

Mapping car (camera part)

Final Report

Weihong Tang , Renzhi Tang
Shanghaitech University

Abstract

We try to use the Ladybug5+ to construct the spherical imaging system that will be worked on the mapping car. Combined with the lidar data, we will use our spherical image to generate the colored point cloud. And we will also get images from eight pointgrey cameras around.

1 Introduction

We have found the ROS driver of ladybug5+, which has some bugs. For example, it will miss some packages and can not set to the JPEG-compressed imaging mode. Then we are going to fix them and access the JPEG data of the 6 cameras of the Ladybug5+ in the Linux operation system. In addition, we modify the driver to get the spherical images, which we can combine with the lidar data to generate colored point cloud. And before these, we should first mount of all the cameras on the designed rig. Then, we run the ORB-Slam with the data record from our system and get the estimated trajectory. Ground true trajectory is obtained by the motion capture system. For the evaluation, RPG Trajectory Evaluation toolbox is used to get the Absolute Trajectory Error (ATE) and Relative/Odometry Error (RE). Eventually, We plan to collect three the datasets involving different kinds of environment: STAR lab(test), basement parking(indoor) and Campus(outdoor) of Shanghaitech University.

2 State of the Art

2.1 ORB-SLAM

In terms of the feature matching, there are lots of SLAM algorithm to use. For example: LSD-SLAM which uses direct image alignment coupled with filtering-based estimation of semi-dense depth maps[1]. However, we will use the algorithm of ORB-SLAM, a feature-based monocular simultaneous localization and mapping (SLAM) system that operates in real time, in small and large indoor and outdoor environments[2]. It can be operated in real-time and in

large environments. Compared other algorithms, it performs more efficient, simple, and reliable. However, it's just for the monocular SLAM system, which means it can not be used for stereo or RGB-D cameras. And then the ORB-SLAM2, which is the development of ORB-SLAM, has been put forward. ORB-SLAM2 is a complete simultaneous localization and mapping(SLAM) system for monocular, stereo and RGB-D cameras, including map reuse, loop closing, and relocalization capabilities[3]. Compared with ORB-SLAM mentioned in [2], ORB-SLAM2 has added the solutions to solve SLAM problems for stereo and RGB-D cameras and the modularization has been slightly better. Also, ORB-SLAM2 has a ROS package, it's convenient and can be set up in a short time.

2.2 RPG Trajectory Evaluation

For system evaluation, we use the trajectory evaluation approach based on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry[5]. More importantly, the authors have released a trajectory evaluation toolbox that implements the methods on the GitHub, which is friendly to us.

Specifically, it includes different trajectory alignment methods (rigid-body, similarity and yaw-only rotation) and uses two error metrics, Absolute Trajectory Error (ATE) and Relative/Odometry Error (RE) for evaluation. It can be used to analyze a single trajectory estimate, as well as compare different algorithms on many datasets with one command. The user only needs to provide the groundtruths and estimates of desired format and specify the trajectory alignment method. The toolbox generates (almost) paper-ready plots and tables.

2.3 ROS packages

- Ladybug5+:
The package is about the driver of the Ladybug5+, which we found on the GitHub. It is an expansion of the Ladybug driver in Autoware driver. The key changes are more config options and added support for the Ladybug 5+ camera. We use the package to invoking the SDK of the Ladybug5+ to get the images as well as adjust the quality of image and the frame rate.
- orb_slam2_ros:
This is the ROS implementation of the ORB-SLAM2 real-time SLAM library for Monocular, Stereo and RGB-D cameras that computes the camera trajectory and a sparse 3D reconstruction (in the stereo and RGB-D case with true scale). It is able to detect loops and relocalize the camera in real time. This implementation removes the Pangolin dependency, and the original viewer. All data I/O is handled via ROS topics. For visualization you can use RViz. This repository is maintained by Lennart Haller on behalf of appliedAI.[7]
- camera_calibration:
This package uses OpenCV camera calibration. Camera_calibration allows easy calibration of monocular or stereo cameras using a checkerboard calibration target.

3 System Description

The spherical imaging system can acquire the images from 6 cameras of the Ladybug5+ spherical cameras. Then the system can record images from any cameras or stitch 6 images into a spherical image.

3.1 Hardware

Our spherical imaging system is composed of 3 Ladybug5+. The positions are following:

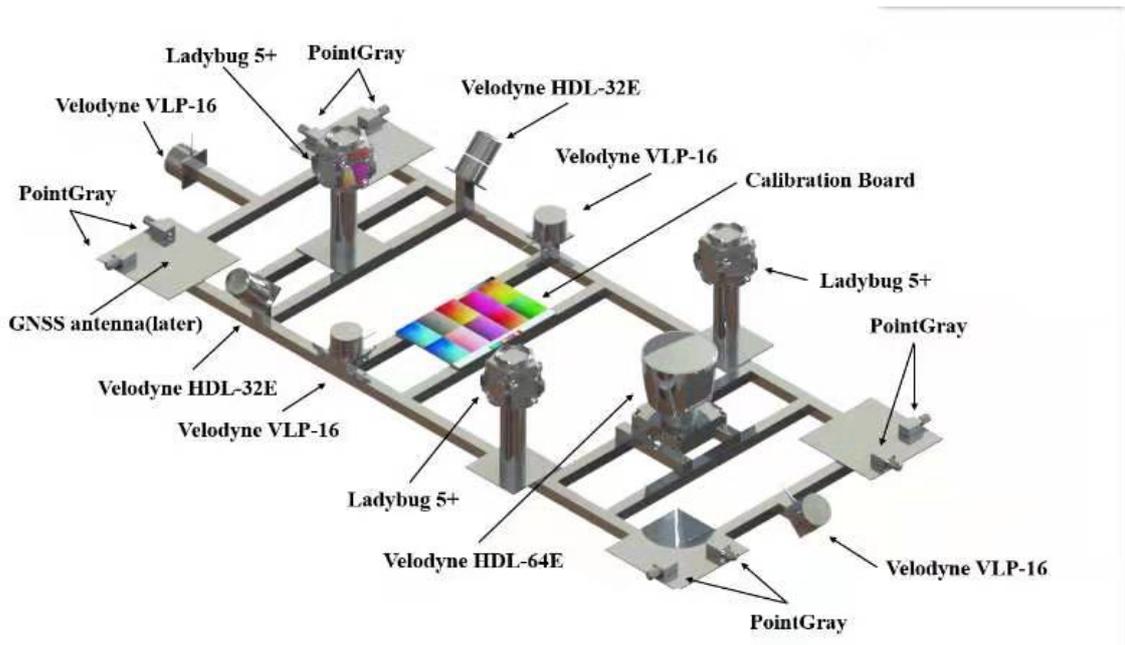


Figure 1: State transitions in the model

Finally, the hardware will be used to mount on an BYD e6 electric car for data collection. And the final product is shown in Figure 2.

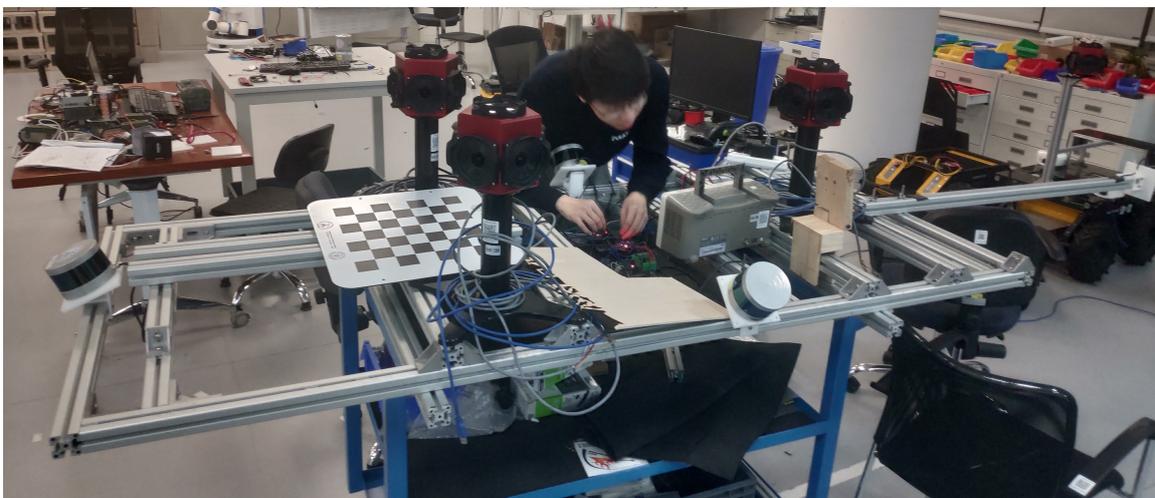


Figure 2: Final hardware

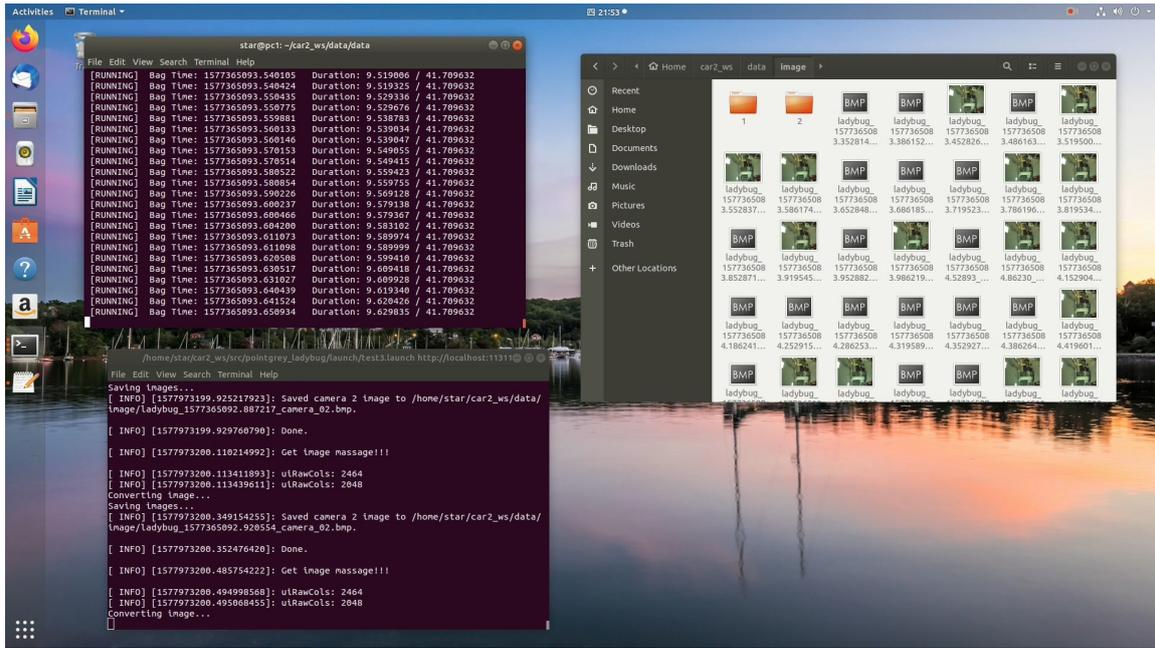


Figure 3: Image splitting

3.2 Software

In the section, we will introduce our code for data acquisition, image split and stitch. This is aim to make our work clear and make it easy for future use.

By searching with google and in github, there is only one ladybug5+ driver for data acquisition which can be used with ROS. However, it can not acquire data at 30Hz. The code sets the dataformat of the acquired image at `LADYBUG_DATAFORMAT_RAW8` which from the official technical reference can only get 13Hz at most. This driver can not meet our needs. By checking Ladybug SDK API document, we finally decided to write a new one for JPEG data acquisition. The following 7 paramters can be set in ROS launch file: `jpeg_percent`, `framerate`, `use_auto_framerate`, `shutter_time`, `use_auto_shutter_time`, `gain_amount`, `use_auto_gain`.

3.2.1 Data Acquisiton

And for record the JPEG compressed data in ROS, no message type can be used to transport ladybug image data between ROS nodes. So we define a new message type in ROS called `Ladybug5PImage`, which contains all infomation from the raw image data. For example: compressed 4 channel image data ,timestamp, GPS information and so on. Instead of publishing processed images using `senser_msg/Image` message type, we publish the raw data for meeting the requirement of frequency(30Hz). If we process first, the frequency will decrease to at most 3Hz.

3.2.2 Image Split

Splitting images function has also been realized in our driver which is shown in Figure 3. After recording data, replay the rosbag and the driver can split the special format compressed JPEG data into 6 individual images. We publish the 6 individual images in 6 different topics in

ROS and we also reserve the interface for saving image which can be seen obviously in the c++ code file. It is for future use and in this project we did not use it.

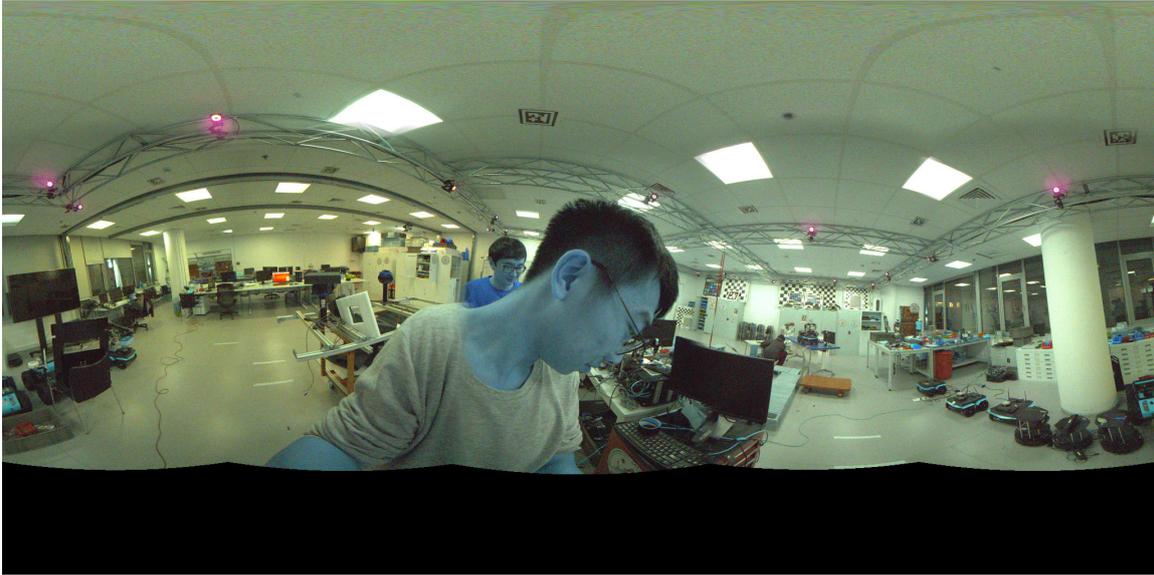


Figure 4: Panoramic image

3.2.3 Image Stitch

Stitching 6 images to a panoramic image is generally using the API in Ladybug SDK called 'ladybugUpdateTextures' and 'ladybugRenderOffScreenImage'. This is not our main work and not use in ORB-SLAM2. We think it will be used in the future to run the SLAM using panoramic images or combine the panoramic image with LiDAR data to build the colored pointcloud. Like the splitted images, the panoramic image can also be published in ROS and the interface for saving is also reserved in the c++ code.

4 System Evaluation

4.1 Data Record Frequency

From Ladybug's technical reference(Section 8.1), it's clear that ladybug5+ can acquire images at 30Hz. By running the code we wrote, which is mentioned in Section 3.2.2, the frame rate of image acquisition reaches 30Hz while monitored by the code 'rostopic hz'. To ensure 30Hz, ladybug5+ is set the following parameters:

- ladybug dataformat : LADYBUG_DATAFORMAT_COLOR_SEP_JPEG8
- ladybug frame rate : (int) 30
- ladybug gain : (int) 30
- ladybug JPEG quality : (int) 90

Additionally, it's important to point out that the frame rate is not always at 30Hz. It depends on the state of USB cables. If the state of the cable is not good, which may depends on the timing to

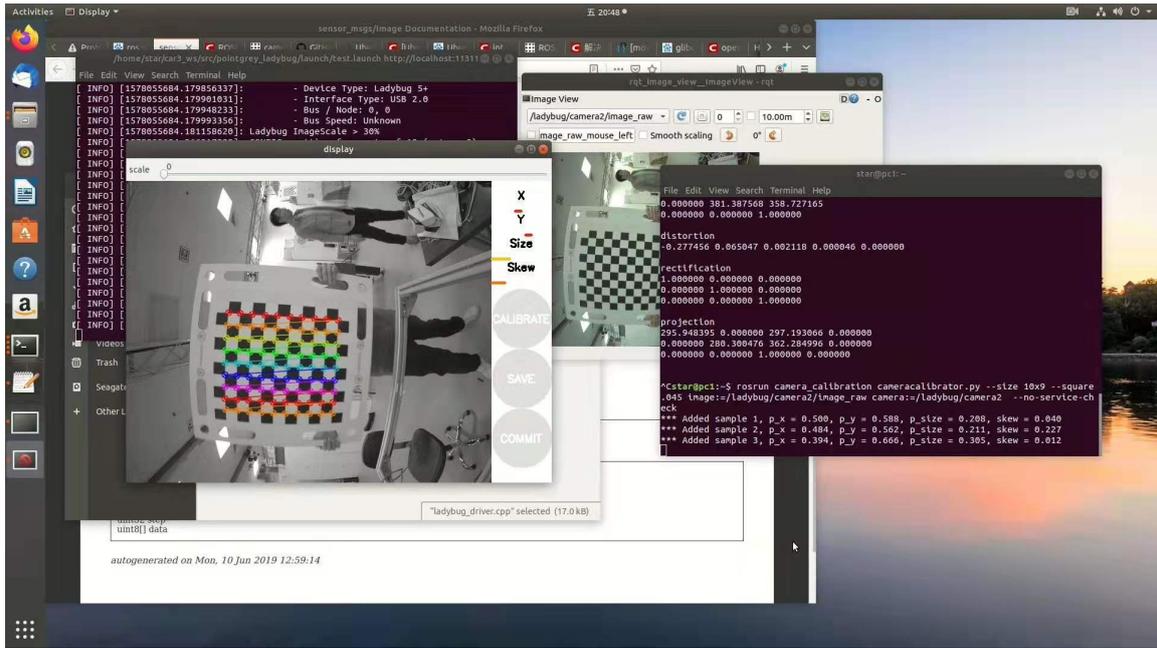


Figure 5: Calibration with 9×10 45mm chessboard

plugin the cable, the frame rate will be around 20Hz. It's still higher than the 13Hz, the highest rate to acquire full image without using JPEG compression.

4.2 ORB-SLAM2

For the system evaluation, a SLAM system generally outputs the estimated camera trajectory along with an estimate of the resulting map. Accurate ground truth maps are obtained by Motion Capture System. Before recording data for ORB-SLAM2, we first calibrate the one of the six cameras which we used later. To be specific, the index of the camera is 2. For calibration, a 9×10 45mm chessboard is used to get intrinsic parameters of the choosed camera with ROS package camera_calibraiton see Figure 5. The camera matrix we get is:

$$\begin{bmatrix} 1250.185184 & 0.000000 & 1201.451330 \\ 0.000000 & 1247.012318 & 1057.979207 \\ 0.000000 & 0.000000 & 1.000000 \end{bmatrix}$$

And the distortion is:

$$\begin{bmatrix} -0.275641 & 0.065761 & -0.000272 & 0.002035 & 0.000000 \end{bmatrix}$$

A more detail result is in the calibration file folder in project gitlab.

After calibration, we record two data bags contain both ground truth and ladybug image data named 'Data1' and 'Data2'. Both of these two bags are recorded by handhold ladybug5+ walking around a desk.

- Data1: 2 clockwise trajectories following with 2 anticlockwise trajectories
- Data2: 2 clockwise trajectories

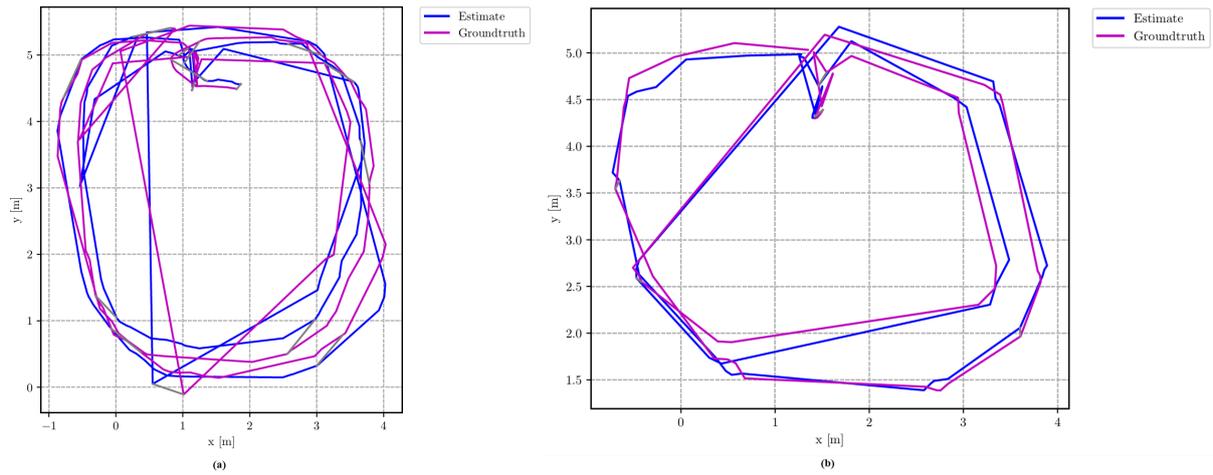


Figure 6: Estimated trajectory and ground truth.(a) Data1 (b)Data2

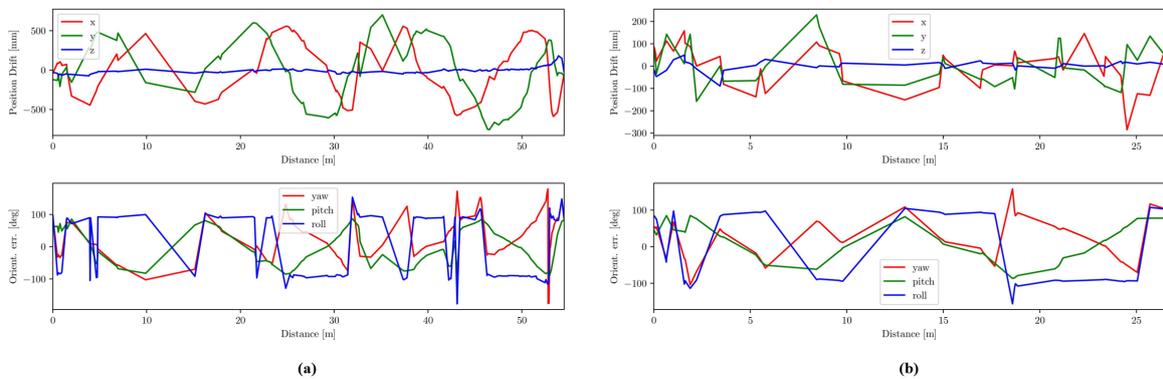


Figure 7: Absolute position and rotation error.(a) Data1 (b)Data2

By playing rosbag at x0.1 speed, we split the compressed JPEG data to 6 images and publish them. Using the published topic, we do the ORB-SLAM2 and get the estimated trajectories in Figure 6. This takes lots of time because we play the bag at x0.1 speed. And the ground truth trajectories are converted by the python script `bag_to_pose.py` provided in `rpg_trajectory_evaluation` ROS bag.

To evaluate the error of ORB-SLAM, we use the toolbox for quantitative trajectory evaluation of VO/VIO mentioned in [5]. And we give the esimated trajectory and the ground truth. Set the align type to `sim3`. The toolbox generates the final plots and tables which are attached in the gitlab. And Figure 7 shows the absolute position and orientation error with respect to the traveled distance, computed from the aligned trajectory and the groundtruth. The relative position and rotation errors for sub-trajectories of different lengths are plotted in Fig. 8.

From Figure 6-8, we can find that the error in Data1(about 50cm) is larger than the error in Data2(about 20 cm).

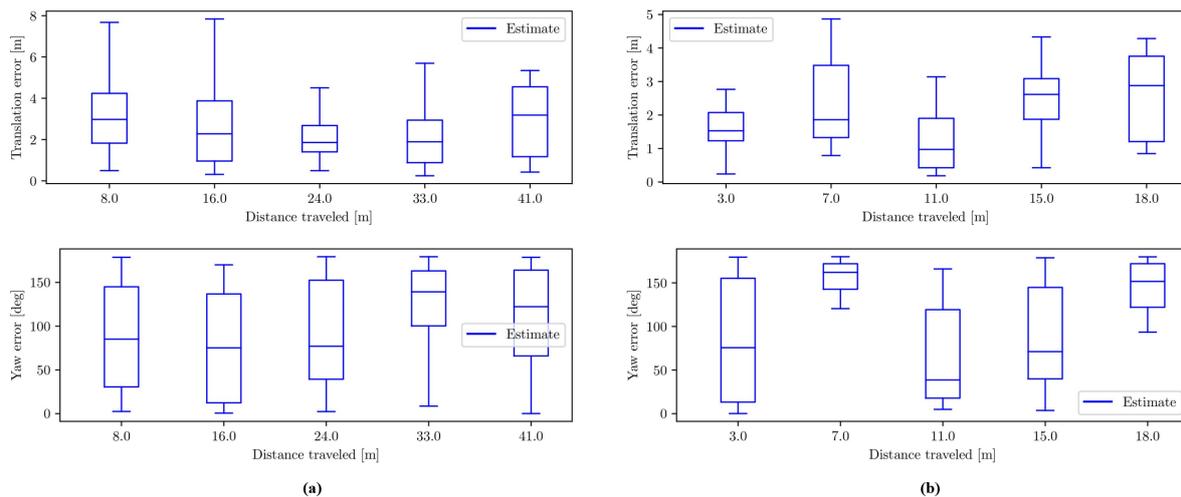


Figure 8: Relative position and rotation error.(a) Data1 (b)Data2

5 Conclusion

Firstly, we write a driver for ladybug5+ with ROS which aims to acquire images at frame rate of 30Hz. Also we write a new message type for publishing raw data acquired from ladybug5+. A launch file for splitting raw data to 6 images and a launch file for stitching panoramic images are attached in our project. Secondly, we record 2 bags of data to do ORB-SLAM2 and the evaluation for the SLAM has done in our report. The panoramic images will be used to generate the colored point cloud by combining with lidar data in the future.

References

- [1] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.
- [2] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.
- [3] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.
- [4] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. A benchmark for the evaluation of rgb-d slam systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 573–580. IEEE, 2012.
- [5] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.
- [6] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[7] saoto28 plieningerweb lennarthaller, hoangthien94. A ros implementation of orb-slam2. https://github.com/appliedAI-Initiative/orb_slam_2_ros. Accessed October 16, 2019.

Appendix

Install

1. Download Autoware (ver12.0) and install it by source follow the official step.
2. Download ladybugSDK (1.16)

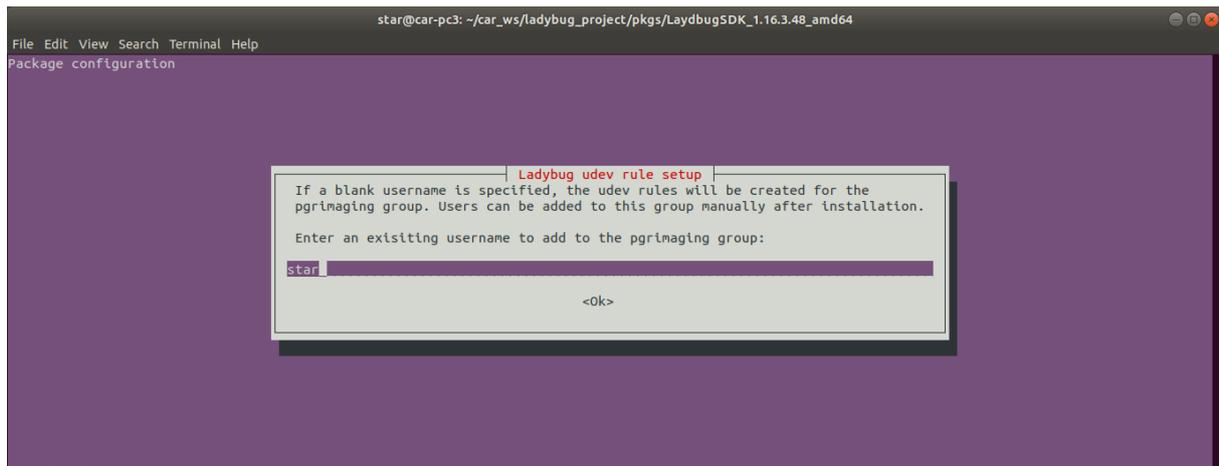
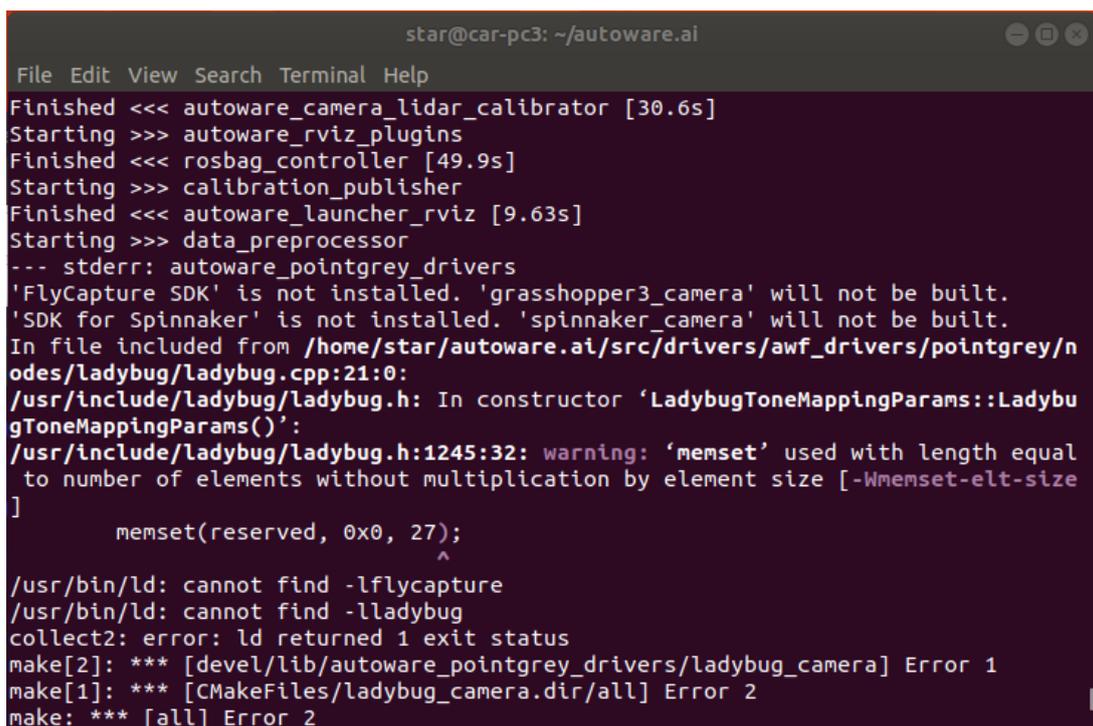
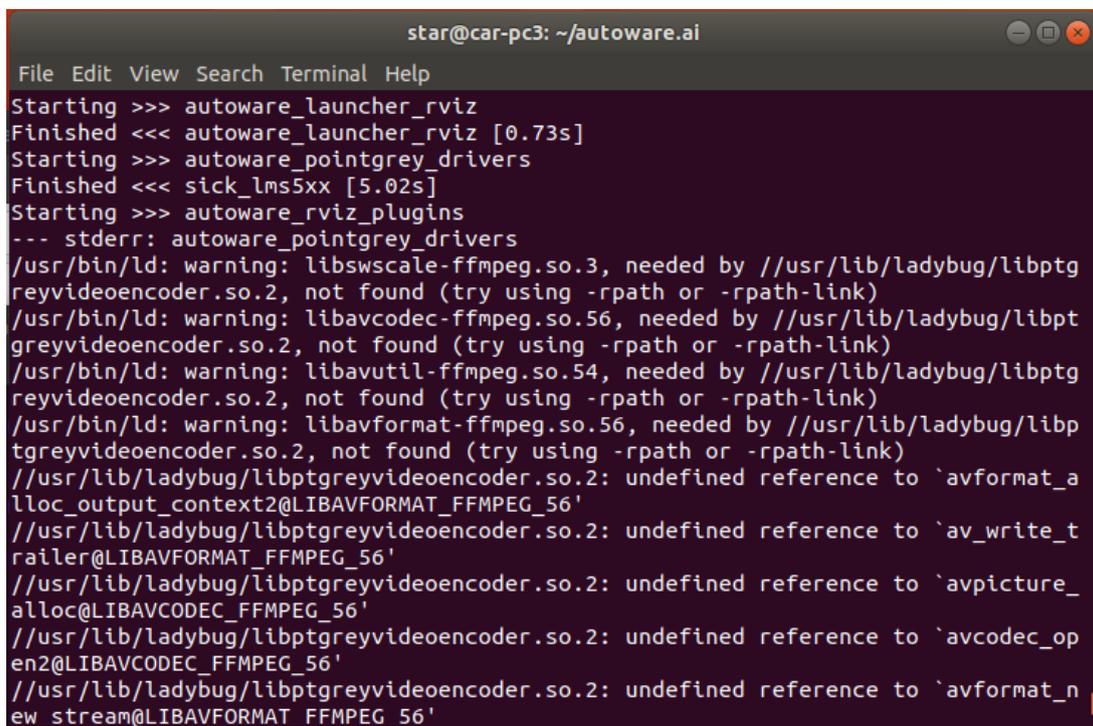


Figure 9: If see this enter you username

3. Install ladybugSDK (pkgs missed can be found in the folder [pkgs] or in <https://opsx.alibaba.com/mirror>)
4. Clone src from ladybug_project repo in star-center's gitlab
5. Catkin_make it





```
star@car-pc3: ~/autoware.ai
File Edit View Search Terminal Help
Starting >>> autoware_launcher_rviz
Finished <<< autoware_launcher_rviz [0.73s]
Starting >>> autoware_pointgrey_drivers
Finished <<< sick_lms5xx [5.02s]
Starting >>> autoware_rviz_plugins
--- stderr: autoware_pointgrey_drivers
/usr/bin/ld: warning: libswscale-ffmpeg.so.3, needed by //usr/lib/ladybug/libptgreyvideoencoder.so.2, not found (try using -rpath or -rpath-link)
/usr/bin/ld: warning: libavcodec-ffmpeg.so.56, needed by //usr/lib/ladybug/libptgreyvideoencoder.so.2, not found (try using -rpath or -rpath-link)
/usr/bin/ld: warning: libavutil-ffmpeg.so.54, needed by //usr/lib/ladybug/libptgreyvideoencoder.so.2, not found (try using -rpath or -rpath-link)
/usr/bin/ld: warning: libavformat-ffmpeg.so.56, needed by //usr/lib/ladybug/libptgreyvideoencoder.so.2, not found (try using -rpath or -rpath-link)
//usr/lib/ladybug/libptgreyvideoencoder.so.2: undefined reference to `avformat_alloc_output_context2@LIBAVFORMAT_FFMPEG_56'
//usr/lib/ladybug/libptgreyvideoencoder.so.2: undefined reference to `av_write_trailer@LIBAVFORMAT_FFMPEG_56'
//usr/lib/ladybug/libptgreyvideoencoder.so.2: undefined reference to `avpicture_alloc@LIBAVCODEC_FFMPEG_56'
//usr/lib/ladybug/libptgreyvideoencoder.so.2: undefined reference to `avcodec_open2@LIBAVCODEC_FFMPEG_56'
//usr/lib/ladybug/libptgreyvideoencoder.so.2: undefined reference to `avformat_new_stream@LIBAVFORMAT_FFMPEG_56'
```

Figure 10: Missing pkgs