Estimating Robot Localization Error Using Visual Marker Pose Estimation

by

Sean Scheideman

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

This thesis proposes a method to estimate robot localization error without having a ground-truth measurement of robot position. Robot localization refers to estimating a robot position and orientation (pose) within a known map, where the error is the difference between the robot's ground-truth pose and the algorithms estimated pose. Ground-truth measurement systems (eg. motion capture) while accurate are expensive and tend to be difficult to set up in new environments. A new landmark-based method which uses visual markers placed throughout the environment is proposed as an alternative to ground-truth systems.

The method requires visiting a visual marker twice, collecting the localization pose and robot-to-marker pose on both visits. After enough samples are collected the localization error is calculated using generative latent optimization (GLO). Experiments are run using the proposed method to estimate the localization error for several different open source algorithms. The method is accurate within an order of magnitude of ground-truth established using a motion capture system, inexpensive and easy to setup.

Acknowledgements

First, I would like to thank my supervisor Professor Hong Zhang for his expert guidance throughout my degree. Also, I would like to thank Dr. Ron Kube for all his advice and feedback.

Finally, thank you to my wife Kaitlyn, whose support and encouragement has made it possible for me to complete this, I can't thank her enough.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The mobile robotics industry is expanding rapidly with applications in warehouse automation, self-driving-cars, service robots, agriculture and more. As a result, researchers and companies require convenient methods for quantifying robot navigation performance.

Robot navigation involves determining the robot pose within its environment (localization and mapping) and moving to goal locations (path planning) [30]. Robot pose refers to the position and orientation of the robot. Poses are often represented as a 4×4 rigid-body transformation matrix.

Mapping involves building a representation of the robot's environment. Two common map types used for localization are position-based and feature-based [42]. With position-based maps each index in the map represents a location for example, an occupancy grid. For feature-based maps each index contains a feature representation plus the coordinates of the feature. Fig. 1.1 shows an example of the two different map types.

Localization is the problem of estimating the robot's pose within a known map. The combination of localization and mapping is called Simultaneous Localization and Mapping (SLAM). Path planning relies on pose estimates from localization or SLAM so the robot can determine where it is relative to a goal.

Path planning is the problem of finding trajectories to goals that avoid obstacles [38]. To build maps, localize and safely navigate planned trajectories robots need to measure their internal state and perceive the external features

in the world using sensors [38].

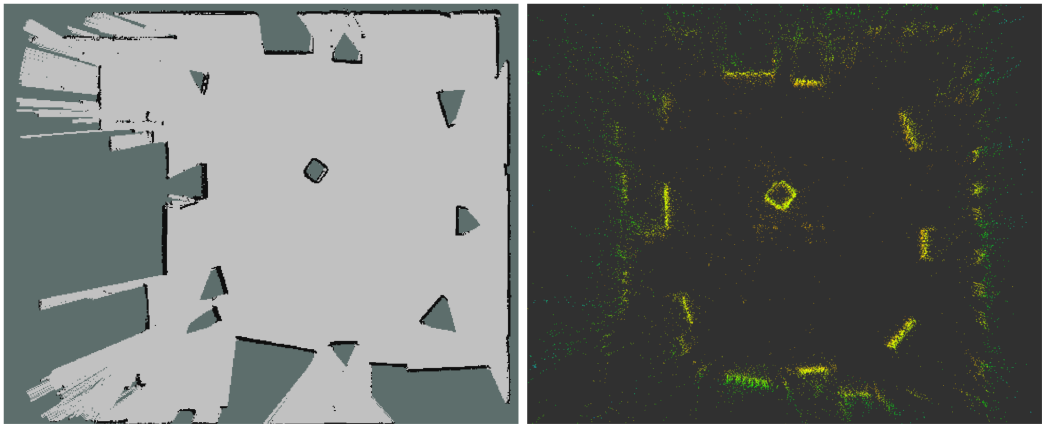


Figure 1.1: Example of two different types of maps. On the left, there is a occupancy grid map (position-based) created using Gmapping, and on the right is point cloud map (feature-based) built using RGB-D Orbslam2.

Sensors are used by robots to estimate their pose, perceive obstacles, and understand their environment. All sensor data has noise and usually does not provide complete state information therefore, most robots use a suite of sensors whose data is combined to produce more accurate state estimates.

Some common sensors used by mobile robots include wheel odometry, inertial measurement units (IMU), laser sensors and computer vision. Wheeled robots have wheel odometers to measure the rotational position and velocity of the robot's wheels. IMUs use accelerometers, magnetometers and gyroscopes to measure the acceleration, orientation and velocity. Laser based sensors include Laser Rangefinders and LiDAR (light detection and ranging), and can measure the distance to objects. Computer vision sensors include monocular RGB cameras, stereo vision which use multi-camera systems to measure scene depth and also RGB-D cameras which use infrared to measure depth. Computer vision is used for many tasks including scene understanding, place recognition and visual odometry. All the sensors discussed can be used in some way to help estimate the robot pose. When evaluating how well a navigation system works the quality of pose estimation is often used.

This thesis will focus on evaluating robot navigation performance using localization error. The localization error measures how well the robot can

determine its current position. Being able to measure localization error is useful for comparing different algorithms and tuning parameters. Ground-truth methods for measuring a robot's navigation performance directly, such as a motion capture system, are expensive and inconvenient to set up and calibrate. Because ground-truth methods are expensive and difficult to set up many researchers rely on datasets, which are useful for providing a consistent benchmark but may not test all required situations. Having a simple way of measuring robot localization performance would make it easy to test a localization system in a variety of environments.

This thesis presents a method for evaluating robot localization performance using position measurements of randomly placed visual markers.

1.1 Robot Localization Performance Evaluation

To evaluate localization performance you typically need ground-truth, examples of ground-truth methods include, motion capture and GPS. Trajectory error is one of the most useful metrics for evaluating localization performance as its a single number that provides a concise measure of performance. The proposed method in this thesis estimates the mean Absolute Trajectory Error (ATE). Mean ATE is also referred to as mean position error or mean translational error. Other metrics for evaluating localization and SLAM include robustness, computational complexity and map quality.

1.2 Contribution

The proposed method does not require ground-truth measurements and only uses the robot's estimated poses as well as the robot-to-marker poses (see $\mathbf{X}$ in Fig. 1.2) of visual markers placed in the robot's navigation space. The robot moves around its navigation space collecting samples then uses the samples to calculate constraints for an optimization technique used to estimate the mean position error. Several different localization and SLAM algorithms are compared, with three sensors and two robots in multiple environments. The

results show the proposed method can estimate the mean translation error within an order of magnitude of the ground-truth error obtained using a motion capture system.

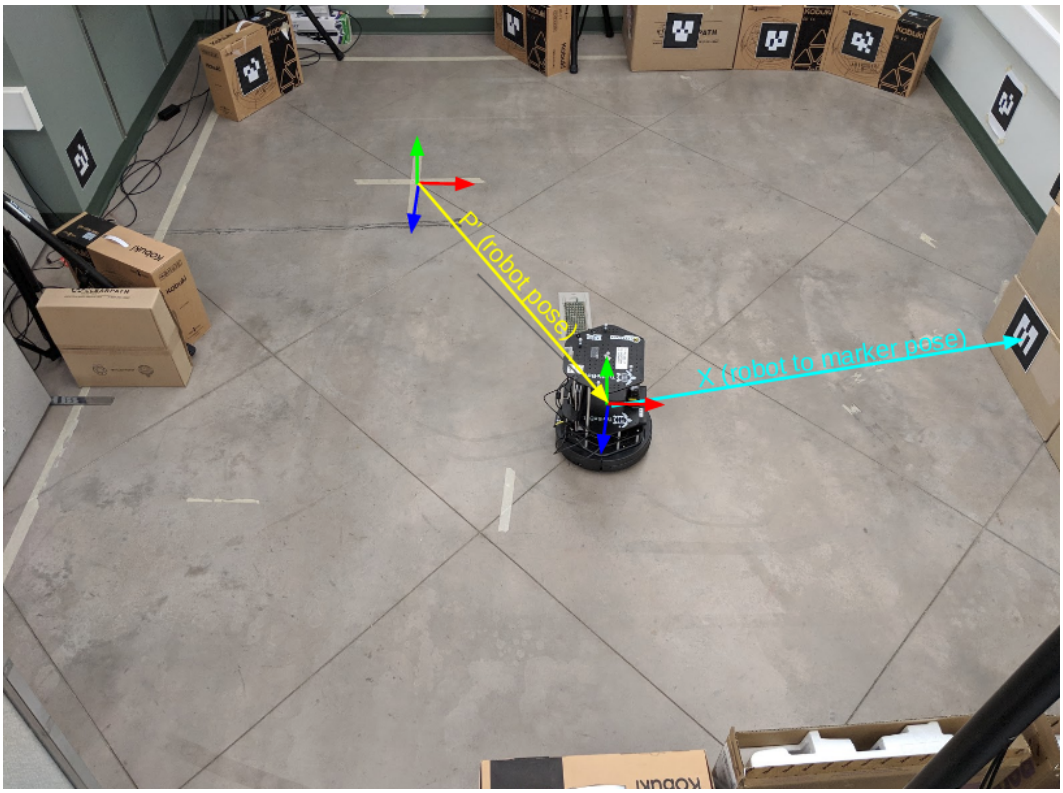


Figure 1.2: A robot navigating around its workspace performing localization while sampling visual marker poses. Given the ground-truth pose $\mathbf{P}'$ (for example from a motion capture system) the localization error could be calculated directly. The proposed method does not use ground-truth measurements and instead uses robot-to-marker poses, $\mathbf{X}$, of markers placed at unknown locations for evaluating localization performance.

1.3 Organization

Chapter 2 gives an overview of Markov Localization as well as Simultaneous localization and mapping (SLAM) and then summarizes the four different algorithms that the proposed method was tested on. Chapter 3 gives of summary of the metrics used for evaluating localization and SLAM systems, then provides an overview of several different ground-truth methods for evaluating localization performance. Chapter 4 describes the proposed method for esti-

mating the mean position error of a localization system, and then Chapter 5 discusses the experiments and results.

Chapter 2

Robot Localization and SLAM

In this section, an overview of robot localization is provided including Markov Localization and SLAM. The goal is to give the reader some background knowledge for Section 2.3 where the localization and SLAM methods used for experiments are described.

In robotics, localization refers to estimating the robot's position and orientation given a map using the robot's sensor data. Some sensors can directly provide global localization such as GPS and motion capture; however, in most cases data from multiple sensors must be blended [42]. Systems that combine localization and mapping are called Simultaneous Localization and Mapping (SLAM).

2.1 Markov Localization

Markov Localization encompasses techniques that use the Bayes filter for robot localization. The textbook Probabilistic Robotics [42] provides a thorough study of Markov Localization. The Bayes filter allows blending prediction of robot movement with measurements from sensors, given that both sources of information have noise. The Bayes filter state is updated in two steps the motion update and the measurement update which incorporates the data from sensor measurements, a high-level depiction of the algorithm is shown in Fig. 2.1 [24]. The final state should fall between the predicted state and measured state.

With Markov Localization the state is represented as a probability density

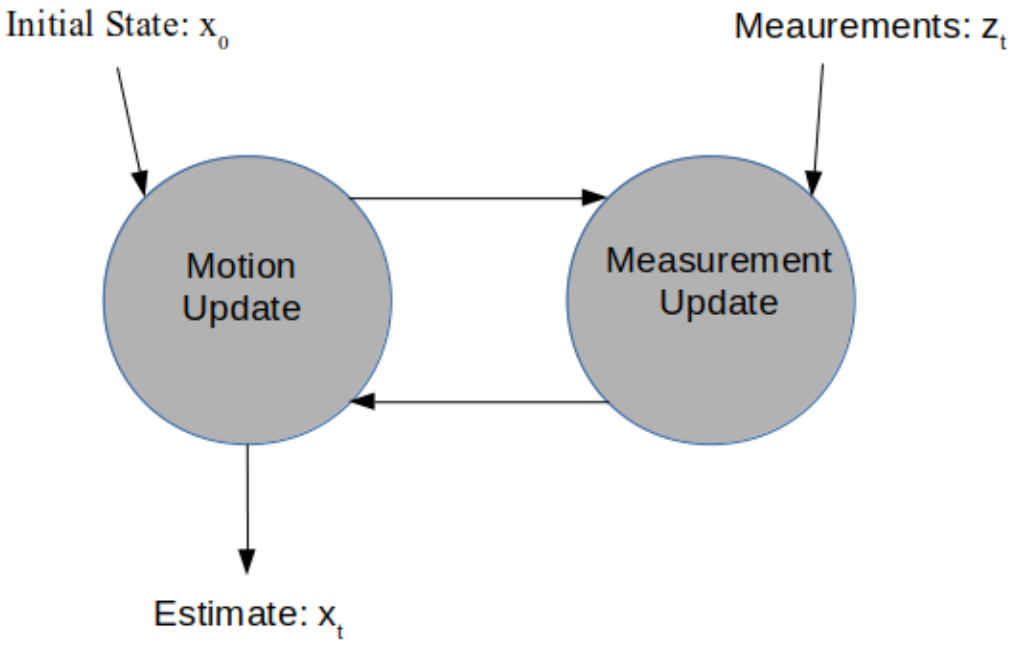


Figure 2.1: The Bayes filter Algorithm is the basis for filtering based methods like the Kalman filter and particle filter.

function over the space of robot locations. During the motion update, the next state estimate is predicted using the motion model of the robot, for example, using the velocity measured from wheel encoders. Then during the measurement update the measurement model is used to update the state based on the likelihood that a given measurement could have been observed. Two common methods are Kalman Filters and Monte Carlo Localization (MCL).

Kalman filters represent the state of the robot using a Gaussian with some mean and covariance. The mean is the estimated value for the robot position and the covariance is the estimate uncertainty. Because robot control and measurement are non-linear the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF) are used for robot localization. In the UKF and EKF, the non-linear motion and measurement model are approximated using different linearization methods.

The MCL algorithm is a particle filter method where the robot state is represented by the set of particles. Particle filters are popular as they are easy

to implement and non-parametric so they can represent multimodal probability distributions [42]. Each particle is a separate sample of the robot state, for example, x,y, heading. Further, each particle has a weighting that is a measure of how likely a particle represents the actual state of the system. The algorithm works by running the Bayes filter algorithm on each particle. The algorithm has the following steps [24]:

1. Sample particles: from previous distribution and calculate their weight.
2. Motion step: update particles position using robot motion model with noise.
3. Measurement step: given new measurements re-weight particles based on the likelihood that a measurement matches the particle state.
4. Re-sample: replace unlikely particles (low weight) with more likely particles.
5. Compute Mean: compute weighted mean to get estimated state.

Global localization is achieved by spreading the particles evenly throughout the map to start as shown in Fig. 2.3 (b). However, the default MCL algorithm cannot handle if global localization fails (converges on the wrong position) or the kidnapped robot problem because when AMCL is running only the most likely particles survive; so if it converges to the wrong solution there is no recovery. To solve this problem and add robustness to the algorithm random particles are added with some probability.

2.2 SLAM

In scenarios where a map is not available then SLAM can be used. SLAM builds a map and at the same time localizes the robot in the map. Cadena et al. provide a review of the SLAM literature including the state of art and future research directions [6]. Two general categories of SLAM are smoothing based and filtering based methods. Smoothing methods typically utilize maximum

a posteriori (MAP) estimation, where the problem is represented as a pose graph. Filtering SLAM methods use a probabilistic formulation estimating the robot pose and map using different variants of the Bayes filter algorithm.

SLAM filtering methods estimate a probability distribution over the robot state and a map. Like with Markov Localization they use a variant of the Bayes filter algorithm where there are motion and measurement updates, using the motion and measurement models respectively. Because maps can have many features the state space becomes much larger when moving from localization to SLAM with filtering techniques; as a result, many methods marginalize the map state to make the problem easier to solve.

The EKF-SLAM algorithm is one example of a filtering method. The EKF-SLAM map is assumed to be made up of landmarks and the robot and landmarks state estimates are updated using the Extended Kalman Filter. Some drawbacks of the EKF-SLAM algorithm are that the number of landmarks usually needs to be limited due to computational complexity and it is sensitive to incorrect loop closures [10].

Another filtering approach to the SLAM problem is the Rao-Blackwellized Particle Filter (RBPF). The advantage of RBPFs over EKF-SLAM is that particle filters can represent multimodal state estimation and non-linear motion and measurement models. The motivation for using the Rao-Blackwellized filter is that particle filters are inefficient in high-dimensional spaces. With RBPFs the particle filter only needs to estimate the robot trajectory and for each particle the map is computed analytically given the particles estimated robot trajectory and the past sensor observations. The FastSLAM and GMapping methods are two examples of RBPFs. FastSLAM assumes a landmark map representation, whereas GMapping uses occupancy grids.

Graph-based SLAM systems generally have two parts: the frontend and the backend as shown in Fig. 2.2 [6]. The frontend's task is to extract features from sensor data and perform data association to construct the graph [6]. For example, the visual SLAM (VSLAM) system ORBSLAM [34] has a frontend which extracts and tracks ORB features from camera images. The graph nodes are landmarks and robot poses while the edges are sensor measurements and

loop closures [19]. The backend then performs graph optimization to find the best node configuration to satisfy the edge constraints. The constraint optimization problem is solved using non-linear least squares minimization. Local regions of the map will typically be associated with some landmark representation such as keyframes (ORB-SLAM2) or submaps (Cartographer) to be combined into a globally consistent map and rearranged after global optimization.

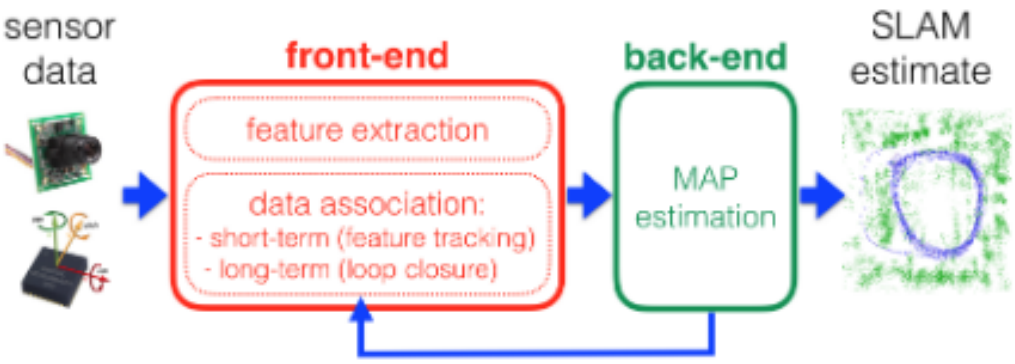


Figure 2.2: The typical structure of a SLAM system (figure from [6]).

2.3 Localization and SLAM Systems used for Experiments

Several different SLAM and localization algorithms were evaluated using the proposed method in this thesis. This section summarizes the different systems to give some insight as to why one method performs better than another. The algorithms used for experiments are AMCL [42], RTAB-map [29], Google Cartographer [23] and GMapping [18].

2.3.1 AMCL

Adaptive Monte Carlo Localization (AMCL) is an implementation of MCL that changes the number of particles using KLD-sampling [13]. Being able to change the number of particles depending on uncertainty in the current state estimate makes it more efficient than other methods and also has better

localization performance than the standard MCL algorithm. A visualization of the AMCL particles filter in different states is shown in Fig. 2.3. The ROS version[1] uses laser scan and odometry data.

The default Likelihood Field measurement model [42] is used when running AMCL for this thesis, where each beam end-point of the scan in projected into the global euclidean coordinate space and its nearest neighbor is found in the map. Next, the probability of the laser scan matching the particle state is calculated incorporating measurement, failure and random noise. Since AMCL is a particle filtering method it uses the sampling odometry measurement model [42] where the robot motion is decomposed into a rotation, then translation and another rotation. Then the change in pose is simply guessed by sampling from the posterior distribution with noise.

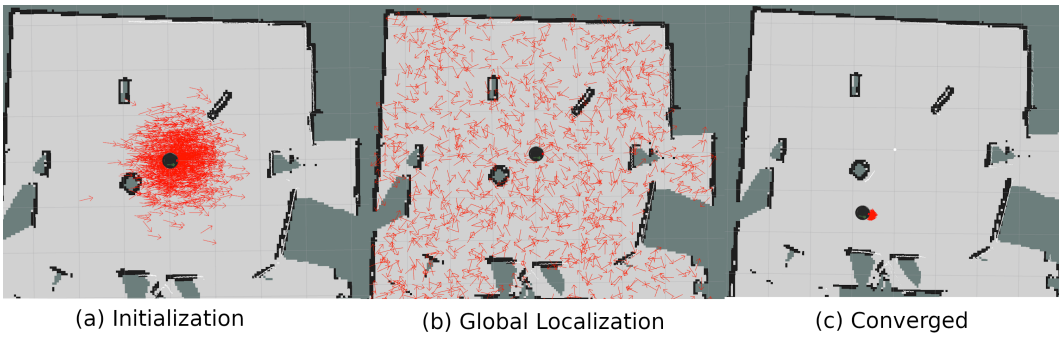


Figure 2.3: Visualization of AMCL particle filter in different states. (a) shows the particle filter after initialization with the particles spread around the estimated starting pose of the robot. (b) shows the particle filter after running Global Localization, where all the particles have been spread evenly throughout the map. (c) shows the particle filter that has converged to a low variance estimation of the robot state.

2.3.2 RTAB-Map

RTAB-Map is a graph-based SLAM system with appearance-based loop-closure and requires an RGB-D or stereo camera and odometry [29]. The system can also incorporate 2D laser scan or 3D point cloud data. See Fig. 2.4 for a high-level overview of the RTAB-Map system [29].

[1] http://wiki.ros.org/amcl

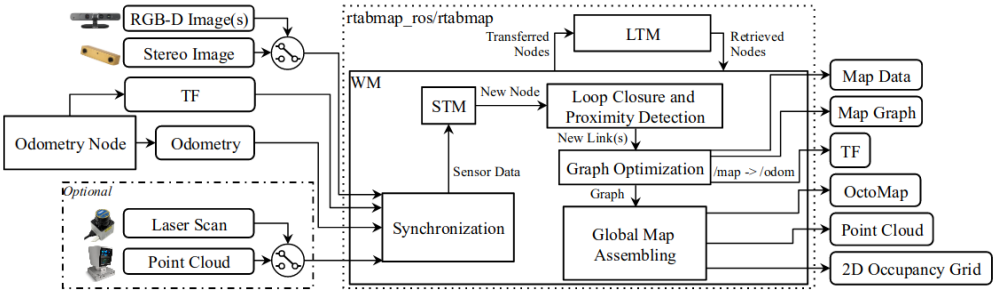


Figure 2.4: High-level overview of RTAB-Map system (figure from [29]).

If the robot does not have an odometry sensor or the odometry is not very accurate, the authors of RTAB-Map also included in their library visual and laser odometry modules. The visual odometry module utilizes feature matching and Perspective-n-Point RANSAC followed by local bundle adjustment to estimate the motion of the robot. Besides their original visual odometry module, they include seven other approaches including ORB-SLAM2 [34] for easy comparison. The lidar odometry module uses the iterative-closest-point (ICP) algorithm for scan matching and motion estimation.

The RTAB-Map graph map is made up of nodes and links where the links are constraints for graph optimization. The links can be created either by loop closure, proximity detection or neighbors. Loop closure is implemented using the Bag of Words approach. Proximity detection is used to localize nodes nearby using a laser scan and is useful in situations where visual loop closure is not reliable. The neighbor links are added by RTAB-Map's Short-term Memory module (STM) when new odometry messages are received.

When a new node is added by the STM then a local occupancy grid is also created and associated with that node. The occupancy grids can be created using laser scans or point clouds and can be either 2D or 3D. The local occupancy grids are combined into a global occupancy grid using the global map poses from their associated nodes. An example of a merged RTAB-Map global occupancy grid point cloud is shown in Fig. 2.5. When loop closure detection occurs then graph optimization is run and the occupancy grid is reassembled using updated poses. RTAB-Map gives the option of using TORO

[17], g2o [28] or GTSAM [11] for global optimization as each has different strengths and weaknesses.

Figure 2.5: Example of RTAB-Map's global map point cloud.

2.3.3 Google Cartographer

Cartographer is a real-time laser-based graph SLAM method [23]. Cartographer can function with just laser data but can also incorporate odometry and Inertial Measurement Unit (IMU) data. An overview of the full Cartographer system is shown in Fig. 2.6 [22]. A cartographer map is built up of submaps which are small chunks of the world.

Submaps are represented by grids that give the probability of being obstructed. Each submap contains a set number of consecutive scans for the area it represents. Scans are matched to submaps using a local optimization where the transformation from the scan frame to submap frame is found. This local optimization is a non-linear least squares optimization using the ceres-solver [1]. Once a submap has enough scans it is considered finished and used for checking for loop closure detection.

Loop closure detections are found by matching scans against the scans in nearby submaps. The global scan matching process happens in the background using a brand and bound algorithm [8] to achieve real-time performance. If there is a good match then it is used as a constraint in the global optimization.

The global optimization rearranges the submaps so they form a consistent global map. The graph optimization uses nodes (pose trajectory) and submaps. The constraints are relative poses between nodes and submaps as well as between scans and submaps. Again the ceres-solver is used for optimization.

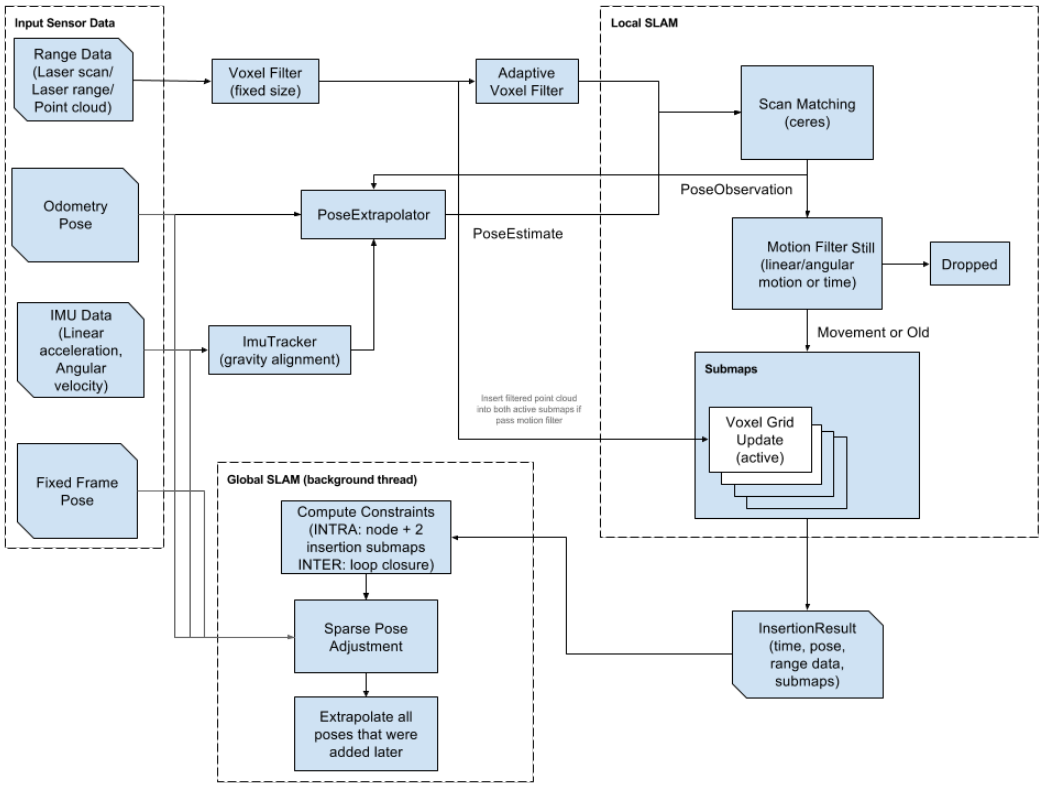


Figure 2.6: High-level overview of Google Cartographer system (figure from [22]).

2.3.4 GMapping

GMapping is an improvement on the Rao-Blackwellized particle filter (RBPF) [9] method where each particle contains an estimate of the map and robot trajectory. GMapping builds occupancy grid maps like the one shown on the

left side of Fig. 1.1. The two improvements GMapping introduced are better particle generation by improving the proposal distribution and better particle re-sampling [18].

GMapping improves the particle proposal distribution by using laser range data. The standard particle filtering sampling method uses the odometry motion model; however, this approach may not be optimal if the odometry data is not very accurate. Grisetti et al. combine the odometry model with the most likely pose from scan-matching to generate particles with better estimates of the current state. The improvement leads to better maps and pose estimation, as a result, fewer particles are required.

The particle re-sampling method is improved in GMapping so that re-sampling only occurs when needed. In GMapping they estimate how well the current particle set estimates the target posterior based on how spread out the particle importance weights are. Where the assumption is that all particle weights would be the same if they were drawn from the target distribution. If the particles become too dispersed then they re-sample. By only re-sampling when needed they reduce the chance of removing good particles from the particle set.

2.4 Summary

Localization is the task of estimating the robot's position within a known map. Traditional approaches to localization use the Bayes filter algorithm. When a map is not available then SLAM approaches are used, where the algorithm builds a map while localizing the robot within the map. Modern SLAM systems typically consist of a frontend, which provides data association between sensor readings and constructs a pose graph, and a backend that runs global optimization to adjust the graph using measurement constraints.

In this thesis, several different localization and SLAM systems are evaluated using the proposed localization error estimation method. AMCL is a pure localization method that uses particles filters and known occupancy grid maps. RTAB-Map is a graph-based SLAM system that builds dense 3D point cloud

maps, uses appearance-based loop closure detection and requires an RGB-D or stereo camera. Cartographer is a graph-based SLAM system, which uses laser data and builds occupancy grid maps. Finally, GMapping is an RBPF SLAM method that is also laser-based and builds occupancy grid maps.

Chapter 3

Evaluating Robot Localization Performance

To evaluate a robot localization system the trajectory error evaluation metric is used. Calculating the pose error requires a ground-truth measurement. The ground-truth pose is commonly obtained using accurate sensor systems like GPS+IMU and motion capture or using external markers that can be accurately localized. Datasets are the most popular way to evaluate a robot SLAM or localization method.

There are many SLAM datasets that exist and are used for evaluating different SLAM and localization systems [5], [15], [32], [39], [41]. These datasets typically provide a ground-truth robot pose, which is used for calculating the position error and rotational error of the system being evaluated. The method proposed in this thesis only estimates position error; however, since rotation error leads to position error then rotation error can be indirectly measured with only position error [41]. Datasets are extremely useful for providing a repeatable benchmark for comparing methods, however, they only provide a subset of possible scenarios to test.

The first part of this chapter goes over different evaluation metrics used for evaluating localization systems including the one used for Chapter 5 experiments. The second part reviews the commonly used ground-truth measurement systems and some of the datasets that use them.

3.1 Evaluation Metrics

Evaluation metrics are used when comparing localization methods or tuning them to get better results. A variety of metrics exist including map quality, computational efficiency, robustness and trajectory error.

3.1.1 Trajectory Error

Trajectory error is the most popular metric used for comparing localization systems as it does not depend on the type of map or sensor used [27], which allows researchers to compare one algorithm against another. It also provides a concise numerical result that can be used for feedback when tuning a system. There are two ways of calculating trajectory error: Absolute Trajectory Error (ATE) and Relative Pose Error (RPE) [41]. Sturm et al. note that when using ATE and RPE for evaluation of SLAM systems the relative ordering tends to be consistent between the two metrics [41].

RPE is the difference in ground-truth motion and estimated motion over a fixed time period Δ and measures the local accuracy of localization [41]. Given the estimated trajectory $\mathbf{P}_1, ..., \mathbf{P}_n \in SE(3)$ and the ground-truth trajectory $\mathbf{P}'_1, ..., \mathbf{P}'_n \in SE(3)$ the RPE for time i can be calculated as follows:

$$RPE_i = (\mathbf{P}_i'^{-1}\mathbf{P}'_{i+\Delta})^{-1}(\mathbf{P}_i{}^{-1}\mathbf{P}_{i+\Delta}) \tag{3.1}$$

Using the above equation the RMSE, mean or median of the translational and rotational components can be calculated over the entire trajectory. Typically researchers report the RMSE or mean. To measure the global error of a trajectory it is recommended to calculate the mean (or RMSE) RPE for all possible Δ and take the average, which provides a good evaluation of a SLAM system.

ATE requires aligning the estimated robot trajectory with the ground-truth trajectory to calculate pose error and evaluates the global consistency of a trajectory [41]. Given the rigid-body transformation $\mathbf{S}$ mapping the estimated trajectory ($\mathbf{P}$) into the ground-truth trajectory's ($\mathbf{P}'$) coordinate frame, the

ATE can be calculated for time i:

$$ATE_i = \mathbf{P}_i'^{-1}\mathbf{S}\mathbf{P}_i \tag{3.2}$$

The method presented in this thesis estimates the mean translational error, which is the mean ATE, as a result when calculating the ground-truth mean ATE is used:

$$Mean(ATE_{1:n}) = \frac{1}{n}\sum_{i=1}^{n} ||trans(ATE_i)|| \tag{3.3}$$

3.1.2 Others

Besides trajectory error, it is also important to consider robustness. In [40] they present a protocol for evaluating a complete navigation system over a long period of time. Their tests involve a standard environment where they provide a specification for the size and layout for the testing areas the robot will navigate in. In standardized environments, they then have a set of challenges that not only test trajectory position error, but also robustness using the number of failures, time to failure, distance to failure and average speed. A key part of the challenges is that the navigation must happen over a long period and the environment will be modified throughout (eg. objects are moved or added, dynamic obstacles, etc.). The position error is measured using an external marker based method that they developed [25]. They also use a reference robot (Adept Pioneer 3DX) and reference navigation system that are a baseline across all navigation areas. They use their evaluation test-bed to compare two different robot systems at two different institutions using the reference robot at both locations to provide a baseline.

Another consideration when comparing systems is their computational complexity Bodin et al. designed a Visual SLAM benchmark which includes metrics for measuring computation speed, power consumption and memory usage [3]. At the time of publishing they supported eight different SLAM algorithms. They aimed to make it dataset agnostic and easy to test new

algorithms with it. For their experiments, they run the algorithms on three different machines of varying computational speed: Odroid, NVIDIA Jetson TK1, and a desktop with an i7 intel CPU and NVIDIA GTX 1080 GPU. Metrics that measure computational complexity are useful for hyperparameter tuning of SLAM systems to optimize the speed/accuracy tradeoff which is important on memory constrained devices.

With SLAM systems you can also evaluate based on the map as the map quality directly affects localization performance. For example, you can compare a SLAM map with the building floor plan, or compare the relative distance between landmarks [27]. Filatov et al. present three quantitative evaluation metrics for 2D SLAM maps: the proportion of occupied and free cells, corner count and the number of enclosed areas [12]. The proportion of occupied cells shows how blurry a map is or if you have duplicate walls and is a measure of map quality. Filatov et al. assert a map with fewer corners will have fewer artifacts and is usually more consistent than a map with more. Finally, enclosed areas likely represent places where walls have overlapped due to drift and failed loop closure [12].

3.2 Ground-Truth Localization Error Estimation Methods

Most methods for evaluating trajectory error need ground-truth measurements. This section goes over some of the different ground-truth methods, including GPS and IMU sensor fusion, motion capture, simulation and external markers. This section also summarizes the different datasets that have ground-truth measurements provided.

3.2.1 GPS and IMU Sensor Fusion

Outdoors Ground-truth is usually obtained from fusing data from precise sensors like Global Positioning System (GPS), inertial measurement unit (IMU), and Light Detection and Ranging (LiDAR). This section summarizes three datasets [15], [32], [39] that include GPS and other sensor data for ground-

truth.

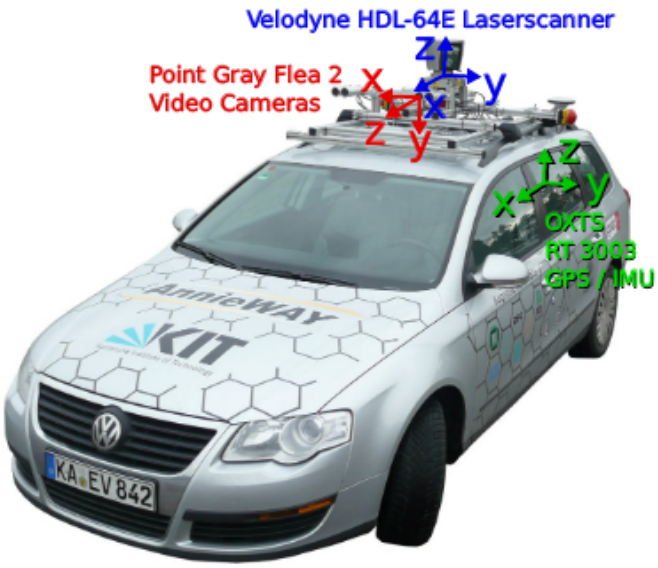


Figure 3.1: The recording platform used for collecting the KITTI dataset (figure from [14]).

The KITTI dataset [14], [15] was collected on autonomous vehicle in an urban area and includes data for benchmarking a variety of autonomous driving tasks including visual odometry and SLAM. Their goal was to provide a large scale and challenging dataset in an uncontrolled environment. The sensor suite includes GPS and IMU localization which is used to measure ground-truth 6 degree of freedom (DOF) position. Fig. 3.1 shows the car and sensors used for collecting the KITTI dataset [14]. For the visual odometry and SLAM evaluation set they selected approximately 40 KM of driving with high-quality ground-truth localization. They made sure to include loops in the evaluation set so that SLAM loop closure detection can be tested. The KITTI benchmark suite [15] uses the RPE evaluation metric. They use their dataset to evaluate several comparable visual odometry systems without loop closure detection; however, since being published, this dataset has become one of the most popular datasets for comparing SLAM algorithms.

The Oxford RobotCar Dataset [32] contains over 1000 Km of driving data on an autonomous car. The dataset was collected on one route over one year.

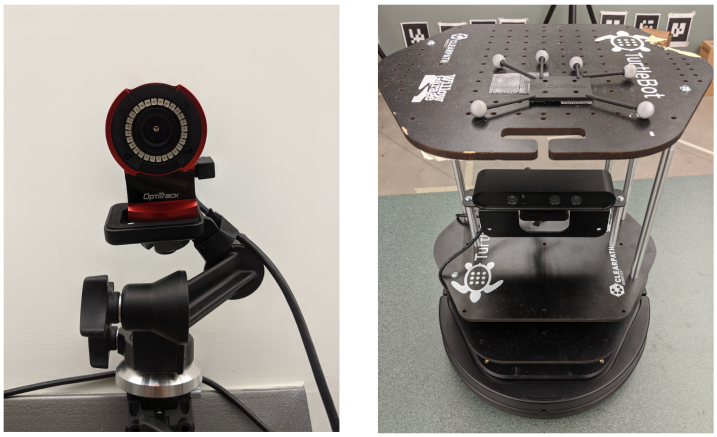
The goal was to provide a large-scale dataset with a variety of conditions (weather, illumination, etc.) on one route so that research could be done on long-term autonomy. To aid long-term autonomy research they provide tags for each traversal that describe the conditions for that traversal, for example, clouds or night. The dataset provides GPS, INS (Inertial Navigation System), LiDAR, images and stereo visual odometry. Because the quality of the GPS data varies throughout the dataset they do not recommend using it directly as the ground-truth robot pose. Also, the pose provided by the visual odometry system drifts over time so is not suitable to be used directly as ground-truth either.

The New College Vision and Laser Data Set [39] was collected on a two-wheeled mobile robot driving around a University campus. The dataset was collected to be used for robot navigation and mapping research. The data includes odometry, two 2D laser scanners, stereo images, panoramic camera and GPS. Further, the dataset contains three sections, one for each area the robot was navigated in.

Fusing IMU and GPS produces reliable 3D data on a moving vehicle and works outdoors [15]. However, using multiple sensors increases the complexity of data synchronization and sensor calibration [15], as well as increases the cost of the system.

3.2.2 Motion Capture

Indoors GPS is not an option so motion capture systems are used. Motion capture systems measure precise 6 DOF pose using multiple cameras and reflective markers. Fig. 3.2 shows an example motion capture camera and the reflective markers used for tracking placed on a robot. The accuracy of motion capture makes it useful for evaluating robot navigation. Some motion capture systems can be used outdoors using specialized hardware, software and/or active markers. This section reviews two different datasets that use motion capture ground-truth [5], [41], as well as an example of motion capture used for robot localization performance evaluation [37] and finally a method that is similar to motion capture using external sensors.



(a) Motion capture camera (b) Reflective markers on Turtlebot

Figure 3.2: Optitrack Flex 13 motion capture camera and reflective markers used on robot.

Sturm et al. created the TUM dataset for benchmarking visual SLAM systems and they used a motion capture system to get the ground-truth [41]. The dataset contains sequences of RGB and depth images collected in office and industrial environments using an RGB-D camera. Some of the sequences are collected using a handheld camera and others were collected from a mobile robot. For their motion capture system they use eight MotionAnalysis Raptor-E cameras. For extrinsic calibration they placed motion capture reflective markers on the camera and the four corners of a checkerboard marker and they found that the motion capture system has an absolute error of less than 10 mm and 0.5°.

Another SLAM dataset which uses motion capture for ground-truth is EuRoc [5]. The dataset was collected on a Micro Aerial Vehicle and contains hardware synchronized stereo images and IMU data. The dataset is used for comparing different visual-inertial SLAM systems and was used as part of the evaluation for the European Robotics Challenge (EuRoc). Two environments are included in the dataset. The first is a large unstructured machine hall and the ground-truth is provided by a Leica total station. The machine hall also has variable illumination which makes it more challenging. The second environment is 8 m $\times$ 8.4 m $\times$ 4 m room with artificial objects added and the ground-truth is obtained from a Vicon motion capture setup. The ground-

truth measurements from the motion capture and Leica are aligned with the sensor coordinate frame to calculate the ground-truth error.

Röwekämper et al. also use a motion capture system to obtain the ground-truth for measuring the position accuracy of a localization system [37]. They evaluated Monte-Carlo localization running on a Kuka omniRob platform running in an industrial environment. The robot navigates to set reference positions while being evaluated and the error is evaluated relative to the reference positions. For evaluation metrics Röwekämper et al. use position error and yaw error. Further, they provide both the positioning error and localization error. The positioning error measures how well the robot can get back to a fixed waypoint and is dependent on the robot controller and hardware.

Ceriani et al. present two methods for indoor ground-truth data collection GTvision and GTlaser [7] that use external sensors to locate the robot similar to a motion capture system. GTvision uses a network of cameras to track the robot's 6 DOF ground-truth position by estimating the pose of AR markers attached to the robot. To get the extrinsic calibration for the camera network they use two checkerboards rigidly mounted to a metal frame to get the relative pose between two cameras, they do this for all camera pairs chaining the transformation back to the world reference frame camera. The robot pose is found using the identified markers on the robot to solve the Perspective-N-Point problem. GTlaser uses a network of 180° FOV laser scanners to track the robot's 3 DOF ground-truth position by identifying a custom hull attached to the robot. The extrinsics for the laser network are found by aligning the output of the scans using objects with known shapes placed in the environment. To find the robot pose using the laser network the Iterative Closest Point algorithm is used. They tested their two ground-truth systems in two different environments and use manual measurements to validate them. The GTlaser method has a mean translation and angular error of 20 mm and 0.15°, respectively. The GTvision method is less accurate but more flexible with mean translation error of 112 mm and mean angular error of $-0.8°$. Both methods require a large amount of setup and calibration.

Motion capture systems provide a very accurate pose estimate that does

not drift over time [37], [41]; however, for each new test environment a motion capture system must be taken down, setup and re-calibrated. Also, the number of cameras available restricts the area covered by the motion capture system.

3.2.3 Simulation

In simulation calculating trajectory error is trivial as the robot's exact ground-truth pose can be queried. However, simulators usually provide very clean data that does not contain outliers or realistic noise, which means evaluation results from simulation generally do not transfer to the real world. Still though, because simulators make evaluation easy and can provide detailed sensor data they are useful. For example, Handa et al. published a dataset for SLAM research using simulated data [21].

The ICL-NUIM dataset is a simulated dataset for testing SLAM and visual odometry algorithms [21]. They use an open source tool POVray[1] to simulate two different scenes and provide exact ground-truth pose for camera trajectories in those scenes. The key feature of this simulated dataset is that they also provide ground-truth surface models, which are hard to obtain in real life. The surface models can be used for comparing maps or evaluating 3D surface reconstruction. For the sequences they provide clean noiseless RGB-D. In their evaluation they compare different algorithms using the clean data versus data they apply a noise model to.

3.2.4 External Markers

Another way to evaluate pose error is to use external markers that can be localized accurately. For example, [43] uses retroreflective markers and [25] uses visual markers.

Tong et al. created a ground-truth method that does not require external sensors and calibration [43]. They place retroreflective markers in the environment and locate them using a SICK LMS291 laser mounted on a pan-tilt unit to get a 360° by 180° scan. For a maximum sensing range of 40 meters

[1] http://www.povray.org/

they use 1.2 m $\times$ 1.2 m markers. The landmarks are identified using k-means clustering on the scan data. Their system requires three visible retroreflective markers to localize the robot using batch SLAM. The SLAM algorithm aligns point clouds while also estimating the position of the visible landmarks to estimate the robot position. They evaluated their method using a motion capture system in a lab environment and report a relative root mean square translation error of 2.1 cm and 0.51° angular error. They also show that their method works in a larger scale 40 m diameter environment.

Kikkeri et al. [25] demonstrated a method to measure the localization error for a navigation system. The inexpensive landmark-based system uses an RGB camera to locate printed patterns placed on the ceiling at waypoints throughout the environment. For visual markers, they use custom pattern combining checkboard and circle patterns, which allows them to detect the marker even if partially occluded. To initialize their system, waypoints are collected for each marker as well as the camera-to-marker pose. During their evaluation phase, they have the robot drive to poses collected during initialization and then compare the current camera-to-marker pose with the one saved during initialization, this difference in pose is the metric used to evaluate the accuracy of the localization. They evaluate their method on Adept Pioneer 3DX robot and report an average position error of 15 mm and average angular error of $-0.4°$ in a 1.5 m $\times$ 1.5 m area. They also demonstrate their method scales well to large environments using a 15 marker sequence in an approximately 57 m by 52 m building and provide qualitative results.

Using external markers makes evaluating in new environments easy and can be used both indoors and outdoors. The method presented in this thesis uses external markers (AR tags) for estimating localization position error.

3.3 Summary

Trajectory error is the most popular metric for evaluating localization performance used in research, as it is easy to compare completely different algorithms using it. Other metrics like robustness, efficiency and map quality become

more important when deploying a system outside of research. The method presented in this thesis estimates the mean ATE of a localization system.

Ground-truth robot pose is required to calculate trajectory error. High accuracy GPS and IMU systems are a great way to get ground-truth robot pose outdoors. But even outdoors GPS is not always available. Indoors motion capture can be used but it requires significant setup and calibration when you want to move to new environments. Systems that use external markers are an accurate way to localize and the markers are generally flexible for use both outdoors and indoors. The next chapter describes the proposed method for estimating localization performance without a ground-truth measurement system.

Chapter 4

Robot Localization Performance Evaluation Using Visual Marker Pose Estimation

4.1 Introduction

This section presents a low-cost and convenient method for calculating robot localization error. The approach does not require the ground-truth robot position and only needs a camera and visual markers placed around the environment to calculate the localization error. The type of marker is not important, as long as its pose can be accurately estimated relative to the robot. For the experiments we use AR tags as a marker. The error can be computed by independently sampling the same marker twice to create optimization constraints and then using Generative Latent Optimization (GLO) [4] to estimate the error. The markers can be placed anywhere in the environment as long as the robot's camera can see them.

4.2 Estimating 2D Position Error Using GLO

The proposed method estimates the absolute error for a 2D localization system using measurements of robot poses with respect to visual markers at unknown locations in the robot's environment. Since the robot-to-marker pose estimation is local at a close range to the marker, the error is considered relatively small and negligible. When the robot visits a visual marker it records its pose

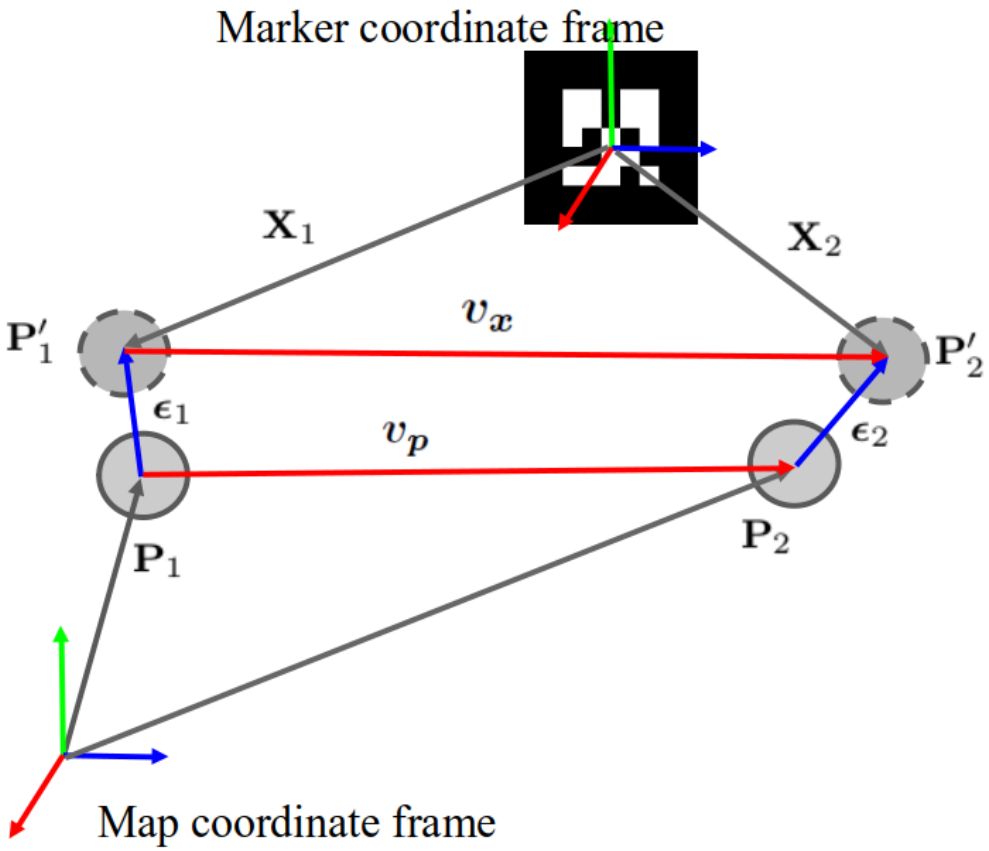


Figure 4.1: How to estimate robot localization error. A robot performing localization visits a visual marker twice. $\mathbf{P}_1$ and $\mathbf{P}_2$ are the estimated poses in the map coordinate frame, from the localization system being evaluated. $\mathbf{P}'_1$ and $\mathbf{P}'_2$, are the unknown ground truth robot poses. $\mathbf{X}_1$ and $\mathbf{X}_2$ are the marker to robot pose estimates, in the marker's coordinate frame. The goal is to estimate the position error statistics of $\boldsymbol{\epsilon}_1$ and $\boldsymbol{\epsilon}_2$, which are assumed to be independent and identically distributed random variables, from $\mathbf{P}_1$, $\mathbf{P}_2$, $\mathbf{X}_1$ and $\mathbf{X}_2$. Note that $\boldsymbol{v_p}$ and $\boldsymbol{v_x}$ can be computed but they are expressed in two different coordinate frames.

within the robot map according to the localization algorithm. At the same time, the robot also computes its pose relative to the marker. After several visits to the same marker, the method uses GLO [4] to estimate the error of the localization algorithm with respect to the map reference frame. GLO is used as it can estimate a variable that cannot be measured directly but is a function of some other observable values. Therefore, the method can quantify the performance of a localization algorithm from local relative measurements only without the need for a global measurement system such as motion capture

Algorithm 1: The first step of the proposed method for estimating 2D localization error. First $\boldsymbol{v_x}$ and $\boldsymbol{v_p}$ pairs (see Fig. 4.1) are collected into a dataset D, each pair from visiting a visual marker more than once by a robot performing localization.

Step 1: Collect samples

Input : N = number of samples to collect
Output: $D = \{ (\boldsymbol{v}_{\boldsymbol{x}}^i, \boldsymbol{v}_{\boldsymbol{p}}^i)\ i = 1, ..., N\}$
$i = 0$
while $i < N$ **do**
  check if visual marker in range
  if *repeat visit* **then**
    compute $\boldsymbol{v}_{\boldsymbol{x}}^i$ with $\mathbf{X}_1^i$ and $\mathbf{X}_2^i$
    compute $\boldsymbol{v}_{\boldsymbol{p}}^i$ with $\mathbf{P}_1^i$ and $\mathbf{P}_2^i$
    i++
  end
end

or global ground truth positions.

The intuition for the method is illustrated in Fig. 4.1, where $\boldsymbol{v_p}$ and $\boldsymbol{v_x}$ can be calculated after two visits to the same visual marker. $\mathbf{P}_i$ and $\mathbf{P}'_i$ are the global pose estimated by the localization algorithm and the ground truth pose, respectively, and $\mathbf{P}'_i$ is unknown. $\boldsymbol{\epsilon}_i$ is the translational localization error whose statistics are to be determined. Note that the magnitude of $\boldsymbol{\epsilon}_i$ is the mean ATE. The position of the visual marker in the map coordinate frame is also unknown, and as a result $\boldsymbol{\epsilon}_i$ cannot be calculated directly. Assume the error in robot-to-marker pose estimation is negligible and therefore the error in the measured length of $\boldsymbol{v_x}$ is negligible. At the same time, $\boldsymbol{v_p}$ can be calculated. Most importantly, the discrepancy between the lengths of $\boldsymbol{v_p}$ and $\boldsymbol{v_x}$ is directly related to the length of $\boldsymbol{\epsilon}_i$. Exploiting this relationship, we use GLO to estimate the statistics of the x and y components of $\boldsymbol{\epsilon}_i$, assumed to be i.i.d., from N samples of ($\boldsymbol{v_x}$ and $\boldsymbol{v_p}$) that best explain the discrepancy between $\boldsymbol{v_x}$ and $\boldsymbol{v_p}$. The remainder of this section will first explain how GLO works and then show how to use it to estimate the statistics of mean ATE.

Algorithm 2: Steps 2 and 3 of the proposed method for estimating 2D localization error. Step 2: D is used as constraints in an optimization method called GLO (generative latent optimization), to estimate the the variance σ of a zero-mean random variable, which is a function of the localization error. Step 3: the statistics of the localization error, i.e., its mean and variance, are derived from σ assuming the error follows a Rayleigh distribution. In Step 2, m is a hyperparameter.

Step 2: Parameter estimation

Input : $D = \{ (\boldsymbol{v}_{\boldsymbol{x}}^i, \boldsymbol{v}_{\boldsymbol{p}}^i)\ i = 1, ..., N \} \}$
Output: σ_{est}
$c_{min} = \infty$
for σ *in range*$(\sigma_{min}, \sigma_{max})$ **do**
  $\mathbf{z} = \{\boldsymbol{z}_i ... \boldsymbol{z}_m\}$ where $\boldsymbol{z}_i \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_d)$
  $\mathbf{z}_\sigma = \sigma * \mathbf{z}$
  $c = \sum_{i=1}^{N} \left[\min \left[(\|\boldsymbol{v}_{\boldsymbol{p}}^i + \boldsymbol{\varepsilon}\|_2^2 - \|\boldsymbol{v}_{\boldsymbol{x}}^i\|_2^2) \text{ for } \boldsymbol{\varepsilon} \text{ in } \mathbf{z}_\sigma \right] \right]$
  if $c < c_{min}$ **then**
    $c_{min} = c$
    $\sigma_{est} = \sigma$
  end
end

Step 3: Localization error calculation

Input : σ
Output: μ_l, σ_l (mean and standard deviation of the translational localization error)
$\hat{\sigma} = \frac{\sigma}{\sqrt{2}}$
$\mu_l = \hat{\sigma}\sqrt{\frac{\pi}{2}}$
$\sigma_l = \hat{\sigma}\sqrt{\frac{4-\pi}{2}}$

4.2.1 Generative Latent Optimization (GLO)

GLO [4] is an algorithm, popularized in GAN (generative adversarial networks) [16] research, to learn the parameters of a function g, called generator, by minimizing reconstruction loss. For the proposed method GLO is used with a g that has a single unknown parameter whereas in image generation research, which GLO and GAN come from, the generator is a neural network with many parameters. GLO is defined by Eq(4.1):

$$\min_{\sigma \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \left[\min_{\boldsymbol{z}_i \in \mathcal{Z}} \ell(g_\sigma(\boldsymbol{z}_i), x_i) \right] \tag{4.1}$$

Consider a set of samples of unknown distribution, $x_1, ..., x_N$, where $x_i \in \mathcal{X}$. Second, a set of random vectors $\boldsymbol{z}_1, ..., \boldsymbol{z}_N$ where $\boldsymbol{z}_i \in \mathcal{Z}$ are generated then GLO attempts to learn the parameters σ of a generator $g_\sigma : \mathcal{Z} \rightarrow \mathcal{X}$ by solving Eq(4.1), where $\ell : \mathcal{X} \times \mathcal{X}$ is a loss function measuring the reconstruction error from $g_\sigma(\boldsymbol{z}_i)$ to x_i.

For this application, the unknown (latent) parameter is the standard deviation of x and y components of $\boldsymbol{\epsilon}_i$, which cannot be estimated directly from the marker pose measurements and the localization estimates. However, a function g_σ of the unknown parameter can be constructed, and used to compute the samples x_i in Eq(4.1) from the marker and robot pose measurements. As a result, GLO is applied to estimate the unknown parameter, as will be shown in detail in the next section.

4.2.2 Application of GLO to Localization Error Estimation

Now consider the situation where a robot visits a marker twice, as shown in Fig. 4.1. During both visits the robot localizes itself with respect to the map and computes the marker pose relative to the robot. Assuming that the two marker pose estimates, $\mathbf{X}_1$ and $\mathbf{X}_2$, have negligible errors.

Using Fig. 4.1 there is the following relationship between $\boldsymbol{v_x}$, $\boldsymbol{v_p}$ and the two unknown localization errors $\boldsymbol{\epsilon}_1$,$\boldsymbol{\epsilon}_2$:

$$-\boldsymbol{\epsilon}_1 + \boldsymbol{v_p} + \boldsymbol{\epsilon}_2 = \boldsymbol{v_x} \tag{4.2}$$

$\boldsymbol{v_p}$ and $\boldsymbol{v_x}$ are the translation vectors between two visits to the visual marker observed by the localization algorithm and by visual marker pose estimation, respectively. However, $\boldsymbol{v_x}$ is not available in the map reference frame, so the above equation is not of practical use. Re-arranging Eq(4.2):

$$\boldsymbol{v_x} = \boldsymbol{v_p} + \boldsymbol{\varepsilon} \; or \; |\boldsymbol{v_x}|^2 = |\boldsymbol{v_p} + \boldsymbol{\varepsilon}|^2 \tag{4.3}$$

where $\boldsymbol{\varepsilon} = \boldsymbol{\epsilon}_2 - \boldsymbol{\epsilon}_1$. Note that the magnitude $|\boldsymbol{v_x}|^2$ is invariant with respect to the reference frame and can now be used. Eq(4.3), can be rewritten in terms of individual coordinates as

$$\sum {v_x}^2 = \sum (v_p + \varepsilon)^2 \tag{4.4}$$

where the summation is over individual coordinates whose indexes are not shown for clarity. In 2D, Eq(4.4) sums over x and y directions, i.e $\boldsymbol{\varepsilon}$ has two components. Based on the symmetry of the problem it is assumed these two components follow the same distribution, i.e., they are i.i.d.. Further, since $\boldsymbol{\varepsilon} = \boldsymbol{\epsilon}_2 - \boldsymbol{\epsilon}_1$ then $E(\boldsymbol{\varepsilon}) = 0$ and the standard deviation of each component of $\boldsymbol{\varepsilon}$, σ, is $\sqrt{2}$ that of the standard deviation of each component of $\boldsymbol{\epsilon}_i$, $\hat{\sigma}$. That is, σ and $\hat{\sigma}$ are related by

$$\hat{\sigma} = \frac{\sigma}{\sqrt{2}} \tag{4.5}$$

σ is estimated using Eq(4.3) and GLO. Then Eq(4.5) is used to determine $\hat{\sigma}$.

Now within the GLO framework, consider $x_i = \sum v_x^2$ and $\boldsymbol{\varepsilon}_i = \sigma * \boldsymbol{z}_i$ and define the generator function as

$$g_\sigma(\boldsymbol{\varepsilon}) = \sum (v_p + \sigma * z)^2 \tag{4.6}$$

and the loss function defined as

$$\ell_\sigma = \left(\sum (v_p + \sigma * z)^2 - \sum v_x^2 \right)^2 \tag{4.7}$$

next solve for σ using GLO and calculate $\hat{\sigma}$ using Eq(4.5).

Given the assumption that each component of $\boldsymbol{\epsilon}_i$ have zero mean and equal variance, $\mathcal{N}(0, \hat{\sigma}^2)$, the magnitude of $\boldsymbol{\epsilon}_i$ (i.e., mean ATE) will follow a chi

distribution. Further, the localization error only has two degrees of freedom so it is a special case of the chi distribution, the Rayleigh distribution.[1]

Using the parameter $\hat{\sigma}$ calculated with Eq(4.5) the mean and standard deviation of ATE are calculated using the following properties of a Rayleigh distribution:

$$\begin{aligned} \mu_l &= \hat{\sigma}\sqrt{\frac{\pi}{2}} \\ \sigma_l &= \hat{\sigma}\sqrt{\frac{4-\pi}{2}} \end{aligned} \tag{4.8}$$

To summarize the proposed method for estimating 2D localization error:

1. Run the robot to visit visual markers and collect N measurements of $\boldsymbol{v_p}$ and $\boldsymbol{v_x}$.

2. Solve Eq(4.1) by randomly choosing σ and identifying the σ that minimizes the loss function Eq(4.7).

3. Assuming the 2D localization error follows a Rayleigh distribution, use the estimated parameter to calculate the mean localization error and standard deviation using Eq(4.8).

4.3 Summary

The discussed method uses GLO to determine the 2D mean position error for a localization system using randomly generated samples of the error. First, the robot collects samples of robots localization poses and robot-to-marker poses of visual markers, given multiple visits, and uses them to calculate a constraint on the localization error. Then an optimization method is used to estimate the statistics of the sampling constraint. Lastly, the 2D mean translational error is calculated assuming it follows a Rayleigh distribution.

[1] https://en.wikipedia.org/wiki/Rayleigh_distribution

Chapter 5

Experiments

5.1 Introduction

The proposed method was validated with two different robots using several different localization and SLAM algorithms. The experiments were run indoors and outdoors and used three different localization sensors. Using different robots, sensors, algorithms and navigation areas shows that the method is easy to use. Further, the quantitative results show that the method is accurate for evaluating localization performance indoors. This section first describes the experimental setup, and then presents the results for the proposed method.

5.2 Setup

5.2.1 Localization/SLAM algorithms

Four different open source localization and SLAM systems were used for experiments: AMCL [42], GMapping [18], Google Cartographer [23] and RTAB-Map [29]. The experiments demonstrate that the method can be used to differentiate the performance between several different algorithms. For all the algorithms, experiments were run using existing ROS implementations or wrappers.

The only pure localization method tested was AMCL which is a particle filtering method. GMapping, Cartographer and RTAB-Map are SLAM methods. GMapping is a Rao-Blackwellized particle filtering SLAM algorithm whereas RTAB-Map and Cartographer are graph SLAM methods. For RTAB-Map and

Figure 5.1: Two robots were used for experiments: Clearpath Jackal (right) and Turtlebot2 (left).

Cartographer maps were built first then used to run the algorithms in localization mode. All these methods use laser scan data and odometry, however, RTAB-Map also requires an RGB-D or a stereo camera.

5.2.2 Robots

Two different robots the Clearpath Jackal and the Turtlebot2 were used for experiments. The Turtlebot2 is a two-wheel differential drive robot whereas the Jackal is a four-wheeled skid-steer drive robot. Skid-steer drive robots have more wheel slippage than differential drive robots. Methods to reduce odometry error by accounting for wheel slippage have been developed [2], [26], [33], [44], however, the Jackal robot does not compensate for wheel slippage in its low-level controller. Therefore, it is expected that the odometry error will be larger for the Jackal compared to the Turtlebot and that the Jackal will generally have larger localization error when using the same localization algorithm and sensor combination.

5.2.3 Sensors

The localization performance should depend on the quality of the sensor and should be observed using the presented performance evaluation method. An Astra Orbbec RGB-D Camera (Astra), a Hokuyo UTM-30LX-EW (Hokuyo-UTM) and a Hokuyo URG-04LX (Hokuyo-URG) were used for experiments. Fig. 5.2 shows an image of the three sensors. When mounting the sensors on the robot, the extrinsics were obtained through manual calibration.

The Astra has a recommended depth operating range of 0.6 - 8.0 m and the error and variance of its depth estimates increases quadratically in with range [20]. Also, the narrow horizontal FOV of the Astra at 60°, which is similar to other RGB-D cameras, decreases its mapping and localization performance compared to a laser scanner with a wide FOV [45].

The Hokuyo-URG and Hokuyo-UTM both have a measuring accuracy of 30 mm; however, the Hokuyo-UTM has a larger range at 30 m versus 5.6 m of the Hokuyo-URG. Also, the Hokuyo-UTM has a 270° scan angle with a 0.25° angular resolution versus a 240° scan angle with a 0.35° angular resolution for the Hokuyo-URG. Finally, the Hokuyo-URG sensor has more noise in its depth values compared to the Hokuyo-UTM [35]. Rogers et al. tested several different sensors including the Hokuyo-UTM and Hokuyo-URG using a SLAM method based on GTSAM [11] and found that the Hokuyo-UTM sensor had the lowest localization error [36].

The Hokuyo-URG and Hokuyo-UTM are expected to perform better than the RGB-D camera at localization because they have less measurement noise and larger FOV. Further, the Hokuyo-UTM should have the lowest error, since it has the widest angular range, the lowest measurement noise, and has been shown to perform better than the Hokuyo-URG for SLAM [36].

5.2.4 Navigation Areas

The proposed performance evaluation method was tested in three different areas, with different sizes and complexities. In all the areas AR tags are placed on walls, furniture and boxes to be used as visual markers. By testing in

Figure 5.2: Three localization sensors were used: Hokuyo UTM-30LX-EW (UTM), Hokuyo URG-04LX (URG) and Astra Orbbec RGB-D camera (Astra) (in order left to right).

multiple areas the method is shown to be applicable in different environmental conditions.

The first area, labeled as Room1 (see Fig. 5.3), is an approximately 4 m $\times$ 4 m space with smooth cement floors. Room1 is surrounded by barriers on all sides so that the robot cannot exit the space when performing a random walk. Room1 is small and simple where a random walk navigation strategy can be used for sampling.

The second area, labeled as Room2 (see Fig. 5.4) is approximately 5 m $\times$ 5 m in size (area covered by motion capture system) with tiled floors. Room2 is larger than Room1 and the motion capture area is not as self-contained as in Room1. In Room2 boxes are placed throughout the space to make the area more difficult to perceive and navigate.

The third area, is labeled Outdoor as shown in Fig. 5.4. The outdoor space is an approximately 12 m $\times$ 12 m flat cement walkway between two buildings. AR tags are placed on some of the walls and planters in the area, and also boxes and tripods were used to set AR tags on. Without a way to obtain the ground-truth robot position the outdoors data can only be used for qualitative comparison among different sensors and localization algorithms.

Figure 5.3: Room1 navigation space with visual markers randomly placed on the perimeter. Room1 is approximately 4 by 4 meters with smooth cement floors. Robot to marker poses are sampled and used for evaluating localization performance.

Figure 5.4: Room2 is approximately 5 by 5 meters with tiled floors. AR markers are randomly placed throughout the navigation area. Further boxes are used to make the environment more complex.

Figure 5.5: Outdoor experiment with Jackal robot. Constrained Jackal navigation to a roughly 12×12 meter flat area in between two buildings. Outdoor results have no ground-truth but are used for qualitative evaluation.

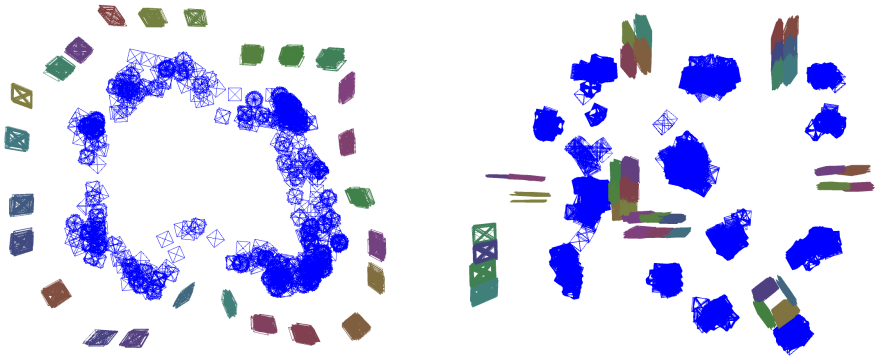
Testing outdoors shows the method is easy to set up and use both indoors and outdoors.

5.2.5 Navigation Strategies

Two navigation strategies were used when collecting data. The first strategy is a simple random walk behavior which was used in Room1. The second navigation strategy was random waypoint navigation. A visualization of the poses collected using both strategies is shown in Fig. 5.6.

The random walk behavior has two parts, first the robot turns a random amount and then drives until it reaches an obstacle. Obstacles are detected using a laser scanner and the robot stops when it is less than a threshold distance away from any obstacle. As mentioned AR tags are placed around the outside of the space so that when the robot approaches an obstacle it can collect robot-to-marker pose samples.

The second behavior is navigation to random waypoints from a set collected before the experiment. The behavior is used in Room2 and Outdoor as these areas are not self-contained like Room1 and more complex. As a result, using waypoint navigation the robot can cover the entire space more efficiently and collect visual marker samples faster. The waypoint navigation is implemented



(a) Random Walk (a) Waypoint Navigation

Figure 5.6: For the experiments two different navigation strategies were used. In Room1 (Fig. 5.3) the robot does a random walk because it is self-contained and the motion capture can cover the entire area. In Room2 (Fig. 5.4) and outdoors (Fig. 5.5) random waypoint navigation was used, as the environments were more complex and open making it harder to use a random walk. See the left image (a) for a visual example of samples collected using the random walk and the right image (b) for samples collected using waypoint navigation. The blue squares with a X are robot pose samples and the other colored squares are marker pose samples.

with the ROS package move_base[1] using AMCL[2] localization and with a map built using GMapping.[3] Before an experiment is run waypoints are collected nearby and facing the visual markers.

5.2.6 Visual Marker Pose Sampling

When a visual marker is in the camera's field of view and within range its pose is sampled, with respect to the AR tag as well as using the motion capture where it is available. The visual markers are placed around the space at roughly the height of the camera. The tool used for computing robot-to-marker poses is ar_track_alvar.[4] With ar_track_alvar only the RGB image from the Astra camera was used for estimating AR Tag pose. During evaluation only the poses of markers less than 0.8 meters from the robot are used to obtain a relative pose of sufficient accuracy for the evaluation method. The range limit

[1]http://wiki.ros.org/move_base
[2]http://wiki.ros.org/amcl
[3]http://wiki.ros.org/GMapping
[4]http://wiki.ros.org/ar_track_alvar

is used because the error in ar_track_alvar as determined experimentally (see Fig 5.8). Fig. 5.7 shows an example of the AR tags used.

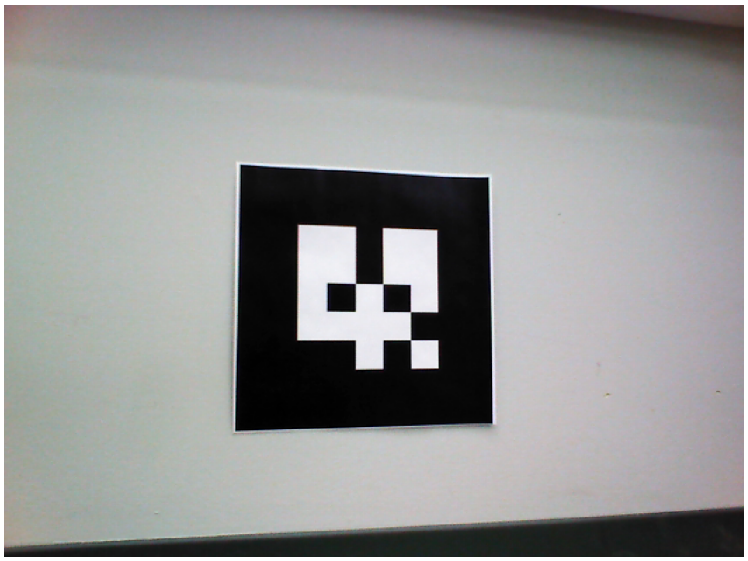


Figure 5.7: Robot's view of visual marker. The marker pose is tracked using ar_track_alvar ROS package.

The error in marker position estimation should be ideally an order of magnitude smaller than the error in the localization system for Eq(4.2) to hold true practically. Using a motion capture system the error in ar_track_alvar position is measured. See Fig. 5.8, which shows the ar_track_alvar tracking error versus distance from the marker for the samples collected. The average tracking error in the robot's sampling range (< 0.8 m) is 1.4 mm $\pm$ 1.5 mm; therefore, the proposed method will not be appropriate for estimating mean translational error below 1.4 cm with the visual markers used for experiments.

5.2.7 Ground-truth

An Optitrack Flex13[5] motion capture system with eight cameras was used to sample the robot's ground-truth trajectory. Optitrack claims the Flex 13 camera has a motion tracking tolerance of 0.5 mm. The motion capture trajectory and localization trajectory are aligned; subsequently, the ground-truth mean translational error can be calculated for evaluating the proposed method.

[5]https://optitrack.com/products/flex-13/

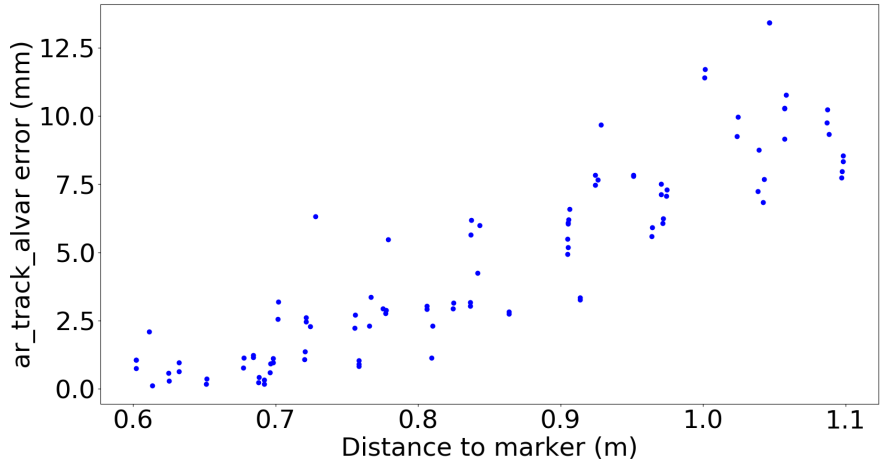


Figure 5.8: The position error in pose estimate using the AR tag estimates as distance from the marker increases. The plot shows how accurately ar_track_alvar can track $\mathbf{X}_1$ and $\mathbf{X}_2$ (see Fig. 4.1) within the range: 0.6 meters to 1.1 meters. Only marker poses less than 0.8 meters from the robot are sampled during experiments. The plot shows that this error increases linearly with the distance to marker within this range. The ground truth robot and marker pose were measured using an Optitrack Flex 13 motion capture system.

5.2.8 Implementation Details

Algorithm 1 and 2 shows pseudocode for the three steps of the proposed method. While step 1 shows an online data collection method, the $(\boldsymbol{v_p}, \boldsymbol{v_x})$ pairs are actually generated offline. Before step 2, 500,000 $(\boldsymbol{v_p}, \boldsymbol{v_x})$ pairs are randomly selected from the set of all pairs to reduce computation time. Next, step 2 is run 50 times on a small batch, each time calculating σ and the mean of those runs is used to calculate $\hat{\sigma}$ in step 3. In step 2 m is a hyperparameter, Bojanowski et al. set m to N, whereas for a similar method Implicit Maximum Likelihood Estimation (IMLE) [31], they recommend setting $m \geq N$. Batches of size 500 and $m = 2000$ were used, after some manual tuning. Also, before running step 2, outlier $(\boldsymbol{v_p}, \boldsymbol{v_x})$ pairs are removed using the difference in length between $\boldsymbol{v_p}$ and $\boldsymbol{v_x}$. The outliers are removed using the 1.5 Interquartile Range (IQR) heuristic, where IQR is the difference between the third quartile (Q_3) and the first quartile (Q_1), and data that is greater than Q_3 + 1.5 * IQR or less than Q_1 - 1.5 * IQR is considered an outlier.

5.3 Results

Table 5.1 and 5.2 show the results for AMCL, GMapping, Cartographer and RTAB-Map. In Room1 only AMCL with the Turtlebot were used. In Room2 all the methods and the Jackal robot used. However, GMapping was not used with the Jackal due to data corruption yet to be corrected at the time of writing. Also, for Cartographer only the UTM and URG sensors are used as Cartographer was not tuned to work reliably on simulated laser scan data from RGB-D.

The results show that the proposed method can estimate within an order of magnitude of the ground-truth error. This is the most significant result as it verifies the feasibility of the proposed method. Further, for most cases the Hokuyo-UTM sensor has the best performance and the RGB-D sensor has the worst performance, as expected. The only exceptional case is the Jackal AMCL data for Room2. Also, there is a noticeable difference between the Turtlebot results (Table. 5.1) and Jackal (Table. 5.2) results as expected due to the Jackal's larger odometry drift.

If the mean translational error does not follow a Rayleigh distribution then the proposed method fails. In the result tables, there are three outliers marked by an asterisk: Room2 Jackal Cartographer with the Hokuyo-URG and both rows involving the Hokuyo sensors for Room2 Turtlebot RTAB-Map. These outliers rows have a much larger absolute error than the other rows, showing that although this method still produces useful results, it fails to estimate the error with the same accuracy. To understand the underlying cause of these cases, in Fig. 5.10 (d-f) the histograms for the three outlier rows are shown. Fig. 5.10 also plots a Rayleigh distribution estimated probability density function (PDF) and a Kernel Density Estimate (KDE) PDF fitted to the data.

If the translational error follows a Rayleigh distribution then the PDF found using the non-parametric estimation method (KDE) should roughly match the PDF fitted using the Rayleigh distribution. For Fig. 5.10 (d-f) it is clear that the Rayleigh and KDE PDFs do not match, and this is most

Table 5.1: Turtlebot: Position error estimation in mm using the proposed method. Compared with motion capture to create ground-truth. With different algorithm and sensor combinations there should be different localization performance. The sensors used are: RGB-D (Astra Orbbec), Hokuyo-URG (Hokuyo URG-04LX), and Hokuyo-UTM (Hokuyo UTM-30LX-EW) on a Turtlebot. The Ground-truth column shows the mean and standard deviation calculated using position errors measured with the motion capture system. The Ours column shows the estimated mean (μ_l) and standard deviation (σ_l) of the localization error calculated using the proposed method. The final column shows the absolute value of the difference between the Ground-truth and Ours in mm.

Turtlebot: Comparison of Absolute Position Error in mm				
Room 1				
Localization Method	Sensor	Ground-truth (mm)	Ours ($\mu_l \pm \sigma_l$) (mm)	Abs. error (mm)
AMCL	Hokuyo-UTM	43 ± 19	43 ± 23	0
	Hokuyo-URG	45 ± 22	42 ± 22	3
	RGB-D	81 ± 38	79 ± 41	2
Room 2				
Localization Method	Sensor	Ground-truth (mm)	Ours ($\mu_l \pm \sigma_l$) (mm)	Abs. error (mm)
AMCL	Hokuyo-UTM	33 ± 16	33 ± 17	0
	Hokuyo-URG	39 ± 21	39 ± 20	0
	RGB-D	53 ± 25	48 ± 25	5
GMapping	Hokuyo-UTM	17 ± 13	24 ± 13	7
	Hokuyo-URG	29 ± 16	35 ± 18	6
	RGB-D	163 ± 74	165 ± 86	2
Cartographer	Hokuyo-UTM	11 ± 6	15 ± 8	4
	Hokuyo-URG	34 ± 20	35 ± 18	0
RTAB-Map	Hokuyo-UTM	14 ± 49	23 ± 12	9
	Hokuyo-URG	41 ± 200	22 ± 12	19*
	RGB-D	272 ± 298	170 ± 89	102*

* Outlier results where error does not follow Rayleigh distribution. See Section 5.3 for further discussion.

likely the reason for the poor estimates shown in the table. Specifically, for the outlier examples mentioned a translational error versus time plot like Fig. 5.9 (b) is typical, which shows an example of data with many outliers. When there

Table 5.2: Jackal: Position error estimation in mm using the proposed method. Compared with motion capture to create ground-truth. With different algorithm and sensor combinations there should be different localization performance. The sensors used are: RGB-D (Astra Orbbec), Hokuyo-URG (Hokuyo URG-04LX), and Hokuyo-UTM (Hokuyo UTM-30LX-EW) on a Jackal robot. The Ground-truth column shows the mean and standard deviation calculated using position errors measured with the motion capture system. The Ours column shows the estimated mean (μ_l) and standard deviation (σ_l) of the localization error calculated using the proposed method. The final column shows the absolute value of the difference between the Ground-truth and Ours in mm.

Jackal: Comparison of Absolute Position Error in mm				
Room 2				
Localization Method	Sensor	Ground-truth (mm)	Ours ($\mu_l \pm \sigma_l$) (mm)	Abs. error (mm)
AMCL	Hokuyo-UTM	90 ± 39	100 ± 52	10
	Hokuyo-URG	96 ± 43	98 ± 51	2
	RGB-D	80 ± 34	81 ± 42	1
Cartographer	Hokuyo-UTM	53± 38	62 ± 32	9
	Hokuyo-URG	137 ± 87	103 ± 54	33*
RTAB-Map	Hokuyo-UTM	35 ± 68	32 ± 17	3
	Hokuyo-URG	43 ± 29	47 ± 25	4
	RGB-D	206 ± 100	208 ± 109	2

* Outlier results where error does not follow Rayleigh distribution. See Section 5.3 for further discussion.

are many outliers this causes the error to become multimodal and breaks the Rayleigh distribution assumption. Ideally, an additional step could be added to the method where the Rayleigh distribution assumption can be checked, however, this is difficult to verify without being able to directly sample the error (eg. via ground-truth methods) and is a limitation of the proposed method.

Table 5.3 shows the results from running the Jackal robot outdoors. In this case, only AMCL was used on the data; both Cartographer and RTAB-Map experienced serious map drift over time and losing track of the correct position, to yield useful experimental data. GMapping is not used on the outdoor Jackal

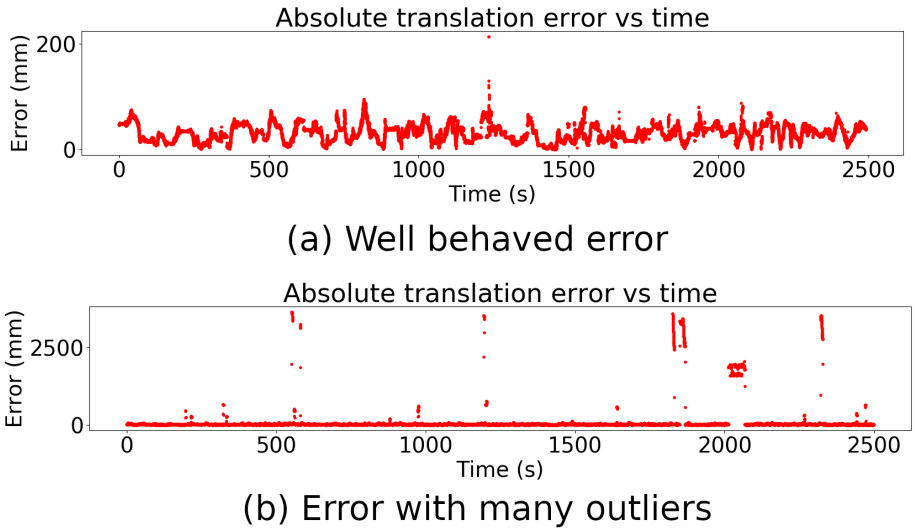


Figure 5.9: Graphical comparison of well-behaved error data (top) versus data with outliers (bottom). These plots show the ground-truth translational error versus time. For the proposed method to work it is assumed that the data is unimodal, if the data has a lot of outliers (bottom) then this assumption breaks.

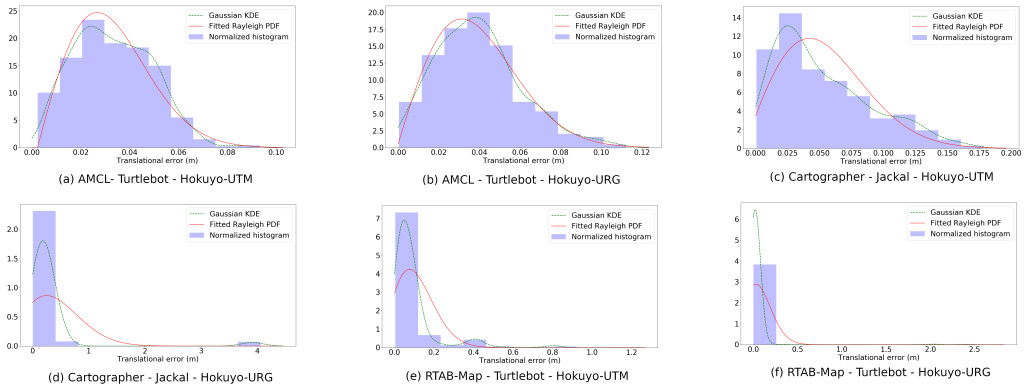


Figure 5.10: Normalized histograms of ground-truth translational error for a selection of Room2 experiments. The red lines are Rayleigh probability density functions (PDF) fitted to the ground-truth error. The green lines are PDFs found using non-parametric Kernel Density Estimation (KDE) with a Gaussian Kernel.

data for the same reason that it was not used on the indoor Jackal data.

Table 5.3: The estimated localization error for AMCL in millimeters, where the data was collected on a Jackal robot. The outdoor estimates should be consistent with the indoor results. The sensors used are: RGB-D (Astra Orbbec), Hokuyo-URG (Hokuyo URG-04LX), and Hokuyo-UTM (Hokuyo UTM-30LX-EW).

Outdoor - Jackal		
	RGB-D	Ours
AMCL	Hokuyo-UTM	70 ± 37
	Hokuyo-URG	97 ± 51
	RGB-D	189 ± 99

The outdoor AMCL results are as expected showing the most error for the RGB-D sensor and the least error for the Hokuyo-UTM. Also, there is a bigger than normal difference between the Hokuyo-UTM and Hokuyo-URG, which is expected given that the max range for the URG of 5.6 m, versus that of the Hokuyo-UTM at 30 m. In a larger environment this is a big advantage. Qualitatively the difference between the indoor and outdoor results for the RGB-D sensor are convincing as the localization error is over two times larger than the indoors. This large difference is expected as the Astra Orbbec's depth measurement performs poorly outside with sunlight. However, the Hokuyo-URG and Hokuyo-UTM results are less intuitive, as there is lower error outdoors than indoors for these sensors. This result is likely because outdoors in a large space the Jackal has to turn less nearby markers and can approach the waypoints in straight lines. Because of the Jackal skid-steer drive wheel slippage turning less often means smaller mean translational error.

5.4 Summary

This section demonstrated that the proposed method is easy to use and set up by testing in multiple environments and with multiple robot and sensor combinations. Also, because it only requires visual markers place around the navigation space, such as AR tags, it is inexpensive. Finally, the results show that the proposed method can estimate the mean translational error within

an order of magnitude of the error measured by a motion-capture system.

Chapter 6

Conclusion

This thesis presents a method for evaluating robot localization performance that is accurate, inexpensive and easy to setup. Chapter 2 gave a brief review of Markov Localization and SLAM then summarizes the four localization methods used for experiments. AMCL is a particle filter algorithm that uses laser scan data and requires an existing occupancy grid map. GMapping is an RBPF SLAM algorithm that builds occupancy grid maps with a laser scanner. RTAB-map and Cartographer are both graph-based SLAM systems, but Cartographer requires laser scan data, whereas RTAB-map requires 3D vision but can also use laser scan data as additional input.

Chapter 3 reviewed existing ground-truth methods and robot localization evaluation metrics. Trajectory error is a commonly used metric as it provides a concise measure of performance. There are two types of trajectory error RPE which is a measure of local error between poses and ATE which measure the global error in the map frame. Metrics measuring robustness, computational complexity and map quality are also sometimes used to evaluate localization and SLAM systems. Ground-truth methods for estimating localization performance include motion-capture, GPS and IMU sensor fusion, simulation and external marker systems.

Chapter 4 described the proposed method for estimating mean translational error for a localization system. Experiments show that the method is easy to use and set up by testing in multiple environments with multiple robot and sensor combinations. The experiments were run with Clearpath Jackal and

Turtlebot2 robots using either the Astra Orbbec RGB-D, Hokuyo-UTM laser rangefinder or Hokuyo-URG laser rangefinder for a localization sensor. Also, the method is inexpensive as it only requires visual markers (eg. AR tags) placed around the navigation space. The error estimation algorithm has three steps, first robot and marker pose pairs are collected from multiple visits to the same marker. Second, the samples are used to calculate constraints on the localization error and used in an optimization method for estimating the unknown parameter. Finally, the localization error is calculated assuming it follows a Rayleigh distribution.

Chapter 5 provides experimental results on the proposed method's accuracy. The results show that the method can estimate the mean translational error within an order of magnitude of the error measured by a motion-capture system.

Ground-truth systems are expensive and hard to setup. While datasets paired with ground-truth data make it easy to compare algorithms, the dataset might not contain all the conditions and environments necessary to fully test a localization system. The method presented in this thesis provides a cheaper alternative to ground-truth systems that is easy to set up and use in new environments.

A limitation of the proposed method is that it relies on the mean ATE to follow a Rayleigh distribution, therefore future work will include developing an outlier rejection technique that ensures the mean ATE will always follow a Rayleigh distribution.

The outdoor experiments presented are very preliminary, and future work will also include running additional experiments outdoors. These experiments will include testing multiple SLAM algorithms and using a ground-truth system to quantitatively evaluate the method outdoors.

Finally, while artificial visual markers are generally easy to use they still require some setup and camera calibration. An extension of the proposed method would be to use visual landmarks inherent in the scene. Therefore future work will include testing a method that does not need artificial landmarks but relies on natural visual landmarks.

References

[1] S. Agarwal, K. Mierle, and Others, *Ceres solver*, http://ceres-solver.org. 13

[2] G. Anousaki and K. J. Kyriakopoulos, "A dead-reckoning scheme for skid-steered vehicles in outdoor environments," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 1, 2004, 580–585 Vol.1. DOI: 10.1109/ROBOT.2004.1307211. 36

[3] B. Bodin, H. Wagstaff, S. Saeedi, L. Nardi, E. Vespa, J. Mayer, A. Nisbet, M. Luján, S. Furber, A. Davison, P. Kelly, and M. O'Boyle, "Slambench2: Multi-objective head-to-head benchmarking for visual slam," in *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, 2018. 19

[4] P. Bojanowski, A. Joulin, D. Lopez-Pas, and A. Szlam, "Optimizing the latent space of generative networks," in *Proceedings of the 35th International Conference on Machine Learning*, J. Dy and A. Krause, Eds., ser. Proceedings of Machine Learning Research, vol. 80, Stockholmsmässan, Stockholm Sweden: PMLR, 2018, pp. 600–609. [Online]. Available: http://proceedings.mlr.press/v80/bojanowski18a.html. 28, 29, 31

[5] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. DOI: 10.1177/0278364915620033. eprint: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract. 17, 22, 23

[6] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *Trans. Rob.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2624754. [Online]. Available: https://doi.org/10.1109/TRO.2016.2624754. 8–10

[7] S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei, "Rawseeds ground truth collection systems for indoor self-localization and mapping," *Auton. Robots*,

vol. 27, no. 4, pp. 353–371, Nov. 2009, ISSN: 0929-5593. DOI: 10.1007/s10514-009-9156-5. [Online]. Available: http://dx.doi.org/10.1007/s10514-009-9156-5. 24

[8] J. Clausen, *Branch and bound algorithms – principles and examples*, 1999. 14

[9] A. Doucet, N. d. Freitas, K. P. Murphy, and S. J. Russell, "Rao-blackwellised particle filtering for dynamic bayesian networks," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, ser. UAI '00, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2000, pp. 176–183, ISBN: 1-55860-709-9. [Online]. Available: http://dl.acm.org/citation.cfm?id=647234.720075. 14

[10] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part i," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006, ISSN: 1070-9932. DOI: 10.1109/MRA.2006.1638022. 9

[11] *Factor graphs and gtsam: A hands-on introduction*, 2012. [Online]. Available: https://smartech.gatech.edu/handle/1853/45226. 13, 37

[12] A. Filatov, A. Filatov, K. Krinkin, B. Chen, and D. Molodan, "2d slam quality evaluation methods," in *Proceedings of the 21st Conference of Open Innovations Association FRUCT*, ser. FRUCT'21, Helsinki, Finland: FRUCT Oy, 2017, pp. 120–126. DOI: 10.23919/FRUCT.2017.8250173. [Online]. Available: https://doi.org/10.23919/FRUCT.2017.8250173. 20

[13] D. Fox, "Kld-sampling: Adaptive particle filters," in *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, ser. NIPS'01, Vancouver, British Columbia, Canada: MIT Press, 2001, pp. 713–720. [Online]. Available: http://dl.acm.org/citation.cfm?id=2980539.2980632. 10

[14] A Geiger, P Lenz, C Stiller, and R Urtasun, "Vision meets robotics: The kitti dataset," *Int. J. Rob. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013, ISSN: 0278-3649. DOI: 10.1177/0278364913491297. [Online]. Available: http://dx.doi.org/10.1177/0278364913491297. 21

[15] A. Geiger, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, ser. CVPR '12, Washington, DC, USA: IEEE Computer Society, 2012, pp. 3354–3361, ISBN: 978-1-4673-1226-4. [Online]. Available: http://dl.acm.org/citation.cfm?id=2354409.2354978. 17, 20–22

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2014, pp. 2672–2680. [Online]. Available: http:

//papers.nips.cc/paper/5423-generative-adversarial-nets.pdf. 31

[17] G. Grisetti, S. Grzonka, C. Stachniss, P. Pfaff, and W. Burgard, "Efficient estimation of accurate maximum likelihood maps in 3d," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3472–3478. DOI: 10.1109/IROS.2007.4399030. 13

[18] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE Transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007, ISSN: 1552-3098. DOI: 10.1109/TRO.2006.889486. 10, 15, 35

[19] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, "A tutorial on graph-based slam," *IEEE Intelligent Transportation Systems Magazine*, vol. 2, pp. 31–43, 2010. 10

[20] G. Halmetschlager-Funek, M. Suchi, M. Kampel, and M. Vincze, "An empirical evaluation of ten depth cameras: Bias, precision, lateral noise, different lighting conditions and materials, and multiple sensor setups in indoor environments," *IEEE Robotics Automation Magazine*, vol. 26, no. 1, pp. 67–77, 2019, ISSN: 1070-9932. DOI: 10.1109/MRA.2018.2852795. 37

[21] A. Handa, T. Whelan, J. McDonald, and A. Davison, "A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM," in *IEEE Intl. Conf. on Robotics and Automation, ICRA*, Hong Kong, China, 2014. 25

[22] W. Hess, D. Kohler, H. Rapp, and D. Andor, *Google cartographer documentation: Algorithm walkthrough for tuning*, https://google-cartographer-ros.readthedocs.io/en/latest/algo_walkthrough.html. 13, 14

[23] ——, "Real-time loop closure in 2d lidar slam," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278. DOI: 10.1109/ICRA.2016.7487258. 10, 13, 35

[24] R. R. L. Jr, *Kalman-and-bayesian-filters-in-python*, https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python. 6, 8

[25] H. Kikkeri, G. Parent, M. Jalobeanu, and S. Birchfield, "An inexpensive method for evaluating the localization performance of a mobile robot navigation system," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4100–4107. DOI: 10.1109/ICRA.2014.6907455. 19, 25, 26

[26] K. Kozłowski and D. Pazderski, "Modeling and control of a 4-wheel skid-steering mobile robot," *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 4, 477–496, 2004, ISSN: 1641-876X. 36

[27] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, p. 387, 2009, ISSN: 1573-7527. DOI: 10.1007/s10514-009-9155-6. [Online]. Available: https://doi.org/10.1007/s10514-009-9155-6. 18, 20

[28] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3607–3613. DOI: 10.1109/ICRA.2011.5979949. 13

[29] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019. DOI: 10.1002/rob.21831. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/rob.21831. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21831. 10–12, 35

[30] J. J. Leonard and H. F. Durrant-Whyte, "Mobile robot localization by tracking geometric beacons," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 376–382, 1991, ISSN: 1042-296X. DOI: 10.1109/70.88147. 1

[31] K. Li and J. Malik, "Implicit maximum likelihood estimation," *CoRR*, vol. abs/1809.09087, 2018. arXiv: 1809.09087. [Online]. Available: http://arxiv.org/abs/1809.09087. 44

[32] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 Year, 1000km: The Oxford RobotCar Dataset," *The International Journal of Robotics Research (IJRR)*, vol. 36, no. 1, pp. 3–15, 2017. DOI: 10.1177/0278364916679498. eprint: http://ijr.sagepub.com/content/early/2016/11/28/0278364916679498.full.pdf+html. [Online]. Available: http://dx.doi.org/10.1177/0278364916679498. 17, 20, 21

[33] A. Mandow, J. L. Martinez, J. Morales, J. L. Blanco, A. Garcia-Cerezo, and J. Gonzalez, "Experimental kinematics for wheeled skid-steer mobile robots," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 1222–1227. DOI: 10.1109/IROS.2007.4399139. 36

[34] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An open-source SLAM system for monocular, stereo and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017. DOI: 10.1109/TRO.2017.2705103. 9, 12

[35] F. Pomerleau, A. Breitenmoser, M. Liu, F. Colas, and R. Siegwart, "Noise characterization of depth sensors for surface inspections," in *2012 2nd International Conference on Applied Robotics for the Power Industry (CARPI)*, 2012, pp. 16–21. DOI: 10.1109/CARPI.2012.6473358. 37

[36] J. G. Rogers, A. J. B. Trevor, C. Nieto-Granda, A. Cunningham, M. Paluri, N. Michael, F. Dellaert, H. I. Christensen, and V. Kumar, "Effects of sensory precision on mobile robot localization and mapping," in *Experimental Robotics: The 12th International Symposium on Experimental Robotics*, O. Khatib, V. Kumar, and G. Sukhatme, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 433–446, ISBN: 978-3-642-28572-1. DOI: 10.1007/978-3-642-28572-1_30. [Online]. Available: https://doi.org/10.1007/978-3-642-28572-1_30. 37

[37] J. Röwekämper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard, "On the position accuracy of mobile robot localization based on particle filters combined with scan matching," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012, pp. 3158–3164. DOI: 10.1109/IROS.2012.6385988. 22, 24, 25

[38] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, "Introduction to autonomous mobile robots," in *Introduction to Autonomous Mobile Robots*. MITP, 2011, ISBN: 9780262295321. [Online]. Available: https://ieeexplore.ieee.org/document/6277373. 1, 2

[39] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman, "The new college vision and laser data set," *Int. J. Rob. Res.*, vol. 28, no. 5, pp. 595–599, May 2009, ISSN: 0278-3649. DOI: 10.1177/0278364909103911. [Online]. Available: http://dx.doi.org/10.1177/0278364909103911. 17, 20, 22

[40] C. Sprunk, J. Röwekämper, L. Spinello, G. D. Tipaldi, W. Burgard, M. Jalobeanu, and G. Parent, "An experimental protocol for benchmarking robotic indoor navigation," International Symposium on Experimental Robotics, 2014. [Online]. Available: https://www.microsoft.com/en-us/research/publication/an-experimental-protocol-for-benchmarking-robotic-indoor-navigation/. 19

[41] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of rgb-d slam systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, 2012. 17, 18, 22, 23, 25

[42] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005, ISBN: 0262201623. 1, 6, 8, 10, 11, 35

[43] C. H. Tong and T. D. Barfoot, "A self-calibrating 3d ground-truth localization system using retroreflective landmarks," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 3601–3606. DOI: 10.1109/ICRA.2011.5979613. 25

[44] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation," *IEEE Transactions on Robotics*, vol. 25, no. 5, pp. 1087–1097, 2009, ISSN: 1552-3098. DOI: 10.1109/TRO.2009.2026506. 36

[45] S. Zug, F. Penzlin, A. Dietrich, T. T. Nguyen, and S. Albert, "Are laser scanners replaceable by kinect sensors in robotic applications?" In *2012 IEEE International Symposium on Robotic and Sensors Environments Proceedings*, 2012, pp. 144–149. DOI: 10.1109/ROSE.2012.6402619. 37