Design And Implementation Of a Controller To Advance In Multi-Lane Traffic Similar To Human Behavior Using Control Improvisation

Filip Klaesson, Jin Ge, Richard M. Murray
California Institute of Technology

***Abstract*—An important traffic situation that needs to be considered in the development of autonomous vehicles is multi-lane driving. When developing a control system for multi-lane driving, it is crucial to consider the human-vehicle interaction. Another aspect is whether the overall efficiency actually improves when the majority of cars change lane autonomously. One approach to design a control system for autonomous multi-lane driving considering the traffic efficiency and the human-vehicle interaction is to design a improvisation controller to simulate the human behaviour. The control system can be divided into two problems. The first is to design a randomized lane-changing rule which determines if a possible lane change should be carried through. The second part is to move the car autonomous to the desired pose. In this study, the second part is implemented on an F1/10 unit to demonstrate autonomous lane changes. The F1/10 unit is using SLAM in order to localize and build a map of the environment based on lidar data. The path planning is implemented with motion primitives and the trajectory is tracked with a PD controller.**

I. Introduction

Since autonomous vehicles are believed to improve the traffic in the future with higher traffic flow, increasing safety etc., it is of great importance to study control systems for different traffic situations and how to implement them. Even though great advancements has already been made with multiple autonomous features implemented on commercial car, e.g. adaptive cruise control and autonomous parking, there are still many complex traffic situations that has to be taken care of in order to reach an entirely autonomous vehicle. A particularly common traffic situation that is crucial to automate is multi-lane driving. Research on how to advance in such traffic has already been made and different approaches has been developed, e.g. by Du, Wang and Chan in [1], which focus on the actual lane changing maneuver itself. In [1], a lane change is carried out by following a planned trajectory if one cannot reach the desired speed in the current lane. Tesla has already implemented autonomous lane changes where the driver has to give a turning signal in order to evoke the lane change according to [2]. In order to reach a fully autonomous vehicle, the decision has to be taken by the vehicle itself. However, with a naive decision making rule deciding if a possible lane change should be carried through or not, another concern regarding traffic efficiency arises. For instance, Evans and Peters in [3], has predicted that traffic efficiency might decrease when the majority of cars run on adaptive cruise control. If naive lane changes are taken into account, a ping-pong effect might occur and the traffic efficiency would get even worse. Another pivotal aspect which has to be considered is the human-vehicle interaction. A naive decision making rule would evoke a large number of lane changes which is most likely not appreciated by human passengers and other traffic participants. Therefore, a more human-like decision making rule is expected to be higher valued by the community.

In this paper an approach based on improvisation control is proposed. The improvisation control problem is more deeply described in [4] and [5]. The considered control system can be divided into mainly two problems: A randomized lane-changing rule, and a control system that moves the vehicle to the desired pose. The randomized lane-changing rule generates a distribution from which the decision to change lane or to stay in the current lane is based on. In order to move the vehicle to the desired pose, multiple problems need to be considered in order to implement the control system. To begin with, hardware such as sensors and actuators, etc., have to be configured and set up such that data about the vehicle and the environment is gathered properly. The sensor data is used to map the environment, which allows the vehicle to localize itself withing the environment. This can be done using Simultaneous Localization And Mapping, called SLAM. When the vehicle can localize in a fix global frame, path planning to the desired pose has to be implemented online and the generated trajectory has to be tracked sufficiently close. In this paper, the focus is on the second part of the control system and the main goal is to implement it on an F1/10 unit using Robot Operating System, ROS.

This paper is organized as follows: Section II includes a preparation that consist of the basic idea of the approach, followed by a brief introduction to ROS. Section III describes the F1/10 units architecture and hardware used and its implementation. In addition, a mathematical model of the F1/10 units kinematics will be presented. In Section IV, SLAM is briefly described and implemented. This is followed by a brief description of path planning with motion primitives in Section V, after which the vehicle is able to localize and plan a trajectory. In section VI, a low level control, LL control, to map the high level control output to the vehicles dynamics is designed. In Section VII, the HL control is designed to follow the desired trajectory. Experimental results will be presented and discussed in Section VIII and conclusions based on the discussion will be presented in Section IX. Lastly, future work and applications will be considered in Section X.

II. Preparation

A. Basic idea

The approach that will be used in this study to create and implement the control system on an F1/10 unit is divided into 4 parts: sensor and actuator data processing, navigation and localization using SLAM with lidar data, path planning using motion primitives and trajectory tracking.

Before every component in the control system is more deeply examined, the basic idea of the setup is described. Consider the feedback loop illustrating the setup in Fig. 1. The reference signal r feed to the control system is the desired pose and the output y is the current pose. The pose deviation is used as input into the HL controller to calculate an appropriate output u_{HL} in terms of the F1/10 units kinematic variables, speed v and front tire steering angle δ. The HL output u_{HL} is required to be converted into a Pulse Modulation Width signal, PWM, to be used as input to the actuators. For this purpose, a LL controller is introduced to handle the conversion from speed v and front tire steering angle δ to PWM, which is the output u_{LL} from the LL controller.

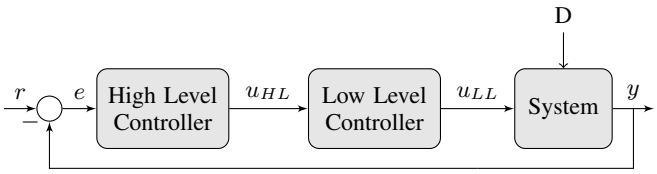

Fig. 1: Block-diagram describing the feedback loop.

B. Robot Operating System

ROS is an open source robotics middleware, which serves the purpose of providing services in robotics development. It abstracts away the communication between processes to allow easy data sharing. A process in ROS is called a node, and multiple nodes can be combined to create a graph of nodes, which can communicate with each other. Nodes communicate and trade messages/services in so called topics that is data streams with specific message types. A node can write data to the topic by initializing a publisher and publish messages of the correct message type to the topic. Other nodes can then access the data flowing in the topic by initializing a subscriber and subscribe to the topic. In this study ROS Indigo is used on the F1/10 unit. All code related to this research is provided in [6] and described in the appendix.

III. F1/10 units architecture

A. Architecture

F1/10 is a project developed by University of Pennsylvania, which takes an Traxxas RC car and rebuilds it to a unit for autonomous testing and racing. On the F1/10 unit, a Nvidia Jetson TX1/TX2 is mounted running Ubuntu Xenial 16.04.01 to use for data processing and computations. The sensors mounted on the unit are a Hokuyo URG-04LX-UG01 Lidar and a SparkFun 9DoF Razor IMU M0, that will be used for gathering data about the environment. The Hokuyo lidar gathers data with a frequency of 10 Hz, a range of $5,6$m and a angular range of $240°$ with a resolution of $0.352°$. The range data from the Hokuyo lidar is send via the serial port to the $hokuyo\_node$, which is then published in a topic called $scan$. By subscribing to the $scan$ topic, one can easily use the lidar data for different purposes, e.g. construct a wall following control or map and localize with SLAM. This will be covered in the next section. The actuators used are a Titan 12-Turn 550 motor, which is connected to a Traxxas ESC XL5 to control the throttle, and a Traxxas Servo 2080 to control the steering. A Teensy 3.1 micro-controller is used to generate the appropriate signal and feed it directly to the esc and servo. The teenzy subscribes to a topic called $drive\_pwm$ where integers corresponding to the desired PWM signal are published. The PWM signal is a 100Hz square wave in which the duty cycle varies. According to the F1/10 units website [7], a duty cycle of 10% corresponds to the signal the motor needs to go in full reverse and the servo the adjust the steering angel of the front wheels maximum to the left.

From a 20% duty cycle, maximum throttle from the motor and a maximum steering angle to the right by the servo is attained. Also according to the F1/10 units website [7], the mapping between integers feed to the teenzy and corresponding PWM is experimentally tested and given as described in Table I. A Teenzy value of 6554 will generate a PWM signal with 10% duty cycle and a Teenzy value of 13108 will generate a PWM signal with 20%. For simplicity, the Teenzy values are mapped to the interval $[-100, 100]$. In this research, these will be called PWM values. Accordingly, a PWM value of 100 corresponds to full throttle and maximum steering angle to the right and a PWM value of -100 corresponds to a full reverse throttle and a maximum steering angle to the left.

PWM [%]	Teenzy Value	PWM Value
10%	6554	-100
20%	13108	100

TABLE I: Mapping between PWM, Teenzy values and PWM values.

The F1/10 units architecture is well described by Fig. 2 from [7]. It shows how all components including sensors, Jetson and actuators, contribute. One additional component is added to the unit that is not shown in Fig. 2. In order to run and edit code wireless from a remote laptop, a Picostation m2 is mounted and configured to the Jetson as a hotspot so that a remote laptop can use ssh to run scripts on the Jetson wirelessly.

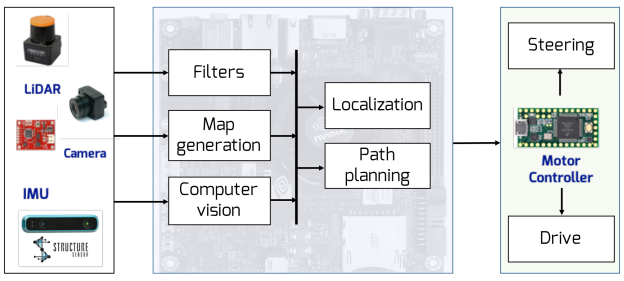

Fig. 2: F1/10 units high level architecture.

B. Mathematical modeling of four-wheeled ground vehicle

Fig. 3 from [8] illustrates a well known and widely used model of a four-wheeled ground vehicle. This model, called the bicycle model, will be used in this study to represent the F1/10 unit.

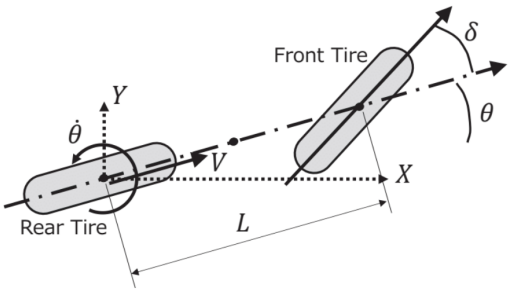

Fig. 3: The bicycle model.

L is the wheelbase, V is the speed, θ is the orientation of the vehicle and δ is the front tire steering angle. For the F1/10 unit, the wheelbase L is 0.33m. From Fig. 3, the kinematic equations of the vehicle can be derived:

$$\dot{X} = V\cos\theta \tag{1}$$
$$\dot{Y} = V\sin\theta \tag{2}$$
$$\dot{\theta} = \frac{V}{L}\tan\delta \tag{3}$$

Since the lane change is performed on a straight highway with high speed, the orientation θ of the car and the front tire steering angel δ will be small at all times. This motivates that a linearization of the kinematic equations still model the system accurately. Linearization the kinematic equations around $V = V_0$, $\theta = 0$ and $\delta = 0$ gives the following linearized equations:

$$\dot{X} = V \tag{4}$$
$$\dot{Y} = V_0\theta \tag{5}$$
$$\dot{\theta} = \frac{V_0}{L}\delta \tag{6}$$

When a constant PWM value is feed to the teenzy, the power output in the motor and servo is constant, leading to a constant speed and front tire steering angle. This means that the input to the F1/10 unit is speed v and front tire angle δ. The state space representations state vector $\mathbf{x}$ and input vector $\mathbf{u}$ is therefore chosen as:

$$\mathbf{x} = [X, Y, \theta]^T \tag{7}$$
$$\mathbf{u} = [V, \delta]^T \tag{8}$$

The linearized system can now be written on a state space for as:

$$\dot{\mathbf{x}} = A\mathbf{x} + B\mathbf{u} \tag{9}$$

where the matrices A and B are given by:

$$A = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & V_0 \\ 0 & 0 & 0 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & \frac{V_0 h}{L} \end{bmatrix} \tag{10}$$

IV. Simultaneous Localization And Mapping

Simultaneous Localization And Mapping, SLAM, is a optimization method used to map the environment and localize in it using lidar scan data. SLAM uses scan matching which means that a small local map is built for every lidar scan. These small local maps are then matched against the global map in order to extend and update the global map and keep track of how the vehicle is moving. Fig 4 from [7] shows the basic idea of scan matching where the goal is to find the new pose of the vehicle by comparing the new scan with the global map.

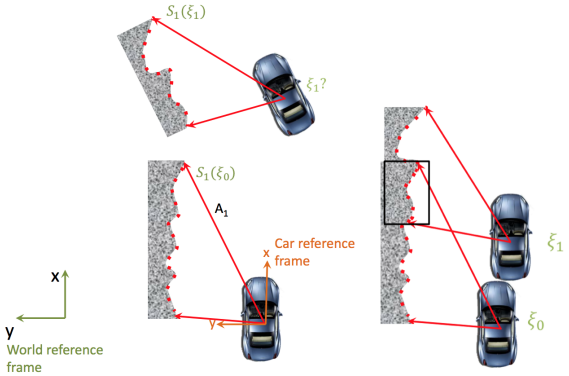

Fig. 4: Scan matching using SLAM.

Scan matching is nothing else than an optimization problem. The most likely position of the vehicle ξ^* is the position that minimizes the mismatch between two consecutive scans. The mismatch is calculated by the following sum:

$$\Sigma_{i=1}^{n}[1 - M(S_i(\xi))]^2 \tag{11}$$

where $S_i(\xi)$ is the position of the scan point i in the world frame, M is a function that measure the match of $S_i(\xi)$ with respect to the previous scan and returns a value between 0 and 1, where 1 means perfect match and 0 no match at all. There are many things to consider when designing the SLAM algorithm, e.g. how to design the function M and how to take obstacles visible in the previous scan but not in the current scan into account. However, there are numerous SLAM algorithms developed and several of them are already implemented in ROS. In this research, a ROS package called Hector SLAM is used. In practise, Hector SLAM subscribes to the *scan* topic and produces the transform from the fixed world frame to the base frame of the vehicle and returns the estimated pose of the vehicle relative the world frame. A map of the environment is then successively built up. Fig. 5 shows the map being generated with Hector SLAM on the F1/10 unit.

The pose calculated by Hector SLAM is published in a topic called *slam_out_pose*. This data, position and orientation, will be used while tracking the desired trajectory.

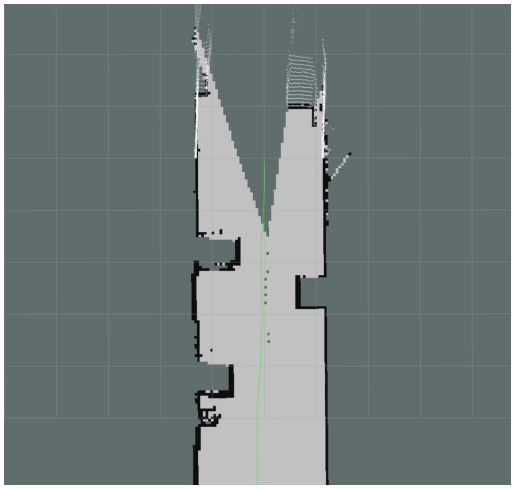

Fig. 5: Hector SLAM map built up successively.

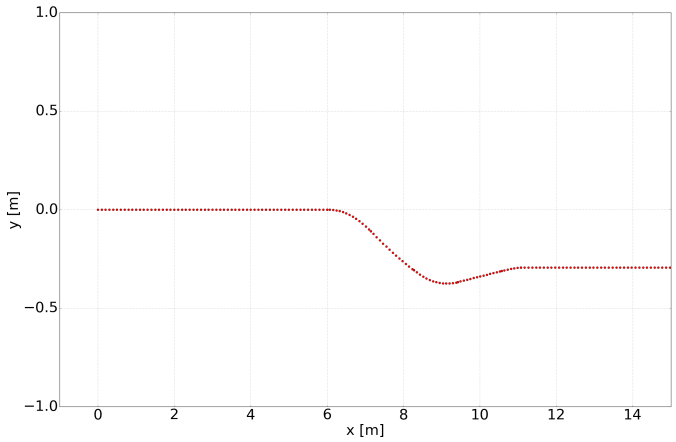

Fig. 6: Trajectory generated with motion primitives.

V. Path planning using Motion Primitives

There are multiple methods to plan a path for vehicles. Yet, a problem with online path planning is the amount of time it takes to calculate the path, which is too long in many applications. In traffic situations, it is important to generate a path quickly but also to guarantee its safety. In [9], an efficient and formal path planning approach is developed for autonomous vehicles. In order to generate a safe trajectory, the reachable sets of all traffic participants are calculated. The reachable sets can be represented as e.g. zonotopes to simplify the reachability analysis to set operations between each traffic participants reachable set. There are different ways to generate the trajectory within the safe set. MPC is a well known method to generate an optimal trajectory in some sense while considering hard and soft constraints. However, the computational time might be to long in traffic applications. [9] propose using pre-calculated motions primitives to efficiently find a trajectory. The motion primitives are short trajectory segments associated with a set of parameters such as speed and front tire steering angle. The pre-calculated motion primitives are stored in a library. In order to generate a trajectory, motion primitives from the library are put together using a tree search. The path generated may or may not be the optimal solution depending on the tree search algorithm, e.g. width first search and A* would generate different trajectories. In this study a greedy search is used to generate the trajectory quickly. A greedy search makes the optimal choice of path at each step independently of the other steps. Fig. 6 show the trajectory generated using motion primitives with a greedy search. This trajectory will be used as desired trajectory for the rest of this research.

VI. LL Control Design
PWM - Kinematic correlation

Before the HL controller is designed, the correlation between the HL output u_{HL}, which is speed V and front tire steering angle δ, and the corresponding PWM value feed to the actuators, needs to be determined. A LL controller is introduced to take care of this conversion as seen in Fig. 1. In order to determine this relation, two different experiments will be carried out.

Recall that the interval for PWM values, $[-100, 100]$, was mapped from the Teenzy interval, $[6554, 13108]$. The Teenzy interval from the F1/10 website is experimentally determined using a different setup. Therefore, the PWM intervals bound by the maximum/minimum speed and maximum/minimum front tire steering angle is expected to differ from $[-100, 100]$.

A. PWM - Speed Correlation

To determine the speed corresponding to a certain PWM value, an experiment is carried out where a constant PWM value is applied for the speed while the steering is manually controlled. The position and timestamp of the F1/10 unit is published by SLAM in the topic $slam\_out\_pose$. From the discrete position and timestamp data, an approximation of the speed is calculated as the difference in position divided by the difference in timestamp for two consecutive data points. Since the position data from the SLAM has an uncertainty, the speed calculated is very noisy. To get a more realistic representation of the actual speed, the calculated speeds are run through a low pass filter. The extract the speed corresponding to the constant PWM value applied, one can look at the acceleration to determine an interval in which the speed can be considered constant. The acceleration is calculated as the difference in speed divided by the difference in timestamp between two consecutive calculated speed points. Again, the calculated acceleration is run through a low pass filter to remove noise. In this study, the vehicle is consider to have reached a constant speed when the acceleration is within $\pm 10\%$ of the maximum acceleration. Fig. 7, 8 and 9 shows three experimental results of the speed, filtered speed and filtered acceleration corresponding to the PWM values 17, 20 respectively 23.

The experiment is performed twice for every PWM value between 15 and 25 with integer steps. Now, the average speed and standard deviation is calculated in the intervals where the car is considered to have a constant speed. The result is presented in Table 10.

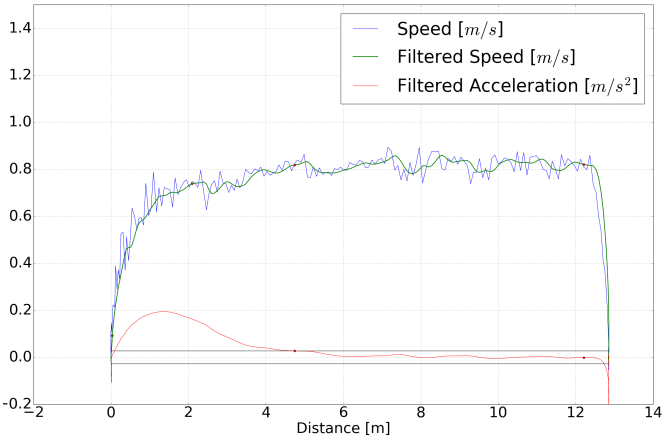

Fig. 7: Speed, filtered speed and filtered acceleration for a PWM value of 17.

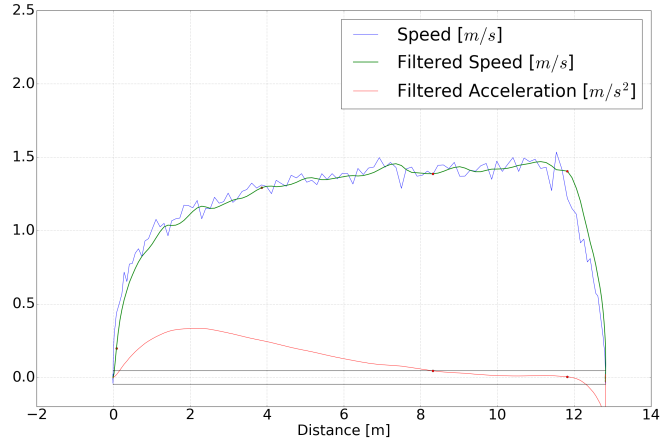

Fig. 8: Speed, filtered speed and filtered acceleration for a PWM value of 20.

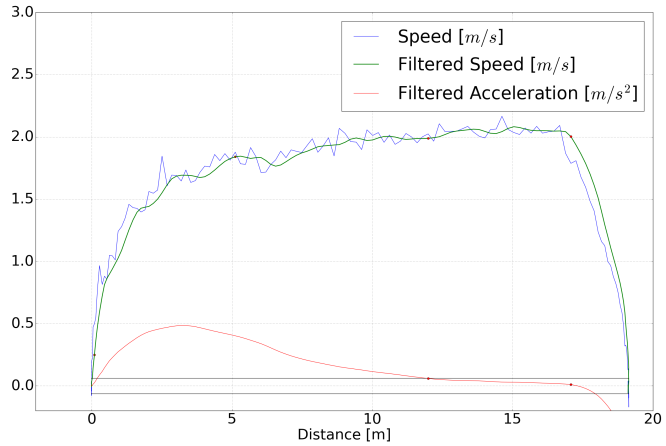

Fig. 9: Speed, filtered speed and filtered acceleration for a PWM value of 23.

PWM Value	Avg. Speed [m/s]	Std. Dev. [m/s]
15	0.506	0.033
15	0.491	0.037
16	0.597	0.051
16	0.615	0.036
17	0.815	0.033
17	0.817	0.035
18	0.984	0.048
18	0.987	0.041
19	1.193	0.047
19	1.190	0.048
20	1.438	0.045
20	1.425	0.054
21	1.582	0.033
21	1.568	0.038
22	1.870	0.041
22	1.864	0.039
23	1.986	0.042
23	2.042	0.053
24	2.092	0.089
24	2.076	0.106
25	2.293	0.041
25	2.286	0.071

TABLE II: Experimental data relating PWM values to average speed and standard deviation.

Plotting the average speed with respect to the PWM value results in Fig. 10. The standard deviation is represented by the vertical lines in each data point. A linear fit to the data gives equation 12 which is the relation between PWM value and speed sought for in this section.

$$\text{Speed [m/s]} = 0.187 \cdot \text{PWM} - 2.35 \tag{12}$$

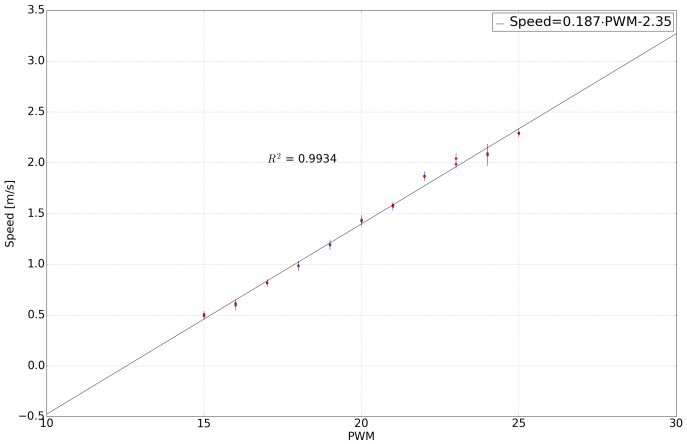

Fig. 10: Speed with respect to PWM value with standard deviation.

Now that the relation between PWM and speed is known, the focus is turned to find a similar relation between PWM and front tire steering angle.

B. PWM - Front Tire Steering Angle Correlation

In order to find the correlation between PWM value and front tire steering angle δ another experiment is designed. A constant PWM value results in a constant front tire steering angle, which cause the F1/10 unit to drive in a circle. The angular velocity of a rigid body in circular motion is given by:

$$V = \omega r = \dot{\theta} r \tag{13}$$

where V is the speed, r is the radius of the circle and $\dot{\theta}$ is the angular velocity of the rigid body. Assuming no slip between the tire and the ground, Eq. 3 and Eq. 13 gives the front tire steering angle δ as a function of only the radius of the circles:

$$\delta = tan^{-1}\left(\frac{L}{r}\right) \tag{14}$$

If the F1/1 unit is driven slow the slip can be neglected, and the front tire steering angle can be calculated from the radius from the lidar position data. The experiment is performed for different PWM values and the raw position data from SLAM is presented in Fig 11.

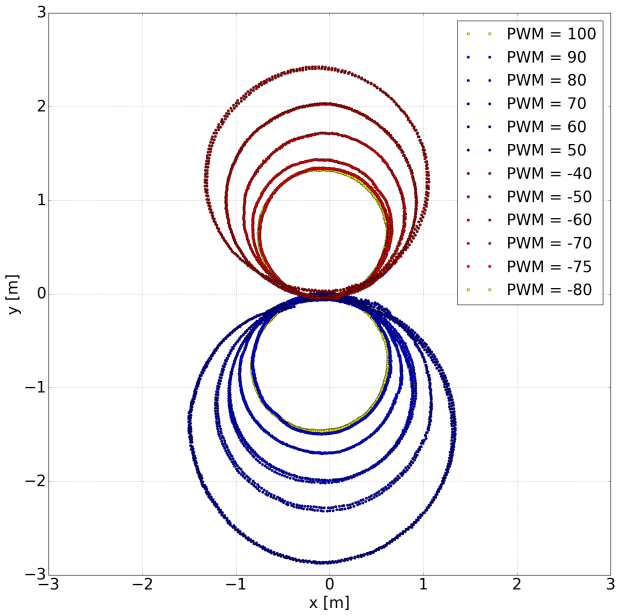

Fig. 11: Raw position data from SLAM for different PWM values.

Perfect circles are now fitted to the raw position data using least square fit. The fitted circles are presented in Fig. 12.

The circle corresponding to a PWM value of 100 is almost completely overlapped by the circle corresponding to a PWM value of 90, the same thing can be seen for the circles corresponding to a PWM value of -75 and -80. This means

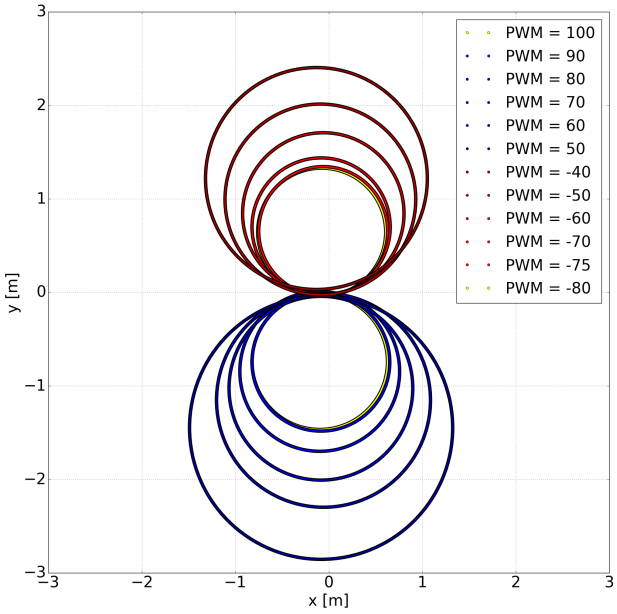

Fig. 12: Circles fitted to position data from SLAM for different PWM values.

that a PWM value above 90 and below -75 does not give a larger front tire steering angle. Hence, the interval containing the PWM values for all possible front tire steering angles is $[-75, 90]$. Eq. 14 is now used to calculate the front tire steering angle based on the radius of the fitted circles. The front tire steering angle is then plotted with respect to the corresponding PWM value and presented in Fig. 13.

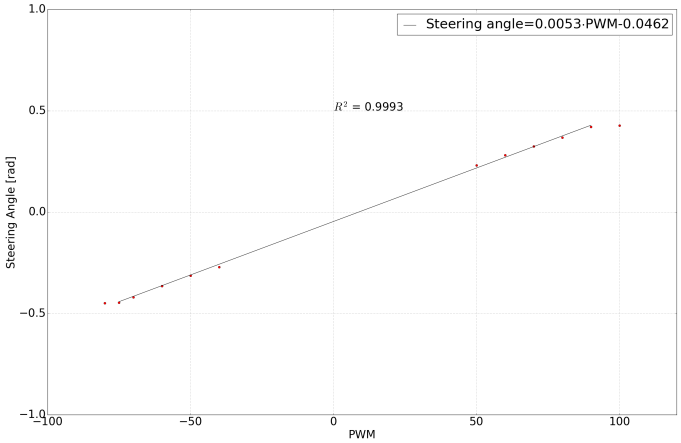

Fig. 13: Front tire steering angle with respect to PWM values.

A linear fit to the data points in Fig. 13 results in Eq. 15 that describes the relation between the PWM value and the corresponding front tire steering angle.

$$\text{Steering angle [rad]} = 0.0053 \cdot \text{PWM} - 0.0462 \tag{15}$$

Now that the correlation between the HL output u_{HL} and PWM value u_{LL} is known, the HL controller can be designed in order to follow the desired trajectory.

VII. HL Control Design

Line Following Control

From Section V, a desired discrete trajectory is obtained. In this section, a HL controller that calculates the HL output, u_{HL}, needed in order to stay on the trajectory, is designed. The approach that is going to be used is to try to minimize a predicted future perpendicular distance to the desired trajectory. Fig. 14 illustrates the geometry and notation that will be used in this section.

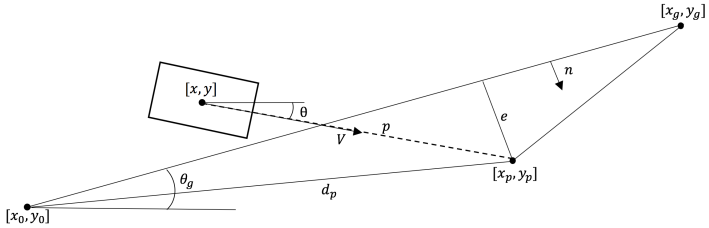

Fig. 14: Illustration of the geometry used in the line following controller where e is the predicted future perpendicular distance the line following controller should minimize.

From Fig. 14 it can be seen that (x_0, y_0) is the position of the previous discrete trajectory point and that (x_g, y_g) is the next desired trajectory point, called the goal point. The current position and heading of the vehicle is (x, y) respectively θ. The predicted future position of the vehicle is denoted (x_p, y_p) and is defined by:

$$(x_p, y_p) = (x + p\cos\theta, y + p\sin\theta) \tag{16}$$

where p is the prediction distance which is a design parameter. The predicted future perpendicular distance e can be calculated as:

$$e = d_p \cdot n \tag{17}$$

where n is the unit normal vector of the line connecting (x_0, y_0) and (x_g, y_g), and d_p is the vector from (x_0, y_0) to (x_p, y_p). These vectors can be calculated as:

$$n = [\sin\theta_g, -\cos\theta_g]^T \tag{18}$$

$$d_p = [x_p - x_0, y_p - y_0]^T \tag{19}$$

$$\theta_g = tan^{-1}\left(\frac{y_g - y_0}{x_g - x_0}\right) \tag{20}$$

In order to minimize the distance e, a PD controller is used. The orientation θ of the vehicle should be adjusted such that e is eliminated. Therefore, the desired controller is a PD controller for the angular velocity $\dot{\theta}$ with the predicted future perpendicular distance e as the the error:

$$\dot{\theta} = k_p e + k_d e \tag{21}$$

where k_p and k_d are design parameters.

However, the input is not angular velocity but front tire steering angle. Using Eq. 3, the controller Eq. 21 can be rewritten as:

$$\delta = tan^{-1}\left(\frac{L}{V}(k_p e + k_d e)\right) \tag{22}$$

Now, Eq. 22 is the desired controller that minimizes the predicted future perpendicular distance e by adjusting the front tire steering angle. The parameters k_d, k_p and p are chosen based on experiments to:

$$k_p = 7.1 \tag{23}$$

$$k_d = 2.7 \tag{24}$$

$$p = 0.1 \tag{25}$$

It is interesting to look at the pole positions with the chosen parameters. Using Eq. 22 and assuming constant speed, the state space equation, Eq. 26 can be written as:

$$\dot{\mathbf{x}} = C\mathbf{x} + D \tag{26}$$

where the matrices C and D are given by:

$$C = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & V_0 \\ \frac{Ck_p\theta_g}{1+Ck_dp} & \frac{-Ck_p}{1+Ck_dp} & \frac{-Ck_pp-k_dV}{1+Ck_dp} \end{bmatrix} \tag{27}$$

$$D = \begin{bmatrix} V \\ 0 \\ \frac{-Ck_p\theta_g x_0 + Ck_p\theta_g p + Ck_p y_0 + k_d\theta_g V}{1+Ck_dp} \end{bmatrix} \tag{28}$$

The poles of the linearized state space equation are the eigenvalues of matrix C. The poles are plotted in Fig. 15.

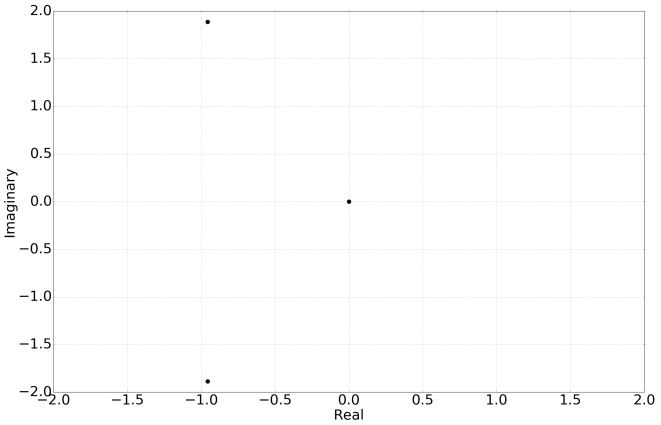

Fig. 15: Pole positions of the linearized state space equations.

The numerical values of the pole positions are presented in Table III.

p_1	0+0i
p_2	-0.96+1.89i
p_3	-0.96-1.89i

TABLE III: Numerical values of the pole positions of the linearized state space equations.

Apart from the pole in the origin, which doesn't cause instability, the real part of the poles are in the left half plane, the system is stable with the chosen controller and parameters. The pole position analysis is performed considering the linearized system using continuous time analysis. Since the sensors and controller work in discrete time, it is more representative to calculate the poles using discrete analysis. However, the continuous analysis can be used as a tool to understand how the different parameters affect the systems behaviour.

Now that the HL controller is fully designed and the HL output $u_H L$, will be converted to the corresponding PWM values and feed to the actuators by the LL controller. In the experiments, a constant speed will be used. However, a similar controller could be designed if that's not the case. In the next section, the experimental result when tracking the desired trajectory is presented.

VIII. Experimental results

Now that all components are designed, configured and implemented on the F1/10 unit it is time to perform experiments to track the trajectory calculated in Section V using the controllers designed in Section VI and VII. The experiment is performed with a constant speed of 0.8 m/s. The position data generated by SLAM is used to analyze the trajectory traveled by the F1/10 unit. The trajectory is plotted together with the desired trajectory in Fig. 16.

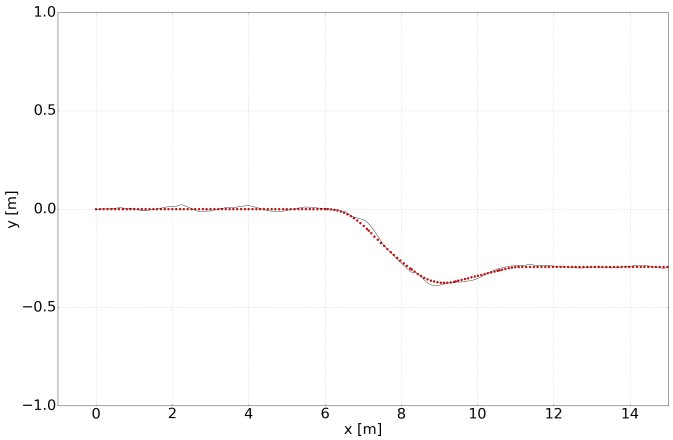

Fig. 16: F1/10 units trajectory together in black with the desired trajectory generated with motion primitives in red.

As can be seen in Fig. 16, the F1/10 unit is following the desired trajectory. There are small oscillations along the whole trajectory and an even larger deviation from the desired trajectory in the turns of approximately 3cm. It is important to realize that the accuracy of the position calculated by SLAM varies depending on the environment and the speed for instance. Some oscillations would most likely disappear with more tuning of the PD controller or by introducing an integral part to the controller. However, since the system is nonlinear there might not be a choice of PD parameters such that the deviations are decreased both when driving on a straight line and in the sharpest turns. For this purpose, the study would benefit from a nonlinear or adaptive control design and the trajectory could most likely be tracked with small deviations.

IX. Summary and Conclusion

The overall goal of this study was to design and implement a control system that takes the F1/10 unit from the initial point to a desired goal point, autonomously. The first step was to implement all hardware components. To gather data about the environment, sensors where mounted and configured. A Hokuyo lidar was used to gather distance data of the obstacles. The lidar data was analyzed by Hector SLAM that maps the environment and keeps track of the pose of the F1/10 unit simultaneously. Path planning was performed using motion primitives to quickly generate a trajectory from the initial point to the goal point. In order to track the calulated trajectory, a HL controller was designed as a line following controller. The idea behind the line following control is to minimize a predicted future perpendicular distance to the desired trajectory. A LL controller was designed in order to convert the HL output to a PWM signal that can be interpreted by the actuators. The LL controller, or the relation between the F1/10 units kinematics, was determined by performing two experiments. To map the PWM value to speed, the first experiment was performed by applying a constant PWM value and calculate the speed from the position data retrieved by SLAM. The second experiment was performed in order to map the PWM value to front tire steering angle. By applying a constant PWM value, the F1/10 unit drove in a circle. From the assumption that there is no slippage between the tire and the road, the front ire steering angle is a function of the radius of the circle. By performing the experiment in low speed, the desired relation between PWM value and front tire steering angle was calculated solely the radius of the circle. At this point, both all the hardware component and the two controllers were fully design and implemented and the final experiment was ready to be performed, which was to autonomously follow the desired trajectory generated by the path planning. The F1/10 unit was able to follow the desired trajectory despite minor deviations and oscillations, especially in the turns. From the analysis of the pole positions the system is guaranteed stable. However, the analysis is based on continuous system and controllers, but since the sensors work in discrete time the controller output is calculated in discrete time. A discrete analysis of the pole positions would be more representative. The developed and implemented control system can be used in multiple experiments to simulate a traffic situation, not only straight lane changes.

X. Future work

To start off the discussion concerning future work, lets recall to motivation of this study which was improving traffic flow and the human-vehicle interaction in multi-lane traffic using control improvisation. Since this study has dealt with the second part of the improvisation controller, the next step is to design the randomized lane-changing rule and integrate it in the current control system. In order to experimentally verify if the improvisation controller improves the traffic flow, it would be beneficial to run the experiments on a ring road with multiple F1/10 units or simulated traffic participants. There are also possible improvements to the currently implemented

control system. As discussed in Section VII, implementation of a nonlinear or adaptive control would be beneficial. This would enable the usage of the F1/10 unit to simulate more general traffic situations where the linearized model used in this study is not sufficient. Also, a speed controller to be able to track a desired speed instead of performing the experiments with a constant speed would be valuable.

ACKNOWLEDGMENT

The author would like to express great gratitude to the project supervisor, Richard M. Murray and co-mentor Jin Ge. Their support and guidance throughout the whole project has been highly appreciated. Also, thanks to California Institute of Technology, KTH The Royal Institute of Technology and NSF for making this project possible.

APPENDIX

The scripts used in implementation of this research is provided in [6]. In this section, the scripts are briefly described.

Run

To start the nodes that gathers and analyzes data, run the python script Run. It starts the nodes $hokuyo\_node$, imu_node, $serial\_node$ and $talker$ and launches the launch file $f110$. In addition to running Run, a node that feed trajectory points to the controller is needed.

Talker

$Talker$ is the node that communicates with the Teenzy. It subscribes to the topic $drive\_parameters$ where the PWM values in the interval [-100,100] are published. $Talker$ then convert the PWM value using the function $PWMconverter$, into the corresponding Teenzy value and publish it in the topic $drive\_pwm$, which the Teenzy subscribes to. $Talker$ also subscribes to the topics $record\_data$, imu, $scan$, $current\_speed$ and $slam\_out\_pose$ to gather experimental data which is then written in a text file $positions.txt$ for future analysis. The data is written into $positions.txt$ every time new position data is available.

Serial_node

Rosserial is a ROS package that contain tools to easily set up a connection with a device through a serial port. It handles the setup and all communication between the connected device and roscore. $Serial\_node$ is a node that uses rosserial to setup the connection and data transfer with the lidar. This is a standard setup and is deeply described in [7].

Hokuyo_node

The $hokuyo\_node$ is the node that receives the lidar data from the serial port. The lidar data is then converted and published in the topic $scan$ with message type $sensor\_msgs/LaserScan$. The message contains all the distance data and information one needs to map the environment. The ROS package $Hector\ SLAM$ include this node and all the tools needed.

Imu_node

Even though the IMU is not used in this research, the IMU is fully configured and implemented for future work. The code used to flash the IMU is provided as a zip file in [6]. Additionally, the imu_node is provided as the node that recieves the imu data from the serial port and publish it in the imu topic with the message type $sensor\_msgs/Imu$.

F110.launch

In order to use Hector SLAM to map and localize within the environment, a SLAM launch file has to be executed. The launch file $f110$ is a SLAM launch file modified to work with the rest of the code in the F1/10 unit. It starts multiple nodes related to SLAM and defines relevant transforms between coordinate systems. The SLAM nodes use the lidar data to estimate the pose of the F1/10 unit and publish the calculated data in the topic $slam\_out\_pose$ with the message type $PoseStamped$.

Gotocontrol

$Gotocontrol$ contains the HL controller and LL controller. However it is not a node, it is just a script containing methods which can be used by other nodes. It subscribes to the topic $slam\_out\_pose$ to get position data to calculate the HL output u_{HL}. The output is converted to the PWM value in the interval [-100,100] and is then published in the topic $drive\_parameters$. The design parameters for the PD controller can be edited in this file.

Trajectory

The desired trajectory calculated is stored in the text file $trajectory_data.txt$. $Trajectory$ reads the position data of the desired trajectory and call the $goto$ function with the next desired trajectory point.

Record

$Record$ is a node that allows the user to start and stop recording data and also erase old data from $positions.txt$. After starting the node, press r to record data, t to stop redord data and e to erase old data.

Kill

$Kill$ is a node that is used to kill the communication between the ROS nodes and the Teenzy. If emergency stop is activated, the Teenzy does not react to new data sent to it.

Running rviz on remote laptop

To visualize data running on the F1/10 unit on a remote laptop the ROS environment has to be properly set up on both devices. First, the IP address of both the Jetson and the remote laptop has to be set to static. Assume that the ip address of the Jetson is 192.168.x.y and that the ip address of the remote laptop is 192.168.w.z. SSH into the Jetson from the remote laptop by typing $ssh\ username@192.168.x.y\ -X$

in the terminal on the remote laptop. In order to transfer all data running in ROS on the Jetson to the remote laptop the ROS_MASTER_URI, ROS_HOSTNAME and ROS_ID has to be set up on both devices. This is done be entering the following commands in the terminal:

Jetson:

export ROS_MASTER_URI=http://192.168.x.y:11311
export ROS_HOSTNAME=192.168.x.y
export ROS_IP=192.168.x.y

Remote laptop:

export ROS_MASTER_URI=http://192.168.x.y:11311
export ROS_HOSTNAME=192.168.w.z
export ROS_IP=192.168.w.z

These commands has to be repeated in every new terminal. A suggestion is to put the above commands along with sourcing the catkin workspace in the $\sim$ /.bashrc file so that it is executed in every terminal. Now the data running in ROS on the Jetson can be used in the remote laptop to visualize the F1/10 unit using rviz.

REFERENCES

[1] Y. Du, Y. Wang, and C. Chan, "Autonomous lane-change controller," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 386–393.
[2] (2018) Auto lane change. [Online]. Available: https://www.tesla.com/sites/default/files/model_s_owners_manual_north_america_en_us.pdf
[3] Evans and Peters, "Cooperative adaptive cruise control: Human factors analysis," 2013.
[4] I. Akkaya, D. J. Fremont, R. Valle, A. Donzé, E. A. Lee, and S. A. Seshia, "Control improvisation with probabilistic temporal specifications," *CoRR*, 2015.
[5] D. J. Fremont, A. Donzé, and S. A. Seshia, "Control improvisation," *CoRR*, 2017.
[6] F. Klaesson. (2018) F110 github. [Online]. Available: https://gits-15.sys.kth.se/filipkl/F1tenth
[7] (2018) F110 website. [Online]. Available: http://f1tenth.org/
[8] M. Obayashi, K. Uto, and G. Takano, "Appropriate overtaking motion generating method using predictive control with suitable car dynamics," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 4992–4997.
[9] B. Schurmann, D. Hess, J. Eilbrecht, O. Stursberg, F. Koster, and M. Althoff, "Ensuring drivability of planned motions using formal methods," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, Oct 2017, pp. 1–8.