3D surface reconstruction from point-and-line cloud

Takayuki Sugiura Akihiko Torii Masatoshi Okutomi
Tokyo Institute of Technology*
{ tsugiura@ok., torii@, mxo@ }ctrl.titech.ac.jp

Abstract

We present a method for reconstructing 3D surface as triangular meshes from imagery. The surface reconstruction requires 3D point cloud for composing vertices of triangle meshes. A standard approach uses incremental structure from motion (SfM) to obtain camera poses and sparse 3D point cloud that are given based on 2D key-point matching. As the 3D surface directly reconstructed from the sparse 3D point cloud often lack detail of objects, multiple-view stereo (MVS) is commonly used to generate dense 3D point cloud. A known problem with the densification is that MVS generates many small patches even for planar flat objects that degrade the quality of surface model. Using dense 3D point cloud also requires high memory capacity for visualization. In this work, we propose to reconstruct 3D surface using sparse 3D point cloud generated by SfM and 3D line segments (3D line cloud) computed from multiple views since these two elements can complement well for representing man-made structures. The proposed method extends the tetrahedra-carving method as it can use 3D point-and-line cloud under the global optimization framework. We demonstrate that the proposed method can efficiently produce surface models whose quality are at least as good as the baseline method using dense 3D point cloud.

1. Introduction

Given a collection of unordered or sequences of images, the state-of-the-art Structure from Motion (SfM) [27, 1, 4, 32, 23, 26] robustly gives camera poses and sparse 3D point cloud. 3D surface can be then computed directly from the SfM point cloud [14, 34]. As the 3D surface recovered from sparse 3D point cloud often lack detail, a standard approach densifies 3D point cloud for representing surface with fine shapes. This can be achieved by Multiple View Stereo (MVS) [7, 6, 31] and surface extraction-fitting [17, 20].

Accuracy of recovered 3D surfaces is dependent on density of 3D point cloud and depth estimation for every pixel (or patch) in MVS. MVS based on photometric similarity of local patches works well for well-textured scenes but poorly on texture-less scenes such as man-made objects and modern buildings. These structures are dominated by 3D line segments which make them ideal for line (segment) based reconstruction [15, 11, 12, 22]. Since the recovered 3D line segments (refered as 3D line cloud in this paper) intuitively represent the scene structures, they can be directly used for visualization. Furthermore, since the 3D line cloud compliment geometric shapes missing from 3D point cloud, they can be integrated with point-based surface reconstruction method to produce better models.

Although the surface reconstruction with dense 3D point cloud can provide impressive results for many scenes, a known problem is that the 3D surface representing planar objects become unnecessarily rough since noise on plane normal direction is amplified. Smoothing surface is a natural solution but not trivial due to the shape scale problem [5].

In this work, given camera poses and a sparse 3D point cloud obtained from SfM, we propose a method for efficiently reconstructing high quality 3D surface without redundantly densifying 3D point cloud (figure 1). We first reconstruct 3D line cloud based on robust 2D line matching guided by initial 3D surface [20, 13], efficiently extracted from the sparse 3D point cloud. We then integrate the reconstructed 3D line cloud with the 3D point cloud and finally extract 3D surface by finding the global optimum via tetrahedra-carving method. It is important to note that the proposed method effectively fuses the 3D point and line clouds for the 3D surface extraction.

1.1. Related work

For high quality 3D surface generation from imagery, a standard approach starts from densifying a sparse 3D point cloud resulted from SfM based on 2D feature (keypoint) matching. This 3D point cloud can be densified to obtain

*Department of Mechanical and Control Engineering, Graduate School of Science and Engineering, Tokyo Institute of Technology. 2-12-1, Ookayama, Meguro-ku, Tokyo, 152-8550, Japan.

978-1-4673-8332-5/15 $31.00 © 2015 IEEE
DOI 10.1109/3DV.2015.37

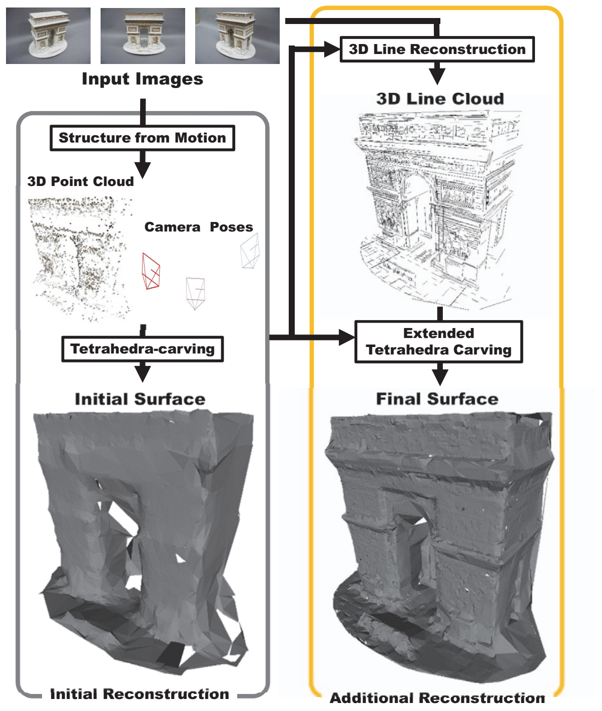

Figure 1. Overview of the proposed method. See text in section 1.3 for details.

a detailed model by plane sweeping [31] or MVS based on local photometric consistency [6]. After densification of 3D point cloud, the 3D surface can be reconstructed based on oriented 3D points via Poisson surface reconstruction [18] or ray information of 3D points via tetrahedra-carving methods [20, 16, 31].

While densifying the 3D point cloud often improves quality of reconstructed 3D surface, uniformly sampling pixels may be redundant especially for reconstructing planar objects. To achieve a memory-efficient representation of 3D surfaces, geometric primitives are used for representing 3D model. Larfarge *et al*. [21] uses hybrid representation that combines planes, cylinders and triangular meshes. Wu *et al*. [33] obtains 3D surface by sweeping contours on slices holizontal and vertical planes. These methods give compact 3D models in high quality for man-made objects but adopting post-processing methods such as smoothing are not trivial since the 3D models are represented by polygons specific to the associated primitives.

As a fundamental geometric primitive, 3D line cloud can be also used for representing 3D models. Recent approaches in 3D line reconstruction consists of two steps: it first performs endpoint-less 2D line matching as robust as possible; it further refines the 2D line correspondences in the 3D line triangulation step by performing neighboring 3D line fusion [15, 11, 12] or bundle adjustment [22]. Notice that these methods result 3D line cloud but not 3D surface. Hofer *et al*. [12] only add the reconstructed 3D line cloud in the form of 3D points sampled on the 3D line segments. No existing methods have considered the use of 3D line cloud as constraints in surface reconstruction.

When computational efficiency is severely required in surface reconstruction, one of common approaches simply uses the 3D point cloud by SfM. This approach is often used in fully incremental fashion [14, 13, 34] to estimate camera pose of new frame and update 3D surface model simultaneously. Densification of 3D point cloud is possible when sequential images (video) are used [24] but prevents efficient reconstruction of 3D surface. Reconstructing high quality 3D surfaces represented by triangle mesh in a memory and computationally efficient way remains a big challenge.

1.2. Contributions

In this paper, we propose a method to efficiently reconstruct 3D surface as triangular meshes without loss in quality when compared to 3D point cloud densification methods. The proposed method uses both the 3D sparse point cloud and the geometric information contained in 3D line cloud to produce 3D surface which preserves sharp edges and flat planes. We achieve this by smoothly integrating 3D point and line clouds in surface extraction step while introducing “3D triangular rays” for intersection computation between tetrahedra and point and line cloud. This way also preserves the global optimality ensured by the tetrahedra carving method.

1.3. Overview of the proposed method

As illustrated in figure 1, the proposed method consists of four steps: (1) The sparse 3D point cloud and camera poses are first estimated from a set of input images by SfM [32]. (2) Initial 3D surface from the sparse 3D point cloud is estimated using the tetrahedra-carving [20]. This algorithm is reviewed in section 3.1. (3) 3D line cloud are reconstructed using multiple-view correspondences computed by 2D line-to-line matching for pairs of images (section 2). (4) Finally, the 3D surface model is generated from reconstructed 3D line cloud and sparse 3D point cloud using a new tetrahedra-carving method (section 3).

2. 3D line reconstruction

In this section, we describe how 3D line cloud is reconstructed. The algorithm starts with extracting 2D line segments and pre-computing descriptors for every image (section 2.1). The extracted 2D line segments are then matched using the camera poses and the initial 3D surface (section 2.2) as a guide. Finally, the matched 2D line segments are triangulated and mismatched correspondences are removed (section 2.3). In contrast to the recent 3D line reconstruction approaches that recover many potential 3D line segments, we aim at generating 3D line segments with as little mismatch as possible even if some correct matches are missing.

2.1. Pre-processing

In the first step of 3D line reconstruction, we detect 2D line segments in every input image using Line Segment Detector (LSD) [8] which can robustly detect 2D line segments with parameter-free. For robust 2D line-to-line matching, we next detect DAISY descriptor [30] which is a dense (pixel-wise) local descriptor and has SIFT-like histogram of gradients designed for matching wide baseline images. Finally, we compute depth maps by detecting intersections of the rays (half line connecting a camera center and pixel) to the initial 3D surface estimated by the 3D point cloud. These depth maps are used for determining the initial 2D positions in the following 2D line-to-line matching.

2.2. 2D line-to-line matching

2D line-to-line matching is performed on pairs of images which share a large field of view. We simply compute the overlap between a pair of images using the number of common 3D points.

In a pair of images, we choose the image having fewer number of 2D line segments as the reference image I_{ref} and the other as the target image I_{tar}. Each detected 2D line segment $l_i \in L_{\mathrm{ref}}$ in the reference image is matched to the 2D line segments detected in the target image.

In detail, we first sample N_{sample} 2D points $\mathcal{P}_{\mathrm{ref}} = \{p\}$ from a 2D line segment l_i in I_{ref} and seek the best match for each sampled 2D point p from its epipolar line in the target image using the precomputed DAISY descriptors (figure 2). For efficient search, we start from the position q_0 in I_{tar} computed using the initial depth map and seek the local minimum q of DAISY descriptor distances using gradient descent. Finally, cross matching is performed to remove outliers, *i.e.* the match is accepted if $\hat{p}$ matched with q is in the vicinity (1 pixel) of the reference point p. Note that this cross matching based on sampled 2D points can remove unstable matches due to partial occlusion on the 2D line segments.

After finding the set of 2D points $\mathcal{P}_{\mathrm{match}}$ in the target image matched to the sampled 2D points $\mathcal{P}_{\mathrm{ref}}$ on l_i, we fit a 2D line segment $\hat{l}_{\mathrm{match}}$ (figure 3). We then search the most similar line to the $\hat{l}_{\mathrm{match}}$ in the set of detected 2D line segments $\mathcal{L}_{\mathrm{tar}}$ by evaluating slopes and distances between lines and endpoints. If there exists a line segment l_{match} similar to $\hat{l}_{\mathrm{match}}$, we accept it as a putative match of the reference 2D line segment after passing line-wise cross matching. This matching is performed for every 2D line segment in the reference image.

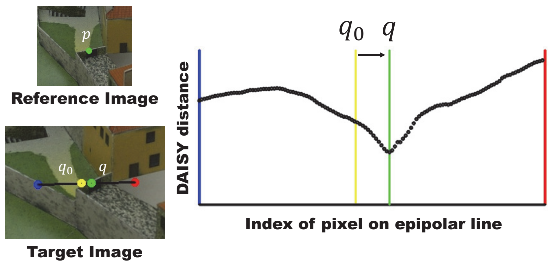

Figure 2. DAISY descriptor matching using gradient descent. We search the local minimum q on the epipolar line by starting at the initial position q_0 computed from the initial depth map.

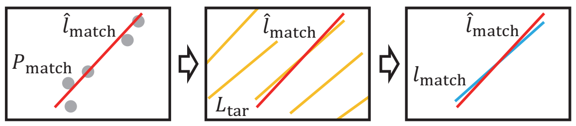
$\hat{l}_{\mathrm{match}}$
P_{match}
$\hat{l}_{\mathrm{match}}$
L_{tar}
$\hat{l}_{\mathrm{match}}$
l_{match}

Figure 3. Robust 2D line-to-line matching in the target image. (Left) A line (red line) $\hat{l}_{\mathrm{match}}$ is fitted to the set of points $\mathcal{P}_{\mathrm{match}}$ which are matches to the points sampled on the reference line. (Middle) The line most similar to $\hat{l}_{\mathrm{match}}$ (red line) is selected as a candidate from the set of lines $\mathcal{L}_{\mathrm{tar}}$ (orange lines) detected by LSD. (Right) The candidate line is accepted as a matched line if the l_{match} (blue line) and $\hat{l}_{\mathrm{match}}$ (red line) are close.

2.3. 3D line segment triangulation using multiple views

We reconstruct 3D line segments using 2D line matches from multiple views. We trace the putative pairs of 2D line segments to generate tracks. The 3D line segment is triangulated by computing intersection of the viewing planes of 2D line segments in the track. The endpoints of reconstructed 3D line segment are computed in such a way that a 2D line segment of a member of the track should be included between the endpoints projected in the image. 3D line segments visible from only two views and those that have large reprojection errors are removed.

It is important to note that there is a degeneracy in 3D line segment triangulation [9] when the reconstructed 3D line segment and camera positions are on a plane. In this case, the estimated depth is ambiguous even if the matching is correct. To avoid this, the 3D line segments are removed when the distance between the reconstructed 3D line and the camera baseline is smaller than a threshold.

3. Surface reconstruction from point-and-line cloud

In this section, we describe the proposed extended tetrahedra-carving method that uses the reconstructed 3D line cloud as additional input for surface extraction.

We first review the surface extraction algorithm based on tetrahedra-carving method [20] to clarify contributions of the proposed method. Next, the original tetrahedra-carving method is extended to make use of both 3D points and line cloud.

3.1. Review of tetrahedra-carving method

Given the 3D point cloud $\mathcal{P} = \{p\}$, camera positions $\mathcal{C} = \{c\}$ and ray information $\mathcal{R} = \{r\}$, the 3D space is first segmented into the tetrahedra $\mathcal{T} = \{t\}$ by 3D Delaunay triangulation (tetrahedralization) of $\mathcal{P}$.

Formulation. To extract 3D triangular-mesh surface as a set of triangle facets of tetrahedra, every tetrahedra t is labelled whether it belongs to object or to free-space. This is formulated as a binary labeling problem where a directed graph $\mathcal{G}$ consists of nodes $\mathcal{T}$ (tetrahedra $\mathcal{T} = \{t\}$) and edges $\mathcal{E} = \{e = (t, v)\}$ (facets $\mathcal{F} = \{f\}$ shared by adjacent tetrahedra). This problem can be solved by minimizing the following cost function:

$$E(\mathcal{L}) = \sum_{t \in \mathcal{T}} \{U_t(l_t) + \sum_{v \in \mathcal{V}_t} B_{t,v}(l_t, l_v)\} \tag{1}$$

where l_t is the label for a tetrahedron t which indicates object ($l_t = OBJ$) or free-space ($l_t = FS$). $\mathcal{V}_t = \{t\}$ is the set of four tetrahedra adjacent t.

The unary term U_t is the penalty for inconsistencies between labels and rays around t.

$$U_t(l_t) = \begin{cases} N_{\text{obj}}(t) & \text{if } l_t \neq \text{OBJ} \\ N_{\text{fs}}(t) & \text{if } l_t \neq \text{FS} \end{cases} \tag{2}$$

N_{obj} is the number of rays intersecting to a tetrahedra t, see figure 4). For the specific tetrahedra that include cameras inside, N_{fs} is computed as the number of rays emanated from the cameras. This is based on the idea that all the cameras should exist in the free-space region.

The binary term $B_{t,v}$ is the smoothness of labels at the facet shared by adjacent tetrahedra.

$$B_{t,v}(l_t, l_v) = \begin{cases} N_{\text{intrsct}}(t, v) & \text{if } l_t = \text{FS} \cap l_v = \text{OBJ} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $N_{\text{intrsct}}(t, v)$ is the number of rays intersecting the facet $f(t, v)$ shared by adjacent terahedra from the side of t to v.

Note that minimizing the cost $E(\mathcal{L})$ gives the minimum number of inconsistent intersections between the extracted surface and the rays observed from cameras.

Optimization. To minimize $E(\mathcal{L})$, the conventional max-flow/min-cut algorithm [2] can be used. Using the s-t graph-cut, the 3D surface can be extracted as a solution of the globally optimal binary labeling problem to the nodes $\mathcal{T}$ in the graph $\mathcal{G}$. The weight of each edge $e = (t, v) \in \mathcal{E}$ in the graph is assigned with the number of related rays by detecting intersections among tetrahedra and rays. The tetrahedra skewered by each ray r composed from a 3D point p and a camera c is efficiently detected by starting the trace from the tetrahedra having the vertices p until reaching the tetrahedron including c inside. See [20] for more details.

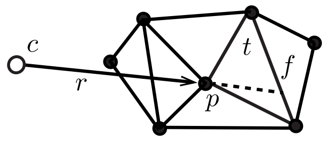

Figure 4. Notations w.r.t. ray intersection illustrated in 2D. The solid line is the facet f of tetrahedra t. The arrow is the ray r composed from the camera c and the point p. The dashed line indicates the extended ray of r which is used for computing $N_{\text{obj}}(t)$ in the cost function (equation 1).

Post-processing. The two additional processes to refine the 3D surface are followed after the extraction by tetrahedra-carving method. One is that very large 3D triangle patches are removed by evaluating the longest edges of triangles. The other process is the surface smoothing by incrementally updating every vertex as

$$p_{i+1} = p_i + \lambda \sum_{q_i \in V_p} w_{p_i,q_i}(q_i - p_i) \tag{4}$$

where V_p is the set of neighboring 3D points of p on the surface, λ is a parameter for controlling the update, and $w_{p_i,q_i} =$ is the weight computed using the inverse distance $\phi(p, q)$ of two points. This simple smoothing process improves the surface, especially the planar objects.

3.2. Surface extraction with 3D line cloud

To extract 3D surface using both 3D points and line cloud we extend the original tetrahedra-carving method [20] as described in section 3.1.

Delaunay triangulation with line segments. To effectively use 3D line cloud in surface extraction, the 3D line segments have to remain as edges of tetrahedra. A simple approach is to adopt the constrained Delaunay triangulation [25]. This method can generate tetrahedra while preserving every 3D line segment (figure 5(b)). However, the constraint Delaunay triangulation tends to generate tetrahedra with long edges that deteriorate the quality of surfaces. We confirmed this negative artifact in our preliminary experiments (not included).

To overcome this problem, we sample every 3D line segment with a constant interval d_{smp} in 3D space, integrate the 3D points generated from SfM and the 3D points p_l sampled on the 3D line segments, and compute tetrahedra from them using the standard Delaunay triangulation. When the interval d_{smp} is smaller than the spatial density of neighboring

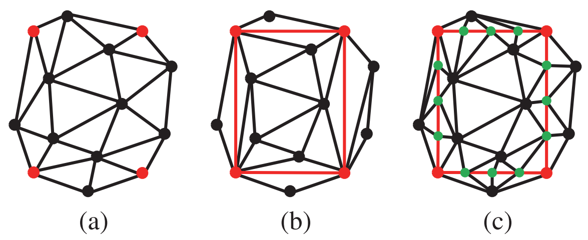
Figure 5. Delaunay triangulation with points and line segments (illustrated in 2D). (a) Delaunay triangulation using points from SfM (black dots) and endpoints of four line segments (red dots). (b) Constrained Delaunay triangulation using the line segments (red lines) as firm constraints. Notice that long and thin triangles appear as a result of triangulation. (c) Delaunay triangulation using points (green dots) sampled on the line segments (red lines).

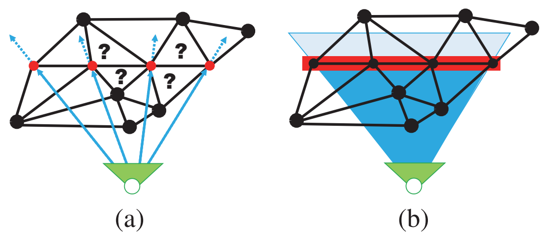
Figure 6. Effect of triangular-ray for line segment (illustrated in 2D). (a) Using standard (point-to-point) rays (blue arrows) defined in the original method to the points (red points) sampled on the constraint line segment, some tetrahedra around the line segment have no contribution to the extraction of surface. (b) The proposed triangular-ray (line-to-point) ray (blue triangle) can detect every tetrahedron around constraint line segment (red line).

vertices, the 3D line segments remain as edges of tetrahedra though some are decomposed to several pieces (figure 5(c)).

Rays for 3D line segments. On top of the Delaunay triangulation with both 3D points and line segments, we next aim at including rays associated to 3D line segments in the cost function (equation 1), *i.e.* adopting the intersection of them into the weights in the graph-cut formulation. Since the 3D line cloud may be reconstructed from mismatches and may have noise, the 3D surface should be robustly extracted. We re-formulate the cost function that takes both 3D points and line cloud in the unified minimization framework, as its global minimum gives the best 3D surface.

If we naively apply the original tetrahedra-carving method to the 3D points (red dots in figure 6(a)) sampled on 3D line segments, no intersection occurs for some tetrahedra as no ray exist among the sampled 3D points and the labels for these tetrahedra become unstable. This is because the tetrahedra having no intersections gives no effect on the cost function.

To overcome this problem, we use a 3D triangular-ray composed from a 3D line segment and a camera position (figure 6(b)). In detail, we compute weights associated with 3D line segments on the graph $\mathcal{G}$ by detecting intersections between 3D triangular-rays and facets of tetrahedra. In this approach, all the intersections w.r.t. the 3D line segments should be detected because using the 3D triangular-ray is equivalent to infinitely 3D points sampled on the 3D line segment. Similar to the case of ray-and-facet intersection detection in the original method, the intersections between a 3D triangular-ray and facets of tetrahedra can be detected by using queue processing:

- Mark all tetrahedra $\mathcal{T}$ as 0 (not checked).
- As an initial queue, find the tetrahedra Q that include at least one 3D point p_l sampled on a 3D line segment as its vertex.
- Pick a tetrahedron t in the queue Q, evaluate whether each of four facets of t intersects the associated 3D triangular-ray, and mark t as 1 (finished).
- If the tetrahedron t intersects with a 3D triangular-ray and the neighboring terahedron u sharing a facet with t is not marked yet, add u to the queue Q.

All the tetrahedra intersecting with the 3D triangular-rays can be detected by repeating this queue processing until convergence (we have experimentally verified the convergence).

Notice that the intersections of 3D triangular-rays and facets can be regarded as a generalization of ray-and-facet intersection because a standard ray (point-to-point) can be also represented as a 3D triangular-ray (line-to-point) when we consider a point as infinitely short line (edge). Therefore, we can deal with 3D points and 3D line segments in the same global optimization framework.

Weighted smoothing for extracted 3D surface. We also modify the smoothing step after the surface extraction. If the 3D points sampled on the 3D line segments are dealt equivalently with the 3D points from SfM, the edges that appear on the extracted 3D surface will be over-smoothed. Therefore, we use different smoothing parameters λ in equation 4 for SfM 3D points and 3D points sampled on 3D line segments. We set the parameter λ_p for the SfM 3D points larger than the parameter λ_s for the sampled 3D points in order to keep the edges sharp. We use $\lambda_p = 1$ and $\lambda_s = 0.1$ throughout the experiments.

4. Experiments

In this section, we describe the experimental results. First, we qualitatively evaluate extracted surfaces in comparison with the baseline methods that use point cloud only. We next compare the efficiency in computational cost and in memory requirement. Finally, we evaluate the impact of adding 3D line segments in the surface reconstruction.

We evaluated the proposed method on a desktop PC with Intel Core i7, CPU 3.20GHz, RAM 64GB and Linux 64bit OS using C++ implementation with the following libraries: OpenCV [3]; CGAL [29]; dynamic graph-cut library [19]; Line Segment Detector [8]; DAISY descriptor [30]. The pair-wise line-to-line matching and intersection detection in tetrahedra carving are multi-threaded programs.

4.1. Qualitative evaluation for the proposed method

We evaluate the 3D surfaces reconstructed by the proposed method using both 3D points and 3D line segments in comparison with the baseline methods [6, 18] publicly available.

Daliborka dataset. Figure 7 shows the results of surface reconstruction on the Daliborka dataset [10]. We use 40 images (a) of 1100×850 pixels capturing a paper craft for 2 laps around. The camera poses and sparse point cloud (b) are estimated by VisualSfM [32]. The initial surface (c) roughly represents geometry of the objects but lacks some details, *e.g*. ridges of roofs. Using 3D line segments (d) reconstructed by the proposed method in section 2, the proposed surface reconstruction produced the 3D surface model (e) with more detail while preserving planar facets. Notice that this is the advantage of using sparse point cloud and line segments since triangular patches are not unnecessarily created. Our result also shows the improvement in comparison with the surface (g) reconstructed using the dense point cloud from PMVS [6] (f) and Poisson reconstruction [18]. Optionally, our method can use the surface (h) computed from dense point cloud by PMVS as an initial surface for the line-to-line matching and reconstruct the surface using both dense point cloud and line segments. The resulted final surface (i) shows some improvement when compared to the initial surface (h) but not as significant as when using sparse point cloud (e).

Castle-P19 dataset [28]. Castle-P19 dataset (figure 8) consists of 19 images (a) of 3072×2048 pixels. The result produced by the proposed method (e) has more detail, *e.g*. windows and roofs, when compared with using sparse point cloud only (c). PMVS+Poisson surface reconstruction also gives fine details (g) but it has rough planar facets. When our method uses the dense point cloud and line segments (i), the result shows less significant improvement but has sharper edges and corners.

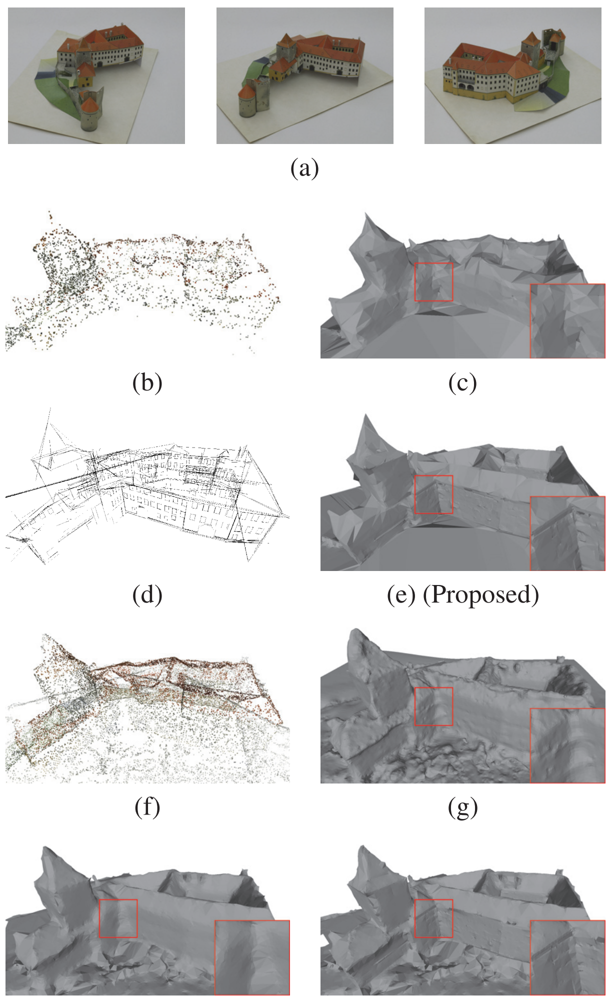

Figure 7. Results of surface reconstruction on Daliborka dataset. (a) Examples of input images. (b) Point cloud generated by SfM. (c) Initial surface recovered using (b). (d) 3D line segments reconstructed by the proposed line-to-line matching. (e) 3D surface resulted by the proposed method. (f) Dense point cloud generated by PMVS [6]. (g) 3D surface reconstructed by Poisson reconstruction [18] using (f). (h) Initial surface recovered using (f). (i) 3D surface using (f) and line segments as an optional result of our method.

4.2. Efficiency of the proposed method

We also evaluate the efficiency of the proposed method. The statistics of the proposed method and the baseline methods are summarized in Tables 1 and 2. Each column corresponds the result in figures 7 and 8.

The computational time is shown in the third row. The proposed method is 5 and 8 times faster than PMVS and Poisson surface reconstruction in each dataset. The computational time required in the proposed method depends on the size of images in the line-to-line matching and the number of 3D points and line segments in tetrahedra carving. Notice that our method is significantly faster than all the other methods even when DAISY is computed for all the pixels.

In addition to the computational time, the proposed method has an impact on the efficiency of 3D surface model representation, *i.e*. the number of points and patches. The surface reconstructed by PMVS and Poisson surface recon-

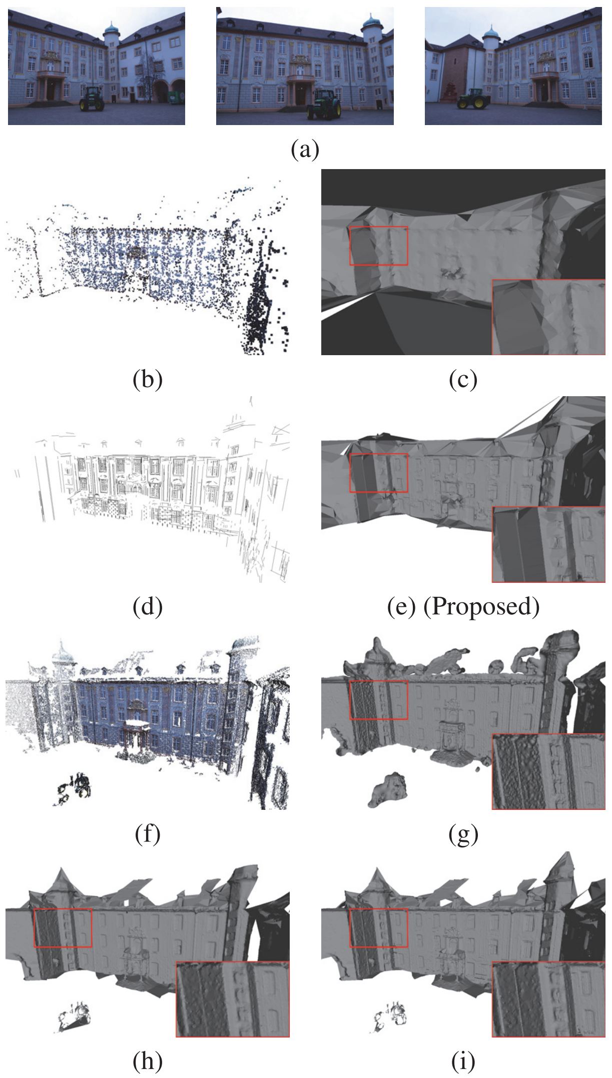
Figure 8. The reconstruction of our method in **Castle-P19** dataset. See the caption of Figure 7 for detail.

struction has more than 9 times the vertices and triangle patches created from the proposed method in the Daliborka and Castle-P19 dataset. In the Daliborka dataset, Poisson surface reconstruction produces 179,670 regular-grid vertices when using the dense point cloud produced by PMVS which consist of 48,598 points.

4.3. Which step helps?

Impact of 3D triangular rays. Figure 9 shows effect of using 3D triangular rays in the tetrahedra carving. When using only rays for the point sampled from the 3D line constraints, the resulted surface (c) has jaggy edges due to missing ray-tetrahedra intersection information among the sampled points. The proposed method can reconstruct the surface (d) with straight lines by detecting all the intersection

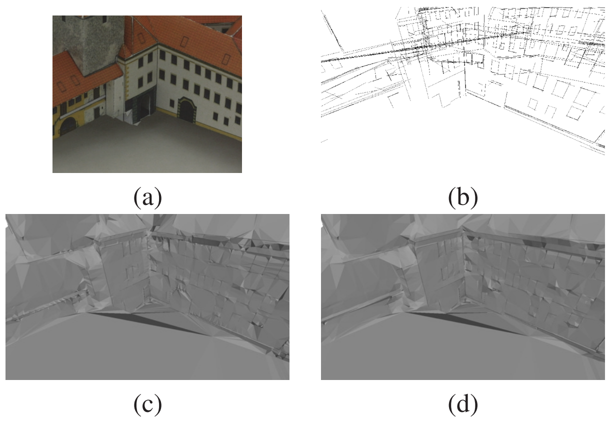
Figure 9. Effect of ray information about 3D line segment. (a) Close-up of example input image. (b) The 3D line segments reconstructed by the proposed method. (c), (d) The surface with 3D line constraints in black lines. (c) Using only rays for the point sampled on the 3D line constraints, there are jaggy noise around the constraints on the surface. (d) The surface is more smoothed on the constraints with rays for 3D line segments.

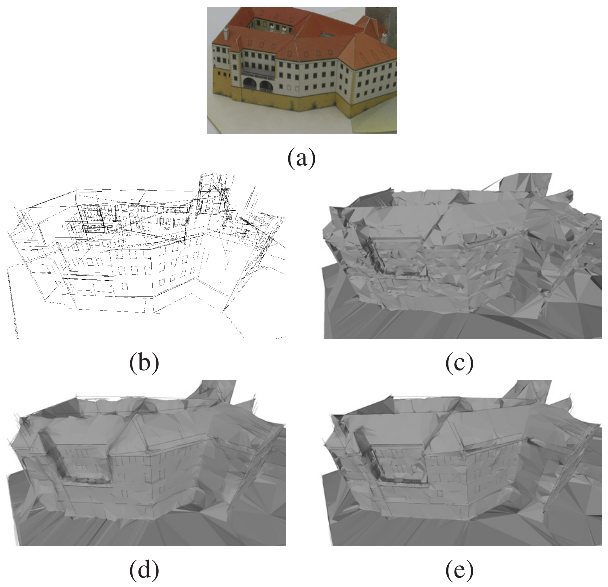
Figure 10. Effect of 3D line constraint in smoothing surface. (a) Close-up example of input image. (b) The 3D line segments reconstructed by the proposed method. (c) The surface without smoothing. (d) The smoothed surface with the weight λ for feature-based points and points sampled on 3D lines. (e) Our smoothed surface with different weight λ_p and λ_s for each type of points.

of tetrahedra and 3D triangular rays.

Impact of smoothing weights. We also evaluate effect of weights λ_p and λ_s in smoothing process. Figure 10 shows 3D line segments (b), the raw surfaces (c) extracted by the proposed tetrahedra carving, and the smoothed surfaces us-

Table 1. Statistics in Daliborka dataset

	[32]+[20]	Proposed	[6] + [18]	[6] + [20]	[6] + Proposed
	Fig. 7(c)	Fig. 7(e)	Fig. 7(g)	Fig. 7(h)	Fig. 7(i)
Computational time [sec]	0.19	8.32	29 + 11	29 + 1.9	29 + 12.29
Number of points	7,746	19,131	179,670	48,598	59,862
Number of lines	-	1,122	-	-	1,130
Number of patches	15,070	34,655	385,232	96,956	114,303

Table 2. Statistics in Castle-P19 dataset

	[32]+[20]	Proposed	[6] + [18]	[6] + [20]	[6] + Proposed
	Fig.8(c)	Fig.8(e)	Fig.8(g)	Fig.8(h)	Fig.8(i)
Computational time [sec]	2.2	36.5	214 + 82	214 + 21	214 + 214
Number of points	14,941	62,244	824,602	531,368	582,270
Number of lines	-	2,655	-	-	2,821
Number of patches	29,271	120,713	1,639,551	1,069,885	1,157,835

ing the same parameter (d) and the separate parameters (e). The smoothed surface (d) partly lacks some edges such as ridges of roofs. The surface (e) produced by the proposed method has sharp edges while preserving flat facets.

5. Conclusion

We propose a method to efficiently reconstruct high quality 3D surface as triangular meshes by integrating the 3D line segments with the point clouds obtained from SfM. The proposed method first reconstructs 3D line segments by robust and efficient line-to-line matching. The surfaces are extracted by finding the global optimum in the modified tetrahedra carving approach that can use points and line segments in the same framework. The experimental results showed significant improvements when using both points and line segments and the statistics promised the efficiency of the method.

Acknowledgement

This work was partially supported by the Grants-in-Aid for Scientific Research (no. 25240025, 15H05313) from the Japan Society for the Promotion of Science.

References

[1] S. Agarwal1, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building Rome in a day. In *ICCV*, pages 72–79, 2009.

[2] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *PAMI*, 26(9):1124–1137, 2004.

[3] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] 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 *ECCV*, pages 368–381, 2010.

[5] S. Fuhrmann and M. Goesele. Floating Scale Surface Reconstruction. In *SIGGRAPH*, 2014.

[6] Y. Furukawa and J. Ponce. Accurate, dense, and robust multiview stereopsis. *PAMI*, 32:1362–1376, 2010.

[7] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. Seitz. Multi-view stereo for community photo collections. In *ICCV*, pages 1–8, Oct 2007.

[8] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. LSD: a Line Segment Detector. *IPOL*, 2:35–55, 2012.

[9] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[10] M. Havlena, A. Torii, J. Knopp, and T. Pajdla. Randomized structure from motion based on atomic 3D models from camera triplets. In *CVPR*, pages 2874–2881, 2009.

[11] M. Hofer, M. Donoser, and H. Bischof. Semi-global 3d line modeling for incremental structure-from-motion. In *BMVC*, 2014.

[12] M. Hofer, M. Maurer, and H. Bischof. Improving sparse 3d models for man-made environments using line-based 3d reconstruction. In *3DV*, 2014.

[13] C. Hoppe, M. Klopschitz, M. Donoser, and H. Bischof. Incremental surface extraction from sparse structure-from-motion point clouds. In *BMVC*, 2013.

[14] C. Hoppe, M. Klopschitz, M. Rumpler, A. Wendel, S. Kluckner, H. Bischof, and G. Reitmayr. Online feedback for structure-from-motion image acquisition. In *BMVC*, pages 70.1–70.12, 2012.

[15] A. Jain, C. Kurz, T. Thormählen, and H.-P. Seidel. Exploiting global connectivity constraints for reconstruction of 3d line segment from images. In *CVPR*, 2010.

[16] M. Jancosek and T. Pajdla. Multi-view reconstruction preserving weakly-supported surfaces. In *CVPR*, pages 3121–3128, 2011.

[17] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Eurographics*, SGP '06, pages 61–70, 2006.

[18] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM Trans. Graphics*, 32(3):29:1–29:13, 2013.

[19] P. Kohli and P. H. S. Torr. Effciently solving dynamic markov random fields using graph cuts. In *ICCV*, pages 922–929, 2005.

[20] P. Labatut, J.-P. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *ICCV*, pages 1–8, 2007.

[21] F. Lafarge, R. Keriven, M. Brédif, and H.-H. Vu. Hybrid multi-view reconstruction by jump-diffusion. In *CVPR*, pages 350–357, 2010.

[22] B. Micusik and H. Wildenauer. Structure from motion with line segments under relaxed endpoint constraints. In *3DV*, 2014.

[23] P. Moulon. openMVG. http://imagine.enpc.fr/∼moulonp/openMVG/index.html, 2012.

[24] R. A. Newcombe, S. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *ICCV*, pages 2320–2327, 2011.

[25] H. Si. Tetgen, a delaunay-based quality tetrahedral mesh generator. *ACM Trans. Math. Softw.*, 41(2):11:1–11:36, 2015.

[26] F. L. Simon Fuhrmann and M. Goesele. MVE - A Multi-View Reconstruction Environment. In *Proceedings of the Eurographics Workshop on Graphics and Cultural Heritage (GCH)*, 2014.

[27] N. Snavely, S. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *IJCV*, 80(2):189–210, 2008.

[28] C. Strecha, W. von Hansen, L. J. V. Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008.

[29] CGAL. Computational Geometry Algorithms Library. http://www.cgal.org.

[30] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *PAMI*, 32(5):815–830, May 2010.

[31] H.-H. Vu, P. Labatut, J.-P. Pons, and R. Keriven. High accuracy and visibility-consistent dense multiview stereo. *PAMI*, 34(5):889–901, 2012.

[32] C. Wu. VisualSFM: A visual structure from motion system. http://homes.cs.washington.edu/ ccwu/vsfm/, 2011.

[33] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Schematic surface reconstruction. In *CVPR*, pages 1498–1505, 2012.

[34] S. Yu and M. Lhuillier. Incremental reconstruction of manifold surface from sparse visual mapping. In *3DIMPVT*, pages 293–300, 2012.