A Robotic Traffic Simulator for Teaching of Advanced Control Methods

Martin Kalúz * **Juraj Holaza** * **Filip Janeček** *
Slavomír Blažek * **Michal Kvasnica** *

* *Institute of Information Engineering, Automation and Mathematics STU in Bratislava, Radlinského 9, 812 37 Bratislava, Slovakia (e-mail: {martin.kaluz,juraj.holaza,michal.kvasnica}@stuba.sk)*

Abstract: This paper presents a development and educational application of robotic traffic simulator. Setup presented in his work consists of ten laboratory-scale vehicles designed for simulations of various traffic situations as well as the evaluation of advanced control scenarios. These can be the control of traffic fluency, such as congestion movement, vehicle group acceleration, breaking, obstacle avoidance and other situations known from everyday traffic. Further, the paper describes the technical realization of simulator from both, the hardware and software point of view. Moreover, the applicability of solution is discussed over the various situations, which can be solved in educational as well as the scientific matter. The educational value of developed traffic simulator is demonstrated on the case study, where an optimal control strategy using the Model Predictive Control was designed and evaluated by a master's degree student.

Keywords: Traffic, Robotics, Simulator, Control Education, Optimal Control, MPC

1. INTRODUCTION

The problems of traffic regulation are one of the most discussed topics in recent years, not only at the public forum, but also in the scientific circles. As the result of vehicle number growth and non-sufficient changes in the road infrastructures the undesired traffic situations become an everyday problem in all major cities. These often result in collisions, traffic jams, and overall slowdown of traffic flow which also result into the increased fuel consumption and exceeded production of gas emissions. To reduce the risk of such situations, car manufacturers equip their vehicles with information and control systems. The purpose of these systems is to help drivers to effectively solve the traffic situations.

One of the current topics addressed by many research teams in the world is the problem of computer-based traffic fluency control. Frequently used approach is the control of each vehicle as the autonomous system with the respect to the environment and other traffic participants. Another approach considers the control of a set of vehicles as one distributed system. In this scenario the upper-level control system is able to actively communicate with each vehicle, collect the operational data from it, and provide it with the information how to behave.

Various control design approaches are used to control traffic congestions movement, such as adaptive control (Kesting et al., 2008), genetic algorithm (Dezani et al., 2012), model predictive control (MPC) (Zegeye et al., 2009), etc. MPC is an advanced control method, based on computations of optimal control inputs using the information about system's model and measurement or estimation of its states. The MPC performs the prediction of system's behavior and therefore it allows control system to react to the future situations in advance. In literature, a numerous works are specifically aiming on the application of MPC to traffic congestion control (Deo et al., 2009). An interesting applications on the simulator of vehicle queues have been conducted at Czech Technical University in Prague, where several methods of distributed control of traffic have been applied (Šebek and Hurák, 2011; Martinec et al., 2012).

One of the main problems along with the development of control methods is their evaluation in practice. This fact does not apply only for scientific environment, but also for education institutions. This apply even more when the topic is the control of transportation vehicles, which are not only expensive and difficult to acquire in higher numbers, but also difficult to deploy into practical experiments. The majority of available works use the mathematical models of traffic congestions and the applications are mainly in the form of computer-based simulations (Mather and Hsieh, 2012). This trend is obvious, mostly due to the application requirements, since the execution of such experiments in real traffic would be very expensive. Due to these reasons, one of the most applied approaches is the use of small-scale laboratory models of vehicles (Marcolino and Chaimowicz, 2009). This brings an advantage of cost reduction in both, the development phase and operation.

At the Institute of Information Engineering, Automation and Mathematics (IAM), we have developed such a laboratory scale traffic simulator, which is intended for both, the research applications as well as the education purposes and students' projects.

2. ROBOTIC VEHICLES

Robotic vehicles (Fig. 1) have been developed on the low cost platform Arduino Yún [1] which consists of two separate computing units, communication peripherals and electric signal interface. Arduino Yún contains a microcomputer with Atheros processor and lightweight Linux distribution OpenWrt, which is supplemented by various peripherals such as Ethernet, WiFi, MicroSD card slot and USB interface. Second computing unit is an 8-bit microcontroller Atmel ATMega32u4 that manages the signal interface of development board. The communication between processing units is ensured by UART TTL serial link. The robotic vehicle itself acts as an extension shield for Arduino board. In this work a Shield Bot development electronic platform was used [2]. This platform consists of a car-shape electronic board, equipped with two independent DC-motor-driven wheels, motor control electronics, battery, five surface reflection sensors for line tracking, extension ports for sensors, and has been additionally equipped with ultrasonic distance sensor.

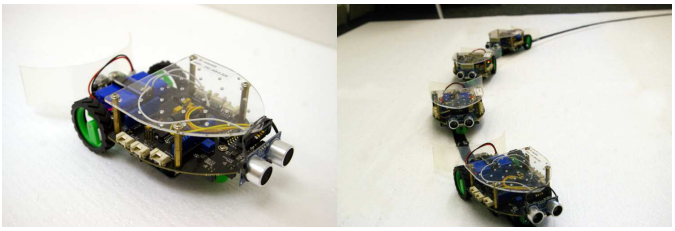

Fig. 1. Robotic vehicles

3. SOFTWARE AND COMMUNICATION

The whole vehicle control system is separated into three main parts: the low-level program in microcontroller (written in C Language) that serves the sensors, actuators and basic control of vehicle's motion; the upper-level program running in OpenWrt of Arduino Yún (written in Python) that represents back-end communication service for front-end software; and open front-end that can be any software that allows HTTP-based communication (e.g. standard Web page, MATLAB, etc). The principles of communication with MATLAB environment is shown in Fig. 2.

The communication scheme is very simple and it uses the JavaScript Object Notation (JSON) as data structure. The program or application that is intended to control the vehicle can use two type of communication methods. First is based on standard HTTP POST requests which opens the connection only for one pair of request-response at a time, and second uses the WebSocket, which is a full-duplex TCP-based protocol. WebSocket opens the communication channel and keeps it open until it is terminated by protocol instruction. This allows communication to be faster, since the connection does not need to be established every time the data is transferred. This method is also less demanding on the amount of supplementary data (headers, cookies) transferred in each message. The communication services in vehicles are designed to provide on-demand response to control software (e.g. MATLAB).

[1] `https://www.arduino.cc/en/Main/ArduinoBoardYun`
[2] `http://www.seeedstudio.com/wiki/Shield_Bot_V1.1`

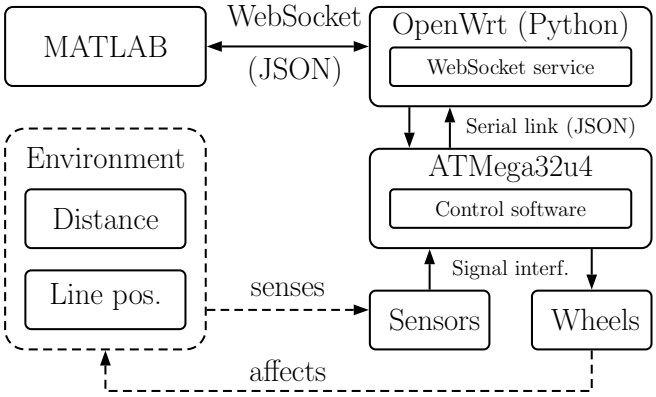

Fig. 2. The communication scheme of a single robotic vehicle using MATLAB.

Each vehicle provides three methods of connection to control software. These are:

- *WebSocket server* - In this method a control software (MATLAB, Web browser, Python, etc.) adopts a role of client and can establish one connection per vehicle using their WebSocket services published on the network. In this case the vehicles act as the servers, and communication with each vehicle is handled separately.
- *WebSocket client* - In this case the vehicles act as the clients in the communication setup. Every client establishes a WebSocket connection with a message broker, which is a Python-based service running on a separate computer. The message broker provides another WebSocket service for control software. This allows the client, which is in this case the control software itself, to simultaneously communicate with several vehicles at once.
- *HTTP server* - Provides a similar principle of communication as first method. The main difference is that it does not use the switching protocol, but separate requests instead.

4. VARIOUS CONTROL SCENARIOS

The laboratory scale traffic simulator consists of 10 real robotic vehicles, white flat surface with roads made of glossy black tape and obstacle objects, computer equipped with control software (e.g. MATLAB), wireless network to which the vehicles and computer connect, and computer vision system for recognition of vehicles' position, which however is still in the development phase. Using this setup a various control situations can be tested. The main control scenarios cover the following situations:

a) Control of the movement of heterogeneous congestions and distance between vehicles. This also include the problems of congestion acceleration and breaking as a whole.
b) Control of sudden congestion breaking in the case of unavoidable obstacle, e.g. a crash site blocking the highway.
c) Control of traffic at crossroads, where different rights of way apply.
d) Line merging in congestions.
e) Single and multiple obstacle avoidance.

f) Different tasks of automatic parking.

In developed simulator setup mostly the situations *a*, *b* and *e* can be evaluated. This is due to the lack of sensing capabilities that current system has. Currently the vehicles can sense only small amount environmental information, namely the position of guide line and distance of object strictly in the front of the vehicle. However, in the future work, the use of marker-based computer vision system for exact position detection is planned, and therefore it will be possible to apply all mentioned situations in laboratory environment. This vision system is being developed using the ArUco (Garrido-Jurado et al., 2014), an open-source OpenCV-based C++ library.

5. LEARNING OBJECTIVES

The robotic traffic simulator has found its versatile use in several courses such as the *Diploma project*, *Semestral project*, *Model Predictive Control* and *Control of Embedded Systems*, and it provides a wide spectrum of problems that students can solve. These are for example:

- *Mathematical modeling of cars* – Here students learn basic principles of modeling of physical systems (cars). The derivation of model is a prerequisite for later controller resign.
- *Identification of unknown parameters of cars* – The more complex models of cars contain the parameters that describe the dynamics of motors, steering capability, speed drop off based on battery level, etc. All of these parameters must be acquired from the experimental identification.
- *Design of simple control loops* – To become familiar with the robotic vehicles, students firstly start to work on simple control loops such as PID control for line following or distance control of two vehicles.
- *Upper-level control design* At this level, students are encouraged to design more advanced control schemes, where knowledge of observer design or model predictive control is applied.

6. USAGE OF THE TRAFFIC SIMULATOR

In the education and research at IAM, the experimenters are using mostly two software solutions developed at our institute to control the simulator. First is the MATLAB application programming interface (API) specifically designed for the traffic simulator. This API is mostly intended for the researchers and students with the higher skills in programming. The second control software is *Node-RED* [3], a universal JavaScript-based platform that allows simple program composition using the node-based visual editor.

In the courses focused on automatic control the students use the simulator mostly via the MATLAB API, since they are familiar with this computational environment. The usage is very simple and requires them only to connect their computers to the simulator's network, where robotic vehicles connect automatically after they are powered on. The design of a particular experiment (track shape, number of vehicles, control objective) depends on the current topic being taught in lesson. It can be e.g. the constrained control of vehicle distances, speed, etc. Students design their own algorithms MATLAB based on the knowledge control method and simple mathematical model of vehicles, and use the API to directly incorporate the simulator's inputs and outputs to their program. Since the simulator is a set of real objects, they can observe the behavior of the control system more realistically than just using the pure computer simulation.

6.1 Control via MATLAB

The MATLAB-based API for robotic vehicle control provides the class called `RoboBug` which ensures connection and configuration of robots, as well as the methods for data acquisition and command issuing. The use of class is very simple and intuitive. To create an object that represents an instance of robot, the following command is executed.

```
vehicle_01 = RoboBug('192.168.0.201');
```

The class constructor accepts only one argument that represents the IP address of robot within the local network. After the instance of vehicle is created the connection command can be issued.

```
vehicle_01.connect();
```

This command invokes the WebSocket connection between the MATLAB and robot. Since the WebSocket standard is not natively supported by MATLAB yet, the `RoboBug` class requites two additional dependencies to be installed. These are Java-based classes *matwebsocks* and *wsclient*, which can be obtained and installed via toolbox manager *tbxmanager* [4].

After the connection is established successfully, the robot can be directly controlled and monitored using the following commands.

```
% enable automatic line following
vehicle_01.setAuto(1);
% define speed of the vehicle
vehicle_01.setSpeed(100);
% define steering via motor power distribution
vehicle_01.setDrvFact(0.5);
% define stop distance [cm]
vehicle_01.setStopDist(2.5);
% get reading from distance sensor [cm]
distance = vehicle_01.getDistance();
% get array of reflection sensor readings
sensors = vehicle_01.getSensors();
% trigger immediate stop
vehicle_01.stop();
% disconnect from robot
vehicle_01.close();
```

The `RoboBug` class has been also extended by a Simulink library. This allows students to quickly and effectively design their own control schemes without directly using the API or writing their own code.

6.2 Control via Node-RED and SmartNodes Library

Another effective control design environment developed at IAM is the extensional library for *Node-RED*, which is primarily intended for creating the solutions for the

[3] `http://nodered.org/`

[4] `http://www.tbxmanager.com/`

Internet of Things. This framework allows its users to create the algorithms in visual schematic environment similar to Simulink. Since the *Node-RED* is not directly designed to work with control algorithms, we have developed the *SmartNodes*, a library of nodes that allow to incorporate many of commonly used controllers and supplementary features into the existing environment. The *SmartNodes* provide a set of signal routing nodes with vector data structure similar to that used by MATLAB and definition of various representations of dynamical systems and controllers such as:

- standard PID,
- polynomial controller,
- continuous-time transfer function,
- discrete-time transfer function,
- state-space system,
- state feedback controller,
- sequential search table for explicit MPC.

Additionally, some other nodes have been created for easy implementation of custom control algorithms. These are e.g. nodes for user interface controls such as switch button and slider, and universal representation of used-defined function written in JavaScript. The user-defined function allows the user to incorporate any functionality that is not directly provided by *SmartNodes*. To extend the communication features, a WebSocket service has been implemented as well.

In standard functionality the *Node-RED* is designed to perform one execution of designed scheme at a time. This functionality has been extended in order to achieve the cyclic time-based program execution that is required for evaluation of time-based control algorithms. One of the main benefits of *SmartNodes* is the capability to compile the visual scheme into standalone executable JavaScript application for web browsers. In this principle, user-defined algorithm can be exported from *Node-RED* and loaded into any device equipped with JavaScript execution environment (e.g. Web browser).

7. CASE STUDY

In this study we show how to control traffic fluency as one centralized system. We consider traffic system represented by n cars that are tracking the single-lined rounded road with diameter of 90cm. Here, the first car of the convoy denotes the *leader* that has constant speed and is chased by the other $n-1$ cars, which are denoted as *followers*. The control objective is to manipulate speed of cars (*followers*) such that they will preserve pre-defined safety distance from each other. To achieve this goal we propose to derive a mathematical model for system of n cars, which will be subsequently used as a prediction model in MPC strategy. The efficiency of the designed control algorithm will be demonstrated for the system of two cars as well as for the system of four cars. It is worth to mention that majority of the proposed results are derived by the student and form an important part in his bachelor thesis.

7.1 Model

Accurate mathematical model plays a vital role in MPC policy. This is due to the fact that the algorithm operates over it and pre-calculates future states and outputs, respectively.Therefore the model directly influences the optimized control inputs and thus the control quality. In our case, mathematical model of the aforementioned system can be intuitively derived by using basic knowledge of physics. This allows students not only to exploit heir skills from different subjects, but also to better understand the principles of the system dynamics.

To design a mathematical model of the centralized system we need to incorporate two types of dynamics. The first one is the line tracking. Here each car, based on signals from five reflective sensors, have to weight the power input between two motors such that it stays on the road. The second one is dedicated to the relationship between the measured distance from the car ahead (obtained from ultrasonic sensors) and the power added to motors. For brevity we consider that the algorithm, which allows cars to autonomously track the road, to be already embedded in each car.

The associated mathematical model of the centralized system can be defined as

$$\dot{v}_0(t) = 0, \tag{1a}$$

$$\dot{d}_i(t) = v_{i-1}(t) - v_i(t), \tag{1b}$$

where $i \in \{1, 2, \ldots, n-1\}$, $\dot{v}_0(t)$ is the derivative of the *leader's* speed , $\dot{d}_i(t)$ denotes derivative of the *follower's* distance (gap between it and the car in front of it) , $v_i(t)$ and $v_{i-1}(t)$ represents the speed of the i-th *follower* and the speed of the car ahead, respectively.

By converting (1) into discrete-time domain (e.g. via the Euler method) one obtains

$$\frac{v_0(t+1) - v_0(t)}{Ts} = 0, \tag{2a}$$

$$\frac{d_i(t+1) - d_i(t)}{Ts} = v_{i-1}(t) - v_i(t), \tag{2b}$$

where terms $v_0(t+1)$ and $d_i(t+1)$ denote *leader's* speed and i-th *follower's* distance at the next sample period Ts. Finally, the general model of the centralized system can be defined as a steady state model

$$x(t+1) = Ax(t) + Bu(t), \tag{3a}$$

$$y(t) = Cx(t), \tag{3b}$$

with matrices

$$x(t) = \begin{bmatrix} v_0(t) \\ d_1(t) \\ d_2(t) \\ \vdots \\ d_{n-1}(t) \end{bmatrix}, u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \\ u_3(t) \\ \vdots \\ u_{n-1}(t) \end{bmatrix}, y(t) = \begin{bmatrix} d_1(t) \\ d_2(t) \\ d_3(t) \\ \vdots \\ d_{n-1}(t) \end{bmatrix},$$

$$A = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ Ts & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 & 0 \\ -Ts & 0 & 0 & \ldots & 0 & 0 \\ Ts & -Ts & 0 & \ldots & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \ldots & Ts & -Ts \end{bmatrix},$$

$$C = \begin{bmatrix} 0 & 0 & 0 & \ldots & 0 \\ 0 & 1 & 0 & \ldots & 0 \\ 0 & 0 & 1 & \ldots & 0 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & 1 \end{bmatrix},$$

where $x(t) \in \mathbb{R}^n$ is the vector of states, $u(t) \in \mathbb{R}^{n-1}$ is vector of control inputs and $y(t) \in \mathbb{R}^{n-1}$ is vector of outputs. Recalling that n represents the number of cars.

Note that the state space model (3) can be extended by the line tracking dynamics. This will, however, lead to additional five binary states (from the reflecting sensors) and one control input (weighting coefficient of the motor power) per car. This would result in the total of $\tilde{x}(t) \in \mathbb{R}^n \times \{0,1\}^{5n}$ states, $\tilde{u}(t) \in \mathbb{R}^{2n-1}$ control inputs and $\tilde{y}(t) \in \mathbb{R}^{n-1}$ outputs. Hence, such augmented model is more complex and thus harder to obtain. Moreover, as a prediction model in MPC, it leads to mixed-integer programming what might cause implementation issues as the sampling period has to be increased due to the enhanced computation time. The bottom line is, the simpler model (3) is much easier to devise and thus it is more suitable for education purposes.

7.2 Model Predictive Control

In this study we consider MPC optimization problem of the form

$$\min_{u_0,\ldots,u_{N-1}} \sum_{k=0}^{N-1} (||Q_{\mathrm{y}}\,(y_k - y_{\mathrm{ref}})\,||_2^2 + ||Q_{\mathrm{u}}\,(u_k)\,||_2^2 + ||Q_{\mathrm{du}}\,(\Delta u_k)\,||_2^2 + ||Q_{\mathrm{s}}\,(s_k)\,||_2^2), \tag{4a}$$

$$\text{s.t. } x_{k+1} = Ax_k + Bu_k, \tag{4b}$$

$$y_k = Cx_k, \tag{4c}$$

$$x_0 = x(t) \tag{4d}$$

$$\Delta u_k = u_k - u_{k-1}, \tag{4e}$$

$$u_{\min} \le u_k \le u_{\max}, \tag{4f}$$

$$y_{\min} - s_k \le y_k \le y_{\max} + s_k, \tag{4g}$$

$$s_k \ge 0, \tag{4h}$$

where constraints (4b)−(4h) are enforced for $k \in \{1,\ldots,N-1\}$. In the objective function (4a) N denotes the prediction horizon, Q_{y}, Q_{u}, Q_{du} and Q_{s} are the weighting matrices and term e.g. $||Q_{\mathrm{u}}\,(u_k)\,||_2^2 = u^T Q_{\mathrm{u}} u$ represents the squared Euclidean norm. Next $x(t)$ is the state measurement, x_k is the vector of predicted states, y_k is the vector of predicted outputs, u_k is the vector of optimized control inputs and A, B and C are the state-space matrices defined in (3). Note that the notation distinguish between the real-time values $x(t)$ and their predictions x_k at the k-th step. Furthermore, Δu_k is the difference between two subsequent control inputs (4e), s_k are slack variables and $u_{\min}$, $u_{\max}$, $y_{\min}$ and $y_{\max}$ are input and output boundaries.

The optimization problem defined as in (4) is chosen to illustrate the basic advantages of MPC strategy that are prohibitive in other conventional control approaches such as PID or LQR. The basic MPC formulation, which is denoted by constraints (4b)−(4d) and first two terms in the objective function (4a), basically represents to the LQR policy with finite prediction horizon. By adding hard constraints (4f) we force the optimization problem (4) to operate only with, e.g. physically applicable, control inputs inside of the restricted boundaries $u_{\min}$ and $u_{\max}$. Another advanced feature of MPC embedded in (4) is the delta u formulation represented by (4e) and the third term in the objective function (4a), which provides offset-free tracking, however only if the mathematical model matches the controlled plant (system of cars). Finally, the output constraints (4g) restrict distances between each car. We need to keep in mind that this shrinks the domain of the MPC policy. Therefore if a larger disturbance of the measured outputs occurs, what is quite common with ultrasonic sensors, the optimization problem become infeasible and no control input would be sent to cars. This limitation can be easily handled by adding slack variables (4h) that are heavily penalized in the objective function. Note that input constraints (4f) omits slacks. This is due to the fact that, in our study, the upper and the lower boundaries $u_{\min}$ and $u_{\max}$ represent physical limitations that can not by exceeded.

The student objective in this project is to devise the MPC optimization problem (4) in MATLAB/Simulink environment. The most fundamental approach is to exploit basic MATLAB functions in computationally efficient manner. This can be, however, quite time consuming and thus we allow one to employ advanced toolboxes like YALMIP (Löfberg, 2004) or MPT3 (Herceg et al., 2013) that allows students to formulate optimization problems (as in (4)) by means of several simple commands. The algorithmic burden is therefore mitigated and students can more focus on control performance or enhancing the MPC formulation by additional advanced approaches such as trajectory preview or move-blocking. Moreover, via the reduced math and algorithmic requirements, MPC policy becomes accessible also for wider range of students (e.g. in their bachelor studies).

7.3 Experimental results

Here we illustrate experimental results of student ongoing project, where two case studies are elaborated. The first setup considers only two cars, one *follower* that is placed right behind the *leader*, which speed remains constant. The control objective is to manipulate speed of the *follower* such that distance from the *leader* tracks the specified reference. This scenario serves as a preparation for a distributed control scheme, where such control has to be embedded into each car on the road to ensure autonomous control. The second approach focuses to control system of four cars, composed of one *leader* and three *followers*, as one centralized control scheme. Here we employ external computation hardware to communicate with all cars on the road. Its objective is to acquire data of distances between all cars and based on this information compute optimal speed of cars to achieve safe and fluent traffic control.

To proceed, we have constructed mathematical models as in (3) for $n = 2$ and $n = 4$, respectively. The sampling time was set to $Ts = 0.1$. Subsequently, the MPC policy as in (4) was devised for both approaches with parameters $N = 30$, $v_0 = 70$, weighting matrices $Q_{\mathrm{y}} = 750$, $Q_{\mathrm{u}} = 1$, $Q_{\mathrm{du}} = 1$, $Q_{\mathrm{s}} = 10^4$, input constraints $u_{\min} = 0$, $u_{\max} = 127$, and output constraints $y_{\min} = 5$, $y_{\max} = 15$. The experiment was performed for 600 of sampling periods, what corresponds to the duration of 60 seconds. The reference $y_{\mathrm{ref}} \in \{8, 12\}$ was changing after each 200 sampling instances. Measured data are reported in Fig. 3 for the first scenario and in Fig. 4 for the second one.

In both scenarios the MPC policy reports good tracking property, while respecting input limitations. On the other

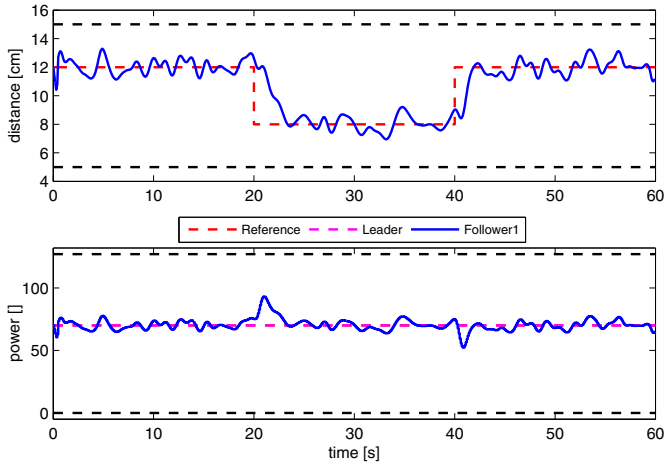

Fig. 3. Control performance of the two car system. The blue line represents output and input profiles of the *follower*, red dashed line is the distance reference, magenta dashed line corresponds to the constant speed of the *leader* and black dashed lines are output and input boundaries.

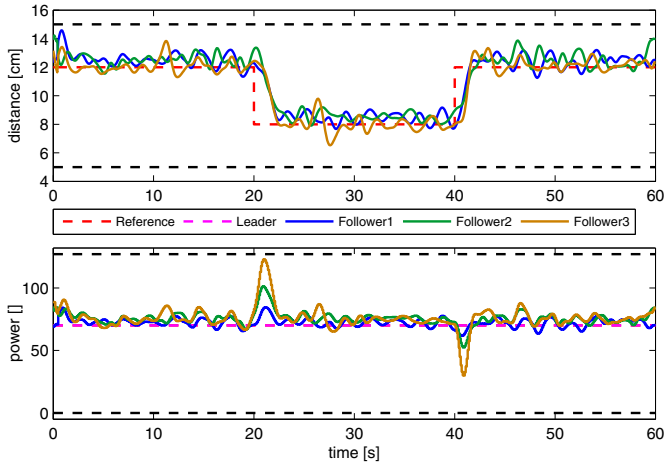

Fig. 4. Centralized system of four cars. The blue, green and brown lines represent output and input profiles of a corresponding *follower*. Red dashed line denotes the reference separation gap between individual vehicles. The magenta dashed line is the *leader's* speed. Finally, black dashed lines represent output and input constraints, respectively.

hand, as can be seen in output profiles, the soft boundaries are violated. These overlaps are, however, mitigated due to the large penalization on slack variables. Another interesting observation can be made in input profiles of Fig. 4 where one can see that the optimal response to the reference step is instantaneous acceleration of all three *followers* at the same time and not one by one as it is common in the real traffic.

8. CONCLUSIONS AND FUTURE WORK

In this paper we have shown an importance of practical experimentation in the control education using the real laboratory equipment. Students are provided with the traffic simulator, where they can apply their theoretical knowledge in the range begging with simple logical and time-continuous controllers up to advanced optimal control methods such as the MPC. As the case study has shown, the simulator can be operated by student-designed MPC controllers even with a very simple formulation that is taught in the basics of the course *Model Predictive Control*. Moreover, the developed traffic simulator provides wide capabilities for future students' works in standard courses focused in automatic control as well as their individual projects and thesis works.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge the contribution of the Scientific Grant Agency of the Slovak Republic under the grant 1/0403/15. The first author acknowledges the financial support by an internal STU grant 1379. The second author acknowledges the financial support by an internal STU grant 1605.

REFERENCES

Deo, P., De Schutter, B., and Hegyi, A. (2009). Model Predictive Control for Multi-Class Traffic Flows. In *12th IFAC Symposium on Control in Transportation Systems (2009)*, 25–30. doi:10.3182/20090902-3-US-2007.00005.

Dezani, H., Gomes, L., Damiani, F., and Marranghello, N. (2012). Controlling traffic jams on urban roads modeled in coloured petri net using genetic algorithm. In *IECON 2012 - 38th Annual Conference on IEEE Industrial Electronics Society*, 3043–3048. doi:10.1109/IECON.2012.6389412.

Garrido-Jurado, S., Muñoz Salinas, R., Madrid-Cuevas, F., and Marín-Jiménez, M. (2014). Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6), 2280 – 2292. doi:http://dx.doi.org/10.1016/j.patcog.2014.01.005.

Herceg, M., Kvasnica, M., Jones, C., and Morari, M. (2013). Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, 502–510. Zürich, Switzerland. `http://control.ee.ethz.ch/~mpt`.

Kesting, A., Treiber, M., Schnhof, M., and Helbing, D. (2008). Adaptive cruise control design for active congestion avoidance. *Transportation Research Part C: Emerging Technologies*, 16(6), 668 – 683. doi:http://dx.doi.org/10.1016/j.trc.2007.12.004.

Löfberg, J. (2004). YALMIP : A Toolbox for Modeling and Optimization in MATLAB. In *Proc. of the CACSD Conference*. Taipei, Taiwan. Available from `http://users.isy.liu.se/johanl/yalmip/`.

Marcolino, L. and Chaimowicz, L. (2009). Traffic control for a swarm of robots: Avoiding target congestion. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, 1955–1961. doi:10.1109/IROS.2009.5354407.

Martinec, D., Šebek, M., and Hurák, Z. (2012). Vehicular platooning experiments with racing slot cars. In *Control Applications (CCA), 2012 IEEE International Conference on*, 166–171. doi:10.1109/CCA.2012.6402709.

Mather, T. and Hsieh, M. (2012). Ensemble modeling and control for congestion management in automated warehouses. In *Automation Science and Engineering (CASE), 2012 IEEE International Conference on*, 390–395. doi:10.1109/CoASE.2012.6386498.

Šebek, M. and Hurák, Z. (2011). 2-D Polynomial Approach to Control of Leader Following Vehicular Platoons. In *18th World Congress of the International Federation of Automatic Control (IFAC)*. Milano, Italy.

Zegeye, S., De Schutter, B., Hellendoorn, H., and Breunesse, E. (2009). Reduction of travel times and traffic emissions using model predictive control. In *American Control Conference, 2009. ACC '09.*, 5392–5397. doi:10.1109/ACC.2009.5159942.