Making the Milrem Themis UGV ready for autonomous operations

Mathiassen, Kim, Baksaas, Magnus, Græe, Sindre Aas, Mentzoni, Eilert André, Nielsen, Niels Hygum

Kim Mathiassen, Magnus Baksaas, Sindre Aas Græe, Eilert André Mentzoni, Niels Hygum Nielsen, "Making the Milrem Themis UGV ready for autonomous operations," Proc. SPIE 11758, Unmanned Systems Technology XXIII, 117580S (12 April 2021); doi: 10.1117/12.2585879

Event: SPIE Defense + Commercial Sensing, 2021, Online Only

Making the Milrem Themis UGV ready for autonomous operations

Kim Mathiassen[a], Magnus Baksaas[a], Sindre Aas Græe[b], Eilert André Mentzoni[a], and Niels Hygum Nielsen[a]

[a]Norwegian Defence Research Establishment, Instituttveien 20, Norway
[b]Norwegian Defence Materiel Agency, Rødskiferveien 20, Norway

ABSTRACT

The usage of Unmanned Ground Vehicles (UGVs) in defence application is increasing, and much research effort is put into the field. Also, many defence vehicle producers are developing UGV platforms. However, the autonomy functionality of these systems are still in need of improvement. At the Norwegian Defence Research Establishment a project for developing an autonomous UGV was started in 2019 and use a Milrem THeMIS 4.5 from Milrem Robotics as the base platform for achieving this.

In this paper we will describe the modifications made to the vehicle to make it ready for autonomous operations. We have added three cameras and a Lidar as vision sensors, for navigation we have added a GNSS, IMU and velocity radar, and all sensors get a common time stamp from a time server. All the sensors have been mounted on a common aluminium profile, which is mounted in the front of the vehicle. The vision and navigation sensors have been mounted on the common aluminium profile to ensure that the direction the vision sensors observe is known with as little uncertainty as possible.

In addition to the hardware modification, a control software framework has been developed on top of Milrem's controller. The vehicle is interfaced using ROS2, and is controlled by sending velocity commands for each belt. We have developed a hardware abstraction module that interfaces the vehicle and adds some additional safety features, a trajectory tracking module and a ROS simulation framework. The control framework has been field tested and results will be shown in the paper.

Keywords: Unmanned Ground Vehicles, path following, autonomous vehicles, mobile robots, autonomous unicycle

1. INTRODUCTION

The usage of Unmanned Ground Vehicles (UGVs) in defence application is increasing, much research effort is put into the field, and many defence vehicle producers are developing UGV platforms. However, the autonomy functionality of these systems are still in need of improvement. At the Norwegian Defence Research Establishment (FFI) a project for developing an autonomous UGV was started in 2019, and use a Milrem THeMIS 4.5, produced by Milrem Robotics, as the base platform for achieving this.

Off-road and military ground robotics is an active research topic among several military and academic institutions. The Defense Advanced Research Projects Agency (DARPA) Grand Challenge of 2004[1] and 2005[2] were two major events in pushing research into outdoor robotics. In the first Grand Challenge the goal was to develop a mobile robot that could autonomously traverse an outdoor route of 140 miles which ran over dirt roads, trails, lakebeds, rocky terrain and gullies.[3] None of the 15 finalists completed the course and the robot driving the farthest drove only 7.4 miles. This clearly showed the difficulty in developing a mobile robot for off-road driving. In the second competition Stanley[4] won the competition, and four other teams completed the course.

The European Land Robot Trial (ELROB) is a competition that started in 2006.[5] It was designed to test the capabilities of unmanned systems in realistic scenarios,[6] and is a showcase for many academic and military

Further author information: (Send correspondence to K.M.)
K.M.: E-mail: kim.mathiassen@ffi.no

Unmanned Systems Technology XXIII, edited by Hoa G. Nguyen, Paul L. Muench,
Brian K. Skibba, Proc. of SPIE Vol. 11758, 117580S · © 2021 SPIE
CCC code: 0277-786X/21/$21 · doi: 10.1117/12.2585879

researchers. The trial is divided into different scenarios, which include reconnaissance and surveillance mission, transportation missions which can be carried out by a single vehicle or in a convoy of at least two vehicles. The autonomous vehicle MuCAR, based on a VW Touareg, has participated in ELROB a number of times,[7,8] and won the convoy scenario in 2016.[9] Another contestant is RTS-HANNA, which is based on the Kawasaki Mule 3010 side-by-side vehicle, and is capable of autonomous obstacle avoidance and object recognition as well as localization and map building.[10]

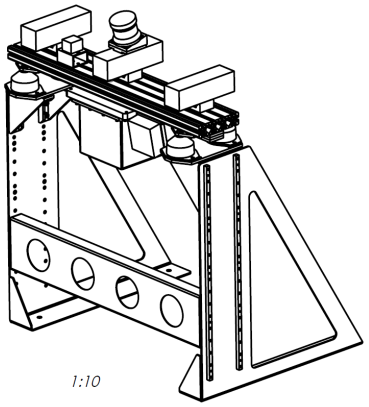
Apart from competitions for outdoor mobile robots several nations has developed their own systems for autonomous off-road driving for various military applications. The U.S. Army is developing autonomous ground vehicles, and they have split their development into vehicle specific kits (B-kits) and autonomy kits (A-kits).[11] The B-kits include drive-by-wire and safety features. B-kits have been developed for several different platforms, including HMMWV, MRZR, Jeep Rubicon, John Deere 6x6 Gator and CERV. The autonomy kits focuses on giving aid to the dismounted soldier. The soldier can order the vehicle to follow waypoints, follow a soldier or to move in front of a soldier.

The University of Defence in the Czech Republic has developed a system named TAROS.[12] It is a medium-sized UGV designed for combat and logistics support, reconnaissance missions, and special forces. A study[13] for the Spanish Army has developed a UGV using a military all-terrain vehicle already in service in the Spanish Army. They have developed two modes, one tele-operated mode and one autonomous mode, as requested by their military experts. The vehicle is intended for surveillance missions and is capable of following a dirt path. In a finish study[14] a UGV is developed as a capability demonstrator, focusing on achieving a high level of autonomy. A commercial off-road vehicle was selected as a UGV platform and necessary modifications were made to get drive-by-wire capabilities. The vehicle was demonstrated to complete a surveillance mission, driving mostly autonomously and some parts tele-operated.

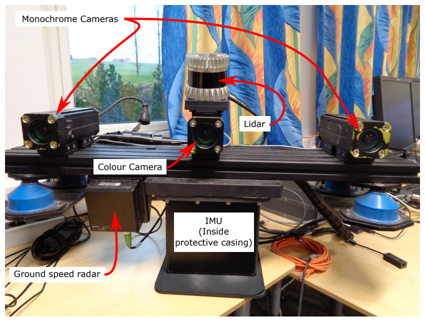
One of the core capabilities of autonomous vehicle is the ability to follow a path or trajectory given by the higher level control architecture. The vehicles are broadly divided into two categories; Ackermann vehicles or car-like vehicles that are modeled as bicycles[15], and differential drive or skid-steer vehicles that are modeled as unicycles[16]. However, modelling tracked vehicles as unicycles has some potential problems as variation of the relative velocity of the two tracks results in slippage as well as soil shearing and compacting in order to achieve steering.

Different researchers have tried to solve the slippage problem in different ways. In one study[17] a path follower method using a skid-steer wheeled vehicle was presented. This method first designs a controller using the nominal dynamics of the system and then a disturbance observer is added to the controller to compensate for unmodeled dynamics and terrain variation. For skid-steer vehicles the Instantaneous Center of Rotation (ICR) is the point where the motion of the vehicle can be represented by a rotation and no translation occurs, and the location of this point can be modeled.[16] In the studies[18,19] the ICR is estimated using an extended Kalman filter, and this was used to improve a traditional unicycle control strategy and for online odometry estimation. In another study[20] a traditional unicycle control strategy is extended by modelling and compensating for the longitudinal slip effect. A Model Predictive Control (MPC) strategy is used in another study.[21] Here the slip is modeled using six parameters and estimated online. These parameters are used in an augmented kinematic model of the vehicle, and the model is used to predict the control inputs. Active Disturbance Rejection Control is used in in another study[22] to online estimate and compensate for the effect of slip for path following using a skid-steer tracked vehicle.

In this paper we will describe the modifications made to the Milrem THeMIS to make it ready for autonomous operations, and transforming it into our test platform *Tor*. We have added three cameras and a Lidar as vision sensors, for navigation we have added a Global Navigation Satellite Systems (GNSS) receiver, Inertial Measurement Unit (IMU) and velocity radar, and all sensors get a common time stamp from a time server. All the sensors have been mounted on a common aluminium profile, which is mounted in the front of the vehicle. The vision and navigation sensors have been mounted on the common aluminium profile to ensure that the direction the vision sensors observe is known with as little uncertainty as possible. In addition to the hardware modification, a control software framework has been developed on top of Milrem's controller. The vehicle is interfaced using ROS2, and is controlled by sending velocity commands for each belt. We have developed a hardware abstraction component that interfaces the vehicle and adds some additional safety features, a path following component and



(a) CAD drawing of UGV sensor rig with vibration dampened sensor platform and sensors mounted



(b) Sensor platform with sensors in the lab

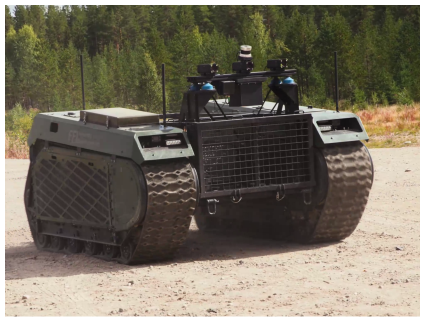

(c) UGV Tor with sensor-rig

Figure 1: Sensor platform design, assembly and mounting on Milrem THeMIS.

a ROS simulation framework. The path following component use a traditional unicycle controller. The control framework has been evaluated in simulation and field tests to verify if the traditional controller has adequate performance. The vehicle is intended as a research platform and possible application of the vehicle include base defence[23], communication relay[24], dismounted soldier teaming and Intelligence, Surveillance and Reconnaissance (ISR) missions.

2. HARDWARE MODIFICATIONS

To make the UGV ready for autonomous operation, we have mounted various equipment, both fabricated in-house and bought off the shelf. We started off by making a height adjustable sensor rig, that we mounted in the front part of the UGV on the cargo-bed. On this rig we mounted a vibration and shock isolated rigid platform, which we in turn mounted our sensors on to. The sensors we have used so far are two monochrome and one colour machine vision camera, one Lidar, one IMU and one ground speed Doppler radar. More details on the sensor-rig in Section 2.1.

Some of our goals by designing the sensor load this way, is to 1) make the sensor platform relatively easy to remove for either transport or for testing in a lab, 2) Easy to adjust sensor height relative to the UGV (560 mm span in 40 mm steps), 3) all sensors in a known and rigid position relative to each other.

We mounted two rugged computers on the UGV, one for low-level control and the other for scene analysis, as well as two GNSS-antennas, one as time-source and the other for navigation. We also mounted a rugged time synchronization server, and a self made Data Synchronization Unit (DSU), that uses a FPGA to timestamp raw data from the sensors without this capability built in. More details on the sensors themselves are found in Section 2.2, and other hardware in Section 2.3.

2.1 Sensor rig

A CAD-drawing of the sensor rig is shown in Figure 1a. The sensor platform is mounted on the sensor rig with silicone-rubber vibration and shock dampeners (blue), seen in Figure 1b. The whole assembly is also seen bolted to the cargo bed on Tor in Figure 1c.

The rig itself is fabricated in-house by the prototype workshop. The sensor platform, that the sensors are attached to, is made of an industrial extruded anodized aluminium profile[*], that measures $860 \times 160 \times 40$ mm.

[*]Profile 8, 160x40. MB building kit system, from *item Industrietechnik GmbH.*

Make	Model	Description
FLIR	BFS-PGE-31S4C-C	Blackfly S colour camera, 3.1 MP, Gigabit Ethernet interface
FLIR	BFS-PGE-23S3M-C	Two Blackfly S monochrome cameras, 2.3 MP, Gigabit Ethernet interface
Edmund Optics	Stock #68-669	Three 3.5mm FL Wide Angle Low Distortion Lenses, used on all cameras
Ouster	OS-1-64	Lidar, 64 beams, 850 nm, $\pm 16.6°$ vertical field of view, 360° horizontal field of view
RADARXSENSE	RXS-SOG-10	Ground speed radar, $11° \times 11°$ beam, mounted with $-10°$ downward tilt, measurement range ± 80 m/s, minimum speed measured 0.1 m/s, accuracy $\pm 0.5\%$
Honeywell	HG9900	IMU used for inertial navigation
uBlox	NEO M8T	GNSS receiver

Table 1: Sensors

2.2 Sensors

A close up image of the sensor platform with annotations can be seen in Figure 1b. Table 1 lists all sensors mounted with intended function, make and model. Table 2 lists other autonomy hardware mounted.

The cameras and Lidar are communicating directly with the scene analysis computer via Gigabit Ethernet. The cameras are powered via PoE[†]-enabled ports. The Lidar is powered by a FFI-made Power/Ethernet-splitter, which contains a DC/DC converter that feeds the Lidar with 24 VDC converted from the UGVs native 28 VDC. The IMU and radar are powered from, and communicating with, a FFI-made DSU. More on this unit in Section 2.3.

2.3 Other hardware

Table 2 lists other hardware things mounted on Tor, excluding sensors which are listed in Table 1. We have mounted two computers on Tor. One for scene analysis, and the other for low level control. The computers are both running Ubuntu 20.04 LTS Desktop version.

A rugged time synchronization server is also mounted, and receives time from GNSS if possible. This server also contains a OCXO[‡] oscillator that enables the unit to serve reliable time even in GNSS-denied environments. The time synchronization server is the master clock for the whole system, and serves time both via Ethernet by NTP and PTP protocols, NMEA-0183 serial link and/or frequency like $1PPS$.

The DSU, which powers and communicates with the IMU and radar sensors, is running headless embedded Linux. It contains FPGA-logic that timestamps the raw data from IMU and radar, before sending the data to the scene analysis computer via Ethernet. It also runs navigation software that calculates position and orientation based on input from GNSS and IMU data.

Two GNSS-antennas are mounted directly to the chassis on Tor. The reason for not placing them with the other sensors on the sensor platform was to avoid radio shadows caused by the sensor-rig and -platform, and that we experienced poor reception when GNSS antennas was placed too close to other sensors with high frequency data transfers in an earlier project. One of the antennas gives GPS-time to the time synchronization server, and the other serves positioning data to the DSU.

The cameras are all mounted in IP67-classified camera enclosures, and the IMU is mounted in a sound dampened solid aluminum box, for physical protection and audible noise reduction.

[†]Power over Ethernet

[‡]Oven Controlled crystal (Xtal) Oscillator

Make	Model	Description
Onlogic	K700-X2	Scene analysis computer. Rugged computer with Nvidia RTX-2060 GPU, Intel i7 CPU and PoE-ports that powers the cameras. Cameras and Lidar are directly connected, IMU and ground speed radar are connected via DSU that timestamps the raw data first.
SDK Embedded Systems	SDK-A77RMG	Control computer. Rugged IP67 certified computer. Runs ROS, which is used to control Tor.
FFI	DSU	Data Synchronization Unit. Unit using custom FPGA to timestamp raw data from IMU and ground speed radar, before sending the data on to the scene analysis computer. Time received from Time synchronization server. Also running navigation SW that calculates position using input from both GNSS-antenna and IMU data.
Oriola	VersaSync	Time synchronization server. Rugged GPS and frequency Master-clock source, that provides time via ethernet using NTP og PTP protocols, frequency like e.g. 1 PPS, or by serial link using NMEA-0183.
Tallysman	TW3972	Two GNSS antennas. One is connected to the time synchronization server, the other to the DSU. Can receive Beidou, Galileo, GLONASS and GPS signals.
autoVimation GmbH	Salamander	Three Compact IP67 camera enclosures for each camera in Table 1, profile M, IR coated mirrors. Bodies anodized black, lids painted due to rubber seals.

Table 2: Other equipment used for autonomy purposes

3. SOFTWARE MODIFICATIONS

In this section we will describe the software addition we have made to the Milrem THeMIS platform. This paper only describes the foundation for making the THeMIS autonomous, and will be incorporated into a larger autonomy framework. Figure 2 shows the planned framework, and build upon previous work[25] at FFI. The solid boxes are components described and evaluated in this paper, while dashed boxes are components that will be integrated in the future. The Robotic Operating System (ROS) is used as a middleware in the system.

In this section we will start with the *Low-level controller*. In this case the Milrem THeMIS' proprietary controller is used, and we have created a software abstraction layer between the Milrem THeMIS interface, which is based on ROS2, and our software framework, which is based on ROS1. Next is the *Localization* component, which is responsible for finding the vehicle's local and global position. The *Path follower* component is responsible for ensuring that the vehicle is located on the planned path. It receives a set of positions and headings and makes a continuous path between the received waypoints. In addition to the components in Figure 2 we have created a simulator that emulates the Milrem THeMIS. The above mentioned components will be described in the following section.

Later we will add further autonomy components to the system. The *Motion planning* component will be based on the Traversability Hybrid A*[26] method, with some adaptions. This component receives a route from the *Decision making* which the vehicle should follow, and plans a local path based on a traversability map. The map is created in the *Perception* components, which process sensor data into the map.[27]

The *Decision making* component will be using the Hybrid Autonomy Layer (HAL),[28] which is a hybrid control system architecture with two layers. The top layer is a discrete task layer which decompose a complex task into a task tree. The bottom layer is a continuous behavior layer which execute different behaviors at a fixed update frequency, for instance control laws or monitoring.

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.[29] The bridge converts between ROS messages and UGV

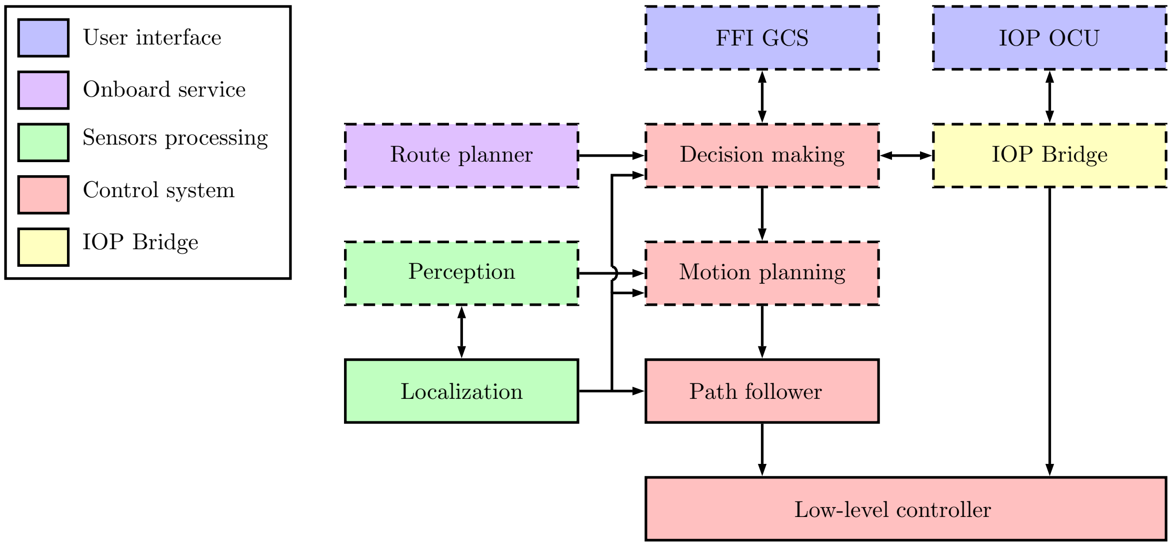


Figure 2: Software architecture. Components in solid lines are covered in this paper, while components in dotted lines are planned extensions.

IOP messages. Using the bridge it is possible for UGV IOP compliant Operator Control Units (OCUs) to control the robot.

3.1 Vehicle abstraction

Milrem Robotics has provided a set of ROS2 messages for communicating with the vehicle. The ROS2 messages gives us the same possibilities for controlling the vehicle as with Milrem Robotics' remote controller. The messages make it possible to steer the vehicle, turn the diesel generator on and off and select different modes, control the lights and IOs, etc. We have created an abstraction software between the vehicle's software and our autonomy software that simplify the control of the vehicle so that the autonomy software only have to turn the vehicle on and send commands. The abstraction software keeps the vehicle alive and hide all vehicle specific commands. We also want to control the vehicle with our own remote controller and using IOP. The different controllers are managed by a MUX that is controlled by the Decision Making software. Figure 3 shows the vehicle abstraction software, called THeMIS Controller. The abstraction software is divided into three parts, a user interface, a logical layer and a vehicle interface. The user interface converts different controller inputs to a set of known commands. The vehicle interface talks with the vehicle and convert the commands into vehicle messages. The logical layer is receiving all user commands and sending them to the vehicle interface. The logical layer handles the vehicle status, i.e. if the vehicle is off, on or in emergency stop. Figure 4 shows the states that represents the vehicle in the logical layer. The abstraction software publishes only messages to the vehicle when it is in OFF or ON.

3.2 Localization

The navigation system consists of a Inertial Measurement Unit (IMU) and a Global Navigation Satellite System (GNSS) that are processed in the Inertial Navigation System (INS) NavP. At the vehicle, a Honeywell HG9900 IMU and a u-Blox NEO M8T GNSS receiver are installed. NavP is developed at FFI and is a real time solution of NavLab,[30] and it has been used for years in the FFI's Autonomous Underwater Vehicle (AUV) HUGIN. The INS gives a high accuracy position and orientation for the vehicle control and perception system at 300 Hz. Data from the INS is sent in UDP messages that are received in a ROS-node that redistribute the data as the standard ROS messages. The messages consists of a global position, i.e. latitude and longitude represented in WGS84 and height represented in Mean Sea Level (MSL) and a global orientation of the vehicle's body frame (front-right-down – FRD) to the local level frame (North-East-Down – NED).

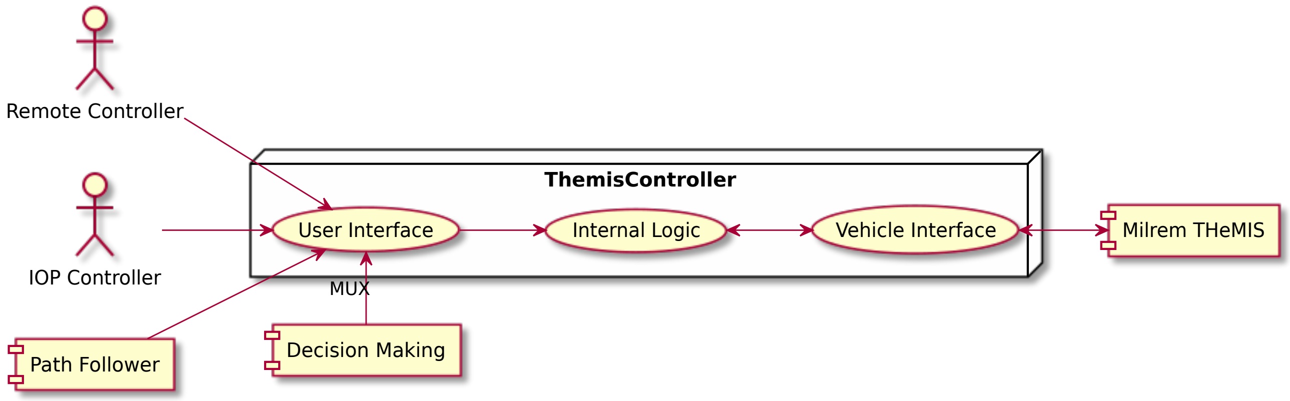
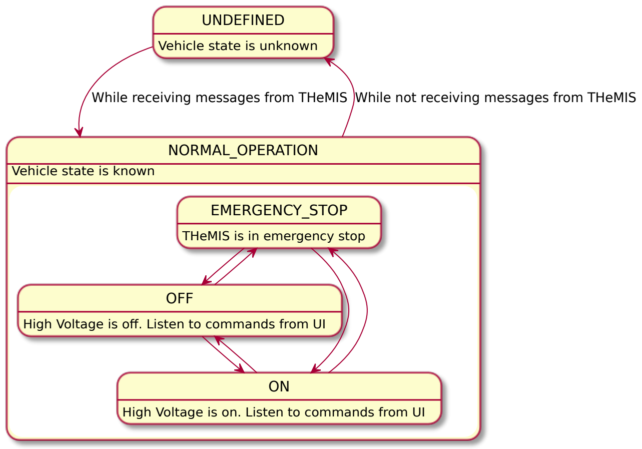


Figure 3: Vehicle Interface



UNDEFINED
Vehicle state is unknown
While receiving messages from THeMIS
While not receiving messages from THeMIS
NORMAL_OPERATION
Vehicle state is known
EMERGENCY_STOP
THeMIS is in emergency stop
OFF
High Voltage is off. Listen to commands from UI
ON
High Voltage is on. Listen to commands from UI

Figure 4: Vehicle states

3.3 Path following

The path following module operates in a local Cartesian frame and accepts a set of waypoints. The waypoints are Cartesian x and y positions and a vehicle heading θ, all in an East-North-Down (ENU) frame. The module interpolates a path between the waypoints, and ensures that the vehicle follows this path. Currently the vehicle has a constant forward velocity while following the path, but in the future the vehicle should adapts its forward velocity. The module consist of two parts, one for interpolating a path between the waypoints and one for following the path.

The path following module assumes the unicycle model

$$\dot{x} = v\cos\theta \tag{1a}$$
$$\dot{y} = v\sin\theta \tag{1b}$$
$$\dot{\theta} = \omega \tag{1c}$$

where x, y and θ are the configuration variables of the system. The two first represents the vehicles position in a Cartesian space and the last represents the vehicle's heading. The control inputs v and ω are the forward velocity and rotational velocity respectively. These values need to be converted into the speed for each belt, and this will be explained later in this section.

3.3.1 Path representation

Finding a path between the waypoints depends on the waypoints, and three different cases are considered in this paper; driving forward, driving backwards and neutral turn. The interpolation scheme, or path representation, depends on the case, and two different representations taken from[31] has been used.

Cartesian polynomials For driving forward and backwards the Cartesian polynomials are used for interpolating between two waypoints. Given an initial state $\boldsymbol{x}_i = [x_i, y_i, \theta_i]^T$ and a final state $\boldsymbol{x}_f = [x_f, y_f, \theta_f]^T$ the path between these two waypoints is given as[31]

$$x(s) = s^3 x_f - (s-1)^3 x_i + \alpha_x s^2 (s-1)^2 + \beta_x s (s-1)^2 \tag{2a}$$
$$y(s) = s^3 y_f - (s-1)^3 y_i + \alpha_y s^2 (s-1)^2 + \beta_y s (s-1)^2 \tag{2b}$$

where $s \in [0, 1]$ and represents the vehicle's position on the path and

$$\alpha_x = k \cos\theta_f - 3x_f \qquad \alpha_y = k \sin\theta_f - 3x_f \tag{3a}$$
$$\beta_x = k \cos\theta_i + 3x_i \qquad \beta_y = k \sin\theta_i + 3x_i \tag{3b}$$

Because $x(s)$ and $y(s)$ are *flat outputs* for the unicycle the heading, forward velocity and rotational velocity can be calculated based on these functions, and are given as

$$v_n = \sqrt{(x'(s))^2 + (y'(s))^2} \tag{4a}$$
$$\omega_n = \frac{y''(s)x'(s) - x''(s)y'(s)}{(x'(s))^2 + (y'(s))^2} \tag{4b}$$

The above equations hold both for driving forward and backwards. In the later case the sign of the velocity is changed, and the heading is in the opposite direction(e.g by adding π to the heading).[31]

The parameter k is a free parameter and influences the path. The parameter can be considered a geometric velocity parameter, and the higher the parameter is the longer will the vehicle follow the initial heading direction in the beginning of the path. In the current implementation of the method we have chosen to scale k with the distance between the waypoints, but at the same time limit k to a maximal value k_{max}. This is achieved with the following equation

$$k = \max\left(k_{max}, \frac{1}{2}\sqrt{(x_i - x_f)^2 + (y_i - y_f)^2}\right) \tag{5}$$

Chained form The above path representation does not permit neutral turns. If two consecutive waypoints are located in the same position, the above method would send the vehicle in a circle. Instead a method using the *chained form* is used for this particular case. In this case the change in heading is explicitly represented as a linear function of s. The transformation we use in this paper, from the unicycle model to the chained form, incorporates the initial position of the vehicle and the final heading, and is given as[31]

$$z_1 = \theta - \theta_f \tag{6a}$$
$$z_2 = (x - x_i)\cos\theta + (y - y_i)\sin\theta \tag{6b}$$
$$z_3 = (x - x_i)\sin\theta - (y - y_i)\cos\theta \tag{6c}$$

Given the above equations the initial state $\boldsymbol{x}_i$ and final state $\boldsymbol{x}_f$ is converted to the chained form initial state $\boldsymbol{z}_i$ and final state $\boldsymbol{z}_i$. The path between these points are given as

$$z_1(s) = z_{1,f}s - (s-1)z_{1,i} \tag{7a}$$
$$z_3(s) = s^3 z_{3,f} - (s-1)^3 z_{3,i} + \alpha_3 s^2 (s-1)^2 + \beta_3 s (s-1)^2 \tag{7b}$$

where

$$\alpha_3 = z_{2,f}(z_{1,f} - z_{1,i}) - 3z_{3,f} \tag{8a}$$
$$\beta_3 = z_{2,i}(z_{1,f} - z_{1,i}) - 3z_{3,i} \tag{8b}$$

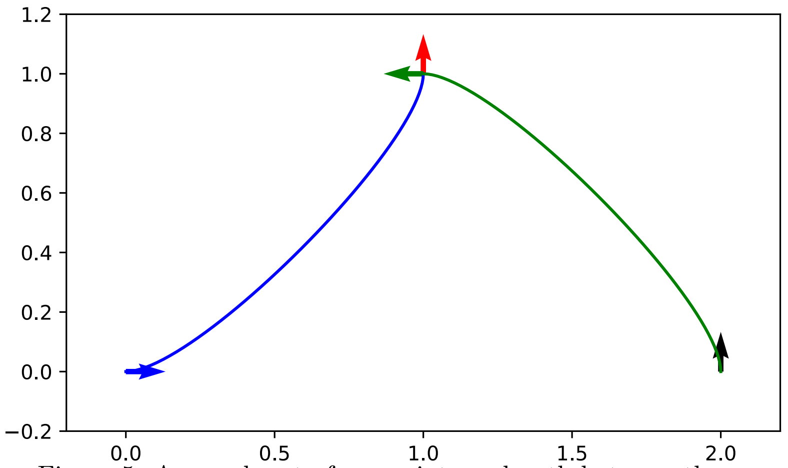


Figure 5: A sample set of waypoints and path between them.

In this case $z_1(s)$ and $z_3(s)$ are flat outputs for the unicycle, and the remaining state is found as $z_2(s) = \frac{z_3'(s)}{z_1'(s)}$. The control input is converted to the chained for by the following transform.

$$v_n = v_2 + z_3 v_1 \tag{9a}$$

$$\omega_n = v_1 \tag{9b}$$

The transformed control input can be found with the flat outputs as

$$v_1(s) = z_1'(s) \tag{10a}$$

$$v_2(s) = \frac{z_3''(s)}{z_1'(s)} \tag{10b}$$

With the above equations the path, configuration states and control inputs for the unicycle are uniquely defined between the two points $\boldsymbol{x}_i$ and $\boldsymbol{x}_f$.

Generating a path from a discrete set of points The path following module receives a list of waypoints that is should follow, and each waypoint consist of a position and heading. After receiving the list of waypoints, a path is generated in the module. Each two consecutive waypoints is checked if they are co-located. If this is the case then the chained form representation is used, and the parameters for this representation calculated and stored.

For the other waypoints the Cartesian polynomials representation is used, but a check is needed to determine if the vehicle should drive forwards or backwards. If the second waypoint lies behind the first waypoint, then the vehicle should reverse. Behind is defined by the direction of the heading of the first waypoint. A line tangent to the heading direction of the first waypoint, which pass through the position in the first waypoint, defines which points that are behind or in front of the first waypoint. After this check the parameters for the path is calculated and stored.

In Figure 5 the three cases are shown. The first waypoint located in (0,0) with a heading of 0 and shown in blue, the second is located in (1,1) with a heading of $\frac{\pi}{2}$, and shown in red. Between these waypoints the Cartesian polynomials representation is used and the vehicle drives forward. The path is shown in blue. The third waypoint is located in (1,1) with a heading of π, and shown in green. The chained form representation is used between the second and third waypoint, because the waypoints are co-located. The fourth and last waypoint is located in (2,0) with a heading of $\frac{\pi}{2}$ and shown in black. As the fourth waypoint lies behind the third, the vehicle will reverse using the Cartesian polynomials representation. The path is shown in green.

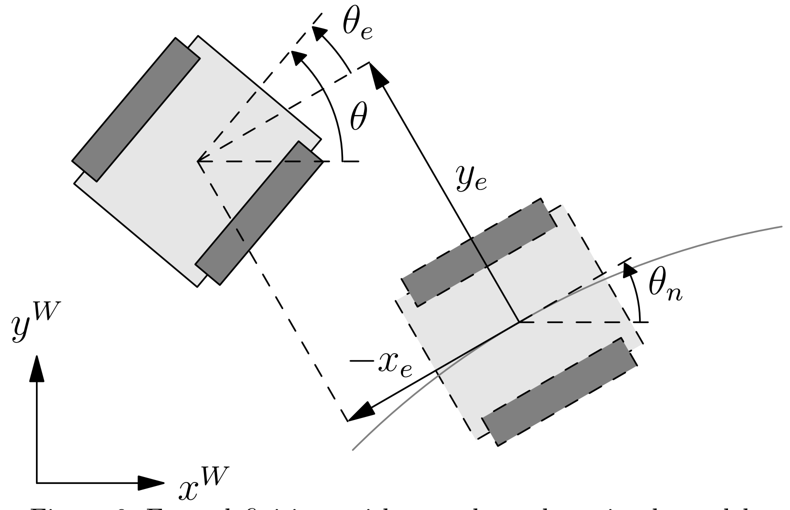


Figure 6: Error definitions with regards to the unicycle model.

3.3.2 Path following

The path following method used in this paper is found in Chapter 34.4.2 in the Handbook of Robotics[32]. The kinematic model for our system is given in (1) where v is the commanded forward velocity and ω is the commanded rotation velocity of the vehicle. Both these values are converted to velocity for each belt and then sent to the low level proprietary controller. First the path following method will be described, and then the method for scaling the velocity sent to the low level controller will be described.

Path following controller The path following method assumes a reference vehicle that is subject to the same constraints as the actual vehicle. The reference vehicle's position, orientation and control inputs are obtained from the paths described in the previous section, but how the reference vehicle's location on the path is updated will come later in this section. The errors are shown in Figure 6, and include the lateral error y_e, longitudinal error x_e, heading error θ_e. Based on the reference vehicle, the error dynamics using the reference vehicle frame have been found to be[32]

$$\dot{x}_e = \omega_n y_e + v\cos(\theta_e) - v_n \tag{11a}$$

$$\dot{y}_e = -\omega_n x_e + v\sin(\theta_e) \tag{11b}$$

$$\dot{\theta}_e = \omega - \omega_n \tag{11c}$$

where subscript e denotes errors and subscript n denotes nominal values from the reference vehicle. x_e represents how far the vehicle is behind the reference vehicle, y_e represents the off track error and θ_e represents the heading error (i.e. $\theta - \theta_n$), v_n represents the nominal forward velocity and ω_m represents the nominal rotation velocity. Using the following change of coordinates and control variables

$$z_1 = x_e \tag{12a}$$

$$z_2 = y_e \tag{12b}$$

$$z_3 = \tan(\theta_e) \tag{12c}$$

$$w_1 = v\cos(\theta_e) - v_n \tag{12d}$$

$$w_2 = \frac{\omega - \omega_n}{\cos^2(\theta_e)} \tag{12e}$$

yields the system

$$\dot{z}_1 = \omega_n z_2 + w_1 \tag{13a}$$
$$\dot{z}_2 = -\omega_n z_1 + v_n z_3 + w_1 z_3 \tag{13b}$$
$$\dot{z}_3 = w_2 \tag{13c}$$

The following non-linear controller has been previously proposed and shown to give a globally asymptotically stable system[32]

$$w_1 = -k_1 |v_n| (z_1 + z_2 z_3) \tag{14a}$$
$$w_2 = -k_2 v_n z_2 - k_3 |v_n| z_3 \tag{14b}$$

In this paper we will use the above method as a path following method, rather than a trajectory tracking method, i.e. that we do not require the vehicle to be at a specific point at the trajectory at a specific time, but rather wants the vehicle to follow the path using a reference velocity. This means that we can disregard w_1 and only use w_2 for controlling the rotation velocity ω. Rearranging for ω yields the controller

$$\omega = w_2 \cos^2(\theta_e) + \omega_n \tag{15}$$

This means that the reference rotation velocity ω_n is used as a feed forward action and that proportional controller given in w_2 is scaled depending on the heading error θ_e. The controller will stop working if the heading error exceeds $\frac{\pi}{2}$.

Determining reference vehicle position In order to calculate the control input the reference vehicle's location on the path must be found. Since we are using the method as a path following method we cannot use time to shift the reference vehicle position along the path. Instead we iterate over the possible positions on the path, and find the first position that has closeness to the path below a certain threshold and satisfies a set of safety constraints. The closeness to the path can be two different factors depending on which path representation that is used. When the Cartesian polynomials representation is used, the vehicle is driving forward or backwards and the longitudinal error x_e is used to determine closeness. When the chained form representation is used, the vehicle is performing a neutral turn and the heading error θ_e is used to determine closeness. The threshold is $x_{e,max}$ in the first case and $\theta_{e,max}$ in the second case.

The location on the path also has to satisfy two safety constraints. The total position difference between the actual vehicle and the reference vehicle must be below the threshold $r_{e,safe}$ and the heading error θ_e have to be below the threshold $\theta_{e,safe}$.

When finding the new location on the path the parameter s is increased for each iteration, and the conditions are checked. If the conditions are not satisfied the parameter is increased by s_{step}. For new paths s start at zero, while for existing paths s from the previous time step of the controller is used.

Converting to belt speed and scaling velocity The control method finds the control input for a unicycle, namely forward velocity v and rotational velocity ω, while the Milrem THeMIS proprietary velocity control require velocity for each belt. Let $v_{b,r}$ and $v_{b,l}$ be the velocity for the right and left belt, and the following conversions are used[31]

$$v_{b,r} = \omega \frac{L}{2} + v \qquad\qquad v_{b,l} = 2v - v_{b,r} \tag{16}$$

where L is the distance between the tracks.

While following a path the vehicle is commanded to hold a constant forward velocity v_{cmd}. The nominal control input from the path is scaled according to this before it is used in the controller. When scaling the control input, both the forward and rotational velocity needs to be multiplied with a scaling factor. In this case the scaling factor is $|\frac{v_{cmd}}{v_n}|$.

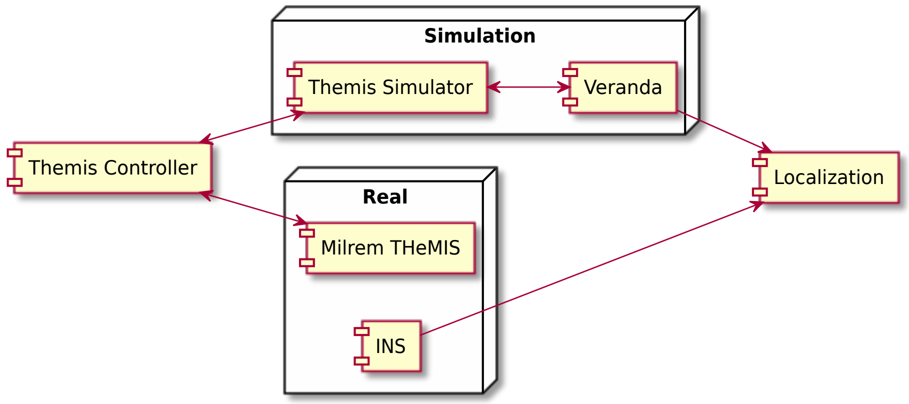


Figure 7: Illustration of data data flow.

After the control input is calculated from the controller, a series of checks are made to ensure that the velocity commands sent to the vehicle is below certain thresholds. If they are above the threshold the control input is scaled so that the velocity is at the threshold. First the rotational velocity is checked if it is above ω_{cmd}. If it is, forward velocity v and rotational velocity ω are scaled down. If the vehicle performs a neutral turn then ω_{cmd} will be the rotational velocity in that turn. After this check the control input is converted to belt speed. If one or both of the belt speeds are above $v_{b,max}$ the velocity is scaled down.

3.4 Simulation

In order to ease test and development of ROS-nodes on Tor, we have built a Software-In-the-Loop (SIL) simulation environment where Tor is simulated in a 2D world. An illustration of the components and data flow in the SIL simulation environment can be seen in Figure 7. In the SIL simulation environment, the behaviour and ROS2 interface of the Milrem THeMIS controller is simulated by the node THeMIS Simulator. In particular THeMIS Simulator publishes rotation speed of the two tracks which is being used by the ROS2 robot physics simulator Veranda[33, 34] to do a 2D physics simulation of Tor. In Veranda Tor is being modelled as a two wheeled robot where the density of the two "wheels" have been tuned such that the simulated behaviour in Veranda is similar to Tor driving on an even grass field. In Veranda some physics coefficients like the friction-coefficient is set to 1 which means that the simulated friction forces can essentially be tuned by tuning the density. The parameters used for simulating Tor in Veranda are listed in table 4. Veranda also generates simulated GPS and IMU data which are used by the localization node to publish position and orientation messages.

4. EXPERIMENTS

This section describes the experiments conducted to verify the path following methods presented in the previous section. First the simulations are presented, then the real world experiments.

4.1 Simulations

Two different experiments are conducted in simulation. In the first experiment the simulated vehicle is commanded to drive three straight paths. The first path starts at the same point as the vehicle and points in the same direction and extends 30 m forward. The second and third paths are similar, but the vehicle has an offset of 0.5 m and 2 m to the side of the path, and the path is 30 m and 40 m long, respectively. The first path will check if the controller manages to follow the path, while the second and third will check how the controller manages a large position deviation.

The second experiment will check how the controller performs with a longer path. Since the first experiment only uses the Cartesian polynomials representation and is going forward, this second experiment will also include reverse and neutral turns. The path consist of nine waypoints, and is more than 160 m long and is shown along with the results.

Parameter	Value	Comment
$x_{e,max}$	0.05 m	Max position error when searching for a new reference vehicle position
$\theta_{e,max}$	0.01	Max orientation error when searching for a new reference vehicle position
s_{step}	0.005	Step size of s when searching for a new reference vehicle position
$r_{e,safe}$	5.0 m	Position safety constraint for new reference vehicle position
$\theta_{e,safe}$	$\frac{\pi}{2}$	Orientation safety constraint for new reference vehicle position
k_{max}	$5^{\dagger}$	Pseudo-velocity parameter for Cartesian polynomials representation
v_{cmd}	1 m/s	Reference forward velocity
ω_{cmd}	$\frac{1}{8}\pi$	Max rotational velocity, and reference rotational velocity for neutral turns
$v_{b,max}$	6 m/s	Max belt velocity
k_2	0.1	Control gain for position error
k_3	0.7	Control gain for orientation error

Table 3: Parameters for path following methods used in the experiments

Wheel radius [m]	Wheel width [m]	Wheel density [m^{-2}]	Base width [m]	Base height [m]
0.5	0.3	10	2	2.4

Table 4: Parameters for the THeMIS 2D physics model used in Veranda.

The parameters used for the path following method in the simulation is given in Table 3 and the parameters used for simulating the vehicle is given in Table 4. Note that in the second simulation the parameter k_{max} is set to infinite (and therefore marked with †), as this gives smoother paths when the distance between the waypoints is long.

4.2 Real world experiments

For the real world experiments the first simulation experiment is repeated. The vehicle was driven in a snowy field, with approximately 10 cm of wet snow. It was also a slight upwards slope. The vehicle was commanded to follow a straight line, using an offset of 0 m, 0.5 m and 2 m and a line length of 30 m, 30 m and 40 m. The same parameters was used as in the simulation, given in Table 3.

5. RESULTS

5.1 Simulations

The results of the three paths driven in the simulations are given in Figure 8. The blue line ($-k_2 y_e$) indicates the control action caused by a cross track error, the orange line ($-k_3 \tan\theta_e$) indicates the control action caused by a heading error, and the green line (w_2) indicates the combined control action, as given in (14). Since the nominal velocity equals one, this term is removed in the plot's legends. From (15) we see that the control action w_2 is scaled down by $\cos^2(\theta_e)$, and this is most relevant when there are large heading errors. The red line (ω_n) shows the nominal control action. As the control gain k_2 equals 0.1, we get the position error in meters from the blue line by multiplying the y-axis by ten.

The path driven in the second simulation experiment is given in Figure 9. When arriving to the position (20,40) the vehicle performs a neutral turn and is then driven backwards up to (20,50). The rest of the path is driven forward. The maximal cross track error (y_e) and heading error (θ_e) while following the path were 0.26 m and 0.177 rad (10.14°), respectively.

Figure 10a shows the control actions when driving, and Figure 10b shows the control velocities. After driving in approximately 75 s, the vehicle starts to perform a neutral turn, and one can see that the rotational velocity is approximately 0.4 rad/s and the forward velocity 0 m/s. After approximately 90 s the vehicle starts to reverse and the forward velocity is -1 m/s. At some points the forward velocity is reduced. This is when the rotational velocity exceeds the velocity limit ω_{cmd}. This happens in the transitions between waypoints, as the rotational velocity does not have any constraint at the waypoints. Increasing the parameter k will mitigate this, as one would get a longer slope towards the waypoint and thus reduce the rotational velocity.

5.2 Real world experiments

The results of the three paths driven in the real world experiment are given in Figure 11. The plots are identical to the simulated case, except for using the real world data. One thing to note is that the nominal rotational velocity has a large negative value in the beginning of the path and a large positive value in the end of the path. This occurs sometimes with the Cartesian polynomials representation method, and is discussed in the next section.

6. CONCLUSION AND FURTHER WORK

In the result of the first simulation experiment, where the vehicle should follow a straight line, one clearly sees that the errors goes to zero. In the real world experiment the errors are also going towards zero, but there is more noise in the signals. An INS is used in the real world experiment, and this is the main cause of the noise. The position errors are within the errors from the INS system.

In the transitions between waypoints, and sometimes in the beginning or end of a path, there is sometimes a large nominal rotational velocity. This comes from the Cartesian polynomials representation. It is a third order polynomial, meaning that the derivatives of the positions are second order functions. This also causes the velocity to increase in the beginning of the segment and decrease towards the end of the segment. When two of the parameters in the polynomial match, they will cancel each other out and make a linear nominal rotational velocity. A way to mitigate this unwanted behavior is to choose a different k parameter. If we use the distance between the waypoints, rather than half the distance, the forward velocity will likely be around the parameter k rather than increase. Increasing k further will cause the forward velocity to decrease in the beginning of the segment and increase at the end. Further work is needed to verify and test these suggestions.

The second simulation experiment, where the vehicle drove a longer path, shows that the path following method presented in this paper is able to drive forward, backwards and perform neutral turns, and has a relatively low maximum position error. It is likely that this error can be further decreased if the issue mentioned in the above paragraph is solved.

In this paper we have presented the hardware and software modification we have made to a Milrem THeMIS platform in order to make i ready for autonomous operations. We have added perception and localization sensors and computers and made a sensor platform to mount all the sensors on. We have also implemented a path following module and created a ROS2 simulator of the Milrem THeMIS. Some verification of the performance of the path following module has been shown in this paper, and further work will be to field test the method more extensively.

REFERENCES

[1] Behringer, R., "The DARPA grand challenge - autonomous ground vehicles in the desert," *IFAC Proceedings Volumes* **37**, 904–909 DOI: 10.1016/s1474-6670(17)32095-5.

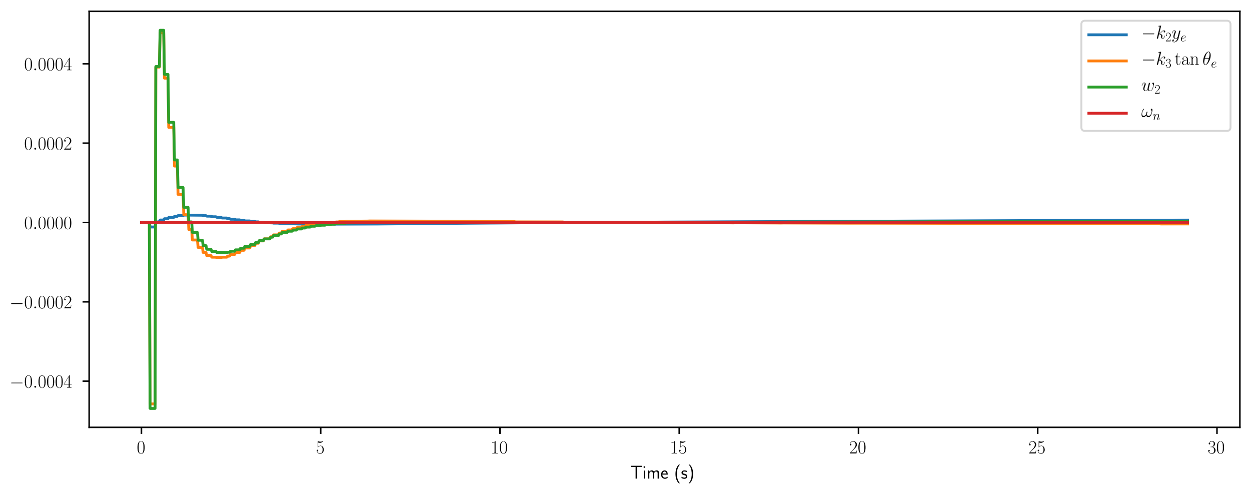
[2] Buehler, M., Iagnemma, K., and Singh, S., eds., [*The 2005 DARPA Grand Challenge: The Great Robot Race*], vol. 36, Springer DOI: 10.1007/978-3-540-73429-1.

[3] Iagnemma, K. and Buehler, M., "Editorial for journal of field robotics—special issue on the darpa grand challenge," *Journal of Field Robotics* **23**(9), 655–656 DOI: 10.1002/rob.20154.

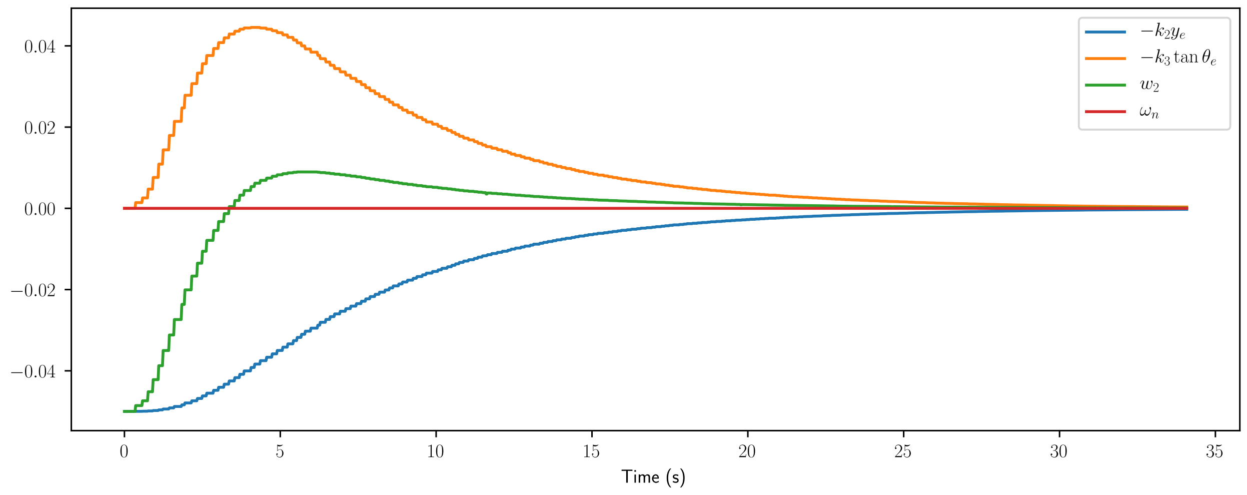
[4] Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., Fong, P., Gale, J., Halpenny, M., Hoffmann, G., Lau, K., Oakley, C., Palatucci, M., Pratt, V., Stang, P., Strohband, S., Dupont, C., Jendrossek, L.-E., Koelen, C., Markey, C., Rummel, C., van Niekerk, J., Jensen, E., Alessandrini, P., Bradski, G., Davies, B., Ettinger, S., Kaehler, A., Nefian, A., and Mahoney, P., "Stanley: The robot that won the DARPA grand challenge," *Journal of Field Robotics* **23**(9), 661–692 DOI: 10.1002/rob.20147.

[5] Schneider, F. E., Wildermuth, D., and Wolf, H.-L., "Professional ground robotic competitions from an educational perspective: A consideration using the example of the european land robot trial (elrob)," in [*2012 6th IEEE International Conference Intelligent Systems*], 399–405, IEEE DOI: 10.1109/is.2012.6335168.

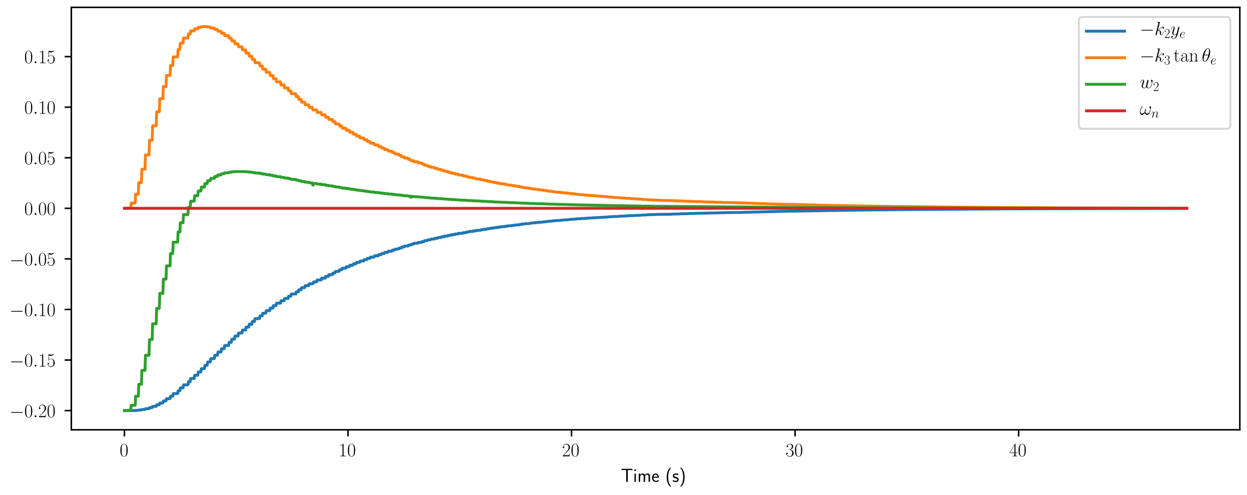
[6] Schneider, F. E., Wildermuth, D., and Wolf, H.-L., "ELROB and EURATHLON: Improving search & rescue robotics through real-world robot competitions," in [*2015 10th International Workshop on Robot Motion and Control (RoMoCo)*], IEEE DOI: 10.1109/romoco.2015.7219722.



(a) 0 m offset



(b) 0.5 m offset



(c) 2 m offset

Figure 8: Simulation results of driving a straight line with and without an offset in the vehicle's starting position perpendicular to the line.

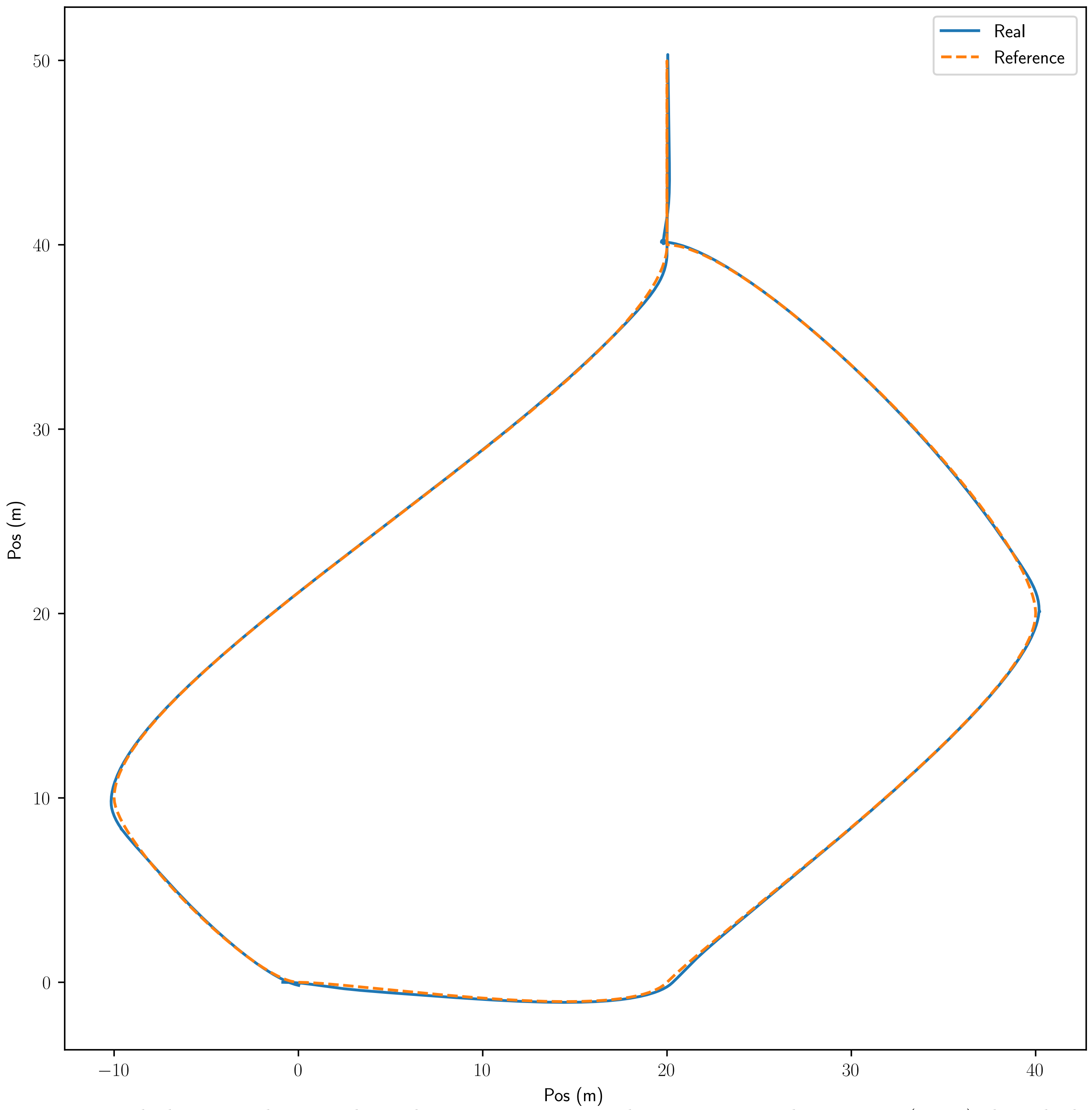
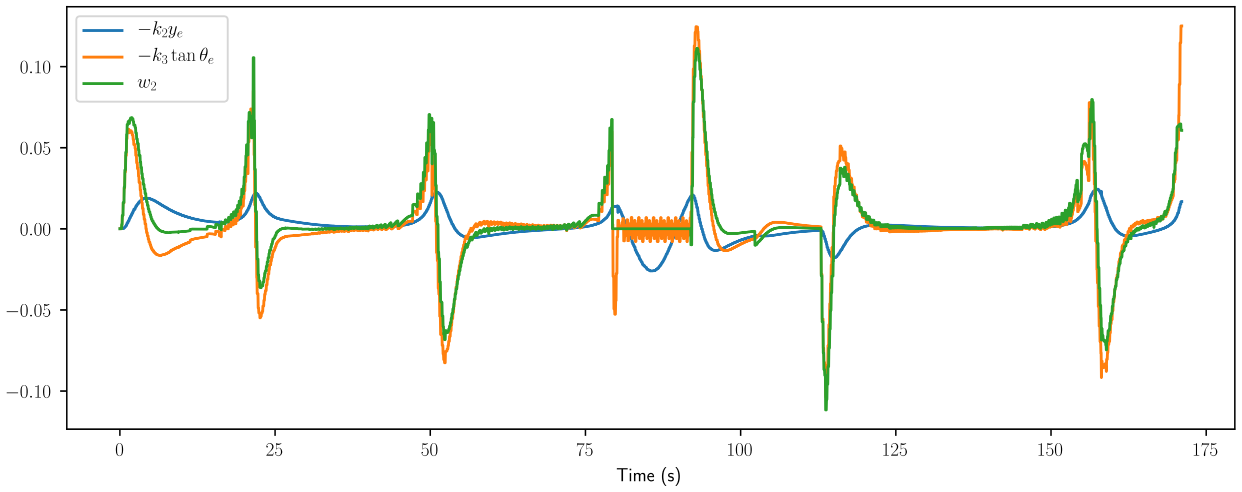
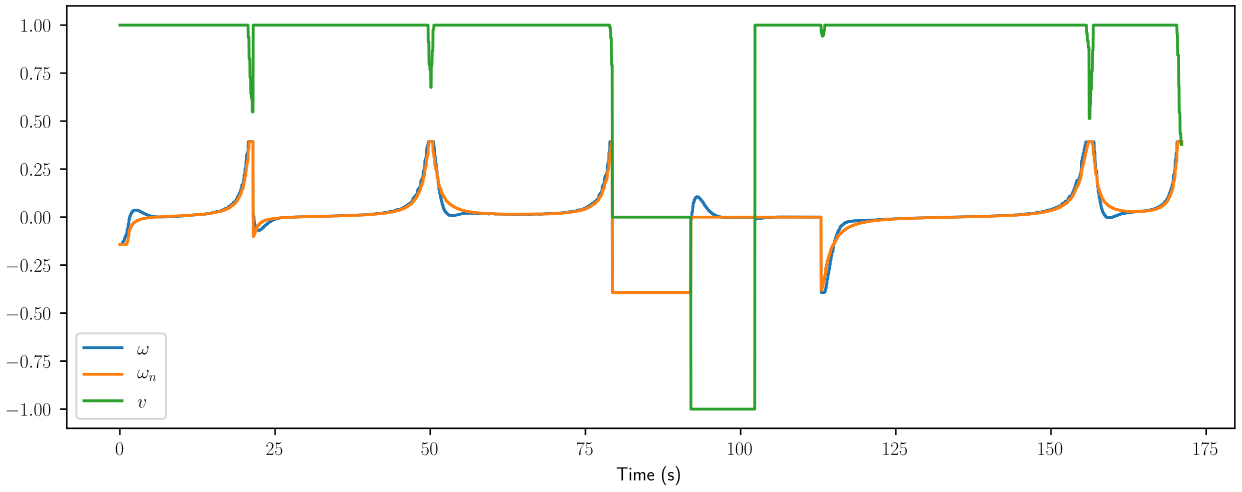


Figure 9: Path driven in the second simulation experiment. When arriving to the position (20,40) the vehicle performs a neutral turn and then driven backwards up to (20,50). The rest of the path is driven forward.

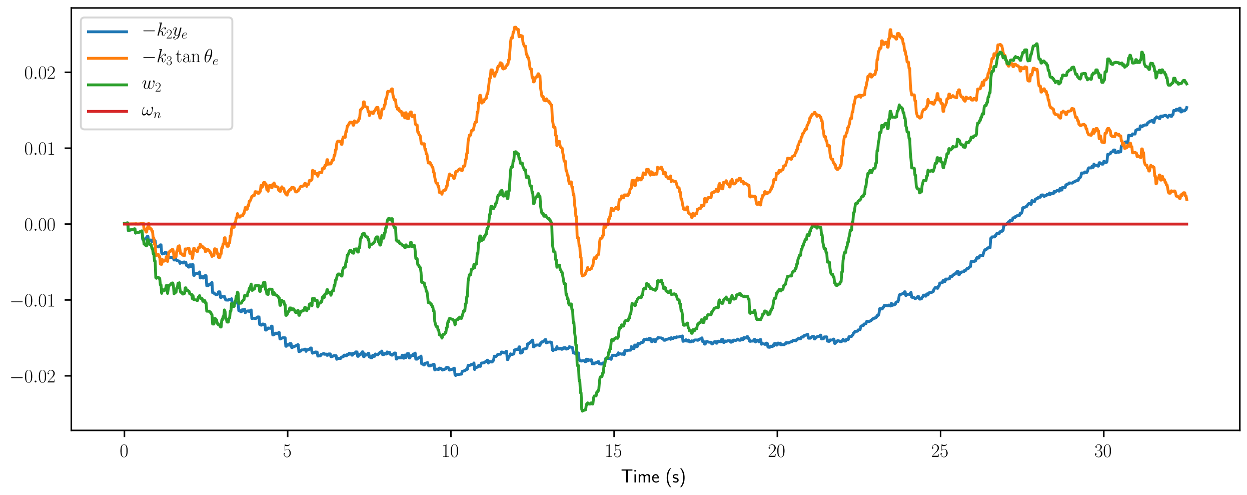


(a) Control actions

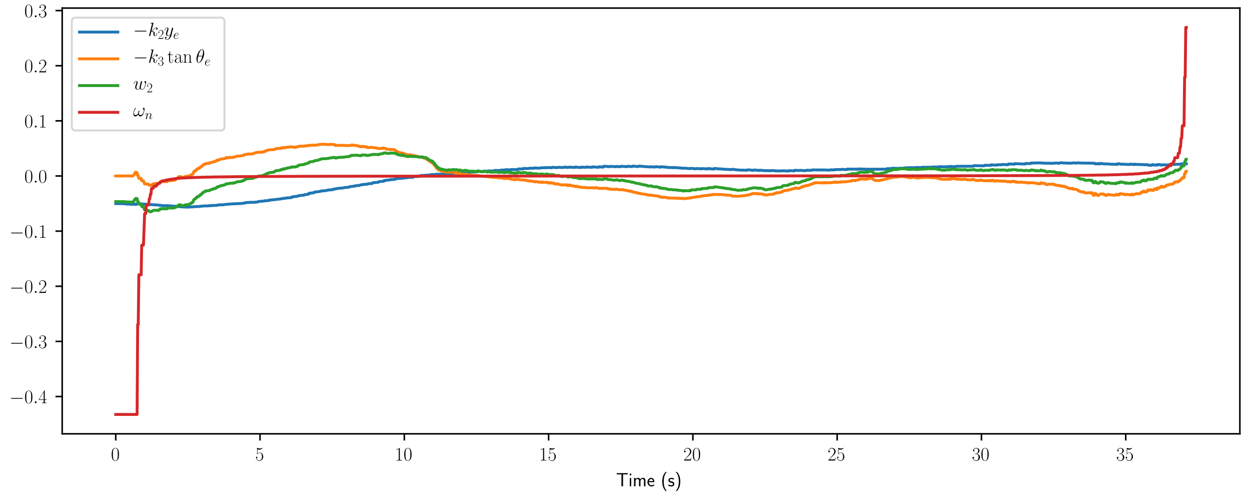


(b) Control velocities

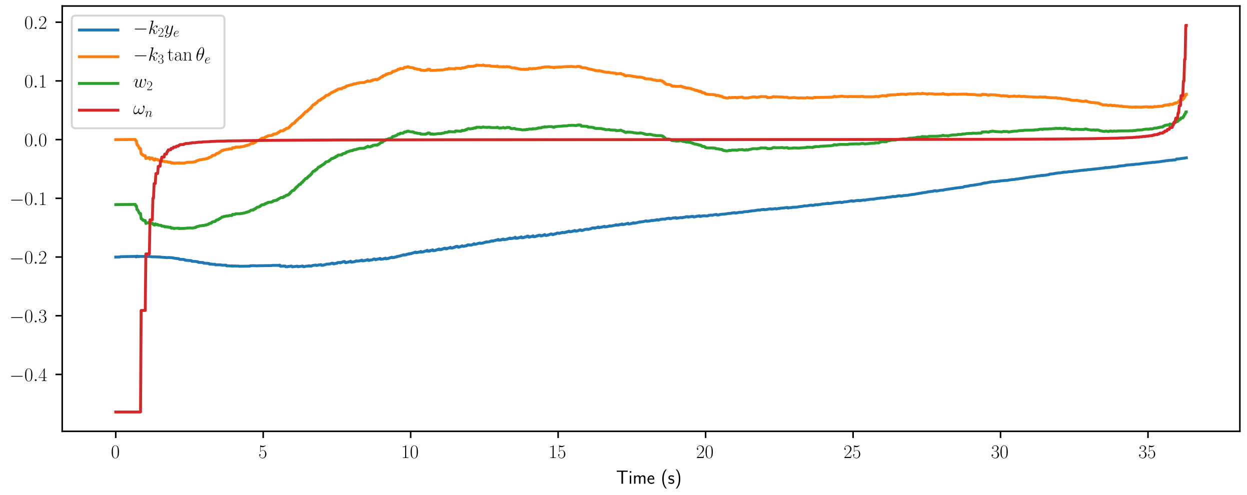
Figure 10: Simulation results when driving a longer path.



(a) 0 m offset



(b) 0.5 m offset



(c) 2 m offset

Figure 11: Real world experiment results of driving a straight line with and without an offset in the vehicle's starting position perpendicular to the line.

[7] Himmelsbach, M., von Hundelshausen, F., Luettel, T., Manz, M., Mueller, A., Schneider, S., and Wuensche, H.-J., "Team MuCAR-3 at C-ELROB 2009," in [*Proceedings of 1st Workshop on Field Robotics, Civilian European Land Robot Trial 2009*], *ISBN 978-951-42-9176-0*, University of Oulu, Oulu, Finland

[8] Schneider, F., "ELROB 2016 event catalogue," 20th-24th June, Eggendorf, Austria (2016).

[9] Fries, C., Burger, P., Kallwies, J., Naujoks, B., Luettel, T., and Wuensche, H.-J., "How MuCAR won the convoy scenario at ELROB 2016," in [*2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*], IEEE DOI: 10.1109/itsc.2017.8317711.

[10] Schneider, F., "ELROB 2010 event catalogue," 17th-20th May, Hammelburg, Germany (2010).

[11] Kania, R., Frederick, P., Pritchett, W., Wood, B., Mentzer, C., and Johnson, E., "Dismounted soldier autonomy tools (dsat)–from conception to deployment," in [*2014 NDIA Ground Vehicle Systems Engineering and Technology Symposium*],

[12] Mazal, J., Stodola, P., and Podhorec, M., "Ugv development with supervised autonomy," in [*Mechatronics - Mechatronika (ME), 2014 16th International Conference on*], 359–363 DOI: 10.1109/MECHATRONIKA.2014.7018284.

[13] Naranjo, J. E., Clavijo, M., Jimenez, F., Gomez, O., Rivera, J. L., and Anguita, M., "Autonomous vehicle for surveillance missions in off-road environment," in [*2016 IEEE Intelligent Vehicles Symposium (IV)*], IEEE DOI: 10.1109/ivs.2016.7535371.

[14] Appelqvist, P., Knuuttila, J., and Ahtiaine, J., "Mechatronics design of an unmanned ground vehicle for military applications," in [*Mechatronic Systems Applications*], InTech DOI: 10.5772/8919.

[15] Amer, N. H., Zamzuri, H., Hudha, K., and Kadir, Z. A., "Modelling and control strategies in path tracking control for autonomous ground vehicles: a review of state of the art and challenges," *Journal of intelligent & robotic systems* **86**(2), 225–254

[16] Martínez, J. L., Mandow, A., Morales, J., Pedraza, S., and García-Cerezo, A., "Approximating kinematics for tracked mobile robots," *The International Journal of Robotics Research* **24**, 867–878 DOI: 10.1177/0278364905058239.

[17] Shin, J., Kwak, D., and Lee, T., "Robust path control for an autonomous ground vehicle in rough terrain," *Control Engineering Practice* **98**, 104384 DOI: 10.1016/j.conengprac.2020.104384.

[18] Pentzer, J., Brennan, S., and Reichard, K., "The use of unicycle robot control strategies for skid-steer robots through the ICR kinematic mapping," in [*2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*], IEEE DOI: 10.1109/iros.2014.6943006.

[19] Pentzer, J., Brennan, S., and Reichard, K., "Model-based prediction of skid-steer robot kinematics using online estimation of track instantaneous centers of rotation," *Journal of Field Robotics* **31**, 455–476 DOI: 10.1002/rob.21509.

[20] Gonzalez, R., Rodriguez, F., Guzman, J., and Berenguel, M., "Localization and control of tracked mobile robots under slip conditions," in [*2009 IEEE International Conference on Mechatronics*], IEEE DOI: 10.1109/icmech.2009.4957141.

[21] Zhao, Z., Liu, H., Chen, H., Hu, J., and Guo, H., "Kinematics-aware model predictive control for autonomous high-speed tracked vehicles under the off-road conditions," *Mechanical Systems and Signal Processing* **123**, 333–350 DOI: 10.1016/j.ymssp.2019.01.005.

[22] Sebastian, B. and Ben-Tzvi, P., "Active disturbance rejection control for handling slip in tracked vehicle locomotion," *Journal of Mechanisms and Robotics* **11** DOI: 10.1115/1.4042347.

[23] Mathiassen, K., Hyndøy, J. I., Østevold, E., Valaker, S., Danielsen, T., Baksaas, M., Olsen, L. E., Thoresen, M., Ruud, E.-L., Selvåg, J., and Sandrib, J., "Base Defence Demonstration at Trident Juncture 2018 - TACT Unmanned Systems for Base and Force Protection," FFI-Report 19/00807, Norwegian Defence Research Establishment (2019).

[24] Larsen, E., Mathiassen, K., Landmark, L., Larsen, M. V., and Øivind Kure, "Autonomous Sensors and Communications Infrastructure," FFI-Report 19/00808, Norwegian Defence Research Establishment (2019).

[25] Mathiassen, K., Baksaas, M., Olsen, L. E., Thoresen, M., and Tveit, B., "Development of an autonomous off-road vehicle for surveillance missions," in [*Proceedings of IST-127/RSM-003 Specialists' Meeting on Intelligence & Autonomy in Robotics*], NATO Science and Technology Organization NATO UNCLASSIFIED.

[26] Thoresen, M., Nielsen, N. H., Mathiassen, K., and Pettersen, K. Y., "Path Planning for UGVs based on Traversability Hybrid A*," *IEEE Robotics and Automation Letters* DOI: 10.1109/lra.2021.3056028.

[27] Haavardsholm, T. V., Smestad, R., Larsen, M. V., Thoresen, M., and Dyrdal, I., "Scene Understanding for Autonomous Steering," in [*Proceedings of IST-127/RSM-003 Specialists' Meeting in Intelligence & Autonomy in Robotics*], NATO Science and Technology Organization NATO UNCLASSIFIED.

[28] Wiig, M. S., Løvlid, R. A., Mathiassen, K., and Krogstad, T. R., "Decision autonomy for unmanned vehicles," in [*Proceedings of IST-127/RSM-003 Specialists' Meeting on Intelligence & Autonomy in Robotics*], NATO Science and Technology Organization NATO UNCLASSIFIED.

[29] Mathiassen, K., Schneider, F. E., Bounker, P., Tiderko, A., Cubber, G. D., Baksaas, M., Główka, J., Kozik, R., Nussbaumer, T., Röning, J., Pellenz, J., and Volk, A., "Demonstrating interoperability between unmanned ground systems and command and control systems," *International Journal Intelligent Defence Support Systems* **6**(2) (In press).

[30] Gade, K., "NavLab, a generic simulation and post-processing tool for navigation," *Modeling, Identification and Control: A Norwegian Research Bulletin* **26**(3), 135–150 DOI: 10.4173/mic.2005.3.2.

[31] Siciliano, B., Sciavicco, L., Villani, L., and Oriolo, G., [*Robotics - Modeling, planning and control*], Springer London DOI: 10.1007/978-1-84628-642-1.

[32] Morin, P. and Samson, C., "Motion control of wheeled mobile robots," in [*Springer Handbook of Robotics*], Siciliano, B. and Khatib, O., eds., ch. 34, 799–826, Springer-Verlag Berlin Heidelberg DOI: 10.1007/978-3-540-30301-5.

[33] Kendra Diezel, Jeff McGough, Chris Smith, Andrew Stelter, Riley Sutton, Samuel Williams., "Veranda." https://github.com/roboscienceorg/veranda.

[34] Niels Hygum Nielsen, "Veranda fork." https://github.com/NielsHygum/veranda.