H2020 – Secure societies - Protecting freedom and security of Europe and its citizens
SU-DRS02-2018-2019-2020– Technologies for first responders – Research and Innovation Action (RIA)

Search & Rescue

Emerging technologies for the Early location of Entrapped victims under Collapsed Structures & Advanced Wearables for risk assessment and First Responders Safety in SAR operations

D3.6 Multi sensors data fusion and Object detection algorithms for in-disaster scene situation awareness

Workpackage:	WP3 – Situation Awareness
Authors:	THALIT
Status:	Final
Due Date:	M18
Version:	1.00
Submission Date:	31/12/2021
Dissemination Level:	PU

(*) Dissemination level.-PU: Public, fully open, e.g. web; CO: Confidential, restricted under conditions set out in Model Grant Agreement; CI: Classified, Int = Internal Working Document, information as referred to in Commission Decision 2001/844/EC.

Search and Rescue Project Profile

Grant Agreement No.: 882897

Acronym:	Search and Rescue
Title:	Emerging technologies for the Early location of Entrapped victims under Collapsed Structures & Advanced Wearables for risk assessment and First Responders Safety in SAR operations
URL:	https://search-and-rescue.eu/
Start Date:	01/07/2020
Duration:	36 months

Partners

	NATIONAL TECHNICAL UNIVERSITY OF ATHENS (NTUA) Co-ordinator	Greece
	AIDEAS OÜ (AIDEAS)	Estonia
	SOFTWARE IMAGINATION & VISION S.R.L (SIMAVI)	Romania
	MAGGIOLI SPA (MAG)	Italy
	KONNEKT-ABLE TECHNOLOGIES LIMITED (KT)	Ireland
	THALES ITAIA Italia SPA (THALIT)	Italy
	ATOS IT SOLUTIONS AND SERVICES IBERIA SL (ATOS)	Spain
	ETHNIKO KENTRO EREVNAS KAI TECHNOLOGIKIS ANAPTYXIS (CERTH)	Greece
	UNIVERSITA DEGLI STUDI DI CAGLAIRI (UNICA)	Italy

	UKEMED GLOBAL LTD (UGL)	Cyprus
	PUBLIC SAFETY COMMUNICATION EUROPE FORUM AISBL (PSCE)	Belgium
	UNIVERSITA DEGLI STUDI DI FIRENZE (UNIFI)	Italy
	DEUTSCHES FORSCHUNGSZENTRUM FUR KUNSTLICHE INTELLIGENZ (DFKI)	Germany
	UNIVERSITA CATTOLICA DEL SACRO CUORE (UCSC)	Italy
	VRIJE UNIVERSITEIT BRUSSEL	Belgium
	SYNYO GmbH (SYNYO)	Austria
	UNIVERSITEIT HASSELT (UHASSELT)	Belgium
	SPOLECZNA AKADEMIA NAUK (SAN)	Poland
	GIOUMPITEK MELETI SCHEDIASMOS YLOPOIISI KAI POLISI ERGON PLIROFORIKIS ETAIREIA PERIORISMENIS EFTHYNIS (UBITECH)	Greece
Search and Rescue End-Users		
	ELLINIKI OMADA DIASOSIS SOMATEIO (HRT)	Greece

	ENOSI PTYCHIOYCHON AXIOMATIKON YPAXIOOMATIKON PYROSVESTIR OY SOMATEIO (EPAYPS)	Greece
	JOHANNITER-UNFALL-HILFE EV (JOHAN)	Germany
	JOHANNITER OSTERREICH AUSBLIDUNG UND FORSCHUNG GEMEINNUTZIGE GMBH (JOAFG)	Austria
	CONSIGLIO NAZIONALE DELLE RICERCHE	Italy
	POMPIERS DE L'URGENCE INTERNATIONALE (PUI)	France
	ASOCIATA CLUSTERUL ROAMN RENTRU PROTECTIE SI ECOLOGIE IN DOMENIUL MATERIALELOR CHIMICE, BIOLOGICE, RADIOLOGICE/NUCLEARE SI EXPLOZIVE (PROECO)	Romania
	SERVICIO MADRILENO DE SALUD (SERMAS)	Spain
	FUNDACIÓN PARA LA INVESTIGACIÓN E INNOVACIÓN BIOSANITARIA DE ATENCIÓN PRIMARIA (FIIBAP)	Spain
	ESCUELA ESPANOLA DE SALVAMENTO Y DETECCION CON PERROS (ESDP)	Spain

Document History

Version	Date	Author (Partner)	Remarks/Changes
0.10	16/12/2020	Viola Sorrentino (THALIT)	ToC first draft
0.11	14/10/2021	Vincenzo Di Massa (THALIT)	THALIT contributions to named sections
0.12	05/11/2021	Marco Guerri (THALIT)	Format Internal review
0.13	19/11/2021	Gianluca Mandò	Internal Review with comments
0.14	23/11/2021	Vincenzo Di Massa (THALIT)	Added literature references
0.20	29/11/2021	Marco Guerri (THALIT)	General Internal Thales review and Partners submission for review
0.22	09/12/2021	Nikolas Mueller (DFKI)	Added details of how the images management from Robot.
0.30	07/12/2021	Ioannis Symeonidis (CERTH)	Internal Review
0.31	21/12/2021	Patrik Karlsson (AIDEAS)	Internal Review
0.40	24/12/2021	Marco Guerri (THALIT)	Added executive summary
0.41	28/12/2021	Iliana Malliou (NTUA)	Quality Control
1.00	31/12/2021	Christos Ntanos (NTUA)	FINAL VERSION TO BE SUBMITTED

Table of Contents

List of Figures

Executive Summary

This Deliverable called "Multi sensors data fusion and Object detection algorithms for in-disaster scene situation awareness" is the result of the Task 3.4 of Search&Rescue Project where the design and development of Obstacle Detection System (ODS) is described.

In particular the following topic are introduced: the scope of a Robotic solution with an ODS onboard, the importance of the ODS within the S&R Project and the specific advantages to have a multi sensor data fusion algorithm.

Then the System Overview is presented treating the related references, the Machine learning, AI, the Processing Unit, the Implementation notes and finally their integration in the selected S&R rescue Robot.

In the following chapters the main topics are described: the architecture of Obstacle Detection System with smart sensors, special synchronisation and depth estimation; the implementation of Obstacle Detection Algorithms and finally the Fusion and Tracking operations.

At the end of the deliverable there are references to the Robot System Integration (with recall to the D5.4) and the related Verification and Validation of the System.

1 Introduction

1.1 Scope

Robotic solutions that are properly sized with suitable sensors, modularised mechanical structures, semi-autonomous navigations systems that are well adapted to the conditions of the field can improve the safety and the security of personnel as well as their work efficiency and flexibility.

With its feet on the ground, the semi-automotive vehicles are experiencing their own industrial revolution, increasingly closer to aerospace technology. The race of car manufacturers and technological firms to take advantage of technology without a driver is a wake-up call for the first responders' industry. A semi-autonomous vehicle for first responders will be able to do three main things: First, you must know where it is. Secondly, it must be able to identify what is in the environment around. And finally, it must be able to make decisions about whether it moves or not. Sensors play a fundamental role in these matters. S&R engineering teams have worked hard together to identify the type of sensors that work best for this kind of applications.

The optical sensors and cameras are already the trend in autonomous vehicles like cars or trams. Moreover the Laser Imaging Detection and Ranging (LIDAR), for example, uses lasers to generate precise images in 3D. The work of the S&R partners have been to focus on the mechanisms of sensor fusion. These will allow the robotic vehicles to draw a clear image of the surroundings, combining data from several sensors.

The integration between Robot, all sensors, and others devices is detailed in the Deliverable D5.4 "*Testing of RESCUE MIMS on-board robotic platforms and drones*" while the purpose of this document is to describe the Object Detection System (ODS) and its main functionalities with respect to the Search and Rescue project. The system is introduced in section 2. In section 3, an overview of the system architecture is described, and technologies used for object detection and sensor fusion are described in sections 4 and 5. Finally, the test environment will be described in section 6.

1.2 The importance of Obstacle Detection System (ODS) within the S&R Project

Teleoperated robots has being successfully deployed in many rescue operations in urban environments with collapsed buildings and other structures [13]. From the after-action reports analysis, one of the main areas of improvement is considered the enhancement of situation awareness of the robots [12]. On this basis the obstacle detection system is developed from THALIT. The teleoperated or semi-autonomous robots require a system for detecting moving objects as humans, vehicles and rescue dogs in the field. The system can support the operator or be used for obstacle avoidance during path planning from the robot. The ODS is taking the full advantage of all the available robot sensors in order to estimate with high confidence the object position. The obstacle detection system is also tracking the moving objects and can further predict their future position, avoiding interfering with their motion. With the obstacle avoidance system, the robots can be safely used in the field and support the rescue units or move in areas dangerous for humans.

1.3 Advantages of multi sensor data fusion

Multi-sensors data fusion is used by the Object Detection System (ODS) to aggregate information coming from different sensors. The fusion is accomplished as a result of a Sensor Fusion Algorithm (SFA) which is executed after the object detection stage. For each type of input sensor, a detection model is implemented and used. The input sensors used in Search&Rescue are the following:

- LiDAR sensor, for which a clustering algorithm is implemented to detect objects;
- Stereo camera, for which a Convolutional Neural Network is used to detect objects.

The object detection task is further explained in chapter 4.

Once multiple objects are detected, the measures are aggregated and tracked. This is done with association algorithms (Global Nearest Neighbor) and tracking algorithms (Kalman filters). The Sensor Fusion task is further explained in chapter 5.

Due to the Sensor Fusion Algorithm, more robust measures can be achieved, by employing filtering approaches. Moreover, the use of a sensor fusion algorithm allows the robot to continue its task even in case of failure of a sensor (degraded mode).

2 System overview

2.1 Reference system

The Obstacle Detection System is a prototype developed within the T3.4. The main objectives are the following:

- Detect and track obstacles, fusing information coming from different technologies of sensors;
- Notify detected objects to the robot. The robot will use this notification to stop itself in case of a potential collision;
- Make the robot's pilot aware of the surrounding environment and notify him/her of obstacles detected;
- Cover different scenarios despite weather conditions and unavailable sensors.

Consequently, ODS main functions are:

- Acquire sensors (LiDAR and Camera) raw data;
- Filter and process raw data at sensors outputs;
- Detect obstacles in front of the robot, associating information from different sensors and tracking them.

The ODS block diagram is represented in Figure 1. The ODS architecture is composed of different modules with different responsibilities. These modules are combined in a pipeline in which each output of a module is the input of another one. ODS includes two different sensors, whose data will be fused together in order to improve obstacles detection probability. Moreover, the use of multiple sensors allows to improve the detection in different environments in which certain sensors perform better than the others. As an example, in nightly scenarios, a LiDAR sensor can be used to detect objects with more accuracy with respect to a non-infrared camera.

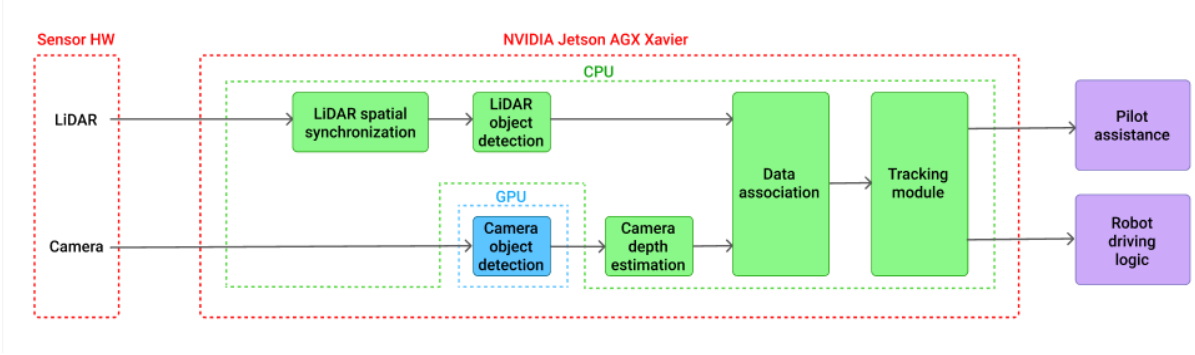

Figure 1: ODS block diagram

The ODS software runs on a dedicated NVIDIA Jetson AGX Xavier board, which is presented in section 2.3. Implementation notes for ODS modules are presented in section 2.4. In-depth analysis of ODS architecture is presented in Chapter 3.

2.2 Object detection

The object detection task allows to identify objects from the raw data coming from different sensors (LiDAR, camera). The data is represented in the same reference system, using the spatial

synchronization module. The algorithms used in the object detection task depend on the input data. In particular, objects are detected on input images through the use of a Convolutional Neural Network, which will be further explained in chapter 4.1. Objects are detected in LiDAR's point cloud input through the use of a clustering algorithm, which will be further explained in chapter 4.2.

2.2.1 Machine learning and AI

Machine learning approaches and, in particular, deep neural networks, will be deeply used by the Object Detection System to detect objects from the input data. Machine learning algorithms are based on the training task: a model is able to perform certain operations by using training data to learn how to approximate a certain function. Machine learning approaches are further described in chapter 4.1.

2.3 Processing unit

ODS runs on the NVIDIA Jetson AGX Xavier board with a Linux-based OS installed. This board features:

- CPU: 8-core ARM v8.2 (x64) @2.26 GHz, 8MB L2 + 4MB L3;
- GPU: 512-core Volta GPU @1.37 GHz with Tensor Cores;
- RAM: 32GB 256-Bit LPDDR4x (137 Gbps).

Figure 2: Nvidia Jetson AGX Xavier board

The Obstacle Detection Systems processes will be divided between the CPU and the GPU of the Xavier board, as showed in Figure 1, thanks to its 512-core Volta GPU.

The ODS will be installed on the robot through the use of a self-contained box, which will be now described. The ODS prototype is composed by the Peli 1400EU Protector Case including the Xavier board and a DC-DC convert from 24 V to 12 V. The dimensions of the ODS HW prototype are presented in Figure 3, together with the three interfaces. Reading Figure 3 from left to right, the following connectors are illustrated:

- RJ45 interface, for the Ethernet connection between the Xavier board and the robot;

- Power supply interface, which is a circular 3 pins connector;
- Green LED, which is turned on when the ODS HW prototype is fed

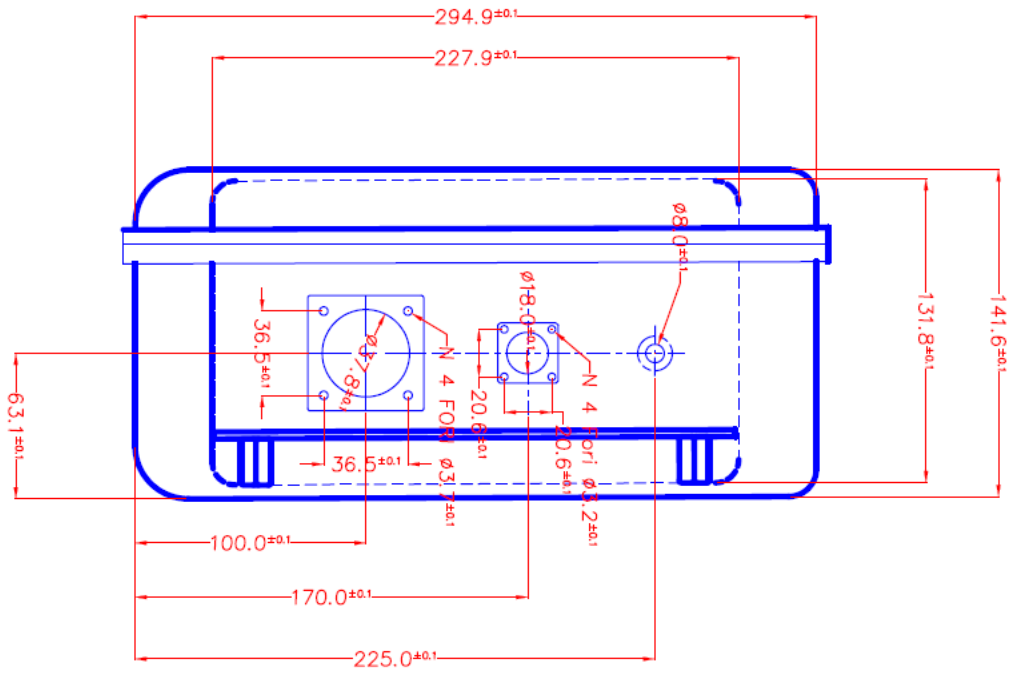

Figure 3: ODS Design of the interfaces of the HW prototype

2.4 Implementation notes

The Obstacle Detection System is based on the Robot Operating System (ROS) framework, which allows easy inter-process communication. ROS allows for modelling each functional block as a separate application which can communicate with the other modules using ROS messages.

The ODS system will be implemented in the aforementioned board in which ROS is installed. From the ROS ecosystem we borrow the libraries that allow different processes (applications) to exchange messages efficiently, as well as tools to record and play back a session of messages exchange. This technology is particularly useful when dealing with many different sensors, as it allows decoupling the main application from the software modules that link to the sensors and, in principle, to abstract from the technical details of a particular sensor, facilitating the possibility of changing the set of sensors in use.

Docker containers are used to implement separated and containerised microservices: each module of the pipeline runs in a Docker container and can communicate with the other modules via ROS messages. Containers are orchestrated using Docker Compose: using composition files written in YAML format, services that correspond to ODS modules can be defined. These services can easily be enabled/disabled to choose which module should be started and which not. Due to Docker containers, each module runs on a container generated by a predefined Docker image that can easily be managed to include the needed packages. This allows for easy isolation and prototyping for the ODS modules.

2.5 Integration of the Sensor Fusion Algorithm in the DFKI robot

The Obstacle Detection System will be integrated in the DFKI robot in order to detect objects from raw data coming from the robot's sensors.

In order to do that, the ODS will receive input data from:

- Stereo cameras: input data are the video frames captured from the camera sensors
- LiDAR sensor: input data are the 3D point cloud captured from the sensor

Objects detected by the system will be notified to the robot and to the driver Operator.

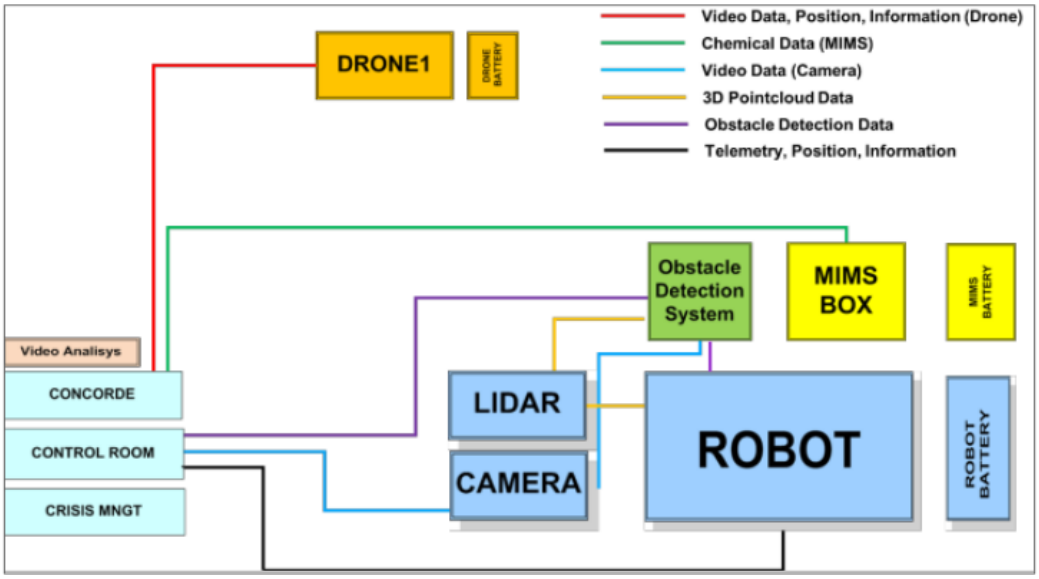

Figure 4: Data Connection Block Diagram

The Obstacle Detection System can provide the information regarding obstacles, such as bounding boxes position and objects speed to the robot using ROS messages. More information about the output format will be presented in Chapter 3.1. Moreover, the pilot operator can be guided in its operations by a visual support composed of JPEG images sent from the ODS. Raw images are collected from the cameras and the bounding boxes of camera-detected object classes will be drawn on them. These bounding boxes are enriched with the information coming from LiDAR sensor (position and speed data) and the resulting images are sent by the system to the operator's console.

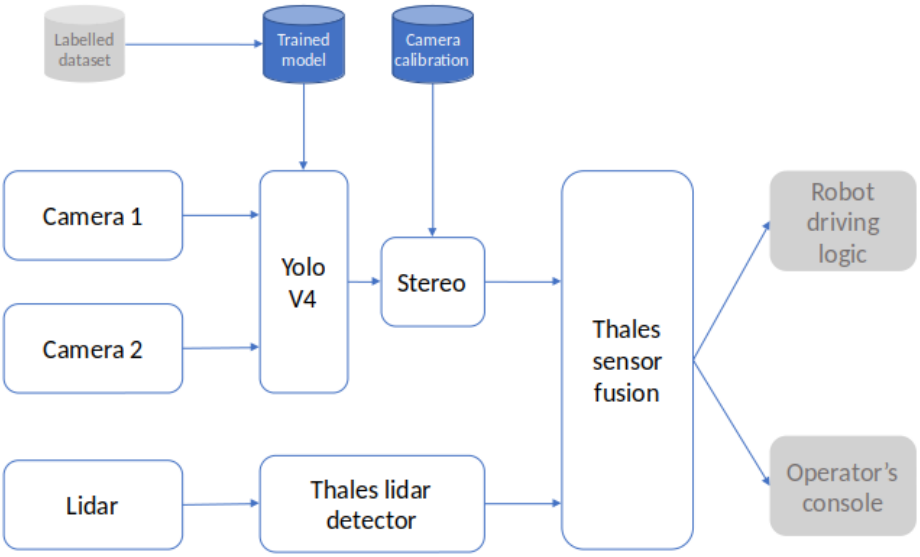

Figure 5: ODS integration for Search&Rescue

3 Obstacle Detection system architecture

3.1 Overview

The block diagram of the ODS system architecture, which will be implemented in the AGX Xavier platform, is represented in Figure 6. It is possible to distinguish seven different macro blocks:

- Smart sensors;
- Spatial synchronization;
- Object detection;
- Depth estimation;
- Data association;
- Tracking;
- Decision

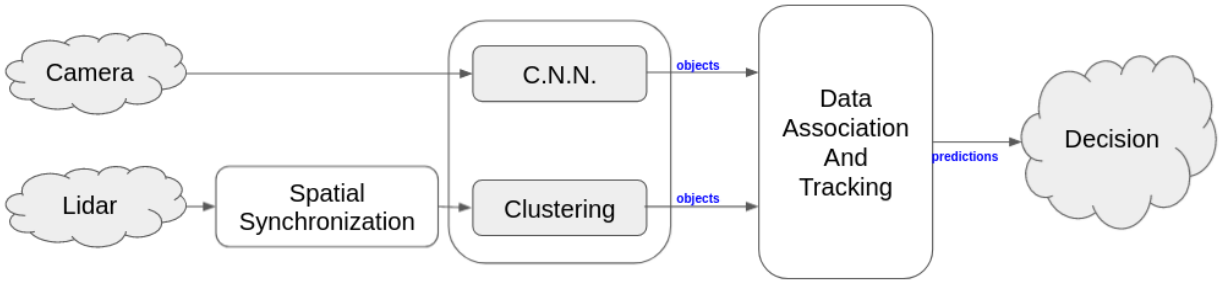

Figure 6: Block diagram of the ODS architecture

Inputs of the system are smart sensors: each block is composed by the hardware device and a sensor driver that is in charge of receiving the data in form of ROS messages and forwarding it to the following blocks.

Output of sensors can be processed by a spatial synchronization module, which is responsible for applying roto-translations to input data or to represent the information in the same reference systems. More details are presented in section 3.3.

Then objects are detected on sensors raw data. So:

- Convolutional Neural Network (CNN) will run on video frames, enclosing objects detected by the model in Bounding Boxes (BBs). In particular, the YOLOv4 network is used in this context, which is presented in chapter 4.1.
- Clustering algorithm will be implemented, to aggregate LiDAR 3D point clouds. This algorithm is presented in chapter 4.2.

Later, the depth estimation module represents the information (like central point coordinates, width, and length) on the robot Cartesian coordinate reference system. At the end of this process, objects detected by LiDAR and camera are represented in the same reference system and they can be compared.

The Sensor Fusion Algorithm is a fundamental and crucial stage of the ODS. This is composed of two steps: data association and tracking. They analyse at one time step the given data and decide on the most likely measurement-to-track associations. Objects detected coming from different sensors are

associated using the Global Nearest Neighbor (GNN) algorithm. A limited number of the associated objects can be tracked with Linear Kalman Filters (LKFs) or Unscented Kalman Filters (UKFs).

The following figure shows how objects coming from different sensors (LiDAR and camera) are associated and tracked from the Sensor Fusion Algorithm in a 2/3 configuration.

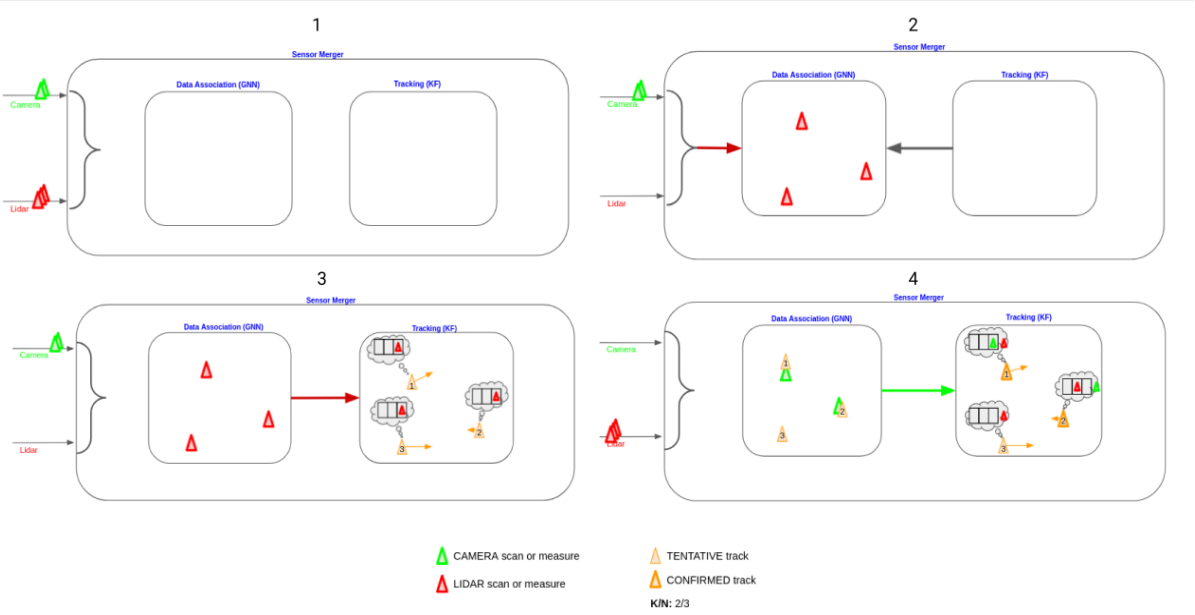

Figure 7: Data association & tracking of camera and LiDAR detected objects

In particular, the figure shows the steps accomplished by the system at different times:

1. Objects are detected from LiDAR and camera acquired data. These detections are sent to the Sensor Merger module;
2. LiDAR detected objects are received and processed by the module. The first step is the Data Association, in which the objects coming from different sensors are associated with an association algorithm (Global Nearest Neighbor);
3. The next step is the tracking of the associated objects. The tracking step is accomplished with the use of a Kalman Filter for each object. Objects are grouped in two categories: Tentative Tracks and Confirmed Tracks. This is done with the K/N logic: once enough measures are received for an object, the track become Confirmed. Since only a measure has been received for each object, each track is in Tentative state;
4. Camera detected objects are received and processed by the modules and the two steps (associations and tracking) are executed again. Thus, the camera measures are associated with the LiDAR measures to determine objects correspondence. Then Kalman filters uses the association data to update their state. In this case, two objects become Confirmed, since two measures out of three are received. The third object is still in Tentative state since no new measures has been associated with the previously received one.

In-depth analysis of SFA will be presented in chapter 5.

The output information of the ODS for the robot will be the following:

- Reference time of detected objects;

- Objects position (x, y, z) ;
- Objects speed;
- Measure of uncertainty of the given data;
- Objects class, which belongs to the set <Pedestrian, Car, Dogs>.

Moreover, the system will output JPEG images by aggregating raw images from stereo camera, bounding boxes detected from CNN and additional information, such as speed, coming from LiDAR data processing. This data will be sent to the operator's console to assist the operator in his/her tasks.

3.2 Smart sensors

3.2.1 Camera input

The cameras used for this integration form a stereo setup. Each camera will provide to the system the captured frames as ROS messages (*Image*) in the topics */stereo/cam_left/image_raw* and */stereo/cam_right/image_raw*, which contains the raw data encoded as RGB 8 bits. Moreover, size of the frames (width, height) is also provided in this message. Once a frame is received, it is forwarded to the ODS pipeline for further processing.

3.2.2 LiDAR input

The LiDAR used for this integration is a Velodyne VLP16. LiDAR data are 3D point clouds expressed in the sensor cartesian reference system and divided on 16 detection layers. The data is received by the ODS as ROS message (*PointCloud2*) in the topic */velodyne/velodyne_points* with a rate of 10Hz. The message contains, for each point, the following information:

- Cartesian coordinates (x, y, z)
- Intensity of the measurement
- Layer of detection

Once received, this data is forwarded to the pipeline as a ROS message.

3.3 Spatial synchronization

Raw data at the output of each smart sensor needs to be referred to the same reference system. The spatial synchronization is the module in charge of synchronizing in space the data collected from the sensors. The advantage of introducing spatial synchronization is to increase the probability of detection of the system.

To do this, the dedicated module applies coordinate transformation and roto-translations to compensate the offset of the various sensors' reference systems. Similar compensations are also needed to correct the orientation errors. These corrections must be tuned separately for each sensor and are stored in the ROS server.

Moreover, this module is capable of filtering the data before transmitting them to the downstream modules, according to different criteria:

- by defining a Region-of-Interest (ROI): discards everything that is outside of it;
- by defining filters on the radar metadata in order to discard unwanted radar tracks (e.g., invalid states, artifacts, etc.)

In particular, roto-translations are applied to LIDAR raw data in order to represent the detected points in the robot reference system.

The Figure 8 represents the location of the sensors with respect to the robot reference system. In particular, the following distances will be used by the Spatial Synchronization module:

- a = 0.08m
- b = 0.023m
- c = 0.08m
- d = 0.16m
- e = 0.658m

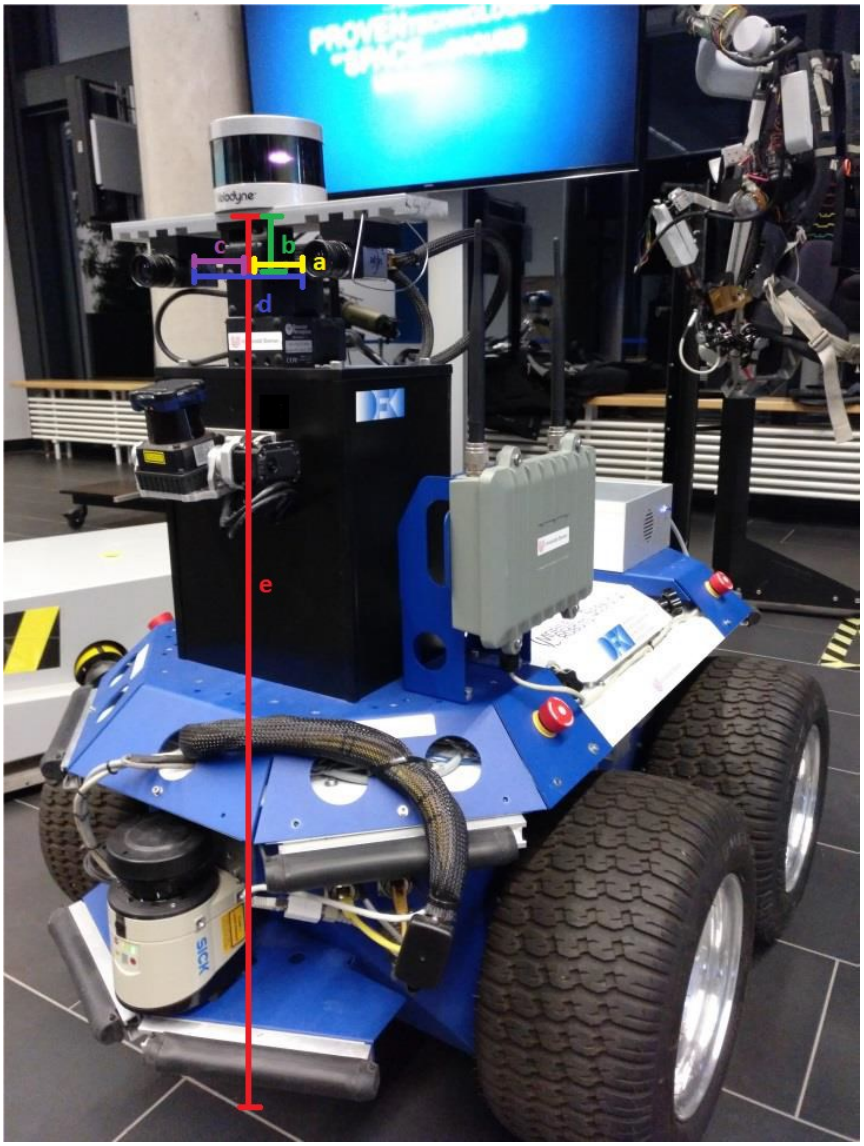

Figure 8: Seekur robot's sensors displacement

3.4 Depth estimation

Depth estimation exploiting multi-view geometry [1] is a technique used to estimate the depth of certain objects on a scene by leveraging on stereo images. Stereo images are a couple of synchronized images acquired by a stereoscopic camera setup, for which calibration parameters are known.

The synchronized couple of images can be used to evaluate a depth map: an image that contains distance information about the object in the scene. Using this technique, the depth of the object detected by the Convolutional Neural Network can be evaluated. This is done by using both the images that come from the stereo camera: first both frames are fed to the CNN model, then detected objects are matched between the two frames and finally depth estimation is used to find the depth of objects in matched pairs.

To evaluate the 3D position of an object, cameras must be modelled using the pinhole camera model, which approximates the behaviour of how the sensor represents a real-world point on the image plane. A camera can be mathematically described using this model, which is represented in the following equation:

$$s\,p = A\,\left[\,R|t\,\right]\,P_w$$

Figure 9: Camera Pinhole Model

In this equation, A (sometimes also named K in different notations) is the camera intrinsic matrix. R and T are, respectively, the rotation and the translation between the world coordinate system and the camera coordinate system. R and T together form the so-called extrinsic matrix. P_W is the world 3D point, s is the scale factor and p is the point represented in the image plane.

Intrinsic and extrinsic parameters can be obtained for a camera using a calibration procedure [14]. This approach consists in the estimations of intrinsic/extrinsic properties of the camera by leveraging on camera's multiple captures of a calibration pattern of a known size.

In order to retrieve the position information for a real-world point, various techniques have been developed over the years. The most common method to estimate the 3D position of a point consists in using a stereo camera setup. By using two sensors, a two-view of the scene can be obtained. As a result of this additional view, a certain point on the scene can be found on both image planes. This allows, under certain conditions, to easily estimate the 3D position by leveraging on stereo disparity. In particular, the disparity estimation in stereo vision relies on the epipolar geometry, which is a particular type of geometry that describes the relationships between the 3D points and their projections in the two image planes, which are related by constraints. These constraints allow to restrict the search space for points in an image plane to lines in the other image plane (epipolar lines).

The relationship of the points in the two views can be described using the Fundamental matrix, which describes the point correspondence in terms of epipolar geometry, and the Essential matrix, which describes this correspondence through the use of calibration parameters.

In order to estimate these constraints between cameras and their scenes, the relationships between the two stereo heads must be known. To do this, a stereo calibration procedure is needed to obtain the transformation that allows to operate a change of basis between the first and the second camera, and vice versa, and so, to apply epipolar geometry on the two scenes. Once these values are obtained,

rectification of images can be accomplished. Stereo rectification [15] allows the two image planes to lay on the same plane and so, to make all the epipolar lines parallel. This operation led to a great simplification of the search procedure, allowing for dense depth estimation. When the same point is found on the two image planes, the 3D point coordinates can be reconstructed through triangulation.

The operations needed to perform depth estimations are modularized and applied in the context of the ODS architecture to obtain 3D coordinates of objects detected through the use of the YOLOv4 model and by feeding these measures to the Sensor Fusion Algorithm.

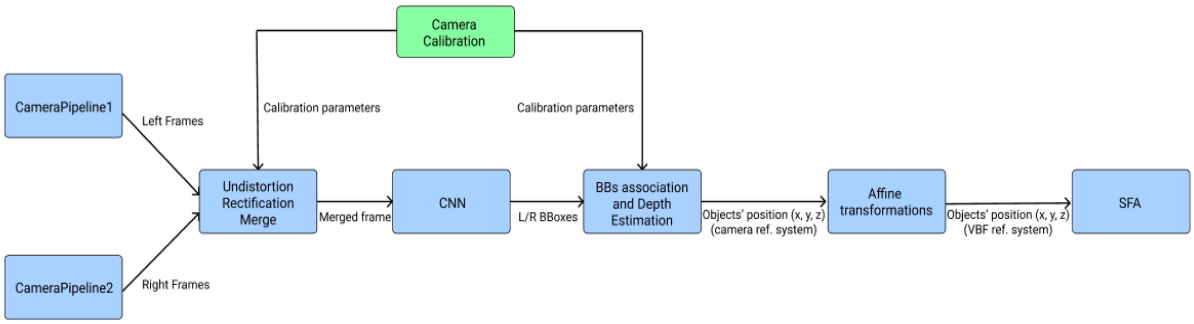

Figure 10: Depth Estimation Pipeline

In particular, the left and right camera's frames are received by a module that is in charge to separately undistort and rectify the left and right frames using the camera parameters previously obtained through calibration. The resulting frames are then merged and fed to the YOLOv4 module, which detects the objects in the merged images. The merged image and the bounding boxes are sent to the *StereoDepthEstimator* module, which is responsible of:

- associating left and right bounding boxes: each left frame's bounding box must be associated with one right frame's bounding box;
- detecting keypoints and extracting descriptors [16][17][18] in each bounding box;
- matching descriptors between associated bounding boxes;
- triangulate matched keypoints to estimate 3D coordinates of objects contained in bounding boxes;

These points in 3D coordinates are in one of the camera reference systems, therefore, to be able to use them along with 3D objects coming from the other sensors, they need to be referenced to the robot reference system. In order to do that, a series of affine transformations (rotations and translations) are applied to these points. Then, these objects are sent to the Sensor Fusion Algorithm which will use them as input measures to perform association and tracking operations.

4 Object detection algorithms

4.1 Object detection on video frames with CNNs

The object detection algorithm for S&R is an implementation of the single stage YoloV4 CNN algorithm [5]. The purpose of the algorithm implemented by the YOLO models [2] [3] [4] [5] is to perform real time detection of object instances appearing on the video frames from the camera sensors. Once the objects are detected into a video their detected properties are sent downstream through the processing pipeline. Downstream components refine the detection thanks to information from other sensors, filtering and tracking algorithms as described in Chapter 5.

The information provided by the YoloV4 algorithm about each object in each frame is composed of:

- An object bounding box that describes the position of the object in terms of the smallest rectangle on the image (x1, y1), (x2, y2) containing every pixel belonging to the object;
- An object class as an enumerated value (pedestrian, car, dog);
- A confidence value about the presence of an instance of an object of the detected class into the detected bounding box.

Said information is computed by means of a convolutional (artificial) neural network.

Artificial Neural Networks (ANNs) are a computation model inspired by the human brain. In ANNs the core computing units are called neurons. One simple and very used neuron type, the perceptron, computes an output vector given input vector by means of the application of a linear transformation. Typically, such operation is composed with a non-linear (activation) function. The term perceptron was originally used for just the linear neurons, but it is now common to use it also for nonlinear ones. The reason for the composition with a nonlinear function goes beyond the purpose of this text. In brief the nonlinearity is necessary to allow interconnected layers of neurons (feed forward neural networks, or multi-layer perceptrons, MLPs) to have a very important property: an MLP is a universal function approximator (Cybenko, in 1989, demonstrated it for sigmoid activation functions [6]). MLPs are theoretically able to approximate any function f, given enough computing power and a well-defined and big enough set of pairs of input vectors and corresponding output vectors computed using the f function itself. An input sample for which the output of the f function is known is called a labelled example. The parameters of the linear function are called weights and are "learned by example", i.e. specific "backpropagation" algorithms are used to fit the weights to minimize the average difference over the examples between the real f function output and the MLP output. Such a theoretical "universal function approximation" property of MLPs is in practice hindered by many problems which make their use impossible or very impractical in many contexts. The main reasons are the huge computing time necessary, the difficulty of obtaining well defined datasets and other problems mainly related to the "vanishing gradient" problem.

Object detection on images is one of the contexts where traditional MLPs have never been able to produce satisfactory results. For this reason, direct use of MLPs on images has been abandoned in favour of new techniques that allow to successfully train "deeper" networks, i.e. networks with many more layers, thus the name DNN, deep neural networks. DNNs can more efficiently cope with the specificities of the image-understanding related problems. DNNs are still using the same principles and similar algorithms to those used in MLPs but, like many modern neural networks, they can literally "bypass" the vanishing gradient problem thanks to very efficient techniques that allow building networks

with thousands of layers, compared with MLPs which typically are composed of just two layers. Indeed, theoretical results show that an MLP, to be able to approximate any function, just needs two layers. In practice, by building biologically inspired networks, practitioners have discovered that deeper networks can build intermediate representations in inner layers. The intermediate representations allow more structured "artificial reasoning" and thus provide better results, by far, compared to previous approaches. The DCNNs (deep convolutional neural networks) family of networks is special because it employs convolutional layers which have been a breakthrough for the adoption of neural networks on visual tasks. A 2D convolutional layer is a layer that applies a square convolution operation (called a convolution kernel) to its input. The convolution operation is performed by means of adapting a perceptron neuron as the 2D kernel: the 1D perceptron's output vector is interpreted as the concatenation of all the output columns of the convolution. The output of the convolution on the input 2D array is an either smaller or padded 2D array where each element is the result of the convolution of one rectangular sub-regions of the input with the kernel. It is important to note that the same input array element influences many "nearby" output array elements. Which output elements are influenced depends on the chosen "kernel size", but the magnitude of such influence is learned. Thanks to this mechanism, the presence of many consecutive convolutional layers generates the so-called inner neurons "receptive fields", i.e. each internal neuron focuses only on learning from its input region. Moreover, inner neurons process information already elaborated/simplified/abstracted by neurons in previous layers. In this way the last convolutional layers, the ones close to the final output of the network, are able to generate outputs which have large receptive fields and can learn very high level or abstract concepts.

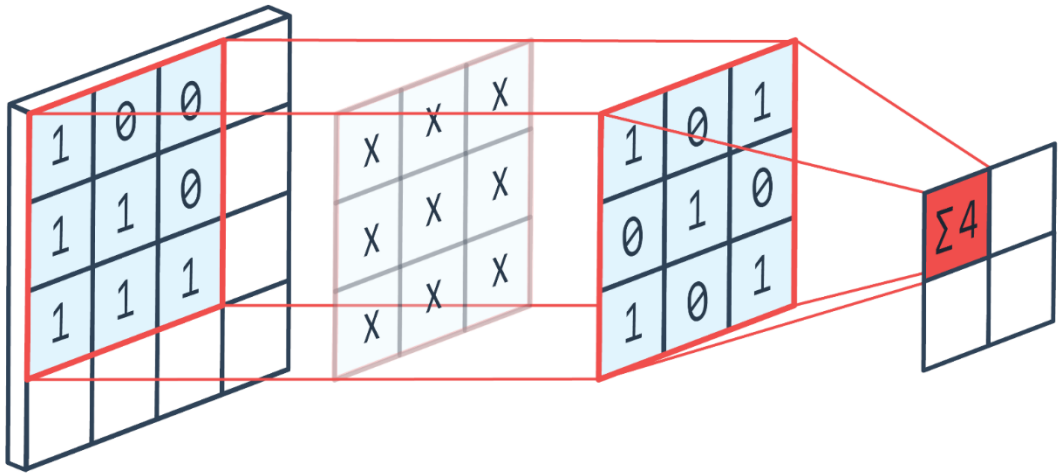

Figure 10: An example of a 2D convolution, X denotes the convolution parameters

The design from scratch of a new DNN network architecture is a very complex task specially because of the vast amount of design choices available. A common and very successful approach is to build new networks reusing or repurposing parts of successful networks. Moreover, it is possible to change just a little part of the architecture to specially fit the model to a new task. This is the approach followed by S&R: S&R is using a repurposed and retrained YoloV4 model.

The network architecture of the YoloV4 family of networks is described in the diagram below:

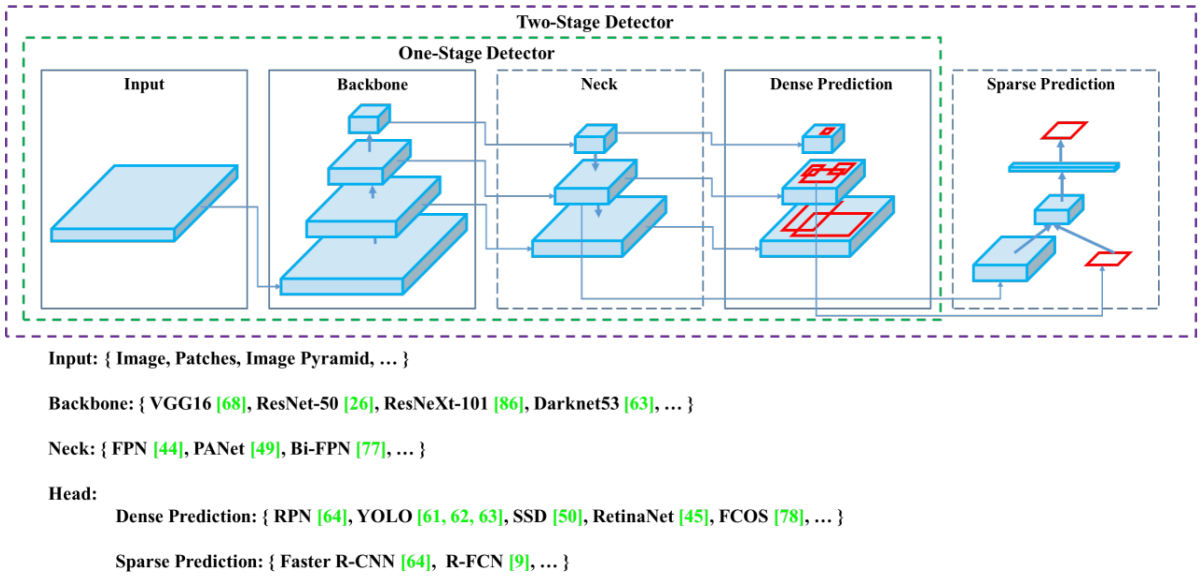

Figure 11: Two-stage and one-stage networks architectures

As can be seen from the image above, different alternative implementations are possible. Our implementation is 162 layers deep and has been trained on the MS COCO dataset. The network final layers are modified to give better results on a smaller number of classes compared to the full set of COCO classes. In particular, YoloV4 is an excellent choice for embedded systems that need to provide results with small latency.

In S&R implementation each video frame received from the camera is processed in a timeframe of approximately 30ms to 50ms and the generated output is sent a ROS message on the detected object ROS topic which spawns a list of detected objects data for every video frame.

4.2 Object detection on LiDAR point cloud

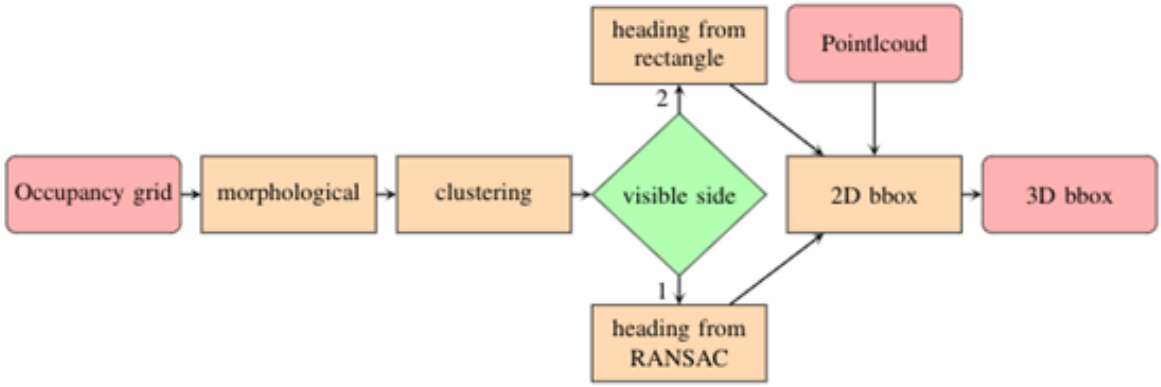

Figure 12: Obstacle detection pipeline

Processing Pipeline

We designed a custom pipeline for obstacle detection by implementing a multi-layer occupancy grid mapping [7]. The first steps are similar to classical geometrical approaches, as shown in Figure 12, until we compute the 2D occupancy grid. The only difference will be the size of the cell, which is smaller (i.e., 0.05 m) to allow us to perform operations directly on the grid.

The main differences are in the clustering. In a simpler pipeline, 2D bounding boxes can be retrieved directly from the grid, which are then used to extract a portion of the original PointCloud, and compute all the information on that one. In this scenario, since fitting a vertical plane on a single lidar layer is not feasible, we want to retrieve the heading from the 2D grid. To do so, after computing the connected components, we proceed to reconstruct a convex hull for each element. Then we differentiate two possible scenarios, displayed in Figure 13

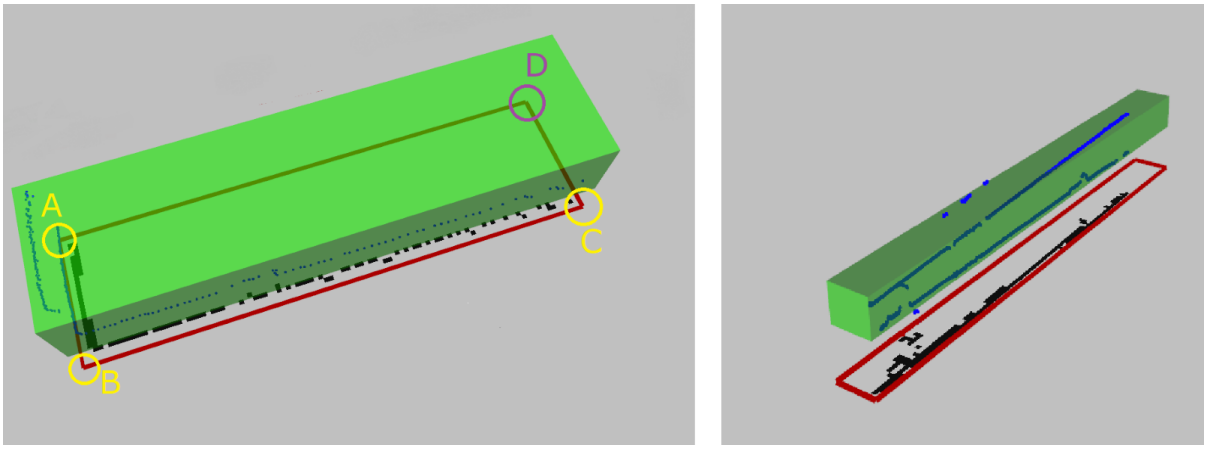

Figure 13: two possible scenarios of heading computation. Two sides of the obstacle are visible (left). Only one side is visible (right)

If both dimensions of the convex hull are above a fixed threshold, it means the obstacle is not perpendicular with respect to the lidar, and therefore two sides are visible, Fig. 13 (left). In this case, the convex hull generally has a triangular shape, since only two sides of an obstacle are visible. Accordingly, we retrieve the three vertices of the triangle and infer from those the possible fourth vertex. From those four points, it is possible to calculate the heading of the obstacle. In particular, to filter noise, we compute the relative angle of each side and average it, with the assumption of a rectangular shape for the obstacle.

If one size is under the threshold, we assume that the obstacle is parallel or perpendicular to the lidar, Fig. 13 (right). In this case, it is impossible to employ the described approach, and computing the heading based only on two points, which might be close together, would be excessively noisy. Therefore, we proceed to fit a line using the RANSAC algorithm on the occupancy grid points. The slope of this line is the heading of the obstacle. This solution is still noisier than computing the values on the two obstacle sides, but it allows us to retrieve the heading even when only one side of the obstacle is visible.

To increase the accuracy, we still take the cropped area of the original PointCloud around this 2D bounding box, using the PointCloud point to compute the size and center of the box and a minimum height. Since some obstacles might be described by only one lidar plane, we are not able to return the exact height of the obstacle, but only the maximum value retrieved by the lidar.

In this pipeline, we assume that obstacles have a rectangular shape to compute the heading. Since this requirement is not satisfied by a pedestrian, we check the convex hull size before performing the heading computation. If both dimensions are under a fixed threshold and are more similar to a pedestrian than a vehicle, this computation is not performed. We still compute the 3D box using the information from the grid and the original PointCloud, but we will not provide a heading.

Classification

The output of the previously described pipeline is a list of 3D bounding boxes. The final step is, therefore, to assign a class to each box. Since the number of points describing an obstacle is extremely low, and a geometrical approach has been employed to perform the detection, a rule-based solution is also used to perform classification. In particular, each box is described by a set of features; length, width, and height. Then we defined a list of thresholds for four possible classes of obstacles; pedestrian, vehicle, building, and undefined. To assign a class, we analyze each bounding box size and compare it to the thresholds. Finally, the unknown type is employed to cover all the scenarios where the combination of the three-dimension does not match any of the other three possible classes.

5 Fusion and tracking

The sensor merger module is the component that performs actual data fusion and tracking of detected targets. Each sensor transmits its data independently of others and according to its own data rate. Thus, each sensor's scan of the environment arrives at this module asynchronously. Inside this module, different policies can be implemented, in order to change the way the fusion is carried out:

- Complete asynchronous approach: each scan arrives as soon as it is available, and it is processed on its own. This approach is also known as completely serialized as all the inputs are serialized into a single input and processed on its own
- Synchronous approach: scans from the different sensors are hold until they are all available (at least one scans from each sensor), so that they can be fused together, and the result can be used to feed the tracking bank

Detections are compared to existing targets under tracking to perform data association via GNN (Hungarian algorithm). Associated detections are used to feed a KF-based tracking algorithm (UKF).

5.1 Data association algorithms

Global Nearest Neighbor (GNN) has been chosen as the method to select measurements to associate to a certain track. GNN is a sequential tracking method, so it processes in one step the set of candidate assignments and decides on the most likely measurement-to-track associations.

To give this method more robustness we can add constrained extra information. For example, we can impose that only detections with the following features:

- similar speed of the track;
- same class of the track;
- similar RCS of the track;
- similar width of the track;

are selected. A maximum likelihood estimation (MLE) function will be implemented to maximize the probability of detection. This is a method of estimating the parameters of a probability distribution by maximizing a likelihood function, so that under the assumed statistical model the observed data is most probable.

GNN considers all possible measurement-to-track assignments within appropriate gating regions and generates the most likely assignment hypothesis by solving a 2D binary assignment problem. The assignment hypothesis is used to set irrevocable assignments that cannot be modified by future data. The GNN method consists of the following steps:

- Predict the measurements and their covariances to estimate the validation gates.
- Compute the cost for all possible measurement-to-track assignments within each gating region.
- Formulate the 2D assignment problem and obtain a global optimal solution as the best assignment hypothesis.
- Perform tracking by updating the state of each object and its covariance from the assignment result.

5.2 Object tracking

The tracking block has two sub-modules: the track manager and the update/prediction track.

- In the track manager the logic to manage a track is implemented. This logic is the K/N method, which can be implemented in disjoined time frames or using a sliding window, according to which N detections are evaluated. The K/N logic converts a Tentative track in a Confirmed track if it appears K times out of N.
- The update/prediction track sub-block is composed of a list of UKF, one for each detected object. The number of tracked objects and consequently the number of implemented UKF will be optimized according to the maximum computational load of the Xavier platform, favouring the nearest objects to the robot.

5.2.1 Kalman Filters

The Kalman Filter (KF) [8] is an algorithm that provides estimates of some unknown variables given some of their observed measurements over time. It implements optimal predictive estimators, based on a set of mathematical equations which minimize the covariance of the estimated error, when certain assumed conditions are satisfied. The Kalman Filter can be used for linear problems under the assumption that Gaussian noise is added to the system. However, KF solutions are far from optimal in a non-linear system with non-Gaussian noise added.

Extended Kalman Filter (EFK) [9] [10] has been developed over time in order to implement an optimal predictive estimator for non-linear systems with Gaussian noise. EFKs have the same structure of KFs, but the first uses Taylor expansions to linearize and approximate all the non-linear transformations. EFK relies on Jacobian matrices to substitute the linear transformations of the original Kalman Filter.

Unscented Kalman Filter (UFK) [11] provides a further improvement to EFK by lowering the computational complexity avoiding the need to compute the Jacobian matrices and achieving better accuracy in many scenarios. The basis of the UKF is the unscented transformation, which is a method for approximating the moments of a non-linear random variable. Using this tool, the UKF approximates the probability density obtained from the non-linear transformation of such random variable. This is done by evaluating the non-linear function with a minimal set of carefully chosen sample points called sigma points. These sigma points are then propagated through the true non-linear system and allow estimation of the posterior mean and covariance that is accurate to the third order for any non-linearity.

Although the EKF and UKF are computationally efficient, it is worth remembering that their accuracy is limited by the validity of the approximations necessary to arrive at closed-form expressions for the posterior probability density function.

The Object Detection System can use the Linear Kalman Filter and the Unscented Kalman Filter to track objects. The type of filter to be used will be chosen after tests on the real environment.

6 Robot System Integration (refer to D5.4 "Testing of RESCUE MIMS on-board robotic platforms and drones")

6.1 Robot Functional requirements

To provide first responders with crucial information of the environment in form of spatial maps, the robot needs a proper network interface as well as hardware interface to mount the necessary tools for the job. This includes in addition to ODS

a) a 4G/WiFi router
b) an additional voltage regulator to provide power to the ODS
c) an ethernet hub to connect these components, so they can access the network independently as well as work as a unit for controlling the robot.

The sensors of the robot need to provide the data to the Obstacle Detection System, which then provides obstacle information to the robot for semi-autonomous control as well as to the driver in the control room for manual control, in case the semi-autonomous system fails or does not lead to desired robot operation.

In particular, the Robot will integrate the Obstacle Detection System in order to elaborate the output of the Detection SW and to decelerate/stop the movement, the ODS will provide information to the onboard Robot System by outputting fused obstacles position and velocity data in the robot reference system on a ROS topic.

7 Verification and Validation of the multi sensor data fusion algorithm in laboratory

7.1 Test setup

Tests for the Object Detection System can be grouped in two categories:

- Tests on synthetic data to evaluate the performance of the sensor fusion algorithm;
- Tests pre-built datasets provided by DFKI;
- Tests on target to evaluate correctness of input and output data in the deploy environment.

The tests on synthetic data will be executed on the host environment by simulating the objects' positions for certain time steps. This methodology allows to assess the quantitative accuracy of the Sensor Fusion Algorithm;

The tests on the pre-built datasets allow to validate the correctness of the system with respect to the input data. For these tests, ROS files are provided, which allows to test the input data handling and to qualitatively assess the precision of the ODS in the context provided by the bag file. Figure 14 shows two frames extracted by the provided videos, which show two objects in a possible context of operation.

Figure 14: Contextual frames of provided videos

The tests for the target environment will be executed on the target device. All Docker containers have to be switched on. Feeding the ODS HW prototype and connecting the Ethernet cable to the robot, it is possible to check if the defined ROS topics are published and their consistency. The contents of ROS

messages are under definition according to the developments of the ODS algorithms. These tests are aimed to prove the correctness of the pipeline implementation from a qualitative point of view: quantitative measurements cannot be obtained without the object's ground truths.

7.2 Test description

- Synthetic tests: synthetic object positions will be fed to the system and the output of the Sensor Fusion Algorithm will be compared to simulation data to evaluate the performance of the algorithm. The Figure 14 the pipeline that is used to obtain quantitative results on synthetic data.
  The following errors and KPIs are defined to measure the accuracy of the Sensor Fusion Algorithm:
  Association errors:
  - False positive: output that does not describe an actual (annotated) target
  - False negative: output target that is missed by any hypothesis
  - KPI: Multi Object Tracking Accuracy (MOTA): tracker's overall strength based on the sum of false positives ratio, false negatives ratio and mismatches ratio.

  Tracking errors:
  - Root mean square error: square root of the mean of the squared differences between original and estimated positions
  - KPI: Multi Object Tracking Precision (MOTP): average dissimilarity between all true positives and their corresponding ground truth targets.
- Datasets tests: the ODS runs on the host environment and the provided bag files are played.
- Target tests: the robot will have to be controlled using manual control and the semi-autonomous exploration algorithm in an outdoor and indoor environment. The environment must contain obstacles (like pedestrians, cars and dogs), which the ODS can detect.

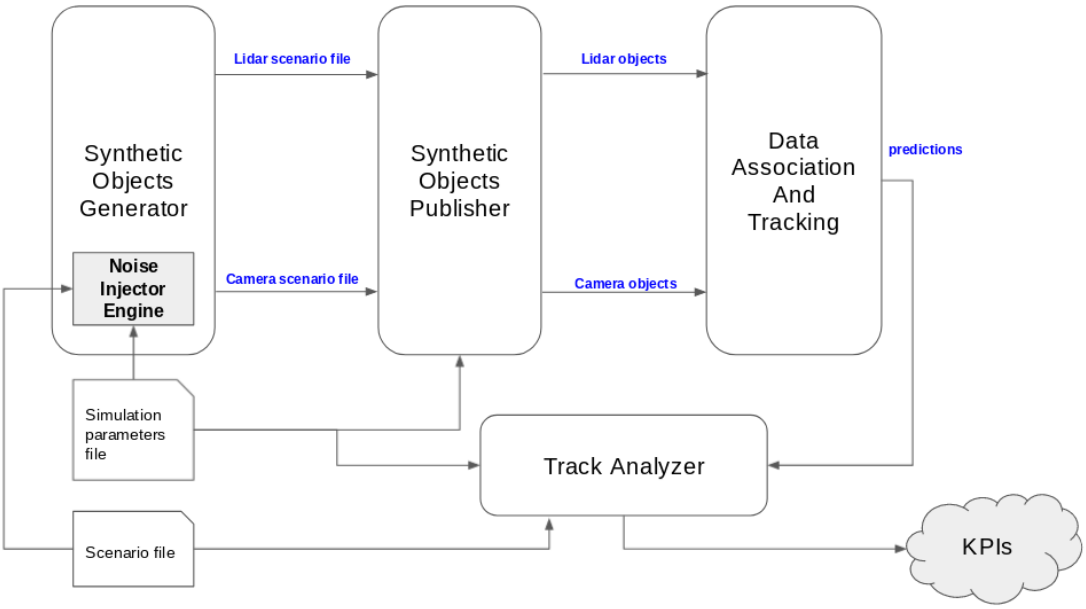

Figure 15: Synthetic test pipeline

7.3 Test results

- Synthetic tests: KPIs will be generated using the comparison of SFA output and simulation data;
- Datasets tests: the ODS must correctly handle the dataset ROS topic and the message exchange. The system must detect and track objects with enough qualitative accuracy on the provided datasets.
- Target tests: the main expectation is the detection of three types of elements during the Robot movement: Pedestrian, Cars and Dogs. The output of the ODS SW will be towards the Robot in order to hard stop the Robot operation moreover to show the camera images with the Detections to the Robot Driver.

8 Conclusion

This document has presented the Object Detection System (ODS) developed for task 3.4. This system allows obstacle detection and tracking by fusing information coming from different sensors that are mounted on the robot (cameras and LiDAR). ODS overview has been given in chapter 2, by describing the features, the role within the overall system and the implementation notes. In chapter 3, the ODS architecture has been described by examining the module of its pipeline and its input/output data. Then, in chapter 4 and 5, the main technologies used by the system have been described. In particular, for the object detection task, CNN (camera) and clustering (LiDAR) methods have been presented in chapter 4. The data association and tracking task has been presented in chapter 5 by describing the main algorithm used (GNN and LKF/UKF). Finally, in chapter 6, the testing methodologies have been presented. Test methodologies can be grouped in two categories: synthetic tests to perform quantitative and qualitative analysis on simulated data, and tests on the real environment to prove the correctness of input/output data with qualitative measurements.

The Object Detection System has been described in its key features. Thanks to them, the system can accomplish the objectives described in chapter 1.2.

Annex I: References

[1] Hartley, R., Zisserman, A. (2003). Multiple View Geometry in Computer Vision. New York, NY, USA: Cambridge University Press. ISBN: 0521540518

[2] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv.

[3] Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. arXiv.

[4] J. Redmon and A. Farhadi (2018). Yolov3: An incremental improvement. arXiv.

[5] Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv.

[6] Cybenko, G.V. (1989). Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals and Systems, 2, 303-314.

[7] Mentasti, S., & Matteucci, M. (2019). Multi-layer occupancy grid mapping for autonomous vehicles navigation. 2019 AEIT International Conference of Electrical and Electronic Technologies for Automotive (AEIT AUTOMOTIVE), 1-6.

[8] Kalman, R. E. & Others (1960). A new approach to linear filtering and prediction problems. Journal of basic Engineering, 82, 35—45

[9] H. W. Sorenson, editor (1985). Kalman filtering: theory and application. IEEE Press.

[10] J. K. Uhlmann (1992). Algorithms for multiple target tracking. American Scientist, 80(2):128–141

[11] Wan, E. & van der Merwe, R. (2000). The Unscented Kalman Filter for Nonlinear Estimation

[12] Murphy, R. R., & Burke, J. L. (2005, September). Up from the rubble: Lessons learned about HRI from search and rescue. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting (Vol. 49, No. 3, pp. 437-441). Sage CA: Los Angeles, CA: SAGE Publications.

[13] Davids, A. (2002). Urban search and rescue robots: from tragedy to technology. IEEE Intelligent systems, 17(2), 81-83.

[14] Zhengyou Zhang (2000). A flexible new technique for camera calibration. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 22(11):1330–1334

[15] Richard I Hartley (1999). Theory and practice of projective rectification. International Journal of Computer Vision, 35(2):115–127.

[16] Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision 60, 91–110.

[17] Bay, Herbert & Tuytelaars, Tinne & Van Gool, Luc. (2006). SURF: Speeded up robust features. Computer Vision-ECCV 2006. 3951. 404-417. 10.1007/11744023_32.

[18] S. Leutenegger, M. Chli and R. Y. Siegwart (2011). BRISK: Binary Robust invariant scalable keypoints. 2011 International Conference on Computer Vision, pp. 2548-2555, doi: 10.1109/ICCV.2011.6126542.