FFI Norwegian Defence Research Establishment

22/00274

FFI-RAPPORT

LandX20 Experiment Report

– experiment for future land warfare capabilities with focus on increased situational awareness and unmanned systems

Kim Mathiassen
Magnus Baksaas
Idar Dyrdal
Brage Gerdsønn Eikanger
Fredrik Gulbrandsen
Daniel Gusland
Martin Vonheim Larsen
Eilert Mentzoni
Mathias Minos-Stensrud
Jones Moen
Eivind Bergh Nilssen
Olav Rune Nummedal
Tønnes Nygaard
Sigmund Rolfsjord
Aleksander Simonsen
Marius Thoresen
Lorns H. Bakstad
Dan Helge Bentsen
Christian Eggesbø
Eirik Grimstvedt
Trym V. Haavardsholm
Marius Halsør
Øistein Hoelsæter
David Kolden
Thomas Røbekk Krogstad
Robert Macdonald
Niels Hygum Nielsen
Guri Nonsvik
Else-Line Malene Ruud
Rikke Amilde Seehuus
Martin Syre Wiig
Einar Østevold

LandX20 Experiment Report

– experiment for future land warfare capabilities with focus on increased situational awareness and unmanned systems

Kim Mathiassen
Magnus Baksaas
Idar Dyrdal
Brage Gerdsønn Eikanger
Fredrik Gulbrandsen
Daniel Gusland
Martin Vonheim Larsen
Eilert Mentzoni
Mathias Minos-Stensrud
Jones Moen
Eivind Bergh Nilssen
Olav Rune Nummedal
Tønnes Nygaard
Sigmund Rolfsjord
Aleksander Simonsen
Marius Thoresen
Lorns H. Bakstad
Dan Helge Bentsen
Christian Eggesbø
Eirik Grimstvedt
Trym V. Haavardsholm
Marius Halsør
Øistein Hoelsæter
David Kolden
Thomas Røbekk Krogstad
Robert Macdonald
Niels Hygum Nielsen
Guri Nonsvik
Else-Line Malene Ruud
Rikke Amilde Seehuus
Martin Syre Wiig
Einar Østevold

Norwegian Defence Research Establishment (FFI) 21 March 2022

Keywords
Situasjonsforståelse
Ubemannede luftfarkoster (UAV)
Ubemannede bakkekjøretøy (UGV)
Radiopeiling
Brukergrensesnitt
Autonomi

FFI report
22/00274

Project number
1579,1400,1505,1514

Electronic ISBN
978-82-464-3390-5

Approvers
Lorns Harald Bakstad, *Research Manager*
Halvor Ajer, *Director of Research*

The document is electronically approved and therefore has no handwritten signature.

Summary

LandX20 was a collaborative experiment and demonstration where four research projects from the Norwegian Defence Research Establishment (FFI) participated. The goal of the experiment was two-fold. The first goal was to develop a situational awareness system that could use unmanned assets, and to field test it under controlled conditions. The second goal was to demonstrate the future possibilities for such a system to invited guests. The experiment and demonstration took place between 31 August to 4 September 2020, the integration week, and between 14 and 18 September, where the last day was the visitor's day.

The experiment included many different systems developed at FFI. Two Sentry systems, which include radars and Pan-Tilt-Zoom (PTZ) cameras, were used to track persons and vehicles. The tracked persons and vehicles were displayed on a map in a newly developed Graphical User Interface (GUI), which is web based using the CesiumJS framework. A swarm of four Unmanned Aerial Vehicles (UAVs) was also present. These have been designed and built at FFI, and can be tasked to monitor an area, detect and track persons, and report the results in the new GUI. Additionally, two Unmanned Ground Vehicles (UGVs) were present at the experiment. One of the UGVs was used for autonomous terrain navigation and the other was equipped with the Sentry system. The last system was a passive Radio Frequency (RF) sensor system used for detection and localization of radio emitters.

The experiment was successful. The system for situational awareness, which consisted of the two Sentry systems, the drone swarm and the newly created GUI, was demonstrated to the visitors, along with the passive RF sensor system. However, due to hardware problems, the UGV was unable to drive autonomously during the event.

Sammendrag

LandX20 var et eksperiment og en demonstrasjon hvor fire prosjekter ved Forsvarets forskningsinstitutt (FFI) deltok. Målet med eksperimentet var todelt. Det første målet var å utvikle et situasjonsforståelsessystem som kunne bruke ubemannede enheter og teste systemet i felt under kontrolerte forhold. Det andre målet var å demonstrere framtidige muligheter for et slikt system til inviterte gjester. Eksperimentet ble gjennomført perioden 31. august til 4. september 2020, som var integrasjonsuken, og perioden 14. til 18. september, hvor det var besøksdag.

Eksperimentet inkluderte mange forskjellige systemer som er utviklet ved FFI. To Sentry-systemer, som inkluderer radar og Pan-Tilt-Zoom-kamera, ble brukt til å detektere og følge personer og kjøretøy. Posisjonen til og kamerastrøm av personene og kjøretøyene ble vist i det nylige utviklede brukergrensesnittet, som er basert på CesiomJS. En sverm med fire UAV-er ble også brukt. Disse har blitt designet og bygd på FFI, og kan bli satt til å overvåke et område, detektere og følge personer og rapportere tilbake resultatene til det nyutviklede brukergrensesnittet. I tillegg var to UGV-er til stede på eksperimentet. Den ene UGV-en ble brukt til autonom terrengkjøring og den andre ble utstyrt med Sentry-systemet. Det siste systemet var et passivt radio frekvens system (RF-system) som ble brukt til å detektere og lokalisere radiosendere.

Eksperimentet var vellykket. Systemet for situasjonsforståelse, som besto av de to Sentry-systemene, dronesvermen og det nye brukergrensesnittet ble demonstrert til de besøkende, sammen med passivt RF-system. Imidlertid fikk ikke UGV-en kjørt autonomt på grunn av hardware problemer.

Contents

Summary 3

Sammendrag 4

1 Introduction 7
- 1.1 Envisioned scenario 7

2 System overview 10
- 2.1 User interface 10
  - 2.1.1 Technological Overview 11
  - 2.1.2 Usage 12
- 2.2 Sentry 12
  - 2.2.1 Multi-sensor tracking 13
  - 2.2.2 Platforms 13
  - 2.2.3 Software overview 14
  - 2.2.4 Sensor Processing and Target Tracking 15
  - 2.2.5 Acoustic sensor 17
- 2.3 Valkyrie 19
  - 2.3.1 Platform and hardware 19
  - 2.3.2 Flamingo 19
  - 2.3.3 Sensor Processing and Situational awareness – Warpath 20
  - 2.3.4 Decisional Autonomy – HAL 21
- 2.4 UGV 26
  - 2.4.1 Milrem Themis and hardware modifications 26
  - 2.4.2 Autonomy 29
  - 2.4.3 Perception and localization 31
- 2.5 Passive RF sensors 34

3 Experiment 35
- 3.1 Sentry, Valkyrie and user interface 35
  - 3.1.1 Scenario 35
  - 3.1.2 Tracking 37
  - 3.1.3 User interface 37
  - 3.1.4 UAV flights 37
- 3.2 UGV 39
- 3.3 Passive RF sensors 39

4 Conclusion and future work 41

References 42

1 Introduction

LandX20 was a collaborative experiment and demonstration where four FFI projects participated. It was initiated by the FFI project *1400 Situational awareness and active protection systems*. As it was relevant for other projects, the experiment was expanded to include relevant projects in the FFI Research Programs *Combat systems* and *Autonomous systems*. This included the projects *1505 Autonomy* and *1514 Unmanned ground vehicle - technology project*. The project originating the idea was finished before the experiment was completed, therefore the project *1579 Future maneuver warfare* joined the experiment, as this project continued the research efforts in the *1400 Situational awareness and active protection systems* project.

The goal of the experiment was two-folded. The first goal was the development of a situational awareness system that could use unmanned assets and to field test it under controlled conditions. The second goal was to demonstrate the future possibilities for such a system to invited guests.

1.1 Envisioned scenario

A vision of what the experiment should include was created in the initial planning phase. This was further concretized by creating a scenario containing all relevant components. Figure 1.1 shows a sketch of the envisioned scenario. In the figure, yellow indicates ground, green indicates forest areas which you cannot see through, gray indicates a road and light blue cones are areas observed by different units. The friendly units are *SVERM*, *TOR* and *TRANSIT* which represent a UAV, UGV and a manned combat vehicle controlling the unmanned assets, respectively. All of the friendly units are equipped with different sensors to locate enemy forces. Red forces are an infantry squad in the north and a vehicle in the east. The movement of the units are indicated with an arrow, and the letters and numbers next to the arrows are used in the below scenario description to indicate the flow of events.

The scenario proceeds as follows. A set of units consisting of one manned armored vehicle, an unmanned vehicle and a swarm consisting of four drones have been tasked to monitor an area. The manned vehicle managed to observe to the north and east, but has a blind spot behind the forest area to the northeast of the vehicle. Also, there is an unmonitored area behind the forest in the northwest. The commander decides to send the swarm to the northwest (1), in order to monitor the area that cannot be observed by the vehicles. When in position the swarm detects a group of enemy soldiers heading east (A), and starts to track the soldiers and send their position to the manned vehicle. The swarm is detected by the enemy infantry squad and they manage to eliminate one of the drones. Since the swarm is self-configuring it still manages to complete its mission and track the enemy soldiers.

The commander, concurrently with dispatching the swarm, sends the unmanned vehicle north in order to monitor the blind spot behind the forest area to the northeast of the manned vehicle (2). In this position the unmanned vehicle will also be able to detect the enemy infantry squad before the manned vehicle.

After the unmanned assets have arrived at the positions, an enemy combat vehicle comes in from the south. It is detected before it comes into view, as the manned vehicle has a Radio Frequency sensor that detects radio communication emitted by the incoming vehicle. The enemy combat vehicle continues on the road moving north (B). It is then detected and tracked by the manned vehicle's camera and radar, and is out of sight from the unmanned vehicle. While the enemy combat vehicle is moving north, the enemy infantry squad is moving east (C). Eventually the enemy combat

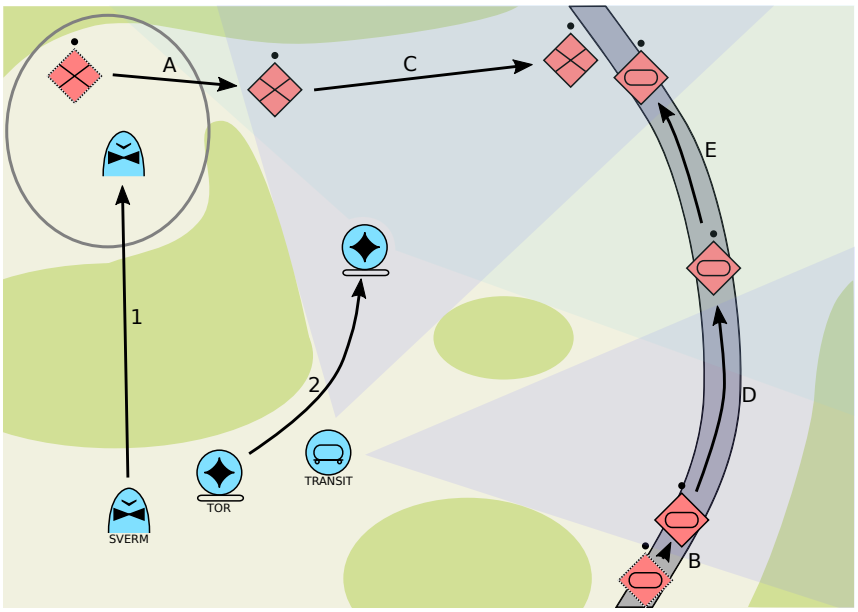


Figure 1.1 The scenario envisioned for LandX20. Yellow indicate ground, green indicate forest areas which you cannot see through, gray indicates a road and light blue cones are observed areas by different units. SVERM *is a UAV swarm,* TOR *is a UGV and* TRANSIT *is a manned combat vehicle. Red forces are an infantry squad in the north and a vehicle in the east.*

vehicle gets in the unmanned vehicle's field of view, and is detected and track by both ground units (D). When the enemy combat vehicle moves further north it it out of the field of view of the manned vehicle, and then reappears again (E).

The scenario is created to highlight a number of features that are needed in future situational awareness systems and how unmanned assets can be used on the battlefield. The main points are:

Detection and tracking of personnel using a swarm Sending out a swarm of UAVs that are able to cover a large area, and report back detection of possible threats is very valuable. It is even more valuable if the swarm can track the detection and send their position to be displayed on the Command and Control (C2) system.

Self-configuring swarm One of the benefits with a swarm is that it can take losses and reconfigure itself autonomously.

Autonomous driving Unmanned ground assets are valuable when they do not have to be continuously controlled by a human operator, but can drive by themselves in the terrain.

AI-powered automatic target detection Using AI powered algorithms to automatically detect and classify targets.

Sensor fusion of radar and camera to track enemy units Using different types of sensors with different abilities can make it easier to correctly detect and classify objects. Combining these measurements will make it easier and more robust to track the objects.

Tracking one unit from two or more sources One challenge when having two or more sensor packages at different locations is that tracking the same object is difficult, because it is hard for the system to know if the object detected by the two sensor packages are actually the same

object or two different objects.

Use RF sensors to detect radio signals Detecting enemy radio communication signals can give hints on where the enemy is located even without having to see them using visual sensors.

The report will first give an overview of the different systems used in the experiment in Chapter 2. Then Chapter 3 describes what was done during the experiment. Chapter 4 summarizes the results of the experiment and lessons learned.

2 System overview

This chapter introduces the systems demonstrated at LandX20. The chapter will start with an overview of the system architecture of the system-of-systems and their dependencies and interconnections. A large portion of LandX20 was focused on situational awareness. Because of this, most of the systems for situational awareness were connected in the same network, whereas other systems such as the UGV and localization using passive RF sensors were not connected to the same network.

An overview of the systems is shown in Figure 2.1. The situational awareness was achieved using the two Sentry sensor platforms (described in Section 2.2), and the sensor-equipped UAVs (Section 2.3), which were all connected together and visualized using a custom user interface (Section 2.1). The other two systems, the UGV (Section 2.4) and passive RF sensors (Section 2.4) were not part of that network, but used their own network to connect to their specific control stations.

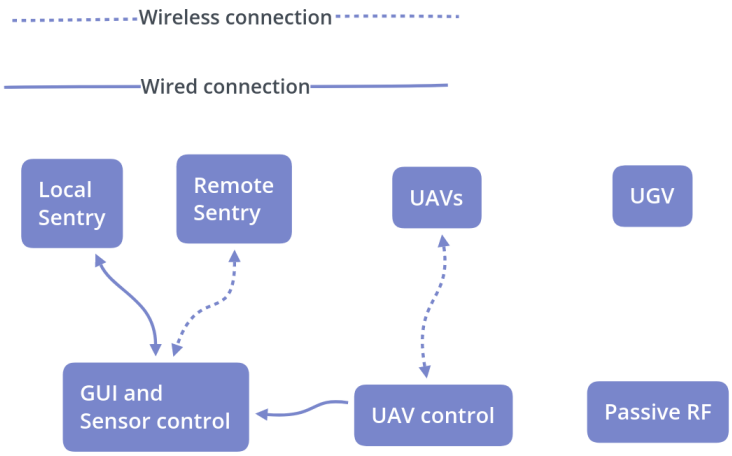


Figure 2.1 High level system overview.

2.1 User interface

This section will present the User Interface (UI) used during LandX20, both technologically and from a user perspective. The UI provides situational awareness and means of centralized control. It provides a central hub for information coming from the connected systems, giving the user a more complete picture of the situation than information from the systems individually. The user can also synchronize commands to multiple systems with greater ease than using separate interfaces for each system.

The sensor systems all stream information to the UI, and this introduce some challenges. The amount of data produced can easily overwhelm an operator rather than increase their situational awareness, additionally one would need high bandwidth link to transmit all the data. Aggregating information from multiple systems, using abstractions, reducing datastreams down to a manageable level of information, is therefore vital for such an UI. Presenting the operator with only the most relevant information.

The next two sections will give a brief overview of the user interface used during LandX20.

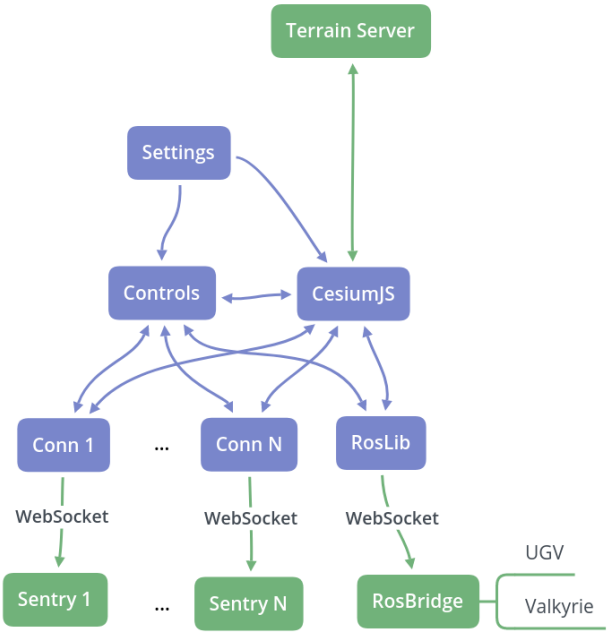


Figure 2.2 Main modules of the user interface

2.1.1 Technological Overview

The UI demonstrated at LandX is built around a software package called CesiumJS[1], a javascript library for creating 3D globes and maps. Cesium allows us to visualise a variety of maps, and render vehicles and other entities onto those maps. The maps are collected from either a public or local map server, giving a choice between flexible development and offline usage. Apart from Cesium, the LandX-UI consists of three modules; settings, control, and connection modules. Figure 2.2 shows a simplified model of the UI.

The connection modules allow the UI to communicate with other parts of the system using Web Sockets (WS) to set up the connection. One specific connection to note is the Robot Operating System (ROS) connection. This connects the UI to a ROS Master running on a computer and allows the UI to subscribe and publish to topics on that computer. Another notable connection is with the Hybrid Autonomy Layer (HAL) system [17]. HAL uses Battle Management Language (BML) to both give and receive orders [12], and as such, BML is implemented in the LandX-UI, and can be used to give orders to vehicles controlled by HAL.

The settings module contains the main setup for the UI. Mainly, the connections to the map servers, the connections to the vehicles, such as the Sentry platforms, and which incoming streams of data to visualize on the screen. The settings are easily adjustable while running, and stores the configurations for the next time the UI is launched.

The control module transforms user input to commands to the UI itself, or the systems connected to the vehicles. The control of the vehicles and sensors is heavily inspired by Real-Time Strategy (RTS) games, which makes the control intuitive for many users. The swarm of Valkyries, for instance, can be selected by a left click, and ordered to move by right clicking. Similar

[1] `https://cesium.com/platform/cesiumjs/`

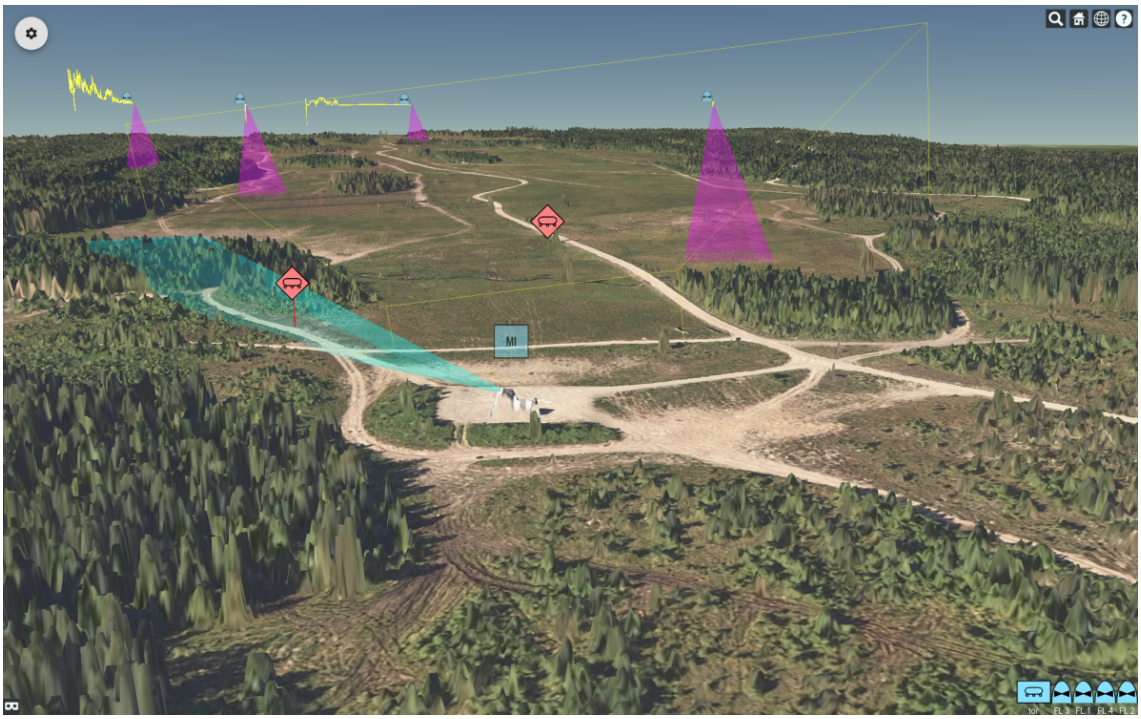


Figure 2.3 The bottom right of the map shows connected vehicles, four UAVs and one UGV. The purple frustums visualizes the Field of View (FOV) of the cameras on UAVs, while the yellow trails show the paths taken by the UAVs. The Sentry system, marked with a blue MI, shows the camera FOV in light blue and the radar FOV is outlined in yellow. The red markers show two hostile vehicles being tracked.

combinations are used to issue take off and landing commands to one or more UAVs in the swarm.

2.1.2 Usage

The user interface runs a local map and terrain server, so high quality 3D maps can be utilized without any other connections, but we also provide interfaces to add external mapping information if available. We display red and blue forces with symbols according to STANAG APP-6, based on the aggregated information. We also provide options for displaying sensor coverage and unit movement history.

2.2 Sentry

Sentry is FFI's concept for multi-sensor situational awareness and is used for Counter Unmanned Aerial System (C-UAS) [19, 3, 18], situational awareness and surveillance [7]. Sentry is both a software and hardware concept, which focuses on multi-sensor detection, classification and tracking. The overall goal of the Sentry platform is to build knowledge about tactical multi-sensor situational awareness across all domains, and make this knowledge and technology available to the Norwegian Armed Forces and industry.

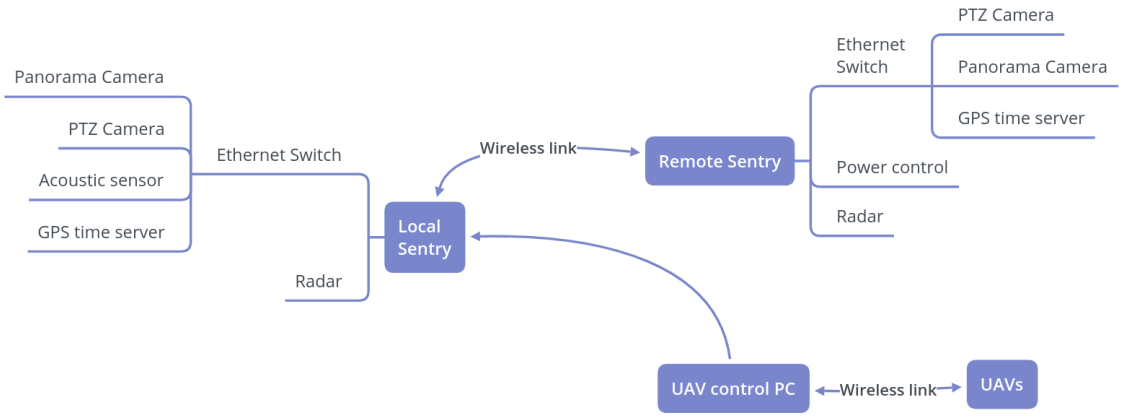


Figure 2.4 Hardware overview of the two Sentry platforms during LandX20.

2.2.1 Multi-sensor tracking

The motivation for multi-sensor tracking is the acknowledgement that every sensor has its benefits and disadvantages. Sensors differ in their ability to detect certain targets in given conditions, and their accuracy in both localization and classification. Choosing a single sensor for situational awareness is therefore challenging and will often lead to sub-optimal results. The multi-sensor tracking in Sentry aims to exploit each sensor's strengths and use sensors with highly complementary features to achieve good performance.

One example of such complementary sensors is the combination of PTZ-cameras and radar sensors. A radar sensor is well-suited for continuous surveillance of large volumes, detecting moving targets, and precisely measuring target range. The directional and classification accuracy of radars is typically only fair, and detecting stationary targets can be challenging. A PTZ-camera can provide highly accurate direction measurements and classification, especially for static targets. However, the PTZ-camera is neither suited for scanning large volumes continuously, nor does it provide accurate range measurements. By using these two sensors together, Sentry can exploit the strengths of each of the sensors and mitigate their weaknesses.

Combining observations from different sensors is challenging, and if not done correctly, the combined performance will likely be worse than just using the best sensor alone. To succeed with multi-sensor tracking, each sensor must provide time-synchronized data with sufficient accuracy in suitable reference frames. If a sensor can provide a target signature, classification or other relevant information, this should be made available to the tracker. Additionally, we must continuously be able to model each sensor's expected performance. These aspects put stringent requirements on the system hardware, covered later in the section. Simultaneously, a good general tracking framework should be able to handle multiple reference frames and imperfect clock-synchronization and exploit different types of information from the sensors.

2.2.2 Platforms

There were two Sentry sensor platforms present at LandX20, one located near the command center, and one set to be a forward observer. Figure 2.4 shows the system overview of both

Figure 2.5 The two Sentry platforms at LandX20. To the left, the remote sentry is shown mounted on a UGV, to serve as an autonomous forward observer. To the right, the main Sentry platform is shown.

platforms. The two platforms were connected via a wireless link, and the primary Sentry was connected to the main processing network with an Ethernet cable. Both platforms were equipped with a radar, PTZ-camera and a panorama camera, and the primary Sentry also had an acoustic sensor. For time-synchronization we used a Global Positioning System (GPS)-based Network Time Protocol (NTP) time server at both platforms. Alternatively, we could have used one of the Sentries as the time server together with the wireless link for synchronization. The primary sentry was also connected to the UAV control station to receive information from the UAVs. Figure 2.5 shows an image of the two sentry platforms.

2.2.3 Software overview

Sentry includes both flexible hardware configurations combined with a common and flexible software framework specifically made for situational awareness. A high-level flowchart of this software framework is shown in Figure 2.6. The main components of the system are:

Sensor modules: These contain bespoken processing approaches for each of the specific sensors. The main goal for each sensor is to provide highly accurate detection, and to describe the detection using multiple metrics, such as position, class and velocity, by exploiting the sensors' strengths. This is achieved using both conventional detection and detection using deep learning algorithms.

Tracker: This module can combine all the well-described detection from each sensor into a single fused-image.

Utilities: This module is used to support the operation of the other processes and simplify development. It contains features such as position managing, time managing, logging and reading log files synchronously.

Calibrator can utilize the sensor specific code along with a collaborative target to automatically calibrate both the sensors' absolute and relative position, as well as the timing difference of the sensors. This is all connected to a GUI that enables the operator to control the sensors (*SensorControl*), such as queuing the PTZ camera to point towards an object that was detected by the radar.

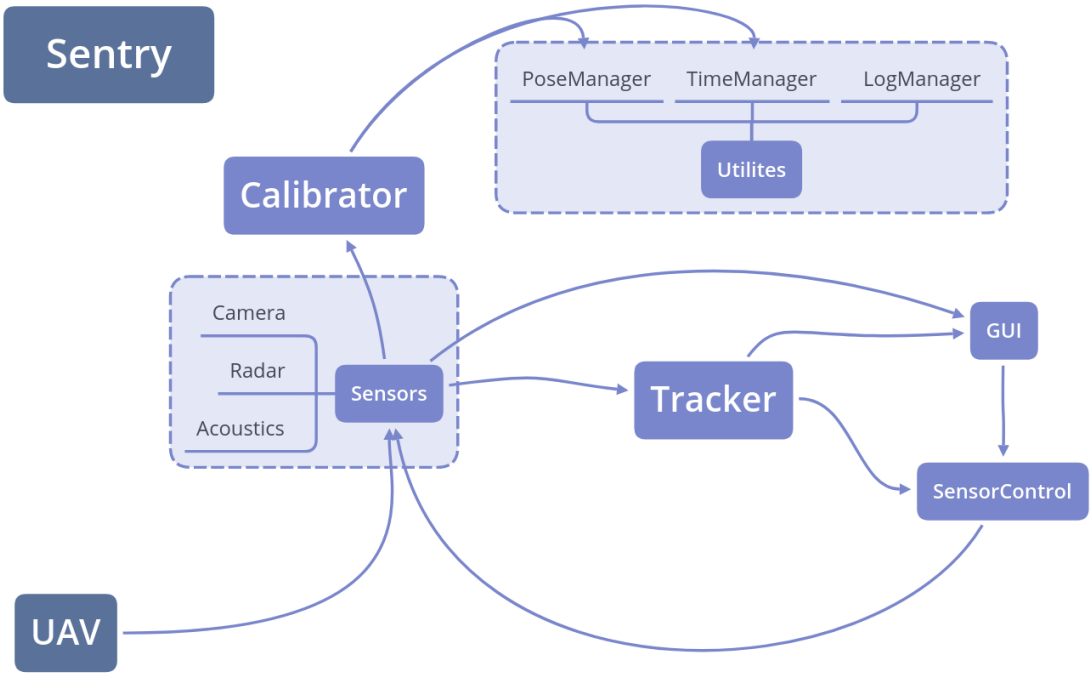


Figure 2.6 Sofware overview of Sentry

The graphical user interface displays the condensed data from the tracker in a clear and comprehensible way to the operator and allows for user feedback. A snapshot of the user interface is shown in Figure 2.7.

2.2.4 Sensor Processing and Target Tracking

The Sentry sensor processing and tracking is structured as a *tracking-by-detection* [2] architecture built around a concept we call *multi-feature tracking*. Instead of having a fixated format for detection data and target models, our framework has a general interface for describing detections and tracks probabilistically. Depending on our specific sensor setup, we are able to detect and measure different properties, or *features*, of the observed objects. Multi-feature tracking makes it possible to add probabilistic models for the sensors and object features as plugins to the tracker. In turn, this means the tracker can properly leverage data from any type of sensor, as long as we can provide probabilistic models for the object properties and how they are measured.

We will not go into the details of how the probabilistic models should be defined in the tracking framework, but instead give a brief overview of what is required. Figure 2.8 sketches the components in the architecture. Upon startup, we register models for each feature that we will be using with the tracker. The *tracker manager* first registers models for object features that are common for all sensors. Next, we register sensor-specific feature models. Each of these feature models must be able to provide three functions:

[2]Tracking-by-detection simply means that we as a first step make *detections* from sensor data alone, and then gather detections over time to form *tracks*. The alternative is *track-before-detect*, where information about existing tracks is also used when making detections from new sensor data, but this is far less common.

Figure 2.7 Screenshot of the Sentry UI during LandX20.

- **Comparison**: How likely is it that a given measurement originate from a given track?
- **Prediction**: What is the expected state of a given track at some given time in the future?
- **Update**: Update the current state estimate of a given track with a given measurement.

Every time a sensor acquires a new piece of data (for instance an image, or a radar scan), the data is run through a detection algorithm, which produces a set of *detections*. These detections are fed to the tracker, which will then use the feature models to predict its tracks, compare the tracks to the new detections, and update the track estimates. Alongside the detections the sensor must also supply a *sensor model*, which provides information about what the sensor can expects to detect and not. The tracker then uses this sensor model to assess its confidence in each track: Does the track represent a real object, or is it just a result of random clutter?

The most important role of the sensor models is to inform the tracker of what the sensor *does not* see. A camera is obviously not able to detect objects outside its field of view. Similarly, a radar which relies on Doppler shift to make detections cannot detect objects that have no radial velocity. Without the sensor model mechanism, both the camera and the radar would inexplicably be "missing" detections for many of the tracks. The absence of these detection would force the tracker to either falsely dismiss more tracks that represent real objects, or fail to dismiss tracks that in reality are made up from clutter detections.

In Sentry for LandX20 we used these three features in the tracker:

- **Motion**, which at any point in time has one of three representations:
  - 2D position (bearing and range) and velocity. Used for tracks only seen by the radar.
  - Direction (bearing and elevation) and direction rate. Used for tracks only seen by the cameras.
  - 3D position (Cartesian) and velocity. Used for fused tracks seen by both sensors.
- **Classification**, which is a distribution between the classes: Person, Vehicle, UAV, Bird and Clutter
- **Visual signature**

Since we only used a 2D-radar at LandX, the radar is only able to provide bearing, range and range rate for each detection. This means that we do not know the elevation for radar detections, nor

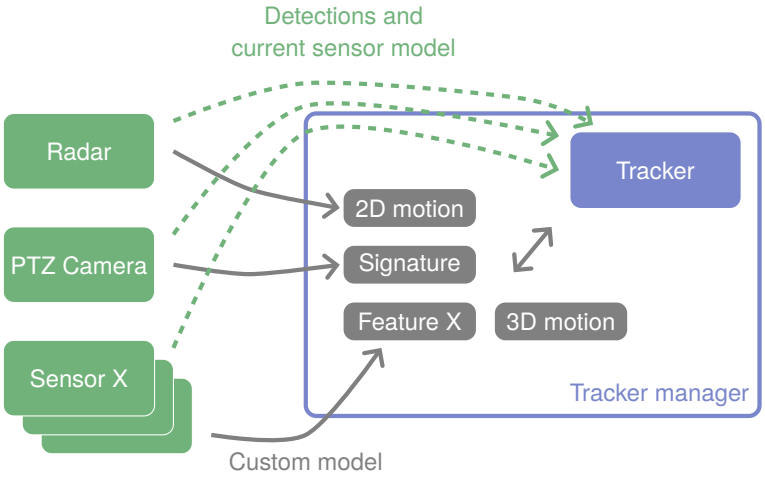


Figure 2.8 The tracking architecture within Sentry. The tracker manager defines the default 3D motion model, but each of the sensors can also provide additional models for the tracker to use. As a sensor processes data, it will provide sets of detections to the tracker, as well as a model of its current state describing what the sensor expects to detect.

for tracks that have only used the radar detections. Conversely, the camera provides both bearing and elevation, but generally has little range information. We use sensor-specific representations for tracks that have only been observed by a single sensor. This means that no information is lost as long as we track using a single sensor, and that as soon as we detect the object with the other sensor, we can obtain the fused estimate at full representation (3D Cartesian position and velocity). We could have used a 3D Cartesian representation also for tracks that have only been detected by either camera or radar, by allowing these tracks to have a large uncertainty in the unknown direction. However, this would come at a great cost in precision loss, and would really only work well for tracks that have been detected by both sensors for some time.

Due to the flexibility of the feature- and sensor models, we can easily add more sensors to the system. This could be sensors that are completely different from cameras and radars. Although we did not perform full fusion between the remote and local Sentry, this could have been achieved by adding the sensors of the remote Sentry as additional sensors on the local Sentry, and forwarding their detections via the radio link. The combined flexibility provided by the sensor models and multi-feature tracking is the key to be successful with multi-sensor tracking.

2.2.5 Acoustic sensor

An acoustic sensor for detection, bearing estimation and classification of sound sources, i.e. military targets, has been developed for use as a component in Sentry in combination with cameras and radar. The front-end of the system is the Discovair-G1 microphone array from the Norwegian company Squarehead Technologies. The microphones (256 MEMS-microphones) are mounted as a 16 by 16 array on a square panel of size 40 x 40 cm. The microphone panel also includes a web-camera, showing the area of coverage of the sensor. Array processing techniques are applied to compute beamformed acoustic signals over a grid of points (i.e. spacial directions) within the area

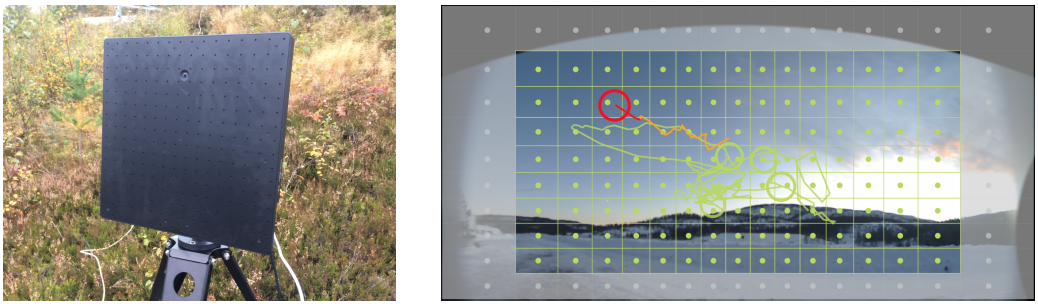


Figure 2.9 Discovair G1 sensor panel (left) and the web-cam image showing the grid layout and examples of tracked drones (right).

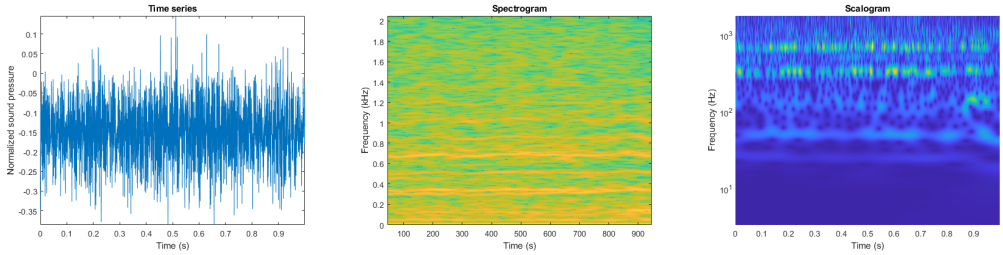


Figure 2.10 Example showing time series of one second duration (left), corresponding spectrogram (middle) and scalogram (right).

of coverage. Figure 2.9 shows the sensor panel and the grid setup used for the LandX20 trial. A strong beamformed signal in a particular grid point is thus an indication of an acoustic source in the corresponding direction. Detection of local maxima over the grid is carried out to to identify separate sound sources, which can the be tracked individually.

The sensor system was delivered with the so-called *discovair software* for detection and tracking of drones. The green and red tracks shown in the figure is generated with this software from Squarehead Technologies. For use in Sentry we utilized the existing beamforming software, which is executed on a signal processing unit (SPU) connected to the sensor panel. However, to enable integration of the acoustic sensor in Sentry in time for the LandX20 trial, we had to develop our own experimental software for bearing estimation based on the grid of beamformed signals. Here, local maxima in sound level could in principle be estimated, based on interpolation over the grid, to represent the bearing to multiple sound sources of interest. However, in the version developed for use in LandX20, only the strongest source was selected for further consideration. The beamformed signal from the corresponding direction is then classified by a deep convolutional neural network (CNN) with 53 layers, developed at FFI.

This network was trained on an in-house database of acoustic recordings of a range of sound sources (ground vehicles, drones, helicopters and other aerial vehicle, and background noise from a number of sources, both man made and natural). Since this type of network requires input in the form of images, short intervals of sound (typically one second time intervals) are converted to a time-frequency representation in the form of spectrograms or scalograms. For the LandX20 version of the system, scalograms were used. Figure 2.10 shows examples of a recorded signal and the corresponding spectrogram and scalogram of the signal.

The software is executed on a MacBook Pro connected to the Squarehead system (sensor panel

and SPU). The beamformed signals are transferred from the SPU to the MacBook, where bearing estimation and classification is executed in Matlab. Finally, the results (bearing and class) is passed to Sentry through a web-socket approach.

A full integration of the acoustic sensor in Sentry was not developed for this trial, due to limitations in time and resources. The preliminary web-socket solution proved to be unstable, and was thus only used for short periods during the trial. However, the acoustic sensor was tested through most of the trial in a stand-alone mode. In general the sensor proved to be capable of detecting both ground vehicles, drones and passing helicopters on many occasions during the trial, where ground truth could be established. However, it was not possible to do systematic tests to measure detection capability and false alarm rates during LandX20.

2.3 Valkyrie

Valkyrie is a research system for distributed autonomy developed at FFI. It integrates autonomy software components from FFI, such as the perception system Warpath and the decision autonomy system HAL [6] on multiple autonomous UAVs. The Valkyrie architecture is modular with respect to platform, payload and communication infrastructure, and is built around a companion computer. The UAVs that we use are a prototype platform built at FFI called the *Flamingo* which is a small quadcopter drone with an onboard companion computer [11].

2.3.1 Platform and hardware

The Valkyrie system used at LandX20 consisted of four UAVs and a ground station. This is the normal configuration for the Valkyrie system, but UAVs can be added or removed from the system depending on the situation or scenario. Each UAV was equipped with a thermal camera with a 640x512 pixel resolution, a 5.8 Ghz mesh radio, and an on-board computer. The mesh radio provided communication between the UAVs and the ground station, sending sensor information and flight statistic from the UAV to the ground station, and sending commands from the ground station to the UAVs. The same radio also provides UAV to UAV communication making it possible for the UAVs to conduct joint missions by cooperating as a swarm.

2.3.2 Flamingo

Flamingo is a four motor UAV developed at FFI for autonomy experiments[11], and is shown in Figure 2.11. The UAV weighs from 2.2kg, depending on payload and radio configuration. The Flamingo used in the LandX experiment was equipped with a thermal camera and a 5.8 Ghz mesh-radio, giving it a total weight of 2.8 kg. A 200 Wh battery gives the UAV a total flight time of approximately 45 minutes with the 2.8 kg configuration, but most flights are limited to 30 min to limit battery wear. By having two batteries for each UAV, using one and charging the spare, it was possible to have several drones airborne at the same time over a longer period of time.

The UAV has an integrated Nvidia Jetson TX2 companion computer for on-board sensor processing. Processing the sensor data on board makes the UAVs more independent and autonomous since they don't require a continuous radio link to the ground station, and can operate on their own. On-board processing also makes the system more scalable, as the ground station computer doesn't have to perform sensor processing for each UAV in the system, and the network doesn't have to handle a continuous sensor stream for all the UAVs simultaneously.

Figure 2.11 A Flamingo with a thermal camera

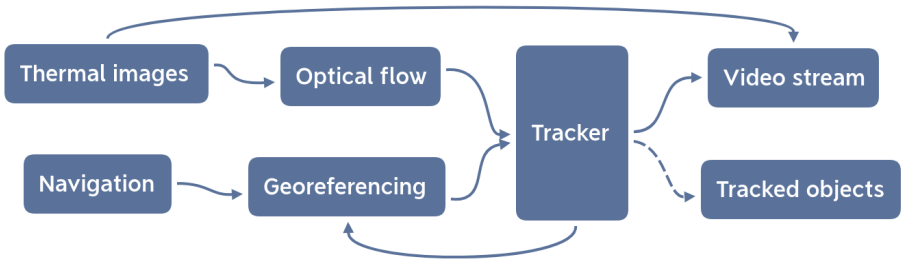


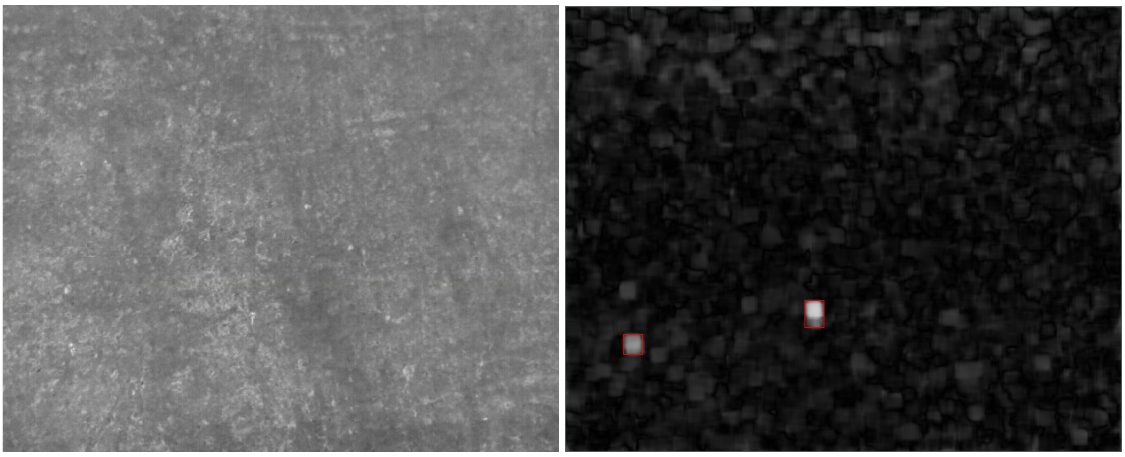
Figure 2.12 Overview of the Warpath system in Valkyrie

2.3.3 Sensor Processing and Situational awareness – Warpath

Warpath is our framework for sensor processing on autonomous platforms, and is used across most of the unmanned platforms at FFI. The main purpose of Warpath in Valkyrie is to process input from sensors and output a situational awareness that can be used both as input to HAL, and as feedback to operators through the control station. Warpath provides unified interfaces for representing sensor data, as well as handling navigation, target tracking and mapping. These interfaces make it easy to re-use methods for sensor processing across platforms, with different sensor and navigation setups.

In Valkyrie, the main task of Warpath is motion detection and tracking from the thermal camera, together with streaming video to the control station. An overview of the Warpath processing pipeline is shown in FLIR Boson camera. An *optical flow* algorithm is applied on the thermal images to detect moving objects. The *tracker* determines whether a motion detection is from the same object as earlier detections, if it is a new object moving in the scene, or if it is noise, such as moving trees. Tracks with enough confidence are *georeferenced*. In order to georeference, the position and orientation of the camera relative to a geographic origin is needed, which is obtained from the *navigation* module. *Tracked objects*, which has been georeferenced, are communicated to HAL. A *video stream*, with the tracked objects visualized, is sent to the operator.

In realistic conditions, especially in the Norwegian summer, it can be close to impossible to spot persons or other objects of interest in thermal still images. With summer temperatures and with reflection from direct sun, the surface temperature of the ground is often close to the surface temperature of a person. Figure 2.13 (i) shows an example of this, where two persons are out



(i) Raw image from the FLIR Boson.

(ii) Processed optical flow in the same image, showing two detected moving objects.

Figure 2.13 Two persons walking in a field during a warm summer day. Personnel can be easy to detect in the thermal video feed, but appear almost invisible in still images such as in (i). Using optical flow processing we can highlight the motion, as shown in (ii), where the persons appear as two moving blobs.

walking in a grassy field in the sun. If, however, an opeator were to look at the thermal video feed, one would discover the persons immediately. This indicates that it is the motion of the persons which makes it possible to detect the them. In this case, an optical flow algorithm is a good choice for detector, since this algorithm measures the motion between two images. It uses two sequential images and determines how a pixel has moved from one image to the next. By subtracting the average motion in the image, we can detect objects that move differently than the camera motion, as is visualized in Figure 2.13 (ii). One drawback with this method is the high sensitivity to motion in the video. In high winds, the drones might have difficulty maintaining their positions, resulting in parallax of trees. This would be detected as moving objects by the optical flow algorithm.

In a conventional camera tracker pipeline, we would operate on georeferenced positions of objects. This makes it easy to incorporate prior knowledge into our motion model, for instance about how fast a person can move, or how fast a vehicle can accelerate. In order to georeference objects, however, we rely on poor measurements of heading and altitude, as is the case for most modern drones, because of weight constraints. The result is large uncertainty in the georeferenced position. Instead, we represent tracks as objects that have a position and velocity in the current image plane, and then track these objects through image frames. We still require relative navigation between consecutive images, but for short time periods, this *relative tracking* enables more accurate tracking association. Figure 2.14 shows successful tracking of four persons from two drones.

2.3.4 Decisional Autonomy – HAL

HAL has been developed in order to serve as an autonomous decision making module for all domains of autonomous platforms, air, sea and ground. On UAVs, HAL enables fully autonomous flight as well as operating with an operator in-the-loop. In the latter case, many of HAL's capabilities and features can still be used and various control aspects can be automated which makes operation of multiple assets much more feasible for a single UAV operator. The operator can select individual or

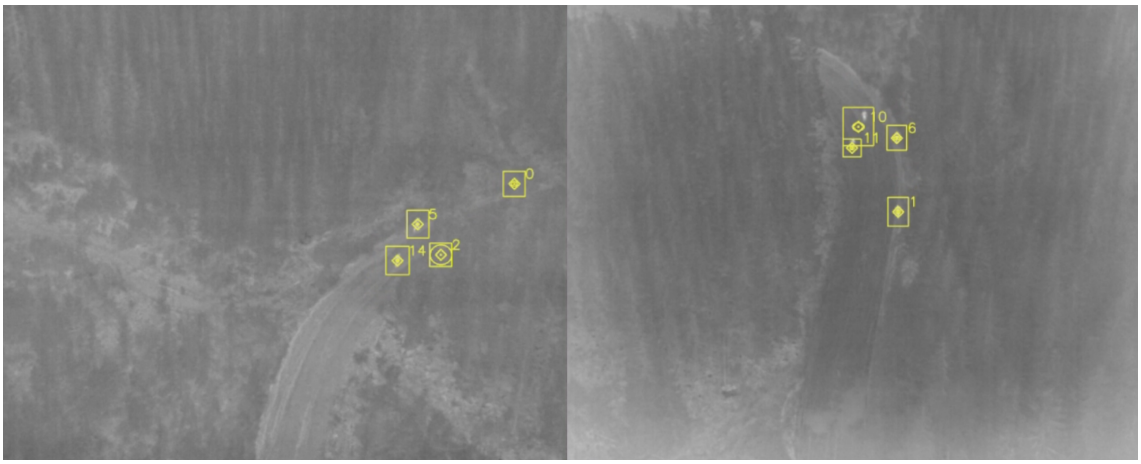


Figure 2.14 Two UAVs observing the same group of four people walking along a forest path. The yellow boxes indicate tracks on moving entities.

groups of units and provide high-level control inputs such as:

- Move here
- Cover this location
- Return home and land
- Search in this area

There are several aspects that a human UAV operator consciously or subconsciously considers and reasons about when being instructed to simply cover a location with an airborne sensor. The UAV must be brought to an appropriate altitude, which corresponds to the sensor type and field of view of the currently equipped payload. The sensor should be oriented towards the location of interest with the ideal angle of elevation, the best aspect angle towards the target and at an appropriate distance away. If multiple UAVs operate in the same area, either by the same operator or by others, the airspace needs to be segregated, or the flight paths coordinated. If other assets already monitor the same location, the new sensor should either overlap with the already existing ones, or minimize the overlap, depending on intention behind the sensor placement. Either way, for optimal utilization, the operator must explicitly consider the placement of other sensors in the system.

The HAL framework reasons about such complex high-level commands by breaking them down into a sequence of simpler, low-level sub-tasks that are more suitable for a machine to execute. All tasks, both high-level and low-level, can start and stop *behaviors* which are processes that produces the appropriate outputs or internal state changes, that solves the task at hand. The most obvious example is the *Waypoint Guidance* behavior that actually sends steering commands to the autopilot system in order to move the UAV to the waypoints computed by various tasks or other behaviors. Figure. 2.15 illustrates how HAL receives data from all the subsystems, and how it reasons internally about the sequence of steps that must be executed.

For example, in the case of a *Cover Location* task, this task is divided into subtasks that can be reasoned about separately. First, the *Find Location* task uses all available information about its own payload, about the objective and other UAVs or agents, to calculate an ideal sensor placement. In this task, all limitations in the control system and airspace coordination can be ignored as these are problems that are handled by other tasks later in the task tree. Then, when a valid solution is

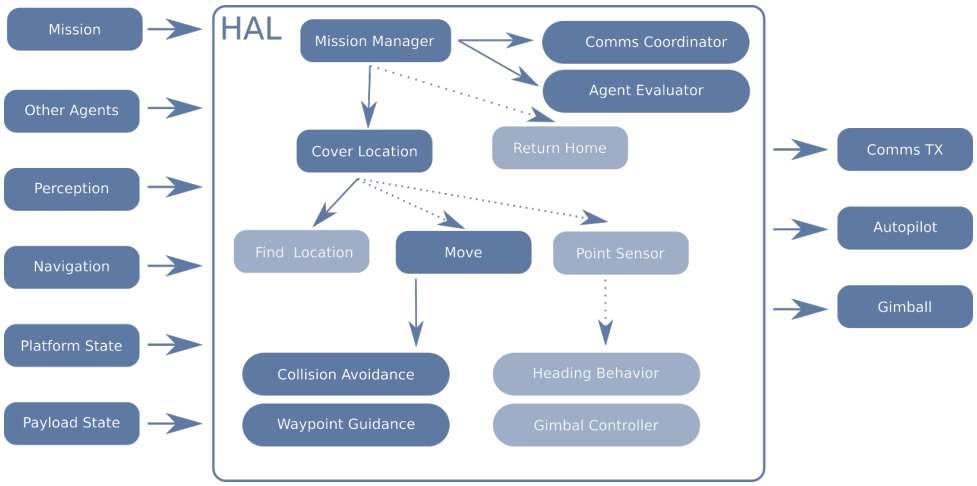


Figure 2.15 Illustration of the HAL information flow. Streams of information from subsystems like navigation, perception, communication, etc. are processed by HAL in the mission manager component. The shaded and dotted lines indicate planned/completed tasks necessary for completing a specific objective whereas the fully opaque boxes correspond to the current steps being executed.

estimated, the *Move* task is responsible for simply moving the aircraft to the calculated location. This task has several other problems to reason about, like the UAV's maneuvering capabilities, avoiding collisions, and coordinating the flight path with other assets. However, this task do not need to tackle the sensor configuration, region of interest or other aspects of the mission, which can be safely ignored during the movement phase. Finally, when the UAV (and payload) is in position, the *Point Sensor* task is executed in which the platform orients the sensor towards the location of interest and manages the perception modules. In parallel with all of this, HAL runs multiple other *behaviors*, which each are responsible for handling specific situations or subproblems.

Of course, any maneuvers performed to execute the overall task should be collision free and as quick as possible. *Valkyrie* is a swarm system in which the individual agents (UAVs) are independent autonomous assets, but can cooperate and share information. In the current configuration, all traffic between UAVs and the operator station is also intercepted by any other connected agents making each UAV aware of the position, goals and payloads of the other UAVs. Each unit can also belong to a *group*, which indicates that these assets are currently solving the same problem, for instance monitoring a location from multiple aspect angles. This information is used to increase the performance of the swarm, by collaborating on certain tasks, and also to ensure safety, by running active collision avoidance algorithms when multiple aerial vehicles operate in a densely populated airspace. For collision avoidance, HAL implements multiple algorithms, but uses the an Artificial Potential Field [13] method as default (see Figure 2.16). Here, the UAV constantly steers along virtual force fields which are generated by attractive forces (towards the desired location) and repulsive forces (away from dangers or other agents). Since the desire to avoid collisions are programmed to be stronger than reaching the goal through the strengths of the repulsive and attractive fields, respectively, the UAVs will tend to generate paths that avoid collision (provided that the information is available and current).

Another important aspect of the control autonomy layer is to report or handle anomalies in

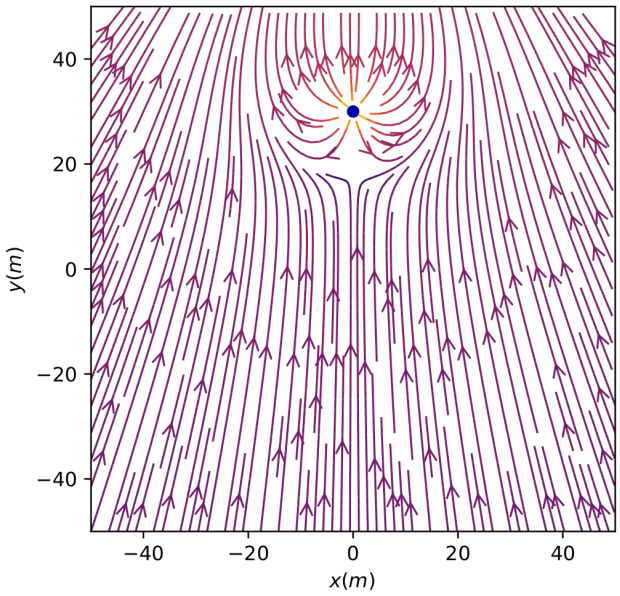


Figure 2.16 Example of virtual force fields generated by the artificial potential field algorithm. The field lines tend to point north (towards a goal) and the blue circle represents another UAV which generates a local repulsive field, thus bending the field lines around it.

a graceful manner. UAV systems have complicated flight control units with several states that the operator needs to be aware of at all times, like the systems' remaining flight time, the current altitude (above ground), the flight mode of the autopilot and the signal strength of the control communication link. With multiple assets, all this information load multiplies and is tedious to monitor simultaneously. These states are continuously monitored on-board by the HAL system. In addition HAL can take pre-programmed or reactive actions if a state is unexpected or potentially dangerous. For example, in the case of a critical failure, HAL has also scheduled a *Return Home* task which can be triggered by any events that is interpreted as a critical failure. Note that this task does not have any scheduled sub-tasks (Figure 2.15), as these will be planned when more accurate information is available and executed only if necessary. The type of reaction to a critical event can be reasoned about by HAL as well, and different actions can be taken depending on what is most appropriate for the mission or the current state of the system. For example, in the case of loss of communication with the control station, multiple actions can be taken, like continue the mission autonomously, stop and wait for re-connection, or plan a path to a location where a better signal is expected.

Valkyrie is currently integrated with ground control systems and operator stations using a variant of the FFI Coalition Battle-Management-Language (C-BML) implementation [12]. The swarm can therefore be controlled and monitored by multiple systems, as long as they comply with the FFI C-BML variant. During LandX20, we used two different systems, a dedicated UAV command and control system, as well as the more generic GUI system presented in Section 2.1. This is partly due to the immaturity of the swarm system, causing the need for more explicit control of aerial assets than currently available in the Cesium based GUI. As such, there were two operators of the

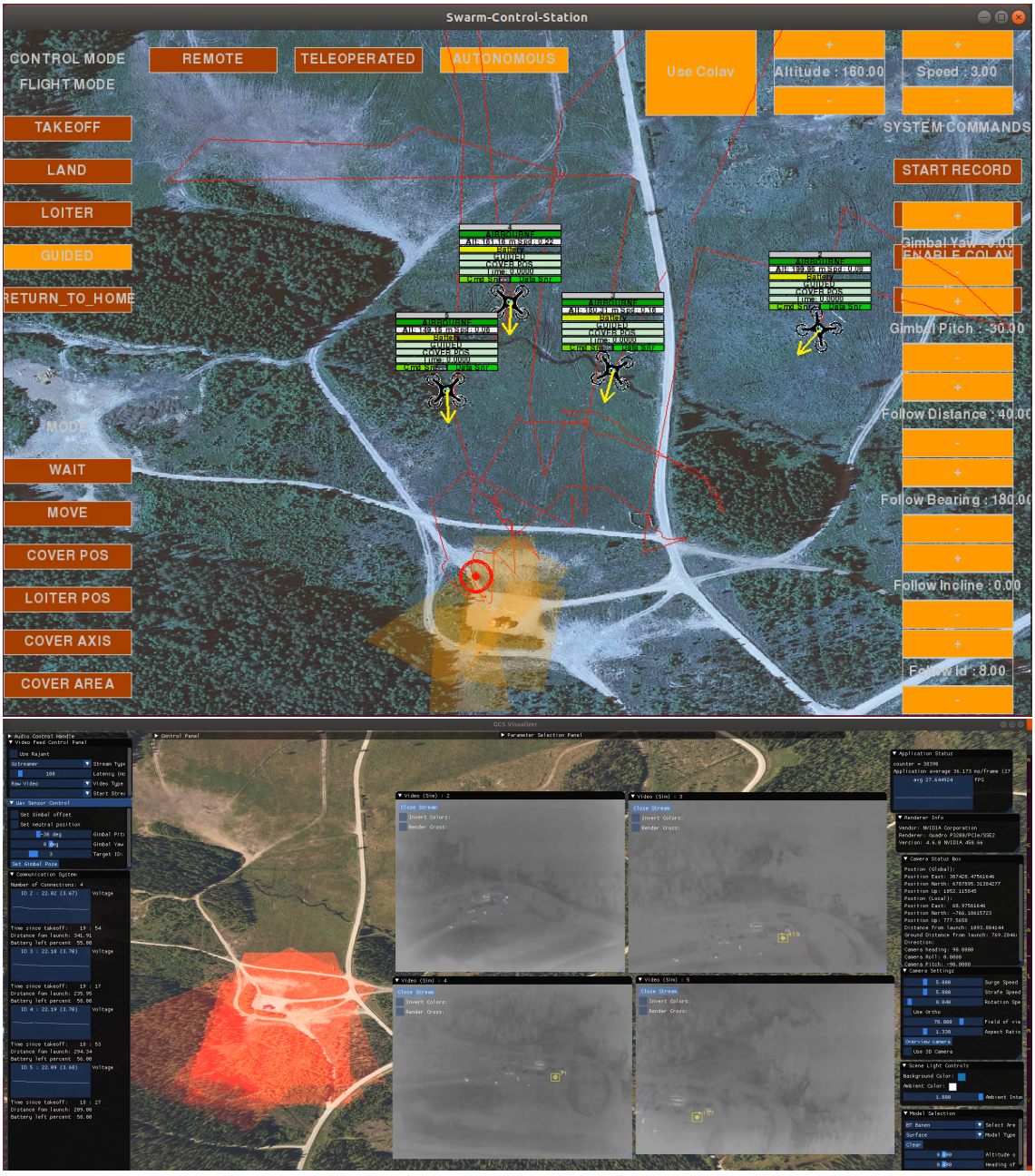


Figure 2.17 *Screenshots from the* Valkyrie *swarm control interface. The upper part shows the command and control map in which units and groups can be selected and tasked. The lower part shows the monitoring screen in which various UAV specific information is displayed, as well as live sensor feeds from the thermal cameras.*

integrated Sentry-Valkyrie system with two operating stations, one controlling the aerial sensors and one for the ground based ones, although all the data were assembled in the Sentry system. An example of the aerial control interface is shown in Fig 2.17.

2.4 UGV

This section will describe the two UGVs participating at LandX20. Both are THeMIS 4.5 platforms produced by Milrem Robotics. One is used as an autonomy research platform and is named *Tor* and shown in Figure 2.19. The other, named *Siv*, is currently unmodified, but will be modified and used for Manned-Unmanned Teaming experiments with soldiers, and shown in Figure 2.5. The hardware modifications made to *Tor* is described in Section 2.4.1.

The purpose of the UGV Tor in the LandX experiment was to function as a mobile remote sensor platform (for instance for Sentry) that may be moved to different observation positions. It can move in two modes, autonomously and tele-operated. In tele-operation, an operator drives the UGV directly, either using line-of-sight to see where to drive, or by using a camera feed from the vehicle to steer from. Tele-operation requires full attention from the operator to drive, and is therefore not ideal, as it either prohibits the operator from operating the rest of the system, or it means that the UGV can not be used continuously. In autonomous mode, the UGV can move autonomously between positions that are selected by an operator. Once the mission is specified the UGV does not require any inputs from the operator at all. Therefore, an autonomous UGV takes minimal attention away from the operator, and can therefore be a great addition to the system with minimal drawbacks.

Figure 2.18 shows the planned software framework, and builds upon the work made in [9]. The operator defines the mission for the vehicle using the *FFI Ground Control Station (GCS)*, and sends the mission to the vehicle's *decision making* component. This component interfaces with the on board *route planning* service and sends global paths to the *motion planning* component. These four components are the vehicle's main autonomy functions and are described in Section 2.4.2.

The *motion planning* component receives traversability maps from the *perception* component, and plan local paths that avoid obstacles in the map, and at the same time tries to follow the global route from the *decision making* component. Many components get the vehicle's position and orientation from the *localization* component. The *perception* and *localization* are closely linked together and described in Section 2.4.3.

The local path planned by the *motion planning* component is sent to the *path follower* component. This component is responsible for ensuring that the vehicle is located on the planned path. It receives a set of positions and headings and computes a continuous path between the received waypoints. It sends control signals to the proprietary low-level controllers on the platform. These components are described in [8].

The vehicle will also be compliant with the UGV Interoperability Profiles (IOP), using an *IOP bridge*, which is based on the work done in the NATO group IST-149[10], although this was not tested at the trials. The bridge converts between ROS messages and UGV IOP messages. Using the bridge, it is possible for UGV IOP compliant Operator Control Units (OCUs) to control the robot.

2.4.1 Milrem Themis and hardware modifications

The two UGVs used in LandX20 are produced by Milrem Robotics and are both of the type THeMIS 4.5. From the THeMIS 4.5 Operator Manual:

> The THeMIS is a tracked unmanned ground vehicle (UGV) that features diesel-

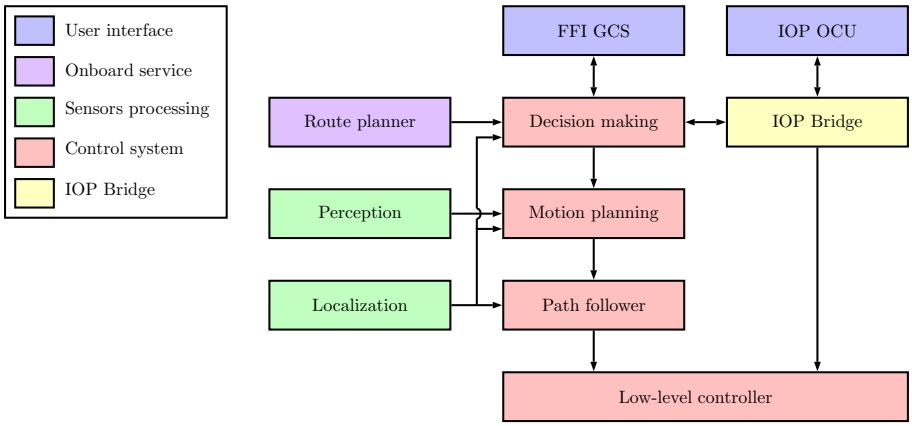


Figure 2.18 Software architecture of Tor *UGV.*

electric drive. All the necessary components needed to drive the vehicle are placed inside the tracked modules. Also, the tracked modules are similar in construction, the difference being in the components inside the tracked modules. One module contains a diesel generator and the other module houses the battery pack.

The vehicle is capable of moving on rough terrain (e.g. soft soil, snow, sand). In addition, regarding terrain properties, the vehicle is able to cross ditches and climb up to a 31° incline. The THeMIS moving speed is up to 20 km/h (12.4 mph). Driving time for the THeMIS is approximately 0.5–1.5 hours on batteries and 10 – 12 hours using the diesel generator. Driving time depends on the load and terrain.

Tor was modified and equipped to be used for autonomy experimentation. Figure 2.19 shows a picture of Tor with the sensor platform attached. Siv was used as a platform for the remote Sentry sensor package as shown in Figure 2.5.

Tor and Siv both had a sensor rig with a vibration dampened sensor platform mounted during LandX20. See Figure 2.20 for details on Tor's sensor rig used for autonomous operation. Siv had no other modifications, apart from the aforementioned sensor platform, but Tor has been modified to a degree. Tor's modifications in addition to sensor rig and platform includes, but are not limited to:

- Changed internal network 100 Megabit Ethernet switchgear to Gigabit Ethernet switchgear
- Added autonomy control computer
- Added scene analysis computer
- Added time synchronization server
- Added Data Synchronization Unit (DSU)
- Made and mounted new lid that has room for the scene analysis computer
- Added WiFi access point and antennas
- Added Warning blinker light
- Added wireless emergency stop remote charger
- Modified left hand side vehicle interface module (VIM), adding connectors for:
  - Power output for control computer
  - Power output for warning blinker
  - Override input from wireless emergency stop remote charger

Figure 2.19 Tor with sensor platform.

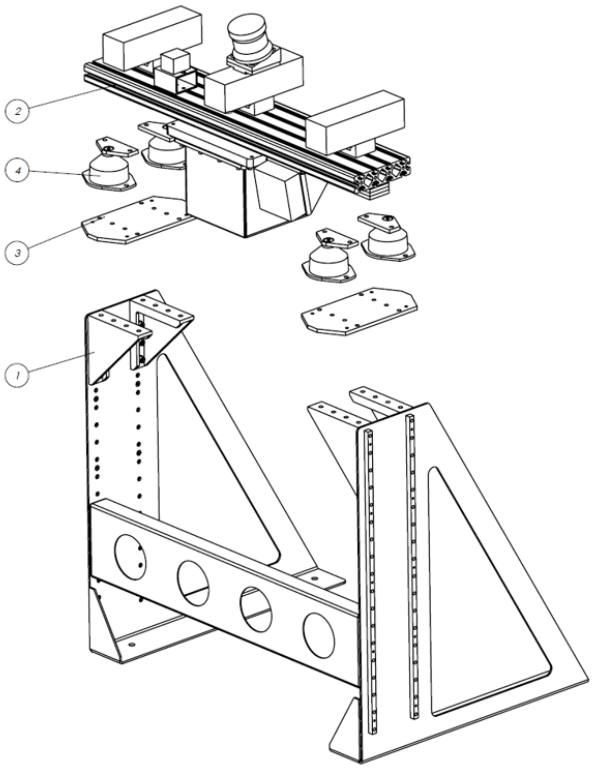


Figure 2.20 Exploded view of Tors sensor rig and platform with sensors

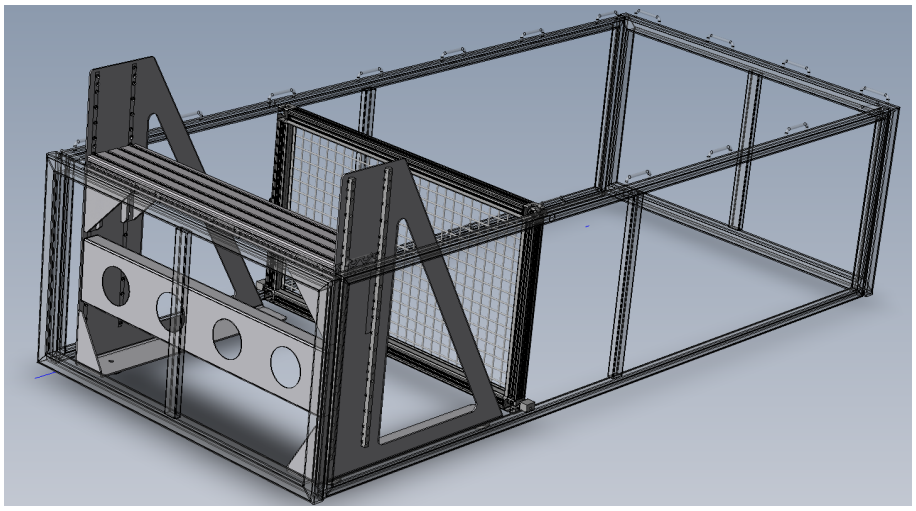


Figure 2.21 Image of the gate protecting sensors and initial design of sensor rig and platform

- Added GNSS antennas
- Modified lid over high voltage (96VDC) relays to ease access to manual relay override
- Made and added low voltage (28VDC) distribution with six 10A outlets, fed from THeMIS low voltage output LV1
- Made and added removable gate to protect sensor rig from potential other loose loads, e.g. soldier backpacks or munitions. See Figure 2.21 for details of gate and initial design of sensor rig and platform

2.4.2 Autonomy

The UGV will be deployed in light terrain environments and is expected to drive autonomously through both terrain and on roads. In order to achieve the necessary autonomous capabilities, a relatively advanced autonomy system is required.

For decisional autonomy the the high-level autonomy module HAL [17, 6] is used with the Ground Control Station (GCS), which is the operators interface to the UGV. The GCS is used to specify missions that are sent to HAL, which breaks them down into manageable parts. HAL also incorporates a route planner that can find a coarse route from the current position to the goal. Hence, HAL corresponds to the decision making module in Figure 2.18.

The route from HAL together with the maps from Warpath are used by a motion planner that plans the exact trajectories that the UGV follows. It is the motion planner that is responsible for avoiding collisions by planning trajectories that avoid the obstacles. The trajectories from the motion planner are then used by the path follower, which controls the vehicle so that it follows these trajectories as closely as possible.

High-level autonomy: HAL and the ground control station

In Figure 2.22 we see the interface of the ground control station. This is used to specify missions that can consist of several tasks. BML is used for sending the mission to the UGV [12]. For LandX20, the *Move* task was the only task used, and this task simply makes the UGV move to

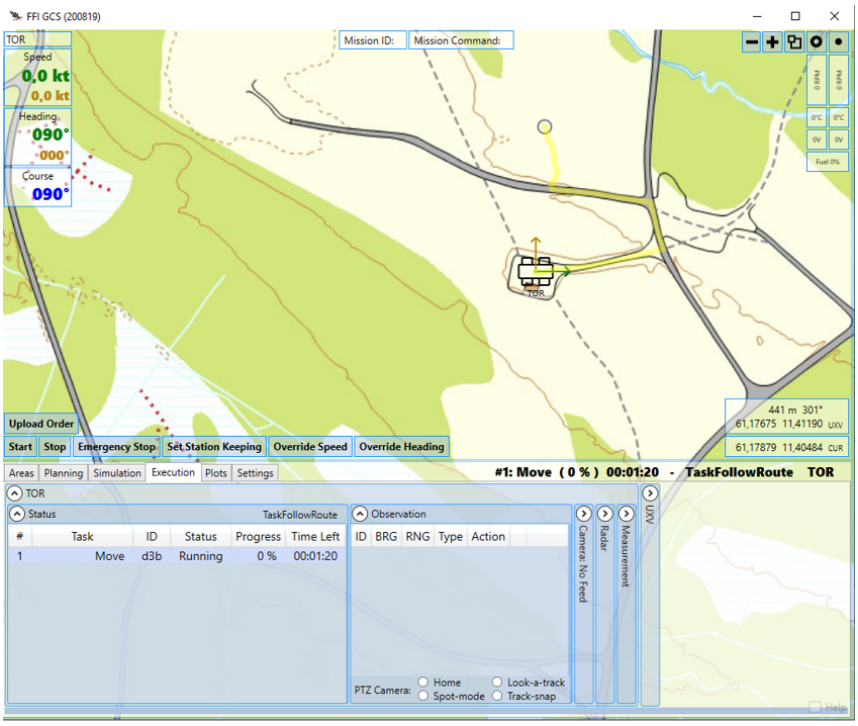


Figure 2.22 Ground control station interface for Tor.

a specified coordinate that is selected in the map. In this snapshot of the GCS, Tor can be seen executing a Move task where it attempts to drive to the goal marked with a circle.

When driving, Tor follows the route shown in yellow in the map. The route is obtained from HAL, which uses the route planning service Magellan [1] to plan a route. Magellan is a route planner that uses a-priori map data as a basis for planning. The terrain data are processed and used to create a graph that has information about how it is possible to move in the terrain. It also uses information about the road network, so that roads can be preferred as we see in Figure 2.22. In Figure 2.23 we see the underlying graph of the route planner Magellan. The mesh shows all the possible paths that can be taken through the terrain. The color represents the difficulty of traversing each segment, with red being the hardest, yellow being intermediate and green, which we see on the roads is the easiest to traverse. This allows the route planner to select a route that traverses the terrain with the lowest difficulty possible, which increases the chances of mission success.

Motion planner

The motion planner takes the route from HAL and the traversability maps from Warpath and use these to plan a collision free path that the UGV can follow. The motion planner only plans a short distance ahead towards a waypoint on the route from HAL, and is an adapted version from [16]. This path is then sent to the path follower which uses this to steer after. After a few seconds, the path is replanned based on the most recent traversability maps, and because Tor has driven past the previous position, the new path stretches past the previous one. This ensures that Tor is always following an up-to-date path, and Tor is never meant to follow a path in its entirety. Another reason for this receding horizon strategy is that the traversability maps are most accurate near Tor, and the further away from Tor the greater the chance of misclassification of the terrain in the traversability

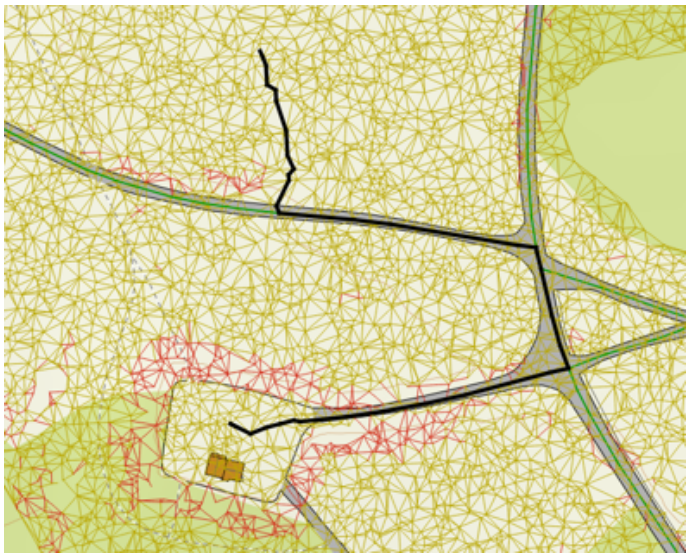


Figure 2.23 Route planning graph for Tor.

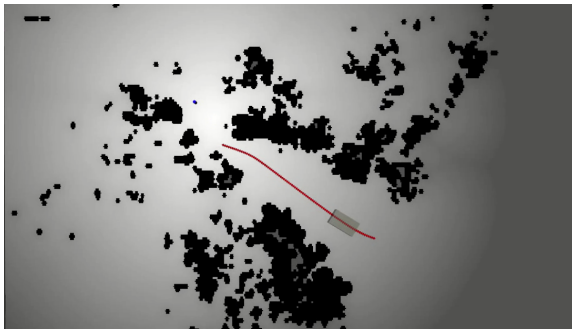


Figure 2.24 Motion planner

map.

An example of a motion planner running can be seen in Figure 2.24. This shows an obstacle map where the black areas are obstacles. The grey rectangle is the UGV and the red path is the path that the motion planner has planned and the UGV is following. However, as the UGV itself is under development, the individual autonomy modules has been tested separately but not together.

2.4.3 Perception and localization

The task of the perception component is to supply the rest of the system with an accurate representation of the world around the UGV. For this FFI's Warpath framework is used, as with the other unmanned systems. For a UGV one of the most important uses of the perceived world representation is in generating a map of driveable terrain for mobility purposes. The work is a further development of [4].

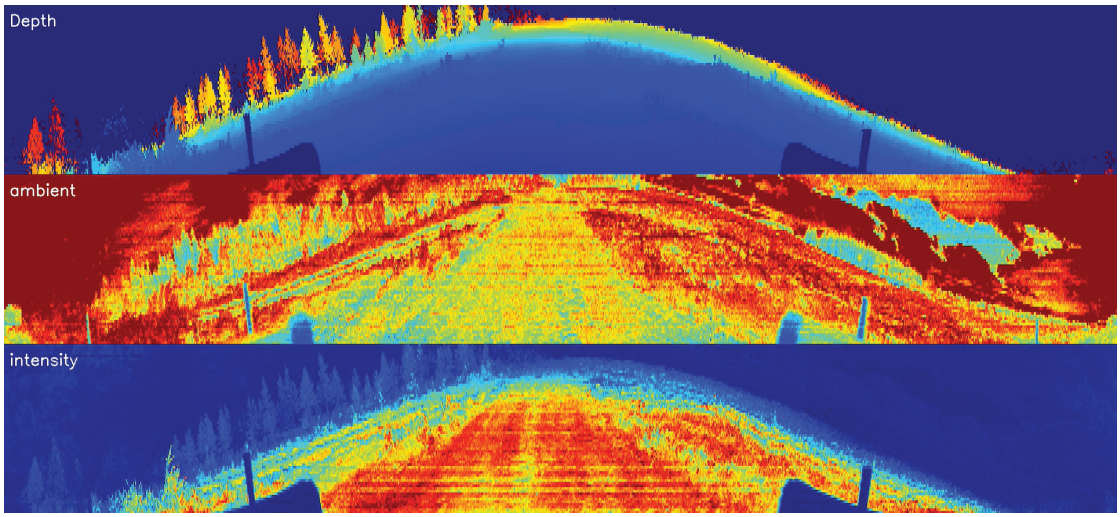


Figure 2.25 A LiDAR image from the trials at Rena with the three different image modes – depth at the top, ambient in the middle, and intensity at the bottom.

Sensors

The UGV carries several different sensors, seen on the sensor platform in Figure 2.19. These were chosen to complement each other well and together give the platform the information needed to create a good overview of its operating environment.

The Light Detection And Ranging (LiDAR) sensor provides an accurate 3-dimensional measurement of its surroundings. Only a sparse representation is generated, however, as the resolution is fairly limited. More dense measurements of the environment can be achieved by doing consecutive measurements while the UGV is moving. The sensor outputs three images – depth, signal-intensity and ambient, seen in Figure 2.25. The depth is the measured distance from the sensor to the object in the physical world. This is the traditional output of a LiDAR sensor, and is what is mainly used for our perception methods. The signal-intensity shows the return strength of the laser beam, and varies with factors like the surface material and texture for the imaged object, as well as the angle of arrival. This was not used actively during these trials, but can be used for segmentation, classification, as well as characterization tasks in the future. The sensor also outputs a measure of ambient light (mostly sunlight), which makes it able to act closer to a normal camera. It does, however, have a much larger aperture and shorter exposure time, which together with the spatial correlation and temporal matching with the other LiDAR images makes it very promising for providing new functionalities to the perception system.

The color camera is centered on the vehicle is pointing forwards. This sensor is used for visualization. Other sensor data or processed data is projected into to image, to visualize how Tor perceives the world. It only generates a two dimensional picture of the surroundings, and this is combined with other sensors to connect the information to specific areas in the three dimensional world.

A grayscale camera is mounted on each side of the vehicle, facing forwards. These take a higher number of pictures per second than the color camera, and is used for monochrome stereo vision. This can be used to construct a three dimensional measurement in front of the vehicle. It is less accurate than the LiDAR, but higher resolution can potentially give a more dense measurement of

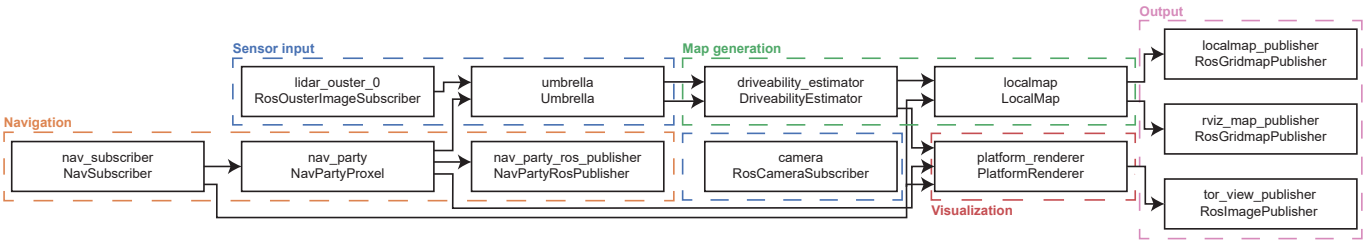


Figure 2.26 Processing graph for the experiments.

the surroundings.

The localization system consists of a a Honeywell HG9900 Inertial Measurement Unit (IMU) and a U-Blox Global Navigation Satellite Systems (GNSS) receiver, that are processed in the Inertial Navigation System (INS) software NavP. NavP is developed at FFI and is a real-time solution of NavLab [2], and it has been used for years on the FFI's Autonomous Underwater Vehicle (AUV) HUGIN [5]. The INS gives highly accurate measurements for the vehicle control and perception system at 300 Hz. The navigation solution is sent from the INS with UDP messages to a ROS node that redistributes the data as standard ROS messages onto the ROS network.

Algorithms and methods

The perception framework is called Warpath [4]. It processes sensor data using the processing framework Superflow[14], to produce traversability maps and visualization of the traversability map in the camera view. It uses the concept of *proxels*, which is our concurrent processing unit. The proxel can be connected to several other proxels, either by receiving output from other proxels or sending the processed results to other proxels. Figure 2.26 shows the processing architecture of the perception system. The graph can be split up into 5 groups: sensor input, navigation, map generation, visualization, and output.

There are separate proxels that takes input from the LiDAR and camera. These do the neccessary conversion to get the data on a format ready for further processing. Based on the LiDAR data and separate maps for the terrain and obstacles are generated.

Navigation data is received from the navigation system through a ROS subscriber before being processed in a navigation proxel. The proxel uses the absolute navigation data and stores them in a buffer for further use. The proxel also runs a smoothing algorithm on the data in the buffer. It supplies navigation and timing information to the rest of the system, and has an Application Programming Interface (API) that allows other proxels to ask for both absolute navigation data at different timestamps and relative poses between different timestamps.

The drivability estimator proxel takes the terrain and obstacle maps and generates a new map with drivability scores. This estimate is sent to the localmap proxel which stores and processes both the map and incoming requests for information from other proxels.

The platform renderer uses navigational data to project 3D information from the generated maps into a 2D image of the scene. This provides a convenient way for a human operator to both oversee the system during operation and debug during development.

The framework outputs both the maps and the visualization output so components outside Warpath can utilize them.

The main product generated by the perception is the map of the surroundings. This consist of an occupancy map, divided into a grid of adjustable granularity. Motion planning uses these maps

to calculate safe and efficient paths for the UGV to drive.

2.5 Passive RF sensors

The sensor type used in this experiment is called LINE 3 [15], which detects radar signals in the frequency band 9.3-9.5 GHz and determines their angle of arrival using a four-channel phase interferometer. The sensor consist mainly of three parts: the antennas, a four-channel receiver, and a processing unit, all contained in a box as shown in Figure 2.27

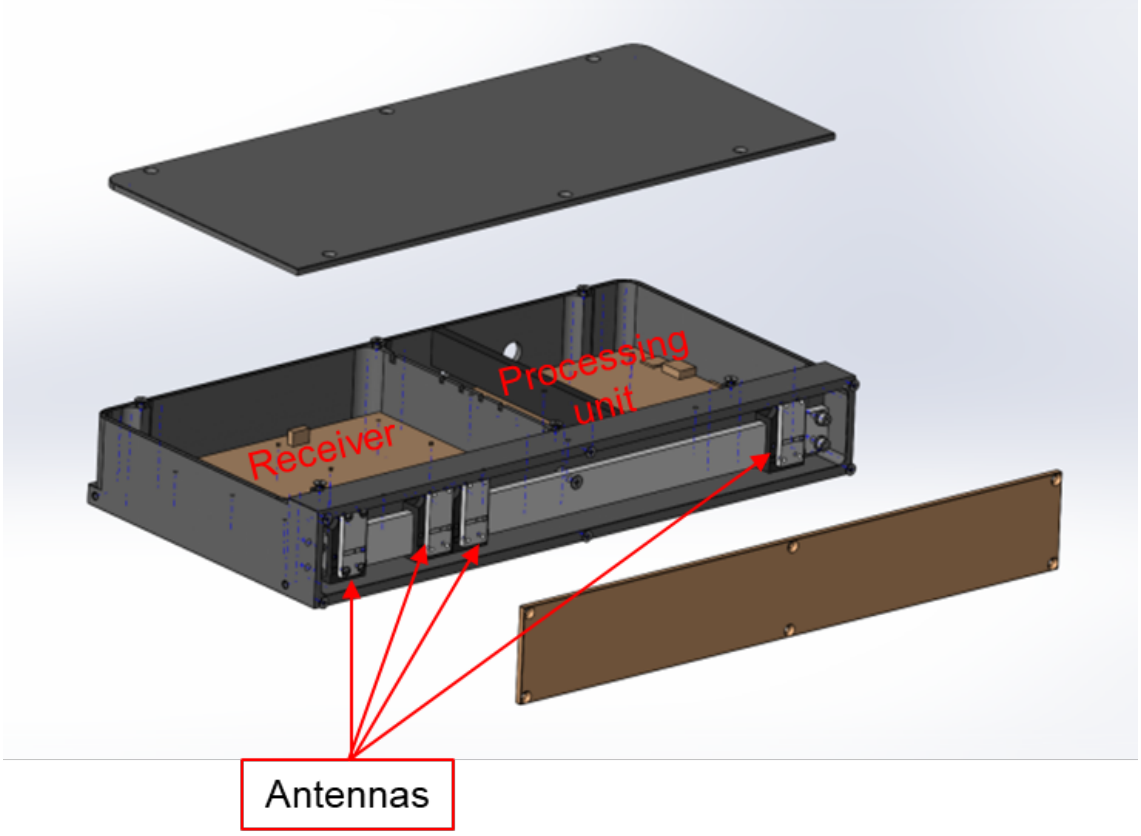


Figure 2.27 The casing for a LINE 3 passive RF sensor.

By combining the result from two (or more) sensors, one can perform near instantaneous cross-bearing geolocation, which means that you can determine the location of an emitter almost as soon as it emits by calculating where the lines of bearing from the different sensors intersect (see Figure 3.4).

3 Experiment

LandX20 planning and preparation was initiated in January 2020, and originally the experiment should have been held before the summer vacation. However, due to the Covid-19 outbreak the experiment was moved to September, where the week 31st of August to 4th of September was an integration week, and the week 14th to 18th of September was the experiment and demonstration week. The experiment was held at BT-banen at Rena Army Camp. This chapter describes the activities during these two weeks.

3.1 Sentry, Valkyrie and user interface

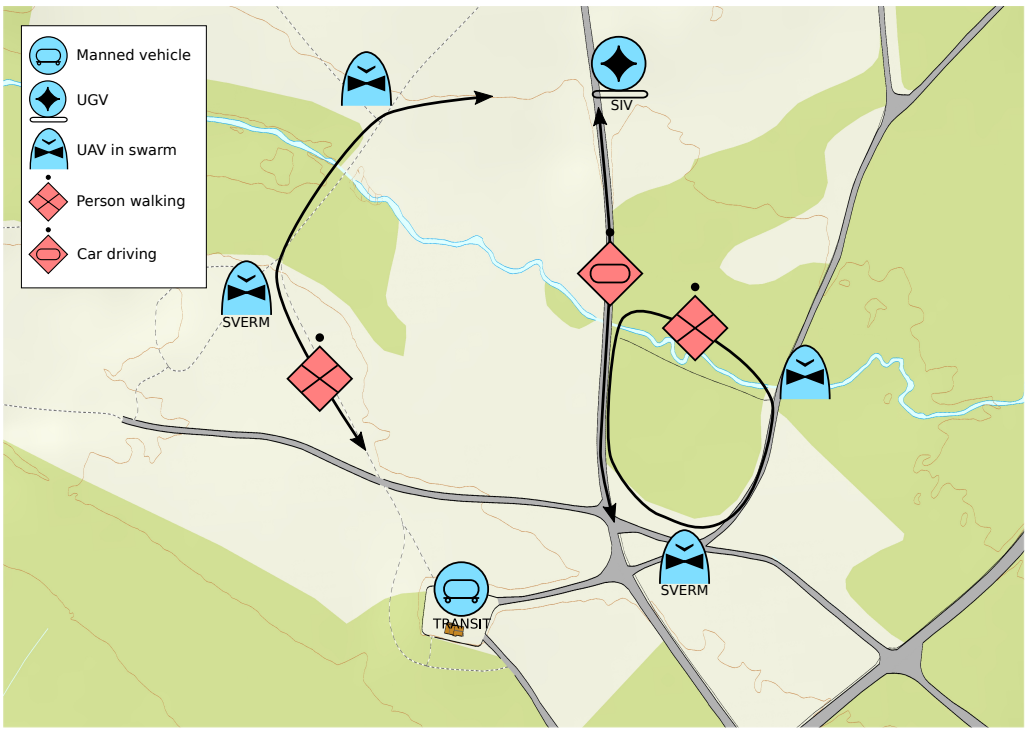
This section explains the experiments conducted to demonstrate and test the systems for situational awareness, namely the Sentries, Valkyrie and the common user interface. The section will primarily be a discussion on the choice of approach and the team's experience with the outcome along with suggested improvements for both the approach and systems.

3.1.1 Scenario

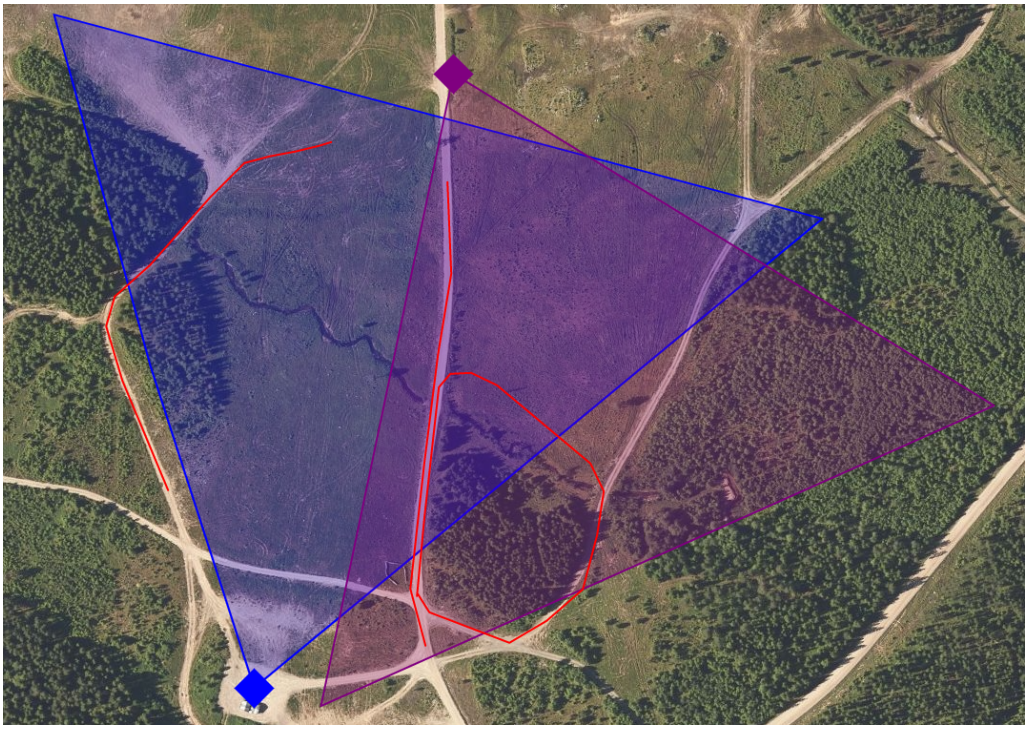
The original scenario for LandX20 was planned to resemble a scenario relevant for the armed forces with an attacking force approaching the base as explained in detail in Section 1.1. The plan was for the Situational Awareness (SA)-system to provide full red-force tracking and thereby ease the decision making for the defending force. This type of scenario would, however, last for an extended period, possibly close to 1 hour. It would require the SA system to function flawlessly for the entire scenario, while only being useful for a very short period of time, which is not very appropriate for a demonstration of SA-capabilities.

The scenario was therefore switched to a scenario more suitable for demonstrating SA-capabilities. In this scenario there were three targets (two persons and one vehicle) continuously moving in the field. The trajectories of the targets were chosen so that they would move in and out of the areas of observation for the platforms. The scenario and the positions and observation directions of the two Sentry platforms are shown in Figure 3.1. A Sentry (TRANSIT) was used as the main observing platform for the base. To cover the large area behind the wooded area to the east, a forward observing Sentry (SIV) was positioned to surveil this area. The UAVs (SVERM) were tasked to observe the areas that could not be covered by the Sentry platforms. When positioning the sensors it was clear that achieving line of sight to all the relevant positions on the field was challenging, even though operational area is a fairly flat and large area. The UAVs are therefore a highly valuable asset, by providing coverage that was not possible to achieve with the larger platforms on the ground.

The target to the west a walking person, and moves in and out of a wooded area, essentially outside the observation area of both the Sentry platforms. This target is primarily observed by two UAVs. The target vehicle is a civilian car, and it is observed by both Sentry platforms throughout it's entire range of movement. The person to the east walks around a wooded area, and moves in and out of the observation area of both the Sentry platforms. Two UAVs are therefore also tasked to observe the target when it is outside the surveillance region of both the Sentry platforms.



(i) Scenario used at the demonstration.



(ii) Area observed by the Sentries during LandX20.

Figure 3.1 Figure (i) shows the scenario used at the demonstration and Figure (ii) shows the position of the sensor platforms and their observation direction.

Table 3.1 Flight activity during LandX20

Date	Sorties	Hours flight
2. sept	7	3
3. sept	16	3
4. sept	20	3
14. sept	7	3
15. sept	20	3
16. sept	20	7
17. sept	20	8
Totalt	110	39

3.1.2 Tracking

Tracking objects continuously using multiple sensors at multiple positions and domains is a challenging task, but also a valuable capability. It makes the system capable of tracking more diverse targets and to maintain a unique track ID for longer periods of time. When such a system works flawlessly, the operator should not necessarily be aware of which platform is used to track the target, rather then the fact that the target is indeed being tracked.

During the experiments, the tracking performance was good. The sensor systems quickly acquired track on new objects and kept a unique track ID over long periods of time.

3.1.3 User interface

Creating an efficient user interface for these types of high-capability situational awareness systems is a balancing act. A goal is to present a lot of valuable information to an operator, while avoiding *information overload*. A typical scenario of information overload would be if we presented raw data from all the sensors to the operator, which would result in too many video streams to monitor for the operator. The fusion and multi-sensor tracking therefore functions as a filtering operation to extract the most relevant information. The task of the user interface is to present this to an operator in an efficient and convenient way.

LandX20 was the first presentation of the new 3D user interface developed at FFI. Tracks from all the platforms were presented in a high-fidelity local map in three dimensions. At the same time the operator could see where all the observation platforms were positioned and where they were heading. The operator could choose a tracked target in the map and choose to focus a PTZ camera on that target. A PTZ camera in the designated Sentry would then slew to the target and automatically follow the target.

3.1.4 UAV flights

Three different UAV systems was flown in the experiment period. The Flamingo UAV was the main platform with regards to hours flown, but also a DJI Phantom and a DJI Mavic was operated in the period. UAVs was flown Tuesday, Wednesday and Thursday week 36 with the call sign “Fuji unmanned” 1, 2, 3, and Monday to Thursday week 28 under the call sign “Casio unmanned” 1, 2, 3. See Table 3.1 for an overview of flight activity.

There is a relative high flight activity, both manned and unmanned, in area 105 at Rena. This means that it was highly likely that we need to postpone our flights awaiting air space. This is not a

Figure 3.2 Three Flamingo UAVs on a joint mission

Figure 3.3 Picture from LandX20 that shows three Flamingo UAVs performing simultaneous take-off during an experiment.

problem during long test sessions, where we are flexible with regards to flight times, but this can be a serious problem when it comes to demonstrations, where we are not guarantied air space at certain times. We did not have any problems with this during our demonstration.

The Flamingo UAV were flown in both windy conditions and rain. Figure 3.2 shows three airborne Flamingos conducting a joint mission, and Figure 3.3 simultaneous take-off.

3.2 UGV

The main goal with regards to the UGV Tor was to get it to a functional level equal of the previous UGV, Olav [9]. This goal was unfortunately not met, mainly due to hardware problems and not being able to complete the needed software integration and adaption.

3.3 Passive RF sensors

The goal of this part of the experiment was to demonstrate the use of passive RF sensors to contribute to the situational awareness. Passive RF sensors are capable of detecting, classifying and locating electromagnetic emitters; anything from handheld radios to air surveillance radars. For a passive RF sensor to be able to do this, it needs to cover a large radio bandwidth, ideally from the VHF band, i.e. from about 100 MHz, to perhaps 40 GHz. In this experiment, we used a passive RF sensor designed for maritime surveillance and only capable of listening to the upper maritime navigation band, 9.3-9.5 GHz, for the simple reason that it was what we had available. This again meant we had to use test emitters operating in that band, so we used two maritime radars as emitters. This limited capability was sufficient to demonstrate the operative potential of passive RF sensors in this setting.

We used two LINE 3 passive RF sensors originally made for maritime surveillance, and two maritime navigation radars as target emitters. In our first configuration, one LINE 3 sensor was mounted on an UGV that remained stationary throughout this experiment, and the other one was mounted on a tri-pod. The target radars were mounted on tri-pods and placed within line of sight of both sensors. In the second configuration we placed both LINE 3 sensors on tri-pods to achieve better geometric coverage of the area within line of sight, and one target emitter was mounted on the roof of a van to play the role of a moving, emitting target.

In both cases, the sensors were able to provide reliable classification of the target emitters, and reliably gave lines of bearing that matched with where the emitters were actually located. Visual inspection in the Geographic Information System (GIS) map interface showed that the bearings from the two sensors matched the known positions of the target emitters.

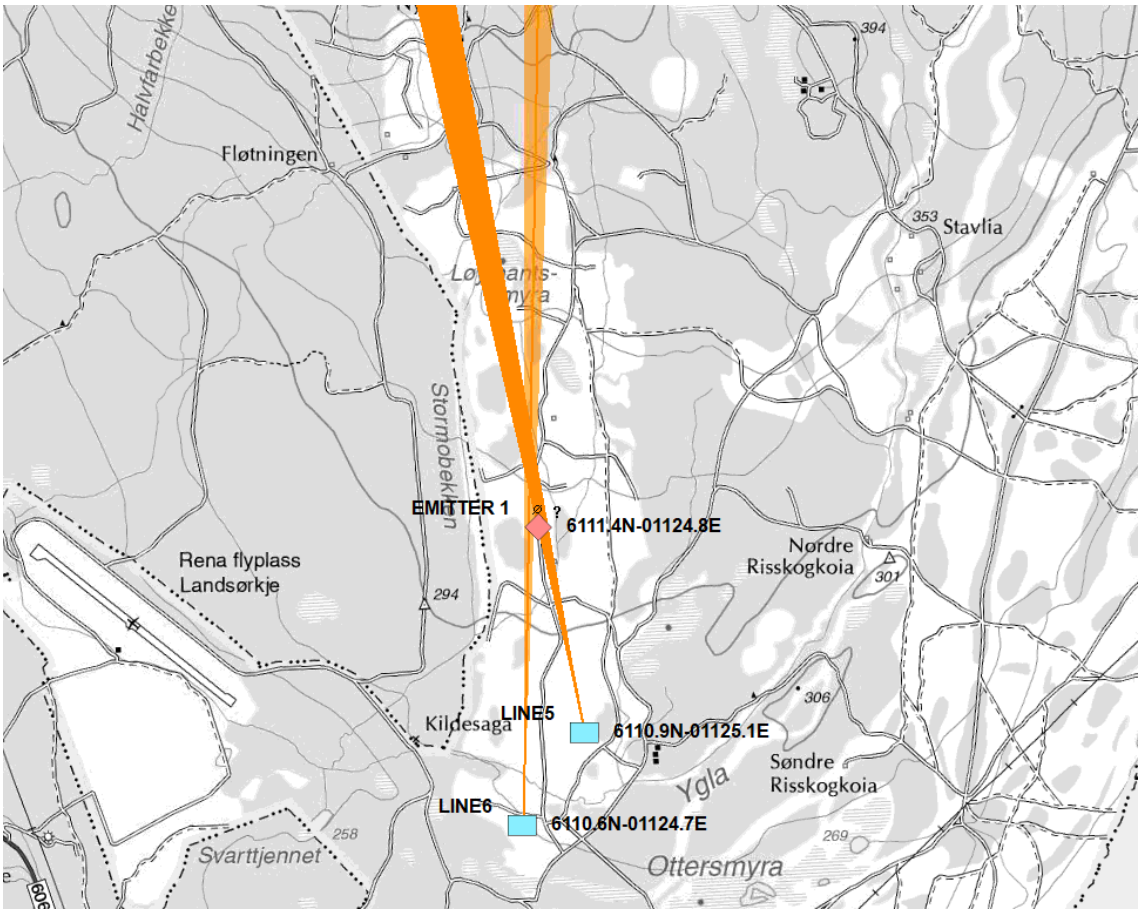


Figure 3.4 Simple cross-bearing geolocation of single emitter with two passive RF sensors.

4 Conclusion and future work

The planning and preparation for the experiment was hampered by the Covid-19 pandemic, and this resulted in moving the experiment from before the summer vacation to September. As the pandemic and home office hampered the development, this gave us some additional time in preparing for the experiment.

The event itself was successful. The system using two Sentries and four Flamingo drones, to obtain the situational awareness presented in the Cesium based GUI worked as intended. Also, the localization using passive RF sensors was successfully demonstrated. There were also many visitors on during the experiment. Unfortunately, mostly due to hardware problems, the Tor UGV was not able to drive autonomously.

The systems will continue to be developed, and we plan to have a new LandX experiment in 2021.

References

[1] Solveig Bruvoll, Knut Landmark, Brage G. Eikanger, Espen Messel, Marius Thoresen, and Martin S. Wiig. Systembeskrivelse av ruteplanleggingstjenesten Magellan. FFI-Report 19/00225, Norwegian Defence Research Establishment, Kjeller, 2019. Exempt from public disclosure, cf. Freedom of Information Act §21.

[2] Kenneth Gade. NavLab, a generic simulation and post-processing tool for navigation. *Modeling, Identification and Control: A Norwegian Research Bulletin*, 26(3):135–150, 2005. URL: https://doi.org/10.4173%2Fmic.2005.3.2, doi:10.4173/mic.2005.3.2.

[3] Daniel Gusland, Jonas Myhre Christiansen, Kyrre Strøm, Jabran Akhtar, and Børge Torvik. Radar for C-UAS application. FFI-Report 20/00978, Norwegian Defence Research Establishment, 2020. Exempt from public disclosure, cf. Freedom of Information Act §13.

[4] Trym Vegard Haavardsholm, Ragnar Smestad, Martin Vonheim Larsen, Marius Thoresen, and Idar Dyrdal. Scene Understanding for Autonomous Steering. In *Proceedings of IST-127/RSM-003 Specialists' Meeting in Intelligence & Autonomy in Robotics*, Bonn, Germany, Oct. 2016. NATO Science and Technology Organization. NATO UNCLASSIFIED.

[5] Bjørn Jalving, Kenneth Gade, Ove Kent Hagen, and Karstein Vestgard. A toolbox of aiding techniques for the HUGIN AUV integrated inertial navigation system. In *Oceans 2003. Celebrating the Past ... Teaming Toward the Future (IEEE Cat. No.03CH37492)*. IEEE, 2003. URL: https://doi.org/10.1109%2Foceans.2003.178505, doi:10.1109/oceans.2003.178505.

[6] Thomas Røbekk Krogstad, Kim Mathiassen, Else-Line Malene Ruud, Rikke Amilde Seehuus, Aleksander Skjerlie Simonsen, and Martin Syre Wiig. HAL - A decisional autonomy module for unmanned systems. FFI-report 19/00489, Norwegian Defence Research Establishment, 2019. Exempt from public disclosure, cf. Freedom of Information Act §21.

[7] Martin Vonheim Larsen, Jonas Trondsen, Gorm Krogh Selj, and Olav Hegdal. System for automatisert områdeovervåking - eksperimentrapport for EP 1864. FFI-Report 20/01402, Norwegian Defence Research Establishment, 2020. RESTRICED, The Norwegian Security Act §§ 5-3 and 5-4.

[8] Kim Mathiassen, Magnus Baksaas, Sindre Aas Græe, Eilert André Mentzoni, and Niels Hygum Nielsen. Making the milrem themis UGV ready for autonomous operations. In Paul L. Muench, Hoa G. Nguyen, and Brian K. Skibba, editors, *Unmanned Systems Technology XXIII*. SPIE, apr 2021. doi:10.1117/12.2585879.

[9] Kim Mathiassen, Magnus Baksaas, Lars Erik Olsen, Marius Thoresen, and Bjørn Tveit. Development of an autonomous off-road vehicle for surveillance missions. In *Proceedings of IST-127/RSM-003 Specialists' Meeting on Intelligence & Autonomy in Robotics*, Bonn, Germany, oct 2016. NATO Science and Technology Organization. NATO UNCLASSIFIED.

[10] Kim Mathiassen, Frank E. Schneider, Paul Bounker, Alexander Tiderko, Geert De Cubber, Magnus Baksaas, Jakub Główka, Rafał Kozik, Thomas Nussbaumer, Juha Röning, Johannes Pellenz, and André Volk. Demonstrating interoperability between unmanned ground systems and command and control systems. *International Journal Intelligent Defence Support Systems*, 6(2), 2021. doi:10.1504/IJIDSS.2021.115236.

[11] Olav Rune Nummedal. Flamingo - a UAV for autonomy research. FFI-Report 21/00318, Norwegian Defence Research Establishment, 2021. URL: http://hdl.handle.net/20.500.12242/2839.

[12] Rikke Amilde Seehuus, Kim Mathiassen, Else-Line Malene Ruud, Aleksander Skjerlie Simonsen, and Fredrik Hermansen. Battle management language for robotic systems. In *International Conference on Modelling and Simulation for Autonomous Systems (MESAS 2018)*, pages 302–320. Springer International Publishing, 2018. doi:10.1007/978-3-030-14984-0_23.

[13] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics - Modeling, planning and control*. Springer London, 2009. doi:10.1007/978-1-84628-642-1.

[14] Ragnar Smestad, Martin Vonheim Larsen, and Trym Vegard Haavardsholm. Superflow - an efficient processing framework for modern C++. FFI-Report 19/00776, Norwegian Defence Research Establishment, Kjeller, 2019. Exempt from public disclosure, cf. Freedom of Information Act §21.

[15] Tore Smestad, Robert Macdonald, Eivind Bergh Nilssen, Eirik Grimstvedt, and Fredrik Gulbrandsen. LINE prototypes towards applications for maritime surveillance and navigation aids. FFI-Report 19/01064, Norwegian Defence Research Establishment, Kjeller, 2019.

[16] Marius Thoresen, Niels Hygum Nielsen, Kim Mathiassen, and Kristin Y. Pettersen. Path Planning for UGVs based on Traversability Hybrid A*. *IEEE Robotics and Automation Letters*, 2021. doi:10.1109/lra.2021.3056028.

[17] Martin S. Wiig, Rikke A. Løvlid, Kim Mathiassen, and Thomas R. Krogstad. Decision autonomy for unmanned vehicles. In *Proceedings of IST-127/RSM-003 Specialists' Meeting on Intelligence & Autonomy in Robotics*, Bonn, Germany, oct 2016. NATO Science and Technology Organization. NATO UNCLASSIFIED.

[18] Einar Østevold, Jonas Myhre Christiansen, Idar Dyrdal, Erlend Finden, Daniel Gusland, Sigmund Rolfsjord, Thomas Thoresen, and Tore Ulversøy. Anti-dronesystem (CUAS) – vurderinger av sensorer og sensormiks. FFI-Report 19/00258, Norwegian Defence Research Establishment, 2020. Exempt from public disclosure, cf. Freedom of Information Act §21.

[19] Einar Østevold, Idar Dyrdal, Daniel Gusland, Marius Halsør, Martin Vonheim Larsen, and Sigmund Rolfsjord. Antidrone-system, CUAS, basert på RWS – hovedrapport for prosjekt 1494. FFI-Report 20/01020, Norwegian Defence Research Establishment, 2020. Exempt from public disclosure, cf. Freedom of Information Act §21.

Acronymes

AGC Automated Gain Control
API Application Programming Interface
AR Augmented Reality
AUV Autonomous Underwater Vehicle
BML Battle Management Language
C2 Command and Control
C-BML Coalition Battle-Management-Language
C-UAS Counter Unmanned Aerial System
DNN Deep Neural Network
FFI Norwegian Defence Research Establishment
FOV Field of View
GCS Ground Control Station
GIS Geographic Information System
GNSS Global Navigation Satellite Systems
GPS Global Positioning System
GUI Graphical User Interface
HAL Hybrid Autonomy Layer
IMU Inertial Measurement Unit
INS Inertial Navigation System
ISR Intelligence, Surveillance and Reconnaissance
LiDAR Light Detection And Ranging
LOI LuftOperativt Inspektorat
NARFA National Allied Radio Frequency Agency
NTP Network Time Protocol
RF Radio Frequency
ROS Robot Operating System
PTZ Pan-Tilt-Zoom
RTS Real-Time Strategy
SA Situational Awareness
UAV Unmanned Aerial Vehicle
UGV Unmanned Ground Vehicle
UI User Interface
VHF Very High Frequency
WS Web Sockets

About FFI

The Norwegian Defence Research Establishment (FFI) was founded 11th of April 1946. It is organised as an administrative agency subordinate to the Ministry of Defence.

FFI's mission

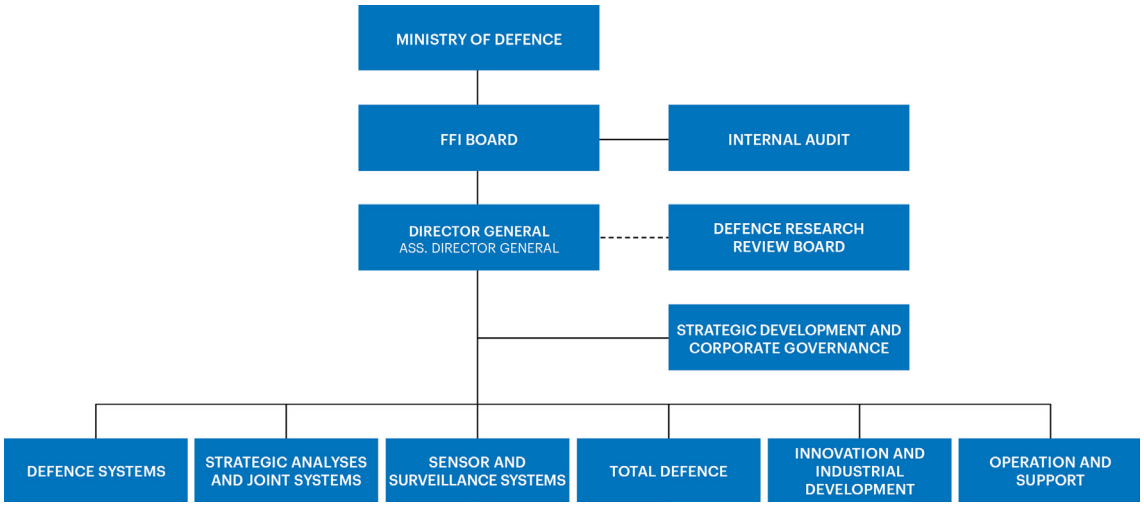
FFI is the prime institution responsible for defence related research in Norway. Its principal mission is to carry out research and development to meet the requirements of the Armed Forces. FFI has the role of chief adviser to the political and military leadership. In particular, the institute shall focus on aspects of the development in science and technology that can influence our security policy or defence planning.

FFI's vision

FFI turns knowledge and ideas into an efficient defence.

FFI's characteristics

Creative, daring, broad-minded and responsible.



Forsvarets forskningsinstitutt
Postboks 25
2027 Kjeller

Besøksadresse:
Instituttveien 20
2007 Kjeller

Telefon: 63 80 70 00
Telefaks: 63 80 71 15
Epost: post@ffi.no

Norwegian Defence Research Establishment (FFI)
P.O. Box 25
NO-2027 Kjeller

Office address:
Instituttveien 20
N-2007 Kjeller

Telephone: +47 63 80 70 00
Telefax: +47 63 80 71 15
Email: post@ffi.no