

Multi-View Reconstruction Preserving Weakly-Supported Surfaces

Michal Jancosek and Tomas Pajdla

Center for Machine Perception, Department of Cybernetics
Faculty of Elec. Eng., Czech Technical University in Prague

{jancom1,pajdla}@cmp.felk.cvut.cz

Abstract

We propose a novel method for the multi-view reconstruction problem. Surfaces which do not have direct support in the input 3D point cloud and hence need not be photo-consistent but represent real parts of the scene (e.g. low-textured walls, windows, cars) are important for achieving complete reconstructions. We augmented the existing Labatut CGF 2009 method with the ability to cope with these difficult surfaces just by changing the t -edge weights in the construction of surfaces by a minimal s - t cut. Our method uses Visual-Hull to reconstruct the difficult surfaces which are not sampled densely enough by the input 3D point cloud. We demonstrate importance of these surfaces on several real-world data sets. We compare our improvement to our implementation of the Labatut CGF 2009 method and show that our method can considerably better reconstruct difficult surfaces while preserving thin structures and details in the same quality and computational time¹.

1. Introduction

Recent approaches to multi-view reconstruction [21, 4, 22, 14, 5, 9, 8] attain the degree of accuracy and completeness comparable to laser scans [18, 20]. Yet, producing complete reconstructions of outdoor and complicated scenes is still an open problem. Most of the state-of-the-art multi-view reconstruction methods produce a 3D point cloud, which is later used to compute a mesh representing the 3D world. There are several approaches to computing the point cloud, such as plane-sweeping based [6], stereo based [19], or growing based [8] methods. We are focusing on the multi-view reconstruction from a given 3D point cloud and cameras producing it. We further assume that for every point in the cloud we have the list of cameras that should see it.

¹The authors were supported by SGS10/186/OHK3/2T/13, FP7-SPACE-241523 PRoViScout and MSM6840770038. We thank M. Havlena for technical assistance.

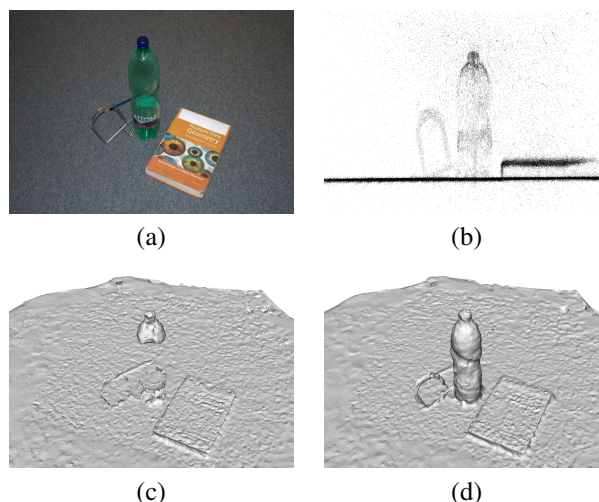


Figure 1. **Results for the 'bottle' data set.** (a) Input image, (b) input 3D point cloud, (c) results using our implementation of [15], (d) the technique presented in this work reconstructs weakly-supported surfaces (the bottle) better.

Surfaces which do not have direct support in the input 3D point cloud and hence do not have to be photo-consistent with it but represent real surfaces in the scene (e.g. low-textured walls, windows, cars, ground planes) are important for achieving complete reconstructions. Such surfaces may be transparent, highly reflective, lacking textures or in the case of ground planes they can be occluded by moving objects like people and cars. The PET bottle shown in Figure 1(a) is an example. Another frequent example would be the ground plane in tourist photo collections which is often blurred since the cameras are mostly focused on houses and people above the ground. It is almost impossible to reconstruct such *difficult surfaces* directly using standard stereo matching techniques which are based on photo-consistency measures [21, 4, 22, 14, 5, 9, 8]. *Difficult surfaces* can co-exist with highly textured *easy surfaces* which can be quite accurately reconstructed with the standard techniques. We consider a surface densely sampled by the input 3D points as an *easy surface*. Figure 2 shows a highly-supported-free-

space cone (light blue triangles with dark blue shape outline), it is the part of the space which is between an *easy surface* and its associated camera. The idea of the Visual-Hull [16] method is that the silhouette of objects splits the image into foreground and background. The silhouette, together with the camera centre c , defines the silhouette cone which contains the object. Given N silhouette images, the scene can be restricted to the intersection of the corresponding cones. The another explanation is that the result is the complement to the union of the complements to the silhouette cones.

Motivated by the idea of the Visual-Hull, we define a *weakly-supported surface* as a specific part of a difficult surface, see Figure 2. The weakly-supported surfaces (green) are the surfaces of real objects which are weakly sampled by the input 3D points and are close to the border of the union of the highly-supported-free-space cones, i.e. close to highly-supported-free-space boundary. Our aim is to reconstruct them. We focus on creating an initial surface which can be later refined as in [21] to achieve excellent high detailed results.

Similarly to [15], our approach casts the surface reconstruction as an energy minimisation problem that can be globally optimised by computing a minimum s-t cut in an s-t graph. The new element in our approach is in computing the free space support, then detecting the highly-supported-free-space boundary and finally changing the t-weights of the graph to reconstruct even the weakly-supported surfaces. We demonstrate the performance of our method in synthetic example (section 3.3) and real experiments (section 5). Our software is available online at <http://ptak.felk.cvut.cz/sfmservice/methods/jancosek/cvpr-2011/index.html>.

Related work. Visual-hull was introduced by Laurentini in [16]. This technique relies on the ability to clearly separate objects from the background and therefore is not useful for reconstructing outdoor scenes.

Space carving [13] produces an approximate reconstruction called Photo-Hull, which is, under very strict conditions, guaranteed to subsume all other photo-consistent reconstructions. The assumptions are so strict that the method is useless for reconstructing outdoor scenes. It also does not reconstruct weakly-supported surfaces well since all non-photo-consistent volume is carved out. Photo-Flux was recently introduced in [3]. It is closely related to the Photo-Hull but allows to recover finer shape details without over-smoothing them while still handling noise robustly. In [17] they present an approach to multi-view image-based 3D reconstruction by statistically inverting the ray-tracing based image generation process. All these methods ([13, 3, 17]) use volumetric representation of the space. As the number of voxels depends on the resolution of the volume

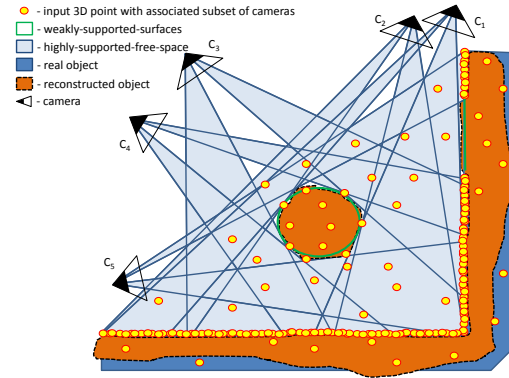


Figure 2. **Weakly-supported surfaces.** A highly-supported-free-space is the part of the space which is between the surfaces densely sampled by the input 3D points and the associated cameras. Weakly-supported surfaces (green) are the surfaces of the real object which are weakly sampled by the input 3D points and are close to the border of the union of the highly-supported-free-space cones.

cubically, their computational and memory cost quickly reaches computer limits. Therefore the volumetric methods are suited for reconstruction of small compact objects and are not scalable.

The latest state-of-the-art methods [8, 7] are focused on producing noise free oriented point clouds to generate 3D mesh using [12]. This approach is rather conservative and tends to reconstruct only those surfaces which are strongly supported by data. It leaves the space free where the support is not sufficient.

We build on the state-of-the art global method to 3D reconstruction [15]. It is based on the minimal s-t cut of a graph derived from the Delaunay tetrahedralization of the input 3D point cloud in order to label tetrahedra as inside or outside. This approach uses a strong geometrical prior to reconstruct many surfaces well but, unfortunately, it does not construct the weakly-supported surfaces well. This imperfection of [15] is illustrated in Figure 1(c), where the bottle has not been reconstructed. It shows how important it is to deal correctly with weakly supported surfaces. Figure 3(d) shows another example where the ground plane is missing completely.

2. The Base-line method

We build on work [15] which we call the *base-line method*. Let us assume that we have a Delaunay tetrahedralization of a point cloud where a set of cameras is associated to each point. Work [15] considers the surface reconstruction problem as a binary labelling problem. Tetrahedra are labelled as being inside or outside. The reconstructed surface is the union of oriented faces (triangles) of the Delaunay tetrahedralization which is guaranteed to bound a

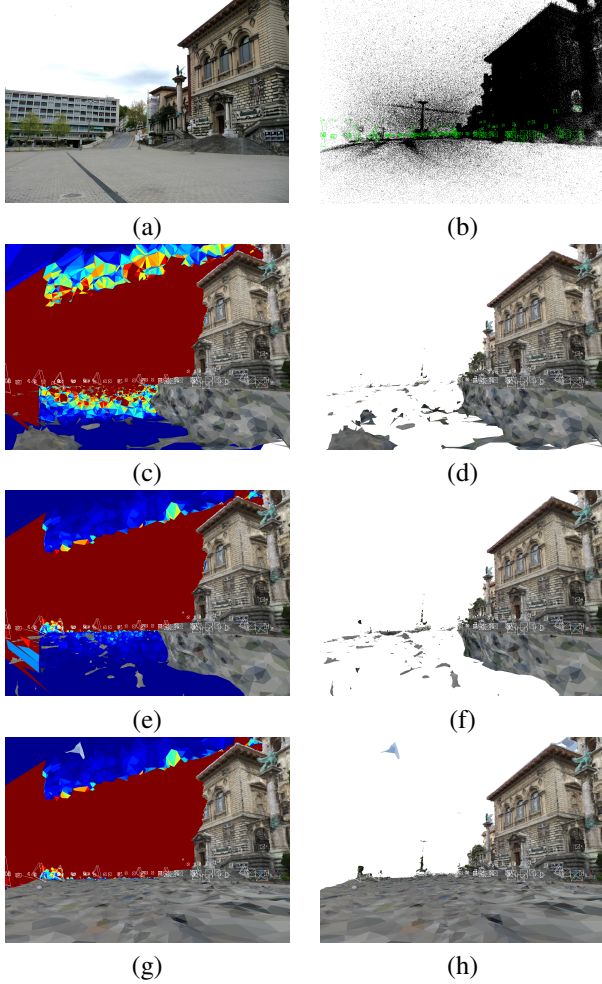


Figure 3. **Results for the 'strecha' data set.** (a) Input image, (b) input 3D point cloud, (c) the free-space-support of the tetrahedra on a cut of the tetrahedralization with a plane using $\alpha_{vis} = 32$ as in the base-line method, (e), (g) the same but (e) using $\alpha_{vis}(p)$ instead and (g) our method. The free-space-support is different. (d),(f) are reconstructions of our implementation of the base-line method, (d) using $\alpha_{vis} = 32$, (f) using $\alpha_{vis}(p)$. The results are similar. (h) is the result using our method. (d), (f) demonstrate that the introduction of $\alpha_{vis}(p)$ into the base-line method is not sufficient to reconstruct weakly-supported ground plane. (h), (g) shows that our method, reconstructs the weakly-supported ground plane. The jet-color-map was used for the range $(0, 1000)$, where the blue color corresponds to 0 and the dark red color corresponds to 1000.

volume, i.e. it is watertight and self non-intersecting. The surface separating the empty space from the full space is found as a minimal s-t cut of the following directed graph. The nodes of the graph correspond to the Delaunay tetrahedra and directed edges correspond to the oriented triangles between adjacent tetrahedra. Let us consider a face f between adjacent tetrahedra a and b viewed from the side of the tetrahedron a . The face f is represented in the graph

by the edge oriented from the node a to the node b . The graph is augmented with an additional source (representing the outside) and a sink (representing the inside) nodes and with edges from the source to each node (s-edges) and from each node to the sink (t-edges). Finally, the directed edges of the cut (going from the node labeled as source to the node labeled as sink) correspond to triangles on the reconstructed oriented surface.

Figure 4 shows how the weights of the edges in the graph are computed. Let us consider a point p and an associated camera centre c . Let us denote all tetrahedra (resp. faces), which intersect the line segment $(c, p + \frac{p-c}{|p-c|}\sigma)$ ordered by the distance of the intersection point to the c in ascending order, as *crossing-tetrahedra* (resp. *faces*).

The weights of all edges of the s-t graph are initially set to zero. For each point p and camera centre c assigned to p , weights of the edges of the *crossing-faces* (which are intersected by the line segment mentioned above) are increased by a constant value α_{vis} . The value of infinity is assigned to the first *crossing-tetrahedron* (tetrahedron i_1 in Figure 4). Finally, the α_{vis} value is added to the t-edge of the last *crossing-tetrahedron* (see Figure 4). See [15] for more details.

The weight of an oriented edge of the s-t graph is determined by the number of camera-point pairs for which the point is occluded by the face corresponding to the edge in the camera. A t-edge weight of a node is determined by the number of camera-point pairs for which the point occludes the corresponding tetrahedron in the cameras up to the depth σ behind the point: in other words the point projects to the area of the projection of the tetrahedron in the camera.

The free-space-support of the tetrahedron is the sum of weights of all incoming edges to the node corresponding to the tetrahedron. The free-space-support is related to the total number of pairs (camera, point) for which the point is occluded by the tetrahedron in the camera.

We have to point out that we are using different α_{vis} value for each 3D point p of the tetrahedralization in our method while in the base-line method the α_{vis} is set to a constant. We denote it as $\alpha_{vis}(p)$. The Delaunay tetrahedralization points are not the points of the input 3D point cloud X . Let us denote the number of cameras associated with a point $x \in X$ from the input 3D point cloud as $N_c(x)$. In our method we group the points ($x \in X$) from the input 3D point cloud into the points ($p \in T$) of the tetrahedralization such that $S(p) \subset X$ within a distance γ surrounding of the p is not empty and $\forall p_i, p_j \in T; |p_i - p_j| > \gamma$. The $\alpha_{vis}(p)$ is as follows.

$$\alpha_{vis}(p) = \sum_{x \in S(p)} N_c(x) \quad (1)$$

The $\alpha_{vis}(p)$ is not the same as $N_c(p)$, because more points associated with the same camera can be involved in

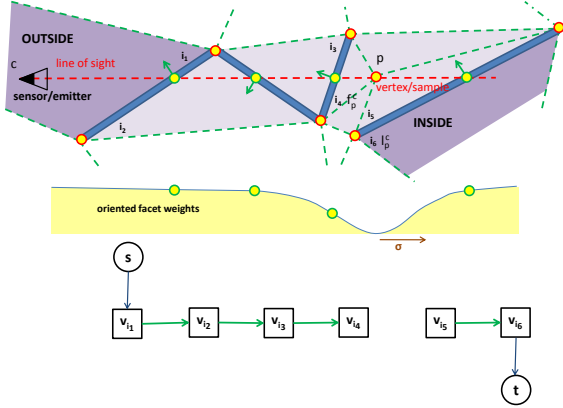


Figure 4. **Illustration of the s-t graph based surface reconstruction.** (Figure 4 (b) in [15]). Symbol f_p^c denotes the first tetrahedron i_4 which contains point p and is next to the nearest *crossing-face* of the line segment (c, p) . We denote the last tetrahedron i_6 which is in the σ distance from the point p crossing the directed line $(p, p - c)$ as i_p^c .

the $\alpha_{vis}(p)$. Therefore the value $\alpha_{vis}(p)$ should be small for a noisy point p because noisy points are usually created by a small number of cameras and have no other points within their γ -neighbourhood. In another words the good situation is when the point is supported from more cameras or from more 3D points. See section 4.1 for more details.

Figure 3 shows the influence of the $\alpha_{vis}(p)$ on computation the free-space-support values of the tetrahedra. Figure 3 (c) shows the free-space-support values of the tetrahedra on a cut of the tetrahedralization with a plane using $\alpha_{vis} = 32$ as in the base-line method and (e) using $\alpha_{vis}(p)$. The free-space-support is different. We refer the reader to the next sections to see the reason for using $\alpha_{vis}(p)$.

The problem of the base-line method is in that the formulation is not sufficient for obtaining weakly-supported surfaces. Figure 2 illustrates a situation when weakly-supported surfaces are present. This situation is demonstrated by a real world experiment in Figure 1. Figure 1(c) shows that the base-line method does not reconstruct weakly-supported surfaces (the bottle). Note that for relatively densely sampled surfaces, where the amount of noise is significantly lower than the density of surface samples, the base-line method works perfectly and our method gives exactly the same results.

3. Our approach

Section 3.1 describes why there should be a large free-space-support jump on the real surface. Section 3.2 introduces the weakly-supported surfaces t-weight assumption. While we cannot exactly prove that our approach works in general we prove that our method solves the problem on a synthetic example, (shown by Figure 6), section 3.3. We

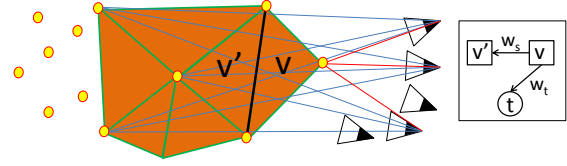


Figure 5. **Weakly-supported surfaces t-weight example.** The black face occludes 3 points in 3 cameras. Therefore the corresponding edge has weight $w_s = 9$. The corresponding tetrahedron v , which is nearest to the cameras is behind just one point, which has 3 cameras associated (and ray defined by the camera centre and the point intersects v). Hence the corresponding t-edge has weight $w_t = 3$. Furthermore, each point, which is occluded by the black face in c associated cameras, increases the w_s by the number c while w_t remains unchanged.

also experimentally observed that the example is representative and therefore our method should work in general.

3.1. Large free-space-support jump

The free-space-support of a tetrahedron which is near or lies on the real surface, and should be labeled as outside, should be much larger than free-space-support of a nearby tetrahedron which should be labeled as inside. This holds for surfaces which are densely sampled by the input point cloud as well as for weakly-supported surfaces. Figure 3(e) shows that it indeed holds. The free-space-support jump is large on both the densely sampled facade as well as on the weakly-supported ground plane.

3.2. Weakly-supported surfaces t-weight assumption

In the base-line method formulation, t-edge weights for tetrahedra, which are near or lie on the real but weakly-supported-surface and should be labeled as inside, are much smaller than the free-space-supports of the corresponding nodes. Let us assume that $\sigma = 0$ and denote a tetrahedron which lies on the real but weakly-supported-surface, and which should be labeled as inside, by v . In this situation, the t-edge weight w_t of the tetrahedron v depends on the number of cameras associated with the four points of the tetrahedron v . The free-space-support w_s of the tetrahedron v depends on the number of cameras of all points which are occluded (in the cameras) by the tetrahedron v . Therefore, even a small number of wrongly reconstructed points makes the value w_s much greater than the value w_t (see Figure 5). While we assumed $\sigma = 0$ the conclusion holds for small σ , too. The parameter σ can not be set to large value (see [15]).

3.3. Synthetic example

The top of Figure 6(a), (b) illustrates the situation when a weakly-supported-surface appears as a part of the tetrahedralization. The bottom represents the corresponding

graph. The middle represents the weights for the corresponding edges. Weights for edges between nodes are light blue and weights for t-edges are dark brown. There is a large free-space-support jump on the surface; w_{56} is much higher than w_{67} (w_{78}). In this example there holds: $\infty > w_{12} > \dots > w_{56} > w_{67} + t_6 > w_{78} + t_6 + t_7 > w_{89} + t_6 + t_7 + t_8 < t_6 + t_7 + t_8 + t_9$ and $w_{56} > w_{57}$. The minimal cut in this synthetic example is the cut illustrated by the red lines. The labelling for the minimal cut does not correspond to the correct solution.

The weakly-supported surfaces t-weight assumption holds; w_{67} is much higher than t_7 and all other t_i are approximately same as t_7 , Figure 6(a). This causes that the minimal cut will be further from the correct surface. This effect is demonstrated by the real experiment in Figure 3. The free-space-support jump is large near the weakly-supported ground plane, 3(c),(e) but using $\alpha_{vis}(p)$ alone in the base-line method is not sufficient to reconstruct the weakly-supported-ground plane, Figure 3(f). Therefore, in the place of a large jump, we multiply t_7 by a conveniently chosen x so that the cut described in Figure 6(b) becomes minimal and we get the correct labelling. One can see that if we set x to $w_{56} - w_{78}$, then we will achieve the correct solution. Our approach is sequential. We first compute all weights in the same way as the base-line approach. Then we search for all large jumps and multiply the corresponding t-edge weights as demonstrated in the example. This is supported experimentally in Figure 3(h) by the fact that our method only changes weights of t-edges as described by the synthetic example and it is sufficient to reconstruct the weakly-supported-ground plane.

It is clear that in situation where the amount of noise is significantly lower than the density of surface samples, t-edges weights of the tetrahedra which are in the σ distance to the real surface and are in the real object are greater than the free-space-supports of the corresponding nodes. Otherwise the base-line method would not give any result. Therefore in this situation our method gives exactly the same result.

4. Implementation details

Our multi-view reconstruction pipeline is divided into two modules. The first module creates a depth map for each camera. To compute the depth-maps, we reimplemented the approach to plane-sweeping proposed in [10]. A 3D point cloud can be computed for each depth-map using a related camera matrix. We collect all 3D point clouds from all depth-maps into one 3D point cloud, where the camera from which the point is computed is related to each 3D point. This point cloud is used as the input to the second module, which constructs a triangulated surface.

Initially, we create the tetrahedralization from the input point cloud using the Computational Geometry Algorithms

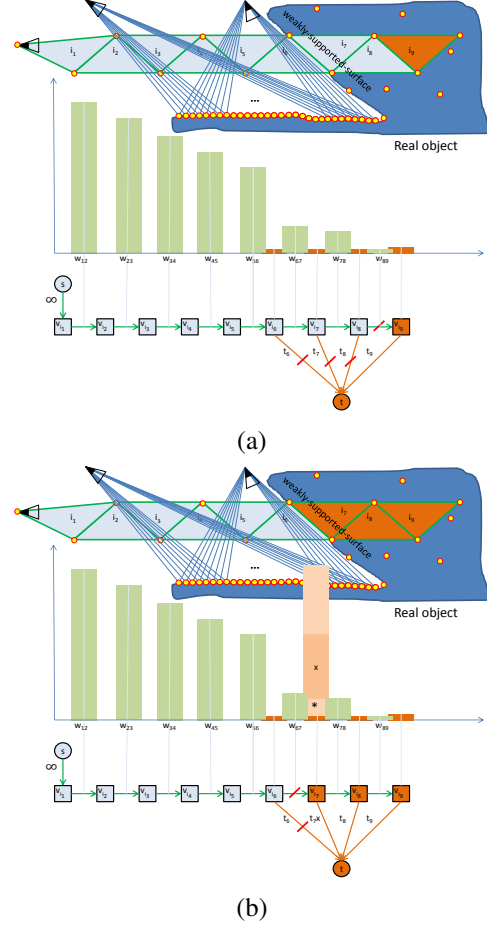


Figure 6. **Representative example.** Light blue: source label (free space), brown: sink label (full space). (a,b) top: a part of the triangulation, bottom: the associated s-t graph, middle: the weights of the associated edges. (a) Minimal cut for weights computed by the base-line approach leads to a wrong solution (b) Multiplying t_7 by $(w_{56} - w_{78})$ leads to the correct solution.

Library (CGAL)² [1], described in section 4.1. Then we build the s-t graph as described in section 2. Next we compute the weights of all edges as described in the section 4.2. Finally, we solve the minimal s-t cut problem using the software³ described in [2].

4.1. Creating tetrahedralization

The tetrahedralization is created online. When adding a new point x from the point cloud we first check if there is a 3D point p of the actual tetrahedralization within a distance γ from x . If there is such a point then we do not add x to the tetrahedralization but associate the camera with the nearest point p and increase $\alpha_{vis}(p)$ by 1. Otherwise we add the point p to the tetrahedralization, associate the camera with it, and initialize an $\alpha_{vis}(p)$ to 1. The approach is similar to

²<http://www.cgal.org/>

³<http://www.adastral.ucl.ac.uk/vladkolm/software.html>

[14]. As a consequence, each point has a set of cameras and the $\alpha_{vis}(p)$ value associated.

4.2. Setting up the weights

We denote the first tetrahedron which contains point p and is related to the nearest *crossing-face* of the line segment (c, p) as f_p^c . We denote the last tetrahedron which is within distance σ from the point p crossing the directed line $(p, p - c)$ as l_p^c . See Figure 4.

Our approach to setting up the weights of the graph is sequential. In the first step, we compute all weights of all edges except weights of the t-edges in exactly the same way as in the base-line method but instead of α_{vis} we use $\alpha_{vis}(p)$. In the second step, we compute weights of the t-edges which are updated in a similar way to the base-line method, but with different weights. For each point p and each associated camera c we take the free-space-support $e(f_p^c)$ of the f_p^c tetrahedron and the free-space-support $e(l_p^c)$ of the l_p^c tetrahedron. And we add the value $t(c, p)$

$$t(c, p) = \begin{cases} \alpha_{vis}(p)(e(f_p^c) - e(l_p^c)) \frac{e(l_p^c)}{e(f_p^c)} < \delta \wedge e(f_p^c) < \beta \\ \alpha_{vis}(p) \frac{e(l_p^c)}{e(f_p^c)} \geq \delta \vee e(f_p^c) \geq \beta \end{cases} \quad (2)$$

to the t-edge of the l_p^c tetrahedron. The parameter δ determines that there is a large jump i.e. when there should be a surface. We use very conservative approach and set $\delta = 0.5, \beta = 1000$ in all of our experiments.

4.3. Performance discussion

The base-line algorithm (i) iterates through n points of tetrahedra. For each i^{th} point it then (ii) iterates through c_i associated cameras and for each j^{th} camera it then (iii) iterates through the *crossing-tetrahedra* of the ray from the j^{th} camera centre up to the σ depth behind the i^{th} point. We denote the average number of each of these three iteration steps as $n_p = n$, n_c and n_t . Therefore, the computation processing time of the overall graph weights is $O(n_p n_c n_t)$.

In the first step of our algorithm (section 4.2), we do the same number of iterations but also remember for each point p and camera c the last tetrahedron l_p^c . The speed is the same as the speed of the base-line algorithm $O(n_p n_c n_t)$. In the second step (section 4.2), our algorithm iterates through n points of tetrahedralization. For each i^{th} point it iterates through c_i associated cameras and for each j^{th} camera it just update the t-weight of the remembered l_p^c tetrahedron according to equation 2. The number of iterations in the second step is $O(n_p n_c)$. It is much faster than the first step because it does not need to do the final iteration (iii) through tetrahedra for each ray. According to n_t in the table 1, the second step is several hundred times faster than the first one. This shows that the speed of our approach is almost the same as the speed of the base-line approach. Table 2

data set	$\#_c$	n_c	n_p	$\#_t$	n_t
castle	30	3	1, 2M	7, 6M	255
dragon	114	4	3M	20M	538

Table 1. **Performance data for different data sets.** $\#_c$ is the number of input cameras, n_c is the average number of cameras associated with a point, n_p is the number of points in the tratrahedralization, $\#_t$ is the number of tetrahedra in the tratrahedralization, n_t is the average number of tetrahedra crossing the line segment from a camera to the σ depth behind a point (for each pair (point, associated camera)).

data set	t_l	t_{o_1}	t_{o_2}
castle	30	30	2
dragon	90	91	3

Table 2. **Running times of different parts of the algorithm for different data sets.** t_l is the time of filling the graph using our implementation of the base-line method in minutes, t_{o_1} is the time of the first iteration of our method and t_{o_2} is the time of the second one (in minutes).

shows the running times of different parts of the algorithm for different data sets.

5. Results

All experiments were computed on Intel Core i7 CPU machine with nVidia 285GTX graphics card, 12GB RAM and 64-bit Windows 7 OS. We use four different calibrated image sets (data sets) in this paper: bottle, strecha, castle and dragon. The bottle data set contains 24 1950×1307 images. The street-view data set 'strecha' was provided by Christopher Strecha and contains 1514 2000×1500 images. We compute the depth map for each of the 1500 images but to reconstruct different parts of the city we choose 3D points from different volumes parts defined by boxes in order to fit into memory. The castle data set is the benchmark data set [20] and contains 30 3072×2048 . The dragon data set was used in [10] and contains 114 1936×1296 images. We use the same α_{vis} and σ as in the base-line method. The parameter δ is set to 0.5 in all of our experiments. The parameter γ is set in terms of the two points re-projection pixels distance to a value in the range $\langle 2, 10 \rangle$ in order to be able to fit the data into memory.

We have to note that as our method produces more complete scenes than the base-line method but it tends to produce more hallucinations. We use the approach described in [11] to remove them.

Figures 1, 3, 8, 9, 10, 7 demonstrate that our method reconstructs weakly-supported surfaces better than the base-line method. The weakly-supported surface in the Figure 1(a) is the bottle. The weakly-supported surface in the date-sets 3, 8, 9, 10 is mostly the ground plane.

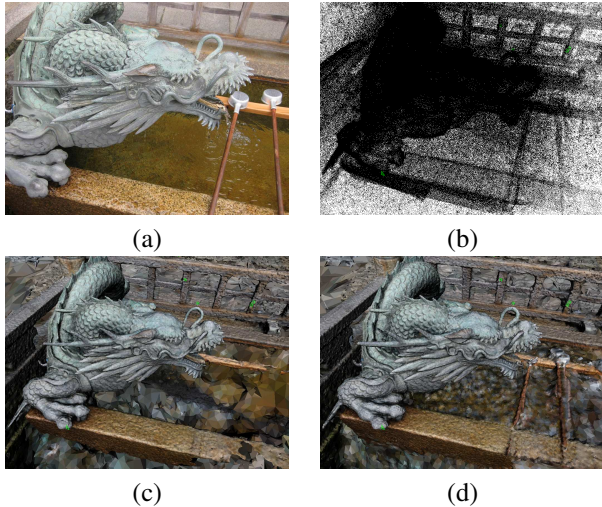


Figure 7. **Results for the 'dragon' data set.** (a) Input image, (b) input 3D point cloud, (c) result of our implementation of the base-line method, (d) result using our method. Our method reconstructs densely sampled surfaces at the same level of detail as the base-line method but also reconstructs weakly-supported surfaces i.e. water, and ladles.

Figure 7 demonstrates that our method reconstructs densely sampled surfaces at the same level of detail as the base-line method. Additionally, our method reconstructed weakly-supported surfaces i.e. water, and ladles. Note that the ladles were not at the same place in all images. Figure 10 presents results on the castle data set from the standard Strecha's [20] evaluation database. The histograms (c), (d) show that our reconstruction achieves almost the same quality as the method [21] which uses an additional mesh refinement step. Figure 10 (e), (f) shows that, unlike the base-line method (f), our method (e) can reconstruct weakly supported ground planes. Additionally, the histograms (c), (d) show that our reconstructions are more or less on the same level at 2σ and 3σ as the base-line method which means that our method produces results on the same level of detail as the base-line method.

6. Conclusion

We have presented a new multi-view reconstruction method. First, a depth-map for each camera is computed and a 3D point cloud is generated from the depth-maps. The Delaunay tetrahedralization of the 3D point cloud is then generated. The aim is to label the tetrahedra as empty or occupied by finding the minimal s-t cut of a corresponding graph. Our modification of the graph weights leads to reconstructing weakly-supported surfaces robustly while achieving the same quality on other surfaces. We have shown that the speed of our method is almost the same as the speed of our implementation of the base-line method. We have demonstrated that our method achieves better re-

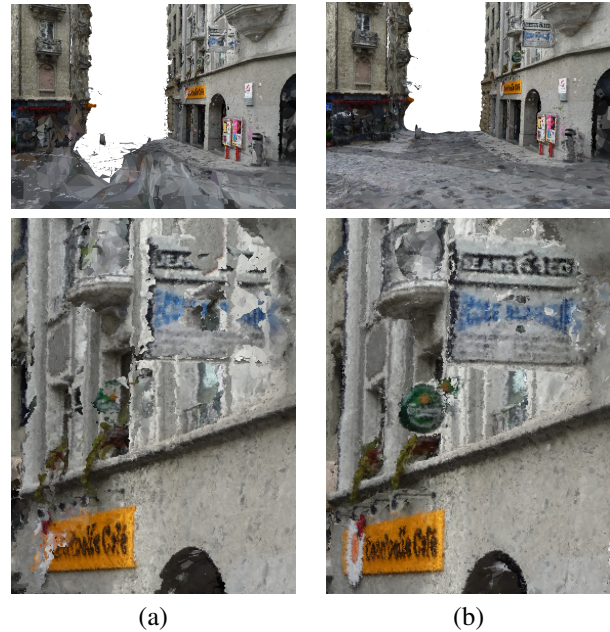


Figure 8. **Results for the 'strecha' data set.** (a) reconstruction using our implementation of the base-line approach, (b) reconstruction using our approach. Weakly-supported surfaces are better reconstructed using our approach than by base-line approach.

sults on several real data sets.

References

- [1] CGAL, Computational Geometry Algorithms Library. <http://www.cgal.org>.
- [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] Y. Boykov and V. Lempitsky. From photohulls to photoflux optimization. In *BMVC*, 2006.
- [4] D. Bradley, T. Boubekeur, and W. Heidrich. Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *CVPR*, 2008.
- [5] N. Campbell, G. Vogiatzis, C. Hernandez, and R. Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*, 2008.
- [6] R. Collins. A space-sweep approach to true multi-image matching. Technical Report UM-CS-1995-101, 1995.
- [7] Y. Furukawa, B. Curless, S. Seitz, and R. Szeliski. Towards internet-scale multi-view stereo. In *CVPR*, 2010.
- [8] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. In *CVPR*, 2007.
- [9] M. Goesele, N. Snavely, B. Curless, H. Hoppe, and S. Seitz. Multi-view stereo for community photo collections. In *ICCV*, 2007.
- [10] M. Jancosek and T. Pajdla. Hallucination-free multi-view stereo. In *RMLE*, 2010.
- [11] M. Jancosek and T. Pajdla. Removing hallucinations from 3D reconstructions. Technical Report CMP CTU, 2011.

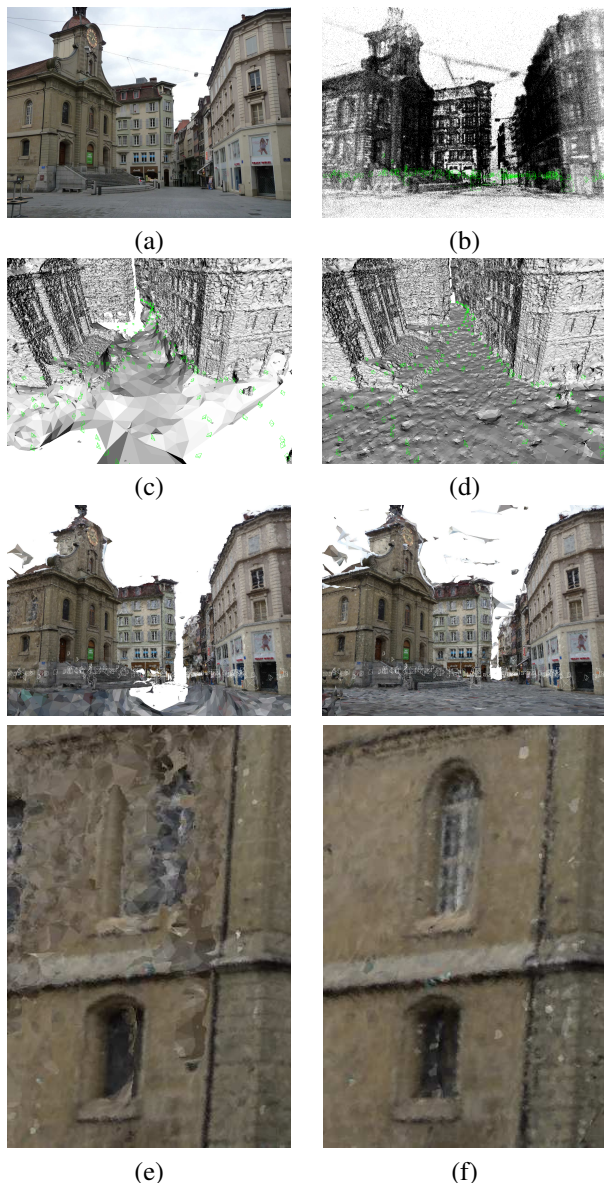


Figure 9. **Results for the 'strecha' data set.** (a) Input image, (b) input 3D point cloud, (c) result of the our implementation of the base-line method untextured, (d) result using our method untextured, (e) result of the our implementation of the base-line method textured, (f) result using our method textured.

- [12] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *SGP*, pages 61–70, 2006.
- [13] K. Kutulakos and S. Seitz. A theory of shape by space carving. Technical Report TR692, 1998.
- [14] P. Labatut, J. Pons, and R. Keriven. Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts. In *ICCV*, 2007.
- [15] P. Labatut, J. Pons, and R. Keriven. Robust and efficient surface reconstruction from range data. In *Computer Graphics Forum*, 2009.

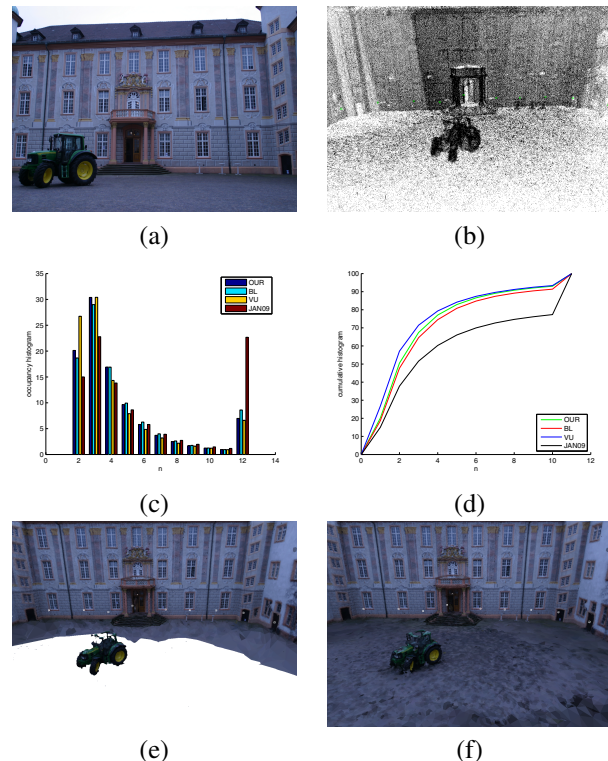


Figure 10. **Results for the 'castle' data set.** (a) Input image, (b) input 3D point cloud, (c),(d) Strecha's evaluation [20] for the Castle-P30 data sets. OUR - this paper, BL - our implementation of the base-line method [15], VU [21], JAN09 [14]. (c) Histograms of the relative error with respect to $n\sigma$ for all views. The σ is determined from reference data by simulating the process of measurement and can vary across the surface and views, (d) relative error cumulated histograms, (e) result of our implementation of the base-line method textured, (f) result using our method textured.

- [16] A. Laurentini. The visual hull concept for silhouette-based image understanding. *PAMI*, 16(2):150–162, February 1994.
- [17] S. Liu and D.-B. Cooper. Ray markov random fields for image-based 3d modeling: Model and efficient inference. In *CVPR*, 2010.
- [18] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006.
- [19] C. Strecha, R. Fransens, and L. Van Gool. Wide-baseline stereo from multiple views: a probabilistic account. In *CVPR*, 2004.
- [20] C. Strecha, W. von Hansen, L. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *CVPR*, 2008.
- [21] H. Vu, R. Keriven, P. Labatut, and J.-P. Pons. Towards high-resolution large-scale multi-view stereo. In *CVPR*, 2009.
- [22] A. Zaharescu, E. Boyer, and R. Horaud. Transformesh: A topology-adaptive mesh-based approach to surface evolution. In *ACCV*, 2007.