Fast Marching for Robust Surface Segmentation

Falko Schindler and Wolfgang Förstner

Department of Photogrammetry, University of Bonn
Nussallee 15, 53115 Bonn, Germany
falko.schindler@uni-bonn.de,wf@ipb.uni-bonn.de
http://www.ipb.uni-bonn.de

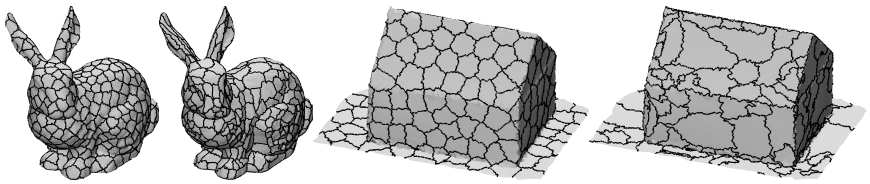
Abstract. We propose a surface segmentation method based on Fast Marching Farthest Point Sampling designed for noisy, visually reconstructed point clouds or laser range data. Adjusting the distance metric between neighboring vertices we obtain robust, edge-preserving segmentations based on local curvature. We formulate a cost function given a segmentation in terms of a description length to be minimized. An incremental-decremental segmentation procedure approximates a global optimum of the cost function and prevents from under- as well as strong over-segmentation. We demonstrate the proposed method on various synthetic and real-world data sets.

1 Introduction

Multi-view stereo reconstruction and laser range acquisition is gaining importance in photogrammetry and computer vision. Not only is aerial photogrammetry and airborne laser scanning a well established source for dense 2.5D or 3D reconstructions of millions and billions points; upcoming technologies like small unmanned aerial vehicles, time-of-flight cameras and even low-cost sensors developed for entertainment industries are emerging, enabling to capture colored point clouds in very short time. Many algorithms tackling the task of semantic understanding, however, require a data reduction pre-process to reduce computational complexity by removing redundant information. Pre-segmentation of images and 2.5D or 3D surfaces is a promising strategy for data reduction that we will focus on throughout this paper.

Given a triangulated point cloud we want to find homogeneous segments w.r.t. pre-determined features, e.g., direction of normals, location of gravity centers or color. While an under-segmentation will lead to loss of important information, over-segmenting the given surface is acceptable. Consecutive algorithms can easily merge segments based on more complex model knowledge, e.g., graphical models or incremental merging strategies (Attene et al. 2006). The unknown level of detail to be preserved implies the demand for a reasonable stopping criterion, e.g., the final number of segments.

We propose a segmentation algorithm based on Fast Marching Farthest Point Sampling (FastFPS, Moenning and Dogson 2003). To address the problem of specifying a stopping criterion we derive a theoretically founded cost



(a) Bunny, uniform (b) Bunny, adaptive (c) House, uniform (d) House, adaptive

Fig. 1. FastFPS for uniform (a, c) or adaptive (b, d) surface segmentation on the free-form object Bunny (a, b) and the polyhedral House (c, d). Note how the adaptive segmentation is sensitive to local curvature, thus yields more details at legs and ears and generalizes more strongly at the back (b). Especially when reconstructing polyhedral objects like buildings we prefer an adaptive, edge-preserving segmentation (d).

function, enabling to evaluate given segmentations. Our contribution is to introduce FastFPS for surface segmentation, to formulate a cost function, enabling to regulate an incremental and decremental segmentation strategy, and to introduce and to analyze distance metrics suited for surface segmentation.

The Bunny[1] in Fig. 1(a) is an example for a uniformly over-segmented, curved surface. We will propose distance metrics to obtain an adaptive segmentation as in Fig. 1(b) that is particularly practical for piecewise planar, polyhedral objects like buildings (Fig. 1(c) and 1(d)).

2 Related Work

Our segmentation approach is based on FastFPS (Moenning and Dogson 2003). It iteratively applies farthest point sampling (Eldar et al. 1997) to cover the sampling domain as uniformly as possible. To determine the farthest point a Voronoi diagram is constantly updated via fast marching (Sethian 1996).

The idea of farthest point sampling is to repeatedly place a new seed point in an area of the sampling domain that is farthest from the current seeds (Eldar et al. 1997). In terms of Voronoi diagrams such a farthest point is usually found on the boundary between two or more Voronoi cells. At the beginning there are no seeds and all points are infinitely far away from the – currently empty – set of seeds. A first seed point is chosen randomly and its distance to all remaining points is to be determined. More seeds are found by iteratively picking the farthest point and updating all distances. One way to efficiently compute the distances is to apply fast marching.

Fast marching was introduced for planar domains (Sethian 1996) and extended for triangulated domains (Kimmel and Sethian 1998). It allows to approximate distance maps $d(\boldsymbol{x})$ from given seed points $\boldsymbol{x}_0$ for the whole sampling

[1] `http://graphics.stanford.edu/data/3Dscanrep/`

domain by propagating a wave front following the Eikonal equation

$$|\nabla d(\boldsymbol{x})| = F(\boldsymbol{x}) \ , \tag{1}$$

with the boundary condition $d(\boldsymbol{x}_0) = 0$, i.e. the distance at the seed point is 0. Usually one is interested in geodesic distances, i.e. the friction $F(\boldsymbol{x})$ or speed $F^{-1}(\boldsymbol{x})$ is constant for all $\boldsymbol{x}$. An assignment of each sample point to its closest seed point is obtained implicitly. Alternatively we can formulate the propagation time dependent on local features, as it has been done for fast marching (Peyré and Cohen 2006) and for level sets (Xu et al. 2004).

The combination of fast marching and farthest point sampling was proposed for sampling point clouds and implicit surfaces (Moenning and Dogson 2003). Sampling seed points and updating a Voronoi diagram is equivalent to segmenting the sampling domain, i.e. determining a labeling l_n for each input point $n \in \{1 \dots N\}$, as summarized in Algorithm 1.

A similar segmentation approach is based on geodesic centroidal tesselation (Peyré and Cohen 2004). It addresses reconstruction tasks in computer aided-design and thus solely focuses on laser scans or synthetic data with a very small amount of noise and rare outliers compared to visual reconstructions.

In the context of segmentation FastFPS is distantly related to Watershed (Beucher and Lantu 1979) being applied to mesh segmentation using fast marching (Page et al. 2003). Instead of heights locally defined for each pixel or surface point (Mangan and Whitaker 1999) FastFPS is based on pairwise distances between points that depend on the current seed distribution and vary permanently. FastFPS can locally adapt to the underlying image content or surface shape evaluating the pre-defined distance metric F. As proposed for point sampling (Moenning and Dogson 2003) a favoured number of samples is required. We will demonstrate how to determine the stopping criterion automatically.

Another related concept for surface segmentation starts bottom-up by labeling each triangle of a triangulated surface differently and iteratively merges similar neighbors w.r.t. to planarity (Garland et al. 2001). Like FastFPS this approach needs a user-defined number of favoured segments.

The remainder of the paper is organized as follows: We first will describe the concept of surface segmentation, thereby specify our objective in terms of an optimal segmentation result and provide the incremental/decremental sampling strategy. Section 4 gives details of the algorithm, especially the distance metric F, how to involve color and the incremental/decremental choice of seed points. Experiments on synthetic data demonstrate the sensitivity w.r.t. noise and small surface patches. Real data shows the versatility of the method.

3 Theory

3.1 Task Specification

We assume to be given a set of N points $\{\boldsymbol{x}_n\}$ in 3D space, observed with equal and independent uncertainty $\sigma = \sigma_x = \sigma_y = \sigma_z$. Introducing full covariance

initialize distances $\boldsymbol{d} = \boldsymbol{\infty}$ and labels $\boldsymbol{l}$ as undefined;
for $l \leftarrow 1$ **to** L **do**
  set new seed $s \leftarrow \text{argmax}_n\, d_n$, $d_s \leftarrow 0$, $l_s \leftarrow l$;
  initialize new front $\mathcal{Q} \leftarrow \{s\}$;
  while *front is not empty* $\mathcal{Q} \neq \{\}$ **do**
    remove front vertex $u \leftarrow \text{argmin}_{q \in \mathcal{Q}}\, d_q$, $\mathcal{Q} \leftarrow Q \setminus u$;
    foreach *neighbor* $v \in \mathcal{N}_u$ **do**
      compute new distance $d'_v = d_u + F(v)$;
      if *new distance is smaller* $d'_v < d_v$ **then**
        update neighbor $d_v \leftarrow d'_v$, $l_v \leftarrow l_u$;
        add vertex to front $\mathcal{Q} \leftarrow \mathcal{Q} \cup v$;
      end
    end
  end
end

Algorithm 1: Fast Marching Farthest Point Sampling. New seeds s are added iteratively by choosing the farthest point w.r.t. a distance map $\boldsymbol{d}$ that is constantly updated using fast marching and a metric F, yielding a labeling $\boldsymbol{l}$.

matrices Σ_n is possible as well. We assume a triangulation of the points to be available. The task is to assign labels $l_n \in \{1 \dots L\}$ to each point n, such that equally labeled points are topologically connected via triangle edges. Connected components build the desired *segments*. Neighboring points living on one segment are supposed to share similar features like normals, curvature or color, while points on different segments are assumed to be different in normal, curvature or color. Normals and curvature can be derived from small neighborhoods.

A segmentation based on FastFPS would proceed as follows (Algorithm 1). Chose a random seed point s from where to compute geodesic distances d_n to all other points n using fast marching. I.e. initialize a set of front points $\mathcal{Q}$ with the neighbors of seed s and incrementally remove the front point u with smallest distance d_u and add its neighbors $\{v\}$ to front $\mathcal{Q}$. Update distances d_v and labeling $l_v \leftarrow l_u$ if $d_v + F(u, v)$ is smaller than a previously computed d_v. Fast marching stops, as soon as front $\mathcal{Q}$ is empty. Farthest point sampling continues with new seeds s always being the currently farthest point, until the required number of segments L is reached. Note that the first iteration is by far most expensive. Varying the number of segments can lead to under- or over-segmenting the surface.

To find a trade-off between under- and over-segmentation we formulate a cost function in terms of the description length for encoding all given points using a segmentation and the surface parameters of each segment. Our cost function

$$\Phi(\boldsymbol{l} \mid L) = \underbrace{N \operatorname{lb} L}_{\text{labeling}} + \underbrace{3L \operatorname{lb} \frac{R}{\varepsilon}}_{\text{parameters}} + \underbrace{2N \operatorname{lb} \frac{R}{\varepsilon}}_{\text{2D locations}} + \underbrace{\sum_n \left\{ \frac{\widehat{v}_n(l_n)^2}{2\sigma^2 \ln 2} + \operatorname{lb} \frac{\sqrt{2\pi}\sigma}{\varepsilon} \right\}}_{\text{residuals}} \tag{2}$$

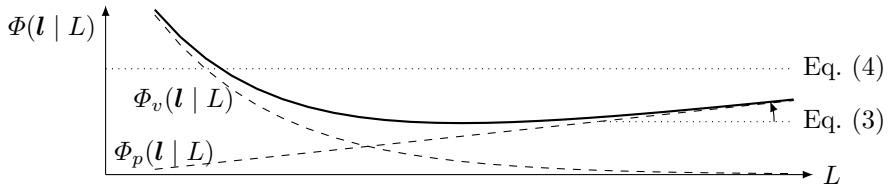


Fig. 2. Schematic diagram of $\Phi(\boldsymbol{l} \mid L)$. It is the sum $\Phi_v(\boldsymbol{l} \mid L) = \Omega(\boldsymbol{l} \mid L)$ of the squared residuals, principally decreasing with the number l of segments, and the bits $\Phi_p(\boldsymbol{l} \mid L)$ for coding the segments and the labeling of the points, which principally is slowly increasing with the number of segments. In reality, due to noise, the real optimization function is a noisy version of the smooth curve.

incorporates three terms, with R being the maximum range of one coordinate and the resolution $\varepsilon = \sigma/10$:

- *labeling*: For each point we to encode the segment it belongs to. Assuming the segments to be of approximately the same size the number of bits for each of the N points therefore is $\operatorname{lb} L$.
- *parameters*: We need to code surface parameters for each segment, e.g., the three parameters of a plane. Based on a resolution ε and a range R to be covered we need approximately $\operatorname{lb} R/\varepsilon$ bits to code each parameter.
- *2D locations* and *residuals*: We restrict all points n to live on their segment's surface. Therefore only the residuals $\widehat{v}_n(l_n)$ w.r.t. a best fitting surface for segment l_n and the 2D locations within the surface need to be coded (Leclerc 1989, Förstner 1989).

We expect Φ to rapidly decrease during the first iterations and to reach a local minimum, before gradually increasing again (Fig. 2). This is due to the exponentially decreasing sum of squared residuals Ω in combination with almost linearly increasing costs w.r.t. the growing number of segments L. The cost function Φ, however, is a noisy version of the ideal function shown in Fig. 2. This is due to the suboptimal criterion. The function Φ is N-dimensional, with N being the number of points, and highly non-convex. Nevertheless we will demonstrate an optimization strategy to approximate a global optimum using FastFPS.

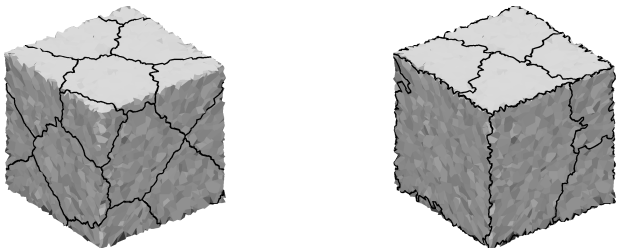
3.2 Incremental/Decremental Search

The local minimum is characterized by an increase in Φ being smaller than the differential costs for encoding the L segments only

$$0 < \Phi(\boldsymbol{l} \mid L) - \Phi(\boldsymbol{l} \mid L-1) < N(\operatorname{lb}(L) - \operatorname{lb}(L-1)) + 3\frac{\operatorname{lb} R}{\varepsilon} \ . \tag{3}$$

In order to prevent from being prone to noisy data and outliers, leading to a wobbly behaviour of Φ, we only apply the first criterion in case

$$\widehat{\Omega} = \sum_n \widehat{v}_n^2 < T_\Omega \cdot N \cdot \sigma^2 \ . \tag{4}$$



(a) Euclidean distance (b) Curvature-based distance

Fig. 3. Segmentation results for Euclidean and curvature-based distance measure on the data set CUBE with 6000 vertices and $\sigma = 5$ mm noise at 1 m edge length.

It requires the estimated sum of squared residuals $\widehat{\Omega}$ to be below a pre-defined threshold mainly depending on the expected variance of the data. The factor $T_\Omega \approx 10$ allows for larger residuals caused by violations of the expected Gaussian error distribution, e.g., few outliers not relevant for the overall segmentation.

4 Implementation

We apply the very same procedure (Algorithm 1) to surface segmentation with pre-defined number of segments and investigate several metrics F aiming at adaptive, robust, edge-preserving segmentations of polyhedral surfaces. Further we adjust the sampling strategy to better approximate an optimal segmentation.

4.1 Distance Metrics

Geodesics, i.e. pairwise Euclidean distances, are widely used for surface segmentation, possibly combined with shape-adaptive metrics, e.g., local curvature (Moenning and Dogson 2003). We will focus on the latter metric, compare it to Euclidean distances, propose a robustification and add radiometric features.

Euclidean distance vs. curvature. As shown in Fig. 3 and previously in Fig. 1, using a distance metric based on local curvature is crucial for segmenting planar patches without destroying straight edges. In Fig. 3(b) we compute normals for all points, by averaging the normals of all neighboring triangles. The distance F between two points is defined as the Euclidean distance of their normals. The effect of curvature-dependent segmentations is also visible with the ELLIPSOID in Fig. 4. At areas with low curvature the density is low.

Robust curvature. We improve the segmentation by computing the point's normals robustly. Rather than averaging the normals of neighboring triangles we take the median normals of the neighbors of second order, i.e. the neighbors of neighbors, or third order. Thereby the median of multiple normals is the

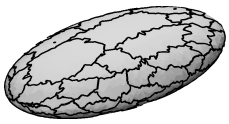
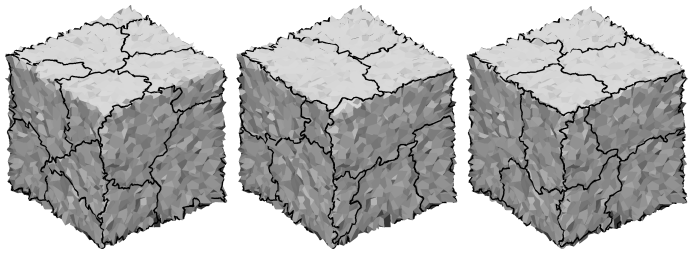


Fig. 4. Segmentation results for data set ELLIPSOID with 6000 vertices and $\sigma = 1$ mm noise at 4 m × 3 m × 2 m bounding box. The average segment size is clearly larger near the flat poles than at the relatively sharp equator.



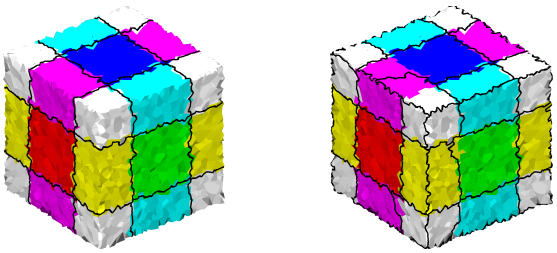
(a) 1st order (b) 2nd order (c) 3rd order

Fig. 5. Influence of the neighborhood size for robust curvature determination on the data set CUBE, here with 10 mm noise. Normals are obtained from directly neighboring triangles (a), neighbors of second order (b) or third order (c).

coordinate-wise median, normalized to unit length (Croux et al. 2002). As previously defined for first order normals, the distance F is the Euclidean distance between the normals of two neighboring points. Even when increasing the noise of the CUBE data set the segmentation successfully preserves all edges, when using robust normals of second or third order (Fig. 5).

Seed points at low curvature only. One effect to be considered when sampling based on curvature is to prevent seed points from being sampled at noisy edges and other highly non-planar regions. Points with large curvature tend to have large distances to all neighbors. They are likely chosen as seed point, but due to the large distance the front will not propagate to any neighbors and a tiny segment is generated, not improving the segmentation much. Many seeds are needed until all points of this type have been sampled. Instead of sampling the farthest point out of all available ones, we sort them by local curvature and restrict to the more planar half of them. Points close to edges are ignored when sampling seeds, leading to faster and more reasonable segmentations.

Color. We can easily extend the feature space or replace normals with different features, e.g., color. In case of photogrammetric reconstructions or laser range data combined with radiometric observations we may want to support the segmentation process using this information. The segmentation of a COLORED



(a) Color (b) Color + robust normals

Fig. 6. Segmentation of the data set COLORED CUBE with $\sigma = 5$ mm noise. Using color only, the surface is correctly segmented into 26 segments (a). When adding robust normals, the geometric edges of the cube are preserved as well (b).

CUBE solely relying on color (Fig. 6(a)) or in combination with robust normals (Fig. 6(b)) yields intuitive segmentations.

4.2 Incremental/Decremental FastFPS

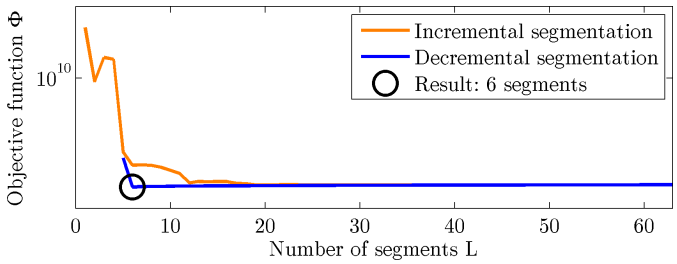
In Sect. 3 we formulated a cost function to be minimized. Applying Algorithm 1 yields promising results already, as shown in the previous section. Now we want to make use of the cost function (2) to determine the optimal number of segments automatically, while computing the segmentation. We search for the best segmentation in two steps: (1) increasing the number of segments sequentially until we likely find an over-segmentation and (2) decreasing the number of segments until a further reduction is likely to produce an under-segmentation.

When performing FastFPS, we evaluate the cost function in each iteration. The upper, orange line in Fig. 7(a) shows the decreasing costs for the iteratively increasing number of segments. At 6 segments, the expected number of segments, the labeling does not follow the edges well. The sum of squared residuals $\widehat{\Omega}$ is larger than expected and the costs Φ are still decreasing.

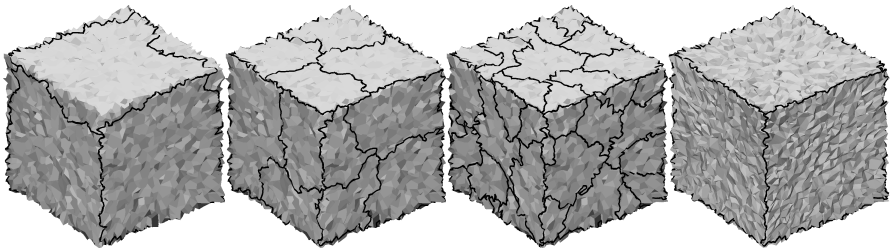
At 21 segments both criteria (3) and (4) are fulfilled. We do not stop increasing the number of segments immediately, but continue sampling two times more the current number of segments for even more robust results.

While the segmentation after the expected number of 6 iterations is rather poor (Fig. 7(b)), all edges are preserved when generating 63 segments (Fig. 7(d)). To avoid oversampling we start to decrease the number of segments decrementally. In each iteration we reset all points n' within the smallest region l': We remove the labels $l_{n'} = l'$, set the distances $d_{n'}$ back to infinity and set the current front $\mathcal{Q}$ to all labeled neighbors of points n'. We continue fast marching and obtain a complete labeling $\boldsymbol{l}$ and distance map $\boldsymbol{d}$ without label l'.

We proceed decrementally at almost constant costs Φ, as illustrated with the lower, blue line in Fig. 7(a). Suddenly, when removing one of the last 6 segments, the cost function rapidly increases by more than 10 %, alerting us to stop re-



(a) Value of cost function $\Phi(\boldsymbol{l} \mid L)$ during incremental (upper, orange line) and decremental (lower, blue line) segmentation process



(b) Incremental, 6 segments

(c) Incremental, 21 segments

(d) Incremental, 63 segments

(e) Decremental, 6 segments

Fig. 7. Influence of the incremental-decremental sampling strategy on the data set Cube with $\sigma = 10$ mm noise. The surface is correctly segmented into 6 segments, approximating a minimum of the objective function $\Phi(\boldsymbol{l} \mid L)$ defined in Sect. 3.

moving segments. We keep the labeling $\boldsymbol{l}$ at 6 segments, showing a significantly improved segmentation, while completely avoiding over-segmenting the surface.

Note that we remove the smallest segment in each iteration of the decremental part. Thus larger planar areas might not be merged, when smaller, important segments are present, causing the algorithm to stop before being removed.

5 Experiments

After evaluating the sensitivity of our proposed segmentation procedure w.r.t. small structures and noisy data, we demonstrate its performance on multiple data sets generated by structure-from-motion or laser range acquisition methods.

5.1 Sensitivity Analysis

We demonstrate the sensitivity of our method using a synthetically generated Stair of 1 m $\times$ 1 m size with varying height h from 10 cm down to 1 cm and Gaussian noise from 1 mm up to 10 mm. Table 1 lists the qualitative results of all 16 combinations, ranging from successfully segmented (✓) to not recognized as edge at all (×). Figure 8 shows the segmentation result for 4 combinations.

Table 1. Visual evaluation of the segmentation for data set STAIR with 5000 vertices, average sampling distance of $\Delta x = 1.3$ cm, variable height h and noise σ. For some scenarios the algorithm successfully segmented the vertical part preserving straight edges (✓). At increasing noise and decreasing height the vertical segment was segmented partly only (○) and finally not at all (×). Circled combinations are depicted in Fig. 8.

$\Delta x = 1.4$ cm	$\sigma = 1$ mm	2 mm	5 mm	10 mm
$h = 10$ cm	ⓒ	✓	✓	✓
$h = 5$ cm	✓	ⓒ	○	○
$h = 2$ cm	✓	○	ⓒ	○
$h = 1$ cm	○	○	×	ⓒ

(a) $h = 10$ cm, $\sigma = 1$ mm (b) $h = 5$ cm, $\sigma = 2$ mm (c) $h = 2$ cm, $\sigma = 5$ mm (d) $h = 1$ cm, $\sigma = 10$ mm

Fig. 8. Segmentation results for data set STAIR, sampled with 5000 vertices, increasing noise and decreasing height of the vertical segment.

At 5 cm height and 2 mm noise the vertical segment is perfectly preserved. When further decreasing the signal-to-noise ratio, the segmentation becomes incomplete (○), e.g., only preserving one of two edges, until the stair is not recognized at all (×), e.g., at $h = \sigma = 1$ cm.

5.2 Real-world Data

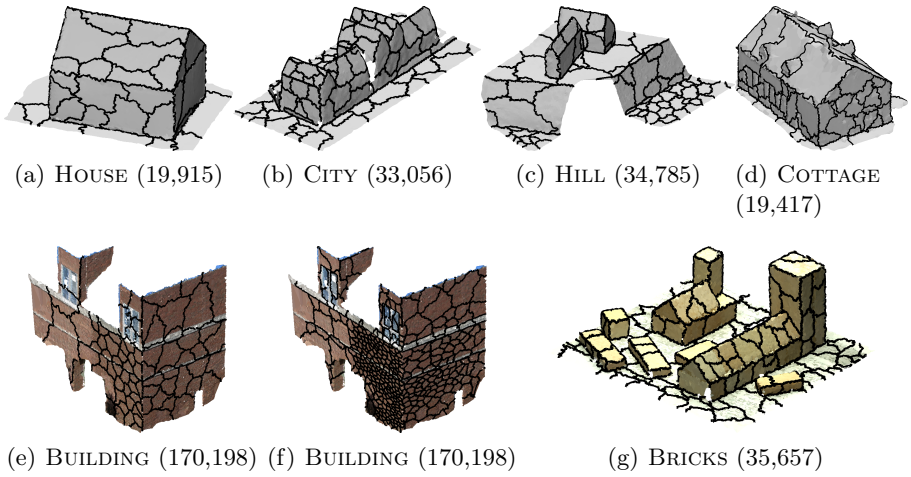
We have applied our segmentation to multiple real data sets.

HOUSE is a triangulated high-resolution point-cloud of a small building model captured using a Perceptron ScanWorks V5 laser scanner mounted on a Romer measuring arm. This data set is characterized by very small noise in the order of tens of micrometers. Figure 9(a) shows the generated segmentation result. Planar areas are slightly over-segmented, while all edges are accurately preserved.

CITY and HILL are triangulations obtained from a historic city model using structured-light 3D scanning. Even smooth edges are preserved due to our robust, curvature-adaptive distance metric, as shown in Fig. 9(b) and 9(c).

COTTAGE is a meshed 3D point cloud, captured with terrestrial laser scanning. Seven scans from different viewpoints have been registered into one common coordinate frame. The segmentation in Fig. 9(d) is mostly reasonable. Large edges are completely preserved, while minor structures like doors, windows and some dormers are not prominent enough to be perfectly segmented.

BUILDING is a reconstruction from 18 images using the structure-from-motion software Bundler (Snavely et al. 2006) and the Patch-based Multi-view Stereo



(a) House (19,915) (b) City (33,056) (c) Hill (34,785) (d) Cottage (19,417)

(e) Building (170,198) (f) Building (170,198) (g) Bricks (35,657)

Fig. 9. Segmentation results for various real-world data sets. Robust normals have been used for all examples (a – g), while (f) was segmented using RGB color as well. The number of vertices is indicated in brackets.

software (PMVS, Furukawa and Ponce 2007). The segmentation based on robust normals is shown in Fig. 9(e). Note the different segmentation density due to higher point density near the corner that is covered by most of the images. In case of visual reconstructions we can use normals in combination with color, as shown in Fig. 9(f), preserving radiometric edges as well.

Bricks is a visual reconstruction of multiple wooden bricks, captured with 48 images on a rotating turntable and processed with Bundler and PMVS.

Note that for real data our assumptions, e.g. Gaussian noise, are violated in some areas, leading to the visible over-segmentation, but still preserving edges.

All experiments were performed on a dual core machine using a Matlab implementation. Feature extraction takes a few minutes for point clouds of some ten thousands of points. The segmentation itself is computed within seconds.

6 Conclusion

We proposed a surface segmentation algorithm making use of the fast sampling approach FastFPS. We formulated the segmentation task as minimizing a cost function and presented an incremental-decremental algorithm to approximate a global optimum. Thereby our objective is to avoid loss of important information due to under-segmentation, since similar segments can be easily merged in a post-processing step, possibly using model knowledge. The cost function also allows to automatically stop the segmentation at statistically reasonable level-of-detail. Our robust distance metric is suitable for adaptive, edge-preserving segmentations of triangulated meshes with strong noise and outliers present, as shown in various experiments on both synthetic and real-world data.

The proposed cost function currently evaluates geometric features only and is designed for planar segments. It can be adapted to second-order or free-form surfaces, encoding both geometry and radiometry.

Acknowledgements. We are grateful to Friedrich Keller and Jérome Sänger for providing us with structured-light data from the historic city model of Hamburg, Germany, being result of their master thesis (Keller and Sänger 2010), supervised by Prof. Thomas Kersten and Prof. Jochen Schiewe.

References

Attene, M., Falcidieno, B., Spagnuolo, M.: Hierarchical Mesh Segmentation Based on Fitting Primitives. VISUAL COMPUT, **22**(3) (2006) 181–193

Beucher, S., Lantuéjoul, C.: Use of Watersheds in Contour Detection. Int. Workshop on Image Processing, Real-time Edge and Motion Detection/Estimation (1979) 17–21

Croux, C., Haesbroeck, G., Rousseeuw, P.J.: Location Adjustment for the Minimum Volume Ellipsoid Estimator. STAT COMPUT, **12**(3) (2002) 191–200

Eldar, Y., Lindenbaum, M., Porat, M., Zeevi, Y.Y.: The Farthest Point Strategy for Progressive Image Sampling. IEEE T IMAGE PROCESS, **6**(9) (1997) 1305–1315

Förstner, W.: Image Analysis Techniques for Digital Photogrammetry. Photogrammetrische Woche (1989) 205–221

Furukawa, Y., Ponce, J.: Accurate, Dense, and Robust Multi-view Stereopsis. CVPR (2007) 1362–1376

Garland, M., Willmott, A., Heckbert, P.S.: Hierarchical Face Clustering on Polygonal Surfaces. SIGGRAPH (2001) 49–58

Keller, F., Sänger, J.: Automatisierte Generierung von historischen 4D-Stadtmodellen für die Darstellung innerhalb der Google Earth Engine am Beispiel der Freien und Hansestadt Hamburg. Master thesis, HafenCity University Hamburg (2010)

Kimmel, R., Sethian, J.A.: Computing Geodesic Paths on Manifolds. P NATL ACAD SCI USA, **95**(15) (1998) 8431–8435

Leclerc, Y.G.: Constructing Simple Stable Descriptions for Image Partitioning. INT J COMPUT VISION, **3**(1) (1989) 73–102

Mangan, A.P., Whitaker, R.T.: Partitioning 3D Surface Meshes using Watershed Segmentation. IEEE T VIS COMPUT GR, **5**(4) (1999) 308–321

Moenning, C., Dodgson, N.A.: Fast Marching Farthest Point Sampling. Eurographics (2003)

Page, D.L., Koschan, A.F., Abidi, M.A.: Perception-based 3D Triangle Mesh Segmentation Using Fast Marching Watersheds. CVPR (2003) 27–32

Peyré, G., Cohen, L.: Surface Segmentation Using Geodesic Centroidal Tesselation. 3DPVT (2004) 995–1002

Peyré, G., Cohen, L.: Geodesic Remeshing Using Front Propagation. INT J COMPUT VISION, **69**(1) (2006) 145–156

Sethian, J.A.: A Fast Marching Level Set Method for Monotonically Advancing Fronts. P NATL ACAD SCI USA, **93**(4) (1996) 1591–1595

Snavely, N., Seitz, S.M., Szeliski, R.: Photo Tourism: Exploring Photo Collections in 3D. ACM T GRAPHIC (2006) 835–846

Xu, M., Thompson, P.M., Toga, A.W.: An Adaptive Level Set Segmentation on a Triangulated Mesh. IEEE T MED IMAGING, **23**(2) (2004) 191–201