The Iby and Aladar Fleischman
Faculty of Engineering
Tel Aviv University

kudan

Visual-LiDAR-odometry fusion SLAM

Project Number: 19-3-2-1990

Name: Kengo Akano **ID:** 933655698
Adviser: Tatsuya Aizawa
Project performed at Kudan Inc.

Abstract

Simultaneous localization and mapping (SLAM) is a technique applied in artificial intelligence mobile robot for a self-exploration in numerous geographical environment. SLAM becomes a fundamental research area in recent days as it promising solutions in solving most problems related to the self-exploratory oriented mobile robot field.

As the researchers in the domains of visual SLAM and 2D Lidar SLAM are barely cross-bordered, it is hard to find the existed projects working on the fusion SLAM of them. Thus, we resulted in assuming that the synthesis SLAM supplemented the weak points of visual SLAM with 2D Lidar SLAM is more robust and inventive if we can integrate.

Aiming at creating robust SLAM with the fusion data of Visual and 2D by setting off merits against demerits each other, the fusion SLAM algorithm using data from camera, Lidar (Light Detection and Ranging) implemented on a mobile robot, and robot wheel is designed and observed.

The experiments based on robotic operating system (ROS) kinetic, C++ coding, and TurtleBot2 KOBUKI, using stereo camera and Lidar to acquire visual information and 2D laser scanned matching data for each, KudanSLAM provided from Kudan Inc. and an open-source LittleSLAM package for 2D Lidar SLAM is adopted to achieve indoor mapping in an unknown environment. As the supportive information in the case that the robot lost its visual information, we make use of wheel odometry date from turtleBot2 for path plotting.

The experimental results show that the fusion SLAM designing scheme is feasible and robust, and it can construct a high-precision map in real-time, even though the robot locates in a difficult environment for SLAM based on a single kind of dataset.

Per the simulations, we conclude the assumption is legitimate to suggest the concept is promising and precise for mapping and path-sketching.

Contents

Introduction

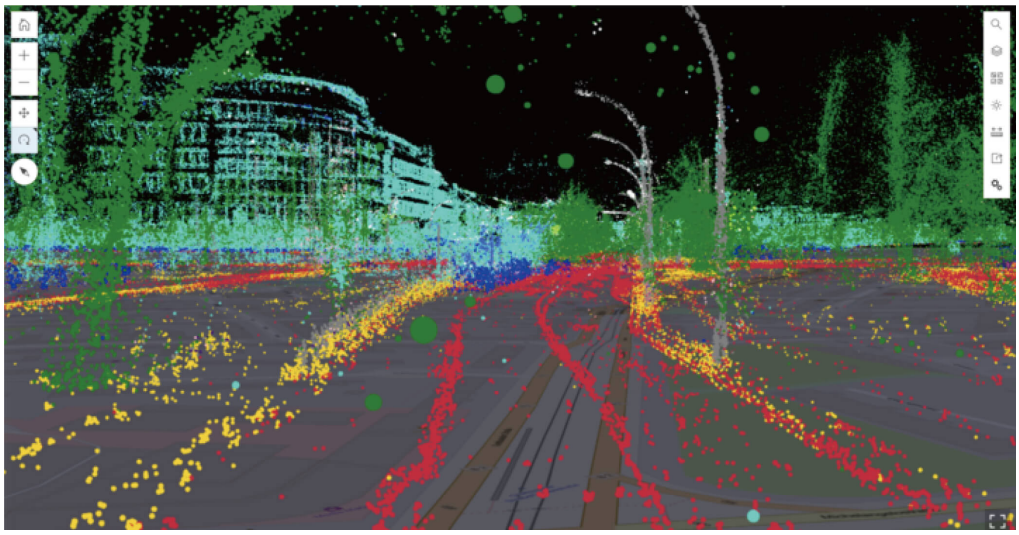
SLAM is a method used for autonomous vehicles that lets you build a map and localize your vehicle in that map at the same time. SLAM has been the subject of technical research for many years and has become a highly important topic within the computer vision community, and is receiving particular interest from a large number of industries. As a result, there are a growing number of technology components used to achieve SLAM. With a variety of SLAM systems being made available, from both academia and industry, it is worth exploring what exactly is meant by SLAM.

The objective of this project is to develop Visual-LiDAR-odometry fusion SLAM and prove it robust enough to achieve practical implementation even in challenging environments such as dark or no texture situations because this concept or the competitive ones have not been produced for practical use in the market.

This introduction will give a brief overview of some of the fundamental concepts of the popular methods of SLAM; visual SLAM and Lidar SLAM and the systems that we specifically use in this project; KudanSLAM, LittleSLAM, and wheel odometry.

Visual SLAM

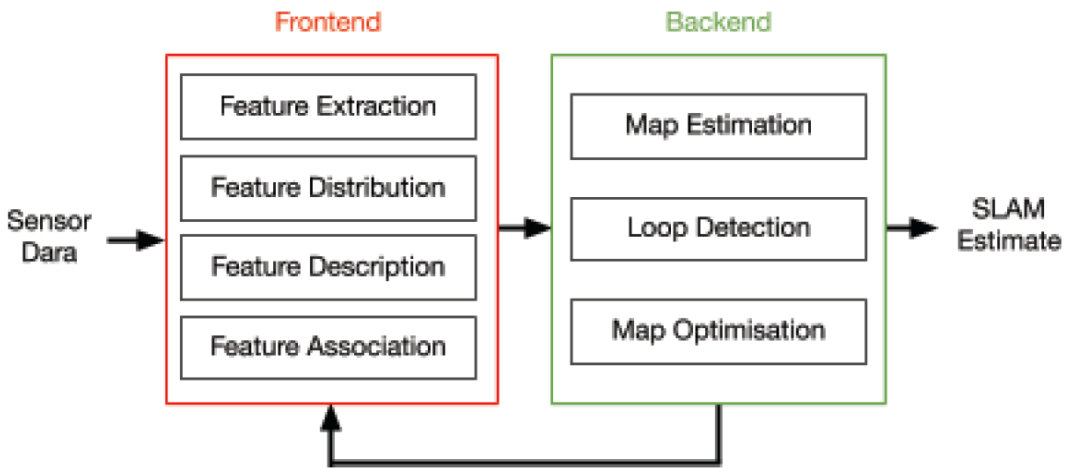
Visual SLAM uses images acquired from cameras and other image sensors such as simple cameras (wide-angle, fish-eye, and spherical cameras), compound eye cameras (stereo and multi-cameras), and RGB-D cameras (depth and ToF cameras). Visual SLAM can be implemented at a low cost with relatively inexpensive cameras. Besides, since cameras provide a large volume of information, they can be used to detect landmarks. At the same time, visual SLAM is not accurate or lose tracking when a robot is inside a small building that has monotonic walls or a dark place.



KudanSLAM is primarily a Visual SLAM system, which means that the main input data is a camera feed. This feed can be optionally augmented by a variety of sensors:

- A second camera to create a Stereo Visual SLAM system.
- A depth sensor in the case of RGBD SLAM.
- An Inertial Measurement Unit (IMU) for Visual Inertial SLAM.

KudanSLAM divides tracking and mapping tasks into two main threads, that work in parallel to provide the real-time position and orientation of the main camera and an optimized map of the environment.



At Kudan, their approach to software development is based on several key principles which ensures that we remain ahead in a rapidly developing and competitive field. These principles are based on providing highly-performant cross-platform SLAM software and a supportive base for their clients to ensure they can receive all the advantages of using our KudanSLAM system. Moreover, their customer-based approach allows them to deliver a boutique, tailor-made SLAM software package, allowing clients to fully exploit the capabilities of their hardware.

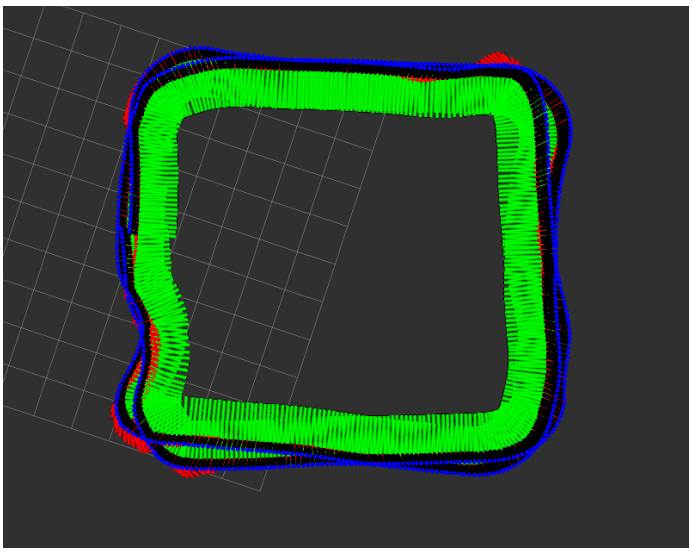
Lidar SLAM

Lidar SLAM generally called 2D or 3D Lidar SLAM is a method that primarily uses a laser sensor or distance sensor. Compared to cameras, ToF, and other sensors, lasers are significantly more precise and are used for applications with high-speed moving vehicles such as self-driving cars and drones. The output values from laser sensors are generally 2D (x, y) or 3D (x, y, z) point cloud data. The laser sensor point cloud provides high-precision distance measurements and works very effectively for map construction with SLAM even if in a dark place. On the other hand, point clouds are not as finely detailed as images in terms of density and do not always provide sufficient features for matching. For example, in places where there are few obstacles or walls, it is difficult to align the point clouds and this may result in losing track of the vehicle location. Besides, point cloud matching generally requires high processing power, thus it is necessary to optimize the processes to improve speed.

LittleSLAM is a program for learning 2D Lidar SLAM. It takes a file containing 2D laser scan data and odometry data as input and outputs the trajectory of the robot position and a 2D point cloud map on gnuplot/rviz. LittleSLAM consists of elemental technologies such as scan-matching, sensor fusion of laser scanner and odometry, and loop closure based on Graph-based SLAM. LittleSLAM is a program designed as educational material for a reference book [1], and it uses a simple algorithm.

Wheel Odometry

Wheel Odometry that we will use as supportive data for the camera and Lidar information is the output from a robot's inertial measurement units (IMUs) and wheel encoder, which can measure physical quantities such as velocity and orientation. This can continuously yield the distance and direction messages as a robot travels unless the robot is kidnapped or its wheels slip due to the floor material.



Wheel odometry from turtleBot2 showed on rviz

In a summary, the three methods above mentioned have advantages and disadvantages such as below.

	Pros	Cons
Visual SLAM	• Can be implemented at low cost with relatively inexpensive cameras • Detect a landmarks since cameras provide a large volume of information • Can be achieved flexibility in SLAM implementation	• Not accurate or lose tracking when a robot is in a inside the small building has monotonic walls or dark place • If a single camera as the only sensor, it is challenging to define depth
2D Lidar SLAM	• Provides high-precision distance measurements which works very effectively for map construction with SLAM • Used for applications with high-speed moving vehicles • Works even in the dark environments	• Lidar is relatively expensive • point clouds are not as sufficiently detailed as images in terms of density for matching • Point cloud matching generally requires high processing power
Wheel Odometry	• Can be acquired from the robot wheel without the additional hardware implementation	• Slip and Idling can be happened due to the floor material and steps • Error of rotation direction

About Kudan

Kudan Inc. was founded in Bristol, the UK in 2011. Kudan has aspired to be an irreplaceable provider of eyes for all future machines and devices, disrupting both industrial landscapes and consumer lives. Our technology provides a critical component for these systems, enabling machines to perceive the world. Kudan has been providing proprietary Artificial Perception technologies based on SLAM to enable use cases with significant market potential and impact on our lives such as autonomous driving, robotics, AR/VR, and smart cities.

Planning

As the first step, design the initial fusion SLAM based on LittleSLAM which outputs the point cloud map from the Lidar scan data and robot's trajectory according to the input from Kudan SLAM.

Secondly, we build the upgraded version of it that can draw the path even in the situation the visual information is lost such as the dark and no texture environment. This means the initial SLAM does not have the input from a camera in that circumstance which ends up with that it loses track of the movement records. To complement the visual information, we integrate wheel odometry to let the robot continue to track its location. When the camera is masked or the robot enters the dark place, the SLAM automatically detects the status transition (i.e. Visual SLAM has been lost) and switch from Visual SLAM to wheel odometry.

For simulations, we conducted test-runs for both of the prototyped SLAMs to verify that they smoothly work precisely and our assumption that all the inputs image, scan, and odometry can complement each other is right.

Implementation

We here divide the implementation section into two parts: hardware and algorithm.

Hardware

As a prototype robot, we mounted Hokuyo LiDAR (UTM-30LX-EW), LeadSense stereo camera, and mobile buttery on a ROS standard platform robot TurtleBot2 KOBUKI and also set up KOBUKI to be able for us to remotely control it with DUALSHOCK 4. With this robot, we conducted the test runs to check if the plotted pathway was accurate enough comparing with the actual route throughout our simulations.

The mobile robot we integrated for prototyping

Overviews of each component

Hokuyo LiDAR (UTM-30LX-EW): The UTM-30LX-EW is a compact, lightweight 2D LiDAR sensor used for obstacle detection and localization on autonomous mobile robots (AMR) and automated guided vehicles and carts (AGV, AGC). Equipped with an Ethernet interface, it can obtain measurement data in a 270° field-of-view up to 30 meters. With IP67 protection and multi-echo integration, this sensor offers reliable object detection and environmental mapping in all weather conditions for outdoor robotic applications.

LeadSense stereo camera: LeadSense, a binocular passive 3D sensory module, can process graphics collected by the cameras on each side to generate real-time high-resolution outputs of binocular images and corresponding depth maps. The product can be used in indoor and outdoor environments, suitable for robot/drone navigation, image analysis, 3D measurement, and automatic warehouse. The product can output stereo image, depth image, and IMU data (only by N-series).

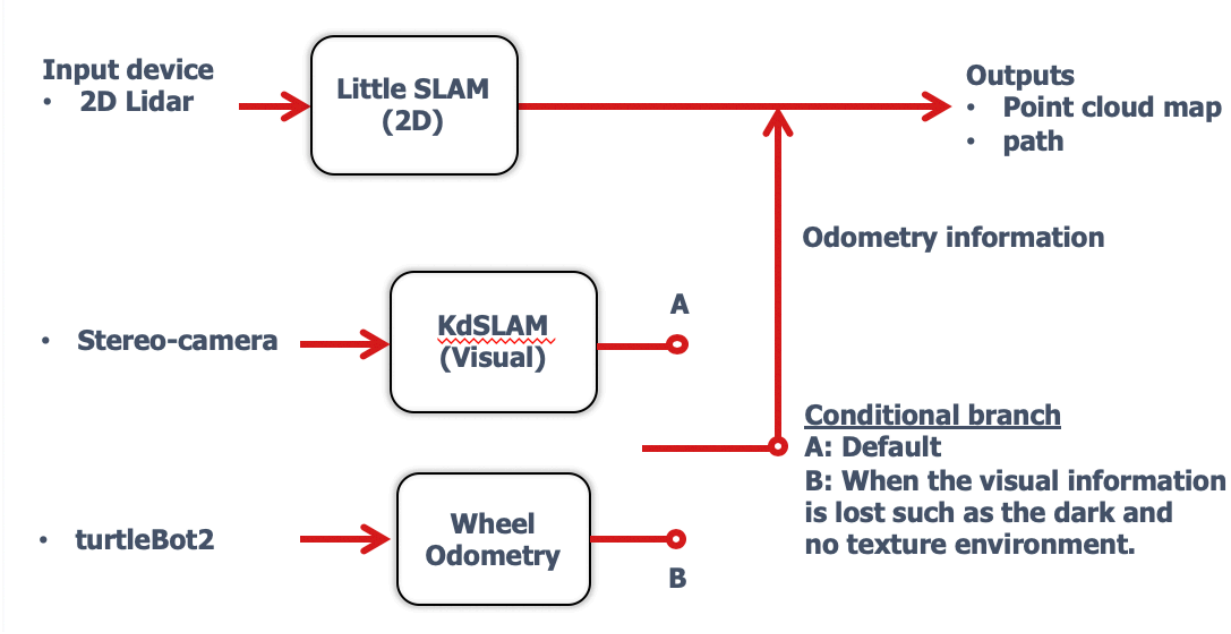
TurtleBot2 KOBUKI: TurtleBot is a low-cost, personal robot kit with open-source software. With TurtleBot, you'll be able to build a robot that can drive around your house, see in 3D, and have enough horsepower to create exciting applications. TurtleBot 2 consists of an YUJIN Kobuki base, a 2200 mAh battery pack, a Kinect sensor, an Asus 1215N laptop with a dual-core processor, fast charger, and a hardware mounting kit attaching everything and adding future sensors.

DUALSHOCK 4: a controller capable of providing vibration feedback, was based on the onscreen actions taking place in the game, as well as analog input through two analog sticks.

Algorithm

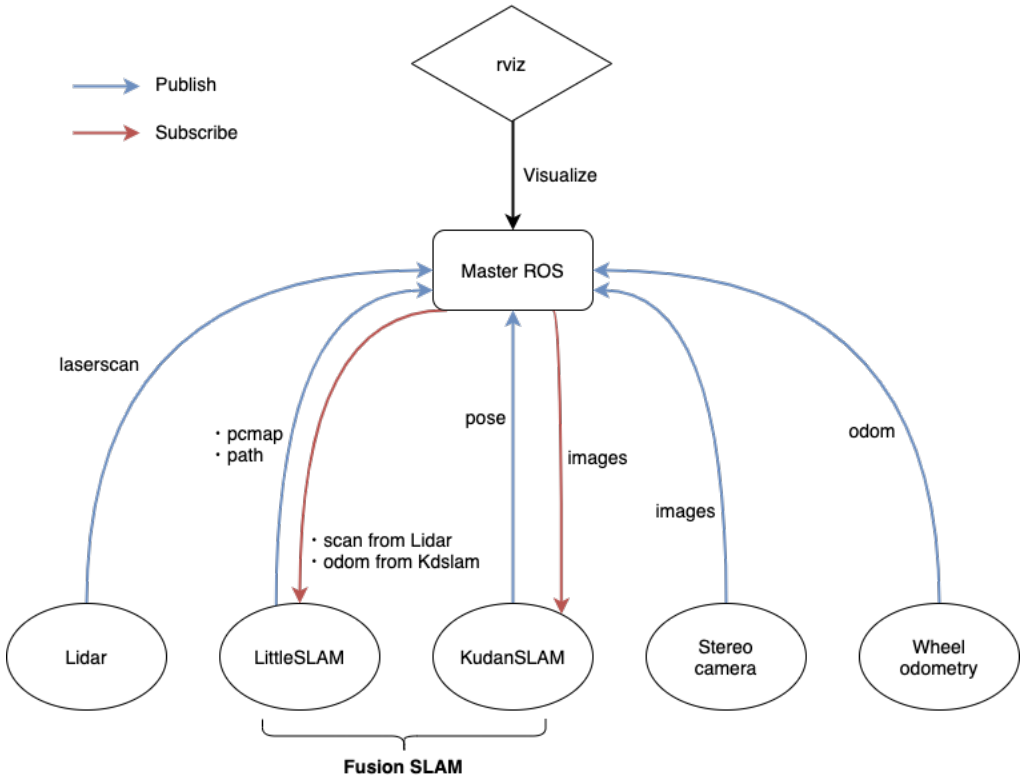
First of all, we started to implement on ROS the first fusion SLAM **(A)** based on open source Little SLAM which makes a point cloud map by scan-matching algorithm with the scan data of Lidar from each frame and initial pose from the output of Kudan SLAM.

After we had confirmed the first test result was solid to show (A) is working as we expected, we coded to subscribe wheel odometry from TurtleBot and publish it to LittleSLAM when the camera lost the visual information **(B)**. This is the final version of Fusion SLAM that we are pursuing.



Logic flow of the algorithm

Nodes structure on ROS



Structure of nodes

We designed and coded nodes also recognized as subscribers/publishers and topic communications as above. rviz is a 3D visualization tool for ROS that shows our test results.

Key Concepts of the algorithm

- Coordinate transformation

In principle, coordinate systems in ROS are always in 3D, and are right-handed, with X forward, Y left, and Z up.
While the output of Kudan SLAM and TurtleBot wheel odometry publish the coordinate information as quaternion: position (x, y, z) and orientation (x, y, z, w), 2D LittleSLAM only requires (x, y, θ) to draw the robot trajectory. To transform the original orientation into θ, we need to convert quaternions into Euler angles using ROS default package `tf`.
tf is a package that lets the user keep track of multiple coordinate frames over time. tf maintains the relationship between coordinate frames in a tree structure buffered in time and lets the user transform points, vectors, etc between any two coordinate frames at any desired point in time.

As you can see in Function 1, we succeed in the transformation of quaternions to Euler angles with `tf::quaternionMsgToTF` function for converting Quaternion msg to Quaternion and `getRPY` helper function for getting raw, pitch, and yaw from a Quaternion message.
After the conversion to Euler angles, we substitute as following *(x, y) = position(x, y)* and θ *= yaw angle + 90° (∵* `RAD2DEG` *conversion radian into degree results in - 90° shift).*

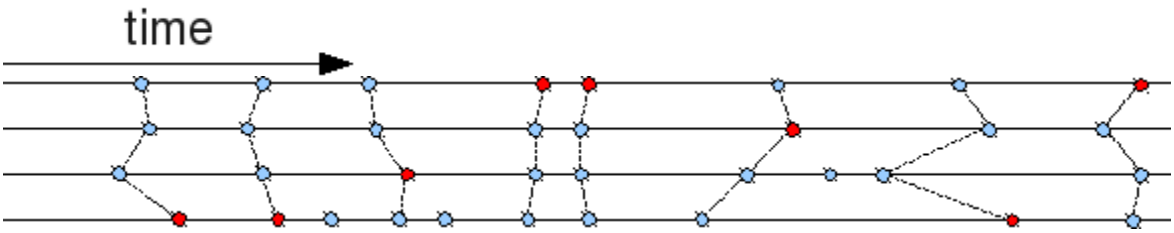
```
109
110 // Little SLAM scan2D in the normal states
111 static bool make_scan2d_kdslam(Scan2D &out_scan, const sensor_msgs::LaserScanConstPtr& scan, const geometry_msgs::PoseStamped::ConstPtr& geo)
112 {
113     // Quartanion to Euler
114     ros::Subscriber odom_sub_; //quantanion
115
116     // kdslam odometry
117     // the incoming geometry_msgs::Quaternion is transformed to a tf::Quaterion
118     tf::Quaternion quat;
119     tf::quaternionMsgToTF(geo->pose.orientation, quat);
120
121     // the tf::Quaternion has a method to acess roll pitch and yaw
122     double roll_k, pitch_k, yaw_k;
123     tf::Matrix3x3(quat).getRPY(roll_k, pitch_k, yaw_k);
124
125     //double kx, ky;
126     // the found angles are written in a geometry_msgs::Vector3 (represents a vector in free space)
127     //geometry_msgs::Vector3 rpy_k; // vector message (member: x, y, and z)
128     kx_cur = geo->pose.position.x;
129     ky_cur = geo->pose.position.y;
130     kth_cur = RAD2DEG(yaw_k)+90;
131
132     //path
133     out_scan.pose.tx = kx_cur;
134     out_scan.pose.ty = ky_cur;
135     //out_scan.pose.th = RAD2DEG(tf::getYaw(tr.getRotation()));
136     out_scan.pose.th = kth_cur;
137     out_scan.pose.calRmat();
138
```

Function 1: Substitutes KudanSLAM quarternions into LittleSLAM coordinates

• Time Synchronizer

The `TimeSynchronizer` filter synchronizes incoming channels by the timestamps contained in their headers and outputs them in the form of a single callback that takes the same number of channels.

Since the message publish/subscribe rates of scan (Lidar), /kdslam_ros_stereo/pose (stereo camera), odom (robot wheel), and processFrameResult (process frame of KudanSLAM) are different, we need to match messages coming on a set of topics to process corresponding coordinate transformations. For example, Time goes from left to right, each line is a topic, and each dot is a message. The red message is the pivot, and the broken line links the messages in a set.



Thus, we here apply ApproximateTime Policy from message_filters package. The `message_filters::sync_policies::ApproximateTime` policy uses an adaptive algorithm to match messages based on their timestamp. Contrary to `message_filters::sync::ExactTime`, it can find the best messages match even if they have different time stamps.

As in Function 2, `MysyncPolicy1` is a synchronizer in the case of logic (A); on the other hand, `MysyncPolicy2` is for logic (B).

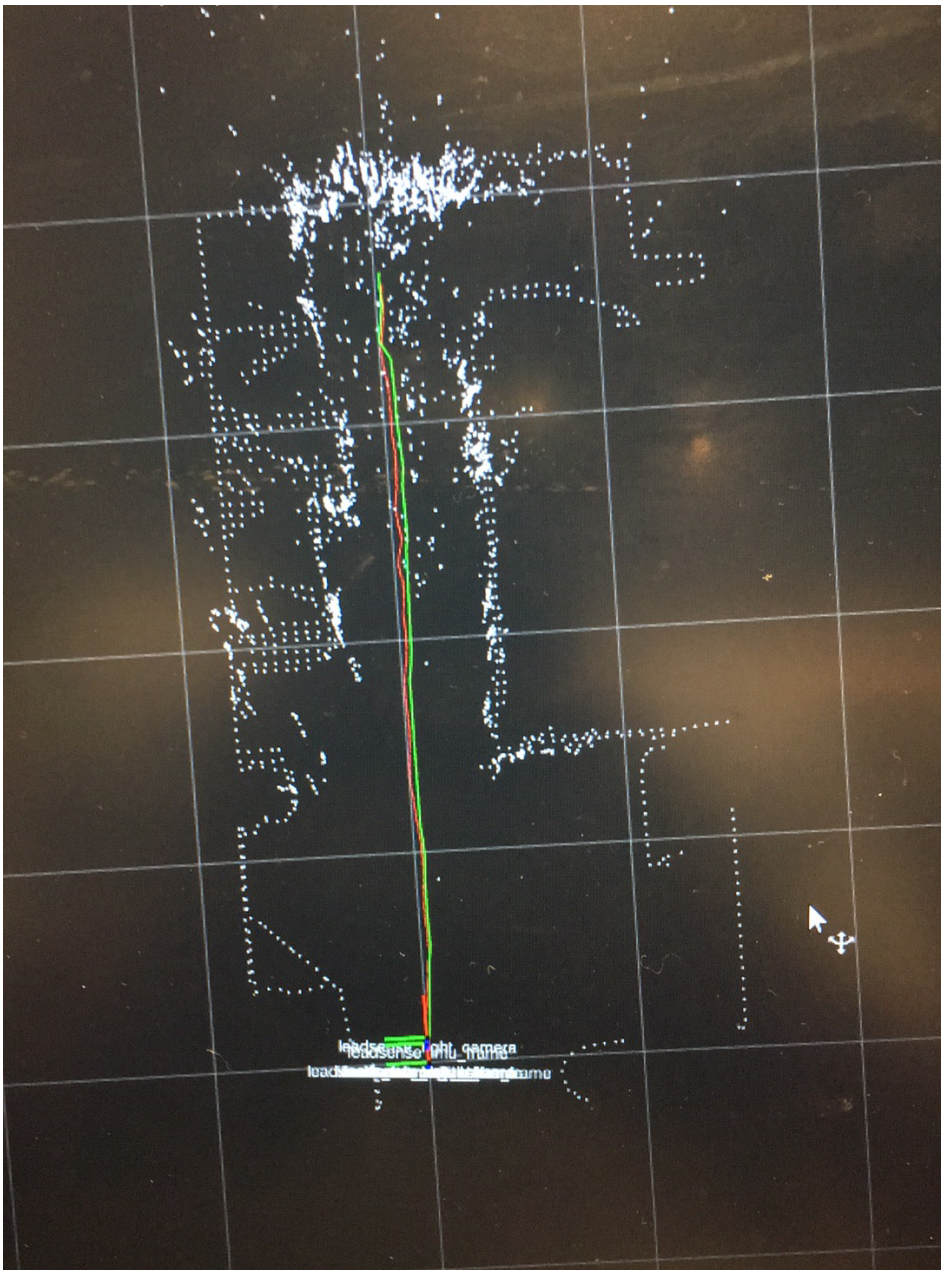
```
471
472     message_filters::Subscriber<sensor_msgs::LaserScan> laser_sub_(n, "scan", 1);
473     message_filters::Subscriber<geometry_msgs::PoseStamped> odom_sub_(n, "/kdslam_ros_stereo/pose", 1); //quantanion
474     //message_filters::Subscriber<geometry_msgs::Pose2DStamped> pose_sub_(n, "pose", 1);
475     //The wheel odometry topic from turtlebot2 Kobuki odom
476     message_filters::Subscriber<nav_msgs::Odometry> wheel_sub_(n, "odom", 1);
477     message_filters::Subscriber<sensor_msgs::Temperature> pfr_sub_(n, "processFrameResult", 1);
478     //The node sends a message in the type of sensor_msgs/PointCloud2 & nav_msgs/Path to
479     //the topics called "pcmap_pub_" & "path_pub_". The max buffer size is 10.
480     ros::Publisher pcmap_pub_ = n.advertise<sensor_msgs::PointCloud2>("pcmap", 10);
481     ros::Publisher path_pub_ = n.advertise<nav_msgs::Path>("path", 10);
482
483
484     typedef message_filters::sync_policies::ApproximateTime<sensor_msgs::LaserScan, geometry_msgs::PoseStamped, nav_msgs::Odometry> MySyncPolicy1;
485     message_filters::Synchronizer<MySyncPolicy1> sync1(MySyncPolicy1(3), laser_sub_, odom_sub_, wheel_sub_);
486     sync1.registerCallback(boost::bind(&callback1, _1, _2, _3));
487
488     typedef message_filters::sync_policies::ApproximateTime<sensor_msgs::LaserScan, nav_msgs::Odometry, sensor_msgs::Temperature> MySyncPolicy2;
489     message_filters::Synchronizer<MySyncPolicy2> sync2(MySyncPolicy2(3), laser_sub_, wheel_sub_, pfr_sub_);
490     sync2.registerCallback(boost::bind(&callback2, _1, _2, _3));
491
```

Function 2: Time Synchronizer section in the main function

Simulations and Results

Our simulations were taken place at the office of Kudan Inc. We drive TurtleBot2 KOBUKI mounted devices to verify the functions of the developed algorithm.

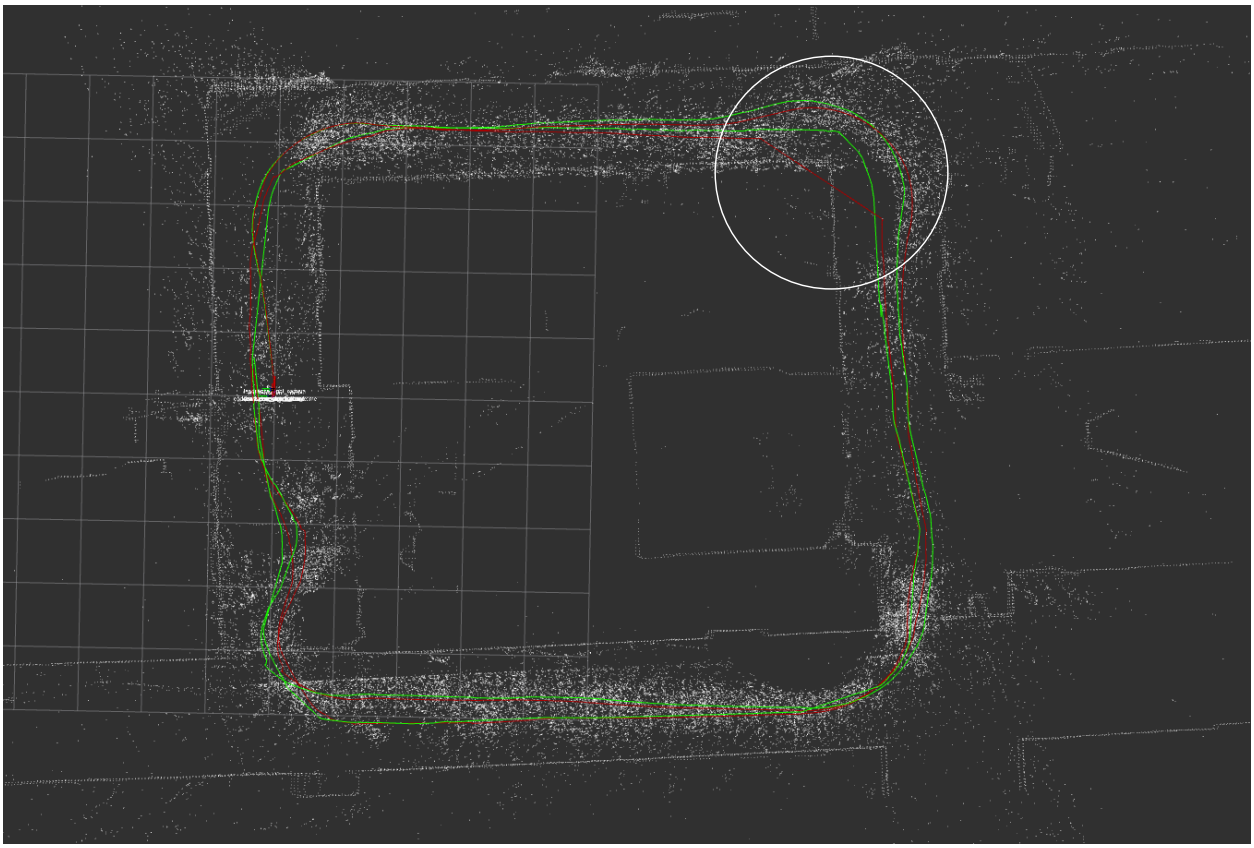
We conducted the first test run to check if the logic (A) is properly working and the plotted pathway was accurate enough comparing with the actual route and the path from Kudan SLAM. In this simulation, KOBUKI run a simple straight pathway. Result 1 shows the point cloud map correctly demonstrates the environment and both results of the trajectory from Kudan SLAM and LittleSLAM essentially match. Hence, we proceeded to develop the algorithm (B) and examine its simulations as the next step.
(Since I lost the simulation data, I here attached a picture of the result screen.)



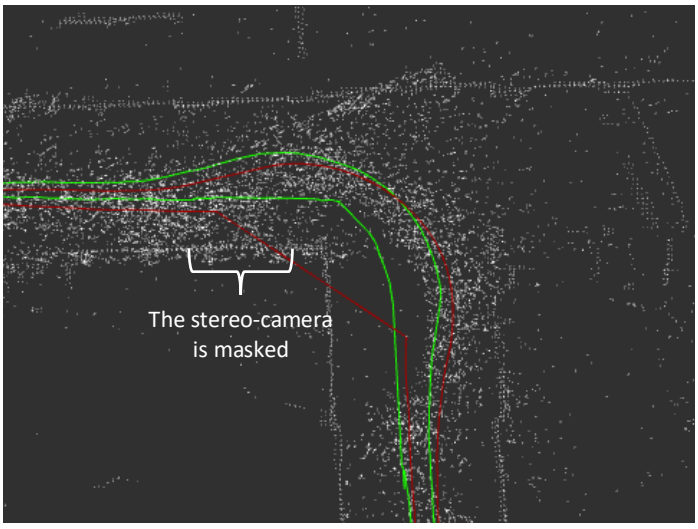
Result 1 (Red: Kudan SLAM, Green: Fusion SLAM)

In the second simulation to test the logic (B), KOBUKI drove two loops; it normally run until loop closure and mapping completion in the first loop, and then we sometimes masked the camera in the middle of the second loop to see if wheel odometry was properly called and draw path instead of Kudan SLAM.

The result of the second test run is that mapping is practically faultless, the processing is slightly slow, and the path is almost precise even though the camera was partially masked. In conclusion, the path and map results show that the developed SLAM is robust and precise enough to assist the assumption is correct.



Result 2-1 (Red: Kudan SLAM, Green: Fusion SLAM)



Result 2-2: Closeup picture of circle domain of Result 2-1

When we take a look at the closeup picture, the plotted path from Kudan SLAM which is a red line is lost during the camera is masked and after some frames. On the other hand, Fusion SLAM smoothly keeps drawing the robot trajectory by successfully switching the input source of odometry from Kudan SLAM (A) to wheel odometry (B). Compared with the actual path where KOBUKI passed, we could verify the result is robust enough to conclude Fusion SLAM is practically accurate even if in the dark and no texture environment.

Summary, Conclusions, and proposals for future work

Simultaneous Localisation and Mapping (SLAM) has a long history and a diversity of techniques has been adopted to solve it at different times.
We assume that the new method of SLAM with the fusion data of Visual and 2D would complement their disadvantages with each other and be robust enough for practical use. Currently, the biggest disadvantage of Visual SLAM is that it is not accurate or lose tracking when a robot is inside a small building that has monotonic walls or a dark place. We set the goal to cover this weakness with our method: Visual-LiDAR-odometry fusion SLAM.
This project is supported by Kudan, Inc. providing their proprietary visual SLAM Kudan SLAM and conducted under the technical supervisor Mr. Tatsuya Aizawa from Kudan, Inc.

For the development based on robotic operating system (ROS) kinetic, we designed the initial fusion SLAM based on open-source 2D Lidar SLAM LittleSLAM which outputs the point cloud map from the Lidar scan data and robot's trajectory according to the input from Kudan SLAM. Then, we built the upgraded version of it that can draw the path even in the situation the visual information is lost such as the dark and no texture environment. This means the initial SLAM does not have the input from a camera in that circumstance which ends up with that it loses track of the robot's movement records. To complement the visual information, we integrate wheel odometry to let the robot continue to track its location. When the camera is masked or the robot enters the dark place, the SLAM automatically detects the status transition (i.e. Visual SLAM has been lost) and switch from Visual SLAM to wheel odometry.

For simulations, a mobile robot TurtleBot2 KOBUKI mounting stereo camera and Lidar on to acquire visual information and 2D laser scanned matching data was applied. We conducted test-runs for both of the prototyped SLAMs to verify that they smoothly work precisely and our assumption that all the inputs image, scan, and odometry can complement each other is confirmed.

The test results show Fusion SLAM smoothly keeps drawing the robot trajectory by switching the input source of odometry from Kudan SLAM to wheel odometry and is virtually accurate and robust even if in the dark and no texture environment. According to the results, we can say Visual-LiDAR-odometry fusion SLAM has the potential to become a popular method of SLAM in the research and commercial field.

For future work, I would suggest working on improving the processing efficiency and sophisticating the SLAM function such as navigation (path planning, obstacle avoidance, etc.) to build an autonomous robot or drone.

List of References

[1] M. Tomono. ”Introductory guide to simultaneous localization and mapping (SLAM 入門)”, Ohmsha, Ltd., 2019.
[2] D. Fox., S. Thrun., and W. Burgard. “Probabilistic Robotics”, MIT Press, Aug 19, 2005
[3] Y. Pyo., R. Kurazume., and R. Jung. “ROS robot programming bible”, Ohmsha, Ltd., 2018.
[4] Kudan, Inc., “KudanSLAM Technical Information”
[5] https://github.com/furo-org/LittleSLAM
[6] https://github.com/kiyoshiiriemon/little_slam
[7] ROS.org, “message_filters::sync::ApproximateTime”, http://wiki.ros.org/message_filters/ApproximateTime
[8] MathWorks – ”Whar Is SLAM? 3 things you need to know", https://www.mathworks.com/discovery/slam.html
[9] Hokuyo, “UTM-30LX-EW :: Hokuyo - Hokuyo USA”, https://hokuyo-usa.com/products/lidar-obstacle-detection/utm-30lx-ew
[10] LeadSense, “LeadSense stereo camera”, http://leadsense.ilooktech.com/?lang=en
[11] TurtleBot, “TurtleBot2”, https://www.turtlebot.com/turtlebot2/