2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Vision-based Reactive Autonomous Navigation with Obstacle Avoidance: Towards a non-invasive and cautious exploration of marine habitat

F. Geovani Rodríguez-Telles[1], Ricardo Pérez-Alcocer[1], Alejandro Maldonado-Ramírez[1], L. Abril Torres-Méndez[1], Bir Bikram Dey[2] and Edgar A. Martínez-García[3]

***Abstract*— We present a visual based approach for reactive autonomous navigation of an underwater vehicle. In particular, we are interested in the exploration and continuous monitoring of coral reefs in order to diagnose disease or physical damage. An autonomous underwater vehicle needs to decide in real time the best route while avoiding collisions with fragile marine life and structure. We have opted to use only visual information as input. We have improved the Simple Linear Iterative Cluster algorithm which, together with a simple nearest neighbor classifier, robustly segment and classify objects from water in a fast and efficient way, even in poor visibility conditions. From the resulting classification and the current robot's direction and orientation, the next possible free-collision route can be estimated. This is achieved by grouping together neighboring *water* superpixels (considered as "regions of interest"). Finally, we use a model-free robust control scheme that allows the robot to autonomously navigate through the free-collision routes obtained in the first step. The experimental results, both in simulations and in practice, show the effectiveness of the proposed navigation system.**

I. INTRODUCTION

In underwater environments, autonomous navigation is the key to a success for a great variety of applications, such as environmental monitoring, oceanographic mapping, and infrastructure inspections in deep sea. Considering the challenge that these highly dynamic and unstructured environments represent, a great amount of research in underwater robotics has been focused on the path planning, tracking and obstacle avoidance problems. Although different underwater sensing technologies has been used to solve these problems, visual sensors, in particular, have not been the first choice due to their limited detection ranges and poor visibility. Vision systems for aquatic robots [1], [2], [3] must cope with color distortions, dynamic lighting conditions and turbidity. All these distortions cause poor visibility and hinder computer vision tasks. In a reactive navigation task, achieving obstacle avoidance in real time is a high priority. Having visual data for close range object detection can be very useful. However, in underwater environments, due to existing stochastic disturbances (e.g. currents) which alter the estimation of the exact dynamic parameters of the vehicle, it is crucial to detect objects that are not very close so the robot can plan for the next motion. Thus, the object detection, free path estimation and control schemes must be robust enough in order to have a reliable navigation system.

[1]These authors are with the Robotics and Advanced Manufacturing Group, CINVESTAV Campus Saltillo, Ramos Arizpe, Coahuila, MEXICO
[2]Bir Bikram Dey is with SubC Control Inc., Clarenville, NL, CANADA
[3]Edgar A. Martínez-García is with Lab. de Robótica, Inst. of Eng. and Tech., Universidad Autónoma de Cd. Juárez, Juarez, Chihuahua, MEXICO

In this paper, an autonomous reactive navigation approach based on visual information for an underwater vehicle is presented. Our method is divided in two stages. In the first stage, we use our improved version of the Simple Linear Iterative Clustering (SLIC) superpixel algorithm [4] to robustly segment low-resolution images based on their color features. We exploit the fact that high-resolution information is not needed for navigation (see [5], [6] for studies on the subject) and also that for underwater images, the CIELab color space can be adjusted to highlight only the color on objects that do not represent water itself. Given that this algorithm has a linear computational complexity, the objects and RoI (water) can be segmented in a fast and efficient way. After this, a simple nearest neighbor classifier is applied to separate and detect objects from water. In the second stage, from the resulted classification and the current direction and orientation of the robot, the next possible free-collision route (also called direction of escape) can be estimated. This is achieved by obtaining the geometric center of the area covered by a RoI. We have previously used this approach for indoor mobile robot navigation with promising results [7]. Finally, we present a model-free robust control scheme that allows the robot to autonomously navigate by mapping the image error (in terms of the geometric center of RoI) to robot motion directly. The experimental results show the effectiveness of the proposed navigation system.

II. RELATED WORK

Autonomous underwater navigation is an open research problem with significant applications in the field of robotics and oceanic engineering. In the following, we present some of the building blocks of our work.

A. Vision and Non-vision based Autonomous Navigation

There are few research work in the area of autonomous navigation of underwater robots using visual information as the main method for navigation. The reason for this is due to poor visibility conditions in underwater environments. Several factors include the lack of natural ambient light (even in the cleanest water); the frequency-dependent scattering and absorption, both between the camera and the environment, and also between the light source (the sun) and the environment. The quality of the water determines its filtering properties. The greater the dissolved and suspended matter, the greener (or browner) the water becomes. Since many, if not all, of the above factors are constantly changing, we cannot really know all the effects of water. The result is an

image that appears bluish, blurry and out of focus. Recent work include that of Girdhar *et al.* [8], in which a navigation for exploring a coral reef is accomplished by using a real time online topic-modeling based technique. Their approach semantically models the scene and estimates a surprise score, which is used to control the speed of exploration given a fixed trajectory. Kim and Eustice [9] reported a monocular visual SLAM algorithm for ship hull inspection by using local and global visual saliency measures. Although the results of above methods look promising, they do not include the obstacle avoidance problem.

Some other recent works perform autonomous navigation of AUV by using acoustic sensors (sonar) either alone or combined with other sensors (cameras, IMU, pressure sensors). In [10] the main source of information to navigate is a sonar and in [11] the acoustic sensor data is combined with information from other sensors. Only in [11] there is an obstacle avoidance algorithm and it is important to mention that the proposed algorithm must be trained by using the AUV in remotely operated vehicle mode.

It is worth noting that the approaches mentioned are executed in real-time, but not all of them evade obstacles, and those that do it, use acoustic sensors to avoid collision. However, the use of acoustic sensors to detect obstacles underwater it is not always possible or in some cases it is not preferable in some environments (e.g., in coral reefs) due to their invasive properties.

B. Object segmentation in unstructured environments

To detect or segment the area where an underwater robot can navigate (RoI), we need to establish which features are to be searched. These features must indicate when an object is present (e.g. marine life, rocks, sand, man-made structures, etc.) and when there is only water. In recent years, the use of superpixels for image segmentation has become very popular. Superpixels algorithms group pixels with similar characteristics, for example according to their color (in one or more color models) and their position in the image. The goal of grouping these pixels is to reduce image complexity to facilitate post-processing as the operations are carry on the superpixels instead of the total number of image pixels. There are several approaches to generating superpixels [12], [13], [14], [15]. The more recent one, the SLIC superpixels algorithm [4], presents clear advantages in terms of speed, ability to adhere to image boundaries, and impact on segmentation performance.

III. METHODOLOGY

In this section, we describe our vision-based reactive autonomous navigation approach. Figure (1) shows the general diagram. It is important to highlight that the type of navigation is reactive in the sense that the robot does not have a defined goal or destination, i.e. the robot will navigate the environment by orienting its body to the regions of interest, therefore implicitly avoiding the collision with objects and marine life.

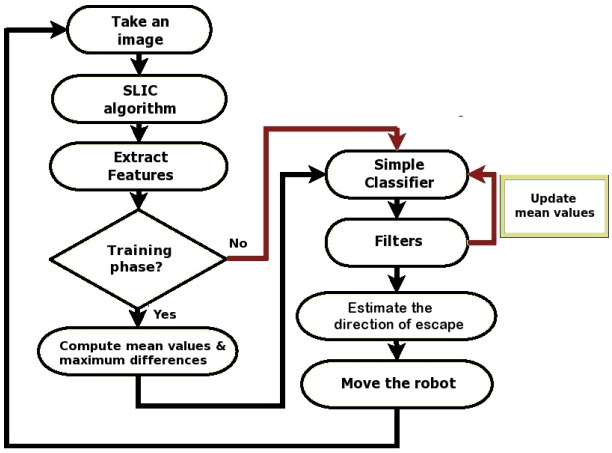


Fig. 1. General scheme navigation for underwater robot.

In the following we describe each of the steps in our framework.

A. The improved SLIC superpixels segmentation algorithm

The SLIC superpixel algorithm starts from a regular grid of centers or segments, and grow the superpixels by clustering pixels around the centers. At each iteration, the centers are updated, and the superpixels are grown again. The original method is efficient but does not run in real-time. We solve this problem by subsampling the image to a small resolution (320×240) [7]. A parameter to define is the number of the desired superpixels K in the image. The approximate size of each superpixel is N/K, where N is the number of pixels in the image. For superpixels having a similar number of pixels, a superpixel center C_k is set at intervals $S = \sqrt{N/K}$. Each superpixel center is a $5-$tuple defined by $C_k = [L_k, a_k, b_k, x_k, y_k]$, with $k = [1..K]$ at regular grid intervals S. L_k, a_k and b_k are the corresponding channels in the CIELab color space, and x_k, y_k the pixel coordinates.

We have improved the SLIC algorithm in two aspects. First, we fixed the number of iterations. Originally, the algorithm stops when the residual error between the current and previous centers of all superpixels does not change. We analyzed the convergence of residual error in several underwater images and found that after 5 iterations, the residual error does not change significantly. Second, we reinforce the connectivity between pixels to build up better superpixels. This is done in the penultimate iteration before calculating the new centers of superpixels. The connectivity step has two functions: the first function reassigns labels to pixels that are spatially separated from pixels that were grouped in the same superpixel. This is done according to the number of neighboring pixels that belong to the same group or superpixel. And the second function assigns a superpixel to those pixels that do not belong to any superpixel. This may happen (although rarely) when the new center of superpixels are updated and there may exist pixels far (up to a distance $2S$) from all the superpixel centers. The neighboring pixels

of the unassigned pixel are analyzed and the superpixel with more neighboring pixels is assigned to it. The improved SLIC algorithm is summarized in Algorithm 1.

Algorithm 1 The improved SLIC superpixel segmentation.

Initialize $C_k = [L_k, a_k, b_k, x_k, y_k]^T$.
Set label $l(i) = -\infty$ for each pixel i.
Set distance $d(i) = \infty$ for each pixel i.
$iterations = 1$
while $iterations \leq iterations_{allowed}$ **do**
  for each center pixel coordinate in the superpixel C_k **do**
    for each pixel i in a $2S \times 2S$ region close to C_k **do**
      Calculate distance D_s between C_k and i.
      if $D_s < d(i)$ **then**
        $d(i) = D_s$
        $l(i) = k$
      end if
    end for
  end for
  if $iterations$ == $iterations_{allowed}$ **then**
    Reinforce connectivity
  end if
  Calculate the new center of superpixels and their area
  $iterations$++
end while

B. Feature extraction

In order to visually classify objects, humans consider certain features in the scene, such as color, texture, shape and position. To achieve this with a computer vision algorithm, a good selection of invariant features must be done. In our case, the robot needs to classify which superpixels belongs to water and which to objects. The features we consider from the superpixels segmented image are: color (in the CIELab color space, but considering only the a and b channels), texture, position (center's pixel coordinates) and shape. For the latter, we have observed that shapes of superpixels near object boundaries are not regular, we can exploit this characteristic in our segmentation algorithm. We start by defining a dependent variable which value will indicate the probability that a superpixel belongs to a RoI. All the above features are grouped together in a vector to form a descriptor. Figure 2 shows the features we consider in our classification.

C. Training phase

To obtain a correct detection of RoIs, our algorithm first needs to learn which superpixels correspond to a RoI. We have done this manually (when testing our algorithm offline) and automatically (in online testing). In the first case, the user can indicate them in the image (by mouse clicking). For the case of online testing, this is done by pointing the robot's front camera, for few seconds, to a location where it can see only water. Thus, all superpixels in the segmented images may be considered as RoI and be used for training. Algorithm 2 shows the learning process. After selecting the n superpixels corresponding the RoI, we compute the mean values of each feature in the descriptor and also their standard deviation, to finally keep the maximum standard deviation for each feature.

Algorithm 2 Training phase.

for $i = 1$ to n **do**
  $area(i)$ = number of pixels of superpixel i
  $color(i,:)$ = channels a_i, b_i of CIELab color space
  $width(i), height(i)$ = (width, height) of superpixel i
  $d^1(i), d^2(i)$ = diagonal distance (d^1, d^2) of superpixel i
end for
$Data_2 = [area, color, width, height, d^1, d^2]$
Obtain the mean values of each array element in $Data_2$
for $i = 1$ to n **do**
  $Data_{2SD}$ = standard deviation of each feature in $Data_2$
end for
Initialize to zero Max_{dif}
for $i = 1$ to n **do**
  for $j = 1$ to 7 **do**
    if $Data_{2SD}(i,j) > Max_{dif}(j)$ **then**
      $Max_{dif}(j) = Data_{2SD}(i,j)$
    end if
  end for
end for

D. Classification of the RoI

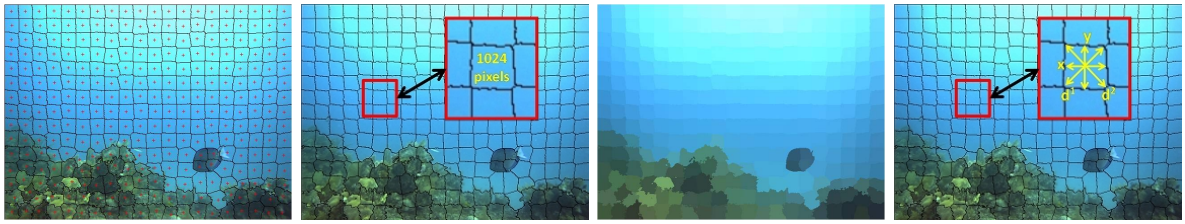
From the information obtained in the training phase we can classify if a superpixel belongs to a RoI. First, by using the mean values, we obtain the standard deviation of a superpixel descriptor to be classified. Second, we use a normalized SSD measure and, if the result is less than a given threshold then the superpixel belongs to a RoI. Otherwise it is considered an obstacle. To normalize the SSD measure we use the maximum values of the differences (Max_{dif}) from the training phase.

We have analyzed the effect of using the CIELab color space in our classification. We note that false changes in intensity caused by specularities are kept mostly by the luminance (L) channel. These specularities are present due to illumination conditions (sunlight above) and properties of the underwater environment itself. However, the a and b channels basically ignore these specularities as they keep only chromaticity changes.

We also apply the well-known erosion and dilation filters to improve the classification of superpixels. We build a graph G_S where each superpixel in the image represent a node and the edges indicate the connectivity between neighboring superpixels. A superpixel S_A is a neihgbor of superpixel S_B if the distance between their centers is less than a given threshold Θ_C. The information in the graph will help the filters to easily obtain the connectivity information of neighboring superpixels as well. We apply these filters to obtain a more precise RoI, thus avoiding to have sparse and small regions of interest, which are not relevant for navigation purposes. Figure 3 shows one classification result after applying the whole process described above.

E. Estimating the direction of escape

Once the region of interest is obtained we estimate the direction of escape in which the robot should move. To do this, we need to take into account several factors. On one hand those related to the *spatial sense* of the robot, i.e., the dimensions its body occupies with respect to the free space,



(a) Position. (b) Area. (c) Central color. (d) Shape feature.

Fig. 2. Features considered for classification. (a) The resulted 300-superpixel segmented image after applying our modified SLIC superpixel algorithm. Each superpixel's center is indicated with a red dot. (b) Area feature: pixels in a superpixel. (c) Color associated to the center of superpixel is assigned to all pixels in that superpixel (for visualization only). (d) Shape feature: width (x), height (y) and diagonal distances (d^1 y d^2), respectively.

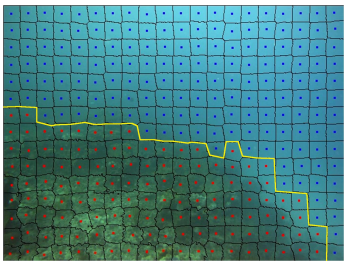


Fig. 3. An example of the classification of RoI. Blue dots (above yellow line) indicate superpixels classified as RoI whereas red dots represent objects (obstacles).

and also a notion of distance from the robot to the RoI. This helps to get a clearer perception of how much the robot has moved to a given direction/orientation, related to where it was, and associate it to what it sees next. On the other hand, since we are dealing with an unstructured environment, factors related to camera, such as size of field of view, abrupt illumination changes and distorsions on the image due to camera motion and environmental disturbance, also need be considered in our model. In general, by updating the superpixels mean values at each image frame, the classifier can be adapted to the changes these factors may cause. The maximum difference values, computed in the training phase, remain constant through the process. The updated mean values are obtained by averaging the current mean values with the previous ones. This update is only carried on if the number of superpixels in the RoI varies abruptly, up to a given threshold, from one frame to the next. The reasoning of this is that when navigating through a RoI, the total number of pixels in consecutive frames varies slightly. Of course, there exist exceptions to the rule, for example when a dynamic object (fish, diver) appears suddenly in front of the robot. Those cases are complex as there will not be enough response time to change the current direction of the robot so to avoid collision. In that case, it is always expected that the moving object attempts to dogde the robot.

To obtain the direction of escape, we first determine the position of the mass center of the RoI. If there exists more than one RoI then we start analyzing the one with greater area. Once the mass center is calculated, we test if it is safe for navigation by setting a circle around the center mass of radius size r (see Figure 4). If all superpixels inside the circle are in the RoI, then this is the direction of escape, otherwise the pitch angle is set to $-2kp_1$ (an upward direction), where kp_1 is a constant coefficient that indicates the actual proportion of upward motion and its related to the camera's field of view. Thus, our navigation control scheme uses an image-based visual servoing, where our control law is constructed to map the image error (in terms of the mass center of RoI) to robot motion directly.

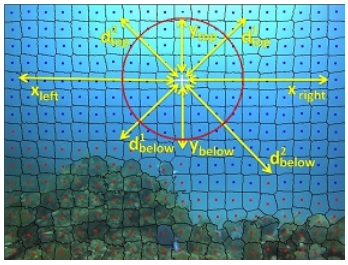


Fig. 4. Determining the mass center of a RoI. The yellow arrows are the measures used to determine the center of mass. If all superpixels inside the red circle are in the RoI then that superpixel is the direction of escape.

All the described algorithms that conform the navigation framework, run in O(n) time.

F. Control scheme

Although the robot platform comes with an in-house designed autopilot, we have developed our own robust control scheme which is independent of the dynamic model of the vehicle and allows the robot to autonomously navigate at various speeds while maintaining its orientation, despite external disturbances. In this section, we briefly describe the control scheme (for more details see [16]). The control scheme is a sub-optimal robust control for position tracking task of underwater vehicle. The main idea behind the design is to add to an existing PID lineal regulator, a nonlinear feedback loop to improve the robustness and performance of the resulting closed-loop system over the linear regulator. Also, the passivity property of the system is used for control design, which ensures locally stable behavior when the desired velocities and accelerations of the vehicle are bounded

as well as the position error, the velocity and acceleration of the fluid. The sub-optimal robust control for the underwater robot is:

$$F_u = \hat{M}\dot{\nu}_r + \hat{D}_v\nu_r - J^{-T}(q)\left(K_s s + K_i \int s + \beta\|s\|^2 s\right), \quad (1)$$

with constant positive definite matrices $\hat{M}$, $\hat{D}_v$, K_s, K_i, and Λ and positive scalar β, with s and the reference vehicle twist ν_r defined as

$$s = \dot{\tilde{q}} + \Lambda\tilde{q}, \quad (2)$$

$$\begin{aligned}\nu_r J(q)\dot{q}_r &= J(q)(\dot{q}_d - \Lambda\tilde{q}) \quad (3)\\ &= \nu_d - J(q)\Lambda\tilde{q},\end{aligned}$$

With this control, there always exist positive gains (β, K_s, K_i, Λ), such that close-loop regime is locally stable, i.e. $s \rightarrow 0$, provided that the feedback signals ν_d, $\tilde{q}$ are bounded, and consequently, both errors $\tilde{q}$ and $\dot{\tilde{q}}$ are also locally stable.

IV. FIELD TESTING

We now discuss the experimental setup and results. First, we describe the robot platform we use to verify our method.

A. Robot platform

We use an amphibious robot named Mexibot (see Figure 5). It belongs to the family of AQUA robots [17]. Contrary to traditional aquatic robots and teleoperated devices that use thrusters and remote control for mobility, the AQUA robots are cabaple of untethered amphibious operation. In water, the robot's propulsion is based on 6 fins that can provide motion in 5 DoF up to depths near 35 meters (see [18] for more details). Mexibot is of medium size ($60 \times 45 \times 12$ cm) and weights around 16.5kg. This allows for easy maneuverability which is important in the time response on the robot control when navigating with the purpose of closely monitoring an unstructured environment.

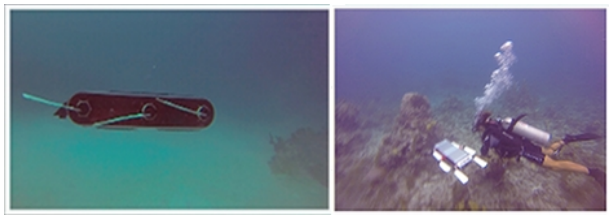


Fig. 5. Mexibot, our robot platform.

B. Experimental setup

To test our approach for segmentation, classification and direction of escape estimation, we first carried out offline experiments. We have built a benchmark data containing underwater videos captured by our robot. We run each of the algorithms explained in the previous section and try different values for the threshold parameters. For all our trials we set the number of superpixels at each frame to be $K = 300$. Thus, for an image with size of 480×640, the approximate size of each superpixel is 1024 and, for an image with 120×160, the average size would be 64 pixels. Since we require real-time navigation, we focus mainly on reducing as much as possible the computational time of our segmentation algorithm while keeping its robustness. For the connectivity estimation, we set $\Theta_C = \sqrt{2}$. The mean values are updated if and only if the previous frame contains at least 20 superpixels labeled as a RoI.

Figure 6 shows some offline results. Each image shows the classification of superpixels. Those superpixels labeled with blue dots are considered inside a RoI and those labeled with red dots are considered objects (i.e., obstacles). The white cross indicates the direction of escape to which the robot should move. It can be observed that all of the examples

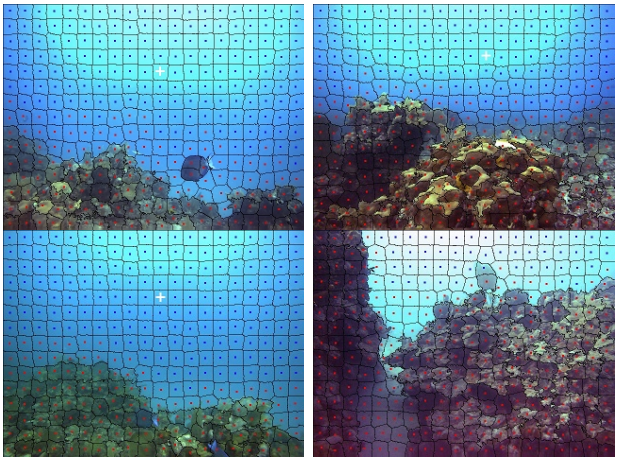


Fig. 6. Offline results when running the superpixel segmentation, classification of RoI, and estimation of direction of escape algorithms. Superpixels with blue dots are inside a RoI (water). The white cross indicates the direction of escape to which the robot should move. Note than in the right bottom image a direction of escape was not found. Therefore, the next robot motion must be in an upward direction to avoid collision.

show good results in the classification of the RoI, which in turns allows for a good estimation of the direction of escape. The offline tests allowed us to get the right values on the algorithm parameters to be used in the experimental trials. We carried out the trials in two different environments. The first one was in deep ocean water (about 15 meters) where most of the time the robot was navigating around dynamic objects (divers swimming around). The second set of experiments were carried on a lake, with no good water conditions – the water was very turbid and also since the trials were done in shallow water, the illumination conditions changed abruptly from frame to frame, due to sun rays reflection of water surface. For this experiment, we use dynamic artificial objects, specifically a fin, and a person's foot, in order to evaluate the performance of our reactive navigation framework in practice. Figures 7 and 8 show some snapshots with the results on the classification and direction of escape estimation (we have attached a video showing the performance of our reactive autonomous navigation framework). It can be observed that the classification and estimation of the direction of escape were successfully achieved in most of the image frames. However, few frames

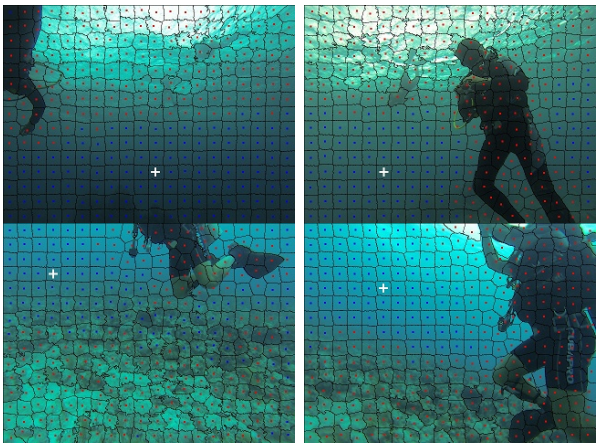


Fig. 7. Results of our visual-based reactive navigation framework with obstacle avoidance in deep ocean water. Blue dots indicate superpixels inside a RoI. The white cross indicates the direction of escape to which the robot should move. Note that even with illumination changes and different color tones of the sea water our approach performs very well.

in which the classification was not correct, is mainly because of the abrupt changes in the illumination due to the reflection on the sun light in the water surface. Nevertheless, our approach is robust for most cases presenting illumination variations. It is important to highlight that this approach has the ability to explore the natural unstructured environment with full autonomy, i.e., we do not use artificial underwater target objects. Other aspect is that we do not use artificial illumination to enhance the visualization, which can be particular dangerous for some species of fish.

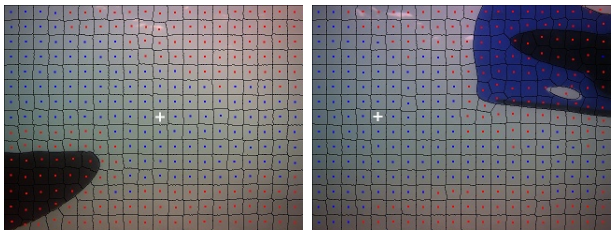


Fig. 8. Results of our reactive navigation framework with obstacle avoidance in shallow lake water. It can be observed that although the water conditions were poor, the performance of our approach is really good.

V. CONCLUSIONS AND FUTURE WORK

We have presented a novel method for underwater autonomous navigation with real time obstacle avoidance. Unlike previous work in underwater robotics research, which use a combination of active and passive sensors, our method is based only on visual information. We have adapted the SLIC superpixel segmentation algorithm to allow for a fast and robust segmentation of underwater images. The segmentation is based on a connectivity graph of the superpixels, which is also useful to accurately detect the regions of interest from which the direction of escape is estimated. The training phase of the regions of interest is quite robust, as it is done online and it can be updated as time passes, so allowing to adjust to the local water conditions. We have successfully tested the proposed framework in our aquatic robot both in deep ocean water (10 to 18 meters) and in a lake, in clear and turbid water conditions.

Our ongoing future work is focused on developing better strategies for exploring the coral reef in terms of recognizing high-level patterns in superpixels to semantically segment the scene and control the robot's speed in the navigation.

VI. ACKNOWLEDGEMENTS

We would like to thank CONACyT for their support and project funding.

REFERENCES

[1] C. Georgiades, A. German, A. Hogue, H. Liu, C. Prahacs, A. Ripsman, R. Sim, L. A. Torres-Méndez, P. Zhang, M. Buehler, G. Dudek, M. Jenkin, and E. Milios, "AQUA: an aquatic walking robot," in *IROS*, vol. 3, pp. 3525–3531, 2004.

[2] P. Corke, C. Detweiler, M. Dunbabin, M. Hamilton, D. Rus, and I. Vasilescu, "Experiments with underwater robot localization and tracking," in *ICRA*, 2007.

[3] M. Dunbabin, K. Usher, and P. Corke, "Visual motion estimation for an autonomous underwater reef monitoring robot," in *Intl. Conf. on Field and Service Robotics*, pp. 31–42, Springer Verlag, 2006.

[4] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "SLIC superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. on PAMI*, vol. 34, no. 11, pp. 2274–2282, 2012.

[5] D. A. Owens, *Twilight vision and road safety*, ch. Visual perception: The influence of H. W. Leibowitz. Decade of behavior. Washington: American Psychological Association, 2003.

[6] J. Brooks and D. A. Owens, "Effects of luminance, blur, and tunnel vision on postural stability," *Journal of Vision*, vol. 1, no. 3:304, 2001.

[7] F. G. Rodríguez-Telles, L. A. Torres-Méndez, and E. A. Martínez-García, "A fast floor segmentation algorithm for visual-based robot navigation," in *Intl. Conference on Computer and Robot Vision*, 2013.

[8] Y. Girdhar, P. Giguere, and G. Dudek, "Autonomous adaptive exploration using realtime online spatiotemporal topic modeling," *Intl. Journal of Robotics Research*, 2013.

[9] A. Kim and R. M. Eustice, "Real-time visual SLAM for autonomous underwater hull inspection using visual saliency," *IEEE Trans. on Robotics*, vol. 29, no. 3, pp. 719–733, 2013.

[10] P. King, A. Vardy, P. Vandrish, and B. Anstey, "Real-time side scan image generation and registration framework for auv route following," in *IEEE/OES Autonomous Underwater Vehicles*, pp. 1–6, 2012.

[11] F. García-Córdova and A. Guerrero-González, "Intelligent navigation for a solar powered unmanned underwater vehicle," *Intl. Journal of Advanced Robotic Systems*, 2013.

[12] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Trans. on PAMI*, vol. 22, no. 8, pp. 888–905, 2000.

[13] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *IJCV*, vol. 59, pp. 167–181, September 2004.

[14] A. Moore, S. Prince, J. Warrell, U. Mohammed, and G. Jones, "Superpixel lattices," in *CVPR*, 2008.

[15] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE Trans. on PAMI*, vol. 31, no. 12, pp. 2290–2297, 2009.

[16] R. Pérez-Alcocer, E. Olguín-Díaz, and L. A. Torres-Méndez, "Model-free robust control for fluid disturbed underwater vehicles," in *ICIRA*, pp. 519–529, 2012.

[17] G. Dudek, P. Giguere, C. Prahacs, S. Saunderson, J. Sattar, L. A.Torres-Mendez, M. Jenkin, A. German, A. Hogue, A. Risman, J. Zacher, E. Milios, H. Liu, P. Zhang, M. Buehler, and C. Georgiades, "AQUA: An amphibious autonomous robot," *Computer*, vol. 40(1), pp. 46–53, 2007.

[18] J. Sattar, G. Dudek, O. Chiu, I. Rekleitis, P. Gigure, A. Mills, N. Plamondon, C. Prahacs, Y. Girdhar, M. Nahon, and J.-P. Lobos, "Enabling autonomous capabilities in underwater robotics," in *IROS*, pp. 3628–3634, 2008.