Graphical Simulation of a Mobile Robot

JUAN A. GOMEZ-PULIDO, JOSE M. GIRON-SIERRA, and BONIFACIO ANDRES-TORO
Departamento de Informatica y Automatica, Facultad de Fisicas, Universidad Complutense de Madrid, Paraninfo, 28040 Madrid, Spain

ABSTRACT

Some facets of our robotics research require the simulation of a mobile vehicle to study navigation and control problems. We have developed a 3-D animated graphical simulation of the movement of a wheeled vehicle under interactive control of the computer user. The article describes the mathematical formulation of the dynamics and the programming procedure to display the vehicle movement. Besides the possible use of our program for robotics education, the aspects covered by this article could be useful for motivating mechanics and graphics programming. © 1994 John Wiley & Sons, Inc.

INTRODUCTION

In recent years the interest in mobile robotics has increased, not only in response to the special challenges of planetary terrain exploration, but also for daily industrial or services applications, and for the substitution of humans in dangerous activities. Textbooks on robotics [1] include chapters on this new topic, and we believe that it is interesting to have an animated graphical simulation of the movement of a wheeled vehicle for illustration purposes.

Several years ago, a branch of our department dedicated to robotics started the implementation of an autonomous robot vehicle, with several scientific objectives. From the beginning we realized the need for computer simulations to study motion control strategies. Hence, we developed a graphical simulation showing on screen the dynamical behavior of the vehicle while receiving orders from the controlling entity.

Our simulation displays both 2-D and 3-D animated views of the vehicle with a trace of its trajectory. By means of interactive menus, the user of the simulation has control over the direction and speed of the vehicle, so that it is possible to play with trajectories (perhaps to study maneuvers or obstacle-avoidance tricks).

It is believed that the presented simulation could be useful for teaching and research (kinematics and dynamics, navigation algorithms, integration of several levels of control mechanisms, etc.), and is also a programming example for 3-D animated graphics developments.

MOBILE ROBOTS

A variety of mobile robot conceptions are now under study for planetary exploration, legged and/or wheeled. They must navigate within an environment, perhaps teleoperated, use external references, or be completely autonomous. There are difficult issues of path planning and obstacle avoidance.

For industrial needs, the use of robot carts is rapidly increasing in automated flexible factories. Frequently, these carts rely, by magnetic or optical means, on tracks present on the floor to guide them. There are some other applications demanding more flexibility or mobility, so the robot must be tel-
operated or autonomous, at the price of more involved control.

Both industry and research show interest for various vehicle geometries with a number of wheels: three, four, six, and so forth (two wheels only means a problem of static and dynamic stability). Most wheeled land vehicles fall into one of two categories: steer-wheeled or differential drive. The typical car and the tricycle belong to the first category. On the other hand, a military tank is an example of the second category: the steering is affected by differences in the velocity of the two tracked wheels, with the advantage that they have a zero turning radius.

We are interested in autonomous wheeled vehicles capable of intelligent motion without requiring a guide to follow. We shall combine autonomous decisions with teleoperation by a powerful computer running a simulation. As we want to enjoy a versatile maneuvering performance, we decided to build a six-wheeled, differential-drive vehicle. From the six, only two are driving wheels (using two DC servo motors); the other four are simple casters at the four corners. Therefore, the independent velocities of the two wheels present the only two degrees of freedom for control. Such a system is referred to as nonholonomic.

One of the most complete books about robot vehicles is by Schmidt [2]. It includes articles about the main aspects from the scientific and research points of view: guidance and control, sensors and sensing, navigation, maps, motion planning, and some examples of real systems. With a technical accent, a book by Cox and Wilfong [3] takes the information processing look at the autonomous vehicle control: perception, navigation, knowledge bases, learning, and so forth. Interest in this kind of system is open to many alternatives, including the use of toys and small robots in the pedagogic or research laboratory.

Figure 1 The mobile vehicle with its reference system.

![Figure 1](image-url)

KINEMATICS OF THE DEVELOPED ROBOT VEHICLE

Every simulation is based on a model: the more suitable the model, the better the simulation. We started by modeling the kinematics of the proposed vehicle.

Figure 1 shows the proposed vehicle: a rectangle with three wheels on each side. There are two opposed independent drive wheels on the left and right sides, with their common geometric axis passing through the center of the vehicle. The front and rear wheels are independent casters. The radius of the drive wheels is $D_r = 12$ cm; the distance between these wheels is $D_a = 53$ cm. The length of the cart is $D_l = 140$ cm, the width is $D_w = 70$ cm, and the height $D_h = 25$ cm. $C$ denotes the center of the vehicle, which is the robot position reference point. As shown in Figure 1, there is a vehicle reference axis system ($X_v, Y_v$) that moves with the vehicle.

To discretize the calculations, we used sampling with a short period, $dt$, between two consecutive instants. We assumed that the velocity of the driving wheels could only be changed in these instants and remain constant during these periods. We measured the speed in rpm: positive when going forward, negative when backward. The vehicle movement was assumed to be always on a plane.

Turning

When there is a difference between the velocity of the driving wheels, the vehicle turns. The turning direction is determined by the magnitude and sign of the speed of each driving wheel.

Suppose, for example, that the two driving wheels have $v_l$ (left wheel) and $v_r$ (right wheel) speeds at the instant $t_0$, remaining constant along $dt$. The displacement caused by each wheel is:

$$
\begin{align*}
    s_l &= v_l D_r dt \\
    s_r &= v_r D_r dt.
\end{align*}
$$

These displacements make the vehicle turn around the point $G$ (Fig. 2). The angle turned is $\delta$ and $R$ is the distance from $G$ to the next wheel. Hence,

$$
\delta = s_l / R = s_r / R + D_e,
$$

and we then have:
Figure 2  Turning because the driving wheels have different speeds.

\[
\begin{align*}
R &= \frac{D_r s_i}{s_r - s_i} \\
\delta &= \frac{s_i}{R} = \frac{s_r - s_i}{D_e}.
\end{align*}
\]

(3)

After the interval \( dt \), the vehicle will reach a new position, established by the distances:

\[
\begin{align*}
a_c &= (R + D_c/2)\tan \delta \\
b_c &= \sqrt{d_c^2 - a_c^2} \\
d_c &= 2(R + D_c/2)\tan(\delta/2), \quad \delta < \Pi.
\end{align*}
\]

(4)

**Location of the Vehicle**

Consider \( \vec{c} \) to be the position vector of \( C \) on a (world)fixed coordinate system \((Xw, Yw)\) (Figure 3). We know the initial location \( \vec{c}(t_0) \), and it is desired to find the end location \( \vec{c}(t_0 + dt) \):

\[
\vec{c}(t_0 + dt) = \vec{c}(t_0) + \vec{d}_c
\]

(5)

where \( \vec{c}(t_0) \), \( \delta \), and \( \vec{d}_c = |\vec{d}_c| \) are already known.

As may be seen from Figure 3, we denote the angle at \( t_0 \) between the two coordinate systems, vehicle and world, by \( \beta \). Also, \( \beta_1 \) is the angle at \( (t_0 + dt) \); hence,

Figure 3  Vehicle location in the world.

**Figure 4** Calculation strategy to follow the vehicle dynamics.

**Figure 5** Views for 3-D graphical display.
\[ \beta_i = \beta + \delta. \]  

\[ c_i(t_0 + dt) = c_i(t_0) + d_i \cos(\alpha + \beta) \]  

\[ c_i(t_0 + dt) = c_i(t_0) + d_i \sin(\alpha + \beta), \]  

The problem is solved by a simple change of coordinates of \( d_i \):

**Figure 6** Initial screen of the simulation program.

**Figure 7** Screen to specify main observer parameters.
where $\alpha$ is the angle between the axis $X_V$ at $t_0$, and $\tilde{d}_c$:

$$\alpha = \frac{11}{2} + \frac{\delta}{2}. \quad (8)$$

This provides us with all the information needed to determine the vehicle location as a function of time. The calculation strategy is represented in Figure 4. From some initial parameters (including velocities of the driving wheels $v_l$ and $v_r$, position of the vehicle center $C$ and $b$, angle between the two coordinate systems $\beta$), we iterate along time by computing in each period $dt$ the displacements and angles. These parameters are then used to calculate and follow the evolution of the vehicle motion. In the special case of having the two driving wheels moving with the same speed, the turning calculations are not made.

### 3-D ANIMATED GRAPHICS

There are several alternatives to showing the results of a simulation: numbers, scientific curves, graphs, and so forth. Trying to offer a motivating simulated view of the cart motion, we developed an animated 3-D graphical display of the vehicle kinematics. In the following, we shall describe the programming basis of the developed graphic package.

As illustrated in Figure 5, we used a camera model [4] in which the view on the screen was similar to taking a picture of the vehicle by a camera. The location of the camera is arbitrary in the world coordinates $(x_w, y_w, z_w)$. It is assumed that in looking directly toward a location with a viewing angle, how much of the scene is captured on the viewing plane is determined by the distance. Consider the following three coordinate systems:

- World (fixed) coordinate system: $(x_w, y_w, z_w)$. Center: $O$.
- Observer coordinate system: $(x_o, y_o, z_o)$. Center: $E$.
- Screen coordinate system: $(x_s, y_s, z_s)$. Center: $Q$.

The coordinates of the observer eye $E$ in the world reference system, are:

$$\begin{align*}
\text{polar: } & \rho, \varphi, \theta \\
\text{cartesian: } & X_E = \rho \sin \varphi \cos \theta \\
& Y_E = \rho \sin \varphi \sin \theta \\
& Z_E = \rho \cos \varphi.
\end{align*} \quad (9)$$

Given a point with coordinates $(x_w, y_w, z_w)$ in the world reference system, this point will have the following coordinates in the observer reference system:
Figure 9  Screen to define vehicle dimensions.

\[(x_e, y_e, z_e) = (x_w, y_w, z_w, 1).\]

Moreover, the coordinates of this point in the screen coordinate system are:

\[
\begin{bmatrix}
    -\sin \theta - \cos \varphi \cos \theta & -\sin \varphi \cos \theta & 0 \\
    \cos \theta - \cos \varphi \sin \theta & -\sin \varphi \sin \theta & 0 \\
    0 & \sin \theta & -\cos \theta \\
    0 & 0 & \rho \\
\end{bmatrix}.
\]  

(10)

\[
\begin{align*}
    x_s &= d \cdot x_e / z_e + x_{center} \\
    y_s &= d \cdot y_e / z_e + y_{center}.
\end{align*}
\]  

(11)

Figure 10  Screen showing the results of the simulation.
where $x_{center}$ and $y_{center}$ are the coordinates of the point $Q$ relative to the center of the screen. The distance $d$ may be determined depending on the value of $\rho$ so that good viewing results may be achieved:

$$d = \rho \cdot \text{image\_size/object size.} \quad (12)$$

**SIMULATION PROGRAM**

The program was developed using Turbo C++ v. 2.0 (Borland Co.) for MS-DOS computers. Having tested the code on different platforms, we noticed the animated graphics requires enough computing speed, color, and good graphic definition. A 486 computer with VGA is considered suitable for adequate execution.

With the help of C Windows Toolchest [5] (a library of routines for windows and mouse interface from Mix Co.), we created a friendly interactive interface for handling the simulations. The complete executable file of the developed program occupies 190 Kbyte.

As may be seen in Figure 6, the initial screen presents the main horizontal menu with four entries. With “Help” one may access the basics of the simulation and the users guide. “Exit” terminates the program and returns the user to MS-DOS. The other two entries will be discussed in detail.

"Settings"

The program provides initial default values. The user has the opportunity of returning to these default values. The “Settings” entry deploys a submenu to allow for changes in the settings with the following choices:

"Main." In this menu (Fig. 7), one can establish the location of the observer ($\rho, \varphi, \theta$), the vehicle location and orientation ($a, b, \beta$), the distance from the observer to the screen ($d$), and the length of the axes (useful as a visual reference). We use meters and hexadecimal degrees. There are data entry fields and buttons to introduce the desired values, or one may simply accept the default values (button “Def”). The graphical results of the parameters introduced are displayed by means of the main simulation screen.

"Map." With this entry (Fig. 8) the user can specify the characteristics of a second observer: position ($\rho_m, \varphi_m, \theta_m$) and distance ($d_m$) to the screen. During the simulation the user can display two graphical windows: for instance one in 3-D and the other in 2-D from the ceiling to see the path followed by the vehicle. The graphical results of the parameters introduced are displayed as before.

"Dimensions." The user can define (Fig. 9) dimensions of the vehicle, adapting the simulation to a real vehicle or to an ideal one that the user wants to study. The screen shows the aspect of the new vehicle.

"Parameters." With this entry the user can change the timing (beginning, end, step) and the initial speed values of the wheels (rpm).

"Simulation"

This is the core of the developed program. The simulated movements of the vehicle (as we see in Fig. 10) are displayed in two windows: the “main” (larger) and “map” (smaller) windows. Three buttons permit the basic control of the simulation: start, pause, abandon. The user can change the position of the “main” observer and the “map” observer (in this case, only the distance to the origin of the world coordinates). Two entry fields are used to change the speed of each driving wheel.

To modify the position of the observers, it is necessary to pause, introduce the new values, and then push “Ok” to continue the simulation. Thus there is no need to get to the “Settings” submenu to change the point of view. The user can change, online, the speed of the wheels without stopping the simulation; hence, the user is driving the cart.

In addition, the user can toggle between two views on the main window by using two buttons (“SM” and “MS”): one of the views offers more detail and proximity to study precision movements.

**CONCLUSIONS**

A program for an animated 3-D graphical simulation of a wheeled vehicle is described. The code is fully accessible and could be interesting for study or as a template for further developments.

The kinematics modeling may be seen as an interesting classical mechanics problem. There is a second additional problem: the dynamics of a 110-kg vehicle driven by geared DC servo motors under smoothing control [6,7].
Present work deals with the inclusion of obstacles and maps in the simulation and the navigational issues of the control. The authors are open to, and will be pleased to receive comments and suggestions on the developed program.

REFERENCES


BIOGRAPHIES

Jose M. Giron-Sierra is Professor of Automatic Control and Systems Engineering in the Department of Computer Science and Automatic Control, University Complutense of Madrid. He received a PhD degree in Physics from the University Complutense in 1978. He is involved in the organization of the Spanish Society for Simulation. His research interests are simulation and experimental control of robotic devices and chemical pilot plants, using modern technologies.

Juan A. Gomez-Pulido is Assistant Professor of Automatic Control and Systems Engineering in the Department of Computer Science and Automatic Control, University Complutense of Madrid. He received a PhD degree in Physics from the University Complutense in 1993. His research interests are simulation and experimental control of robotic devices and power plants, using modern software technologies.

Bonifacio Andres-Toro is Adjunct Lecturer of Automatic Control and Systems Engineering in the Department of Computer Science and Automatic Control, University Complutense of Madrid. He received a BS degree in Physics from the University Complutense in 1989. His research interests are simulation and experimental control of chemical pilot plants, using modern technologies.