ORB SLAM 2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras

Raul ur-Artal and Juan D. Tardos

Presented by:
Xiaoyu Zhou
Bolun Zhang
Akshaya Purohit
Lenord Melvix

Outline

- Background
- Introduction
- Tracking
- Local mapping
- Loop closing
- Experiments and Results
Motivation

What is SLAM?
- Simultaneous localization and mapping

Why SLAM?
- In an environment without GPS, how is localization achieved?
**Visual SLAM: Main Parts**

- **Sensor data**
- **Front-end**
  - Feature extraction
  - Data association:
    - Short-term (feature tracking)
    - Long-term (loop closure)
- **Back-end**
  - MAP estimation
- **SLAM estimate**

**Visual SLAM: Front-End, Back-End flow chart**

**Front end**
- Image sequence
- Feature detection
- Feature match (tracking)
- Motion estimation: 2D–2D, 3D–2D, 3D–3D

**Back end**
- Bundle Adjustment & Camera Pose Optimization
Visual SLAM: Front-End flow chart

- Image sequence
  - Feature detection
  - RANSAC, PnP
  - Feature matching

Visual SLAM: Front-End

Motion estimation:  
2D-2D: Essential Matrix, Planar Projective Transformation Matrix

- minimize reprojection error
- Impossible if the camera purely rotates
Visual SLAM: Front-End

Motion estimation: **3D-3D: Iterative Closest Point (ICP)**

Given two sets of 3D points, iteratively estimate the transformation $T_k$ that can minimize the 3D-3D distance.

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg \min_{T_k} \sum_{i} ||\tilde{x}^i_k - T_k \tilde{x}^i_{k-1}||$$

Visual SLAM: Front-End

Motion estimation: **3D-2D: Perspective from n Points (PnP)**

The solution is found by determining the transformation that minimizes the reprojection error.

$$T_k = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \\ 0 & 1 \end{bmatrix} = \arg \min_{T_k} \sum_{i} ||p^i_k - \hat{p}^i_{k-1}||^2$$
Visual SLAM: Back-End flow chart

- Each node represents a pose of the camera
- Each edge represents a constraint between two nodes.
- Minimize function below to improve camera’s poses.

\[ \sum_{e_{ij}} \|C_i - T_{e_{ij}} C_j\|^2 \]
**Visual SLAM:** Back-End

Bundle Adjustment (BA):

- Very similar to camera-pose optimization,
- Also optimize the position of 3D points, minimize reprojection error.
- Extremely time consuming.

**Visual SLAM:** Strongest Constraint

Loop Closure:

- The most valuable constraint for pose-graph optimization.
- Usually between nodes that are far away, which may have large drift.
- Very afraid of false positive, which can destroy the entire map.
Outline

- Background
- **Introduction**
- Tracking
- Local mapping
- Loop closing
- Experiments and Results

**ORB-SLAM2**: System Overview

- Feature-based
- Monocular, Stereo, and RGB-D
- Loop closing, relocalization and map reuse
- Three threads running in parallel
  - Tracking
  - Local Mapping
  - Loop Closing
ORB-SLAM2: Map

- **Map points**
  - 3D position
  - Viewing direction
  - Representative ORB descriptor
  - Viewing distance

- **Keyframes**
  - Camera pose
  - Camera intrinsics
  - ORB features in the frame

ORB-SLAM2: Map

- **Covisibility Graph**
  - Node: Keyframe
  - Edge: Share observations of map points
  - Min shared map points: 15

- **Essential Graph**
  - Subgraph of covisibility graph
  - Spanning tree, high weight edges, loop closure edges
  - Min shared map points: 100
**ORB-SLAM2: Place Recognition**

- **Visual Vocabulary**
  - Offline vocabulary of ORB descriptors extracted from a large set of images

- **Recognition Database**
  - Database built incrementally, which stores for each visual word in the vocabulary, in which keyframes it has been seen.
  - Vocabulary tree using hierarchical k-means clustering.
  - Leaves are the visual words.
Outline

- Background
- Introduction
- **Tracking**
- Local mapping
- Loop closing
- Experiments and Results

Tracking

- Localize the camera with every frame and decide when to insert a new keyframe.
Tracking: Preprocess Input

- Preprocess the input to extract features at salient keypoint locations
- All system operations are based on these features.
- Stereo Keypoints: \((u_L, v_L, u_R)\)
  - Close: depth < 40X baseline
  - Far: Otherwise

Tracking: Preprocess Input (Extract ORB)

- Extremely fast to compute than SIFT or SURF
**Tracking:** Preprocess Input (Extract ORB)

ORB features in ORB-SLAM  
ORB features in general

**Tracking:** Pose Prediction or Relocalization

- **Pose Estimation From Previous Frame**
  - Constant velocity motion model to predict the camera pose
  - Perform a guided search.
  - Pose optimization

- **Pose Estimation via Global Relocalization (if tracking lost)**
  - Convert the frame into bag of words
  - Query the recognition database: Get matching Keyframes
  - Outlier rejection: RANSAC
  - PnP to get pose
  - Guided Search
  - Pose optimization
Tracking: Pose Prediction or Relocalization

- Pose Optimization using Motion-only bundle adjustment:
  - Optimize camera orientation $\mathbf{R}$ and position $\mathbf{t}$
  - Minimizing error between matched 3D points in world coordinates and key points
  - Levenberg-Marquadt for non-linear optimization

\[
\{\mathbf{R}, \mathbf{t}\} = \arg\min_{\mathbf{R}, \mathbf{t}} \sum_{i \in \mathcal{X}} \rho \left( \left\| x_i^i - \pi(\cdot) (\mathbf{R}x_i^i + \mathbf{t}) \right\|^2 \right)
\]

\[
\pi_m \left( \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} \quad \pi_s \left( \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \right) = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \\ f_x \frac{X-b}{Z} + c_x \end{bmatrix}
\]

Tracking: Track Local Map

- Look into the local map for more map point correspondences.
- Pose optimization
**Tracking:** Track Local Map

1. Project in current frame
2. Angle between current viewing ray and map point mean viewing direction
   - Discard if angle > 60°
3. Distance from map point to camera center
   - Discard if distance not in [d_min, d_max]
4. Scale in the frame by the ratio d/d_min.
5. Compare map point descriptor with unmatched ORB features in the frame near z, take best match

**Decision criteria:**
- More than 20 frames must have passed from the last global relocalization.
- Local mapping is idle, or more than 20 frames have passed from last keyframe insertion.
- Current frame tracks at least 50 points.
- Current frame tracks less than 90% points than Kref.

---

**Tracking:** New KeyFrame

**Decision criteria:** (all required):

- More than 20 frames must have passed from the last global relocalization.
- Local mapping is idle, or more than 20 frames have passed from last keyframe insertion.
- Current frame tracks at least 50 points.
- Current frame tracks less than 90% points than Kref.
Outline

- Background
- Introduction
- Tracking
- **Local mapping**
- Loop closing
- Experiments and Results

Local Mapping

- Process new keyframes and performs local BA to optimize the map points and the poses of the keyframes
**Local Mapping: Keyframe Insertion**

- Update the covisibility graph
  - Add new node and update edges
- Update the spanning tree in essential graph
  - Link with the keyframe with most shared points
- Compute the bags of words representation
  - Help triangulating new points

**Local Mapping: Recent Map Points Culling**

- Removal test after creation
  - Can be found in more than 25% of the predicted visible frames
  - Can be observed in at least three keyframes
- Keyframe culling
- Local BA discarding
**Local Mapping:** New Map Point Creation

- New keyframe $K_i$ and connected keyframes $K_c$ in the covisibility graph
- For unmatched ORB in $K_i$, search match in $K_c$
  - Epipolar constraint
  - Speeds up by vocabulary tree
- Triangulate ORB pairs
  - Check depth, parallax, reprojection error, and scale consistency

---

**Local Mapping:** New Map Point Creation

- Determine new map point properties
  - Mean unit vector of all its viewing directions
  - Representative descriptor
  - Observation distance
- Search correspondences in other keyframes
  - Connected keyframes $K_j$ in covisibility graph
  - Neighbor keyframes $K_2$ to the keyframes $K_j$
  - Project new map points to $K_1$ and $K_2$
  - Update covisibility graph
**Local Mapping:** Local Bundle Adjustment

- Optimize poses and map points
  - Current keyframe $K_i$
  - Connected keyframes $K_c$ in the covisibility graph
  - Map points seen in $K_i$ and $K_c$
- Fixed constraint
  - Keyframes with same map points but not connected to $K_i$
- Discard map points outliers and modify poses and map point coordinates

\[
\{X^i, R_l, t_l | i \in P_L, l \in K_L\} = \arg\min_{X^i, R_l, t_l} \sum_{k \in K_L} \sum_{j \in X_k} \rho(E(k, j))
\]

\[
E(k, j) = \left\| x^j_{(\cdot)} - \pi(\cdot) \left(R_k X^j + t_k\right) \right\|^2_{\Sigma}
\]

where $K_L$ are set of co-visible keyframes, $P_L$ are all points in those keyframes and $K_F$ are other keyframes not in $K_L$ observing points in $P_L$
**Local Mapping:** Local Keyframe Culling

- Reduce BA complexity and limit the number of keyframes
- Culling policy
  - Any keyframe in $K_c$ whose 90% of the map points can be seen in at least three other keyframes

Outline

- Background
- Introduction
- Tracking
- Local mapping
- **Loop closing**
- Experiments and Results
Loop Closing

Loop closing is the act of correctly asserting that a vehicle has returned to a previously visited location

Why close loops?

- Previously visited location gets remapped in wrong global location
- Error accumulates out-of-bound
- Incorrect loop detection is even more harder to recover.
Loop Closing in ORB-SLAM2

Loop Detection

Loop Detection

New Keyframe

Compute similarity between all neighbors in Costability Graph

$\delta_{min}$

Loop candidates less than or equal to $\delta_{min}$ difference

For each candidate

- ORB correspondence matching
- First closed form solution using Horn's Method
- Found Similarity transform with enough inliers

Yes

No

Costability Graph

All Keyframes

Discard directly connected keyframes

Loop Detection

Compute SE3

Query Database
Loop Correction

- New Keyframe
- Fuse Map Points
  - Insert NEW edges
  - Covisibility Graph
  - Correct keyframe pose
  - Project map points of neighbors to keyframe
  - Fuse matched map points that are inliers

- Detected Loop
- Updated edges
- Propagate correction to all neighbors
- All Keyframes

Full Bundle Adjustment

- Optimize all KeyFrames and Points in the map
- Performed on separate thread after loop closure
- If new loop is detected, abort full BA and start again.
Outline

- Background
- Introduction
- Tracking
- Local mapping
- Loop closing
- Experiments and Results

Experiments and Results

<table>
<thead>
<tr>
<th>Error (Units)</th>
<th>$t_{rel}$ (deg/100m)</th>
<th>$t_{abs}$ (m)</th>
<th>$\tau_{rel}$ (%)</th>
<th>$\tau_{abs}$ (deg/100m)</th>
<th>$\tau_{abs}$ (m)</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0.70</td>
<td>0.25</td>
<td>1.3</td>
<td>0.63</td>
<td>0.26</td>
</tr>
<tr>
<td>01</td>
<td>1.39</td>
<td>0.21</td>
<td>10.4</td>
<td>2.36</td>
<td>0.36</td>
</tr>
<tr>
<td>02</td>
<td>0.76</td>
<td>0.23</td>
<td>5.7</td>
<td>0.79</td>
<td>0.23</td>
</tr>
<tr>
<td>03</td>
<td>0.71</td>
<td>0.18</td>
<td>0.6</td>
<td>1.01</td>
<td>0.28</td>
</tr>
<tr>
<td>04</td>
<td>0.48</td>
<td>0.13</td>
<td>0.2</td>
<td>0.38</td>
<td>0.31</td>
</tr>
<tr>
<td>05</td>
<td>0.40</td>
<td>0.16</td>
<td>0.8</td>
<td>0.64</td>
<td>0.18</td>
</tr>
<tr>
<td>06</td>
<td>0.51</td>
<td>0.15</td>
<td>0.8</td>
<td>0.71</td>
<td>0.18</td>
</tr>
<tr>
<td>07</td>
<td>0.50</td>
<td>0.28</td>
<td>0.5</td>
<td>0.56</td>
<td>0.29</td>
</tr>
<tr>
<td>08</td>
<td>1.05</td>
<td>0.32</td>
<td>3.6</td>
<td>1.11</td>
<td>0.31</td>
</tr>
<tr>
<td>09</td>
<td>0.87</td>
<td>0.27</td>
<td>3.2</td>
<td>1.14</td>
<td>0.25</td>
</tr>
<tr>
<td>10</td>
<td>0.60</td>
<td>0.27</td>
<td>1.0</td>
<td>0.72</td>
<td>0.33</td>
</tr>
</tbody>
</table>

- Comparison with previously most successful open source stereo SLAM--LSD SLAM

KITTI dataset
Experiments and Results

- Generated camera trajectory compared with ground truth

ORB-SLAM
Thank You!!!