3D Modeling with Depth Sensors

Marc Pollefeys, Viktor Larsson

Spring 2019

http://www.cvg.ethz.ch/teaching/3dvision/
<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb 18</td>
<td>Introduction</td>
</tr>
<tr>
<td>Feb 25</td>
<td>Geometry, Camera Model, Calibration</td>
</tr>
<tr>
<td>Mar 4</td>
<td>Features, Tracking / Matching</td>
</tr>
<tr>
<td>Mar 11</td>
<td><strong>Project Proposals by Students</strong></td>
</tr>
<tr>
<td>Mar 18</td>
<td>Structure from Motion (SfM) + papers</td>
</tr>
<tr>
<td>Mar 25</td>
<td>Dense Correspondence (stereo / optical flow) + papers</td>
</tr>
<tr>
<td>Apr 1</td>
<td>Bundle Adjustment &amp; SLAM + papers</td>
</tr>
<tr>
<td>Apr 8</td>
<td><strong>Student Midterm Presentations</strong></td>
</tr>
<tr>
<td>Apr 15</td>
<td>Multi-View Stereo &amp; Volumetric Modeling + papers</td>
</tr>
<tr>
<td>Apr 22</td>
<td>Easter break</td>
</tr>
<tr>
<td><strong>Apr 29</strong></td>
<td>3D Modeling with Depth Sensors + papers</td>
</tr>
<tr>
<td>May 6</td>
<td>3D Scene Understanding + papers</td>
</tr>
<tr>
<td>May 13</td>
<td>4D Video &amp; Dynamic Scenes + papers</td>
</tr>
<tr>
<td>May 20</td>
<td>papers</td>
</tr>
<tr>
<td>May 27</td>
<td><strong>Student Project Demo Day = Final Presentations</strong></td>
</tr>
</tbody>
</table>
Previously

- Obtaining “depth maps” / “range images” via stereo matching (Lecture 5)
- Volumetric modeling from multiple images and their depth maps (last lecture)
Today

- **Actively** obtaining “depth maps” / “range images”
  - unstructured light
  - structured light
  - time-of-flight
- Registering range images for 3D modeling

(some slides from Szymon Rusinkiewicz, Brian Curless)
Taxonomy

3D modeling

passive
- stereo
- shape from silhouettes

active
- photometric stereo
- laser scanning
- structured/unstructured light
Unstructured Light

Project texture to disambiguate stereo

Image credits: Thomas Schöps
Space-time stereo

Davis, Ramamoorthi, Rusinkiewicz, CVPR’ 03

Spatial Neighborhood
\[ V_s(x_1) \]

I_1

Temporal Neighborhood
\[ V_t(x_1, t_0) \]

search epipolar line

search epipolar line

I_2
Space-time stereo

Davis, Ramamoothi, Rusinkiewicz, CVPR’ 03
Space-time stereo

Zhang, Curless and Seitz, CVPR’03

Static cases:
- A fronto-parallel surface
  - Solve for x shift.
- An oblique surface
  - Solve for x shift, x scale, y shear.

Moving case:
- An oblique surface
  - Solve for x shift, x scale, y shear, t shear.
Space-time stereo

Zhang, Curless and Seitz, CVPR’ 03
Light Transport Constancy

Davis, Yang, Wang, ICCV05
Triangulation Scanner

“Peak” position in image reveals depth
Triangulation: Moving the Camera and Illumination

- Moving independently leads to problems with focus, resolution
- Most scanners mount camera and light source rigidly, move them as a unit, allows also for (partial) pre-calibration
Triangulation: Moving the Camera and Illumination
Triangulation: Extending to 3D

- Alternatives: project dot(s) or stripe(s)
Triangulation Scanner Issues

- Accuracy proportional to working volume (typical is ~1000:1)
- Scales down to small working volume (e.g. 5 cm. working volume, 50 \(\mu\)m. accuracy)
- Does not scale up (baseline too large...)
- Two-line-of-sight problem (shadowing from either camera or laser)
- Triangulation angle: non-uniform resolution if too small, shadowing if too big (useful range: 15°-30°)
Triangulation Scanner Issues

- Material properties (dark, specular)
- Subsurface scattering
- Laser speckle
- Edge curl
- Texture embossing

Where is the **exact** (subpixel) spot position?
Surface

Range point

Sensor

Resulting range surface

Typical range error = 1 mm for 1 mm wide laser

ρ₁ ρ₂ Surface

Range point

Sensor

Resulting range surface

Typical range error = 0.5 mm for 1 mm wide laser
Space-time analysis

Curless, Levoy, ICCV '95
Space-time analysis

Curless, Levoy, ICCV '95

Edge curl reduction

Two thin strips  Traditional analysis  Spacetime analysis

Improved shape extraction

Shape ribbon  Traditional analysis  Spacetime analysis
Poor man’s scanner

The idea

Desk Lamp

Camera

Stick or pencil

Desk

Bouguet and Perona, ICCV’98
Projector as camera
Multi-Striped Triangulation

- To go faster, project multiple stripes
- But which stripe is which?
- Answer #1: assume surface continuity

E.g. Eyetronics’ ShapeCam
Multi-Stripe Triangulation

- To go faster, project multiple stripes
- But which stripe is which?
- Answer #2: colored stripes (or dots)
Multi-Stripe Triangulation

- To go faster, project multiple stripes
- But which stripe is which?
- Answer #3: time-coded stripes
Time-Coded Light Patterns

- Assign each stripe a unique illumination code over time [Posdamer 82]
Better codes...

- Gray code
  Neighbors only differ one bit
Kinect

- Infrared „projector“
- Infrared camera
- Works indoors (no IR distraction)
- „invisible“ for human

Depth Map: note stereo shadows!
Color Image (unused for depth)
IR Image
Kinect

- Projector Pattern „strong texture“
- Correlation-based stereo between IR image and projected pattern possible

stereo shadow  Bad SNR / too close  Homogeneous region, ambiguous without pattern
Pulsed Time of Flight

- Basic idea: send out pulse of light (usually laser), time how long it takes to return

\[ d = \frac{1}{2} c \Delta t \]
Pulsed Time of Flight

- **Advantages:**
  - Large working volume (up to 100 m.)

- **Disadvantages:**
  - Not-so-great accuracy (at best ~5 mm.)
    - Requires getting timing to ~30 picoseconds
    - Does not scale with working volume

- Often used for scanning buildings, rooms, archeological sites, etc.
Depth cameras

2D array of time-of-flight sensors

Kinect v2

Azure Kinect
3D modeling

- Aligning range images
  - Pairwise
  - Globally

(some slides from S. Rusinkiewicz, J. Ponce,...)
Aligning 3D Data

- If correct correspondences are known (from feature matches, colors, ...), it is possible to find correct relative rotation/translation
Aligning 3D Data

For $T$ as general $4 \times 4$ matrix:
Linear solution from $\geq 5$ corrs.

$T$ is Euclidean Transform:
3 corrs. (using quaternions)

[Horn87] “Closed-form solution of absolute orientation using unit quaternions”

$T_i' = T X_i$
Aligning 3D Data

- How to find corresponding points?
- Previous systems based on user input, feature matching, surface signatures, etc.
Spin Images

- [Johnson and Hebert ’97]
- “Signature” that captures \textit{local} shape
- Similar shapes $\rightarrow$ similar spin images
Computing Spin Images

• Start with a point on a 3D model
• Find (averaged) surface normal at that point
• Define coordinate system centered at this point, oriented according to surface normal and two (arbitrary) tangents
• Express other points (within some distance) in terms of the new coordinates
Computing Spin Images

- Compute histogram of locations of other points, in new coordinate system, ignoring rotation around normal:

\[ \alpha = |p \times \hat{n}| \text{ "radial dist."} \]
\[ \beta = p \cdot \hat{n} \text{ "elevation"} \]
Computing Spin Images

2-D points

spin-image

"elevation"

"radial dist."

3-D surface mesh

2-D points

spin-image

2-D points

spin-image
Spin Image Parameters

- **Size of neighborhood**
  - Determines whether local or global shape is captured
  - Big neighborhood: more discriminative power
  - Small neighborhood: resilience to clutter

- **Size of bins in histogram:**
  - Big bins: less sensitive to noise
  - Small bins: captures more detail
Alignment with Spin Image

• Compute spin image for each point / subset of points in both sets
• Find similar spin images => potential correspondences
• Compute alignment from correspondences

⇒ Same problems as with image matching:
  - Robustness of descriptor vs. discriminative power
  - Mismatches => robust estimation required
Aligning 3D Data

Alternative: assume closest points correspond to each other, compute the best transform...
Aligning 3D Data

... and iterate to find alignment

Iterated Closest Points (ICP) [Besl & McKay 92]

Converges if starting position “close enough”
ICP Variant – Point-to-Plane Error Metric

- Using point-to-plane distance instead of point-to-point lets flat regions slide along each other more easily [Chen & Medioni 92]
Finding Corresponding Points

- Finding closest point is most expensive stage of ICP
  - Brute force search – $O(n)$
  - Spatial data structure (e.g., k-d tree) – $O(\log n)$
  - Voxel grid – $O(1)$, but large constant, slow preprocessing
Finding Corresponding Points

- For range images, simply project point [Blais/Levine 95]
  - Constant-time, fast
  - Does not require precomputing a spatial data structure
Next week:
3D Scene Understanding