Efficient Multiview Stereo by Random-Search and Propagation

Youngjung Uh
Yonsei University
Seoul, Korea
youngjung.uh@yonsei.ac.kr

Yasuyuki Matsushita
Microsoft Research Asia
Beijing, China
yasumat@microsoft.com

Hyeran Byun
Yonsei University
Seoul, Korea
hrbyun@yonsei.ac.kr

Abstract

We present an efficient multi-view 3D reconstruction method based on randomization and propagation scheme. Our method progressively refines 3D point estimates by randomly perturbing the initial guess of 3D points and propagates photo-consistent ones to their neighbors. In contrast to previous refinement methods that perform local optimization for a better photo-consistency, our randomization approach takes lucky *matchings for reducing the computational complexity. Experiments show favorable efficiency of the proposed method with the accuracy that is close to the state-of-the-art methods.*

1. Introduction

Multi-view stereo (MVS) reconstructs dense 3D points from a set of calibrated images by estimating correspondences across the images. Much progress has been made on improving the quality of 3D reconstruction, and it is getting close to that of laser scans in recent approaches [5, 11]. However, the computational complexity becomes one of the major limiting factors for MVS methods when a large amount of data needs to be processed. Improving the computational efficiency of MVS is actively investigated by several recent works [7, 15, 27, 33].

The major reason why a conventional MVS demands a high computational cost is that, for each hypothesized 3D point, its photo-consistency needs to be evaluated using multiple views that see the 3D point. The number of the operations is proportional to the amount of hypothesized 3D points and the number of views. Thus, reducing the number of photo-consistency assessment is one of the keys to greater efficiency.

In this paper, we present an efficient MVS method based on a random-search and propagation scheme. The random-search and propagation approach is shown useful in the recent PatchMatch method [2], also in its application to binocular-stereo [4]. The heart of PatchMatch method is the observation that random sampling can sometimes find good matches, and that propagating the sparse good matches can efficiently overwrite nearby bad matches thanks to the local coherence of an image. The use of this scheme in the MVS context has a few advantages. First, it can significantly reduce the number of photo-consistency evaluations by the random-search, unlike point-wise local optimization approaches. Second, local smoothness is implicitly enforced by the propagation in an efficient manner. As a result, the computational demands can be significantly reduced by our method in comparison with recent efficient MVS approaches (*e.g.*, $3\times$ faster than [11]). Finally, our approach is robust against errors in the initial guess; in fact, our method is able to begin with any random guess of the 3D points.

While the random-search and propagation scheme has these advantages, it is not straightforward to apply it in the MVS context where a 3D point is reconstructed via multi-view photo-consistency. Shen [27] utilizes such a scheme to speed up depth-map estimation on each pair of images and merges the resulting point cloud by checking occlusions with neglecting the multi-view photo-consistency, which is found often important in previous studies [8, 33]. Unlike binocular stereo matching where the PatchMatch scheme has been successfully applied [3, 4, 14], our case requires to handle significant view-point variations in both angle and distance and resolution gaps caused by them, and inter-view propagation needs to be carefully designed for efficient and effective 3D reconstruction. This paper presents an approach to address these issues and develops an efficient MVS method. The proposed method is evaluated using Middlebury multiview stereo benchmark and a few different outdoor multi-view images. Our experiment shows the effectiveness of the proposed method in comparison with previous state-of-the-art methods in efficiency and accuracy.

2. Related work

MVS has been extensively studied in the literature and compared using comprehensive benchmark datasets [25, 30]. This section reviews some of the works that are closely related to our work.

Efficiency of MVS has been pursued in a few different approaches. One class of methods casts the MVS problem into a combination of local (typically, pairwise) stereo matching and merging processes. The merging process, often referred to as depth-map merging, has been approached using various different ways. Li *et al.* [18] efficiently merge depth-maps in a bundle optimization scheme by ensuring depth consistency across multiple views. Bradley *et al.* [5] merge depth-maps into a point cloud by estimating local surface normal and performing fast meshing in a low dimensional space. Merrell *et al.* [19] present a method for real-time visibility-based fusion of depth-maps by reducing number of depth hypothesis.

Another class of approaches performs clustering of input images such that highly redundant computations can be avoided [10] or independent parts can be handled in parallel [9, 13, 35]. Furukawa *et al.* [10] perform clustering using three cues; compactness, size, and coverage. By introducing a function that measures the expected reconstruction accuracy, they minimize the total number of images from the clusters subject to the function with pre-defined maximum number of images per cluster. Consequently, they reduced computational cost by half for St. Peter's Basilica dataset[1]. 3D reconstruction using a massive amount of community photo collections has been shown by Goesele *et al.* [13]. Their method selects a small set of views among candidate views for matching in order to speed up the depth computation, by a criterion designed to prefer views that are photometrically consistent and provide a sufficiently wide range of observation. Frahm *et al.* [9] propose appearance-based clustering by extracting global appearance descriptor *gist* [21] for each image and verifying the clusters via epipolar geometric consistency. They improve performance by leveraging the constraints from appearance clustering and location independence, and by performing computation in a parallel manner at the cluster level.

Parallelism is indeed an important aspect for an efficient MVS method. Some of the computation blocks, such as image re-projection and visibility checking, are naturally parallelized as shown in [15, 22]. Li *et al.* [18] use a *track*, which refers to a set of pixel matches, for performing optimization at the track-level in parallel. Agarwal *et al.* [1] show city-level reconstructions in a day by designing parallel image matching pipeline. Furukawa *et al.* [10] propose a merging algorithm that is designed to run on individual MVS points in a parallel manner.

Recent progress in GPU parallelization has made real-time 3D reconstruction possible. Newcombe and Davison [20] present a rapid and dense reconstruction method of scenes browsed by a live camera using a base mesh and its warped depth maps. Stühmer *et al.* [31] show the reconstruction of a scene with small displacements from nearly-static handheld camera. Furthermore, a live 3D reconstruction on mobile phones has been introduced by Tanskanen *et al.* [32], where the embedded inertial sensor is used for determining camera poses. However, these methods are limited to low-resolution or short-baseline settings, which make them efficient and tractable but inapplicable to more general cases.

[1]http://grail.cs.washington.edu/rome/rome/index_5.html

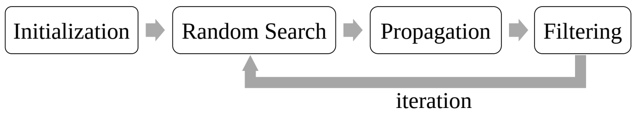


Figure 1: Flow chart of the proposed method. Latter three components are iterated. See Sec. 3 for more details.

In contrast to these approaches, we aim at reducing the computational cost by neglecting costly yet small-gain computations using the random-search and propagation scheme, motivated by the recent success of PatchMatch technique [2] and its applications. Bleyer *et al.* [4] apply the PatchMatch method to binocular stereo matching with slanted support windows and have shown accurate estimation results. It is further improved by combination with a belief propagation-based solution method by Besse *et al.* [3]. Furthermore, Heise *et al.* [14] impose regularization with the Huber norm to achieve the state-of-the-art accuracy in Middlebury stereo benchmark [24] for 0.5-pixel accuracy.

3. Proposed method

Our method is built upon random-search and propagation scheme and consists of four steps: (1) initialization, (2) random-search, (3) propagation, and (4) filtering as illustrated in Figure 1. We use a small 3D patch that is defined for a 3D point. The patch is a rectangular plane that is defined by its center position (corresponding to the 3D point), normal direction, and the number of grid points and size. In the initialization step, patches are first assigned random positions and normals. From those randomized patches, a set of best patch candidates is then selected by evaluating each patch's photo-consistency across all the views, and the rest of the patches are removed. In the random-search step, the original patches are perturbed and assigned new positions and normals. Patches are then propagated to their neighbors in the 3D coordinates, regulated by the input image coordinates. Finally, the filtering step eliminates outliers. The steps from the random-search to the filtering are iterated until convergence. In what follows, we describe our 3D patch model and explain the details of each step.

3.1. Patch model and photo-consistency

Our patch is a local 3D plane defined in the world coordinates system. It is represented as $p(\mathbf{c}, \mathbf{n}) \in \mathcal{P}$, where $\mathbf{c} \in \mathbb{R}^3$ is the 3D position of the patch center, $\mathbf{n} \in \mathbb{R}^3$ is the normal direction. $\mu \times \mu$ grid points are defined on the patch plane, and the interval between each grid point is adaptively determined using the rays spanned at the pixel interval from the closest image. The grid points are projected onto the image coordinates, and corresponding pixel intensities are sampled to create an observation vector for each view.

Photo-consistency score of patch p is calculated as follows. We collect a set of visible images $\mathcal{V}(p)$ for patch p by evaluating visibility using camera's field of view and the angle between patch's surface normal and the ray direction from the camera. From $\mathcal{V}(p)$, we select the closest view as the reference image $R(p)$ by evaluating the projected areas of the patch with a unit area. Once the reference image $R(p)$ is determined, the 3D patch grid is created as described above, and an observation vector $\mathbf{f} \in \mathbb{R}^{\mu^2}$ is generated for each image in $\mathcal{V}(p)$. Using the observation vectors, we define the photo-consistency cost $E(p)$ as the average of one minus normalized cross-correlation (NCC) scores between the reference and other observation vectors, $\mathbf{f}_r$ and $\mathbf{f}_i$, and

$$E(p) = \frac{1}{|\mathcal{V}(p) \setminus \mathcal{R}(p)|} \sum_{i \in \mathcal{V}(p) \setminus \mathcal{R}(p)} 1 - \left\langle \frac{\widehat{\mathbf{f}}_r}{\|\widehat{\mathbf{f}}_r\|}, \frac{\widehat{\mathbf{f}}_i}{\|\widehat{\mathbf{f}}_i\|} \right\rangle, \quad (1)$$

where $\langle \cdot, \cdot \rangle$ represents the inner product of vectors, $\widehat{\mathbf{f}} = \mathbf{f} - \overline{\mathbf{f}}$ and $\overline{\mathbf{f}}$ represents average of the observations in the vector. In this manner, the greater photo-consistency results in the lower cost $E(p)$. To increase the robustness against occluded views, we remove the NCC scores from the above calculation that are smaller than the pre-defined threshold α. In addition, to avoid the aperture problem, we set $E(p) = \infty$ when $\mathbf{f}_r$ is textureless, *i.e.*, the magnitude of the reference vector $\widehat{\mathbf{f}}_r$ is close to zero.

In our method, we use a view-dependent candidate patch map $B_i \in \mathbb{P}^{W \times H}$, which has the same dimensions (W, H) as the image of camera i. It retains the visible patch that has the highest photo-consistency, *i.e.*, smallest $E(p)$, for each ray from each camera i. The view-dependent candidate patch map B is used for achieving computational efficiency, by working in a set of image coordinates, rather than directly in the world 3D coordinates. A similar concept has been used in PMVS [11], which stores all the patches that the ray penetrates, but our candidate patch map B stores only the best one for each ray (or none for the ray that does not intersect any patch). Our method updates the candidate patch maps B whenever patches p are updated. We use a vector $\mathbf{x} = (x, y) \in \mathbb{Z}_+^2$ for indicating the coordinates of the candidate patch map B_i, and use $B_i(\mathbf{x})$ for indicating the corresponding patch p that is stored in the candidate patch map B_i at location $\mathbf{x}$.

3.2. Initialization

In the initialization step, a set of 3D points are randomly generated, and for each 3D point, a 3D patch p is assigned.

Algorithm 1 random-search

```
Input: Original patch $p^{(t)}$, camera $i$, ray $\mathbf{r}$ through target
    pixel, range parameters $r_c, \phi, \theta, \psi$, threshold $\epsilon$
Output: New patch $p^{(t+1)}$
 1: while $(r_c > \epsilon)$ do
 2:   Pick random $\Delta_d \in [-r_c, r_c]$
 3:   Pick random $(\Delta_\phi, \Delta_\theta, \Delta_\psi)^T \in [-(\phi,\theta,\psi)^T, (\phi,\theta,\psi)^T]$
 4:   $\mathbf{R} := R_x(\Delta_\phi) R_y(\Delta_\theta) R_z(\Delta_\psi)$
 5:   $\mathbf{c}_{p'} := \mathbf{c}_p + \Delta_d \mathbf{r}$
 6:   $\mathbf{n}_{p'} := \mathbf{R}\mathbf{n}_p$
 7:   if $E(p') < E(p^{(t)})$ using Eq. (1) then
 8:     $p^{(t+1)} := p'$
 9:   end if
10:   $(r_c, \phi, \theta, \psi) := (r_c, \phi, \theta, \psi)/2$
11: end while
```

Algorithm 2 spatial propagation

```
Input: Candidate patch maps $\{B_i^{(t)}\}$, cameras $\{i\}$, propa-
    gation direction $w \in \{-1, 1\}$
Output: Updated candidate patch maps $\{B_i^{(t+1)}\}$
 1: for each image $I_i$ do
 2:   $\mathbf{c}_i$ := position of camera $i$
 3:   for each pixel location $\mathbf{x}(= (x, y))$ do
 4:     $\mathbf{X}' := \{(x + w, y), (x, y + w)\}$
 5:     for each $\mathbf{x}' \in \mathbf{X}'$ do
 6:       $\mathbf{b}' := B_i^{(t)}(\mathbf{x}')$
 7:       $\mathbf{r}' = (\mathbf{b}' - \mathbf{c}_i)/||\mathbf{b}' - \mathbf{c}_i||$
 8:       $d := (\mathbf{c}_{b'} - \mathbf{c}_i)^T \mathbf{n}_{b'} / (\mathbf{r}'^T \mathbf{n}_{b'})$
 9:       $\mathbf{c}_{p'} := \mathbf{c}_i + d\mathbf{r}'$
10:       $\mathbf{n}_{p'} := \mathbf{n}_{b'}$
11:       if $E(p') < E(B_i^{(t)}(\mathbf{x}'))$ then
12:         $B_i^{(t+1)}(\mathbf{x}') := p'$
13:       end if
14:     end for
15:   end for
16: end for
```

The patches p are further assigned random orientations $\mathbf{n}$. In case no prior information is given about the scene, our method begins with randomly producing 3D points in a bounding box or intersection of multiview frustum. If some knowledge about the scene is available, *e.g.*, sparse correspondences obtained in the structure-from-motion process [29, 34], our method takes them as input and generates additional random 3D points around the original ones.

3.3. Random-search

Our method uses a random-search approach for generating more diverse patches around the previous candidates as shown in Figure 2a. Unlike previous approaches that com-

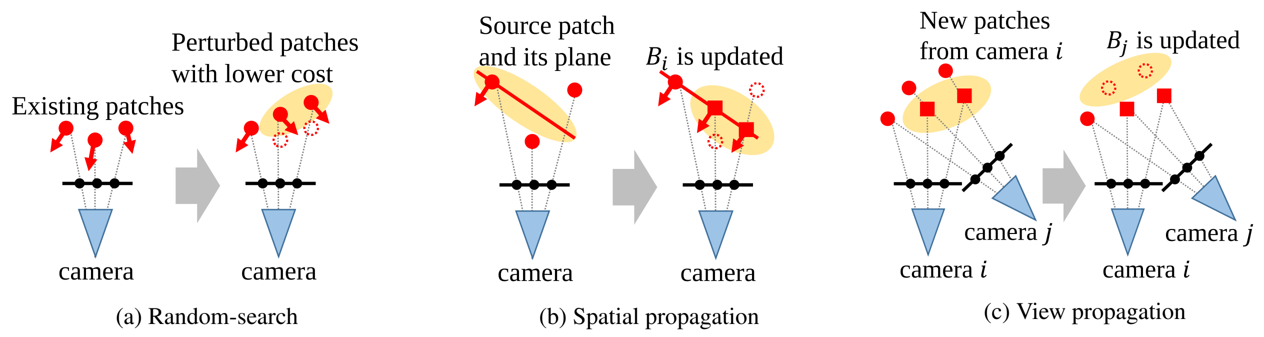


Figure 2: Illustration of essential steps. (a) An existing patch is replaced with a better one that is randomly produced by perturbation. (b) From an existing patch, new patches are generated from left to right on its extended plane. (c) The newly propagated patches are further propagated to another view j to update B_j.

putes local optima using gradient descent, our method attempts to efficiently find a sub-optimal solution in a randomization scheme. In addition, because of the randomization, the method has a chance of escaping from local minima, which is potentially non-optimal.

We use the candidate patch maps B for generating new patches. For each view i, we pick a patch p stored in $B_i^{(t)}(\mathbf{x})$, where $^{(t)}$ indicates the result of the t-th iteration. A new patch p' is created by perturbing p's position and normal. The perturbation of the position is performed along the ray of pixel $\mathbf{x}$ in camera i, whose displacement $\Delta_d \in \mathbb{R}$ is randomly chosen within the perturbation range $[-r_c, r_c]$. With this displacement, the new patch's position $\mathbf{c}_{p'}$ becomes $\mathbf{c}_{p'} = \mathbf{c}_p + \Delta_d \mathbf{r}$. The surface normal is perturbed using a rotation matrix $\mathbf{R} \in SO(3)$ that is randomly defined by a predefined range of the Euler angles $\left[-(\phi,\theta,\psi)^T, (\phi,\theta,\psi)^T\right]$. The new patch's surface normal $\mathbf{n}_{p'}$ is thus defined as $\mathbf{n}_{p'} = \mathbf{R}\mathbf{n}_p$. The photo-consistency of the new patch p' is then evaluated, and if it is better than that of the original patch p, B_i is updated and stores p'. The procedure is summarized in Algorithm 1.

3.4. Propagation

This step propagates patches to its neighbors using the candidate patch maps B. This operation is similar to the original PatchMatch method [2]; however, to apply the procedure in the context of MVS, we take a two-step approach for the propagation. Namely, the patches are first propagated to their neighbors in each candidate patch map B_i, and further propagated across different views i.

The spatial propagation is performed on each candidate patch map B_i, starting from the top-left to bottom-right corner in odd iterations, and then the backward direction in even iterations, as done in [2]. As shown in Figure 2b, a source patch p of $B_i^{(t)}(\mathbf{x})$ is propagated to its neighboring target coordinates $\mathbf{x}'$, and a new patch p' is defined. The new patch's location $\mathbf{c}_{p'}$ is determined by the intersection of patch p and the camera ray through the target coordinates $\mathbf{x}'$. The surface normal of the new patch p' is maintained the same as the source patch p. Subsequently, photo-consistency of the new patch $E(p')$ is compared to that of the patch that has been defined at the same coordinates, *i.e.*, $E(B_i^{(t)}(\mathbf{x}'))$. Then, the candidate patch map B_i is updated to contain p' at $B_i^{(t+1)}(\mathbf{x}')$ and the patch p is removed if $E(p')$ shows the better photo-consistency (lower score of E). To avoid unreliable propagations, the propagation is terminated if the angle between the patch normal $\mathbf{n}_p$ and the ray direction $\mathbf{r}$ is greater than a pre-defined threshold θ_p. The procedure is summarized in Algorithm 2.

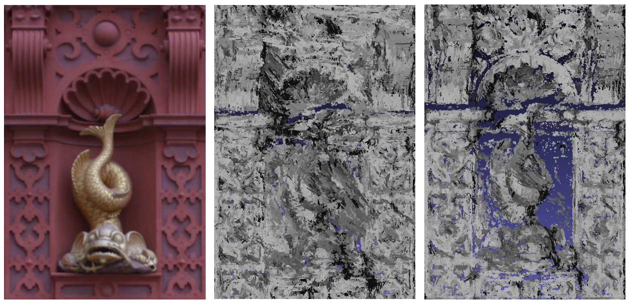


Figure 3: Effect of filtering. Left: observation, middle: before filtering, right: after filtering. More details become visible after the filtering.

Once the spatial propagation is performed for each B_i, inter-view propagation takes place to further propagate patches to B_j as shown in Figure 2c. Every new patch p' retained in B_i^{t+1} is projected onto all the other cameras j to locate their target coordinates $\mathbf{u}$. If the new patch p' has a better photo-consistency than the patch of $B_j(\mathbf{u})$, the patch is replaced with the new one at $B_j(\mathbf{u})$ where $\mathbf{u}$ is projection of the patch p'. At the same time, the patch that is overwritten at $B_j(\mathbf{u})$ is removed from all the other candidate patch maps.

3.5. Filtering

To remove the artifacts caused by wrong propagations, we use two filtering methods after the propagation step. The first filtering is the one used in Furukawa and Ponce's method [11] for discarding visibility-inconsistent patches and neighbor-inconsistent patches. In addition, we use smoothness-based trimming as the second filtering. Smoothness ζ_p of a patch p is defined by the angle between its normal $\mathbf{n}_p$ and the average normal $\bar{\mathbf{n}}_p$ in the fixed-size window W_p as $\zeta_p = \arccos(\mathbf{n}_p^T \bar{\mathbf{n}}_p)$. We also simply discard patches that show greater angles than a pre-defined threshold θ_ζ. More details become visible after these two filtering steps as in Figure 3.

4. Experimental Results

In this section, we assess the performance of our method using public datasets provided by Seitz *et al.* [26] and Strecha [30], and our own street side dataset. The accuracy of our method is evaluated using the Middlebury multi-view stereo evaluation score and its relative error to the ground truth. The measurement of computational cost for Middlebury dataset is performed using the *normalized time* in a certain fixed environment provided by Middlebury evaluation system. Except for Middlebury dataset, our experiments including comparison with PMVS are performed on an Intel Xeon CPU E5-2690 2.90GHz (32 cores) with PPL[2] for parallelization, where we set the number of iterations of random-search, propagation, and filtering to 4.

Throughout the experiments, we use the following parameters. The resolution of the grid μ (in Sec. 3.1) is fixed to 7, the threshold α for the energy $E(p)$ of Eq. (1) is set to 0.7. The maximum angle θ_p (in Sec. 3.4) between $\mathbf{n}_p$ and the ray $\mathbf{r}$ from the camera is set to 60 degrees. The smoothness threshold for filtering θ_ζ in Sec. 3.5 is set to $\arccos 0.85$. Perturbation ranges of surface normal are set to $\phi = \theta = \psi := \pi$. Because the size of the target objects varies (inferred by camera parameters), we change the parameters of the perturbation range r_c of position and threshold ϵ depending on the dataset: $(r_c, \epsilon) = (0.01, 0.001)$ for the Middlebury dataset, $(0.1, 0.03)$ for Fountain-P11, and $(0.1, 0.03)$ for Herzjesu.

Quantitative evaluation procedures for MVS methods are provided in the Middlebury benchmark [25, 26]. First, we show the benchmark for accuracy and completeness in Table 1 in comparison with other state-of-the-art methods

[2]Parallel Patterns Library(PPL), http://msdn.microsoft.com/en-us/library/dd492418.aspx

Dataset	nIter	1	2	3	4
templeSR 16 images	GenPatch	0:11	0:21	0:33	0:45
	nPatch	55K	106K	194K	232K
	PSR	0:59	1:11	1:20	1:21
	nTriangle	0.8M	0.9M	1.6M	1.7M
templeR 47 images	GenPatch	1:39	3:07	4:37	6:10
	nPatch	119K	251K	476K	549K
	PSR	1:11	1:44	2:22	2:12
	nTriangle	1.3M	2.0M	3.7M	4.0M
dinoSR 16 images	GenPatch	0:15	0:32	0:49	1:07
	nPatch	99K	190K	337K	387K
	PSR	1:19	1:22	1:59	1:57
	nTriangle	1.1M	1.5M	2.6M	2.8M
dinoR 48 images	GenPatch	2:33	5:01	7:25	9:57
	nPatch	176K	380K	734K	837K
	PSR	1:22	1:57	3:00	3:39
	nTriangle	1.8M	2.9M	5.2M	5.6M

Table 2: Running time (min:sec) with respect to the varying number of patches and triangles over iterations. (GenPatch:generating patches, nP:number of patches, PSR:Poisson surface reconstruction, nT:number of triangles)

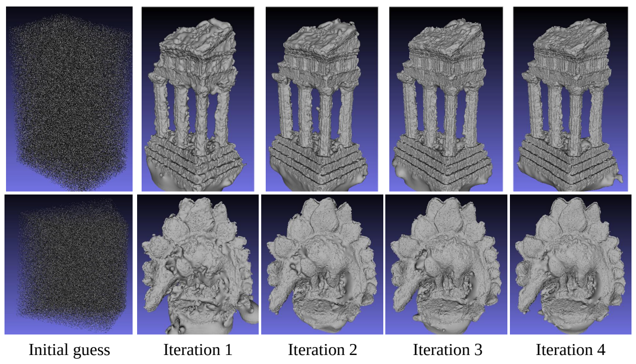
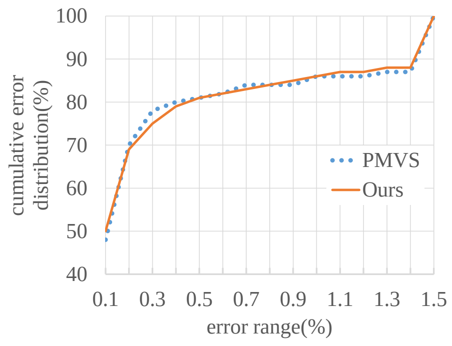


Figure 4: Our reconstruction results over iterations on the Middlebury dataset. From totally random initial guess in a bounding box, our method iteratively refine the 3D shape. Meshes are generated with Poisson surface reconstruction [16]. Top: TempleRing data, bottom: DinoRing data.

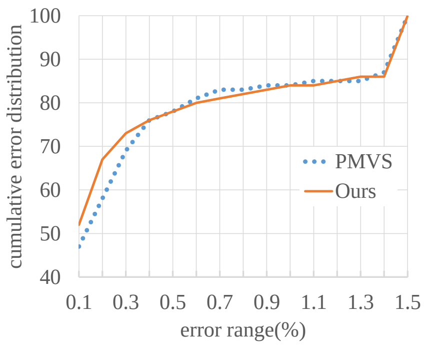
along with the *normalized* computation time. It shows the efficiency of the proposed method without a significant loss of accuracy and completeness. Table 2 shows component-wise running time with respect to the varying number of patches and triangles over iterations. Figure 4 shows the evolution of our 3D reconstruction over iterations. Our method faithfully recovers 3D shape in several iterations, even with starting from the totally random initial guess. For the TempleRing and DinoRing datasets of Middlebury [25], our method runs in a couple of minutes for patch optimization followed by a couple of minutes of Poisson surface re-

		Temple						Dino					
		Ring			SparseRing			Ring			SparseRing		
Method	Processor	Acc	Comp	Time	Acc	Comp	Time	Acc	Comp	Time	Acc	Comp	Time
Ours	CPU	0.51	96.4	7.4	1.23	90.2	**1.9**	0.32	97.3	12.1	0.42	96.7	**2.7**
Furukawa3 [11]	CPU	0.47	99.6	211	0.63	**99.3**	129	**0.28**	**99.8**	301	**0.37**	**99.2**	152
Hiep [15]	GPU	**0.45**	**99.8**	**1.5**	-	-	-	0.53	99.7	**1.5**	-	-	-
Bradley [5]	CPU	0.57	98.1	11.4	**0.48**	93.7	3.5	0.39	97.6	23.5	0.38	94.7	7
Campbell [6]	CPU	0.48	99.4	59	0.53	98.6	22.5	-	-	-	-	-	-
Goesele [12]	CPU	0.61	86.2	2040	0.87	56.6	687	0.46	57.8	2516	0.56	26.0	843
Li [18]	CPU	0.64	98.2	3.6	-	-	-	0.43	99.7	5.9	-	-	
Kolev3 [17]	CPU	0.7	98.3	539	0.97	92.7	114	0.42	99.5	470	0.48	98.6	100

Table 1: Quantitative evaluations using the Middlebury dataset [25] with default thresholds(Acc:90%, Comp:1.25mm). Each column for each dataset shows accuracy in mm, (Acc: the lower the better), completeness in % (Comp: the higher the better), and normalized running time in minutes (Time).



(a) Fountain-P11



(b) Herzjesu-P8

Figure 5: Quantitative evaluation of our method compared to PMVS [11] on two data sets of [30]. The last(15^{th}) bin collects the pixels with error ratio greater than 1.5%.

construction [16] that generates millions of triangles. Our CPU implementation is slower than the GPU implementation of Hiep *et al*. [15] with a GeForce 8800 GTX that has 128 stream processors. While we cannot directly compare the performance with their method due to the difference of CPU and GPU implementations, our method shows performance improvement over other CPU implementations.

We also use Strecha's MVS dataset [30], Fountain-P11 and Herzjesu-P8, for evaluating the performance of our method. Figure 5 shows the cumulative histogram of error ratio in depth to the ground truth, *i.e.*, $e = (d_r - d_g)/d_g$, where d_r is the depth estimate and d_g is the ground truth. It shows our result has similar accuracy with PMVS. The qualitative results are also shown in Figure 6. They show good overall reconstruction of the scenes that is close to PMVS [11], while the level of reconstruction details is varying due to the fact that our propagation assumes local smoothness as shown in the close-up views.

We also use a more life-like street side scene dataset that we recorded for evaluating our method. The dataset is recorded in a rather uncontrolled setting, and a structure-from-motion technique [28] has been used for recovering camera poses. In Figure 7, we show the subset of input images and the reconstruction result in comparison with PMVS [11], performed via VisualSFM [34]. The experiments are performed with the consistent parameters for the both methods, *e.g.*, minimum number of supporting images$=$ 2, threshold for one minus NCC score$=$ 0.7. Computation time on this dataset is 26 and 83 minutes for our method and PMVS, respectively. Our result shows comparable overall reconstruction quality to PMVS's result. While the details are better recovered by PMVS, our method recovers larger areas (*e.g.*, left most building) with denser points from the same set of input images and parameters. The larger coverage area of our method comes from the difference in the initialization step. While PMVS limits the points to be generated from feature matching across images, our method initializes the points by random search over the candidate area.

5. Discussions

By taking the random-search and propagation approach, we have developed an efficient MVS method. The result shows that the proposed method achieves high efficiency with accuracy that is close to the state-of-the-art methods.

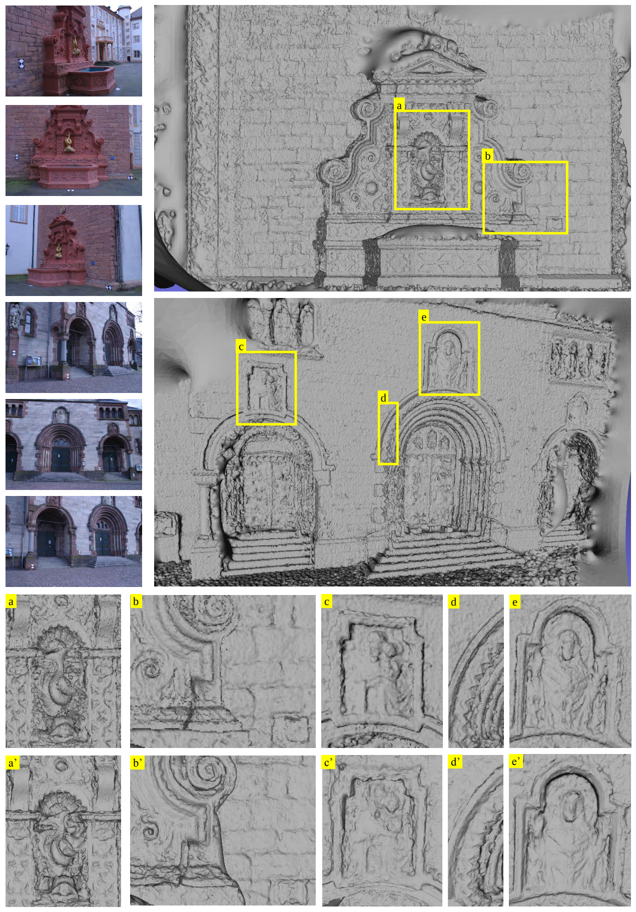


Figure 6: Result of our method on two data sets of [30]; Fountain-P11 (11 images) and Herzjesu-P8 (8 images). A subset of the input images are shown on the top left corner, reconstruction results are shown on the top right corner, their close-up views (a – e), and PMVS [11] results of corresponding regions (a' – e') are shown, respectively, at the bottom.

In situations where a large amount of data needs to be processed, we believe that our method has a particular strength. Our method is indeed highly pararllelizable: The random-search step can be implemented in a parallel manner for each 3D point, the spatial propagation step can use the jump flood scheme [23], and view-propagation step can also run in a parallel manner for each image. This motivates us for our future work of GPU implementation of additional parallelizable parts including NCC computation and image re-projection.

One of the drawbacks of our method is that its running time depends on the number of candidate patch maps (equivalent to the number of input images). This issue could be addressed by incorporating image selection techniques such as [9, 10]. Another limitation is that our propagation step relies on local smoothness of the scene, therefore, small

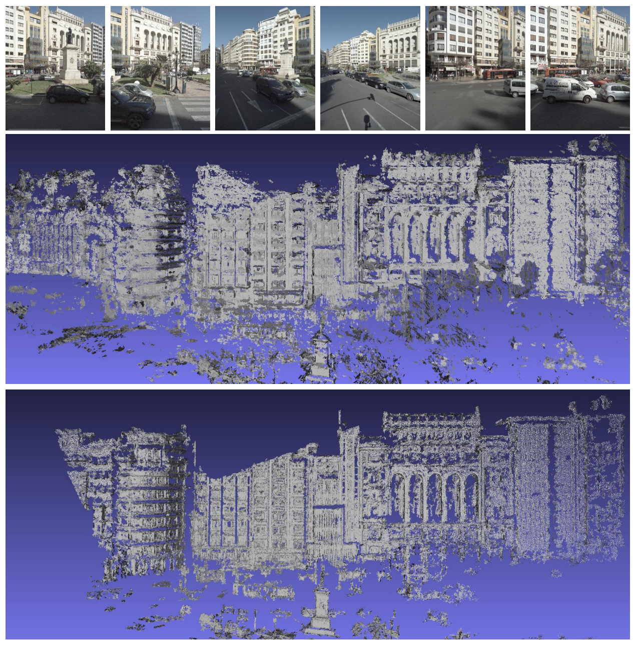


Figure 7: Street scene reconstruction result from 30 images in resolution of 6M. The first row shows some of the input images. The second row shows the point cloud recovered by our method, and the bottom one shows PMVS [11] result.

objects or high-frequency details in a small region are difficult to be reconstructed in case their corresponding image areas are small. Our future work includes further improving the accuracy of these regions by adaptively increasing the number of random-search operations.

Acknowledgements

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government(MEST) (No. NRF-2013R1A2A1A01015870).

Part of this work has been done while the first author is visiting MSRA as a research intern.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building rome in a day. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, pages 72–79. IEEE, 2009. 2

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. on Graph.*, 28(3):24, 2009. 1, 2, 4

[3] F. Besse, C. Rother, A. W. Fitzgibbon, and J. Kautz. Pmbp: Patchmatch belief propagation for correspondence field estimation. In *Proc. of British Machine Vision Conf. (BMVC)*, 2012. 1, 2

[4] M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo - stereo matching with slanted support windows. In *Proc. of British Machine Vision Conf. (BMVC)*, 2011. 1, 2

[5] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. 1, 2, 6

[6] N. D. Campbell, G. Vogiatzis, C. Hernández, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *Proc. of European Conf. on Computer Vision (ECCV)*, pages 766–779, 2008. 6

[7] J. Y. Chang, H. Park, I. K. Park, K. M. Lee, and S. U. Lee. Gpu-friendly multi-view stereo reconstruction using surfel representation and graph cuts. *Computer Vision and Image Understanding*, 115(5):620–634, 2011. 1

[8] Y. Deng, Y. Liu, Q. Dai, Z. Zhang, and Y. Wang. Noisy depth maps fusion for multiview stereo via matrix completion. *Selected Topics in Signal Processing, IEEE Journal of*, 6(5):566–582, 2012. 1

[9] J.-M. Frahm, P. Fite-Georgel, 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. of European Conf. on Computer Vision (ECCV)*, pages 368–381, 2010. 2, 7

[10] Y. Furukawa, B. Curless, S. M. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1434–1441, 2010. 2, 7

[11] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(8):1362–1376, 2010. 1, 3, 5, 6, 7

[12] M. Goesele, B. Curless, and S. M. Seitz. Multi-view stereo revisited. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 2402–2409, 2006. 6

[13] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. M. Seitz. Multi-view stereo for community photo collections. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, 2007. 2

[14] P. Heise, S. Klose, B. Jensen, and A. Knoll. Pm-huber: Patchmatch with huber regularization for stereo matching. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, 2013. 1, 2

[15] V. H. Hiep, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1430–1437, 2009. 1, 2, 6

[16] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Proc. of Sympdium on Geometry Processing*, 2006. 5, 6

[17] K. Kolev, T. Pock, and D. Cremers. Anisotropic minimal surfaces integrating photoconsistency and normal information for multiview stereo. In *Proc. of European Conf. on Computer Vision (ECCV)*, pages 538–551. Springer, 2010. 6

[18] J. Li, E. Li, Y. Chen, L. Xu, and Y. Zhang. Bundled depth-map merging for multi-view stereo. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2769–2776, 2010. 2, 6

[19] P. Merrell, A. Akbarzadeh, L. Wang, P. Mordohai, J.-M. Frahm, R. Yang, D. Nistér, and M. Pollefeys. Real-time visibility-based fusion of depth maps. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*, 2007. 2

[20] R. A. Newcombe and A. J. Davison. Live dense reconstruction with a single moving camera. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1498–1505. IEEE, 2010. 2

[21] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *Int'l Journal of Computer Vision*, 42(3):145–175, 2001. 2

[22] J.-P. Pons, R. Keriven, and O. Faugeras. Modelling dynamic scenes by registering multi-view image sequences. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 822–827, 2005. 2

[23] G. Rong and T.-S. Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proc. of Symposium on Interactive 3D Graphics and Games*, pages 109–116, 2006. 7

[24] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *Int'l Journal of Computer Vision*, 47:742, 2002. 2

[25] S. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. Middlebury stereo vision webpage. `http://www.middlebury.edu/stereo`. 1, 5, 6

[26] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 519–528, 2006. 5

[27] S. Shen. Accurate multiple view 3d reconstruction using patch-based stereo for large-scale scenes. *IEEE transactions on image processing*, 22(5):1901–1914, 2013. 1

[28] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM Trans. on Graph.*, volume 25, pages 835–846, 2006. 6

[29] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *Int'l Journal of Computer Vision*, 80(2):189–210, 2008. 3

[30] C. Strecha. Dense mvs dataset. `http://cvlabwww.epfl.ch/~strecha/multiview/denseMVS.html`. 1, 5, 6, 7

[31] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Pattern Recognition*, pages 11–20. Springer, 2010. 2

[32] P. Tanskanen, K. Kolev, L. Meier, F. Camposeco, O. Saurer, and M. Pollefeys. Live metric 3d reconstruction on mobile phones. In *Proc. of Int'l Conf. on Computer Vision (ICCV)*. IEEE, 2013. 2

[33] E. Tola, C. Strecha, and P. Fua. Efficient large scale multi-view stereo for ultra high resolution image sets. *Machine Vision and Applications*, 23(5):903–920, 2012. 1

[34] C. Wu. Visualsfm: A visual structure from motion system. `http://ccwu.me/vsfm/`, 2011. 3, 6

[35] A. Zaharescu, C. Cagniart, S. Ilic, E. Boyer, R. Horaud, et al. Camera-clustering for multi-resolution 3–d surface reconstruction. In *Workshop on Multi-camera and Multi-modal Sensor Fusion Algorithms and Applications-M2SFA2 2008*, 2008. 2