DEVELOPMENT OF A COMPUTER VISION SYSTEM FOR USE IN SUBMARINE ROBOTS

Manuel Faysse

DIRECTOR DEL TRABAJO FIN DE GRADO:

Antonio Barrientos Cruz,
Andrés Martín-Barrio

JULIO 2018
“On peut braver les lois humaines, mais non résister aux lois naturelles.”

“We may fight human laws, but we cannot resist the ones set by nature.”

**Jules Verne** — “*Vingt mille lieues sous les mers*”
ACKNOWLEDGEMENTS

This project taught me the bases of computer vision and machine learning, fields I had always wanted to experiment with. However, through the course of the project, I also acquired some valuable project management skills, such as not hesitating to ask for help when needed, and the value of good time management. It truly was a great opportunity for me to be able to work on such a project and I would like to thank Antonio Barrientos for presenting me with it.

The person I’d most like to thank is Andrés Martín Barrio. He offered me constant help, was always very understanding and sympathetic and overall was a great TFG tutor. I am also amazed by the kindness of Leo Gómez who twice made the effort to come after work across Madrid to help me solving the hardware problems I encountered without asking for the slightest thing in return.

Finally, I would like to thank Jorge Poveda, Diego Moratilla, Evelio Garcia Martín and Alberto Huerta for answering the different questions I had about the project, helping me gather material, and being great friends.

¡Gracias a todos ustedes, ha sido un placer conocerlos a todos!
Natural or artificial disasters such as the nuclear power plant failure at Fukushima in 2011 reveal the ever-growing need and potential of autonomous vehicles and robots to assist in emergency response protocols. Quicker, more efficient, safer, the capabilities of such technology is huge and humans are bound to increase their reliance on them in the near future. This end of degree project (TFG) focuses more specifically in the utilization of computer vision for underwater rescue missions carried out by autonomous submarine robots. This paper presents a series of computer vision methods and algorithms implemented for subaquatic object detection and identification, as well as the results of conducted experiments in different conditions and ideas for future improvements.
Executive Summary

**Computer vision**, term referring to the analysis and understanding of images and video by a computer system is now a scientific research field over 50 years old. Recent improvements in camera quality and computer speed, new access to huge image databases through the internet, and revolutionary new techniques such as Convolutional Neural Networks have however given a new life to the field and these past few years have seen huge amounts of new related applications and improvements.

One specific application to the use of computer vision is the control of autonomous vehicles and robots to assist humans in tasks that are deemed too difficult or unsafe for human workers. Such tasks include the guidance of emergency response autonomous vehicles in the case of natural or artificial disasters such as nuclear power plant failures.

![Figure 1: Robot sent in the Fukushima plant after the disaster](image1)

The poor performance of autonomous vehicles during the Fukushima power plant failure of 2011 (*Figure 1, Kawatsuma 2012*) inspired the creation of a robotics international competition, the European Robotics League Emergency Challenge that aims to improve current emergency robots. In this competition, robots moving on land, in water and in the air have to interact and cooperate to complete several challenges a real life emergency situation may pose.

![Figure 2: SPARUS II robot used in the ERL competition](image2)
This TFG focuses on the development of computer vision algorithms for the underwater autonomous vehicles (AUV) used in the competition. The objective is to allow the submarine robot (SPARUS II) (Figure 2, IQUA Robotics 2016) to correctly detect different types of objects that it could be faced with during real emergency rescue situations. For success in the ERL, the robot first has to detect orange signalization buoys numbered from 1 to 9. They are used as points the robot has to locate on an internal map, or as landmarks for the robot to estimate its position. Another of the robot’s mission is to be able to spot a human shaped mannequin (Figure 3, ERL Rulebook 2017) representing a human worker trapped underwater. Finally, the robot has to detect underwater pipelines and inspect them for leaks and failures. The pipelines in the competition are made to simulate the pipes used in Fukushima and similar plants to transport seawater to nuclear reactors as a refrigerant.

In order to develop the algorithms during the span of this project, a replica marine environment was rebuilt at a smaller scale using an aquarium. Every object is also proportionally scaled down; the orange signalization buoys are replicated with orange ping-pong balls, the pipes used in the competition are replaced by smaller diameter PVC tubing, and finally the real scale human dummy is simulated by similarly shaped LEGO figures. The camera used for the experiments is the same as the one used by the AUV, a low-light analog camera that can capture colors even in dark situations. It is important to note the conditions were often less ideal to computer vision recognition than the real sea environment because of low lighting, the presence of unwanted edges of the aquarium, etc…

The computer vision methods used for the detection of each type of objects are very different and present different difficulties.

To detect the orange buoys a color detection method is used. Each frame of the video input is first preprocessed in real time with various filters (Histogram Equalization, Median Filter) in order to de-noise the image and improve the performance of the rest of the algorithm. The image is then converted into a different color model, HSV, and the orange color is isolated from the rest of the image. In function of the shape and size of the detected orange pixels, the
buoy is located, and a confidence factor for the detection is established by using past detection history.

The zone of the image in which the buoy is located is then filtered to isolate the potential black digit printed on the side of the buoy. When a digit is present on the filmed side of the buoy, the digit is isolated and classified from 0 to 9 using machine learning models. The digit’s dimensionality (amount of data needed to describe it) is first reduced through the technique of the Histogram of Oriented Gradients (HOG), then put into one of ten categories through Support Vector Machine classification.

The color detection of the buoys was established through the experiments to be very performant (Figure 4). Very disturbed and murky waters however degraded the detection rate because of poor propagation and restitution by the camera of the orange light waves. The digit recognition functions very well at close range but is also impacted by the murkiness of the water.

Pipeline detection was achieved through a series of methods based on edge detection. The image is first converted into grayscale and pre-processed similarly to the buoy. Using Sobel and Canny filters, the image’s edges are found, and Hough Transforms detect if straight lines exist amongst those edges. Using the information from the detected straight lines in the image, the computer determines the presence and position of the pipelines and builds a confidence factor based on past images.
This method of detection also yielded good experimental (Figure 5) results and was more robust than color detection faced with murky water. Disturbed water with quick moving floating particles did however cause the detection’s quality to severely drop because it caused the edges of the images the algorithms relies on to be distorted and broken.

Finally, human body recognition was achieved through machine learning techniques. After pre-processing, a small window is moved around the image to detect the presence of groups of pixels similar to a human body, using HOG feature vectors and SVM classification. The window’s size is then increased and moved around the image again. This “sliding window” process is repeated a few times. It is notable that this process is relatively slow and to be able to operate in real time, it is only done in periodic intervals instead of in every frame.

The results were very good (Figure 6) when the body was inclined along the vertical axis of the image and had sufficient size but detection was not performant with other body orientations (training data did not include them).

The work done in this project will hopefully have uses for performances in the ERL robotics competition but is above all a preliminary step in improving computer vision systems in emergency response autonomous vehicles.
# TABLE OF CONTENTS

1. INTRODUCTION.................................................................................................................. 10
   1.1 Motivations .................................................................................................................. 10
   1.2 Background ................................................................................................................ 11
   1.3 Objectives .................................................................................................................. 12

2. STATE OF THE ART ............................................................................................................. 14
   2.1 History of Computer Vision .......................................................................................... 15
   2.2 Applications of computer vision .................................................................................. 19
   2.3 Computer vision in subaquatic environments ............................................................... 22

3. COMPUTER VISION TECHNIQUES IMPLEMENTED ....................................................... 24
   3.1 HOG Feature descriptor ............................................................................................... 24
   3.2 SVM Classification ...................................................................................................... 27
   3.3 Color Detection .......................................................................................................... 30
   3.4 Noise reduction and image pre-processing ................................................................. 31
   3.5 Canny Filter .............................................................................................................. 34
   3.6 Hough Line Detection ............................................................................................... 35

4. BUOY DETECTION ............................................................................................................. 38
   4.1 Preprocessing .............................................................................................................. 38
   4.2 Color Detection .......................................................................................................... 39
   4.3 Morphological operations ............................................................................................ 40
   4.4 Confidence Evaluation with Buffer Technique ......................................................... 41
   4.5 Digit Detection .......................................................................................................... 42
   4.6 Digit Recognition ....................................................................................................... 43

5. PIPELINE DETECTION ..................................................................................................... 45
   5.1 Preprocessing .............................................................................................................. 46
   5.2 Canny Filtering and Line Detection .......................................................................... 46
   5.3 Line selection ............................................................................................................ 48
   5.4 Final Tube Detection .................................................................................................. 48

6. HUMAN BODY DETECTION .............................................................................................. 50

7. RESULTS ............................................................................................................................. 52
   7.1 Experimental Setup .................................................................................................... 52
   7.2 Problems encountered ............................................................................................... 53
   7.3 Buoy detection .......................................................................................................... 55
   7.4 Tube detection .......................................................................................................... 57
1. INTRODUCTION

1.1 Motivations

On March 11 2011, the biggest nuclear disaster since Chernobyl in 1986 occurred at the Fukushima Daiichi nuclear power plant in Japan. The combined destructive power of an earthquake followed by a tsunami caused the reactor’s cooling system to malfunction and ultimately led to nuclear meltdowns, explosions and the release of radioactive material (IEEE Spectrum, 2011).

The risk of explosions and radioactivity rendered the power plant sector unsafe for human rescuers and unmanned robots were deployed to get inside the plants and carry out reconnaissance tasks. (Figure 7, Kawatsuna 2012) The response was however considered too slow, due to a previous lack of communication between researchers, power plants and robot manufacturers. Poor previous maintenance of the on-site robots was also an issue (Kawatsuna 2012).

![Figure 7: Robot entering into reactor building](image)

Conclusions drawn from the disaster stated that the response could have been much improved with better autonomous robot support. Inspired by Fukushima, the European Robotics League (ERL) Emergency robot competition was invented (previously named euRathlon). The objective of this competition is to replicate Fukushima like conditions and challenge teams to develop different robots in order to solve different real life emergency response problems. The goal is ultimately to improve present day technology and allow for better responses in future natural or environmental disasters.
1.2 Background

The ERL Emergency Robot competition is centered on the need for cooperation between robots functioning in different terrains (land, air and sea). (Figure 8, ERL Rulebook 2017)

Figure 8: euRathlon 2015 competition field

Marine, airborne and land based robots have to interact to successfully complete the challenges of the competition. Other specifications include maximum masses for every type of robot, the fact that they have to be battery powered, be built with an emergency stop button, etc… These specifications are based on the needs of real life situations (ERL rulebook, 2017).

For marine missions, one robot teams used for past editions is the SPARUS II. First developed in Girona, Spain, in 2010 and later improved in its second version in 2013, the SPARUS II (Figure 9, IQUA Robotics 2016) is a lightweight autonomous underwater vehicle (AUV) that allows for long autonomy and use in waters up to 200 meters in depth (EU robotics). Shaped like a torpedo, it has efficient hydrodynamics but is also capable of hovering without moving. Easy to program (ROS architecture) and to operate, it is ideal for scientific and industrial applications.
The robot is equipped with cameras and the received video feed is processed directly in the microcontroller located in the SPARUS II. The images treated through computer vision algorithms are essential for successful completion of some of the underwater challenges of the competition.

1.3 Objectives

The objective of this project was to code computer vision algorithms to detect underwater objects with good accuracy for the ERL competition. (Figure 10, ERL Rulebook 2017) The marine part of the challenge is made of several parts.

The AUV has to patrol the emergency zone to try and detect different objects. Large signalization buoys with a unique digit printed on the side are to be found and added on the robot’s internal map of the marine environment it develops in real time. The digit has to be recognized. The objective of buoy detection in real life scenarios is to perform landmark based navigation. This method of mobile robotics navigation is based on the recognition of objects with known locations by the robot, and the use of this information to calculate its own position. It is easy to imagine real emergency situations where previously placed buoys are used for navigation by the autonomous robots.

The robot also has to search for missing workers potentially trapped underwater. Their body location has to be sent to emergency brigades to allow for the body to be recovered.
Other tasks the robot has to accomplish are the detection and localization of pipelines used for bringing seawater to the reactors cooling system. The pipes have to be inspected for possible damages and in the case of leaks, robots must close the valve allowing the flow of water through the pipe. In the spectrum of this project, the pipelines will only be detected. This task has real situation applications of preventing radioactive materials from spreading outside of the power plant and into the sea.

In the following TFG, computer vision algorithms have been developed for successful and accurate detection of those three type of objects. Once detected, the robot may use that information to complete the required tasks.

The project had to be developed in C++ with the OpenCV library.
2. STATE OF THE ART

Human vision, known as the ability to interpret the environment using the visible light wave spectrum, is a scientific research subject that still holds many secrets. Nobody yet really understands how the brain processes visual input and is capable of easily recognizing objects, faces and even emotions in such a huge range of different conditions. We however do know that this activity is complex and requires a big part of the brain, the visual cortex, which represents up to around 50% of the total brain volume in certain mammal species such as macaques. Once the era of computation and robotics started, it was only a matter of time before scientists started trying to automate this process, and started a new research and engineering field, computer vision.

Computer vision is based on the analysis and understanding of digital images and video input by a computer system. The goal is often to replace the need for human vision and instead to automate different tasks in order to extract information and data from a set of images (Figure 11, Everingham 2008). The effectiveness of these automation techniques rely on the use of many different scientific branches including but not limited to physics, statistics, geometry, machine learning... This field has an endless amount of applications and the development of neural networks at the beginning of the decade has accelerated its growth even more. Nevertheless, much progress still hast to be made in order for Computer Vision to equal or surpass human capabilities.

Figure 11: Examples of image classification
2.1 History of Computer Vision

The beginnings of computer vision are often traced back to 1966, when professor Marvin Minsky of the MIT asked an undergraduate student to get a computer to describe the objects it saw through a camera as a summer project. This anecdote remains famous amongst computer vision specialists because of the sheer underestimation of the problem’s difficulty. It actually took about 50 years (ImageNet competition 2012) to start getting acceptable results for this task and it is still very far from being solved.

Underestimating the difficulty of progress yet to be made was in reality a typical mistake of the era made by most of the artificial intelligence experts in American top universities. Herbert Simon went as far as to say that “machines will be capable, within twenty years, of doing any work a man can do”. The truth was a bit different and artificial intelligence and computer vision are to this day still far from these objectives.

Nevertheless, computer vision was born and research bases were founded. In the 1970's, research was centered around the reconstruction of 3D scenes from 2D image inputs. This implied the use of edge detection and edge labelling in some early algorithms, and basic feature recognition in order to implement stereo correspondence (using two cameras. Some work was also carried out into understanding how illumination, intensity and shadows affect images, and on optimizing algorithms and hardware for efficiency.

The 1980's brought to the table more complex mathematical techniques in computer vision. The concept of image pyramids, which is a technique that consists in changing the resolution of an image and scaling it in multitude of sizes, became useful in image blending applications (image mosaics) as well as primitive object detection. Research also continued forward in stereo vision, different other 3D reconstruction techniques (texture, shading, range date processing, etc…) (Figure 12, Burt 1983), dynamic contour tracking and optimization algorithms...

![Figure 12: Image blending of an apple and an orange using spline reconstruction](image-url)
The 1990's contributed to the field in several different aspects. The improvement of projection algorithms (geometric transformations) to regularize images helped improve recognition, and significant improvements in stereo vision, tracking algorithms and optimization were made. Physics-based vision was invented; it is about interpreting an image based on an underlying knowledge of how the image was taken and the use of the laws of physics. Detecting surfaces based on light reflection, reconstructing a 3D scene of a moving object with missing information using a Newtonian physics model, estimating illumination… The applications and techniques are very wide. Image segmentation, extracting only interesting parts of the images to improve detection speed (Figure 13, Belongie, Fowlkes, Chung et al. 2002), was also an active research subject. It is at the time we also started beginning to see learning algorithms, a classification of images based on statistical techniques applied mainly to face recognition and curve tracking. Finally, computer vision and computer generated graphics began to closely interact in multiple fields such as image morphing, and image based rendering, reconstructing a realistic textured image based on different images taken from other viewpoints (3D modeling) or in a different context.

While progress was made during both of the previous decades mentioned, it was rather slow and this period is commonly referred to as a period of “AI winter” in the sense that field-changing discoveries rarely occurred… This started to change in the early 2000s with a few revolutionary innovations. The clear increase in computing power and the new use of the internet as the world’s biggest database were big factors in the field’s newfound dynamic. The availability of huge amounts of images allowed new possibilities with learning techniques based on statistical techniques of classification and other machine learning algorithms. Face detection algorithms which at the time were pretty inefficient were instantly outdated by the Viola-Jones algorithm (Figure 14, Viola, Jones 2004) which produced way better results, and several feature extraction techniques were developed such as Haar features, the Histogram of Oriented Gradients (HoG), Scale Invariant Feature Transform (SIFT), etc… Feature extraction drastically improved the speed and efficiency of object recognition using machine learning models but also had many other applications like image stitching, localization recognition, human action recognition, 3D modeling (Figure 19, Zhang 2004)…
The revolution that came next is very recent. This past decade, the performances of deep learning based techniques, which completely bypass the need of feature extraction, are so great that they completely revolutionized the computer vision field and allowed a complete rethinking of the possible application range. This revolution is usually dated back to the 2012 ImageNet Large Scale Visual Recognition Challenge, when Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton submitted the only deep learning based algorithm, and outperformed the second place runner up by a huge 11% percent margin (85% recognition rate). A year later, all of the contest entries were deep learning based and some convolution neural network (CNN) based algorithms surpassed the human 95% recognition ratio in the 2015 contest. Rapid progress is presently being made (Faster R-CNN, YOLO) and the field is in an extremely fast growing period (Figure 15, Redmon, Joseph and Farhadi, 2018).
Figure 15: YOLOv3 object recognition
2.2 Applications of computer vision

Computer vision has an almost endless range of applications, and the great advances of recent years opened the door even more for groundbreaking technologies based on computer vision.

An obvious trendy use are self-driving cars. Computer vision is essential in a project like this, as it is used in a multitude of different ways. Lane detection, vehicle and pedestrian detection, traffic sign recognition and classification using deep learning, are only a few of the many uses for computer vision in such technology (Figure 16, Zuccolo 2017).

Face recognition is another well talked about application of computer vision which improved drastically over the last couple of years and was implemented in the last generation of iPhones and in the Facebook tag feature. Gesture recognition, optical character recognition (OCR), QR and barcode reading, autonomous robotics and computer-human visual interactions are all also very popular machine learning based computer vision applications.

The truth is however that it is impossible to list all of the present day computer vision utilizations and it would probably be easier to list off fields still unaffected. This wouldn’t be too interesting for the purpose of this document and we will instead establish a list of the most widespread uses. (Szeliski, R., 2010):

- **Process control**: Detection of anomalies in autonomous industrial processes
- **Inspection**: Rapid detection of possible defects in machinery or manufactured products or agricultural material.
- **Identification tasks**: Used to drastically improve human recognition rates in big data treatment of images (identifying criminals through video cameras, inappropriate images on social media), using satellite imagery for big data scientific research (Figure 17, Le Bris 2016).
• **Medical:** Medical image processing and information extraction, tumor detection, pulse, organ size and blood flow measures, noise-reduction in human analyzed X-Ray images... *(Figure 18, Zhang 2015)*

• **Military:** missile guidance, enemy soldiers, weapons and vehicle detection through drones and satellite imagery, battlefield information extraction...

• **Space exploration:** Autonomous landings and control of mars space rovers

• **Event detection:** Surveillance applications such as people counting
- **Database organization**: Indexing databases of images such as the google image search.
- **Photography**: Image stitching, resolution improvement, HDR
- **Visual effect creation**: used for special effects in the cinema industry (image morphing) or shot improvement (camera tracking)
- **Modeling**: 3D modeling and image and scene reconstruction, Neuroscience, Big Data extraction
- **Motion analysis**: Tracking, optical flow…
- Many more...

We can infer without taking too much risks that as vision techniques continue improving, the number of uses for it will go up with it. We live in exciting times.

*Figure 19: 3D reconstruction with stereo matching*
2.3 Computer vision in subaquatic environments

These past few years have seen the development of many unmanned vehicles be it in space, on land or in the air. Unmanned remote-controlled underwater vehicles also started to play a vital role in different scientific, military or even commercial applications such as inspection and repair of manmade human underwater constructions, seabed mapping, biology, wreck discovery...

Several problems can however occur in those type of missions. Underwater, both acoustic and electromagnetic waves are rapidly attenuated and though it complicates accurate sensor observations, a major problem this causes is the fact communication with external human operators is usually limited, and to achieve complete vehicle autonomy, it is best when calculations and tasks are automated and performed onboard (Garrison 2004). Computer vision techniques have therefore been used as an autonomous navigation source, but also in order to track cables on the seabed for inspection or for wreckage, geological and biological surveying.

Computer vision is however more complex in subaquatic environments. Four main phenomenon exist and degrade image quality:

- Light attenuation (Figure 20, Garrison 2004)
  - Light in water is rapidly dimmed and filtered and it is usually impossible for cameras to perform adequately for missions exceeding 10 meters in depth. This often imposes the need of artificial lighting. In addition to general light attenuation, water acts as a filter which tends to absorb way more efficiently longer wavelengths (red - 800nm) than shorter ones (blue -400nm). Without artificial lighting, it is estimated sea water absorbs more than 99% of red light in as little as 4 meters of depth. Thus observation distance is a crucial factor.
  - Other effects influencing light dimming include turbidity (water clarity), and surface conditions since light reflection on the air-water interface is dependent on its shape.

Figure 20: Underwater color absorption
• Scattering
  o Subaquatic suspended particles or air bubbles can deflect photons along their initial straight trajectory. This effect is known as scattering. Backscattering occurs when light is reflected as such from the light source to the camera lens and can cause bright spots in the resulting image (marine snow) (Figure 21.b, US National Ocean Service), or affect image contrast. Forward scattering happens when light is only deflected from its original path by a small angle, causing a reduced image contrast and some edge blurring. (Figure 21.a, Garrison 2004)

![Figure 21: Representation of back and forward scatter (left) and picture of marine snow (right)](image)

• Image distortion
  o Pressure conditions and camera waterproofness require the use of special lenses or casing with design imperfections that can cause distortions. Moreover, light changes materials a minimum of two times (water-glass interface and glass-air interface). This change in refractive index causes a refraction effect and deform the image non-linearly. Models can be made to simulate the refraction and be able to correct its effects.

• Image processing
  o Image processing is complicated underwater due to lack of recognizable features in the marine environment and on the sea floor, moving shadows caused by artificial lighting, the effects listed above...

Underwater vision therefore can be used to perform different aquatic tasks and shows many advantages compared to the use of other sensors (cost, ease of inclusion, flexibility and accuracy…). However, a lot of research has yet to be made in the domain to overcome the set of challenges the environment poses. Improvements in artificial intelligence, and sensor fusion are ongoing improvement aspects that promise a bright future to underwater computer vision.
3. COMPUTER VISION TECHNIQUES IMPLEMENTED

Multiple computer vision techniques were used during this project. They are described in this chapter.

3.1 HOG Feature descriptor

A feature descriptor is the representation of an image through a set of numerical weights, with the end goal of simplifying the image by extracting relevant information while disregarding the rest. Feature descriptors often convert 3-channel 2D images to a feature vector of real numbers. Feature vectors are then reduced enough in regards to the original image to be used, amongst other things, in machine learning algorithms for classification purposes such as Support Vector Machines (SVM).

In this TFG, the HOG (Histograms of Oriented Gradients) feature descriptor is used. The method of calculating HOG descriptors is explained.

Step 1:

The first step consists in extracting from the original image the region of interest, and resizing it so that it corresponds to a specific fixed size. In the case of human detection, it is common to choose a size of 64 by 128 pixels.

Step 2:

The second step is achieved by filtering the image with two kernels (matrix) (Figure 22), a horizontal and a vertical one. This works by superposing the kernel to every single pixel of the image, calculating their new values as the sum of the multiplication of the original image’s pixel value by the overlying kernel coefficient. Typically, the 1 by 3 kernel applied an individual pixel would subtract from the value of the pixel to the right, the value of the pixel to the left and store this result in a new matrix.
Figure 22: Horizontal and Vertical kernels used

This allows to detect both vertical (with the horizontal kernel) and horizontal gradients (with the vertical kernel) of the image. Using this information, it is possible to compute the pixel by pixel magnitude and direction of the gradient with the following formulas:

\[ g = \sqrt{g_x^2 + g_y^2} \]
\[ \theta = \arctan \frac{g_y}{g_x} \]

Figure 23: Sobel Edge Detection process on picture of Usain Bolt

This process is commonly called Sobel Edge detection (Figure 23, Satya Mallick 2016).
Step 3:

The image is then divided into equal sized square cells. The optimal size of these cells depend essentially on the applications of the HOG feature vector created and on the objects it intends to represent.

For each cell, the objective is to extract 9 weights. Each weight represents an angle, ranging from 0 to 160 degrees with a step of 20 (0, 20, 40, 60, 80, 100, 120, 140, 160). It is considered that the sign of the gradient is not important, therefore negative angles between 180 and 360 degrees are assimilated to their positive counterparts. Each weight is then incremented in function of the magnitude and angle data of each pixel in the cell (Figure 24, Mallick 2016).

![Figure 24: Bin repartition in HOG vector creation](image)

The final value of the weight vector can be represented like this (Figure 25, Mallick 2016):
Step 4:

The weight vectors are normalized to prevent the histogram of being affected by lighting variations. The vectors are then concatenated into a big final vector, the HOG feature vector which can then be used for classification.

3.2 SVM Classification

In machine learning classification problems, once the data is processed, the goal is to identify which category it falls into. In the case of digit recognition for example, the 10 possible categories would be all the digits 0 through 9. The most common sort of classification is binary classification, in which the machine intents to sort data into two categories (dog or cat for example). Modifications to the classifying algorithm can be used to generalize techniques in order to apply n-class (or multiclass) classification and differentiate input data in a wider range of independent output.

Several classifying algorithms exist, ranging from decision trees to more complex convolutional neural networks (CNN) but in this project, Support Vector Machine (SVM) classification was used. An SVM model represents data points in n-dimension space given n-dimension vectors as input data. It then tries to find the best hyperplanes of dimension n-1 in order to separate the vector to the maximal extent. Though in many cases, deep learning methods based on CNNs are nowadays more precise, SVMs still have a lot of benefits, including the facts that the model is very quick to train and usually needs a lot less training data to produce satisfying results.
In the above graph (Figure 26, Mallick 2018), the data points in blue or red are the labeled input data, representing vectors in 2 dimensions (X1, X2). The SVM algorithm intends to separate these 2D data points with a 2-1 = 1-dimension plane, thus a line, the decision boundary. If the points were in one dimension, a single threshold value would be sufficient to linearly separate the data points, in three dimensions, we would need a 2 dimensional plane, ...

The closest points to the decision boundary plane are the support vectors, and to optimize the problem, these points must be at the greatest possible distance of the separating hyperplane and reside on opposite sides of it. Real world data is however always imperfect, can be noisy, and the best model sometimes has to take into accounts the fact it is hard to cleanly separate the data with a plane without any misclassification. To solve this issue, SVM a uses a parameter C (Figure 27, Mallick 2018) which controls the tradeoff between a more robust classification that can sometimes misclassify outliers and a classification that prefers reducing to the maximum the numbers of misclassifications.
All of the previous examples discussed linear classification but researchers improved SVM with the kernel trick to allow classification of non-linearly classifiable data. By adding an additional dimension to input data, mapped with a predetermined function (such as a Gaussian Kernel), it can become possible to linearly separate the data in a higher dimension. *(Figure 28, Mallick 2018)*
Training a digit recognition model

The data used to train the digit recognition model used in this project is made up of a set of 500 binary images of each digit (Figure 29, Mallick 2017), 0 through 9. 90% of these digits are used as training data that the computer evaluates and tries to find patterns in for classification, while the remaining 10% are test data that the algorithm doesn’t train on but uses to evaluate its performance and tweak the influence it gives to each parameter in order to improve detection. OpenCV includes a function trainAuto() that automatically selects the best parameters for the SVM classification (Gamma parameter, C parameter, Kernel Type, etc…). Once the model is trained, it is saved to a YML file and can be opened and used in the necessary algorithms.

3.3 Color Detection

Color detection and identification is one of the oldest, simplest, and most used technique of computer vision. Although this technique presents many problems, mainly its dependency to lighting conditions, noise, cameras used, it is sometimes the simplest and most efficient way to detect an object in certain types of situations.

According to Wikipedia, color spaces are specific organizations of colors usually associated with color models, a mathematical representation of the way colors can be represented as n-tuples of numbers (triplets in RGB color model for example.) In this project two main color models were used, RGB and HSV.
RGB (Red, Green, Blue) is an additive color model that represents the intensity needed by each of the three type of lights to be emitted in order to produce a given color. It is the most widely used color model, in use in computer and phone displays, TVs, cameras…

HSV (Hue, Saturation, Value) is an alternative to the RGB color model (Figure 30). Hue (dominant wavelength) determines the angle the colors will be radially arranged around the color cylinder, saturation represents the purity of the light wave and value represents the intensity of the wave. This color model presents the benefit of only having the parameter H to describe color which makes it easier to isolate a certain range of similar hues in an image. This model is a bit more impervious to lighting changes since most differences will only change the V value without affecting the H, this adds robustness to color identification.

![Figure 30: Color cylinder based on HSV color model](image)

### 3.4 Noise reduction and image pre-processing

In computer vision, working with the best input image possible is essential. Noise reduction is often one of the first step taken to improve image quality before processing it. Image noise is defined as a random variation of brightness and color information in an image caused by electronics and sensor’s imperfectness or unwanted particles in the shot. Different types of noise exist and are filtered using different techniques.

**Gaussian noise** (Figure 32, Guo 2013) arises due to electronic circuit noise and the cameras imperfection during the shot. The distribution of the noise follows to the Gaussian distribution (normal distribution). **Gaussian filters** use a Gaussian kernel to calculate the weighted value of the center pixel of the kernel. Gaussian kernels give more importance to pixels closer to the center as opposed to box kernels in **box filters** which just calculate the unweighted mean of all pixels in the kernel. The kernel is applied to each pixel of the image, thus reducing the
impact random valued pixel outliers caused by noise had on the image. The size of the kernel influences the amount of filtering and the blurriness of the output image.

In the case of the project, the elected kernel sizes were relatively low (5x5 matrix in most cases) (Figure 31, OpenCV documentation).

Salt and Pepper noise (Figure 3, Guo 2013) is usually caused by sharp disturbances in the image signal. This causes sparsely occurring black and white pixels around the image. This type of noise can be reduced using a median filter. This filter replaces the value of each pixel by the median value of all of the neighboring pictures in a square neighborhood. This way, the outliers are eliminated and the image is treated.
Many other types of noise (Shot noise, quantization noise, film grain, periodic noise, anisotropic noise, ...) and filters (Bilateral filters, blurs, derivative filters, ...) exist but will not be discussed in this project.

Another way to improve input images in grayscale images is to make sure to distribute the intensities as evenly as possible over the entire range of possible values. This is called **Grayscale Histogram Equalization** and helps enhance the contrast of the image and improve further image processing (*Figure 33, OpenCV documentation*).

*Figure 33: Comparison of non-equalized and equalized grayscale image*

In the above images, we see that the intensity values were previously all packed between 120 and 210. Using Histogram Equalization stretches the possible values the image takes to use the entire 0 to 255 range.
3.5 Canny Filter

A Canny filter is an edge detection algorithm that improves upon the Sobel operator (described in 3.1) by using the gradients given by the Sobel output and finding the local intensity maximums perpendicularly to the edge direction in order to only keep one pixel wide “thin” edges (non-maximum suppression).

Canny filters are improved by using a process called hysteresis thresholding (Figure 34, OpenCV documentation), (Figure 35-36).

![Graphical example of hysteresis thresholding](image)

Any pixel with intensity gradients superior to the pre-specified maxVal are automatically included (sure-edges) in the final output image. On the other hand, any pixel with values below minVal are automatically discarded. Pixels with intensity gradients between maxVal and minVal are only included if they are connected to pixels above maxVal. In the above image, the edge A is above maxVal and is kept, and the C edge is connected to A so it is also considered a valid edge. The B edge however is not in any way connected to sure-edges and is thus discarded.
minVal and maxVal have to be chosen appropriately to yield the best possible results.

3.6 Hough Line Detection

Hough line transformation is a method used to detect straight lines in preprocessed images (with canny filters for example). It works by computing each infinite line that can pass through each position of the white pixels in the input image. The family of lines are represented not by the (a,b) parameters in Cartesian space (\( y = ax + b \)) but by pairs in polar coordinates (\( r, \theta \)), with \( r \) depending on \( \theta \):

\[
r_\theta = x_0 \cdot \cos \theta + y_0 \cdot \sin \theta
\]

For given \( x \) and \( y \) positions, plotting the infinite number of lines that can go through gives a sinusoid (Figure 37, OpenCV documentation).
By applying the same operation to every white pixel of the image we will get intersections between the sinusoids. If three and more points intersect at the same spot, this means those points are aligned along the same line. We can define the minimum number of points necessary to confidently detect a line (Figure 38, OpenCV documentation).

It is then possible to reconstruct the line with simple trigonometry using the formula:

$$ y = \left( -\frac{\cos \theta}{\sin \theta} \right) x + \left( \frac{r}{\sin \theta} \right) $$

This technique is called **Standard Hough Line Transform**. **Probabilistic Hough Line Transform** (Figure 39) also exists and is a more efficient method to detect imperfect lines. It
generalizes the Hough Transform to include probable lines not detected by the standard transform.

Figure 39: Detected lines (in red) using Probabilistic Hough Transform
4. BUOY DETECTION

One of the objectives of this project was to detect the orange buoys present in the water and to identify the digit written on their surface. In the competition, the orange buoys have to be located and placed on the internal map of the AUV. They are orange, spherical and between 25 and 50 cm in height (Figure 40, ERL Rulebook 2017). The digits printed on them have heights ranging from 10 to 15 cm and may be repeated up to three times along the equatorial axis.

![Figure 40: Competition Buoys](image)

In this project, for simulation purposes (7.1) ping-pong balls were used to replicate the buoys. The following image pipeline was used to achieve these goals:

4.1 Preprocessing

The video file is read frame by frame using an executable compiled with OpenCV libraries. Each frame then has its pixel intensity equalized with a slight variation to the Grayscale Equalize Histogram method that allows for colored images to be treated. Finally, the image is filtered to remove noise. The noise in the experimental video footage appears to be a combination of salt and pepper noise and Gaussian noise. Bad lighting conditions, electrical noise, water quality and camera defects are all possible causes of the noise captured. It was found empirically that the most effective treatment was using a median blur of kernel size 5 after testing all other types of blurs and kernel sizes (Figure 41).
4.2 Color Detection

The 3 channels of the RGB image are then converted to the HSV color model in order to facilitate and improve the robustness of color detection. An experimentally predetermined range of values of the Hue, Saturation, and Value parameters is kept while the rest of the pixels are discarded. This range corresponds to the orange color the buoys have underwater and is relatively wide so as to not be too affected by illumination changes, water purity, etc... This process is called inRange Thresholding and outputs a binary image (Figure 42).
4.3 Morphological operations

The images left at this stage are often composed of various groups of pixels, some bigger than others, with a few groups that are false positives. Based on the assumption that bigger are the groups of adjacent pixels in our desired range, bigger are the chances that they represent the desired orange object, morphological operations are applied to the binary image (Figure 43). These operations consist in expanding or eroding the edges of the pixel groupings by “accumulating onto” or “eating away” exterior pixels of the pixel groups. These two operations can be used to morphologically “open” or “close” an image.

**Opening** an image consists of first eroding by a few pixels each pixel group. This has the effect of eliminating from the image all the groups smaller than a certain size and reducing the size of the others. All the pixel groups are then expanded in order to get back to their original size. The basic function of this operation is therefore to reduce noise by only keeping the biggest pixel groupings while discarding the rest.

**Closing** an image is the inverse operation. First the pixel groups are expanded, then they are eroded to get back to their original size. This has the effect of closing down any internal holes the pixel group may have had and to smooth out the edges of the groups. The resulting image is composed of more compact groups.

In the case of this project, the image is opened to eliminate all false positive orange detected pixels and to ideally only keep one group of pixels representing the buoy. If the circularity of pixel groups had to be calculated using the ratio of area to perimeter, it would then be a good idea to close the image to smooth out the edges and to remove any holes caused by non-orange pixels inside the buoy (the black painted digits for example.) Low image quality however prevented this technique to properly function in the majority of images and morphological closing was not applied to the image.

The pixel group with the biggest outside contour is then detected and the position of the center of mass, the area, and the perimeter of the pixel group are calculated. This information will be used later on to determine with confidence if a buoy was detected or not.
4.4 Confidence Evaluation with Buffer Technique

During the initial test phase of the color detection algorithm previously implemented, a problem that frequently occurred was detection instability. The movement of the water, the reflections on the sides of the aquarium, illumination changes and more all caused very short (1 or 2 frame long) false detection errors, with the largest pixel group being detected in wrong areas of the image with completely different dimensions, or not detected at all. To overcome this problem, a confidence evaluation technique was developed.

For the last 10 frames of the image, the information about the size and position of the largest pixel group is kept in a circular buffer. This buffer is a first in first out (FIFO) data structure in which every time a new frame is processed, it replaces the oldest one. The average x position, y position, width and height of the pixel group during the last 10 frames is then calculated. For each of the frames, the squared difference between the average and the current value of the 4 parameters is summed, then rooted to give an error parameter:

$$\sum_{k=0}^{10} \sqrt{(\text{width of frame } k - \text{average width})^2 + (\text{height of frame } k - \text{average height})^2 + (\text{x pos of frame } k - \text{average x})^2 + (\text{y pos of frame } k - \text{average y})^2}$$

If no pixel group is detected in one or more of the frames, the error parameter is incremented by an experimentally determined penalization coefficient. It is possible to get a graphical understanding of the performed calculations by thinking of 4 dimensional vectors formed by the 4 parameters. The distance of each of these vectors to the average vector is summed to give the final error parameter. This parameter represents the stability of the detected pixel group. If this group changes too much too quickly, the error value will quickly rise, responding to this abnormal behavior. On the other hand, a stable detection of pixels staying in a relatively close by area of the image with a stable size will give a very low error value. All is left to do is choose a threshold confidence value to be able to algorithmically detect the buoy with the desired confidence level.
4.5 Digit Detection

Once it is established with confidence that a buoy has in fact been detected, the following objective is to be able to read what is written on it. To optimize the process, only frames with high confidence factor (Figure 45) and in which the detected buoy was big enough to read were put through the digit recognition process.

The zone of the image containing the buoy, or zone of interest, is isolated and resized. The image is then converted from RGB to grayscale and image is thresholded in order to only keep the darkest spectrum of the gray pixels corresponding to the black painted digits.
The black digit is only visible on one side of the buoy which signifies it is possible to have a buoy without a digit. To ignore these cases, the ratio of the area taken by the black pixels over the area of the zone of interest is calculated and only images with ratio comprised of ratios ranging from 8 to 33% are processed further on. A bounding box for the digit is then evaluated (Figure 46), and if all the conditions imposed upon this box are met, the digit is centered and the the box is resized to a 20 x 20 size.

### 4.6 Digit Recognition

Given as an input 20 by 20 binary images of the digit to recognize, the recognition is carried out in several steps (Figure 47).

The first step is to “deskew” the digit, that is to calculate the image’s moments and main axes and re-orientate it to a constant straight angle. This helps remove all the recognition problems caused by digits or images slightly tilted to the left or right.

Next, a Hog feature descriptor is calculated as explained in section 3.1. This step outputs an 81 dimensions Hog descriptor vector. Finally using multiclass Support Vector Machine Classification (SVM, section 3.2), the value of the detected digit is returned with great accuracy.

To further improve the confidence level of the detection, a circular buffer similar to the one used in buoy detection (4.4) is used. It measures over the span of 10 frames the most frequently detected digit and the confidence level represents the percentage of times this digit was detected over these last frames.
Figure 47: Image pipeline of the detection of the digit and binary thresholding
5. PIPELINE DETECTION

The second mission the AUV had to carry out during its emergency response mission was pipeline detection. In the competition script, pipelines were used to transport seawater to the nuclear reactors in order to cool them down (Figure 48, ERL Rulebook 2017). Detecting the pipelines, then inspecting them for defaults allowed the robots to close the valves controlling the water flow through the tubes if necessary. In real life situations, this would be vital for limiting the spread of radioactive materials through the sea.

![Figure 48: Image of competition pipelines](image)

They were simulated with PVC pipes during algorithm development (Figure 49). The image pipelining process for underwater pipeline detection is similar to the one used in buoy detection.

![Figure 49: Original image taken with the camera in the experimental setup](image)
5.1 Preprocessing

Input images are grayscaled, their grayscale intensity is equalized and they are filtered with a median blur of kernel size 7. The filtering in tube detection is the most important because the goal will later on be to detect straight lines. De-noising images to the maximum is therefore crucial to get good results later on. Several types of filters and kernel sizes were tested in different lighting conditions before arriving to a final decision (Figure 50).

![Figure 50: Original image with grayscale equalization (left) and median filter of kernel size 7 (right)](image)

5.2 Canny Filtering and Line Detection

As described in section 3.5, a canny filter was used on the image followed by a probabilistic Hough transform (3.6) for line detection which allows the detection of several straight lines in the image including the tube edges.
Figure 51: Canny filtering of image in previous figure

Figure 52: Probabilistic Hough line detection
5.3 Line selection

All the detected lines are not lines that are necessarily interesting. In the conducted tests, the experimental conditions furthermore complicated the detection task because the aquariums geometry added multiple edges and reflections which would not occur in normal aquatic conditions.

To weave out undesired lines, a selection process was implemented. The objective was to only select almost parallel lines that had a minimum length and had a distance separating them comprised in a pre-specified range. The distance was calculated in an absolute way, as a number of pixels. This was not intended to measure the pipeline’s diameter but to filter out results in which lines were almost superposed or inversely detected as the image’s borders. Amongst the remaining lines, the two best (according to the Hough line transform confidence weights) were selected (Figure 53).

![Figure 53: Final line selection drawn on original image](image)

5.4 Final Tube Detection

To overcome detection instability problems, the same circular buffer and squared error technique implemented for the buoy in section 4.4 was implemented for the tube. The calculated parameters were in the tube case the average X position and average Y position of the tube pixels and the angle of the detected tube line (between 0 and 180 degrees). This gave an error parameter used to estimate the confidence level of the detection.

Other included information given by the detection was the tube’s angle and position.
Figure 54: Line detection algorithm real time output
6. HUMAN BODY DETECTION

In the ERL competition, mannequin recognition is part of the top priority mission to succeed in the contest. They are meant to represent the bodies of workers (Figure 55, ERL Rulebook 2017) who didn’t have time to evacuate the power plant in time and were trapped inside. The robot’s task is to detect the body and to send out its positions for human rescuers to retrieve it. This part of the competition has obvious applications in multiple real-life situations. In the experiments, LEGO's with similar body positions were used to replicate the mannequins.

![Figure 55: Competition mannequin](image)

Detecting humans swimming or drowning in the water is probably the trickiest part of this project. Human detection has been one of the biggest problems in computer vision history and although HOG feature descriptors associated with SVM classification (3.2) greatly improved all existing algorithms for pedestrian detection in 2005 (Dalal and Triggs), only the emergence of deep neural networks a few years ago truly helped achieved satisfying results in detection for wide ranges of body positions, camera inclinations, types, etc…

As it will be shown in the results section of this paper, the human detection algorithms implemented in this project only yielded partial results. Very good detection was achieved in certain situations but the algorithms didn’t fare well when subjected to bodies oriented along non vertical axis or changes of camera orientation.

The technique used, after classic preprocessing, is based on HOG descriptors and the sliding window technique with SVM classification. The model is pre-trained with a data set of thousands of grayscale pictures of standing humans from the front, side and back and is included in OpenCV as the “Default People Detector”. Multi-scale detection, also known as the sliding window technique with image pyramids is then carried out. It consists in trying to run the SVM classification on a small part of the image (window) and then moving this window all around the image to try and detect the desired objects. The window size is then increased...
and the process starts again. At the end, the best matches are kept, and the technique allows to localize the position of the searched objects in the initial image (Figure 56).

The problem of this technique is that there is a lot of comparisons to be made which causes it to be very slow. It is also rotation dependent, which means that it will only recognize objects that are angled the same way as the objects in the training set. It is of course possible to rotate the image and then run the detector on each new orientation but the process becomes even slower and the speed was not satisfying for the project’s needs. For this project, in order to be able to work with real time data, only one in every 5 frames is processed and scanned with the multiscale technique (Figure 57).

Again, the cyclical buffer technique was used (4.4) to give a confidence level to the detection using previous positions of the detected objects. The multiscale detection function also returns a confidence weight parameter for each detected object which was also taken into account in the final confidence level calculation.

Figure 56: Person detection example

Figure 57: Output screens of computer vision interface for human detection
7. RESULTS

In order to truly evaluate the performance of the algorithms, images may not always be sufficient. Video results that give a real time understanding of the efficiency of the software are available at this link:

https://docs.google.com/document/d/13PYBGUEQnC_xA4DX-PTBO2_XbKkrTRW-uJ1pNwkmsMk/edit?usp=sharing

7.1 Experimental Setup

In most engineering projects, arriving at the final stage is an incremental approach and tests and simulations have to be conducted before completing the final version of the product. For financial and organizational reasons, it is often a good idea to conduct the initial tests in simulated environments before trying the product in real life conditions.

This approach was used in this project. It would have been impossible to only work with the final robot in a real marine environment for obvious monetary and convenience issues. The algorithms were thus developed in partially replicated marine conditions using an aquarium, murky water and small scale versions of the objects to detect (Figure 58).

![Aquarium and waterproof case of the camera](image)

Figure 58: Aquarium and waterproof case of the camera

The buoys were simulated with similarly shaped and colored ping-pong balls, the pipelines with smaller diameter PVC pipes, and the human dummy with human shaped LEGOs as can be seen in previous images. Different lighting conditions were achieved by putting different sources of lights or conducting the experiments at different hours of the day. Finally, water purity was altered by adding dust, sand and vegetal objects to the water to better experiment the good operation of the algorithms in more difficult conditions.
The camera that was used (Figure 59) was a low-light analog camera sold by the BlueRobotics company, composed of the 1/3” Sony Super HAD 810 CCD sensor and the Sony Effio-A 4151 System Integrated Circuit. Sold as having an extremely performant low-light sensitivity (0.0003 lux according to the manufacturer), this camera is normally able to detect colors in dark situations and is easily capable of adapting to differences in illumination. It has a 2.1mm lens and a field of view of 128 degrees horizontally and 96 degrees vertically.

Initially, the implemented vision system was to also be tested in a pool with the final robot but time constraints sadly didn’t permit those stage 2 tests to occur.

### 7.2 Problems encountered

During the course of this project, several problems occurred.

The main one was due to the hardware failure of a USB video adapter made to convert the analog video input into a readable format for the computer. The problem was mistakenly identified for a very long time to be a software problem and a ton of drivers were reinstalled, Ubuntu was updated, etc… It turned out the part was defective and after replacing it (Figure 60), the video input from the underwater cameras became readable but a lot of time was lost with this issue.
Another problem arose because of the non-ideal experimental conditions. The tests were conducted in a small aquarium which had very visible edges (figure 61) and whose glass sides allowed the objects inside to produce unwanted reflections which deteriorated the quality of the detection. The very shallow water inside of the aquarium also in many cases allowed light to be reflected on the water surface thus causing up to 6 unwanted reflections which would not have happened in a marine environment with a subaquatic robot.

These problems were never really physically fixed but software was made much more robust to ignore all of the image perturbations caused. It is logical to infer that if the software works in bad conditions with images that are far from ideal, it will work even better in better conditions without extra edges and reflections. Time constraints sadly made testing the software in better conditions impossible…
7.3 Buoy detection

Buoy detection yielded some very good results in clear water. The orange color was always detected during the experiments, at all possible distances within the aquarium (1.3 m) as can be seen in the videos. Though the testing distances may appear to be very short, the detected object also has a reduced size and proportionality is kept with the real situation. The results are thus relevant. The confidence levels calculated gave very accurate depictions of the detection strength by being very high during still phases where the buoy was somewhere in the image (Figure 62) and low during frames where the buoy was not or less visible. The algorithm was also tested using real life recorded images and performed without a glitch.

![Figure 62: Detection occurs when the buoy is small and barely on screen, orange isolation on right](image)

Digit recognition was also very efficient (Figure 63). Although false classification did occur, particularly in situations where the entire digit was not visible, the buffer system helped reduce to a maximum the number of misclassification and produced accurate results.
The most frequently occurring mistakes were the misclassification of the 3 into the 2 category (in about 5% of the recorded video data, often when the buoy is tilted) and the 9 into the 7 category (around 15% of the recorded video data, also when the buoy is a bit tilted). A bigger classification problem occurred at the time of classifying the 6 (Figure 64). It was misclassified over 80% of the time as a 0, 5 or 9. Troubleshooting the problem didn’t give much results and it is thought the training data made of handwritten digits used to construct the HOG descriptor-based model of the 6 was too different from the 6 printed on the buoy.

The 0, 1, 2, 4, 5, 7 and 8 were detected with 100% accuracy during the conducted tests in relatively clear water.
7.4 Tube detection

Tube detection also yielded some very acceptable results in clear water conditions (*Figure 65*). Although the edge detection was fairly unstable, with false detections often occurring over the span of one or two frames before coming back to detecting the correct object, the big majority of the time, detection was accurate and the buffer technique allowed to quantify whether the instability was normal and acceptable or on the contrary too frequent and meant no object were truly detected.

![Successful pipe detection](image)

*Figure 65: Successful pipe detection*

False positives (*Figure 66, 67*) were caused by several factors. An inaccurate measurement of the minimum distance caused some lines to be detected twice in a small portion of the cases (case 1), while the detection of aquarium edges (case 2) and surface and side reflections of the tube accounted for the rest of the false detections. These two last types of errors would not occur in natural sea environment and are merely experimental errors.

![Inaccurate distance measurement](image), ![Detection of the reflection](image)

*Figure 66: Inaccurate distance measurement (left), detection of the reflection (right)*
7.5 Human detection

As previously explained, human detection is very precise but only in certain types of situations. This is due to the training data only comprised of standing persons. The detection is thus very performant when the body is oriented along the vertical axis (Figure 68) but has a much lower recognition rate in other cases (Figure 69). Very few false positives are produced and the program can detect a rightly oriented human body at a large distance in clear water and slightly murky water.
7.6 Murky Water Tests

Tests were also conducted with dirty/murky water. Sand and dirt were added to the aquarium and stirred around to keep them floating around. 4 levels of water purity were tested: perfectly clean water (level 0), slightly murky water (level 1), murky water but with settled particles (level 2) and finally murky with moving dirt particles (level 3) (Figure 70).

Figure 70: Frame of the dirty water without buoy (level 3)
Close range buoy detection remained good, even in very dirty waters, and the computer was even in certain cases quicker to spot the buoy than human observers (Figure 71).

![Figure 71: Slight buoy detection is observed](image)

![Figure 72: Buoy trust function is non-zero but confidence is low](image)

However, as the distance increased, the colors were filtered and altered by the murky water and recognition rates decreased drastically (Figure 72). The color was not orange anymore and the color detection method chosen was not always robust enough to yield good results. However, on real images shot by last year’s robot, the color detection was always able to detect the buoys (on tens of meters of distance) (Figure 73).
Digit recognition also had a hard time functioning in difficult conditions. In slightly improved conditions (when the biggest particles had settled), digit recognition and buoy detection gave much better results (Figure 74).

As far as pipe detection is concerned, the troubled water had more or less the same negative impact; it didn’t affect the results much in close range situations but the recognition drastically decreased with greater distances. Looking at all the probabilistic Hough transforms (Figure 75,76), it is possible to notice the detected number of lines is greatly reduced compared to when the water is clearer. This is due to the particle noise which breaks line continuity and distorts the light waves. It is observable that in murky water without moving floating particles, the pipeline detection functions relatively well (much better than buoy detection at similar
distances) but doesn't yield good results with moving particles. Once again video footage of pipe detection in murky water is included in the video link.

Figure 75: Edges are detected but distorted in level 3 water, lines are not detected

Figure 76: At closer range, edges appear straighter

Human body detection (Figure 77) is a lot less affected by murky water compared to pipeline and buoy detection. This is due to the fact the detection is based on statistical similarities (SVM) of the HOG feature vector, so noise affects the detection much less. The images are not as neat so the recognition rate slightly decreases but the computer is still often able to correctly classify the noisy patch of pixels containing the LEGO.
In the following graph, human vision (purple) is compared to the vision system in function of the murkiness of the water, the distance and the type of objects. Conclusions obtainable from this graph is that in most cases human vision is still superior to the algorithms implemented in this project. However, in good water conditions, algorithms perform relatively well. It is interesting to note that SVM algorithms are more robust faced with water quality changes (body detection) but don’t work that well at long range, even in clear conditions. On the other hand, line detection algorithms are more affected by water movement (pipeline) and color detection algorithms fare relatively well to distance, but only in clear water conditions. Different methods thus have different benefits and combining them could yield better results (Figure 78).
8. IMPACT OF THE PROJECT

8.1 Uses and benefits

The algorithms implemented in this project relied for the most part on widespread techniques already implemented in other projects (HOG, SVM, Hough Detectors). They are pretty specific to the initial objectives of the project: detecting certain objects underwater. The obvious use they may have is to improve robot performance in the ERL emergency robot competition. Performant algorithms allow for quicker detection time and better results in the competition.

It would however be a shame to think the competition in itself is the end objective of such a project. The ERL challenge is made so that performant AUVs in the competition should also be performant in real life scenarios. As mentioned before, every task of the competition was inspired on potential real emergency tasks and having a better computer vision system could only improve on response time, operator safety, rescuing operations…

Furthermore, computer vision in submarine robots are not limited to rescue missions. Gas companies currently use submarine autonomous robots to map the seafloor bed before installing pipelines and underwater infrastructures or for pipeline inspection (Figure 79, Krupinski 2014). Scientists also use robots to facilitate lake and marine research and the military has uses for them in surveillance, reconnaissance, mine countermeasures, oceanography, payload delivery, etc…

The market for AUVs is huge and constantly keeps expanding. Computer vision is one of the essential technical challenges of such devices and future improvements will likely lead to an ever more widespread use of this technology.

Figure 79: Pipeline inspection with AUV
In a more personal sense, working on this TFG has also allowed me to discover a field I had no knowledge about. It interested me a lot and gave me the desire to further my studies acquiring skills in computer vision and machine learning and to develop new projects and applications with these skills. This project was very fulfilling.

8.2 Future Research Lines

This project is not a perfect solution to the initial problem. A lot of improvements are still possible and would not be too difficult to achieve with large amounts of data. I believe the first step would be to implement deep learning through convolutional neural networks (Figure 80, The data science blog 2016) for a much better object detection and classification. These sorts of algorithms have completely taken over the computer vision field over the span of these last five years and seduce because of their efficiency and speed (YOLO, Faster R-CNN). The training data and training time needed is much bigger than in other machine learning techniques (SVM) but the counterpart is that they yield better and quicker real time results.

The issues this project has can all be improved. Replacing color detection with other techniques (shape recognition for example) was attempted during this project but was finally not selected. It is however probably the best option to improve the robustness of the detection through several types of lighting and water murkiness. The human detection implemented in this project relied too much on correct inclination of the body. Augmenting the training data size to include other body positions or using a rotation invariant feature descriptor like SIFT (Scale-Invariant Feature Transform) could resolve this issue without choosing to use convolutional neural networks.

It would also be very interesting to continue researching and improving my buffer technique to really take advantage of past frame history to improve recognition. Better mathematical error models could be developed or implemented and more accurate trust factors could be calculated. Using different recognition techniques and merging them could also allow for improved trust coefficients.
The optimization of computing resources is a part of the project that was not explored with much depth. Improvements in this sector could be made by improving the sliding window technique used for body recognition which is the costliest process of the project in terms of number of calculations. Using techniques such as image segmentation (Figure 81), separation of the image in different zones of interest, could allow for the multi-scale detection of the body not to be performed on the entire image but only in zones that could potentially contain the body. This would speed up the process immensely.

![Figure 81: Image segmentation, only 10 windows would have to be processed in this case](image)

Finally, the SPARUS II has more than one camera on board that it receives video input from. By combining detection capacities from two of these cameras, it could be possible to improve detection recognition rates but also to calculate object distance (stereo reconstruction).
9. CONCLUSIONS

The final result this project produced is a set of algorithms allowing the computer to detect with confidence certain types of objects. Multiple algorithms with different methods were used to optimize the detection of every object.

All of the images were first pre-processed with various filters to improve the detection quality. The orange buoys were then detected using color detection in the HSV color model and by creating a confidence coefficient based on past detected buoy movement. This allowed good experimental results in most cases, but poorer performances in very disturbed water. Digits on the buoy were detected using machine learning algorithms such as Support Vector Machines and the detection performed extremely well in all cases but the digit 6. Training data could be improved to correct this slight error. Pipeline detection was achieved through Canny filters and probabilistic Hough transforms. For added stability, the past detection history was also made to influence the algorithm. Finally, human body detection was also achieved through SVM classification and the sliding window method.

Through experiments, it was discovered without much surprise that affecting different experimental parameters had varied impacts on the implemented computer vision methods. Color detection was most affected by distance and water murkiness, edge detection responded badly to disturbed water with moving particles, and human body detection was robust but not rotation invariant and only produced results when the body was oriented along the vertical axis of the image. These issues are all improvable through the use of different techniques and the use of convolutional neural networks could be the most impactful amelioration. Algorithms of the sort however require huge amounts of training data.

All in all, the algorithms functioned well throughout the experiments conducted. Nevertheless, it is important to note that the experiments were far from completely replicating real marine environments and that tests would have to be conducted before truly being able to validate the work conducted in this project. The experimental conditions were however in no way more ideal than real life conditions. The light was dim, causing the low light camera to produce a low quality image, and the aquarium presented a lot of unwanted features that could potentially disturb the good function of the algorithms. For a more in depth project, it would be essential to perform real scale marine tests.

As a conclusion, the realization of this project combined interesting computer vision techniques to successfully perform tasks imposed by the ERL emergency challenge. Knowledge yielded from this project will hopefully be used to improve future underwater computer vision problems of the sort. Next time a disaster situation occurs, techniques learned from such competitions and projects will hopefully serve to improve rescue missions and save lives! In the meantime, the project personally taught me a lot of things and helped me decide on the direction of my future studies.
10. SOURCES


“Sparus II AUV.” IQUA Robotics, iquarobotics.com/sparus-ii-auv.

“ERL Rulebook”, ERL, eu-robotics.net/robotics_league/erl-emergency/about/requirements.html, 2017

"Fukushima accident upgraded to severity level 7". IEEE Spectrum. 12 April 2011.

“About.” Eu-Robotics.net, ERL, eu-robotics.net/robotics_league/erl-emergency/about/index.html.


“OpenCV: OpenCV Modules.” OpenCV: Image Processing, 2018, docs.opencv.org/3.4/.


ANNEX 1: TEMPORAL PLANNING

To optimize the workflow of this project, a Gantt Diagram was made. It combines the different tasks needed to complete the project with the order in which they have to be sequentially achieved to find the most optimal planning.

<table>
<thead>
<tr>
<th>Name</th>
<th>Begin date</th>
<th>End date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choice of TFG</td>
<td>2/1/18</td>
<td>2/28/18</td>
</tr>
<tr>
<td>Meeting with tutor</td>
<td>3/1/18</td>
<td>3/1/18</td>
</tr>
<tr>
<td>Project approval by exchange responsibles</td>
<td>3/15/18</td>
<td>3/10/18</td>
</tr>
<tr>
<td>Familiarization with OpenCV</td>
<td>3/19/18</td>
<td>3/30/18</td>
</tr>
<tr>
<td>Experimental material acquisition</td>
<td>3/19/18</td>
<td>4/20/18</td>
</tr>
<tr>
<td>Initial algorithms without simulation</td>
<td>4/2/18</td>
<td>4/20/18</td>
</tr>
<tr>
<td>Experimental Setup</td>
<td>4/23/18</td>
<td>4/23/18</td>
</tr>
<tr>
<td>Buoy detection</td>
<td>4/23/18</td>
<td>5/11/18</td>
</tr>
<tr>
<td>Digit Recognition</td>
<td>5/1/18</td>
<td>5/21/18</td>
</tr>
<tr>
<td>Human detection</td>
<td>5/4/18</td>
<td>5/21/18</td>
</tr>
<tr>
<td>Pipeline Detection</td>
<td>4/23/18</td>
<td>5/2/18</td>
</tr>
<tr>
<td>Algorithm Optimization</td>
<td>5/22/18</td>
<td>5/28/18</td>
</tr>
<tr>
<td>Test phase</td>
<td>5/29/18</td>
<td>6/27/18</td>
</tr>
<tr>
<td>Final meeting with tutor</td>
<td>6/2/18</td>
<td>6/28/18</td>
</tr>
<tr>
<td>Material return</td>
<td>6/29/18</td>
<td>7/2/18</td>
</tr>
<tr>
<td>Project Theses Reduction</td>
<td>4/23/18</td>
<td>7/19/18</td>
</tr>
<tr>
<td>End of Project</td>
<td>7/20/18</td>
<td>7/20/18</td>
</tr>
</tbody>
</table>

Figure 82: List of tasks as displayed in GanttProject Program

The graphical representation is displayed in the following figure.
Figure 83: Gantt Chart
ANNEX 2: ECONOMIC STUDY

The project consists in the development of vision algorithms for a subaquatic robot. It is thus a project for an independent contractor payed to achieve the required objectives. It is therefore on him and not on the contractor to estimate his budget and optimize the costs to be able to make a profit at the end. In the case of this TFG, the worker is a student residing in Lausanne, Switzerland and Swiss laws apply.

Costs can be divided into two categories, direct and indirect costs. Direct costs include the hourly wage of the worker, material amortization costs and material costs. Indirect costs include all the necessities such as workspace, internet connection, electrical energy costs required to be able to work on the project.

The hourly wage cost is equivalent to the teaching assistant wage for a student of the Polytechnic school of Lausanne (EPFL), 24.50 CHF, all social security and taxes included. With current exchange rates, this represents 20.93 euros per hour. It is estimated the student worked for 250 hours for the realization of this project, so wage costs represent 5232 euros.

The material costs are:

<table>
<thead>
<tr>
<th>Material</th>
<th>Number needed</th>
<th>Buying cost per unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low light USB video camera</td>
<td>1</td>
<td>89</td>
</tr>
<tr>
<td>Waterproof camera case</td>
<td>1</td>
<td>15</td>
</tr>
<tr>
<td>USB 2.0 video capture converter</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>Connection wires</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>12V power source</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Used aquarium</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>Waterproof markers</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Ping-pong balls</td>
<td>3 x unit of 6</td>
<td>4</td>
</tr>
<tr>
<td>Lego figures</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>PVC pipe</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>185</strong></td>
<td></td>
</tr>
</tbody>
</table>

Material amortization costs will only include the amortization costs of the computer used during this project. The computer is needed for 250 hours during this project. It is estimated the computer is on average used 3 hours a day during three years (3285 hours) and costed 500 euros, plus 100 euros of reparations Thus the amortization cost for 250 hours should be 250/3285 * (500+100) = 45.68 euros.
It is possible to consider that the student worked on this project on the EPFL campus using the school’s resources. This way, he has access to water, electricity, Wi-Fi, printers, etc... The cost of these services will correspond to the portion of the school’s semester fee (622 CHF = 531.43 euros). For a normal 30 ECTS credit semester, a student is estimated to work for 900 hours. 250 hours thus represents approximately 1/4\textsuperscript{th} of the time spent on schoolwork so \(\frac{531.43}{4} = 132.86\) euros.

The total costs finally sum up to 5600 euros.
ANNEX 3: ENVIRONMENTAL IMPACT

The realization of this project may have an effect in a variety of different ways.

A.3.1 Ecological impact

The project is purely a software project. Nothing is built, no resources are used more than normally, and the ecology is in no way affected more than usual. It is of course possible to include in this study the energy used to power the laptop and the water used to fill the aquarium, and to also take into account the materials and resources used to manufacture the computer, the camera, the aquarium, etc., but it would clearly be overdoing it since these objects have a life cycle much longer than the project.

A.3.2 Economic impact

The developed software is intended to be included in autonomous rescue submarine robots. These type of robots could be useful in both private and military application and a big market could exist for them. This implies big economic repercussions. It is however impossible to quantify the economic impact the machine vision software has on the sale of the robots.

It is also notable that swift emergency response to nuclear plant failure could save up to billions of dollars in damages if the radiation can be contained by detecting it earlier, closing water valves to prevent radiation propagation etc…

A.3.3 Social impact

The biggest impact this project actually creates is a social impact. If it is proven that computer vision can really aid in rescue missions during dangerous situations by guiding autonomous robots, it will be possible to improve rescue time and save lives, both of people in distress and of rescuers.
ANNEX 4: TABLE OF FIGURES AND GRAPHS

Figure 1: Robot sent in the Fukushima plant after the disaster ........................................... 3
Figure 2: SPARUS II robot used in the ERL competition ....................................................... 3
Figure 3: Objects to recognize during the competition .......................................................... 4
Figure 4: High confidence detection of orange buoy and digit (left), lower confidence in murky waters (right) ...................................................................................................................... 5
Figure 5: Successful and problematic pipeline detection ........................................................... 6
Figure 6: Human Body detection .............................................................................................. 6
Figure 7: Robot entering into reactor building ........................................................................ 10
Figure 8: euRathlon 2015 competition field ............................................................................ 11
Figure 9: SPARUS II robot in marine environment ................................................................. 12
Figure 10: Mannequin representing a missing worker and signalization buoys ...................... 13
Figure 11: Examples of image classification ............................................................................ 14
Figure 12: Image blending of an apple and an orange using spline reconstruction ............... 15
Figure 13: Image segmentation .............................................................................................. 16
Figure 14: Face Recognition with Viola-Jones algorithm ......................................................... 17
Figure 15: YOLOv3 object recognition ..................................................................................... 18
Figure 16: Self-driving car applications of CV ......................................................................... 19
Figure 17: Wild oyster reef aerial sensing ............................................................................... 20
Figure 18: Structural MRI, Medical Imaging technique ........................................................... 20
Figure 19: 3D reconstruction with stereo matching ................................................................. 21
Figure 20: Underwater color absorption .................................................................................. 22
Figure 21: Representation of back and forward scatter (left) and picture of marine snow (right) .................................................................................................................................. 23
Figure 22: Horizontal and Vertical kernels used ....................................................................... 25
Figure 23: Sobel Edge Detection process on picture of Usain Bolt ......................................... 25
Figure 24: Bin repartition in HOG vector creation ................................................................... 26
Figure 25: Graphical representation of HOG bins ................................................................... 27
Figure 26: 2D SVM classification graphical representation ..................................................... 28
Figure 27: Influence of C parameter in 2D graphical representation of SVM ....................... 29
Figure 28: Non-linear SVM classification using kernel trick .................................................. 29
Figure 29: Digit recognition training data ............................................................................... 30
Figure 30: Color cylinder based on HSV color model ............................................................. 31
Figure 31: Representation of Gaussian Kernel (left) and Box kernel (Right) ......................... 32
Figure 32: Examples of Salt and Pepper (left) and Gaussian noise (right) .............................. 32
Figure 33: Comparison of non-equalized and equalized grayscale image ............................. 33
Figure 34: Graphical example of hysteresis thresholding ....................................................... 34
Figure 35: Original image, grayscale image, Gaussian blurred grayscale image ................. 35
Figure 36: Sobel filter output image, Canny filter output image ............................................ 35
Figure 37: r - e space representation of the line family that passes through point (X0, Y0) = (8,6) .................................................................................................................................. 36
Figure 38: The line families of 3 points are represented in r - e space and intersect; they are aligned ......................................................................................................................... 36
Figure 39: Detected lines (in red) using Probabilistic Hough Transform ................................. 37
Figure 40: Competition Buoys .............................................................................................. 38
Figure 41: Image treated with median blur and HSV conversion ............................................ 39
Figure 42: Experimental image (left) thresholded to keep orange pixels (right) ..........................................39
Figure 43: Input image (left), morphed using opening technique (right) to only keep the buoy pixels and its surface reflection ........................................................................................................40
Figure 44: Buoy detection output frame ....................................................................................................42
Figure 45: Buoy detection using color and digit recognition ....................................................................42
Figure 46: Detection of the digit and binary thresholding .......................................................................43
Figure 47: Image pipeline of the detection of the digit and binary thresholding ..................................44
Figure 48: Image of competition pipelines ..................................................................................................45
Figure 49: Original image taken with the camera in the experimental setup ..............................................45
Figure 50: Original image with grayscale equalization (left) and median filter of kernel size 7 (right) ..........46
Figure 51: Canny filtering of image in previous figure .................................................................................47
Figure 52: Probabilistic Hough line detection .............................................................................................47
Figure 53: Final line selection drawn on original image .............................................................................48
Figure 54: Line detection algorithm real time output .....................................................................................49
Figure 55: Competition mannequin ...........................................................................................................50
Figure 56: Person detection example ..........................................................................................................51
Figure 57: Output screens of computer vision interface for human detection ............................................51
Figure 58: Aquarium and waterproof case of the camera .........................................................................52
Figure 59: Low-light analog camera used ....................................................................................................53
Figure 60: Faulty video converter (left), new functioning one (right) ..........................................................54
Figure 61: Reflection problems on the side and surface ..............................................................................54
Figure 62: Detection occurs when the buoy is small and barely on screen, orange isolation on right ..................55
Figure 63: Digit Recognition Program Output ............................................................................................56
Figure 64: Unsuccessful digit recognition ..................................................................................................56
Figure 65: Successful pipe detection ........................................................................................................57
Figure 66: Inaccurate distance measurement (left), detection of the reflection (right) ...............................57
Figure 67: Unwanted detection of edges of the aquarium .........................................................................58
Figure 68: Human body detection in slightly murky water ..........................................................................58
Figure 69: Inaccurate detection due to body tilt ........................................................................................59
Figure 70: Frame of the dirty water without buoy (level 3) .........................................................................59
Figure 71: Slight buoy detection is observed .............................................................................................60
Figure 72: Buoy trust function is non-zero but confidence is low ...............................................................60
Figure 73: Real competition image provided by last year footage .............................................................61
Figure 74: Functioning digit recognition in level 2 water ............................................................................61
Figure 75: Edges are detected but distorted in level 3 water, lines are not detected .................................62
Figure 76: At closer range, edges appear straighter ....................................................................................62
Figure 77: Body detection in murky water yields good results .................................................................63
Figure 78: Graph of algorithm performance faced with different water purity levels ..............................63
Figure 79: Pipeline inspection with AUV ....................................................................................................64
Figure 80: Diagram of CNN for object recognition (an explanation is out of the scope of this TFG) .........65
Figure 81: Image segmentation, only 10 windows would have to be processed in this case .................66
Figure 82: List of tasks as displayed in GanttProject Program .................................................................71
Figure 83: Gantt Chart ...............................................................................................................................72
ANNEX 5: ABBREVIATIONS

**AUV:** Autonomous Underwater Vehicles  
**CNN:** Convolutional Neural Network  
**ERL:** European Robotic League  
**HOG:** Histogram of Oriented Gradients  
**HSV:** Hue Saturation Value  
**ML:** Machine Learning  
**RGB:** Red Green Blue  
**SVM:** Support Vector Machines