2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

A Competitive Online Algorithm for Exploring a Solar Map

Patrick A. Plonski and Volkan Isler

***Abstract*— In this paper, we study the problem of quickly building the 3D model of an outdoor environment from measurements obtained by a robot equipped with a solar panel. The robot knows the angle of the sun and the locations of the objects in the environment. It does not know, however, the height of the objects. For example, it might be possible to use satellite images to obtain locations of trees in a field but not their heights.**

In order to compute the height of an object, the robot must find the projection of the object's highest point. This is where the shadow of the object ends. The robot can find it by tracing the shadow (moving parallel to the sun) until the measurement switches from shadow to sun or vice versa. The robot's goal is to compute the height of every object as quickly as possible using only solar measurements.

We formulate this as an online optimization problem. The optimal offline algorithm is given by the Traveling Salesman path of the transition points. The robot does not know these locations a priori. It must search for each of them. We present an algorithm with the property that for n objects, our distance traveled is guaranteed to be within a factor $\mathcal{O}(\log n)$ of this optimal offline tour. In addition to analytical proofs, we demonstrate the algorithm with simulations using solar data collected from field experiments, and examine its performance for uniformly distributed sites.

I. Introduction

Mobile outdoor robots have been shown to be useful for a variety of exploration and environmental monitoring applications. However, their long-term feasibility is often constrained by limited battery life. A common approach used to address this problem is the addition of solar photovoltaic panels to the robot.

Previously the utility of solar-aware path planning has been demonstrated. In [4] we constructed a solar map by using only previous measurements of solar power associated with positions. This method implicitly relied on having densely sampled the relevant environment prior to the path being planned.

In this work we address the exploration component necessary to generate an accurate solar map. Any coverage pattern or random walk will eventually explore everything, however when a robot executes these strategies it might have to travel a great distance. We would like some method to ensure that we expend the least energy and time in the exploration step. To this end, we use a quadtree-based exploration strategy which uses observations of sun and shade to refine its estimate of the heights of shadow casting objects in the environment. We demonstrate that this strategy obtains a bounded worst-case competitive ratio between the distance traveled using our exploration algorithm and the distance traveled with the optimal algorithm. The bound is logarithmic in the number of critical points that must be observed.

P. A. Plonski and V. Isler are with the Department of Computer Science and Engineering, University of Minnesota, 200 Union Street SE, Minneapolis MN 55455 USA. e-mail: {plonski, isler}@cs.umn.edu.

A. Related Work

Information about solar energy collected from the environment has been used to inform path planning for long term missions. In open environments, such as in Antarctica [5] or on the open ocean [6], the energy depends only on the sun position. However the TEMPEST mission-level path planner [7] uses the known position of the sun together with known nearby terrain to perform raytracing and compute a shadow map, which is then used to compute the net energy of any potential path. Given that sensing the terrain in high detail is a difficult task, in [4] we examined the problem of learning the solar map using only measurements of position and solar current. This technique relied on having enough of these measurements, spread out over the relevant positions. In this work we look to ensure this is the case.

Analyzing an exploration strategy is challenging because we don't know what the environment looks like. A common method in the literature is to look at the competitive ratio between the online strategy and an optimal offline strategy with full information. This allows us to reason about the actual worst case environment to explore, instead of merely making arguments on expectation. Some recent examples of exploration strategies analyzed using competitive ratios include [1], [2].

II. Problem Statement

We parameterize the problem of fully exploring the shadow map as the following: We have n objects in our environment with known position but unknown height. To be able to claim that we have fully explored the environment, we need to precisely determine each of these n heights. From the heights we will be able to obtain the shadow map for any time of day. We know the angle of the sun at any moment (see [3]), so to find the height of an object it suffices to find the length of its shadow. This is equivalent to finding the critical point where a solar panel will switch from collecting direct insolation to only collecting diffuse insolation. We call these critical points *sites* $\{s_1, s_2, \ldots, s_n\} = \mathcal{S}$. We assume there are upper and lower bounds on the object heights, so each site therefore lies on a known line segment. We also assume that during the exploration step the sun does not appreciably change position, thus all line segments are parallel. Without loss of generality, therefore, we perform a coordinate transform based on the angle of the sun. The x positions of the sites, $\{s_1^x, s_2^x, \ldots, s_n^x\} = \mathcal{S}^x$, are known a

978-1-4799-3684-7/14/$31.00 ©2014 IEEE

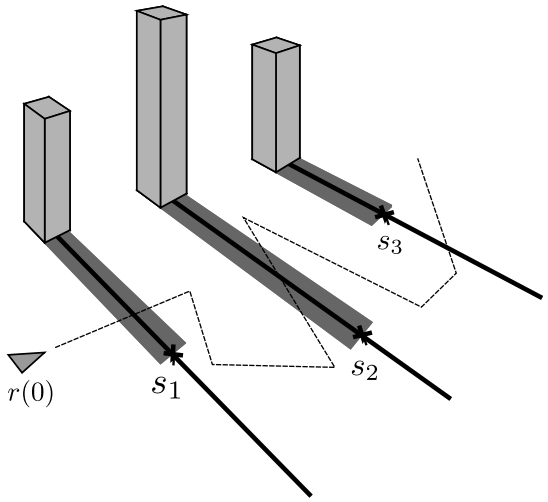

Fig. 1: Here is an example of our problem setup. Each object has known position on the plane but unknown height. To find the height the corresponding site in $\mathcal{S}$ must be visited. The positions of $\mathcal{S}$ are unknown to the robot, however a bound on the height of each object constrains its corresponding site to lie on a line segment parallel to the sun.

priori, and it is only the y positions, $\{s_1^y, s_2^y, \ldots, s_n^y\} = \mathcal{S}^y$ that are uncertain. However, there is an initial estimate of the range that each s_i^y lies in, denoted y_i^u and y_i^ℓ for the upper and lower limits, respectively. Without loss of generality, $\forall s_i^x \in \mathcal{S}^x, s_i^x \geq s_{i-1}^x$. See Figure 1 for an example problem setup for an environment with three objects.

We assume our exploring robot r is a holonomic point robot. We are primarily interested in the case where r begins outside the x range, so we fix the initial x coordinate of r at 0 and require that all $\mathcal{S}^x$ are positive. As r explores the environment, it learns more about the true positions of $\mathcal{S}$. We assume that walking farther away from an object than a critical point will result in sun, and walking closer will result in shade; $(s_i^x \in \mathcal{S}^x) = r^x$, the robot learns whether s_i^y or r^y is greater, by measuring the solar insolation on its panel.

Our aim is to find an algorithm for r that explores the environment and visits all of $\mathcal{S}$. We do not require that r return back to the starting position; the mission ends when the last site is visited. To judge the quality of our exploration algorithm we do not assume any probability distribution but rather consider the worst case competitive ratio between the distance r travels when commanded by our algorithm, $\ell(\mathcal{R})$, and the distance r would travel were it commanded by an optimal offline algorithm that simply solves the traveling salesman path for the starting position of r and the true positions of the sites. We denote the TSP solution as $\mathcal{R}^*$, and we can formally state now that our goal is to minimize the following competitive ratio:

$$C = \max_{\mathcal{S}} \left(\frac{\ell(\mathcal{R})}{\ell(\mathcal{R}^*)} \right)$$

In this work we present an exploration algorithm for r and demonstrate that its worst case competitive ratio is logarithmic in the number of objects in the environment n.

III. X-Sweep Strategy

Before we introduce our full algorithm we first present a basic naíve strategy which has poor worst-case performance but which we will later use as a subroutine. The X-Sweep strategy simply visits all the input line segments in order from lowest x-coordinate to highest. It goes to the closest point on the segment first and walks along it until it reaches the site, then moves on until it has visited every site.

Clearly if there is not much difference between $\max(\mathcal{S}^y)$ and $\min(\mathcal{S}^y)$, the X-Sweep Strategy will perform close to optimal. However, it is easy to construct a situation where X-Sweep performs poorly. Suppose the $s_j^y \in \mathcal{S}^y$ alternate between 0 and some large value m, and furthermore suppose $d = x_n - x_0$ is very small compared with m. In this case the optimal strategy cuts along $y = 0$, then cuts back along $y = m$, with a total path length close to m. In comparison, X-Sweep walks up and down the line segments and its total path length is close to nm. Therefore the worst case competitive ratio for X-Sweep is at least as bad as $\mathcal{O}(n)$.

IV. Quad-Explore Strategy

We saw that the X-Sweep strategy can have a competitive ratio as bad as $O(n)$. This is because it can backtrack very far with each walk along a segment. To obtain a better bound we need a method to balance walking along the segments with cutting across the segments. One such method is our Quad-Explore strategy: it cuts across segments when they are long and executes X-sweep once the belief line segments are short enough that performing X-sweep is guaranteed inexpensive.

A. Description

Here is the description of Quad-Explore.

First we consider the case where $d = x_n - x_1 = max(\mathcal{Y}^u) - min(\mathcal{Y}^\ell)$. That is, the arena is square.

Quad-Explore is based on constructing a recursive quadtree structure that we call Q. See Figure 2 for an example of the structure of Q. The structure is constructed as it is traversed, depth first. Each node q_i of Q consists of a square in the plane $A(q_i)$, a pointer to the parent, and pointers to the possibly four children $\{c_1, \ldots\} = \mathcal{C}(q_i), 0 \leq |\mathcal{C}| \leq 4$. Nodes are numbered as they are created, and q_0 corresponds with the entire arena. In addition to Q the robot must maintain knowledge about its position, and its belief $\mathcal{Y}^\ell$ and $\mathcal{Y}^u$ on the lowest and highest possible values for $\mathcal{S}^y$ consistent with the initial belief and the online observations.

For each q_i the algorithm generates four candidate squares $\{a_1, a_2, a_3, a_4\} = \mathcal{A}(q_i)$ by evenly dividing $A(q_i)$. Those candidate squares which contain sites will become associated with new nodes that are children of q_i, and control will be passed to them, counter-clockwise, in turn, before returning up to the parent of q_i. We fix the maximum depth of Q as $h(Q) = \lceil \log_2(n) \rceil$ so that the edge length of a leaf square is proportional to d/n. We execute a leaf strategy whenever this depth is reached. Also, we execute the same leaf strategy if at any time there is a q that only contains one site.

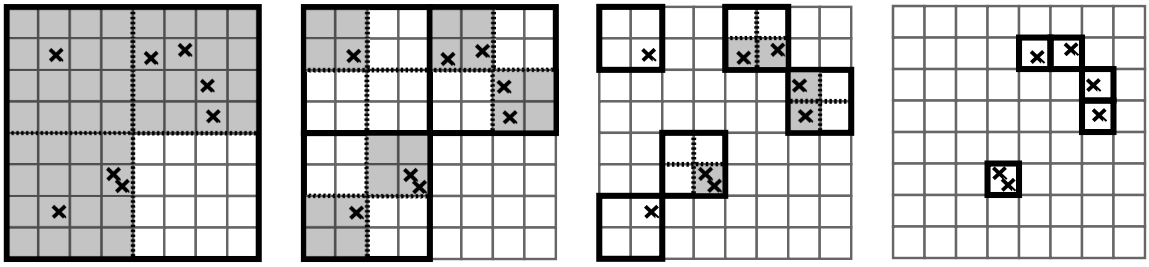

Fig. 2: Here is an example Q constructed by Quad-Explore for an environment that has 8 sites. Q is shown here separated by level but it is constructed in depth first fashion, with children explored in counter-clockwise order from the upper right. At each node it is determined which of the four candidate squares $\mathcal{A}$ contain sites; those that contain sites become squares for children. If the square for a node only contains a single site the node executes the leaf strategy and creates no children. This leaf strategy is also executed whenever the maximum depth $h(Q)$ is reached.

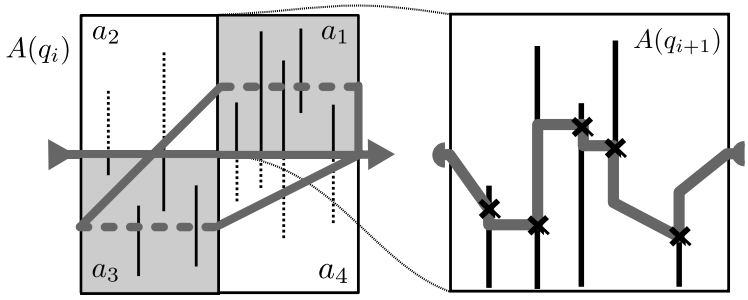

Fig. 3: In the left of the figure is the Quad-Explore strategy for a node q_i that is not a leaf. After performing the cut from left to right it becomes clear that all sites in the node exist in a_1 and a_3, the upper right and lower left candidate subsquares, thus $|\mathcal{C}(q_i)| = 2$. Assuming that the next level is the maximum depth $h(Q)$, each of the two children must perform the leaf strategy, which is demonstrated in the right of of the figure. The leaf strategy for $c_1 \in \mathcal{C}(q_i)$ is performed right to left, and it simply performs X-sweep and visits all of the sites in a_1 in turn.

See Figure 3 for an example of our strategy for a leaf node and a non-leaf node.

The complete description of the actions commanded by a leaf node q_i is the following:

The leaf node actions are functionally almost identical to performing the X-sweep strategy with $A(q_i)$ as the arena. From the leftmost or rightmost border of $A(q_i)$, drive to a point p on the line $x = s_j^x$ for whichever $s_j \in A(q_i)$ has the least x distance from r^x. The y value of p is selected such that $y_j^\ell \leq p^y \leq y_j^u$ and the distance between p^y and r^y is minimized. Then walk along the line segment of s_j until s_j is visited, and then move on to the next closest $s_k^x | s_k \in A(q_i)$. Repeat this process until every site in $A(q_i)$ has been visited. Then pass control back to the parent by visiting the intermediate y of $A(q_i)$ at the opposite x edge from the start of $r \in q_i$.

The complete description of the actions commanded by a non-leaf node q_i is the following:

Perform a y-cut by driving r along the line segment that splits the upper candidate subsquares $\{a_1, a_2\} \in \mathcal{A}(q_i)$ from the lower ones $\{a_3, a_4\} \in \mathcal{A}(q_i)$. Adjust all $\mathcal{Y}^u$ and $\mathcal{Y}^\ell$ that pass through the cut. It is clear at this point which of $\mathcal{A}(q_i)$ contain sites, so associate these squares with children. For child j, q_i is responsible for delivering r to the middle y of a_j, at either the left edge or the right edge of a_j (right edge for a_1 and a_2, left edge for a_3 and a_4). Each child will return control at the opposite side of its square. After the last child returns control, q_i passes control back to its parent by returning r back to the ending point of the y-cut.

Now, consider the case where the arena is not square.

If the arena is longer than it is tall, we simply take any square with edge length d that contains the initial belief line segments as $A(q_0)$ in the algorithm.

If the arena is taller than it is long, we need to introduce a special strategy only used by q_0. The edge lengths of the children of q_0 in this case are all d, so to remain consistent the children are defined as level 0 and the special q_0 is defined as level -1.

The complete description of the actions commanded by an initial q_0 that has a non-square corresponding region is the following:

Perform the first y-cut at $y = 0$ and adjust $\mathcal{Y}$ based on the observations. If afterwards there is any $y_i^u > d, y_i^u \in \mathcal{Y}^u$, perform the next y-cut at $y = d$. All cuts in q_0 will occur at $y = kd$ where k is an integer. The order of cuts is: first increment k until $\max(\mathcal{S}^y)$ is discovered, and then decrement k from 0 until $\min(\mathcal{S}^y)$ is discovered. All subsquares are bounded above and below by instances of $y = kd$. When after any cut it becomes certain there is one or more sites in a subsquare, a child is created for the subsquare and control is passed to the child at the usual extreme x, intermediate y position.

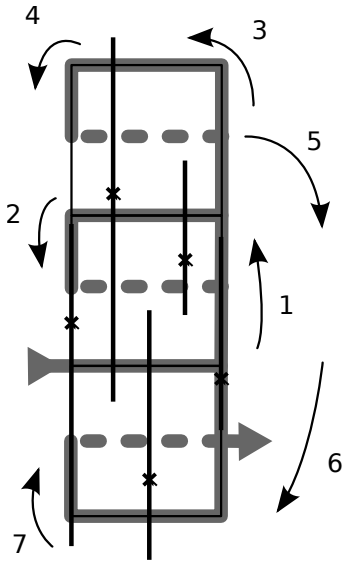

Fig. 4: Here is an example of our level -1 strategy when $\max(\mathcal{S}^y) - \min(\mathcal{S}^y) > d$. The starting position $(0, 0)$ is on the lower left and r follows the thick solid gray path, in the order given by the arrows. When it is certain there is a site in one of the subsquares $a_i \in \mathcal{A}(q_0)$, a child is created for that subsquare and control is passed to the child at the start of the dashed gray line. Control is resumed by q_0 at the other end of the dotted line.

See Figure 4 for the exploration procedure Quad-Explore uses for an example input that is taller than it is long.

B. Analysis

Here we will prove that the worst case competitive ratio C between the length of the robot path with our algorithm $\ell(R)$ and the length of the optimal offline shortest exploration path $\ell(R^*)$ is upper bounded by $k_c \log(n)$ where n is the number of sites and k_c is a constant.

We will prove this by showing that each node in Q does work proportional to its edge length showing that the sum of the edge lengths at each level in Q is linearly upper bounded by $\ell(R^*)$. The proof is completed by observing that the number of levels in Q is logarithmic in n so if each level does proportional work the total work is logarithmic.

First, we introduce some new notation: Denote the set of all paths traveled by r while under control of q_i as $R(q_i)$. The sum of the lengths of these paths is $\ell(R(q_i))$, or, for simplicity, $\ell(q_i)$. Denote the number of sites contained in $A(q_i)$ as $n(q_i)$. This means $n(q_0) = n$ the total number of sites. Denote the depth of q_i as $h(q_i)$, and denote the set of all nodes in Q that are at depth h as $\mathcal{Q}_h$.

Lemma 1: There exists some constant k_1 such that for any square non-leaf node $q_i \in Q$,

$$\ell(q_i) \leq k_1 \frac{d}{2^{h(q_i)}}.$$

Proof: Recall that the first task in node q_i is a cut which is not longer than $\frac{d}{2^{h(q_i)}}$.

Following the cut, q_i needs to deliver r to up to five more target positions (once for each child, and then once for returning to the parent). Hence we can obtain the loose upper bound $k_1 \leq 6\sqrt{2}$ because six targets are traveled to along straight lines and no two points in the square can be farther away than $\sqrt{2}\frac{d}{2^{h(q_i)}}$. ■

Lemma 2: There exists some constant k_2 such that for any leaf node $q_i \in Q$,

$$\ell(q_i) \leq k_2 n(q_i) \frac{d}{2^{h(q_i)}}.$$

Proof: In a leaf node the path of r is monotonic in x, so the sum of all x components of $R(q_i)$ is upper bounded by $\frac{d}{2^{h(q_i)}}$. Recall that r in a leaf node seeks sites in a well-defined order. While seeking any site in $A(q_i)$, the path of r is monotonic in y. The initial y value is intermediate, so the y component while seeking the first site can't cost more than half the edge length of $A(q_i)$. Similarly the final step, the return to intermediate y so that control can be passed back to the parent, also cannot have y distance greater than half the edge length. All other sites can require up to the entire edge length. Therefore we can say the sum of all y components of $R(q_i)$ is upper bounded by $\frac{n(q_i)d}{2^{h(q_i)}}$. Combining the x components and the y components we end up with a worst case

$$\ell(q_i) \leq \frac{(1 + n(q_i))d}{2^{h(q_i)}} \leq \frac{2n(q_i)d}{2^{h(q_i)}}.$$

Therefore $k_2 \leq 2$. ■

Lemma 3: There exists some constant k_3 such that

$$\ell(\mathcal{Q}_h) = \sum_{q_i \in \mathcal{Q}_h} \ell(q_i) \leq k_3 \ell(R^*)$$

Proof: First, consider the case where h is not the maximum depth $h(Q)$ and it is not -1. Denote the set of squares associated with $\mathcal{Q}_h$ as $A(\mathcal{Q}_h)$. $A(\mathcal{Q}_h)$ are a subset from a grid with edge length $\frac{d}{2^h}$, and each of $A(\mathcal{Q}_h)$ contains at least one site. Therefore R^* must enter each of these squares. Now consider the number of squares on the level h grid R^* enters, denoted $|R^*|_h$. If any path is divided into segments of length $\sqrt{2}\frac{d}{2^h}$, each segment can enter at most 7 squares in the level h grid (this case occurs when the segment is diagonal across a grid square). Therefore,

$$|\mathcal{Q}_h| \leq |R^*|_h \leq 7 \left\lceil \frac{\ell(R^*)2^h}{d\sqrt{2}} \right\rceil.$$

Any of $\mathcal{Q}_h$ that are leaf nodes must only contain one site, because the depth is not the max depth so the leaf node strategy is only executed when there is one site. Therefore we can relate the number of nodes with the sum of the paths in all the nodes at depth h as follows, using Lemmas 1 and 2, and defining k' as $\max(k_1, k_2)$:

$$|\mathcal{Q}_h| k' \frac{d}{2^h} \geq \ell(\mathcal{Q}_h).$$

Putting this together with the bound on $|\mathcal{Q}_h|$, and using

the fact that $\ell(R^*) \geq d$, we have:

$$\ell(\mathcal{Q}_h)\frac{2^h}{d} \leq k'|R^*|_h \leq 7k'\left(1+\frac{\ell(R^*)2^h}{d\sqrt{2}}\right)$$
$$\ell(\mathcal{Q}_h) \leq 7k'\left(\frac{d}{2^h}+\frac{\ell(R^*)}{\sqrt{2}}\right)$$
$$\leq 7k'\left(\ell(R^*)+\frac{\ell(R^*)}{\sqrt{2}}\right)$$
$$\leq 7k'\left(1+1/\sqrt{2}\right)\ell(R^*).$$

For $k' \leq 6\sqrt{2}$, this provides us with $k_3 \leq 42(\sqrt{2}+1)$.

Now, consider the case where h is the maximum depth $h(Q)$. Since each site can exist in at most one node at a given level, we can conclude from Lemma 2 that

$$\ell(\mathcal{Q}_h) \leq k_2\frac{d}{2^h}\sum_{q_i\in\mathcal{Q}_h} n(q_i) \leq k_2\frac{d}{2^h}n.$$

Recall that we selected $h(Q) = \lceil \log_2(n) \rceil \geq \log_2(n)$, therefore $\ell(\mathcal{Q}_h) \leq k_2 d \leq k_2\ell(R^*)$.

Finally, consider the case where $h = -1$. The uppermost y-cut will occur at $y_c^u = \max\left(0, d\left\lceil\frac{\max(\mathcal{S}^y)}{d}\right\rceil\right)$ and the lowermost will occur at $y_c^\ell = \min\left(0, d\left\lfloor\frac{\min(\mathcal{S}^y)}{d}\right\rfloor\right)$. The maximum span between the upper and lower cut is therefore:

$$y_c^u - y_c^\ell \leq \max(0, \max(\mathcal{S}^y + d)) - \min(0, \min(\mathcal{S}^y - d))$$
$$\leq 2d + \ell(R^*) \leq 3\ell(R^*)$$

And the maximum number of candidate subsquares for the initial node is:

$$|\mathcal{A}(q_0)| \leq \left\lceil\frac{y_c^u - y_c^\ell}{d}\right\rceil \leq \left\lceil\frac{3\ell(R^*)}{d}\right\rceil \leq \frac{3\ell(R^*)+d}{d} \leq \frac{4\ell(R^*)}{d}$$

Associate each candidate subsquare with the *outer* cut, with greatest absolute value of y. In this analysis, the actions commanded by q_0 in any $a_i, a_i \in \mathcal{A}(q_0)$ are one cut, plus the path walking up to the uppermost cut and walking back down to the lowermost cut, plus deviation of half a square edge to pass control to the child and another half a square edge when control is returned from the child. This is a total distance of not greater than $4d$ commanded by q_0 in any of its candidate subsquares. To this we add another d for the unaffiliated first cut at $y = 0$, and obtain the following:

$$\ell(R(q_0)) \leq d + 4d|\mathcal{A}(q_0)| \leq d + 4d\frac{4\ell(R^*)}{d}$$
$$\leq d + 16\ell(R^*) \leq 17\ell(R^*).$$

This concludes the proof for each of the three types of level. The worst $k_3 \leq 42(\sqrt{2}+1)$ occurs when $0 \leq h < h(Q)$. ■

Theorem 1: There exists some constant k_c such that for any $\mathcal{S}, |\mathcal{S}| \geq 2$, a robot following the Quad-Explore strategy and starting from $r(0) = (0,0)$ will visit every site and the total length of its path $\ell(R)$ will satisfy the following competitive ratio:

$$\frac{\ell(R)}{\ell(R^*)} \leq k_c \log(n).$$

Proof: Here is how we know r will visit every site: The square $A(q_0)$ contains every site. The union of subsquares $\mathcal{A}(q_i)$ contains all of the square $A(q_i)$ and thus every site in $A(q_i)$. If q_i has children, every square in $\mathcal{A}(q_i)$ that contains a site will become the square for a child, therefore the set $\mathcal{C}(q_i)$ will among it have responsibility for every site in $A(q_i)$. If q_i does not have children, it is a leaf node, and the leaf node strategy finds every site in $A(q_i)$.

Here is how we bound the maximum length of ℓR across all possible inputs: Combining the maximum number of levels $1 + h(Q) = 1 + \lceil \log_2(n) \rceil$ with the result of Lemma 3 we obtain:

$$\ell(R) = \sum_h^{1+h(Q)} \ell(\mathcal{Q}_h)$$
$$\leq (1+h(Q))k_3\ell(R*) \leq (1+\lceil\log_2(n)\rceil)k_3\ell(R*)$$
$$\leq (2+\log_2(n))k_3\ell(R*) \leq 3k_3\log_2(n)\ell(R*).$$

This provides an absolute worst case of:

$$k_c \leq 126(\sqrt{2}+1) \leq 305.$$

However we expect that this constant is a very loose estimate which can be improved by carefully considering the different possible cases at each step.

■

V. Simulations

We examined the performance of X-Sweep and Quad-Explore when exploring a real solar map constructed for the McNamara Alumni Center at the University of Minnesota. To do this we used our high density June 9, 2012 solar data set from [4] to construct an approximate heightmap consistent with all of the sun and shade rays measured. Then we constructed our exploration problem by manually fixing 18 positions on the plane where the trees were rooted (and 2 false positives that had height 0), and raytracing from the known position of the sun at 16:23 CDT and the constructed object heights. This procedure gave us $\mathcal{S}$; to obtain the initial y bounds we assumed a prior height distribution between 0 and 60 meters. We simulated a robot starting at the position of the bottom of the northwesternmost tree, and executing both X-Sweep and Quad-Explore. In our Quad-Explore implementation we added a heuristic where the length of each cut could be reduced when there was no information to be gained by travelling farther.

See Figure 5 for a satellite image of the environment and plots of the paths followed by X-Sweep and Quad-Explore. The length of the X-Sweep exploration path was 390.35 meters, compared 644.19 for Quad-Explore. We can lower bound $\ell(R^*)$ with d, which was 39.44; thus our competitive ratio was at most 16.33. X-Sweep performed well in this case because the lines of similar trees ensured that sequential critical points were not too far offset in the y direction.

To examine what happens when the problem size increases, we also generated synthetic environments with sites uniformly distributed in the unit square, and no initial information about y coordinates. See Figure 6 for a chart of the

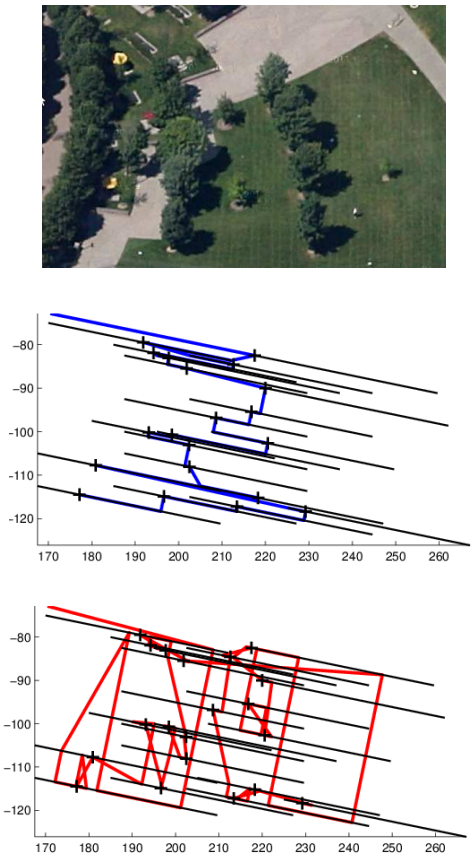

Fig. 5: Top is a satellite image of the McNamara Alumni Center, courtesy of Google Maps. Middle and bottom are the paths followed by X-Sweep and Quad-Explore while exploring the McNamara Alumni Center in simulation. X-Sweep does well in this case because the critical points are close together in the y-direction (for the rotated coordinate frame). Quad-Explore carefully narrows down its belief for the site positions before at the end walking a guaranteed short distance down a particular line segment.

performance of X-Sweep and Quad-Explore as the number of sites increases. We found that Quad-Explore had a slighly higher constant factor but as the problem size grew the linear term in X-Sweep came to dominate over the logarithmic term in Quad-Explore.

Even though the theoretical performance of Quad-Explorer is far superior to X-sweep, these simulations suggests that X-sweep might be a better in some practical scenarios when the number of sites to visit is not large.

VI. Conclusions

In this work we examined competitive strategies to explore an unknown solar map. This exploration can provide the information a solar-aware path planning algorithm needs to plan an energy-minimizing path. We examined first a simple strategy X-sweep which has poor worst case competitive performance of $\mathcal{O}(n)$, and then a more sophisticated strategy Quad-Explore which has superior asymptotic performance

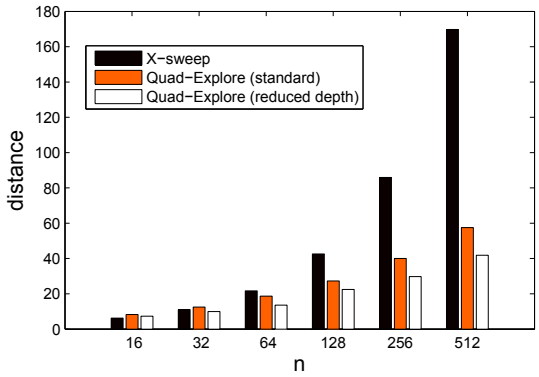

Fig. 6: A comparison of the distance traveled by three exploration algorithms when exploring an increasing number of sites uniformly distributed in the unit square, with no prior y information. Each bar is the average of 10 random trials. In addition to the X-sweep and Quad-Explore presented in this paper, we also tested a variant of the latter with a reduced maximum depth of $\lceil \log_8(n) \rceil$ instead of the usual $\lceil \log_2(n) \rceil$.

of $\mathcal{O}(\log n)$. We simulated both strategies on a real data set and on a series of synthetic environments. We found that X-sweep performs well in practice when the number of sites to visit is not large, however Quad-Explore's asymtotic performance is superior for uniformly distributed sites.

The most direct avenue of further work on this topic is improving the loose bound on the maximum competitive ratio of Quad-Explore. Another future area of interest is examining the robustness of our algorithms to slightly violated assumptions (such as a slightly moving sun, or a tree with a narrow trunk that lets some light through near the ground but not near the top). Finally, the general principles of Quad-Explore are likely relevant for a variety of related but slightly different sensing models, some of which might be useful for solving different real-world exploration problems.

Acknowledgment

This material is based upon work supported by the National Science Foundation under grant number 1111638.

References

[1] S. P. Fekete, R. Klein, and A. Nüchter. Online searching with an autonomous robot. *Computational Geometry*, 34(2):102–115, May 2006.

[2] R. Fleischer, T. Kamphans, R. Klein, E. Langetepe, and G. Trippen. Competitive Online Approximation of the Optimal Search Ratio. *SIAM Journal on Computing*, 38(3):881–898, Jan. 2008.

[3] D. Y. Goswami, F. Kreith, and J. F. Kreider. *Principles of Solar Engineering*. Taylor & Francis, 2nd editio edition, 1999.

[4] P. A. Plonski, P. Tokekar, and V. Isler. Energy-efficient path planning for solar-powered mobile robots. *Journal of Field Robotics*, 30(4):583–601, 2013.

[5] L. Ray, J. Lever, A. Streeter, and A. Price. Design and Power Management of a Solar-Powered Cool Robot for Polar Instrument Networks. *Journal of Field Robotics*, 24(7):581–599, 2007.

[6] C. Sauze and M. Neal. Long term power management in sailing robots. In *OCEANS, 2011 IEEE - Spain*, pages 1 –8, june 2011.

[7] P. Tompkins, A. Stentz, and D. Wettergreen. Mission-level path planning and re-planning for rover exploration. *Robotics and Autonomous Systems*, 54(2):174–183, 2006.