Visual Obstacle Avoidance for Autonomous Watercraft using Smartphones

Tarek El-Gaaly[1] *, Christopher Tomaszewski [2], Abhinav Valada[2], Prasanna Velagapudi[2], Balajee Kannan[2] and Paul Scerri[2] **

[1] Rutgers University
[2] Carnegie Mellon University

Abstract. This paper presents a visual obstacle avoidance system for low-cost autonomous watercraft in riverine environments, such as lakes and rivers. Each watercraft is equipped with a smartphone which offers a single source of perceptual sensing, via a monocular camera. To achieve autonomous navigation in riverine environments, watercraft must overcome the challenges of limited sensing, low computational resources and visually noisy dynamic environments. We present an optical-flow based system that is robust to visual noise, predominantly in the form of water reflections, and provides local reactive visual obstacle avoidance. Through extensive field testing, we show that this system achieves high performance visual obstacle avoidance.

1 Introduction

Understanding and monitoring natural water systems is critical to understanding the eco-system in which we live. Monitoring large expanses of water is a challenge due to limited local observability (due to static sensors that only sense within the local vicinity), complex and dynamic environments, high cost and the need for constant supervision by human operators [1].

In order to achieve distributed monitoring of large water systems, a fleet of autonomous surface watercraft (also called airboats) has been developed by [1]. This fleet of low-cost autonomous watercraft (shown in figure 3) provide distributed and real-time data measurements over large bodies of water and have a broad set of applications, including: environmental monitoring, disaster mitigation, depth mapping and fish farming [1].

* [1]Computer Science Dept., Rutgers University: Tarek El-Gaaly - tgaaly at cs.rutgers.edu

** [2]Field Robotics Center, Robotics Institute, Carnegie Mellon University: Chris Tomaszewski - ckt at andrew.cmu.edu, Abhinav Valada - avalada at cmu.edu, Prasanna Velagapudi - pkv at andrew.cmu.edu, Balajee Kannan - bkannan at ri.cmu.edu, Paul Scerri - pscerri at cs.cmu.edu

Controlling a whole fleet of watercraft is a challenging process for human operators. A human operator has the capability of defining waypoints which sets the watercraft on pre-determined paths. These paths cover *approximate* regions of water. The regions are not exact because a number of factors affect the *actual* path of the boats. These factors include: GPS inaccuracy (mainly due to interference by overhanging and high trees), water currents and the presence of other boats. The watercraft commonly drift off their predefined path. More importantly, a human operator does not have the ability of defining each and every obstacle the boat will encounter in the environment. Obstacles such as lilies, banks, floating debris are not marked on reference maps and hence will adversely affect the boats (in many cases fatally causing the boat to get damaged or stuck). For this reason, maximal autonomy is needed to ensure that the fleet of watercraft are constantly up and running. To achieve maximal autonomy, watercraft require a system of visual obstacle avoidance (VOO) to be locally applied along their predetermined paths. This prevents watercraft from colliding with potential hazards, such as, other boats, floating debris and riverine banks.

In this work, we build an onboard real-time visual obstacle avoidance (VOO) system for low-cost multi-watercraft in riverine environments. The predominant challenges with such a system in riverine environments are the following:

1. Low computational resources
2. Visual sensing via monocular cameras
3. Visual noise in riverine environments

Fig. 1. Multiple watercraft robots autonomously monitoring oxygen and temperature levels in a riverine environment

The watercraft platform developed in [1] have mounted smartphones which provide visual sensing via monocular cameras. An optimized optical flow-based algorithm is built and tested to track potential obstacles. The VOO system is optimized to make it adapted to low computational resources on the smartphone which is also being used for the infrastructure of the watercraft fleet (*i.e.* communication with a central server, streaming commands to a servo controller onboard the watercraft, GPS localization and inertial sensing).

The VOO system is composed of two core visual components: optimized optical flow and reflection detection algorithms. The optimized reflection detection algorithm removes tracked features that correspond to object reflections on the water's surface. In addition feature trajectories corresponding to specular reflections and small harmless obstacles, such as, algae and leaves are filtered out by a clustering method which looks for high density regions over persistent tracked features.

After the visual processing, a local occupancy grid is built and maintained automatically in real-time. The occupancy grid is filled with probabilities of containing an obstacle based on visual observations (*i.e.* high density clusters of tracked features). Each cell in the occupancy grid is decayed over each time step which temporally smooths out observations and thus eliminates a large portion of noise due to spurious noisy features. In addition, the occupancy grid allows the watercraft to plan a local motion trajectory over the cells of minimum obstacle probability.

Extensive field testing was conducted to evaluate the performance of the VOO system (extensive field testing proved to be an additional practical challenge that was faced to quantify the performance of this system). Multiple test scenarios were run to quantify the performance. In our test scenarios in a riverine environment, obstacles included: lily pads and riverine banks (grass and concrete). We also evaluate the performance of the local reactive VOO as the airboats execute waypoint navigation patterns over a large area of the lake. The key objective of this work is to minimize collisions and maximize the ratio of obstacle detection to false alarms (*false positives*). In other words we want to maximize *precision* and *recall*. We achieve a low collision performance of 4.90%. Occasional collisions are tolerated due to the light-weight structure, small depth profile of the watercraft (which allows it to overcome small obstacles) and the relatively low velocity. Avoidance of hazardous areas that can damage or cause the boat to get stuck is more important and with the low collision performance we achieve, these hazards are avoided.

Section 3 describes the watercraft robot platform and its visual sensing capability. The onboard real-time VOO system is described in detail in section 4. Section 5 outlines the field experiments conducted and discuss the results obtained. Finally section 6 provides a conclusion and a brief plan for future research.

2 Related Work

For a detailed description of unmanned surface vehicles or autonomous surface craft refer to [15]. Most vehicles in past work have been tele-operated by human operators or utilize navigation systems to achieve autonomous behavior. The most relevant work on surface vehicles that use visual sensing are the following.

Visual avoidance systems have been developed by the Space and Naval Warfare Systems Center [3]. These systems are single robot watercraft that utilize an arsenal of sensors making them costly and not adapted to monitoring large water expanses.

River detection and segmentation using visual cues was presented in [4]. This approach utilizes a surface vehicle that is pre-trained to visually detect water as it navigates. Novel environments are dealt with by an online learning approach. This system has not been tested on a large variety of obstacles that are exceedingly hazardous to our fleet of airboats (*e.g.* floating debris, plants, branches and other boats).

3 Watercraft Robotic Platform

Figure 3 depicts one of the watercraft vehicles in our fleet. The yellow water-tight box holds the onboard electronic components: battery, Arduino board [5], Electronic Speed Controller (ESC) and cooling fan. The box has connectors that interface with the actuated propulsion system (refer to figure 3).

The smartphone is contained in a water-tight transparent box on the bow of the boat with the monocular lens facing forward. The box is mounted on the boat such that the smartphone is stable. This prevents motion blur from affecting the captured image frames. The phone communicates with the Arduino board via bluetooth which in-turn relays motion commands to the shroud servo. The smartphone also maintains a wifi connection to transmit data to a central off-water server at which a human operator monitors and controls the fleet. The smartphone also provides GPS localization and inertial measurements (a built-in gyroscope, accelerometer and magnetometer).

The hull is constructed from light-weight foam and has a small depth profile to assist it in navigating shallow debris-full riverine environments.

Lightweight application-specific sensors are mounted onto the boat and connected to one of the ports interfacing it on-board electronics.

The airboat can be tele-operated by a human operator via a large area wi-fi network or through 3G technology. The developed GUI allows the user to tele-operate the boat and also set waypoints and navigation patterns for the boats to follow. Some examples of these navigation patterns are: the *lawnmower* and the *zig-zag* pattern which are used for monitoring large areas of a lake or river. Multiple boats can be set on autonomous waypoint patterns. Each boat transmits its GPS location to the human operator through the designed GUI. Figure 1 displays multiple airboats out on a lake measuring water temperature and oxygen levels over a large portion of it.

The airboat has a Samsung Galaxy Note mounted on top which has a native camera resolution of 8 mega-pixels (full HD 1920x1080 at a maximum frame rate of 24-30 fps). The phone has a 1.4 GHz dual core processor and a built-in memory of 16GB.

The programming of the airboat's smartphone was done on the open-source Android operating system. Computer vision processing was programmed in C++ using OpenCV and the Java Native Interface (JNI) for real-time visual processing [6], [7].

4 Optical Flow-based Visual Obstacle Avoidance

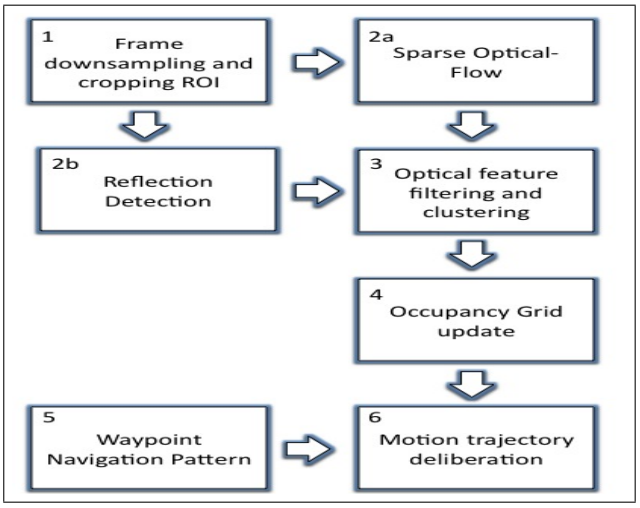

Fig. 2. The VOO system pipeline

The visual obstacle avoidance pipeline is illustrated in fig. 2. In this section we will describe each step of the pipeline.

4.1 Visual Sensing Preprocessing

Image frames captured by the smartphone consist of high-resolution images captured at a maximum rate of 30fps. Visual processing is just one of the tasks the smartphone must execute in parallel. Communication links with the central server and Arduino, over wi-fi and bluetooth, respectively, must be maintained and the necessary data transmitted through them. Inertial measurements (using an in-phone gyroscope, magnetometer, accelerometer and compass) and GPS locations are being constantly monitored in the field. These tasks slow down the

fps rate considerably (up to ≈50%) leaving little computational room for the actual visual processing and motion deliberation. For this reason, optimizing our VOO system is critical.

To achieve this, the first step in our VOO pipeline is to pre-process the incoming image frames. The captured frames are down-sampled before any visual processing takes place. We down-sample the high-resolution native images to QVGA resolution (*i.e.* 320 x 240). This of course reduces the image quality but is needed to achieve real-time processing on the smartphone. Empirically, we determine that the loss in image detail does not adversely affect the visual processing of the VOO system.

A region of interest (ROI) is defined over the down-sampled frame. The underlying assumptions here is that the VOO system should visually process the lower portion of the frame and not the upper portions which are dominated by objects that do not lie in the local vicinity of the watercraft and hence do not pose potential risk (*e.g.* sky and trees). The ROI is selected to capture a large area in front of the boat (assuming that the water surface consists of a horizontal plane). We select a ROI that coincides with a region of about 20 meters in front of the boat. This large distance is required in riverine environment due to a couple of reasons: firstly, the inertia/momentum of a relatively fast moving vehicle in water dampens its reactivity, secondly the boat has no method of reversing to avoid fast approaching obstacles.

The ROI is the final cropped region of the frame that is streamed into the optical-flow algorithm and reflection detection algorithm (see system pipeline - Fig. 2).

4.2 Optimized Optical-Flow for Visual Obstacle Avoidance

Optical flow is an algorithm for tracking pixels or point features over video (*i.e.* image frames) [2]. It is a widely used method in robotics for visual robot ego-motion estimation for indoor or ground-based vehicles. There are two main versions of this algorithm: Dense optical flow [8] and sparse optical flow [2]. The former tracks pixel-level features or dense grids of point features over video frames. The latter tracks a number of sparse strongly detected features over multiple frames. Sparse optical-flow has less computational overhead and has the added advantage of not enforcing the computation of optic flow over a set of pre-defined points. For that reason, dense optical flow is often followed by inlier motion estimation to remove outlier optical flow trajectories [13], [14].

We base our VOO system on sparse optical flow due to its lower computational footprint. We use the Lucas-Kanade Sparse Optical Flow algorithm [2]. This algorithm first detects good features to track (*e.g.* Harris corners, SIFT and ORB features [12], [11], [10]) and tracks them over an arbitrary-sized window of image frames. Feature trajectories (or flow) are hence recovered over the window of frames (Fig. 4). This sensory cue of optic flow trajectories enables the watercraft to track feature points as it moves about in the environment.

Feature trajectories of small length (relative to the number of frames being analyzed) are filtered out to recover the more robust persistent trajectories. This

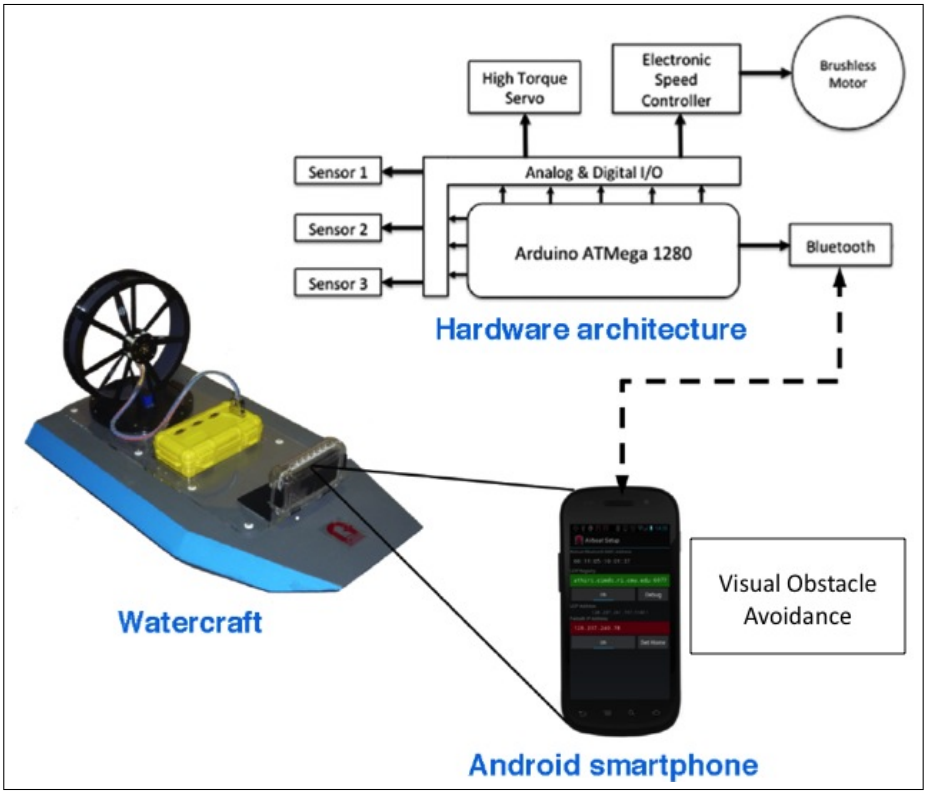

Fig. 3. The system architecture composed of: watercraft, hardware system and smartphone. The watercraft vehicle is composed of a light-weight small depth profile hull, shroud providing above-surface propulsion, e-box: containing the controller/electronics (yellow), camera box containing the mobile phone (which communicates with the e-box via bluetooth).

minimizes the affect of noisy tracked features being picked up in the environment mainly due to specular reflections appearing on the surface of the water. Tracked feature trajectories are also analyzed to remove any outliers which do not conform to the overall motion in local regions of the image frame. The ROI is divided into a 3x6 grid. Each cell of this grid is used as a local region where outliers are removed. The way this is done is by computing the average slope of feature trajectories within each cell. Any trajectories that exhibit slope larger than a set standard deviation from that slope are considered outliers and removed. In this way, a more robust set of optical flow trajectories are recovered.

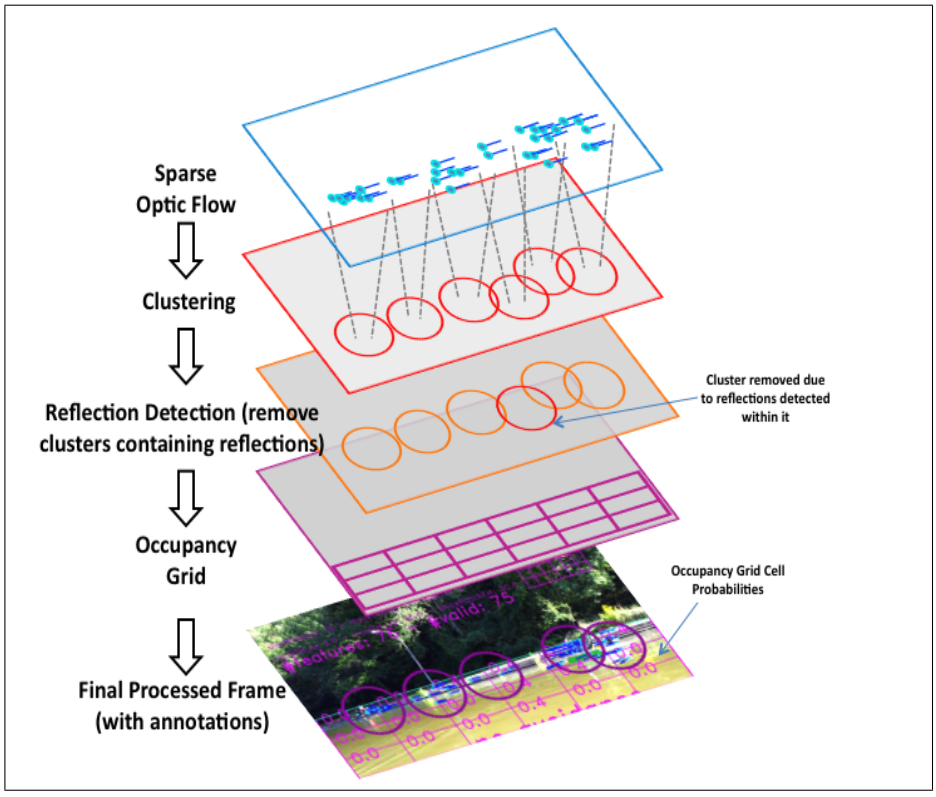

Fig. 4. An illustration of the pipeline. Features are detected and tracked over a window of frames. These features are then clustered. Clusters containing a high number of reflections are removed from the pipeline. Clusters that overlap cells of the occupancy grid increase the obstacle probability of those cells. The 3x6 occupancy grid accumulated probabilities of containing obstacles with applied decay. In this scene, another boat is caught crossing paths (from left to right) with the observing boat. By zooming in, one can see that repetitive observations of the boat causes some of the occupancy grid cells to have higher probabilities.

The last step in the visual processing part of the VOO system pipeline is the clustering step. Random spurious tracked features arise due to the relatively low-frame rate, specular reflections on the water surface and specs of algae or floating leaves. We use $k - means$ clustering [9] to cluster tracked features together and detect high density regions of trajectories. Bounded circular regions are centered at the centers of these high density clusters and any features lying outside the radius of these regions are considered outliers and hence ignored. The high density clusters are then recognized as detected obstacles to be avoided (*e.g.* bank, floating branches, large debris, protruding rocks...*etc*).

4.3 Temporal Smoothing via Local Occupancy Grid Mapping

To account for the noisy visual environment due to reflections and fast linear/angular velocities of the airboats (this is needed to fight strong dynamic currents), an occupancy grid is maintained in real-time over the ROI. Each cell represents a rectangular region in front of the boat. An occupancy grid of 3x6 cells was built (as seen in Fig. 4).

The goal of an occupancy grid is two fold: to contain and update probabilities of obstacle observations and to discretize the region of water in front of the boat. This discretization offers a basis for selecting from a discrete set of avoidance maneuver trajectories. This allows local path-planning to be performed in the form of motion trajectories over the cells of the occupancy grid. The cells of the grid are analyzed in real-time and the path with the least probability of containing obstacles is chosen as the avoidance maneuver. We select a set of 7 smooth motion trajectories which have the ability to maneuver the boat over vacant cells of the occupancy grid. The motion trajectories are shown in figure 4.

To temporally smooth the occupancy grid to again account for noisy observations in riverine environments, a decay parameter (value less than 1) is used to dampen individual spurious observations and strengthen repetitive observations. If a cluster of high density feature flow overlaps a cell of the occupancy grid then that cell's obstacle probability increases by Δ_{obs}. The decay parameter was applied by using the following update equation:

$$p_t^{ij} = p_{t-1}^{ij} * d^{\Delta_t}$$

where p_t^{ij} is the obstacle probability of cell in row i and column j at time step t. d is the decay parameter and Δ_t is the time elapsed since last observing an obstacle in this cell. This is a non-linear smoothing of the occupancy grid which enables temporal smoothing and only strengthens the probability of cells that exhibit multiple repetitive observations.

4.4 Reflection Detection

Reflections in riverine environments come in two main types. The first is specular reflections off ripples on the surface of the water. Ambient or directional sunlight

is reflected off the ripples directly into the camera lens. The second type of reflections that takes place in riverine environments are the reflections of objects in the water or on land. Any object lying on the bank/shore of a river or lake has very similar appearance to its reflection on the water surface. On a particularly calm day, especially during the early hours, reflections mirror everything on the bank (Fig 5).

Fig. 5. Mirror like appearance of water during sunny early hours of the day

Polarizer filters do not remove specular reflections or object reflections to the degree needed to avoid tracking features lying on reflections. The reason for this is that on calm water reflections are strong and their polarization varies across the field of view of the camera. The polarization of light is dependent on the sunlight or ambient light direction (or overcast illumination), the reflecting surface orientation (which is dynamic due to water ripples) and the relative angle to the camera viewing direction. For this reason an additional module was built (step 2b in figure 2) to remove optical flow trajectories that are picked up erroneously due to reflections.

Reflections are detected by dividing the captured frame into two halves (upper and lower). The assumption here was that the camera is oriented such that the horizon was located approximately across the center of the image frame (the horizon here being the shoreline far off from the boat and not the actual horizon which can be measured using onboard inertial sensors). An ORB feature detection algorithm [10] detects and extracts feature descriptors from both halves of the image and attempts to find matches between the two sets. ORB features were chosen due to their fast computation, when compared to other possible features. Brute force matching was found to be too computationally expensive. The complexity of brute forcing two sets of features is $O(n^2)$, where n is the

number of features found in each half of the image frame. A more optimized approach was developed to replace this expensive brute-force step. The approach is as follows: a predicted reflection region was estimated by projecting the lower half features into the upper half of the image (similar to a mirror reflection) and searching within a bounding box around the estimated reflection location. This proved to be much more optimized and served its purpose in removing object reflections on the surface of the water.

If more than a certain threshold of features are found to be reflections, the cluster in which they are contained is removed from the pipeline. Due to the limited number of features processed in the reflection detection algorithm, larger regions of reflections may be missed. By removing clusters of features that have a large number of reflection features within them, we ensure that larger reflection regions are removed.

4.5 Integration with Waypoint Navigation

To enable distributed monitoring of large areas of water systems, multiple robots are deployed to run in repetitive motion patterns between pre-defined waypoints. A human operator deploys multiple airboats and sets their waypoint navigation patterns (*e.g.* zig-zag pattern depicted in figure 6). These patterns are sets of GPS waypoints defining each leg of the boat's complete repetitive path. Once deployed, the boats are difficult to monitor. Each boat may face hazards along its pre-determined path. These hazards include GPS drift which may run it ashore, other boats and floating debris.

We developed an integrated system which allows the airboats to follow its waypoint navigation patterns, but overrides this motion locally in the case of when a potential obstacle is detected. This is summarized in Alg. 1:

The airboat goes into a reactive avoidance maneuver based on the motion trajectory that is least likely to be obstructed by obstacles (based on the probability of the occupancy grid cells). This maneuver is triggered for a period of time (*e.g.* order of seconds) and then the waypoint navigation takes control again. If an obstacle is re-detected k times along the path to the upcoming waypoint, then the boat overrides the upcoming waypoint and targets the next one in its navigation pattern.

This integration ensures that the boat autonomously reacts and avoids obstacles detected in its local vicinity and simultaneously follows, as close to physically possible, the given waypoint pattern navigation.

5 Experiments & Results

To validate our approach we performed numerous experiments at Panther Hollow lake near Carnegie Mellon University. Panther Hollow is a small lake with banks ranging from concrete/grass banks to banks filled with vegetation (ranging from grass, tall reeds, bushes and trees) all around. The lake has parts jutting out into

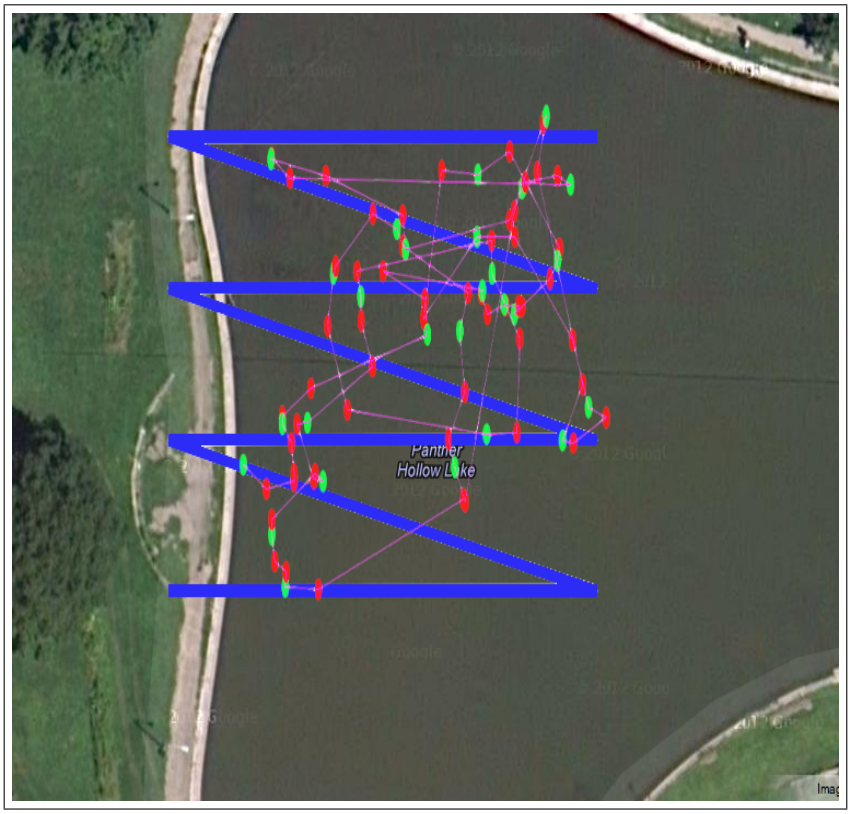

Fig. 6. Figure showing *zig-zag* pattern overlaid on top of Google maps. The point distribution shows the locations of airboat deliberations as it executed the navigation pattern. Red points signify avoidance maneuvers and green points indicate where the airboat deliberates to move on to the next waypoint after, in this example, 2 consecutive avoidance maneuvers. One example is the upper part of the annotations: the boat was moving along the upper most leg towards the leftmost upper waypoint of the *zig-zag*pattern. The boat made two obstacle detections (red points) and then continued to the next waypoint on the upper rightmost part of the pattern. Notice that the boat was able to cover more than 75% of the pre-determined region (shown by the zigzag pattern) and not collide with the bank.

Algorithm 1 Input: Waypoint Pattern $W\{w_i : i = 1, \cdots, N\}$

```
while i < N do
  while current waypoint w_i has not been reached do
    obstacles ← getObstacles()
    if obstacles is empty then
      continue heading to w
    else {obstacles detected}
      while time passed is < s seconds do
        avoidObstacles()
        avoidanceCount ← avoidanceCount + 1
      end while
    end if
    if avoidanceCount ≤ k then
      i++
      break
    end if
  end while
end while
```

Table 1. Performance of the test scenarios in % over multiple test runs

Test Runs	Avoidances (TPs)	False Alarms (FPs)	Collisions (FNs)
North Bank	67.68	32.32	0.00
West Bank	89.29	10.71	0.00
South Bank	56.24	43.75	0.00
Lily pads	68.62	11.76	19.61
Average	70.46	24.63	4.90

the water and lily-pad obstacles. Floating debris in the form of broken branches and algae are also common in this environment.

Using a user-interface (GUI) developed, we are able to set waypoints for the airboat to follow. For the purpose of our tests we set waypoints on or behind obstacles (w.r.t the airboat) to force the boat to head towards them and thus test the VOO system.

We conducted other test scenarios to test the capability of our VOO system in the context of applications that require waypoint navigation patterns. We instructed the airboats to follow *zig-zag* patterns where half the waypoints lie on the lake bank and the other half lie in the water. This forces the boat to encounter the bank as an obstacle during every other leg of its path and hence allows to quantitatively analyze the VOO behavior.

For all the obstacle avoidance experiments, we record the number of avoidances (*true positives*), false alarms (*false positives*) and collisions (*false negatives*). We use one watercraft and perform 20 runs per test. For the waypoint

navigation test (Fig. 6), the airboat covered the pre-defined area 4 times. We limit the optical flow to 300 features to limit processing time. Table 1 summarizes our results for the various test scenarios. Each bank of the lake exhibited slightly different visual features. The north bank consists of overhanging trees. The south and west bank consists of bushes and trees in the background. The lily pads were centered in a small inlet (or bay) of the lake which stood as a large obstacle preventing the boats from entering and exploring this region of the lake (see supplementary material).

We experimented under various natural illumination conditions, including: overcast, cloudy, midday sunlight, afternoon sunlight and evening sunlight. Weather conditions varied throughout our tests. Relatively high wind picked up often in the test environments. This disturbs the smooth motion of the watercraft by slowing it down in certain directions and speeding it up in others. The test scenario with the most wind is the south bank in table 1. This explains the low true positive rate and relatively high false positive rate. Wind caused ripples on the surface of the water which, along with the low frame-rate, created visual noise in the form of falsely tracked features.

The lily pad avoidance tests produced lower performance when compared to the other test scenarios. The reason for this lies in the fact that the lily pads protrude above the water's surface by a few centimeters. With the height of our mounted camera, it is hard to observe the lily pads when far off. The airboat has to get close to the lily pads in order to actually observe and detect them as obstacles. This caused it to collide more often than the bank tests.

Figure 6 shows the waypoint navigation test scenario. The boat was given a *zig-zag* pattern to follow which forced it onto the west bank on every other leg of the path. Roughly 50% of the deliberations made were correct avoidance maneuvers (*true positives*) and the other 50% were false positives picked up by noise in the form of specular reflections. There were no collisions with the bank in this test and the airboat was able to cover an estimated 75% of the physically accessible part of *zig-zag* pattern region (refer to fig. 6). The seemingly random motion of the boat is due to GPS inaccuracy, water currents and the avoidance maneuvers.

For the test scenarios conducted on the lily pads and banks of the lake, we achieve a low average percentage of collisions (4.90%). This meets the objective of this work; to use a low-cost approach to minimize collisions with obstacles that arise due to the dynamic complexity of the environment. Occasional collisions are tolerated due to the light-weight structure, small depth profile of the watercraft (which allows it to overcome small obstacles) and the relatively low velocity. Avoidance of hazardous areas that can damage or cause the boat to get stuck is more critical and with the low collision performance we achieve, these hazards are avoided considerably. Correctly detected obstacles are found 70.46% of the time and false alarms are found 24.63% of the time. The false alarm rate is acceptable given the fact that collisions are minimized considerably.

6 Conclusion and Future Work

In this work, we present a visual obstacle avoidance system adapted to low-cost and low computationally powerful airboat robots in challenging riverine environments. The visual obstacle avoidance system has been shown to be robust to many obstacles common in riverine environments.

For future directions we plan on extending the VOO to dynamic obstacles and integrating obstacle information into a global map shared by the fleet. In this way, vehicles can share geographical information about obstacles in the environment and collaborate for the greater good of the fleet.

References

1. A. Valada, P. Velagapudi, B. Kannan, C. Tomaszewski, G. Kantor and P. Scerri, Development of Low Cost Multi-Robot Autonomous Marine Surface Platforms. International Conference on Field and Service Robotics (FSR) 2012.
2. B. D. Lucas and T. Kanade, An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, 1981.
3. J. Larson, M. Bruch and J. Ebken, Autonomous Navigation and Obstacle Avoidance for Unmanned Surface Vehicles. Space and Naval Warfare Systems Center, San Diego. SPIE Unmanned Systems Technology VIII 2006.
4. S. Achar, B. Sankaran, S. T. Nuske, S. Scherer and S. Singh, Self-Supervised Segmentation of River Scenes. ICRA 2011.
5. Arduino: An Open-Source Electronics Prototyping Platform, www.arduino.cc.
6. OpenCV: Open Source Computer Vision. http://opencv.org.
7. Java Native Interface, www.oracle.com.
8. H. Liu, R. Chellappa and A. Rosenfeld, Accurate dense optical flow estimation using adaptive structure tensors and a parametric model. IEEE Transactions on Image Processing, 2003.
9. T. Kanungo, D. M. Mount, N. S. Netanyahu, An efficient k-means clustering algorithm: Analysis and implementation. Pattern Analysis and Machine Intelligence, 2002.
10. E. Rublee, V. Rabaud, K. Konolige and G. Bradski, ORB: An Efficient Alternative to SIFT or SURF. International Conference on Computer Vision (ICCV), 2011.
11. D. Lowe, Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision, 60, 2 (2004), pp. 91-110.
12. C. Harris and M. Stephens, A Combined Corner and Edge Detector. Proceedings of the 4th Alvey Vision Conference, pp. 147-151, 1988.
13. P. Santana, L. Correia, Improving Visual Odometry by Removing Outliers in Optic Flow. In Proceedings of the 8th Conference on AUtonomous RObot Systems and Competitions, 2008.
14. D. Sun, S. Roth, M. J. Black, Secrets of Optical Flow Estimation and Their Principles. Computer Vision and Pattern Recognition (CVPR) 2010.
15. J. E. Manley, Unmanned Surface Vehicles, 15 Years of Development. Battelle Applied Coastal and Environmental Services, OCEANS 2008.