Design and Validation of a Robotic System to Interactively Teach Geometry

Lorenzo Riano and Martin McGinnity
Intelligent Systems Research Centre
University of Ulster
Londonderry, UK

***Abstract*—Learning geometry can be significantly improved if the student interacts with shapes and their transformations. We present the design and validation of a robotic system that teaches geometry by natural interaction. The robot is able to draw arbitrary shapes in the environment by means of its own movement. It is also able to detect and track the student movements, representing them as geometrical shapes and reproducing them.**

We will show four experiments where a robot is able to draw mathematical shapes, including an affine transformation, and two experiments where the robot reproduces trajectories a student previously showed it.

Fig. 1. The Scitos-G5 during a demo

I. Introduction and System Overview

In the late sixties Seymour Papert invented Turtle Graphics and LOGO [1], a programming language for geometric drawing as a tool for basic programming and geometry teaching. A real or simulated robot, shaped as a turtle, is programmed to move in a $2D$ space and to draw lines while moving. The main reason for its development and success is that it places the learner in an interactive environment where he/she can experiment with the geometry involved in drawing a figure, while receiving direct visual feedback from the turtle movements [2].

It is commonly assumed that the teaching of geometry should contribute to the learning of, among others, "the movement between theoretical objects and their spatial representation" [3]. Observing an abstract shape being drawn by a real robot should therefore significantly contribute to a student's understanding of geometry. Other concepts such as affine transformations can be more efficiently learnt by a student if they observe their effects on an arbitrary shape or observethe effects of varying parameters [4].

In this paper we present a robotic system that interactively teaches geometry according to the guidelines above. In particular, it describes shapes in the environment by means of its own motion. To the best of our knowledge this is the first application of a real robot to teach geometry by natural interaction.

Such interaction starts from the student, who defines a geometric shape. This can be accomplished in two ways: i) by using the mathematical description of a shape, or ii) by moving in the environment, thus describing a shape via his/her own movements. Once a shape has been chosen, the robot shows what its approximation of it looks like, using a simulated trajectory. The student can vary parameters such as the smoothness of the approximation and the speed at which the robot should travel. All of these parameters have an intuitive interpretation, and the student can immediately visualise the effect of changing them by observing how much the proposed robot trajectory matches the original shape.

When the student is satisfied with the proposed trajectory, the robot starts moving along it. This way it is virtually "drawing" a shape in the space by moving in the environment, providing a visual feedback to the student. Once the robot stops moving the student can manipulate the shape by using affine transformations, and observe again the robot moving along it. This way the student can "see" geometry, and he/she can interact with it by manipulating figures and observing the effect on the robot's movements.

The main target audience of the proposed system are primary school pupils, who are learning the basic concepts of geometry. In this case observing the shapes being drawn by the robot's movements can improve their understanding of the subject [3]. Our system can as well be used in an undergraduate robotics course, where the students are presented with problems of path planning and trajectory following, and they can observe the results of varying several parameters on the physical robot itself.

In order to carry on the above task the robot requires several components, namely i) people detection and tracking, ii) trajectory extraction and iii) path following. The rest of the paper describes in more detail the proposed system and the experimental results that validate the system.

l.riano@ulster.ac.uk, tm.mcginnity@ulster.ac.uk

This research is supported by the Centre of Excellence in Intelligent Systems (CoEIS) project, funded by Northern Ireland Integrated Development Fund and InvestNI.

This paper is dedicated to the memory of Prof. Ulrich Nehmzow, whose inspiration and ideas led to the development of this work.

TABLE I
CONFUSION MATRIX FOR (LEFT) THE VIOLA-JONES CLASSIFIER AND (RIGHT) WHEN COMBINED WITH THE RBFN, FOR A PERSON PRESENT OR NOT PRESENT.

	Present	Not Present
Detected	98%	24%
Not detected	2%	76%
	Present	Not Present
Detected	78%	0%
Dot detected	22%	100%

II. PEOPLE DETECTION AND TRACKING

An effective people detection and tracking system is the one that minimises both false positives and negatives, i.e. it does not mistake objects in the environment as human.

1) Face detection: In order to reliably detect people in the environment, we used the Viola and Jones approach to face detection [5]. This classifier is scale and light conditions invariant.

In order to measure its performance, we conducted two experiments, the first one with a person always facing the robot camera, the second one without any person in the environment. During the first experiment the robot was following the person, while during the second one it was randomly moving. Both experiment lasted around 15 minutes each, and the results are summarised in the confusion matrix shown in Table I, left.

These results show that, although the Viola-Jones has an outstanding true positives rate of 98%, it performs poorly in false negatives with a rate of 24%. A second drawback of the Viola-Jones approach is that a person has to directly face the robot to be detected. This is a major problem for our application as we want a person to describe a trajectory in the space, and this would not be easy if he/she has to face all the time the robot. A third drawback of the Viola-Jones approach is that it can detect people only up to about $2m$ away from the camera when using wide angle lens. We will describe a solution to this problem in section II-3 with the introduction of the particle filter for tracking.

2) RBFN for legs detection: The Viola-Jones approach described above relies on vision to detect faces. A robot is usually equipped with several sensors, which can be combined to make a classifier stronger. In the past the laser sensor has been used to detect people using their legs, either as the sole sensor [6] or together with a camera [7], [8]. In most of the previous works, in order to train a laser based classifier a huge set of legs laser scans had to be manually constructed and labeled [6], or the authors created an ad-hoc classification algorithm [7]. Our approach is to use the face detection algorithm described before to train a Radial Basis Function Network (RBFN) classifier [9] for legs detection. This way the training process is completely automated.

In order to train a RBFN to classify leg patterns in laser scans, we collected the training data using the Viola-Jones face detection algorithm described above. Specifically, we had the robot running for 20 minutes with a person constantly in front of it, while recording the laser scan readings in a set C_1. As we stated before, the detection rate of the face detection algorithm is 98%, so almost all the laser scan readings refer to legs. The camera had been calibrated so that for every pixel it is possible to calculate the angle θ between that pixel and the camera itself. Considering that both the camera and the laser are vertically aligned, θ identifies a unique laser reading, taken at angle θ, in a whole laser scan. For every face detected, its centroid is extracted and the corresponding angle θ is calculated. Every laser scan in C_1 is then clipped between angles $\theta - \pi/6$ and $\theta + \pi/6$. This way C_1 is a training set for the class 1, "legs in a laser scan", composed by 60-dimensional vectors. We then built a second set of laser scans C_0 by letting the robot randomly move in an environment with no people in it for 20 minutes. This set represents all the laser scans with no legs in it. Both C_0 and C_1 are then been used to train the RBFN.

The face detector and the legs detector have been combined to create a new people detection algorithm, that outputs 1 if both Viola-Jones and the RBFN detect a person. The new people detector confusion matrix is summarised in Table I, right. It can be seen that the true positives detection rate dropped from 98% to 78%, but the number of false positives is now 0%, thus solving the problem with the Viola-Jones only approach. Once both classifiers agree that a person has been detected, the robot can switch to laser classification only. This solves the problem of a person having to face the robot camera all the time.

The only drawback of this approach is that the training set C_1 contains legs patterns which are only closer than about $2m$, because this set was "created" by the Viola-Jones detector. This means that the RBFN legs classifier can detect people only up to about $2m$.

3) Particle filter: The 2m limit described before does not allow a person to move arbitrarily in the environment, which is necessary to describe shapes. In order to solve this problem we employed a particle filter. An excellent review of this technique is in [10], while an application of it to people tracking using a laser sensor is in [11].

A particle filter requires a big number of particles to work reliably [12]. As a computational trade-off, in our application each particle does not try to detect people, but it tracks only a single laser scan. For this reason the adopted tracker relies heavily on a people detector that does not report false positives.

We tested the particle filter in several experiments using 2000 particles. In the worst case the tracker failed after 4 minutes of continuous operations, but in the average it lasted around $10m$. Moreover, the particle filter is able to track a person movements up to 8 meters away from it.

III. TRAJECTORY FOLLOWING

An arbitrary shape is represented by the robot as a parametric B-Spline. A motor controller that integrates feedforward and feedback signals is used to drive the robot along that shape. In the following we will give an overview of both B-Splines and the proposed controller.

A. B-Splines

A Basic Spline (B-Spline) is a parametric curve often used for interpolation and regression [13]. They have been widely used in robotics to approximate trajectories [14], [15], [16].

Given $m+1$ real numbers $t_0 \leq t_1 \leq \cdots \leq t_m$ and $m+1$ *control points* $p_i, i = 0, \cdots, m$, a B-Spline of degree n is a parametric curve $S(t)$ composed of a linear combination of basis functions $b_{i,n}$ of degree n, as given in (1).

$$S(t) = \sum_{i=0}^{m} p_i b_{i,n}(t) \tag{1}$$

where $b_{i,n}(t) \neq 0$ only if $t_{i-1} \leq t < t_i$. This makes a B-spline piecewise defined, i.e. the basis functions are non-zero only in a closed interval. In order to obtain a C^2 smooth function, usually a cubic polynomial is used as a basis function $b_{i,3}$. When used to approximate a parametric curve, B-Splines are defined as in (2).

$$S(t) \equiv (S_x(t), S_y(t)), t = 0, \cdots, 1 \tag{2}$$

where $S_x(t)$ and $S_y(t)$ are two B-Splines.

A B-Spline can be used to approximate a function $f(x)$ represented as a finite set of points (x_i, y_i). In this case the sum of squares error (3) is zero.

$$\sum_{i=1}^{N} (y_i - S(x_i))^2 \tag{3}$$

When the data points (x_i, y_i) are noisy, the interpolation requirement is not reasonable. In this case the smoothing spline in (4) is preferred [17].

$$\sum_{i=1}^{N} (y_i - S(x_i))^2 + \lambda \int_{x_1}^{x} N \left[\ddot{S}(x)\right]^2 dx = \min \tag{4}$$

where $\lambda \geq 0$ is a *smoothing factor*. When $\lambda = 0$, we obtain the interpolation again. The higher λ, the less the spline is constrained to pass through the data points (x_i, y_i). In the following we will use λ to generate smooth approximations of people trajectories.

B-Splines are *affine invariant*, i.e. if an affine transformation is applied to a B-spline curve, the result can be constructed from the affine images of its control points. This is very important in our application as we want the robot to show the effect of geometric transformations on shapes. Moreover, the derivatives of a B-Spline can be analytically calculated. This will come at hand for the development of the controller.

B. Controller

The trajectory tracking for mobile robots is characteristically a nonlinear problem. Several solutions have been proposed in the past. However, in most of them only simulation results are presented [18], [19], [20], or they require complex calculations that are not feasible in a real-time controller [21], [22]. In this work we decided to adopt a feedforward-feedback control law: the robot follows the trajectory described by the B-Spline, and at the same time tries to minimise a distance error using a PID controller.

1) Feedforwad control law: The motion of a unicycle robot can be described by the system of differential equations in (5).

$$\begin{cases} \dot{x} = v \cos\theta \\ \dot{y} = v \sin\theta \\ \dot{\theta} = \omega \end{cases} \tag{5}$$

where x, y, θ are the robot position and orientation, and v, ω are the robot linear and angular speeds. If we consider a time interval dt sufficiently small, then we can approximate the robot movement as piecewise circular, where the radius ρ of the circle in the time interval $[t, t+dt]$ is in (6).

$$\rho = \frac{v(t)}{\omega(t)} \tag{6}$$

Any parametric curve $(x(t), y(t))$ can be approximated the same way by a set of arcs, whose curvature κ is in (7).

$$\kappa = \frac{|\dot{x}\ddot{y} - \dot{y}\ddot{x}|}{(\dot{x}^2 + \dot{y}^2)^{\frac{3}{2}}} \tag{7}$$

Considering that the curvature of a circle of radius ρ is constantly equal to the reciprocal of the radius, we can rewrite (6) as in (8).

$$\frac{v(t)}{\omega(t)} = \frac{1}{\kappa} \tag{8}$$

Moreover, the angular coefficient of the tangent to a point (x_0, y_0) of a parametric curve is given in (9), which is also the instantaneous linear speed v of a point moving along that curve.

$$v = \sqrt{\dot{x}(x_0) + \dot{y}(y_0)} \tag{9}$$

By substituting (7) and (9) into (8), we obtain the feedforward control law for the robot angular speed in (10), which is a function of the B-Spline derivatives and the robot linear speed.

$$\omega_f = \frac{|\dot{x}\ddot{y} - \dot{y}\ddot{x}|}{\dot{x}^2 + \dot{y}^2} v \tag{10}$$

As outlined in the introduction, the user can select two parameters, namely the time approximation dt and the robot speed v. Once the user has selected them, the robot shows a simulated trajectory. This is created by using a first order Euler approximation of (5), where ω is given by (10) and dt, v are supplied by the user. This is useful to check if the robot is theoretically able to follow a given trajectory. However, as (5) does not take into consideration the robot dynamics, this method is necessary but not sufficient to guarantee a correct trajectory following.

2) Feedback control law: The role of the feedback controller is to correct the robot when it deviates from the desired trajectory. This deviation can be calculated by estimating the distance between the robot and the trajectory. As there is no analytical solution to this problem, we used the Newton method to find a zero of the function

Given a point (p, q) and a parametric curve $(x(t), y(t))$, the squared distance between the point and any other point on the curve is given in (11).

$$\Delta x(t)^2 + \Delta y(t)^2 \tag{11}$$

where we defined $\Delta x(t) = x(t) - p$ and $\Delta y(t) = y(t) - p$. To find the minimum of (11) we differentiate and equate to zero, as in (12)

$$2\Delta x(t)\dot{x} + 2\Delta y(t)\dot{y} = 0 \tag{12}$$

The Newton method is an iterative method to find the zeros of a function $f(t)$. It starts with a guess t_0 and every step it updated the candidate solution as in (13)

$$t_i \leftarrow t_{i-1} - \frac{f(t)}{\dot{f}(t)} \tag{13}$$

By applying (12) to (13) we obtain the iteration step (14) to find the minimum distance between a point (p, q) and a curve trajectory.

$$t_i \leftarrow t_{i-1} - \frac{\Delta x\dot{x} + \Delta y\dot{y}}{\dot{x}^2 + \dot{y}^2 + \Delta x\ddot{x} + \Delta y\ddot{y}} \tag{14}$$

During our experiments we found that the Newton method requires only a few iterations to converge to a solution, so it is usable in a real-time controller.

If the robot position is (x_r, y_r) and the closest point on the desired trajectory is (x_d, y_d), then we can define the angular error as in (15).

$$e_\theta = \arctan\left(\frac{y_d - y_r}{x_d - x_r}\right) \tag{15}$$

The feedback control law is a PID with the error in (15) to steer the robot. The final robot controller is given in (16)

$$\begin{cases} v = const \\ \omega = \omega_f + \omega_b \end{cases} \tag{16}$$

where ω_f is the output of the feedforward controller (10) and ω_b is the output of the PID when supplied with the angular error (15).

IV. Experimental results

All the experimental results have been produced using a Scitos-G5 robotics platform equipped with a laser range finder and a camera (Figure 1). The trajectories have been recorded using the Vicon tracking system, that allows tracking an object with an accuracy of $1mm$. Every trajectory, before being replicated by the robot, has been rotated and translated so that it starts from the robot position and it is aligned with the robot x axis.

We performed two groups of experiments: in the first group the user asks the robot to follow a pure geometric shape, while in the second group the user defines a shape by moving in the environment. For every shape we show the corresponding one in the robot frame of reference, the simulated approximation (see section III-B1) and the real shape as produced by the robot movements. Table II shows the parameters and the statistics for every shape.

For every shape we calculated the approximation error as the mean absolute distance between the user-provided trajectory and the approximated one. The errors for each experiment are summarised in Table II. Note that if we were using the mean error the results would have been close to zero, as they are equally positive and negative (cf. Figures 2,3 and 4). However, the mean absolute error presents a clearer picture of the worst case scenarios.

TABLE II
Parameters and statistics for the tested trajectories.

	v	dt	λ	error [m]
Cosine	0.3	0.08	0	0.17
Spiral	0.2	0.08	0	0.21
Square	0.3	0.08	0.01	0.2
Sheared square	0.3	0.08	0.01	0.19
Trajectory 1	0.4	0.1	2	0.30
Trajectory 2	0.4	0.1	2	0.28

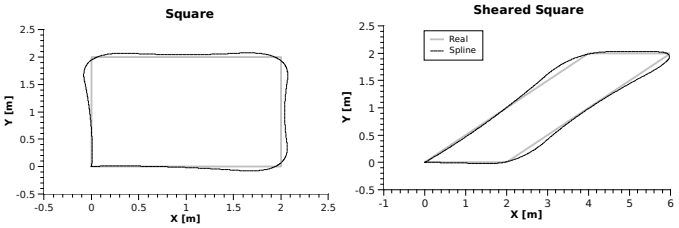

Fig. 2. The square and the sheared square used during our experiments.

3) Mathematical shapes: We tested four mathematical shapes: i) a cosine, ii) a spiral, iii) a square and iv) a sheared square. Figure 2 shows the last two shapes together with the spline approximation. The approximations of the cosine and the spiral match exactly the original data, so they are not shown here. Figure 3 shows the simulated and the real robot trajectories for all the mathematical shapes.

4) User produced shapes: We tested two user produced shapes. Both of them have been produced by the user moving in front of the robot while being tracked. Figure 4 top row shows the trajectories as observed by the tracker and the corresponding spline approximation. Figure 4 bottom row shows the simulated and the real robot trajectories for both the observed shapes.

5) Discussion: Among the mathematical shapes, both the square and the sheared square have discontinuities at the corners. For this reason the splines deviate significantly from the desired shapes when around the corners. This is due to the "smooth" nature of splines, which are not suitable to approximate a piecewise linear curve like a square. In order to avoid sharp turns at the corners, we opted for a slightly smoothed approximation with a λ parameter of 0.01.

The average error along all of the mathematical shapes is low, as shown in Table II. The only time the robot noticeably deviated from the desired trajectory was at the beginning of the spiral shape, as shown in Figure 3, top right. This is due to the initial high curvature of the spiral, which is not reproducible by the robot given its physical constraints. However, the feedback control law quickly corrected the error and drove the robot back on the desired trajectory. The same graph highlight the differences between the real trajectory and the simulated one, and the role physical constraints play in a robot controller.

The sheared square has been produced by shearing the square by $1m$ along the x axis. This shows how our proposed system can be used to interactively teach affine transformations.

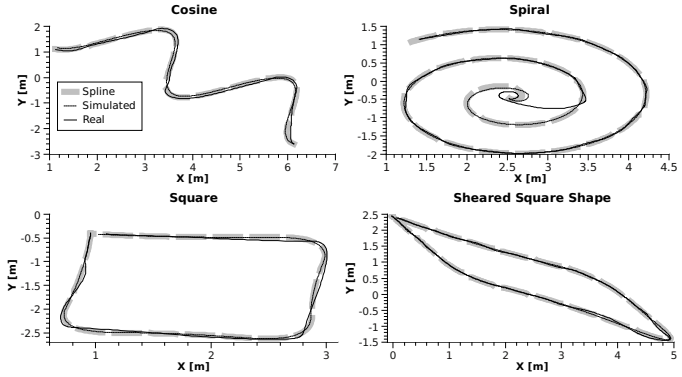

Fig. 3. The mathematical shapes with the corresponding simulated and real robot trajectories. All the graphs are rotated to match the robot frame of reference.

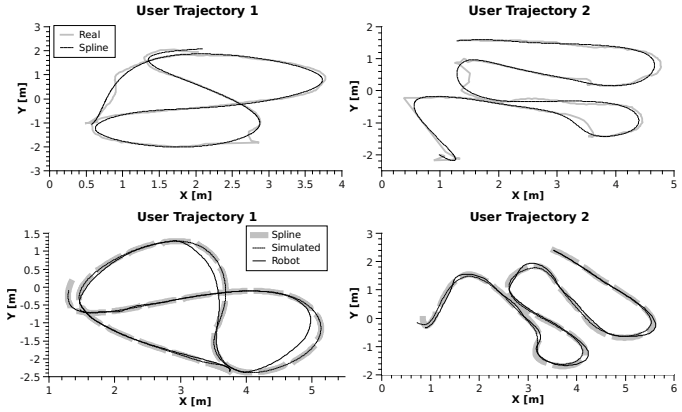

Fig. 4. (Top row) The user produced trajectories observed by the tracker, and the corresponding spline approximations. (Bottom row) The simulated and real robot trajectories. All the graphs are rotated to match the robot frame of reference.

The trajectories observed by the tracker are noisy and discontinuous in several points, as shown in Figure 4. Trajectory 2 benefited the most from the parameter λ, with the effect of obtaining a smooth non interpolating curve. This is reflected in the higher errors both the simulated and the real trajectories exhibit.

V. Conclusions and Future Work

In this paper we presented a robotic system to interactively teach basic geometry. The interaction with a student happens when the user selects a shape, and when the robot shows what the shape will look like when it will reproduce it, given the parameters chosen by the student. At the end of the interaction the robot describes the shape in the environment by moving along it.

The proposed system is composed of two main parts: a passive one, which detects and tracks people movements, and an active one, which plans the robot movements and drives it along a shape. We showed with several experiments that the system is reliably able to carry on both tasks.

In this paper we focused mainly on the application an on the results. In the future we will perform a survey among students and teachers to identify the main points where this system can be improved. We will also modify the controller so that it will include the robot physical constraints.

References

[1] H. Abelson and A. A. Disessa, *Turtle geometry: The computer as a medium for exploring mathematics*. The MIT Press, 1986.

[2] T. Fujita, K. Jones, and S. Yamamoto, "Geometrical intuition and the learning and teaching of geometry," in *Topic Study Group on the teaching of geometry at the 10th International Congress on Mathematical Education*, 2004, p. 411.

[3] C. Laborde, C. Kynigos, K. Hollebrands, and R. Strasser, "Teaching and learning geometry with technology," in *Handbook of research on the psychology of mathematics education: Past, present and future*, 2006, p. 275304.

[4] K. Jones, "Issues in the teaching and learning of geometry," in *Aspects of Teaching Secondary Mathematics: perspectives on practice*, 2002, pp. 121–139.

[5] P. Viola and M. J. Jones, "Robust Real-Time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, May 2004.

[6] K. O. Arras, O. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2d range data," in *Proc. of the int. conf. on robotics & automation*, 2007.

[7] N. Bellotto and H. Hu, "Multisensor-Based human detection and tracking for mobile service robots," *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSPART B: CYBERNETICS*, vol. 39, no. 1, p. 167, 2009.

[8] M. Kobilarov, G. Sukhatme, J. Hyams, and P. Batavia, "People tracking and following with mobile robot using an omnidirectional camera and a laser," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2006, p. 557562.

[9] T. Poggio and F. Girosi, "Networks for approximation and learning," *Proceedings of the IEEE (special issue: Neural Networks I: Theory and Modeling)*, vol. 78, no. 9, p. 14811497, 1990.

[10] A. Doucet, N. D. Freitas, and N. Gordon, *Sequential Monte Carlo methods in practice*. Springer Verlag, 2001.

[11] D. Schulz, W. Burgard, D. Fox, and A. B. Cremers, "People tracking with mobile robots using sample-based joint probabilistic data association filters," *The International Journal of Robotics Research*, vol. 22, no. 2, p. 99, 2003.

[12] S. Thrun, "Probabilistic algorithms in robotics," *AI Magazine*, vol. 21, no. 4, p. 93, 2000.

[13] C. de Boor, *A Practical Guide to Splines*. Springer-Verlag, 1978.

[14] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the DARPA grand challenge," *Journal of field Robotics*, vol. 23, no. 9, p. 661692, 2006.

[15] S. Gulati and B. Kuipers, "High performance control for graceful motion of an intelligent wheelchair," in *IEEE International Conference on Robotics and Automation*, 2006, p. 39323938.

[16] H. Eren, C. C. Fung, and J. Evans, "Implementation of the spline method for mobile robot path control," in *Instrumentation and Measurement Technology Conference, 1999. IMTC/99. Proceedings of the 16th IEEE*, vol. 2, 1999, pp. 739–744 vol.2.

[17] P. H. C. Eilers and B. D. Marx, "Flexible smoothing with b-splines and penalties," *Statistical Science*, vol. 11, no. 2, pp. 89–121, 1996.

[18] E. Guechi, J. Lauber, and M. Dambrine, "On-line moving-obstacle avoidance using piecewise bezier curves with unknown obstacle trajectory," in *Control and Automation, 2008 16th Mediterranean Conference on*, 2008, pp. 505–510.

[19] J. H. Hwang, R. C. Arkin, and D. S. Kwon, "Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003, p. 14441449.

[20] B. V. G., H. S. A., and J. D. S., "Auto guided vehicle control using expanded time b-splines," in *Systems, Man, and Cybernetics, 1994. 'Humans, Information and Technology'., 1994 IEEE International Conference on*, vol. 3, 1994, pp. 2786–2791 vol. 3.

[21] L. Lapierre, D. Soetanto, and A. Pascoal, "Nonsingular path following control of a unicycle in the presence of parametric modelling uncertainties," *International Journal of Robust and Nonlinear Control*, vol. 16, no. 10, p. 485504, 2006.

[22] C. G. Bianco, A. Piazzi, and M. Romano, "Smooth motion generation for unicycle mobile robots via dynamic path inversion," *Trans. Robot. Automat*, vol. 17, p. 413422, 2001.