Design Specification
Search and Rescue - Land

Version 1.0

Author: Rickard Wretlind
Date: October 17, 2022

Status

Reviewed	Jakob Åslund	2022-10-17
Approved	Jakob Åslund	2022-10-17

Course name:	Reglerteknisk projektkurs, CDIO	E-mail:	oplandlubber@gmail.com
Project group:	OWL	Document responsible:	Rickard Wretlind
Course code:	TSRT10	Author's E-mail:	ricwr413@student.liu.se
Project:	Search and Rescue - Land	Document name:	Design_Specification.pdf

Project Identity

Group E-mail:	oplandlubber@gmail.com
Homepage:	https://tsrt10.gitlab-pages.liu.se/2022/sbd/
Orderer:	Jakob Åslund, Linköping University
	E-mail: jakob.aslund@liu.se
Customer:	Torbjörn Crona, Saab Dynamics
	E-mail: torbjorn.crona@saabgroup.com
Course Responsible:	Daniel Axehill, Linköping University
	E-mail: daniel.axehill@liu.se
Advisors:	Anja Hellander, Linköping University
	E-mail: anja.hellander@liu.se
	Linus Wiik, Saab Dynamics
	E-mail: linus.wiik@saabgroup.com
	Joel Wikner, Saab Dynamics
	E-mail: joel.wikner@saabgroup.com
	Åke Johansson, Saab Dynamics
	E-mail: ake.johansson1@saabgroup.com

Group Members

Initial	Name	Responsibility	E-mail (@student.liu.se)
DG	Daniel Goderik	Project Manager	dango893
AL	Anton Larsson	Head of Design	antla594
DS	Daniel Sandvall	Head of Hardware - Drone	dansa201
SF	Sebastian Fagerstedt	Head of Testing	sebfa953
ES	Eric Sevonius	Head of Software	erise263
RW	Rickard Wretlind	Head of Documentation	ricwr413
AW	Albin Westlund	Head of Hardware - Rover	albwe662

Document History

Version	Date	Changes made	Sign	Reviewer
0.1	2022-09-29	First draft	RW, AL, ES	Jakob Åslund
0.2	2022-10-07	First revision	DG	Jakob Åslund
0.3	2022-10-15	Second revision	RW	Jakob Åslund
0.4	2022-10-15	Third revision	ES, SF	Jakob Åslund
1.0	2022-10-17	First version	RW	Jakob Åslund

Contents

1 Introduction 1

1.1 Parties 1

1.2 Purpose and goal 1

1.3 Usage 1

1.4 Background Information 1

1.5 Definitions 2

2 System Overview 3

2.1 Subsystems 3

2.2 Mission 4

2.3 Communication 5

3 Base Station 6

3.1 Discovery and Search Algorithm 6

3.1.1 Phase one 6

3.1.2 Phase two 6

3.2 Task Planner 10

3.2.1 Behaviors 10

3.3 GUI 11

4 Rover 12

4.1 Hardware 12

4.2 Odometry 13

4.3 SLAM 15

4.3.1 LIDAR 16

4.3.2 Robot Localization 16

4.3.3 Occupancy Grid-Representation 16

4.4 Navigation 16

4.4.1 Motion Planner 17

4.4.2 Vehicle Motion Control 17

4.4.3 Motor Controller 17

4.5 Tracking 17

4.5.1 Recognition 17

4.5.2 Tracking Algorithm 18

4.6 Modes of Operation 21

4.6.1 Manual Mode 21

4.6.2 Autonomous Mode 21

5 UAV 22

5.1 Hardware 22

5.1.1 Hardware Components 22

5.1.2 Safety Features 23

5.2 Software 23

5.2.1 Onboard Communication 23

5.2.2 Position Command 23

5.2.3 Position Feedback 23

5.2.4 Recognition and Tracking Using the Camera 24
5.2.5 Motion Planner 24
5.2.6 Take-off and Landing Services 25
5.2.7 Deliver Supplies 25
5.3 Modes of Operation 25
5.3.1 Manual Mode 25
5.3.2 Autonomous Mode 26

6 Simulation **27**
6.1 Gazebo11 27
6.2 Simulation of the Robots 27
6.2.1 Rover 27
6.2.2 UAV 27
6.3 Simulation of the Map 28
6.4 Simulation of Distressed Persons 28

1 Introduction

This document is the design specification for the Search and Rescue - Land project, in the course TSRT10 Automatic Control - Project Course. It presents the specifications for the design of the system.

1.1 Parties

There are principally two different parties involved in this project: Linköping University and SAAB Dynamics. In the context of the course, the SAAB Dynamics party acts as customers interested in a Search and Rescue-platform, and the University party acts as a company tasked with delivering the product.

The party from Linköping university consists of: Anja Hellander acting as advisor, Jakob Åslund acting as orderer, and the project group which consists of seven master students. Furthermore, the party from SAAB Dynamics consists of: Torbjörn Crona fulfilling the role of customer, and Linus Wiik, Joel Wikner and Åke Johansson acting as additional advisors.

1.2 Purpose and goal

The main purpose of this project is to develop a Search and Rescue-system consisting of an unmanned ground vehicle, called the Rover, and an Unmanned Aerial Vehicle (UAV) collaborating to identify, track and supply people in distress. To accomplish this, the system should be able to map and navigate an area with people in distress. Specifically, the Rover should use a LIDAR-sensor to map the environment, whilst, using cameras, collaboratively searching with the UAV for distressed people.

Thus, the goal is to deliver a system according to the aforementioned specifications. The system should also follow design guidelines such as adhering to Google's code standard, developing the system in ROS2 and extending the docker integration of the current project.

1.3 Usage

As mentioned above, the intended use case of the product is deployment in a complex environment with distressed people to save. However, the finished product will only function in an environment as described in "Project_Plan.pdf"[8].

1.4 Background Information

To search for and rescue distressed people by foot can be dangerous and time-consuming. In order to increase the safety and and improve the chance of finding and rescuing distressed person, robots, such as rovers and UAV:s, can be used instead.

This project has been developed for many years at Linköping university and has had different development goals through the years. Today the system consists of one UAV and one Rover which are equipped with different sensors, cameras, and LIDAR which makes it possible to map the environment and creating motion plans in search and rescue missions.

The project has a sister project called "Search and Rescue – Underwater". The cooperation between the two project is slight, while there are visions to unify the project in the distant future.

1.5 Definitions

Below some definitions and acronyms are explained which are recurring in this document.

- **Rover** - Tracked vehicle driving autonomously that maps the test area and seeks distressed persons.
- **UAV** - A quadcopter flying autonomously and seeks distressed persons.
- **Agent** - participant in a mission, Rover and/or UAV.
- **Base Station** - A computer that handles the information from the Rover and UAV.
- **Distressed person** - In simulation, this is a virtual marker that should be found by the Rover and UAV. When doing real tests, this will be RC-cars colored with bright colors.
- **SLAM** - Simultaneous Localization and Mapping.
- **LIDAR** - Light Detection and Ranging.
- **SIL** - Software In The Loop.
- **Qualisys** - Sensor system in the room Visionen that uses cameras and reflective targets to deliver position data.
- **ROS2** - "Robot Operating System", Framework for robot software development.
- **No-fly zone** - A zone where the UAV is restricted from flying into.
- **PDDL** - Planning Domain Definition Language.
- **RPi** - Raspberry Pi.
- **Pixhawk** - The flight controller *Pixhawk 4* that is mounted on the UAV.
- **HW** - Hardware.
- **SW** - Software.
- **Rviz2** - A visualization manager that displays the generated map and agent positions during the mission.
- **Gazebo** - Simulation environment.
- **RC-car** - Small RC-cars controlled by the user, that are used to simulate distressed persons.

2 System Overview

This section aims to give an overview of the entire system. The system utilizes SIL, which means it can either be run on hardware or in simulation without subsystems knowing which mode it is run on. Furthermore, the system is divided into different subsystems and three of those subsystems, if not run in simulation mode, are hardware based. Those subsystems include Base Station, Rover and UAV. A schematic overview of those can be seen in Figure 1.

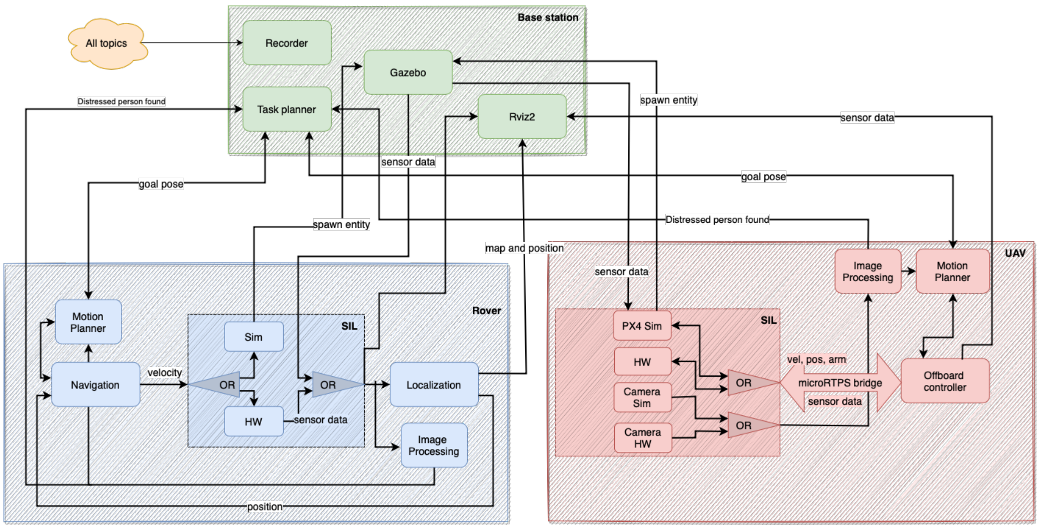

Figure 1: An overview of the entire system with Base Station, UAV and Rover.

The idea for the system is to use the Base Station as an intermediator for communication between the Rover and UAV. The Base Station handles the communication via a network, and coordinates the Rover and UAV with a task planner. Both the Rover and UAV shall receive a task from the Base Station and be able to plan how to execute the task itself, using predefined behaviors. Briefly explained, the Rover should be able to use SLAM in order to map and localize itself with the use of a LIDAR, as well as track distressed people with a camera. The UAV will however only be able to track people with a camera while also being able to deliver supplies.

Not visualized in Figure 1 is that the system will use Qualisys implemented in the arena Visionen. Qualisys will provide an estimated position (similar to a GPS) for the Rover and UAV.

2.1 Subsystems

The system is divided into the following subsystems:

- Base Station
- Rover
- UAV
- Simulation Environment

2.2 Mission

Briefly summarized, the mission is to search an unknown area with the Rover and UAV until all distressed persons are found. At first the Rover will map the area while searching for distressed persons and the UAV will search for distressed persons. When the area is mapped, the Rover and UAV will search different parts of the area simultaneously. When, and if, a distressed person is found, the Rover will start to track the distressed person while the UAV will collect and deliver supplies. When all distressed persons are found, the Base Station will signal mission complete.

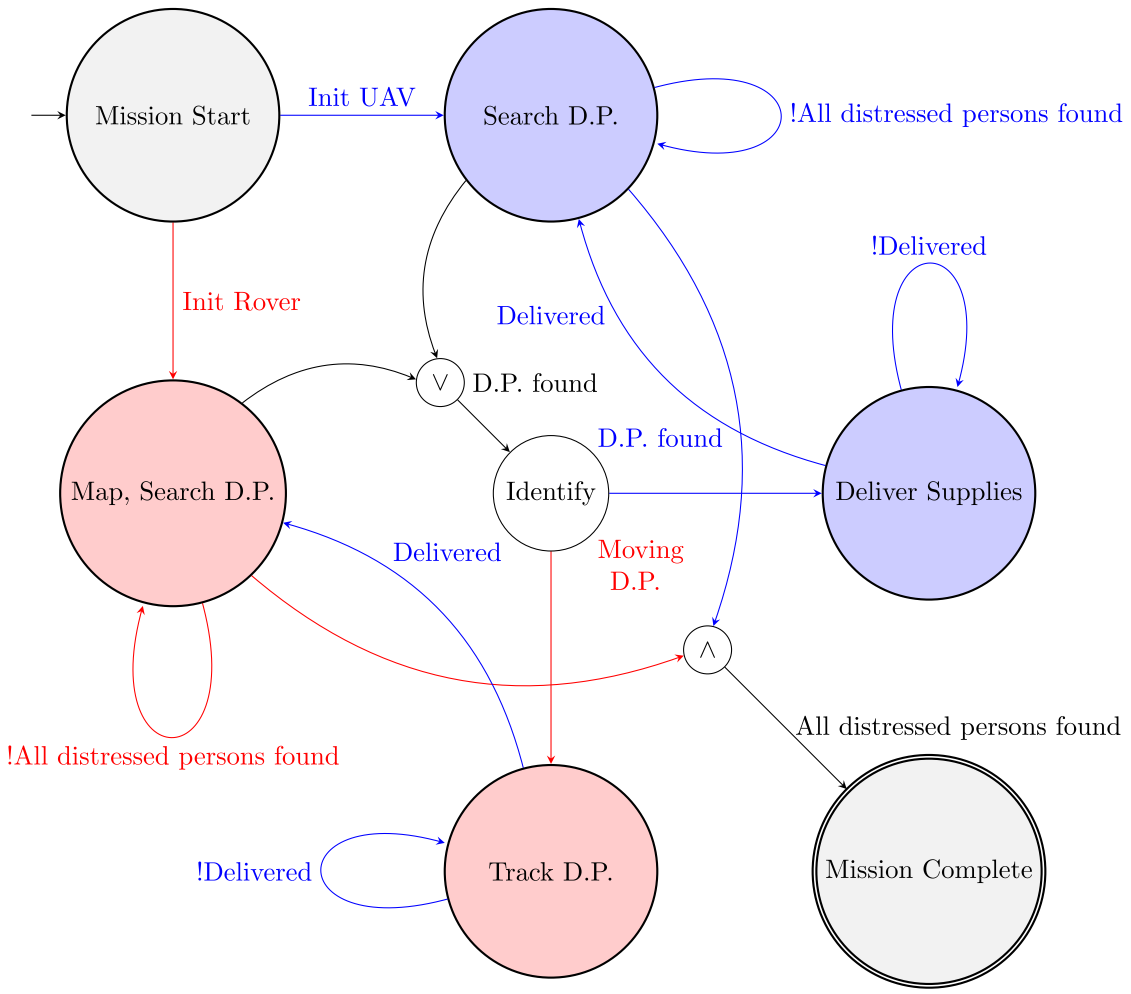

Figure 2: Illustration of the sequence of events for a mission. Red arrows indicate actions dependent on the Rover and blue arrows indicate actions dependent on the UAV, red states are only visited by the Rover and blue states by the UAV. Black arrows and text indicate global actions. *D.P. = Distressed person, ! = FALSE,* $\vee$ *= OR,* $\wedge$ *= AND*

The missions states are illustrated in Figure 2 and a detailed explanation of the states can be seen below:

- **Mission start:** The mission is defined and the Rover and UAV is ready to be deployed.

- **Search D.P:** The UAV is flying above the area searching for distressed persons with its camera.
- **Map, Search D.P:** The Rover is mapping the area as well as searching for distressed persons.
- **Identify:** The Rover or UAV finds a distressed person and identifies it.
- **Deliver Supplies:** The UAV delivers supplies to the distressed person.
- **Track D.P:** The Rover tracks the distressed person in the event of it moving around.
- **Mission Complete:** The map is full explored and potential distressed persons are found and delivered supplies to.

2.3 Communication

The primary software used for communications, both between subsystems and inside a subsystem, is ROS2. ROS2 enables communications via:

- Topics
- Services
- Actions

Topics will be primarily used in the communication utilizing ROS2. Furthermore, the communications between the Base Station, Rover and UAV is accomplished through a network (which enables the use of ROS2 topics). While running simulations on a single computer, a bridge network will be employed to communicate between the different systems locally, whilst running on hardware, each system will act as host on their computing platform and communicate over Wi-Fi.

3 Base Station

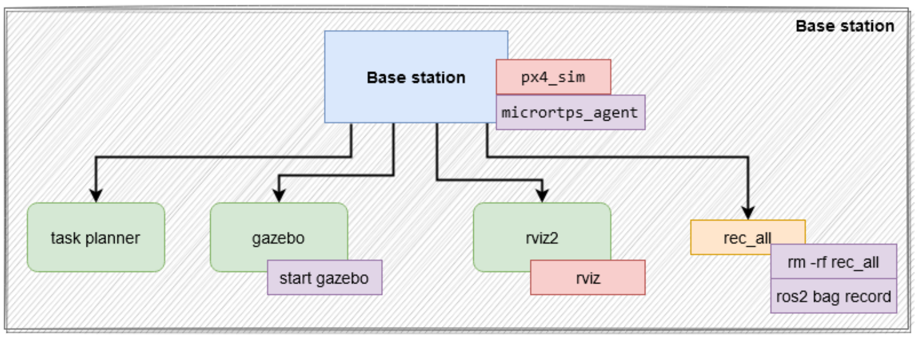

Figure 3: Schematic view of the Base Station's launch file.

The purpose of the Base Station is to plan and coordinate the missions. It will send instructions to the Rover and the UAV and make sure they execute the mission goals. The Base Station is also responsible for running Rviz2, which visualizes the LIDAR-data and positioning. Figure 3 shows an overview of the base station's responsibilities.

3.1 Discovery and Search Algorithm

To find distressed persons, different algorithms will be used depending on the phase of the mission, explained in Section 3.1.1 and Section 3.1.2.

3.1.1 Phase one

During phase one, the area will be mapped by the Rover to prepare for phase two. To map the environment, the Rover will run a discovery algorithm called Frontier-Based exploration. This algorithm uses frontiers as boundaries between open space and unexplored space. The idea is then to move towards this boundary to gain the most new information about the world. By moving the Rover towards this boundary, it will use SLAM to scan the nearby unexplored area and update the map. By continuously doing so, the whole area will be explored. [11]

While the Rover maps the area, the UAV will run a predefined search pattern over the entire environment, searching for distressed persons. Different patterns will be tested depending on the time it takes the Rover to map the environment. One idea could be to move in a parallel search pattern across the environment.

3.1.2 Phase two

Once the area is mapped, then the resulting occupancy grid will be used to create a grid representation of the area. This will be accomplished by changing the resolution of the resulting occupancy grid from phase 1, changing each grid point from a resolution of 0.25 to 0.5 meters. That is, each grid point will be an average of four grid points in the original occupancy grid. This occupancy grid will then be masked such that all obstacles are removed by assigning them the value -1 (representing spaces which are disregarded) and the rest of the space is assigned the value zero. This occupancy grid will be used to model the most probable positions of a distressed person.

Let $X = \{x_1, x_2, ..., x_n\}$ be a set representing a partitioning of the free space (space not occupied by obstacles) in the map into quadratic disjoint areas of the same size. The problem is formulated in terms of locating a moving distressed person z_t in the areas x_i during discrete time steps t. We associate with the set X a probability mass function p_t which defines the probability of the distressed person z_t occupying the locations $x_i \in X$ during a time step t, that is, $p_t(x_i) = P(z_t = x_i)$. It holds that $0 \leq p_t(x_i) < 1$, and that $\sum_{i=1}^{n} p_t(x_i) = 1$, and thus, p_t truly defines a probability density function. Furthermore, we restrict ourselves to the vehicles occupying a single location at a time, with an orientation north, west, south, or east, and only being able to move between locations in the same directions (that is, not diagonally). We let the upscaled and masked occupancy grid represent our set of locations X.

We define for each robot a directed and connected graph $G = (V, A)$, where V represents the different possible nodes and A the possible actions (edges) between them. For the Rover, this means that V is chosen to be all the locations $x \in X$ with an attached orientation, θ, which can be either north, east, west, or south. Thus, for the Rover, the edges $a \in A$ represents either rotations in the same location or moving in the current orientation to a new location. If there is something obstructing the path between the locations, then that a will not be included in the set A. For the UAV, which is assumed to be omnidirectional, $V = X \backslash \{\text{no-fly zones}\}$, and $a \in A$ is chosen such that the locations in V are connected to its neighbors.

Let $\varphi(x_i)$ denote a search function which acts during a discrete time step, returning $\varphi(x_i) = 1$ if a distressed person was detected during the time step, and $\varphi(x_i) = 0$ otherwise. We assume that there is a chance for false negatives during searching of a location x_i, and therefore associate a probability of detection $p^{\varphi}(x_i)$ to $\varphi(x_i)$. That is, given that a distressed person occupies a location in the area searched by the searcher during the time step t, then $p^{\varphi}(x_i)$ is the probability of detecting the distressed person. We do not, however, model the chance of false detection, instead the probability of false detection is kept close to zero by requiring that the detection algorithm spots the distressed person for many consecutive frames. Furthermore, should a false detection occur, then the target tracking algorithm will quickly lose the non-existent target and the target will be counted as lost, and thus, the search will resume with the previous distribution of the probability mass function.

A searcher may detect a distressed person in an area around the searcher's position x_i, depending on the sensor type and how it is mounted. Therefore, we consider the search function $\varphi(x_i, \theta, X)$ to not only operate on x_i, but over the set of locations $x \in X$. The detection chance $p^{\varphi}(x_i, \theta, X)$ is modelled to depend on the sensor's performance range, that is, the detection chance is 0 outside the field of view of the sensor and may vary inside the field of view depending on the sensor type and its mounting. This method of modelling the field of view of the searchers is inspired by [12].

Since the Rover's and UAV's cameras operate using different perspectives, the probability of detection for the locations $x \in X$ in the same vehicle location and with the same vehicle orientation, will be different. The Rover, with its position perpendicular to the ground plane, will approximately sense targets in a cone, see Figure 4. The RPi camera has a horizontal field of view of 62.2° [4], which will approximated to being able to view unobstructed points $x \in X$ in a 45° cone emanating from the middle of the location that the Rover is positioned in, with locations at the edge of the periphery associated with a lower detection chance. The UAV's camera on the other hand is mounted parallel to the ground plane. The RPi camera has a vertical field of view of 48.8° [4], which means that the view from above depends on the orientation of the UAV. We choose to disregard this, and approximate the field of view as symmetric around the UAV's position, see Figure 5. Once an area is searched by one of the vehicles, the result of the search is used to obtain the posterior probability for p_t, using Bayes' rule and the detection probability for each location $x \in X$.

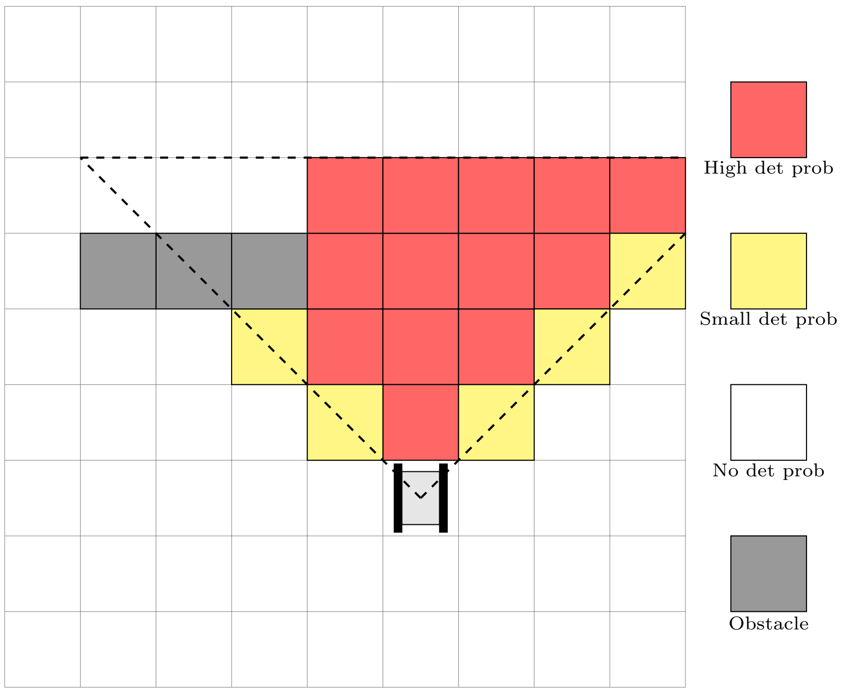

Figure 4: Schematic view of the search function for the Rover. The dashed line represents the 45° cone emanating from the middle of the Rover's location. The edges of the camera-view have a lower detection rate than the middle of the view. The part of the view obstructed by the obstacle has zero detection rate.

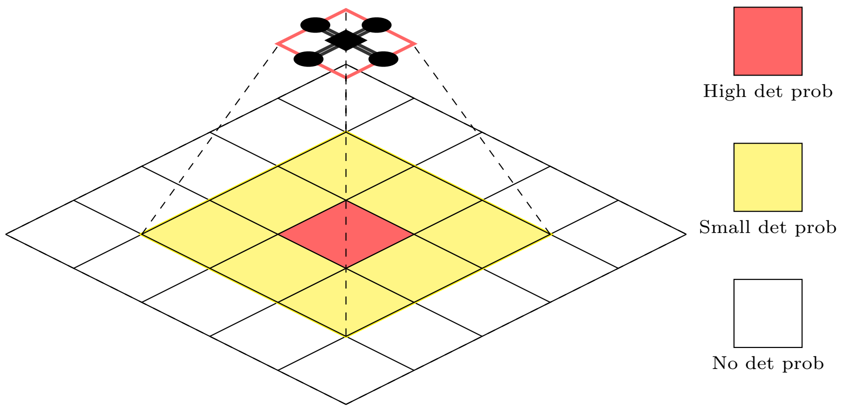

Figure 5: Schematic view of the search function for the UAV. The edges of the camera-view have a lower detection rate than the middle of the view. The UAV's search function is, as opposed to the Rover's, assumed to be agnostic to orientation in a given location. Note, the size of the detected area is only for visualization, the UAV will have a larger swathe of red high detection locations around it, with a brim of yellow low detection around it.

This method is quite general and works well in that it fuses the information from multiple searchers. It can also quite easily be extended to use more information about the scenario. If, for instance, there is some prior information about the distressed persons' location, say they lit a distress beacon, then it can be incorporated into the prior distribution of p_t. Furthermore, if there is information about the distressed persons' movement, such as speed, movement pattern and so on, then a motion model can be attributed to them.

In summary, we have defined the locations that can be searched, how they can be searched, established a representation of the knowledge gained through the searching in the form of a probability mass function, and also detailed a representation of how the vehicles can move between locations. Now it only remains to choose a decision policy for how the vehicles will move between the locations. We will employ a decision policy which boils down to a Markov chain. The Markov chain is obtained by using the graph (V, A) to generate a stochastic matrix P_t, which defines the possible state transitions and associates state transition probabilities.

By considering every node in V as a state for the searcher, then the possible state transitions are defined by A, and the associated state transition probabilities are obtained by normalizing the target probabilities in the possible states' search function (with respect to the prior). The associated probability for the turning state transition for the Rover might be reduced (and thus the moving state transition increased) with some tuning parameter in order to disincentivise too much turning in place. Thus, we define a stochastic matrix, P_t, by considering A and associating the edges with the state transition probabilities.

P_t will also need to be updated every time p_t is updated, since the underlying probabilities used to compute the state transition probabilities will have changed. In order for the amount of calculations to be feasible, the Markov chain will be of the third degree, that is, three steps in the chain (three movements) will be calculated at a time. This means that the updated posterior is only used for three steps at a time. The result is that the communication between the robots and the Base Station, which computes the updated probabilities and the Markov chain steps, will be limited, but for two out of three steps outdated information will be used to make the decision.

This decision policy has a few advantages: firstly, it is weighted to prefer the locally optimal policy, secondly, the stochastic nature of the policy will lead to some locally non-optimal decisions that can disrupt evasive actions of an evading target.

Unfortunately, this method only works for searching for one distressed person. The probability mass function which is updated is essentially a marginalized version of the probability mass function describing the movement of all the distressed persons. However, as it is likely that the distressed persons will move independently, this is not a big problem. We will find multiple distressed persons by restarting the method after convergence, but it could have been handled by using multiple 'marginalized' probability density functions, such as in [3].

3.2 Task Planner

The task planner will be responsible for coordinating the UAV and the Rover during the mission. The robots have different capabilities, meaning that one of the robots can perform tasks that the other one cannot. E.g., the UAV can easily fly back and pick up supplies, which is more cumbersome for the Rover. The task planner will take this into account and give the robots tasks depending on their capabilities to complete the mission according to Figure 2. All different behaviors can be seen in Section 3.2.1

3.2.1 Behaviors

The behaviors listed below are actions that the task planner will be able to instruct the robots to perform. The behaviors will be decided based on information that the task planner receives from the robots.

Rover The Rover will have the following behaviors.

- Map area - The Rover will run a discovery algorithm to map up the area. When, complete, it will inform the Base Station that Phase one is completed.
- Search an area – The Rover will search a selected area for distressed persons. When completed, it will inform the Base Station.
- Track a person – The Rover will follow the distressed person until the UAV arrives with supplies.
- Land UAV – Stop the Rover to land the UAV.

UAV The UAV will have the following behaviors.

- Phase one search - The UAV will search the whole environment for distressed persons with a predefined search pattern.
- Search an area – The UAV will search a selected area for distressed persons. When completed, it will inform the Base Station.
- Collect supplies – The UAV will travel to the supply point and collect supplies.
- Deliver supplies – The UAV will deliver the supplies to the distressed person.
- Land – The UAV will perform a landing maneuver on the Rover or on the ground.

- Track a person – The UAV will follow the distressed person until the Rover arrives.
- Take-off - The UAV will perform a take-off maneuver from the Rover or from the ground.

3.3 GUI

A simple GUI will be implemented in order to visualize the mission and add a simple way of interacting with the mission. The GUI will be able to visualize the map. including the rover, UAV, obstacles, No-fly zones, and distressed persons. Additionally, Rviz2 will be displayed, which will show the exploration of the map. A rough draft of what the GUI might look like can be seen in Figure 6.

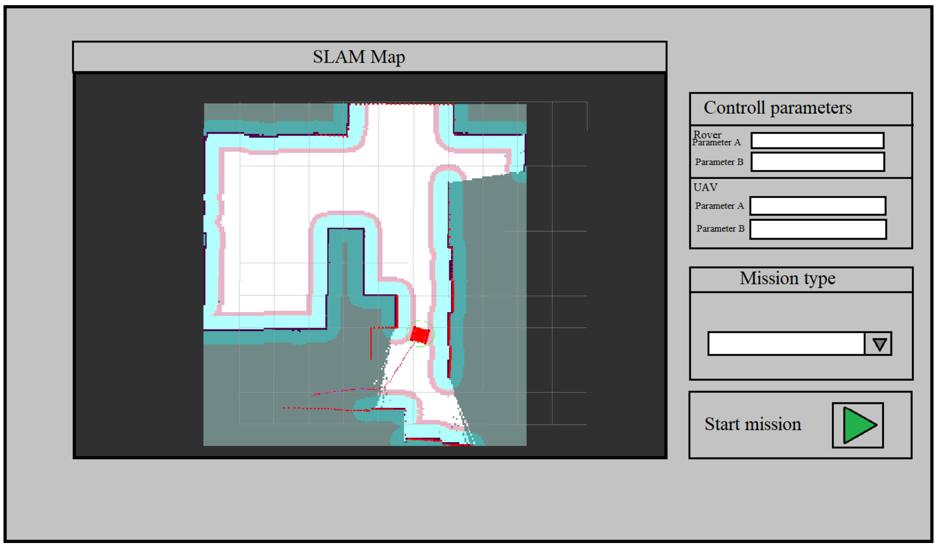

Figure 6: A rough draft of the GUI. The drone and planed mission should also be displayed in the map.

4 Rover

This chapter will describe the design of the Rover. It will describe both hardware and software, as well as different modes of operation. An overview of the Rover design for both hardware and simulation can be seen in Figure 7.

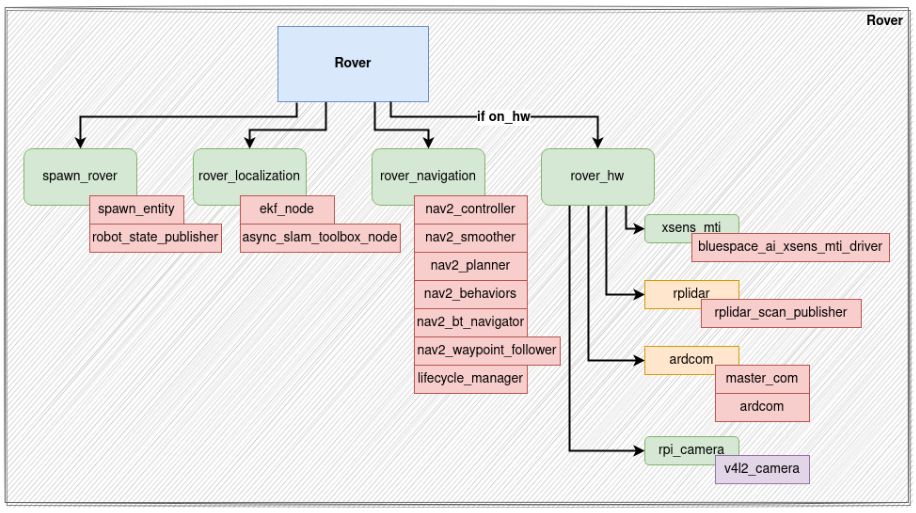

Figure 7: Schematic view of the launch file for the Rover.

4.1 Hardware

The rover will contain a multitude of individual hardware components. An overview showing which hardware components that communicate with each other can be seen in Figure 8. The included hardware and short descriptions are listed below.

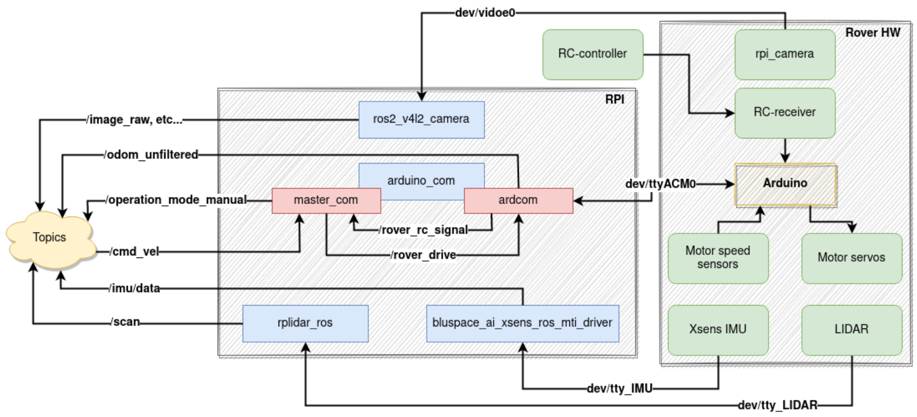

Figure 8: Overview of the hardware communication.

- **Raspberry Pi 4** - Onboard computer with Wi-Fi.
- **LIDAR, RPi LIDAR A2 from SLAMtech** - Rotating laser to obtain heading and range data that is used in SLAM.

- **RPi Camera** - Camera connected to the RPi.
- **Arduino** - Microcontroller to receive odometer data and control the tracks.
- **RC-receiver** - Receiver for manual drive.
- **RC-controller** - Controller to manually control the Rover.
- **Odometer** - Instrument used to calculate the distance the Rover has traveled since start.
- **Motor servos** - One servo per track to drive the Rover.
- **IMU** - Inertial measurement unit, used for position estimation.
- **Qualisys balls** - Reflective markers needed for positioning in Qualisys.

4.2 Odometry

The odometry node parses the wheel encoder data and publishes an estimate of the position and velocity of the Rover to the ROS network. A standard message type for odometry is published containing estimated positions, orientation, linear velocity and angular velocity. This data is then passed to an EKF-filter and used to estimate the position and velocity of the Rover. Equations (1 - 13) [7] describe how the parsed odometry data is used to estimate the position and velocity of the Rover.

The different variables are explained in Table 1. The wheel encoder data is useful for estimating the position of the rover over small distances, but drifts over travelled distance, and is therefore mostly useful for supporting the estimation of the odometry in smaller areas where the Qualisys system has poor coverage.

Table 1: Variables used in the odometry Equations (1 - 13)

Variable	Description
R	Wheel radius
Resolution	Number of ticks per revolution
T$_W$	Track width
$ticks_{l,curr}$	Current total number of ticks for the left track
$ticks_{r,curr}$	Current total number of ticks for the right track
Δt	Time since previous measurement
$ticks_{l,old}$	Previous total number of ticks for the left track
$ticks_{r,old}$	Previous total number of ticks for the right track
$\Delta ticks_l$	Number of ticks since previous measurement for the left track
$\Delta ticks_r$	Number of ticks since previous measurement for the right track
$\Delta\theta_l$	Rotation in radians for the left wheel since previous measurement
$\Delta\theta_r$	Rotation in radians for the right wheel since previous measurement
Δs_l	Distance the left track has moved since previous measurement
Δs_r	Distance of the right track has moved since previous measurement
Δs	Distance of the Rover have moved since previous measurement
$\Delta\theta_{yaw}$	Difference in yaw rotation of the Rover since previous measurement
$\theta_{yaw,old}$	Previous global yaw rotation
θ_{yaw}	Current global yaw rotation
x_{prev}	Previous global x-coordinate
x_{new}	Current global x-coordinate
y_{prev}	Previous global y-coordinate
y_{new}	Current global x-coordinate
ω	Current angular velocity
v	Current linear velocity

The odometry data received from the Rover is used to estimate its current position, linear velocity and angular velocity. The difference in number of ticks since the last measurement is calculated as.

$$\Delta ticks_l = ticks_{l,curr} - ticks_{l,old} \tag{1}$$

$$\Delta ticks_r = ticks_{r,curr} - ticks_{r,old} \tag{2}$$

This data is then used to calculate the angular difference for each track as.

$$\Delta\theta_l = \Delta ticks_l \frac{2\pi}{2000} \tag{3}$$

$$\Delta\theta_r = \Delta ticks_r \frac{2\pi}{2000} \tag{4}$$

The angular difference is used to estimate the distance each track have moved since the last measurement and is calculated as.

$$\Delta s_l = \Delta\theta_l R \tag{5}$$

$$\Delta s_r = \Delta\theta_r R \tag{6}$$

The estimated Euclidean distance the Rover has travelled and the difference in yaw angle since the last measurement is then calculated as.

$$\Delta s = \frac{\Delta s_l + \Delta s_r}{2} \tag{7}$$

$$\Delta\theta_{yaw} = \frac{\Delta s_r - \Delta s_l}{T_W} \tag{8}$$

The local rotation since the start is estimated as.

$$\theta_{yaw} = \theta_{yaw,old} + \Delta\theta_{yaw} \tag{9}$$

The current position in x and y coordinates, linear velocity and angular velocity which is published to the ROS network is estimated as.

$$x_{new} = x_{prev} + \Delta s \cos(\theta_{yaw,old} + \frac{\Delta\theta_{yaw}}{2}) \tag{10}$$

$$y_{new} = y_{prev} + \Delta s \sin(\theta_{yaw,old} + \frac{\Delta\theta_{yaw}}{2}) \tag{11}$$

$$\omega = \frac{\Delta\theta_{yaw}}{\Delta t} \tag{12}$$

$$v = \frac{\Delta s}{\Delta t} \tag{13}$$

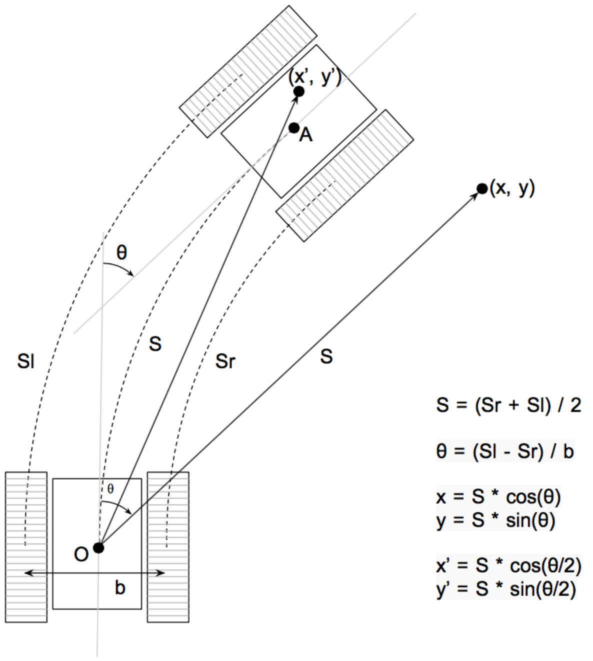

Figure 9: Visualisation of parameters in the odometry model. Source: Solderspot [10]

4.3 SLAM

The Rover will use an algorithm for SLAM. This will map the area while keeping track of the Rover's relative location in the that area. The SLAM-algorithm uses data from the LIDAR and the positional data from Qualisys to map the world and localize the Rover. The ROS2-package Slam Toolbox is used to implement the slam algorithm, and it uses a graph-based 2D-implementation of slam. It is a continuation of SRIs old open source slam algorithm OpenKarto [6], which had multiple popular ROS1 implementations [9]. The package requires published LIDAR data and a valid transform from the vehicle (the base_link-frame) to the vehicle's starting position (the odom-frame). In turn, it publishes the map in the form of an occupancy grid and the estimated position of the vehicle in the map-frame.

4.3.1 LIDAR

The Rplidar_Ros2 package generates measurement data from the mounted LIDAR sensor, consisting of the distance and angle to the detected objects. This data is published on the topic \scan, and is used by Slam Toolbox. The package is maintained by Slamtech, the producers of the RPlidar product.

4.3.2 Robot Localization

The robot localization package is a classic ROS-package which has been patched to work with ROS2. It implements an EKF-filter which fuses data from the different odometry sources and publishes the filtered odometry data and the associated transform from the base_link-frame to the odom-frame, which Slam Toolbox requires. The package has support for fusing data from a plethora of odometry sensors, including: GPS, IMU and wheel encoders. There is no native support for the Qualisys message-type, however, the data from those messages can be converted in terms of either a GPS-signal or wheel-encoder data.

4.3.3 Occupancy Grid-Representation

As mentioned in Section 4.3, the Slam Toolbox package represents the map as a 2D-occupancy grid. An occupancy grid is a data structure which assigns a value between 0-1 to an ordered set of cells representing the map. A value of 0 indicates a completely unobstructed area, essentially air, and a value of 1 represents a fully occupied area, i.e a wall. The occupancy grid-representation has a few advantages over a simple binary representation, in that it can model uncertainty in the mapping and a cost associated with being near objects. Mapping methods are not perfect, and thus, it is natural to represent uncertainties in the mapping using a value in-between occupied and unoccupied. Furthermore, since estimation of the position of the vehicle is never perfect either, it is natural to represent the cost of entering areas around objects in much the same way. The motion planner and motion controller (described in greater detail in Section 4.4.2) use the occupancy grid information as weights when planning paths and while following them, in a way that balances the costs associated to entering partially occupied areas with the potential to save time.

Both a local and global occupancy grid is maintained, the so-called local and global costmaps, with differing levels of resolution. The local costmap is centered around the vehicle and has a higher resolution, but is not maintained over the whole map. The global costmap, as the name implies, is a costmap spanning the whole explored area, but with a reduced resolution. These costmaps serve complementary roles; the local costmap is useful for the motion controller and motion planner in navigating the vehicle in real-time, whilst the global costmap is needed for the motion planner and mission planner to plan and coordinate the actions of the vehicle on the global scale.

4.4 Navigation

The Rover uses a motion planner to find the best path from its current position to the goal nodes decided in the search algorithm. To follow the path, a motion controller is used.

4.4.1 Motion Planner

The motion planner will use the goal nodes delivered by the Search Algorithm to plan out a path for the Rover. It will use information from SLAM to calculate a path that does not collide with obstacles. An A* motion planner from Nav2 is already implemented and works in simulation. In Nav2, there are many planners with different pros and cons. During the project, different planners will be tested and evaluated and the best planner for our mission will be chosen.

4.4.2 Vehicle Motion Control

The Rover will also have a vehicle motion controller. It ensures that the Rover follows the path provided from the motion planner. It has to take into account the freedom of movement of the vehicle it controls, the Rover is a non-holonomic skid-steered robot and can therefore only use a subset of controllers which support those specifications. A path tracking controller from Nav2 using a regulated pure pursuit implementation is already in use. The Rover will, however, need two different motion controllers: one for controlling the vehicle while tracking a target, and another for the rest of the time. The need is motivated by the fact that the motion controller used while tracking needs to not simply follow a path in the most efficient manner, but also maintain an angle to the target while moving. This is important since the camera needs to be pointed towards the target during tracking, or else the Rover risks losing the target. Furthermore, the Rover will for now not reverse, but instead turn around and move forward, which probably is not the most optimal solution and will most likely change.

4.4.3 Motor Controller

A motor controller is already implemented on the Rover. It uses a pair of PID-controllers to control the motor output of the Rover. One is responsible for the heading speed and the other for the yaw speed. The controller is needed for the motor output, since the dynamics of the Rover is nonlinear and not particularly well-approximated with a linear equation. Especially, the power needed to overcome the initial friction while turning in place has deadzone-characteristics. However, after the initial friction is overcome, the relationship between the motor output and achieved velocity is far closer to a linear relationship. Thus, the integrating part of the PID-controllers is the most important, as it is used to overcome the initial friction.

4.5 Tracking

Tracking will be used in order to identify and continuously estimate the position of a distressed person. The tracking should therefore be able to distinguish between a distressed person and a stationary distressed person.

4.5.1 Recognition

While the Rover is not tracking a distressed person, an image recognition algorithm will run, tasked with identifying distressed persons. Image recognition algorithms generally are more computationally intense than tracking algorithms, but do not need to be initialized with a target in the frame. Therefore, it is suitable to use in order to detect distressed persons in the whole frame, but less good at estimating the target position from frame to frame. The algorithm will search for the presence of colored pixels, using a sliding window over each frame, to exhaustively search for color. Since the RGB-space is less robust in handling noise, the data will be converted to the HSV-space before the algorithm is initiated. This method assumes that the environment used is white and that the different kinds of distressed persons have distinct and different colors, so that the color searched for is only present when a distressed person is in frame.

The higher computational complexity will mean that the algorithm will not be able to be run on every received image, but will necessarily have to be quick enough that the distressed person is not missed.

4.5.2 Tracking Algorithm

The Rover will use a modified Camshift Algorithm to handle the tracking of distressed persons. It will combine the Camshift Algorithm [1] with a Kalman Filter using a motion model for the target. Additionally, a pinhole camera model for the monocular camera module will be used to estimate the target's distance from the Rover, achieving 3D tracking, as opposed to the 2D tracking, which the traditional algorithm achieves.

The classical Camshift Algorithm is an improvement of the Meanshift Algorithm. The Meanshift is a non-parametric clustering algorithm, which, given an initialization, uses an iterative search to find the nearest dominant peak in a probability distribution. The idea of the algorithm is illustrated in Figure 10. More rigorously, given n samples $x_i, i = 1, 2, ..., n$ in R^d, then the meanshift vector $m(x)$ is defined as

$$m(x) = \frac{\sum_{x_i \in S_h} K(x_i - x)x_i}{\sum_{x_i \in S_h} K(x_i - x)}, \tag{14}$$

where S_h is a closed ball in R^d, centered on x, and K is a kernel function which weights points x_i inside S_h. Thus, the meanshift vector forms an average, with respect to the kernel function $K(x)$, of the samples in a neighborhood S_h around a point x. Now consider the difference $m(x) - x$, which corresponds to a *mean shift*, i.e a shift to the densest region in S_h. The iterative Meanshift Algorithm computes the mean shift given some initial point x_0, and then iteratively recomputes the mean shift using the output mean shift from the previous iteration as a new point in the algorithm, $x_{k+1} = m(x_k) - x_k$. The method converges to some local optima with respect to the density of the samples in the data.

In the target tracking setting, an image with an initial window containing the target is used to create a histogram. Since RGB-space is prone to noise, usually the HSV-space, which separates out hue (H), saturation (S), and value (V) from an image, is used instead [1]. While more of the dimensions can be used, the standard algorithm is based on the hue, and thus it is the hue which is sampled from the target in the image and stored as a 1-d histogram.

A probability distribution is computed using the histogram in a process called *back projection*. A back projection of an image given a model histogram is computed by, for each pixel value, finding the corresponding bin in the histogram and assigning the back projected image the value stored in the bin at the sampled point. Thus, an image, $I(x, y)$, is attained which represents a probability distribution of the target in the image.

The back projected image $I(x, y)$ can then be used as an input to the Meanshift Algorithm, along with an input window representing the closed ball, in order to locate the target. Most commonly, the kernel function is chosen as an *image moment*, $M_{i,j}$, in the meanshift algorithm for images. The image moment represents a weighted average of the pixel intensities of an image $I(x, y)$, and is computed as

$$M_{i,j} = \sum_x \sum_y x^i y^j I(x, y). \tag{15}$$

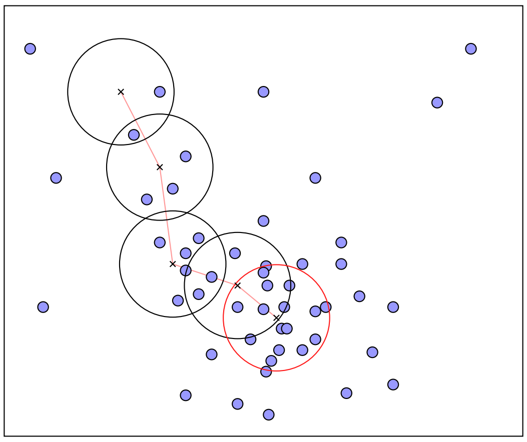

Figure 10: The Figure shows an example of the Meanshift Algorithm. The blue dots represent samples of some distribution, and each black circle represents the closed ball in an iteration of the algorithm, which terminates at the red circle having reached some goal threshold.

Specifically, the first moments $M_{1,0}$ and $M_{0,1}$ are typically used for the x- and y-coordinate respectively. From equations (14) and (15), it follows that x- and y-coordinates are normalized by the zeroth moment $M_{0,0}$. Using the back projected image $I(x, y)$ and the Meanshift Algorithm with the image moments kernel, the following center position (x_c, y_c) is obtained for every iteration:

$$(x_c, y_c) = (\frac{M_{1,0}}{M_{0,0}}, \frac{M_{0,1}}{M_{0,0}}). \tag{16}$$

The Meanshift Algorithm terminates once the initial window has been moved to a local optima in the back projected image. This process is then repeated for every incoming image from a video feed, where the previous converged position is used as initial position in the subsequent frame.

The Meanshift Algorithm works well in a setting where the tracked target has a constant size, that is, it struggles with targets moving to and from the camera, deforming targets, or partially occluded targets. The Camshift Algorithm's contribution to the Meanshift Algorithm is in tackling this problem by adaptively changing the frame size s of the target as a function of the zeroth order moment $M_{0,0}$. In the original paper [1] the window size s was updated as

$$s = 2\sqrt{\frac{M_{0,0}}{256}}, \tag{17}$$

which will be employed in the tracking algorithm. The Camshift Algorithm is summarized as a flowchart in Figure 11.

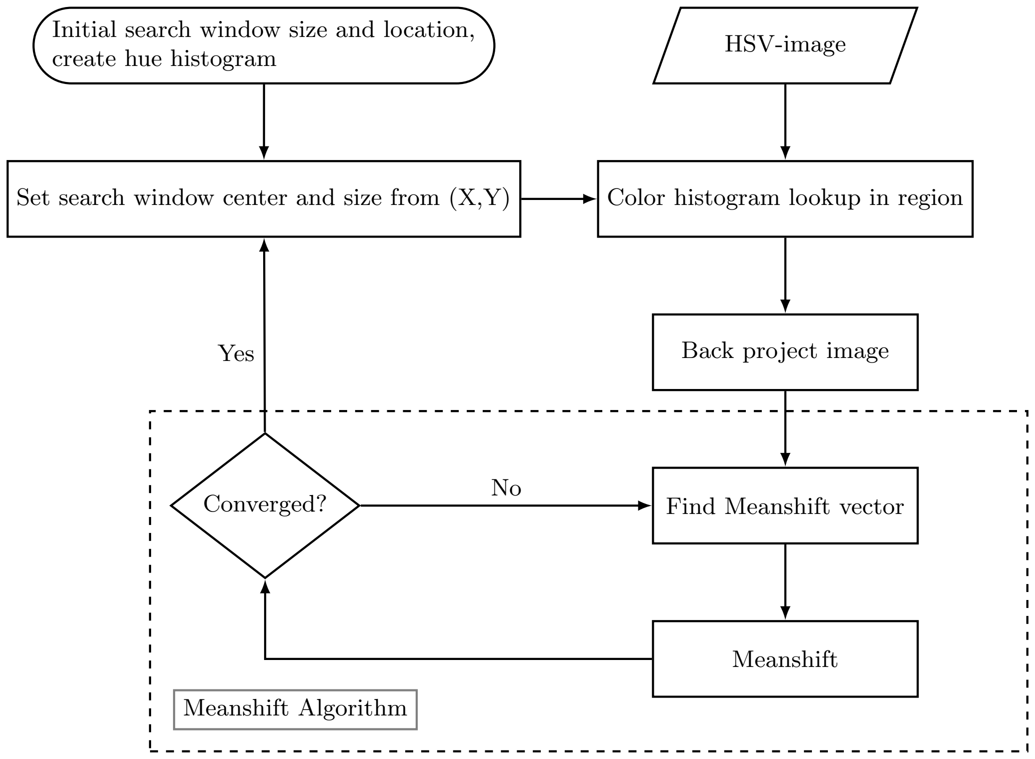
Initial search window size and location, create hue histogram
HSV-image
Set search window center and size from (X,Y)
Color histogram lookup in region
Back project image
Yes
Converged?
No
Find Meanshift vector
Meanshift
Meanshift Algorithm

Figure 11: The Figure shows a flowchart of the Camshift Algorithm. The dashed box represents the steps in the Meanshift Algorithm. The algorithm starts at the rounded rectangle and terminates once there are no new HSV-images to process.

In order to incorporate the depth-dimension in the estimated position of a target, a pinhole model will be employed. Given the focal length of the camera, and given the height of an observed object in a frame, the distance of the object to the camera can be estimated [2]. Thus, the depth of the target can be measured using images from the camera.

The Camshift Algorithm will be extended to include a Kalman Filter in order to achieve better tracking in the presence of occlusion of the target or other disturbances. The target will be described by a constant velocity model, which will be used in the time update of the Kalman Filter to estimate the target in a dead-reckoning fashion when it is occluded. Occlusion will be measured using the Bhattacharyya coefficient [5], which is a measure of the overlap between two statistical samples. In this case, the statistical samples consist of the original histogram of the target, $p(x)$, and a histogram from the converged Camshift Algorithm, $q(x)$. Given these quantities, the Bhattacharyya coefficient, $C_B(p,q)$, is defined as

$$C_B(p,q) = \sum_{x \in X} \sqrt{p(x)q(x)}, \tag{18}$$

where X is the set of hue-bins in the histograms, and it holds that $0 \leq C_B(p,q) \leq 1$. A low value of $C_B(p,q)$ indicates similar samples, and a high value indicates dissimilarity between the samples. The coefficient is used by fixing a threshold α, beyond which the object is said to be occluded. When the target is visible, the output of the Camshift Algorithm will be taken as a measurement update in the Kalman Filter. The time update will be used to update the starting position of the Camshift Algorithm between frames.

See Figure 12 for an overview of how the combined Camshift and Kalman algorithm will work.

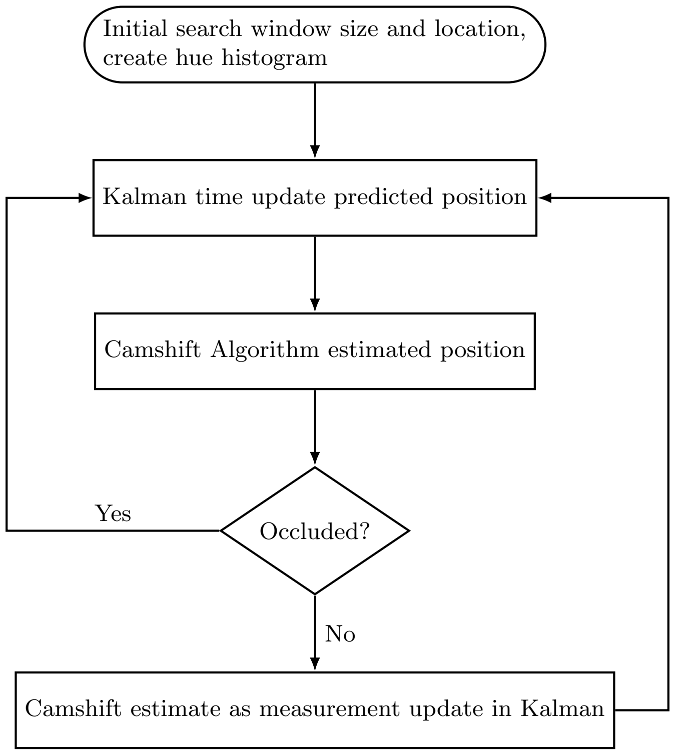
Initial search window size and location, create hue histogram
Kalman time update predicted position
Camshift Algorithm estimated position
Occluded?
Yes
No
Camshift estimate as measurement update in Kalman

Figure 12: An overview of the Camshift Kalman algorithm to estimate the position of the distressed persons. Occlusion is measured using the Bhattacharyya coefficient and a predetermined threshold.

4.6 Modes of Operation

There are two modes of operations for the Rover, manual and autonomous mode. On the RC controller there is a switch which is used to switch between the modes. The Arduino receives the RC-signals and transmits them to the master_com node, which decides if the Rover should drive autonomously or manually.

4.6.1 Manual Mode

If the Rover is in manual mode, the master_com node will translate the RC signals into motor signals. The manual mode has the same software as the autonomous mode, but the difference is that the missions will be performed as the driver desires and not performed by discovery algorithms and task planners.

4.6.2 Autonomous Mode

In autonomous mode the Rover is driven autonomously, and the motor signals depend on controllers, planners, and search algorithms.

5 UAV

The overview of the UAV subsystem is visualized by the UAV launch file in Figure 13. When running the launch file, it initiates the UAV offboard controller, the camera and the image processing nodes, and the node for the motion planner. If the UAV is to be simulated, the launch file spawns the UAV in Gazebo and sends a command to the Base Station to start the simulation. If the UAV is running on HW, the microRTPS agent is initiated.

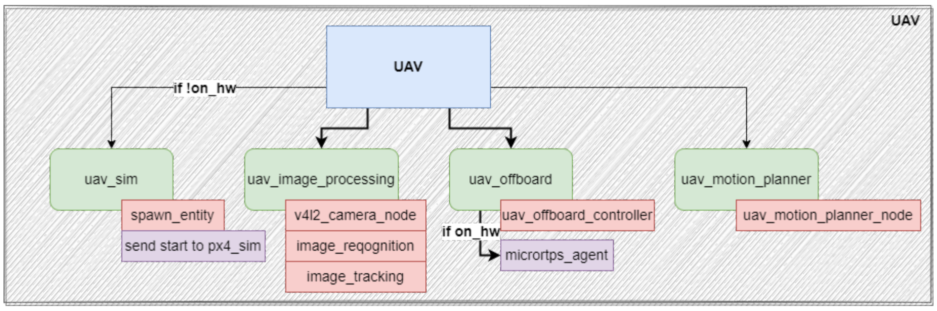

Figure 13: Schematic figure of the launch file for the UAV.

5.1 Hardware

The UAV is a quadcopter based on the *Lumenier QAV-R 2* frame with components listed in Section 5.1.1. An overview of the system can be seen in Figure 14, where both HW and SW components are illustrated.

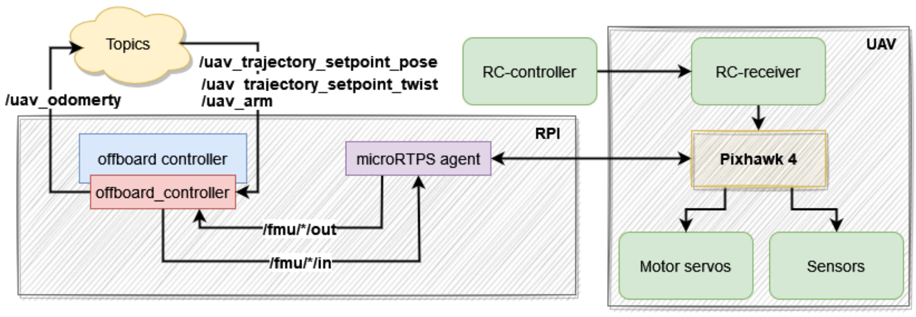

Figure 14: Overview of the UAV hardware.

5.1.1 Hardware Components

- **Pixhawk 4 -** Flight controller with internal IMU and barometer
- **Raspberry Pi 4b -** Onboard computer with Wi-Fi
- **RPi camera -** Camera connected to the RPi
- **RC-receiver -** Receiver for manual flight
- **Motors and ESC -** Propeller motors and Electronic Speed Controller

- **Battery** - 11.1V Lithium Polymer (LiPo) battery
- **RC-controller** - Controller connected to the UAV for manual flight

5.1.2 Safety Features

The UAV will have some safety features to minimize the risk of crashes and bodily injury. The Pixhawk already has some built-in fail-safes that, when triggered, will land the UAV immediately. The fail-safes used are the following: A fail-safe which activates when the battery has reached a specified minimum voltage, an RC loss fail-safe that will activate when the RC-signal to the manual controller is lost, and a fail-safe which activates when the Pixhawk loses connection to the RPi.

In addition to these fail-safes, a fail-safe for losing Qualisys positioning data will be implemented. The UAV will land if it has not received positioning data in the last 2 seconds.

The RC-controller will also have a kill switch that stops all rotors, and the user will be able to switch to manual control at any time.

5.2 Software

In this section, all software on the onboard RPi will be presented.

5.2.1 Onboard Communication

The communication between the RPi and the Pixhawk is carried out over a microRTPS bridge. This bridge converts ROS2 topics to uORB topics the Pixhawk can interpret. An agent on the RPi receives data from a ROS2 topic, converts it, and sends it to a client on the Pixhawk. If data is sent from the Pixhawk instead, it works the other way around and the agent publishes the received data on a ROS2 topic.

A ROS2 node called ”offboard_control” acts as an intermediary between the agent and the rest of the system. The node needs to publish a position or speed command to the Pixhawk with a frequency of at least 0.5 Hz, otherwise a fail-safe will be initiated, and the UAV will land. The node will by default publish these commands with a frequency of 10 Hz.

5.2.2 Position Command

To control the UAV, a ROS2 ”Pose” message containing a desired position and orientation will be sent. This message will be sent to the ”offboard_control” node on the topic ”UAV_trajectory_setpoint_pose” and will then be sent to the Pixhawk. The UAV will then fly to the position and, with its internal PID controllers, hold that position. This will allow the user to send simple commands with a position to the UAV, which stabilizes around the point by itself.

5.2.3 Position Feedback

The internal controllers in the Pixhawk need external positioning to close the feedback loop. In the simulation, a GPS module is mounted to the UAV, which provides it with position data. However, on the HW, the UAV will depend on the Qualisys positioning system in Visionen. The Pixhawk has support for receiving a ”VehicleMocapOdometry” message which contains position, orientation, velocity and angular velocity. A function to convert the received position from Qualisys to a "VehicleMocapOdometry" message will be implemented. The "offboard_control" node will call a callback function when receiving position data, convert it to the correct message, then publish this to the microRTPS agent. The fact that the UAV and the Qualisys system have different coordinate frames needs to be taken into account and converted in the process.

5.2.4 Recognition and Tracking Using the Camera

The UAV will use the same image recognition algorithm as the Rover to identify distressed persons. Furthermore, the UAV will also use the Camshift Kalman Algorithm to track the moving distressed persons. However, unlike the Rover, a depth estimate will not be computed since, as long as the UAV does not tilt overly much, the camera plane of the UAV will line up with the desired tracking space (the 2D projected plane of the room onto the floor). Furthermore, the depth dimension of the UAV's camera can be better estimated using the estimated height of the UAV in conjunction with the Pythagorean identity (if the target is not directly underneath the UAV).

The bird's eye view, combined with the higher degree of freedom of movement in comparison with the Rover, makes the UAV a better tracker in a free space. However, it is severely limited by the fact that the distressed persons can move into the no-fly zones, in which the UAV cannot enter. Excepting, attempting to gain altitude and then tilting the UAV in the air at the edge of the no-fly zones, there are no reliable ways for the UAV to track the distressed persons in the no-fly zones. Given this problem, it will be the main responsibility of the Rover to track the moving distressed persons, whilst the UAV will fulfill a supportive role of attempting to track the targets to its best ability until the Rover has intercepted them, whereupon the UAV will go and fetch supplies.

5.2.5 Motion Planner

A motion planner will be implemented for the UAV. The motion planner will calculate a way to fly using a sequence of waypoints from a start point to a goal point, while avoiding No-fly zones. In Nav2, there are many planners with different pros and cons. A big minus for the Nav2 motion planners is that they are 2D-based, and thus, will not take into account the altitude of the UAV. However, the aim is to keep the UAV at the same height throughout the mission, so there is not much lost in discarding the altitude dimension. During the project, different planners will be tested and evaluated and the best planner for our mission will be chosen.

No-fly Zones

A No-fly zone is an area where the UAV is not allowed to fly in. The No-fly zones will be initialized in the Base Station, which means that the motion planner will know to not plan paths through No-fly zones. No-fly zones can therefore not be discovered with algorithms such as SLAM, unlike the obstacles for the Rover.

Safety Distance

The UAV will fly at a constant height of 2 meters. In addition, a safety distance of 50 cm to distressed persons will be implemented that will be taken into consideration when landing and delivering supplies.

5.2.6 Take-off and Landing Services

ROS2 services will be implemented for taking off, and landing the UAV. These services will receive a start command, execute the action, and send back a result indicating that the UAV has finished its action.

The take-off service will receive a take-off command and the UAV will start flying up to an altitude between two and four meters, where it will hover. When the UAV has reached the correct altitude, the service will send a message back confirming the take-off is completed.

The landing service will receive the position of the landing site when it is called. The UAV will then fly to a position above the landing site using the motion planner. When it has reached the position above the landing site, the UAV will start to descend until it has landed. When the landing is complete, it will send a confirmation message that it has landed.

In addition to the basic functionality for take-off and landing, the UAV will be able to start and finish on top of the Rover. Take-off from the Rover will not differ too much from starting on the ground, since the only difference is the starting height. Landing on the Rover will require that the Rover is stationary, and the landing site coordinates sent to the landing service must include the actual height of the Rover's landing platform.

To prevent the UAV from landing on a distressed person, a safety feature will be implemented that will cancel the landing if a distressed person is identified below the UAV.

5.2.7 Deliver Supplies

The UAV will deliver supplies to the distressed persons while the Rover tracks them. The UAV will pick up supplies at a given supply point, whereupon it will return to the distressed person to deliver the supplies. Due to the scope of the project, it will not be prioritized to implement a service which picks up physical supplies. Instead, when the UAV has hovered over the supply point and then moved to the distressed person where it descends to the height of one meter, it will perform a shaking action to signal that the task is completed. The shaking action will be implemented as setting two position commands to the UAV on either side of the current position. It is also possible to signal this by spinning horizontally.

5.3 Modes of Operation

The UAV has two main modes of operation: Manual and autonomous. When the UAV is in manual mode, it is controlled by the RC-controller, and in autonomous mode, the UAV is controlled by the onboard RPi.

5.3.1 Manual Mode

There are a few different types of manual modes: Manual/Stabilized, Altitude and Position. The manual/stabilized mode lets the user fly the UAV fully manually, altitude mode stabilizes the UAV at an altitude using the barometer to measure height, and position mode utilizes the external positioning to hold a position which the user can change using the controller. The position mode will be used to verify that the UAV utilizes the position given by the Qualisys positioning system. When flying the UAV autonomously, the user can at any time switch to a manual mode to take over control.

5.3.2 Autonomous Mode

The autonomous mode, called offboard mode, implies that the UAV is fully controlled by the onboard RPi. If the UAV does not receive data from the RPi with at least 2 Hz, the UAV will enter fail-safe mode.

6 Simulation

An important part of the project is the simulation environment. The system uses SIL, which enables testing in simulation in contrast to only testing on hardware. This will be crucial for the testing because of the limited testing-time, and minimizes the risk of damaging the hardware. The simulations will be handled in the base station system.

6.1 Gazebo11

Gazebo is a simulation software that will be used in the project. It is integrated into ROS2 and can handle communicating via ROS2 topics, which is appropriate for the project. It can simulate the world and multiple robots, as well as handle the physics in the simulation.

6.2 Simulation of the Robots

Generally, when simulating robots in Gazebo11, SDF-files are used. An SDF-file includes the information that Gazebo11 needs to simulate the robots accurately, such as, dimensions, sensors, materials etc. The SDF-files for the UAV and the Rover have slightly different origins in how they are created and where they are obtained from.

6.2.1 Rover

The Rover's SDF-file is created using a URDF-file. The URDF-file is where the Rover model is described, including its physical measurements and the sensors used with associated positions. This is then used to create the SDF-file. The URDF-file was created to describe the project's Rover as closely as possible, and used the Rover's (the specific Rover used in this project's) dimensions.

6.2.2 UAV

The UAV has a different method for generating the SDF-file. It is generated when compiling the PX4-Autopilot git project. The SDF-file defines the structure of the UAV, including its sensors. Furthermore, the git project contains a software program which simulates a PX4 flight controller. The simulated PX4 is able to communicate with the microRTPS bridge running on the simulated UAV. However, the simulated drone has different dimensions than the actual drone used in this project. This is because the simulated UAV is a standard UAV from the SIL simulation in the submodule PX4-Autopilot and has all the sensors and flight controller already built in. To create a UAV from scratch that has the same dimensions and structure would take time, and analyzing the benefits of creating a new model, it does not seem beneficial. The simulated UAV is slightly larger, but should not have any major impact on the testing using the simulation, since the flight controller will control the different sizes of UAV similarly. Since the main purpose of the project is to execute a mission and not looking into controlling the UAV, the fact that the UAV will behave slightly different between simulation and real tests can be overlooked. A camera will be implemented on the simulated UAV to make it possible to test image recognition and tracking in simulation.

6.3 Simulation of the Map

The tests in the project will be conducted in Visionen. Therefore, a simulation environment which is designed to look like Visionen will be created. The map will have a white floor and walls with similar dimensions as Visionen. In the simulation environment, it will be possible to add boxes as obstacles and No-fly zones, similar to what will be a part of the physical mission in Visionen.

6.4 Simulation of Distressed Persons

To implement moving distressed persons in simulation, a similar method to simulate robots will be used. An SDF-file will be constructed and pre-defined movement patterns, such as moving in a line or a circle, will be used to emulate the movement of a distressed person.

References

[1] Gary R. Bradski. *Computer Vision Face Tracking For Use in a Perceptual User Interface.* 1998.

[2] D. Burschka and G. Hager. "Vision-based control of mobile robots". In: *Proceedings 2001 ICRA. IEEE International Conference on Robotics and Automation (Cat. No.01CH37164).* Vol. 2. 2001, 1707–1713 vol.2. DOI: `10.1109/ROBOT.2001.932857`.

[3] Timothy H. Chung and Joel W. Burdick. "Multi-agent probabilistic search in a sequential decision-theoretic framework". In: *2008 IEEE International Conference on Robotics and Automation.* 2008, pp. 146–151. DOI: `10.1109/ROBOT.2008.4543200`.

[4] Raspberry Pi Foundation. *Specifications for RPi camera.* `https://www.raspberrypi.com/documentation/accessories/camera.html`. 2022-03-08. (Visited on 10/14/2022).

[5] Weipeng Guan et al. "Visible Light Dynamic Positioning Method Using Improved Camshift-Kalman Algorithm". In: *IEEE Photonics Journal* 11.6 (2019), pp. 1–22. DOI: `10.1109/JPHOT.2019.2944080`.

[6] Kurt Konolige et al. "Efficient Sparse Pose Adjustment for 2D mapping". In: *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 2010, pp. 22–29. DOI: `10.1109/IROS.2010.5649043`.

[7] G.W. Lucas. *A Tutorial and Elementary Trajectory Model for the Differential Steering System of Robot Wheel Actuators.* `https://rossum.sourceforge.net/papers/DiffSteer/DiffSteer.html`. 2000. (Visited on 10/14/2022).

[8] Operation Watch Landlubber with editor Rickard Wretlind. *Project Plan Search and Rescue - land.* Linköping, Sweden, 2022.

[9] slam_karto. *slam_karto - ROS Wrapper and Node for OpenKarto.* `http://wiki.ros.org/slam_karto`. 2019. (Visited on 10/03/2022).

[10] solderspot. *Zumo: Systematic Errors Dead Ahead.* `https://solderspot.net/2014/03/07/zumo-systematic-errors-for-dead-reckoning-oh-my/`. 2014-03-07. (Visited on 10/14/2022).

[11] Brian Yamauchi. *A Frontier-Based Approach for Autonomous Exploration.* Naval Research Laboratory Washington, DC 20375-5337, 1997. DOI: `10.1109/CIRA.1997.613851`.

[12] Huili Yu et al. "Cooperative Path Planning for Target Tracking in Urban Environments Using Unmanned Air and Ground Vehicles". In: *IEEE/ASME Transactions on Mechatronics* 20.2 (2015), pp. 541–552. DOI: `10.1109/TMECH.2014.2301459`.