Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 Keller Hall
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 15-005

Navigation Around an Unknown Obstacle for Autonomous Surface Vehicles Using a Forward-Facing Sonar

Patrick A. Plonski, Joshua Vander Hook, Cheng Peng, Narges Noori, Volkan Isler

April 16, 2015

Noname manuscript No.
(will be inserted by the editor)

Navigation Around an Unknown Obstacle for Autonomous Surface Vehicles Using a Forward-Facing Sonar

Patrick A. Plonski, Joshua Vander Hook, Cheng Peng, Narges Noori, Volkan Isler

Received: date / Accepted: date

Abstract A robotic boat is moving between two points when it encounters an obstacle of unknown size. The boat must find a short path around the obstacle to resume its original course. How should the boat move when it can only sense the proximity of the obstacle, and does not have prior information about the obstacle's size? We study this problem for a robotic boat with a forward-facing sonar.

We study two versions of the problem. First, we solve a simplified case when the obstacle is a rectangle of known orientation but unknown dimensions. Second, we study a more general case where an arbitrarily shaped obstacle is contained between two known parallel lines. We study the performance of the algorithms analytically using competitive analysis and present results from field experiments. The experimental setup is relevant for harbor patrol or autonomous navigation in shallow water.

1 Introduction

Imagine an autonomous robot equipped with a forward facing sensor. The robot detects an obstacle in front of it. How can the robot go around the obstacle as quickly as possible? This is a classical robot navigation problem.

Now imagine the robot is a boat, equipped with a forward facing sonar (Figure 1). The narrow width of the sonar, the noisy and sometimes ambiguous sonar readings, and the motion constraints of the boat make the problem challenging in this domain.

Fig. 1 The autonomous boat (top), with sonar visible (bottom).

In the literature, there are two primary approaches in solving this type of online navigation problems. BUG Algorithms [1] provide easy to implement strategies which work well in simple environments with small obstacles. However, they do not have strong performance guarantees and their performance can be arbitrarily bad as the obstacles grow larger. In contrast, online navigation algorithms with provable performance guarantees have been proposed in the literature [2–4]. However, these algorithms are rarely imple-

The authors are with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, USA. This work was supported by a MnDRIVE RSAM grant and National Science Foundation Awards #1111638 and #0917676. Emails: {plonski, jvander, peng0175, noori, isler}@cs.umn.edu. A preliminary version of this paper was submitted to the 2015 International Conference on Robotics and Automation.

mented on real robot systems since they do not address motion and sensing constraints.

In this paper, we present novel navigation strategies which are both provably efficient and suitable for field implementation. We first start with the case of a rectangular obstacle with unknown size. The justification for this assumption is that many artificial objects (ships, docks etc.) have rectangular shapes. Moreover, long shorelines or boundaries of vegetated areas can be approximated by a line. Obtaining the orientation of the line or the face of the rectangle amounts to fitting a line to initial readings.

We provide an adaptation of the doubling strategy (explained in Section 3) which accounts for motion and sensing constraints and analyze its performance. While testing the algorithm in field experiments, we realized that in some cases it becomes very hard to fit a line to initial readings. Moreover, the assumption of linearity is violated in most settings. Therefore, we also study the problem in a more general setting where we are given two parallel lines bounding an arbitrarily shaped obstacle. Here the parallel lines represent a weak prior on the extent of the obstacle. Our analysis shows that this generality comes at the expense of slightly reduced theoretical performance. However, field experiments demonstrate the effectiveness of the strategy.

2 Related Work

Several techniques have recently been developed to enable obstacle avoidance in sonar equipped Autonomous Surface Vehicles (ASVs).

Heidarsson and Sukhatme [5] implemented the Vector Field Histogram (VFH) [6] method for sonar-based obstacle avoidance. They also introduced an echo filtering process, which averaged the local maximas. Vector field methods are subject to local minima and the same vehicle with the same potential function in a different environment can result in success or failure with no guarantee.

Calado *et al*. [7] used Histogramic In-Motion Mapping (HIMM) [8] to build grid maps of the sensed obstacle. A convex polygon obstacle is built based on the line segments extracted by Hough transform. Given the constructed map, a potential field method was used for navigation.

Petillot *et al*. mapped the surrounding obstacles by segmenting and extracting obstacle features from the sonar image [9]. Then vehicle motion planning was converted into a nonlinear programming problem while the extracted obstacles were treated as inequality constraints [10]. Nonlinear programming approaches can be less succeptible to local minima than vector field methods. However, they require prior knowledge of the location and size of the obstacles. They also do not guarantee the length of the path and the time required to avoid an obstacle with arbitrary shape and size, whereas our approach can handle arbitrarily large obstacles and still reach the target in finite time with a near optimum competitive ratio.

3 Preliminaries

An *online algorithm* is an algorithm which does not have access to its entire input in advance. Instead, the input is revealed during the execution of the algorithm. For example, a memory management algorithm must choose which pages to retain in the cache without knowing future page requests. The obstacle avoidance problem studied in this paper is an online problem since we must choose a motion strategy that reacts to the sonar measurements received during execution without knowing the exact geometry of the obstacle in advance.

The performance of an online algorithm A is measured using its competitive ratio which is given by

$$c(A) = max_{\sigma} \frac{A(\sigma)}{OPT(\sigma)} \tag{1}$$

where σ varies across all inputs, $A(\sigma)$ is the performance of A for input σ and $OPT(\sigma)$ is the optimal *offline* performance – i.e. the performance of an optimal algorithm which has access to the entire input σ in advance. The competitive ratio is a measure of worst-case deviation from the optimal offline behavior. Further information on online problems can be found in [11].

The *lost-cow problem* is a classical online optimization problem which highlights important aspects of online algorithm design. In this problem, a short sighted cow is lost and tries to find the only gate on a straight fence. The problem is formulated as follows. The cow and the gate are on a line, the gate's location is unknown, and the cow starts from $x = 0$ in order to find the gate. The true location is chosen by an adversary. Note that the cow can not simply pick a direction and move until finding the gate as this strategy would have unbounded competitive ratio (the adversary chooses the gate location in the opposite direction.) Instead, the cow can follow the so-called *doubling strategy* to effectively find the gate: Initially, at round $i = 0$ the cow is at $f_0 = 0$. It moves in such a way that at the i^{th} round, the cow is at location f_i where $f_i = (-2)^{i-1}$ for $i \geq 1$. In other words, in odd rounds the cow is to the right of the origin while in even rounds the cow is to the left of the origin (Fig. 5). Using elementary computations, it can be shown that the doubling strategy has a competitive ratio of 9 [12].

Due to sensing and motion uncertainty and constraints, designing a similar online algorithm for obstacle avoidance is non-trivial. In this paper, we present novel online strategies, analyze their competitive ratios and validate them in field experiments.

4 System Description

Our test system is an Oceanscience Q-Boat 1800D[1]. The Q-Boat is 1.8m long, with a cruising speed of about 1 m/s and a turning radius of $R_t = 5m$. During the experiments, we moved at a slower speed of about 0.5 m/s. The Q-Boat is autonomously controlled with a laptop computer. Localization was achieved through the use of a GPS unit and a compass, filtered with an Extended Kalman Filter.

A forward-looking, single-beam sonar unit was mounted on the bow of the vehicle, as shown in Figure 1. The sonar unit is an Imagenex 852 Digital Echo Sounder[2]. It has a conical acoustic transducer with 10 degree beam width. The sonar sound frequency is 675kHz. As configured the sonar unit provides 500 measurements for each pulse, each measurement representing the intensity of the sonar return in an evenly spaced range bin determined by the configured maximum range. For the maximum range of 50m, each range bin represents a 0.1m increment. The sonar pulses were sent once a second. After each pulse, we smooth the data returned by convolving the 500 bin vector with a Gaussian filter that has $\sigma = 15$. For each return we detect the distance that has the maximum return intensity. If this intensity is above our threshold of 30 out of 127, we place an obstacle point at the appropriate distance in the appropriate direction from the estimated position of the boat.

Heidarsson *et al* [5] indicated that tilting the sonar by 10 degrees towards the lake surface or towards the lake bottom had little effect on the detection of obstacles. However we found that in shallow water, tilting the sonar could cause the bottom to be detected as an obstacle even if the water was clearly deep enough for our ASV to comfortably operate (1.5 meters in depth). In the end we decided to operate with the sonar tilted downward slightly. This ensured that the ASV could clearly detect the rising lakebed before the shore, but did not cause any false positives when the ASV was facing away from the shore.

5 Motivation and Problem Formulation

We address an online navigation problem, where a vehicle similar to the ASV described above detects an obstacle. First we describe the model used for the ASV. See also Figure 2.

Motion Constraints: The ASV can position its rudder to direct thrust from a single motor. The maximum angle of the rudder, β, determines a minimum turning radius r. The forward velocity can be set independently of the rudder. In the remainder of the paper, we consider the forward velocity fixed at a constant, labeled v. In practice, the forward velocity determines the time to complete a turn, but has little effect on the turning radius. Thus, we assume a constant turning radius for the boat, given by $R_t \approx 5m$.

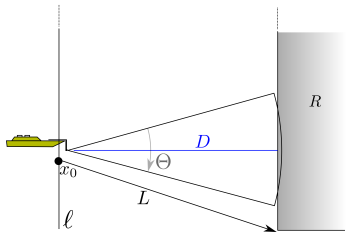

Fig. 2 The setup for Problem 1. The robot, moving from the left, encounters the rectangular obstacle R. With no information other than the orientation of the leading edge, the robot must find the path past the obstacle (L) or one of comparable length. The robot is equipped with a range sensor which can sweep out an angle Θ, and detect obstacles up to range D.

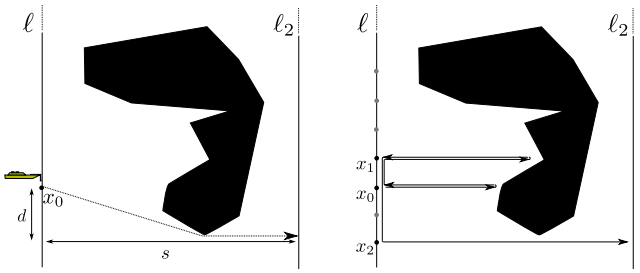

Fig. 3 Left: Problem setup for Problem 2. The optimal path is shown as an arrow, which lies tangent to one of the extreme points of the polygon. The optimal path, therefore, has length at least $\sqrt{d^2+s^2}$. The robot will move up and down the line ℓ, probing forward for the extreme points of the polygon by travelling toward line ℓ_2, up to distance s, or returning to ℓ if it has encountered the obstacle. Right: An example execution of `AdvanceRetreat`. The goal is to make sufficient progress s past the obstacle. The robot chooses points to *probe* according to the doubling strategy. Each point is checked until the robot is able to make sufficient progress.

Sensing Constraints: The ASV is equipped with a sonar (a range sensor), which detects the ranges to objects in a cone of angular width Θ, and up to a maximum range D. Both Θ and D are measured from the front of the vehicle. In our system, Θ is on the order of 10 degrees.

Next we define the obstacle avoidance problem: We start with an idealized case where the obstacle is a rectangle R. Suppose from the initial measurement, the ASV can infer the line containing a side of R. How can the ASV go around the nearest corner of R as quickly as possible? We formalize this problem as follows.

Problem 1 (Rectangular Obstacle) Given a rectangle R, a line ℓ containing the nearest edge of R, and a vehicle subject to motion and sensing constraints described above, compute a strategy for the vehicle to reach the nearest corner of R as quickly as possible using online sensor measurements (i.e. without knowing the exact shape of R in advance).

The problem setup is illustrated in Figure 2.

[1] www.oceanscience.com
[2] www.imagenex.com

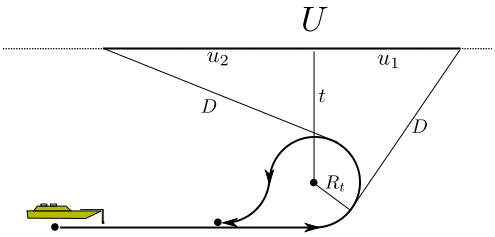

Fig. 4 The Sweep maneuver. The robot with maximum sensing distance D, and beginning the maneuver from distance $t+R_t$ can search an a portion of the obstacle with width $U \geq 2u_1 = 2\sqrt{R_t^2+D^2-t^2}$.

In the next section we present an online algorithm for Problem 1, show that it has constant competitive ratio and demonstrate its performance on the field. The strategy relies on two assumptions (1) the line ℓ supporting the obstacle can be obtained from initial readings, and (2) the shape of the obstacle is a rectangle. Field experiments reveal that these assumptions can be too strong in some cases. This lead us to a second, more general problem.

Problem 2 (Arbitrary Obstacle Bounded by Lines) Given two parallel lines, ℓ and ℓ_2, which are known to contain a single obstacle, and a vehicle subject to motion and sensing constraints as described, compute a strategy for the vehicle to reach the second line ℓ_2 as quickly as possible using online sensor measurements.

This second problem is illustrated in Figure 3. In Section 7, we present an extension to our algorithm for Problem 1 which can solve the second problem. The generality comes at the expense of increased competitive ratio, but field experiments reveal the algorithm is effective in practice.

6 Strategy for Problem 1 and Analysis

In this section we present a solution to Problem 1. We are required to use motion primitives that respect the minimum turning radius of our ASV. The first, Sweep, is used to search the portion of the obstacle which is near the robot. If a corner of the rectangle is detected, the robot can plan a clear path around the obstacle. The second, GoTo, simply moves the robots between two points.

Sweep: The robot, moving on line ℓ, begins a turn toward the obstacle along a circle with radius R_t. When the turn is $\frac{3}{4}$ completed, the robot re-aligns with the line ℓ, and moves in the opposite direction. This maneuver is illustrated in Figure 4.

GoTo: The robot, which is on line ℓ and oriented parallel to ℓ, moves straight ahead until a destination point on ℓ is reached.

Some simple calculations reveal the following properties of these operations.

1. Sweep traces out a path of length $2\pi R_t$.
2. After Sweep has been executed, the robot is facing the opposite direction to the direction it was facing initially.
3. Sweep searches a portion of the obstacle of width $U \geq 2\sqrt{R_t^2+D^2-t^2}$, when the maneuver is started at distance $t+R_t$, as shown in Figure 4
4. GoTo follows a path of length equal to the Euclidean distance between two points.

Algorithm 1 CircleSweep

```
θ ← orientation of the line ℓ
i ← 0
x_i ← (0,0), origin at the point of first detection
Sweep
while obstacle detected at x_i do
    i ← i+1
    x_i ← ((−2)^{i−1} cos θ, (−2)^{i−1} sin θ)
    GoTo x_i and Sweep
end while
```

We can now describe our algorithm for finding a corner of a rectangular object. We call the algorithm CircleSweep. Let the robot detect the obstacle at position x_0, which we treat as the origin. Similar to the lost-cow algorithm described in Section 3, the robot will GoTo waypoints which alternate left and right from x_0, such that each point x_i is at position $U(-2)^{i-1}$. At each of these points, including the first detection point, the robot will execute a Sweep, which simultaneously scans the object for the edge and turns the robot toward the next waypoint. The algorithm terminates when the robot either detects the edge, or detects that it has passed beyond the edge. See Algorithm 1 for the detailed steps of this algorithm.

The analysis of the cost of this algorithm proceeds in two parts. First, we bound the distance travelled during all GoTo phases, then, we show that the cost of all the Sweep operations is bounded with respect to the optimal cost. Combining these, we prove a competitive ratio in Theorem 1.

Lemma 1 (GoTo cost)
The GoTo cost is upper bounded by $12d$*, where* d *is the distance to the closest point from which the robot could detect the corner.*

Proof The last GoTo operation occurs immediately before the robot performs the Sweep operation that detects the corner of the rectangle or detects that the robot has moved beyond a corner of the rectangle. Let d be the nearest point to the origin, from which the robot could detect the corner while performing a Sweep. In the worst case, the robot travelled to a position just short of detecting the corner, (i.e., $d-\varepsilon$). Note, the lost-cow algorithm will arrive at the point d while traveling no farther than $9d$ [12]. However, we cannot stop at d because we cannot continually sense the obstacle. Instead we must continue on to the next Sweep location.

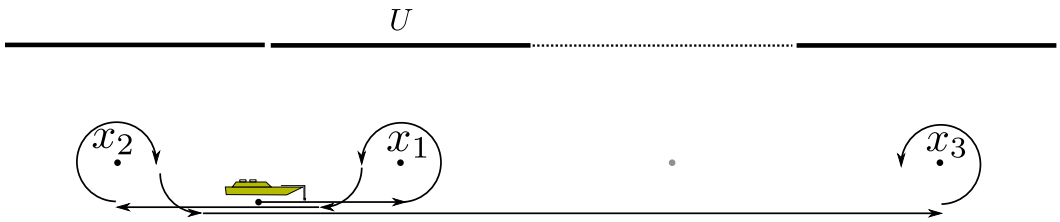

Fig. 5 The doubling strategy. At i^{th} round, the boat is at $x_i = (-2)^{i-1}$. At each point, a `Sweep` maneuver is performed to search a portion of the obstacle for the corner.

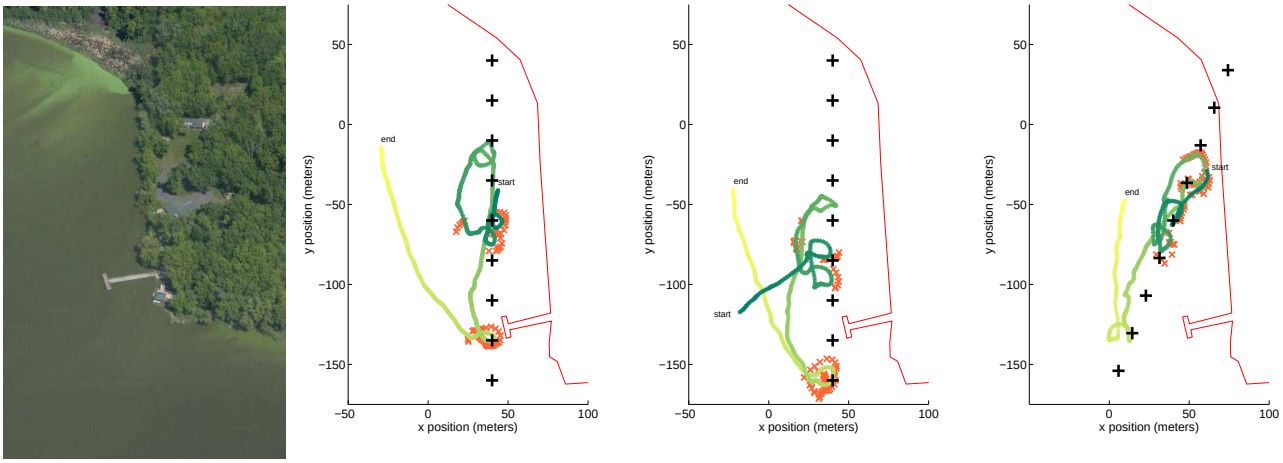

Fig. 6 A Google Maps satellite image of the Lake Staring shoreline, and three trial executions of `CircleSweep`, using the shoreline of Lake Staring as the object to avoid. The initial point the object was detected was assumed to be at $(40,-60)$. The crosses give the orientation of the line the robot moves along, and they are spaced apart by 25 meters, corresponding with our $U = 25m$. The shoreline is in red, the detected sonar objects are x markers, and the path followed by the ASV is labeled. On the left and in the middle, the corner of the object is mistakenly detected at the third `Sweep` operation. On the right, the corner of the object is mistakenly detected at the fourth `Sweep` operation. This indicates that the algorithm must be extended if it is to perform well in this practical situation.

Because the robot travels to the point $d-\varepsilon$ in step $n-2$, then to $-2d$ in step $n-1$, the final sweep location is at $4d$. Thus, we have "overshot" the location d by an additional a travel distance of $3d$ in the worst case, which we add to the cost of the lost-cow algorithm.

Lemma 2 (`Sweep` cost)
Let d be the distance to point from which the nearest corner of the rectangle can be detected. Then the total cost of all `Sweep` operations is less than $4\pi R_t \lceil \log_4 \frac{d}{U} \rceil$.

Proof Let d be the nearest point to the origin, from which the robot could detect the corner while performing a `Sweep`. We will just consider the "positive" steps, with $x_i > 0$ and double the result. Then each positive step reaches $x_i = U \cdot 4^i$. The algorithm terminates when the robot reaches a point beyond d and completes a `Sweep` operation. If the n^{th} step is the first to pass d, then $U \cdot 4^{n-1} \leq d \leq U \cdot 4^n$, which implies $n = \lceil \log \frac{d}{U} \rceil$ steps are required on the positive side, or at most $n = 2\lceil \log_4 \frac{d}{U} \rceil$ steps are required overall. Since each sweep has a cost of $2\pi R_t$, the lemma statement follows.

Theorem 1 *`CircleSweep` has a competitive ratio of* $13 + 4\pi \frac{R_t}{U}$

Proof By combining the previous two lemmas, we see that the total cost of using `CircleSweep` to find the corner point is less than $12d + 4\pi R_t \lceil \log_4 \frac{d}{U} \rceil$. Then, the robot must travel past the obstacle, which adds a cost less than L. Dividing this cost by the optimal cost, L the resulting ratio is $\frac{12d}{L} + \frac{4\pi R_t \lceil \log_4 \frac{d}{U} \rceil}{L} + \frac{L}{L}$. Since $d < L$,

$$c(\texttt{CircleSweep}) \leq 13 + \frac{4\pi R_t \lceil \log_4 \frac{d}{U} \rceil}{d} \tag{2}$$

$$= 13 + 4\pi R_t \frac{U \lceil \log_4 \frac{d}{U} \rceil}{Ud} \tag{3}$$

$$= 13 + 4\pi \frac{R_t}{U} \frac{\lceil \log_4 \frac{d}{U} \rceil}{\frac{d}{U}} \tag{4}$$

Also note $\frac{\lceil \log_b x \rceil}{x} \leq 1$, producing the theorem statement.

Note, the last sweep operation will orient the boat so that it can travel on a straight path past the obstacle, thus Theorem 1 does not include an additional turning cost.

6.1 Field Experiments

For our first field experiments we executed `CircleSweep` at Lake Staring, Minnesota, USA. We used the north east shoreline of the lake as a proxy for a large-scale obstacle to avoid. At first we attempted to fit a line to the sonar returns observed during the initial circle performed after detecting the obstacle, but we found that there was not nearly enough information to decide the direction of the line. Our strategy requires a good estimate on the line direction or it may simply retreat away from the object and declare the problem solved. In the end, to execute `CircleSweep` we needed to manually define the line to walk along. We used $t = 10m$ and $U = 25m$; this is reasonable as long as $D \geq 15.21m$. D is nominally $50m$, but in practice, given our tilt angle, 20 meters is a good estimate of the true D at which we can reliably detect an obstacle.

The paths followed by the robot along with the estimated positions of all detected sonar objects are shown in Figure 6. The sonar returns indicate that the shallow water close to the shore is correctly detected as an obstacle to avoid. Our strategy assumes the obstacle is a rectangle with known orientation but unknown dimensions. In practice this is almost never the case, and the basic strategy performs poorly when its assumptions are violated: In each of our three trials it would have incorrectly detected the rectangle corner and terminated early.

7 Strategy for Problem 2 and Analysis

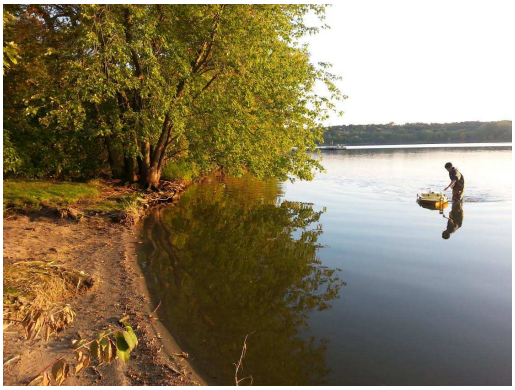

Fig. 8 Close up of the experiment area. Approximately 180 meters of shoreline was used as a proxy for a large obstacle. The shore curved east, providing a convenient corner to serve as the edge of the obstacle.

Given the insights from our first field experiments, we now solve the more general case presented in Problem 2. In this section we propose and analyze an algorithm for moving past an arbitrarily-shaped object. This object is paramaterized by two parallel lines ℓ and ℓ_2 which are inferred based on the assumed width and orientation of the obstacle. The safety line ℓ is assumed to lie on the same side of the obstacle as the robot, and the goal line ℓ_2 is assumed to lie on the opposite side. The distance between the lines is s.

Algorithm 2 AdvanceRetreat

$\theta \leftarrow$ orientation of line ℓ
$i \leftarrow 0$
$\mathbf{x}_i \leftarrow (0,0)$, origin at the point of first detection
probe forward by travelling toward ℓ_2
while obstacle detected during last probe toward ℓ_2 **do**
    complete *probe* by returning to $\mathbf{x}_i$
    $i \leftarrow i+1$
    $\mathbf{x}_i \leftarrow ((-2)^{i-1}\cos\theta, (-2)^{i-1}\sin\theta)$
    `GoTo` $\mathbf{x}_i$
    probe forward by travelling toward ℓ_2
end while

First, we discuss the optimal path for a given obstacle. Consider the polygonal obstacle shown in Figure 3. The optimal path, starting at position x_0, does not enter the convex hull of the polygon. It passes through a "corner point"– a point which is a maxima or minima with respect to the line ℓ, as shown. Thus, the optimal path has length at least $\sqrt{d^2+s^2}$.

Our algorithm, called `AdvanceRetreat`, proceeds as follows. From the starting location, which is defined as the origin, the robot will move forward (perpendicular to ℓ) up to distance s, and return if an obstacle is detected. This is called a *probe* step. Upon returning, the robot will move to a location which is distance $U \cdot (-2)^{i-1}$ along the line ℓ and repeat the process (see Figure 3 and Algorithm 2).

There are two important differences from `CircleSweep`. First, the robot cannot "sweep" out a portion of the obstacle, because during a probe it may only detect a very small part of the obstacle. In this case, we terminate when we can detect no part of the obstacle, not just when we detect the corner. In practice, U can be represented by the beam width, and a non-detection is simply no echo returns. Second, we assume that the robot can always turn around to return to the line, thus the obstacle should not have very narrow "channels" in which the robot can get stuck. In most settings, this assumption is not restrictive.

We now analyze the cost of this strategy. First, note that the distance travelled while searching for the closest location past the corner point is at most $12d$ as given in Lemma 1. What remains is to analyze the cost of each *probe* step.

Lemma 3 (Number of Probes)
During an execution of `AdvanceRetreat`*, the robot makes* $2\lceil \log_4 \frac{d}{U} \rceil$ probe *steps, each with cost less than* $2s + 2\pi R_t$.

Proof This cost of each probe step is upper bounded using s the max distance from ℓ to the object plus the cost to make four quarter turns. Without loss of generality, let d be the point on ℓ from which the *probe* operation would

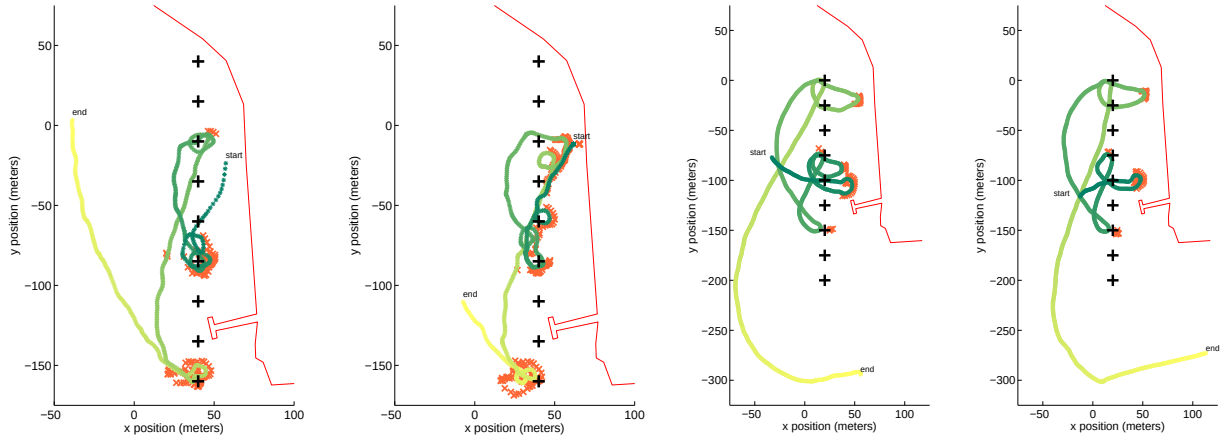

Fig. 7 Four trial executions of `AdvanceRetreat`, using the shoreline of Lake Staring as the object to avoid. The initial point the object was detected was assumed to be at $(40,-60)$ for the first two, and $(20,-100)$ for the second two. The crosses give the orientation of the safety line ℓ, and they are spaced apart at the unit distance $U = 25m$. The portion of the shoreline we are avoiding is assumed to lie between parallel north-south lines ℓ and ℓ_2, which is assumed to be $50m$ east of ℓ. The shoreline is in red, the detected sonar objects are x markers, and the path followed by the ASV is labeled. In each probe that is not a feasible route around the object, the shoreline is successfully detected. In the last two trials, the boat successfully navigates around the shoreline to the other side of ℓ_2.

first succeed in passing the obstacle. The algorithm terminates when the robot travels past d and executes a probe step. Using the same analysis as Lemma 2, we obtain an upper bound of $\lceil \log_4 \frac{d}{U} \rceil$ probe steps required on the positive side, or $2\lceil \log_4 \frac{d}{U} \rceil$ probes required in total.

Theorem 2 (Cost of `AdvanceRetreat`) *`AdvanceRetreat` has a competitive ratio that is* $\Theta(\log \lceil \frac{d}{U} \rceil)$.

Proof We know the robot travels no more than distance $12d + (2s + 2\pi R_t)\lceil \log_4 \frac{d}{U} \rceil$, by combining the travel (Lemma 1) and probe (Lemma 3) steps. Note that the optimal path is at least length $\sqrt{d^2+s^2}$, as illustrated in Figure 3. Thus,

$$
\begin{aligned}
&c(\texttt{AdvanceRetreat}) \\
&\quad = \frac{12d + (4s + 4\pi R_t)\lceil \log_4 \frac{d}{U} \rceil}{\sqrt{d^2+s^2}} \\
&\quad = \frac{12d}{\sqrt{d^2+s^2}} + \frac{(4s + 4\pi R_t)\lceil \log_4 \frac{d}{U} \rceil}{\sqrt{d^2+s^2}} \\
&\quad \leq 12 + \lceil \log_4 \frac{d}{U} \rceil \left(4 + \frac{4\pi R_t}{\sqrt{d^2+s^2}} \right),
\end{aligned}
$$

which proves the theorem statement.

Note that the part of the competitive ratio that grows logarithmicly is the cost from the logarithmic number of probes. If s is small, the cost of each probe is also small and the competitive ratio of `AdvanceRetreat` is linear. Next we present results from repeated field tests of the proposed algorithm.

7.1 Field Experiments

For our second field experiments we executed `AdvanceRetreat` at Lake Staring. The obstacle to avoid was the same north east part of the shoreline as before. The safe line ℓ was selected manually but the trajectory of the ASV was determined online from the sonar measurements. As before, we used $U = 25m$ and $t = 10m$.

The paths followed by the robot, along with the estimated positions of all detected sonar objects, are shown in Figure 7. The boat successfully navigated around the obstacle in both of the trials where it was run to completion, and at every probe the boat correctly determined whether or not there was an object present. These results indicate that our strategy is in practice a feasible method to find a path around an object.

We found that `AdvanceRetreat` performed much better in practice. This was because `CircleSweep` requires that the obstacle is a line with known orientation, and the boundary of Lake Staring is poorly approximated by a straight line. However, the northeast boundary of Lake Staring is well approximated as a polygon that stays within distance s of a straight line; this was the setup of Problem 2 for which we used the `AdvanceRetreat` strategy. We expect many common obstacles are well approximated by the parallel bounding lines ℓ and ℓ_2.

8 Conclusion

In this paper we presented strategies for an Autonomous Surface Vehicle equipped with a fixed angle sonar to navi-

gate around an obstacle. We first studied a simple case where the obstacle is a rectangle of unknown size. For this problem, we presented a strategy with a constant-factor competitive ratio. In other words, the ratio of the distance traveled to the distance traveled by the optimal offline solution is bounded by a constant. In the field, we observed that fitting a line to the sonar sensor readings was difficult. This made it difficult to determine the orientation of a straight obstacle. Furthermore, a realistic non-straight obstacle was handled poorly by our first strategy, even when the line was a reasonable fit for its boundary. Therefore, we also studied a more general case in which the obstacle has arbitrary shape but it is contained between two bounded lines. The competitive ratio of our algorithm for this case depends on the distance between the lines. In field experiments, we showed that the algorithm can be executed in a robust fashion to navigate around large, unknown obstacles including shallow waters around a shoreline.

An interesting but challenging avenue for future work is to study navigation in the presence of multiple unknown obstacles and provide performance bounds. Another avenue for research is to study an environment with moving obstacles such as other boats.

References

1. J. Ng and T. Bräunl, "Performance comparison of bug navigation algorithms," *Journal of Intelligent and Robotic Systems*, vol. 50, no. 1, pp. 73–84, 2007.
2. P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosén, and M. Saks, "Randomized robot navigation algorithms," in *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1996, pp. 75–84.
3. S. Albers and M. R. Henzinger, "Exploring unknown environments," *SIAM Journal on Computing*, vol. 29, no. 4, pp. 1164–1188, 2000.
4. P. Berman, "On-line searching and navigation," in *Online Algorithms*. Springer, 1998, pp. 232–241.
5. H. K. Heidarsson and G. S. Sukhatme, "Obstacle detection and avoidance for an Autonomous Surface Vehicle using a profiling sonar," *2011 IEEE International Conference on Robotics and Automation*, pp. 731–736, May 2011.
6. J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots," *Robotics and Automation, IEEE Transactions on*, vol. 7, no. 3, pp. 278–288, 1991.
7. P. Calado, R. Gomes, M. B. Nogueira, J. Cardoso, P. Teixeira, P. B. Sujit, and J. B. Sousa, "Obstacle avoidance using echo sounder sonar," *OCEANS 2011 IEEE - Spain*, pp. 1–6, Jun. 2011.
8. E. Rimon and D. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, 1992.
9. Y. Petillot, I. T. Ruiz, D. Lane, Y. Wang, E. Trucco, and N. Pican, "Underwater vehicle path planning using a multi-beam forward looking sonar," in *OCEANS'98 Conference Proceedings*, vol. 2. IEEE, 1998, pp. 1194–1199.
10. D. Lane and Y. Wang, "Subsea vehicle path planning using nonlinear programming and constructive solid geometry," *IEE Proceedings - Control Theory and Applications*, vol. 144, no. 2, pp. 143–152, Mar. 1997.
11. A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. Cambridge University Press, 1998.
12. J. C. C. Ricardo A. Baeza-Yates and G. J. E. Rawlines, "Searching with uncertainty," Indiana Univeristy, Tech. Rep., 1988.