Autonomous Feature-Based Exploration

P. Newman, M. Bosse and J. Leonard
Massachusetts Institute of Technology

Abstract— **This paper presents an algorithm for feature-based exploration of *a priori* unknown environments. We aim to build a robot that, unsupervised, plans its motion such that it continually increases both the spatial extent and detail of its world model - its map. We present a method by which the planned motion at any instant is motivated by the geometric, spatial and stochastic characteristics of the current map. In particular each feature within the map is responsible for determining nearby unexplored areas that if visited are likely to constitute exploration. We assume that the location of the features is uncertain and represented by a set of probability distribution functions (pdfs). These distributions are used in conjunction with the robot path history to determine a robot trajectory suited to exploration. We show results that demonstrate the algorithm providing real-time exploration of a mobile robot in an unknown environment.**

I. Motivation

This paper describes a method that enables a robot using feature-based navigation techniques to autonomously explore its environment. Recently, there has been much written about feature based Simultaneous Localization and Mapping (SLAM) algorithms [1], [2], [3], [4], [5]. Such algorithms attempt to answer the question"where am I and what is around me ?". That answered the next obvious question is "where should I go next?". The exploration algorithm presented in this paper provides a sensible answer to this question.

In [4] an indoor mobile robot was manually directed around a populated indoor environment while performing SLAM. Upon instruction it then autonomously returned to its starting position to with only a few cm of error. The work presented in this paper removes the need for manually guided exploration producing truly autonomous operation.

We address the explore problem not from an idealized theoretic perspective [6] but rather from within the context of real-time navigation with noisy sensors in an unknown and uncertain environment.

P. Newman and J. Leonard are at the Department of Ocean Engineering at MIT, M. Bosse is at the Laboratory of Computer Science MIT. email: {pnewman,ifni,jleonard}@mit.edu

The underlying philosophy we adopt is that the observed features themselves should inspire the exploration process. Collectively the features form a map of the known environment. Intuitively, examining an existing map should inspire a plan to expand its coverage and refine its detail. Walls, for example, (line features) tend to collate into long connected paths through the environment, defining free space boundaries. It makes sense then for mapped lines to entice excursions to their end points, where mapping of a new adjacent segment is most likely to occur. Similarly we would benefit from having point-like targets (door frames and corners) advocate driving an arc around them.

The benefits of our feature-based approach to exploration can be enumerated as follows:

- The goal of exploration is to discover and enable mapping of a novel area. It is predicated on good navigation. Good navigation is itself reliant upon trustworthy observations of the environment. In feature-based SLAM this requires remaining in sight of real-world features. This can be achieved if exploration paths are dictated by mapped features.
- If the focus of exploration is derived from the location and geometry of features they can serve to vector the robot efficiently to places of interest outside the immediate vicinity of the robot.
- If only nearby features are considered then the computational complexity and memory requirements of the algorithm are constant and independent of map size and mission length. If however, all mapped features are considered then the complexity is still only proportional to the number of mapped features.
- Feature-based exploration is independent of the kind of sensing employed. Any suitable proprioceptive sensor data can be fused to form a feature-based representation. The evidence for the existence of a feature can be accrued over multiple time steps and robot locations [5].

Several successful contemporary techniques [7] [8] take an opposite approach and use free space analysis such as Voronoi diagrams to decide where to go next and model free-space. This however is counter to the

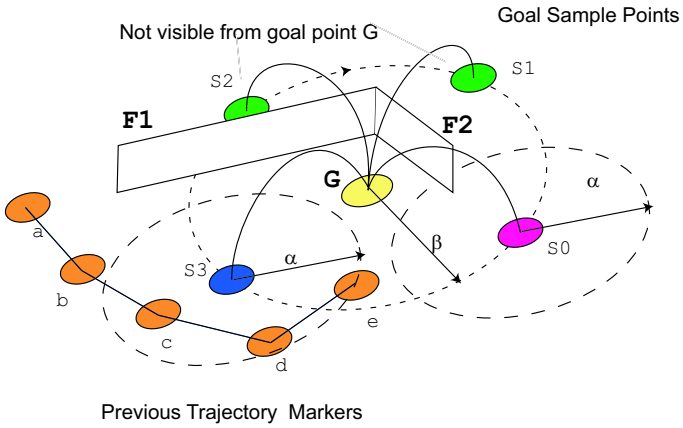

Fig. 1. A visualization of the explore algorithm.

needs of sensor-based navigation techniques which depend on proximity (visibility) of features to produce useful output. Other techniques use evidence grids [9] to motivate exploration. For example in [10] a "frontier boundary" between explored and unexplored grid cells is defined. It is well known that grid-based approaches do not lend themselves to good, long-term CML/SLAM navigation and mapping performance. Cell-based techniques also suffer from grid to world alignment complications. Furthermore the memory costs of exploration using grid based approaches scales poorly with time. Bauer and Rencken [11] use feature based mapping but retain a grid-based exploration strategy. Exploration is more motivated by the trajectory history of the robot (a visitation count in each cell) than by it's perception of its environment.

II. The Exploration Algorithm

We want to produce a means by which an autonomous robot in possession of a feature-based representation of its surroundings can decide where to move so as to best explore it environment. This is an action selection problem. There is a large literature on action literature [12], [13], [14] but little written from the perspective of map building in the presence of uncertainty - the issue this paper addresses.

We assume the vehicle is equipped with a SLAM algorithm that is capable of consistent mapping and localization. No restrictions whatsoever are placed on the form of this algorithm. It could use a single large map or multiple small maps. The uncertainties in map and vehicle estimates could be manipulated with EKF or particle filter techniques. All that is required is an output stream of feature and vehicle descriptions (type, location, orientation and uncertainty).

For the purposes of this paper we need to define what constitutes exploration. We propose the following properties:

(a) Try to visit areas that are 'open' and sparsely populated with features.
(b) Try to stay away from areas that have already been visited.
(c) Preferentially explore locally – try to visit areas that are close to and reachable from the current position.
(d) Only when the local area appears to be explored consider traversing to a distant 'un-explored region'. An good example of this behavior is backing out of a dead-end corridor. We call this *global traversal*.
(e) Be able to preempt a global traversal when the local area appears to be substantially more interesting. For example, if a door that had previously been closed was now open, we would like to investigate the newly exposed room.
(f) Try to explore with increasing resolution. Initially try to determine the broad characteristics and shape of the environment. Then revisit areas to obtain a finer grain model of the world.

The action selection is performed by evaluating the utility of visiting any one of a set of locations. Each possible location is generated by a parent feature and shall be referred to from now on as as a *goal*. Our policy is simple – having selected the most promising goal steer towards it. If no direct line of sight path exists then plan one using a "free-space highway" that is built incrementally during exploration. Of course our exploration plan may have to be influenced by obstacle avoidance tactics but this does not impinge on the action selection process.

III. Action Selection

This section details the goal generation and evaluation steps involved in selecting the most profitable (in terms of explorations) location to visit. Figure 1 provides a visualization of the following algorithm.

A. Goal Generation

The n^{th} feature f_n is defined by a location vector $\mathbf{f_n}$. Each feature generates a set of m goals $\mathbf{g^n_{1:m}}$ with respect to its own local coordinate frame n. These goals may be expressed in the map frame by composition with the features location vector $\mathbf{f_n}$.

$$\mathbf{g}^n_{1:m} = \mathbf{f}_n \oplus \mathbf{g}^n_{1:m}$$

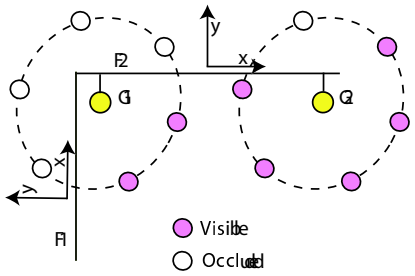

Fig. 2. Sampling the sparsity and openness of line segments

In this way the course vector $\mathbf{u}$ to any goal i from the current robot pose $\mathbf{x_r}$ is simply

$$\mathbf{u} = \ominus \mathbf{x}_r \oplus \mathbf{g}_i^n$$

where $\ominus$ is the inverse composition operator. The generation of the goals around the feature is a function of both feature type and geometry. It should produce goals that if selected would fulfill or at least contribute to the exploration qualities we defined in section II.

B. Goal Evaluation

For each generated goal $\mathbf{g}_{1:m}^n$, a set $\mathbf{S}$ of s_0 sample points are regularly distributed around it at radius β. The aim is to assign each goal a score $\eta \in [0 \rightarrow 1]$ which is an indication of the utility in terms of exploration of visiting it. A score of one implies a strong return while zero infers no expected gain in visiting the goal.

We shall evaluate η by operating on the set of sample points $\mathbf{S}$. We sequentially remove elements from $\mathbf{S}$ during a two-part selection process. At termination η is calculated as the ratio of the final size of $\mathbf{S}$ to its initial size s_0. Thus the fewer removals (or larger the size of $\mathbf{S}$) the more utility we assign to the goal in question. The motivation for and details of each stage will now be discussed.

• **Step 1 : Sparsity Evaluation** Using a method that will be discussed in III-C, each sample point is tested for a clear path between itself and the goal performing the sampling.
Sample points which have no clear path back to the goal are removed from $\mathbf{S}$. For example in an area richly populated by mapped features, sample points are more likely to be occluded by features and hence removed from $\mathbf{S}$. Conversely, goals generated by features in sparse areas are likely to have more unoccluded samples and preserve the size of $\mathbf{S}$. This behavior then fits with the first of our definitions of exploration: investigation of sparse regions. The sample points enable a local measure of feature sparsity around the goal location. Recall that the actual location of the goal is determined by the feature's goal generation function. In this way different feature geometries can cause sparsity or 'openness' to be evaluated at relevant locations. Take for example the case of a line segment shown in Figure III-B. By generating goals near its end points we can use the size of $\mathbf{S}$ to measure the openness of the line at both ends.

• **Step 2 : Novelty Evaluation** For each of the remaining samples in $\mathbf{S}$, a list of nearby trajectory markers is built. We define 'near' to be within a distance α of that sample. Typically we set $\alpha = \frac{3\beta}{2}$. Once again the visibility criterion is invoked – this time between the sample point and each of the nearby trajectory markers. If any of the tested trajectory markers are visible, then the current sample is removed from $\mathbf{S}$

• **Scoring** A score η ranging from 0 to 1 is assigned to the goal. It is calculated as the ratio of the final and initial size of $\mathbf{S}$. The final size of $\mathbf{S}$ is the number of samples points for the considered goal that:
– a) have a clear line of sight to the goal
– b) have no line of sight to any nearby trajectory marker

C. Visibility Evaluation

The action selection process requires the determination of the existence of a clear path between different entities. If as is the case with feature-based CML techniques each feature has an associated estimated uncertainty in location and orientation then it is desirable to use this information in deciding if a clear path exists.

We define a visibility function $\mathbf{V}$ which evaluates to a non-zero value $\mathbf{v_{i,j}}$ if a clear path of width r exists between any two entities $\mathbf{e}_i$ and $\mathbf{e}_j$ given all other entities. In this paper, without prejudice, we limit the entities $\mathbf{e}_i$ and $\mathbf{e}_j$ to be lines or points but in general more complex objects can be used.

$$\mathbf{v_{i,j}} = \mathbf{V}(\mathbf{e}_i, \mathbf{e}_j, r) \tag{1}$$

Figure 3 illustrates the action of $\mathbf{V}$. The figure shows the pdfs of two orthogonal line features and two point-like entities - for example current vehicle location and a point on the vehicle's past trajectory.

We can define the 'probable boundary' of the i^{th} feature by the locus of the n^{th} sigma bound $\mathcal{L}_{n\sigma}^i$ of its pdf and truncate the distribution along this boundary. Typically n is set to 2 or 3.

The evaluation of the visibility function $\mathbf{V}$ now requires finding intersections of the hatched region in

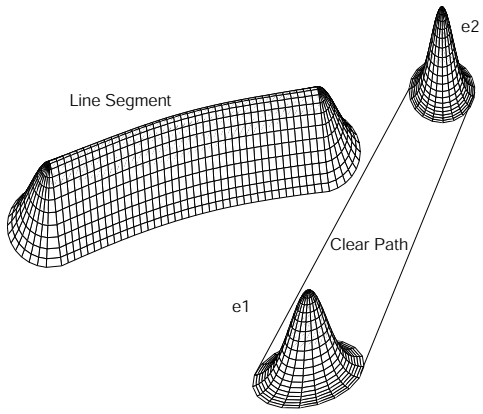

Fig. 3. Visibility testing between entities

Figure 3 joining the n-sigma bounds of $e1$ and $e2$ and the regions enclosed by $\mathcal{L}^i_{n\sigma}$.

The form of $\mathcal{L}^i_{n\sigma}$ is a function of the kind entity under consideration. That of a point-like feature is simply a scaled version of the 1σ covariance ellipse.

The pdf of a line segment is more interesting. The parabolic shape of the line pdf in Figure 3 stems from the combination of lateral and angular uncertainty in the line location. The covariance of a line is given by

$$\mathbf{P} = \begin{bmatrix} p_{yy} & p_{y\theta} \\ p_{\theta y} & p_{\theta\theta} \end{bmatrix}$$

where p_{yy} is the lateral variance, $p_{\theta\theta}$ is the angular variance and $p_{y\theta}$ is the covariance between angular and lateral uncertainties with respect to the center of the line. Transforming this uncertainty to a reference frame at a distance ρ along the line results in

$$\mathbf{P}' = \mathbf{J}_\rho \, \mathbf{P} \mathbf{J}_\rho^{\mathbf{T}}$$

where $\mathbf{J}_\rho$ is the Jacobian of the translation along the line. This allows the lateral uncertainty p'_{yy} to be expressed as a function of distance along the line.

$$p'_{yy}(l) = l^2 + 2\,l\,p_{y\theta} + p_{yy}$$

The locus of $\sqrt{p}'_{yy}$ can be used to describe the probable boundary of a line feature along its length. The ends of the pdf are formed by a symmetric normal distributions with $\sigma = p'_{yy}(\pm L/2)$ where L is the total length of the line.

Depending on the computational resources available and the form of the pdf boundaries it may be necessary to reduce the complexity of evaluating V by approximating the probable feature boundaries with a set of line segments. For example in the results we present a scaled convex hull was used to represent line pdfs. The determination of a clear path is then simply a matter of finding line intersections.

D. Contexts and Free Space

An explore algorithm needs to operate in both local and global contexts. When operating locally decisions are made about what visible and nearby features should be visited. For example when traversing down a previously unvisited corridor there may be a local focus at the end of line segments describing the corridor walls. The global mode is required to determine interesting regions distant from the current robot location. For example a corridor that was being explored under local control is now determined to be a dead-end – the local area is now explored and a new area, perhaps far away, must be found that is ripe for exploration.

Upon determination of a new global focus the vehicle needs to be able to plan and execute a path to it. The path is built from a collection of free space markers that is constructed as the robot moves about its environment. The location of the vehicle marks a region of free space. If we keep track of past locations we create a trajectory history which defines a set of reachable points. We refer to these points as free space markers. When a free space marker is created (almost always at the current robot location) its adjacency to other existing markers is calculated. Marker adjacency is synonymous with co-visibility – if there is an uninterrupted line of sight between two markers they are considered adjacent. As the robot moves through the environment we create a graph in which nodes of markers locations and edges of adjacencies (Figure 5). This graph can be conceived of as a highway of known free space [15] or a skeletal representation of reachable locations [16][17].

Free space markers are only created when no other visible markers are within a given distance of the current robot location. Typically a threshold distance of 1m is chosen. Choosing too large a distance is more likely to cause nodes in the graph with no adjacencies – ie nodes that cannot be directly reached from any other node. Figure 6 for example shows a pathological case of the vehicle turning swiftly round a corner. The markers dropped are not mutually visible and a route cannot be found to marker $k+1$ through the obvious choice of marker k. In some ways this reflects a desirable property in that routes are not planned through mapped features. On the other hand it can create an orphan region of the work domain – the graph becomes disconnected. One obvious way to overcome this problem is to drop markers regularly so as to minimize the chances of this happening. A more robust technique would involve keeping a short-

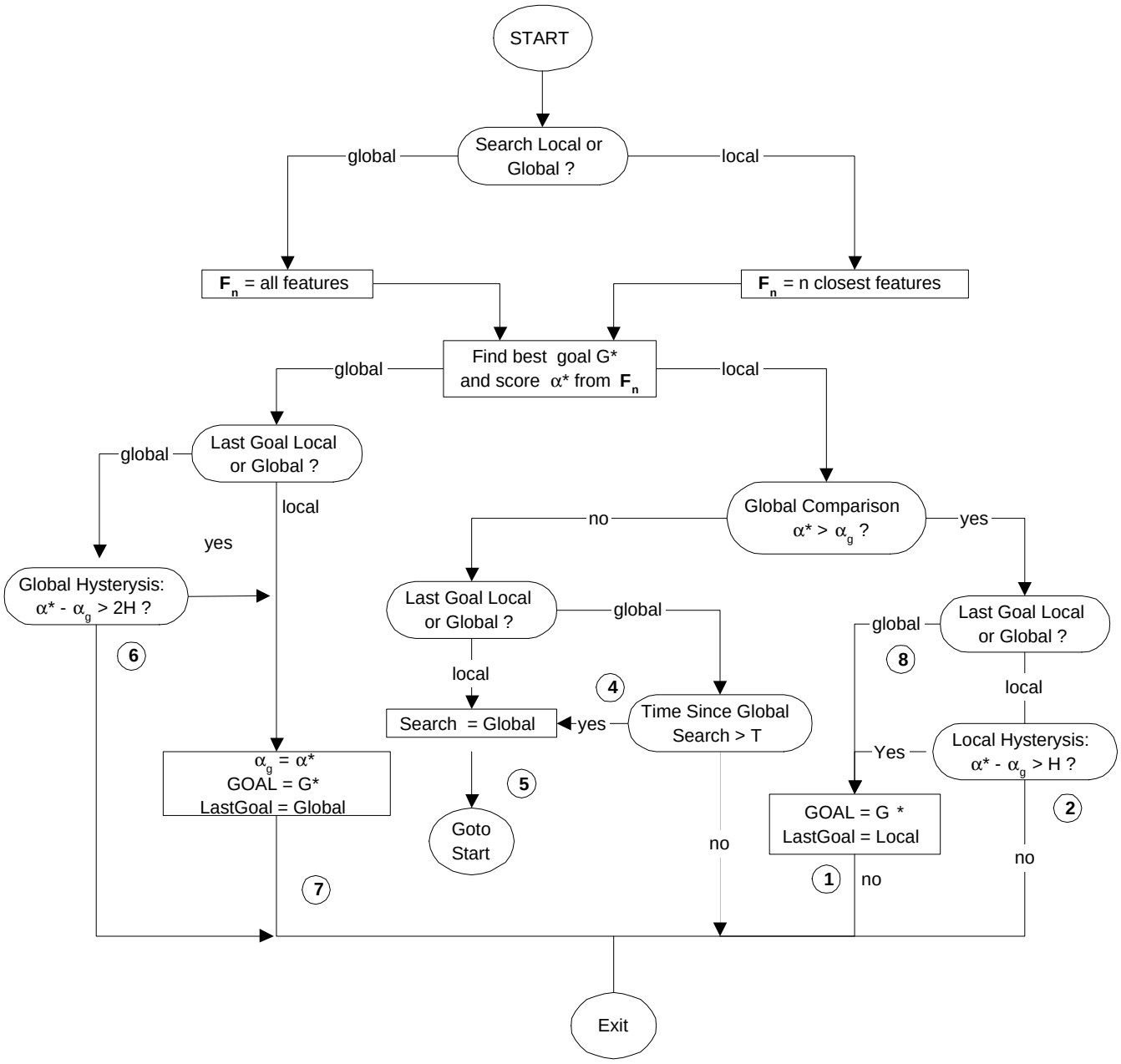

Fig. 4. The Exploration Algorithm. Points of interest are labelled with numbers and discussed in Section III-E. Note that one iteration may take two passes through the flow chart as the search for goals transitions from local to global contexts.

term memory of the recent trajectory of the the vehicle. When a marker is created for which no adjacencies can be found, an additional marker is added half-way along the trajectory between the last good marker and the current vehicle location. If this marker is adjacent to both the previous and newly created ones then the graph remains connected. If however the graph remains disconnected the process continues – bisecting the vehicle trajectory, adding another marker and calculating adjacencies – until graph connectivity is restored. This process is guaranteed to succeed given that markers lie on the path that robot itself executed. When a region to explore has been determined via global search it is unlikely that it is directly visible from the robot's current location. In this case the free-space graph is used to find a route to the desired location. The vehicle steers to the closest marker and then uses a shortest path algorithm to find the shortest route to a marker closest to the desired point. The robot controls itself from marker to marker until the last one is reached at which point in leaves the free space highway and attempts to steer

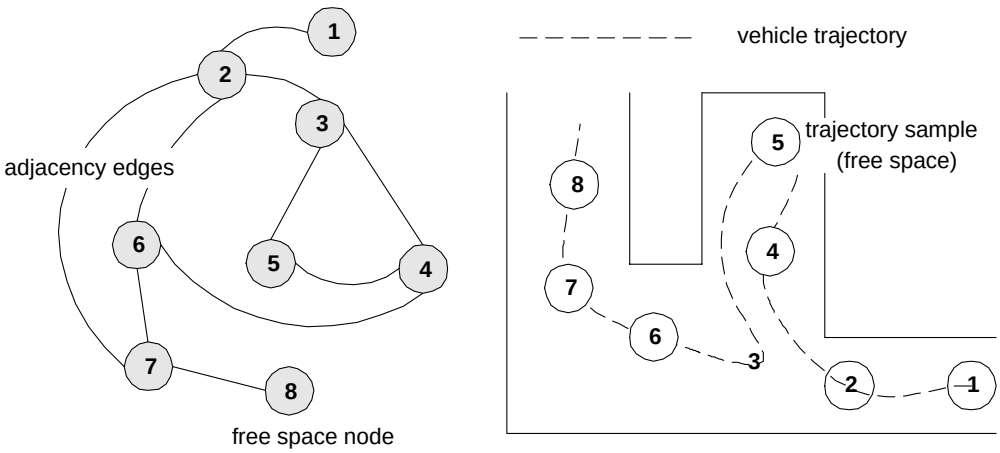

Fig. 5. Graphing free-space adjacency

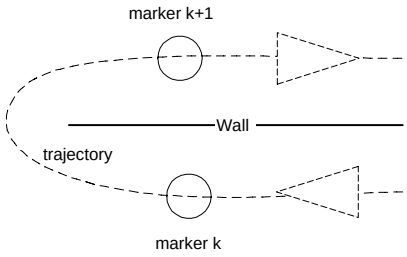

Fig. 6. Pathological case producing a disconnected graph

to the desired destination. The fact that a CML algorithm instantiates features from observations taken from current robot locations (or multiple robot locations [18]) means that all features must be observable by moving through and between past locations. Furthermore as we advocate a feature-based exploration in which the the focus of exploration is close to and derived from mapped features, the robot will be able to observe any location dictated by the exploration algorithm.

E. Integration

Figure 4 is a flow chart of the entire explore algorithm. It is designed to allow direct translation into an implementation. At every iteration the algorithm flows from *Start* to *Exit*. The flow of the algorithm is regulated by the state variables and constants described in table III-E. The *Start* values correspond to the assignment of named variables at the start of the entire algorithm. If the variable has a 'yes' in the *Reset* column then at every iteration of the algorithm it is reset to this value otherwise it remains unchanged. The significance of some parts of the diagram warrant discussion. These regions have been numerically labelled and are now discussed.

1 `GOAL` is set to a new, locally found goal.
2 The best local goal is not substantially better than the already selected goal `GOAL` which was also found locally. This hysteresis prevents unnecessary swapping between goals that are by definition in the same area.
3 While on a transit to a globally determined goal (`LastGoal`=*global*) a more interesting local goal was found. The transit to the global goal is preempted with this new focus of exploration.
4 The last goal was found globally, no local goal looks any better but T seconds have passed and the global goal is re-evaluated to ensure that it is still the most interesting place to visit.
5 No local goal was found that was more interesting than the overall exploration index α_g. The process begins again but now sets `Search` to *global*. The algorithm will next flow down the left hand side of the chart.
6 A new global goal was found but it was not substantially better than the one already selected and in transit to. Note that the hysteresis implies a cost of switching global goals equal to twice that of switching local ones. This reflects the fact that globally found goals are likely to be substantially distance and take an appreciable time to transit to. It would be better to finish exploring the current focus region even though it is not the absolute favorite and then move on.
7 A new goal is set to a globally found one. This always happens if a global search has been undertaken

Name	Range	Start	Reset	Meaning
α_g	0:1	1	no	the overall exploration index
`GOAL`	location	NULL	no	the best current exploration goal
`LastGoal`	*global/local*	*local*	no	mode in which last goal was set
`Search`	*global/local*	*local*	yes	current search mode
H	0.1 (constant)	0.1	no	hysteresis for changes in `GOAL`
T	any	4	no	time between forced re-appraisal of a global goal
n	any	5	no	The number of features considered in a goal search

TABLE I
TABLE OF PARAMETERS

and the last found goal was found locally, ie we are on the second pass of the algorithm as described by label [5]. This is the only place that α_g is set because it is the only place that a global appraisal of the global state of exploration has occurred. Together α_g and `GOAL` represent the score and location of the most promising areas for exploration in the mapped environment.

F. Obstacle Avoidance and Local Path Planning

Obstacle avoidance is an important component of autonomy. Lack of space permits only a broad explanation of the techniques we employed in this work. The motion commands are issued in terms as desired poses (x,y,θ) in odometry coordinates. A path of polynomial curvature is found that leads to the desired goal pose subject to constraints of minimum path length and initial radius of curvature being equal to that currently being driven. The platform's angular and linear velocities are controlled to achieve the desired instantaneous radius or curvature. Obstacle avoidance is achieved by selecting a radius of curvature as close to the desired one as possible but which would not cause a collision with a detected obstacle within the near future (in our case 2 seconds).

IV. Results

The explore algorithm described in this paper has been implemented in C++ and integrated with the MIT Department of Ocean Engineering's "MOOS" vehicle software suite.

The exploration algorithm was run live on a B21 at the AAAI Robotics Challenge in Alberta, Canada. The combination of exploration, SLAM, and obstacle avoidance enabled the vehicle to explore and build a map of the conference area during a coffee break between conference sessions. However the path taken to avoid the multitude of coffee drinkers obfuscated the underlying actions of the exploration algorithm. As an alternative, the results we present are taken from an autonomous explore session running inside the corridors of MIT. The explore module is unaware of the source of feature descriptions – simply receiving them over a network connection. To date the explore module has been run successfully with both the "CMLKernel" [4] and the Atlas framework [19] CML frameworks. The results presented here were generated using a B21 mobile robot employing SICK laser scanner and using the Atlas framework.

Figure 7 shows a plan view of the explore algorithm in progress. The potential exploration goals are marked with squares (locally visible) and circles (globally found goals). The exploration score attributed to each goal the last time it was evaluated is written above the relevant symbol. The currently selected goal is filled with a solid triangle. Note how the 0.86 goal is not substantially better than the nearby and locally visible 0.71 goal. The preference for local exploration wins out here and the robot heads in the direction indicated by the solid line. Figure 8 shows the vehicle returning from a completed exploratory excursion into an area filled with lockers and carrying on up a previously seen corridor. The tessellating trajectory markers being dropped as the vehicle moves are shown as hexagons. Figure IV depicts the situation some time later. The explore algorithm is directing the vehicle up a corridor. The end of the corridor has been reliably observed but the walls only partially so, creating intriguing openings at the corridor's end. A few moments later,see Figure 10, the side walls have been mapped and the corridor has become closed off before the robot has reached its previously intended

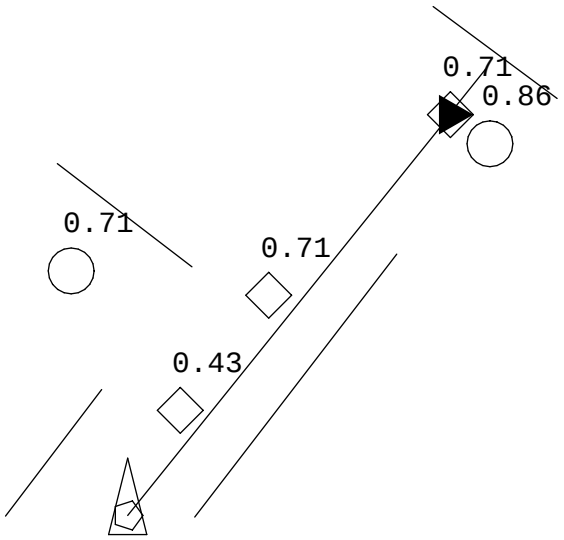

Fig. 7. The opening stages of the exploration. From its starting location the vehicle maps four walls. Immediately the north-east wall is evaluated as the most open and visible - a path is plotted towards it.

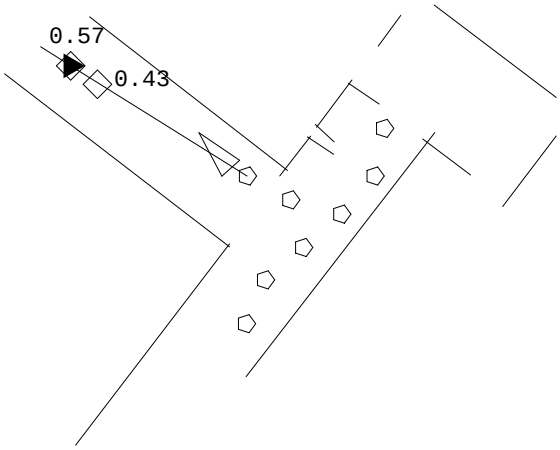

Fig. 8. The initial are of interest (a set of lockers) has been mapped. With the local area explored, a path is chosen that takes the vehicle down a corridor that was initially passed over for exploration.

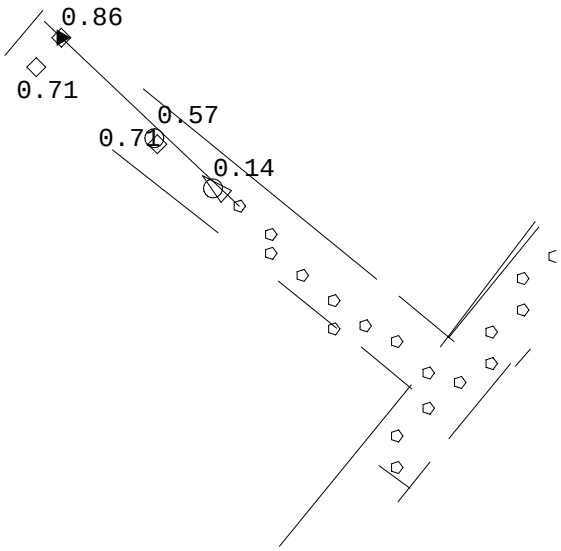

Fig. 9. Fifteen minutes later the robot is exploring a long corridor that appears to be open at its ends.

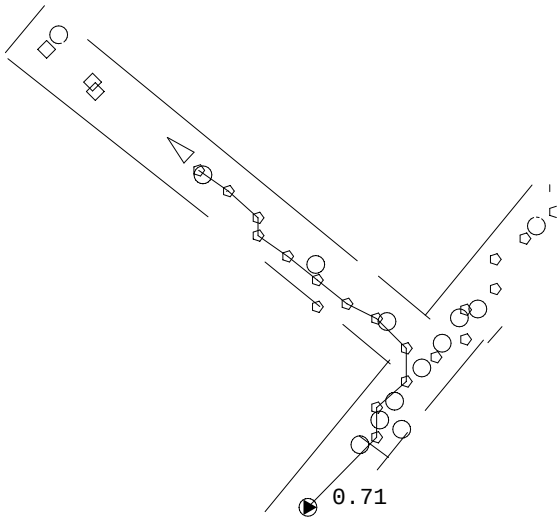

Fig. 10. A little further up the corridor more of the walls have been mapped. The local environment now appears to be closed off and the road map of free-space is used to plan a route to new "promising" area for exploration in the south-west corner.

goal. At this point the local area is well explored and a new exploration goal has been found back down the corridor and to the left. The free-space graph has found a route to the desired goal and is shown as a sequence of lines connecting some of the free space markers in the figure. This process of feature-inspired exploration was allowed to continue for half an hour until the vehicle completed a loop and returned to its starting area. Figure 11 is a 3D-extruded view of the map built by the CML algorithm under the guidance of the exploration. Throughout the session the low level obstacle avoidance task produced local path planning and obstacle avoidance.

V. Extensions and Further Work

The results presented are for a medium-scale indoor environment. Extension to truly large scale problems covering several kilometers is the subject of current research. We intend to couple the explore algorithm to the multiple-map "Atlas" frame work [19] using multiple local maps. Another useful property of an explo-

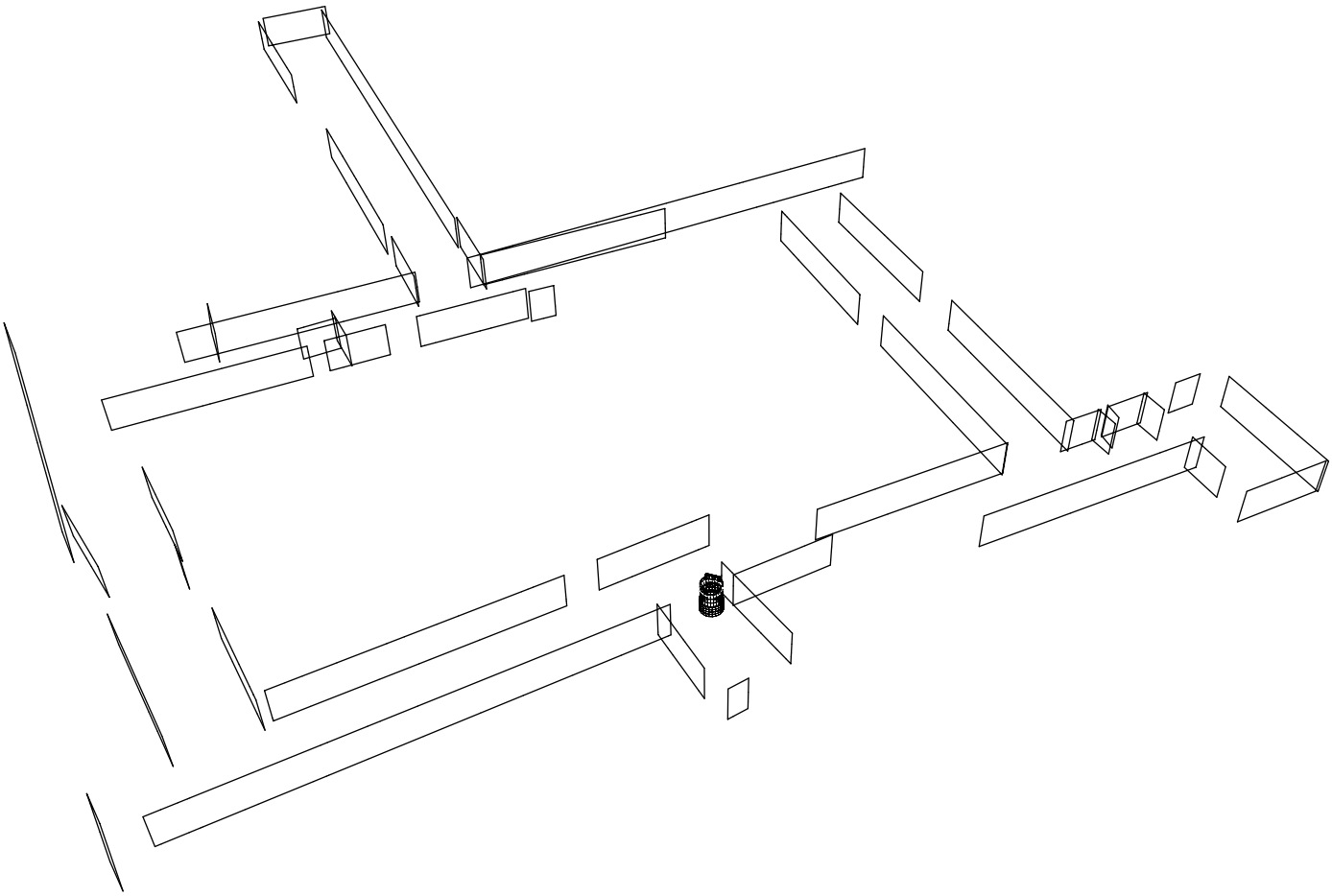

Fig. 11. A 3D extruded view of the map built under the direction of the explore algorithm. The robot is shown nearing its start point having circum-navigated a loop of a building. Each side of the loop is around 25m long

ration algorithm that has not been discussed here is that of planned motion and perception to aid not just mapping but localization as well. With analysis of the correlations between vehicle and locally observed features and the uncertainty of the vehicle pose estimate it should be possible to mitigate some of the conditions that lead to localization failure. For example, after executing a sharp turn around a corner in a corridor it is advantageous to briefly turn back and observe the corridor walls to better estimate the new vehicle heading. Finally, at the time of writing the algorithm does not take the path length to goal into account when deciding where to drive the vehicle. This simple addition will prevent the robot undertaking long traversal to visit a point only marginally more interesting than that one much closer to its current location.

In conclusion, the algorithm we have presented produces reliable and sensible autonomous exploration of unknown environments. The approach uses a feature-based world representation to deduce robot trajectories that are likely to expand and refine the map of the robot's environment. We have demonstrated truly autonomous exploration and mapping running real-time in an everyday indoor environment.

References

[1] M. W. M. G. Dissanayake, P. Newman, H. F. Durrant-Whyte, S. Clark, and M. Csorba, "An experimental and theoretical investigation into simultaneous localization and map building," in *Sixth International Symposium on Experimental Robotics*, March 1999, pp. 265–274.

[2] J. Guivant and E. Nebot, "Optimization of the simultaneous localization and map building algorithm for real time implementation," *IEEE Transactions on Robotic and Automation*, vol. 17, no. 3, pp. 242–257, June 2001.

[3] A. J. Davison, *Mobile Robot Navigation Using Active Vision*, Ph.D. thesis, University of Oxford, 1998.

[4] P. Newman, J.Leonard, J.D. Tardós, and J.Neira, "Explore and return: Experimental validation of real time concurrent mapping and localization," in *Proc. IEEE Int. Conf. Robotics and Automation*. IEEE, 2002, pp. 1802 –1809.

[5] R. Rikoski, J. Leonard, and P. Newman, "Stochastic mapping frameworks," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2002, pp. 426– 433.

[6] J-C. Latombe, *Robot Motion Planning*, Boston: Kluwer Academic Publishers, 1991.

[7] H. Choset and K. Nagatani, "Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization," *ieeetra*, vol. 17, no. 2, pp. 125–137, 2001.

[8] A. Victorino, A. Rives, and J. Borrelly, "Mobile robot navigation using a sensor-based control strategy," in *Proc. IEEE Int. Conf. Robotics and Automation*, 2001, pp. 2753–2758.

[9] H. Moravec, "Sensor fusion in certainty grids for mobile robots," in *Sensor Devices and Systems for Robotics*, pp. 253–276. Springer-Verlag, 1989, Nato ASI Series.

[10] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proc. Int. Symp. on Computational Intelligence in Robotics and Automation*, 1997, pp. 146– 151.

[11] R. Bauer and W.D Rencken, "Sonar feature based exploration," in *Proc. IEEE Int. Workshop on Intelligent Robots and Systems*, 1995, vol. 1, pp. 148–152.

[12] Paolo Pirjanian, *Multiple Objective Action Selection & Behavior Fusion*, Ph.D. thesis, Aalborg University, 1998.

[13] Julio K. Rosenblatt, *DAMN: A Distributed Architecture for Mobile Navigation*, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, 1997.

[14] Michael R. Benjamin, *Interval Programming: A Multi-Objective Optimization Model for Autonmous Vehicle Control*, Ph.D. thesis, Brown University, Providence, RI, 2002.

[15] J. Hsu and L. Hwang, "A graph-based exploration strategy of indoor environments by an autonomous mobile robot," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1998, pp. 1262– 1268.

[16] B.Mirtich and J. Canny, "Using skeletons for nonholonomic path planning among obstacles," in *Proc. IEEE Int. Conf. Robotics and Automation*, 1992.

[17] J.F. Canny, *The complexity of Robot Motion Planning*, MIT Press, Cambridge,MA,USA, 1988.

[18] J. J. Leonard and R. Rikoski, "Incorporation of delayed decision making into stochastic mapping," in *Experimental Robotics VII*, D. Rus and S. Singh, Eds., Lecture Notes in Control and Information Sciences. Springer-Verlag, 2001.

[19] M. Bosse, P. Newman, and J. Leonard, "An atlas framework for scalable mapping," Submitted for publication for ICRA 2003, 2002.