Learning to Select Useful Landmarks

Russell Greiner and Ramana Isukapalli

Abstract

To navigate effectively, an autonomous agent must be able to quickly and accurately determine its current location. Given an initial estimate of its position (perhaps based on dead-reckoning) and an image taken of a known environment, our agent first attempts to locate a set of landmarks (real-world objects at known locations), then uses their angular separation to obtain an improved estimate of its current position. Unfortunately, some landmarks may not be visible, or worse, may be confused with other landmarks, resulting in both time wasted in searching for undetected landmarks, and in further errors in the agent's estimate of its position. To address these problems, we propose a method that uses previous experiences to learn a selection function that, given the set of landmarks that might be visible, returns the subset that can be used to reliably provide an accurate registration of the agent's position. We use statistical techniques to prove that the learned selection function is, with high probability, effectively at a local optimum in the space of such functions. This report also presents empirical evidence, using real-world data, that demonstrate the effectiveness of our approach.

Keywords

autonomous navigation, probably-approximately-correct learning, position estimation, landmark selection

I. Introduction

To navigate effectively, an autonomous agent R must be able to quickly and accurately determine its current location. R can usually obtain fairly accurate estimates of its position using dead-reckoning; unfortunately, the errors in these estimates accumulate over long distances, which can lead to unacceptable performance (read "bumping into walls" or "locating the wrong office"). An obvious way to reduce this problem is to observe the environment, and use the information in these observations to improve our estimate of R's position; *cf.*, the works using Kalman filters [18], [6] and other techniques [29], [20], [10], [9]. Our agent models the environment using only a set of "landmarks", each a (potentially visible) real-world object at a known location; these objects could be doors, corners and pictures when specifying the hallways within building, or major buildings, junctions and prominent signs when specifying the streets within a city. Then, given an initial estimate of its position (perhaps based on dead-reckoning) and an image taken of a known environment, R first attempts to locate a set of possibly visible landmarks, then uses their angular separation to obtain an improved estimate of its current position.

Landmark-based position estimation is a popular technique in robot navigation; *cf.*, [5], [32], [31], [22] and many others. Many of these landmark-based methods assume that all landmarks can be found reliably. Unfortunately, some landmarks may not be visible; for example, certain corners may always be in shadow and so be difficult to see, or some hanging pictures may have been removed after the floor-plan was given to the agent. These can force R to waste time, searching in vain for invisible landmarks. Worse, some landmarks may be easily confused with others; *e.g.*, door A may be mistaken for door B, or some landmark A (say the convex corner of a wall) may be occluded by another object B (say the convex corner of filing cabinet) that looks sufficiently similar that R might think that B is A; see Figure 1. As this can cause R to believe that A is located at B's position, these mis-identified objects can produce further errors in R's estimate of its position. Finally, R will use a *set* of identified landmarks to locate its position; depending on the geometric positions of these landmarks, small errors in landmark location may lead to very large errors in R's positional estimate. We of course prefer landmark sets that provide position estimates that are relatively insensitive to such errors in landmark location.

This is an extended version of a paper that appeared in the *Proceedings of the Twelfth Annual National Conference on Artificial Intelligence (AAAI94)*, Seattle, August 1994.

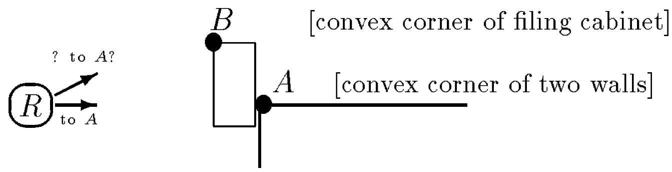

Fig. 1. Problem with Occluding Objects: The reported angle from agent R to landmark A (upper vector from R) is incorrect.

It therefore makes sense to search for only the *subset* of the potentially visible landmarks that can be found reliably, cannot be confused with others, etc. Unfortunately, it can be very difficult to determine this good subset *a priori*, as (1) the landmarks that are good for one set of R-positions can be bad for another; (2) the decision to seek a landmark can depend on many difficult-to-incorporate factors, such as lighting conditions and building shape; and (3) the reliability of a landmark can also depend on unpredictable events; *e.g.*, exactly where R happens to be when it observes its environment, how the building has changed after the floor-plan was finalized (*e.g.*, whether new cubicles were installed, or new pictures hung), and whether objects (perhaps people) are moving around the area where R is looking [33]. These factors make it difficult, if not impossible, to designate the set of good landmarks ahead of time — *i.e.*, to "engineer" the solution.

This report provides a way around this problem. Section II describes a method that learns a good "selection function" that, given the set of landmarks that may be visible, returns the subset that can usually be found correctly. It also uses statistical techniques to prove that this learned selection function is, with high probability, effectively at a local optimum in the space of such functions. Section III presents a corpus of empirical results which demonstrate that the associated LearnSF algorithm works effectively, and discusses its (in)sensitivities to various parameters. Section IV concludes by discussing other applications, within the scope of robot navigation, that can use a similar learning algorithm. We first close this section by presenting a survey of some of the relevant literature.

A. Literature Survey

We noted above how our research relates to some other ways of using observations to estimate an agent's location, including works using Kalman filters and/or landmarks. This subsection compares our approach with yet other research topics: First, we use the term "landmark" to refer to a real-world object at a known location; this usage contrasts with others, that view a landmark as a location *of the sensing agent* [7], or as a "sensory state" [23]. We also assume that the set of possible landmarks is provided initially; this is reasonable for our navigation task, as these landmarks are often required to describe the navigation task itself, *e.g.*, to specify the destination or some required intermediate points. By contrast, some other systems also attempt to learn from observations the significant features of various locations, which may correspond to the set of landmarks; *cf.*, [19], [23][1] and others. Notice that our landmark-*selection* approach (embodied in LearnSF) is complementary to those landmark-*acquiring* systems: while those other systems acquire a set of landmarks that seem useful in some situations and so *may* be useful in general, our LearnSF algorithm can then identify which of these landmarks are *truly* useful, over the agent's overall distribution of situations. Hence, there are obvious reasons to consider combining both algorithms.

Yamauchi and Beer [33] designed a system that can cope with an environment that can change in various ways, including topological changes (*e.g.*, rearranged furniture and doors that open and close), and transient changes (*e.g.*, people moving). That work, in essence, learns how to deal with each

[1] As one example, Mataric [23] presents a model, motivated by the organization and function of the rat hippocampus, that allows a robot, equipped with sonar sensors and a compass, to learn a map by boundary tracing and "landmark" detection. Its focus is primarily comparative: it presents biological, psychological and neurobiological data and compares the physical rat hippocampus with the synthetic rat implementation, drawing analogies between the two in terms of both "what" information is encoded in map learning and "how" it is encoded.

individual landmark. We could modify our LearnSF system to similarly deal with each individual landmark — *i.e.*, learn to ignore a *particular* landmark ℓ_i if it cannot be found reliably, perhaps because the object has moved from its initial map position (topological changes), or because people are often blocking the agent's view of ℓ_i (transient changes), etc. Our LearnSF system, however, deals with general categories of landmarks (*e.g.*, all corners, or all landmarks of a certain size and at a certain distance), as this means fewer sample images are required to produce statistically appropriate decisions.

Notice also that our task, of identifying a good subset of landmarks, has some similarities to the techniques of robust analysis and outlier removal [16]. Such techniques, however, first accumulate all of the possible information, and then remove or de-emphasize certain individual components. By contrast, as our performance system (which is providing navigational information to the agent) is also concerned with efficiency, it will simply not collect the problematic information; that is, LearnSF will produce a selection function which tells the performance system to *not even seek* certain landmarks.

Finally, our overall LearnSF system is an instance of the very general "probabilistic hill-climbing" learning algorithm [11], as it uses a set of "experiences" (each an image labeled with actual agent position, etc.) to climb in a discrete space of "performance elements" (here, the space of individual selection functions) seeking one whose expected "utility" is optimal (here, whose expected error is minimal). (This learning/performance dichotomy appears in [30], and the view that learning corresponds to improving the performance of a performance system on its performance task, discussed in [28] and elsewhere, is the basis for learning systems that range from standard decision tree learners like C4.5 [26] and CART [3] that seek the decision tree whose expected accuracy is maximal, through speed-up learning systems [25], [21] that seek a set of macros that yield an optimally efficient program, to neural net learners [27], [24], [14] that seek a setting of the weights that produces an optimal classifier, etc.) Moreover, LearnSF qualifies as a "wrapper learning system" [17], [4], as it views its "performance elements" (the individual selection functions) as black boxes, whose behavior can be sampled, but whose internals are unavailable. Notice this makes it fairly easy to adapt the LearnSF system to work with other types of performance elements, in other contexts; see Section IV-A.

II. Function for Selecting Good Landmarks

A. Performance Task: Position Estimation

The current RatBOT system [13] maintains estimates $\hat{\mathbf{x}}$ ($\hat{\sigma}$) of its current position $\mathbf{x}$ (uncertainty, σ). It uses two algorithms when computing these values:

- `LMs( x )`, which specifies the subset of the landmarks that *may* be visible from each position `x`. As we are assuming that RatBOT's estimate of its position is relatively close to its true position, RatBOT will actually use `LMs(` $\hat{\mathbf{x}}$ `)` (which it can compute as it knows $\hat{\mathbf{x}}$) for `LMs( x )` (which it cannot compute, as it does not know `x`).
  This algorithm uses a comprehensive "landmark-description" of the environment, which is a complete list of all of the relevant objects in that environment that could be visible, together with their respective positions. This could be based on the floor-plan of a building, which specifies the positions of the building's doors, walls, wall-hangings, etc.; or in the city-navigation context, it could be a map of the roads of a city, which specifies the locations of the significant buildings, signs, and so forth. Figure 2 shows a subset of the landmarks we used, from part of one of the hall-ways.
- `Locate(` $\hat{\mathbf{x}}$, $\hat{\sigma}$, `img, lms )` which, given RatBOT's estimate of its position $\hat{\mathbf{x}}$ and uncertainty $\hat{\sigma}$, an image `img` taken at RatBOT's current position and a set of pertinent landmarks `lms`, returns a new estimated position $\hat{\mathbf{x}}$ and uncertainty $\hat{\sigma}$ for RatBOT.
  This algorithm first attempts to find each landmark $l_i \in$ `lms` within the image `img`; here it uses $\hat{\mathbf{x}}$ and $\hat{\sigma}$ to specify where in the image to look for this l_i. It will find a subset of these landmarks, each at some angle (relative to a reference landmark). `Locate` then uses geometric reasoning to obtain

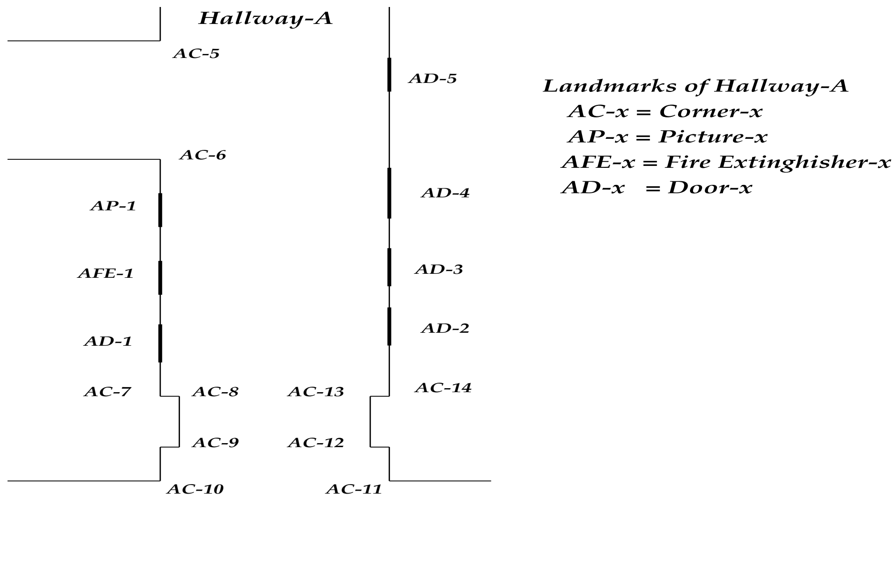

Fig. 2. Representation of Landmarks (viewing part of Hallway-A, from above)

a new estimate of RATBOT's position and uncertainty, which are then returned. (Section III-A provides more details about the low-level vision parts of this algorithm; and [13] describes the overall RATBOT system.)

As our goal is an *efficient* way of locating the agent's position, our implementation uses an inexpensive way of finding the set of landmarks based on simple tests on the visual image. *N.b.*, we are not using a general vision system — *e.g.*, we are not attempting to identify specific objects, nor specify particular qualities, from the visual information.

B. *Seek only a* Subset *of the Landmarks*

The `LMs(` $\hat{x}$ `)` function returns the set of all landmarks that might be visible in an image. Many navigation systems would then attempt to find *all* of these landmarks, and use the obtained information to estimate the agent's position; *i.e.*, would compute and use `Locate(` $\hat{x}$, $\hat{\sigma}$, `img, LMs(` $\hat{x}$ `) )`. As argued above, however, it may be better to seek only a subset of these landmarks: By avoiding "problematic" landmarks (*e.g.*, ones that tend to be not visible, or confusable), R may be able to obtain a more accurate estimate of its location, and moreover, obtain that estimate more efficiently.

We therefore want to identify and ignore these bad landmarks. To motivate the approach we decided to use, we first present two false leads. First, one immediate suggestion is simply to exclude the bad landmarks from the catalogue of all landmarks that `LMs` uses, which insures that `LMs`$(\cdot)$ will never return those landmarks. One obvious complication is the complexity of determining which landmarks are bad, as this can depend on many factors, including the color of the landmark, the overall arrangement of the entire environment (which would specify which landmarks could be occluded), sensor noise, the lighting conditions, etc. A more serious limitation is due to the fact that a landmark that is hard to see from one R position may be easy to see, and perhaps crucial, from another. This means that R should be able to use a landmark when registering its location from some positions, but not from others.

We therefore decided to use a selection function SEL that filters out the bad landmarks from the set of possibly visible landmarks, `lms = LMs(` $\hat{x}$ `)`: That is, SEL(`lms`, $\hat{x}$, $\hat{\sigma}$) returns a subset SEL(`lms`, $\hat{x}$, $\hat{\sigma}$) $=$ `lms`$'$ $\subseteq$ `lms`, which R then uses to compute its location, returning `Locate(` $\hat{x}$, $\hat{\sigma}$, `img`, `lms`$'$). We therefore want a selection function SEL such that `Locate(` $\hat{x}$, $\hat{\sigma}$, `img`, `lms`$'$) is reliably

```
Algorithm SEL^(i)( lms: landmark_set, x̂: position, σ̂: variance ): landmark_set
    OK_LMs ← {}
    ForEach ℓ ∈ lms
        KeepLM ← True
        ForEach f_k ∈ Filters( SEL^(i) )
            If [ f_k(ℓ, x̂, σ̂) ≡ Ignore ] Then KeepLM ← False;    break;
        End (inner) ForEach
        If [ KeepLM ≡ True ]  Then OK_LMs ← OK_LMs + ℓ
    End (outer) ForEach
    Return( OK_LMs )
End SEL^(i)
```

Fig. 3. PseudoCode for $\textsc{Sel}^{(i)}$ Selection Function

close to R's true position, $\mathtt{x}$. To make this more precise, let

$$\mathrm{Err}(\,\textsc{Sel},\ \langle \mathtt{x}, \hat{\mathtt{x}}, \hat{\sigma}, \mathtt{img}\rangle\,) \quad = \quad \|\ \mathtt{x}\ -\ \mathtt{Locate}(\,\hat{\mathtt{x}},\ \hat{\sigma},\ \mathtt{img},\ \textsc{Sel}(\mathrm{LMs}(\hat{\mathtt{x}}),\ \hat{\mathtt{x}}, \hat{\sigma})\ )\ \| \tag{1}$$

be the error[2] obtained when using the selection function $\textsc{Sel}$ for the "situation" $\langle \mathtt{x}, \hat{\mathtt{x}}, \hat{\sigma}, \mathtt{img}\rangle$, and let

$$\mathrm{AveErr}(\,\textsc{Sel}\,) \ = \ E_{\langle \mathrm{x}, \hat{\mathrm{x}}, \hat{\sigma}, \mathrm{img}\rangle}[\,\mathrm{Err}(\,\textsc{Sel},\ \langle \mathtt{x}, \hat{\mathtt{x}}, \hat{\sigma}, \mathtt{img}\rangle\,)\,]$$

be the expected error, over the distribution of situations $\langle \mathtt{x}, \hat{\mathtt{x}}, \hat{\sigma}, \mathtt{img}\rangle$, where $E.[\cdot]$ is the expectation operator. We want a selection function $\textsc{Sel}$ that minimizes this expected value.

The second false lead is to "engineer" this optimal selection function. One problem, as observed above, is the difficulty of determining "analytically" which landmarks are going to be problematic for any single situation. Worse, our goal is really to find the selection function that works best *over the distribution of situations*, which depends on the distribution, when the `Locate` function is called, of R's actual positions, the intensity of light sources, what other objects have been moved where, etc. Unfortunately, this distribution is not known *a priori*.

We are therefore following a third (successful) approach: of *learning* a good selection function. Here, we first specify a large (and we hope, comprehensive) class of possible selection functions $\mathcal{S} = \{\textsc{Sel}^{(i)}\}$. Then, given "labeled samples" — each including R's estimates of its position and uncertainty, the relevant landmark-set and actual image, and as the label, R's actual position — we hope to identify the selection function $\textsc{Sel}^{(i)}$ which minimizes $\mathrm{AveErr}(\,\textsc{Sel}^{(i)}\,)$.

To specify the space of selection functions, we define each selection function $\textsc{Sel}^{(i)} \in \mathcal{S}$ as a conjunction of its particular set of "filters", $\mathrm{Filters}(\textsc{Sel}^{(i)}) = \{f_1, \ldots, f_m\}$, where each filter f_k is a predicate that accepts some landmarks and rejects others. Hence, the $\textsc{Sel}^{(i)}(\ \mathtt{lms},\ \hat{\mathtt{x}},\ \hat{\sigma}\ )$ procedure will examine each $\ell \in \mathtt{lms}$ individually, and reject it if *any* f_k filter rejects it; see Figure 3.

While we can define a large set of such filters (see Section IV-A), this report focuses on only two parameterized filters:

$$\begin{array}{llll} \mathrm{TooSmall}_{k_1,k_2}(\ell, \hat{\mathtt{x}}, \hat{\sigma}): & \text{Reject } \ell & \text{if} & \|\mathrm{Posn}(\ell) - \hat{\mathtt{x}}\| > k_1 \\ & & & \text{and } \mathrm{AngleWidth}(\ell,\ \hat{\mathtt{x}}) < k_2 \\ \mathrm{BadType}_{k_3}(\ell, \hat{\mathtt{x}}, \hat{\sigma}): & \text{Reject } \ell & \text{if} & \mathrm{Type}(\ \ell\ ) \not\in k_3 \end{array}$$

[2] As we are also considering the efficiency of the overall process, we actually used the slightly more complicated error function presented in Equation 4 below.

where Type(ℓ) refers to the type of the landmark ℓ, which can be "Door", "BlackStrip", etc.[3] The k_3 parameter specifies the subset of landmark-types that should be used (*i.e.*, these are the "good types"). "Posn(ℓ)" refers to ℓ's real-world coordinates and "AngleWidth(ℓ, $\hat{\mathtt{x}}$)" refers to the angle subtended by the landmark ℓ, when viewed from $\hat{\mathtt{x}}$. Hence, $\text{TooSmall}_{k_1,k_2}(\ell, \hat{\mathtt{x}}, \hat{\sigma})$ rejects the landmark ℓ if ℓ is both too far away (greater than k_1 meters) and also too small (subtends an angle less than k_2 degrees), from R's estimated position $\hat{\mathtt{x}}$.

Using these filters, $\mathcal{S} = \{\text{SEL}_{k_1,k_2,k_3}\}$ is the set of all selection functions, over a combinatorial class of settings of these three parameters. As stated above, we want to find the best settings of these variables — *i.e.*, the values of $\langle k_1, k_2, k_3\rangle$ such that the expected error AveErr(SEL_{k_1,k_2,k_3}) is minimal.

C. Hill-Climbing in an Uncertain Space

There are two obvious challenges to the task of computing the best SEL_{k_1,k_2,k_3}. First, as noted above, the error function depends on the distribution of situations, which is not known initially. Secondly, even if we knew that information, it is still difficult to compute the optimal parameter setting, as the space of options is large and ill-structured (*e.g.*, k_3 is discrete, and there are subtle non-linear effects as we alter k_1 and k_2). Below we address these challenges, in reverse order.

We use a standard hill-climbing approach to address the second challenge; here seeking a local optimum, to avoid the complications inherent in finding the global optimum. This requires a set of operators $\mathcal{T} = \{\tau_j\}$ for mapping one selection function to another; *i.e.*, for each $\text{SEL} \in \mathcal{S}$, $\tau_j(\text{SEL}) \in \mathcal{S}$ is another selection function. The set $\mathcal{T}(\text{SEL}) = \{\, \tau_j(\text{SEL}) \mid \tau_j \in \mathcal{T}\,\}$ forms the "neighborhood" around the SEL selection function, which will be examined. Here, we use the obvious set of operators: τ_1^+ multiplies the value of k_1 by 2 and τ_1^- divides k_1's value by 2; hence $\tau_1^+(\ \text{SEL}_{4,\ 5,\ \{t1,t3,t7\}}\ ) = \text{SEL}_{8,\ 5,\ \{t1,t3,t7\}}$ and $\tau_1^-(\ \text{SEL}_{4,\ 5,\ \{t1,t3,t7\}}\ ) = \text{SEL}_{2,\ 5,\ \{t1,t3,t7\}}$. Similarly, τ_2^+ and τ_2^- respectively increment and decrement the k_2 value (by increments of 2°). There are 9 different τ_3^i operators, each of which "flips" the i^{th} bit of k_3; hence $\tau_3^{t1}(\ \text{SEL}_{4,\ 8,\ \{t1,t3,t7\}}\ ) = \text{SEL}_{4,\ 8,\ \{t3,t7\}}$ and $\tau_3^{t8}(\ \text{SEL}_{4,\ 8,\ \{t1,t3,t7\}}\ ) = \text{SEL}_{4,\ 8,\ \{t1,t3,t7,t8\}}$.

We are still left with the first challenge: dealing with the *unknown* distribution. Here, we employ the standard statistical technique of using a set of observed examples to estimate the relevant information: Let

$$\hat{E}_i^{(\mathcal{U})} \quad = \quad \hat{E}^{(\mathcal{U})}[\,\text{Err}(\,\text{SEL}^{(i)},\,\cdot\,)\,] \quad = \quad \frac{1}{|\mathcal{U}|}\sum_{u_j\in\mathcal{U}} \text{Err}(\,\text{SEL}^{(i)},\,u_j\,)$$

be the empirical average error of the selection function $\text{SEL}^{(i)}$ over the set of samples $\mathcal{U} = \{u_j\} = \{\ \langle \mathtt{x}_j, \hat{\mathtt{x}}_j, \hat{\sigma}_j, \mathtt{img}_j\rangle\ \}$, which we assume to be independent and identically distributed.[4] We can use some statistical measure to quantify our confidence that $\hat{E}_i^{(\mathcal{U})}$ will be close to the real mean $\mu_i = E_{u_j}[\,\text{Err}(\,\text{SEL}^{(i)},\,u_j\,)\,] = \text{AveErr}(\,\text{SEL}^{(i)}\,)$, as a function of the number of sample images seen. In particular, we need a function $m(\cdots)$ such that, after $m(\alpha,\beta)$ samples, we can be at least $1-\beta$ confident that the empirical average $\hat{E}_i^{(\mathcal{U})}$ will be within α of the population mean μ_i; i.e.,

$$|\mathcal{U}| \;\geq\; m(\alpha,\beta) \quad\Rightarrow\quad Pr[\,|\hat{E}^{(\mathcal{U})} - \mu| \,>\, \alpha\,] \;<\; \beta$$

We also need an "inverse function" $\alpha(\,m,\,\beta\,)$, which bounds the one-sided error, with $1-\beta$ confidence, after m samples,

$$|\mathcal{U}| \;\geq\; m \quad\Rightarrow\quad Pr[\,\hat{E}^{(\mathcal{U})} - \mu \,>\, \alpha(\,m,\,\beta\,)\,] \;<\; \beta$$

[3]The current system deals with nine different types: Miscellaneous, BlackStrip, ConcaveCorner, ConvexCorner, DarkColoredDoor, LightColoredDoor, Picture, FireExtinguisher and SupportBetweenWindows.

[4]That is, we assume there is no explicit correlation between the errors encountered from one instance to the next, which is a very reasonable, and standard, assumption.

For general distributions, we can use Hoeffding's inequality [15] to obtain

$$
\begin{aligned}
m_{HI}(\,\alpha,\,\beta\,) &= \tfrac{1}{2}\left(\tfrac{\Lambda}{\alpha}\right)^2 \ln\tfrac{2}{\beta} \\
\alpha_{HI}(m,\,\beta\,) &= \Lambda\sqrt{\tfrac{1}{2m}\ln\left(\tfrac{1}{\beta}\right)}
\end{aligned}
\tag{2}
$$

where here $\Lambda = \max_{Sel,u}\{\ \mathrm{Err}(\ \textsc{Sel},\ u\ )\ \}$ is the largest value of $\mathrm{Err}(\ \textsc{Sel},\ u\ )$ for any selection function $\textsc{Sel}$ and for any sample u. These bounds require only that the situations $u_j = \langle \hat{\mathtt{x}}_j, \hat{\sigma}_j, \mathtt{img}_j, x_j\ \rangle$ correspond to independent, identically-distributed bounded random variables. Note that there are no further constraints; in particular, their common distribution does not have to correspond to a normal distribution.

However, if we can assume that the underlying distribution of error values is effectively a normal distribution (*i.e.*, $\mathrm{Err}(\ \textsc{Sel},\ \cdot\ ) \sim \mathcal{N}(\mu,\sigma)$ for some mean μ and standard deviation σ) then we can use

$$
\begin{aligned}
m_{Norm}(\,\alpha,\,\beta\,) &= \left(\tfrac{\Lambda}{\alpha} z^{-1}(1-\tfrac{\beta}{2})\right)^2 \\
\alpha_{Norm}(m,\,\beta\,) &= \sqrt{\tfrac{s^2(m)}{m}}\ \ t^{-1}_{m-1}(1-\beta)
\end{aligned}
\tag{3}
$$

where $z(p) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{p} e^{-\frac{x^2}{2}}dx$ computes the p^{th} quantile of the standard normal distribution $\mathcal{N}(0,1)$ [2];

$$
s^2(m) \quad = \quad \frac{1}{m-1}\left[\sum_{\ell=1}^{m}\mathrm{Err}(\ \textsc{Sel},\ u_\ell\ )^2 - \frac{1}{m}\left(\sum_{\ell=1}^{m}\mathrm{Err}(\ \textsc{Sel},\ u_\ell\ )\right)^2\right]
$$

is an unbiased estimator of the variance of the $\mathrm{Err}(\ \textsc{Sel},\ u_\ell\ )$ variables after m samples; and $t_m(p)$ is the p^{th} quantile of the Student's T distribution with m degrees of freedom.

The LEARNSF algorithm, sketched in Figure 4 and summarized below, combines the ideas of hill-climbing with statistical sampling:[5] Given an initial selection function $\textsc{Sel}^{(0)} = \textsc{Sel}_{k_1,k_2,k_3} \in \mathcal{S}$, and two parameters ϵ and δ, LEARNSF will use a sequence of example situations $\{u_i\}$ to climb from the initial $\textsc{Sel}^{(0)}$ through successive neighboring selection functions $(\textsc{Sel}^{(1)}, \textsc{Sel}^{(2)}, \textsc{Sel}^{(3)}, \ldots)$ until reaching, and returning, a final $\textsc{Sel}^{(m)}$. With high probability, this $\textsc{Sel}^{(m)}$ is essentially a local optimum. Moreover, LEARNSF requires relatively few samples for each climb. To state this more precisely:

Theorem 1: [6] The $\mathrm{LEARNSF}(\ \textsc{Sel}^{(0)},\ \epsilon,\ \delta\ )$ process incrementally produces a series of selection functions $\textsc{Sel}^{(1)}, \textsc{Sel}^{(2)}, \ldots, \textsc{Sel}^{(m)}$, such that each $\textsc{Sel}^{(j+1)} = \tau_j(\textsc{Sel}^{(j)})$ for some $\tau_j \in \mathcal{T}$ and, with probability at least $1-\delta$, both

1. the expected error of each selection function is strictly better than its predecessors *i.e.*,
   $\forall 1 \le j \le m:\ \mathrm{AveErr}(\textsc{Sel}^{(j)}) \ < \ \mathrm{AveErr}(\textsc{Sel}^{(j-1)});$ and
2. the selection function returned by LEARNSF, $\textsc{Sel}^{(m)}$, is an "ϵ-local optimum" — *i.e.*,
   $\neg\exists\,\tau \in \mathcal{T}:\quad \mathrm{AveErr}(\ \tau(\textsc{Sel}^{(m)})\ ) \ < \ \mathrm{AveErr}(\ \textsc{Sel}^{(m)}\ ) - \epsilon\ .$

given the appropriate statistical assumptions (*viz.*, $\mathrm{LEARNSF}_{Norm}$ requires that the underlying distribution is normal; $\mathrm{LEARNSF}_{HI}$ does not require any such assumption). Moreover, LEARNSF will terminate with probability 1, and will stay at any $\textsc{Sel}^{(j)}$ (before either terminating or climbing to a new $\textsc{Sel}^{(j+1)}$) for a number of samples that is polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $|\mathcal{T}|$ and Λ. □

[5] Notice LEARNSF is actually estimating the value of $\hat{E}^{(\mathcal{U})}[\,\textsc{Sel}^{(i)}\,] - \hat{E}^{(\mathcal{U})}[\textsc{Sel}']$, rather simply $\hat{E}^{(\mathcal{U})}[\,\textsc{Sel}^{(i)}\,]$. As the range of such values is actually 2Λ (from $[-\Lambda,+\Lambda]$), both the $m_\chi(\alpha,\beta)$ and the $\alpha_\chi(\,n,\,\beta\,)$ functions will use "2Λ" rather than "Λ". Also, following standard practice, the "*Norm*" system will skip the early climb and early termination tests (which use $\alpha_{Norm}(\,i,\,\cdot\,)$) until $i = 4$. Otherwise, the implicit assumption that the sum of the random variables is normal, is likely to be violated, which could cause the system to take an inappropriate action.

[6] The proof of this theorem is isomorphic to the proof that appears in [12].

```
Algorithm LearnSF_χ( Sel^(0): selection_function, ε: ℜ+, δ: ℜ+ ) : selection_function
  For   j = 0..∞  do
    Let δ_j  ←  δ 6 / ((j+1)^2 π^2) ,
       T( Sel^(j) )  ←  { τ(Sel^(j)) ∈ S | τ ∈ T  &  τ(Sel^(j)) ≠ Sel^(j) },
                                          % T( Sel^(j) ) are Sel^(j)'s neighbors
       L_j  ←  m_χ( ε/2, δ_j / (2|T(Sel_j)|) )      % L_j is max # of samples, Sel^(j) iteration
    ForEach  Sel' ∈ T( Sel^(j) )  do  Let Δ(Sel^(j), Sel', 0)  ←  0 .
    For  i = 1..L_j  do                   % Get and process i^th image
      Get sample q_i  (from oracle)
      ForEach  Sel' ∈ T( Sel^(j) )  do
          Let  Δ( Sel^(j), Sel', i )  ←  Δ( Sel^(j), Sel', i−1 )  +  [ Err( Sel', q_i ) − Err( Sel^(j), q_i ) ] .
      If   i < L_j
           [ If  ∃Sel' ∈ T( Sel^(j) )  s.t.  (1/i) Δ( Sel^(j), Sel', i )  >  α_χ( i, δ_j / (2(L_j−1)|T(Sel_j)|) )
           | Then  Let Sel^(j+1) ← Sel'
      Then |       Exit For (inner loop)
           | Else If  ∀Sel' ∈ T( Sel^(j) ):  (1/i) Δ( Sel^(j), Sel', i )  <  ε − α_χ( i, δ_j / (2(L_j−1)|T(Sel_j)|) )
           [       Then  Return[ Sel^(j) ]  (Exiting both inner & outer For Loops)
           [ If  ∃Sel' ∈ T( Sel^(j) )  s.t.  (1/L_j) Δ( Sel^(j), Sel', L_j )  >  ε/2
      Else | Then  Let Sel^(j+1) ← Sel'
           |       Exit For (inner loop)
           [ Else  Return[ Sel^(j) ].   (Exiting both inner & outer For loops)
    End For (inner loop)
  End For (outer loop)
End LearnSF
```

Fig. 4. PseudoCode for LearnSF Algorithm

To summarize the code: LearnSF examines a sequence of images, one by one. On seeing each image, LearnSF computes the error of the given $\text{Sel}^{(0)}$ selection function, summed over all of the images seen so far, and compares that value with comparable values for each of $\text{Sel}^{(0)}$'s neighbors. If any neighbor appears to be significantly better, it becomes the new performance element $\text{Sel}^{(1)}$. LearnSF then compares $\text{Sel}^{(1)}$'s performance with that of $\text{Sel}^{(1)}$'s neighbors over the next set of images; and once again, if any of $\text{Sel}^{(1)}$'s neighbors appears much better, LearnSF will climb to this apparently-superior element $\text{Sel}^{(2)}$, and so forth. On the other hand, if no neighbor looks significantly better, LearnSF will exercise other portions of its code: If all of $\text{Sel}^{(j)}$'s neighbors appear comparable to or worse than the current $\text{Sel}^{(j)}$, LearnSF will terminate, returning $\text{Sel}^{(j)}$. If neither of these conditions holds, LearnSF will, in general, simply process the next image, then use this image, in addition to the previous ones, when comparing the current $\text{Sel}^{(j)}$ to its neighbors. However, if LearnSF has stayed on the current $\text{Sel}^{(j)}$ for a sufficiently large number of queries (L_j), LearnSF will use easier-to-satisfy thresholds to decide whether to climb to some $\text{Sel}^{(j+1)}$ or terminate, and will necessarily perform one of those actions.

Two final observations: First, notice that LearnSF will (probably) process more images using later selection function than using the earlier ones, as its tests are increasingly more difficult to pass. This behavior is desirable, as it means that the overall system is dealing with larger numbers of images using later, and therefore probably better, selection functions.

Second, observe that our empirical approach handles sensor noise appropriately: If the effects of sensor noise is especially problematic with respect to certain landmarks, we expect that the empirical scores obtained using these landmarks to be inferior to those based on other sets, which means

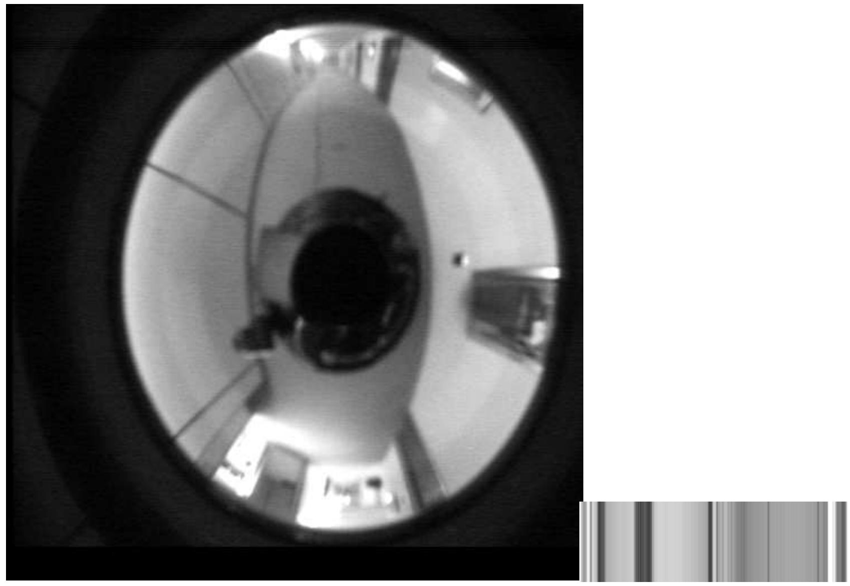

Fig. 5. The RatBOT Platform; RatBOT's view (looking up at christmas tree ornament); and a "Strip", corresponding to an annulus in the view

LearnSF will learn to ignore such landmarks, as desired. Alternatively, if these errors are completely uncorrelated with the landmarks, the best strategy is simply to ignore this factor and select the landmark set leading to the best empirical score [8]; notice again that this is precisely what LearnSF will do.

III. Empirical Results

The arguments above suggest that a good selection function should help an autonomous agent to register its position efficiently and accurately, and also that LearnSF should help find such a good selection function. To test these theoretical claims, we implemented various selection functions and the LearnSF learning algorithm, and incorporated them within an implemented autonomous agent, the RatBOT system described in [13]. This section describes our empirical results.

A. Performance System

The physical apparatus consists of a CCD camera attached to the top of a "Nomad 200" robot, pointing up at a spherical mirror (which is actually a christmas tree ornament); see left image of Figure 5. This produces images containing a 360° panoramic view of the environment, such as the one shown in the middle of Figure 5. The performance system then extracts a 1-pixel annulus from each of these images; these values (computed by averaging the intensity values of the appropriate regions of the image) correspond to the light intensity at each of 360 1-degree positions around RatBOT, at the real-world height of the center of its circular mirror. This is represented as an array of 360 8-bit intensity values, such as the one shown on the right of Figure 5.[7]

This 1-D strip is passed as the third argument to the `Locate` algorithm (*i.e.*, it is the "`img`" in `Locate(` $\hat{x}$, $\hat{\sigma}$, `img, lms ))`, which uses this information to produce estimates of RatBOT's position and uncertainty. `Locate` first extracts the "edges" in this 1-D image (read "zero-crossings of second derivative"), and uses the obvious algorithm to match the angles (corresponding to these edge positions) to the candidate landmarks: matching each landmark to the edge closest to its anticipated position, subject to the constraint that the edges associated with the landmarks appear in the "proper

[7] There are several obvious reasons for considering such 1-D panoramic views, *e.g.*, they require relatively little effort to produce and little space to store, and they render the vision system relatively insensitive to the camera's orientation. Our project committed to using them when we found that they worked effectively for our navigation task.

TABLE I
INITIAL, AND FINAL, SELECTION FUNCTIONS

$\text{SEL}^{(\chi)}$	k_1	k_2	k_3	AveErr($\text{SEL}^{(\chi)}$)	AveErr($\text{SEL}^{(\chi:\omega)}$)	k_1	k_2	k_3
$\text{SEL}^{(A)}$	10	0	111111111	0.885	0.186	1.25	4	111011111
$\text{SEL}^{(B)}$	5	10	000000000	1.723	0.276	10	10	001000000
$\text{SEL}^{(C)}$	5	2	111010111	0.367	0.173	1.25	4	111010111
$\text{SEL}^{(D)}$	5	2	101111010	0.381	0.247	2.5	4	101111110

order": *i.e.*, if landmark ℓ_i should be "clockwise" from ℓ_j, relative to RATBOT's estimated position, then the matching algorithm will not match ℓ_i to an edge that is counter-clockwise from an edge it matched to ℓ_j. (Notice that some of the landmarks sought may not be found, and that some of the edges may be unmatched.) Given this edge-to-landmark mapping, `Locate` uses the Betke/Gurvits algorithm [1] to efficiently produce an estimate of the agent's position and uncertainty.[8]

We gathered 270 such "pictures" at known locations within three halls of our building. We also identified 157 different landmarks in these regions, each represented as an object of a specified type (one of the nine categories), located between a pair of real-world coordinates $\langle x_1, y_1\rangle$ and $\langle x_2, y_2\rangle$; where, once again, the $\langle x, y\rangle$ plane is parallel to the floor and goes through the center of the spherical mirror; see Figure 2.

Each experiment used a particular initial selection function, values for ϵ and δ, error function, uncertainty, and statistical assumption. We will first describe one experiment in some detail, then present a battery of other experiments that systematically vary the experimental parameters.

B. Experiment#1 Specification

The first experiment used the selection function $\text{SEL}^{(A)}$, $\epsilon = 0.1$ m, $\delta = 0.05$, "ratio" $= 0.01$, uncertainty value $\sigma = 0.3$ m, and the "normality" assumption. Now to explain these terms:

LEARNSF began with the obvious degenerate $\text{SEL}^{(A)}$ selection function that uses *all* landmarks; see columns one through four of the top row of Table I. (As nothing can subtend an angle less than 0 degrees, $\text{TooSmall}_{k_1=10,k_2=0}$ will not reject any landmark, and as all of k_3's bits are "1", BadType_{k_3} will also accept every landmark.)

The $\delta = 0.05$ setting means that we are willing to accept roughly 1 mistake in 20 runs. Setting $\epsilon = 0.1$ means that we do not care if the average error of two selection functions differs by less than 0.1 m; as the error can be as large as 4 m,[9] this corresponds to an allowable tolerance of only 2.5%. The "normality" assumption means we are assuming that the error values are normally distributed, which sanctions the use of the LEARNSF_{Norm} algorithm, which uses m_{Norm} and α_{Norm} functions from Equation 3.

To explain "ratio= 0.01", recall that our goal is to minimize both positional error and computational time. We therefore use an error function that is the weighted sum of the positional error (which is the difference between the obtained position estimate and the real position) and the number of landmarks

[8] This description is intentionally brief, as the `Locate` algorithm is *not* the focus of our research. As noted above, LEARNSF views this algorithm as a black box, whose performance it can sample, but whose internals are unavailable. This means we expect LEARNSF to have similar learning behavior if it used another more elaborate landmark-to-location algorithm. It is worth noting only that the Betke/Gurvits algorithm requires that at least 3 landmarks be identified for its triangulation routine to be meaningful, and so if `Locate` receives under 3 landmarks, `Locate` will simply return its first argument $\hat{\mathbf{x}}$ as the current new positional estimate.

[9] We "topped" off the value of Err(SEL, u) at 4.0, meaning its range is Err(SEL, u) $\in (0, 4]$.

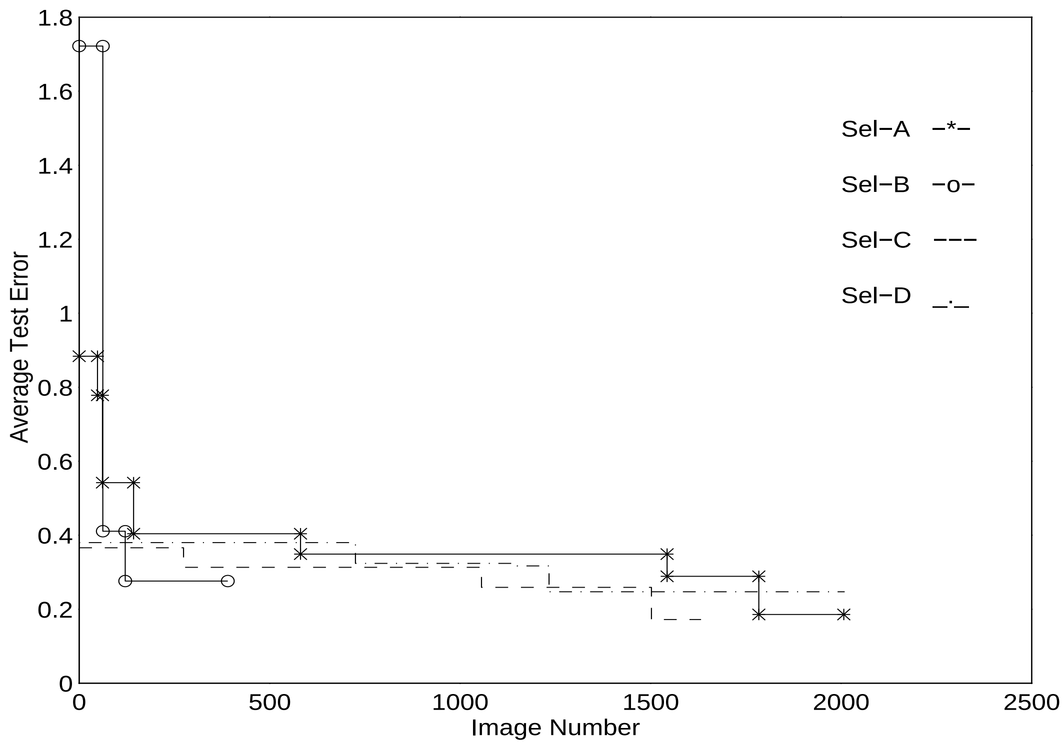

Fig. 6. LEARNSF's Hill-Climbs: $\sigma = 0.3$ m

that were selected, with weights of 1 and "ratio", respectively; hence, the error function used here is

$$\begin{aligned} \text{Err}(\,\textsc{Sel},\ \langle \mathtt{x}, \hat{\mathtt{x}}, \hat{\sigma}, \mathtt{img}\rangle\,) \;=\; & \|\, \mathtt{x} \;-\; \mathtt{Locate}(\,\hat{\mathtt{x}},\ \hat{\sigma},\ \mathtt{img},\ \textsc{Sel}(\text{LMs}(\hat{\mathtt{x}}),\ \hat{\mathtt{x}}, \hat{\sigma})\,)\,\| \\ & +\;\; \text{ratio} \;\times\; \#\text{Landmarks\_sought} \end{aligned} \tag{4}$$

Setting the ratio to 0.01 means, in effect, that each additional landmark "costs" 0.01 m. That is, suppose selection function $\textsc{Sel}^{(\alpha)}$ has an average accuracy of L_α and seeks on average M_α landmarks, while selection function $\textsc{Sel}^{(\beta)}$ has an accuracy L_β and seeks M_β landmarks. If both $L_\beta < L_\alpha$ and $M_\beta < M_\alpha$, then clearly $\textsc{Sel}^{(\beta)}$ is better than $\textsc{Sel}^{(\alpha)}$; and similarly $\textsc{Sel}^{(\alpha)}$ is better if both $L_\alpha < L_\beta$ and $M_\alpha < M_\beta$. The hard cases are if $L_\beta < L_\alpha$ but $M_\beta > M_\alpha$, or vice versa. Setting "ratio$= 0.01$" means that, for $\textsc{Sel}^{(\beta)}$ to be preferred, L_β must be 0.01 m better that L_α to justify each additional landmark in $M_\beta - M_\alpha$.

Finally, to explain the uncertainty value $\sigma = 0.3$ m: While *we* know that image $\mathtt{img}_i$ is taken at location $\mathbf{x}_i$, it is unrealistic to assume that RATBOT will know that information; in general, we assume that RATBOT will instead have computed an approximation, $\hat{\mathtt{x}}_i$. We model this by setting $\hat{\mathtt{x}}_i = x_i + \nu_i^{(\sigma)}$, where each $\nu_i^{(\sigma)}$ is a normally-distributed random value with mean zero and variance σ. Here, we used $\sigma = 0.3$ m. Recall also that the `Locate` function needs a value for $\hat{\sigma}$ to constrain its landmark-location process; we also set $\hat{\sigma}$ to be σ.

(There is no need to include other parameters within our model. In particular, as we are using real data from a real camera, there is no reason to include an explicit model of sensor noise, etc.)

C. Experiment#1 Results

Given these settings, LEARNSF observed 48 samples before climbing to the new selection function[10] $\textsc{Sel}^{(A:1)} = \langle\,\langle 10,\, 0\rangle;\ [111011111]\,\rangle$, which differs from $\textsc{Sel}^{(A)}$ only by rejecting all `Concave_Corners`. It continued using this selection function for 13 additional samples, before climbing to the $\textsc{Sel}^{(A:2)} =$

[10]In general, $\textsc{Sel}^{(x:j)}$ refers to the selection function reached after j climbs, when starting from $\textsc{Sel}^{(x)}$. Hence, $\textsc{Sel}^{(x:0)} \equiv \textsc{Sel}^{(x)}$.

TABLE II

DATA FOR LEARNSF'S CLIMBS; FOR $\text{SEL}^{(A)}$, $\epsilon = 0.1$ M, $\delta = 0.05$, $\sigma = 0.3$ M

Sample #	Selection Function	E[TestErr]	E[Pos'n Err]	E[# of LMs]
0	$\text{SEL}^{(A:0)} = \langle\, \langle 10.\ \ , 0\rangle;\ [111111111]\, \rangle$	0.885	0.327	55.77
48	$\text{SEL}^{(A:1)} = \langle\, \langle 10.\ \ , 0\rangle;\ [111011111]\, \rangle$	0.778	0.335	44.36
61	$\text{SEL}^{(A:2)} = \langle\, \langle 10.\ \ , 2\rangle;\ [111011111]\, \rangle$	0.542	0.329	21.36
143	$\text{SEL}^{(A:3)} = \langle\, \langle\ 5.\ \ , 2\rangle;\ [111011111]\, \rangle$	0.405	0.281	12.38
581	$\text{SEL}^{(A:4)} = \langle\, \langle\ 2.50, 2\rangle;\ [111011111]\, \rangle$	0.360	0.265	8.43
1543	$\text{SEL}^{(A:5)} = \langle\, \langle\ 2.50, 4\rangle;\ [111011111]\, \rangle$	0.290	0.220	6.96
1784	$\text{SEL}^{(A:6)} = \langle\, \langle\ 1.25, 4\rangle;\ [111011111]\, \rangle$	0.186	0.141	4.52

$\langle\, \langle 10,\ 2\rangle;\ [111011111]\, \rangle$ which also rejects any landmark that is more than 10 meters from R's estimated position and also subtends an angle less than 2^o. It continued using this $\text{SEL}^{(A:2)}$ function for another 82 samples, before climbing to $\text{SEL}^{(A:3)} = \langle\, \langle 5,\ 2\rangle;\ [111011111]\, \rangle$, which rejects any landmarks that is more than 5 m from R, and less than 2^o. The next three climbs respectively cut the distance required to reject a landmark to 2.5 m ($\text{SEL}^{(A:4)} = \langle\, \langle 2.5,\ 2\rangle;\ [111011111]\, \rangle$ on image 581[11]), set the angular threshold to 4^o ($\text{SEL}^{(A:5)} = \langle\, \langle 2.5,\ 4\rangle;\ [111011111]\, \rangle$ on image 1543) and set the distance threshold to 1.25 m ($\text{SEL}^{(A:6)} = \langle\, \langle 1.25,\ 4\rangle;\ [111011111]\, \rangle$ on image 1784). LEARNSF then examined another 223 images before terminating, and declaring this selection function to be a "0.1-local optimum" — *i.e.*, none of $\text{SEL}^{(A:6)}$'s neighbors has a score that is more than $\epsilon = 0.1$ m better than $\text{SEL}^{(A:6)}$. (In fact, $\text{SEL}^{(A:6)}$ is actually a *bona fide* local optimum, as all of $\text{SEL}^{(A:6)}$'s neighbors are strictly worse.)

The "-*-" line in Figure 6 (associated with "Sel-A") shows LEARNSF's performance here. Each horizontal line-segment corresponds to a particular selection function, where the line's y-value indicates the "average test error" of its selection function, which was computed by running this selection function through all 270 images. (To avoid testing on the training data, we computed this value using a *new* set of randomly-generated positional estimates, $\{\hat{\mathbf{x}}'_i = x_i + \nu_i^{(\sigma)\prime}\}$, where each $\nu_i^{(\sigma)\prime}$ is a new random variable, drawn from a 0-mean σ-variance distribution, whose value is probably different from the $\nu_i^{(\sigma)}$ variable used to specify $\hat{\mathbf{x}}_i$.[12]) These horizontal lines are connected by vertical lines whose x-value specifies the sample number when LEARNSF climbed. The first horizontal line shows that RATBOT used $\text{SEL}^{(A)}$ for 48 samples before "climbing" — indicated by the vertical line at $x = 48$. The y-value of this first horizontal line, 0.885, is the "average test error" of this selection function. The second horizontal line corresponds to the $\text{SEL}^{(A:1)}$ selection function, whose average test error is 0.778; etc. The final line ends at $x = 2007$, meaning that LEARNSF terminated after seeing the 2007^{th} sample.

Table II presents a more detailed break-down, showing the average accuracy and average number of landmarks sought, for these seven different selection functions (given the other parameters specified above). Several comments are in order: First, observe the "E[TestErr] = AveErr(SEL)" values are strictly decreasing, as desired. However, the *positional error* E[Pos'n Err] went up between $\text{SEL}^{(A:0)}$

[11] This required cycling through the collection of 270 images two full times, then reaching image number 581 as the 41^{st} image in the third epoch.

[12] We actually produced 10 such $\{\hat{\mathbf{x}}'_i\}$ variable-sets, and computed an estimate of AveErr(SEL) from each set. The y-values plotted are the average of these values. The variances of these values were quite small — with empirical standard deviations well under 1% of the empirical mean. For example, the empirical standard deviation for AveErr($\text{SEL}^{(A)}$) was 0.000091 over these ten estimates, which is about 0.01% of the mean. Also, to account for the fact that positional errors can accumulate if there is no feedback, we also imposed an additional penalty on the SEL selection function's score if SEL was unable to find at least three landmarks on the previous image(s). In particular, if SEL was unable to find three landmarks for the previous k consecutive images, $\texttt{img}_{i-k}$, $\texttt{img}_{i-k+1}$, $\ldots$ $\texttt{img}_{i-1}$, then the value we used for SEL's positional error (within Equation 4) was $\|\ \mathbf{x}_i\ -\ \texttt{Locate}(\ \hat{\mathbf{x}}_i,\ \hat{\sigma}_i,\ \texttt{img}_i,\ \text{SEL}(\text{LMs}(\hat{\mathbf{x}}_i),\ \hat{\mathbf{x}}_i, \hat{\sigma}_i)\ )\ \| \times 1.1^k$. Notice this reduces to the actual positional error in the standard case, when SEL found at least three landmarks on $\texttt{img}_{i-1}$, and so $k = 0$.

and $\text{SEL}^{(A:1)}$. This rise was compensated by the decrease in the number of landmarks sought; *i.e.*, on average, $\text{SEL}^{(A:0)}$ looked for 11.41 more landmarks than $\text{SEL}^{(A:1)}$. The second point is that the average positional error of initial selection function $\text{SEL}^{(A:0)}$, 0.327 m, exceeds the 0.3 m expected due to the variance. This increased error is due partly to errors in our measurements of the landmarks; such errors were one of the initial motivations for this enterprise. (Of course, this would not be an issue in the anticipated future contexts, when we might begin with the building's actual floor-plans.) Notice, however, that the positional error of the final $\text{SEL}^{(A:6)}$ selection function was only 0.141 m — significantly below 0.3 m — which illustrates the effectiveness of finding the useful landmarks. Notice, finally, that this resulting selection function also has better performance than simply "closing our eyes" and accepting the anticipated error of 0.3 m, formed by adding the positional error of 0.3 m and the "landmark penalty" of 0 $m = 0.01 \frac{m}{\text{landmark-sought}} \times 0$ landmark-sought; see also Section III-D below.

Timing Information: The overall LEARNSF system, including both performance and learning components, worked fairly efficiently, requiring only 4.32 CPU-seconds on a SUN MP690 to (process 48 images and) climb to $\text{SEL}^{(A:1)}$; another 1.58 seconds to (process 13 more images and) reach $\text{SEL}^{(A:2)}$ then respectively 12.48, 41.58, 100.96 and 25.37 seconds to process the 82, 438, 962 and 241 images required to reach $\text{SEL}^{(A:2)}$ through $\text{SEL}^{(A:6)}$; and finally an additional 18.33 more second to process 233 more samples and terminate. Hence, it processed over 2000 images, and performed six climbs as well as one termination, in under 3.5 minutes. Each of these numbers corresponds to the time required to compute a position estimate using each of between 11 to 15 selection functions; here, for each such SEL', LEARNSF first uses SEL' to select a subset of landmarks, then seeks these landmarks within the image, and finally uses the obtained edge-to–landmark correspondences to obtain an estimate of the agent's position. Hence, the average time to process a single image, using a single selection function, is approximately 7.8 milliseconds. (As such, the system required around $13 \times 0.0078 = 0.10$ seconds to process each image.) This does not include the time required to pre-process the image, which involved extracting the annulus and computing the set of edges. (This low-level pre-processing step produces information that was shared by all selection functions.)

These timing numbers are all based on a C-code implementation, which we have not yet attempted to optimize. In particular, a non-trivial amount of time was spent computing information that was used to produce the comparison information presented here.

Variability: As mentioned above, the performance system uses its estimate of RATBOT's position to compute a more accurate position; we model this estimate as $\hat{\mathbf{x}}_i = x_i + \nu_i^{(\sigma)}$, where $\nu_i^{(\sigma)}$ is a normally-distributed random value with mean 0 and variance $\sigma = 0.3$ m. The run above uses only a single set of 270 $\{\nu_i^{(\sigma)}\}$ values, one for each RATBOT position. To get a sense of the system's "stability", we ran LEARNSF ten more times, all in the same $\langle \text{SEL}^{(A)},\ \epsilon = 0.1,\ \delta = 0.05,\ \sigma = 0.3\ m, \text{Ratio} = 0.01, \text{"Normal"} \rangle$ context, but differing by using different random seeds, and hence dealing with different sets of $\{\nu_i^{(\sigma)}\}$ values. We observed that in all 10 cases, LEARNSF climbed first through the same sequence of selection functions ($\text{SEL}^{(A:1)}$ to $\text{SEL}^{(A:2)}$ to ... $\text{SEL}^{(A:6)}$), and then terminated. The number of images varied, however: Using the "mean±standard-deviation" notation, the number of (additional) images required for each climb, and to termination was

Climb to...	$\text{SEL}^{(A:1)}$	$\text{SEL}^{(A:2)}$	$\text{SEL}^{(A:3)}$	$\text{SEL}^{(A:4)}$	$\text{SEL}^{(A:5)}$	$\text{SEL}^{(A:6)}$	termination
# of images	48 ± 0	13 ± 0	76.9 ± 3.9	470.3 ± 73.7	767.0 ± 247.0	294.6 ± 58.5	253.2 ± 66.3

That is, LEARNSF required 48 ± 0 images for the first climb, then 13 ± 0 *additional* images for the second, etc. (LEARNSF exhibited the same variability in several other contexts as well: In each, it progressed through essentially the same sequence of selection functions, but the actual numbers of images used on each step often varied considerably.)

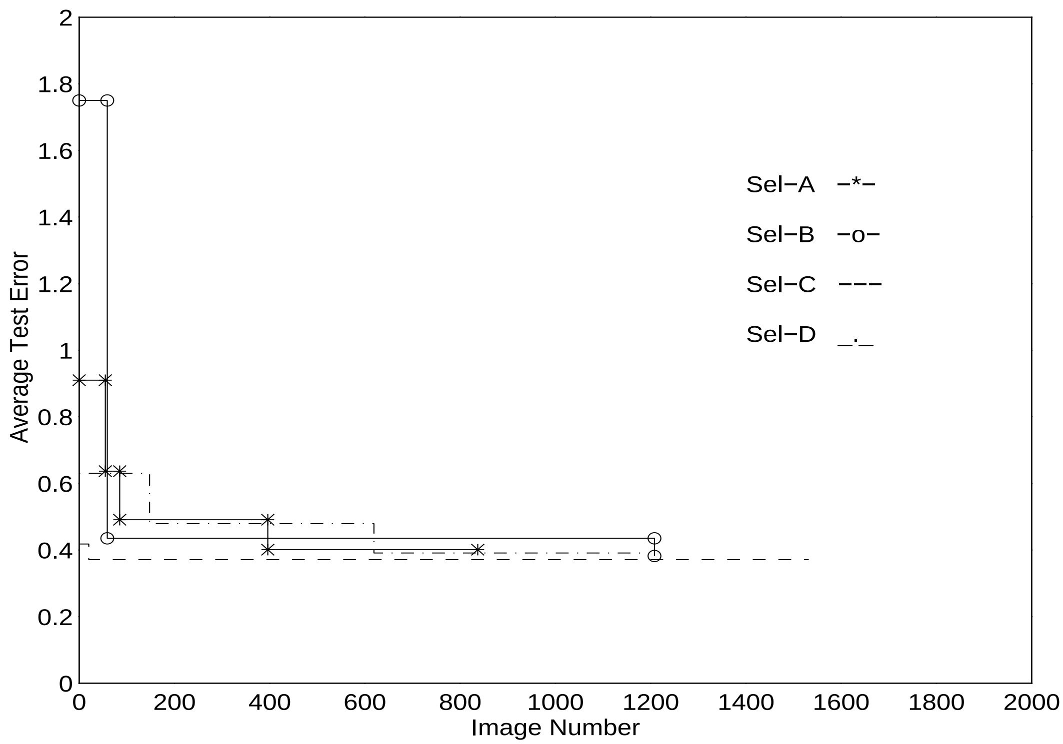

Fig. 7. LEARNSF's Hill-Climbs: $\sigma = 0.5$ m

D. Other Experiments

Different Selection Functions: While the $\text{SEL}^{(A)}$ function is an obvious choice, in some situations we may (think that we) know more about the environment, and so begin LEARNSF using some another initial selection function. This subsection considers three other such initial functions, including $\text{SEL}^{(B)} = \langle \langle 5,\ 10\rangle;\ [000000000] \rangle$, which *rejects every* landmark, as well as $\text{SEL}^{(C)} = \langle \langle 5,\ 2\rangle;\ [111010111] \rangle$, which rejects a landmark if either it is more than 5 meters away from the agent's estimated position and also subtends an angle less than 2 degrees,) or if the landmark's type is either of "ConvexCorner" or "LightColoredDoor" (these are the fourth and sixth types, corresponding to the bits that are 0 in the $\text{SEL}^{(C)}$ row, first k_3 column, of Table I); and $\text{SEL}^{(D)} = \langle \langle 5,\ 2\rangle;\ [101111010] \rangle$, which rejects a landmark for either the size-and-angle reason shown above, or if it is one of "BlackStrip", "Picture" or "SupportBetweenWindows". Figure 6 graphs the results of these functions. Notice that LEARNSF finds improvements in all four cases, climbing a total of 14 times.

The right-side of Table I presents the final selection functions obtained, for each initial selection function, along with the average test error for each. We were surprised to see that these average test values were relatively close to one another, despite the fact that the various final selection functions, themselves, were very different. We found this happens in other contexts as well (e.g., when using other values of σ, ϵ, etc.; see below), which suggests two things: (1) that these scores perhaps correspond to the best that is possible, given these sensors and basic algorithms; and (2) that the space being searched is sufficiently "smooth" that our hill-climbing algorithm can reach such optima.

Different σ Values: Figure 7 (resp., Figures 8) presents the results of running LEARNSF on the same 4 selection functions, but with $\sigma = 0.5$ m (resp., $\sigma = 1.0$ m); all of the other conditions are the same. While there are some minor differences among these four graphs, their overall characteristics are the same, in that LEARNSF (almost always) climbed to successively better selection functions, and always terminates at an appropriate ϵ-local optimum.The one exception was when using the $\text{SEL}^{(C)}$ function, when $\sigma = 1000$. Here, we see that $\text{SEL}^{(C:4(1000))}$ was actually worse than $\text{SEL}^{(C:3(1000))}$,

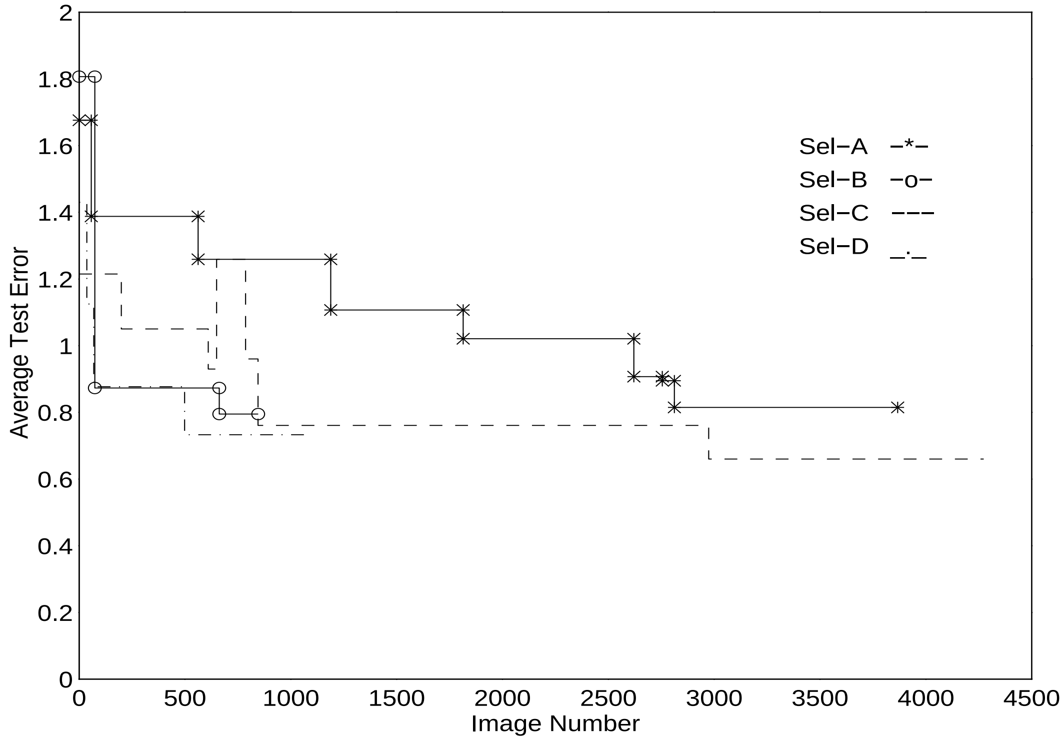

Fig. 8. LearnSF's Hill-Climbs: $\sigma = 1.0$ m

meaning LearnSF's fourth climb was inappropriate. (We will discuss this more, in the summary below.)

Different ϵ and δ Values: We also varied the values of ϵ and δ. Tables III-D and IV presents the data, using the $\text{Sel}^{(A)}$ selection function and $\sigma = 0.3$ m. (We obtained similar results using other settings.) Notice that LearnSF climbed to the same selection functions in all cases; the only difference was the number of samples required for each climb, and when each terminated. In particular, LearnSF required slightly fewer images to climb or terminate as we increased the δ value; of course, it is also more likely to make a mistake here (although such mistakes did not happen during these particular runs).

We also observed that LearnSF needed more samples for each decision as we decreased the ϵ value. There was a surprisingly sharp change as we descended from $\epsilon = 0.075$ to $\epsilon = 0.060$: LearnSF went from requiring about 2000 total samples for the entire run, to well over 20,000! This is because the separation between the scores of $\text{Sel}^{(A:6)}$ and its best neighbor was very small; smaller than the user-specified "don't care" criterion. That is, by setting $\epsilon = 0.0060$, the user is stating that he does not want LearnSF to return a selection function $\text{Sel}^{(m)}$ if it has a neighbor whose score is more than 0.060 better; *i.e.*, if there is a $\text{Sel}' = \tau(\text{Sel}^{(m)})$ such that $\text{AveErr}(\text{Sel}') \leq \text{AveErr}(\text{Sel}^{(m)}) - 0.060$. In practice, of course, if the user observed LearnSF requiring an unreasonably large number of samples without climbing or terminating, he could then decide whether he *really* needs a selection function that is an ϵ-local optimum, for the current ϵ value. If the answer is "No", he can then simply terminate the learning system prematurely.

Two final comments: First, LearnSF did not require such large numbers of samples for any larger values of ϵ, in any of the other contexts we considered. Second, notice that the performance system is actually using a locally-optimal selection function for all of these images, which means these overall learning+performance system will process these samples relatively efficiently. (This argues that it may not be that necessary to terminate the system prematurely.)

TABLE III
VARYING ϵ (USING $\text{SEL}^{(A)}$, $\delta = 0.05$, $\sigma = 0.3$ M, RATIO $= 0.01$)

ϵ	to $\text{SEL}^{(A:1)}$	to $\text{SEL}^{(A:2)}$	to $\text{SEL}^{(A:3)}$	to $\text{SEL}^{(A:4)}$	to $\text{SEL}^{(A:5)}$	to $\text{SEL}^{(A:6)}$	terminate
1.000	–	–	–	–	–	–	6
0.500	–	–	–	–	–	–	9
0.200	46	52	134	–	–	–	305
0.150	47	59	139	–	–	–	566
0.125	47	60	140	581	–	–	1164
0.100	48	61	143	581	1543	1784	2007
0.075	49	62	147	619	1543	1856	2210
0.060	50	64	151	673	1552	1860	20231
0.050	50	64	153	678	1552	1860	26548
0.025	52	67	159	700	1567	1965	$> 54000^*$

* Here, we terminated LEARNSF after 200 epochs.

TABLE IV
VARYING δ (USING $\text{SEL}^{(A)}$, $\epsilon = 0.1$ M, $\sigma = 0.3$ M, RATIO $= 0.01$)

δ	to $\text{SEL}^{(A:1)}$	to $\text{SEL}^{(A:2)}$	to $\text{SEL}^{(A:3)}$	to $\text{SEL}^{(A:4)}$	to $\text{SEL}^{(A:5)}$	to $\text{SEL}^{(A:6)}$	terminate
0.5	44	50	129	468	1280	1507	1756
0.1	47	59	139	574	1541	1781	2003
0.05	48	61	143	581	1543	1784	2007
0.01	53	66	155	687	1557	1862	2065
0.005	52	67	158	696	1567	1969	2116
0.001	55	62	159	814	2002	2311	2597

Different "#Landmarks to Positional Error" Ratios: Table V summarizes the effects of varying the landmark-to-error ratio. A value of 0 means we are concerned with accuracy alone, independent of the number of landmarks sought. The actual selection function reached were different for different error-functions, as were the score produced. Notice, however, that it was uniformly able to climb to superior function, throughout the range ratio $\in$ [0.0, 0.020].

Weaker Statistical Assumption: The m_{Norm} and α_{Norm} functions used within LEARNSF_{Norm} are not guaranteed to work effectively unless the error values are normally distributed. While there is no *a priori* reason to believe that this claim must be true, the fact that there were so few bad climbs (where a $\text{SEL}^{(\chi:i)}$-to-$\text{SEL}^{(\chi:i+1)}$ climb is bad if AveErr($\text{SEL}^{(\chi:i+1)}$) $>$ AveErr($\text{SEL}^{(\chi:i)}$)) provides one empirical datapoint that buttresses the "normality assumption".

We also experimented with the slightly different LEARNSF_{HI} algorithm, which uses the stronger m_{HI} and α_{HI} functions (defined in Equation 2) which are (probabilistically) guaranteed to work for any bounded distribution of error values, not just normal distributions. We found that this LEARNSF_{HI} system had essentially the same functionality: in almost all cases, it performed the same climbs that the LEARNSF_{Norm} system had. The big difference was sample efficiency: LEARNSF_{HI} required in general a factor of 10 more samples than LEARNSF_{Norm}; *e.g.*, it required hundreds to thousands of samples to perform a climb that LEARNSF_{Norm} would perform after tens to hundreds of samples.

E. Summary of Empirical Results

There are several obvious conclusions. First, we see that selection functions are useful; notice in particular that the landmarks they returned enabled R to obtain fairly good positional estimates (within a few tenths of a meter, which was sufficient for our purposes of identifying offices, etc. [13]).

TABLE V

VARYING LANDMARK-TO-ERROR RATIO (USING $\text{SEL}^{(A)}$, $\epsilon = 0.1$ M, $\delta = 0.05$, $\sigma = 0.3$ M)

Ratio	Err	$\text{SEL}^{(A:1)}$/Err	$\text{SEL}^{(A:2)}$/Err	$\text{SEL}^{(A:3)}$/Err	$\text{SEL}^{(A:4)}$/Err	$\text{SEL}^{(A:5)}$/Err	$\text{SEL}^{(A:6)}$/Err
0	346	2022/ 280					
2	438	102/ 383	609 / 301				
5	606	64/ 467	149 / 349				
10	885	48/ 778	61 / 543	143 / 404	581 / 350	1543/ 290	1784/ 186
20	1443	18/ 1222	28 / 1174	47 / 664	52 / 634	96/ 478	996/ 426

The ratio, and error values, are in millimeters. Most of these $\text{SEL}^{(A:\chi)}$ selection function were different from the ones reached for the "ratio $= 0.01\frac{m}{\text{landmark}} = 10\frac{mm}{\text{landmark}}$" values.

Notice also that the obvious degenerate selection function, $\text{SEL}^{(A)}$ which accepted all landmarks, was *not* optimal; *i.e.*, there were functions that worked more effectively.

We also saw that LEARNSF worked well, as it was able to climb to successively better selection functions, in a wide variety of situations. The performance of the final system was (surprisingly) insensitive to the initial selection function used, reaching comparable final selection functions for all 4 initial functions, in each context — *i.e.*, the values of AveErr(LEARNSF($\text{SEL}^{(\chi)}, \epsilon, \delta$)) seemed almost independent of $\text{SEL}^{(\chi)}$. (Of course, both the actual LEARNSF($\text{SEL}^{(\chi)}, \epsilon, \delta$) selection function, and the number of samples required to reach it, *did* depend on which initial $\text{SEL}^{(\chi)}$ function was used.) The values of ϵ and δ had the expected effects: In general, LEARNSF climbed slightly faster as the δ value increased, but was slightly more prone to mistakes; and LEARNSF climbed faster as ϵ increased, but performed fewer climbs. We also found LEARNSF required many more samples before termination when using sufficiently small values of ϵ. (Notice this is not particularly problematic, as here LEARNSF is, in fact, using an optimally effective selection function.) Finally, LEARNSF was also fairly efficient, requiring on average around 7.8 milliseconds per image–selection-function.

This section has described 25 different runs using the value of $\delta = 0.05$, during which LEARNSF_{Norm} climbed a total of 88 times, and terminated 24 times. It made only 1 mistake (one climbing error in $\langle \text{SEL}C, \epsilon = 0.1,\ \delta = 0.05,\ \sigma = 0.1\ m, \text{Ratio} = 0.01, \text{"Normal"} \rangle$) in the 112 occasions in which it could make a mistake (by either climbing or terminating inappropriately), which is within our expectations. (Using $\delta = 0.05$ means we would expect around 5 such mistakes.)

Furthermore, the number of samples required to decide to climb depended on the differences of the error rates of the original and successor selection functions; relatively few samples were required when that difference was large. We also found that the "normality-based" LEARNSF_{Norm} seemed to climb as effectively as LEARNSF_{HI}, but required many fewer samples. Finally, LEARNSF's behavior was also quite insensitive to the accuracy of R's estimated position, over a wide range of errors.

IV. CONCLUSION

A. *Other Applications of this Learning Approach*

This paper has described a learning algorithm, LEARNSF, that can identify which landmark-selection function (from a relatively simple set of such functions, each of the form shown in Figure 3) works well for the task of estimating an agent's position within a known environment. It also demonstrated that this algorithm works effectively, over a wide range of contexts, despite the simplicity of these functions. There are many other ways of applying this general learning approach, and algorithm, within the general context of robotic navigation.

One immediate variant is to consider a different space of selection functions, perhaps based on other types of filters and on other inputs. For example, we observed that selection functions worked differently for different hallways; *i.e.*, one that worked well for HallwayA may do poorly for HallwayB.

This suggests building selection functions out of filters that can use other arguments, such as the current hallway, the ambient light, etc.

As a second variant, recall LearnSF's evaluation function (Equation 4) is based on inaccuracy, which cannot be computed unless we know R's correct position $\mathtt{x}$. Imagine, however, that this learning algorithm only has access to unlabeled data — just $\langle \hat{\mathtt{x}}_j, \hat{\sigma}_j, \mathtt{img}_j \rangle$, but not $\mathtt{x}_j$. Here, to learn a good selection function, we would need an evaluation function that does not use the (here unavailable) $\mathtt{x}_j$ values. One proposal is to use the "size" of the covariance matrix (*i.e.*, the sum of the values on the principle diagonal), based on the standard assumption that a small covariance is associated with an accurate estimate. We could then, of course, learn the best selection function using an obvious variant of LearnSF that used this evaluation function.

Third, while our descriptions (of both RatBOT and LearnSF) deal only with "visual information", nothing in our work is specific to this sensing modality. We suspect, for example, that (an analogue of) LearnSF could learn to select useful *sonar* "landmarks".

Finally, note that our basic "probabilistic hill-climbing" LearnSF algorithm can be applied, *mutatis mutandis*, to any other task that requires searching through a discrete space of "performance elements" (such as the above space of selection functions) seeking an element whose average performance is optimal; see [11]. As such, a related learning system can be used to address many other tasks required by autonomous agents, ranging from setting low-level discrete parameters (*e.g.*, the starting window size for each image feature) to establishing various high-level "parameters" (*e.g.*, deciding which set of heuristics to use when planning the next action).

B. Contributions

To work effectively, a navigating agent must be able to determine its current location. While there are many techniques that use observed landmarks to identify an agent's position, they all depend on being able to effectively find an appropriate set of landmarks, and will exhibit degraded or unacceptable performance if the landmarks are not found, or mis-identified. We can avoid this problem by using only the subset of "good" landmarks. As it can be very difficult to determine this subset *a priori*, we present an algorithm, LearnSF, that uses a set of training samples to *learn* a function that can select the appropriate subset of the landmarks; our agent can then use this learned landmark-selection to identify which landmarks to use to robustly determine its position. We prove that this learning algorithm works effectively, both theoretically (Theorem 1) and empirically, based on real data obtained using an operational robot.

Acknowledgements

We gratefully acknowledge the many helpful comments we received from Ramesh Visvanathan and the members of the RatBOT project, especially Stephen Judd, Thomas Hancock and Long-Ji Lin. We also thank the anonymous referees, as well as the editor Marco Dorigo, for their useful, and detailed, comments.

References

[1] M. Betke and L. Gurvits, "Mobile robot localization using landmarks," *Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, IEEE, Sept. 1994.

[2] P. J. Bickel and K. A. Doksum, *Mathematical Statistics: Basic Ideas and Selected Topics.* Holden-Day, Inc., Oakland, 1977.

[3] L. Breiman, J. Friedman, J. Olshen, and C. Stone, *Classification and Regression Trees.* Wadsworth and Brooks, Monterey, CA, 1984.

[4] R. Caruana and D. Freitag, "Greedy attribute selection," *Proceedings of the Eleventh International Machine Learning Workshop*, pp. 28–36, N.J., Morgan Kaufmann, 1994.

[5] M. Case, "Single landmark navigation by mobile robots," *Proceedings of the SPIE, Conference on "Mobile Robots"*, vol. 727, pp. 231–238. SPIE — The International Society for Optical Engineering, Oct. 1986.

[6] I. Cox and G. Wilfong, eds., *Autonomous Robot Vehicles.* Springer-Verlag, 1990.

[7] T. Dean, K. Basye, and L. Kaelbling, "Uncertainty in graph-based map learning," in *Robot Learning* (J. Connel and S. Mahadevan, eds.), pp. 171–192, Kluwer Academic Publishers, 1994.

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

[9] S. P. Engelson, "Active place recognition using image signatures," *SPIE Symposium on Intelligent Robotic Systems, Sensor Fusion V*, pp. 393–404. SPIE — The International Society for Optical Engineering, 1992.

[10] C. Fennema, A. Hanson, E. Riseman, J. Beveridge, and R. Kumar, "Model-directed mobile robot navigation," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 20, no. 6, pp. 1352–1369, 1990.

[11] R. Greiner, "Probabilistic hill-climbing: Theory and applications," *Proceedings of CSCSI-92*, pp. 60–67, Vancouver, June 1992, Morgan Kaufmann.

[12] R. Greiner, "PALO: A probabilistic hill-climbing algorithm," Technical report, Siemens Corporate Research, 1995.

[13] T. Hancock and S. Judd, "Ratbot: Robot navigation using simple visual algorithms," *1993 IEEE Regional Conference on Control Systems* (T. Chang, ed.), pp. 181–184, NJIT, Aug. 1993.

[14] G. Hinton, "Connectionist learning procedures," *Artificial Intelligence*, vol. 40, no. 1-3, pp. 185–234, 1989.

[15] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[16] P. Huber, *Robust Statistics*. Wiley, NY, 1981.

[17] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," *Proceedings of the Eleventh International Machine Learning Workshop*, pp. 121–129, N.J., 1994, Morgan Kaufmann.

[18] A. Kosaka and A. C. Kak, "Fast vision-guided mobile robot navigation using model-based reasoning and prediction of uncertainties," *Computer Vision, Graphics, and Image Processing–Image Understanding*, vol. 56, no. 3, pp. 271–329, 1992.

[19] B. J. Kuipers and Y.-T. Byun, "A robust, qualitative method for robot spatial learning," *Proceedings of the Seventh National Conference on Artificial Intelligence*, pp. 774–779, Morgan Kaufmann, 1988.

[20] B. J. Kuipers and T. S. Levitt, "Navigation and mapping in large-scale space," *AI Magazine*, vol. 9, no. 2, pp. 25–43, 1988.

[21] J. E. Laird, P. S. Rosenbloom, and A. Newell, *Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies*. Kluwer Academic Press, Hingham, MA, 1986.

[22] T. S. Levitt and D. T. Lawton, "Qualitative navigation for mobile robots," *Artificial Intelligence*, vol. 44, pp. 305–360, 1990.

[23] M. Mataric, "Navigation with a rat brain: A neurobiologically-inspired model for robot spatial representation," *Proceedings of the First International Conference on Simulation of Adaptive Behavior: From Animals to Animats*, pp. 169–175, MIT Press, 1991.

[24] J. L. McClelland, D. E. Rumelhart, and the PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 2: Psychological and Biological Models, MIT Press, Cambridge, 1986.

[25] T. M. Mitchell, R. M. Keller, and S. T. Kedar-Cabelli, "Example-based generalization: A unifying view," *Machine Learning*, vol. 1, no. 1, pp. 47–80, 1986.

[26] J. R. Quinlan, *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, San Mateo, 1992.

[27] D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, eds., *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, vol. 1: Foundations, MIT Press, Cambridge, 1986.

[28] H. A. Simon, "Why should machines learn?," in *Machine Learning: An Artificial Intelligence Approach* (R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, eds.), Palo Alto, CA, Tioga Publishing Company, 1983.

[29] R. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1987.

[30] R. G. Smith, T. M. Mitchell, R. Chestek, and B. G. Buchanan, "A model for learning systems," *Proceedings of IJCAI-77*, pp. 338–343, Morgan Kaufmann, 1977.

[31] K. Sugihara, "Location of a robot using sparse visual information," *Robotics Research: The Fourth International Symposium* (R. Bolles and B. Roth, eds.), pp. 319–326, MIT Press, 1987.

[32] K. Sugihara, "Some location problems for robot navigation using a single camera," *Computer Vision, Graphics and Image Processing*, vol. 42, no. 1, pp. 112–129, 1988.

[33] B. Yamauchi and R. Beer, "Spatial learning for navigation in dynamic environment," this issue.