Flinders University,

- Adelaïde, South Australia -, Australia

ENSTA Bretagne

- Brest, Finistère - France

Internship Report

Object Detection from Multibeam Sonar Imaging

05/25/2020 - 08/28/2020

Internship carried out as part of the Nicolas Baudin exchange program in view of the validation of the engineering diploma delivered by ENSTA Bretagne.

Conducted by :

- Danieau Pierre-Louis, Student engineer at ENSTA Bretagne.

Internship masters :

- Sammut Karl, Professor at Flinders University and Director of Centre for Maritime Engineering,
- Wheare Jonathan, Doctor at Flinders University.

Academic referent :

- Clément Benoît, Professor at ENSTA Bretagne.
- Malicoutis Zacharie, Head of International Mobilities at ENSTA Bretagne

Table of contents

Acknowledgements 3

Introduction 4

I / Context 5

1) Presentation 5

a) Concept of an AUV and presentation of the X300 robot 6

b) My interest in underwater robotics 6

2) Internship Presentation 7

a) Course of the internship 7

b) Internship topic 7

II / Program structure 9

1) Description 9

2) Thresholding 11

3) Main problem 13

III / Feature tracking 15

1) Kalman filter 15

2) DBSCAN clustering 17

Presentation 17

Objective 17

Disadvantages 18

3) Solution 20

IV / Results 21

1) Processing frequency 21

2) Accuracy 22

3) Areas of improvement : 3 frames differencing 23

Conclusion 25

Appendix 25

References 26

Acknowledgements

This second year engineering school internship at ENSTA Bretagne was a rich experience, although it was carried out in teleworking. It was a perfect fit with my engineering student curriculum in Autonomous Robotics.
I would like to thank again the people who contributed to make these 3 months a very enriching period in terms of technical skills :

- Professor Karl Sammut from Flinders University, for accepting me for an internship despite the health crisis and for accompanying me during the internship during phone calls and email exchanges. He made sure to clearly define the subject of the internship in order to focus on a specific task.

- Professor Jonathan Wheare from Flinders University, for helping me with any technical questions I may have had and providing me with the necessary files, without which I would not have been able to work.

- Professor Benoît Clément, the referent professor from ENSTA Bretagne, who provided good advice during the team meetings and personal support during the setting up of the course.

- Zacharie Malicoutis, Head of International Mobilities at ENSTA Bretagne who was of invaluable help in the preparation of the trip to Australia (although it did not take place) and organisational issues.

- Larissa Pearse Manager at Flinders University, for answering my questions and editing the internship convention.

Introduction

The second year engineering school internship is an internship that fits perfectly with my studies at ENSTA Bretagne. It represents a key turning point in the construction of my professional project. It is an opportunity for me to apply the technical skills I learned during my second year in the field of autonomous robotics and to get an idea of the robotic engineer job. This internship was proposed by the *Nicolas Baudin* exchange program, offering French students internship topics in Australia and vice versa.

Passionate about mathematics and new technologies, robotics was a natural sector for me to work in. A sector at the heart of technological innovation, it is an integral part of the future of science. ENSTA Bretagne was a favourite choice after two years of preparatory classes, particularly for its field of maritime expertise. I could not have hoped for anything better than an internship in underwater robotics. Especially in the context of a global health crisis which prevented many students from doing an internship, and even more so if it had been done abroad. This internship was also an opportunity for me to work in a team with Aurélien Grenier, a third-year military engineering student at ENSTA Bretagne whose internship topic was close to mine and to put into practice professional English with Professors Karl Sammut and Jonathan Wheare during telephone meetings.

This report refers to the work carried out during my internship period from 25 May 2020 to 28 August 2020. Its aim is to give an objective account of the assignments carried out but also of the work to be continued in order to improve the results obtained. It then seems obvious to begin by presenting the context of the internship and the proposed missions, then to present the work done and the results.

I / Context

1) Presentation

a) Concept of an AUV and presentation of the X300 robot

An AUV for Autonomous Underwater Vehicle is an autonomous underwater robot whose vocation is to carry out underwater missions autonomously. The missions of an AUV are diverse and varied, they can be for example surveillance, recognition, detection, mapping ... [1] AUVs are recent robots at the cutting edge of technology because their purpose is to venture into hostile environments (humidity, high pressure, salt water, darkness ...) inaccessible to humans. Once the AUV's mission is over, it must be able to dock autonomously on a station provided for this purpose. This task is more complicated than it seems, as it requires the development of an internal guidance system for the AUV while it is circulating in an environment with currents, obstacles and little visibility.
The X300 underwater robot at *Flinders University Research Laboratories* is itself an AUV, here is a picture of it:

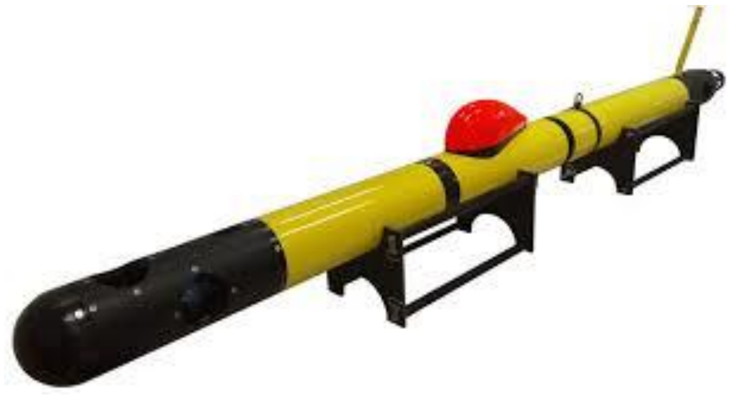


Fig. 1 X300 - AUV

The X300 robot is 3 meters long and 0.15 meters in diameter It is equipped with several sensors that allow it to navigate autonomously. Among these sensors is a high-resolution multibeam sonar to locate the docking station.[2]
Some work is done directly from the robot in the research laboratory at *Flinders University* in order to retrieve some of the recording files of the images taken by the multibeam sonar. On the other hand, some simulation work is done from the software commonly used in robotics, the *Gazebo* software. A model of the X300 robot has been created on this software as well as an underwater environment similar to that found in the real world. *Gazebo* then makes it possible to simulate the robot in a maritime environment while limiting material damage and the cost of the operation.

b) My interest in underwater robotics

The autonomous robotics speciality at ENSTA Bretagne was for me a key element in my decision concerning the engineering school to join after my two years of preparatory classes. ENSTA Bretagne is internationally recognized in this field, particularly in underwater robotics, a field in which it has won numerous awards:

- 1st prize in the European competition of autonomous underwater robotics (SAUC-E) in 2016.
- 1st and 2nd prize at the World Cup of Sailing Robots (WRSC 2018, 2015, 2014 and 2013). [3]

The courses given by the teacher-researchers Luc Jaulin, Benoît Zerr and Fabrice Le Bars of ENSTA Bretagne allowed me to acquire a solid technical basis in this field. Many fields are involved in this discipline (computer science, electronics, mechanics...) allowing me to always learn more. Autonomous robotics research being also a field of excellence of Flinders University, a partnership was born in 2019 between the two research centres. This cooperation agreement covers research, development and training, including internship programs offered in the academic laboratories of the two research centers. This was a dream opportunity for me, who wanted to do my second year internship in underwater robotics abroad. The technical skills required for the internship were perfectly compatible with those taught during the first two years of engineering school at ENSTA Bretagne. This allowed me to use them in the professional world.

During my engineering student course at ENSTA Bretagne in autonomous robotics I had the opportunity to work on several projects. In particular, I had the opportunity to code the school's DART robot in order to make it completely autonomous for a challenge between students. In addition, each year, the robotics students do an internship at the Lac de Guerlédan in Brittany in order to use their knowledge in autonomous underwater robotics. It was for me the first opportunity to work on such robots. I thus acquired various and varied skills in robotics allowing me, I hope, to have understood the main stakes of the internship carried out.

Here are some examples of robots that ENSTA's robotics engineering students have designed and are used to working on:

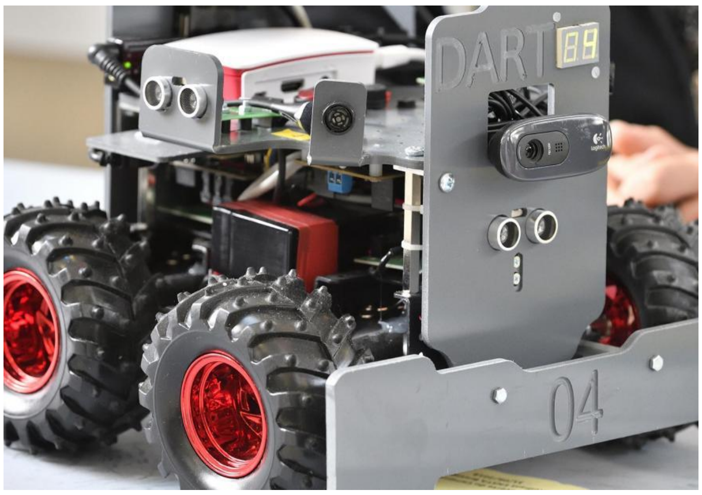


Fig: 4-wheel DART robot

Fig: Robot catamaran HELIOS

2) Internship Presentation

a) Course of the internship

The course took place in a particular context, that of the coronavirus 2020 health crisis. It was therefore initially intended to take place face-to-face in the premises of Flinders University in Adelaide, Australia. Then, little by little, several countries closed their borders, as well as France and Australia. It was therefore impossible for the French students who were doing an internship in Australia as part of the Nicolas Baudin program to do their internship in person. However, Professor Karl Sammut and his team of researchers were able to adapt so that we could do our internship from France by teleworking. As the time difference between the two countries did not allow us to work the same hours, I was able to organize myself to work on the usual working hours of the French, i.e. between 6 and 8 hours a day (morning & afternoon), 5 days a week. Videoconference meetings were held once a week on common working hours with the members of the project. These meetings were intended to present the work done last week and to explain what I planned to do the following week. Professors Sammut, Wheare and Clément, who were present at the meetings, allowed me to refer to the technical issues of the work to be done. They helped me decide which methods to use and advised me to contact certain people who were experts in a particular field. I would like to take this opportunity to thank them once again.

I also had the opportunity to work in a team with Aurélien Grenier, a military engineering student at ENSTA Bretagne doing his internship in the last year of engineering school, also in the research laboratory of Flinders University, teleworking from France. Although his internship subject is different from mine, he is working on the X300 robot. We were mainly able to help each other on the use of ROS, the middleware used in our two respective internships.

b) Internship topic

The field of autonomous underwater robotics is booming in the 21st century and requires expertise in many fields allowing the robot to carry out its missions autonomously. Sonar and video images are important sources of data allowing the robot to know its environment to obtain information during its missions but also to find its way around. For the X300 robot, researchers at Flinders University have opted to use an *Occulus* multibeam sonar rather than a camera for obvious reasons of visibility in an environment such as the seabed. The objective of my internship is therefore to detect and track a target using image sequences from bag files recorded in the Flinders University laboratories. The target to track is in my case, a docking cable allowing the robot to track it is to dock autonomously. Several tests have been performed to simulate the X300 robot in different contexts. Indeed, the cable can move in different ways (horizontally, vertically etc.), the different tests aim to represent the reality as much as possible. Here is an example of an image obtained during the acquisitions :

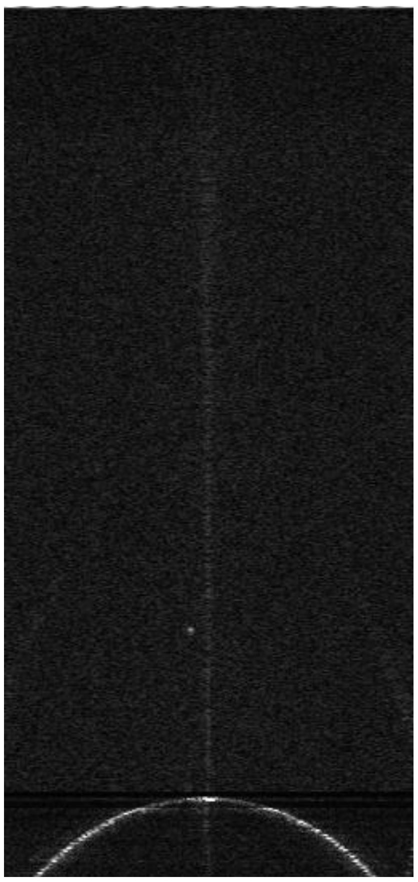


Fig: example of sonar images
(The bright dot represents the cable to be detected)

Target tracking is an essential field in computer-aided vision and is present in many fields (not only in robotics). In general, target tracking is the estimation of the position and movement of one or more parts of an image within a sequence. There are several methods of target tracking. The one used in our case is to follow the contours of characteristics of an image. This is because the regions of the image of interest (around the cable) have certain visual characteristics that make them clearly identifiable regions.[4] These characteristics are followed in each image of the sequence and thus form a trajectory. In order to be able to detect these particular regions, we will need the help of an artificial vision library. I chose OpenCV [5], an Open Source library known for its efficiency in this field. Although there are OpenCV modules that allow us to perform certain tasks that will be useful to us, we will first have to process each image with several filters as we will see later.

II / Program structure

1) Description

My main goal is to implement a computer program to read each of the images and thus detect the target. For this I had to choose which programming language and which libraries to use. So I decided to code with the python language for several reasons:

- its ease of use and compatibility with the different libraries to use.
- its large community on the internet.
- the many courses I have had on this language.
- the speed of execution which is sufficient for the work required (computation frequency higher than 30 Hz, we will review this point later in the report)

Concerning the libraries used, I chose OpenCV for image processing for the reasons mentioned above. The classical python libraries: numpy, matplotlib, time. But also libraries allow to make the link between ROS and python middleware such as : *cv_bridge* and *rosbag*. And finally, Machine Learning libraries for the clustering algorithm: *scikit-learn* and *pandas*. These python libraries are very easy to use and complete. We don't find similar ones with other languages such as C/C++, another reason to choose python as a programming language. The computer program takes as input parameter a bag file containing the recording of successive sonar images made in the Flinders University laboratories. So I have 4 bag files with different cable movements each time. This is a good basis for the implementation of the first algorithms. I will discuss the key points of the algorithm in specific parts. However, I will explain the overall structure of the program for processing each image in order:

- Reading images from a bag file: I use the *rosbag* module and the *read_message* function to read each image published on a ROS topic. Then I use a function from the *CvBridge* library, *imgmsg_to_cv2* which converts each image into a numpy array with a value for each pixel, a so-called classical way to store an image in a computer.

- White filtering: This filter is online and reduces sonar noise by averaging the pixel values in each row and column and applying a threshold to darken areas of the image that are too bright. Example of white filtering on one image (before filtering and after filtering) :

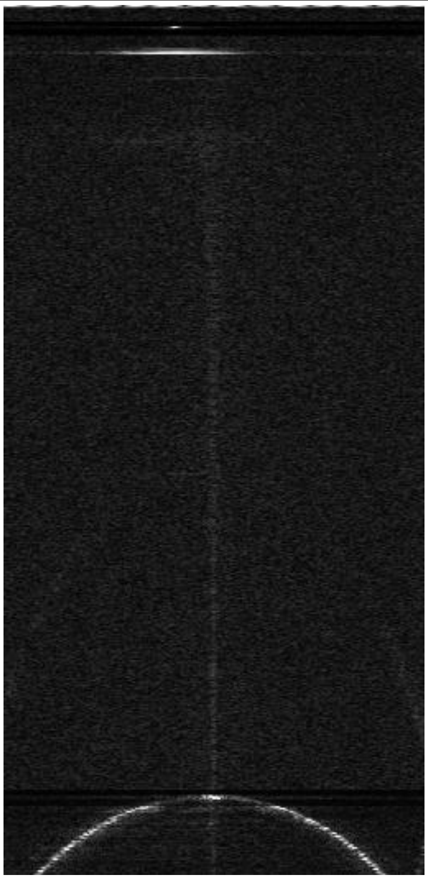


Fig: Before white filtering

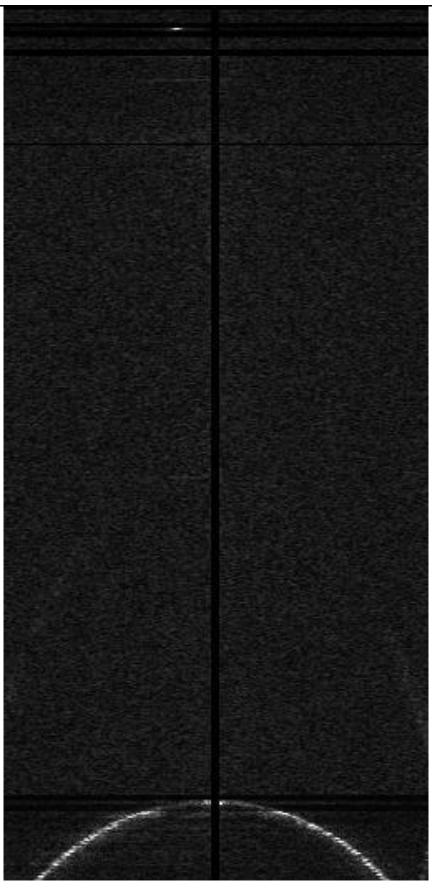


Fig: After white filtering

- Threshold: This method is widely used in computer image processing because it allows to detect regions of the image according to their colors. This is very practical in our case because the major part of the image is black while the cable is in the form of a lighter point. We will come back to this part in the rest of the report to explain it in more detail.

- Find contours: Once these preprocessing operations are done, we use the *findcontours* function of OpenCV which allows using gradient calculation to detect the contours of a region that is different from the others. [6]

- DBSCAN Algorithm and Kalman filter: These two filters will be useful to reduce image noise and track the target. We will discuss these two algorithms in a section dedicated to each of them.

- The display: This part intervenes at the end of the processing of each image and allows us to display a white point at the estimated position of the cable by the algorithm and a rectangle of 15 pixels * 15 pixels which corresponds to a certain degree of confidence (we will see it in the last part). The program indicates the coordinates (X,Y) of the detected point as follows (knowing that the origin is in the upper left corner of the image) :

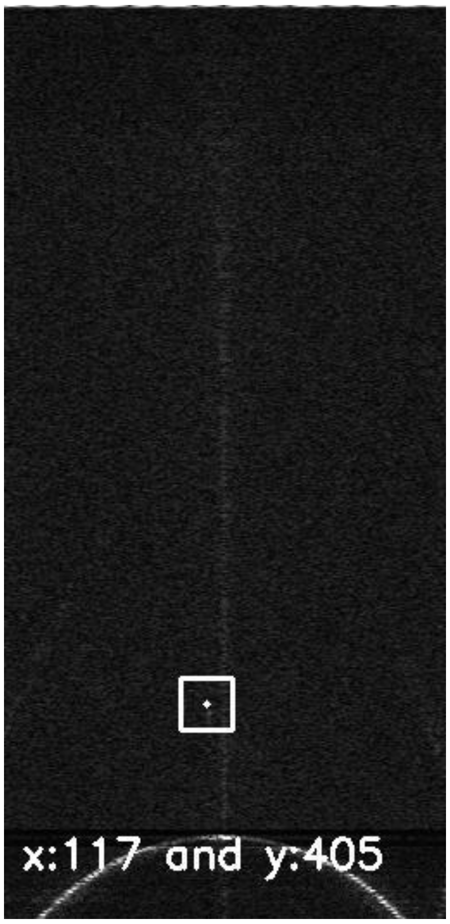


Fig: Final representation of one image

This process is then repeated for each image.

2) Thresholding

As explained earlier, one of the most common operations in image processing is the pre-processing. Several steps can be performed, including thresholding to improve the image and make it more usable afterwards. Remember that an image is represented by an array of pixels where each pixel has a value. The closer the pixel value is to 0, the darker the pixel appears, and conversely with a pixel close to 255. The principle of thresholding is therefore to replace one by one each pixel of the image thanks to a threshold value. [7]

Here is for example a classical image obtained before any processing :

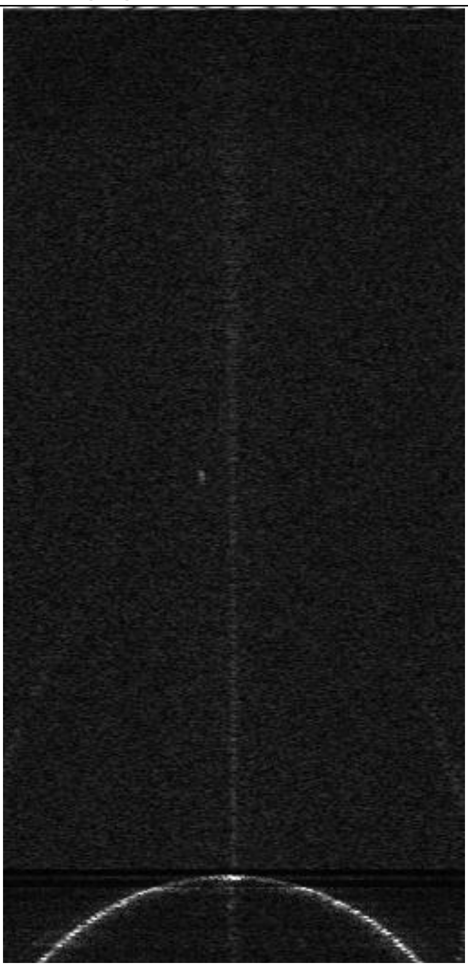


Fig: pre-threshold image

We notice on this image that there are dark regions (pixel value close to 0) and bright regions (pixel value close to 255). However, the distinction between the two may seem complicated. This is why we will use a thresholding method. The principle is then to further distinguish the separation between dark and light areas. For this we will apply a binary thresholding. After the processing of this filter, the pixels of our image will have only 2 possible values. Either 0 if the pixel is lower than the value of the threshold, or 255 if it is higher. All that remains is to arbitrarily choose this threshold value, which must be neither too large (otherwise we risk losing information), nor too small (otherwise the filter is not efficient enough). So I tested several threshold values and I kept the one that allowed me to have optimal performance on all the received files. This value is : 80.

However, it must be understood that only the thresholding will not allow us to detect only the cable in the image, because we notice on the image above that the color of a pixel is not enough to detect if it is the cable or the noise because several regions of the image have almost the same pixel values. Moreover, the cable does not always have the same light intensity during recording, so an intermediate value must be taken which best suits the situation. Here is the thresholding rendering on the same image as shown above for a threshold value equal to 80.

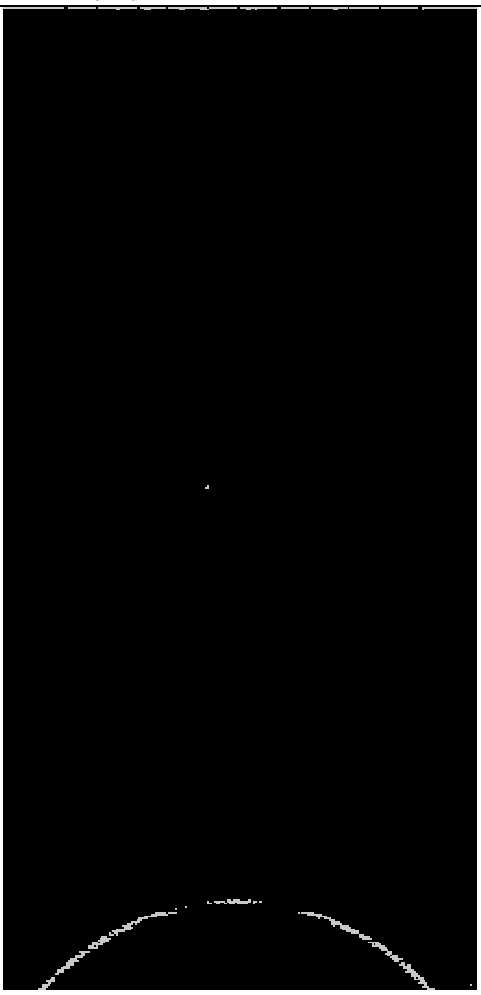


Fig: post-threshold image with threshold = 80

We notice that the point representing the cable still appears in the image (in the middle) but there is still a region (the lower part) with the same pixel value.

Concerning the implementation of this method in python, an *OpenCV* function, called *threshold*,[7] allows us to realize it. It must be given in parameter the image to be processed, the value of the threshold, and the type of thresholding (binary thresholding in our case). This function returns the thresholded image.
This first filter thus makes it easier to identify the cable but is not sufficient to determine its exact position, we will see why below.

3) Main problem

It might be tempting to think that only a threshold filter can be used to detect the docking cable. In absolute terms, if the data acquired by the sonar were perfect, i.e. noiseless, it could work. However, this is not the case; every signal is always more or less noisy. In addition, there could be other objects other than the cable in the sonar's field of view, so you would have to be able to detect the cable from what is not. Here is an example of an acquisition of a noisy recording:

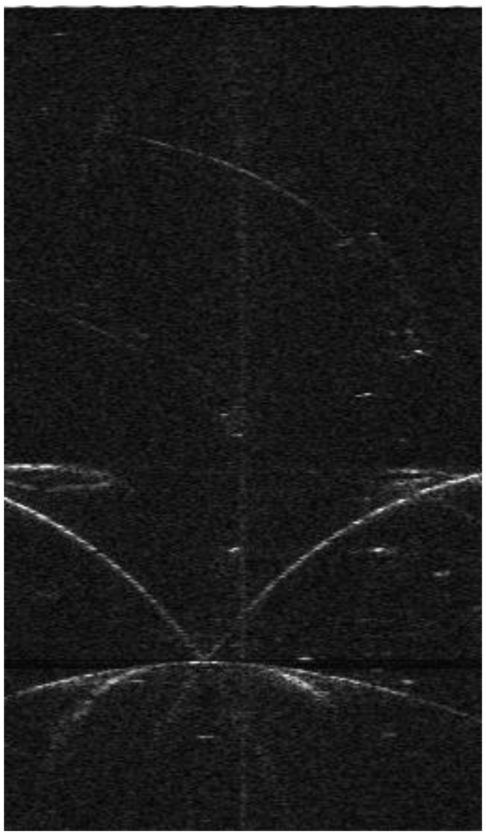


Fig: Image before thresholding

Fig: Image after thresholding

In this example, we see that after thresholding the image, although some points are highlighted, it is impossible to detect the docking cable. This is due to the fact that several pixels have a value too close to that of the cable to be able to differentiate between them. This does not mean that the thresholding method is not effective, but it is not sufficient. It is therefore necessary to find a complementary detection method.

Until now, the physical characteristic of the cable has been used as a basis (e.g. its brightness), but it has other characteristics. Indeed, the cable and the robot are in motion in relation to each other. The movement of the cable in relation to the robot therefore draws a trajectory. The following example shows a graph plotting all the contours detected over the total duration of an acquisition with only a thresholding method as image processing:

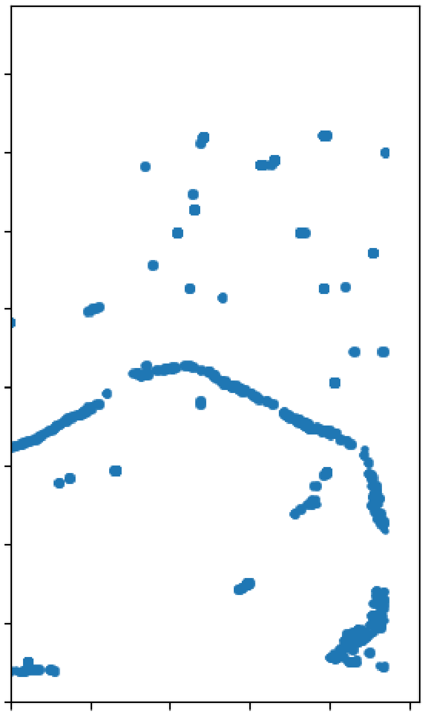


Fig: Contour display detected on successive frames with thresholding only

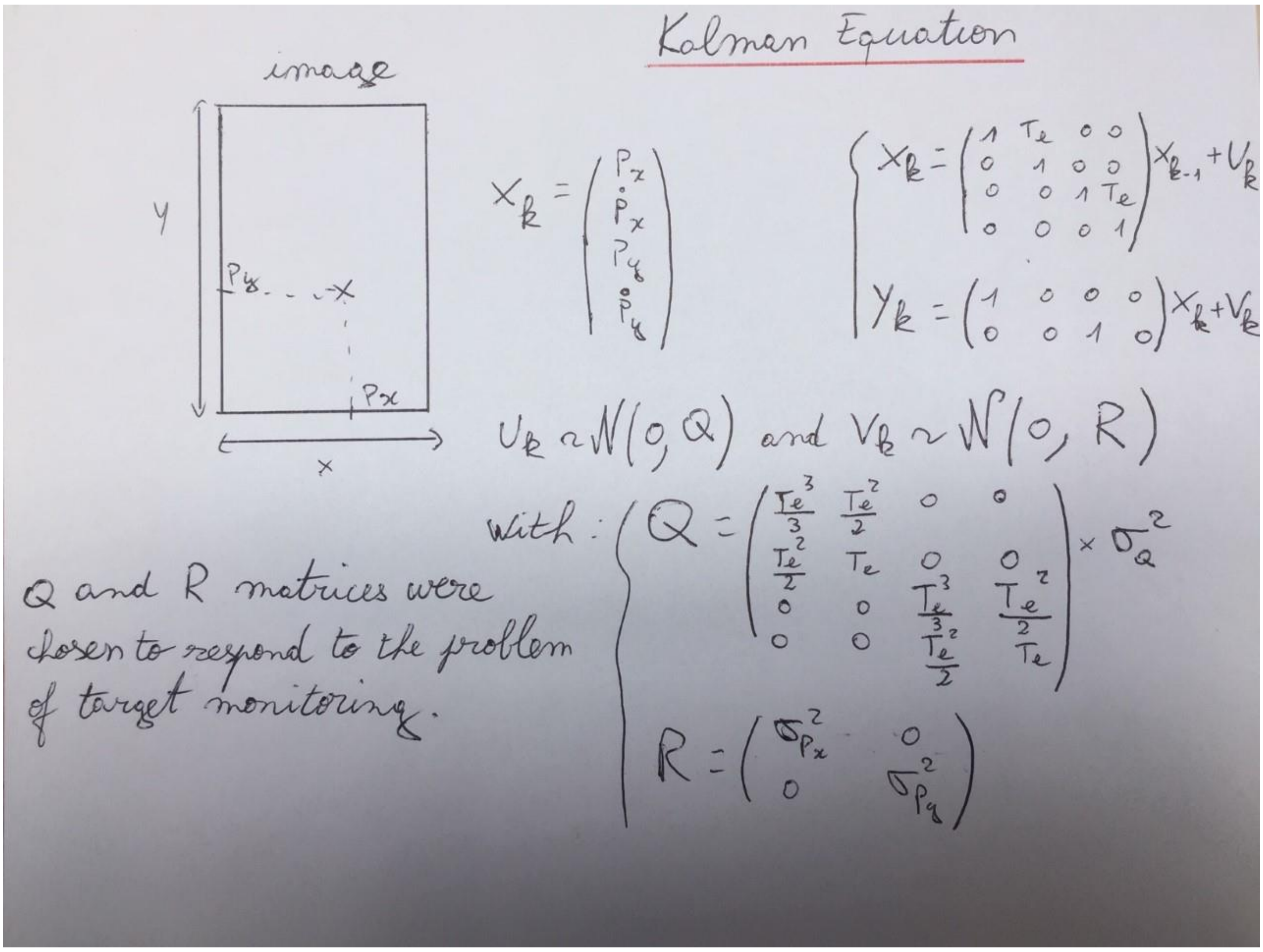
This time it is easier to detect the position of the cable because its trajectory is noticeable. The noise doesn't draw any trajectory so it is easy to identify if a point is in the cable's trajectory or if it represents noise. This characteristic explains why we are going to set up other detection methods that we will see in the next section that are based on the principle that the cable moves in relation to the X300 robot.

III / Feature tracking

1) Kalman filter

The Kalman filter is a filter commonly used in many scientific applications such as target tracking. This filter estimates the states of a dynamic system from a series of incomplete or noisy measurements. For example, in the case of radar target tracking, information such as target position and velocity is recorded at all times but with many approximations due to sonar noise and experimental errors. The Kalman filter will therefore use the dynamics of the target to obtain more accurate data. In our case, the algorithm will both compute this data in the present time, which is called filtering, and in the future, which is called prediction. Indeed, two distinct scenarios will oppose each other in our case:
Either the algorithm detects an approximate position of the cable and the Kalman filter will predict its position and speed from the current image to get an improved accuracy on the state of the system. (Filtering). Either the algorithm misses the detection of the cable and the Kalman filter will have to predict the state of the system based on the information from the previous images. (Predicting) [8]

To do this, we model the system in the form of 2 equations, called the state equation (target dynamics) and the observation equation (sonar measurement). Since our problem is in a two-dimensional plane, the state vector is a four-dimensional column vector with the positions and velocities in both dimensions of the plane as components. Here is a summary of the equations obtained in the case of a two-dimensional target tracking [9] :



Kalman Equation

$$X_k = \begin{pmatrix} P_x \\ \dot{P}_x \\ P_y \\ \dot{P}_y \end{pmatrix}$$

$$\begin{cases} X_k = \begin{pmatrix} 1 & T_e & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T_e \\ 0 & 0 & 0 & 1 \end{pmatrix} X_{k-1} + U_k \\ Y_k = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} X_k + V_k \end{cases}$$

$$U_k \sim \mathcal{N}(0, Q) \text{ and } V_k \sim \mathcal{N}(0, R)$$

$$\text{with : } \begin{cases} Q = \begin{pmatrix} \frac{T_e^3}{3} & \frac{T_e^2}{2} & 0 & 0 \\ \frac{T_e^2}{2} & T_e & 0 & 0 \\ 0 & 0 & \frac{T_e^3}{3} & \frac{T_e^2}{2} \\ 0 & 0 & \frac{T_e^2}{2} & T_e \end{pmatrix} \times \sigma_Q^2 \\ R = \begin{pmatrix} \sigma_{P_x}^2 & 0 \\ 0 & \sigma_{P_y}^2 \end{pmatrix} \end{cases}$$

Q and R matrices were chosen to respond to the problem of target monitoring.

Fig : Kalman equation for target tracking

These two equations show two things:

- The first is that the position of the target at time k is the position of the target at time k-1 plus its velocity and a null mean term corresponding to the noise.
- The second is that the information observed with sonar is the target position at time k plus a zero mean error term as well.

The *Q* and *R* matrices are chosen according to parameters determined by the problem such as its period and the amount of noise.

In the rendering algorithm, the Kalman filter is decomposed into 3 functions :

- *kalman_correc*: function which calculates the corrected estimate and the corrected covariance matrix of the system as a function of the estimate at time k, the observation vector and the system-specific matrices.

- *kalman_predict*: Function that calculates the predicted estimate and covariance matrix based on the corrected estimate and covariance matrix.
- *kalman*: Function that combines the above two functions to establish the Kalman filter as a whole.

For each sequence, if after the thresholding method, the algorithm detects a possible position of the cable, then it is sufficient to apply the above Kalman filter with an observation vector, a two-dimensional column vector representing the position *(X,Y)* of the cable. If this is not the case, just tell the system to take the last observation in memory as the observation of the current image.

2) DBSCAN clustering

a) Presentation

The DBSCAN algorithm is an unsupervised machine learning method. It allows several groups of points to be grouped into distinct clusters according to their properties. In our case, it is the positions of the points that define their membership to a group or not. There are several clustering algorithms such as: K-Means, K-Medoids, CLARA... [10] These algorithms all have particular interests, here is the one of DBSCAN and why it could fit our problem :
The algorithm looks for groups of nearby points. To do this, it searches, for each point, the points that are part of its immediate vicinity, and it groups together all the points that can be reached from one point to another. Once all the points are grouped together, the groups of points are considered either as clusters or as outliers, if the number of observations is below a certain threshold.[10] The big interests of this algorithm compared to other clustering algorithms are that it can adapt to all forms of clusters, that it allows to classify a point as noise and that it adapts to large dataset.

To implement this algorithm we need three parameters: epsilon, min_smaples and metric.[10]

- epsilon: 2 points are considered neighbors if their Euclidean distance is less than the epsilon distance. This parameter is easy to set because on all the received bag files, the distance between 2 consecutive frames from the point representing the position of the cable varies very little depending on the bag file.

- min_smaples: The minimum number of neighbors a given point should have in order to be classified as a core point. This parameter will allow the algorithm to cluster only regions with a large number of points. The noise being almost random, if we choose this parameter large enough, the algorithm will remove this noise.

- Metric: The type of distance to be used. In our case, it is the Euclidean distance.

b) Objective

The main problem being that at each frame (for a noisy bag file), the classical thresholding algorithm detects about 10 points, 9 of which represent noise. The objective is to keep in memory the successive position of the detected points on some frames in order to apply the DBSCAN algorithm and thus to identify the noise and remove a part of it. In order for the algorithm to take into account the temporal aspect of the simulation, one possibility is to perform the DBSCAN algorithm on a sliding window of successive frames. For example take successively the first 10 frames then the frames from 1 to 11 then from 2 to 12 then from 3 to 13 and so on... It is necessary to try several sizes of sliding window like this one to obtain the one which adapts to the problem. The goal of this algorithm is to reduce the noise considerably then to use the kalman filter already implemented for the target tracking.

Here, then, is an example taken from one of the given bag files :

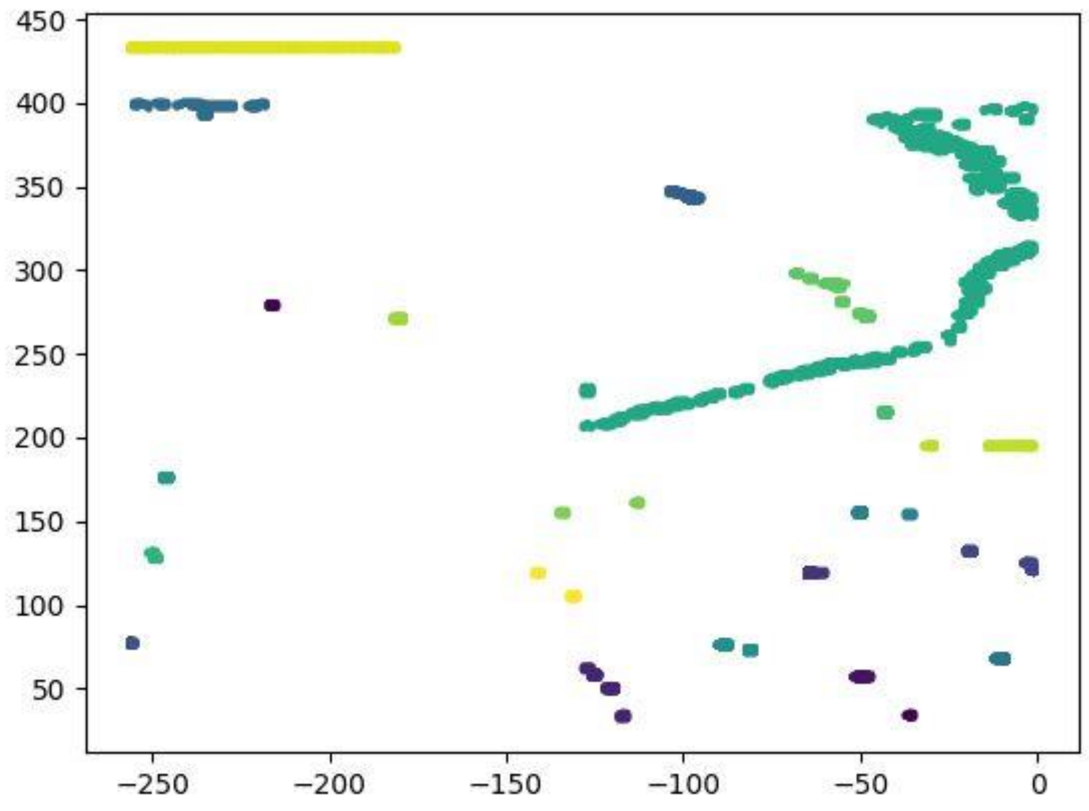


Fig : Image displaying the successive position of all detected points over about 100 frames with the DBSCAN algorithm.

The blue/green part represents the successive position of the cable, the rest being either a cluster with fewer detected points or noise. With this example we notice that in the next frame, when the main algorithm will have detected about ten points, it will be able to assign them to such or such clusters or assign them to noise.

c) Disadvantages

In this part, we will see why this algorithm was not retained in the final program and what the reasons are. Indeed, in order to decide whether a target tracking algorithm is efficient or not, we need to compute precision measures in different situations. So I decided to calculate 3 performance indicators for this algorithm which are :

- Frequency (F) = Image processing frequency.
- Accuracy (A) = Percentage detection of the point in a square of 15 pixels * 15 pixels (error less than 4%).
- NF = Number of Frames required to detect the first position of the cable.

So I tested the DBSCAN algorithm with different input parameters such as the number of successive frames for the sliding window and more or less noisy files. Here are the results:

Experiments	Kalman + Clustering (3 frames)	Kalman + Clustering (10 frames)
Non-noise files	Accuracy = 95% of the time in the 15 pixels square around the cable. Frequency = 40Hz NF = 3 frames to detect the cable	Accuracy = 93% of the time in the 15 pixels square around the cable. Frequency = 20Hz NF = 10 frames to detect the cable
Noisy files	Accuracy = 35% of the time in the 15 pixels square around the cable. Frequency = 20Hz NF = around 30 frames to detect the cable (1.5 second)	Accuracy = 45% of the time in the 15 pixels square around the cable. Frequency = 12 Hz NF = around 40 frames to detect the cable (3.5 second)

We notice that for noise-free files, the precision is relatively high, but the addition of several frames in the sliding window considerably reduces the calculation frequency. (In particular, the frequency must be above 30Hz). But it is the noisy files that interest us in our case because it is to reduce the noise that we thought of DBSCAN. We see that the accuracy does not even exceed 50% and that the calculation frequency is very low. This is due to the fact that there is a lag between the prediction by the Kalman filter and the DBSCAN algorithm. If the number of successive frames is too large, the algorithm does not detect the rapid variations of the cable and if it is too small, it is not sufficient to reduce the noise significantly. This algorithm is therefore not sufficient to solve our problem, however there are other types of clustering algorithms that might be interesting to try. In our case, we had to find a solution to increase the precision in the case of noisy files. We are going to see the one that has been retained in the following part.

3) Solution

As we have seen, a simple kalman filter on the points detected after the thresholding method is not sufficient to achieve sufficient accuracy on detection. So we had to find a solution.
For that we notice that on a given sequence, noise is present on the same pixels of each image. This noise corresponds to the noise of the sonars which is fixed on all the images of the sequence as shown in the example below :

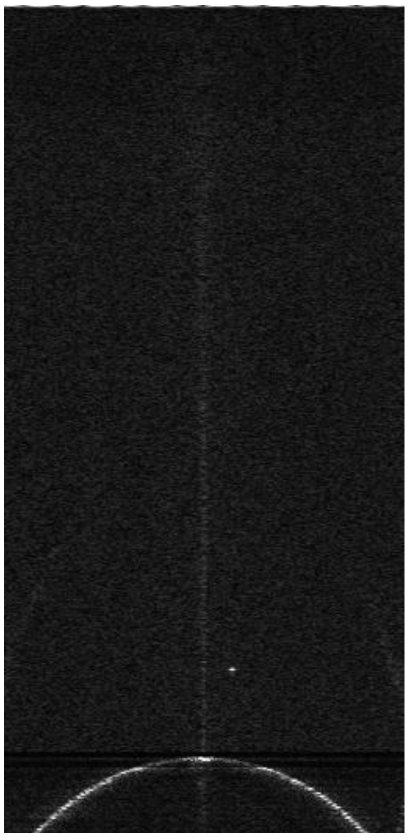


Fig : Frame 10

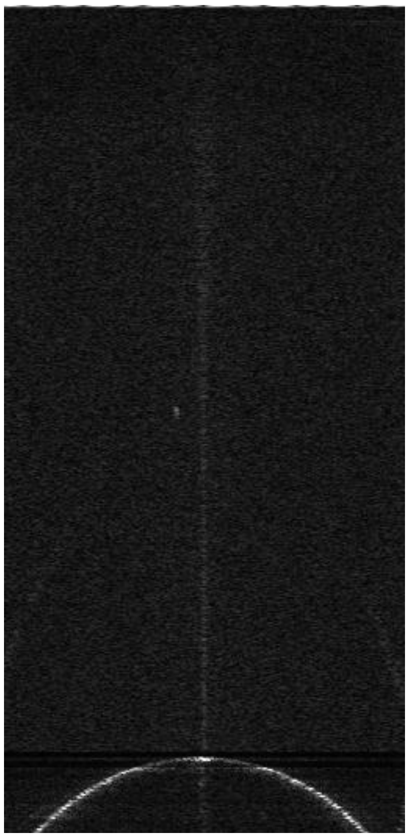


Fig : Frame 150

This noise is very constraining because it limits the detection of the cable. For this reason we will make comparisons between the images. First, we record all the detected points of the first image after the thresholding. Then, for each processed image, once the threshold is applied, we compare the position of the detected contours with those of the first recorded frame. If we find the same positions for a point, then we delete it because it corresponds to noise. This operation is expensive in computing time but it allows to remove more than 80% of the noise of an image. Here is an example of all the contours detected on the first image of the bag file :

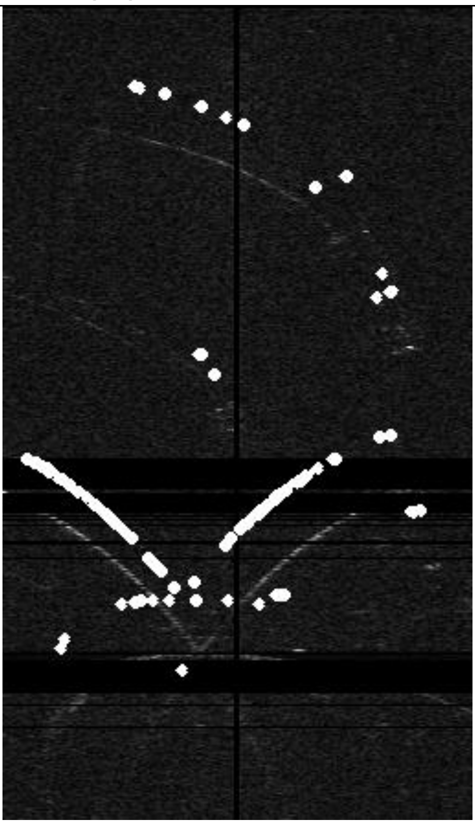


Fig : Set of contours detected during the first frame

In subsequent images, if the position of an outline is the same as one of the previously recorded positions in the first image then it will be deleted. This method removes the overall noise of the sequence, that is, the noise present in each image is removed. But there is still some temporary noise that is randomly added to each image which also interferes with detection. Although this noise is minimal compared to the overall noise, it still interferes with detection.
For the removal of the temporary noise, we rely on the Kalman filter already implemented. It is sufficient to compare the estimated position and the observed position of each detected contour for each successive image. We then keep the contours whose estimation corresponds closest to the observation. This method allows to remove the remaining noise and thus to detect the real position of the cable after having performed the first method described in this part.

Iv / Results

1) Processing frequency

In order for the X300 robot to have enough time to process each of the images returned by the sonars, the frequency of the cable detection program must be at least 30Hz, i.e. process and therefore detect the cable at 30 images per second.

The frequency of the programme depends on several parameters, including the ease of detection. The clearer the cable appears on the screen, the easier it is to detect it. Indeed, when the file is noisy, i.e. when other "white dots" similar to that of the cable appear on the image, the algorithm must activate several filters based on a Kalman filter. These filters are costly in terms of calculation time and the more noisy the file is, the lower the frequency will be.

When the cable follows a linear trajectory (horizontal or vertical for example), the file is often low noise, which leads to a detection frequency higher than 100Hz, which is more than enough in our case. However, when it follows a more random movement, the frequency can go down to around 20Hz. Note that these results were obtained with a computer with a CPU: *Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz* with 8 processors and a program written in Python. The results obtained with a GPU will be better, especially if the program is translated into C/C++, which is quite possible because the libraries used exist in these languages. In conclusion, the frequency of an object detection algorithm applied to the X300 robot will not be a problem if it is optimized in the case of noisy files.

2) Accuracy

In order to evaluate the quality of the rendered algorithm, it is essential to calculate beforehand precision measurements on detection. For this purpose, 2 measures have been calculated:

- Accuracy: The number of times the algorithm detects the cable in a square of 15 pixels side whose center is the true position of the cable. The value of 15 pixels has been chosen to represent an error on the accuracy of less than 4% of the total image.

- Number of Frames (NF): Since some files are noisy, the algorithm may sometimes take a while to "remove" noise by calculating a kalman filter on each point detected at the beginning of the file playback. To do this, it is interesting to calculate the number of frames necessary for the algorithm to be able to detect the cable in a 15-pixel square at the beginning of the file's reading. We will call this measurement: Number of Frames (NF). To calculate this measure, I have at my disposal 4 different files, more or less noisy and with different movements in relation to the cable. I will therefore list these results in a table.

	Bag file 1 (unoised)	Bag file 2 (unoised)	Bag file 3 (unoised)	Bag file 4 (noisy)
Accuracy	97%	95%	92%	60%

Number of Frames (NF)	1	1	1	24 frames = 1.2 seconds

We notice that for noise-free files, the algorithm detects the cable from the first image and that the accuracy is on average 95%. However, for a more noisy file, the accuracy decreases and the time to detect the cable for the first time comes after about 1 second, the time the algorithm records several kalman filters on several detected points to separate the noise from the real cable.

3) Areas of improvement : 3 frames differencing

In order to considerably reduce the detected noise there is a method: the Three-Frame Differencing-Algorithm. This method has been developed to improve the performance of target tracking algorithms. [11] This method is based on the comparison of 3 successive images which we will call hereafter: k-1,k and k+1. To do this, two binary images must be obtained from the difference between the pixels of the k and k-1 images as well as the k and k+1 images. [11]. This method requires the choice of a threshold value according to the given problem. Once the two binary images have been obtained, the last image called: three frame difference image has to be computed. The operations between all the images are very well detailed in the article: *Three-Frame Difference Algorithm Research Based on Mathematical Morphology* by Yanzhu Zhang, Xiaoyan Wang and Biao Qu. [11].
So this algorithm allows several things:

- It reduces the effect of strong changes in the external light.
- It removes a good part of the noise.
- Detects the moving object.

The objectives of this algorithm meet the needs of the cable detection algorithm developed during this course, so it was natural to apply it to our case. So I developed a Python algorithm which, from 3 successive images taken in a given bag file, to compute the three frame difference image and compare it to the only thresholded image. Here is the result :

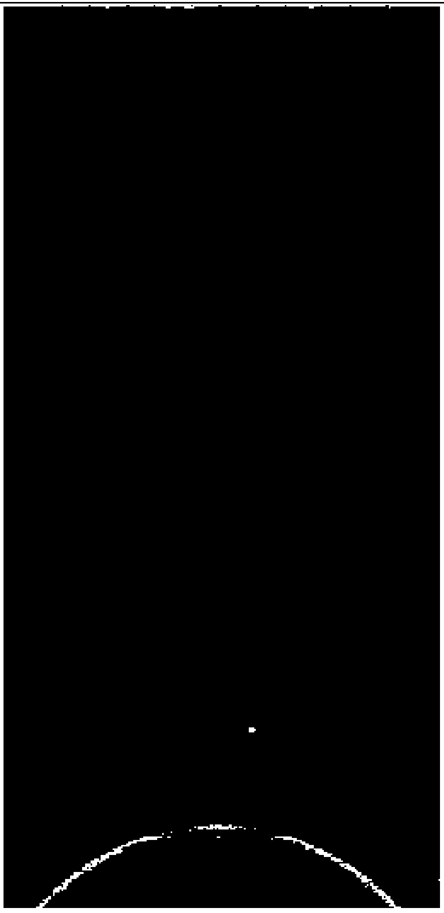


Fig : Thresholding image

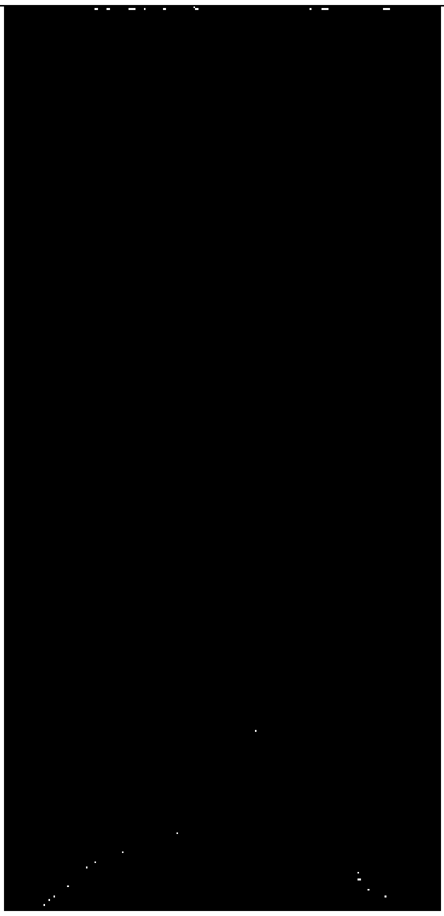


Fig : Three frame difference image

We notice that this method allows us to remove a large part of the fixed noise while detecting the moving object (we can see a white point on the three frame difference image at the place where the cable is located). At first glance it would be very interesting to implement this method in our target tracking algorithm. However, this method requires a lot of computing time. Indeed to apply to a single image, the three frame differencing method, the laptop I'm working on requires about 2 seconds of computation (0.5Hz). Knowing that the minimum expected frequency is 30Hz, this method is far from satisfying the criterion of speed. The low frequency of this algorithm comes from the 3 loops nested in the developed code to compare each pixel of each frame. For the moment this method is therefore not feasible. However, with the installation of a GPU and the improvement of the code optimization, it could become so. The three frame differencing method also has other limitations:

- If there is too little distance between the moving object on the 3 successive frames, the algorithm may not be powerful enough to detect it.
- This method requires the use of a predefined threshold value and therefore does not adapt to all problems simultaneously.

Conclusion

This internship has been a real opportunity for me to acquire many skills in various fields such as robotics, artificial vision and learning English. I am aware that given the health context I am lucky to have been able to do a second year internship during my training course at ENSTA Bretagne. This is why I would like to take this opportunity once again to thank all the people who contributed to the smooth running of this internship; in particular Professor Sammut from Flinders University.

The field of target tracking in robotics is a growing field in which many techniques have already been developed. I tried to choose and implement those that seemed to be the most appropriate for the problem. Although the rendered algorithm is functional and the detection accuracy is satisfactory in the easiest cases (rectilinear movement of the cable with respect to the X300 robot), there are still many areas for improvement. Several techniques have been used and described in this report (thresholding operation, Kalman filter, clustering, three frames differencing...). The documentation on this subject is vast on the internet, so it is easy to discover new methods to improve the tracking of a moving target.

Concerning the further work to be done, it remains to test this algorithm on a simulator such as Gazebo, having first developed the driver of a sonar similar to the *Occulus*, a sonar used in the laboratories of Flinders University, from which the bag files used were recorded.

Appendix

In this part we will see some technical tips on how the program is structured, how to run it, how to modify it, the versions used etc. The final render is therefore composed of a single Python script (*Python 2.7.16*).

The libraries used are imported at the beginning of the program and are the following:

- rosbag : rosbag
- cv_bridge : cv_bridge
- cv2, version 4.2.0 : OpenCV
- numpy, version 1.16.5 : Numpy
- time, version : Time

The program consists of 14 functions, each with a distinct role. These functions are written in the upper part of the program and are all documented and commented to make them easier to understand. One of these functions is a main function, which is the function that starts first when the program is executed. The program takes as parameter only one argument on which the user has the right to choose. This argument corresponds to the name of the bag file stored in the same folder as the program. During my internship I had the opportunity to implement the algorithm using 4 bag files. In order to choose one of them rather than another, you just have to uncomment its name at the end of the program (it is indicated as a comment in the program). The next time you register in the Flinders University labs, you just have to add the file in the right folder and add its name at the end of the program. I also decided, for viewing clarity reasons, to display each image on the screen for 40 ms. This duration can be modified by changing the value inside *key = cv2.waitKey()* on line 377. It is also sufficient to press the letter "q" on the computer keyboard to interrupt the program in progress.

References

https://oceanservice.noaa.gov/facts/auv-rov.html [1]
https://www.udt-global.com/__media/libraries/draft-abstracts--slides/52-Karl-Sammut.pdf [2]

https://www.ensta-bretagne.fr/en/autonomous-robotics-0 [3]
https://medium.com/visionwizard/object-tracking-675d7a33e687 [4]
https://docs.opencv.org/2.4/index.html [5]
https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours [6]
https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html [7]
https://www.mathworks.com/help/vision/examples/using-kalman-filter-for-object-tracking.html [8]
http://www-public.imtbs-tsp.eu/~petetin/TP_Kalman.pdf [9]
https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html [10]
Three_frames_differencing_image [11]

Internship presentation : Feature Tracking in Video and Sonar Subsea Sequences with Applications