September 1993 Report No. STAN-CS-93-1488

Planning the Motions of a Mobile Robot in a Sensory Uncertainty Field

by

H. Takeda, C. Facchinetti, J.-C. Latombe

Department of Computer Science

Stanford University

Stanford, California 94305

Planning the Motions of a Mobile Robot in a Sensory Uncertainty Field

Haruo Takeda* Claudio Facchinetti† Jean-Claude Latombe
Robotics Laboratory
Department of Computer Science, Stanford University
Stanford, CA 94305, USA

Abstract: *Failures in mobile robot navigation are often caused by errors in localizing the robot relative to its environment. This paper explores the idea that these errors can be considerably reduced by planning paths taking the robot through positions where pertinent features of the environment can be sensed. It introduces the notion of a "Sensory Uncertainty Field" (SUF). For every possible robot configuration* $\boldsymbol{q}$*, this field estimates the distribution of possible errors in the robot configuration that would be computed by a localization function matching the data given by the sensors against an environment model, if the robot was at* $\boldsymbol{q}$*. A planner is proposed which uses a precomputed SUF to generate paths that minimize expected errors or any other criterion combining, say, path length and errors. This paper describes in detail the computation of a specific SUF for a mobile robot equipped with a classical line-striping camera/laser range sensor. It presents an implemented SUF-based motion planner for this robot and shows paths generated by this planner. Navigation experiments were conducted with mobile robots using paths generated by the SUF-based planner and other paths. The former paths were tracked with greater precision than the others. The final section of the paper discusses additional research issues related to SUF-based planning.*

Acknowledgments: The research reported in this paper was conducted in the Robotics Laboratory, Department of Computer Science, Stanford University, under a grant by Hitachi, Ltd. Claudio Facchinetti was supported by a grant from the FNSRS (Fonds National Suisse de la Recherche Scientifique).

*Current affiliation: Systems Development Laboratory, Hitachi, Ltd,, 1099 Ohzenji, Asao-ku, Kawasaki 215, Japan.

†Current affiliation: Institute for Microtechnology, Neuchâtel, Switzerland

1 Introduction

The inner, fast-rate control loops of a mobile robot make use of dead-reckoning techniques to estimate the current robot configuration. As these techniques ultimately yield large cumulative uncertainty, environment sensors (e.g., vision) are often used by higher-level, slower-rate control loops to reduce uncertainty. Then a separate estimate of the current robot's configuration is computed by matching the incoming sensory data against a prior model of the environment. We call this second estimate the *sensed configuration.*

As environment sensing is not perfect, the sensed configuration is not error-free either. However, unlike the dead-reckoning estimate, the error in the sensed configuration does not depend on past history, nor it results in cumulative uncertainty. It mainly depends on the portion of the environment that is currently perceptible by the sensors (in addition to being a function of the intrinsic quality of the sensors). If appropriate environment features can be identified and localized by the sensors from the current actual robot configuration, a sensed configuration with relatively small uncertainty can be computed.

The navigation system of a mobile robot should strive to compute an optimized estimate of the current configuration at every instant. To that purpose, various techniques (e.g., Kalman filtering) have been proposed to combine over time the estimates provided by both dead-reckoning and sensory matching techniques. However, the precision of these localization techniques depends critically on the path followed by the robot. While many collision-free paths may connect the initial and goal configurations of the robot, some may allow the sensors to perceive more environment features, hence resulting in significantly reduced uncertainty at navigation time. The robot's navigation system should therefore include a planner that generates such paths.

In this paper we introduce a new approach to motion planning with uncertainty for mobile robots. Given a model of the robot's environment, a *Sensory Uncertainty Field* (SUF) is precomputed over the collision-free subset of the robot's configuration space. For every configuration $\boldsymbol{q}$ in this subset, the SUF estimates the distribution of possible errors in the robot configuration that would be computed by a localization function matching the data given by the sensors against the environment model, if the robot was at $\boldsymbol{q}$. The planner uses the SUF to generate paths that minimize expected

errors, by traversing areas of the workspace where environment features are visible. Other criteria, for example, combining path length and expected errors can be specified to the planner.

In Section 2 we relate our work to previous research. In Section 3 we give a general presentation of the notion of a SUF. In Section 4 we describe the construction of a particular SUF, for a robot equipped with a horizontal line-striping camera/laser range sensor. The same construction, or a similar one, would also apply to other range sensors, for example, a ring of infra-red range sensors. In Section 5 we describe an implemented SUF-based motion planner and we show paths generated by this planner. In Section 6 we report on navigation experiments done to measure how well a robot equipped with a rather generic localization function track paths generated by our planner, relative to other paths. In Section 7 we briefly discuss additional research issues related to our SUF-based planning approach.

This paper includes and extends results previously presented in [49, 50].

2 Related Work

Uncertainty in robot localization is a critical issue in mobile robot navigation. It has been the topic of a considerable amount of research.

First there has been substantial work aimed at developing techniques to compute the position (or configuration) of a robot by matching data acquired by one or several sensors against a prior model of the environment (*static localization*). Some of these techniques have been adapted and integrated with dead-reckoning techniques, e.g., odometry, to periodically update and optimize an estimate of the robot's position while it is moving (*dynamic localization*). Static localization involves complex combinatorial matching and is prone to matching errors due to imperfect sensory data. In contrast, dynamic localization, which often uses some sort of Kalman filtering, mainly consists of correcting sensory expectations based on a fast, initial estimate of the current robot's position obtained by dead-reckoning techniques only. Such expectations reduce combinatorial matching to a verification operation that not only requires less computation, but is also less sensitive to imperfection in sensory data. They may further be used to control the perceptual activities of the robot, e.g., by focusing a camera toward a pertinent feature of the environment, hence reducing computation even more. Techniques for static and dynamic localization are described in many papers, including

[2, 3, 7, 10, 15, 19, 26, 27, 29, 35, 36, 44, 47, 48, 54]. However, while these techniques are essential to make a robot track given motion paths as precisely as possible, they do not address the problem of generating these paths. If a path is selected that does not allow the robot to sense enough pertinent environment features along its way, these techniques will be powerless. Our research addresses that issue and focuses on planning motion paths allowing dynamic localization techniques to apply as well as possible. The navigation experiments reported in Section 6 were conducted with robots running a dynamic localization function embedding results described in the papers cited above.

Another line of research, which is closer to our work, has developed techniques for motion planning in the presence of uncertainty [30]. Significant results have been reported along this line, especially for generating fine part-mating motions in the mechanical assembly context. Several approaches have been proposed: skeleton refining [25, 38, 46, 52]), inductive learning [17], iterative removal of contacts [11, 32], and preimage backchaining [6, 12, 20, 21, 22, 31, 39]. The first three approaches operate according to two phases: a motion plan is first generated assuming null uncertainty and then transformed to deal with uncertainty. This approach may suit mechanical assembly rather well because many assembly operations are so constrained by the geometry of the parts that there does not exist a wide variety of paths to mate them. However, for mobile robots, uncertainty often affects the structure of a plan to the extent that it cannot be generated by transforming an initial one generated under the null-uncertainty assumption. The fourth approach, preimage backchaining, takes uncertainty into account throughout planning. However, it raises difficult computational issues that have led to using very simple sensor models (for example, localization errors are often assumed to be uniformly distributed in a disc of constant radius) and/or restricting its application to the planning of short sequences of motions. However, recent work on this approach has made it more attractive for mobile robot navigation. In particular, preimage backchaining has been extended to using vision sensors, allowing "visually compliant" motions to be included in a motion plan [24, 28]. Progress on the computational front has also been made by incorporating the notion of landmarks in the uncertainty model, thus reducing planning to selecting motion commands for going from landmarks to landmarks until the goal is attained [33, 34]. SUF-based planning also takes sensory uncertainty into account throughout planning, but it makes it possible to use more sophisticated models of sens-

ing uncertainty than preimage backchaining. On the other hand, as we will discuss in Section 7, there are other issues (e.g., dealing with uncertainty in motion control) that preimage backchaining seems to address more directly.

The notion of a SUF is closely related to that of a landmark as explored in several papers (e.g., [34, 37, 55]). A landmark is a characteristic physical feature of the environment which the robot can sense and use to localize itself. While in most previous work, landmarks tend to be binary (either they are visible, or they aren't), our SUF has a more continuous flavor expressing the fact that the precision of robot localization using a landmark depends on the relative position of the robot and the landmark. Variants of the landmark concept have been proposed with different names, e.g. "atomic region" [4], "perceptual alias" [18], "signature neighborhood" [41], and "perceptual equivalent class" [13, 14]. Using a similar concept, a planning method which makes use of a map consisting of four types of equivalent regions is described in [42]; in each region the robot can sense either no environmental edge, or a single edge, or two parallel edges, or two non-parallel edges. However, the variation of uncertainty within each region is not considered. The computation of the best viewpoint in terms of visibility of landmarks is studied in [53]. Techniques to determine the best sensor configuration to look at a given object are developed in [9, 51].

Many approaches to planning with uncertainty, including ours, assume that a complete prior model of the environment is available. Several planning approaches have been proposed when knowledge is incomplete (e.g., [8, 40]), but these approaches usually do not explicitly deal with uncertainty in sensing. However, see [12] for an extension of the preimage backchaining approach that combines uncertainty in control and sensing with uncertainty in the environment model. In Section 7 we will briefly discuss the issue of model incompleteness in SUF-based motion planning.

3 Notion of a Sensory Uncertainty Field (SUF)

In this section we give a general presentation of the notion of a sensory uncertainty field and how it can be used in motion planning. This presentation, which is independent of the robot and its sensors, is intended to serve as a guideline for the more technical following section, which instantiates the SUF notion for a mobile robot equipped with a line-striping range sensor.

Consider a robot moving in some environment represented by a model

M. The robot is subject to cumulative dead-reckoning errors. Therefore, even if it accurately knew its configuration at time zero, after a sufficiently long motion, the robot would no longer know its current configuration with enough precision to guarantee safe navigation. However, if it is equipped with environmental sensors, it can use them as an additional source of information to localize itself. Indeed, let the robot be at configuration $\boldsymbol{q}$ at time t. Let S denote the set of sensory data at this instant. By transforming S into a perceived partial geometric model of the environment and matching this partial model against the given model M, a localization function can compute a best estimate, $\hat{\boldsymbol{q}}$, of the current configuration $\boldsymbol{q}$.

The difference $\delta\boldsymbol{q} = \boldsymbol{q} - \hat{\boldsymbol{q}}$ is the localization error. The magnitude of this error depends on the type and quality of the robot's sensors and on the subset of the environment that is visible by them. For example, a mobile robot equipped with a range sensor would have no way to localize itself (other than by pure dead-reckoning) if it lies on an infinite empty planar surface. Instead, if various objects are distributed over the surface and if the range sensor can perceive some of them, such as the corner made by two walls, it can use this information to recover its position. However, the value of $\delta\boldsymbol{q}$ will depend on the position of the robot relative to the sensed objects. For instance, if both sides of a corner are largely visible, the error will be smaller than if one side is almost parallel to the sensor's line of view. Furthermore, sensory data are not fully deterministic. Two sets of sensory data, D_1 and D_2, obtained at two different times from exactly the same robot's configuration will not be identical. This leads us to consider $\delta\boldsymbol{q}$ as the value of a random variable $\Delta\boldsymbol{q}$. The distribution $\boldsymbol{U}(\boldsymbol{q})$ of this random variable at every collision-free configuration $\boldsymbol{q}$ characterizes the accuracy to which the robot can localize itself using its environment sensors. We call the function $\boldsymbol{U}$ the sensory uncertainty field.

If a probabilistic model of the robot's sensors is available, we can simulate this sensor assuming that the robot is at $\boldsymbol{q}$, use the model M to generate a set S of simulated sensory data, and run the matching algorithm on them. We could repeat this process as many times is necessary to get a reasonable model of the probabilistic distribution $\boldsymbol{U}$ at $\boldsymbol{q}$. By performing this same computation at all the collision-free nodes of a regular grid placed across configuration space, we would get a discrete approximation of $\boldsymbol{U}$ over the free space. This is essentially the idea developed in the next section for a mobile robot equipped with a horizontal line-striping range sensor. However, for this particular case, rather than performing many sensor simulations at

every configuration $\boldsymbol{q}$, which could be very time consuming, we will propose a more direct way to compute $\boldsymbol{U}$.

Once the SUF has been computed it is used by a planner applying a dynamic programming technique to search the collision-free subset of the robot's configuration space for a path optimizing a given cost function. This function can be chosen as simple as a norm of $\boldsymbol{U}$, hence seeking for highly reliable paths. But this choice may yield too long paths. Other cost functions, combining path length with magnitude of expected errors, result in paths whose reliability and length are both reasonable.

A SUF can be used for other purposes than planning. For example, it may be used to assess a new navigation environment (e.g., by determining if it contains large areas leading high localization errors) and decide whether it is worth engineering new landmarks. It can also be used in a reactive way during navigation. For example, if the robot considers that it does not know its current configuration with sufficient accuracy to accomplish a delicate operation (e.g., docking against an object), it may perform a short motion along the negated gradient of the SUF. A similar idea, called "recognizability gradient," is proposed in [14].

4 Computation of a SUF

We now consider a mobile robot rolling on a flat terrain. The robot can translate in any direction.[1] It is equipped with a line striping camera/laser range sensor. The laser projects a horizontal plane of light (actually, a cone). The camera senses points along the line where this plane intersects objects in the robot's environment. A simple tranformation computes the coordinates of these points in the horizontal plane from the coordinates of the corresponding pixels in the digitized image obtained from the camera [5]. The angular scope of vision of the sensor is limited. However, the sensor is mounted on a turret allowing the sensor to point in any direction. The rotation of the turret is independent of the rest of the robot. The retina of the camera is located at the center of the robot, which is also the center of rotation of the turret.

The configuration of the robot is defined by three parameters: the two coordinates of the center of the robot in the plane (the position of the robot)

[1] One robot used in our experiments is a NOMAD 200 of Nomadic Technologies.

and an angle measuring the orientation of the sensor. The sensor is geometrically contained in the cylindrical body of the robot. Hence, the objects in the environment (obstacles) map into the robot configuration space as cylinders whose cross-section is independent of the sensor orientation. The projection of these cylinders into the position subspace (horizontal plane) is obtained by growing the objects in the environment, isotropically, by the robot's radius.

We first present the uncertainty model of the range sensor. Next we compute the SUF corresponding to this model. Our sensor model is more realistic than most models used by other motion planners. However, it is still simpler than some models used by interpretation routines at execution time (e.g., see [15, 19, 43]). In fact, while a planner must anticipate all possible sensory data, a run-time navigation system only deals with the specific data returned by the sensors, which makes it easier to use sophisticated sensor models. The computation of the SUF is thus based on some simplifying assumptions. However, as far as planning is concerned, it is only important that the SUF captures the main effects of the environment features on localization uncertainty. Moreover, the fact that the planner uses one model does not prevent the run-time interpretation routine from using a more sophisticated one.

4.1 Uncertainty Model: The One-Line Case

We assume that all objects in the environment have polygonal cross-sections. Hence, the range sensor sees polygonal lines. Consider the case where a single infinite environment line is visible. Sensing this line allows the robot to estimate its distance to the line and its orientation relative to it. We compute the uncertainty of this estimate for a model of the sensor presented below.

Sensor model Let ℓ denote the environment line that is visible by the sensor. The sensor measures distances between the center point O of the robot and ℓ, along $2n+1$ horizontal rays $s_{-n}, \cdots, s_0, \cdots, s_n$ cast from O (Figure 1). Let ϕ_i denote the angle between the direction pointing from O perpendicularly to ℓ and the ray s_i, $i = -n, \cdots, n$. We assume that the sensing rays are uniformly distributed so that:

$$\phi_i = \phi_0 + i\tau, \quad i = -n, \cdots, -1, 0, 1, \cdots, n.$$

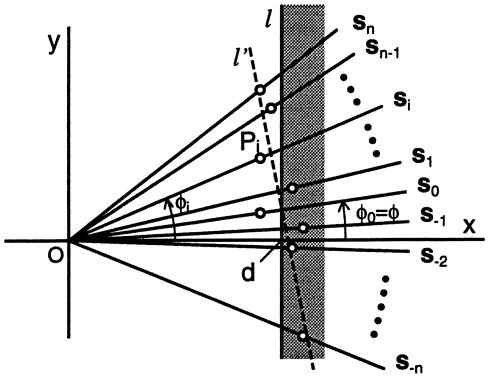


Figure 1: Sensor model

The rays s_i are called the *sensing rays* of the sensor. The ray s_0 is called the *axis* of the sensor. The angle $2n\tau$ is the *scope of vision* of the sensor.

Let Oxy be an orthogonal coordinate system with its x-axis perpendicular to ℓ. Let d be the actual distance between O and ℓ, and ϕ be the angle between the x-axis and the actual axis of the sensor (hence, $\phi = \phi_0$). Neither ϕ, nor d, is known by the robot.

Let d_i be the sensed distance between O and ℓ along the ray s_i, P_i be the point at distance d_i of O along this ray, and (x_i, y_i) be the coordinates of P_i in the Oxy system. Because of sensing errors, the points P_i are usually not located in ℓ. Fitting a line through them and assuming that this line is ℓ allow the robot to estimate ϕ and d with some error.

To establish the distribution of possible errors in the estimates of ϕ and d, we assume that the sensing error in the range data along any ray s_i is bounded by R and lies in an interval centered at the intersection of this ray and ℓ, with the length of this interval proportional to this distance. We also assume that the angle τ between two successive sensing rays is known exactly.[2] More formally, we regard x_i and y_i as the values of two random variables $\mathcal{X}_i$ and $\mathcal{Y}_i$ defined as follows:

$$\begin{cases} \mathcal{X}_i &= d(1+\mathcal{R}_i), \\ \mathcal{Y}_i &= \mathcal{X}_i \tan\phi_i, \end{cases} \tag{1}$$

where $\mathcal{R}_{-n}, \cdots, \mathcal{R}_0, \cdots, \mathcal{R}_n$ are independent random variables with the same

[2]We will see that these simplifying assumptions do not have significant impact on the computed SUF.

probability distribution in the interval $[-R, +R]$. The specific distribution of these variables, which does not have to be uniform, is not important here.

In the above model, sensing uncertainty increases with distance to sensor. We can furthermore limit the range of the sensor to some predefined value ζ_{max} by considering that every ray $\boldsymbol{s}_i$ such that $d/\cos\phi_i > \zeta_{max}$ senses nothing.

Uncertainty in ϕ and d If the range sensor was actually located at distance d from the environment line ℓ, with its axis pointing along the direction ϕ (as shown in Figure 1), sensing the positions of the points P_i would correspond to generating values of the random variables $\mathcal{R}_i$, hence values x_i and y_i of $\mathcal{X}_i$ and $\mathcal{Y}_i$. Using these coordinates, the equation of a line ℓ' would be established by running a line-fitting algorithm. By matching ℓ' to ℓ, the parameters ϕ and d would be estimated as $\phi^\star$ and $d^\star$.

The errors in these estimates are:

$$\begin{aligned} \delta\phi &= \phi - \phi^\star, \\ \delta d &= d - d^\star. \end{aligned}$$

The error in $d^\star$ corresponds to considering that the location of the robot's reference point O is in a line L parallel to ℓ at abscissa δd. No estimate of the ordinate of this location can be inferred by matching ℓ' to ℓ .

We regard $\delta\phi$ and δd as the values of two random variables $\Delta\phi$ and Δd. Every sensing of the environment line ℓ corresponds to generating a pair of values for these two variables. We define $u_{\phi,d,n}(\delta\phi, \delta d)$ as the joint probability density function of $\Delta\phi$ and Δd.

Let $\boldsymbol{r} = (r_{-n}, \cdots, r_0, \cdots, r_n)$ be a list of values for the random variables $\mathcal{R}_{-n}, \cdots, \mathcal{R}_0, \cdots, \mathcal{R}_n$. The domain of $\boldsymbol{r}$ is the hypercube $A \subset \Re^{2n+1}$ centered at the origin and defined by $\|\boldsymbol{r}\|_\infty \leq R$, where $\|\boldsymbol{r}\|_\infty$ is the sup norm of $\boldsymbol{r}$. Let $f_{\phi,d,n}(\boldsymbol{r})$ denote the function that maps any point $\boldsymbol{r} \in A$ to the error pair $(\delta\phi, \delta d)$. This function is completely defined by the above expressions and the line-fitting algorithm.

In the following we assume a classical eigenvector line-fitting algorithm [16]. This algorithm is described in Appendix A for completeness, along with the corresponding expressions of $\phi^\star$, $d^\star$, $\delta\phi$, and δd. The curves shown below have been computed using these expressions.

The image of A by $f_{\phi,d,n}$ is a region $B \subset \Re^2$. It represents the locus of all possible error pairs $(\delta\phi, \delta d)$ for our sensor model, i.e., the domain

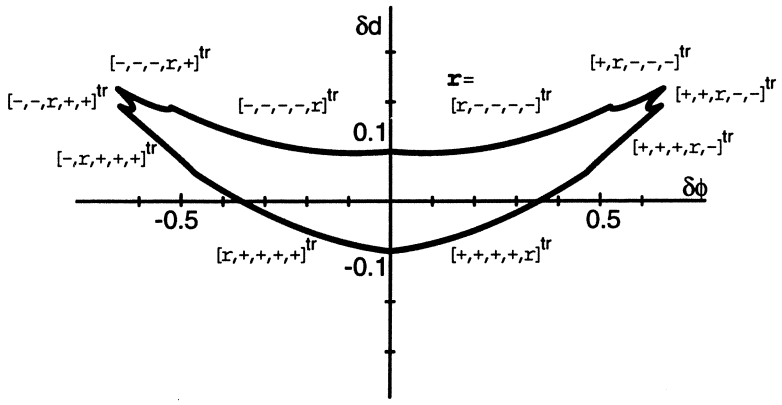


Figure 2: Boundary of the region $f_{0,1,2}(A)$

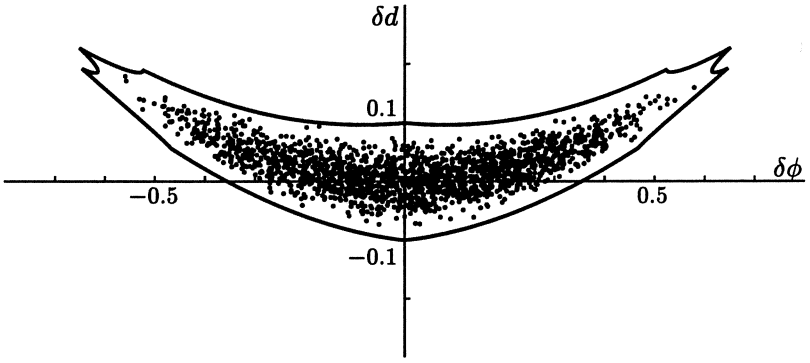
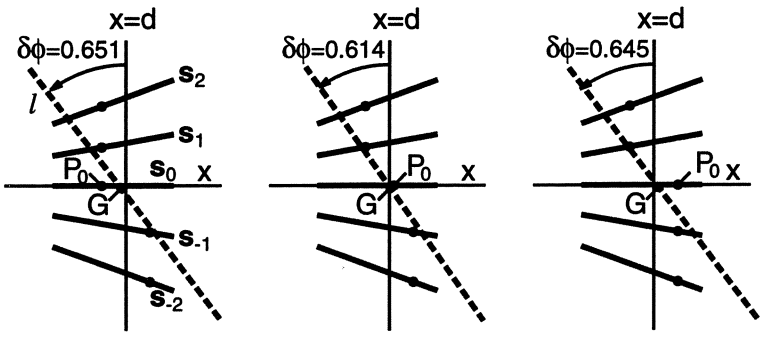


Figure 3: Sample errors with a uniform distribution of the r_i's

outside which $u_{\phi,d,n}$ is identically null. The image of every edge of A is a curve segment. By computing the image of many points randomly selected in A, we have verified experimentally (but not proven analytically) that the boundary ∂B of B is a subset of these curve segments.

To illustrate, consider Figure 2. It shows the computed boundary of the region $f_{0,1,2}(A)$ with $\tau = 5°$ and $R = 0.1$. This boundary consists of 10 segments, each one corresponding to an edge of the hypercube A. For example, the curve segment marked $[+,+,+,r,-]^{tr}$ is the portion of ∂B



(a) r= -0.1 (b) r=0.0187 (c) r=0.1

Figure 4: Displacement of the best-fit line

that corresponds to the edge $\{[0.1, 0.1, 0.1, r, -0.1]^{tr} \mid r \in [-0.1, +0.1]\}$. Figure 3 shows the same curve along with 2000 points representing error pairs $(\delta\phi, \delta d)$ computed for random values of the r_i's uniformly distributed in $[-0.1, +0.1]$, with the parameters τ, ϕ, and d set as above.

Comment: The conspicuous shapes of the segments $[+, +, r, -, -]^{tr}$ and $[-, -, r, +, +]^{tr}$ can be explained as follows. Consider, for example, $[+, +, r, -, -]^{tr}$. When $r = -0.1$, the point P_0 is as shown in Figure 4(a) and the mean point G of the P_i's is on the left of the line ℓ, slightly under the horizontal x-axis as $\boldsymbol{s}_{-1}$ and $\boldsymbol{s}_{-2}$ are located further away from this axis than $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$, respectively. When r increases toward 0 and beyond, P_0 moves toward the right. G also moves toward the right, but by a very small amount compared to P_0. In fact, for the purpose of this discussion, we can consider that G is fixed. When P_0 gets closer to ℓ (on the left side), ℓ' becomes more vertical and $\delta\phi$ decreases (however, the variation is small and barely visible in Figure 4). For some value of r close to 0 (actually computed as 0.0187, which corresponds to the situation when P_0G is perpendicular to ℓ'), $\delta\phi$ is minimal (Figure 4(b)). When r increases further toward 0.1, ℓ' tilts again and $\delta\phi$ increases (Figure 4(c)).

Other examples Figure 5 shows the boundary of $f_{\phi,1,n}(A)$, when $\tau = 5°$ and $R = 0.1$, computed for $\phi = 0°, 20°, 40°, 60°$ and $n = 2, 4, 6$. (When $n = 6$ and $\phi = 60°$ the ray of direction ϕ_6 does not intersect the line ℓ. This is why the boundary shown in Figure 5(l) is computed for $\phi = 50°$.)

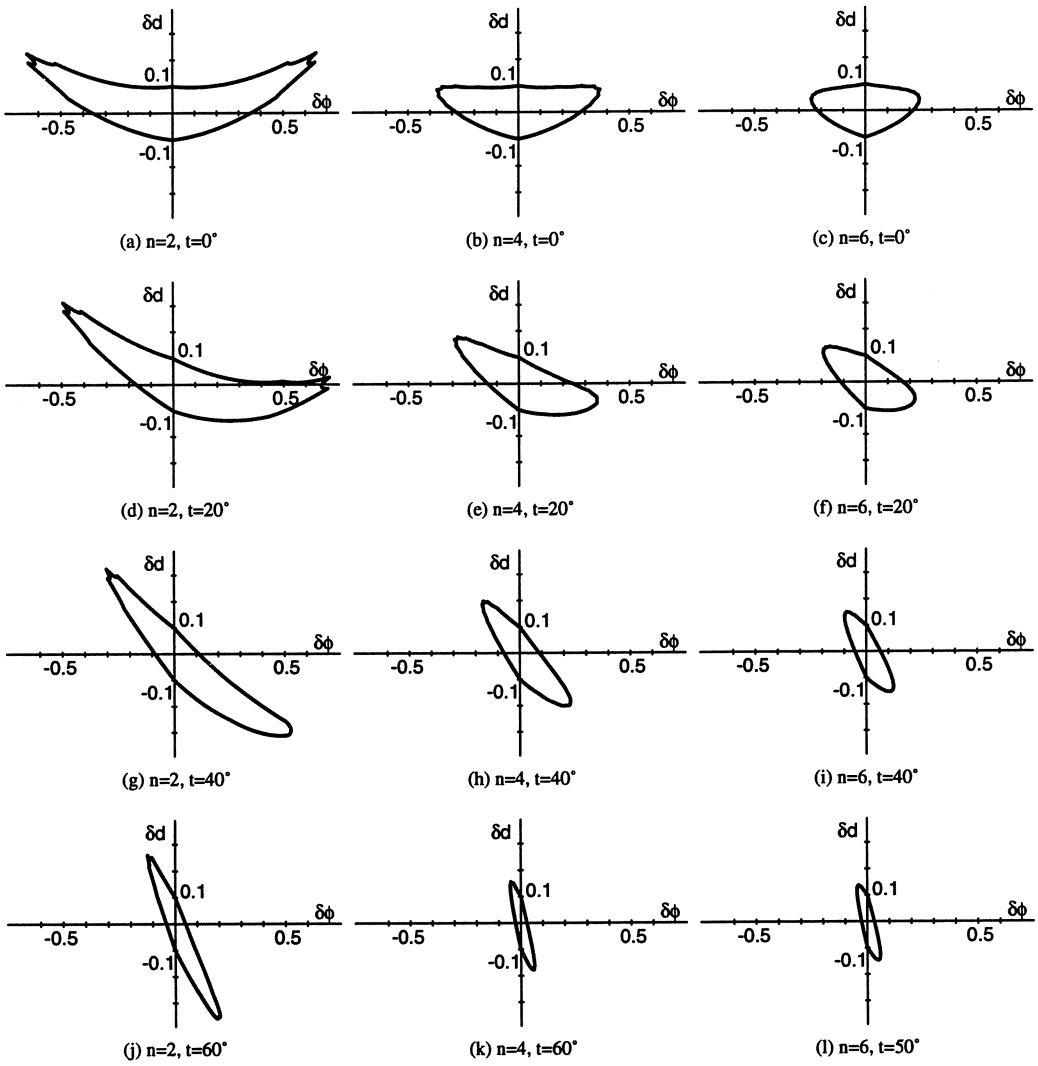


Figure 5: Boundary of $f_{\phi,1,n}(A)$ for various values of ϕ and n

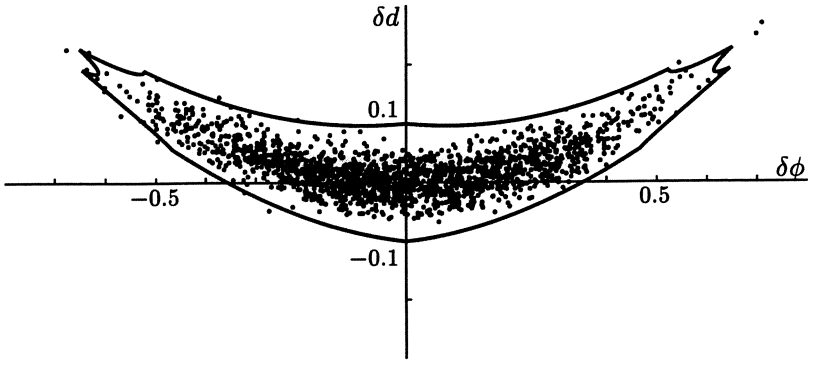


Figure 6: Sample errors with a normal distribution of the r_i's

The length of the projection of a boundary on the δd-axis is the range of possible errors in the estimate of d. One can see that, for a fixed value of n, it first increases as ϕ increases from 0, and later decreases after it reached a maximum. This nonmonotonic variation is the result of two more basic variations with opposite effects. When ϕ augments, both the length of the portion of the line ℓ that is visible by the sensor and the average length of the rays increase. While the increase in the length of the visible portion of ℓ tends to reduce uncertainty, the increase in the average length of the rays tends to augment uncertainty. For a fixed value of the sensing direction ϕ, uncertainty monotonically decreases as n (i.e., the scope of vision) increases.

Discussion The above results hold for any bounded distribution of the r_i's. Moreover, the curve ∂B constructed as above still gives a good approximation of a domain outside which the density function $u_{\phi,d,n}(\delta\phi, \delta d)$ takes very small values when the r_i's have a non-bounded normal distribution. For instance, Figure 6 shows error pairs $(\delta\phi, \delta d)$ computed for 2000 random values of $\boldsymbol{r}$ with the parameters τ, ϕ, and d set as in Figure 2 and a normal distribution of the r_i's of variance 0.0333 (the variance of a uniform distribution over $[-0.1, +0.1]$).

The above results remain satisfactory even if we allow some errors in the ray directions. Figure 7 shows error pairs $(\delta\phi, \delta d)$ under the same conditions as above, except that, to compute the 2000 error pairs, we let each angle ϕ_i

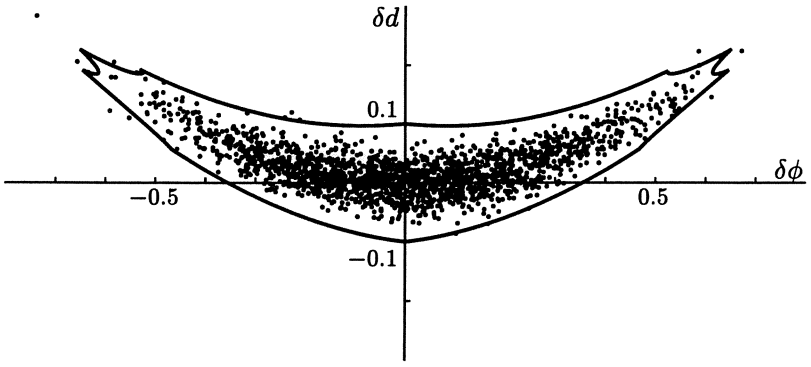


Figure 7: Sample errors with errors in the ray directions

take normally distributed random values with standard deviation 1°.

4.2 SUF Computation

We now consider the case where the range sensor detects multiple non-parallel environment edges in a polygonal environment. Matching these edges to a prior geometric model M of the environment allows the robot to estimate its configuration (X, Y, Φ), where (X, Y) is the position of the center point O in a coordinate system ΩXY attached to the environment (the horizontal plane) and Φ is the angle between the X-axis of this coordinate system and the axis of the sensor.

Estimation of the robot configuration Assume that, at some time t during navigation, the robot's actual configuration (unknown by the navigation system) is (X, Y, Φ). Assume that m environment edges e_i (finite line segments) are visible by the sensor at this configuration. Let ℓ_i denote the infinite line supporting e_i $(i = 1, \cdots, m)$. (If the sensor has limited range ζ_{max}, we only consider the environment edges e_i, or the portions of them, which are contained in the disk of radius ζ_{max} centered at $\boldsymbol{q} = (X, Y, \Phi)$.)

Let us assume that the sensor detects all the m edges as e_i', $i = 1, \cdots, m$. This is a realistic assumption for a line-striping sensor, if the edges have sufficient length. Indeed, during navigation the robot can use an expectation-

driven dynamic localization function, which virtually eliminates all matching errors for this kind of sensor (see Section 6). Let ℓ'_i, $i = 1$ to m, denote the lines supporting these sensed edges obtained by the line-fitting algorithm. Matching them against the N lines in the environment model M gives a map $g : i \in [1, m] \mapsto j \in [1, N]$ that associates some line ℓ_j in the model with every sensed line ℓ'_i. The map g may not necessarily be a surjection, nor an injection.

The map g is then used to compute an estimate $(X^\star, Y^\star, \Phi^\star)$ of (X, Y, Φ). From each line pair (ℓ'_i, ℓ_j) such that $j = g(i)$, one can infer a value of $\Phi^\star$, as well as a line L_{ij} parallel to ℓ_j containing $(X^\star, Y^\star)$. If there were no errors in sensing, all the line pairs (ℓ'_i, ℓ_j) would result both in the same value of $\Phi^\star$ and in lines L_{ij} intersecting at a single point. The value of $(X^\star, Y^\star)$ would be the coordinates of this point. Due to imperfect sensing, the various pairs (ℓ'_i, ℓ_j) usually result in slightly different estimates of Φ and in lines L_{ij} that are not concurrent. Various techniques (e.g., averaging) can be used to produce a best estimate $(X^\star, Y^\star, \Phi^\star)$.

We regard the errors in the computed estimate

$$\delta X = X - X^\star, \quad \delta Y = Y - Y^\star, \quad \delta\Phi = \Phi - \Phi^\star,$$

as the values of three random variables ΔX, ΔY, and $\Delta\Phi$.

Construction of the SUF Recall from Section 3 that computing the SUF at some configuration $\boldsymbol{q}$ consists of "simulating" the sensor at this configuration and computing the distribution $\boldsymbol{U}(\boldsymbol{q})$ of the localization errors. Hence, our goal is now to characterize the joint probabilistic distribution of δX, δY, and $\delta\Phi$ from the model of the sensor. Rather than simulating the sensor a large number of times, we propose a more direct technique presented below.

Consider that the robot is at $\boldsymbol{q} = (X, Y, \Phi)$. A simple visibility computation using the environment model allows us to extract the environment edges that are visible from $\boldsymbol{q}$. Let us denote these edges by e_k, $k = 1, \cdots, p$. Notice that an environment edge e_j ($j \in [1, N]$) may not be completely visible. Hence, each e_k may only be one portion of some e_j and several e_p's may be disconnected portions of the same environment edge e_j. Let ℓ_k be the supporting line of e_k, d_k the perpendicular distance from O to ℓ_k and α_k the direction of the inward normal to ℓ_k.

Figure 8 illustrates these notations. The scope of vision of the simulated

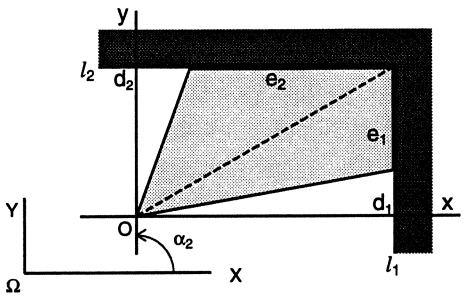


Figure 8: Detection of multiple edges

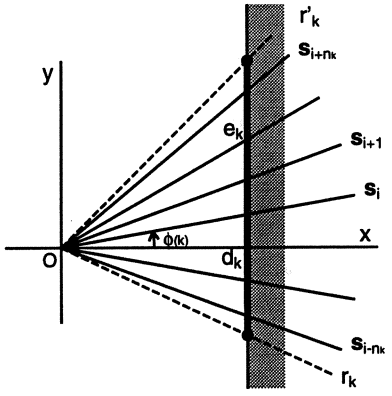


Figure 9: Restriction of the sensor to one edge

sensor contains two (partial) edges e_1 and e_2 supported by two perpendicular lines ℓ_1 and ℓ_2. We have $\alpha_1 = 0°$ and $\alpha_2 = 90°$.

Let r_k and r'_k be the two rays drawn from the center point O and passing through the endpoints of e_k (see Figure 9). Let $\phi_{(k)}$ denote the angle between the line of direction α_k (perpendicular to ℓ_k) and the sensing ray $\boldsymbol{s}_i$ ($i \in [-n, +n]$) that is the closest to the bisector of the two rays r_k and r'_k. Let n_k be the largest integer such that all the rays $\boldsymbol{s}_{i-n_k}, \cdots, \boldsymbol{s}_{i+n_k}$ lie between r_k and r'_k. Below we consider only those edges e_k for which $n_k \geq 1$. This corresponds to setting a threshold on the acceptable length of an edge portion; if an edge portion is shorter than this threshold, it is not considered to be visible by the sensor with sufficient reliability.

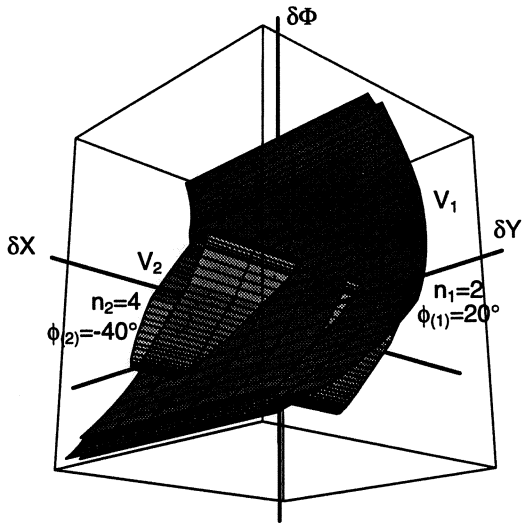


Figure 10: Construction of the region where $\mathcal{U}_{X,Y,\Phi}(\delta X, \delta Y, \delta\Phi) \neq 0$

We define:

$$\mathcal{U}_{X,Y,\Phi}(\delta X, \delta Y, \delta\Phi) \;=\; \prod_{k=1}^{k=p} u_{\phi_{(k)},d_k,n_k}(\theta, |x\cos\alpha_k + y\sin\alpha_k|).$$

The function $\mathcal{U}_{X,Y,\Phi}(\delta X, \delta Y, \delta\Phi)$ can be interpreted as an approximation of the joint probabilistic density of ΔX, ΔY, and $\Delta\Phi$, when the sensor's configuration is (X, Y, Φ). We compute the *sensory uncertainty field* as the functional $\boldsymbol{U}(X, Y, \Phi) = \mathcal{U}_{X,Y,\Phi}(.,.,.)$.

The region of $\Re^3$ where $\mathcal{U}_{X,Y,\Phi}$ is non-zero approximates the range of possible localization errors at run-time. Figure 10 illustrates the construction of this region for the example shown in Figure 8. It is obtained by intersecting two volumes, V_1 and V_2, corresponding to two visible edges e_1 and e_2, respectively. The cross-section of the volume V_1 (corresponding to e_1) at any constant Y is the region shown in Figure 5(d) with a scaling factor d_1.[3] The cross-section of the volume V_2 (corresponding to e_2) at any constant X is the region shown in Figure 5(h) with a scaling factor d_2. We compute a discretized approximation of the intersection of V_1 and V_2 by discretizing the parameter space and determining which errors fall in the intersection of V_1 and V_2.

[3]We show in Appendix B that $u_{\phi,d,n}(\delta\phi, \delta d) = u_{\phi,1,n}(\delta\phi, (\delta d)/d)$.

Discussion If no two non-parallel edges are visible by the sensor at some configuration (X, Y, Φ), then the region where the function $\mathcal{U}_{X,Y,\Phi}$ is non-zero extends to infinity in some direction. In our implementation, we arbitrarily set $\mathcal{U}_{X,Y,\Phi}$ to a uniform distribution over a large error domain, along this direction.

Our computation of $\mathcal{U}_{X,Y,\Phi}$ does not account for the fact that environment edges are line segments of finite lengths. Doing so would be meaningful only if the edge endpoints could be sensed precisely, which the range sensor described here is not very good at. On the other hand, whenever the two edges abutting a corner are visible, the location of the corner does not provide any additional information, since it is implicitly taken into account by fusing the information given by the two edges.

Our computation of $\mathcal{U}_{X,Y,\Phi}$ does not take into account the specific technique used by the localization function to infer the best estimate $(X^\star, Y^\star, \Phi^\star)$ from the various pairs (ℓ'_i, ℓ_j) provided by the matching map g. However, the effect of such a technique on the localization error tends to be small compared to the impact of the arrangement of edges that are visible by the sensor. More generally, the constructed SUF seems to be only marginally dependent on the particular localization function that is used (including the line-fitting algorithm), provided that this function is a reasonably good one.

5 SUF-Based Motion Planning

5.1 Planning Method

We now describe a simple form of motion planning using the sensory uncertainty field computed as in the previous section. Our planner performs a simple dynamic search in a regular grid (the grid over which the SUF was computed) in order to find a path that optimizes some given criterion.

A planning problem consists of generating a collision-free path ν between two given configurations $\boldsymbol{q}_{init}$ and $\boldsymbol{q}_{goal}$ that minimizes the functional:

$$\mathcal{J}(\nu) = \int_\nu [\boldsymbol{F}(X, Y, \Phi)]^\gamma \, d\nu,$$

where $\boldsymbol{F}$ is a function measuring the "magnitude" of the SUF. In our planner, we use $\boldsymbol{F}$ defined by:

$$\boldsymbol{F}(X, Y, \Phi) = \iiint_{\mathcal{U}_{X,Y,\Phi} \neq 0} d(\delta X)\, d(\delta Y)\, d(\delta \Phi).$$

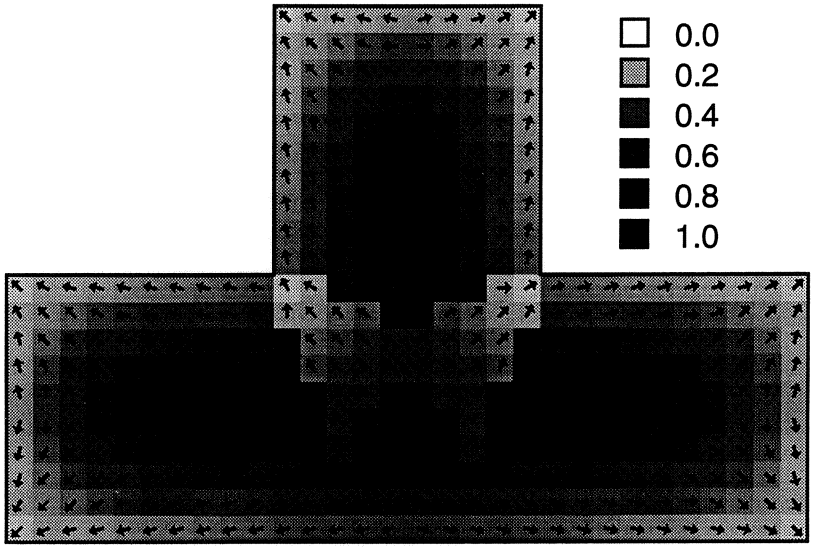


Figure 11: Minimum SUF values in a simple environment

It measures the volume of the region where $\mathcal{U}_{X,Y,\Phi}$ is non-zero, hence the volume of the error domain. It is computed by placing a grid in the error space and counting the number of points of this grid where $\mathcal{U} \neq 0$. To construct the grid we select the metric along the $\delta\Phi$-axis, on the one hand, and the metric along the δX and δY-axis, on the other hand, so that the maximal errors in all directions defined as above have roughly the same order of magnitude.

Figures 11 and 12 display maps showing the values of $\boldsymbol{F}(X, Y, \Phi)$ over a simple T-shaped environment. In this example, as in the following ones, we assume for simplicity, but without loss of generality,[4] that the robot is a point. Figure 11 shows the minimal value of $\boldsymbol{F}(X, Y, \Phi)$ for all possible orientations Φ of the sensor, at every position (X, Y) in a grid placed over the environment. The grey intensity represents this value; the darker the pixel, the higher the uncertainty. At every position, the small arrow shows the orientation of the sensor that gives the minimum uncertainty at this position. Observe that the map is not continuous, for instance around the two concave corners, because of changes in the visible edges. Figure 12 displays a map of the SUF values at sampled positions in the grid for a discretized set of sensor

[4]Recall from the beginning of Section 4 that configuration space obstacles are cylinders.

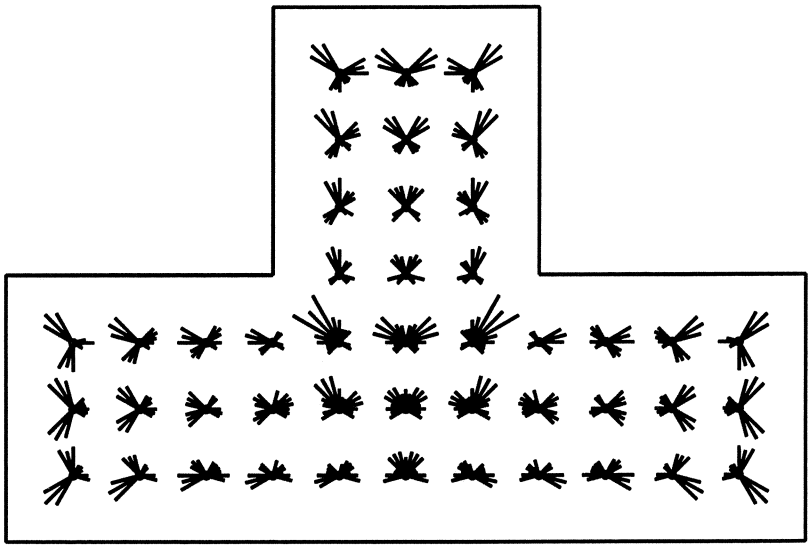


Figure 12: SUF values for several sensor orientations

orientations. The length of each small ray is proportional to the inverse value of the SUF; hence, the longer the ray, the lower the uncertainty.

The set of collision-free positions is discretized into a regular grid. A path ν is described as sequence $\{\boldsymbol{q}_i\}_{i=0,\ldots,s}$ of adjacent configurations in this grid. The functional $\mathcal{J}$ is computed as the discrete sum:

$$\mathcal{J}(\nu) = \sum_{i=1}^{s-1} \frac{1}{2}(\boldsymbol{F}(\boldsymbol{q}_i)^\gamma + \boldsymbol{F}(\boldsymbol{q}_{i+1})^\gamma)\, D(\boldsymbol{q}_i, \boldsymbol{q}_{i+1}),$$

where $D(\boldsymbol{q}_i, \boldsymbol{q}_j)$ is the distance between the two configurations $\boldsymbol{q}_i$ and $\boldsymbol{q}_j$. $\mathcal{J}(\nu)$ depends on both the length of the path ν and the sensing uncertainty along ν. The exponent γ allows us to weigh path reliability versus path length. The larger γ, the more important path reliability in $\mathcal{J}$. Hence, if we increase the value of γ we will typically get more reliable, but longer paths.

A path minimizing $\mathcal{J}$ is constructed by searching the configuration space grid starting at the initial configuration, using the Dijkstra algorithm, i.e., the A^* algorithm with the trivially admissible null heuristic function [1, 45]. This algorithm is guaranteed to return the optimal path. If the goal orientation is unspecified, the search is conducted with a goal set of configurations.

The successors of a configuration $\boldsymbol{q}_k = (X_k, Y_k, \Phi_k)$ generated at every

iteration of the search (i.e., the neighborhood of $\boldsymbol{q}_k$) are all the configurations $\boldsymbol{q}'_k = (X'_k, Y'_k, \Phi'_k) \neq \boldsymbol{q}_k$ in the grid such that $X'_k \in \{X_k - 1, X_k, X_k + 1\}$, and $Y'_k \in \{Y_k - 1, Y_k, Y_k + 1\}$ (we assume that every increment along all axes of the grid is normalized to 1). This definition allows the robot to rotate to any desired orientation at every step. The distance between two configurations $\boldsymbol{q}_i = (X_i, Y_i, \Phi_i)$ and $\boldsymbol{q}_j = (X_j, Y_j, \Phi_j)$ is computed as:

$$D(\boldsymbol{q}_i, \boldsymbol{q}_j) = \max(\sqrt{(X_j - X_i)^2 + (Y_j - Y_i)^2}, \frac{1}{\mu} |\theta_j - \theta_i|).$$

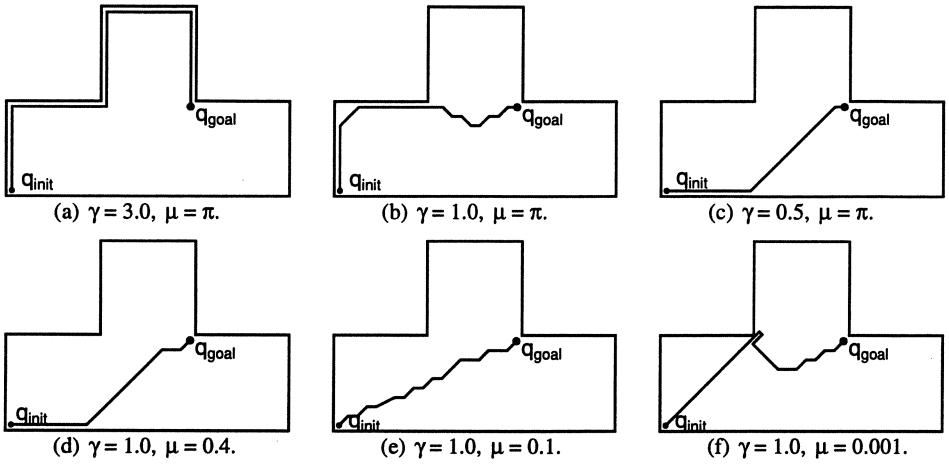
This definition of D corresponds to assuming that rotation has some cost attached to it. Indeed, large changes in orientation between two consecutive configurations may require the robot to slow down its translation. In the above definition, μ can be interpreted as the ratio of the robot's maximal speeds in rotation and translation. The larger μ, the faster the rotation.

5.2 Examples of Generated Paths

The computation of the SUF and the SUF-based planning method described in the previous sections have been implemented in a program written in C which runs on a DEC 5000 workstation. In this section we show paths computed by the planner. The tuning parameters γ and μ allow us to weigh path reliability versus path length, and to vary the ability of the sensor to rotate.

Figure 13 shows six paths constructed by the implemented planner for different values of γ and μ in the T-shaped environment of Figure 11. The first three paths are generated with a rather large constant $\mu = \pi$. Then the sensor can rotate to any orientation as the robot translates between two adjacent positions at maximum speed. Figure 13(a) is obtained with $\gamma = 3.0$. Then path reliability is important, and the planner generates a long path that follows environment edges and comes close to corners (remember that in these examples the robot is a point). Figure 13(c) is obtained with a small value of γ ($\gamma = 0.5$), and the length of the path is then more important than its reliability. (Since the search of a path is restricted to 8 directions in translation, the planned path is not just the straight line segment connecting the initial and the goal positions.) Figure 13(b) is obtained with an intermediate value of γ ($\gamma = 1.0$).

The next three paths are generated with a constant $\gamma = 1.0$. Figure 13(d) is obtained with $\mu = 0.4$. Then, while the robot translates be-



(a) $\gamma = 3.0$, $\mu = \pi$. (b) $\gamma = 1.0$, $\mu = \pi$. (c) $\gamma = 0.5$, $\mu = \pi$.

(d) $\gamma = 1.0$, $\mu = 0.4$. (e) $\gamma = 1.0$, $\mu = 0.1$. (f) $\gamma = 1.0$, $\mu = 0.001$.

Figure 13: Paths generated by the SUF-based planner

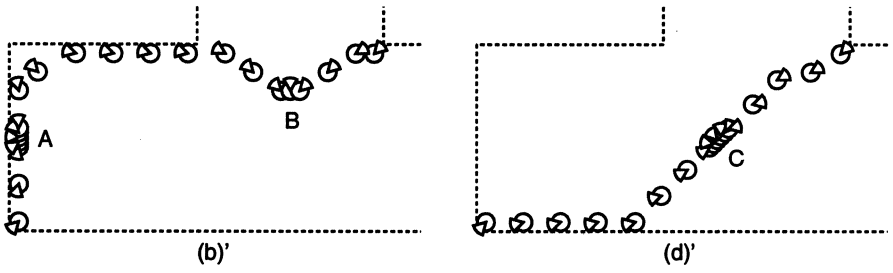


Figure 14: Influence of the rotation cost

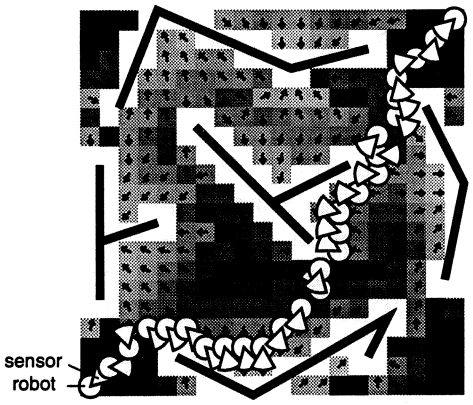
tween two adjacent positions at its maximum speed, the rotation is limited to $0.4\,rad = 22.9°$ for 4-neighboring positions and $0.4\sqrt{2}\,rad = 32.4°$ for diagonal neighboring positions. Figure 14(b)' and (d)' shows the robot (as a small circle) and the sensor scope (as a small cone) along the paths of Figure 13(b) and (d), respectively. In Figure 14(b)' the sensor rotates by almost 180° twice, along the portions of the path marked by A and B. Instead, in Figure 14(d)' only one such rotation occurs (portion marked by C). With $\gamma = 1.0$, the cost of the first path is lower when $\mu = \pi$; the cost of the second path is lower when $\mu = 0.4$.

Figure 13(e) is obtained with an even stronger constraint on the sensor rotation ($\mu = 0.1$). The path has the same length as in Figure 13(d), but it comes closer to the concave corner so that it can rotate at a more reliable point, at the expense of passing through less reliable points where it does not have to rotate. Figure 13(f) is an extreme case with a strong rotational constraint.

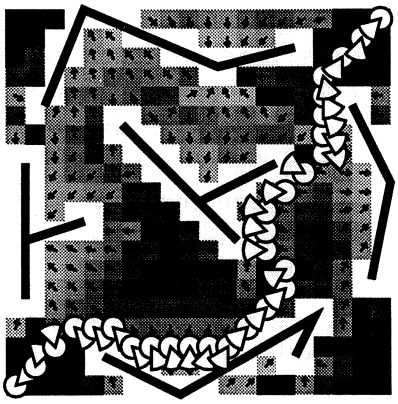
A more complicated example of SUF-based planning is shown in Figure 15. The environment contains multiple barriers with various directions and lengths. It is contained in a larger square environment (not shown in the figure) bounded by four walls aimed at making sure that the SUF has a definite value at every configuration. Three paths produced by the planner are shown in the figure. The values of γ and μ for each path are given in the figure. The most reliable path is displayed in Figure 15(c). But, because it closely follows more environment features than the other two, it is also much longer. The path of Figure 15(a) achieves a compromise between path length and reliability. The path of Figure 15(b) it is similar, but the cost of rotation along this path was set bigger than for the other two paths.

In the example of Figure 15, the grid was a 20×20 array and the orientation was discretized into 24 values. Computing $\boldsymbol{F}$ took 14.9 minutes for the entire configuration space, hence an average of 135 milliseconds per configuration. About 95% of the computation is spent in computing volume intersections (Figure 10). Each of the three paths was generated in 20 to 40 seconds.

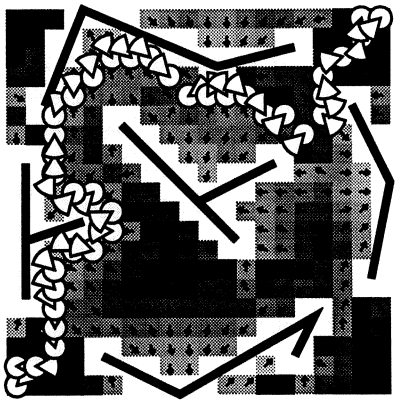
A simple variant of the planner is obtained by assuming that the robot can rotate to any orientation while it translates by one increment in the workspace at maximal translation speed. Then the search for a path can be conducted in the two-dimensional grid placed over the workspace (with the 8-neighborhood) by using the minimal values of $\boldsymbol{F}$ at every position. We ran



(a) $\gamma = 1.0$, $\mu = \pi$



(b) $\gamma = 1.0$, $\mu = 0.1\pi$



(c) $\gamma = 3.0$, $\mu = \pi$

Figure 15: SUF and generated paths in another environment

this variant with the example of Figure 15. The computation time is now of the order of 30 milliseconds. This reduction derives from the fact that every node (position) in the new search graph has at most 8 successors, while with the original method every node (configuration) had up to 215 successors. This result strongly suggests that the range sensor should be placed on a fast rotating turret whose orientation can be controlled separately from that of the robot.

6 Navigation Experiments

We experimented with paths generated by the SUF-based planner using two mobile robots equipped with a line-striping range sensor. The first robot is a prototype robot, called GOFER, described in [5]. The second robot is a NOMAD 200 robot of Nomadic Technologies. The mobile platforms of both robots have the same classical three wheel synchronous drive mechanical system allowing null turning radius, i.e., pure rotation about the center O. However, unlike NOMAD 200, GOFER is not equipped with an independent turret to orient the range sensor. We dealt with this limitation by stopping GOFER at every sensing operation and rotating the robot to point the range sensor along the direction suggested by the motion path.

We equipped the navigation system of both robots with a dynamic localization function (see Section 2) which we sketch below (a more detailed description of this function is given in [23]). A path is input as a sequence $\{\boldsymbol{q}_0, \boldsymbol{q}_1, \cdots\}$ of configurations called *viapoints*, starting at the initial configuration and ending at the goal one. The robot is placed at the initial configuration with sufficient precision to apply the dynamic localization function. (An alternative would be to compute the initial configuration using a static localization technique.) When the robot thinks it has reached the i^{th} viapoint $\boldsymbol{q}_i$ (actually, it is not exactly at $\boldsymbol{q}_i$), it acquires the data S_i provided by the range sensor, computes a first estimate $\hat{\boldsymbol{q}}_i^{(1)}$ of its current configuration (we will say in one instant how this is done), and performs a motion servoed through the odometric sensors (encoders) that is intended to go from $\hat{\boldsymbol{q}}_i^{(1)}$ to $\boldsymbol{q}_{i+1}$. During the time spent executing this motion, the data S_i are matched against the model M of the environment in order to generate a second (and better) estimate, $\hat{\boldsymbol{q}}_i^{(2)}$, of the robot's configuration at the i^{th} viapoint. When the robot thinks it has reached the $(i+1)^{\text{th}}$ viapoint, it generates the first

estimate $\hat{q}_{i+1}^{(1)}$ of its current configuration as:

$$\hat{q}_{i+1}^{(1)} = \hat{q}_i^{(2)} + (q_{i+1} - \hat{q}_i^{(1)}).$$

And so on.

Whenever a configuration estimate $q_i^{(k)}$ ($k = 1$ or 2) is computed, an uncertainty region is associated with it ($q_i^{(k)}$ is the center of this region). This region, which is assumed to contain the actual configuration of the robot at the i^{th} viapoint, is obtained by intersecting uncertainty regions previously associated with the data used to compute $q_i^{(k)}$. In our implementation, the uncertainty region for the robot's position is an ellipsoid; the uncertainty region for the orientation is an interval [23]. When two regions are intersected, the resulting region is approximated and given the same general geometric shape. Such data fusion techniques are described in detail in several papers, including [18, 23, 36, 48, 54].

Once the first estimate $\hat{q}_i^{(1)}$ of q_i and its uncertainty region have been computed, our localization function computes the second estimate $\hat{q}_i^{(2)}$ in two steps:

(1) A few (typically, 4 to 8) configurations evenly distributed in the uncertainty region of $\hat{q}_i^{(1)}$ are considered. For every such configuration, a visibility analysis is carried out on the model M assuming that the robot is exactly at this configuration. For every sensing ray of the sensor, the distance between the point of M that should be seen and the actual location of the sensed point is computed. The sum of these distances for all rays is used to characterize the quality of the match at each configuration considered in the uncertainty region of $\hat{q}_i^{(1)}$. The best match is defined as the one that munimizes this sum.

(2) A side-effect of the first step is to label every sensed point along a sensing ray by an edge of M. The labeling established for the best match is used. The points labeled by the same edge are grouped together and the line-fitting algorithm of Appendix A is applied to them. Comparing the equation of each estimated edge with the equation of the corresponding model edge yields an uncertainty region for the robot's configuration at the i^{th} viapoint. The regions obtained for all edges and the uncertainty region of $\hat{q}_i^{(1)}$ are intersected together to form the uncertainty region of $\hat{q}_i^{(2)}$. The estimate $\hat{q}_i^{(2)}$ is the centerpoint of this region.

Notice that these two steps compute $\hat{q}^{(2)}$ as a refinement of $\hat{q}_i^{(1)}$, i.e.,

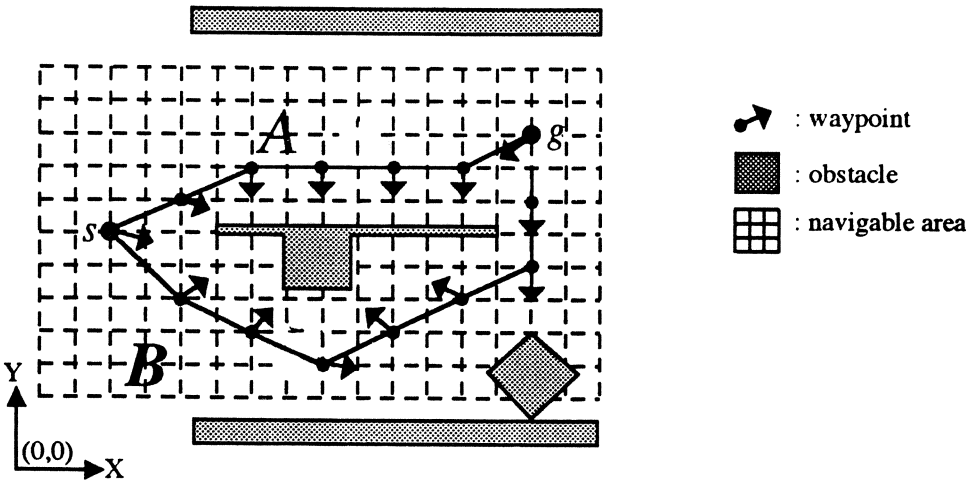


Figure 16: Schematic of an experimental setup

the uncertainty region of $\hat{q}_i^{(2)}$ is fully contained in the uncertainty region of $\hat{q}_i^{(1)}$. In the particular case where the robot senses no environment edge, the two region are the same. Moreover, since uncertainty caused by dead-reckoning cannot grow to infinity along a finite path, uncertainty regions remain always bounded.

Experiments have shown that this two-step localization is very robust. The first step eliminates the difficulties usually encountered in segmenting a sequence of sensed points to extract line segments. The labeling it creates also avoids most matching errors. Small errors may subsist close to corners (e.g., one point on one side of a corner may be assigned to the other side), but points falling too close to corners can easily be discarded in the first step.

The navigation system and its localization function are written in C and run on a SUN-3/80 workstation used as the host computer for the two robots in the experiments reported here. The average processing time for a localization cycle is about 1 second. A more recent implementation of this function on a 68040 processor runs in about 1/10 second.

Figure 16 represents an environment in which we carried out experiments. The workspace is a 15 × 10-square-foot rectangular surface (NOMAD 200 has a diameter of 21 inches). Two polygonal obstacles lie in this workspace. Two other obstacles (side walls) lie outside the workspace, but

1	0	4		6		5		5
2	2	3	4	5	5	4	6	5
3	1	3	4	4	5	4	4	5
4	1	2	3	4	4	4	5	4
5	2	0	3	2	4	3	3	4
6	2	0	3	1	2	2	3	3
7	1	0	2	2	2	3	5	4
8	1	1	2	3	4	4	4	4
9	1		2		4		4	

Table 1: Drift along path B using localization

can nevertheless be perceived by the range sensor. The figure shows two paths, A and B, between an initial configuration s and a goal configuration g. Both paths are defined as a sequence of viapoints at approximately the same resolution. Path B (9 viapoints) was generated using the SUF-based planner, while path A (7 viapoints) was constructed by hand.

Tables 1, 2, 3, and 4 show experimental data obtained for one set of experiments using the NOMAD 200 robot. Similar results were obtained with GOFER.

Table 1 records the drift in the robot position (magnitude of the position error, in inches) as it performed four round-trips along path B. The numbers in the first column denote the 9 viapoints along path B (the first viapoint being s and the last one being g). The $2i^{\text{th}}$ columns ($i = 1, 2, 3, 4$) contain the drift when the robot moves from s to g. The $(2i+1)^{\text{th}}$ columns ($i = 1, 2, 3, 4$) shows the drift when the robot returns from g to s. Column 2 should be read downward, column 3 upward, column 4 downward, etc. Drifts were measured by hand. The robot was accurately positioned at the initial configuration. After some time, the drift stabilized to a few inches. A detailed interpretation of the drift values is difficult. Indeed, some localization errors were increased by the fact that the floor (made of wood) was not really flat, leading the range sensor to measure distances in a slightly non-horizontal plane.

Table 2 records the drift of the robot along path A. The motion terminated at the end of the third round-trip before reaching viapoint 2, when

1	0	1		4		
2	0	1	2	3	3	†
3	0	1	1	2	3	6
4	1	1	1	2	4	5
5	1	1	1	3	4	4
6	1	2	2	3	3	3
7	2		3		3	

Table 2: Drift along path A using localization

1	0	5		9		16		20
2	0	4	5	8	9	14	15	17
3	0	2	4	7	8	10	13	16
4	1	4	3	7	7	9	13	13
5	1	4	5	6	6	6	11	8
6	3	5	5	6	7	6	9	9
7	4	5	6	6	7	7	7	8
8	5	4	5	5	8	6	8	8
9	5		4		8		9	

Table 3: Drift along path B without localization

it collided with an obstacle (†). The table shows that the drift tends to increase monotonically (actually, some other experiments showed that trend even more neatly). The reason is that path A never allows the robot to see two non-parallel edges.

Tables 3 and 4 record the drift of the robot along paths B and A, respectively, when the robot relies on dead-reckoning only. The drift then grows much faster than in Tables 1 and 2.

Remark: Why is the drift in Table 2 smaller than in Table 4? One reason is that the visible edge allows the dynamic localization function to periodically reduce the position error along the direction perpendicular to this edge (the y-axis of the world coordinate system). However, the main reason for that is elsewhere: The visible edge also allows reducing the error

1	0	5		
2	1	5	6	†
3	2	6	6	8
4	2	5	7	8
5	3	6	8	9
6	4	6	8	9
7	6		9	

Table 4: Drift along path A without localization

in the orientation of the robot's wheels. The kinematics of NOMAD makes this error the leading cause of error in the position estimate computed by dead-reckoning (each motion of the robot is indeed a translation along the direction pointed by the wheels). Hence, while reducing the error in the orientation of the wheels has no direct effect on the second position estimate (the one computed by the localization function), it reduces the error in the first position estimate (the computed by dead-reckoning). If our dynamic localization function did not reduce wheel orientation errors, the comparison between Table 1 and Table 2 would be much more favorable to the path generated by suf-based planner.

In general our experiments showed that paths generated by our planner can be tracked with good precision using dynamic localization, yielding reliable navigation. The tracking precision for other paths depends of course on the geometry of these paths relative to the environment. But the SUF-based planner consistently produces paths of better or similar quality. In particular, it avoids very poor paths (with respect to sensing), when other paths are possible.

7 Discussion

In this section we briefly discuss further research issues that have not been addressed above.

Accuracy of environment model. We assumed that the input environment model is accurate. In reality, such a model is at best correct within

some tolerance. In the case of our planar range sensor, one way to deal with errors in the input model would consist of taking them into account in the probability distributions of the random variables $\mathcal{R}_i$ used in Equation (1). But the $\mathcal{R}_i$'s would no longer be independent, subsequently yielding more complicated computation. Our experience, however, is that for many environment (especially, indoor environments), errors in the input model can be neglected. For example, in our experiments they where less than one inch.

Completeness of environment model. We also assumed that the environment model is complete. In a realistic setting the robot will encounter unexpected objects at execution time. For example, in an office environment, such objects are easily movable objects, e.g., chairs and small tables. These objects may hide expected environment edges from the sensor. There seems to be no simple way to deal with this issue at planning time since, by definition, unexpected objects are unknown at that time. Actually, if all environment features were hidden, no localization would be possible other than pure dead-reckoning. If unexpected objects are relatively sparse, a path generated by the SUF-based planner is likely to offer enough features to the sensors to allow safe tracking even if some of these features turn out not visible. In any case, on the average, such a path will certainly be better than a path generated without consideration for the environment features.

Another issue to consider is that unexpected obstacles hidding environment features may result in matching errors, yielding large localization errors. However, our experience with the implemented dynamic localization function is that matching errors can be avoided for the most part by using the computed uncertainty regions to eliminate blatantly inconsistent sensory data. This is done as follows: At step (1) of the function (see Section 6), whenever we associate a point in the model to a sensed point, we verify that the distance between the two is compatible with the current uncertainty; if it is not, the sensed point is discarded.

Reliability of sensor. We assumed that the sensor is quite reliable. But an actual sensor may fail to detect edges, or detect spurious edges, or generate incorrect matches between detected edges and the input world model. The notion of reliability is different from that of precision, suggesting that a "sensory (un)reliability field" should be introduced to complement the sensory uncertainty field. More sophisticated sensor models have been developed in the literature (e.g., see the model of a sonar range sensor proposed in [43]). However, it is still not clear how such models can be used at planning

time.

In our experiments, the combination of a line-striping range sensor, clear obstacles, and a two-step dynamic localization function turned out very reliable, and the above issue did not arise. Other environments (e.g., outdoor environments with bushes and trees) and other sensors (e.g., a more general vision sensor) would probably not be so favorable.

Errors in robot control. The planner computes a path by minimizing a cost function. However, errors in control lead the robot to depart from this path and follow another path along which the SUF may be slightly different (especially at configurations where the SUF varies sharply). One way to alleviate this problem is to average the value of the function $\boldsymbol{F}$ at a configuration with its values at neighboring configurations. Another way is to avoid regions with small expected errors if they are too close of regions with high expected errors. Preimage backchaining (see Section 2), which explicitly models control uncertainty, seems to address this issue in a neater way.

Dealing with gaps in accurate sensing. Areas where sensing is inaccurate may considerably affect the generated path, while, if they are not too large, they could be traversed safely using dead-reckoning. One way to deal with this issue is to combine the methods of this paper with preimage backchaining. As suggested in [34], the magnitude of the SUF could be thresholded to construct "islands" (called landmark areas in [34]) in which sensing and control are very precise. Preimage backchaining would then be used to connect the goal to the initial configuration through these islands assuming some bounded (but possibly large) control errors outside the islands. A simple variant of this approach would consist of thresholding the SUF, building the connected regions with low expected errors, discarding the small regions, and planning a path as a sequence of motions through centers of mass of remaining regions.

Tracking walls. Our planning method favors areas where sensing is accurate in all directions. This is a drawback when the workspace contains long walls. Indeed, a wall can be tracked reliably, from one end to the other, even if position sensing along the direction parallel to the wall is not good. This requires reasoning about individual parameters of the robot's configuration and about the fact that it does not matter that the localization error along a direction grows large, provided that a feature will eventually appears to

reduce this error. By decomposing the workspace into areas, according to the number of edges that can be seen by the sensor, the planning method of [42] allows wall tracking, suggesting a similar extension to our method.

8 Conclusion

In this paper we have described a new approach to motion planning with uncertainty. This approach consists of estimating the uncertainty in the configuration that will be computed by the robot sensors, in the form of a sensory uncertainty field (SUF), and using this field to compute a path that minimizes a function combining expected errors and path length. The approach was implemented in a planner and experiments have been conducted with mobile robots equipped with a horizontal line-striping range sensor. Experiments with the planner alone show that very different paths can be generated depending on how we weigh path reliability versus path length. The speed of rotation relative to translation is another important factor determining the shape of the generated paths. Experiments with real robots showed that paths produced by the SUF-based planner can usually be tracked with higher precision than other paths. We recognize, however, that the success of our experiments was partially due to good-quality sensing (laser line-striping with clear obstacles). Although all issues raised by the SUF-based planning approach have not been completely explored, we believe that it provides a reasonable and practical alternative to previously proposed motion planning approaches in the mobile robotic context. As discussed in Section 7, it could also greatly benefit from being combined with other, separately developed techniques.

Appendix A: Line-Fitting Model

Consider Figure 1 discussed in Section 4.1. The robot can use the coordinates of the $2n+1$ sensed points P_i to fit a line ℓ' through them and estimate the orientation $\phi = \phi_0$ of the sensor and the distance d from the center O to the environment line ℓ. The uncertainty in these estimates depends to some extent on the particular line-fitting algorithm that is used to compute ℓ'. In our planner, the SUF is computed assuming the following eigenvector algorithm [16]. This algorithm generates a line ℓ' that minimizes the sum of the squared perpendicular distances from the points P_i to ℓ'.

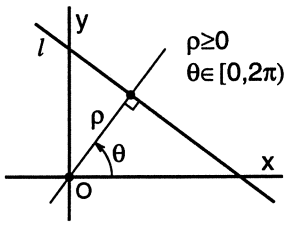


Figure 17: Parameterization of a line

Let us express the equation of the best-fit line ℓ' as:

$$x\cos\theta + y\sin\theta - \rho = 0$$

with $\rho > 0$ and $\theta \in [0, 2\pi)$ defined as shown in Figure 17. Then the perpendicular distance ε_i from P_i to ℓ' is:

$$\varepsilon_i = |x_i\cos\theta + y_i\sin\theta - \rho|.$$

In order for the sum of the squared distances, $E = \sum_i \varepsilon_i^2$, to be minimum, we must have:

$$\rho = \sum_i (x_i\cos\theta + y_i\sin\theta)/(2n+1).$$

Hence, the equation of ℓ' can be expressed as:

$$(x - \bar{x})\cos\theta + (y - \bar{y})\sin\theta = 0,$$

with:

$$\begin{cases} \bar{x} &= \sum_i x_i/(2n+1), \\ \bar{y} &= \sum_i y_i/(2n+1), \end{cases}$$

which makes explicit that ℓ' passes through the mean of the sensed points (x_i, y_i).

To establish the equation of ℓ' we still have to determine θ. Let:

$$\boldsymbol{v}_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \begin{bmatrix} \bar{x} \\ \bar{y} \end{bmatrix} \quad \text{and} \quad \boldsymbol{w} = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}.$$

Then the sum of the squared distances can be written as:

$$E = \sum_i (\boldsymbol{w}^{tr}\boldsymbol{v}_i)^2 = \boldsymbol{w}^{tr}\left(\sum_i \boldsymbol{v}_i\boldsymbol{v}_i^{tr}\right)\boldsymbol{w}.$$

We let S denote the symmetric matrix $\sum_i \boldsymbol{v}_i \boldsymbol{v}_i^{tr}$.

The vector $\boldsymbol{w}$ which minimizes the quadratic form $\boldsymbol{w}^{tr} S \boldsymbol{w}$ subject to $\|\boldsymbol{w}\| = 1$ is the eigenvector of S associated with the smallest eigenvalue [16].

S is of the form:

$$S = \begin{bmatrix} s_1 & s_3 \\ s_3 & s_2 \end{bmatrix},$$

where

$$\begin{cases} s_1 &= \sum (x_i - \bar{x})^2, \\ s_2 &= \sum (y_i - \bar{y})^2, \\ s_3 &= \sum (x_i - \bar{x})(y_i - \bar{y}). \end{cases}$$

Assume that $s_3 \neq 0$ (then, both s_1 and s_2 are also non-zero). Then the smallest eigenvalue of S is:

$$\lambda_0 = \frac{s_1 + s_2 - \sqrt{(s_1 - s_2)^2 + 4s_3^2}}{2}.$$

The associated eigenvector $[w_1, w_2]^{tr}$ satisfies:

$$s_1 w_1 + s_3 w_2 = \lambda_0 w_1.$$

Since we assumed $s_3 \neq 0$, we derive:

$$\begin{aligned} \theta &= \tan^{-1}\left(\frac{w_2}{w_1}\right) \\ &= \tan^{-1}\left(\frac{s_2 - s_1 - \sqrt{(s_2 - s_1)^2 + 4s_3^2}}{2s_3}\right). \end{aligned}$$

If $s_3 = 0$ and $s_1 = 0$, then all the x_i's have the same value $\bar{x}$; hence, we have $\theta = 0$. Similarly, if $s_3 = 0$ and $s_2 = 0$, we have $\theta = \pi/2$. If $s_3 = 0$, $s_1 \neq 0$, and $s_2 \neq 0$, the points P_i are evenly distributed around their mean, and we cannot infer a best value of θ. By construction, the case where s_1 and s_2 are simultaneously zero is impossible.

From the above equations, we obtain the following estimates $\phi^\star$ and $d^\star$ of ϕ and d:

$$\begin{aligned} \phi^\star &= \arg(x_0, y_0) - \theta, \\ d^\star &= \rho, \end{aligned}$$

where $\arg(x, y)$ denotes the argument of the complex number $z = x + jy$.

The errors in these estimates are:

$$\begin{aligned} \delta\phi &= \phi - \phi^\star &&= \theta, \\ \delta d &= d - d^\star &&= d - |\bar{x}\cos\theta + \bar{y}\sin\theta|. \end{aligned}$$

Appendix B: Properties of the Function $u_{\phi,d,n}$

From the assumptions on the random variables $\mathcal{R}_i$ stated in Section 4.1 and the equations defining $\delta\phi$ and δd given above, we derive that:

$$\begin{aligned} u_{-\phi,d,n}(\delta\phi, \delta d) &= u_{\phi,d,n}(-\delta\phi, \delta d), \\ u_{\phi,d,n}(\delta\phi, \delta d) &= u_{\phi,1,n}(\delta\phi, (\delta d)/d), \end{aligned}$$

These relations allow us to precompute and tabulate $f_{\phi,d,n}(A)$ (see Figure 5) for positive values of ϕ and $d = 1$, and use the constructed table to compute $f_{\phi,d,n}(A)$ for other values of ϕ and d (see Section 4.2). The first of the above two relations is obvious. We prove the second one below.

Let us consider two sets of $2n+1$ points, $\{(x_i, y_i)\}$ and $\{(X_i, Y_i)\}$, such that:

$$\begin{aligned} x_i &= 1 + r_i, \\ y_i &= x_i \tan\phi_i, \end{aligned}$$

and

$$\begin{aligned} X_i &= d \cdot x_i, \\ Y_i &= d \cdot y_i. \end{aligned}$$

In the following, $\bar{X}$, $\bar{Y}$, S_1, S_2, S_3, Θ, $\delta\Phi$, and δD have the same definition as $\bar{x}$, $\bar{y}$, s_1, s_2, s_3, θ, $\delta\phi$, and δd, except that they apply to the set $\{(X_i, Y_i)\}$, instead of $\{(x_i, y_i)\}$. We have:

$$\begin{aligned} \bar{X} &= d \cdot \bar{x}, \\ \bar{Y} &= d \cdot \bar{y}, \\ S_i &= d^2 \cdot s_i. \end{aligned}$$

Thus:

$$\begin{aligned} \delta\Phi &= \Theta \\ &= \tan^{-1}\left(\frac{S_2 - S_1 - \sqrt{(S_2 - S_1)^2 + 4S_3^2}}{2S_3}\right), \end{aligned}$$

which yields:

$$\delta\Phi = \delta\phi.$$

On the other hand, we have:

$$\begin{aligned} \delta D &= d - |\bar{X}\cos\Theta + \bar{Y}\sin\Theta| \\ &= d \cdot (1 - |\bar{x}\cos\theta + \bar{y}\sin\theta|) \\ &= d \cdot (\delta d). \end{aligned}$$

Therefore:

$$u_{\phi,d,n}(\delta\phi, \delta d) = u_{\phi,1,n}(\delta\phi, (\delta d)/d).$$

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *Data Structures and Algorithms*, Addison-Wesley, Reading, MA, 1983.

[2] N. Ayache, *Artificial Vision for Mobile Robots: Stereo Vision and Multisensory Perception*, The MIT Press, Cambridge, MA, 1991.

[3] B. Bhanu and O.D. Faugeras, "Shape Matching of Two-Dimensional Objects," *IEEE Tr. on Pattern Analysis and Machine Intelligence*, 6(2), 1984, pp. 137-156.

[4] S.J. Buckley, *Planning and Teaching Compliant Motion Strategies*, Ph.D. Dissertation, Department of Electrical Engineering and Computer Science, MIT, Cambridge, MA (1986).

[5] P. Caloud, W. Choi, J.C. Latombe, C. Le Pape, and M. Yim, "Indoor Automation with Many Mobile Robots," *Proc. of IEEE Int. Workshop on Intelligent Robots and Systems*, Tsuchiura, Japan, 1990, pp. 67-72.

[6] J.F. Canny, "On Computability of Fine Motion Plans," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 177-182.

[7] R. Chatila and J.P. Laumond, "Position Referencing and Consistent World Modeling for Mobile Robots," *Proc. of IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, 1985, pp. 138-145.

[8] W. Choi and J.C. Latombe, "A Reactive Architecture for Planning and Executing Robot Motions with Incomplete Knowledge," *Proc. of IEEE/RSJ Int. Workshop on Intelligent Robots and Systems*, Osaka, 1991, pp. 24-29.

[9] C.K. Cowan and P.D. Kovesi, "Automatic Sensor Placement from Vision Task Requirements", *IEEE Tr. on Pattern Analysis and Machine Intelligence*, 10(3), 1988, pp. 407-416.

[10] J.L. Crowley, "World Modeling and Position Estimation for a Mobile Robot Using Ultrasonic Ranging," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Scottsdale, AZ, 1989, pp. 674-680.

[11] R.S. Desai, *On Fine Motion in Mechanical Assembly in Presence of Uncertainty*, Ph.D. Dissertation, Dept. of Mechanical Engineering, University of Michigan, 1988.

[12] B.R. Donald, "A Geometric Approach to Error Detection and Recovery for Robot Motion Planning with Uncertainty," *Artificial Intelligence J.*, 37(1-3), 1988, pp. 223-271.

[13] B. Donald and J. Jennings, "Sensor Interpretation and Task-Directed Planning Using Perceptual Equivalence Classes," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, 1991, pp. 190-197.

[14] B. Donald and J. Jennings with contributions from R. Brown, "Constructive Recognizability for Task-Directed Robot Programming," *J. Robotics and Autonomous Systems*, 9, 1992, pp. 41-74.

[15] M. Drumheller, *Mobile Robot Localization Using Sonar*, Tech. Rep. 826, AI Lab., MIT, 1985.

[16] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, New York, 1973.

[17] B. Dufay and J.C. Latombe, "An Approach to Automatic Robot Programming Based on Inductive Learning," *The Int. J. of Robotics Research*, 3(4), 1984, pp. 3-20.

[18] H. Durrant-Whyte, "Concerning Uncertain Geometry in Robotics," *Artificial Intelligence J.*, 37, 1986.

[19] A. Elfes, "Dynamic Control of Robot Perception Using Multi-Property Inference Grids," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Nice, France, 1992, pp. 2561–2567.

[20] M.E. Erdmann, *On Motion Planning with Uncertainty*, Tech. Rep. 810, AI Lab., MIT, 1984.

[21] M.E. Erdmann, *On Probabilistic Strategies for Robot Tasks*, Ph.D. Dissertation, Tech. Rep. 1155, AI Lab., MIT, Cambridge, MA, 1990.

[22] M.E. Erdmann and M.T. Mason, "An Exploration of Sensorless Manipulation," *Proc. of IEEE Int. Conf. on Robotics and Automation*, San Francisco, 1986, pp. 1569-1574.

[23] C. Facchinetti, "Motion Planning and Control with Uncertainty While Sensing the Environment," *Proc. of Swiss Vision Conf.*, Zurich, Switzerland, September 1993, pp. 45–52. Also published in *Proc. of ICSPAT'93*, Santa Clara, CA, September 1993.

[24] A. Fox and S. Hutchinson, *Exploiting Visual Constraints in the Synthesis of Uncertainty-Tolerant Motion Plans*, Tech. Rep . UIUC-BI-AI-RCV-92-05, The Un. of Illinois at Urbana-Champaign, October 1992.

[25] S.N. Gottschlich and A.C. Kak, "Dealing with Uncertainties in CAD-Based Assembly Motion Planning," *Proc. of 9th Nat. Conf. on Artificial Intelligence*, AAAI Press, 1991, pp. 646-652.

[26] W.E.L. Grimson and T. Lozano-Pérez, "Localizing Overlapping Parts by Searching the Interpretation Tree," *IEEE Tr. on Pattern Analysis and Machine Intelligence*, 9(4), 1987, pp. 469-482.

[27] L. Guibas, R. Motwani, and M. Raghavan, "The Robot Localization Problem in Two Dimensions," *Proc. of Symp. of Discrete Algorithms (SODA)*, 1991, pp. 259-268.

[28] S. Hutchinson, "Exploiting Visual Constraints in Robot Motion Planning," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, 1991, pp. 1722-1727.

[29] D.J. Kriegman, E. Triendl, and T.O. Binford, "Stereo Vision and Navigation in Buildings for Mobile Robots," *IEEE Tr. on Robotics and Automation*, 5(6), 1989, 792-803.

[30] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic Publishers, Boston, MA, 1991.

[31] J.C. Latombe, A. Lazanas and S. Shekhar, "Robot Motion Planning with Uncertainty in Control and Sensing," *Artificial Intelligence J.*, 52(1), 1991, pp. 1-47.

[32] C. Laugier and P. Théveneau, "Planning Ssensor-Based Motions for Part-Mating Using Geometric Reasoning Techniques," *Proc. of the European Conf. on Artificial Intelligence*, Brighton, UK, 1986.

[33] A. Lazanas and J.C. Latombe, "Landmark-Based Robot Navigation," *Proc. of the 10th Nat. Conf. on Artificial Intelligence*, San Jose, CA, 1992.

[34] A. Lazanas and J.C. Latombe, *Landmark-Based Robot Navigation*, Technical Report, Department of Computer Science, Stanford University, 1992.

[35] J.J. Leonard and H.F. Durrant-Whyte, "Mobile Robot Localization by Tracking Geometric Beacons," *IEEE Tr. on Robotics and Automation*, 7(3), 1991, pp. 376-382.

[36] J.J. Leonard, H.F. Durrant-Whyte, and I.J. Cox, "Dynamic Map Building for an Autonomous Mobile Robot," *Int. J. of Robotics Research*, 11(4), 1992, pp. 286–298.

[37] T.S. Levitt, D. T. Lawton, D.M. Chelberg, and P.C. Nelson, "Qualitative Navigation," *Proc. of the DARPA Image Understanding Workshop*, Los Angeles, CA, 1987, pp. 447-465.

[38] T. Lozano-Pérez, *The Design of a Mechanical Assembly System*, Tech. Rep. AI-TR 397, AI Lab., MIT, Cambridge, MA, 1976.

[39] T. Lozano-Pérez, M.T. Mason, and R.H. Taylor, "Automatic Synthesis of Fine-Motion Strategies for Robots," *The Int. J. of Robotics Research*, 3(1), 1984, pp. 3-24.

[40] V. Lumelsky and T. Skewis, "Incorporating Range Sensing in the Robot Navigation Function," *IEEE Tr. on Systems, Man, and Cybernetics*, 20(5), 1990, pp. 1058-1069.

[41] S. Mahadevan and J. Connell, *Automatic Programming of Behavior-based Robots using Reinforcement Learning*, Research Rep., IBM T.J. Watson Research Center, Yorktown Heights, NY, 1990.

[42] D. Miller, "A Spatial Representation System for Mobile Robots," *Proc. of IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, 1985, pp. 122-127.

[43] H. Moravec and A. Elfes, "High Resolution Maps from Wide Angle Sonar," *Proc. of IEEE Int. Conf. on Robotics and Automation*, St. Louis, MO, 1985, pp. 116-121.

[44] Y. Nakamura and Y. Xu, "Geometrical Fusion Method for Multi-Sensor Robotic Systems," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Scottsdale. AZ, 1989, pp. 668-673.

[45] N.J. Nilsson, *Principles of Artificial Intelligence*, Morgan Kaufmann, Los Altos, CA, 1980.

[46] J. Pertin-Troccaz and P. Puget, "Dealing with Uncertainty in Robot Planning Using Program Proving Techniques," *Robotics Research 4*, The MIT Press, 1988, pp. 455-466.

[47] Y. Roth, A.A. Wu, R.H. Arpaci, T. Weymouth, and R. Jain, "Model-Driven Pose Correction," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Nice, France, 1992, pp. 2625–2630.

[48] A. Sabater and F. Thomas, "Set Membership Approach to the Propagation of Uncertainty Geometric Information," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, 1991, 2718–2723.

[49] H. Takeda and J.C. Latombe, "Sensory Uncertainty Field for Mobile Robot Navigation," *Proc. of IEEE Int. Conf. on Robotics and Automation*, Nice, France, 1992.

[50] H. Takeda and J.C. Latombe, *Planning the Motions of a Mobile Robot in a Sensory Uncertainty Field*, Rep. No. STAN-CS-92-1424, Dept. of Computer Science, Stanford University, April 1992.

[51] K. Tarabanis, R.Y. Tsai and P.K. Allen, "Automated Sensor Planning for Robotic Vision Tasks", *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, USA, 1991, pp. 76-82.

[52] R.H. Taylor, *Synthesis of Manipulator Control Programs from Task-Level Specifications*, Ph.D. Dissertation, Dept. of Computer Science, Stanford University, 1976.

[53] S. Xie, "View Planning for Mobile Robots", *Proc. of IEEE Int. Conf. on Robotics and Automation*, Cincinnati, OH, 1990, pp. 748-754.

[54] Z. Zhang and O. Faugeras, "A 3D World Model Builder with a Mobile Robot," *Int. J. of Robotics Research*, 11(4), 1992, pp. 269–285.

[55] J.Y. Zheng, M. Barth and S. Tsuji, "Autonomous Landmark Selection for Route Recognition by a Mobile Robot", *Proc. of IEEE Int. Conf. on Robotics and Automation*, Sacramento, CA, USA, 1991, pp. 2004-2009.