CGI2013 manuscript No.
(will be inserted by the editor)

Textured Mesh Surface Reconstruction of Large Buildings with Multi-View Stereo

Chen Zhu · Wee Kheng Leow

Abstract Reconstruction of 3D models of buildings has many applications. It can be used for virtual reality, games, digital special effects, heritage preservation, and city planning. There are three main approaches for reconstructing 3D models of buildings, namely laser scanning, structure-from-motion (SfM), and multi-view stereo (MVS). Laser scanning is accurate but expensive and limited by the laser's range. SfM and MVS recover 3D point clouds from multiple views of a building. MVS methods, especially patch-based MVS, can achieve higher density than do SfM methods. Sophisticated algorithms need to be applied to the point clouds to construct mesh surfaces. The recovered point clouds can be sparse in areas that lack features for accurate reconstruction, making recovery of complete surface difficult. Moreover, segmentation of the building's surfaces from surrounding surfaces almost always requires some form of manual inputs, diminishing the ease of practical application of automatic 3D reconstruction algorithms.

This paper presents an alternative approach for reconstructing textured mesh surfaces from point cloud recovered by patch-based MVS method. To a good first approximation, a building's surfaces can be modeled by planes or curve surfaces which are fitted to the point cloud. 3D points are resampled on the fitted surfaces in an orderly pattern, whose colors are obtained from the input images. This approach is simple, inexpensive, and effective for reconstructing textured mesh surfaces of large buildings. Test results show that the reconstructed 3D models are sufficiently accurate and realistic for 3D visualization in various applications.

Department of Computer Science, National University of Singapore
Computing 1, 13 Computing Drive, Singapore 117417
zhuchen, leowwk@comp.nus.edu.sg

Keywords Building reconstruction · multi-view stereo · mesh surface reconstruction

1 Introduction

Reconstruction of 3D models of buildings has many applications. It can be used to create 3D models for virtual reality, games, and digital special effects. It can also be used for preserving heritage architectures and city planning. There are three main approaches for reconstructing 3D models of buildings, namely laser scanning, structure-from-motion (SfM), and multi-view stereo (MVS). Laser scanning [1,4,8,9,14,35] is accurate but expensive and limited by the laser's range. SfM and MVS recover 3D point clouds from multiple views of a building. SfM recovers both 3D data and camera motion simultaneously [2,5–7,17,18,28,29,32,33,36]. The required camera parameters can be calibrated explicitly or computed by self-calibration methods [16]. MVS applies various geometric constraints to determine point correspondence for triangulation of 3D points given known camera parameters [13,20,26]. In particular, patch-based MVS (PMVS) [12,13,20,23] can achieve higher density than do SfM methods.

Laser scanning typically acquires 3D points in an ordered pattern. So, reconstructing mesh surfaces from laser range data is relatively easy. On the other hand, SfM and MVS recover unordered 3D point cloud. Sophisticated algorithms [10,13,20,22] need to be applied to the point cloud to construct mesh surfaces. Moreover, the recovered point cloud can be sparse in areas that lack features for accurate reconstruction, making recovery of complete surface difficult.

Regardless of the technology used, reconstruction of an entire building cannot be achieved in practice by a single laser scan or single point cloud of the building's frontal view alone. It is always necessary to repeat the same process to reconstruct and merge various parts of the building, making laser scanning technology more cumbersome to apply. Moreover, segmentation of the building's surfaces from surrounding surfaces almost always requires some form of manual inputs, diminishing the ease of practical application of automatic 3D reconstruction algorithms.

This paper presents an alternative approach for reconstructing textured mesh surfaces from point cloud recovered by patch-based MVS method. It regards a building's surfaces as effectively flat surfaces which can be modeled by planes or curve surfaces. This is a good first approximation when the scene depth is much smaller than the distance of the building to the camera, which is true of many modern architectural design. In this case, it is not necessary to apply sophisticated algorithms to reconstruct detailed mesh surfaces from point cloud. Instead, simple surfaces are fitted to the point cloud, and 3D points are resampled on the fitted surfaces in an orderly pattern, whose colors are obtained from the input images. In this way, complete textured mesh of building surfaces can be reconstructed. This approach is thus simple, inexpensive, and effective for reconstructing textured mesh surfaces of large buildings.

In the remainder of this paper, we first review existing methods for reconstructing 3D models of buildings (Section 2). Next, we present details of our textured mesh surface reconstruction algorithm (Section 3). Test results (Section 4) show that the reconstructed 3D models are sufficiently accurate and realistic for 3D visualization in various applications.

2 Related Work

Many methods have been proposed over the past decades for the acquisition of 3D models of objects. They can be broadly categorized as *active methods* and *passive methods*. Active methods use controlled light sources to illuminate the scene and acquire 3D data from the illuminated patterns. These methods include time-of-flight, shape-from-shading, structured light, active stereo, and photometric stereo [27]. They are computationally less demanding than do passive methods, and laser scanning and structured light are already adopted in commercial products. They are applicable when the objects of interests can be appropriately illuminated.

Passive methods acquire 3D data only from input images. Methods such as shape-from-texture, shape-from-occlusion, shape-from-defocus, and shape-from-contour acquire 3D data from single view point [27]. Due to the nature of the features used, these methods are restricted to scenes that are rich in the required features, and they tend to be less accurate than active methods. Methods such as passive stereo, multi-view stereo, structure-from-motion, shape-from-silhouettes acquire 3D data using images captured from two or more view points [27].

Among these methods, laser scanning, structure-from-motion (SfM), and multi-view stereo (MVS) have been successfully demonstrated for 3D reconstruction of buildings and street scenes. Laser scanning is very accurate and efficient, but requires an expensive laser scanner and is limited by the laser's range. In reconstructing large historic sites, laser scanning is often used together with other data such as engineering drawing, aerial images, ground-plane images, etc. [1,4,8,9,14,35]

SfM methods recover 3D data as well as camera positions and orientations from a sequence of input images [2,5–7,17,18,28,29,32,33,36]. The required camera parameters can be calibrated explicitly or computed by self-calibration methods [16].

MVS methods apply various geometric constraints to determine point correspondence for triangulation of 3D points given known camera parameters [13,20,26]. They can be more accurate than SfM methods [30]. In particular, patch-based MVS (PMVS) can recover denser 3D point clouds than do SfM methods [12,13, 15,20,23,24]. SfM and MVS may be used together in a pipeline with MVS serving as the refinement stage [6,25]. Sophisticated algorithms are required to reconstruct mesh surfaces from point cloud. For example, [10,13] use Poisson surface reconstruction method [21], and [20,22] employ graph cut algorithm with geometric constraints.

3 Textured Surface Reconstruction Algorithm

Our proposed algorithm continues from where PMVS ends. It consists of the following main stages:

1. Recover a 3D point cloud of a building using PMVS.
2. Reconstruct main surfaces from 3D point cloud by robust surface fitting (Section 3.1) and splitting (Section 3.2).
3. Resample 3D points to refine surfaces (Section 3.3).

First, the Bundler algorithm [31] is applied to the input images to extract matching feature points and camera parameters. Next, the PMVS algorithm [11–13] is applied to recover a 3D point cloud with color information (Fig. 9, 10). Then, robust surface splitting is performed on the 3D point cloud to split it into multiple parts each corresponding to a single surface, and the surfaces are reconstructed by robust surface fitting. Next, 3D points

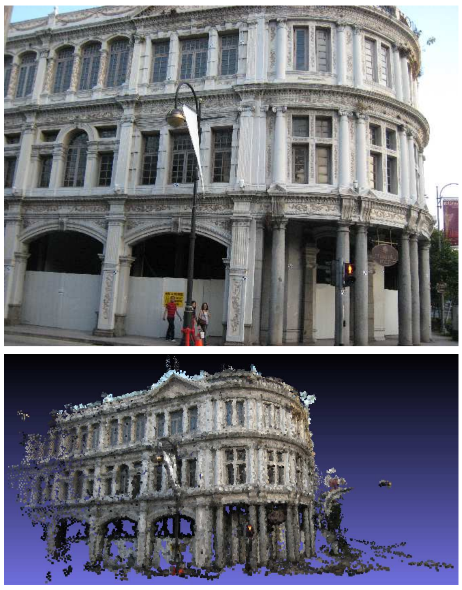


Fig. 1 Recovery of 3D point cloud. (Top) A sample input image of a building. (Bottom) Recovered 3D point cloud.

are resampled over the entire fitted surfaces and their colors are obtained from the input images. Finally, a 3D mesh model that is complete with color texture is constructed from the resampled 3D points to represent the building surfaces.

For a large building, a single point cloud cannot cover the full extent of the building. In this case, multiple point clouds are recovered for different parts of the building, and our algorithm is applied to reconstruct different parts of the building's surfaces, which are aligned and merged together.

3.1 Robust Surface Fitting

First, we consider the case of fitting a single surface to a set of 3D points. Without loss of generality, we assume that the surface's normal is not perpendicular to the Z-axis, a degenerate case of our algorithm. For a degenerate case, we can always rotate the surface to align its normal to the Z-axis, perform the computation, and then rotate the solutions back to the surface's original orientation.

A plane in 3D space can be defined by the equation

$$Z = S(X, Y) = a_1X + a_2Y + a_3. \tag{1}$$

Given a set of n 3D points $\mathbf{X}_i = (X_i, Y_i, Z_i)$, the parameters (a_1, a_2, a_3) of the plane can be recovered as the linear least square solution of the matrix equation:

$$\begin{bmatrix} X_1 & Y_1 & 1 \\ & \vdots & \\ X_n & Y_n & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}. \tag{2}$$

A curved surface can be recovered in a similar way. For example, a quadratic surface of the form

$$Z = a_1X^2 + a_2Y^2 + a_3XY + a_4X + a_5Y + a_6 \tag{3}$$

can be recovered as the linear least square solution of the matrix equation:

$$\begin{bmatrix} X_1^2 & Y_1^2 & X_1Y_1 & X_1 & Y_1 & 1 \\ & & \vdots & & & \\ X_n^2 & Y_n^2 & X_nY_n & X_n & Y_n & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ \vdots \\ a_6 \end{bmatrix} = \begin{bmatrix} Z_1 \\ \vdots \\ Z_n \end{bmatrix}. \tag{4}$$

Other forms of curved surfaces can be recovered in the same manner. Our current implementation restricts to planner and quadratic surfaces which are sufficient for the test cases. The type of surface is specified by the user.

The surface recovered in this way is severely affected by the outliers in the point cloud. It is too tedious to manually identify the inliers and exclude the outliers. So, a robust surface fitting algorithm is adopted to automatically identify the inliers. The robust algorithm adopts a robust error measure $E = \text{median}_i\ r_i$, where r_i is the residual error of a point $\mathbf{X}_i$ with respect to the fitted surface:

$$r_i = (Z_i - S(X_i, Y_i))^2. \tag{5}$$

It iteratively identifies and excludes outliers from the point cloud. The robust surface fitting algorithm can be summarized as follows:

Robust Surface Fitting

```
1. Initialize $P \leftarrow$ input 3D point cloud, $E \leftarrow \infty$.
2. Repeat while $E > \tau_s$:
   (a) Fit a surface $S$ to the points in $P$.
   (b) Compute robust error $E$ of $S$ on $P$.
   (c) Remove from $P$ the points with residual $r_i \geq E$.
```

Empirical tests show that a threshold of $\tau_s = 5 \times 10^{-7}$ yields good results. Figure 2 shows that the algorithm can converge within a small number of iterations and the solution has very small error (Fig. 3). Although a standard robust algorithm such as RANSAC may be used, the above algorithm is much more efficient than RANSAC. It is stable and accurate because of the large number of inliers available in the point cloud for accurate fitting.

3.2 Robust Surface Splitting

The above algorithm fits a single surface to 3D points. Now, we illustrate how to split the input 3D point cloud into multiple parts for fitting different surfaces.

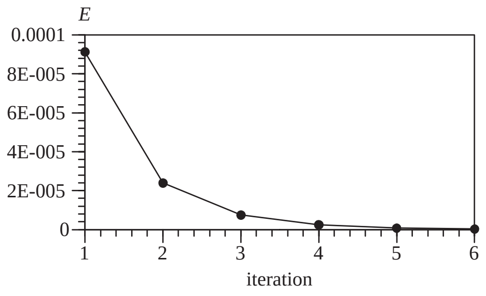


Fig. 2 Convergence curve of robust surface fitting.

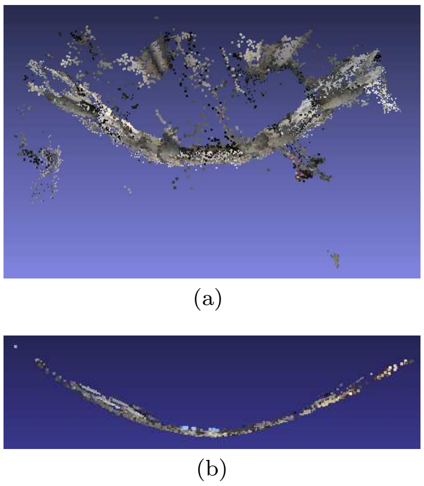


Fig. 3 Robust surface fitting. (a) 3D points around a curved surface. (b) Inliers identified by robust fitting algorithm lie very close to the surface.

Two non-parallel surfaces of the form $Z = S_1(X, Y)$ and $Z = S_2(X, Y)$ intersect along a line or, more generally, a curve. Define $D(X, Y)$ as the absolute difference between the two surfaces:

$$D(X, Y) = |S_1(X, Y) - S_2(X, Y)|. \tag{6}$$

Then, the equation $D(X, Y) = 0$ gives the intersection of the two surfaces, and $D(X, Y)$ measures the distance of any surface point $\mathbf{X} = (X, Y, Z)$ to the intersection.

Let us project the 3D data points along the Z-axis onto the X-Y plane, and let $l(\mathbf{X})$ denote the equation of a line manually drawn on the X-Y plane to divide the data points into two parts (Fig. 4):

$$l(\mathbf{X}) = b_1 X + b_2 Y + b_3 = 0. \tag{7}$$

Then, points $\mathbf{X}_i$ not lying on the line have non-zero values $l(\mathbf{X}_i)$. So, a set of 3D points $\mathbf{X}_i$ can be split into two subsets according to the sign of $l(\mathbf{X}_i)$. After splitting, a surface can be fitted to each of the two subsets using the robust surface fitting algorithm. The intersection of these surfaces induces a change of $l(\mathbf{X})$, and the whole process can be iterated to obtain optimal splitting. So, the robust surface splitting algorithm can be summarized as follows:

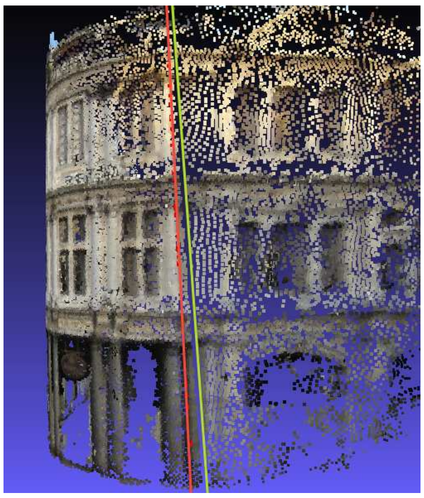


Fig. 4 Splitting of point cloud for reconstructing multiple surfaces. (Green) Initial splitting line. (Red) Refined splitting line.

Robust Surface Splitting

1. Let P be a 3D point cloud to be split and l be the initial splitting line.
2. Repeat:
   (a) Split P into P_1 and P_2 according to the sign of $l(\mathbf{X}_i)$, for all pints $\mathbf{X}_i \in P$.
   (b) Perform robust fitting of surface S_1 to P_1 and S_2 to P_2.
   (c) Compute $D(\mathbf{X}_i)$ of each point $\mathbf{X}_i$ and select a subset Q of points with the smallest $D(\mathbf{X}_i)$, which are near the intersection.
   (d) Fit l to the points in Q using linear least square method.

This algorithm is iterated for a fixed number of iterations to split the input point cloud and reconstruct the surfaces.

In the current implementation, the size of Q is set to 50. Empirical tests show that 2 iterations are sufficient to obtain good splitting. The same algorithm is repeatedly applied to different parts of the point cloud to robustly split and reconstruct multiple surfaces of a building. To define the boundaries of a quadrilateral surface, 4 splitting lines need to be manually drawn. For a surface with more complex boundaries, more splitting lines are needed.

As the split surfaces are fitted to their corresponding 3D points independently, they may not join perfectly at the intersection resulting in the appearance of a gap or seam (Fig. 5(a)). This problem is resolved by aligning the surfaces as follows.

Recall that Q is the set of 3D points near the intersection. Then, for each point $\mathbf{X}_i$ in Q, we can obtain the corresponding points $\mathbf{X}_{1i}$ and $\mathbf{X}_{2i}$ on the surfaces S_1 and S_2 as follows:

$$\begin{aligned} X_{1i} &= X_{2i} = X_i, \\ Y_{1i} &= Y_{2i} = Y_i, \\ Z_{1i} &= S_1(X_i, Y_i), \\ Z_{2i} &= S_2(X_i, Y_i). \end{aligned} \tag{8}$$

Given the corresponding points $\mathbf{X}_{1i}$ and $\mathbf{X}_{2i}$, we can compute the best similarity transformation to align the surfaces using well-known algorithms such as [19,34]. Here, we present the algorithm of [19] for completeness.

Suppose we want to transform $\mathbf{X}_{1i}$ to align with $\mathbf{X}_{2i}$. Then, the similarity transformation between them is given by

$$\mathbf{X}_{2i} = s\,\mathbf{R}\,\mathbf{X}_{1i} + \mathbf{T} \tag{9}$$

where s, $\mathbf{R}$, and $\mathbf{T}$ are the scaling, rotation, and translation. The algorithm for finding the best-fit similarity transformation is as follows:

Similarity Transformation

1. Remove translation by moving the point clouds' centroids to the origin of the coordinate system:

   $$\mathbf{r}_{1i} = \mathbf{X}_{1i} - \overline{\mathbf{X}}_1\,, \quad \mathbf{r}_{2i} = \mathbf{X}_{2i} - \overline{\mathbf{X}}_2 \tag{10}$$

   where

   $$\overline{\mathbf{X}}_1 = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_{1i}\,, \quad \overline{\mathbf{X}}_2 = \frac{1}{n}\sum_{i=1}^{n} \mathbf{X}_{2i}\,. \tag{11}$$

2. Determine scaling factor by comparing variance:

   $$s^2 = \frac{\sum_{i=1}^{n} \|\mathbf{r}_{2i}\|^2}{\sum_{i=1}^{n} \|\mathbf{r}_{1i}\|^2} \tag{12}$$

3. Compute rotation matrix as follows:
   Form matrix $\mathbf{M}$ and $\mathbf{Q}$ such that

   $$\mathbf{M} = \sum_{i=1}^{n} \mathbf{r}_{2i}\,\mathbf{r}_{1i}^{\top}, \quad \mathbf{Q} = \mathbf{M}^{\top}\mathbf{M}. \tag{13}$$

   Perform eigendecomposition of $\mathbf{Q}$:

   $$\mathbf{Q} = \mathbf{V}\boldsymbol{\Lambda}\mathbf{V}^{\top} \tag{14}$$

   where $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$. Next, compute

   $$\mathbf{Q}^{-1/2} = \mathbf{V}\mathrm{diag}\left(\frac{1}{\sqrt{\lambda_1}}, \frac{1}{\sqrt{\lambda_2}}, \frac{1}{\sqrt{\lambda_3}}\right)\mathbf{V}^{\top}. \tag{15}$$

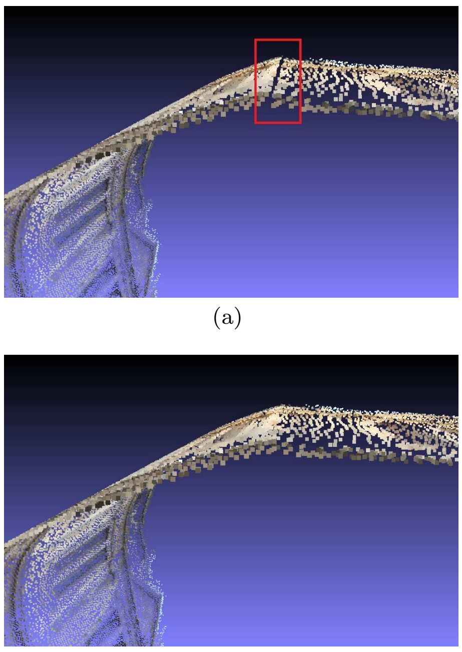
   Then, the rotation matrix is given by

   $$\mathbf{R} = \mathbf{M}\,\mathbf{Q}^{-1/2}. \tag{16}$$

4. Compute translation:

   $$\mathbf{T} = \overline{\mathbf{X}}_2 - s\,\mathbf{R}\,\overline{\mathbf{X}}_1. \tag{17}$$

After computing the optimal scaling, rotation, and translation, the points $\mathbf{X}_{1i}$ can be transformed to align with $\mathbf{X}_{2i}$ using Eq. 9 (Figure 5).


(a)

(b)

Fig. 5 Alignment of surfaces. (a) Before alignment, there is a gap (in the red box) between the two surfaces. (b) After alignment, the gap is removed.

3.3 Resampling of Surface Points

The 3D point cloud recovered by PMVS typically does not cover a surface completely. To obtain complete color texture for a surface, it is necessary to resample the color information from the input images.

First, the set P of 3D points of a surface are projected onto the X-Y plane. Next, corner points of the desired resampling region are manually marked on the X-Y plane. A line segment $l(\mathbf{X})$ is computed between two connected corners such that $l(\mathbf{X})$ is positive for points inside the sampling region and negative outside (Fig. 6). This property holds for all convex regions and allows for easy decision of whether a point is in the resampling region. Finally, 3D points are sampled at regular spacing within the resampling region using the equation of the surface (Eq. 1, 3; Fig. 7). The resampling rate is determined by the user according to the resolution required and it is independent of the sampling density of the point cloud.

The colors of the resampled surface points are obtained from the images. Among the multiple views of a building, the image that corresponds to the frontal view of a building surface is used. Let $\tilde{\mathbf{X}} = (X, Y, Z, 1)^{\top}$ and $\tilde{\mathbf{x}} = (x, y, 1)^{\top}$ denote the homogeneous coordinates of 3D point $\mathbf{X}$ and image point $\mathbf{x}$ respectively. Then, the

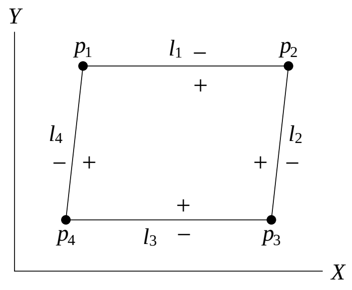


Fig. 6 Resampling region. The lines of the resampling region are computed such that points in the region have positive signs.

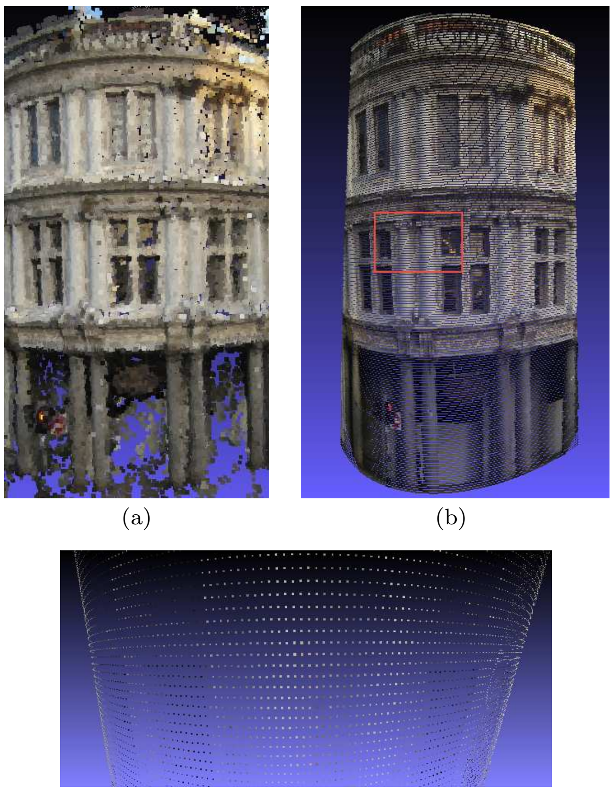


Fig. 7 Resampling of surface points. (a) 3D points in PMVS point cloud is sparse in some regions. (b) Resampled 3D points cover the surface completely. (c) Zoomed-in view of the region in the red box.

3D point projects to the image point according to the perspective projection equation:

$$\rho\tilde{\mathbf{x}} = \mathbf{P}\tilde{\mathbf{X}} \tag{18}$$

where ρ is a scaling parameter and $\mathbf{P}$ is the projection matrix computed by PMVS algorithm. Denoting each row of $\mathbf{P}$ as $\mathbf{P}_k^\top$, Eq. 18 can be written as

$$\rho \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{P}_1^\top \\ \mathbf{P}_2^\top \\ \mathbf{P}_3^\top \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{19}$$

Rearranging Eq. 19 yields the solution

$$x = \frac{\mathbf{P}_1^\top \mathbf{X}}{\mathbf{P}_3^\top \mathbf{X}}, \quad y = \frac{\mathbf{P}_2^\top \mathbf{X}}{\mathbf{P}_3^\top \mathbf{X}}. \tag{20}$$

So, the color of $\mathbf{X}$ can be obtained from the image at pixel location $\mathbf{x} = (x, y)^\top$. Typically, $\mathbf{x}$ has real-valued coordinates. Its color should be derived from those of the four nearest pixels around it using bilinear interpolation.

In some cases, an image may contain building surfaces at oblique view. Retrieval of pixel colors from the oblique view may result in gross distortion (Fig. 8a). To correct for the distortion, the image that presents the surface at frontal view should be used. When multiple images are used for different parts of a surface, seams can appear at the intersections of the images. In this case, color blending should be applied across the seams to remove them (Fig. 8c). Finally, 3D mesh model of the recovered surfaces is constructed by applying ball-pivoting algorithm [3]. Ball-pivoting algorithm is adequate for producing a good mesh because the 3D points are resampled densely at regular spacing.

4 Test Results and Discussions

Two large buildings with curve surfaces were used as the test cases: one with a convex curved surface, the other with a concave curved surface. Multiple images of the buildings were taken from various viewing angles. For each test case, PMVS algorithm was executed on the multiple views to recover 3D point clouds, and our textured mesh surface reconstruction algorithm was executed to reconstruct the surfaces.

Table 1 tabulates the results of applying the algorithms on the test cases. Bundler and PMVS took significant amounts of time to compute the matching feature points, camera parameters, and point clouds. Building 2 had a long extended wall. So, two point clouds were recovered from two sets of images to cover the walls of the building. Figure 9 and 10 show samples inputs, recovered point clouds, and reconstruction results of the buildings. Notice that the point clouds are sparse in some parts of the surfaces. Nevertheless, our algorithm can resample the color textures in those areas from the input images and reconstruct complete mesh of the surfaces.

In the current implementation, our algorithm does not differentiate between the buildings and the occluders, such as the trees, in front of the buildings. So, the occluders are regarded as part of the buildings' textures. To remove the occluders, it is necessary to capture the images with the camera located in between a building and the occluders. With the camera located

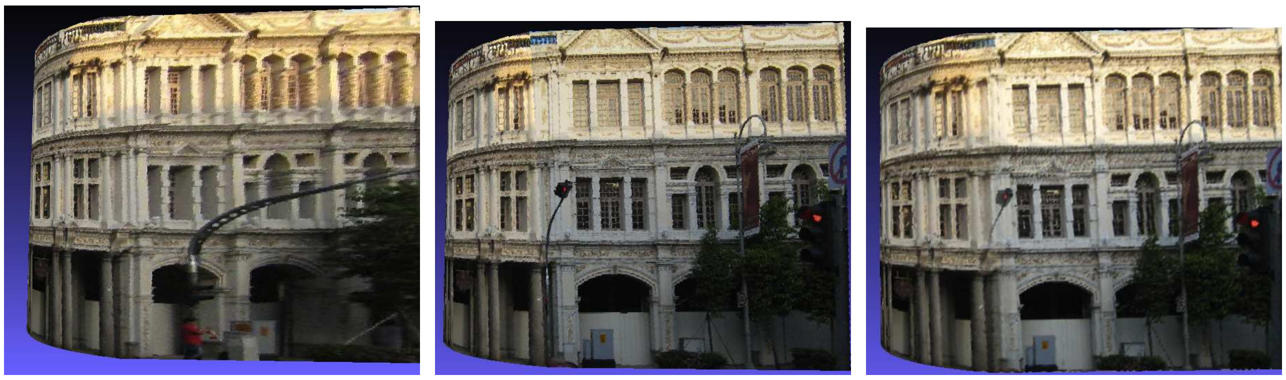


Fig. 8 Removal of distorted color texture. (a) Color texture with gross distortion. (b) Retrieving colors from a frontal view image removes distortion but induces a visible seam. (c) Color blending removes the seam.

Table 1 Results of mesh surface reconstruction. Execution times are measured in minutes, excluding manual inputs.

building	no. of images	no. of point clouds	no. of surfaces	Bundler run time	PMVS run time	our algo run time	total run time
1	27	1	3	20	12	6	38
2	105	2	3	79	47	14	140

at a close distance from the building, it is necessary to construct the building's surfaces in multiple parts and then merge them together. This is technically possible but practically tedious to perform.

5 Conclusions

This paper presented a simple, inexpensive, and effective method for reconstructing textured mesh surfaces of large buildings with curved surfaces. The method applies PMVS to recover point clouds from multiple images of a building. Then, robust surface splitting and fitting algorithms are applied to fit multiple surfaces to different parts of the point clouds. These surfaces are then aligned, merged, and color blended to produce a single textured mesh model of the building. The mesh model is already segmented from the surrounding and can be used directly in various applications. Test results show that the building models reconstructed by our algorithm are sufficiently accurate and realistic for 3D visualization in various applications.

References

1. Allen, P.K., Troccoli, A., Smith, B., Murray, S., Stamos, I., Leordeanu, M.: New methods for digital modeling of historic sites. IEEE Computer Graphics and Applications **23**(6), 32–41 (2003)
2. Beardsley, P., Torr, P., Zisserman, A.: 3D model acquisition from extended image sequences. In: Proc. ECCV, pp. II:683–695 (1996)
3. Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., Taubin, G.: The ball-pivoting algorithm for surface reconstruction. IEEE. Trans. Visualization and Computer Graphics **5**(4), 349–359 (1999)
4. Blais, F.: Review of 20 years of range sensor development. J. Electronic Imaging **13**(1), 232–240 (2004)
5. Cornelis, N., Cornelis, K., Van Gool, L.: Fast compact city modeling for navigation pre-visualization. In: Proc. CVPR (2006)
6. Cornelis, N., Leibe, B., Cornelis, K., van Gool, L.J.: 3D urban scene modeling integrating recognition and reconstruction. Int. J. Computer Vision **78**(2–3), 121–141 (2008)
7. Dellaert, F., Seitz, S.M., Thorpe, C.E., Thrun, S.: Structure from motion without correspondence. In: Proc. IEEE CVPR (2000)
8. El-Hakim, S., Whiting, E., Gonzo, L., Girardi, S.: 3-D reconstruction of complex architectures from multiple data. In: Proc. ISPRS Int. Workshop on 3D Virtual Reconstruction and Visualization of of Complex Architectures (2005)
9. Frueh, C., Zakhor, A.: Constructing 3D city models by merging aerial and ground views. IEEE Computer Graphics and Applications **23**(6), 52–61 (2003)
10. Furukawa, Y., Curless, B., Seitz, S., Szeliski, R.: Towards internet-scale multi-view stereo. In: Proc. CVPR (2010)
11. Furukawa, Y., Ponce, J.: Patch-based multi-view stereo software. http://www.di.ens.fr/pmvs/
12. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. In: Proc. CVPR (2007)
13. Furukawa, Y., Ponce, J.: Accurate, dense, and robust multi-view stereopsis. IEEE Trans. PAMI **32**(8), 1362–1376 (2010)
14. Guidi, G., Micoli, L., Russo, M., Frischer, B., De Simone, M., Spinetti, A., Carosso, L.: 3D digitization of a large model of imperial Rome. In: Proc. Int. Conf. 3D Imaging and Modeling, pp. 565–572 (2005)
15. Habbecke, M., Kobbelt, L.: Iterative multi-view plane fitting. In: Proc. Fall Workshop on Vision, Modeling, and Visualization (2006)

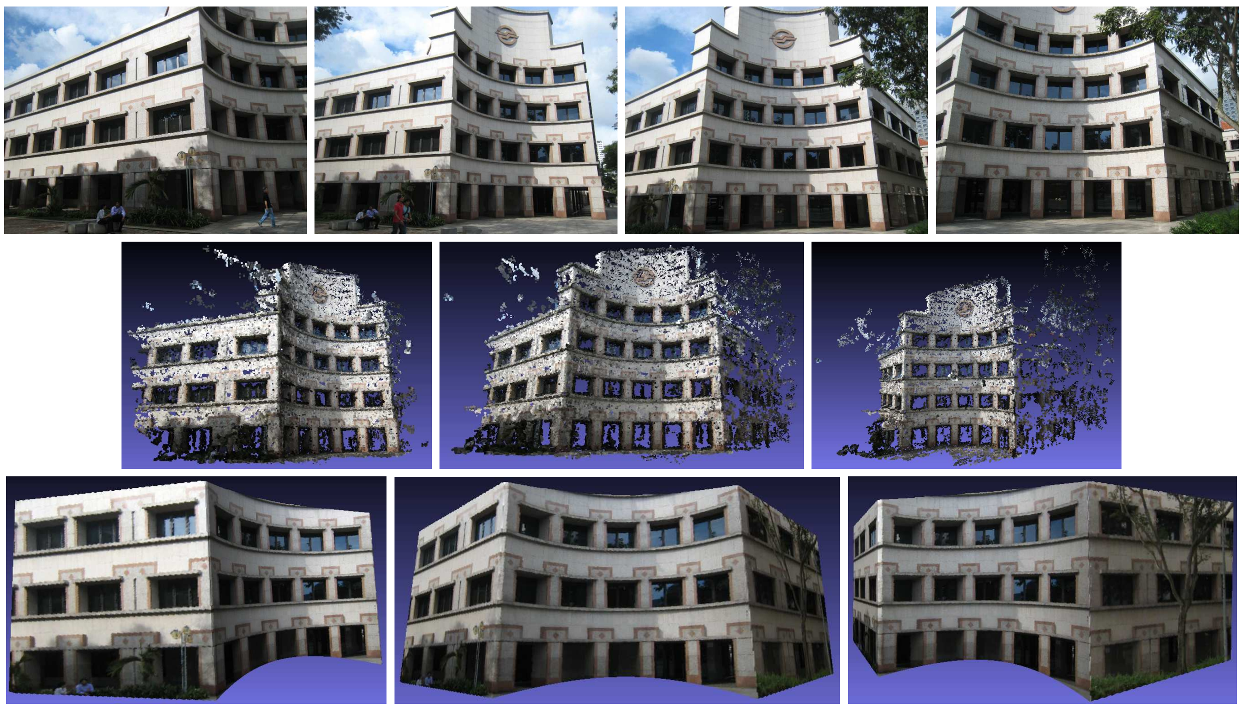


Fig. 9 Reconstructed mesh models of building 1. (Row 1) Sample images from various views. (Row 2) Point cloud recovered by PMVS and (Row 3) reconstructed textured mesh model at various viewing angles.

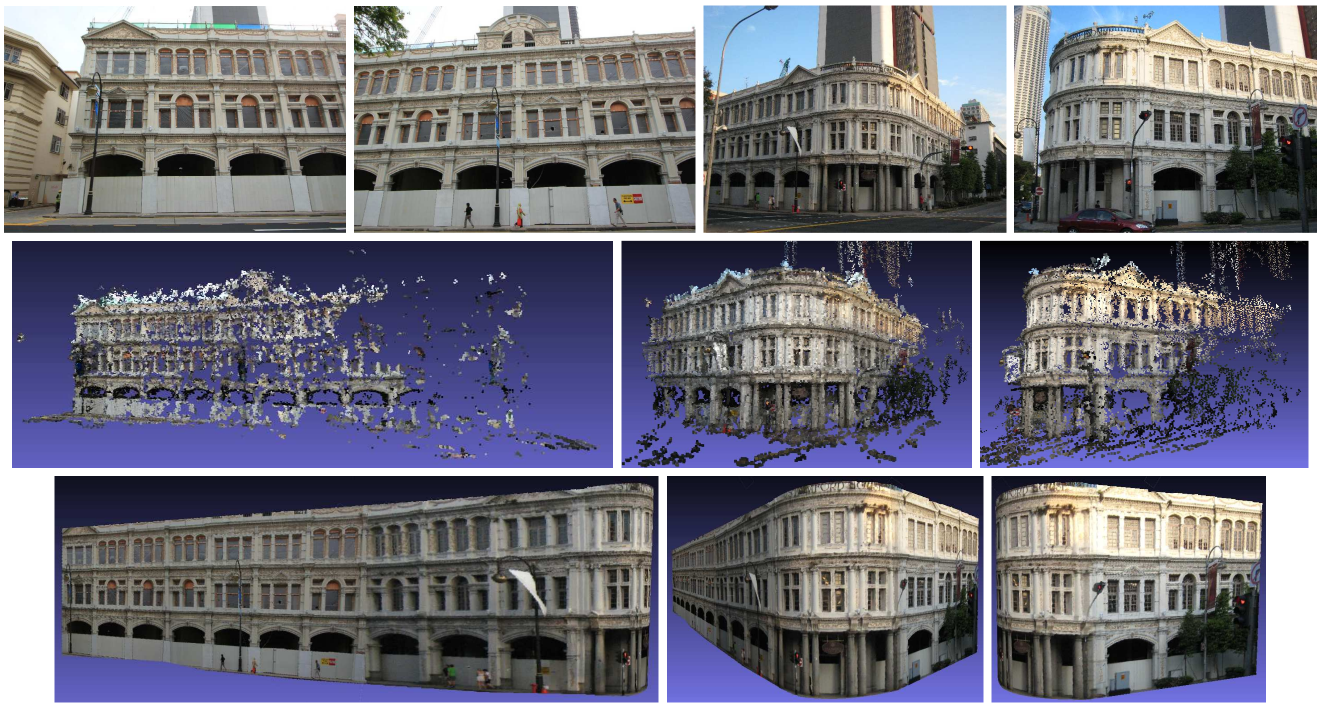


Fig. 10 Reconstructed mesh models of building 2. (Row 1) Sample images from various views. (Row 2) Point clouds recovered by PMVS and (Row 3) reconstructed textured mesh model at various viewing angles.

16. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2000)
17. Havlena, M., Torii, A., Knopp, J., Pajdla, T.: Randomized structure from motion based on atomic 3D models from camera triplets. In: Proc. CVPR (2009)
18. Havlena, M., Torii, A., Pajdla, T.: Efficient structure from motion by graph optimization. In: Proc. ECCV (2010)
19. Horn, B.K.P., Hilden, H.M., Negahdaripour, S.: Closed-form solution of absolute orientation using orthonormal matrices. J. Optical Society of America A **5**(7), 1127–1135 (1988)
20. Jancosek, M., Pajdla, T.: Multi-view reconstruction preserving weakly-supported surfaces. In: Proc. CVPR (2011)
21. Kazhdan, M., Bolitho, M., Hoppe, H.: Poisson surface reconstruction. In: Proc. Int. Symp. Geometry Processing, pp. 61–70 (2006)
22. Labatut, P., Pons, J., Keriven, R.: Robust and efficient surface reconstruction from range data. Computer Graphics Forum (2009)
23. Labatut, P., Pons, J.P., Keriven, R.: Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In: Proc. CVPR (2007)
24. Lhuillier, M., Quan, L.: A quasi-dense approach to surface reconstruction from uncalibrated images. IEEE Trans. PAMI **27**(3), 418–433 (2005)
25. Liang, Y., Sheng, Y., Song, H., Wang, Y.: 3D reconstruction of architecture from image sequences. World Academic Publishing (2013)
26. Mičušík, B., Košecká, J.: Piecewise planar city 3D modeling from street view panoramic sequences. In: Proc. CVPR (2009)
27. Moons, T., Van Gool, L., Vergauwen, M.: 3D reconstruction from multiple images, Part 1: Principles. Foundations and Trends in Computer Graphics and Vision **4**(4), 287–398 (2008)
28. Morris, D., Kanade, T.: A unified factorization algorithm for points, line segments and planes with uncertainty models. In: Proc. ICCV, pp. 696–702 (1998)
29. Poelman, C., Kanade, T.: A paraperspective factorization method for shape and motion recovery. IEEE Trans. PAMI **19**(3), 206–218 (1997)
30. Seitz, S.M., Curless, B., Diebel, J., Scharstein, D., Szeliski, R.: A comparison and evaluation of multi-view stereo reconstruction algorithms. In: Proc. CVPR (2006)
31. Snavely, N.: Bundler: Structure from motion (SfM) for unordered image collections. http://phototour.cs.washington.edu/bundler/
32. Snavely, N., Seitz, S.M., Szeliski, R.: Photo tourism: Exploring image collections in 3D. In: Proc. SIGGRAPH (2006)
33. Torii, A., Havlena, M., Pajdla, T.: From google street view to 3D city models. In: Proc. Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (2009)
34. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE. Trans. PAMI **12**(4), 376–380 (1991)
35. Vosselman, G., Dijkman, S.: 3D building model reconstruction from point clouds and ground plans. In: Int. Archives of Photogrammetry and Remote Sensing, vol. XXXIV-3/W4 (2001)
36. Xiao, J., Fang, T., Zhao, P., Lhuillier, M., Quan, L.: Image-based street-side city modeling. In: Proc. SIGGRAPH Asia (2009)