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

Christof Hoppe, Manfred Klopschitz*,
Michael Donoser, Horst Bischof

Graz University of Technology

* Imaging and Computer Vision
Research Group Video Analytics
Corporate Technology, Siemens AG, Austria, Graz
Motivation

Structure-from-Motion
Point Cloud

- Up to City Scale
- Obtained in real-time (SLAM)
- Sparse representation
- AR and robotics require surface
- Not suitable for occlusion handling, navigation etc.

Volumetric Surface Reconstruction*

- High quality surface reconstruction
- Volumetric approach
- Limited scene size
- GPGPU required to handle computational effort

* Image taken from [Graber 2012]
Motivation

- Can we reconstruct a surface from sparse SfM points?
  - Consistent surface
    - Robust against outliers
  - Fully incremental to be integrated into SLAM
  - In real-time
  - Arbitrary camera motion
Challenges

- **Inhomogeneous** density of the scene information
- Severe **outliers**
- When using in combination with SLAM
  - Continuously **growing**
  - **Arbitrary camera motion** - “revisiting” of already reconstructed parts
Outline

- Related Work
- Formulation as Labeling Problem
- Incremental Surface Reconstruction
- Experiments
Related Work

- Irregular discretization of space into tetrahedra
- Perform 3D Delaunay triangulation of sparse 3D points
  - Fast, can be incrementally updated
- Classification into free / occupied space using visibility information
  - Interface is between free and occupied is surface
- Methods
  - Free-space carving [Lovi et al. 2010]
    → not robust to outlier
  - Formulation as labeling problem solved with graph cuts [Labatut et al. 2007]
    → Energy function motivated by free-space carving
    → robust against outliers, not suitable for incremental reconstruction
  - Aggregation of “free” tetrahedra for incremental reconstruction
    → [Poster yesterday, Litvinov et al. 2013, Lhuillier et al. 2013]
Contributions

- Robust free / occupied labeling of Delaunay triangulated sparse point cloud
- Formulation as Conditional Random Field
- Energy function can be easily adapted to modified Delaunay triangulation (DT)
  - New 3D points can be easily integrated into the DT
- Integration of new scene information leads to series of energy functions
  - Optimization using dynamic graph cuts
Our Approach
Our Approach
Our Approach
Our Approach
Our Approach
Random Field Formulation

- **Goal:** Classify each tetrahedron \( V_i \) into free or occupied given the visibility information / rays \( R \)
- \( R \) set of all line segments that connects a sparse 3D point to a camera center
- Energy function to minimize

\[
E(\mathcal{L}) = \sum_i (E_u(V_i, R_i) + \sum_{j \in \mathcal{N}_i} E_b(V_i, V_j, R_i))
\]

- \( R_i \) line segments connected to the vertices of the tetrahedron \( V_i \)
- **Unary and binary potentials only depend on local ray information** \( R_i \)
- Submodular function \( \rightarrow \) Can be optimized by graph cuts

**Smoothness across neighbouring tetrahedra**

**Probability tetrahedron free or occupied**
Unary Potentials

- Unary terms motivated by truncated signed distance function
- Probability that a tetrahedron “in front” of 3D point is free is high
- Probability that a tetrahedron “behind” a 3D point is occupied is high
- “In front” → tetrahedron intersected by a ray connected to its vertices
- “Behind” → tetrahedron is in extent of a ray connected to its vertices
- Counting how often a tetrahedron is “in front” or “behind”
  - No ray/tetrahedron intersection required
  - Delaunay data structure speeds up the counting

\[
V_1 \quad V_2
\]
Binary Potentials

- Typically only 50% of all tetrahedra obtain unary potentials
  - Strong regularization required
- It is very unlikely that (Vi, Vj) obtain different labels
  - Costs for assigning different labels is set to a high value
- Except neighboring tetrahedra that are not crossed by common rays
Incremental Energy Update

- New 3D point changes the Delaunay triangulation
  - But only locally
- Existing tetrahedra are deleted, new ones are created
- Energy has to be updated $E_n \rightarrow E_{n+1}$
  - Deletion of tetrahedra removes terms from the energy
  - New tetrahedra add new terms
- Unaries and binaries depend only on local visibility information
- Energy update is quite fast → 1000 points require 0.5 seconds
Incremental Labeling

- Delaunay triangulation update-able
- Energy function easily update-able
  - Series of energies $E_n$ to be optimized
- Problem: Number of terms in energy grow over time
- Solving from scratch prevents scalability
Incremental Labeling

- Delaunay triangulation update-able
- Energy function easily update-able
  - Series of energies $E_n$ to be optimized
- Problem: Number of terms in energy grow over time
- Solving from scratch prevents scalability

**Solution:** Dynamic graph cut [Kohli et al. 2007]

- Optimization of series of energies that can be solved by graph cuts
- Re-use result from minimization of $E_{n-1}$
- Complexity depends on the number of changed terms, not on the overall number of terms
Experiments – Static

- Static case
  - All 3D points and visibility information is available
  - Input: SfM point cloud obtained by standard SfM pipeline like Bundler
    - \(77,300\) 3D points, connected to 4.4 rays on average
  - Size of reconstructed area: 200m x 50m
Free-space carving
78 seconds

Labatut et al.
79 seconds

Ours
32 seconds

Intel i7, Single Core
Experiments – Static

- Strecha Fountain 11 dataset
- 7123 3D points

Labatut et al.  Ours
Experiments – Incremental
Experiments – Incremental

Time for integrating 1000 new points

- Rays
- Time

Time in s

0 0.2 0.4 0.6 0.8 1 1.2 1.4 1.6

1 5 9 13 17 21 25 29 33 37 41 45 49 53 57 61 65 69 73 77
Dynamic Graph Cut

Static Graph Cut

Dynamic Graph Cut
Conclusion

- Can we reconstruct a **consistent** mesh from a sparse 3D point cloud?
  - Robustness by random field formulation labeling
- Can we reconstruct it **incrementally** and in **real time**?
  - 2000 sparse 3D points per second
  - Independent from overall scene size thanks to dynamic graph cut
  - Without GPGPU
- Are we limited to **specific camera** motion?
  - No, 3D points can be inserted on arbitrary parts in the scene
- Is it difficult to **implement**?
  - No, thanks to libraries like CGAL (DT) and the publicly available dynamic graph cut
Thanks for your attention!

[Graber 2012] G. Graber, Realtime 3D reconstruction, Masterthesis, TU Graz  

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
Unary - Occupied
Unary - Free
Free Space Carving
Structure-from-Motion

- High-resolution, overlapping images
- Estimation of camera poses
- Estimation of sparse / dense 3D scene points
Incremental Surface Reconstruction
Incremental Surface Reconstruction