SPIRAL SEARCH AS AN EFFICIENT
MOBILE ROBOTIC SEARCH TECHNIQUE

SCOTT BURLINGTON AND GREGORY DUDEK
CENTER FOR INTELLIGENT MACHINES MCGILL UNIVERSITY
3480 RUE UNIVERSITY, MONTRÉAL (QC)
CANADA, H3A 2A7
{SCOTTYB, DUDEK}@CIM.MCGILL.CA

ABSTRACT. In this paper we consider the problem of robotic search: that is, having a mobile robot find a target object in an unknown environment with obstacles. As a side-effect, the robot explores the environment.

Based on previous results, we present a formal description of our problem and an algorithm to solve it. Our work is a derivative of spiral search techniques developed in more abstract contexts. We show that this algorithm has good worst-case performance, in terms of its competitive ratio. We also show experimental data validating the feasibility of our approach and typical results.

1. INTRODUCTION

Search is one of the fundamental tasks of life. Especially for artificial life. Numerous potential artificial intelligence problems have been cast as search problems and many of the most natural applications for mobile robots are search problems. Reasonable behaviour, even in natural settings, is the result of identifying and choosing an appropriate course of action from a very large set of possibilities. Humans and animals have uncanny abilities to make these decisions in an unpredictable world. In order for artificial agents to succeed, it is absolutely necessary to efficiently search through and quantify the decisions it is presented with.

In this paper, we consider the problem of automatically searching an unknown environment using a mobile robot. In contrast to classic AI search problems, this variant of search entails a penalty for moving between previously visited locations (what are traditionally described as nodes of a search tree). Prior work [1] has demonstrated theoretically optimal strategies and associated bounds on search in very simple environments. In addition, several authors have considered apatial search using heuristic rules or sensory ones, [4, 8, 7, 5, 6, 2].

Here, we elaborate on this theme and consider robotic search in complex worlds. In particular we provide an algorithm, analytical results in the form of a competitive ratio and simulation data illustrating how to perform robotic search in a planar environment with obstacles.

2. PRIOR RESULTS

2.1. **Exploration Strategies.** In this paper, we are interested primarily in searching a planar environment (or in practice a topological equivalent). The simplest deterministic methods may be those of breadth-first search [BFS] and depth-first search [DFS]. Somewhat more sophisticated methods depend on interative modifications: depth-first search [IDDF] and cost sensitive IDDF (known as

A^* search). These methods, although popular and very general, each have restrictions that impair a mobile robot. Refer to *Algorithms* by [3] for a more detailed explanation than will be presented here.

Breadth-first search is often inappropriate to robotic search tasks as it incurs high overhead in terms of the amount of storage space required and too high an overhead in moving between previously visited locales. Physical motion being much more costly than computation for a mobile robot implies that a more judicious method will be favoured highly over one such as BFS that requires intensive motion. Consider for example searching a long (endless) hallway for a water fountain (assuming there is one in the hallway somewhere). A BFS searcher would spend all its time moving back and forth between opposite ends of the hallway.

Depth-first search avoids the shortcomings of BFS. It is very efficient in terms of the storage requirements of the algorithm. More importantly, in light of the difficulty of physical motion, nodes in the search tree are expanded according to their physical proximity during DFS, rather than their logical proximity as in BFS. This implication assures us of short physical motions for discovery of new territory. In bounded environments on n vertices DFS requires at most $2n$ vertices for a complete traversal of the search tree. However in a potentially unbounded environment DFS fails badly in term of any definition of "reasonable behaviour".

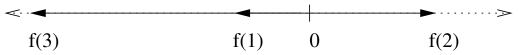
Iterated depth-first search conquers this shortcoming of DFS. IDDF maintains the advantages of DFS, while holding intact the *principle of locality* to some extent. There is a parameter which needs fixing here, when and how to we qualify the points of iteration. This is what typical A^* search attempts to ascertain by assigning a cost-based iterating point. What can we use to generate such a cost in the complex world of a mobile robot? This is the insight that we offer. We suggest on the strength of planar search optimality in [1] that an efficient method of iteration may be dictated by *logarithmic spiral search.*

2.1.1. *Searching for a Point in a Line.* Back to our problem of finding the water fountain in an endless hallway. What is the best that we can hope for to guarantee that we find a drink regardless of which direction we choose to start down first?

We can describe the solution as a function $f(i) : Z \to Z$ where $f(i)$ reports the number of steps to take to the right or left of the origin before the i-th turn. This function, provided with the *progress* condition of,

$$f(i) \geq f(i-2) + 1 \tag{1}$$

is enough to specify a solution to the problem. The *progress* condition simply ensures that our solution will indeed explore new territory each time there is a change in direction. This is assured by taking at least one more step into unexplored territory than has been taken so that the search will terminate. Since this is an alternating search, $f(i)$ and $f(i+2)$ are consecutive distances from the origin on the same end of the line.



Baeza-Yates et al. [1] show that the following function,

$$f(i) = 2^i, \quad \forall i \geq 1 \tag{2}$$

results in an assymptotically optimal solution. With this function the total distance that is walked is $2\sum_{i=1}^{\lfloor \log n \rfloor + 1} 2^i + n$, since we walk out and back on every iteration of 2^i steps except for the last

Info Given	Search Distance
Normal	n
Distance, Slope	$3n$
Distance, Horiz or Vert	$3\sqrt{2}n \approx 4.24n$
Distance	$(1+\sqrt{3}+7\pi/6)n \approx 6.39n$
Slope	$9n$
Horiz or Vert	$13.02n$
Nothing	$13.81n$

TABLE 1. Hierarchy of Search Distances bearing successive assumptions from [1]

probe, during which we walk n. This is easily seen to be bounded by $9n$ steps,

$$\begin{aligned} 2\sum_{i=1}^{\lfloor \log n \rfloor + 1} 2^i + n &< 2\left(2^{\lfloor \log n \rfloor + 2}\right) + n \\ &\leq 2\left(2^2 \cdot 2^{\lfloor \log n \rfloor}\right) + n \\ &= 2^3 n + n \\ &= 9n \end{aligned}$$

with a little bit of slack in the first inequality. In fact there exist algorithms which perform at $9n - \Theta(\log n)^i$, $\forall i$.

This particular problem, in a 2 dimensional setting is equivalent to looking for a line in a plane knowing only the slope of the line (we are along the normal for the intersection point).

2.1.2. *Searching the Plane.* It is worthwhile now to consider search techniques in the plane. The objective is to find some better method for searching in the plane. Bearing in mind that the environment is unknown (perhaps unbounded) in addition to the real problem of unreliable localization. Given the hierarchy collected by Beaza-Yates, Culberson and Rawlins in 2.1.2 we can get a flavour of how we may do better than DFS in a real environment.

Under the assumption that we have to pay a significant cost to "probe" in our environment proportional to the distance of the probe from our current location there may be better ways to search. Table 2.1.2 shows a collection of known best results for such search problems under the restrictions given.

2.1.3. *Logarithmic Spirals.* It is suggested by the results of 2.1.2 and by the rest of the findings in [1] that an appropriate way to proceed is through the continued application of these *logarithmic spirals*. We have chosen to use this approach to choose our iterating values for our IDDF.

There are a few things to mention about our representing a map by a tree structure. We make the representation of a polygonal environment by the search path that we would follow as seen from a perspective envisioning the environment as a whole. We presume that (in a hallway) the details of the search across the hallway, from side to side, are uninteresting as opposed to the progress of the search down the hallway. This perspective on the search progress allows a representation of the polygonal environment as the set of search paths that are contained therein. Figure 1 shows how a star-shaped[1] polygon is represented as a set of concurrent (co-terminal) rays.

In an unbounded region our method still applies. The lack of a strict polygonal bound is not the issue here. In reality, with a limited range sensor, searching a local environment that is larger than the sensor range requires that we comb through the environment according to some tesselating pattern. The impact of the details of this part of the search will only turn up in the low order terms of the overall performance.

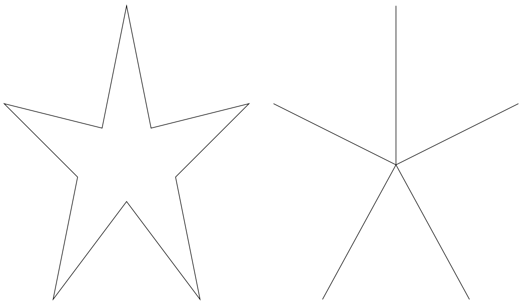


FIGURE 1. The representation of a star shaped polygon by a set of rays all co-terminal. This can be thought of as a potential search path.

Thence the definition of m-concurrent would be such a collection of m rays. If we allow smooth deformations of this base search shape without allowing rays to cross over (creating cycles) then we can slacken this definition of concurrent rays to include homeomorphisms of concurrent rays.

Alas, all worlds are not this simple. It is possible that a path, and hence the associated ray, bifurcate. A discussion on which. The lifts the requirement that the rays be co-terminal. Even if the rays are allowed to bifurcate at points not at the origin, then the interpretation of spirally expanding search can still be applied.

A system capable of detecting cycles in the search tree would necessitates accurate odometry, which is beyond the scope of this simulation.

2.1.4. *M Concurrent Rays.* The case for searching m-concurrent rays is similar to that of searching for a point in a line, but offers further insight into that same problem. The assumptions are that the robot is begins at the intersection of m rays and endeavours to find a target located on one of the rays. If the distance to the target is known, but not which ray the target lies within (comparable to knowing distance but not direction) then the obvious $(2m-1)n$ algorithm is optimal. If the distance is unknown, then this becomes the counterpart of the point-in-a-line problem and yields some interesting results.

It is in star-shaped environments [hulls] that this analysis can be directly adapted to provide both an algorithm and performance bounds. Given that there are m rays, the order is not important since the names of any of the rays are simple labels and can be permuted, so cyclical indexing is fine from $\{m \in Z | m = 1, 2, \dots\}$. Later in the case of non-concurrent, or bifurcating, paths this issue come up

---

[1] A star-shaped polygon is one that contains a subregion known as a *kernel*, such that every point in the polygon is visible from any point in the kernel.

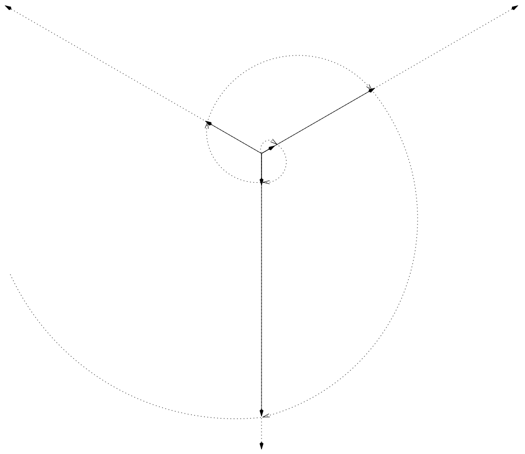


FIGURE 2. The path traced by $f(i) = \left(\frac{3}{2}\right)^i$ intersecting with $m = 3$ concurrent rays.

again, since the order of visitation will make a difference in the overall performance. Again we see that in order to study the problem, we have the *progress requirement* that the function, $f(i)$, which relates how far to walk from the origin before the $i-th$ turn is,

$$\begin{aligned} f(i) &= f(i-m)+1 \qquad \forall\, i \geq 1, \\ \text{where } f(-j) &= 0\,\forall\, 0 \leq j \leq m-1 \end{aligned} \tag{3}$$

This has been termed by Baeza-Yates et al. as *generalized linear spiral search*, since the points we are interested in are the intersection points of the two sets in figure 2 and the actual spiral set which is the set of dotted points forming the logarithmic spiral depicted. The generalised linear spiral search algorithm is defined in a similar manner the linear spiral search from equation 2 as the number of steps to walk before the ith turn starting from the origin. Using the exploration function assures,

$$f(i) = \left(\frac{m}{m-1}\right)^i \qquad \forall\, i \geq 1. \tag{4}$$

optimal performance as shown in [1]. Thus we are instructed about what base to choose for the denominator in the underlying function. In the point-in-a-line case ($m = 2$) we have $f(i) = 2^i$, for the case of $m = 5$, for instance, we see that the function we should use is $f(i) = (5/4)^i$. This comes naturally from the proof (see [1]), with a worst performance ratio of,

$$1 + 2\frac{m^m}{(m-1)^{m-1}} \qquad \text{(for large } m\text{)}. \tag{5}$$

This is a *competitive ratio*. This is generated by determining the worst that this algorithm can possibly do in an environment chosen by an unfriendly arbitrator versus the best possible solution in that same environment.

3. OPTIMAL SEARCHING

3.1. Searching M-Non-Concurrent Rays. Demanding that we have concurrent rays is too much of a constraint to place on any system that we expect to use in the real world. Indeed we would like to be able to use the techniques that the theory provides for us, but it must be adapted to actual use in many ways.

Concurrency of the search rays facilitates the analysis of the algorithm but is infeasible in practice. Typically we wish to use our robot to perform well in more complex environments. In particular the pathways that the robot is traversing may bifurcate. Figure 3 gives some indication how non-concurrent rays may model this type of world.

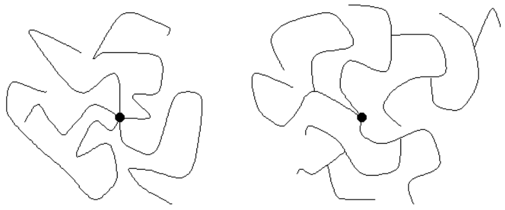


FIGURE 3. Examples of environments depicting concurrent and bifurcating paths.

The way that we propose to deal with this issue is to keep track of the base of each ray. That is, keep track of the distance travelled from the origin of each bifurcation in the ray structure. Since we have an exponential search algorithm, $f(i) = m^i$, the distance to the base of the ray is fixed and linear, and hence will be a low order term in the asymptotic performance of the algorithm.

4. THE EXPERIMENTAL APPROACH

4.1. **The Exploration.** In this section we present an experimental verification of the performance of our algorithm. This not only validates the worst case analysis, but indicates the types of results that can be expected in practice.

In the simulation presented here, we posit the existence of an idealised range sensor *with limited range*. The simulator returns accurate readings radially for a distance roughly equivalent to eight times the radius of the virtual agent. Our results are only weakly dependent on the faithfulness of the sensor in order to have an aspect of scalability to an actual robot.

The environment will grow in expanding concentric circles centered at the original location of the robot. The radial distances will reflect the current "search depth" that the robot is operating at. The concept that will be embedded here is the "Searching M-Concurrent Rays" idea, but with much more functional flexibility. Thus the robot wakes and begins looking about along these level curves of search depth. Initially the number of rays is 0. Figure 4 indicates the scenario thus far.

As the exploration begins the robot will work outwards along level curves. When an obstacle is encountered the path bifurcates to accommodate this and the number of concurrent rays is then incremented dynamically. Exploration continues as normal, only under the new circumstance of an increased number of rays. This impacts the distance down the current ray that we will search according to the theoretical optimal search ratio of $f(i) = \left(\frac{m}{m-1}\right)^i$.

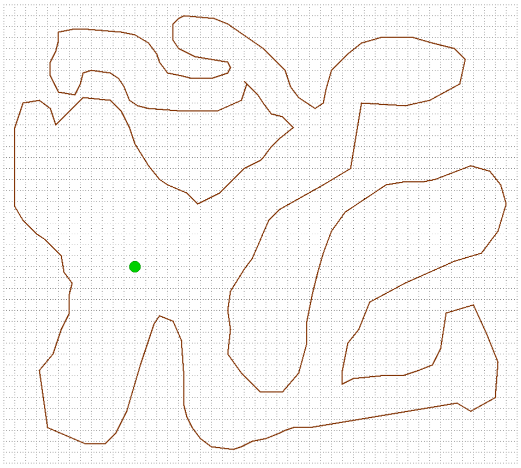


FIGURE 4. A possible scenario. The robot upon waking, however, knows nothing at all of the environment in which it finds itself.

As we progress through the environment, we see where the increments take place in figure 5. These are labelled in the diagram for clarity albeit incorrectly. They are incorrect in reality since *decrements* also occur when rays of the search have been exhausted[2].

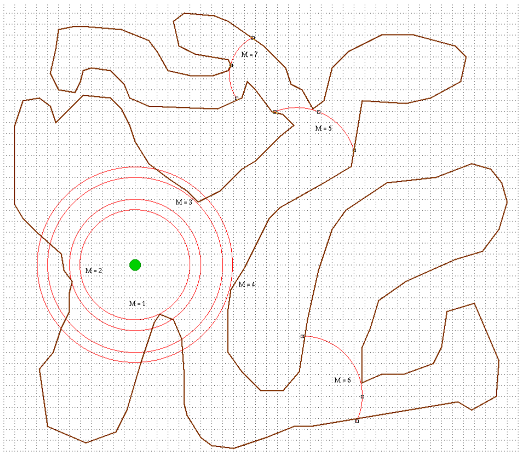


FIGURE 5. The concentric regions indicate the explored area at the time when the number of concurrent rays in increased. This occurs when an obstacle is encountered at a point in the middle of the exploration frontier within an area that was presumed to be a single ray, *ie* at a ray bifurcation.

---

[2] For instance the lower left region would certainly have been completely explored before the $M = 5$ vertex would be discovered and hence would have resulted in removal from the list of open rays to explore along with a decrement in the number of rays. So that by the time the $M = 5$ vertex is discovered there will actually be at most 3 active search rays, however this is a pedantic issue.

Thus we have an algorithm for extending the frontier of our search and we know that it is optimal under the ideal circumstance of concurrent rays. We have a method to adapt it to the non-concurrent situation that we are sure to run into in the real world, as our search paths bifurcate and unify, in order to guide us in expanding our knowledge of the environment in a locality-centric manner.

5. SIMULATION

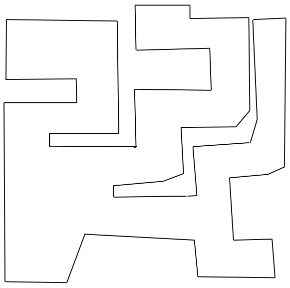
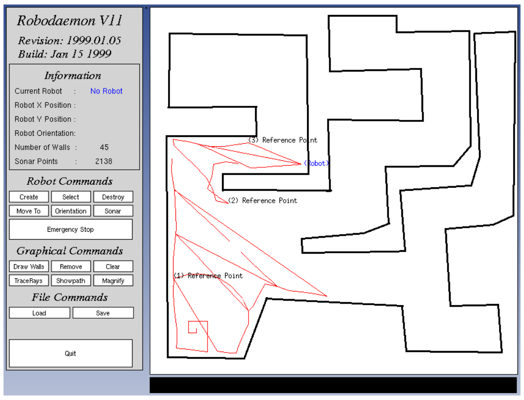


FIGURE 6. The original map which was used in the simulation.

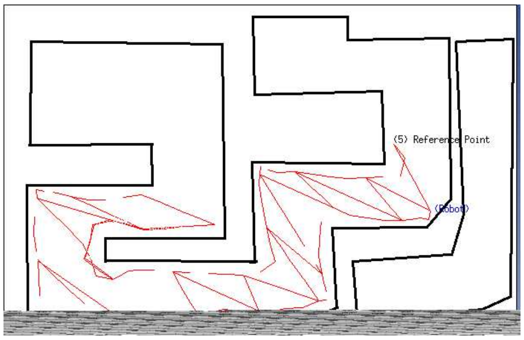
Pictured are a series of iterations of a single run of our algorithm on a hand drawn, preconstructed map representing perhaps a set of hallways in a building. The scope of the entire map is roughly 200× that of a real robot, for perspective sake. The original map is shown in figure 6.



Robodaemon V11
Revision: 1999.01.05
Build: Jan 15 1999

Information
Current Robot : No Robot
Robot X Position :
Robot Y Position :
Robot Orientation:
Number of Walls : 45
Sonar Points : 2138

The rest of the screen shots indicate the progress of the simulated robot at the end of each successive iteration. The extent of the map is not so large that a second cycle results in complete traversal of the environment. This brief exhibit should suffice to indicate the nature of the progression of the algorithm in simulation.


(5) Reference Point
(Robot)

trajectory through the environment, and that it can estimate its current position at any time. In fact, the explorati
could even be manually controlled.

Since our objective is to construct a virtual environment that appears subjectively realistic to human observers
our approach is inspired by models of human visual environment exploration. In particular, human exploration
either an environment or an image is driven by a shifting attentional "spotlight". In building models of human attention, substantial research has been devoted to the computational mechanism involved. We concentrate here, rather, on the *locations* to which attention is driven. One class of attentional processing is characterised by visua
saccades to areas of high curvature, or sharp angles. More generally, things which are "different" or inconsistent with their surroundings tend to attract visual attention. Thus, our approach is to compute a map over an imag
(perhaps a 3-dimensional image) of how much each point attracts attention. Selecting the extrema of this
provide a set of attentional features.

Our computational procedure for defining features is dependent on finding edges in the image. Edge structure h
been used extensively in computational vision for several reasons including the apparent psychophysical relevance
edge structures, and because there is evidence that that visual edges tend to be highly correlated with the projection
onto the image of real physical events in the world (e.g. object boundaries, markings). Higher-level features such as large-scale edges, curves, circles, or corners, are difficult to detect robustly, and organising edge elements produced
by an edge operator (*edgels*) into semantic tokens is a notoriously difficult problem. Edgels, however, have th
advantage of being robust to variations in illumination, and are strongly suggestive of actual geometric structures
the environment.

It is with this in mind that we have formulated a metric for visual attention based on edgel density. For examp
areas with rich geometric content will have a high edgel density. To focus attention at locales that are notable, ou
attention mechanism is driven to locations where the local edgel density differs substantially from the mean edg
density.

2. APPROACH

The set of all possible views or images obtainable from a fixed location in the environment can be described as
viewing sphere or spherical image. More specifically, for every ray projected from a location in a direction
along the unit sphere S we get an associated intensity from the environment. This transformation can be expresse
as.

$$M_D : R \; S \to R \tag{1}$$

or

$$M_D(x, y, z, \theta, \phi) \mapsto i \tag{2}$$

where (x, y, z) are spatial coordinates, (θ, ϕ) refer to the orientation of a light ray, and i is the intensity obs
This parameterisation of light rays is related to the *light ray manifold* defined by Langer and Zucker and the
Lumigraph.

In our particular case, we have a camera mounted on a pan and tilt unit at a fixed location on a mobile robot.
For the purposes of this paper, let us assume that the robot is constrained to a flat floor, and thus we restrict the
camera to a 2-dimensional plane. This constrains the origin of the ray to R, we have the idealised 2-D observer
in a 3-D world.

$$M_D : R \; S \to i \tag{3}$$

or

$$M_D(x, y, \theta, \phi) \mapsto i. \tag{4}$$

A minor variation is the case of an idealised camera which only pans, which is the case for much image-based V
Since we are now dealing with a camera, as opposed to a single ray, the result of the transformation is an image
a set intensities given by a cone about the camera direction.

$$M_C : R \; S \to R \tag{5}$$

where $M_C(x, y, \theta) = I$ now denotes a pixel-indexed image $I(a, b)$ implicitly dependent on the field of view of t camera. Each pixel is, of course, also specified by Eq. 4. An entire [illegible] panoramic image $I(x, y)$ where e pixel is a ray corresponding to Eq. 4 is given by

$$M_S : R \to R^n \qquad (6)$$

where n is the number of pixels in the image, thus leading to a parameterisation of a [illegible] of images $I(a, b)$ whose individual pixels implicitly depend on the viewing parameters of the camera.

It will prove convenient to consider the problem of directing attention to a sub-image of a large image. In thi case, we specify its pixels as $I_{x,y}(\theta, \phi)$ so that it notationally resembles the specification of a pixel from a set o panoramas in Eq. 6.

$$I_{x,y}(\theta, \phi) = I(x + \theta, y + \phi) \qquad (7)$$

An image-based VR model is founded on approximating a continuous set of spherical images given by Eq. 6 fro some (discrete) set of representative points in the environment. In practice, image based VR allows a user to between specific locations and look in (almost) any viewing direction from any of these locations.

To construct an image-based model, we must first gather a set of images from each point $P_i(x_i, y_i)$ in the environment we wish to model. These images are then tiled into a mosaic which can be subsequently mapped onto viewing volume. In practice, the mosaic is produced by "stitching" or fusing all of the images from one sample location into one composite image. This involves registering consecutive images with one another using methods analogous to those used in stereo correspondence. In practice, this implies that camera rotation should be abou the nodal point of the camera, that the scene should be static (or the sequence should be acquired as quickly a possible), that lighting should remain constant, and that camera motion and parameters must be minimised. Th types of constraints, while conceptually trivial, substantially complicate the manual acquisition of image data VR mosaics.

The shape of the panoramic image that is used can vary. both spherical and cylindrical projections have attractiv properties, with cylindrical projections being predominant in existing applications. This gives the viewer a limit viewing hemisphere, in that information is lacking at the vertical extremes. For any viewing vector $v = (r, \theta,$ where r represents the zoom factor, and θ, ϕ are the Euler angles, we can then map the appropriate field of view onto a planar surface for display. The sampling location P_i in the environment now encompasses all possible viewing directions, within the constraints of the cylindrical map, and is defined as a [illegible]. To construct a navigable environment, several such nodes must be created, as well as a method defined for inter-nodal movement. In practic one can define [illegible] within the images to create such links in the nodal graph. The desired result is to obtain graph composed of such nodes which encompass all the distinctive regions in the environment, as well as a means navigating smoothly between them. That is, if two nodes are chosen which have no overlapping visual informatio it would be desirable to have a node in between which would allow a smooth transition. It is the automated selectio of the nodal positions P_i which we will now develop further.

2.1. The 2-D Analogue

The logistics of acquiring sufficient data to evaluate alternative viewpoint selection algorithms in real environment is substantial. In order to readily test alternative methodologies in a controlled and pragmatic setting, we have use a analogue of the full problem based on selecting regions of interest within a two dimensional image. This also h the advantage of being closely related to existing models of human attentional processing.

As noted above, the problem of selecting a sub-image from within a larger image can be expressed in the form o Eq. 7. If we consider spherical panoramas under orthogonal projection (i.e. without zooming due to perspective geometry and semantics of windowing within a 2-D image and real panorama selection from a set of spherical ima are very similar. Thus, we can evaluate panoramic viewpoint selection methods by looking at their performance region-of-interest selectors in 2-D images.

3. METHODOLOGY

To encode an environment using image-based VR nodes, we must first determine which locations in the environmen that is, which viewing cylinders from the set of all those possible, are most worthy of retention. We accompli this by establishing which viewing spheres are most *interesting*, where interest is measured by the extent to which a location attracts visual attention.

To construct our attention operator, we will consider the case of simple two dimensional images. In principl we would like to exploit image geometry and semantics. Work in human psychophysics suggests that various type of geometric structure such as line endings, oriented line segments, or curves "pop out" of an image when the are different from the rest of the scene. Since edge linking and segmentation remain open problems in a generic context, we settle, instead, on exploiting the variations in the *distribution* of edge *elements* as cues to attentional fixation. Since both psychophysics and intuition suggest that we wish to concentrate on regions that are unusua or distinctive, we can evaluate the extent to which regions of an image differ from the mean. We begin with a image $I_{x,y}(a,b)$ and compute the binary edge map $E(I_{x,y}(a,b))$. A generic and computable metric for image content is local edge-element density. We compute this by convolving the image with a windowing operator to obtain the local edgel density $D(i,j)$. While in principle a Gaussian windowing function is suitable, in the interest of real- performance we use a square-wave kernel of size AB.

$$D(i,j) = \frac{1}{4AB}\int_{j-B}^{j+B}\int_{i-A}^{i+A} E(I_{x,y}(a,b))\,da\,db \qquad (8)$$

Note that this can be computed in the context of images from either Eq. 6 or 7.

The interest value of a point is then given by the absolute deviation from the mean local edgel density $\bar{D}$.

$$\mathcal{D}(i,j) = |\bar{D} - D(i,j)| \qquad (9)$$

We then sort these points, P_i, based on their absolute deviation $\mathcal{D}$ from the mean to provide a list of the K most-interesting locations for which nodes are created. In practice, additional constraints such as assuring no two poi are too close together are desirable. For the purposes of the present synopsis, we will simply assure that no tw regions on the list of interesting places are permitted to overlap. If they do, we evaluate pairwise combinations a delete the less interesting of the two.

In order to illustrate and experimentally exemplify our approach in this paper, we first model the environmen using a single image as in Eq. 7.

4. IMPLEMENTATION

Our approach to attention described above assumes that the statistics of the edge distribution of the environmen are fully available when decisions are made. Such a paradigm is sometimes referred to as an *off-line algorithm*. this context, it involves an analysis of image data from every point in the environment, followed by a selection of t best few locations for which panoramic image nodes are subsequently created and interconnected. Examples of t performance of this approach are presented in Section 5.

4.1. On-line Viewpoint Selection with ff-backtracking

In practice, as the robot moves through the environment it would be highly advantageous to make decisions when locations are encountered so that there is no need to either acquire and store immense amounts of data, or backtrac to re-visit selected locations to obtain the panoramic images. To do this, nodes must be selected based only on partial information of the statistical distribution of image content over the environment giving rise to an *on-l* *algorithm*. Assuming that the off-line algorithm performs well, we seek an on-line algorithm whose performance is good approximation of that obtained with the off-line method.

We can assure that the on-line algorithm exhibits arbitrarily good performance, as compared to the ideal of th off-line algorithm, by permitting the robot to backtrack. We can define the *forward interest* of a point from partial information as

$$D_t(i,j) - \bar{D}_t \quad D(i,j)| \qquad (10)$$

where the subscript t denotes statistics computed from the initial fraction t ∈ (0,1] of the entire data set. W define [illegible] as a variant of the off-line algorithm such that the best K non-overlapping points are selected as the exploration proceeds. As each point is selected, a corresponding panoram node is constructed. Density values are also stored for all other points visited. As the exploration proceeds t increa and the forward interest values of previously visited locations may evolve. If a prior unselected point [illegible] becomes more interesting than one of the K selected points, the robot backtracks and uses it instead of the point it replaces. Clearly, the performance (in terms of points selected) of this algorithm approaches the ideal as ff approaches one.

4.2. Single Image

Due to the complexity of testing a system involving a mobile robot, we have implemented a system capable of analysing a single image representing the entire environment as described in section 2. For the off-line version the problem, we consider moving a box of constant size over the entire image and calculating the edgel densities, a discussed in section 3. This enables us to easily compute the required statistics to choose the regions of the ima which have edge densities which are the largest absolute deviations. One key point of using the absolute variance exemplified when considering a textured image. In this case, it is regions which are large deviations [illegible] the mea which will be of interest.

For the on-line version of the problem, we move a rectangular window over a sub-part of the image and calculate the approximations incrementally. This can provide a compromise that achieves good performance with limite storage requirements. Unfortunately, the characteristics and performance of the on-line algorithm are outside scope of this paper.

4.3. Environmental VR

Our approach to environmental VR is based on having a mobile robot traverse the 2-dimensional environment to be mapped.[7] It is independent of the traversal strategy, although it assumes that the topology of the trajectory is known so that the multi-node model can be constructed. In addition, in order to avoid closely-spaced sample nodes an approximate [illegible] estimate of distance is desirable (e.g. from odometry). Our technique has been impleme to function with robots from both RWI and Nomadic Technologies, both of which provide odometry data whose accuracy is far in excess of our requirements.

The robot we used has a digital camera mounted on a pan and tilt unit, and is capable of translating and rotating thus giving a the final camera position a total of 4 degrees of freedom (DOF). In practice, we have fixed the tilt o camera, and therefore define an (x, y, θ) location for each image taken in the environment. During the explora the robot periodically stops and gathers a set of images in one or more orientations, so that they may be evaluated for candidacy within the set of salient points. The most complete sampling of the environment demands a set images which collectively cover the view in every direction, that is the data presented in this paper.

The difference between the off-line and on-line implementations is apparent at this point. In the case of th on-line algorithm the robot must immediately decide whether or not to create a VR node and obtain additional data if necessary. If the current location is not selected, the image data acquired can be discarded. In the off-line ca on the other hand, the robot must retain all of the data. The final decision regarding node selection is made afte the exploration is complete. Figure 2 shows a flow chart of the two approaches. In the off-line case, the robot mus acquire all the additional data for a panorama at each location, or return to the selected locations after selectio is complete. These nodes are then incorporated together to form a complete image-based VR representation of th previously unknown environment.

The need for additional images for panorama creation, above and beyond those needed for viewpoint selection, ha several sources. Under-sampling a location may be sufficient for interest determination as a result of task constrai In addition, several existing algorithms for stitching images to generate a panorama require substantial overlap (the order of 50 [illegible]) between source images. Since an overlap of this size simply multiplies the number

---

[7] [illegible]

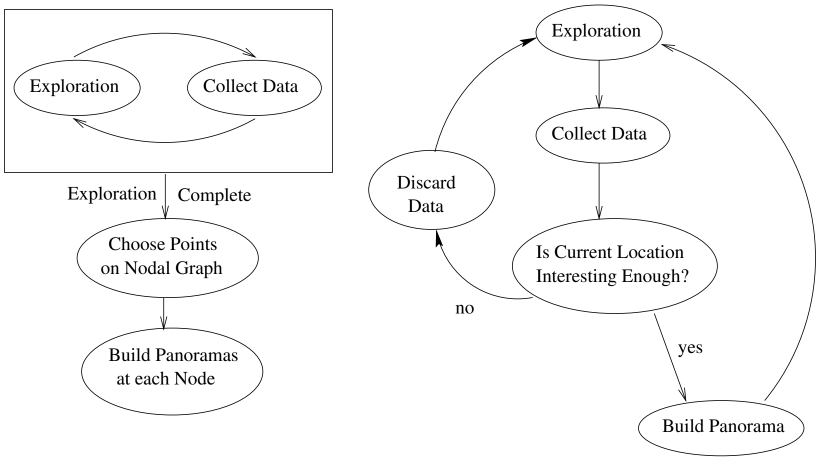


Figure 2. Flow charts for the off-line (left) and on-line (right) implementations of the algorithm.

of times we have seen artifacts in the environment, it does not have an effect on the final outcome of the chose locations. Therefore taking images which barely overlap allows the amount of data stored to be reduced.

5. EXPERIMENTS

5.1. Single Image

For the purposes of our single image experiments, we acquired several photographs of natural scenes using a digi camera. The goal of these experiments was to choose points in the images which most closely matched the poin which would attract the focus of a human observer. The software we developed for this analysis convolved the imag using the kernel described in Eq. 8, and subsequently sorted the resulting interest points by their P descending absolute deviation. A simple heuristic was used to guarantee that there was no overlap in the resultant selection For the images shown in Fig. 3, a kernel of size 90 90 *pixels was used.*

In order to demonstrate the advantages of our interest operator, we chose photographs which had interest point which were well defined semantically, but which contained substantial texture, thus allowing them to be potentiall confusing to other types of operators.

We believe that the locations chosen by our interest operator were in fact the correct ones, that is, that they match most closely what would attract the attention of a human observer. Although the formal substantiation o these results is beyond the scope of this paper, we are pursuing this objective.

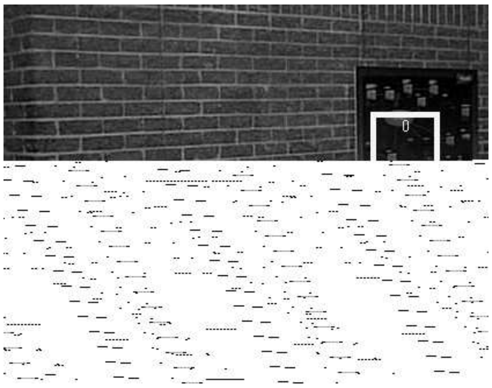
5.2. Environmental VR

We have examined the performance of our environmental sampling technique in the context of a small fully-control test environment that can be manipulated at will (see Fig. 1). In the configuration shown here, the environ resembled an office-like world or a section of a maze. We used a small mobile robot with a top-mounted camera to navigate this environment.[x] *The field of view of the camera mounted on the pan and tilt unit of the robot was such that the tops of the images were just slightly below the tops of the walls of the test maze. This was to ensure tha no spurious image data was brought into the experiment. Two pictures were mounted on the inside of the walls a different locations, and a few small objects were dispersed throughout the environment. At each of the junctio there were views into the "open world" which were considerably different from the somewhat constrained interna environment.*

The robot followed the path shown in Fig. 4, and collected 35 *images from* 12 *vantage points for a total of* 420 *images gathered within the environment. In this experiment the robot used a simple path-planner to guide*

---

[x] An alternative, larger, robot is used for conventional environments.



odometry) is a real problem. This suggests that the optimal search strategy may involve returning to previously-explored regions slightly more frequently that might be suggested by the analysis presented here.

A related issue is that most real range sensors have an accuracy that diminishes with distance, and hence can be described in probabilistic terms. In some cases it may be preferable to search more deeply in regions were visibility is obstructed in order to assure a uniform coverage of free space along different paths. This suggests that probabilistic search may have a slightly different form from that described here.

Finally, we are currently examining parallel implementations of search for use by multiple robots: this leads to issues to how to subdivide the problem and how and when to merge partial results.

REFERENCES

1. Ricardo A. Baeza-Yates, Joseph C. Culberson, and Gregory J. E. Rawlins, *Searching in the plane*, Information and Computation **106** (1993), no. 2, 234–252.
2. Tucker Balch and Ronald C. Arkin, *Communication in reactive multiagent robotic systems*, Autonomous Robots **1** (1994), no. 1, 27–52.
3. Thomas Cormen, Charles Leiserson, and Ronald Rivest, *Introduction to algorithms*, McGraw Hill, 1990.
4. Amitava Datta and Christian Icking, *Competitive Searching in a Generalized Street*, Proceedings of the Tenth Annual Symposium on Computational Geometry (Stony Brook, NY), ACM Press, June 6–8 1994, pp. 175–182.
5. Schmuel Gal, *Search games*, Academic Press, New York, 1980.
6. J. R. Isbell, *An optimal search pattern*, Naval Research Logistics Quarterly **8** (1957), 357–360.
7. Christos H. Papadimitriou and Mihalis Yannakakis, *Shortest paths without a map*, Theoretical Computer Science **84** (1991), no. 1, 127–150.
8. T. C. Schelling, *The strategy of conflict*, Harvard University Press, Cambridge, 1960.