Manhattan-world Stereo and Surface Reconstruction

Kuan-Ting Yu
CSAIL MIT
Cambridge, USA
peterkty@csail.mit.edu

Abstract

Depth estimation from 2D images has been extensively studied in the computer vision society. Stereo vision is one that calculates depth from disparities in two or more images. Traditional stereo research focused on how to reliably find correspondences in two images. Hence, successful feature point detection and matching are crucial steps. However, for some scenes e.g. *textureless walls, there is no enough texture that can be served as feature points. In this project, we attempt to implement the multiview stereo algorithm with Manhattan-world assumption as structure prior [3]. We believe this is how human reason the depth information when seeing a scene of textureless surfaces. Our contributions are solving important implementation details that was omitted in the original paper, and discuss interesting findings in experiment.*

1. Introduction

Shape from visual sensing is an important topic in computer vision. To build a model, measuring an object directly is one naive way, but sometimes we cannot use this intrusive measurement. The reason may be the object is too large, *e.g*. a building, or with too much fine details, *e.g*. a human face. Light turns out to be a useful measuring tool because it travels with one exact direction, unlike sound that radiates in space. In computer vision, triangulation is a technique to recover depth from disparity in two images. Regions that can be inferred are required to have textures. Though for some textureless area in the triangulation will fail, human use some common sense in structure to recover dense depth information. In this project, we aim to evaluate the stereo algorithm based on Manhattan-world assumption as described in [3]. The assumption assumes the surfaces in a scene are perpendicular to each other. More specifically, the normal of surfaces lie in three orthogonal directions, a.k.a dominant directions. Many artificial scenes like buildings, or interiors of them are following this assumption as shown in Figure 1.

Figure 1. Man-made scenes or objects usually follow the Manhattan-world assumption but with textureless surfaces.

This report describes the system pipeline following the data process order. A system diagram is presented in Figure 2. Besides a concise version of algorithm described in [3], Our contribution to the implementation details are highlighted in the following sections. In Section 2.4, we describe an geometric approach to estimate an important parameter - 3D sample rate. In Section 2.5, we show how to downsample the input properly to run each stage in reasonable amount of time.

we added our low-level description of camera characteristic from geometry Section 2.4, and how to realize each step in reasonable amount of time Section 2.5. Time is crucial because the number of reconstructed points from PMVS [4], the number of sites that we need to infer in MRF (Markov random field), the number of hypothesis plane, and the number of input views all together contribute to a fairly large search space. Many intuitive implementation will take hours to run, which makes the experiment infeasible. In Section 3.3 we present our solution to the violation of submodular constraint while running α-expansion for MRF. Finally, the experimental results are discussed in Section 4.

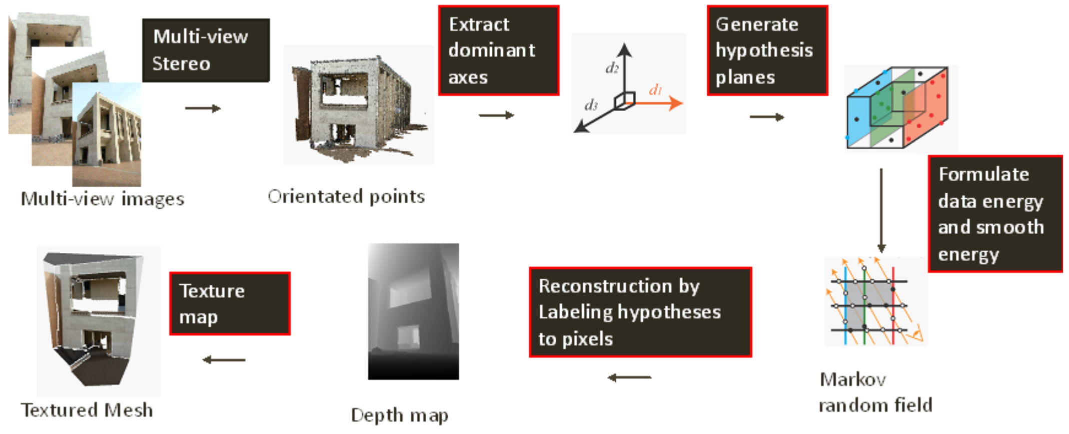

Figure 2. System diagram.

2. Hypothesis Planes

2.1. MVS Preprocessing

Given multiple images from several views of a scene, we aim to reconstruct 3D oriented points (positions with normals). First, we used a publicly available program, Bundler [8], which will output sparse reconstructed points without normal, and camera projection matrices Q_j associated to each view.

$$Q_j = K_j \cdot R_j \cdot T_j \tag{1}$$

where K_j is a 3-by-3 camera intrinsic matrix, R_j is a 3-by-3 rotation matrix, and T_j is a 3-by-4 homogeneous translation matrix.

$$K_j = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2}$$

$$T_j = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \end{pmatrix} \tag{3}$$

The Bundler program needs an initial guess of focal length. If we have the camera, we can use camera calibration tool box [1] to find out precise focal length in pixel. We found that through experiments with accurate initial guess of focal length, the success rate of running Bundler is rather high. Inside Bundler, SIFT key point detection and matching are used for finding correspondences in different views. RANSAC is also used to reject outlying matches. [1] Second, we input the sparse point cloud and camera matrices into PMVS, Patch-Based Multiview Software. The output will be a dense point cloud with normal estimates.

[1]We still rely on keypoint triangulation on contours for model skeleton, but not too much in surface area.

In practice, we used the Catena framework [7] which combined Bundler and PMVS and allows us to easily experiment with the tool chain with different paramenters. As we have tried, it was strenuous to install and run each program manually line due to issues of old dependency needed by the two program also impede the experiment.

2.2. Extract Dominant Axis

To infer the three dominant axes in a Manhattan World, we use the voting scheme described in [3]. We first convert normal direction $N_i = (n_x, n_y, n_z)$ associated with P_i into a hemispherical coordinate (θ, ϕ), where θ denotes longitude, and ϕ denotes latitude, $-\pi \leq \theta \leq \pi, 0 \leq \phi \leq \pi/2$. Because we don't care about the signed direction of the normal when defining a plane, we use a hemispherical coordinate, instead of a whole sphere. We devide ϕ into G discrete ranges, and ϕ into 4G. Thus, there will be $4G \times G$ bins for voting. $G = 33$ in our experiment. First axis $\vec{d_1}$ is extracted by averaging the N_i that are in the bin of highest vote. Second axis $\vec{d_2}$ is found among the bins are $80^\circ \sim 100^\circ$ against $\vec{d_1}$. Same as $\vec{d_1}, \vec{d_2}$ is the average of N_i that is in the highest valid bin. Third axis $\vec{d_3}$ is found in the bins that are $80^\circ \sim 100^\circ$ against $\vec{d_1}$ and $\vec{d_2}$, and with the same averaging process. Figure 3 shows the histograms for extracting dominant axes of hall dataset.

2.3. Generating Hypothesis Planes

For each dominant axis $\vec{d_m}$ and reconstructed point P_i, we can create a plane hypothesis. Having $\vec{d_m}$ as normal, pasing through P_i, the plane will be equation $(\vec{d_m} \cdot X = \vec{d_m} \cdot P_i)$. The offsets $\vec{d_m} \cdot P_i$ is fed into meanshift clustering [2] with bandwidth $\sigma = R$ or $2R$ where R is the 3D sample rate that will be described in Section 2.4. Clusters with less than 50 supporting offsets are discarded. Each cluster center of offsets combines with the normal direction $\vec{d_m}$ to generate hypothesis planes. Figure 4 shows P_i with hypothesis plane

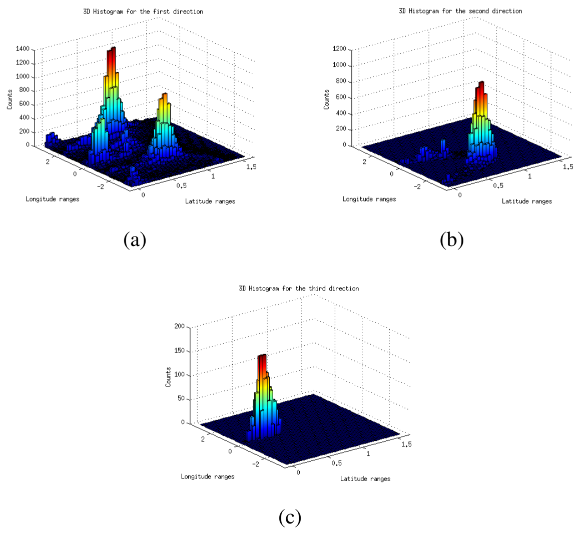

Figure 3. Finding dominant axes of hall dataset with histogram of N_i (a) $\vec{d_1}$ is the average of N_i in maximum voting bin. (b) $\vec{d_2}$ is averaged from maximum bins that are $80^\circ \sim 100^\circ$ against $\vec{d_1}$. (c) $\vec{d_3}$ is averaged from maximum bins that are $80^\circ \sim 100^\circ$ against $\vec{d_1}$ and $\vec{d_2}$.

it belongs to in 3 directions. Different hypothesis plane can be distinguished by color.

2.4. 3D Sample Rate

The 3D sample rate describes how a pixel of the 2D target image corresponds to a sphere diameter R_i centered at P_i. We used camera geometry to calculate R_i as shown is Figure 5.

$$R_i = \frac{\left|(P_i - Q_O^t) \cdot \vec{Q_N^t}\right|}{f^t}, \tag{4}$$

where Q_O^t is the camera optical center, f^t is the focal length of target view, $\vec{Q_N^t}$ is the camera direction. The overall averaged 3D sample rate is

$$R = \sum_{P_i} \frac{R_i}{nP}, \tag{5}$$

where nP is number of P_i.

To calculate focal length f^j given projection matrix Q_j, we tried to formulate a system of equations but without success. Thus, we propose a matrix factorization approach, by factoring Q_j into the 3 seperate matrices as in Eq. 1. Let

$$A = Q_j(1:3, 1:3) = K \cdot R, \tag{6}$$

where K is an upper triangular matrix. R is an orthogonal matrix.

$$AA^T = KRR^TK^T = KK^T, \tag{7}$$

The common Cholesky factorization of a matrix S computes an upper triangular matrix U where $U^TU = S$, but

```
function [K,R,T] = factor_camera_matrix(P)
  A = P(:,1:3);
  T = -A\P(:,4);
  K = rot90(chol(rot90(A*A',2)),2)';
  R = (K\A)';
```

Table 1. Matlab code for camera matrix factorization

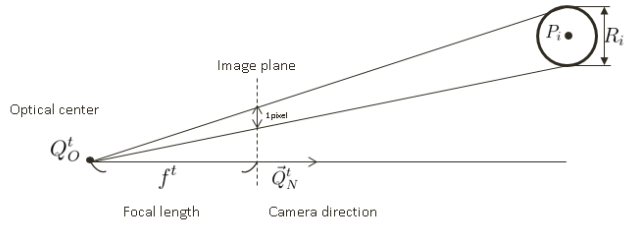

Figure 5. Geometry for computing 3D sample rate R_i at point P_i

we need a U such that $UU^T = S$. We use a rotation trick to keep K upper.

$$K = \text{rot}(\text{chol}(\text{rot}(AA^T)))^T, \tag{8}$$

where rot() rotates matrix 180°, and chol() computes the upper triangular matrix of Cholesky factorization. Here we assume focal length in x and y axes on image are the same, so $f^j = K(1,1)$. A very compact matlab code for camera matrix factorization can be found at Table 1.

2.5. Downsampling Tricks

Because we are going to use MRF to infer the 2D depth map of a 2D image pixel-by-pixel. Resolution of the target image should be about 0.1M pixels for reasonable running time (less than hour per view). However if we downsample the image before the PMVS stage, the PMVS reconstruction would hardly succeed. Our trick is to use high resolution image for PMVS, and in the later construction use a downsampled image with lower dimension, say, 200×300. Besides the image downsampling, the projection matrix Q_j requires modification. The focal length would be therefore scale down by a factor of B. We decompose the camera matrix and modified the focal length in K matrix, and let modified K times with original R and T to form a new projection matrix for downsampled images.

3. Reconstruction

From the input multiview images, select one target view t that we are going to work on. For that $w \times h$ image, [3] formulates it as a MRF labeling problem for each pixel. The labels are one of the hypothesis planes. A good plane hypothesis assignment would have few conflicts with the reconstructed PMVS points P_i and with few discontinuities.

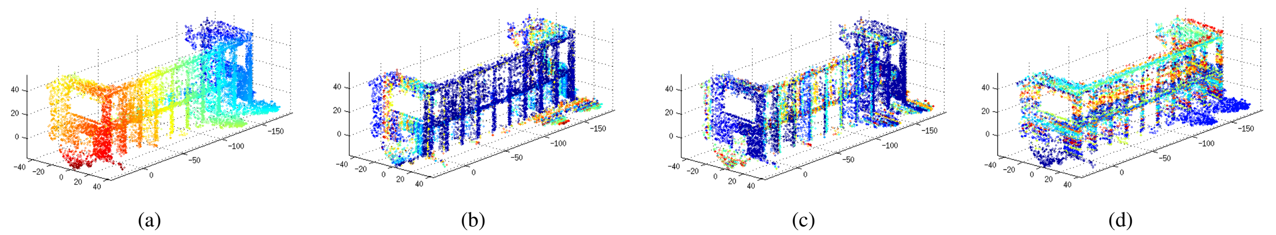

Figure 4. Hypothesis planes of hall dataset in three directions. Points belong to the same cluster, will contribute to the same hypothesis plane are colored the same. (a) PMVS point cloud model with color mapping along depth. (b) Hypothesis planes in $\vec{d_1}$. (c) Hypothesis planes in $\vec{d_2}$. (d) Hypothesis planes in $\vec{d_3}$.

The following energy function formulates this characteristic

$$E = \sum_{p} E_d(p) + \lambda \sum_{\{p,q\}\in N(p)} E_s(k_p, k_q), \tag{9}$$

where k_p is a hypothesis assigned to P, and $N(p)$ is the set of 4-connected neighboring pixels of p. λ is set to 0.1 by default, which is lower than the usual choice in [3], because we found if λ is too high, each pixel will be labeled with the same hypothesis plane, causing every pixel labeled with the same plane.

3.1. Data term

E_d captures the inconsistency between a plane assignment k_p at pixel p and all the reconstructed points P_i by PMVS. If k is assigned to pixel p, we can infer the 3D point formed by the intersection of plane k and the viewing ray that passes through the optical center of target view and pixel p on target image. We denote this 3D point as X_p^k. There are three cases that will cause a conflict between X_p^k and P_i. Let $\pi^j(X)$ denotes projecting a 3D point X on view j, and let t be the target view index. We define a signed distance of two points with respect to view j

$$\Delta_d^j(P,Q) = (Q-P)\cdot \frac{O_j - P}{\|O_j - P\|}, \tag{10}$$

Case 1. If P_i is visible in target image I_t and $\pi^t(P_i) = q$ and q is inside target image. X_q^k must be near to P_i within a distance of γ. Thus, if $\left|\Delta_d^j(P_i, X_q^k)\right| > \gamma$, then P_i conflicts with X_q^k.[2]

Case 2. If P_i is not visible in target view and $\pi^t(P_i) = q$, then X_q^k must be closer to optical center of target image O^t, and therefore blocks the sight to P_i. Thus, if $\Delta_d^j(P_i, X_q^k) < -\gamma$, then P_i conflicts with X_q^k. In [3], this rule was incorrectly written as $\Delta_d^j(P_i, X_q^k) > \gamma$.

Case 3. For any view j, except target view, if P_i is visible on view j, and any X_q^k, $\pi^j(X_q^k) = \pi^j(P_i)$, $\Delta_d^j(P_i, X_q^k) > \hat{\gamma}_{i,j}$, P_i conflicts with X_q^k. $\gamma_{i,j}$ is a modified threshold

$$\hat{\gamma}_{i,j} = \frac{\gamma}{N_{k_p}\cdot r_j(P_i)}, \tag{11}$$

where N_{k_p} is the normal of the plane corresponding to k_p, and $r_j(P_i)$ is the normalized viewing ray directing from I_j to P_i.

$$E_d^i(h_p) = \begin{cases} \max(0, C(P_i) - 0.7) & \text{if } h_p \text{ conflicts with } P_i \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

where $C(P_i)$ is the photometric consistency of patches P_i. If $C(P_i)$ is less than 0.7, the evidence of P_i is not solid enough, and therefore not counted. Finally, the data term for h_p is given as

$$E_d(k_p) = \min(0.5, \sum_i E_d^i(k_p)), \tag{13}$$

3.2. Smoothness term

The smoothness term E_s is a combination of discontinuity E_s^c and prior from dominant edge.

$$E_s^c(k_p, k_q) = \left|X_{p,q}^{k_p} - X_{p,q}^{k_q}\right|, \tag{14}$$

where p and q are neighboring pixels, and $X_{p,q}^k$ is the intersection point of viewing ray passing through the center of pixel p and q, and hypothesis plane k_p, k_q. If $k_p = k_q$ $E_s^c = 0$.

3.2.1 Dominant Edges

Dominant edges are edges that point toward a vanishing point corresponding to a dominant axis. The place of dominant edge has higher possibility that a transition of two planes occurs. To find dominant edges, we first do canny edge detection, and use gradient filters to find out the direction of each edge. Let $\vec{e_p}$ denote the edge direction at pixel p. Let V_m denote the m^{th} vanishing point in target image. We calculate V_m as follows. Let $\vec{d_m} = (n_x, n_y, n_z)^T$ be a normal of dominant axis, and a line passing through

[2]The visibility of P_i on view j. is provided in output *.patch file from PMVS

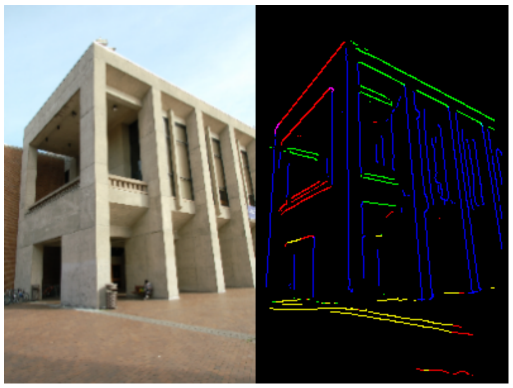
Figure 6. Dominant edge detection on hall dataset. Red, green, and blue denote the dominant direction of edges. If two vanish points and the edge point are almost on the same line in 2D, an ambiguous detection would occur and labeled with mixed color, *e.g.* yellow. However, we only rely on whether a pixel is a dominant edge regardless of the direction it belong to.

$P = (P_x, P_y, P_z)^T$ with direction $\vec{d_m}$ is $\vec{d_m} \cdot t + P$. We project this line on target view and let $t \to \infty$.

$$d \begin{pmatrix} V_x \\ V_y \\ 1 \end{pmatrix} = Q_t \left(\begin{pmatrix} n_x \\ n_y \\ n_z \\ 1 \end{pmatrix} t + \begin{pmatrix} P_x \\ P_y \\ P_z \\ 1 \end{pmatrix} \right) \tag{15}$$

$$\begin{aligned} V_m &= \lim_{t \to \infty} \pi^t(\vec{d_m} t + P) \\ &= \left(\frac{n_x Q_{11} + n_y Q_{12} + n_z Q_{13}}{n_x Q_{31} + n_y Q_{32} + n_z Q_{33}}, \frac{n_x Q_{21} + n_y Q_{22} + n_z Q_{23}}{n_x Q_{31} + n_y Q_{32} + n_z Q_{33}}\right) \end{aligned} \tag{16}$$

For each edge point, place a 7×21 (pixel) window centered at that point with long side following the direction of that edge point. For each cell i in the window, we can interpolate the gradient direction $\vec{e_{p,i}}$, at cell i let $\vec{l_m}$ be the direction from p to V_m, and $\vec{l_m^\perp}$ be perpendicular to $\vec{l_m}$. If

$$\frac{\sum_i \vec{l_m^\perp} \cdot \vec{e_{p,i}}}{\sum_i \vec{l_m} \cdot \vec{e_{p,i}}} > \beta, \tag{17}$$

then we delare p to be a dominant edge. Here, we choose $\beta = 2$. Figure 6 is a dominant edge detection result, the color red, green, and blue denote dominant edges of different dominant axes. If two vanishing points and p lies almost on the same line, an ambiguous result occurs. Finally, we want the smooth penalty is small at dominant edges, so we define E_s as following:

$$E_s(k_p, k_q) = \begin{cases} 0.01 \times E_s^c(k_p, k_q) \text{ if } p \text{ is dominant edge} \\ \qquad\qquad E_s^c(k_p, k_q) \end{cases} \tag{18}$$

3.3. MRF Labeling

We used gco [9] software for labeling. From [6], we found that the smooth function should conform to submodular constraint: suppose α, β, γ are three labeling of two neighboring pixel p, q,

$$E_s^c(\alpha, \gamma) \le E_s^c(\alpha, \beta) + E_s^c(\beta, \gamma), \tag{19}$$

For real number distance, this constraint will be satisfied because $X_{p,q}^k$ lies on the same line and

$$\left\|X_{p,q}^\alpha - X_{p,q}^\gamma\right\| \le \left\|X_{p,q}^\alpha - X_{p,q}^\beta\right\| + \left\|X_{p,q}^\beta - X_{p,q}^\gamma\right\| \tag{20}$$

However, in practice, α-expansion only accepts integer cost function. One solution is to scale up all costs by a factor and truncate the fractions, but this would cause violations of submodularity. We amend the solution by taking ceil() operation after scaling and before truncation. With a simple proof, the ceil() can maintain submodularity in the process of quantization.

4. Experiment

We compare our result with the state-of-the-art surface reconstruction-Poisson surface reconstruction [5]. Figure 7, 8, 9, and 10 show some of the reconstructed hall dataset. The images from left to right are: camera view, dominant edge detection, Poisson surface reconstruction, our Manhattan-world surface reconstruction, our reconstruction with texture, and texture mesh with side view. Our observations are as follows:

1. Sky: This model does not consider depth at infinitely far away. There is no reconstructed PMVS points for sky. Thus, the sky will share the dominant plane with neighboring regions of the building.
2. Ground: In Figure 9, there are very few P_i for ground, so for some view, the ground share the same vertical plane, instead of its horizontal plane.
3. Discontinuity: The plane discontinuity matches the pixel detected as dominant edges (Figure 7). On the other hand, if we do not provide the edges, the reconstructed output would be labeled with the same plane. Hence dominant edge information is crucial.
4. Insufficient evidence: At some view points, there are too few P_i associated with that view, so the data cost is not informative enough to guide the labeling, and therefore the smoothness term dominates. The MRF result would be just one single plane (Figure 10).
5. Although our model is not perfect, we can avoid the "blob" effect in the Poisson reconstruction. The reason is that we have the Manhattan-world assumption where only perpendicular planar surfaces exist.

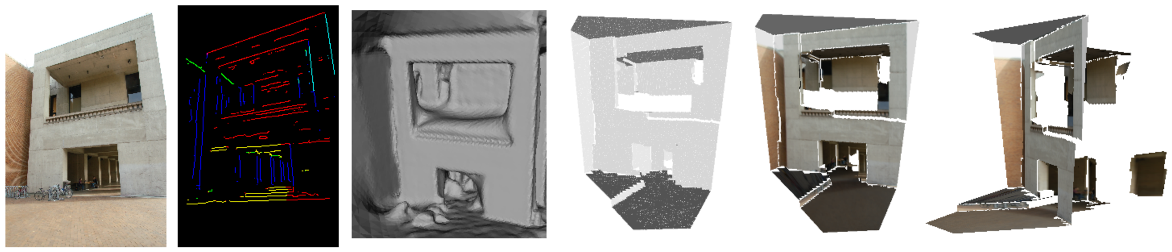

Figure 7. A reconstructed view from hall dataset.

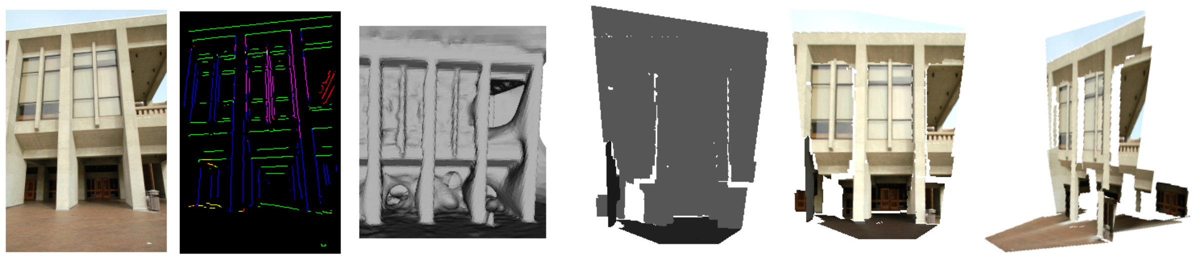

Figure 8. A reconstructed view from hall dataset.

We combined some successfully reconstructed textured mesh from several views, and build a whole model of the hall dataset (Figure 11).

5. Conclusion

In this project, we implemented the Manhattan-world stereo system proposed in [3]. First, we contribute the first publicly available implementation of this system at `http://people.csail.mit.edu/peterkty/code/manhattan/`. Second, we fill in some gaps that are crucial to complete the implementation with the knowledge of Computer Vision learned in class 6.869. Third, we successfully run our system on the hall dataset and generate textured Manhattan-world plane reconstruction from the point cloud.

6. Acknowledgments

We thank Matt Antone for his idea on camera matrix factorization, and Antonio Torralba and Aditya Khosla on various project suggestions.

References

[1] J. Bouguet. Camera calibration toolbox@ONLINE.

[2] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[3] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Manhattan-world stereo. In *IEEE Conference on Computer Vision and Pattern Recognition, 2009. CVPR 2009.*, pages 1422–1429, 2009.

[4] Y. Furukawa and J. Ponce. Pmvs @ONLINE.

[5] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, 2006.

[6] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[7] A. Rossi, H. Rhody, C. Salvaggio, and D. Walvoord. Abstracted workflow framework with a structure from motion application. In *Image Processing Workshop (WNYIPW), 2012 Western New York*, pages 9–12, 2012.

[8] N. Snavely. Bundler: Structure from motion (sfm) for unordered image collections@ONLINE.

[9] O. Veksler and A. Delong. Gco-v3.0 @ONLINE.

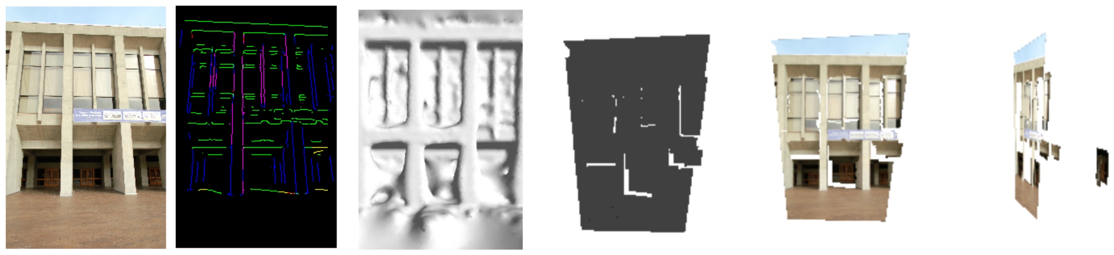
Figure 9. A failing reconstructed view from hall dataset, where ground plane is vertical

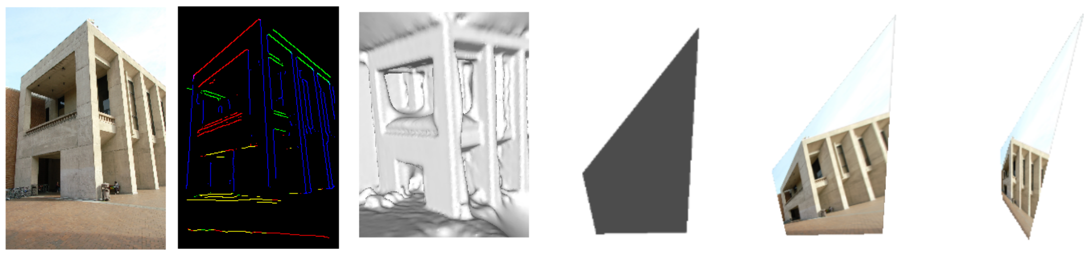
Figure 10. A failing reconstructed view from hall dataset, where all pixels are labeled with the same plane hypothesis

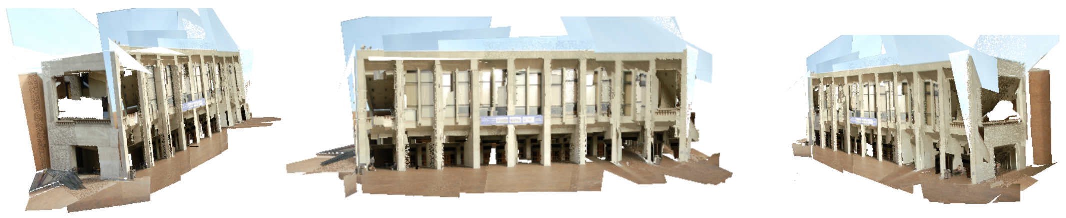
Figure 11. Combining all successful reconstructed meshes to build the hall model.