Chapter 9
Localization and Mapping Corrections

In the last chapter, the main focus was on inertial body reference, *idiothetic* sensors, which provide *internal information* about the subject's movements. In this chapter, we add information from local reference, or *allothetic* sensors, which provide *external information* about the environment. In Chaps. 2–7, there has been much discussion of RF ranging sensors which are a type of allothetic sensor providing ranging to fixed beacons or other tracked personnel/platforms. Another example of a common local reference sensor is an image sensor. Even inertial sensors, which are typically used as body reference sensors, can provide local map reference data by inferring the location of terrain features based on the sensor data (Funk et al. 2007; Bandyopadhyay et al. 2008). These allothetic sensors allow us to create a feature map of what is around us and to locate ourselves within that map—localization and mapping.

In this chapter, we review some different allothetic sensors and the types of features that can be extracted for localization. The ability to extract unique features that can be recognized when "seen" again is the basis for creating feature maps that can be used to aid in localization. Next, the theoretical formulation and common solution approaches for the localization and mapping problem are reviewed. Finally, an example is given that addresses some of the practical issues for implementing localization and mapping solutions.

9.1 Localization and Mapping Overview

The goal of localization and mapping is to compute the most probable observer location within the discovered map given the past sensor and control values (if available). Called *simultaneous localization and mapping* (SLAM), SLAM requires the use of sensors to construct a geometric or topological map of the environment and then use that map for localization (Smith and Cheeseman 1986; Durrant–Whyte 1988; Smith et al. 1990; Dissanayake et al. 2001; Guivant and Nebot 2001; Montemerlo et al. 2003a; Montemerlo and Thrun 2003b; Thrun et al. 2006).

The map information also enables us to constrain the growth of errors in body reference sensor systems. The ability to constrain the errors is dependent on the quality the idiothetic and allothetic sensors.

In SLAM, both the trajectory of the observer—positions, velocities, and headings (etc.)—together with features of the map are estimated online without the requirement for any a priori knowledge of location. Although, navigation and mapping systems may have access to pre-existing map data. This map data might consist of GIS (geographic information system) shape files (including building outlines, roads, etc.), satellite imagery, elevation maps, and building maps (CAD files, floor plans, etc.). This existing map information can be used for to refine SLAM algorithms results where map data exists while still allowing new features that are discovered to be included in the global map.

Work by Meyer and Filliat (Filliat and Meyer 2003; Meyer and Filliat 2003) provides a useful summary of map-based navigation, which involves three processes:

- **Map-learning**—the process of transforming the data acquired during exploration to a suitable representation and structure constituting a map.
- **Localization**—the process of deriving the current position within the map.
- **Path-planning**—the process of choosing a course of actions to reach a goal, given the current position and map.

Localization and map-learning are interdependent processes; the positions of tracked entities and discovered features/landmarks are estimated relative to the currently known map. On the other hand, path-planning is a somewhat independent process that takes place once the map has been built and the subject's position estimated.

These three processes may rely on both idiothetic and allothetic sensor data. Idiothetic information may include speed, acceleration, leg movement for dismounts, wheel rotation for vehicles, etc. Through dead reckoning, these data provide position estimates of the subject in a metric space. Idiothetic sensors can also provide local map reference data by inferring the location of terrain features based on how the subject moves through the environment. For example, they have been effectively used to locate features in structured environments such as stairways and elevators in buildings (Funk et al. 2007; Bandyopadhyay et al. 2008).

Allothetic information can be used to directly recognize a place or a situation; in this case, any cue such as image features, sonar time-of-flight, color, etc., may be used. Allothetic information can also be used to derive subject motion from measurements of the environment. That is accomplished by converting information expressed in the space related to the idiothetic data based on metric models of the associated sensors. With such a metric model, it is possible to infer the relative positions of two places in which allothetic information has been gathered (Filliat and Meyer 2003). For example, frame-to-frame stereo camera feature tracking can be used to solve for six degrees of freedom motion of the camera (see Chap. 8).

The limitations and advantages of these two sources of information are complementary. Indeed, the main problem associated with the derived metric motion information is that, because it involves a dead reckoning process, it is subject to ***cumulative error*** (for example, heading error in an inertial system). This leads to a continuous decrease in quality; therefore, such information cannot be trusted over long periods of time. On the contrary, the quality of feature based map information is constant over time, but it suffers from the ***perceptual aliasing problem***, e.g., for a given sensor system, two distinct places (landmarks) in the environment may appear the same, for example, doors or light fixtures.

Consequently, to build reliable maps and to navigate for long periods of time, the user track and map information must be combined. In other words, map information must compensate for sensor information drift while user motion/track information must allow perceptually aliased allothetic information to be disambiguated. When both allothetic and idiothetic sources of information are available, there are many ways to integrate them in a representation useful for navigation. Classically, the corresponding representations are referred to as metric maps or topological maps (Filliat and Meyer 2003).

In metric maps, geometric properties of the environment such as the positions of objects are stored in a common reference frame. A metric map can be represented as a 2D floor plan or a 3D architectural map. The quality of the synthetic metric map is dependent on the quality of the idiothetic and allothetic sensors. For example, the scale and shape of the metric map are affected by the quality of the position estimated by idiothetic sensors. The drift of the position estimate is difficult to correct without making assumptions about particular properties of the environment, such as orthogonal hallways; or alternatively, without closing the loop, that is, revisiting a feature with previously recorded location and using that knowledge to estimate biases and correct computed position errors. Converting raw allothetic information such as range to a feature into a metric space is dependent on the properties of the sensor, such as measurement accuracy, and also on the local properties of the environment, for example, optical features are difficult to extract from blank walls or dimly lit areas.

In topological maps, it is the allothetic characterizations of places (features/landmarks) that the subject can reach that are stored, along with some information about their relative positions, for example, a list of discovered features/landmarks with connections to other features that can be directly accessed from the given feature. This type of high-level connection diagram of the environment is valuable in path planning. Additional details about the advantages and drawbacks of these representations can be found in Filliat and Meyer (2003).

9.2 Map Features

For each sensor type, extracting reference information from sensor measurements that can be used for navigation requires finding "unique" information, *a feature*, in the sensor data that is suitable for tracking. This means a feature that can be recognized by the sensor algorithms when encountered again. The sensor features (also referred to as landmarks) can be saved to form a map of the environment which is used to aid navigation. Mapped features can be used to provide navigation corrections when a feature is revisited.[1]

When we think of a map for navigation, several types of maps may come to mind, for example, GIS maps like Open Street Maps or elevation contour maps. These are maps that humans can interpret to aid in navigation. In SLAM, as the subjects traverse the world, they collect map landmarks or features to be used by navigation algorithms. The types of features collected can be quite different. In this section, we review a few types of map features that might be used by a navigation system.

9.2.1 Optical Features

The easiest setting to think about SLAM is in the context of an optical navigation system. The system "sees" a landmark and its relative location and logs it. Then, when the subject revisits the landmark, if any errors in position have accumulated, the subject's location can be updated based on the landmark's prior location estimate. The human brain is quite adept at selecting and matching landmarks in varying conditions, but this is a difficult problem for a machine vision system.

One of the classic challenges for computer vision systems is to make object identification reliable when the same object is viewed from different perspectives and distances, and in different lighting conditions. The premise of many vision algorithms is that interesting features on an object can be extracted together with their relative spatial locations to provide a feature based description of the object that is robust to changes in these parameters.

Another classic challenge for computer vision is to detect objects and structures that are partially blocked. Feature based approaches are well suited to tackle these problems because they treat an object as the sum of its parts rather than the precise match of the whole.

Optical landmarks also suffer from perceptual aliasing, for example, in an office building, many doors look the same. There has been significant research in computer vision system to address these issues and algorithms have been developed with varying degrees of robustness. Algorithms trade off computational complexity to achieve better object recognition performance.

In computer vision research, feature extraction methods have been developed in an attempt to overcome these issues. Corner based features are useful for detecting, characterizing and identifying man-made objects. A well-known algorithm is Harris corner detector (Harris and Stephens 1988). Selected features must be sufficiently distinct so there is low probability of mismatch. Identifying distinctive landmarks is not always simple. For example, viewed from varying distances the

[1] Feature tracking can also be used to directly solve for the resulting motion of a sensor if enough information is gathered to infer the relative movement of features in a metric map as a result of the subject motion, for example, stereo camera feature tracking.

objects will have different scales. Lindeberg introduced the concept of automatic scale selection. He showed that for feature detectors expressed in terms of Gaussian derivatives, when estimating image deformations, such as in image matching computations, scale levels with associated deformation estimates can be selected from the scales at which normalized measures of uncertainty assume local minima with respect to scales (Lindeberg 1998).

Another common image processing algorithm used for object identification is *Scale Invariant Feature Transform* (SIFT). (Lowe 1999) The algorithm is designed to detect and describe *local* features in images. Its basic premise is that objects can most reliably be recognized based on local features and their relative spatial locations. The SIFT algorithm identifies "key points" based on contrast gradients. The SIFT key points allow one to efficiently match small portions of cluttered images under rotations, scaling, change of brightness and contrast, and other transformations (Lowe 1999, 2004).

SIFT gets mixed reviews when used for SLAM applications. One major complaint is that the algorithm is computationally intensive (Lemaire and Lacroix 2007), which hinders real-time implementation. When Lowe used SIFT as a means for conducting stereo vision SLAM, the system ran at 2 Hz on a Pentium III 700 MHz processor (Se et al. 2005), a very slow computer by today's standards. The positive aspect of SIFT is its ability to produce distinctive features from natural landmarks (Miro et al. 2005; Se et al. 2005; Sim et al. 2005; Elinas et al. 2006). The distinctiveness of SIFT allows SLAM algorithms to perform global localization more easily and allows closing-the-loop approaches to work robustly (Se et al. 2005; Elinas et al. 2006).

Speeded Up Robust Features (SURF) Bay et al. (2008) was developed to address some of the computational issues of SIFT. It is loosely based on SIFT, but it uses integral images for image convolutions which is computationally faster. An integral image is an image where the value at any point (x, y) in the image is the sum of all the pixels from the origin of the original image up to and including (x, y) (Bay et al. 2006). SURF approximated, and even outperformed, SIFT and select variants (PCA–SIFT and GLOH) with respect to repeatability, distinctiveness, robustness; it also computed and compared features much faster (Bay et al. 2006).

In work for TRX, Karvounis implemented a SURF demonstration running at 30 Hz on a desktop computer—I7 Quad-Core 2.4 GHz processor (Karvounis 2011a). For these tests, a Logitech 9000 webcam at 320 × 240 resolutions was used to capture images. A database of known landmarks was created manually containing images of several "landmarks" in an office setting.

A SURF visualization was created that displays real-time updates of the camera image in the bottom panel (see Fig. 9.1). Then, as the camera is moved, each captured camera frame from the bottom box is compared with all the landmark images stored in the database. The top image displays a black box until the bottom frame matches one of the images stored in the database. Once a match is found, see Fig. 9.1, the top image shows the matched landmark from the database. The green lines indicate the feature matches and the cyan frame indicates the relative position of the captured image with respect to the database landmark image.

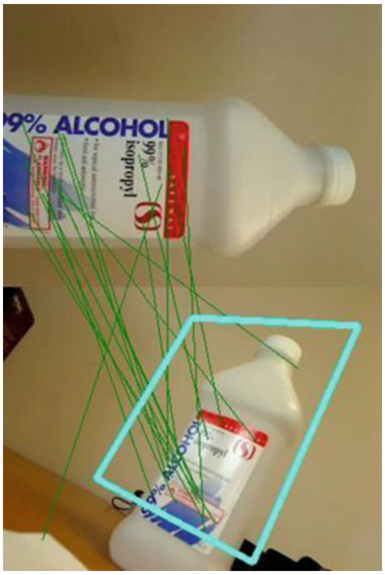


Fig. 9.1 Surf feature match

Color is not used in these algorithms. Color is an important property used by humans for object recognition; however, color perception in machine vision is very complex. A person is able to perceive color as relatively constant in differing lighting conditions. On the other hand, machine vision systems are generally not so sophisticated. For example, the color histogram derived from a digital image may vary markedly for the same object under differing lighting conditions. But as long as the illumination is held fairly constant, color histograms can be a very effective feature for object identification (Abdel-Hakim and Farag 2006; Sande et al. 2010). Frame-to-frame lighting is more likely to be nearly constant but over longer periods lighting is likely to change.

The discussion of the algorithms and software in this section is centered on image recognition. An in-depth discussion of hardware is beyond the scope of this chapter; however, since a camera may serve as the "eye" of the navigation system, its characteristics can greatly affect the functionality of the system. The quality of the images produced by the camera directly affects the processing speed as well as the ability to identify objects. The properties of the lens directly affect the field of view and the ability to carry out optical ranging. At greater distances, the resolution of the camera can be the limiting factor for feature recognition and ranging.

9.2.2 Inference-Based Features

The desire for improved localization using only the sensors available on a cell phone is driving researchers to focus on developing methods that leverage only the cell phone's embedded sensor information to its maximum benefit for pedestrian navigation. A useful source of environmental information can be derived from a tracked subject's motion. A standard approach to tracking is to use an inertial navigation unit (INU) in a dead reckoning mode making use of only the idiothetic dead reckoning information provided by the INU sensors. Inertial sensors can provide allothetic map reference data by inferring the location of terrain features based on how the subject moves through the environment. In making use of this additional information, the capability of the INU is improved to function as a smart, standalone positioning device providing a rich set of inputs for SLAM algorithms. For example, just as a stereo-optical sensor might provide SURF features and descriptors with range information for each selected feature detected in an optical frame, a "smart" navigation unit can provide inertial building and shape features and signal-based features (e.g., magnetic or signal strength when these sensors are available in the navigation unit) for input to SLAM algorithms.

For example, in buildings, floor plans represent a specific partition of a 2D space into spatially characteristic areas such as hallways, rooms, points of entry or exit including stairwells, elevators, and escalators. The existence of a hallway might be inferred if a subject moves for a long period in a confined straight area. Climbing stairs indicates the presence of a stairwell and an elevation change without climbing stairs might imply an elevator. The location and orientation of each inferred feature are known based on the idiothetic information. Several researchers have now shown that these inferred features can be used to mitigate the accumulation of inertial dead reckoning errors (Funk et al. 2007; Bandyopadhyay et al. 2008; Robertson et al. 2009a, b, 2010; Borenstein 2010; Wang et al. 2012). For example, inferred knowledge of hallways and other building grid constraints may be enforced on the navigation solution to yield an effective angular drift correction.

TRX has developed algorithms that detect such building features from track histories (Funk et al. 2007; Bandyopadhyay et al. 2008). These types of algorithms have been tested and evaluated in realistic scenarios with inertial sensors alone and found to markedly improve position accuracy. For example, in one 25 min long test, the error from pure inertial-based location estimate was reduced from 48 m to less than 3 m using the mapped-based constraint algorithms; see Fig. 9.2. Adding other sensor signature data can be used to improve uniqueness of inferred features. Investigators from Duke University and EJUST have begun to pick up on these ideas for recognizing and associating inertial signal features with fixed building features (Wang et al. 2012).

Investigators at the German Aerospace Center have developed a similar pedestrian 2D map inference system called FootSLAM (Robertson et al. 2009a, b, 2010). The algorithms builds on occupancy grid methods developed for robotic SLAM that use odometry based path data to develop a 2D map of open areas based on where the robot travelled. Instead of odometry, FootSLAM uses inertial-based dead reckoning as the input to FAST-SLAM algorithms (see section on Particle Filter based SLAM). Similar to the work at TRX, no visual or ranging sensors are used; instead the 2D is inferred based on the path data.

GPS and INUs are baseline metric sensors but they can provide inferred allothetic information. They should be distinguished from cameras, thermal imagers,

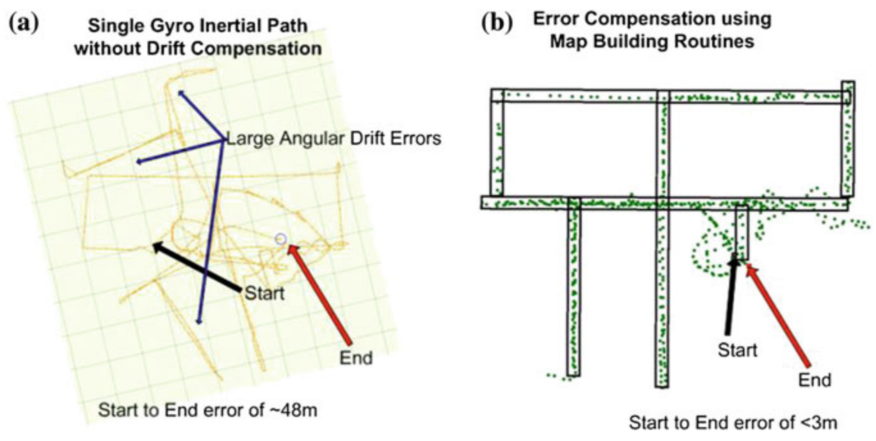
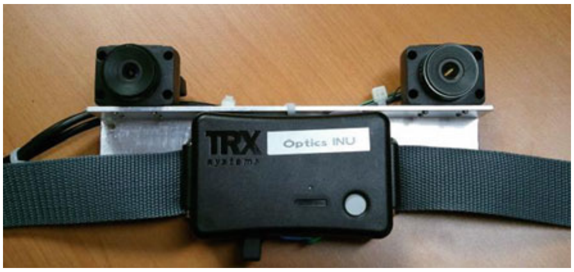


Fig. 9.2 **a** Uncompensated inertial path. **b** Generated map and compensated inertial path

Fig. 9.3 A composite sensor—an optical INU integrating stereo vision for feature extraction with an INU



etc. that can produce "pure" topological measurements of relative range, range rate or bearing to a landmark. *Composite sensors*, for example combining vision and inertial measurements (Fig. 9.3) can combine metric and topological data to build hybrid maps that enable long term navigation. Together the combined sensors produce a "composite data array" consisting of a vector-valued path of INU position, velocity, heading, etc., together with the time-space paths of environmental features extracted from the cameras and inferred from subject motion.

For example, one might infer a hallway in a building by walking down it with only an inertial sensor; however, combining the inertial data with optical (or other) information, one may be able to estimate the length and width of the hallway as well. Figure 9.4 shows the stereo left and right camera images from the optical INU. The blue lines indicate algorithm detected hallway features. The red blocks show features that had a stereo match and the yellow lines link to matched features in the left and right image. The hallway width estimation results based on stereo line detection and matching is 1.61 m. The actual width of the hallway is about 1.52 m so the estimate is off by 0.09 m (3.5 inches).

In buildings, rigid assumptions can be made on the architecture of buildings to aid in identifying building features and the underlying map. These same assumptions do not necessarily hold in natural structures, such as caves.

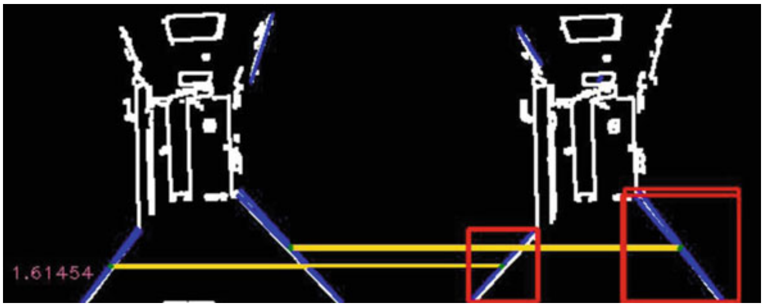


Fig. 9.4 Stereo images displaying feature matched hallway width computation

Nevertheless, it is reasonable to assume that natural trackable features will exist. Assumptions on the types of features can be adapted to allow the identification of key natural features in the environment. For example, different regions can have unique magnetic signatures, which can be measured by three-axis magnetic field sensors found in most smartphones, or received signal strength signatures (RSS), which can be accessed from most radios including, for example, Wi-Fi and Bluetooth. *Fingerprinting* methods for radio and other signals was discussed in detail in Chap. 4. In these techniques, the facility signatures are mapped a priori and then the signature map is used for localization.

9.2.3 Magnetic Features

Figure 9.5 shows an example of results from one of a sequence of magnetic signature experiments collected using a YAS529—MS-3C 3[2] axis magnetic field sensor while the tracked subject traversed the hallways of the AV Williams Building at the University of Maryland. Each corridor was found to display a consistent magnetic signature when the corridor was traversed multiple times. These signatures were recorded for three corridors as shown in Fig. 9.5. In each of the plots, the total magnetic field magnitude is plotted (y-axis) versus the sample number (x-axis) for two different traversals of each hallway. Note, there is some small variation between the two traversals for each hallway, but the hallways are clearly distinguishable.

To further test the uniqueness of the signatures, once the magnetic signatures for each hallway were recorded, tests were conducted where a small segment of one of the hallways was traversed resulting in a magnetic path signature. These magnetic signatures were tested against the three corridor database and the segments could be correctly identified in the part of the corridor where they were

[2] http://pdf1.alldatasheet.com/datasheet-pdf/view/205144/YAMAHA/YAS529.html.

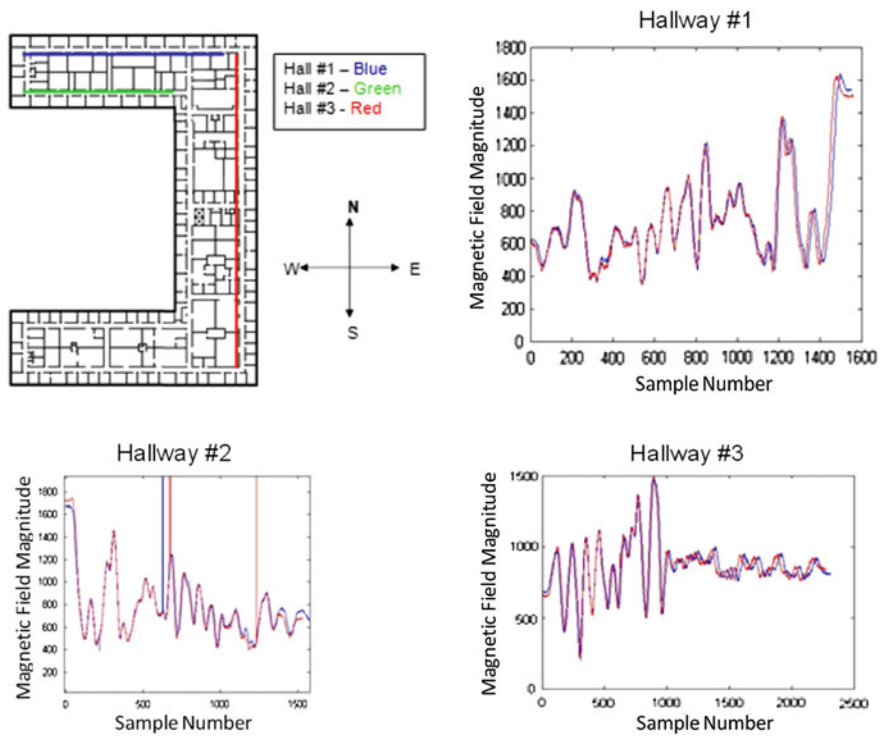


Fig. 9.5 Unique magnetic signatures of hallways

recorded.[3] These results clearly demonstrate the promise of magnetic signature fingerprints in aiding indoor localization when the signatures are available a priori.

A limitation of recording sampled signature data is that the data is speed and direction dependent. A computational method such as dynamic time warping is needed to account for variations in walking speed during the data collections. Dynamic time warping is a well-known technique for finding optimal alignment between two time dependent sequences and it is often is used in video and audio processing (Sakoe and Chiba 1978; Muller 2007).

Continuously matching path segments (in a large dataset) is computationally costly. Additionally, one may not have an a priori map as we did in the above experiment. Building a map of magnetic or signal features as the subject traverses an area, and using them for corrections in a SLAM implementation is an alternative to the fingerprinting techniques from Chap. 4.

Selecting only *interesting* features will minimize computation. Careful consideration of feature selection is critical for robustness. For example, an

[3] The subjects walked close to the center of the hallways during these tests at constant speed.

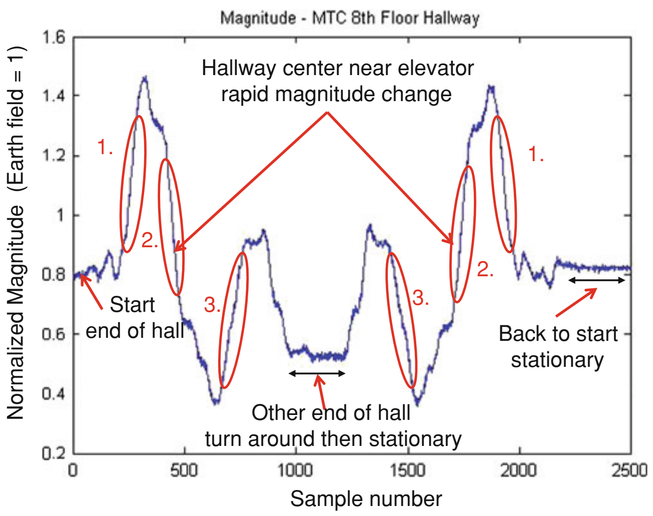


Fig. 9.6 Magnetic field Magnitude variation over time

approximately constant field may be fairly easy to match. Indoor environments typically provide a rich set of features for magnetic signatures. In outdoor environments, magnetic features may be sparse or indistinguishable. Once a feature is confirmed it can be deemed a landmark with an associated position. Recognized revisits to the landmark would subsequently provide a mechanism for mitigating accumulated dead reckoning errors.

To simplify computation, consider a well-localized magnetic feature, for example, an extreme or a *sharp transition in magnetic magnitude*. Sharp transitions are common in manmade structures with power systems and other metal causing magnetic disturbances. Figure 9.6 shows the magnitude of the magnetic field vector as a subject traverses back and forth in the hallway in an office building demonstrating the consistency of the signature. From Fig. 9.6, three sharp transition features are selected from the hallway traversal. These same three transition features are easily seen in each traversal.

The magnetic features are shown superimposed on a plot of the inertial path data of a user traversing back and forth in this hallway 10 times (Fig. 9.7). The inertial path shows clear scaling and drift errors. The path color represents the magnetic field magnitude. For each of the three magnetic features a minimum (triangle) and maximum (square) value are marked on the path.

Figure 9.8 shows a zoomed view of going back and forth in the hallway once. It is clear from these figures that the features may offer some scaling and drift correction if the features can be recognized and matched.

For signal based features such as magnetic fields, high sample rate data or other derived parameters can be saved as a feature descriptor detailing the unique aspects of the feature which can be used for matching if the features are observed at a later time. While the example above focuses on magnetic data, signature

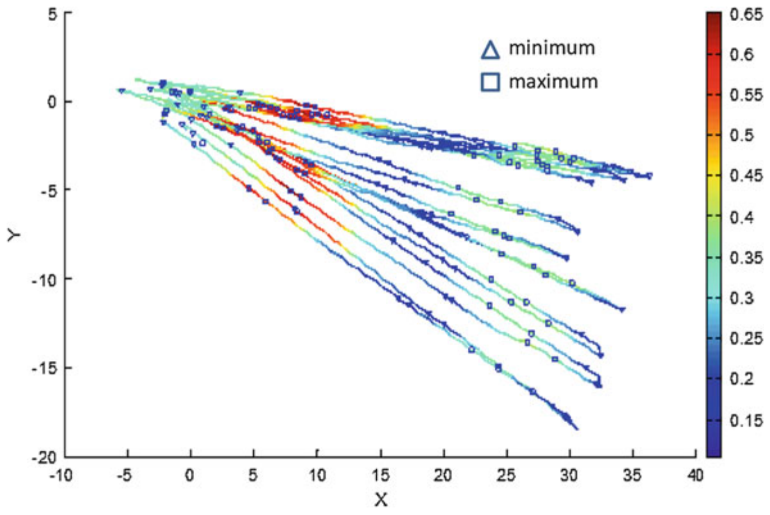


Fig. 9.7 Magnetic features superimposed on inertial track (X, Y position in meters from start point)

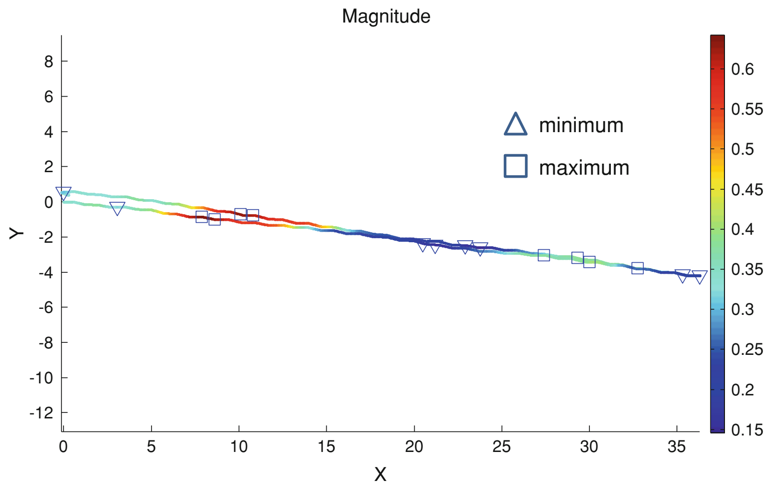


Fig. 9.8 Magnetic features superimposed on inertial track—zoom in view (X, Y position in meters from start point)

features are also valuable from other types of sensor data, such as radiation measurements or received signal strength.

We have highlighted a few allothetic sensors that provide map features that can be used for SLAM formulations. There are others we have not touched on such as LIDAR and SONAR. In Thrun et al. (2006) provide models for these and other sensors in the context of SLAM application. In the next sections, the formulation and solution of the SLAM problem, as well as issues with real-time implementation are discussed.

9.3 Simultaneous Localization and Mapping Formulation

In robotics, the SLAM problem is considered "solved." This theoretical solution has been one of the notable successes of the robotics community (Smith and Cheeseman 1986; Durrant-Whyte 1988; Smith et al. 1990; Durrant-Whyte and Bailey 2006a, b). Because position estimates and measurements are imperfect, the solution to the SLAM problem required the development of a way to update an uncertain geometric or topological environment model based on new observations that maintained consistent interpretation of relations between all of the uncertain features (Smith and Cheeseman 1986; Durrant-Whyte 1988). Work by Smith and Cheesman and Durrant-Whyte (Smith and Cheeseman 1986; Durrant-Whyte 1988) established a statistical basis for describing relationships between fixed landmarks with geometric uncertainty. A key contribution of this work was to show that, due to the common error in estimated observer location between landmarks, there must be a high degree of correlation between estimates of the location of different landmarks in a map. In fact, these correlations grow with successive observations of the landmarks. *Practically, this means that the relative location between any two landmarks may be known with high accuracy, even when the absolute location of a specific landmark is quite uncertain*. The combined mapping and localization problem, once formulated as a single estimation problem, is convergent—that is, the estimated map converges monotonically to a relative map with zero uncertainty. Additionally, the absolute accuracy of the map and subject location reaches a lower bound defined only by the uncertainty in the initialization (Smith and Cheeseman 1986; Durrant-Whyte 1988). The correlations between landmarks are the critical part of the problem and the stronger the correlations grow, the better the solution (Smith and Cheeseman 1986; Durrant-Whyte 1988; Smith et al. 1990; Durrant-Whyte and Bailey 2006a, b).

The SLAM problem can be broken into two pieces. The *observation model* (*or sensor model*) $p(z_t|x_t)$ describes the probability of making an observation z_t of selected landmarks when the observer location and landmark locations are known. In SLAM, the system state x_t includes the observer pose as well as the map. It is reasonable to assume that once the observer location and map are defined, observations are conditionally independent given the map and the current observer state. The *motion model* $p(x_t|u_t, x_{t-1})$ for the observer is assumed to be a Markov process in which the next state depends only on the immediately preceding state x_{t-1} and the applied control u_t (which may be unknown as is the case in personnel tracking) and is independent of both the observations and the map. The SLAM algorithm is then solved by a Bayes filter in a standard two-step time update, measurement update form.

1. **Time Update**: prediction of the state given the previous state and the control input

$$p(x_t|z_{1:t-1}, u_{1:t}) = \int p(x_t|u_t, x_{t-1})p(x_{t-1}|z_{1:t-1}, u_{1:t-1})dx_{t-1},$$

and

2. **Measurement Update**: update of the predicted value given the most recent sensor data

$$p(x_t|z_{1:t}, u_{1:t}) = \eta p(z_t|x_t)p(x_t|z_{1:t-1}, u_{1:t})$$

where η is a normalization constant (Thrun et al. 2006).

The derivation of this and similarly all the popular recursive state estimation filters rely on the Markov assumption, which postulates that past and future data are independent given the current state. The Bayes filter is not practically implementable at this level of abstraction. Approximations are often made to control computational complexity, e.g., linearity of the state dynamics, Gaussian noise, etc. The resulting unmodeled dynamics or other model inaccuracies can cause violations of this assumption. In practice, the filters are surprisingly robust to such violations (Thrun et al. 2006).

In probabilistic form, the SLAM problem requires that the *joint* posterior probability density of the landmark locations and tracked subject's state (at time *t*), given the recorded observations and control inputs up to and including time *t* together with the initial state of the tracked subject, be computed for all times *t*. Solutions to the probabilistic SLAM problem involve finding an appropriate representation for both the observation model and the motion model, preferably recursive, which allows efficient and consistent computation of the prior and posterior distributions.

The SLAM problem has been formulated and solved as a theoretical problem in a number of different forms. However, issues remain in realizing general SLAM solutions in practice and notably in building and using perceptually rich maps as part of a SLAM algorithm. By far, the most common representation is in the form of a state-space model with additive Gaussian noise, leading to the use of the extended Kalman filter (EKF) to solve the SLAM problem.

The popularity stems from the fact that the EKF provides a recursive solution to the navigation problem and a means of computing consistent estimates for the uncertainty in subject and map landmark locations. This is despite the fact that many sensor noise models are not well represented by additive Gaussian noise.

9.3.1 Kalman Filter

Here, we take a short diversion to briefly discuss one of the most popular Bayesian filters, the Kalman Filter and a couple of its extensions, and to highlight some of the properties of the Kalman filter that drive its popularity. More detailed discussions and complete derivations can be found in Kailath (1980), Thrun et al. (2006).

A Kalman filter is a computationally tractable mechanism to incorporate

1. imprecise knowledge about a system—system dynamic models, noise models
2. system observations—measurements, sensor models

to yield an estimate of the current state. Under the assumptions that the system is linear and the model and observation errors are independent Gaussian random variables, the Kalman state estimate is an ***optimal*** estimate. There are several possible definitions for optimality

$$\begin{aligned}
\text{Minimum Mean Square Error} \quad \hat{\theta} &= \underset{\theta}{\text{argmin}} \; \|y - f(\theta)\|^2 \\
\text{Maximum Likelihood} \quad \hat{\theta} &= \underset{\theta}{\text{argmax}} \; p(\theta|y) \\
\text{Minimum Variance} \quad \hat{\theta} &= \underset{\breve{\theta}}{\text{argmin}} \; E(\theta - \breve{\theta})^2 \\
\text{Maximum a Posteriori (MAP)} \quad \hat{\theta} &= \underset{\theta}{\text{argmax}} \; p(y|\theta)
\end{aligned}$$

It turns out because of the nice properties for linear systems and Gaussian noise, the Kalman state estimate satisfies all of these optimality criteria. In addition, the estimate is

- Unbiased[4]: the expected value of the estimate is the same as the parameter, and
- Consistent: the variance decreases to 0 with further observations

Because computer realizations of the algorithm are necessarily implemented in discrete time, here we summarize the Kalman filter for a discrete linear system. The linear system state is $x_k \in \mathbb{R}^n$, the control $u_k \in \mathbb{R}^p$, the measurements $y_k \in \mathbb{R}^m$, and additive, independent, zero mean, state noise $w_k \in \mathbb{R}^n$ and measurement noise $v_k \in \mathbb{R}^m$:

$$\begin{aligned} x_k &= Ax_k + Bu_k + w_k \\ y_k &= Cx_k + v_k \end{aligned}, \qquad \begin{pmatrix} w_k \\ v_k \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q & 0 \\ 0 & R \end{bmatrix} \right)$$

Figure 9.9 shows the system model and Fig. 9.10 shows the standard form of the recursive estimator. Starting with an estimate of the initial state, $\hat{x}_0 \in \mathbb{R}^n$ and given a control input $u_1 \in \mathbb{R}^p$ the next state, $\hat{x}_{2|1}$, is predicted. The observations at time 2 are then used to update the state $\hat{x}_{2|2}$ and so on. It would be a good guess that the best prediction of the state given the control inputs can be obtained by

[4] The Cramer Rao Lower Bound (CRLB) gives smallest variance achievable by an unbiased estimate.

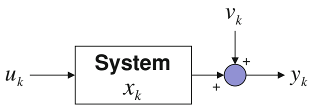


Fig. 9.9 System model

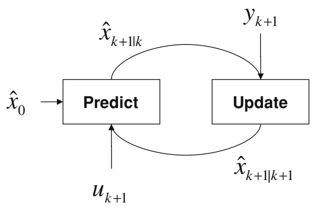


Fig. 9.10 Recursive estimator

simply applying the system model, the difficult piece is to decide how to optimally update the state given the observations. The Kalman filter provides the optimal update and additionally provides an error covariance that provides information on how good the estimate is.

Figure 9.11 shows the discrete time Kalman Filter algorithm.

The Kalman filter prediction step uses the system model to update the state given the control $\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k$ and the state estimate given observations is $\hat{x}_{k+1|k+1} = \hat{x}_{k|k+1} + K\left(y_{k+1} - C\hat{x}_{k+1|k}\right)$. The Kalman gain, K, is chosen to minimize the error covariance, P.

The derivations of the update equations for the Kalman gain and error covariance are made under the assumptions that the system is linear and the model and observation errors are independent Gaussian random variables. Given a complete system model, these assumptions imply a Markov property that past and future data are independent given the current state (Kailath 1980; Thrun et al. 2006).

The mathematical model introduced above is similar to the Markov model introduced in Chap. 4 for robot localization. A key difference is that each saved map feature is added to the system state and also tracked. This can cause a large increase in computational complexity over methods that assume a known map. A method for overcoming some of the practical implementation issues associated with the added computational complexity is discussed in the section SLAM Implementation.

If the system and or measurement model is nonlinear,

$$\begin{aligned} x_k &= f(x_k, u_k, w_k) \\ y_k &= g(x_k, v_k) \end{aligned}$$

an extension of the Kalman Filter (the Extended Kalman Filter EKF) is made by substituting a linearized version of the system model, $\bar{A}_k = \frac{\partial f}{\partial x}\Big|_{(\hat{x}_{k|k}, u_k)}$, $\bar{B}_k = \frac{\partial f}{\partial u}\Big|_{(\hat{x}_{k|k}, u_k)}$ and measurement model $\bar{C}_k = \frac{\partial g}{\partial x}\Big|_{(x_{k|k})}$, into the computation of the prediction and update of error covariance, and computation of Kalman gain.

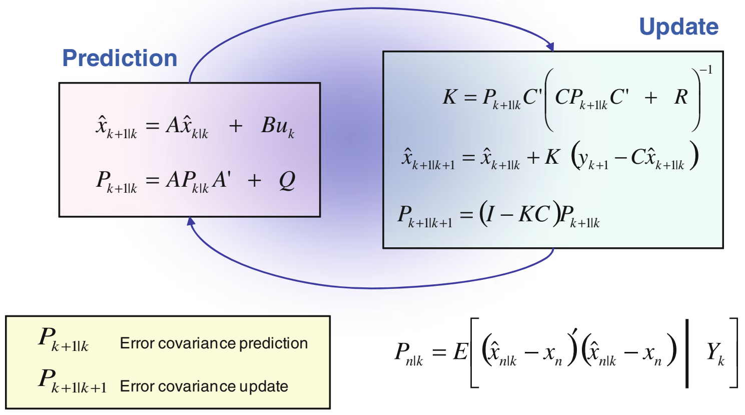

Prediction
$\hat{x}_{k+1|k} = A\hat{x}_{k|k} + Bu_k$
$P_{k+1|k} = AP_{k|k}A' + Q$
Update
$K = P_{k+1|k}C'\left(CP_{k+1|k}C' + R\right)^{-1}$
$\hat{x}_{k+1|k+1} = \hat{x}_{k+1|k} + K\left(y_{k+1} - C\hat{x}_{k+1|k}\right)$
$P_{k+1|k+1} = (I - KC)P_{k+1|k}$
$P_{k+1|k}$ Error covariance prediction
$P_{k+1|k+1}$ Error covariance update
$P_{n|k} = E\left[\left(\hat{x}_{n|k} - x_n\right)'\left(\hat{x}_{n|k} - x_n\right) \middle| Y_k\right]$

Fig. 9.11 Discrete time Kalman filter

The model prediction and update use the full nonlinear system model. The resulting unmodeled dynamics or other model inaccuracies induce violations of the original assumptions of the Kalman Filter derivation and so the EKF is no longer an optimal solution. Despite this, in practice the EKF often provides a useful solution when the linearization offers a good estimate over the current operating range of the system. Unfortunately, this is not always the case. This is because the linearization does not preserve the true mean and true covariance of the posterior distributions (Thrun et al. 2006).

A popular method that often performs better is the unscented Kalman Filter (UKF). The UKF performs a stochastic linearization through the use of a weighted statistical linear regression process, see Thrun et al. (2006) for more details. While the EKF is accurate to the first-order Taylor series expansion, the UKF is accurate to the first two terms in the expansion (Thrun et al. 2006).

The standard formulation of both the EKF-SLAM and UKF-SLAM solution is especially vulnerable to incorrect association of observations to landmarks. A single incorrect data association can induce divergence into the algorithm for map estimation, often causing catastrophic failure of the localization algorithm. (Durrant-Whyte and Bailey 2006a, b) One way to handle uncertain association of observations to landmarks is to generate a separate track estimate for each association hypothesis, creating over time an everbranching tree of tracks. This *multi-hypothesis data association* is important for robust SLAM implementation. Multihypothesis data association is especially important in loop closure, allowing a separate hypothesis for suspected loops and also a "no-loop" hypothesis for cases where the perceived environment is structurally similar.

A major hurdle in multihypothesis data association is the computational overhead of maintaining separate map estimates for each hypothesis. The number of tracks is typically limited by the available computational resources, and low-likelihood tracks must be pruned from the hypothesis tree.

9.3.2 Particle Filter

An important alternative to Kalman Filtering methods is the use of particle filters. Particle filters are a class of nonlinear filters that impose no restriction on the system model, measurement model, or nature of the noise statistics. Particle filters compute a solution based on sequential Monte Carlo simulations of particles that are selected to represent the posterior distributions. Particle filters are only optimal given infinite computational resources, but even with limited resources, they can give better solutions than the EKF in cases where the operational region is highly nonlinear. (Gustafssson et al. 2002; Ristic et al. 2004; Thrun et al. 2006).

One thing to be very cautious about is that computational complexity for nonlinear filters generally grows exponentially with the dimension of the system, whereas for the Kalman filter computational complexity grows as the cube of the dimension. While there are ways to keep the computational complexity under control, it is something that cannot be overlooked. The particle filter approach to modeling uncertainty is only possible because of the availability of fast, low-cost computers with large memories.

FAST-SLAM, with its basis in recursive Monte Carlo sampling, or particle filtering, was the first method to directly represent the nonlinear process model and nonGaussian pose distribution (Montemerlo et al. 2003a; Montemerlo and Thrun 2003b). Prior to the development of FAST-SLAM, the large state-space dimension in SLAM due to the number of map states made direct application of particle filters computationally infeasible. This issue is solved in FAST-SLAM by using a Rao-Blackwellized particle filter where the joint subject and map state is factored into a subject component, and a map component that is conditioned on the subject trajectory:

$$p(x_{0:t}, m | z_{0:t}, u_{0:t}, x_0) = p(x_{0:t} | z_{0:t}, u_{0:t}, x_0) p(m | x_{0:t}, z_{0:t}).$$

Note that, the probability distribution of the subject is on the entire trajectory rather than the single state as it is in EKF. When conditioned on the trajectory, the map landmarks become independent. This follows since given the exact pose states from which the observations are made, the observations are independent and therefore the map states are also independent.

The independence of map states is an important difference and the reason behind the speed improvements of FAST-SLAM over EKF algorithms. Because of the independence of the map states, updating the map, for a given pose trajectory particle (a single realization of the subject trajectory) is very fast. The map can be represented as a set of independent Gaussians. Each observed landmark can be processed individually as an EKF measurement update from a known pose.

FAST-SLAM linearizes the observation model, which is typically a reasonable approximation for range-bearing measurements when the subject's pose is known. Unobserved landmarks are independent and so unchanged.

Propagating the pose states is performed by particle filtering. The essential structure of FAST-SLAM, then, is a trajectory represented by weighted samples (particles) and a map is computed by EKF updates. The map accompanying each particle is composed of independent Gaussian distributions.

The FAST-SLAM algorithm is inherently a multihypothesis solution, with each particle having its own map estimate. A significant advantage of the FAST-SLAM algorithm is its ability to perform per particle data association (Montemerlo and Thrun 2003b).

Many types of recursive probabilistic state estimate algorithms have been developed to solve the SLAM problem in an approximate, computationally tractable way. While EKF-SLAM and FAST-SLAM are the two most important solution methods, newer alternatives have been proposed (Durrant-Whyte and Bailey 2006a, b; Karvounis 2011a). Information Filters and their extensions are of particular interest. Information Filters are duals of the Kalman Filter that have both computational and representation advantages when applied to location and mapping problems (Thrun et al. 2006).

9.3.3 Graph SLAM

GraphSLAM algorithms are also important SLAM implementations but the solution is typically not computed in real-time so we will not cover them here. For more information on GraphSLAM methods refer to Thurn and Montemerlo (2005); Thrun et al. (2006); Koller and Friedman (2009).

One particular GraphSLAM algorithm that supports real-time implementation is based on Factor Graphs (Loeliger 2004). Factor graphs provide a unified approach for modeling complex systems and to deriving practical message passing algorithms for the associated detection and estimation problems. Factor graphs allow most well-known signal processing techniques including Kalman and particle filtering to be used as components of such algorithms (Loeliger 2004).

Researchers at Georgia Tech and MIT have applied factor graph methods for incremental smoothing in inertial navigation systems (Indelman et al. 2012; Kaess et al. 2012). The system navigation states are nodes in the graph and each IMU measurement introduces a new factor to the graph connecting to the navigation state nodes. This factor may also be connected to other nodes used for parameterizing errors in the IMU measurements such as bias and scale factor. These nodes can be added at a lower frequency than the navigation state nodes. Other aiding sensors are simply additional sources of factors that get added to the graph asynchronously whenever their measurements are available. In this way, the factor graph formulation allows multirate, asynchronous measurements to be incorporated in a natural way (Indelman et al. 2012). The nonlinear optimization problem

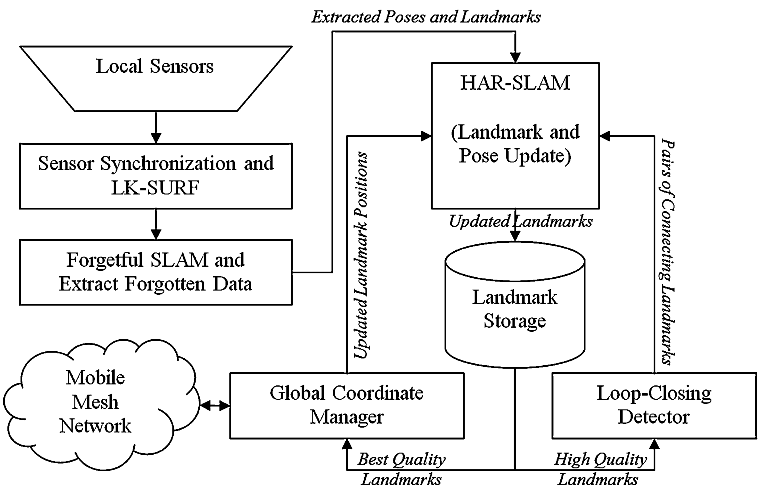

Extracted Poses and Landmarks
Local Sensors
Sensor Synchronization and LK-SURF
Forgetful SLAM and Extract Forgotten Data
HAR-SLAM (Landmark and Pose Update)
Updated Landmark Positions
Pairs of Connecting Landmarks
Updated Landmarks
Landmark Storage
Mobile Mesh Network
Global Coordinate Manager
Loop-Closing Detector
Best Quality Landmarks
High Quality Landmarks

Fig. 9.12 HAR-SLAM Algorithm flow chart

encoded by the factor graph is solved by repeated linearization within a standard Gauss–Newton style nonlinear optimizer. The optimization can proceed incrementally because most of the calculations are the same as in the previous step and can be reused. As long as only sequential IMU measurements are processed, the resulting graph will have a chain-like structure. By maintaining all information within a single graph, the filter and smoother can operate asynchronously. This allows the problem to be split into a high speed navigation component and a higher latency loop closure component (Kaess et al. 2012).

9.4 SLAM Implementation

While theoretically the SLAM problem has been solved, a major issue that is faced in developing real-time implementations of SLAM is that as the number of tracked features/landmarks increases, the computation required at each step increases as a square of the number of landmarks. Required map storage also increases as a square of the number of landmarks (Dissanayake et al. 2001). Many people have developed SLAM implementations to address this issue (Montemerlo et al. 2003a; Montemerlo and Thrun 2003b; Kim and Sukkarieh 2004; Veth 2011; Karvounis 2011a). For example, computational complexity can be reduced by subdividing the map and by making the covariance matrix more sparse.

Here, we consider a particular implementation done at TRX Systems that attempts to address the computational issues in tracking an increasing number of features. Karvounis implemented an extreme version of this approach called Hierarchical SLAM or HAR-SLAM. Figure 9.12 gives a flow chart overview of how the system works. Full details of the algorithms are described in Karvounis

(2011a, c). The HAR-SLAM algorithm has similarities to the factor graph approach (Loeliger 2004; Kaess et al. 2012) in that both result in a chain-like structure of system states.

In this approach, the lowest level SLAM algorithm (Forgetful-SLAM: Fig. 9.12 left hand side) maintains active tracking of only the landmarks that can currently be seen or have been seen in the last N minutes (up to some max number of tracked landmarks). By limiting the set of landmarks tracked, the computational complexity remains bounded. Note that in Forgetful-SLAM landmarks are only matched to the landmarks currently seen by the camera; they are not matched to landmarks from the global map. This is a purely local SLAM layer and it will not offer the capability of correcting based on a previously known landmark (often referred to as "closing the loop"). That type of correction is handled by the higher level algorithm, HAR-SLAM.

The landmarks/features that are dropped from the Forgetful-SLAM algorithm are not actually forgotten; instead they are promoted and tracked within the global map by the HAR-SLAM algorithm, if they are determined to be "good", meaning that their covariance matrix $\mathbf{P}_{\text{landmark}}$ is small and "relevant", meaning that changes in the landmark location will affect the pose. To determine how much a good landmark can affect a pose, a metric combining the cross-covariance matrix between the landmark and pose, $\mathbf{P}_{\text{cross}}$, with the inverse of the landmark covariance matrix, $\mathbf{P}_{\text{landmark}}$, is used:

$$\text{max Eigen value}\left(\mathbf{P}_{\text{cross}}^{T}(\mathbf{P}_{\text{landmark}})^{-1}\mathbf{P}_{\text{cross}}\right)$$

Landmarks are promoted when the max Eigenvalue is greater than a threshold. As landmarks are removed from Forgetful-SLAM and promoted, their correlations are tied to the last pose (historical position and orientation). There is a state vector and covariance matrix per pose, a state vector and covariance matrix per landmark, and a cross-covariance matrix per link.

As new poses are promoted from Forgetful-SLAM, any updates ripple back through the chain of historical pose estimates. Each pose is updated through a correlating Kalman Filter, and each landmark is updated through its own Kalman Filter. This directional update procedure defines the global level update (HAR-SLAM). A key advantage of this method is that both storage and computations grow only linearly with the number of landmarks and poses. (Karvounis 2011a, c).

The global coordinate manager is secondary loop that is run to manage the coordinate transforms for merging map data from other tracked subjects when matching features are detected in their respective maps. This is discussed in more detail later in this chapter. This property of remembering all poses and linking landmarks only to a single pose allows multiple tracked subjects to link maps together and asynchronously update portions of the map.

Figure 9.13 shows a high level diagram of HAR-SLAM. Each landmark and pose has a state vector x_i and associated covariance matrix P_i. In the Forgetful-SLAM section, features are fully linked to each other and the tracked subject's pose by cross-covariances $P_{i,j}$. Features no longer in view may be selected for

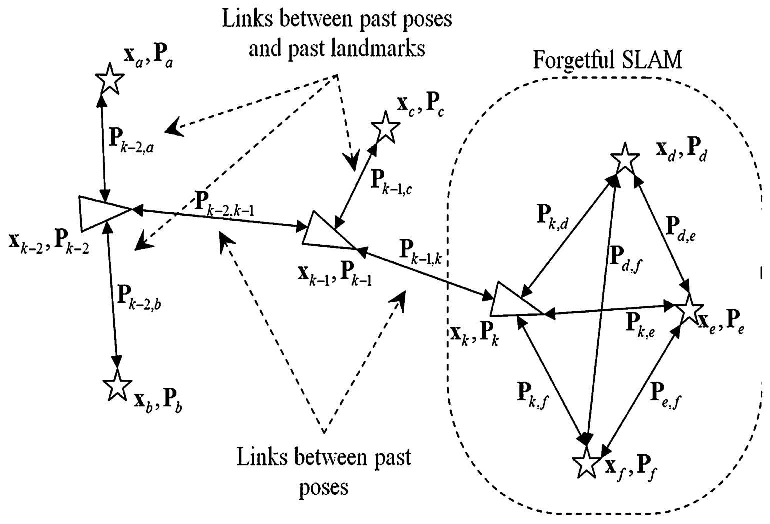

Links between past poses and past landmarks
Forgetful SLAM
$\mathbf{x}_a, \mathbf{P}_a$
$\mathbf{P}_{k-2,a}$
$\mathbf{x}_c, \mathbf{P}_c$
$\mathbf{P}_{k-1,c}$
$\mathbf{x}_{k-2}, \mathbf{P}_{k-2}$
$\mathbf{P}_{k-2,k-1}$
$\mathbf{x}_{k-1}, \mathbf{P}_{k-1}$
$\mathbf{P}_{k-1,k}$
$\mathbf{P}_{k-2,b}$
$\mathbf{x}_b, \mathbf{P}_b$
$\mathbf{x}_d, \mathbf{P}_d$
$\mathbf{P}_{k,d}$
$\mathbf{P}_{d,e}$
$\mathbf{P}_{d,f}$
$\mathbf{x}_k, \mathbf{P}_k$
$\mathbf{P}_{k,e}$
$\mathbf{x}_e, \mathbf{P}_e$
$\mathbf{P}_{k,f}$
$\mathbf{P}_{e,f}$
$\mathbf{x}_f, \mathbf{P}_f$
Links between past poses

Fig. 9.13 Overview diagram of HAR-SLAM

promotion. Once features are promoted cross-correlations tie them to only the last pose from which the landmark was seen. All other links are broken. In the HAR-SLAM update, each pose and any associated landmark(s) are updated depending only on their direct links using Kalman gains.

Remembering the entire path instead of only the best current estimate allows HAR-SLAM to more quickly recover from errors by adjusting the entire historical path and all associated landmarks. This property of remembering historical poses and linking landmarks only to a single pose allows multiple tracked subjects to link maps together and asynchronously update portions of the map.

Feature management and promotion are the key differences between Forgetful-SLAM and the standard EKF-SLAM. In Forgetful-SLAM, features that are no longer seen are removed and considered for promotion to the global map. In order for a high level SLAM algorithm to assemble and track the "forgotten" features, the features need to be recoverable. A cross-correlation matrix is generated per lost feature that relates the feature to the previous pose (the last pose from which the feature was observed). Only features/landmarks where the max eigenvalue metric is greater than a threshold are promoted.

Karvounis developed HAR-SLAM primarily with the goal of limiting computation. As such, a main advantage of HAR-SLAM is its low computational cost. The cost grows linearly with the number of states and landmarks, while typical Kalman based SLAM algorithms are quadratic in cost (Fig. 9.14). FAST-SLAM is the closest to HAR-SLAM in computational cost, with the same linear growth with poses and number of landmarks; however, FAST-SLAM is based on particle filtering and so it maintains several particles, each with its own map, whereas a single map is maintained in HAR-SLAM.

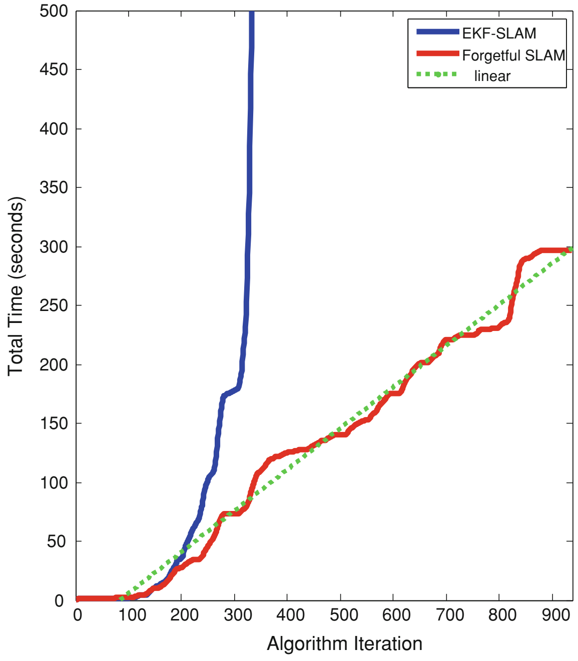


Fig. 9.14 EKF vs forgetful-SLAM

The Markov assumptions or complete state assumption that underlies all Bayesian Filters would imply that knowledge of prior states is not needed. However, unmodeled dynamics, other model inaccuracies or approximations, and correlations in inputs to the filters are all common in implementations and cause violations of the assumption. As it turns out, there are other benefits to maintaining prior pose history.

An added benefit that comes from linking landmarks through poses is that the pose history provides a directed approach for actively correcting the map and the entire path history. Because the historical corrections do not affect the current pose estimate, the rippling changes can have some delay, if necessary, to manage computational resources. Another advantage of keeping the entire pose history is that it facilitates closing the loop when matching features. A simple shortest path algorithm can find the chain of connecting poses between two landmarks, and this provides a directed path for updating the entire system (and computing needed cross correlations). Breaking the update into a chain reduces computation complexity to a point where the lower level SLAM and feature extraction algorithms are where the majority of computational resources are spent.

A key contribution in the development of the first SLAM algorithms was to show that, due to the common error in estimated observer location, there must be a high degree of correlation between estimates of the location of different landmarks in a map (Durrant-Whyte 1988; Smith et al. 1990; Durrant-Whyte and Bailey 2006a, b). The correlations between landmarks are a critical part of the problem and the more the correlations grow, the better the solution (Durrant-Whyte 1988;

Smith et al. 1990; Durrant-Whyte and Bailey 2006a, b). In Forgetful-SLAM, all feature to feature and feature to pose correlations are tracked and only the best features are promoted. Once promoted, in HAR-SLAM the features are extracted into a chain and tied to only to the last pose from which the feature was seen (as shown in Fig. 9.13). This eliminates cross-correlation links between features and between all but one pose. This change from the theoretical fully connected solution was made to improve computational speed and it is been demonstrated to be an effective approach in practice.

9.4.1 Outlier Removal

Kalman Filters are the method of choice for many navigation problems because the Kalman filter offers a computationally efficient optimal solution in the case that the underlying system has linear dynamics and the noise is Gaussian additive. Unfortunately, these assumptions do not hold for many navigation systems.

The standard Kalman filter algorithm is unable to handle the nonGaussian errors frequently encountered in various types of ranging systems, for example:

- incorrectly matching stereo image features,
- missed or incorrect detections caused by poor lighting
- ranging to unexpected people/objects moving in the field of the sensor.

Failure to recognize and reject these disturbances can cause non recoverable navigation errors in Kalman filter based navigation systems.

One option is to estimate the nonGaussian error probability and then apply a particle filter which can handle nonGaussian disturbances (Ristic et al. 2004). Particle filters have been used successfully in this way but at some computational cost. Another option is to develop a robust method for recognizing and rejecting outliers before allowing them to enter the Kalman filter. To minimize computational burden, Karvounis developed a Robust Kalman Filter that is able to recognize and reject the disturbances based on expected motion (Karvounis 2011a, b, c).

Typically, robust filters remove outliers before entering the Kalman Filter stage. What is novel and interesting about Karvounis' approach is that the median filter is inserted between the prediction and measurement update step of the Kalman Filter (Karvounis 2011b). Principle Component Analysis (PCA) is used to map the multidimensional observed features into a 1D space. Error vectors are computed by multiplying the error between the measured and the predicted observation values by the Kalman gain to find the effect of individual observation errors on the state. PCA is used to compute the principal vector in the state error space that causes the projected errors to be maximally distributed, making it sensitive to outliers. This technique is agnostic to the number of dimensions and the number of measurements. Including the Kalman gain scaling is important because it provides a weighting of the observation errors based on the how much the state is affected by the error, not just on the quality of the measurement. Combining PCA with a median filter provides a robust way to remove outliers.

Consider a nonlinear system model

$$\begin{aligned} x_k &= f(x_k, u_k, w_k) \\ y_k &= g(x_k, v_k) \end{aligned}$$

where the system state is $x_k \in \mathbb{R}^n$, the control $u_k \in \mathbb{R}^p$, the measurements $y_k \in \mathbb{R}^m$, with the assumption of Gaussian zero mean state noise $w_k \in \mathbb{R}^n$ with variance Q_k and Gaussian zero mean measurement noise $v_k \in \mathbb{R}^m$ variance R_k. The linearized system is given by

$$\begin{aligned} x_k &= A_k x_k + B_k u_k + w_k \\ y_k &= C_k x_k + v_k \end{aligned}, \begin{pmatrix} w_k \\ v_k \end{pmatrix} \sim N\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} Q_k & 0 \\ 0 & R_k \end{bmatrix}\right)$$

where, $A_k = \frac{\partial f}{\partial x}\Big|_{(\hat{x}_{k|k}, u_k)}$, $B_k = \frac{\partial f}{\partial u}\Big|_{(\hat{x}_{k|k}, u_k)}$ and $C_k = \frac{\partial g}{\partial x}\Big|_{(x_{k|k})}$.

The Kalman filter prediction step is the same as in a standard EKF. It uses the system model to update the state given the control $\hat{x}_{k+1|k} = A_{k+1}\hat{x}_{k|k} + B_k u_k$ and compute a predicted estimate covariance $P_{k+1|k} = A_{k+1}P_{k|k}A_{k+1}^{'} + Q_k$.

In order to accommodate the fact that the measurement noise is not actually Gaussian in practice, the outlier removal function is inserted at this stage into the Kalman Filter. Assuming that each of the N observed features is independent from other features, each feature's covariance can be extracted from the block diagonal measurement covariance matrix R_k. The weights are determined by considering the effect each feature has on the state if the Kalman gain is applied. So for each observed feature,

$$\begin{aligned} K &= P_{k+1|k} C_{k+1}^{(i)'} \left(C_{k+1}^{(i)} P_{k+1|k} C_{k+1}^{(i)'} + R_{k+1}^{(i)} \right)^{-1} \\ \tilde{x}_{k+1}^{(i)} &= K\left(y_{k+1}^{(i)} - g^{(i)}(\hat{x}_{k+1|k}) \right) \end{aligned}$$

For measurement related functions, $C_{k+1}^{(i)}, R_{k+1}^{(i)}, g^{(i)}, y_{k+1}^{(i)}$, the superscript (i) indicates the portion related to the selected feature. Note that each of $K, P_{k+1|k}, \hat{x}_{k+1|k}, \tilde{x}_{k+1}^{(i)}$, are full size. For $\tilde{x}_{k+1}^{(i)}$, the superscript (i) indicates that this is the state correction that is indicated due to the variation of that observed feature from what was predicted.

The mean and variance of the state corrections is then computed over the set of all features.

$$\bar{x}_{k+1} = \frac{1}{n}\sum_{i=1}^{N} \tilde{x}_{k+1}^{(i)} \tilde{\mathrm{X}}_{k+1} = \sum_{i=1}^{N} (\tilde{x}_{k+1}^{(i)} - \bar{x}_{k+1})(\tilde{x}_{k+1}^{(i)} - \bar{x}_{k+1})^{'}$$

The largest eigenvector v of $\tilde{\mathrm{X}}_{k+1}$ is the principal vector in the state space that causes the projected error corrections to be maximally distributed. Each feature's weight is then determined by projecting the state correction for that feature onto that principle vector: $w_i = v' x_{k+1}^{(i)}$. Outliers in this space are then eliminated using a median filter (see Chap. 8). Note that by using the state correction as a common metric for selecting outliers, measurements of different dimensions can be compared.

Next, for each of measurement related functions, $C_{k+1}^{(i)}, R_{k+1}^{(i)}, g^{(i)}, y_{k+1}^{(i)}$, for all i in the set of features that were not eliminated by the median filter, the matrices and functions must be reformed (now having reduced observation dimension). To make clear the reduction in dimension, we indicate them by $\tilde{C}_{k+1}, \tilde{R}_{k+1}, \tilde{g}, \tilde{y}_{k+1}$ in the update equations for the EKF.

$$
\begin{aligned}
K &= P_{k+1|k}\tilde{C}'_{k+1}\left(\tilde{C}_{k+1}P_{k+1|k}\tilde{C}'_{k+1} + \tilde{R}_{k+1}\right)^{-1} \\
\hat{x}_{k+1|k+1}^{(i)} &= \hat{x}_{k+1|k}^{(i)} + K\left(\tilde{y}_{k+1}^{(i)} - \tilde{g}(\hat{x}_{k+1|k}^{(i)})\right) \\
P_{k+1|k+1} &= (I - K\tilde{C}'_{k+1})P_{k+1|k}
\end{aligned}
$$

In the next section, we review some experiments that show the performance benefit of the Robust Kalman Filter when using stereo-optical measurements as part of an optical SLAM algorithm.

9.4.2 Experimental Results

Experiments to compare the performance of selected SLAM algorithms were conducted at TRX. The test was conducted using a robot from the University of Maryland's Autonomous Systems Lab (ASL). The ASL robot has the capability to report location via encoders, which provides a position estimate with roughly a 0.1 % error over distance travelled,[5] however, the robot uses gyros for heading, and these can have a drift in the heading estimate. Figure 9.15, the right image, shows in the center a PC that controls the robot and processes all data. The rear of the robot has a router that is setup to network with other robots but this capability was not used in the experiments reported here. Sonar sensors surrounding the robot were also available but not used in this experiment. The PC is powered by lithium ion batteries and the robot is powered by nickel metal hydride batteries.

The ASL robot was equipped with a TRX INU containing six-axis inertial, three-axis magnetic and barometric pressure sensors and enhanced with stereo Firefly cameras from Point Gray, as circled in the left image of Fig. 9.15. The Firefly cameras have a global shutter, which minimizes image blur, and a trigger that allow us to sync the images with the inertial measurements from the TRX INU. To selected and track stereo-optical features, LK-SURF, a hybrid feature

[5] This assumes traveling in a straight line track without wheel slip.

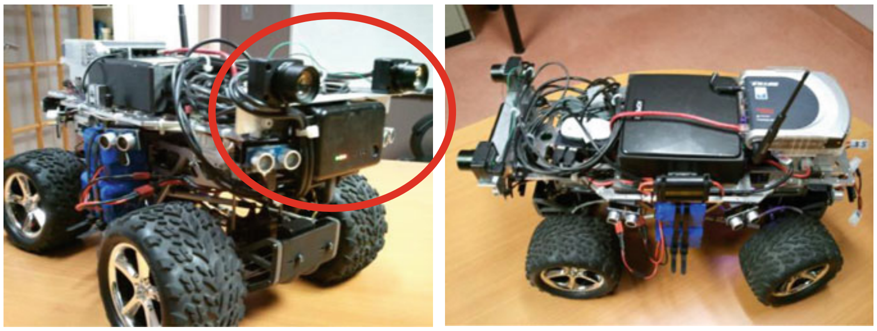


Fig. 9.15 University of Maryland automatic systems lab robot with TRX INU and machine vision cameras

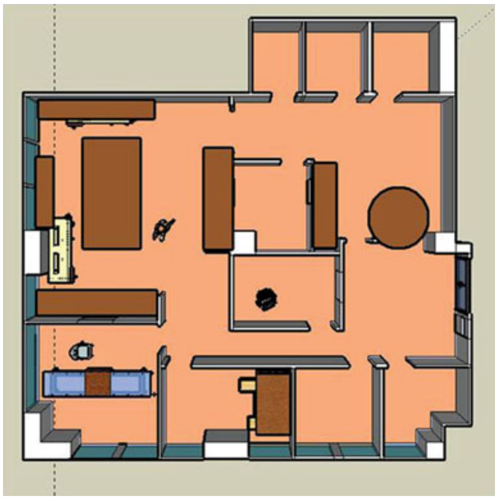


Fig. 9.16 CAD drawing of the test location

tracker, was implemented that uses SURF features for detection and stereo matching then modifies them to use Lucas–Kanade feature tracking over time (Karvounis 2011a, c).

Figure 9.16 shows a CAD drawing of the test location layout with approximate location of furniture.

9.4.2.1 Robust Kalman Filter Versus Standard Extended Kalman Filter

Tests were first run to show the performance of the Robust Kalman Filter versus the standard Extended Kalman Filter for integrating optical and inertial measurements. While the robots have capabilities for autonomous operations, data was collected by remotely controlling the robot in

1. a short path around the lab table,
2. a long path around the center offices,
3. a figure-eight path around the center offices and lab table, and
4. four laps around the center offices.

Images were collected at 20 frames per second and extracted/matched features were logged at each time step to enable a comparison of exactly the same path/ features for each algorithm.

On these simple paths traversed by a robot in a lab, it was demonstrated that the Robust Kalman Filter did not suffer from poorly matched or tracked features and produced a path within about a meter of the true path as long as 50 % of the observed features align with the expected model (Hampel et al. 1986; Karvounis 2011a). The standard Kalman Filter (with no pre-filtering), on the other hand, drastically altered the estimated position of the robot inducing an error of over 60 m in location.

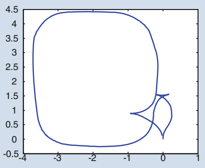
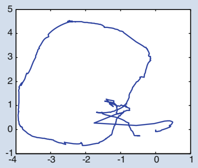
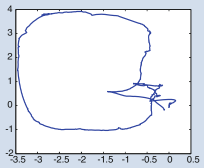
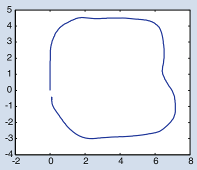
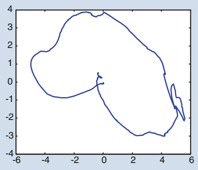
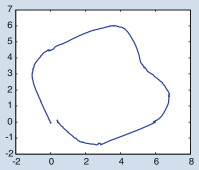
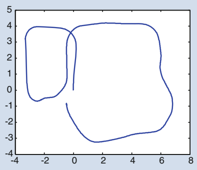
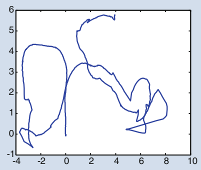
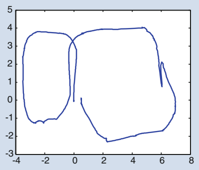
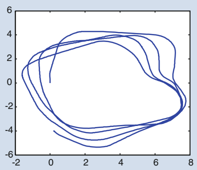
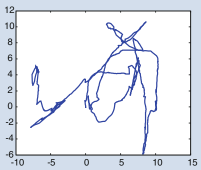
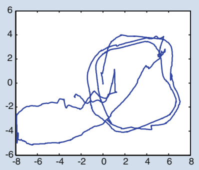
Forgetful-SLAM using a linearized version of a Robust Kalman Filter was evaluated over a series of robot paths and compared with a path based on wheel encoders and gyroscopic angle and Forgetful-SLAM using a standard EKF (without outlier removal). The encoder/gyro path is included as a baseline for performance if the controller has access to other vehicle sensors; optical SLAM algorithms are not used in this computation of this path.

Table 9.1 gives an estimate percent error over distance travelled for each path/ filter. The wheel encoders measure the total distance travelled, and the error is determined by how far the end of the path is from the true location. In each of the test paths, the true location of the end of path is the same as the start location. While percent of distance travelled is not the best metric in for tracking system performance (Chap. 8, section Accuracy Metrics), it allows a comparison of performance of different algorithms on the same base data set when a system for measure ground truth course data is not available. Note that, scaling error is not captured by this metric because the path begins and ends at the same point.

Above each result is a set of small images of the path. This shows the various path shapes and allows us to see visually how each filter performed on the path. The encoder and gyroscope path has error introduced by deviations in distance and by small amount of gyroscopic drift, however the path shape over the first three short tests is visually close to the actual course and the effect of drift is clear in the final test. The EKF and Robust Kalman Filter are able to reduce drift but are affected by outliers.

The Robust Kalman Filter consistently matches shape to the baseline encoder and gyroscopic path. The EKF path suffers significantly from outliers, causing the path to be distorted. The Robust Kalman Filter performed the best by a significant margin in some cases but was out performed by the encoder/gyro path in one of the shorter paths. As the path length/complexity increased, the Robust Kalman Filter showed more consistently good performance. The standard EKF always performed worse because it was unable to reject the outliers.

Table 9.1 Path snapshot and filter performance comparison using percent error over distance travelled. Eight paths used to compare inertial paths, Robust Kalman Filter paths, and Extended Kalman Filter paths

Path	Wheel Encoder and Gyroscope	Forgetful SLAM using an EKF	Forgetful SLAM using RKF
Short Clockwise Path			
Short Clockwise	0.85%	2.75%	1.06%
Long Clockwise Path			
Long Clockwise	2.44%	2.48%	1.76%
Figure-Eight Path			
Figure-Eight	2.75%	18.41%	1.35%
Four Loop Path			
Four Loop	3.85%	8.50%	2.65%

While the Robust Kalman Filter appears to remove most outliers, it is not entirely immune to outliers; the four loop path shows anomalies indicating outliers are present. The median filter outlier rejection rule only works when outliers make up less than 50 % of the samples. If, for example, a moving object covers the entire image frame, there is no guarantee that there are any correctly tracked features. These anomalies may be able to be corrected if the higher level HAR SLAM algorithm were performed to allow a global map to be created. Without the global map, loop closure is not performed.

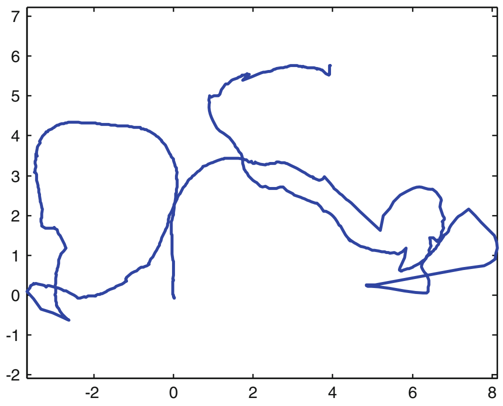


Fig. 9.17 EKF-SLAM

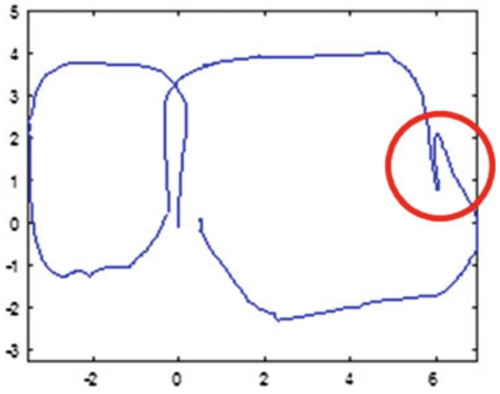


Fig. 9.18 Forgetful-SLAM

9.4.2.2 HAR-SLAM

In the next test, we reexamine the data from the figure-eight loop, to demonstrate that the loop closure in HAR-SLAM is able to correct errors caused by outliers.

To illustrate the importance of feature management methods in any real-time implementation of SLAM, an implementation of EKF-SLAM was run with no attempt to prune selected features, neither to remove outliers nor to reduce computation. The EKF-SLAM implementation (with every single feature ever seen saved!) took approximately 16 h to compute the path estimate for a 2 min path (Karvounis 2011a, c). Even with the extensive time taken, the result has many errors induced by outliers (Fig. 9.17). Simple feature management methods can improve this considerably.

In Forgetful-SLAM, a Robust Kalman Filter is used (Karvounis 2011a), which significantly reduces, but does not eliminate, the affect of outliers on the solution. An error caused by an outlier can be seen at (1, 6) in Fig. 9.18. On the same computer, Forgetful-SLAM took about 5 min to run the same path. While not yet real-time, this is a huge improvement over the EKF-SLAM running time.

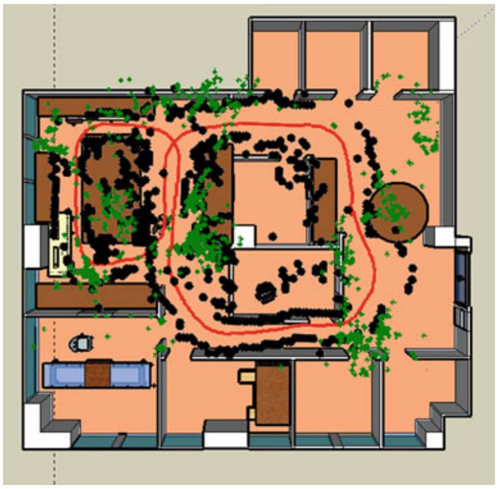


Fig. 9.19 HAR-SLAM map and path drawn on *top* of the CAD drawing

Closing the loop in HAR-SLAM by recognizing a feature and correcting the location enables the errors caused by outliers to be almost completely eliminated. HAR-SLAM took an additional minute to run, making it 6 min total. Figure 9.19 shows the HAR-SLAM map features (gray), filtered sonar data (black) and the HAR-SLAM path (red) overlaid on the CAD drawing. The HAR-SLAM path is very close to the true path.

9.4.3 Map-Joining

HAR-SLAM allows multiple robots to join maps on the fly in near real-time; whereas, other algorithms such as SEIF (Sparse Extended Information Filter) proposed joining maps in batch mode (Dissanayake et al. 2001). As depicted in the HAR-SLAM flow diagram in Fig. 9.12, as landmarks are promoted, a function is run to check for loop closures and another to compare features to map data from other tracked subjects in order to merge the features into a joint map. The promotion criteria is potentially different for the individual subject's global map which used to determine loop closure and the joint map, higher confidence being required to be promoted to the joint map. Landmarks promoted to the joint map are and shared among all tracked subjects on the network. Each landmark promoted to the joint map is check for matching by each individual. The same landmark matching technique that is used in loop-closing is used to determine matches between maps.

In order to join maps, a coordinate transform from each tracked subject's local coordinate system into the common coordinate system must be estimated. The transform consists of a translation to move the origin of the local coordinate

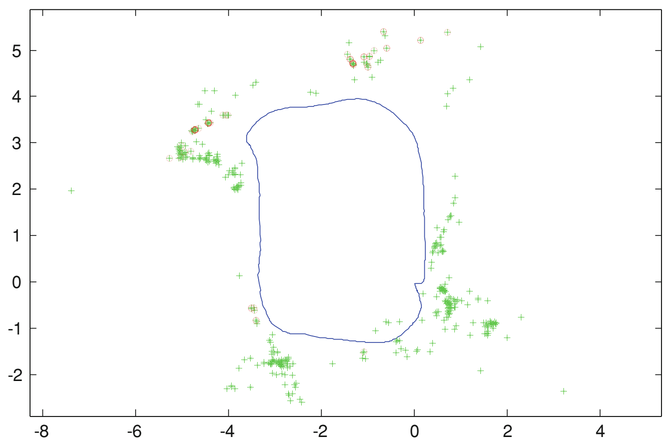


Fig. 9.20 Small office loop

system to that of the common coordinate system, and rotation to align the axes. Because of the uncertainty in map features, the global coordinate manager uses an EKF to estimate the coordinate transform for each track subject. Details of the update can be found in Karvounis (2011a, c) along with some discussion of a method for improving robustness.

The global coordinate manager can update the coordinate transform estimates as often as each time the coordinate manger detects that two or more tracked subjects have seen the same feature, but this is not necessary and computationally it may be too expensive. Each update can affect large portions of the joint map, and as those features are moved, in turn, each individual subject's map and historical track must also be updated. To conserve resources the update might be operated at some fixed interval or after some fixed number of feature matches are detected.

Figures 9.20, 9.21, and 9.22 show the HAR-SLAM results from three independent paths taken in the same test location. The associated features for each path are indicated by '+'. In one path the robot loops around a lab table (small loop), in another the robot loops around the lab table and the center offices (small and big loop), and in a third the robot loops around the center offices (big loop). In addition to path corrections, HAR-SLAM maintains a map of landmarks/promoted features. In each of the figures, the promoted features are circled.

This ability to merge the maps relies on selecting robust optical features. In this experiment only a few features are selected for promotion. The joint map is created by matching promoted features and then performing a global coordinate transform.

Figure 9.23 shows the three paths and the features in the joint map. Many of the selected landmarks are brought within close range of each other.

This example shows the promise of near-real-time joint map discovery using optical features but more work is needed in variable environments (including natural features, variable lighting, moving objects, etc.).

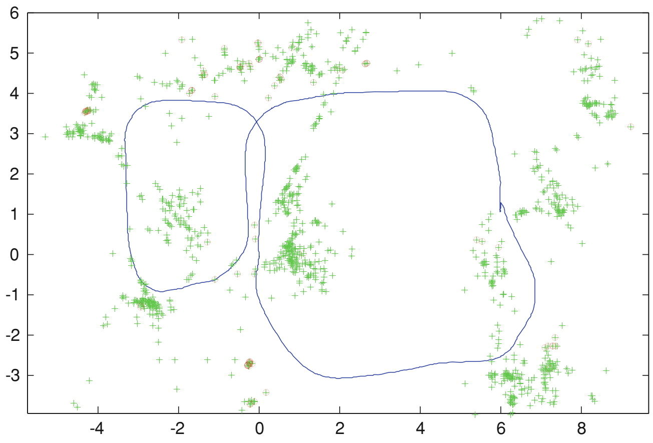


Fig. 9.21 Large and small loops

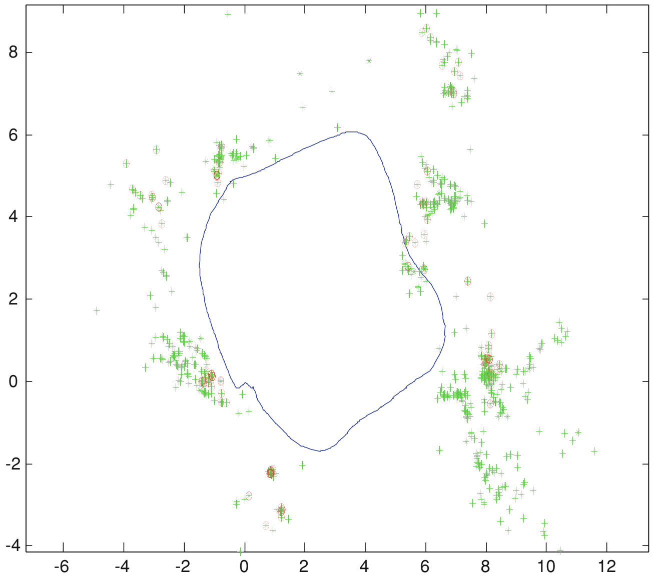


Fig. 9.22 Large office loop

9.5 Summary

This chapter gave an overview of localization and mapping with a focus on near real-time implementation. Methods for feature detection using selected allothetic sensors were reviewed. It is possible to create maps that enable long-term tracking with good accuracy by combining allothetic feature information with the idiothetic

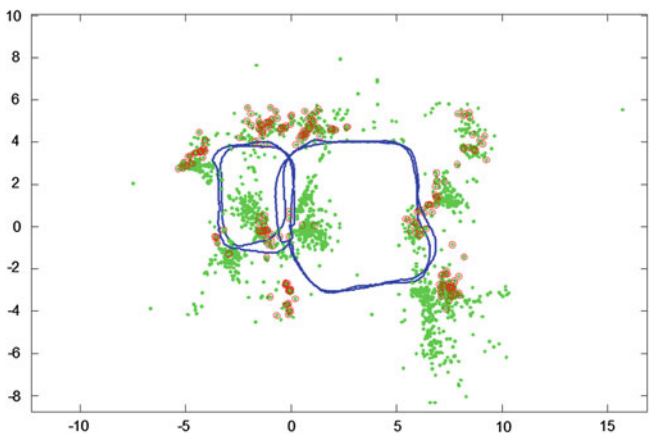


Fig. 9.23 Joint map with optical features

inertial tracking data (Chap. 8), using their complementary characteristics to compensate for sensor drift and allow disambiguation of perceptually aliased features. This ability to simultaneously localize and map is called SLAM.

A probabilistic SLAM problem formulation was given and different approaches for obtaining theoretical solutions were reviewed. A major portion of the chapter was focused on a hierarchical implementation of SLAM (HAR SLAM) designed to address some of the practical implementation issues including rejection of anomalous feature measurements and management of tracked features. A main advantage of HAR-SLAM is that it provides a structure for managing computational cost which facilitates real-time implementation.

The HAR SLAM algorithm was shown to be able join maps created from different traversals of the same environment using only a small subset of good features from each traversal. This early result shows promise for the use of this algorithm in crowd source mapping. This work will be discussed in a later paper.

References

A.E. Abdel-Hakim, A.A. Farag, in *CSIFT*: *A SIFT Descriptor with color invariant characteristics*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2006)

A. Bandyopadhyay, D. Hakim et al., System and method for determining location of personnel and/or assets both indoors and outdoors via map generation and/or map matching techniques. USPTO. US, TRX Systems. Utility (2008)

H. Bay, A. Ess et al., SURF: Speeded up robust features. Comput. Vis. Image Underst. (CVIU) **110**(3), 346–359 (2008)

H. Bay, T. Tuytelaars et al., *SURF*: *Speeded Up Robust Features*. ECCV (2006)

J. Borenstein, Heading Error Removal System for Tracking Devices USPTO, University of Michigan. US 2010/0256939 Al (2010)

G. Dissanayake, P. Newman et al., A solution to the simultaneous localization and map building (SLAM) problem. IEEE Trans. Robot. Autom. **17**(3), 229–241 (2001)

H. Durrant-Whyte, Uncertain geometry in robotics. IEEE J. Robot. Autom. **4**(1), 23–31 (1988)

H. Durrant-Whyte, T. Bailey, Simultaneous Localization and Mapping: Part I. IEEE Robotics & Automation Magazine (June): 99–108 (2006a)

H. Durrant-Whyte, T. Bailey Simultaneous Localization and Mapping (SLAM): Part II. IEEE Robotics & Automation Magazine (September): 108–117 (2006b)

P. Elinas, R. Sim et al., in *σSLAM*: *Stereo vision SLAM using the Rao-Blackwellised particle filter and a novel mixture proposal distribution*. International Conference on Robotics and Automation, Orlando, Florida, IEEE (2006)

D. Filliat, J.-A. Meyer, Map-based navigation in mobile robots: I. a review of localization strategies. Cog. Sys. Res. **4**(4), 243–282 (2003)

B. Funk, A. Bandyopadhyay et al., Method and system for locating and monitoring first responders. USPTO. US, TRX Systems. 0077326, (2007)

J.E. Guivant, E.M. Nebot, Optimization of the simultaneous localization and map-building algorithm for real-time implementation. IEEE Trans. Robot. Automat. **17**(3), 242–257 (2001)

F. Gustafsson, F. Gunnarsson et al., Particle filters for positioning, navigation and tracking. IEEE Trans. Signal Process. **50**(2), 425–437 (2002)

F.R. Hampel, E.M. Ronchetti et al., *Robust Statistics: The Approach Based on Influence Functions* (Wiley, New York, 1986)

C. Harris, M. Stephens, in *A Combined Corner and Edge Detector*. Proceedings of the 4th Alvey Vision Conference (1988)

V. Indelman, S. Williams et al., in *Factor graph based incremental smoothing in inertial navigation systems*. International Conferences on Information Fusion (2012)

M. Kaess, S. Williams et al., in *Concurrent Filtering and Smoothing*. International Conference on Information Fusion (2012)

T. Kailath, *Linear Systems* (Prentice Hall, Englewood Cliffs, 1980)

J. Karvounis, Theory, Design, and Implementation of Landmark Promotion Cooperative Simultaneous Localization and Mapping. Electrical and Computer Engineering. College Park, University of Maryland. Ph.D (2011a)

J. Karvounis, Robust Kalman Filter. Joint Navigation Conference. Colorado Springs, CO, ION, (2011b)

J. Karvounis, Theory, Design, and Implementation of Landmark Promotion Cooperative Simultaneous Localization and Mapping. US—Provisional Patent, TRX. Provisional (2011c)

J. Kim, S. Sukkarieh, in *Improving the Real-Time Efficiency of Inertial SLAM and Understanding its Observability*. International Conference on Intelligent Robots and Systems, Sendai, Japan, (2004)

D. Koller, N. Friedman, Probabilistic Graphical Models: Principles and Techniques (Adaptive Computation and Machine Learning series), (MIT Press, Cambridge 2009)

T. Lemaire, S. Lacroix, SLAM with panoramic vision. J. Field Robot. **24**(1–2), 91–111 (2007)

T. Lindeberg, Feature detection with automatic scale selection. IJCV **30**(2), 79–116 (1998)

H.-A. Loeliger, An Introduction to Factor Graphs. IEEE Signal Processing Magazine (2004)

D.G. Lowe, in *Object recognition from local scale-invariant features*. International Conference on Computer Vision, (1999)

D.G. Lowe, Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)

J.-A. Meyer, D. Filliat, Map-based navigation in mobile robots: II. A review of map-learning and path-planning strategies. Cog. Sys. Res. **4**(4), 283–317 (2003)

J.V. Miro, G. Dissanayake, et al., *Vision-based SLAM using natural features in indoor environments*. Intelligent Sensors, Sensor Networks and Information Processing Conference, IEEE, (2005)

M. Montemerlo, S. Thrun, in *Simultaneous Localization and Mapping with Unknown Data Association using Fast SLAM*. Proceedings of the IEEE International Joint Conference on Robotics and Automation, (2003b)

M. Montemerlo, S. Thrun, et al., in *Fast SLAM 2.0*: *An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges*. International Joint Conference on Artificial Intelligence (2003a)

M. Muller, *Dynamic Time Warping. Information Retrieval for Music and Motion* (Springer, Berlin, 2007)

B. Ristic, S. Arunlampalam et al., Beyond the Kalman Filter Particle Filters for Tracking Applications, Artech House, (2004)

P. Robertson, M. Angermann, et al., Simultaneous Localization and Mapping for Pedestrians using only Foot-Mounted Inertial Sensors Ubicomp Orlando, Florida, (2009a)

P. Robertson, M. Angermann et al., in *Inertial Systems Based Joint Mapping and Positioning for Pedestrian Navigation*. ION GNSS. (Savannah, Georgia, 2009b)

P. Robertson, M. Angermann et al., in *SLAM Dance*: *Inertial-Based Joint Mapping and Positioning for Pedestrian Navigation*. Inside GNSS (2010)

H. Sakoe, S. Chiba, Dynamic programming algorithm optimization for spoken word recognition. IEEE Trans. Acoust. Speech Signal Process **26**(1), 159–165 (1978)

K. van de Sande, T. Gevers et al., Evaluating color descriptors for object and scene recognition. IEEE Trans. Pattern Anal. Mach. Intell. **32**(9), 1582–1596 (2010)

S. Se, D.G. Lowe et al., Vision-based global localization and mapping for mobile robots. Transactions on Robotics, IEEE, (2005)

R. Sim, P. Elinas et al., Vision-based SLAM using the Rao-Blackwellised Particle Filter. IJCAI Workshop on Reasoning with Uncertainty in Robotics, IJCAI, (2005)

R. Smith, P. Cheeseman, On the representation of spatail uncertainty. Int. J. Robot. Res. **5**(4), 56–68 (1986)

R. Smith, M. Self et al., *Estimating uncertain spatial relationships in robotics*. ed. by I.J. Cox, G.T. Wilfon. Autonomous Robot Vehicles. (Springer, New York, 1990), pp. 167–193

S. Thrun, W. Burgard et al., *Probablistic Robotics* (MIT Press, Cambrige, 2006)

S. Thurn, M. Montemerlo, The graph SLAM algorithm with applications to large-scale mapping of urban structures. Int. J. Robot. Res. **5**(6), 403–429 (2005)

M.J. Veth, Navigation using images, a survey of techniques. J. Inst. Navig. **58**(2), 127–139 (2011)

H. Wang, S. Sen et al., Unsupervised Indoor Location. MobiSys, (2012)