

# On Computing Robust Controllers for Mobile Robot Trajectory Calculus: Lyapunov

Radu Bogdan Rusu and Marius Borodi

Robotics Research Group, Faculty of Automation and Computer Science, Technical University of Cluj-Napoca  
 Baritiu Str. no 26-28, Cluj-Napoca, Romania  
 {radu.rusu; marius.borodi}@aut.utcluj.ro

**Abstract**—We present a simple method for a mobile robot’s trajectory calculus using the famous Lyapunov equations. This research has been already conducted by several other researchers before us, so we take no credit for it. We just reproduced the results for ourselves, by studying Lyapunov’s theories and the papers presented in the references section. While the mathematical foundation is quite easy to understand, we feel that these results have not been exploited enough in mobile robotics. Therefore, we strongly encourage people entering this area of research, to glance at them before going deeper and start studying more complex methods.

**Index Terms**—mobile robots, robust controller, trajectory calculus, Lyapunov equations

## I. INTRODUCTION

One of the most common tasks in the field of mobile robotics, is to move a robot from a starting point S to a final goal point G, with the maximum possible speed. The resulting trajectory has to be optimally calculated, without stepping out of the robot’s physical properties boundaries.

The information used for estimating a mobile robot’s position is based on its simplest form on odometry. Problem is, odometry can be very unreliable in the real world, mostly because of perturbations and wheel slippings, hence it can lead to a lot of uncertainties, unknown a-priori. Thus, the need to find a robust controller that will stabilize the system is mandatory, if the proposed performances have to be reached.

In this paper we present the results achieved by using a controller based on some of Lyapunov’s equations. The starting kinematic model’s equations for a 2-wheel differential drive robot are presented in section 2, the calculus method for the control laws in section 3, experimental results in section 4, and conclusions in section 5. The complete source code used to generate the results is presented in section 6.

## II. KINEMATIC MODEL

The starting kinematic model for the mobile robot is presented in the following formula. X and Y are the robot’s coordinates in the Cartesian space, and  $\phi$  is the angle between the robot’s current position and the X axis.

$$\begin{cases} \dot{x} = u * \cos(\phi) \\ \dot{y} = u * \sin(\phi) \\ \dot{\phi} = \omega \end{cases}$$

$u$  is the linear velocity of the mobile robot, and  $\omega$  is the angular velocity. All the above mathematical elements are also presented in figure 1.

The trajectory is actually composed of several *via* points. A special case is the trajectory between two points, a starting point given by  $(x_s, y_s, \phi_s)$  and a goal point given by  $(x_g, y_g, \phi_g)$ . The proposed algorithm will calculate a controller that has to move the robot from the starting point to a goal point equal with  $(0,0,0)$ .

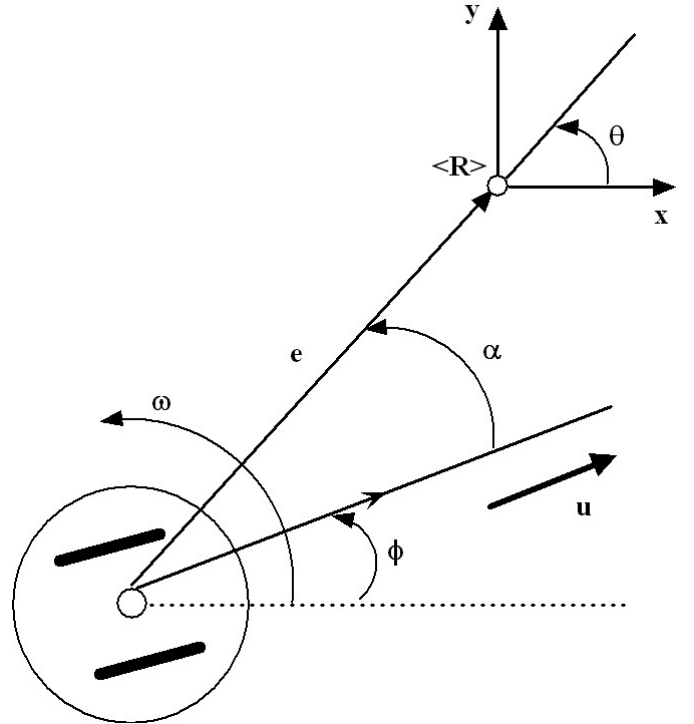


Fig. 1. Mobile robot’s kinematic model elements

The following mathematical substitution is applied.

$$\begin{cases} e = \sqrt{x^2 + y^2} \\ \theta = \text{atanh}(y, x) \\ \alpha = \theta - \phi \end{cases}$$

*atanh* is the tangent’s inverse in the 4th quadrant of the trigonometric circle ( $\text{atanh}(y, x) \in (-\pi, \pi]$ ).

By substituting the new coordinates  $e, \theta$  and  $\alpha$ , the kinematic model becomes:

$$\begin{cases} \dot{e} = -u * \cos(\theta - \phi) \\ \dot{\theta} = u * \frac{\sin\alpha}{e} \\ \dot{\phi} = \omega \end{cases}$$

And, by replacing  $\alpha$  with  $\theta - \phi$ , the final kinematic model is:

$$\begin{cases} \dot{e} = -u * \cos\alpha \\ \dot{\alpha} = -\omega + u * \frac{\sin\alpha}{e} \\ \dot{\theta} = u * \frac{\sin\alpha}{e} \end{cases}$$

Although from the above equations, one can deduct a variety of kinematic models, we choose this one for our algorithm.

One mention must be made: the above equations are only valid if the error  $e$  is not 0. Therefore, the trajectory between the two points will never be calculated as a straight line, but as to always induce a very small error.

When used on a mobile robot, a few limitations are occurring:

- the linear velocity  $u$  is upper bounded by the maximum linear velocity of the robot:  $u \leq u_{max}$ ;
- the angular velocity  $\omega$  is upper bounded by the maximum angular velocity of the robot:  $\omega \leq \omega_{max}$ ;
- the lateral acceleration  $a = u * \omega$  must be smaller than  $a_{max}$ , because our robot model only uses odometry information, and if the lateral acceleration is too big, one of the wheels could detach from the ground on a turn, thus the odometry information will get completely corrupted.

### III. CLOSED LOOP CALCULUS USING LYAPUNOV EQUATIONS

In order to demonstrate the efficiency of the algorithm, we are choosing the parking problem.

Let there be a mobile robot, initially positioned at some coordinates different than  $(0, 0, 0)$  (goal point). Let there be a set of measurable state variables  $[e, \alpha, \theta]$ , with  $e > 0$ . The task is to find a proper control law,  $[u, \omega] = g(e, \alpha, \theta)$  that guarantees a stable asymptotic convergence of the system's state to the goal state  $[0, 0, 0]$ .

The generalized form of a Lyapunov function relevant to our control law is:

$$V = V_1 + V_2 = \frac{1}{2}\lambda e^2 + \frac{1}{2}(\alpha^2 + h\theta^2); \lambda, h > 0$$

$V_1$  and  $V_2$  characterize the mean square error of the distance vector  $e$  and of the alignment vector  $[\alpha, \sqrt{h}\theta]$ . By deriving  $V$ , the Lyapunov function becomes:

$$\begin{aligned} \dot{V} &= \dot{V}_1 + \dot{V}_2 = \lambda e \dot{e} + (\alpha \dot{\alpha} + h\theta \dot{\theta}) \\ &= \lambda e u \cos\alpha + \alpha [-\omega + u \frac{\sin\alpha}{e} (\alpha + h\theta)] \end{aligned}$$

It can be easily deduced from the above equation, that the term belonging to  $\dot{V}_1$  can have a negative value, if the linear velocity  $u$  takes the following form:

$$u = (\gamma \cos\alpha)e; \gamma > 0$$

By replacing  $u$  in the above formula,  $\dot{V}_1$  becomes:

$$\dot{V}_1 = -(\lambda \sin^2\alpha)e^2 \leq 0$$

$V_1$  converges asymptotically to a positive defined limit.

Analogue, for  $V_2$ :

$$\dot{V}_2 = \alpha [-\omega + \gamma \frac{\cos\alpha \sin\alpha}{\alpha} (\alpha + h\theta)]$$

Again, it can be easily deduced from the above equation, that the term belonging to  $\dot{V}_2$  can have a negative value, if the angular velocity  $\omega$  takes the following form:

$$\omega = k\alpha + \lambda \frac{\cos\alpha \sin\alpha}{\alpha} (\alpha + h\theta); (k > 0)$$

By replacing  $\omega$  in the above formula,  $\dot{V}_2$  becomes:

$$\dot{V}_2 = k\alpha^2 \leq 0$$

By replacing  $\dot{V}_1$  and  $\dot{V}_2$ , we obtain the final negative form of  $\dot{V}$ :

$$\dot{V} = \dot{V}_1 + \dot{V}_2 = -\lambda(\gamma \cos^2\alpha)e^2 - k\alpha \leq 0$$

The Lyapunov function converges to a positive finite limit. By using the above equations and applying them to the earlier described kinematic model, we get:

$$\begin{cases} \dot{e} = -(\gamma \cos^2\alpha)e \\ \dot{\alpha} = -(k\alpha - \gamma h \frac{\cos\alpha \sin\alpha}{\alpha}), e(0) > 0 \\ \dot{\theta} = \gamma \cos\alpha \sin\alpha \end{cases}$$

Because  $e$  and  $\theta$  converge to 0, it immediately results that  $\dot{e}$  and  $\dot{\theta}$  also converge to 0.

### IV. EXPERIMENTAL RESULTS

To evaluate the performance of the algorithm, we performed several simulations in Matlab 7. The problem to solve is finding and implementing a robust control law based on Lyapunov's equations to move a mobile robot from an arbitrary initial given starting point  $((-1, -1, 0)$  in our example) to a goal point  $(0, 0, 0)$ . After several attempts, we found the following values for  $\gamma$ ,  $h$  and  $k$ , somewhat optimal for the problem at hand:  $\gamma = 1, h = 1$  and  $k = 6$ .

#### A. No induced errors

Figure 2 and 3 represent the control algorithm simulations when the robot has fairly precise odometry information. Figure 2 represents the curvilinear trajectory that the robot takes from its starting point to the  $(0, 0, 0)$  goal point, while figure 3 presents the linear and angular velocities ( $u$  and  $\omega$ ).

To verify the robustness of the control law, we induced perturbations in the odometry information, here simulated by stochastic additive errors.

#### B. Small induced errors

Figure 4 and 5 represent the simulations for small induced errors ( 10%).

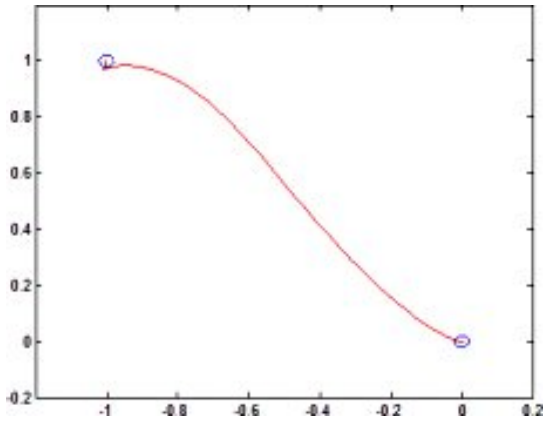


Fig. 2. Trajectory simulation with no errors.

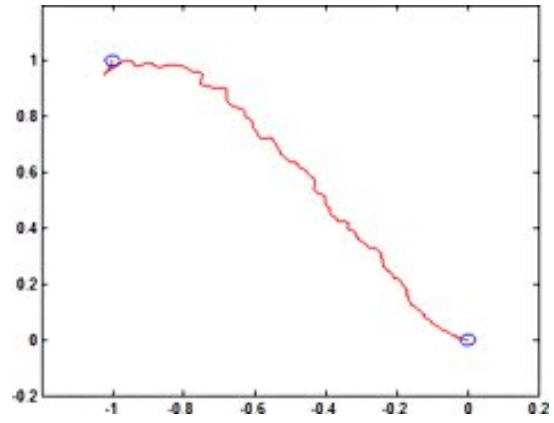


Fig. 4. Trajectory simulation with small odometry errors ( 10%).

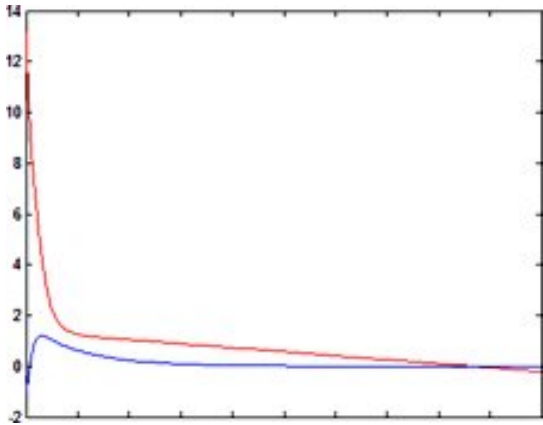


Fig. 3. Linear ( $u$ ) and angular ( $\omega$ ) velocities simulation when no odometry errors are present.

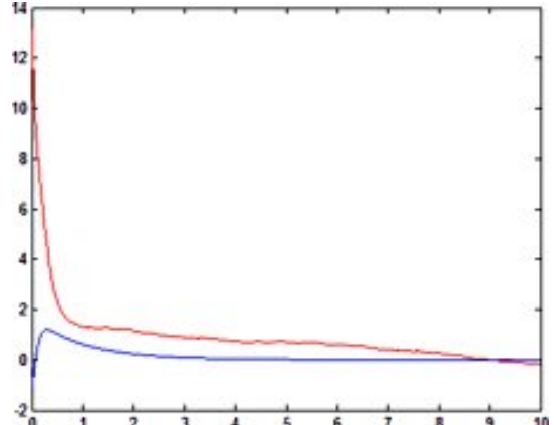


Fig. 5. Linear ( $u$ ) and angular ( $\omega$ ) velocities simulation when small ( 10%) odometry errors are present.

### C. Medium induced errors

Figure 6 and 7 represent the simulations for medium induced errors ( 20%).

### D. Large induced errors

Figure 8 and 9 represent the simulations for large induced errors ( 30%).

Larger odometry errors can be simulated in a similar manner, and will lead to approximately the same results.

## V. CONCLUSIONS

The results presented in this paper, demonstrate the high degree of robustness to perturbations of the Lyapunov closed-loop control law.

Although we did not test the method's efficiency outside the simulator, we presume that it will function correctly on the real robot as well, since the uncertainties modelled are usually higher than the ones on the robot.

## REFERENCES

[1] C. Webers and U.R. Zimmer. Robust tracking under uncertainties. In *OCEANS 2000, Providence, USA*.

[2] M. Aicardi, G. Casalin, A. Bicchi and A. Balestrino. Closed loop steering of unicycle-like vehicle via Lyapunov techniques. In *IEEE Robot. Automat. Mag.*, vol.2, pp.27-35, Mar. 1995

[3] B. Moon Kim and P. Tsiotras. Controllers for Unicycle-Type Wheeled Robots: Theoretical Results and Experimental Validation. In *IEEE Transactions on Robotics and Automation*, vol.18, no.3, June 2002

[4] A. De Luca, G. Oriolo and M. Vendittelli Control of Wheeled Mobile Robots: An Experimental Overview.

[5] W.E. Dixon, D.M. Dawson, E. Zergeroglu and F. Zhang. Robust tracking and regulation control for mobile robots. In *International Journal of Robust and Nonlinear Control, Int.J.Robust Nonlinear Control 2000*, 10:199-216

[6] G. Indiveri. An Introduction to Wheeled Mobile Robot Kinematics and Dynamics. In *Robocamp ? Paderborn (Germany)*, April 8 2002

[7] A. Astolfi, P. Bolzern and A. Locatelli. Path-Tracking of a Tractor-Trailer Vehicle Along Rectilinear and Circular Paths: A Lyapunov-Based Approach. In *IEEE Transactions on Robotics and Automation*, vol.20, no.1, February 2004

[8] G. Oriolo. WMR Control Via Dynamic Feedback Linearization: Design, Implementation and Experimental Validation. In *IEEE Transactions on Control Systems Technology*, vol.10, no.6, November 2002

## VI. SOURCE CODES

```
%-[ main.m ]-
clc;clf;clear;hold off;
plot (0,0);hold on;
axis([-1.2 0.2 -0.2 1.2]);
gamma = 1;
h = 1;
```

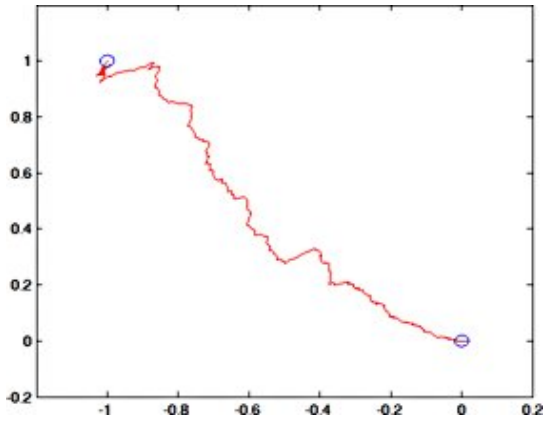


Fig. 6. Trajectory simulation with medium odometry errors ( 20%).

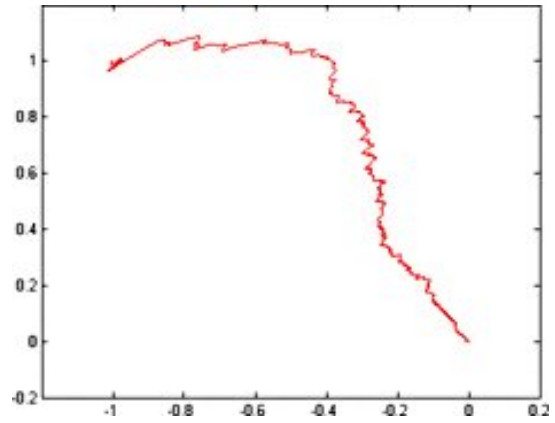


Fig. 8. Trajectory simulation with large odometry errors ( 30%).

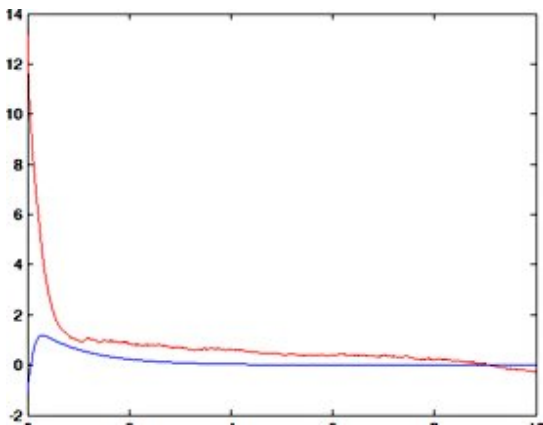


Fig. 7. Linear ( $u$ ) and angular ( $\omega$ ) velocities simulation when medium ( 20%) odometry errors are present.

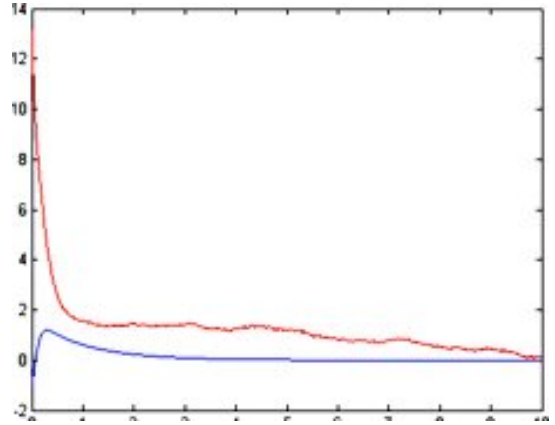


Fig. 9. Linear ( $u$ ) and angular ( $\omega$ ) velocities simulation when large ( 30%) odometry errors are present.

```

k = 6;
[e,th,a] = CalcETA(1,-1,0)
[t,y]=ode45(@f3,[0 10],[e a th]);
x = zeros(length(t));
yy = zeros(length(t));
for i = 1:length(t)
    x(i) = sqrt( y(i,1)^2 / (1+tan(y(i,3))^2) );
    yy(i) = x(i) * tan(y(i,3));
    omega(i) = k*y(i,2) + gama *
        (cos(y(i,2))*sin(y(i,2)) / y(i,2))*(y(i,2) + h * y(i,3));
    u(i) = gama*cos(y(i,2))*y(i,1);
    e = y(i,1);
    theta = y(i,3);
end;
plot(yy,x,'r');
[latc,longc] = scircle1(-1,1,0.02);
plot(latc,longc,'b')
[latc,longc] = scircle1(0,0,0.02);
plot(latc,longc,'b')

pause;
hold off;
plot(t,omega,'r',t,u,'b');
    
```

```

%-[ calcETA.m ]-
function [e,theta,alfa] = CalcETA(x,y,fi)
e = sqrt(x^2 + y^2);
theta = atan2(-y,-x);
alfa = theta - fi;

%-[ f3.m ]-
function dx = f3(t,y)
gama = 1;
h = 1;
k = 6;
dx=[-(gama*cos(y(2))^2*y(1));
-k*y(2) - gama*h*(cos(y(2))*sin(y(2))/y(2));
gama * cos(y(2))*sin(y(2))];
    
```