Perimeter Patrol on Autonomous Surface Vehicles using Marine Radar

Elena Oleynikova, Nicole B. Lee, Andrew J. Barry, Joseph Holler and David Barrett
Olin Intelligent Vehicles Lab
Franklin W. Olin College of Engineering, Needham, MA 02492

***Abstract*— Perimeter patrol enhances the utility of autonomous surface vehicles (ASVs) by enabling many security and scientific missions, including harbor protection, water sampling, and geological survey. We present a novel approach to perimeter patrol that uses only two sensors: commercial off-the-shelf available marine radar and the heading information from a GPS. Our algorithm performs computer vision morphological operations on the radar image to find a suitable path around shore and choose an appropriate next waypoint. Our method has proved robust to a variety of field conditions, allowing us to demonstrate the autonomous navigation of a 3.5 km perimeter lake.**

I. Introduction

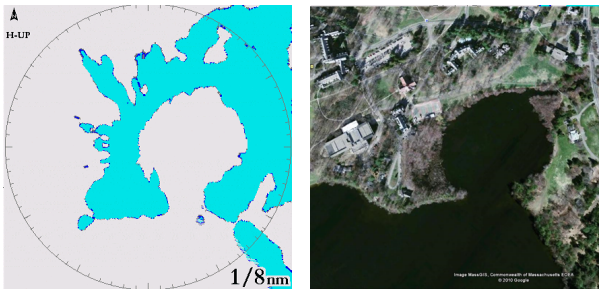
Autonomous Surface Vehicles (ASVs) have many applications, including surveillance, patrol, and various hydrographic and oceanographic surveying methods [1]. Perimeter patrol is an important aspect of ASV behavior, allowing a vehicle to plot an appropriate course without human interaction and respond to a changing environment while maintaining a constant distance from shore.

In this paper we introduce a novel method for perimeter patrol using marine radar. The radar allows our vehicle to detect the shoreline, and other obstacles, providing data for intelligent decisions regarding islands, small boats, buoys, and even waterfowl. Additionally, the long range of the marine radar provides significant flexibility in the ASV's patrol distance offshore. Figure 1 compares raw radar data and satellite imagery, showing the quality of the shore contour image. The robustness of the radar data makes our algorithm effective and reliable.

II. Related Work

Unmanned and Autonomous Surface Vehicles (USVs and ASVs) date back to World War II, though it was not until the 1990s that they reached widespread use. They have many military applications; in particular, ASVs are frequently used for minesweeping, reconnaissance, and surveillance. Many minesweeping surface craft, especially those developed before 1990, have been remotely operated. Only recent Navy minesweeping ASVs, such as the Remote Mine-hunting System, have been truly autonomous. The Navy has also developed numerous ASVs for reconnaissance and surveillance missions, including the Owl MK II and the Roboski, which are being used for unmanned harbor security missions [2].

Another common application of ASVs is in scientific research, as ASVs are often an ideal platform for sample collection and oceanographic research. One example of such



(a) Raw Radar Image (b) Satellite Map

Fig. 1. Comparison of raw radar data (a) and satellite shoreline information (b). The satellite image is scaled and rotated to match the radar image. Note that even without post-processing the entire bay is identifiable using the radar.

an ASV is SESAMO, a robot that is designed to collect data and samples for the study of the sea-air interface [3]. In order to facilitate scientific research, many ASVs are designed to support Autonomous Underwater Vehicles (AUVs). Ferreira *et al.* present an ASV platform capable of traversing a river and supporting AUV missions by acting as a dock to its companion AUV and using an acoustic modem to communicate with the submerged vehicle [4]. Similarly, the ASIMOV project explores on coordination between ASVs and AUVs for scientific research. The ASV described by Pascoal *et al.* is capable of marine and bathymetric data acquisition, precise path following, and assisting the AUV by serving as an acoustic relay between the AUV and a support vessel [5].

These ASVs have a wide range of autonomous behavior. While Benjamin *et al.* use a behavioral model to handle avoiding other surface craft in accordance with coast guard regulations [6], SESAMO relies on human-programmed waypoints for navigation [3].

Marine radar has been used for several autonomous surface vehicle applications, such as maritime search and rescue. Shi *et al.* use radar for detecting small, nearby objects that could potentially be people overboard. Shi *et al.* also use radar data filtering techniques to discard the effect of adverse weather conditions on radar data [7]. Larson *et al.* use a radar for short-term reactive obstacle avoidance. They fuse the radar data with other short-range sensor data to create an occupancy grid and make decisions about nearby obstacles using more traditional path planning methods [8]. Almeida *et al.* use marine radar fused with a network of other sensors to detect obstacles and avoid collisions [9].

978-1-4244-5222-4/10/$26.00 ©2010 IEEE

Fig. 2. Olin Intelligent Vehicles Lab ASV Medea. Medea is based on a 12-foot Hobie catamaran and has a fold-down mast for an 18 inch radome.

We are the first to propose a shore following algorithm using marine radar. In addition to a novel application for marine radar, we also describe a new method for path planning with a given occupancy grid.

III. Platform

A. ASV

We implemented this algorithm on the ASV Medea (Fig. 2), a research vehicle developed in the Olin College Intelligent Vehicles Lab. Specifically, Medea is a 12 foot long catamaran propelled by an electric vectored thrust system, and is equipped with a GPS and marine radar. Medea is based on earlier Olin College Intelligent Vehicles Lab ASVs [10].

Medea uses a high-precision NavCom GPS for localization. A low-cost Garmin 18" analog radar dome (shown on top of the sensor mast) is connected to a Garmin Chartplotter, whose VGA output is input to the main computer through a VGA-to-USB device. Though there are other systems for accessing radar data through a PC, we found this to be the easiest and lowest cost implementation.

B. Software Platform

Our software platform is written in LabView. The lowest level runs on a National Instruments cRIO module and manages low-level motor control of the vectored thrust system. The middle level, Medea's main computer, reads sensor inputs, processes those data, and computes a desired driving command. The highest level of command is an optional ground station, connected to the vehicle via wireless ethernet bridge, which visualizes the path of the vehicle and transmits mission parameters.

We use a highly asynchronous, parallel architecture for sensor acquisition and data processing. Each sensor driver, data processing algorithm, and motor controller runs in its own thread. This architecture allows us to acquire and process sensor data at different rates. For example, our software updates and processes radar data once every two seconds, while updating GPS and drive commands more than once per seciond. Another advantage of this software system is the log saving and replay functionality, which allows us to record data in the field. This, in turn, allows us to test our algorithm with data from previous field tests, saving both development and field testing time.

IV. Algorithm

Our perimeter patrol algorithm can be divided into two stages: path identification and direction selection.

In our implementation, the radar is oriented heading up, where the vehicle is in the exact center of the image and is facing toward the top of the image. The radar data is stored as a bitmap, with detected objects represented as non-zero values, and background represented as zero. Since there is a constant relationship between pixels (in the image) and meters (in the physical world), direction and distance to any obstacle in the radar image can be easily calculated.

A. Path Identification

Our algorithm first finds a path that is offset a set distance, d_{offset}, from the shore. In order to accomplish this, we perform the following operations:

1) Dilation: The first operation performed is a morphological dilation of the shore image with a 3x3 square structuring element. The operation is repeated a number of times given by Eq. 1, where d_{offset} is the desired offset from shore, $c_{smoothing}$ is a parameter for smoothing purposes, and $c_{m/px}$ is the number of meters per pixel [11]. In our implementation, $c_{smoothing}$ has a value of 5, which was determined experimentally.

$$d_{offset} * 1/c_{m/px} + c_{smoothing} \tag{1}$$

2) Erosion: We then perform an erosion operation with the same structuring element as above, and repeat it $c_{smoothing}$ times. This reduces the shore path to the correct size and performs a smoothing operation on the shoreline, discarding many of the more erratic features of the shore [11].

3) Thinning: The thin operation, also known as perimeter detection or binary edge detection, removes all but the edges of particles. That is, for any given pixel, the pixel remains set (has a non-zero value) if it has at least one neighbor that is not set (has a zero value). This operation leaves only the intended path around the shore as non-zero, shown in red in Fig. 3.

B. Waypoint Selection

After isolating the path, we then select the next waypoint, represented as a position vector $\vec{W}$, to direct the vehicle toward. The steps are as follows:

1) Staying on Path: We begin by finding the distance (from our current position, at the center of the image) to each point along the identified path, as well as the distance to the closest point on the shore. Specifically, $\vec{P}_{shore}$ is the vector to the closest position on the shore, $\vec{P}_{path}$ is the vector to the closest position on the path, and $\vec{P}_{current}$ is the current position vector of the vehicle. Eq. 2 describes the desired relation, where $d_{proximity}$ is a parameter for how close to the path the vehicle must be to be considered on path, in meters.

Fig. 3. Radar image after undergoing morphological operations to isolate the path around the shore. The white objects are obstacles as detected by radar, and the red path represents the allowed path around the shore.

$$||\vec{P}_{path} - \vec{P}_{current}|| < d_{proximity} \tag{2}$$

If the relation in Eq. 2 is false, then the vehicle is not on the desired path, and we set the waypoint to be the closest point on the path, $\vec{W} = \vec{P}_{path}$, to correct this error.

2) Selecting a direction: If the vehicle is within $d_{proximity}$ of the path, we select a waypoint on the path that would result in patrol in the correct direction – either clockwise or counterclockwise, as selected by the user.

We achieve this by first finding all points on the path that are within $d_{waypoint}$ of $\vec{P}_{current}$, and denote this set as S. $d_{waypoint}$ is a parameter for the desired distance to next waypoint, in meters. For every position vector in S, denoted $\vec{S}_i$, we calculate two vectors (Eq. 3), and then take their cross-product (Eq. 4).

$$\begin{aligned} \vec{a} &= \vec{P}_{shore} - \vec{S}_i \\ \vec{b} &= \vec{P}_{shore} - \vec{P}_{current} \end{aligned} \tag{3}$$

$$\vec{c} = (\vec{a} \times \vec{b}) \tag{4}$$

Since $\vec{a}$ and $\vec{b}$ only have $\hat{i}$ (horizontal direction in the image plane) and $\hat{j}$ (vertical direction in the image plane) components, their crossproduct is in the $\hat{k}$ (out of the image plane) direction. Based on which direction we wish to patrol in, we select either the position vector in S with the largest magnitude of $\vec{c}$ (if traveling counterclockwise) or the smallest magnitude of $\vec{c}$ (if traveling clockwise) to be the next waypoint ($\vec{W}$).

C. Driving Command

Given a waypoint $\vec{W}$, we then formulate a velocity and heading driving command. By computing $\angle\vec{W}$, we obtain the desired change in heading. We can then add this value to the Course over Ground (CoG) from our GPS to give absolute desired heading. We use a user-defined velocity target, as the ideal behavior is to have the vehicle patrol at a constant velocity. Our software platform then uses the absolute desired heading and velocity combined with Speed over Ground (SoG) and Course over Ground (CoG) from the GPS to steer and drive the vehicle.

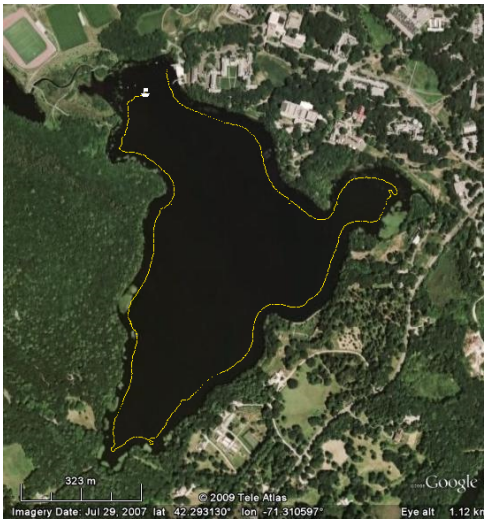


Fig. 4. GPS log of ASV Medea's lake patrol at Lake Waban. The yellow path is the path of the ASV around the lake. The patrol took 60 minutes and spanned approximately 3.5 km.

TABLE I
PARAMETER VALUES USED FOR FIELD TEST

Parameter	Symbol	Units	Value
Smoothing Constant	$c_{smoothing}$	iterations	5
Waypoint Distance	$d_{waypoint}$	meters	20
Proximity Distance	$d_{proximity}$	meters	5
Desired Offset Distance	d_{offset}	meters	11

V. RESULTS

We conducted perimeter patrol tests at Lake Waban, an inland lake with a perimeter of approximately 3.5 km. After several short range tests, we conducted an autonomous long-range mission. During this test, we circumnavigated Lake Waban, successfully avoiding obstacles and accurately tracing shore contours. The system correctly identified and avoided all landmasses and smaller obstacles, including small sailboats and even moving waterfowl. The path of the vehicle during the test is shown in Fig. 4, and Table I shows the parameter values used. It should be noted that most aberrations from the satellite-image shore line are due to the differences between the satellite image and the actual shore during our field test.

The quantifiable results of this test are shown in Fig. 5. The spike in error at $t = 16$ minutes is due to an implementation bug that has since been fixed. We find that the vehicle was within 1 meter of the desired distance from the shore 64.9% of the time. This number could be improved by decreasing the proximity radius, since the vehicle would track the path more

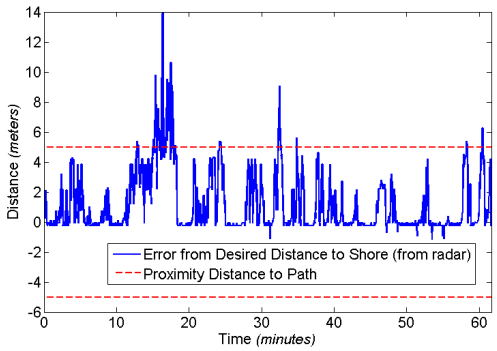


Fig. 5. Error between desired distance from shore and actual distance from shore as a function of time (blue). We also indicate proximity distance from shore (red) as a reference point. For the most part, the algorithm recovers from errors quickly, especially in cases over the proximity distance limit. In the area around $t = 16$ minutes, the high error is due to an implementation problem that has since been fixed.

closely. As can be seen from Figure 5, the vehicle's error from path exceeds the proximity radius only 7 times over the course of an hour-long test (a total of 5.14%, including deviations due to the implementation error), and recovered within 10 seconds in each case (excluding $t = 16$ minutes). Moreover, we find that the vehicle rarely moves closer to land than desired, spending only 0.15% of the travel time more than a meter too close to the shore. This is a particularly desirable property, as straying too close to shore may result in grounding and other complications that could damage or disable an ASV.

VI. Future Work

This paper presents a method for shore identification and tracking. We consider a number of additional uses for the system including utilizing more sensing, further processing, and reducing cost.

To improve obstacle detection, avoidance, and tracking, we are considering the addition of a LIDAR laser rangefinder to the system. The LIDAR would allow us to track non-water targets with improved accuracy at a greater update rate than the radar. Similar LIDAR-based tracking systems have been used extensively on land-based vehicles [12][13].

We can reduce the system's cost by removing the GPS and use only a digital compass's bearing. While this is a good solution for the shore-tracking application, GPS systems are low-cost and provide additional functionality for logging, waypoint, and tracking utility, so we expect that further applications will require GPS.

The addition of a vision system would allow us to label radar obstacles enabling new behaviors such as specific object tracking and following maneuvers. A vision system could also augment the obstacle detection methods we currently employ in a similar manner to combined LIDAR-vision based approaches [14].

VII. Conclusion

We have demonstrated the viability of a perimeter patrol algorithm using only marine radar and GPS heading. Our algorithm allows for robust shoreline patrol with low-cost sensors and limited computational requirements. We find that the algorithm is successful in a variety of shoreline and obstacles environments. We believe these features are ideal for an ASV in a patrol, surveillance, survey, or sampling mission close to shore.

This algorithm could also be used for obstacle avoidance given other types and quantities of sensors, as it operates on an occupancy grid and creates a suitable path around any sufficiently large obstacle. Thus, we could easily adapt this system for an autonomous ground vehicle or an autonomous surface vehicle with a different sensor array.

Acknowledgments

We would like to thank Jacob Izraelevitz for his mechanical work and software suggestions. We also thank Nikolaus Wittenstein, Andrea Striz, Joseph Holler, John Morgan, Joseph Kendall, and Tony Lopez for designing and building our test platform Medea during a program supported by Aurora Flight Sciences. Finally, we would like to thank Wellesley College for allowing us to use Lake Waban during our tests. Elena Oleynikova, Nicole Lee, Andrew Barry, and Joseph Holler are supported by Franklin W. Olin Scholarships.

References

[1] J. Manley, "Unmanned surface vehicles, 15 years of development," in *Proc. MTS/IEEE Oceans' 08 Conference, Québec*, 2008.

[2] J. Veers, "Development of the USV Multi-Mission Surface Vehicle III," in *5th International Conference on Computer Applications and Information Technology in the Maritime Industries*, 2006.

[3] M. Caccia, R. Bono, G. Bruzzone, E. Spirandelli, G. Veruggio, A. Stortini, and G. Capodaglio, "Sampling sea surfaces with SESAMO: an autonomous craft for the study of sea-air interactions," *IEEE Robotics & Automation Magazine*, vol. 12, no. 3, pp. 95–105, 2005.

[4] H. Ferreira, A. Martins, A. Dias, C. Almeida, J. Almeida, and E. Silva, "ROAZ autonomous surface vehicle design and implementation," *Robotica*, 2006.

[5] A. Pascoal, P. Oliveira, C. Silvestre, L. Sebastião, M. Rufino, V. Barroso, J. Gomes, G. Ayela, P. Coince, M. Cardew *et al.*, "Robotic ocean vehicles for marine science applications: the european asimov project," in *IEEE Oceans 1998*, vol. 1, 2000, pp. 409–416.

[6] M. Benjamin, J. Leonard, J. Curcio, and P. Newman, "A method for protocol-based collision avoidance between autonomous marine surface craft," *Journal of Robotic Systems*, vol. 23, no. 5, pp. 333–346.

[7] C. Shi, K. Xu, J. Peng, and L. Ren, "Architecture of vision enhancement system for maritime search and rescue," in *ITS Telecommunications, 2008. ITST 2008. 8th International Conference on*, 2008, pp. 12–17.

[8] J. Larson, M. Bruch, and J. Ebken, "Autonomous navigation and obstacle avoidance for unmanned surface vehicles," in *Proceedings of SPIE*, vol. 6230, 2006, p. 623007.

[9] C. Almeida, T. Franco, H. Ferreira, A. Martins, R. Santos, J. Almeida, J. Carvalho, and E. Silva, "Radar based collision detection developments on USV ROAZ II," May 2009, pp. 1 –6.

[10] J. Holler, "An unmanned surface vehicle for undergraduate engineering science education," Student Poster, *MTS/IEEE Oceans 2008 Quebec*, 2008.

[11] E. Breen, R. Jones, and H. Talbot, "Mathematical morphology: A useful set of tools for image analysis," *Statistics and Computing*, vol. 10, no. 2, pp. 105–120, 2000.

[12] J. Bohren, T. Foote, J. Keller, A. Kushleyev, D. Lee, A. Stewart, P. Vernaza, J. Derenick, J. Spletzer, and B. Satterfield, "Little Ben: The Ben Franklin racing team's entry in the 2007 DARPA urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 598–614, 2008.

[13] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Robotic Systems*, vol. 25, no. 8, pp. 425–466, 2008.

[14] J. P. Hwang, S. E. Cho, K. J. Ryu, S. Park, and E. Kim, "Multi-classifier based lidar and camera fusion," in *2007 IEEE Intelligent Transportation Systems Conference*, Sep. 2007.