Mobile robots at your fingertip: Bezier curve on-line trajectory generation for supervisory control

Jung-Hoon Hwang[1], Ronald C. Arkin[2], Dong-Soo Kwon[1]

(hwangjh@robot.kaist.ac.kr, arkin@cc.gatech.edu, kwonds@kaist.ac.kr)

[1]Department of Mechanical Engineering, Korea Advanced Institute of Science and Technology, Daejeon, Korea

[2]College of Computing, Georgia Institute of Technology, Atlanta, GA, USA

Abstract - A new interfacing method is presented to control mobile robot(s) in a supervised manner. Mobile robots often provide global position information to an operator. This research describes a method whereby the operator controls a mobile robot(s) using his finger or stylus via a touchpad or touchscreen interface. Using a mapping between the robot's operational site and the input device, a human user can provide routing information for the mobile robot.

Two algorithms have been developed to create the robot trajectory from the operator's input. Information regarding numerous path points is generated when the operator moves his finger/stylus. To prune away meaningless point information, a simple but powerful significant points extracting algorithm is developed. The resulting significant points are used as waypoints.

An on-line piecewise cubic Bezier curves (PCBC) trajectory generation algorithm is presented to create a smooth trajectory for these significant points. As the method is based on distance and not on time, the velocity of mobile robot can be controlled easily within its allowable dynamic range. The PCBC trajectory can also be modified on the fly. Simulation results are presented to verify these newly developed methods.

I. INTRODUCTION

A joystick is a common interface to teleoperate a mobile robot. Joystick control, however, can quickly become a tedious and time-consuming task. Since the joystick requires continuous input and attention from the operator, when the user is expected to control more than one mobile robot at a time, it becomes at best extremely difficult to do so without using special methods.

To solve this problem, supervisory control can be used. This paper presents an interface and underlying control algorithms that enable an operator to manipulate a robot manually with minimal effort. Such an interface should be intuitive and simple to reduce the cognitive workload of the operator. At a minimum, 2D spatial input is needed to operate the robot in order to convey the operator's intentional goals. There exist numerous hardware interfaces that can capture the operator's 2D input. Most (e.g., mouse, tablet - 2D digitizer, and many haptic devices) can be used as a supervisory interface with or without video feedback. We chose a touch pad/screen with visual feedback for this research. Figure 1 shows a conceptual overview of the approach.

There is relatively little research involving the use of touch pad/screen for mobile robot teleoperation. For example, Fong *et al* used a touchscreen device (PDA) as a collaborative interface for robot [1]. In this work, they used the touchscreen as a virtual joystick or to generate typical waypoints for the manual control of robot.

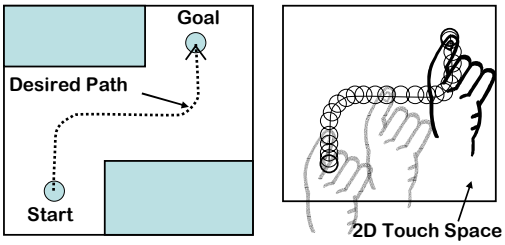


Figure 1. Conceptual diagram of a proposed interface for mobile robot(s)

In this paper, we propose a trajectory generation method using the user's fingertip motions, which is a very simple and intuitive way to create a mobile robot trajectory. The robot's route information representing the operator's intention can be easily obtained from the touch pad/screen. With a collection of simple gestures, e.g., tapping or dragging coupled with a recognition capability, selecting which robot(s) to move and various options for trajectory generation can be performed.

To create a finger-generated route on a touchpad into a smooth mobile robot trajectory, several steps are required. The first step is the generation of a mapping between the input 2D touch space area to the robot's operational 2D space (fig. 2(2)). The second step involves extracting a set of significant points from a large number of points regarding the route information (fig. 2(3)). The third step requires producing a smooth trajectory that matches the requirements generated by the set of extracted significant points (fig. 2(4)). The final step is involves an on-line modification of the planned trajectory in a dynamic environment (fig. 2(5)).

In Section II, the operational space mapping and the significant point set extraction are described. A Bezier curve on-line trajectory generation method including on-line trajectory modification is described in Section III. Simulation results using a real touchpad are presented in Section IV. In Section V, conclusions and future work are discussed.

II. MAPPING AND SIGNIFICANT POINTS EXTRACTING ALGORITHM

A. Mapping

The operational space of a mobile robot is often

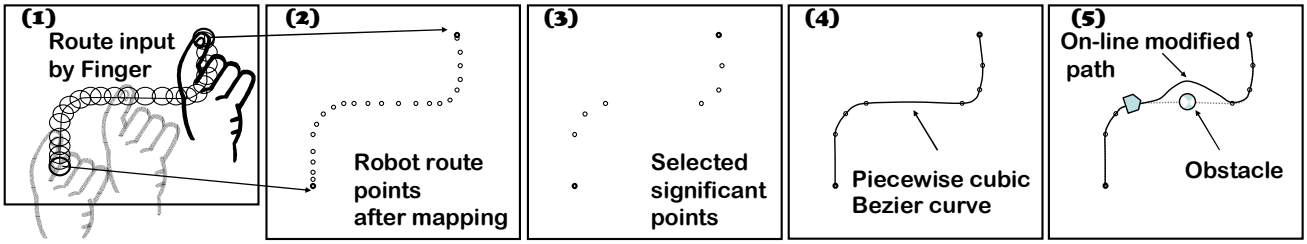


Figure 2. Overall Scope of this paper: (1) route input from 2D touch space by finger, (2) robot route after scaling or topological mapping from input of (1), (3) significant point extraction using linear tolerance scheme, (4) piecewise cubic Bezier curves(PCBC) trajectory generation, (5) consideration of on-line modification of PCBC trajectory

physically bounded. For example, if the normal operational space is the interior of a building, the outside of a home or a fenced yard, the workspace of the mobile robot can be readily determined. Such 2D workspaces can often be bounded using a rectangle or bounding box. The perpendicular ratio and the scale of the bounding rectangle can be different from that of a standard 2D input device, and thus a spatial mapping between them must be created.

When the operator generates an input command using a touch pad/screen, the finger route information that consists of many input points is recorded. The desired route information for a mobile robot can be reconstructed from that information using scaling or topological mapping, Even for a topological mapping, humans can adapt to this form of mapping with some practice.

B. Significant points extraction algorithm

The initial route information produced from the operator's fingertip still needs to be manipulated since the actual robot's route needs to be smoothed. Several significant points extraction algorithms have been used in CAD/CAM and computer graphics to digitize old paper drawings or drafts. The most common and simple method uses corner detection. By detecting a sharp corner point, the entire drawing can be vectorized. Sharp corners can be detected by checking the curvature or the radius of the curve [2,3]. The basis for one of the most popular algorithms is given in equation (1). This algorithm is good at extracting points, which can then be used to represent almost exactly the shape of a given curve.

$$C^k(i) = a_{ik} \cdot b_{ik} \Big/ |a_{ik}||b_{ik}| \qquad i: \text{test point}, k: \text{test interval}$$
$$\text{where } a_{ik} = (x_i - x_{i+k}, y_i - y_{i+k}), \quad b_{ik} = (x_i - x_{i-k}, y_i - y_{i-k}) \qquad (1)$$
$$i \text{ is significant point if } C(i) \text{ is local maxima} \;\&\; C(i) > \text{threshold}$$

A mobile robot, however, does not need to follow the exactly the entire set of mapping points of operator inputs, but rather it only needs a smaller number of waypoints, since the route information from operator input can be noisy or locally meaningless. A new simple, yet significant method, has been developed for this reason. Figure 3 shows the algorithm in flowchart format. The main idea lies in detecting points that are outside of a given linear tolerance.

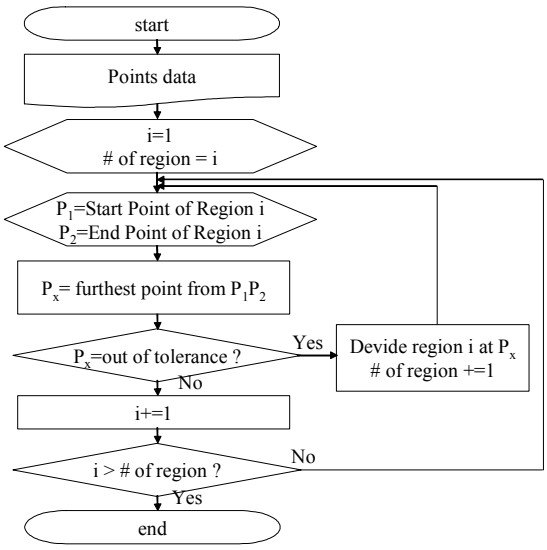


Figure 3. Significant point extracting algorithm

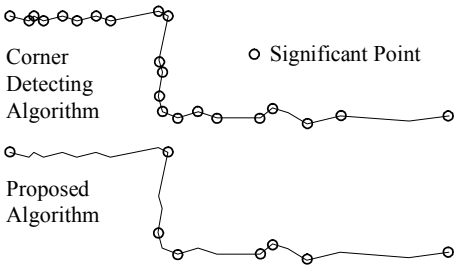


Figure 4. Comparison of significant point extraction algorithms

Figure 4 shows typical results for the corner detection algorithm in equation (1) and the new algorithm in Figure 3. Since equation (1) depends on the angular difference between three points and the new method depends on the tolerance between three points, the new method easily can be made to extract less significant points, but still are enough to represent the global shape of the curve, while excluding small noisy variations of the curve. Furthermore, the corner detection algorithm has two parameters, a test interval k and threshold in equation (1), which need to be tuned, while the new method has only one parameter, a linear tolerance.

III. BEZIER CURVE ONLINE TRAJECTORY GENERATION ALGORITHM

As seen above, the waypoints for a mobile robot can be obtained using a significant point extraction algorithm. In order to make a smooth and continuous trajectory from the resulting waypoints, there exist a few trajectory-planning algorithms which incorporate a spline-based methodology.

Komoriya *et al* incorporated the B-spline in path-planning [4]. They added straight-line segments to make the overall trajectory close to the intended one. But their trajectory did not pass through the exact waypoint. Vazquez *et al* also used B-splines in path-planning by adding a time variable [5]. The time variable was not linear, however, but monotonically increased with real time. The speed of the mobile robot was controlled by the given B-spline itself. Zhang *et al* created a fuzzy trajectory controller, which emulates the B-spline [6]. They stated that the robot could pass through the subgoal, although their robot seemed to be unable to pass through the exact subgoal in their simulation. Khatib *et al* proposed the bubble band concept, connecting the bubbles with bezier curves [7]. They could generate collision free paths to the destination by moving these bubbles. The trajectory was restricted to where the bubbles were located. Eren *et al* also utilized the spline in path-planning [8]. They generated several interior points within a spline and let the robot follow the points in succession. Yamamoto *et al* considered the dynamic and kinematic constraint in B-spline-based path-planning to find the time optimal trajectory in a static environment [9]. Nagatani et al also used the bezier curve in path-planning [10], considering the minimum radius of curvature of vehicle.

Most previous research regarding spline-based path-planning considers a static environment except Khatib's, but even their research had a limitation, which was mentioned above. Since the mobile robot will be used in a dynamic environment which has unexpected or moving obstacles, its trajectory needs to be modified in on-line. With our on-line smooth trajectory generation algorithm, piecewise cubic bezier curves are used and on-line modification of the curves is studied. Piecewise cubic bezier curves can provide C^3 within a curve and C^1 in the connection between two curves.

A. Piecewise cubic Bezier curves

Piecewise cubic Bezier curves (PCBC) are used between each waypoint to connect them with smooth curves. Equation (2) shows the equation for a cubic bezier curve [11]. Cubic Bezier curves require four control points (figure 5). The first control point is the start point, and the last is the end point. Each waypoint is used as a first and last control point to ensure that the resulting path passes through each significant point. Jensen presented a simple method for determining the second and third control point that makes a smooth connection between each cubic Bezier curve [11].

His algorithm is slightly modified in our research to consider the robot's initial and final direction. The tangent vector of the second control point of a first curve is set to the robot initial direction, and the tangent vector of the third control of a last curve is set to the robot's final direction if the final direction is specified.

B. Direction vector generation

Since the parameter of Bezier curves is not time or distance dependent, the curves generated as described in Section 3.1 cannot be used directly as a trajectory. Some researchers used a modification of parameter *t* of curves as a time variable [5,7] while others used an approximation of the curves by dividing the parameter *t* into n intervals [4,8].

In this paper, a numerical method is used to get the parameter *t* of bezier curves with respect to the moving distance *S* of a robot. If the curve parameter *t* can be obtained at the current mobile robot position, the directional vector of the curve can then be calculated at that position. The position on the curve is expressed by parameter *t* as shown in equation (2). The derivative of equation (2) is shown in equation (3). The relation between the distance and the parameter *t* can be expressed as shown in equation (4). To calculate the parameter *t* with respect to the robot's position, the inverse of equation (4) is integrated by a fourth order Runge-Kutta method as shown in equation (5).

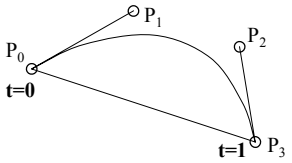


Figure 5. Cubic Bezier curve

$$\vec{p}(t) = \vec{a}t^3 + \vec{b}t^2 + \vec{c}t + \vec{p}_0, \quad \text{For } 0 < t < 1$$
$$\text{where } \vec{p} = (x, y),\ \vec{c} = 3(\vec{p}_1 - \vec{p}_0) \tag{2}$$
$$\vec{b} = 3(\vec{p}_2 - \vec{p}_1) - \vec{c},\ \vec{a} = \vec{p}_3 - \vec{p}_0 - \vec{b} - \vec{c}$$

$$\begin{matrix} \frac{d}{dt}x(t) = 3a_x t^2 + 2b_x t + c_x t \\ \frac{d}{dt}y(t) = 3a_y t^2 + 2b_y t + c_y t \end{matrix}, \text{ where } t \neq 0,1 \tag{3}$$

$$\frac{ds}{dt} = \sqrt{\left(\frac{d}{dt}x(t)\right)^2 + \left(\frac{d}{dt}y(t)\right)^2} \tag{4}$$
$$= S(t)$$

$$t_{n+1} = t_n + T(t_n, s_n)$$
$$\text{where } T(t_n, s_n) = \Delta s \frac{(f_1 + 2f_2 + 2f_3 + f_4)}{6} \tag{5}$$
$$f_1 = \frac{1}{S(t_n)}, \quad \cdots, \quad f_4 = \frac{1}{S(t_n + \Delta s f_3)}$$

The directional vector at the known parameter t is shown in equation (6).

$$\vec{V} = \left(\frac{dx(t)}{dt}, \frac{dy(t)}{dt} \right) \tag{6}$$

C. On-line modification of trajectory

For most cases of teleoperation, there exists an unstructured environment and the mobile robot is required to cope with that circumstance. Arkin *et al* presented a dodging behavior to avoid both moving and stationary obstacles for a schema-based reactive control system [12]. Arkin's motor schema method offers several advantages such as there is no predefined behavioral hierarchy and no arbitration is required for coordination [13]. We combined our teleoperated trajectory generation scheme with this purely reactive dodging behavior for motor schema-based reactive control. Figure 6 depicts the dodging behavior. The collision zone means the predicted collision area. Since our trajectory generation method can produce a desired direction vector at the current mobile robot's position, it is easily integrated with motor schema method.

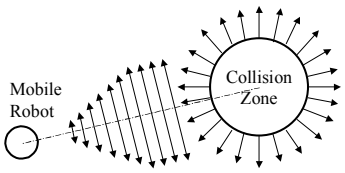


Figure 6. Dodging behavior

There is, however, one problem. If obstacles are present, the mobile robot may require leaving the current PCBC trajectory. To solve this problem and to make the trajectory adjustable, the position of the control point of the PCBC is recalculated in real time after numerically updating the curve parameter t. Equation (7) shows the recalculation rules. In each step, the position of only one control point is recalculated to reduce the computational overhead.

$$\begin{aligned} 0 < t \le 0.25 \quad & P_0 = \frac{\left(P - \left(3t(1-t)^2 P_1 + 3t^2(1-t)P_2 + t^3 P_3\right)\right)}{(1-t)^3} \\ & \vdots \\ 0.75 < t \quad & P_3 = \frac{\left(P - \left((1-t)^3 P_0 + 3t(1-t)^2 P_1 + 3t^2(1-t)P_2\right)\right)}{t^3} \end{aligned} \tag{7}$$

By moving one control point, the mobile robot can be on the PCBC trajectory even though the robot needs to change its path from the planned one due to dynamic events, and the mobile robot can continue following the locally modified PCBC trajectory.

D. Reference velocity generation

By virtue of the algorithm's distance-based path-planning capability, the velocity of a mobile robot can be computed easily. This method determines two conditions in velocity planning. The first condition requires maintaining a maximum velocity except in the presence of the second condition, which requires maintaining the centrifugal force under a safe value, to ensure that the robot does not suffer from sideslip.

The two wheeled mobile robot used in this paper is shown in figure 7. Since each wheel has a speed limit, the

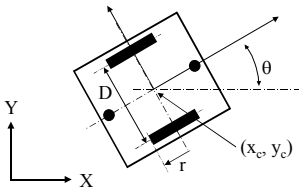


Figure 7. Kinematics of the mobile robot

maximum linear velocity of the robot depends on the angular velocity. Equation (8) is used to keep the robot direction aligned with the directional vector, which is generated by the methods described in Sections 3.2 and 3.3. V_r and V_l represent the linear velocities of each wheel.

$$\text{If } \dot{\theta}(s) > 0 \qquad V_r(s) = V_{\max}, V_l(s) = \frac{\left(V_{\max} - \frac{D}{2}\dot{\theta}\right)}{\left(V_{\max} + \frac{D}{2}\dot{\theta}\right)} V_r \tag{8}$$

$$\text{else vice versa} \qquad V_f(s) = \frac{\left(V_l + V_r\right)}{2}$$

$$\text{where } \dot{\theta}(s) = \frac{\theta}{\Delta t} \text{ and } V_{\max} : \text{maximum linear velocity}$$

These velocities are computed again to ensure that the robot centrifugal force is within a safe value. Equation (9) is the acceleration value induced by the robot's centrifugal force and the velocity bound is then derived from that. To obtain the radius of curvature (figure 8), an approximation equation (10) is used.

$$a_{centrifugal} = \frac{V^2}{R}$$

$$\therefore V_{Max\_centrifugal} = \sqrt{R a_{Max\_centrifugal}} \tag{9}$$

$$R = \left| \frac{uvw}{2E} \right|$$

where u, v, w are distance between points and (10)

$p_{n-1} = (a, b)$, $p_n = (c, d)$,

$$p_{n+1} = (e, f) = p_n + V_f \cdot \Delta t \cdot \frac{\vec{V}}{\left|\vec{V}\right|}$$

$$E = \left| ad + cf + eb - bc - de - fa \right|$$

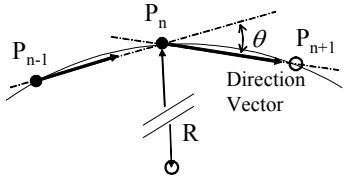
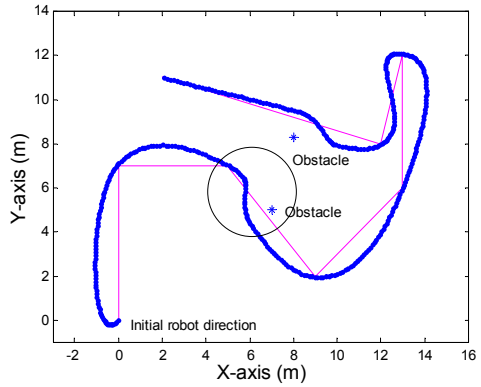
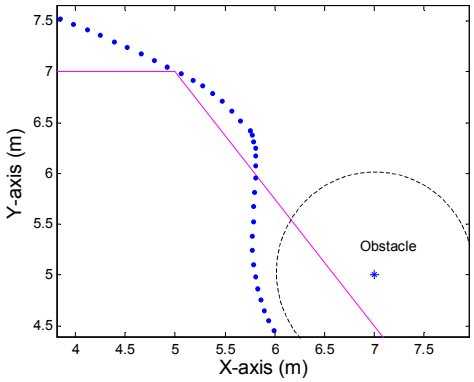


Figure 8. Directional vector and radius of curvature

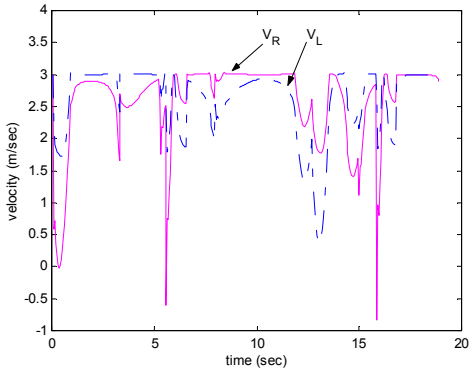
V_f is regulated by the value of $V_{Max\_centrifugal}$ that is function of R. The radius of curvature R, on the other hand, depends on the value of P_{n+1} which is calculated using V_f. The functional relationalships between V_f, $V_{Max\_centrifugal}$, R and P_{n+1} are cross-linked. To get the V_f that ensures the robot centrifugal force within a safe value, an iterative method is used.



(a) Overall trace



(b) Zoomed trace of circle zone



(c) Reference velocities for each wheel

Figure 9. Simulated results of the on-line PCBC trajectory generation algorithm

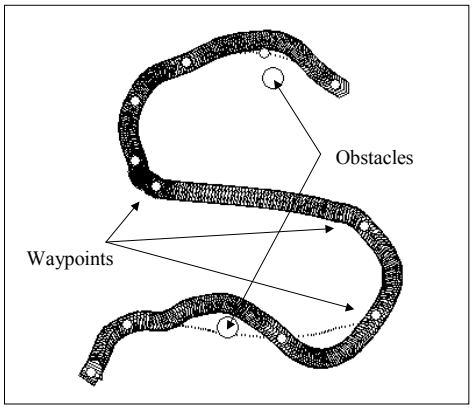
In figures 9(a) and (b), it can be shown that the PCBC trajectory can be modified on the fly when unexpected obstacles are encountered. The velocity of the mobile robot is varied by the turning angle as shown in figure 9(b). From the figure 9(c), it can be seen that the maximum velocity between wheels is reduced to make the robot's centrifugal force remain within a safe value (V_{max}=3m/s, $a_{Max\_centrifugal}$=3m/s^2).

IV. SIMULATION

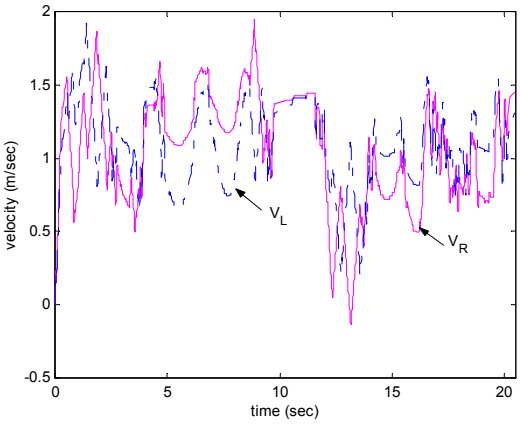
A virtual mobile robot control simulation using the touchpad interface has been conducted to evaluate these algorithms. The size of the touch pad interface is 0.06×0.04 m^2. Simple gesture recognition like tapping and dragging is incorporated to enable selecting a particular robot or to capture the finger-generated route information. During the interaction, the visual feedback of the finger-generated route is given.

TABLE I.
PARAMETERS OF PIONEER 2-DXE

D=0.4m	M=9kg	V_{max}=1.6m/s
b=0.44m	I_z=1.0741kgm^2	U_{max}=3Nm
R=0.1m	I_y=0.0075kgm^2	



(a) Trace of mobile robot



(b) Velocities of each wheel

Figure 10. Simulated results of realistic simulation with touchpad

The mobile robot dynamic equation developed by Yamamoto [9] and the parameters of a Pioneer 2-DXe (Table 1) are used to produce a realistic simulation. A simple PD controller which has a torque bound (*Umax*), is also used. The simulation runs at a 200Hz sampling rate

and the computation is performed in real-time without any significant burden on the CPU (Average CPU usage is lower than 1% for a Pentium3-800MHz).

The trace of a mobile robot and the generated waypoints that were extracted from the finger route are shown in figure 10(a). The dotted line is the finger route. The mobile robot can pass through each waypoint except when obstacles prevent it. Even when navigating around the obstacles, the trajectory was modified in real-time on-line. Figure 10(b) shows the velocity profile for each wheel. The maximum velocity between wheels is lower than V_{max} due to centrifugal force.

This on-line PCBC trajectory modification capability can be better seen in Figure 11. The mobile robot trajectory is continuously modified by a sudden moving obstacle until the obstacle goes out of the range of concern for the robot. After that, the mobile robot continues following the locally modified PCBC trajectory.

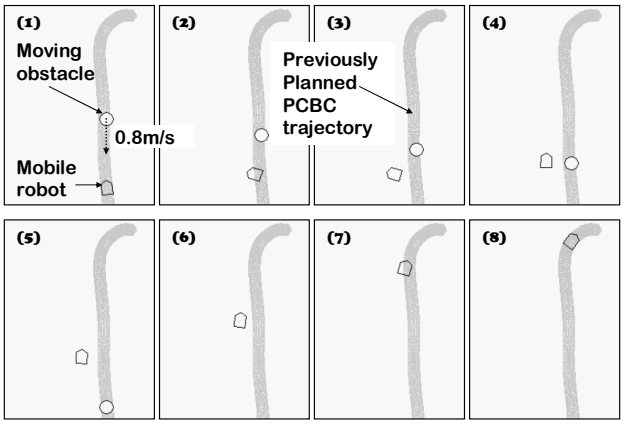


Figure 11. On-line PCBC trajectory modification: The gray trail shows a PCBC trajectory for no obstacle, and robot follows modified PCBC trajectory.

V. CONCLUSIONS

We presented a new interfacing method to control mobile robot(s) in a supervisory manner that is capable of real-time dynamic obstacle avoidance as well. To realize this method, two new algorithms were developed. One is a significant points extraction algorithm for noisy input data, and the other is an on-line piecewise cubic Bezier curves trajectory generation algorithm for a mobile robot. With the significant points extraction algorithm, extraneous significant points could be pruned from the noisy data. A smooth trajectory can also be achieved and modified on-line using our on-line PCBC trajectory generation algorithm.

Each algorithm can be extended to 3D space with simple modification. Also, using the touchpad/screen interfacing method, a robot trajectory can be specified in a simple and intuitive way. As such, this new interfacing method can be potentially used for various kinds of mobile robots such as a personal/house robot, military robot, service robot, and other robots that require easy operation for an end-user.

In the near future, we will apply the newly developed algorithms to real mobile robots and conduct quantitative experiments regarding their performance.

VI. ACKNOWLEDGMENT

This work was partly supported by the Brain Korea 21 Project in 2002.

VII. REFERENCE

[1] T. Fong, F. Conti, S. Grange, C. Baur, "Novel interfaces for remote driving: gesture, haptic and PDA," SPIE 4195-33, *SPIE Telemanipulator and Telepresence Technolgies* VII, Boston, MA, November, 2000.

[2] M. Sarfraz, M.A. Khan, "Automatic Outline capture of Arabic fonts," *Information Sciences* 140, pp. 269-281, 2002.

[3] S.C. Bae, I.S. Kweon, and C.D. Yoo, "COP: a new corner detector," *Pattern Recognition Letters* 23, pp. 1349-1360, 2002.

[4] K. Komoriya and K. Tanie, "Trajectory Design and Control of a Wheel-type Mobile Robot Using B-spline Curve," *Int. Workshop on Intelligent Robots and Systems*, pp. 398-405, 1989.

[5] B. Vazquez G., J. Humberto Sossa A. and Juan L. Diaz-de-Leon S, "Auto Guided Vehicle Control using Expanded Time B-Splines," *Int. Conf. on Systems, Man, and Cybernetics*, pp. 2786-2791, 1994.

[6] J. Zhang, J. Raczkowsky and A. Herp, "Emulation of Spline Curves and Its Applications in Robot Motion Control," *IEEE conf. on Fuzzy Systems*, pp. 831-836, 1994.

[7] M. Khatib, H. Jaouni, R. Chatila, J.P. Laumond, "Dynamic Path Modification for Car-Like Nonholonomic Mobile Robots," *Int. Conf. on Robotics and Automation*, pp. 2920-2925, 1997.

[8] H. Eren, C. C. Fung and J. Evans, "Implementation of the Spline Method for Mobile Robot Path Control," *Instrumentation and Measurement Technology Conference*, pp. 739-744, 1999.

[9] M. Yamamoto, M. Iwamura, and A. Mohri, "Quasi-Time-Optimal Motion Planning of Mobile Platforms in the Presence of Obstacles," *Int. Conf. on Robotics and Automation*, pp. 2958-2963, 1999.

[10] K. Nagatani, Y. Iwai, and Y. Tanaka, "Sensor Based Navigation for car-like mobile robots using Generalized Voronoi Graph," *Int. Conf. on Intelligent Robots and Systems*, pp. 1017-1022, 2001.

[11] Paul Bourke, "http://astronomy.swin.edu.au/~pbourke/curve/bezier/cubicbezier.html, " 2000.

[12] R. C. Arkin, W. M. Carter and D. C. Mackenzie, "Active Avoidance: Escape and Dodging Behaviors for Reactive Control," *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 7, No. 1, pp. 175-192, 1993.

[13] R. C. Arkin, *Behavior-Based Robotics*, MIT Press, Cambridge, 1998.