Incremental Surface Extraction from Sparse Structure-from-Motion Point Clouds

Christof Hoppe[1]
hoppe@icg.tugraz.at
Manfred Klopschitz[2]
manfred.klopschitz@siemens.com
Michael Donoser[1]
michael.donoser@tugraz.at
Horst Bischof[1]
bischof@icg.tugraz.at

[1] Institute for Computer Graphics and Vision,
Graz University of Technology,
Graz, Austria
[2] Imaging and Computer Vision
Research Group Video Analytics
Corporate Technology
Siemens AG Austria, Graz

Abstract

Extracting surfaces from a sparse 3D point cloud in real-time can be beneficial for many applications that are based on Simultaneous Localization and Mapping (SLAM) like occlusion handling or path planning. However, this is a complex task since the sparse point cloud is noisy, irregularly sampled and growing over time. In this paper, we propose a new method based on an optimal labeling of an incrementally reconstructed tetrahedralized point cloud. We propose a new sub-modular energy function that extracts the surfaces with the same accuracy as state-of-the-art with reduced computation time. Furthermore, our energy function can be easily adapted to additional 3D points and incrementally minimized using the dynamic graph cut in an efficient manner. In such a way, we are able to integrate several hundreds of 3D points per second while being largely independent from the overall scene size and therefore our novel method is suited for real-time SLAM applications.

1 Introduction

The accurate reconstruction of large-scale 3D models of urban scenes is nowadays possible within a few hours due to the recent advances in Structure-From-Motion (SfM) research [1, 9, 16]. Mostly a sparse 3D point cloud is provided as final representation. Versatile applications like visual navigation, image based localization or visual tracking in augmented reality (AR) applications directly exploit the 3D point cloud, e. g. by precisely determining the camera pose in real-time. However, many other tasks like occlusion handling in AR, path planning in robotics or the plausible visualization of the current 3D reconstruction additionally require a surface model of the environment.

Extracting surfaces from the 3D point cloud is a complex problem because the density of the points is highly irregular and perturbed by outliers. Existing solutions often either assume a densely, regularly sampled surface [10] or make use of additional knowledge like visibility information [13].

The problem gets even harder if the surface has to be extracted in an incremental manner, as for example required in Simultaneous Localization and Mapping applications (SLAM) [4, 5], where additional scene information is consecutively provided. Such methods have to handle an increasing amount of data in real-time, which means that several hundred points have to be integrated into the surface per second. The state-of-the-art in incremental surface reconstruction such as [7] makes heavy use of powerful GPGPU units which are often not available in application areas like robotics or AR. Furthermore, these approaches represent the scene in an equally discretized voxel space and as a consequence are restricted to a limited scene size.

The basic principle of most existing methods for extracting surfaces from sparse SfM point clouds is the binary labeling of a discretized space into *free* and *occupied* using the visibility information. The surface is then extracted as the interface between free- and occupied space. Many approaches choose an irregular discretization of the space by performing a Delaunay triangulation (DT) [2] on the 3D points. The advantage of the DT is that the size of the discretized units is related to the density of the underlying point cloud and the DT can be efficiently adapted to new 3D information. Existing approaches mainly differ in the way they classify the space as *free* or *occupied*. Pan et al. [15] perform a probabilistic space carving, i. e. all tetrahdra intersected by a ray are labeled as free space and all others are occupied. Their method is sensitive to outliers and is constrained to batch processing i. e. if new points are added the full space carving has to be performed from scratch. Lovi et al. [14] extend the space carving approach to work in an incremental manner but their method requires additional memory that grows with the scene size. Lhuillier et al. [19] present a method that incrementally extracts a surface from sparse points generated by a visual odometry system. Instead of carving free-space tetrahedra, they greedily aggregate free-space tetrahedra and the outer envelope defines the surface. However, this approach can handle only pure forward motion. Labatut et al. [13] formulate the labeling problem as a Markov Random Field. They define an energy function based on visibility information and surface parameters and minimize this energy by graph cuts. Their method is robust against outliers and therefore results in a clean and smooth surface mesh.

To sum up, the aforementioned approaches for incremental surface reconstruction are either based on a strong camera motion assumption [19] or they are not robust to outliers [14, 15]. By contrast, robust methods like [13] are not well suited to be implemented in an incremental manner as we show later.

In this paper we propose a new method to incrementally extract a surface from a consecutively growing SfM point cloud in real-time. Our method is based on a Delaunay triangulation (DT) on the 3D points. The core idea is to robustly label the tetrahedra into free- and occupied space using a random field formulation and to extract the surface as the interface between differently labeled tetrahedra. Therefore, we propose a new energy function that achieves the same accuracy as state-of-the-art methods but reduces the computational effort significantly. Furthermore, our new formulation allows us to extract the surface in an incremental manner, i. e. whenever the point cloud is updated, we adapt our energy function. Instead of minimizing the updated energy with a standard graph cut, we employ the dynamic graph cut of Kohli et al. [12] which allows an efficient minimization of a series of *similar* random fields by re-using the previous solution. The combination of the dynamic graph cut with our new formulation allows us to extract the surface from an increasingly growing point cloud nearly independent of the overall scene size. In the experiments, we compare our approach to the state-of-the-art for static surface extraction from point clouds and show that we achieve the same quality, while reducing the computational effort by more than 50%.

We furthermore demonstrate that our method is able to robustly extract the surface from an increasingly growing sparse point cloud in real-time and show that our approach used in an incremental manner is up to 20 times faster than the state-of-the-art.

2 Energy Function for Surface Extraction

Our method for extracting a surface from a sparse SfM point cloud is motivated by the truncated signed distance function (TSDF), which is known from voxel-based surface reconstructions like [7, 20]. The TSDF models for all voxels along the ray connecting the camera and a 3D point X their probability of being free space or occupied. Typically, this information is aggregated for a large number of 3D points obtained by several dense depth maps, where the resulting surface is then extracted as the zero crossing within the volume exploiting inherent redundancy.

By contrast, when using sparse points as in our intended application field, redundancy is limited and an extraction of the surface by finding the zero crossing is not possible. Therefore, our main idea is that given the tetrahedralized point cloud, we formulate surface extraction as a binary labeling problem, with the goal of assigning each tetrahedron either a *free*- or *occupied* label. For this reason, we model the probabilities that a tetrahedron is free- or occupied space analyzing the entire available visibility information $\mathcal{R}$, which consists of the set of rays that connect all 3D points to image features. Following the idea of the TSDF, a tetrahedron in front of a point X has a high probability to be free space, whereas the tetrahedron behind X is presumably occupied space. We further assume that it is very unlikely that neighboring tetrahedra obtain different labels, except for tetrahedra close to a point X. Such a labeling problem can be elegantly formulated as a pairwise random field.

Formally, given a set of tetrahedra V obtained by the Delaunay triangulation (DT) of the point cloud, we define a random field where the random variables are the tetrahedra of V. Our goal is to identify the binary labels $\mathcal{L}$ that give the maximum a posteriori (MAP) solution for our random field, analyzing the provided visibility information $\mathcal{R}$. The binary labels specify if a certain tetrahedron $V_i \in V$ is free- or occupied space. To identify the optimal labels $\mathcal{L}$, we define a standard pairwise energy function

$$E(\mathcal{L}) = \quad \sum_i (E_u(V_i, \mathcal{R}_i) + \sum_{j \in \mathcal{N}_i} E_b(V_i, V_j, \mathcal{R}_i)) \,, \tag{1}$$

where $\mathcal{N}_i$ is the set of the four neighboring tetrahedra of the tetrahedron V_i and $\mathcal{R}_i$ is a subset of $\mathcal{R}$, consisting of all rays connected to the vertices that span V_i.

For defining the unary costs $E_u(V_i, \mathcal{R}_i)$ we follow the idea of the TSDF that the probability that a certain tetrahedron V_i is free space is high, if many rays of $\mathcal{R}_i$ pass through V_i. Therefore, we set costs for labeling V_i as occupied space to $n_f \alpha_{free}$, where n_f is the the number of rays of $\mathcal{R}_i$ that pass through V_i. In contrast if V_i is located in extend of many rays of $\mathcal{R}_i$ the probability is high that V_i is occupied space. For this reason, the costs for labeling V_i as free space are set to $n_o \alpha_{occ}$, where n_o is the number of rays in front of V_i. Figure 1(a) illustrates the unary costs for a small example. Here, n_f is 1 since only the light green ray passes V_i and n_o is 3 because V_i is in extend of the three green rays. The red rays do not contribute to the unary costs.

For the pairwise terms we assume that it is very unlikely that neighboring tetrahedra obtain different labels, except for pairs (V_i, V_j) that have a ray through the triangle connecting both. Let R_k be a ray of $\mathcal{R}_i$ that passes V_i. If R_k intersects the triangle (V_i, V_j), $E_b(V_i, V_j, \mathcal{R}_i)$ is set to β_{vis}. Triangles (V_i, V_j) that are not intersected by any ray of $\mathcal{R}_i$ are set to β_{init}. Figure 1(b) shows the pairwise costs in an example.

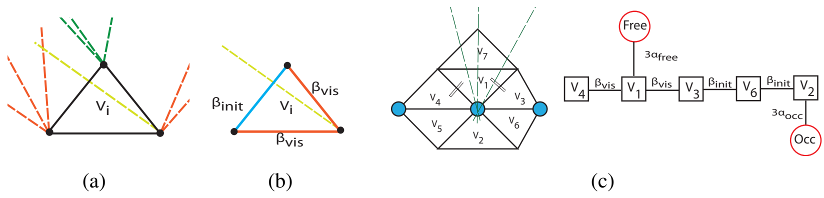


Figure 1: (a) For defining the unary term for a specific tetrahedron V_i we only analyze rays (dashed lines) connected to vertices that span V_i. (b) For the pairwise term we only consider rays that pass through the tetrahedron and that are connected to the tetrahedron vertices. (c) Graph representation of the energy function. The pairwise weights that are not shown are set to β_{init}.

Figure 1(c) visualizes the graphical model of the energy function for a small example. Since V_1 is passed by three rays, the costs for labeling V_1 *free* is set to $3\alpha_{free}$. In contrast, V_2 is in extend of three rays and therefore V_2 is connected to the *occupied* node with the weight $3\alpha_{occ}$. The edge weights between all neighboring tetrahdra are set to β_{init} except the edges (V_1, V_4) and (V_1, V_3) which are set to β_{vis}.

Having defined all terms for our random field formulation, we are then able to derive a global optimal labeling solution for our surface extraction problem using standard graph cuts since our priors are submodular.

At a first glance, our energy seems to be similar to the visibility part of the energy defined by Labatut et al. [13]. The major difference is the de3finition of the pairwise costs which has a large impact on the computational complexity when adapting the energy to a new DT structure. Labatut et al. initialize the pairwise costs with a low value and increase the costs if an arbitrary ray $R_n \in R$ intersects V_i as well as V_j, i. e. if R_n intersects the triangle between V_i and V_j. Therefore, the pairwise costs are not restricted to local visibility around V_i but may depend on the global distribution of the rays. This might drastically increase the computational complexity for updating the energy to a new DT structure, although only a small part of the DT has changed as we demonstrate in the experiments in Section 4.

3 Incremental Surface Extraction

To enable an efficient incremental surface reconstruction, our method has to consecutively integrate new scene information (3D points as well as visibility information) in the energy function and to repeatedly find the optimal labeling. In this section, we first show how the energy terms are updated and second, how the modified optimization problem can be efficiently solved in an incremental manner using the dynamic graph cut.

3.1 Energy Update

The energy function $E(\mathcal{L})$ depends on the structure of the DT and the visibility information $\mathcal{R}$ and therefore has to be updated if either the DT structure changes or new visibility information becomes available. First, we describe the energy update from $E_n(\mathcal{L})$ at time n to the new energy $E_{n+1}(\mathcal{L})$ if new visibility information is available followed by the description how the energy is adapted to a modified DT structure.

Visibility update. The integration of new visibility, i. e. a new ray R_k is added, affects only the tetrahedra next to the 3D point the ray is connected to. To update the unary costs, we determine the tetrahedra V_f and V_b that are located in front and behind the 3D point with

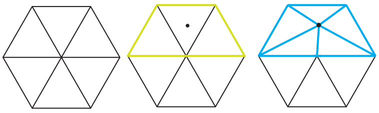


Figure 2: Insertion of a new point in the upper triangle changes the DT. Triangles within the change boundary $\mathcal{A}$ (green lines) are destroyed and the blue triangles are created.

respect to the ray direction respectively and add the costs α_{free} and α_{occ} to our energy. Since the destination of R_k is a point of the DT, V_f and V_b can be efficiently found as follows. We slightly shift the destination according to the ray direction and test in which tetrahedron the shifted point is located. For the pairwise term, we additionally determine the faces of V_f that are not intersected by R_k and set their costs to β_{vis}.

Since the integration of new visibility does not affect the structure of the DT, the number of terms in the energy stays constant and only a few terms are changed: two terms of the unary costs and three terms of the pairwise costs which are the faces of V_f that are not intersected by R_k. Hence, in contrast to space carving algorithms the integration of new visibility information is independent of the number of tetrahedra intersected by the ray.

DT update. The energy function has to be adapted if the DT structure changes, i. e. whenever a new 3D point is added, removed or shifted. Typically, the modification of a single point (usually) only effects a local area $\mathcal{A}$, i. e. some tetrahedra are deleted and new tetrahedra are created. Technically, all tetrahedra within $\mathcal{A}$ are destroyed and the DT is re-triangulated for the points in $\mathcal{A}$ (see Figure 2). Consequently, we remove all terms from $E_n(\mathcal{L})$ that are related to deleted tetrahedra and add costs for new tetrahedra. The costs for the new terms are updated as explained before for the visibility update.

The complexity for adapting the energy $E_n(\mathcal{L})$ to a new DT structure depends on the number of rays connected to 3D points located in $\mathcal{A}$. Assuming N 3D points are located in $\mathcal{A}$ and each is connected to M rays on average, the complexity is $N \times M$.

3.2 Surface Extraction

In order to extract the surface after updating $E_n(\mathcal{L})$ to $E_{n+m}(\mathcal{L})$, we have to solve the minimization problem again. Static graph cuts like [3] are designed to solve a random field only once. For this reason, if we want to directly use [3] we would have to re-build the graph for each energy $E_n(\mathcal{L})$ and repeatedly solve the minimization problem from scratch, where the runtime for finding an optimal solution grows linearly with the number of terms. Although the overall problem size grows over time, the energies $E_n(\mathcal{L})$ and $E_{n+m}(\mathcal{L})$ typically differ only by a few terms. Kohli et al. [12] proposed a dynamic graph cut for such problems where a sequence of energy minimization problems has to be solved and the corresponding energy functions only differ by a few terms. The complexity for updating the weights in the graph is linear in the number of changed weights. In our case, also the time for optimization depends on the number of changed terms and therefore on average is independent of the overall scene size. This property combined with our fast adaption of the energy function to new 3D points and visibility information as described in Section 3.1 allows us a surface extraction in real-time independent of the overall scene size.

We start with the set of initial tetrahedra V_{init} obtained from the DT of the point cloud P_{init}. We setup the energy $E_0(\mathcal{L})$ according to Section 3.1 and minimize $E_0(\mathcal{L})$ with the graph cut algorithm of [3]. We then extract the triangular surface mesh by finding all pairs of tetrahdra (V_i, V_j) where V_i and V_j are labeled differently. Finally, we smooth the resulting

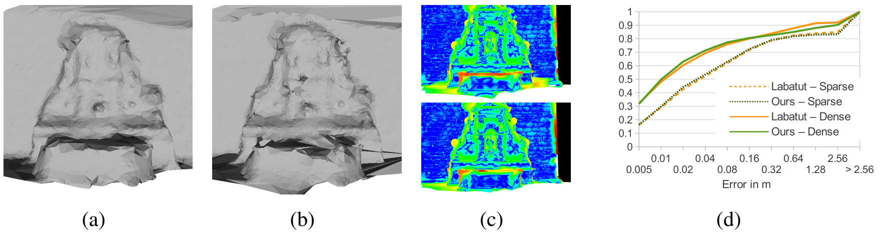


Figure 3: Fountain mesh. (a) Mesh extracted by our approach. (b) Surface extracted by [13] using only the visibility term of their energy function. (c) Color coded depth map error of image 6, where on the top is the error of [13] and at the bottom is the error of our approach. Blue indicates an error of less than 5 mm whereas distances above 2.56 m are coded in red (best viewed in color).

mesh using a Laplacian kernel [8]. For each new 3D point, we first update the DT and the energy function and then integrate the new visibility information. Finally, we solve the labeling problem for the new function $E_{n+m}(\mathcal{L})$ by the dynamic graph cut [12]. Typically, we integrate several new points with their visibility information into the energy before solving the minimization, i. e. m is between 500 and 2000, dependent on the user requirements.

4 Experiments

We first show in Section 4.1 that our proposed energy function reaches the same quality as the computational more complex state-of-the-art function proposed in [13]. We then demonstrate in Section 4.2 that our formulation is suited to incrementally extract a surface from an increasingly growing point cloud in real-time and that the complexity typically is independent from the overall scene size. For all experiments we set the costs as follows: $\alpha_{free} = 10^3$, $\alpha_{occ} = 10^3$, $\beta_{init} = 10^3$ and $\beta_{vis} = 10^{-3}$. For the optimization we use the dynamic graph cut implementation of [12] and for the DT, we use the CGAL [18] software package because it reports which tetrahedra are deleted and created due to the insertion of a new point.

4.1 Comparison to State-of-the-Art

In this experiment, we show that our novel energy achieves the same accuracy on sparse as well as on dense SfM point clouds as the more complex energy of Labatut et al. [13]. For accuracy evaluation, we use the dataset *Fountain* provided by Strecha et al. [17]. The dataset provides 11 high-resolution images and ground truth for camera positions and depth maps for each image. The sparse reconstruction is performed by an approach similar to Bundler [16] and results in 7 123 sparse 3D points. Each point is connected to 4.8 cameras on average.

We apply the surface extraction method of [13] as well as our proposed method on the provided data in a batch-based manner, i. e. we add all 3D points to the DT, setup and minimize the energy function only once. Figure 3(a) and 3(b) show the surfaces obtained by [13] and our method. Both the error maps and a visual comparison demonstrate that the surfaces are very similar. The accumulated histogram of depth map errors in Figure 3(d) quantifies the error in metric scale and gives evidence that both surfaces are very similar. We repeated this experiment with a densified point cloud obtained by PMVS2 [6] (370 000 points). The two upper curves in Figure 3(d) again show that the extracted surfaces are very similar.

On a second dataset we compare our result to [13] as well as to raycasting. This dataset consists of 77 300 3D points each connected to 4.4 rays on average but also around 20%

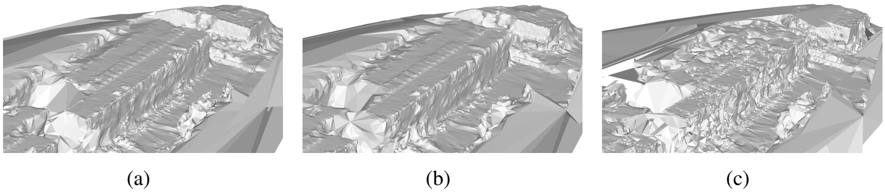


Figure 4: (a) Surface extracted by the method of Labatut et al. [13] (b) Ours. (c) Raycasting.

of the points are triangulated by only two image features and therefore contain significant noise. Figure 4(c) shows that raycasting yields a noisy surface and hence is not suitable for such data. In contrast the surfaces obtained by our method and [13] are nearly identical (Figure 4(a) and 4(b)) but the computational complexity is very different: [13] requires 79 seconds for defining their energy function and solving it by graph cuts, whereas our approach needs only 32 seconds on a Intel Core i7-960 processor. The difference in computational effort is mainly caused by the definition of the energy function. While [13] has to perform a full raycast for each ray, we only have to identify the tetrahedra in front and behind the vertex and the first triangle that is intersected by the ray. Furthermore, our energy can be solved faster by the graph cut. While the optimization of [13] requires 740 ms, our energy is fully optimized in 430 ms.

Note that beside visibility information Labatut et al. [13] also include two further terms, a photo consistency and a smoothness term. Both can be integrated into our energy function without violating the incremental fashion of our method since they only depend on triangle properties of neighboring tetrahedra.

4.2 Incremental Surface Reconstruction

In this experiment, we investigate the computational complexity of our proposed method in an incremental scenario. Similar to SLAM applications, we incrementally add new 3D points and visibility information and update the surface mesh after the integration of several hundred points. We determine the surface of two reconstructions that both consist of around 70 000 3D points. The first sequence was acquired by a Micro Aerial Vehicle showing an elongated building of 200m length. The second scene shows a medieval entrance where two figures are integrated into the wall.

We initialize our method with 1 000 3D points, calculate the unary and pairwise costs, extract the surface and incrementally add new points according to their creation time within the SfM pipeline. Our energy is updated each time a new 3D point is added to the DT and optimized after a defined number of points, e. g. 10 000 points, have been added. Figure 5 shows the surface of the building at different points in time after the integration of 40 000

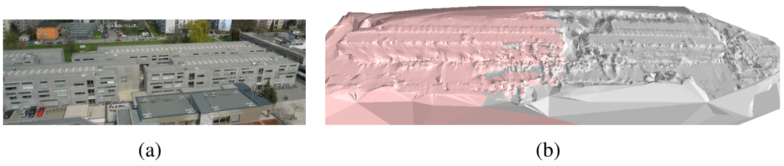


Figure 5: Incremental surface extraction over time. (a) Overview image of the reconstructed scene. (b) Reconstruction obtained at two different points in time. The gray part has been extracted from 40 000 3D points while additional 20 000 points create the red part of the reconstruction (best viewed in color).

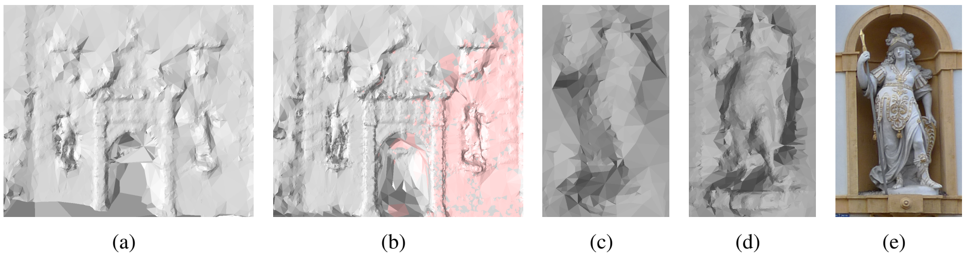


Figure 6: Surface evolution of the entrance sequence after the integration of (a) 10 000 and (b) 40 000 points. Consecutively added 3D points incrementally increase the quality of the right figure. (c) and (d) Cut-out of the right figure. (e) Right figure.

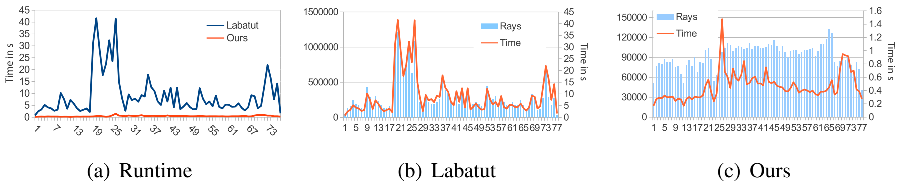
and 60 000 points respectively. The red marked triangles indicate the part of the surface that has been changed since the last optimization. Since the images are recorded by a camera with forward motion, we can observe the growing of the surface over time.

The acquisition ordering of the second scene is quite different. Here, the photographer started the image acquisition with overview images and then took more detailed views of the two figures. This sequence demonstrates that our approach makes no assumption about the camera motion and is able to refine parts of an already extracted surface (Figure 6). After 10 000 points, only the basic structure of the scene is observable. With the integration of more and more sparse points at the figures, the details become more and more visible.

In our approach we assume that camera positions and 3D points are fixed and not modified after the insertion. When integrating our method into a keyframe-based SLAM system like PTAM [11] which uses local bundle adjustment for map optimization this assumption may be violated. To attenuate this problem a *late* integration step can be implemented, i. e. new 3D points are not integrated into the mesh directly after their triangulation but at the time when they have been optimized several times by local bundle adjustment. This decreases the probability that the structure is drastically changed.

For the evaluation of the computational complexity, we compare our approach to an incremental implementation of [13]. Since such an implementation is not yet available, we combine their method with the incremental space carving approach of [14]. We store for each tetrahedron a list of rays that pass through it. When the DT is changed, we update the energy function of [13] and minimize the new energy with the dynamic graph cut. For adapting the energy to a new DT, we have to intersect all rays going through deleted tetrahdra with all new created tetrahdra which is computationally expensive. Furthermore, we have to store the ray to tetrahdra assignment which requires a large amount of memory.

The incremental adaptation of [13] and our method both consist of basically two parts: Update of the energy function according to new 3D points and the optimization using the dynamic graph cut. Figure 7 quantifies the complexity difference when updating the energy to a changed DT for the building sequence. The blue bars show the number of rays that are involved in updating the energy of 1 000 points. In our approach, we have to determine for each ray the tetrahedron in front and behind the destination vertex of the ray and the intersected interface in front of the vertex. On average, the adaption of the energy to the modified DT structure requires 0.44 ms per integrated 3D point. Typical SLAM applications like [4] generate a few hundred 3D points per second which can be integrated into the surface in the same time with our approach. The incremental implementation of [13] has to test for each ray which of the modified tetrahedra are intersected by which ray. The number of rays involved in the energy update of [13] is an order of magnitude higher than in our



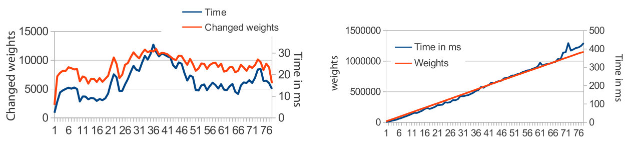
(a) Runtime (b) Labatut (c) Ours

Figure 7: (a) Difference in runtime for updating the energy function for 1 000 3D points. The x-axis shows the total number of 3D points in the mesh. (b) and (c) Time for energy update and the number of rays involved in the update. Please note, that scaling differs significantly.

approach. Since for each ray [13] has to determine the set of tetrahedra that are intersected by the ray, the absolute time is on average more than 20 times higher (9159 ms vs. 440 ms). Another important fact for real-time applications is the variance of the complexity. The large deviations in [13] are caused by the following problem. If a tetrahedron is modified that is intersected by large number of rays, all of these rays have to be taken into account to update the pairwise energy term. For example, in the building sequence several tetrahedra are passed by more than 50 000 rays and if one of these is modified the integration time rises drastically.

The second part is to solve the labeling problem by minimizing the energy function. Standard graph cut methods are designed to solve static problems, i. e. the energy is once defined and the minimum is calculated. In contrast, our approach generates a series of energies with an increasing number of terms. Figure 8(b) shows that the number of terms grows nearly linear in the number of points integrated in the DT. When using a static graph cut solver like [3], the time for solving also increases linearly and requires 430 ms for the final energy. In contrast, the time for solving the dynamic graph cut largely depends on the number of changed terms (Figure 8(a)) and does not depend on the overall problem size. In the building sequence, typically between 10 000 to 15 000 terms are updated when integrating 1 000 points into the reconstruction. The time required for the optimization varies between 20 ms and 30 ms. Compared to the time for the energy update which is around 440 ms for 1 000 points, the time for optimization is relatively small. This comparison gives evidence that the dynamic graph cut reduces the computational complexity and is independent of the overall scene size.

To sum up, our experiments demonstrate that our approach achieves the same accuracy as state-of-the-art methods for sparse SfM point clouds with a reduced computational effort. Our energy is suited to work in an incremental manner and in combination with the dynamic graph cut, computation time for energy minimization largely depends on the number of changed terms in the energy function.



(a) Dynamic graph cut (b) Static graph cut

Figure 8: Dynamic graph cut vs. static graph cut. The runtime for solving the dynamic graph cut depends on the number of changed terms in the energy function, whereas in the static case, the runtime depends of the overall number of terms.

5 Conclusion

In this paper, we proposed a novel method for incrementally extracting a triangular surface mesh from an increasingly growing sparse SfM point cloud in real-time. We formulate the problem as a labeling problem of a tetrahedralized point cloud into *free* and *occupied* space using a random field. We define a new energy function that achieves the same quality as state-of-the-art methods while being computationally efficient. Since our energy depends on local visibility information only, it can be easily adapted to a modified scene structure. In combination with the dynamic graph cut, we can extract the surface from an incrementally created point cloud in real-time largely independent of the overall scene size.

Acknowledgement This work has been supported by the Austrian Research Promotion Agency (FFG) FIT-IT project Construct (830035) and the FP7-ICT EU project Nr. 601139 CultAR.

References

[1] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Building Rome in a day. In *International Conference on Computer Vision (ICCV)*, 2009.

[2] J.-D. Boissonnat and M. Yvinec. *Algorithmic Geometry*. Cambridge University Press, Cambridge, U.K., 1998.

[3] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 23 (11):1222–1239, 2001.

[4] A. J. Davison, I. Reid, N. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29 (6):1052–1067, 2007.

[5] E. Eade and T. Drummond. Scalable monocular SLAM. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 469–476, 2006.

[6] Y. Furukawa and J. Ponce. Accurate, dense, and robust multi-view stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 32(8):1362–1376, 2010.

[7] G. Graber, T. Pock, and H. Bischof. Online 3D reconstruction using convex optimization. In *International Conference on Computer Vision (ICCV) Workshops*, pages 708–711. IEEE, 2011.

[8] G. A. Hansen, R. W. Douglass, and Andrew Zardecki. *Mesh Enhancement*. Imperial College Press, 2005.

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

[10] M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson surface reconstruction. In *Symposium on Geometry Processing*, pages 61–70, 2006.

[11] G. Klein and D.W. Murray. Parallel tracking and mapping for small AR workspaces. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 225–234. IEEE, 2007.

[12] P. Kohli and P.H.S. Torr. Dynamic graph cuts for efficient inference in markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 29 (12):2079–2088, December 2007.

[13] 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 *International Conference on Computer Vision (ICCV)*, 2007.

[14] D. Lovi, N. Birkbeck, D. Cobzas, and M. Jaegersand. Incremental Free-Space Carving for Real-Time 3D Reconstruction. In *Fifth International Symposium on 3D Data Processing Visualization and Transmission(3DPVT)*, May 2010.

[15] Q. Pan, G. Reitmayr, and T. Drummond. ProFORMA: Probabilistic Feature-based On-line Rapid Model Acquisition. In *British Machine Vision Conference (BMVC)*, London, September 2009.

[16] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. In *SIGGRAPH*, pages 835–846, 2006.

[17] C. Strecha, W. von Hansen, L. J. Van Gool, P. Fua, and U. Thoennessen. On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008.

[18] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.0 edition, 2012. http://www.cgal.org/Manual/4.0/doc_html/cgal_manual/packages.html.

[19] S. Yu and M. Lhuillier. Incremental reconstruction of manifold surface from sparse visual mapping. In *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT)*, pages 293–300. IEEE, 2012.

[20] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust TV-L1 range image integration. In *International Conference on Computer Vision (ICCV)*. IEEE, 2007.