Velodyne SLAM

2017.2.13

김태원
Overview

- What is SLAM (Simultaneous Localization and Mapping)

$k-1$  $k$  Pose estimation k-1 to k  $\Delta T$
Overview

- What is SLAM (Simultaneous Localization and Mapping)
Welcome to the KITTI Vision Benchmark Suite!

We take advantage of our autonomous driving platform Annieway to develop novel challenging real-world computer vision benchmarks. Our tasks of interest are: stereo, optical flow, visual odometry, 3D object detection and 3D tracking. For this purpose, we equipped a standard station wagon with two high-resolution color and grayscale video cameras. Accurate ground truth is provided by a Velodyne laser scanner and a GPS localization system. Our datasets are captured by driving around the mid-size city of Karlsruhe, in rural areas and on highways. Up to 15 cars and 30 pedestrians are visible per image. Besides providing all data in raw format, we extract benchmarks for each task. For each of our benchmarks, we also provide an evaluation metric and this evaluation website. Preliminary experiments show that methods ranking high on established benchmarks such as Middlebury perform below average when being moved outside the laboratory to the real world. Our goal is to reduce this bias and complement existing benchmarks by providing real-world benchmarks with novel difficulties to the community.
Data Category: City

Before browsing, please wait some moments until this page is fully loaded.

2011_09_26_drive_0001 (0.4 GB)
- Length: 114 frames (00:11 minutes)
- Image resolution: 1392 x 512 pixels
- Labels: 12 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 2 Cyclists, 1 Trams, 0 Misc
- Downloads: [unsynced-unrectified data] [synced-rectified data] [calibration] [tracklets]

2011_09_26_drive_0002 (0.3 GB)
- Length: 83 frames (00:08 minutes)
- Image resolution: 1392 x 512 pixels
- Labels: 1 Cars, 0 Vans, 0 Trucks, 0 Pedestrians, 0 Sitters, 2 Cyclists, 0 Trams, 0 Misc
- Downloads: [unsynced-unrectified data] [synced-rectified data] [calibration] [tracklets]

2011_09_26_drive_0005 (0.6 GB)
- Length: 160 frames (00:16 minutes)
- Image resolution: 1392 x 512 pixels
- Labels: 9 Cars, 3 Vans, 0 Trucks, 2 Pedestrians, 0 Sitters, 1 Cyclists, 0 Trams, 0 Misc
- Downloads: [unsynced-unrectified data] [synced-rectified data] [calibration] [tracklets]

2011_09_26_drive_0009 (1.8 GB)
- Length: 453 frames (00:45 minutes)
- Image resolution: 1392 x 512 pixels
- Labels: 89 Cars, 3 Vans, 2 Trucks, 3 Pedestrians, 0 Sitters, 0 Cyclists, 0 Trams, 1 Misc
- Downloads: [unsynced-unrectified data] [synced-rectified data] [calibration] [tracklets]
### Additional information used by the methods

- **Stereo:** Method uses left and right (stereo) images
- **Laser Points:** Method uses point clouds from Velodyne laser scanner
- **Loop Closure Detection:** This method is a SLAM method that detects loop closures
- **Additional training data:** Use of additional data sources for training (see details)

<table>
<thead>
<tr>
<th>Method</th>
<th>Setting</th>
<th>Code</th>
<th>Translation</th>
<th>Rotation</th>
<th>Runtime</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>V-LOAM</td>
<td>📸</td>
<td>🎥</td>
<td>0.60 %</td>
<td>0.0016 [deg/m]</td>
<td>0.1 s</td>
<td>2 cores @ 2.5 Ghz (C/C++)</td>
</tr>
<tr>
<td>LOAM</td>
<td>📸</td>
<td>🎥</td>
<td>0.70 %</td>
<td>0.0017 [deg/m]</td>
<td>0.1 s</td>
<td>2 cores @ 2.5 Ghz (C/C++)</td>
</tr>
<tr>
<td>SOFT2</td>
<td>📸</td>
<td>🎥</td>
<td>0.81 %</td>
<td>0.0022 [deg/m]</td>
<td>0.1 s</td>
<td>2 cores @ 2.5 Ghz (C/C++)</td>
</tr>
<tr>
<td>GDVO</td>
<td>📸</td>
<td>🎥</td>
<td>0.86 %</td>
<td>0.0031 [deg/m]</td>
<td>0.09 s</td>
<td>1 core @ &gt;3.5 Ghz (C/C++)</td>
</tr>
<tr>
<td>HypERROCC</td>
<td>📸</td>
<td>🎥</td>
<td>0.88 %</td>
<td>0.0027 [deg/m]</td>
<td>0.25 s</td>
<td>2 cores @ 2.0 Ghz (C/C++)</td>
</tr>
<tr>
<td>SOFT</td>
<td>📸</td>
<td>🎥</td>
<td>0.88 %</td>
<td>0.0022 [deg/m]</td>
<td>0.1 s</td>
<td>2 cores @ 2.5 Ghz (C/C++)</td>
</tr>
<tr>
<td>RGBROCC</td>
<td>📸</td>
<td>🎥</td>
<td>0.88 %</td>
<td>0.0025 [deg/m]</td>
<td>0.3 s</td>
<td>2 cores @ 2.0 Ghz (C/C++)</td>
</tr>
<tr>
<td>EDVO</td>
<td>📸</td>
<td>🎥</td>
<td>0.89 %</td>
<td>0.0030 [deg/m]</td>
<td>0.1 s</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
</tr>
<tr>
<td>vox2</td>
<td>📸</td>
<td>🎥</td>
<td>0.94 %</td>
<td>0.0021 [deg/m]</td>
<td>0.2 s</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
</tr>
</tbody>
</table>

Overview

HDL-64E

KEY FEATURES

› 64 Channels
› 120m range
› 2.2 Million Points per Second
› 360° Horizontal FOV
› 26.9° Vertical FOV
› 0.08° angular resolution (azimuth)
› <2cm accuracy
› ~0.4° Vertical Resolution
› User selectable frame rate
› Rugged Design
Overview

- Point cloud data의 특징
  - Set of local 3d points
  - Unspecified data
  - Non sorted data
Overview

- Feature based method
  - Find matching pair
  - Minimize cost

\[
\arg\min_{T} \left( \sum_{i} \left\| TP_{i}^{k} - P_{i}^{k-1} \right\|^2 \right)
\]

\( N \): num of matching pair
Proposed method

- Minimize point to plane distance
  - No need matching pair
Proposed method

- Iterative closest point (ICP)
  1. Find nearest plane
  2. Calculate point to plane distance

Search problem!! -> too slow
Non-differentiable!! -> Heuristic method
Proposed method

- Proposed method

1. Cylindrical projection
2. Calculate point to plane distance

No search problem!! -> Fast
Differentiable!! -> Numerical optimization

\[
\argmin_T \left( \sum_{i=1}^{N} f(T)^2 \right)
\]
\[ \hat{s}_{k-1,i} = N_{k-1}(P_{k-1,i}) \]

\[ e_i = f(\hat{P}_{k-1,i}, \hat{s}_{k-1,i}) \]

\[ \hat{P}_{k-1,i} = C(\hat{P}_{k-1,i}) \]

G: Rigid body transformation

\[ \hat{P}_{k-1,i} = T(P_{k,i}, G) \]

\[ \hat{P}_{k-1,i} = G \cdot P_{k,i} \]

\[ G = \exp(\xi) \]
\[ \hat{s}_{k-1,i} = N_{k-1}(\hat{p}_{k-1,i}) \]

\[ f : \text{Plane to point distance} \]

\[ \varepsilon_i = f(\hat{P}_{k-1,i}, \hat{s}_{k-1,i}) \]

\[ \hat{P}_{k-1,i} = T(P_{k,i}, G) \]

\[ G = \exp(\xi) \]

\[ C : \text{Cylindrical projection} \]

\[ \hat{p}_{k-1,i} = S(\hat{p}'_{k-1,i}) \]

\[ \hat{p}'_{k-1,i} = C(\hat{p}'_{k-1,i}) \]

\[ \hat{p}'_{k-1,i} = [x_n \ y_n \ z_n]^T, \hat{p}_{k-1,i} = [x \ y \ z]^T \]

\[ [x_n] = \frac{1}{\sqrt{x^2 + y^2}} [x] \]

\[ [y_n] = \sqrt{x^2 + y^2} [y] \]

\[ [z_n] = \sqrt{x_n^2 + y_n^2} [z] \]

\[ \theta = \tan^{-1}\left(\frac{y_n}{x_n}\right), -\pi \leq \theta \leq \pi, \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -s_\theta \theta + c_\theta \\ -s_\theta z + c_\theta \end{bmatrix} \]

\[ s_\theta = \text{width} / 2\pi \]

\[ c_\theta = \text{width} / 2 \]

\[ s_z = \text{height} / \tan(vFov_{\max} - vFov_{\min}) \]

\[ c_z = \text{height} \frac{vFov_{\max}}{vFov_{\max} - vFov_{\min}} \]

\[ vFov_{\max} = 2, vFov_{\min} = -24.9 \]
\[ \hat{s}_{k-1,i} = N_{k-1} \left( \hat{p}_{k-1,i} \right) \]

\[ \epsilon_i = f \left( \hat{P}_{k-1,i}, \hat{s}_{k-1,i} \right) \]

\[ \hat{P}_{k-1,i} = C \left( \hat{P}_{k-1,i} \right) \]

\[ G = \exp(\xi) \]

- Plane to point distance

\[ \epsilon_i = f \left( \hat{P}_{k-1,i}, \hat{s}_{k-1,i} \right) \]

\[ \hat{P}_{k-1,i} = \begin{bmatrix} x & y & z \end{bmatrix}^T, \hat{s}_{k-1,i} = \begin{bmatrix} a & b & c & d \end{bmatrix}^T \]

\[ \epsilon_i = \frac{|ax + by + cz + d|}{\sqrt{a^2 + b^2 + c^2}} \]
\[
\hat{s}_{k-1,i} = N_{k-1}(\hat{P}_{k-1,i})
\]

\[
\epsilon_i = f(\hat{P}_{k-1,i}, \hat{s}_{k-1,i})
\]

\[
\hat{P}_{k-1,i} = C(\hat{P}_{k-1,i})
\]

\[
G = \exp(\xi)
\]

\[
P_{k,i} = T(P_{k,i}, G)
\]

\[
\xi^* = \text{argmin}_{\xi} E(\xi)
\]

\[
E(\xi) = \sum \epsilon_i^2 = \sum \left\{ f(\hat{P}_{k-1,i}, \hat{s}_{k-1,i}) \right\}^2
\]

\[
E(\xi) = \sum \left\{ f \left( T(P_k, \xi), N_{k-1} \left( S \left( C \left( T(P_k, \xi) \right) \right) \right) \right) \right\}^2
\]

- **Cost function**

- **Jacobian**

\[
J = \frac{\partial f}{\partial \xi} = \frac{\partial f}{\partial \hat{P}_{k-1,i}} \frac{\partial \hat{P}_{k-1,i}}{\partial \xi} + \frac{\partial f}{\partial \hat{s}_{k-1,i}} \frac{\partial \hat{s}_{k-1,i}}{\partial \xi}
\]

\[
= \frac{\partial f}{\partial \hat{P}_{k-1,i}} \frac{\partial \hat{P}_{k-1,i}}{\partial G} \frac{\partial G}{\partial \xi} + \frac{\partial f}{\partial \hat{s}_{k-1,i}} \frac{\partial \hat{s}_{k-1,i}}{\partial \xi} \frac{\partial \hat{P}_{k-1,i}}{\partial \hat{G}} \frac{\partial \hat{G}}{\partial \xi} + \frac{\partial f}{\partial \hat{P}_{k-1,i}} \frac{\partial \hat{P}_{k-1,i}}{\partial \hat{G}} \frac{\partial \hat{G}}{\partial \xi} + \frac{\partial f}{\partial \hat{s}_{k-1,i}} \frac{\partial \hat{s}_{k-1,i}}{\partial \xi} \frac{\partial \hat{G}}{\partial \xi}
\]
\[
J = \frac{\partial f}{\partial \xi} = \frac{\partial \hat{P}_{k-1,i}}{\partial \xi} + \frac{\partial f}{\partial \hat{s}_{k-1,i}} \frac{\partial \hat{s}_{k-1,i}}{\partial \xi}
\]

2. \[
\frac{\partial f}{\partial \hat{s}_{k-1,i}} = \begin{bmatrix}
\frac{\partial f}{\partial \hat{s}_{k-1,i,a}} & \frac{\partial f}{\partial \hat{s}_{k-1,i,b}} & \frac{\partial f}{\partial \hat{s}_{k-1,i,c}} & \frac{\partial f}{\partial \hat{s}_{k-1,i,d}} & \frac{\partial f}{\partial \hat{s}_{k-1,i,e}} & \frac{\partial f}{\partial \hat{s}_{k-1,i,f}}
\end{bmatrix}
\]

3. \[
\frac{\partial \hat{s}_{k-1,i}}{\partial \hat{p}_{k-1,i}} = \begin{bmatrix}
\frac{\partial \hat{s}_{k-1,i,a}}{\partial \hat{p}_{k-1,i}} & \frac{\partial \hat{s}_{k-1,i,b}}{\partial \hat{p}_{k-1,i}} & \frac{\partial \hat{s}_{k-1,i,c}}{\partial \hat{p}_{k-1,i}} & \frac{\partial \hat{s}_{k-1,i,d}}{\partial \hat{p}_{k-1,i}} & \frac{\partial \hat{s}_{k-1,i,e}}{\partial \hat{p}_{k-1,i}} & \frac{\partial \hat{s}_{k-1,i,f}}{\partial \hat{p}_{k-1,i}}
\end{bmatrix}
\]

4. \[
\frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}'_{k-1,i}} = \begin{bmatrix}
\frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}'_{k-1,i,a}} & \frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}'_{k-1,i,b}} & \frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}'_{k-1,i,c}} & \frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}'_{k-1,i,d}} & \frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}'_{k-1,i,e}} & \frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}'_{k-1,i,f}}
\end{bmatrix}
\]

5. \[
\frac{\partial \hat{p}'_{k-1,i}}{\partial \hat{p}_{k-1,i}} = \begin{bmatrix}
\frac{\partial \hat{p}'_{k-1,i}}{\partial \hat{p}_{k-1,i,a}} & \frac{\partial \hat{p}'_{k-1,i}}{\partial \hat{p}_{k-1,i,b}} & \frac{\partial \hat{p}'_{k-1,i}}{\partial \hat{p}_{k-1,i,c}} & \frac{\partial \hat{p}'_{k-1,i}}{\partial \hat{p}_{k-1,i,d}} & \frac{\partial \hat{p}'_{k-1,i}}{\partial \hat{p}_{k-1,i,e}} & \frac{\partial \hat{p}'_{k-1,i}}{\partial \hat{p}_{k-1,i,f}}
\end{bmatrix}
\]

6. \[
\frac{\partial \hat{p}_{k-1,i}}{\partial \hat{p}_{k-1,i}} = 3x12
\]

7. \[
\frac{\partial \hat{G}}{\partial \xi} = 12x6
\]
6. \[ \frac{\partial \hat{P}_{k-1,i}}{\partial G} = \begin{bmatrix}
    P_{k,i,y} & 0 & 0 & P_{k,i,z} & 0 & 0 & 1 & 0 & 0 \\
    0 & P_{k,i,x} & 0 & 0 & P_{k,i,y} & 0 & 0 & P_{k,i,z} & 0 & 0 & 1 & 0 \\
    0 & 0 & P_{k,i,x} & 0 & 0 & P_{k,i,y} & 0 & 0 & P_{k,i,z} & 0 & 0 & 1 & 0 \\
\end{bmatrix} \]

7. \[ \frac{\partial G}{\partial \xi} = \begin{bmatrix}
    0 & 0 & 0 & 0 & r_{31} & -r_{21} \\
    0 & 0 & 0 & -r_{31} & 0 & r_{11} \\
    0 & 0 & 0 & r_{21} & -r_{11} & 0 \\
    0 & 0 & 0 & 0 & r_{32} & -r_{22} \\
    0 & 0 & 0 & -r_{32} & 0 & r_{12} \\
    0 & 0 & 0 & r_{22} & -r_{12} & 0 \\
    0 & 0 & 0 & 0 & r_{33} & -r_{23} \\
    0 & 0 & 0 & -r_{33} & 0 & r_{13} \\
    0 & 0 & 0 & r_{23} & -r_{13} & 0 \\
    1 & 0 & 0 & 0 & t_{z} & -t_{y} \\
    0 & 1 & 0 & -t_{z} & 0 & t_{x} \\
    0 & 0 & 1 & t_{y} & -t_{x} & 0 \\
\end{bmatrix} \]
Algorithm 5: Levenberg-Marquardt algorithm

\textbf{input}: $f : \mathbb{R}^n \rightarrow \mathbb{R}$ a function such that $f(x) = \sum_{i=1}^{m} (f_i(x))^2$

where all the $f_i$ are differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$

$x^{(0)}$ an initial solution

\textbf{output}: $x^*$, a local minimum of the cost function $f$

1 \textbf{begin}
2 \hspace{1em} $k \leftarrow 0$
3 \hspace{1em} $\lambda \leftarrow \max \text{diag}(J^TJ)$
4 \hspace{1em} $x \leftarrow x^{(0)}$
5 \hspace{1em} while \( \text{stop-crit and } (k < k_{\text{max}}) \) \ do
6 \hspace{2em} \text{Find } \delta \text{ such that } (J^TJ + \lambda \text{diag}(J^TJ))\delta = J^Tf$
7 \hspace{2em} $x' \leftarrow x + \delta$
8 \hspace{2em} if \( f(x') < f(x) \) then
9 \hspace{3em} $x \leftarrow x'$
10 \hspace{3em} $\lambda \leftarrow \frac{\lambda}{\nu}$
11 \hspace{2em} else
12 \hspace{3em} $\lambda \leftarrow \nu \lambda$
13 \hspace{2em} $k \leftarrow k + 1$
14 \textbf{return} $x$
15 \textbf{end}

[Gradient descent 방법]
$$p_{k+1} = p_k - 2\lambda (J^TJ) + (f'_i(p_k) r_i(p_k), k \geq 0 \quad \text{--- (16')}$$

[가우스-뉴턴법]
$$p_{k+1} = p_k - (J^TJ)^{-1}J^Tr(p_k), \quad k \geq 0 \quad \text{--- (24')}$$

[Levenberg 방법]
$$p_{k+1} = p_k - (J^TJ + \mu I)^{-1}J^Tr(p_k), \quad k \geq 0 \quad \text{--- (30)}$$

[Levenberg-Marquardt 방법]
$$p_{k+1} = p_k - (J^TJ + \mu_k I)^{-1}J^Tr(p_k), \quad k \geq 0 \quad \text{--- (31)}$$
Accuracy

- KITTI Benchmark ground truth
- Proposed method
Problem

- Non planar area
- Moving object
- Occlusion
Flatness filtering

Plane normal

Flatness

Standard deviation
Moving object Problem
Moving object filtering
Conclusion

- Point cloud를 이용한 새로운 pose estimation 알고리즘 제안

- 제안한 알고리즘은 센서 퓨전이나 GPU 가속 없이 빠르고 정확하게 동작

- Moving object removal, loop closing detection등 몇 가지 문제에 대한 해결이 필요하다.