiple sweeping directions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Habbecke, M. and Kobbelt, L. (2006). Iterative multi-view plane fitting. In *11th International Fall Workshop, Vision, Modeling, and Visualization*, pages 73–80.

Hong, L. and Chen, G. (2004). Segment-based stereo matching using graph cuts. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 74–81.

Romero, C. K. (2009). Aerial images of Stockton, California. Website. available at http:// www.cognigraph.com/kique_D80-Card1_101NIKON.

Snavely, N., Seitz, S. M., and Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80.

Taguchi, Y., Wilburn, B., and Zitnick, C. L. (2008). Stereo reconstruction with mixed pixels using adaptive oversegmentation. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8.

Tao, H. and Sawhney, H. S. (2000). Global matching criterion and color segmentation based stereo. In *IEEE Workshop on Applications of Computer Vision*, pages 246–253.

Tola, E., Lepetit, V., and Fua, P. (2010). Daisy: An efficient dense descriptor applied to wide-baseline stereo. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, volume 32, pages 815–830.

Wayne, B. (2007). Aerial images of Walnut Creek, California. Website. available at http:// www.cognigraph.com/walnut_creek_Nov_2005.