
Michael Bosse

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology (MIT)
Cambridge, MA, USA
ifni@mit.edu

Paul Newman

Department of Engineering Science
University of Oxford
Oxford, UK

John Leonard

Seth Teller

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology (MIT)
Cambridge, MA, USA

Simultaneous Localization and Map Building in Large-Scale Cyclic Environments Using the Atlas Framework

Abstract

In this paper we describe Atlas, a hybrid metrical/topological approach to simultaneous localization and mapping (SLAM) that achieves efficient mapping of large-scale environments. The representation is a graph of coordinate frames, with each vertex in the graph representing a local frame and each edge representing the transformation between adjacent frames. In each frame, we build a map that captures the local environment and the current robot pose along with the uncertainties of each. Each map's uncertainties are modeled with respect to its own frame. Probabilities of entities with respect to arbitrary frames are generated by following a path formed by the edges between adjacent frames, computed using either the Dijkstra shortest path algorithm or breath-first search. Loop closing is achieved via an efficient map-matching algorithm coupled with a cycle verification step. We demonstrate the performance of the technique for post-processing large data sets, including an indoor structured environment (2.2 km path length) with multiple nested loops using laser or ultrasonic ranging sensors.

KEY WORDS—mobile robots, SLAM, scaling, navigation and mapping, ATLAS, cyclic environments

1. Introduction

In this paper we describe Atlas, a framework in which existing small-scale mapping algorithms are used to achieve real-time performance in large-scale, cyclic environments. The ap-

proach does not maintain a single, global coordinate frame, but rather an interconnected set of local coordinate frames. The representation consists of a graph of multiple local maps of limited size. Each vertex in the graph represents a local coordinate frame and each edge represents the transformation between adjacent local coordinate frames. In each local coordinate frame, we build a map that captures the local environment and the current robot pose along with their uncertainties. Together, the coordinate frame and the map are referred to as a “map-frame”. The spatial extent of each map is not predefined. Instead a limit is placed on the complexity of each map. An intrinsic performance metric of the local simultaneous localization and mapping (SLAM) processing is used to determine whether to transition to an adjacent map-frame or to generate a new one. The map's complexity is typically bounded by placing a hard limit on the maximum number of features that can be inserted into the map, and the performance metric is typically based on the number of observed features and the local uncertainty of the robot.

Atlas is intended to be a generic framework in which a variety of techniques could be used as the local mapping module. The approach assumes that a suitable local SLAM algorithm is available that can produce consistent maps in small-scale regions with a fixed amount of computation for each new sensor observation. Efficient global performance is not possible if the local SLAM method incurs an ever-growing computational burden. For example, when local SLAM is based on scan-matching, then only a finite set of scans can be retained in local regions. If local processing were based on the matching of a new sensor scan with all of the scans ever obtained in

a local region, then local map complexity would grow linearly with time.

Each map's uncertainties are modeled with respect to its own local coordinate frame. The uncertainty of the edges (adjacency transformations) in the Atlas graph are represented by a Gaussian random variable and are derived from the output of the SLAM algorithm running in a local region. A limit is placed on the per-map computation by defining a measure of complexity for each map-frame, which is not allowed to exceed a threshold (the map capacity). Rather than operating on a single map of ever-increasing complexity, the Atlas framework simply switches its focus to a new or adjacent map-frame.

New links are added to the Atlas graph via an efficient map-matching algorithm. Potential map-matches are entertained only for map-frames that fall within an uncertainty bound of the current map. Coordinate transformations and associated error estimates are generated for the entities in one map-frame with respect to another arbitrary map-frame by following a path formed by the edges between adjacent map-frames. These paths are computed using either (i) the Dijkstra shortest path algorithm (Dijkstra 1959) or (ii) the breadth-first search (BFS). When the Dijkstra shortest path algorithm is used, the uncertainties of the transformations of the edges of the graph serve as a distance metric. Alternatively, using the BFS, the number of intervening map-frames is used as the distance metric. All other components of Atlas have a bounded computational cost and hence the choice of Dijkstra versus BFS determines the overall order of growth of computational complexity of the method.

Assuming that each vertex of the Atlas graph has a bounded degree (the maximum number of adjacent map-frames is bounded), then the computational cost of the Dijkstra computation is $\mathcal{O}(n \log n)$ (where n is the number of map-frames), and when using BFS the computational cost is $\mathcal{O}(n)$. Atlas achieves efficient real-time performance by amortizing this uncertainty projection computation over time. For the environments considered in this paper, the computational effort expended for uncertainty projection and map-match candidate selection is several orders of magnitude less than the computational effort expended for local SLAM processing and map-matching. Hence, we anticipate that the method can effectively handle environments several orders of magnitude larger than those considered in our experiments below.

Loop closing is one of the most difficult issues in SLAM research. Two different types of error can occur in loop closing: false positive matches and missed detected matches. The former refers to situations in which the robot erroneously asserts that a loop has been closed with a false match. The latter case occurs when a loop closure has been missed due to failure to successfully match the current map with a previously mapped area. Atlas adopts a conservative loop closing and verification strategy which attempts to avoid false positive matches at the expense of missing some genuine loop closure events. It is

possible, however, to present an adversely designed environment with highly repetitive structure and a path in which the accumulated uncertainty is so large that our technique, as well as any known SLAM loop closing algorithm, would fail.

The difficulty of a particular environment to loop closures is dependent on the growth of uncertainty in local mapping tasks and on the richness of the local map representation. Richer maps are less likely to be ambiguous and a smaller growth of uncertainty allows for longer paths to be followed before confusing two nearby ambiguous regions.

The main challenge in SLAM is handling uncertainty: local uncertainty due to noise introduced in each measurement, and global uncertainty due to the need to discern whether the region currently observed by the robot is newly explored territory or has been previously visited. Both local and global uncertainty are exacerbated by repeated low-level structure (for example, periodic arrays of doors or windows) or high-level structure (for example, nearby corridors that are nearly indistinguishable).

Uncertainty is a challenge in two respects: it is antagonistic both to achieving correctness and efficiency. When the mapping algorithm mistakes one local feature for another, or one region for another, it performs an incorrect data association and produces an incorrect map. On the other hand, if the algorithm expends excessive effort in an attempt to avoid such misassociations, it sacrifices efficiency. For algorithm designers, then, a key challenge is to construct a correct map with reasonably high probability, while bounding the amount of computation expended.

The use of submaps allows Atlas to handle ambiguity at small scales (feature sizes). For Atlas to handle ambiguity at large scales, the environment must satisfy the condition that this scale must be larger than the error ellipsoid accrued by the robot around any loop traversed prior to encountering the ambiguous region.

After recognizing the closure of an extended loop, we do not constrain the composition of adjacency transformations to be the identity transformation. This is essential to achieving efficient real-time performance, since no global updates are required during the robot's motion. The identity constraint can be applied off-line, however, to refine the global arrangement of the multiple coordinate frames (see Section 4).

2. Related Research

Before describing the components of Atlas in more detail, we first provide a brief review of related research. Probabilistic techniques have proven vital in attacking the large-scale SLAM problem. A variety of approaches have been proposed for representing the uncertainty inherent to sensor data and robot motion, including topological (Kuipers 2000), particle filter (Thrun 2001; Montemerlo et al. 2002), and feature-based (Smith, Self, and Cheeseman 1990) models. Several highly successful SLAM approaches have been developed based on

the combination of laser scan-matching with Bayesian state estimation (Gutmann and Konolige 1999; Thrun 2001). These methods, however, encounter computational difficulties when closing large loops.

The Kalman filter provides the optimal linear recursive solution to SLAM when certain assumptions hold, such as perfect data association, linear motion and measurement models, and Gaussian error models (Smith, Self, and Cheeseman 1990). The convergence and scaling properties of the Kalman filter solution to the linear Gaussian SLAM problem are now well known (Dissanayake et al. 2001). Considerable recent research effort has been extended toward mitigation of the $\mathcal{O}(n^2)$ complexity (where n is the number of features) of the Kalman filter SLAM solution. Efficient strategies for SLAM with feature-based representations and Gaussian representation of error include postponement (Davison 1998), decoupled stochastic mapping (DSM; Leonard and Feder 2001), the compressed filter (Guivant and Nebot 2001), sequential map joining (Tardós et al. 2002), the constrained local submap filter (Williams, Dissanayake, and Durrant-Whyte 2002), and sparse extended information filters (SEIFs; Thrun et al. 2002, 2003). Each of these methods employs a single, globally referenced coordinate frame for state estimation. The Kalman filter can fail badly, however, in situations in which large angular errors and significant data association ambiguities invalidate the Gaussian error assumptions.

In outdoor environments, several SLAM algorithms have been implemented for very large-scale data sets. For example, Guivant and Nebot (2001) have published results for an experiment in Victoria Park, Sydney, using the compressed filter, with an implementation that uses trees as “point” features. (In Section 5.3, we present processing results for this data set using scan-matching.) More recently, Wang, Thorpe, and Thrun (2003) and Hähnel, Schulz, and Burgard (2002) have achieved full three-dimensional (3D) mapping of urban areas with dynamic objects. Thrun et al. (2003) have demonstrated large-scale SLAM in a cyclic underground mine. (In Section 5.4, we also present processing results for this data set.)

Some of the most successful experiments for autonomous loop closure in indoor environments have been performed by Gutmann and Konolige (1999). One notable feature of their work, which we adopt in the current paper, is the use of a hybrid metrical/topological map representation (Gutmann and Konolige 1999):

“A map is represented as an undirected graph: nodes are robot poses with associated scans and links are constraints between poses obtained from dead-reckoning, scan-matching, or correlation.”

However, our approach differs from that of Gutmann and Konolige in several critical respects. Our method operates successfully with either feature-based SLAM (Smith, Self,

and Cheeseman 1990) or laser scan-matching (Lu and Milios 1997) as the local mapping module. The approach of Gutmann and Konolige (1999) relies on the accuracy of dense laser scanner data and would encounter difficulties with sonar data. To detect loop closure events, our approach performs map-matching with constant-size submaps; Gutmann and Konolige (1999) use a local patch which grows linearly with the position uncertainty. In our approach, we reject false positive matches due to repetitive environmental structure with the loop cycle verification step described in Section 3.3.2. Finally, when loops are closed, our approach simply adds a new connection to the Atlas graph, whereas the method of Gutmann and Konolige (1999) requires the recomputation of all of the scan poses that comprise a loop when cycles are detected.

One of the appealing aspects of a hybrid metrical/topological approach to mapping and localization (Chong and Kleeman 1997a; Kuipers 2000; Bailey and Nebot 2001; Choset and Nagatani 2001) is that uncertain state estimates do not need to be referenced to a single global reference frame. The strategy of partitioning a large map into multiple smaller maps is intuitively appealing for both computational efficiency and robustness. Chong and Kleeman (1997b) write “... the local mapping strategy is devised ... to improve efficiency and to curb the accumulated ‘inevitable errors’ from propagating to other local maps continuously.” This is the strategy advocated in this paper. Our approach, however, differs significantly from that of Chong and Kleeman (1997a) in the manner in which the graph of local maps is used. We separate the processes of map-matching (adding new links to the graph) from traversal (determining the map-frame that best describes the current sensor measurements). We cast the problem of projecting the uncertainty of one map-frame relative to another as an amortized shortest path problem, providing a computationally efficient method to close loops in cyclic environments. Using our cycle verification step, our method can tolerate an increased level of ambiguity in environments with repetitive structure.

The hybrid metrical/topological approach allows us to restrict the representation of errors via Gaussian distributions to local regions where linearization works well, rather than representing the entire environment with one Gaussian distribution. The large-scale linearization inherent in methods that use a single global coordinate system for error representation, such as SEIFs or DSM, will fail when closing large loops with unbounded linearization errors. Since Atlas does not enforce cycle consistency constraints, it can handle the non-linearities inherent to uncertain coordinate transformations more gracefully than alternatives that employ a single global reference frame.

An alternative to the use of local linearization would be to adopt a fully non-linear formulation of the SLAM problem, such as FastSLAM (Montemerlo et al. 2002) or SLAM us-

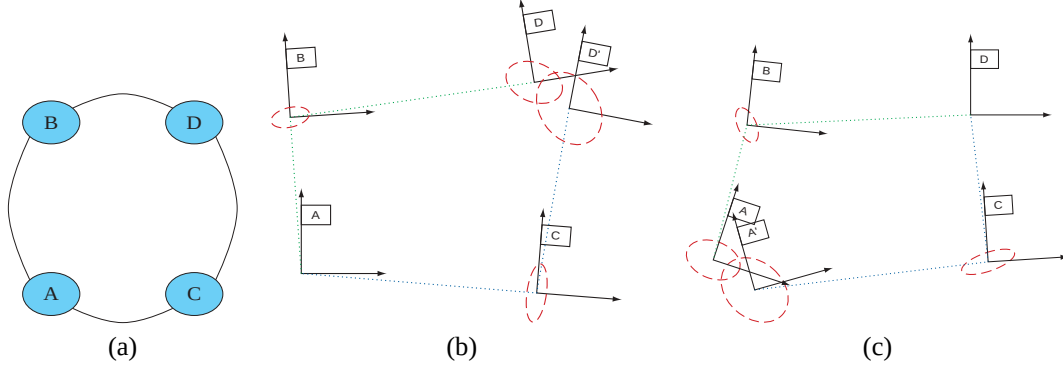


Fig. 1. The Dijkstra projection using two different source nodes. (a) depicts the topological arrangement of the Atlas graph. (b) uses map-frame A as the source of the projection. (c) uses map-frame D as the source. The ellipses on the coordinate frames represent the accumulated projection error. The shortest path from map-frame A to D is clearly via map-frame B.

ing a sum of Gaussians model (Durrant-Whyte et al. 2001). The computational requirements of these methods, however, remain poorly understood in large cyclic environments. In future research, it may be possible to implement one of these techniques as the local mapping strategy within the Atlas framework.

3. Atlas Components

We now provide a detailed description of the core concepts of the Atlas framework: uncertainty projection, competing hypotheses, creation of new map-frames (Genesis), closing loops (map-matching), and instantiating and evaluating new hypotheses in adjacent map-frames (Traversal).

3.1. Uncertainty Projection

Atlas edges contain the information necessary to relate two map-frames. The uncertainty of the transformation edge is used to project a stochastic entity (such as the robot position) from one map-frame into another. However, if the map-frames are not adjacent, these transformations (and their uncertainties) must be composed along a path of edges that link the Atlas vertices. Due to cycles in the graph, there may be more than one path from one vertex to another. Since these cycles are not constrained on-line, distinct paths will not in general produce the same composite transformation. In Figure 1(a), frame D is reachable from A via B or C, resulting in the two possible projections of frame D relative to A, shown in Figure 1(b).

To resolve this ambiguity we use either the Dijkstra shortest path algorithm (Dijkstra 1959) or a BFS to find a unique path between the nodes. For the Dijkstra projection, we use a statistical metric, ρ , based on the uncertainty of the transformation in Atlas edges. The metric we choose is the determinant of the covariance matrix of the composite transformation:

$$T_a^c = T_a^b \oplus T_b^c \quad (1)$$

$$\Sigma_{ac} = J_1(T_a^b, T_b^c) \Sigma_{ab} J_1(T_a^b, T_b^c)^T + J_2(T_a^b, T_b^c) \Sigma_{bc} J_2(T_a^b, T_b^c)^T \quad (2)$$

$$\rho = \det(\Sigma_{ac}). \quad (3)$$

Appendix B reviews the notation that we employ for representing uncertain coordinate frame transformations (Smith, Self, and Cheeseman 1990; Castellanos and Tardós 2000). The determinant of the covariance is a measure of the volume of the n -sigma hyperellipsoid of probability mass for a Gaussian distribution (Feder, Leonard, and Smith 1999).

The determinant is a good metric to use since it is also invariant to deterministic coordinate transformations.

$$T_w^c = T_w^a \oplus T_a^c \quad (4)$$

$$\Sigma_{wc} = J_2(T_w^a, T_a^c) \Sigma_{ac} J_2(T_w^a, T_a^c)^T \quad (5)$$

$$\det(\Sigma_{wc}) = \det(\Sigma_{ac}). \quad (6)$$

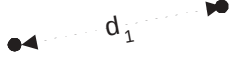
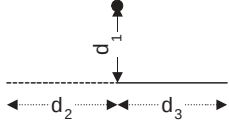
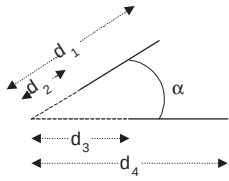
We define the Dijkstra projection of an Atlas graph with respect to a given source vertex as the global arrangement of frames using compositions along Dijkstra shortest paths. This projection has the property of transforming the Atlas graph into a tree of transformations with the source map-frame as the root. We measure the nearness of any map-frame to the source frame as ρ computed from the compositions of transformations up the tree to the root.

3.2. Genesis

Since the complexity of each map is bounded, it is necessary to create new local map-frames when entering unexplored regions for which no existing map-frame can explain the sensor measurements. The Genesis process adds a new vertex and edge to the Atlas graph. Mathematically, the generation of a new map-frame $\mathcal{M}_j$ and robot pose $\mathbf{x}_j$ via genesis is a function of an old map-frame $\mathcal{M}_i$ and robot pose $\mathbf{x}_i$:

$$(\mathcal{M}_j, \mathbf{x}_j) = g(\mathcal{M}_i, \mathbf{x}_i). \quad (7)$$

Table 1. Defining Map Signature Elements Between Pairings of Line and Point Features

Pairing	Geometry	Element
Point–point		$[d_1]$
Point–line		$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \end{bmatrix}$
Line–line		$\begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ d_4 \\ \alpha \end{bmatrix}$

The process of genesis encapsulated by the function g is broken down as follows.

1. The current robot pose defines the origin of a new frame. Thus, the transformation from the old to the new frame is simply the robot's position in the old frame at the time the new frame is created.
2. The robot pose is initialized to zero in the new frame.
3. The uncertainty of the transformation is set to the uncertainty of robot pose in the old frame.
4. The uncertainty of the robot pose in the new frame is zero by definition. All of the uncertainty of the robot pose at the time of genesis is captured by the uncertain transformation.

3.3. Map-matching

Genesis creates new maps to explain unexplored areas. In cyclic environments the robot will eventually revisit an area that it has already explored. We would like to automatically discover such loop closure events and update the connectivity in the Atlas graph. There are essentially three steps to loop closing.

First, the robot has to know which of its previously explored maps it could possibly be returning to given the integrated uncertainty about the loop. This is achieved using the map projections as described in Section 3.1. While computing the map projections from the current map-frame, we maintain a list of potential map-frames that may possibly overlap the current one.

Secondly, we need to check if any of the map-frames in the potential list match the current frame. This is achieved with a map-matching module that compares the structure of two

maps and returns the probability that the two maps match as well as the coordinate transformation that best aligns the two maps.

Finally, when the map-matching module succeeds in finding a match, (and after verifying the consistency of the alignment; see Section 3.3.2), we update the Atlas graph by simply adding an edge to the graph. The edge contains the alignment transformation and uncertainty returned from the map-matching module.

We can describe the map-matching process as a search for a coordinate transformation that aligns two overlapping map-frames. The uncertainty from the prior estimate for the map-frame alignment transformation (as computed by the map projection) may be very large; too large, in fact, to be able to rely on the simple strategy of nearest-neighbor feature gating for data association. Thus, the map-matching module needs to be a method that is robust to large initial errors in the transformation.

In general terms, map-matching is comprised of two steps.

1. Determining the probability m_{ij} that the two maps $\mathcal{M}_i$ and $\mathcal{M}_j$ match by identifying common structure:

$$m_{ij} = P(\mathcal{M}_i \cap \mathcal{M}_j). \quad (8)$$

2. Producing the alignment transformation T_i^j and its uncertainty Σ_{ij} between maps $\mathcal{M}_i$ and $\mathcal{M}_j$:

$$\{T_i^j, \Sigma_{ij}\} = P(T_i^j | \mathcal{M}_i \cap \mathcal{M}_j). \quad (9)$$

The exact form of $\mathcal{M}_i \cap \mathcal{M}_j$ used for determining common structure is not dictated by the Atlas framework, but left as a module to be defined in a particular implementation. For the feature-based SLAM results in this paper, we define the operation $\mathcal{M}_i \cap \mathcal{M}_j$ as the search for correspondence between features in $\mathcal{M}_i$ and $\mathcal{M}_j$. If a small number of features match for two maps, then the probability of a successful match is low. For the scan-matching results in this paper, iterative closest point (ICP) matching is performed between the two maps. In our implementation, each local scan-matching map consists of sets of laser scan points and uncertain vehicle poses (Appendix C), rather than using an occupancy grid as employed by Gutmann and Konolige (1999).

Repetitive structure in the environment may cause false positive matches to be discovered in map-matching. The degree of repetition can be somewhat assessed by map-matching a map with itself. Only map structure elements that match uniquely within a map should be used to evaluate a match to another map.

3.3.1. Feature-Based Map Matching

We now describe a two stage implementation of the map-matching process for our feature-based example summarized as follows.

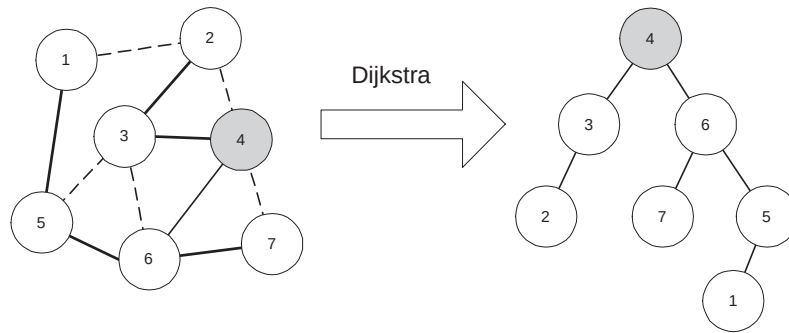


Fig. 2. The Dijkstra projection from a given node in a graph transforms the graph into a tree with the source node as a root. Here we take node 4 as a source. Solid lines correspond to links that are used in the tree representation. Note how in this example the uncertainty between links 4 and 7 is larger than that accrued via traversing links 4–6 and then 6–7. Hence there is no direct link between 4 and 7 in the tree.

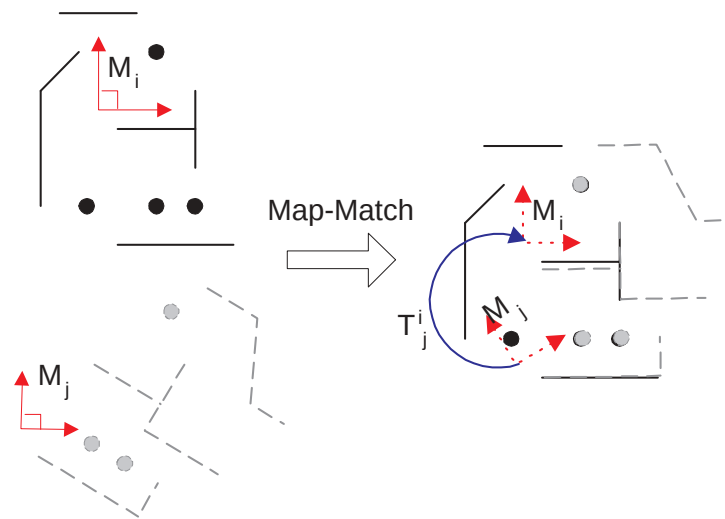


Fig. 3. Map-matching as a search for a transformation between maps that maximally aligns common mapped features. Here two maps i and j share features and a good map-match can be found between them. Note how only a subset of features are matched.

1. A signature for both maps is constructed which is an ordered list of elements describing properties of the map that are invariant to translation and rotation of its coordinate frame. A comparison operator is defined over two signatures which yields a set of correspondences between elements in each list.
2. Each map is matched with itself to identify repetitive structure. Any elements that correspond to other elements in the same map are removed from the map's signature. This dramatically reduces the likelihood of false map-matches due to repetitive structure by focusing on the unique elements of each local environment.
3. The signatures of both maps are now compared. Each element to element correspondence defines a potential alignment transformation from $\mathcal{M}_i$ to $\mathcal{M}_j$.
4. Each potential alignment transformation is applied to $\mathcal{M}_i$. The validity of each transformation is evaluated by counting the number, η , of feature pairs it brings into alignment with nearest-neighbor gating.
5. The correspondences from the best (largest η) potential transformation with $\eta > \eta_m$ are used to refine the transformation and its covariance. Each correspondence defines a constraint on the transformation. The combined set of η constraints are solved in weighted least-squares sense using the covariances of the feature estimates within each map to form the weights. This process also yields the covariance of the transformation.

In this implementation, our maps consist of 2D point and line features. The elements used in creating a map signature are pairings of non-parallel lines, point-line pairs and point-point pairs drawn from the map. For each pairing, the signature element consists of distances and/or angles that are independent of the map-frame's orientation and location. Table 1 defines the invariants used for the three species of pairings.

The number of signature elements to compare when matching maps is of $O(n^2)$, which may lead to $O(n^4)$ matches that need to be performed. However, we can reduce the number of matches that need to be tested to $O(n^2)$ by sorting the signature elements into a canonical order, which then reduces the total computational burden to $O(n^2 \log n)$.

The approach we have adopted for map-matching is not unique. For example, the joint compatibility test with branch and bound technique suggested by Neira and Tardós (2001) could also be utilized. The freedom to choose a map-matching strategy highlights the modularity of the Atlas framework.

3.3.2. Cycle Verification

One of the biggest difficulties to correctly closing large loops is the danger of false matches due to ambiguous structure in the environment. When the environment contains repetitive

structure, map-matches are not unique. Furthermore, the prior uncertainty of the mapping robot before closing the loop may be too large to disambiguate the correct match. Even when there is only one match visible, it could be the case that the correct match just has not been seen yet, and thus we cannot be certain of the single match.

There are several possible approaches to tackling the decision of when to accept a map-match. For example, one could employ multiple hypotheses to track each possible decision branch, as in Austin and Jensfelt (2000). Alternatively, one could use a method capable of representing multimodal probability distributions, such as Monte Carlo localization (Dellaert et al. 1999; Thrun et al. 2000) or sum of Gaussian models (Durrant-Whyte et al. 2001). Another strategy is to temporarily take the maximum likelihood decision, and then detect and fix errors later by rolling back the computation. This technique is used in Thrun et al. (2003), where falsely matched links can be recursively corrected when large errors are detected.

Each strategy has its advantages and disadvantages. Methods that perform multiple hypothesis tracking or that employ multimodal probability distributions are exponentially complex, and aggressive methods for reducing the number of active hypotheses (or modes) are required. Rollback methods may not be able to accurately detect the errors and there is not real bound on how far they may have to rollback the computation to fix the errors. (The computation is at least proportional to the time to detect an error.) Also, most data structures are not suited to efficient rollbacks (such as the standard Kalman filter).

The approach that we adopt in this paper is to defer the decision until more information is available for a verification. One potential disadvantage with deferring decisions is that one may never have enough information for a validation. The mapping algorithm will fail to close loops in some situations, for example, if a path is crossed orthogonally with little overlap in sensor measurements from different passes.

The basic idea for map-match verification is to defer the validation of map-match edges in the Atlas graph until a "small" cycle is formed that is geometrically consistent. When closing a large loop, the prior uncertainty for the matches may be so large that multiple ambiguous matches are possible (Figure 4). By waiting for at least one more distinct map-match, the consistency of cycles formed with the first map-match can be verified. These cycles are much smaller than the large loop's cycle, and thus we can bound the error around it. If the cycle's prior uncertainty is smaller than the expected distance between ambiguous matches and the transformations around the cycle are consistent, we can be confident that we have the single correct map correspondences. Both unvalidated map-match edges are then validated and their effects are propagated in subsequent map projections.

Short cycles containing the current Atlas node are quickly discovered in constant-time complexity by employing a truncated BFS. A cycle is found when the BFS leads to a node that

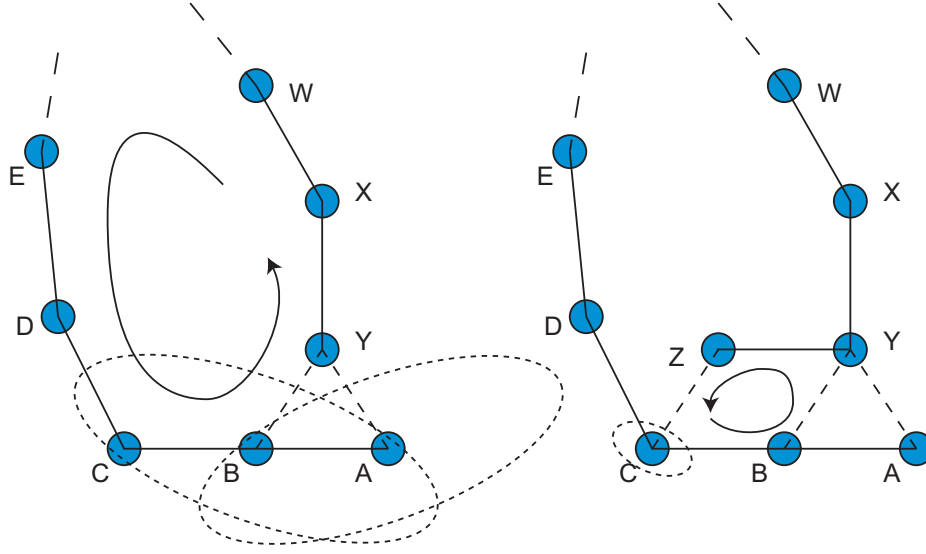


Fig. 4. On the left is the state of the Atlas graph before closing the large loop $ABCDE \dots WXY$. There are two potential map-matches YB and YA ; however, the prior uncertainty of the open-loop transformations is too large to disambiguate the map-matches. On the right is the state of the Atlas graph after mapping node Z and making the match ZC . A small cycle $CBYZC$ is now present with an uncertainty less than the distance between the ambiguities and both edges ZC and YB are validated. The transformations about cycle $YBAY$ are inconsistent w , and hence edge YA is unvalidated.

has been previously visited. The sequence of nodes that make up the cycle is determined by the two distinct paths up the BFS tree from the previously visited node. The BFS is truncated after a maximum depth to ensure constant-time complexity.

The cycle verification step effectively increases the map coverage used to form the match. False positives are then only a problem when the scale of any environment ambiguities is larger than the coverage size of the maps in the verification cycle. Thus, the likelihood of false matches is greatly reduced for an almost insignificant extra computational cost.

3.4. Traversal with Competing Hypotheses

Since each map-frame only covers a localized portion of the environment, the mapping robot will not be able to match sensor measurements with a map-frame once it leaves the volume covered by the map-frame. The robot will either need to use an adjacent map-frame to explain its observations, or it will have to generate a new map if it cannot find a valid map-frame. To determine if adjacent map-frames can better explain the current sensor measurements, we spawn a hypothesis using the transformation in the Atlas edge to instantiate the robot in an adjacent map-frame. Initially new hypotheses (which we call juvenile hypotheses) are prevented from mapping new structure to explain sensor measurements. Juvenile hypotheses are just used to evaluate the localization performance using existing structure.

Thus, at any given time, there are several competing map-frame hypotheses that attempt to explain the current robot pose and sensor observations. The map-frames that best explain the current sensor measurements will have hypotheses that can match most of the measurements to the map structure. On the other hand, invalid hypotheses will fail to match sensor measurements to existing structure and will be consequently pruned. We verify the validity of each map-frame's hypothesis by monitoring a performance metric q for a few time-steps. The performance metric is simply the likelihood that the current sensor measurements Z explain the map $\mathcal{M}_i$ and current robot pose $\mathbf{x}_i$:

$$q_i = P(\mathcal{M}_i, \mathbf{x}_i | Z). \quad (10)$$

The Atlas framework continually maintains a set of several hypotheses to continually determine the best map-frames to activate. Each map-frame can support only one hypothesis at a time, and the maximum number of total hypotheses, $\mathcal{H}_m$, is fixed so that the computational requirements remains bounded. If the number of potential hypotheses is greater than $\mathcal{H}_m$, then they are instantiated only when existing hypotheses are terminated. In practice, this will only occur in highly interconnected regions of the Atlas graph, and the limit has not been shown to have a significant effect on the performance since invalid hypotheses are quickly pruned.

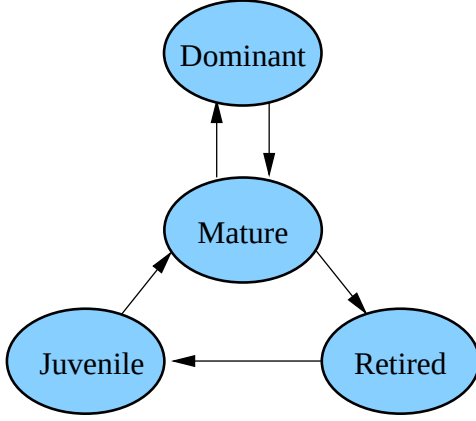


Fig. 5. Atlas hypothesis state transition diagram. The dominant hypothesis is used to provide an output from the algorithm. It is simply the most successful of all the mature hypotheses. Mature hypotheses are able to extend maps and spawn juvenile hypotheses in adjacent map-frames. When a mature hypothesis fails to describe the robot’s environment adequately (the robot has moved away, for example), it is retired. It can be reinstated as a juvenile at some time in the future by an adjacent mature hypothesis. Juveniles are not allowed to modify their map. If, after a probationary period, a juvenile’s map is failing to explain sensor data, then it is deleted. However, a successful juvenile is promoted to mature status.

3.4.1. Traversal

Four types of map-frame hypotheses are used to manage the traversal of focus about the Atlas graph (making transitions between adjacent map-frames). We label the different types of hypothesis as juvenile, mature, dominant, and retired. Retired hypotheses involve no computation and are simply used to mark inactive maps. The two other types of hypotheses, however, process sensor measurements and are evaluated with the same performance metric. Mature hypotheses can extend their maps (provided that they have not reached their map capacity) and spawn juvenile hypotheses into their adjacent map-frames. Juvenile hypotheses are used to test the feasibility of transitioning to neighboring maps, based on how well these maps explain the sensor measurements. They are restricted from extending their maps. See Figure 5 for a state transition diagram.

A juvenile hypothesis can “mature” when after a short probationary period its performance metric q becomes greater than any other mature hypothesis. If at the end of this probationary period the quality of a juvenile hypothesis does not warrant promotion, it is simply deleted. (In our experiments, the probationary period is a duration of 2 s.)

The mature hypothesis with the best performance metric is considered the dominant hypothesis. The dominant hypothesis is used for publishing current robot pose and local features to clients of the Atlas framework. In other words, it is the output of the framework.

Mature hypotheses that fail to perform well are saved and “retired”. A retired hypothesis may be reactivated at a later time as a juvenile.

If we have only one mature but failing hypothesis, then a new one must be created to explain current sensor data. This is done by genesis (Section 3.2). This situation will occur when none of the existing hypotheses can adequately explain the sensor measurements, such as when the robot moves into an unexplored region.

3.4.2. Robot Relocalization

When creating a juvenile hypothesis in a retired map-frame $\mathcal{M}_i$ we reinitialize its robot pose $\mathbf{x}_i$ using the robot pose $\mathbf{x}_j$ from an adjacent map-frame $\mathcal{M}_j$ (see Figure 6). First, we seed the hypothesis with a robot pose $\mathbf{x}_j^*$ projected into frame i :

$$\mathbf{x}_i^* = T_i^j \oplus \mathbf{x}_j \quad (11)$$

$$\Sigma_{\mathbf{x}_i}^* = J_1(T_i^j, \mathbf{x}_j) \Sigma_{ij} J_1(T_i^j, \mathbf{x}_j)^T + J_2(T_i^j, \mathbf{x}_j) \Sigma_{x_j} J_2(T_i^j, \mathbf{x}_j)^T \quad (12)$$

where $J_1(\cdot, \cdot)$ and $J_2(\cdot, \cdot)$ are the Jacobians of the transformation composition operators (Tardós et al. 2002).

The hypothesis now enters a bootstrapping phase, in which a consistent initialization of the vehicle into the juvenile hypothesis is sought. Sensor measurements, interpreted with the seeded robot pose, $\mathbf{x}_i^*$, are accumulated. This continues until we have collected enough measurements, Z , to solve explicitly for the robot pose independently of $\mathbf{x}_i^*$; we call this function w . This approach conserves the statistical independence of map-frames:

$$(\mathcal{M}_i, \mathbf{x}_i^{new}) = w(\mathcal{M}_i, Z). \quad (13)$$

If an explicit solution to w cannot be computed because of lack of explained sensor measurements, then the hypothesis is invalid and terminated. Otherwise we have a tenable juvenile hypothesis.

3.5. Summary

Figure 8 summarizes the processing performed during each cycle of the algorithm. Step 1, local map iteration, runs in constant time, because the complexity of each local map is bounded, and the number of non-retired hypothesis is bounded. Step 2, the management of hypothesis state transitions, is bounded because we assume that each local map is connected to a bounded number of other local maps (bounded degree assumption). For step 3, the computational cost to run

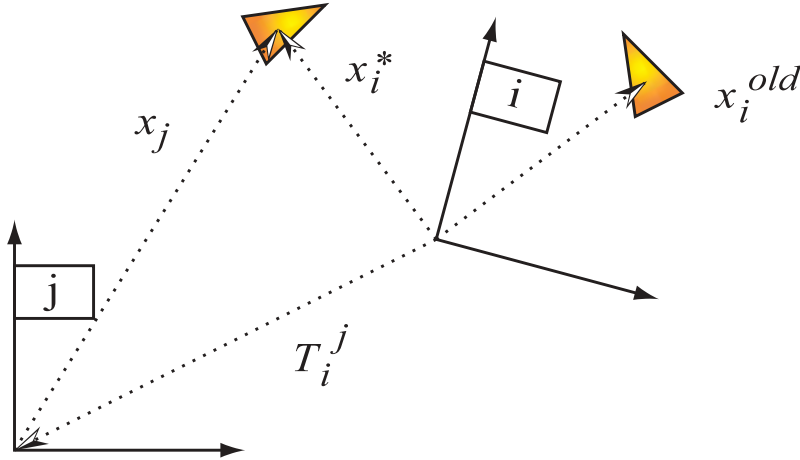


Fig. 6. Seeding the robot position for juvenile hypothesis in frame i using the current pose in the adjacent map-frame j . The retire hypothesis attached to frame i has the robot location at x_i^{old} . The hypothesis is rejuvenated to have the robot pose of x_i^* .

map projection to completion is $\mathcal{O}(n \log n)$ when using the Dijkstra shortest path algorithm and $\mathcal{O}(n)$ when using BFS (where n is the number of map-frames). This computation is amortized over time. Finally, step 4, map-matching, is constant time because the size of the maps is bounded, and the number of maps being matched is bounded.

The purpose of the map uncertainty projection computation is to find potential candidates for map-matching. By amortizing this computation, the effect is a delay in the determination of map-match candidates. During each iteration of the Atlas algorithm, if the dominant map ID changes in step 2, then the computation is restarted; otherwise, a fixed amount of effort is expended to step the computation forward towards completion (Extensions 1 and 2 provide clear illustration of this process).

The failure mode of this approach would be that if the dominant map changes quickly, before the uncertainty projection computation runs to completion, then some map-match candidates would be missed. If we assume, for example, that the map uncertainty projection can be performed for p map-frames each second, and that the vehicle spends t seconds in each submap, then a failure to consider all map-frames for candidacy for map-matching would occur when pt equals n . In practice, for the scale of environments considered thus far, this never occurs. The computational effort expended in the uncertainty projection step is several orders of magnitude smaller than the computational effort expended by the other parts of the algorithm for the size of environments considered in our experiments thus far.

4. Obtaining a Global Map

We are often motivated to provide a single global map of the robot's environment. For example, in Section 5 we compare

an estimated map with an architectural drawing. This “globalized” representation is a result of a post-processing procedure to find a global projection of each map-frame. In other words, we wish to find the position and orientation of each map-frame with respect to a single frame. We choose to reference all maps to the first map-frame created, frame 0.

The Dijkstra projection does this when using map-frame 0 as the source; however, it only uses a minimal subset of the edges in the graph. We wish to find a projection that incorporates all the edges.

When there are loops in the graph, there will be a disparity $v_{i,j}$ between the transformation T_i^j stored in the Atlas graph edge and the transformation derived from the global poses of each frame (T_0^i and T_0^j , respectively):

$$v_{ij} = T_i^j \oplus T_j^0 \oplus T_0^i. \quad (14)$$

We seek to find the global arrangement $\mathcal{T}^*$ of all N frames $\mathcal{T} = \{T_0^1 \dots T_0^N\}$ that minimizes this error over all edges. This can be posed as a non-linear least-squares optimization problem:

$$\mathcal{T}^* = \arg \min_{\mathcal{T}} \sum_{ij} \|v_{ij}\|^2. \quad (15)$$

We use the Dijkstra projection to compute the initial global arrangement, and the optimization typically converges in less than five iterations using the Matlab optimization toolbox.

5. Experimental Results

We have obtained extensive experimental results with the method, using both feature-based SLAM (Smith, Self, and Cheeseman 1990) (described in Appendix A) and scan-matching (described in Appendix C) as the local SLAM

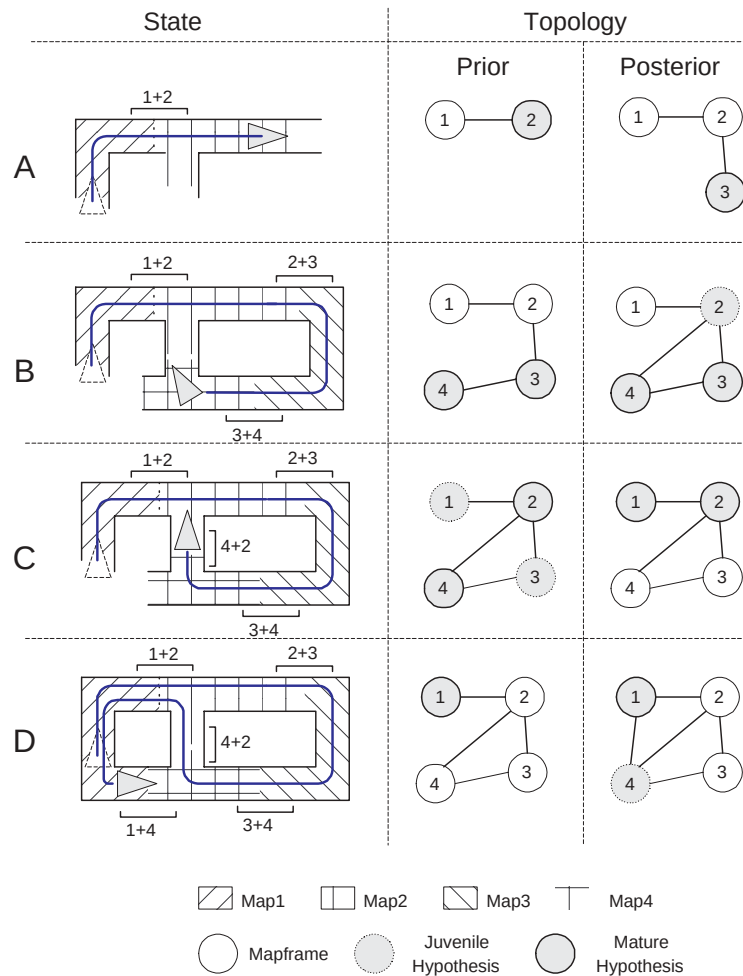


Fig. 7. The spatial and topological evolution of an Atlas. This set of diagrams highlights key aspects of the Atlas framework. The environment depicted is fictional and the robot path is construed to be illustrative. The right-hand column of this diagram uses shading to indicate the map-frames which successfully represent the local region. Note that contrary to what is suggested by the shading, the maps do in fact overlap. This intersection is denoted by labeled brackets on the spatial diagrams. Note also that the extent of a map-frames is not defined or bounded by area covered but by the set of mapped features within it. (A) Genesis. The vehicle has built two maps (1,2). Map 1 no longer explains the surroundings and is inactive. Map 2 is “full” and so the genesis of map 3 occurs. An edge is built between 2 and 3. Map 3 immediately becomes the dominant hypothesis. (B) Map-matching. Mature hypotheses are running in both maps 3 and 4. The Dijkstra projection suggests that map 4 may be “close” to map 2. The map-matching algorithm confirms this conjecture and a new link is created between 4 and 2. This is loop closure. Map-frame 2 is now adjacent to a mature frame (4) and so shortly after the edge creation a juvenile hypothesis is attached to it. (C) Hypothesis cull. The robot has recently traversed into map 2 (from 4) which has also become dominant. Juvenile hypotheses have been installed in the adjacent frames 1 and 3. A short time later the juvenile hypothesis in map 3 has been terminated; it cannot adequately explain the central corridor. The previously mature hypothesis in map 4 has also been culled for the same reasons. The juvenile hypothesis in map 1 however has been promoted to mature status. At this point, the estimate of the transformation between frames 1 and 4 can be updated using an observation constructed from the fact that the vehicle is simultaneously in maps 1 and 4. (D) Loop closure. Initially only one mature hypothesis exists (in map 1). Unlike in case (A) genesis is not imminent (low vehicle uncertainty and map is not full). Instead the map-matching algorithm conjectures that maps 1 and 4 may be adjacent. The map-matching algorithm confirms this and another new link is created. This completes the topological and spatial description of the environment.

Algorithm:

1. **Local map iteration:** execution of one cycle of the local SLAM module (e.g. feature-based or scan-match) in each non-retired map-frame using the new sensor measurements.
2. **Hypothesis state transitions:** creation, promotion, and demotion of hypotheses.
3. **Map projection iteration:** if the dominant map changes in step 2, restart the shortest path computation and clear the map-match candidate list; otherwise, step the shortest path computation and add new candidates for map-matching.
4. **Map-matching:** execute map-matching between the current dominant map and the next map-match candidate; perform cycle verification.

Fig. 8. Summary of the computation performed for each cycle of the Atlas algorithm.

method. For the results reported here, when feature-based SLAM is used as the local mapping method, we restrict the size of each local map by setting an upper bound on the number of features in a map (typically 15 line segments). When laser scan-matching is used as the local mapping method, we restrict the number of vehicle poses in each submap (typically 15 vehicle poses). We have not observed the system to be very sensitive to these choices of parameter values. (Indeed, the scan-matching implementation works successfully when each local map is restricted to be a single laser scan.)

The experiments in Sections 5.1 and 5.2 utilized a standard B21 mobile robot equipped with SICK PLS scanning laser (50 m range, 5 cm resolution) and a ring of 24 Polaroid ultrasonic sensors (7 m range, 5 mm resolution) and standard odometry based on wheel encoders. For feature-based local mapping, the method has been implemented using either laser scanner data (with line segments as features; (Castellanos and Tardós 2000) or sonar data (with points and line segments as features; Leonard et al. 2002) or both laser and sonar data being processed concurrently. Laser scan-matching in general produces more visually appealing maps, but the scan-matching approach does not generalize to noisier sonar data.

5.1. Ten loops in a Small-Scale Environment

We first show an experiment for a situation in which many loops are repeatedly performed in a relatively small-scale environment. The robot executed multiple “figure eight” maneuvers within the environment depicted in Figure 9(a). Each corridor was traversed several times in both directions. The total path driven was 690 m taking 45 min to complete. Figure 9(b) is the “dead-reckoned” path using odometry data. Figure 9(c) shows the output produced for a “single map” scan-matching solution. The map contains 200 vehicle poses. Figure 9(d) shows the output produced for a “single map” laser feature-based solution. Note that the map from scan-matching

“looks better” than the laser feature-based map, which has some “doubled features” in it.

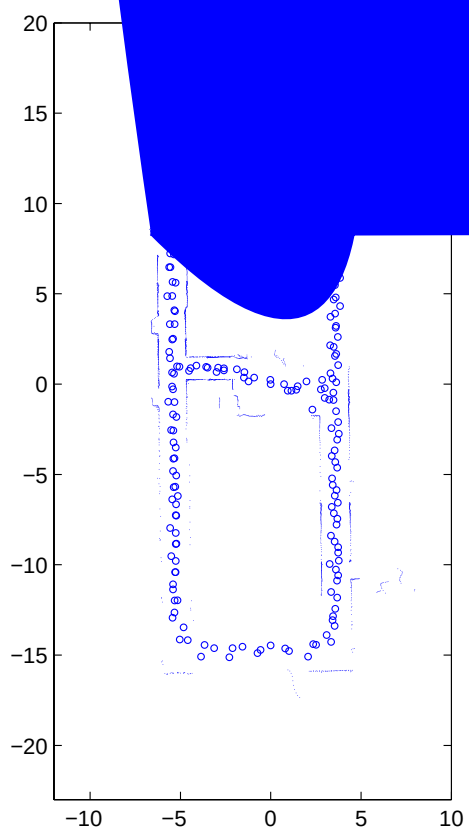
Figure 10(a) shows the Atlas output for this environment with laser feature-based local mapping. A total of 12 map-frames were created. Figure 10(b) shows the adjacency matrix of the Atlas graph. Figure 10(c) plots the ID of the dominant map versus time for this experiment, as well as the number of active hypotheses versus time and the total amount of time spent in each map-frame. About halfway through the run, the algorithm reaches a steady state and does not create any additional maps; the largest map ID no longer increases. Note that the average number of active map-frames (containing a mature hypothesis) at any instant reaches a steady-state value.

Figure 11 shows the output when scan-matching is used as the local mapping module. The maximum number of vehicle poses in a map-frame was set to 15, and 15 map-frames were created.

5.2. Killian Court, MIT

In this section we present results for processing of data from a long-duration mission performed at MIT in Killian Court, a cluster of interconnected buildings at the center of the MIT campus. Figure 12(a) shows the topological path of the vehicle superimposed on an architectural drawing of the Killian Court area. The mission had a path length of approximately 2.2 km and a duration of 2.5 h. The route contained nested loops of various sizes and topologies. Figure 12(b) shows the dead-reckoned path resulting from simply integrating the odometry data. The dead-reckoning path clearly reveals systematic biases in the odometry. For the results reported below, we augment the state vector with a several bias parameters that are estimated on-line.

Figure 13(a) shows the result of applying the global optimization described in Section 4 to the Atlas output based on laser scanner data and odometry. A total of 101 map-frames



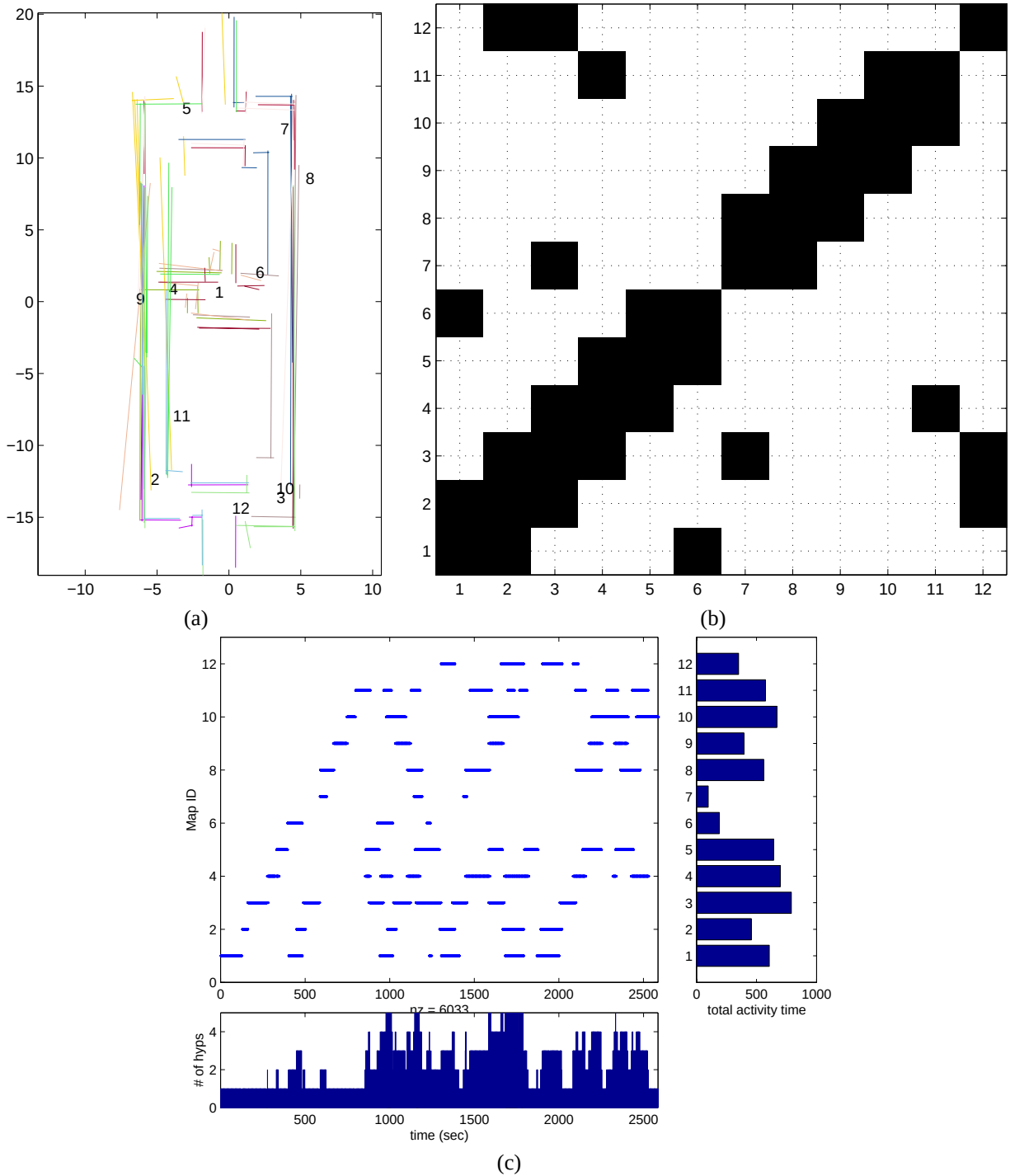


Fig. 10. (a) Atlas optimized map, using laser feature-based SLAM as the local mapping method. The numbers represent the map IDs and are positioned near the origins of each map. Please note that the symbol adjacency is not necessarily indicative of map adjacency (b) Adjacency matrix. (c) Map-frame genesis and activity. The generation of new map-frames drops off as the area becomes fully mapped and maps are reused. Each dash in the central figure corresponds to a period in which a particular map-frame was supporting an active, successful hypothesis. In this implementation, map-frames are not deleted and so frames with erroneous maps might never be selected for reuse. Maps 6 and 7 are such a case; the right-hand figure indicates correspondingly small amounts of activity time. The lower figure shows how the average number of active hypotheses remains constant throughout the experiment.

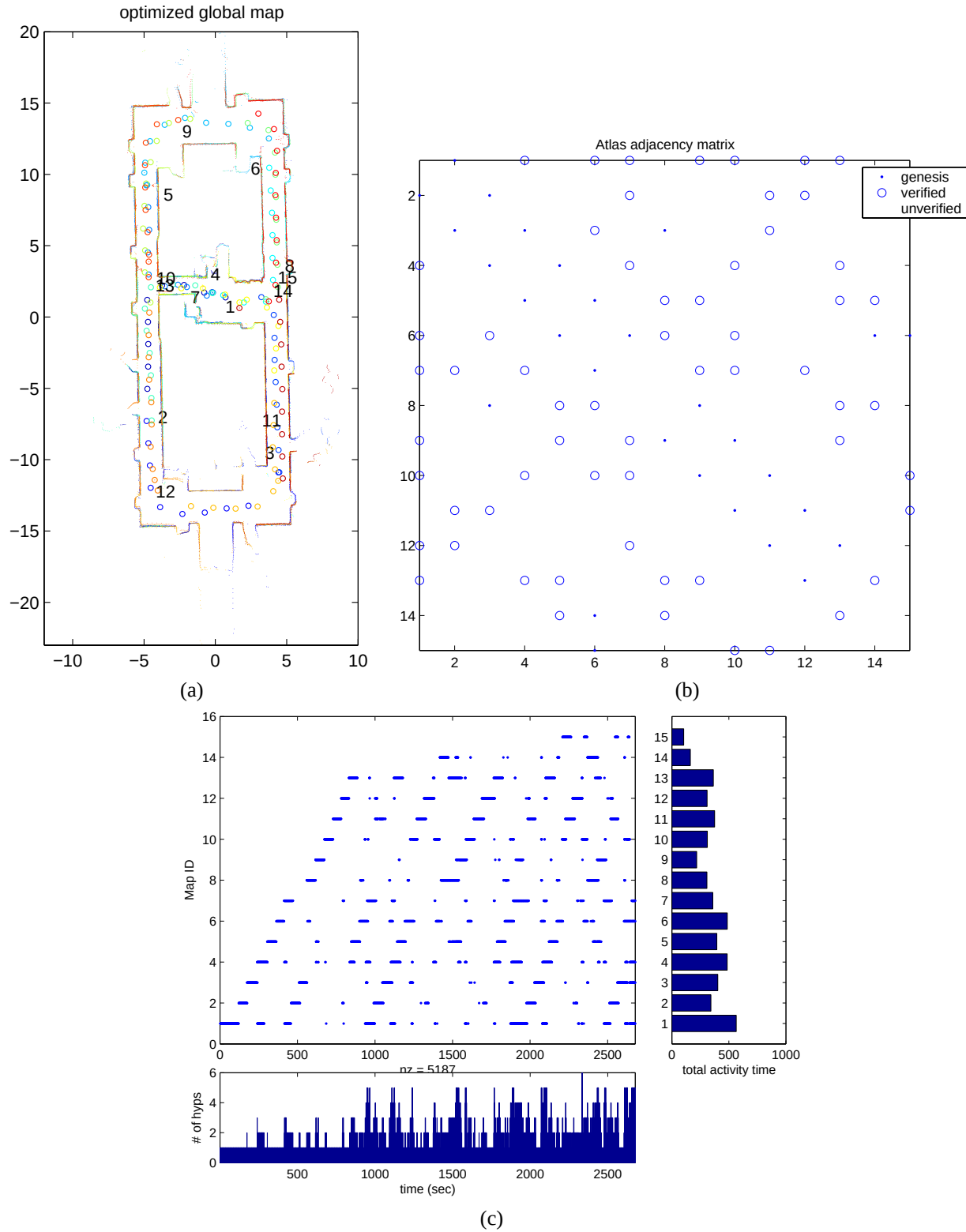


Fig. 11. (a) Atlas optimized map, using scan-matching as the local mapping method. (b) Adjacency matrix. (c) Map-frame genesis and activity.

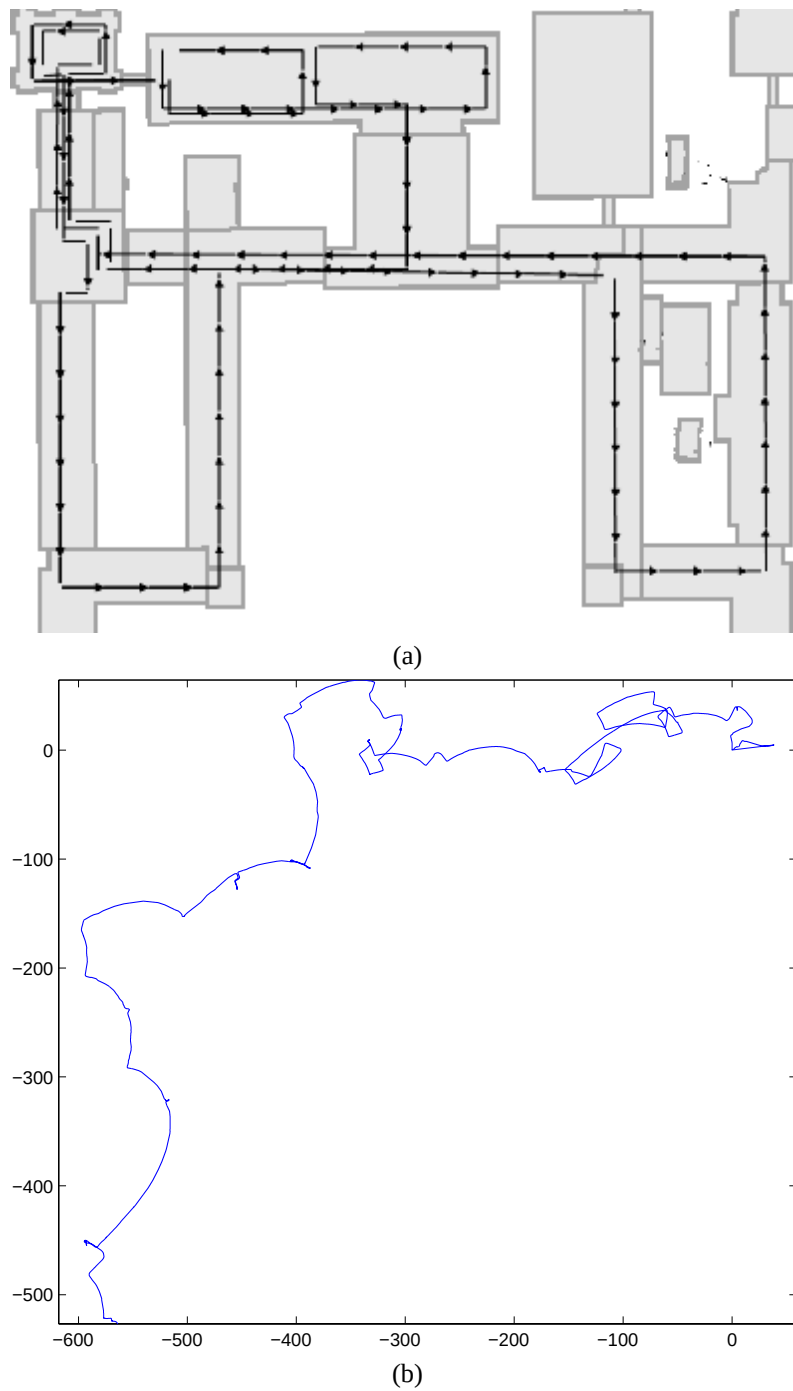


Fig. 12. (a) The manually drawn topology of the driven route overlaid on an architectural drawing of part of the MIT campus (Killian Court). The large east-west passage, known as the “infinite corridor”, is approximately 225 m long. (b) The trajectory derived from odometry alone.

were built, each containing a maximum of 15 mapped line segments. Figure 13(b) shows the instantaneous sum of kernel and user time for the Atlas process as well as its smoothed value. Note that as more features are mapped and more map-frames are created the mean processor load remains constant. Figure 13(c) also plots the numerical label of the dominant map-frame with time. During map-frame genesis, a counter is incremented and the newly created map is labeled with its value. As new ground is covered, the value of the dominant map ID increases. When the robot returns, however, to a previously mapped area, the dominant map ID will decrease if loop closure is successful. For example, approximately 1 h into the experiment the robot returned to an area first mapped 45 min earlier. Similarly, after 2 h 15 min, the vehicle returned to a region mapped just 5 min after the experiment began. The experiment demonstrates the ability of Atlas to close multiple, nested loops.

Figure 14 shows results using data from the same experiment but using the Polaroid sonar data instead of the laser scanner data. The local SLAM method used is described fully in Leonard et al. (2002). Figure 15 shows results using data from the same data set when sonar and laser data are processed concurrently. Figure 15(a) shows the global optimized map, Figure 15(b) shows the map adjacency matrix, and Figure 15(c) shows the active map ID versus time.

Figure 16 shows results using data from the same experiment but using laser scan-matching as the local SLAM module. The maximum number of vehicle poses in a map-frame was set to 15. Figure 16(a) shows the global optimized map, Figure 16(b) shows the map adjacency matrix, and Figure 16(c) shows the active map ID versus time. The visual quality of the final optimized map is better using scan-matching instead of feature-based SLAM; however, the scan-matching approach would not be successful with reduced quality data, such as sonar.

5.3. Victoria Park Using Scan-Matching

We next provide results using scan-matching as the local SLAM module for an outdoor data set, made publicly available by E. Nebot of the University of Sydney. Results for this data set were first published by Guivant and Nebot (2001), using the compressed filter, an efficient method for large-scale SLAM based on the Kalman filter. We refer the reader to Guivant and Nebot (2001) for a detailed description of the experimental setup for acquisition of this data. Processing results for the same data set have been published by Liu and Thrun (2003), using sparse extended information filters. These researchers have used trees as discrete point features in processing of this data set. In contrast, we have processed the data using scan-matching as the local mapping module. These data use a sensor with a much longer range than the indoor scanner used in the Killian Court data set, and mounted on a vehicle with significantly different dynamics. The maximum

number of vehicle poses in a map-frame was set to ten. Using algorithm parameters nearly identical to those used to obtain Figure 16, the output of Atlas is shown in Figure 17.

5.4. Pennsylvania Mine Scan-Match

Finally, we present results for an underground mine data set, made available by S. Thrun of Carnegie Mellon University. Processing results for this data set using sparse extended information filters are available in Thrun et al. (2003). This data set is more challenging because it consists exclusively of laser scanner data, with no odometry. To cope with the absence of odometry data, a motion model was assumed with the vehicle traveling forward at a velocity of 17.5 cm s^{-1} (with a large corresponding uncertainty). For scan-matching, the maximum number of vehicle poses in a map-frame was 15. The output of Atlas is shown in Figure 18.

6. Conclusion

In this paper we have presented Atlas, a general framework for efficient large-scale mapping and navigation. The performance of the approach has been verified using both laser scanner and ultrasonic range data, demonstrating the capability to perform SLAM in large areas comprised of multiple, nested loops in real time. Results have been presented for both feature-based local SLAM (Smith, Self, and Cheeseman 1990) and scan-matching.

The method achieves a growth of complexity of either $\mathcal{O}(n \log n)$ when using the Dijkstra shortest path algorithm to select candidates for map-matching, or $\mathcal{O}(n)$ when BFS is used for this task. Amortization of this computation over the time spent in each submap results in extremely efficient performance. An off-line global alignment step is utilized to generate a single global map for visualization purposes at the end of a mission. This method operates extremely quickly (a few seconds of computation time) for the size of environments considered in this paper.

Several important issues remain for future work. For many real-world SLAM problems in which biases and nonlinearities are significant, the “standard” full $\mathcal{O}(n^2)$ solution can fail badly. For a “true” linear Gaussian SLAM problem with known data association, a submap approach such as Atlas will yield state estimation errors that are larger than the full $\mathcal{O}(n^2)$ solution. In this case, Atlas is indeed suboptimal. The issues of the rate of convergence and the “price paid” for using multiple submaps require further study. In related research, we have shown that when the local origins of submaps are defined by map features shared between adjacent submaps, then one can achieve asymptotic convergence to a solution that is effectively the same as the full solution, for situations when the robot can make repeated traversals of the environment (Leonard and Newman 2003).

This paper has not addressed the issue of achieving global convergence for repeated traversals of the environment. In

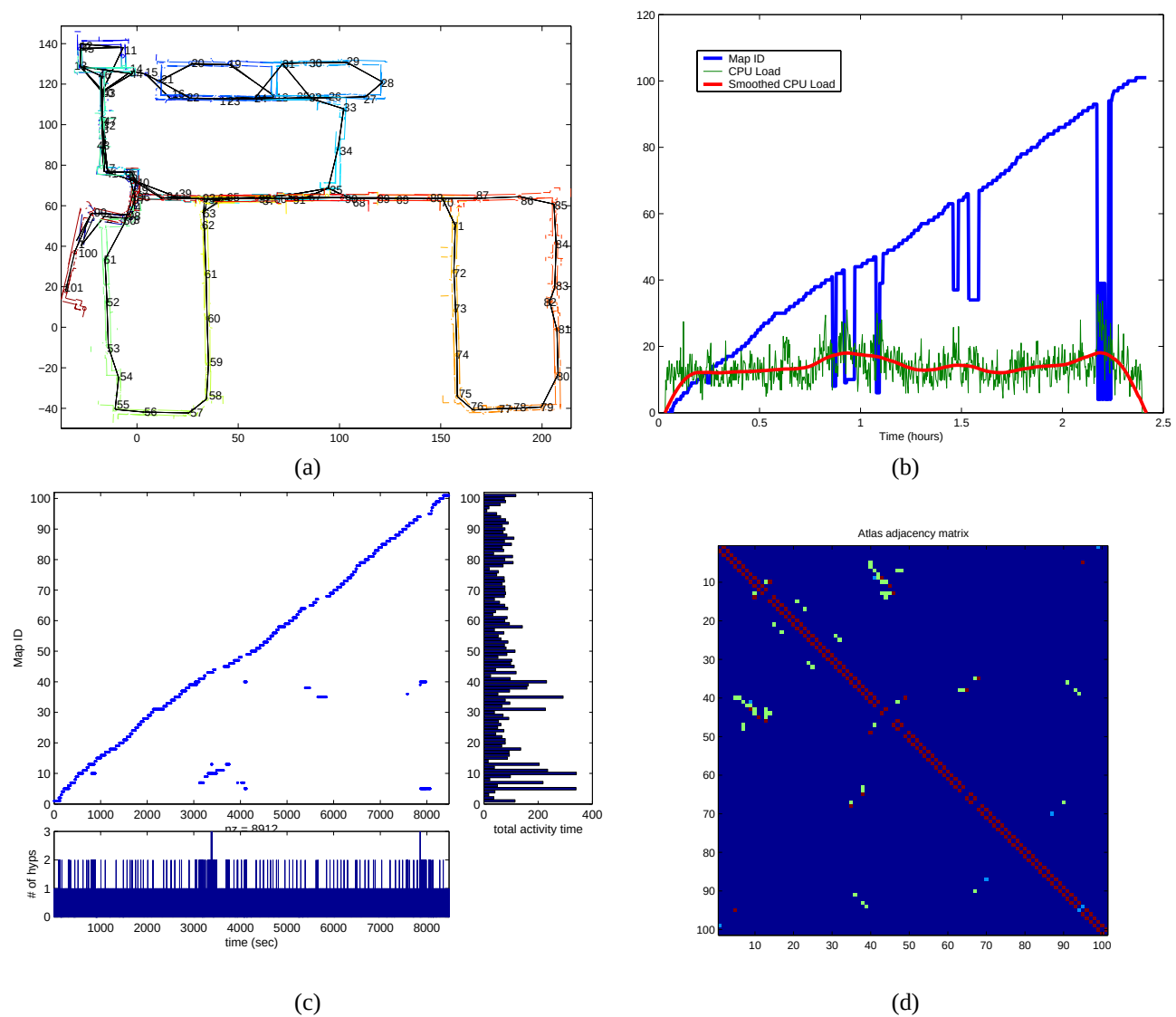


Fig. 13. (a) Global optimized map and Atlas graph for processing of laser data (feature-based local SLAM). (b) Processor load and current map ID for laser data run. The general linear increase in current map ID is indicative of the mapping of new areas. The occasional “fallback” to a map with a lower ID represents successful loop closing—the reuse of an existing map. (The processor load is plotted in arbitrary units.) (c) Map ID versus time, total activity versus map ID, and the number of active hypotheses versus time for the laser feature-based SLAM processing. (d) Adjacency matrix. The beginning and end of the experiment occur on a different floor of the building. The vehicle started on the fourth floor of MIT building 5 and was manually steered to the elevator in MIT building 7 (approximately 5 min into the mission) and was taken to the third floor. Approximately 10 min from the end of the mission, the vehicle was taken back into the elevator and back to the fourth floor, traveling down past the origin to a portion of the environment not previously visited. Hence, the final segment of the trajectory (the leftmost part of the map) does not align well with the longer corridor, which is actually on the third floor of the building (see Extension 1).

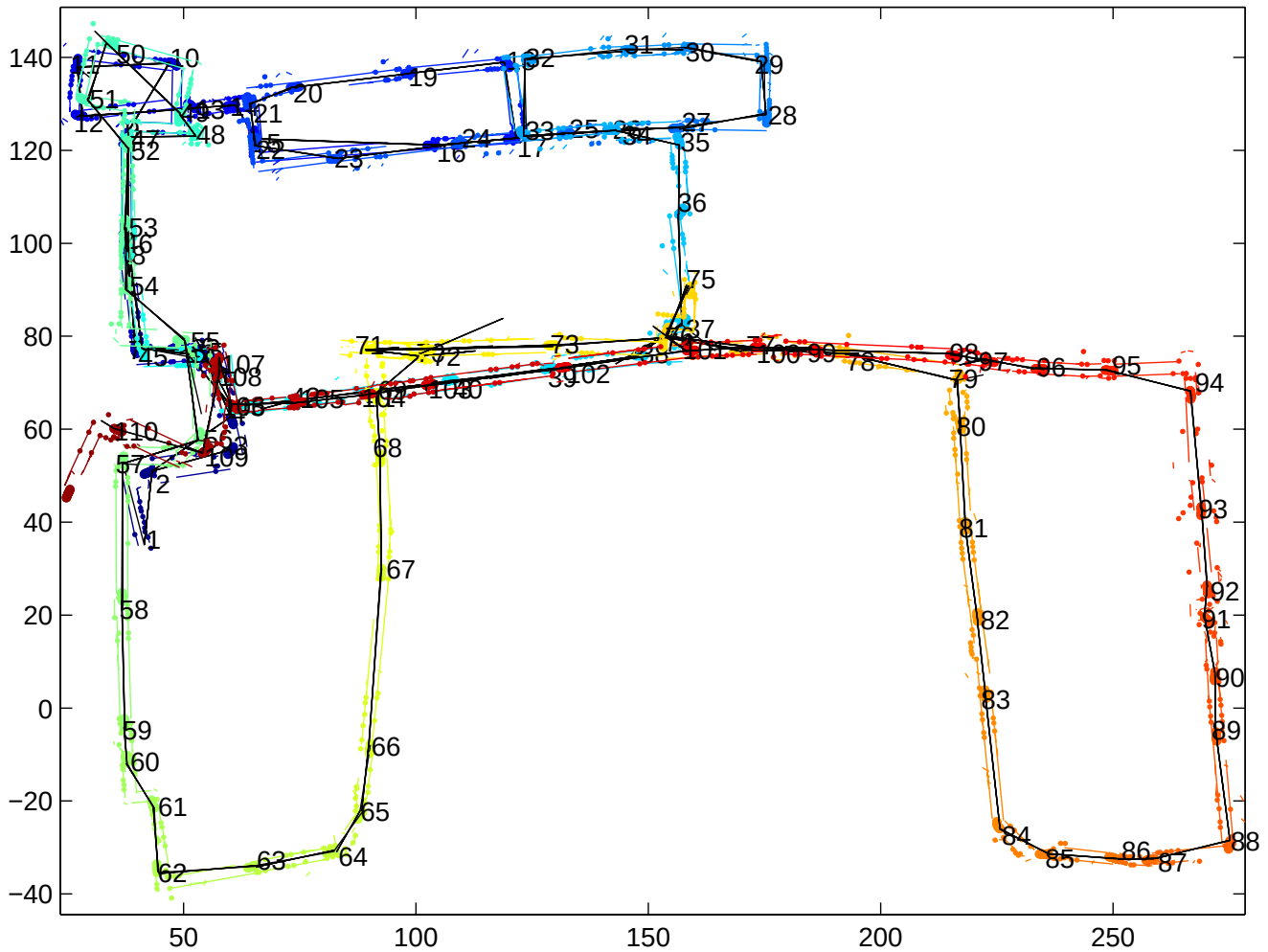


Fig. 14. Global optimized map and Atlas graph for processing of feature-based SLAM processing of sonar and odometry data. The local SLAM algorithm, described in Leonard et al. (2002), uses random sample consensus for feature association and classification and multiple vehicle poses in the state vector to enable consistent feature initialization from multiple vantage points with wide-beam sonar measurements. Map-matching is more difficult with sonar. The results indicate situations where Atlas fails to detect some loop closure events that are detected with the more accurate laser scanner data. For example, maps 73 and 102 fail to produce a verifiable match.

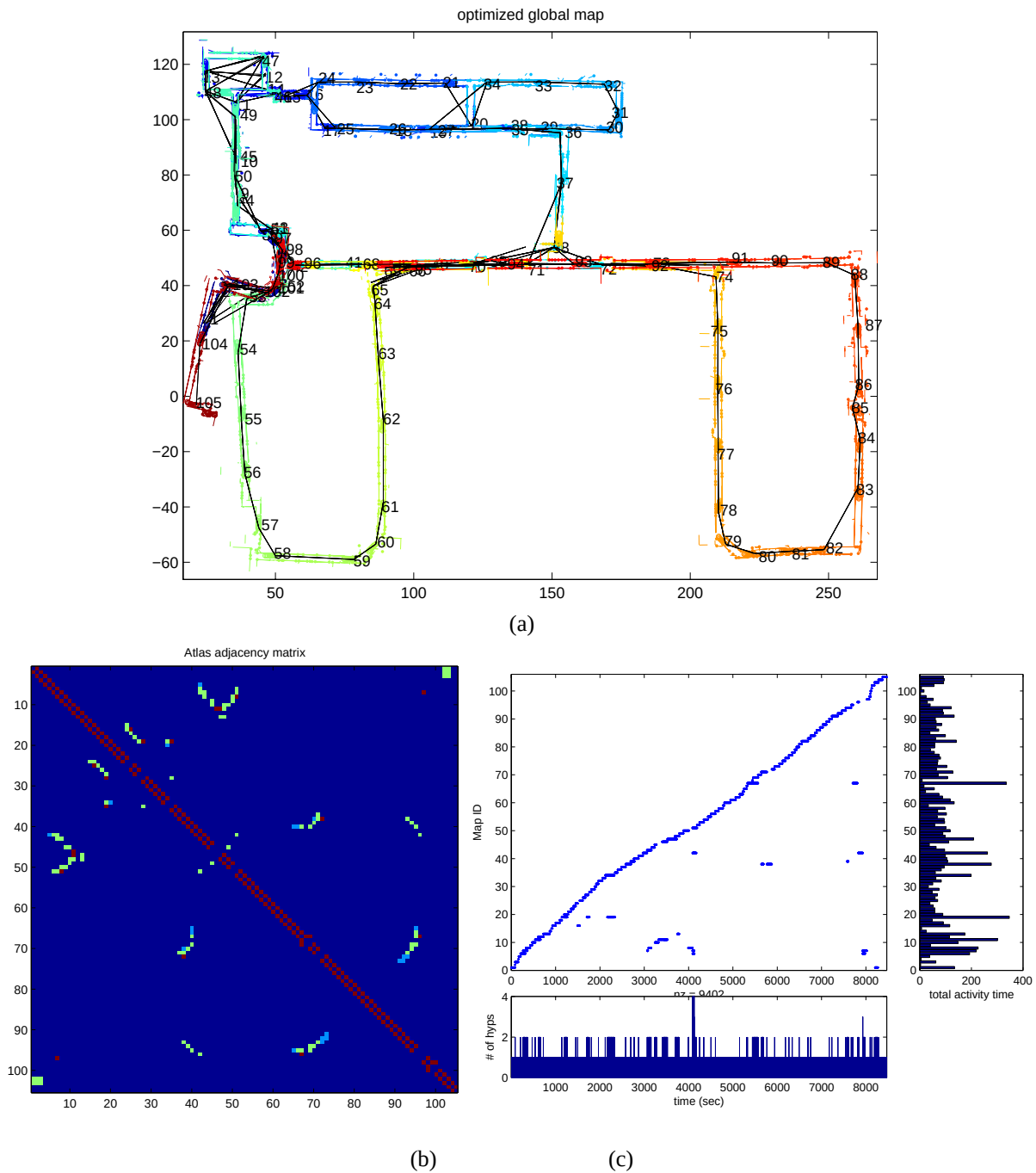


Fig. 15. (a) Global optimized map and Atlas graph for processing of the Killian Court data set using laser and sonar data concurrently (feature-based local SLAM). (b) Map adjacency matrix. (c) Map times. Extension 1 provides a full replay of the data processing. Feature-based SLAM processing with sonar and laser concurrently performs better than processing laser only, in part because the laser scanner has only a 180° field of view (looking forwards), whereas the sonar ring provides 360° coverage. This is advantageous when attempting map-matches for traversals of a corridor in opposite directions.

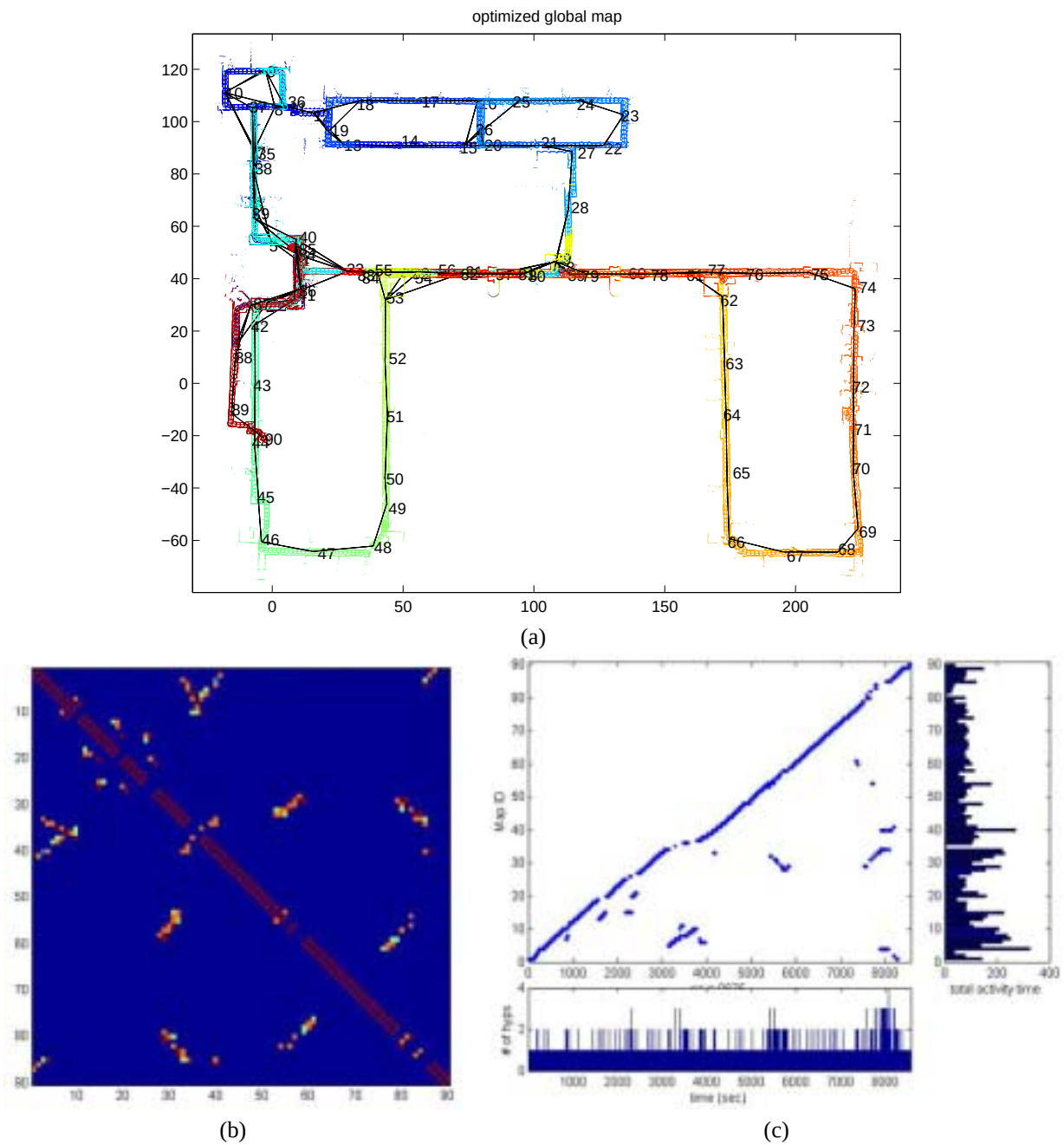


Fig. 16. (a) Global optimized map and Atlas graph for processing of the Killian Court data set using laser scan-matching as the local mapping method (Appendix C). (b) Map adjacency matrix. (c) Map times.

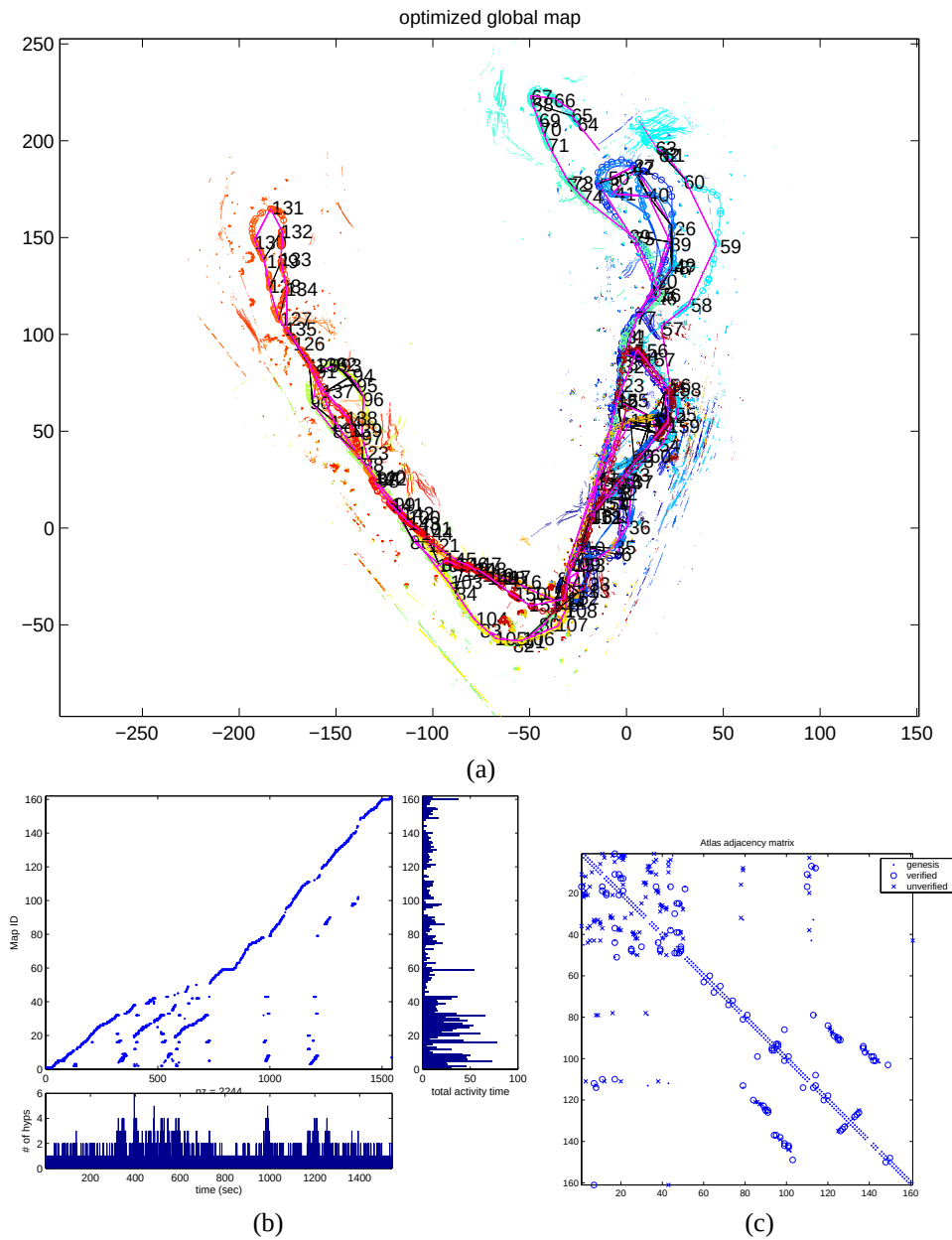


Fig. 17. (a) Optimized map of Victoria Park data set using Atlas scan-matching. (b) Map times. (c) Adjacency matrix. See Guivant and Nebot (2001) and Liu and Thrun (2003) for featured-based SLAM processing results for this data set.

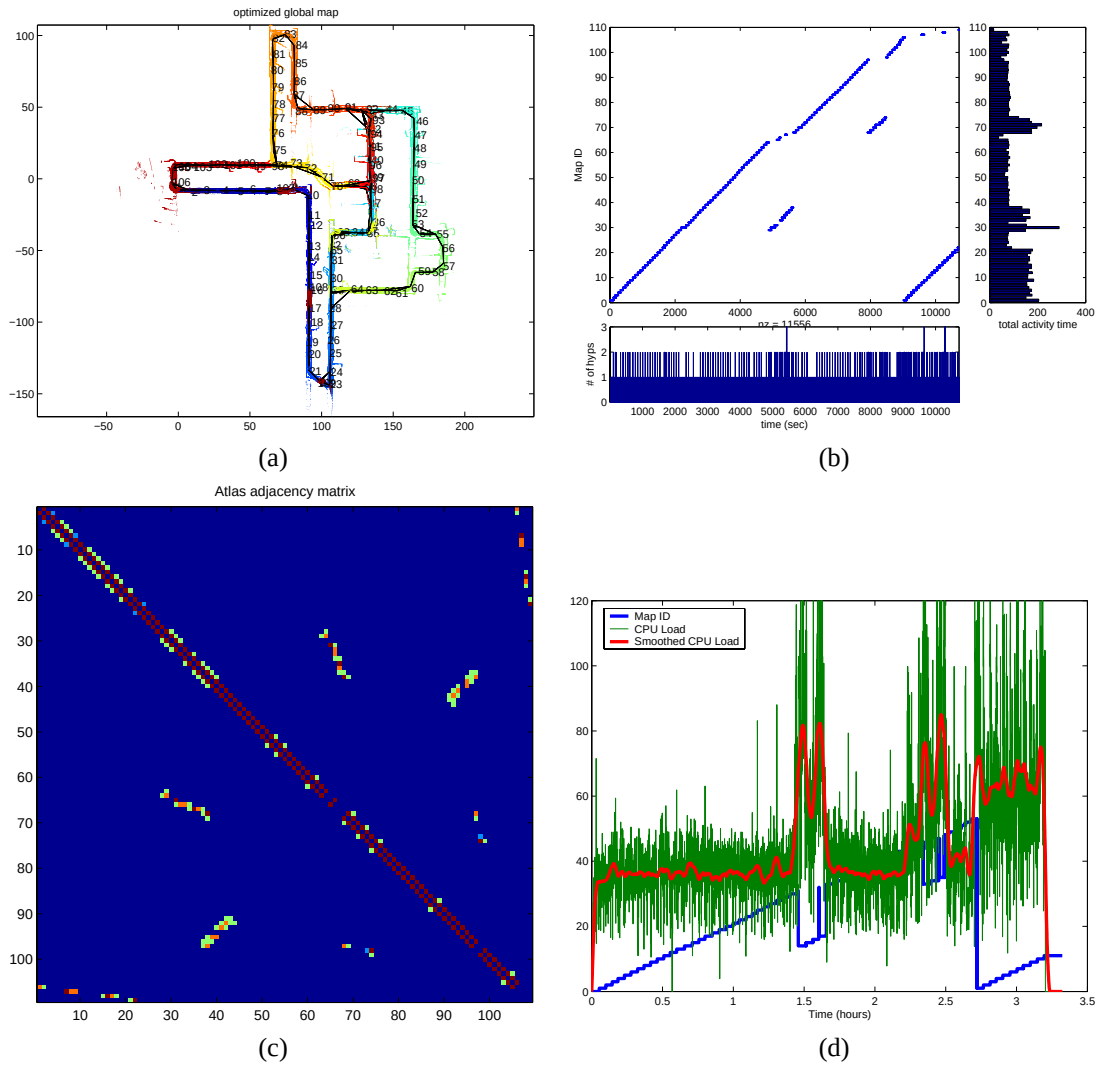


Fig. 18. (a) Optimized map of the Pennsylvania Mine data set using Atlas scan-matching (data made available by S. Thrun; Thrun et al. 2003). (b) Map times. (c) Adjacency matrix. (d) Processing time. 58 min of CPU time were required to process the 178 min of data. The processing rate is proportional to the number of scans matched. We can see about a twofold increase in revisited areas as opposed to new areas, since there are saved scans in front of and behind the robot in the former. Notice that there is no noticeable processing delay when closing a loop. Extension 3 provides a full replay of the data processing.

related research, we have developed techniques whereby, with each new map transition, we can also improve our estimate of the transformation between frames. A related subject is the issue of map fusion and map “clean-up”. Currently, with the conservative approach to map-matching and loop closure adopted in Atlas, the algorithm can generate multiple overlapping submaps for the same region of the environment. For example, in Figure 10, the algorithm produced 12 map-frames to cover the area of the “ten loops” experiment. Using a technique such as sequential map joining (Tardós et al. 2002), these 12 map-frames could be combined to form a single map. For larger-scale environments, such as the Killian Court data set, the combination of many map-frames into a single map would likely fail due to linearization errors.

Ongoing research efforts include the extension of the approach to accommodate 3D, omni-directional video camera data and underwater sonar data.

Appendix A: Feature-Based Local Navigation and Mapping

For the results presented in this paper we chose to adopt a common approach to the SLAM problem: feature-based stochastic mapping using an extended Kalman filter (EKF). There is a rich corpus of literature on this subject (Smith, Self, and Cheeseman 1990). For completeness, the EKF-derived feature-based SLAM algorithm is summarized below.

We seek an estimate $\hat{\mathbf{x}}(i|j)$ at time i using vehicle relative observations of features made up until time j of a state vector defined to be concatenation of vehicle and n feature locations (the map) and its covariance $\mathbf{P}(i|j)$:

$$\begin{aligned}\hat{\mathbf{x}}(i|j) &= [\hat{\mathbf{x}}_v(i|j)^T, \hat{\mathbf{x}}_1(i|j)^T \cdots \hat{\mathbf{x}}_n(i|j)^T]^T \\ &= [\hat{\mathbf{x}}_v(i|j)^T, \hat{\mathbf{x}}_m(i|j)^T]^T \\ \mathbf{P}(i|j) &= \begin{bmatrix} \mathbf{P}_{vv}(i|j) & \mathbf{P}_{vm}(i|j) \\ \mathbf{P}_{vm}(i|j)^T & \mathbf{P}_{mm}(i|j) \end{bmatrix}\end{aligned}$$

The state vector to be estimated $\hat{\mathbf{x}}$ evolves according to a non-linear process model:

$$\mathbf{x}(k+1) = \mathbf{F}(\mathbf{x}(k), \mathbf{u}(k+1)) + \mathbf{v}_v(k+1).$$

Here, $\mathbf{v}$ is a process noise vector which is assumed to be a zero mean, temporally uncorrelated random sequence with covariance given by

$$\begin{aligned}\mathbf{E}[\mathbf{v}(k)] &= \mathbf{0} \\ \mathbf{E}[\mathbf{v}(i) \cdot \mathbf{v}(j)^T] &= \begin{cases} \mathbf{Q}(k) & \text{if } i = j = k, \\ \mathbf{0} & \text{otherwise.} \end{cases}\end{aligned}$$

If we assume that features do not move, the process model can be simplified to

$$\mathbf{x}(k+1) = \begin{bmatrix} \mathbf{F}_v(\mathbf{x}_v(k), \mathbf{u}(k+1)) \\ \mathbf{x}_m(k) \end{bmatrix} + \begin{bmatrix} \mathbf{v}_v(k+1) \\ \mathbf{0} \end{bmatrix}.$$

Observations of features are modeled using a non-linear relationship between state and measurements

$$\hat{\mathbf{z}}(k) = \mathbf{H}(\mathbf{x}(k)) + \mathbf{w}(k).$$

The terms $\mathbf{w}$ is an observation noise vector, which is assumed to be a zero mean, temporally uncorrelated random sequence with covariance given by

$$\begin{aligned}\mathbf{E}[\mathbf{w}(k)] &= \mathbf{0} \\ \mathbf{E}[\mathbf{w}(i) \cdot \mathbf{w}(j)^T] &= \begin{cases} \mathbf{R}(k) & \text{if } i = j = k, \\ \mathbf{0} & \text{otherwise.} \end{cases}\end{aligned}$$

The prediction equations for the EKF are written as follows:

$$\begin{aligned}\hat{\mathbf{x}}(k+1|k) &= \mathbf{F}(\hat{\mathbf{x}}(k|k), \mathbf{u}(k+1), k+1) \\ \mathbf{P}(k+1|k) &= \nabla \mathbf{F}_{\hat{\mathbf{x}}} \mathbf{P}(k|k) \nabla \mathbf{F}_{\hat{\mathbf{x}}}^T + \mathbf{Q}(k+1) \\ \hat{\mathbf{z}}(k+1|k) &= \mathbf{H}(\hat{\mathbf{x}}(k+1|k), k+1).\end{aligned}$$

The term $\nabla \cdot_{\hat{\mathbf{x}}}$ is understood to be the Jacobian of $(\cdot)$ with respect to $\mathbf{x}$ evaluated at $\hat{\mathbf{x}}(k+1|k)$.

Following the prediction, the new observation $\mathbf{z}(k+1)$ is fused with the prior estimate by application of the following equations:

$$\begin{aligned}\hat{\mathbf{x}}(k+1|k+1) &= \hat{\mathbf{x}}(k+1|k) + \mathbf{W}(k+1)\mathbf{v}(k+1) \\ \mathbf{P}(k+1|k+1) &= \mathbf{P}(k+1|k) - \mathbf{W}(k+1)\mathbf{S}_{vv}\mathbf{W}^T(k+1)\end{aligned}$$

where

$$\begin{aligned}\mathbf{v}(k+1|k) &= \mathbf{z}(k+1) - \hat{\mathbf{z}}(k+1|k) \\ \mathbf{S}_{vv} &= \nabla \mathbf{H}_{\hat{\mathbf{x}}} \mathbf{P}(k+1|k) \nabla \mathbf{H}_{\hat{\mathbf{x}}}^T + \mathbf{R}(k+1) \\ \mathbf{W}(k+1) &= \mathbf{P}(k+1|k) \nabla \mathbf{H}_{\hat{\mathbf{x}}}^T \mathbf{S}_{vv}^{-1}.\end{aligned}$$

The nature of the SLAM problem results in sparse Jacobians of the observation and process models. Under the assumption of static features $\nabla \mathbf{F}_{\mathbf{x}}$ becomes

$$\nabla \mathbf{F}_{\mathbf{x}} = \begin{bmatrix} \nabla \mathbf{F}_v & \mathbf{0} & \cdots & \cdots \\ \mathbf{0} & \mathbf{I} & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \mathbf{I} \end{bmatrix}.$$

The observation of feature i is a function of only the vehicle state and the feature being observed. Hence the observation Jacobian is highly sparse and can be written in block form as

$$\nabla \mathbf{H}_{\mathbf{x}} = [\nabla \mathbf{H}_v \cdots \mathbf{0} \cdots \nabla \mathbf{H}_i \cdots \mathbf{0}] \quad (16)$$

where $\nabla \mathbf{H}_i$ is the derivative with respect to the i th feature and $\nabla \mathbf{H}_v$ is the derivative with respect to the vehicle pose. With n features, the sparseness of these matrices allows the prediction step to be performed $\mathcal{O}(n)$ and the update in $\mathcal{O}(n^2)$.

An important aspect of this algorithm is that the dimension of the state vector is not constant; it grows as more features are observed and mapped. An observation $\mathbf{z}$ of an unmapped feature can be used to initialize a new feature via an initialization function $\mathbf{g}(\hat{\mathbf{x}}, \mathbf{z})$:

$$\begin{aligned}\hat{\mathbf{x}} &\Rightarrow [\hat{\mathbf{x}}^T, \mathbf{g}(\hat{\mathbf{x}}, \mathbf{z})^T]^T \\ \mathbf{P} &\Rightarrow \begin{bmatrix} \mathbf{P} & \mathbf{P}\nabla\mathbf{G}_x^T \\ \nabla\mathbf{G}_x\mathbf{P} & \nabla\mathbf{G}_x\mathbf{P}\nabla\mathbf{G}_x + \nabla\mathbf{G}_z\mathbf{R}\nabla\mathbf{G}_z \end{bmatrix}\end{aligned}$$

The association of measurements to features can be accomplished via the commonly applied nearest-neighbor gating technique as described in Dissanayake et al. (2001) and Bar-Shalom and Fortmann (1988).

Appendix B: Jacobians of Transformation Compositions

In this appendix we briefly review our notation for representing uncertain coordinate frame transformations. See Castellanos and Tardós (2000) for more detail.

In the case of two dimensions, we can represent a coordinate frame transformation as a translation followed by a rotation, parametrized as

$$T_i^j = \begin{bmatrix} x_i \\ y_i \\ \theta_i \end{bmatrix},$$

which describes the transformation of coordinates in frame j to frame i .

We can compose transformations with the $\oplus$ operator as follows

$$\begin{aligned}T_i^k &= T_i^j \oplus T_j^k \\ &= \begin{bmatrix} x_i + c_i x_j - s_i y_j \\ y_i + s_i x_j + c_i y_j \\ \theta_i + \theta_j \end{bmatrix}\end{aligned}$$

where c_i and s_i are $\cos(\theta_i)$ and $\sin(\theta_i)$, respectively.

When transforming the covariances of transformations, we need to evaluate the Jacobians of the transformation operator with respect to either of its arguments. The Jacobian of the T_i^k with respect to the first argument of $\oplus$ is $J_1(T_i^j, T_j^k)$:

$$J_1(T_i^j, T_j^k) = \begin{bmatrix} 1 & 0 & -s_i x_j - c_i y_j \\ 0 & 1 & c_i x_j - s_i y_j \\ 0 & 0 & 1 \end{bmatrix}.$$

Likewise, the Jacobian of the T_i^k with respect to the second argument of $\oplus$ is $J_2(T_i^j, T_j^k)$:

$$J_2(T_i^j, T_j^k) = \begin{bmatrix} c_i & -s_i & 0 \\ s_i & c_i & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The operator $\ominus$ forms the inverse transformation $T_j^i = \ominus T_i^j$ such that $T_j^i \ominus T_i^j = I$, where I is the identity transform. The Jacobian of T_j^i with respect to T_i^j is $J_\ominus(T_i^j)$.

$$J_\ominus(T_i^j) = \begin{bmatrix} -c_i & -s_i & s_i x_i - c_i y_i \\ s_i & -c_i & c_i x_i + s_i y_i \\ 0 & 0 & -1 \end{bmatrix}.$$

Appendix C: Local SLAM Based on Scan-Matching

Scan-matching is another local navigation module for Atlas. It uses the laser scan points directly instead of extracting features such as lines from the scans. Scan-matching has been a popular approach to SLAM with dense laser scanner data (Lu and Milios 1997; Gutmann and Konolige 1999; Thrun 2001). In this appendix, we describe a novel implementation that combines scan-matching with a linear Gaussian state estimation formulation (Smith, Self, and Cheeseman 1990). The key to the method is to use past vehicle poses as elements in the SLAM state vector (Leonard et al. 2002), and using scan-matching to formulate measurements that are a function of two different vehicle poses. This provides an effective means to obtain uncertainty estimates for SLAM with scan-matching, an issue that has been identified as problematic in previous research with scan-matching (Wang, Thorpe, and Thrun 2003).

The map representation is a collection of laser scans. Each laser scan is associated with a saved robot pose. The map state vector is the concatenation of the current robot pose and all the saved robot poses. We maintain the joint probability of all the poses with a single multivariate Gaussian probability distribution function. Scans are matched with the ICP algorithm (Besl and McKay 1992). The ICP algorithm produces the relative transformation between two scans, its uncertainty, and an indication of how much the scans overlap.

ICP is a fairly simple algorithm that is used to align two clouds of points with unknown correspondences. The algorithm proceeds in two steps. In the first step, point correspondences are found by matching all the points from one scan to the closest point in the other scan. The second step then finds a coordinate transformation that minimizes the error between the matched point correspondences. These two steps are repeated until convergence is achieved or a maximum number of iterations has occurred.

The algorithm can be improved when surface normals for each point are available. The normals are used in the first step to limit matches between points with normals that are not pointing in the same general direction. In the second step, the error of the point match is only considered in the direction of the surface normal. This alleviates the issue when a surface is not sampled at exactly the same points in the two scans.

To mitigate the effect of outliers in the data (from non-overlapping regions of the scans, or moving objects) the

error of the matches is modified by a Lorentzian which smoothly downweights errors when they become too large. The Lorentzian weighting is equivalent to assuming a Cauchy (instead of a Gaussian) distribution of errors. Even though the optimal minimization of a Cauchy distribution of errors is non-linear and requires multiple iterations for convergences, we have found that one iteration is sufficient since the minimization step is repeated in the iterations of the ICP algorithm anyway.

The current laser scan is matched (with ICP) in succession to each of the saved scans in the current map. If there is significant overlap between the two scans, the transformation from the ICP algorithm is treated as an observation of the relative pose between the two scans.

The maps are naturally bounded by limiting the number of saved scan poses. A new scan pose is added to the map after a fixed distance of robot travel until the map is full. Additionally, a new scan pose is added to the map if none of the previous scans matched more than 50% with the current scan.

When using scan-matching as the local mapping strategy for Atlas, the map-matching module also uses ICP. Since there is typically a larger uncertainty in the initial arrangement between the maps, we need to take some extra steps to ensure that the ICP algorithm converges quickly and on the global minimum.

First, all the points from each saved scan are transferred to the base coordinate frame of the map. Then a down-sampled version of the scan is formed with the aid of a grid. The down-sampled scan consists of the average of all the points from the original scans that fall in each grid cell. A modified ICP algorithm is then run on the down-sampled scans. The ICP is modified to allow matches, not just between nearest pairs of points, but from all points in the second scan that are within a distance threshold of points in the first scan. These multiple point correspondences enable the algorithm to find a transformation within the catchment basin of the global minimum error match.

Once the modified low-resolution ICP match has converged, its output transformation will be slightly biased by the inclusion of multiple correspondences for each point. This transformation is used, however, to initialize a regular ICP match using the full-resolution scans. Since the full-resolution scan is initialized with a transformation that is much better guess, it is less likely to become stuck in a false local minimum of the error function.

Unlikely map-matches can be quickly detected after the low-resolution ICP converges (or does not). It is often unnecessary to attempt the computationally intensive full-resolution match.

Appendix D: Index to Multimedia Extensions

The multimedia extension page is found at <http://www.ijrr.org>. Extension 1 is a video that illustrates the process-

ing sequence for the the Killian Court data set using feature-based local SLAM processing laser and sonar data concurrently (Figure 15). Extension 2 is the raw data file for the Killian Court data set, in CARMEN log file format. Extension 3 is a video that illustrates processing of the CMU mines data set using scan-matching as the local SLAM algorithm (Figure 18). Extension 1 is narrated and provides a description of the annotations and colors used in the two movies.

Table of Multimedia Extensions

Extension	Type	Description
1	Video	Processing of the Killian Court data set using feature-based local SLAM processing laser and sonar data concurrently (Figure 15)
2	CLF data file	Raw Killian Court data set
3	Video	Processing of the CMU mines data set using scan-matching as the local SLAM algorithm (Figure 18)

References

- Austin, D. and Jensfelt, P. 2000. Using multiple Gaussian hypotheses to represent probability distributions for mobile robot localization. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, San Francisco, CA, April 24–28.
- Bailey, T. and Nebot, E. 2001. Localization in large-scale environments. *Robotics and Autonomous Systems* 37(4):261–281.
- Bar-Shalom, Y. and Fortmann, T. E. 1988. *Tracking and Data Association*, Academic, New York.
- Besl, P. J. and McKay, N. D. 1992. A method for registration of 3D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14:239–256.
- Castellanos, J. A. and Tardós, J. D. 2000. *Mobile Robot Localization and Map Building: A Multisensor Fusion Approach*, Kluwer Academic, Boston, MA.
- Chong, K. and Kleeman, L. 1997a. Large-scale sonarray mapping using multiple connected local maps. *Proceedings of the International Conference on Field and Service Robotics*, ANU, Canberra, Australia, pp. 538–545.
- Chong, K. S. and Kleeman, L. 1997b. Sonar-based map building in large indoor environments. Technical Report MECSE-1997-1, Department of Electrical and Computer Systems Engineering, Monash University.
- Choset, H. and Nagatani, K. 2001. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on Robotics and Automation* 17(2):125–137.

- Davison, A. J. 1998. *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford.
- Dellaert, F., Fox, D., Burgard, W., and Thrun, S. 1999. Monte Carlo localization for mobile robots. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Detroit, MI.
- Dijkstra, E. 1959. A note on two problems in connexion with graphs. *Numerische Matematik* 1:269–271.
- Dissanayake, M. W. M. G., Newman, P., Durrant-Whyte, H. F., Clark, S., and Csorba, M. 2001. A solution to the simultaneous localization and map building (slam) problem. *IEEE Transactions on Robotics and Automation* 17(3):229–241.
- Durrant-Whyte, H. F., Majumder, S., de Battista, M., and Scheduling, S. 2001. A Bayesian algorithm for simultaneous localization and map building. *Robotics Research: The 10th International Symposium*, Victoria, Australia, R. Jarvis and A. Zelinsky, editors.
- Feder, H. J. S., Leonard, J. J., and Smith, C. M. 1999. Adaptive mobile robot navigation and mapping. *International Journal of Robotics Research* 18(7):650–668.
- Guivant, J. and Nebot, E. 2001. Optimization of the simultaneous localization and map building algorithm for real time implementation. *IEEE Transactions on Robotics and Automation* 17(3):242–257.
- Gutmann, J.-S. and Konolige, K. 1999. Incremental mapping of large cyclic environments. *International Symposium on Computational Intelligence in Robotics and Automation*, Monterey, CA, November.
- Hähnel, D., Schulz, D., and Burgard, W. 2002. Map building with mobile robots in populated environments. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, EPFL, Lausanne, Switzerland, September 30–October 4.
- Kuipers, B. J. 2000. The spatial semantic hierarchy. *Artificial Intelligence* 119:191–233.
- Leonard, J. and Feder, H. 2001. Decoupled stochastic mapping. *IEEE Journal of Ocean Engineering* 26(4):561–571.
- Leonard, J. and Newman, P. 2003. Consistent, convergent, and constant-time SLAM. *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, August 9–15.
- Leonard, J. J., Rikoski, R. J., Newman, P. M., and Bosse, M. C. 2002. Mapping partially observable features from multiple uncertain vantage points. *International Journal of Robotics Research* 21(10):943–975.
- Liu, Y. and Thrun, S. 2003. Results for outdoor-SLAM using sparse extended information filters. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, September 14–19, pp. 1227–1233.
- Lu, F. and Milios, E. 1997. Globally consistent range scan alignment for environment mapping. *Autonomous Robots* 4:333–349.
- Montemerlo, M., Thrun, S., Koller, D., and Wegbreit, B. 2002. FastSLAM: a factored solution to the simultaneous localization and mapping problem. *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada.
- Neira, J. and Tardós, J. 2001. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation* 17(6):890–897.
- Smith, R., Self, M., and Cheeseman, P. 1990. Estimating uncertain spatial relationships in robotics. *Autonomous Robot Vehicles*, I. Cox and G. Wilfong, editors, Springer-Verlag, Berlin, pp. 167–193.
- Tardós, J., Neira, J., Newman, P., and Leonard, J. 2002. Robust mapping and localization in indoor environments using sonar data. *International Journal of Robotics Research* 21(4):311–330.
- Thrun, S. 2001. A probabilistic on-line mapping algorithm for teams of mobile robots. *International Journal of Robotics Research* 20(5):335–363.
- Thrun, S., Fox, D., Burgard, W., and Dellaert, F. 2000. Robust Monte Carlo localization for mobile robots. Technical Report CMU-CS-00-125, Carnegie Mellon University.
- Thrun, S., Koller, D., Ghahramani, Z., Durrant-Whyte, H., and Ng, A. Y. 2002. Simultaneous mapping and localization with sparse extended information filters. *Proceedings of the 5th International Workshop on Algorithmic Foundations of Robotics*, Nice, France, December 15–17.
- Thrun, S., Hähnel, D., Ferguson, D., Montemerlo, M., Triebel, R., Burgard, W., Baker, C., Omohundro, Z., Thayer, S., and Whittaker, W. 2003. A system for volumetric robotic mapping of abandoned mines. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, September 14–19.
- Wang, C.-C., Thorpe, C., and Thrun, S. 2003. On-line simultaneous localization and mapping with detection and tracking of moving objects: theory and results from a ground vehicle in crowded urban areas. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Taipei, Taiwan, September 14–19.
- Williams, S., Dissanayake, G., and Durrant-Whyte, H. 2002. An efficient approach to the simultaneous localization and mapping problem. *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Washington, DC, May 11–15, pp. 406–411.