HAL
archives-ouvertes.fr

Stochastic optimization of a chain sliding mode controller for the mobile robot maneuvering

Alexander Terekhov, Jean-Baptiste Mouret, Christophe Grand

▶ **To cite this version:**

Alexander Terekhov, Jean-Baptiste Mouret, Christophe Grand. Stochastic optimization of a chain sliding mode controller for the mobile robot maneuvering. International Conference on Intelligent Robots and Systems (IROS), 2011, San Francisco, United States. pp.4360-4365, 10.1109/IROS.2011.6094956 . hal-00687630

HAL Id: hal-00687630
https://hal.archives-ouvertes.fr/hal-00687630

Submitted on 13 Apr 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Stochastic optimization of a chain sliding mode controller for the mobile robot maneuvering

Alexander V. Terekhov, Jean-Baptiste Mouret, Christophe Grand

Abstract— **In this study we present a chain sliding mode controller for the control of a four wheeled autonomous mobile robot performing aggressive turning maneuver to 90 degrees on a loose surface. The controller consists of a set of local sliding mode controllers and the hyperplanes of switching between them. The parameters of the sliding mode controllers and the hyperplanes are obtained using methods of multiobjective stochastic optimization applied to a model of the robot. The obtained controller is used to drive the mobile robot. The results show that is capable to control the robot during aggressive maneuver. In particular, the steering radius obtained with the controller was two times smaller then the minimal steering radius admitted by the robot.**

I. INTRODUCTION

As the results of DARPA Grand Challenge clearly show, visual systems for unmanned vehicles make significant steps forward [1], hence increasing the average speed of autonomous vehicles. However, the path tracking algorithms do not evolve significantly. It may be argued that further increases of the vehicle speed will be delayed by the inability of the path-tracking algorithms to cope with the task, especially during the drift.

With significant progress in the quality of the sensors and perception algorithms the necessity comes to develop the controllers of the vehicle, capable to operate at high movement speeds. Most of the current day controllers are essentially based on the virtual vehicle algorithm [2]. In particular, the vehicle is usually modeled as a non-holonomic system, thus excluding the possibility of the drift and inevitably limiting its maneuverability. Some efforts has been made to enhance the performance of the virtual target algorithm using a sliding mode controller [3] by taking the slippage into account.

The general goal of the developing the new controllers would be to achieve the performance comparable to professional drivers. One of the promising approach is based on recording the actions of human drivers and their fusion with virtual vehicle-like algorithms. Impressive results were recently reported for autonomous aggressive parking [4]. However, this approach is only limited to the autonomous cars. In case of different vehicle mechanics (for example, independently control motor wheels) the experience of the rally racers cannot be directly exploited.

This work was supported by DGA, grant REI 2008.34.0018

The authors are with Institut des Systèmes Intelligents et de Robotique, UPMC-CNRS, 4 Place Jussieu, 75005 Paris, France.

A.V. Terekhov: avterekhov@gmail.com

J.-B. Mouret: mouret@isir.umpc.fr

C. Grand: christophe.grand@isir.upmc.fr

Another way to approach the problem would be to use the classical scheme, used, for example for the ballistic rockets control. Namely, to split the control problem into the combination of finding the optimal trajectory (using dynamical model of the vehicle) and its stabilization it with a presumably linear feedback system. Recently, we made few steps in this direction for a four-wheels mobile robot performing highly aggressive turning maneuver on loose surface [5], [6]. We discovered that the use of stochastic optimization can be very helpful in designing the optimal trajectory and feedforward control inputs, but the problem of the feedback stabilization may be difficult. The difficulties come from the fact that during the aggressive turning maneuver there is significant lateral component in the vehicle's motion, while the vehicle is locally non-controllable in this direction. It means that from the local view point, the lateral movement of the vehicle is ballistic, i.e. it is more like the flight of a bullet, for which only initial speed and direction can be controlled. It follows that if the stabilization is at all possible it would probably require the feedback law, which would be non-linear or time-dependent, or both. Our attempts to use multilayer perceptron to approximate this law resulted in control laws, which are hardly feasible to be implemented in a real robot [6]

In the current paper we consider the problem of the control of a four-wheeled mobile robot performing aggressive 90 degrees turn maneuver on a loose surface. We aim at finding the controller, which would demonstrate the performance, comparable to professional rally drivers. Based on our past experience [5], [6] we search for a way to find the controller that would combine optimal maneuver execution in feedforward manner with feedback mechanisms, stabilizing the execution.

A. Motivation

The ideal controller would follow the optimal trajectory from any point of the state space. For such controller any disturbance would be just a shift of the initial conditions, while the movement would always remain optimal. However, such controllers are possible to build only for the simple systems, which can be solved analytically. This is definitely not the case for the vehicle maneuver control, where the wheel-road interaction can be described by a fast changing significantly non-linear function of the vehicle state. One might think of finding the solutions computationally for all possible initial conditions and putting them in place on demand. However, it is hardly feasible because of the high

dimension of the state space and the sensitivity of the wheel-road interaction model.

Still for some classes of the optimal control problem finding the optimal control law for every point of the state space seems feasible. Thus, for example, it is known that under some conditions the solution of the problem of the movement time minimization has a bang-bang structure, meaning that it switches between the maximum and minimum admissible values of the control inputs. The switching occurs when the state of the controlled object crosses a specific hypersurface in the state space. For such systems the problem of the synthesis of the optimal controller is limited to finding the switching hypersurfaces. The latter is still not so simple because they may be significantly curved. The form of the optimal switching hypersurface is determined by the particular dynamics of the controlled object. For the general optimal control problems the situation is even worse because the control inputs may take intermediate values.

Thus, the problem of the optimal control synthesis seems hardly solvable. However, for our specific purpose it is not necessary to know the optimal solutions for every point of the state space. It may be sufficient to have them only inside a domain of the state space surrounding the trajectory of the maneuver and possible deviations from its execution. In this case the switching surfaces may be approximated by hyperplanes and the change of the control inputs may be approximated by the piecewise linear functions. It must be noted that the piecewise approximation of the optimal control inputs was shown to be able to fit the actions of the professional rally racers [7]. Moreover, the hyperplane-based approximations of the switching hypersurfaces are vastly used in the sliding mode control.

B. Chain sliding mode controller

The chain sliding mode controller is a set of sliding mode controllers involved one after another. The idea is to have different sliding mode controllers in different parts of the state space and to switch between them when the object's state crosses a switching hyperplane.

Let ξ be the vector of the observations, appended with a unit value:

$$\xi = \begin{pmatrix} 1 \\ y \end{pmatrix},$$

where y is the vector of the observations.

As the local controllers we use the second order sliding mode controllers, whose advantages are described in [8]. Every local controller is given by the switching surface, which can be defined by a vector η of the same dimension as ξ. The rate of change of the control input u is given by formula:

$$\dot{u} = k \operatorname{sign}(\eta, \xi), \tag{1}$$

where k is a constant value, limiting the rate of change of the control values; the brackets denote the scalar product.

Unlike classically used sliding mode controllers, the formula (1) has integration. As a consequence the control value u is a piecewise linear function, instead of piecewise constant. Moreover, the sliding surface is shifted from the origin, since the first element of ξ always equals one. Of course, in addition to (1) the values u must be bounded be their maximum and minimum admissible values: $u_{\min} \leq u \leq u_{\max}$.

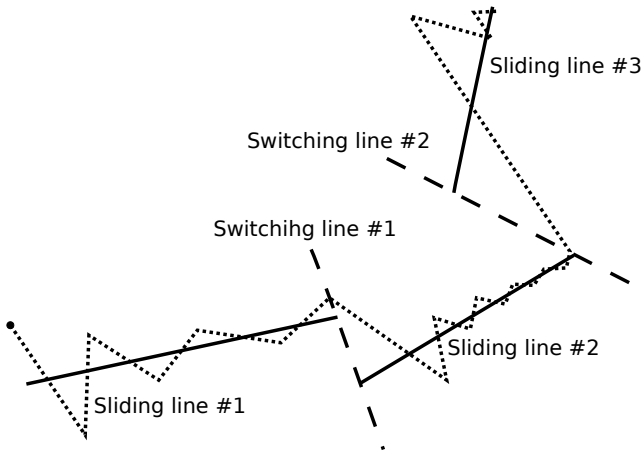

Fig. 1. Schematic representation of the chain sliding mode controller. The trajectory of the object state is denoted with the dotted line. The initial position is shown with the black dot.

The formula (1) defines a local controller. The global controller is defined by a set of local controllers, which are interchanged when the point ξ crosses controller-switching hyperplanes, which we call "switching hyperplane". The switching hyperplanes can also be defined by its normal vector σ. The switching happens when the value

$$s = (\sigma, \xi) \tag{2}$$

changes its sign.

The whole control scheme is illustrated in Fig. 1. Initially the robot is controlled by the controller, defined by the vector η_1 and the coefficient k_1. When the point ξ crosses the hyperplane (line in the figure), defined by the vector σ_1, the scalar product (σ_1, ξ) changes its sign. As soon as it happens the control vector η_1 and value k_1 are substituted by the new control vector η_2 and the new control value k_2, which are used until the hyperplane, defined by σ_2, is crossed. The switching of the controllers is usually accompanied by the jump of the control parameters, resulting from the fact that the sliding mode controllers are not aligned at the switching point.

In this paper we use the methods of stochastic optimization in order to find a chain sliding mode controller to govern an aggressive 90 degrees turning maneuver of a four-wheeled mobile robot on a loose surface. We build a relatively simple dynamical model of the robot for tuning the controller and analyze its robustness based on a more complicated dynamical model. In supplementary materials we present a video of preliminary results of the robot control.

Fig. 2. Mobile robot "fastBot 2", for which the controller is designed.

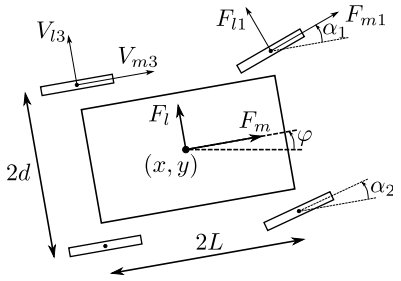

Fig. 3. Schematic representation of the robot.

II. MATHEMATICAL MODEL

A. Short robot description

The controller is designed for the manually designed four wheeled mobile robot "fastBot 2", shown in Fig. 2. The description of the previous version of the platform can be found in [9].

Shortly, the robot represents a four-wheeled mobile platform, whose approximate mass geometrical characteristics are given in table I. The axles of the robot are equipped with differentials. The front axle admits Ackerman steering of the wheels. The breaking system is installed on the rear axle. When the breaks are on the axle rotation is excluded, while the differential admits the rotation of the wheels can rotate in opposite directions.

The propulsion of the robot is performed by the brushless motor, whose torque is transmitted to the front axle through a gear set. The motor is controlled by a servo amplifier, using the output of the encoder, installed on the motor shaft. The servo amplifier works in velocity tracking mode. The steering and the breaks are controlled by servomotors, working in position tracking mode.

B. Mathematical model

The schematic representation of the simplified model of the robot is given in Fig. 3. The position of the robot is defined by the location of its center of mass x, y and the heading angle φ between the x axis and the longitude axis of the robot. The motion of the robot satisfies the equations:

$$\begin{aligned} \dot{x} &= V_m \cos\varphi - V_l \sin\varphi, \\ \dot{y} &= V_m \sin\varphi + V_l \cos\varphi, \\ \dot{\varphi} &= \omega, \\ J\dot{\omega} &= T, \\ M\dot{V}_m &= F_m + M\omega V_l, \\ M\dot{V}_l &= F_l - M\omega V_m, \end{aligned} \tag{3}$$

where ω is the angular velocity of the robot's trunk, V_m and V_l are the projections of the linear velocity of the robot's center of mass on the medial (longitude) and lateral axes respectively, F_m, F_l and T are the total medial and lateral forces and the total torsion torque, defined as the following:

$$\begin{aligned} F_m &= F_{m1}\cos\alpha_1 - F_{l1}\sin\alpha_1 + F_{m2}\cos\alpha_2 - \\ &\quad F_{l2}\sin\alpha_2 + F_{m3} + F_{m4}, \\ F_l &= F_{l1}\cos\alpha_1 + F_{m1}\sin\alpha_1 + F_{l2}\cos\alpha_2 + \\ &\quad F_{m2}\sin\alpha_2 + F_{l3} + F_{l4}, \\ T &= L(-F_{m1}\cos\alpha_1 + F_{l1}\sin\alpha_1 + F_{m2}\cos\alpha_2 - \\ &\quad F_{l2}\sin\alpha_2 - F_{m3} + F_{m4}) + \\ &\quad d(F_{l1}\cos\alpha_1 + F_{m1}\sin\alpha_1 + F_{l2}\cos\alpha_2 + \\ &\quad F_{m2}\sin\alpha_2 - F_{l3} - F_{l4}), \end{aligned}$$

F_{li}, F_{mi} are lateral and medial projections of the tangential forces of wheel-road interaction for each wheel reference frame (see Fig. 3), α_1, α_2 are the steering angles of the left and right wheels, respectively.

For the forces F_{mi} and F_{li} we use the brush model, which is relatively simple computationally, but at the same time captures the main features of the wheel-road interaction. The details of the brush model can be found in [10]. Roughly, the medial and lateral tangential forces are defined as nonlinear function of the lateral and medial projections of the slippage velocity, V_{smi} and V_{sli}, that is the velocity of the contact point of the wheel (in case of no sliding this velocity is zero):

$$\begin{aligned} F_{mi} &= \mu F_{ni} f(V_{smi}/V_i), \\ F_{li} &= \mu F_{ni} f(V_{sli}/V_i), \end{aligned} \tag{4}$$

where V_i is absolute value of the velocity of the axle of i-th wheel, F_{ni} is the normal force at the i-th wheel contact point and μ is the coefficient of Coulomb friction. The function f depends on the tangential stiffness of the tire c_p.

To determine the normal forces in (4) we used the method described in [11]. The resultant normal forces for every wheel is given by the equations:

$$\begin{aligned} F_{n1} &= \frac{M}{4dL}(dLg - hLa_l - hda_m) \\ F_{n2} &= \frac{M}{4dL}(dLg + hLa_l - hda_m) \\ F_{n3} &= \frac{M}{4dL}(dLg - hLa_l + hda_m) \\ F_{n4} &= \frac{M}{4dL}(dLg + hLa_l + hda_m) \end{aligned} \tag{5}$$

where h is the height of the center of mass of the robot, a_m and a_l are projections of the robot's acceleration on the corresponding axes.

The equation (5) describes weight redistribution caused by the acceleration of the center of mass of the robot in case when the pitch and roll angles of the robot are close to zero. However, to compute the medial and lateral acceleration one must provide the total medial and lateral tangential forces, which, according to the brush model, depend on the normal forces themselves. In order to solve this problem we made an assumption that the weight redistribution (5) does not happen instantly but with a characteristic time τ, which roughly correspond to the characteristic time of the suspension system of the robot. We appended the dynamic equations of the robot (3) with the following:

$$\begin{aligned} \tau \dot{a}_l &= F_l/M - a_l, \\ \tau \dot{a}_m &= F_m/M - a_m. \end{aligned} \tag{6}$$

To determine the velocities in (4) we need to have the angular velocities of the wheels, ω_i. We represent of the individual wheels velocities as a superposition of the axle angular velocities ω_f, ω_r and the difference between the left and write wheel velocities at each axles $\delta\omega_f$, $\delta\omega_r$ allowed by the differential:

$$\begin{aligned} \omega_1 &= \omega_f + \delta\omega_f, & \omega_2 &= \omega_f - \delta\omega_f, \\ \omega_3 &= \omega_b + \delta\omega_b, & \omega_4 &= \omega_b - \delta\omega_b. \end{aligned} \tag{7}$$

We assume that the servo amplified tracks velocity ideally the desired angular velocity of the front wheels and thus ω_f becomes a control input. The angular velocity of the rear axle ω_r is assumed to appear from the interaction with the ground reaction forces and the breaking toque T_b:

$$J_w \dot{\omega}_r + \nu_w \omega_r = -\frac{r_w}{2}\left(F_{m3} + F_{m4}\right) + T_b, \tag{8}$$

where J_w is the moment of inertia of the wheel and ν_w is the friction in the axle. For the breaking torque we used dynamic friction model instead of the Coulomb friction model, which would be more appropriate in this case. We preferred the first one because for the given set of parameters the performance of two models is nearly the same, while the first one suits better the numeric simulation.

$$T_b = \begin{cases} -\nu_b \omega_r & \text{breaks on,} \\ 0 & \text{breaks off.} \end{cases} \tag{9}$$

The coefficient of the breaks friction ν_b was chosen sufficiently high, but not threatening the stability of the system.

Finally, we assume that the distribution of the wheels forces within the same axle is determined by the ground interaction forces:

$$\begin{aligned} J_w \delta\dot{\omega}_f + \nu_w \delta\omega_f &= -r_w\left(F_{m1} - F_{m2}\right) \\ J_w \delta\dot{\omega}_r + \nu_w \delta\omega_r &= -r_w\left(F_{m3} - F_{m4}\right) \end{aligned} \tag{10}$$

For sake of simplicity we ignore the dynamics of the steering system, assuming that it the servomotor tracks ideally the desired steering angle, denoted as α. The steering angles of each of the wheels were computed from Ackerman formulas:

$$\begin{aligned} \alpha_1 &= \arctan \frac{\sin\alpha}{1 - \frac{d}{2L}\sin\alpha}, \\ \alpha_2 &= \arctan \frac{\sin\alpha}{1 + \frac{d}{2L}\sin\alpha}. \end{aligned} \tag{11}$$

On the whole the state of the system is 11-dimensional. The system admits 3 control inputs: the desired angular velocity of the front wheels ω_f, the desired steering angle α and the state of the breaking system b.

The parameters of the model used in the simulations are presented in Tab. I. These parameters correspond approximately to those of the mobile robot "fastBot 2" shown in Fig. 2.

TABLE I
PARAMETERS OF THE MODEL USED IN SIMULATION.

M	$7.5\,\mathrm{kg}$	J	$0.1\,\mathrm{kg\,m^2}$
L	$0.20\,\mathrm{m}$	d	$0.175\,\mathrm{m}$
h	$0.06\,\mathrm{m}$	τ	$0.05\,\mathrm{s}$
μ	0.6	c_p	$10^5\,\mathrm{N/m^2}$
ν_w	$0.01\,\mathrm{N\,m/s}$	ν_b	$0.1\,\mathrm{N\,m/s}$
J_w	$0.0003\,\mathrm{kg\,m^2}$	r	$0.075\,\mathrm{m}$

III. ESTIMATION OF THE CONTROLLER PARAMETERS

We search for the optimal parameters of the presented chain sliding mode controller to drive the robot making 90 degrees turn. We assume that the desired path is composed of two orthogonal straight lines. The robot is located 6 m in front of the turn, directed towards it and has initial velocity of 3 m/s. The robot is then let to move for 4 seconds, during which the steering angle, the front axle velocity and the breaks command are controlled. The range of steering angle change is fixed to ± 25 degrees, the linear velocities of the wheels are allowed to vary between 0.1 and 3 m/s, the command to breaks was between 0 and 1. The breaks were off if it was below 0.5 and they were on otherwise.

A. Controller

The controller assigns the steering angle, front wheels velocity and the breaks command based on the observations. We assume that the following information is available to the robot:

1) s – the distance to the turn (negative before the turn and positive after);
2) V – the magnitude of the velocity of the robot's center of mass;
3) ω – the angular velocity of the robot's trunk;
4) δ – the signed distance from the trajectory to the robot's center of mass (positive when deviates left);
5) ψ – the signed angle between the heading direction of the robot and the tangent to the trajectory (positive for left turn).

The corresponding appended vector of the observations is

$$\xi = (1, s, V, \omega, \delta, \psi)^T .$$

The controller with N local controllers is defined by the set of the sliding mode vectors η_i with the corresponding coefficients k_i and by the switching surfaces vectors σ_j, $i = 1, \ldots, N$, $j = 1, \ldots, N-1$. The dimension of each of the vectors η_i and σ_j is equal to 6. On the whole, the controller is parametrized by $7N - 3$ scalar values. This number corresponds to the controller of a single value, i.e. the steering angle or the front wheels velocity, or the breaks command.

In order to simplify the search of the control parameters we assumed that before and after maneuver the robot was controlled by an algorithm, resembling the virtual vehicle controller. In other words, the first and the last sliding mode vectors for the steering angle η^α and the front wheels velocity η^V were set to predefined values:

$$\eta_1^\alpha = (0, 0, 0, 0.5, 0.5, 1)^T$$

and

$$\eta_1^V = (3, 0, -1, 0, 0, 0)^T .$$

The corresponding control laws are the following:

$$\dot{\alpha} = -k_1^\alpha \operatorname{sign}\left(\frac{1}{2}\omega + \frac{1}{2}\delta + \psi\right),$$

$$\dot{V}_F = -k_1^V \operatorname{sign}(V - 3) .$$

The breaks were set to "off":

$$\eta_1^b = (-1, 0, 0, 0, 0, 0)^T .$$

The first controller is the classical path-tracking feedback with the exception that it uses integrated values. The second controller stabilizes the velocity of 3 m/s, the third one brings the breaks command to "off".

In the current paper we restricted the number of the local controllers to 3 per the control value, which results in 33 unspecified parameters. In addition to that we ran optimization for 5 local controllers per control value but we discovered that the number of controllers along the trajectory never exceeded 3 for each value. It means that the hypersurfaces of the controllers switching were such that they were never crossed by the state of the robot.

B. Stochastic optimization

We search for the controller parameters that allow the robot to perform the maneuver as fast as possible with the minimal deviation from the desired path. Instead of arbitrarily aggregating these two objectives into a single function to optimize, we rely on a stochastic multiobjective optimization algorithm that search for the set of *Pareto-optimal trade-offs*, that is solutions that cannot be improved with respect to one objective without decreasing their score with respect to the other one.

Numerous algorithms have been proposed in the literature to optimize several objectives in this fashion [12]; most of them rely on the concept of Pareto dominance, defined as follows: a solution p^* is said to dominate another solution p, if two following conditions are satisfied: (i) the solution p^* is not worse than p with respect to all objectives, (ii) the solution p^* is strictly better than p with respect to at least one objective. Typical algorithms starts with a set of M random points, called a population, and evaluate the objectives for each point; these points are then sorted according to Pareto dominance; the best points are kept and they are perturbed (e.g. by adding a Gaussian noise) to generate new points that will replace the worst ones; the objective are evaluated for each new point and the sorting/perturbation cycle starts again.

We here used the algorithm NSGA-II [13]—which follows the previously described algorithm outline—implemented in the Sferes_v2 framework [14]. We performed 10,000 iterations with a population size of 300.

C. Disturbances

In absence of external disturbances and for the fixed reference trajectory the optimal control can be learned in purely feedforward manner. In order to avoid this sort of over-learning of the control parameters we introduce model disturbances to the system. In addition to the clean simulation we ran the controller under disturbed conditions. The following changes of the simulation parameters were used as disturbance:

1) $V_{\max} = 9\,\mathrm{m/s}$,
2) $V_{\max} = 11\,\mathrm{m/s}$,
3) $\mu = 0.55$,
4) $\mu = 0.65$,
5) $J = 2.5\,\mathrm{kg}\cdot\mathrm{m}^2$ and $M = 30\,\mathrm{kg}$,
6) $J = 3.5\,\mathrm{kg}\cdot\mathrm{m}^2$ and $M = 50\,\mathrm{kg}$,

The accuracy and the average speed of the robot were computed as the worst case over all disturbances, i.e, the maximum deviation from the path and the lowest average speed were taken as the performance characteristics of the given controller.

D. Optimization results

The outcome of the optimization algorithm is an approximation of the Pareto front for the two objective functions: average speed of maneuver and its accuracy (the worst cases over 6 disturbances). In this case all solutions were approximately the same, so we selected the one with the best accuracy.

The trajectory of the selected solution is given in Fig. 4. It can be seen that the controller is able to track the desired trajectory with rather high precision (the maximum deviation is below 6 cm). In the simulations the maneuver is performed with significant understeering: the robot nearly stops close to the turn apex, while slipping slightly in lateral direction.

The the velocity and the breaks control inputs are shown in Fig. 5. It can be seen that the robots decelerate and breaks at the same time, while approaching the turn apex. Short before the turn it starts accelerating, but keeps the breaks "on". At the same time instance it starts steering (see Fig. 6).

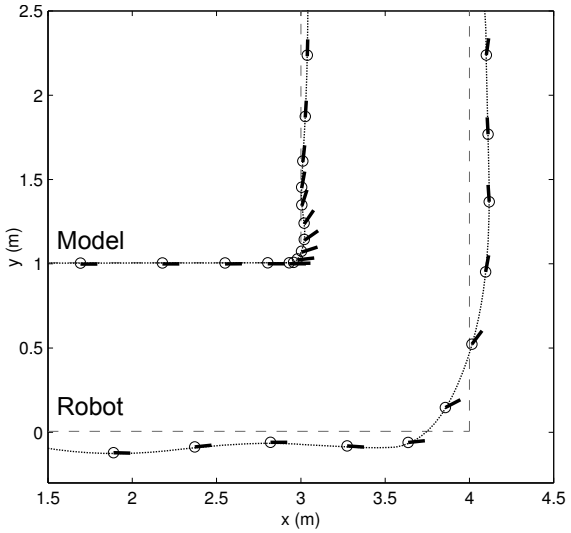

Fig. 4. The trajectories of the model and the robot, controlled by the same algorithm. In both cases the dashed line denotes the reference trajectory.

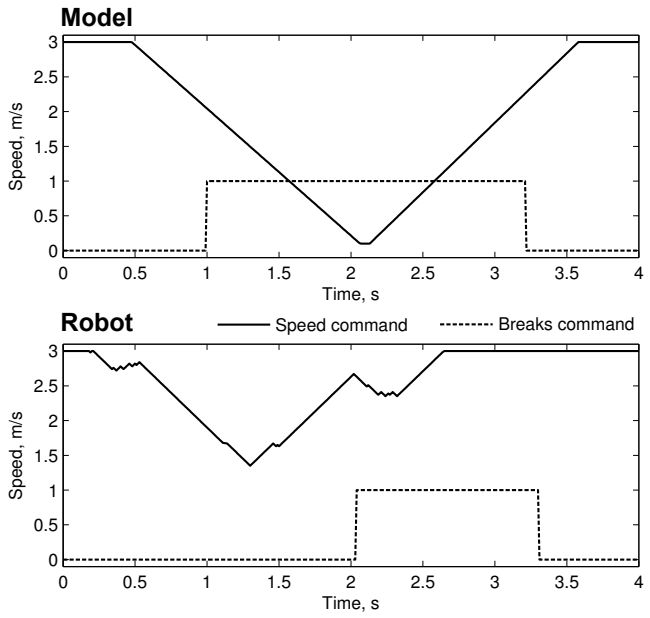

Fig. 5. The velocity and breaks commands of the same controller executed on the model and on the robot. The turn apex occurs at 2.5 sec in both cases.

The steering angle controller has 3 clearly marked phases: the first and the last one coinciding with regular path-tracking algorithm and the middle one, responsible for the steering during maneuver.

In [6] for the same task we developed a controller combining feedforward control inputs with the feedback based on a multilayer perceptron. The resultant control signals were rather ragged and unlikely to work on a real robot. In contrast, the controller presented here, produced signals resembling a lot the feedforward controller we used in our previous study. Moreover, the control commands are qualitatively similar to those reported for professional rally racers [7].

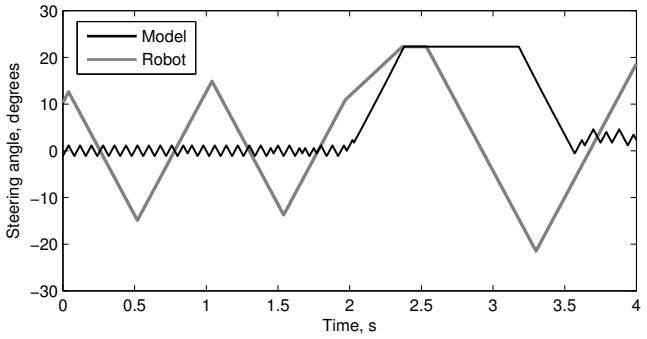

Fig. 6. The steering outputs of the same controller executed on the model and on the robot. The turn apex occurs at 2.5 sec in both cases.

E. Preliminary experiments with robot

The same controller was executed on the real robot. The tests were performed in the university campus on the stone floor. In order to make the contact more slippy we attached plastic tape on the wheels of the robot. A representative example of the robot movement is given in Supplementary Materials.

The obtained trajectory is shown in Fig. 4. Clearly, in the real robot the controller demonstrates much worse performance than in simulations. Moreover, as it can be seen from Fig. 5 and Fig. 6, the same controller produced significantly different commands when applied to the model and to the real robot. In reality the real robot decelerated significantly less when approaching the turn than it did in the model. It might happen because in the real robot the initial speed was slightly above 2 m/s instead of 3 m/s used in simulations. For the slippery wheels the robot had higher speeds were impossible to achieve.

Yet, few details must be noted, which make us enthusiastic about the controller. First of all, the robot actually performed the maneuver. Moreover, under no-slippage condition the robot's steering system admits only turns with the steering radius of 1 m and above. In contrast, in the experiment the robot turned with about 0.5 m steering radius benefiting significantly from the slippage and the breaking system. Both in the simulation and in the experiment the maneuver was performed with significant understeering. The slippage angle was about 35 degrees in the experiment and close to 70 degrees in the simulations.

To be sure that the observed performance achieved due to the proper controller we tried to find the parameters manually on the real robot starting from the scratch, however we did not succeed. Overall, we believe that the proposed control scheme might be used for the robot control.

IV. CONCLUSION

In this paper we presented the chain sliding mode controller, consisting of local sliding mode controllers and the hyperplanes of switching between them. We applied the controllers of this structure to the problem of aggressive maneuver execution by a four-wheeled mobile robot. The maneuver consisted in 90 degrees turn at velocity about

3 m/s. The parameters of the controller were determined using the methods of stochastic multiobjective optimization. The obtained controller produced reasonable commands in the simulations and when working on the real robot.

REFERENCES

[1] S. Thrun *et al.*, "Stanley: The robot that won the darpa grand challenge: Research articles," *J. Robot. Syst.*, vol. 23, no. 9, pp. 661–692, 2006.

[2] M. Bibuli, G. Bruzzone, M. Caccia, and L. Lapierre, "Path-following algorithms and experiments for an unmanned surface vehicle," *J. Field Robot.*, vol. 26, no. 8, pp. 669–688, 2009.

[3] E. Lucet, C. Grand, D. Salle, and P. Bidaud, "Dynamic yaw and velocity control of the 6wd skid-steering mobile robot roburoc6 using sliding mode technique," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pp. 4220–4225, May 2008.

[4] Z. Kolter, C. Plagemann, D. T. Jackson, A. Ng, and S. Thrun, "A probabilistic approach to mixed open-loop and closed-loop control, with application to extreme autonomous driving," in *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, (Anchorage, Alaska, USA), 2010.

[5] A. V. Terekov, J.-B. Mouret, and C. Grand, "Stochastic multi-objective optimization for aggressive maneuver trajectory planning on loose surface," in *Proc. of the IFAC conference on Intelligent and Autonomous Vehicles*, (Lecce, Italy), 2010.

[6] A. V. Terekov, J.-B. Mouret, and C. Grand, "Stochastic optimization of a neural network-based controller for aggressive maneuvers on loose surfaces," in *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems (IROS)*, (Taipei, Taiwan), 2010.

[7] E. Velenis, P. Tsiotras, and J. Lu, "Aggressive maneuvers on loose surfaces: Data analysis and input parametrization," in *15th IEEE Mediterranean Control Conference, June 26-29, Athens, Greece*, 2007.

[8] G. Bartolini, A. Ferrara, E. Usai, and V. I. Utkin, "On multi-input chattering-free second-order sliding mode control," *Automatic Control, IEEE Transactions on*, vol. 45, no. 9, pp. 1711–1717, 2000.

[9] E. Lucet, C. Grand, A. Terekhov, and P. Bidaud, "Experimental study of a fast mobile robot performing a drift maneuver," in *Proceedings of Clawar'10: 12th Int. Conf. on Climbing and Walking Robots*, (Nagoya, Japon), 2010.

[10] H. Pacejka, *Tyre and Vehicle Dynamics*. SAE International, Elsevier, 2 ed., 2005.

[11] E. Velenis, P. Tsiotras, and J. Lu, "Modeling aggressive maneuvers on loose surfaces: The cases of trail-braking and pendulum-turn," in *European Control Conference, Kos, Greece, July 2-5*, 2007.

[12] K. Deb, *Multi-objectives optimization using evolutionnary algorithms*. Wiley, 2001.

[13] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan, "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II," in *Proceedings of the Parallel Problem Solving from Nature VI Conference*, pp. 849–858, Springer. Lecture Notes in Computer Science No. 1917, 2000.

[14] J.-B. Mouret and S. Doncieux, "Sferes_v2: Evolvin' in the multi-core world," in *IEEE Congress on Evolutionary Computation, 2010 (CEC 2010)*, 2010.