Accurate Multiple View 3D Reconstruction Using Patch-Based Stereo for Large-Scale Scenes

Shuhan Shen

***Abstract*— In this paper, we propose a depth-map merging based multiple view stereo method for large-scale scenes which takes both *accuracy* and *efficiency* into account. In the proposed method, an efficient patch-based stereo matching process is used to generate depth-map at each image with acceptable errors, followed by a depth-map refinement process to enforce consistency over neighboring views. Compared to state-of-the-art methods, the proposed method can reconstruct quite accurate and dense point clouds with high computational efficiency. Besides, the proposed method could be easily parallelized at image level, i.e., each depth-map is computed individually, which makes it suitable for large-scale scene reconstruction with high resolution images. The accuracy and efficiency of the proposed method are evaluated quantitatively on benchmark data and qualitatively on large data sets.**

***Index Terms*— 3D reconstruction, depth-map, multiple view stereo (MVS).**

I. Introduction

WITH fast developments of modern digital cameras, huge numbers of high resolution images could be easily captured nowadays. There is an urgent need to extract 3D structures from these images for many applications, such as architecture heritage preservation, city-scale modeling, and so on. Multiple View Stereo (MVS) reconstruction is a key step in image-based 3D acquisition and receives more and more interests recently. Although great efforts have been made in MVS and some efficient algorithms have been proposed, especially for small and compact objects, the handling of large-scale scenes using high resolution images (6 Megapixel and above) is still an open problem.

According to [1], MVS algorithms can be divided into four classes, called voxel based methods [2]–[4], surface evolution based methods [5]–[8], feature point growing based methods [9]–[13], and depth-map merging based methods [14]–[22]. Among these classes, the voxel based methods are only suited for small compact objects within a tight enclosing box; the surface evolution based methods require a reliable initial guess which is difficult to obtain for large-scale scenes; the feature point growing based methods spread points reconstructed in textured regions to untextured ones which may leave holes in final results; the depth-map based methods have been proved to be more adapted to large-scale scenes but their performance is usually lower than those by others in terms of accuracy and completeness.

In this paper we propose a depth-map merging based MVS method for large-scale scenes which takes both *accuracy* and *efficiency* into account. The key of our method is an efficient patch based stereo matching following a depth-map refinement process which enforces consistency over multiple views. Compared to state-of-the-art methods, the proposed method has three main advantages: 1) It can reconstruct quite accurate and dense point clouds since the patch based stereo is able to produce depth-maps with acceptable errors which can be further refined by the depth-map refinement process. 2) It is a computational efficient method which is about 10 to 20 times faster than the state-of-the-art method [11] while attaining similar accuracy. 3) It could be easily parallelized at image level, i.e., each depth-map is computed individually, which makes it suitable for large-scale scene reconstruction with high resolution images.

II. Previous Works

According to the taxonomy given in [1], the four classes of MVS, voxel based methods, surface evolution based methods, feature point growing based methods, and depth-map merging based methods are reviewed in this section.

The voxel based methods compute a cost function on a 3D volume which is a bounding box of the object. Seitz et al. [2] propose a voxel coloring framework that traverses a discrete 3D space in a generalized depth-order to identify voxels that have a unique color, constant across all possible interpretations of the scene. Vogiatzis et al. [3] use graph-cut optimization to compute the minimal surface that encloses the largest possible volume, where surface area is just a surface integral in this photo-consistency field. Since the accuracy of these methods is limited by the resolution of the voxel grid, Sinha et al. [4] present a method that does not require the surface to be lying within a finite band around the visual hull. This method uses photo-consistency to guide the adaptive subdivision of a coarse mesh of the bounding volume, which generates a multi-resolution volumetric mesh that is densely tesselated in the parts likely to contain the unknown surface. However, this method is only suited to compact objects admitting a tight enclosing box, and its computational and memory costs become prohibitive for large-scale scenes.

The surface evolution based methods iteratively evolve an initial guess to improve the photo consistency measurement.

Manuscript received June 19, 2012; revised December 20, 2012; accepted December 28, 2012. Date of publication January 11, 2013; date of current version March 14, 2013. This work was supported in part by the Natural Science Foundation of China, under Grant 61105032, the National 973 Key Basic Research Program of China, under Grant 2012CB316302, and Strategic Priority Research Program of the Chinese Academy of Sciences, under Grant XDA06030300. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Bulent Sankur.

The author is with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: shshen@nlpr.ia.ac.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2013.2237921

1057-7149/$31.00 © 2013 IEEE

Faugeras et al. [5] implement the level set to solve a set of PDEs that are used to deform an initial set of surfaces which then move toward the objects to be detected. Hernandez et al. [6] propose a method based on texture and silhouette information, and fuse the silhouette force into the snake framework. This method evolves an initial surface that is close enough to the object surface using texture and silhouette driven forces. Hiep et al. [7] use a minimum s-t cut to generate a coarse initial mesh, then refine it with a variational approach to capture small details. A main drawback of such methods is the requirement for a reliable initial guess which is hard to obtain for outdoor scenes. To this end, Cremers et al. [8] formulate the reconstruction problem as a convex functional minimization, where the exact silhouette consistency is imposed as convex constraints which restrict the domain of feasible functions. This method does not depend on initialization and can provide solutions that lie within an error bound of the optimal solution. However, this method relies on voxel representation of the space, thus it cannot be used for large-scale scenes.

The feature point growing based methods first reconstruct points in textured regions, and then expand theses points to untextured ones. Lhuillier et al. [9] propose a quasi-dense approach to 3D surface model acquisition. This method first initials sparse correspondence points of interest and then resamples quasi-dense points from the quasi-dense disparity map to densify the feature points to overcome the sparseness of the points of interest. Goesele et al. [10] propose a method to handle Internet photo collections containing obstacles using global and local view selection plus a region growing process from reconstructed SIFT [23] features. Based on these methods, Furukawa et al. [11] present quite an accurate Patch-based MVS (PMVS) approach that starts from a sparse set of matched keypoints, and repeatedly expands these till visibility constraints are invoked to filter away false matches. This method is now considered as the state-of-the-art MVS method. Based on PMVS, Wu et al. [12] propose a Tensor-based MVS (TMVS) method for quasi-dense 3D reconstruction which combines the complementary advantages of photoconsistency, visibility and geometric consistency enforcement in MVS under the 3D tensors framework. These feature point growing based methods aim at reconstructing a global 3D model by using all the images available simultaneously, thus they suffer from the scalability problem as the number of images grows. Although this problem can be partially solved by decomposing input images into clusters that have small overlap [13], the computational complexity remains quite high for large-scale scenes.

The depth-map merging based methods are natural extensions from binocular stereo to multiple views. Such methods first compute depth-maps at each view and then merge them together into a single model by taking visibility into account. Goesele et al. [14] use Normalized Cross Correlation (NCC) based pixel window matching techniques to produce depth-maps then merge them with volumetric integration. Strecha et al. [15] jointly model depth and visibility as a hidden Markov Random Field, and use EM-algorithm to optimize the model parameters. Merrell et al. [16] first use a computationally cheap stereo algorithm to generate potentially

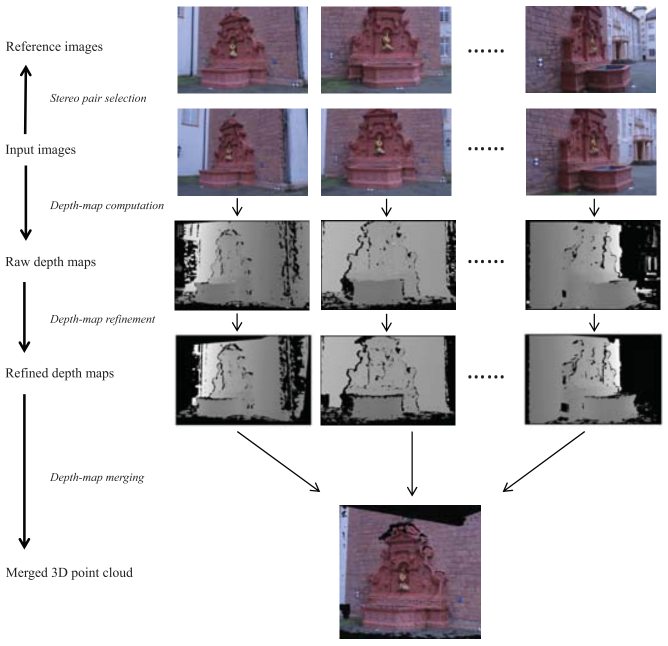


Fig. 1. Framework of the proposed method.

noisy, overlapping depth-maps, and then fuse these depth-maps to obtain an integrated surface based on visibility relations between points. Zach et al. [17] present a method to globally optimize an energy functional consisting of a total variation regularization force and an L_1 data fidelity term. Bradley et al. [18] propose a method which uses robust binocular stereo with scaled matching windows, followed by adaptive point-based filtering of the merged point clouds. Campbell et al. [19] store multiple depth hypotheses for each pixel and use a spatial consistency constraint to extract the true depth in the discrete Markov Random Field framework. Liu et al. [20] produce high quality MVS reconstruction results using continuous depth-maps generated by variational optical flow. But this method requires the visual hull as an initialization. Li et al. [21] generate depth-maps using DAISY [24] feature, and use two stages of bundle adjustment to optimize the positions and normals of 3D points. Tola et al. [22] also use DAISY feature to generate depth-maps, and then merge them by consistency checking at neighboring views. This method is similar to our method, but ours uses patch based stereo instead of merely matching DAISY features along epipolar lines, which can produce more accurate depth-maps without diminishing the computational efficiency.

III. MVS Using Patch Based Stereo

The proposed method consists of four steps: stereo pair selection, depth-map computation, depth-map refinement, and depth-map merging. The framework of the method is illustrated in Fig. 1. For each image in the input image set, we select a reference image to form a stereo pair for depth-map computation. Since these raw depth-maps generated by stereo vision may contain noises and errors, we refine each of them by consistency checking using its neighboring depth-maps. Finally all the refined depth-maps are merged together to get a final reconstruction. Next we elaborate on each of the steps.

A. Stereo Pair Selection

For each image in the image set, we need to select a reference image for it for stereo computation. The selection of stereo image pair is important not only for the accuracy of the stereo matching but also for the final MVS result. Stereo pair selection is a relatively easy task for street-side view cameras on the vehicle [25]–[28] or cameras in a controlled environment like the Middlebury benchmark data [1], but needs to be carefully designed for unordered images. A *good* candidate reference image should have a similar viewing direction as the target image, and have a suitable baseline neither too short to degenerate the reconstruction accuracy nor too long to have less common coverage of the scene.

We use a method similar to [21] to select eligible stereo pairs. Suppose we have n images, and for the i-th one, we compute $\theta_{ij}, j = 1, \ldots, n$ which is the angle between principal view directions of camera i and j. If the camera poses are calibrated using structure-from-motion (SfM) algorithms [29]–[31], a set of sparse 3D points and their visibilities are generated as a by-product of SfM, then a better θ_{ij} could be computed as the average of the the angles between the visible points and the camera centers, respectively, for camera i and j. Besides θ_{ij}, we compute another parameter $d_{ij}, j = 1, \ldots, n$ for each image i, which is the distance between optical centers of camera i and j. Then for images that satisfy $5^\circ < \theta_{ij} < 60^\circ$, we compute the median $\bar{d}$ of their d_{ij}, and remove images whose $d_{ij} > 2\bar{d}$ or $d_{ij} < 0.05\bar{d}$. After these computations, if the number of remaining images is less than k_1, they are considered as neighboring images of the image i, denoted as $N(i)$. Otherwise, the remaining images are sorted in ascending order according to $\theta_{ij} \cdot d_{ij}$, and the first k_1 images form the neighboring images $N(i)$ (in this paper we set $k_1 = 10$). Finally, the one with minimal $\theta_{ij} \cdot d_{ij}$ among $N(i)$ is selected as the i-th image's reference image to form a stereo pair.

B. Depth-Map Computation

For each eligible stereo pair, we follow the idea in [32] to compute the depth-map. The core idea is that, for each pixel in the input image we try to find a good support plane that has minimal aggregated matching cost with the reference image, as shown in Fig.2.

The support plane f is essentially a local tangent plane of the scene surface, which is represented by a 3D point X_i and its normal n_i in the related camera's coordinate system, as shown in Fig. 3.

For the i-th input image I_i in the image set, given its reference image I_j, and the associated camera parameters $\{K_i, R_i, C_i\}$ and $\{K_j, R_j, C_j\}$, where K is the intrinsic parameters, R is the rotation matrix, and C is the camera center, we first assign each pixel p in I_i to a random 3D plane. Suppose p's homogeneous coordinate is:

$$p = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{1}$$

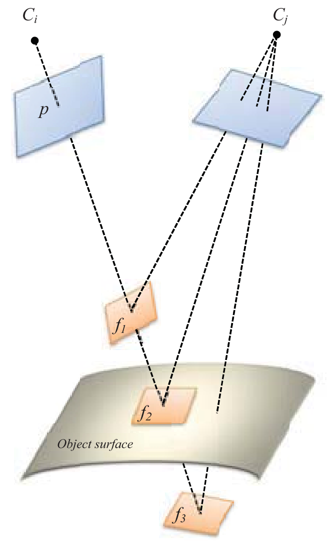
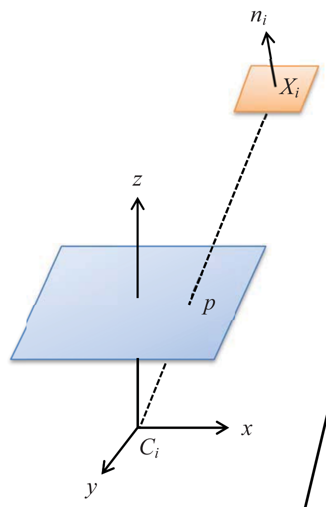


Fig. 2. For each pixel p in the input image, we estimate its corresponding 3D plane. C_i and C_j are the camera centers of the input and reference images respectively, f_1, f_2 and f_3 are three 3D planes in p's viewing ray. Obviously f_2 has the minimal aggregated matching cost.



像素坐标转换为物理坐标？

Fig. 3. Support plane is represented by a 3D point X_i and its normal n_i in camera C_i's coordinate, where C_i is the camera center of the i-th input image, and $C_i - xyz$ is the camera's coordinate.

The 3D point X_i must lie in the viewing ray of p, we select a random depth λ in the depth range $\lambda \in [\lambda_{min}, \lambda_{max}]$, then X_i is computed in C_i's coordinate as:

$$X_i = \lambda K_i^{-1} p \tag{2}$$

Then we assign the normal of the plane randomly in camera C_i's spherical coordinate, as:

$$n_i = \begin{bmatrix} cos\theta sin\phi \\ sin\theta sin\phi \\ cos\phi \end{bmatrix} \tag{3}$$

where, θ is a random angle in the range $[0^\circ, 360^\circ]$, and ϕ is a random angle in the range$[0^\circ, 60^\circ]$. These range settings come from a simple assumption that a patch is visible in image I_i when the angle between the patch normal n_i and the z axis of camera C_i's coordinate system is below a certain threshold (in this paper we set this threshold as 60°).

The above random initialization process is very likely to have at least one good guess for each scene plane in the image, especially for high resolution images in which each scene plane contains more pixels which means more guesses than low resolution ones. We should note that, once the depth-map for image I_i is computed, we can improve the purely random initialization process when computing the depth-map on I_i's reference image I_j. In this scenario, the depth and patch normal of each pixel in I_i's depth-map could be warped to I_j as an initial estimate when computing I_j's depth-map, and pixels in I_j that do not have mappings between I_i and I_j still use random initializations. In this manner, we can assign each warped pixel in I_j a better plane initially than random guess since this plane is consistent for the stereo pair I_i and I_j.

According to [33], given the projection matrices for the two cameras $P = [\mathbf{I}_{3\times3} \mid \mathbf{0}_3]$ and $P' = [R \mid t]$, and a plane defined by $\pi^T X = 0$ with $\pi = (V^T, 1)^T$, then the homography H induced by the plane is:

$$H = R - tV^T \tag{4}$$

Here, $\mathbf{I}_{3\times3}$ is the 3×3 identity matrix and $\mathbf{0}_3$ is a zero 3-vector, which indicates that the world coordinate is chosen to coincide with camera P. In this paper, the camera parameters of the image pair are $\{K_i, R_i, C_i\}$ and $\{K_j, R_j, C_j\}$, and the plane $f_p = \{X_i, n_i\}$ is defined in camera C_i's coordinate. Thus, the projection matrices and plane parameters can be translated into standard forms (put the world origin at C_i), as:

$$P_i = K_i[\mathbf{I}_{3\times3} \mid \mathbf{0}_3], \quad P_j = K_j[R_jR_i^{-1} \mid R_j(C_i - C_j)],$$
$$V^T = -\frac{n_i^T}{n_i^T X_i}$$

According to Eq. 4, the homography for the cameras $\widetilde{P}_i = [\mathbf{I}_{3\times3} \mid \mathbf{0}_3]$ and $\widetilde{P}_j = [R_jR_i^{-1} \mid R_j(C_i - C_j)]$ is:

$$\widetilde{H}_{ij} = R_jR_i^{-1} + \frac{R_j(C_i - C_j)n_i^T}{n_i^T X_i}$$

Applying the transformations K_i and K_j to the images we obtain the cameras $P_i = K_i\widetilde{P}_i$, $P_j = K_j\widetilde{P}_j$ and the resulting induced homography is:

$$H_{ij} = K_j(R_jR_i^{-1} + \frac{R_j(C_i - C_j)n_i^T}{n_i^T X_i})K_i^{-1} \tag{5}$$

We set a square window B centered on pixel p, where $B = w \times w$ (in this paper we set $w = 7$ pixels). For each pixel q in B we find its corresponding pixel in the reference image I_j using homography mapping $H_{ij}(q)$. Then the aggregated matching cost $m(p, f_p)$ for pixel p is computed as one minus the Normalized Cross Correlation (NCC) score between q and $H_{ij}(q)$, as:

$$m(p, f_p) = 1 - \frac{\sum_{q\in B}(q - \overline{q})(H_{ij}(q) - \overline{H_{ij}(q)})}{\sqrt{\sum_{q\in B}(q - \overline{q})^2 \sum_{q\in B}(H_{ij}(q) - \overline{H_{ij}(q)})^2}} \tag{6}$$

Note that some more complex and robust aggregation techniques, like [34]–[36], could be used to generate more reliable results than NCC. However, high resolution images could provide more reliable matches than low resolution ones and a simple NCC is reliable enough to measure the photometric consistency. Besides, most unreliable pixels generated by NCC could be removed in the depth-map refinement step which makes the final reconstruction result of NCC almost equivalent to those of other complex aggregation methods. Thus, in this paper we use simple NCC as the aggregated matching cost which is the same as [11].

After the initialization, each pixel in image I_i is associated with a 3D plane. Then we process pixels in I_i one by one to refine the planes in n_2 iterations. At odd iterations, we start from the top-left pixel and traverse in row wise order until we reach the bottom-right pixel. At even iterations, we reverse the order to visit the pixel from the bottom-right to the top-left pixel, also in row wise order. In this paper we set the number of plane refinement as $k_2 = 3$.

At each iteration, each pixel has two operations, called *spatial propagation* and *random assignment*. Spatial propagation is used to compare and propagate the planes of neighboring pixels to that of the current pixel. In odd iterations, the neighboring pixels are the left, upper, and upper-left neighbors, and in even iterations are the right, lower, and lower-right neighbors. Let p_N denote the neighborhood of the current pixel p, and f_{p_N} denote p_N's plane, we use the matching cost in Eq.6 to check the condition $m(p, f_{p_N}) < m(p, f_p)$. If this condition is satisfied, we consider f_{p_N} is a better choice for p compared to its current plane f_p, and propagate f_{p_N} to p, i.e. set $f_p = f_{p_N}$. This spatial propagation process relies on the fact that neighboring pixels are very likely to have similar 3D planes especially for high resolution images. Theoretically, even a single good guess is enough to propagate this plane on to other pixels of the region after the first and second spatial propagations.

For each pixel p, after spatial propagation, we further refine the plane f_p using random assignment. The purpose of the random assignment is to further reduce the matching cost in Eq.6 by testing several random plane parameters. Given a range $\{\Delta\lambda, \Delta\theta, \Delta\phi\}$, we 1) select a random plane parameter $\{\lambda', \theta', \phi'\}$ in the range $\lambda' \in [\lambda - \Delta\lambda, \lambda + \Delta\lambda]$, $\theta' \in [\theta - \Delta\theta, \theta + \Delta\theta]$, $\phi' \in [\phi - \Delta\phi, \phi + \Delta\phi]$. 2) Compute the new plane $f_p' = \{X_i', n_i'\}$ using Eq.2 and Eq. 3. 3) If $m(p, f_p') < m(p, f_p)$, we accept $f_p = f_p'$ and set $\lambda = \lambda'$, $\theta = \theta'$, $\phi = \phi'$. 4) We halve the range $\{\Delta\lambda, \Delta\theta, \Delta\phi\}$. 5) Go back to step one. The above process is repeated for k_3 times. In this paper, we set the initial range and repetition time as: $\Delta\lambda = \frac{\lambda_{max}-\lambda_{min}}{4}$, $\Delta\theta = 90^\circ$, $\Delta\phi = 15^\circ$, $k_3 = 6$. This random assignment process progressively reduces the search range in order to capture depth and normal details.

The spatial propagation and random assignment idea has already been successfully applied in the patchmatch stereo method [32] and the hybrid recursive matching (HRM) method [37]. This paper extends the idea of [32] to Multiple View Stereo for large-scale scenes using high resolution images. Thus, the novelty of the proposed approach is to modify the pathmatch stereo algorithm [32] in a proper way that makes it more powerful and efficient for the large-scale MVS problem. The main difference between the proposed approach and the method in [32] is that the plane is defined in the image coordinate in [32] but in the camera's coordinate in our work because the stereo pair is not rectified in our paper. Besides, by taking full advantage of multiple high resolution images, we make three simplifications compared with [32] in order to reduce the computational expense. Firstly, the aggregated matching cost $m(p, f_p)$ in our work is a simple normalized cross correlation rather than the more complex adaptive support weight version in [32], because the NCC is reliable enough for high resolution images and the remaining errors could be removed in the following depth-map refinement step. Secondly, we only use the spatial propagation compared to spatial and view propagation in [32] because we only compute the depth-map on I_i, not on both I_i and I_j as in [32]. Thirdly, the method in [32] contains another post-processing step that applies occlusion treatment via left/right consistency checking and fills invalidated pixels as well. Our method does not have this process because the following depth-map refinement step could reach the similar effects.

After the spatial propagation and random assignment processes, we remove unreliable points in the depth-map whose aggregated matching costs are above a certain threshold τ_1 (in this paper we set $\tau_1 = 0.3$).

C. Depth-Map Refinement

Since the raw depth-maps may not completely agree with each other on common areas due to depth errors, a refinement process is carried out to enforce consistency over neighboring views. For each point p in image I_i, we back project it to 3D using its depth λ and the camera parameters, as:

$$X = \lambda R_i^T K_i^{-1} p + C_i \tag{7}$$

where p is the homogeneous coordinate defined in Eq. 1, X is the 3D point in the world coordinate. Then we project X to I_i's neighboring images $N(i)$ which are generated in the stereo pair selection stage. Suppose N_k is the k-th neighboring image in $N(i)$, we denote $d(X, N_k)$ as the depth of X with respect to camera N_k and denote $\lambda(X, N_k)$ as the depth value computed at the projection of X in N_k using N_k's depth-map. If $\lambda(X, N_k)$ is close enough to $d(X, N_k)$, i.e., $\frac{|d(X,N_k)-\lambda(X,N_k)|}{\lambda(X,N_k)} < \tau_2$, where τ_2 is a threshold (in this paper we set $\tau_2 = 0.01$), we say X is consistent in I_i and N_k. If X is consistent for at least k_4 neighboring images in $N(i)$ (in this paper we set $k_4 = 2$), it is regarded as a reliable scene point and its corresponding pixel p in I_i's depth-map is retained, otherwise it is removed.

After the above refinement process, most errors could be removed, which results in a relatively clean depth-map at each view.

D. Depth-Map Merging

After refinement, all the depth-maps could be merged to represent the scene. However, merging depth-maps directly may contain lots of redundancies, because different depth-maps may have common coverage of the scene, especially for neighboring images. In order to remove these redundancies, the depth-maps are further reduced by neighboring depth-map test. As illustrated by Fig. 4, for each pixel in camera

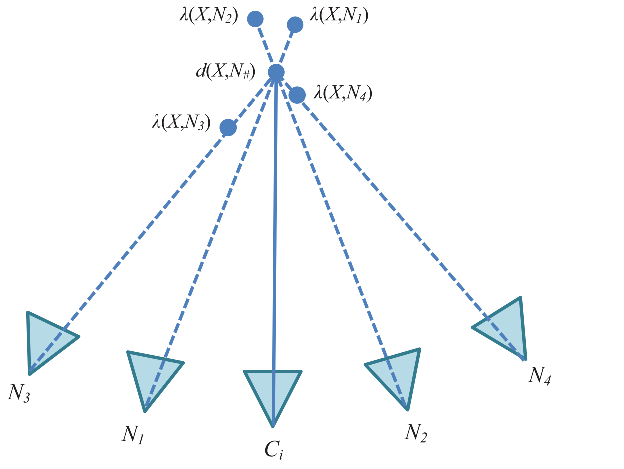


Fig. 4. Illustration of redundancy removing by depth-map test. For each pixel in camera C_i's depth-map, we back project it to 3D as X using Eq. (7) and reproject X to C_i's neighboring cameras $N_1 \dots N_4$. Define $d(X, N_\#)$ by the depth of X with respect to camera $N_\#$ and define $\lambda(X, N_\#)$ by the depth value computed at the projection of X in $N_\#$ using $N_\#$'s depth-map. If $d(X, N_\#) < \lambda(X, N_\#)$, we say the projection of X in $N_\#$ is occluded and remove it from $N_\#$'s depth-map, like in N_1 and N_2. If $\frac{|d(X,N_\#)-\lambda(X,N_\#)|}{\lambda(X,N_\#)} < \tau_2$, we say X is reduplicate in C_i and $N_\#$ and remove it from $N_\#$'s depth-map, like in N_4. Points not satisfied the above two conditions are retained in their depth-maps, like in N_3.

C_i's depth-map, we back project it to 3D as X using Eq. 7 and reproject X to C_i's neighboring cameras. If the depth of X with respect to the neighboring camera is smaller that the depth value computed at the projection of X in the neighboring camera's depth-map, like cameras N_1 and N_2 in Fig. 4, we say the projection of X in this neighboring camera is occluded and remove it from this neighboring camera's depth-map. If these two depth values are close enough, like the camera N_4 in Fig. 4, we say the projection of X in this neighboring camera represents the same point as X which is a redundancy and also remove it from this neighboring camera's depth-map.

Finally, all depth-maps are back projected into 3D, and merged into a single point cloud. The final point cloud is usually quite dense especially when using high resolution images. If we want to make it sparse, we can simply just back project points at sparse locations in the depth-maps. For example, using only points at image locations $(2n, 2n)$ in the depth-map will approximately reduce the size of the point cloud to a quarter of the size that use all points. This gives us a way to control the point cloud size according to memory and storage limitations.

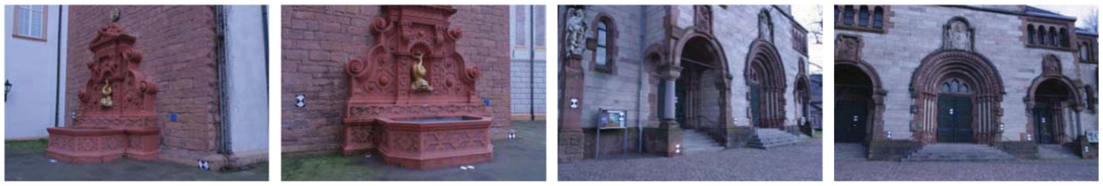
IV. Experimental Results

A. Two State-of-the-Art Methods for Comparison

We compared our method with two state-of-the-art methods [11], [22]. The PMVS [11] method is a feature point growing based method which repeatedly expands a sparse set of matched keypoints and use visibility constraints to filter away false matches. The DAISY based method [22] is a depth-map merging based method for ultra high-resolution image sets. This method is similar to ours, but its stereo

TABLE I
PARAMETER SETTINGS OF THE PROPOSED METHOD

Parameter	Section	Description	Value
k_1	III.A	Maximum number of neighboring images	10
w	III.B	Size of the matching window, $w \times w$	7 pixels
k_2	III.B	Number of iterations for plane refinement	3
$\Delta\lambda$	III.B	Depth range for random plane assignment	$\frac{\lambda_{max}-\lambda_{min}}{4}$
$\Delta\theta$	III.B	Angle range for random plane assignment	90°
$\Delta\phi$	III.B	Angle range for random plane assignment	15°
k_3	III.B	Repetition time for random assignment	6
τ_1	III.B	Matching cost threshold for reliable points	0.3
τ_2	III.C and III.D	Threshold to measure depth closeness	0.01
k_4	III.C	Minimal number of consistent neighboring images	2


(a)

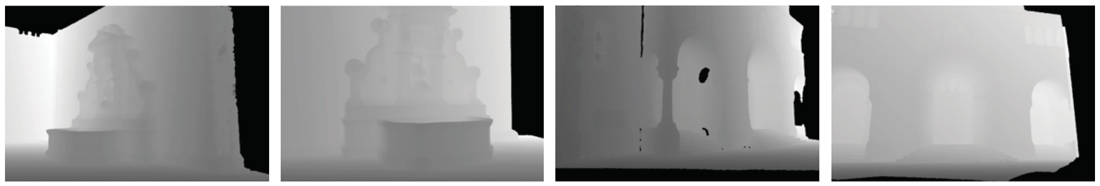

(b)

Fig. 5. Sample images and their ground truth depth-maps of the benchmark data sets. In both (a) and (b), the left two images are from Fountain-P11 and the right two images are from Herz-Jesu-P8.

matching process is based on DAISY [24] descriptors at each pixel.

These two methods are now considered as state-of-the-art MVS methods, and can be used for large-scale scenes as ours. In the next three subsections, the proposed method, together with the PMVS and DAISY methods, are evaluated quantitatively and qualitatively on different data sets. All the experiments are implemented on a Intel 2.8GHz Quad Core CPU with 16G RAM.

B. Parameter Settings

The proposed method has nine parameters, and we have already discussed their value settings in Section III. Table I is a summary.

The key of the DAISY method is the DAISY descriptor, and we set the DAISY parameters as: $R = 8$, $Q = 2$, $T = 4$, and $H = 4$. For the details of these parameters one can refer [22], [24]. The authors of [22] suggest first computing depths at sparse locations of the image in order to constrain the search range for neighboring pixels. Thus, in the experiments we first select control pixels using a sampling step of 10 pixels on the image, and compute their depths. Then we compute depths of other pixels with depth range constrained by its closest four neighboring control pixels.

The PMVS method may run out of memory for large number of high resolution images, thus we use a clustered version of PMVS [13] which first decomposes the input images into a set of image clusters of managable size and then run PMVS on each cluster separately. The authors of [11], [13] provide the source codes of PMVS and clustered-PMVS, and we set its parameters as: $level = 0$, $csize = 1$, $threshold = 0.7$, $wsize = 7$, $minImageNum = 3$. $level$ specifies the level in the image pyramid that is used for the computation, and $level = 0$ means the original resolution images are used. $csize$ controls the density of reconstructions, and $csize = 1$ means the software tries to reconstruct a patch in every pixel. $threshold = 0.7$ is a threshold for photometric consistency measurement which is the same as $1 - \tau_1$ in our method. $minImageNum = 3$ means each 3D point must be visible in at least 3 images for being reconstructed which is suggest by the authors. These parameter settings ensure that the PMVS method tries to reconstruct a 3D point at every pixel with full resolution images, which is the same as ours. For a full description of these parameters, one can refer [11], [13], [38] for details.

The proposed method, as well as the DAISY method, could be easily parallelized at image level, i.e., each depth-map is computed individually. Thus, in these two methods we use four cores for parallel computing. For the PMVS method, we set its parameter $CPU = 4$ which indicates the code to use four cores on the Quad Core CPU as ours.

Since the memory consumption is a key problem for large-scale reconstructions, we analyze the memory requirements of

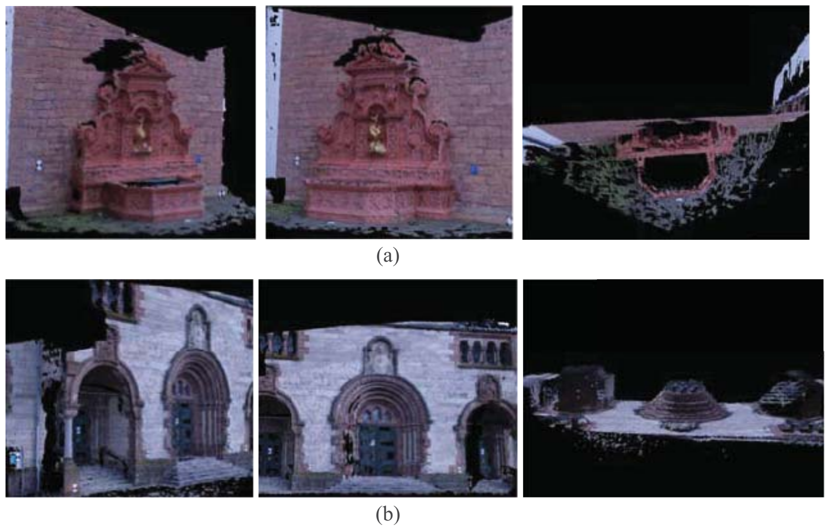


Fig. 6. Final reconstruction results (colorized point cloud rendering) of the proposed method on the benchmark data sets. In both (a) and (b), the results are rendered from three different view points (the right one is seen from the top view).

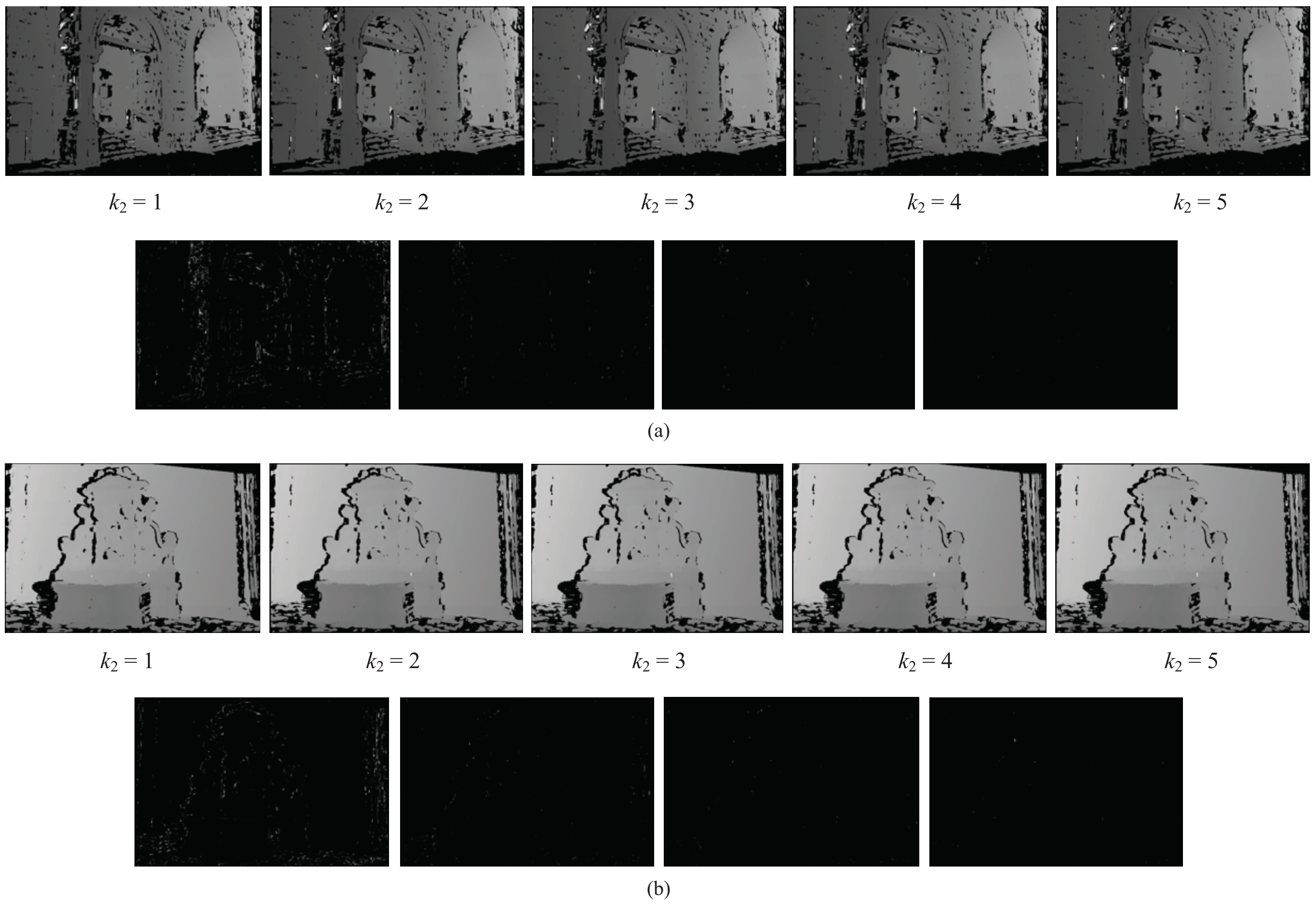


Fig. 7. Depth-map generated after $k_2 = 1, 2, \ldots, 5$ iterations in the depth-map computation process. In both (a) and (b), the top row are depth-maps generated after one to five iterations, respectively, and the bottom row, are absolute depth differences between neighboring iterations.

three evaluated methods. The PMVS method needs to load all images (or images in a cluster) simultaneously, which means it requires $H \times W \times 4 \times n$ of memory, where H and W are the height and width of the image respectively, 4 means four bytes since color images are converted to single-precision floating point gray images, and n is the size of the image set. Apparently, the PMVS method may suffer from the scalability problem (out of memory) as the number of images grows. On

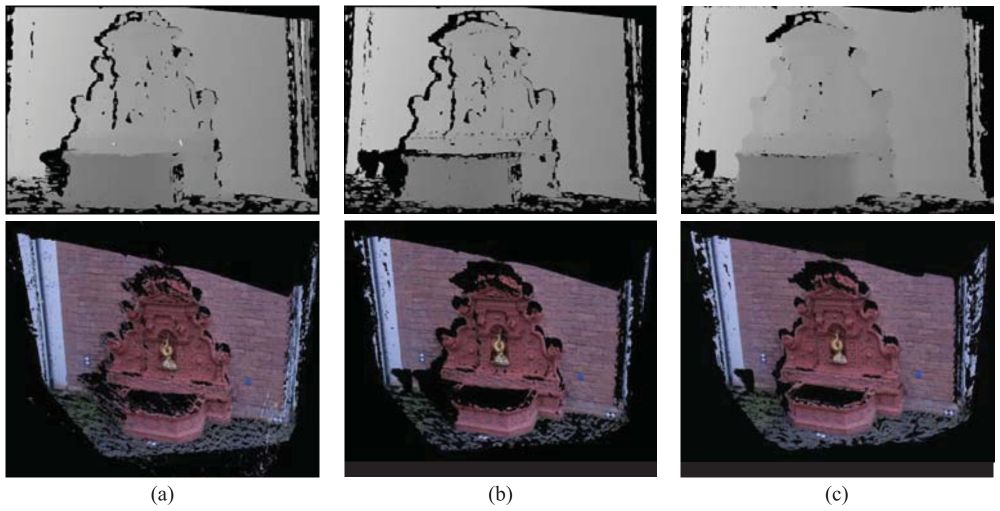


Fig. 8. Depth-map and back projected 3D points (colored rendering) after each step for the fourth image in Fountain-P11. In (a)–(c), the top image is the depth-map and the bottom is the the back projected 3D points.

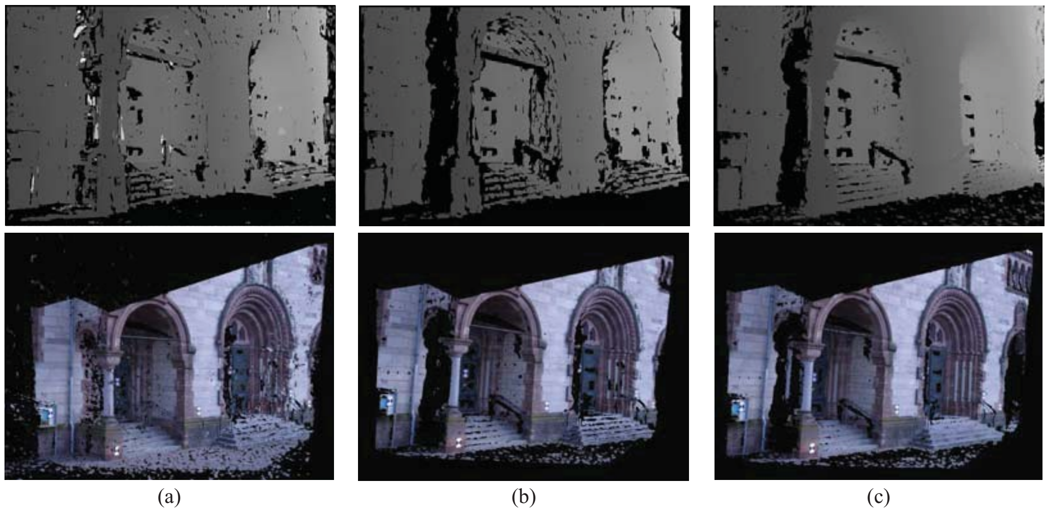


Fig. 9. Depth-map and back projected 3D points (colored rendering) after each step for the second image in Herz-Jesu-P8. In (a)–(c), the top image is the depth-map and the bottom is the back projected 3D points.

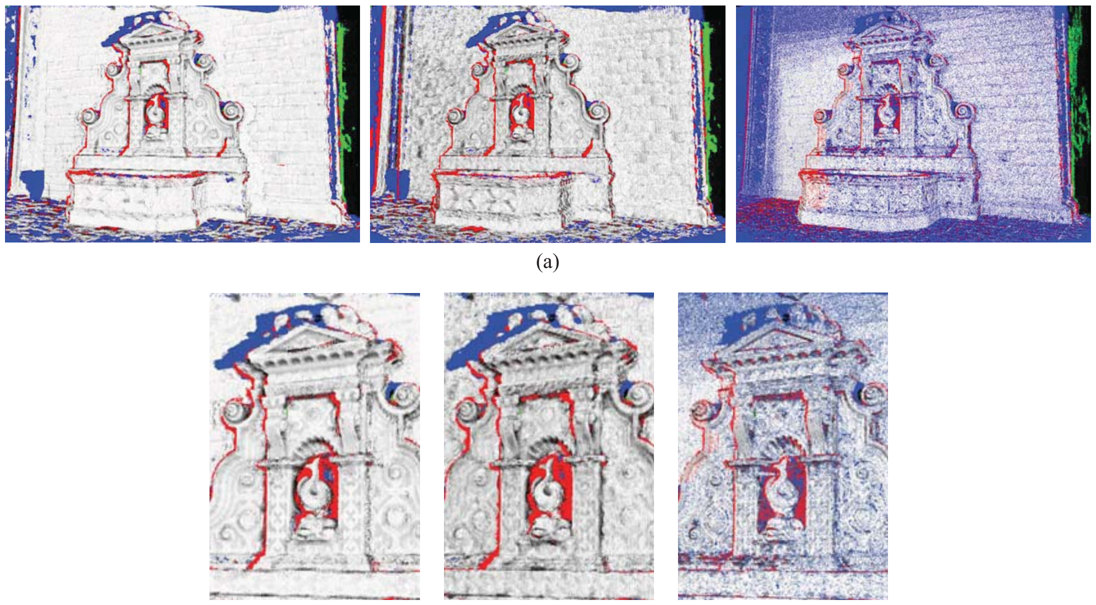
the contrary, the proposed and the DAISY methods can avoid this scalability problem since each depth-map is computed and refined individually. The DAISY method loads two images and computes descriptors on each one of their pixels for depth-map computation. Each descriptor requires 36 floating point numbers, which means that the DAISY method requires $H \times W \times 36 \times 4 \times 2 = H \times W \times 288$ of memory. The proposed method also loads two images for depth-map computation and requires $H \times W \times 4 \times 2 = H \times W \times 8$ of memory. Obviously, the proposed method has much lower memory requirement compared with the DAISY and PMVS methods.

TABLE II
NUMBERS OF CORRECT AND ERROR PIXELS AFTER EACH STEP FOR THE FOURTH IMAGE IN FOUNTAIN-P11 AND THE SECOND IMAGE IN HERZ-JESU-P8

Step	Fountain-P11		Herz-Jesu-P8	
	Correct pixels	Error pixels	Correct pixels	Error pixels
depth-map computation	5 174 163	737 211	4 477 824	990 058
depth-map refinement	4 603 032	165 868	3 992 684	176 736
depth-map merging	5 254 683	260 351	4 306 332	266 224

C. Quantitative Evaluation on Benchmark Data

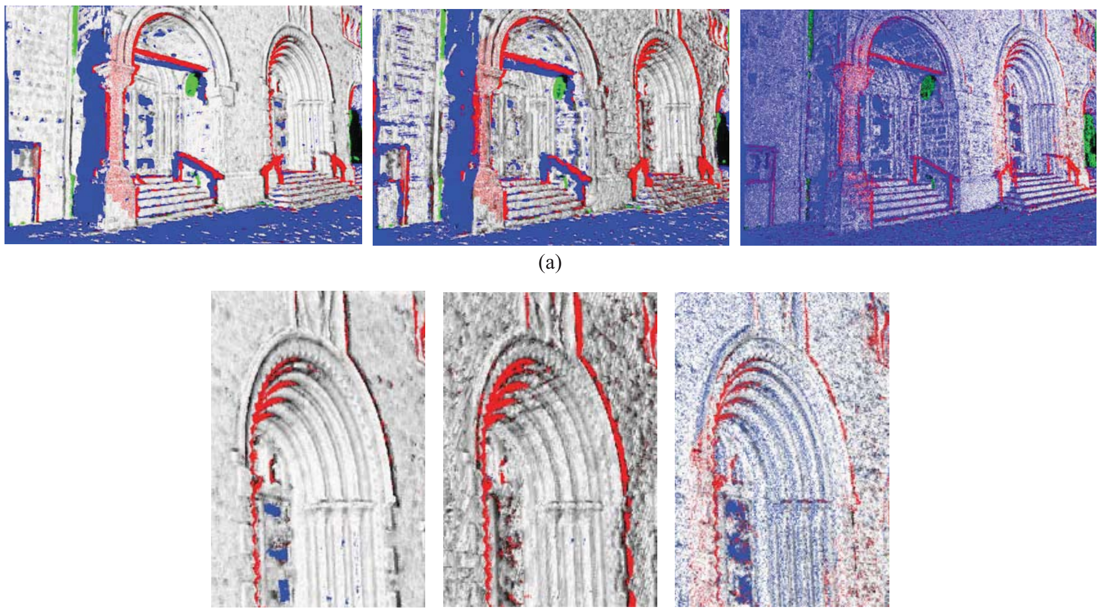
To quantitatively evaluate our method, two benchmark data sets, Fountain-P11 and Herz-Jesu-P8, provided by Strecha et al. [39] [40] are used. Fountain-P11 and Herz-Jesu-P8 has 11 and 8 images respectively with 3072×2048 resolution (6 Megapixel). Fig. 5(a) shows some sample images in the data sets. The ground truth is obtained by laser scanning (LIDAR), which is a single high resolution triangle mesh model. The benchmark data site [40] can quantitatively evaluate the MVS results in triangulated mesh model with the ground truth. This



(a)

(b)

Fig. 10. In both (a) and (b), from left to right: depth error maps for the fourth image in Fountain-P11 using the proposed method, the DAISY method, and the PMVS method, respectively. In all the images, blue pixels encode missing depth values by the MVS method, green pixels encode missing ground truth data, red pixels encode an error e larger than τ_e, and pixels with errors between 0 and τ_e are encoded in gray $[255, 0]$.



(a)

(b)

Fig. 11. In both (a) and (b), from left to right: depth error maps for the second image in Herz-Jesu-P8 using the proposed method, the DAISY method, and the PMVS method, respectively. In all the images, blue pixels encode missing depth values by the MVS method, green pixels encode missing ground truth data, red pixels encode an error e larger than τ_e, and pixels with errors between 0 and τ_e are encoded in gray $[255, 0]$.

mesh model could be easily generated from point cloud using some meshing algorithm [41]. However, the three evaluated methods in this section are all output 3D point could, thus a more direct way is to compare the raw outputs in point form rather than in a refined mesh form. To make this comparison feasible, we first project the ground truth to each image to generate ground truth depth-maps. Since the ground truth model is in 3D triangulated mesh form, the ground truth depth for each pixel is obtained from a 3D triangle mesh by computing the depth of the first triangle intersection with the camera ray going through this pixel. After this process, the ground truth depth-maps are generated. Fig. 5(b) shows some sample ground truth depth-maps.

Three evaluated MVS methods are used to reconstruct point cloud on the benchmark data with the parameters given in Section IV.B, and the results generated by the

TABLE III

NUMBERS OF CORRECT AND ERROR PIXELS USING THREE EVALUATED METHODS FOR THE FOURTH IMAGE IN FOUNTAIN-P11 AND THE SECOND IMAGE IN HERZ-JESU-P8

Method	Fountain-P11			Herz-Jesu-P8		
	Correct pixels	Error pixels	Error/Correct	Correct pixels	Error pixels	Error/Correct
Proposed method	5 254 683	260 351	4.9%	4 306 332	266 224	6.2%
DAISY	5 163 432	263 544	5.1%	4 107 572	382 104	9.3%
PMVS	3 853 304	246 696	6.4%	2 838 744	346 312	12.2%

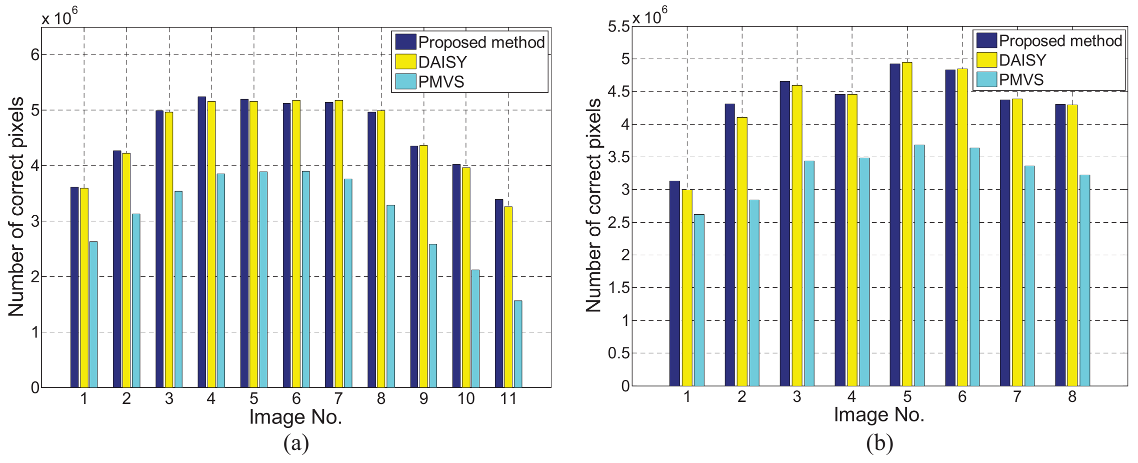


Fig. 12. Number of correct pixels using three evaluated methods. For each pixel, its depth is considered to be correct if the depth error e is below $\tau_e = 0.01$. (a) Is the result for Fountain-P11 which contains 11 images. (b) Is the result for Herz-Jesu-P8, which contains eight images.

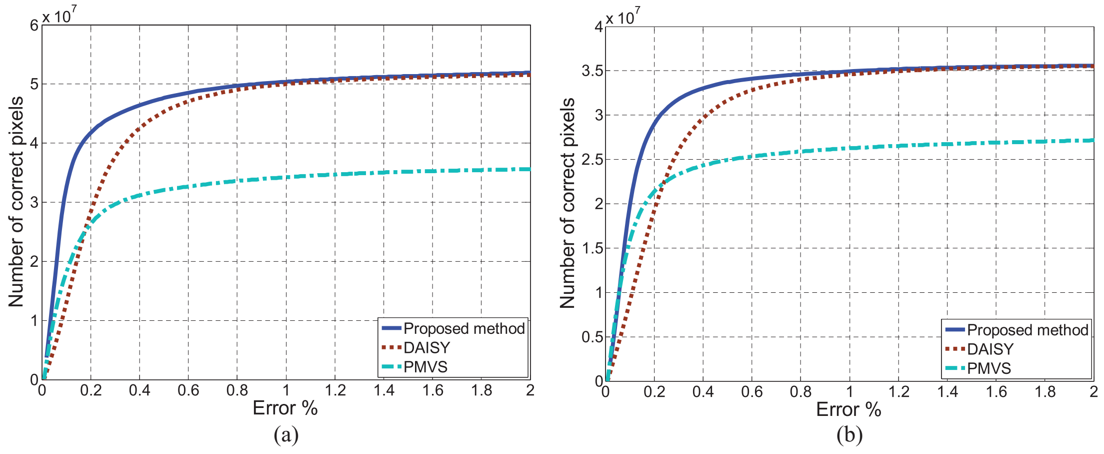


Fig. 13. Total number of correct pixels in all images as a function of the error threshold. (a) Result for Fountain-P11. (b) Result for Herz-Jesu-P8.

proposed method are shown in Fig. 6. Then we project the point cloud computed by different methods to each image for quantitative evaluation with the ground truth depth-map.

For each pixel in the image, we denote the depth computed by MVS method by d and denote the depth of the ground truth by d_{gt}, the depth error between the computed depth and the ground truth could be measured as:

$$e = \frac{\|d - d_{gt}\|}{d_{gt}} \quad (8)$$

If the depth error e is below a threshold τ_e, the depth d is considered as correct (in this paper we set $\tau_e = \tau_2 = 0.01$). Eq. 8 is a quantitative measurement about how *accurate* a reconstructed depth is, and the following evaluations are all based on this measurement.

The first experiment illustrates the effect of the iteration number k_2 for plane refinement in the depth-map computation process, and shows why we choose $k_2 = 3$ as the iteration number. We select the fourth image in Fountain-P11 and the second image in Herz-Jesu-P8, and compute their depth-maps with different values of k_2 ($k_2 = 1, 2, \ldots, 5$) as shown in Fig. 7. In both Fig. 7(a) and 7(b), the top row are depth-maps generated after one to five iterations respectively, and the bottom row are absolute depth differences between neighboring iterations. The results show that quite a few depth

Fig. 14. Sample images of the large data sets. The left two images are from Hull and the right two images are from Life Science Building.

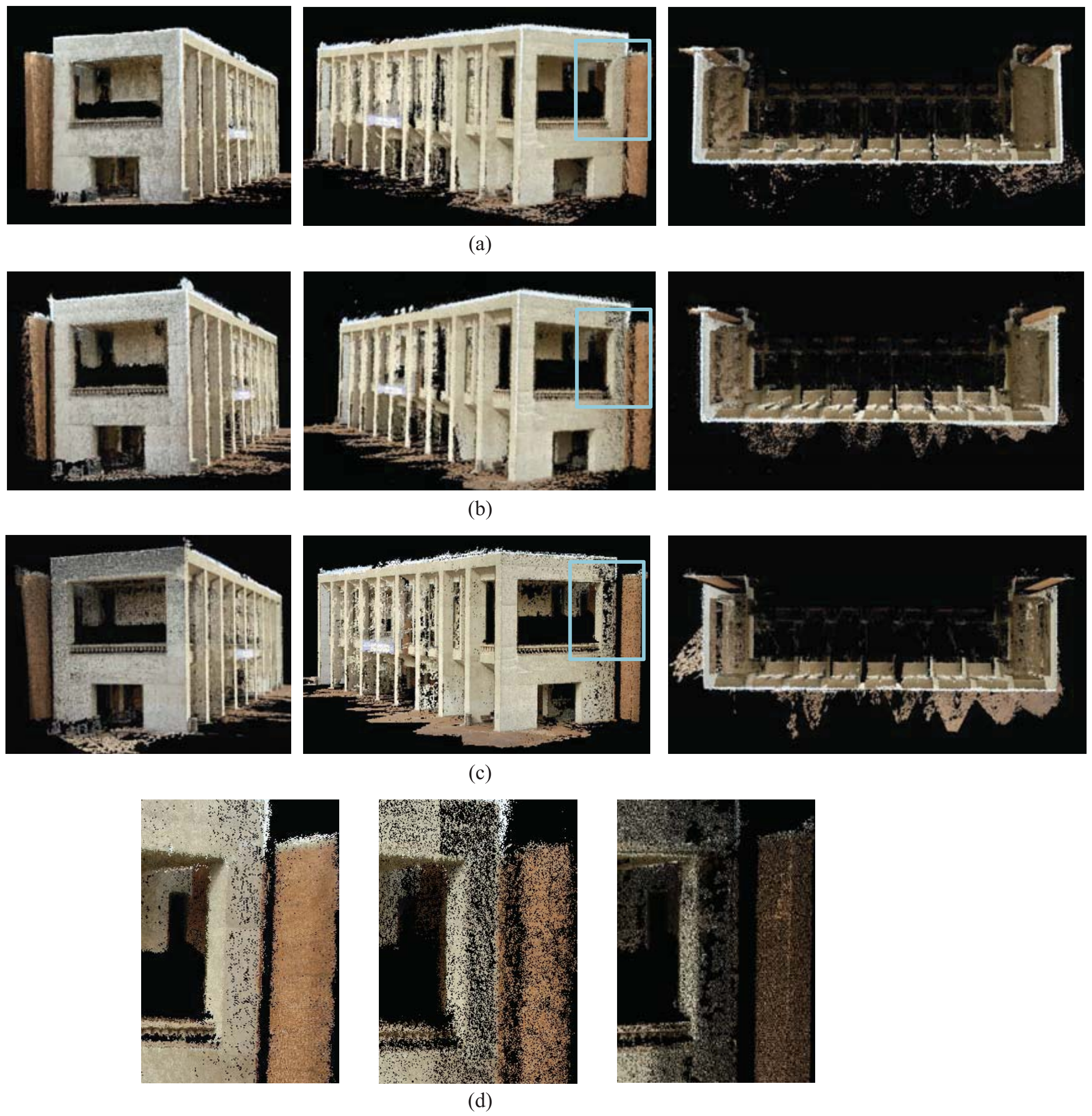


Fig. 15. Final reconstruction results (colorized point cloud rendering) of three evaluated methods on the Hull data set. In (a)–(c), the results are rendered from three different view points (the right one is seen from the top view).

changes could be found after three iterations, thus setting $k_2 = 3$ could provide a good balance between *accuracy* and *efficiency*.

The proposed method has three steps: depth-map computation, depth-map refinement, and depth-map merging. To illustrate the effect of each step, we show the depth-maps and back projected 3D points after each step for the fourth image in Fountain-P11 and the second image in Herz-Jesu-P8 in Fig. 8 and Fig. 9 respectively. Besides visual results, the numbers of correct and error pixels after each step are given in Table II based on the measurement given in Eq. 8. The results show that after depth-map computation the patch based stereo

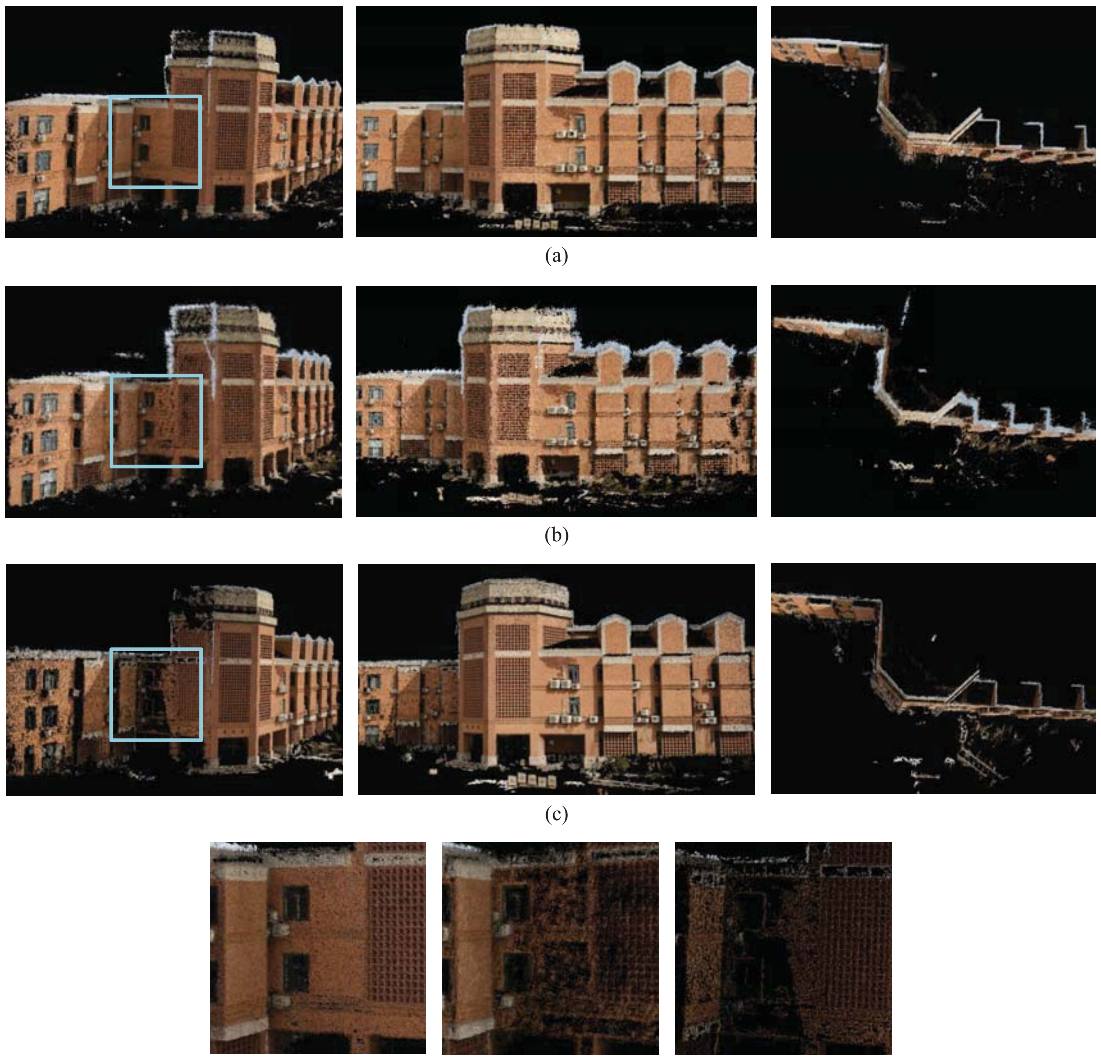


Fig. 16. Final reconstruction results (colorized point cloud rendering) of three evaluated methods on the Life Science Building data set. In (a)–(c), the results are rendered from three different view points (the right one is seen from the top view).

can generate acceptable depth-maps, but still contain certain visible errors. After depth-map refinement, 77% and 82% of the error pixels are removed in Fountain-P11 and Herz-Jesu-P8 respectively, which results in a relatively clean point cloud. Finally, after depth-map merging some holes are filled, like the left part of the fountain's base in Fig. 8.

Fig. 10 and Fig. 11 show the depth error maps for the fourth image in Fountain-P11 and the second image in Herz-Jesu-P8 using three evaluated methods respectively. In these figures, blue pixels encode missing depth values by the MVS method, green pixels encode missing ground truth data, red pixels encode an error e larger than τ_e, and pixels with errors between 0 and τ_e are encoded in gray $[255, 0]$. The results show that our method and the DAISY method can generate much more dense points than the PMVS method. Although the parameters of PMVS have been set to try to reconstruct a 3D point at every pixel, it still leaves lots of pixels without depths. Compared with the DAISY method, the proposed method could achieve more accurate results since its error maps are more *brighter* than that of the DAISY method, and *brighter* means the depth errors are smaller. Table III gives a quantitative evaluation of the results shown in Fig. 10 and 11 by counting the numbers of correct and error pixels. The results show that compared with the DAISY and PMVS methods the proposed method not only produces more correct pixels but also has lower error/correct ratios.

Besides a single image, we compute the depth errors across all images in the data sets and evaluate the overall quality of the reconstruction results generated by three evaluated methods. For each image, we count the number of correct pixels whose depth errors are below τ_e and show the results in Fig. 12. The results show that in all the images the correct pixel numbers of the proposed method and the DAISY method are almost the same, and are approximately 1.5 times larger than that of the PMVS method.

To further evaluate the reconstruction accuracy, we count the total number of correct pixels in all images as a function of the error threshold τ_e. We set τ_e to 200 values uniformity

TABLE IV
COMPUTATIONAL TIME OF THREE EVALUATED METHODS (MINUTES)

Data set	Proposed method	DAISY	PMVS
Fountain-P11 (11 images)	9.5	11.9	127
Herz-Jesu-P8 (eight images)	7.1	7.5	106

TABLE V
COMPUTATIONAL TIME OF THREE EVALUATED METHODS (MINUTES)

Data set	Proposed paper	DAISY	PMVS
Hull (61 images)	45.3	49.2	621
Life Science Building (102 images)	81	77.9	1579

distributed in the range $[0, 0.02]$, and show the results in Fig. 13. The results show that when the error threshold τ_e is quite small (below 0.002), the proposed method and the PMVS method can get more correct pixels than the DAISY method, which indicates that if we concern about the reconstruction result with high accuracy the proposed method and PMVS both outperform the DAISY method. This result comes from the natures of three evaluated methods. The proposed method and the PMVS method rely on refinement of the plane location and normal in a continuous domain, but the DAISY method matches DAISY descriptor at discrete pixel locations along the epipolar line which results in discrete depths in space, and this discrete nature will affect its accuracy.

Finally, we evaluate the speed of different methods. First we analyze the computation complexity of the proposed method. The complexities for depth-map computation, refinement, and merging are $O(HWBD)$, $O(HWk_1)$, and $O(HWk_1)$ respectively, where H and W are the height and width of the image respectively, B is the size of the matching window, D is the number of depths to be tested, and k_1 is the maximum number of neighboring images. In this paper, $B = w \times w = 49$, $k_1 = 10$. Obviously, the complexities of depth-map refinement and merging are much lower than that of the depth-map computation. For each pixel in the depth-map computation step, we compute its cost aggregation once at the beginning, followed by three iterations to refine the plane. At each iteration we compute the pixel's cost aggregation three times using its three neighbor pixel's plane parameters for spatial propagation and six times for random plane assignment. This gives the number of depths to be tested for each pixel: $D = 1 + (3 + 6) \times 3 = 28$. Compared to traditional stereo matching approaches that should test a large number of depth hypotheses, the proposed method can significantly reduce the computational complexity. The speeds of three evaluated methods are shown in Table IV. The results show that the proposed method runs at similar speed with the DAISY method, and is about 13 to 15 times faster than PMVS.

D. Qualitative Evaluation on Large Data Sets

In this section we test the proposed method on large data sets. Two data sets are used here, one is the Hull data set provided by [38] which includes 61 images with 3008×2000 (6 Megapixel) resolution, and another is the Life Science Building data set captured at Tsinghua University which includes 102 images with 3456×2304 (8 Megapixel) resolution. Some sample images of these two data sets are shown in Fig. 14.

We qualitatively evaluate three different methods on these large data sets. Some reconstruction results are shown in Fig. 15 and Fig. 16, and the computational time is shown in Table V. The results show that all the three methods could achieve acceptable reconstruction results, but the proposed method and the PMVS method perform more accurate than the DAISY method which could be seen more clearly from Fig. 16. Compared to the PMVS method, ours can get more complete results, like the walls in Fig. 15(d) and Fig. 16(d). The proposed method and the DAISY method run about 13 times faster than PMVS on the Hull and 20 times faster on the Life Science Building.

V. CONCLUSION

In this paper we propose a depth-map merging based MVS method for large-scale scenes which takes both *accuracy* and *efficiency* into account. The key of the proposed method is an efficient patch based stereo matching plus a depth-map refinement process that enforces consistency over multiple views. Compared to state-of-the-art MVS methods, the proposed method has three main advantages: 1) The reconstructed point cloud is quite accurate and dense since the patch based stereo is able to produce depth-maps with acceptable errors which can be further refined by the depth-map refinement process. 2) The proposed method is quite efficient which is about 10 to 20 times faster than the PMVS method while attaining similar accuracy. 3) It could be easily parallelized at image level, i.e., each depth-map is computed individually, which makes it suitable for large-scale scene reconstruction with high resolution images.

REFERENCES

[1] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2006, pp. 519–528.

[2] S. M. Seitz and C. R. Dyer, "Photorealistic scene reconstruction by voxel coloring," *Int. J. Comput. Vis.*, vol. 35, no. 2, pp. 151–173, Nov. 1999.

[3] G. Vogiatzis, C. Hernandez, P. H. Torr, and R. Cipolla, "Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 29, no. 12, pp. 2241–2246, Dec. 2007.

[4] S. N. Sinha, P. Mordohai, and M. Pollefeys, "Multi-view stereo via graph cuts on the dual of an adaptive tetrahedral mesh," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[5] O. Faugeras and R. Keriven, "Variational principles, surface evolution, pde's, level set methods, and the stereo problem," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 336–344, Jun. 1998.

[6] C. Hernandez and F. Schmitt, "Silhouette and stereo fusion for 3d object modeling," *Comput. Vis. Image Understand.*, vol. 96, no. 3, pp. 367–392, Dec. 2004.

[7] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons, "Towards high-resolution large-scale multi-view stereo," in *Proc. IEEE Comput. Soc. Conf. Comput Vis. Pattern Recognit.*, Jun. 2009, pp. 1430–1437.

[8] D. Cremers and K. Kolev, "Multiview stereo and silhouette consistency via convex functionals over convex domains," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 6, pp. 1161–1174, Jun. 2011.

[9] M. Lhuillier and L. Quan, "A quasi-dense approach to surface reconstruction from uncalibrated images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 27, no. 3, pp. 418–433, Mar. 2005.

[10] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz, "Multi-view stereo for community photo collections," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[11] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 8, pp. 1362–1376, Aug. 2010.

[12] T.-P. Wu, S.-K. Yeung, J. Jia, and C.-K. Tang, "Quasi-dense 3d reconstruction using tensor-based multiview stereo," in *Proc. IEEE Comput. Soc. Conf. Comput Vis. Pattern Recognit.*, Jun. 2010, pp. 1482–1489.

[13] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski, "Towards internet-scale multi-view stereo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1434–1441.

[14] M. Goesele, B. Curless, and S. M. Seitz, "Multi-view stereo revisited," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Oct. 2006, pp. 2402–2409.

[15] C. Strecha, R. Fransens, and L. V. Gool, "Combined depth and outlier estimation in multi-view stereo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Oct. 2006, pp. 2394–2401.

[16] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, and J.-M. Frahm, "Real-time visibility-based fusion of depth maps," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[17] C. Zach, T. Pock, and H. Bischof, "A globally optimal algorithm for robust tv-l^1 range image integration," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2007, pp. 1–8.

[18] D. Bradley, T. Boubekeur, and W. Heidrich, "Accurate multi-view reconstruction using robust binocular stereo and surface meshing," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[19] N. D. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla, "Using multiple hypotheses to improve depth-maps for multi-view stereo," in *Proc. Eur. Conf. Comput. Vis.*, Oct. 2008, pp. 766–779.

[20] Y. Liu, X. Cao, Q. Dai, and W. Xu, "Continuous depth estimation for multi-view stereo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 2121–2128.

[21] J. Li, E. Li, Y. Chen, L. Xu, and Y. Zhang, "Bundled depth-map merging for multi-view stereo," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Aug. 2010, pp. 2769–2776.

[22] E. Tola, C. Strecha, and P. Fua, "Efficient large-scale multi-view stereo for ultra high-resolution image sets," *Mach. Vis. Appl.*, vol. 23, no. 5, pp. 903–920, 2012.

[23] D. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[24] E. Tola, V. Lepetit, and P. Fua, "Daisy: An efficient dense descriptor applied to wide-baseline stereo," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 5, pp. 815–830, May 2010.

[25] D. Gallup, J.-M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2007, pp. 1–8.

[26] M. Pollefeys, D. Nister, J.-M. Frahm, A. Akbarzadeh, P. Mordohai, B. Clipp, C. Engels, D. Gallup, S. J. Kim, P. Merrell, C. Salmi, S. N. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, and H. Towles, "Detailed real-time urban 3D reconstruction from video," *Int. J. Comput. Vis.*, vol. 72, no. 2, pp. 143–167, 2008.

[27] G. Zhang, J. Jia, T.-T. Wong, and H. Bao, "Consistent depth maps recovery from a video sequence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 6, pp. 974–988, Jun. 2009.

[28] D. Gallup, J.-M. Frahm, and M. Pollefeys, "Piecewise planar and non-planar stereo for urban scene reconstruction," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 1418–1425.

[29] N. Snavely, S. M. Seitz, and R. Szeliski, "Modeling the world from internet photo collections," *Int. J. Comput. Vis.*, vol. 80, no. 2, pp. 189–210, 2008.

[30] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building rome in a day," in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep. 2009, pp. 72–79.

[31] J.-M. Frahm, P. George, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys, "Building rome on a cloudless day," in *Proc. Eur. Conf. Comput Vis.*, Sep. 2010, pp. 368–381.

[32] M. Bleyer, C. Rhemann, and C. Rother, "Patchmatch stereo—stereo matching with slanted support windows," in *Proc. Brit. Mach. Vis. Conf.*, Aug.–Sep. 2011, pp. 14.1–14.11.

[33] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[34] K.-J. Yoon and I.-S. Kweon, "Locally adaptive support-weight approach for visual correspondence search," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2005, pp. 924–931.

[35] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann, "Local stereo matching using geodesic support weights," in *Proc. IEEE Int. Conf. Image Process.*, Nov. 2009, pp. 2093–2096.

[36] Q. Yang, L. Wang, R. Yang, H. Stewenius, and D. Nister, "Stereo matching with color-weighted correlation, hierarchical belief propagation, and occlusion handling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 3, pp. 492–504, Mar. 2009.

[37] N. Atzpadin, P. Kauff, and O. Schreer, "Stereo Anal. by hybrid recursive matching for real-time immersive video conferencing," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 3, pp. 321–334, Mar. 2004.

[38] *Patch-Based Multi-View Stereo Software (PMVS—Version 2)*. (2010) [Online]. Available: http://grail.cs.washington.edu/software/pmvs/

[39] C. Strecha, W. von Hansen, L. V. Gool, P. Fua, and U. Thoennessen, "On benchmarking camera calibration and multi-view stereo for high resolution imagery," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.

[40] *Multi-View Evaluation*. (2008) [Online]. Available: http://cvlab.epfl.ch/~strecha/multiview/denseMVS.html

[41] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proc. 4th Eurograph. Symp. Geometry Process.*, Jul. 2006, pp. 61–70.

Shuhan Shen received the B.S. and M.S. degrees from Southwest Jiao Tong University, Chengdu, China, in 2003 and 2006, respectively, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2010.

He is currently an Assistant Professor with the National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His current research interests include 3D reconstruction and image-based modeling.