Evaluation of Two Path Following Controllers for an Ackermann Off-road Vehicle in Winter and Summer Conditions

1st Magnus Baksaas
Defence Systems Division
Norwegian Defence Research Establishment
Kjeller, Norway
magnus.baksaas@ffi.no

2nd Lars Erik Olsen
Defence Systems Division
Norwegian Defence Research Establishment
Kjeller, Norway
lars-erik.olsen@ffi.no

3rd Kim Mathiassen
Defence Systems Division
Norwegian Defence Research Establishment
Kjeller, Norway
kim.mathiassen@ffi.no

***Abstract*—Off-road driving can be a challenging task with rapid changes in the driving conditions, terrain and vehicle behavior. For off-road autonomous vehicles, it is important to be robust to these changes, and parts of this robustness comes from the path following controller of the vehicle. In this paper we compare two different path following controllers using a kinematic model of Ackermann vehicles, the Stanley controller and a controller originally made for unicycles, but adapted for Ackermann vehicles. The comparison is done using an Off-road Polaris vehicle modified for autonomous driving. Two experiments are conducted, where the first experiment is during wintertime, driving with belts in snowy conditions, and the second experiments is during summer time, driving with wheels in muddy conditions. Our research vehicle can be fitted with different payloads that affect the weight and center of gravity. Therefore our main motivation for using kinematic controllers is to have a simple and robust controller that handles all situations using the same control parameters. Both controllers are used on the same path, and the cross-track errors are compared. The main conclusion is that the Stanley controller showed the most robustness against variations in driving conditions and vehicle dynamics, compared to the adapted unicycle method.**

***Index Terms*—Autonomous vehicles, Control equipment, Mechanical variables control**

I. Introduction

Off-road driving can be a challenging task with rapid changes in the driving conditions, terrain and vehicle behavior. The path can be narrow with tight turns and steep hills. The speed varies greatly and is often very low. It takes an experienced off-road driver to be able to read the terrain and know how the vehicle will behave to different driver input. For autonomous vehicles, off-road driving is a different challenge than on-road driving. The vehicle can behave very differently to the same input because of variations in the terrain and traction. An off-road path following controller must be robust to changes in the environment since it is impossible to anticipate what the terrain will be like. Changes can also happen along the route with appearing rain, snow, mud etc.

FFI (The Norwegian Defence Research Establishment) has been involved in Unmanned Ground Vehicle (UGV) research since 2015. The Research program acquired a Polaris Ranger XP 900 EPS vehicle that was converted for autonomous driving [1]. The Polaris is an Ackermann type vehicle with selectable two-wheel drive (2WD) and four-wheel drive (4WD), and automatic gearbox with low and high gear series. The aim for the Polaris vehicle is to be a platform for development of technologies for autonomous off-road driving in both summer and winter conditions, and develop functionality for autonomous mission execution. The vehicle is named "Olav" (Off-road light autonomous vehicle), and is fitted with an Inertial Navigation System (INS), a Light Detection and Ranging sensor (LiDAR), cameras, computers and radios for autonomous operation.

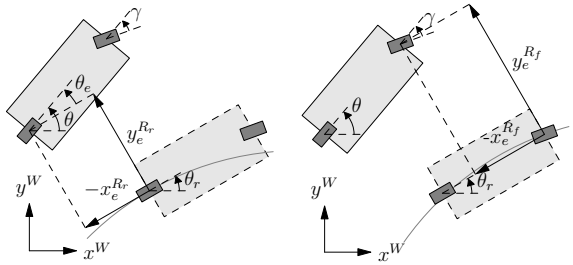
The Norwegian climate has four distinct seasons and large parts of Norway are categorized in the subarctic climate zone with cold winters and lots of snow. Therefore, it is important that our autonomy functionality work in all seasons. Moreover, since Olav can experience many different conditions in one trip and conditions can vary significantly from day to day, we have strived to keep our autonomy software as independent of the current environment conditions as possible. For instance, we do not want to model the wheel to ground friction into our path following controller, since this can vary significantly between different conditions. In addition, Olav can also change between missions. Olav can be fitted with different payloads that affect the vehicle's weight and center of gravity, and use belts instead of wheels (see Figure 1). Considering all this, we have tried to make the control software for Olav as robust and if possible model-free, to make it easier to handle all configurations and different environmental conditions.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

(a) Winter configuration.

(b) Summer configuration.

Fig. 1: Olav in winter and summer configuration.



(a) Rear drive error definitions

(b) Front drive error definitions

Fig. 2: Definitions of error for rear and front wheels. Vehicle with solid lines is the actual vehicle, while vehicle with dotted lines is the reference vehicle.

In [2] a review of state of the art path following strategies is presented. Based on our application, we are mainly focusing on a geometric and/or kinematic controller, as no dynamic or friction model is required using these types of controllers. However, there are several other approaches such as dynamic [3], [4], LQR [5] and MPC [6] controllers, that require such models.

The Stanley controller has previously been used by [7], where it is reported that the Stanley controller outperforms Pure-Pursuit in most scenarios, but that the Stanley controller is not as robust to large errors and non-smooth paths. It also reports that a well-tuned Stanley controller will not cut corners, but will instead overshoot turns. In [8] the Stanley controller is compared with a sliding mode controller using simulations, and it concludes that the Stanley controller gets a smaller tracking error than the sliding mode controller does, and that it has better performance in high speeds. In [9] the difficulty of selecting control parameters is addressed. The control parameters are optimized for different driving speed and heading error using Particle Swarm optimization, and the control parameters were chosen adaptively by interpolating between the optimal parameters using Splines.

Recently, there has been published some papers where the authors combined the Stanley controller and a Pure-Pursuit controller. In [10] a hybrid solution is presented that improves convergence to the path, and that prevents cutting and overshooting in the corners. The method could be improved with an adaptive weight between the Pure-Pursuit controller and the Stanley controller. A weighting method for the controllers are developed and tested in [11]. The result is a more accurate controller than the two controllers are separately.

Kinematic path following controllers include [12], where a path following controller using the path directly to calculate the control outputs based on the positional error an the heading error is presented. This is much like the Stanley controller and the controller presented in this paper, although with a different control equation. In [13] the presented method also takes into account the road margins, and makes the path tracker also consider these constraints. This is done in a two-step manner, where the first step is to find the optimal steering curvature, using the road model as a constraint. The second step is using a controller to track the circular path. It is claimed that this solves the difficulties of tuning look ahead distance, but the cost is a more complex method.

In this paper, we compare two path following controllers to select the one that has best performance for our application. The first controller is based on [14] and is originally made for unicycles, but modified in this paper for Ackermann vehicles. The second controller is the Stanley controller [15], used by the Stanford team to win the DARPA Grand Challenge in 2005 [16]. The same software and tuning parameters for the vehicle were used in the experiments to find out how robust the software is to changing conditions, and to compare how the two different controllers perform in two very different conditions.

II. Model

This section describes two different control methods for path following that have been tested on Olav. The first control method is based on [14], and has a feed forward component. This controller uses the reference vehicle speed and steering angle as a nominal setpoint, and uses the difference between the actual and reference vehicle as errors in the feedback component of the controller. The second controller is based on the Stanley controller used by the Stanford team that won the DARPA Grand Challenge in 2005 [16].

A. Vehicle model

The path following is done by steering towards a reference vehicle that is placed on the trajectory. Throughout this paper, we will be using three frames for defining the position and orientation of the reference and current vehicle positions, and errors used in the controllers. The fixed world frame is denoted with a superscript W and is defined as a Cartesian East, North, Up (ENU) system. Then there are two frames fixed to the reference vehicle, both Cartesian Front, Left, UP (FLU) systems, where the first frame R_f has its origin at the ground between the front wheels and the second frame R_r has its origin at the ground between the rear wheels.

The errors between Olav and the reference vehicle, i.e. lateral error y_e, longitudinal error x_e, heading error θ_e and

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

steering error γ_e, showed in Figure 2. The first control method in this paper is based on a rear wheel model, i.e. all the errors are computed from the reference vehicle's rear axle, i.e. in the R_r frame. The rear wheel model is shown in Figure 2a. The second controller is based on a front wheel controller, i.e. all the errors are computed from reference vehicle's front axle, i.e. in the R_f frame. The front wheel model is shown in Figure 2b.

The reference vehicle's position is moved along the trajectory in front of Olav's position. When it is slippery and the vehicle is understeering, as when it is winter, we want to start turning before the vehicle is in the corner. This can be done in two ways; use feed forward on the steering angle, or pushing the reference vehicle further ahead along the trajectory from Olav.

B. Trajectory representation and generation

1) Representation: The trajectory is represented as a discrete set of positions, heading and nominal control inputs. The kinematic model of the vehicle is used to calculate positions and headings at every point on the trajectory. We use the rear axle as the basis for the kinematic model, and the model is given by

$$\dot{x} = v\cos\theta \tag{1a}$$
$$\dot{y} = v\sin\theta \tag{1b}$$
$$\dot{\theta} = \frac{v}{L}\tan\gamma \tag{1c}$$

where x and y are the vehicle's position in the world frame, θ is the vehicle's orientation in the world frame, v is the vehicle's speed measured in vehicle's body frame and γ is the vehicle's steering angle control input. L is the distance between the vehicle's front and rear axle. Each point in of the trajectory is defined by the vector $\boldsymbol{q}_i = [x_i\ y_i\ \theta_i\ v_i\ \gamma_i\ s_i]^T$, where x_i, y_i and θ_i are the state vector of the kinematic model, v_i and γ_i are the nominal control inputs, and s_i is for how long this trajectory point should be used before using the next trajectory point. We assume that the nominal control input v and γ are constant in the time frame defined by s_i.

Based on the assumption we can solve the differential equation for $\dot{\theta}$, as it has a constant value on the right side. This yields

$$\theta(s) = \theta_i + \left(\frac{v_i}{L}\tan\gamma_i\right)s = \theta_i + k_\theta s \qquad s \in [0, s_i\rangle \tag{2}$$

We can now insert this into the two other differential equations

$$\dot{x} = v_i\cos\left(\theta_i + k_\theta s\right) \tag{3a}$$
$$\dot{y} = v_i\sin\left(\theta_i + k_\theta s\right) \tag{3b}$$

Expanding $\cos$ and $\sin$ in order to separate the constant and time varying parts and integrating the equations yields

$$x(s) = x_i + v_i\left(\frac{c_{\theta_i}}{k_\theta}\sin k_\theta s + \frac{s_{\theta_i}}{k_\theta}\cos k_\theta s - \frac{s_{\theta_i}}{k_\theta}\right) \tag{4a}$$
$$y(s) = y_i + v_i\left(\frac{s_{\theta_i}}{k_\theta}\sin k_\theta s - \frac{c_{\theta_i}}{k_\theta}\cos k_\theta s + \frac{c_{\theta_i}}{k_\theta}\right) \tag{4b}$$

where $s \in [0, s_i\rangle$, $\cos\theta_i$ is denoted c_{θ_i} and $\sin\theta_i$ is denoted s_{θ_i}. When the assumptions are met, the solution above is an exact solution to the continuous kinematic model.

The index i of the trajectory point $\boldsymbol{q}_i$ is increased by one when the s exceeds s_i. Then also s has to be set to zero. The relationship between the current time t and s at index i is

$$s(t, i) = t - \sum_{j=0}^{i-1} s_j \tag{5}$$

2) Generating a trajectory: A trajectory can be generated by using a motion planner algorithm or measured positions, heading and steering angle. In a real world scenario, a motion planner algorithm will be preferred. But in this case, we want a benchmark track and we want to record the trajectory in advance and reuse the exact same trajectory many times. Therefore, to make the trajectory more realistic, a trajectory is generated from the measured trajectory.

The measured trajectory is recorded with an INS. For our application, this typically results in a track with a distance between each position of less than 1 cm. Since the position measurements are influenced by noise, the track is resampled. This results in fewer positions with a larger distance between each position. To generate a trajectory, the positions is converted from global positions, i.e. latitude and longitude, into local positions in a flat earth plane. After the positions are converted, the track is resampled into $I + 1$ samples, with one sample for each meter. The heading in each point of the track is the tangent of the track, and the steering angle is the heading rate, i.e. the angular velocity. Both is computed from the positions. The vehicle's speed is computed as a cosine function of steering angle. When driving straight forward, the speed is at maximum, and when driving in steep corners, the speed is at minimum. The reference vehicle's heading, steering angle and velocity is computed by:

$$\theta_i = \begin{cases} \arctan2(y_{i+1} - y_{i-1}, x_{i+1} - x_{i-1}), & 0 < i < I \\ \arctan2(y_i - y_{i-1}, x_i - x_{i-1}), & i = I \\ \arctan2(y_{i+1} - y_i, x_{i+1} - x_i), & i = 0 \end{cases} \tag{6a}$$

$$\gamma_i = \begin{cases} \arctan2\left(L(\theta_{i+1} - \theta_i), \frac{y_{i+1} - y_i}{x_{i+1} - x_i}\right), & 0 \le i < I \\ 0, & i = I \end{cases} \tag{6b}$$

$$v_i = \begin{cases} \frac{1}{2}\left(\cos\left(\pi\frac{\gamma_i}{\gamma_{max}}\right) + 1\right) \\ \cdot\left(v_{max,straight} + v_{max,corner}\right) \\ +v_{max,corner}, & 0 \le i < I \\ 0, & i = I \end{cases} \tag{6c}$$

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

where x is a position in east direction and y is a position in north direction. L is the axle distance, $v_{max,straight}$ is the maximum speed when driving straight forward and $v_{max,corner}$ is the speed when the steering wheel is at maximum position.

C. Low-level controller

The speed of the vehicle is controlled with the throttle as the control signal in a PI controller. The desired speed is the input/setpoint to the controller; the current speed is the feedback. We also add an element from a feed forward controller taking the pitch and speed of the vehicle into consideration [17]. The feed forward element improves the vehicle ability to maintain its speed in the terrain, removing jerky or oscillating speed variations.

Steering is controlled using a PID controller with feedback from a position/rotation sensor on the steering column and desired steering wheel angle as setpoint. The steering controller gets the setpoint from the path following controller. Figure 3 shows a block diagram of the controllers involved in speed and steering control of the vehicle.

D. Morin controller

Our first controller is based on a controller found in [14] where the controller tracks a reference vehicle with the same kinematics. Since our vehicle does not control the steering rate, but rather the steering angle directly, we use a simplified bicycle model for trajectory control, and modify this to be able to use the linearized unicycle control scheme given in Chapter 34.4.2 in [14].

The kinematic model for our system is given in (1) where v is the commanded velocity and γ is the commanded steering angle, both sent as input to the low-level controller. The state variables $[x\ y\ \theta]^T$ are the position and heading of the vehicle, and L is the distance between the front and back axels. The model is simplified by linearizing $\tan(\gamma)$ around zero, which yields γ. As the steering angle range is small (0.5 rad) this yields only a minimal error. The new kinematic model for the heading is

$$\dot{\theta} = \frac{v}{L}\gamma \tag{7}$$

Using $\tilde{\gamma} = \frac{v}{L}\gamma$ yields the same model as a unicycle. The reference vehicle has the same kinematics, denoted by a subscript r. The error dynamics are derived in [14] using the reference vehicle frame, and are found to be

$$\dot{x}_e = \tilde{\gamma}_r y_e + v\cos(\theta_e) - v_r \tag{8}$$
$$\dot{y}_e = -\tilde{\gamma}_r x_e + v\sin(\theta_e) \tag{9}$$
$$\dot{\theta}_e = \tilde{\gamma} - \tilde{\gamma}_r \tag{10}$$

where subscript e denotes errors. 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. $\theta - \theta_r$). Using the following change of coordinates and control variables

$$z_1 = x_e \tag{11}$$
$$z_2 = y_e \tag{12}$$
$$z_3 = \tan(\theta_e) \tag{13}$$
$$w_1 = v\cos(\theta_e) - v_r \tag{14}$$
$$w_2 = \frac{\tilde{\gamma} - \tilde{\gamma}_r}{\cos^2(\theta_e)} \tag{15}$$

yields the system

$$\dot{z}_1 = \gamma_r z_2 + w_1 \tag{16}$$
$$\dot{z}_2 = -\gamma_r z_1 + v_r z_3 + w_1 z_3 \tag{17}$$
$$\dot{z}_3 = w_2 \tag{18}$$

In [14] the following non-linear controller is proposed and shown to give a globally asymptotically stable system

$$w_1 = -k_1 |v_r| (z_1 + z_2 z_3) \tag{19}$$
$$w_2 = -k_2 v_r z_2 - k_3 |v_r| z_3 \tag{20}$$

Linearizing this system yields a more intuitive controller that is easier to tune. This is done in [14] and yields the controller

$$w_1 = -k_1 |v_r| z_1 \tag{21}$$
$$w_2 = -k_2 v_r z_2 - k_3 |v_r| z_3 \tag{22}$$

This controller locally asymptotically stabilizes the origin of the system. The controller is designed so that tuning the controller with $vr = 1$ and $\gamma_r = 0$ gives good results for all other velocities.

In the unicycle case the control variables can be approximates to $w_1 \approx v - v_r$ and $w_2 \approx \tilde{\gamma} - \tilde{\gamma}_r$ near the origin. In our bicycle case the relationship is not so straight forward. Expanding w_2 yields

$$w_2 = \frac{v\gamma - v_r\gamma_r}{L\cos^2(\theta_e)} \tag{23}$$

Around the origin $\cos^2(\theta_e)$ will be one. In our application it is not important for the vehicle to keep up with the reference vehicle, we rather set the reference vehicle on the trajectory so that $x_e \approx 0$. When doing so we can set $k_1 = 0$ and this implies that $v = v_r$. Inserting this into the above equation yields

$$w_2 = \frac{v_r}{L}(\gamma - \gamma_r) = k_\gamma(\gamma - \gamma_r) \tag{24}$$

We now have almost the same relation as in the unicycle case. Inserting the controller w_2 into the above equation and rearranging to find γ yields

$$\gamma = \gamma_r - \frac{k_2}{k_\gamma} v_r y_e - \frac{k_3}{k_\gamma} |v_r| \tan(\theta_e) \tag{25}$$
$$\gamma = \gamma_r - k_y v_r y_e - k_\theta |v_r| \tan(\theta_e) \tag{26}$$

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

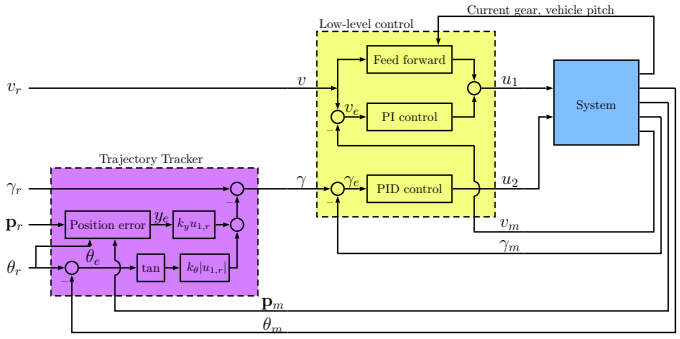


Fig. 3: Complete block diagram for the Morin et. al. controller

Embedding k_γ into the control gains yields the same equation as in the unicycle case. In this controller, γ is non-clamped steering angle setpoint, γ_r is the feed forward steering component computed by the trajectory generator. y_e is the cross-track error and θ_e is the heading error. k_y and k_θ are gains, where k_y is the cross-track gain and k_θ is the heading error gain. To prevent that the vehicle is driving faster than allowed or turn the steering wheels more than it can, the steering angle and speed setpoints are clamped by:

$$u_v = \begin{cases} v, & 0 < v < u_{v,max} \\ u_{v,max}, & v \geq u_{v,max} \\ 0, & v \leq 0 \end{cases} \tag{27a}$$

$$u_\gamma = \begin{cases} \gamma, & |\gamma| < u_{\gamma,max} \\ u_{\gamma,max}, & \gamma \geq u_{\gamma,max} \\ -u_{\gamma,max}, & \gamma \leq u_{\gamma,max} \end{cases} \tag{27b}$$

The complete block diagram for this controller, including the low-level controllers is shown in Figure 3.

E. Stanley controller

The Stanley controller is based on [15]. In this case we use the same velocity controller as for the modified Morin controller, i.e. use the reference vehicle's velocity directly as the control output ($v = v_r$ where v_r is computed by (6c)). The steering controller is given by:

$$\gamma = (\theta_e - \theta_{ss}) + \arctan\left(\frac{k_y y_e}{v_m + k_{v0}}\right) + k_\omega (\omega_m - \omega_r) + k_\gamma (\gamma_m(i-1) - \gamma_m(i)) \tag{28}$$

The Stanley controller is a front wheel controller. This results is that the cross-track error is computed between the vehicle's front axle and the reference vehicle's front axle, as shown in Figure 2b. The vehicle's heading and reference vehicle's heading is the same for both a rear wheel model and a front wheel error.

The controller consist of mainly two terms, one that ensures the correct heading in corners, i.e. zero heading error θ_e, and one that prevents cross-track error, i.e. zero cross-track error y_e. To ensure a smooth and stable transition from a large cross-track error, the cross-track error is placed inside an $\arctan$ function. The result is that the steering angle from a large cross-track error newer will exceed $\pm\pi\ rad$. The cross-track error is scaled with the vehicle's measured speed v_m and a gain k_y. To prevent numerical instability when the speed is low, a constant k_{v0} is added to the vehicle's speed.

In addition to compensating for heading error, i.e. θ_e, the controller also compensating for extra heading caused by oversteering in a corner, θ_{ss}. The steady state heading is computed by using the sums of forces and moments on the tires.

When the speed increases, the damping effect from the tires in corners diminishes. This causes the vehicle to oversteer. To prevent this, the term $k_\omega (\omega_r - \omega_m)$ is added, where ω_r is the reference vehicle's yaw rate, ω_m is the measured yaw rate and k_ω is a gain. The final term compensates for time delay in the steering servo. The term $k_\gamma (\gamma_m(i) - \gamma_m(i-1))$, where $\gamma_m(i)$ is the measured steering angle at discrete time i, $\gamma_m(i-1)$ is the previous measured steering angle at discrete time $i-1$ and k_γ is a gain.

We have done some small modifications to the controller in (28). Our maximum velocity is 6 m/s. Most of the terms in (28) are to mitigate high speed dynamics. Therefore, θ_{ss} and k_ω are set to zero. To have the possibility to lower the impact of heading error, we have added a gain, k_θ, to the heading error, θ_e. Our steering controller becomes

$$\gamma = k_\theta \theta_e + \arctan\left(\frac{k_y y_e}{v_m + k_{v0}}\right) + k_\gamma (\gamma_m(i-1) - \gamma_m(i)). \tag{29}$$

where the steering setpoint is clamped by (27).

When it is slippery conditions, the vehicle will understeer because it starts to turn the steering wheel too late. The heading error and cross-track error will become large. If the corner is steep, the steering angle will usually be maximum and the vehicle will not be able to manage the corner. The solution to this is to start turning before the vehicle is in the corner. With the Morin, this is done by having the reference vehicle's steering angle as a feed forward. The Stanley controller does not use steering angle at all. One approach is therefor to push the reference vehicle further in front of the vehicle. When the vehicle is driving fast, the vehicle has to start turning earlier than when the vehicle is driving slowly. This can be done by pushing the reference vehicle forward by

$$x_e = k_t v_r, \tag{30}$$

where k_t is the time delay from when we want to turn the vehicle when it actually starts to turn.

III. Experiments

To compare the two controllers, a trajectory is created by driving the track manually first. The recording was done during winter conditions and Olav was equipped with belts, see Figure 4a. The trajectory is shown in Figure 5 and is

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

approximately 700 m long. The trajectory was recorded with an INS using a Honeywell HG9900 Inertial Measurement Unit (IMU) and a Trimble SPS855 GNSS receiver. The INS system NavLab [18] gives a trajectory with a position accuracy of less than 0.3 m and a heading accuracy of less than 0.01° using the descibed sensors. The vehicle's velocity and steering angle is also recorded. The trajectory was recorded in wintertime. Since this trajectory is based on real measurements, we have the opportunity to use the modified Morin controller with real steering and velocity as feed forward. But in a real autonomous system, the trajectory would be generated with a motion planner algorithm. The trajectory is generated by resampling the positions, and computing the heading, steering angle and velocity based on (6).

To compare the two methods, we have done three experiments in winter conditions and repeated the same three experiments in summer conditions (with the same tuning parameters):

1) Modified Morin with measured trajectory as input
2) Modified Morin with resampled trajectory and generated heading, steering angle and velocity
3) Stanley with resampled trajectory and generated heading and velocity

In the first experiment, we use the modified Morin controller with real world measured positions and feed forward values. This gives the controller an unfair advantage, since in real world applications; the trajectory and feed forward values would be generated. Therefore, in the second experiment, we use the same controller with resampled trajectory and generated values for steering for the feed forward control. The third experiment uses the Stanley controller with resampled trajectory and generated values for heading and velocity. The three controllers were tested 5 times each in both winter and summer conditions.

During the winter experiment, the path along the trajectory was covered in about 25 cm of fresh snow with some tracks from other military vehicles, see Figure 4a. Every run with Olav created a track in the snow and the vehicle was sometimes caught in tracks from other vehicles or previous runs. In winter conditions, we ran Olav with belts in 4WD with differential locks on both front and rear differential. This causes a lot of understeering since all the belts are forced to have the same speed. In summer conditions, the path was quite muddy after a lot of rain, see Figure 4b. We ran Olav in 2WD without differential lock to reduce understeer.

The Morin controller had the following parameters in both experiments: $k_x = 0$, $k_y = 0.035$ and $k_\theta = 0.2$. By setting $k_x = 0$, the vehicle will only drive as fast as the reference vehicle.

The Stanley controller had the following parameters: $k_\theta = 0.5$, $k_y = 1.0$, $k_{v0} = 2.0$, $k_\gamma = 1.0$ and $k_t = 0.4$ s. By setting k_t to 0.4 s, the reference vehicle will be 2 m ahead of Olav when driving at maximum speed at 5 m/s, and 0.8 m ahead when of Olav when driving in corners at 2 m/s.

(a) Winter conditions. (b) Summer conditions.

Fig. 4: The track in winter and summer conditions.

A. About the vehicle

The experiment vehicle, Olav, is a Polaris Ranger 900 EPS with automatic transmission, electronic fuel injection and electric power steering that has been modified for autonomous driving. The vehicle has been fitted with an industrial Programmable Logic Controller (PLC) from WAGO to read sensor data and generate control signals. The PLC is connected to a computer that runs the low-level controllers and the path following controllers. During the experiments, the computer was running Linux 16.04 and the software was using ROS Kinetic[1]. In autonomous mode, the speed is controlled by sending a generated throttle signal to the electronic fuel injection system. The steering angle is controlled by creating a fake torque sensor signal to the electric power steering. Gear change is done by a linear actuator moving the gear lever, and brakes are applied using a linear actuator pulling the brake pedal using a cable. All actuators are controlled through the PLC. The vehicle can either be operated as a normal vehicle by a driver, or by flicking a switch, be controlled by the controller computer through the PLC and actuators.

The low-level controllers for velocity and steering are implemented using the ROS control toolbox's[2] proportional–integral–derivative (PID) controller. Positive values from the PID controller result in a throttle effort and negative values a brake effort. The throttle effort is zero when braking, and the brake effort is zero when throttle is applied. The parameters for the PID velocity controller is $P = 35$, $I = 4$ and $D = 0$. In addition, we have a clamping in the integration term when it reaches 40. Throttle effort goes from 0 to 100 while the brake effort is from 0 to 80. The throttle effort is converted to a voltage sent to the ECU by a digital-to-analog converter (DAC) in the PLC and the brake effort is converted to a pulling force for the linear actuator. The vehicle's steering angle is controlled by generating a fake torque signal to the electrical power steering. The steering angle is measured with a potentiometer on the steering column. The value from the potentiometer is converted in the PLC by a 16-bit analog-to-digital converter (ADC). The values from the PLC for max left, center and max right steering angles were decided empirically. The parameters for the PID steering controller are $P = 156$, $I = 520$ and $D = 11.7$. In addition, we have a clamping in

[1]https://wiki.ros.org/kinetic
[2]https://wiki.ros.org/control_toolbox

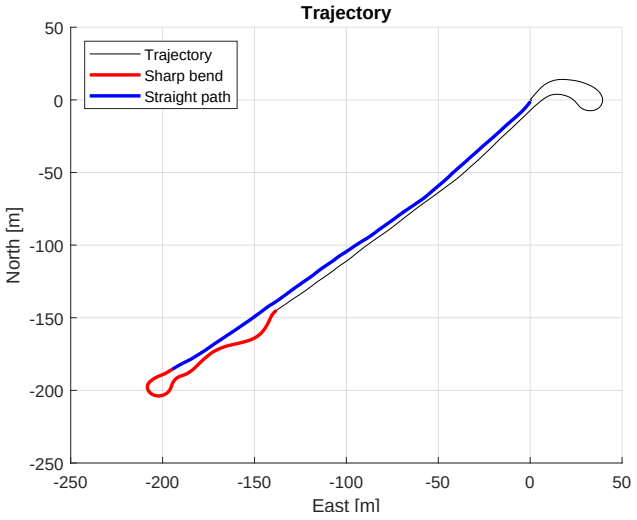


Fig. 5: The trajectory we want to follow. The trajectory is divided into two interesting parts, a sharp bend in red and a straight path in blue.

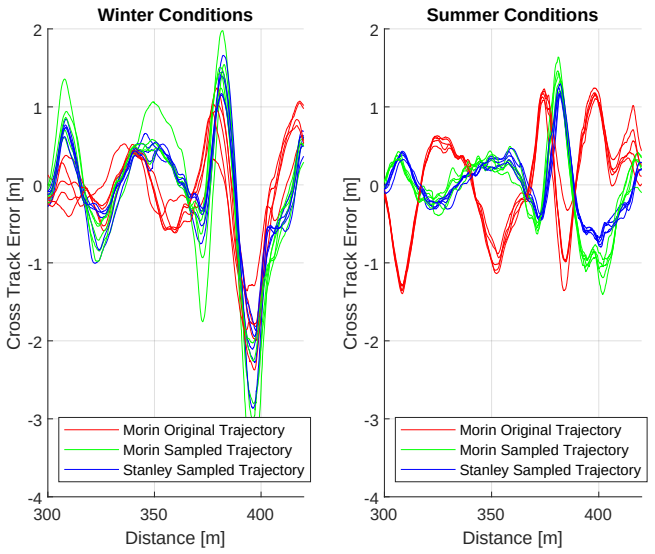


Fig. 6: Cross-track error at the sharp bend in both winter (to the left) and summer (to the right) conditions. The x-axis shows the distance along the trajectory.

the integration term when it reaches 1000. The value from the steering PID goes from -3000 to 3000 and is converted to the fake torque signal to the electronic power steering by a DAC in the PLC. Both controllers run at $50\ Hz$.

IV. Results

The path following can be divided into two parts, the first is how well the vehicle follows a trajectory in sharp bends, and the second is how well the vehicle follows a straight path. In the following results, we are only concentrating on those parts. The sharp bend is marked in red and a straight path is marked in blue in Figure 5. Each controller has been tested five rounds of the trajectory in both winter and summer conditions. We need to compute the actual cross-track error from the same position on the vehicle so the two controllers can be compared. In the following comparison, all cross-track errors are computed from the center of the rear axle and to the nearest position on the trajectory as in Figure 2a. The cross-track error is computed with use of n-vectors[3] [19] since the INS gives us a global position in latitude and longitude and the trajectory is measured in latitude and longitude.

A. Trajectory tracking in a sharp bend

Figure 6 shows the cross-track error at the sharp bend. Each of the three controllers have been tested five times in both winter conditions (to the left) and summer conditions (to the right). As the Figure shows, the modified Morin controller with real measurements as feed forward has the smallest cross-track error in the wintertime. However, the same controller has much larger errors in summer conditions. This is because the controller uses feed forward from real measurements and the trajectory was recorded in winter conditions where the vehicle understeered in every corner. Another observation is that the modified Morin controller and the Stanley controller with resampled trajectory have almost the same cross-track error.

Figure 7 shows a box plot of the five attempts concatenated into one time series for each controller. The blue boxes indicate the 25th and 75th percentiles, and the red central mark indicates the median. The whisker indicate the minimum and maximum cross-track error. The Figure shows that modified Morin with original trajectory has the smallest error in winter conditions, but it has the largest errors in summer time. The modified Morin with resampled trajectory and Stanley have almost the same performance.

To compare the controllers, the mean of root-mean-square-error (RMSD) and mean of standard deviation (STD) over all attempt is computed. The mean of RMSD tells the mean cross-track error for each controller. The mean of STD tells the deviations of the attempts, and therefore the repeatability of the controllers. The mean RMSD is computed by

$$mean(RMSD) = \frac{1}{I}\sum_{i=0}^{I} RMSD(i) \tag{31}$$

where $i \in [0, N-1]$ is a position at the trajectory. The RMSD at a single position at the trajectory is computed by

$$RMSD(i) = \sqrt{\frac{1}{K}\sum_{k=1}^{K} y_e^2(k,i)} \tag{32}$$

where there are k is attempts ($K = 5$) and $y_e(k,i)$ is the cross-track error at position i at attempt k. The mean STD is computed by

[3]https://www.navlab.net/nvector/#example_10

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

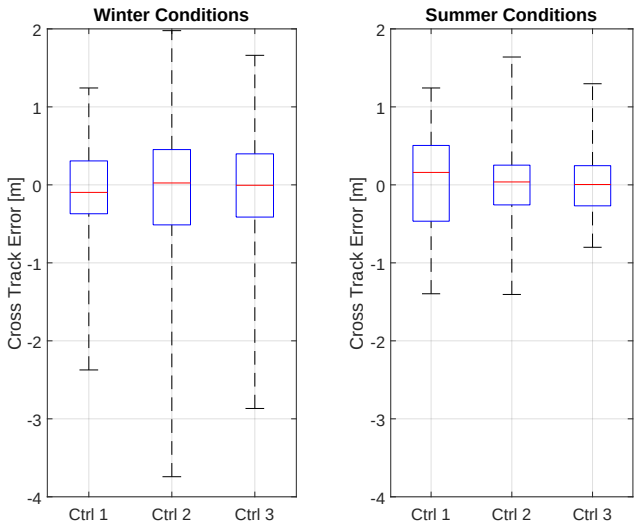


Fig. 7: Box plot of the cross-track error at the sharp bend in both winter (to the left) and summer (to the right) conditions. Ctrl 1 is the modified Morin controller with original trajectory. Ctrl 2 is the modified Morin controller with resampled and generated trajectory. Ctrl 3 is the Stanley controller with resampled trajectory.

$$mean(STD) = \frac{1}{I}\sum_{i=0}^{I} STD(i) \tag{33}$$

where $i \in [0, N-1]$ is a position at the trajectory. The STD at a single position at the trajectory is computed by

$$STD(i) = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(y_e(k,i) - \bar{y}_e(i)\right)^2} \tag{34}$$

where $y_e(k,i)$ is the cross-track error at position i and attempt k and $\bar{y}_e(i)$ is the mean of cross-track error to attempts in the same position.

Table I shows mean RMSD and mean STD for the three controllers at the sharp bend in both winter and summer conditions. As the table shows, the Morin controller with original trajectory has the lowest mean RMSD and mean STD during winter conditions. This is not a surprise, since the controller has a advantage of using the actual measurements as feed forward in the controller. On the other side, the same controller has the largest mean RMSD during summer time.

B. Trajectory tracking at a straight path

Figure 8 shows the cross-track error at the straight path. Each of the three controllers have been tested five times in both winter conditions and summer conditions.

Figure 9 shows a box plot of the five attempts concatenated into one time series for each controller. The blue boxes indicate the 25th and 75th percentiles, and the red central mark indicates the median. The whisker indicate the minimum and maximum cross-track error.

TABLE I: Summary of the sharp bend in both winter and summer conditions. Both Morin controllers are the modified ones. The columns mRMSD show the mean RMSD from (31) and the columns mSTD show the mean RMSD from (33)

	Winter		**Summer**	
Controller	**mRMSD**	**mSTD**	**mRMSD**	**mSTD**
Morin Original	0.171 m	0.012 m	0.183 m	0.011 m
Morin Resampled	0.205 m	0.017 m	0.170 m	0.043 m
Stanley Resampled	0.180 m	0.014 m	0.109 m	0.007 m

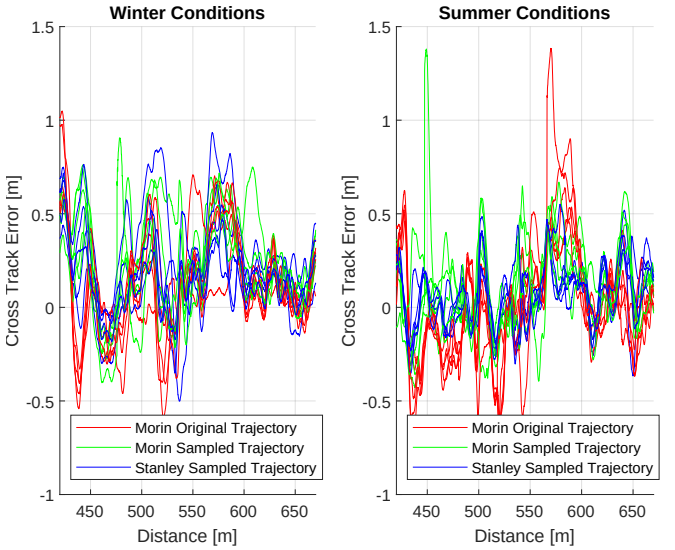


Fig. 8: Cross-track error at the straight path in both winter (to the left) and summer (to the right) conditions. The x-axis shows the distance along the trajectory.

Table II shows mean RMSD and mean STD for the three controllers at the straight path in both winter and summer conditions.

C. Whole track

Figure 10 shows the cross-track error at the whole trajectory. Each of the three controllers have been tested five times in both winter conditions and summer conditions.

Table III shows mean RMSD and mean STD for the three controllers at the whole trajectory in both winter and summer conditions.

V. Discussion

From the results we can see the modified Morin controller preforms really well on the recorded trajectory with "real" recorded values for heading, steering angles etc. when it is used in similar conditions as when recorded. When the same

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

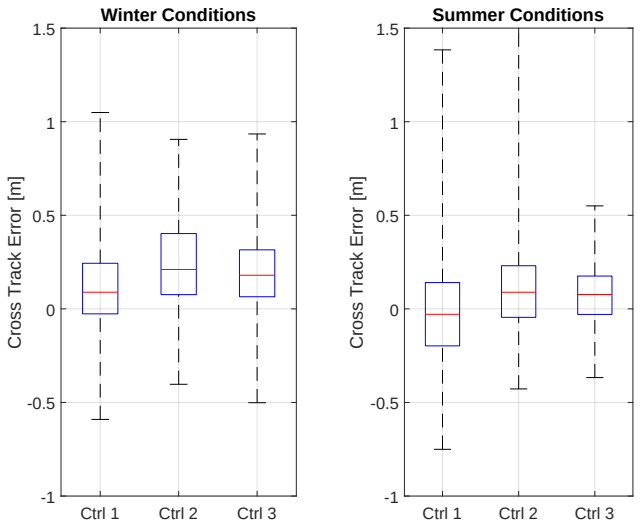


Fig. 9: Box plot of the cross-track error at the straight path in both winter (to the left) and summer (to the right) conditions. Ctrl 1 is the modified Morin controller with original trajectory. Ctrl 2 is the modified Morin controller with resampled and generated trajectory. Ctrl 3 is the Stanley controller with resampled trajectory.

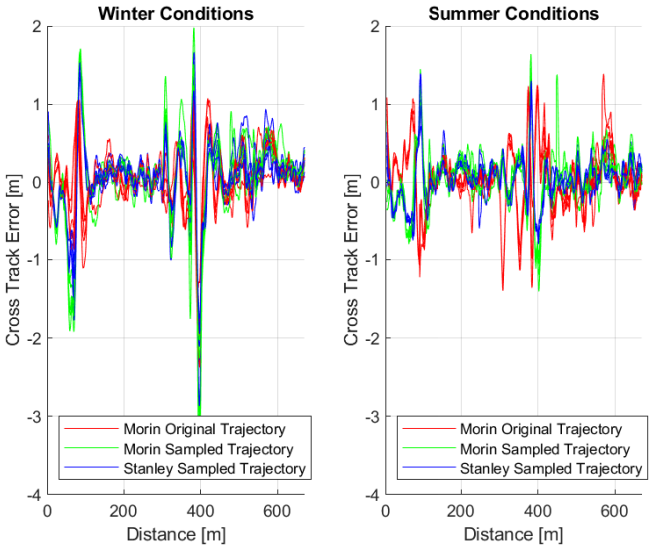


Fig. 10: Cross-track error at the whole trajectory in both winter (to the left) and summer (to the right) conditions. The x-axis shows the distance along the trajectory.

TABLE II: Summary of the straight path in both winter and summer conditions. Both Morin controllers are the modified ones. The columns mRMSD show the mean RMSD from (31) and the columns mSTD show the mean RMSD from (33)

	Winter		**Summer**	
Controller	**mRMSD**	**mSTD**	**mRMSD**	**mSTD**
Morin Original	0.377 m	0.020 m	0.390 m	0.008 m
Morin Resampled	0.567 m	0.018 m	0.351 m	0.025 m
Stanley Resampled	0.453 m	0.011 m	0.274 m	0.007 m

TABLE III: Summary of the whole trajectory both winter and summer conditions. Both Morin controllers are the modified ones. The columns mRMSD show the mean RMSD from (31) and the columns mSTD show the mean RMSD from (33)

	Winter		**Summer**	
Controller	**mRMSD**	**mStd**	**mRMSD**	**mStd**
Morin Original	0.273 m	0.041 m	0.273 m	0.011 m
Morin Resampled	0.389 m	0.051 m	0.215 m	0.008 m
Stanley Resampled	0.307 m	0.019 m	0.163 m	0.004 m

recorded trajectory is played back in different conditions, the controller does not perform that well on recorded data.

In a real world scenario on an autonomous mission, it is unlikely that the vehicle would have access to recorded trajectory data from similar track conditions. Instead the trajectories would be generated from some kind of motion planning software taking sensor data, terrain data, map data etc as inputs. The "true" steering angles that the modified Morin controller need to track the trajectory well, will wary with the driving conditions and the vehicle dynamics. With a vehicle that changes its dynamics with the seasons (belts/wheels) and mission (heavy load/ no load) and driving conditions changing within the seasons and terrain, this becomes an impossible challenge for the motion planner, and the modified Morin controller performance will be very variable and unpredictable for generated/resampled trajectories. From the results in Chapter IV we can see that the modified Morin controller is outperformed by the Stanley controller that we will discuss next.

The Stanley controller uses another approach than the modified Morin controller. It does not use steering angle information from the trajectory. This results in an easier trajectory to generate and less data in the trajectory. The results in Chapter IV shows that the Stanley controller performs better that the modified Morin controller on the generated/resampled trajectory in both winter and summer conditions. For off-road driving where the terrain and driving conditions are guaranteed to change a lot, and with a vehicle that will change its dynamic properties between missions, it is of great benefit to use a path following controller that is robust to changes in these properties. The results show that the Stanley controller performs better that modified Morin controller with the same trajectory data under very different conditions and

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830

vehicle dynamics, showing that it is more robust to changing conditions and are more suited for off-road driving.

VI. Conclusion

The main goal was to find a robust path following controller that performs well in in the challenging and varying conditions. The controller has to be robust, not only to changes in the driving conditions, but also to changes in the vehicle dynamics. The modified Morin controller performs well with recorded trajectory and steering inputs when used in similar conditions as when recorded. The controller does not perform as well with generated trajectory and steering inputs. To generate the trajectory and steering inputs to make the modified Morin controller perform well, will require a high degree of knowledge about driving conditions and vehicle dynamics which is very difficult in off-road driving and with varying vehicle dynamics. The Stanley controller needs less information in the trajectory data and is shown to be more robust against variations in driving conditions and vehicle dynamics. It performs better than modified Morin controller on generated trajectories in both winter and summer conditions.

We have found the Stanley controller to be robust and it works well for our vehicle, Olav, in both winter and summer conditions with different vehicle configurations and challenging driving conditions.

References

[1] Kim Mathiassen, Magnus Baksaas, Lars Erik Olsen, Marius Thoresen, and Bjørn Tveit. Development of an Autonomous Off-Road Vehicle for Surveillance Missions. In *Proceedings of IST-127/RSM-003 Specialists' Meeting on Intelligence & Autonomy in Robotics*, Bonn, Germany, Oct. 2016. NATO Science and Technology Organization. NATO UNCLASSIFIED. doi:10.14339/STO-MP-IST-127.

[2] Noor Hafizah Amer, Hairi Zamzuri, Khisbullah Hudha, and Zulkifli Abdul Kadir. 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, 2017. doi:10.1007/s10846-016-0442-0.

[3] R Lenain, E Lucet, C Grand, B Thuilot, and Faiz Ben Amar. Accurate and stable mobile robot path tracking: An integrated solution for off-road and high speed context. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2010. doi:10.1109/iros.2010.5650213.

[4] Mathieu Deremetz, Roland Lenain, Benoit Thuilot, and Vincent Rousseau. Adaptive trajectory control of off-road mobile robots: A multi-model observer approach. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2017. doi:10.1109/icra.2017.7989509.

[5] R. A. Cordeiro, J. R. Azinheira, E. C. de Paiva, and S. S. Bueno. Dynamic modeling and bio-inspired lqr approach for off-road robotic vehicle path tracking. In *2013 16th International Conference on Advanced Robotics (ICAR)*, pages 1–6, 2013. doi:10.1109/ICAR.2013.6766549.

[6] R. Yu, H. Guo, Z. Sun, and H. Chen. Mpc-based regional path tracking controller design for autonomous ground vehicles. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2510–2515, 2015. doi:10.1109/SMC.2015.439.

[7] Jarrod M. Snider. Automatic steering methods for autonomous automobile path tracking. Technical Report CMU-RI-TR-09-08, Carnegie Mellon University, Pittsburgh, PA, February 2009.

[8] Tomás Carricajo Martin, Marcos E. Orchard, and Paul Vallejos Sánchez. Design and simulation of control strategies for trajectory tracking in an autonomous ground vehicle. *IFAC Proceedings Volumes*, 46(24):118–123, sep 2013. doi:10.3182/20130911-3-br-3021.00096.

[9] N. H. Amer, K. Hudha, H. Zamzuri, V. R. Aparow, A. Faiz Zainal Abidin, Z. A. Kadir, and M. Murrad. Adaptive trajectory tracking controller for an armoured vehicle: Hardware-in-the-ioop simulation. In *2018 57th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, pages 462–467, 2018. doi:10.23919/SICE.2018.8492544.

[10] M. Cibooglu, U. Karapinar, and M. T. Söylemez. Hybrid controller approach for an autonomous ground vehicle path tracking problem. In *2017 25th Mediterranean Conference on Control and Automation (MED)*, pages 583–588, 2017. doi:10.1109/MED.2017.7984180.

[11] Á. Domina and V. Tihanyi. Path following controller for autonomous vehicles. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pages 1–5, 2019. doi:10.1109/ICCVE45908.2019.8964960.

[12] Chang Boon Low. Design, implementation, and experimental validation of a cascaded trajectory tracking controller for nonholonomic car-like wheeled mobile robots with velocity and steering controllers in the loops. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*. IEEE, aug 2017. doi:10.1109/ccta.2017.8062663.

[13] Zhenping Sun, Qingyang Chen, Yiming Nie, Daxue Liu, and Hangen He. Ribbon model based path tracking method for autonomous land vehicle. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2012. doi:10.1109/iros.2012.6385615.

[14] Pascal Morin and Claude Samson. Motion control of wheeled mobile robots. In Bruno Siciliano and Oussama Khatib, editors, *Springer Handbook of Robotics*, chapter 34, pages 799–826. Springer-Verlag Berlin Heidelberg, 2008. doi:10.1007/978-3-540-30301-5.

[15] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun. Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing. In *2007 American Control Conference*, pages 2296–2301, July 2007. doi:10.1109/ACC.2007.4282788.

[16] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006. doi:10.1002/rob.20147.

[17] Per Magnus Auby. Modeling and parameter estimation of an unmanned ground vehicle with a continuously variable transmission. Master's thesis, University of Oslo, 2016. http://hdl.handle.net/10852/51847.

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

[19] Kenneth Gade. A non-singular horizontal position representation. *Journal of navigation*, 63(3):395–417, 2010. doi:10.1017/S0373463309990415.

Dette er en postprint-versjon/This is a postprint version.
DOI til publisert versjon/DOI to published version: 10.1109/ICMRE51691.2021.9384830