
# SEMFIRE

## Segurança, Exploração e Manutenção de Florestas com Integração de Robótica Ecológica

## D3.1b: Artificial Perception System and Robotic Decision-Making – Artificial Perception Base System






Cofinanciado por:



# Document Identification

| | |
|---|---|
| **Title** | **SEMFIRE: Segurança, Exploração e Manutenção de Florestas com Integração de Robótica Ecológica** |
| **Reference** | CENTRO-01-0247-FEDER-032691 |
| **Duration** | 2018-10-01 - 2021-09-30 |
| **Head of Consortium** (*Chefe do Consórcio*) | Ingeniarius, Lda. (Ingeniarius) |
| **Copromoters** (*Copromotores*) | Instituto de Sistemas e Robótica da Universidade de Coimbra (ISR-UC), SFera Ultimate, Lda. (SFera) |
| **Coordinator** | Micael S. Couceiro (Ingeniarius) |

## Work Package/Deliverable

| | |
|---|---|
| **Work Package** (*Atividade*) | **A3. Perception and Decision-Making Systems for Forestry Robotics** (*Sistemas de perceção e decisão para robôs em ambiente florestal*) |
| **Deliverable** (*Entregável*) | **D3.1b**: Artificial Perception System and Robotic Decision-Making – Artificial Perception Base System (*Sistema de Percepção Artificial e Tomada de Decisão – Sistema Base de Percepção Artificial*) |
| **Lead Copromoter** | Instituto de Sistemas e Robótica, Universidade de Coimbra |
| **Due Date** (*Mês*) | 2021-05-31 (*M32*) |

## Change Record

| **Version** | **Date** | **Author** | **Summary of Changes** |
|---|---|---|---|
| v0.1 | 2021-11-18 | João Filipe Ferreira | First version of the document |
| v1.0 | 2021-11-21 | David Portugal & João Filipe Ferreira | Final Version |

# Contents

# List of Figures

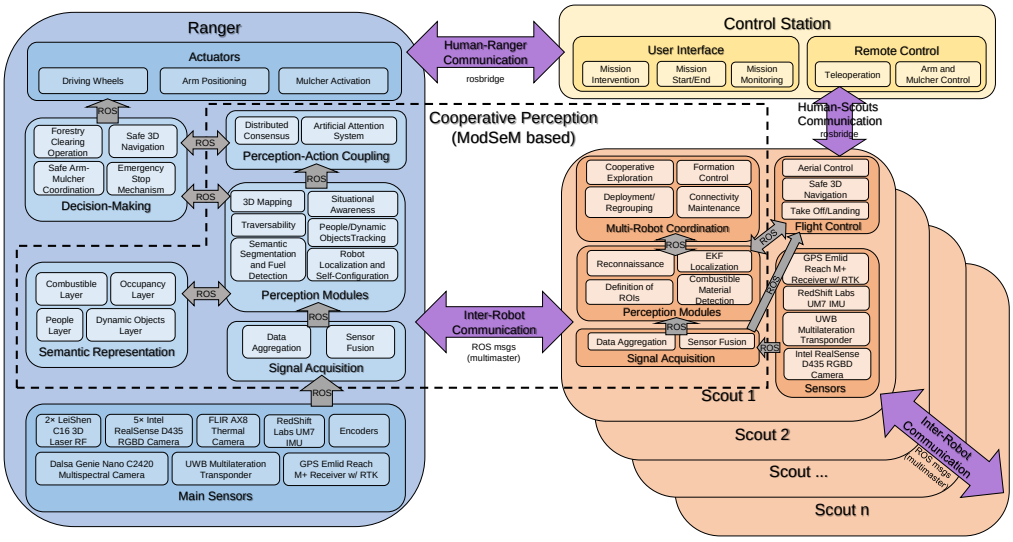



Figure 1: SEMFIRE perception architecture overview. MoDSeM is introduced in section 2.2.

# 1 Introduction

Despite many advances in key areas, the development of fully autonomous robotic solutions for field robotics, in particular for precision forestry is still in a very early stage. This stems from the huge challenges imposed by rough terrain traversability [23], for example due to steep slopes, autonomous outdoor navigation and locomotion systems [22], limited perception capabilities [7], and reasoning and planning under a high-level of uncertainty [21]. Artificial perception for robots operating in outdoor natural environments has been studied for several decades. For robots operating in forest scenarios, in particular, there is research dating from the late 80s-early 90s – see, for example, Gougeon et al. [6]. Nevertheless, despite many years of research, as described in surveys over time [e.g. 24, 12, 15], a substantial amount of problems have yet to be robustly solved.

In this deliverable, we report on the latest developments of our research concerning artificial perception algorithms in field robotics. In this text, we put forth the overview of the perception architecture for the SEMFIRE team of robots (section 2.1), and propose a core design of the perception pipeline (section 2.2) and decision-making module functionality (section 2.3). We then present the technical details of the integration of the software and hardware modules of the Ranger robot in the ROS middleware in Section 3, and provide in Section 4 an overview of the first internal tests developed to validate and verify the proposed artificial perception algorithms. Lastly, section 5 presents the main conclusions of the document and the main lines of future work.

# 2 Artificial Perception Base System

## 2.1 Overview of SEMFIRE perception architecture

In Figure 1, an overview of the SEMFIRE perception architecture is presented, including all perception modules and communication channels that form the framework for individual and cooperative perception capabilities of the SEMFIRE robotic team, but also the sensing and actuating backbones, the decision-making modules (to be detailed in section 2.3) and the Control Station used by the human operator (i.e. the *Foreman*) to remote-control the robots if needed.

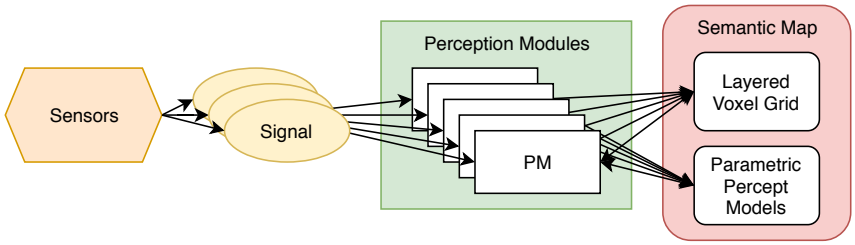



Figure 2: An overview of the framework. Sensors produce signals, which are passed to independent perception modules. Percepts gathered by these modules are aggregated in a Semantic Map, containing layers for different types of information.

Preliminary details for the subsystems involved will be described in the following subsections.

## 2.2 Core design of perception pipeline

SEMFIRE implies cooperative perception, in which each of the members of the robot team contribute to the global knowledge of the system by sharing and cooperatively processing data and percepts from one another, combining the sensorial abilities, perspectives and processing power of various agents to achieve better results.

In the scope of the SEMFIRE project, the **Mo**dular Framework for **D**istributed **S**emantic **M**apping (MoDSeM) [18, 17] was designed, which provides a semantic mapping approach able to represent all spatial information perceived in autonomous missions involving teams of field robots, aggregating the knowledge of all agents into a unified representation that can be efficiently shared by the team. It also aims to formalise and normalise the development of new perception software, promoting the implementation of modular and reusable software that can be easily swapped in accordance with the sensory abilities of each individual platform.

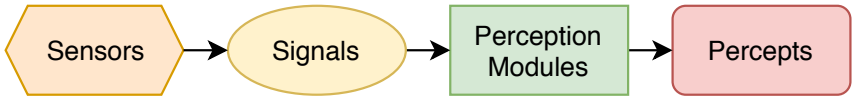
The framework is split into three blocks, as depicted in Fig. 2:

- The **sensors**, which provide raw signals;
- The **Perception Modules** (PMs) which take these signals and produce percepts, *i.e.* processed information;
- The **Semantic Map** (SM), containing a unified view of the state of the workspace/world.

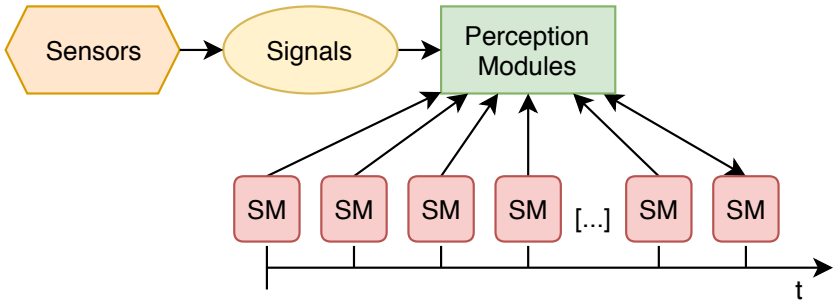
Sensors produce raw signals, which are used by Perception Modules to produce percepts, in this case information grounded on a spatial reference such as occupancy, presence of people, etc. These are taken as inputs to build a Semantic Map, which can in turn be used by any agent in the team to make decisions or coordinate with others. In this case, each sensor is seen as a mere source of data, which is assumed to be preconfigured and working as needed at the time of execution.

In software terms, each Perception Module is expected to be decoupled from all other modules of the system, depending only on sensors and on the Semantic Map itself. Thus, we can ensure that Perception Modules are interchangeable elements of the system, allowing us to swap them at will, depending on the computational power and available sensors on each robot. This allows for great flexibility in the deployment of the framework, enabling modules to be employed in each system without the need to re-design the global representation.

Perception Modules can use the Semantic Map as input, thus making use of the percepts from other techniques. Removing modules from the robot should not impact the remainder of the system. However, this may still result in a cascade reaction, with Perception Modules depending on each others' output for further processing; Perception Module selection should still be a careful process.




(a) Traditional ROS-based perception techniques implement a linear flow from sensors to percepts; signals are processed and percepts are output.




(b) MoDSeM aims to implement a non-linear perception pipeline: Perception Modules can access previous versions of the Semantic Map to execute over previous states.

Figure 3: Linear data flow compared to MoDSeM's non-linear data flow.

The semantic map works as a global output of the perception system. It is split into the Layered Voxel Grid (LVG) and the Parametric Percept Models (PPM). The LVG is composed of a layered volumetric representation of the workspace. Each layer of the LVG is itself a voxel grid, containing information on one aspect of the world, such as occupancy, traversability, relevance to current task, etc. The combination of these layers provides an overview of the state of the world as perceived by the robot team; individually, they provide insight that may be relevant on a particular aspect of the mission. Different Perception Modules can contribute to different layers of the LVG, with for instance a people detector contributing to a people occupancy layer or a mapping technique contributing to a physical occupancy layer. This information can then be used by decision-making routines (section 2.3).

The PPM contains percepts that are represented as parametric models with a spatial representation, thus representing entities without volume, *e.g.* robot poses or human joint configurations. The PPM complements the LVG's expressive power, allowing the system to represent entities without volume, such as robot poses, human joint configurations, navigation goals, etc.

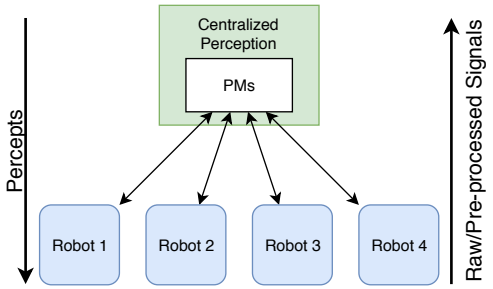
At a basic level, perception can be seen as a one-way pipeline: signals are fed into Perception Modules, resulting in percepts. MoDSeM aims to introduce non-linearity in this flow, allowing Perception Modules to access current and past data, including percepts obtained by other Perception Modules.

In order to achieve this, two measures are taken:

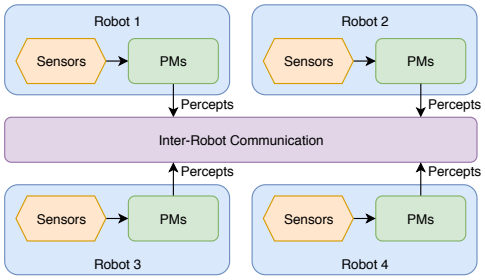
- Perception Modules are allowed to use the Semantic Map as input;
- Perception Modules are allowed to use **previous versions** of the Semantic Map as input.

In the first case, Perception Modules are allowed to depend on the Semantic Map and use it as a complement to any signal input they require. Indeed, some Perception Modules are expected to use solely the Semantic Map as input; *e.g.*, a traversability detector could estimate the traversability of the map using only occupancy, movable object and vegetation information.

Additionally, as in Fig. 3, allowing perception modules to use previous version of the SM as input implies that a history of SMs is kept during operation, which would quickly make




(a) Centralised perception: raw signals are sent to a centralised machine that returns percepts.




(b) Distributed perception: agents contain the necessary perception modules, and propagate percepts to other agents.

Figure 4: Centralized *vs* distributed perception.

the its storage infeasible. This can be mitigated in two ways:

- By storing successive differences between the maps as they are generated, as is done in video compression algorithms and source control systems;
- By intelligently choosing which snapshots of the map should be saved, using information-theoretic techniques, avoiding redundant information.

By fine-tuning these two approaches, it should be possible to establish a history in secondary memory with enough granularity to allow Perception Modules that depend on it to operate. For instance, the total system memory to be used by this mechanism can be set as a parameter, guiding the compression and data selection techniques in their optimization of the amount of information stored, as measured by information-theoretic metrics such as entropy.

Traditionally, multi-robot perception is achieved in one of two ways (Fig. 4): by propagating raw signals from each robot to a centralised perception server, which then replies with percepts; or by endowing each team member with perceptive abilities, as well as the ability to decide when percepts should be propagated among the team.

Centralised perception is preferable when the robots in the team contain little processing power and when the communication infrastructure is a non-issue, *i.e.* is always available and can support the necessary bandwidth. On the other hand, distributed perception is the appropriate technique when bandwidth is limited, when robots have heterogeneous needs and capabilities, and when each robot can be endowed with perception abilities that fit its needs.

A hybrid approach can be used with heterogeneous teams, when for instance one of the robots is significantly more computationally powerful than other team members, who can unload part of their perceptual load to this team mate. As can be easily inferred, this is the case of the SEMFIRE artificial perception architecture (section 2.1)

MoDSeM can be used to implement all of these perceptual modalities:

- the centralised approach is immediately supported, as it involves simply implementing all modules on a single machine;
- the distributed and hybrid approaches can also be achieved by communicating relevant layers of the Semantic Map among robots, using appropriate tools for multi-robot communication[1].

[1] `http://wiki.ros.org/multimaster_fkie`

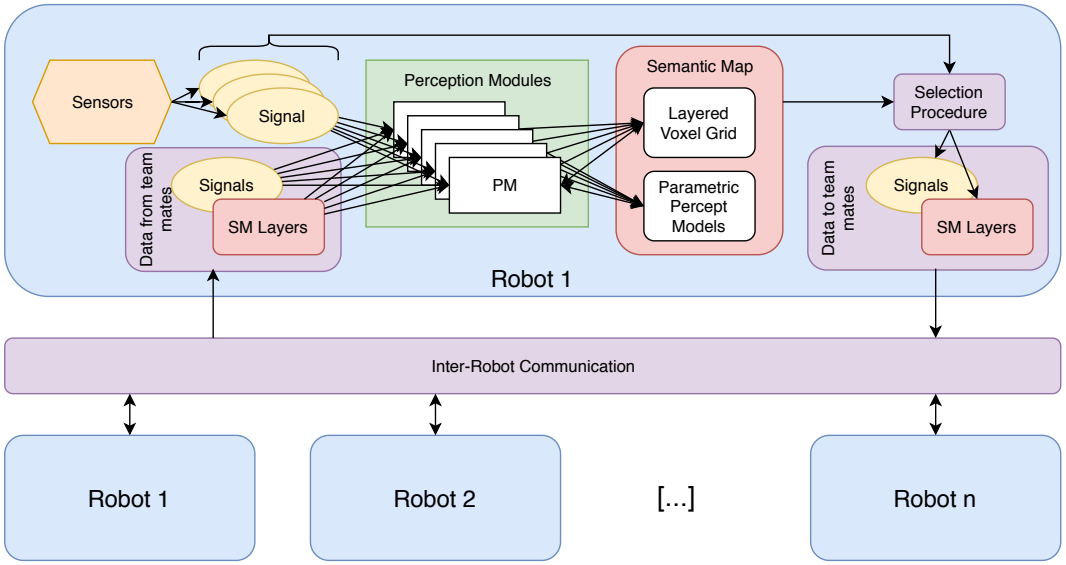



Figure 5: An overview of the SEMFIRE robot team operating with MoDSeM. Each team member can have its own sensors, perception modules and semantic map. These can be shared arbitrarily with the rest of the team, as needed. Each robot is also able to receive signals and Semantic Map layers from other robots, which are used as input by Perception Modules to achieve a unified Semantic Map.

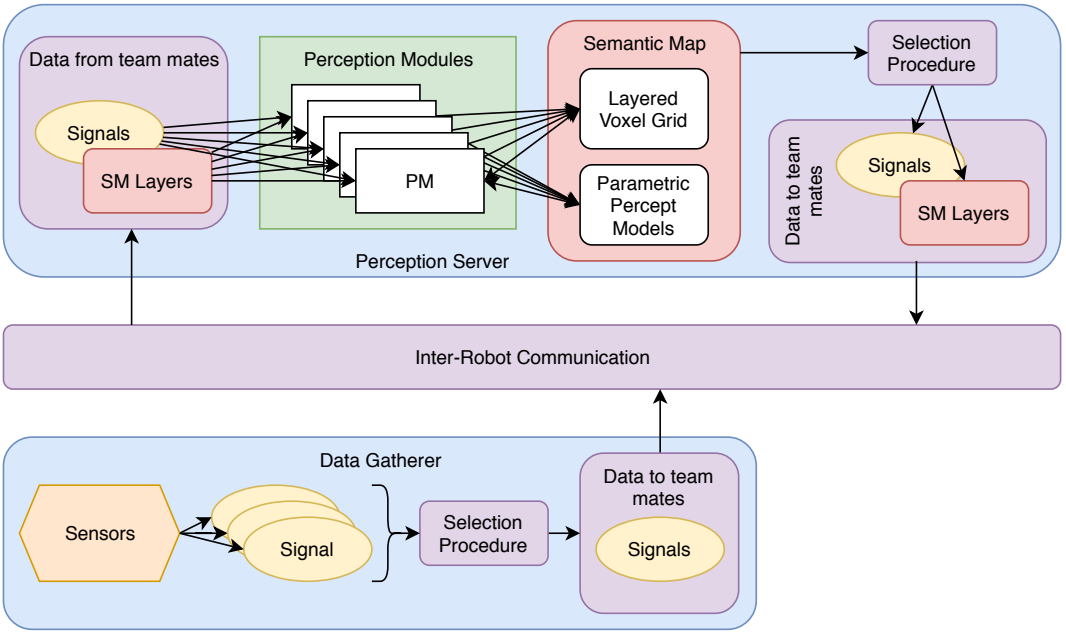



Figure 6: Different topologies (in addition to the hybrid topology of Fig. 5) for multi-robot perception using MoDSeM. Top: a perception server, which receives information and Semantic Map layers from the team and executes the most expensive perception modules. Bottom: a data gatherer agent, which observes the world with its sensors and sends data for processing in other agents.

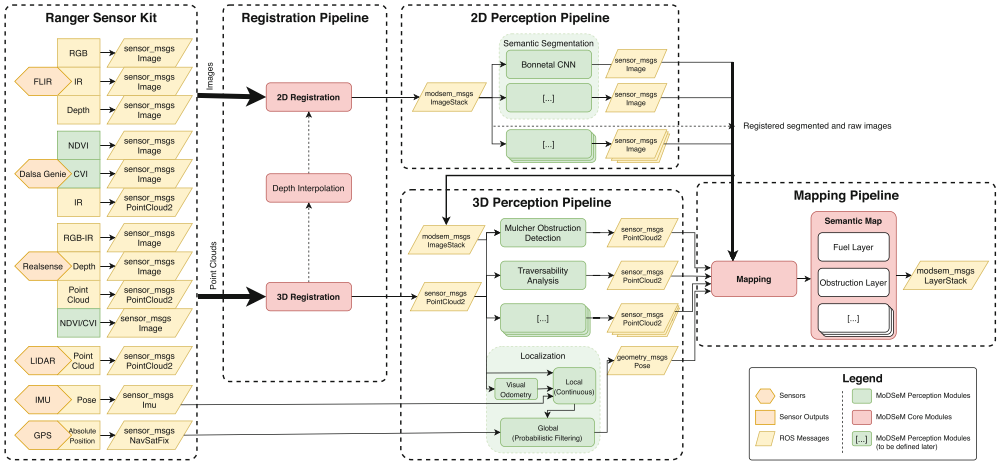



Figure 7: Artificial perception pipeline for the Ranger. Feedback connections are not represented for readability.

Fig. 5 shows an overview of MoDSeM implemented on the SEMFIRE perception architecture. In this case, the agent contains its own sensors, processing modules and semantic map. Its Perception Modules include specialized Perception Modules that are used to fuse information received from other agents, to achieve consensus. The agent also contains a selection procedure, which must be configured or endowed with some decision-making ability, which decides which information it should share with other agents, to be sent for inter-robot communication. Specific Perception Modules in each robot can then fuse these representations, achieving consensus and allowing all robots to plan with the same information. Other topologies can also be achieved with the framework, as in Fig. 6. Different topologies are achieved by mixing-and-matching the necessary components, such as Perception Modules and sensors and their configurations, to achieve different use cases.

Fig. 7 shows the SEMFIRE perception pipeline for the Ranger (which will be adapted to produce analogous pipelines for the Scouts) using MoDSeM perception and core modules as its foundation. Three sub-pipelines are considered in this approach:

**2D perception pipeline** – this involves all processing performed on images, in which semantic segmentation using the multispectral camera is of particular importance (see Fig. 8 for a preliminary example), since it plays the essential role in mid-range scene understanding that informs the main operational states of the Ranger;

**3D perception pipeline** – this involves all processing performed on 3D point clouds yielded by the LIDAR sensors and RGB-D cameras (including 3D-registered images) that support decision-making based on the perceived world to enact the desired landscaping tasks for each operational stage and also, in particular, safe 3D navigation (see section 2.3);

**Mapping pipeline** – this involves the final registration and mapping that takes place to update the MoDSeM semantic map (see Fig. 9 for a preliminary example).

### 2.2.1 Processing and Communication

Sensor data collected by the Ranger is processed by a layered perception architecture based on MoDSeM, allowing it to process data acquired by its own sensors and by the remainder of the team. The project challenges require a distributed architecture capable of processing

Figure 8: Example of semantic segmentation of combustible material using multispectral camera.

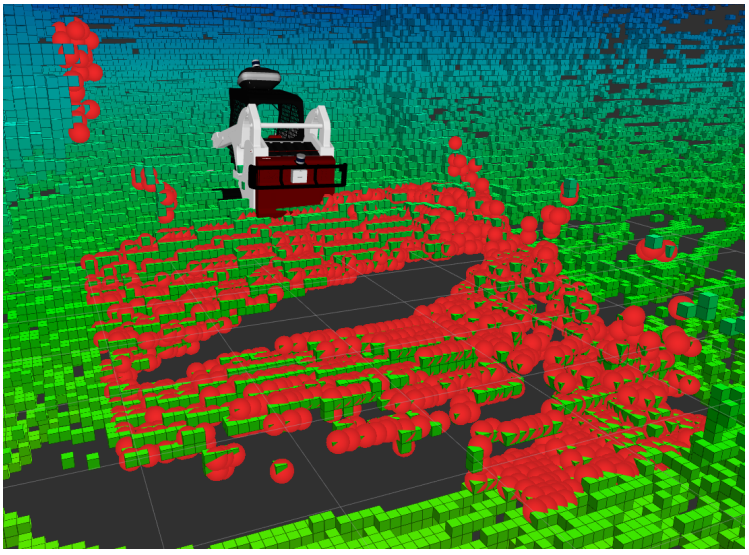


Figure 9: Example of MoDSeM layer mapping visualisation.

large amounts (up to tens of Gigabytes) of data per minute generated by the vision systems distributed across the Ranger and Scouts, making it impossible to store such amounts of data. The Ranger must accommodate the Signal Acquisition, Semantic Representation, Perception Modules, Perception-Action Coupling and the Decision Making modules. Therefore, data is processed at the edge with dedicated hardware accelerators that forward only relevant information to the central processing unit. Having this in mind, the processing and communication setup implemented in the Ranger includes:

- A Mini-ITX computer equipped with a Geforce RTX 2060, an Intel Core i7–8700 CPU and 16 GB of DDR4 RAM;
- A Tulipp FPGA+ARM platform including a Xilinx XCZU4EV TE0820 FPGA and a ARM Quad-core Cortex-A53 CPU with a Mali-400 GPU and 3 GB of DDR4 RAM;
- 5 AAEON Up Boards[2], one for each Intel RealSense device;
- One custom-made CAN bus controller, based on an ATMEGA ARM CPU and two MCP2551 CAN Bus transceivers[3]
- Two TP-LINK TL-SG108 Gigabit Ethernet Switches[4];
- One ASUS 4G-AC68U WiFi Router[5];

The Mini-ITX computer is the central processing unit of the Ranger, gathering relevant information from all components and running high-level algorithms such as planning and decision, using the ROS framework. It is equipped with a powerful Geforce RTX 2060 GPU, which provides the needed power to run heavy computational approaches.

The Xilinx XCZU4EV TE0820 FPGA is responsible for running the low-level drivers and low-level operational behaviors of the platform. It runs ROS and allows for transparent communication with sensors, and the higher-level CPU.

The five AAEON Up Boards are paired with each of the Intel Realsense D415 cameras, and are used to acquire and pre-process RGBD data in ROS as well, forwarding relevant data from these sensors to the main CPU.

The custom-made CAN bus controller allows to inject velocity commands to the underlying CAN system of the Bobcat platform, allowing to control the robot's locomotion, the hydraulic arm, and the operation of the mulcher, which will be later attached to the hydraulic arm. It consists of a custom board that integrates two MCP2551 CAN bus transceivers, which is installed between the machine's manual controls and the actuators, defining two independent CAN buses. This way, it becomes possible to make use of the CAN bus to control the machine.

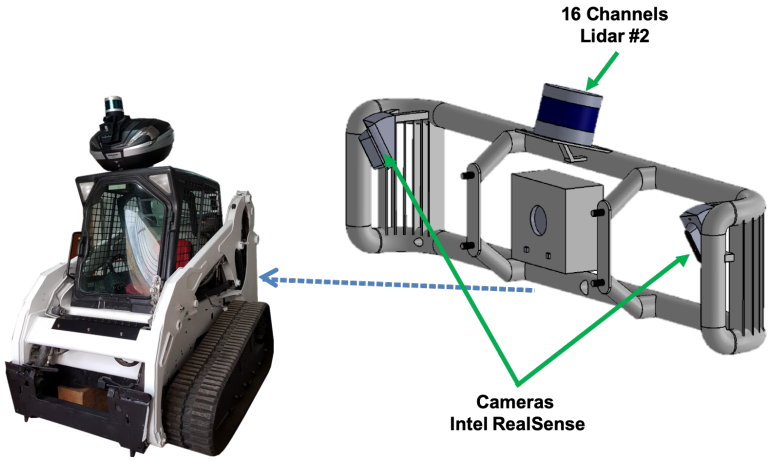
The TP-LINK TL-SG108 Gigabit Ethernet Switches interconnects the CPU, with the FPGA, the five AAEON UP Boards, the C16 lasers and the WiFi router, allowing for fast data acquisition and communication between components, enabling distribution of computation and remote access and control.

## 2.3 Core design of decision-making pipelines

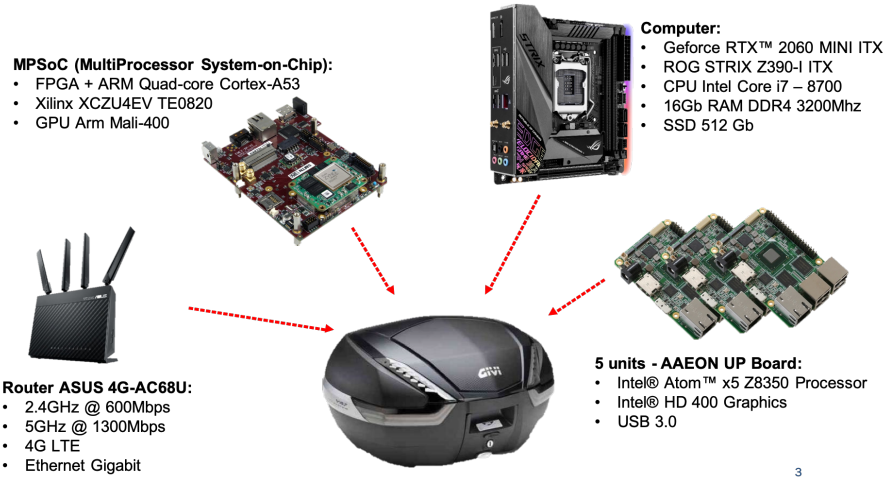
The decision-making modules represented in Fig. 1 use the outputs from the perception and perception-action coupling modules to produce the behaviours needed to enact the operational modes described in [10]. Of particular importance are the resulting Ranger's processing pipelines for overall decision-making pipeline for safe 3D navigation shown in Figs. 11 and 12, respectively. Analogous processing pipelines will be designed for the Scouts in follow-up work.

[2] `https://up-board.org/`
[3] `https://www.microchip.com/wwwproducts/en/MCP2551`
[4] `https://www.tp-link.com/pt/business-networking/unmanaged-switch/tl-sg108/`
[5] `https://www.asus.com/Networking/4G-AC68U/`


(a) Main sensory hub.




(b) Rear sensory hub.




(c) Processing hardware within main sensory hub.

Figure 10: *Ranger* sensor hardware framework.

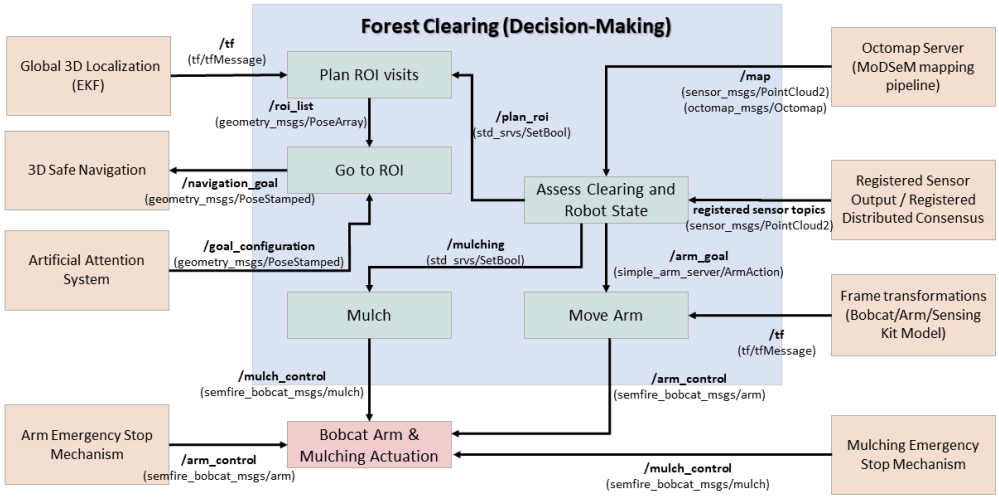



Figure 11: SEMFIRE decision-making pipeline for Ranger.

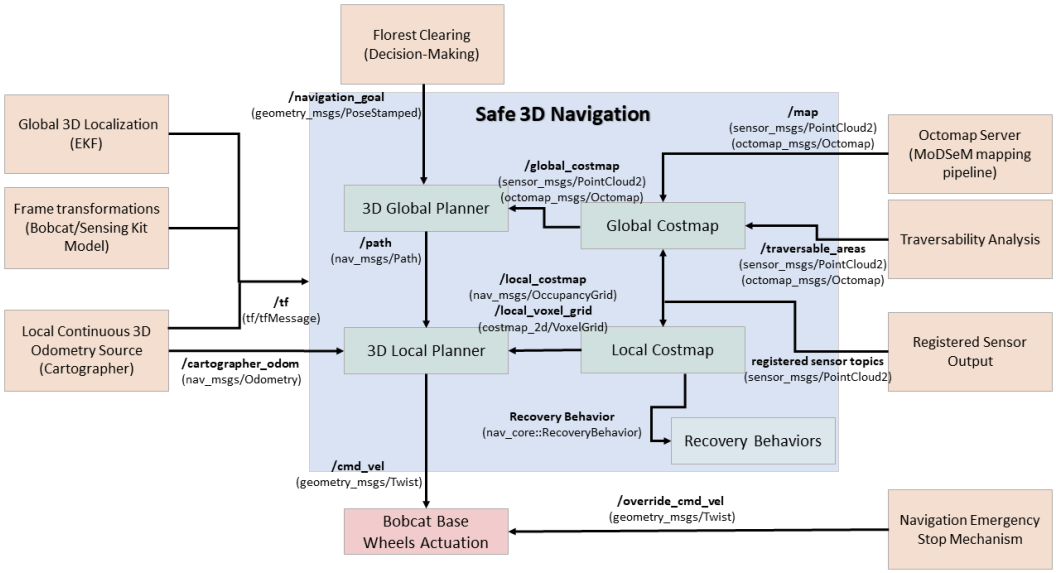
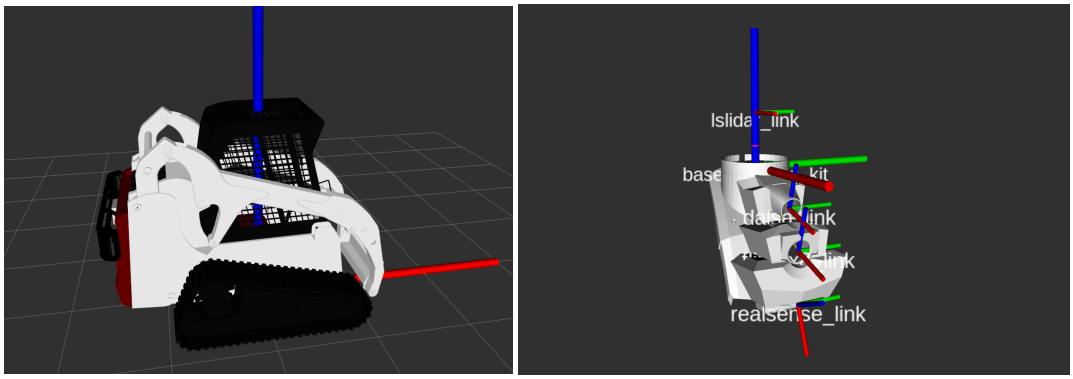



Figure 12: SEMFIRE safe 3D navigation pipeline for Ranger.




(a) Bobcat T190 base model.

(b) Main sensing kit and sensor frames.

Figure 13: Integration of the URDF models in `rviz`.

# 3 ROS Integration

System development and integration comprises the design, implementation and integration of hardware and software components, which in turn depend on the specifications and requirements arising from the first prototyping phase. In this section, we turn our attention to the integration of hardware and software component of the robotic platform using the Robot Operating System (ROS)[6].

ROS is a software framework that aims to ease the development of modular code targeting robots, and a very widespread tool in the field of robotics. ROS establishes a communication layer running on top of an ordinary host operating system, which allows for the exchange of messages between multiple ROS nodes. ROS nodes are programs which use the facilities provided by ROS to communicate and perform their tasks. ROS nodes operate independently and concurrently, and need not even be running on the same computer.

Communication is achieved through two main mechanisms: topics for asynchronous communication, and services for synchronous communication, both of which carry messages. ROS nodes can be implementations of all kinds of functions: data management, mathematical functions, or anything else that can be programmed in any of the languages supported by ROS. Hardware drivers are a prime example of just how powerful ROS's modularity is: for a given robot, it is possible to develop ROS nodes which subscribe to a set of standard topics to receive commands, and that implement the low-level code needed to relay those commands to the robot. ROS allows us to abstract away the hardware intricacies of the robot and to develop as if we were writing code that targeted a standardized robot. As such, ROS promotes code reutilization and has become a *de facto* standard in Robotics.

All software integration will be designed for Ubuntu 18.04 and ROS Melodic Morenia, with nodes written in C++ and Python. Software integration in environments incompatible with Ubuntu, such as microncontrollers, are designed using specific frameworks able to be integrated in ROS, using, for instance `rosserial`[7].

## 3.1 Robot Structure and Sensor Positioning

We have included a properly scaled Bobcat T190 3D model (see Figure 13a) through a Unified Robot Description Format (URDF) in the ROS visualization tool, `rviz`, to serve as a starting point for the definition of transformation frames (`tf`) of all the meaningful parts of our robot. After this, we integrated a URDF model of the main sensing kit, which was 3D printed previously to support the cameras and 3D LIDAR sensor on top of the

[6] `http://www.ros.org`
[7] `http://wiki.ros.org/rosserial`

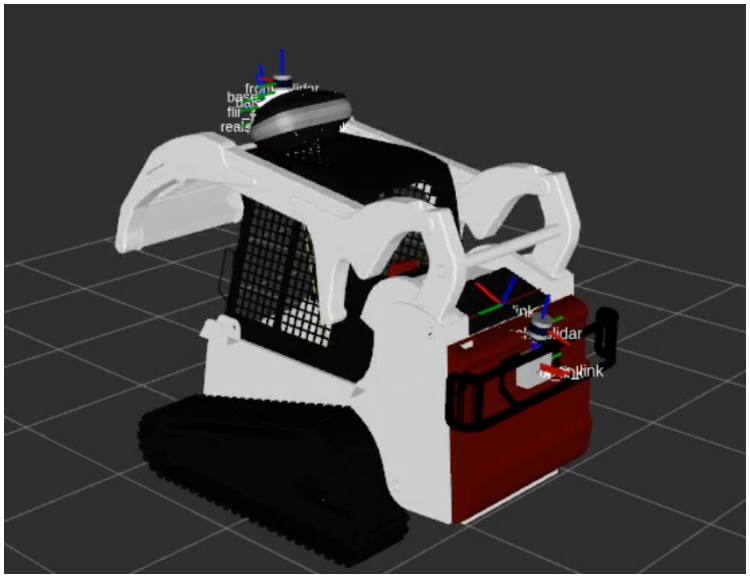


Figure 14: Ranger URDF model in `rviz`. Note the `tf` frames of the platform, and the revolute joint in the robot's hydraulic arm, which is raised in this illustration.

cabine of the robot, at the front. This allowed us to manually measure and define the relationship between the four distinct sensors in relationship to a common frame, i.e. the `base_sensing_kit` frame, as shown in Figure 13b.

Finally, we included in the overall Ranger model the missing parts: namely the processing unit case, the sensors and the rear sensory hub; and defined the robot's arm as a revolute joint, allowing us to raise and lower the arm as needed. This is shown in Figure 14.

## 3.2 Acquisition of Sensor Data

Sensor manufacturers make use of documented communication protocols for acquiring sensor data, and provide their own software frameworks for displaying the real-time results of that same acquisition. Fortunately, the Robotics community provides ready-to-use drivers for popular sensors in ROS, due to its widespread use and the need to access all data in a common framework. In the particular case of the Ranger UGV, we have benefited from the ROS drivers available for the Intel Realsense cameras[8], the Emlid Reach RS GPS-RTK device[9], the UM7 Inertial Measurement Unit[10], and the Leishen C16 LIDARs[11]. The latter was ported into ROS Melodic and Ubuntu 18.04 by us, and support was added for multiple LIDARs with different frame IDs. Moreover, the UWB system is an in-house developed solution provided by Ingeniarius, Ltd., coordinator of the SEMFIRE R&D project[12].

On the other hand, we could not find any readily available for the FLIR AX8 Thermal camera nor the Teledyne Dalsa Genie Nano C2420 Multispectral Camera. Therefore, we developed our own ROS drivers for these two cameras.

### 3.2.1 FLIR AX8

The FLIR AX8 Thermal camera connects to the remaining system via an Ethernet interface, having a fixed IP address configured in the same network as the main Ranger's CPU. The

[8] `https://github.com/IntelRealSense/realsense-ros`
[9] `https://github.com/enwaytech/reach_rs_ros_driver`
[10] `https://github.com/ros-drivers/um7`
[11] `https://github.com/tongsky723/lslidar_C16`
[12] The UWB system ROS driver is currently not publicly available.

manufacturer provides a streaming service using the Real Time Streaming Protocol (RTSP) on the camera's local IP address and port 554. Therefore, we developed a simple ROS driver, which establishes a connection to the RTSP server to acquire the $640 \times 480$ camera feed, and publish the retrieved images to the ROS network in a `sensor_msgs/Image` topic at a rate of 9 Hz.

The above simple driver provides a lightweight and stable solution for integration of the FLIR AX8 camera in ROS. However, we noticed that the streaming service provided by the manufacturer displays thermal images with an acquisition delay of around 2 seconds. Therefore, we developed an alternative ROS driver to provide images with a diminished delay. For this, we log in into the web server interface of the thermal camera via HTTP and the camera's IP. Then, we extract the consecutive JPG thermal camera images provided, which have a more reduced acquisition delay of around 0.25 seconds.

This is possible by starting the Mozilla Firefox browser in the background with a dedicated thread and extract the consecutive screenshots provided by the camera web server interface through web scraping tools, publishing them in ROS in a similar way to the simple ROS driver initially developed.

This solution, which is more complex, has the evident advantage of reducing the acquisition delay of the camera, which comes closer to a real-time acquisition at the cost of more CPU time, since it requires executing a browser instance.

We also make a `sensor_msgs/CameraInfo` topic available with the camera intrinsic parameters loaded from a `yaml` file, as well as launch files, which also publish `sensor_msgs/CompressedImage` topics, and provide customization via parameters, e.g. the IP address of the camera, the frame ID and the camera topic name. Both ROS drivers are distributed as open source[13].

### 3.2.2 Dalsa Genie Nano C2420

The Dalsa Genie Nano C2420 Multispectral camera is not a common sensor found in robots. In the SEMFIRE project, we make use of multispectral imagery to characterize, segment and classify relevant entities (e.g. vegetation) in the forestry environment.

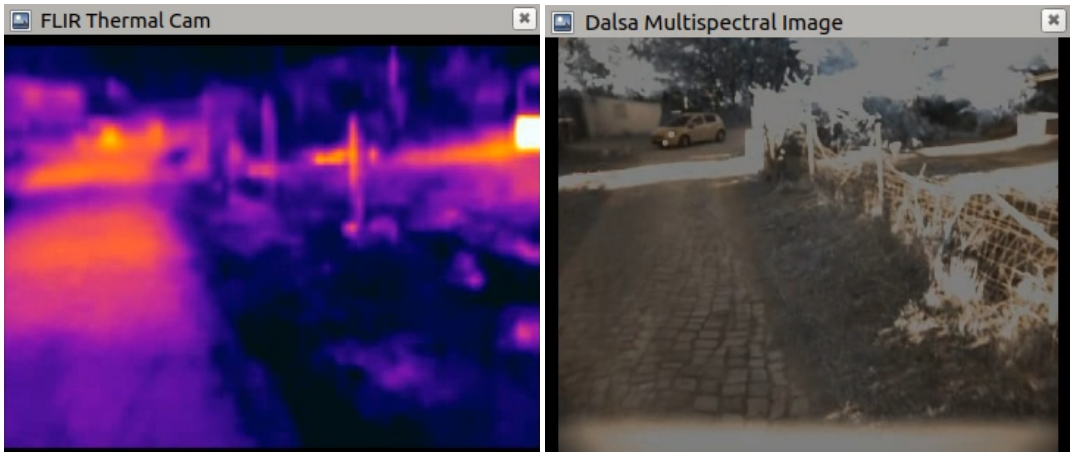
Similarly as above, this multispectral camera connects to the remaining system via an Ethernet interface, with a fixed IP address configured in the same network as the main Ranger's CPU. The manufacturer instructs users to acquire images using the GigE Vision SDK for Gigabit Ethernet cameras[14]. Therefore, we have developed a ROS driver, which makes use of the GevAPI library from the GigE Vision SDK, as well as the Generic Interface for Cameras – GenICam[15], allowing us to retrieve the camera feed into a ROS node, which links these libraries with `catkin` and `OpenCV` libraries.

The ROS driver starts by finding the camera IP on the network automatically, connect to it, retrieve its working parameters, receiving then the stream of images. Afterwards, the images are consecutively converted into a `sensor_msgs/Image` message which is published into a dedicated topic.

The camera is configured to stream with a $2464 \times 2056$ image resolution at 10 Hz, and the ROS driver publishes the image feed into the ROS network with the same specifications, as well as a downscaled 720p version of the image and an optional monochromatic version of both resolutions. The image published has the following three channels: NIR[16],G,R.

We have developed an additional node, which allows to publish useful alternative single-channel images from the manipulation of the original image. These include the Normalized Difference Vegetation Index (NDVI) image, the Chlorophyll Vegetation Index (CVI) image, the Triangular Vegetation Index (TVI) image, the Normalized Near infrared image, the

[13] `https://github.com/davidbsp/flir_ax8_simple_driver`
[14] `http://www.ab-soft.com/activegige.php`
[15] `https://www.emva.org/standards-technology/genicam/`
[16] NIR stands for the Near Infrared channel, commonly used in multispectral imaging.




(a) FLIR AX8 Thermal image. (b) Dalsa C2420 NIR,G,R Image.

Figure 15: Thermal and Multispectral ROS images of the same scene.

Normalized Green image, and finally the Normalized Red image. In Figure 16, we illustrate some of the obtained single-channel images, using a winter colormap for better visualization.

We also make the `sensor_msgs/CameraInfo` topic available with the camera intrinsic parameters loaded from a `yaml` file for both resolutions available, as well as a launch file, which publishes `sensor_msgs/CompressedImage` topics, and provide customization via parameters, e.g. the frame ID, the camera topic name, whether to publish the monochromatic topic, etc. We also provide an additional `yaml` parameter file to choose the alternative single-channel images to be published. The Dalsa Genie Nano C2420 ROS driver is distributed as open source[17].

## 3.3 Low-Level Control

The low-level controller of the Ranger intends to be a standard in leading the research into existing heavy-duty autonomous machines, with the integrated Society of Automotive Engineers (SAE) J1939 Controller Area Network (CAN) protocol. Even though this might seem like a hard constraint, the SAE J1939 has become the accepted industry standard for heavy-duty machines in agriculture, forestry and construction. The SEMFIRE team successfully conceived a microcontroller bridge, capable of interpreting and generating J1939 messages, thus allowing to seamlessly exchange messages with the central controller of the compact track loader. A low-level PID controller has been developed to continuously adjust the Ranger's locomotion based on the difference between the desired velocity, provided by the navigation layer, and the measured velocity, estimated by the visual odometry and the higher-level localization approach. This allows us to use standard teleoperation procedures to remotely control the Ranger, and develop autonomous outdoor navigation approaches in ROS, which output desired velocity commands to the Ranger's skid steering system via standard `geometry_msgs/Twist` messages.

The Ranger will be additionally endowed with a lift arm, with high level of rigidity, hose protection, and durability. The lift arm provides 2 DoF: lift (base rotation) and tilt (gripper rotation), with both breakout forces above 2 tons, capable of tearing out a tree root. Taking advantage of the CAN protocol, we are currently working towards the process of integration of the hydraulic arm of the UGV for motion planning and control using the `MoveIt!` motion planning framework for ROS[18].

[17] `https://github.com/davidbsp/dalsa_genie_nano_c2420`
[18] `http://moveit.ros.org`

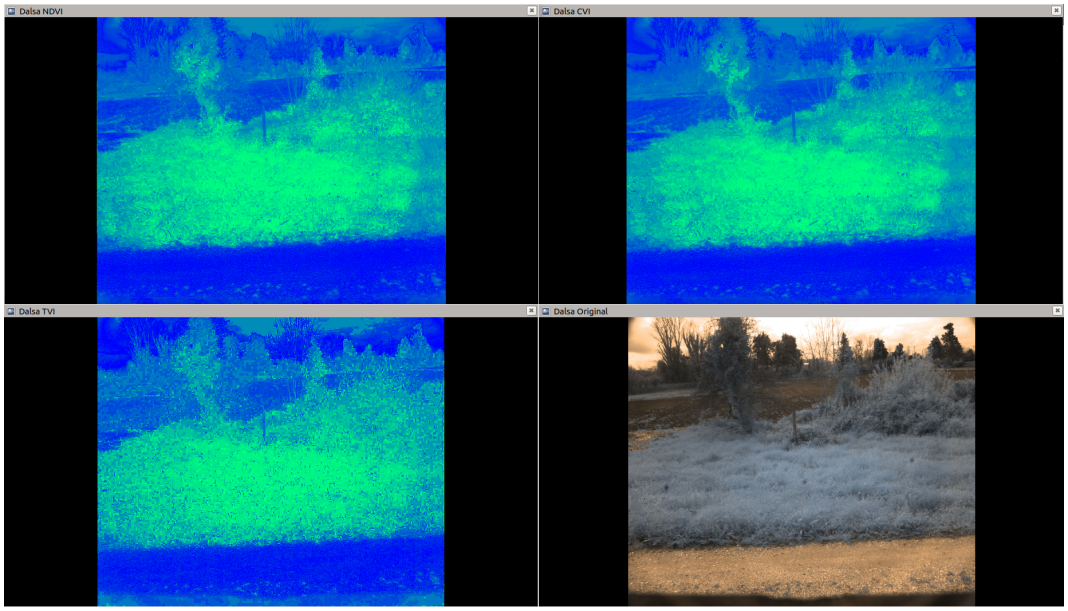



Figure 16: Dalsa Vegetation Index single-channel images illustrated in `rviz` using a winter color map (blue pixels at 0, green pixels at 255). NDVI image (top left), CVI image (top right), TVI image (bottom left) and original NIR,G,R three-channel image (bottom right).

.

### 3.4 Overall ROS System

The integration of the distinct components of the system in ROS allows for a decoupled and modular robot architecture, adhering to the black box approach, where the implementation of each software component is opaque, and only its inputs and outputs are important for the rest of the system.

One of the most important aspects about the integration in ROS is the definition of the coordinate frames and transforms[19] using the ROS `tf` API. Figure 17 illustrates the `tf` tree with the relationship of the several coordinate frames of the Ranger robot. We have defined a parent reference frame `bobcat_base` attached to the Ranger's mobile base, placed at the rotational center of the robot, and all other relevant frames (i.e. sensors and parts of the robot) have been defined with respect to this reference. Frame transforms are defined in the `URDF` description file of the platform (see Section 3.1), allowing for example to know the relationship of `sensor_msgs/PointCloud2` sensor data published by the front LSLIDAR in respect to the robot's `bobcat_base` frame. Furthermore, all of these transforms are static, meaning that the relationship between the sensors and robot parts does not change relatively to the base reference. The noteworthy exception is the `bobcat_base` to `arms_link` transform, which tracks the tilting arm up and down motion.

Another key aspect about the ROS integration is system bring up, i.e. the reliable initialization of all modules of the ROS system. This is done via a dedicated package for our robot, which includes a bring up `roslaunch XML` file. This launch file envelopes the launch files of all drivers, including sensors and low-level control; the robot `URDF` model, and tracks the Ranger's internal transforms via a `robot_state_publisher` node. Moreover, due to its 360° FOV and positioning on the Ranger, the Back LSLIDAR retrieves points from the robot's own body, which should be filtered out for most high-level perception algorithms in order not to be mistaken as obstacles. Therefore, we make use of a `pcl_ros` filter nodelet[20] to remove the robot's body data from the point cloud, by defining three intervals (in x, in y,

[19] `http://wiki.ros.org/tf/Overview/Transformations`.
[20] `http://docs.ros.org/melodic/api/pcl_ros/html/classpcl__ros_1_1CropBox.html`

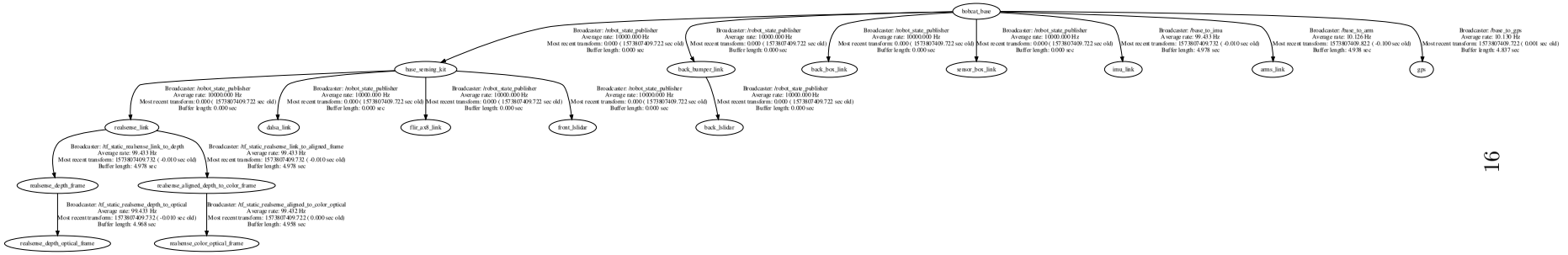



Figure 17: `tf` tree of the frames defined for the Ranger ROS system.

Figure 18: Ranger's remotely controlled experiment in the field. Full video available at: `https://www.youtube.com/watch?v=Jo99beTU2J8`.

and in z) in 3D space, which correspond to a 3D box around the robot. This filtered point cloud is then published in a designated topic.

During the development and testing of robotic algorithms with the Ranger, e.g. localization, teleoperation, navigation, multi-sensor registration, semantic segmentation and mapping; dedicated launch files are created for each approach, and started along with the bring up launch file, enabling the separation of the common ROS platform base system and the higher level robotic behaviors.

# 4 System Tests

In this section, we briefly overview some internal tests and work in progress towards endowing the Ranger heady-duty UGV with the capability of correctly perceiving and operating in the forestry environment.

## 4.1 Basic Perception Capabilities

Initial tests focused on **remotely controlling** the machine to test the **teleoperation** ability and to collect early `rosbag`[21] datasets with the sensing equipment included in the Ranger, for analysis and offline development of artificial perception algorithms. Figure 18 illustrates these successful tests, and a video of the robot operation is provided in the caption.

Besides extracting the intrinsic calibration parameters[22] of the thermal and mutispectral cameras (see Section 3.2), we have been conducting work on **multi-sensor registration** to augment the perception of the robot by joining complementary data given by different sensor modalities, and improve the original transforms between sensors obtained through manual measurements and prior information on sensor positioning (see Section 3.1). In Figure 19, we illustrate initial registration results between the multispectral camera and the front 3D LIDAR, which lets the camera image be augmented with depth information, and we are currently investigating depth completion methods.

[21] `http://wiki.ros.org/rosbag`
[22] `http://wiki.ros.org/camera_calibration`

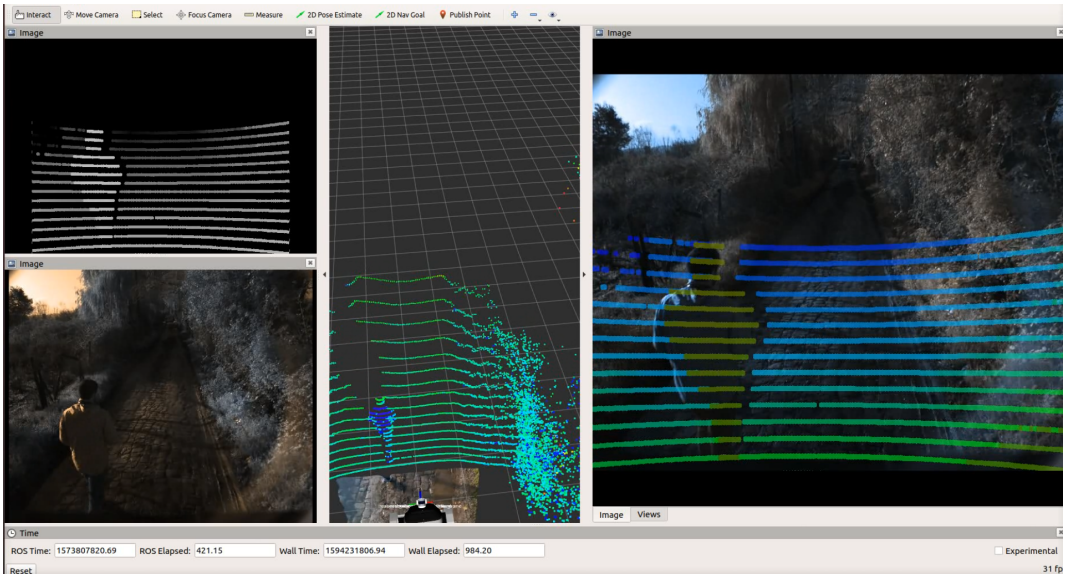



Figure 19: Work in progress on the registration between the Teledyne Dalsa Genie Nano C2420 multispectral camera and the front Leishen C16 3D LIDAR. Top left image: artificial depth image within the camera's FOV built from 3D LIDAR data. Bottom left image: Original multispectral camera image. Center: Point cloud extracted from the front 3D LIDAR (and Intel Realsense colored point cloud on the floor in front of the robot). Right image: Overlaid LIDAR depth data on top of the multispectral image. The ranger was safely teleoperated during this test, thus the proximity of a person on the left side of the UGV.

Multimodal 6D **localization** $(x, y, z, \varphi, \theta, \psi)$ and **mapping** are another key feature of the robot for supporting operations in forestry environments. In Figure 20, we illustrate tests on generating pose estimates from a sensor fusion hierarchical localization approach, which combines LIDARs, IMU, Depth Cameras and GPS data. The localization estimation is then passed on to the Octomap library[23] to build a detailed 3D map of the environment. Accurate localization is a mandatory requirement for field operations, including robotic navigation or large-scale mapping and perception, which should be robust to highly dynamic and unstructured outdoor environments, encompassing hard challenges for perception such as GPS dropouts, undistinctive visual features in forestry scenes and high perturbation in motion due to rough terrain traversability. This is a work in progress that we are constantly aiming to improve. For instance, we intend to make the approach robust to missing inputs over time, and we also plan to intensively test more parametric combinations of inputs, as well as additional inputs, e.g. the integration of Ultra Wideband (UWB) technology as a source for global localization in situations where GPS dropouts do not allow the system to maintain a georeferenced system.

The **decision-making** modules use the outputs from the perception and perception-action coupling modules to produce the behaviours needed to enact the Ranger's operational modes. Of particular importance are the resulting Ranger's processing pipelines for overall decision-making for safe 3D navigation shown in Figures 11 and 12, respectively. Analogous processing pipelines will be designed for the Scouts in follow-up work.

The Ranger platform has a high degree of mobility, being based on one of the most powerful, comfortable, versatile and recognised compact track loaders available in the market. The Ranger is tuned for soft, sandy, wet and muddy conditions. It presents a climb capacity and traversal capacity of approximately 35°, which covers a wide range of slopes available in

[23] `http://wiki.ros.org/octomap_mapping`

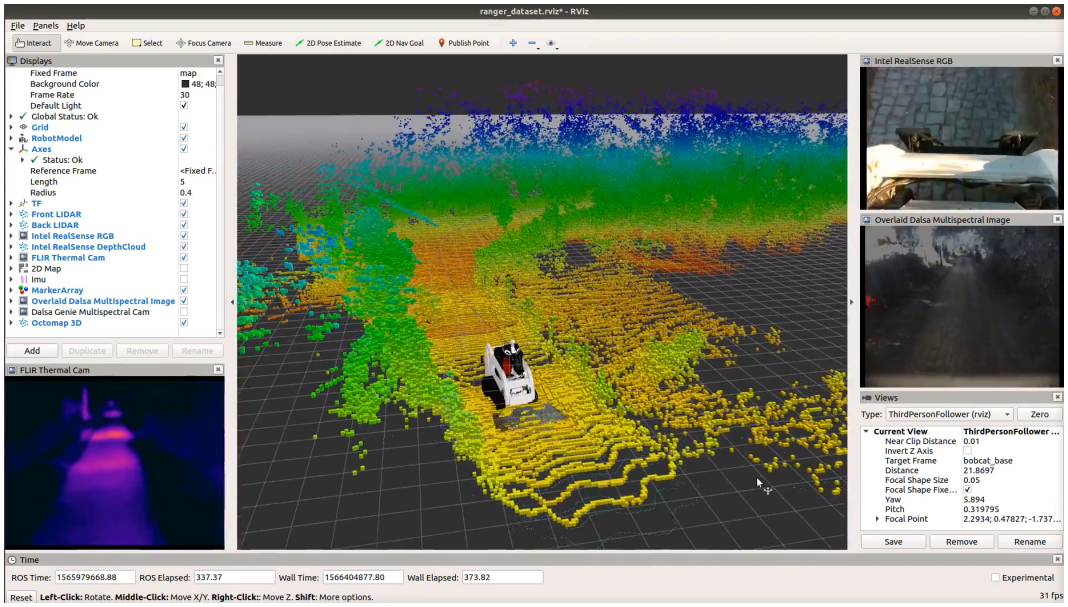



Figure 20: 6D localization and 3D map in a challenging outdoor environment. Full video available at: `https://www.youtube.com/watch?v=Gucs9otr2Ns`.

forestry scenarios. Efforts are currently being made for the implementation of the Ranger **navigation architecture**, which is heavily based on [16] (cf. Figure 12). We are extending the 2D navigation and path planning approach to 3D, by testing and improving the baseline approach in the SEMFIRE simulator, as illustrated in Figure 21.

Regarding **3D path planning**, we have been working on extracting the 3D gradient of obstacles in relation to the robot, so as to distinguish ramps and slopes from obstacles, such as trees. We plan to incorporate the gradient of the local occupancy points around the robot as an observation source to the navigation framework, and we are then able to threshold along the gradient (we take into account that the robot does not climb slopes with more than 45 degrees of inclination) to understand whether we have an obstacle ahead of the robot or a ramp or slope [14]. This allows for the local costmap around the robot to distinguish between obstacles and pathways. The higher-level software, which inspects the costmaps to reach a navigation goal, and sends adequate velocity commands to the mobile robot base[24] is currently being adapted. When the work in simulation is stable, we intend to test the 3D planning approach in the Ranger with the existing low-level track actuation controller (see Section 3.3).

In parallel, we have also been exploring **traversability** mapping[25] for mobile rough terrain **navigation**. We have generated a probabilistic robot-centric traversability map that estimates the terrain features for mobile robots in the form of a grid-based elevation map[26] including upper and lower confidence bounds. The data in the map is updated based on the uncertainty of the incremental motion as the robot moves, incorporating the state estimation and a noise model of the distance sensor [5], i.e. the 3D LIDAR sensors. Figure 22, on the right side, illustrates the local traversability map acquired by the Ranger robot during an outdoor experimental trial [20].

[24] `http://wiki.ros.org/move_base`
[25] `https://github.com/leggedrobotics/traversability_estimation`
[26] `https://github.com/ANYbotics/elevation_mapping`

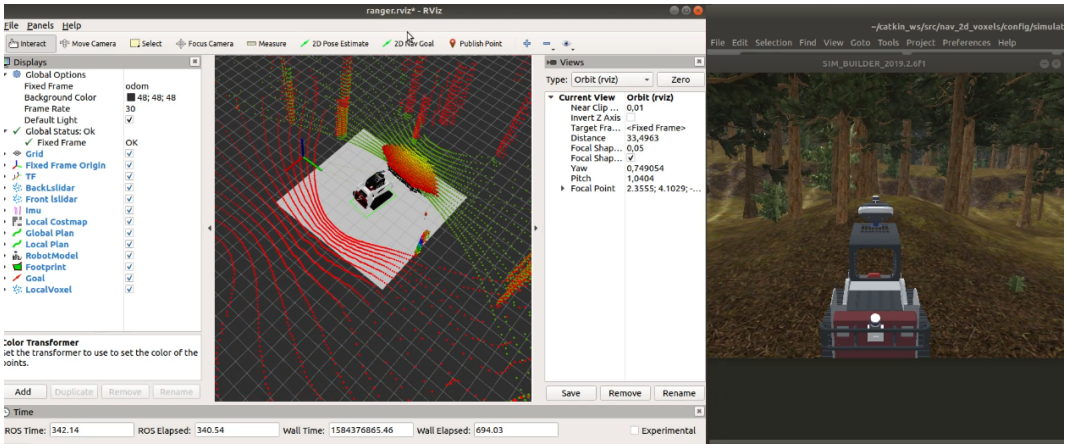



Figure 21: Ranger Navigation simulation tests.

Figure 22: Combined Traversability Analysis and Image Segmentation.

## 4.2 Semantic Segmentation

Extensive work has also been conducted on **semantic segmentation** for the identification of undesired vegetation that represent flammable material that exponentially increases the potential spread of wildfires, as a solution for the perception problem for forestry robots [1]. The approach consists of a preexisting deep learning-based semantic scene understanding method adapted for the use of multispectral imaging. Preliminary results of applying our approach in real-world conditions in an outdoor scenario are presented in Figure 8 and 22, on the left side. We then conducted a benchmarking of different CNN architectures in forestry environments.

### 4.2.1 Benchmarking CNN Architectures for Semantic Segmentation for Landscaping with Forestry Robotics

For our purposes, we are interested in segmenting the image pixel-wise according to the 6 classes listed in Table 1 (see also Fig. 23). The image inputs for our solution currently consist

Table 1: Semantic classes and respective color coding.

| **Classes** | **Colors** |
|---|---|
| Background | Black |
| Live flammable material (aka Fuel) | Red |
| Canopies | Green |
| Trunks | Brown |
| Humans | Yellow |
| Animal | Purple |

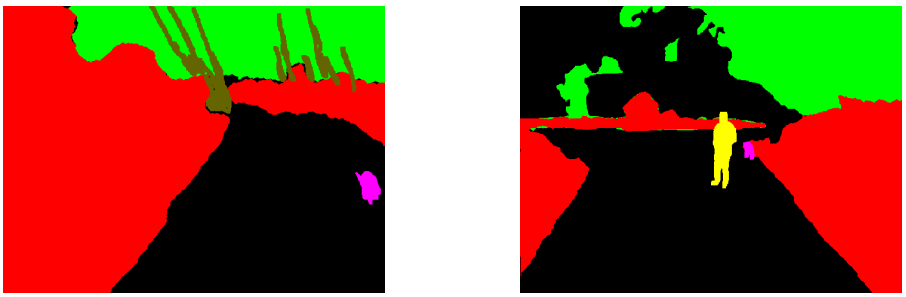


Figure 23: Two examples of ground truth labeling of multispectral images using the classes listed in Table 1.

of three image streams conveyed by the multispectral camera, namely NIR (Near Infrared), Red and Green (NGR) at a rate of 10 frames per second (fps), a frame-rate which imposes a requirement on the execution times for inference to be under 100ms. Another particularly important requirement for our application is to minimize false positive classifications in the Fuel class, a conservative approach which ensures the preservation and safety of local flora and fauna, respectively. To this end, we evaluated the performance of a set of state-of-the-art neural network based-solutions together with a custom-made solution based on an adaptation of an alternative decoder-encoder architecture.

The set of state-of-the-art, off-the-shelf architectures that we tested was comprised by *MobileNetV2* [9] and *ResNet50* [8]). These were chosen because *MobilenetV2* is reportedly a fast and accurate network while *ResNet50* is slower but more robust.

In addition to these two architectures, the encoder-decoder architecture proposed by [25], consisting of the *Adapnet++* encoder and the *eASPP* (efficient Atrous Spatial Pyramid Pooling) decoder, was adapted and implemented. More specifically, *Adapnet++* is a modification of current *ResNet50* with the addition of 2 residual units to the architecture, while *eASPP* is an adaptation of *ASPP* (Atrous Spatial Pyramid Pooling) developed by DeepLab [3], and was created to reduce the amount of parameters while maintaining the accuracy in the results. A simplified representation of this architecture can seen in the Figure 24. Since *Adapnet++* is based on *ResNet*, it was important to compare how they both performed for our application.

Very importantly for real-time operation, the three architectures meet our 10 fps (100ms) requirement (see Table 5). To achieve the greatest flexibility in model selection and/or implementation and the best possible performance in training, deployment and inference, we used an off-the-shelf, stable, easy to use, robotics-friendly tool with a modular codebase called Bonnetal [19], which implements semantic segmentation using CNNs designed to be easily integrated with the ROS (Robotics Operating System) framework while still taking full advantage of available computational resources via user-friendly configuration. This tool allows developers to easily develop new research approaches while avoiding the effort of

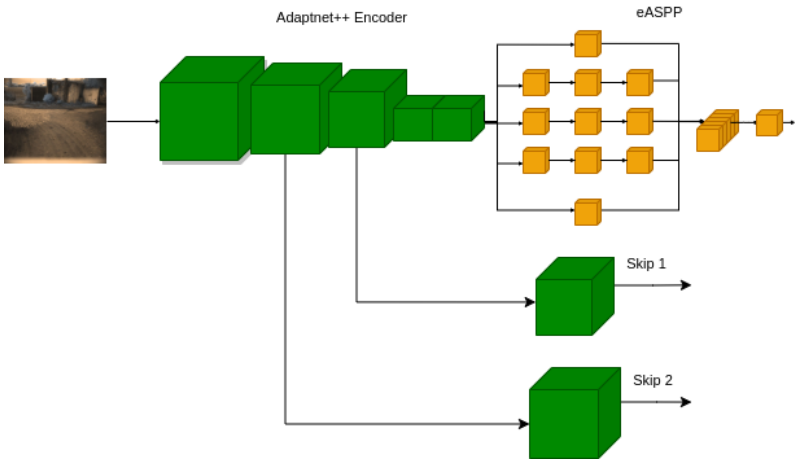



Figure 24: Simplified representation of the Adapnet++ – eASPP architecture where the green blocks all go through a convolution, ReLU and Batch Normalization and the yellow blocks are 3x3 convolutions with 3, 6, and 12 dilation rate [25].

re-implementing them from scratch or modifying the available code until it becomes at least marginally usable for the research purpose. Fundamentally, Bonnetal mixes and matches any encoder and decoder sets (including backbones such as MobileNet and ResNet) using any of a set of backends (e.g. the PyTorch framework) for the build. Furthermore, it allows changes to hyper-parameters, dataset and architectures effortlessly due to its configuration file and structure.

To complement Bonnetal, for evaluation/benchmarking of the possible solutions we used a specialized tool named Weights and Biases [2]. This tool logs each desired metric to identify the ideal backbone. In our approach, considering our requirements, the metrics used were Jaccard Index (or $IoU$), *Recall* and the *F1 score*:

$$JI/IoU = \frac{TP}{TP + FN + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

$$F1\ Score = \frac{2TP}{2TP + FN + FP},$$

where:

- $TP$ = *True Positive*, correspond to correctly classified pixels,
- $FP$ = *False Positive*, correspond to pixels wrongly classified,
- $FN$ = *False Negative*, correspond to pixels that wrongly indicate that the class is absent.

Table 2: Jaccard Index comparison between different architectures using transfer learning techniques.

| Test # | Encoder | Decoder | Pre-Weights | Class Weights | mIoU |
|---|---|---|---|---|---|
| #1a | MobileNetv2 | ASPP | None | None | 0.314 |
| #1b | MobileNetv2 | ASPP | ✓ | None | 0.405 |
| #2a | ResNet50 | ASPP | None | None | 0.419 |
| #2b | ResNet50 | ASPP | None | ✓ | 0.544 |
| #2c | ResNet50 | ASPP | Decoder only | None | 0.562 |
| #2d | ResNet50 | ASPP | ✓ | ✓ | **0.729** |
| #3a | Adaptnet++ | eASPP | None | None | 0.388 |
| #3b | Adaptnet++ | eASPP | None | ✓ | 0.425 |
| #3c | Adaptnet++ | eASPP | ✓ | ✓ | 0.660 |

Figure 25: Example output for model consisting of *MobileNetv2* backbone, *ASPP* progressive decoder and fine tuning trained on Bonnetal with no pre-weights. Ground truth image is shown on the left and corresponding prediction on the right.

These metrics were selected because each provide insightful information on an encoder's quality. Specifically, $IoU$ provides the most comprehensive test as it calculates the positive result in comparison to all the possible values, while *Recall* analyzes the quality of the positive values. For example, if the value for People is near 1, it means it rarely misidentifies people with any other class. This would be ideal as we would like to avoid humans to be identified as fuel. Lastly, the $F1$ score calculates as the Jaccard Index but it attributes a larger weight to true positives therefore potentially providing a more coherent outcome.

In the description that follows, training was performed using a GeForce RTX 2070 GPU and Core i7 8th Gen CPU on the SEMFIRE dataset, which is composed by 500 labeled NGR images (NIR, Green and Red Channels) according to the classes listed in Table 1.

The first set of tests conducted were to identify if any kind of transfer learning would be advantageous for an image that it is not RGB. Since our dataset contains NGR images, it was not known if the pre-trained models would transfer features well. Therefore, a few tests were conducted utilizing the same hyper-parameters – architecture configurations and quantitative results for all tests are summarized in Table 2.

In Test #1, the *MobileNetv2* backbone was used with the *ASPP progressive* decoder and no frozen layers, both with and without pre-weights (Figs. 26 and 25, respectively). The ImageNet pre-trained model, a dense dataset with 1.2 million RGB images and 60 different classes [4], was used. As can be observed, using no pre-weights results in substantial misidentifications, most likely due to the dataset's size.

Empirical testing shown on Table 2 shows transfer learning techniques do indeed improve the $IoU$ value in all neural networks analyzed. Even though the pre weights image are based on the usual RGB channels (i.e. a different modality from NRG), it did improve the quality

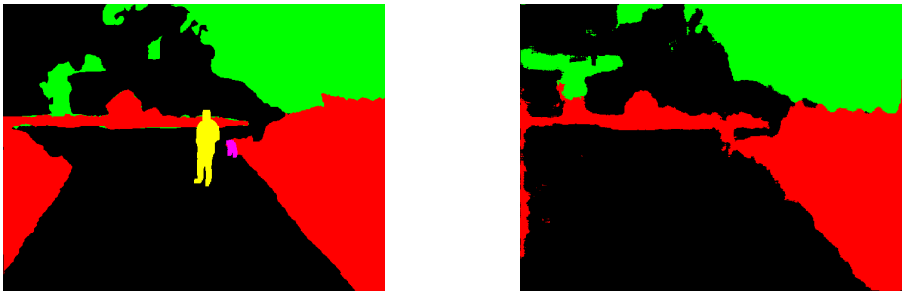


Figure 26: Example output for model consisting of *MobileNetv2* backbone, *ASPP* progressive decoder and fine tuning trained on Bonnetal using Imagenet pre-weights. Ground truth image is shown on the left and corresponding prediction on the right.

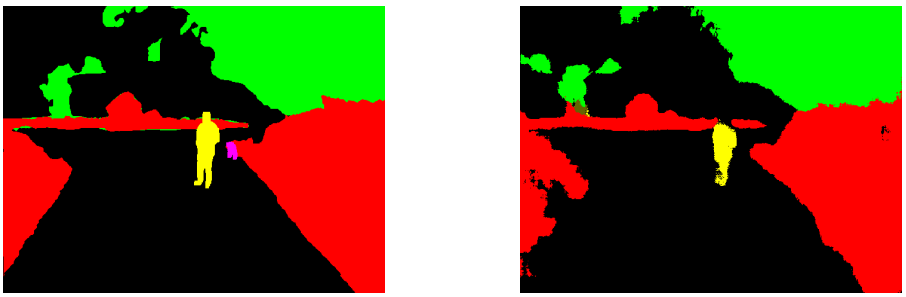


Figure 27: Example output for model consisting of *ResNet50* backbone, *ASPP* progressive decoder and fine tuning trained on Bonnetal using Imagenet pre-weights with frozen encoder layers. Ground truth image is shown on the left and corresponding prediction on the right.

of training. Moreover, both *Adapnet++* and *Resnet50* improve classification using special label weighting for each class instead of the default value **1**, a technique used to help with class imbalance when there are classes with few instances [26].

Therefore, we assumed that transfer learning was beneficial in all cases, and proceeded to test the effect of freezing layers on transfer learning for ResNet. For Test #2, *ResNet50* was used as a decoder while keeping the same encoder (*ASPP progressive*). Fig. 27 shows the outcome for this architecture using ImageNet pre-weights, 500 epochs and frozen encoder layers. Conversely, Fig. 28 shows results for the same architecture using 700 epochs and special label weighting for each class instead of the default value **1**, a technique used to help with class imbalance when there are classes with few instances [26]. Using this setup resulted in improved results for *ResNet50* in rarely occurring cases such as Animals. It also showed that false positives were minimized when comparing to *MobileNetv2*, possibly because *ResNet50* is a more robust encoder.

In conclusion, Test #2 shows that traditional transfer learning (i.e. freezing encoder layers when the features are similar) does not produce optimal quality. This can be due the different modalities used for transfer learning and training the final model (i.e. RGB vs NGR), but more tests are needed to confirm this hypothesis. Overall, transfer learning techniques such as weight initialization with class imbalance and pre-trained weights have shown to improve the overall quality of the semantic segmentation classification.

Finally, Test #3 served to compare with the *Adapnet++–eASPP* architecture if encoder layers are not frozen (i.e. best-case scenario), showing slightly lower but still similar performance to *Resnet50*, which is understandable as the backbone of the former is an adaptation of the latter.

Figure 28: Example output for model consisting of *ResNet50* backbone, *ASPP* progressive decoder and fine tuning trained on Bonnetal using Imagenet pre-weights for the whole model. Ground truth image is shown on the left and corresponding prediction on the right.

Figure 29: Example output for model consisting of *Adapnett++* backbone, *eASPP* progressive decoder and fine tuning trained on Bonnetal using Imagenet pre-weights for the whole model. Ground truth image is shown on the left and corresponding prediction on the right.

#### 4.2.2 Hyper-parameter optimization

The second set of experiments was designed to determine the optimal set of hyper-parameter values for each architecture. Given that the first set of experiments showed that transfer learning does indeed improve results, the next set of tests already included pre-trained models and weight initialization for class imbalance. The hyper-parameters were modified using the Weights and Biases tool [2], which uniformly changes each input between a fixed range to find the optimal validation loss, which was chosen because it is inversely proportional to the evaluation metrics – in other words, a lower loss yields higher *IoU*, *Recall* and *F-1* Score values. Each configuration was trained for 500 epochs and for each backbone, tests were conducted at least 20 times. The hyperparameters in question are listed and explained in Table 3.

Fig. 30 and 31 show the results from the sweep made by Weights and Biases for *Adapnet++*, *ResNet50* and *MobileNetv2*. . These show that a specific configuration may yield a negligible *IoU* value for Fuel using the *MobileNetv2* backbone while its maximum is near the values found for ResNet50 and *Adapnet++*. This same effect can be better seen in *IoU* for Humans, where *ResNet50* has a lower variance between best and worst results due to the robustness of the system holding up regardless of the hyperparameters.

Fig. 30 shows comparison results using the *IoU* metric, demonstrating that changes in hyper-parameters significantly affect the performance of each architecture. For instance, when considering the *IoU* for the Fuel class, *MobileNetV2* produces results close to 0 or near the best results of *ResNet50*. The same phenomenon happens for *Adapnet++* in the segmentation of humans and animals. This demonstrates the importance of choosing the right hyper-parameters and finding which has the greatest influence on segmentation quality. As can be seen in the plot, segmentation quality does not differ much in a class with a great number of samples such as Fuel. The highest values for each architecture are similar, with

Table 3: Configurable training parameters and their purposes.

| **Parameters** | **Purpose** |
|---|---|
| Max LR | Stochastic Gradient Descent (SGD) maximum learning rate. It is an adjustment in the weights of our network with respect to the loss gradient descent |
| Min LR | Warmup initial learning rate |
| Up Epochs | Decides the number of epochs used for warmup |
| Down Epochs | Decides number of epochs used for warmdown |
| Max/Min Momentum | SGD momentum max/min when LR is max/min respectively |
| Final Decay | Learning rate decay per epoch from Min LR |
| W Decay | Weight decay value for L2 regularization |
| Batch Size | Number of training examples utilized in one iteration |
| Backbone/Decoder Dropout | Number of layers dropped out while training for backbone/decoder |
| Backbone/Decoder Batch Normalization Decay | Decay on batch normalization to reduce noise on the backbone/decoder |

*ResNet50* achieving the best results and *MobileNetV2* the worst performance. However, in classes with smaller number of samples, such as Humans and Animals, results differ significantly. The Humans class has similar values between all three, but *ResNet50* widens the gap to the other two architectures after 200 epochs, reaching a 0.1 gap between itself and *Adapnet++*. Conversely, the Animals class exhibits a more unpredictable outcome. In fact, by 400 epochs, only *Adapnet++* is able to detect animals at all in every experiment. This effect can be due to the other networks needing more epochs to start recognizing that particular class more effectively. It also shows that the new residual units from *Adapnet++* have a positive effect when identifying classes with low number of instances.

In summary, even though *ResNet50* performs best on Humans and Fuel, *Adapnet++* achieves a better overall performance, since it is able recognize a specific class for which all others fail.

The *Recall* metric, as mentioned previously, is the most appropriate metric for evaluating the performance of all three architectures in what concerns our requirement of avoiding as many false positives for Fuel as possible. For parameter optimization, it follows the same pattern as *IoU* regarding the range of detection between the worst and best models trained for each architecture. This aligns with our expectations as both depend on the same values of true positives and false negatives.

Fig. 31 shows impressive *Recall* values for Fuel, with *Adapnet++* and *ResNet50* achieving the same results at 0.9 while *MobileNetV2* obtaining 0.85. This means that Fuel is rarely misclassified. However, to ensure the best performance, it is necessary to obtain high *Recall* values for Humans and Animals so they are not misidentified as fuel. *Recall* for Humans shows great promise as it is above 0.9 for all architectures. *MobileNetV2* marginally attained the best values at 0.995, while *ResNet50* and *Adapnet++* achieve slightly lower results at 0.979 and 0.934 respectively. However, as with the *IoU* metric, only *Adapnet++* is able to recognize animals up to 400 epochs at 0.865.

Fig. 32 shows qualitative results for all architectures for a representative example including instantiations of all classes except Animals. As seen in the figure, all three results

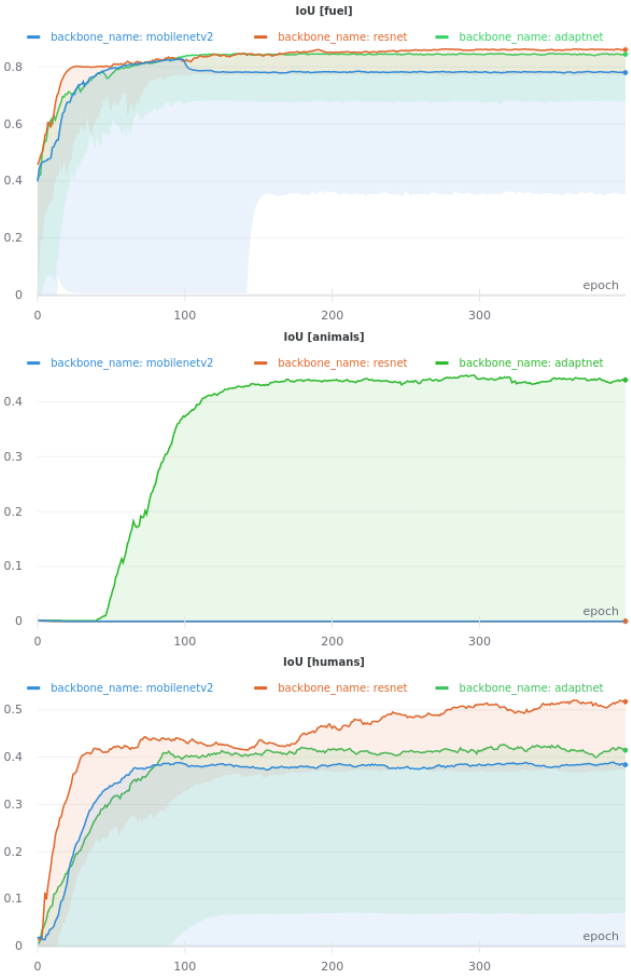



Figure 30: Comparison plots of IoU *vs* Epoch for the *Adapnet++*, *ResNet50* and *MobileNet* encoders, for the Fuel, Animals and Humans classes resulting from the sweep made using Weights and Biases (respectively, from top to bottom; green for *Adapnet++*, yellow for *ResNet50* and blue for *MobileNetv2*). Lighter, faded colors represent all values depending on the chosen hyperparameters, while darker, contrasting colors correspond to the best results.

visually seem to be near the expected ground truth with *Adapnet++* probably being the best match from visual inspection.

Lastly, Table 4 shows the overall result for each encoder with all metrics, including the $F1score$. As shown previously, while *ResNet50* has a slight advantage over *Adapnet++* for specific Classes such as Fuel and Humans, the latter achieves the best mean results as it is the only architecture that recognizes animals at all.

Even though *Recall* values are high for Humans, Fig. 32 shows the architectures tend to overestimate the size of humans and fuel – in fact, most categorized the post in the background as fuel, which would clearly represent a safety hazard if the high-level modules of the mulching robot were to take that classification as accurate.

Moreover, the architectures many times mislabel, in a non-cluster-like fashion, most background features as either Human, Animal or Fuel, as can be seen in Fig. 33. Most likely, this happens because that section of the image is the side of a truck with a logo and there are no similar scenarios in the dataset. A possible solution for these issues is to include


Figure 31: Comparison plots of Recall *vs* Epoch for the *Adapnet++*, *ResNet50* and *MobileNet* encoders, for the Fuel, Animals and Humans classes resulting from the sweep made using Weights and Biases (respectively, from top to bottom). Coloring scheme follows same convention as with Fig. 30.

other modalities such as depth. In fact, it could also improve accuracy and lower amount of false positives.

### 4.2.3 Complexity and inference execution time analysis

Finally, all architectures were compared in terms of number of parameters used and inference execution times – results are summarized in Table 5. As expected, *MobileNetv2* is by far the fastest running architecture matched by the considerably lower number of parameters it uses, while the other two architectures exhibit much closer figures, with *Adapnet++–eASPP* being lighter and faster than *ResNet50*.

Following the work on semantic segmentation for the identification of relevant entities categories in the forestry environment, we have focused on Depth Completion to fulfill the goal of localizing flammable material in 3D to support the decision-making components of the Ranger and plan for potential actuation, i.e., mulching forestry debris to consolidate landscape maintenance.

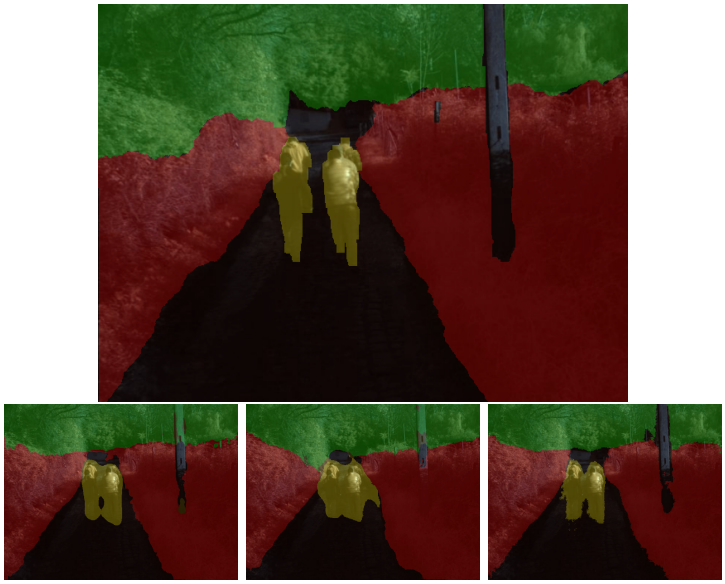


Figure 32: Qualitative Results for *MobileNetv2* (left), *ResNet50* (middle) and *Adapnet++* (right) in comparison to the ground truth (top).

Table 4: Results for best mean for evaluation metrics.

| **Encoder** | **Decoder** | **mIoU** | **mRecall** | **mF1-Score** |
|---|---|---|---|---|
| MobileNetv2 | ASPP | 0.5352 | 0.7672 | 0.6328 |
| ResNet50 | ASPP | 0.5326 | 0.7283 | 0.6339 |
| Adaptnet++ | eASPP | **0.6475** | **0.8967** | **0.7677** |


Figure 33: Example of ground truth (left) and respective prediction (right) made by *MobileNetv2* backbone and ASPP progressive decoder with fine tuning showing mislabeling of features on the lettering on the side of the truck as being member of other than the expected Background class.

Table 5: Complexity and inference execution time for $640 \times 480$ input images.

| Encoder | Decoder | # Parameters | Inf. Time (ms) |
|---|---|---|---|
| MobileNetv2 | ASPP | **2,154,862** | **11.6** |
| ResNet50 | ASPP | 49, 652, 118 | 53 |
| Adapnet++ | eASPP | 32, 345, 870 | 40.9 |

#### 4.2.4 Depth Completion and 3D Fuel Localization

Depth map acquisition follows specific steps to ensure proper appraisal from the camera's perspective:

1. Intrinsic Calibration of the Multispectral Camera and the 3D LiDAR;
2. Acquisition of world coordinate poses of the sensors;
3. Extrinsic Calibration of the Multispectral Camera and the 3D LiDAR to find relative poses, i.e. translation and rotation transforms;
4. Projection of LiDAR point cloud into camera image space;
5. Sparse depth map registration.

The camera and LiDAR were calibrated to extract the intrinsic and extrinsic values. For calibration, a checkboard provided the information for the camera's intrinsic and aruco markers transformed it into 3D, providing the camera's world coordinate. Together with the aruco marker, these were necessary to find correspondence between both sensors and find the extrinsic values using Kabsch algorithm [11].

We use the calibrated LiDAR and camera static transformation matrices to project the LiDAR point clouds into pixel coordinates. To minimize computational costs, we employ a filtered LiDAR point cloud which has roughly the same dimensions as the convex hull formed by the camera field-of-view. An example image is illustrated in Figure 34.

As mentioned above, projecting a LiDAR point cloud onto the image plane results in a sparse depth map. In fact, while 720p images have 620k pixels, in that same frame, the LiDAR projection only covers around 15k of these pixels (roughly 3%). Therefore, depth completion is needed to obtain a dense depth map, which is necessary to provide enough coverage of the operational scene in order to properly assist decision making for forestry landscaping.

Methods based on multi modal neural networks have shown promising results, by utilizing the sparse depth image and the equivalent RGB image to extrapolate the remaining pixels according to objects and its boundaries. These methods have shown to be robust since they fuse different sensor modalities. However, it is a computationally costly option and we require lighter options guaranteeing real-time performance. Therefore, we have chosen IP-Basic [13], which uses traditional computer vision techniques to estimate missing points. While it does not achieve the best results in the current KITTI benchmark, with a score of 1288.46 Root Mean Squared Error (RMSE), it has the fastest runtime. An example of it is illustrated in Figure 34. As can be observed, it does improve the depth density greatly by filling gaps without information. Albeit, due to the LiDAR's specific tilt causing large gaps in the image's top half, the resulting extrapolation lacks precision where no point cloud is found and it repeats the last known values nearby.

Being able to identify live flammable material and localize it in 3D is of utmost importance to the landscape maintenance task. We developed a technique in which the pipeline

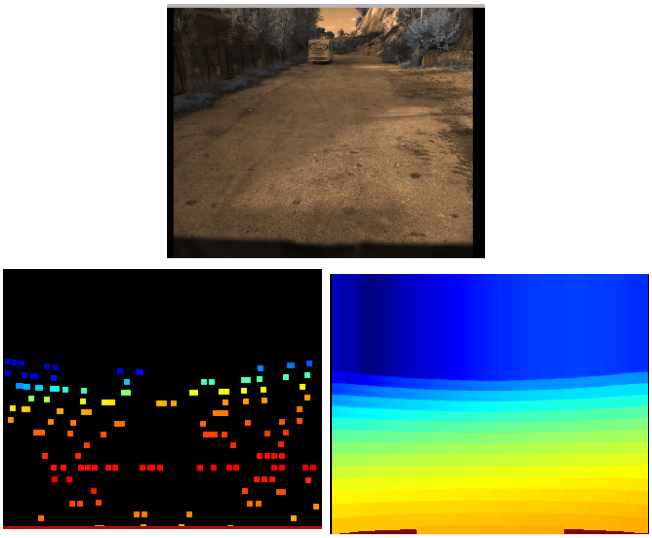


Figure 34: Example of depth map and its completion. Top image is the NGR sample; Middle image is the colored depth map extracted from the LiDAR (pixels artificially dilated for easier visualization); and bottom image is the depth completion output using the IP Basic technique.

receives each semantic segmented image and computes the centroid for each fuel cluster. Each centroid is then projected from pixel coordinate (u,v), into world coordinates (x,y,z). However, to avoid any outlier depth points and improve the estimation, we also take into account all non zero neighbor values, calculate the mean and only consider the bottom two-thirds of the image where the depth completion has the most information for extrapolation. Both can be seen in Figure 35. Each cluster has its own pose with respect to the center of the Ranger. This information is then crucially taken as input by the higher-level decision modules, which are beyond the scope of the work reported in this paper.

### 4.2.5 First Pilot Testing Results

Experiments occurred for a 2-hour duration on the Ranger platform. Due to the complex logistics of operating the heavy-duty machine in a forestry scenario, the location chosen was a parking lot with some elements of natural woodland in Coimbra, Portugal. While it was not ideal for our project, this demonstration allowed to examine our perception pipeline and realize the future improvements we require for further testing.

A set of 500 images were extracted from the acquired dataset and labeled to evaluate the semantic segmentation performance by applying the evaluation metrics *IoU* and *F-1 score*. In contrast, the fuel localization and depth map were analyzed qualitatively since there were no means to determine ground truth to be compared to the LiDAR point clouds.

Both LiDAR and camera sensors acquire data at a frequency of 10Hz in the Ranger and all perception modules in this framework are able to generate output at the same frequency, i.e 10Hz, with the onboard computer which has an Intel Core i7-8000 CPU and a RTX 2060 Super GPU.

**Semantic Segmentation** The semantic segmentation results can be seen in Table 6. It shows the targeted values (Key Performance Indicators) for the project and what we were able to achieve at this stage. Even though the performance has not reached the targeted values for both fuel (i.e live flammable material) and the other classes, the fuel classification attained is less than 15% away from the goal values and therefore, we believe that these are

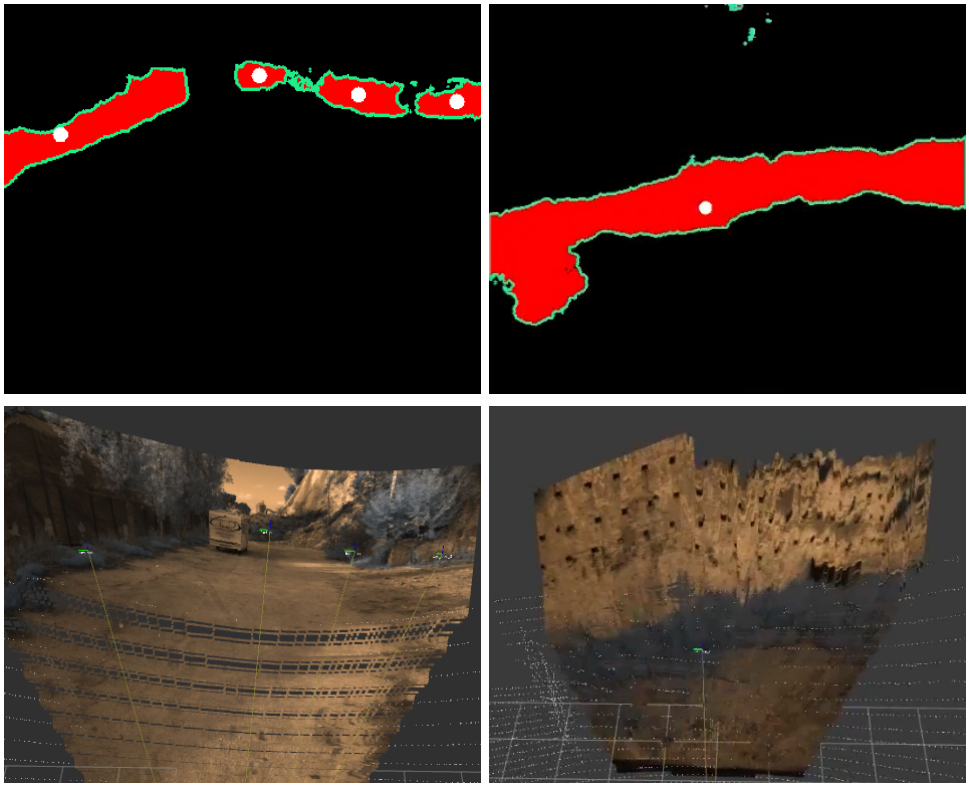


Figure 35: Results of center of mass projected to world coordinates on a colored point cloud. Top image shows each red 'fuel' cluster contour in green and its center of mass, in its pixel coordinate, as a white dot. The bottom image is the equivalent position in the world coordinate represented by a ROS TF on a colored point cloud.

promising results. These scores are also justified by our current conservative design which aims at lowering false positives on fuel. Indeed, Figure 36 indicate how the fuel is sometimes mislabeled as canopies on the top right part of the image. Ideally, it is better to overestimate trunks and canopies while also underestimating fuel to ensure the flora's and robot's safety.

The chosen environment does not contain enough elements that sufficiently relate to a natural woodland scenario and the neural network had a harder time identifying the differences between canopies and fuel. A scenario that is similar to the expected in a Portuguese forest would inform us more precisely of the results' quality.

Moreover, the training dataset also suffers from lack of diversity in its class instances. This can be seen in Figure 36 as the semantic segmentation prediction has issues identifying trunks and humans. Hence, increasing dataset diversity should also improve performance. For instance, canopies that have a large amount of examples in our current dataset had an *F-1 score* of **0.757** while trunks, which are only present in 1/30th of the dataset, scored **0.018**.

Lastly, luminosity also contributed to lower scores. Cameras are sensitive to light variance, which affect the number of quality features a system can obtain. This demonstrates the need for the diversification of sources of sensory information.

**Fuel Localization** As mentioned previously, depth map generation and fuel localization was assessed at a purely qualitative level, due to the current lack of ground truth for a quantitative study.

Visual inspection of the results shown in Figure 35 show that the centroid for each fuel cluster is created correctly. Its position on the world coordinates match the pixel coordinate

Table 6: Measure Performance Evaluation Metrics

| Metrics | Targeted | Achieved |
|---|---|---|
| **F-1 Score [Fuel]** | 0.850 | 0.706 |
| **F-1 Score [Others]** | 0.750 | 0.493 |
| **IoU [ Fuel])** | | 0.546 |
| **mIoU [Others]** | | 0.419 |

in x,y position. However, its z position depends on the depth map completion estimation, which may have Euclidean distance errors of up to 3 meters. While it seems to estimate correctly where there are LiDAR points, its extrapolation in the occluded parts are not often well estimated. Figure 34 shows an example of this phenomenon which happens because the current depth estimation has no other information to derive from and it only extrapolates from the available points which all lie at the bottom of the frame. In the following section, we report on our exploration of new methods that include other sensory information, namely NGR/RGB images and LiDAR points as inputs for depth completion, thus taking advantage of all known information from each modality over the whole span of the image.

Lastly, the semantic segmentation detection reshapes between frames due to camera motion resulting for example in the separation of one bush into two, making the fuel localization module detect two centroids instead of one. Accordingly, the fuel positioning was not smooth and it oscillates with each frame[27], which should be corrected by the 3D metric-semantic mapping process.

#### 4.2.6 Depth Completion Using LiDAR- and Vision-Based Data Fusion

As we move on towards the final trials, we are working on an improved version of depth completion that fuses information from LiDAR and cameras using convolutional neural network.

This solution is being trained using synthetic images generated from photo-realistic forestry environment simulations and preliminary results are already demonstrating the significantly improved performance of this approach as compared to IP Basic presented in the previous sections – see Fig. 37.

# 5 Conclusion

In this deliverable, we put forth an overview of the perception architecture for the SEMFIRE team of robots (section 2.1), and proposed a core design of the perception pipeline (section 2.2) and decision-making module functionality (section 2.3).

Additionally, we built on these foundations, improving and further developing the Ranger's perception and decision-making architecture, also investigating and studying deployment strategies to efficiently use the computational resources available. We have described the sensory extensions made to the base platform, its integration in the Robot Operating System (ROS) middleware, and reported on internal system tests concerning the diverse capabilities provided by the Artificial Perception system (section 4).

Following the extension of the project granted due to the worldwide COVID-19 pandemic, work up to the final trials will focus on the base-arm coordination for effective and swift clearing of forestry debris, increasing the maturity of the high-level algorithms for 6D localization, 3D semantic-metric mapping, 3D navigation in rough terrains, multi-sensor registration, and semantic segmentation (e.g. using more accurate depth completion techniques

[27] An illustrative video of the results is available on `https://youtu.be/VnxUgL_WHQs`

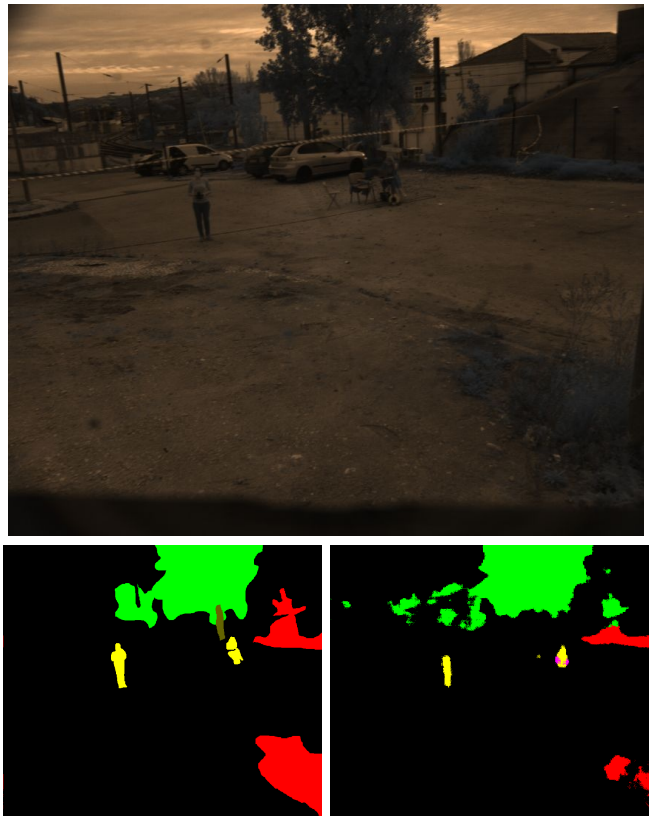


Figure 36: Semantic segmentation results in our experimental setup. Original multispectral image on top; Ground truth on the bottom left, prediction on the bottom right. Colors are yellow for humans, red for fuel, green for canopies, brown for trunks and purple for animals.

to also include a second input to the system such as RGB to increase extrapolation's precision) with the Ranger, and implementing the analogous Scout architectures, and putting in place the robot team communication and collaboration infrastructures.

# References


[1] ME Andrada, J De Castro Cardoso Ferreira, D Portugal, and M Couceiro. Testing different cnn architectures for semantic segmentation for landscaping with forestry robotics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), Workshop on Perception, Planning and Mobility in Forestry Robotics (WPPMFR 2020)*. IEEE, 2020.

[2] Lukas Biewald. Experiment tracking with weights and biases, 2020. URL `https://www.wandb.com/`. Software available from wandb.com.

[3] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016. URL `http://arxiv.org/abs/1606.00915`.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

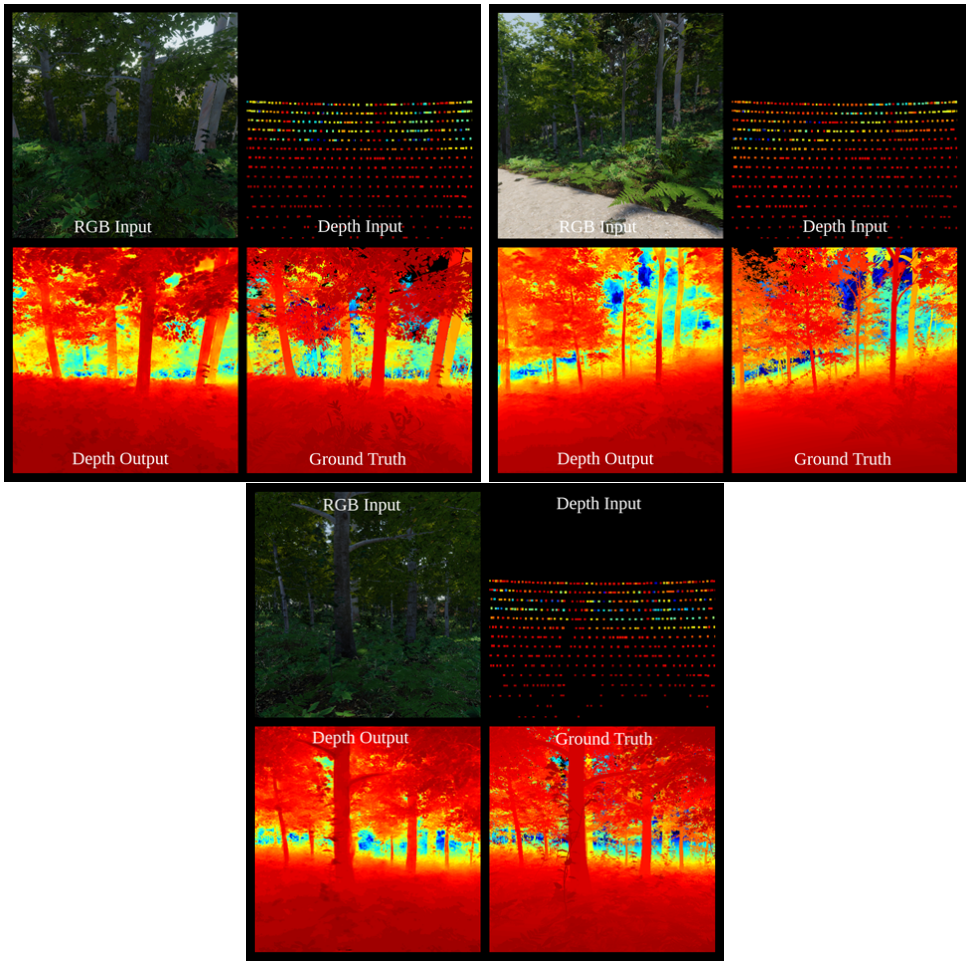



Figure 37: Preliminary results for depth completion using convolutional neural networks by fusing camera and LiDAR information (cf. results for IP Basic in Fig. 34).

[5] Péter Fankhauser, Michael Bloesch, and Marco Hutter. Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4):3019–3026, 2018.

[6] F. A. Gougeon, P. H. Kourtz, and M. Strome. Preliminary research on robotic vision in a regenerating forest environment. In *Proc. Int. Symp. Intelligent Robotics Systems*, volume 94, pages 11–15, 1994. URL `http://cfs.nrcan.gc.ca/publications?id=4582`.

[7] M. K. Habib and Y. Baudoin. Robot-Assisted Risky Intervention, Search, Rescue and Environmental Surveillance. *International Journal of Advanced Robotic Systems*, 7(1), 2010.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL `http://arxiv.org/abs/1512.03385`.

[9] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. URL `http://arxiv.org/abs/1704.04861`.

[10] Institute of Systems and Robotics – University of Coimbra. Deliverable 1.2 – Functional and Technical Specification. Technical report, SEMFIRE P2020 R&D Project, 2018.

[11] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5), 1976.

[12] Alonzo Kelly, Anthony Stentz, Omead Amidi, Mike Bode, David Bradley, Antonio Diaz-Calderon, Mike Happold, Herman Herman, Robert Mandelbaum, Tom Pilarski, Pete Rander, Scott Thayer, Nick Vallidis, and Randy Warner. Toward Reliable Off Road Autonomous Vehicles Operating in Challenging Environments. *The International Journal of Robotics Research*, 25(5–6):449–483, January 2006. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364906065543. URL `http://ijr.sagepub.com/content/25/5-6/449`.

[13] Jason Ku, Ali Harakeh, and Steven L Waslander. In defense of classical image processing: Fast depth completion on the cpu. In *2018 15th Conference on Computer and Robot Vision (CRV)*, pages 16–22. IEEE, 2018.

[14] D Lourenço, J F Ferreira, and D Portugal. 3d local planning for a forestry ugv based on terrain gradient and mechanical effort. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), Workshop on Perception, Planning and Mobility in Forestry Robotics (WPPMFR 2020)*. IEEE, 2020.

[15] Stephanie Lowry and Michael J Milford. Supervised and unsupervised linear learning techniques for visual place recognition in changing environments. *IEEE Transactions on Robotics*, 32(3):600–613, 2016.

[16] Eitan Marder-Eppstein, Eric Berger, Tully Foote, Brian Gerkey, and Kurt Konolige. The office marathon: Robust navigation in an indoor office environment. In *2010 IEEE international conference on robotics and automation*, pages 300–307. IEEE, 2010.

[17] Gonçalo S. Martins, J. F. Ferreira, David Portugal, and Micael S. Couceiro. MoDSeM: Modular Framework for Distributed Semantic Mapping. In *2nd UK-RAS Robotics and Autonomous Systems Conference – 'Embedded Intelligence: Enabling & Supporting RAS Technologies'*, Loughborough, UK, January 2019. URL `Trabalhos/Conf/MoDSeM_UKRAS19_v3.pdf`.

[18] Gonçalo S. Martins, J. F. Ferreira, David Portugal, and Micael S. Couceiro. MoDSeM: Towards Semantic Mapping with Distributed Robots. In *20th Towards Autonomous Robotic Systems Conference*, London, UK, July 2019. Centre for Advanced Robotics, Queen Mary University of London. URL `Trabalhos/Conf/TAROS2019.pdf`.

[19] Andres Milioto and Cyrill Stachniss. Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using cnns. *CoRR*, abs/1802.08960, 2018. URL `http://arxiv.org/abs/1802.08960`.

[20] Ahmad Kamal Nasir, André G Araújo, and Micael S Couceiro. Localization and navigation assessment of a heavy-duty field robot. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2020), Workshop on Perception, Planning and Mobility in Forestry Robotics (WPPMFR 2020)*. IEEE, 2020.

[21] S. Panzieri, F. Pascucci, and G. Ulivi. An outdoor navigation system using GPS and inertial platform. *IEEE/ASME transactions on Mechatronics*, 7(2):134–142, 2002.

[22] R. Siegwart, P. Lamon, T. Estier, M. Lauria, and R. Piguet. Innovative design for wheeled locomotion in rough terrain. *Robotics and Autonomous Systems*, (40):151–162, 2002.

[23] B. Suger, B. Steder, and W. Burgard. Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3D-lidar data. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA 2015)*, Seattle, Washington, May 2015.

[24] Chuck Thorpe and Hugh Durrant-Whyte. Field robots. In *Proceedings of the 10th International Symposium of Robotics Research (ISRR'01)*, 2001. URL `http://www-preview.ri.cmu.edu/pub_files/pub3/thorpe_charles_2001_1/thorpe_charles_2001_1.pdf`.

[25] Abhinav Valada, Rohit Mohan, and Wolfram Burgard. Self-supervised model adaptation for multimodal semantic segmentation. *International Journal of Computer Vision (IJCV)*, jul 2019. ISSN 1573-1405. doi: 10.1007/s11263-019-01188-y. Special Issue: Deep Learning for Robotic Vision.

[26] Min Zhu, Jing Xia, Xiaoqing Jin, Molei Yan, Guolong Cai, Jing Yan, and Gangmin Ning. Class weights random forest algorithm for processing class imbalanced medical data. *IEEE Access*, 6:4641–4652, 2018.