Real-time path planning and Obstacle Detection for autonomous underwater vehicle

Department of Electrical Engineering, Ben-Gurion University

Raz Malka; Itzik Mizrachy

We are part of a fourth-year engineering student team, that are participating in an international robotics competition from the AUVSI. Our team, Vision, is responsible for writing the image processing algorithms in order to perform various tasks in real-time. In this article we will introduce the three first tasks: Path detection, color balloons detection, and gate detection. We will also introduce the test app that we wrote in order to manage videos and test our algorithms on various pictures.

Keywords- image processing; real time robotic competition; obstacle detection

1. INTRODUCTION AND PROBLEM DESCRIPTION

The autonomous vehicle would have a standalone Linux computer machine with a front camera and an underground camera. The underground camera is responsible for path detection in order to reach the tasks while the algorithms missions are handled by the front camera. The path in the water is basically an orange colored rectangular bench our main Goal in this case was to search for a Square (and not only rectangle!) whose most of it's area is orange. The balloons detection main goals were to detect the three different colors and check if they are circles by using Hough circles and doing some morphological operations. The gate main goal was to detect 3 vertical green lines that together define a unique square, and return it's center as an output. We have done it by using Suzuki algorithm and several morphological operations.

All the tasks in the front camera are running together so we also had to ensure that we don't get false detection, for example: false green balloon detection which is actually a small part of the gate. All algorithms our running on Linux machine and were implemented on C++ with OpenCV libraries.

2. OUR SOLUTION

Due to the independency of our goals we divided our problem to 3 different tasks which are, as well, independence one by another and all three together unite is our solution.

Gate algorithm – in this algorithm, the main subject is to define the green colored gate from the background. one can see this problem is not an easy task due to changes in lightness and in background pattern which is already a combination of blue and green. We will proceed by noting our main steps taken on each frame followed by an reasoned explanation.

Main steps: 1.Delete red channel to obtain GB frame . 2. Convert GB to HSV and taking only Hue channel (gray image). 3. Re-scaling the hue channel, i.e. for each pixel p:

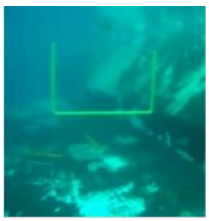
$$p' = \frac{p - min_{p \in I}}{max_{p \in I} - min_{p \in I}} \quad (1)$$

4. Threshold the re-scaled gray image by mean criteria to obtain BW image. 5. Fill holes and filter small noises by morphologic operations (dilate and erode, aka "closing"). 6. distinguish between objects in BW by finding each object's contour. 7. For each object, calculate the area of the object found in BW and find the minimal area rectangle to surround it. 8. Find the minimal ratio by dividing object area by rectangle area and check if the ratio is small enough. Yes – return the center point of the rectangle surrounding minimal ratio object. No- returns anything.

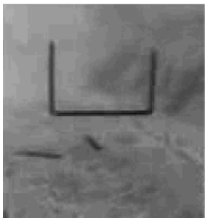
Description- as noted earlier, the stumbling-block here is defining the green gate from the noisy image (background and foreground). To do so, we created a criterion which is robust for light and background changes. By deleting the red channel we obtain a more homogeneous background (the red intensity appears mainly on objects balloon, path, etc.). The hue channel in HSV space is much more robust to light change but there will always be a difference in pixel intensity so for threshold we made the linear transformation (1). Then we check the mean of the re-scaled image to obtain the wanted threshold. after finding each object contour we filter small objects by a minimal area criteria under the assumption that the gate is not so small, and the minimal area rectangle as well. In the last part of validation we take advantage of two basic assumptions: 1. the desired object is a gate, so the rectangle area will be much bigger than the gate's area and this is a very strong tool for false detection. 2. The gate base is longer than it's beam so one can estimate the rectangle's orientation and angles with the axis meaning it is possible to know if there is a full, partial or false detection.

Example1 - Gate detection:

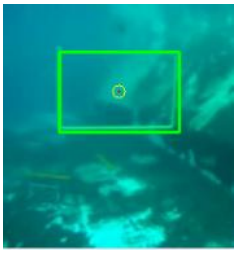
1.1-origin 1.2-hue filter




1.3-scaled hue 1.4-binary image




1.5-detection

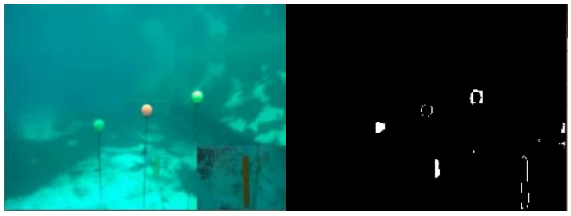


Balloon algorithm - in order to separate the colors we took the hue channel of the image which is more durable in different light conditions, empirically we have chosen three different ranges of hue value to get three different binary images for each color. In order to eliminate the noise in the binary images and to get a rounder shape we did some morphological operations. First we did closing (Dilation followed by erosion) with a large elliptical structuring element. Afterwards, we perform an open with a smaller structuring element to eliminate some additional noise from the image. The next step was to determine which segment in our filtered image has a rounded diameter. We used Hough circles transform and chose the parameters (e.g. minimum distance between detected centers, Upper threshold for the internal Canny edge detector, Maximum radius to be detected) according to the resolution of the

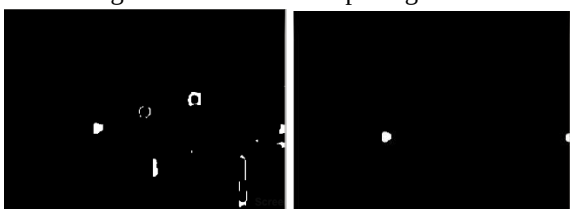
image and the minimum/maximum visible ball. The last step was to take into account only the detected circle with the longest radius. The assumption in this method is that we don't have colorful (green, red or yellow) circular shapes that are bigger than the balls we are searching for.

Example2 - green ball detection:

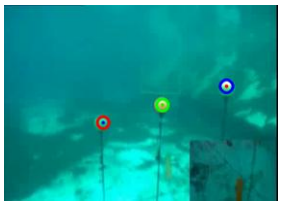
2.1-origin `` 2.2-hue filter



2.3-closing 2.4-opening



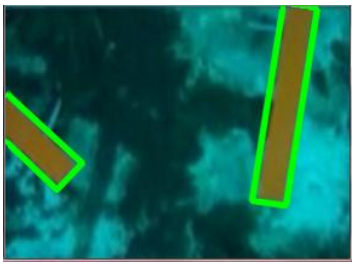
2.5-detection



Path algorithm – In this algorithm we seek for an orange rectangle pointing to the correct path to proceed. The main task is to identify the correct direction and much more important, the recognition of a partial path and its orientation. For the partial recognition we extrapolated the border with more pixels in each direction giving us a way to recognize partial rectangles.
Main steps: 1. obtaining only Hue channel as before. 2. Search for orange pixels and zero them. 3. Enlarge to image by extrapolating the boarder as described before. 4. Finding contours of object's same as in Gate algorithm. 5. For each contour we check if it is similar to a rectangle by area similarity. The object is defined as a rectangle if the similarity is 70% and more. 6. If the object is a rectangle we proceed to another color recognition – calculate the percent of orange pixels in the object by calculating the histogram of the object. The rectangle is defined as path if the orange percent is bigger than 50%. The algorithm output is a data base representing the path.

Example3 - Path detection:

3.1- Path detection



3. EXPERIMENTS AND RESULTS

In order to manage our videos/photos and to test our algorithms we wrote a test app with the following features: 1. dividing selected video into e frames 1 second interval. 2. Test one or more algorithms with a specific image or range of images. 3. Selective output data (e.g. images, vectors).

While testing the Gate algorithm we encountered an interesting problem – changing the image resolution affect on the thresholds parameters, so for now, the gate algorithm works good only on a specific video. Apart from this, the algorithm works good under the assumptions above for the whole video. False detection occurred when there was extremely noisy background but due to our mechanism false detection will never be translated as a full gate but as a partial detect, i.e. brings back only center point but not the gate shape.

The resolution problem repeated when testing the path algorithm on different resolution videos, so the thresholds is severely affect, to result with no detection. Apart of that, the detection is very good even on cut paths, which means that it is possible to detect not only whole rectangle but even sliced. If the path seen on the image is most of it sliced, we choose there will be no detection due to the fact that it will lead to more false detections.

For the ball detection algorithm the tests were satisfying. We encountered two problems which occurred only in a few frames. The first one was that when the vehicle is far from the balls it could detect wrong cycle shapes. This occurs because the radius of the required ball is smaller than the radius of the another round shape with the equivalent hue. The second problem is that when no balls are shown in the image our algorithm could detect cycles. Although these two problems are minor we are planning to avoid them by applying a dynamic range of circle detection that will take into account the expected radius according to the current number of frame.

4. CONCLUSIONS AND FUTURE IDEAS

First of all our main conclusion is that object detection underwater is not similar to grounded object detection. This lies in the fact that underwater objects gain blue color as they go deeper. As a result color filtering is not satisfying and to get better results one should use the scaled hue channel as was illustrated in formula (1). Another conclusion is that by obtaining morphological operations (e.g. closing, opening) and taking into account the orientation of the detected objects would obtain more accurate results. Our next work would be to implement our code to stereo vision. With stereo vision we can get the actual size of the detected object and get even more accurate results by taking this parameter into account. We are also working on new feature in the test app –changing threshold parameters on-the-go .

5. REFERENCES

1. AUVSI foundation, ***ROBOSUB journal papers***,2011-2012. Online http://www.auvsifoundation.org/foundation/competitions/robosub/

2. Suzuki, S. and Abe, K., Topological Structural Analysis of Digitized Binary Images by Border Following. CVGIP 30 1, pp 32-46 (1985).

3. Rami Hajaj, ***Lecture Notes in Digital Image Processing***, EE Department, Ben Gurion University In The Negev

4. Kenneth R. Castleman, ***Digital Image Processing***, Prentice Hall Englewood Cliffs, New-Jersy, 1996

5. Gary Bradski, Adrian Kaebler, ***Computer vision with the OpenCV Libary***, O'REILLY, 2008

6. Michael Lindenbaum, ***Lecture Notes in Digital Image Processing***, TheComputer-Science Department, The Technion, Israel Institute of Technology