CSE 571
Robotics

RGB-D Mapping

University of Washington
Dieter Fox

RGB-D Mapping Overview

Visual Features

• Detector
  – Repeatable
  – Stable
  – Invariances:
    • Illumination
    • Rotation
    • Scale
• Descriptor
  – Discriminative
  – Invariant

Say we have 2 images of this scene we’d like to align by matching local features
What would be good local features (ones easy to match)?

Courtesy: S. Seitz and R. Szeliski

**Invariant local features**
- Algorithm for finding points and representing their patches should produce similar results even when conditions vary
  - Buzzword is “invariance”
    - geometric invariance: translation, rotation, scale
    - photometric invariance: brightness, exposure, ...

**Feature Descriptors**

**Scale Invariant Feature Transform**

Basic idea:
- Take 16x16 square window around detected feature
- Compute gradient for each pixel
- Throw out weak gradient magnitudes
- Create histogram of surviving gradient orientations

**SIFT keypoint descriptor**

Full version
- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

**Properties of SIFT**

Extraordinarily robust matching technique
- Can handle changes in viewpoint
  - Up to about 60 degree out of plane rotation
- Can handle significant changes in illumination
  - Sometimes even day vs. night (below)
- Fast and efficient—can run in real time
- Lots of code available
  - [http://www.vlfeat.org](http://www.vlfeat.org)
Feature distance

- How to define the difference between two features $f_1$, $f_2$?
  - Simple approach is $SSD(f_1, f_2)$
    • sum of square differences between entries of the two descriptors
    • can give good scores to very ambiguous (bad) matches

- Better approach: ratio distance $= SSD(f_1, f_2) / SSD(f_1, f_2')$
  • $f_2$ is best SSD match to $f_1$ in $I_2$
  • $f_2'$ is 2nd best SSD match to $f_1$ in $I_2$
  • gives small values for ambiguous matches

Are descriptors unique?

No, they can be matched to wrong features, generating outliers.
Strategy: RANSAC

- RANSAC loop:
  1. Randomly select a seed group of matches
  2. Compute transformation from seed group
  3. Find inliers to this transformation
  4. If the number of inliers is sufficiently large, re-compute least-squares estimate of transformation on all of the inliers

- Keep the transformation with the largest number of inliers

Simple Example

- Fitting a straight line

Why will this work?

RANSAC example: Translation

Select one match, count inliers

Find "average" translation vector

• Pros
  – Simple and general
  – Applicable to many different problems
  – Often works well in practice

• Cons
  – Lots of parameters to tune
  – Can’t always get a good initialization of the model based on the minimum number of samples
  – Sometimes too many iterations are required
  – Can fail for extremely low inlier ratios
Visual Odometry

• Compute the motion between consecutive camera frames from visual feature correspondences.
• Visual features from RGB image have a 3D counterpart from depth image.
• Three 3D-3D correspondences constrain the motion.

Visual Odometry Failure Cases

• Low light, lack of visual texture or features
• Poor distribution of features across image
• But: RGB-D camera still provides shape info

ICP (Iterative Closest Point)

• Iterative Closest Point (ICP) uses shape to align frames
• Does not require the RGB image
• Does need a good initial “guess”
• Repeat the following two steps:
  – For each point in cloud A, find the closest corresponding point in cloud B
  – Compute the transformation that best aligns this set of corresponding pairs

ICP Variants

• Correspondence
  – Outliers as absolute or percentage
  – No many-to-one correspondences
  – Reject boundary points
  – Normal agreement
• Error metric
  – Point-to-point
  – Point-to-plane
  – Weight by color / normal agreement
ICP (Iterative Closest Point)
• Iteratively align frames based on shape
• Needs a good initial estimate of the pose

ICP Failure Cases
• Not enough distinctive shape
• Don’t have a close enough initial “guess”
• Here the shape is basically a simple plane...

Optimal Transformation
• Jointly minimize feature re-projection and ICP:

\[
t^* = \arg \min_t \left[ \left( \frac{1}{|A_f|} \sum_{i \in A_f} |Proj(t(f_i')) - Proj(f_i')|^2 \right) + \beta \left( \frac{1}{|A_d|} \sum_{j \in A_d} w_j |(t(p_j') - p_i') \cdot n_j|^2 \right) \right]
\]


Joint Optimization (RGBD-ICP)

```
input: source RGB-D frame P_s, target RGB-D frame P_t, previous transformation T_p
output: Optimized relative transformation T^*
1  F_s ← Extract.RGB.Point.Features(P_s)
2  F_t ← Extract.RGB.Point.Features(P_t)
3  (T^*, A_f) ← Perform.RANSAC.Alignment(F_s, F_t)
4  if |A_f| < γ then
5      T^* = T_p
6  else
7      A_f = ∅
8      repeat
9      A_d ← Compute.Closest.Points(T^*, P_s, P_t)
10     T^* ← Optimize.Alignment(T^*, A_f, A_d)
11     until (Change(T^*) ≤ θ) or (Iterations > MaxIterations)
12  return T^*
```

Algorithm 1: RGB-D ICP algorithm for matching two RGB-D frames.
Experiments

• Reprojection error is better for RANSAC:

<table>
<thead>
<tr>
<th>Method</th>
<th>Inliers per frame</th>
</tr>
</thead>
<tbody>
<tr>
<td>EE-RANSAC</td>
<td>60.3</td>
</tr>
<tr>
<td>RE-RANSAC</td>
<td>116.7</td>
</tr>
</tbody>
</table>

• Errors for variations of the algorithm:

<table>
<thead>
<tr>
<th>Method</th>
<th>Inliers per frame (±std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel-Day</td>
<td>0.11 (±0.05)</td>
</tr>
<tr>
<td>Intel-Night</td>
<td>1.09 (±0.88)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Inliers per frame (±std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel-Day</td>
<td>0.15 (±0.00)</td>
</tr>
<tr>
<td>Intel-Night</td>
<td>0.17 (±0.06)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Inliers per frame (±std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel-Day</td>
<td>0.10 (±0.04)</td>
</tr>
<tr>
<td>Intel-Night</td>
<td>0.15 (±0.08)</td>
</tr>
<tr>
<td>Two-Stage</td>
<td>0.11 (±0.05)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Inliers per frame (±std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel-Day</td>
<td>1.15 (±0.89)</td>
</tr>
<tr>
<td>Intel-Night</td>
<td>0.15 (±0.00)</td>
</tr>
<tr>
<td>Two-Stage</td>
<td>0.15 (±0.09)</td>
</tr>
</tbody>
</table>

• Timing for variations of the algorithm:

<table>
<thead>
<tr>
<th>Method</th>
<th>Time (±std)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel-Day</td>
<td>0.21 (±0.03)</td>
</tr>
<tr>
<td>Intel-Night</td>
<td>0.20 (±0.05)</td>
</tr>
<tr>
<td>Two-Stage</td>
<td>0.21 (±0.03)</td>
</tr>
</tbody>
</table>

Loop Closure

• Sequential alignments accumulate error
• Revisiting a previous location results in an inconsistent map

Loop Closure Detection

• Detect by running RANSAC against previous frames
• Pre-filter options (for efficiency):
  – Only a subset of frames (keyframes)
  – Only keyframes with similar estimated 3D pose
  – Place recognition using vocabulary tree
    • Scalable recognition with a vocabulary tree, David Nister and Henrik Stewenius, 2006
• Post-filter (avoid false positives)
  – Estimate maximum expected drift and reject detections changing pose too greatly
Loop Closure Correction (TORO)

- TORO [Grisetti 2007, 2009]:
  - Constraints between camera locations in pose graph
  - Maximum likelihood global camera poses

\[
\sum_{c_i \in C} \sum_{p_j \in P} v_{ij} | \text{Proj}(c_i, p_j) - (\bar{u}, \bar{v}, \bar{d})|^2
\]
A Second Comparison

TORO  SBA

Timing

Resulting Map
Map Representation: Surfels

- Circular surface patches
- Accumulate color / orientation / size information
- Incremental, independent updates
- Incorporate occlusion reasoning
- 750 million points reduced to 9 million surfels
Application: Quadrocopter

• Collaboration with Albert Huang, Abe Bacharach, and Nicholas Roy from MIT

Larger Maps
Conclusion

- Kinect-style depth cameras have recently become available as consumer products
- RGB-D Mapping can generate rich 3D maps using these cameras
- RGBD-ICP combines visual and shape information for robust frame-to-frame alignment
- Global consistency achieved via loop closure detection and optimization (RANSAC, TORO, SBA)
- Surfels provide a compact map representation