In Fourth International Workshop on Algorithmic Foundations of Robotics (WAFR 2000), Hanover, NH

Reliable Mobile Robot Navigation From Unreliable Visual Cues

Amy J. Briggs, *Middlebury College, Middlebury, VT, USA*
Daniel Scharstein, *Middlebury College, Middlebury, VT, USA*
Stephen D. Abbott, *Middlebury College, Middlebury, VT, USA*

Vision-based mobile robot navigation requires robust methods for planning and executing tasks due to the unreliability of visual information. In this paper we propose a new method for reliable vision-based navigation in an unmodeled dynamic environment. Artificial landmarks are used as visual cues for navigation. Our system builds a visibility graph among landmark locations during an exploration phase and then uses that graph for navigation. To deal with temporary occlusion of landmarks, long-term changes in the environment, and inherent uncertainties in the landmark detection process, we use a probabilistic model of landmark visibility. Based on the history of previous observations made, each visibility edge in the graph is annotated with an estimated probability of landmark detection. To solve a navigation task, our algorithm computes the expected *shortest paths between all landmarks and the specified goal, by solving a special instance of a Markov decision process. The paper presents both the probabilistic expected shortest path planner and the landmark design and detection algorithm, which finds landmark patterns under general affine transformations in real-time.*

1 Introduction

Vision-based mobile robot navigation is often planned using landmarks, either artificial or extracted from the environment. Such landmarks must be detected quickly and as reliably as possible. When landmark detection is unreliable due to factors such as temporary occlusion or varying lighting conditions, the planner should compute motion paths dynamically to always find the current best path. This paper makes two major contributions: (1) a navigation system that uses probability estimates of landmark visibility in order to compute expected shortest paths between landmarks; (2) a novel landmark pattern together with a real-time algorithm for landmark detection that can handle a wide range of affine transformations.

1.1 Navigation using expected shortest paths

Our system uses artificial landmarks as visual cues for navigation in an unknown environment. The robot first explores the environment to learn the relative locations of the landmarks and builds a graph of landmark locations that it subsequently uses for navigation. During navigation, the robot plans motion paths along edges of the landmark visibility graph. If it wants to navigate from landmark s to landmark g, it plans and executes a path starting with a landmark visible from s. As it moves through the environment, it continually updates the graph with any newly acquired data, such as measured distance between two landmarks, or changes to the landmark visibility information. As is discussed in Sections 2 and 3, the planner uses estimates of the probabilities of landmark visibility to find paths with *expected shortest length* by solving a Markov decision process.

1.2 Landmark-based navigation

Many techniques have been employed for sensor-based navigation and localization. Industrial mobile robots have traditionally navigated by following painted lines on the floor or tracking buried wires or infrared beacons. The disadvantage of these approaches is that they require substantial engineering of the environment. Recently many researchers have employed landmarks — either artificial or extracted from the environment — to guide the motion of a mobile robot in indoor environments. The most commonly used approach with artificial landmarks is heuristic: landmarks are designed and placed so that landmark detection under normal circumstances is straightforward. The problem with the heuristic approach is that it only works under certain restricted conditions that are enforced for the sake of speed: the patterns must be viewed from a narrow range of distances and angles and will not be recognized if partially occluded.

We propose a self-similar pattern specifically designed for the application of mobile robot navigation. The pattern is quickly recognizable under a variety of viewing conditions, even when partially occluded or mounted at an angle. In contrast to existing approaches that require two-dimensional analysis of an image, our method finds matches along individual scanlines, without any preprocessing, making it suitable for real-time applications.

1.3 Related work

Traditionally, vision-based robot navigation has proceeded from three-dimensional maps of the environment, constructed, for example, using stereo vision techniques [3]. More recently, landmarks have been used to navigate without a full environment model. Techniques for mobile robot navigation based on landmarks include those that are primarily reactive [5], those planned within a geometric environment map enhanced with perceptual landmarks [9, 11], and those based on a topological description of landmark locations without a global map [8, 13, 17].

Our navigation system uses artificial landmarks placed throughout the environment as visual cues. A topological map of current landmark locations is first constructed during an exploratory phase and then used for navigation without requiring a global geometric map. To compensate for occlusion and unreliability of landmark detection, our navigation algorithm employs probabilistic techniques to construct reliable and efficient motion paths. Several different approaches to probabilistic path planning have been developed in related work. Blei and Kaelbling [2] describe Markov decision processes for finding shortest paths in stochastic graphs with partially unknown topologies. Their work differs from ours in that they assume that an edge is either passable or not, but that the state of each edge is only known with a certain probability. Kavraki and Latombe [7] propose a randomized method for configuration space preprocessing that generates a network of collision-free configurations in a known environment. Overmars and Švestka [12] describe a similar probabilistic learning approach that extends to a number of motion planning problems, including those for free flying planar robots, car-like robots, and robots with high degrees of freedom. Finally, a Markov model is used by Simmons and Koenig [18] to plan navigation strategies in partially observable environments.

Rather than relying on landmarks extracted from the environment [4, 5, 11, 13, 17], the approach taken in this paper and by a number of other research groups [1, 6, 10, 15, 19, 20] is to use artificial landmarks that can be easily and unobtrusively added to the environment. Becker *et al.* [1] use simple landmarks attached to the ceiling of the environment, and use a recognition algorithm that relies on a fixed distance of the pattern to the camera. Taylor and Kriegman [20] utilize the projective invariance of cross-ratios, but their approach cannot handle partial occlusion and requires specialized hardware for real-time performance. Lin and Tummala [10] propose three-dimensional landmarks consisting of two disks, which can be detected using Hough transforms from a restricted set of viewing angles. Unlike these approaches, our method uses simple 2D landmarks that can be recognized under a wide range of affine transformations in real-time without specialized hardware.

1.4 Outline of the paper

Each of the two central themes of the paper — robust navigation and the design of artificial landmarks — is discussed in two sections. Section 2 presents our framework for planning based on unreliable sensor data and develops an algorithm for computing expected shortest paths. Section 3 discusses how a visibility graph can be annotated with estimates of the unreliability of observations. Section 4 then introduces self-similar functions for the design of an optimally recognizable intensity pattern. Finally, Section 5 presents an algorithm for finding such patterns in an image under general affine transformations in real time.

2 Robust navigation using unreliable sensors

The discussion in this section is based on the following scenario:

We assume an unknown environment augmented with visual landmarks $\{L_1, L_2, \ldots, L_N\}$ that can be detected by the robot, albeit unreliably. We will use lowercase letters $a, b, \ldots$ when referring to individual landmarks. The robot navigates the environment by traveling along the edges of the visibility graph defined by the landmarks. We assume an edge from landmark a to landmark b has associated probability $p_{ab} \in [0, 1]$ and length $l_{ab} > 0$. The probability p_{ab} represents

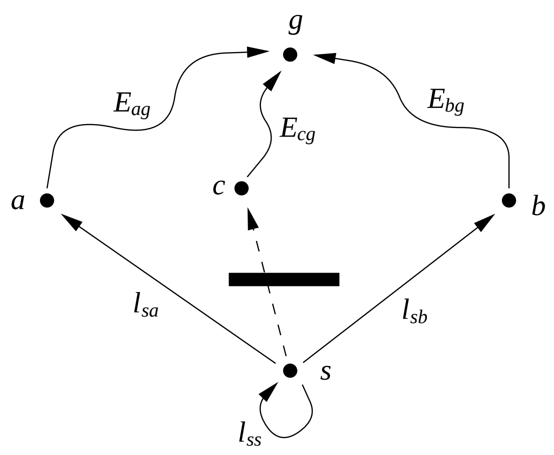

Figure 1: *Path planning example: The robot at landmark s can currently see landmarks a and b, but not c due to temporary occlusion.*

the likelihood that landmark b can be detected from landmark a. The length l_{ab} can be, for example, the physical distance between a and b, or the time it takes the robot to travel from a to b. In this section we investigate the problem of robot navigation given such a visibility graph. In Section 3 we discuss how such a graph can be constructed, and how probability and length factors can be estimated.

Path planning in visibility graphs typically employs shortest-path algorithms. Given a directed graph whose edges have fixed lengths, the shortest path from a start node s to a goal node g can be computed easily, for example using Dijkstra's algorithm. In our scenario, landmark detection is unreliable, and thus the edges of the graph can only be traversed some of the time. Therefore, we must change the notion of a shortest path to that of a *path with shortest expected length*, or *expected shortest path*.

2.1 Navigation using expected shortest paths

Before explaining how these shortest expected lengths can be computed, let us see how the robot can use them to plan its path. Suppose that the robot at landmark s can currently see landmarks a and b, but not landmark c (see Figure 1). Let E_{ag} denote the expected length of the shortest path from a to goal g. The total expected length of the path through a will be $l_{sa} + E_{ag}$. Similarly, the total expected length of the path through b will be $l_{sb} + E_{bg}$. Thus, the smaller of those two sums will indicate a candidate shortest path. Note that these lengths are independent of the probabilities p_{sa} and p_{sb}, since *at the current moment*, both a and b are visible. The path with overall shortest expected length, however, may go through neither a nor b. It is possible that an expected shorter path to g goes through landmark c, which usually is visible from s, but at the current moment is not (for example, due to temporary occlusion). This would be reflected in a low expected length E_{sg}. In this case, it would be better to stay at s and wait for c to become visible, rather than going to either a or b. To prevent the robot from staying at a landmark indefinitely, we associate a non-zero cost with this option (for example, the time it takes to acquire a new image). That is, each node n in the graph has a self-edge $n \to n$ with cost $l_{nn} > 0$ and probability $p_{nn} = 1$ (staying is always an option).

In summary, the robot will make its decision of whether to go to a, to go to b, or to stay at s based on which of the three sums $(l_{sa} + E_{ag})$, $(l_{sb} + E_{bg})$, and $(l_{ss} + E_{sg})$ is the smallest. If landmark c is permanently occluded, it may seem that the robot could "get stuck" at s. As will be discussed in Section 3, however, the current estimate of p_{sc} — the probability that c is visible from s — will decrease after a repeated failure to detect c. This will in turn increase the expected length of the path from s to g, until eventually the expected length of going through a or b will be shorter.

2.2 Deriving the expected lengths of the shortest paths

Given a designated goal g, we will now relate the unknown quantities E_{ng} (the expected lengths of the shortest paths from each node to the goal) in recursive equations. In the next section, we show that there is a unique solution to this system of equations if there is a path with non-zero probabilities from each node in the graph to goal g. It turns out that this problem is a special instance of a Markov decision process, which is discussed in section 2.4.

The following relations for the unknowns E_{ng} are motivated by the discussion in the previous section. To start, the expected length of the shortest path from the goal to itself is

$$E_{gg} = 0. \tag{1}$$

Next, let us consider a node n with only a single outgoing edge $n \to a$. The expected length E_{ng} of the shortest path from n to g can be expressed as a weighted sum of two terms that correspond to whether or not a is visible from n:

$$\begin{aligned} E_{ng} &= (1-p_{na})\,(l_{nn}+E_{ng}) \\ &+ p_{na}\,\min(l_{na}+E_{ag},\, l_{nn}+E_{ng}). \end{aligned} \tag{2}$$

The first term represents the case that a is not visible, which occurs with probability $1-p_{na}$. In this case the only choice is to remain at n and acquire another image, which incurs cost l_{nn} and results in an expected length of $l_{nn}+E_{ng}$. The second term represents the case that a is visible, which occurs with probability p_{na}. In this case the expected length of the shortest path is the smaller of $l_{na}+E_{ag}$ and $l_{nn}+E_{ng}$, corresponding to the options of going to a or staying at n. Recall that we are assuming that the goal is reachable from any node, and thus $E_{ag} < \infty$. Given that the edge $n \to a$ is the *only* edge leaving n, we know that all paths from n to g have to go through a, and thus that $E_{ng} \ge l_{na}+E_{ag}$. This allows us to solve equation (2) for E_{ng}, yielding

$$E_{ng} = \frac{1-p_{na}}{p_{na}}\,l_{nn} + l_{na} + E_{ag}. \tag{3}$$

Now, let us consider a node n with two outgoing edges $n \to a$ and $n \to b$. The relation for the expected length of the shortest path from n to g can be expressed analogously, except that now there are four cases, depending on which of a and b are visible. Using the shorthand $\overline{p}$ to denote $(1-p)$, we obtain the following relation for E_{ng}:

$$\begin{aligned} E_{ng} = {} & \overline{p_{na}}\,\overline{p_{nb}}\,(l_{nn}+E_{ng}) \\ & + p_{na}\,\overline{p_{nb}}\,\min(l_{na}+E_{ag},\, l_{nn}+E_{ng}) \\ & + \overline{p_{na}}\,p_{nb}\,\min(l_{nb}+E_{bg},\, l_{nn}+E_{ng}) \\ & + p_{na}\,p_{nb}\,\min(l_{na}+E_{ag},\, l_{nb}+E_{bg},\, l_{nn}+E_{ng}). \end{aligned} \tag{4}$$

It is easy to see how the equation generalizes to nodes with more than two outgoing edges. A node n with k outgoing edges $n \to a,\ n \to b,\ \ldots,\ n \to z$ yields an equation with 2^k terms:

$$\begin{aligned} E_{ng} = {} & \overline{p_{na}}\,\overline{p_{nb}}\,\ldots\,\overline{p_{nz}}\,(l_{nn}+E_{ng}) \\ & + p_{na}\,\overline{p_{nb}}\,\ldots\,\overline{p_{nz}}\,\min(l_{na}+E_{ag},\, l_{nn}+E_{ng}) \\ & + \overline{p_{na}}\,p_{nb}\,\ldots\,\overline{p_{nz}}\,\min(l_{nb}+E_{bg},\, l_{nn}+E_{ng}) \\ & + p_{na}\,p_{nb}\,\ldots\,\overline{p_{nz}}\,\min(l_{na}+E_{ag},\, l_{nb}+E_{bg}, \\ & \qquad l_{nn}+E_{ng}) \\ & + \ldots \\ & + p_{na}\,p_{nb}\,\ldots\,p_{nz}\,\min(l_{na}+E_{ag},\, l_{nb}+E_{bg},\, \ldots, \\ & \qquad l_{nz}+E_{zg},\, l_{nn}+E_{ng}). \end{aligned} \tag{5}$$

Unlike in the case for only a single edge, it is no longer possible to explicitly solve these equations for E_{ng} because of the minimum expressions, which recursively relate the unknown quantities E_{ig}. For example, to determine the value of the minimum expressions in equation (4), we need to know the ordering among K_a, K_b, and K_n, where K_i denotes $l_{ni}+E_{ig}$. While one can still deduce that K_n cannot be the smallest of the three, each of the remaining 4 orderings is possible:

$$\begin{array}{ll} K_a < K_b < K_n & K_a < K_n < K_b \\ K_b < K_a < K_n & K_b < K_n < K_a \end{array} \tag{6}$$

Note that if such orderings were known for all nodes, equations (4) and (5) would simplify to linear equations involving the unknowns E_{ng}. The expected lengths of the shortest paths could then be computed simply by solving a system of $N-1$ linear equations, where N is the total number of landmarks (including the goal). The presence of the minima, however, prohibits this approach.

To summarize, the landmark visibility graph defines a system of $N-1$ equations — each of the form of equation (5) — for the $N-1$ unknowns E_{ng}, $n \ne g$. We now turn to the question of how this system of equations can be solved.

2.3 An algorithm for computing the expected shortest paths

For notational convenience, let us collect the $N-1$ unknowns E_{ng}, $n \ne g$, into a vector $\mathbf{X} = [x_1, x_2 \ldots, x_{N-1}] \in \mathbb{R}^{N-1}$, and rewrite equation (5) in vector form:

$$x_n = f_n(\mathbf{X}). \tag{7}$$

Collecting the individual functions $f_n : \mathbb{R}^{N-1} \to \mathbb{R}$ (each of which is a linear combination of minima of components of $\mathbf{X}$ plus constant offsets) into a function $\mathbf{F} : \mathbb{R}^{N-1} \to \mathbb{R}^{N-1}$,

$$\mathbf{F} = (f_1, f_2, \ldots, f_{N-1}), \tag{8}$$

we can then rewrite the entire system of equations concisely:

$$\mathbf{X} = \mathbf{F}(\mathbf{X}). \tag{9}$$

Thus, the desired solution $\mathbf{X}$ of this system of equations is a *fixed point* of function $\mathbf{F}$. The properties of $\mathbf{F}$ are summarized in the following theorem; a proof of the theorem can be found in Appendix A.

Theorem 1 *If there exists a path to the goal from every node in the graph, and if all edges in the graph have non-zero probabilities $p_{ij} \in (0,1]$ and positive lengths $l_{ij} > \mathbf{0}$, then $\boldsymbol{F}$ has a unique fixed point $\boldsymbol{X}^*$ in $\mathbb{R}^{N-1}$. Moreover, the iterative process*

$$\boldsymbol{X}^{(k+1)} = \boldsymbol{F}(\boldsymbol{X}^{(k)}) \tag{10}$$

converges to this fixed point given the initial value $\boldsymbol{X}^{(0)} = \mathbf{0}$:

$$\boldsymbol{X}^* = \lim_{k\to\infty} \boldsymbol{X}^{(k)}. \tag{11}$$

This theorem translates literally into an efficient algorithm for computing the expected shortest paths E_{ig}. Starting with an initial estimate $\mathbf{X}^{(0)}$, iterate equation (10) until the value converges[1]. Convergence is geometric (i.e., the error decreases exponentially) once the current value is in the vicinity of the fixed point. In practice, usually no more than a few hundred iterations are necessary for the values to converge to within machine precision. (It is possible, however, to construct graphs for which convergence is arbitrarily slow.)

Instead of iterating equation (10), it is also possible to repeatedly *solve* the linear system of equations (9), using the previous solution to hypothesize which terms are the minima in each equation. Thus, instead of iterating over the *values* of the expected lengths, we can iterate over the *ordering* of the expected lengths of the outgoing paths at each node (as in equation (6)). The iteration terminates when the solution to the current system of equations yields the same ordering as the previous solution. It can be shown that this second algorithm (iterating over orderings) converges much faster than the first (iterating over values), but each iteration requires solving a system of linear equations, rather than just evaluating it.

[1] In fact, it can be shown that the process converges for any initial value $\mathbf{X}^{(0)} \in \mathbb{R}^{N-1}$.

2.4 Relation to Markov Decision Processes

The problem of expected shortest paths can be viewed as a special instance of a Markov decision process (MDP). Briefly, a MDP consists of a set of states S, and a set of allowable actions A_s associated with each state $s \in S$. Each action $a \in A_s$ taken in state s yields a reward $r(s,a)$, and results in a new (random) state s' according to a transition probability distribution $p(\cdot \mid s,a)$. The objective is to devise a *policy* or *decision rule* $d : S \to A_s$ that selects a certain (fixed) action in each state so as to optimize a function of the total reward. This brief discussion ignores many common variations of MDPs, including time-dependent or discounted rewards, and non-stationary policies. For a comprehensive introduction to Markov decision processes, see the book by Puterman [14].

Our problem of expected shortest paths translates into a non-discounted negative expected total-reward MDP. This means that each reward is interpreted as cost or penalty, and that the objective is to minimize the total expected cost. Upon reaching the goal g, no further cost is incurred. The key insight for relating our problem to a MDP is that the states in the MDP are not simply the nodes in the graph. Rather, each state encodes a node together with the set of outgoing edges currently passable. That is, a node with k outgoing edges contributes 2^k states. In each state, the allowable actions are to traverse any of the passable (visible) edges, including the self-edge. The cost associated with an action is the length of the edge traversed. The transition probabilities are the probabilities associated with the different states (visibility scenarios) of the *destination* node. For example, if the action is to go to a node x with 2 outgoing edges $x \to y$ and $x \to z$, the resulting state will be one of the 4 states encoding which of the 2 edges will be passable once x is reached; the corresponding probabilities are $\overline{p_{xy}}\,\overline{p_{xz}}$, $\overline{p_{xy}}\,p_{xz}$, $p_{xy}\,\overline{p_{xz}}$, and $p_{xy}\,p_{xz}$.

Note that while the corresponding MDP has many more states than there are nodes or edges, a policy can be specified by ordering the outgoing edges at each node (as in equation (6)). Then, for each subset of visible edges, the edge corresponding to the shortest path is chosen. Each such ordering can in turn be derived from a current estimate of the expected lengths of the shortest paths from each node to the goal. Thus, while the number of states $|S|$ of the MDP is

$$|S| = \sum_{n=1}^{N} 2^{od(n)},$$

where $od(n)$ is the out-degree of node n, the entire MDP can be concisely described with the $N-1$ unknowns E_{ng}.

Our algorithms for computing the expected shortest paths presented in section 2.3 are variants of two algorithms known as *value iteration* and *policy iteration* in the MDP community. An important difference is that in our case both algorithms require the iteration (or solution) of only $N-1$ equations, rather than of $|S|$ equations, as discussed in the previous paragraph.

3 Building the visibility graph

Now that we are armed with the ability to quickly compute the expected shortest paths from all nodes to a given goal node, we can apply the navigation strategy outlined in Section 2.1: At each node along the way, determine which landmarks are currently visible, and select among those the one that yields the overall expected shortest path (which includes the option of staying at the current position). Repeat the process at each subsequent node, until the goal is reached. If scanning for all visible landmarks is significantly more expensive than just looking for the next landmark on the predicted shortest path (for example, if scanning requires a 360 degree rotation of the robot), the strategy can be modified as follows: If possible, travel to each landmark in the sequence corresponding to the expected shortest path. Only if a landmark in the planned sequence cannot be detected, scan for all visible landmarks and replan.

3.1 Deriving cost and probability estimates

The remaining problem is how the visibility graph can be constructed and how length (cost) and probability factors can be estimated and maintained. Let us first discuss how to estimate the lengths of edges. Clearly, once a visibility edge has been traversed by the robot, its length l_{ij} is known and can be stored. Note that "length" will typically refer to the *time* it takes to traverse the edge. When artificial landmarks with known size are used (such as the ones presented in Section 4), we can also estimate the lengths of edges that have not yet been traversed, based on the size of the landmark in the image. Such estimates are fairly accurate, and are immediately available with each new visibility edge; they can be replaced with the measured distance once the edge has been traversed.

We now address the less obvious problem of determining the probabilities p_{ij} (that landmark j is visible from landmark i). Since these probabilities are unknown, the best we can do is to compute estimates $\hat{p}_{ij}$ for the true p_{ij} as a function g of the history of observations $\mathbf{O}_{ij}$:

$$\hat{p}_{ij} = g(\mathbf{O}_{ij}). \tag{12}$$

The history of observations of landmark j from landmark i is

$$\mathbf{O}_{ij} = [o_{ij}^{(1)}, o_{ij}^{(2)}, \ldots, o_{ij}^{(m_i)}], \tag{13}$$

where m_i is the total number of observations made from landmark i up to this point in time, and each $o_{ij}^{(k)} \in \{0,1\}$ records whether landmark j was visible. Note that observation histories can have "leading zeros"; that is, even if j was not visible the first few times an observation was made, it is possible to reconstruct the complete observation history for j by keeping track of all observations ever made at landmark i.

Let us now turn to the function g: how can we derive probability estimates from observation histories? In the simplest scenario, assuming independent observations made with a fixed probability p_{ij}, the optimal estimate $\hat{p}_{ij}$ is given by a function g that returns the ratio of detections ("ones") to the total number of observations m_i. In reality, however, the observations will neither be independent, nor will the p_{ij} stay constant over time. While some failures to detect a landmark will have truly random causes (for example, occlusion by a person walking by), others will be caused by lighting changes throughout the day, or perhaps even by permanent changes to the environment (most extremely, the removal or addition of a landmark). Typically, observations closely spaced in time will be highly correlated. Therefore, in practice, a more sophisticated estimation function g should be used. It is also a good idea to record a time stamp with each observation, so that the temporal distribution of observations can be taken into consideration.

3.2 Exploration and navigation

In our landmark-based navigation system, the robot operates in two modes: exploration and navigation.

In exploration mode, the robot explores the environment using a depth-first search among the unvisited landmarks. A visibility edge between two landmarks can be traversed by visual servoing, using the real-time recognition algorithm discussed in Section 5. At every newly-visited landmark, the robot scans for all landmarks visible from this position, records their relative angles and estimates of their distances, and starts an observation history for this landmark. As is discussed in Section 5.2, the landmarks all have unique barcodes, which are used as node labels in the graph. As mentioned above, in the process of exploring, the robot replaces distance estimates with more accurate odometry measurements. Also, as landmarks are revisited during the exploration phase, the observation histories are updated and the probability estimates are refined.

Once part of the environment has been explored, the robot can enter navigation mode, and accept navigation tasks from the user. For a given goal, the expected shortest paths are computed and used for path planning as described above. Navigation mode and exploration mode can be interleaved seamlessly; length and probability factors are continuously updated in both modes based on observations made and edges traversed. In summary, our navigation system is able to operate robustly in the presence of unreliable sensory input, and can cope both with the temporary occlusion of landmarks and with permanent changes to the environment, such as the removal and addition of new landmarks.

4 A self-similar landmark pattern

Our vision-based navigation system relies on real-time detection of landmarks in the environment. We have designed a self-similar intensity pattern [16] that can be quickly and reliably detected in images taken by the robot. In this section we motivate our use of a self-similar pattern and describe the pattern we have developed.

4.1 Self-similar functions

We say a function $f : \mathbb{R}^+ \rightarrow \mathbb{R}$ is *p-similar* for a scale factor p, $0<p<1$, if $f(x) = f(px)\ \forall x > 0$. The graph of a p-similar function is self-similar; that is, it is identical to itself scaled by p in the x-direction. Note that a p-similar function is also p^k-similar, for $k = 2, 3, \ldots$ Self-similar intensity patterns are attractive for recognition since the property of p-similarity is invariant to scale, and thus the distance of the pattern to the camera does not matter.

A p-similar pattern can be detected by comparing the observed intensity pattern to a version of itself scaled by p. We can accommodate patterns of limited spatial extent by restricting the comparison to a window of width w. Let $d_{p,w}(f)$ be the average absolute difference between the original and scaled functions over w:

$$d_{p,w}(f) = \frac{1}{w} \int_0^w |f(x) - f(px)|\, dx. \qquad (14)$$

Then f is *locally* p-similar over w if and only if $d_{p,w}(f) = 0$. A simple method for detecting a locally p-similar pattern in a one-dimensional intensity function $I(x)$ would then be to minimize the above measure over translations $I_t(x) = I(x + t)$:

$$t_{\text{match}} = \arg\min_t d_{p,w}(I_t).$$

The minimal value of 0 is achieved only if I is p-similar over w at translation t. Unfortunately, all constant functions are p-similar for any p. Thus $d_{p,w}(I_t)$ would also be minimal in regions of constant intensity. To exclude locally constant functions, we must detect patterns that are self-similar *only* for scale p (and $p^2, p^3, \ldots$), and not for other scales. This can be achieved by *maximizing* the *mismatch* at scale $p^{1/2}$ (and $p^{3/2}, p^{5/2}, \ldots$). Let us assume without loss of generality that the range of observable intensities is $[0, 1]$. A maximal mismatch for scale $\sqrt{p}$ is then given if $|f(x) - f(\sqrt{p}\,x)| = 1$, or locally, if

$$d_{\sqrt{p},w}(f) = \frac{1}{w} \int_0^w |f(x) - f(\sqrt{p}\,x)|\, dx = 1. \qquad (15)$$

In this case we say that f is (locally) $\sqrt{p}$*-antisimilar*.

We can combine equations (14) and (15), and revise our method for detecting self-similar patterns that are only p-similar for a given scale p. We will maximize the match function

$$m(t) = d_{\sqrt{p},w}(I_t) - d_{p,w}(I_t) \qquad (16)$$

over all translations t. Note that the range of $m(t)$ is $[-1, 1]$, since the range of both terms on the right-hand side of (16) is $[0, 1]$. A locally constant function will

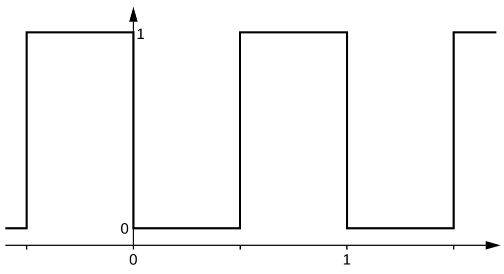

Figure 2: *The square wave function $S(x)$.*

yield $m(t) = 0$. Similarly, a random intensity pattern that is neither p-similar nor $\sqrt{p}$-similar will yield a response close to zero. A significant positive response is only expected for intensity patterns that are p-similar but not $\sqrt{p}$-similar.

4.2 An optimally recognizable pattern

Given the match measure m defined in (16), we now need to find an intensity function that yields the optimal response $m = 1$. That is, we seek a p-similar, $\sqrt{p}$-antisimilar function $s_p(x)$. To derive such a function, let us consider the periodic "square-wave" function S depicted in Figure 2:

$$S(x) = \begin{cases} 0, & x - \lfloor x \rfloor < \frac{1}{2} \\ 1, & x - \lfloor x \rfloor \geq \frac{1}{2} \end{cases} = \lfloor 2(x - \lfloor x \rfloor) \rfloor, \quad (17)$$

This function has the property that $S(x+1) = S(x)$ and $S(x+\frac{1}{2}) = 1 - S(x)$, i.e., it is a 1-periodic function that is similar under a *translation* of 1 and anti-similar under a translation of $\frac{1}{2}$. It is easy to show that we can transform S into a p-similar, $\sqrt{p}$-antisimilar function s_p by substituting $\log_p x$ for x:

$$s_p(x) = S(\log_p x) = \lfloor 2(\log_p x - \lfloor \log_p x \rfloor) \rfloor \qquad (18)$$

with the desired properties of $d_{p,w}(s_p) = 0$ and $d_{\sqrt{p},w}(s_p) = 1$ for any w (see Figure 3).

4.3 Two-dimensional landmarks

We now have all the components for landmark design and recognition. The key step for moving to two dimensions is to use a pattern that is p-similar in one direction (say, horizontally), and constant in the other direction. See Figure 4 for an illustration. If such a pattern is then sampled along any non-vertical line, the resulting intensity function is still p-similar because of

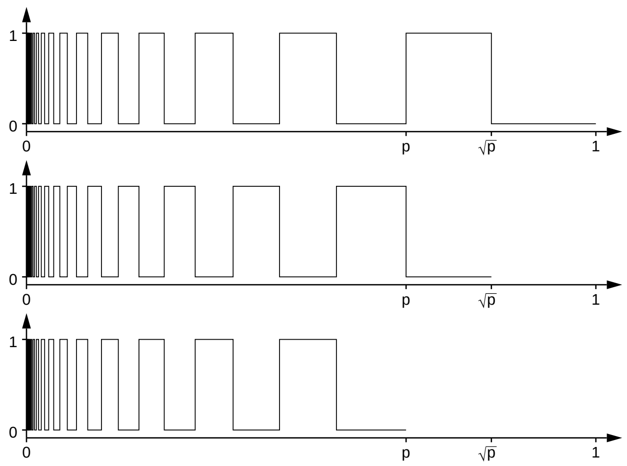

Figure 3: *Self-similar square wave $s_p(x)$ for $p = \frac{2}{3}$ (top); maximal mismatch at scale $p^{1/2}$ (middle); and match at scale p (bottom).*

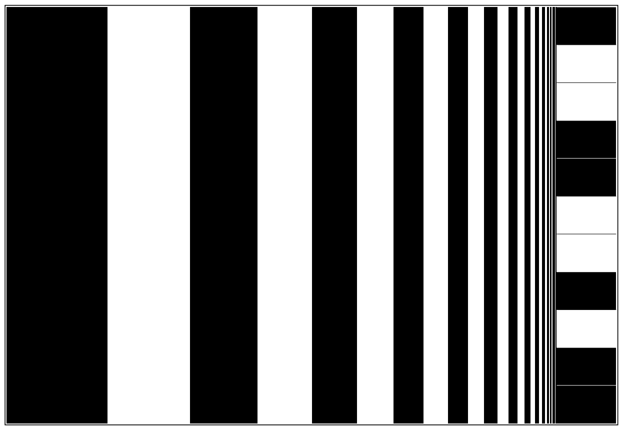

Figure 4: *Our self-similar landmark pattern with barcode.*

the scale invariance of self-similarity. This allows us to detect two-dimensional p-similar patterns that have undergone an affine transformation by examining isolated scanlines.

Formally, let

$$L(x, y) = s_p(x) \qquad (19)$$

be our two-dimensional landmark pattern, where $s_p(x)$ is the self-similar square wave function (equation (18)) for a fixed p. An affine transformation yields

$$A(L(x, y)) = L(ax+by+c, dx+ey+f) = s_p(ax+by+c).$$

Sampled at $y = y_0$, we get $s_p(ax + (by_0 + c)) = s_p(ax + t)$. Thus, the problem of finding an affine transformation of the two-dimensional pattern L has been reduced to finding a translation t of the one-dimensional pattern s_p.

5 The landmark recognition algorithm

The idea underlying the recognition algorithm is to find locations (x_m, y_m) in the image at which a scanline is locally p-similar and $\sqrt{p}$-antisimilar. To do this, we adopt the matching function m from equation (16) for scanlines in an image $I(x, y)$:

$$m_y(x) = \frac{1}{w}\int_0^w |I(x+\xi, y) - I(x+\sqrt{p}\,\xi, y)|\, d\xi - \frac{1}{w}\int_0^w |I(x+\xi, y) - I(x+p\xi, y)|\, d\xi. \quad (20)$$

The value of $m_y(x)$ depends on the intensities along the line (x, y) to $(x+w, y)$, and is constrained to the interval $[-1, 1]$. If the pattern s_p is present at (x, y), then $m_y(x) = 1$. It is easy to see that the pattern cs_p with reduced contrast $c < 1$ (i.e., difference between maximal and minimal intensities) will only yield a response $m_y(x) = c$. Other (non p-similar) intensity patterns will yield responses close to or below zero. An algorithm for finding affine transformations of a landmark with intensities $L(x, y) = s_p(x)$ (for a known p) in an image can be formulated as follows:

```
for every k-th scanline y
    for all x
        compute m_y(x)
    mark all strong local maxima of m_y as matches.
```

This simple algorithm requires only $O(nw/k)$ operations for an n-pixel image. The computation of $m_y(x)$ can be adapted to discrete images by replacing the integrals in equation (20) by summations, and determining inter-pixel intensities using linear interpolation. The two parameters, k, the spacing of scanlines to search, and w, the window size, only depend on the smallest expected size of the landmark pattern and can be fixed. Typical values for an image of size 640×480 are $k = 6$ and $w = 45$.

5.1 Finding matches reliably

The interesting question is: what constitutes a "strong" local maximum? A simple answer would be to require that a local maximum be greater than a fixed threshold $c_{\min}$ (corresponding to a minimum contrast). More information, however, can be gained from observing the shape of $m_y(x)$ in the vicinity of a local maximum. Figure 5 shows several intensity profiles of scanlines that were synthesized from continuous functions using 10-fold oversampling. Below each intensity plot is a plot of $m_y(x)$ for $p = \frac{2}{3}$ and $w = 50$. The length of each scanline is 400. The top two patterns are locally $\frac{2}{3}$-similar square waves s_p with full and half contrast, respectively. Both patterns result in clear peaks at the correct location $x_m = 350$; however, the value of the maximum $m_y(x_m)$ is less than expected. The observed

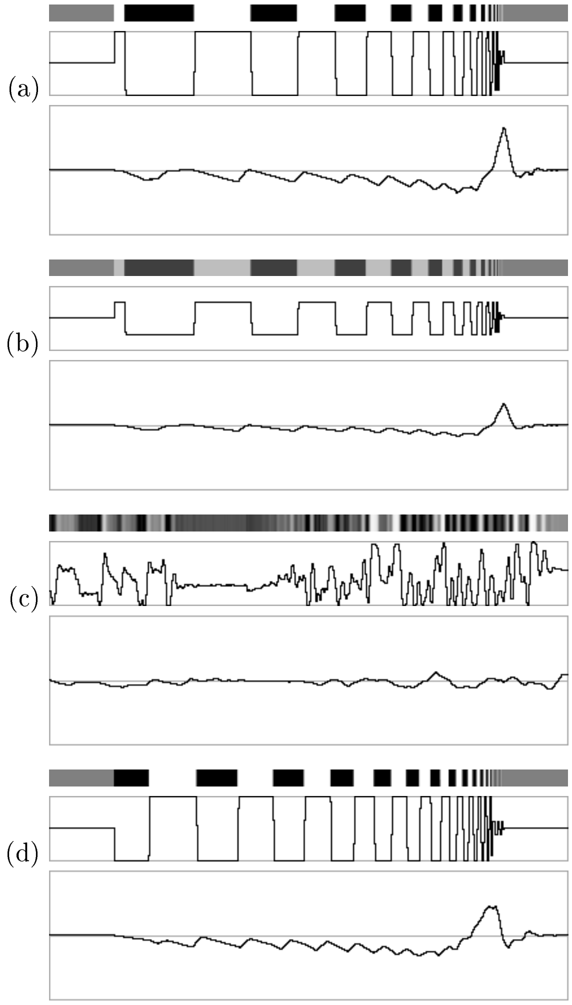

Figure 5: *Various intensity patterns $I(x)$ and match functions $m_y(x)$ for $p = \frac{2}{3}$ and $w = 50$. (See Section 5.1 for details.)*

values are 0.66 and 0.33, while the expected values are 1 and $\frac{1}{2}$. The differences are due to sampling and interpolation, in particular at locations of discontinuities or strong change. The bottom two patterns, on the other hand, are *not* $\frac{2}{3}$-similar: (c) is a "random" intensity pattern taken from a real image, and (d) is a $\frac{3}{4}$-similar square wave. The match function for the random pattern (c) has no strong peaks, but there is a distinctive local maximum in the match function for (d). It is, however, much more rounded and not as "sharp" as the top two peaks.

An experimental study analyzing the shape of $m_y(x)$ for a wide range of parameters has revealed a simple but effective test for "sharpness": check that a peak at half its height is no wider than a fixed threshold, which only depends on the window size w. That is, given a local maximum $v = m_y(x_m)$ greater than a threshold $c_{\min}$, test whether $m_y(x_m \pm \delta_x) < v/2$, where δ_x is approximately $w/10$.

5.2 Grouping matches

The actual landmark patterns can be detected in an image by grouping individual matches found on consecutive scanlines. To be able to distinguish different landmarks, we add a simple binary barcode to the right side of the self-similar pattern (as shown in Figure 4). The patterns can then be recognized and identified as follows: First, the position and orientation of all landmark patterns in the image is determined by fitting a straight line to each set of three or more individual matches detected on consecutive scanlines. The exact vertical extent of each pattern is then estimated from the intensity distribution to the left of this line. Once the locations of the patterns are known, their barcodes can be decoded easily. Figure 6 shows a Pioneer 2 mobile robot equipped with a camera, and two sample pictures taken by the robot. All landmarks are detected correctly, and all landmark numbers are decoded except for one that was too far away.

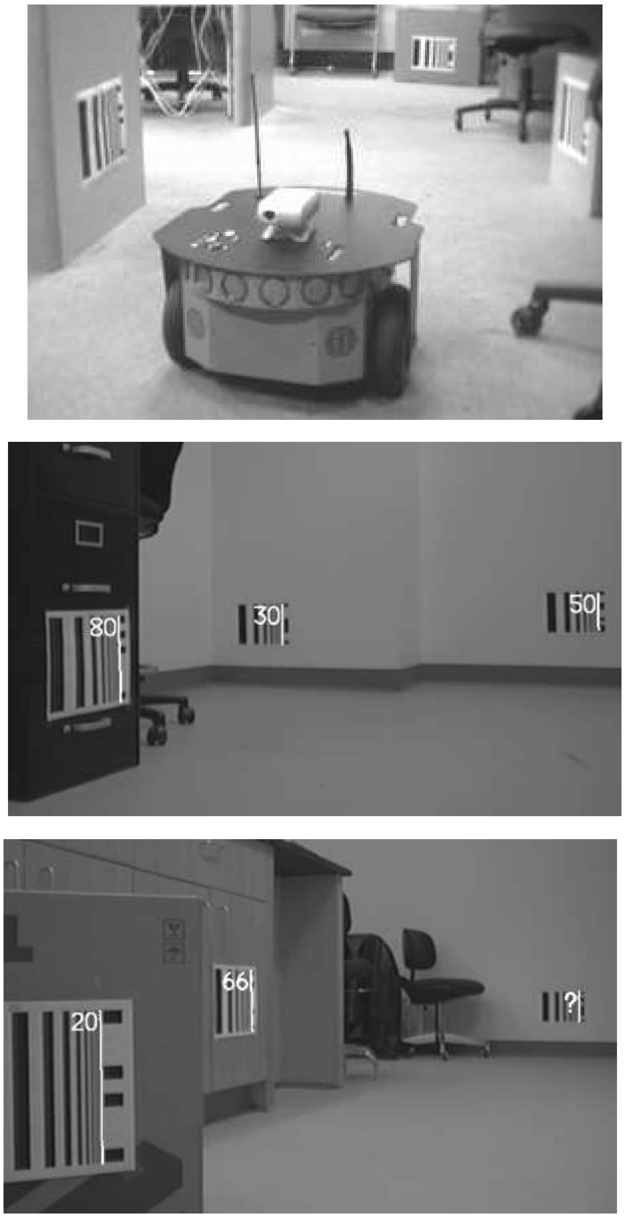

Figure 6: *The mobile robot, and two images taken with its camera. All landmarks have been found, and all but one have been identified by their barcodes.*

5.3 Achieving real-time performance

The recognition algorithm as described above is fairly fast: typical running times for a straightforward implementation on a 450 MHz Pentium II machine are 0.33 seconds for 640×480 images and 0.08 seconds for 320×240 images (using parameters $w = 45$ and $k = 6$).

A dramatic speed-up can be achieved by further restricting the set of pixels for which $m_y(x)$ is computed. Recall that we are already considering only every k-th scanline. We can start by looking at only every $2k$-th scanline, and, only if a match is found, look at its neighboring scanlines as well. Since isolated matches are discarded anyway in the grouping process, no landmark will be missed. This technique can yield a speed-up of up to 2 if few or no landmark patterns are present in the image. Another opportunity for restricting the search is to make use of the fact that peaks corresponding to matches have a certain minimum width at a given height h (e.g., $h = c_{\min}/4$). Given a lower bound l for this width, it is possible to scan for peaks by looking only at every l-th pixel. A conservative bound for l is typically given by δ_x, i.e., half the allowable peak width used in the test for sharpness discussed in Section 5.1. Typical values for this number are 4 or 5. We have found, however, that even with much higher values for l (e.g., 10), only very few peaks are overlooked (typically those resulting from patterns with low contrast), so further speedup can be achieved with minimal impact on robustness.

These modifications result in new running times of 0.027 seconds for 640×480 images and 0.012 seconds for 320×240 images, corresponding to frame rates of 37 and 83 frames per second, respectively. Thus, the implementation runs at video frame rate even for full-size images, making it the first real-time method for landmark detection under affine transformations.

6 Conclusion

Artificial landmarks that can be unobtrusively added to an indoor environment can serve as practical and inexpensive aids for mobile robot navigation and localization. When detected quickly and reliably they can make task execution more robust by reducing uncertainty due to control and sensing errors. When detection is unreliable, the navigation strategies planned should be as robust as possible. This paper has addressed the robust planning problem by providing two major contributions. First, we have described a probabilistic path planner that computes paths of expected shortest length, given landmark visibility histories. Second, we have proposed a novel landmark pattern together with the first practical method employing full affine invariants for real-time detection of landmarks. The method operates on single scanlines without any preprocessing and runs at video frame rate without specialized hardware. An implementation of the landmark detection algorithm is publicly available at `http://www.middlebury.edu/~schar/landmark/`.

Acknowledgements

We are grateful to Darius Braziunas, Huan Ding, Cristian Dima, Cuong Nguyen, Sorin Talamba, and Peter Wall for their insights and their assistance in implementing the algorithms presented here. Many thanks to Tim Huang and Bill Peterson for helpful discussions and literature pointers, and to Leslie Pack Kaelbling and the anonymous reviewers for their insightful comments.

Support for this work was provided in part by the National Science Foundation under grant CCR-9902032, POWRE grant EIA-9806108, VT-EPSCoR grant OSR-9350540, by Middlebury College, and by a grant to Middlebury College from the Howard Hughes Medical Institute.

Appendix A: Proof of Theorem 1

Proof: The strategy of the proof is to show that each component of $\mathbf{X}^{(0)}$, $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, ... forms a bounded increasing sequence, and thus has a limit. Setting $\mathbf{X}^{(0)} = [0, 0, \ldots, 0]$, it is straightforward to compute that the entries of the vector $\mathbf{X}^{(1)} = \mathbf{F}(\mathbf{X}^{(0)})$ are all strictly positive. Of particular importance here is the observation that every component of $\mathbf{X}^{(1)}$ is at least as big as (and in this case strictly greater than) the corresponding component of $\mathbf{X}^{(0)}$. With this base case verified, we can now argue by induction that each individual component entry of the sequence $\mathbf{X}^{(0)}$, $\mathbf{X}^{(1)}$, $\mathbf{X}^{(2)}$, ... forms a monotone increasing sequence. To see why, assume that the claim is true for $\mathbf{X}^{(0)}$, $\mathbf{X}^{(1)}$, ..., $\mathbf{X}^{(k)}$, and consider $\mathbf{X}^{(k+1)} = \mathbf{F}(\mathbf{X}^{(k)})$. The n-th component of $\mathbf{X}^{(k+1)}$ is given by $f_n(\mathbf{X}^{(k)})$ while the n-th component of $\mathbf{X}^{(k)}$ is given by $f_n(\mathbf{X}^{(k-1)})$. Using the induction hypothesis that every component of $\mathbf{X}^{(k)}$ is at least as big as that of $\mathbf{X}^{(k-1)}$, it is not hard to see from equation (5) that $f_n(\mathbf{X}^{(k)}) \geq f_n(\mathbf{X}^{(k-1)})$.

We must now make a case for boundedness of each component sequence. To begin, consider a node n in the graph containing an edge connected directly to the goal g. Looking at equation (5), the component function for this node would have the form

$$\begin{aligned} f_n(\mathbf{X}) \;&=\; \overline{p_{na}} \ldots \overline{p_{ng}} \ldots \overline{p_{nz}}\, (l_{nn} + x_n) \qquad (21) \\ &+\; p_{na} \ldots \overline{p_{ng}} \ldots \overline{p_{nz}}\, \min(l_{na} + x_a,\, l_{nn} + x_n) \\ &+\; \cdots \\ &+\; p_{na} \ldots p_{ng} \ldots p_{nz}\, \min(l_{na} + x_a, \ldots, \end{aligned}$$

$$l_{ng},\ \ldots, l_{nz}+x_z,\ l_{nn}+x_n).$$

The letters $a,\ldots,z$ represent nodes that are potentially visible from node n, so that x_n as well as $x_a,\ldots,x_z$ are just some subset of components from the vector $\mathbf{X}=[x_1,x_2,\ldots,x_{N-1}]$.

By choosing a node n that is adjacent to the goal g, we ensure that the length of the edge l_{ng} appears as a candidate in the final minimum of equation (21). The length l_{ng} also appears in several other minima of equation (21), but the special significance of this last minimum expression is that our hypothesis of non-zero probabilities guarantees that $p_{na}\ldots p_{ng}\ldots p_{nz}>0$.

Now construct the function $u_n(\mathbf{X})$ from $f_n(\mathbf{X})$ in the following way. Consider each minimum expression appearing in f_n. If l_{ng} is among the options, replace the minimum with l_{ng} (regardless of whether or not it represents the minimum). If l_{ng} does not appear, then replace the minimum with the $l_{nn}+x_n$ option (present in every minimum expression). In terms of the underlying graph, this amounts to ignoring all outgoing edges from node n except for l_{ng} and l_{nn}. By replacing each minimum with a particular candidate, we have certainly made the value of the function larger, i.e., $u_n(\mathbf{X})\geq f_n(\mathbf{X})\ \forall\mathbf{X}$. Moreover, combining terms, u_n reduces to a simpler equation for a node with only a single outgoing edge, similar to equation (2), and thus has the more accessible form

$$u_n(\mathbf{X})=p_{ng}\,l_{ng}+\overline{p_{ng}}\,(l_{nn}+x_n), \tag{22}$$

where $p_{ng}>0$ and thus $\overline{p_{ng}}<1$. For simplicity, we will write p instead of p_{ng} in the following discussion. The function u_n, although technically defined on $\mathbb{R}^{N-1}$, only depends on the n-th coordinate entry x_n. Also, since $\overline{p}<1$, u_n is *geometrically contractive* in the sense that for any two points x_n and x'_n we have

$$|u_n(x_n)-u_n(x'_n)|=\overline{p}\,|x_n-x'_n|.$$

What this implies is that iterating a point x_n with u_n yields a sequence $x_n,\ u_n(x_n),\ u_n^{(2)}(x_n),\ \ldots$ where the distance of the k-th iteration from the starting point x_n must satisfy

$$\begin{aligned}
&|x_n-u_n^{(k)}(x_n)| \qquad\qquad (23)\\
&\quad\leq\ |x_n-u_n(x_n)|+|u_n(x_n)-u_n^{(2)}(x_n)|+\ldots\\
&\qquad\quad +|u_n^{(k-1)}(x_n)-u_n^{(k)}(x_n)|\\
&\quad=\ |x_n-u_n(x_n)|\,(1+\overline{p}+\overline{p}^{\,2}+\ldots+\overline{p}^{\,k-1})\\
&\quad<\ |x_n-u_n(x_n)|\,\frac{1}{1-\overline{p}}.
\end{aligned}$$

In other words, the sequence is bounded. Now since for any $\mathbf{X}=[x_1,\ldots,x_n,\ldots,x_{N-1}]$ we have $u_n(x_n)\geq f_n(\mathbf{X})$, it follows that the monotone sequence in the n-th coordinate of $\mathbf{X}^{(0)},\ \mathbf{X}^{(1)},\ \mathbf{X}^{(2)},\ \ldots$ will also be bounded, and hence convergent.

To explicitly calculate the upper bound, set $x_n=0$ in equation (23) and solve to get

$$u_n^{(k)}(0)\ \leq\ (p\,l_{ng}+\overline{p}\,l_{nn})\frac{1}{p}\ =\ l_{ng}+\frac{\overline{p}}{p}\,l_{nn}.$$

The observant reader will recognize this as the expected length of the shortest path from a node with a single outgoing edge to the goal (equation (3)).

The preceding argument shows that the sequence $\mathbf{X}^{(k)}$ converges in any component corresponding to a node that is adjacent to the goal. With this fact established, we can now repeat the proof for any component of $\mathbf{X}^{(k)}$ corresponding to a node adjacent to a node previously handled. More explicitly, assume node r has an edge to node n, and assume we have shown that the n-th component of $\mathbf{X}^{(k)}$ is bounded and hence converges. The final minimum in the expression for $f_r(\mathbf{X})$ will contain, among other options, the expression $l_{rn}+x_n$. Knowing that $l_{rn}+x_n$ is bounded, by say b_n, we construct $u_r(\mathbf{X})=u_r(x_r)$ from $f_r(\mathbf{X})$ by replacing each minimum containing $l_{rn}+x_n$ with the upper bound b_n, and selecting $l_{rr}+x_r$ in all other cases. The remainder of the argument is the same.

Given our hypothesis that from every node there exists a path to the goal, this bootstrapping technique eventually leads to the conclusion that every component of our sequence $\mathbf{X}^{(0)},\ \mathbf{X}^{(1)},\ \mathbf{X}^{(2)},\ \ldots$ is an increasing bounded sequence of real numbers. By the Monotone Convergence Theorem, we may set $\mathbf{X}^*=(x_1^*,x_2^*,\ldots,x_{N-1}^*)$ where each x_n^* is the limit of the n-th components of $(\mathbf{X}^{(k)})$.

This can be summarized with the statement $\lim_{k\to\infty}\mathbf{X}^k=\mathbf{X}^*$. (Technically this is a coordinate-wise limit but we certainly get convergence using most any topology on $\mathbb{R}^{N-1}$.) Now the continuity of the component functions f_n which make up $\mathbf{F}$ allow us to conclude that

$$\begin{aligned}
\mathbf{F}(\mathbf{X}^*)\ &=\ \mathbf{F}(\lim_{k\to\infty}\mathbf{X}^{(k)})=\lim_{k\to\infty}\mathbf{F}(\mathbf{X}^{(k)})\\
&=\ \lim_{k\to\infty}\mathbf{X}^{(k+1)}=\mathbf{X}^*.
\end{aligned}$$

Finally, to argue that $\mathbf{X}^*$ is the unique fixed point of $\mathbf{F}$, let $\mathbf{Y}^* \in \mathbb{R}^{N-1}$ also satisfy $\mathbf{F}(\mathbf{Y}^*) = \mathbf{Y}^*$. Consider a particular component function f_n of $\mathbf{F}$ (see equation (21)) and again pay special attention to the final minimum term where the variables of all potentially visible nodes are included. If $\mathbf{X}^*$ is fixed by f_n, i.e., $f_n(\mathbf{X}^*) = x_n^*$, then $l_{nn} + x_n^*$ *cannot* be the minimum here since otherwise it would be the minimum throughout and we would have $f_n(\mathbf{X}^*) = x_n^* + l_{nn}$. (This is were we need the hypothesis that all edge lengths are strictly positive.) The same observation of course holds for $l_{nn} + y_n^*$. But now, using the fact that the probability preceding this final minimum is strictly positive, we can show that

$$|f_n(\mathbf{X}^*) - f_n(\mathbf{Y}^*)| < |x_n^* - y_n^*|.$$

Since $\mathbf{X}^*$ and $\mathbf{Y}^*$ are both assumed to be fixed by $\mathbf{F}$, this is only possible if $\mathbf{X}^* = \mathbf{Y}^*$. □

References

[1] C. Becker, J. Salas, K. Tokusei, and J.-C. Latombe. Reliable navigation using landmarks. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 401–406, June 1995.

[2] D. M. Blei and L. P. Kaelbling. Shortest paths in a dynamic uncertain domain. In *Proceedings of the IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*, 1999.

[3] O. Faugeras. *Three-Dimensional Computer Vision.* MIT Press, Cambridge, MA, 1993.

[4] C. Fennema, A. Hanson, E. Riseman, J. R. Beveride, and R. Kumar. Model-directed mobile robot navigation. *IEEE Transactions on Systems, Man, and Cybernetics*, 20(6):1352–1369, 1990.

[5] D. P. Huttenlocher, M. E. Leventon, and W. J. Rucklidge. Vision-guided navigation by comparing edge images. In Goldberg, Halperin, Latombe, and Wilson, editors, *1994 Workshop on the Algorithmic Foundations of Robotics, A. K. Peters*, pages 85–96, 1995.

[6] M. Kabuka and A. Arenas. Position verification of a mobile robot using standard pattern. *IEEE Journal of Robotics and Automation*, RA-3(6):505–516, December 1987.

[7] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 2138–2145, May 1994.

[8] A. Kosaka and J. Pan. Purdue experiments in model-based vision for hallway navigation. In *Proceedings of the Workshop on Vision for Robots in IROS'95, Pittsburgh, PA*, pages 87–96, 1995.

[9] A. Lazanas and J.-C. Latombe. Landmark-based robot navigation. *Algorithmica*, 13(5):472–501, May 1995.

[10] C. Lin and R. Tummala. Mobile robot navigation using artificial landmarks. *Journal of Robotic Systems*, 14(2):93–106, 1997.

[11] B. Nickerson, P. Jasiobedzki, D. Wilkes, M. Jenkin, E. Milios, J. Tsotsos, A. Jepson, and O. N. Bains. The ARK project: Autonomous mobile robots for known industrial environments. *Robotics and Autonomous Systems*, 25:83–104, 1998.

[12] M. H. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In Goldberg, Halperin, Latombe, and Wilson, editors, *1994 Workshop on the Algorithmic Foundations of Robotics, A. K. Peters*, pages 19–37, 1995.

[13] C. Owen and U. Nehmzow. Landmark-based navigation for a mobile robot. In *Proceedings of Simulation of Adaptive Behaviour.* MIT Press, 1998.

[14] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, New York, NY, 1994.

[15] J. Salas, J. L. Gordillo, and C. Tomasi. Visual routines for mobile robots: Experimental results. *Expert Systems with Applications*, 14:187–197, 1998.

[16] D. Scharstein and A. Briggs. Fast recognition of self-similar landmarks. In *Workshop on Perception for Mobile Agents (in conjunction with IEEE CVPR'99)*, pages 74–81, June 1999.

[17] R. Sim and G. Dudek. Mobile robot localization from learned landmarks. In *Proceedings of IEEE/RSJ Conference on Intelligent Robots and Systems (IROS), Victoria, BC*, October 1998.

[18] R. Simmons and S. Koenig. Probabilistic navigation in partially observable environments. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1080–1087, 1995.

[19] K. Tashiro, J. Ota, Y. C. Lin, and T. Arai. Design of the optimal arrangement of artificial landmarks. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 407–413, June 1995.

[20] C. J. Taylor and D. J. Kriegman. Vision-based motion planning and exploration algorithms for mobile robots. In Goldberg, Halperin, Latombe, and Wilson, editors, *1994 Workshop on the Algorithmic Foundations of Robotics, A. K. Peters*, pages 69–83, 1995.