2009 American Control Conference
Hyatt Regency Riverfront, St. Louis, MO, USA
June 10-12, 2009

Obstacle Avoiding Real-Time Trajectory Generation and Control of Omnidirectional Vehicles

Ji-wung Choi, Renwick E. Curry and Gabriel Hugh Elkaim

Abstract— **In this paper, a computationally effective trajectory generation algorithm of omnidirectional mobile robots is proposed. The algorithm plans a reference path based on Bézier curves, which meet obstacle avoidance criteria. Then the algorithm solves the problem of motion planning for the robot to track the path in a short travel time while satisfying dynamic constraints and robustness to noise. Accelerations of the robot are computed such that they satisfy the time optimal condition for each sample time interval. The numerical simulation demonstrates the improvement of trajectory generation in terms of travel time, satisfaction of dynamic constraints and smooth motion control compared to previous research.**

I. INTRODUCTION

Many researchers have worked on vehicle motion planning. The form of the vehicle includes car-like, differential drive, omni-directional, and other models. Balkcom [3] developed the time optimal trajectories for the bounded velocity model of differential drive robots. Jung [4] and Moore [5] dealt with omnidirectional vehicles; the control strategy employed by these papers consists of building a geometric path and tracking the path by using feedback control. Huang [6] proposed an approach to vision-guided local navigation for nonholonomic robot based upon a model of human navigation. The approach uses the relative headings to the goal and to obstacles, the distance to the goal, and the angular width of obstacles, to compute a potential field. The potential field controls the angular acceleration of the robot, steering it toward the goal and away from obstacles. Hamner [7] maneuvered an outdoor mobile robot that learns to avoid collisions by observing a human driver operate a vehicle equipped with sensors that continuously produce a map of the local environment. The paper describes implementation of steering control that models human behavior in trying to avoid obstacles while trying to follow a desired path. Hwang [8] developed the trajectory tracking and obstacle avoidance of a car-like mobile robot within an intelligent space via mixed H_2/H_∞ decentralized control. Two CCD cameras are used to realize the position of the robot and the position of the obstacle. A reference command for the proposed controller of the robot is planned based on the information from these cameras.

J. Choi is a Ph.D. candidate in Computer Engineering Department at the University of California, Santa Cruz, 95064, USA. jwchoi@soe.ucsc.edu
R. Curry is an Adjunct Professor in Computer Engineering Department at the University of California, Santa Cruz, 95064, USA. rcurry@ucsc.edu
G. Elkaim is an assistant professor in Computer Engineering Department at the University of California, Santa Cruz, 95064, USA. elkaim@soe.ucsc.edu

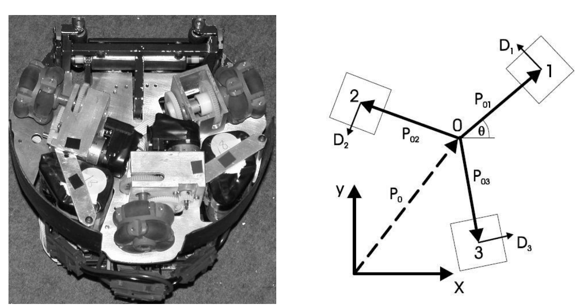
This paper focuses on two papers: Kalmar-Nagy [2] and Sahraei [1]. Kalmar-Nagy [2] has proposed a minimum time trajectory generation algorithm for omnidirectional vehicles, that meets dynamic constraints, but no obstacles are considered. A near-optimal control strategy is shown to be piecewise constant (bang-bang type) in the paper. Sahraei [1] has presented a motion planning algorithm for omnidirectional vehicles, based on the result of [2]. The paper has claimed that the algorithm satisfies obstacle avoidance as well as time optimality given in discrete time system.

The paper shows that Sahraei's algorithm is problematic. To resolve the problems, a new motion planning algorithm for omnidirectional vehicles is proposed, which also satisfies obstacle avoidance and dynamic constraints in a discrete time system. The numerical simulations provided in this paper demonstrate a better solution to the problem of motion planning by the proposed algorithm than Sahraei's.

The paper is organized as follows. Section II describes dynamic constraints of the robots based on the result of [2]. In section III, Sahraei's algorithm [1] is introduced. Section IV proposes the new algorithm. Finally, a numerical simulation is presented in Section V.

II. DYNAMIC CONSTRAINTS OF THE OMNIDIRECTIONAL VEHICLE

Fig. 1(a) shows the bottom view of an omnidirectional vehicle that consists of three wheels. This type of vehicle is able to move in any direction and spin as it moves. Kalmar-Nagy described a model that relates the amount of torque available for acceleration to the speed of the three wheeled omnidirectional vehicle [1]. This section is based on the results of [2].



(a) Bottom view [2] (b) Geometry [2]

Fig. 1. The omnidirectional vehicle

It is shown that the drive velocities are defined as linear functions of the velocity and the angular velocity of the

978-1-4244-4524-0/09/$25.00 ©2009 AACC

robot:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -\sin\theta & \cos\theta & L \\ -\sin(\frac{\pi}{3}-\theta) & -\cos(\frac{\pi}{3}-\theta) & L \\ -\sin(\frac{\pi}{3}+\theta) & -\cos(\frac{\pi}{3}+\theta) & L \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}, \quad (1)$$

where L is the distance of the drive units from the center of mass of the robot, v_i are the individual wheel velocities, θ is the angle of counterclockwise rotation (See Fig. 1(b)). New time and length scales are introduced

$$T = \frac{2m}{3\beta}, \quad \Psi = \frac{4\alpha m U_{max}}{9\beta^2}, \quad (2)$$

to normalize x, y, and t to the nondimensional variables

$$\bar{x} = \frac{x}{\Psi}, \quad \bar{y} = \frac{y}{\Psi}, \quad \bar{t} = \frac{t}{T}. \quad (3)$$

The constants α and β are determined by the motor character. U_{max} is the maximum value of the voltage applied to the motor, and m is the mass of the robot. Then the constraint of the robot (after dropping the bars) becomes

$$q_x^2(t) + q_y^2(t) \le 1, \quad (4)$$

(see [2] for the full derivation), where the two components of control $q_x(t)$ and $q_y(t)$ are

$$q_x(t) = \ddot{x} + \dot{x}, \quad (5)$$
$$q_y(t) = \ddot{y} + \dot{y}. \quad (6)$$

It has been shown that the time-optimal control strategy is achieved when

$$q_x^2(t) + q_y^2(t) = 1, \quad t \in [0, t_f], \quad (7)$$

where t_f is the final time. Kalmar-Nagy [2] solves the problem of time-optimal motion trajectory by ensuring the equality, but no obstacles are considered.

III. SAHRAEI'S ALGORITHM

Sahraei [1] proposed a trajectory generation algorithm based on the results of [2]. The algorithm is differentiated from Kalmar-Nagy's algorithm by two properties: real-time trajectory generation and obstacle avoidance. The first step is to construct the Voronoi diagram to find a path that avoids obstacles. Voronoi diagram is the partitioning of a plane with n points into n cells. The partitioning is made such that each cell includes one point and every point in a given cell is closer to the captured point. After constructing the Voronoi diagram, the start and target points, s and t are added to it with corresponding edges which connect these two points to their cell vertices. Then Dijkstra's shortest path algorithm is run. The resulting path is the shortest path whose edges are in the Voronoi diagram. Two Bézier curves are used to find a smooth path near the resulting path with regards to initial and final conditions.

A Bézier Curve of degree n is represented by $n+1$ control points $P_0, \ldots, P_n$:

$$P(\lambda) = \sum_{i=0}^{n} B_i^n(\lambda) P_i, \quad \lambda \in [0,1], \quad (8)$$

$$B_i^n(\lambda) = \binom{n}{i} (1-\lambda)^{n-i} \lambda^i, \quad i \in \{0, 1, \ldots, n\}. \quad (9)$$

The curve passes through P_0 and P_n and is tangent to $\overline{P_0 P_1}$ and $\overline{P_{n-1} P_n}$. Also, it lies within the convex hull of control points.

Let $p_0, p_1, \ldots, p_n$ denote the vertices of the shortest path and p_0, and p_n denote s and t, respectively. The first Bézier curve, $P_a(\lambda)$ for $\lambda \in [0,1]$, is constructed by p_0, q, r, and p_1, where control points q and r are introduced to satisfy slope of initial velocity constraint and continuity of curve and its slope in p_1. The second Bézier curve $P_b(\lambda)$ is constructed by $p_1, \ldots, p_n$. Following equations describe boundary conditions:

$$\frac{\dot{P}_a(0)}{|\dot{P}_a(0)|} = \frac{v_0}{|v_0|}, \quad \frac{\dot{P}_a(1)}{|\dot{P}_a(1)|} = \frac{\dot{P}_b(0)}{|\dot{P}_b(0)|}. \quad (10)$$

Fig. 2 shows an example of the paths.

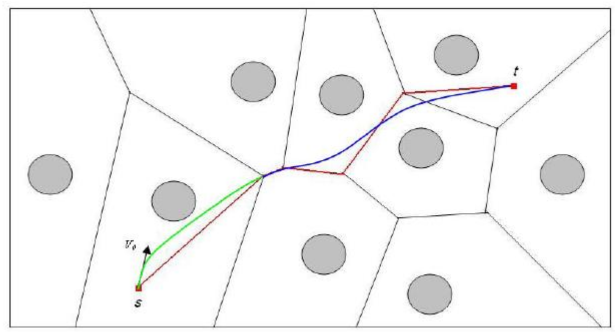


Fig. 2. A smooth path resulted from two Bezier curves. The first Bezier curve is illustrated in green and the second one is shown in blue [1].

Finally Sahraei assigned a velocity magnitude to each point on the generated curve $P(\lambda) = (X(\lambda), Y(\lambda))$. To implement this, the paper tried to find a function $\alpha : t \in \{0, h, 2h, \ldots\} \to \lambda \in [0,1]$ such that $X(\alpha(t))$ and $Y(\alpha(t))$ satisfy the following dynamic constraint

$$(\dot{X} + \ddot{X})^2 + (\dot{Y} + \ddot{Y})^2 \le 1, \quad (11)$$

where $h \in \mathbb{R}^+$ is the sample time interval. Note that the variables X, Y, and t for (11) are normalized values by (2). For the sake of optimality $\alpha(t)$ was calculated such that the left side of the inequality constraint (11) approaches 1. To find $\lambda_n \triangleq \alpha(nh)$ for all n's Sahraei used derivative approximations to define the function f:

$$\begin{aligned} f(\lambda) = &\left(\tfrac{X(\lambda) - X(\lambda_{n-2})}{2h} + \tfrac{X(\lambda) + X(\lambda_{n-2}) - 2X(\lambda_{n-1})}{h^2} \right)^2 \\ &+ \left(\tfrac{Y(\lambda) - Y(\lambda_{n-2})}{2h} + \tfrac{Y(\lambda) + Y(\lambda_{n-2}) - 2Y(\lambda_{n-1})}{h^2} \right)^2 - 1 + \varepsilon. \end{aligned} \quad (12)$$

λ_n is calculated by solving the equation

$$f(\lambda) = 0 \quad (13)$$

based on λ_{n-1} and λ_{n-2} by Newton's method. $\varepsilon \in \mathbb{R}^+$ in (12) guarantees that the result of Newton's method makes the left side of inequality (11) less than but close to 1. Since λ_n relies on two previous values of λ, at the first step λ_0 and

λ_1 should be calculated. It is straightforward that $\lambda_0 = 0$. To calculate λ_1, a hypothetical λ_{-1} is introduced to approximate the position of the robot before initial time, $X(\lambda_{-1})$ defined by

$$\frac{X(\lambda_0) - X(\lambda_{-1})}{h} \approx v_{x_0}, \tag{14}$$

and so forth for Y.

One of major drawbacks of this approach is the fact that there is no guarantee of existence of λ_n for all n's to satisfy (13) with small enough value of error, $|f(\lambda)|$. That is mainly because $X(\lambda)$ and $Y(\lambda)$ are constrained by the polynomial of $P(\lambda)$. This drawback will result in a violation of dynamic constraint of (4).

IV. Proposed Algorithm

This section proposes a new algorithm for obstacle avoiding real-time trajectory generation of omnidirectional vehicles. To describe this method, let $a_n = (a_{x_n}, a_{y_n})$, $v_n = (v_{x_n}, v_{y_n})$, $z_n = (x_n, y_n)$ denote the acceleration, velocity, and position of the vehicle, respectively, at sample time nh. All of the variables used in this section are nondimensional variables scaled by (2).

The new algorithm uses a Voronoi's diagram and a Bézier curve to generate the reference trajectory as Sahraei did. A problem of Sahraei's algorithm is that it did not do any calculation to ensure that the Bézier curves miss the obstacles. To resolve this problem, this paper ensures that the convex hull constructed by the control points of the Bézier curve do not contain any obstacles.

The algorithm also deals with velocities and accelerations of the robot in a discrete time system sampled at $t = \{0, h, 2h, \ldots\}$. However, it computes accelerations a_n that meet optimal condition (7) as opposed to that Sahraei computes positions z_n. The set of all such accelerations $\mathfrak{A}_n$ is represented as

$$\mathfrak{A}_n = \{(a_{x_n}, a_{y_n})| \quad (a_{x_n} + v_{x_n})^2 + (a_{y_n} + v_{y_n})^2 = 1\}. \tag{15}$$

If v_n is given, (15) can be rewritten as

$$\mathfrak{A}_n = \{(-v_{x_n} + \cos\theta_n, -v_{y_n} + \sin\theta_n)| \quad \theta_n \in [0, 2\pi)\}. \tag{16}$$

The beauty of (16) is that it guarantees satisfaction of (7) while Sahraei's algorithm only has the left side of the equation approaching 1. It also simplifies the value of the accelerations to one variable θ_n. Geometrically, $\mathfrak{A}_n$ is the circle that has center at $(-v_{x_n}, -v_{y_n})$ and radius of 1.

In this algorithm, the robot is assumed to follow the constant acceleration equations of motion with a_n for time interval $t \in [nh, (n+1)h)$. Once a_n is determined, the velocity and the position at next sample time are calculated by applying the constant acceleration equations with a_n:

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\mathbf{a}_n, \quad \mathbf{a}_n \in \mathfrak{A}_n, \tag{17}$$

$$\mathbf{z}_{n+1} = \mathbf{z}_n + h\mathbf{v}_n + \frac{h^2}{2}\mathbf{a}_n, \quad \mathbf{a}_n \in \mathfrak{A}_n. \tag{18}$$

In the problems that we consider, v_0 and z_0 are initially given. So a_0 is solely determined by selecting θ_0 in (16). Once a_0 is determined, v_1 and z_1 are obtained by applying (17) and (18) and using a_0, and so on. Thus we only need to find θ_n for all n's in order to fulfill motion planning of the robot, represented by a set of z_n. We also can generalize that v_n and z_n are give when we calculate θ_n at $t = nh$.

Equation (18) can be rewritten as the sum of two vectors:

$$\mathbf{z}_{n+1} = \mathbf{c}_{n+1} + \mathbf{r}_{n+1}, \tag{19}$$

where

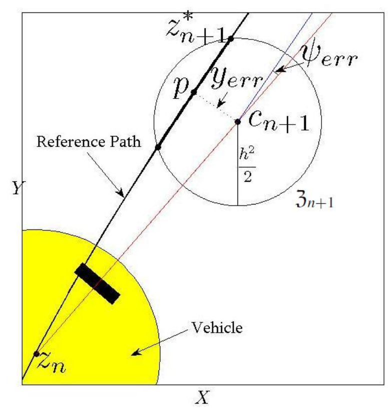
$$\mathbf{c}_{n+1} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} + (h - \frac{h^2}{2}) \begin{bmatrix} v_{x_n} \\ v_{y_n} \end{bmatrix}, \tag{20}$$

$$\mathbf{r}_{n+1} = \frac{h^2}{2} \begin{bmatrix} \cos\theta_n \\ \sin\theta_n \end{bmatrix}. \tag{21}$$

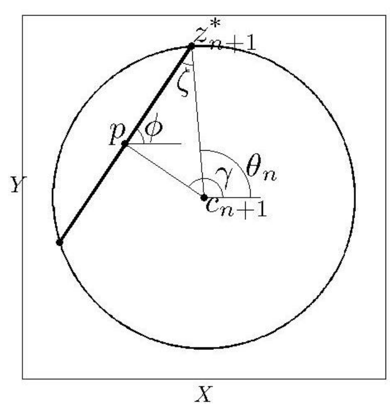
Since v_n and z_n are given at $t = nh$, $\mathbf{c}_{n+1}$ is known, while $\mathbf{r}_{n+1}$ depends on θ_n. So the set of all z_{n+1} corresponding to $\mathfrak{A}_n$, $\mathfrak{Z}_{n+1}$ is given by

$$\mathfrak{Z}_{n+1} = \{\mathbf{z}_{n+1}| \quad |\mathbf{z}_{n+1} - \mathbf{c}_{n+1}| = \tfrac{h^2}{2}\}. \tag{22}$$

Geometrically, $\mathfrak{Z}_{n+1}$ can be interpreted as the circle that has center at $\mathbf{c}_{n+1}$ and radius of $\frac{h^2}{2}$ as shown in Fig. 3(a). The intersections of $\mathfrak{Z}_{n+1}$ and the pre-generated Bézier curve $P(\lambda)$ satisfy the optimal condition (7). We will select z^*_{n+1}, the intersection further from current position since that provides a shorter travel time (See Fig. 3(a)).



(a) $\mathfrak{Z}_{n+1}$ on a reference trajectory.



(b) The enlarged $\mathfrak{Z}_{n+1}$.

Fig. 3. Geometry of $\mathfrak{Z}_{n+1}$ from z_n on a reference trajectory. p is defined by the point on the reference trajectory, which is the closest to c_{n+1}

In reality, the acceleration cannot be constant over the sample interval because the dynamic constraint (4) operates at all times. That is, even though $a_n \in \mathfrak{A}_n$ guarantees satisfaction of the optimal condition (7) at $t = nh$, the velocity changes due to a_n over the sample interval $t \in \big(nh, (n+1)h\big)$ and that will violate the dynamic constraint. To resolve this problem, we provide the closed-form analytical solution that obeys the dynamic constraints at all times as follows.

The constraint operating at $t \in \big[nh, (n+1)h\big)$ is

$$\mathbf{a}(t) = \frac{d\mathbf{v}}{dt}(t) = -\mathbf{v}(t) + \mathbf{A}, \tag{23}$$

$$\mathbf{v}(t) = \frac{d\mathbf{z}}{dt}(t), \tag{24}$$

where

$$\mathbf{A} = \begin{bmatrix}\cos\theta_n & \sin\theta_n\end{bmatrix}^T. \tag{25}$$

Assuming θ_n is constant over the sample interval $t \in \big[nh, (n+1)h\big)$, the velocity and position can be found in closed form

$$\mathbf{v}(t) = e^{-t}\mathbf{v}_n + (1-e^{-t})\mathbf{A} \tag{26}$$

$$\mathbf{z}(t) = \mathbf{z}_n + (1-e^{-t})\mathbf{v}_n + \big(t-(1-e^{-t})\big)\mathbf{A} \tag{27}$$

A second order Taylor series is used for small time intervals, h, yielding an expression for the position at the end of the sample interval

$$\mathbf{z}(nh+h) \quad = \mathbf{z}_n + (h - \tfrac{h^2}{2})\mathbf{v}_n + \big(h - (h - \tfrac{h^2}{2})\big)\mathbf{A}, \tag{28}$$

which is the exact same equation as the expression of $z_{n+1} \in \mathfrak{Z}_{n+1}$ in (22). That is, the exact closed-form solution is the same, to second order, as the assumption of constant acceleration.

Assuming that the reference path is planned such that $\mathfrak{Z}_{n+1}$ intersects the path for all n's, we only need to find θ_n corresponding to z^*_{n+1} for motion planning of the vehicle. However, noise in a real system and large parh curvatures may cause $\mathfrak{Z}_{n+1}$ to miss the path. For this case, another path following heuristic is required. The algorithm is divided into two modes depending on whether $\mathfrak{Z}_{n+1}$ intersects the reference path or not: *intersect-reference-trajectory (IR)* and *out-of-reference-trajectory (OR)*.

A. IR *mode*

In *IR* mode, θ_n corresponding to z^*_{n+1} is calculated in computationally efficient way. Firstly, we define the point on the Bézier curve, p which is the closest to c_{n+1} as shown in Fig. 3(a). To calculate p, we introduce the function f:

$$f(\lambda) = (X(\lambda) - c_{x_{n+1}})^2 + (Y(\lambda) - c_{y_{n+1}})^2, \quad \lambda \in [0,1]. \tag{29}$$

Let λ^p denote the parameter that minimizes $f(\lambda)$:

$$\lambda^p = \arg\min_{\lambda\in[0,1]} f(\lambda). \tag{30}$$

λ^p is computed by the steepest descent or Newton algorithms using the backtracking line search. Then p is given by

$$p = \big(X(\lambda^p), Y(\lambda^p)\big). \tag{31}$$

In Fig. 3(a), the portion of the Bézier curve inside of the circle $\mathfrak{Z}_{n+1}$ can be considered to be line segment of which the slope is the slope of tangent at p, given that time interval h is small enough. So we can approximate z^*_{n+1} as the intersection point between the circle and the straight line segment. Let ϕ denote the slope of the tangent line:

$$\phi = \tan^{-1}\big(\frac{\dot{Y}(\lambda^p)}{\dot{X}(\lambda^p)}\big). \tag{32}$$

Looking at the geometry of p, z^*_{n+1}, and c_{n+1} in Fig. 3(b), θ_n is given by

$$\theta_n = \phi + \zeta, \tag{33}$$

where ζ can be calculated by applying law of sines for $\triangle z^*_{n+1} p c_{n+1}$:

$$\zeta = \sin^{-1}\big(\frac{2|p - c_{n+1}|}{h^2}\sin(\pi - \gamma + \phi)\big). \tag{34}$$

and γ is the signed angle of direction of the vector $\overrightarrow{c_{n+1}p}$.

B. OR *mode*

To account for noise present in a real system, an efficient path following heuristic is presented. To describe this method, we introduce two terms: y_{err} and ψ_{err}. y_{err} is defined by the distance between c_{n+1} and p. ψ_{err} is defined by the angle difference from the current heading of the robot, ψ_n to the slope of tangent at p (See Fig. 3(a)).

The feedback control is designed such that the robot approaches the reference trajectory while making ψ_{err} small. So we use a PID steering control given by

$$\delta\psi = k_p y_{err} + k_d \psi_{err} + k_i \int y_{err} dt, \tag{35}$$

where $\delta\psi$ is the deflection of the heading of the robot:

$$\delta\psi = \psi_{n+1} - \psi_n. \tag{36}$$

The angle θ_n of the acceleration a_n that produces the desired $\delta\psi$ can be calculated in cost efficient way. Fig. 4 shows the relationship of $\delta\psi$ and θ_n in acceleration frame. From (16), $\mathfrak{A}_n$ is the circle that has center at $(-v_{x_n}, -v_{y_n})$ and radius of 1. Rewriting (17), the acceleration can be represented as the sum of two vectors:

$$\mathbf{a}_n = -\frac{1}{h}\mathbf{v}_n + \frac{1}{h}\mathbf{v}_{n+1} \tag{37}$$

The direction of the known vector $\frac{1}{h}\mathbf{v_n}$ is ψ_n. The other vector $\frac{1}{h}\mathbf{v_{n+1}}$ depends on the value of a_n. If we choose some point on the circle as a_n, then the vector from the tip of $-\frac{1}{h}\mathbf{v_n}$ to a_n will be $\frac{1}{h}\mathbf{v_{n+1}}$ whose angle is ψ_{n+1}. Since the desired $\delta\psi$ determines ψ_{n+1}, the intersections between the vector $\frac{1}{h}\mathbf{v}_{n+1}$ and the circle $\mathfrak{A}_n$ determine the acceleration at $t = nh$. When the number of intersections is two, the longer $-\frac{1}{h}\mathbf{v}_{n+1}$ is chosen so that travel time is smaller. $|\delta\psi|$ is bounded within $|\delta\psi|_{max}$ when the tip of the vector $-\frac{1}{h}\mathbf{v}_n$ is outside of the circle. $|\delta\psi|_{max}$ is defined by the $|\delta\psi|$ when the number of the intersections is one. So

$$|\delta\psi|_{max} = \begin{cases} \sin^{-1}\big(\frac{1}{(1/h-1)|v_n|}\big), & \text{if } (\frac{1}{h}-1)|v_n| > 1 \\ \pi, & \text{if } (\frac{1}{h}-1)|v_n| \le 1 \end{cases} \tag{38}$$

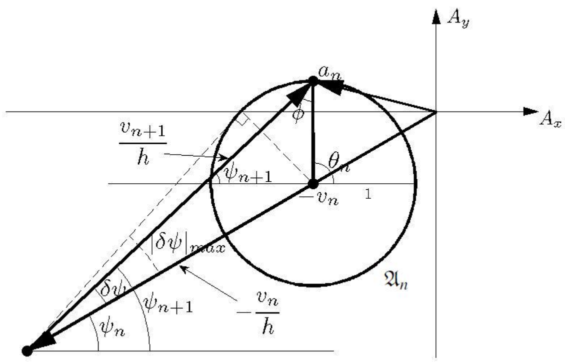


Fig. 4. Geometry of θ_n and a_n.

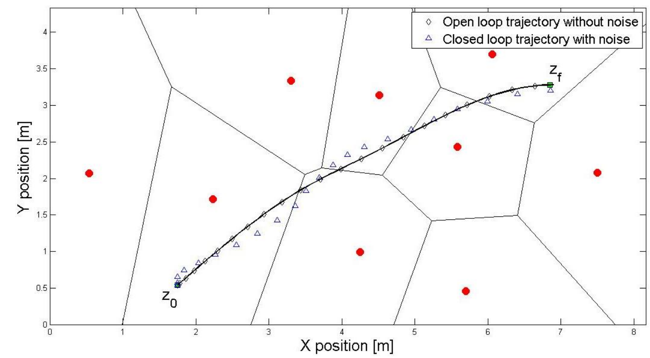


Fig. 5. The resulting trajectories by different algorithms over the reference trajectory (bold black curve).

In Fig. 4, θ_n can be represented as

$$\theta_n = \psi_{n+1} + \phi = \psi_n + \delta\psi + \phi, \tag{39}$$

where ϕ can be obtained by using law of sines:

$$\phi = \sin^{-1}\big((\frac{1}{h} - 1)|v_n|\sin(\delta\psi)\big). \tag{40}$$

Note that $\delta\psi$ is a signed angle and so ϕ is determined by $\delta\psi$. Equation (35) can be written as (41) by using (38), (39), and (40).

$$\theta_n = \psi_n + \phi + k_p y_{err} + k_d \psi_{err} + k_i \int y_{err} dt \tag{41}$$

subject to

$$\psi_n + \phi - |\delta\psi|_{max} \leq \theta_n \leq \psi_n + \phi + |\delta\psi|_{max} \tag{42}$$

In order to satisfy obstacle avoidance, the maximum y_{err} should be less than the minimum distance from obstacles to the pre-generated Bézier curve. For computational efficiency, the minimum distance is measured as minimum distance from obstacles to control points of the Bézier curve.

V. Numerical Simulations

Simulations provided in this section demonstrate improvement of trajectory generation and control by the proposed algorithm in terms of travel time, satisfaction of the dynamic constraint, and smooth motion control compared to Sahraei's algorithm. Also, they show robustness of the proposed algorithm. Fig. 5 shows the course used for the simulation. Red circles indicate obstacles.

The initial and final conditions are given by:

$$z_0 = (1.75, 0.54)[m], \tag{43}$$

$$v_0 = (0,0)[m/s], \tag{44}$$

$$z_f = (6.85, 3.28)[m]. \tag{45}$$

The sample time interval h is given by

$$h = 0.0033[s]. \tag{46}$$

Characteristic variables are given by

$$\alpha = 1[N/V],\ \beta = 1[kg/s],\ m = 1[kg],\ Umax = 3[v]. \tag{47}$$

The reference trajectory is constructed by a Bézier curve for which the control points are

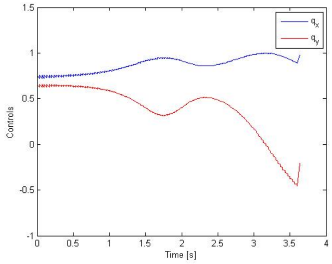
$$\begin{aligned} p_0 &= (1.75, 0.54), p_1 = (3.49, 2.05), p_2 = (3.72, 2.14), \\ p_3 &= (4.55, 2.04), p_4 = (5.35, 3.24), p_5 = (6.85, 3.28), \end{aligned} \tag{48}$$

and illustrated as bold black curve in Fig. 5. The simulation of Sahraei's algorithm has been done with the same parameters above and $\varepsilon = 0.01$.

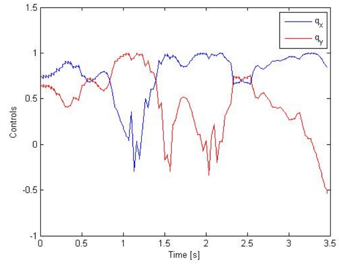
In Fig. 5, two kinds of trajectories are generated depending on addition of noise. The open loop trajectory without noise is generated by applying two different algorithms: the proposed algorithm and Sahraei's algorithm. The closed loop trajectory with noise is generated by the proposed algorithm. The reference trajectory is generated smooth enough that $\mathfrak{Z}_{n+1}$ contains a portion of the trajectory for all n's. So, in simulation of the proposed algorithm, only *IR* logic is used for the open loop trajectory while combination of *IR* and *OR* is used for the closed loop trajectory. The resulting closed loop trajectory shows the robustness to noise in Fig. 5. The noise was modeled as white noise with magnitude of $0.05m$ and added to actual position. The simulation results are listed in table I. The resulting final times t_f by the proposed algorithm are substantially shorter than the one by Sahraei's algorithm. In the table, the cross track error c_{err} is defined as the distance of the line normal to the path and passing through the vehicle position. Sahraei's algorithm leads to violation of the dynamic constraint (4). We can see that $q_x^2 + q_y^2$ exceeds boundary condition 1 in Fig. 6(f). On the other hand, $q_x^2 + q_y^2$ by the proposed algorithm is 1 at every sample time interval as shown in Fig. 6(d) and 6(e). In addition, the proposed algorithm generates smoother controls q_x and q_y and velocities v_x and v_y than Sahraei's algorithm as shown in Fig. 6(a), 6(b), 6(g) and 6(h).

VI. Conclusions

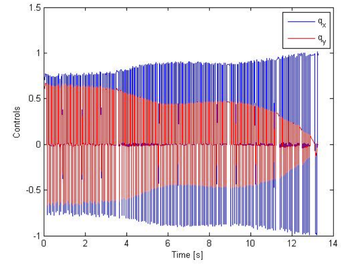
This paper proposes a collision-free real-time motion planning algorithm for an omnidirectional mobile robot. It has been shown that planned motion of the robot is a computationally effective way to satisfy obstacle avoidance as well as robustness, and the proposed algorithm leads to short travel times. Numerical simulations demonstrate the



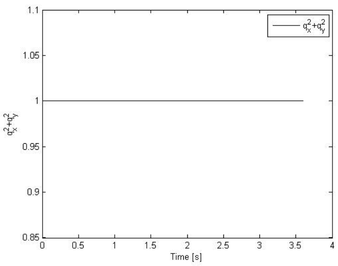
(a) q_x and q_y by IR.



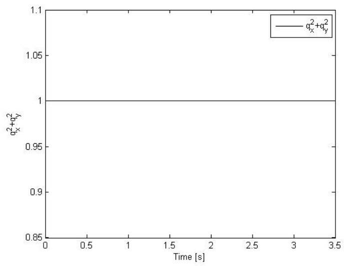
(b) q_x and q_y by IR and OR (noise).



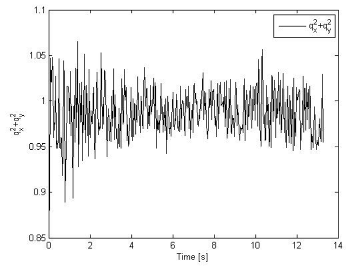
(c) q_x and q_y by Sahraei.



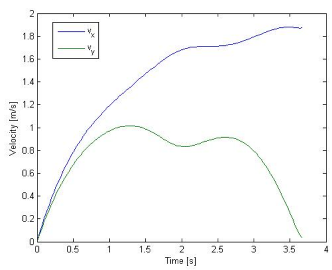
(d) $q_x^2+q_y^2$ by IR.



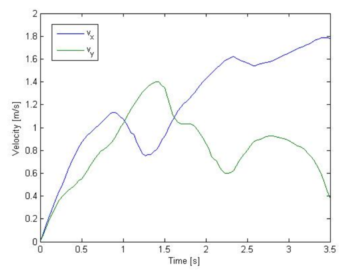
(e) $q_x^2+q_y^2$ by IR and OR (noise).



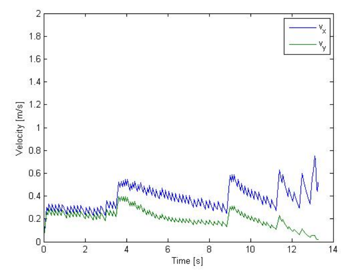
(f) $q_x^2+q_y^2$ by Sahraei.



(g) v_x and v_y by IR.



(h) v_x and v_y by IR and OR (noise).



(i) v_x and v_y by Sahraei.

Fig. 6. The results obtained by the proposed algorithm and Sahraei's algorithm.

TABLE I
RESULTS OF THE SIMULATION

Methods	t_f $[s]$	Violation of (4) $[\%]$	$\int_0^{t_f} \lvert c_{err} \rvert dt$
IR without noise	3.6667	0	0
IR and OR with noise	3.4667	0	0.0018
Sahraei without noise	13.2667	31.91	0

improvement of the motion planning compared to Sahraei's algorithm.

REFERENCES

[1] A. Sahraei, M. T. Manzuri, M. R. Razvan, M. Tajfard and S. Khoshbakht, "Real-Time Trajectory Generation for Mobile Robots," *The 10th Congress of the Italian Association for Artificial Intelligence (AIIA 2007)* September 10-13, 2007 .

[2] Kalmar-Nagy T., D'Andrea R., Ganguly P., "Near-optimal dynamic trajectory generation and control of an omnidirectional vehicle," *Robotics and Autonomous Systems, Volume 46, Number 1,* 31 January 2004 , pp. 47-64(18), Elsevier.

[3] Balkcom, D. and Mason, M. Time Optimal, "Trajectories for Bounded Velocity Differential Drive Robots," *IEEE International Conference on Robotics and Automation (ICRA 00),* p. 2499 - 2504, 2000.

[4] Jung, M., Shim, H., Kim, H. and Kim, J., "The Miniature Omni-directional Mobile Robot OmniKity-I (OK-I)," *International Conference on Robotics and Automation,* p. 2686-2691, 1999.

[5] Moore, K. L. and Flann, N. S., "Hierarchial Task Decomposition Approach to Path Planning and Control for an Omni-Directional Autonomous Mobile Robot," *International Symposium on Intelligent Control/Intelligent Systems and Semiotics,* p. 302-307, 1999.

[6] Huang, W. H., Fajen, B. R., Fink, J. R., Warren, W. H., "Visual navigation and obstacle avoidance using a steering potential function," *Robotics and Autonomous Systems,* vol. 54, Issue 4, p. 288-299, 28 April 2006.

[7] Hamner, B., Singh, S., Scherer, Se., "Learning Obstacle Avoidance Parameters from Operator Behavior," *Special Issue on Machine Learning Based Robotics in Unstructured Environments, Journal of Field Robotics,* vol. 23, 11/12, p. 1037-1058, December 2006.

[8] Hwang, C., Chang, L. "Trajectory Tracking and Obstacle Avoidance of Car-Like Mobile Robots in an Intelligent Space Using Mixed H_2/H_∞ Decentralized Control," *Mechatronics, IEEE/ASME Transactions on,* vol. 12, Issue 3, p. 345-352, June 2007