Street-side mobile mapping

- no upper floors / roofs
- uncovered pedestrian-only areas
- undriven districts
- no courtyards
- occlusions (trees, parking cars, fences)
- often no ground

DENSE 3D POINT CLOUD FROM IMAGERY

INCOMPLETE MAP COVERAGE
Airborne 3D acquisition

**MULTI-VIEW STEREO FROM 15-CENTIMETER NADIR IMAGERY CAPTURED AT 3 KM**

- no detail on street-side
- walls not visible
- shadows
- smoothed (blurry) walls
- defects (holes)

**TYPICAL DEFECTS DUE TO MISSING DATA**
Idea of airborne/street-side fusion

Requirements: efficient + watertight + large-scale
This paper: efficient volumetric airborne/streetside fusion

Assumptions:
- visibility data (lines of sight) given
- point clouds are accurately geo-registered (via precise GPS/IMU or GCP’s)
This paper: efficient volumetric airborne/streetside fusion

AIRBORNE ONLY (FOR COMPARISON)

OUR FUSION RESULT
This paper: efficient volumetric airborne/streetside fusion

Our Fusion Result

Airborne Only (For Comparison)
Related work

**General surface reconstruction**
- explicit methods, e.g. zippering
- depth map integration via TSDF, e.g. VRIP, KinectFusion (Curless & Levoy SG’96, Izadi et al. ’06)
- s/t cut over voxel grid (unsigned, UDF) (Hornung & Kobbelt EG’06, Lempitsky & Boykov CVPR’07)
- Poisson Surface Reconstruction (octree) (Kazhdan EG’06)
- convex variational (voxels or height map), e.g. TV-L1, TGV-fusion
- cell complex (Chauve CVPR’10)
- 3DT approaches (Labatut et al. ICCV’07, Lafarge & Alliez EG’13)

**Street-side & airborne data**
- superpixel meshes (Bodis et al. CVIU’16)
- most address geo-localization / registration
- DSM + street-side LiDAR (Fruh & Zakhor CGA’03)
- Octree & Dual Contouring (Fiocco et al. 3DIM’05)
- Poisson Surface Reconstruction (Shan et al. 3DPVT’13)

Superpixel meshes (Bodis et al. CVIU’16)
Contributions

- 1\textsuperscript{st} to propose 3DT-fusion for airborne/street-side
- Point cloud blending against gross ray conflicts
- Techniques to reduce workload (large urban scenes)
- Many experiments on detail vs. workload (sparse & dense input)
Volumetric surface reconstruction via 3DT
Volumetric surface reconstruction via 3DT

NOISE-FREE
INPUT SAMPLES
Volumetric surface reconstruction via 3DT

3D DELAUNAY TRIANGULATION

Alternatives
- voxels (poor scalability)
- stixels (2.5D)
- octrees (more complicated)
- cell complex (poor scalability, needs planes)

Why 3DT?
- simple
- adaptive / scalable
- efficient
- detail-preserving
Volumetric surface reconstruction via 3DT

RAY SHOOTING

SENSOR
Volumetric surface reconstruction via 3DT

INSIDE/OUTSIDE
VOLUMETRIC
CLASSIFICATION
Volumetric surface reconstruction via 3DT

**REAL DATA (NOISY)**

**Post processing**
- Denoising: simple smoothing
- Remove floating components

**Expensive alternative**
- Mesh tuning for photoconsistency (see Vu et al. PAMI 2012)
Raycasting and voting scheme

sensor

visibility ray
tetrahedra

$\nu$
$t_1$
$t_2$
$t_3$
$t_4$
$t_5$
$t_6$
$t_7$
$t_8$
$t_9$
Raycasting and voting scheme

Raycasting and voting scheme

\[ S_l(r, t) = 1 - e^{-d_i^2(r, t)/(2\sigma_l^2)} \quad l \in \{\text{in, out}\} \]

inside and outside score per ray per tetrahedron
Volumetric optimization

\[ E(\mathcal{L}) = \sum_{i : t_i \in \mathcal{T}} E_i(l_i) + \sum_i \sum_{j : j < i} E_{ij} \cdot \mathbb{I}[l_i \neq l_j] \]

- ** unary preference for in/out per tetrahedron**
- ** penalty for a face to be part of the surface**

**Optimization**: globally optimal GCO

\[ l_i \in \{\text{in, out}\} \quad \text{binary labels} \]
\[ \mathcal{T} = \{t_i\} \quad \text{tetrahedra} \]
Volumetric optimization

\[ E(\mathcal{L}) = \sum_{i: t_i \in \mathcal{T}} E_i(l_i) + \sum_i \sum_{j: j < i} E_{ij} \cdot \mathbb{I}[l_i \neq l_j] \]

- **Unary preference for in/out per tetrahedron**
- **Penalty for a face to be part of the surface**

\[ U_i(l) = \sum_r S_i(r, t_i) \]

**Sum over all rays**

\[ E_i(l) = 1 - e^{-U_i(l)/\gamma_l} \]

**Normalization**

\[ l_i \in \{\text{in, out}\} \quad \text{binary labels} \]

\[ \mathcal{T} = \{t_i\} \quad \text{tetrahedra} \]

**Optimization:** globally optimal GCO
Volumetric optimization

\[ E(\mathcal{L}) = \sum_{i : t_i \in \mathcal{T}} E_i(l_i) + \sum_i \sum_{j : j < i} E_{ij} \cdot \mathbb{I}[l_i \neq l_j] \]

- unary preference for in/out per tetrahedron
- penalty for a face to be part of the surface

\[ U_i(l) = \sum_r S_i(r, t_i) \]

sum over all rays

\[ E_i(l) = 1 - e^{-U_i(l) / \gamma_l} \]

normalization

\[ l_i \in \{in, out\} \]

binary labels

\[ \mathcal{T} = \{t_i\} \]

tetrahedra

**Optimization**: globally optimal GCO

\[ E_{ij} = \lambda A_{ij} \]

area penalty

(worked best with our unary)

**Other examples tried**
- triangle elongatedness
- long edge-triangles
- beta-skeleton (Labatut et al., CGF 2009)
Airborne / street-side surface fusion
Airborne / street-side surface fusion
Airborne / street-side surface fusion
Airborne / street-side ray conflicts

rays to low-quality airborne surface ruin the solution

INSIDE VOTES (BEHIND RAYS)

OUTSIDE VOTES (CROSSED BY RAYS)
Airborne / street-side ray conflicts

INSIDE VOTES (BEHIND RAYS)

OUTSIDE VOTES (CROSSED BY RAYS)

rays to low-quality airborne surface ruin the solution
Airborne / street-side point cloud blending

Observations
- airborne covers map
- airborne has much less detail
- not practical to do weighting instead of blending
Airborne / street-side point cloud blending
Airborne / street-side point cloud blending

\[ \mathcal{P} = \{p_i\} \]

\[ Q = \{q_i\} \]
Airborne / street-side point cloud blending

\[ \phi_i = \phi(d_i, \theta_i) = e^{-d_i^2/(2\sigma_i^2)} \cdot \max\{0, \cos \theta_i\} \]

substitute likelihood

\[ Q = \{q_i\} \]

\[ P = \{p_i\} \]

distance term

normal term
Airborne / street-side point cloud blending

\[ E^b(\mathcal{L}) = \sum_{i:p_i \in \mathcal{P}} E^b_i(l_i) + \lambda_b \sum_{ij} \psi(p_i, p_j) \cdot \mathbb{I}[l_i \neq l_j] \]

\[ \mathcal{P} = \{p_i\} \]

\[ Q = \{q_i\} \]

\[ \theta_i \]

\[ d_i \]

\[ \psi(p_i, p_j) = e^{-d_{ij}/\text{med}d_{ij}} \]

Blending energy

Likelihood

Smoothness

Smoothness term
Airborne / street-side point cloud blending
Airborne / street-side point cloud blending

3DT-BASED SURFACE RECONSTRUCTION
Data reduction / speed boost for large datasets

- $O(10^7)$ points (street / aerial)
- $O(10^8)$ rays
- $O(\text{km}^2)$ area

(illustration: mesh from airborne-only input)
Data reduction / speed boost for large datasets

(1) Point decimation

(2) Ray decimation

(3) Ray truncation
Data reduction / speed boost for large datasets

- voxel grid (with hashing)
- point clustering
- centroid prototype
- merge rays
- parameter: voxel size
Data reduction / speed boost for large datasets

(1) Point decimation

- voxel grid (with hashing)
- point clustering
- centroid prototype
- merge rays
- parameter: voxel size

(2) Ray decimation

(3) Ray truncation
Data reduction / speed boost for large datasets

(1) Point decimation
- voxel grid (with hashing)
- point clustering
- centroid prototype
- merge rays
- parameter: voxel size

(2) Ray decimation

(3) Ray truncation
Data reduction / speed boost for large datasets

(1) Point decimation
- voxel grid (with hashing)
- point clustering
- centroid prototype
- merge rays
- parameter: voxel size

(2) Ray decimation
- rays are dense
- keep one per point
- most perpendicular to reduce mistakes

(3) Ray truncation
-
Data reduction / speed boost for large datasets

- voxel grid (with hashing)
- point clustering
- centroid prototype
- merge rays
- parameter: voxel size

- rays are dense
- keep one per point
- most perpendicular to reduce mistakes
Data reduction / speed boost for large datasets

(1) Point decimation
- voxel grid (with hashing)
- point clustering
- centroid prototype
- merge rays
- parameter: voxel size

(2) Ray decimation
- rays are dense
- keep one per point
- most perpendicular to reduce mistakes

(3) Ray truncation
- similar to limiting penetration depth
- truncate rays at points
- less tetrahedra crossed
Data reduction / speed boost for large datasets

(1) Point decimation
- voxel grid (with hashing)
- point clustering
- centroid prototype
- merge rays
- parameter: voxel size

(2) Ray decimation
- rays are dense
- keep one per point
- most perpendicular to reduce mistakes

(3) Ray truncation
- similar to limiting penetration depth
- truncate rays at points
- less tetrahedra crossed
Pipeline summary

AERIAL POINTS & RAYS

PREPROCESSING (NORMALS & DATA REDUCTION)

STREET-SIDE POINTS & RAYS

POINT CLOUD BLENDING

VOLUMETRIC SURFACE RECONSTRUCTION (INTERPOLATORY)

POST-PROCESSING (DE-NOISING)

Implementation

- 3DT partitioning (Matlab ~ CGAL)
- Graph cuts (GCoptimization library)
- 3DT adjacencies & ray shooting (C++)
## Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Point neighborhood for normals</td>
<td>$k$</td>
<td>10</td>
</tr>
<tr>
<td>Blending distance control</td>
<td>$\sigma_b$</td>
<td>2 m</td>
</tr>
<tr>
<td>Blending smoothness</td>
<td>$\lambda_b$</td>
<td>1</td>
</tr>
<tr>
<td>Inside scoring distance softness</td>
<td>$\sigma_{in}$</td>
<td>0.1 m</td>
</tr>
<tr>
<td>Outside scoring distance softness</td>
<td>$\sigma_{out}$</td>
<td>0.5 m</td>
</tr>
<tr>
<td>Ray penetration limit</td>
<td>$\delta_{in}$</td>
<td>3$\sigma_{in}$</td>
</tr>
<tr>
<td>Ray truncation distance (optional)</td>
<td>$\delta_{out}$</td>
<td>3$\sigma_{out}$</td>
</tr>
<tr>
<td>Inside ray count softness</td>
<td>$\gamma_{in}$</td>
<td>2</td>
</tr>
<tr>
<td>Outside ray count softness</td>
<td>$\gamma_{out}$</td>
<td>2</td>
</tr>
<tr>
<td>Smoothness weight (area term)</td>
<td>$\lambda$</td>
<td>1–3</td>
</tr>
</tbody>
</table>

fixed in our experiments
Datasets

**Airborne**
- Nadir
- 15 cm GSD
- CapturingReality
- 50 GCPs

**Street-side**
- Hand held camera
- VisualSfM (sparse) / PMVS (dense)
- registered to the airborne data (via IMU/GPS in industrial mobile mapping)

Munsterhof (140x160 m²) 629 street-side images

Limmatquai (400x400 m²) 847 street-side images
Registration accuracy in our experiments

registration errors between airborne and street-side point clouds at overlaps
(distances in normal direction between mutual NNs)
Results: Process on Munsterhof

Airborne input (272k points) → Street-side input (1.5M points, PMVS) → Blending unary energies
Results: Process on Munsterhof

Point cloud segmentation  →  Blended point clouds  →  Surface reconstruction
Munsterhof Results

140x160 m² area
1.8 M points
11 M tetrahedra
13.2 M rays
8 points/m² aerial
108 points/m² street
5 min @ 3.4 GHz 1core
<1 min with decim / trunc
Limmatquai Results

400x400 m² area
2.6 M points
16.7 M tetrahedra
67.8 M rays
7 points/m² aerial
26 points/m² street
(20 cm vox decimation)
12 GB memory
21 min @ 3.4 GHz 1 core
<5 min with decim / trunc
Limmatquai Results

- **400x400 m² area**
- **2.6 M points**
- **16.7 M tetrahedra**
- **67.8 M rays**
  - 7 points/m² aerial
  - 26 points/m² street
  - (20 cm vox decimation)
- **12 GB memory**
- **21 min @ 3.4 GHz 1core**
- **<5 min with decim / trunc**
Results on different input types

Fusion aerial + PMVS
~108 points/m²

Fusion aerial + VisualSfM
~19 points/m²

Airborne only
~8 points/m²
**Cumulated histogram of mesh errors (vs runtimes)**

*reference*: our highest-density fused mesh (Munsterhof)

*timings*: on a single 3.4 GHz CPU core

<table>
<thead>
<tr>
<th>Method</th>
<th>Quality</th>
<th>Mean Error (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PMVS, ray truncation</td>
<td>good</td>
<td>182s</td>
</tr>
<tr>
<td>PMVS, ray decim (with/out trunc)</td>
<td>good</td>
<td>137s</td>
</tr>
<tr>
<td>PMVS, point decim (10 / 20 / 30 cm voxels)</td>
<td>good/poor</td>
<td>82s/23s</td>
</tr>
<tr>
<td>VisualSfM with/out reductions</td>
<td>poor</td>
<td>70s/47s</td>
</tr>
<tr>
<td>Reference</td>
<td></td>
<td>285s</td>
</tr>
</tbody>
</table>

*aerial-only input with/out reductions*
### Numerical details (see paper)

#### Parameter variations (others fixed)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Detailed statistics</th>
<th>Accuracy measures</th>
<th>Timings</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Münsterhof</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMVS (*)</td>
<td>0 3</td>
<td>272k 7.8 11.9</td>
<td>233k 18.7 1.0</td>
</tr>
<tr>
<td>PMVS (a)</td>
<td>0 √ 3</td>
<td>272k 7.6 11.9</td>
<td>233k 18.7 1.0</td>
</tr>
<tr>
<td>PMVS (b)</td>
<td>0</td>
<td>272k 7.8 1.0</td>
<td>233k 19.0 1.0</td>
</tr>
<tr>
<td>PMVS (c)</td>
<td>0 √ 1</td>
<td>272k 7.7 1.0</td>
<td>233k 19.0 1.0</td>
</tr>
<tr>
<td>PMVS (d)</td>
<td>0</td>
<td>272k 7.8 1.0</td>
<td>233k 19.0 1.0</td>
</tr>
<tr>
<td>PMVS (e)</td>
<td>0 √ 3</td>
<td>272k 6.2 11.9</td>
<td>not used</td>
</tr>
<tr>
<td>PMVS (f)</td>
<td>0.35 1</td>
<td>213k 6.3 1.0</td>
<td>127k 6.3 1.0</td>
</tr>
<tr>
<td>PMVS (*)</td>
<td>0.35 1</td>
<td>213k 6.3 1.0</td>
<td>127k 6.3 1.0</td>
</tr>
<tr>
<td>SfM (*)</td>
<td>0</td>
<td>272k 7.8 11.9</td>
<td>233k 18.7 1.0</td>
</tr>
<tr>
<td>SfM (a)</td>
<td>0 √ 3</td>
<td>272k 6.7 11.5</td>
<td>233k 18.7 1.0</td>
</tr>
<tr>
<td>SfM (b)</td>
<td>0</td>
<td>272k 6.9 1.0</td>
<td>233k 19.0 1.0</td>
</tr>
<tr>
<td>SfM (c)</td>
<td>0 √ 3</td>
<td>272k 6.8 1.0</td>
<td>233k 19.0 1.0</td>
</tr>
<tr>
<td>SFM (a)</td>
<td>0</td>
<td>272k 6.2 11.9</td>
<td>not used</td>
</tr>
<tr>
<td>SFM (a)</td>
<td>0 √ 1</td>
<td>272k 6.3 1.0</td>
<td>not used</td>
</tr>
<tr>
<td><strong>Limmatquai</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PMVS (*)</td>
<td>0.20 3</td>
<td>1.65M 6.8 11.5</td>
<td>1.14M 26.0 43.6</td>
</tr>
<tr>
<td>PMVS (a)</td>
<td>0 √ 3</td>
<td>1.65M 6.7 11.5</td>
<td>1.14M 26.3 43.6</td>
</tr>
<tr>
<td>PMVS (b)</td>
<td>0</td>
<td>1.65M 6.9 1.0</td>
<td>1.14M 26.5 1.0</td>
</tr>
<tr>
<td>PMVS (c)</td>
<td>0 √ 3</td>
<td>1.65M 6.8 1.0</td>
<td>1.14M 26.9 1.0</td>
</tr>
<tr>
<td>PMVS (a)</td>
<td>0</td>
<td>1.68M 6.5 11.5</td>
<td>not used</td>
</tr>
<tr>
<td>PMVS (a)</td>
<td>0 √ 1</td>
<td>1.68M 6.6 1.0</td>
<td>not used</td>
</tr>
<tr>
<td>SfM (*)</td>
<td>0</td>
<td>1.68M 6.5 11.5</td>
<td>259k 9.1 4.6</td>
</tr>
<tr>
<td>SfM (a)</td>
<td>0 √ 1</td>
<td>1.68M 6.5 11.5</td>
<td>259k 9.2 4.6</td>
</tr>
<tr>
<td>SfM (b)</td>
<td>0</td>
<td>1.68M 6.6 1.0</td>
<td>259k 9.0 1.0</td>
</tr>
<tr>
<td>SfM (c)</td>
<td>0 √ 1</td>
<td>1.68M 6.5 1.0</td>
<td>259k 9.5 1.0</td>
</tr>
<tr>
<td>SfM (a)</td>
<td>0</td>
<td>1.68M 6.0 11.5</td>
<td>not used</td>
</tr>
<tr>
<td>SfM (a)</td>
<td>0 √ 1</td>
<td>1.68M 6.0 1.0</td>
<td>not used</td>
</tr>
</tbody>
</table>
Qualitative effect of prior blending with normals

With Blending

Without Blending

With Normals

Without Normals (Distance Only)
Qualitative effect of ray decimation and truncation

No reductions
Qualitative effect of ray decimation and truncation

Ray decimation

![Images of buildings with ray decimation and truncation effects for PMVS and SfM techniques]
Qualitative effect of ray decimation and truncation

Ray decimation and truncation

PMVS

SfM
Quick Comparison to Poisson Surface Reconstruction

- Ours
  - PMVS+aerial
  - after blending
  - full rays
Quick Comparison to Poisson Surface Reconstruction

- PMVS+aerial
- after blending
- Depth 12, Div 8
- no rays used
- Inflated / smoother
- ground collapses
- narrow struct lost
- aerial artifacts remain
Conclusions

- 3DT-fusion of airborne & street-side data
- Point cloud blending against gross ray conflicts
- Reduction techniques for large urban scenes
- Detailed runtime vs. quality experimentation
- Complete & detailed (LoD-3) models in minutes / km²
Efficient Volumetric Fusion of Airborne and Street-Side Data for Urban Reconstruction

András Bódis-Szomorú, Hayko Riemenschneider, Luc Van Gool

Thank you!