# 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 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 $H_2/H_\infty$ 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. 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.

This 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.

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

## II. BACKGROUND

### A. Bézier Curve

Bézier Curves were invented in 1962 by the French engineer Pierre Bézier for designing automobile bodies. Today Bézier Curves are widely used in computer graphics and animation. 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 $P_0 \to P_1$ and $P_n \to P_{n-1}$ at $P_0$ and $P_n$ respectively.
- They always lie within the convex hull of their control points.

J. Choi is a Ph.D. student at Autonomous Systems Lab, Computer Engineering Department in University of California, Santa Cruz, 95064, USA. He received BSE degree in Electrical Engineering from Yonsei University in South Korea. jwchoi@soe.ucsc.edu

R. Curry is an president of Applied Aeronautical Systems Inc. He received an AB degree in Physics from Middlebury College, and Ph.D. degrees in Aeronautics and Astronautics from the Massachusetts Institute of Technology (MIT). rcurry@ucsc.edu

G. Elkaim is an assistant professor at Autonomous Systems Lab, Computer Engineering Department in University of California, Santa Cruz, 95064, USA. He received BSE degree in Mechanical Aerospace Engineering from Princeton University, and MSE and Ph.D. degree in Aeronautics and Astronautics from Stanford. elkaim@soe.ucsc.edu

## III. Dynamic Constraints of the Omnidirectional Vehicle

Fig. 1 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].

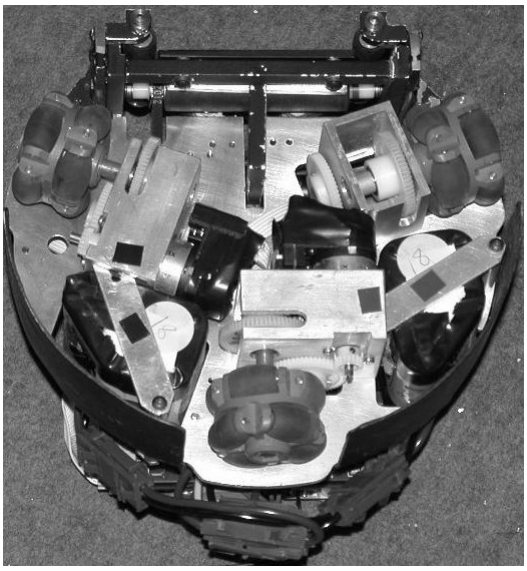


Fig. 1. Bottom view of the omnidirectional vehicle [2]

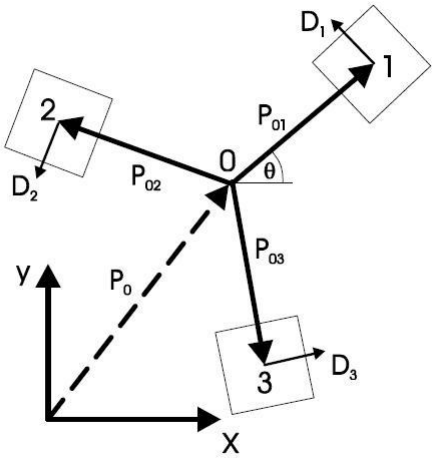



Fig. 2. Geometry of the omnidirectional vehicle [2]

It is shown that the drive velocities are defined as linear functions of the velocity and the angular velocity of the 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, $\theta$ is the angle of counterclockwise rotation (See Fig. 2). 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 $\alpha$ and $\beta$ 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

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

$$q_y(t) = \ddot{y} + \dot{y}. \quad (8)$$

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

$$q_x^2(t) + q_y^2(t) = 1, \quad t \in [0, t_f], \quad (9)$$

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. 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 sketch path that keeps away from 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. After constructing the graph Dijkstras 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. 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:

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

$$\frac{\dot{P}_a(1)}{|\dot{P}_a(1)|} = \frac{\dot{P}_b(0)}{|\dot{P}_b(0)|}. \quad (11)$$

Fig. 3 shows an example of the paths.

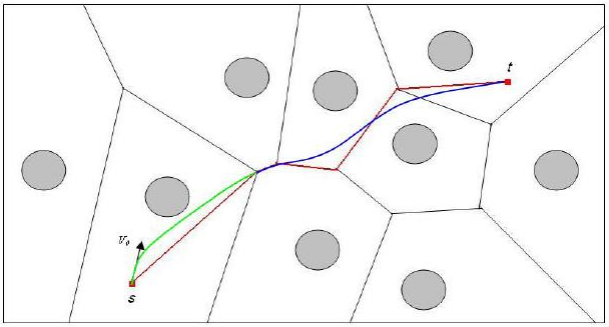



Fig. 3. A smooth path resulted from two Bezier curves. The first Bezier curve is illustrated in green and the second one is shown in blue [1].

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

$$(\dot{X} + \ddot{X})^2 + (\dot{Y} + \ddot{Y})^2 \leq 1, \tag{12}$$

where $h \in \mathbb{R}^+$ is the sample time interval. Note that the variables $X$, $Y$, and $t$ for (12) are normalized values by (4). For the sake of optimality $\alpha(t)$ was calculated such that the left side of the inequality constraint (12) approaches 1. To find $\lambda_n \triangleq \alpha(nh)$ for all $n$'s Sahraei used derivative approximations to define the function $f$:

$$\begin{aligned} f(\lambda) = \quad & \left(\tfrac{X(\lambda)-X(\lambda_{n-2})}{2h} + \tfrac{X(\lambda)+X(\lambda_{n-2})-2X(\lambda_{n-1})}{h^2}\right)^2 \\ & + \left(\tfrac{Y(\lambda)-Y(\lambda_{n-2})}{2h} + \tfrac{Y(\lambda)+Y(\lambda_{n-2})-2Y(\lambda_{n-1})}{h^2}\right)^2 - 1 + \varepsilon. \end{aligned} \tag{13}$$

$\lambda_n$ is calculated by solving the equation

$$f(\lambda) = 0 \tag{14}$$

based on $\lambda_{n-1}$ and $\lambda_{n-2}$ by Newton's method. $\varepsilon \in \mathbb{R}^+$ in (13) guarantees that the results of Newton's method make the left side of inequality (12) less than but close to 1. Since $\lambda_n$ relies on two previous values of $\lambda$, at the first step $\lambda_0$ and $\lambda_1$ should be calculated. It is straightforward that $\lambda_0 = 0$. To calculate $\lambda_1$, a hypothetical $\lambda_{-1}$ is introduced to approximate the position of the robot before initial time, $X(\lambda_{-1})$ defined by

$$\frac{X(\lambda_0) - X(\lambda_{-1})}{h} \approx v_{x_0}, \tag{15}$$

and so forth for $Y$.

## V. Proposed Algorithm

This section proposes a new algorithm for obstacle avoiding real-time trajectory generation of omnidirectional vehicles. To describe this method, let $a_n = (a_{x_n}, a_{y_n}) = (\ddot{x}(nh), \ddot{y}(nh))$, $v_n = (v_{x_n}, v_{y_n}) = (\dot{x}(nh), \dot{y}(nh))$, $z_n = (x_n, y_n) = (x(nh), y(nh))$ denote the acceleration, velocity, and position of the vehicle, respectively, at sample time. Suppose that all of the variables used in this section are nondimensional variables scaled by (4).

Note that the first derivatives of the nondimensional variables are given by

$$\frac{d\bar{x}}{d\bar{t}} = \frac{dt}{d\bar{t}}\frac{d}{dt}\bar{x} = \frac{dt}{d\bar{t}}\frac{d}{dt}\frac{x}{\Psi} = \frac{T}{\Psi}\frac{dx}{dt} \tag{16}$$

and so forth for $\frac{d\bar{y}}{d\bar{t}}$. Using similar derivation, the second derivatives are

$$\frac{d^2\bar{x}}{d\bar{t}^2} = \frac{T^2}{\Psi}\frac{d^2x}{dt^2} \tag{17}$$

and so forth for $\frac{d^2\bar{y}}{d\bar{t}^2}$.

The new algorithm uses 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 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.

The algorithm also deals with velocities and accelerations of the robot in discrete time system sampled by $t = \{0, h, 2h, \ldots\}$. However, it computes accelerations $a_n$ that meet optimal condition (9) as opposed to that Sahraei computes positions $z_n$. The set of all such accelerations $\mathfrak{A}_n^p$ is represented as

$$\mathfrak{A}_n^p = \{(a_{x_n}, a_{y_n}) | \quad (a_{x_n} + v_{x_n})^2 + (a_{y_n} + v_{y_n})^2 = 1\}. \tag{18}$$

If $v_n$ is given, (20) can be rewritten as

$$\mathfrak{A}_n^p = \{(-v_{x_n} + \cos\theta_n, -v_{y_n} + \sin\theta_n) | \quad \theta_n \in [0, 2\pi)\}. \tag{19}$$

The beauty of (19) is that it guarantees satisfaction of (9) while Sahraei's algorithm only has the left side of the equation approach 1. It also simplifies the value of the accelerations to one variable $\theta_n$.

The set of all feasible accelerations that meet dynamic constraint (6), $\mathfrak{A}_n$ is represented as

$$\mathfrak{A}_n = \{(a_{x_n}, a_{y_n}) | \quad (a_{x_n} + v_{x_n})^2 + (a_{y_n} + v_{y_n})^2 \leq 1\}. \tag{20}$$

Geometrically, $\mathfrak{A}_n^p$ is the circle that has center at $(-v_{x_n}, -v_{y_n})$ and radius of 1. $\mathfrak{A}_n$ is the union of boundary and area inside of the circle.

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

$$\mathbf{v}_{n+1} = \mathbf{v}_n + h\mathbf{a}_n, \quad \mathbf{a}_n \in \mathfrak{A}_n^p, \tag{21}$$

$$\mathbf{z}_{n+1} = \mathbf{z}_n + h\mathbf{v}_n + \frac{h^2}{2}\mathbf{a}_n, \quad \mathbf{a}_n \in \mathfrak{A}_n^p. \tag{22}$$

In the problems that we consider, $v_0$ and $z_0$ are initially given. So $a_0$ is solely determined by selecting $\theta_0$ in (19). Once $a_0$ is determined, $v_1$ and $z_1$ are obtained by applying (21) and (22) and using $a_0$, and so on. Thus we only need to find $\theta_n$ 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 $\theta_n$ at $t = nh$.

Equation (22) can be rewritten as the sum of two vectors:

$$\mathbf{z}_{n+1} = \mathbf{c}_{n+1} + \mathbf{r}_{n+1}, \tag{23}$$

where

$$\mathbf{c}_{n+1} = \begin{bmatrix} x_n \\ y_n \end{bmatrix} + (h - \frac{h^2}{2}) \begin{bmatrix} v_{x_n} \\ v_{y_n} \end{bmatrix}, \tag{24}$$

$$\mathbf{r}_{n+1} = \frac{h^2}{2} \begin{bmatrix} \cos\theta_n \\ \sin\theta_n \end{bmatrix}. \tag{25}$$

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

$$\mathfrak{Z}_{n+1}^p = \{\mathbf{z}_{n+1} | \quad |\mathbf{z}_{n+1} - \mathbf{c}_{n+1}| = \tfrac{h^2}{2}\}. \tag{26}$$

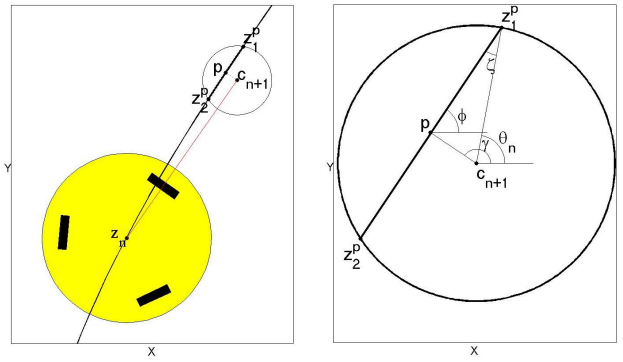
The set of all feasible $z_{n+1}$ corresponding to $\mathfrak{A}_n$, $\mathfrak{Z}_{n+1}$ is represented as

$$\mathfrak{Z}_{n+1} = \{\mathbf{z}_{n+1} | \quad |\mathbf{z}_{n+1} - \mathbf{c}_{n+1}| \leq \tfrac{h^2}{2}\}. \tag{27}$$

Geometrically, $\mathfrak{Z}_{n+1}^p$ 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. 4(a). $\mathfrak{Z}_{n+1}$ is the union of boundary and area inside of the circle. Let $\Omega_z$ denote the set of all feasible positions on the pre-generated Bézier curve $P(\lambda) = \big(X(\lambda), Y(\lambda)\big)$ at next sample time $t = nh$:

$$\Omega_z = \mathfrak{Z}_{n+1} \cap P(\lambda). \tag{28}$$

In the set, the intersections of $\mathfrak{Z}_{n+1}^p$ and $P(\lambda)$, $z_1^p$ and $z_2^p$ satisfy the optimal condition (9). We will select $z_1^p$, the one further down from current position as $z_{n+1}$ provides shorter travel time.




(a) $\mathfrak{Z}_{n+1}^p$ on a reference trajectory. (b) The enlarged $\mathfrak{Z}_{n+1}^p$.

Fig. 4. Geometry of $\mathfrak{Z}_{n+1}^p$ from $\mathbf{z}_n$ on a reference trajectory. $p$ is defined by the point on the reference trajectory, which is closet to $\mathbf{c}_{n+1}$

Assuming that the reference path is planned such that $\mathfrak{Z}_{n+1}$ intersect the path for all $n$'s, we only need to find $\theta_n$ corresponding to $z_1^p$ for motion planning of the vehicle. However, noise in a real system may have $\mathfrak{Z}_{n+1}$ miss the path. For this case, another path following heuristic is required. 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 *IR* mode, $\theta_n$ corresponding to $z_1^p$ 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. 4(a). To calculate $p$, we introduce the function $f$:

$$f(\lambda) = (X(\lambda) - c_{x_{n+1}})^2 + (Y(\lambda) - c_{y_{n+1}})^2. \tag{29}$$

Then $p$ is given by

$$p = \big(X(\lambda^p), Y(\lambda^p)\big), \tag{30}$$

where $\lambda^p$ is the solution to the equation $f(\lambda) = 0$, which is calculated by applying Newton's method. In Fig. 4(a), the portion of the Bézier curve inside of the circle can be considered to be line segment of which slope is equal to the slope of tangent at $p$, given that time interval $h$ is small enough. So we can approximate $z_1^p$ as the intersect point between the circle and the tangent line at $p$. Let $\phi$ denote the slope of the tangent line:

$$\phi = \tan^{-1}\big(\frac{\dot{Y}(\lambda^p)}{\dot{X}(\lambda^p)}\big). \tag{31}$$

Looking at the geometry of $p$, $z_1^p$, and $c_{n+1}$ in Fig. 4(b), $\theta_n$ is given by

$$\theta_n = \phi + \zeta, \tag{32}$$

where $\zeta$ can be calculated by applying law of sines for $\triangle z_1^p p c_{n+1}$:

$$\zeta = \sin^{-1}\big(\frac{2|p - c_{n+1}|}{h^2}\sin(\pi - \gamma + \phi)\big). \tag{33}$$

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

### B. OR *mode*

To account for noise presents in a real system, an efficient path following heuristic is presented. To describe this method, we introduce two terms: $y_{err}$ and $\psi_{err}$. The cross track error $y_{err}$ is defined by the distance between $c_{n+1}$ and $p$. The heading error $\psi_{err}$ is defined by the angle difference from the current heading of the robot, $\psi_n$ to the slope of tangent at $p$ (See Fig. 4(a)).

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

$$\delta\psi = k_p y_{err} + k_d \psi_{err} + k_i \int y_{err} dt \tag{34}$$

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

$$\delta\psi = \psi_{n+1} - \psi_n \tag{35}$$

$\theta_n$ of the acceleration $a_n$ that produces the desired $\delta\psi$ can be calculated in cost efficient way. Fig. 5 shows the relationship of $\delta\psi$ and $\theta_n$ in acceleration frame. From (19), $\mathfrak{A}_n^p$ is the circle that has center at $(-v_{x_n}, -v_{y_n})$ and radius of 1. Rewriting (21), the acceleration can be represented as the sum of two vectors:

$$\mathbf{a}_n = -\frac{1}{h}\mathbf{v}_n + \frac{1}{h}\mathbf{v}_{n+1} \tag{36}$$

Since $v_n$ is given, $-\frac{1}{h}\mathbf{v_n}$ is deterministic. It is straightforward that the direction of the vector $\frac{1}{h}\mathbf{v_n}$ is $\psi_n$. 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 the tip of $\mathbf{a}_n$ will be $\frac{1}{h}\mathbf{v_{n+1}}$. Similarly, direction of the vector is $\psi_{n+1}$. Since the desired $\delta\psi$ determines $\psi_{n+1}$, the intersect points between the vector $\frac{1}{h}\mathbf{v}_{n+1}$ and the circle $\mathbf{A}_n^p$ 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 $|\mathbf{v}_{n+1}|$ 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 intersect points is one. So

$$|\delta\psi|_{max} = \begin{cases} \sin^{-1}\left(\frac{1}{(1/h-1)|\mathbf{v}_n|}\right), & \text{if } (\frac{1}{h}-1)|v_n| > 1 \\ \pi, & \text{if } (\frac{1}{h}-1)|v_n| \le 1 \end{cases} \quad (37)$$

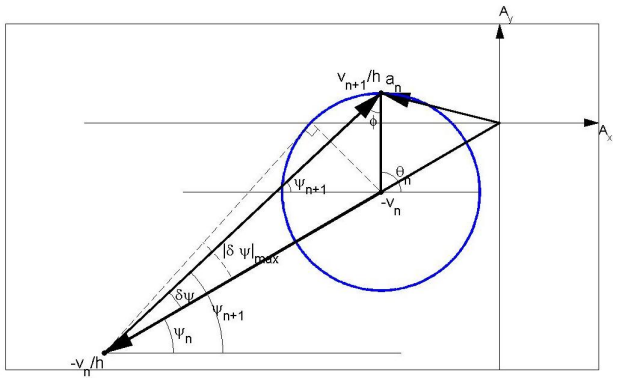



Fig. 5. Geometry of $\theta_n$ and $a_n$.

In Fig. 5, $\theta_n$ can be represented as

$$\theta_n = \psi_{n+1} + \phi = \psi_n + \delta\psi + \phi, \quad (38)$$

where $\phi$ can be obtained by using law of sines:

$$\phi = \sin^{-1}\left((\frac{1}{h}-1)|\mathbf{v}_n|\sin(\delta\psi)\right). \quad (39)$$

Note that $\delta\psi$ is signed angle and so is $\phi$ determined by $\delta\psi$. Equation (34) can be written as (40) by using (37), (38), and (39).

$$\theta_n = \psi_n + \phi + k_p y_{err} + k_d \psi_{err} + k_i \int y_{err} dt \quad (40)$$

subject to

$$\psi_n + \phi - |\delta\psi|_{max} \le \theta_n \le \psi_n + \phi + |\delta\psi|_{max} \quad (41)$$

In order to meet obstacle avoidance, maximum $y_err$ 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. Refining Accelerations

As explained in Section V, $a_n \in \mathfrak{A}_n^p$ guarantees optimal condition (9) at $t = nh$. Given that we choose small enough value for $h$, violation of the dynamic constraint (6) due to deviation of velocity for time interval $t \in \big(nh, (n+1)h\big)$ can be ignored. However, dealing with the dynamic constraint for the interval gives more reliability and accuracy for motion planning of the robot. In this section, we are to find accelerations that nearly guarantees the optimal condition while satisfying the dynamic constraint for $t \in \big[nh, (n+1)h\big)$.

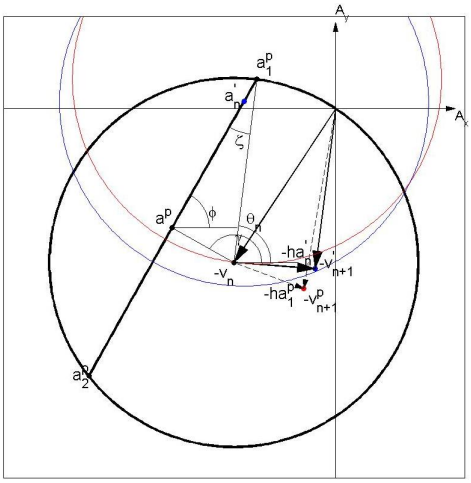



Fig. 6. Adjusted acceleration.

### A. IR *mode*

We refer Fig. 6 to explain refinement of accelerations in *IR* mode. Fig. 6 shows $\mathfrak{A}_n$, the circle that has center at $-v_n$ and radius of 1 in acceleration frame. $a^p$, $a_1^p$, and $a_2^p$ denote the corresponding accelerations to $p$, $z_1^p$, and $z_2^p$, respectively (See Fig. 4(a) and 4(b)). They are given by using (22). Since $a_n$ and $z_{n+1}$ have linear relationship in (22), angle variables $\phi$, $\zeta$, $\gamma$, and $\theta_n$ are same as the ones in position frame. The curve segment $\Omega_a$ that pass through $a_1^p$, and $a_2^p$ is the set of the accelerations corresponding to $\Omega_z$:

$$\Omega_a = \{a_n | \quad a_n = \frac{2(z_{n+1} - z_n - v_n h)}{h^2}\}, \quad z_{n+1} \in \Omega_z. \quad (42)$$

In Fig. 6, $v_{n+1}^p$ denotes the velocity at next step when $a_1^p$ is chosen as $a_n$:

$$\mathbf{v}_{n+1}^p = \mathbf{v}_n + h\mathbf{a}_1^p. \quad (43)$$

Notice that the vector $-h\mathbf{a}_1^p$ propagates outward of the circle that has center at $a_1^p$ and radius of 1 from $-v_n$. It means that the dynamic constraint (6) is violated for $t \in \big[nh, (n+1)h\big)$.

We are to find new feasible acceleration $a_n' \in \Omega_a$ so that the dynamic constraint is satisfied for the time interval. Geometrically, the circle $\mathfrak{A}_n'$ that has center at $a_n'$ and radius of 1 should contain the trajectory of $-v(t)$ for $t \in \big(nh, (n+1)h\big)$. For the sake of short travel time, the acceleration is chosen

near $a_1^p$. Since $\Omega_a$ can be considered to be line segment for small $h$, the acceleration is represented as

$$\mathbf{a}'_n = \mathbf{a}_1^p + k\mathbf{m}, \quad k \in \mathbb{R}^+, \tag{44}$$

where

$$\mathbf{m} = -\begin{bmatrix}\dot{X}(\lambda^p) & \dot{Y}(\lambda^p)\end{bmatrix}^T. \tag{45}$$

Recall that the vehicle follows constant acceleration equations of motion with $a_n$ for $t \in [nh,(n+1)h)$. So $v_x(t)$ and $v_y(t)$ linearly increase or decrease for the interval. That is, the trajectory of $v(t)$ for $t \in [nh,(n+1)h)$ is the line connecting $v_n$ and $v_{n+1}$. So, it is sufficient that $\mathfrak{A}'_n$ contains $-v_n$ and $-v_{n+1}$. Since $a'_n \in \mathfrak{A}_n$, $\mathfrak{A}'_n$ always contains $-v_n$. Thus we only need to find $k$ such that $-v_{n+1}$ is within $\mathfrak{A}'_n$. Since smaller $k$ leads to shorter travel time, $k$ is calculated by solving

$$|\mathbf{a}'_n + \mathbf{v}_n + h\mathbf{a}'_n| = 1. \tag{46}$$

The equation (46) can be rewritten as the second polynomial with respect to $k$:

$$ak^2 + bk + c = 0, \tag{47}$$

where

$$\begin{aligned} a &= (1+h)^2(m_x^2 + m_y^2) \\ b &= 2(1+h)(m_x(v_{x_n} + (1+h)a_{x_1}^p) + m_y(v_{y_n} + (1+h)a_{y_1}^p)), \\ c &= ((1+h)a_{x_1}^p + v_{x_n})^2 + ((1+h)a_{y_1}^p + v_{y_n})^2 - 1 \end{aligned} \tag{48}$$

$\begin{bmatrix}m_x & m_y\end{bmatrix}^T = \mathbf{m}$ and $\begin{bmatrix}a_{x_1}^p & a_{y_1}^p\end{bmatrix}^T = \mathbf{a}_1^p$. The smaller root of (47) is selected as $k$.

### B. OR *mode*

We also refer Fig. 6 to describe refinement of accelerations in *OR* mode. In *OR* mode, $a_1^p$ and $a_2^p$ is the accelerations determined by intersections between $\mathfrak{A}_n^p$ and the vector $\frac{1}{h}\mathbf{v}_{n+1}$ from $-v_n$ in acceleration frame (See Fig. 5). The new acceleration $a'_n$ is represented by (44) with $\mathbf{m}$ given by

$$\mathbf{m} = -\begin{bmatrix}\cos(\psi_n + \delta\psi) & \sin(\psi_n + \delta\psi)\end{bmatrix}^T. \tag{49}$$

Similarly, the smaller root of (47) is selected as $k$.

## VII. 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, and smooth motion control compared to Sahraei's algorithm. Also, they show robustness of the proposed algorithm. Fig. 7 shows the course used for the simulation. Red circles indicate obstacles.

The initial position and the velocity are given by:

$$z_0 = (1.75, 0.54) \quad [m], \tag{50}$$

$$v_0 = (0,0) \quad [m/s]. \tag{51}$$

The final position is given by

$$z_f = (6.85, 3.28) \quad [m]. \tag{52}$$

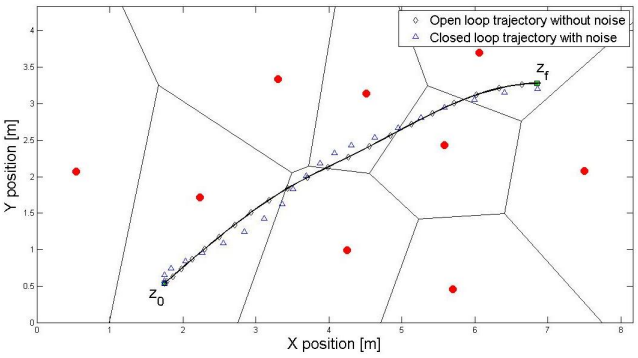



Fig. 7. The resulting trajectories by different algorithms over the reference trajectory (bold black curve).

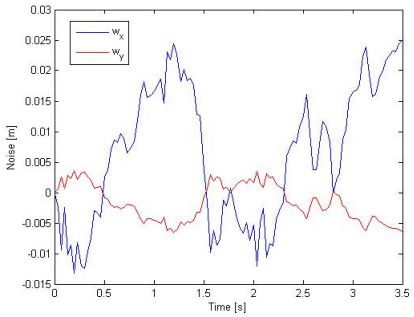



Fig. 8. Noise for the close-loop trajectory.

The sample time interval $h$ is given by

$$h = 0.0033[s]. \tag{53}$$

Characteristic variables are given by

$$\alpha = 1[N/V], \quad \beta = 1[kg/s], \quad m = 1[kg], \quad Umax = 3[v]. \tag{54}$$

The reference trajectory is constructed by a Bézier curve of which control points are given by

$$\begin{aligned} p_0 &= (1.75, 0.54), p_1 = (3.49, 2.05), p_2 = (3.72, 2.14), \\ p_3 &= (4.55, 2.04), p_4 = (5.35, 3.24), p_5 = (6.85, 3.28), \end{aligned} \tag{55}$$

and illustrated as bold black curve in Fig. 7. The simulation of Sahraei's algorithm has been done with the same parameters above and $\varepsilon = 0.01$.

In Fig. 7, two kinds of trajectories are generated depending on addition of noise. The open loop trajectory without noise is generated by applying *IR* logic of the proposed algorithm and Sahraei's algorithm. The close loop trajectory with noise is generated by combining *IR* and *OR* logic of the proposed algorithm. The resulting trajectory shows the robustness to noise in Fig. 7. Noise was modeled as white noise as shown in Fig. 8 and added to actual position. The simulation results are listed in table I. The resulting final time $t_f$ by the proposed algorithm is 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

TABLE I

RESULTS OF THE SIMULATION

| Methods | $t_f$ $[s]$ | Violation of (6) [%] | $\int_0^{t_f} \lvert y_{err}\rvert^2 dt$ |
|---|---|---|---|
| Open-loop without noise by IR | 3.6667 | 0 | 0 |
| Close-loop with noise by IR and OR | 3.6333 | 0 | $9.2435 \times 10^{-5}$ |
| Open-loop without noise by Sahraei | 13.2667 | 31.91 | 0 |

boundary condition 1 in Fig. 9(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. 9(d) and 9(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. 9(a), 9(b), 9(g) and 9(h).

## VIII. 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 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.

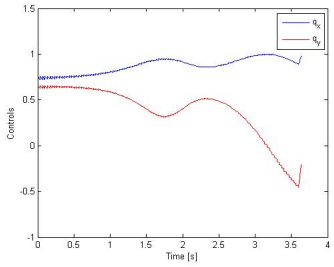
[5] Jung, M., Shim, H., Kim, H. and Kim, J., "The Miniature Omnidirectional Mobile Robot OmniKity-I (OK-I)," *International Conference on Robotics and Automation,* p. 2686-2691, 1999.

[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.

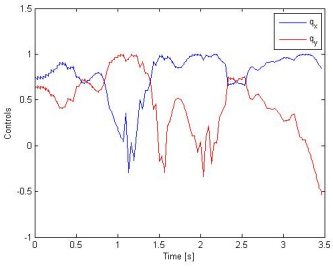
[7] Huang, W. H., Fajen, B. R., Fink, J. R., Warren, W. H., "Visual navigation and obstacle avoidance using a steering potential function," *Robotics and Autonomous Systems,* vol. 54, Issue 4, p. 288-299, 28 April 2006.

[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.

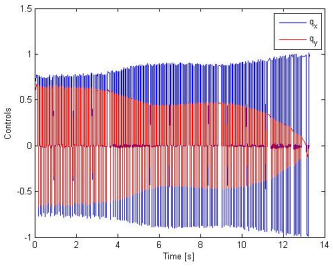
[9] Hwang, C., Chang, L. "Trajectory Tracking and Obstacle Avoidance of Car-Like Mobile Robots in an Intelligent Space Using Mixed $H_2/H_\infty$ Decentralized Control," *Mechatronics, IEEE/ASME Transactions on,* vol. 12, Issue 3, p. 345-352, June 2007



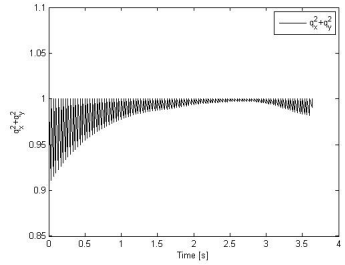

(a) $q_x$ and $q_y$ by IR.



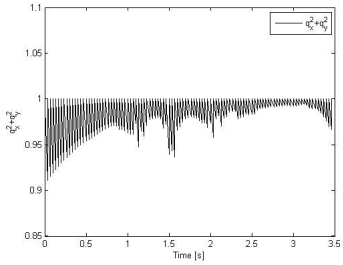

(b) $q_x$ and $q_y$ by IR and OR (noise).



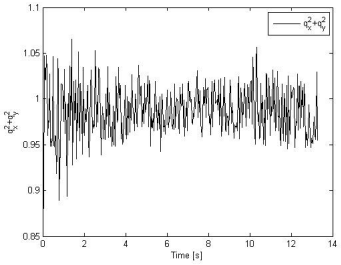

(c) $q_x$ and $q_y$ by Sahraei.



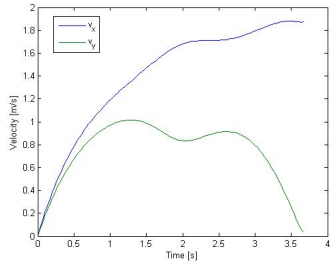

(d) $q_x^2+q_y^2$ by IR.



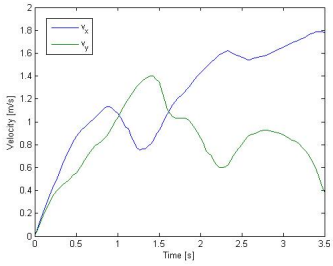

(e) $q_x^2+q_y^2$ by IR and OR (noise).



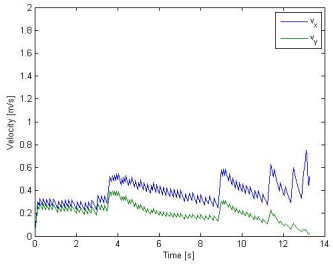

(f) $q_x^2+q_y^2$ by Sahraei.




(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. 9. The results obtained by the proposed algorithm and Sahraei's algorithm.