# An *Atlas* Framework for Scalable Mapping

Michael Bosse[1] Paul Newman[2] John Leonard[2] Martin Soika[3] Wendelin Feiten[3] Seth Teller[1]
[1]Laboratory of Computer Science and [2]Ocean Engineering
Massachusetts Institute of Technology
{ifni,pnewman,jleonard,seth}@mit.edu
[3]Siemens Corporate Technology, Information and Communications
{Martin.Soika,Wendelin.Feiten}@mchp.siemens.de

***Abstract*— This paper describes *Atlas*, a hybrid metrical/topological approach to 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 via Dijkstra's shortest path algorithm. Loop closing is achieved via an efficient map matching algorithm. We demonstrate the technique running in real-time in a large indoor structured environment (2.2 km path length) with multiple nested loops using laser or ultrasonic ranging sensors.**

## I. Introduction

This paper describes *Atlas*, a framework in which existing small-scale mapping algorithms can be used to achieve real-time performance in large-scale, cyclic environments. The approach 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, which we refer to as a *map-frame*, that captures the local environment and the current robot pose along with their uncertainties. The spatial extent of the map-frames is not predefined but rather determined by an intrinsic metric of the contained map. The same metric is used to invoke either a transition to an adjacent frame or the genesis of a new one.

*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, if local SLAM was based on scan-matching, then only a finite set of scans could be retained in local regions. If local processing was 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 (1) Dijsktra's shortest path algorithm [3], or (2) breadth-first search (BFS). When Dijkstra's shortest path algorithm is used, the uncertainties of the transformations of the edges of the graph serve as a statistical distance metric, with a resulting order of growth of complexity of amortized $\mathcal{O}(\log n)$ (where $n$ is the number of map-frames). Alternatively, using BFS, the number of intervening map-frames is used as the distance metric, and amortized $\mathcal{O}(1)$ computational complexity is achieved. All other components of *Atlas* have a bounded computational cost, and hence the choice of Dijsktra vs. BFS determines the overall order of growth of computational complexity of the method. In the experiment results generated for this paper, the two methods give identical performance.

Loop closing is clearly one of the most difficult issues in SLAM research. Two different types of errors can occur in loop closing, false positive matches and false negative matches. The former refers to situations where the robot erroneously asserts that a loop has been closed,

www.manaraa.com

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 very conservative loop closing strategy which attempts to avoid false positive matches, at the expense of missing some genuine loop closure events. It is possible, however, for the technique to fail in situations where the accumulated uncertainty is quite large and the environment has a highly repetitive structure. In *Atlas*, we employ a match verification procedure that has been very effective in preventing false positive loop closure matches. Nonetheless, one can envision adversely designed, maze-like environments in which any known SLAM loop closing algorithm would fail.

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 IV)..

## II. 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 simultaneous localization and mapping (SLAM) problem. A variety of approaches have been proposed for representing the uncertainty inherent to sensor data and robot motion, including topological [9], particle filter [16], [12], and feature-based [14] models. Several highly successful SLAM approaches have been developed based on the combination of laser scan matching with Bayesian state estimation [7], [16]. These methods, however, incur computational difficulties that make real-time performance impossible in 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 Guassian error models [14]. The convergence and scaling properties of the Kalman filter solution to the linear Gaussian SLAM problem are now well-known [4]. 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 [2], decoupled stochastic mapping [10], the compressed filter [6], sequential map joining [15], the constrained local submap filter [18], and sparse extended information filters [17]. Each of these methods employs a single, globally referenced coordinate frame for state estimation. The Kalman filter can fail badly, however, in situations with large angular errors and significant data association ambiguities, invalidating the Gaussian error assumption. The odometry data shown below in Figure 6(b) in Section V provides a dramatic illustration of this type of situation.

One of the appealing aspects of a hybrid metrical/topological approach to mapping and localization [1], [9] is that uncertain state estimates do not need to be referenced to a single global reference frame. This is the strategy advocated in this paper. With *Atlas*, we obtain the best of both global and local mapping approaches, by restricting the representation of errors to local regions, where linearization works well, but also providing methods by which the local submaps can be efficiently pieced together to provide global results.

An alternative to the use of local linearization would be to adopt a fully nonlinear formulation of the SLAM problem, such as FastSLAM [12] or SLAM using a sum of Gaussians model [5]. The computational requirements of these methods, however, remain poorly understood in large cyclic environments. In future research, it may be possible to implement *Atlas* using one of these techniques as the local mapping strategy.

## III. Atlas Components

We now provide a detailed description of the six core concepts of the *Atlas* framework: uncertainty projection, competing hypotheses, creation of new map-frames (genesis), closing loops (Map-Matching), instantiating and evaluating new hypotheses in adjacent map-frames (Traversal), and transformation edge refinement. These six components are now discussed.

### A. 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. There may be more than one path from one vertex to another. Since these cycles are are not constrained online, 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 Dijkstra's shortest path algorithm [3] or breadth-first search to find a unique path between the nodes. For the Dijkstra projection, we use a statistical metric, $\rho$, 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.

www.manaraa.com

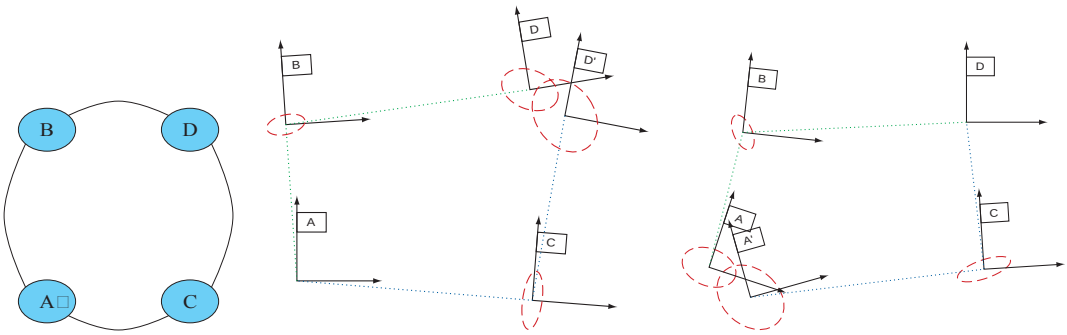


Fig. 1. The Dijkstra Projection using two different source nodes. (a) depicts the topological arrangement of the *Atlas* graph. (b) uses map-frame A as the source of the projection. (c) uses map-frame D as the source. The ellipses on the coordinate frames represent the accumulated projection error. The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

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

$$\Sigma_{ac} = \mathrm{J}_1\left(T_a^b, T_b^c\right)\Sigma_{ab}\mathrm{J}_1\left(T_a^b, T_b^c\right)^T + \mathrm{J}_2\left(T_a^b, T_b^c\right)\Sigma_{bc}\mathrm{J}_2\left(T_a^b, T_b^c\right)^T \quad (2)$$

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

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 $\rho$ computed from the compositions of transformations up the tree to the root.

### B. Competing Hypotheses

At any given time, there are several competing map-frame hypotheses that attempt to explain the current robot pose and sensor observations. There can only be one hypotheses per map-frame. We verify the validity of each map-frame's hypothesis by monitoring a performance metric $q \in [0 \rightarrow 1]$ for a few time steps.

The metric $q$ depends on the hypothesis's map-frame $\mathcal{M}_i$, robot pose estimate $\mathbf{x}_i$, and recent sensor measurements $Z$.

$$q_i = q(\mathcal{M}_i, Z, \mathbf{x}_i) \in [0 \rightarrow 1] \quad (4)$$

For example, a suitable form of $q$ for a feature-based approach is

$$q = \alpha\left(1 - \frac{\|\Sigma_{\mathbf{x}_i}\|}{\|\Sigma_{\mathbf{x}_\star}\|}\right) + (1-\alpha)\frac{\eta_a}{\|Z^k\|} \quad (5)$$

where $\eta_a$ is the number of matched observations, and $\|Z\|$ is the total number of recent observations. The parameter $\alpha \in [0 \rightarrow 1]$ reflects the relative importance placed on the robot uncertainty and successful explanation of sensor data. If in 2D, we specify maximum acceptable uncertainties in location and orientation, $\sigma_x, \sigma_y, \sigma_\theta$, then $\|\Sigma_{\mathbf{x}_\star}\|$ is the product $\sigma_x^2\sigma_y^2\sigma_\theta^2$.

At any given time, there may be several active hypotheses in an *Atlas* graph. 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 remain 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.

### C. Genesis

Since we bound the complexity (for example, the number of features) of each map, when we enter unexplored regions we need to create new local map-frames. 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 $\mathbf{x}_i$:

$$(\mathcal{M}_j, \mathbf{x}_j) = g(\mathcal{M}_i, \mathbf{x}_i). \quad (6)$$

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.

www.manaraa.com

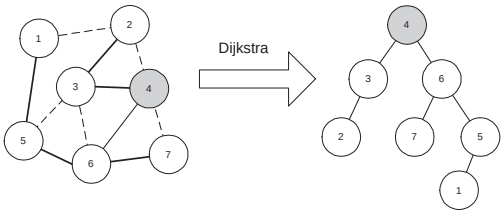


Fig. 2. The Dijkstra projection from a given node in a graph transforms the graph into a tree with the source node as a root. Here we are taking node 4 as a source. Solid lines correspond to links that are used in the tree representation. Note how in this example the uncertainty between link 4 and 7 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.

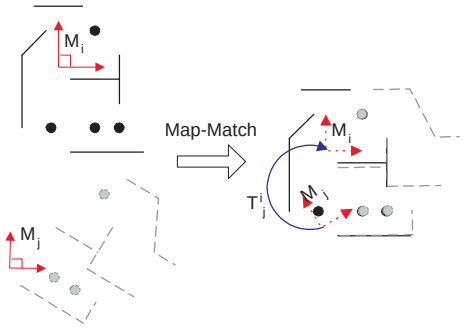


Fig. 3. Map-Matching as a search for a transformation between maps that maximally aligns common mapped features. Here two maps $i$ and $j$ share features and a good map match can be found between them. Note how only a subset of features are matched.

### D. Map-Matching

Genesis creates new maps to explain unexplored areas. However, eventually the robot will revisit an area that it has already explored. The Map-Matching process detects these situations and forms an edge in the *Atlas* graph between two unconnected vertices (map-frames).

We can describe the Map-Matching process as a search for a coordinate transformation that aligns overlapping map-frames. The Dijkstra projection is used to form the prior estimate of the transformation between two map-frames, but its uncertainty 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 we need a method that is robust to large initial errors in the transformation.

In general terms, Map-Matching is comprised of two steps:

1) A scalar $m_{ij} = m(\mathcal{M}_i, \mathcal{M}_j)$ identifies common structure between two maps $\mathcal{M}_i$ and $\mathcal{M}_j$.
2) A function $t(\mathcal{M}_i, \mathcal{M}_j)$ forms an estimate of the transformation $T_i^j$ and its covariance $\Sigma_{ij}$ between maps $\mathcal{M}_i$ and $\mathcal{M}_j$.

The exact form of $m$ used for determining common structure is not dictated by the *Atlas* framework. For the results in this paper, based on the use of feature-based SLAM within local map-frames, we define the operation $\mathcal{M}_i \cap \mathcal{M}_j$ as the search for correspondence between features in $\mathcal{M}_i$ and $\mathcal{M}_j$. We define $\eta_m$ as the minimum number of correspondences required for $m_{ij}$ to be considered a positive Map-Match.

Following a positive Map-Match between map-frames $\mathcal{M}_i$ and $\mathcal{M}_j$, we compute the estimate of the transformation $T_i^j$ between the coordinate frames of $\mathcal{M}_i$ and $\mathcal{M}_j$, as well as its uncertainty, $\Sigma_{ij}$. Note that $t$ does not depend on the robot pose. This is crucial since the navigation errors accumulated around a loop may be larger than simple data association of sensor measurements can handle.

Repetitive structure in the environment may cause ambiguous edges to be formed by Map-Matching. The degree of repetition can be assessed by Map-Matching a map with itself. Only structure elements that match uniquely within a map should be used to evaluate a match to another map.

We now describe a two stage implementation of the Map-Matching process for our feature-based example summarized as follows:

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, $\eta$, of feature pairs it brings into alignment with nearest–neighbor gating. This is the implementation of function $m$ defined above.
5) The correspondences from the best (largest $\eta$) 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 $\eta$ 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. This step is the implementation of function $t$ as defined above.

In this implementation, our maps consist of 2D point and line features. The elements used in creating a map

www.manaraa.com

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.

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 [13] could also be utilized. The freedom to choose a Map-Matching strategy highlights the modularity of the *Atlas* framework.

### E. Traversal

Once the robot has mapped an area, it can reuse previously built maps when the current frame is no longer adequate. When the robot leaves the area around which a map-frame was created, it will fail to match sensor measurements, indicating poor performance. Therefore, in a continuous attempt to find the best explanation of the sensor measurements, we run hypotheses in adjacent map-frames by projecting the current robot pose and uncertainty across the transformation edges in the local neighborhood of the current *Atlas* vertex.

The traversal of the atlas graph (making transitions between adjacent map-frames) is managed using four types of map-frame hypotheses, which we label as juvenile, mature, dominant and retired. See Figure 4 for a state transition diagram. All types, except retired ones, process sensor measurements and evaluate the same performance metric $q$. Mature hypotheses can extend their maps and spawn new hypotheses. Juvenile hypotheses can only process sensor data with regard to the existing map. Juvenile hypotheses are restricted from extending their maps because they are used to test how well a particular map explains the sensor data and not whether a new map could be built from the data.

A juvenile hypothesis can "mature" when after a probationary period its performance metric $q$ becomes greater than any other mature hypothesis. If at the end of this probationary period a juvenile hypothesis' quality does not warrant promotion it is simply deleted.

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.

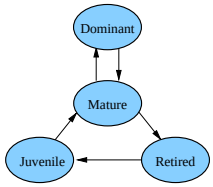


Fig. 4. *Atlas* hypothesis state transition diagram. The dominant hypothesis is used to provide an output from the algorithm. It is simply the most successful of all the mature hypotheses. Mature hypothesis are able to extend maps and spawn juvenile hypotheses in adjacent map-frames. When a mature hypothesis 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.

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 III-C). This situation will occur when the robot moves into an unexplored area.

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 5.) First we seed the hypothesis with a robot pose $\mathbf{x}_j^\star$ projected into frame $i$:

$$\mathbf{x}_i^\star = T_i^j \oplus \mathbf{x}_j \tag{7}$$

$$\Sigma_{\mathbf{x}_i}^\star = \mathrm{J}_1\left(T_i^j, \mathbf{x}_j\right) \Sigma_{ij} \mathrm{J}_1\left(T_i^j, \mathbf{x}_j\right)^T + \mathrm{J}_2\left(T_i^j, \mathbf{x}_j\right) \Sigma_{\mathbf{x}_j} \mathrm{J}_2\left(T_i^j, \mathbf{x}_j\right)^T \tag{8}$$

where $\mathrm{J}_1(\cdot,\cdot)$ and $\mathrm{J}_2(\cdot,\cdot)$ are the Jacobians of the transformation composition operators [15].

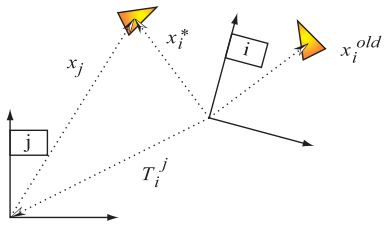


Fig. 5. Seeding the robot position for juvenile hypothesis in frame $i$ using the current pose in the adjacent map-frame $j$. The retire hypothesis attached to frame $i$ has the robot location at $\mathbf{x}_i^{old}$. The hypothesis is rejuvenated to have the robot pose of $\mathbf{x}_i^\star$.

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, $\mathbf{x}_i^\star$, are accumulated.

www.manaraa.com

This continues until we have collected enough measurements, $Z$, to solve explicitly for the robot pose independently of $\mathbf{x}_i^\star$; we call this function $w$. This approach conserves the statistical independence of map-frames.

$$(\mathcal{M}_i, \mathbf{x}_i^{\text{new}}) \quad = \quad w(\mathcal{M}_i, Z) \tag{9}$$

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.

### F. Edge Refinement

When there is more than one mature hypothesis, we can refine the estimate of the transformation between them — the *Atlas* graph edge. Consider the case of two mature hypotheses in adjacent map-frames, $\mathcal{M}_i$ and $\mathcal{M}_j$. Both maintain an estimate of the current robot pose, $\mathbf{x}_i$ and $\mathbf{x}_j$ respectively. In combination they form a measurement of the transformation $T_i^j$:

$$T_i^j \oplus \mathbf{x}_j = \mathbf{x}_i \tag{10}$$

$$T_i^j = \mathbf{x}_i \ominus \mathbf{x}_j \tag{11}$$

From Equation 11, we can write the covariance of the observation, $\Sigma_{ij}$, as a function of the robot uncertainties $\Sigma_{\mathbf{x}_i}$ and $\Sigma_{\mathbf{x}_j}$.

$$\begin{aligned} M_i &= \mathrm{J}_1(\mathbf{x}_i, \ominus\mathbf{x}_j) \\ M_j &= \mathrm{J}_2(\mathbf{x}_i, \ominus\mathbf{x}_j)\,\mathrm{J}_\ominus(\mathbf{x}_j) \\ \Sigma_{ij} &= M_i \Sigma_{\mathbf{x}_i} M_i^T + M_j \Sigma_{\mathbf{x}_j} M_j^T \end{aligned} \tag{12}$$

We update the prior estimate $T_i^{j-}$ (the existing graph edge) with the observation $T_i^j$ to form the refined estimate $T_i^{j+}$, and its uncertainty $\Sigma_{ij}^+$. Since we do not maintain the cross-covariances between robot estimates in different maps, we advocate the use of Covariance Intersection [8] to perform the update.

$$\Sigma_{ij}^+ \quad = \quad \left[\omega\left(\Sigma_{ij}\right)^{-1} + (1-\omega)\left(\Sigma_{ij}^-\right)^{-1}\right]^{-1} \tag{13}$$

$$T_i^{j+} \quad = \quad \Sigma_{ij}^+ \left[\omega\Sigma_{\mathbf{x}_i}^{-1}\mathbf{x}_i + (1-\omega)\Sigma_{\mathbf{x}_j}^{-1}\mathbf{x}_j\right] \tag{14}$$

Where

$$\omega \quad = \quad \arg\min_\omega \left\|\Sigma_{ij}^+\right\|. \tag{15}$$

If the uncertainty in local maps decreases, then with each map transition we can also improve our estimate of the transformation between frames.

## IV. Obtaining a Global Map

We are often motivated to provide a single global map of the robot's environment. For example, in Section V 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 $\nu_{i,j}$ 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).

$$\nu_{ij} = T_i^j \oplus T_j^0 \oplus T_0^i \tag{16}$$

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

$$\mathcal{T}^* = \arg\min_{\mathcal{T}} \sum_{ij} \|\nu_{ij}\|^2 \tag{17}$$

We use the Dijkstra projection to compute the initial global arrangement, and the optimization typically converges in less than 5 iterations using the Matlab optimization toolbox.

## V. Experimental Results

This section presents results for processing of data from a long-duration mission performed within and around MIT's "Infinite Corridor". The experiments utilized a standard B21 mobile robot equipped with SICK scanning laser and a ring of 24 Polaroid ultrasonic sensors. The results presented here, using eith sonar or laser sensors along with odometry, are from a real-time C++ implementation of the *Atlas* framework using an extended Kalman filter for local navigation. Onboard odometry was the only other source of information used. In related work, we have also successfully implemented the method using scan-matching as the local mapping method.

Figure 6(a) shows the topological path of the vehicle superimposed on an architectural drawing of the MIT main campus. The mission had a path length of approximately 2.2 km and a duration of 2.5 hours. The route contained nested loops of various sizes and topologies. Figure 6(b) shows the dead-reckoned path resulting from simply integrating the odometry data.

Figure 7(a) shows the result of applying the global optimization described in Section IV to the *Atlas* output

www.manaraa.com

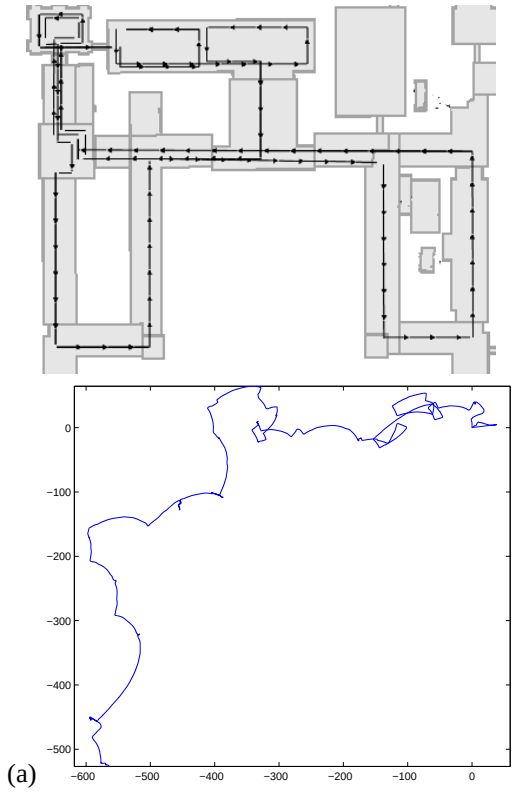


Fig. 6. (a) The manually drawn topology of the driven route overlaid on an architectural drawing of part of the MIT campus. The large east-west passage (a.k.a. the "Infinite Corridor") is about 250m long. (b) The trajectory derived from odometry alone.

based on laser scanner data and odometry. A total of 101 map-frames were built, each containing a maximum of 15 mapped line segments. Figure 8 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 stays constant. Figure 8 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 one hour into the experiment the robot returned to an area first mapped 45 minutes earlier. Similarly, after 2 hours and 15 minutes, the vehicle returned to a region mapped just five minutes after the experiment began.

Figure 7(b) shows results using data from the same experiment but using the Polaroid ultrasonic rangers instead of the laser scanner. The local navigation method used is described fully in [11]. Additional results, including concurrent processing of both laser

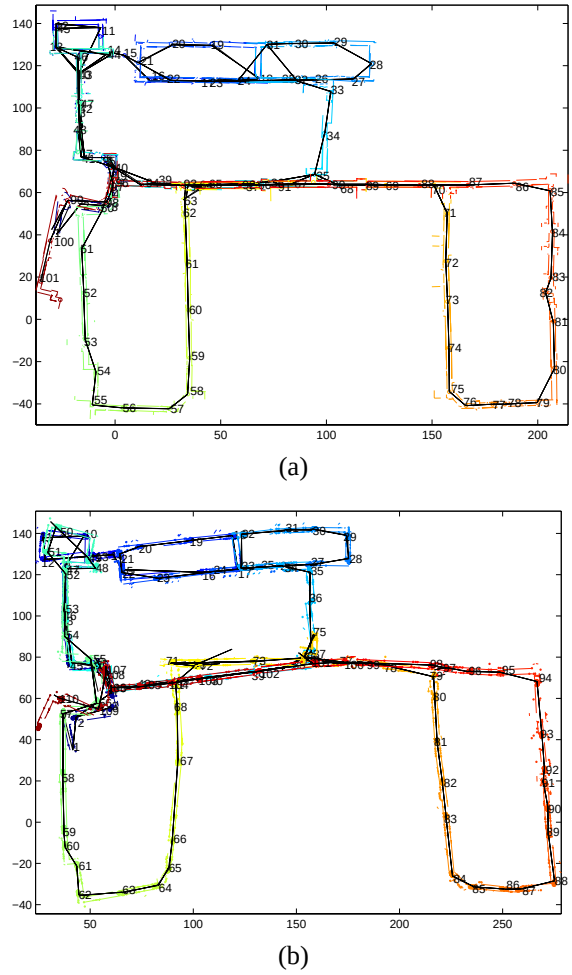


Fig. 7. Global optimized map and *Atlas* graph for processing of (a) laser and (b) sonar.

and sonar data, the use of scan-matching for local SLAM, movie files demonstrating *Atlas* processing, and the raw data for this experiment, can be accessed at: `http://graphics.lcs.mit.edu/~ifni/atlas`

## VI. Conclusion

This paper has presented *Atlas*, an 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. The method achieves a growth of complexity of either amortized $\mathcal{O}(\log n)$ when using Dijsktra's shortest path algorithm to select candidates for map-matching, or $\mathcal{O}(1)$ when breadth-first search is used for this task. 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 is

www.manaraa.com

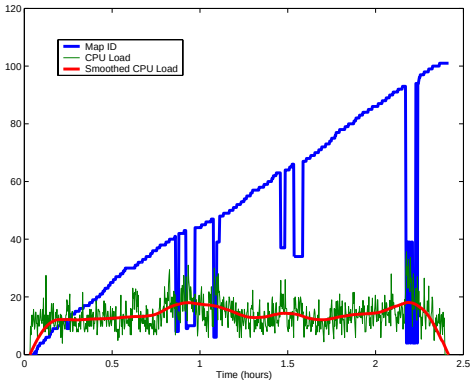


Fig. 8. Processor load and current map ID for laser data run. The general linear increase in current map ID is indicative of the mapping of new areas. The occasional "fall-back" to a map with a lower ID represents successful loop closing —the re-use of an existing map.

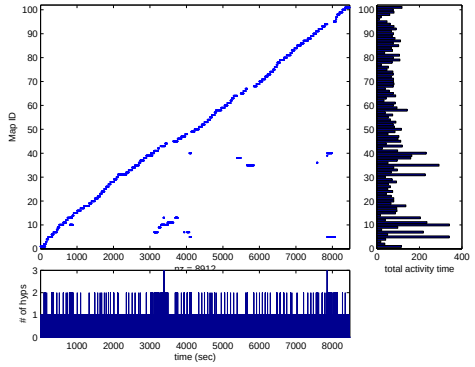


Fig. 9. Map ID vs. time, total activity vs. map ID, and the number of active hypotheses vs. time for the laser data run.

not constant-time, but operates extremely quickly (a few seconds for generation of the results in this paper).

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

## VII. REFERENCES


[1] K. Chong and L. Kleeman. Large scale sonarray mapping using multiple connected local maps. In *International Conference on Field and Service Robotics*, pages 538–545, ANU, Canberra, Australia, December 1997.

[2] A. J. Davison. *Mobile Robot Navigation Using Active Vision*. PhD thesis, University of Oxford, 1998.

[3] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Matematik*, 1:269–271, 1959.

[4] 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.

[5] H. F. Durrant-Whyte, S. Majumder, M. de Battista, and S. Scheding. A Bayesian algorithm for simultaneous localisation and map building. In R. Jarvis and A. Zelinsky, editors, *Robotics Research: The Tenth International Symposium*, Victoria, Australia, 2001.

[6] 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.

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

[8] S. J. Julier and J. K. Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the IEEE American Control Conference*, volume 4, pages 2369–2373, Albuguerque, NM, USA, June 1997.

[9] B. J. Kuipers. The spatial semantic hierarchy. *Artificial Intelligence*, 119:191–233, 2000.

[10] J. Leonard and H. Feder. Decoupled stochastic mapping. *IEEE J. Ocean Engineering*, 26(4):561–571, 2001.

[11] J. J. Leonard, R. J. Rikoski, P. M. Newman, and M. C. Bosse. Mapping partially observable features from multiple uncertain vantage points. *Int. J. Robotics Research*, October 2002.

[12] 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.

[13] 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.

[14] R. Smith, M. Self, and P. Cheeseman. Estimating uncertain spatial relationships in robotics. In I. Cox and G. Wilfong, editors, *Autonomous Robot Vehicles*, pages 167–193. Springer-Verlag, 1990.

[15] 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*, 2002.

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

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

[18] 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, 2002.


TABLE I

ATLAS PARAMETER SETTINGS FOR THE PRESENTED RESULTS.

| Description | Parameter | Value |
|---|---|---|
| minimum probationary period for a juvenile hypothesis | $\tau_j$ | 3 sec |
| maximum number of active hypotheses | $\mathcal{H}_m$ | 5 |
| maximum number of features in map | | 15 |
| maximum local uncertainty in robot | $\sigma_x^\star, \sigma_y^\star$<br>$\sigma_\theta^\star$ | 0.2 m<br>2 deg |
| minimum number of matched features | $\eta_m$ | 4 |


www.manaraa.com