Dynamic Obstacle Avoidance for an Omni-Directional Mobile Robot

Robert L. Williams II and **Jianhua Wu**
Ohio University, Athens, OH 45701

Final Manuscript
Journal of Robotics
Hindawi Publishing Corporation
September, 2010

Keywords: mobile robot, dynamic obstacle avoidance, moving obstacles, global deliberate approach, local reactive approach, coupled nonlinear dynamic model.

Abstract:

We have established a novel method of obstacle-avoidance motion planning for mobile robots in dynamic environments, wherein the obstacles are moving with general velocities and accelerations and their motion profiles are not pre-known. A hybrid system is presented in which a global deliberate approach is applied to determine the motion in the desired path line (*DPL*), and a local reactive approach is used for moving obstacle avoidance. A machine vision system is required to sense obstacle motion. Through theoretical analysis, simulation, and experimental validation applied to the Ohio University RoboCup robot, we show the method is effective to avoid collisions with moving obstacles in a dynamic environment.

Corresponding author:

Robert L. Williams II, Professor
Department of Mechanical Engineering
259 Stocker Center, Ohio University
Athens, OH 45701-2979
Phone: (740) 593-1096
Fax: (740) 593-0476
E-mail: williar4@ohio.edu
URL: oak.cats.ohiou.edu/~williar4

1. INTRODUCTION

An omni-directional robot is a holonomic robot that can move simultaneously in rotation and translation (Pin et al., 1994). Most work on omni-directional robots is in robot development; the few studies on dynamic models are Watanabe et al. (1998), Moore and Flann, (2000), Williams et al. (2002), and Kalmar-Nagy et al. (2004). These models all have decoupling between the wheels, which is not complete; thus, we first briefly summarize a new coupled non-linear dynamics model for three-wheeled omni-directional robots.

The potential field method was first suggested by Andrews and Hogan (1983) and Khatib (1985) for obstacle avoidance of manipulators and mobile robots. Obstacles exert a virtual repulsive force, while the goal applies a virtual attractive force to the robot. Koren and Borenstein (1991) identify potential field limitations (robot trapped by local minima, oscillation in presence of obstacle, the lack of passage between closely-spaced obstacles). To overcome these problems they developed the vector field histogram. Ge and Cui (2000) mentioned an additional shortcoming, a non-reachable goal with an obstacle nearby, and presented a new repulsive function to overcome it, increasing complexity and computation. Adams (1999) presented a simulation study using the potential field method considering low-level robot dynamics, with static obstacles. Guldner and Utkin (1995) proposed a method that took the gradient of the potential field as the desired vector field for path planning. Tsourveloudis et al. (2001) proposed an electrostatic potential field for an autonomous mobile robot in a planar dynamic environment; their method depends on obstacle prediction accuracy, and slow environment changes.

The velocity space method to deal with moving obstacle avoidance in a pre-known environment was suggested by Fiorini and Shiller (1998), who discussed the velocity obstacle concept. This method lacks potential field simplicity, and is effective only if the obstacle moves with constant speed. Chakravarthy and Ghose (1998) and Tsoularis and Kambhampati (1998) proposed methods which use relative speed to detect collisions. Fujimura and Samet (1989), and Conn and Kam (1998) presented studies of motion planning in a dynamic environment, but in both of their studies the dynamic environments need to be completely pre-known.

Chang et al. (1994) presented a two-phase neural-network-based deliberate path-planning and reactive motion-planning hybrid navigation system for static obstacles. Xu et al. (2003) proposed a reactive motion planner that uses local rather than global environment information for static obstacles.

Recently other authors have presented works on moving obstacle avoidance for robots. Leng, et al. (2008) presented an improved artificial potential field method by introducing an Anisotropic-Function. The motion ability of an Omni-directional robot in different directions was considered to improve the motion planning and the moving obstacle velocity was pre-set. Belkhouche (2009) presented reactive path planning for a dynamic environment. der Berg and M. Overmars (2007) presented kinematic and dynamic motion planning roadmaps for dynamic environments.

The potential field method is normally used for path planning and obstacle avoidance in a static environment; still relatively few papers deal with moving obstacle avoidance. The velocity space method applies relative robot/obstacle speed and position to detect a possible collision and plan a motion and normally requires pre-known constant obstacle and robot speeds. No fixed time motion is found in literature on obstacle avoidance. In the current article we present a new dynamic obstacle avoidance method with a hybrid (globally deliberate and locally reactive) navigation system and the concept of using relative velocity to detect possible collisions. We have established a novel method of obstacle-avoidance motion planning for mobile robots in dynamic environments, wherein the obstacles are moving with general velocities and accelerations and their motion profiles are not pre-known. A hybrid system is presented in which a global deliberate approach is applied to determine the motion in the desired path line (*DPL*), and a local reactive approach is used for moving obstacle avoidance.

This article first summarizes a new coupled non-linear dynamics model for wheeled holonomic omni-directional mobile robots, followed by development of the novel dynamic obstacle avoidance algorithm, with simulation examples and experimental validation.

2. THREE-WHEELED OMNI-DIRECTIONAL ROBOT MODELING SUMMARY

This section presents a brief summary of kinematic and dynamic modeling, plus controller development. For more details, see Wu (2004). Figure 1 shows a bottom view of the Ohio University RoboCup robot. In Figure 2, {*w*} is the fixed world coordinate frame and {*m*} is the moving frame, with the same origin as {*w*} but rotating with the robot. The x_m axis is set to always be perpendicular to traction force T_1, and ϕ is defined as the angle of x_m with respect to x_w. ${}^{t}\mathbf{F} = [T_1\ T_2\ T_3]^T$ is the traction force of the ground on the wheels and ${}^{m}\mathbf{F} = [{}^{m}F_x\ {}^{m}F_y\ {}^{m}T_Z]^T$ is the Cartesian force and moment on the robot in the moving frame. L is the radial distance to the wheels from the robot center.

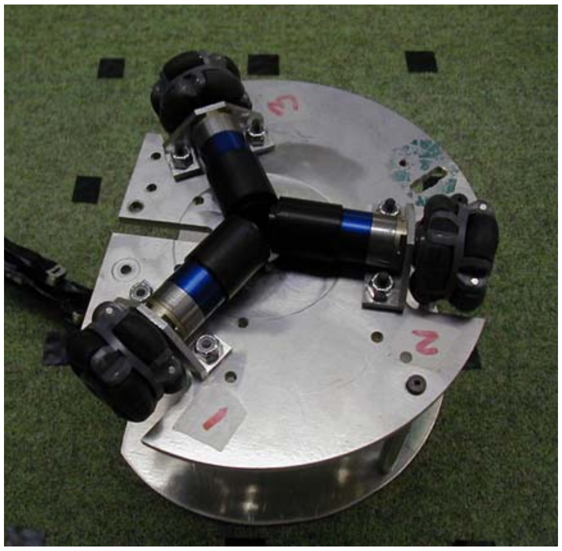


Figure 1. RoboCup Robot (Bottom)

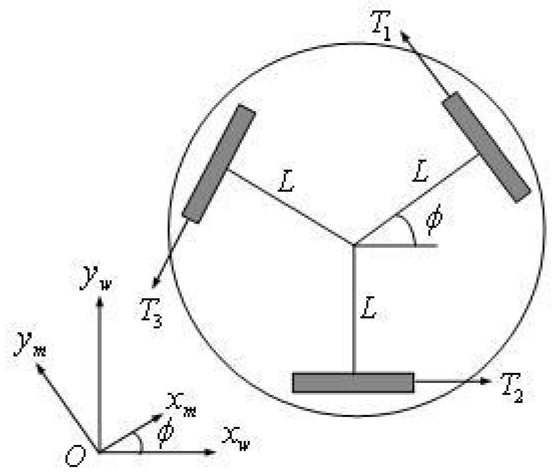


Figure 2. Omni-Directional Robot Geometry

r is the radius of the wheels; ${}^{w}\dot{\mathbf{X}} = [\dot{x}_w\ \dot{y}_w\ \dot{\phi}_w]^T$ and ${}^{m}\dot{\mathbf{X}} = [\dot{x}_m\ \dot{y}_m\ \dot{\phi}_m]^T$ are the robot Cartesian velocity in {*w*} and {*m*}, and $\dot{\mathbf{q}}_L = [\dot{q}_{L1}\ \dot{q}_{L2}\ \dot{q}_{L3}]^T$ are the wheel angular velocities. The velocity kinematics equations in {*m*} or {*w*} are (1) and (2) is the acceleration kinematics equation.

$$ {}^{m}\dot{\mathbf{X}} = r\left[\mathbf{B}^{\mathbf{T}}\right]^{\mathbf{-1}}\dot{\mathbf{q}}_L \qquad {}^{w}\dot{\mathbf{X}} = {}^{w}_{m}\mathbf{R}\ {}^{m}\dot{\mathbf{X}} = r\ {}^{w}_{m}\mathbf{R}\left[\mathbf{B}^{\mathbf{T}}\right]^{\mathbf{-1}}\dot{\mathbf{q}}_L \tag{1} $$

$$ {}^{w}\ddot{\mathbf{X}} = r\left({}^{w}_{m}\dot{\mathbf{R}}\left[\mathbf{B}^{\mathbf{T}}\right]^{\mathbf{-1}}\dot{\mathbf{q}}_L + {}^{w}_{m}\mathbf{R}\left[\mathbf{B}^{\mathbf{T}}\right]^{\mathbf{-1}}\ddot{\mathbf{q}}_L\right) \tag{2} $$

where ${}^{w}_{m}\mathbf{R}$ is the orthonormal rotation matrix which rotates vectors in {*m*} to {*w*} (Craig, 2005).

Without derivation our coupled nonlinear dynamics model is (Wu, 2004):

$$\mathbf{E} = \frac{k_1}{k_{lr}k_M}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\ddot{\mathbf{q}}_m + \frac{k_2}{k_{lr}k_M}\begin{bmatrix}0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 0\end{bmatrix}\ddot{\mathbf{q}}_m + (\frac{k_3}{k_{lr}k_M} + k_E)\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\dot{\mathbf{q}}_m + \frac{k_4\dot{\phi}}{k_{lr}k_M}\begin{bmatrix}0 & 1 & -1\\ -1 & 0 & 1\\ 1 & -1 & 0\end{bmatrix}\dot{\mathbf{q}}_m \tag{3}$$

where:

$$k_1 = J_m + \frac{J_L}{n^2} + \frac{(4mL^2 + J)\, r^2}{9L^2 n^2} \qquad k_2 = \frac{(-2mL^2 + J)\, r^2}{9L^2 n^2} \qquad k_3 = c_m + \frac{c_L}{n^2} \quad k_4 = \frac{2\sqrt{3}mr^2}{9n^2} \tag{4}$$

$$\dot{\phi} = \frac{r\,(\dot{q}_{L1} + \dot{q}_{L2} + \dot{q}_{L3})}{3L} = \frac{r\,(\dot{q}_{m1} + \dot{q}_{m2} + \dot{q}_{m3})}{3nL} \tag{5}$$

J and m are the rotational inertia and mass of the robot, $\dot{\mathbf{q}}_m = [\dot{q}_{m1}\ \dot{q}_{m2}\ \dot{q}_{m3}]^T$ are the motor angular velocities, c_m and c_L are the motor and load rotational damping coefficients, J_m and J_L are the motor and load rotational inertias, n is the gear ratio, $\mathbf{E} = [E_1\ E_2\ E_3]^T$ where E_i is the i^{th} motor voltage input, k_E and k_M are the motor back *emf* and torque constants, and k_{lr} is the inverse of the motor terminal resistance R.

In simulation and hardware we have implemented a two-level closed-loop controller (Figure 3). The outer loop is a Cartesian pose controller based on machine vision and the inner loop is a wheel velocity controller based on motor encoder feedback. For each level we use independent linear PI controllers where the gains were obtained through trial and error in simulation; the same gains worked well for the hardware. For many straight-line, triangular, and circular motion commands, the simulation and experiment agrees well with the commanded motion indicating that our model is good and the controller is effective to compensate for the wheel coupling and nonlinearity in (3). See Figure 4 for one sample experimental motion without obstacles, comparing desired and actual position and orientation of the robot. The robot is commanded to move in a straight line between two points, dwell for 2 seconds, and return to the original point, with constant orientation. The robot follows the desired path smoothly and closely.

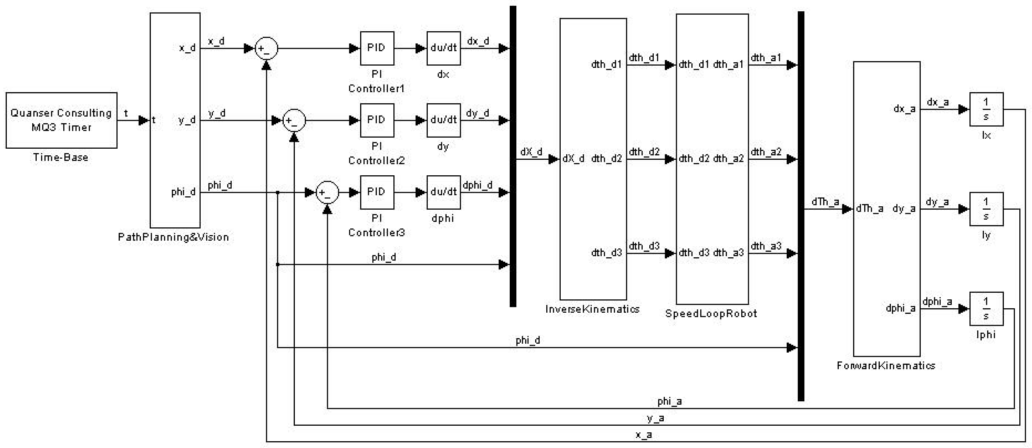


Figure 3. Two-Level Control Architecture

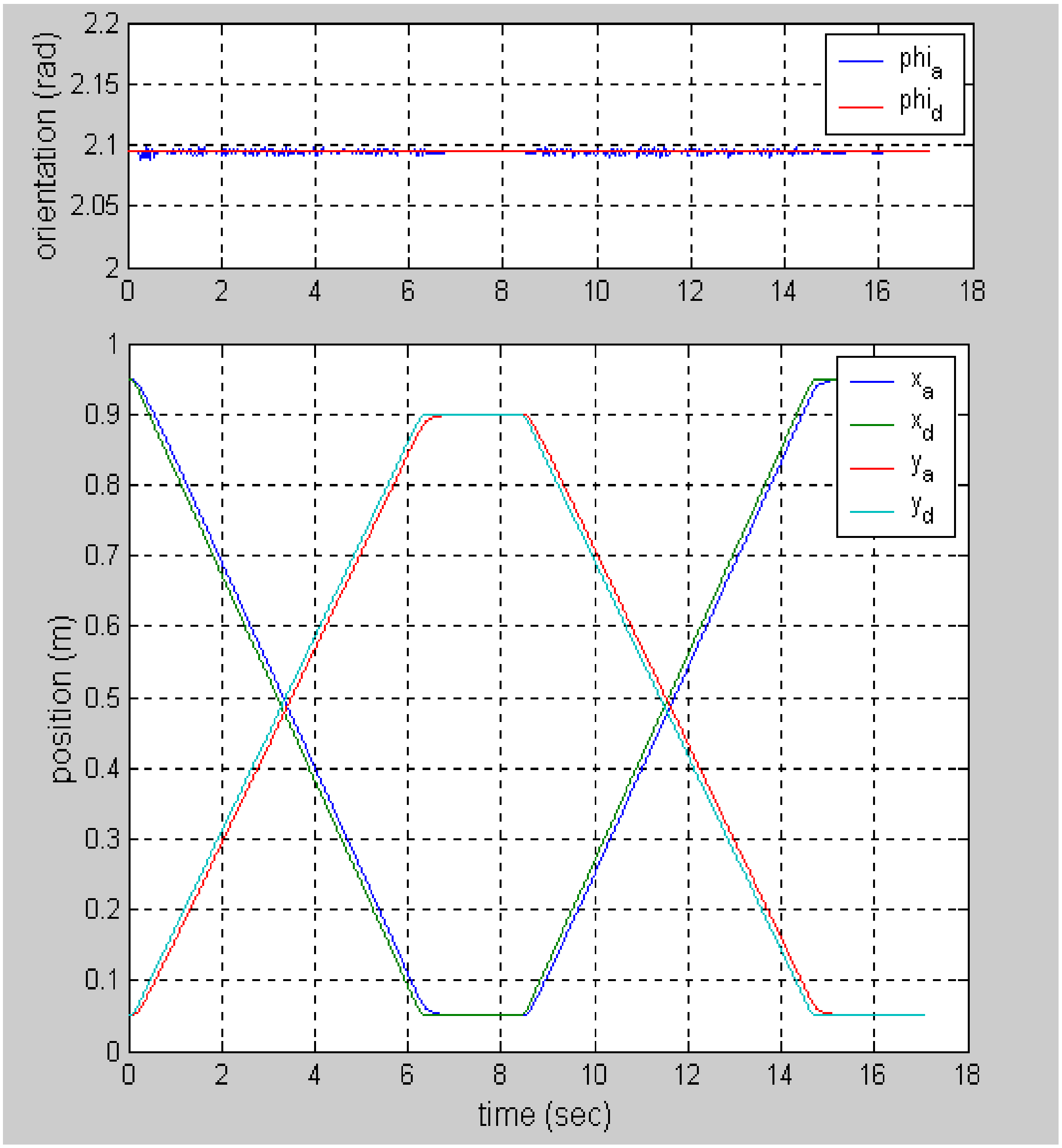


Figure 4. Robot Motion Test

We have developed novel Velocity and Acceleration Cones to characterize the practical kinematic and dynamic constraints for holonomic three-wheeled omni-directional mobile robots. These are applied to ensure the path planner does not request more velocity than the robot is capable of kinematically nor excessive acceleration that causes actuator saturation or wheel slippage. Though we have no space to present this, our obstacle avoidance method in this article is subject to these practical constraints (see Wu et al., 2006, for details).

3. DYNAMIC OBSTACLE AVOIDANCE

3.1 Problem Statement and Approach

As shown in Figure 5, a mobile robot at point *A* must reach goal point *B* in a fixed time $t_B - t_A$, while avoiding collisions with static and moving obstacles. Line *AB* is a given desired path line (*DPL*). The obstacles' motions are not pre-known, but will be sensed in real-time via machine vision. We assume the speed and acceleration capacity of all obstacles are similar to that of the mobile robot.

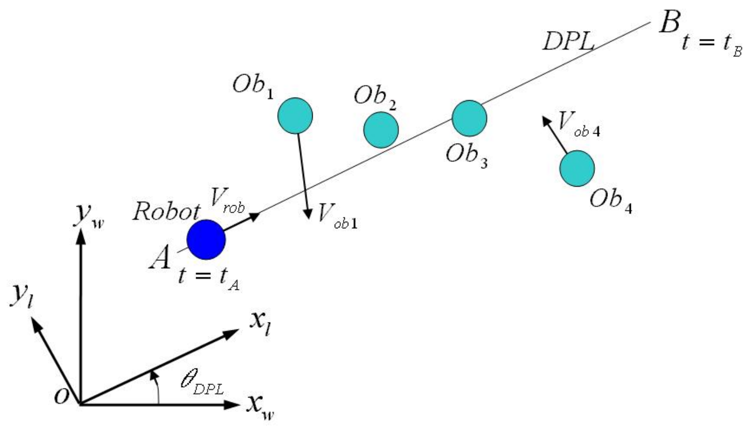


Figure 5. Mobile Robot in Dynamic Environment with Moving Obstacles

We present a hybrid approach wherein a global deliberate approach is applied to motion along the *DPL* while a local reactive approach is used to avoid collisions with obstacles. This obstacle avoidance in a non-pre-known environment is not necessarily optimal (when reviewed at a later time), because the obstacles' motions are not pre-known. Obstacle-avoidance decisions made by the robot are based only on past and current obstacle motion data from the vision system to estimate the current velocity of the obstacle. No further prediction about the future obstacle motion is made because smart obstacles may change motion according to our robot motion. All obstacle avoidance robot motions will be restricted to a fixed time motion problem: along the direction of the *DPL*, the motion of the robot is a nearly constant speed motion (ramping up from and down to zero velocity and the start and end, according to the robot acceleration capacity). Obstacle avoidance is realized by changing the robot speed in the direction perpendicular to the *DPL*. This means if the obstacle is on the line between the robot and target, the robot will change the speed in the direction perpendicular to the *DPL* while keeping the same speed in the direction along with *DPL* to reach the target in the fixed time period.

3.2 Moving Obstacle Avoidance

In Figure 5, {*w*} is the fixed world coordinate system. The local, or *DPL*, coordinate system {*l*} has the same origin as {*w*} and x_l is parallel to the *DPL*. In our obstacle avoidance algorithm, the starting and destination points and other robot motion conditions are transformed to *DPL* coordinates and then transformed back to {*w*} for the robot path. The motion along the x_l direction is pre-determined (due to the fixed-time requirement), while the motion in the y_l direction will be determined in *DPL* coordinates for obstacle avoidance, subject to kinematics and dynamics constraints. $\theta_{DPL} = \text{atan2}(\sqrt{y_B - y_A}, \sqrt{x_B - x_A})$ is the angle of x_l with respect to x_w where ${}^w\mathbf{X}_A = \begin{bmatrix} x_A & y_A \end{bmatrix}^T$ and ${}^w\mathbf{X}_B = \begin{bmatrix} x_B & y_B \end{bmatrix}^T$ are the vector position of points A and B in {*w*}, and atan2 is the quadrant-specific inverse tangent function. The Cartesian position and translational velocity (of the robot or obstacles) in *DPL* coordinates are found from (6), where ${}^w\dot{\mathbf{X}}$ is the Cartesian velocity in {*w*} and ${}^l_w\mathbf{R}$ is the inverse (transpose) of the orthonormal rotational matrix ${}^w_l\mathbf{R}$:

$$ {}^l\mathbf{X} = {}^l_w\mathbf{R}\,{}^w\mathbf{X} \qquad {}^l\dot{\mathbf{X}} = {}^l_w\mathbf{R}\,{}^w\dot{\mathbf{X}} \qquad {}^l_w\mathbf{R} = \begin{bmatrix} \cos\theta_{DPL} & \sin\theta_{DPL} \\ -\sin\theta_{DPL} & \cos\theta_{DPL} \end{bmatrix} \qquad (6) $$

3.2.1 Motion in x_l ***Direction****.* In *DPL* coordinates, we assign the robot motion along the *DPL* (x_l) as shown in Figure 6:

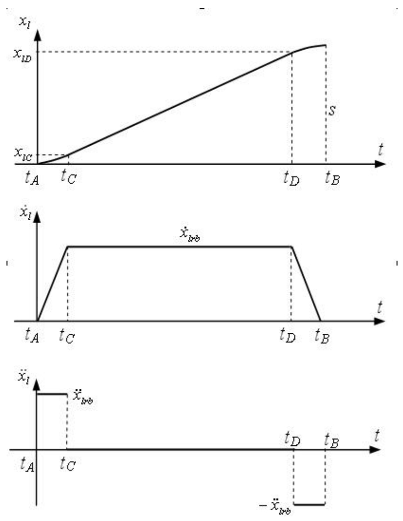


Figure 6. Motion of Robot in x_l Direction

Equations (7) are for the intermediate time points where *s* is the distance between points *A* and *B*, and $\dot{x}$ and $\ddot{x}$ are the velocity and acceleration of the robot in the x_l direction.

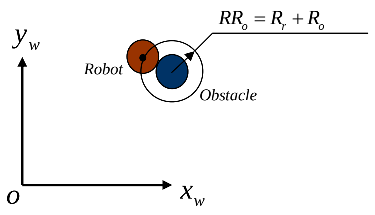
$$t_C = \dot{x}/\ddot{x} \qquad t_D = s/\dot{x} \qquad t_B = s/\dot{x} + \dot{x}/\ddot{x} \qquad \Delta t_{DC} = s/\dot{x} - \dot{x}/\ddot{x} \tag{7}$$

From kinematics we can find the motion in time periods Δt_{CA}, Δt_{DC} and Δt_{BD}:

$$\begin{aligned}
x_l &= \frac{1}{2}\ddot{x}t^2 && t < \frac{\dot{x}}{\ddot{x}} \\
x_l &= \frac{\dot{x}^2}{2\ddot{x}} + \dot{x}(t - \frac{\dot{x}}{\ddot{x}}) && \frac{\dot{x}}{\ddot{x}} \le t < \frac{s}{\dot{x}} \\
x_l &= s - \frac{\dot{x}^2}{2\ddot{x}} + \dot{x}(t - \frac{s}{\dot{x}}) - \frac{1}{2}\ddot{x}(t - \frac{s}{\dot{x}})^2 && \frac{s}{\dot{x}} \le t < \frac{s}{\dot{x}} + \frac{\dot{x}}{\ddot{x}} \\
x_l &= s && t \ge \frac{s}{\dot{x}} + \frac{\dot{x}}{\ddot{x}}
\end{aligned} \tag{8}$$

The robot starts from point A, constantly accelerates by $\ddot{x}$ to velocity $\dot{x}$ then moves with constant velocity $\dot{x}$ in Δt_{DC}, then constantly decelerates by $\ddot{x}$, and stops at point B. During time period $\frac{\dot{x}}{\ddot{x}} \le t < \frac{s}{\dot{x}}$, the speed of the robot in the x_l direction is the constant $\dot{x}$. The total time to move from points *A* to *B* is t_B from (7), assuming t_A is reset to zero for each motion. $\ddot{x}$ can be assigned as the maximum achievable robot acceleration in the x_l direction. If the motion time and $\ddot{x}$ are given, $\dot{x}$ is determined from t_B of (7). The robot velocity and acceleration must remain within their practical kinematic and dynamic constraints (Wu et al., 2006).

3.2.2 Basic Strategy of Obstacle Avoidance. We can simplify by expanding the obstacle radius to $RR_o = R_r + R_o$ (Figure 7) where R_r is the robot radius and R_o is the obstacle radius. The robot then becomes a point which must avoid the expanded obstacle.



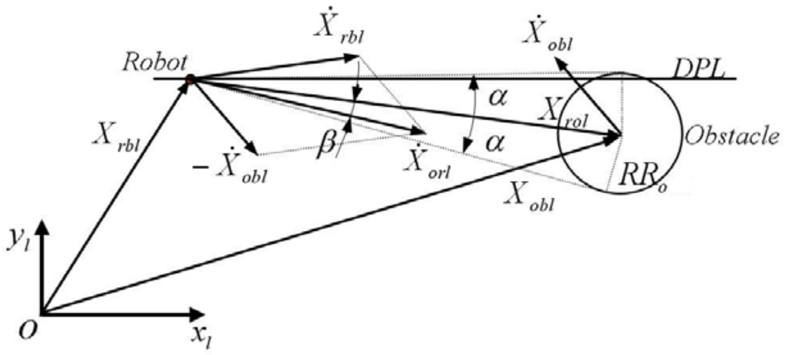


Figure 7. Expanded Obstacle **Figure 8. Basic Concept of Moving Obstacle Avoidance**

Figure 8 shows the vectors used in the moving obstacle avoidance algorithm. In *DPL* coordinates, vector $\mathbf{X}_{rol}$ shows the relative position between the point robot and an expanded obstacle (from the robot to the obstacle). From vector loop closure $\mathbf{X}_{rol} = \mathbf{X}_{obl} - \mathbf{X}_{rbl}$, where $\mathbf{X}_{obl}$ and $\mathbf{X}_{rbl}$ are position vectors of the obstacle and the robot. The distance between the robot and obstacle is $dis_{rol} = \|\mathbf{X}_{rol}\|$. If the robot and obstacle velocities are $\dot{\mathbf{X}}_{rbl}$ and $\dot{\mathbf{X}}_{obl}$, the relative velocity of the robot to the obstacle is $\dot{\mathbf{X}}_{orl} = \dot{\mathbf{X}}_{rbl} - \dot{\mathbf{X}}_{obl}$. Angle α is between the tangent line from the expanded obstacle circle to the point robot and relative the position vector $\mathbf{X}_{rol}$:

$$\alpha = \operatorname{atan2}(RR_{obst}, \sqrt{dis_{rol}^2 - RR_{obst}^2}) \tag{9}$$

β is the angle between the relative position vector $\mathbf{X}_{rol}$ and the relative velocity vector $\dot{\mathbf{X}}_{orl}$. If $\beta > \alpha$ while the robot passes an obstacle, there will be no collision (Figure 8 pictures the opposite case, i.e. $\beta < \alpha$, where there will be a collision). The strategy for moving obstacle avoidance is to check angles α and β in each time step and command the motion of the robot in the y_l direction (while the planned x_l motion continues) to keep the relative velocity vector $\dot{\mathbf{X}}_{orl}$ away from the possible collision range determined by angle α.

Whether to increase or to decrease the velocity of the robot in the y_l direction depends on which is the easier way to turn the relative velocity $\dot{\mathbf{X}}_{orl}$ away from the collision range. In the case of single obstacle avoidance, if we start checking the relationship between α and β early enough ($\dot{y}_l = 0$ at the start), we can simply choose to increase or decrease $\dot{y}_l$ of the robot by comparing vectors of $\dot{\mathbf{X}}_{orl}$ and $\mathbf{X}_{rol}$. Two unit vectors $\mathbf{u}_{vl} = \dot{\mathbf{X}}_{orl} / \|\dot{\mathbf{X}}_{orl}\|$ and $\mathbf{u}_{pl} = \mathbf{X}_{rol} / \|\mathbf{X}_{rol}\|$ are compared to determine the direction of $\dot{y}_l$ (Figure 9). If $\mathbf{u}_{vl}(y) > \mathbf{u}_{pl}(y)$, we choose increasing $\dot{y}_l$ of the robot, and choose decreasing $\dot{y}_l$ (increasing $\dot{y}_l$ in the negative direction) in the reverse case. In the case shown in Figure 9, $u_{vl}(y)$ is more negative than $\mathbf{u}_{pl}(y)$, thus we increase $\dot{y}_l$ in the negative y_l direction until $\beta > \alpha$. The original $\dot{\mathbf{X}}_{orl}$ is changed to $\dot{\mathbf{X}}_{orl_{\text{mod}}}$.

Forwarding to the *DPL* is also considered in our algorithm as choosing the easier way to turn away the obstacle results moving less in the direction perpendicular to the *DPL*, consequently robot can move back to *DPL* faster and easier. The "easier way" is judged not only by the relative velocity but also the relative position between the robot and the moving obstacle, both in *DPL* coordinates. In Fig 8 for example, if the robot turns left to avoid moving obstacle instead of turning right, the robot has to either start moving right earlier or faster to avoid the collision with the moving obstacle.

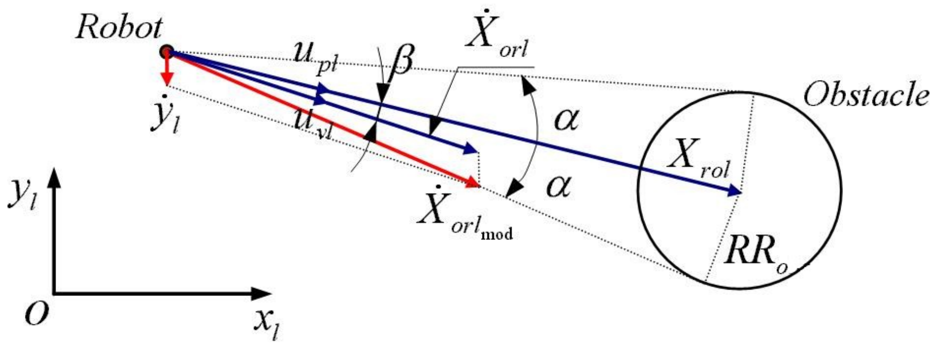


Figure 9. Changing $\dot{y}_l$ in Obstacle Avoidance

It will be more complicated in cases where there is not enough distance between the robot and obstacle when a robot starts to turn off the *DPL*, or a robot already has a velocity in the y_l direction. This is the multiple-obstacle avoidance case (Section 3.2.6).

3.2.3 Range of Checking Collision. An important issue is how to determine when the robot should start to leave the *DPL*. It will be easier to avoid collisions if the robot starts to turn off the *DPL* earlier and thus get more time to turn the relative velocity $\dot{\mathbf{X}}_{orl}$ out of the collision range. On the other hand, velocities of obstacles are not constant; it might not be necessary to turn off the *DPL* if obstacles change their velocity in later time steps. Therefore, we should keep the robot moving along the *DPL* as long as possible, especially for multiple moving obstacles.

Assume the maximum velocity $\dot{\mathbf{X}}_{obl\max}$ of an obstacle is the same as the maximum speed of the robot $\dot{\mathbf{X}}_{rbl}$. We consider the special worst case when the obstacle moves towards the robot along the *DPL*, both at their maximum speed as shown in Figure 10.

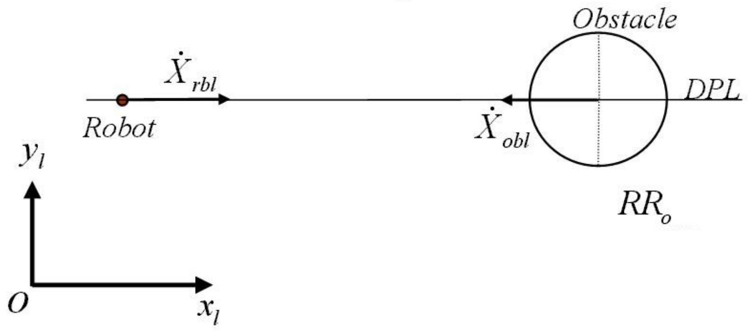


Figure 10. Worst-Case Collision Scenario

$\dot{\mathbf{X}}_{orl} = \dot{\mathbf{X}}_{rbl} - \dot{\mathbf{X}}_{obl} = 2\dot{\mathbf{X}}_{rbl\_m} \le 2\dot{x}_{l\max}$ is the relative speed between the robot and obstacle in this case. If the maximum velocity and acceleration of the robot in the y_l direction is $\dot{y}_{l\max}$ and $\ddot{y}_{l\max}$, the time needed to move the robot a distance RR_o in the y_l direction to avoid the collision is:

$$\Delta t = \begin{cases} \sqrt{(2RR_o / \ddot{y}_{l\max})} & RR_o < \dot{y}_{l\max}^2 / 2\ddot{y}_{l\max} \\ RR_o / \dot{y}_{l\max} + \dot{y}_{l\max} / 2\ddot{y}_{l\max} & RR_o \ge \dot{y}_{l\max}^2 / 2\ddot{y}_{l\max} \end{cases} \tag{10}$$

The distance between the robot and obstacle at which the robot should start checking collision is then

Dis$_{ov}$ from (11). The relative distance in this worst-case scenario is greater than the distance for all other cases, in which the relative speed along the relative distance will be smaller.

$$Dis_{ov} = \begin{cases} 2\dot{x}_{l\max}\sqrt{(2RR_o/\ddot{y}_{l\max})} + RR_o & RR_o < \dot{y}_{l\max}^2 / 2\ddot{y}_{l\max} \\ 2\dot{x}_{l\max}(RR_o/\dot{y}_{l\max} + \dot{y}_{l\max}/2\ddot{y}_{l\max}) + RR_o & RR_o \geq \dot{y}_{l\max}^2/2\ddot{y}_{l\max} \end{cases} \quad (11)$$

Since an obstacle can change velocity it is reasonable to use the relative distance *Dis*$_{ov}$ to set a range that the robot starts to check possible collisions. When the relative distance with an obstacle is longer than *Dis*$_{ov}$, the possibility of collision with an obstacle is not checked, and the robot will keep moving on the *DPL* until its relative distance with the obstacle is less than or equal to *Dis*$_{ov}$.

***3.2.4 Position for Returning to DPL*.** After passing by the obstacle, the robot must return to the *DPL*. Figure 11 shows the robot passing by an obstacle. Points *A*, *B* and *C* represent three typical situations. At point *A*, the robot has not yet passed by the obstacle, and it has to keep its velocity in the y_l direction, or possibly increase speed if the obstacle increases its velocity. Point *B* is the critical point wherein the relative velocity vector $\dot{\mathbf{X}}_{orl}$ is perpendicular to the relative position vector $\mathbf{X}_{rol}$ and the robot is just passing the obstacle. The robot cannot start to turn back to the *DPL* at point *B* because decreasing speed here may result in a collision with the obstacle in the next time step. Point *C* is the intersection of two tangent lines of the obstacle circle, one parallel to the relative velocity $\dot{\mathbf{X}}_{orl}$ and the other parallel to the y_l axis. Point *C* is safe since changing the speed of the robot in the y_l direction at point *C* does not cause a collision with the obstacle. The β angle between $\dot{\mathbf{X}}_{orl}$ and $\mathbf{X}_{rol}$ at point *C* is:

$$\beta_c = 3\pi/4 + \operatorname{atan2}(|\dot{\mathbf{X}}_{orl}(y)|, |\dot{\mathbf{X}}_{orl}(x)|)/2 \quad (12)$$

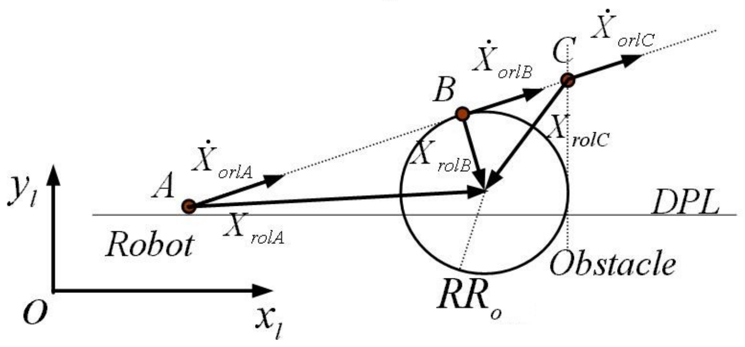


Figure 11. Position for Returning to *DPL*

It is possible for the robot to start returning to the *DPL* at any point between *B* and *C*. The robot can still collide with the obstacle if $\dot{y}_l$ changes too fast, or if the obstacle changes velocity. The relative velocity vector $\dot{\mathbf{X}}_{orl}$ from the point robot should not intersect the obstacle circle when the robot returns to the *DPL*. The robot has to keep moving with the same velocity in the y_l direction until $\beta \geq \beta_c$.

3.2.5 Motion in y_l Direction. The robot motion in the y_l direction is classified into three categories:

1. Increase (or decrease) $\dot{y}_l$ to achieve $\beta < \alpha$
2. Keep $\dot{y}_l$ when $\alpha < \beta < \beta_c$
3. Decrease (or increase) $\dot{y}_l$ to return to the *DPL* after it passes the obstacle.

When the relative distance with an obstacle is within Dis_{ov}, the robot starts to check collisions. It starts to increase (or decrease) the speed in the y_l direction if $\beta < \alpha$. The velocity in the y_l direction is conditionally calculated as:

$$\dot{y}_l = \begin{cases} \dot{y}_l + \ddot{y}_{l\max}\Delta t & \dot{y}_l < \dot{y}_{l\max} \ \& \ u_{vl}(y) > u_{pl}(y) \ \& \ \beta < \alpha \\ \dot{y}_l - \ddot{y}_{l\max}\Delta t & \dot{y}_l < \dot{y}_{l\max} \ \& \ u_{vl}(y) < u_{pl}(y) \ \& \ \beta < \alpha \\ \dot{y}_l & |\dot{y}_l| = |\dot{y}_{l\max}| \ \ or \ \alpha < \beta < \beta_c \end{cases} \tag{13}$$

If $\beta > \alpha$ but $\beta < \beta_c$, the robot will maintain its speed in the y_l direction until it reaches the position of returning to the *DPL* ($\beta = \beta_c$). The robot will not decrease its speed before that because the obstacle might change its velocity in the next step; this avoids zigzag motions in the y_l direction.

After reaching the position of returning to the *DPL* ($\beta = \beta_c$), the robot starts to decrease (or increase) its velocity in the y_l direction to return to the *DPL*. $\dot{y}_l$ is accelerated in the direction of returning to *DPL*. The direction of acceleration will be reversed when a robot is near the *DPL* so that $\dot{y}_l$ can end at zero when the robot reaches the *DPL* ($y_l = 0$). The velocity in the y_l direction after the robot reaches the position of returning to the *DPL* is conditionally calculated as follows:

$$\dot{y}_l = \begin{cases} \dot{y}_l + \ddot{y}_{l\max}\Delta t & \dot{y}_l < 0 \,\&\, |y_l - y_{lA}| > \dot{y}_l^2 / 2\ddot{y}_{l\max} \\ \dot{y}_l - \ddot{y}_{l\max}\Delta t & \dot{y}_l > 0 \,\&\, |y_l - y_{lA}| > \dot{y}_l^2 / 2\ddot{y}_{l\max} \\ \dot{y}_l + \dot{y}_l^2 / 2(y_l - y_{lA}) & |y_l - y_{lA}| < \dot{y}_l^2 / 2\ddot{y}_{l\max} \end{cases} \tag{14}$$

From Figure 11 we can see that angle β is changed from less than $\pi/2$ to greater than $\pi/2$ when the robot passes by an obstacle. The value of the dot product of $\mathbf{X}_{rol}$ and $\dot{\mathbf{X}}_{orl}$ will change from positive to negative when the robot passes by the obstacle.

3.2.6 Multiple Obstacle Avoidance. Multiple obstacle avoidance can be categorized into (1) avoiding collisions with obstacles one-by-one and (2) avoiding collisions at the same time. Group (1) cases can be handled sequentially with the above method. For Group (2) when the robot is avoiding an obstacle, other obstacles move within Dis_{ov}. The strategy used in multiple-obstacle avoidance is to determine the robot motion by checking all obstacles that are within Dis_{ov}. The position of leaving the *DPL* and robot motion in the y_l direction is determined by the obstacle that first appears within Dis_{ov}, with $\beta < \alpha$. When the second obstacle also has $\beta < \alpha$, the robot will modify its motion, trying to avoid collisions with both obstacles. The position of returning to the *DPL* is also affected by all obstacles within Dis_{ov} of the robot.

As an example, obstacle 1 is assumed to enter within Dis_{ov} first. The robot starts to increase the velocity in the y_l direction to force the relative velocity with obstacle 1 $\dot{\mathbf{X}}_{orl1}$ out of collision range determined by α_1. Then obstacle 2 enters within Dis_{ov}. At the moment shown in Figure 12, though $\dot{\mathbf{X}}_{orl1}$ is already out of the collision range, as $\dot{\mathbf{X}}_{orl2}$ is still in the collision range determined by α_2, the robot will keep increasing its velocity in the y_l direction until $\dot{\mathbf{X}}_{orl2}$ is out of the collision range.

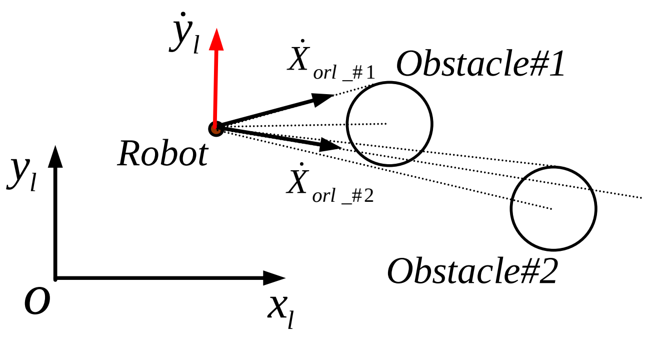


Figure 12. Multiple-Obstacle Avoidance

The multiple-obstacle avoidance algorithm is summarized as follows:

1. The x_l motion along the *DPL* is the same as that from single obstacle avoidance.
2. *Checking collision range:* For all moving obstacles with $\mathbf{X}_{rol} \cdot \dot{\mathbf{X}}_{orl} < 0$, if the relative robot/ obstacle distance is less than Dis_{ov}, the robot starts checking collisions.
3. *Leaving DPL position:* The robot leaves the *DPL* if any of the obstacles within the *checking collision range* meets the condition $\beta < \alpha$. The robot will increase $\dot{y}_l$ when $u_{vl}(y) > u_{pl}(y)$ relative to that obstacle, and decrease $\dot{y}_l$ in the opposite case.
4. If any of the obstacles within the *checking collision range* satisfies the condition $\beta < \alpha$, the robot will keep increasing (or decreasing) $\dot{y}_l$ until $\beta > \alpha$.
5. The robot will keep its velocity in the y_l direction if $\alpha \le \beta < \beta_c$ for all the obstacles within the *check collision range*.
6. *Returning to DPL*: When $\beta > \beta_c$ (for all the obstacles within the *check collision range*), and $|y_l - y_{lA}| > 0$, the robot will decrease (or increase) $\dot{y}_l$ to return to the *DPL* according to (14).

4. OBSTACLE AVOIDANCE SIMULATIONS AND EXPERIMENTS

4.1 Simulation and Experimental Setup

We developed a Simulink model for dynamic obstacle avoidance simulation. The methods of this article were used in conjunction with the developed controller (Figure 3) and coupled nonlinear dynamics model (3). Some details are missing in this article due to lack of space: all motions are subject to practical kinematic and dynamic constraints via novel *Velocity* and *Acceleration Cones* (Wu et al., 2006); these were implemented in Simulink to avoid velocities beyond the robot kinematic capabilities and to avoid accelerations leading to actuator saturation or wheel slippage.

The same Simulink model was then used to control the experimental mobile robot, using a PC with a Quanser MultiQ-3 board and Wincon 3.1 software, with cables for communication. The coupled nonlinear dynamics model (3) is replaced by the real robot hardware, but the same controller is used. A machine vision system senses in real time the position of the robot and obstacles (from which the translational velocities are derived from previous steps). Our three-wheeled omni-directional robot moves on a carpeted field as shown in Figure 13. A colored ball serves as the obstacle, moved by hand (via a long darkened handle, not shown) in the robot field. The hardware control diagram is Figure 14.

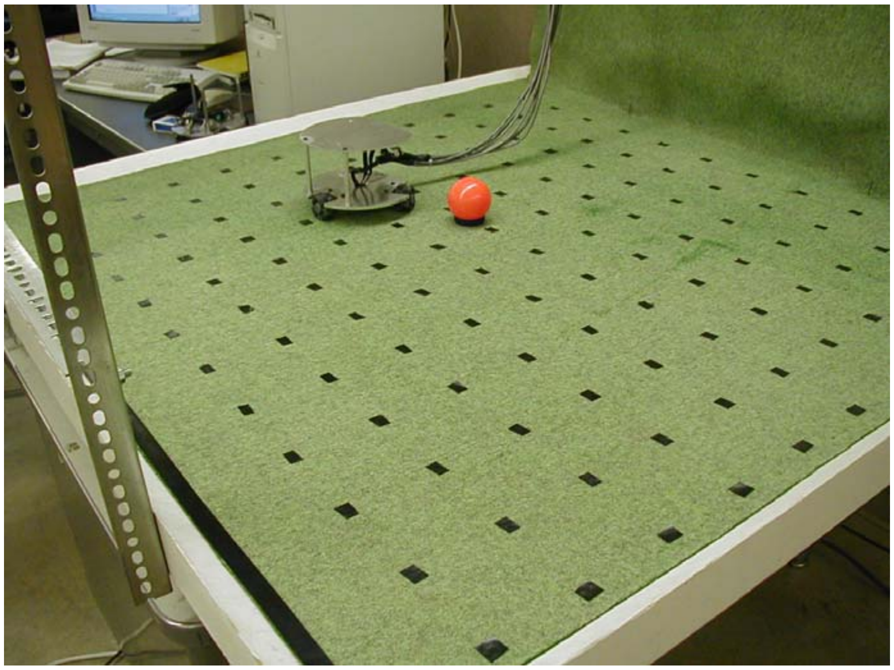


Figure 13. Experimental Setup with Omni-Directional Robot and Obstacle

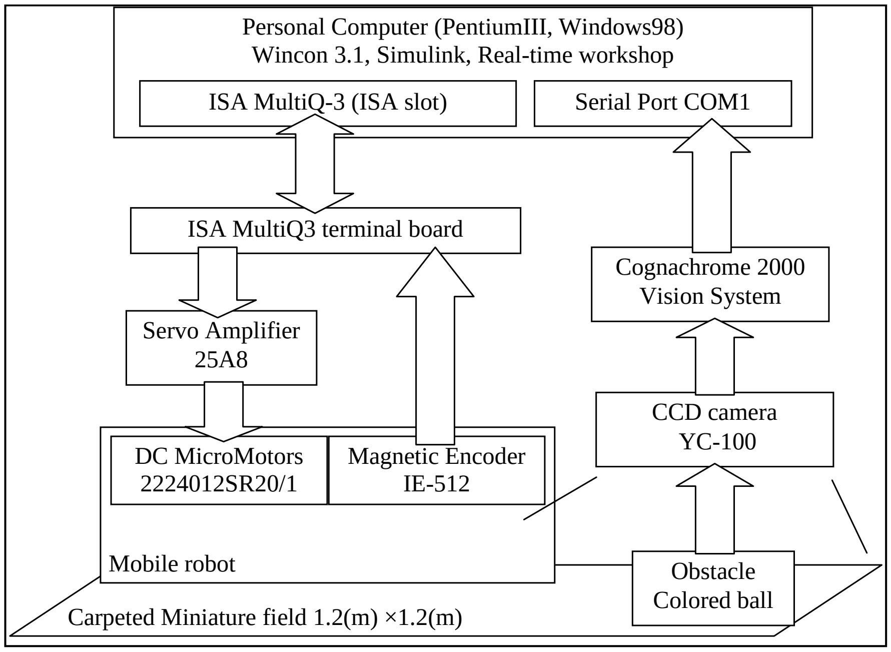


Figure 14. Hardware Control Diagram

The velocity of an obstacle is required as an input for the moving obstacle-avoidance algorithm. One way to obtain the velocity signal is to use a numerical differentiation formula or the Simulink differentiation block, but this leads to a noisy signal. Instead we applied a low-pass filter velocity estimation algorithm. As seen in Figure 15, this estimation method yields much smoother velocity for feedback than the differentiation block; this is an experimental result where the obstacle was moved by hand in a sinusoidal pattern.

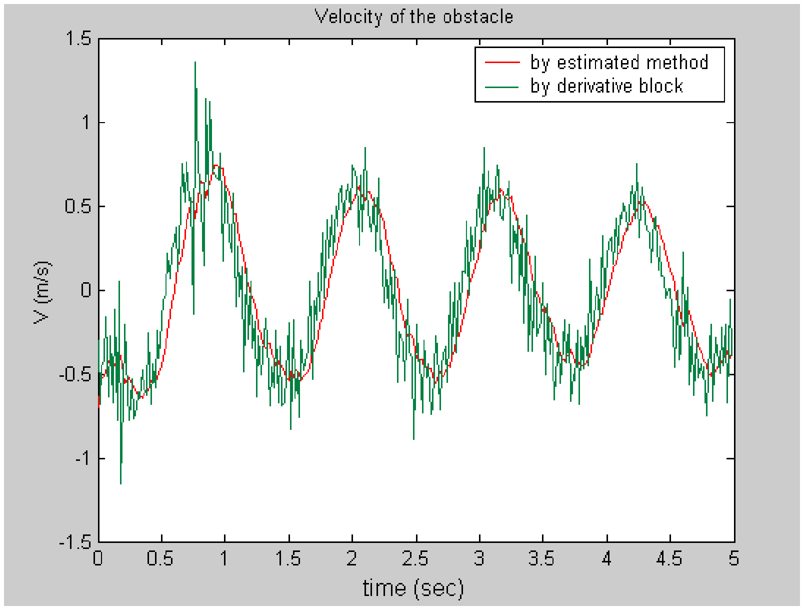


Figure 15. Experimentally-Sensed Obstacle Velocity

Next we present simulation and experimental validation examples for static and moving obstacle avoidance; we also present a simulation of multiple moving obstacle avoidance. For more examples and simulation/experimental validation, see Wu (2004).

4.2 Example 1: Static Obstacle Avoidance

In the simulation, the circular robot and obstacles are approximated by octagons to decrease the calculation requirement. Static obstacle avoidance is a special case of moving obstacle avoidance with zero obstacle velocity. Table I shows the static obstacle avoidance simulation data (in SI units).

Table I. Static Obstacle Avoidance Simulation Data

${}^{w}x_{rbA}$	${}^{w}y_{rbA}$	${}^{w}x_{rbB}$	${}^{w}y_{rbB}$	${}^{l}\dot{x}_{rb}$	${}^{l}\ddot{x}_{rb}$	${}^{w}x_{ob}$	${}^{w}y_{ob}$
0.95	0.05	0.05	0.9	0.6	1.5	0.44	0.6

Figure 16a shows the static obstacle avoidance simulation results with the conditions of Table I. The obstacle is placed at constant position $x_w = 0.44, y_w = 0.6$. The robot is commanded to move from starting point ($x_w = 0.95, y_w = 0.05$) to destination point ($x_w = 0.05, y_w = 0.9$) along a straight line with a motion pattern in which the robot accelerates to $v = 0.6 \ \ m/s$ with $a = 1.5 \ \ m/s^2$, maintain that speed for 2.5 seconds (by Equation 7), and decelerate to stop at the destination point.

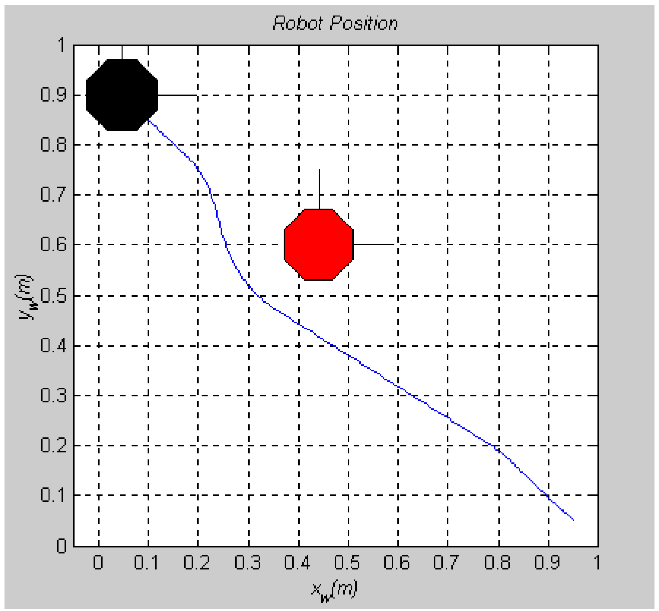


Figure 16a Simulated Static Obstacle Avoidance

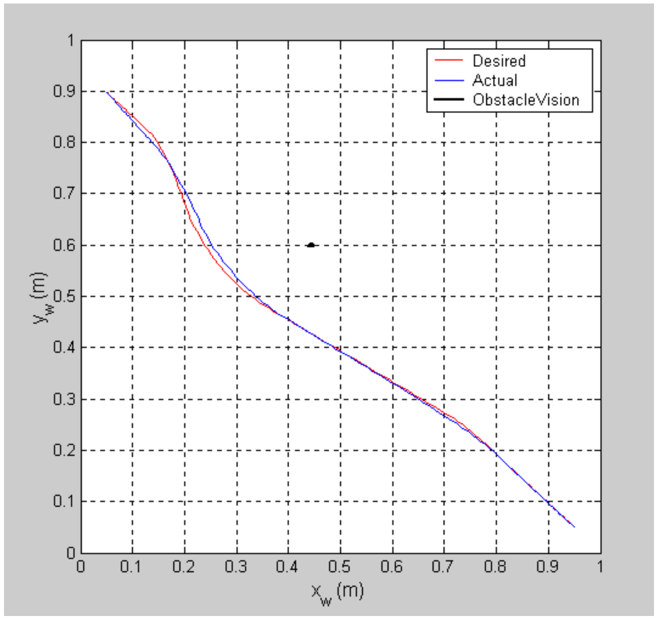


Figure 16b Experimental Static Obstacle Avoidance

A hardware static obstacle avoidance experiment was performed with the same conditions as the simulated case. The results are shown in Figure 16b. The actual path is compared with the commanded path (generated by the algorithm in real-time, not pre-planned), and the vision signal of the obstacle is displayed. We see that the simulated and experimental mobile robot paths are similar and both successfully avoided the static obstacle. In Figure 16b we also see that the actual path follows the commanded path well, indicating that our hardware controller development is effective to compensate for the wheel coupling and nonlinear dynamics effects. Also, the static obstacle position signal is steady from the machine vision system.

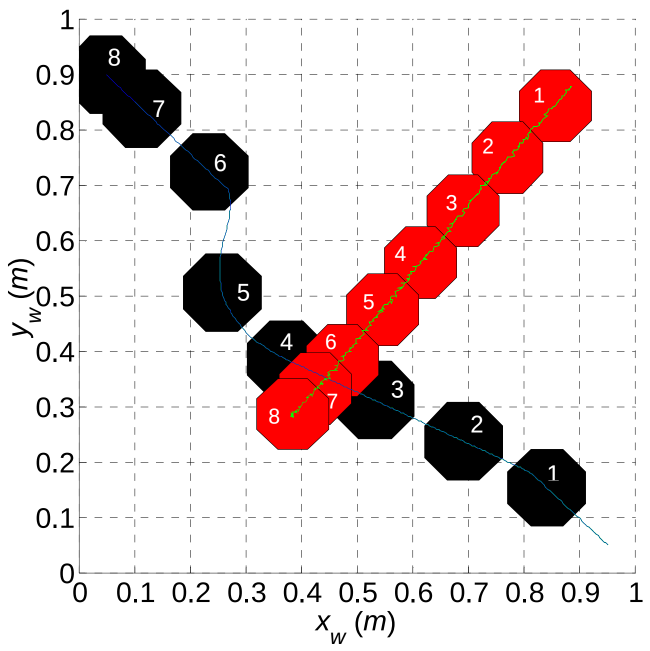
4.3 Example 2: Moving Obstacle Avoidance

Table II shows the data (in SI units) for moving obstacle avoidance simulation.

Table II. Moving Obstacle Avoidance Simulation Data

${}^{w}x_{rbA}$	${}^{w}y_{rbA}$	${}^{w}x_{rbB}$	${}^{w}y_{rbB}$	${}^{l}\dot{x}_{rb}$	${}^{l}\ddot{x}_{rb}$	${}^{w}x_{ob}$	${}^{w}y_{ob}$	${}^{w}v_{ob}$	${}^{w}\theta_{vob}$
0.95	0.05	0.05	0.9	0.6	1.5	0.88	0.88	0.20	$3\pi/4$

Figure 17a shows the moving obstacle avoidance simulation results with the conditions of Table II. The robot is commanded to move from starting point ($x_w = 0.95, y_w = 0.05$) to destination point ($x_w = 0.05, y_w = 0.9$) along a straight line with the motion pattern in which the robot accelerates to $v = 0.6$ *m/s* with $a = 1.5 (m/s^2)$, maintains that speed for 2.5 seconds (by Equation 7), and decelerates to stop at the destination point. The obstacle starts at ($x_w = 0.88$, $y_w = 0.88$) at $t = t_A$, with speed 0.2 (m/s) and direction $3\pi/4$ *rad* with respect to the x_w axis. The robot effectively avoids the moving obstacle by increasing the speed in the y_l direction. After the robot passes by the obstacle, it returns to the *DPL* and continues motion along it until the destination point. Sequential robot and obstacle positions in Figure 17a represent different snapshots in time, so we see there were no collisions in simulation (robot and obstacle motion is right to left in Figure 17a).



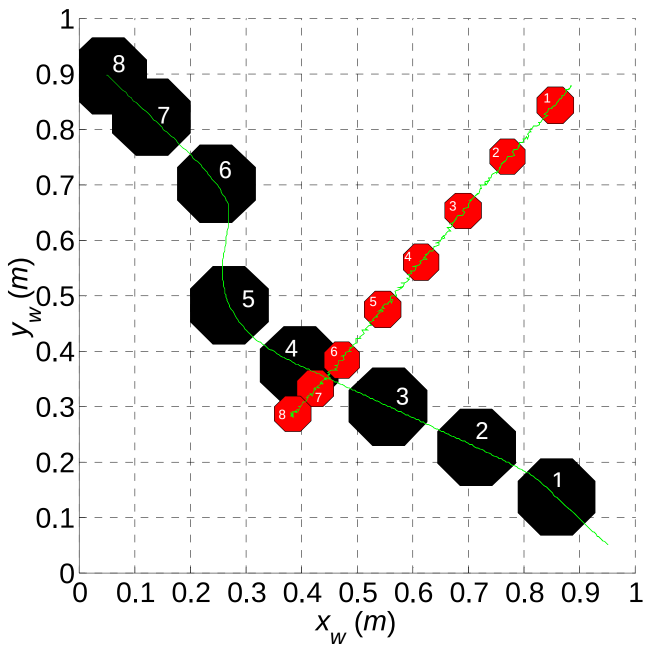


Figure 17a Simulated Moving Obstacle Avoidance   **Figure 17b Experimental Moving Obstacle Avoidance**

A hardware moving obstacle avoidance experiment was performed with the same conditions as the simulated case; the results are shown in Figure 17b. The actual experimental path on the right (generated by the algorithm in real-time, not pre-planned) compares well with the simulated path on the left. Since the experimental obstacle is moved by hand, it is difficult to exactly match the simulated obstacle motion; the vision signal of obstacle motion is displayed on the right. We see that the simulated and experimental mobile robot paths both successfully avoided the moving obstacle.

4.4 Example 3: Multiple Moving Obstacles Avoidance

As a final example we present a simulation of the mobile robot avoiding multiple obstacles. No experimental results are given in this case since our experimental obstacles are guided by hand. Three obstacles are arranged to get close to the *DPL* when the robot travels to a destination point on the field. We focus on a Group (2) case wherein more than one obstacle is in the check collision range at once.

Figure 18 is an example of multiple-obstacle avoidance simulation, in which the robot effectively avoids collisions with multiple obstacles. The data for simulation are in Table III (SI units).

Table III. Multiple-Obstacle Avoidance Simulation Data

${}^{w}x_{rbA}$	${}^{w}y_{rbA}$	${}^{w}x_{rbB}$	${}^{w}y_{rbB}$	${}^{l}\dot{x}_{rb}$	${}^{l}\ddot{x}_{rb}$	${}^{w}v_{ob1}$	${}^{w}\theta_{vob1}$	${}^{w}x_{ob2}$	${}^{w}y_{ob2}$	${}^{w}v_{ob3}$	${}^{w}\theta_{vob3}$
0.65	0.1	2.8	1.9	0.62	1.2	0.48	0.78π	1.8	1.1	0.3	1.34π

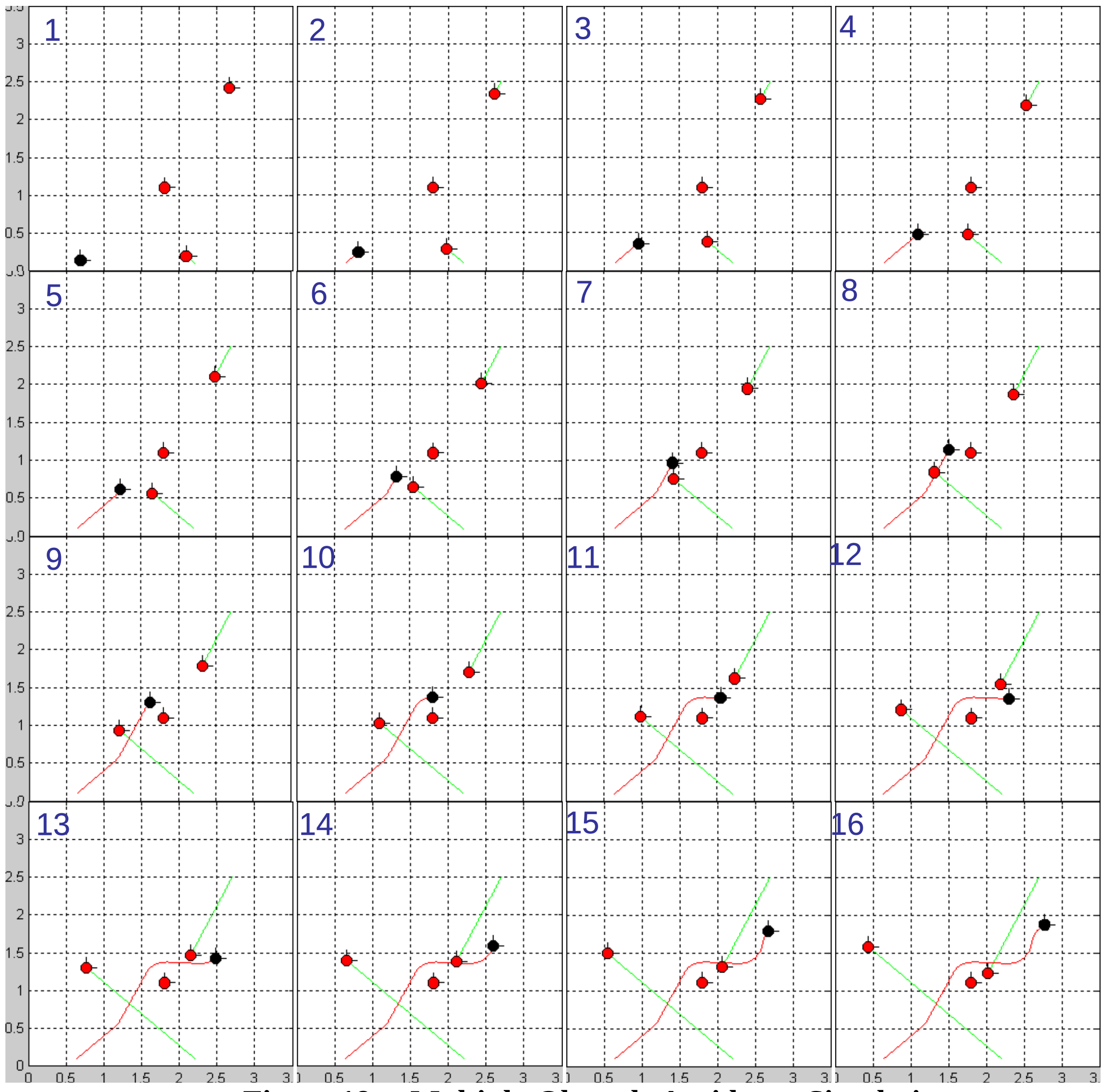


Figure 18. Multiple-Obstacle Avoidance Simulation

Obstacles 1 and 3 are moving, while obstacle 2 is static. As shown in Figure 18, the motion of the robot in the y_l direction is first determined by obstacle 1 (snapshots 5 and 6). Instead of returning to the *DPL*, the robot maintains its velocity after it passes obstacle 1 as obstacle 2 has already entered the *checking collision range* with $\beta < \alpha$ (snapshots 7, 8, and 9). The robot returns to the *DPL* after passing the static obstacle (snapshots 10 and 11). Instead of slowing down to return to the *DPL*, the robot passes through the *DPL*, as obstacle 3 enters the *checking collision range* with $\beta < \alpha$. The robot maintains the speed (snapshots 12 and 13) until it passes obstacle 3 (snapshots 14 and 15). Snapshot numbering is left-to-right and top-to-bottom.

This example is among those that the robot avoids collisions with multiple moving and static obstacles. However, successful obstacle avoidance is not always possible. Under some conditions, the robot is not able to realize multiple-obstacle avoidance by the developed algorithm; this is discussed in the next section on limitations of our methods.

5. LIMITATIONS OF THE DYNAMIC OBSTACLE AVOIDANCE METHOD

Assuming the obstacle is not so large as to violate the practical robot kinematic and dynamic constraints (or placed too close to the destination point) and that the experimental system is functioning properly, the static obstacle avoidance algorithm should work in all cases. In this section, we discuss the limitations of our moving obstacle avoidance algorithm. The algorithm is real-time, the obstacles' motions are not pre-known (we assume the obstacle motion limits are similar to the robot's), and obstacles can change velocities during motion.

5.1 Velocity and Acceleration Limits of the Obstacle

With the developed algorithm, it is possible that robot cannot avoid collisions if the velocity and the acceleration of the obstacle in the y_l direction are higher than those of the robot.

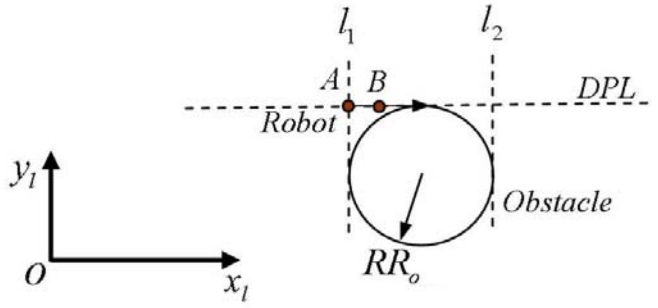


Figure 19. Limitation of the Obstacle Avoidance Algorithm

As shown in Figure 19, l_1 and l_2 are two tangent lines of the obstacle circle parallel to the y_l axis in *DPL* coordinates. Point *A* is the intersection of the *DPL* and l_1, and *B* is a point on the *DPL* between l_1 and l_2. Suppose that the velocity of the obstacle in the y_l direction is zero when the robot is at point *A*. The robot moves along the *DPL* as $\beta \geq \alpha$ at point *A*. When the robot is at point *B*, if the obstacle starts to increase its velocity in the y_l direction with an acceleration higher than the maximum acceleration of the robot, and maximum velocity in the y_l direction higher than that of the robot, the robot is not able to avoid a collision with the obstacle.

Failure of avoiding collisions with the obstacle is because the obstacle y_l motion ability is higher than that of the robot, and the motion of the obstacle is not pre-known. If the motion of the obstacle is pre-known, it is still possible to plan a path to let the robot reach the destination point within a fixed time while avoiding the collisions with the obstacle.

From the above discussion, we conclude that for single obstacle avoidance, the maximum velocity and maximum acceleration of the obstacle in the y_l direction should be lower than (or equal to) that of the robot, $\dot{y}_{obl\max} \le \dot{y}_{l\max}$ and $\ddot{y}_{obl} \le \ddot{y}_{l\max}$. Since the robot moves with constant x velocity for most of the time along the *DPL* (except near starting or destination points), the acceleration condition is rewritten as $\ddot{y}_{obl} \le a_{\max}$, where $a_{\max}$ is the maximum robot acceleration.

The first condition is rewritten as $\dot{y}_{obl\max} \le \sqrt{v_{\max}^2 - \dot{x}_l^2}$, where $v_{\max}$ is the maximum robot velocity and the planned robot velocity in the x_l direction is $\dot{x}_l$. If the obstacle moves perpendicular to the *DPL,* we have $v_{ob} \le \sqrt{v_{\max}^2 - \dot{x}_l^2}$ and $a_{ob} \le a_{\max}$, which can be considered as the limitation of the acceleration and velocity of the obstacle for algorithm success. In our simulation and experimental study (Section 4), we used $v_{ob} \le \dot{x}_{l\max} = \dot{y}_{l\max} = (\sqrt{2}/2)v_{\max}$. In real applications, if it is necessary to avoid collisions with obstacles having higher velocities than the robot, we must decrease the velocity of the robot in the x_l direction (thus extending the motion time period).

5.2 Failure Cases in Multiple-Obstacle Avoidance

If the velocity and acceleration of the obstacle are limited as in Section 5.1, the developed algorithm can effectively avoid collisions with single obstacles. However, for multiple obstacles, there are still some cases in which the robot cannot avoid collisions with the obstacle. One case is that the robot increases its velocity in the y_l direction to avoid collisions with one obstacle. When the robot is at its maximum velocity in the y_l direction, a second obstacle appears in the *checking collision range*. In order to avoid collisions with the second obstacle, the robot has to increase its velocity in the y_l direction, but the robot has already reached its maximum velocity. Another case is that when the robot has to increase its velocity in the y_l direction for avoiding the first obstacle, the second obstacle appears in the *checking obstacle range*, which requires the robot to decrease its velocity in the y_l direction. This also results in the failure of collision avoidance with the second obstacle. These failures are due to the fact that obstacles' motions are not pre-known and the robot motion along x_l is fixed due to the fixed time requirement. A path to the destination point in fixed time is not guaranteed when there are more than one obstacle within the checking collision range near the *DPL*. In such collision avoidance failures, the robot must halt, waiting until it can try for point *B* again, with a new starting point *A*.

5.3 Failure Cases Observed in the Experiment

Four failure modes were observed in the experiment. First, sometimes the robot cannot react to the motion of the moving obstacle; later we found (by monitoring the vision signal) this is because the vision system did not give the correct signal of the moving obstacle, as the hand and arm moving the obstacle is sometimes visible. This was largely eliminated after we paid attention to the vision range. Secondly, if we move the obstacle too fast, the robot will try to avoid the collision with the obstacle behind the motion of the obstacle. This is a correct reaction in the developed algorithm. However, the tethered wires of the robot interrupt the vision signal or sometimes interfere with a static obstacle, which results in failure cases. This is not an issue for untethered mobile robots. Thirdly, if we move the obstacle too fast (exceeding the velocity limitations discussed in Section 5.1), the robot could not completely avoid collisions. However, we still can see that the robot correctly reacts to the motion of the moving obstacle as best it can. Fourthly, if we move the obstacle differentially to hinder the robot for a relatively long time, the robot could possibly get no chance to reach the destination position, as the field is relatively small.

Because the experimental study for moving-obstacle avoidance was only performed for a single moving obstacle (the method handles any number of moving obstacles; up to eight have been simulated (not shown in this article)), few failure cases were observed. Certainly additional failure cases would arise in experimental multiple moving obstacle avoidance, which is planned for the near future.

6. CONCLUSION

A novel hybrid real-time approach has been developed for mobile robot motion planning focusing on moving obstacle avoidance. A global deliberate approach has been applied to the motion along the desired path line (*DPL*) while a local reactive approach is used to avoid the collisions with obstacles. The motions were subject to kinematic and dynamic constraints expressed by novel velocity and acceleration cones to avoid velocities and accelerations the robot cannot achieve due to kinematics, actuator saturation, and wheel slippage (these were not presented, see Wu et al., 2006). A coupled nonlinear dynamics model and controller were summarized for holonomic three-wheeled omni-directional mobile robots.

The reactive approach applied to local motion planning used relative velocity between the robot and obstacles to detect and avoid collisions. In addition, instead of using all the global information in the field, the collision detection only uses local information within a range surrounding the robot. Furthermore, with the developed approach, there is no trapping of the robot by local minima and no undesired influence by the obstacle when the robot passes by (these drawbacks are characteristic of the widely-applied potential field method of obstacle avoidance).

A fixed time problem has been implemented, i.e. the robot strives to reach its goal in a given time, while avoiding obstacles. The method generally handles any obstacle motions (as long as the motion characteristics of the obstacles are similar to the robot), which are not pre-known, and obstacle motion can change during the process. A machine vision system senses the obstacles' motions for use in the algorithm. We have identified limitations of our method, both algorithmic and experimental. Simulation with experimental validation have shown the effectiveness of the developed approach for moving obstacle avoidance by a mobile robot in an unknown changing environment.

REFERENCES

M.D. Adams, 1999, “High Speed Target Pursuit and Asymptotic Stability in Mobile Robotics,” IEEE Transactions on Robotics and Automation, 15(2): 230-237.

J.R. Andrews, and N. Hogan N, 1983, “Impedance Control as a Framework for Implementing Obstacle Avoidance in a Manipulator,” Control of Manufacturing Processes and Robotic Systems, Eds. Hardt, D.E. And Book, W., ASME, Boston, 243-251.

F. Belkhouche, 2009, “Reactive path planning in a dynamic environment”, IEEE Transactions on Robotics, 25(4): 902-911

J.V. der Berg and M. Overmars, 2007, "Kinadynamic motion planning on roadmaps in dynamic environments," Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst., San Diego, CA, October: 4253-4258.

A. Chakravarthy and D. Ghose, 1998, “Obstacle Avoidance in a Dynamic Environment: A Collision Cone Approach,” IEEE Transactions on System, Man and Cybernetics, 28(5): 562-574.

T. Y. Chang, S.W. Kuo, and Y.J. Hus, 1994, “A Two-Phase Navigation System for Mobile Robots in Dynamic Environment,” Proc. IEEE Int. Conf. on Intelligent Robots and Systems, Germany.

R.A. Conn and M. Kam, 1998, “Robot Motion Planning on N-Dimensional Star Worlds among Moving Obstacles,” Transactions on Robotics and Automation, 14(2): 320-325.

J.J. Craig, 2005, Introduction to Robotics: Mechanics and Control, 3rd Edition, Pearson Prentice Hall, Upper Saddle River, NJ.

P. Fiorini and Z. Shiller, 1998, “Motion Planning in Dynamic Environments Using Velocity Obstacle,” The International Journal of Robotics Research, 17(7): 760-772.

K. Fujimura and H. Samet, 1989. “A Hierarchical Strategy for Path Planning among Moving Obstacles,” IEEE Transactions on Robotics and Automation, 5(1): 61-69.

S.S. Ge and Y. J. Cui, 2000, “New Potential Functions for Mobile Robot Path Planning,” IEEE Transactions on Robotics and Automation, 16(5): 615-620.

J. Guldner and V.I. Utkin, 1995,”Sliding Mode Control for Gradient Tracking and Robot Navigation Using Artificial Potential Fields,” IEEE Trans Robotics and Automation, 11(2): 247-254.

T. Kalmar-Nagy, R. D’Andrea, and P. Ganguly, 2004, “Near-Optimal Dynamic Trajectory Generation and Control of an Omni-directional Vehicle,” Robotics and Autonomous Systems, 46: 47-64.

O. Khatib, 1985, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots,” International Conference on Robotics and Automation, St. Louis, Missouri, 500-505.

Y. Koren and J. Borenstein, 1991, “The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robot,” IEEE Transactions on Robotics and Automation, 7(3): 278-288.

C. Leng, C. Cao and Y. Huang, 2008, “A Motion Planning Method for Omni-directional Mobile

Robot Based on Anisotropic Characteristics," International Journal of Advanced Robotics Systems, Vol.5, No.4: 327-340

K.L. Moore and N.S. Flann, 2000, "A Six-Wheeled Omnidirectional Autonomous Mobile Robot," IEEE Control Systems Magazine, 20(6): 53-66.

F.G. Pin and S. M. Killough, 1994, "A New Family of Omni-directional and Holonomic Wheeled Platforms for Mobile Robots," IEEE Transactions on Robotics and Automation, 17(2): 480-489.

A. Tsoularis and C. Kambhampati, 1998, "On-line Planning for Collision Avoidance on the Nominal Path," Journal of Intelligent and Robot Systems, 21: 327-371.

N.C. Tsourveloudis, and K.P. Valavanis, and T. Hebert, 2001, "Autonomous Vehicle Navigation Utilizing Electrostatic Potential Field and Fuzzy Logic," IEEE Transactions on Robotics and Automation, 17(4): 490-497.

K. Watanabe, S. Yamamoto, S. G. Tzafestas, J. Tang, and T. Fukuda, 1998, "Feedback Control of an Omnidirectional Autonomous Platform for Mobile Service Robots," Journal of Intelligent and Robotic Systems, 22: 315-330.

R.L. Williams II, B. Carter, P. Gallina, and G. Rosati, 2002, "Dynamic Model with Slip for Wheeled Omnidirectional Robots," IEEE Transactions on Robotics and Automation, 18(3): 285-293.

J. Wu, R.L. Williams II, and J.Y. Lew, 2006, "Velocity and Acceleration Cones for Kinematic and Dynamic Constraints on Omni-Directional Mobile Robots", ASME Transactions on Dynamic Systems, Measurement, and Control, 128:1-12, December.

J. Wu, 2004, "Dynamic Path Planning of an Omni-Directional Robot in a Dynamic Environment", Ph.D. Dissertation, Ohio University, Athens OH.

F. Xu, H.V. Brussel, M. Nuttin, and R. Moreas, 2003, "Concept for Dynamic Obstacle Avoidance and their Extended Application in Underground Navigation," Robotics and Autonomous System, 42: 1-15.