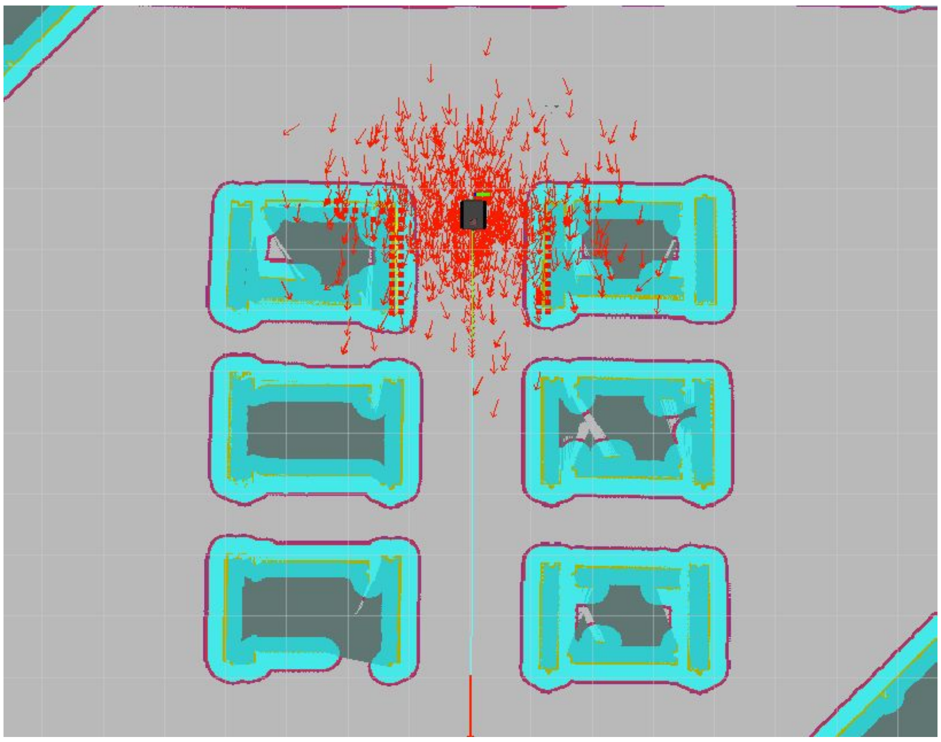
UNIVERSITY OF GOTHENBURG



Systematic Evaluation of Selected Algorithms for Sensor Based Localization for a Mini Autonomous Vehicle

Sensor fusion and localization algorithms' evaluation in a simulation framework under experimental conditions

Bachelor of Science Thesis in Software Engineering and Management

GABRIELE KASPARAVICIUTE

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2016

Systematic Evaluation of Selected Algorithms for Sensor Based Localization for a Mini Autonomous Vehicle
Sensor fusion and localization algorithms' evaluation in a simulation framework under experimental conditions

GABRIELE KASPARAVICIUTE

Examiner: JAN SCHRÖDER

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden June 2016

Systematic Evaluation of Selected Algorithms for Sensor Based Localization for a Mini Autonomous Vehicle*

Gabriele Kasparaviciute[1]

Abstract— **This paper evaluates two different sensor fusion algorithms and their effect on a localization algorithm in the Robot Operating System. It also describes algorithms' strengths and weaknesses. In order to evaluate these algorithms experiment was conducted in three different scenarios for both fusion algorithms, results were collected under the same conditions. The data compares the final robot's position to ground truth.**

I. INTRODUCTION

A. Background

Localization has become an important topic in the field of robotics especially for technologies such as self-driving vehicles which critically rely on accurate and timely location information. There are various approaches and methods to localization, which can be divided into different categories, such as precise localization problems or identifying the environment in which a robot is moving (static or dynamic). Thrun et al [9] differentiate localization problems in their various degrees of difficulty based upon the information available to the robot at the outset:

1) Position tracking. In this situation, a robot knows its initial position and where it is heading. This is also called a local problem.
2) Global localization. In this situation, the robot is placed somewhere on a map but it does not receive any information about its current position from the user. It has to rely on its sensors to evaluate its current position.
3) Kidnapped robot problem. This problem derives from the global localization problem, but is more challenging. In this situation, a robot is placed in a different location from its initial position but is not aware of this change, making its location information a false positive. In global localization problems a robot is at least aware that it does not know its position—but here the robot does not know that it does not know its actual position. The significance of this problem lies in its ability to test an algorithms ability to recover from localization failures.

By being able to solve the global and/or kidnapped localization dilemmas, it is possible for robots to solve complex real life problems. Examples can be seen in the DARPA and Urban competitions, where real autonomous cars face different scenarios and are expected to avoid obstacles [21], [5]. During these contests, autonomous cars use a variety of different sensors, one of the most important (yet expensive) sensors being GPS (Global Positioning System), still considered reliable even though it provides a car's position with an error margin to within a few centimeters at best [6], [15]. Furthermore, GPS is still highly unreliable in urban scenarios [21]. Therefore, it is important to depend on other sensors such as wheel encoders, inertia measurement units (IMUs), and laser scanners. Simply employing previously mentioned sensors is however not enough, as they cannot solve complex localization issues due to high sensor data uncertainty. It is critical to be able to use and analyze the information these sensors provide in order to get the most accurate data about a robot's surroundings [22], [23].

B. Problem Domain & Motivation

It is for the above mentioned reasons a robot needs to be aware of two state estimations regarding a map it is attempting to navigate. The first state estimation is local, describing the robots prompt surroundings, which aids in avoiding obstacles and path planning. The second estimation is global, providing the robot with knowledge of its position within a global coordinate frame. In the local estimation, robots usually adopt sensor fusion, which is a process that makes use of an algorithm to combine data from multiple sensors to provide more accurate data about the current environment. By combining local and global state estimations, it is possible to improve the robustness and accuracy of a robot within a given map.

Previous work has noted a gap in localization research, where robots have usually depended on only either local estimates or fusing data onto a global estimate frame [24]. There are few papers which focus on creating a third coordinate, which fuses sensor data and subsequently provides the data to a global estimate—a robot could adopt this information and calculate its global position onto a coordinate frame, making use of another algorithm [25]. Furthermore, there is a lack of research of local state algorithms effect on the global one in this precise sensor setup in simulation environment. Therefore the objectives of my research are stated as follows:

1) Model the simulation environment to replicate stakeholder's mini vehicle with the chosen algorithms.
2) Model the simulation environment to test algorithms.
3) Analyze algorithms' performance.

*This work was supported by ÅF AB.
Academic supervisor: Christian Berger.

[1]G. Kasparaviciute is with Department of Computer Science and Engineering, University of Gothenburg, Gothenburg, Sweden `guskaspga@student.gu.se`

C. *Research Goal & Research Questions*

This paper compares two different sensor fusion algorithms (Extended Kalman Filter and Unscented Kalman Filter) and their influence on another algorithm—which computes a robot's position estimate (Monte Carlo Localization)—in a simulation performed in an experiment with the support of an industrial stakeholder. The localization algorithm which outputs the end result solves the global localization problem. The inputs chosen for the experiment are described in detail in Section IV. The expected output is a robot's position (x, y and heading angle) compared to ground truth data retrieved from simulation software. Test cases are run under a controlled case study framework. By running simulations with appropriate parameter setup it eases the robot's transition to real life experiments.

The experiment is conducted by running automatic black box software tests, taking into account a high level understanding of the algorithms and the system. Running an experiment in the simulation allows to maintain stakeholder's requirements, evaluate prototypical solution possibilities and confirm hypothesis in a controlled and systematic way which requires large quantity of data for validation.

This research raises the following questions:

1) What algorithms are relevant for an autonomous vehicle to localize itself?
2) Which of the two algorithms provides better localization accuracy?

D. *Contributions*

The aim of this paper is to systematically compare the influence of two different sensor fusion algorithms on a localization algorithm implemented in the Robot Operating System (ROS) software, affecting the positioning and movement of a robot. This algorithm setup has not been tested before in ROS with the proposed sensor arrangement. By running black box software tests on the proposed robot, it increases the link to simulations to as an instrument to clarify requirements, evaluates prototypical solution possibilities, or/and allows to confirm hypothesis in a controlled and systematic way.

E. *Scope*

Idustrial stakeholder offered predefined sensors, chosen to be installed in a 1/10 scale autonomous mini vehicle (robot). It includes:

- two wheel encoders which measure traveled distance;
- an inertia measurement unit which measures angular velocity and acceleration;
- a RPLIDAR laser range scanner measuring distance to objects.

Furthermore, stakeholder determined Robot Operating System (ROS) as a software simulation requirement to conduct the experiment. This was chosen for several reasons: firstly, ROS is the most prominent framework for robot software development, run by an open source community maintaining various libraries and device drivers [27], [26].

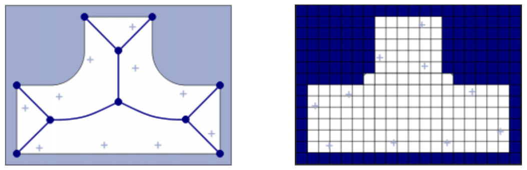

(a) Topological map example (b) Grid based map example

Fig. 1: Two major map representation samples based on [34].

Furthermore, ROS allows researchers to run various simulations and allows for the same setups to be run on hardware (more information in section III). Lastly, ROS was the software package used for stakeholder's final project.

F. *Structure of the Article*

This paper is structured as follows: Section II describes relevant algorithms for both localization and sensor fusion, in addition to common approaches taken to solve localization problems within a given map. It also distinguishes three major map categories used in algorithms as inputs. Section III outlines the background of the software package used for simulation and data gathering. The design of the experiment is detailed in section IV. Data collection and validation from the experiment is reported in Section V while interpretation of this data is provided in Section VI. Finally the findings are summarized in Section VII which describes how the results of the study can be carried into future work.

II. Related work

A. *Environmental representations*

A robot's sensors can identify objects in its environment by criteria such as shape and color. Using a static map as an input for a localization algorithm enables a robot to compare the current data received by its sensors against the previously installed map's data, computing for position and direction of travel. Maps are a crucial input, and therefore detailed research is needed for them. There are three main categories of maps discussed in the literature: topological maps, metric maps, and hybrid maps. This section discusses each, providing their advantages and disadvantages.

1) Topological maps: Topological maps are an abstract of an environment based on connected nodes and arcs. These maps allow for distorting point positions without changing spatial relationships between the nodes. Nodes resemble places, while arcs resemble paths on which a robot can move (see Fig. 1 a). This type of map can be built in different ways, such as building it from grid map information, because it is a low-resolution model focusing on information like open spaces and passageways [2], [1]. Since it only focuses on nodes and arcs, topological maps efficiently help planning processes with data and is resistant to faulty localization and wheel slippage [3], [4]. However, this category of maps also has some disadvantages. For example, these maps are time consuming to build, especially if the environment requiring representation is complex [9]. In addition, a vehicle depending on topological maps uses its sensors to understand its surrounding based on landmarks or distinct sensor readings. This creates a problem if two places are similar yet are reached via different paths.

2) Metric maps: Metric maps represent environments by making divisions of equally spaced grids. Each grid may hold information about an obstacle within. These kinds of maps are rather easy to construct, and succeed in extensive environments as they are represented by two-dimensional grids (see fig. 1b). One of the most popular ways to acquire such a map is by using a laser scanner on a vehicle. Furthermore, recognition of places on metric maps is not ambiguous and is viewpoint independent. Additionally these maps help when calculating the shortest path. To accomplish this, a vehicle's position is computed incrementally using odometric information (data acquired from motion sensors such as wheel encoders) and other sensor readings.

Nonetheless, grid based metric maps have weaknesses too, such as requiring more space on hardware and being less efficient when compared to topological maps [6]. This is due to the fact that a high resolution of grid is needed if all details are to be captured [3].

3) Hybrid maps: There is one more alternative if neither two approaches fit—hybrid maps. As the name states, it is possible to apply two mapping approaches that would compensate each other. For example, Fassbender et al [7], [8] used a metric-topological map for navigation in an outdoor environment, stating that it helped to combine the accuracy of metric maps with the efficiency of topological maps.

After taking into account these three maps' benefits and weaknesses, it was decided to choose a metric grid based map due to its ease of construction and also due to low environment complexity.

B. Algorithms

The two most popular approaches used for sensor fusion are Kalman filter based method while the standard localization algorithm is based on Monte Carlo Localization (MCL) methods, also known as particle filters [10]. Before jumping into the algorithms directly, it is important to clear up definitions for concepts they rely on: Gaussian distribution, covariance matrix, variance, belief, and odometry.

1) Gaussian distribution. Mobile robots receive information about their surroundings from sensors which are prone to error. Mathematically this uncertain data is portrayed by random variables and probability theory. Gaussian distribution (also known as normal distribution, the bell curve shape) represents the possibility for a random variable to fall under a given value (see Fig. 2). Its mean is in the peak of the curve, which distributes all other values equally (50%) around the mean[33].
2) Covariance matrix shows whether variables in a vector correspond to each other in a positive way (if variables x and y increase, that means they tend to covary)[4].
3) Variance is the average distance from mean.

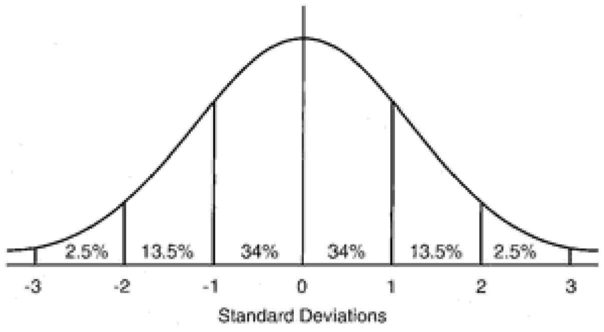


Fig. 2: Normal distribution example.

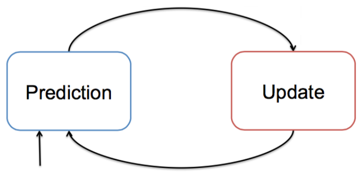


Fig. 3: A high level presentation of Extended Kalman filter manner according to [28].

4) Belief is a robots understanding of the state environment, including its past sensor measurements and controls.
5) Odometry data is data retrieved from a robot's motion. In our case, it will be retrieved from wheel encoders.

1) Extended Kalman Filter: Extended Kalman filter (EKF) has been known for decades and is an improved version of Kalman filter (KF) which can only deal with linear functions [17], [18], [19], [20]. However, in real life situations data is nonlinear. In our case, measurements taken from a laser scanner behave similar to sine or cosine wave forms, rather than a straight line. The Extended Kalman filter (EKF) has been known for decades, and is an improved version of KF that can be applied to nonlinear data. EKF is thus more significant in our case and more applicable than KF. The basic idea of EKF is linearization. EKF takes the mean estimate it received from a previous time stamp and linearizes nonlinear functions around that mean, making it possible to apply the original Kalman filter to update the robot's state[24]. A visualization of how EKF is carried out is shown in figure 3.

The input of EKF algorithm is t (time), μ (mean calculated previously), covariance Σ, u (control) and z (measurement). All these variables represent a belief (see fig. 4). In lines 2 and 3, EKF applies a first order method called Taylor expansion, which constructs a linear function and covariance of the current estimate. These two steps are called prediction steps, which output a predicted belief μ and Σ representing the belief one-time stamp before combining the measurement z. The belief is retrieved by combining the control (u). The mean is then updated using a function that represents the state (g) with an alternative previous mean. In line 3,

```
1:  Extended_Kalman_filter($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$):

2:      $\bar{\mu}_t = g(u_t, \mu_{t-1})$
3:      $\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + R_t$

4:      $K_t = \bar{\Sigma}_t\, H_t^T (H_t\, \bar{\Sigma}_t\, H_t^T + Q_t)^{-1}$
5:      $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
6:      $\Sigma_t = (I - K_t\, H_t)\, \bar{\Sigma}_t$
7:      $return\ \mu_t, \Sigma_t$
```

Fig. 4: The Extended Kalman filter algorithm [9].

covariance is updated by using a linearized state function (now G).

In lines 4 through 6, the estimated belief is altered into a desired belief by involving the measurement zt. The variable Kt in line 4 is called the Kalman gain, which dictates to what degree observations should have effect when compared to the prediction, while also adding process noise covariance (Q) which models uncertainty in the prediction stage of filtering. Line 5 transforms the mean by incorporating the Kalman gain while in line 6 new covariance of belief adjusts its information in regards to the Kalman gain and measurement (h).

To simplify, if a robot is moving around a plane and calculates that the gain is equal to 1, that means that the previous state does not matter and its current state estimation is calculated from observations. While if the gain is equal to 0, it suggests that data received from measurements is insignificant, and original calculations are used relating to its current state.

The Extended Kalman filter handles nonlinearities by using Jacobians (G and H instead of A, B and C matrices in Kalman filter). Jacobians are the first derivatives, which simply say how far the current state estimate is from the best estimate it had before. G and H need to be recomputed in every point in time because the linearization point changes. However, it depends on the degree of a function's nonlinearity. If the linearized functions are approximately linear, then in general EKF's belief is accurate. Otherwise, large uncertainties will lead to increased errors and may contain expensive matrix operations [29]. There have been some attempts to use EKF in vehicle localization, for example, Dantanarayana et al [14] have fused odometry information and range bearing sensor data for their robot, while Hoang et al [30] fused omnidirectional camera and a laser rangefinder using EKF. Others have used infrared sensors and odometry data [31], [32].

2) Unscented Kalman Filter: An Unscented Kalman filter (UKF) is more complex compared to an EKF as it involves more steps (see Fig. 5). However, it similarly includes the prediction and update steps (also known as estimation). The basic idea behind this algorithm is as follows: UKF computes a set of sigma points (where the first one is the mean), then it maps those points through a nonlinear function and uses those points to reconstruct a normal distribution (which are, as mentioned above, all the Kalman filter can handle) [33]. It also assigns weighted points to these sigma points. The difference between EKF and UKF is that UKF refrains from solely linearizing around the mean—in other words it takes into account points away from the mean. Sigma points and weights are free parameters that can be customized for each scenario. There is no unique solution of choosing them, but there are some properties that it must follow. For example, all sigma points should sum to 1 and if the mean is reconstructed from weight points, the result should be the original mean with the same covariance matrix.

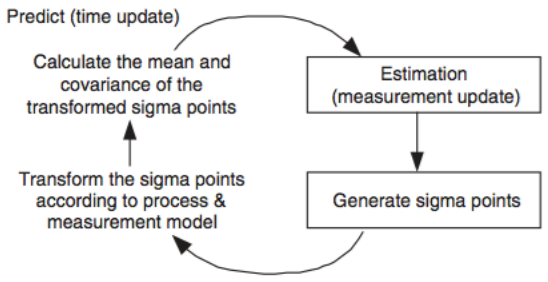

Predict (time update)
Calculate the mean and covariance of the transformed sigma points
Estimation (measurement update)
Transform the sigma points according to process & measurement model
Generate sigma points

Fig. 5: A high level presentation of Unscented Kalman filter manner according to [28].

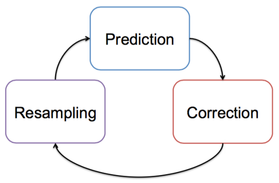
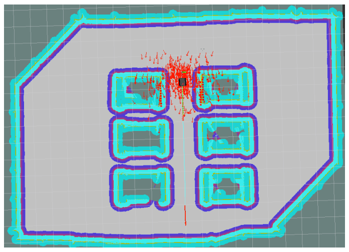


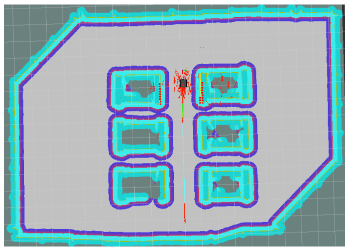
Fig. 6: A high level presentation of Monte Carlo Localization.

3) Monte Carlo Localization Algorithm: In robot localization the most important concept is belief as the robot does not know its position, which cannot be measured directly. This conveys that the robot calculates its position from the given map and sensor data [44]. Position is expressed by a vector which includes two dimensional coordinates x and y and the robot's heading Θ (theta) . The most common algorithm used to determine belief is the Bayes filter algorithm, which simply calculates the probability for a state that a robot could be in. Mathematical derivations of the Bayes filter can be found in [9]. There are many filters within the Bayes family, but when it comes to localization, Monte Carlo Localization (MCL) has been characterized as the gold standard [36], [37], [38], [39].

The concept of MCL is as follows: its input is any arbitrary distribution (not Gaussian) and a map. It distributes particles which hold the state hypothesis (possible position). Each of these samples has a weight indicating the probability of that area (fig. 6). To put it simply, the more samples are clustered in an area, the higher probability a robot is located in the



(a) High uncertainty of robot's position.



(b) Low uncertainty of robot's position.

Fig. 7: Particle filter in action in ROS.

corresponding area. For example, if there are *n* number of samples (particles), that means a system could have *n* number of states. This means that the higher number of samples used in the system, the higher probability of finding the robot's state [33]. However, there is also a drawback to this. Since computational resources mostly are scarce, the number of samples has to be chosen carefully. Compared to the Kalman filter which requires the same computational resources if the area is 10cm large or 1000cm large, MCL's computation resources highly depend on the number of samples[33]. In order for MCL to be able to generate all these particle filters and choose the way to spread them, it depends on the motion model of the robot and observation, which retrieves odometry data. For example, if a robot moves 1cm forward, particles would also move 1cm forward in the map while also adding to the noise around it. This spawns the next generation of samples. MCL's advantage in this case is that it does not require linearization (compared to KF) since it bases sampling on the motion model [45].

An illustration of how a particle filter works is shown in figure 7. In figure 7a, it shows a robot that is unsure of its position. However, the second it starts moving towards its goal, the particle filter propagates all samples forward according to the motion of robot, weighs them and then re-samples, thus reducing the particle swarm (pic. b). This is done by not only depending on the motion model, but also on observations, which are usually received by various environment recognition sensors (laser scanners, ultrasonic sensors). After a few steps the system identifies where a robot must be in line of the observations. The more information the robot can retrieve from observation, the fewer particles are spread around (particles with low weight are discarded). This whole process is also called global localization.

In comparison to EKF, rather than calculating mean and covariance in the prediction step, UKF computes sigma

```
<rosparam param="odom0_config" >
[true, true, false                    x, y, z
false, false, true,                   roll, pitch, yaw
false, false, false                   x, y and z velocities
false, false, true                    roll, pitch and yaw's velocities
false, false, false]                  x, y, and z accelerations
</rosparam >
```

TABLE I: Odometry configuration.

points. Then it uses those points for expected observation and Kalman gain to compute the belief. Furthermore, UKF does not use Jacobians, thus it has a better approximation for nonlinear models. UKF has been applied in different scenarios. For example, Kim et al [35] implemented UKF to perform sensor fusion for an automatic guided vehicle with a laser scanner, proximity sensors, encoders, and a gyroscope installed. Zhang et al and Li et al [28] presented an example where UKF was used for GPS and IMU sensor fusion. MCL has the advantage of being able to maintain many states while KF can only maintain one. MCL works well in low dimensional space, for example 2D, because it does not require as many particles. It is also mostly used in localization required situations where robot does not have predefined landmarks. There have been various applications of MCL in research [22], [40], [41]. [42], [43] presented results of a vehicle using a laser scanner and odometry data while trying to localize themselves globally.

It is worth mentioning that nowadays due to better and cheaper hardware, the localization problem is attempted to be solved using only vision [12], [13], [11]. The extra sensors used in our project's robot would signify additional costs compared to a robot with only one sensor, however a more robust system machine is obtained.

III. Background on Simulation

Robot Operating System (ROS) is an open source robot software framework. The key ideas behind its application are nodes, messages, and topics. Nodes represent software modules that interact with each other in a peer-to-peer way. These nodes correspond to each other by broadcasting messages. A message holds information that describes a data structure (boolean, floating point, integer, and so on). Nodes communicate messages by publishing them to a chosen topic (for example, a map). If a node requires data it subscribes to a topic, which means there can be many nodes connected to one topic and vice versa. The most simple communication method is a pipeline (see fig. 9). The robot used in this paper is a derivation of a Clearpath Robotics Jackal[1]. Our project required to remove some sensors from the stock unit (such as GPS) and adjust its kinematics (most robots use differential drives with actuators for each wheel).

In the figure 8 the main graph is odom (which corresponds to odometry). The second graph (*base_link*) subscribes to odom, which then publishes data to *chassis_link*. The latter publishes data to other sensor links—for example, all

[1]http://www.clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/

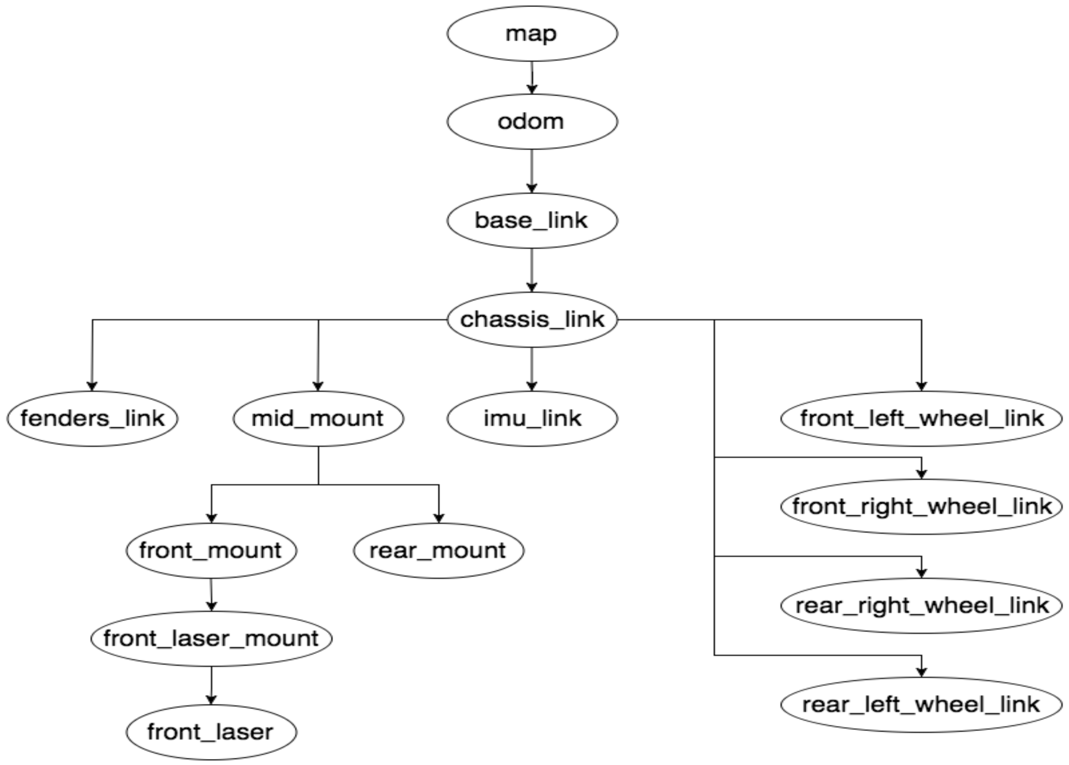


Fig. 8: Robot's coordinate frames in Robot Operating System.

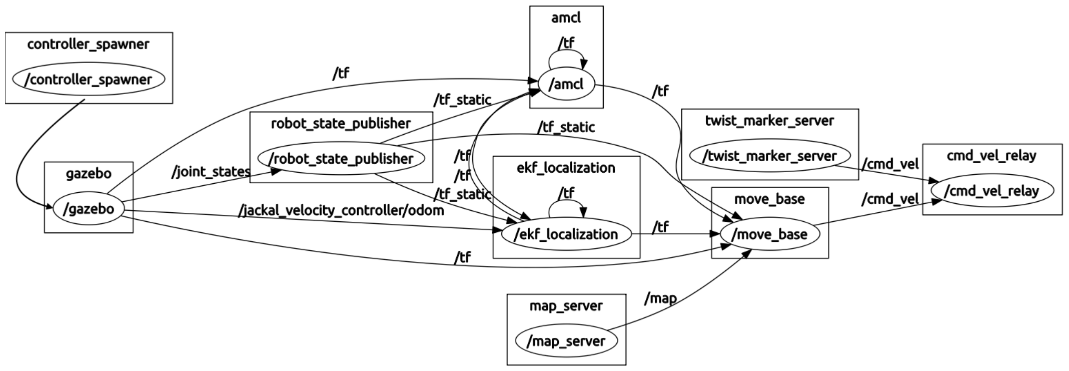


Fig. 9: Visualization of robot's computational graph in Robot Operating System.

separate wheels, laser scanner link, and so on. Some of these graphs describe the physical equipment needed to hold a sensor, for example (*mid_mount*) which can play an important role in case of collisions. As it is clear, there are many nodes sending these images (see figure 9). In order for ROS to be able to keep up with the information about the robot's states, there are some predefined nodes keeping track of it.

In figure 9 the node */robot_state_ publisher* gathers information about the robot's state and publishes it to whomever subscribes. It gathers joint angle data of the robot and publishes 3D positions of those links based on the robot's kinematics.

/tf is a package that maintains information about multiple coordinate frames (the tree structure seen in figure 8).

/map_server loads and provides a map.

/jackal_velocity_controller/odom collects odometry data from Gazebo [2], a realistic robot simulator.

/controller_spawner, */twist_marker_server*, */cmd_vel_relay* provides data required to mimic robot controllers.

*/*kf_localization* contributes to sensor fusion. In this paper's case it is IMU and two wheel encoders.

/amcl supplies Monte Carlo Localization approach and tracks the robot's pose.

Even though most of these setups are common for any robot, the key items requiring adjustment for the purpose of our study were:

1) Make the robot as similar to the stakeholder's requests as possible; making sure it has the same sensor setup with similar specifications (for example, adapting ROS

[2]http://gazebosim.org/

laser scanner to simulate physical RPLIDAR laser ranger)

2) Navigate the robot and constrain its kinematics in such way that it would resemble an automobile.
3) Map creation.

The first item was resolved by adjusting such ROS parameters as odometry configuration. This configuration defines which variables will be fused to the final estimate. Refer to table I to see how the odometry configuration looks like in our case. To the right are explanations of boolean values. In this example, *x* and *y* positions, yaw, *x* and yaw velocities are fused to the final estimate. IMU data looks similar.

The second item was addressed by applying a time elastic band (TEB) path planner, which focuses on navigation of robot with respect to avoiding contact with obstacles, planning the shortest path to a goal [16]. In our case it was also allowed to limit the wheel's turning radius, which made it similar to car-like robots.

The third item was resolved by creating a map in the robot simulation (Gazebo) and commanding the robot to roam around it while using Simultaneous Localization and Mapping algorithm, which output a 2D grid map.

IV. METHODOLOGY

In this section experimental setup is described. After an extensive literature review, the relevance of the three algorithms was confirmed and therefore all of them were used in the study. Various Kalman filters approaches' advantages are outlined in a detailed different localization techniques comparison study in [50]. Other researchers have shown agreement in [14], [24], [28], [29]. Similar situation has been noted with another algorithm; Monte Carlo Localization has gained its popularity after the DARPA Urban challenge and it has proven its benefits [21], [5].

To be able to evaluate the Extended Kalman Filters and the Unscented Kalman Filters fusion effect on the localization algorithm, it was necessary to set up a simulation environment (see Section III for more details).

The mini-vehicle used in this experiment had a differential drive with a limited turning radius of 30 degrees with two wheel encoders mounted on its back wheels and an inertia measurement unit (IMU). Laser scanner was placed on the top of robot.

The robot was tested with two different maps (see fig. 10). Each map's area was around 900m^2. The reason these maps and the following scenarios were chosen came about after observing a pattern in previous papers [24], [48], [47], [32], [51], [49]. Most papers state that their experiments started with a simple straight line with an obstacle sitting next to a robot and introduced turning motions to the robot. Later on difficulty was increased by introducing robot to various trajectory shapes. The first map tests a simple scenario where a robot simply drives straight between obstacles (see fig. 10 a). Similar tests were produced in [24], [32]. Additionally, another test course was run within this map simulating a city block. This included making the robot drive around a rectangle shape, as was suggested by [47]. The second map included some building blocks in it, forcing the vehicle to drive on a prepared figure-of-eight trajectory, creating a scenario putting its wheel encoders to a challenging task [46].

Controlled Variable	Clarification
Sensors	In ROS simulation it is possible to adjust sensor noise. Noise was added to laser scanner measurements with a mean of 0.0 and standard deviation of 0.01. According to specifications, 99% of samples fell within 30mm of actual measurements. Odometry noise was left as default.
Final state estimate	It specifies which variables are used for sensor fusion. See table I.
Particle number	Particle number was set between 500 and 2000. This number has been left by default from the previous model. As the map is not large it should be acceptable.
Process noise covariance and estimate covariance	Also denoted as Q and P respectively in the Kalman filter, have been left as default.

TABLE II: Controlled variables of the experiment.

Dependent Variable	Clarification
Robot's estimated pose	*x*, *y* coordinates followed with heading angle.
Position variance matrix	*x*, *y* and trajectory deviation from mean.

TABLE III: Dependent variables of the experiment.

Sensors were fused by selecting their appropriate output values (see tables I and V). Wheel encoders and IMU supplied acceleration, angular rate, and velocity data. In the odometry matrix, *x*, *y*, *yaw* and *yaw velocity* values have been designated for sensor fusion with IMU's *yaw*, *yaw velocity*, *x acceleration* and *y acceleration*. Filters receive duplicate data of *yaw* and *yaw velocity* in order for the robot's trajectory to be calculated in the most accurate way. As we are operating in a 2D coordinate frame, all Z axis values have been set to false. The second wheel encoder's data is fused into the filter by providing a parameter to ROS taking care of its matrix.

Both IMU and wheel encoders have a set Gaussian noise, with IMU providing data at a rate of 50 Hz. Measurement data can be extracted from wheel encoders at a rate of 20 Hz. These numbers have been chosen by determining an average based on studying previous experiments [24], [49]. The robot on average drove each instance for approximately 15m.

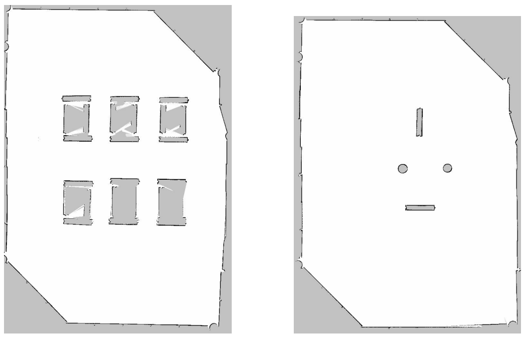

(a) Urban map. (b) figure-of-eight shaped map.

Fig. 10: Maps used in simulation.

Independent Variable	Clarification
*kf_node_localization	State estimation node in ROS.

TABLE IV: Independent variables of the experiment.

<rosparam param="imu0_data" >	
[false, false, false	x, y, z
false, false, true,	roll, pitch, yaw
false, false, false	x, y and z velocities
false, false, true	roll, pitch and yaw's velocities
true, true, false]	x, y, and z accelerations
</rosparam >	

TABLE V: IMU data final estimate matrix.

The experiment was conducted as a controlled case study, which determined the variables seen in tables II, III and IV. Each of the three scenarios (straight line, rectangle, and figure-of-eight) were repeated five times and their average performance was taken into final calculations.

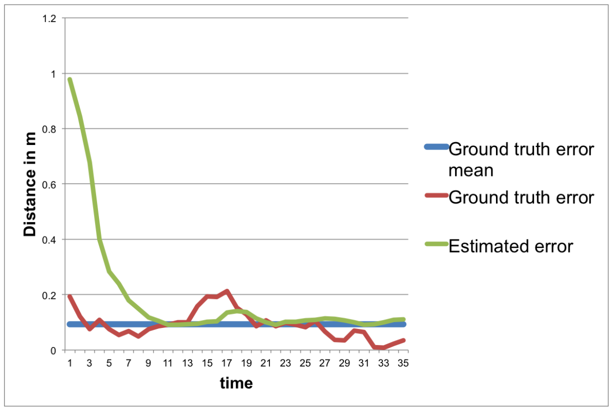
Mini-vehicle was controlled by setting points on the map and letting it navigate through the map until it reached each point.

V. RESULTS

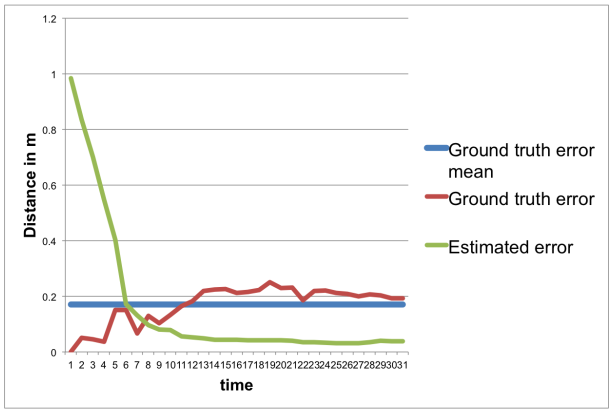
Results for the Extended Kalman filter and Unscented Kalman Filter's effect on the Monte Carlo Localization algorithm are discussed in this section. Data was recorded while running each scenario and then extracted. Trajectory orientation was output as quaternions, an alternative way to represent spacial rotation, though it was decided to compare trajectory in Euler angles, which are more familiar to most researchers. The final results for each scenario are expressed as the mini-vehicle's position (x, y and *heading angle in degrees*) in ground truth data from the Gazebo simulation and the position as estimated by algorithms. Ground truth provides the true, also known as the correct robot's position data, to evaluate robot's estimated position. The focus of this paper is to evaluate which setup of algorithms is best, determining this using the following strategy:

- Ground truth root-mean-square (RMS) estimate error, which measures the difference between ground truth and the robots given estimate. The lower this value is, the more accurate a position is provided by the algorithm.
- Estimation errors, calculated by looking at the covariance magnitude as a function per time. This reveals the robot's calculated uncertainty over time.
- Mean of ground truth RMS.

For the straight line scenario, it is observed that the robot using EKF started the experiment with a high uncertainty for all values in position vector; all estimated error variables are approximately 5 times higher than the ground truth estimated error (see fig. 11a). The average ground truth error is approximately 0.5 m. We can see that for the x coordinate ground estimated error follows ground truth error coarsely, showing that robot underestimated its position regarding x. However, when the robot overestimated its y position; the ground truth error mean was high (50 cm) and accumulated over time even though the robot's uncertainty was rather



(a) x plane results using EKF.



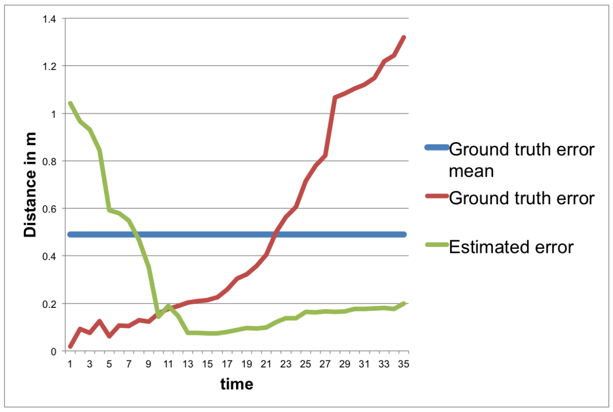
(b) x plane results using UKF.

Fig. 11: Ground truth error, mean, and estimated error for x plane in a straight line scenario.

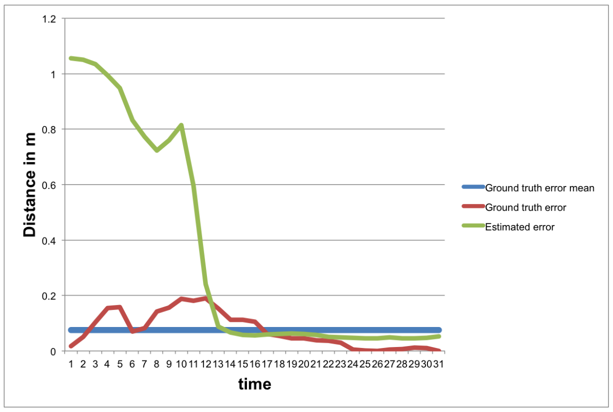
low (see fig. 12a). Trajectory error is similarly low, usually staying below the ground truth mean (see fig. 13a).

Testing for UKF had similar results. The robot once again started with high uncertainty of its position even though ground truth error halts around the average mean error. To be more specific, x results show that the average error between ground truth and the estimated position was less than 20 cm, with the robot rather certain about its x position (see fig. 11b). It took a longer time for the robot to reduce its estimated error position y. Nonetheless, the ground truth error mean was only around 8 cm, showing little difference between ground truth and estimated position of y (see fig. 12b). Robot trajectory was initially erroneous, with the mean of ground truth RMS less than 0.5 degrees. However, the spikes in ground truth error show that trajectory was not estimated well (see fig. 13b).

When running the robot through the figure-of-eight course, results from the implemented EKF algorithm showed many spikes for x and y coordinates. Every spike respectively begins with a turn around an obstacle. We can see that x estimation starts off with a some uncertainty (around 40 cm), however it corrects itself with each time step before a turn. Estimated error follows the ground truth RMS error coarsely, meaning the robot was aware of its uncertainty (see

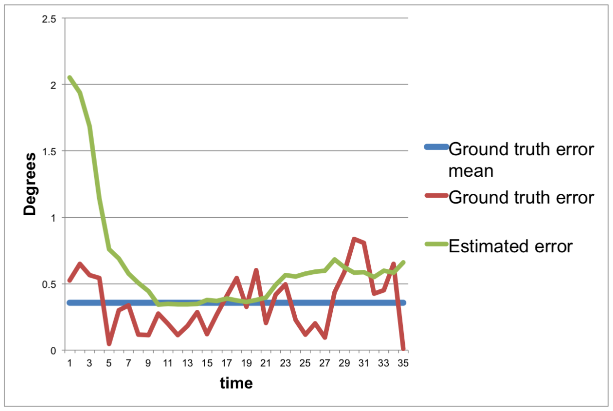


(a) y plane results using EKF.

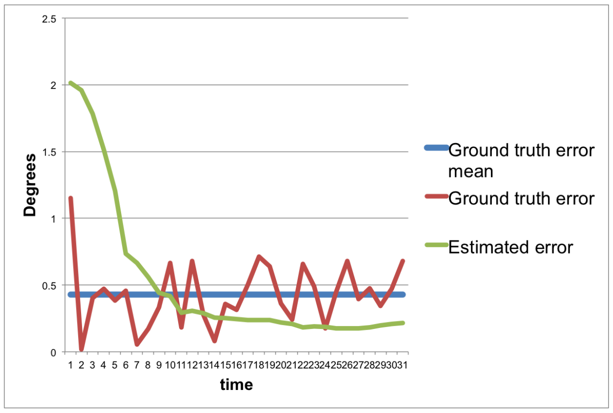


(b) y plane results using UKF.

Fig. 12: Ground truth error, mean, and estimated error for y plane in a straight line scenario.



(a) Heading results using EKF.



(b) Heading results using UKF.

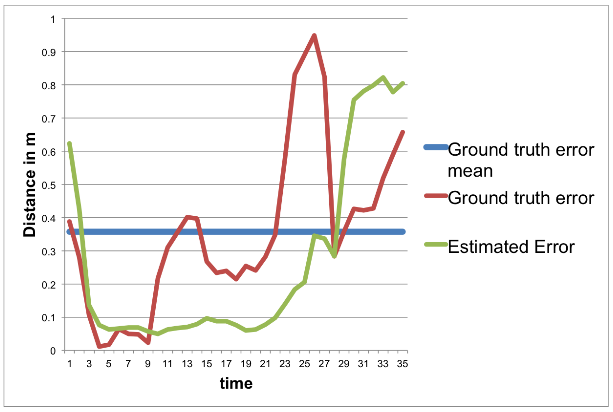
Fig. 13: Ground truth error, mean, and estimated error for heading in a straight line scenario.

fig. 14a). The robot overestimated its position regarding y, with many spikes measured even though it mostly stays under the ground truth RMS mean. Furthermore, the estimated error did not follow the ground truth RMS, which means the robot was not fully aware of its y position (see fig. 18a). The situation is better in the trajectory results: ground truth error was below the mean and estimated error followed roughly the ground truth. It is clear that in at the finish the robot is absolutely lost in regards to its trajectory since the error value jumped to nearly 160 degrees (see fig. 15a).

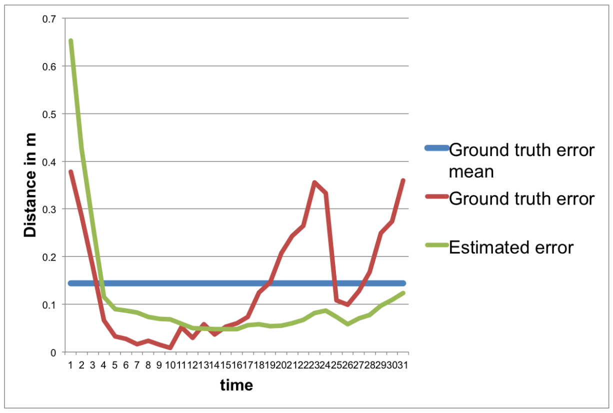
UKF showed more promising results in this scenario; spikes are scarce. The robot started with a high x plane uncertainty, which saw a twofold reduction by the middle of this scenario. The average ground truth error was low (15 cm). However, the ground truth error peaked twice at the end of this experiment. Since the robot's estimated error follows the ground truth error, this reveals that the robot is aware of its uncertainty (see fig. 14b). y plane showed similar results as x (see fig. 18b). Trajectory output showed that the robot had little uncertainty and error when compared to ground truth—most of the time below average except for at the end of experiment, when it peaked to over 20 degrees (see fig. 15b).

The final scenario challenged the robot to navigate a rectangle shaped obstacle. For the robot employing EKF, the x plane stays under the 20 cm average. The robot's error accumulated even though the actual difference is rather low (see fig. 16a). Similar results are displayed for the y plane. However, in this case the ground truth and estimated error increased by the end of the experiment (see fig. 17a). Spikes in trajectory reveal that the robot's course varied greatly. Similar curves are not exhibited in estimated error, which accumulated over time (see fig. 19a).

The robot using UKF shows similar spikes in all position variables. x plane results present that the robot's estimation is relatively close to the ground truth. Only in the end of the experiment does the robot miscalculate its estimation even though it is in an approximate position (see fig. 16b). The robot begins the experiment with high uncertainty regarding the y plane, but is very accurate throughout most of the scenario: the ground truth RMS stays below the mean (see fig. 17b). The robot's heading displays a high uncertainty, especially regarding the estimated error, which does not follow the ground truth error (see fig. 19b).

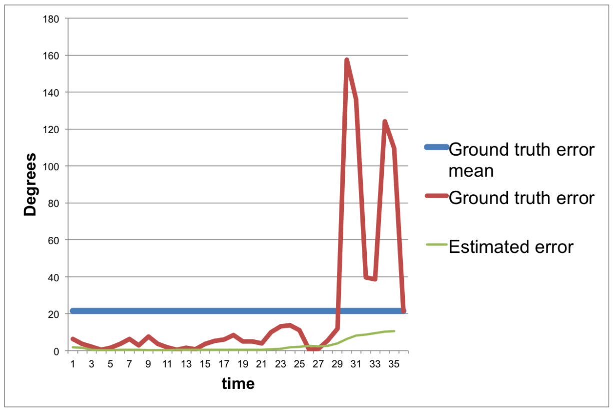


(a) *x* plane results using EKF.

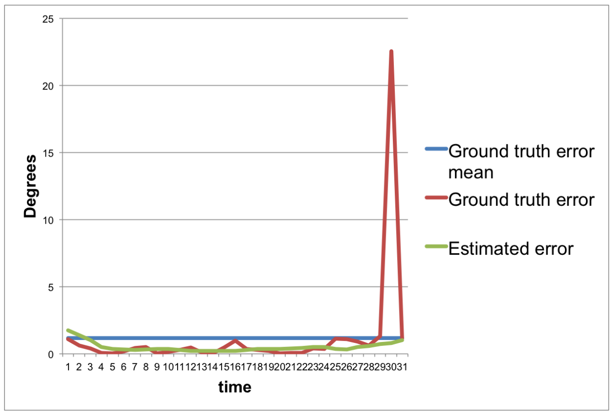


(b) *x* plane results using UKF.

Fig. 14: Ground truth error, mean, and estimated error for *x* plane in figure-of-eight scenario.



(a) Heading results using EKF.



(b) Heading results using UKF.

Fig. 15: Ground truth error, mean, and estimated error for heading in figure-of-eight scenario.

VI. ANALYSIS AND DISCUSSION

A. Relevant localization algorithms

Robot localization can be achieved by applying various approaches. One of the most popular approaches includes exclusively implementing Kalman filter based algorithms [46], [31], [32]. Most popular reasons are ease of implementation and reduced computation burden. Another approach involves only the use of particle filters [45], [39]. All three algorithms have gained their acclaim in the research field [50], [5]. The reason for choosing last option is based on low computational resources [3], [6].

B. Algorithms evaluation

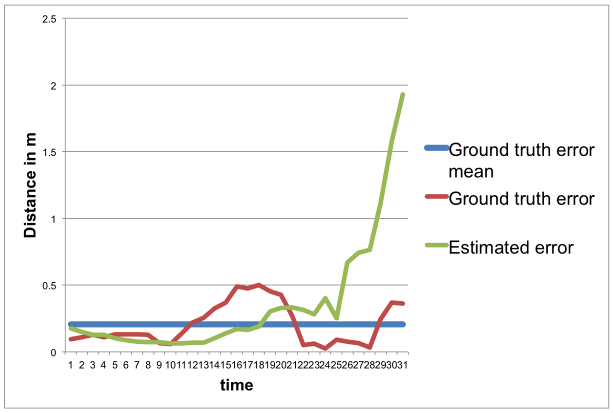
The Unscented Kalman filter showed its robustness in all the scenarios, mostly evident by a low (sometimes even by double) ground truth error mean value when compared to the Extended Kalman filter. Additionally, results show that the ground truth error using EKF was spiking much more often and to higher values. One possibility for this is that there is a severe non-linearity, which is difficult for the EKF to handle. Similar results have been shown in [15], [23], [45], [46]. Researchers reported that most of the time UKF performed better, except for when the robot's trajectory was set to a circle. In that case robot performed with similar results [35].

To increase the accuracy of the algorithms, we would have to adjust some of the initial controlled variables set in the experiments methodology. For example, measurement and process noise accumulate rapidly, and thus should without question be taken into account. Even though the experiment included three different scenarios, it does not prove conclusively that the UKF performs better in all of them.

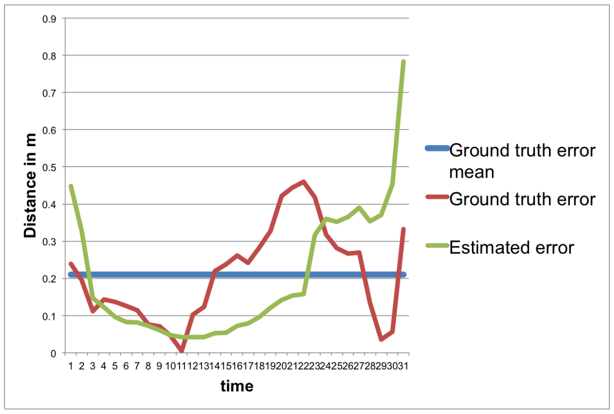
Since exact experimental setup has not been tested previously, it can not be fully compared to previously published research papers. This may be due to the same scarce computation resources reason in practice since the application of two algorithms may be too cost intensive [52], [45].

C. Threats to Validity

One major limitation of this study was that the setup was not optimal. Even though the requested robot model was expected to use car-like kinematics, it was only possible to adapt a differential drives mechanism. Additionally, the chosen map was too large for the robot, given its sensor arrangement. The laser scanner was set to replicate RPLIDAR, which measures distance up to 6 meters. This generated difficulties for the robot to detect obstacles, which in turn

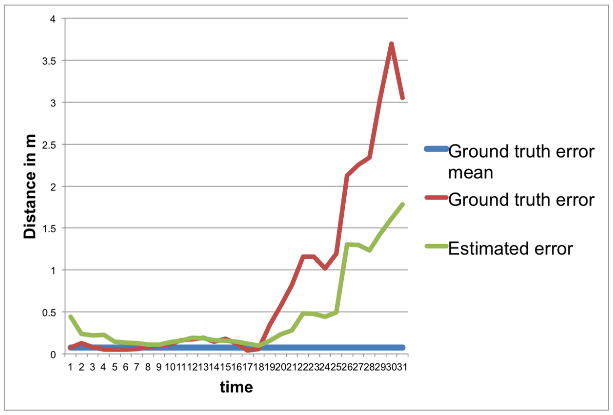


(a) x plane results using EKF.

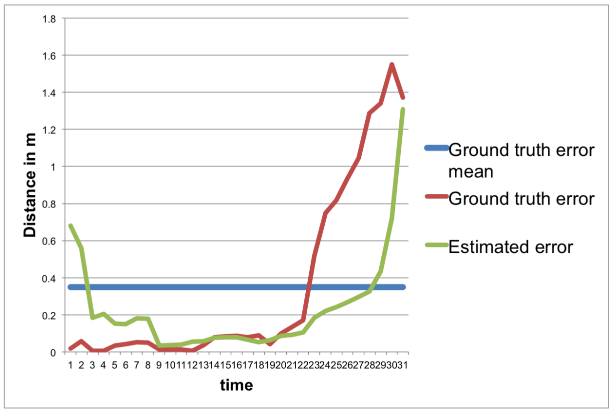


(b) x plane results using UKF.

Fig. 16: Ground truth error, mean, and estimated error for x plane in rectangle scenario.



(a) y plane results using EKF.



(b) y plane results using UKF.

Fig. 17: Ground truth error, mean, and estimated error for y plane in rectangle scenario.

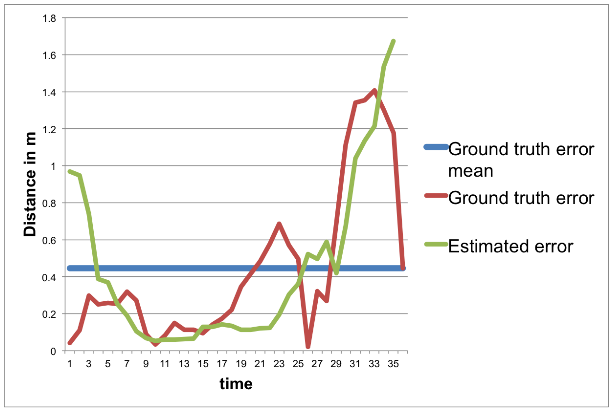
hindered the robots ability to locate itself more accurately. Furthermore, instead of using teleoperation in the simulation, the robot navigated using manually-assigned points. This could have generated some errors resulting in the robot not completely following the same path for each scenario.

Simulation aids in developing a successful model and enables testing various real life scenarios. However, it is important to bear in mind some points. First of all, in this paper's setup robot's tires to ground friction has not been taken into account. It has been asserted that it influences robot's final position [24]. Furthermore, before conducting real life tests, it is substantial to consider sensors calibration which would decrease their noise. Moreover, sensor's rate also influences sensor fusion algorithm's output. In the simulation, the rate remained unchanged. However, there is a possibility that in reality one sensor may have a lower rate than the other or it may be affected by some external factors which would decrease the end performance. This could generate another source of discrepancy between simulation and actual performance in real life. Research could also be conducted in a different context by implementing sensor fusion with GPS and comparing results to the current study. Along with a diverse sensor setup, this experiment can be also be conducted applying a different method, for example, limited lab experiment on an actual physical mini vehicle. This would provide evidence if the hypothesis (in this case if UKF outperforms EKF) is true.

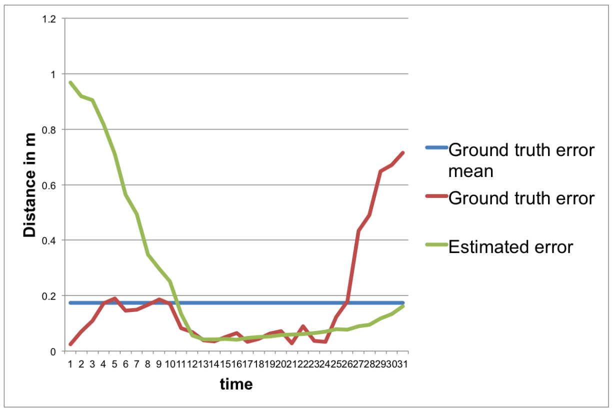
VII. Conclusion and Future Work

In this paper the effect of two sensor fusion algorithms (Extended Kalman filter and Unscented Kalman filter) were tested in regards to the Monte Carlo Localization algorithm through a simulation conducting an experiment. Algorithms were analyzed in three different scenarios. Each of these algorithm's advantages and disadvantages have been discussed, following with an explanation of simulation environment setup. This exact setup has not been tested before. Furthermore, different map categories used in robot localization were described with their advantages and disadvantages. The results state that the setup is possible and it shows that UKF in all scenarios performed better than EKF.

In future work, simulation could adapt a real world environment by installing a sample map from Open Street Map. This could lead to more real life test cases of an autonomous mini vehicle. Furthermore, a different algorithm setup (skipping the sensor fusion step and only using a laser scanner) would be interesting to test in the same environment as this experiment and compare results on a physical mini

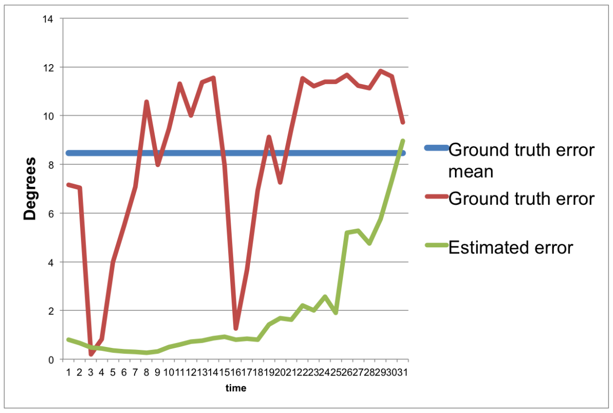


(a) y plane results using EKF.

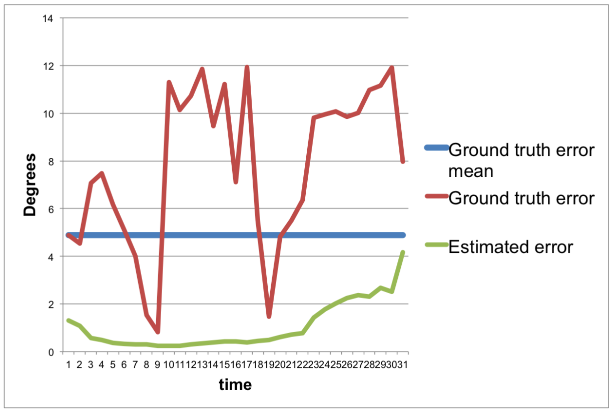


(b) y plane results using UKF.

Fig. 18: Ground truth error, mean, and estimated error for y plane in figure-of-eight scenario.



(a) Heading results using EKF.



(b) Heading results using UKF.

Fig. 19: Ground truth error, mean, and estimated error for heading in rectangle scenario.

vehicle including computation resources. Even though one of the reasons for sensor fusion is to get rid of dependency from only using one sensor.

REFERENCES

[1] Zwynsvoorde, V., Simon, D., & Alami, R. (2000). Incremental topological modeling using local Voronoi-like graphs. In Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on (Vol. 2, pp. 897-902). IEEE.

[2] Fabrizi, E., & Saffiotti, A. (2000). Extracting topology-based maps from gridmaps. In Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on (Vol. 3, pp. 2972-2978). IEEE.

[3] Thrun, S. (1998). Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence, 99(1), 21-71.

[4] Thrun, S., & Bcken, A. (1996, August). Integrating grid-based and topological maps for mobile robot navigation. In Proceedings of the National Conference on Artificial Intelligence (pp. 944-951).

[5] Levinson, J., Montemerlo, M., & Thrun, S. (2007, June). Map-Based Precision Vehicle Localization in Urban Environments. In Robotics: Science and Systems(Vol. 4, p. 1).

[6] Levinson, J., & Thrun, S. (2010, May). Robust vehicle localization in urban environments using probabilistic maps. In Robotics and Automation (ICRA), 2010 IEEE International Conference on (pp. 4372-4378). IEEE.

[7] Fassbender, D., Kusenbach, M., & Wuensche, H. J. (2015, September). Landmark-based navigation in large-scale outdoor environments. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on (pp. 4445-4450). IEEE.

[8] Drouilly, R., Rives, P., & Morisset, B. (2015, September). Hybrid metric-topological-semantic mapping in dynamic environments. In Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on (pp. 5109-5114). IEEE.

[9] Thrun, S., Burgard, W., & Fox, D. (2005). Probabilistic robotics. Cambridge, Mass: MIT Press.

[10] Kristensen, S., & Jensfelt, P. (2003, October). An experimental comparison of localisation methods, the mhl sessions. In Intelligent Robots and Systems, 2003.(IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on(Vol. 1, pp. 992-997). IEEE.

[11] Lategahn, H., & Stiller, C. (2014). Vision-only localization. Intelligent Transportation Systems, IEEE Transactions on, 15(3), 1246-1257.

[12] Nagai, I., Watanabe, K. ”Path tracking by a mobile robot equipped with only a downward facing camera”, Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on,On page(s): 6053 - 6058

[13] Jo, K., Jo, Y., Suhr, J. K., Jung, H. G., & Sunwoo, M. (2015). Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras. Intelligent Transportation Systems, IEEE Transactions on, 16(6), 3377-3392.

[14] Dantanarayana, L., Dissanayake, G., Ranasinghe, R., & Furukawa, T. (2015, December). An extended Kalman filter for localisation in occupancy grid maps. In 2015 IEEE 10th International Conference on Industrial and Information Systems (ICIIS) (pp. 419-424). IEEE.

[15] Zhang, H., Chen, J. C., & Zhang, K. (2014, April). RFID-based localization system for mobile robot with Markov Chain Monte Carlo. In American Society for Engineering Education (ASEE Zone 1), 2014 Zone 1 Conference of the (pp. 1-6). IEEE.

[16] Rösmann C., Feiten W., Wosch T., Hoffmann F. and Bertram. T.: Trajectory modification considering dynamic constraints of autonomous robots. Proc. 7th German Conference on Robotics, Germany, Munich, 2012, pp 7479.

[17] G.L. Smith, S.F. Schmidt and L.A. McGee, Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle, 1962.

[18] R.E. Kalman, A New Approach to Linear Filtering and Prediction Problems, Transactions of the ASME pp. 3545, 1960.

[19] G. Welch and G. Bishop, An introduction to the Kalman filter, 1995

[20] Moore, T., & Stouch, D. (2016). A Generalized Extended Kalman Filter Implementation for the Robot Operating System. In Intelligent Autonomous Systems 13 (pp. 335-348). Springer International Publishing.

[21] Li, H., Nashashibi, F., & Toulminet, G. (2010, September). Localization for intelligent vehicle by fusing mono-camera, low-cost gps and map data. In Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on (pp. 1657-1662). IEEE.

[22] Floros, G., van der Zander, B., & Leibe, B. (2013, May). OpenStreetSLAM: Global vehicle localization using OpenStreetMaps. In Robotics and Automation (ICRA), 2013 IEEE International Conference on (pp. 1054-1059). IEEE.

[23] Prez-Higueras, N., Ramn-Vigo, R., Caballero, F., & Merino, L. (2014, September). Robot local navigation with learned social cost functions. In Informatics in Control, Automation and Robotics (ICINCO), 2014 11th International Conference on (Vol. 2, pp. 618-625). IEEE. Chicago

[24] Anjum, M. L., Park, J., Hwang, W., Kwon, H. I., Kim, J. H., Lee, C., ... & Cho, D. I. D. (2010, October). Sensor data fusion using unscented kalman filter for accurate localization of mobile robots. In Control Automation and Systems (ICCAS), 2010 International Conference on (pp. 947-952). IEEE.

[25] Marin, L., Valles, M., Soriano, A., Valera, A., & Albertos, P. (2013). Multi sensor fusion framework for indoor-outdoor localization of limited resource mobile robots. Sensors, 13(10), 14133-14160.

[26] Machado Santos, J., Portugal, D., & Rocha, R. P. (2013, October). An evaluation of 2D SLAM techniques available in robot operating system. In Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on (pp. 1-6). IEEE.

[27] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

[28] Zhang, P., Gu, J., Milios, E. E., & Huynh, P. (2005). Navigation with IMU/GPS/digital compass with unscented Kalman filter. In Mechatronics and Automation, 2005 IEEE International Conference (Vol. 3, pp. 1497-1502). IEEE.

[29] Van Hinsbergen, C. P., Schreiter, T., Zuurbier, F. S., Van Lint, J. W. C., & Van Zuylen, H. J. (2012). Localized extended kalman filter for scalable real-time traffic state estimation. Intelligent Transportation Systems, IEEE Transactions on, 13(1), 385-394.

[30] Hoang, V. D., Le, M. H., Hernandez, D. C., & Jo, K. H. (2013, November). Localization estimation based on Extended Kalman filter using multiple sensors. In Industrial Electronics Society, IECON 2013-39th Annual Conference of the IEEE (pp. 5498-5503). IEEE.

[31] Faisal, M., Hedjar, R., Alsulaiman, M., Al-Mutabe, K., & Mathkour, H. (2014, November). Robot localization using extended kalman filter with infrared sensor. In Computer Systems and Applications (AICCSA), 2014 IEEE/ACS 11th International Conference on (pp. 356-360). IEEE.

[32] Pinto, M., Moreira, A. P., & Matos, A. (2012). Localization of mobile robots using an extended Kalman filter in a LEGO NXT. Education, IEEE Transactions on, 55(1), 135-144.

[33] Stachniss, C. (2014) Robot Mapping [Power point slides]. Retrieved from http://ais.informatik.uni-freiburg.de/teaching/ws13/mapping/index_en.php

[34] Choset, H., (2010) Robotic Motion Planning [Power point slides]. Retrieved from http://www.cs.cmu.edu/ motionplanning/

[35] Kim, J., Jung, K., Kim, J., Song, H., & Kim, S. (2013). Positioning accuracy improvement of laser navigation using unscented Kalman filter. In Intelligent Autonomous Systems 12 (pp. 807-816). Springer Berlin Heidelberg.

[36] Klaas, M., De Freitas, N., & Doucet, A. (2012). Toward practical N2 Monte Carlo: the marginal particle filter. arXiv preprint arXiv:1207.1396.

[37] Moradkhani, H., DeChant, C. M., & Sorooshian, S. (2012). Evolution of ensemble data assimilation for uncertainty quantification using the particle filterMarkov chain Monte Carlo method. Water Resources Research, 48(12).

[38] Xie, H., Gu, T., Tao, X., Ye, H., & Lu, J. (2015). (In Press) A reliability-augmented particle filter for magnetic fingerprinting based indoor localization on smartphone. IEEE Transactions on Mobile Computing, 13(9), 1-14.

[39] Yatim, N. M., & Buniyamin, N. (2015). Particle filter in Simultaneous Localization and Mapping (SLAM) Using a Differential Drive Mobile Robot. Jurnal Teknologi, 77(20). Chicago

[40] Schindler, A. (2013, June). Vehicle self-localization with high-precision digital maps. In Intelligent Vehicles Symposium (IV), 2013 IEEE (pp. 141-146). IEEE.

[41] Jo, K., Jo, Y., Suhr, J. K., Jung, H. G., & Sunwoo, M. (2015). Precise Localization of an Autonomous Car Based on Probabilistic Noise Models of Road Surface Marker Features Using Multiple Cameras. Intelligent Transportation Systems, IEEE Transactions on, 16(6), 3377-3392.

[42] Chong, Z. J., Qin, B., Bandyopadhyay, T., Ang, M. H., Frazzoli, E., & Rus, D. (2013, May). Synthetic 2D LIDAR for precise vehicle localization in 3D urban environment. In Robotics and Automation (ICRA), 2013 IEEE International Conference on (pp. 1554-1559). IEEE.

[43] Ruchti, P., Steder, B., Ruhnke, M., & Burgard, W. (2015, May). Localization on OpenStreetMap data using a 3D laser scanner. In Robotics and Automation (ICRA), 2015 IEEE International Conference on (pp. 5260-5265). IEEE.

[44] Hiemstra, P., Nederveen, A., (2007). Monte Carlo Localization.

[45] Dellaert, F., Fox, D., Burgard, W., & Thrun, S. (1999). Monte carlo localization for mobile robots. In Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on (Vol. 2, pp. 1322-1328). IEEE.

[46] Myung, Hyun, et al. ”Mobile robot localization using a gyroscope and constrained Kalman filter.” SICE-ICASE, 2006. International Joint Conference. IEEE, 2006.

[47] Hwang, W., Park, J., Kwon, H. I., Anjum, M. L., Kim, J. H., Lee, C., ... & Dan Cho, D. I. (2010, October). Vision tracking system for mobile robots using two Kalman filters and a slip detector. In Control Automation and Systems (ICCAS), 2010 International Conference on (pp. 2041-2046). IEEE.

[48] Houshangi, N., & Azizi, F. (2006, July). Mobile robot position determination using data integration of odometry and gyroscope. In Automation Congress, 2006. WAC’06. World (pp. 1-8). IEEE.

[49] Biswas, J., & Veloso, M. (2014, May). Episodic non-markov localization: Reasoning about short-term and long-term features. In Robotics and Automation (ICRA), 2014 IEEE International Conference on (pp. 3969-3974). IEEE.

[50] Gonzalez, R., Rodrguez, F., Guzmn, J. L., & Berenguel, M. (2009, March). Comparative study of localization techniques for mobile robots based on indirect kalman filter. In Proceedings of IFR Int. Symposium on Robotics (pp. 253-258).

[51] Merriaux, P., Dupuis, Y., Vasseur, P., & Savatier, X. (2014, October). Wheel odometry-based car localization and tracking on vectorial map. In Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on (pp. 1890-1891). IEEE.

[52] Rigatos, G. G. (2010). Extended Kalman and Particle Filtering for sensor fusion in motion control of mobile robots. Mathematics and computers in simulation, 81(3), 590-607.