Course Work Final Report with Literature Review – Sensor Fusion for autonomous navigation car-like robot

Filipenko Maksim - MS-AIR

Contents

1.Introduction

1.Significance

Developments in the field of Autonomous Ground Vehicle (AGV) have made significant progress in the last decade. With the development of technologies, it became possible to create small AGVs, but there remain a number of problems. Since any stand-alone installation has a number of physical limitations associated with the power reserve, processing power, the number of sensors placed at the same time, the cost and, depending on the various applications, the inability to use certain sensors. The problem of an effective and fault-tolerant method for determining the position of AGV relative to other environmental objects for specific applications remains open.

2.Relevance

Currently, the literature lacks sufficient reliable experimental data for some applications on the effectiveness of various methods for determining the location of AGV relative to other environmental objects.

3.Literature review

The problem of building a map in an unknown environment using on-board sensors while using these data to solve the localization problem is known as Simultaneous Localization and Mapping (SLAM)[1]. This task can be solved with the use of different sensors. The choice of suitable sensors plays a special role for the effective operation of AGV due to the limited autonomy resource. Most mobile robots contain an Inertial Measurement Unit (IMU), which includes an accelerometer, a gyroscope, and a magnetometer. From these data it is possible to measure orientation, angular velocity and acceleration. This approach to assess the position of the mobile robot is known as the Inertial Navigation System (INS)[2]. This approach cannot be used for navigation due to a large error.

On the other hand, the camera is a good sensor in view of the relatively small cost and simplicity of the configuration [3]. Solving the problem of SLAM using a video camera as the only source of information about the surrounding world refers us to the task of Visual SLAM (V-SLAM [4]). All methods of V-SLAM can be divided into two groups: Feature-based, which use the selected features to build maps and Direct, which work with the entire image as a whole. The first work on visual navigation was carried out using a binocular stereo camera [5][6]. At the same time, a similar problem was considered using a monocular camera, the method was called MonoSLAM [7]. Over the past decade, numerous methods have been proposed, including Parallel Tracking and Mapping (PTAM, [8][9]), REgularized MOnocular Depth Estimation (REMODE,[10]), Oriented FAST [11] and Rotated BRIEF [12] (ORB-SLAM, [13], [14]), Dense Tracking and Mapping in Real-Time (DTAM,[15]), Large-Scale Direct monocular SLAM (LSD-SLAM,[16]), Fast semi-direct monocular visual odometry (SVO,[17]), Direct Sparse Odometry (DSO,[18]), ElasticFusion: Dense SLAM without a Pose Graph (ElasticFusion, [19]), Con-volutional Neural Networks SLAM, (CNN-SLAM,[20]). Critical analysis for some methods was presented by a group of researchers TUM Computer Vision. Each of these methods has its advantages and disadvantages. Common features include weak resistance to unfavorable conditions, experimentation, poor performance with a weak geometric diversity of the environment, sensitivity to pure rotations [21], [22]. These methods do not provide metric information that is required in some applications. This task is known as Scale ambiguity. The solution of this problem occurs in the literature. For example, in [23]], an approach is proposed for restoring metric information using information about the geometric arrangement of the sensor. In [24]] it was suggested to use additional sensors in the form of IMU. Initialization of the map is a separate task and is encountered in the literature [25],[26]. The type of camera shutter also contributes [27].

As the main sensor is often used LIDAR. With this sensor, the SLAM task can also be solved. So in the literature meets the solution of Hector SLAM [28].

To reduce costs and develop prototypes, Robot Operating System (ROS, [29]). has proved itself well. There are a large number of implemented solutions as separate modules. Many already done modules can support your idea. The module teb local planer [30] was designed for upgrade navigation stack ROS for car-like robot.

Combining information from various sensors is known as the task of sensor fusion. To solve this problem, use Kalman Filter(KF,[31]), Extended Kalman Filter(EKF, [32]), unscented Kalman Filter(UKF, [33]) and Particle Filter (PF, [34])

4.Objective

This paper aims to compare different methods for determining the position of AGV relative to the environment using well-known sensors and solutions for a specific application. A also consider the possibility of using different sensors at the same time to improve the solution of the task.

2.Code description

The solution contains two main part. The first part addressed to autonomous robot navigation. The second part is dataset based. It has analyzed information from different sensors with applying different methods for processing. It also has methods for sensor fusion with Extended Kalman Filter and Particle Filter.

1.Autonomous navigation car-like robot

This repository was designed to make research platform RACECAR autonomous move to special goal.

Link to the repository: https://github.com/mfilipen/self-driving-car

The main packages:

- car_joystick - it is used for teleoperation mode.
- cmd_vel_to_ackermann_drive - it is used for solving forward kinematics problem.
- emergency_traxxas - check system consistency.
- racecar_description - URDF model of the robot.
- racecar_navigation - it is used for ROS navigation stack.
- razor_imu_9dof - it is used for working with IMU sensor.
- traxxas_driver - it is low-level actuators driver.
- zed-ros-wrapper-1.0.0 - it is used for working with zed camera sensor.

2.Sensor fusion

This repository was designed for analyzing data from LIDAR, ZED camera, IMU and also has modules for sensor fusion with EKF and PF

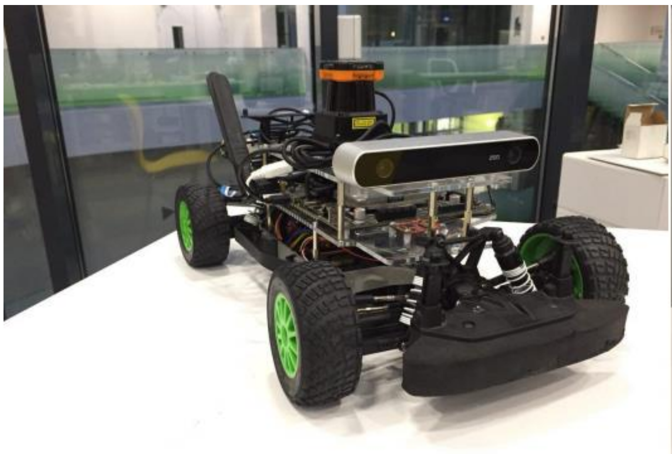
This research is dataset based. It requires dataset but suddenly it is too large for sharing via the internet (15 GB). If you need dataset contact to me.

Link to the repository: https://github.com/mfilipen/lidar_IMU_zed_fusion

The main packages:

- racecar_description - it is for rendering the dimensional model of the car in RViz
- sensor_fusion - it is for fusing data from different sensors.
- /sensor_fusion/scripts/ExtendedKalmanFilter/ - sensor fusion with Extended Kalman Filter
- /sensor_fusion/scripts/ParticalFilter/ - sensor fusion with Particle Filter
- other modules - data processing, error calculation, plotting data

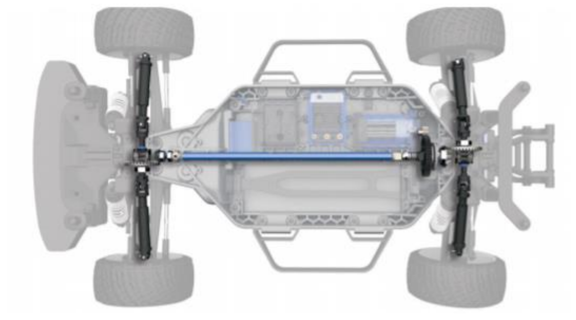
3.Experimental setup



The robot for experiment has Ackerman steering model, powerful NVidia Jetson TX1 for onboard computation, LIDAR (Hokuyo UTM-30LX) as the main sensor, driver, motor, servo for robot motion.

1.Hardware

Chassis - 4WD Traxxas #74076



It has Efficient Shaft-Driven 4WD, Low-CG Chassis. For more information: https://traxxas.com/products/models/electric/74076-1rally.

The computing unit - NVidia Jetson TX1

- NVIDIA Maxwell ™
- 256 CUDA cores
- Quad ARM® A57
- Memory: 4 GB 64 bit LPDDR4 25.6 GB/s
- Other: UART, SPI, I2C, I2S, GPIOs
- For more information: http://www.nvidia.com/object/embedded-systems-dev-kits-modules.html

2.Sensors

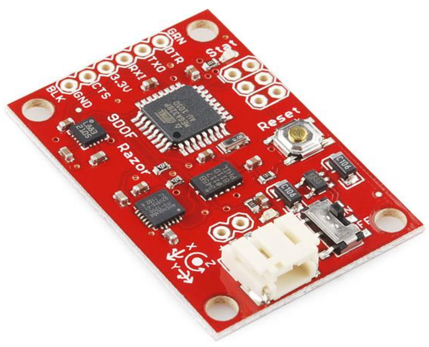
LIDAR - Hokuyo UTM-30LX

For more information: https://www.hokuyo-aut.jp/search/single.php?serial=169

Stereo camera - ZED

https://www.stereolabs.com/

Inertial measurement unit - Razor 9DoF SEN-10736



For more information: https://www.sparkfun.com/products/retired/10736

Driver & Motor SK-300058-04

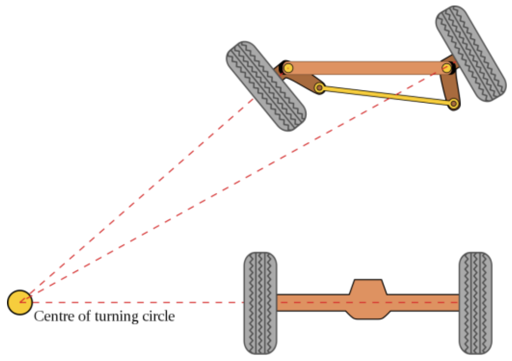
For more information: http://www.skyrc.com/Cheetah?search=%20SK-3000

3.Software

ROS indigo and Ubuntu 14.04 was chosen as foundation for reduce development cost. It are open source projects and have large community.

4.Robot model

1.Kinematic model - Ackerman steering model

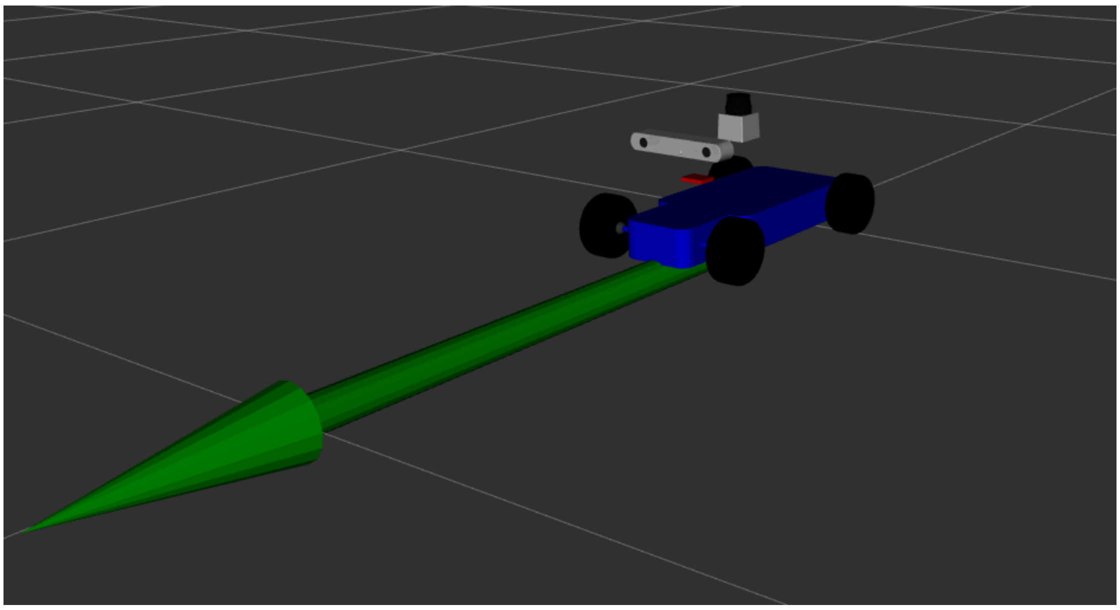


The robot platform satisfies to Ackerman steering model. It is used as motion model of experimental setup.

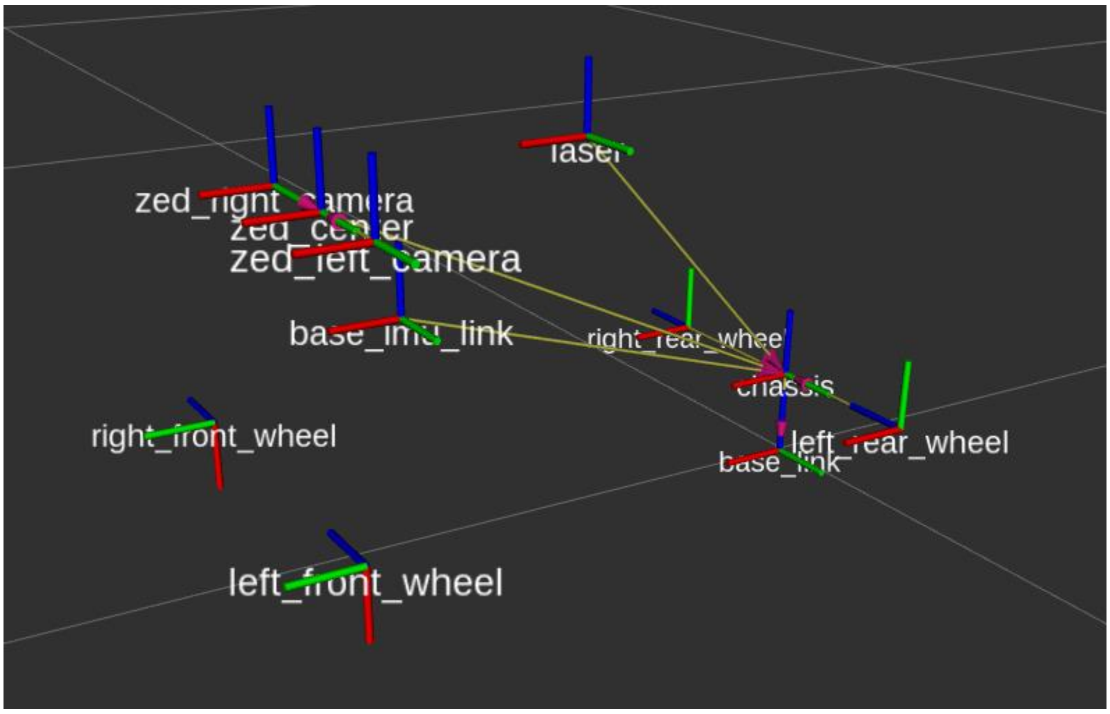
For more information: https://en.wikipedia.org/wiki/Ackermann_steering_geometry

2.URDF Robot Model

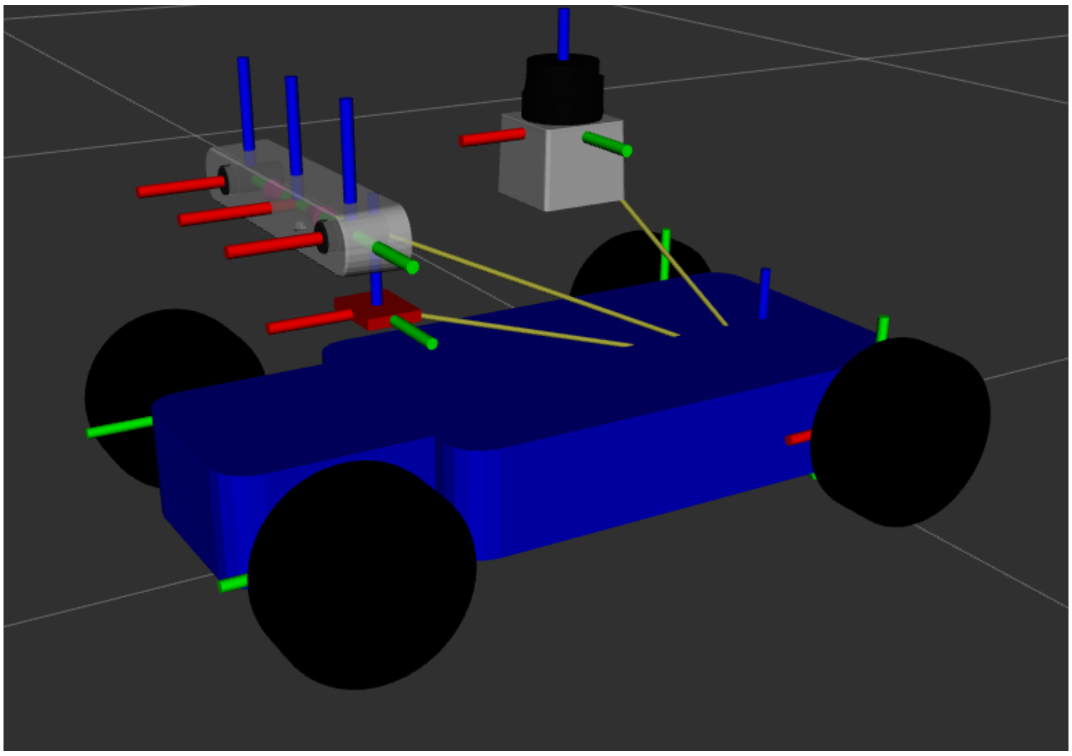
It was designed URDF model for simplify coordinate transformation and have easy understandable visualization of the robot.



It has coordinate frame for all-important part of the robot. These parts have hard connection to the base link, which was chosen according to kinematic model of the robot.

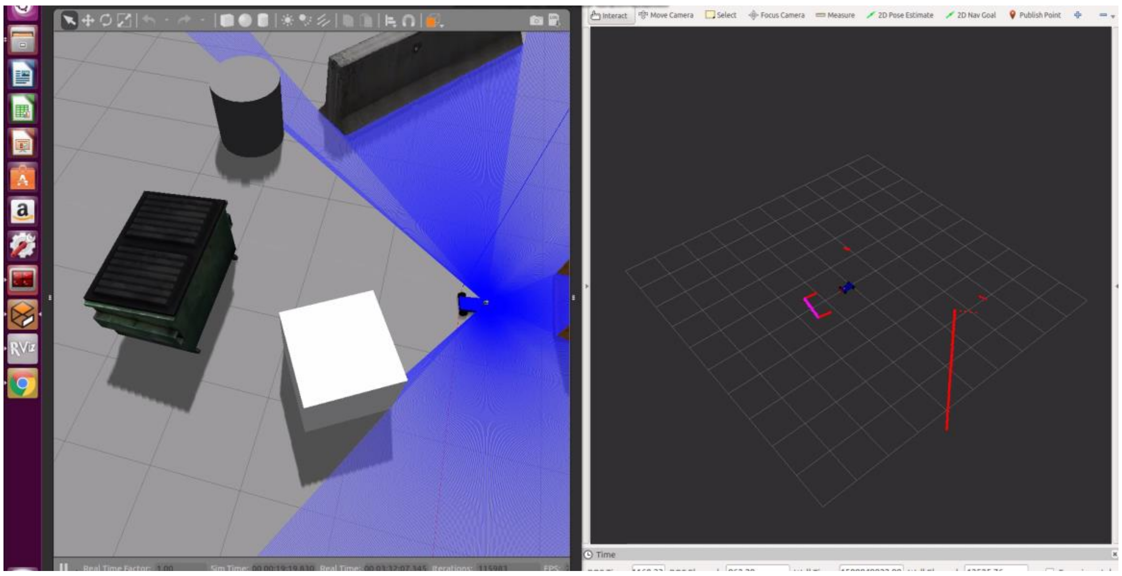


Distance between different components of the model corresponds with real robot model.



3.Robot modeling in gazebo

The robot model has mass and can be used in simulator gazebo. However, at this point it has not working kinematic plugin for gazebo.

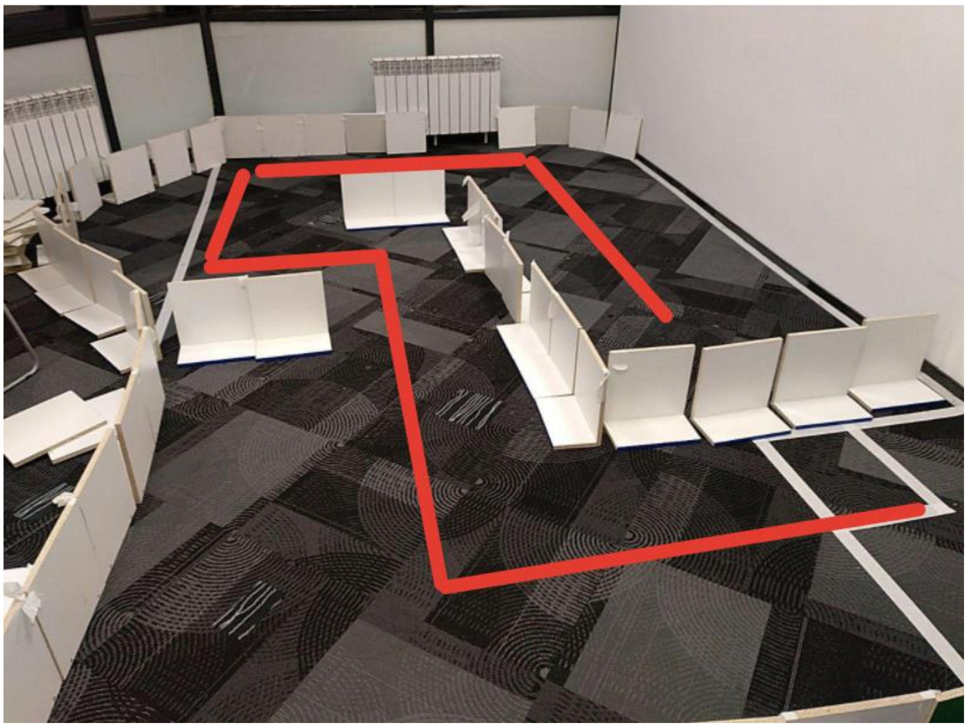


For more information: https://www.youtube.com/watch?v=yt9Jz-HH_zg

5.Autonomous navigation

1.Test environment

The labyrinth with at least 5 turns was used as test environment for the experiment.

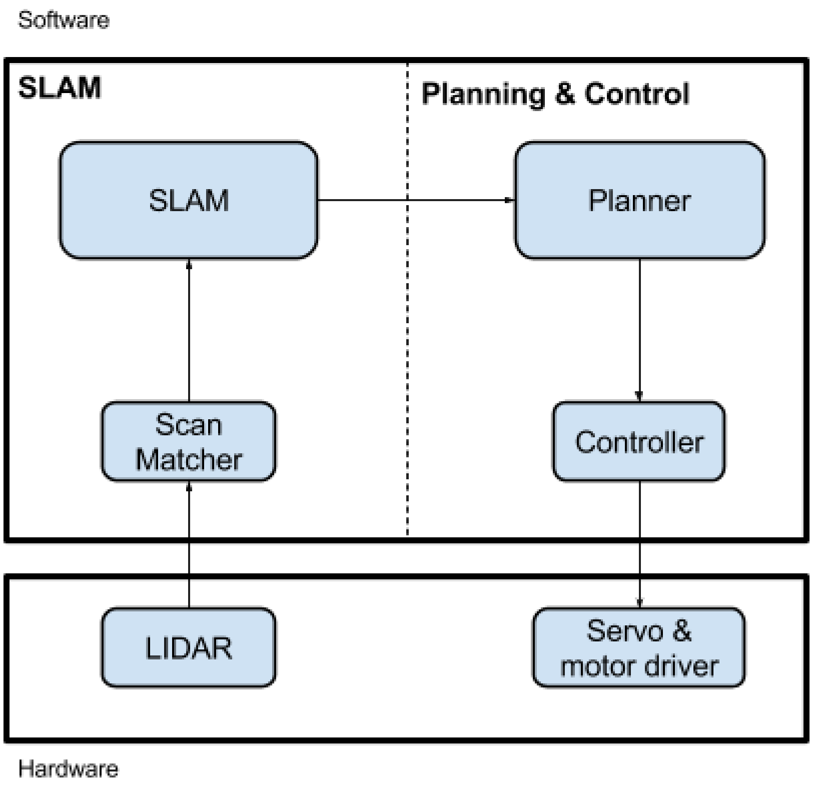


2.Experimental conditions

The robot was placed in test environment. All computation was on board. The host computer was connected via ssh connection and was used to send navigation goal command. The goal of the prototype is achieve to suggested goals.

3.Architecture

The main goal of this test setup is evaluate robot planning algorithm because of it was decided to do not do this system complicated with large amount of sensors. The LIDAR was used as main sensor. It provides information about environment. Software build map of environment and solve localization problem based on this information. Finally, planner build appropriate way and code this way to exact motions of actuators base on map and location.

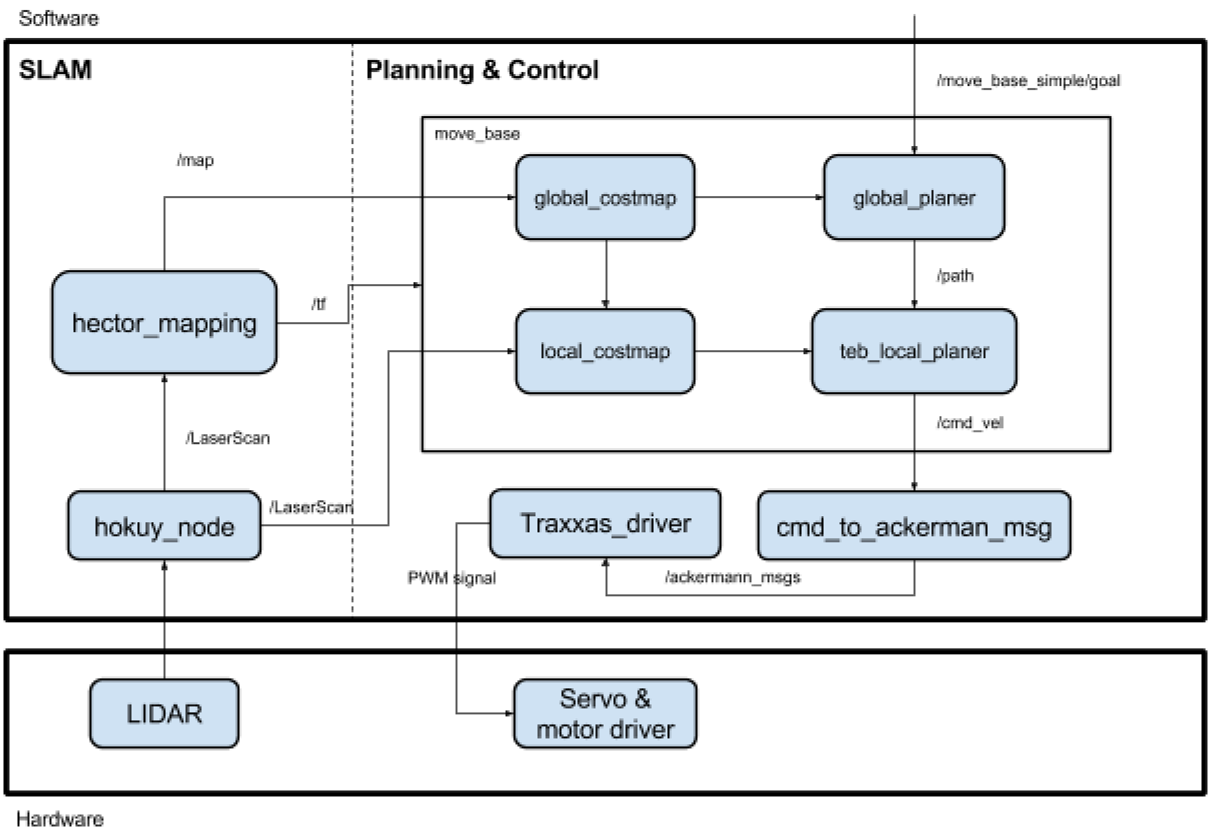


4.ROS Architecture

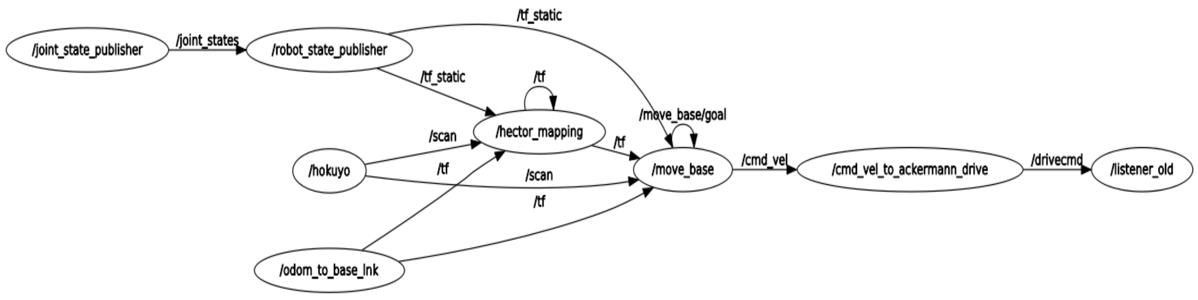
The following nodes were used:

hokuyo_node (http://wiki.ros.org/hokuyo_node) for collecting raw data from sensors. hector_mapping (http://wiki.ros.org/hector_mapping) was used for generating a map from LIDAR data and solving localization problem. move_base (http://wiki.ros.org/move_base) is part of ROS navigation stack (http://wiki.ros.org/navigation) and is used for solving global and local path planning problem. But default local planner is not applicable for Ackerman steering model. It was replaced with teb_local_planner (http://wiki.ros.org/teb_local_planner) for build a local plan for car-like robot. cmd_to_ackerman_msg is a node for solving kinematics for Ackermann steering model. Traxxas_driver is used for sending signals to the actuators.

The following schema describes how nodes connected and works.

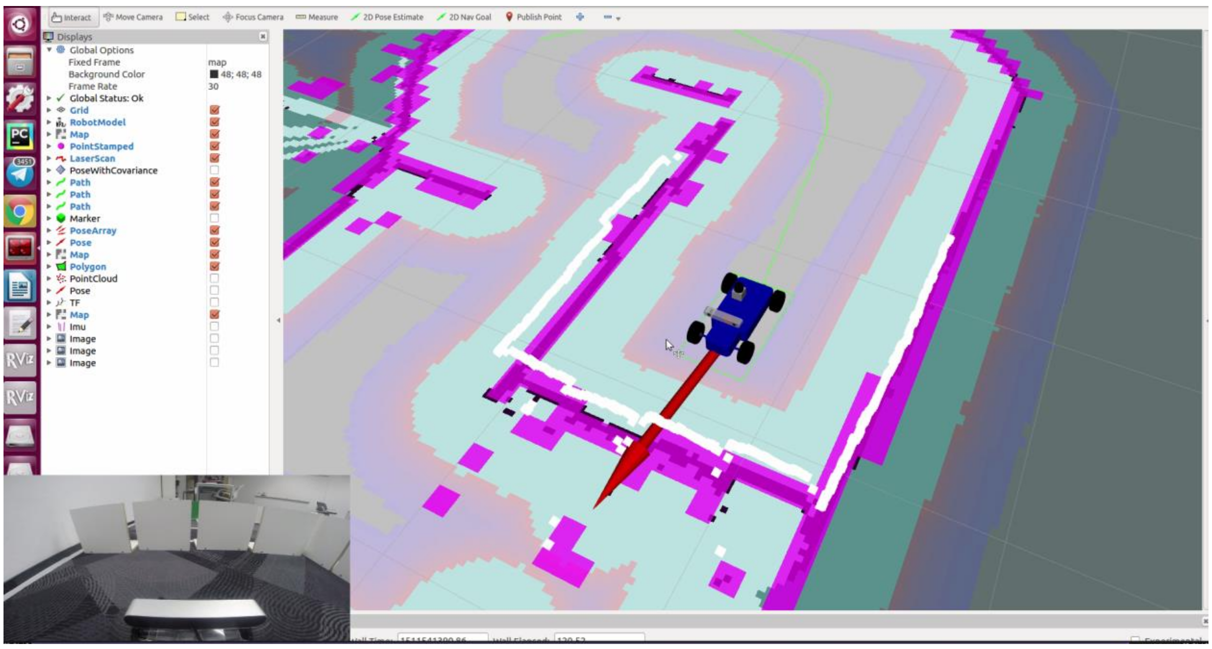


Next schema shows topic exchanges from different nodes.



After fitting all parameters of the system we have good enough autonomous mobile robot.

5.Example of working autonomous robot navigation



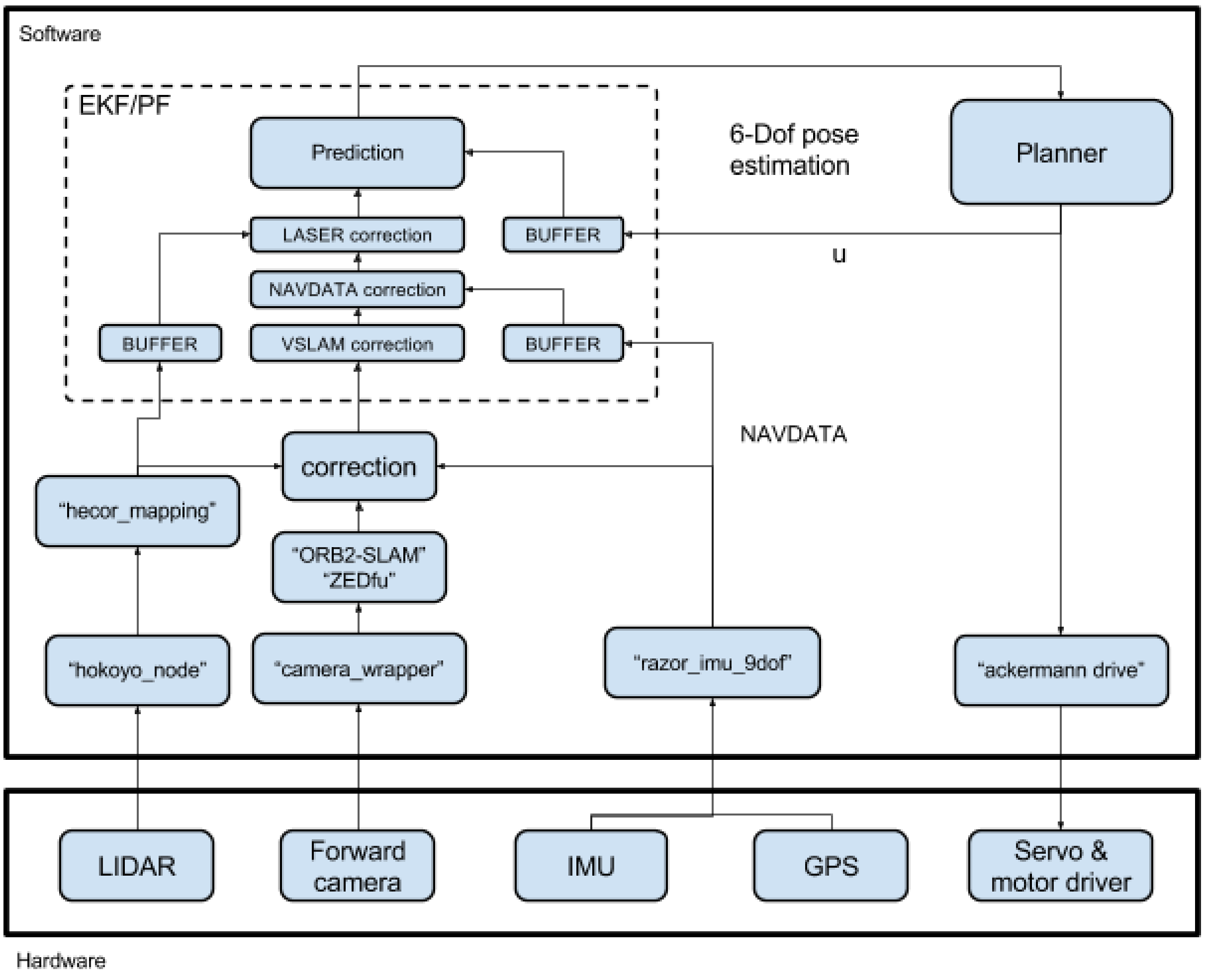
Fixed Frame map
Background Color 48; 48; 48
Frame Rate 30

Video available via link: https://www.youtube.com/watch?v=mu7d6ygTFf0

6.Sensor Fusion

1.Architecture

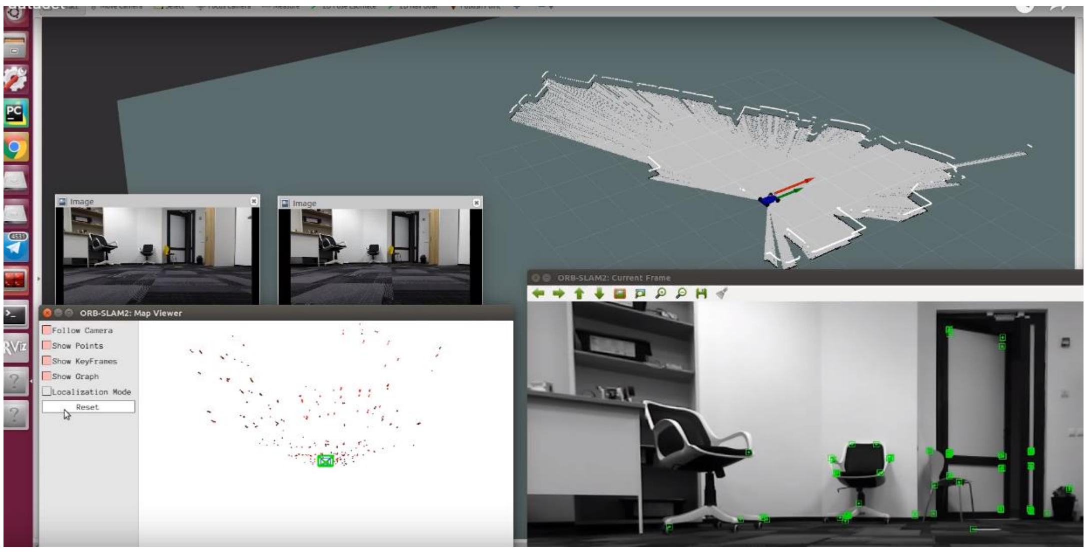
The goal is combine data from different sensors and control commands for more precise pose estimation.



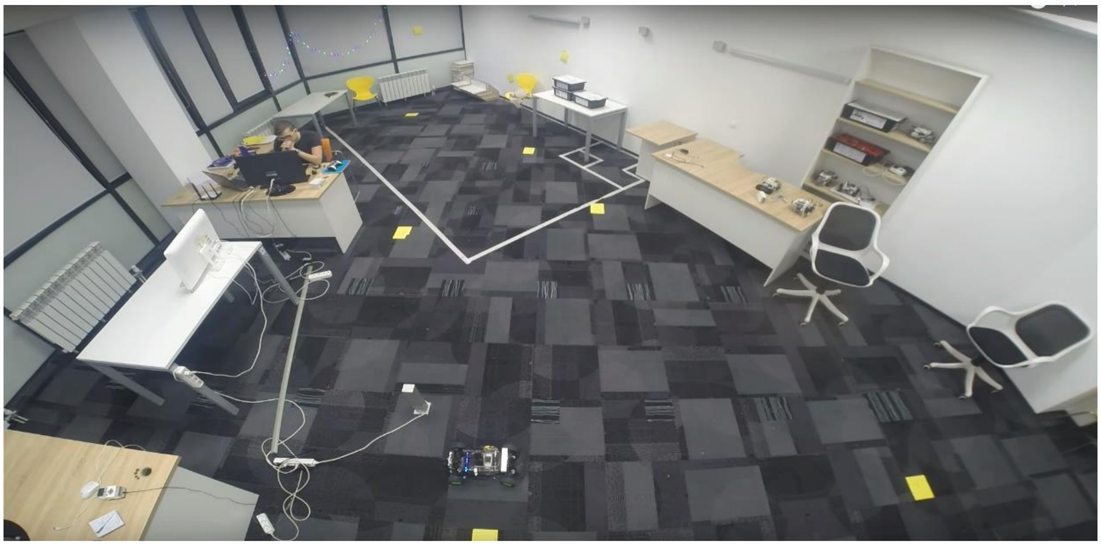
2.Dataset

It was collected a dataset to examine data from different sensors. Experiment was organized in this way. The robot was placed in test environment, which was usual office room. The robot was tele operated by kinematic mode from host computer via ROS. The robot has finished three full circles and record data from sensors.

The following videos show progress of experiment



Video available via link: https://www.youtube.com/watch?v=TLYLvVxFIdE



Video available via link: https://www.youtube.com/watch?v=RBj-wnN51l4

Only ORB SLAM was processed on dataset. All other algorithms computing in run time on board.

3.Sensor calibration

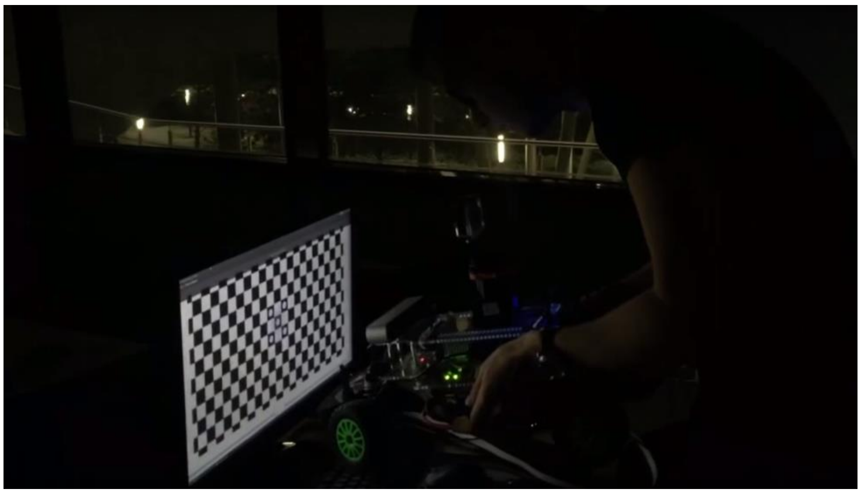
9DOF Razor IMU calibration

I found that default calibration does not provide enough precise information. It was necessary to do calibration before fusing data. The procedure of calibration was done how it was described here. I had many problems with the installation of programs for calibration on Jetson TX1 (which is used as the basis for the robot). It is found that there is a problem with running Oracle Java (which is required for run calibration software) on Jetson TX1 (last version, which I tried - Linux ARM 64 Hard Float ABI). Finally, calibration was done using another computer.

Video available via link: https://www.youtube.com/watch?v=6SqMOvdJ9xU

ZED camera calibration

ZED camera was calibrated. A precise calibration allows as to getting metric information about environment.

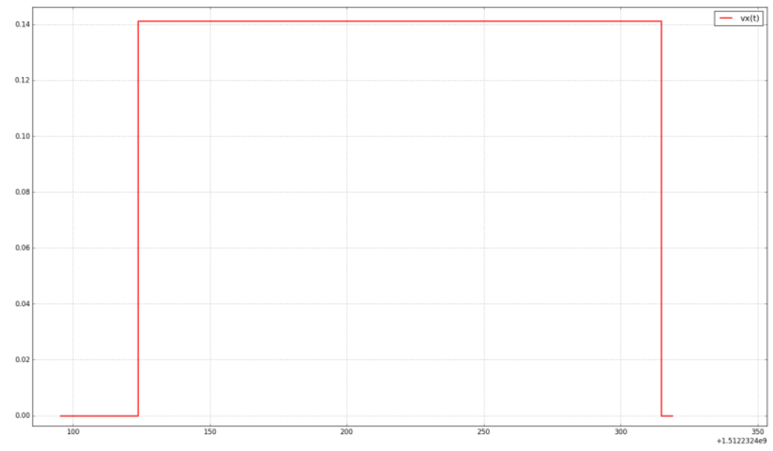


Video available via link: https://www.youtube.com/watch?v=oGXNo2LMekM

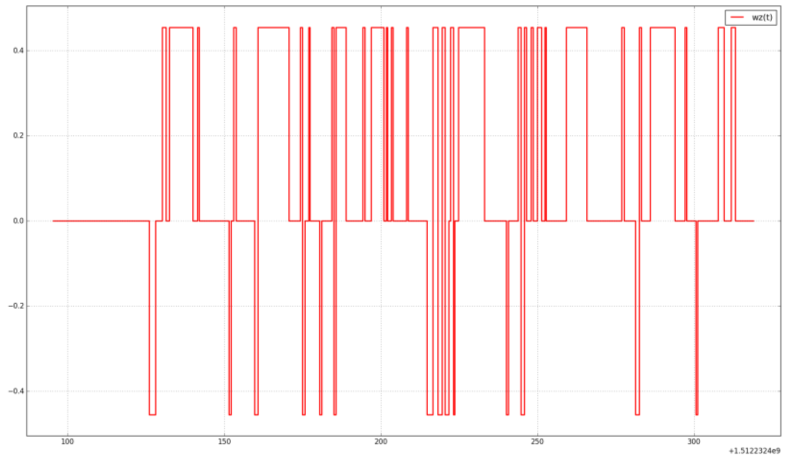
7.Data

1.Control command Data

The robot has two main parameters. It are steering angle and speed of rotation of the motor. The robot already had a first principle model. Because of it we have as input control commands $vx(t) = f(rotation\ of\ the\ motor)$ and$wz(t) = g(sterling\ angel, parameters\ of\ the\ robot)$.



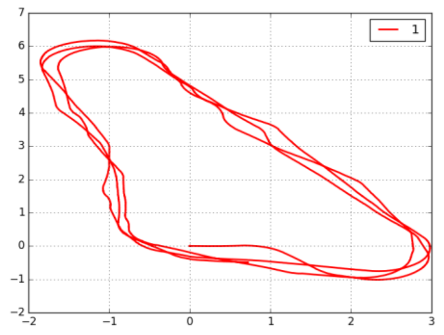
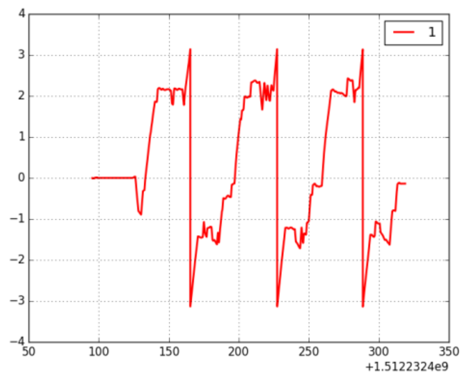
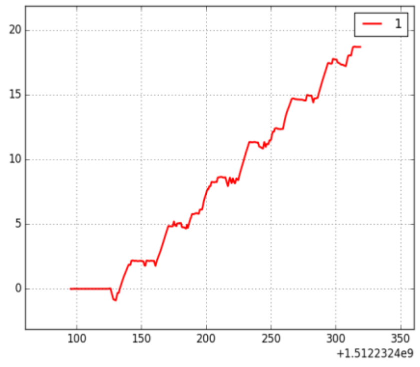
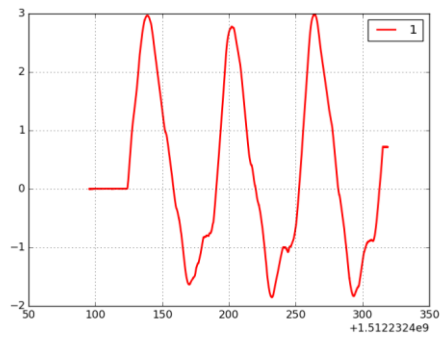
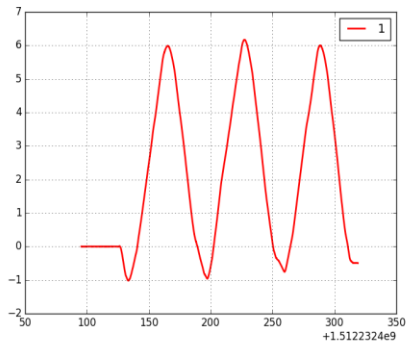
$vx(t)$ - Linear speed in coordinate frame attached to the robot.



$wz(t)$ - Speed of rotation.

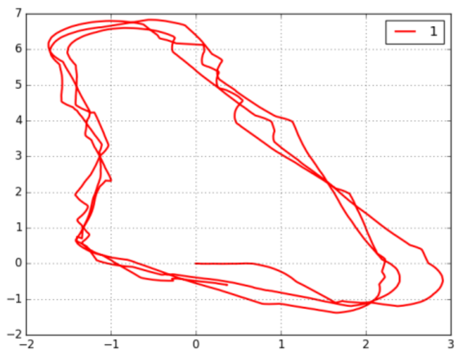
2.Pose Lidar Data

hector_slam (http://wiki.ros.org/hector_slam) was used for solving localization problem from LIDAR data.

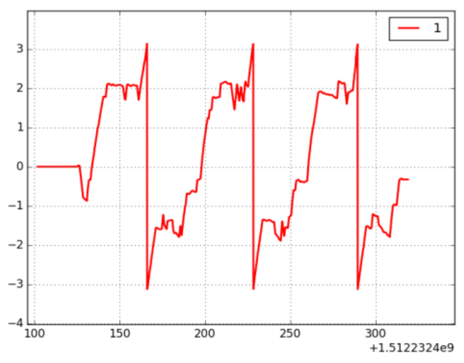
	
The trajectory of the robot (x,y).	
	
yaw(t) normalized	yaw(t)
	
x(t)	y(t)

3.Pose ZED camera Data

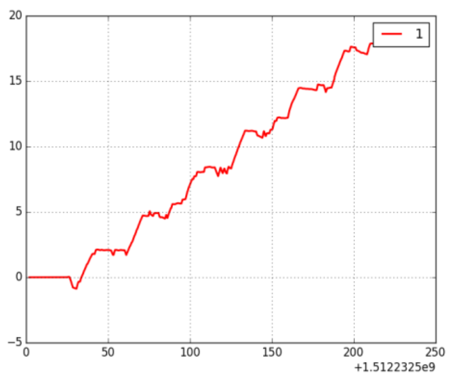
ZEDfu provides metric information about pose.



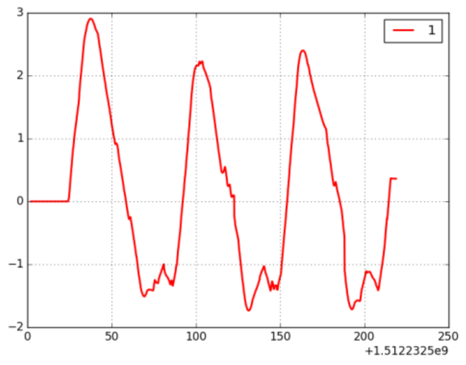
The trajectory of the robot (x,y).



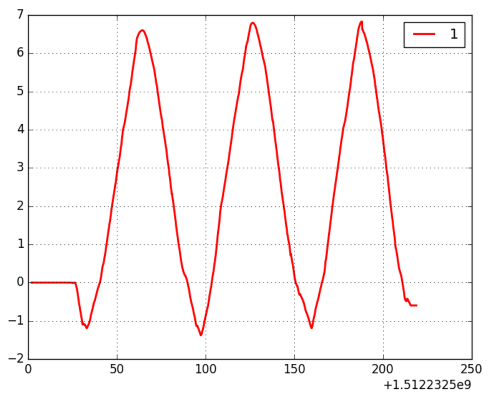
yaw(t) normalized



yaw(t)



x(t)

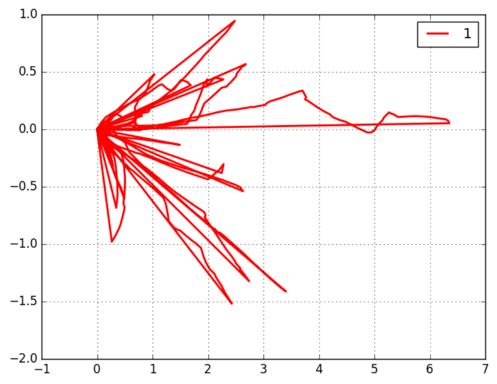
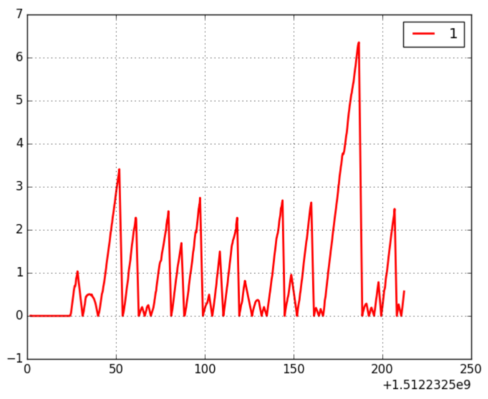
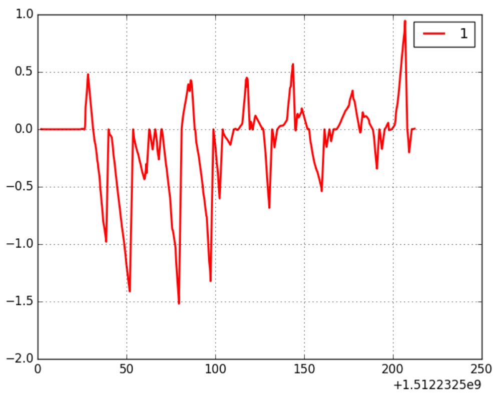


y(t)

4.ORB SLAM Pose Data

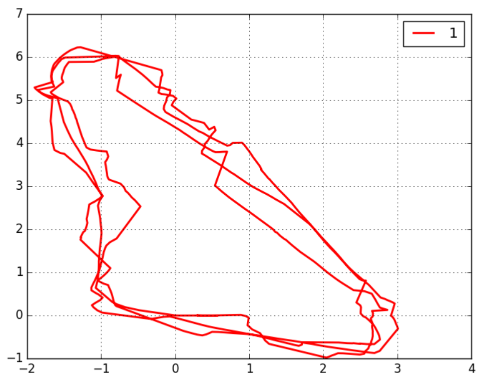
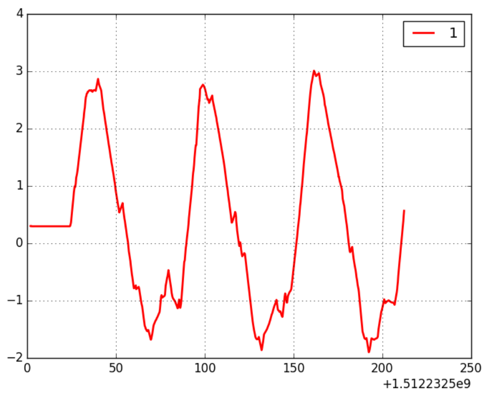
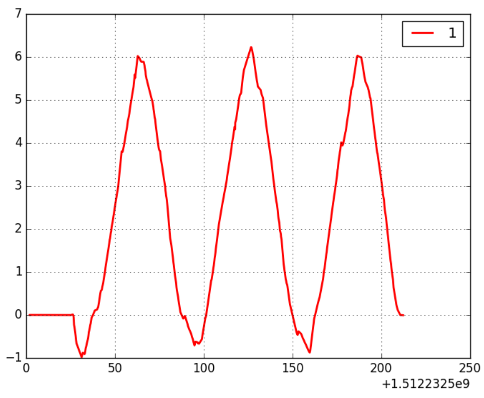
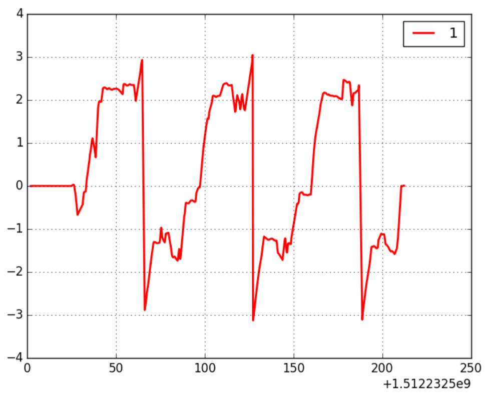
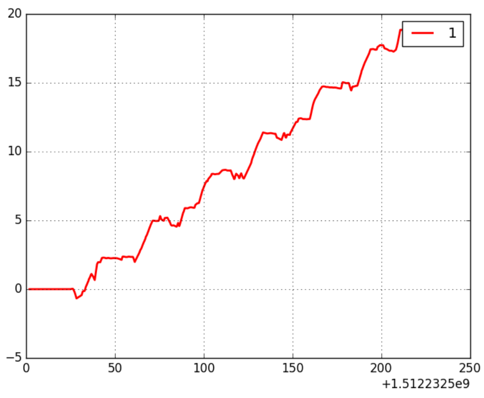
As i have calibrated stereo camera it is possible with ORB_SLAM to get metrical pose information. But the main problem that this approach is not resistant to the fast rotation, but it is provide additional information on straight line moving.

Raw data ORB SLAM odometry

	
(x,y)	
	
x(t)	y(t)

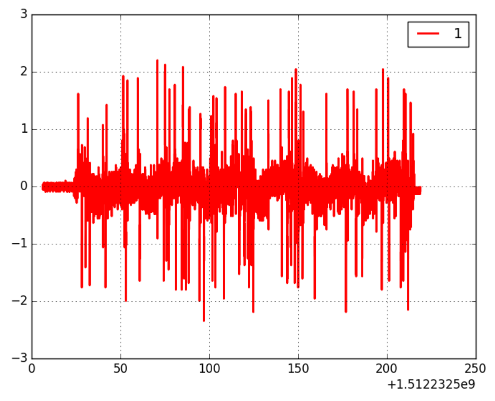
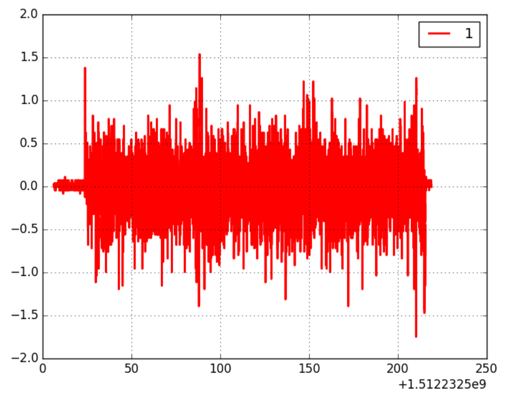
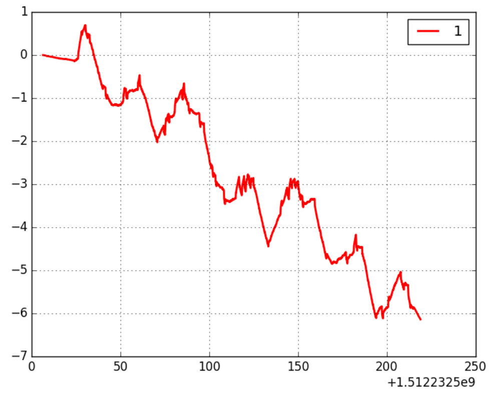
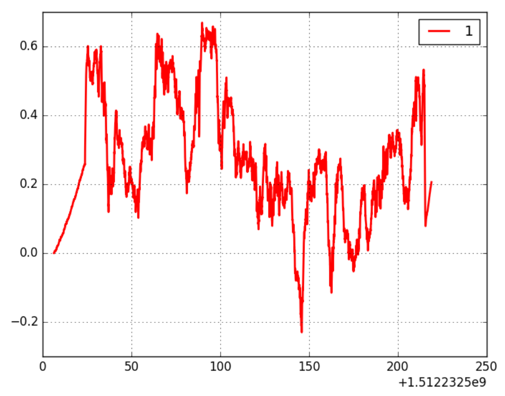
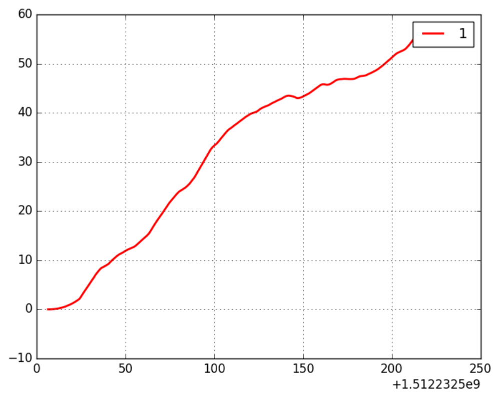
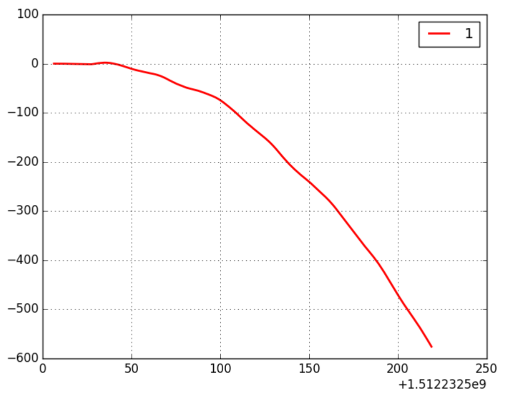
The result complete in line with expectation. As it lose track we have map reset and pose to zero update. The idea is initialize the each fragment with data from lidar.

The idea is to initialize each fragment with data from LIDAR odometry. Using this approach we can additional information from sensor about pose on straight enough segments of way. Finally, we have following graphics.

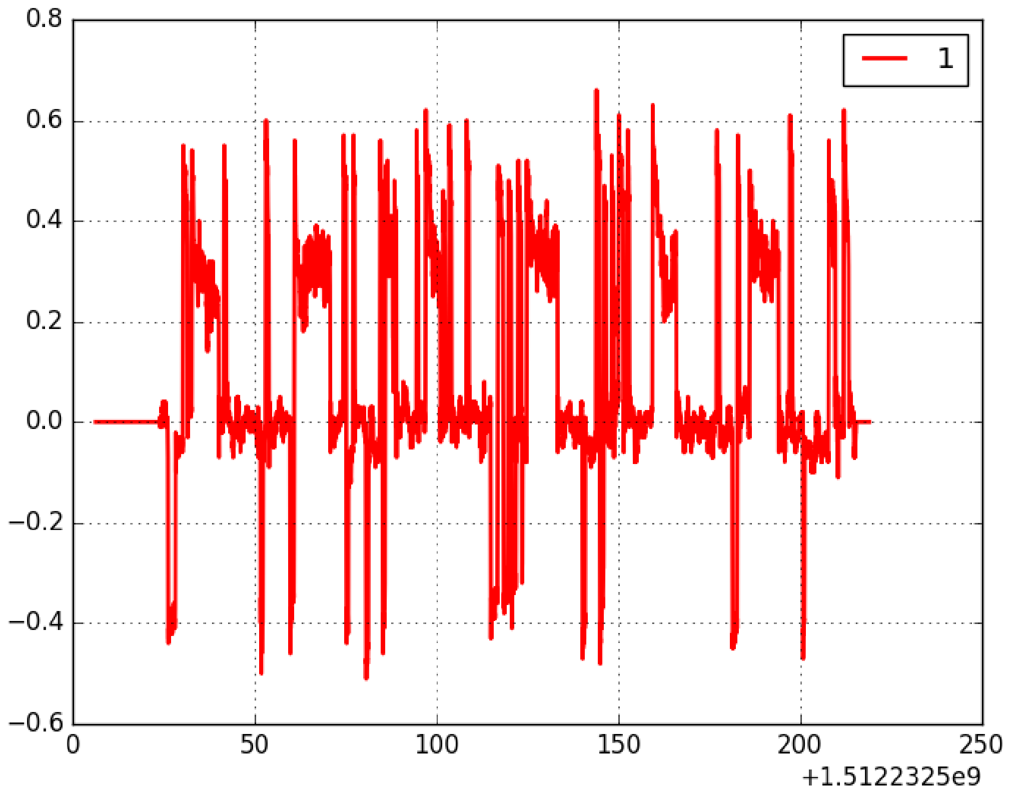
	
(x,y)	
	
x(t)	y(t)
	
yaw(t) normalized	yaw(t)

5.IMU data

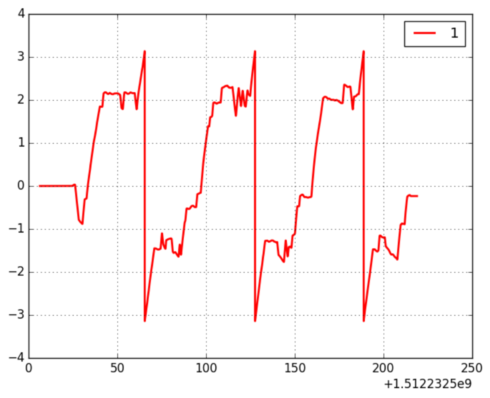
Accelerometer data

	
ax(t)	ay(t)
	
vx(t) - numerical first integral ax(t)	vy(t) - numerical first integral ay(t)
	
x(t) - numerical second integral ax(t)	y(t) - numerical second integral ay(t)

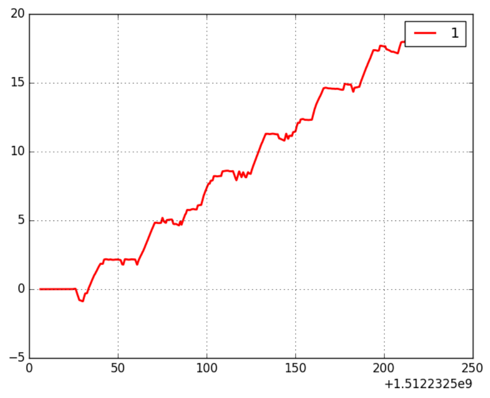
Gyroscope data



$wz(t)$ - speed of rotation.

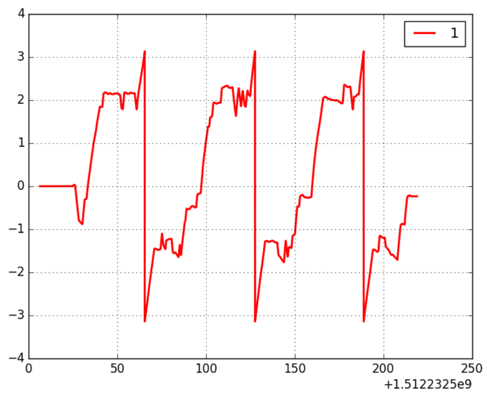
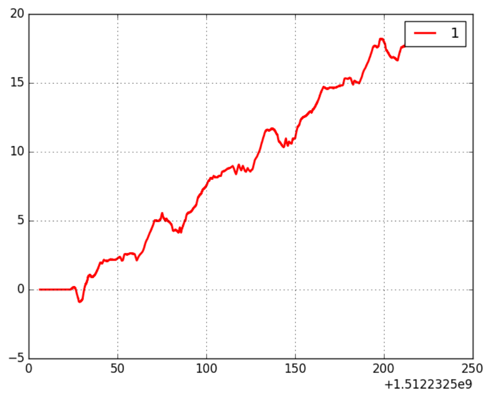


yaw(t) - numerical first integral with normalization



yaw(t) - numerical first integral

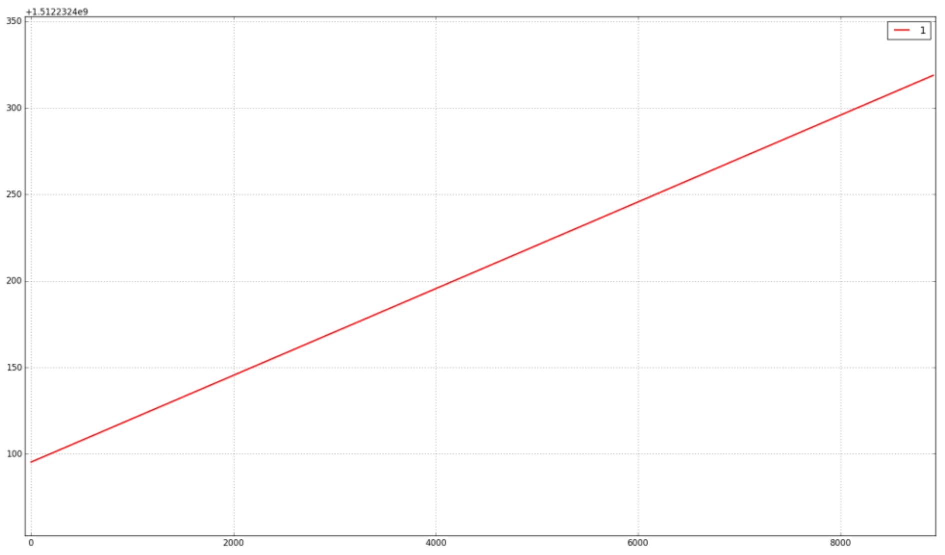
Magnetometer data

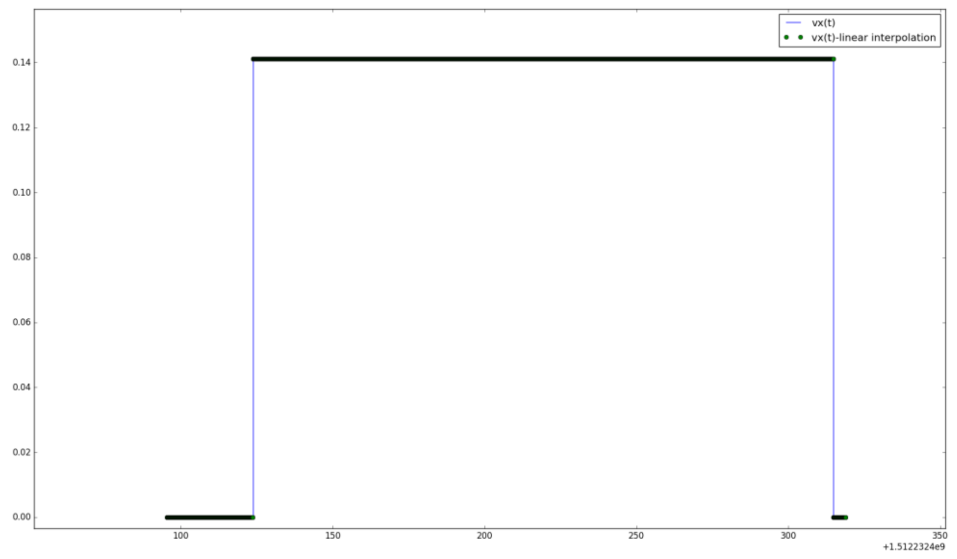
	
yaw(t) normalized	yaw(t)

8.Data Set Preparation

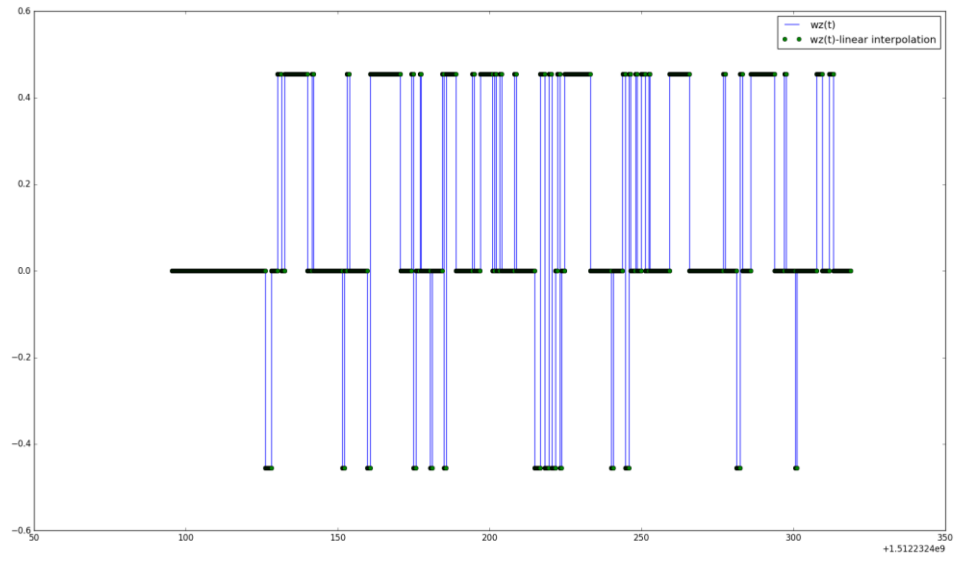
1.Control command interpolation

As we have different frequency of the data from control command and lidar. Control commands data were interpolated in time grid of hector_SLAM.


(№, t_i)- the time grid is linear. Time stamp is step 0.0250586 s.



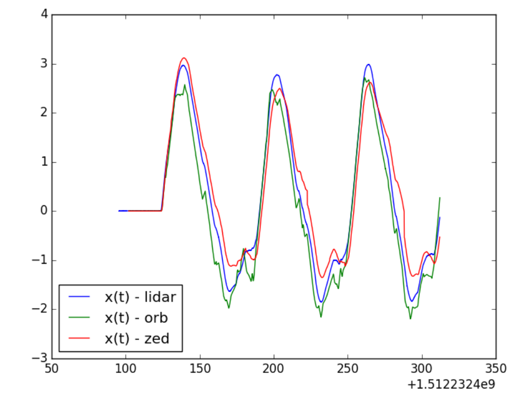
$vx(t)$- linear interpolation.

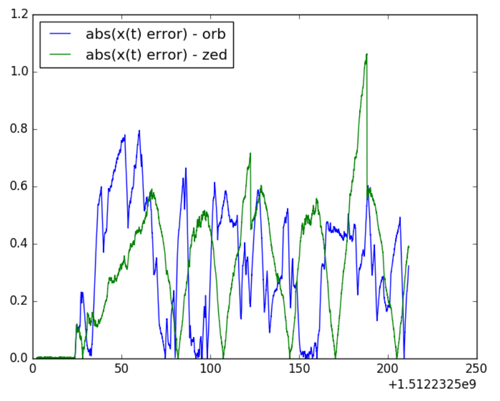


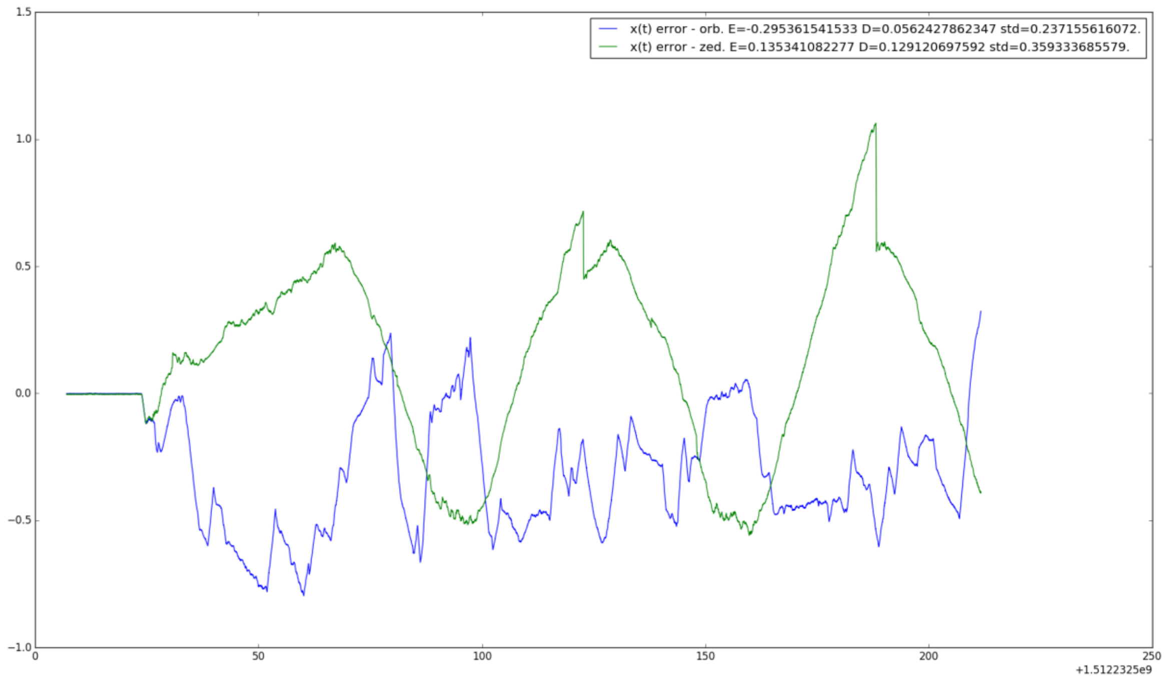
$wz(t)$- linear interpolation.

2.Data for sensor fusion

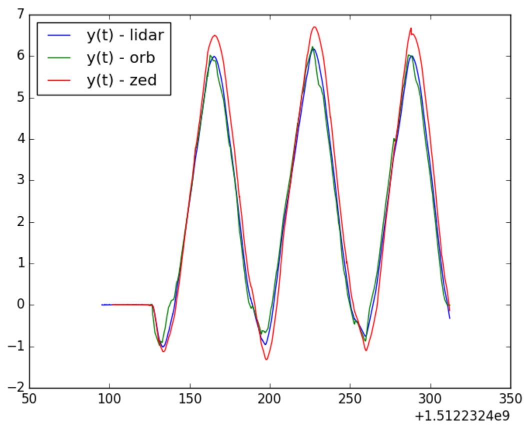
X_POSE data

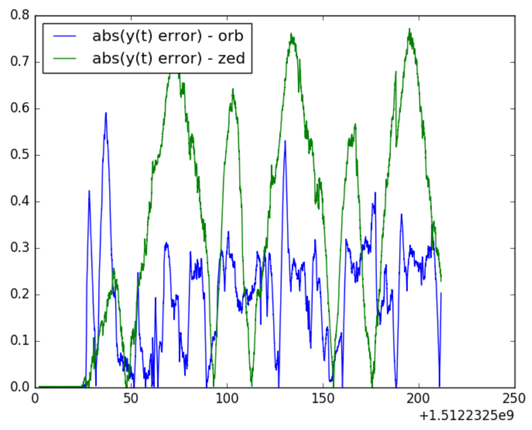


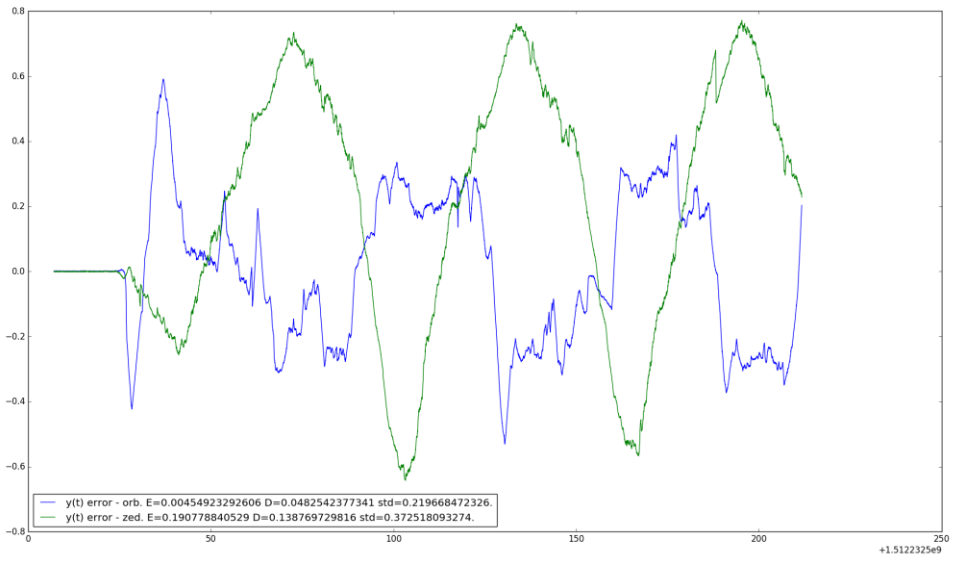




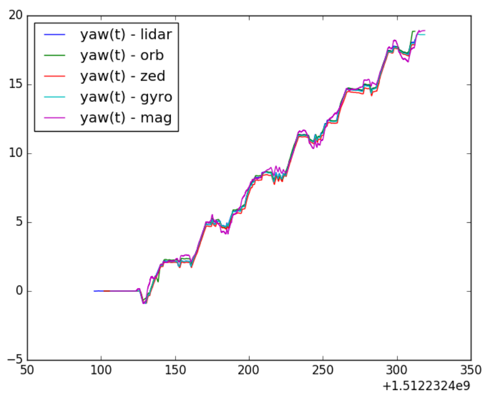
Y_POSE data

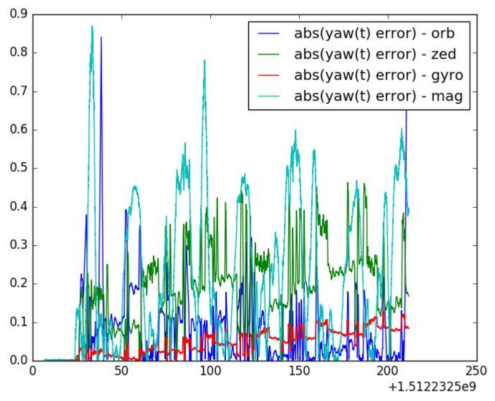

y(t) - lidar
y(t) - orb
y(t) - zed
+1.5122324e9

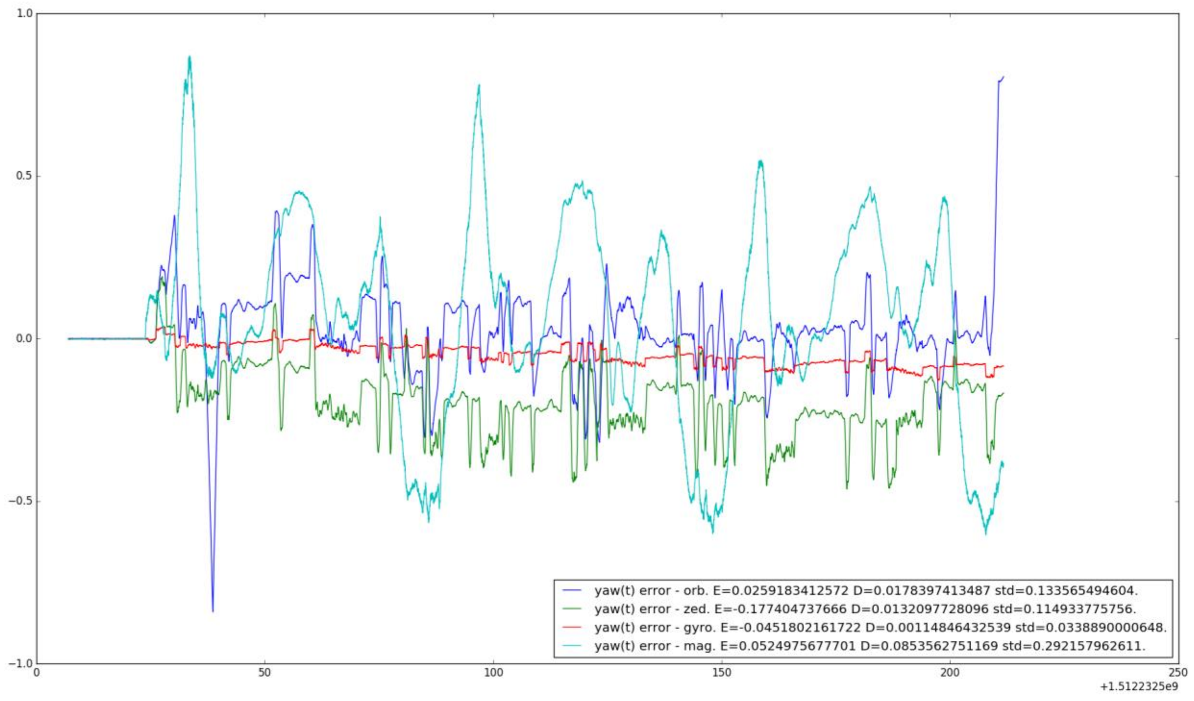

abs(y(t) error) - orb
abs(y(t) error) - zed
+1.5122325e9


y(t) error - orb. E=0.00454923292606 D=0.0482542377341 std=0.219668472326.
y(t) error - zed. E=0.190778840529 D=0.138769729816 std=0.372518093274.
+1.5122325e9

YAW_pose data







9.Sensor Fusion

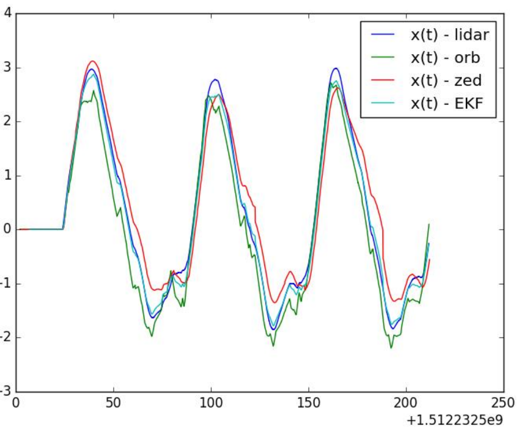
Motion model

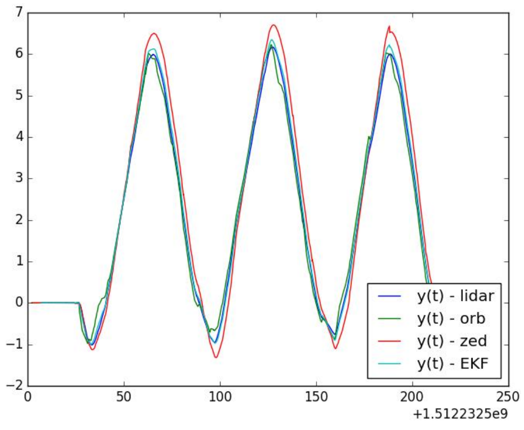
$$yaw_i = yaw_{i-1} + w_{z_i} \cdot dt_i$$

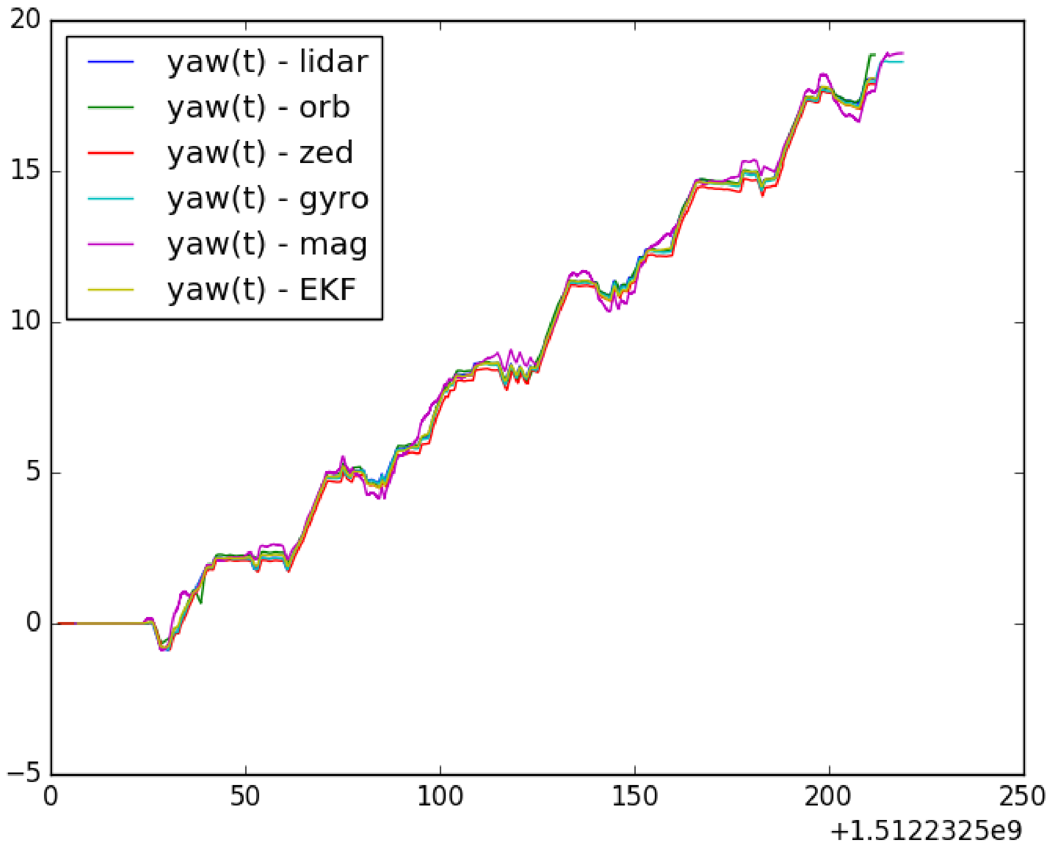
$$x_i = x_{i-1} + v_i \cdot cos(yaw_i) \cdot dt_i$$

$$y_i = y_{i-1} + v_i \cdot sin(yaw_i) \cdot dt_i$$

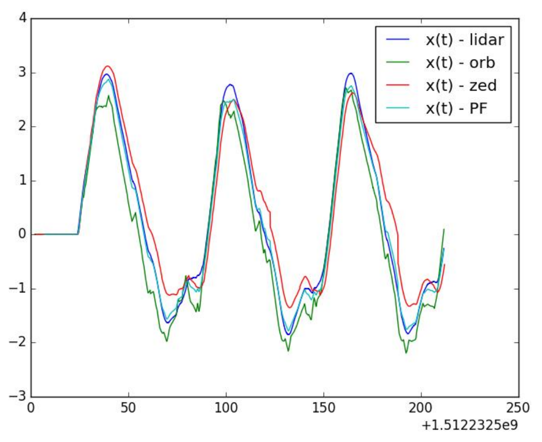
Extended Kalman Filter

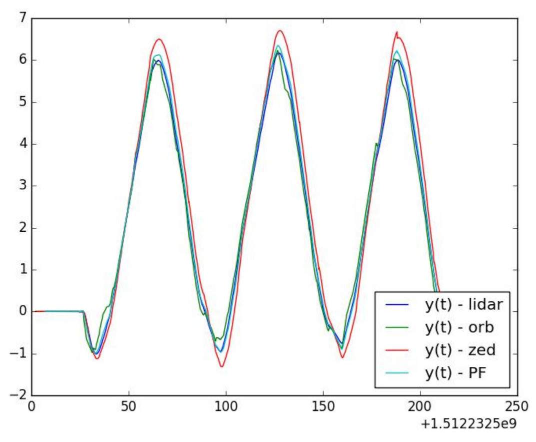


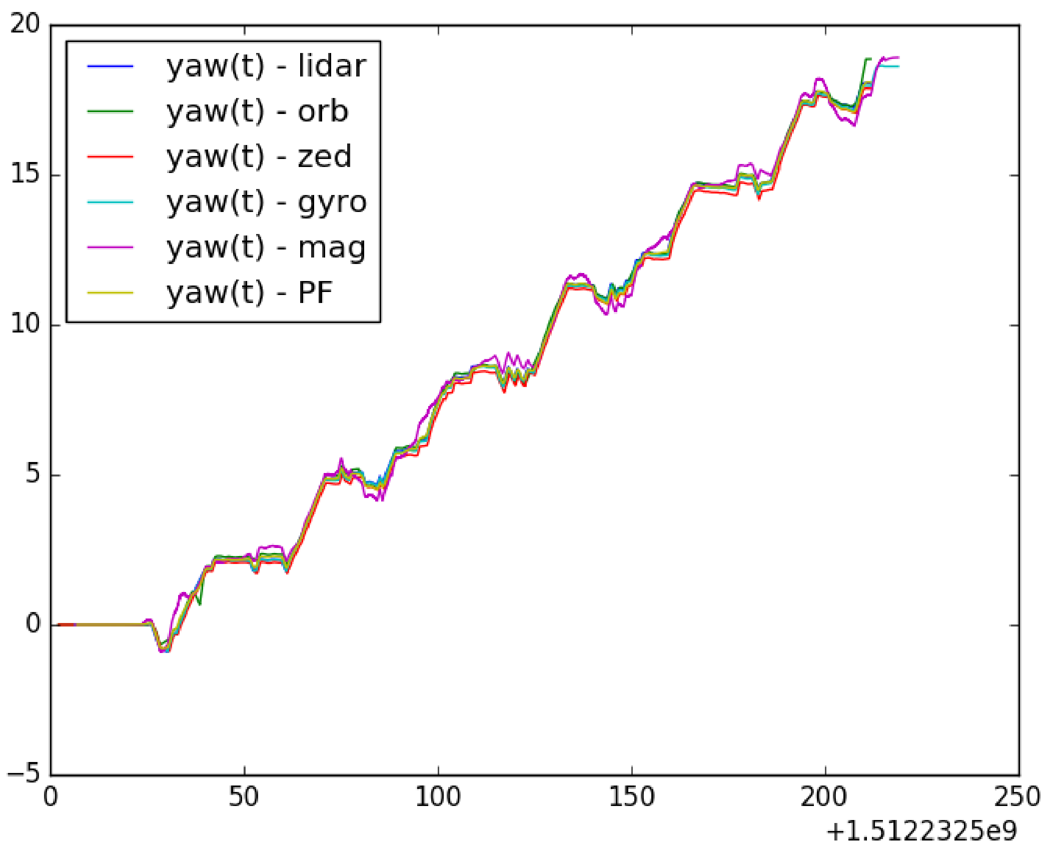




Particle Filter







10.Conclusion

During the work was designed real world autonomous car-like robot. Created robot model. Evaluated and comparing work of different sensors and methods for precise pose estimation of robot.

11.Literature

[1] T. Bailey and H. Durrant-Whyte, “Simultaneous localization and mapping (SLAM): Part I,” *IEEE Robot. Autom. Mag.*, vol. 13, no. 3, pp. 108–117, 2006.

[2] B. Barshan and H. F. Durrant-Whyte, “Inertial Navigation Systems for Mobile Robots,” *IEEE Trans. Robot. Autom.*, vol. 11, no. 3, pp. 328–342, 1995.

[3] T. Taketomi, H. Uchiyama, and S. Ikeda, “Visual SLAM algorithms: a survey from 2010 to 2016,” *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, p. 16, 2017.

[4] J. Fuentes-Pacheco, J. Ruiz-Ascencio, and J. M. Rendón-Mancha, “Visual simultaneous localization and mapping: a survey,” *Artif. Intell. Rev.*, vol. 43, no. 1, pp. 55–81, 2012.

[5] S. Se, D. Lowe, and J. Little, “Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks,” *Int. J. Rob. Res.*, vol. 21, no. 8, pp. 735–758, 2002.

[6] C. F. Olson, L. H. Matthies, M. Schoppers, and M. W. Maimone, “Rover navigation using stereo ego-motion,” *Rob. Auton. Syst.*, vol. 43, no. 4, pp. 215–229, 2003.

[7] A. J. Davison, “Real-time Simultaneous Localisation and Mapping with a Single Camera,” *Iccv*, vol. 2, pp. 1403–1410, 2003.

[8] and D. M. K. Georg, “Parallel Tracking and Mapping for Small ARWorkspaces,” pp. 225–234, 2007.

[9] B. Y. T. I. M. Bailey and H. Durrant-whyte, “Simultaneous Localization and Mapping (SLAM),” *Update*, vol. 13, no. September, pp. 108–117, 2006.

[10] M. Pizzoli, C. Forster, and D. Scaramuzza, “REMODE : Probabilistic , Monocular Dense Reconstruction in Real Time,” pp. 2609–2616, 2014.

[11] E. Rosten and T. Drummond, “Machine Learning for High Speed Corner Detection,” *Comput. Vis. -- ECCV 2006*, vol. 1, pp. 430–443, 2006.

[12] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” pp. 778–792, 2010.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, “Orb-slam: a versatile and accurate monocular slam system,” *IEEE Trans. Robot.*, vol. 31, no. 5, pp. 1147–1163, 2015.

[14] E. Rublee and G. Bradski, “ORB : an efficient alternative to SIFT or SURF,” *Distribution*, pp. 2564–2571, 2011.

[15] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, “DTAM: Dense Tracking and Mapping in Real-Time,” pp. 2320–2327, 2011.

[16] J. Engel, T. Schöps, and D. Cremers, “LSD-SLAM: Large-Scale Direct monocular SLAM,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8690 LNCS, no. PART 2, pp. 834–849, 2014.

[17] C. Forster, M. Pizzoli, and D. Scaramuzza, “SVO: Fast semi-direct monocular visual odometry,” *Proc. - IEEE Int. Conf. Robot. Autom.*, pp. 15–22, 2014.

[18] J. Engel, V. Koltun, and D. Cremers, “Direct Sparse Odometry,” vol. 8828, no. c, 2016.

[19] T. Whelan, S. Leutenegger, R. Salas Moreno, B. Glocker, and A. Davison, "ElasticFusion: Dense SLAM Without A Pose Graph," *Robot. Sci. Syst. XI*, 2015.

[20] K. Tateno, F. Tombari, I. Laina, and N. Navab, "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction," 2017.

[21] S. Gauglitz, C. Sweeney, J. Ventura, M. Turk, and T. Höllerer, "Live tracking and mapping from both general and rotation-only camera motion," *ISMAR 2012 - 11th IEEE Int. Symp. Mix. Augment. Real. 2012, Sci. Technol. Pap.*, pp. 13–22, 2012.

[22] C. Pirchheim, D. Schmalstieg, and G. Reitmayr, "Handling pure camera rotation in keyframe-based SLAM," *2013 IEEE Int. Symp. Mix. Augment. Reality, ISMAR 2013*, no. October, pp. 229–238, 2013.

[23] Z. Dingfu, Y. Dai, and H. Li, "Reliable scale estimation and correction for monocular Visual Odometry," *IEEE Intell. Veh. Symp. Proc.*, vol. 2016–Augus, no. Iv, pp. 490–495, 2016.

[24] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *J. Intell. Robot. Syst. Theory Appl.*, vol. 61, no. 1–4, pp. 287–299, 2011.

[25] A. Mulloni, M. Ramachandran, G. Reitmayr, D. Wagner, R. Grasset, and S. Diaz, "User friendly SLAM initialization," *2013 IEEE Int. Symp. Mix. Augment. Reality, ISMAR 2013*, no. October, pp. 153–162, 2013.

[26] C. Arth, C. Pirchheim, J. Ventura, D. Schmalstieg, and V. Lepetit, "Instant Outdoor Localization and SLAM Initialization from 2.5D Maps," *IEEE Trans. Vis. Comput. Graph.*, vol. 21, no. 11, pp. 1309–1318, 2015.

[27] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline Fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras," *Procedings Br. Mach. Vis. Conf. 2013*, p. 93.1-93.11, 2013.

[28] S. Kohlbrecher, O. Von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable SLAM system with full 3D motion estimation," *9th IEEE Int. Symp. Safety, Secur. Rescue Robot. SSRR 2011*, pp. 155–160, 2011.

[29] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009.

[30] C. Rösmann, F. Hoffmann, and T. Bertram, "Planning of multiple robot trajectories in distinctive topologies," *2015 Eur. Conf. Mob. Robot. ECMR 2015 - Proc.*, 2015.

[31] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *J. Basic Eng.*, vol. 82, no. 1, p. 35, 1960.

[32] K. Fujii, "Extended Kalman Filter," *Ref. Man.*, 2013.

[33] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," *Proc. IEEE 2000 Adapt. Syst. Signal Process. Commun. Control Symp. (Cat. No.00EX373)*, pp. 153–158, 2002.

[34] S. Thrun, "Particle Filters in Robotics (Invited Talk)," 2012.