Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

An iterative linear quadratic regulator based trajectory tracking controller for wheeled mobile robot[*]

Hao-jie ZHANG[†], Jian-wei GONG[†‡], Yan JIANG, Guang-ming XIONG, Hui-yan CHEN
(*Intelligent Vehicle Research Center, Beijing Institute of Technology, Beijing 100081, China*)
[†]E-mail: {haojie.bit; gjwmit}@gmail.com

Received Dec. 26, 2011; Revision accepted May 3, 2012; Crosschecked July 6, 2012

Abstract: We present an iterative linear quadratic regulator (ILQR) method for trajectory tracking control of a wheeled mobile robot system. The proposed scheme involves a kinematic model linearization technique, a global trajectory generation algorithm, and trajectory tracking controller design. A lattice planner, which searches over a 3D (x, y, θ) configuration space, is adopted to generate the global trajectory. The ILQR method is used to design a local trajectory tracking controller. The effectiveness of the proposed method is demonstrated in simulation and experiment with a significantly asymmetric differential drive robot. The performance of the local controller is analyzed and compared with that of the existing linear quadratic regulator (LQR) method. According to the experiments, the new controller improves the control sequences (v, ω) iteratively and produces slightly better results. Specifically, two trajectories, 'S' and '8' courses, are followed with sufficient accuracy using the proposed controller.

Key words: Lattice planner, Global trajectory, Kinematic model, Trajectory tracking controller, Iterative linear quadratic regulator (ILQR)
doi:10.1631/jzus.C1100379 **Document code:** A **CLC number:** TP242.6

1 Introduction

Navigation and control is one of the most studied problems in robotics. There are two main phases for this: trajectory planning and trajectory tracking. In the first phase, trajectory planning, a suitable trajectory which satisfies robot kinematics is generated to connect the initial posture and the final posture. Once the feasible trajectory is obtained, the navigation and control process enters the second phase, trajectory tracking. For nonholonomic systems such as mobile robots, the existence of nonholonomic constraints makes the time derivatives of some configuration variables nonintegrable. The trajectory tracking control of such systems is difficult to manage, as demonstrated by the results of Luca and Benedetto (1993). Currently two main methods have been proposed to deal with nonholonomic constraints: smooth time-varying control laws (Kolmanovsky and McClamroch, 1995; Morin and Samson, 2000) and discontinuous feedback control laws (Samson, 1993).

Linearization is the most widely used technique to design a controller for nonholonomic systems (Walsh *et al.*, 1994). It works very well when the trajectory tracking control approach is based on the error model of the tracking system. However, the controller generated by linearization can guarantee only local gradual stability. A differential flat strategy is presented with spline theory to generate the local optimal trajectory and the global tracking control law (Oriolo *et al.*, 2002). This method adopts dynamic feedback control to linearize the control system and converges exponentially to zero. The weakness of this approach is that it is not suitable for closed-loop systems which have several singularities or high dimensions. Slide mode control is also successfully applied to trajectory tracking control for mobile robots (Li and Tian, 2000). However, the angular velocity

[‡] Corresponding author
[*] Project (Nos. 90920304 and 91120015) supported by the National Natural Science Foundation of China

cannot reach zero and singularity exists, which violate the assumed terminal slide mode conditions. Some adaptive controllers have also been proposed (Whitcomb *et al.*, 1996; Tomei, 2000). An important point in these methods is that the tracking errors will converge eventually and thus the overall stability of the controller can be guaranteed (Slotine and Li, 1987; Arimoto, 1996). However, this method relies on the exact kinematic model of the system. The trajectory tracking controller based on backstepping technology is introduced and used to track a desired reference trajectory for a four-wheeled robot moving on a horizontal plane (Kumar and Sukaranam, 2008). Divelbiss and Wen (1997) used a time-varying linear quadratic regulator (LQR) to design the local tracking controller for a car trailer system. Our method is an improved method based on this approach. The major difference arises in the technique of obtaining control commands.

Some intelligent control approaches such as fuzzy logic control (Castillo *et al.*, 2006), genetic algorithms (GAs) (Narvydas *et al.*, 2007), and neural networks (Velagic *et al.*, 2008) have also been used to design the trajectory tracking controller. Martínez *et al.* (2009) designed a trajectory tracking controller using type-2 fuzzy logic and GAs. By integrating fuzzy logic control and a GA, a hybrid tracking controller is developed for mobile robots (Astudillo *et al.*, 2007). This method uses a GA to minimize the stabilization error of the kinematic model and synthesizes the controller by type-2 fuzzy logic.

The goal of this paper is to develop a new trajectory tracking controller and compare its performance with those of the existing methods on the global trajectory tracking problem for a nonholonomic wheeled mobile robot. The new method based on LQR control theory linearizes iteratively the nonlinear system around a nominal trajectory. Then, it computes a local optimal feedback control law using a modified iterative LQR (ILQR) technique. The control law obtained is applied to the system again and generates the local optimal trajectory. The results of obstacle detection based on the local trajectory are used to improve the control law incrementally. Compared with the existing methods, the new method improves the control law iteratively yielding better results. The new method converges substantially faster. Most importantly, we are implementing the controller on a real-world significantly asymmetric differential drive robot, called Segbot, in our laboratory.

2 Mobile robot system

2.1 Platform description

The Segbot used for experiments in this study (Fig. 1a) is a new mobile robotic platform based on the self-balancing Segway human transporter (HT). This platform has been used in a variety of autonomous mobile robot experiments, including navigation and localization, global and local trajectory planning, and trajectory tracking.

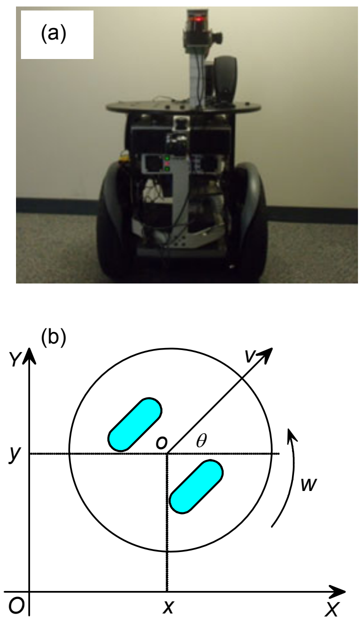

Fig. 1 Photo of Segbot (a) and the simple kinematic model used for tracking control (b)

The Segbot is equipped with two laser range finders (Hokuyo UTM-30LX and URG-04LX), an on-board computer, a base controller, and other devices. The Hokuyo laser range finders are used to detect obstacles and collect information from the environments. Based on the collected information, we construct a global cost map for global trajectory planning and a local cost map for trajectory tracking. The on-board computer processes messages from laser range finders and computes the desired control commands to track the global trajectory. Then, it sends the control commands to the base controller.

The base controller receives control commands from the computer and drives Segbot around.

2.2 System architecture

The operating system of Segbot contains two modes: teleoperation mode and autonomous mode. Teleoperation mode allows the user to control the robot through a joystick and build the environment map by driving the robot, while in autonomous mode the robot needs to generate and track a trajectory autonomously. In autonomous mode, trajectory planning and trajectory tracking run in the robot operating system (ROS) which provides libraries and tools to help software developers create robot applications. The entire software system consists of three major blocks: global planner, trajectory generator, and base controller (Fig. 2).

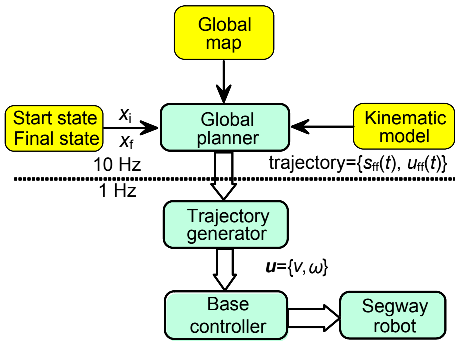

Fig. 2 Structure diagram of the trajectory tracking system
x_i is the start state, x_f is the goal state, $s_{ff}(t)$ is the state sequence, $u_{ff}(t)$ is the nominal control sequence, $\boldsymbol{u}$ is the desired control vector, v is the desired translational velocity, and ω is the desired angular velocity

The global planner component is used to search through the entire graph-based configuration space and to compute the minimum cost trajectory which is collision-free with obstacles from the initial posture to the goal posture, based on the knowledge of the robot's kinematic model and the global cost map. It is essential to guide the motion of the robot towards the goal.

The trajectory generator component receives the global trajectory (in the form of nominal state sequence $s_{ff}(t)$ and nominal control sequence $u_{ff}(t)$) from the global planner and computes the real-time control output $\boldsymbol{u}$ to follow the global trajectory. The control vector $\boldsymbol{u}$ is then applied to the robot system and simulated to generate the local trajectory p for a short time period. If p is collision-free with obstacles, it will send the control commands $\boldsymbol{u}$ to the base controller component.

The base controller component is responsible for receiving the desired control commands $\boldsymbol{u}$ from the trajectory generator component and driving Segbot around.

3 Global trajectory generation

The global planner layer is responsible for generating a global trajectory that is collision-free with obstacles from an initial posture towards a goal posture. To efficiently plan a smooth trajectory to the goal posture, we use the Anytime Dynamic A* (AD*) planner that searches over the lattice state space which is constructed using the robot position (x, y) and orientation (θ). The lattice state space uses a set of discrete states to represent the configuration space. For these states, each connection between them represents a feasible trajectory (Pivtoraiko *et al.*, 2009). Fig. 3 gives an example of a lattice. The lattice state space is very suitable for planning for non-holonomic and highly-constrained robotic systems. For more information about the lattice state space and the AD* planning algorithm, please refer to Likhachev and Ferguson (2009).

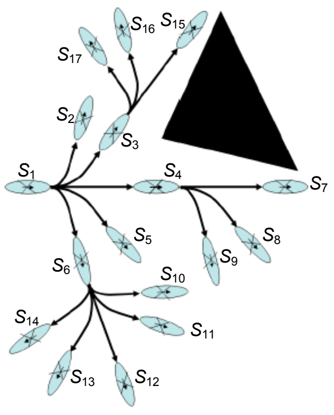

Fig. 3 Example of a lattice

The set of possible local maneuvers considered for each state (x, y, θ) in the planner's search space is constructed off-line using the robot's model, so that they can be accurately executed by the robot. The off-line construction of the lattice state space is based on the work by Pivtoraiko and Kelly (2005), which attempts to create near-minimum spanning action spaces. This planner searches in a backward direction from the goal posture to the start posture and generates a trajectory which consists of a sequence of feasible maneuvers that are collision-free with the static obstacles observed in the environment. The global trajectory generated is a sequence of states, denoted as nominal state sequence $s_{\text{ff}}=\{s_0, s_1, \ldots, s_N\}$ (Fig. 4).

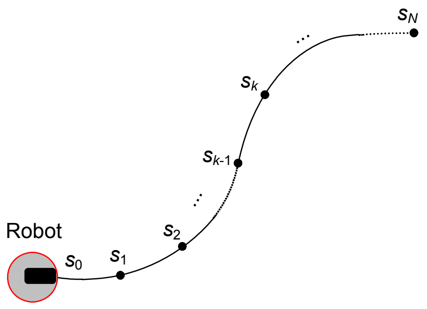

Fig. 4 Example of the global trajectory

The generated global trajectory is also used to calculate the nominal control sequence u_{ff}. As shown in Fig. 4, for segment (s_{k-1}, s_k) on the global trajectory, the nominal control $\boldsymbol{u}_k$ can be calculated by

$$\boldsymbol{u}_k=\begin{bmatrix} v_k \\ \omega_k \end{bmatrix}=\begin{bmatrix} \sqrt{(x_k-x_{k-1})^2+(y_k-y_{k-1})^2}\,/\,\mathrm{d}t \\ (\theta_k-\theta_{k-1})\,/\,\mathrm{d}t \end{bmatrix}. \tag{1}$$

The nominal control sequence u_{ff} is defined as

$$u_{\text{ff}}=\{\boldsymbol{u}_1,\boldsymbol{u}_2,\cdots,\boldsymbol{u}_N\}, \tag{2}$$

where $\boldsymbol{u}_k$ (k=1, 2, …, N) represents the nominal control from state s_{k-1} to state s_k.

The nominal trajectory sequence s_{ff} and nominal control sequence u_{ff} are used for trajectory tracking in Section 4.

4 Trajectory tracking with Segbot

4.1 Kinematic model of Segbot

The Segbot is a two-wheel circular robot (Fig. 1a). It is assumed that the robot does not have any slip on the surface. In other words, the robot wheels are completely in touch with the ground. The simple kinematic model of the robot is shown in Fig. 1b. Point o is located in the center of the axis between two wheels, and θ is the angle between the line perpendicular to the wheel axis and the X axis. The translational velocity and angular velocity of the robot are denoted as v and ω, respectively.

The posture of the robot in the unknown environment at any time is defined as state (x, y, θ) in the reference coordinate system. The kinematic model of Segbot is defined with

$$\begin{cases} \dot{x}=v\cos\theta, \\ \dot{y}=v\sin\theta, \\ \dot{\theta}=\omega. \end{cases} \tag{3}$$

The robot is controlled by its translational velocity v and angular velocity ω. The control vector $\boldsymbol{u}$ is therefore defined as

$$\boldsymbol{u}=[v,\omega]^{\mathrm{T}}. \tag{4}$$

Combined with position information, $\boldsymbol{u}$ can be expanded to a seven-state vector $\boldsymbol{x}$ as the state variable:

$$\boldsymbol{x}=[x,y,\theta,v,\omega,\delta v,\delta\omega]^{\mathrm{T}}. \tag{5}$$

Define t as time for the linear movement, the angular movement, and the sampling movement. The state $\boldsymbol{x}_{k+1}$ at time $(k+1)\mathrm{d}t$ can be expressed by

$$\boldsymbol{x}_{k+1}=\begin{bmatrix} x_{k+1} \\ y_{k+1} \\ \theta_{k+1} \\ v_{k+1} \\ \omega_{k+1} \\ \delta v_{k+1} \\ \delta\omega_{k+1} \end{bmatrix}=\begin{bmatrix} x_k+v_k\cos\theta\mathrm{d}t \\ y_k+v_k\sin\theta\mathrm{d}t \\ \theta_k+\omega_k\mathrm{d}t \\ u_k(0) \\ u_k(1) \\ u_k(0)-v_k \\ u_k(1)-\omega_k \end{bmatrix}=f(\boldsymbol{x}_k,\boldsymbol{u}_k). \tag{6}$$

4.2 Trajectory controller

The system of the robot is a nonlinear dynamical system with state variable $\boldsymbol{x}_k\in\mathbb{R}^{n_x}$ and control variable $\boldsymbol{u}_k\in\mathbb{R}^{n_u}$, as indicated by Eq. (6). We use the ILQR technique based on the LQR method to design the local controller. We begin with a nominal control sequence u_{ff}, and a corresponding nominal trajectory sequence s_{ff} obtained by applying $\boldsymbol{u}_k$ to the dynamical system in an open loop iteratively. An improved sequence u_{ff} is produced by linearizing the system dynamics around control $\boldsymbol{u}_k$ and state $\boldsymbol{x}_k$ and then solving a modified ILQR problem. The iteration is repeated until convergence. Let the deviations from $\boldsymbol{u}_k$ and $\boldsymbol{x}_k$ be $\delta\boldsymbol{u}_k$ and $\delta\boldsymbol{x}_k$, respectively. The linearization is

$$\delta\boldsymbol{x}_{k+1} = \boldsymbol{A}_k\delta\boldsymbol{x}_k + \boldsymbol{B}_k\delta\boldsymbol{u}_k, \tag{7}$$

where $\boldsymbol{A}_k=J_xf(\boldsymbol{x}_k, \boldsymbol{u}_k)$, $\boldsymbol{B}_k=J_uf(\boldsymbol{x}_k, \boldsymbol{u}_k)$. J_x and J_u denote the Jacobians of $f(\cdot)$ with respect to $\boldsymbol{x}$ and $\boldsymbol{u}$ respectively, and the Jacobians are evaluated along $\boldsymbol{x}_k$ and $\boldsymbol{u}_k$.

After the linearization model is established using Eq. (7), the ILQR problem can be solved with the cost function below:

$$\begin{aligned} J = &\frac{1}{2}(\boldsymbol{x}_N + \delta\boldsymbol{x}_N - \boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{Q}_{\mathrm{f}}(\boldsymbol{x}_N + \delta\boldsymbol{x}_N - \boldsymbol{x}^*) \\ &+\frac{1}{2}\sum_{k=0}^{N-1}\{(\boldsymbol{x}_k + \delta\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}_k + \delta\boldsymbol{x}_k) \\ &+(\boldsymbol{u}_k + \delta\boldsymbol{u}_k)^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{u}_k + \delta\boldsymbol{u}_k)\}. \end{aligned} \tag{8}$$

We add the constraint to Eq. (8) and construct a value function

$$\begin{aligned} V = &\frac{1}{2}(\boldsymbol{x}_N + \delta\boldsymbol{x}_N - \boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{Q}_{\mathrm{f}}(\boldsymbol{x}_N + \delta\boldsymbol{x}_N - \boldsymbol{x}^*) \\ &+\frac{1}{2}\sum_{k=0}^{N-1}\{(\boldsymbol{x}_k + \delta\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}_k + \delta\boldsymbol{x}_k) \\ &+(\boldsymbol{u}_k + \delta\boldsymbol{u}_k)^{\mathrm{T}}R(\boldsymbol{u}_k + \delta\boldsymbol{u}_k) \\ &+\boldsymbol{\lambda}_{k+1}^{\mathrm{T}}(\boldsymbol{A}_k\delta\boldsymbol{x}_k + \boldsymbol{B}_k\delta\boldsymbol{u}_k - \delta\boldsymbol{x}_{k+1})\}, \end{aligned} \tag{9}$$

where $\boldsymbol{\lambda}_{k+1}^{\mathrm{T}}$ is the Lagrange multiplier.

We begin with the Hamiltonian function

$$\begin{aligned} H_k = &\frac{1}{2}(\boldsymbol{x}_k + \delta\boldsymbol{x}_k)^{\mathrm{T}}\boldsymbol{Q}(\boldsymbol{x}_k + \delta\boldsymbol{x}_k) \\ &+(\boldsymbol{u}_k + \delta\boldsymbol{u}_k)^{\mathrm{T}}\boldsymbol{R}(\boldsymbol{u}_k + \delta\boldsymbol{u}_k) \\ &+\boldsymbol{\lambda}_{k+1}^{\mathrm{T}}(\boldsymbol{A}_k\delta\boldsymbol{x}_k + \boldsymbol{B}_k\delta\boldsymbol{u}_k). \end{aligned} \tag{10}$$

After substituting Eq. (10) into the value function (9), a reformulation leads to

$$\begin{aligned} V = &\frac{1}{2}(\boldsymbol{x}_N + \delta\boldsymbol{x}_N - \boldsymbol{x}^*)^{\mathrm{T}}\boldsymbol{Q}_{\mathrm{f}}(\boldsymbol{x}_N + \delta\boldsymbol{x}_N - \boldsymbol{x}^*) \\ &-\frac{1}{2}\left(\boldsymbol{\lambda}_N^{\mathrm{T}}\delta\boldsymbol{x}_N + H_0 + \sum_{k=1}^{N-1}(H_k - \boldsymbol{\lambda}_k^{\mathrm{T}}\delta\boldsymbol{x}_k)\right). \end{aligned} \tag{11}$$

When V takes the minimal value, the necessary condition is as follows:

$$\frac{\partial H_k}{\partial(\delta\boldsymbol{x}_k)} = \boldsymbol{\lambda}_k,\ \frac{\partial H_k}{\partial(\delta\boldsymbol{u}_k)} = \boldsymbol{0},\ \frac{\partial H_k}{\partial(\delta\boldsymbol{x}_N)} = \boldsymbol{\lambda}_N. \tag{12}$$

Thus,

$$\boldsymbol{\lambda}_k = \boldsymbol{A}_k^{\mathrm{T}}\delta\boldsymbol{\lambda}_{k+1} + \boldsymbol{Q}(\delta\boldsymbol{x}_k + \boldsymbol{x}_k), \tag{13}$$

$$\boldsymbol{0} = \boldsymbol{R}(\boldsymbol{u}_k + \delta\boldsymbol{u}_k) + \boldsymbol{B}_k^{\mathrm{T}}\delta\boldsymbol{\lambda}_{k+1}, \tag{14}$$

$$\boldsymbol{\lambda}_N = \boldsymbol{Q}_{\mathrm{f}}(\boldsymbol{x}_N + \delta\boldsymbol{x}_N - \boldsymbol{x}^*). \tag{15}$$

Based on the boundary condition (13), $\boldsymbol{\lambda}_k$ is assumed as

$$\boldsymbol{\lambda}_k = \boldsymbol{S}_k\delta\boldsymbol{x}_k + \boldsymbol{v}_k \tag{16}$$

for some unknown sequences $\boldsymbol{S}_k$ and $\boldsymbol{v}_k$. The boundary conditions for $\boldsymbol{S}_k$ and $\boldsymbol{v}_k$ are as follows:

$$\boldsymbol{S}_N = \boldsymbol{Q}_{\mathrm{f}},\ \ \boldsymbol{v}_N = \boldsymbol{Q}_{\mathrm{f}}(\boldsymbol{x}_N - \boldsymbol{x}^*). \tag{17}$$

Solving Eqs. (7), (14), and (16), we obtain

$$\delta\boldsymbol{u}_k = -\boldsymbol{K}\delta\boldsymbol{x}_k - \boldsymbol{K}_v\boldsymbol{v}_{k+1} - \boldsymbol{K}_u\boldsymbol{u}_k, \tag{18}$$

where

$$\boldsymbol{K} = (\boldsymbol{B}_k^{\mathrm{T}}\boldsymbol{S}_{k+1}\boldsymbol{B}_k + \boldsymbol{R})^{-1}\boldsymbol{B}_k^{\mathrm{T}}\boldsymbol{S}_{k+1}\boldsymbol{A}_k, \tag{19}$$

$$\boldsymbol{K}_v = (\boldsymbol{B}_k^{\mathrm{T}}\boldsymbol{S}_{k+1}\boldsymbol{B}_k + \boldsymbol{R})^{-1}\boldsymbol{B}_k^{\mathrm{T}}, \tag{20}$$

$$\boldsymbol{K}_u = (\boldsymbol{B}_k^{\mathrm{T}}\boldsymbol{S}_{k+1}\boldsymbol{B}_k + \boldsymbol{R})^{-1}\boldsymbol{R}, \tag{21}$$

$$\boldsymbol{S}_k = \boldsymbol{A}_k^{\mathrm{T}}\boldsymbol{S}_{k+1}(\boldsymbol{A}_k - \boldsymbol{B}_k\boldsymbol{K}) + \boldsymbol{Q}, \tag{22}$$

$$\boldsymbol{v}_k = (\boldsymbol{A}_k - \boldsymbol{B}_k\boldsymbol{K})^{\mathrm{T}}\boldsymbol{v}_{k+1} - \boldsymbol{K}^{\mathrm{T}}\boldsymbol{R}\boldsymbol{u}_k + \boldsymbol{Q}\boldsymbol{x}_k. \tag{23}$$

In Eq. (18), $\delta\boldsymbol{x}_k$ represents the error between the robot's current state and the closest state to the robot on the nominal trajectory sequence s_{ff}. With the boundary condition $\boldsymbol{S}_N$ given as the final state weighting matrix in the cost function (8), we can solve for the entire sequence of $\boldsymbol{S}_k$ and an entire sequence of $\boldsymbol{v}_k$ by the backward recursion Eqs. (22) and (23), respectively. Once the modified ILQR problem is solved, an improved nominal control sequence can be constructed:

$$\boldsymbol{u}_k^* = \boldsymbol{u}_k + \delta\boldsymbol{u}_k, \tag{24}$$

where $\boldsymbol{u}_k$ is the nominal control and $\boldsymbol{u}_k^*$ is the improved control.

4.3 Implementation of trajectory tracking

In the phase of implementing trajectory tracking, we find the point s_k which is the closest one on the global trajectory to the robot's current position (Fig. 5).

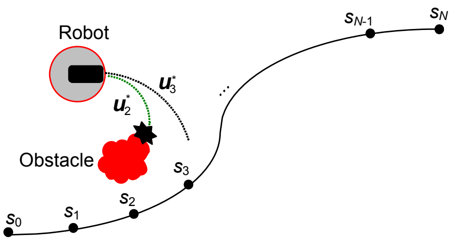

Fig. 5 Implementation of trajectory tracking

As shown in Algorithm 1, Eq. (24) is used to calculate the improved control vector $\boldsymbol{u}_k^*$ (line 2). We simulate and generate the local trajectory with the improved control $\boldsymbol{u}_k^*$ over a short time period (~0.1 s). The simulated trajectory of the robot is presented as a sequence of 5-dimensional states $(x, y, \theta, v, \omega)$ with a discrete-time approximation of its dynamics. The simulated trajectory p_k projected on the (x, y) plane is a smooth, continuous 2-dimensional curve and is used for collision checking by using the footprint of the robot (line 3). If it is in collision with the obstacles (line 4), we find the next point s_{k+1} to calculate the improved control vector $\boldsymbol{u}_{k+1}^*$ (line 5) and the simulated trajectory p_{k+1} is still used for collision checking until p_{k+1} is collision-free with obstacles. Then, we use the improved control $\boldsymbol{u}_{k+1}^*$ to drive the robot (line 8). As shown in Fig. 5, s_2 is the closest point to the robot's current position and the simulated trajectory p_2 is shown as the line near $\boldsymbol{u}_2^*$. Trajectory p_2 is in collision with obstacles. Then, s_3 is chosen to calculate the control vector $\boldsymbol{u}_3^*$ and the simulated trajectory p_3, which is the line near $\boldsymbol{u}_3^*$, is collision-free with obstacles. As a result, control vector $\boldsymbol{u}_3^*$ is used to drive the robot.

Algorithm 1 Implementation of trajectory tracking

```
1 find the closest point $s_k$
2 calculate the improved control vector $u_k^*$
3 simulate and generate trajectory $p_k$ by using $u_k^*$
4 if $p_k$ is in collision with obstacles
5     $k$++
6 end if
7 else
8     apply $u_k^*$ to the robot
9 end
```

5 Experimental results

To show the benefits and effectiveness of the controller, we ran experiments both in simulation and on Segbot.

5.1 Simulation

We set up two environments: 'S' course and '8' course (Fig. 6). The 'S' course environment is found in everyday driving. It was chosen to test the controller's rapid response to the steering commands. The '8' course environment includes two circular paths and a point with discontinuous curvature where the robot needs to travel from one circle to another. This environment is not commonly found in real-world scenarios. However, it is valuable to test some important characteristics of a controller, such as the steady state characteristics while executing a constant nonzero curvature, and robustness to discontinuous curvature.

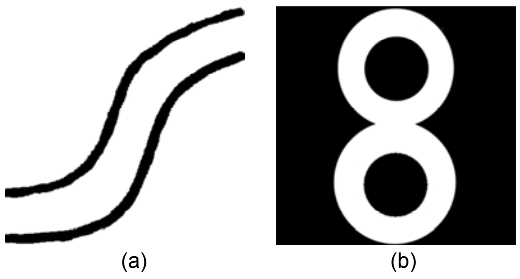

Fig. 6 'S' (a) and '8' (b) course environments

For these two simulations, we set the initial posture and the global posture manually. Then the global trajectory was generated using the global trajectory generation algorithm as described in Section 3, and the controller was implemented on tracking the global trajectory. Determining matrices $\boldsymbol{Q}$ and $\boldsymbol{R}$ is not a trivial task. In our experiments, $\boldsymbol{Q}$ and $\boldsymbol{R}$ are chosen to be time-invariant and diagonal:

$$\boldsymbol{Q} = \mathrm{diag}\{q_1, q_2, \ldots, q_7\}, \tag{25}$$

$$\boldsymbol{R} = \mathrm{diag}\{r_1, r_2\}, \tag{26}$$

where $q_1=q_2=q_3=10$, $q_4=q_5=0.001$, $q_6=q_7=2$, $r_1=r_2=0.001$.

The experimental results of comparison with the LQR controller are shown in Figs. 7 and 8, when the robot follows the 'S' course trajectory and the '8' course trajectory, respectively.

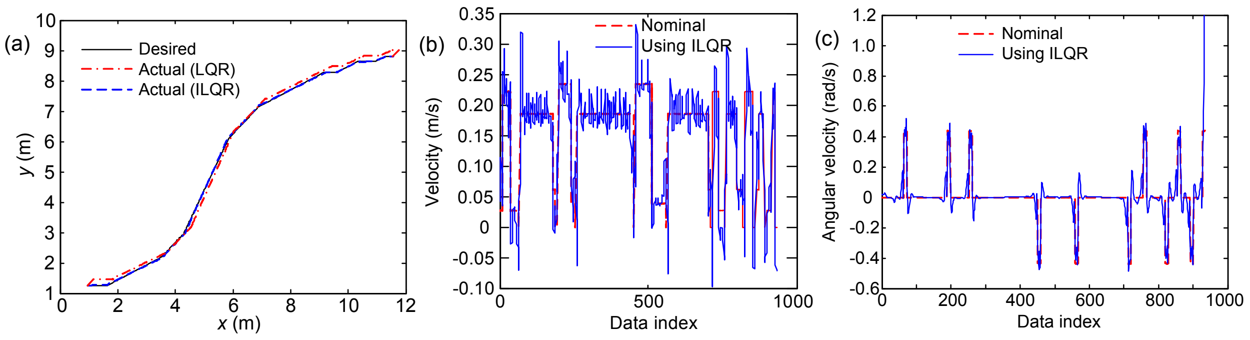

Fig. 7 Simulation results from the 'S' course environment
(a) Desired trajectory and two actual trajectories taken by the robot using the LQR trajectory controller and our ILQR trajectory controller; (b) Nominal translational velocity and improved translational velocity; (c) Nominal angular velocity and improved angular velocity

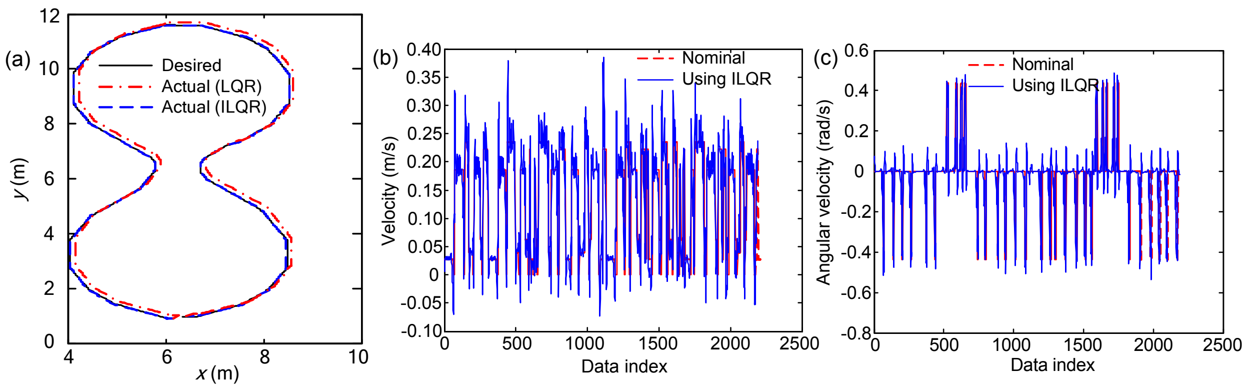

Fig. 8 Simulation results from the '8' course environment
(a) Desired trajectory and two actual trajectories taken by the robot using the LQR trajectory controller and our ILQR trajectory controller; (b) Nominal translational velocity and improved translational velocity; (c) Nominal angular velocity and improved angular velocity

Figs. 7a and 8a show that the errors arising in tracking the desired global trajectory when the ILQR trajectory controller is adopted is sufficiently small in comparison with the errors when the LQR trajectory controller is used. Figs. 7b and 8b show the nominal velocities and the improved translational velocities after using ILQR of the robot. Figs. 7c and 8c show the angular velocities and the improved angular velocities after using ILQR of the robot. The velocities and angular velocities after using ILQR are changing relatively smoothly. As a result, it is easy for the robot to reach the improved velocities and angular velocities.

5.2 Tests on Segbot

To show that the controller works well in real-world environments, we implemented it in ROS and tested it on Segbot. We wrote a base local planner node which was plugged into the ROS's navigation stack. When the global trajectory is generated, the navigation stack will awake the base local planner to follow the global trajectory. The base local planner then safely generates a local trajectory and sends the control vector (including v, ω) to the base controller of the robot.

The test scenario is that the robot gets stuck by the obstacles (Fig. 9). The global trajectory is generated using the lattice planner. To follow the global trajectory, the controller will drive the robot to move backwards. As expected, the robot followed the global trajectory perfectly and got out of the dead end (the full motion video is available from https://sites.google.com/site/haojie9527/video).

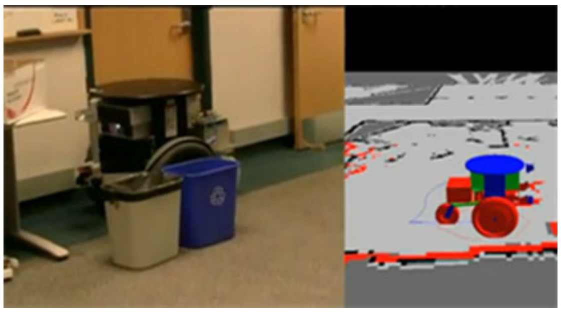

Fig. 9 The experiment scenario
The Segbot is getting out of the dead end. The solid line behind the robot in the right-hand graph is the global trajectory

6 Conclusions

In this paper we present the experimental results of the trajectory tracking control of a wheeled mobile robot system. The lattice planner is used for global trajectory planning and ILQR technology is used to design the local controller. We obtain the nominal control vector from the global trajectory and improve it using the ILQR technique. 'S' and '8' course environments are used to test the new tracking method. The results of the tests show that trajectory tracking is stable and that the tracking error is minimal.

References

Arimoto, S., 1996. Control Theory of Nonlinear Mechanical Systems: a Passivity-Based and Circuit-Theoretic Approach. Clarendon Press, Oxford, UK.

Astudillo, L., Castillo, O., Aguilar, L.T., Martínez, R., 2007. Hybrid control for an autonomous wheeled mobile robot under perturbed torques. *LNCS*, **4529**:594-603. [doi:10.1007/978-3-540-72950-1_59]

Castillo, O., Aguilar, L.T., Cardenas, S., 2006. Fuzzy logic tracking control for unicycle mobile robots. *Eng. Lett.*, **13**(2):73-77.

Divelbiss, A.W., Wen, J.T., 1997. Trajectory tracking control of a car-trailer system. *IEEE Trans. Control Syst. Technol.*, **5**(3):269-278. [doi:10.1109/87.572125]

Kolmanovsky, I., McClamroch, N.H., 1995. Developments in nonholonomic control systems. *IEEE Control Syst. Mag.*, **15**(6):20-36. [doi:10.1109/37.476384]

Kumar, U., Sukaranam, N., 2008. Backstepping based trajectory tracking control of a four wheeled mobile robot. *Int. J. Adv. Robot. Syst.*, **5**(4):403-410.

Li, S.H., Tian, Y.P., 2000. Trajectory tracking control of mobile vehicle. *Control Dec.*, **15**(5):626-628.

Likhachev, M., Ferguson, D., 2009. Planning long dynamically feasible maneuvers for autonomous vehicles. *Int. J. Robot. Res.*, **28**(8):933-945. [doi:10.1177/0278364909340445]

Luca, A.R., Benedetto, M.D., 1993. Control of nonholonomic systems via dynamic compensation. *Kybernetica*, **29**(6):593-608.

Martínez, R., Castillo, O., Aguilar, L.T., 2009. Optimization of interval type-2 fuzzy logic controllers for a perturbed autonomous wheeled mobile robot using genetic algorithms. *Inf. Sci.*, **179**(13):2158-2174. [doi:10.1016/j.ins.2008.12.028]

Morin, P., Samson, C., 2000. Practical Stabilization of a Class of Nonlinear Systems. Application to Chained Systems and Mobile Robots. Proc. 39th IEEE Conf. on Decision Control, p.2989-2994. [doi:10.1109/CDC.2000.914274]

Narvydas, G., Simutis, R., Raudonis, V., 2007. Autonomous Mobile Robot Control Using Fuzzy Logic and Genetic Algorithm. 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, p.460-464. [doi:10.1109/IDAACS.2007.4488460]

Oriolo, G., de Luca, A., Vendittelli, M., 2002. WMR control via dynamic feedback linearization: design, implementation and experimental validation. *IEEE Trans. Control Syst. Technol.*, **10**(6):835-852. [doi:10.1109/TCST.2002.804116]

Pivtoraiko, M., Kelly, A., 2005. Generating Near Minimal Spanning Control Sets for Constrained Motion Planning in Discrete State Spaces. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, p.3231-3237. [doi:10.1109/IROS.2005.1545046]

Pivtoraiko, M., Knepper, R.A., Kelly, A., 2009. Differentially constrained mobile robot motion planning in state lattices. *J. Field Robot.*, **26**(3):308-333. [doi:10.1002/rob.20285]

Samson, C., 1993. Time-varying feedback stabilization of car-like mobile robots. *Int. J. Robot. Res.*, **12**(1):55-64. [doi:10.1177/027836499301200104]

Slotine, J.J.E., Li, W., 1987. On the adaptive control of robot manipulators. *Int. J. Robot. Res.*, **6**(3):49-59. [doi:10.1177/027836498700600303]

Tomei, P., 2000. Robust adaptive friction compensation for tracking control of robot manipulators. *IEEE Trans. Autom. Control*, **45**(11):2164-2169. [doi:10.1109/9.887661]

Velagic, J., Osmic, N., Lacevic, B., 2008. Neural network controller for mobile robot motion control. *World Acad. Sci. Eng. Technol.*, **47**:193-198.

Walsh, G., Tilbury, D., Sastry, S., Murray, R., Laumond, J.P., 1994. Stabilization of trajectories for systems with nonholonomic constraints. *IEEE Trans. Autom. Control*, **39**(1):216-222. [doi:10.1109/9.273373]

Whitcomb, L.L., Arimoto, S., Naniwa, T., Ozaki, F., 1996. Experiments in adaptive model-based robot force control. *IEEE Control Syst. Mag.*, **16**(1):49-57. [doi:10.1109/37.482150]