Politecnico di Torino

Master Degree in Mechatronic Engineering
A.a. 2022/2023
Sessione di Laurea Dicembre 2023

Sensors Selection for Obstacle Detection: Sensor Fusion and YOLOv4 for an Autonomous Surface Vehicle in Venice Lagoon

Relatori:
prof. Stefano Mauro
phd. Matteo Melchiorre
phd. Mauro Bonfanti

Candidato:
Simone ARGESE

Abstract

Developments in the field of autonomous navigation have led recent studies to focus on addressing the challenges of maritime transportation in cities with rivers and canals. In this context, the construction of an Autonomous Surface Vehicle (ASV) capable of navigating through complex environments like the canals of the Venice lagoon presents a significant challenge. This thesis aims to elucidate key aspects of the perception part of this project. Initially, a comprehensive analysis of potential sensitization techniques for autonomous vessels is conducted to understand the appropriate sensor architecture. Achieving optimal environmental perception places great importance on the selection of sensors such as LiDAR and stereo cameras. On the other hand, choosing the right sensor fusion approach in the context of the selected collision-avoidance system is challenging. In the literature, various classifications of sensor fusion algorithms exist, and many versions, ranging from traditional to innovative. Finally, an object detection algorithm based on YOLOv4 has been developed using a dataset of images collected in the Venetian lagoon through Google Earth Pro. The detector distinguishes three different classes of typical Venetian boats giving in output parameters, as bounding boxes coordinates, reliability scores and probability with respect to the class to which it belongs, which are essential to perform an optimal obstacle avoidance.

Table of contents

List of figures

List of Tables

1 Introduction

The consequence of population growth and tourism in cities has resulted in an increasing infrastructure, including transportation systems. Canals can serve as an alternative to land transportation for coastal and riverside cities, thereby alleviating roads traffic congestion. Traditionally, human-operated vessels have been employed for the transportation of goods and passengers within these urban canals. In such scenarios, the pilot assumes responsibility for route planning and vessel navigation, avoiding collisions with other vessels while adhering to regulations regarding collision prevention and speed limits.

At the same time, the advance of technology has enabled the development of unmanned systems (US) used in the water which can consider as an effective alternative. Autonomous electric boats are equipped with navigation and control systems capable of autonomously planning and following trajectories and are also equipped in order to sensing the navigation environment without the need for human presence. In this field, sensors and measurements have a crucial role in order to ensure that unmanned systems work properly, meet the requirements of the target application, provide and increase their navigation capabilities, and suitably monitor and gain information on several physical quantities in the environment around them [1]. The goal of these systems is to overcome many limitations coming from the human intervention satisfying different application task providing improved safety and reduced operational costs.

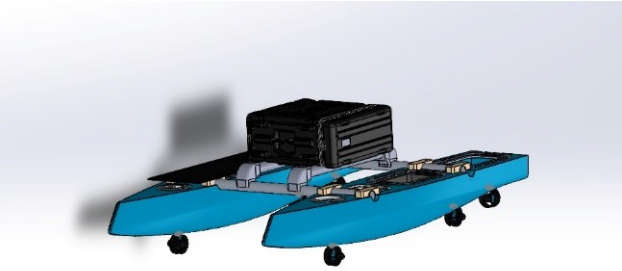


Figure 1 - Model of the MORElab vessel project

Widespread adoption of autonomous boats has the potential to significantly enhance canal efficiency. This thesis was conducted as part of a project at the Marine Offshore Renewable Energy Lab (MORE Lab) at the Politecnico di Torino. The primary objective of the project is to develop an autonomous electric boat prototype capable of navigation. This small vessel, with dimensions [2m x 1.5m], is intended to operate in complex environments: the canals of Venice. This unique and captivating environment, with its intricate waterways and numerous obstacles, presents an ideal challenge for autonomous navigation.

As said before, a critical aspect in achieving this goal pertains to sensor technology dedicated to collision avoidance. The sensors used to collect the data required by unmanned vehicles to operate can be very different in terms of their technology and performance, with different measurement accuracy capabilities needing to be taken into account. In a dynamic marine environment, the ability to perceive and react to obstacles is essential. My primary role within the team was to meticulously select sensors suitable for collision avoidance purposes, including Lidar and cameras, supported by a powerful processor for data processing. Subsequently, my focus shifted first to analyze the state of art related to the Sensor Fusion algorithms and then to the implementation of image object detection algorithms for the detection and monitoring of objects in the surrounding environment using depth cameras. This element is fundamental to ensuring safety during autonomous navigation, especially in complex maritime traffic situations.

In this context, it's worth emphasizing the pivotal role of machine learning and Convolutional Neural Networks (CNN) in the field of image object detection. These technologies play a crucial role in enabling the autonomous system to recognize and respond to objects and obstacles in real-time, contributing significantly to the safety and efficiency of autonomous navigation.

The work is developed in different sections: first, Unmanned Surface Systems (USV). Then, all the possible measurements needed by an autonomous system are defined, in particular for an USV. Next, collision avoidance theory related to our project are described showing the selected sensors for CA task, data acquisition, data processing and possible solution of sensor fusion. Finally, a part of the process of object detection is presented in order to show the results of the image object detection of different classes of boats produced in the case study of Venice canals through the YOLO (you only look once) algorithm.

2 State of art

2.1 Unmanned Vehicle

Unmanned Vehicles are a general reference to any kind of autonomous system. Although it is more automotive field oriented, according to the Society of Automobile Engineers (SAE) [2], there are six distinguishable levels of automation like shown in fig.[3].

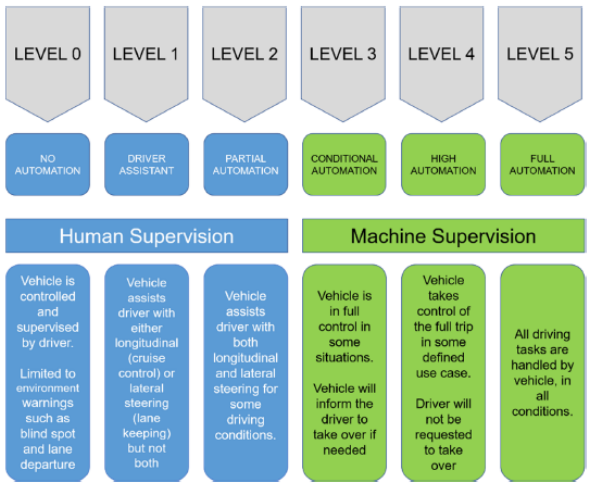


Figure 2 - Unmanned Vehicles Automation levels [3]

These levels could be perfectly adapted to the case of a vessel which has to navigate both in open sea or in canals.

2.2 Unmanned Surface Vehicle

USVs are autonomous vessels which operate on the water surface[1]. These vessels can be realized in different forms according to the specific work they will do. Although these systems may operate in complex environments or in difficult weather condition making the navigation dangerous and risky for human safety, their advantages are low maintenance costs and compact size.

At first UVSs were used in typical naval application, such as, for example, surveillance and reconnaissance, while, nowadays, their presence is growing also in civil application, like transport in open sea or, like in our case study of a prototype, able to transport people in complex traffic scenario as Venice canals. In order to be able to do their mission without any problem it is necessary a great perception to first detect and then avoid possible static or dynamic objects[1]. In general, these vessels have to navigate in adverse weather conditions such as rain and extreme wind. During the selection of the sensors this topic is taken in account but with tolerance related to the specific environment of our study case: Venice canals. There are a lot of differences in whether conditionings between navigation on the open sea and in small canals.

2.3 Measurements for Unmanned Vehicles

UVs performances are strictly related to the navigation capabilities and the physical quantities monitoring ability which are dependent to the sensor measurement system. In addition to the capacity for interpreting and analyze information coming from the sensors measurements, autonomous systems are also dependent on the capability of combination of information coming from different sensors, often of different type. The goal is to take advantage of their different optimal working ranges making better understanding of the surroundings. So, as a general rule of thumb, any UV has performances depending on how well they can perceive the environment around them. In this field, Obstacle avoidance is the main capacity for a safe navigation of these kind of systems.

It is important to highlight that the complexity of environmental conditions and the specific design constraints vehicles and requirements, which depending on the considered type of autonomous vehicle, makes the selection of UV measuring equipment a very difficult task.

2.3.1 Measurements for USVs

Going into the specifics of autonomous marine vehicles, sensor measurements are essential for USVs having a huge impact on their performances in particular because the marine environment can be affected by many disturbances, from wind to water reflection. In general, in the field of collision avoidance we can distinguish two kinds of measurements: active perception (Lidar or radar) and passive perception (Visual sensor)[1]. Passive sensors produce outputs receiving energy emitting from the surroundings. Conversely, active sensors emit energy into the environment measuring its “response” to that energy to produce outputs [4]. In this work particular attention has been devoted to the obstacle recognition and detection, representing one of the most important challenging requirements for achieving autonomous USV operation. The sourced sensors are selected in order to respect requirements related to the MORElab prototype dimension. In particular the choice made was to purchase top-of-the-line sensors as they are compliant to the prototype's requirements and can subsequently be reused for larger vessels if needed.

2.4 Collision Avoidance

The designing of the anti-collision system and the developing of a data fusion method for detecting obstacles for an UV is a very difficult task because it is based on the prediction of condition under which it will be operated starting from the assumption of the navigation environment.

2.4.1 Anti-collision System structure

As said before, these kinds of systems are based on the registration of many sensors. From a general point of view, the global path, such as the pre-programmed route, must be supplemented in real time with static obstacles and those in motion. Each of the sensors used to detect obstacles has a number of limitations related to range, detection resolution and optimal working conditions[5]. Proper selection of sensors allows

mutual complementation of the obtained information and thus as accurately as possible to determine the parameters of the obstacles necessary to avoid them in the safe way.

2.4.1.1 RADAR

The Doppler radar operates thanks to the Doppler effect, which causes variations in the frequency of waves when a wave-emitting object and a target object move relative to each other. In practice, the radar emits high-frequency radar waves through an antenna. When these waves hit a moving object, some of the wave's energy is reflected back to the radar, and the frequency of these reflected waves changes based on the speed of the moving object. A receiver inside the radar measures this frequency variation by comparing it with the frequency of the original radar waves. The difference between these frequencies allows the radar to calculate the speed of the moving object in the direction of the radar. The Doppler effect causes a shift in frequency, either upward or downward, depending on whether the object is approaching or moving away from the radar. These kinds of device in application like the one discussed in this thesis could have positive and negative aspects.

- Advantages:
    - Long detection range;
    - Good velocity estimates;
    - All-weather conditions;
- Disadvantages:
    - Skewed data from fast turning maneuvers
    - Limited small and dynamic target detection capability
    - Susceptible to high waves and water reflectivity

For our purposes we decided to do not choice a long-range radar because we do not test our prototype in the open sea. We decided to choose better Lidar and camera adding to them simple low-range parking radar sensors in order to guarantee the safest possible docking maneuver. In this way the collision avoidance system is based on a Lidar-Camera solution as it will be explained.

2.4.1.2 Lidar

The use of LiDAR sensors in the anti-collision systems of the autonomous navigation is fundamental for obtaining high accuracy, speed and range[5]. Its working principle consists in a laser rangefinder which scans the area in a plane transverse to the direction of motion measuring the distance between the LIDAR sensor and an object or surface. It operates by emitting high-intensity laser pulses in various directions. When these pulses hit a surface, some of the light is reflected back to the sensor. The amount of reflected light depends on the distance between the sensor and the struck object or surface. LIDAR measures the time it takes for the laser pulse to travel to the object and return, allowing for distance calculation. Time-of-flight measurements in different directions are used to create a three-dimensional representation of the environment, known as a 3D point cloud. Data collected by LIDAR is then processed to create detailed 3D maps. The avoiding obstacles process is strictly connected to an accurate representation of the environment. LiDAR data are the best way to analyze object orientation in 3D and for this reason it is the best sensor for the anti-collision system. While using LiDAR in real-time realization could be a difficult task, AI allows a very accurate visualization of the situation around the vessel being essential for USV navigation. It is possible to classify the LiDAR in different way. The LiDAR output data are in form of 3D and 360° cloud points for which x,y,z ,intensity and beam number have been determined. The position of each point is represented by the three coordinates while intensity parameter indicates the strength of the signal returning to the receiver. Instead, the beam number determinates in which beam the returned point is located[5].

- Advantages:
    - Obstacle detection range is good;
    - High resolution;
    - High depth range;
- Disadvantages:
    - Angular resolution both in vertical and horizontal direction;

LiDAR sensors can further be categorized as mechanical LiDAR or solid-state LiDAR (SSL). The first one is the most used among the environment scanning solutions. It uses the high-grade optics and rotary lenses driven by an electric motor to direct the laser beams and capture the desired FOV around the ASV while the rotating lenses can achieve a 360° HFOV covering the entire vessel surroundings. Contrarily, using Solid State LiDAR comports no rotating parts avoiding mechanical failures. SSLs use a multiplicity of micro-structured waveguides to direct the laser beams to perceive the surroundings [4].

Our choice is the mechanical Velodyne Puck which performs measurements in an angular range [H:360° V:30°], it has compact dimensions, low weight(830g) and low power consumption which making it correspond to the best choice for a small vessel. For what concern the surrounding registration it is performed with high accuracy through 16 channels, 100m long range and up to 600 000 points per second.

2.4.1.3 Stereo Camera

Cameras are one of the most adopted technologies for perceiving the surroundings. A camera works on the principle of detecting lights emitted from the surroundings on a photosensitive surface through a camera lens to produce clear images of the surrounding [4]. The stereo camera is a device that captures three-dimensional images using two photographic sensors placed at a certain distance from each other. This approach simulates how humans see the world, with two eyes providing different perspectives on a scene, enabling the perception of depth and three-dimensionality of objects. The process involves the simultaneous capture of two images, the search for common features between them, and the calculation of disparity, which is the difference between the positions of features in the two images. Disparity allows determining the distance between the camera and objects in the scene, thereby enabling the creation of a three-dimensional representation.

Stereo vision-based anti-collision systems detect objects that may interfere with the implemented path of the vessels. The differentiation of objects generated by this kind

of devices is related to their shape, color and texture [5]. Finally, through specific algorithms, also implemented in this work, it is possible to determine the kind of object that the control system has to take into account during maneuvers calculation to avoid it. Also, in this case it is possible to differentiate positive and negative aspects:

- Advantages:
    - High lateral and temporal resolution;
    - Simplicity of using;
    - Low weight;
- Disadvantages:
    - Low depth resolution and accuracy;
    - Challenge to real-time CNN implementation;
    - Susceptible to light and wheatear condition;

From the recap of advantages and disadvantages it is evident how LiDAR and Camera could compensate each other in particular in term of resolution and accuracy. Where one of them is worst the other can improve the object detection. In this way mounting these two sensors together is a fast way to increase system reliability.

Our choice is one of the best devices on the market, the Intel D455. It has also the infrared sensor embedded in it improving the general performance of the device. The use of infrared has as main goal improvement for environment perception applications, especially in unfavorable light conditions and night vision. Often, the combination of LiDAR and a stereo-camera with infrared sensor could bring to better performances. However, extreme weather condition, such as high temperatures, could dramatically compromise the performances[3].

2.5 State of art: Sensor Fusion

It is easy to generate data from a single independent sensor source but using it, alone, in a complex application as USV could be very difficult due to several limitations related to the sensors. Besides, these devices could suffer several inadequacies, depending, for example, from the nature of the sensed environment. To summarize,

limitations and inadequacies could comport a significant degradation of the performance due to many possible causes, such as drifting errors, low sensor resolution, surface irregularities and sensor' short-range [3]. To overcame this problem, information coming from multiple sensors are fused together generate a better understanding for USV.

Nowadays, Data Fusion and Information Fusion are synonyms terminology but it is possible to do a distinguish between them. The first is usually related to raw data which coming directly from sensors while the other one is more connected to already preprocessed data [6].

Literature presented many different approved definitions of this topic but the most recognized one is provided by the Joint Directors of Laboratories (JDL) workshop [7]: "A multi-level process dealing with the association, correlation, combination of data and information from single and multiple sources to achieve refined position, identify estimates and complete and timely assessments of situations, threats and their significance."

Briefly, the process of fusion could be resumed in the combining of the output of individual sensors or the outputs of specific algorithms generating a new combined outcome that is characterized by lower detection error probability and higher reliability bringing to more confident and robust representation of the environment.

Starting from the concept for which UVs are based on four different pillars: Sensation, Perception, Planning and Control, the first two steps are the place in which Sensor Fusion works to help the UVs. Sensation and perception are connected on two main topics such as Self-Awareness and Situational-awareness[8]. Based on the target topic the useful sensors are different. The anti-collision system described earlier in this chapter it is obviously part of the situational-awareness topic, depending on object detection, and is characterized from the already mentioned sensors. In chapter 3 there are the specific models of the sensors which will be used in the prototype vessel while

in chapter 4 there is the proposed approach to fusing the information coming from these sensors in the field of situational-awareness, and so of the collision avoidance.

2.5.1 Sensor Fusion algorithms classification

Assuming that Sensor Fusion is a multidisciplinary area which, involving many different fields, makes difficult establishing a clear and strict classification. Federico Castanedo in [6] describes a way, taking into account several studies presented in literature, to divide the employed methods and techniques starting from the following criteria:

I. Basing on input data source. These relations can be defined as
    a) complementary
    b) redundant
    c) cooperative data
II. according to the input/output data types and their nature, as proposed by Dasarathy [9];
III. Following an abstraction level of the employed data:
    a) raw measurement
    b) signals
    c) characteristics or decisions
IV. Depending on the architecture type:
    a) Centralized
    b) decentralized
    c) distributed

2.5.1.1 Classification Based on the Relation between the Data Sources

This kind of classification is at competition level and it aims to answer to the question about 'What' is fused. It takes into account the input data and what they analyze. As an example, considering two sensors, without taking into account if they have the same nature or not, in this classification became important understand what each of them scans. Classification criteria[6]:

a) Complementary: when input sources represent different part of the same scene making possible to have, through the information fusion a more complete global understanding.
b) Redundant: when multiple input sources are focused to give information about the same target incrementing, through data fusion, the confidence about it. In [8] it is also called competitive fusion among multiple sensors.
c) Cooperative: when the provided information is combined into more complex new information[6]. In [8] it is called coordinated fusion.

2.5.1.2 Dasarathy's Classification

One of the most recognized and known classification is the one explained by Dasarathy in [9]. It is composed in five categories as represented in fig. 3[6]

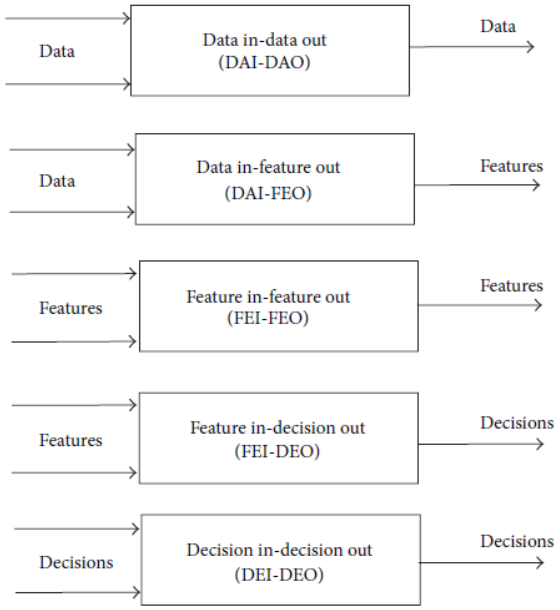


Figure 3 - Dasarathy's classification [6]

a) Data in – Data out: This type of data fusion process inputs and outputs raw data producing more reliable and accurate results. At this level, the fusion happened immediately after the data are gathered from the sensors;
b) Data in – Feature out: At this level there are still raw data at the input from the sources but the output is characterized by feature extracted to describe the entity of the environment;
c) Feature in – Feature out: with both input and output characterize by features. This level is also known as intermediate level fusion because it is a symbolic fusion, that means directly fusion of already preprocessed and extracted information;
d) Feature in – Decision out: this level obtain a set of features in input and provides decision in output. It is a high level of fusion.
e) Decision in – Decision out: It is a decision fusion which fuses input decision to obtain better and new decisions.

2.5.1.3 Classification based on the Type of Architecture

In [8] it is explained that for this kind of classification the goal, in this case, is to answer to the question about where the data fusion process will be performed. In other words, the fusion can be done from the main CPU or each sensor can do its detection and fusion autonomously. The difference among these situations is only the typology of architecture selected and this classification helps to resume all the possibilities:

I. Centralized architecture:
   The fusion node resides in the central processor that receives the information from all of the input sources[6]. All fusion processes are executed in this CPU using the provided raw measurements from the sources. Assuming correct data alignment and data association the centralized scheme could be considered as the optimal solution which reduce a lot possible incurring errors.
II. Decentralized architecture:

It is characterized by a network of nodes in which each of them has the processing responsibility without just a single point of data fusion[6] Each node receives information from its peers and elaborate them.

III. Distributed architecture:
In this architecture each source node processed independently the information before sending them to the fusion node. So, data association and state estimation are performed in the source node. In this way, each node performs and provide an estimation of the object state based on only their local views. This information is the input of the fusion node which provides a fused global view[6].

IV. Hierarchical architecture: result of a combination of decentralized and distributed nodes

In fig.4 [8] are represented the different architectures in a schematic way considering the three main possible sensors to take into account in the anti-collision system.

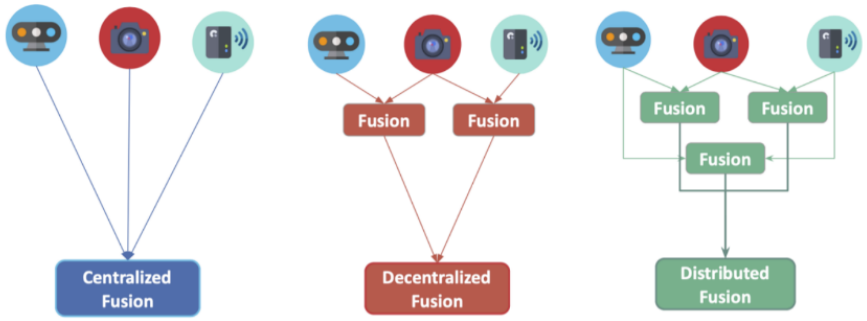


Figure 4 – multi-sensor possible architecture for Data Fusion

2.5.1.4 Classification based on Abstraction level

This classification is the most used in literature because of its intuitiveness. There are two possible sub-classification, one it is exposed in [6] for which there are four abstraction levels:

I. Signal level: directly addresses the signals that are acquired from sensors;

II. Pixel level: improve image processing task because operates at image level;
III. Characteristic: employs features that are extracted from the images or signal;
IV. Symbol: it is a decision level

Another possible classification at abstraction level, and also the common one, is described by [6][8]. For a better understanding of this classification following are used as example the case of the fusion between the two selected sensor for the anti-collision purposes of the MORElab ASV: camera and LiDAR. In particular, in [8] the classification aims to answer to the last possible question about when is better to do the fusion distinguish three different levels of algorithms:

I. Low Level Fusion (Early Fusion)
   It consists in the fusion of the raw data coming from multiple sensors. An example could be fusing the point cloud coming from the LiDAR with the pixels which compose the cameras images. But this fusion project is very expensive to make because it comports, taking into account the example, at each millisecond, the fusion of hundreds of thousands of points with hundreds of thousands of pixels. According to the fusion example of camera and LiDAR, which is the case of study of the chapter 4, usually object detection is used in this process but with the addition of 3D point clouds projection into a 2D image allowing an easy association with the pixels of the image. Fig. 5 [8] schematize this level.

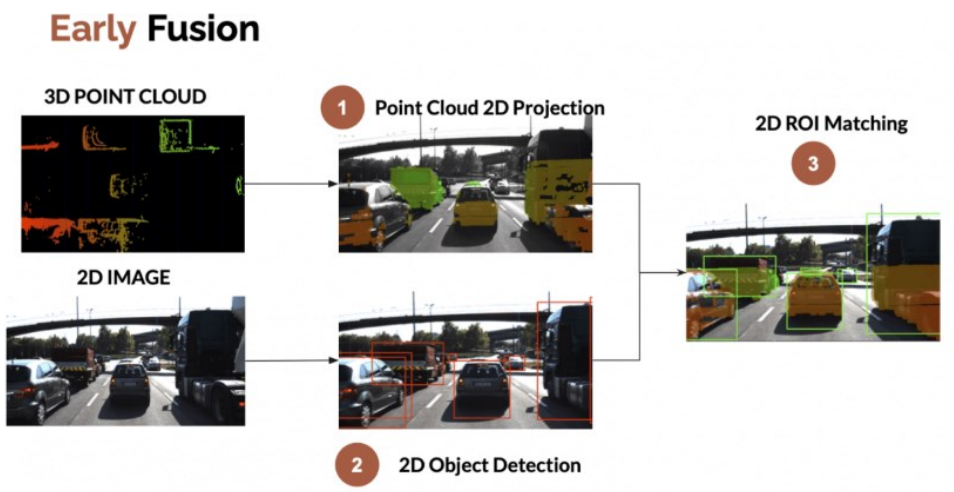


Figure 5 - Low level Sensor Fusion Algorithm: example of Camera and LiDAR

II. Mid Fusion Algorithm (Late Fusion)

It consists in the fusion of the independently detected objects on sensor data. Taking into account the used example, the object detection done by the camera are then fused with the detection of the same object done by the LiDAR making a better estimation of the state and class of it. Using traditional approaches one of the most used is the Kalman filter fusion approach or a Bayesian algorithm, it will be introduced in the first paragraph of the fourth chapter within the description of the possible sensor fusion approaches. Although late fusion relies heavily on detectors, because the failing of one of them can comport the whole fusion failing, this process is the easiest to understand and, moreover, it contains several existing implementations [8].

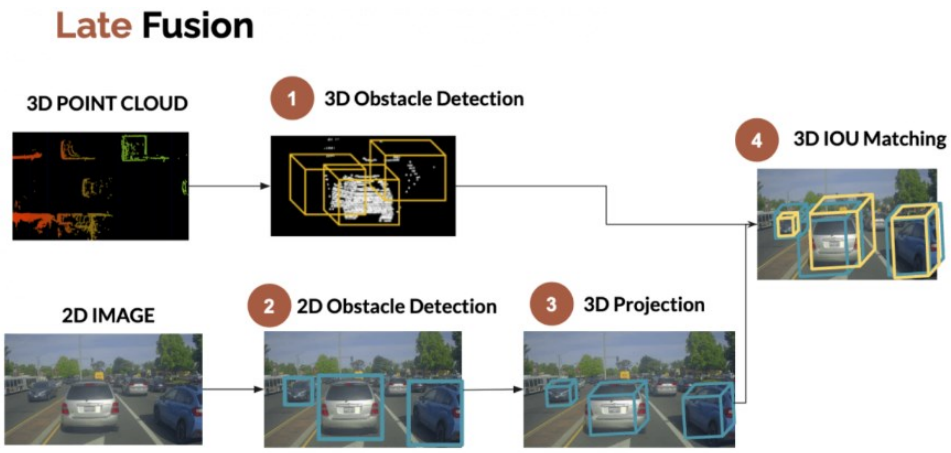


Figure 6 - Mid Level Fusion Algorithm: Camera and Lidar

In fig.6, there is the fusing of 3D bounding boxes from a LiDAR with a 2D bounding box from an object detection algorithm. However, the process could also be reversed according to the project needing or the development preference.

III. High Level Fusion – Fusing the tracks

This level of fusion is related to the fusion of both objects detected and their tracks, not only relying on detections, but also on prediction of future states of the objects tracking them [8]. Being an high level approach its simplicity is obviously an advantage but the worst aspect of this approach is the high probability of losing information.

2.5.2 Selected Fusion approach

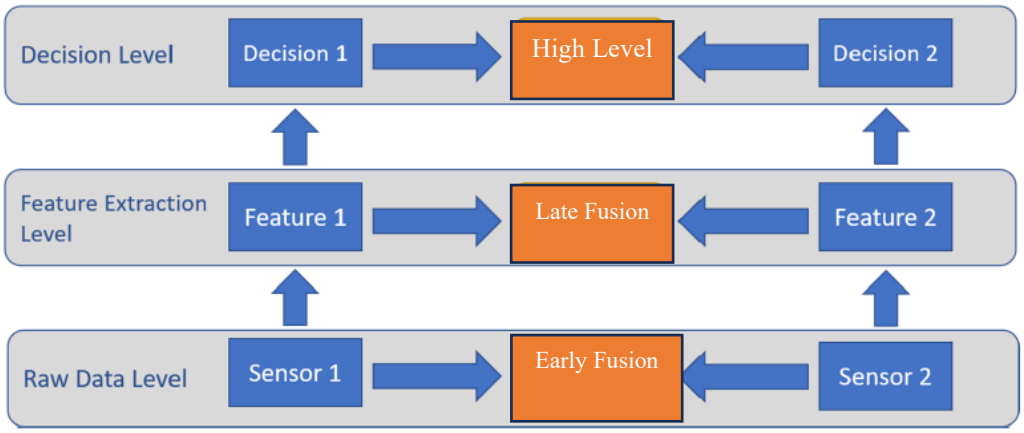


Figure 7 - Abstraction Level classification recap

In chapter four it is descripted the selected possible approach that is a late fusion. This choice is done by considering the general advantage and disadvantage coming from all possible approaches explained during this chapter.

3 ASV Sensor selection for anti-collision

According to what it is described in the paragraph 2.4, the anti-collision system of any kind of autonomous vehicle is based on three main sensors: Radar, LiDAR and Camera. As said before, taking into account the purposes of the MORElab prototype in this early development phase, the anti-collision system will be composed from LiDAR and Camera which will be helped by the low-range detection of simple parking Radar. This choice was made on the basis of a cost-benefit analysis for a small prototype. These radars will be helpful, in particular, during parking maneuvers while LiDAR and Camera will be used to detect objects and, through a sensor fusion algorithm described in chapter four, to indicate some parameters, like obstacle dimensions or shapes, and coordinates then used by the control system to generate the final safety path. In any case, if needed, the integration of a more powerful type of radar will be easily achievable both from a hardware and data processing software perspective, as well as for sensor fusion algorithm chosen. In particular, about this last topic, chapter 2 has presented a lot of possible configurations that could be helpful during a radar integration in the sensor fusion process. An example could be using a hybrid version of a decentralized fusion represented in fig.4 which presents the fusion between the Camera and Lidar already available with the new data coming from the Radar. Unfortunately, a clear way to approach this problem could be find only after a specific data collection campaign which can emphasize possible needing of the specific ASV system.

Finally, the last selected device is a GPU which can help to process the data coming from the sensors allowing a faster and accurate global detection of obstacles.

3.1 LiDAR: Velodyne Puck

There are multiple specifications taken in consideration during the choice process. This LiDAR presents one of the best measurement ranges on the market with a high range accuracy. There are, in particular, two main aspects which differentiate this device from the other of the same family, like the Puck Hi-res: the vertical field of view (VFOV) and the vertical angular resolution. This LiDAR version exhibits better performances in the VFOV at cost of reduced angular resolution. At the end of the selection, it was preferred to choose a LiDAR with better field of view. This choice is made considering the final environment in which the vessel has to navigate and the fact that it is characterized by a complex traffic scenario, with many different boats. The Puck is a mechanical LiDAR which consists in a 360° horizontal field of view (HFOV) with only one device which rotates at a nominal rate of 15Hz. Moreover, it presents a high horizontal resolution which, combined with the vertical one, makes it as the best possible sensor for an accurate environment understanding. In LiDAR systems, "channels" refer to the number of laser detection channels in the device. Each channel represents a laser emitter and a receiver that work together to generate a laser beam, emit it into the environment, and detect the return of the beam after it has been reflected by objects in the environment. A LiDAR with more channels can emit and receive multiple laser beams simultaneously, increasing its ability to capture data efficiently and accurately. Each channel may cover a specific portion of the LiDAR's vertical field of view, allowing for broader coverage of the environment. For example, a "16 channel" LiDAR, as in this case, has 16 laser detection channels, which means it can emit and detect 16 laser beams simultaneously. This allows for a higher data point density and better spatial resolution, making the device suitable for applications that require detailed environmental sensing, such as autonomous driving and object detection. Another presented possibility consists in the better Velodyne LiDAR possible, which has 32 channels but, for the purposes of the project, they are considered too much form a cost-benefit analysis.

Figure 8 - Velodyne Puck

For what concern the mechanical and electrical aspects many considerations had done. From both electrical and mechanical point of view the hi-res puck presents low dimensions, low weight and low power consumption making it as great option for a small ASV prototype. Moreover, it has the environment protection IP67 which guarantees good performances also in a complex environment as the marine one. Finally, the operating temperatures is perfectly compatible with the temperature ranges which characterize the Venice lagoon.

	HFOV	VFOV
FOV	360°	30°
	HORIZONTAL	VERTICAL
ANGULAR RESOLUTION	0.1° - 0.4°	2°
	RANGE	ACCURACY
DEPTH	100m	± 3 cm

Table 1 - Puck Hi-Res performance specification recap

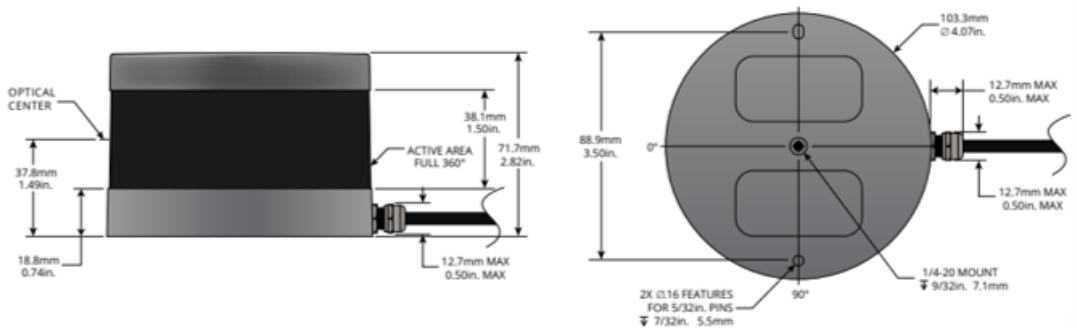


Figure 9 - Puck Hi-Res mechanical design [10]

	Operating Voltage	Power consumption
Consumption	9 V – 18 V	8W
	Weight	Dimensions
Design parameters	830g	71.7 mm x 103.3 mm (d)
	Operating temperature	Storage Temperature
Temperature	-10°C to 60°C	-40°C to 105°C

Table 2 - Puch Hi-Res mechanical specification recap

3.2 CAMERA: Intel D455

The specifications of this camera model are perfectly compatible with the other cameras in the same segment on the market except for what concern the presence of the infrared sensor which gives better performance in the depth reconstruction available through this stereo camera, if it is wanted. Having this kind of technology aims the possibility to use the Intel D455 for other purposes, in future improvements of the project, achieving better reconstruction of the environment. The working principle are sightly different from the other stereo-camera because of the using of the structured lighting to capture 3D data. It consists in the projection of a series of infrared light patterns into the surrounding environment. These patterns are captured by the cameras

and used to calculate the distance between the camera and the object in the environment. The disparity information created through the stereo-camera principle exposed in chapter 2 between the two images are used to create a depth map of the environment. In conclusion, the depth map and captured images can be used to perform various tasks as, for example, object detection.

Figure 10 - Intel D455

The D455 has a large HFOV and a good VFOV achieving a very large diagonal field of view (DFOV) of 90°. The resolution is full HD for RGB images. From a mechanical point of view the device has low weight, low dimensions and low power computation making it, as for the Puck Hi-Res, a great option for the development of a small ASV.

	HFOV	VFOV	DFOV
FOV	87°	58°	90°
RESOLUTION	1280x720 (Full HD)		
	RANGE	ACCURACY	
DEPTH	0.6 – 6 m	<2% (up to 4m)	

Table 3 - Intel D455 performance specifications

	Operating Voltage	Input Current
Consumption	5 V	700 mA
	Weight	Dimensions
Design parameters	393g	110x110x71.6 mm
	Operating temperature	
Temperature	0°C to 50°C	

Table 4 - Intel D455 Mechanical specification

3.3 Nvidia Jetson AGX Orin 32 GB

The processes of object detection and tracking are based on Deep Learning. The NVIDIA Jetson AGX Orin 32GB is a powerful embedded processing system designed for artificial intelligence (AI) and deep learning applications. It is one of the latest additions to NVIDIA's Jetson family and is known for its high computing capabilities and advanced AI support.

Figure 11 - Nvidia Jetson AGX

The NVIDIA Jetson AGX Orin 32GB is a robust embedded processing system renowned for its potent capabilities in artificial intelligence and deep learning applications. It leverages the cutting-edge NVIDIA Ampere compute architecture, featuring an 8-core ARM Cortex-A78 CPU, an NVIDIA Ampere GPU with 2048 CUDA cores, and 32GB of LPDDR5 RAM. This device excels in supporting advanced AI functions such as image recognition, natural language processing, speech recognition, and autonomous robotics, even handling intricate machine learning workloads. It boasts versatile connectivity options through various ports like USB, Ethernet, HDMI, and PCIe, making it suitable for linking external devices such as sensors and cameras. Running on an NVIDIA Linux-based OS, it provides a solid development environment and extends compatibility with a broad spectrum of deep learning libraries and frameworks. NVIDIA further enhances the platform with a suite of software tools, including NVIDIA CUDA, TensorRT, and cuDNN, optimizing machine learning software and its execution. The Jetson AGX Orin finds its applications in a wide range of fields, with a particular focus on autonomous vehicles, robotics, computer vision systems, and industrial automation.

	Operating Voltage	Power Consumption
Consumption	5 V	15W
	Weight	Dimensions
Design parameters	300g	100x87 mm
	Operating temperature	
Temperature	0°C to 50°C	

Table 5 - NVIDIA Jetson AGX Orin 32GB mechanical specifications

3.4 Radar parking sensors

The selected devices are the Valeo been&park which are characterized by the sourced depth low-range, low consumption and tolerable operational temperature range.

V	P	A	Operational temp
12.5V		<250mA	-40° to 85°

Table 6 - Parking Sensor relevant mechanical parameters

3.5 Recap Tables of the chosen sensors

An important aspect to take into account is the fact that each sensor has been selected with an IP certification conformable with the their working environment.

Name	HFOV	VFOV	DFOV	Depth acc.	Depth range	Resolution	Angres	range acc
Puck Hi-res	360°	30°		0.03 m	100m		1.33 °	∓3cm
Intel D455	87°	58°	90°	<2% 4m	0.6-6m	1280x720		
Valeo beep&park 632203					0.1-1.7m			

Table 7 - Sensor relevant performance parameters for object detection

Name	V	P	A	Operational temp	W	dimensions(mm)
Puck	12V	8W		-20 to 60°	830 g	103 (d) x 72
D455	5V		700mA	0-40°	393 g	110x110x71.6
JETSON AGX ORIN 32GB	5V	15 W		0-50°	300 g	100x87
Valeo beep&park 632203	12.5V		<250mA	-40° to 85°		

Table 8 - Relevant Mechanical Parameters

Name	Signal format	Communication interface	Data acquisition frequency
Puck	.pcd,.las,.bin	Ethernet 100Mbit/s	0.3 million point/s
D455	raw,ply,bag,png,csv,bin	USB 5Gbit/s	30 frame per sec
JETSON AGX ORIN 32GB		USB,Ethernet,interfaccia MIPI CSI	

Table 9 - Other relevant parameters

3.6 Overview on the global electronic architecture of the ASV

This section aims to describe, in a schematic way, the whole electronic architecture with emphasis on the sensors of the anti-collision system. The goal is to show how these sensors are connected in the global hardware architecture of the ASV.

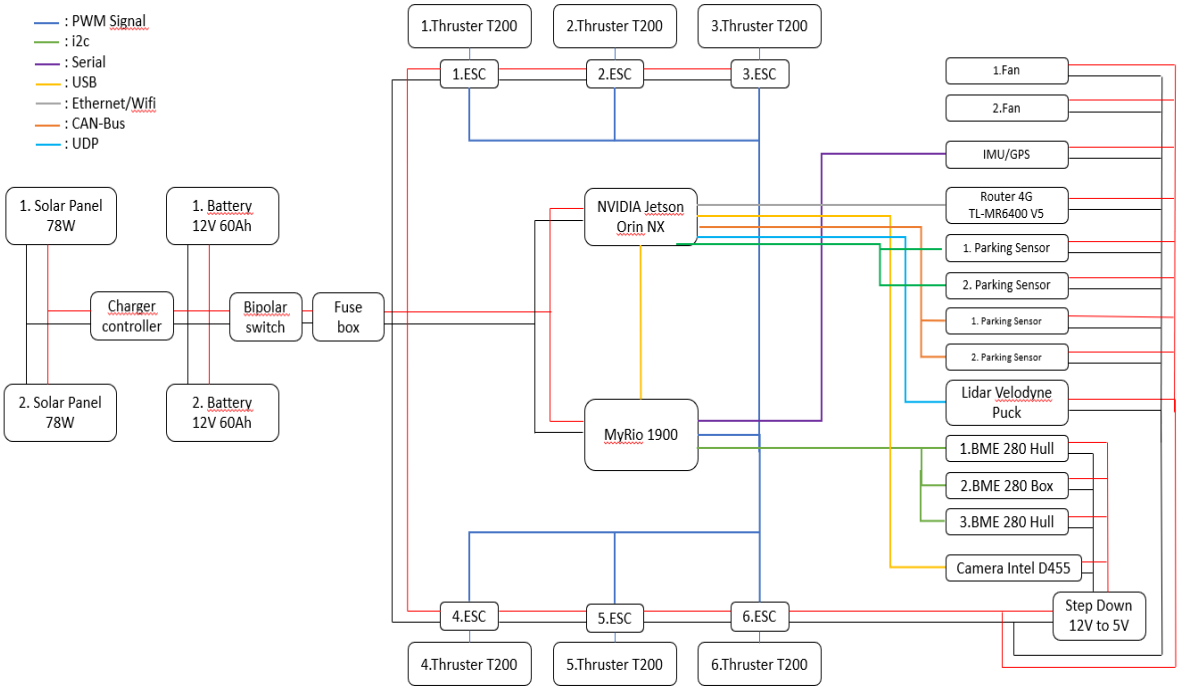


Figure 12 - ASV Electronic Architecture

From the legend it is possible to understand which are the communication interfaces involved in the scheme. For what concern the anti-collision system sensors the possible interfaces related to each sensor are resumed also in table 9.

The Lidar, the parking sensors and the NVIDIA are supplied by 12V while the Intel D455 is supplied from 5V after the application of a step down (12V to 5V). The scheme shows how all the sensors related to the collision avoidance and obstacle detection are connected to the Jetson processor because, as already said, it performs with better accuracy then the MyRio deep learning algorithms which are fundamental in the field of the object detection. Instead, the MyRio will be responsible of the management of the thrusters' action, temperature and pression sensors measurements analysis and for the localization path, through the IMU/GPS information processing.

4 Sensor Fusion

This chapter illustrates a possible method to fuse information coming from the LiDAR and the Camera sensors. The goal of the resulted fusion is supplying the collision avoidance system allowing the control system avoiding detected obstacles. The chapter is organized in order to, first, give a general overview of the possible actionable strategies and then describe a chosen one based on the selected Puck and Intel D455.

In literature there are several papers based on two main approaches, for this reason it is possible to divide sensor fusion techniques in classical algorithms and deep-learning-based algorithms.

4.1 Traditional Sensor Fusion Approaches

Numerous conventional algorithms are available for implementing data fusion in applications that necessitate dealing with data imperfections, such as inaccuracies and uncertainties [3]. These algorithms employ various techniques based on uncertainty theories, as depicted in Figure 13 [3].

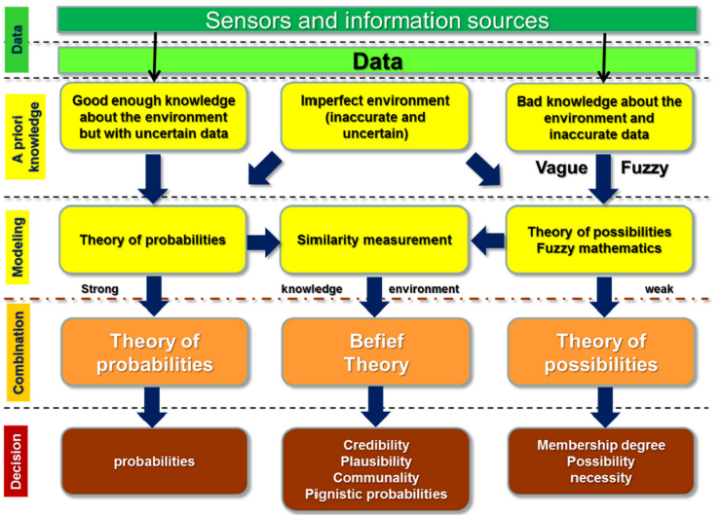


Figure 13 - Theories of uncertainty for modelling and processing of different levels of data imperfection

From a high point of view, theories of uncertainty in sensor fusion consist to methods and approaches used to address uncertainty in data and information collected from various sensors and sources. These theories provide a framework for handling and integrating imperfect or uncertain data coherently to obtain a more accurate and reliable estimation of what is happening in the surrounding environment. Some common techniques encompass probabilistic, statistical, knowledge-based (including fuzzy logic and possibility), interval analysis, and evidential reasoning methods. Figure 14 [3] provides an overview of the different variations within each category.

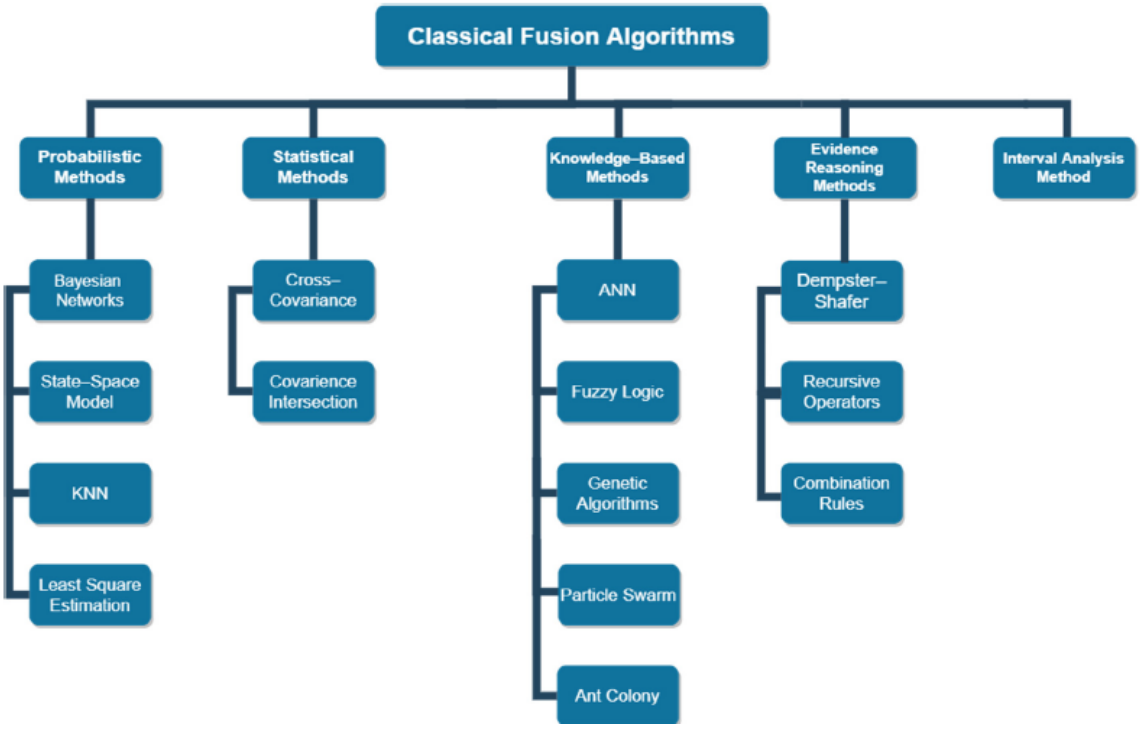


Figure 14 - Traditional Approaches for Sensor Fusion algorithms

The functionalities of each of these theories can be synthetized as:

- Probability Theory: This theory is based on the use of probability distributions to represent and manage uncertainty in data. Probability is used to quantify confidence in the results obtained from sensors.

- Statistical Methods: Statistical approaches leverage techniques such as parameter estimation, residual analysis, and error analysis to assess uncertainty and improve the accuracy of estimates.
- Knowledge-Based Methods: These methods incorporate expert or knowledge-based information, such as fuzzy logic and possibility theory, to handle uncertainty in a more flexible manner.
- Interval Analysis: This approach uses intervals to represent uncertainty in data, allowing the definition of ranges of possible values rather than single points.
- Evidential Reasoning: This approach is based on Bayesian reasoning and the use of evidence or proofs to combine information from different sources coherently, considering the uncertainty associated with each source.

For your reference, Table 10 in this section offers a concise overview of these classical algorithms compatible with a low/mid-level of fusion, highlighting their respective strengths and weaknesses.

Algorithm Methods	Characteristics	Advantages	Disadvantages	Applications	Fusion Level
Statistical	Employed to improve data imputation by applying a statistical model to represent sensory information	Able to manage unknown correlations	High computation and limited to linear estimation	Estimation	Low
Probabilistic	Information coming from sensors are elaborated through a	Uncertainty is handled also in nonlinear systems through	Needing of prior knowledge	Estimation/Classification	Low/Mid

	probability representation	particle filters, UKF or EKF			
Knowledge-based	Computational intelligence approaches	Handle complex non-linear system	Expertise knowledge dependency	Classification/decision	Mid/High

Table 10 - A comparison among traditional sensor fusion algorithms

4.1.1 Probabilistic Method: Bayesian Network

Bayesian Networks, often referred to as belief networks or causal-Bayesian networks, are a powerful and flexible tool in the context of probabilistic methods in sensor fusion. They are commonly considered as the most used approach among the traditional methods exposed in figure 14. These networks provide a graphical representation of probabilistic dependency relationships between random variables, see figure 15. In the context of sensor fusion, random variables can represent data from different sensors or information about the state of a system.

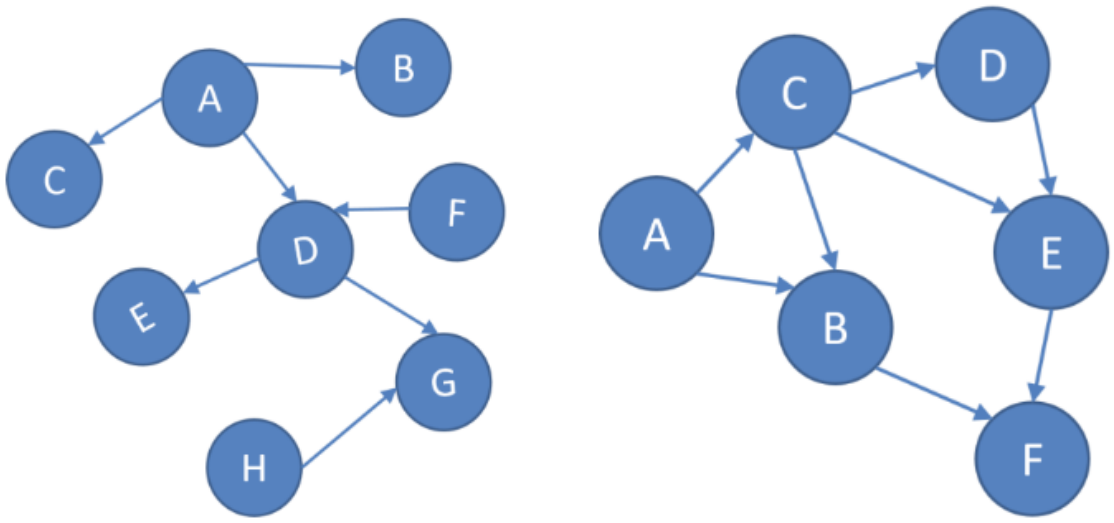


Figure 15 - Bayesian Network graph example: a) singly-connected network b) multiply-connected network

In a Bayesian Network, the nodes of the graph represent these random variables, while the edges between nodes represent the probabilistic relationships between them. For

example, in an autonomous driving application, there are nodes representing the vehicle's position, Lidar measurements, and camera data. The edges between these nodes reflect how these random variables are correlated.

The power of Bayesian Networks lies in their ability to model uncertainty and perform probabilistic inferences. This means that these networks can be used to calculate the probability of random variables given sensor measurements.

Furthermore, Bayesian Networks allow for the propagation of uncertainty through the graph. This means that having uncertainty in sensor measurements, the network can take it into account and provide estimates based on this uncertainty. This is particularly useful in situations where sensor data may be incomplete or noisy.

In summary, Bayesian Networks provide a mathematical framework to represent, model, and manage uncertainty in sensor data, allowing for more accurate and reliable estimates in sensor fusion applications.

4.1.2 Kalman Filters

Kalman filters represent a fundamental approach in sensor fusion, especially when it comes to integrating data from different sensors, such as LiDAR and a camera, to achieve more accurate and reliable estimates. These filters are widely used in the field of sensor fusion due to their ability to handle data uncertainty and provide optimized estimates. The Kalman filter can be employed to combine this information into a single probabilistic model, taking sensor uncertainties into account. The filter estimates the system's state, such as the position and velocity of objects in the environment, optimally considering sensor measurements and their probabilistic relationships. This allows for more accurate and consistent estimates while reducing the impact of data uncertainty. Furthermore, Kalman filters can be extended to address more complex situations, such as tracking moving objects over time. This is especially valuable in applications like autonomous driving and, consequently, in the case of Autonomous Surface Vehicles (ASVs).

The Kalman filter (KF) is considered an optimal linear estimator in cases where both process noise and measurement noise can be adequately described by white Gaussian noise. The KF relies on the first two moments of the state (i.e., mean and covariance) in its update process. However, when dealing with nonlinear problems or non-Gaussian noise affecting the signals, the Kalman filter may offer a suboptimal solution. To address nonlinear challenges, the extended Kalman filter (EKF) comes into play. The EKF is based on the concept of linearization of the state transition matrix and the observation matrix using Taylor series expansions. By assuming that all transformations are quasi-linear, the EKF effectively linearizes the previously nonlinear transformations and substitutes Jacobian matrices for these linear transformations in the KF equations. It's important to note that this linearization approach can lead to subpar performance and even divergence for highly nonlinear problems. An improvement over the extended Kalman filter is the unscented Kalman filter (UKF). The UKF approximates the probability density resulting from the nonlinear transformation of a random variable by evaluating the nonlinear function with a minimal and thoughtfully selected set of sample points. The posterior mean and covariance, estimated from these sample points, maintain accuracy up to the second order for any nonlinearity [11].

4.1.3 Camera Lidar Fusion with KF traditional method: Operative Example

In [12] it is presented a traditional fusion strategy which is replicable also in the case of the MORElab ASV. The architecture of the possible sensor fusion could be resumed as:

1. Targets Data Acquisition:

   It consists in the data collection through the ASV with the final setup about the positioning of the sensors.

2. Camera Image Processing:

   Image Object Detection in images captured by camera with classification and boundary of targets. Also centroid of them are obtained and collected.

3. Projection:

   Lidar's point cloud has to be projected from 3D to 2D images procured by camera.

4. LiDAR frame processing:

   Segmentation and detection of the obstacle centroid in the cloud.

5. Data Association:

   Fundamental step to ensure that the same obstacle is being detected by the two sensors.

6. Bayes Fusion:

   This step aims to create a better measurement with less noise.

7. Estimation Filter:

   Obstacle Path Estimation using Kalman Filter.

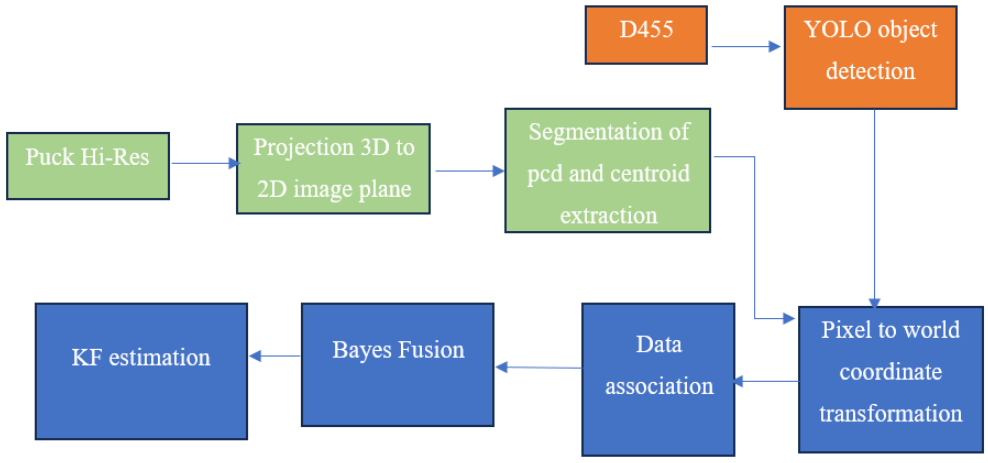


Figure 16 - High level scheme of a traditional approach to fuse D455 and Puck Hi-Res

There are many variants of object detection algorithms to process images coming from a stereo-camera, in figure 16 YOLO is the selected one because it will be analyzed in details in chapter 5 achieving object detection purposes. The steps related to the LiDAR system could be done, as for the camera, using deep learning and neural network for both process of ground segmentation and features extraction while the first passage

related to the projection is easily makeable through any computer vision software (MATLAB/Phyton).

After the process of data acquisition and processing, before doing fusion, it is important to be sure that the same object is detected by different sensors. For this reason, the data association became essential to make sure that the data coming from different sources belong to the same frame of reference. The sensor alignment is a process done off-line. Its input are sensor data while the outputs correspond to the calibration parameters, which are rotation and translation matrix. This process is also known as multi-sensor extrinsic calibration that allows the estimation of a relative general point in a common plane among n sensors. In [13] there is a deep explanation of the automated common way to perform an extrinsic calibration among sensors. In brief, the extrinsic calibration using a chessboard consists in a process that automatically determines calibration parameters among sensors. This process involves using a checkerboard pattern positioned in various locations and capturing images of it from different angles. The images are then analyzed to identify key points on the checkerboard. Through these correspondences between key points in different images, the software calculates extrinsic calibration parameters, such as rotation and transformation matrices, that describe the relative position and orientation of the sensors in a common coordinate system. Figure 17 shows a representation of this process. Moreover, could be relevant highlight that there are many solutions which, using recent development of deep Convolutional Neural Network, can achieve better calibration results at cost of more computational effort.

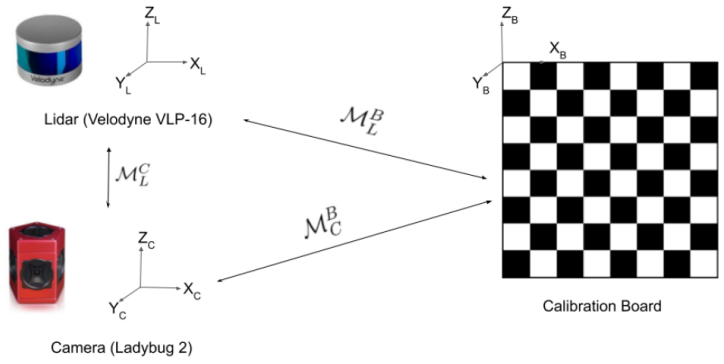


Figure 17 - Extrinsic calibration using checkboard[14]

To efficiently perform sensor fusion, the data coming from the sensors have to be processed through the calibration process. To carry-out this process, the calibration parameters produced by the extrinsic calibration process are fundamental. These parameters are the rotational matrix and the translation matrix.
Here's how they work:

- Rotation Matrix: A rotation matrix is used to describe rotations in space. In the context of sensor fusion, this matrix takes into account the relative orientation between sensors. For example, having camera oriented in one way and a LiDAR oriented in a different direction, the rotation matrix captures how the camera's orientation differs from that of the LiDAR. By applying this rotation matrix to the camera's measurements, it is possible to transform them into a common coordinate system with the LiDAR.

- Translation Matrix: The translation matrix, or transformation matrix, represents translations in space. This matrix captures differences in position between sensors. For instance, if LiDAR is mounted at a certain distance from the AVS's center, and the camera is mounted in a different position, the transformation matrix accounts for these differences in position. Applying this matrix to the camera's

measurements, it is possible spatially align the data in a common coordinate system with the LiDAR.

Together, these matrices allow to transform sensor measurements into a common reference system. For example, it is possible apply the rotation matrix first to align the orientation and then the translation matrix to align the position. This ensures that data from different sensors are aligned in the same coordinate system, enabling accurate information fusion.

Finally, take place the presented techniques of sensor fusion as Bayes fusion, to reduce noise, and estimation filtering through KF.

4.2 Hints of Deep Learning: CNN based Sensor Fusion approach

From the example shown in 4.1.3 it is noticeable how deep learning became a relevant aspect also for the new versions of traditional algorithms. In that case, both image and point cloud processing are based on the capability of object detection through several possible detectors and, often, the choice is a DL-based detector, just like in the example. For this reason, and for a better understanding of the potentiality of the image object detection algorithm that will be presented in chapter 5, now there will be introduced many hints related to the deep-learning in the Sensor Fusion field.

Deep learning could be considered as an improvement of the neural networks. It is part of the artificial intelligence and machine learning field aiming to reproduce human brain functionality. The algorithms involve the creation of multiple layers networks, allowing them to process raw data coming from source sensors and extract certain patterns useful to perform complex task [3] as, for example, object detection.

The frequently used deep learning methods can be listed as in figure 18 [3].

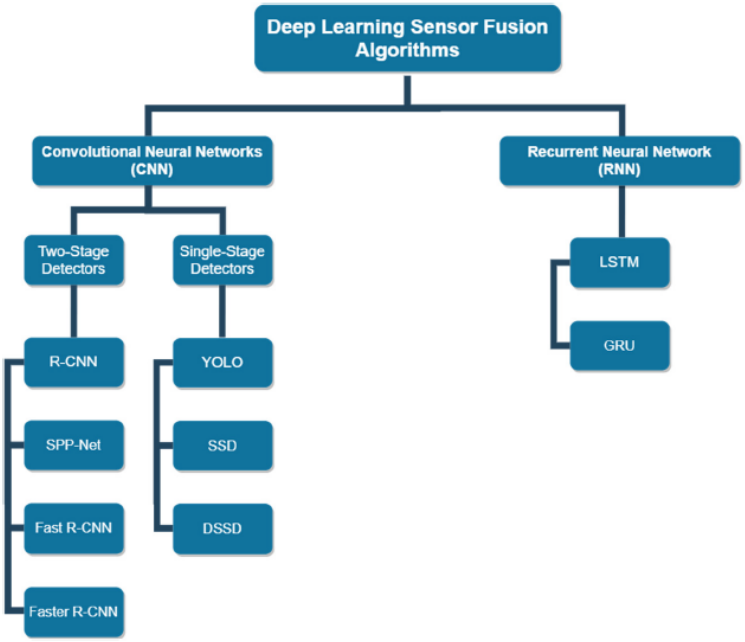


Figure 18 - Common Deep Learning algorithms used in autonomous guidance[3]

Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are two types of deep neural networks with different purposes. While CNNs are primarily used for processing structured grid data, such as images, RNNs are specialized in analyzing sequential data, such as text or time series. CNNs use convolutional layers to identify spatial patterns in the data, like edges, textures, or objects in images. The connections between neurons in a CNN are local and shared, meaning that each neuron is responsible for only a small portion of the input. This architecture is ideal for computer vision applications like image recognition and object detection. RNNs, on the other hand, are designed to handle sequential data, leveraging recurrent connectivity that allows them to consider the current input along with the previous time steps. This structure makes them suitable for problems involving temporal relationships, such as speech recognition or automatic translation. Both architectures have some limitations: CNNs struggle with complex sequential data, while RNNs can suffer from vanishing gradient problems and are less effective at processing structured grid data.

Before the advent of CNN in 2012, the choice for image recognition and classification was the multilayer perceptron (MLP). MLP is a type of feed-forward, fully connected neural network composed of an input layer, a hidden layer, and an output layer. However, as recent progress in the field has revealed, MLP has several shortcomings, rendering it insufficient for the following reasons [3]. Firstly, MLP suffers from an increasing number of parameters that necessitate training. Secondly, it fails to retain crucial spatial information and the pixel layout of an image. Lastly, it lacks the ability to maintain consistency under translations, meaning it's not translation-invariant [3].

On the other hand, CNN uses convolution operation to process pixels in image. One of the main differences with respect to the MLP is the architecture which presents layers organized into three dimensions of width, height and depth. Moreover, the neuron of CNN is not fully connected to the layers [3]

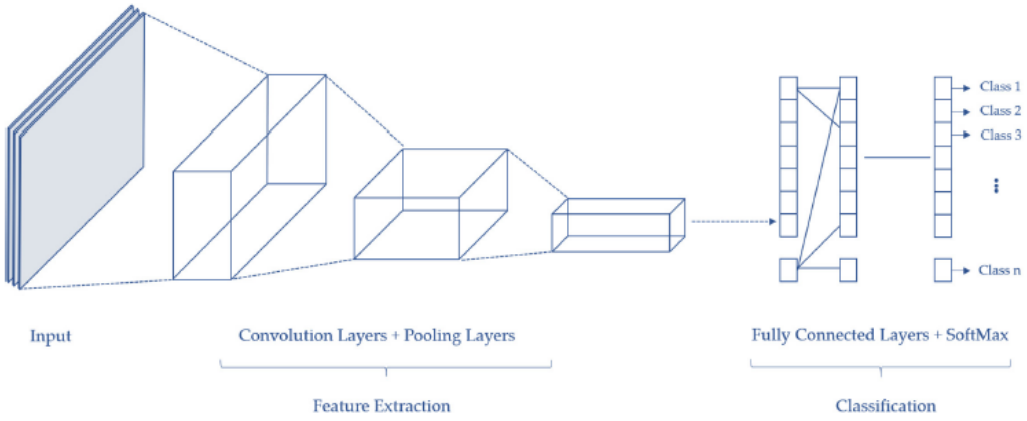


Figure 19 - CNN processing layers for object detection and classification

A general CNN usually is structured as in figure 19 where:

- Input Layer: It is a layer that contains data coming from sensors
- Convolutional Layers: they perform convolutional operation to extract important feature from the image.
- Pooling Layers: their work is in between two convolutional layers and it consists in minimizing the computational cost reducing, at the same time, some spatial information of the convoluted image.

- Fully connected layer: It has the role of a classifier connecting all weights and neurons
- Output layer: Storing the final output which consists in the probability of each input information belonging to a specific class.

In the next chapter will be discussed in more details the functionalities of each layer taking as reference a real algorithm coded to accomplish an object detection task.

4.2.1 Camera Lidar Fusion with CNN data processing: Operative Example

The operative example shown in 4.1.3 is fundamental to understand from a high point of view how to perform each step of a sensor fusion architecture between two sensors. However, an important aspect to emphasize is that, nowadays, the task related to the processing of the sensors data are based on CNN algorithms. So, step 2 and 4 in figure 16, which consist in the detection task of camera and LiDAR, are strongly dependent to these kinds of algorithms.

4.2.1.1 Camera Detection

Image Object detection isn't an easy task because it is not known in advance how many objects will be in each input image and which is their dimensions [15]. Often Artificial Neural Networks are the best way to solve this task. The advent of deep learning (DL) has revolutionized object detection methods, surpassing many traditional techniques in terms of both accuracy and speed. In the realm of DL-based image object detection, two main approaches are commonly used.

The first approach revolves around region proposals, exemplified by Faster R-CNN. This method begins by processing the entire input image through convolutional layers to generate a feature map. Subsequently, a dedicated region proposal network utilizes these convolutional features to suggest potential regions for detection. The final part of the network is responsible for classifying these proposed regions. However, this architecture, consisting of two parts for bounding box prediction and classification, can significantly reduce processing speed.

The second approach employs a single network for both region prediction and label classification, as demonstrated by You Only Look Once (YOLO). In the case of YOLO, the input image is divided into coarse grids, and each grid is associated with a set of base bounding boxes. YOLO predicts offsets from the actual location, a confidence score, and classification scores for each base bounding box, indicating whether an object is present within that grid location.

4.2.1.2 LiDAR Detection

In the realm of LiDAR detection, the primary challenge lies in classifying points within a sparse 3D point cloud. Various methods have been developed to address this task, often tailored to the nature of the input data. Some approaches focus on dense airborne LiDAR data characterized by high-level point clouds and employ eigen-feature analysis of weighted covariance matrices along with an SVM classifier. Others utilize methods based on manually-labeled object locations. However, it's worth noting that in contemporary times, the majority of LiDAR object detection algorithms rely on CNNs (such as VoxNet) or SVMs [15].

4.2.1.3 Sensors processing and fusion

As mentioned before, there are a lot of ways to perform a sensor fusion task among multiple sensors. This section focusing on the input/output data which pass through each step of the processing part. The output signals from both the camera and LiDAR can be routed directly to the signal processing unit, where they undergo distinct processing. The LiDAR signal processing produces object locations represented by distance in meters, azimuth angles in degrees, and a discriminant value, crucial for distinguishing objects from non-objects. In contrast, the camera's output includes detection information in terms of relative bounding box positions within the image, along with confidence scores for the detections and their classifications.

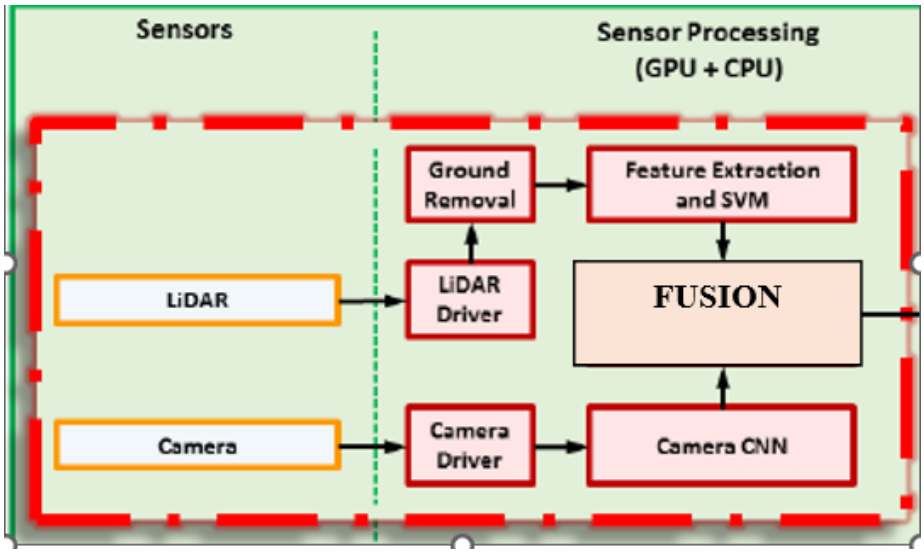


Figure 20 - Camera&LiDAR fusion: pre-Fusion data processing steps

As visible in figure 20 the sensors output pass throughout their own driver in order to generate manageable data for the specific object detection algorithms workflows which differs based on the sensor type.

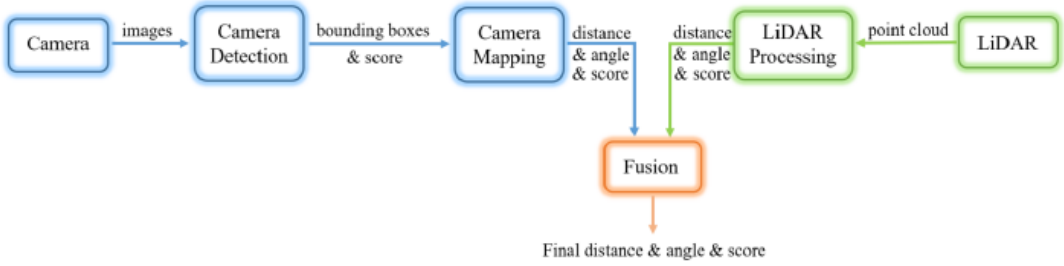


Figure 21 - Input/Output data for each pre-Fusion processing steps

Figure 21 is a fundamental one because allows to understand each passage of the data transformation during the entire execution. Finally, the fusion algorithm gives final distance, angle and score of the global detection of a specific object. The real-time implementation could be very obstacle but devices like NVIDIA Jetson family are produced for this particular goal, improving performance and time.

To summarize the entire workflow:

The mentioned fusion depends on two different branches for both the selected sensors: LiDAR and Camera. Each branch has the same output, then merged through the fusion session. The data which are taken as input of the chosen fusion algorithm are detection distance, detection angle and detection score. These values come from object detection processing step of data obtained from each kind of sensor.

Now, focusing on the LiDAR branch, figure 21 shows that the point cloud coming from the sensor passes through a processing step which is based on several sub-steps that can be summarized as follows:

- Intensity-based and density-based clustering (DBSCAN algorithm) on point cloud;
- Each point is examinate near the cluster centroid by extracting feature after a ground removal
- Linear SVM is used to classify objects and non-objects
    - More complex classificatory can be used to do a better classification in a multi class way. But, in general, SVM is sufficient thanks to the help of the multi-class classification which coming from the Camera image object detection.

In the other hand, the branch related to the camera presents two main steps. The first one takes as input images which comes from the sensor and perform an object detection, this step is deeply analyzed in the next chapter where will be also implemented. The second step starting from the results of the detection performs a mapping which consents to generate the same output values of the LiDAR branch in order to allow the following detection data fusion session. It is important to re-emphasize how the mapping process is fundamental, as mentioned before talking about the transformation matrices at the end of the 4.1 paragraph, for having an effective fusion which requires a common coordinate space among sensors.

About the final fusion step, it is possible to implement a small workflow as shown in the blue steps of figure 16. Here, is presented another possibility coming from the knowledge-based methods family in figure 14: the Fuzzy Logic fusion method [15].

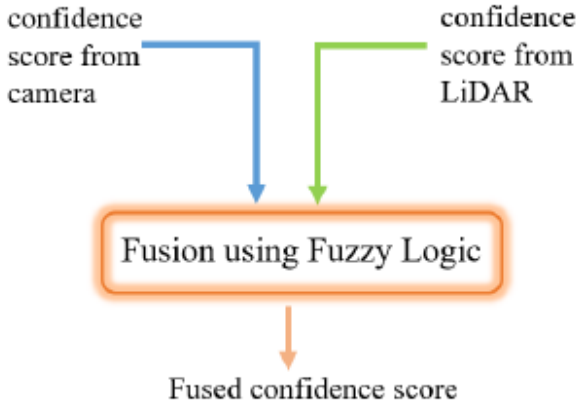


Figure 22 - Fusion of confidence score

In the fusion of confidence scores, fuzzy logic is employed to generate a conclusive confidence score in situations where both camera and LiDAR detections are present. The fuzzy logic system applies the following rules [15]:

I. If the LiDAR confidence score is high, then the probability for detection is high.
II. If the camera confidence score is high, then the probability for detection is high.
III. If the LiDAR confidence score is medium, then the probability for detection is medium.
IV. If the LiDAR confidence score is low and the camera confidence score is medium, then the
V. probability for detection is medium.
VI. If the LiDAR confidence score is low and the camera confidence score is low, then the probability
VII. for detection is low.

Ultimately, it is feasible to introduce a hyperparameter within the range [0,1] for comparison with the combined confidence score [0,1] explained earlier. This hyperparameter could serve as a universal threshold for the desired confidence score in line with the application's objectives.

5 YOLO: Image Object Detection

Due to delays in the project for the construction of the autonomous surface vehicle, it was not possible to implement a sensor fusion algorithm based on the previous chapter's discussion and using the sensors selected in Chapter 3. This was impossible due to the inability to work directly with the selected sensors as they were not yet in possession. Additionally, it was not possible to find an open-source database created from data collected through LiDAR and Camera in marine environments. Furthermore, considering the importance of calibration, it is necessary to conduct a data collection campaign in a defined environment after establishing the relative position between the sensors to be fused, ensuring an identical coordinate system. In this regard, my goal was to develop an algorithm that can process the steps highlighted in red in Figure 22.

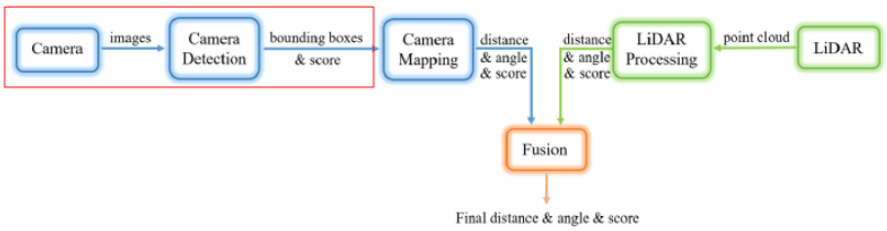


Figure 23 - Elaborated steps in red

The chosen programming language is MATLAB.

5.1 Hint of Computer Vision

As said in 4.2, in this chapter, before going into details of the YOLOv4 algorithm elaborated for the Venice canals, Computer Vision and Deep Learning CNN will be described with more effort.

Computer Vision involves the interpretation of image content, encompassing various tasks. This field is dedicated to tasks like categorizing entire images, as seen in systems classifying internet-uploaded photos. Additionally, it deals with object recognition within images, ranging from identifying faces to detecting car license plates. Moreover,

Computer Vision extends to detecting specific aspects within an image, exemplified by applications like cancer detection in biomedical images.

Founded in the 1970s as a sub-discipline of Artificial Intelligence, Computer Vision aimed to replicate the perceptual capabilities inherent in the human visual system, encompassing both the eyes and the majority of the brain. The initial aspiration was to develop a system that effortlessly interprets scenes, distinguishing between numerous categories with ease and swiftly identifying objects within a fraction of a second. The human visual system exhibits a remarkable ability to adapt and switch between various recognition processes, displaying a complexity and dynamic nature that remains not entirely comprehended. However, this ambitious goal proved challenging to achieve.

Despite nearly 50 years of evolution, Computer Vision remains on the frontier of technological advancements. Its success in contemporary times is attributed less to groundbreaking algorithms and more to the substantial improvement in computer speed and memory. Notably, the last decade has witnessed a resurgence in Computer Vision, driven by the integration of Deep Learning algorithms, enabling the proficient classification of extensive image collections [16].

Moreover, there are many fields related to computer vision with Image processing which is one of the most connected. Image Processing involves altering or manipulating an image with the objective of highlighting specific aspects, such as contrast enhancement, or extracting low-level features like edges and blobs. In contrast, Computer Vision focuses more on the extraction of higher-level features and their interpretation for recognition purposes [16].

5.2 Image Dataset: Google Earth Pro

To train a system for a particular classification task, the initial hurdle involves gathering suitable learning materials. This phase of collection is a labor-intensive process and can often prove to be more challenging than anticipated.

The ultimate goal of the entire project, as mentioned, is to build an ASV capable of navigating the canals of Venice. For this reason, I have thought of developing an algorithm that can recognize and classify various types of boats, especially those most commonly found in the canals of Venice.

To do this, it is essential to collect images from a marine environment. To make the results even more relevant to the ultimate goal of the project, images were obtained from the canals of Venice using Google Earth Pro. This was done to implement an algorithm that, while already yielding excellent results, is strongly oriented towards improvement after a future acquisition of a greater number of images through the D455 camera when it will be possible to carry out an experimental acquisition campaign on-site.

Figure 24 – Images of the Venice Lagoon dataset examples

Google Earth Pro allows the user to save images in different resolution, so I decided to set the resolution of the D455 in order to implement the same image processing needed by the images which will be collected by the D455.

Analyzing the datasets images, it is decided to try to recognize the boats dividing them in three different classes: waterbus, motorboat and gondola. Subsequently, the gathered images undergo processing using the MATLAB ImageLabeler App, enabling the manual labeling of training images to identify objects belonging to distinct classes.

It is important to understand that one should thereby pay attention to several possible issues that may be experienced by the dataset. In particular [16]:

- Scene Background: When images belonging to a particular object class consistently feature the same background scene, there is a risk that the system might end up learning the background more than the object itself.
- Forced Detection: When in each training image there is almost one object. It could comport that the detector learns to detect something in each test image although there is no object detectable.
- Class Balance (inter-class): It is advisable to maintain a balanced count of images for each class, ensuring that each class has approximately an equal number of images. Otherwise, an issue known as class imbalance may arise. A straightforward approach to address this challenge involves implementing oversampling and under sampling techniques. Oversampling involves artificially generating new images for the class with fewer instances, often through augmentation methods. On the other hand, under sampling entails reducing the number of images in the class with a surplus of instances.
  In Venice Dataset there is a significant unbalance as shown in figure 25.

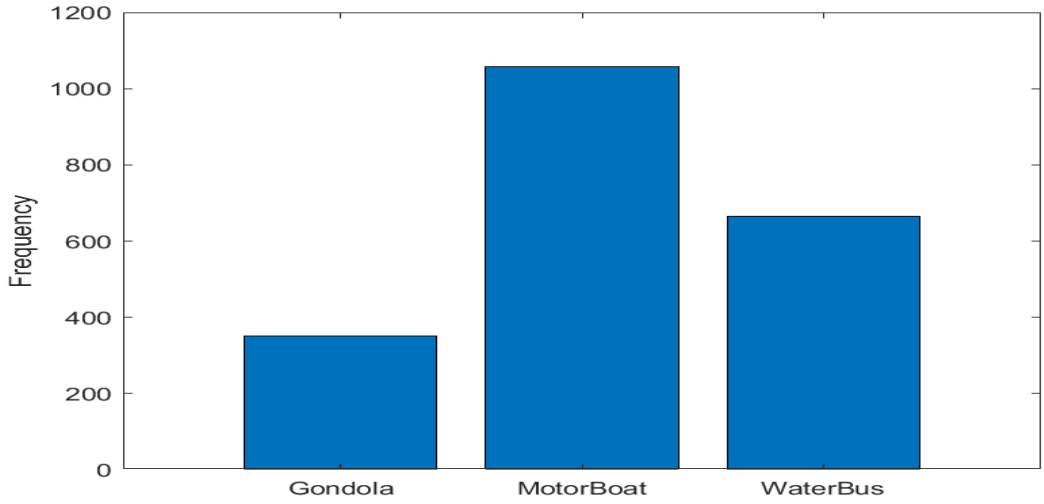


Figure 25 - Balance of the Google Earth Pro Venice Dataset

To Overcome this problem are developed multiple data augmentation techniques.

- Objects Sizes Overlap (inter-class): for a good detector performance inter-class overlap is essential. Figure 26 shows that this dataset is great from this point of view.

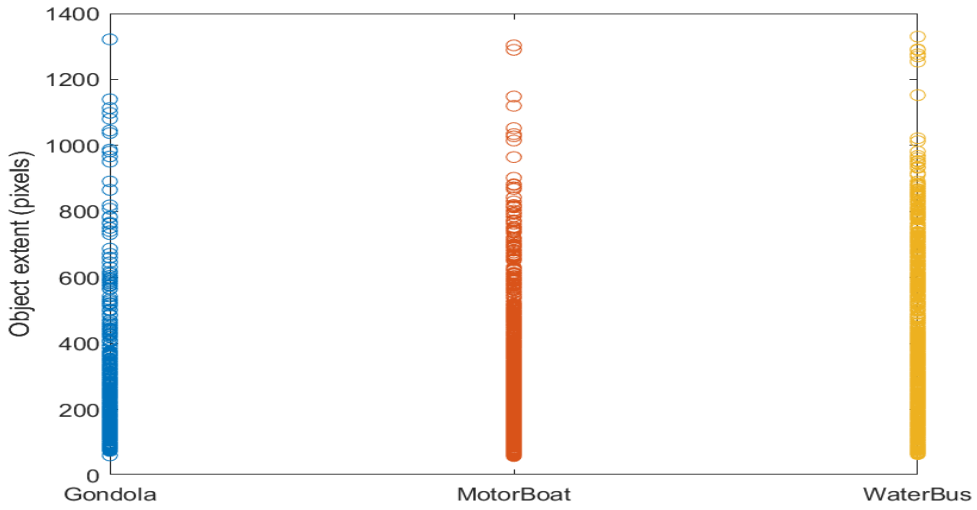


Figure 26 - Labelled Object dimensions

- Intra-Class dimension variance: The choice of the better version of YOLO detector is strictly related to this issue. Figure 26 illustrates a huge variance in term of dimensions in each class. This issue means that the detector must

detect in different scales, so a multi-scale object detector is preferable to a single-scale detector.

5.2.1 Intra-Class dimension variance: Problem resolution

As just said the significative dimension variance in each class could create several performance problems. To overcome this issue the selection of the right version of YOLO as detector is essential. At first, I tried to perform the detection through YOLOv2. This detector results right class detection but experienced problems in defining the correct bounding boxes in the right position. In other words, an example image presents the correct assignations, if the picture shows a motorboat, then the detector correctly classifies it, but the problem is that the bounding boxes related to the corresponding motorboat could be smaller or larger than the right one and at the same time could be in a wrong position.

The problem comes from the fact that YOLOv2 is a mono-scale detector which means that, based on the architecture developed in phase of coding, the detector can be optimal only in a specific scale of dimension. For this reason, the final version of YOLO selected is the YOLOv4 which, being a multi-scale detector, performs with optimal detection in different scales.

5.2.2 Class imbalance: Problem Resolution

Oversampling and undersampling are two distinct approaches to address the issue of imbalance between classes in a dataset. In oversampling, the focus is on the minority class, aiming to generate artificial data to increase the number of examples and bring it closer to that of the majority class. This often involves the use of augmentation techniques, such as duplicating existing samples or applying transformations. However, oversampling carries the risk of overfitting, as the model may memorize the minority class examples.

On the other hand, in undersampling, the goal is to reduce the number of examples from the majority class by eliminating some of them. This simplifies the dataset,

reducing computational complexity and training time. However, there is a risk of losing important information present in the majority class, potentially negatively impacting model performance. The choice between oversampling and undersampling depends on the specific context of the problem and the available data, often requiring experimentation to assess which technique works better to improve model performance.

The decision made with the existing dataset involves the utilization of Data Augmentation techniques. Opting for an undersampling solution could lead to substantial loss of information, especially given that the dataset obtained through Google Earth Pro is not extensive.

In MATLAB it is used the augmentData() function through transform().

```
function data = augmentData(A)
data = cell(size(A));
for ii = 1:size(A,1)
    I = A{ii,1};
    bboxes = A{ii,2};
    labels = A{ii,3};
    sz = size(I);

    if numel(sz) == 3 && sz(3) == 3
        I = jitterColorHSV(I,...
            contrast=0.0,...
            Hue=0.1,...
            Saturation=0.2,...
            Brightness=0.2);
    end

    tform = randomAffine2d(XReflection=true,Rotation=[-30 30],Scale=[1 1.5]);
    rout = affineOutputView(sz,tform,BoundsStyle="centerOutput");
    I = imwarp(I,tform,OutputView=rout);

    [bboxes,indices] = bboxwarp(bboxes,tform,rout,OverlapThreshold=0.25);
    labels = labels(indices);

    if isempty(indices)
        data(ii,:) = A(ii,:);
    else
        data(ii,:) = {I,bboxes,labels};
    end
end
end
```

Figure 27 - augmentData() function

The supporting function `augmentData` performs data augmentation for a set of images and their corresponding bounding box labels, which have been generated through the processing of images in the MATLAB ImageLabeler App. The applied transformations include random horizontal flipping, random scaling along the X and Y axes, as well as random color modification of the images. Additionally, the function takes into account the presence of bounding boxes in the images, adapting the transformations accordingly. Specific transformations involve the option to introduce noise to the color of the images by adjusting parameters such as contrast, hue, saturation, and brightness. The function returns a new set of data with the modified images and their respective bounding box labels. Figure 28 shows four different examples of augmentation of the same image.

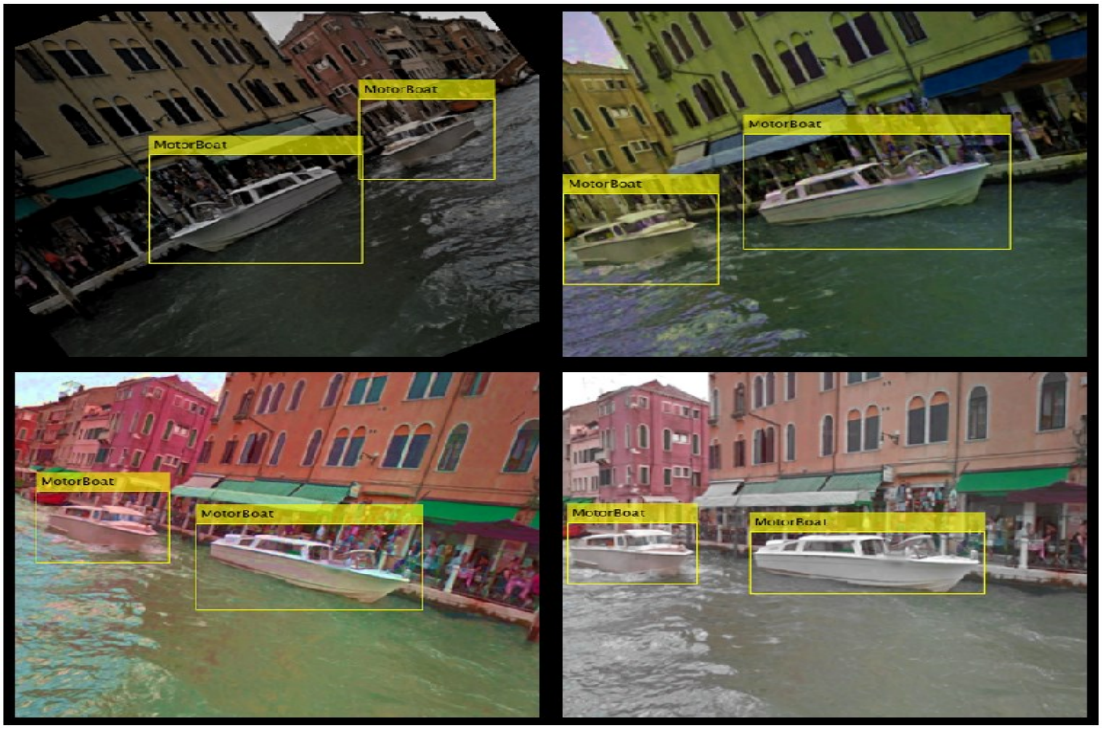


Figure 28 - Augmentation images example

5.3 CNN: Convolutional Neural Networks

Starting from what it is anticipated in 4.2, this paragraph aims to recap several fundamental hints about CNN.

Convolutional Neural Networks (CNN), represent a category of deep learning models specifically designed for processing structured data in grid-like formats, such as images. Unlike traditional neural networks, CNNs utilize convolutional layers to detect spatial patterns in images through the application of convolutional filters. These filters allow for the capture of hierarchical features, recognizing low-level details like edges and textures, up to high-level features such as shapes and complex objects. CNNs are also characterized by the presence of pooling layers, which progressively reduce the spatial dimensions of the data while retaining salient information. Thanks to these features, CNNs are widely used in computer vision applications, pattern recognition, and image classification, proving particularly effective in capturing the spatial structure and contextual relationships present in images.

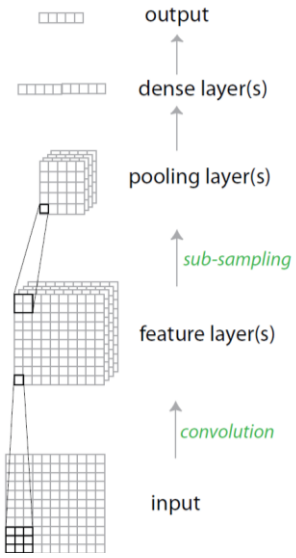


Figure 29 – Typical Simplified CNN architecture

As previously mentioned, CNNs are distinguished by the sequential arrangement of multiple layers, each serving a distinct purpose. Figure 30 depicts numerous possible combinations found in CNN networks, with the most common one highlighted in red. Subsequently, a description of each layer's task is provided in order to better understand the entire CNN used for this project.

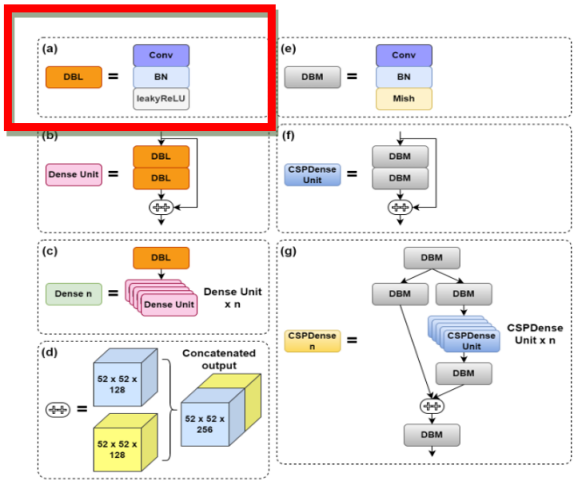


Figure 30 - Possible CNN block layers

- Convolutional Layer:

- Input: images in the form of intensity values matrix, generated through the imread() function of MATLAB.
- Filter (Kernel): local receptive field. It corresponds to a sub-area of the image

  The kernel passes through the image where each pixel will have an intensity weight, so a kernel 3x3 will have 9 weights plus one bias for a total of 10 parameters.

  For RGB images, like the ones of the Venice Dataset, the kernel will have 28 weights because they are in 3D [3x(3x3) + 1]. These parameters will be used for generates a hidden neuron which will be part of the feature map in the next stadium.

  Finally, each feature map coming from these layers is composed by hidden neuron with a value resulting from:

$$y = \sum\sum w_{ij}x_{ij}$$

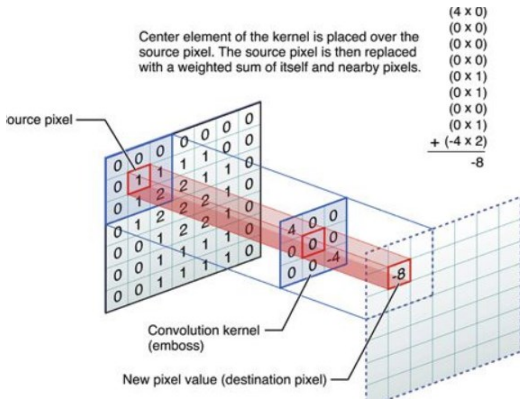


Figure 31 - Convolutional Layers and feature maps extraction

- Output: A number of Feature Maps corresponding to how many Kernels are used. Each feature map highlights the presence of specific attributes or patterns in the region of the image to which the filter has been applied. Each filter is designed to detect particular features, such as edges, textures, or shapes, contributing to the creation of diverse feature maps. The

information contained in the feature maps becomes increasingly complex and abstract through the successive layers of the neural network, enabling the CNN to learn more sophisticated representations of the visual input during the training process.

- Padding: a 2-D convolutional layer allows for the use of "padding," which involves adding zero values around the input image or feature map to control the size of the output. Padding can be "valid" (no padding) or "same" (padding to keep the dimensions the same).
- Stride: The layer can use a stride to specify the step with which the filter moves through the image or feature map. A stride of 1 indicates one-pixel shifts at a time, while a larger stride increases the spacing between filter positions.

- Batch Normalization Layer:
  - Normalization of Feature Maps:

    After the "conv2D" layer has performed convolution on the input feature map, the "batch normalization" layer is responsible for normalizing the feature maps. Normalization involves transforming the feature maps so that they have a mean close to zero and a standard deviation close to one. This helps stabilize the learning process in the network.

  - Distribution Adjustment:

    Batch normalization adjusts the distribution of values in the feature maps to be more balanced and not excessively large or small. This helps prevent issues related to vanishing gradients and exploding gradients during the training process.

  - Efficient Learning:

    Batch normalization helps the neural network learn more efficiently, reducing the number of epochs required to achieve good performance. This is particularly useful in deep neural networks.

- Reduction of Overfitting:
  Batch normalization can also help to reduce overfitting in neural network models.

In essence, the "batch normalization" layer after a "conv2D" layer is a fundamental part of a CNN's architecture. It normalizes and adjusts the feature maps, contributing to improved stability and effectiveness in the network's training. Its presence is common in many convolutional neural networks to ensure that the learning process is smooth and that the model converges to optimal solutions.

- Leaky ReLu Layer

  A Leaky ReLU layer, positioned after the previously mentioned convolutional and batch normalization layers, is a crucial element in a neural network, introducing non-linearity to the feature maps derived from convolution and normalization.

  The role of a Leaky ReLU layer can be summarized as follows:

  After the batch normalization layer has normalized the feature maps, the Leaky ReLU layer introduces non-linearity into the network. Unlike the traditional ReLU (Rectified Linear Unit), Leaky ReLU allows a small negative slope for negative values, maintaining positive values unchanged. Specifically, for $x \geq 0$, $f(x) = x$, and for $x < 0$, $f(x) = \text{alpha} * x$, where alpha is a small constant typically in the range of 0.01-0.3.

  This introduction of non-linearity through Leaky ReLU enables the model to capture complex representations and non-linear patterns in the data. This proves particularly advantageous in computer vision tasks where the relationship between image pixels and desired features can exhibit high non-linearity.

  Leaky ReLU helps prevent the problem known as "dying ReLU," where neurons in a layer become inactive and contribute minimally to the learning

process. Introducing the negative slope allows neurons to remain active even when they receive negative inputs.

In essence, the Leaky ReLU layer injects non-linearity into the feature maps post-normalization through batch normalization. This facilitates the network in learning intricate data relationships and addresses concerns related to "dying ReLU." It is a common choice as an activation function in various neural networks, including convolutional neural networks utilized in image processing.

5.4 Transfer Learning

Transfer Learning is a strategy in machine learning where a model previously trained on a task is used as a starting point for a new, related task. This approach differs significantly from learning from scratch, where a model is trained entirely with random parameters without any prior knowledge. In learning from scratch, the model begins training with random parameters and learns all the necessary representations and features directly from the new dataset. In Transfer Learning, the model benefits from pre-existing knowledge acquired from a previous task. This can occur through weight sharing in some layers of the pre-trained model or by using the pre-trained model as a feature extractor. Transfer Learning is particularly advantageous in situations where the new task has a limited dataset. By utilizing a pre-trained model, already equipped with useful and general representations learned from a larger dataset, a significant performance improvement can be achieved without the need for extensive training data.

This is especially valuable in contexts where obtaining large datasets is expensive or impractical.

An example is when working with small datasets, like in the Venice Dataset created through Google Earth Pro. Instead of training a model from scratch, one can leverage a pre-trained model on a related task and fine-tune it for the new context. This reduces

the risk of overfitting, especially when dataset sizes are limited. In summary, Transfer Learning is an effective strategy to enhance model performance, allowing models to capitalize on knowledge gained in previous situations. It is particularly advantageous when dealing with limited datasets.

The transfer learning process can occur in various modes:

- Feature Extraction: In this approach, a pre-trained model is used as a feature extractor. The model is truncated after one or more layers, and only these layers are retained while the rest of the model is replaced. The retained layers act as feature extractors and are connected to one or more additional layers, in the case of YOLO network the sequence of this layers is called head, that are trained for the new task which could be the recognition and localization of an object from the feature maps created from the first pre-trained part of the chain.
- Fine-Tuning: In this case, we start with a pre-trained model, and some of its final layers or all layers are trained to adapt to the new task. This additional training phase allows the model to specifically adjust to the features of the new task.

5.4.1 YOLOv2 chain example

As mentioned before, as first try YOLOv2 was selected to perform the requested boat recognition, then it was set aside. Now, its structure could be used to show practically an example of transfer learning from a pre-trained network.

YOLOv2 is based on a pre-trained, on COCO dataset, darknet19 network. In figure 32 there is the chain of the darknet19 network which shows the alternating layers mentioned in 5.3. This figure is the results of the analyzeNetwork() function of MATLAB which allows to illustrates CNN in details.

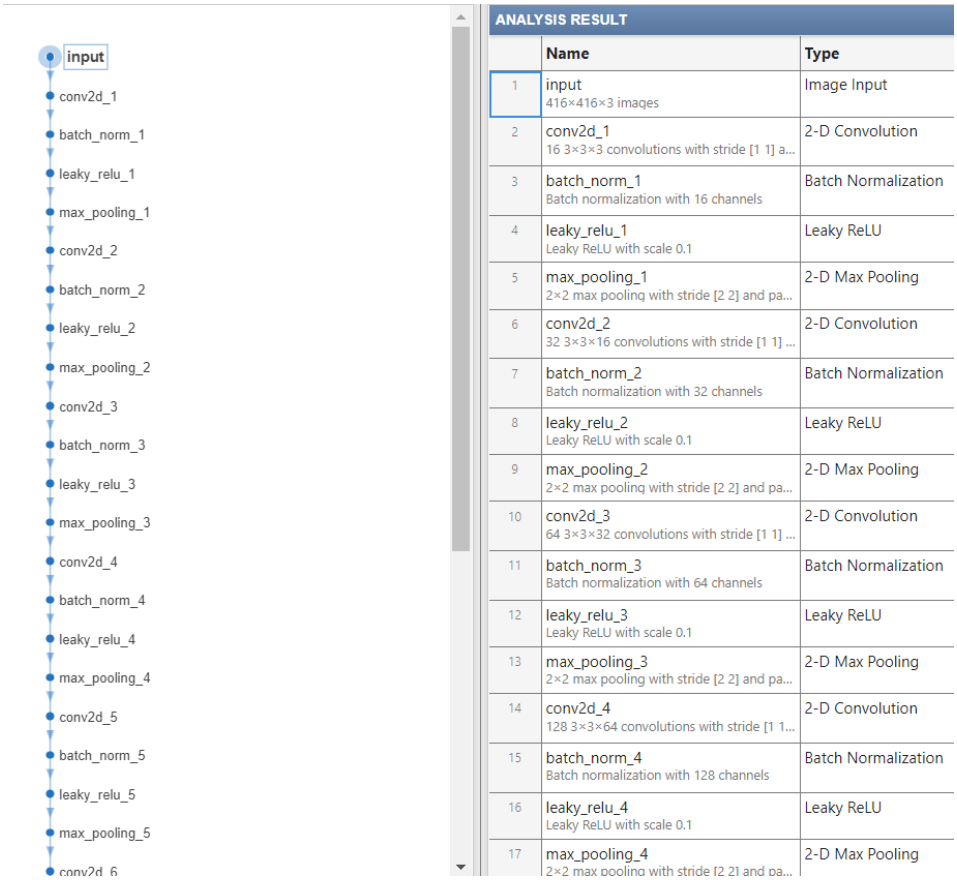


Figure 32 - Darknet19 pre-trained network

Now, to create a possible final chain it is necessary to perform the transfer learning process in one of the two possible solutions previously explained. As example, in YOLOv2 attempt the leaky_relu_5 is set as feature extractor layer, in other words the last layer which has the task of feature map extraction, and the, with the help of the yolov2Layers() function the detection layers are added as shown in figure 33.

Figure 33 - Tiny YOLOv2 chain with feature extraction on Leaky_Relu_5

Finally, it is more understandable why YOLOv2 is a mono-scale detector. The detection depends on the selected feature extractor layer, it means that choosing Leaky_relu_5 is different from choosing Leaky_relu_7 because they performances are related to different scale. Choosing later layers means allow the action of more convolutional layers, BN layers and leaky_relu layers resulting in better performance in detecting big object with respect to small ones.

5.5 YOLO overview

YOLO, an acronym for "You Only Look Once," stands out as a prominent object detection algorithm that has brought about a paradigm shift in the realm of computer vision and object detection. Its revolutionary methodology has propelled YOLO to become a widely adopted tool for visual perception in diverse applications like autonomous vehicles, video surveillance, and image analysis. YOLO deviates from

conventional object detectors that employ a region-of-interest-based strategy, necessitating multiple stages for detection. Instead, YOLO embraces a "one-stage detection" approach, introducing several key characteristics compared to traditional methods.

In the classic object detector, a two-stage detection process is employed. It initiates with region-of-interest (ROI) detection to identify potential areas containing objects, followed by a subsequent step involving the classification and refinement of these regions to identify the objects. YOLO, in contrast, executes the entire object detection and classification process in a single pass, eliminating the need for generating ROIs and substantially enhancing speed.

Considering the number of passes, the classic object detector necessitates two separate passes: one for generating ROIs and another for object classification within those regions. YOLO, on the other hand, accomplishes the entire process in a single pass, enabling real-time object detection and classification.

Regarding the consideration of global context, the classic object detector typically focuses on the local context of ROIs, analyzing each region independently. YOLO, however, takes into account the global context of the entire image during the detection process, enhancing the understanding of relationships between objects and their surrounding environment.

In terms of speed and efficiency, the classic object detector is often slower due to the two distinct passes and ROI computation. In contrast, YOLO, with its one-stage architecture and streamlined approach, is renowned for its speed and adaptability in real-time detection applications.

Applications of classic object detectors span various domains but may be less suitable for applications requiring rapid and real-time detection. YOLO, given its speed and global context consideration, finds extensive use in real-time applications such as autonomous driving, security, video surveillance, and more [17].

In summary, YOLO represents a significant breakthrough in object detection, characterized by its one-stage regression approach, speed, and efficiency, making it a preferred choice for real-time applications.

5.6 YOLOv4 Architecture

In 5.4.1 there is a hint to the construction of the architecture of a YOLOv2. The fourth version differs slightly due to the inclusion of a middle chain separator (neck) positioned between the feature extraction part (backbone) and the detection part (heads). Additionally, a notable distinction lies in the fact that version 4 incorporates three heads, each functioning at different scales, facilitating optimal detection of objects of varying sizes.

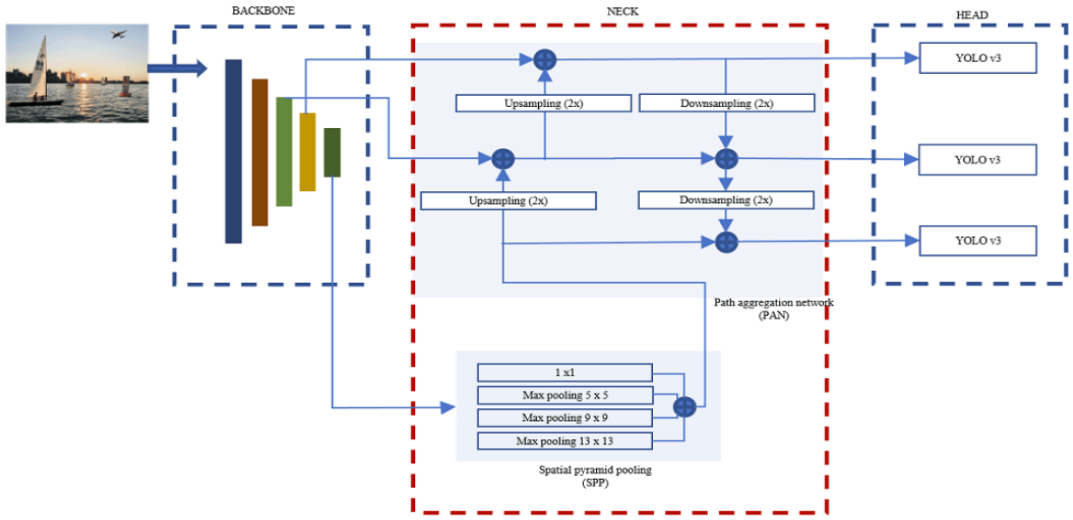


Figure 34 - YOLOv4 Architecture

The backbone serves as the foundation, comprising a sequence of convolutional layers designed to efficiently capture characteristics at various scales. The chosen network for the backbone is CSPdarknet53, where CSP denotes 'cross space partial.' This network enhances communication among feature maps from different layers, and it often employs pretrained weights, crucial for effective transfer learning from a larger dataset.

The neck, a subset of the bag of specials [17], acts as a feature aggregator by collecting feature maps from different stages of the backbone. Essentially, it plays a role in consolidating features.

The SPP (Spatial Pyramid Pooling) serves as an additional block, processing the feature maps generated by the backbone to capture diverse spatial information at different scales. This architecture incorporates multiple pooling grids with varying dimensions and scales, each corresponding to a distinct region of the feature maps. By covering regions of different dimensions, the model can consider details at various scales. In general, SPP reduces the dimensions of feature maps while preserving critical details. The PAN (Path Aggregation Block) introduces a progressive attention mechanism. It divides the attention of the block across the feature maps, becoming more pronounced in specific regions.

The head encompasses layers responsible for detection and classification. It takes as input feature maps at different scales with spatial information, employing three different feature maps for three heads (two tiny). The output consists of a series of grids covering the input image, where each cell is tasked with detecting objects in its region and producing predictions for the (x, y) bounding box, (w, h) bounding box, score, and class probability. The process concludes with non-maximum suppression.

5.6.1 Transfer Learning: pre-trained network for the backbone

YOLOv4 utilizes as pre-trained network CSPDarknet53 shown in figure 35 (a). For computational reason the selected YOLOv4 architecture is the tiny one because it is less expensive on a laptop and because it is perfect for elaborates a small dataset of images with the goal of classify only three different classes. The CSPDarknet53-tiny is used as illustrate in figure 35 (b) [18].

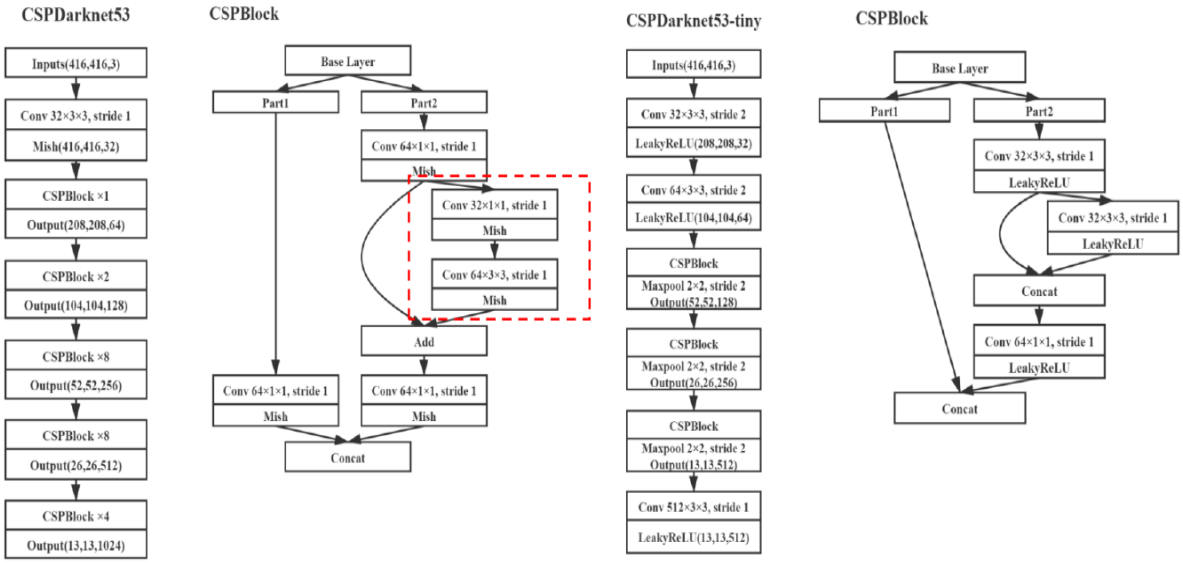


Figure 35 - CSPdarknet53 (a) - CSPdarknet53-tiny (b)

With analyzerNetwork() function it is possible to compare the network structure coming from the literature with the one used in the project which is shown in figure 36.

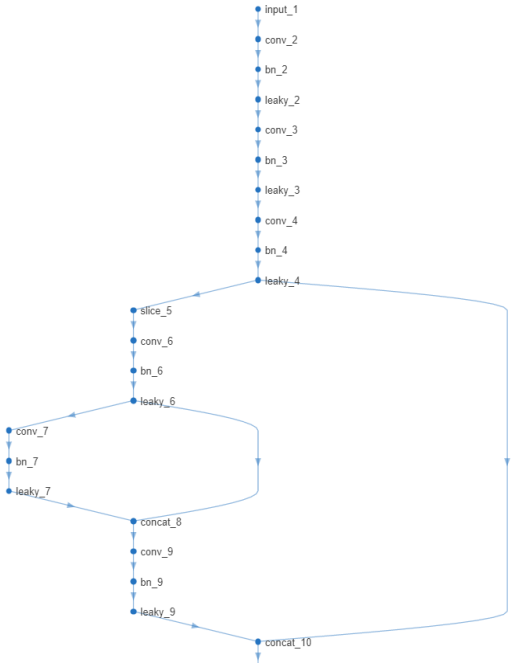


Figure 36 - Small part of the YOLOv4 architecture used in the project

It's crucial to note that CSPDarknet59 expects images with an input dimension of 416x416x3. Therefore, data processing techniques are employed to resize the image dimension from 1280x720x3 to meet the specified requirements. To accomplish this task, the preprocessData() helper function of MATLAB, as depicted in figure 37, is utilized.

```
function data = preprocessData(data, targetSize)
% Resize the images and scale the pixels to between 0 and 1. Also scale the
% corresponding bounding boxes.

% Copyright 2021 The MathWorks, Inc.

for ii = 1:size(data,1)
    I = data{ii,1};
    imgSize = size(I);

    % Convert an input image with single channel to 3 channels.
    if numel(imgSize) < 3
        I = repmat(I,1,1,3);
    end
    bboxes = data{ii,2};

    I = im2single(imresize(I,targetSize(1:2)));
    scale = targetSize(1:2)./imgSize(1:2);
    bboxes = bboxresize(bboxes,scale);

    data(ii, 1:2) = {I, bboxes};
end
end
```

Figure 37 - preprocessData() function to re-set the image dimension

5.7 Experimental Setup

5.7.1 Anchor boxes

Anchor boxes, or anchor boxes, are a crucial element in the YOLO algorithm for object recognition in images. These anchor boxes are essentially predefined bounding boxes with specific sizes and proportions.

The main role of anchor boxes in YOLO is to facilitate the prediction and association of bounding boxes with object classes. Since YOLO predicts bounding boxes directly during training, without the need to generate regions of interest (ROIs) as in other

approaches, anchor boxes provide a way to handle the variety of sizes and proportions of objects in the image. The training process of YOLO involves assigning each bounding box in an image to a specific anchor box, and this association is based on the similarity of sizes and proportions. During the inference phase, the predictions of bounding boxes are then transformed and associated with the corresponding anchor boxes to obtain the final positions of the objects. Anchor boxes allow YOLO to effectively capture objects of different sizes and shapes, contributing to making the algorithm robust and capable of handling a variety of scenarios. The use of anchor boxes also helps simplify the training process and improve the accuracy of predictions, especially when there are objects of various sizes within the images.

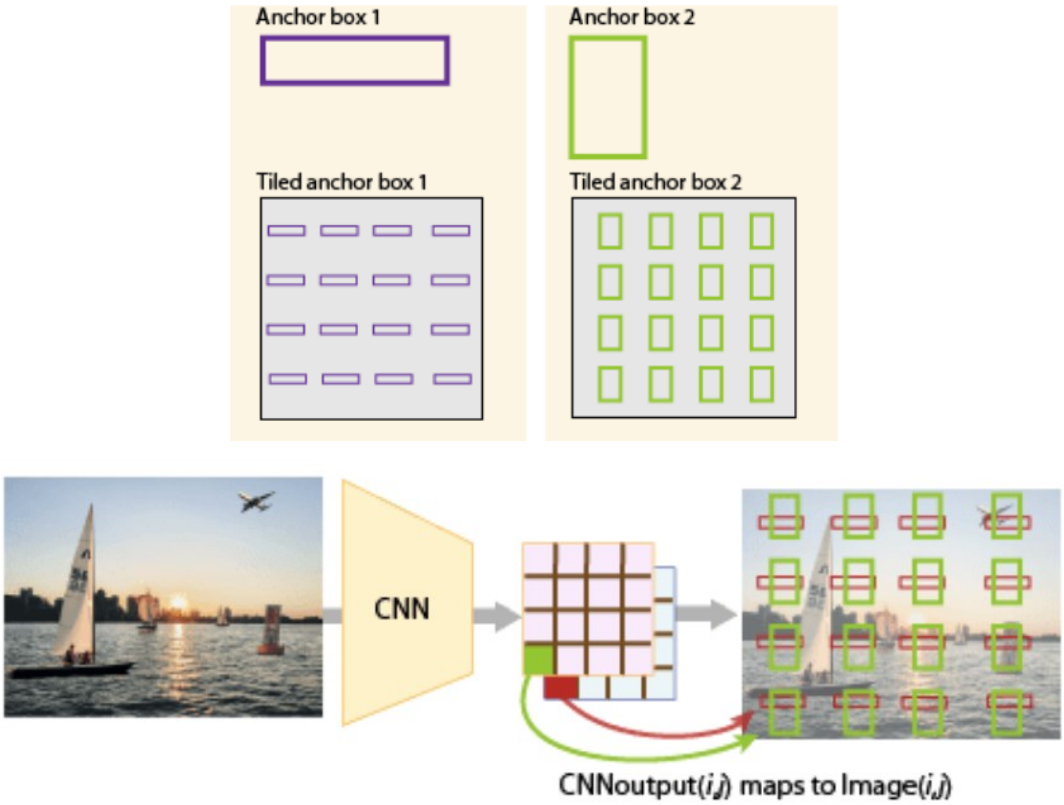


Figure 38 - (up) two anchor boxes tiled in a specific region - (down) same case but shown with an image example

In YOLOv2 the number of anchor boxes was determinate experimentally taking into account a trade-off between the number of anchors and the mean IoU metric. This metric represents the average Intersection over Union (IoU) calculated for a set of predictions. IoU reflects how much the predicted bounding boxes overlap with the actual ones, with a high value indicating accurate precision in object localization. Measuring "mean IoU" provides an overall assessment of the quality of predictions made by the YOLO object detection model.

Generally, a high number of anchors can result in increased computational costs and, in certain situations, an abundance of false positives. On the contrary, achieving the highest possible mean IoU is a desirable objective. Figure 39 provides insights into finding a tradeoff to be utilized in the case of YOLOv2.

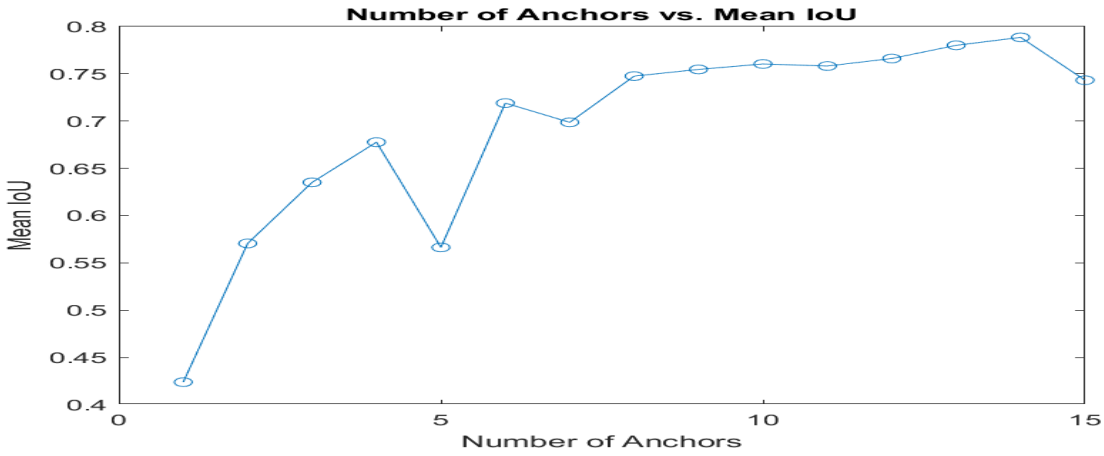


Figure 39 - YOLOv2 empirical research of needed number of anchor boxes

YOLOv4 introduces an enhancement in the determination of the required number of anchors. It necessitates 9 anchors, or just 6 anchors in the case of its tiny version. As previously mentioned, the tiny version features only two heads compared to the full version 4, which has 3 heads. Regardless, the total number of anchors is divided into groups of three for each head, depending on the number of heads present. The estimation of anchor boxes can be conveniently carried out using the estimateAnchorBoxes() function implemented in MATLAB. This function performs an estimation based on the available dataset.

The previously highlighted enhancement in YOLOv4 for recognizing objects at different scales relies significantly on the arrangement of anchor boxes. The specification of anchorBoxMasks is crucial to determine the anchor boxes to be utilized in both detection heads. AnchorBoxMasks is a cell array of [Mx1], where M represents the number of detection heads. Each detection head comprises a [1xN] array of row indices of anchors in anchorBoxes, where N is the number of anchor boxes to be used. The selection of anchor boxes for each detection head is based on size, prioritizing larger anchor boxes at lower scales and smaller anchor boxes at higher scales.

5.7.2 Transfer Learning Feature Extraction technique applied

In 5.4 are described the transfer learning techniques and, in particular, in 5.4.1 is described the feature extraction one adapted to the YOLOv2 case. Here, the same approach is applied to YOLOv4. In case of YOLOv4-tiny two instances convolution2dlayer are created. For each one of them the number of output channels is calculated as product of the length of Masks and the number of predictors per anchor box. The number of predictions per anchor box is set to 5 ([x,y,h,w] score) plus the number of object classes. "5" denoted the 4 bounding box attributes and 1 object confidence.

```
if isrow(classNames)
    classNames = classNames';
end
numClasses = size(classNames, 1);
numPredictorsPerAnchor = 5 + numClasses;
```

Figure 40 - Setting of numPredictorsPerAnchor

Finally, conv_31 and conv_38 are replaced by the new final output layers.

```
lgraph = layerGraph(net);

if strcmp(modelName, 'YOLOv4-coco')
    yoloModule1 = convolution2dLayer(1,length(anchorBoxMasks{1})*numPredictorsPerAnchor,'Name','yoloconv1');
    yoloModule2 = convolution2dLayer(1,length(anchorBoxMasks{2})*numPredictorsPerAnchor,'Name','yoloconv2');
    yoloModule3 = convolution2dLayer(1,length(anchorBoxMasks{3})*numPredictorsPerAnchor,'Name','yoloconv3');

    lgraph = replaceLayer(lgraph,'conv_140',yoloModule1);
    lgraph = replaceLayer(lgraph,'conv_151',yoloModule2);
    lgraph = replaceLayer(lgraph,'conv_162',yoloModule3);

    networkOutputs = ["yoloconv1"
        "yoloconv2"
        "yoloconv3"
        ];
elseif strcmp(modelName, 'YOLOv4-tiny-coco')
    yoloModule1 = convolution2dLayer(1,length(anchorBoxMasks{1})*numPredictorsPerAnchor,'Name','yoloconv1');
    yoloModule2 = convolution2dLayer(1,length(anchorBoxMasks{2})*numPredictorsPerAnchor,'Name','yoloconv2');

    lgraph = replaceLayer(lgraph,'conv_31',yoloModule1);
    lgraph = replaceLayer(lgraph,'conv_38',yoloModule2);

    networkOutputs = ["yoloconv1"
        "yoloconv2"
        ];
end
end
```

Figure 41 - Layer Graph modification based on the number of the detectable classes

Similarly, it is worth highlighting that the disparity in the number of "yoloconv" layers between YOLOv4 and YOLOv4-tiny stems from differences in the architecture of the two models. YOLOv4, being a larger and more intricate model, is crafted to detect a broader array of objects across various classes. In contrast, YOLOv4-tiny is a streamlined variant optimized for constrained computational capabilities, providing quicker but less intricate detection. In the case of YOLOv4, three YOLO modules (typically named "yoloconv1," "yoloconv2," and "yoloconv3") are present, each tasked with detecting objects in distinct regions of the image at varying scales. These modules can be configured to identify objects of diverse sizes, hence necessitating a higher count of "yoloconv" layers. Conversely, YOLOv4-tiny is a simplified rendition of YOLOv4, designed to be more lightweight and faster. As a result, it features only two YOLO modules ("yoloconv1" and "yoloconv2"). The network is smaller and less intricate, rendering it suitable for devices with restricted computational capabilities or scenarios where detection speed holds paramount importance.

5.7.3 Training Options specifications

The training of an object detector model is empirical research. The most important parameters to tune during empirical research of the best detector in our environment are shown in the code in figure 42.

```
numEpochs = 90;
miniBatchSize = 6;
learningRate = 0.001;
warmupPeriod = 1000;
l2Regularization = 0.01;
penaltyThreshold = 0.5;
velocity = [];
```

Figure 42 - Training Options selected

numEpoches signifies a complete iteration through the entire training dataset, where each epoch involves training the model with each sample to adjust the network weights. The structure of each epoch unfolds through various steps. First is the forward pass, where each sample traverses the network or mini-batch, generating predictions from the model. Following this, there is the loss calculation step, where each sample produces a loss indicating the disparity between the prediction and the actual label. Subsequently, during backpropagation, the model computes the gradient of the loss concerning its weights, reflecting the model's error for that specific sample. Finally, in the weights update step, an optimization solver is employed to adjust the model's weights based on the calculated gradients, refining the model's performance.

In the context of MiniBatchSize, a batch refers to a cluster of images processed simultaneously during a training session. The utilization of batches offers several advantages. Firstly, it enhances stability, leading to quicker convergence of the model. Additionally, it contributes to better generalization of the model. Typical values for

MiniBatchSize include 16, 32, 64, and 128. When dealing with a small dataset, different considerations come into play. Higher values, such as 64 or 128, come with the advantage of expedited convergence, but they also pose the drawback of being computationally intensive, introducing more variability in the training process. On the contrary, smaller values, such as 16 or 32, provide the benefit of lower computational effort. However, they come with the disadvantage of potentially slower training and convergence. The choice of MiniBatchSize involves a trade-off between computational efficiency and training effectiveness, considering the specific characteristics of the dataset.

The technique known as L2 Regularization is employed to counteract overfitting and enhance the generalization capabilities of a model. During training, a new regularization term is introduced into the model's loss function, proportional to the sum of the squares of the weights. This regularization term aims to penalize neurons with excessively heavy weights, thereby preventing the weights from becoming too large throughout the training process. The L2 term is mathematically represented as $L2\ term = 0.5 * \gamma * \sum w_i^2$, where γ is a coefficient regulating the strength of the penalty. A higher value of γ contributes to effective regulation and reduces overfitting, but it may result in a model that is too simplistic, potentially overlooking complex data patterns. It is important to note that determining the optimal value for γ involves empirical research, striking a balance that avoids overfitting without excessively penalizing the model's complexity.

The InitialLearnRate parameter plays a pivotal role in the early stages of the training process, dictating the magnitude of the steps taken to update the model's weights during each iteration or batch of training. The selection of an appropriate initial learning rate holds significant importance, as it can markedly influence the overall training procedure. An excessively high learning rate may lead to non-convergence, characterized by oscillations or divergence of the loss function. Conversely, if the learning rate is too low, the training process becomes overly sluggish, potentially causing the model to become trapped in suboptimal local minima. Striking the right

balance in setting the InitialLearnRate is crucial to ensure a smooth and efficient training process, avoiding issues related to convergence, speed, and the model's ability to find optimal solutions.

Finally, warmup period helps in stabilizing the gradients at higher learning rates. Specifying the penalty threshold as 0.5. Detections that overlap less than 0.5 with the ground truth are penalized. Initializing the velocity of gradient as [] helps SGDM to store the velocity of gradients.

5.8 Evaluation Metrics and Results

Precision and recall are two important metrics used to evaluate the performance of classification and object detection models, including those used in YOLOv2 object detection. These metrics provide insights into how well a model is performing in terms of correctly identifying positive instances and minimizing false positives and false negatives.

Precision is a measure of the accuracy of positive predictions made by a model. It is the ratio of true positive predictions to the total number of positive predictions (true positives plus false positives). In mathematical terms, precision is defined as:

$$P = \frac{TP}{TP + FP}$$

Precision tells how many of the predicted positive instances are actually correct. A high precision indicates that when the model predicts an object, it is likely to be correct.

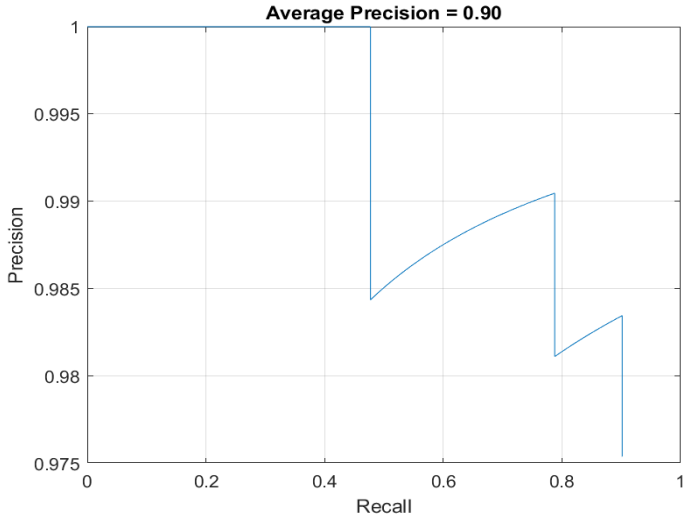
Recall is a measure of the model's ability to identify all relevant instances in the dataset. It is the ratio of true positive predictions to the total number of actual positive instances (true positives plus false negatives). Mathematically, recall is defined as:

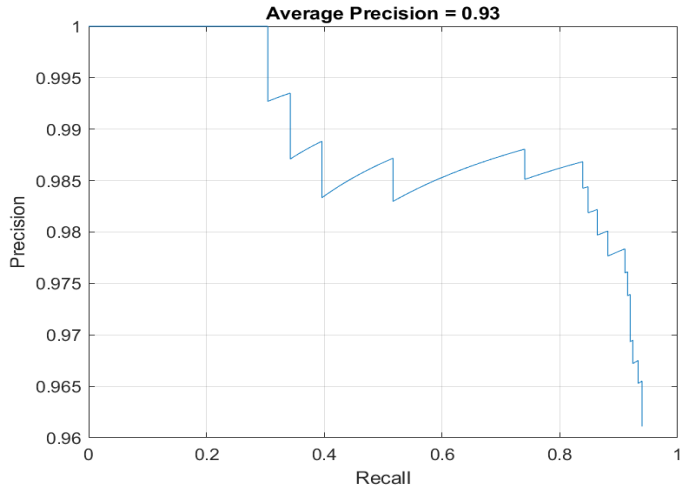
$$P = \frac{TP}{TP + FN}$$

Recall tells us how many of the actual positive instances the model was able to find. A high recall indicates that the model can identify a significant portion of the positive instances.

Essentially, precision measures how accurate the model's object predictions are, while recall measures how well the model can find all instances of objects in the scene. These two metrics are often used together to provide a balanced view of model performance. A common trade-off exists between precision and recall: increasing one may lead to a decrease in the other. Therefore, it's essential to consider both metrics when evaluating and fine-tuning object detection models like YOLOv4.

Figure 43 (a) – (b) – (c) illustrates the precision-recall curve for each class. With the evaluateDetectionPrecision() function of MATLAB precision and recall, for each detection, are computed and also the mean average precision for the entire class is evaluated.


Average Precision = 0.90
Precision
Recall
1
0.995
0.99
0.985
0.98
0.975
0
0.2
0.4
0.6
0.8
1


Average Precision = 0.93
Precision
Recall
1
0.995
0.99
0.985
0.98
0.975
0.97
0.965
0.96
0
0.2
0.4
0.6
0.8
1

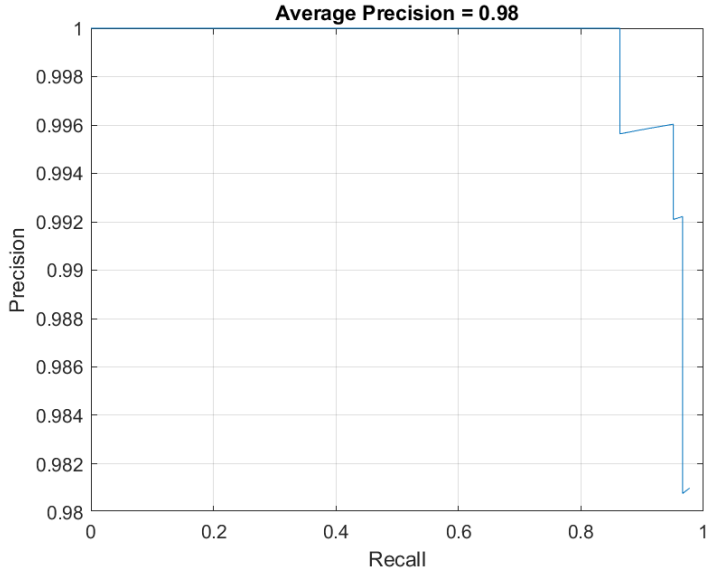

Average Precision = 0.98
Precision
Recall
1
0.998
0.996
0.994
0.992
0.99
0.988
0.986
0.984
0.982
0.98
0
0.2
0.4
0.6
0.8
1

Figure 43 PR curve with meanAP for (a) Gondola - (b) Motorboat - (c) Waterbus

The mAP values demonstrate a commendable performance across all classes, with each class exhibiting a consistently high precision even as recall increases. This can be attributed to the nature of the dataset created using Google Earth Pro, which primarily consists of straightforward cases. The decision was initially made to employ a simple labeling approach, ensuring the selected implementation functioned effectively. Subsequently, as additional data will be collected using our sensor, a more detailed labeling effort will be undertaken. The term "simple cases" refers to scenarios with minimal overlap among objects and a scarcity of boats situated at considerable distances. Additionally, the reliance on Google Earth Pro to construct the input database imposes limitations related to the available images from Google Street View.

A more extensive and diverse database would enable a more specific labeling operation. For the purposes of this thesis, priority was given to the fundamental development of the recognition algorithm, even at the expense of intricate labeling details.

Next a series of test images used for evaluate practically the detection.

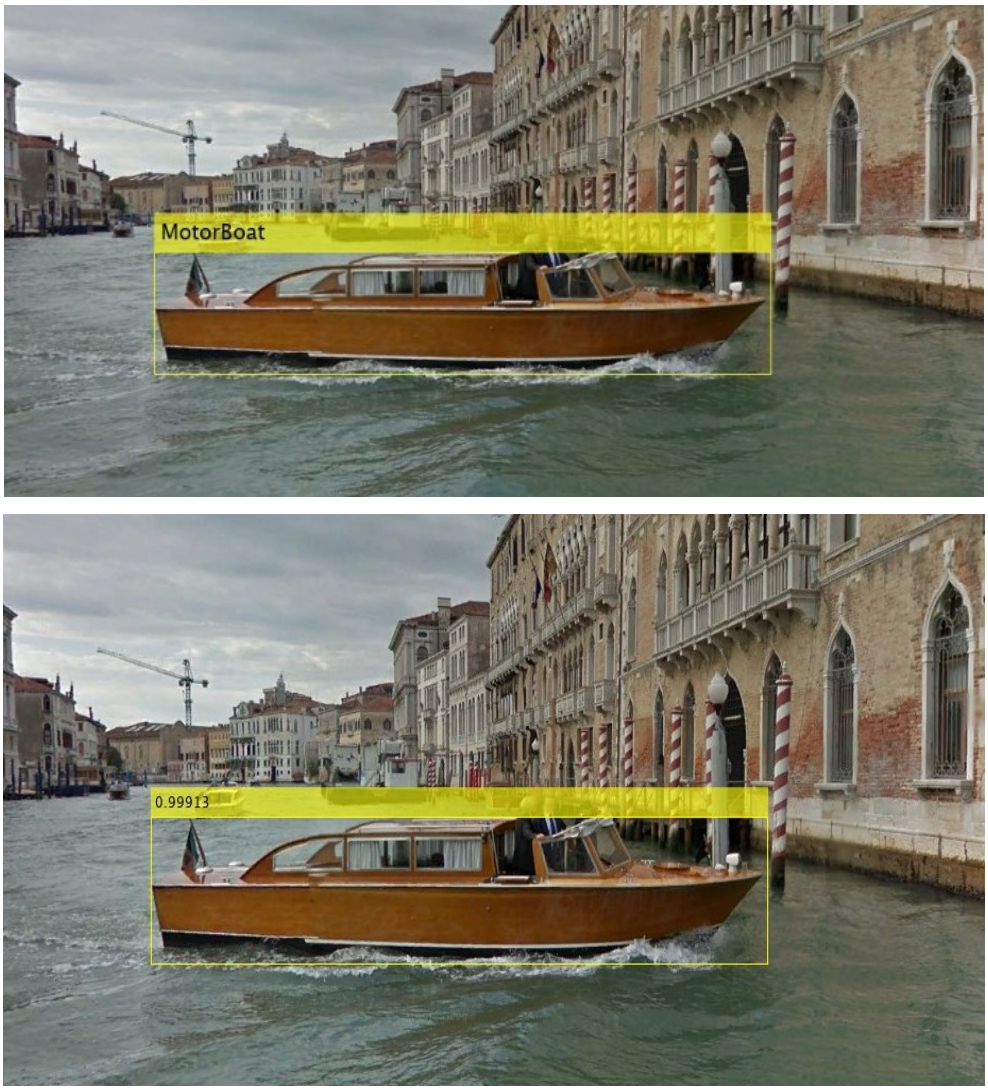


Figure 44 - Test image with a detection of a brown Motorboat

Figure 44 illustrates the detector's ability to accurately identify the boat's class in the image, accompanied by a remarkably high confidence score. Conversely, Figure 45 displays an image featuring two objects correctly classified by the detector into their respective classes. This demonstrates the trained detector's capability to differentiate between various classes within the same image, achieving a commendable confidence score.

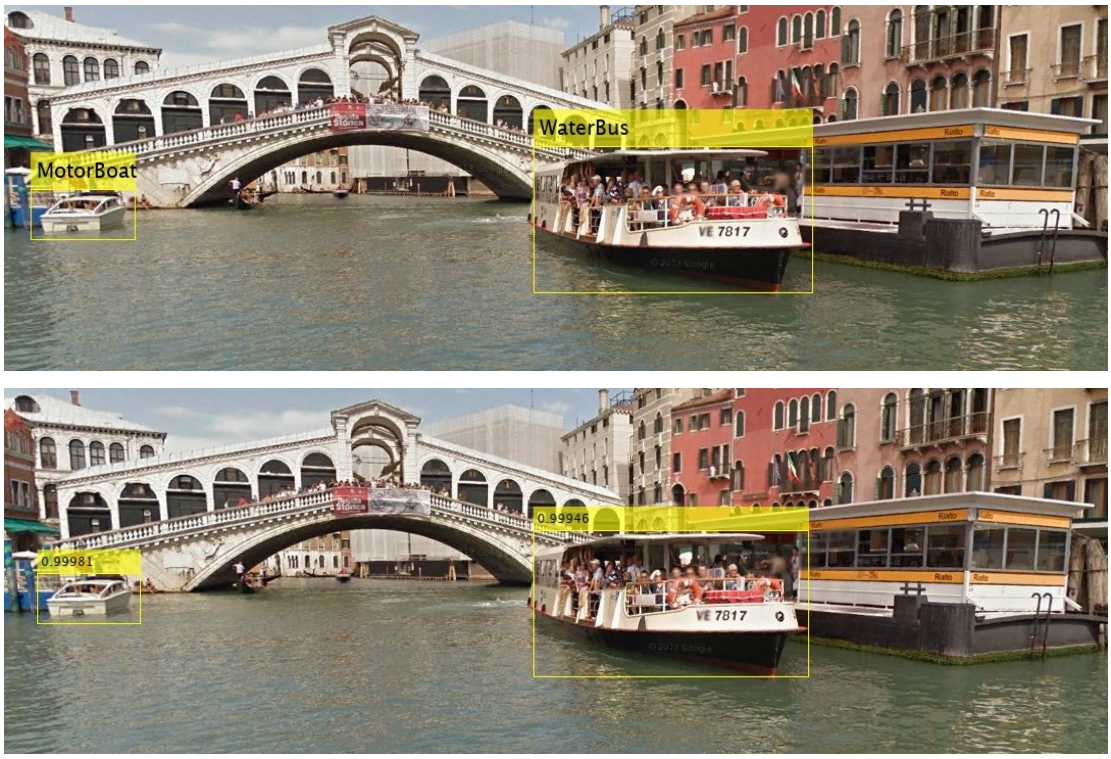


Figure 45 - Test image with a detection of two objects which belong to two different classes

Finally, figure 46 shows a scenario with the last class which is composed by Gondola boat.

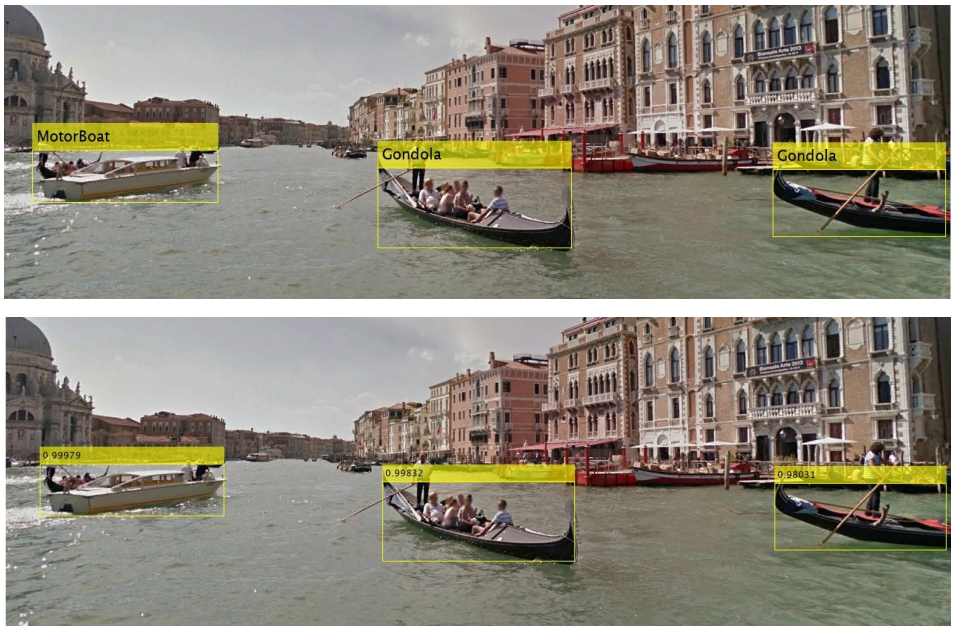


Figure 46 - Scenario with detected motorboat (1) and gondole (2)

6 Conclusions

This thesis focused on designing an autonomous guidance system for a small autonomous surface vehicle in the Venetian lagoon. The objectives included sensor selection, exploration of sensor fusion algorithms applicable in similar contexts, and the implementation of the YOLOv4 algorithm for boat recognition using a dataset created through Google Earth Pro's Street View feature. Sensor selection demonstrated the crucial integration of lidar, camera, parking sensors, and NVIDIA Jetson to address specific challenges in autonomous navigation within the Venetian lagoon, enabling accurate perception of the surrounding environment. The investigation into various sensor fusion algorithms provided the definition of a feasible algorithm for implementation by the ASV's processor when the model is constructed and operational. The synergistic integration of data from different sensors was highlighted as contributing to a more complete and reliable perception, allowing safer and optimal decision-making by the guidance system. Finally, the implementation of the YOLOv4 algorithm showed promising results in boat recognition. Precision, recall, and accuracy measurements attest to the model's effectiveness in identifying objects of interest in the Venetian lagoon, despite the use of a limited dataset due to restrictions imposed by the need to gather images from Google Earth Pro. The inability to conduct an autonomous data acquisition campaign, due to project delays and time constraints, limited the quality of labeling—a situation that will be addressed in greater detail once a proprietary and more extensive image database becomes available. The outcomes of this study provide a solid foundation for building the perception and collision avoidance phase when the vessel is ready to initiate a data collection campaign. As mentioned earlier, there were several limitations. The actual absence of the ASV, still under construction, and the consequent lack of sensors made it impossible to simultaneously collect LiDAR and camera data, preventing the implementation of the entire chosen Sensor Fusion algorithm. Additionally, the absence of online marine datasets for experimenting with external data was another limitation. However, the use

of Google Earth Pro allowed the creation of a sufficient image database to develop the camera-related branch of the fusion algorithm, focusing directly on a Venetian context.

In conclusion, the research achieved the outlined goals, providing a foundation for integrating the selected sensors and effectively adopting artificial intelligence techniques in the context of autonomous navigation in complex maritime environments.

In view of future developments, several directions are highlighted to consolidate and expand the work conducted in this thesis.

First and foremost, it is recommended to collect new images directly using the selected sensors. This step will enrich the dataset and allow for more detailed labeling, thus promoting optimal performance of the detector even in complex contexts characterized by extensive overlaps between objects to be identified. A second key aspect involves developing the branch of the sensor fusion process related to the Velodyne Puck LiDAR. This development, also enriched by collecting specific data in a maritime environment, can be integrated with the information previously acquired from the D455 camera. The goal is to further enhance environmental perception, improving the performance of the autonomous navigation system in complex maritime situations. Finally, there is a proposal to integrate the data processed through the selected fusion algorithm in this thesis to empirically evaluate its performance. This step will provide a more comprehensive view of the operational capabilities of the system and enable a practical assessment of the effectiveness of the sensor fusion algorithm in the specific context of the Venetian lagoon.

These development lines outline a roadmap for future research, allowing for continuous growth and refined optimization of the autonomous navigation system for the small ASV in the Venetian lagoon.

Bibliography

[1] E. Balestrieri, P. Daponte, L. De Vito, and F. Lamonaca, “Sensors and measurements for unmanned systems: An overview,” *Sensors*, vol. 21, no. 4. MDPI AG, pp. 1–27, Feb. 02, 2021. doi: 10.3390/s21041518.

[2] “SAE_AV levels”.

[3] J. Fayyad, M. A. Jaradat, D. Gruyer, and H. Najjaran, “Deep learning sensor fusion for autonomous vehicle perception and localization: A review,” *Sensors (Switzerland)*, vol. 20, no. 15. MDPI AG, pp. 1–34, Aug. 01, 2020. doi: 10.3390/s20154220.

[4] D. J. Yeong, G. Velasco-hernandez, J. Barry, and J. Walsh, “Sensor and sensor fusion technology in autonomous vehicles: A review,” *Sensors*, vol. 21, no. 6. MDPI AG, pp. 1–37, Mar. 02, 2021. doi: 10.3390/s21062140.

[5] A. Stateczny and W. Gierski, “The concept of anti-collision system of autonomous surface vehicle,” in *E3S Web of Conferences*, EDP Sciences, Nov. 2018. doi: 10.1051/e3sconf/20186300012.

[6] F. Castanedo, “A review of data fusion techniques,” *The Scientific World Journal*, vol. 2013. Hindawi Publishing Corporation, 2013. doi: 10.1155/2013/704504.

[7] “DATA FUSION LEXICON The DATA FUSION SUBPANEL of the Joint Directors of Laboratories, Technical Panel for C3 v\,” 1991.

[8] ThinkAutonomous, “9 Types of Sensor Fusion Algorithms.” Accessed: Oct. 21, 2023. [Online]. Available: https://www.thinkautonomous.ai/blog/9-types-of-sensor-fusion-algorithms/

[9] B. V Dasarathy, “Sensor Fusion Potential Exploitation-Innovative Architectures and Illustrative Applications,” 1997.

[10] "| www.velodynelidar.com High Resolution Real-Time 3D LiDAR Sensor Puck Hi-Res." [Online]. Available: www.velodynelidar.com

[11] S. Konatowski and A. T. Pieniężny, "A comparison of estimation accuracy by the use of KF, EKF & UKF filters," in *WIT Transactions on Modelling and Simulation*, 2007, pp. 779–789. doi: 10.2495/CMEM070761.

[12] "Implementation of vision... KALMAN FILTER example".

[13] A. Kassir and T. Peynot, "Reliable Automatic Camera-Laser Calibration."

[14] C. Debeunne and D. Vivet, "A review of visual-lidar fusion based simultaneous localization and mapping," *Sensors (Switzerland)*, vol. 20, no. 7. MDPI AG, Apr. 01, 2020. doi: 10.3390/s20072068.

[15] P. Wei, L. Cagle, T. Reza, J. Ball, and J. Gafford, "LiDAR and camera detection fusion in a real-time industrial multi-sensor collision avoidance system," *Electronics (Switzerland)*, vol. 7, no. 6, Jun. 2018, doi: 10.3390/electronics7060084.

[16] C. Rasche, "Computer Vision," 2019. [Online]. Available: https://www.researchgate.net/publication/336460083

[17] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," Apr. 2020, [Online]. Available: http://arxiv.org/abs/2004.10934

[18] P. Xu *et al.*, "On-board real-time ship detection in hisea-1 sar images based on cfar and lightweight deep learning," *Remote Sens (Basel)*, vol. 13, no. 10, May 2021, doi: 10.3390/rs13101995.