Stochastic optimization of a neural network-based controller for aggressive maneuvers on loose surfaces

Alexander Terekhov, Jean-Baptiste Mouret, Christophe Grand

Stochastic optimization of a neural network-based controller for aggressive maneuvers on loose surfaces

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

Abstract— **In this study we develop a feedback controller for a four wheeled autonomous mobile robot. The purpose of the controller is to guarantee robust performance of an aggressive maneuver (90 degrees turn) at high velocity (about 10 m/s) on a loose surface (dirty road). To tackle this highly non-linear control problem, we employ multi-objective evolutionary algorithms to explore and optimize the parameters of a neural network-based controller. The obtained controller is shown to be robust with respect to uncertainties of the robot parameters, speed of the maneuver and properties of the ground. The controller is tested using two mathematical models of significantly different complexity and accuracy.**

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. The classical "virtual target" algorithm or its variations are still used in dominant amount of applications [2]. Roughly, in this algorithm the robot steering commands are proportional to the angle between the robot's heading direction and the direction towards a virtual target, moving along the desired trajectories at some distance in front of the robot. The virtual target algorithm has been vastly studied in the framework of nonholonomic models of robots, which do not admit any slippage. 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.

It seems that in order to achieve the control performance comparable to the one exhibited by professional rally drivers, one might have to change the point of view on the problem. Instead of developing universal path tracking algorithms, one could focus on obtaining a library of maneuvers and conditions for their execution. Within such approach the controller of each maneuver may significantly rely on a priori information about the desired path, the properties of the soil, the parameters of the robot/vehicle. Ideally, the maneuvers are approximations of the optimal solution for a particular movement.

Recently attempts were made to build a controller inspired by the performance of the professional rally racers [4]. The actions of professional rally during aggressive turning were recorded and analyzed [4]. They appeared to be quite reproducible: the steering angle and break/throttle commands could be sufficiently well approximated by piecewise linear functions of time. Moreover, similar vehicle trajectories can be obtained as solutions of optimal control problem [4].

The aim of the current paper is to develop these results and to obtain a feedback controller that would allow a mobile robot to perform a maneuver (90 degrees turn) at high speed. We start with finding a feedforward solution for the maneuver, thus partly reproducing the results from [4] for a four-wheeled mobile robot. The obtained solution is then used as a reference for the feedback controller. The feedback controller is searched in the class of feedforward neural networks, with the inputs corresponding to the information that we assume professional rally drivers might possess: distance to turn, vehicle orientation, deviation from the path, linear velocity of the vehicle and its angular velocity. The controller is required to be robust with respect to disturbances of the model and initial conditions.

The search of the solutions for both feedforward and feedback controller rely on multiobjective stochastic optimization. We have to involve this method because both problems are highly non-convex and multiobjective and because traditional methods—when applicable—converge to local minima which are often distant from the global one. The stochastic algorithms are highly demanding in terms of computation; to be able to find a solution within reasonable time, we therefore use a highly simplified model of the robot. The solutions obtained this way are then tested on a more complex and accurate model[1].

II. MATHEMATICAL MODELS

A. Simplified model

The schematic representation of the simplified model of the robot is given in Fig. 1. 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

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

[1]The source code of the models and of the numerical experiments presented in this paper are available at: http://www.isir.upmc.fr/~mouret/evodb/

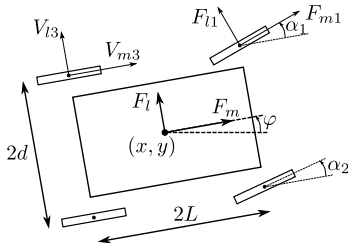

Fig. 1. Schematic representation of the robot.

of the robot. The motion of the robot satisfies the equations:

$$\begin{aligned} \dot{x} &= V_m \cos\varphi - V_l \sin\varphi, \\ \dot{y} &= V_m \sin\varphi + V_l \cos\varphi, \\ \dot{\varphi} &= \omega, \\ J\dot{\omega} &= T, \\ M\dot{V}_m &= F_m + M\omega V_l, \\ M\dot{V}_l &= F_l - M\omega V_m, \end{aligned} \tag{1}$$

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. 1), 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 [5]. 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):

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

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 we used the method described in [6]. The resultant normal forces for i-th wheel is given by the equations:

$$F_{ni} = \frac{M}{4dL}(dLg + hL_i a_l + hd_i a_m), \tag{2}$$

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 (2) 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 (2) 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 (1) with the following:

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

For sake of simplicity, we ignore the dynamics of the wheels and assumed that there is a controller in each wheel, which tracks the desired wheel rotation velocity perfectly. We make the same assumption regarding the functioning of the steering system. Thus, 2 steering angles and 4 wheel rotation velocities represent the control inputs to the model. In order to reduce the dimension of the control inputs we regard 3 independent control inputs: the steering angle α of the front wheels and the linear velocities of the front and rear wheels, V_F, V_R, correspondingly. The steering angles and velocities of the wheels are computed using Ackermann rule:

$$\begin{aligned} R\omega_1 &= V_F\left[\left(1 - \tfrac{d}{2L}\sin\alpha\right)^2 + \sin^2\alpha\right]^{1/2}, \\ R\omega_3 &= V_R\left(1 - \tfrac{d}{2L}\sin\alpha\right), \\ \alpha_1 &= \arctan\tfrac{\sin\alpha}{1 - \frac{d}{2L}\sin\alpha}, \end{aligned} \tag{3}$$

where ω_1, ω_3 are the left front and rear wheel angular velocities; α_1 is the steering angle of the left front wheel; R is the wheel radius. The formulas for the right wheels velocities and the steering angle are the same as (3), with the exception that every "minus" sign is substituted by "plus".

The following values of parameters of the model were used in the simulations: $M = 40\text{kg}$, $J = 3\text{kg}\cdot\text{m}^2$, $L = 0.5\text{m}$, $d = 0.25\text{m}$, $h = 0.1\text{m}$, $\tau = 0.05\text{s}$, $\mu = 0.6$, $c_p = 10^5\text{N/m}^2$. These parameters roughly correspond to those of a light mobile robot, which is being currently developed at our institute. The Coulomb friction coefficient corresponds to the movement on a dirty road.

B. Full model

In addition to the simplified model we use a more complex and accurate model, which we call the "full". In the full model three dimensional displacement of the robot is admitted. Thus, the position of the robot trunk is given by 3 coordinates and 3 angles. In addition, the suspension system of the robot is modeled in more details. We assume that each wheel axle admits linear displacement in the direction, parallel to the vertical line of the trunk. The displacement is

restricted by the spring-damping force acting on the wheel from the trunk and representing the suspension system. The normal force of wheel-road interaction is proportional to the amount of the wheel's lowest point penetration into the ground and the speed of the penetration. The force is constrained to be unidirectional. The tangential forces of the wheel-road interactions are described by the brush model the same way as in the simplified model. The wheels are assumed to be actuated by motors. The torque of each motor is considered proportional to the difference between the current and the desired speed of the wheel rotation. The total number of degrees of freedom of the model is equal to 14: 6 for the trunk translation/rotation, 4 for the suspension system, and 4 for the motors of the wheels. We skip more detailed description of the full model. We would like to add that the full model was obtained using Maxima software for symbolic computations. The equations were then transformed into a C++ form, in which they count approximately 30,000 symbols.

III. OPEN-LOOP CONTROLLER

At the first stage, we search for an optimal trajectory of the robot making 90 degrees turn at high velocity. We assume that the desired path is composed of three parts: two linear parts orthogonal to each other and one circular part connecting them (see Fig.4). The robot is located 30 m in front of the turn, directed towards it and has initial velocity of 10 m/s. The robot is than allowed to move for 10 seconds, during which the steering angle and front and rear wheels velocities change piecewise linearly with time. The parametrization of the control inputs is illustrated in Fig. 2. The range of steering angle change is fixed to ± 40 degrees, the linear velocities of the wheels are allowed to vary between 1 and 10 m/s. On the whole, the control inputs are described by 18 parameters, 6 parameters for each.

We search for a solution that minimizes the deviation from the desired path and maximizes the average velocity of the robot's center of mass. As these two objectives are clearly conflicting we use the framework of multiobjective optimization. We find it more convenient than, for example, fixing maximum allowed deviation and then maximizing average velocity of the maneuver. One of the advantages of the multiobjective optimization is that it provides a set of solutions, rather than a single one, thus, giving a researcher a possibility to choose the best one according to his/her criteria. Since the result of the optimization usually represents an approximation of the global optimum, the latter is particularly valuable.

Recent research in stochastic optimization proposed numerous algorithms to simultaneously optimize several objectives [7]; 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.

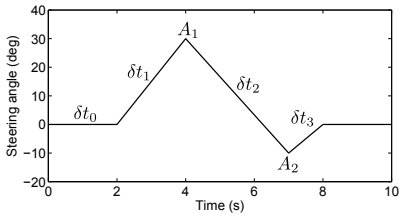

Fig. 2. Schematic representation of the control inputs parametrization for the steering angle.

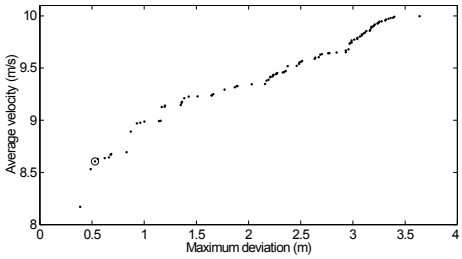

Fig. 3. The Pareto front approximation for open-loop controller. The circle denotes the selected solution (see the text).

The non-dominated set of the entire feasible search space is the globally Pareto-optimal set (Pareto front). It represents the set of optimal *trade-offs*, that is solutions that cannot be improved with respect to one objective without decreasing their score with respect to another one.

Typical algorithms of multiobjective optimization first generate a set of N random points, called a population. Then they enter a loop of four steps until a convergence criteria is met (in this work, a fixed number of iterations is performed):

1) Sort population with respect to dominance such that non-dominated candidate solutions are ranked 1, those which are only dominated by non-dominated ones ranked 2, etc. Candidate solutions that are attributed the same rank are then sorted with regard to a diversity measure (in objective space) to favor solutions in the less crowded parts.
2) Keep only the best N solutions (during the first iteration, this step is useless).
3) Use the sorted population to generate new candidate solutions by perturbing the kept ones (e.g. by adding a Gaussian noise).
4) Merge the newly generated candidate solutions and the previous population; this gives the new population.

Here we used Sferes_v2 framework [8], implementing NSGA-2 [9], which follows the introduced computation scheme. We performed 10,000 iterations with the population size equal to 300.

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. As it can be seen from Fig. 3 the objectives are indeed conflicting. All solutions have the average velocity above 8 m/s. In general, for this range of velocities we need the most precise solution

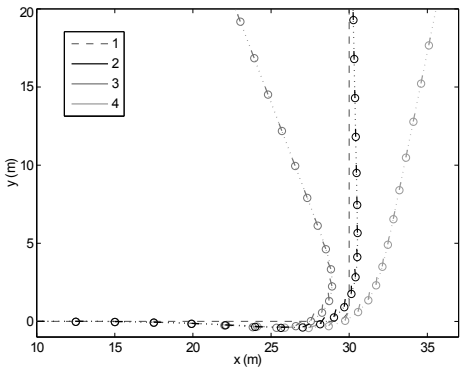

Fig. 4. Desired trajectory (*1*). Robot trajectory obtained by optimization (*2*). Open loop trajectories for disturbed models (*3*, *4*). *3*: $J = 2.5\,\mathrm{kg}\cdot\mathrm{m}^2$ and $M = 30\,\mathrm{kg}$. *4*: $J = 3.5\,\mathrm{kg}\cdot\mathrm{m}^2$ and $M = 50\,\mathrm{kg}$. For *2–4* the circles denote locations of the center of mass of the robot every 250 ms, the lines denote the heading direction of the robot.

available. However, the constraints on suitable solutions are not limited to those on velocity and precision. For this reason we inspected the basic characteristics of the motion of the best 20 solutions (the left most in Fig. 3). It appeared that the best solution has very high value of the slippage angle (angle between the heading direction and the linear velocity): its maximum value exceeded 75 degrees. In opposite, the 3rd best solution (denoted with circle in Fig. 3), which maximum deviation is just 15 cm greater, has considerably smaller value of the slippage angle: about 40 degrees. Since its slippage angle is the smallest among the 5 best solutions, we choose this solution for our further analysis.

The trajectory of the selected solution is given in Fig. 4. About 10 m before the turn the robot slightly steers in the direction opposite of the turn and then about 4 m before the turn changes the steering direction to the one of the turn. During this phase the slippage angle increases dramatically. After the robot reaches desired orientation the steering is steeply changed to zero (see Fig. 8). Simultaneously the angular velocity and the slippage angle of the robot decrease. This manner of driving resembles "pendulum turn" maneuver, used by professional rally racers [4].

The obtained solution is however very sensible to uncertainties. Applying the same control inputs to a model with slightly modified parameters results in significant deviation from the desired trajectory. As it is shown in Fig. 4 the change of mass-inertial properties of the robot dramatically alters the resultant trajectory. Such performance is not surprising and it confirms the necessity of a feedback stabilization system.

IV. CLOSED-LOOP CONTROLLER

A. Architecture

We can assume that the stabilization strategies used by professional rally racers are significantly different during different phases of the maneuver. They probably stabilize the vehicle using something similar to virtual target algorithm when they approach the turn or after the maneuver is almost

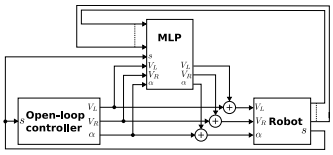

Fig. 5. Schematic representation of the feedback controller used in the current work.

finished, i.e. when the vehicle's angular velocity and slippage angle are small. During the "aggressive" part of the maneuver such controller is evidently inapplicable and other strategies must be used.

We therefore assume that a non-reactive controller is required. However, neither input values nor the mapping between them and the outputs of the controller are known. In such a situation the problem of building the controller can be approached using methods of machine learning. Given a family of non-linear function approximators and efficiency objectives, learning algorithms can find a non-linear function that best matches the objectives. In this work, we selected a multi-layer perceptron (MLP) to represent the family of non-linear mapping functions. Such a simple neural network can approximate a large class of non-linear functions provided that enough hidden neurons are used. Other classes of neural networks (e.g. recurrent neural networks) or any other "universal approximators" could also be employed with the method we describe here.

MLP are classically trained in a supervised fashion with the back-propagation algorithm [10], which uses the error between the observed output and the desired output to correct synaptic weights. This implies that we know the desired outputs of the neural network for each time step and each point in the state space. However, we cannot assume that this information is available, since these values are precisely what we are seeking in the current study. The latter forces us to use an alternative approach, namely, to involve methods of stochastic optimization such as evolutionary algorithms, described above (see section III). Such methods search for the parameters of the approximator, which maximize the required objectives. Multi-objective evolutionary algorithms are known to be quite efficient in neural networks training [11], [12], which makes them perfect candidates for training the MLP to stabilize the motion of the robot in our task.

Since the set of the sensory data that might be required for the controller is unknown, we provide to the neural network all the information that a professional rally-driver might have. We also provide the control values computed by the open-loop controller. They could represent the a priori knowledge of the driver. These control values cannot be parametrized by time because any small speed variation of the robot would "desynchronize" them with the trajectory, making them useless. To avoid this, we resample the outputs of the feedforward controller as functions of distance to the turn. Last, we assume that these distance-parametrized control values require just slight modifications by the MLP. That is why the outputs of the MLP are summed with those

of the open-loop controller (Fig. 5).

In concrete terms, we chose the following inputs to the MLP:

1) s – distance to the turn (negative before the turn and positive after), ranging between -35 and 80 m;
2) magnitude of the velocity of the robot's center of mass, from 0 to 12 m/s;
3) ω – angular velocity of the robot trunk, ± 3 rad/s;
4) signed distance between the trajectory and the robot's center of mass, ± 3 m;
5) angle between the heading direction of the robot and the tangent to the trajectory, ± 270 deg;
6) slippage angle, ± 270 deg;
7) feedforward steering angle (as provided by open-loop controller);
8) feedforward front wheels velocity;
9) feedforward rear wheels velocity.

The outputs of the neural network are the corrections to the steering angle, front and rear velocities of the wheels. These values are supposed to lay within the range of ± 0.2 rad and ± 2 m/s, respectively.

The neural network has therefore 9 inputs and 3 outputs (V_L, V_R and α). One hidden layer with 15 neurons is used. All values are scaled to be in the range $[-1, 1]$ before entering the neural network and scaled back to fit the range for the outputs after leaving it. Neurons have the classical sigmoid transfer function in $[-1, 1]$ range:

$$\varphi(x) = \frac{2}{exp(-\lambda(x+b))} - 1$$

where $\lambda = 7$, x is the weighted sum of inputs and b is a bias, which represents the parameters of the controller. The MLP is therefore parametrized by 198 real values (synaptic weights and biases), each one in the range $[-5, 5]$.

B. Training

The training of the neural network controller is performed off-line. The online schemes, which were proved to be efficient for refining the model of the robot [13], cannot be adopted here, because the efficiency of the movement cannot be estimated locally, and requires it to finish. Thus, the search of the optimal parameters of the MLP is essentially the same as of those of the open-loop controller. However, in order to make the result robust to various disturbances and model uncertainties, we evaluate the performance of the controller for different initial velocities, Coulomb friction coefficients and mass-inertial characteristics of the robot. Instead of using the deviation and the average velocity of a single trajectory, we use the maximum deviation and the minimum average velocity over all listed conditions. These values are minimized / maximized using NSGA-2 (see section III).

The following model parameter changes are used in training session:

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

The population size and the number of iterations of the optimization algorithms were the same as in section III.

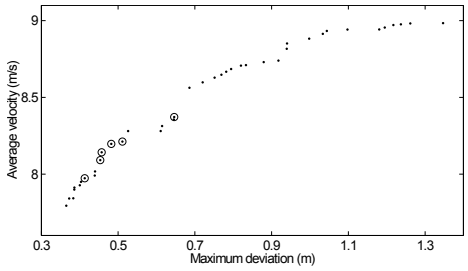

Fig. 6. Pareto from approximation for closed-loop controller.

C. Results

The obtained Pareto front is given in the Fig. 6. As it can be seen, the minimum value of the deviation from the desired trajectory is very small: about 0.4 m. One must remember that this value corresponds to maximum deviation over 7 conditions (1 referent and 6 disturbed). The control outputs generated by one of the solutions are presented in Fig. 8. As it can be seen, the neural network significantly alters the shape of the control (compare the curves "1" and "2").

We are particularly interested in the abilities of the controller to generalize for different types of disturbances. To check it we run a series of "testing" simulations for the following set of disturbances:

1) $\mu = 0.55$ and $J = 2.5\,\mathrm{kg \cdot m^2}$, $M = 30\,\mathrm{kg}$,
2) $\mu = 0.65$ and $J = 2.5\,\mathrm{kg \cdot m^2}$, $M = 30\,\mathrm{kg}$,
3) $\mu = 0.55$ and $J = 3.5\,\mathrm{kg \cdot m^2}$, $M = 50\,\mathrm{kg}$,
4) $\mu = 0.65$ and $J = 3.5\,\mathrm{kg \cdot m^2}$, $M = 50\,\mathrm{kg}$,
5) angle of turn set to 85 degrees,
6) angle of turn set to 95 degrees.

Here we use mixed disturbance of the mass-inertia characteristics and the Coulomb friction coefficient, but also a brand new type of disturbance — the angle of turn. We would like to emphasize that our controller significantly uses a priori information about the angle of turn, so it can be expected not to work correctly for the angles of turn significantly different from 90 degrees. However, we want it to be robust with respect to some uncertainties in the angle of turn.

We tested the best 20 controllers (20 left most points in Fig. 6) in the disturbed conditions, listed above. We found that all of them were capable to cope with the disturbances, however, some of them performed significantly better than the others. To measure the ability of the controllers to generalize we compared the maximum errors in training and testing conditions. The increase of the maximum error in testing conditions ranged from 10% to 500%. In 6 out of 20 solutions the increase of the maximum error was less than 25%. These solutions are denoted with circles in Fig. 6. The disturbed trajectories for one of these individuals are given in the Fig. 7 (left plot).

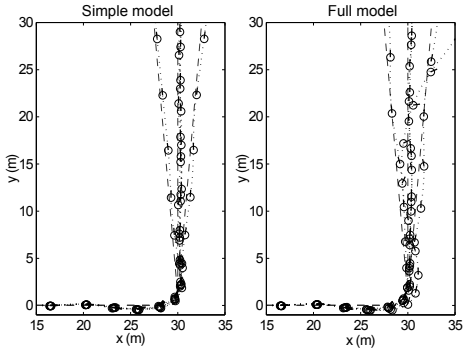

Fig. 7. The trajectories produced by the feedback controller for various disturbances in the simplified and the full model.

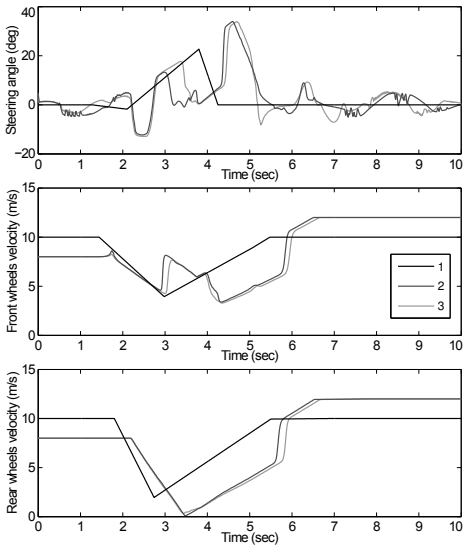

Fig. 8. The control inputs generated by *1*: feedforward controller, *2*: feedback controller with the simplified model, *3*: feedback controller with the full model.

In addition, we use completely different type of disturbance: we run our controllers with the full model. The most essential difference between the full model and the simple one are the following: (i) in full model each wheel has its own dynamics and cannot change the velocity instantaneously; (ii) the suspension system has 4 degrees of freedom that alters the normal forces distribution significantly.

Only 4 controllers out of the preselected 6 worked properly with the full model for training disturbances. None was able to cope with all the testing disturbances. The most hard disturbance for the controllers is the one when the mass and inertia are increased simultaneously with the decrease of the Coulomb friction coefficient. 2 controllers worked properly in all except for that conditions: the maximum deviation from the trajectories stayed below 1.0 m. The trajectories of one of those controllers for test disturbances are presented in Fig. 7. It can be seen that all except for one trajectory stay rather close to the desired ones. The outputs of this controller for the simple and full models are given in Fig. 8. Clearly, they differ significantly, which confirms that switching to full model represents a challenge for the controller.

We would like to note that we used rather large disturbances of the mass-inertia characteristics — about 25 %. For the disturbances of 10 %, the controller works perfectly even with the full model and the deviations stay below 0.6 m.

V. CONCLUSION

In this paper we obtained a closed-loop controller for aggressive maneuver of a wheeled mobile robot moving over loose surface. The maneuver consisted in 90 degrees turn at velocity about 8 m/s. We analyzed the robustness of the controller by varying parameters of the robot itself, the properties of the contact with the ground and running the controller with significantly different models of the robot. The controller appeared to be rather robust with respect to most of the disturbances. We see the following main directions of the development of the current work: (i) developing methods for estimating the region of robustness of the proposed controller, (ii) analyzing relative importance of different sensory inputs to the controller, (iii) implementing the obtained controller on the real mobile robot, which is being currently developed at our institute.

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] 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.

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

[6] 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.

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

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

[9] 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.

[10] S. Haykin, *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, 1998.

[11] X. Yao, "Evolving Artificial Neural Networks," *Proceedings of the IEEE*, vol. 87, no. 9, p. 1423, 1999.

[12] J.-B. Mouret, S. Doncieux, and J.-A. Meyer, "Incremental evolution of target-following neuro-controllers for flapping-wing animats," in *From Animals to Animats: Proceedings of the 9th International Conference on the Simulation of Adaptive Behavior (SAB)*, pp. 606–618, 2006.

[13] M. K. Bugeja and S. G. Farbi, "Multilayer perceptron adaptive dynamics control of mobile robots: Experimental validation," in *European Robotics Symposium 2008, Springer Tracts in Advanced Robotics*, 2008.