Conference Paper

# Detection of Large Bodies of Water for Heterogeneous Swarm Applications[1]

**Benjamin Champion[1]*, Patrick Benavidez[2], Mo Jamshidi[3], and Matthew Joordens[2]**

[1]Work was supported, in part, by grant number FA8750-15-2-0116 from Air Force Research Laboratory and OSD through NCA&T State University
[2]School of Engineering, Deakin University, Waurn Ponds, VIC, Australia
[3]Electrical and Computer Engineering Department, College of Engineering, University of Texas at San Antonio, TX, USA

## Abstract

Multiple robot systems are becoming popular, as introducing more robots into a system generally means that the system is able to finish a task quickly, as well as making the system more robust. Generally, these systems are homogenous in nature as they are easier to build, test and conceptualise. More applications of these types of systems in a heterogeneous sense is becoming a must, as these robots are acting in more than one medium such as on land and underwater. In this paper a subsystem of a heterogeneous swarm is investigated where a land based robot is to drive up to the edge of a pool and stop autonomously, allowing for the transfer of an object from an underwater robot. To detect the edge of a pool an Xbox Kinect sensor is used as it was found that by using the IR feed of the camera the problem becomes significantly simpler.

**Keywords:** Underwater robotics, single camera depth, underwater object retrieval, water detection camera

Corresponding Author:
Benjamin Champion; email:
benjamin.champion@deakin
.edu.au

Received: 28 November 2016
Accepted: 4 December 2016
Published: 9 February 2017

Publishing services provided by Knowledge E



Selection and Peer-review under the responsibility of the DesTech Conference Committee.

OPEN ACCESS

## 1 Introduction

Cooperation between different robots has become a widely studied area, with many different tasks being able to be accomplished faster and with more robustness when multiple robots are used, as opposed to the standard single robot approach. Traditionally, these robots are homogenous in nature, meaning that they are of the same type. In recent years there has been a need for heterogeneous robotic swarms to be developed to be able to solve much more complicated tasks in a more diverse environment. This paper investigates a subsection of one of these systems, this being a heterogeneous swarm with underwater and land based robots working together for object retrieval. This paper focusses on how the land based robot is able to find the edge of the water, in this case a pool, to allow for the object transfer between the two systems to happen.

The proposed solution utilises the IR image from a Kinect sensor to detect the edge of the pool, as it was found that the IR light was not reflected well from the water

when pointed at a slight angle to the surface of the water. It is also known that when the Kinect is more than 50cm away from the water, it is unable to detect objects below the water line due to the high water absorption of IR light [1]. Previous work has been completed to try and detect large bodies of water in autonomous vehicles, such as using sky reflections for long range water detection [2] and reflection intensity for short range detection [3]. Methods such as measuring entropy of the water [4] and looking for predictable effects that a robot can have on the water[5] have also been explored.

## 2 Image processing

Initially the RGB image from an Xbox Kinect was trailed to find the water's edge. It was determined that this was not an easy problem though. Simple methods for detecting objects in the RGB space could not be relied upon, such as colour segmentation, as the colour and the shape of the water could vary significantly. It was then thought that because water reflects light and becomes "shimmery" that this could be a property to look for. Unfortunately, water does not always display this ability, particularly in low light conditions and therefore it cannot be relied upon. In the Kinect IR image stream (grayscale), it was found that regions with water appeared black (no IR light detected) in the image as IR light is either absorbed or scattered. As this seemed to work consistently in all tested lighting conditions, it was determined that this would be the best method to find large bodies of water.

The next step was to figure out how to identify the edge of the water autonomously. The first thing that needed to be decided upon was the package that would be used to process the images. The Robot Operating System (ROS) is used for the robot middleware, which provides support for both C++ and Python programing languages. It was decided that the de-facto standard Open Computer Vision Library (OpenCV) would be used as it had the capabilities that we needed and was compatible with our robot middleware. OpenCV has support for C++, Python and an expansive set of tutorials. OpenCV is also compatible with ROS and image streams can be directly interfaced with ROS applications such as rviz via a ROS API called cv_bridge.

When the IR image was inspected, it was found that the occasional reflection from the IR source was received, causing the water to "sparkle" a little bit in this view. Due to this observation, initially a difference image was generated to separate the water from the other obstacles. This involved taking the previous IR image received from the camera and finding the differences between this and the current image. Assuming that nothing else was moving in the image and the camera was pointed at the ground, the only regions remaining in the image would be the body of water. The biggest drawback with this method was that if the robot was moving, any other objects that were on the

ground would also appear in the difference image making it very difficult to separate the water from the other objects. Therefore, another method had to be devised.

Solving the water detection problem turned out to be relatively simple and could be solved using mainly OpenCV functions. First, a dilation was applied to the IR image, as the IR image from a Kinect contains a pattern of IR dots scattered from a diffraction grating installed in front of the IR laser. To find the solid bodies and the voids in the image, these dots need to be fused together which is being accomplished by the dilate command. The standard OpenCV dilate method was used, with the structuring element size set to 40×40.

If the dilate function is solely used, then the edge of the water within the image would actually be pushed back out into the real water, meaning that the robot would think that the water's edge is father away from the robot than what it really is. To overcome this issue an erode of the image, using the same size structuring element (40×40), was used to put the edge of the water back into its original position as much as possible. As the erode method only effects the edges of the already processed image, the dots do not re-appear as they have already been fused together into a single blob.

A binary threshold of the image was then taken, with the threshold value set at 115 on an 8-bit scale. This returns a true black and white image so further processing becomes much easier and simpler. It was also found that at this threshold level, at both the lab where the algorithm was initially developed and the pool where it was tested, most of the other noise form things such as the floor, objects on the floor, people's shoes, etc. were eliminated. It also left the image with a clearly defined edge of where the water was, as the water appeared black and the rest of the environment was white in the image.

After the thresholding, the image was eroded and dilated again to get rid of any noise that might still present in the image, particularly if any of the "sparkles" mentioned previously managed to get through the previous filtering. Again the structuring element for both the erode and the dilate were the same (30×30) to ensure that a minimal amount of effect was placed on where the edge of the water was in relation to the robot.

Two more processes were performed to the image to be able to extract the edge of the water from the image. Firstly, a Canny filtering of the image was taken to ensure that only the edges of the regions remain. Once this has been accomplished, the contours of the image can then be extracted using the *findContours* function. This step essentially found the closed black areas of the image. From this data, the largest contour region is determined which we can assume to be the water in most cases. To ensure that the water is in fact in view, the area of the contour is checked to ensure that it is contains more than 15000 square pixels. This again is to ensure that no noise is picked up and deemed to be the water's edge. This value was selected as it was found that this was larger than any of the areas that had not already been filtered out, but still large enough that the water is not eliminated when it significantly enters the frame.

# 3 Distance to the edge of the water

## 3.1 Extracting the Point

Once the water has been found, the edge of the water needs to be extracted. Again, this is not a complicated process. Firstly, a new image is created that only contains the edges of the contour detected and chosen using the process described in Section 2. To determine a point that the robot needs to reach, the pixel closest to the bottom of the image is selected. This is accomplished using the *findNonZero* function, which returns a list of pixel coordinates that are not black in the referenced image, which in this case should only be the pixels of the edge of the selected contour. Using this list of pixels, the closest one can be determined by simply generating a vector from the bottom of the screen to each pixel, and then sorting by the magnitude of the vectors to find the smallest one which will correspond to the closest point on the water's edge.

## 3.2 Distance to the Point

Extracting the actual distance to this point is a significantly more challenging problem. When a Kinect is used, a point cloud is generated that can be used to find the depth to the points associated with the RGB image. Initially, when it was thought that the RGB image would be the best way to find the water's edge, it was thought that using the information from this already processed point cloud would be the best way to find the edge of the water as distance information was already provided. There are some very big drawbacks with this process though.

As described earlier, the water makes the IR image sparkle. As the depth for the point cloud is based off the intensity of the IR image, the sparkling that was observed with the IR image became a significantly large problem resulting in a point cloud with significant errors. This sparkling was present, and seemed to be amplified within the point cloud. Therefore, the depth values from this point cloud around the water were to inaccurate to be of any use in their current form.

Once it was determined that the IR image would be used to find the edge of the water, not the RGB image, it was then thought that by filtering out the random values and taking an average of the reaming surrounding depth points, the point cloud could still be used to get a very good idea of where the edge of the water would be. Unfortunately, when this was tested it was found that using the IR image meant that the point cloud could not be accessed (due to a hardware limitations), therefore making this method much more difficult. To overcome this, the *image_proc* library in ROS was used to try and re-create the same point cloud while still being able to access the IR image. It was found that even with this point cloud, there was still too much noise, with the points being separated by too far a margin for it to be a viable method.

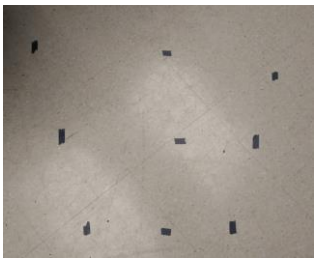

**Figure** 1: KINECT VIEW.

To overcome all of these obstacles, the method that was finally relied upon was to map out field of view of the camera and then calculate the point where the pixel should be in the real world based off these values. This method assumes that the environment that the robot will be operating on is flat, and that the point that you are detecting will be on this flat plane not on a raised surface or an incline. As we have already declared that the environment will be free of large objects that are close enough to the camera that they will show up as voids, and therefore the robot could possible mistake them as the water source, and water will always level out, it is safe to make these assumptions for this environment. It is also very computationally inexpensive compared to other methods of finding depth with a single camera [6–10] and an unknown reference point which is what the problem has turned into as the methods used to generate a point cloud unfortunately can no longer be relied upon.

The first issue that needs be overcome is that the field of view of the Kinect IR stream is not a square, but instead a skewed rectangle that narrows from top to bottom like the letter V, as depicted in Figure 1. Marking out the field of view like this also allows for the measurements of the X and Y widths relatively accurately. For the y-coordinate measurements the V shape is not an issue, as they can be treated the same as a normal camera as there is no dilation in this axis. Therefore, as the camera was set to a fixed distance and angle due to it being permanently mounted to the robot, the height of the y frame can be measured which in this case was 54.5cm. Since the IR frame is 480 pixels high, we can then say that the distance in the y direction that the point is away from the robot (in metres) will be:

$$y = \frac{H_m}{H_p} y_{pixelCoord} + y_0$$

Where $H_m$is the height in metres of the viewing area and $H_p$ height of the viewing area in pixels. $y_0$ is the distance, in metres, that the bottom of the frame is away from the robot, which in in this case was 28cm. It needs to be noted that the Kinect camera was positioned upside down for this application due to mounting constraints. If this was not the case, to make the distance measured from the bottom of the frame, the y-coordinate would need to be taken from the height of the camera frame in pixels. For the x distance, this proves to be a more challenging problem as this is where the dilation is. It was noticed that the dilation was very linear, and therefore an equation to directly

**Figure** 2:

map the x-coordinate width to the y-coordinate could be used. To find this relationship three measurements were taken: one at the bottom (y = 28cm) (x = 43cm), one in the middle (y = 54.2cm) (x = 58.6cm) and one at the top (y = 82.5cm) (x = 76.5cm) and the individual x ratio for these points found. Therefore, by using a simple linear relationship the $x_{ratio}$ can be found at any corresponding y coordinate, which for this case turn into:

$$x_{ratio} = (9.891 \times 10^{-4}) \times (y - y_0) + (6.563 \times 10^{-4})$$

Once this is known, then the distance in the x direction is found by using:

$$x = \left(\frac{W_p}{2} - x_{coord}\right) x_{ratio}$$

Where $W_p$ is the width of the image in pixels and is done to ensure that the x distance is from the centre of the frame, not the far edge, and $x_{coord}$ is the coordinate of the desired point in pixels. Finally, this point can then be inserted into the global frame by rotating the points about the yaw of the robot and then adding these points to the robot's current x and y position. This will allow the robot to remember where the detected water is, and therefore be able to drive back to it much easier.

## 4 Results and Future Work

It can be seen in Figure 2 and Figure 3 that the edge of the water was able to be detected fairly accurately. The images in these figures shows where the robot was in relation to the water as well as the data the robot is currently processing. Both figures show the raw IR image on the left and the processed image on the right, with the green dot on the processed image indicating where the closest point that the robot needs to drive to is located. The robot was tested and found to be able to drive up the edge of the pool and stop, without falling into the water, consistently. In Figure 3, the image on the far right also shows that the filtering of the image is also successful, as the noise and detected contours, these being the red lines, are successfully removed and only the water is left when determining where to send the robot, this being indicated by the green dot.

Some improvements could be made to this system to make it more versatile. As can be seen in Figure 2 intense reflections off the water, in this case a skylight at the pool on a very sunny day in Texas, can cause the robot to think that this area is not a water

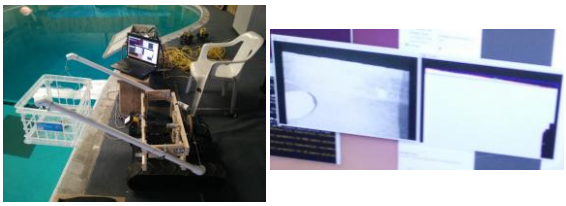

**Figure** 3:

but an object or the ground. This is most likely caused by the IR light being reflected from the sun, as it is of such a high intensity that it can be picked up by the Kinect sensor. To overcome this issue, incorporating the RGB camera into the system could prove beneficial. The RGB camera would be able to detect these really high intensity objects, and remove them from the corresponding position in the IR image. Another issue with this system is that if an object gets too close to the IR sensor, it will black out part of the image and therefor show up as a black region. This again could be solved by incorporating the RGB camera as it could view all of the regions of the IR image that are black and decide whether they are actually some other object based off other attributes, such as their shape, colour, intensity, etc.

[1]Work was supported, in part, by grant number FA8750-15-2-0116 from Air Force Research Laboratory and OSD through NCA&T State University

## References

[1] A. Dancu, M. Fourgeaud, Z. Franjcic, and R. Avetisyan, Underwater reconstruction using depth sensors, in SIGGRAPH Asia 2014 Technical Briefs, 2014, p. 2.

[2] A. L. Rankin, L. H. Matthies, and P. Bellutta, Daytime water detection based on sky reflections, in Robotics and Automation (ICRA), 2011 IEEE International Conference on, 2011, pp. 5329–5336.

[3] A. Rankin, and L. Matthies, Daytime water detection based on color variation, in Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on, 2010, pp. 215–221, (2010).

[4] P. Santana, R. Mendon, and J. Barata, Water detection with segmentation guided dynamic texture recognition, in Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on, 2012, pp. 1836–1841.

[5] R. Pombeiro, R. Mendon, P. Rodrigues, F. Marques, A. Louren, and E. Pinto, *et al.*, Water detection from downwash-induced optical flow for a multirotor UAV, in OCEANS 2015 - MTS/IEEE Washington, 2015, pp. 1–6.

[6] J. Song, S. Na, K. Hong-Gab, H. Kim, and L. Chun-shin, A depth measurement system associated with a mono-camera and a rotating mirror, in Pacific-Rim Conference on Multimedia, 2002, pp. 1145–1152.

[7] J. Michels, A. Saxena, and A. Y. Ng, High speed obstacle avoidance using monocular vision and reinforcement learning, in Proceedings of the 22nd international conference on Machine learning, 2005, pp. 593–600.

[8] T. Nagai, T. Naruse, M. Ikehara, and A. Kurematsu, Hmm-based surface reconstruction from single images, in Image Processing. 2002. Proceedings. 2002 International Conference on, 2002, pp. II-561-II-564 vol. 2.

[9] D. An, A. Woodward, P. Delmas, G. Gimelfarb, and J. Morris, Comparison of active structure lighting mono and stereo camera systems: Application to 3d face acquisition, in 2006 Seventh Mexican International Conference on Computer Science, 2006, pp. 135–141.

[10] G. C. Gini, and A. Marchi, Indoor robot navigation with single camera vision, in PRIS, 2002, pp. 67–76.