Collision Free Real-Time Motion Planning for 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 meets obstacle avoidance. Then the algorithm solves the problem of motion planning for the robot to track the path in short travel time while satisfying dynamic constraints and robustness to noise. Accelerations of the robot are computed and refined 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 constraint and smooth motion control compared to a 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 [4] developed the time optimal trajectories for the bounded velocity model of differential drive robots. Jung [5] and Moore [6] dealt with omnidirectional vehicles; the control strategy employed by these papers consists of building a geometric path and tracking path by using feedback control. Huang [7] 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 over the robot heading. The potential field controls the angular acceleration of the robot, steering it towards the goal and away from obstacles. Hamner [8] 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 [9] 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 pose of the robot and the position of the obstacle. Based on the authority of these cameras, a reference command for the proposed controller of the robot is planned.

This paper especially focuses on two papers: Kalmar-Nagy [2] and Sahraei [1]. Kalmar-Nagy [2] has proposed minimum time trajectory generation algorithm for omnidirectional vehicles, that meets dynamic constraints, but no obstacles are considered. 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. This paper shows that Sahraei's algorithm is problematic. To resolve the problems, 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.

This paper is organized as follows. Section III describes dynamic constraints of the robots based on the result of [2]. Section IV presents the reference path generation algorithm of [1] with minor modification. Then the new motion planning algorithm is proposed in Section V. Finally, a numerical simulation is presented in Section VI.

J. Choi is a Ph.D. candidate at Computer Engineering Department in University of California, Santa Cruz, 95064, USA. jwchoi@soe.ucsc.edu

R. Curry is a Research Associate at Computer Engineering Department in University of California, Santa Cruz, 95064, USA. rcurry@ucsc.edu

G. Elkaim is an assistant professor at Computer Engineering Department in University of California, Santa Cruz, 95064, USA. elkaim@soe.ucsc.edu

II. BACKGROUND

A. Bézier Curve

A Bézier Curve of degree n can be represented as

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

$$B_i^n(\lambda) = \binom{n}{i} (1-\lambda)^{n-i} \lambda^i, \quad i \in \{0,1,\ldots,n\} \tag{2}$$

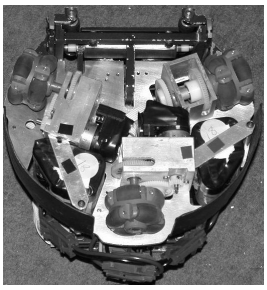
Bézier Curves have useful properties for path planning:

- They always pass through P_0 and P_n.
- They are always tangent to the lines connecting $P_0 \rightarrow P_1$ and $P_n \rightarrow P_{n-1}$ at P_0 and P_n respectively.
- They always lie within the convex hull of their control points.

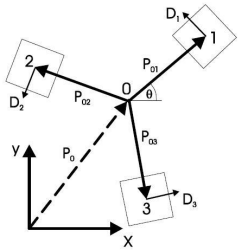
III. DYNAMIC CONSTRAINTS OF THE OMNIDIRECTIONAL VEHICLE

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].

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



(a) Bottom view of the omnidirectional vehicle [2].



(b) Geometry of the omnidirectional vehicle [2].

Fig. 1. Omnidirectional vehicle.

robot:

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

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

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

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

$$\bar{x} = \frac{x}{\Psi}, \quad \bar{y} = \frac{y}{\Psi}, \quad \bar{t} = \frac{t}{T}. \quad (5)$$

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

$$q_x^2(t) + q_y^2(t) \leq 1, \quad (6)$$

where the two components of control $q_x(t)$ and $q_y(t)$ are

$$q_x(t) = \ddot{x}(t) + \dot{x}(t), \quad q_y(t) = \ddot{y}(t) + \dot{y}(t). \quad (7)$$

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

$$q_x^2(t) + q_y^2(t) = 1, \quad t \in [0, t_f], \quad (8)$$

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.

IV. Path Planning

This section presents path planning algorithm based on the result of [1]. The first step is to construct the Voronoi diagram against obstacles. Start and target points, s and t are added to this graph with corresponding edges which connect these two points to their cells vertices. Dijkstra's shortest path algorithm is run resulting in the shortest path whose edges are in the Voronoi diagram. Then two Bézier curves are used to find a smooth path near the resulting path with regards to initial and final conditions. Let $p_0, p_1, \ldots, p_n$ are vertices of the shortest path and p_0, and p_n are 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:

$$\frac{\dot{P}_a(0)}{|\dot{P}_a(0)|} = \frac{v_0}{|v_0|}, \quad \frac{\dot{P}_a(1)}{|\dot{P}_a(1)|} = \frac{\dot{P}_b(0)}{|\dot{P}_b(0)|}. \quad (9)$$

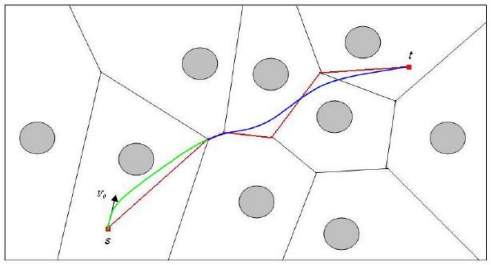


Fig. 2. A smooth path resulted from two Bézier curves. The first Bézier curve is illustrated in green and the second one is shown in blue [1].

A problem of this method in [1] is that it did not do any calculation to ensure that the Bézier curve misses the obstacles. To resolve this problem, this paper ensures that the convex hull constructed by the control points of the Bézier curve does not contain any obstacles. If it does, the control points are positioned so that the intersections disappear.

V. Path Following

This section proposes a new algorithm for real-time trajectory generation of omnidirectional vehicles. The algorithm deals with velocities and accelerations of the robot in discrete time system sampled by $t = nh$ for $n \in \{0,1,2,\ldots\}$ and $h \in \mathbb{R}^+$. We make use of the following notation. $a_n = (a_{x_n}, a_{y_n}) \in \mathbb{R}^2$, $v_n = (v_{x_n}, v_{y_n}) \in \mathbb{R}^2$, $z_n = (x_n, y_n) \in \mathbb{R}^2$, and $\psi_n \in [0, 2\pi)$ denote the acceleration, velocity, position, and heading of the vehicle, respectively, at sample time nh. Also, $P(\lambda) = (X(\lambda), Y(\lambda)) \in \mathbb{R}^2$ denotes the coordinates of the Bézier curve used as a reference path. Suppose that all of the variables used in this section are nondimensional variables scaled by (4).

The motion of the robot is planned such that it tracks the pre-generated reference path while satisfying optimal condition (8) at each sample time, which can be represented as

$$(a_{x_n} + v_{x_n})^2 + (a_{y_n} + v_{y_n})^2 = 1. \quad (10)$$

In this paper, 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, thus, the velocity and the position at next sample time is given by

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\mathbf{a}_n, \quad (11)$$

$$\mathbf{z}_{n+1} = \mathbf{z}_n + h\mathbf{v}_n + \frac{h^2}{2}\mathbf{a}_n. \quad (12)$$

In the problems that we consider, v_0 and z_0 are initially given. So unknown is a_0 in (10) for $n = 0$. Once a_0 is determined,

v_1 and z_1 are obtained by applying (11) and (12) and using a_0, and so on. Thus we only need to find a_n for all n's in order to fulfill motion planning of the robot. We also can generalize that v_n and z_n are known when we calculate a_n in (10). Let $\mathfrak{A}_n$ denote the set of all accelerations that satisfy (8):

$$\mathfrak{A}_n = \{(a_{x_n}, a_{y_n}) | \quad (a_{x_n} + v_{x_n})^2 + (a_{y_n} + v_{y_n})^2 = 1\}, \quad (13)$$

or

$$\mathfrak{A}_n = \{(-v_{x_n} + \cos\theta_n, -v_{y_n} + \sin\theta_n) | \quad \theta_n \in [0, 2\pi)\}. \quad (14)$$

Geometrically, $\mathfrak{A}_n$ is the circle that has center at $(-v_{x_n}, -v_{y_n})$ and radius of 1. The set of all z_{n+1} corresponding to $\mathfrak{A}_n$, $\mathfrak{Z}_{n+1}$ is obtained by using (12) and (14):

$$\begin{aligned} \mathfrak{Z}_{n+1} = \{&(c_x + \tfrac{h^2}{2}\cos\theta_n, c_y + \tfrac{h^2}{2}\sin\theta_n) | \\ &c_x = x_n + (h - \tfrac{h^2}{2})v_{x_n},\ c_y = y_n + (h - \tfrac{h^2}{2})v_{y_n}\}. \end{aligned} \quad (15)$$

Geometrically, $\mathfrak{Z}_{n+1}$ can be interpreted as the circle that has center at $c = (c_x, c_y)$ and radius of $\frac{h^2}{2}$.

One can concern that the assumption of constant acceleration over the sample interval might cause the violation of the dynamic constraint (6). That is, even though $a_n \in \mathfrak{A}_n$ guarantees satisfaction of the optimal condition (8) at $t = nh$, deviation of velocity due to a_n over the sample interval $t \in \big(nh, (n+1)h\big)$ might violate the dynamic constraint. To resolve this problem, we provide the closed-form analytical solution as following. The constraint operating at $t \in \big[nh, (n+1)h\big)$ is

$$\mathbf{a}(t) = \frac{d\mathbf{v}}{dt}(t) = -\mathbf{v}(t) + \mathbf{A}, \quad (16)$$

$$\mathbf{v}(t) = \frac{d\mathbf{z}}{dt}(t), \quad (17)$$

where

$$\mathbf{A} = \begin{bmatrix} \cos\theta_n & \sin\theta_n \end{bmatrix}^T. \quad (18)$$

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

$$\mathbf{v}(t) = e^{-t}\mathbf{v}_n + (1 - e^{-t})\mathbf{A} \quad (19)$$

$$\mathbf{z}(t) = \mathbf{z}_n + (1 - e^{-t})\mathbf{v}_n + \big(t - (1 - e^{-t})\big)\mathbf{A} \quad (20)$$

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

$$\begin{aligned} \mathbf{z}(nh + h) &= \mathbf{z}_n + (h - \tfrac{h^2}{2})\mathbf{v}_n + \big(h - (h - \tfrac{h^2}{2})\big)\mathbf{A} \\ &= \begin{bmatrix} x_n \\ y_n \end{bmatrix} + (h - \tfrac{h^2}{2}) \begin{bmatrix} v_{x_n} \\ v_{y_n} \end{bmatrix} + \tfrac{h^2}{2} \begin{bmatrix} \cos\theta_n \\ \sin\theta_n \end{bmatrix}, \end{aligned} \quad (21)$$

which is the exact same equation as the expression of $z_{n+1} \in \mathfrak{Z}_{n+1}$ in (15). That is, the closed-form solution is the same as if we make the assumption of constant acceleration, with the constraint satisfied at $t = nh$.

The algorithm is divided into two modes depending on if $\mathfrak{Z}_{n+1}$ intersect the reference path or not: *intersect-reference-trajectory (IR)* and *out-of-reference-trajectory (OR)*.

A. IR *mode*

In most case, the robot is in *IR* mode given that the deviation of slope of the reference path $P(\lambda)$ is small enough for $t \in [nh, (n+1)h)$. In the mode, z_{n+1} can be calculated in computationally efficient way in following subsections.

Firstly, we define the point on the Bézier curve, p which is the closest to c as shown in Fig. 3(a). The distance of $\overline{cp}$ is denoted as c_{err}. Let the slope of the tangent at z_n and at p denote ϕ_n and ϕ^p, respectively. In *IR* mode, in most case, $\mathfrak{Z}_{n+1}$ have two intersections with $P(\lambda)$, which satisfy the optimal condition (8). We will select the one further down from current position as z_{n+1} provides shorter travel time (See Fig. 3(a)). This paper proposes two approaches to calculate the point.

1) Coordinate Transformation and Newton's Method Approach: The point can be considered as the intercept of $P(\lambda)$ and top-semi-circle of $\mathfrak{Z}_{n+1}$ with respect to the line normal to the tangent at z_n and passing c. Having new coordinate system $X_c - Y_c$ such that the origin is at c and Y axis is collinear to ϕ_n, the semi-circle can be formulated as

$$y = \sqrt{\frac{h^4}{4} - x^2}, \quad (22)$$

where x and y are with respect to X_c and Y_c. To calculate the λ corresponding to z_{n+1}, we introduce a function f:

$$f(\lambda) = Y(\lambda) - \sqrt{\frac{h^4}{4} - X^2(\lambda)}, \quad (23)$$

where the coordinates of the reference path, X and Y are also with respect to X_c and Y_c. It is only needed to convert control points of P with respect to the new coordinate system to obtain them. Then z_{n+1} is given by

$$z_{n+1} = \big(X(\lambda^*), Y(\lambda^*)\big) \quad (24)$$

where λ^* is the solution to the equation $f(\lambda) = 0$, which is calculated by applying Newton's method. The acceleration a_n that produces the z_{n+1} is given by applying constant acceleration equations of motion:

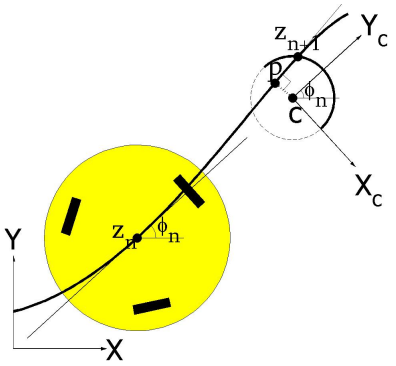
$$\mathbf{a}_n = \frac{2(\mathbf{z}_{n+1} - \mathbf{z}_n - \mathbf{v}_n h)}{h^2} \quad (25)$$

This method holds under the assumption that one of the two intercepts has positive y value and the other has negative with respect to $X_c - Y_c$. Let Ω_z denote the portion of P inside of the circle $\mathfrak{Z}_{n+1}$. Ω_z can be considered to be line segment of which slope is equal to ϕ^p, given that time interval h is small enough. So we can approximate p as the mid-point of two intersect points between the circle and the tangent line at p as shown in Fig. 3(b). It is straightforward that z_{n+1} is above and the other is below X_c when Assumption 1 holds.

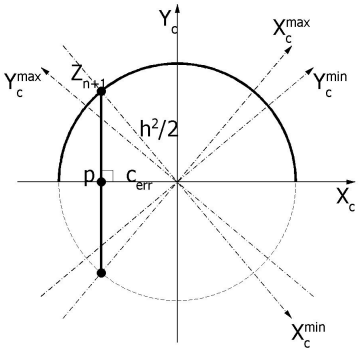
Assumption 1: $|\phi^p - \phi_n| \leq \cos^{-1}(\frac{2c_{err}}{h^2})$.

2) Quadratic Approach: This section presents another approach to find the Bézier parameter λ, where the reference path intersects the circle of achievable positions. That is, we are to find the larger root of the equation

$$f(\lambda) = \frac{h^2}{2}, \quad (26)$$



(a) z_{n+1} is calculated as the intercept of semi-circle of $\mathfrak{Z}_{n+1}$ and a reference path.



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

Fig. 3. Geometry of $\mathfrak{Z}_{n+1}$ from z_n on a reference trajectory.

where

$$f(\lambda) = (X(\lambda) - c_x)^2 + (Y(\lambda) - c_y)^2. \tag{27}$$

The algorithm consists of two components. The first component is to estimate a scale factor $\frac{ds}{d\lambda}$, where s is path length. This scale factor is used to convert an estimate of the next change in distance into a change in λ. The derivative of (27) is given by

$$\frac{df}{d\lambda} = 2(X(\lambda) - c_x)\frac{dX}{ds}\frac{ds}{d\lambda} + 2(Y(\lambda) - c_y)\frac{dY}{ds}\frac{ds}{d\lambda}, \tag{28}$$

where

$$\frac{dX}{ds} \cong \cos\psi, \quad \frac{dY}{ds} \cong \sin\psi, \quad \frac{ds}{d\lambda} \approx const = \mu. \tag{29}$$

Then

$$\frac{df}{d\lambda} = 2(X(\lambda) - c_x)\mu\cos\psi + 2(Y(\lambda) - c_y)\mu\sin\psi, \tag{30}$$

$$\frac{d^2f}{d\lambda^2} = 2\frac{dX}{ds}\frac{ds}{d\lambda}\mu\cos\psi + 2\frac{dX}{ds}\frac{ds}{d\lambda}\mu\sin\psi = 2\mu^2 \tag{31}$$

Initial guess at the next λ is computed by

$$\hat{\lambda}_{n+1} = \lambda_n + \frac{\|c\| + \frac{h^2}{2}}{\mu_n}, \tag{32}$$

where μ_n is the current estimate of $\frac{ds}{d\lambda}$.

The second component is to refine the estimate of λ assuming a quadratic form of the f. The desired λ is the larger root of the following:

$$\begin{aligned}\frac{h^2}{2} = \; & f(\hat{\lambda}_{n+1}) + \frac{df}{d\lambda}(\hat{\lambda}_{n+1})(\lambda - \hat{\lambda}_{n+1}) + \\ & \frac{1}{2}\frac{d^2f}{d\lambda^2}(\hat{\lambda}_{n+1})(\lambda - \hat{\lambda}_{n+1})^2,\end{aligned} \tag{33}$$

where right side is a Taylor series expansion of f about $\hat{\lambda}_{n+1}$. Solve this quadratic for the new λ and repeat the computation of the derivatives for a new quadratic if necessary. When this iteration is finished it yields λ_{n+1} at the intersection point z_{n+1}. Update the value of μ_{n+1} by the following equation:

$$\mu_{n+1} = (1-\alpha)\mu_n + \alpha\frac{\lambda_{n+1} - \lambda_n}{\|z_{n+1} - z_n\|}, \tag{34}$$

where α is smoothing constant needed only if the raw $\frac{\lambda_{n+1}-\lambda_n}{\|z_{n+1}-z_n\|}$ values are noisy. μ_0 is found by taking a small step on λ, say $\overline{\lambda} = 0.0001$, computing the corresponding $z(\overline{\lambda})$, then

$$\mu_0 = \frac{\overline{\lambda}}{\|z(\overline{\lambda}) - z_0\|}. \tag{35}$$

B. OR *mode*

Big curvature of a reference path or noise in a real system may have the robot miss the path at a certain point. To account for this situation, an efficient path following heuristic is presented. To describe this method, we introduce ψ_{err} defined by the angle difference from ψ_n to ϕ^p.

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

$$\delta\psi = k_p c_{err} + k_d\psi_{err} + k_i\int c_{err}dt, \tag{36}$$

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

$$\delta\psi = \psi_{n+1} - \psi_n. \tag{37}$$

c_{err} relies on p. The λ corresponding to p is computed by minimizing (27) by applying Newton's method. a_n that produces the desired $\delta\psi$ can be calculated in cost efficient way as following. Recall that $a_n \in \mathfrak{A}_n$ can be represented as:

$$a_n = (-v_{x_n} + \cos\theta_n, -v_{y_n} + \sin\theta_n). \tag{38}$$

Fig. 4 shows the relationship of $\delta\psi$ and θ_n in acceleration frame. Rewriting (11), the acceleration can be represented as the sum of two vectors:

$$\mathbf{a}_n = -\frac{1}{h}\mathbf{v}_n + \frac{1}{h}\mathbf{v}_{n+1} \tag{39}$$

Since v_n is known, $-\frac{1}{h}\mathbf{v_n}$ is deterministic. It is straightforward that the direction of the 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 $\frac{1}{h}\mathbf{v_{n+1}}$. Similarly, direction of the vector 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 $\mathfrak{A}_n^p$ are perspective accelerations at $t = nh$. When the number of the points are two, the one further from tip of the vector $-\frac{1}{h}\mathbf{v}_n$ is chosen. This is because it makes $|v_{n+1}|$ bigger than

the other so that travel time gets 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

$$|\delta\psi|_{max} = \begin{cases} \sin^{-1}\left(\frac{1}{(1/h-1)|v_n|}\right), & \text{if } (\frac{1}{h}-1)|v_n| > 1 \\ \pi, & \text{if } (\frac{1}{h}-1)|v_n| \leq 1 \end{cases} \quad (40)$$

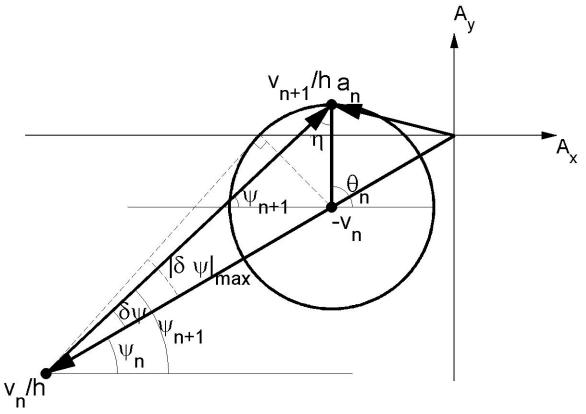


Fig. 4. Geometry of θ_n and a_n.

In Fig. 4, θ_n can be represented as

$$\theta_n = \psi_{n+1} + \eta = \psi_n + \delta\psi + \eta, \quad (41)$$

where η can be obtained by using law of sines:

$$\eta = \sin^{-1}\left((\frac{1}{h}-1)|v_n|\sin(\delta\psi)\right). \quad (42)$$

Note that $\delta\psi$ is signed angle and so is η determined by $\delta\psi$. Equation (36) can be written as (43) by using (40), (41), and (42).

$$\theta_n = \psi_n + \eta + k_p c_{err} + k_d \psi_{err} + k_i \int c_{err} dt \quad (43)$$

subject to

$$\psi_n + \eta - |\delta\psi|_{max} \leq \theta_n \leq \psi_n + \eta + |\delta\psi|_{max} \quad (44)$$

In order to meet obstacle avoidance, maximum c_{err} should be less than 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.

VI. 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 dynamic constraint, smooth motion control, and robustness to noise compared to Sahraei's algorithm. Fig. 5 shows the course used for the simulation. Red circles indicate obstacles. The initial and final conditions and the sample time interval are given by

$$\begin{aligned} z_0 &= (1.75, 0.54)[m], \quad z_f = (6.85, 3.28)[m], \\ v_0 &= (0,0)[m/s], \quad h = 0.0033[s]. \end{aligned} \quad (45)$$

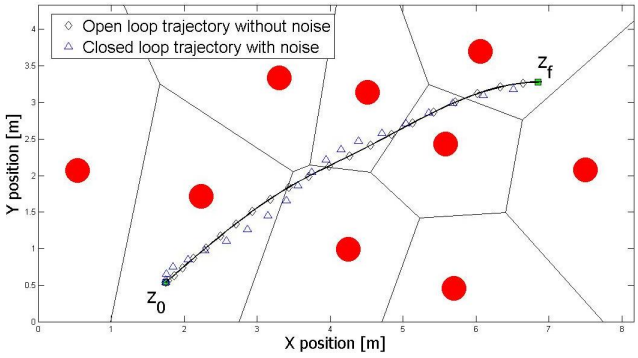


Fig. 5. The resulting trajectories by different algorithms over the reference trajectory (bold black curve).

Characteristic variables are given by

$$\alpha = 1[N/V], \ \beta = 1[kg/s], \ m = 1[kg], \ Umax = 3[v]. \quad (46)$$

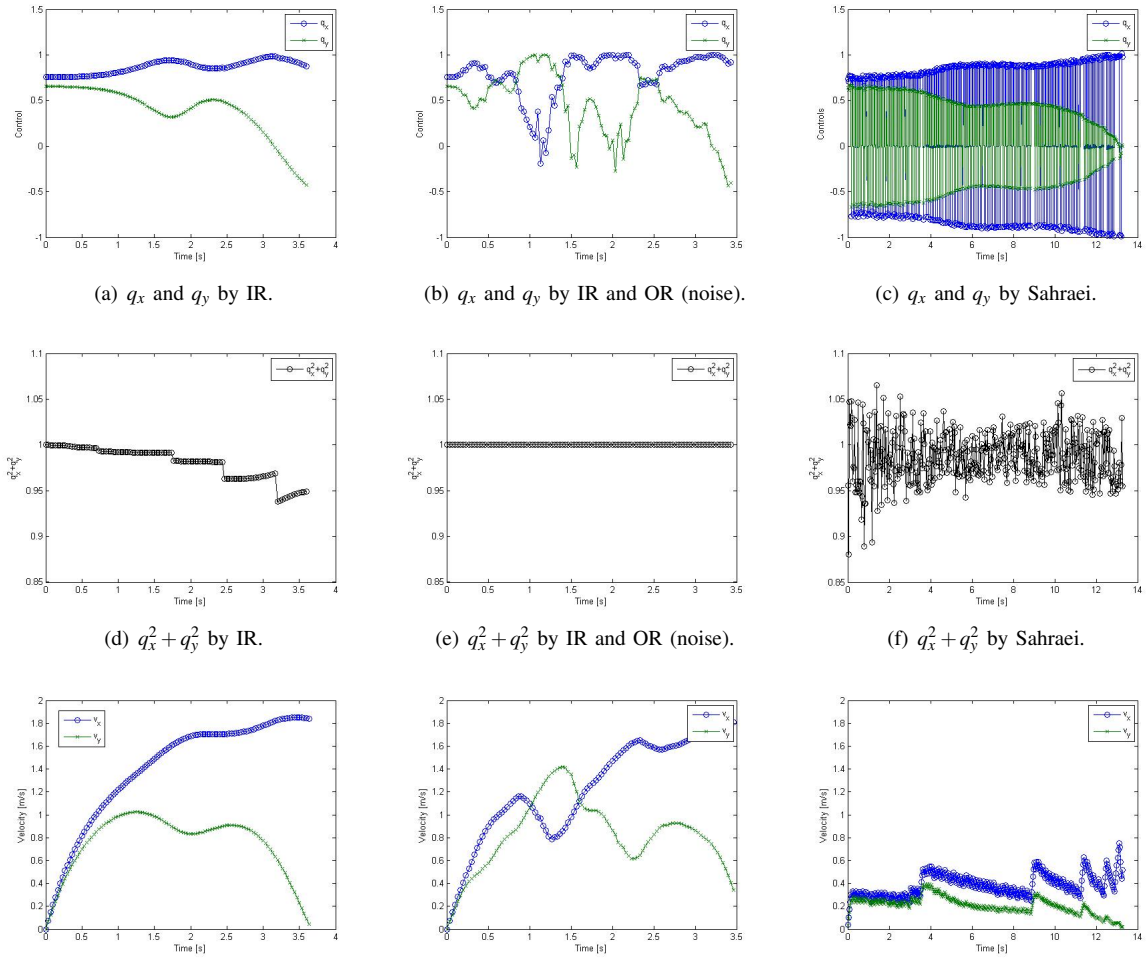
The reference path is illustrated as bold black curve in Fig. 5. The simulation of Sahraei's algorithm is done with the same parameters above. In Fig. 5, two kinds of trajectories are generated depending on addition of noise. The open loop trajectory without noise is generated by applying two different 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. PID gains used in *OR* mode are given by

$$k_p = 20, \quad , k_d = 1, \quad k_i = 20. \quad (47)$$

Since deviation of the slope of the reference path is small, cross track error should be more penalized than heading error. So bigger proportional and integral gains are used compared to differential gain. The resulting closed loop trajectory shows the robustness to noise which is modeled as white noise with intensity of $5cm$ from a uniform distribution for positions. 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. Sahraei's algorithm leads to violation of the dynamic constraint (6). 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 the end of every sample time interval as shown in Fig. 6(d). 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), and 6(g).

VII. 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



(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 (6) $[\%]$	Cross Track Error $[cm]$
IR	3.6333	0	0
IR and OR (with noise)	3.6333	0	54.55
Sahraei	13.2667	31.91	0

short travel times. Numerical simulations demonstrate the 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] Athans, M. and Falb, P. L., "Optimal Control: An introduction to the theory and its applications," McGraw-Hill, New York, 1966.

[4] 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.

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

[6] 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.

[7] 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.

[8] 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.

[9] 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