COBEM UBERLÂNDIA 2019

ABCM Associação Brasileira de Engenharia e Ciências Mecânicas

25th ABCM International Congress of Mechanical Engineering
October 20-25, 2019, Uberlândia, MG, Brazil

COB-2019-0669
DETECTION AND TRACKING OF UNDERWATER PIPELINES USING COMPUTER VISION

Carlos A. Soto B.
Jun Okamoto Jr.
Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos. Av. Professor Mello Moraes, 2231 — 05508-030 São Paulo, SP
csotob@usp.br, jokamoto@usp.br

Ettore A. de Barros
Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos. Av. Professor Mello Moraes, 2231 — 05508-030 São Paulo, SP
eabarros@usp.br

Abstract. *In the last years, the use of Autonomous Underwater Vehicles (AUV) has been increased in the inspection of offshore structures. This area is significant for petroleum and gas companies because they need to inspect their underwater pipelines to prevent accidents due to corrosion and consequently fluid leaks. For an automated pipeline inspection task with an AUV, the development of a tracking system is necessary. This paper proposes a method for the detection and tracking of underwater pipelines with video-cameras. The proposed algorithm combines image processing tools to estimate the orientation and position of a pipeline. These parameters are sent to the AUV control system. A real-time simulator of the movement of an AUV is used for validating the tracking control. The results of these tests are presented and demonstrate that this method is robust for pipeline tracking in a real submarine environment.*

Keywords: *AUV, Pipelines, Kalman Filter, Tracking system, Image Processing*

1. INTRODUCTION

For an automated pipeline inspection with AUVs, the development of a tracking system is necessary. This system has to detect the pipeline and correct the course of the vehicle in order to follow the pipeline contour during maneuvers. For this task, algorithms have been extensively studied by many researchers. Thus, some publications show different methods to detect and track pipelines (Goyal. D. *et al.*, 2016), (Zhang *et al.*, 2013), (Mitchell *et al.*, 2014). Most of these publications assume the detection of pipelines by video cameras and their benefits are emphasized in the work of Khan *et al.* (2016). Through image processing, it is possible to estimate the pipeline position and orientation within the image (Chen *et al.*, 2015). Concerning the tracking control issue, few authors have proposed control strategies, such as those ones based on the PID algorithm (Shi *et al.*, 2017), or a nonlinear technique (Narimani *et al.*, 2009).

This paper describes a robust method for the detection of the orientation and position of a pipeline using image processing with a single underwater video camera. A real-time simulator of the movement of an AUV and a PID controller are used to test the tracking algorithm. This work is presented as follows. Section 2 describes the proposed method to detect the pipeline, section 3 presents the tracking control algorithm, section 4 shows the analysis of validation results and section 5 presents the final conclusions and future steps of this research.

2. IMAGE PROCESSING FOR PIPELINE TRACKING

Firstly, individual frames from an RGB camera are saved as images and converted to grayscale as the color information is not necessary and grayscale images processing is much faster than color images processing. Then, a median filter was applied to smooth the image, and later, a Canny operator was used to find edges in the filtered image. Finally, the Hough transform was employed to find straight lines in the processed image. A Median and a Kalman Filter were applied in the data output to estimate the best value of the angle and position of the pipeline in the frame. In order to evaluate the algorithm shown in this paper, some videos of a real underwater pipeline were used. The algorithm was implemented by using in C++ Language with OpenCV library in Linux Platform. These operations are described in detail as follows.

2.1 Image acquisition

The images were acquired from a video recorded from the seabed with 1 minute and 45 seconds of duration. This video has a resolution of 25 frames per second, and each frame represents an image to be processed. This frame is a color image in RGB scale and has 876x494 pixels of size. An example of this image is shown in Fig. 1.

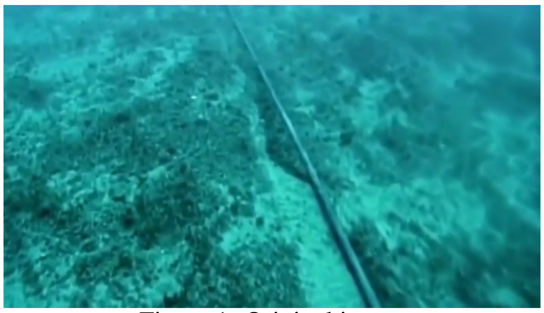


Figure 1: Original image

2.2 Grayscale Conversion

In this step, the image was transformed from RGB color space to grayscale, as shown in Fig. 2.

2.3 Image Filter

Grayscale images are easier to be processed than the colored ones. However, this monochromatic image still shows noise problems that could interfere with subsequent procedures. For this reason, a median filter of size 12x12 was used to reduce the noises and preserve the information about the edges. The operation of this filter consists of running through each element of the image and replaces each pixel with the median of its neighboring pixels. The filtered image is shown in Fig. 3.

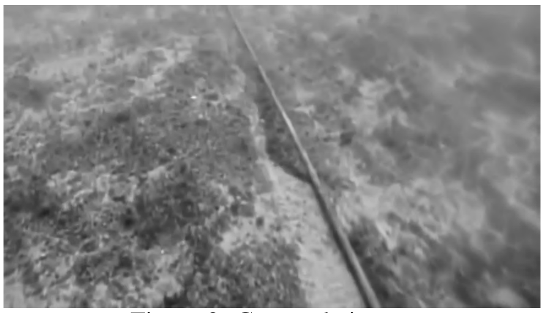


Figure 2: Grayscale image

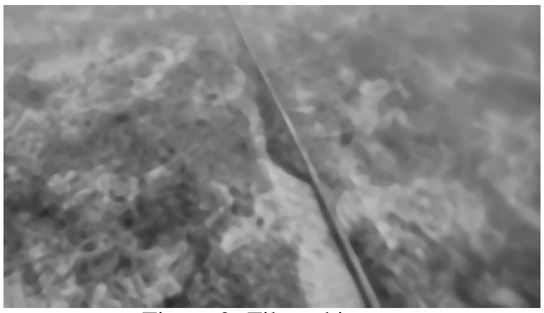


Figure 3: Filtered image

2.4 Edge Detection

In this step, the image was segmented to highlight the edges of the pipeline. This segmentation created a binary image through image operators that are described as follows:

Scharr Operator

This operator uses two kernels of size 3x3 to estimate the gradient of the intensity of the image in each pixel, as described in Eq. (1) and Eq. (2).

$$\mathbf{G_x} = \begin{bmatrix} -3 & 0 & +3 \\ -10 & 0 & +10 \\ -3 & 0 & +3 \end{bmatrix} * \mathbf{I(x,y)} \tag{1}$$

$$\mathbf{G_y} = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ +3 & +10 & +3 \end{bmatrix} * \mathbf{I(x,y)} \tag{2}$$

The images produced by $\mathbf{G_x}$and $\mathbf{G_y}$, that represent the gradient about the axes x and y respectively, are averaged to create a single image (Fig. 4).

Thresholding

The threshold value is used to divide the image in two intensities, white and black. In this study, the threshold value was calculated using Otsu's Method, Otsu (1979).

Canny Detector

This operator removes edges that are below the threshold value and analyzes if some edges are concatenated to be considered as possible edges of the pipeline. Figure 5 shows the binary image after the application of the Canny Detector.

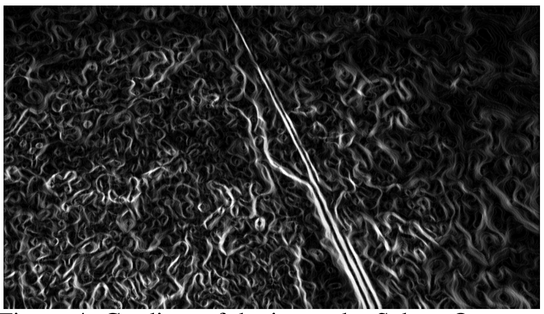


Figure 4: Gradient of the image by Scharr Operator

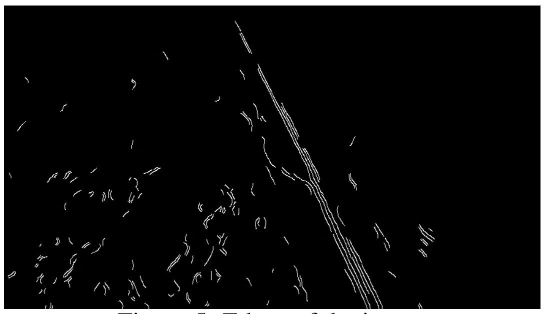


Figure 5: Edges of the image

2.5 Hough Transform

In this step, the Hough Transform was used to detect straight lines in the binary image. This process was improved using some constraints to identify which lines represent the edges of the pipeline. The main constraint was to assume just two straight lines such as edges of the pipeline (Fig. 6). The pipelines should correspond to the maximum number of pixels. The last constraint imposes the angle difference between those lines to be less than twenty degrees.

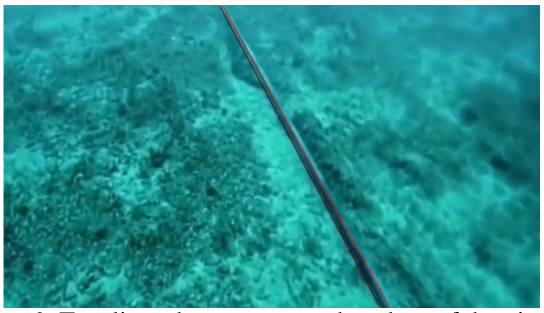


Figure 6: Two lines that represents the edges of the pipeline

The Hough transform defines each straight line by two parameters: an angle ($\theta_{h:1,2}$) and a distance ($\rho_{h:1,2}$). Finally, the two straight lines were averaged to achieve a single one. In this way, a unique value of angle (θ_h) and distance (ρ_h) are produced.

2.6 Median Filter

In this step, a median filter was used to eliminate some peaks from the measurement of the angle (θ_h) and the distance (ρ_h) previously calculated by the Hough Transform. Thus, for each frame processed, this filter calculates the median of the last ten values of θ_h and ρ_h, and produces new values for the angle (θ_m) and distance (ρ_m).

It is important to emphasize that this filter depends on the measurement of θ_h and ρ_h. In the case of too inaccurate or invalid measures in consecutive frames, the filter produces unsatisfactory results.

2.7 Kalman Filter

In this step, the Kalman Filter was used to improve the accuracy of θ_m and ρ_m values. Apart from using the filtered values produced by the Median filter, this filter also uses a mathematical model (Eq. (4)) to estimate the parameters of the pipeline (θ and ρ) , producing more accurate results. In this way, this filter does not depend exclusively on the measure of θ_m and ρ_m to carry out the estimation; however, to obtain a correct estimation, it is necessary to know if the measured values of θ_m and ρ_m are valid or not. For this reason, a pre-adjustment in the input to the Kalman Filter was created through the application of a confidence index (IC). This index uses some conditions ($IC_{1\to9}$) to verify if the line detected corresponds to the pipeline.

$$IC = \frac{\sum\limits_{i=1}^{9} IC_i}{9} \tag{3}$$

- IC_1 is the quantity of pixels that form the first straight line in the Hough transformation.
- IC_2 is the quantity of pixels that form the second straight line in the Hough transformation.
- IC_3 is the difference of the angle value (θ_h) between two consecutive frames before applying the median filter.
- IC_4 is the difference of the distance value (ρ_h) between two consecutive frames before applying the median filter.
- IC_5 is the difference of the filtered angle value (θ_m) between two consecutive frames after applying the median filter.
- IC_6 is the difference of the filtered distance value (ρ_m) between two consecutive frames after applying the median filter.
- IC_7 is the difference of the filtered angle value (θ_m) and the non-filtered angle value (θ_h) between two consecutive frames.
- IC_8 is the difference of the filtered distance value (ρ_m) and the non-filtered distance value (ρ_h) between two consecutive frames.
- IC_9 is the threshold value in the edge detection step.

During the image collection, the IC value was compared to the threshold value 100 in order to validate the measurement. If the IC value is less than 100, the measure is considered invalid and the Kalman filter does not execute the update process; in this case, the filter only executes the propagation process according to the linear model (Eq. (4)). However, this propagation process affects the P covariance value which will increase until a valid pipeline detection measurement is found. If the P covariance value is greater than 10, the pipeline is considered lost, and the program interrupts the calculation of estimates. This decision prevents the vehicle follows a wrong path and runs the risks of getting lost.

Figure 7 explains how the Kalman filter Propagation and Update steps work in the proposed algorithm for estimating vehicle position.

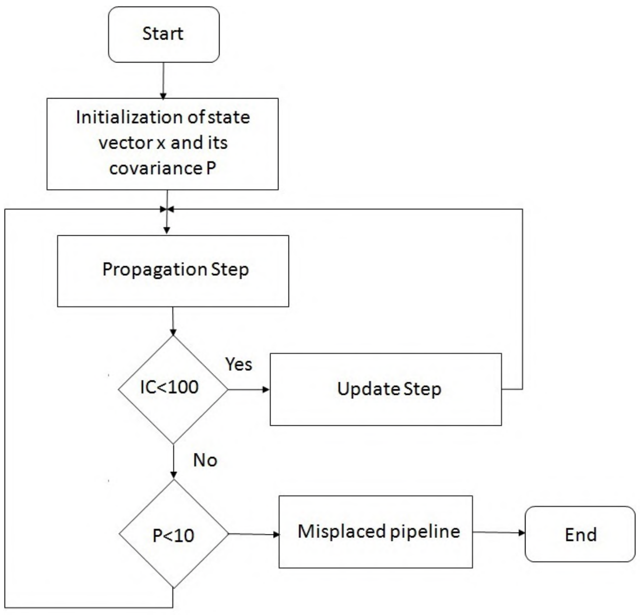


Figure 7: Flowchart for Kalman Filter Propagation and Update steps

Equation 4 shows the linear model that was used in the Kalman Filter. In this equation, a variable dt is the time interval between one frame and the subsequent one.

$$\boldsymbol{A} = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

The state vector of the model is expressed in Eq. (5).

$$\boldsymbol{x} = \begin{bmatrix} \theta_m \\ \rho_m \\ \bar{\theta_m} \\ \bar{\rho_m} \end{bmatrix} \tag{5}$$

Where θ_m and ρ_m are the angle and the distance produced by the Median filter, respectively. Thus, $\bar{\theta_m}$ and $\bar{\rho}_m$ are the rates of change between the two consecutive values of θ_m and ρ_m. Figures 8 and 9 show the estimated angle θ and the estimated distance ρ. These parameters are necessary for estimating the pipeline pose and posteriorly the heading AUV reference angle to track the pipeline.

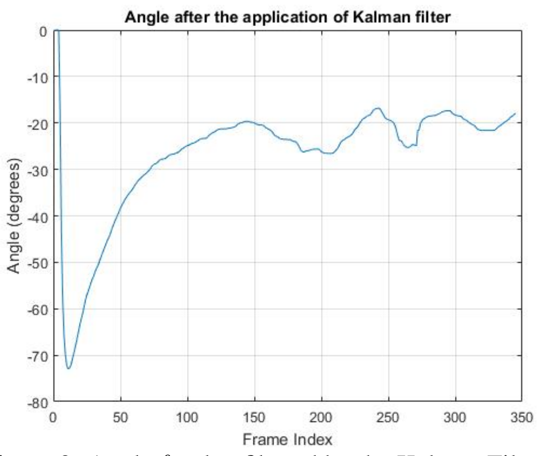


Figure 8: Angle θ value filtered by the Kalman Filter

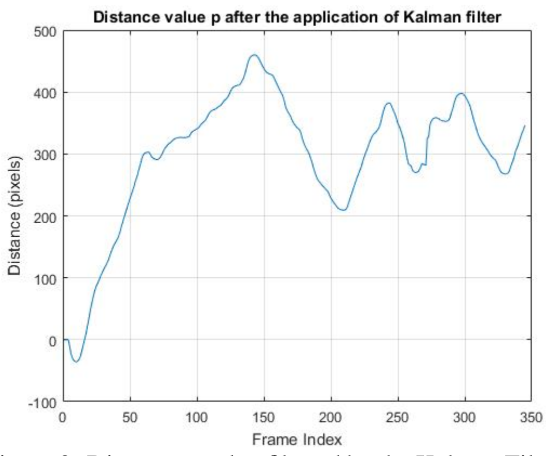


Figure 9: Distance ρ value filtered by the Kalman Filter

2.8 Estimation of the pipeline pose

The output of the Kalman filter included two parameters of the pipeline: θ and ρ, that are used to calculate the input variables to the pipeline tracking system. In this way, a geometrical relationship is used in order to estimate the pipeline pose as shown in Fig. 10.

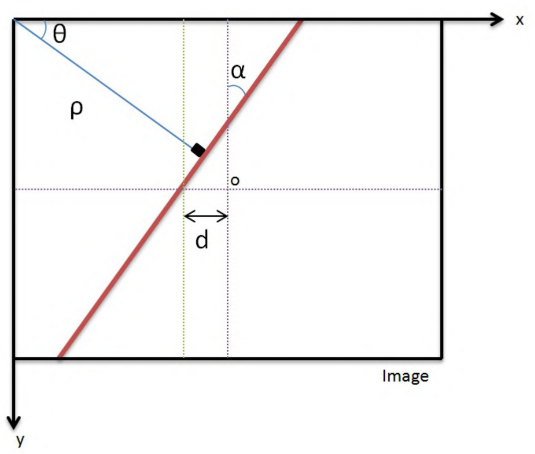


Figure 10: Representation of pipeline pose

The angle α and the distance d represent the pipeline pose and are considered the input variables to the pipeline tracking system. The angle α is the angle between the pipeline and the vertical axis, while, the distance d is the displacement in relation to the video frame center in the axis x. The values of α and d were shown in Fig. 11 and Fig. 12 for each frame processed.

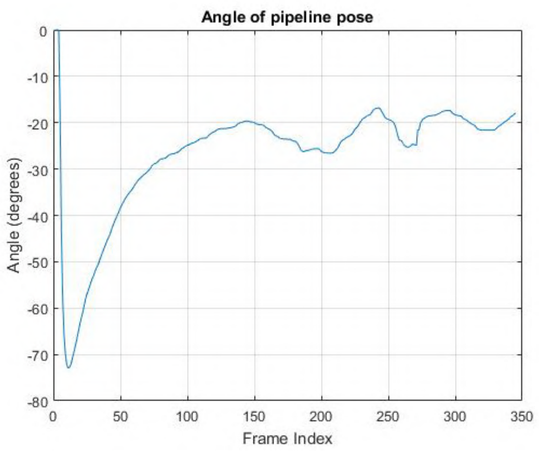


Figure 11: Angle α of pipeline pose

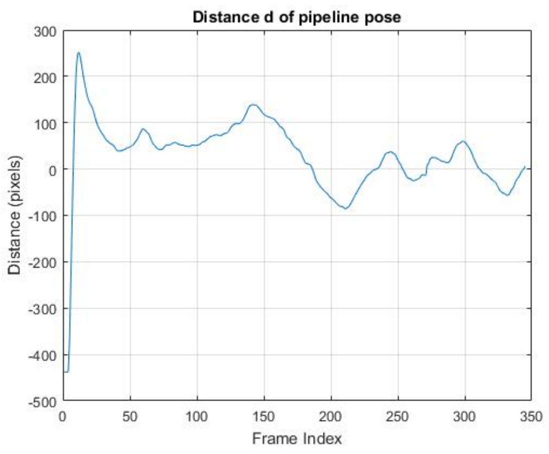


Figure 12: Distance d of pipeline pose

Equations 6 and 7 show how the pipeline pose was calculated.

$$\alpha = \theta \tag{6}$$

$$d = \frac{(x_1 + x_2)}{2} - x_m \tag{7}$$

where,

$$x_m = \frac{\#pixels_{cols}}{2} \tag{8}$$

$$x_1 = \frac{c}{m} \tag{9}$$

$$x_2 = \frac{\#pixels_{rows} - c}{m} \tag{10}$$

The parameters $\#pixels_{rows}$ and $\#pixels_{cols}$ are the number of rows and columns of the image, respectively. Therefore, the values of c and m depend on θ and ρ that were calculated previously.

$$c = \frac{\rho}{\sin\theta} \tag{11}$$

$$m = \frac{-\cos\theta}{\sin\theta} \tag{12}$$

3. THE TRACKING CONTROL SYSTEM

The image processing algorithm detects the pipeline inside a video frame and estimates the angle between the pipeline and the vertical axis and also its displacement in relation to the video frame center. These parameters are necessary for calculating the reference of the heading angle φ of the AUV according to the Eq. (13).

$$\varphi = \varphi_1 + \varphi_2 \tag{13}$$

The distance d and the angle α determine the desired heading of the vehicle. Both contribute to the reference heading angle, as described by Eq. (14), and Eq (15), respectively.

$$\varphi_1 = abs(d) * \varphi_{max} * \frac{2}{\#pixels_{cols}} \tag{14}$$

$$\varphi_2 = \alpha \tag{15}$$

The value of φ_{max} depends on the angle α according to the following expression:

$$\varphi_{max} = \begin{cases} \alpha - 90 & d < 0 \\ \alpha + 90 & d > 0 \end{cases} \tag{16}$$

This value is sent to a PID control with the following parameters of K_p, K_i, K_d:

- $K_p = 0.35$
- $K_i = 0$
- $K_d = 1.5$

In this work, a simulation of the Pirajuba AUV dynamic system was used to validate the proposed algorithm. Figure 13 shows the block diagram representing the image processing system, and the AUV simulator together with the tracking system that has been implemented in different computers that use serial communication for data exchange. The control variable is saturated at $\pm 20°$, because of the limits of the rudder deflection. The control frequency was set to 5 Hz. During the tracking system simulation, the ratio between the scenario size and the pixels was set to 20:1, so that 1 meter is equivalent to 20 pixels. Only the values of yaw angle setpoint and the simulated yaw angle are exchanged between those computer systems.

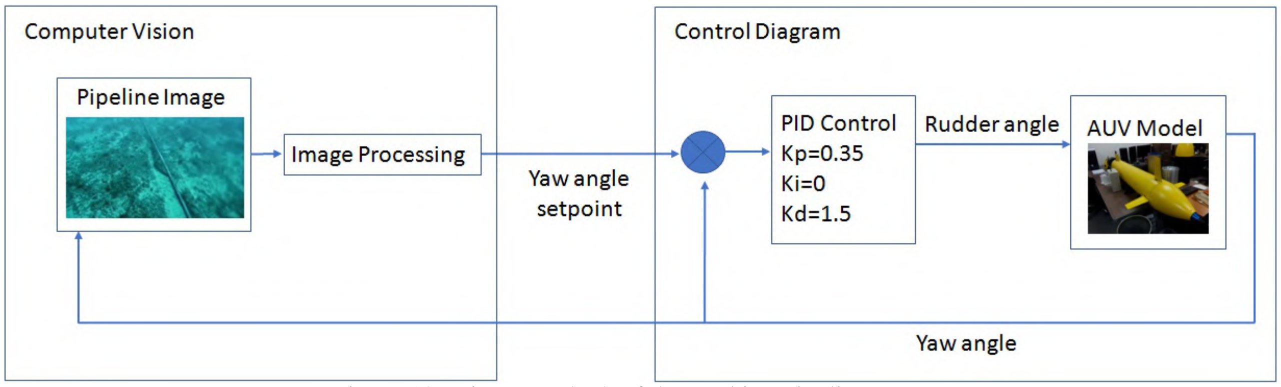
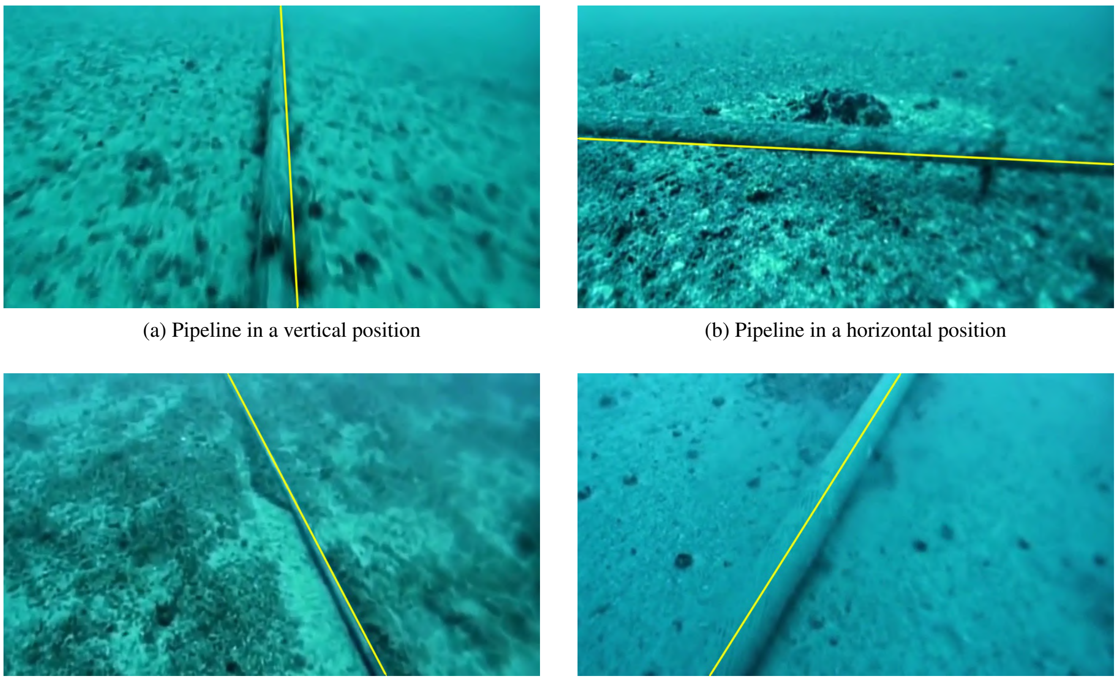


Figure 13: Diagram Block of the tracking pipeline system

4. RESULTS

Some trials have been carried out in order to verify the robustness of the detection method. Figure 14 shows four images from different situations where the pipeline was detected. There are seaweed and stones that act as noise in all images. In Fig. 14a, the pipeline is in a vertical position and the quantity of noise is regular. Figure 14b shows a pipeline in a horizontal position, the noise is regular too and the image is slightly blurred on its right. In Fig. 14c, the pipeline is farther away from the camera and the noise is a bit smaller. Finally, Figure 14d shows the pipeline more closely, the noise is a bit smaller too.


(a) Pipeline in a vertical position
(b) Pipeline in a horizontal position
(c) Pipeline away from the camera
(d) Pipeline near to the camera
Figure 14: Images of the pipeline represented by a straight line

The processing time for each frame of the video is presented in Fig. 15. In the image processing stage, the images were processed according to this processing time. Thus, to simulate a real-time environmental, each image were processed skipping some frames.

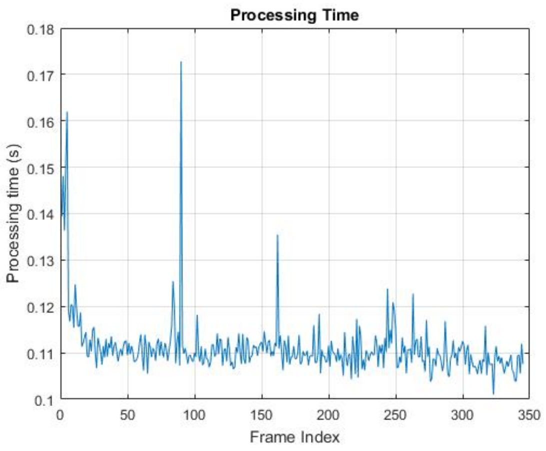


Figure 15: Processing time for each frame processed

The processing time is about 110 ms for each frame. The algorithm was implemented in C++ with the Open CV Library and tested in a 2.4GHz Processor Intel i5 with Linux. An image frame from the video was used for testing the tracking system. The simulated AUV motion was represented by rotations and translations of the original image. The simulation of the AUV was performed in Simulink – Matlab. The result of the response of yaw of the vehicle is shown in Fig. 16.

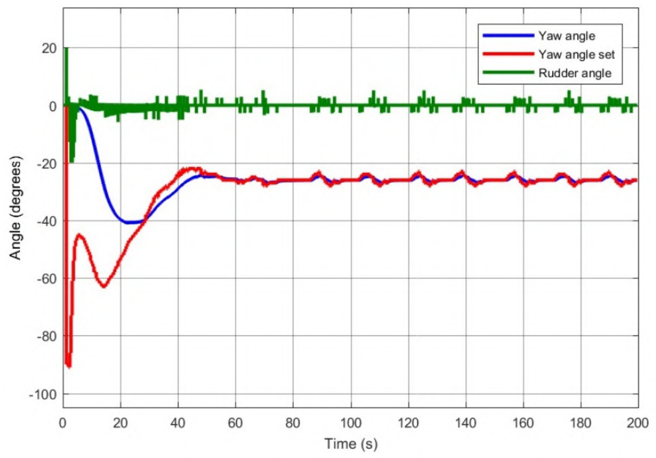


Figure 16: AUV response

The image that was used to test the tracking control is shown in Fig. 17. This image is rotated and translated depending on the AUV response; then, the tracking simulation is carried out.

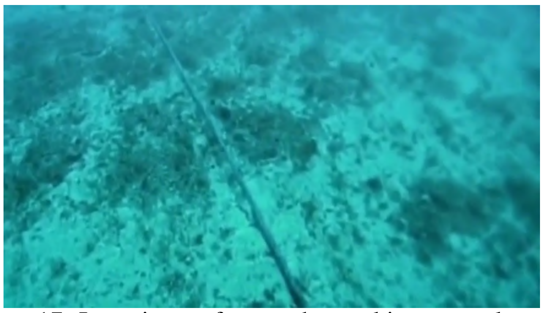


Figure 17: Input image for test the tracking control system

Figures 18 and 19 show that angle and displacement converge to zero. That means the pipeline image converges to the middle of the picture with a vertical position in the simulation of the AUV tracking control system.

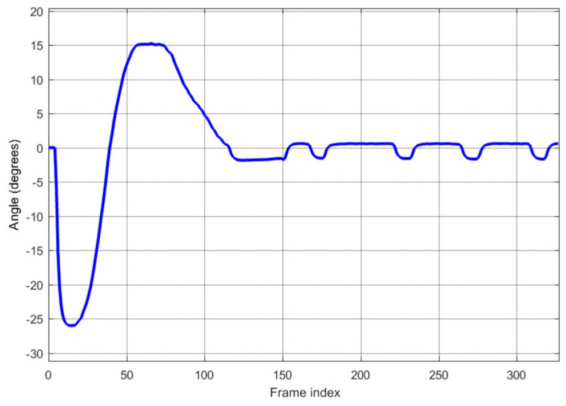


Figure 18: Angle α between the vertical axis and the pipeline image

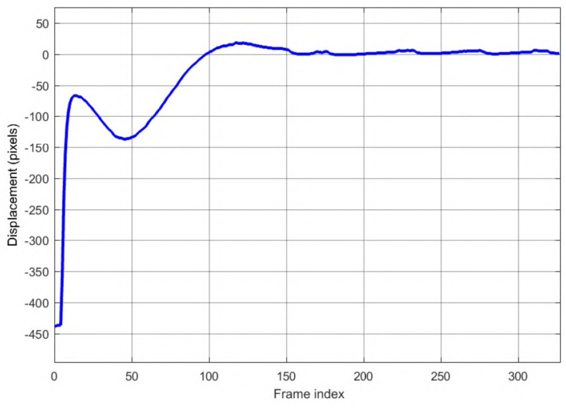


Figure 19: Distance d between the pipeline and the center of the image

Overall, the results indicated that there is a robust detection because the pipeline was found in different situations as can be seen in Fig. 14. The use of the Kalman filter improves the detection considerably. Its use can be seen in different works as Shi *et al.* (2017), Goyal. D. *et al.* (2016), Zhang *et al.* (2012), Asif and Rizal (2006), Ortiz *et al.* (2002); where its application also obtains satisfactory results.

Concerning the processing time, Asif and Rizal (2006) obtained a processing time of 1 second, for this reason, they recommended using the C ++ language to reduce this processing time. In this work, the algorithm was implemented using the C++ language and the results show that the processing time is really low compared to the work of Asif and Rizal

(2006). On the other hand, the work of Goyal. D. *et al.* (2016) have a processing time of 94.02 ms, however, the image data was acquired differently. Their videos were recorded in a water tank using a simulator, while in this work a real submarine pipeline in the bottom of the sea was used.

5. CONCLUSION

In this paper, an image processing system was developed to detect an underwater pipeline. Unlike other pipeline detection algorithms, the results obtained in this research demonstrate the robustness of the proposed method to find the pipeline pose with low processing time. This pipeline pose is used to calculate the desired heading angle reference of the vehicle. This heading value is used in the AUV tracking control in real-time without the demand of other robust control processing. In this way, the cost of developing the controller is significantly reduced because a simple control can be used to make a fully reliable tracking system. Finally, using hardware in the loop shows that the proposed method is effective to detect and track real underwater pipelines. A pipeline tracking simulator was developed to demonstrate the correct performance of this algorithm. Future studies will focus on the real-time tests using an underwater vehicle for tracking a pipeline laying on the sea bottom.

6. ACKNOWLEDGMENTS

The present work was supported by the Postgraduate Students Program – PEC-PG, of CNPq – Brazil. The video and the images that were used in this work was provided by the Diver's World Company.

7. REFERENCES

Asif, M. and Rizal, M., 2006. "An Active Contour and Kalman Filter for Underwater Target Tracking and Navigation". *Mobile Robots: towards New Applications*, , No. June 2014. doi:10.5772/4699.

Chen, H.H., Chuang, W.N. and Wang, C.C., 2015. "Vision-based line detection for underwater inspection of breakwater construction using an ROV". *Ocean Engineering*, Vol. 109, pp. 20–33. ISSN 00298018. doi:10.1016/j.oceaneng. 2015.09.007. URL `http://dx.doi.org/10.1016/j.oceaneng.2015.09.007`.

Goyal. D., Shetti, K. and Bretschneider, T., 2016. "Robust Vision-Based Detection and Tracking of Underwater Pipelines for AUVs". *Proceedings of the 2012 International Conference on Detection and Classification of Underwater Targets*, , No. 1, pp. 254–261.

Khan, A., Ali, S.S.A., Malik, A.S., Anwer, A., Hussain, N.A.A. and Meriaudeau, F., 2016. "Control of autonomous underwater vehicle based on visual feedback for pipeline inspection". *2016 2nd IEEE International Symposium on Robotics and Manufacturing Automation (ROMA)*, pp. 1–5. doi:10.1109/ROMA.2016.7847814. URL `http://ieeexplore.ieee.org/document/7847814/`.

Mitchell, B., Mahmoudian, N. and Meadows, G., 2014. "Autonomous Underwater Pipeline Monitoring Navigation System". Vol. 9090, pp. 1–8. ISSN 1996756X. doi:10.1117/12.2055178.

Narimani, M., Nazem, S. and Loueipour, M., 2009. "Robotics vision-based system for an underwater pipeline and cable tracker". *OCEANS '09 IEEE Bremen: Balancing Technology with Future Needs*, pp. 0–5. doi:10.1109/OCEANSE. 2009.5278327.

Ortiz, A., Simó, M. and Oliver, G., 2002. "A vision system for an underwater cable tracker". *Machine Vision and Applications*, Vol. 13, No. 3, pp. 129–140. ISSN 09328092. doi:10.1007/s001380100065.

Otsu, N., 1979. "A threshold selection method from gray-level histograms". *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, pp. 62–66. ISSN 0018-9472. doi:10.1109/TSMC.1979.4310076.

Shi, L., Chen, Z., Guo, S., Guo, P., He, Y., Pan, S., Xing, H., Su, S. and Tang, K., 2017. "An underwater pipeline tracking system for amphibious spherical robots". *2017 IEEE International Conference on Mechatronics and Automation, ICMA 2017*, pp. 1390–1395. doi:10.1109/ICMA.2017.8016020.

Zhang, J., Liu, L., Wang, B., Chen, X., Wang, Q. and Zheng, T., 2012. "High speed automatic power line detection and tracking for a UAV-based inspection". *Proceedings of the 2012 International Conference on Industrial Control and Electronics Engineering, ICICEE 2012*, pp. 266–269. doi:10.1109/ICICEE.2012.77.

Zhang, T., Wan, L. and Li, Y., 2013. "Underwater Pipeline Detection and Following of AUV Based on the Optic Vision". pp. 1–5.

8. RESPONSIBILITY NOTICE