An Examination of Localization Techniques Used on ABBY

In robotics, localization is the task of determining where the robot is. Localization methods can be classified into one of two groups. Relative localization methods determine the robot’s location with respect to the robot’s previous location, and absolute localization methods determine the robot’s location with respect to an absolute location reference in the robot’s environment, sometimes called the “ground truth.” There are advantages and disadvantages to both classes of localization methods, and ABBY’s localization uses a combination of relative and absolute localization.

Relative localization methods have several advantages over absolute localization methods. Relative localization methods tend to be computationally simple. Odometry, for instance, can be accomplished on a two-wheeled differential drive robot in only 14 mathematical operations (7 addition, 4 multiplication, and 3 trigonometric). Because relative localization is so computationally simple, it allows for high frequency update rates and implementation on embedded processors or in logic circuitry. Relative localization requires no knowledge of the robot’s environment (such as a map), and it does not require the robot’s environment to be instrumented with sensors (to track the robot) or beacons (for the robot to track). Localization methods can also be accomplished with relatively cheap sensors such as inertial measurement units (IMUs), optical flow sensors, and (in the case of wheeled vehicles) shaft encoders.

Odometry Algorithm

However, relative localization methods all share one major disadvantage. Because each localization update is performed with respect to the previous, error accumulates over time. The source of the error varies from method to method. In the case of wheeled odometry, errors are usually due to wheel slip and the non-linearity of the trigonometric function used to estimate the heading. In the case of IMUs, the error is usually a result of the slow drift of the accelerometers and gyroscopes due to temperature variation. A well-calibrated relative localization system with an accurate observer model will still accumulate error over time, and the estimated position of the robot will slowly diverge from the true position.

There are several different types of absolute localization methods that use different types of sensors. Some methods use external sensors, such as cameras radio frequency (RF) tracking systems to monitor the robot’s position. In order for a robot to use these methods, the operating environment must have already been instrumented with the necessary sensors. Other methods use sensors on the robot to track features of the robot’s environment and compare them to a known map of the robot’s environment. Trackable features include visible features such as lights, signs, or painted patterns; these features may already exist in the environment, such as ceiling lights in an office building, or may be added, such as position-coded labels on a warehouse floor. Trackable features could also be RF beacons, which may already exist (such as WiFi access points), or may be installed specifically for localization. The global positioning system (GPS) is an example of an RF localization system that uses time-of-flight from satellite radio beacons to triangulate the position of the receiver. Systems with ranging sensors can track features of the geometry of the environment itself. To perform an update, all of these localization methods compare sensor data to a map or model of the environment to estimate the robot’s position. This means that they are dependent on an accurate map or model, which may not be possible in an environment with changing features. These methods also depend on the environment having suitable features to localize against. It is difficult to use these methods in featureless environments, such as open fields, and environments with many repeated similar or identical features, such as long hallways.

Although most absolute localization methods require an a priori map (or model) of the environment, one class of absolute localization methods perform simultaneous localization and mapping (SLAM). SLAM algorithms begin with no map of the robot’s environment and incrementally build one as they explore. For example, a SLAM algorithm using a LIDAR can exploit partially-overlapping LIDAR scans to register new scans with respect to previous scans. The translation and rotation required to register the scan can be used to determine the robot’s position with respect to the previous position. Each new scan increases the known region of the environment, slowly building a map. Because SLAM algorithms are incremental, they are more prone to error than localizing against an a priori map or model. In the LIDAR SLAM example, each new LIDAR scan must have enough overlap with the existing map to perform registration. This assumption may be violated if the robot moves too quickly or the geometry of the robot’s environment limits the field of view of the sensor. Once errors are introduced to the robot’s map of the environment, they may be difficult or impossible to correct. Although SLAM methods will usually out-perform relative localization methods, they are less reliable than absolute localization against an a priori map or model. (CITATION)

ABBY’s sensor suite contains several sensors that can be used for localization. The encoders on the wheels are well-suited for odometry, and the gyroscopic yaw-rate sensor can be used for inertial heading measurement. The LIDAR and the Kinect depth camera can both be used for absolute localization using a number of methods, including both SLAM and a priori map localization algorithms. In addition, the Kinect camera could be used for localization based on patterns on the floor. Of these possible methods, odometry was chosen for relative localization, and Adaptive Monte Carlo Localization (AMCL) using LIDAR scans and an a priori 2D occupancy grid map was chosen for absolute localization.

ABBY uses odometry from the encoders on the wheels for relative localization. The physical state observer runs under the real time operating system on the cRIO, and publishes pose estimates with uncertainty (represented as covariance) to the ROS system at FREQUENCY Hz. Relatively high frequency updates to the robot’s pose are required as an input to the local planner, which generates velocity pairs at a rate of FREQUENCY Hz. The odometry system does accumulate error over time. This error was characterized by operating the robot using only the relative localization system and manually driving it along simple geometric paths. When the robot is driven along a ten meter straight line with no observed wheel slip, the odometry error is RESULTS. When the robot is driven in a circle with radius 1m for five laps with no observed wheel slip, the odometry error is RESULTS. As expected, change in heading causes a greater error in the odometric localization. The greater error due to change in heading is a feature of the differential drive system on ABBY. For a differential drive system to turn, one or both wheels must slip, which introduces error into the odometry. In addition, rapid acceleration or deceleration of the robot causes the wheels to slip as the force exerted by the wheels exceeds the static friction between the wheels and the floor. This, in turn, introduces a sudden, relatively large error to the odometry. To limit problems due to wheel slip, the acceleration of the robot was software-limited to below the wheel-slip threshold of (WHEEL SLIP ACCELERATIONS).

It may be possible to improve the accuracy of the relative localization system by fusing in other sensors. The yaw rate sensor on the robot was not used because undiagnosed electrical problems rendered it inaccurate. Other researchers at Case (CITE Perko), have fused rotational velocity data from a gyroscopic yaw rate sensor with odometry using an Extended Kalman Filter (EKF) (CITE EKF). Because the yaw rate sensor is inertial, it is not affected by wheel slip. However, it does drift slowly over time. An EKF with variable measurement covariance, as described in (CITE Perko) can improve the relative pose estimate, but the author of that research concluded that the improvement was minimal.

AMCL is an absolute localization method that model's the robot's pose as a probability distribution (CITE Probabilistic Robotics). The robot's pose is considered probabilistic to represent the uncertainty of the sensor measurements used to determine the pose. ABBY uses AMCL to match LIDAR scans to an *a priori* map. On each update, the previous pose distribution is taken as the Bayesian prior and a new measurement is incorporated to calculate a posterior pose estimate. AMCL represents the (continuous) probability distribution of the pose in discrete space with a particle filter using KLD sampling, which adapts the number of sampled points in the distribution based on the covariance of the distribution. As the robot becomes more sure of its location, fewer particles are needed to accurately represent the distribution. In addition, random particles are added to help break the filter out of a false convergence. By starting with particles uniformly distributed through the map, AMCL can theoretically solve the “wake-up robot problem,” in which the robot that is initialized with no estimated pose. However, testing with ABBY showed that AMCL could not reliably solve this problem, and would often converge on a false pose estimate. Instead, ABBY is initialized to a pose at or near the true pose, with sufficiently large covariance that the true pose is within the likely region of the estimated pose. As the robot runs, the pose estimate will converge on the true pose. Each pose update from AMCL take approximately TIME milliseconds, which means that it can run no faster than FREQUENCY Hz.

Of course, AMCL is only possible with an a priori map. ABBY's maps were generated by the robot itself, using the same LIDAR used for AMCL and the gmapping SLAM package. Gmapping uses a Rao-Blackwellized particle filter to generate maps as it localizes. Gmapping was considered as a possibility for the absolute localization scheme on ABBY. The main advantage of gmapping over AMCL is that it does not require an a priori map, making it easer to install the robot in a novel environment. However, gmapping on ABBY was unable to reliably traverse doorways without accumulating error. This was sufficient reason not to use it for localization. In addition, the maps generated with gmapping were manually edited to remove skews from going through doors.

Combining odometry with AMCL yields results that are better than either one alone. Because AMCL takes so long to compute, it cannot be used to approximate continuous localization. This makes it unsuitable for local planning, which updates at FREQUENCY. Whereas odometry is more suitable for the local planner, the error it accumulates as the robot runs eventually makes it unsuitable for global planning. In order to combine these two methods, two transforms are stored in the robot’s TF tree. One transform is between the robot’s base\_link (a coordinate frame with its origin on the floor between the robot’s wheels) to its parent, the odometric frame odom. This transform is updated by the odometric state estimator on the cRIO at FREQUENCY, and provides an approximately continuous position estimate. Local planning is performed in the odom frame. The top level transform is from the map frame (the absolute coordinate system) to the odom frame and the map frame. This transform is updated by AMCL, which runs an update every time the robot moves more than 0.05 meters in translation or 0.1 radians in rotation. On each AMCL update, the transform from the map to odom frame is change such that it cancels out any error in the tranform from the odom frame to the robot's base\_link. Over a long period of operation the odom transform may accumulate significant error, but the transform from the map frame to the base\_link remains accurate because of the absolute localization updates.

(FIGURE the tf tree from /map to /base\_link with robot model)

Some ways to improve the localization on this robot were beyond the scope of this thesis due to hardware or time limitations. An IMU could have made significant improvements to the relative localization by helping to counteract errors due to wheel slip. Electrical problems with the yaw rate sensor made it unusable for this project, and the robot was not outfitted with accelerometers. However, the recent affordability of six degree of freedom single-chip IMUs (CITE cheap IMU) would make this an excellent avenue of research to improve the localization system. One obstacle to pursuing this route is that it would require rewriting and retuning the localization EKF on the cRIO, which is beyond the scope of this thesis.

Similarly, optical flow sensors looking at the ground might be used to mitigate wheel slip error. Optical flow sensors bounce light off of a surface and measure how quickly that surface is moving relative to the sensor. A pair of optical flow sensors, mounted near the drive wheels, could supplement the odometry. Because they are unaffected by wheel slip, they might prove more accurate than the existing odometry system. (READ SOME STUFF ABOUT OPTICAL FLOW HERE AND EXPLAIN WHY YOU DIDN’T USE THEM)

Another way to potentially improve the robot’s localization would be to use the Kinect. The Kinect’s limited range and field of view (figure) makes it relatively useless for localization using building geometry such as AMCL or gmapping. However, because the Kinect is facing the floor, it could be used for absolute localization based on patterns on the floor. If the floor is marked with landmarks in the form of painted symbols or codes, the Kinect could be used to detect a landmark and localize the robot with respect to the known location of the landmark in the map. This technique has been used in other industrial mobile robots (CITE Kiva). A system of visual landmarks on the floor might be useful as a starting seed pose for AMCL. Currently, when the robot is started, the starting pose must be manually entered using Rviz or a launch script. With visual landmarks on the floor, the robot could be started at any arbitrary point in the environment, provided a landmark was visible to the Kinect, and the robot could generate a starting pose estimate from the landmark.

(FIGURE floor map with landmark symbols)