Improved Analysis of D*

Craig Tovey[1] Sam Greenberg[2] Sven Koenig[1]

[1] College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
{ctovey, skoenig}@cc.gatech.edu

[2] School of Mathematics
Georgia Institute of Technology
Atlanta, GA 30332-0160
SamIAm@math.gatech.edu

***Abstract*— D* is a planning method that always routes a robot in initially unknown terrain from its current location to a given goal location along a shortest presumed unblocked path. The robot moves along the path until it discovers new obstacles and then repeats the procedure. D* has been used on a large number of robots. It is therefore important to analyze the resulting travel distance. Previously, there has been only one analysis of D*, and it has two shortcomings. First, to prove the lower bound, it uses a physically unrealistic example graph which has distances that do not correspond to distances on a real map. We show that the lower bound is not smaller for grids, the kind of map-based graph on which D* is usually used. Second, there is a large gap between the upper and lower bounds on the travel distance. We considerably reduce this gap by decreasing the upper bound on arbitrary graphs, including grids. To summarize, we provide new, substantially tighter bounds on the travel distance of D* on grids, thus providing a realistic analysis for the way D* is actually used.**

I. Introduction

Robot navigation problems in unknown terrain have been studied in both theoretical robotics and theoretical computer science. A good overview is given in [9]. However, empirical robotics researchers have often developed their own planning methods. These methods have been demonstrated on mobile robots that solve complex real-world tasks and perform well in practice. Consider, for example, a robot that has to move to a given goal location in unknown terrain. The sensors on-board a robot can typically sense the terrain only near its current location, and thus the robot has to interleave planning with movement to sense new parts of the terrain. This is called sensor-based planning [2]. Dynamic A* [11], [12], [13] (short: D*) is a sensor-based planning method that always moves the robot from its current location on a shortest presumed unblocked path to the goal location. A presumed unblocked path is a path that does not contain known obstacles. D* traverses the path until it discovers new obstacles. It then repeats the procedure, taking into account all the obstacles that it has learned about. If the robot reaches the goal location, it stops and reports success. If at any point it fails to find a presumed unblocked path from its current location to the goal location, it stops and reports that the goal location cannot be reached from the start location. Figure 1 gives an example of the behavior of D*. White locations are unblocked, black locations are blocked, and grey locations have unknown status. The example assumes that the robot senses the status of its four neighboring locations and can then move to an unblocked neighboring location. The circle shows the current location of the robot. The goal location is in the lower right corner.

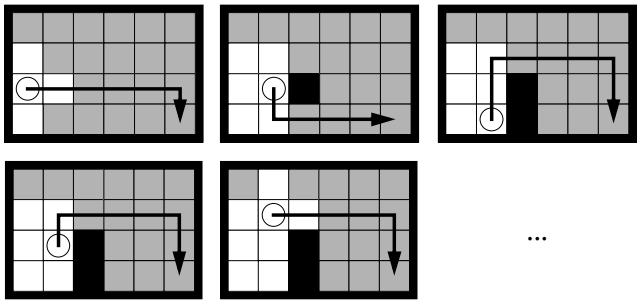

Fig. 1. D* Example

D* has been used outdoors on an autonomous high-mobility multi-wheeled vehicle that navigated 1,410 meters to the goal location in an unknown area of flat terrain with sparse mounds of slag as well as trees, bushes, rocks, and debris [15]. As a result of this demonstration, D* is now widely used in the DARPA Unmanned Ground Vehicle (UGV) program, for example, on the UGV Demo II vehicles. D* is also being integrated into Mars Rover prototypes, tactical mobile robot prototypes and other military robot prototypes for urban reconnaissance [3], [6], [16]. Furthermore, it has been used indoors on Nomad 150 mobile robots in robot-programming classes to reach a goal location in unknown mazes that were built with three-foot high, forty inch long cardboard walls [8], [7]. D* has also been used as the key method of various robot-navigation software [1], [14].

D* is so popular because its repeated replanning can be implemented efficiently [11], [12], [13] and easily [4]. Hence, evaluating the plan-execution times (that is, the travel distances of the robots) is crucial to determining the overall performance of the robots. In practice, the plan-execution times seem to be reasonably small. It is thus

important to understand whether this is due to properties of the test terrains or whether the plan-execution times are indeed guaranteed to be good in any kind of terrain.

So far, there exists one analysis of D* [10]. It considers the worst-case travel distance of D* on arbitrary graphs $G = (V,E)$ and proves a lower bound of $\Omega(\frac{\log|V|}{\log\log|V|}\,|V|)$ steps and an upper bound of $O(|V|^2)$ steps. However, the analysis has two shortcomings. First, while D* can be used on arbitrary graphs (such as Voronoi graphs), it is typically used on grids. However, the example graph used to prove the lower bound was not a grid, and moreover used unrealistic edge lengths not related to physical distances. Thus, the lower bound could potentially be smaller for grids or other physically realizable graphs. We show that this is not the case. Second, there is a large gap between the upper and lower bounds on the travel distance. We reduce this gap by proving an upper bound of $O(|V|^{3/2})$ steps on arbitrary graphs, including grids, which decreases the gap considerably. The proof builds on our analysis of the upper bound on the worst-case travel distance of greedy mapping [5]. We thus provide new, substantially tighter bounds on the worst-case travel distance of D* on grids and a realistic analysis for the way D* is actually used.

II. Assumptions

We study the travel distance of D* analytically. We assume that the robot is omni-directional, point-sized, equipped with only a radial short-distance sensor, and capable of error-free motion and sensing. The sensors on-board the robot uniquely identify its location and the neighboring locations and determine whether the neighboring locations contain obstacles. We model the terrain as graph. Vertices in the graph represent locations in the terrain. Traversing an edge in the graph corresponds to traveling from one location to an adjacent location in the terrain. We are interested in the quality of the plans determined by D* as a function of the number of vertices of the graph. We use the worst-case travel distance of the robot (in edge traversals or, synonymously, steps) to measure the plan quality because a small worst-case travel distance guarantees that the robot performs well in all terrains.

With these assumptions, we can formalize the behavior of D* as follows. We call a graph $G = (V,E)$ vertex-blocked by $B \subset V$ if B is the set of blocked vertices (that is, vertices that cannot be visited). On a finite (undirected) graph $G = (V,E)$ vertex-blocked by B, a robot has to reach a designated goal vertex. D* always moves the robot from its current vertex along a shortest presumed unblocked path to the goal vertex. A presumed unblocked path is one that contains no vertices which are known to be blocked. D* terminates when the robot reaches the goal vertex or there are no presumed unblocked paths to the goal vertex,

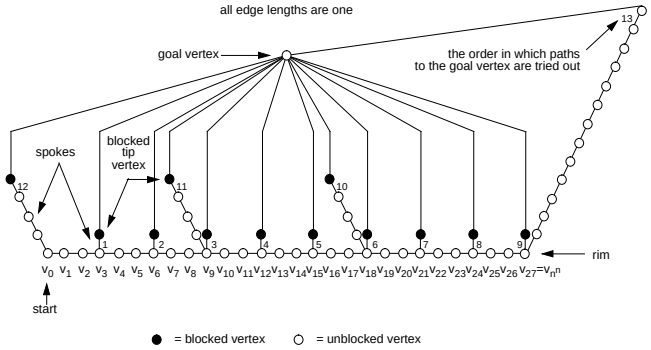

Fig. 2. Previous Example Graph for Lower Bound

in which case the goal vertex is unreachable from the start vertex since the graph is undirected. D* must terminate since each path it tries either reaches the goal or augments the set of known blocked vertices.

III. D*: Lower Bound on Grids

We now prove a lower bound on the worst-case travel distance of D* on vertex-blocked grids. We first review the previous analysis, which employs the key idea of tricking the robot into traversing the same long path back and forth many times. Second, we give a visual and conceptual overview of how to transform that example into a grid without losing the key idea. Third, we explain exactly how the grid is constructed. Finally, we analyze the worst-case travel distance of D* on our grid graph, proving the same lower bound.

A. Previous Result

The previous analysis proved that the worst-case travel distance of D* is $\Omega(\frac{\log|V|}{\log\log|V|}\,|V|)$ steps on vertex-blocked graphs $G = (V,E)$ [10]. This lower bound is achieved with graphs of the structure shown in Figure 2. We now sketch the main idea of its construction, but with our own rim-and-spoke terminology, in order to introduce our much more complex grid construction.

The graph of Figure 2 consists of a long horizontal path of length d^d (where d is a integer parameter), which we call the "rim", and a set of "spokes" of varying lengths attached to the rim at various vertices. The uppermost "tip" vertex of each spoke is blocked and connected to the goal vertex by an edge. Note that the edges from the tips to the goal are physically unrealistic edges, because in the graph the robot can move to the goal from any tip in one step. The possible spoke lengths are $\sum_{i=0}^{h} d^i$ for the nonnegative integers h. We refer to a spoke of length $\sum_{i=0}^{h} d^i$ as a "class h spoke". Longer spokes are spaced farther apart from each other than short spokes. In particular, the vertices where class h spokes attach to the rim have distance d^{h+1} from each other. Hence, if the robot is at a vertex where a class h spoke attaches to the rim, then it is shorter to go

to the goal along the rim to the next class h spoke, than it is to go via any class $h+1$ spoke.

The robot starts at vertex v_0. Initially, it traverses the rim from left to right, checking the class 0 spokes for a path to the goal vertex; then it returns along the rim from right to left, checking class 1 spokes for a path to the goal vertex, and so on. Each class forces the robot to traverse the rim once. If there are d classes, then the robot traverses the rim d times for a total travel distance of d^{d+1}. Since a computation shows that there are $|V| = O(d^d)$ vertices in the graph, the total travel distance is $\Omega(\frac{\log|V|}{\log\log|V|}|V|)$.

B. Conceptual Overview

The lower bound from the previous analysis could potentially be smaller on more realistic graphs such as the grid graphs on which D* is often used. We now prove that this is not the case with a much more complex construction of a grid that uses the key idea from the previous analysis, to fool the robot into traversing the lengthy rim many times. The robot moves back and forth, each time checking spokes of the next class for a path to the goal vertex because the end of each spoke is connected to the goal vertex. Each time, the robot finds that the tip vertex of the spoke is blocked and thus needs to traverse a different spoke. The spokes are spaced at particular distances from each other on the rim to trick the robot into visiting all the class h spokes before visiting any class $h+1$ spokes. However, the graph topology of the previous analysis cannot directly be embedded into a grid because the goal vertex must be simultaneously adjacent to the ends of many spokes of greatly different lengths, which moreover are placed at great distances from each other. In a grid, on the other hand, each cell is adjacent to at most four other cells, and the distance between two adjacent cells is always one. We use several ideas to modify the graph topology of the previous construction to be able to embed it into a grid. A rough conceptual sketch of these ideas is shown in Figure 3.

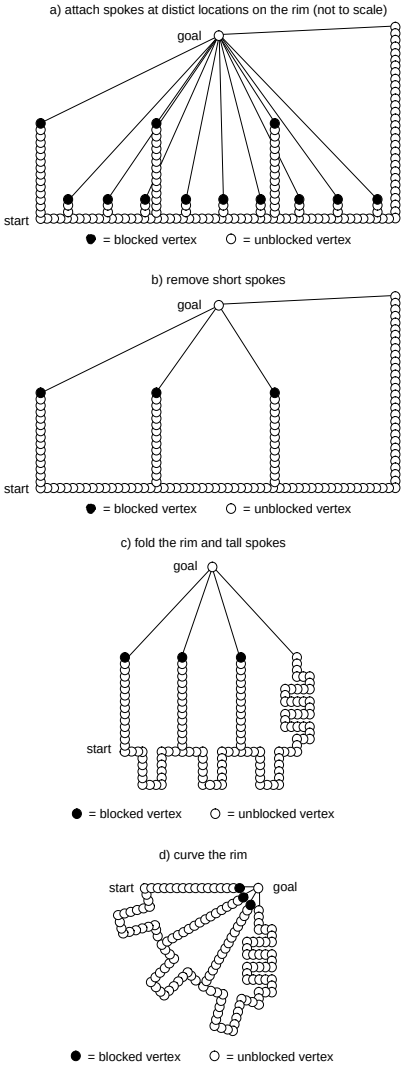

Fig. 3. Steps of the Transformation

1) Attach each spoke at a separate vertex to the rim (Figure 3a).This eliminates the problem of a vertex on the rim being adjacent to too many other vertices. As long as longer class spokes are spaced far enough apart, the robot is still fooled into repeatedly traversing the rim.
2) Remove the very short spokes (Figure 3b). We have to place the goal vertex at some distance D from the rim, and we thus cannot construct spokes of length less than D. This is one of the reasons to use classes $0.8d$ to $0.9d$ instead of using classes 0 to d. As long as there are $\Omega(d)$ classes, the lower bound is the same order.
3) Move the spokes physically closer together, but maintain their distances from each other along the rim. We do this by "squeezing" the rim into an accordion shape (Figure 3c). In particular, the sections of the rim between spokes get bent into long loops, which we call "columns."
4) Redesign the spokes so that they all have the same physical height, while maintaining their original lengths (Figure 3c). In particular, build a pair of blocked walls of the same height, with some space between them. Put a twisty path of the appropriate length in between the walls.
5) Once the spokes are fairly close and of equal height, bend the rim into part of a circular arc, pushing the tips of the spokes together towards the goal vertex (Figure 3d). It is not possible to squeeze too many distinct vertices into a small area on a grid, but this problem is solved by blocking the paths to the goal vertex a bit before the goal vertex.

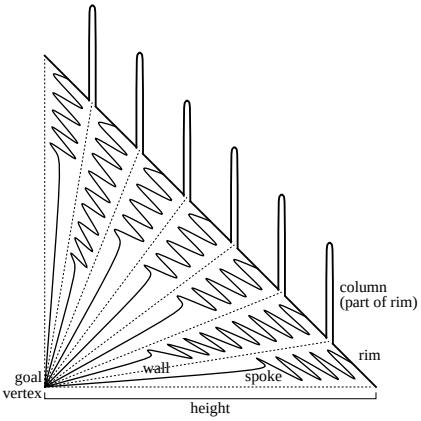

Fig. 4. Conceptual Figure with Two Spoke Classes

C. Structural Overview

Figure 4 illustrates the structure of the actual construction. Place the goal vertex g at $(0,0)$. Place the rim on a diagonal from $(0,z^{0.7z})$ to about $(z^{0.7z},0)$, for some sufficiently large integer z. Note that the rim is two concentric quarter circles in the "taxicab" metric $\mathscr{L}_1$, so each point is within 1 of $z^{0.7z}$ from g. Along the rim, there will be "spoke-base points" and "column-base points", alternating. (Note: To avoid notational clutter in this paragraph and in the foregoing, we omit the "floor" operation notation. Here, for example, $z^{0.7z}$ really means $\lfloor z^{0.7z} \rfloor$.)

From each spoke-base, construct a windy path towards g. Each path has length $2z^j$ for some $j \in \{0.8z, 0.8z+1, \ldots, 0.9z\}$. We call such a path of length $2z^j$ a "spoke of class j". The graph will contain z^{z-j-1} spokes of class j, for each j.

At each column-base point, replace the point with a construction to add distance, increasing the steps needed to pass that point on the rim. This ensures that the distance between successive spokes of class j is z^{j+1}, as well as setting the total rim length to z^z. Put a blocked cell near the end of each spoke except one of class $0.9z$. The robot will be tempted by each of the spokes of class $0.8z$ in turn, following the rim for about z^z steps. The robot then will turn around and be tempted only by the spokes of the next class $0.8z+1$, again traversing the rim for approximately z^z steps, and so on.

D. Construction

Define the outer rim R such that

$$R := \left\{(x,y) \in \mathbb{Z}_{\geq 0} \times \mathbb{Z}_{\geq 0} : z^{0.7z} \leq x+y \leq z^{0.7z}+1\right\}.$$

For each $i \in \{0.8z, \ldots, 0.9z\}$, we will create z^{z-i-1} spokes of class i. Let $S := \frac{z^{0.2z}-z^{0.1z-1}}{z-1}$ be the number of total spokes. For $i \in \{1, \ldots, S\}$, define the i_{th} spoke-base, b_i, such that

$$b_i := \left(\frac{z^{0.7z}}{S}i + \frac{z^{0.7z}}{2S}, z^{0.7z} - \frac{z^{0.7z}}{S}i - \frac{z^{0.7z}}{2S}\right).$$

Therefore $b_i \in R$.

The taxicab distance between two adjacent spoke bases will be $2z^{0.7z}/S$, but to make the construction's key idea work, these distances must be longer for the robot as it moves in the graph. We increase the graph distances by inserting detour loops into the rim. Lemma 1 makes this idea precise.

Lemma 1: Given a function $t : \{1, \ldots, S\} \to \mathbb{Z}_{\geq 0}$ such that $t(i) - t(i-1) \geq \frac{2z^{0.7z}}{S+1} \forall i$, it is possible to modify R using only cells above R in the plane, such that $\forall j > i$, traveling along R from b_i to b_j takes between $(t(j) - t(i) - 4)$ and $(t(j) - t(i) + 4)$ steps.

Proof: For $i \in \{0, \ldots, S\}$, define the i_{th} column-base, c_i such that

$$c_i := \left(\frac{z^{0.7z}}{S}i, z^{0.7z} - \frac{z^{0.7z}}{S}i\right).$$

Therefore $c_i \in R$. At each of the column-bases, we remove the point itself and the point above it from the rim and add two paths traveling upwards, connected at the top. So, if the column-base is at (x,y), remove (x,y) and $(x,y+1)$ from R and add one path from $(x-1,y+2)$ to $(x-1,y+3+h)$ and another path from $(x+1,y)$ to $(x+1,y+3+h)$, with a connecting point at $(x,y+3+h)$. This increases the steps needed to cross this point in the rim by $2h$. We call such a construction a "column of height h". A column of height 3 is illustrated in Figure 5.

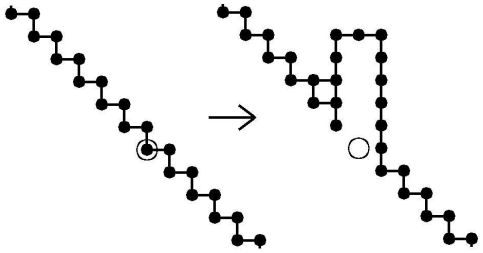

Fig. 5. A column of height 3.

We now define an iterative algorithm for building the columns. For a fixed i, assume the previous columns have been built and let D be the current distance from b_1 to b_i. Let $h := \lfloor \frac{t(i)-t(1)-D}{2} \rfloor$ and build a column of height h at c_{i-1}. This ensures that the distance from b_1 to b_i is now $t(i) - t(1)$, up to the round-off error. Repeat the process for all later i. (Note that the recalculation from $t(1)$ here prevents accumulation of round-off error.) ■

The fourth idea was to build each spoke as a twisty path of the appropriate length. Each spoke will consist of a wedge of sufficient area. The spokes will not overlap, except in a small unblocked triangular region near the goal vertex, within which all paths are direct.

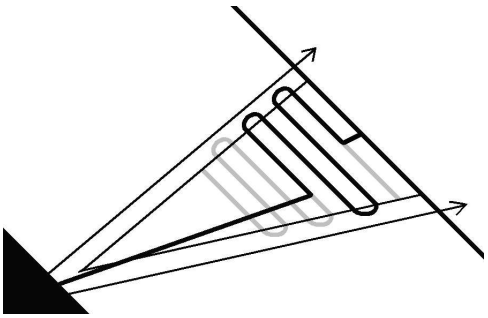

Fig. 6. The hypothetical path H_i, along with the shortening necessary to set the path length to $l(i)$.

Lemma 2: Given a function $l : \{1,\ldots,S\} \rightarrow \mathbb{Z}_{\geq 0}$ such that $z^{0.7z} \leq l(i) \leq z^{z-1}$, $\forall i$, it is possible to construct spokes from b_i such that the distance from b_i to t is between $(l(i)-4)$ and $(l(i)+4)$.

Proof: We would like to connect each of the spokes to g. However, the max degree of a grid prevents this. Therefore we add a triangle T such that

$$T := \left\{(x,y) : x,y \in \mathbb{Z}_{\geq 0} \bigwedge x+y \leq z^{0.5z}\right\}.$$

We may then simply connect the spokes to T. For each $i \in \{1,\ldots,S\}$, define the i_{th} "tip" vertex t_i such that

$$t_i := \left(\frac{z^{0.5z}}{S}i + \frac{z^{0.5z}}{2S}, z^{0.5z} - \frac{z^{0.5z}}{S}i - \frac{z^{0.5z}}{2S}.\right).$$

Therefore $t_i \in T$. This will be the point that the i_{th} spoke connects to.

To construct the paths of length $l(i)$ for each i, construct a hypothetical path H_i from b_i of excessive length. Then, when building the actual graph, simply include as much of H_i as necessary before the graph takes a direct path to t_i. The point where the graph ignores H_i and instead switches to a direct path to t_i depends on $l(i)$. Block the cell on the path just before it reaches t_i.

Building H_i takes a bit of construction. Use Euclidean rays from g to partition the space between R and g into areas A_i and then create many path segments running parallel to R called "levels". H_i runs up and down these levels, traveling back and forth to increase length. Define a space C_i to give room to connect one level to the next without coming close to the rays. This is all illustrated in Figure 6. The triangle in the the lower left corner represents a region of unblocked cells, bordered by the t_i. Since all of the twisty paths have become direct paths by the time they reach their t_i, and the blockages occur prior to reaching t_i, it is OK for the spokes to overlap within this unblocked region.

For each $i \in \{0,\ldots,S\}$, define the i_{th} "ray" r_i to be the Euclidean line from $\left(\frac{z^{0.5z}}{S}i, z^{0.5z} - \frac{z^{0.5z}}{S}i\right)$ to c_i. Hence r_i goes from T to R. For each $i \in \{1,\ldots,S\}$, define the i_{th} area A_i to be the integer points between r_{i-1} and r_i. Define the i_{th} cushion C_i such that

$$C_i := \left\{(x,y) : x,y \in \mathbb{Z}_{\geq 0} \bigwedge d[(x,y),r_i] \leq 8\right\}.$$

For each $i \in \{1,\ldots,z^{0.3z}-2\}$ and each $j \in \{1\ldots,0.1z^{0.7z}\}$, define the level $l_{i,j}$ such that

$$\begin{aligned} l_{i,j} &:= \left\{(x,y) : z^{0.7z} - 6j \leq x+y \leq z^{0.7z} - 6j + 1\right\} \\ &\bigcap \quad (A_i - C_i - C_{i+1}). \end{aligned}$$

Use levels $\{l_{i,0},\ldots,l_{i,0.1z^{0.7z}}\}$ to make H_i, using C_i and C_{i+1} to connect the levels. C_i is large enough to let H_i avoid crossing the ray.

The distance between c_i and c_{i+1} is $\frac{2z^{0.7z}}{S} \geq 2z^{0.3z}$. The levels are parallel and contained in a Euclidean triangle. The smallest is only one tenth of the way to the point, so each level is longer than $z^{0.3z}$. There are $0.1z^{0.7z}$ levels, so the total length of H_i is at least $0.1z^z$.

To build the actual spoke, define a function $s(p)$ for all points $p \in H_i$, such that $s(p)$ is the distance from b_i to t if we were to shorten H_i at p and take a direct path to t_i from p. (Note that this definition involves the actual distance in the graph, avoiding accumulated round-off error.) Let S_i be the point in H_i that minimizes $|s(S_i) - l(i)|$. For any two points p, p' in the same level, if $d[p,p'] = 2$, then $|s(p) - s(p')| = 2$, since $d[p,t] = d[p',t]$. Therefore, if S_i is contained in one of the levels, shortening H_i at S_i gives a spoke within 2 of $l(i)$. If S_i is contained in one of the cushions, we may be able to create an even more precise spoke. For any adjacent $c, c' \in C_i \cap H_i$, $|s(c) - s(c')| \leq 1$, as the path from c to t_i likely passes through c'. Hence, regardless of whether S_i appears in a level or in a cushion, we exceed the precision required by the Lemma. ■

To finally build our graph, define a position function $p[i,j]$ such that

$$p[i,j] := z^{i+1}j + \frac{z^{0.8z+1}}{0.1z+1}(i - 0.8z).$$

This will be the "position" of the j_{th} spoke of class i. Order the spokes by this position function, so the first spoke is the one with lowest position, the second is the one with second lowest position, and so forth.

Define the length function l such that $l[k] := 2z^i$, where i is the class of the k_{th} spoke. Use this l with Lemma 1.

Define t such that $t[k] = p[i,j]$, where i and j are the coordinates for the k_{th} spoke. Use this t with Lemma 2.

E. Analysis

We are now ready to prove the following theorem.

Theorem 1: The worst-case travel distance of D* on vertex-blocked grids $G = (V,E)$ is $\Omega(\frac{\log|V|}{\log\log|V|}|V|)$ steps.

Proof: The distance the robot must travel to find a spoke of class i and then travel to g is at most $z^{i+1} + 2z^i$ steps.

For any $j > i$, simply traveling a spoke of class j will take at least $2z^j \geq 2z^{i+1}$ steps. Hence the robot will walk to the smallest class spoke available, find a blocked cell, and go to the next of that class, traversing the rim.

By the placement of the spokes, notice that $\forall h, h' \in \{0.8z, \ldots, 0.9z\}$, if $h < h'$, then the rightmost h-class spoke is to the right of the rightmost h'-class spoke. Also the leftmost h-class spoke is to the left of the leftmost h'-class spoke. Hence, after visiting every spoke of class i, the robot turns around and finds the first spoke of class $i+1$. Each time the robot traverses the rim, it goes from the leftmost spoke of class i to the rightmost spoke of class i. This distance is more than $z^z - 2z^{i-1} = \Omega(z^z)$. The total travel distance (just on the rim) is therefore at least $0.1z\Omega(z^z) = \Omega(z^{z+1})$.

On the other hand, there are $\theta(z^z)$ vertices in the rim (including the columns). There are $[O(z^{0.5z})]^2 = O(z^z)$ vertices in T. In class i there are z^{z-i-1} spokes, each with $O(z^i)$ vertices, so each class contains $O(z^{z-1})$ vertices. There are $0.1z$ classes, so there are $O(z^z)$ vertices in the spokes. Therefore n, the total number of vertices in the graph, is $\theta(z^z)$. If $n = \theta(z^z)$, then the total distance is $\Omega(z^{z+1}) = \Omega(nz) = \Omega(\frac{n\log(n)}{\log\log n})$. ■

IV. D*: Upper Bound

We now prove an upper bound on the worst-case travel distance of D* on arbitrary vertex-blocked graphs, including grids. We first review the previous analysis that proved an upper bound of $O(|V|^2)$ steps on arbitrary vertex-blocked graphs $G = (V,E)$. We then explain our analysis, that builds on ideas of both the previous analysis of D* and our previous analysis of greedy mapping [5] but proves an upper bound of $O(|V|^{3/2})$ steps. Notice that the worst-case travel distance of any planning method that moves a robot from its current vertex to the goal vertex is $\Omega(|V|)$ because the robot has to visit every vertex in the worst case. Chronological backtracking is a planning method that achieves this bound. Thus, the best upper bound on the worst-case travel distance of any planning method is $O(|V|)$. Our upper bound of $O(|V|^{3/2})$ thus shows that D* is not very badly suboptimal.

A. Previous Result

The previous analysis showed that the worst-case travel distance of D* is $O(|V|^2)$ steps on arbitrary vertex-blocked graphs $G = (V,E)$ [10]. We now sketch the main idea of its proof in order to introduce our more complex proof of a smaller upper bound. We use the term *blockage* to mean a blocked vertex. A *blockage detection* occurs when the robot observes a blockage it did not know about before. When a blockage detection occurs, the robot detects all blockages adjacent to its current vertex. The robot always follows a shortest presumed unblocked path to the goal vertex until it either detects a blockage or stops. Each such path has a length of at most $|V|-1$ steps. A blockage detection can never occur twice at the same vertex, so there can be no more than $|V|$ blockage detections. Thus, the total travel distance can be no longer than $|V|$ paths of length $|V|-1$, each preceding a blockage detection, plus one more path of length $|V|$ leading to the goal vertex. Thus, the worst-case travel distance of D* on vertex-blocked graphs is at most $|V|^2$ steps.

B. New Analysis

The quadratic upper bound from the previous analysis leaves a large gap with the lower bound of the previous section. We now narrow the gap by proving that the worst-case travel distance of D* is $O(|V|^{3/2})$ steps. The proof of this tighter upper bound uses the same ideas of blockage detections and of conceptually separating the route of the robot into paths between successive blockage detections that we just used to justify the quadratic upper bound.

Theorem 2: The worst-case travel distance of D* on vertex-blocked graphs $G = (V,E)$ is (at most) $O(|V|^{3/2})$ steps.

Proof: We call a blockage detection at which the robot discovers that it cannot reach the goal vertex a *terminating* blockage detection. All other blockage detections are *nonterminating*. Let $b \leq |V|$ denote the number of nonterminating blockage detections that occur. Let c^i ($0 \leq i \leq b$) denote the vertex at which the robot is located during the ith blockage detection. (The zeroth blockage detection is considered to occur at the start vertex, just prior to the first edge traversal.) Let L_i ($1 \leq i \leq b$) denote the number of steps between the $(i-1)$st and the ith blockage detection.

The robot cannot take more than $|V|$ steps after the bth blockage detection, because it travels on a shortest presumed unblocked path from c^b to the goal vertex and stops either at the goal vertex or earlier at the terminating blockage detection. Then, the total number of steps taken by the robot is at most

$$|V| + \sum_{i=1}^{b} L_i. \tag{1}$$

For every vertex $v \in V$, let d_v^i ($0 \leq i \leq b$) denote the length of a shortest presumed unblocked path from vertex v to the goal vertex directly after the ith blockage detection. That is, d_v^i is the *presumed distance* from v to the goal vertex, just after the ith blockage detection occurs. We define the value d_v^i as unchanged from the value d_v^{i-1} if D* detects that it cannot reach the goal vertex from vertex v. If the presumed distance from a vertex v to the goal vertex is infinite right from the start, we set $d_v^0 = 0$. Note that $0 \leq d_v^i \leq |V|-1$ and that d_v^i is nondecreasing in i. We define $\phi^i = \sum_{v \in V} d_v^i \geq 0$. Note that ϕ^i is nondecreasing in i and $\phi^b \leq |V|^2$.

The main intuition behind the proof is that the robot "thinks" it is getting closer to the goal vertex with each step it takes along a presumed unblocked path, but gets "disappointed" by a blockage detection which increases the presumed distances d_v to the goal. Intuitively, if there are many large L_i the robot should get many big disappointments, which should greatly increase ϕ, and there is a limit on how large ϕ can get. This should force $\sum_i L_i$ and hence the total travel distance to be small. This intuition is similar to that in a bound proved for greedy mapping in [5]. However, there is a direct relationship between an analogous ϕ function and L_i for greedy mapping, but there is no such direct relationship for D*. Our proof is therefore considerably more complicated. We use a proxy Δ^i instead of counting the number of steps L_i directly.

We define $\Delta^i = d^i_{c^i} - d^{i-1}_{c^i} \geq 0$ $(1 \leq i \leq b)$, that is, Δ^i is the amount by which the presumed distance from the robot to the goal vertex increases when the robot is at vertex c^i and detects the ith blockage.

Let D denote the presumed distance from the current vertex of the robot to the goal vertex (the present time will be clear from the context). Consider the motion of the robot from the $(i-1)$st to the ith blockage detection. The robot makes L_i steps, and D decreases by L_i during those steps. Then, when the robot is at vertex c^i, D increases by Δ^i. Therefore, D increases by a total of $\sum_{i=1}^{b}(\Delta^i - L_i)$ during the execution of D* from the start to directly after the bth blockage detection. On the other hand, $D = d^0_{c^0}$ at the start and $D = d^b_{c^b}$ directly after the bth blockage detection. Therefore, D increases by a total of $d^b_{c^b} - d^0_{c^0}$ during the execution of D* from the start to directly after the bth blockage detection. Thus, it holds that $d^b_{c^b} - d^0_{c^0} = \sum_{i=1}^{b}(\Delta^i - L_i)$. Since $-|V| \leq d^b_{c^b} - d^0_{c^0} \leq |V|$, it holds that $-|V| \leq \sum_{i=1}^{b}(\Delta^i - L_i) \leq |V|$ and $\sum_{i=1}^{b} L_i \leq |V| + \sum_{i=1}^{b} \Delta^i$. From the bound (1), it then follows that the total number of steps taken by the robot is at most

$$|V| + \sum_{i=1}^{b} L_i \leq 2|V| + \sum_{i=1}^{b} \Delta^i. \tag{2}$$

This bound is crucial since it relates Δ^i to the actual quantity of interest.

Let x_z denote any vertex at a presumed distance of z edges or less from c^i $(1 \leq i \leq b)$, where the distance is the length of a shortest presumed unblocked path directly after the ith blockage detection. Then $d^i_{c^i} \leq z + d^i_{x_z}$ since, according to the triangle inequality, the presumed distance directly after the ith blockage detection from c^i to the goal vertex cannot be larger than the presumed distance from c^i via x_z to the goal vertex. Since blockage detections can only increase the presumed distances, x_z must also have been at a presumed distance of z edges or less from c^i directly after the $(i-1)$st blockage detection. Also, since the graph is undirected, the presumed distance from c^i to x_z equals the presumed distance from x_z to c^i. Then $d^{i-1}_{x_z} \leq z + d^{i-1}_{c^i}$ since, according to the triangle inequality, the presumed distance directly after the $(i-1)$st blockage detection from x_z to the goal vertex cannot be larger than the presumed distance from x_z via c^i to the goal vertex. Putting the two inequalities together, we get $d^i_{x_z} - d^{i-1}_{x_z} \geq (d^i_{c^i} - z) - (d^{i-1}_{c^i} + z) = (d^i_{c^i} - d^{i-1}_{c^i}) - 2z = \Delta^i - 2z$. Since the presumed distances $d^i_{x_z}$ are nondecreasing in i, it also holds that $d^i_{x_z} - d^{i-1}_{x_z} \geq 0$ and thus $d^i_{x_z} - d^{i-1}_{x_z} \geq \max(\Delta^i - 2z, 0)$.

For all $0 \leq z \leq d^i_{c^i}$ $(1 \leq i \leq b)$, there must be at least $z+1$ vertices at a presumed distance of z edges or less from vertex c^i directly after the ith blockage detection, since there exists a presumed unblocked path from vertex c^i to the goal vertex. Let x_z $(0 \leq z \leq d^i_{c^i})$ denote a vertex at a presumed distance of z edges or less from vertex c^i directly after the ith blockage detection, with $x_z \neq x_{z'}$ for $z \neq z'$. Note that $x_0 = c^i$. Since $\Delta^i = d^i_{c^i} - d^{i-1}_{c^i} \leq d^i_{c^i}$, it holds that

$$\begin{aligned}
\phi^i - \phi^{i-1} &\geq \sum_{z=0}^{d^i_{c^i}} (d^i_{x_z} - d^{i-1}_{x_z}) \\
&= \Delta^i + \sum_{z=1}^{d^i_{c^i}} (d^i_{x_z} - d^{i-1}_{x_z}) \\
&\geq \Delta^i + \sum_{z=1}^{d^i_{c^i}} \max(\Delta^i - 2z, 0) \\
&\geq \Delta^i + \sum_{z=1}^{\Delta^i} \max(\Delta^i - 2z, 0) \\
&\geq \frac{(\Delta^i)^2}{4}.
\end{aligned}$$

Summing over i results in

$$\sum_{i=1}^{b} \frac{(\Delta^i)^2}{4} \leq \phi^b - \phi^0 \leq |V|^2.$$

We can now bound $\sum_{i=1}^{b} \Delta^i$. The values Δ^i $(1 \leq i \leq b)$ are constrained by $\Delta^i \geq 0$ and $\sum_{i=1}^{b} (\Delta^i)^2 \leq 4|V|^2$. Calculus shows that maximizing $\sum_{i=1}^{b} \Delta^i$ subject to these constraints is achieved by $\Delta^i = 2|V|/\sqrt{b}$ for all $1 \leq i \leq b$. Thus, $\sum_{i=1}^{b} \Delta^i \leq 2b|V|/\sqrt{b} \leq 2|V|^{\frac{3}{2}}$ because $b \leq |V|$. Finally, the upper bound from Formula (2) implies that the total number of steps taken by the robot is at most $2|V| + 2|V|^{\frac{3}{2}}$.

■

V. Conclusion

In this paper, we analyzed the worst-case travel distance of D*, a planning method that always moves a robot in initially unknown terrain from its current vertex on a shortest presumed unblocked path to a given goal vertex.

We improved on the only existing analysis by proving a lower bound of $\Omega(\frac{\log|V|}{\log\log|V|}|V|)$ steps on vertex-blocked grids and an upper bound of $O(|V|^{3/2})$ steps on arbitrary vertex-blocked graphs, including grids. This provides new, substantially tighter bounds on the worst-case travel distance of D* on grids and thus a realistic analysis for the way D* is actually used.

ACKNOWLEDGMENTS

This research is partly supported by NSF awards under contracts IIS-9984827, IIS-0098807, and ITR/AP-0113881. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations and agencies or the U.S. government.

VI. REFERENCES

[1] B. Brumitt and A. Stentz. GRAMMPS: a generalized mission planner for multiple mobile robots. In *Proceedings of the International Conference on Robotics and Automation*, 1998.

[2] H. Choset and J. Burdick. Sensor based planning and nonsmooth analysis. In *Proceedings of the International Conference on Robotics and Automation*, pages 3034–3041, 1994.

[3] M. Hebert, R. McLachlan, and P. Chang. Experiments with driving modes for urban robots. In *Proceedings of the SPIE Mobile Robots*, 1999.

[4] S. Koenig and M. Likhachev. Improved fast replanning for robot navigation in unknown terrain. In *Proceedings of the International Conference on Robotics and Automation*, 2002.

[5] S. Koenig, C. Tovey, and W. Halliburton. Greedy mapping of terrain. In *Proceedings of the International Conference on Robotics and Automation*, pages 3594–3599, 2001.

[6] L. Matthies, Y. Xiong, R. Hogg, D. Zhu, A. Rankin, B. Kennedy, M. Hebert, R. Maclachlan, C. Won, T. Frost, G. Sukhatme, M. McHenry, and S. Goldberg. A portable, autonomous, urban reconnaissance robot. In *Proceedings of the International Conference on Intelligent Autonomous Systems*, 2000.

[7] I. Nourbakhsh. *Interleaving Planning and Execution for Autonomous Robots*. Kluwer Academic Publishers, 1997.

[8] I. Nourbakhsh and M. Genesereth. Assumptive planning and execution: a simple, working robot architecture. *Autonomous Robots Journal*, 3(1):49–67, 1996.

[9] N. Rao, S. Hareti, W. Shi, and S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM–12410, Oak Ridge National Laboratory, Oak Ridge (Tennessee), 1993.

[10] Y. Smirnov. *Hybrid Algorithms for On-Line Search and Combinatorial Optimization Problems*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh (Pennsylvania), 1997. Available as Technical Report CMU-CS-97-171.

[11] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the International Conference on Robotics and Automation*, pages 3310–3317, 1994.

[12] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.

[13] A. Stentz. Optimal and efficient path planning for unknown and dynamic environments. *International Journal of Robotics and Automation*, 10(3):89–100, 1995.

[14] A. Stentz. CD*: A real-time resolution optimal re-planner for globally constrained problems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 605–612, 2002.

[15] A. Stentz and M. Hebert. A complete navigation system for goal acquisition in unknown environments. *Autonomous Robots*, 2(2):127–145, 1995.

[16] S. Thayer, B. Digney, M. Diaz, A. Stentz, B. Nabbe, and M. Hebert. Distributed robotic mapping of extreme environments. In *Proceedings of the SPIE: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, volume 4195, 2000.