Consistent, Convergent, and Constant-Time SLAM*

J. Leonard and P. Newman †

Abstract

This paper presents a new efficient algorithm for simultaneous localization and mapping (SLAM), using multiple overlapping submaps, each built with respect to a local frame of reference defined by one of the features in the submap. The global position of each submap is estimated using information from other submaps in an efficient, provably consistent manner. For situations where the mobile robot is able to make repeated visits to all regions of the environment, the method achieves convergence to a near-optimal result with $\mathcal{O}(1)$ time complexity while maintaining consistent error bounds. Simulation results demonstrate the ability of the technique to converge to errors that are only slightly greater than the full solution, while maintaining consistency.

1 Introduction

The capability of simultaneous localization and mapping (SLAM) is considered vital for the creation of long-lived autonomous mobile agents. Because of the difficulties encountered by SLAM algorithms when applied to larger environments, the "map scaling" problem has been identified as one of the key issues for research in this area. In this paper, we adopt the feature-based, state-space formulation of SLAM [Smith *et al.*, 1987]. Recent related work in SLAM includes submap decomposition methods [Leonard and Feder, 2000; Guivant and Nebot, 2001; Julier and Uhlmann, 2001; Williams *et al.*, 2002; Tardós *et al.*, 2002], FastSLAM [Montemerlo *et al.*, 2002], sparse extended information filters (SEIF's) [Thrun *et al.*, 2002], scan-matching [Gutmann and Konolige, 1999; Thrun, 2001; Hahnel *et al.*, 2002] and topological approaches [Kuipers and Beeson, 2002; Choset and Nagatani, 2001].

This paper examines the SLAM problem using submaps. While there has been a considerable amount of work in SLAM along these lines, no previous method satisfies each of the three criteria of (1) provable consistency, (2) spatial convergence, and (3) constant-time updates. For example, Julier and Uhlmann [2001] provide a consistent, constant-time algorithm for large-scale SLAM, based on split covariance intersection, but this method does not achieve "tight" convergence to the error bounds that would be obtained with a full covariance solution. Methods such as the compressed filter [Guivant and Nebot, 2001], the constrained local submap filter [Williams *et al.*, 2002], and sequential map joining [Tardós *et al.*, 2002] are provably consistent and convergent, but are $\mathcal{O}(n^2)$, where n is the number of the features in the environment. Other techniques such as decoupled stochastic mapping (DSM) [Leonard and Feder, 2000] and SEIF's [Thrun *et al.*, 2002] achieve $\mathcal{O}(1)$ performance, but make approximations that require empirical testing to verify state estimation consistency. The Atlas framework for large-scale SLAM [Bosse *et al.*, 2003] achieves constant-time performance during the motion of the robot, enabling closing of large loops, but does not compute state estimates with respect to a single global reference frame.

The FastSLAM technique stands out in the literature as perhaps the only published technique which has been successfully posed for the general nonlinear SLAM problem with computational effort $\mathcal{O}(\log(n))$. The performance of FastSLAM, however, depends linearly on a "parameter (the number of particles), whose scaling with environmental size is still poorly understood [Thrun *et al.*, 2002]".

This paper considers SLAM with known data association. This is a major assumption, but it enables us to focus on the underlying structure of SLAM state estimation with submaps for the linear Gaussian case. Other research has shown the ability to perform feature detection and measurement association using techniques such as random sample consensus and Joint Compatibility Testing [Neira and Tardós, 2001].

In this paper, we demonstrate how the three criteria of consistency, convergence, and constant-time updates can be achieved with multiple, locally referenced submaps. The key idea is to decouple the estimate of a map's global location from any of the state estimates within the map. This "inside-out" decomposition strategy succeeds because state estimates from one map are never mixed with state estimates from other

*This research has been funded in part by NSF Career Award BES-9733040, the MIT Sea Grant College Program under grant NA86RG0074, and the Office of Naval Research under grant N00014-97-0202.

†P. Newman is at the Robotics Research Group, University of Oxford and J. Leonard is at the Department of Ocean Engineering at MIT, email: pnewman@robots.ox.ac.uk,jleonard@mit.edu

maps. This enables us to guarantee the consistency of the state estimates for the linear Gaussian case.

The structure of this paper is as follows. After reviewing the full (single map) SLAM problem in Section 2, we define our terminology in Section 3. Section 4 summarizes the new SLAM algorithm and Section 5 discusses its consistency and convergence properties. Section 6 describes the performance of the algorithm using simulations and Section 7 provides a concluding discussion.

2 SLAM within a single map

Let us assume that there are n features in the environment, and that they are static. The global frame, designated by G, is a unique, immutable coordinate-frame that is defined at the beginning of a mission. The true state at time k is designated by $\mathbf{x}(k) = [\mathbf{x}_v(k)^T\ \mathbf{x}_1(k)^T \ldots \mathbf{x}_n(k)^T]^T$, where $\mathbf{x}_v(k)$ represent the location of the vehicle, and $[\mathbf{x}_1(k)^T \ldots \mathbf{x}_n(k)^T]^T$ represents the locations of the environmental features. We assume that the vehicle moves from time k to time $k+1$ in response to a known control input, $\mathbf{u}(k)$, that is corrupted by noise. Let U^k designate the set of all control inputs from time 0 through time k, $Z(k)$ designate the set of sensor measurements obtained at time k, and Z^k designate the set of all measurements obtained from time 0 through time k. For each measurement $\mathbf{z}_j(k) \in Z(k)$, there is a corresponding assignment index $\mathbf{a}_j$. The value of $\mathbf{a}_j$ is i if measurement $\mathbf{z}_j(k)$ originates from feature i. Let A^k designate the set of all assignment indices from time 0 through time k. Assuming that the associations are known, the objective is to compute recursively the probability distribution for the location of the robot and the features, with reference to the global reference frame G, given the measurements, control inputs, and assignments:

$$p(\mathbf{x}_v(k), \mathbf{x}_1(k), \ldots, \mathbf{x}_{n_k}(k) | Z^k, A^k, U^{k-1}). \tag{1}$$

For the Linear-Gaussian (LG) SLAM problem, the Kalman filter provides the optimal estimate of this pdf, which is described by its mean $[\hat{\mathbf{x}}_v(k)^T \hat{\mathbf{x}}_1(k)^T \ldots \hat{\mathbf{x}}_n(k)^T]^T$ and covariance $\mathbf{P}(k)$. The properties of single-map LG SLAM solution are well-known [Dissanayake *et al.*, 2001].

3 Machinery for SLAM using multiple maps

We now proceed to define several terms and basic operations that will facilitate description of the new method. A **location vector** is a parameterization of both position and orientation of one coordinate-frame, i with respect to another, j. In $\mathcal{R}^2$ this is represented as a translation by $[x, y]$ followed by a rotation θ. These three parameters are encapsulated in the 3 vector $T_i^j = [x, y, \theta]^T$.

An **entity** is a parameterization of a vehicle or landmark. Each entity is labelled with a unique positive integer — this is referred to as the entity's ID. We can attach a coordinate frame $\mathcal{F}_i$ to any entity i and describe it using a location vector in another coordinate frame. The vector T_i^j should be understood to be a parameterization of a transformation from $\mathcal{F}_i$ to $\mathcal{F}_j$. The uncertainty in this transformation is represented by Σ_i^j.

A **map** is a collection of entities all described with respect to a local coordinate frame. Each map has a unique integer id. Each map has associated with it a **Map Root** entity i and a **Map Location** vector T_m^G. The Map Location vector describes the pose of a map's local coordinate frame in the global frame G. The local coordinate frame of a map is coincident with one of the entities in the local map; this entity is referred to as the Map Root. In other words, the Map Root lies at the origin of the local map, and it is the entity to which all other entities in a map are referenced. If an entity is the Map Root, then by definition its location vector will be $[0, 0, 0]^T$ and its global location vector given by T_m^G — the location of the map in global coordinates.[1] We use the pre-superscript notation $[m, j]$ to denote that all entities in map m are referenced to $\mathcal{F}_j$ where j is the root entity of the map.

The notation is summarized as:

$$^{[\text{mapid,rootid}]}\text{Variable}^{\text{w.r.t. id}}_{\text{variable id}}$$

Using this notation we can write the transformation from $\mathcal{F}_j$ to $\mathcal{F}_k$ in map m with root entity i as $^{[m,i]}T_k^j$. We simplify notation by dropping the right superscript when describing a transformation with respect to the map root:

$$^{[m,i]}T_j^i \rightarrow {}^{[m,i]}T_j$$

The term $^{[m,i]}T_j$ is the pose of an entity j in the local frame of the mth map (which has its Map Root as entity i).

We can manipulate location vectors using the binary transformation operator $\oplus$ and the unary operator $\ominus$ where

$$T_j^i = \ominus T_i^j$$
$$T_k^i = T_j^i \oplus T_k^j$$

Using the notation described above, we can describe the relationship between entities in any given map. Consider the case of map m referenced to feature i denoted $[m, i]$. Taking two entities j and k from m, we can express the transformation from j to k as

$$\begin{aligned} {}^{[m,i]}T_k^j &= {}^{[m,i]}T_i^j \oplus {}^{[m,i]}T_k^i \\ &= \ominus {}^{[m,i]}T_j^i \oplus {}^{[m,i]}T_k^i \\ &= \ominus {}^{[m,i]}T_j \oplus {}^{[m,i]}T_k \end{aligned} \tag{2}$$

where the last step uses the simplification in notation described above.

We define **root-shifting** to be the operation, $\mathcal{S}$, that changes the root of a map from i to j. After this operation all location vectors in a map will be referenced to $\mathcal{F}_j$ rather than $\mathcal{F}_i$, $i, j \in m$. This operation is is simply an extension of Equation 2 to act on all entities in the map:

$$^{[m,j]}T_{1:n} = \mathcal{S}_{i \rightarrow j}(^{[m,i]}T_{1:n}) \tag{3}$$

$$= \begin{bmatrix} \ominus {}^{[m,i]}T_j \oplus {}^{[m,i]}T_1 \\ \vdots \\ \ominus {}^{[m,i]}T_j \oplus {}^{[m,i]}T_n \end{bmatrix} \tag{4}$$

[1] In the SLAM literature, the term "base reference" [Tardós *et al.*, 2002] is a synonym for our term Map Root. Note that in the general case with orientation, a single point feature will be insufficient to define a reference frame. In 2-D, two points will be required and in 3-D three points will be required.

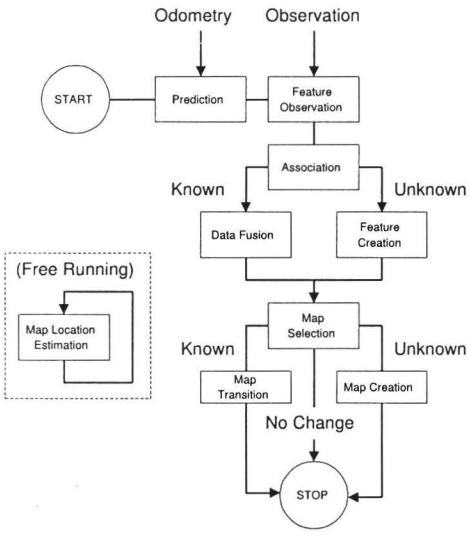

Figure 1: Flow chart for each cycle of the algorithm.

The global location of a feature j in a local map ${}^{[m,i]}T_j$ is computed by simple composition of the local location vector and the already globally referenced Map Location vector:

$$ {}^{[m,\cdot]}T_j^G = T_m^G \oplus {}^{[m,i]}T_j \tag{5} $$

4 The constant-time SLAM algorithm

Figure 1 illustrates the processing steps that are performed by the algorithm. The four key elements are (1) SLAM processing within local maps, (2) Map management (performing transitions between maps and creating new maps), (3) Map Location estimation (determining the best global location estimate for a submap), and (4) Computation of global state estimates for all features in a map. Each of these processes are described in detail below.

4.1 SLAM within local maps

At any one time, there is a single active map. For each map m, we compute a partial solution, $p(\mathbf{x}_m(k)|Z_m^k, A_m^k, U_m^{k-1})$, where $\mathbf{x}_m(k)$ designates the local map state, and Z_m^k, A_m^k, and U_m^{k-1} represent subsets of the measurements, associations, and control inputs, respectively. Each measurement is used in only a single map. (This is vital for ensuring consistency of the global Map Location estimation process.) Each map m contains an estimated mean $[\hat{\mathbf{x}}_v^m(k)^T \hat{\mathbf{x}}_1^m(k)^T \ldots \hat{\mathbf{x}}_n^m(k)^T]^T$ and covariance $\mathbf{P}^m(k)$ corresponding to selected vehicle locations (for time steps when map m is the active map), and only a subset of the features. These estimates are the same as the location vectors ${}^{[m,i]}T_j$ and associated uncertainty ${}^{[m,i]}\Sigma_j$ for the features in the local map.

4.2 Map management

In our scheme, each map has a center, which is defined as the vehicle location at the time of the creation of a map. About this center is defined a region of radius r. This defines a bound of vehicle location and not feature locations — any feature that is observed from a position inside the map region will be added to the local map. The estimated location of the vehicle is used to deduce which map(s) the vehicle is in, and when to make transitions. When the vehicle travels more than $r+h$ from the center, the vehicle is considered to have left the current map. The parameter h is a hysteresis term, to prevent excessive map switching. In the simulation results below, we use $r = 15$ and $h = 5$. (An alternative to circular map regions is the use of convex hulls, but this makes no difference to the fundamental performance of the algorithm.)

We assume that the density of discernible features in a local area is bounded. This provides a bound on the number of features that can belong to a map.

When a vehicle leaves a map, we must determine which map (if any) the vehicle has transitioned to. A list of possible candidates is drawn up from a look up table indexed by quantized vehicle locations. If more than one candidate exists, we choose the map with the lowest ID, i.e., the oldest map. If no candidates are found, then a new map is created at the current location (a distance $r + h$ from the center of the previous map). All of these operations can be performed in constant time.

4.3 Map Location estimation

Map Location estimation is the procedure by which global estimates for feature locations in each local map are improved, resulting in global convergence. This procedure is described as follows:

1. Select a map p, to improve which is currently referenced to the root entity i.
2. Create a set, $\mathcal{N}$, containing the ID's of all nearby maps including p — the map to be improved.
3. For each $q \in \mathcal{N}, \quad q \neq p$ create a set $\mathcal{C}_{p,q}$ of ID's of features that are present in both maps.
4. For the frame $\mathcal{F}_k$ attached to feature $k \in \mathcal{C}_{p,q}$, calculate its globally referenced location ${}^{[q,\cdot]}T_k^G$ and uncertainty ${}^{[q,\cdot]}\Sigma_k^G$ using the location estimate of feature k within map q and its current location estimate.
5. Pick the map $q^\star$ and entity ID $k^\star$ such that:
$$ [q^\star, k^\star] = \arg\min\{|\ {}^{[q,\cdot]}\Sigma_k^G\ |\} $$
6. If $q^\star = p$ and $k^\star = i$ then stop. The map p cannot be improved
7. Root Shift map p to $k^\star$ from i — reference all entities in map p to a coordinate frame attached to entity $k^\star$.
8. Replace T_p^G and Σ_p^G with ${}^{[q^\star,\cdot]}T_{k^\star}^G$ and ${}^{[q^\star,\cdot]}\Sigma_{k^\star}^G$ respectively.

For constant-time operation, Map Location estimation is performed only when the vehicle transitions from one map to another. Alternatively, the procedure can be performed periodically (or at the end of the mission) to all maps. Multiple iterations result in global convergence to a near-optimal solution, but the computation complexity is no longer $\mathcal{O}(1)$.

The work presented in this paper differs from that proposed in [Leonard and Feder, 2000] on the following counts:

- No vehicle information is carried between maps upon map transitions
- The SLAM scheme adopted within each map need not be based on a Kalman filter.
- Mapped entities are represented in local coordinate frames — one frame per map. In contrast, the DSM approach used multiple maps but all registered in one global coordinate frame.

4.4 Obtaining global location estimates for map features

Given an independent, consistent estimate of the location of a submap p (with root i) with respect to the global frame, G, we can produce a consistent global estimate of the location of any feature j in map p by composition of map and feature locations:

$$^{[p,\cdot]}T_j^G = T_p^G \oplus {}^{[p,i]}T_j \tag{6}$$

This estimate is consistent because T_p^G and $^{[p,i]}T_j$ are independent. $^{[p,i]}T_j$ is "internal" to the map and T_p^G is "external" to the map.

The existence of a "shared" feature, s, between two maps p and q allows the location estimate T_p^G, of map p, (with root feature s), to be replaced with T_{p+}^G where

$$T_{p+}^G = T_q^G \oplus {}^{[q,w]}T_s \tag{7}$$

and w, the root of map q is any feature id in map q. Equation 7 should be interpreted as finding an alternative expression for the global location of a shared feature s using quantities associated with map q instead of p. As map p has a root at the shared feature s this expression is by definition an alternative expression for the map location. The minimization step of the algorithm is concerned with finding the best choice of shared feature s.

In the limit each map becomes internally fully correlated and the "min" operation will have no further effect so that for any feature j

$$T_p^G \oplus {}^{[p,\cdot]}T_j = T_q^G \oplus {}^{[q,\cdot]}T_j \tag{8}$$

where $\cdot$ is any choice of root. In other words no root-shifting and replace operation can be found that improves the global uncertainty of feature j. For the linear case

$$\Sigma_p^G + {}^{[p,\cdot]}\Sigma_j = \Sigma_q^G + {}^{[q,\cdot]}\Sigma_j \tag{9}$$

The maps are rooted on features and so as $k \to \infty$ then

$$^{[p,\cdot]}\Sigma_j = {}^{[q,\cdot]}\Sigma_j = 0 \tag{10}$$

and so

$$\Sigma_p^G = \Sigma_q^G \tag{11}$$

which is true for all choices of p and q. Therefore the globally referenced feature location uncertainty $^{[\cdot,\cdot]}\Sigma_j^G$ is the same independent of choice of map $(\cdot)$. The value of this limiting value is clearly given by the smallest possible uncertainty in Map Location which is the uncertainty of the first feature initialized in the first map [Dissanayake *et al.*, 2001]. This point is considered further in section 5.

5 Consistency and convergence

We begin by defining the term consistency with regard to an estimate $\hat{\mathbf{x}}(k|k)$ of an r.v $\mathbf{x}(k)$ at time k given all information up until time k. Defining the estimated error vector $\tilde{\mathbf{x}}(k|k) \triangleq \mathbf{x}(k) - \hat{\mathbf{x}}(k|k)$ and the estimate covariance as $P(k|k)$ we write the condition of consistency as:

$$E[\tilde{\mathbf{x}}(k|k)] = \mathbf{0} \tag{12}$$

$$E[[\tilde{\mathbf{x}}(k|k)][\tilde{\mathbf{x}}(k|k)]^T] \leq P(k|k) \tag{13}$$

To show that the global location estimates produced by Equation 6 are consistent, we rely on the following three properties: (1) local map state estimates $^{[p,i]}T_j$ (obtained from the local SLAM solution $[\hat{\mathbf{x}}_v^m(k)^T \hat{\mathbf{x}}_1^m(k)^T \ldots \hat{\mathbf{x}}_n^m(k)^T]^T$) are consistent, (2) global state estimates for Map Locations T_m^G are consistent, and (3) the composition of these two pieces of information — local state estimates within a map and global information concerning the location of the map — is consistent.

The consistency of local maps follows directly from the properties of the Kalman filter which in the linear gaussian case is the optimal Bayesian estimator. Clearly, choosing to use a possibly inconsistent estimator such as the EKF in a nonlinear scenario will invalidate these claims. However, the LG case allows statements to be made regarding the *underlying* properties of the CTS algorithm. In a non-linear implementation, the consistency of the LG case can be matched to an arbitrary degree by using Monte-Carlo estimators in each sub-map. Regardless of local estimation techniques, the CTS algorithm preserves its constant time property.

Local maps have three differences from the full solution (a) their base reference (root) is defined by one of the features in the map, (b) relocation is periodically performed to re-initialize the local map when the vehicle transitions back into it, and (c) the base reference of the local map is periodically shifted from one feature in the local map to another (Root Shifting). None of these three differences result in a loss of consistency for the local SLAM solution.

The global Map Location estimate T_m^G for a given map, m, is consistent because it is created via the composition of transformations derived from other local maps, and each local map is independent of other local maps. The composition of transformations from different local maps is a consistent operation (for the linear case).

Finally, the composition of the local map state estimates performed in Equation 6 is a consistent operation, because T_m^G and $^{[m,i]}T_j$ are independent of one another.

While the location estimates for different maps are correlated with one another, and this correlation is not computed by the algorithm, the method in none-the-less consistent because this correlation is never needed. We never fuse Map Location estimates estimates, but rather, perform wholesale replacement. The algorithm keeps track of the best estimate for the global location of the root entity of a given map. A guiding principle of this algorithm is that estimated quantities that are "external" to a map never effect an internal quantity.

Due to the relocation step for map transitions, the information from dead-reckoning measurements for the time step

proceeding the map transition is effectively "lost". This information, however, does not affect absolute convergence but only the rate of convergence. Because relocation is a consistent operation [Knight, 2002], each partial solution retains all the properties of a Kalman filter SLAM solution [Dissanayake *et al.*, 2001], and hence is provably consistent and convergent. Because each map is local with its base reference as one of the features in the map, in the limit the uncertainty for each local map converges to zero [Dissanayake *et al.*, 2001]. This implies that, in the limit, in any local map utilizing only a subset of Z^k, the relationship between features becomes perfectly known as $k \rightarrow \infty$. A consequence of this is that the covariance of the transformation between any two features, i and j, in any given map tends to zero as $k \rightarrow \infty$. Hence, for local maps in which the base reference (root) is a feature in the local map, the covariance of any feature tends to zero. The key driver for convergence is the behavior in the first submap — submap 1. To illustrate this, consider a robot which moves swiftly outside submap 1 and spends the rest of its mission driving just outside submap 1's borders. Eventually the maps bordering submap 1 become completely known, including most of the features which appear in submap 1. The global uncertainty of any feature in the bordering submaps can never be less than the global uncertainty in the location of submap 1. Hence, in this scenario, the root shifting mechanism can never decrease the global uncertainty of features in submap 1 (nothing has lower global uncertainty). However as soon as submap 1 is re-entered and begins to be refined, the global uncertainties of maps sharing features with map 1 can be reduced by root-shifting. Hence it is the precision of the first submap built (submap 1) that drives the ultimate performance of the entire system. In addition it is the precision of the first feature mapped within it that drives ultimate performance of submap 1.

Thus in the limit, the lower bound achieved in submap 1 is "inherited" by all other submaps. There are two differences between what occurs in submap 1 in comparison to a full covariance solution that couples estimates for all features in a single map: (a) submap 1 has fewer features in it, and (b) not all of the observations of features that are contained in submap 1 are processed in the submap 1 solution. We believe that consideration (a), the fact that submap 1 has fewer features in it, is what is sacrificed in this approach. Even if some measurements are ignored in submap 1 (vs. the full solution), in the limit as $k \rightarrow \infty$, both maps will converge to a well-defined lower bound. With enough additional time the submap 1 solution can "catch up" to the full solution. However, the fact that the full solution has more features enables in it cannot be compensated for, and hence the full solution achieves a slightly tighter bound. This in effect is the "cost" of computing multiple partial solutions and subsequently combining them, rather than computing one full solution. Our simulations have shown that this result is extremely small (as shown below in Figure 3).

6 Results

This section analyzes the behavior of the technique presented using simulations analysis. The simulations consider an (x, y) "point" vehicle that moves in the plane with process noise Q of 0.05 or 0.01(Monte-Carlo) m/sec standard deviation in both x and y and measurement noise R of 0.2 m/sec standard deviation in both x and y. Features are visible if they are within 23 meters of the vehicle location. One visible feature is selected at random each time step to generate the observations. Results are presented for three different scenarios:

1. a simulation involving six cycles through an environment in which eight maps are created (to illustrate basic error convergence behavior),
2. Monte-Carlo analysis of 200 independent trials of a mission involving ten cycles through an environment in which twelve submaps are created (for empirical consistency verification), and
3. results off-line map adjustment for a mission involving a single cycle through an environment in which eighteen maps are created,

6.1 Comparison with the full covariance solution

Figure 2 show the vehicle path for a mission in which eight maps are created. Figure 3 shows the vehicle error as a function of time as a function of time and active map ID. Spikes in the vehicle error estimate occur at each map transition when relocation (consistent re-initialization of the vehicle pose) is performed. This is particularly obvious towards the end of the experiment when the features are well known and subsequent observations swiftly bring the vehicle covariance down following relocation. Figure 4 shows the difference between the new method and the full solution in the determinant of error covariance for the feature marginals in each map, demonstrating the tight convergence of the method to near-optimal estimates, while never going below the minimum permissible error bounds.

6.2 Monte-Carlo consistency testing

For the purposes of consistency testing, two relative states were logged in the active submap: (1) the difference between the first feature in the state vector and the vehicle, and (2) the difference between the first and second features in the state vector. These two states are stored in a single vector $\hat{\mathbf{y}}^m$ with covariance $\mathbf{Y}^m$ formed from the active map (with id m). This relative vector can be compared to the 'true' relative relationships declared by the simulator producing the noise corrupted measurements. The difference between these vectors is the error vector $\tilde{\mathbf{y}}(k)$. The $\tilde{\mathbf{y}}$ vector calculated for each map m will in general involve different features (by definition as each map contains a subset of the set of all features with only some features in common). However each estimated component of $\hat{\mathbf{y}}^m$ should be consistent, hence any sequence $\{\hat{\mathbf{y}}^{m_1}(1), \ldots, \hat{\mathbf{y}}^{m_k}(k)\} \quad m_{[0:k]} \in [1 : M]$ where M is the total number of maps built, should also be consistent. Figure 6.2 illustrates the results obtained from Monte-Carlo experiments of the CTS algorithm. The parameters used in the simulation of sensor data and its subsequent processing are given in Table 6.2. All the plots are concerned with the statistical properties of $\tilde{\mathbf{y}}(k)$. The first plot of Figure 6.2 shows the

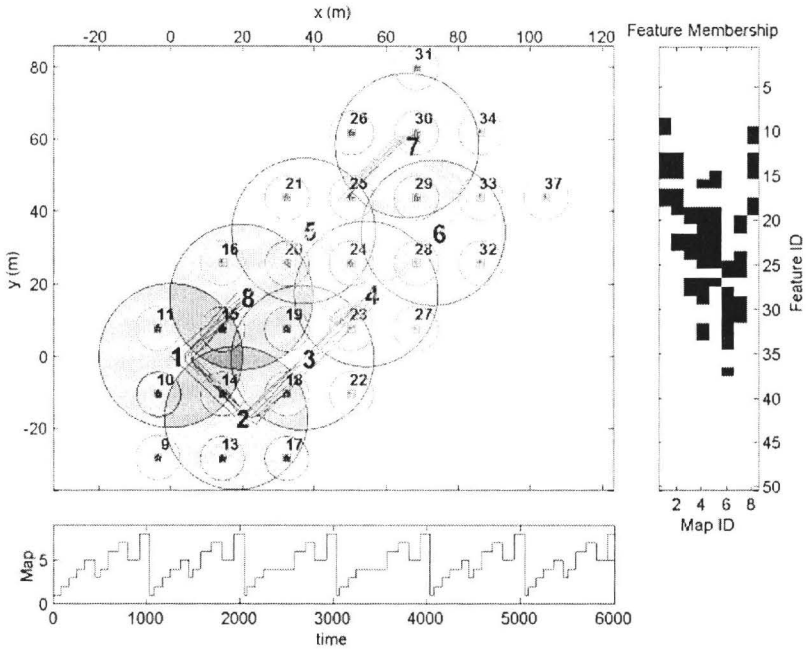

Figure 2: Vehicle path for a mission with eight maps. The eight shaded circles indicate the extent of the submaps in terms of vehicle location alone. Hence features can belong to maps even though they lie outside the submap boundaries. The map ID's are in bold. The right hand plot shows the distribution and sharing of features between maps. Some features are common to several submaps - for example feature 25 is found in submaps 3,4,5,6 and 7 whereas others are found in one alone — feature 37 is only mapped in submap 6. The lower plot shows the map transitions occurring as the vehicle moves. The basic trajectory is one of two overlapping rectangles aligned in North-East direction. The vehicle then returns to the origin (0,0) and repeats the pattern another six times. The feature estimate error ellipses are those resulting at the end of the simulations (in global coordinates) and are three-sigma bounds. The full covariance solution is also plotted on the central figure with its estimated feature locations are plotted with squares. The covariance bounds on features are at this resolution indistinguishable from those of the produced by the CTS algorithm.

mean values of $\tilde{\mathbf{y}}(k)$ — the difference for each time step k between the "true" relative vector y and the the estimate $\hat{\mathbf{y}}$. The upper two plots correspond to the vehicle to feature relative states. Note how the error does not converge to a zero value owing to the continual injection of process noise (odometry errors etc) into the vehicle as it moves. The lower two plots however correspond to the error in the relative location of the first two features in what ever the active map is. No process noise is added to the feature estimate covariances during the prediction stage of the estimation process. The result is the expected convergence to zero error. The second plot of Figure 6.2 shows the plots the characteristics of the N-run average of the NEES (Normalized Estimated Error Squared) of $\hat{\mathbf{y}}(k)$. The 95% confidence region bounds for estimation consistency are plotted on the same axes. the final plot of Figure 6.2 shows the normalized mean estimated error (NMEE) and the associated 95% confidence bounds of the hypothesis that the NMEE sequences are from a consistent estimator.

6.3 Off-line Adjustment

The root shifting operation which searches for a better representation (in terms of uncertainty) of map location and features need not be done as the vehicle transitions between maps. Table 6.3 shows the improvement in successively ap-

Table 1: Monte-Carlo simulation parameters.

Parameter	Value
Monte-Carlo Runs	200
Vehicle Process Noise (std)	0.01 ms^{-1}
Sensor Noise (std)	0.05 m
Cycle Length	360 m
Cycles per run	10
Total Distance Driven per run	3600 m
Loops per Cycle	2
Initial Vehicle uncertainty	0 m
Sensor Range	25 m
Sensor Field of View	100 deg
Feature Density	Every 18m in x and y
Map Radius	15 m
Vehicle Velocity	0.3 ms^{-1}

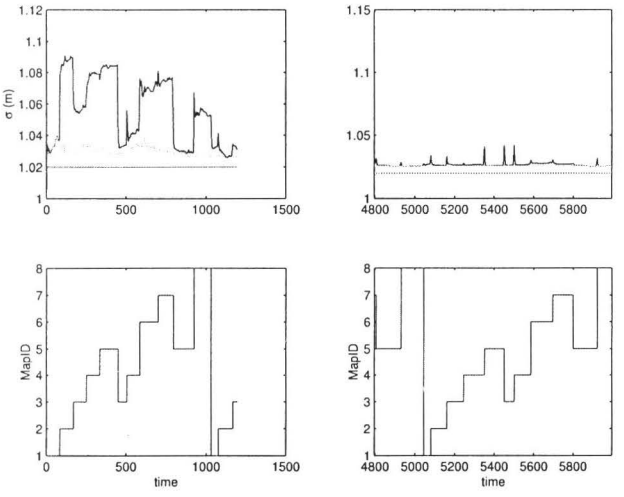

Figure 3: Convergence of vehicle state estimates for the eight map example. The left hand plots show the time steps at the opening stages of the mission while the right hand plots show the closing stages. The straight line is the lowest possible bound on vehicle uncertainty possible for either algorithm in the zero plant noise case. The full vehicle uncertainty given by the full covariance solution is the lighter of the two plots and is always less than or equal to the the CTS solution — the CTS solution is always consistent.

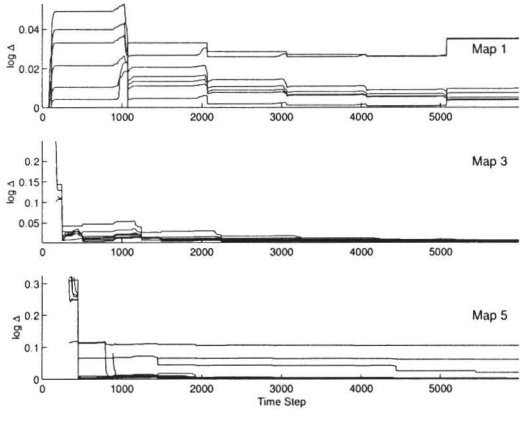

Figure 4: The log of the full-covariance / CTS difference between the determinants of the covariances in the first feature mapped in three randomly chosen submaps. Before the first submap transition occurs, submap 1 computes an answer that is identical to the full solution. Subsequently, submap one performs slightly worse due to the inclusion of less features. Note also that the convergence of submap 1 appears to be slower than the other maps and indeed at times the difference between full and CTS solutions seems to increase. This is because features in submap 1 can only be improved when this submap is active, whereas the full covariance solution updates all features with each new observation.

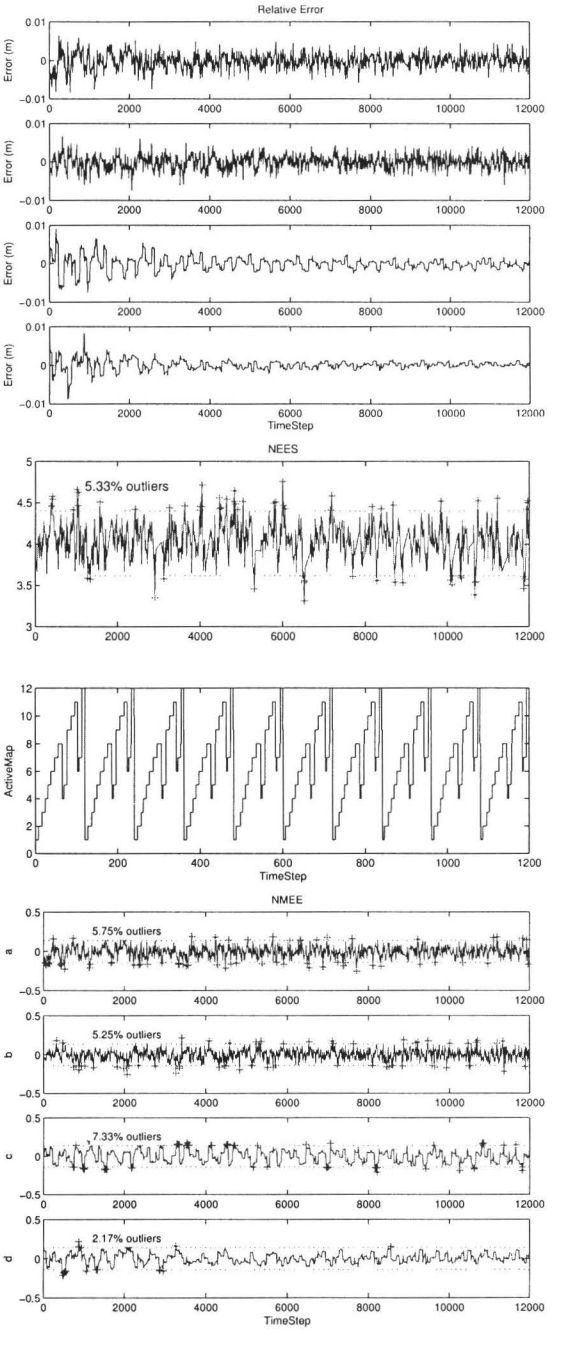

Figure 5: Monte-Carlo Consistency testing. The left hand plot shows the four components of the vector $\tilde{\mathbf{y}}$ which encodes the errors in relative positions of vehicle and the first two features (f_1^m, f_2^m) in the state vector of the active map m (the map in which the vehicle is currently moving). The first upper two sequences are the x and y components of the error in the relative position between vehicle and f_1^m. The second two are the x and y components of the error in the relative positions of f_1^m and f_2^m. The central plot is the NEES of $\tilde{\mathbf{y}}$. The bounds are the 95% confidence intervals on the NEES variable. The right hand plot shows the Normalized Mean Error (NMEE) for the components of $\tilde{\mathbf{y}}$. For consistency the mean estimated error should be zero. The limits plotted are the 95% confidence interval bounds for the consistency hypothesis.

Table 2: Percentage decrease in disparity between full covariance and CTS calculated feature uncertainties during sequential off-line adjustments using the CTS root shifting procedure. After three iterations all eighteen maps in this example showed no further improvement.

Map	iteration 1	iteration 2	iteration 3
1	0.0	0.0	0.0
2	13.6	13.6	13.6
3	10.0	16.8	16.8
4	10.6	18.9	18.9
5	6.3	13.0	17.9
6	7.1	12.6	12.6
7	20.7	25.4	26.6
8	1.8	7.1	8.5
9	6.6	8.3	13.5
10	3.5	5.0	9.5
11	1.6	6.3	7.6
12	15.0	17.8	19.0
13	3.7	5.4	10.1
14	4.1	5.5	9.6
15	4.5	6.0	10.2
16	7.5	9.0	13.4
17	14.4	22.1	22.1
18	12.9	12.9	12.9

plying the CTS algorithm to maps at the termination of the mission for a scenario with 18 maps. The simple approach adopted here begins with submap 2 and seeks to improve the map location estimate, it then progresses to submap 3 and so on until the highest index map has been adjusted. This then repeats until no maps show any further improvement. It is a topic of future research to determine, as a function of shared feature topography, the optimal adjustment sequence rather than the linear one used here. The adjustment was applied to a mission with a vehicle trajectory defined by six overlapping rectangles (similar to Figure 2 but the course was not repeated leading to a more uncertain collection of submaps at the end of the mission). The entries in the table are the percentage decrease in median disparity between the full covariance and CTS calculated feature uncertainties. As expected, the table shows no improvement for submap 1.

7 Conclusion

In this paper, we have presented a new technique for SLAM with multiple maps that achieves consistency of global state estimates with with O(1) time complexity. If the mobile robot is able to make repeated visits to all parts of the environment, by performing map improvement operations only when a map is exited, near-optimal convergence is demonstrated in constant-time.

In this approach, all SLAM filtering is performed in local submaps. Each submap has an associated reference frame, whose global position is estimated using information from other submaps. The transition of the vehicle between submaps is consistently performed using relocation [Neira *et al.*, 2002; Knight, 2002]. By never mixing information between maps, the method yields provably consistent global state estimates, while still achieving global convergence.

The method successfully exploits the fact that overlapping features are estimated in different maps. By changing the base reference of a map to be the feature within the map that has the "best" globally referenced position estimate, the global location estimates for all the features in a local region get improved via operations in other maps.

Ongoing research includes testing of the method with real data, the incorporation of probabilistic data association techniques, and extension of the approach to the nonlinear case.

References

[Bosse *et al.*, 2003] M. Bosse, P. Newman, J. Leonard, M. Soika, W. Feiten, and S. Teller. An atlas framework for scalable mapping. In *Proc. IEEE Int. Conf. Robotics and Automation*, 2003.

[Choset and Nagatani, 2001] H. Choset and K. Nagatani. Topological simultaneous localization and mapping (SLAM): toward exact localization without explicit localization. *IEEE Transactions on Robotic and Automation*, 17(2):125–137, April 2001.

[Dissanayake *et al.*, 2001] M. W. M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotic and Automation*, 17(3):229–241, June 2001.

[Guivant and Nebot, 2001] J. Guivant and E. Nebot. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotic and Automation*, 17(3):242–257, June 2001.

[Gutmann and Konolige, 1999] J-S. Gutmann and K. Konolige. Incremental mapping of large cyclic environments. In *International Symposium on Computational Intelligence in Robotics and Automation*, 1999.

[Hahnel *et al.*, 2002] D. Hahnel, D. Schulz, and W. Burgard. Map building with mobile robots in populated environments. In *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, 2002.

[Julier and Uhlmann, 2001] S. J. Julier and J. K. Uhlmann. Building a Million Beacon Map. In *Sensor Fusion*. SPIE, 2001.

[Knight, 2002] J. Knight. *Towards Fully Autonomous Visual Navigation*. PhD thesis, University of Oxford, 2002.

[Kuipers and Beeson, 2002] B. Kuipers and P. Beeson. Bootstrap learning for place recognition. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[Leonard and Feder, 2000] J. J. Leonard and H. J. S. Feder. A computationally efficient method for large-scale concurrent mapping and localization. In D Koditschek and J. Hollerbach, editors, *Robotics Research: The Ninth International Symposium*, pages 169–176, Snowbird, Utah, 2000. Springer Verlag.

[Montemerlo *et al.*, 2002] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[Neira and Tardós, 2001] J. Neira and J.D. Tardós. Data association in stochastic mapping using the joint compatibility test. *IEEE Trans. on Robotics and Automation*, 17(6):890–897, 2001.

[Neira *et al.*, 2002] J. Neira, J.D. Tardós, and J.A. Castellanos. Linear time vehicle relocation in SLAM. Technical Report RR-2002-08, Universidad de Zaragoza, Depto. de Informática e Ing. de Sistemas, Zaragoza, Spain, 2002.

[Smith *et al.*, 1987] R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *4th International Symposium on Robotics Research*. MIT Press, 1987.

[Tardós *et al.*, 2002] J.D. Tardós, J. Neira, P.M. Newman, and J.J. Leonard. Robust mapping and localization in indoor environments using sonar data. *Int. J. Robotics Research*, 21(4):311–330, April 2002.

[Thrun *et al.*, 2002] S. Thrun, D. Koller, Z. Ghahramani, H. Durrant-Whyte, and Ng. A.Y. Simultaneous mapping and localization with sparse extended information filters. In J.-D. Boissonnat, J. Burdick, K. Goldberg, and S. Hutchinson, editors, *Proceedings of the Fifth International Workshop on Algorithmic Foundations of Robotics*, Nice, France, 2002. Forthcoming.

[Thrun, 2001] S. Thrun. A probabilistic online mapping algorithm for teams of mobile robots. *Int. J. Robotics Research*, 20(5):335–363, May 2001.

[Williams *et al.*, 2002] S. B. Williams, G. Dissanayake, and H. Durrant-Whyte. An efficient approach to the simultaneous localisation and mapping problem. In *Proc. IEEE Int. Conf. Robotics and Automation*, pages 406–411, May 2002.