Pucca Wanders Around: 3D SLAM with Two Tilted Lidars

A project of the 2017 Robotics Course of the School of Information Science and Technology (SIST) of ShanghaiTech University

https://robotics.shanghaitech.edu.cn/teaching/robotics2017

Zhengjia Huang, Zhonglin Nian, Ning Bi
{huangzhj, nianzhl, bining}@shanghaitech.edu.cn

***Abstract*— 3D Simultaneous Localization and Mapping (SLAM) has drawn significant interests in robotics community, as it enables the robotic vehicles to be deployed in a fully autonomous way for various applications. Most works on 3D SLAM use a single Lidar mounted upright on top of a mobile robot, plus an IMU for initial guess of the robots motion. In our project, we studied 3D SLAM using two Lidars simultaneously and the Lidars are tilted to specific angles instead of upright. Moreover, we use only the point clouds without IMU data to estimate the robots motion. We show that by tilting the Lidar, we can see the upper part in the scene (such as the ceiling) which is normally hard using common strategy. And using our method, we can build accurate omnidirectional 3D map on a real time bases.**

Keywords - Simultaneous Localization and Mapping, Tilted Lidar, Iterative Closest Point, Pipeline rebuild.

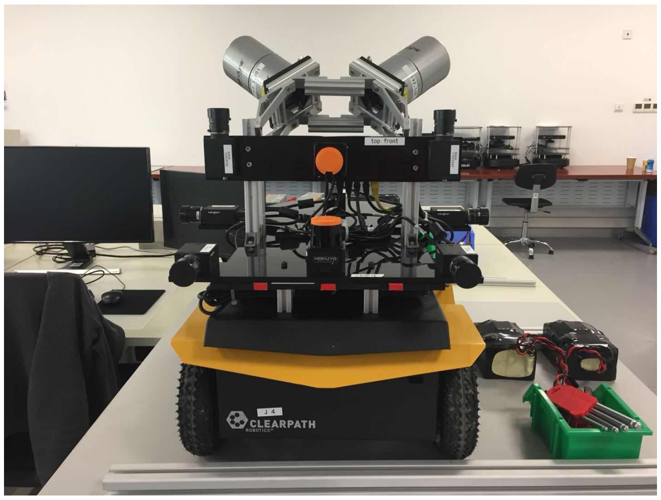


Fig. 1. Our Robot ”Pucca”

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) has drawn significant interests in robotics community, as it enables the robotic vehicles to be deployed in a fully autonomous way for various applications. Recently, more and more researchers have looked towards performing SLAM on mobile robots in fully 3D, indoor and outdoor environments. 3D SLAM can be used in some pioneering systems such as rebuilding the map of an area after earthquakes and also contribute to the navigation systems on autonomous driving cars. And Lidar, due to its high accuracy in 3D range data perceiving, could be a capable sensor in SLAM work flow.

Most of the SLAM algorithms, such as the one by F. Aghili[4], use IMU as part of their localization unit. In this project, we tried to do 3D SLAM using two tilted lidars without IMU. Figure. 1 shows our robot ”Pucca”. Our work mainly built upon the Erik’s BLAM[1] package. We tested our system in a relative small scene (ShanghaiTech Star Center) and a relatively large scene (ShanghaiTech SIST lobby). The result shows that our system can build accurate omnidirectional 3D map on a real time bases using two tilted lidars.

Our contributions are four-fold. First, we calibrated the two lidars mounted on our Pucca robot. Second, we figured out the pipeline of BLAM algorithm and wrote a document for it. Third, we enabled the BLAM algorithm to do SLAM using two tilted lidars so that it can build a 3D omnidirectional map. Fourth, we programed a node to visualize the ICP process between two closed loop laser scans to ease the debugging.

The detail of our work flow is described as follows: Section 2 introduces related works; Section 3 is system description; Section 4 provides evaluation details and the last section gives conclusion of our method.

II. Related Works

A. BLAM[1]

Our work is built upon the Berkeley Localization And Mapping package. It is an open-source software package for LiDAR-based real-time 3D localization and mapping developed by Erik Nelson from the Berkeley AI Research Laboratory (BAIR). Unfortunately, there is no document available about BLAM. So a big part of our work is figuring out the whole pipeline of BLAM and we want this report to also be a document for BLAM. Besides, BLAM was designed for SLAM using one Lidar mounted upright on a robot. We enabled the algorithm to support multiple lidars and the lidars can be tilted.

B. Generalized ICP[2]

$$\hat{x}_t = \underset{x_t}{argmax}\{p(z_t|x_t, \hat{m}^{[t]})\}$$

Maximize the likelihood of the i-th pose and observation relative to the given map(PointClouds). z_t denoted as **current measurement** and $\hat{m}_{[t]}$ denoted as **map constructed so far**. General ICP algorithm consists of four parts:

1) Nearest point search:
   For each point current scan find the nearest correspond point in base frame. And run rejection to guarantee one-to-one correspondences.
2) Compute registration:
   Determine the rigid transform that minimizes the sum of squared distances(SSD) between pairs by RANSAC scheme.
3) Apply Transformation:
   Apply the rigid transform to all points in current scan.
4) Iterate:
   Repeat steps 1 to 3 until convergence.

BLAM uses ICP in localization and loop closure. ICP is a widely used algorithm to register two PointCloud and it is vital for SLAM, since this algorithm could be used to estimate the Odometry of robots and behave as a tool to check loop closure, which is the core of SLAM. The common ICP algorithm simply uses closest points to get the correspondences and then calculate transform matrix. Instead, a more robust variant, which is called Generalized ICP, performs better, and its used in this project. To be specific, this algorithm combine ICP and **point-to-plane** ICP algorithms into a single probabilistic framework. And then use this framework to model locally planar surface structure from both scan, that is to say, plane to plane ICP.

C. 3D SLAM Using IMU and Its Observability Analysis[6]

Currently, our package do not take imu as input, however as for a big loop closure, it's import to take IMU data into account, or the accumulated deviation only using the transform calculated by ICP will destroy the map. This paper investigates 3-dimensional SLAM and the corresponding observability analysis by fusing data from landmark sensors and a strap-down IMU in an adaptive Kalman filter (KF). In addition to the vehicle's states and landmark positions, the self-tuning filter estimates the IMU calibration parameters as well as the covariance of the measurement noise. Examining the observability of the 3D SLAM system leads to the the conclusion that the system remains observable provided that the line connecting the two known landmarks is not collinear with the vector of total acceleration, i.e., the sum of gravitational and inertial accelerations.

D. LiDAR System Calibration Using Overlapping Strips[7]

This paper talks about how to calibrate and produce estimates a set of parameters based on system measurements. These parameters can be used to improve the quality of any subsequently-collected LiDAR data. Current LiDAR calibration techniques require full access to the system parameters and raw measurements (e.g., platform position and orientation, laser ranges, and scan-mirror angles). Unfortunately, the raw measurements are not usually available to end-users. The absence of such information is limiting the widespread adoption of LiDAR calibration activities by the end users. This paper proposes alternative methods for LiDAR system calibration, without the need for the system raw measurements. The simplified method that is proposed in this paper uses the available coordinates of the LiDAR points in overlapping parallel strips to estimate biases in the system parameters and measurements. In this approach, the conventional LiDAR equation is simplified based on a few reasonable assumptions; the simplified LiDAR equation is then used to model the mathematical relationship between conjugate surface elements in overlapping parallel strips in the presence of the systematic biases. In addition, a quasirigorous calibration method is also proposed to deal with non-parallel overlapping strips.

E. VoxelNet[8]

VoxelNet is a generic 3D detection network that unifies feature extraction and bounding box prediction into a single stage. Specifically, VoxelNet divides a point cloud into equally spaced 3D voxels and transforms a group of points within each voxel into a unified feature representation through the newly introduced voxel feature encoding (VFE) layer. In this way, the point cloud is encoded as a descriptive volumetric representation, which is then connected to a Region Proposal Network (RPN) to generate detections. A key innovation of VoxelNet is the VFE layer. It uses fully connect layers and its output feature combines both point-wise features (absolute and relative coordinates) and locally aggregated feature (high level). In train mode, the VoxelNet takes 3D point cloud as input and ground-truth bounding box as label and can be trained end-to-end. In test mode, the VoxelNet takes in 3D Point cloud and outputs predicted bounding boxes of objects. A limitation of VoxelNet is that when dealing with the ununiform distribution of point cloud, it uses random sampling to down sample the points, which will loss some information. Another weakness is VoxelNet performs convolution on 3D voxels, which is super time consuming. Another model called PointNet++ [9] solved the aforementioned problems and the model can be extended to 3D semantic segmentation.

F. LOAM: Lidar Odometry and Mapping in Real-time[10]

A real-time method for odometry and mapping using range measurements from a 2-axis lidar moving in 6-DOF. The method achieves both low-drift and low-computational complexity without the need for high accuracy ranging or inertial measurements.
The key idea in obtaining this level of performance is the division of the complex problem of simultaneous localization and mapping, which seeks to optimize a large number of variables simultaneously, by two algorithms. One algorithm performs odometry at a high frequency but low fidelity to estimate velocity of the lidar. Another algorithm runs at a frequency of an order of magnitude lower for fine matching and registration of the point cloud. Combination of the two algorithms allows the method to map in real-time. By testing on KITTI dataset, the method ranked #1 among all methods evaluated by the benchmark irrespective of sensing modality, including state of the art stereo visual odometry. The average position error is 0.88% of the distance traveled, generated using trajectory segments at

100m, 200m, ..., 800m lengths in 3D coordinates.

G. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks[12]

Object detection networks depend on region proposal algorithms to hypothesize object locations. Runs about 22 fps on NVIDIA Titan-X.

H. ROS Package: libpointmatcher

Libpointmatcher is an open-source implementation of the iteratively closest point method for 2-D/3-D mapping in Robotics.
It supports both point-to-point and point-to-plane ICP. With the former, it is able to solve not only for a rigid transform, but also for a scale change between the clouds. Libpointmatcher also implements a set of filters to help denoise and subsample the input point clouds.
Comparing to poorly documented icp-ros package, libpointmatcher's source code is fully documented based on doxygen to provide an easy API to developers.
The general work flow and main components are: Data Filters (reduce number of points), Matcher (transform, then perform kd-tree), Outlier Filter (remove links which do not correspond to true correspondences), Error Minimizer, Transformation checkers (a threshold on the rotational and translational), Inspectors.

I. Geometric Calibration and Radiometric Correction of LiDAR Data and Their Impact on the Quality of Derived Products[13]

This paper talks about a practical approach for the geometric calibration of LiDAR systems and radiometric correction of collected intensity data while investigating their impact on the quality of the derived products

J. Continuous Trajectory Estimation for 3D SLAM from Actuated Lidar[14]

This paper talks about how to extend the Iterative Closest Point (ICP) algorithm to obtain a method for continuous-time trajectory estimation (CICP) suitable for SLAM from actuated lidar.

K. A flexible and scalable SLAM system with full 3D motion estimation[15]

This paper talks about a flexible and scalable system for solving the SLAM problem that has successfully been used on unmanned ground vehicle with low computational resources.

One method of modeling the map from point cloud data to grid map is first to downsample and removal outliers in order to reduce the pressure of CPU or GPU, but in this paper, they only use endpoints of axis Z within a threshold of the intended scan plan so there is less data to deal with which is faster than donwsample. Also, they developed a method to approach occupancy value at a certain point and the gradient of occupancy gradient on a grid maps which accelerate the speed by linear interpolation.

In order to align and match the data from laser scans with the existing map, they use Gauss-Newton approach without data association searching between beam endpoints or an exhaustive pose search. It can not be denied that the algorithm would lead to a local minima but it works well in practice.

As mentioned before, any method using gradient approach would face the risk of being stuck in a local minima, so the writer mitigate the problem by using multi-resolution map similar to image pyramid. But they are not generated from a single high resolution image, instead, different maps are kept in memory and simultaneously updating. This generative approach ensures that maps are consistent across scales while at the same time avoiding costly downsample operations

As for navigation for 3D state estimation, they use EKF, IMU, and a range sensor to update the pos and velocity of the robot and to promote the accuracy of the motion data.

All in all, this paper could be regarded as a guideline and inspiration for our 3D slam project even though it's a 2D slam based on 3D motion data. And there should be a way to learn and integrate this knowledge.

L. ROS Package: Cartographer

Cartographer is an open-source package with ROS operation system on Kinetic version. It's developed by Google and could be applied for both 2D and 3D SLAM. Multi-sensors are available in this package like LIDAR, IMU and different kinds of cameras which could be helpful in practice. We don't really need to use the whole package, but by knowing the nodes and topics of the package, we would get further information about how 3D slam can be done. This package uses the fundamental tf package to deal with rigid body transform. And there are diverse API that could read multiple data from sensors, dealing with real-time loop closure, pose adjustment, real time scan matching/alignment, building occupancy grid map based on point cloud and path planning.

M. Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning[16]

Since our work is mainly based on Lidar scanner, the calibration work can be both fundamental and essential. This paper discussed the basic calibration process and mathematical model for measurements, which can be the efficient guide essay to our first step, calibration.

N. Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age[17]

Keywords: Robots, SLAM, Factor graphs, Maximum a posteriori estimation, sensing, perception.
This paper surveys the current state of SLAM and consider future directions. They start by presenting what is now the de-facto standard formulation for SLAM. Then review related work, covering a broad set of topics including robustness and scalability in long-term mapping, metric and semantic representations for mapping, theoretical

performance guarantees, active SLAM and exploration, and other new frontiers.

O. Simultaneous Localization and Mapping with Infinite Planes[18]

Simultaneous localization and mapping with infinite planes is attractive because of the reduced complexity with respect to both sparse point-based and dense volumetric methods. This paper shows how to include infinite planes into a leastsquares formulation for mapping, using a homogeneous plane parametrization with a corresponding minimal representation for the optimization.

P. ROS Package: lidar_camera_calibration

The package is used to calibrate a Velodyne LiDAR with a camera (works for both monocular and stereo).
Specifically, Point Gray Blackfly and ZED camera have been successfully calibrated against Velodyne VLP-16 using lidar-camera-calibration.
Since, VLP-16 provides only 16 rings, we believe that the higher models of the Velodyne will also work well with this package.

III. System Description

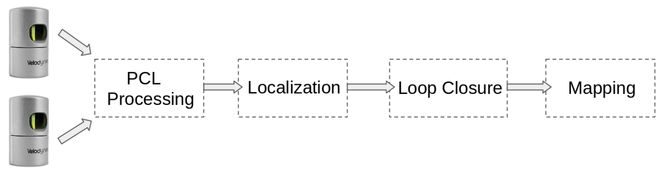


Fig. 2. The SLAM system

As shown in Figure. 2, the system mainly contains 5 parts. The two Velodyne Lidars scan the environment. The PCL Processing part takes in the cloud points, transform and filter them. The processed point cloud is then input into the Localization unit, which estimate the robots current pose using the current scan, previous scan, and the current map. Then, if the current frame is a key frame, the Loop Closure unit check for loop closure between current frame and previous key frames and recalculate the poses if needed. Finally, the mapping unit insert current scan into the map base on the estimated current pose. Below we describe each unit in detail.

A. PCL Processing

The PCL processing procedure is shown in Figure 3. When doing SLAM using both lidars, two Transform nodes will transform the pcls from their lidars bases to the base link of the robot. Then, a Merge node concatenates the transformed pcls using approximate synchronizer. Our lidars are rotating at 10Hz, so the merged two pcls will have a time difference no larger than 0.05s. If we are doing SLAM using only one lidar, only the lidar and its corresponding Transform node

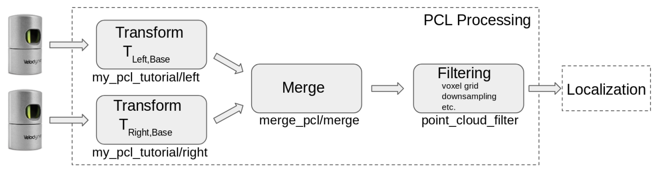


Fig. 3. PCL Processing

will be on and the merge node will not run. The final step is Filtering. Voxel grid with adjustable resolution and random downsampling (90%-98% drop rate) are performed to reduce the size of point cloud.

B. Localization

To locate the robot, the algorithm first take an initial guess based on current laser scan and last laser scan. It performs Generalized ICP between the two pcls to calculate a transform from last frame to current frame. Using the transform, we can transform the current pcl into global frame. Then, the nearest neighbors of current scan in current map are found. After that, the algorithm performs ICP between the scan and its neighboring points to get another transform, which is a correction transform of the initial guess. The final step is to compose the correction transform with the initially guessed pose to get the estimated current pose of the robot.

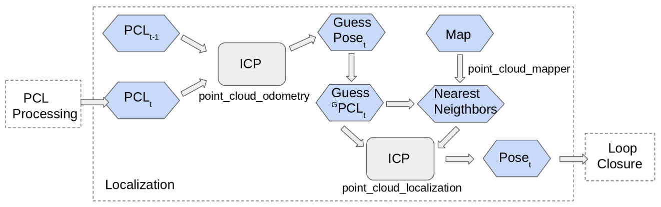


Fig. 4. Localization

C. Loop Closure

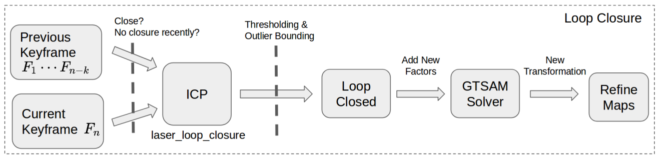


Fig. 5. Loop Closure

BLAM uses GTSAM for factor graph SLAM. An example is shown in Figure 6. The factors are transforms calculated from ICP using laser scans of two poses. For some of the poses (Keyframes), their corresponding laser scans are stored for loop closure and mapping. To register a new keyframe, the robot has to be at a pose that more than 0.5 meters away from the previous keyframe pose. When a loop closure is detected, the poses are recalculated and the map is rebuilt using the new poses.

To check for loop closure, the current frame has to be a keyframe. Loop closure is checked between current keyframe and all the previous keyframes. For a closed loop, the following conditions have to be satisfied: (1) last closed loop happened more than 40 frames ago (2) the two keyframes are at least 40 frames away from each other (3) the two poses have a direct distance smaller than 1.5 meters (4) the ICP fitness must be below a threshold. For every keyframe, all the possible closures are checked if any of them satisfied the closure conditions, new factors will be added to the graph. Poses and map will be recalculated and rebuilt base on the new graph after loop closures.

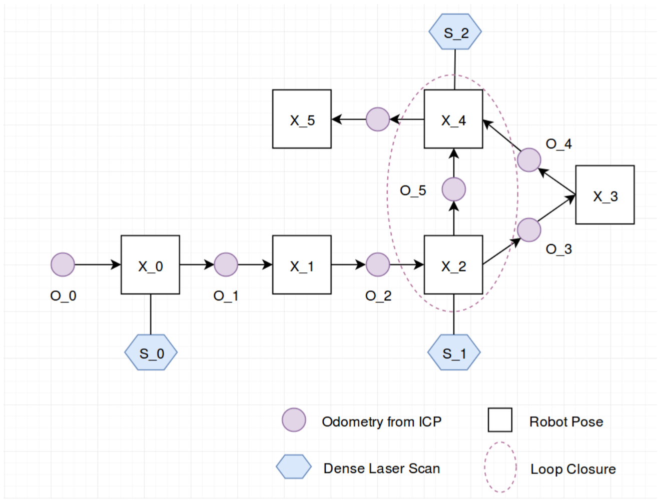


Fig. 6. Pose Graph

D. Mapping

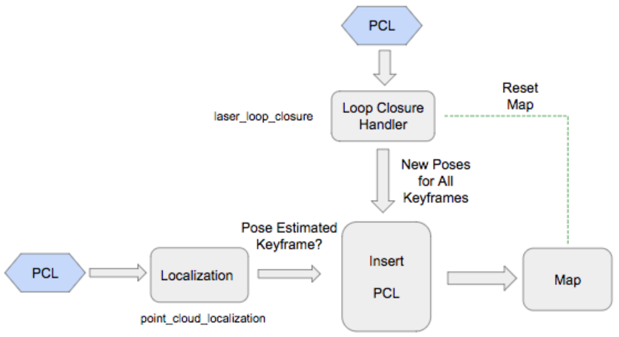


Fig. 7. Mapping

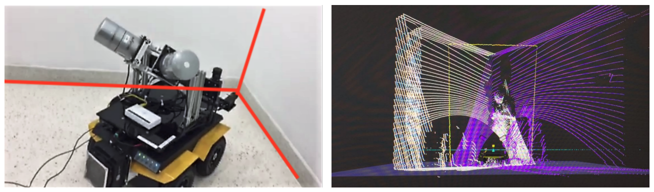
The Mapping section is a combination of results from Localization and Loop Closure. Fig.7 shows the process of mapping in BLAM. For every incoming pcls, the Localization node will perform ICP to estimate current pose and if it's a keyframe(refer to Section: Loop Closure), mapper will transform it to Global frame and add it to the map. In the meanwhile, the Loop Closure node is detecting whether it's forming a closed loop. IF so, the GTSAM solver will produce a series of new transformations for each key frame. And then, reset the map and add each newly transformed point clouds into the map. The map is stored as Oct-tree.

IV. EVALUATION

We evaluate our system in three aspects: calibration, localization, and mapping. We further programed a tool to visualize the ICP process which could ease the debugging process.

A. Calibration

Two Lidars are mounted in two different orientations, so we need to do calibration for these two Lidars to get rigid transformations between $base\_link$ and the frame of two Lidars. **Instead of** an accurate calibration using Lidar-Camera calibration with 3D-3D point correspondences[3], we first perform a rough calibration because the approach proposed in that paper is a simple scenario that Lidar and Camera receive almost same information from the environment like stereo vision. But in our case, the camera and Lidar are mounted in totally different angles which leads to merely no correspondences. As another alternative, we put Jackal robot at the corner of walls so we could measure the alignment of two PointClouds better in RVIZ with the help of 3 perpendicular vanishing lines and two nicely mounted Hokuyo. Figure 8 shows our calibration setting. We tried to maximize and calibrate the overlapping of the corner scanned by the two lidars.

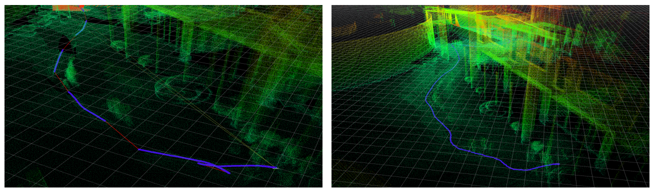


(a) Calibration Setting (b) Calibration in RVIZ

Fig. 8. Calibration

B. Localization

One problem we faced during localization is the drift problem, as shown in Figure.9.a. The long red line means the robot moved a large distance between two frames, which is impossible. After some experiment, we figured out that this is a performance issue. The speed of the SLAM algorithm can't follow up if the point clouds are very dense. So, if we filter out more points, as shown in Figure.9.b, the problem is solved.



(a) Filter out 95% points (b) Filter out 98% points

Fig. 9. Localization

C. Mapping

Figure 10.a shows the SLAM result using two lidars, while in Figure 10.b we only used one lidar to SLAM in the same scene. Comparing the results, it is obvious that the map using two lidars contains more information, such as the structure on the up-right part in the image and some of the inner structure of the room on the right part of the image. However, due to the error caused by calibration and time difference between two lidar scans, the SLAM result using two lidars is more noisy.

Figure 10.c shows a long straight corridor in SIST lobby. This it tricky for loop closure, because it has no legible features for ICP to distinguish different parts of the corridor. Moreover, due to the length of the corridor, a slight error in pose estimation will result in large discrepancy in the map, as shown in Figure 10.d. Another challenge comes from the setting of our lidar system. Because the lidars are tilted, the view of the surrounding is limited. A lidar can only see about 45+45=90 degrees of view instead of 360 degree. So, if the robot has different rotation at the same position, the things it scanned can be totally different. This makes it hard for ICP to converge.

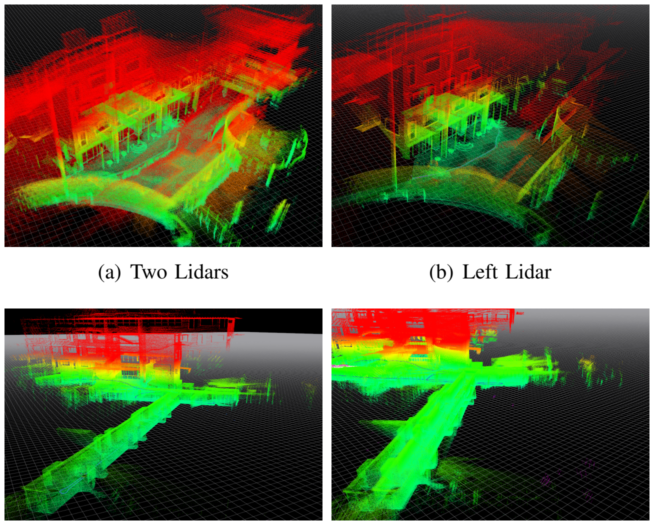

(a) Two Lidars (b) Left Lidar
(c) Before Incorrect Closure (d) After Incorrect Closure

Fig. 10. SLAM Results in SIST Lobby

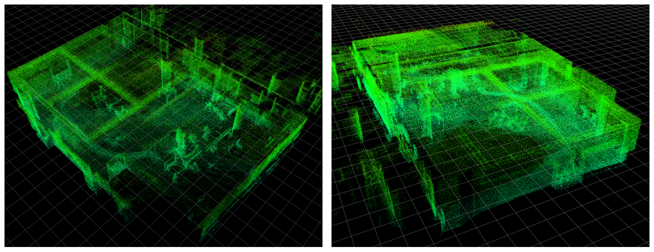

(a) Star Center 1 (b) Star Center 2

Fig. 11. SLAM Results in Star Center Lab

D. Visualizing Tool for ICP

After we notice the incorrect calculation of loop closure mentioned in C Mapping (Fig.10.d), we came up an idea to visualize the process of ICP in Loop Closure detection. This program runs in two parallel threads, the main thread is subscribing paired loop closure scan published by node laser_loop_closure. And meanwhile, sub thread is listening from the keyboard to decide which Loop Closure to visualize and waiting for commands to visualize each iteration of ICP. The approach to visualize ICP is to publish the result of each iteration so that the two pcls could be presented in RVIZ.

Fig.11 show the process of ICP in loop closure detection. These four pictures present the results of before ICP, on iteration 4, on iteration 8, end of ICP. The purple points are target pcl so it's stationary, and the white points are source pcl which are transformed in each iteration. Pcls are presented in World Frame, thus they seem to be well aligned at first, but the we can tell that white points are moving "up" gradually.

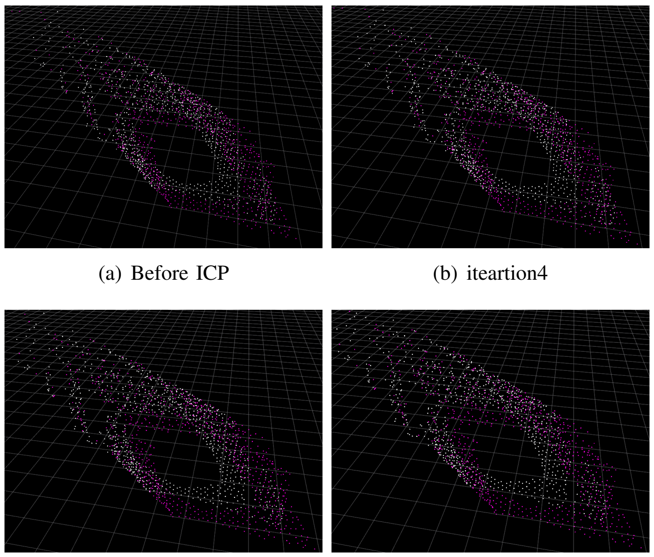

(a) Before ICP (b) iteartion4
(c) iteration8 (d) iteration10

Fig. 12. Visualization Tool(98% filtering)

V. CONCLUSIONS

Our system can build accurate omnidirectional 3D map on a real time bases using two tilted lidars. Making the lidar tilted enables us to see the ceiling of the environment. However, it also limits the view of the surrounding of the robot and make it challenging for ICP to converge to a good result during loop closure.

Beside the contribution for now, there are some promotions we can work on. For example, we could figure out how to calibrate two Velodyne with the help of two Hokuyo and four depth camera nicely. Also, it's meaningful to make use of the IMU data so that we can get better performance of localization since handling large scale laser data can be challenging for single ICP. To handle the incorrect loop closure estimation, adding landmark can be an a good candidate solution. Since pictures contain much more information than laser scan, we could use SIFT or perceptual hashing to compare similarity of scenes from different frames to get a better loop closure performance, or simply, attach QR codes manually in the environment as landmark.

VI. APPENDIX

A. *Code Usage*

To figure out the function of each node in the algorithm, please refer to the figures in this report. We have wrote the corresponding node next to each function in the pipline.

To test the code, please filter out the ’/tf’ topic from your recorded bag using:

```
rosbag filter data.bag data-no-tf.bag ”topic!=’/tf’”
```

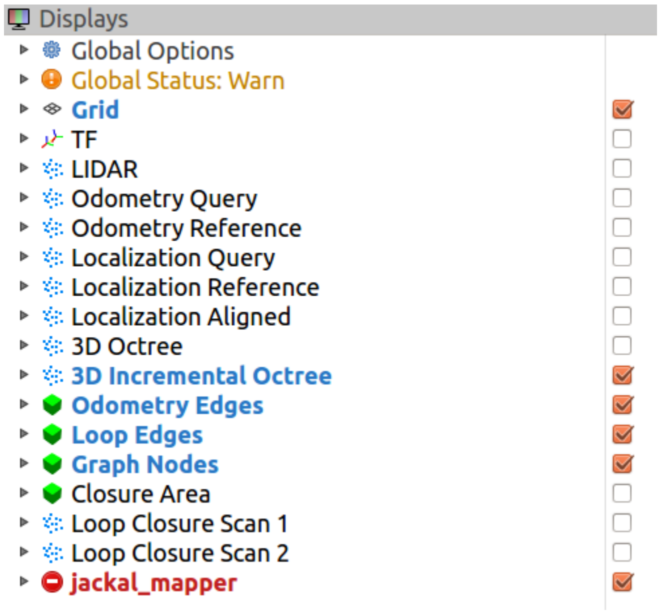
Then play the bag with clock configuration:

```
rosbag play data-no-tf.bag –clock
```

Then you can launch the algorithm. In our setting, we have two velodune lidars. By default, only the left lidar will be used. If you want to use the right lidar or both lidars at the same time, set the ’lidar’ parameter to ’right’ or ’both’, for example:

```
roslaunch blam_example test_online.launch lidar:=right
```

B. *RVIZ Setting*



Displays	
Global Options	
Global Status: Warn	
Grid	✓
TF	☐
LIDAR	☐
Odometry Query	☐
Odometry Reference	☐
Localization Query	☐
Localization Reference	☐
Localization Aligned	☐
3D Octree	☐
3D Incremental Octree	✓
Odometry Edges	✓
Loop Edges	✓
Graph Nodes	✓
Closure Area	☐
Loop Closure Scan 1	☐
Loop Closure Scan 2	☐
jackal_mapper	✓

Fig. 13. RVIZ Setting

When you launch the code, the rviz setting will be automatically loaded. The ’LIDAR’ section shows the current laser scan. The ’Odometry Query’ to ’Localization ALigned’ sections are for localization visualization. The ’3D Octree’ and ’3D Incremental Octree’ are the current map and update for the map. ’Odometry Edges’ to ’Closure Area’ are visualization to the pose graph and loop closure. The ’Loop Closure Scan’ are for visualizing the ICP process of any loop closure. Finally, ’jackal_mapper’ is the 3D model of our robot.

C. *Flow Chart of Mapping*

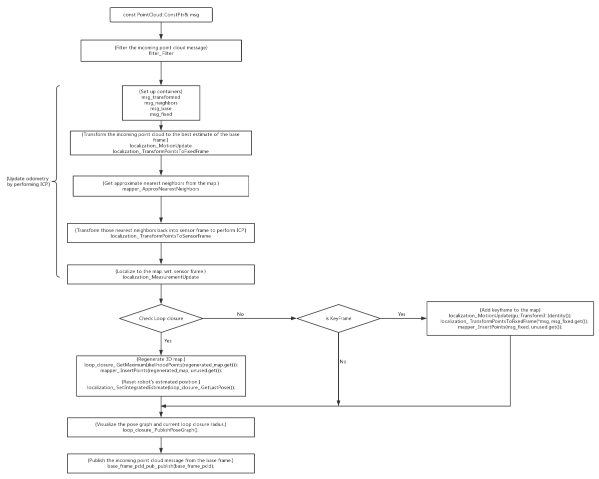

const PointCloud::ConstPtr& msg
{Filter the incoming point cloud message} filter_.Filter
{Set up containers} msg_transformed msg_neighbors msg_base msg_fixed
{Transform the incoming point cloud to the best estimate of the base frame.} localization_.MotionUpdate localization_.TransformPointsToFixedFrame
{Update odometry by performing ICP.}
{Get approximate nearest neighbors from the map.} mapper_.ApproxNearestNeighbors
{Transform those nearest neighbors back into sensor frame to perform ICP.} localization_.TransformPointsToSensorFrame
{Localize to the map. wrt. sensor frame.} localization_.MeasurementUpdate
Check Loop closure
No
is KeyFrame
Yes
{Add keyframe to the map} localization_.MotionUpdate(gu::Transform3::Identity()); localization_.TransformPointsToFixedFrame(*msg, msg_fixed.get()); mapper_.InsertPoints(msg_fixed, unused.get());
Yes
No
{Regenerate 3D map.} loop_closure_.GetMaximumLikelihoodPoints(regenerated_map.get()); mapper_.InsertPoints(regenerated_map, unused.get());
{Reset robot's estimated position.} localization_.SetIntegratedEstimate(loop_closure_.GetLastPose());
{Visualize the pose graph and current loop closure radius.} loop_closure_.PublishPoseGraph();
{Publish the incoming point cloud message from the base frame.} base_frame_pcld_pub_.publish(base_frame_pcld);

Fig. 14. Flow Chart of Mapping

VII. ACKNOWLEDGEMENT

We would like to thank Patric and Professor Sören’s instruction during the project.

REFERENCES

[1] Erik Nelson ,B(erkeley) L(ocalization) A(nd) M(apping).

[2] Aleksandr V. Segal and Dirk Haehnel and Sebastian Thrun, Generalized-ICP

[3] Ankit Dhall and Kunal Chelani and Vishnu Radhakrishnan and K. M. Krishna, LiDAR-Camera Calibration using 3D-3D Point correspondences, 2017

[4] F. Aghili. 3d slam using imu and its observability analysis. pages 377383, Aug 2010.

[5] K. I. Bang, A. Habib, and M. Mller. Lidar system calibration using overlapping strips. 15, 03 2010.

[6] F. Aghili. 3d slam using imu and its observability analysis. pages 377383, Aug 2010.

[7] K. I. Bang, A. Habib, and M. Mller. Lidar system calibration using overlapping strips. 15, 03 2010.

[8] H. S. L. J. G. C. R. Qi, L. Yi. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 06 2017.

[9] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for pointcloud based 3d object detection. 11 2017.

[10] J. Zhang and S. Singh. LOAM: Lidar Odometry and Mapping in Real-time. RSS, 2014

[11] R. Dub, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, C. Cadena. SegMatch: Segment based loop-closure for 3D point clouds. ICRA, 2016

[12] S. Ren, K. He, R. Girshick, J. Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. NIPS, 2015

[13] Ayman F. Habib, Ana P. Kersting, Ahmed Shaker, and Wai-Yeung Yan:Geometric Calibration and Radiometric Correction of LiDAR Data and Their Impact on the Quality of Derived Products

[14] H Alismail, LD Baker, B Browning:Continuous Trajectory Estimation for 3D SLAM from Actuated Lidar

[15] Stefan Kohlbrecher and Oskar von Stryk, Johannes Meyer and Uwe Klingauf: A flexible and scalable SLAM system with full 3D motion estimation

[16] C Glennie DD Lichti:Static Calibration and Analysis of the Velodyne HDL-64E S2 for High Accuracy Mobile Scanning

[17] C Cadena L Carlone H Carrillo Y Latif D Scaramuzza:Past, Present, and Future of Simultaneous Localization And Mapping: Towards the Robust-Perception Age

[18] M Kaess:Simultaneous Localization and Mapping with Infinite Planes