KTH Research Concept Vehicle Autonomous Path Following

Pontus Belvén, Johan Bjurgert, Anders Gidmark and Erik Hallqvist

***Abstract*—In this project, a framework for autonomous path following is developed and evaluated both in real-time simulations and in test drives with the KTH Research Concept Vehicle (RCV). A Model Predictive Control (MPC) solution is used to compute control signals to the RCV, using online linearization. The MPC also allows for taking comfort criteria into account, which is used in this project. The final controller was not able to be tested on the real RCV due to time constraints, instead we present results when we use a hardware-in-the-loop (HIL) simulation.**

I. Introduction

In the automotive industry, more and more emphasis is put on autonomous systems which have become feasible thanks to more efficient hardware. The benefits are many, including safety, increased comfort and possibly less ecological footprint. Autonomous driving requires path generation and a suitable control to follow the generated path.

This project is a part of the Automatic Control Project Course (EL2421) which also cooperates with the Vehicle Dynamics Project Course Part 1 and 2 (SD2229/SD2230), both at the Royal Institute of Technology. The goal of the project is to make the KTH Research Concept Vehicle (RCV) able to drive and follow a path autonomously. The group in the Vehicle Dynamics Project Course have the task to make a reliable model and state estimation of the RCV. Our task is to use this model and state estimation to make a controller which successfully follows a path, defined as checkpoints.

This report is structured as follows. Section II presents related work in the area of autonomous path following. Section III explains the dynamical model of the car and the methods used to solve the autonomous path following problem. The implementation is presented in section IV. In section V the results are given and finally the report is concluded in Section VI.

II. Related Work

In the literature, practitioners seem to focus on either an advanced controller while keeping the path generation simple [1], [2] or using a simple controller while using optimized paths [3], [4]. In [5], the authors construct an MPC using a simple bicycle model. They use a nonlinear model and linearize it during each time step such that the linearized MPC is a Linear Time-Varying (LTV) system, using only the front steering angle as an input.

An example of the other approach is [6], where the idea is to construct optimal paths and use a decentralized PID controller for steering and moment.

	Input	**Output**
Vehicle	Torque	Movement
Sensors	Movements	Measurements
State Estimation	Measurements	Estimation of state
Path Generator	Checkpoints	Path
Path Planner	Path and current state	References
Controller	References and current state	Control signal
Actuator	Control signal	Torque

Table I
Inputs and outputs

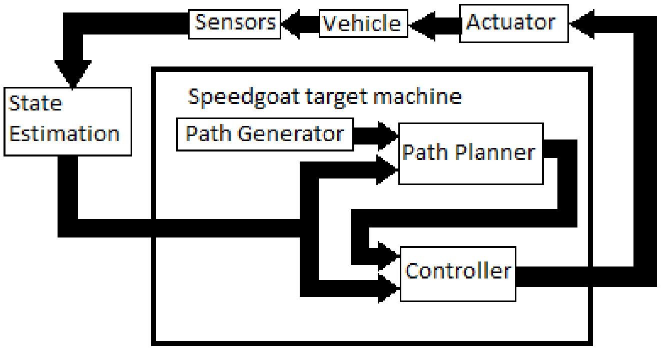

Figure 1. The overall design of the controller.

The benefit of using a more advanced controller is that it allows for comfort criteria to constrain control actions, which is desired in this project.

III. System Design

The overall design for the controller can be seen in Figure 1. The blocks' inputs and outputs are explained in Table I.

A. Vehicle Model

A so called bicycle model [7] is used to model the RCV. It is a simple model to give the essential information about the dynamics of a vehicle, and consists of a wheel on the front axis and one wheel on the rear axis. The states modeled by the bicycle model are longitudinal speed v_x, latitudinal speed v_y, yaw rate $\dot{\psi}$, yaw angle ψ and position in the global coordinates (X, Y). The inputs to the bicycle model are torque on the front and rear wheel M_{12} and M_{34}, respectively, and the steering angle of the front wheel δ_{12} and rear wheel δ_{34}. The purpose of the vehicle model is to use it both as the system model in

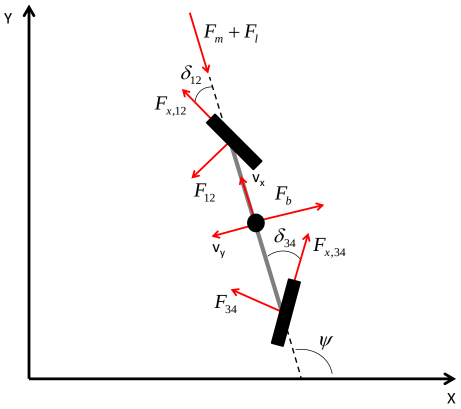

Figure 2. Bicycle model.

the MPC, and as the simulation model when doing the HIL simulations. The bicycle model is described by the following equations of motion

$$\begin{aligned}\uparrow (m+m_j)(\dot{v}_x - \dot{\psi} v_y) &= F_{34}\sin(\delta_{34}) - F_{12}\sin(\delta_{12}) \\ &+ F_{x,12}\cos(\delta_{12}) + F_{x,34}\cos(\delta_{34}) \\ &- F_m - F_l v_x^2, \qquad (1a) \\ \rightarrow (m+m_j)(\dot{v}_y + \dot{\psi} v_x) &= F_{34}\cos(\delta_{34}) + F_{12}\cos(\delta_{12}) \\ &+ F_{x,12}\sin(\delta_{34}) - F_{x,34}\sin(\delta_{34}) \\ &- F_b, \qquad (1b) \\ J_z\ddot{\psi} &= f(F_{12}\cos(\delta_{12}) + F_{x,12}\sin(\delta_{12})) \\ &- b(F_{34}\cos(\delta_{34}) - F_{x,34}\sin(\delta_{34})), \\ & \qquad (1c)\end{aligned}$$

where m is the mass of the vehicle, including two average passengers, m_j is the equivalent mass of the rotating parts, and J_z is the inertia around the z-axis through the center of gravity CG. Seen in Figure 2 are v_x and v_y, which are the speeds in the x and y directions respectively, ψ is the yaw angle, and f and b are the distances from CG to the front and rear axles respectively. δ_{12} and δ_{34} are the steering angles for the front and rear wheels respectively. The tire forces F_{12} for the front tire, and F_{34} for the rear tire are given by

$$\begin{aligned}F_{12} &= -C_{12}\alpha_{12}, \qquad (2a) \\ F_{34} &= -C_{34}\alpha_{34}, \qquad (2b)\end{aligned}$$

where C_{12} and C_{34} are the cornering stiffness of the front and rear axle respectively. α_{12} and α_{34} are the slip angles of the front and rear tires respectively and are given by:

$$\alpha_{12} = \tan^{-1}\left(\frac{v_y + \dot{\psi} f}{v_x}\right) - \delta_{12}, \qquad (3a)$$

$$\alpha_{34} = \tan^{-1}\left(\frac{v_y - \dot{\psi} b}{v_x}\right) - \delta_{34}. \qquad (3b)$$

F_m, F_b, are given by

$$\begin{aligned}F_m &= mg\sin(\theta) + fr, \qquad (4a) \\ F_b &= mg\sin(\phi), \qquad (4b)\end{aligned}$$

where f_r is the tire rolling resistance, θ is the road slope, ϕ is the road bank, and g is the gravitational constant. F_l is given by

$$F_l = \frac{1}{2}\rho c_{air} A, \qquad (5)$$

where ρ is the air density, c_{air} is the aerodynamic drag coefficient, and A is the projected frontal area [8], [9].

The equations of motion are used to calculate the state space representation of the bicycle model and $\dot{X}$, $\dot{Y}$, and $\dot{\psi}$ are added from [9]. The resulting state space model is

$$\begin{aligned}\dot{v}_x &= \frac{1}{m+m_j}(F_{34}\sin(\delta_{34}) - F_{12}\sin(\delta_{12}) + F_{x,34}\cos(\delta_{34}) \\ &+ F_{x,12}\cos(\delta_{12}) - F_m - F_l v_x^2) + \dot{\psi} v_y, \qquad (6a) \\ \dot{v}_y &= \frac{1}{m+m_j}(F_{34}\cos\delta_{34} + F_{12}\cos\delta_{12} - F_{x,34}\sin(\delta_{34}) \\ &+ F_{x,12}\sin(\delta_{12}) - F_b) - \dot{\psi} v_x, \qquad (6b) \\ \ddot{\psi} &= \frac{1}{J_z}(fF_{12}\cos(\delta_{12}) - bF_{34}\cos(\delta_{34}) \\ &+ F_{x,12}\sin(\delta_{12}) + b(F_{x,34}\sin(\delta_{34})), \qquad (6c) \\ \dot{\psi} &= \frac{\sqrt{v_x^2 + v_y^2}(\tan(\delta_{12}) - \tan(\delta_{34}))}{(f+b)} \\ &\times \cos\left[\tan^{-1}\left(\frac{f\tan(\delta_{34}) + b\tan(\delta_{12})}{(f+b)}\right)\right], \qquad (6d) \\ \dot{X} &= v_x\cos(\psi) - v_y\sin(\psi), \qquad (6e) \\ \dot{Y} &= v_x\sin(\psi) + v_y\cos(\psi). \qquad (6f)\end{aligned}$$

The control input u is given by

$$u = [M_{12} \quad M_{34} \quad \delta_{12} \quad \delta_{34}]^{\mathsf{T}}, \qquad (7)$$

where $F_{x,12} = M_{12}/R$ is the amount of available force on the front axle and $F_{x,34} = M_{34}/R$ is the amount of force available on the rear axle. M_{12} and M_{34} is the amount of torque on the front and rear axle, respectively. R is the tire rolling radius.

Values on all the parameters are obtained from [9], and can be seen in Table II.

B. Path Generator

The path generator takes a manually defined map with checkpoints and creates a path through the checkpoints in the correct format for the path planner described in subsection III-C. The path generator runs once during the initialization of the controller; it gives a path with discrete points separated by a euclidean distance d_{path} between subsequent points. The path generator supplies speed references and also stop flags

Parameter	Value
m [kg]	607
m_j [kg]	40
J_z [kgm^2]	1181
f [m]	1.1
b [m]	0.9
C_{12} [kN/rad]	2.6
C_{34} [kN/rad	2.6
g [m/s^2]	9.82
f_r	0.0175
ρ [kg/m^3]	1.22
c_{air}	1.3
A [m^2]	2.25
R [m]	0.318

Table II
PARAMETERS

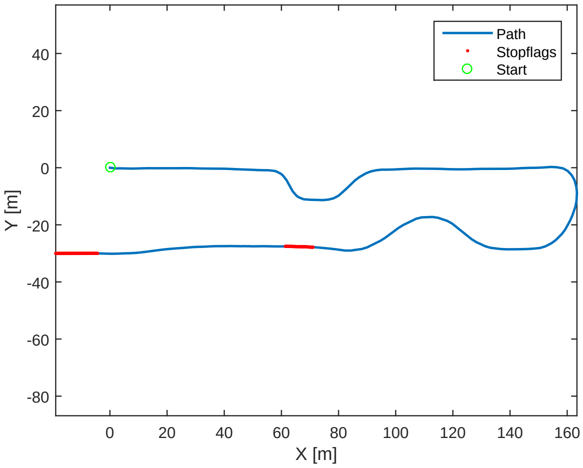

Figure 3. Examination Track

in the case of a stop light or stop sign on the path. It takes an input which contains manually defined checkpoints and translates these into a path. Each checkpoint is created to define a change in position, speed reference and/or a stop sign. It's also possible to translate a set of GPS-coordinates to match the input of the path generator. In Figure 3, is a track generated from the path generator which is a recording of the track to follow in the examination.

C. Path Planner

The path planner takes the path generated from the path generator, subsection III-B, and the current estimation of the position as inputs. At each iteration the path planner takes the point on the path with the shortest euclidean distance to the current position, as its position on the path. The path planner then provides references for the path within the look ahead distance, which is defined as the approximate distance traveled during a number of sampling periods. This equation uses the speed reference as the traveling speed, which is constant for the whole track. The reference outputs are speed, position and heading.

D. Controller

The aim for the controller is to follow the predefined path closely, while at the same time fulfilling the comfort constraints on acceleration, braking, and cornering, seen in Table III [9]. Cornering is the latitudinal movement for the RCV.

	Acceleration [$\frac{m}{s^2}$]	Jerk[$\frac{m}{s^3}$]
Acceleration (longitude)	[0, 1]	[0, 0.5]
Braking (longitude)	[-1, 0]	[-0.5, 0]
Cornering (latitude)	[-2, 2]	[-1.5, 1.5]

Table III
COMFORT CONSTRAINTS [9]

Since there are various constraints on both the control signals and the state variables, we chose to use an MPC; it solves an optimization problem during each sample time in which it is simple to implement constraints.
The MPC takes a linear model, the current state and a reference as inputs and computes the optimal control signal according to the weights and constraints put in the MPC. The optimization problem that defines the MPC is given in (8), where $f(x,u)$ is the non-linear continuous state space model given in Eq. (6a) - (6f), T_s is the sampling time and Φ and Γ describe the discretized and linearized $f(x,u)$ around a working point (x_0,u_0). W and W_u are positive semi-definite weight matrices. The constraint (8b) describes how the system evolves from one time step to another, we derive this system further later in this subsection. In (8c) and (8d), we ensure that some physical constraints on the states and controls are followed, e.g. the speed has to be lower than some max speed or a limitation on the maximum/minimum torque. In (8e), we limit the change of the control signal from one sampling time to the next.

To simplify our implementation of the MPC, we use only two inputs; for torque we use $M_{12} = M_{34} = M/2$, where M is the total applied torque; for the steering angle we use $\delta_{34} = -\delta_{12} = -\delta$, meaning our control vector can be written as

$$u = [M \quad \delta]^\intercal. \tag{9}$$

Now we will describe the linearization and discretization in further detail.

1) Linearization: To linearize a non-linear continuous time-invariant system

$$\dot{x} = f(x,u), \tag{10}$$

around some working point (x_0,u_0), we perform the following calculations.

$$\dot{x} = f(x_0,u_0) + J_x|_{(x_0,u_0)}\,(x(t)-x_0) + J_u|_{(x_0,u_0)}\,(u(t)-u_0), \tag{11}$$

where J_x and J_u are the Jacobian matrices of $f(x,u)$ with respect to x and u, respectively.

2) Discretization: When we have a linear time-invariant system as in Eq. (11), we can discretize the system by solving the system of equations under zero-order hold. Note that we assume that there are no delays in the system.

$$x(k+1) = T_s f(x_0,u_0) + \Phi(x(k)-x_0) + \Gamma(u(k)-u_0), \tag{12}$$

where

$$\Phi = \exp\left(J_x|_{(x_0,u_0)}\, T_s \right)$$

$$\min_{u} \sum_{k=1}^{N} (x(k) - x_{ref}(k))^T W (x(k) - x_{ref}(k)) + (u(k+1) - u(k))^T W_u (u(k+1) - u(k)) \tag{8a}$$

$$\text{s.t.} \quad x(k+1) = T_s f(x_0, u_0) + \Phi(x(k) - x_0) + \Gamma(u(k) - u_0), \tag{8b}$$

$$x_{min} \leq x(k) \leq x_{max}, \tag{8c}$$

$$u_{min} \leq u(k) \leq u_{max}, \tag{8d}$$

$$u_{minrate} \leq u(k+1) - u(k) \leq u_{maxrate}. \tag{8e}$$

and

$$\Gamma = \int_0^{T_s} \exp\left(J_x|_{(x_0,u_0)} \tau \right) d\tau \; J_u|_{(x_0,u_0)} .$$

3) Numerical weights and constraints: The weights for the MPC are as follows:

$$W = \begin{bmatrix} 20 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 100 & 0 & 0 \\ 0 & 0 & 0 & 0 & 100 & 0 \\ 0 & 0 & 0 & 0 & 0 & 80 \end{bmatrix}, \; W_u = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.1 \end{bmatrix}. \tag{13}$$

We chose to put zero weight on v_y and $\dot{\psi}$ since we do not have a reference for these states. We will have piece-wise constant references for v_x, and new X, Y and ψ references each sample time. The lower speed weight in relation to the other weights means that the MPC can lower or increase the speed to help reach the target X, Y and ψ. Further, the constraints for the MPC can be found in Table IV, all units are SI units. For the control change constraints, we have set that $|dM| \leq 400T_s$, and a similar expression for $d\delta$; this expression means that during one second, M can change at most 400 Nm. A change from full negative torque to full positive torque will thus take 2 seconds at least. To limit the acceleration, M is also limited to $|M| \leq 400$ Nm.

E. State Estimation

The states are estimated using an Unscented Kalman Filter (UKF), which takes into account the system model and the input from the sensors. Sensor inputs come from a three axial accelerometer, a three axial gyroscope and a GPS. The details are provided in [9].

IV. Implementation

A. Model and Vehicle Problems

The examination path contains two stops where the first is a temporary stop, and the last the final stop. However, this is not easily achieved due to two problems. Firstly, the only method to autonomously break the vehicle is to reverse the engines, which is not a very good method to get the vehicle to a complete halt. Secondly, the model of the vehicle is unstable when $v_x \to 0$. Because of these problems, there is no way to get a simulation where the vehicle makes a complete stop at a specific position. So for the simulation, the vehicle reverses the engines using a PI controller with a reference speed input of 0 when a stop flag is seen and then starts to accelerate to continue with the path when v_x gets close to 0. When the final stop flag is discovered the vehicle reverses the engines and the simulation is aborted when v_x got close to 0.

B. Hardware-in-the-Loop Simulation

To test and verify the path generator, path planner, and the MPC before connecting with the RCV, Hardware-in-the-Loop (HIL) simulations are performed. This is done using Speedgoat's Mobile real-time target machine (SGTM) [10], and Simulink Real-Time [11]. The state space representation of the bicycle model, in equations (6a) - (6f), are used to build a model of the RCV in Simulink. The path generator, path planner, and the MPC are loaded and run on the SGTM, and HIL simulations are performed by connecting each of the Controller Area Network (CAN) ports on the SGTM with each other, the model and the controller sent and received data on separate CAN ports.

The idea behind using HIL simulations is that the path generator, path planner, and the MPC are connected to the real-time target machine, and run as if they were connected to the RCV.

Simulink Real-Time is used to load the Simulink model of the RCV onto the real-time target machine. With the path generator, path planner, and the MPC connected to the real-time target machine, a simulation of the autonomous path following can be run in real-time with Simulink Real-Time.

C. KTH Research Concept Vehicle

More information about the RCV can be found in [12], [13]. The idea is to use the same controller used in the HIL-Simulation and there should not be any differences for the controller side of the system.

V. Results

In order to test and verify the system a number of checkpoints, based on a earlier manually driven test run is used. This track can be seen in Figure 3. The task is to follow these recordings as accurately as possible.

A. Hardware-in-the-Loop Simulation

The setup of the simulation is explained in subsection IV-B. The trajectory of the simulated RCV and the target path can be seen in Figure 4. The simulated path only

State Size			Control Size			Control Change		
x_{min}	x	x_{max}	u_{min}	u	u_{max}	du_{min}	du	du_{max}
0	v_x	$\frac{30}{3.6}$	-400	M	400	$-400T_s$	dM	$400T_s$
-1	v_y	1	-0.22	δ	0.22	$-0.22T_s$	$d\delta$	$0.22T_s$
-1	$\dot{\psi}$	1						
-∞	x	∞						
$-\infty$	y	∞						
-2π	ψ	2π						

Table IV
CONTROLLER CONSTRAINTS

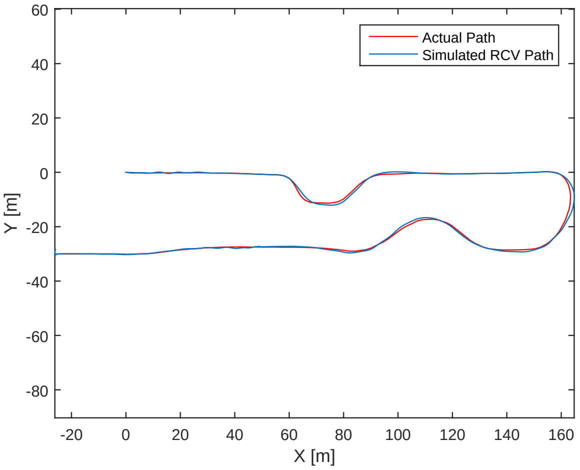

Figure 4. Resulting path of HIL-Simulation

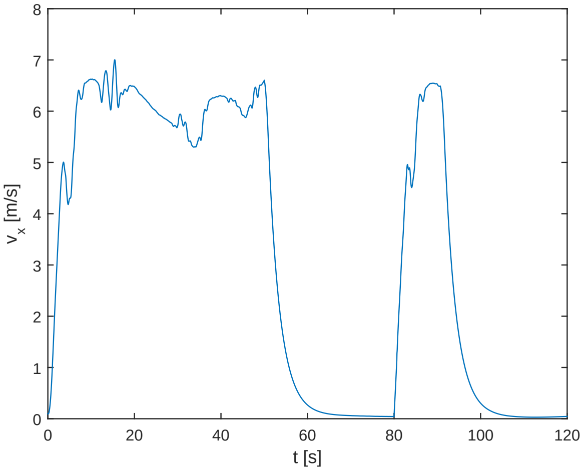

Figure 5. Velocity of the vehicle in HIL-Simulation

give a maximum deviation of 1.58 m from the path on the Examination Track seen in Figure 3, which is during the curve on the right-most end of the curve. The mean deviation is 0.21 m. The vehicle is intended to stop at predesignated points; it is not successfully able to stop due to the stopping problem mentioned in subsection IV-A. The vehicle slows down to $v_x \approx 0$ m/s as can be seen in Figure 5, but never makes a complete halt.

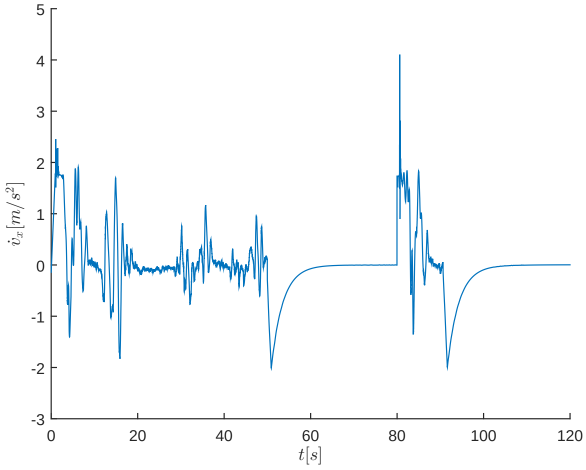

Figure 6. Longitudinal acceleration of the vehicle in HIL-Simulation

In Figure 6 and 7, we find that the simulation does not conform to the comfort constraints; the longitudinal acceleration reaches 2 m/s^2 while the set constraint is 1 m/s^2. We find that these constraints do not seem reasonable, however, these constraints mean that it would take approximately 10 seconds to go from 30 km/h to 0 km/h, and the same for 0-30 km/h. This is a very slow acceleration and we question the accuracy of the numbers. The jerk appears to be more important, but there does not exist a proper way to measure the jerk from the RCV, and thus we question the accuracy of those numbers as well.

B. *Experimental*

We were not able to perform a proper test on the test track due to a combination of bad weather and time constraints, as such there are not any experimental results.

VI. CONCLUSION

During this short project we have found that MPC is a reasonable choice of controller to control a vehicle, mainly

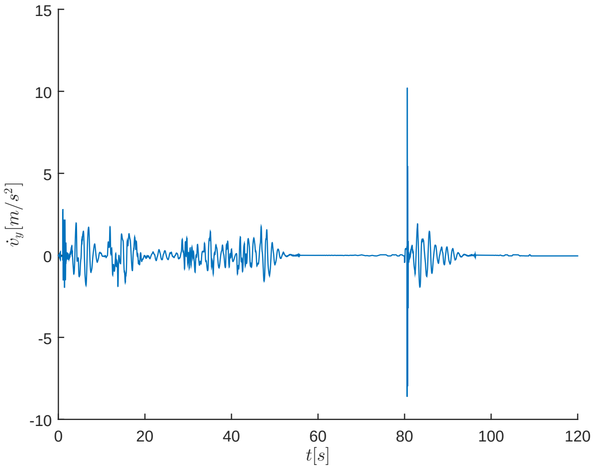

Figure 7. Lateral acceleration of the vehicle in HIL-Simulation

due to the ease of setting controller constraints. We were not able to test the controller on the RCV due to time shortage; the weather constrained us during the scheduled test and the RCV got its wheels taken off shortly thereafter. The few preliminary tests we were able to do suggest that it would be able to work.

References

[1] K. D. Do, Z.-P. Jiang, and J. Pan, "A global output-feedback controller for simultaneous tracking and stabilization of unicycle-type mobile robots." *IEEE Transactions on Robotics*, vol. 20, no. 3, pp. 589–594, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/trob/trob20.html#DoJP04

[2] A. P. Aguiar and J. P. Hespanha, "Trajectory-tracking and path-following of underactuated autonomous vehicles with parametric modeling uncertainty," *IEEE TRANS. ON AUTOMAT. CONTR*, 2005.

[3] C. G. L. Bianco, A. Piazzi, and M. Romano, "Smooth motion generation for unicycle mobile robots via dynamic path inversion." *IEEE Transactions on Robotics*, vol. 20, no. 5, pp. 884–891, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/trob/trob20.html#BiancoPR04

[4] A. Piazzi, C. Guarino Lo Bianco, M. Bertozzi, A. Fascioli, and A. Broggi, "Quintic g2-splines for the iterative steering of vision-based autonomous vehicles," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 3, no. 1, pp. 27–36, Mar 2002.

[5] P. Falcone, F. Borrelli, J. Asgari, H. Tseng, and D. Hrovat, "Predictive active steering control for autonomous vehicle systems," *Control Systems Technology, IEEE Transactions on*, vol. 15, no. 3, pp. 566–580, May 2007.

[6] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, "Stanley: The robot that won the darpa grand challenge: Research articles," *J. Robot. Syst.*, vol. 23, no. 9, pp. 661–692, Sep. 2006. [Online]. Available: http://dx.doi.org/10.1002/rob.v23:9

[7] E. Wennerström, S. Nordmark, and B. Thorvald, *Vehicle Dynamics*. KTH Royal Institute of Technology, 2011.

[8] P. Tomner, "Design and implementation of control and actuation for an over-actuated research vehicle," Master's thesis, KTH Royal Institute of Technology, 2014.

[9] A. L. O. O. A. S. Robert Ekman, Henrik Koponen, "Autonomous driving and steering feedback rcv," *KTH Internal Report*, January 2015.

[10] Speedgoat. Mobile Real-Time Target Machine. Accessed: 2014-11-19. [Online]. Available: http://www.speedgoat.ch/Products/Real-timetargetmachines-Mobile.aspx

[11] Mathworks. Simulink Real-Time. Accessed: 2014-11-19. [Online]. Available: http://se.mathworks.com/products/simulink-real-time/

[12] O. Wallmark, M. Nybacka, D. Malmquist, M. Burman, P. Wennhage, and P. Georén, "Design and implementation of an experimental research and concept demonstration vehicle," 2014.

[13] KTH Transport Labs. KTH Research Concept Vehicle. Accessed: 2014-11-19. [Online]. Available: https://www.kth.se/en/forskning/forskningsplattformar/transport/initiativ/t-labs/projects/kth-research-concept-vehicle-1.476469