Proposal for GSoC 2019

[JdeRobot] Project #6 Improving SLAM-testbed tool

About Me

Name: Shunkai Li

Email: lishunkaipku@gmail.com

Github URL: https://github.com/Lishunkai

Education: Peking University, Beijing, China **2018 - 2021**

Master of Computer Science Research interests: SLAM/VO, machine learning

University of Oxford (as an Exchange Student), Oxford, UK **2016**

Was awarded as the Excellent Student of Oxford Summer Institute (Top 5%).

Nankai University, Tianjin, China **2014 - 2018**

Major: Opto-electronic Information Science and Engineering (Bachelor)

Cumulated Grade Point Average: 90.94/100 Ranking:1/33

Related Experience:

1. **Self-supervised online learning for VO/SLAM** **2019**
   Current research
2. **Stereo SLAM for online 3D reconstruction** **2019**
   Key words: visual SLAM, on-the-fly 3D reconstruction
3. **Stereo SLAM for indoor robotics** **2019**
   Key words: visual SLAM, 3D reconstruction, visual localization
4. **Beyond tracking: selecting memory and refining poses for deep visual odometry (CVPR 2019 oral)** **2018**
   Key words: supervised learning, LSTM, attention-based aggregation
5. **Research on visual SLAM based on deep learning (dissertation)** **2017-2018**
   Key words: feature-based SLAM, dense depth creation from sparse points
6. **Application of monocular SLAM in augmented reality (Internship in Samsung Research China)** **2017-2018**
   Key words: visual-inertial SLAM, plane detection and fitting
7. **Application of SLAM in autonomous driving (Internship in CalmCar)** **2017**
   Key words: feature-based SLAM, foreground object motion prediction

Paper:

- Fei Xue, Shunkai Li, Xin Wang, Qiuyuan Wang, Junqiu Wang, and Hongbin Zha. *Beyond Tracking: Selecting Memory and Refining Poses for Deep Visual Odometry.* CVPR 2019 oral
- Shunkai Li, Yifan Wang, Weichen Wu, and Yanmei Lianag. *Predictive Searching Algorithm for Fourier Ptychography*. Journal of Optics (SCI) Impact factor: 2.33

Skills: SLAM/VO/VIO C++ Python ROS Linux Unsupervised Learning

Project

Project Name: Project #6 Improving SLAM-testbed tool

Description: https://jderobot.org/GSoC-2019#Project_.236:_Improving_SLAM-testbed_tool

Programming language: C++

Main goal:

1) Refine current release
2) Increase the compatibility with new benchmarks and SLAM algorithms
3) Make SLAM-testbed an **independent** tool for evaluation, and make it a **plug-in** ROS package for other purposes.

Input: ground truth and calculated trajectories

Output: a score

Interface: a graphical tool

Outline and Proposal:

1. Data compatibility

There are many formats for trajectories and poses, and converting them into a unified format becomes a necessity for evaluation.

1) Types of poses

The commonly used formats include:

[1] 6DoF (Euler angle): timestamp t_x t_y t_z r_1 r_2 r_3

[2] 7DoF (quaternion): timestamp t_x t_y t_z q_x q_y q_z q_w

[3] 3*4 matrix: timestamp $\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{pmatrix}$

[4] 4*4 matrix: timestamp $\begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$

[5] Other formats

2) Prepare data for evaluation

Each type of above mentioned formats includes relative pose and global pose. The SLAM testbed tool should include a program or module to prepare data. The plug-in module is designed to be compatible with different types of poses and convert them into a unified format for trajectory creation and evaluation.

2. Trajectory alignment

The alignment of calculated and ground truth trajectories is a prerequisite for evaluation. It seems easy at first, but it is actually complicated, especially for monocular visual SLAM which has the scale ambiguity problem. According to my experience and related materials, the methods of trajectory alignment can be classified into several types:

1) First: scale alignment (for monocular visual SLAM)

There is no gold standard for scale alignment. Two common ways are usually used in practice:

[1] Alignment using all/multiple estimated states

This method tends to give a lower error if later an error metric for the whole trajectory (e.g., ATE) is used.

[2] Alignment using a single state

This method aligns two trajectories with only the first state. It gives an intuitive error distribution that the estimation error increases over time.

It is worth noticing that the alignment is based on a least squares solution, which is valid only when all states are of the same uncertainty. If we have the knowledge about the quality of the state estimate (e.g., covariance from SLAM), more sophisticated methods can be used to account for this process.

2) Trajectory alignment (for evaluation and visualization)

[1] Align the first state

[2] Align for the most similar shape of two trajectories

3. Data for evaluation

1) Benchmark datasets

The evaluation of SLAM algorithms often uses many datasets corresponding to different environments, such as:

[1] KITTI (outdoor, driving)

[2] EuRoC (indoor, industrial and living room)

[3] TUM (indoor)

[4] Oxford (outdoor, often used for loop-closing)

[5] NYU-v2-dataset (indoor)

[6] 7-Scenes (indoor)

The interface of SLAM-testbed should include choosing different datasets, loading and modifying them to uniform format automatically.

2) Self-collected data

The scenes from datasets are usually limited. In most cases, engineers and researchers are more willing to use self-collected data or use camera for real-time acquisition. Therefore, it is also important to make SLAM-testbed compatible with self-collected data.

4. Evaluation metrics

I summarize the following evaluation metrics according to my research experience. Some of them are commonly used at present, and others are new proposed criteria to evaluate SLAM algorithms more comprehensively.

1) Trajectory accuracy

[1] Absolute Trajectory Error (ATE)

[2] Relative Error (RE)

[3] Absolute Positional Error (APE)

[4] Relative Positional Error (RPE)

[5] Absolute Rotational Error (ARE)

[6] Relative Rotational Error (RRE)

[7] Completeness:

The ratio between the number of valid (threshold) poses and the total number

of all poses. Poses before the first initialization are not included.

[8] t_{rel}: average translational RMSE drift (%)

[9] r_{rel}: average rotational RMSE drift (°/100m)

[10] Translation error against path length

[11] Rotational error against path length

[12] Translation error against speed

[13] Rotational error against speed

2) Initialization quality

[1] t_{init}: initialization time

[2] $t_{init-conv}$: the time for scale to converge

Accurate scale is quite important in AR/VR/robotics applications. At the beginning, the scale usually fluctuates. The system can only be used after the scale converges.

[3] ϵ_{scale}: the quality of converged scale

3) Tracking robustness

[1] Re-localization Error

The tracking result should be consistent after recovering from lost status.

[2] Lost time

The smaller the better

4) Re-localization time

[1] Force to enter lost state

Manually add black frames.

[2] Re-localization time measurement

VIO/VI-SLAM tends to continue IMU propagation even without sufficient feature matches. Detect re-localization by the jump in trajectory.

5. Interface

Graphical interface can be developed to assess SLAM algorithms more effectively. The interface should include:

1) Selecting path of ground truth and calculated poses (maybe multiple trajectories from different SLAM algorithms)
2) Selecting types of input poses
3) For SLAM algorithms with scale ambiguity, choose the number of poses (starting from the beginning) that will be used in the trajectory alignment, from 1 to all poses
4) Choose the alignment type
5) Select evaluation metrics
6) Select path for saving results

6. Save results

After the evaluation is done, the results will be saved in the certain path. The items to save may include:

1) Trajectories
2) Plots
3) Tables

4) Numbers

7. Summary article and documentation

After the coding part is completed, a summary article and documentation will be written to describe the work we have done. The article will include the explanation of a brief introduction to 3D transformation and evaluation metrics, the details of the design of SLAM-testbed, and instructions of how to use this tool. I think my experience of writing academic papers will help me in this part. This article and documentation will be put on JdeRobot community or wherever appropriate.

P.S.:

I have read some related works on SLAM evaluation metrics and tested the evaluation tool developed by University of Zurich you mentioned. They are very helpful for me to get familiar with details and get new ideas for this project.

Expected Timeline

Date	Work
Present-May 6	• Get familiar with the JdeRobot code and the first version of SLAM-testbed. • Download related datasets about SLAM for preparation. • Keep diving into the first version of SLAM-testbed and trying to fix some issues.
May 6-May 27	• Discuss with mentor about the implementation of the details. Investigate the evaluation metrics and related works which we can refer to. • Settle down a pipeline for programming SLAM-testbed.
May 27-June 24	• Design the SLAM-testbed with basic evaluation metrics described above, and do validation on datasets. • Make SLAM-testbed compatible with ROS, so as to let users use self-collected data. • Write a document of evaluation.
July 1-22	• Fix some bugs. • Add more evaluation metrics. • Extend some new functions if needed. • Write a document of changes on the code.
July 26-August 19	• Update the document of SLAM-testbed. • Fix some bugs.
August 19-31	• Write a summary article and final documentation. • Buffer for unexpected delay. • Try to add some new functions if there is still time left.

Other Information

Working Time

I will be in China during this summer. So my working time is for GMT +8 timezone. According to the project description and my proposal, I believe that it will take about 10 weeks for me to complete this project. I will keep doing research on SLAM during the days when I wait for the result announced by GSoC, so it is not hard for me to learn some knowledge or analyze code related to SLAM. Therefore, I'm willing to start with some preparation about this project if necessary.

Reason for Participation

I love the field of 3D computer vision and I've been doing many projects, research, and internship about visual SLAM for two years. Besides, being a developer to make contributions to visual SLAM and robot development communities is one of my dream. GSoC 2019 provides me a great chance to make contributions to open source projects with mentorship from excellent developers all over the world, and Project #6 provided by JdeRobot is about SLAM. So I believe it is really amazing and valuable to participate in GSoC 2019 and develop open source programs to JdeRobot. If I've been selected, I will try my best to complete this project.

I've been using JdeRobot these days, it is really a good tool for robotics and computer vision, and more importantly, it is ROS friendly. So I've decided to use JdeRobot to during my study and research on SLAM in Peking University. Since I will continue using it in the future, I'm willing to keep maintaining and improving JdeRobot after this summer program ends.

I'm looking forward to working on the project during GSoC 2019!