2008 American Control Conference
Westin Seattle Hotel, Seattle, Washington, USA
June 11-13, 2008

Exponential navigation functions with a learning algorithm

K. Bendjilali ‡, F. Belkhouche† and T. Jin †
‡ECE Department, Lehigh University, PA, USA
kb5@Lehigh.edu
†Texas A & M Intl University, Laredo, TX
(fbelkhouche, tjin)@tamiu.edu

Abstract— **This paper suggests a method for autonomous wheeled mobile robots navigation under the nonholonomic constraint. The suggested method uses navigation functions that are based on the polar kinematics equations, where the steering angle and the orientation angle of the robot are included in an exponential function of the line of sight angle. Another control law is suggested for the robot's linear velocity to drive the robot to a desired position with a desired final orientation angle. The exponential navigation functions depend on various navigation parameters that allow to change the robot's path. This approach is combined with the collision cone technique to avoid collision. A Q-learning algorithm is suggested to select automatically the appropriate values of the navigation parameters. Simulation is used to illustrate the method.**

I. Introduction

Autonomous robot navigation and path planning are among the most important topics in robotics. The topic of navigation is widely discussed, since navigation is an elementary task that is necessary to accomplish various other tasks. Based on different criteria, navigation methods can be divided into various families, such as (1) model-based methods; (2) behavioral–based methods; and (3) sensor-based methods. Potential field methods ([1], [2]) are among the most discussed methods. In these methods, the robot navigates in a field that is the sum of an attractive force resulting from the goal and repulsive forces resulting from the obstacles. These methods suffer from the local minima problems, where the robot can be trapped without reaching the goal. In many situations, the local minima problems are inevitable when the robot moves in an unknown environment, and cannot predict the position of the obstacles. Various methods have been suggested to improve the classical potential field method, and therefore solve the local minima problems. In [3], the relationship between potential field–based navigation and dynamic constraints optimization is discussed. Numerical potential fields are discussed in [4]. Here, the potential fields are constructed numerically, rather than analytically; this allows to reduce the computation time. A vector–based approach to the potential field method is also suggested in [2]. In [5], an electrostatic potential field is suggested. It is proven through the classical laws of electrostatics that the derived potential field is a global navigation function, and that the field is free of all local minima and that all paths necessarily lead to the goal position.

The behavioral control paradigm was introduced in [6]. In behavioral control methods, the control problem is decomposed based on the task instead of the function. Unlike the classical approach, behavioral modules are connected in parallel. This provides better robustness to the system. Various works have been suggested in this direction. One example is given in reference [7], where a behavioral approach integrated architecture is suggested for mobile robots navigation.

Sensor-based methods represent another important family of navigation methods. Different types of sensors such as sonar-sensors ([8], [9]), infra–red sensors, and vision sensors are used. Vision sensors are of a particular interest, since they provide a more comprehensive overview to the scene. In vision, the perception of the environment is accomplished through image processing. A nice survey can be found in [10]. Note that artificial vision is also used in intelligent transportation ([11], [12], [13]). In [14], monocular vision is used to develop a navigation algorithm. In [15], global vision is combined with Voronoi roadmaps. The global vision system is mainly used for obstacles detection. In [10], an integrated inertial navigation system is suggested for unmanned air vehicles. The algorithm is illustrated through simulation. Vision is also used in [16] for small robot agents. Other sensors are also used. A sensor-based method is suggested in [17] to replace the traditional planned architectures. Another sensor–based algorithm that uses generalized Voronoi graph is discussed in [18]. Qualitative vision is used in [19], where a teach–replay approach is used. An evolutionary approach to visual sensing for vehicle navigation is discussed in [20]. The main drawback of vision-based navigation methods is the prolonged processing time of the huge amount of data coming from the camera.

In the last decade, machine learning has become an important research topic. An increasing number of algorithms use different learning techniques to improve autonomous behavior for mobile robots. Navigation, localization, and mapping are widely considered ([21], [22], [23], [24]). The goal of machine learning is to increase the autonomy of the robot.

In this paper, we introduce a new method for autonomous robot navigation. Our method is based on the kinematics equations, where the robot navigates according to exponential navigation functions. These navigation functions depend on navigation parameters. They are different from classical navigation functions, which are mainly based on the potential field idea. Kinematics–based navigation functions that are

based on a linear formulation are discussed in previous work ([25], [26], [27]). Linear navigation functions present interesting properties. However, they have restrictions on the values of the navigation parameters. Exponential navigation functions allow to solve this problem. Q-learning is used to plan the path of the robot by choosing the appropriate values for the navigation parameters. This paper is organized as follows: In section II, the model of the robot and other definitions are introduced. In section III the control law is introduced and discussed. In section IV, the learning algorithm is suggested. Simulation is carried out in section V.

II. Modeling and position of the problem

The workspace is attached to a reference frame of coordinates, and is clustered with obstacles B_i, $i = 1...N$, where N is the total number of obstacles. We refer to the simplified model of the car-like robot depicted in figure 1. The distance between the reference point P and the middle point of the driving wheels is h. In this paper, the robot moves according to the kinematics model given by

$$\begin{matrix} \dot{x}_p \\ \dot{y}_p \\ h\dot{\theta} \\ \dot{\phi} \end{matrix} = \begin{bmatrix} \cos\phi\cos\theta \\ \cos\phi\sin\theta \\ \sin\phi \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega \qquad (1)$$

where (x_p, y_p) denote the coordinates of point P. The configuration of the car-like robot is given by four variables: $X = (x, y, \theta, \phi)$. The model given by (1) presents an nonholonomic constraint. The orientation angle of the car-like robot is denoted by θ. The steering angle is ϕ. The configuration space of this system is four dimensions, i.e., $C = R^2 \times S^2$, where S represents the unit circle. v is the linear velocity of the driving wheels. ω is the time derivative of the steering angle, it represents the angular velocity. The kinematics model given by (1) is expressed in terms of the coordinates of the reference point P. However, in our model, we use the kinematics equations written in terms of point Q. This is accomplished by a transformation of coordinates. From figure 1, the relationship between the coordinates of the reference points P and Q is given as follows:

$$\begin{aligned} x_q &= x_p + h\cos\theta \\ y_q &= y_p + h\sin\theta \end{aligned} \qquad (2)$$

The time derivative of (2) allows us to get the following velocities

$$\begin{aligned} \dot{x}_q &= v\cos(\theta+\phi) \\ \dot{y}_q &= v\sin(\theta+\phi) \end{aligned} \qquad (3)$$

Point $G(x_g, y_g)$ is the goal of the robot. We define the line of sight angle α as follows

$$\tan\alpha = \frac{y_g - y_q}{x_g - x_q} \qquad (4)$$

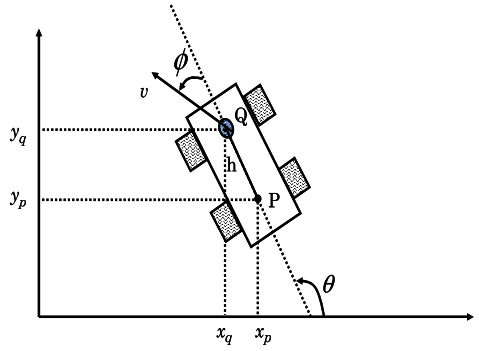

Fig. 1. An illustration of the geometry of the navigation problem

when $x_g \neq x_q$. The relative range between point Q and the goal is given by

$$l = \sqrt{(y_g - y_q)^2 + (x_g - x_q)^2} \qquad (5)$$

Consider the relative velocity between the robot (point Q) and the goal (G) given by

$$\begin{aligned} \dot{x}_{rel} &= \dot{x}_g - \dot{x}_q \\ \dot{y}_{rel} &= \dot{y}_g - \dot{y}_q \end{aligned} \qquad (6)$$

The relative velocity can be expressed as follows

$$\begin{aligned} \dot{x}_{rel} &= -v\cos(\theta+\phi) \\ \dot{y}_{rel} &= -v\sin(\theta+\phi) \end{aligned} \qquad (7)$$

Based on this model, it is easy to determine whether the robot is approaching, or moving away from the goal. It is also possible to obtain the relative velocity in polar form. This task is accomplished by considering the following change of variables $x = l\cos\alpha, y = l\sin\alpha$. Under these conditions, we obtain

$$\begin{aligned} \dot{l} &= -v\cos(\theta+\phi-\alpha) \\ l\dot{\alpha} &= -v\sin(\theta+\phi-\alpha) \end{aligned} \qquad (8)$$

This is a simple model that is equivalent to (7). However, this model gives more information concerning the navigation process, since the first equation allows to determine the range robot—goal. A negative range rate implies that the robot is approaching from the goal. The second equation allows to determine the rate of turn of the robot with respect to the goal.

III. Control law

Our control law is based on exponential navigation functions with deviation terms. As we mentioned previously, our navigation laws are based on the kinematics equations, and thus, they are different from classical navigation laws, which are based mainly on the potential field method. Under our control law, the relationship between the orientation angle, the steering angle, and the line of sight angle is given by

$$\theta + \phi = \alpha + A\exp\left(\frac{\alpha}{A}\right) + c_f + c_i\exp(-bt) \qquad (9)$$

where A is a real number called the navigation parameter, c_i, c_f and b are also navigation parameters satisfying certain conditions. The term $c_i \exp(-bt)$ is a heading regulation term that goes to zero with time. It will be proven that under this control law, the robot will reach the goal successfully. By taking the derivative in (9) we obtain the following equation

$$\dot{\theta} + \dot{\phi} = \dot{\alpha}\left(1 + \exp\left(\frac{\alpha}{A}\right)\right) - bc_i \exp(-bt) \quad (10)$$

which is equivalent to

$$\dot{\theta} + \omega = \dot{\alpha}\left(1 + \exp\left(\frac{\alpha}{A}\right)\right) - bc_i \exp(-bt) \quad (11)$$

By replacing θ by its value, we obtain

$$v\frac{\sin\phi}{h} + \dot{\phi} = \dot{\alpha}\left(1 + \exp\left(\frac{\alpha}{A}\right)\right) - bc_i \exp(-bt) \quad (12)$$

Clearly, the implementation of the control law requires the knowledge of the line of sight angle and its rate. The expression of α under our control law is given by

$$l\dot{\alpha} = -v\sin\left(A\exp\left(\frac{\alpha}{A}\right) + c_f + c_i\exp(-bt)\right) \quad (13)$$

A second law is derived for the robot's linear velocity. In this case, we put

$$v = kl \quad (14)$$

where k is a positive real number. The robot slows down near the goal and stops when it reaches the goal. The following result concerns the navigation under our control law.

Proposition 1: Under the control laws given by 9 and 14, the robot reaches its goal from any initial position.

Proof: After the heading regulation phase, the range rate equation under our control law is given by

$$\dot{l} = -lk\cos\left(A\exp\left(\frac{\alpha}{A}\right) + c_f\right) \quad (15)$$

The equation for the line of sight rate is given by

$$\dot{\alpha} = -k\sin\left(A\exp\left(\frac{\alpha}{A}\right) + c_f\right) \quad (16)$$

The equilibrium position for the dynamical system given by equations $(15, 16)$ is given by $(l_{eq}, \alpha_{eq}) = \left(0, A\ln\left(-\frac{c_f}{A}\right)\right)$. Clearly, A and c_f have opposite signs and $c_f \neq 0$. By taking the Jacobian of the system, we obtain

$$J = \begin{bmatrix} -k\cos(Af + c_f) & klf\sin(Af + c_f) \\ 0 & -kf\cos(Af + c_f) \end{bmatrix} \quad (17)$$

with

$$f = \exp\left(\frac{\alpha}{A}\right) \quad (18)$$

The Jacobian matrix at the equilibrium position is given by

$$J = \begin{bmatrix} -k & 0 \\ 0 & -k \end{bmatrix} \quad (19)$$

Both the eigenvalues are real and negative, which implies that the equilibrium positions are asymptotically stable, thus $\alpha \to \alpha_{eq}$ and $l \to 0$.

■

The navigation law given by (9), depends on various parameters. The main advantage of this method is the possibility to obtain different paths for different navigation parameters, and also change the path on-line by changing the values of the navigation parameters. For collision avoidance, a simple collision cone is used, where the aim is to keep $\theta + \phi$ outside the collision cone. In fact when an obstacle is within a given distance from the robot, the robot deviates towards an intermediary goal by changing the value of the navigation parameters.

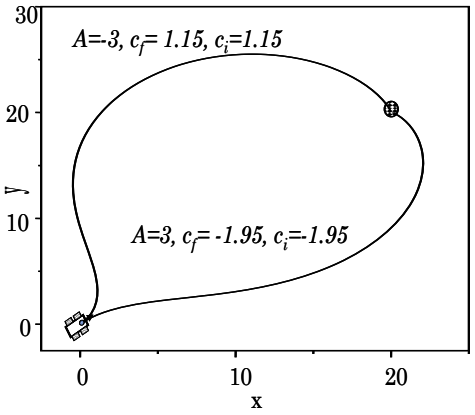

Fig. 2. An illustration of the right and left detours

The path of the robot in the (l, α) plane (polar coordinates) is given by

$$\frac{dl}{l} = \frac{d\alpha}{\tan\left(A\exp\left(\frac{\alpha}{A}\right) + c_f\right)} \quad (20)$$

which gives

$$\ln l - \ln l_0 = \frac{d\alpha}{\tan\left(A\exp\left(\frac{\alpha}{A}\right) + c_f\right)} \quad (21)$$

where l_0 is the initial range. Clearly, the robot's path does not depend on the linear velocity.

A. Left and right deviations

Exponential navigation functions offer a very important flexibility for robot navigation. One possibility is the left and right detour around an obstacle. By simply changing the value of A and the other navigation parameters, the robot can deviate to the left or the right. This is illustrated in figure 2, where for $A = -5, c_f = 2.75, c_i = 1.52$, the robot turns right, and for $A = 5, c_f = -4, c_i = 1.85$, the robot turns left.

B. Turning under the nonholonomic constraint

The heading regulation phase allows to take the nonholonomic constraint into account. The radius of curvature can be fixed by using the parameter c_i and b. Long turns correspond to smaller values of b, and short turns correspond to larger values of b. Figure 3 shows the robot turning at different turning radii.

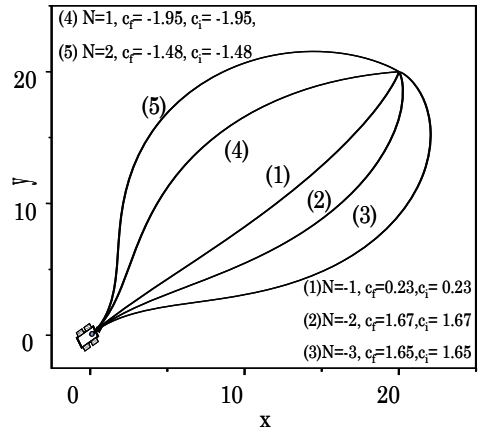

Fig. 3. Different paths for different navigation parameters

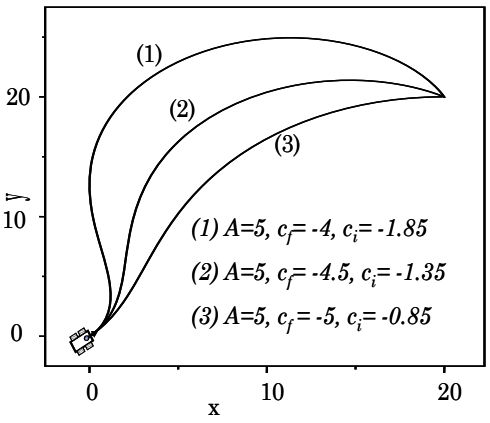

Fig. 4. Robot reaching its final destination using different values of c_i and c_f; $A = 5$

C. Final value for the robots steering and orientation angles

As we have seen previously, the final value for the line of sight angle is given by

$$\alpha_f = A \ln\left(-\frac{c_f}{A}\right) \tag{22}$$

Thus the final value of the line of sight angle depends on A and c_f. A and c_f always have opposite signs. From equation (9), we can write

$$\beta_f = \theta_f + \phi_f = A \ln\left(-\frac{c_f}{A}\right) \tag{23}$$

Thus, the final value for β_f can be fixed by the choice of A and c_f. In general A is chosen to avoid collision; and therefore, c_f is chosen to fix the final value of β_f. In this case the formula for c_f is given by

$$c_f = -A \exp\left(\frac{\beta_f}{A}\right) \tag{24}$$

An example is shown in figure 4, where the final value of β_f is $\pi/2$.

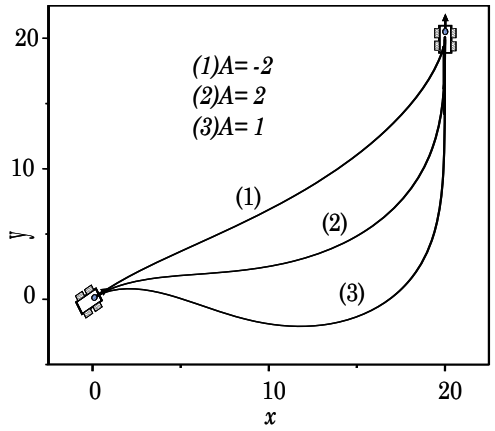

Fig. 5. Robot reaching its final position (20,20,$\beta_f = \pi/2$) from different initial positions using different navigation parameters

IV. Q-LEARNING OF NAVIGATION FUNCTIONS

Initially, the robot navigates using the control law given by equation (9). The Q–learning algorithm is used for learning navigation functions by choosing the appropriate values for the navigation parameters that allow to avoid obstacles. In Q-learning, the values of state-action pairs are defined and estimated on-line. Let $Q(s,a)$ be the expected discounted reinforcement of taking action a in the state s. The general algorithm works as follows

1) $Q(s,a)$ is initialized arbitrarily
2) repeat:
3) Initialize s
4) Repeat for each step:
5) Choose a from s using policy derived from Q
6) Take action a, observe r and s'
7) $Q(s,a) \leftarrow Q(s,a) + \mu\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right]$
8) $s \leftarrow s'$
9) until s is terminal

The state consists of the robot's position with respect to the nearby obstacles. The algorithm learns a value function that maps states to action. The reward function is determined by the robot's orientation angle and the position of the obstacles. With reference to figures 6 and 7, we define four states based on which the reward function is derived:

A

$$\theta + \phi \in [\alpha_0 - \varepsilon_1, \alpha_0 + \varepsilon_1] \tag{25}$$

B

$$\begin{array}{c} \theta + \phi \in [\alpha_0 - \varepsilon_1 - \varepsilon_2, \alpha_0 - \varepsilon_1] \\ \theta + \phi \in [\alpha_0 + \varepsilon_1, \alpha_0 + \varepsilon_1 + \varepsilon_2] \end{array} \tag{26}$$

C

$$\begin{array}{c} \theta + \phi \in [\alpha_0 - \varepsilon_1 - \varepsilon_2 - \varepsilon_3, \alpha_0 - \varepsilon_1 - \varepsilon_2] \\ \theta + \phi \in [\alpha_0 + \varepsilon_1 + \varepsilon_2, \alpha_0 + \varepsilon_1 + \varepsilon_2 + \varepsilon_3] \end{array} \tag{27}$$

D

$$\begin{array}{c} \theta + \phi \in [0, \alpha_0 - \varepsilon_1 - \varepsilon_2 - \varepsilon_3] \\ \theta + \phi \in [\alpha_0 + \varepsilon_1 + \varepsilon_2 + \varepsilon_3, 2\pi] \end{array} \tag{28}$$

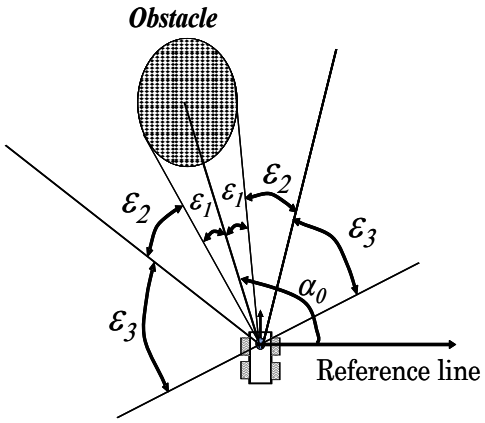

Fig. 6. Zone partition around the obstacle

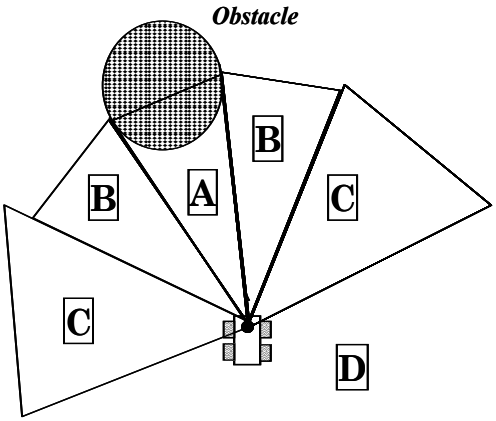

Fig. 7. Zones around an obstacle

In the presence of more than one obstacle near the robot, the same approach is used for each robot. A simple collision cone test allows to detect the state of the robot. Clearly, state A corresponds to collision. The initial reward matrix is given by

$$r = \begin{array}{c} state\backslash action \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} -1000 & 50 & NA & NA \\ -1000 & 25 & -100 & NA \\ NA & 20 & 15 & -100 \\ NA & NA & 10 & 5 \end{bmatrix} \end{array} \tag{29}$$

The negative numbers in the matrix indicate negative rewards, i.e., punishment. The robot learns through many episodes, and the Q-matrix converges to the following matrix

$$Q = \begin{array}{c} state\backslash action \\ A \\ B \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} -880 & 150 & NA & NA \\ -880 & 125 & -100 & NA \\ NA & 120 & 111 & -15 \\ NA & NA & 106 & 90 \end{bmatrix} \end{array} \tag{30}$$

Let A_A, A_B, A_C and A_D be the values of the navigation parameter corresponding to states A, B, C, and D, respectively. The values of the navigation parameters change as follows: $A_D \rightarrow A_C \rightarrow A_B$, and $A_A \rightarrow A_B$. When A reaches A_B, the robot continues using the same value until it passes the obstacle. The process is repeated whenever the robot is in a collision course with the obstacle.

V. Simulation

Simulation of the previous algorithms is suggested in this section. The working space has four obstacles as shown in figure 8 . The robot starts from the origin and aims to reach point $(50, 50)$. Figure 8 shows the different paths. Deviation from obstacles in path 1 and path 2 is accomplished by changing the value of A. In path 3, the value of A is kept constant, and deviation from the obstacles is accomplished by changing the value of c_f and c_i. The dots in figure 8 show the different segments of the path where the robot changes its navigation parameters. The values of the navigation parameters for each path and each segment are shown in tables I and II.

TABLE I
PATH 1 (4 SEGMENTS)

	seg. 1	seg. 2	seg. 3	seg. 4
A	-2	2	3	-1
c_f	1	-2	-2	1.0114
c_i	0.35	-1.35	-1.0119	1

VI. Conclusion

In this paper, we presented a method for robot navigation under the nonholonomic constraint; our method is based on exponential navigation functions that are derived based on the kinematics of the robot. The method depends on various parameters. These parameters can be changed in real time to adjust the path of the robot and avoid possible collisions. The method allows reaching a desired final position with a desired orientation angle from any initial position. The turning rate is taken into account. A Q–learning algorithm is used in order to learn the navigation parameters. The method is illustrated via a simulation example.

TABLE II
PATH 2 (4 SEGMENTS)

	seg. 1	seg. 2	seg. 3	seg. 4
A	-2	-3	3	-1
c_f	1	0.96	-3	0.36
c_i	0.35	1	-2.45	1

TABLE III
PATH 3 (5 SEGMENTS)

	seg. 1	seg. 2	seg. 3	seg. 4	seg. 5
A	-1	-1	-1	-1	-1
c_f	0	2	1	-0.25	1
c_i	0	-2	0.72	1.25	-1.25

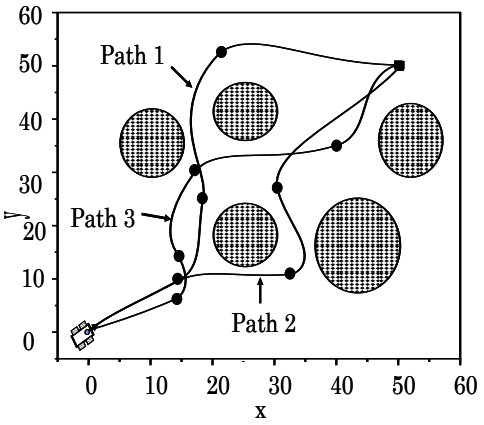

Fig. 8. Navigation in the presence of obstacles

REFERENCES

[1] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.

[2] S. Masoud and A. Masoud, "Constrained motion control using vector potential fields," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 30, no. 3, pp. 251–272, 2000.

[3] G. Dozier, A. Homaifar, S. Bryson, and M. Bikdash, "Artificial potential field-based motion planning/navigation, dynamic constrained optimization and simple genetic hill climbing," *Simulation*, vol. 71, no. 3, pp. 168–181, 1998.

[4] J. Barraquand, B.Langlois, and J. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[5] K. Valavanis, T. Hebert, R. Kolluru, and N. Tsourveloudis, "Mobile robot navigation in 2-d dynamic environments using an electrostatic potential field," *IEEE Transactions on Systems, Man and Cybernetics, Part A*, vol. 30, no. 2, pp. 187–196, 2000.

[6] R. Brooks, "New approaches to robotics," *Science*, vol. 253, pp. 1227–1232, 1991.

[7] J. Chung, B.-S. Ryu, and H. S. Yang, "Integrated control architecture based on behavior and plan for mobile robot navigation," *Robotica*, vol. 16, no. 4, pp. 387–399, 1998.

[8] S.-Y. Yi and B.-W. Choi, "Autonomous navigation of indoor mobile robots using a global ultrasonic system," *Robotica*, vol. 22, no. 4, pp. 369–374, 2004.

[9] A. M. Flynn, "Combining sonar and infrared sensors for mobile robot navigation source," *International Journal of Robotics Research*, vol. 7, no. 6, pp. 54–64, 1988.

[10] G. DeSouza and A. Kak, "Vision for mobile robot navigation: A survey," *IEEE Transactions on Pattern analysis and machine intelligence*, vol. 24, no. 2, pp. 237–267, 2002.

[11] A. Broggi, M. Bertozzi, A. Fascioli, C. G. L. Bianco, and A. Piazzi, "Visual perception of obstacles and vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 1, no. 3, pp. 164–177, 2000.

[12] U. Franke, D. Gavrila, S. Grzig, F. Lindner, F. Paetzold, and C. Whler, "Autonomous driving approaches downtown," *IEEE Intelligent Systems*, vol. 13, no. 6, pp. 1–14, 1999.

[13] M. Bertozzi, A. Broggi, M. Cellario, A. Fascioli, P. Lombardi, and M. Porta, "Artificial vision in road vehicles," *Proceedings of the IEEE*, vol. 90, no. 7, pp. 1258–1271, 2002.

[14] C. Sagues and J. Guerrero, "Motion and structure for vision-based navigation," *Robotica*, vol. 17, no. 4, pp. 355–364, 1999.

[15] W. L. Roque and D. Doering, "Trajectory planning for lab robots based on global vision and voronoi roadmaps source," *Robotica*, vol. 23, no. 4, pp. 467–477, 2005.

[16] ThomasBrunl and BirgitGraf, "Small robot agents with on-board vision and local intelligence," *Advanced Robotics*, vol. 14, no. 1, pp. 51–64, 2000.

[17] S. Garrido, L. Moreno, D. Blanco, and M. L. Munoz, "Sensor-based global planning for mobile robot navigation," *Robotica*, vol. in press, 2007.

[18] Y. I. Keiji Nagatani and Y. Tanaka, "Sensor-based navigation for car-like mobile robots based on a generalized voronoi graph," *Advanced Robotics*, vol. 17, no. 5, pp. 385–401, 2003.

[19] Z. Chen and S. Birchfield, "Qualitative vision-based mobile robot navigation," in *Proc. IEEE International Conference on Robotics and Automation*, Florida, May 2006, pp. 2686–2692.

[20] B. A, M. Cellario, P. Lombardi, and M. Porta, "An evolutionary approach to visual sensing for vehicle navigation," *IEEE Transactions on Industrial Electronics*, vol. 50, no. 1, pp. 18–29, 2003.

[21] S. Thrun, "Learning maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, pp. 21–71, 1998.

[22] ——, "Bayesian landmark learning for mobile robot localization," *Machine Learning*, vol. 33, no. 1, pp. 41–76, 1998.

[23] A. Billard and M. J. Mataric, "Learning human arm movements by imitation: Evaluation of a biologically inspired connectionist architecture," *Robotics and Autonomous Systems*, vol. 37, no. 2-3, pp. 145–160, 2001.

[24] M. Nicolescu and M. J. Mataric, "Learning and interacting in human-robot domains," *IEEE Transactions on Systems, Man, Cybernetics*, vol. 31, no. 5, pp. 419–430, 2001.

[25] F. Belkhouche, B. Belkhouche, and P. Rastgoufard, "Autonomous navigation and obstacle avoidance using navigation laws with time-varying deviation functions," *Advanced Robotics*, vol. 21, no. 5-6, pp. 555–581, 2007.

[26] F. Belkhouche and B. Belkhouche, "Wheeled mobile robot navigation using proportional navigation," *Advanced Robotics*, vol. 21, no. 3-4, pp. 395–420, 2007.

[27] F. Belkhouche, "Nonholonomic robots navigation using linear navigation functions,," in *Proc. American Control Conference*, New York, July 2007.