INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Theory and Experimental Investigation of Tracking Control of Wheeled Mobile Robots

Lin Ke Wang

A Thesis

in

The Department

of

Mechanical and Industrial Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

March 2002

ABSTRACT

Theory and Experimental Investigation of Tracking Control of Wheeled Mobile Robot

Lin Ke Wang

A variety of theoretical and tracking control problems have been studied for various classes of nonholonomic wheeled mobile robot (WMR). The relative difficulty of the tracking control problem depends not only on nature structural properties of the nonholonomic WMR system but also on the tracking control objective.

In this thesis several structural properties regarding controllability, stabilizability and nonholonomy of the kinematic state models of WMRs have been given, taking into account the restriction to robot mobility induced by the constraints. By using the concepts of *degree of mobility* and *degree of steerability*, this study provides a general and unifying presentation of the modeling issue of WMRs. For particular prototypes of WMRs, the posture kinematic models for WMRs are available in the literature. However, in this thesis, a more general viewpoint is adopted for the consideration of a general class of WMRs with an arbitrary number of wheels of different types. Three-wheel mobile robot-Nomad 200 used in the experiment is described in details.

The control objective in this investigation is to determine a control law so that the robot can track a desired trajectory. Two control algorithms are therefore presented. The first control algorithm is using a "virtual vehicle approach", which is shown to be robust with respect to error and disturbances; the second one is a time-varying adaptive control algorithm developed for a general class of nonholonomic mechanical system, which guarantees the global stability of the closed-loop system.

Simulation results with SIMULINK software and Nomad simulator confirmed the validity of the proposed algorithms, and the implementation of the both algorithms on a nonholonomic WMR, Nomad 200 platform, clearly, verify that proposed tracking control algorithms can achieve the control objective in a stable and robust way.

Acknowledgement

I would like to thank Dr. Chun-Yi Su for his professional supervision, guidance, and patience, throughout this study. It really has been a pleasure to work with him and learn from him, especially in the period of my research and my thesis preparation. I would also like to thank Dr. K. Demirli for his supervision, experimental device supply, technical support, understanding, and assistance during the experimentation.

I wish to thank Dr. C. K. Jen for his encouragement, timely help, useful comments and suggestion during my thesis preparation. I also wish to thank Mr. Mohammad Molhim who always helped me to maintain Nserver simulation system and gave me valuable advice during my experimentation.

Finally, I would like to express my special thanks to my wife Li You and other family members for their love, encouragement and support.

Table of Contents

List of Figures

List of Tables

Nomenclature

Latin Symbols

A	The center of the fixed
b	An arbitrary inertial base frame
c	Castor wheel
d	The "look-ahead" distance
$e_1 \cdot e_2 \cdot e_3$	Tracking errors
f	Fixed wheel
$k_0 \cdot k_1 \cdot k_2 \cdot k_3$	Control gains
L	The distance from the center of the robot to the fixed wheel
l	The distance from *A to P*
m	The moving frame, which is attached to the mobile robot.
N_c	The numbers of Castor wheels
N_f	The numbers of fixed wheels
Ns	The numbers of steering wheels
N_s	The numbers of Swedish wheels
O	The origin point of the base frame
P	The origin of the moving frame, the center of the robot
q	Posture matrix of the robot

q_d	Matrix of desired trajectory
$R^T(\theta)$	Matrix of the robot system
r	The radius of the wheel
s	Steering wheel
sw	Swedish wheel
T	Sample time
u, v	Matrix of velocity
x, y	The coordinates of the moving frame
Xb, Yb	The coordinates of the base frame
Xm, Ym	The coordinates of the moving frame

Greek Symbols

α	The angle between the base frame coordinates Xb and the distance l of A from P
β	The orientation of the plane of the wheel with respect to l, and the rotation of the rod with respect to the cart
$\delta_f(t)$	Steering angle
δ_m	The degree of mobility
δ_s	The degree of steerability
φ	The yaw angle (orientation of the car)
φ_d	The desired orientation

γ	The angler of the zero component of the velocity of the contact point on the wheel plane
$\eta_1 \cdot \eta_2 \cdot \eta_3$	The robot velocity components along $X_m \cdot Y_m$, and the angular velocity, respectively
θ	The orientation angle of the moving frame

Chapter 1

Introduction

1.1 Mobile Robots

Robotics is useful, attractive and vast field, which combines many different disciplines. These include computer science, electrical and electronic engineering, mathematics, mechanical engineering, and structure design, etc. Mobile robots form one important branch, and share several characteristics and difficulties with other kinds of robots, such as industrial robot manipulators. But they also have some important problems concerns related to particularly move around and perform practical task. For example, industrial robot manipulators, which are now widely used in car factories, work in well-structured and closely controlled environments, apart from being stationary. Very little disturbance happens in these environments that is not out of a direct consequence of actions of the robot. There is also very little variation in what happens, how it happens, and in where things are or what the robot has to manipulate and how. We categorize these kinds of robot environments as *well-structured controlled environments*. They are necessary for industrial robots to work properly.

The interest in investigating and developing mobile robots has been largely motivated by both the need and desire to have robots that can work with and be useful for people in their normal work or day-to-day environment: in offices, hospitals, museums and galleries, libraries, supermarkets and shopping centers, sports centers, exhibition centers, airports, railway stations, universities and schools, etc. and one day, in our homes too. All these are, however, examples of a very different kind of environment from the one industrial robots work in. They are structured: they are all designed and built for us to live, work, and play in. This structure of our every day environments is not, however, designed specifically for robots and nor would we want them to be. It is also generally unacceptable to people to retrain their work or living places so as to make it possible for robots to operate in them.

Mobile robots are mechanical devices that are equipped with an on-board power source, computational resources, sensors and actuators. The robots' ability to move autonomously and freely can be regarded as a gain or a loss. As a gain, the robots can be used for tasks that require movement (e.g. transportation, surveillance, inspection and cleaning task), and can position themselves optimally for their operation. They are therefore uniquely suited to large-area operation in environments that are inaccessible or dangerous to humans. On the other hand, as a loss, the autonomous movement in semi-structured environments, i.e. environments that have not been specially prepared for robot operation, can produce unexpected events, fluctuations and uncertainties. Control algorithms for autonomous mobile robots need to take the above-mentioned parameters together with noise, unpredictability and variation into account.

Mobile robots are preferred to be able to operate in our everyday environments, with all the normal variations and uncertainties that are characteristics of the laces in which we work, live, and play. These environments and many activities are going on, while the robot functions. We categorize these kinds of environments semi-structured uncontrolled environments, so as to distinguish them from that industrial robots must operate in.

Getting around and doing things in semi-structured uncontrolled environments is typically not something we find difficulty to do. Unless we are confined to a wheelchair, or suffer some kind of mobility or perceptual impairment. For mobile robots, however, getting around and doing things in real environments remain a big challenge in general. This requires a kind of intelligence that industrial robots, and other robots that work in well-structured controlled environments, do not have, and need.

1.2 Control Problems for Nonholonomic Systems

Considering WMR can be described as a class of mechanical system with nonholonomic dynamic of mobile robots constraints in the literature, a variety of theoretical and applied control problems have been studied for various classes of nonholonomic control systems. The relative difficulty depends not only on nonholonomic nature of the system but also on the control objective. For some control objectives, classical non-linear control approaches (e.g., feedback linearization and certain dynamic inversion, as developed in [35]) are effective. Examples of such

control objectives include stabilization to a suitably defined manifold that contains the equilibrium manifold [4, 5, 36, 37, 38], stabilization to certain trajectories [42], dynamic path following [41], and output tracking [34, 39]. Consequently, there are classes of control problems for nonholonomic systems for which standard nonlinear control methods can be applied.

However, many of the mostly common control objectives, e.g., motion planning and stabilization to equilibrium state, can not be solved using the standard nonlinear control methods. Therefore new approaches have been developed. Substantial research has been devoted to motion planning, e.g., the study of (open loop) controls that transfer the system from a specified initial state to a specified final state. Conditions have been developed that guarantee when motion-planning problems have solutions, and a variety of construction procedures for determining such controls have been proposed. In addition, feedback control of nonholonomic systems has been studied where the objective was to accomplish specified closed-loop performance objectives, including the classical control objectives of stabilization, asymptotic tracking, disturbance rejection, robustness improvement, etc.

In this thesis we will study the modeling and control of WMR. A WMR is a wheeled vehicle, which is capable of an autonomous motion (without external human driver) because it is equipped, for its motion, with actuators that are driven by an embarked computer.

1.3 Outline of the Thesis

In Chapter 2, the modeling and structural properties of a wheeled mobile robot, especially a three-wheel robot, is reviewed. In Chapter 3, posture kinematic model, mobility, steerability and maneuverability, are reviewed. In Chapter 4, we introduced background information of Nomad 200 mobile robot that is used in the experiment. In Chapter 5, a virtual vehicle approach is presented in solving the wheeled mobile robots tracking problem. In Chapter 6, an adaptive tracking control algorithm is presented. The conclusions and future work are presented in Chapter 6.

Chapter 2

Modeling and Structural Properties

The aim of this chapter provides a general and unifying presentation of the modeling issue of wheeled mobile robots. Several examples of derivation of kinematic models for these robots are available in the literature for particular prototypes. Here, a more general viewpoint is adopted and a general class of WMR with an arbitrary number of wheels of different types and actuation is considered. We will point out the structural properties of the kinematic models, taking into account the restriction to robot mobility induced by the constraints. By using the concepts of *degree of mobility* and *degree of steerability* [43,44], notwithstanding the variety of possible robot constructions and wheel configurations, the set of WMR can be partitioned into five classes.

Throughout this thesis, we will use posture kinematic model to describe the behavior of WMRs [43, 44]. The *posture kinematic model* is the simplest state space model able to give a global description of WMRs. It can be shown that within each of

five classes. this model has a particular generic structure. which allows the understanding of the maneuverability properties of the robot.

2.1 Robot Description

Without the loss of generality and keeping the mathematical derivation as simple as possible. we will assume that the WMRs under study are made up a rigid cart equipped with *non-deformable wheels* and moving in a horizontal plane. The position of the robot in the plane is shown in Figure 2.1 described as follows. An arbitrary inertial base frame b is fixed in the plane of motion. Xb and Yb are the coordinates of base frame. Xm and Ym are the coordinates of frame m. which is attached to the mobile robot.

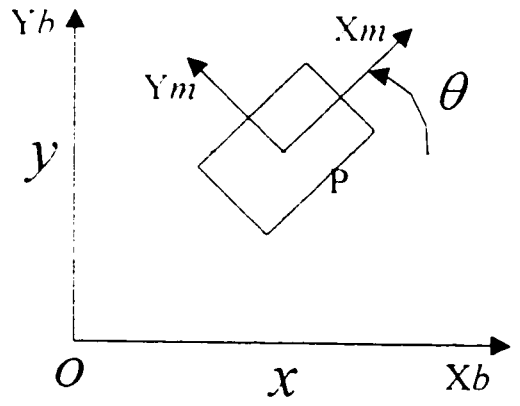

Figure 2.1. Posture coordinates.

The robot posture can be described in terms of the two coordinates x, y of the origin P of the moving frame and by the orientation angle θ of the moving frame, both with respect to the base frame with the origin of the coordinates at O. Hence, the robot posture can be given by the (3x1) vector.

$$\xi = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix}. \tag{2.1}$$

And the rotation matrix expressing the orientation of the base frame with respect to the moving frame is

$$R(\theta) = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.2}$$

We assume that, during motion, the plane of each wheel remains vertical and the wheel rotates about its (horizontal) axle whose orientation with respect to the cart can be fixed or varying. Two basic classes of idealized wheels: namely, the *conventional wheels* and the *Swedish wheels* are adapted in the literature. In each case, it is assume that the contact between the wheel and the ground is reduced to a single point of the plane.

For a *conventional wheel*, the contact between the wheel and the ground is supposed to satisfy both conditions of *pure rolling* and *non-slipping* along the motion. This means that the velocity of the contact point is equal to zero. This implies that the two components, respectively parallel to the plane of the wheel and orthogonal to this plane, of this velocity are equal to zero. For a *Swedish wheel*, only one component of velocity of the contact point of the wheel with the ground is supposed to be zero along the motion. The direction of this zero component of velocity is a priori arbitrary but fixed with respect to the orientation of the wheel. Below the expressions of the constrains for conventional and Swedish wheels will be listed. For all the details, please refer to [43, 44]

2.1.1 Conventional Wheels

In general, conventional wheels include fixed wheels, steering wheels and Castor wheels.

2.1.1.1 Fixed Wheel

The center of the fixed wheel, denoted by A, is a fixed point of the cart as shown in Figure 2.2. The position of A in the moving frame is characterized using polar coordinates, i.e., the distance l of A from P and the angle α. The orientation of the plane of the wheel with respect to l is represented by the constant angle β. The rotation angle of the wheel about its (horizontal) axle is denoted by φ and the radius of the wheel by r.

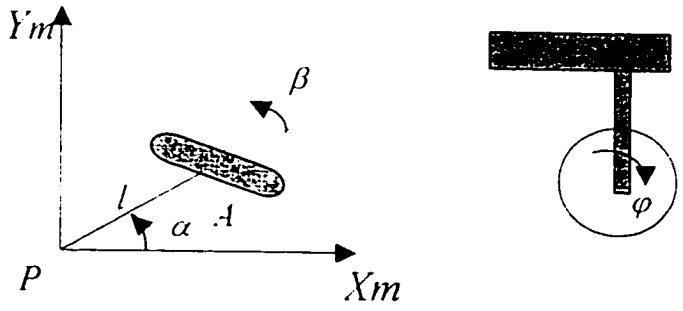

Figure 2.2. Fixed wheel or steering wheel

The position of the wheel is thus characterized by four constants: α, β, l, r and its motion by a time varying angle $\varphi(t)$. With this description, the components of the velocity of the point are easily computed and the two following constraints can be deduced:

- On the wheel plane.

$$\begin{pmatrix} -\sin(\alpha+\beta) & \cos(\alpha+\beta) & l\cos\beta \end{pmatrix} R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.3}$$

- Orthogonal to the wheel plane.

$$\begin{pmatrix} \cos(\alpha+\beta) & \sin(\alpha+\beta) & l\sin\beta \end{pmatrix} R(\theta)\dot{\xi} = 0 \tag{2.4}$$

2.1.1.2 Steering Wheel

A steering wheel is such that the motion of the wheel plane with respect to the cart is a rotation about a vertical axle passing through the center of the wheel as shown in Figure 2.2. The description of the position is the same as for a fixed wheel, except that now the angle β is not constant but time varying. The position of the wheel is characterized by three constants: α, l, r and its motion with respect to the cart by two time varying angle $\beta(t)$ and $\varphi(t)$. The constants have the same form as above, i.e.,

$$\begin{pmatrix} -\sin(\alpha+\beta) & \cos(\alpha+\beta) & l\cos\beta \end{pmatrix} R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.5}$$

$$\begin{pmatrix} \cos(\alpha+\beta) & \sin(\alpha+\beta) & l\sin\beta \end{pmatrix} R(\theta)\dot{\xi} = 0 \tag{2.6}$$

2.1.1.3 Castor Wheel

A *Castor wheel* is a wheel which is orientable with respect to the cart, but the rotation of the wheel plane is with respect to a vertical axle which does not pass through the center of the wheel as shown in Figure 2.3. In this case, the description of the wheel configuration requires more parameters.

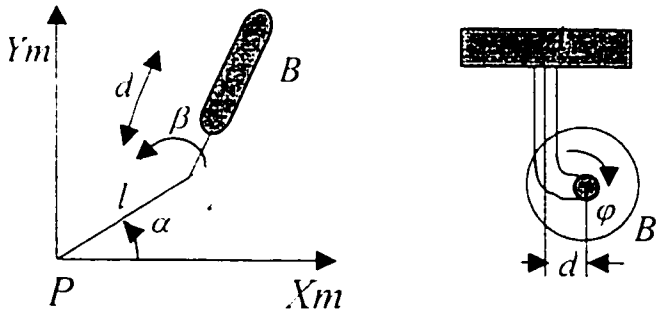

Figure 2.3. Castor wheel.

The center of the wheel is now denoted by B and is connected to the cart by a rigid rod from A to B of constant length d, which can rotate about a fixed vertical axle at point A. This point A is itself a fixed point of the cart and its position is specified by the two polar coordinates l and α as above. The rotation of the rod with respect to the cart is represented by the angle β and the plane of the wheel is aligned with d.

The position of the wheel is described by four parameters: α, l, r, d while its motion by two time-varying angles $\beta(t)$ and $\varphi(t)$. With these notations, the constraints have the following form:

$$\begin{pmatrix} -\sin(\alpha+\beta) & \cos(\alpha+\beta) & l\cos\beta \end{pmatrix} R(\theta)\dot{\xi} + r\dot{\varphi} = 0 \tag{2.7}$$

$$\begin{pmatrix} \cos(\alpha+\beta) & \sin(\alpha+\beta) & d+l\sin\beta \end{pmatrix} R(\theta)\dot{\xi} + d\dot{\beta} = 0 \tag{2.8}$$

2.1.2 Swedish Wheel

The position of the *Swedish wheel* with respect to the cart is described, as for the fixed wheel, by three constant parameters: α, β, and l. An additional parameter is required to characterize the direction, with respect to the wheel plane, of the zero component of the velocity of the contact point represented by the angler γ as shown in Figure 2.4.

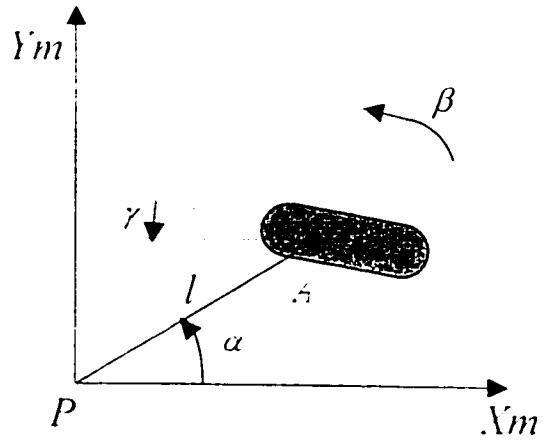

Figure 2.4. Swedish wheel.

The motion constraint is expressed as

$$\begin{pmatrix} -\sin(\alpha+\beta+\gamma) & \cos(\alpha+\beta+\gamma) & l\cos(\beta+\gamma) \end{pmatrix} R(\theta)\dot{\xi} + r\cos\gamma\dot{\varphi} = 0 \qquad (2.9)$$

2.2 Restrictions on Robot Mobility

We now consider a general mobile robot, equipped with N number of wheels of the four above-described classes. We use the four following subscripts to identify quantities relative to the four classes: *f* for fixed, *s* for steering, *c* for castor and *sw* for Swedish wheels. The numbers of the wheels of each type are denoted by N_f, N_s, N_c, N_{sw} with $N_f+N_s+N_c+N_{sw}$ N. The configuration of the robot is fully described by the following coordinate vectors, respectively:

- *Posture coordinates* $\xi(t) = (x(t) \quad y(t) \quad \theta(t))^T$ for the position in the plane;
- *Orientation coordinates* $\beta(t) = (\beta_s^T(t) \quad \beta_c^T(t))^T$ for the orientation angles of the steering and castor wheels, respectively;
- *Rotation coordinates* $\varphi(t) = (\varphi_f(t) \quad \varphi_s(t) \quad \varphi_c(t) \quad \varphi_{sw}(t))^T$ for the rotation angles of the wheels with respect to their horizontal axle of the rotation.

The whole set of posture, orientation and rotation coordinates $\xi, \beta_s, \beta_c, \varphi$ is termed the set of configuration coordinates in the sequel. The total number of the configuration coordinates is clearly

$Nf+2Ns+2Nc+Nsw+3$.

With the above notations, the constraints can be written in the general matrix form:

$$J_1(\beta_s, \beta_c)R(\theta)\dot{\xi} + J_2\dot{\varphi} = 0 \tag{2.10}$$

$$C_1(\beta_s, \beta_c)R(\theta)\dot{\xi} + C_2\dot{\beta}_c = 0 \tag{2.11}$$

In (2.10), it is

$$J_1(\beta_s, \beta_c) = \begin{pmatrix} J_{1f} \\ J_{1s}(\beta_s) \\ J_{1c}(\beta_c) \\ J_{1sw} \end{pmatrix}$$

where J_{1f}, J_{1s}, J_{1c} and J_{1sw} are respectively $(N_f \times 3)$, $(N_s \times 3)$, $(N_c \times 3)$ and $(N_{sw} \times 3)$ matrices, whose forms were derived directly from the constraints (2.3), (2.5), (2.7) and (2.9), respectively. In particular, J_{1f} and J_{1sw} are constant, while J_{1s} and J_{1c} are time varying, respectively with regard to $\beta_s(t)$ and $\beta_c(t)$. Further, J_2 is a constant $(N \times N)$ matrix whose diagonal entries are the radii of the wheels, except for the radii of the Swedish wheels which are multiplied by $\cos\gamma$.

On the other hand, in equation (2.11), it is

$$C_1(\beta_s, \beta_c) = \begin{pmatrix} C_{1f} \\ C_{1s}(\beta_s) \\ C_{1c}(\beta_c) \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0 \\ 0 \\ C_{2c} \end{pmatrix}$$

where C_{1f}, C_{1s}, and C_{1c} are three matrices, respectively, of dimensions $(N_f \times 3)$, $(N_s \times 3)$, and $(N_c \times 3)$, whose were derived from the non-slipping constraints (2.4), (2.6), and (2.8), respectively. In particular, C_{1f} is constant while C_{1s}, and C_{1c} are time-varying. Further, C_{2c} is a diagonal matrix whose diagonal entries are equal to $d\sin\gamma$ for the N_c Castor wheels.

We introduce the following assumption concerning the configuration of the Swedish wheels.

Assumption 2.1 For each Swedish wheel $\gamma \neq \pi/2$.

The value $\gamma = \pi/2$ would correspond to the direction of the zero component of the velocity being orthogonal to the plane of the wheel. Such a wheel would be subject to a constraint identical to the non-slipping constraint of conventional wheels, hence loosing the benefit of implementing a Swedish wheel.

Consider now the first (N_1+N_2) non-slipping constraints from (2.11) and they can be written explicitly as

$$C_{1f} R(\theta)\dot{\xi} = 0 \tag{2.12}$$

$$C_{1s}(\beta_s)R(\theta)\dot{\xi} = 0 \tag{2.13}$$

These constraints imply that the vector $R(\theta)\dot{\xi} \in N(C_1^*(\beta_s))$ where

$$C_1^*(\beta_s) = \begin{pmatrix} C_{1f} \\ C_{1s}(\beta_s) \end{pmatrix} \tag{2.14}$$

Obviously, it is rank $(C_1^*(\beta_s)) \leq 3$. If rank $(C_1^*(\beta_s))=3$, then $R(\theta)\dot{\xi} = 0$ and any motion in the plane is not possible. More generally, restrictions on robot mobility are related to the rank of C_1^*.

Before that, it is worth noticing that coordinates (2.12) and (2.13) have an interesting geometrical interpretation. At each time instant, the motion of the robot can be viewed as an instantaneous rotation about the *instantaneous center of rotation* (ICR)

whose position with respect to the cart can be time varying. Hence, at each time instant, the velocity vector of any point of the cart is orthogonal to the straight line joining this point and the ICR. In particular, this is true for the centers of the fixed and steering wheels. This implies that, at each time instant, the horizontal axles of all the fixed and steering wheels intersect at the ICR. This fact is illustrated in Figure 2.5 and is equivalent to the condition that rank $(C_1^*(\beta_s)) \leq 2$.

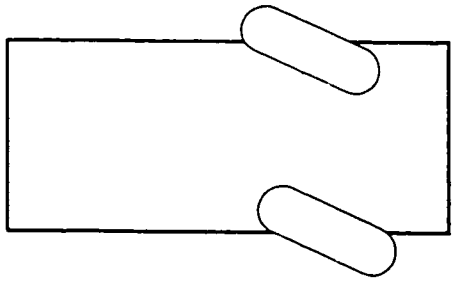

Figure 2.5 Instantaneous center of rotation.

Clearly, the rank of matrix $C_1^*(\beta_s)$ depends on the design of the mobile robot. We define the *degree of mobility* δ_m of a mobile robot as

$$\delta_m = \dim(N(C_1^*(\beta_s))) = 3 - rank(C_1^*(\beta_s))$$

Let us now examine the case rank $(C_{1f}) = 2$. It implies that the robot has at least two fixed wheels and, if there are more than two, their axles intersect at the ICR whose position with respect to the cart is *fixed*. In such case, it is clear that the only possible motion is a rotation of the robot about a fixed ICR. Obviously, this is not acceptable in practice and thus we assume that rank $C_{1f} \leq 1$. Furthermore we assume that the robot structure is *non- degenerate* in the following sense.

Assumption 2.2 A mobile robot is non-degenerate if

Rank $(C_{1f}) \leq 1$ rank $(C_1^*(\beta_s))$ =rank (C_{1f}) +rank $(C_{1s}(\beta_s)) \leq 2$.

This assumption is equivalent to the following conditions.

- If the robot has more than one fixed wheel ($N_f > 1$), then they are all on a single common axle.
- The center of the steering wheels does not belong to this common axle of the fixed wheels.
- The number rank($C_{1s}(\beta_s)$) ≤ 2 is the number of steering wheels that can be oriented independently in order to steer the robot. We name this number the *degree of steerability*

$$\delta_s = \text{rank}(C_{1s}(\beta_s)).$$

The number of steering wheels N_s and their type are obviously a privilege of the robot designer. If a robot is equipped with more than δ_s steering wheels ($N_s > \delta_s$), the motion of the extra wheels must be coordinated to guarantee the existence of the ICR at each time instant.

It follows that only *five* nonsingular structures are of practical interest, which can be inferred by the following conditions.

- The degree of mobility δ_m satisfies the inequality

$$1 \le \delta_m \le 3; \tag{2.15}$$

The upper bound is obvious, while the lower bound means that we consider only the case where a motion is possible, i.e., $\delta_m \neq 0$.

- The degree of steerability δ_s satisfies the inequality

$$0 \le \delta_s \le 2; \tag{2.16}$$

The upper bound can be reached only for robots without fixed wheels ($N_f = 0$), while the lower bound corresponds to robots without steering wheels ($N_s = 0$).

- The following inequality is satisfied:

$$2 \le \delta_m + \delta_s \le 3; \tag{2.17}$$

The case $\delta_m + \delta_s = 1$ is not acceptable because it corresponds to the rotation of the robots about a *fixed* ICR as we have seen above. The cases $\delta_m \geq 2$ and $\delta_s = 2$ are excluded because, according to **Assumption 2.2**, $\delta_s = 2$ implies $\delta_m = 1$.

Table2.1 Degree of mobility and degree of steerability for possible WMRs.

δ_m	3	2	2	1	1
δ_s	0	0	1	1	2

Therefore, there exist only five types of wheeled mobile robots, corresponding to the five pairs of values of δ_m and δ_s that satisfy inequalities (2.15), (2.16), (2.17) according to Table 2.1.

In the sequel, each type of structure will be designated by using a denomination of the form "Type (δ_m, δ_s) robot." The main design characteristics of each type of mobile robot are now briefly presented [43,44].

2.2.1 Type (3,0) Robot

In this case it is

$$\delta_m = \dim(N(C_1^*(\beta_s))) = 3 \qquad \delta_s = 0.$$

These robots have *no* fixed wheels ($N_f = 0$) and *no* steering wheels ($N_s = 0$). They only have castor and /or Swedish wheels. Such robots are called omnidirectional because they have a full mobility in the plane, which means they can move at each time

instant in any direction without any reorientation. In contrast, the other four types of robots have a restricted mobility (degree of mobility less than three).

2.2.2 Type (2,0) Robot

In this case it is

$$\delta_m = \dim(N(C_1^*(\beta_s))) = \dim(N(C_{1f})) = 2 \qquad \delta_s = 0.$$

These robots have *no* steering wheels (N_s =0). They have either one or several fixed wheels but with a single common axle, otherwise rank (C_{1f}) would be greater than one. The mobility of the robot is restricted in the sense that, for any admissible trajectory $\xi(t)$, the velocity $\dot{\xi}(t)$ is constrained to belong to the two-dimensional distribution spanned by the vector fields $R^T(\theta)s1$ and $R^T(\theta)s2$, where $s1$ and $s2$ are two constant vectors spanning $N(C_{1f})$.

2.2.3 Type (2,1) Robot

In this case it is

$$\delta_m = \dim(N(C_1^*(\beta_s))) = \dim(N(C_{1s}(\beta_s))) = 2 \qquad \delta_s = 1.$$

These robots have *no* fixed wheels (N_f =0), and at least *one* steering wheel ($N_s \geq 1$). If there is more than one steering wheel, their orientations must be coordinated in such a way that rank $(C_{1s}(\beta_s)) = \delta_s = 1$. The velocity $\dot{\xi}(t)$ is constrained to belong to two-dimension distribution spanned by the velocity fields $R^T(\theta)s_1(\beta_s)$ and

$R^T(\theta)s_2(\beta_s)$, where $s_1(\beta_s)$ and $s_2(\beta_s)$, are two vectors spanning $N(C_{1s}(\beta_s))$ and parameterized by the angle β_s of one arbitrarily chosen steering wheel.

2.2.4 Type (1,1) Robot

In this case it is

$$\delta_m = \dim(N(C_1^T(\beta_s))) = 1 \qquad \delta_s = 1.$$

These robots have one or several fixed wheels with a single common axle. They also have one or several steering wheels, with the condition that the center of one of them is not located on the axle of the fixed wheels-otherwise the structure would be singular, and that their orientations are coordinated in such a way that rank $(C_{1s}(\beta_s)) = \delta_s = 1$. The velocity $\xi(t)$ is constrained to belong to a one-dimensional distribution parameterized by the orientation angle of one arbitrarily chosen steering wheel. Mobile robots that are built on the model of a conventional car (often called car-like robots) belong to this class.

2.2.5 Type (1,2) Robot

In this case it is

$$\delta_m = \dim(N(C_1^T(\beta_s))) = \dim(N(C_{1s}(\beta_s))) = 1 \qquad \delta_s = 2.$$

These robots have no fixed wheels($N_f = 0$). They have at least two steering wheels ($N_s \geq 2$). If there are more than two steering wheels, their orientations must be coordinated in such a way that rank $(C_1(\beta_s)) = \delta_s = 2$. the velocity $\dot{\xi}(t)$ is constrained to belong to a one-dimensional distribution parameterized by the orientation angles of two arbitrarily chosen steering wheels of the robot.

In order to avoid useless notational complications, we will assume from now on that the degree of steerability is precisely equal to the number of steering wheels, i.e.,

$$\delta_s = N_s.$$

This is certainly a big restriction from a robot design viewpoint. However, for the mathematical analysis of the behavior of mobile robots, there is no loss of generality in this assumption, although it is considerably simplifies the technical derivation. Indeed, for robots having an excess of steering wheels ($\delta_s < N_s$), it is always possible by appropriate (but possible tedious) manipulation to reduce the constraints (2.13) to a minimal subset of exactly δ_s independent constraints, and to ignore the other slave steering wheels in the analysis. These constraints correspond to the wheels δ_s that have been selected as the master steering wheels of the robot (see independent comment after **Assumption 2.2**)

2.3 Three Wheel Robots

Now, we present six practical examples of mobile robots to illustrate the *five* types of structures that have been presented above. We consider of robots with *three* wheels.

As we have shown in section 2.1, the wheels of a mobile robot are described by (at most) six characteristic constants: namely, three angles α, β, γ, and three lengths l, r, d. For each example, we provide successively a table with the numerical values of these characteristic constants and a presentation of various matrices J and C involved in the mathematical expressions (2.10) and (2.11) of the constraints.

However, we will assume that the radii r and the distance d are identical for all the wheels of all the examples. Hence, we will specify only the values of α, β, r, l.

2.3.1 Type (3,0) Robot with Swedish Wheels

The robot has three Swedish wheels located at the vertices of the cart that has the form of an equilateral triangle as shown in Figure 2.6. The characteristic constants are specified in Table 2.2.

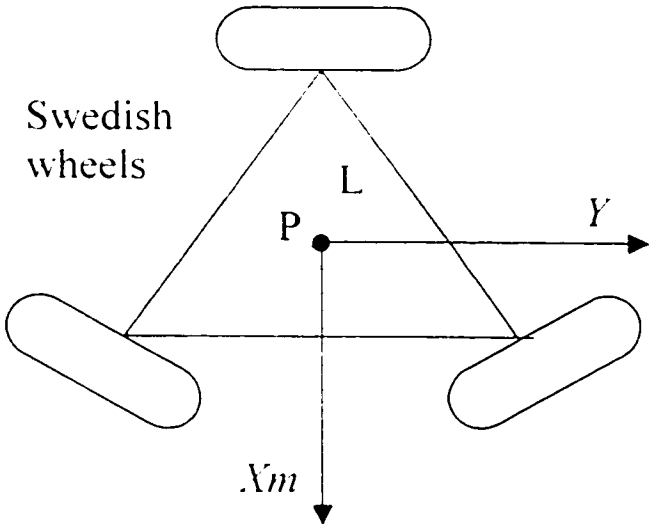

Figure 2.6 Type (3,0) robot with Swedish wheels.

Table 2.2: Characteristic constants of Type (3,0) robot with Swedish wheels.

wheels	α	β	γ	l
1sw	$\pi/3$	0	0	L
2sw	π	0	0	L
3sw	$5\pi/3$	0	0	L

The constraints have the form (2.10) where:

$$J_1 = J_{1sw} = \begin{pmatrix} -\sqrt{3}/2 & 1/2 & L \\ 0 & -1 & L \\ \sqrt{3}/2 & 1/2 & L \end{pmatrix}$$

$$J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}.$$

2.3.2 Type (3,0) Robot with Castor Wheels

The robot has three castor wheels as shown in Figure 2.7. The characteristic constants are specified in Table 2.3.

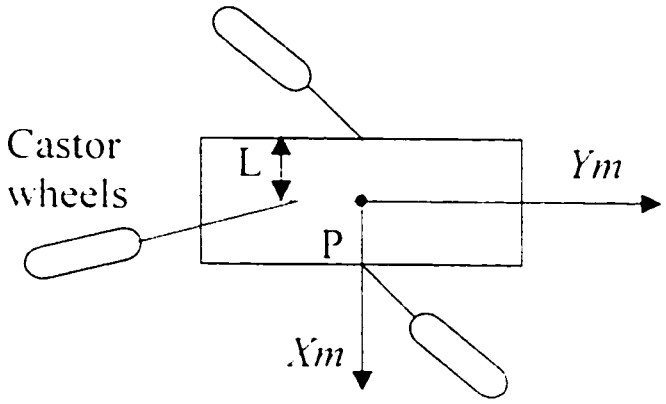

Figure 2.7 Type (3,0) robot with Castor wheels.

Table 2.3: Characteristic constants of Type (3,0) robot with castor wheels.

wheels	α	β	l
1c	0	-	L
2c	π	-	L
3c	$3\pi/2$	-	L

The constraints have the form (2.10) and (2.11) where:

$$J_1 = J_{1c}(\beta_c) = \begin{pmatrix} -\sin\beta_{c1} & \cos\beta_{c1} & L\cos\beta_{c1} \\ \sin\beta_{c2} & -\cos\beta_{c2} & L\cos\beta_{c2} \\ \cos\beta_{c3} & \sin\beta_{c3} & L\cos\beta_{c3} \end{pmatrix}$$

$$J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}$$

$$C_1 = C_{1c}(\beta) = \begin{pmatrix} \cos\beta_{c1} & \sin\beta_{c1} & d + L\sin\beta_{c1} \\ -\cos\beta_{c2} & -\sin\beta_{c2} & d + L\sin\beta_{c2} \\ \sin\beta_{c3} & -\cos\beta_{c3} & d + L\sin\beta_{c3} \end{pmatrix}$$

$$C_2 = C_{2c} = \begin{pmatrix} d & 0 & 0 \\ 0 & d & 0 \\ 0 & 0 & d \end{pmatrix}$$

2.3.3 Type (2,0) Robot

The robot has two fixed wheels on the same axle and one castor wheel as shown in Figure 2.8, and it is typically referred to as the *unicycle robot*. The characteristic constants are specified in Table 2.4.

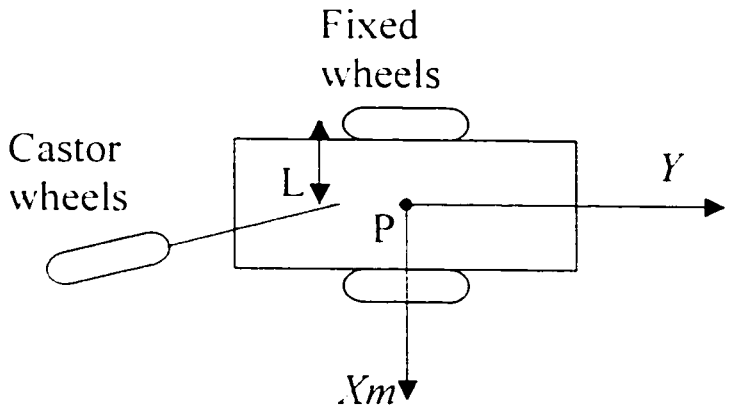

Figure 2.8 Type (2.0) robot.

Table 2.4: Characteristic constants of Type (2.0) robot with castor wheels.

wheels	α	β	l
1f	0	0	L
2f	π	0	L
3c	$3\pi/2$	-	L

The constraints have the form (2.10) and (2.11) where:

$$J_1 = \begin{pmatrix} J_{1f} \\ J_{1c}(\beta_{c3}) \end{pmatrix} = \begin{pmatrix} 0 & 1 & L \\ 0 & -1 & L \\ \cos\beta_{c3} & \sin\beta_{c3} & L\sin\beta_{c3} \end{pmatrix}$$

$$J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}$$

$$C_1 = \begin{pmatrix} C_{1f} \\ C_{1s}(\beta_{s3}) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ \sin\beta_{s3} & -\cos\beta_{s3} & d + L\sin\beta_{s3} \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 0 \\ C_{2s} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix}$$

We note that the non-slipping constraints of the two fixed wheels are equivalent (See the first two rows of C_1); hence, the matrix C_1^* has rank equal to one as expected.

2.3.4 Type (2,1) Robot

The robot has one steering wheel and two castor wheels as shown in Figure 2.9. The characteristic constants are specified in Table 2.5.

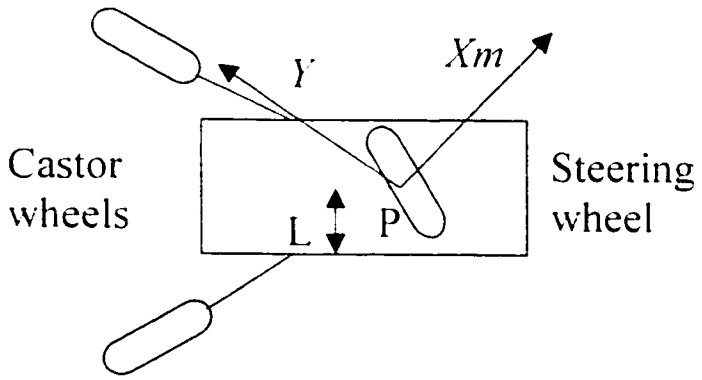

Figure 2.9 Type (2,1) robot.

Table 2.5: Characteristic constants of Type (2,1) robot.

wheels	α	β	l
1s	0	-	L
2c	$\pi/2$	-	$L\sqrt{2}$
3c	π	-	$L\sqrt{2}$

The constraints have the form (2.10) and (2.11) where:

$$J_1 = \begin{pmatrix} J_{1s}(\beta_{s1}) \\ J_{1c}(\beta_{c2}, \beta_{c3}) \end{pmatrix} = \begin{pmatrix} -\sin\beta_{s1} & \cos\beta_{s1} & 0 \\ -\cos\beta_{c2} & -\sin\beta_{c2} & \sqrt{2}L\cos\beta_{c2} \\ \sin\beta_{c3} & \cos\beta_{c3} & \sqrt{2}L\cos\beta_{c3} \end{pmatrix}$$

$$J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}$$

$$C_1 = \begin{pmatrix} C_{1s}(\beta_{s1}) \\ C_{1c}(\beta_{c2}, \beta_{c3}) \end{pmatrix} = \begin{pmatrix} \cos\beta_{s1} & \sin\beta_{s1} & 0 \\ -\sin\beta_{c2} & \cos\beta_{c2} & d + \sqrt{2}L\sin\beta_{c2} \\ -\cos\beta_{c3} & -\sin\beta_{c3} & d + \sqrt{2}L\sin\beta_{c3} \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 0 \\ C_{2c} \end{pmatrix} = \begin{pmatrix} 0 & 0 \\ d & 0 \\ 0 & d \end{pmatrix}$$

2.3.5 Type (1,1) Robot

The robot has two fixed wheels on the same axle and one steering wheel, like the children tricycles as shown in Figure 2.10. The characteristic constants are specified in Table 2.6.

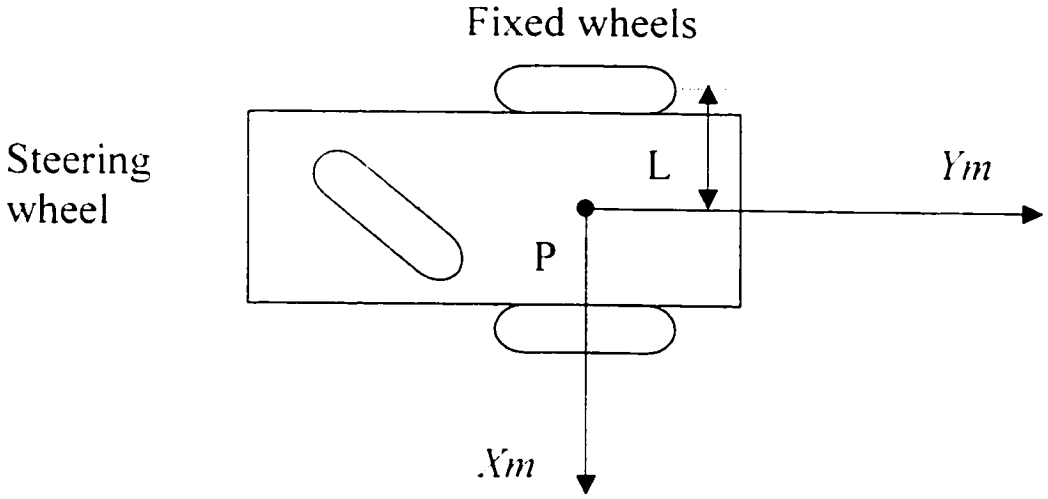

Figure 2.10 Type (1,1) robot.

Table 2.6: Characteristic constants of Type (1,1) robot with castor wheels.

wheels	α	β	l
1f	0	0	L
2f	π	0	L
3s	$3\pi/2$	-	L

The constraints have the form (2.10) and (2.11) where:

$$J_1 = \begin{pmatrix} J_{1f} \\ J_{1s}(\beta_{s3}) \end{pmatrix} = \begin{pmatrix} 0 & 1 & L \\ 0 & -1 & L \\ \cos\beta_{s3} & \sin\beta_{s3} & L\cos\beta_{s3} \end{pmatrix}$$

$$J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}$$

$$C_1 = \begin{pmatrix} C_{1f} \\ C_{1s}(\beta_{s3}) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -1 & 0 & 0 \\ \sin\beta_{s3} & -\cos\beta_{s3} & L\sin\beta_{s3} \end{pmatrix}$$

$$C_2 = 0$$

2.3.6 Type (1,2) Robot

The robot has two steering wheels and one castor wheel as shown in Figure 2.11. The characteristic constants are specified in Table 2.7.

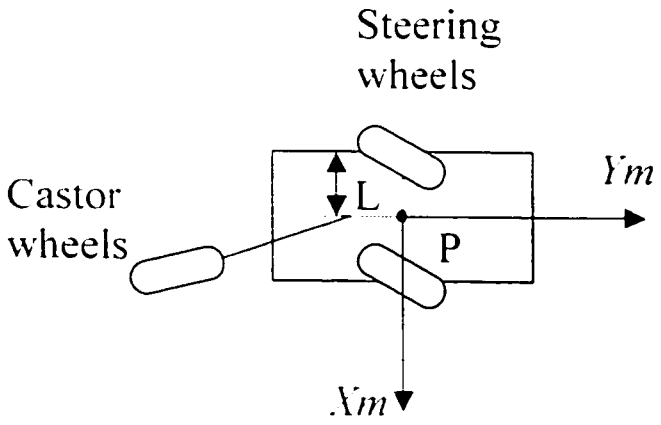

Figure 2.11 Type (1,2) robot.

Table 2.7: Characteristic constants of Type (1,2) robot.

wheels	α	β	l
$1s$	0	-	L
$2s$	π	-	L
$3c$	$3\pi/2$	-	L

The constraints have the form (2.10) and (2.11) where:

$$J_1 = \begin{pmatrix} J_{1s}(\beta_{s1}, \beta_{s2}) \\ J_{1c}(\beta_{c3}) \end{pmatrix} = \begin{pmatrix} -\sin\beta_{s1} & \cos\beta_{s1} & L\cos\beta_{s1} \\ \sin\beta_{s2} & -\cos\beta_{s2} & L\cos\beta_{s2} \\ \cos\beta_{c3} & \sin\beta_{c3} & L\cos\beta_{c3} \end{pmatrix}$$

$$J_2 = \begin{pmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & r \end{pmatrix}$$

$$C_1 = \begin{pmatrix} C_{1s}(\beta_{s1}, \beta_{s2}) \\ C_{1c}(\beta_{c3}) \end{pmatrix} = \begin{pmatrix} \cos\beta_{s1} & \sin\beta_{s1} & L\sin\beta_{s1} \\ -\cos\beta_{s2} & -\sin\beta_{s2} & L\sin\beta_{s2} \\ \sin\beta_{c3} & -\cos\beta_{c3} & d + L\sin\beta_{c3} \end{pmatrix}$$

$$C_2 = \begin{pmatrix} 0 \\ C_{2c} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ d \end{pmatrix} .$$

Chapter 3

Posture Kinematic Model

In this chapter, the analysis of mobility and steerability, as discussed in Chapter 2, is reformulated into a state space form, and will be useful for our subsequent developments.

It has been shown that, whatever the type of mobile robot, the velocity $\dot{\xi}(t)$ is restricted to a distribution Δ_c [43] defined as

$$\dot{\xi}(t) \in \Delta_c = span\{col(R^I(\theta)\sum(\beta_c))\} \quad \forall t$$

Where the columns of the matrix $\sum(\beta_c)$ form a basis of $N(C_1^*(\beta_c))$ i.e.,

$$N(C_1^*(\beta_c)) = span\{col(\sum(\beta_c))\}.$$

This is equivalent to the following statement: for all t, there exists a time-varying vector $\eta(t)$ such that

$$\dot{\xi} = R^I(\theta)\sum(\beta_c)\eta. \tag{3.1}$$

The dimension of the distribution Δ and, hence, of the vector $\eta(t)$ is the degree of mobility δ_m of the robot. In the case where the robot has no steering wheels ($\delta_s = 0$), that matrix Σ is constant and expression (3.1) reduces to

$$\dot{\xi} = R^T(\theta)\Sigma\eta. \tag{3.2}$$

In the opposite case ($\delta_s \geq 1$), the matrix Σ explicitly depends on the orientation coordinates β_s and the expression (3.1) can be augmented as follow:

$$\dot{\xi} = R^T(\theta)\Sigma(\beta_s)\eta \tag{3.3}$$

$$\dot{\beta}_s = \zeta. \tag{3.4}$$

The representation (3.2) (or (3.3) and (3.4)) can be regarded as a state space representation of the system, termed the *posture kinematic model*, with the posture coordinates ξ and (possibly) the orientation coordinates β_s as state variables while η and ζ --which are homogeneous to velocities-can be interpreted as control inputs entering the model linearly. Nevertheless, this interpretation shall be taken with care, since the true physical control inputs of a mobile robot are the torque provided by the embarked actuators; the kinematic state space model is in fact only a subsystem of the general dynamic model.

3.1 Generic Model of Wheeled Robots

In Chapter 2, we introduce a classification of all non-degenerate wheeled mobile robots according to the values of degree of mobility δ_m and degree of steerability δ_s. We consider WMRs moving on a horizontal plane, as shown in Figure 2.1. Suppose that there is no skidding between the wheels and the ground. The WMRs are classified according to the mobility $1 \le \delta_m \le 3$ and the steerability $0 \le \delta_s \le 2$ as type (m, s) robot [43]. It is easy to determine to which class a particular robot belongs to just by inspecting the number and the configuration of the fixed and steering wheels. Various examples are given below.

Here, we wish to emphasize that for any particular wheeled mobile robot, whatever its constructive peculiarities, it is always possible to select the origin and orientation of the moving frame (see Figure 2.1). So that the posture kinematic model of the robot takes a *generic* form which is unique for each class and completely determined by the two characteristic numbers δ_m and δ_s. In other terms, all robots of a given type can be described by the same posture kinematic model.

The five generic posture kinematic models are now presented [43, 44].

3.1.1 Type (3,0) Robot

The point P and axes X_m and Y_m can be selected arbitrarily (see Figure 2.7). The matrix $\sum$ can always be chosen as an (3×3) identity matrix.

The posture kinematic model reduces to

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \\ \eta_3 \end{pmatrix} \tag{3.5}$$

where η_1 and η_2 are the robot velocity components along X_m and Y_m, respectively, and η_3 is the angular velocity.

3.1.2 Type (2,0) Robot

The point P can be arbitrarily chosen along the axle of the fixed wheels, while axis X_m is aligned with this axle (see Figure 2.8). The matrix Σ is selected as

$$\Sigma = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The posture kinematic model (3.2) reduces to

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -\sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} \tag{3.6}$$

where η_1 is the robot velocity component along Y_m and η_2 is the angular velocity.

3.1.3 Type (2,1) Robot

The point P is the center of the steering wheel of the robot. The orientation of the moving frame can be arbitrarily chosen: let us refer to the choice in Figure 2.9.

The matrix $\Sigma(\beta_s)$ is selected as

$$\Sigma(\beta_s) = \begin{pmatrix} -\sin\beta_{s1} & 0 \\ \cos\beta_{s1} & 0 \\ 0 & 1 \end{pmatrix}.$$

The posture kinematic model (3.3) and (3.4) reduces to

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -\sin(\theta + \beta_{s2}) & 0 \\ \cos(\theta + \beta_{s1}) & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} \tag{3.7}$$

$$\dot{\beta}_{s2} = \xi_1. \tag{3.8}$$

3.1.4 Type (1,1) Robot

The point P must be located on the axle of the fixed wheels, at the intersection with the normal passing through the center of the steering wheel: axis Y_m is aligned with this normal (see Figure 2.10).

The matrix $\sum(\beta_s)$ is selected as

$$\sum(\beta_s) = \begin{pmatrix} 0 \\ L\sin\beta_{s3} \\ \cos\beta_{s3} \end{pmatrix}$$

The posture kinematic model (3.3) and (3.4) reduces to

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -L\sin\theta\sin\beta_{s3} \\ L\cos\theta\sin\beta_{s3} \\ \cos\beta_{s3} \end{pmatrix} \eta_1 \tag{3.9}$$

$$\dot{\beta} = \zeta \tag{3.10}$$

3.1.5 Type (1,2) Robot

The point P is the midpoint of the segment joining the centers of the two steering wheels; axis X_m is aligned with this segment (see Figure 2.11). The matrix $\sum(\beta_s)$ is selected as

$$\sum(\beta_s) = \begin{pmatrix} -2L\sin\beta_{s1}\sin\beta_{s2} \\ L\sin(\beta_{s1}+\beta_{s2}) \\ \sin(\beta_{s2}-\beta_{s1}) \end{pmatrix}$$

The posture kinematic model ((3.3) and (3.4)) reduces to

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -L(\sin\beta_{s1}\sin(\theta+\beta_{s2})+\sin\beta_{s2}\sin(\theta+\beta_{s1})) \\ L(\sin\beta_{s1}\cos(\theta+\beta_{s2})+\sin\beta_{s2}\cos(\theta+\beta_{s1})) \\ \sin(\beta_{s2}-\beta_{s1}) \end{pmatrix} \eta_1 \tag{3.11}$$

$$\dot{\beta}_{s1} = \zeta_1 \tag{3.12}$$

$$\dot{\beta}_{s2} = \zeta_2 \tag{3.13}$$

3.2 Mobility, Steerability and Maneuverability

It is convenient from now on to rewrite the posture kinematic model of mobile robots in the following compact form

$$\dot{z} = B(z)u \tag{3.14}$$

with either $(\delta_s = 0)$

$$z = \xi \qquad B(z) = R^T(\theta)\Sigma \qquad u = \eta \tag{3.15}$$

or $(\delta_s > 0)$

$$z = \begin{pmatrix} \xi \\ \beta_s \end{pmatrix} \qquad B(z) = \begin{pmatrix} R^T(\theta)\Sigma(\beta_s) & 0 \\ 0 & I \end{pmatrix} \qquad u = \begin{pmatrix} \eta \\ \zeta \end{pmatrix} \tag{3.16}$$

This posture kinematic model allows us to discuss further the maneuverability of WMRs. *The degree of mobility* δ_m is a first index of maneuverability. It is equal to the number of degrees of freedom that can be directly manipulated from the inputs η, without reorientation of the steering wheels. Intuitively it corresponds to how many "degrees of freedom" the robot could have instantaneously from its current configuration, without steering any of its wheels. This number δ_m is not equal to the overall number of "degrees of freedom" of the robot that can be manipulated from the inputs η and ζ. In fact this number is equal to the sum $\delta_M = \delta_m + \delta_s$ that we could call *degree of maneuverability*. It includes the δ_s additional degrees of freedom that are accessible from the inputs ζ. But the action of ζ on the posture coordinates ξ is indirect, since it is achieved only through the coordinates β_s that are related to ζ by an integral action. This reflects the fact that the modification of the orientation of a steering wheel cannot be achieved instantaneously.

The maneuverability of a WMRs depends not only on δ_M, but also on the way these δ_M degrees of freedom are partitioned into δ_m and δ_s. Therefore, two indices are needed to characterize maneuverability: δ_M and δ_m, or, equivalently, δ_m and δ_s, which are the two indices identifying the five classes of robots illustrated above.

Two robots with the same value of δ_M, but different δ_m, are not equivalent. For robots with $\delta_M = 3$, it is possible to freely assign the position of the ICR, either directly

from η, for Type (3,0) robots, or by orientation of one or two steering wheels for Type (2,1) and Type (1,2) robots. For robots with $\delta_{sf} = 2$, the ICR is constrained to a straight line (the axle of the fixed wheel). Its position on this line is assigned either directly for Type (2,0) robots, or by the orientation of a steering wheel for Type (1,1) robots.

Similarly, two WMRs with the same value of δ_m, but different δ_{sf}, are not equivalent: the robot with the largest δ_{sf} is more maneuverable. Compare, for instance, a Type (1,1) robot and a Type (1,2) robot with $\delta_m = 1$ and, respectively, $\delta_{sf} = 2$ and $\delta_{sf} = 3$. The position of the ICR for a Type (1,2) robot can be assigned freely in the plane, just by orientating 2 steering wheels, while for a Type (1,1) robot, the ICR is constrained to belong to the axle of the fixed wheels, its position on this axle being specified by the orientation of the steering wheel. Since the steering directions of the steering wheels can usually be changed quickly, especially for small indoors robots, it follows, from a practical viewpoint, that a Type (1,2) robot is more maneuverable than a Type (1,1) robot.

Obviously, the ideal situation is that of omni directional robots where $\delta_m = \delta_{sf} = 3$.

Chapter 4

Background Information of Nomad 200

This section provides background information of Nomad 200 [5, 23, 24, 43], it is useful to the reader to understand the Nomad 200 mobile robot, as well as the potential field method implemented to realize the motion of robots.

4.1 Nomad 200 Mobile Robot

The Nomad 200 mobile robot is produced by Nomadic Technologies. As show in Figure 4.1. The Nomad 200 is an integrated mobile robot system with different sensing options. Besides tactile, infrared, ultrasonic, and 2D laser sensors the platform is being controlled by means of an on-board multi-processor system, which performs sensor and motor control, host computer communication, and supports on-board programming. There is an integrated software package for the host computer including a graphic interface, a robot simulator, and a library of motion planning, motion control, and sensory data interpretation functions - the application programs is by radio ethernet system. This makes the robot an efficient tool especially for development of new path

planning algorithms and obstacle avoidance. The platform is can also be equipped with a lift mechanism to perform mobile manipulation tasks.

4.2 Mechanical System

A Nomad 200 mobile robot is shown in Figure 4.1. It has a cylindrical shape with an approximate radius of 0.23m, and a kinematical model equivalent to a unicycle. The upper turret, which carries 16 Polaroid ultrasonic sensors, may be independently rotated. The robot control software runs under Linux on computer that communicates with Nomad 200 through a radio link. All the algorithms for map building and navigation have been implemented in the C language.

The Nomad 200 mobile base is three servos, three-wheels synchronous drive non-holonomic system with zero gyro-radius. The three wheels translate together (controlled by one motor) and rotate together (controlled by a second motor), as shown in Figures 4.2 and 4.3. A third motor controls the angular position of the turret. The robot can only translate along the forward and backward directions, which the three wheels are aligned (this is referred to as non-holonomic constraint, similar to that of a car). The robot has a zero gyro-radius, i.e. the robot can rotate around its center.

The Nomad 200 has a maximum translational speed of 50 cm per second and a maximum rotational speed of 45^{0} per second.

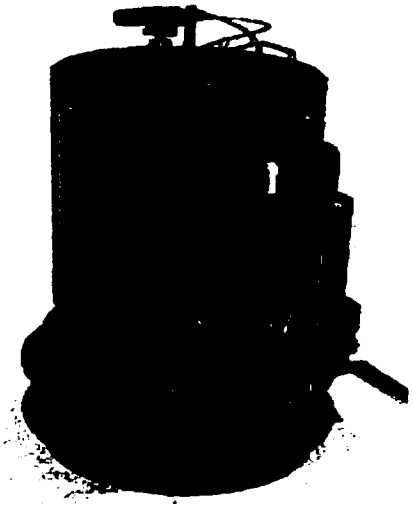

Figure 4.1. Robot Nomad 200

Table 4.1 Technical Data

size	(mm)	970x530
maximum velocity	(m/s)	0.5
payload	(daN)	23
dead weight	(kg)	59
sensors	Tactile sensor system Fixed ultra sound system Vision system Laser-navigation system	
on-board control system	Pentium 133MHz, 32 MB RAM Speech-synthesis module Operating system: LINUX	

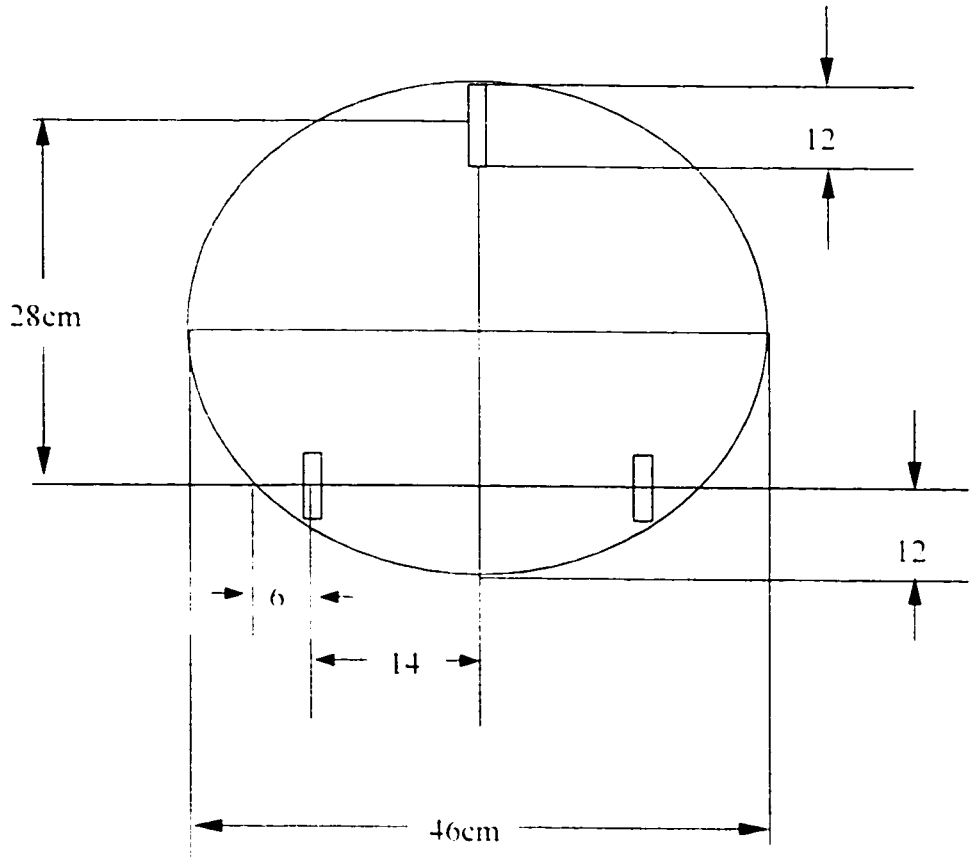

Figure 4.2 Bottom view of Nomad 200

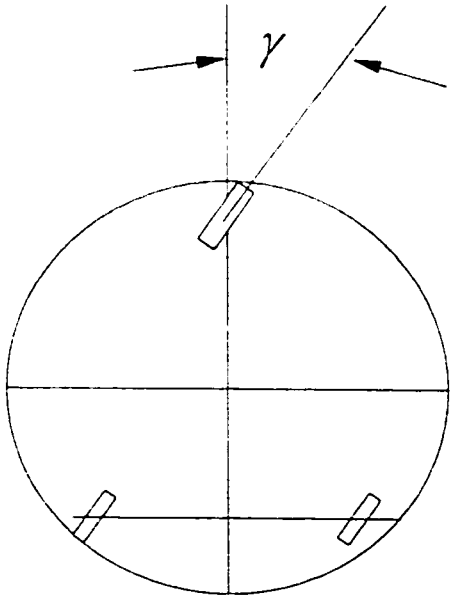

Figure 4.3 Steering diagram of Nomad 200

4.3 Sensor Systems

The robot's sensor systems include tactile (bumper) sensors, ultrasonic sensors. The tactile system, which consists of two bumper rings, is used to detect contact with any object.

The Nomad 200 also has a 16 channel sonar ranging system which can supply range information from 43 cm to 648 cm with 1 percent accuracy over the entire range. The sonar system is a time of flight ranging sensor based upon the return time of an acoustic signal. The sensors are standard Polaroid transducers with a beam width of 25°. The circumference of the robot is covered by 16 sensors.

Although the user manuals [23, 24] for the Nomad 200 robot state that the maximum sonar range is 648 cm, it is determined by two parameters (half-cone and overlap) that are stored in the *robot setup* file. The sonar sensors have a non-zero beam-width, i.e., they can detect an object as long as it overlaps with this cone. Their half beam-width for sonar is specified by half-cone in the setup file. If this variable is set to 0, the sensor can detect an object only if it intersects with the ray drawn from the sensor along the direction where the sensor is pointing, and is within detectable distance. When this variable is set to some positive values, the sensors can detect objects that fall within the cone specified by this variable assuming there is sufficient overlap between the object and the sonar cone (as specified by overlap in the setup file), and is within detectable distance. Briefly, half-cone sets half the angular range of the main lobe of the sonar while overlap sets the minimal apparent size of a surface to be detected when using the conical model.

The sonar half-cones has to overlap in order to cover 360 degrees around the robot. If they overlap quite a distance away from the robot, then the triangular area that stays between two consecutive sonar and the intersection point of sonar beams cannot be covered at all. On the other hand, anything in the overlapped zone is seen by both sonar. This means that the position of that object cannot be calculated accurately. Worst of all, the sonar that see the same object that is in their overlapped zone, cannot sense another object even if it is on the line of sight, unless it is closer than the one in overlapped zone. So, letting the sonar beams overlap very close to the robot creates a huge overlapped zone, which in turn affects the sight of the sonar negatively. The maximum sonar range in the area that is exactly in the middle of two consecutive sonar that is shorter than the one in line of sight area. The values of half-cone and overlap have to be set in such a way to best balance these factors.

4.4 Posture Kinematic Model of Nomad 200

As discussed in Chapter 3, the posture kinematic model of wheeled robot is equivalent to the following statement: for all t, there exists a time-varying vector $\eta(t)$ such that

$$\dot{\xi} = R^{-1}(\theta)\sum(\beta_i)\eta \tag{4.1}$$

$$\beta_i = \varsigma_i \tag{4.2}$$

The representation (4.1) and (4.2) can be regarded as a state space representation of the system, termed the *posture kinematic model*, with the posture coordinates ξ and (possibly) the orientation coordinates β_i as state variables while η and ς --which are homogeneous to velocities which can be interpreted as control inputs entering the model linearly.

It is convenient from now on to rewrite the posture kinematic model of mobile robots in the following compact form

$$\dot{z} = B(z)u \tag{4.3}$$

where

$$z = \xi \qquad B(z) = R^T(\theta)\sum(\beta_i) \qquad u = \eta \tag{4.4}$$

As we discussed above, the Nomad 200 mobile robot base is three servos, three-wheels synchronous drive non-holonomic system with zero gyro-radius. The three wheels translate together (controlled by one motor) and rotate together (controlled by a second motor), as shown in Figures 4.2 and 4.3. The robot can only translate along the forward and backward directions, which the three wheels are aligned (this is referred to as non-holonomic constraint, similar to that of a car). The robot has a zero gyro-radius, i.e. the robot can rotate around its center. The kinematical model equivalent to a unicycle, according to the definition of degree of mobility and degree of steerability, this wheeled robot is belong to Type (2,0), that's means the degree of

mobility of Nomad 200 is two ($\delta_m = 2$), and degree of steerability is zero ($\delta_s = 0$), so the equation (4.3) and (4.4) can be expressed as below:

$$\dot{z} = B(z)u \tag{4.5}$$

where

$$z = \xi \qquad B(z) = R^T(\theta)\sum \qquad u = \eta \tag{4.6}$$

As show in Figure 2.8, the point P can be arbitrarily chosen along the axle of the fixed wheels, while axis Y_m is aligned with this axle. The matrix $R^T(\theta)$ and $\sum$ is selected as

$$R^T(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}, \qquad \sum = \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The posture kinematic model of Nomad 200 can be written as:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -\sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \eta_1 \\ \eta_2 \end{pmatrix} \tag{4.7}$$

where x, y are coordinates of the origin P of the moving frame and θ is the orientation angle of the moving frame (see Figure 2.1), both with respect to the base frame with the origin of the coordinates at O; η_1 is the robot velocity component along Y_m and η_2 is the angular velocity.

4.5 Robot Simulator

The Nomadic Host Software Development Environment is a full-featured object based mobile robot software development package for the Nomad 200 mobile robot. It consists of two parts: *the server* and *the client*. The server performs four functions:

Host Robot Interface, which allows complete control of the robot from a host computer.

Robot Simulator which simulates the Nomad 200 robot (including its basic motions: translation, steering and rotation, and its basic sensor systems: tactile, infrared, sonar and laser

Graphic User Interface, which provides graphic display for various sensory information and convenient interface with the robot and the robot simulator.

Client Server Language User Interface which allows users' C or Lisp program (acts as a client process) to access the server. The client part provides the link (in terms of a set of interface functions) between the application program and the server.

The graphical user interface consists of several windows that give information and allow control of a robot. First, there is the world (map) window, which provides an

overall view of the environment (real or simulated) that the robots are in as well as the positions of each robot relative to the environment and each other. Also there is the robot window, (one copy for each robot), which contains information about each individual robot, such as current command executed, position, orientation, and sensor data history. Attached to each robot window are two windows that give more detailed information about the current sensor readings, including ray and half-cone data. Each time any of the functions in the return sensor data is called, the sensory data return as well as the current positions of robots is displayed graphically on these windows. The users are allowed to draw maps in the world window to simulate the environment. The figures that show the simulation results in the following are the snapshots of the world window taken at different time instants during a simulation.

In order to run the simulator, the executable server program (*Xserver*), the setup files for the world (*world.setup*) and for each robot (*robot.setup*), as well as the *license* file must be in the same directory. To start the server, one simply executes the *Xserver*. Individual setup files can be specified as command line parameters. If the setup files are not specified, the server will automatically look for *world.setup* and *robot.setup*. It is necessary to have a separate setup file for each robot to be created. The name of each robot setup file must be specified in the *world.setup* file. The best way to discriminate the robots is to set a different color for each robot in its own *robot.setup* file. The application program for each robot should run simultaneously but as a separate process, by taking advantage of multitasking capabilities of Linux operating system. This makes

debugging easy and provides the possibility to test each behavior independently as well as to add or remove some robots during the run.

4.6 Motion Commands

To fully understand the robot motion commands it is helpful to have some knowledge about the drive system of robot, which consists of three independent axes. The first axis is the translational axis: it drives the three wheels of the robot that are mechanically coupled to reduce slip. The second axis is the steering axis, which changes the direction of the robot's wheels. The wheels remain parallel to each other at any time. This allows the robot to turn in place. Both the translational axis and the steering axis move the robot in a base plane in a non-holonomic fashion. In addition to the two axes that are responsible for the motion of the base there is a third axis that move the turret axis to enable the user navigate the robot base while maintaining a constant heading with the turret.

All three axes can be controlled in two different control modes: velocity and position control. In velocity control the goal is to maintain a given velocity, whereas position control attains and maintains a given position relative to the current position of the robot.

For all motion commands velocities are specified in 1/10 of cm per second for translational axis and in 1/10 of degrees per second for the steering and the turret axis. Positions are specified relative to current position. Positive velocities and positive position for rotations are considered to be counterclockwise: they are relative to the

current position of steering and turret axes. Note that moving the steering axis at a constant speed during translation will cause the robot to move on a circular arc.

In the following description of the motion commands *t*, *s*, and *r* are used to indicate that the arguments of the functions refer to the translational, steering, and turret axes, respectively. All commands return TURE upon successful transmission to the robot, FALSE otherwise.

Int ac(int t_ac, int s_ac, int r_ac)

This command sets the acceleration for the three axes. The unit is the unit of velocity divided by seconds. The acceleration must be lower than:

-300 $\frac{0.1in}{s^2}$ for translational axis,

-300 $\frac{0.1deg}{s^2}$ for steering axis, and

-300 $\frac{0.1deg}{s^2}$ for turret axis.

While performing only very small motions with position control the accelerations should be set to small value. Otherwise the robot will accelerate abruptly into the opposite direction to compensate for the error, move further than the desired position, and so on.

Int sp(int t_sp, int s_sp, int r_sp)

This command set the maximum speeds for the three axes. The maximum velocity must be lower than

-200 $\frac{0.1in}{s}$ for the translational axis.

-450 $\frac{0.1\deg}{s}$ for the steering axis, and

-450 $\frac{0.1\deg}{s}$ for the turret axis.

Int vm(int t_vm, int s_vm, int r_vm)

This command controls the three axes of the robot in velocity mode. The arguments to this function are the desired velocities for the three axes; they can be negative but their absolute value has to stay below the maximal value given above. A velocity of zero will maintain the current position

When a *vm* command is issued the robot will move its axes at the requested velocity and will continue moving unless another motion command is issued, or a timeout occurs.

Note: after a *vm* command the user has to issue the command *st* (stop) and *ws* (wait to stop), before the robot can be moved with *pr*.

Int pr(int t_pr, int s_pr, int r_pr)

This command controls the three axes of the robot in position mode. The arguments are the desired position relative to the current one. Note that *pr*

(100,200,200) will cause all axes to move at the same time. As a result, the motion of the robot will not be a straight line, but a circular arc. However, if the rotation of the steering is completed before the completion of translation (perhaps due to different speed or acceleration), the last part of the trajectory will be a straight line. In either case, the length of the trajectory will be 10. Note also that the function call will return immediately. To wait for the motion to be completed the user should issue the command *ws* after *pr*. A velocity of zero will maintain the current position.

Note: If the command proceeding a *pr* is a *vm*, the command, stand *ws*, have to be issued prior to *pr*.

Int mv(int t_mode, int t_mv, int s_mode, int s_mv, int r_mode, int r_mv)

This command allows the user to drive all three axes independently from each other in velocity or position control mode. There are two arguments for each axis:

1. The first argument determines the control mode by specifying

-MV_VM for velocity control,

-MV_PR for position control, and

-MV_IGNORE if the motion for that axis should remain unaltered, and

2. The second argument is a velocity if velocity control is the specified mode or a position id position mode is requested.

For example, *mv*(MV_VM, 100, MV_PR, 300, MV_IGNORE, MV_IGNORE) will cause the robot to start translating 25 cm per second, the steering will rotate 30

degrees counterclockwise and the motion that was executed on the turret axis will remain unchanged.

Int st(void)

The robot will stop after this command has been sent. However, since the functioncall returns immediately, the robot can still be a decelerating after the termination of command. It is recommended to issue a *ws*() after a *st*().

Int ws(unsigned char t_ws, unsigned char s_ws, unsigned char r_ws)

This allows the user to wait for some or all of the axes to be stopped. The first three arguments are TURE if the command should return after stopping of that particular axis; the argument should be FALSE if the status of an axis does not need to be monitored. If not all of specified axes have stopped before the elapse of timeout seconds the function returns.

Int lp(void)

Even after the robot has stopped it will still try to maintain its position. That means that any attempt to push it will cause the motors to drive the robot into the opposite direction. This command causes the robot to go in limp mode. After executing it, position will no longer be maintained and the robot can be pushed around. This control mode is sometimes referred to as floating.

Int zr(void)

Since the turret of the robot can be rotated independently of the base plane, they can be misaligned. This command causes steering and turret to have the same heading. Therefore, it should be called at the beginning of an application program as part of the initialization. The function call *zr* will only return after the process of zeroing is completed.

Int dp(int x, int y)

Sets the robot position to (x,y).

Int da(int th, int tu)

The orientation of steering and the turret is set to *th* and *tu* $\frac{1}{10}$s of degrees, respectively.

Int get_rc(void)

The issuing of the command '*Int get_rc*(void)' updates the following entries in the state vectors:

STATE_CONF_X, STATE_CONF_Y, STATE_CONF_STEER, STATE_CONF_TURRET.

Int get_rv(void)

And the issuing of the command "*Int get_rv*(void)" updates the following entries in the state vectors:

STATE_VEL_TRANS, STATE_VEL_STEER, STATE_VEL_TURRET.

Their input variables are:

translation speed, steering speed, position (defined by coordinates),and

measurable variables are:

translation speed, steering speed, position (x, y), and orientation of mobile robot.

Chapter 5

Control of a Wheeled Robot Using a Virtual Vehicle Approach

In this chapter a solution using a "virtual" vehicle approach to the problem of controlling a wheeled robot is given in [31], which is experimentally compared with the proposed approach in Chapter 6. Here we shall use one generic path following control strategy proposed in [31], which is model independent, and uses position and orientation error feedback for the experimental test. One of the advantages of this approach is that it is quite robust with respect to measurement errors and external disturbances. If both errors and disturbances are within certain bounds, the reference point is going to move along the desired trajectory while the robot follows it. Otherwise, the reference point might "stop" to wait for the robot. For this reason we call the reference point together with the associated differential equation a virtual vehicle.

5.1 Control Problem

The control problem is to find a steering angle $\delta_f(t)$ so that the car follows a virtual vehicle $s(t)$ moving on a smooth reference path (i.e. $p'^2 + q'^2 \neq 0 \ \forall s$)

$$x_d = p(s)$$

$$y_d = q(s) \tag{5.1}$$

In other words, we require

$$\lim_{t \to \infty} \rho(t) = d \tag{5.2}$$

$$\lim_{t \to \infty} |\varphi - \varphi_d| \leq \Delta \tag{5.3}$$

where

$$\rho(t) = \sqrt{\Delta x^2 + \Delta y^2}$$

$$\Delta x = x - x_d$$

$$\Delta y = y - y_d \tag{5.4}$$

here φ is the yaw angle (orientation of the car), $\varphi_d = \dfrac{(y - y_d)}{(x - x_d)}$ is the desired orientation, and (x, y) is a reference point on the car, for example the center of gravity or the middle point on the front axle. Furthermore, $\Delta > 0$ is a small number that depends on the maximum curvature of the reference path, and d is the "look-ahead" distance.

5.2 Control Algorithm

From (5.1) it can be obtained that

$$\dot{x}_d = p'(s)\dot{s}$$

$$\dot{y}_d = q'(s)\dot{s} \tag{5.5}$$

which implies

$$\dot{s} = \frac{p'(s)}{p'^2(s)+q'^2(s)}\dot{x}_d + \frac{q'(s)}{p'^2(s)+q'^2(s)}\dot{y}_d \tag{5.6}$$

and this suggests that if the car (x, y) tracks the path perfectly we have

$$\dot{s} = \frac{p'(s)}{p'^2(s)+q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s)+q'^2(s)}\dot{y} \tag{5.7}$$

On the other hand, if the velocity of the car $(\dot{x}, \dot{y})^T$ is perpendicular to the tangential direction of the reference path at index s, then $\dot{s}$ in (5.7) would be zero. In order to avoid the situation where the virtual vehicle might get stuck, and make the algorithm work globally, we introduce the following perturbation in $\dot{s}$

$$\dot{s} = kv\rho e^{-\rho_d} \frac{p'(s)}{p'^2(s)+q'^2(s)}\dot{x} + \frac{q'(s)}{p'^2(s)+q'^2(s)}\dot{y} \tag{5.8}$$

where k and d are two positive constants that need to be adjusted. The first term in (5.8) would give the virtual vehicle a positive "push" when the real car is approaching it. In this, we do not allow the virtual vehicle to go backwards, and therefore we set

$$\dot{s} = \begin{cases} \eta & if \quad \eta \geq 0 \\ 0 & if \quad \eta < 0 \end{cases} \tag{5.9}$$

where $\eta = kv\rho e^{-\rho/d} \dfrac{p'(s)}{p'^2(s)+q'^2(s)} \dot{x} + \dfrac{q'(s)}{p'^2(s)+q'^2(s)} \dot{y}$

Naturally in order for (5.2) to be realized, the robot should be steered close to the virtual vehicle. For this the following steering control is adopted [31]:

$$\delta_f = -k(\varphi - \varphi_d) \tag{5.10}$$

where δ_f is the steering angle, and k should be chosen to reflect the constraint on the maximum steering angle (since, $(\varphi - \varphi_d) \in [-\pi, \pi]$).

5.3 Simulation Example

We desired the trajectory as follow

$$\begin{aligned} x_d &= r\cos(t) \\ y_d &= r\sin(t) \end{aligned} \tag{5.11}$$

which is a circular path on the plane.

From (5.11) it can be derived

$$\dot{x}_d = p'(s)\dot{s} = -r\sin(\theta)\dot{\theta}$$

$$\dot{y}_d = q'(s)\dot{s} = r\cos(\theta)\dot{\theta} \tag{5.12}$$

we obtain

$$\dot{\theta} = \frac{-r\sin(\theta)}{(-r\sin(\theta))^2 + (r\cos(\theta))^2}\dot{x}_d + \frac{r\cos(\theta)}{(-r\sin(\theta))^2 + (r\cos(\theta))^2}\dot{y}_d \tag{5.13}$$

and we suggest that if the car (x,y) tracks the path perfectly ,it is

$$\dot{\theta} = \frac{-r\sin(\theta)}{(-r\sin(\theta))^2 + (r\cos(\theta))^2}\dot{x} + \frac{r\cos(\theta)}{(-r\sin(\theta))^2 + (r\cos(\theta))^2}\dot{y}$$

$$= \frac{-\sin(\theta)}{r}\dot{x} + \frac{\cos(\theta)}{r}\dot{y}$$

$$= \frac{-\dot{x}\sin(\theta) + \dot{y}\cos(\theta)}{r} \tag{5.14}$$

If the velocity of the car $(\dot{x},\dot{y})^T$ is perpendicular to the tangential direction of the reference path at index s, then $\dot{s}$ in (5.7) would be zero. In order to avoid the

situation where the virtual vehicle might get stuck, and make the algorithm work globally, the following perturbation in $\dot{\theta}$ is introduced.

$$\dot{\theta} = kv\rho e^{-\rho/d} + \frac{-\dot{x}\sin(\theta) + \dot{y}\cos(\theta)}{r} \tag{5.15}$$

where k and d are two positive constants that need be turned. The first term in (5.8) would give the virtual vehicle a positive "push" when the real car is approaching it. Here, we do not allow the virtual vehicle to go backwards, and therefore set

$$\dot{\theta} = \begin{cases} \eta & if \quad \eta \geq 0 \\ 0 & if \quad \eta < 0 \end{cases} \tag{5.16}$$

where $\eta = kv\rho e^{-\rho/d} + \frac{-\dot{x}\sin(\theta) + \dot{y}\cos(\theta)}{r}$

In order for (5.2) to be realized, the robot should be steered close to the virtual vehicle, and thus, we propose the following steering control:

$$\delta_f = -k(\varphi - \varphi_d) \tag{5.17}$$

where δ_f is the steering angle, and k should be chosen to reflect the constraint on the maximum steering angle (since, $(\varphi - \varphi_d) \in [-\pi, \pi]$).

Three experiments are performed on Nomad 200 where the robot tracks a circular path, the radius of the circular is 254cm, each time starting from a different initial position. The simulation examples can be seen in Figures 5.1- 5.3. In Figure 5.1, the translational velocity along the x axis is set as 7.5cm/s, the initial position is set as:

x(0)=0, y(0)=0, $\theta(0) = 0^{\circ}$

In Figure 5.2, the translational velocity along the x axis is set as 7.5cm/s, the initial position is set as:

x(0)=508cm, y(0)=0, $\theta(0) = 0^{\circ}$

In Figure 5.3, the translational velocity along the x axis is set as 7.5cm/s, the initial position is set as:

x(0)=381cm, y(0)=203cm, $\theta(0) = 0^{\circ}$

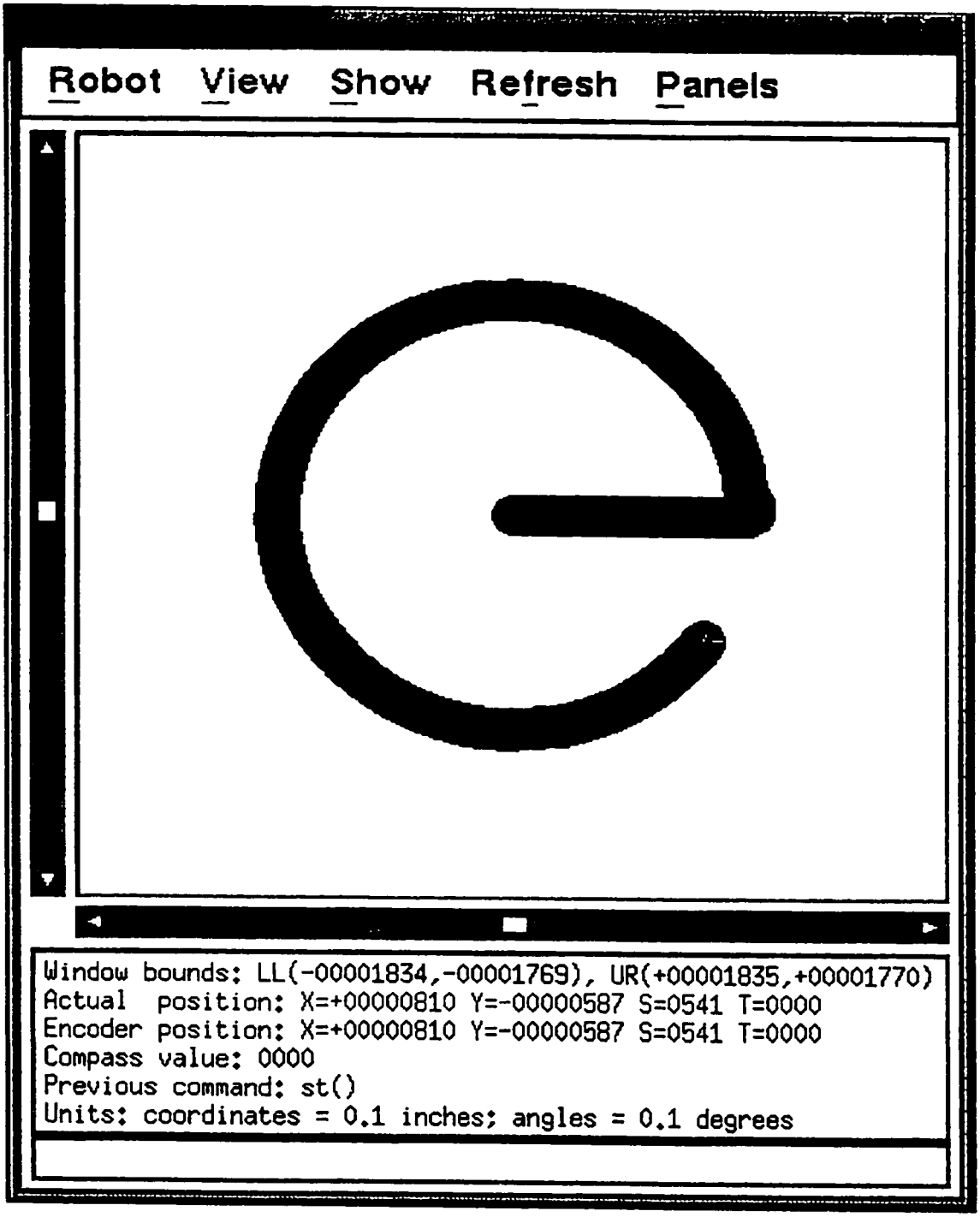

Figure 5.1 Nomad 200 simulation result with

r=254cm, x(0)=0, y(0)=0, $\theta(0) = 0^{\circ}$

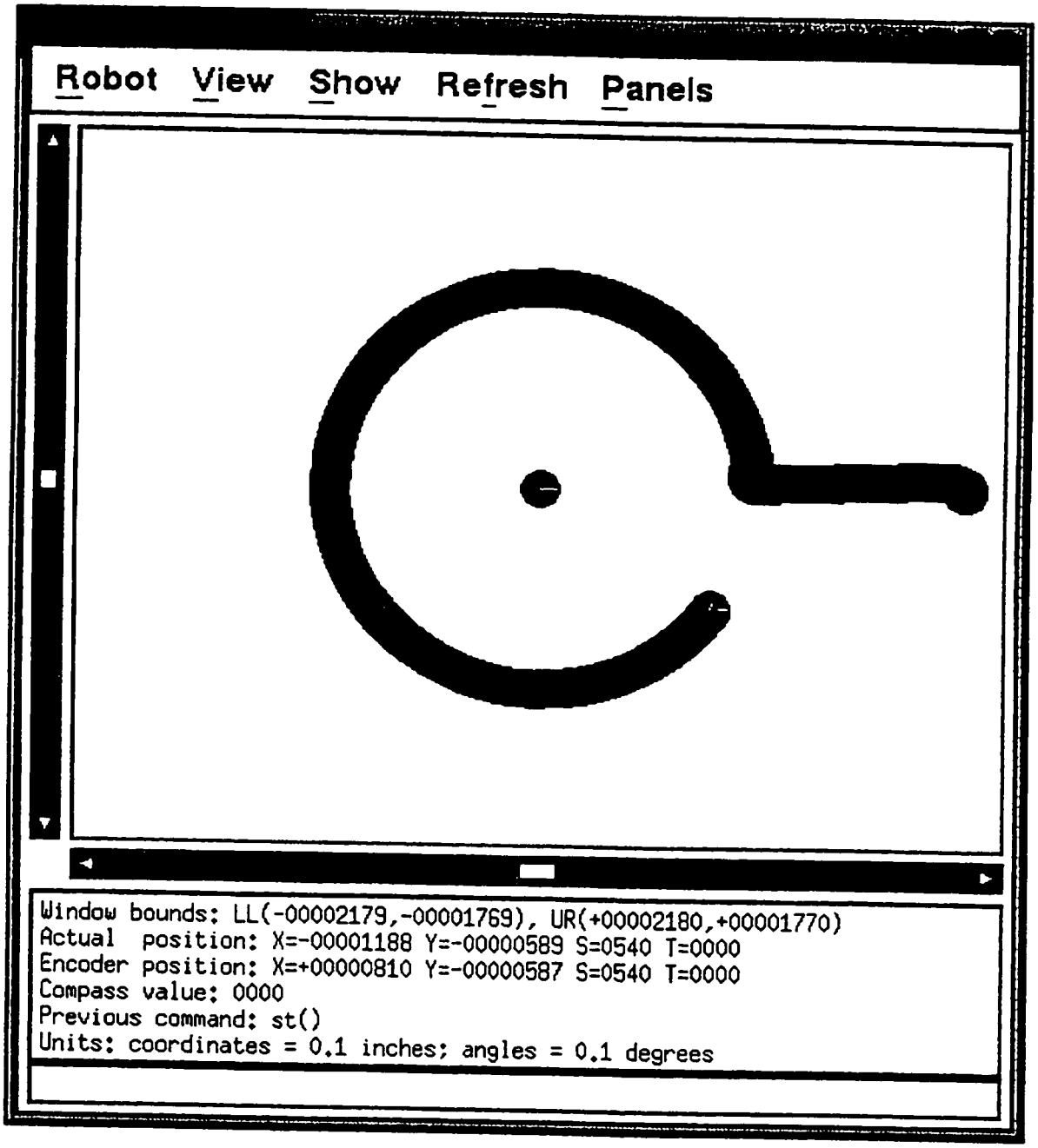

Figure 5.2 Nomad 200 simulation result with

r=254cm, x(0)=508cm, y(0)=0, $\theta(0) = 0^{\circ}$

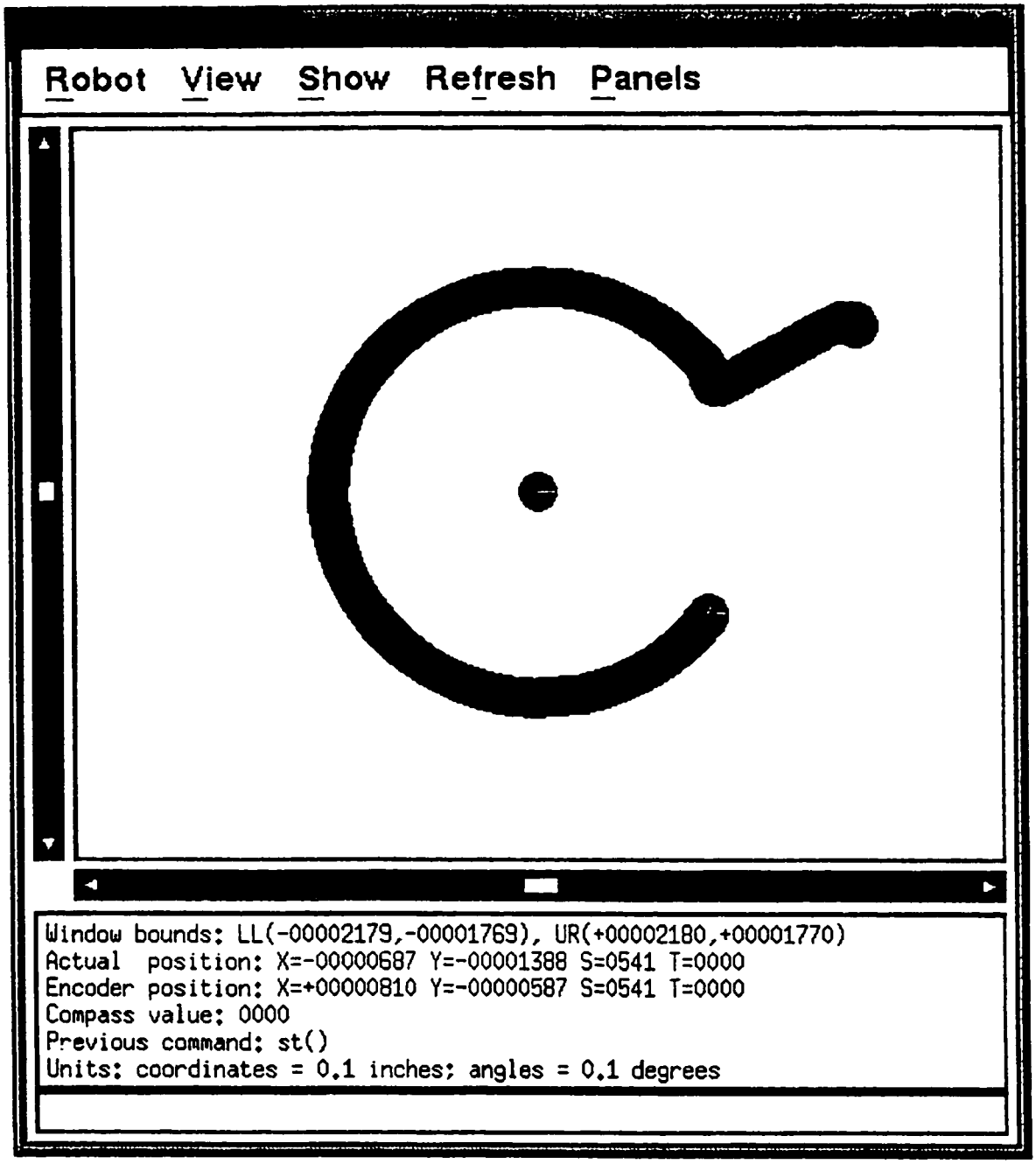

Figure 5.3 Nomad 200 simulation result with

r=254cm, x(0)=381cm, y(0)=203cm, $\theta(0) = 0^{\circ}$

5.4 Implementation of the Algorithm on Nomad 200

Extensive simulations on Nomad 200 are carried out to verify the developed algorithm. The results are shown in Figures 5.1-5.3, which show the validity of the algorithm. Based on the simulation results, the experiments are therefore conducted on the Nomad 200 at the "Fuzzy Systems Research Lab" at Concordia University. In the implementation, the parameters of the control algorithm are set as:

sample time: T=1.0s, k=0.8, d=1.0cm

The desired trajectory is a circular on the plane; the circle radius is set as r=100cm, and the robot starts from the different original situation, the linear translational velocity along the x axis is set constant v=10cm/s. The initial position and orientation, we set as:

Case 1:

x(0)=10cm, y(0)=5.5cm, $\theta(0) = 10^{\circ}$

Case 2:

x(0)=150cm, y(0)=13cm, $\theta(0) = 5^{\circ}$

Case 3:

x(0)=150cm, y(0)=80cm, $\theta(0) = 0^{\circ}$

Implementing these ideas on actual robots provides us real experimental system that behaves satisfactorily.

The experimental result with r=100cm, x(0)=10cm, y(0)=5.5cm, $\theta(0) = 10^{\circ}$ is shown in Figure 5.4. Figure 5.5 shows the experimental result with r=100cm, x(0)=150cm, y(0)=13cm, $\theta(0) = 5^{\circ}$. Experiment result with r=100cm, x(0)=150cm, y(0)=80cm, $\theta(0) = 0^{\circ}$ is shown in Figure 5.6. The experimental tests match the results given in the simulation. The control program and experiment data are given in Appendix I.

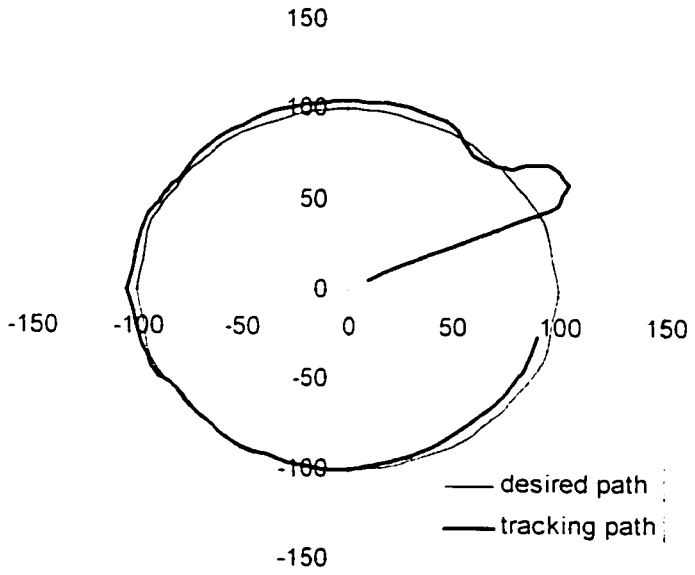

Figure 5.4 Experiment result with r=100cm,x(0)=10cm,y(0)=5.5cm, $\theta(0) = 10^{\circ}$

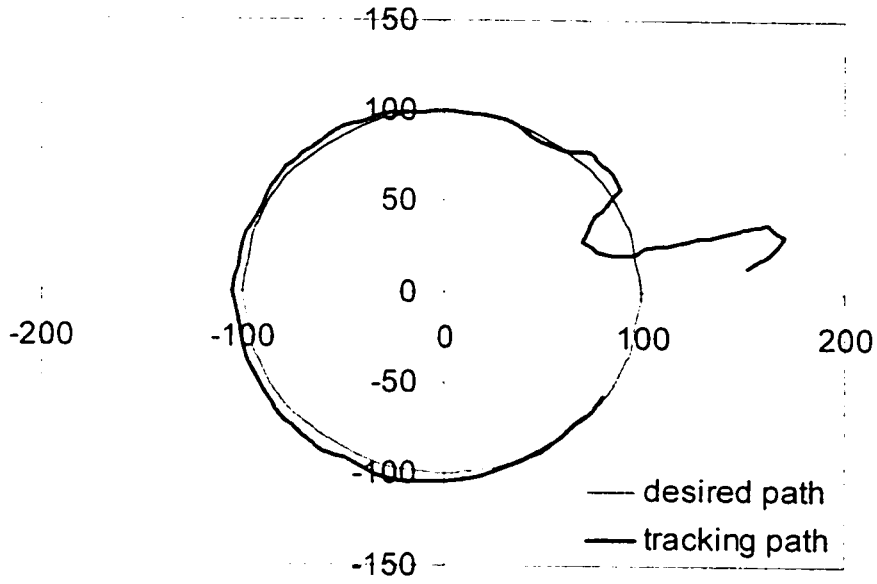

Figure 5.5 Experiment result with r=100cm,x(0)=150cm,y(0)=13cm, $\theta(0) = 5°$

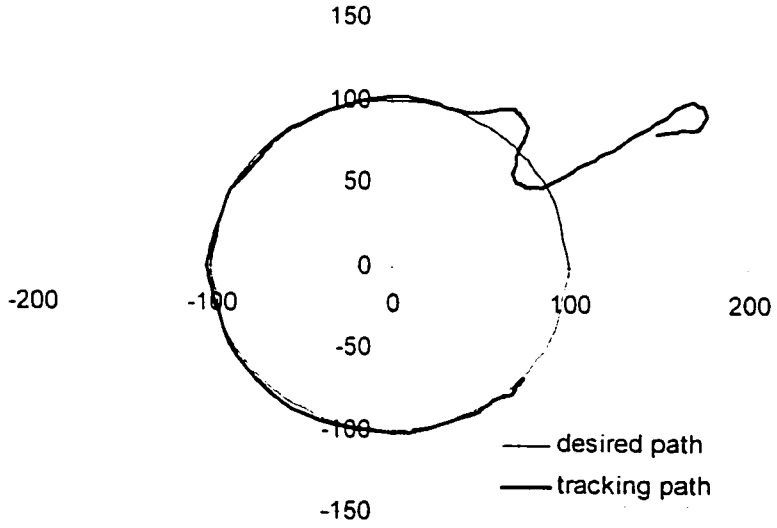

Figure5.6 Experiment result with r=100cm,x(0)=150cm,y(0)=80cm, $\theta(0) = 0°$

The implementation results show that this algorithm can track the desired trajectory. But the robot could not track the desired trajectory perfectly when the translational velocity was getting large. In general, the translational velocity is set as a small constant, and only the angular velocity is adjusted. Thus, this control algorithm is not a fully close-loop feedback, and it could not correct the tracking errors. In the next chapter two inputs that control the robot, the translational velocity and angular velocity, are controlled to realize the closed-loop control.

Chapter 6

Adaptive Control of Wheeled Robots

In this chapter, we now focus out attention to the general posture kinematic model given in (3.1), which can be re-written as

$$\dot{q} = R^T(\theta)\Sigma(\beta_s)v. \tag{6.1}$$

where $\dot{\xi}$ is replaced by a new symbol $\dot{q}$, and η is replaced by v.

6.1 Main Results

We now investigate the tracking problem for the wheeled robot systems discussed above. In the following development, we only consider two independent generalized velocities $(p = 2)$ case for the sake of simplicity, but the results can easily be extended to a general case. We should emphasize that our goal in this chapter is to develop a tracking control strategy in a simple setting that reveals its essential features.

6.2 Controller Design

Using a coordinate transformation, $y = \Psi(q)$ and a state feed back, $v = \Omega_1(q)u$, the kinematic system (6.1) with $p = 2$ can be locally or globally converted to the so called chained form

$$\begin{aligned} \dot{y}_1 &= u_1 \\ \dot{y}_j &= u_1 y_{j+1}, \quad (2 \le j \le n-1) \\ \dot{y}_n &= u_2 \end{aligned} \tag{6.2}$$

A necessary and sufficient condition for the existence of the transformation of the kinematic system (6.1) with p=2 into this chained form (single-chain) was given in [17][3]. For the general case (multi-chain case), the discussion on the existence condition of the transformation may refer to [25].

Since the desired trajectory q_d should satisfy the constraint equation (6.1), there must exist a desired v_d satisfying

$$\dot{q}_d = R^T(\theta)\Sigma(\beta_s)v_d \tag{6.3}$$

Remark: It should be noted that the choice of the feasible trajectory for nonholonomic systems is strongly related to its *flatness* [6] or the choice of back-followable outputs and itself is an active research topic [26]. It is beyond the 'scope of this thesis to address this issue.

Based on the fact that the kinematic system (6.1) can be converted into the chained form (6.2), there must exists a transformation $y_d = \Psi(q_d)$ and a state feedback, $v_d = \Omega_1(q_d)u_d$, such that the equation (6.3) can be converted into,

$$\begin{aligned} \dot{y}_{d1} &= u_{d1} \\ \dot{y}_{dj} &= u_{d1} y_{dj+1}, \qquad (2 \le j \le n-1) \\ \dot{y}_{dn} &= u_{d2} \end{aligned} \tag{6.4}$$

With the above transformations, the tracking problem considered in this thesis can be restated as seeking a strategy to specify a control u for the system (6.2) such that $\{y, \dot{y}\} \to \{y_d, \dot{y}_d\}$.

For the development of control law, the following assumptions are required.

Assumption (A1): The trajectories y_d and $\frac{d^i y_{d1}}{dt^i}(1 \le i \le n-1)$ are bounded and $\liminf_{t \to \infty} |u_{d1}| > 0$.

Remark: As will be seen from stability analysis, Assumption (A1) on the boundedness of y_d can be relaxed to that, $y_{di}(2 \le i \le n-1)$ are bounded, and there is no boundedness requirement for y_{d1}. This is the case used in the simulation example.

In the following, we define $e = [e_1, e_2, \cdots, e_n]^T = y - y_d$ and $\alpha = [\alpha_1, \alpha_2, \alpha_3, \cdots, \alpha_n]^T$ with

$$\alpha_1 = 0$$

$$\alpha_2 = 0$$

$$\alpha_3 = -k_2 e_2 u_{d1}^{2-1}$$

$$\alpha_4 = -(e_2 - \alpha_2) - k_3(e_3 - \alpha_3)u_{d1}^{2-1} + \frac{1}{u_{d1}} \sum_{i=0}^{0} \frac{\partial \alpha_3}{\partial u_{d1}^{[i]}} u_{d1}^{[i+1]} + \sum_{i=2}^{2} \frac{\partial \alpha_3}{\partial e_i} e_{i+1}$$

$$\vdots$$

$$\alpha_n = -(e_{n-2} - \alpha_{n-2}) - k_{n-1}(e_{n-1} - \alpha_{n-1})u_{d1}^{2-1} + \frac{1}{u_{d1}} \sum_{i=0}^{n-4} \frac{\partial \alpha_{n-1}}{\partial u_{d1}^{[i]}} u_{d1}^{[i+1]} + \sum_{i=2}^{n-2} \frac{\partial \alpha_{n-1}}{\partial e_i} e_{i+1} \tag{6.5}$$

where $l = n - 2$, $u^{[i]}$ means the i-th derivative of u_{d1}, and k_i are positive constants.

The tracking control law is then defined as

$$u = \begin{bmatrix} u_{d1} + \eta \\ u_{d2} - s_{n-1}u_{d1} - k_n s_n + \sum_{i=0}^{n-3} \frac{\partial \alpha_n}{\partial u_{d1}^{[i]}} u_{d1}^{[i+1]} + \sum_{i-2}^{n-1} \frac{\partial \alpha_{n-1}}{\partial e_i} e_{i+1} \end{bmatrix} \tag{6.6}$$

$$\dot{\eta} = -k_0 \eta - k_1 s_1 - \sum_{i=2}^{n-1} s_i y_{i+1} + \sum_{i=3}^{n} s_i \sum_{i=2}^{i-1} \frac{\partial \alpha_i}{\partial e_i} y_{i+1} \tag{6.7}$$

where s is defined as $s = e - \alpha$.

6.3 Stability Analysis of the Closed-Loop System

The closed loop system of (6.2) and (6.6)-(6.7) can be written as

$$\dot{s}_1 = \eta$$

$$\dot{s}_2 = s_3 u_{d1} - k_2 s_2 u_{d1}^{2l} + \eta y_3$$

$$\dot{s}_3 = s_4 u_{d1} - s_2 u_{d1} - k_3 s_3 u_{d1}^{2l} + \eta\left(y_4 - \frac{\partial \alpha_3}{\partial e_2} y_3 \right)$$

$$\vdots$$

$$\dot{s}_{n-1} = s_n u_{d1} - s_{n-2} u_{d1} - k_{n-1} s_{n-1} u_{d1}^{2l} + \eta\left(y_n - \sum_{i=2}^{n-2} \frac{\partial \alpha_{n-1}}{\partial e_i} y_{i+1} \right)$$

$$\dot{s}_n = -k_n s_n - s_{n-1} u_{d1} + \eta \sum_{i=2}^{n-2} \frac{\partial \alpha_n}{\partial e_i} y_{i+1} \tag{6.8}$$

$$\dot{\eta} = -k_0 \eta - \Lambda_1 \tag{6.9}$$

where $\Lambda_1 = k_1 s_1 + \sum_{i=2}^{n-1} s_i y_{i+1} - \sum_{i=3}^{n} s_i \sum_{i=2}^{i-1} \frac{\partial \alpha_i}{\partial e_i} y_{i+1}$

Before given the stability analysis of the above closed-loop system, the following lemmas are required.

Lemma 1. Given a differentiable function $\phi(t)$: $R^+ \to R$, if $\phi(t) \in L_2$ and $\dot{\phi}(t) \in L_\infty$, then $\phi(t) \to 0$ as $t \to \infty$ [29].

Lemma 2. If a given a differentiable function $\phi(t)$: $R^+ \to R$ converges to some limit value when $t \to \infty$, and if the derivative $(d\phi/dt)(t)$ of this function is the sum of two terms, one being uniformly continuous and other one tending to zero when $t \to \infty$ [16].

The stability of the above closed-loop system is established in the following theorem.

Theorem: consider the mechanical system described by (6.2) subject to assumptions (A1), the controller specified by equations (6.6)-(6.7) ensures that all closed-loop signals are bounded. Furthermore, $y_1, y_2, \cdots, y_n$ asymptotically converge to $y_{d1}, y_{d2}, \cdots, y_{dn}$.

Proof: Let us consider the positive function

$$V(t) = \frac{1}{2}\left(\sum_{i=2}^{n} s_i^2 + k_1 s_1^2 + \eta^2 \right) \tag{6.10}$$

The time derivative of $V(t)$ along solution trajectory of (6.8) is

$$\dot{V}_1 = \sum_{i=2}^{n} s_i \dot{s}_i + k_1 s_1 \dot{s}_1 + \eta \dot{\eta}$$

$$= -\sum_{i=2}^{n-1} k_i s_i^2 u_{d1}^{2l} - k_n s_n^2 + \tilde{u}_2 s_n + \eta\left(\sum_{i=2}^{n-1} s_i y_{i+1} - \sum_{j=3}^{n-1} s_j \sum_{i=2}^{j-1} \frac{\partial \alpha_j}{\partial e_i} y_{i+1}\right) + k_1 s_1 \eta - k_0 \eta^2 - \eta \Lambda_1$$

$$= -\sum_{i=2}^{n-1} k_i s_i^2 u_{d1}^{2l} - k_n s_n^2 - k_0 \eta^2 \tag{6.11}$$

Based on (6.11), V' can be expressed as

$$V' \le -k_0 \eta^2 - k_n s_n^2 - \sum_{i=2}^{n-1} k_i s_i^2 u_{d1}^{2l}$$

$$\le 0 \tag{6.12}$$

The considered positive function $V(t)$ is thus non-increasing. This in turn implies that η and s are bounded, and $V(t)$ converges to limit value $V_{\lim}$. By the definition of s, it concludes that e is bounded. Using the assumption (A1) in the theorem, we obtain that y is bounded. In view of (6.8), (6.9), $\dot{s}$ and $\dot{\eta}$ are bounded. Thus, η and s are uniformly bounded. Next, we will prove that s, $\dot{s}$ and η tend to zero. Since V' is bounded and V is uniformly continuous, by Lemma 1, V' tend to zero. Therefore, η, $s u_{d1}^l (2 \le i \le n-1)$, s_n, tend to zero. By the assumption that $\liminf_{t \to \infty} |u_{d1}| > 0$, one concludes that $s_i (2 \le i \le n)$ tends to zero.

Differentiating $u_{d1}^l \eta$ yields

$$\frac{d}{dt} u_{d1}^l \eta = -k_1 u_{d1}^l s_1 + l u_{d1}^{l-1} \dot{u}_{d1} \eta - k_0 u_{d1}^l \eta - u_{d1}^l \left(\sum_{i=2}^{n-1} s_i y_{i+1} - \sum_{j=3}^{n} s_j \sum_{i=2}^{j-1} \frac{\partial \alpha_j}{\partial e_i} y_{i+1}\right)$$

where the first term is uniformly continuous and the other terms tend to zero. By Lemma 2, one concludes that $\frac{d}{dt}(u_{d1}\eta)$ converges to zero, which in turn implies that $u_{d1}^{l}s_1$ and s_1 tend to zero. Therefore, s and $\dot{s}$ tend to zero. To complete the proof and establish asymptotic convergence of the tracking error, it is necessary to show that $\{y,\dot{y}\} \to \{y_d,\dot{y}_d\}$ as $t \to \infty$. This is accomplished by the following arguments.

Based on the definition of α given in (6.5) and the relationship $s = e - \alpha$, it is obvious that $s_i = 0 (i = 1,2)$ yields $\lim_{t\to\infty} y_i = y_{di}$ and $\lim_{t\to\infty} \dot{y}_i = \dot{y}_{di} (i = 1,2)$ because of $\alpha_1 = \alpha_2 = 0$. From the boundedness of u_{d1}, one obtains that α_3 and $\dot{\alpha}_3$ converge to zero, which resulting in $\lim_{t\to\infty} y_3 = y_{d3}$ and $\lim_{t\to\infty} \dot{y}_3 = \dot{y}_{d3}$. The converges of α_3 and $\dot{\alpha}_3$ lead to the conclusion that α_4 and $\dot{\alpha}_4$ converge to zero, thus $\lim_{t\to\infty} y_4 = y_{d4}$ and $\lim_{t\to\infty} \dot{y}_4 = \dot{y}_{d4}$. Similarly, we can prove that $\lim_{t\to\infty} y_i = y_{di}$ and $\lim_{t\to\infty} \dot{y}_i = \dot{y}_{di} (5 \le i \le n)$. In summary, we have proved that $\{y,\dot{y}\} \to \{y_d,\dot{y}_d\}$ as $t \to \infty$. Thus, the theorem is valid.

Remarks: From above theorem, it can be seen that $\{y,\dot{y}\} \to \{y_d,\dot{y}_d\}$ as $t \to \infty$, which, in turn, implies that $\{q,\dot{q}\} \to \{q_d,\dot{q}_d\}$. Therefore, this chapter provides a general solution for the tracking problem of the nonholonomic systems.

6.4 Simulation results for Nomad 200

Given the desired trajectory $q_d = [2\cos t, 2\sin t, t]^T$, which is a circular path on the plane, the control objective is to determine a control law so that the trajectory $q = [x, y, \theta]^T$ follow q_d. The posture kinematic model of Nomad 200 is written as:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} -\sin\theta & 0 \\ \cos\theta & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} \tag{6.13}$$

where v_1 is the robot translational velocity component along Y_m and v_2 is the angular velocity.

Therefore, the kinematic system (6.13) can be written as

$$\begin{aligned} \dot{x} &= -v_1 \sin\theta \\ \dot{y} &= v_1 \cos\theta \\ \dot{\theta} &= v_2 \end{aligned} \tag{6.14}$$

Using a diffeomorphism transformation, $y = \Psi(q)$, and a state feedback, $v = \Omega_1(q)u$, which are defined as

$$\begin{aligned} y_1 &= \theta \\ y_2 &= x\cos\theta + y\sin\theta \\ y_3 &= -x\sin\theta + y\cos\theta \end{aligned}$$

$$u_1 = v_2$$

$$u_2 = v_1 - (x\cos\theta + y\sin\theta)v_2 \tag{6.15}$$

the above kinematic system is transferred into the chained form

$$\dot{y}_1 = u_1$$

$$\dot{y}_2 = y_3 u_1$$

$$\dot{y}_3 = u_2 \tag{6.16}$$

For the given $R(q)$, the desired trajectory $q_d = [2\cos t, 2\sin t, t]^T$ satisfies $\dot{q}_d = R(q_d)v_d$ with $v_{d1} = 2$ and $v_{d2} = 1$. Using a diffeomorphism transformation, the desired kinematic system $\dot{q}_d = R(q_d)v_d$ can be expressed as

$$y_{d1} = t$$

$$y_{d2} = 2$$

$$y_{d3} = 0$$

with $u_{d1} = 1$ and $u_{d2} = 0$. It is seen that y_d and u_d satisfy Assumption (A1) regarding the desired trajectories.

The tracking controller (6.6) is used.

$$\dot{\eta} = -k_0 - k_1 s_1 - y_3 s_2$$

$$u = \begin{bmatrix} u_{d1} + \eta \\ u_{d2} - s_2 u_{d1} - k_3 s_3 - k_2 e_2 u_{d1} - k_2 e_3 u_{d1} \end{bmatrix}$$

with

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} y_1 - y_{d1} \\ y_2 - y_{d2} \\ y_3 - y_{d3} \end{bmatrix} \qquad \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 + k_2 e_2 u_{d1} \end{bmatrix}$$

Figure 6.1 shows the system control block diagram, and Figure 6.2 the SIMULINK diagram of the control system.

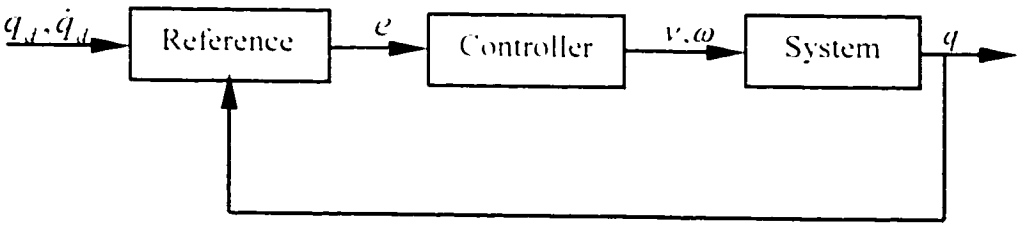

Figure 6.1 Closed-loop trajectory tracking control system

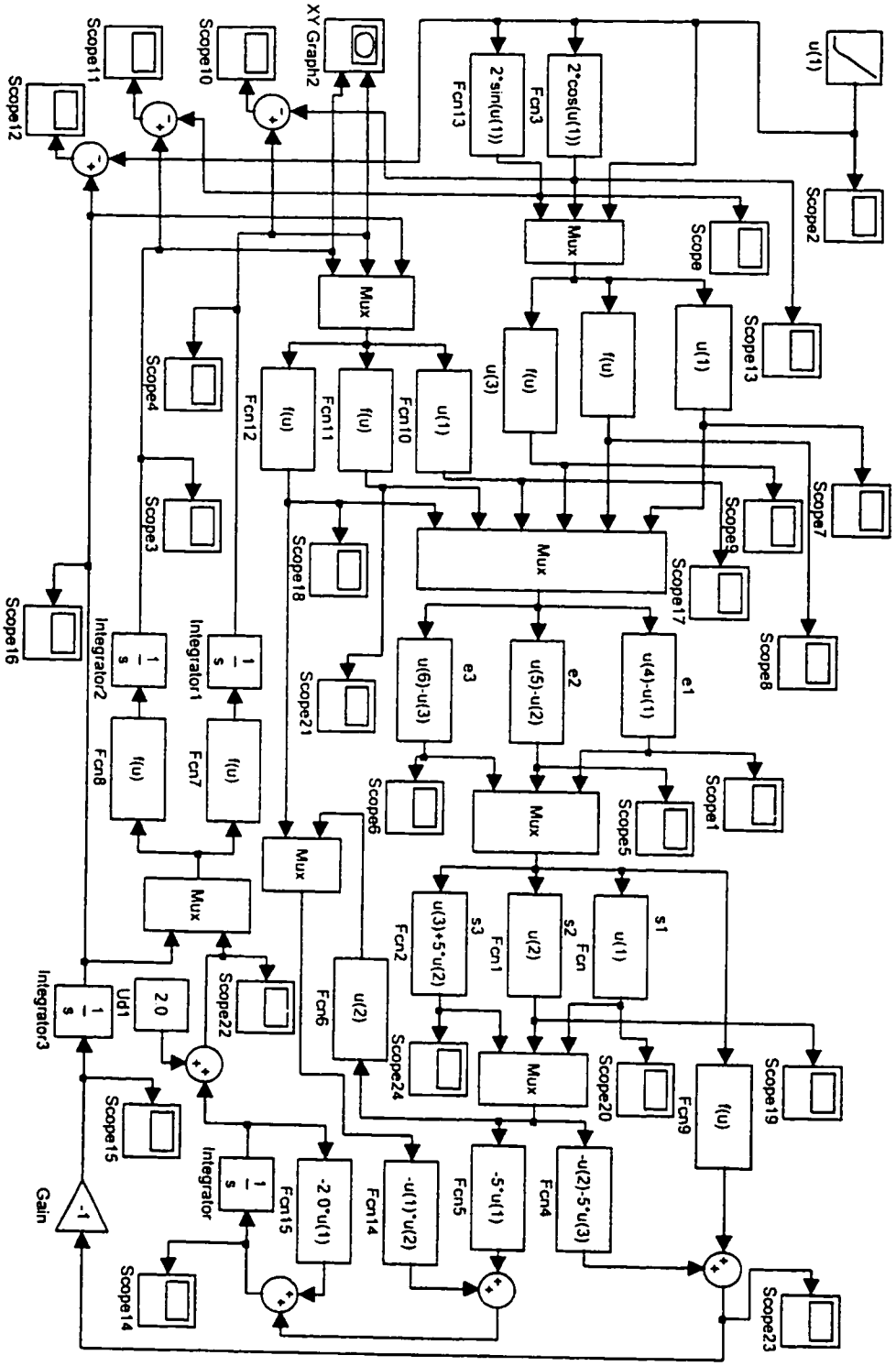

Figure 6.2 Simulink diagram of the control system

The initial positions and velocities of the wheeled robot are chosen as $q(0) = [3.0.0.5]$ and $\dot{q}(0) = [0.0.0]$. In the simulation, we set the control gain $k_0 = k_1 = k_2 = k_3 = 5$, $\eta(0) = 0$. The geometric trajectory of x via y is shown in Figure 6.3 and Figure 6.4 shows the trajectory tracking error of $x(t) - x_d(t)$. Figure 6.5 shows the trajectory tracking error of $y(t) - y_d(t)$, and the trajectory tracking error of $\theta(t) - \theta_d(t)$ is shown in Figure 6.6. Figure 6.7 shows the Nomad 200 mobile robot simulation result with r=254cm, x(0)=508cm, y(0)=0, $\theta(0) = 28.6^\circ$, and the result with r=508cm, x(0)=762cm, y(0)=0, $\theta(0) = 28.6^\circ$ is shown in Figure 6.8. These results confirm the validity of the proposed algorithm.

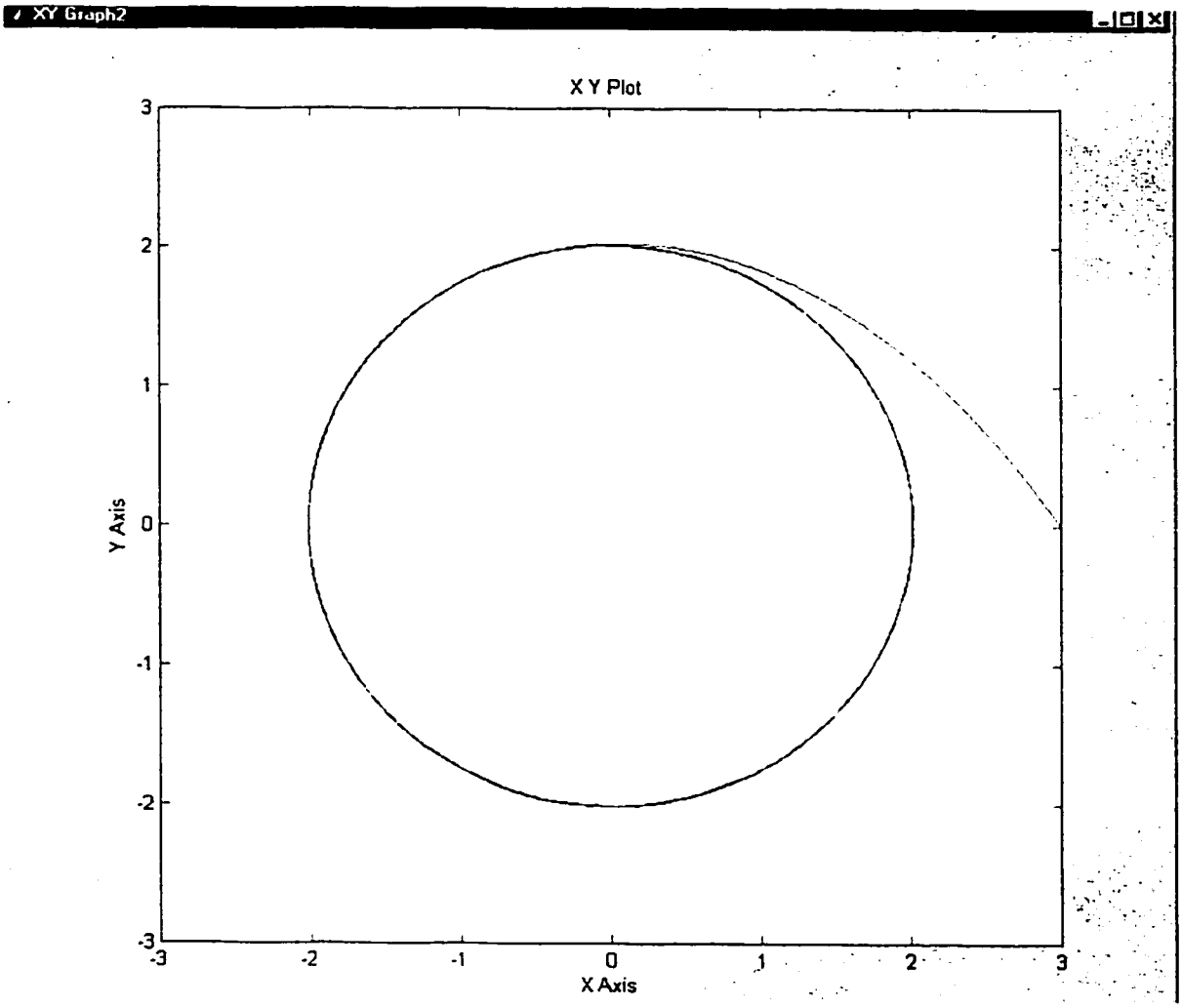

Figure 6.3 Geometric trajectory of x via y

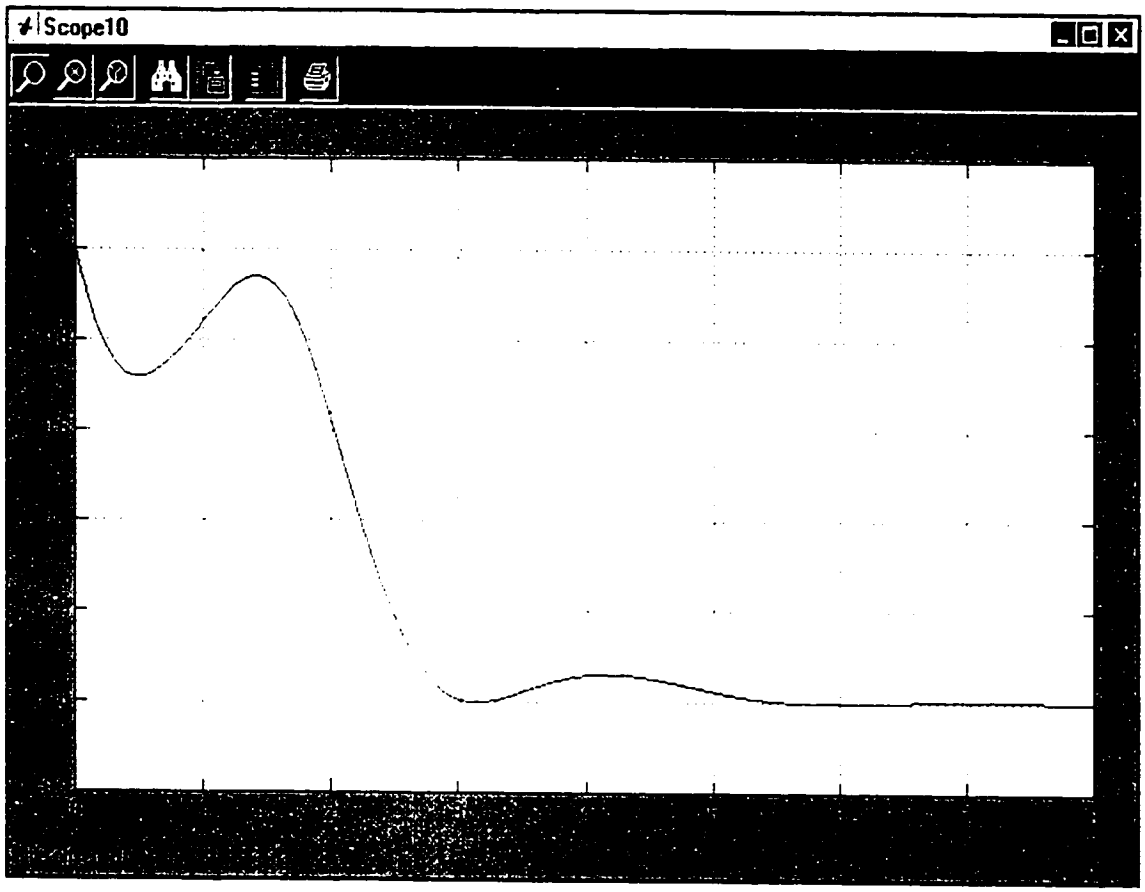

Figure 6.4 Tracking error of $x(t) - x_d(t)$

Figure 6.5 Tracking error of $y(t) - y_d(t)$

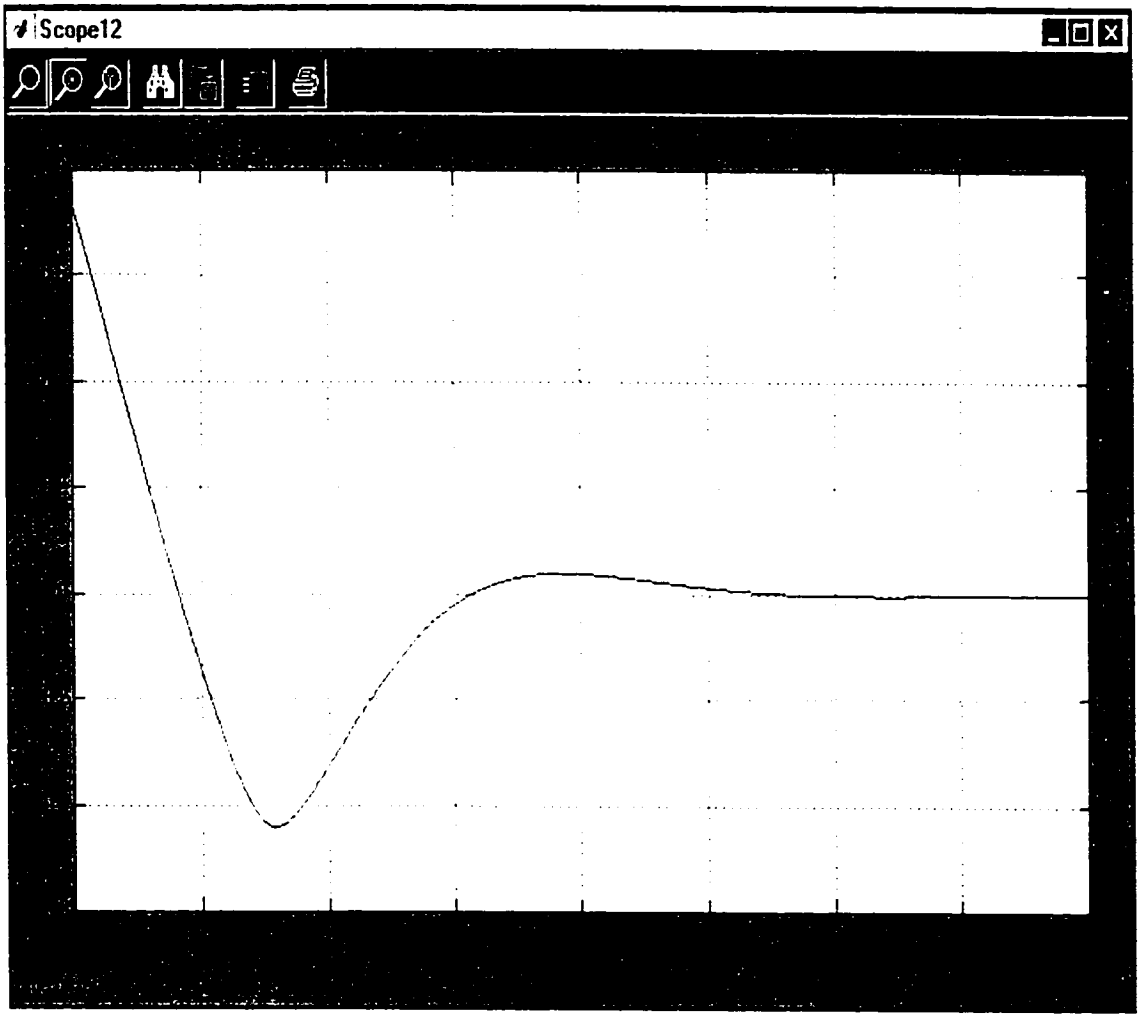

Figure 6.6 Tracking error of $\theta(t)-\theta_d(t)$

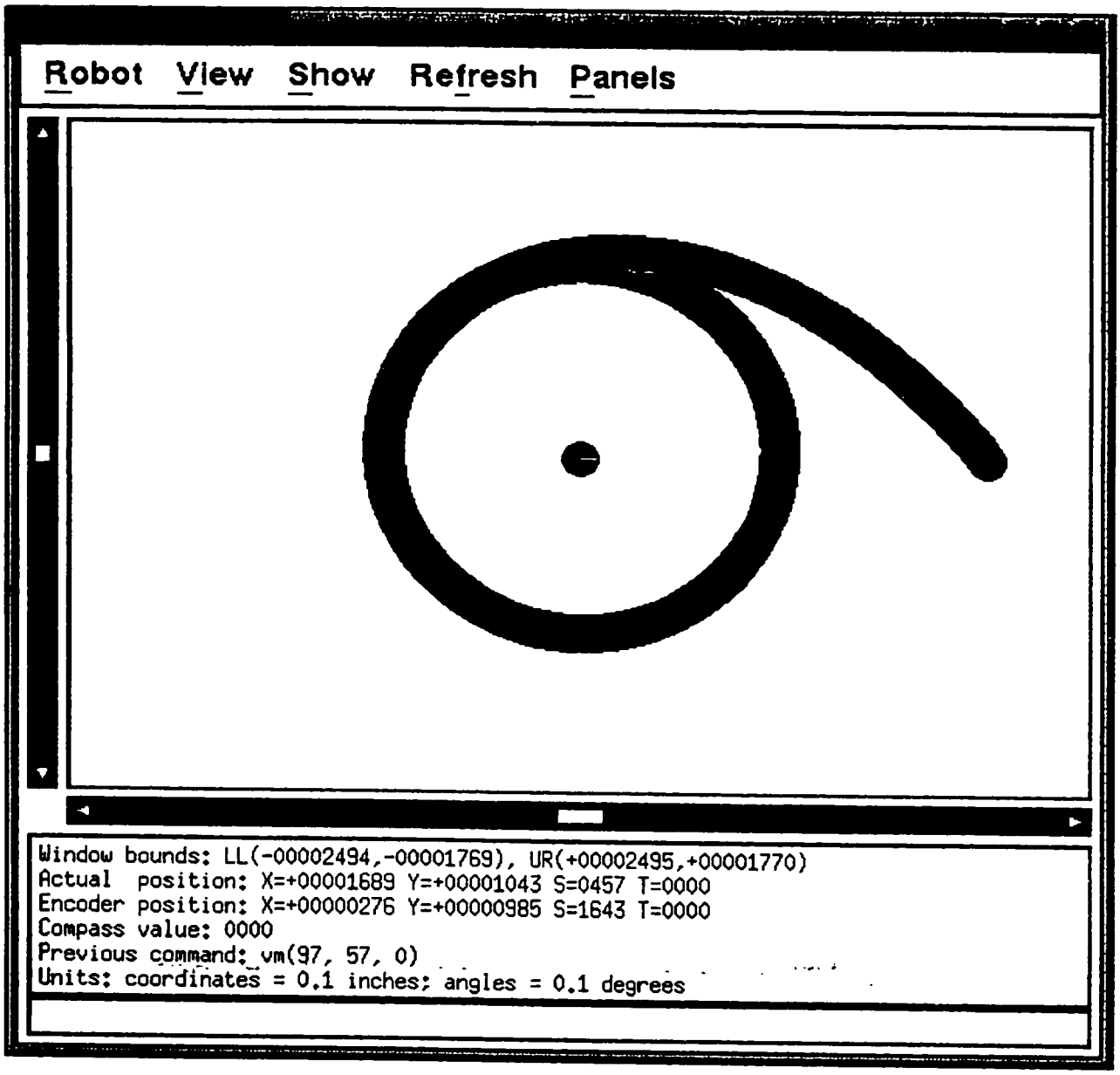

Figure 6.7 Nomad 200 simulation result with r=254cm, x(0)=508cm, y(0)=0, $\theta(0) = 28.6^{\circ}$

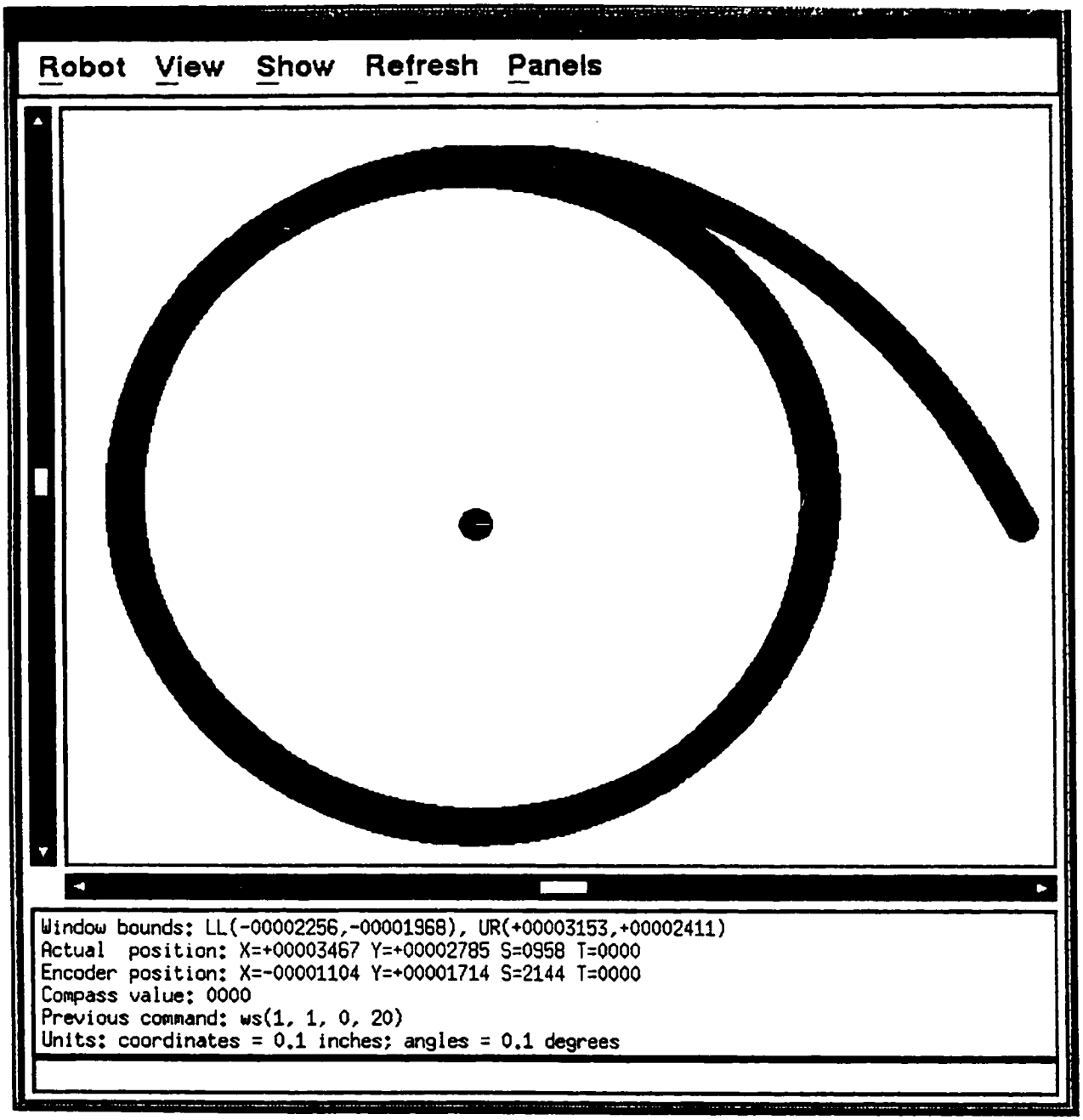

Figure 6.8 Nomad 200 simulation result with r=508cm, x(0)=762cm, y(0)=0, $\theta(0) = 28.6^{\circ}$

6.5 Implementation of the Algorithm on Nomad 200

The simulation on Nomad 200 is firstly carried out to verify the developed algorithm in section 6.4. In this section, the issue for a real-time implementation is addressed. The experiments are conducted at the "Fuzzy Systems Research Lab" at Concordia University. In the implementation, the true parameters of the control algorithm are set as:

T=0.005s, k0=k1=5.0, k2=k3=3.0, $\dot{q}(0) = [0,0,0]$.

The desired trajectory is a circular on the plane, which starts from the different original situation-the initial positions and orientations as given below:

Case 1:

radius r=100cm, x(0)=150cm, y(0)=0cm, $\theta(0) = 28.6^{\circ}$

Case 2:

radius r=150cm, x(0)=200cm, y(0)=0cm, $\theta(0) = 28.6^{\circ}$

Experimental result with r=100cm, x(0)=150cm, y(0)=0, $\theta(0) = 28.6^{\circ}$ is shown in Figure 6.9, and experiment results with r=150cm, x(0)=200cm, y(0)=0, $\theta(0) = 28.6^{\circ}$ can be seen in Figure 6.10.

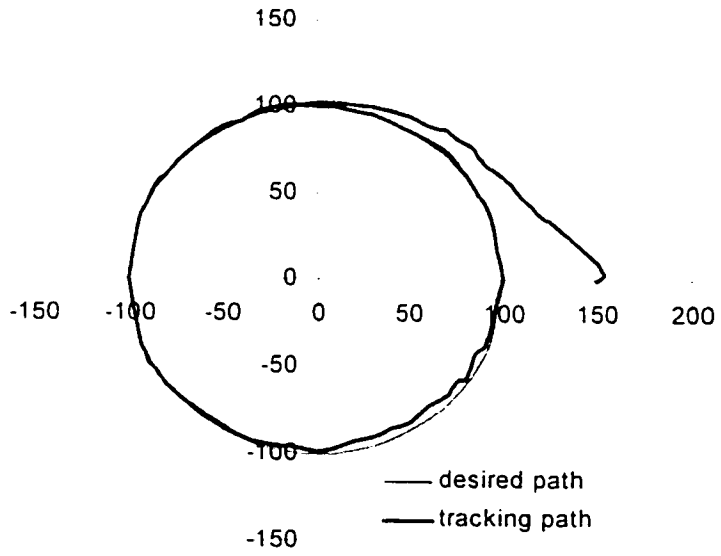

Figure 6.9 Experiment result with r=100cm, x(0)=150cm, y(0)=0, $\theta(0) = 28.6^{\circ}$

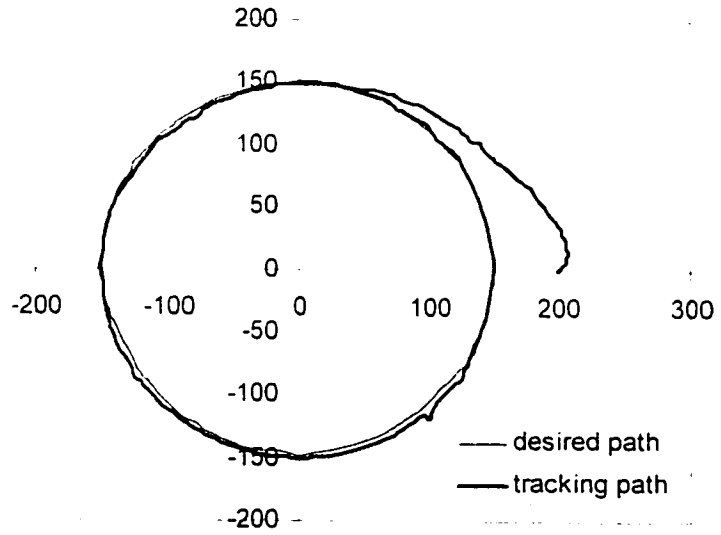

Figure 6.10 Experiment result with r=150cm, x(0)=200cm, y(0)=0, $\theta(0) = 28.6^{\circ}$

The control program and experimental data are given in Appendix II. The implementation results on the Nomad 200 system clearly verify that the tracking control algorithm worked globally in a stable and robust way. It is obvious that the tracking performance using the control algorithm developed in this chapter is superior to the one described in the Chapter 5. Therefore the proposed time-varying adaptive control algorithm is suitable for controlling this robot.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

In this thesis various structural properties regarding controllability, stabilizability and nonholonomy of the kinematic state models of WMRs have been given, taking into account the restriction to robot mobility induced by the constraints. By using the concepts of *degree of mobility* and *degree of steerability*, this study provides a general and unifying presentation of the modeling issue of WMRs. Several example of derivation of posture kinematic models for WMRs are available in the literature for particular prototypes of wheeled mobile robots. Here, a more general viewpoint is adopted for the consideration of a general class of wheeled mobile robots with an arbitrary number of wheels of different types. As an example, a three-wheel mobile robot-Nomad 200 is used in the experiment and its structural properties were described in details.

We proposed one control algorithm in this thesis. Both of them are feedback tracking control, given the desired trajectory q_d, which is a circular path on the plane.

The purpose of the control objective is to determine a feedback control law so that the trajectory q follows q_d. The first control algorithm is to control a wheeled nonholonomic robot by using a "virtual vehicle approach", which was shown to be robust with respect to error and disturbances. The second one is adaptive tracking control strategy of a nonholonomic mechanical system, which just matches the noholonomic wheeled mobile robot-Nomad 200.

With the investigation of the nonholonomic WMR kinematic model, feedback control design method for trajectory tracking of the general nonholonomic WMR is proposed. We found sufficient conditions to achieve the tracking by using the feedback control laws. The application to type (2,0) of Nomad 200 mobile robot shows that the proposed algorithm was practically applicable. It could be realized by adjusting the input parameters in the control functions according to the behavior of the desired trajectory. Simulation results show that the proposed control schemes worked well and are robust against the uncertain inertia parameters, unknown disturbance and external errors.

Simulation results with SIMULINK software and Nomad simulator, and also the experimental results on Nomad 200 confirm the validity of the proposed algorithm. The test runs, where the robot track the circle from different initial position and configurations, clearly verify that our proposed tracking control algorithm works globally in a stable and robust way.

7.2 Future Work

In this study, we discuss and present only kinematic WMR models, and only perform experiments by using car-like robot and unicycle robot model. The unicycle robot belongs to the class of Type (2,0). Furthermore, we did not take friction into account. Therefore, the future work could validate the other four types of kinematic mobile robot models, and develop the dynamic model of the five-type mobile robot.

References

[1] R. W. Brockett, "Asymptotic Stability and Feedback Stabilization," in Differential Geometric Control Theory, R. W. Brockett, R. S. Millman, and H. J. Sussmann (eds.), Birkhuser, 1983.

[2] I. Kolmanovsky and N. H. McClamroch, "Developments in Nonholonomic Control Problems," IEEE Control Systems Magazine, vol. 15, No. 6, pp. 20-36, 1995.

[3] I. Kolmanovsky,d M. Reyhanoglu, and N. H. McClamroch, "Switched Mode Feedback Control Laws for Nonholonomic Systems in Extended Power Form," Systems & Control Letters, vol. 27, pp. 29-36, 1996.

[4] A. M. Bloch, M. Reyhanoglu, and N. H. McClamroch, "Control and Stabilization of Nonholonomic Dynamic Systems," IEEE Trans. on Automatic Control, Vol.37, pp.1746-1757, 1992.

[5] U. Nehmzow "Mobile Robotics: A Practical Introduction".

[6] A. Bloch and N. H. McClaamroch, "Control of Mechanical Systems with Classical Nonholonomic Constraints".Proceedings of the 28th IEEE Coference on Decision and Control, pp.201-205, 1989.

[7] A. Kapitanovsky, A. A. Goldenberg, and J. K. Mills, "Dynamic Control and Motion Planning Techniqure for a Class of Nonlinear Systems with Drift," Systems & Control Letters, vol. 21, pp. 363-369, 1993.

[8] R. M'Closkey and R. Murray, "Extending Exponential Stabilizers for Nonholonomic Systems from Kinematic Controllers to Dynamic controllers," Proc. IFAC Symp. Robot Control, 1994.

[9] B. Maschke and A. van der Schaft, "A Hamiltonian Approach to Stabilization of Nonholonomic Mechanical Systems," Proc. 33rd IEEE Conf. on Decision and Control, 1994.

[10] C.-Y. Su and Y. Stepanenko, "Robust Motion/Force Control of Mechanical Systems with Classical Nonholonomic Constraints," IEEE Trans. on Automatic Control, vol. 39, No. 3, 1994.

[11] R. M. Murray, " Nonlinear Control of Mechanical Systems: A Lagrangian Perspective," IFAC Symposium on Nonlinear Control Systems Design, pp.349-360, 1995.

[12] R. Colbaugh, E. Barany, and K. Glass, "Adaptive Control of Nonholonomic Robotic Systems," Journal of Robotic Systems, vol. 15, pp.365-393, 1998.

[13] W. Dong, W. L. Xu, and W. Huo, "Trajectory Tracking Control of Dynamic Nonholonomic Systems with Unknown Dynamics," Int. J. of Robust and Nonlinear Control, vol. 9, pp. 905-922, 1999.

[14] J.-M. Yang and J.-H. Kim, "Sliding Mode Control for Trajectory Tracking of Nonholonomic Wheeled Mobile Robots," IEEE Trans. on Robotics and Automation, vol. 15, pp. 578-587, 1999.

[15] Neimark, Ju. I. And Fufaev, N. A., Dynamics of Nonholonomic Systems, Translations of Mathematical Monographs, vol. 33, AMS, 1972.

[16] C. Samson, "Control of Chained Systems Application to Path Following and Time-varying Point-stabilization of Mobile Robots," IEEE Trans. on Automatic Control, vol. 40, pp. 64-77, 1995.

[17] M. Aicardi, G. Casalino, A. Bicchi, "Closed-loop Smooth Steering of Unicycle-like Vehicles," Proceedings of the 33rd IEEE Decision and Control Conference, pp.2455-2458, 1994.

[18] Y. Stepanenko and C.-Y. Su, "Adaptive Motion/force Control of Mechanical Systems with Nonholonomic Phaffian Constraints," in Proc. American Control Conf., pp. 375-379, 1995.

[19] N. Sarkar, X. Yun, V. Kumar, "Control of Contact Interactions with Acatastatic Nonholonomic Constraints," International J. of Robotics Research, vol. 16, pp. 357-374, 1997.

[20] Z.-P. Jiang and H. Nijmeijer, " A Recursive Technique for Tracking Control of Nonholonomic Systems in Chained Form, " IEEE Trans. On Automatic Control, vol. 42, pp. 265-279, 1999.

[21] C.-Y. Su, Y. Stepanenko, and A. A. Goldenberg, "Reduced Order Model and Robust Control Architecture for Mechanical Systems with Nonholonomic Pfaffian Constraints," IEEE Trans. On Systems, Man, and Cybernetics- Part A: Systems and Humans, vol.29, 1999.

[22] O. J. Sordalen and C. Canudas de Wit, "Exponential Control Law for a Mobile Robot: Extension to Path Following," Proc. Of 1992 IEEE int. Conf. on Robotics and Automation, pp 2158-2163, 1992.

[23] "Language Reference Manual", Software Version 2.6, March 11, 1997.

[24] "User's Manual", Software Version 2.6, March 11, 1997.

[25] G. Walsh and L. G. Bushnell, "Stabilization of Multiple Input Chained Form Control Systems," Systems & Control Letters, vol. 25, pp. 227-234, 1995.

[26] M. A. Benayad, G. Campion, V. Wertz, and M. E. Achhab, "Steering a Mobile Robot: Selection of A Velocity Profile Satisfying Dynamical Constraints," Asian Journal of Control, vol. 2, pp. 219-229, 2000.

[27] Z. P. Jiang and J.-B. Pomet, "Global Stabilization of Parametric Chanied-Form Systems by Time-varying Dynamic Feedback," Int. J. of Adaptive Control and signal Processing, vol. 10, pp. 47-59, 1996.

[28] M. W. Spong, "On the Robust Control of Robot Manipulators," IEEE Trans. on Automatic Control, vol. 37, pp. 1782-1786, 1992.

[29] V. M. Popov, Hyperstability of Control Systems, Springer-Verlag, 1973.

[30] M. Aicardi, G. Casalino, A. Bicchi, A. Balestrino "Closed- loop Steering of Unicycle-like Vehicles via Lypunov Techniques" IEEE Robostics & Automation magazine, pp. 27-35, March 1995.

[31] M. Egerstedt, X. Hu, and A. Stotsky "Control of a Car-like Robot Using a Virtual Vehicle Approach" Proceedings of the 37th IEEE Conference on Decision & Control, Tampa, Florida USA, pp.1502-1507, December, 1998.

[32] J. L. Jones, A. M. Flynn, B. A. Seiger "Mobile Robots: Inspiration to Implementation", Second Edition.

[33] G. Campion, B. d'Andrea-Novel and G. Bastin, "Controllability and State Feedback Stabilizability of Nonholonomic Mechanical Systems," Advanced Robot Control, C. Canudase de Wit, editor, LNCIS 162, Springer, pp. 106-124, 1991.

[34] N. Getz, " Control of Balance for A Nonlinear Noholonomic Non-minimum Phase Model of a Bicycle," Proceedings of the American Control Conference, pp. 148-151, 1994.

[35] A. Isidori, Nonlinear Control System, 2nd edition, Springer-verlag, 1989.

[36] A. Kapitanovsky, A. A. Goldenberg and J. K. Mills, "Dynamic Control and Motion Planning Technique for A Class of Nonlinear Systems with Drift," Systems & Control Letters, vol. 21, pp. 363-369, 1993.

[37] C. C. de Wit, "Trends in Mobile Robot and Vehicle Control", Control Problems in Robotics, Lecture Notes in Control and Information Sciences 230, pp.151-176, eds. B.Siciliano and K.P.Valavanis, Springer-Verlag, London, 1998.

[38] G. Oriolo and Y. Nakamura, "Control of Mechanical Systems with Second-order Nonholonomic Constraints: Underactuated Manipulators," Proceedings of the 30th IEEE Conference on Decision and Control, pp. 306-308, 1993.

[39] C. Rui and N. H. McClamroch, "Stabilization and Asymptotic Path Tracking of a Rolling Disk," Proceedings of the 34th IEEE Conference on Decision and Control, 1995.

[40] C. Samaon, "Time-varying Feedback Stabilization of a Car-like Wheeled Mobile Robots" International Journal of Robotics Research, vol. 12, No. 1, pp. 55-66, 1993.

[41] N. Sarkar, X. Yun and V. Kumar, "Dynamic Path Following: A New Control Algorithm for Mobile Robots " Proceedings of the 32nd IEEE Conference on Decision and Control, pp. 2670-2675, 1993.

[42] G. C. Walsh, D. Tillbury, S. Sastry, R. Murray, and J. P. Laumond, "Stabilization of Trajectories for Systems with Nonholonomic Constraints," IEEE Trans. on Automatic Control, vol. 39, pp. 216-222, 1994.

[43] C. C. de Wit, B. Siciliano and G. Bastin, "Theory of Robot Control".

[44] G. Gampion, G. Bastin and B. D'Andrea-Novel "Structure Properties and Classification of Kinematic and Dynamic Model of Wheeled Mobile Robots", IEEE Transactions on Robotics and Automation, Vol.12, No.1, Feb. 1996.

[45] M. Fliess, J. Levine, P. Martin,and P. Rouchon, "Flatness and Defect on Non-linear Systems: Introductory theory and Examples," Int. J. of Control, vol. 61, No.6,pp. 1327-1361, 1995.

[46] Guldner J. and V. Utkin: Stabilization of Non-holonomic Mobile Robots Using Lyapunov Function for Navigation and Sliding Mode Control, Proc. of the 33rd CDC, Lake Buena Vista, FL- pp. 2967-2972, December, 1994.

[47] A. M. Bloch, N. H. McClaamroch and M. Reyhanoglu, "Controllability and Stabilizability of a Nonholonomic Control System," proceedings of the 29th IEEE Conference on Decision and Control, pp.1312-1314, 1990.

[48] R. M. Murray and S. S. Sastry, "Nonholonomic Motion Planning: Steering Using Sinusoids," IEEE Trans. on Automatic Control, vol. 38, No. 5, pp. 700-716, 1993.

Appendix I

Program for Algorithm I and Experiment Data

```
/* This program will connect to the robot, configure locomotion, and
move the robot to track a circle on the plane with different initial
position by using various command.
* It assume that a server is running and connect to it.
*  To compile:  gcc -o motiontest motion.c Nclient-linux.o -
DSIMULATION=1 OR
*    gcc -o motiontest motion.c Nclient-linux.o -DSIMULATION=1
*/

#include<math.h>
#include<stdio.h>
#include "Nclient.h"
#include<stdlib.h>
#define PI 3.14159
#define ROBOT_ID  1  /* Currently, only robot #1 allowed */

#ifndef SIMULATION
#define SIMULATION 0
#endif

int main()
{
double r,a,b,c,d,e,f,x1,y1,T,k,p,q,s,R;
double fe1;
double beta,theta,delta1,fei1,beta1;
int delta,beta1,angle;
int feid,x,y,v;
int r1,x2,y2,v1;
double xd,yd;
double deltax, deltay;
/* connection */
SERV_TCP_PORT=7019;  /* matches the number given in world.setup  */
strcpy(SERVER_MACHINE_NAME, "Fuzzy3.concordia.ca");/* The machine the
server is running on */
//      connect_robot(1);
/* establish connection */
```

```
if (!connect_robot(ROBOT_ID))
exit(0);
if(SIMULATION)
simulated_robot();      /* Commands will be sent to simulator */
else
real_robot();
//   connect_robot(1);  /* Commands will be sent to real robot  */

/* initialize the robot */
//    printf("Zeroing...\n");
//    zr(); /* Zero the robot */
/* Inintialize motion parameters */
ac(300,300,0);    /* translation, steering, turret accelerations in
.1in/s2, .1deg/s2 */
sp(200,450,0);    /* translation, steering, turret speed in .1 in/s,
.1deg/s */
printf("Please enter the radius r:");
scanf("%d",&r1);
printf("Please enter the coordinate x:");
scanf "%d",&x2);
printf("Please enter the condinate y:");
scanf("%d",&y2);
printf("Please enter the velocity v:");
scanf("%d",&v1);

r=r1;
x=x2;
y=y2;
v=v1;
k=0.8;
printf(" Hit any key to start, it will wait for the motion to
stop\n");
getchar();
dp(x,y);
x=State[STATE_CONF_X];
y=State[STATE_CONF_Y];
```

```
beta1=State[STATE_CONF_STEER];
fei=0.0;
beta1=0;
T=1.0;
delta=0;
d=1.0;
beta=0.0;
beta2=(double)(beta1*PI/1800);
x1=x;
y1=y;

xd=r*cos(fei);
yd=r*sin(fei);

if(x1!=0.0)
beta=(atan(y1/x1));

theta=beta;
do
{
deltax=(double)(x1-xd);
deltay=(double)(y1-yd);
if(deltax==0.0)
{
if(y>=0)
fei=PI/2;
if(y<0)
fei=-PI/2;
}
else
fei=atan(deltay/deltax);

if(theta>=0.0 && theta<=PI/2)
{
if((x1*x1+y1*y1)>r*r && x1>=0.0)
{
```

```
if(deltax>0.0 && deltay<0.0)
fei=PI+fei;
if(deltax>0.0 && deltay>0.0)
fei=PI+fei;
if(deltax<0.0 && deltay>0.0)
fei=2*PI+fei;
}
if((x1*x1+y1*y1)<r*r && x1>=0.0)
{
if(deltax<0.0 && deltay>0.0)
fei=2*PI+fei;
if(deltax<0.0 && deltay<0.0)
fei=fei;
if(deltax>0.0 && deltay<0.0)
fei=PI+fei;
}
}
if(theta>=PI/2 && theta<=PI)
{
if((x1*x1+y1*y1)<r*r && x1<=0.0)
{
if(deltax<0.0 && deltay<0.0)
fei=2*PI+fei;
if(deltax>0.0 && deltay<0.0)
fei=PI+fei;
if(deltax<0.0 && deltay>0.0)
fei=2*PI+fei;
}
if((x1*x1+y1*y1)<r*r && x1<=0.0)
{
if(deltax>0.0 && deltay>0.0)
fei=PI+fei;
if(deltax>0.0 && deltay<0.0)
fei=PI+fei;
if(deltax<0.0 && deltay<0.0)
fei=fei;
```

```
}
}
if(theta>=PI && theta<=3*PI/2)
{
if((x1*x1+y1*y1)>r*r && y1<=0.0)
{
if(beta1<900 && beta1>0)
beta1=beta1+3600;
if(deltax<0.0 && deltay>0.0)
fei=2*PI+fei;
if(deltax<0.0 && deltay<0.0)
fei=fei+2*PI;
if(deltax>0.0 && deltay<0.0)
fei=2*PI+fei;
}
if((x1*x1+y1*y1)<r*r && y1<=0.0)
{
if(deltax>0.0 && deltay>0.0)
fei=PI+fei;
if(deltax>0.0 && deltay<0.0)
fei=PI+fei;
if(deltax<0.0 && deltay>0.0)
fei=2*PI+fei;
}
}
if(theta>=3*PI/2 && theta<=2*PI)
{
if(beta1<900 && beta1>0)
beta1=beta1+3600;

if((x1*x1+y1*y1)>r*r && x1>=0.0)
{
if(deltax>0.0 && deltay<0.0)
fei=PI+fei;
if(deltax<0.0 && deltay<0.0)
fei=2*PI+fei;
```

```
if(deltax>0.0 && deltay>0.0)
fei=PI+fei;
}

if((x1*x1+y1*y1)<r*r && x1>=0.0)
{
if(deltax>0.0 && deltay>0.0)
fei=PI+fei;
if(deltax<0.0 && deltay>0.0)
fei=2*PI+fei;
if(deltax<0.0 && deltay<0.0)
fei=2*PI+fei;
}
}
p=sqrt(deltax*deltax+deltay*deltay);
q=exp(-p/d);
s=(double)(1.0*v*p*q);

b=(double)(-(v)/r);
c=sin(theta-beta2);
e=b*c;
f=s+e;
if(f<0.0)
f=0.0;
theta=theta+f*T;
xd=r*cos(theta);
yd=r*sin(theta);
feid=(int)(fei*1800/PI);

delta=int(-k*(beta1-feid));
vm(v,delta,0);
x=State[STATE_CONF_X];
y=State[STATE_CONF_Y];
beta1=State[STATE_CONF_STEER];
beta2=(double)(beta1*PI/1800);
x1=x;
```

```
yl=y;
theta=atan(yl/xl);

if(xl>=0.0 && yl>=0.0)
theta=theta;
if(xl<=0.0 && yl>0.0)
theta=PI+theta;
if(xl<0.0 && yl<0.0)
theta=PI+theta;
if(xl>=0.0 && yl<0.0)
theta=2*PI+theta;
R=sqrt(xl*xl+yl*yl);

printf("The beta2 is:%6.5f\n",beta2);
printf("The xl is: %6.2f\n",xl);
printf("The yl is: %6.2f\n",yl);
printf("The R is: %6.2f\n",R);
printf("The xd is: 6.2f\n",xd);
printf("The yd is: 6.2f\n",yd);
printf("The deltax is: 6.2f\n",deltax);
printf("The deltay is: 6.2f\n",deltay);
printf("The v is: 4d\n",v);
printf("The f is:%6.5f\n",f);
printf("The theta is: 6.5f\n",theta);
printf("The fei is: 6.5f\n",fei);
printf("The betal is:%4d\n",betal);
printf("The feid is:%4d\n",feid);
printf("The delta is: %4d\n\n\n",delta);
}while(theta<1.8*PI);

printf(" End of demo, quitting the server...\n");
st();
disconnect_robot(ROBOT_ID);   /*     Kill     the     server;     just
disconnect_robot if the server is to be used again */
return(1);
}
```

Case 1 (r=100cm, x(0)=10cm, y(0)=5.5cm, $\theta(0) = 10^{\circ}$)

X(cm)	y1(cm)	y2(cm)	y3(cm)	y4(cm)	y5(cm)
105			58	58	
100	0	0	47.5	66	
95	31.22	-31.22	44.5	69	
90	43.58	-43.58	42	69.5	-25.5
85	52.68	-52.68	40	69	-43
80	60	-60	37.5	67.5	-51
75	66.14	-66.14	35.5	68.5	-58.5
70	71	-71	33.5	70	-64
60	80	-80	29	75	-72.5
50	86.6	-86.6	24.5	91	-81
40	91.68	-91	19.5	97	-88
30	95.39	-95.39	15	101.5	-92.5
20	97.97	-97.97	10.5	103.5	-96
10	99.5	-99.5	5.5	104	-98.5
0	100	-100		105	-100.5
-10	99.5	-99.5		104	-100
-20	97.97	-97.97		102.5	-98.5
-30	95.39	-95.39		100	-96.5
-40	91.68	-91.68		97	-92
-50	86.6	-86.6		90.5	-88
-60	80	-80		85	-81
-70	71	-71		76.5	-71
-75	66.14	-66.14		70	-65
-80	60	-60		62	-57
-85	52.68	-52.68		57	-52
-90	43.56	-43.58		47	-47
-95	31.22	-31.22		41	-36.5
-100	0	0		24	-21
-105				0	0

Case2 (r=100cm, x(0)=150cm, y(0)=13cm, $\theta(0) = 5^{\circ}$)

X(cm)	y1(cm)	y2(cm)	y3(cm)	y4(cm)	y5(cm)	y6(cm)	y7(cm)
170			31	31			
166			24.5	33.5			
162			20.5	37			
158			18	36.5			
152			13	35			
148				34			
144				33.3			
140				32			
136				31			
132				29.5			
124				29			
120				28			
116				26			
108				25			
104				24			
100	0	0		23.5			
95	31.22	-31.22		20.5			
90	43.58	-43.58		20	57	57	
85	52.68	-52.68		20.5	52	66	
80	60	-60		21	45	70	-57.5
75	66.14	-66.14		24	40	76.5	-66
70	71	-71		28	28	77.5	-70
60	80	-80				79	-80
50	86.6	-86.6				82.5	-88
40	91.68	-91				90.5	-92.5
30	95.39	-95.39				96	-97
20	97.97	-97.97				98.5	-101
10	99.5	-99.5				99.5	-103
0	100	-100				100	-105
-10	99.5	-99.5				99.3	-104
-20	97.97	-97.97				99	-104.5
-30	95.39	-95.39				97.5	-102.5
-40	91.68	-91.68				94	-97
-50	86.6	-86.6				92	-92
-60	80	-80				84	-88.5
-70	71	-71				76.5	-79.5
-75	66.14	-66.14				70.5	-74
-80	60	-60				65	-69.5
-85	52.68	-52.68				58	-61
-90	43.56	-43.58				48.5	-54.5
-95	31.22	-31.22				37	-44
-100	0	0				27	-29

Case3 (r=100cm. x(0)=150cm. y(0)=80cm. $\theta(0) = 0^\circ$)

X(cm)	y1(cm)	y2(cm)	y3(cm)	y4(cm)	y5(cm)	y6(cm)	y7(cm)
177			91	91			
175			87	96.5			
173			84	97.5			
170			82.5	99			
165			82	98.5			
160			81.5	96			
155			81	91			
150			80	87			
145				84			
140				81.5			
135				77.5			
130				74			
125				71			
120				68.5			
115				65			
110				62.5			
105				59.5			
100	0	0		56.5			
95	31.22	-31.22		53			
90	43.58	-43.58		50			
85	52.68	-52.68		48			
80	60	-60		47			
77	64.38	-64.38		47.5	84	84	
76	65.01	-65.01		48	79	86	
75	66.14	-66.14		48.5	75.5	88	-68
74	67.31	-67.31		49	74	89.5	-69
73	68.34	-68.34		49.5	71.5	92	-71.5
72	69.4	-69.4		50	68.5	92.5	-72
71	70.42	-70.42		50.5	67	93.5	-72.5
70	71	-71		51	64.5	93.5	-73
69	72.38	-72.38		52	57.5	94	-75
68	73.32	-73.32		56.5	56.5	94.5	-78
60	80	-80				94.5	-80
50	86.6	-86.6				92.5	-88.5
40	91.68	-91				93	-92.5
30	95.39	-95.39				96	-96
20	97.97	-97.97				99.5	-99.5
10	99.5	-99.5				101.5	-101.5

0	100	-100				102	-102
-10	99.5	-99.5				101	-100.5
-20	97.97	-97.97				98.5	-99.5
-30	95.39	-95.39				95	-97.5
-40	91.68	-91.68				90.5	-94
-50	86.6	-86.6				85.5	-90
-60	80	-80				79.5	-84
-70	71	-71				68	-73.5
-75	66.14	-66.14				62	-69
-80	60	-60				56	-63
-85	52.68	-52.68				50.5	-56
-90	43.56	-43.58				44	-47.5
-95	31.22	-31.22				30	-33
-100	0	0				11	-11
-103						0	0

Appendix II

Program for Algorithm II and Experiment Data

```
/* This program will connect to the robot, configure locomotion, and
move the robot to track a circle on the plane by using various
command.
* It assume that a server is running and connect to it.
*  To compile:  gcc -o motiontest motion.c Nclient-linux.o -
DSIMULATION=1 OR
gcc -o motiontest motion.c Nclient-linux.o -DSIMULATION=1
*/
#include<math.h>
#include<stdio.h>
#include "Nclient.h"
#include<stdlib.h>
#define PI 3.14159
#define ROBOT_ID  1  /* Currently, only robot #1 allowed */
#ifndef SIMULATION
#define SIMULATION 1
#endif

int main()
{
double y1,y2,y3,yd1,yd2,yd3,e1,e2,e3,s1,s2,s3;
double nita,nita0,hita_dot;
double theta,theta0,xd_t0,yd_t0,theta_dot0;
double xd1,yd1,delt,x,y;
int t,r1,x1,y1,v1,w1,r2,v2,angle,theta1,v11,v21;
double x0,y0,x_dot,y_dot,xd,yd,ud,wd,v,w;
double k,k1,k1,k2,k3,t,f,T,deltax,deltay,deltath;
/* connection */
SERV_TCP_PORT=7019;  /* matches the number given in world.setup  */
//    strcpy(SERVER_MACHINE_NAME,     "Fuzzy3.concordia.ca");/*     The
machine the server is running on */
//      connect_robot(1);
/* establish connection */
if (!connect_robot(ROBOT_ID))
exit(0);
if (SIMULATION)
```

```
simulated_robot();      /* Commands will be sent to simulator */
else
real_robot();
/* Inintialize motion parameters */
ac(300,300,0);     /* translation, steering, turret accelerations in
.1in/s2, .1deg/s2 */
sp(200,450,0);     /* translation, steering, turret speed in .1 in/s,
.1deg/s */

printf("Please enter the radius r:");
scanf("%d",&r1);
printf("Please enter the initial coordinate x:");
scanf("%d",&x1);
printf("Please enter the initial coordinate y:");
scanf("%d",&y1);
printf("Please enter the initial angular theta:");
scanf("%d",&theta1);
printf("Please enter the initial translational speed:");
scanf("%d",&v11);
printf("Please enter the initial angular speed:");
scanf("%d",&v21);

r=r1/1000;
x0=x1/1000;
y0=y1/1000;
v1=v11/1.0;
v2=v21/1.0;
theta0=theta1/10.0;
printf("The theta1 is: %d\n",theta1);
printf("The theta0 is: %6.3f\n",theta0);
T=0.005;
t=0.0;
f=1.0;
angle=int(theta0*10.0/9);
k0=k1=5.0;
k2=k3=3.0;
```

```
printf("The angle is:%d\n",angle);

xdot0=yact =theta_dot0 ...;
hita0=0.0;
printf(" Hit any key to start, it will wait for the motion to
stop\n");
getchar();

dp x1,y1);
ws 1,1,0,2 );
da(angle+900,0);
ws(1,0,0,2));

xd=r*cos(f*t);
yd=r*sin(f*t);

x=x0+xdot0*T;
y=y0+ydot0*T;
theta=theta0+theta_dot0*T;

yd1=t*t;
yd2=xd*cos f** +yd*sin f*t;
yd3=-xd*sin(f** +yd*cos(f*t;

y1=theta;
y2=x*cos(theta +y*sin theta;
y3=-x*sin(theta +y*cos theta ;

e1=y1-yd1;
e2=y2-yd2;
e3=y3-yd3;

ud1=v2;
ud2=v1-y2*v2;

s1=e1;
```

```
s2=e2;
s3=e3+k1*e2*udl;

hita_dot=-k0*hita0-k1*s1-y3*s2;
hita=hita0+hita_dot*T;

v=udl+hita;
w=-(ud2-s2*udl-k3*s3-k2*e2*udl-k2*e3*udl);

theta=theta0+w*T;

xdot=-v*sin(theta);
ydot=v*cos(theta);

vl=int(v*3.6);
va=int(w*180/PI);

vm(1,1)=vl;
va(1,1)=va;

ws(1,1,1,2)=0;

t=t+T;

dc
(
xd=r*cos(f*t);
yd=r*sin(f*t);

x=x+xdot*T;
y=y+ydot*T;

deltax=x-xd;
deltay=y-yd;
deltath=theta-t;

yd1=f*t;
```

```
yd2=xd*cos(f*t)+yd*sin(f*t);
yd3=-xd*sin(f*t)+yd*cos(f*t);

y1=theta;
y2=x*cos(theta)+y*sin(theta);
y3=-x*sin(theta)+y*cos(theta);

e1=y1-yd1;
e2=y2-yd2;
e3=y3-yd3;

ud1=v2;
ud2=v1-y2*v2;

s1=e1;
s2=e2;
s3=e3+k2*e2*ud1;

hita_dot=-k3*(k1*s1+y3*s3);
hita=hita+hita_dot*T;

v=ud1+hita;
w=-(ud2-s2*ud1-k3*s3-k2*e2*ud1-k2*e3*ud1);

theta=theta+w*T;

xdot=-v*sin(theta);
ydot=v*cos(theta);

v1=int(v*8);
va=int(w*1.0/PI);

vm(v1,va,0);
ws(1,1,0,20);
printf("The x is: %6.2f\n",x);
printf("The xd is:  6.2f\n",xd);
```

```
printf("The deltax is: %6.2f\n",deltax);
printf("The y is: %6.2f\n",y);
printf("The yd is: %6.2f\n",yd);
printf("The deltay is: %6.2f\n",deltay);
printf("The theta is: %6.2f\n",theta);
printf("The theta is: %6.2f\n",theta);
printf("The t is:%6.2f\n",t);
printf("The deltath is: %6.2f\n",deltath);
printf("The r is %4d\n",r);
printf("The v1 is:%4d\n",v1);
printf("The v2 is: %4d\n",v2);
printf("The e1 is: %6.2f\n",e1);
printf("The e2 is: %6.2f\n",e2);
printf("The e3 is:%6.2f\n",e3);
printf("The s1 is:%6.2f\n",s1);
printf("The s2 is:%6.2f\n",s2);
printf("The s3 is: %6.2f\n",s3);
printf("The y1 is: %6.2f\n",y1);
printf("The y2 is: %6.2f\n",y2);
printf("The y3 is: %6.2f\n",y3);
printf("The yw1 is: %6.2f\n",yw1);
printf("The yw2 is: %6.2f\n",yw2);
printf("The yw3 is: %6.2f\n",yw3);
printf("The v is: %6.2f\n",v);
printf("The w is: %6.2f\n\n",w);

t=t+T;

}while(t<=50.005);
printf(" End of demo, quitting the server...\n");
sti();
disconnect_robot(ROBOT_ID);    /*     Kill     the     server;     just
disconnect_robot if the server is to be used again */
return(1);
}
```

Case1 (r=100cm, x(0)=150cm, y(0)=0, $\theta(0) = 28.6^{\circ}$)

X(cm)	y1(cm)	y2(cm)	y3(cm)	y4(cm)	y5(cm)	y6(cm)
154			3.5	3.5		
153			2	5		
152			1	7.5		
151			0.5	9.5		
150			0	11		
145				15		
140				20		
135				24.5		
130				30		
125				34.5		
120				37.5		
115				43		
110				47.5		
105				55		
100	0	0		60	0	0
95	31.22	-31.22		63.5	-23	32
90	43.58	-43.58		67	-38	45
85	52.68	-52.68		76	-43	53
80	60	-60		78.5	-56	61
75	66.14	-66.14		82	-59	68
70	71	-71		86	-65.5	73
60	80	-80		89	-72	81
50	86.6	-86.6		95	-82	86
40	91.68	-91		98	-85	91.5
30	95.39	-95.39		100	-90	95.5
20	97.97	-97.97		101.5	-93	98
10	99.5	-99.5		102	-97	99.5
0	100	-100		102.5	-99	100
-10	99.5	-99.5		101	-97	
-20	97.97	-97.97		99.5	-95.5	
-30	95.39	-95.39		97.5	-94	
-40	91.68	-91.68		92	-90	
-50	86.6	-86.6		88	-84	
-60	80	-80		81	-77.5	
-70	71	-71		72	-70	
-75	66.14	-66.14		66	-65	
-80	60	-60		61	-60.5	
-85	52.68	-52.68		55	-52	
-90	43.56	-43.58		45	-47	
-95	31.22	-31.22		29	-27	
-100	0	0		0	0	

Case 2 (r=150cm, x(0)=200cm, y(0)=0, $\theta(0) = 28.6^\circ$)

X(cm)	y1(cm)	y2(cm)	y3(cm)	y4(cm)	y5(cm)	y6(cm)
208			13.5	13.5		
207			12	15		
206			7	18		
205			5.5	20		
204			4	27		
203			3	30		
202			2	33		
201			1	35		
200			0	36		
195				42		
190				50		
185				55.5		
180				63		
175				67.5		
170				73		
165				77		
160				81.5		
155				85.5		
151				89	0	0
150	0	0		90.5	-11	9
145	38.39	-38.39		95.5	-36	39
140	53.85	-53.85		102	-51	55
135	65.38	-65.38		103.5	-57.5	66.5
130	74.77	-74.77		108	-76.5	75
125	87.22	-87.22		110.5	-77.5	85
120	90	-90		115	-82.5	93
115	96.3	-96.3		118.5	-89.5	98
110	101	-101		122	-93.5	103
105	107.11	-107.11		125	-99	108
100	111.8	-118.8		128	-105	115
95	116.08	-116.08		131	-109	118.5
90	120	-120		133	-114	122
85	123.59	-123.59		135	-117.5	125
80	126.89	-126.89		137	-121	127.5
75	129.9	-129.9		139	-123	130.5
70	132.66	-132.66		141	-126	134

60	137.47	-137.47		143.5	-132.5	139
50	141.42	-141.42		145	-135.5	143
40	144.56	-144.56		146.5	-140	145.5
30	146.97	-146.97		147.5	-143	148.5
20	148.66	-148.66		148.5	-144.5	149.5
10	149.66	-149.66		149.5	-146.5	150
0	150	-150		149.5	-148.5	150
-10	149.66	-149.66		149	-146	
-20	148.66	-148.66		147.5	-145	
-30	146.97	-146.97		145	-143.5	
-40	144.56	-144.56		143	-142	
-50	141.42	-141.42		137.5	-137.5	
-60	137.47	-137.47		135	-134.5	
-70	132.66	-132.66		130.5	-130	
-75	129.9	-129.9		125	-126	
-80	126.89	-126.89		120.5	-122.5	
-85	123.59	-123.59		118.5	-120	
-90	120	-120		114	-117	
-95	116.08	-116.08		110.5	-113	
-100	111.8	-111.8		107.5	-108	
-105	107.11	-107.11		104	-101.5	
-110	101	-101		97.5	-95.5	
-115	96.3	-96.3		91.5	-89	
-120	90	-90		84.5	-83.5	
-125	87.22	-87.22		78	-74	
-130	74.77	-74.77		71	-64.5	
-135	65.38	-65.38		62	-53	
-140	53.85	-53.85		52	-42	
-145	38.39	-38.39		37	-31	
-150	0	0		7	-8	
-151				0	0	