J Intell Robot Syst (2006) 47: 341–360
DOI 10.1007/s10846-006-9080-2

Guidance-Based On-Line Robot Motion Planning for the Interception of Mobile Targets in Dynamic Environments

F. Kunwar · F. Wong · R. Ben Mrad · B. Benhabib

Received: 11 May 2006 / Accepted: 29 August 2006 /
Published online: 9 November 2006
© Springer Science + Business Media B.V. 2006

Abstract This paper presents a novel method for the interception of moving targets in the presence of obstacles. The proposed method provides simultaneous positional interception and velocity matching of the target moving in a dynamic environment with static and/or mobile obstacles. An acceleration command for the autonomous robot (i.e., interceptor) is first obtained from a rendezvous-guidance technique that takes into account the kinematic and dynamic limitations of the interceptor, but not the motion of the obstacles. This command is subsequently augmented, though only when necessary, in order to avoid those obstacles that are about to interfere with the time-optimal motion of the interceptor. The *augmenter* acceleration command is obtained in our work through a modified cell-decomposition method. Extensive simulation and experimental results have clearly demonstrated the efficiency of the proposed interception method, tangibly better than other existing obstacle-avoidance methods.

Key words moving-object interception · navigation-guidance · obstacle-avoidance · online trajectory planning · rendezvous-guidance

1 Introduction

Autonomous navigation in dynamic cluttered environments is the first and foremost requirement for *intelligent vehicles*. Intelligent vehicles are defined as those that can move and navigate autonomously in highway and/or urban traffic, in unstructured manufacturing environments, etc. In flexible manufacturing systems, for example, a common material-handling system is the Automated Guided Vehicle (AGV), used to transport tools and parts to and from workstations. Present day AGVs are guided by wires embedded below the

F. Kunwar (✉) · F. Wong · R. B. Mrad · B. Benhabib
Computer Integrated Manufacturing Laboratory,
Department of Mechanical and Industrial Engineering,
University of Toronto, 5 King's College Road,
Toronto, ON, Canada, M5S 3G8
e-mail: kfaraz@mie.utoronto.ca

surface or strips placed/painted on the surface and stop their motion when encountered with an obstacle [1]. Thus, their autonomy and efficiency could be tangibly increased, if they were capable of making on-line routing decisions in (mobile or static obstacle) cluttered environments.

In the above context, the focus of this paper is on two autonomy aspects of intelligent and autonomous vehicles: (a) guidance-based time-optimal interception and rendezvous with a moving target and (b) obstacle avoidance. In the following subsections, the most pertinent literature is reviewed prior to the overview of the proposed system in Section 2.

1.1 Guidance Based Interception

The literature on missile guidance can be traced to as early as 1870s, when the destruction of enemy vessels by guided torpedoes was proposed to the Prussian Ministry of War [2]. As the field has developed, missile-guidance techniques have been classified into five main categories [3]: Line-Of-Sight (LOS) guidance, in which the interceptor remains on a hypothetical line connecting the point of control to the target; Pure Pursuit (PP), in which the missile always heads towards the target; Proportional Navigation Guidance (PNG), in which the rate of rotation of LOS is kept zero while the distance is decreased; Optimal Guidance (OG), in which an objective function is optimized using optimal control theory such that interception is guaranteed; and, other guidance methods including the use of differential game theory. Missile guidance laws assume that the future trajectory of the target is completely defined either analytically, or by a probabilistic model [4–6].

The tools used to compute tactical missile guidance laws are similar to those used in dynamic motion planning. In particular, the PNG law uses the homing triangle for computing the acceleration of a missile pursuing an evading target. The homing triangle is defined by the interceptor, the target, and the point of interception. This control law makes the interceptor's acceleration normal to its path and proportional to the rate of change of the line of sight vector to the target. This ensures that the angle between the velocity vector of the interceptor and the LOS is constant. This in turn results in a homing triangle with fixed angles, which collapses to a point at interception. Due to its low computational requirements, simplicity of on-board implementation and time optimality characteristics, PNG has been the most widely used and researched guidance technique among all missile-guidance methods [7]. However, the problem of velocity matching introduced in this paper has not been an issue in missile-guidance applications.

The need for Rendezvous-Guidance (RG) methods arose from spaceship rendezvous missions with space stations and/or satellites. Rendezvous has been defined as the simultaneous matching of the position and velocity vectors of the autonomous vehicle with those of the target. A PNG-based RG method for the docking problem of two space vehicles was proposed in [8]. In [9], the use of exponential-type guidance, which is a special form of PNG-RG, was suggested for asteroid rendezvous. The problem of rendezvous with an object capable of performing evasive manoeuvres in order to avoid the rendezvous (as opposed to a non-evasive *friendly* target) was addressed in [10].

The utilization of a guidance-based technique in robot motion planning, with the purpose of improving upon the interception time achievable by visual-servoing techniques, was first reported in [11–14]. Although, these works showed that guidance-based methods could yield shorter interception times compared to other available techniques, all were limited to environments with no static or dynamic obstacles. Thus, this paper proposes a new guidance-based method that overcomes this limitation by modifying the RG law and enabling it to optimally deal with obstacles.

1.2 Motion Planning and Obstacle Avoidance for Mobile Robots

In the early 1960s, manufacturers such as General Motors began to use robot manipulators to aid them in various manufacturing tasks such as welding. A common way of specifying the movement of a robot was using a pendant to manually move the robot first through the required motions and teaching it collision-free paths. As tasks became more complicated, this method of path planning was too difficult and time-consuming. In response, a simple collision-avoidance method was proposed in [15]. It used a visibility graph to find a collision-free path in the workspace. A visibility graph is a roadmap method in which the vertices of the obstacles, nodes, are connected to each other if they are not obstructed. The shortest path is, then, found heuristically by searching through these nodes. A series of papers addressing the problem of 'piano movers,' were published after this work [16–18], where a collision-free path through a set of polygonal obstacles is found without the use of heuristics. The same problem was addressed in [19] as an optimal-control problem by minimizing the distance functions between potentially colliding parts.

One of the first mobile robots that could navigate through its surroundings autonomously was reported in [20] in 1967. Starting in early 1980s, mobile robotics became a common research theme and many papers have been published since. As one of the early examples, the outdoor land vehicle proposed in [21] used a navigation system that incorporated the use of sensor data and pre-loaded global maps. The autonomous mobile robot proposed in [22] used a collision-avoidance method that detected possible collisions and either slowed down or sped up the robot to avoid the collisions.

Starting in 1990s, collision-avoidance techniques matured and evolved to yield algorithms that are "feasible, believable, and practical" [23]. Many of the individual techniques for static-obstacle avoidance methods have been classified into three main categories: roadmap methods (visibility and Voronoi graphs), potential field techniques, and cell decomposition methods [24–26]. In Cell Decomposition (CD), the focus of this paper, the workspace is divided into cells that represent 'obstacles' and 'free space.' A path is the constructed by connecting the cells that guide the robot to its goal: in Exact CD (ECD), the workspace is divided into polygons. The union of the free cells defines exactly the free space; in Approximate CD (ACD), square cells are used to approximate the workspace. Each square may be 'filled' or 'empty,' depending on the presence of an obstacle in each square cell. ACD is further divided into three types: regular grid, quadtree, and framed-quadtree [27, 28].

One major advantage of ECD over ACD, and potential-field techniques, is its robustness in finding a solution: ECD guarantees a solution when one exists. The major computational requirement of ECD, however, is the triangulation or division of the workspace. The solutions of ACD are similar to those of the visibility graphs, where the robot travels very near the edges and vertices of the obstacles, while ECD yields a path through the midpoints on the edges that separate obstacles. Thus, the robustness of ECD is in its ability to guarantee a solution that is safer than that of ACD.

2 System Overview

A schematic diagram of the proposed implementation of RG and Modified ECD (MECD) method is shown in Figure 1. A vision module first obtains the positional and velocity $(\boldsymbol{p}, \boldsymbol{v})$ state vectors, respectively, of all the objects in the workspace, namely, the robot's $(\boldsymbol{p_R}, \boldsymbol{v_R})$, the target's $(\boldsymbol{p_T}, \boldsymbol{v_T})$ and the obstacles' $(\boldsymbol{p_{O1}}, \boldsymbol{v_{O1}})$, $(\boldsymbol{p_{O2}}, \boldsymbol{v_{O2}})$, etc., and passes this information to the RG and MECD algorithms. The RG method first determines the maximum

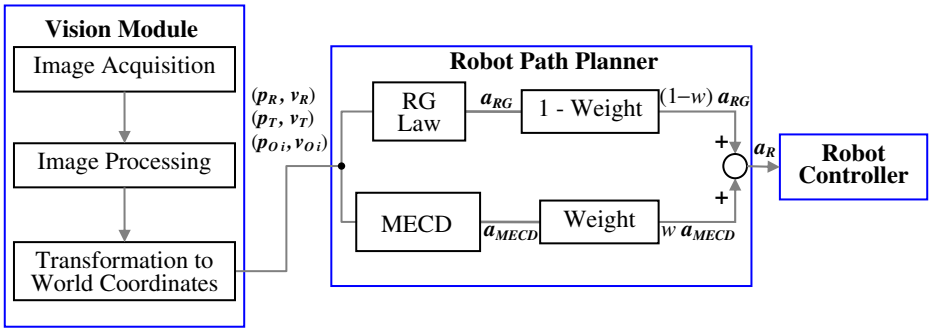

Figure 1 Schematic of proposed hybrid RG-MECD method.

allowable closing velocity component, which represents the desired velocity to be achieved by the robot in the next time instant, and then computes the acceleration command to achieve this velocity, $\boldsymbol{a_{RG}}$. In parallel, the MECD acceleration command, $\boldsymbol{a_{MECD}}$, is obtained. A weighting factor, w, is used to combine the two accelerations. When the robot is far away from the obstacle, more emphasis is given to the acceleration command $\boldsymbol{a_{RG}}$, conversely, in close vicinity of the obstacle more emphasis is given to the acceleration command $\boldsymbol{a_{MECD}}$.

The robot controller receives the combined acceleration command, $\boldsymbol{a_R}$, and transforms it into the necessary right- and left-wheel velocities required to move the robot in the desired direction at the desired velocity. One should note, however, that the largest lateral acceleration which a mobile robot with a differential drive can achieve without causing slippage is dependent on the wheels' rotational speeds as well as the kinematic/dynamic model of the robot. If the acceleration command given by the path planner were to exceed the maximum allowable acceleration of the robot, the robot simply adopts the maximum feasible value.

The proposed algorithm is reiterated until rendezvous with the target is achieved with a matching velocity component. Sections 2.1 and 2.2 present a detailed discussion of the RG law and MECD method, respectively. Section 3 explains how the RG law and the MECD method are combined to obtain the final acceleration command for the robot.

2.1 Interception Using Rendezvous Guidance

A Line-of Sight (LOS) is defined as the relative position vector, $\boldsymbol{r}$, connecting the interceptor/robot to the target as shown in Figure 2. The parallel-navigation rule states that the direction of LOS should remain constant relative to a non-rotating frame, while the interceptor approaches the target. Namely, the relative velocity, $\dot{\boldsymbol{r}}$, between the robot and the target should remain parallel to the LOS, $\boldsymbol{r}$, at all times [2]. If this rule holds throughout the motion of the interceptor, the distance between the interceptor and the target would decrease until they collide. Furthermore, if the target moves with a constant velocity, parallel navigation results in global time-optimal interception.

The parallel-navigation law is expressed by the following two relationships:

$$\boldsymbol{r} \times \dot{\boldsymbol{r}} = 0, \tag{1}$$

and

$$\boldsymbol{r} \cdot \dot{\boldsymbol{r}} < 0. \tag{2}$$

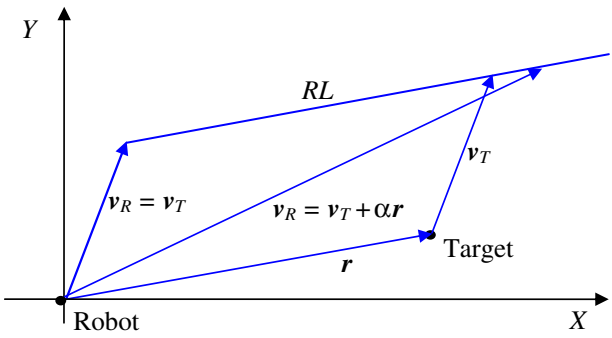

Figure 2 Construction of Rendezvous Line (*RL*).

Equation (1) guarantees that the LOS and relative velocity remain parallel, while equation (2) ensures that the interceptor is not receding from the target. The above equations can be solved for $\dot{\boldsymbol{r}}$ in a parametric form to yield

$$\dot{\boldsymbol{r}} = -\alpha \boldsymbol{r}, \tag{3}$$

where α is a positive real number. The instantaneous relative velocity, also referred to as 'closing velocity,' can be, then, written in terms of the robot and object velocities, denoted by $\boldsymbol{v_R}$ and $\boldsymbol{v_T}$, respectively, as follows:

$$\dot{\boldsymbol{r}} = \boldsymbol{v_T} - \boldsymbol{v_R}. \tag{4}$$

Substituting equation (3) into equation (4) and solving for the robot velocity yields:

$$\boldsymbol{v_R} = \boldsymbol{v_T} + \alpha \boldsymbol{r}. \tag{5}$$

The goal of the trajectory planner is to obtain a robot-velocity command for the next command instant, according to the parallel-navigation law. The vectors $\boldsymbol{r}$ and $\boldsymbol{v_T}$ are determined based on data received from a vision module based on the instantaneous positions of the robot and the target. Substituting these two known vectors into equation (5) would result in a locus for the robot's velocity vectors, $\boldsymbol{v_R}$, all lying on a semi-line parameterized by α. This semi-line, referred to as the Rendezvous Line (*RL*), is depicted in Figure 2. The centre of the frame of coordinates is located on the robot to show the instantaneous relative position of the target. The end-points of the velocity vectors show the position of the target or the robot, after one unit of time has passed, should they adopt the corresponding velocities. If the robot continually adopts a velocity command that falls on the instantaneous *RL*, the direction of LOS would remain constant and positional matching between the robot and the manoeuvring target is guaranteed.

Unlike in missile-guidance applications, in order to rendezvous with a target, the velocity of the robot/interceptor must also match the velocity of the manoeuvring target at the time of interception. The velocity commands generated based on equation (5) guarantee position matching. Thus, the next task is to find an α value such that velocity matching is also assured. An early attempt to match the target velocity may unnecessarily increase the interception time. Thus, it is desirable to determine the maximum allowable closing velocity without violating the velocity-matching condition.

Let us assume that from the current instant until interception, the robot is guided by velocity commands that lie on the instantaneous *RL*. This assumption allows us to consider the interception problem only in the direction of LOS. Let us, furthermore, consider that the

acceleration capability of the robot in this direction is given by A. This acceleration would be used to bring the closing velocity down to zero. Assuming a constant acceleration for the rest of the robot motion, the simultaneous reduction of velocity and position differences in the direction of LOS for interception may, then, be written as

$$\begin{cases} \dot{r}_{max}^{rend} - At_r = 0, \\ r - \dot{r}_{max}^{rend} t_r + {}^1\!/_2 A t_r^2 = 0 \end{cases}, \tag{6}$$

where $\dot{r}_{max}^{rend}$ is the magnitude of the maximum allowable closing/rendezvous velocity (hence, the superscript *rend*), and t_r is the time remaining to intercept the target from the current instant. The maximum instantaneous allowable closing velocity is then obtained by solving equation (6),

$$\dot{r}_{max}^{rend} = \sqrt{2rA}. \tag{7}$$

The maximum closing velocity as imposed by the frequency of velocity command generation by the trajectory planner for a fast asymptotic interception is given by

$$\dot{r}_{max}^{cr} = {}^r\!/_{n\Delta t}. \tag{8}$$

The value of n is determined experimentally. The final allowable closing velocity component of the velocity command is, then, obtained by considering equations (7) and (8) simultaneously:

$$v_{max}^{rel} = \min\left\langle \dot{r}_{max}^{rend}, \dot{r}_{max}^{cr} \right\rangle. \tag{9}$$

The end points of all velocity command vectors on *RL* that have a closing-velocity component smaller than v_{max}^{rel} constitute a line segment extending from $\boldsymbol{v_R}=\boldsymbol{v_T}$ to $\boldsymbol{v_R} = \boldsymbol{v_{R,max}}\left(= \boldsymbol{v_T} + \boldsymbol{v}_{max}^{rel}(\boldsymbol{r}/\|\boldsymbol{r}\|)\right)$. This set of points is referred to herein as the Rendezvous Set (*RS*), Figure 3a.

The velocity represented by v_{max}^{rel} (Figure 3a) may not be achievable by the robot within the time interval Δt. Therefore, we define a feasible velocity region representing all velocities achievable by the robot within the time interval Δt, taking into account the

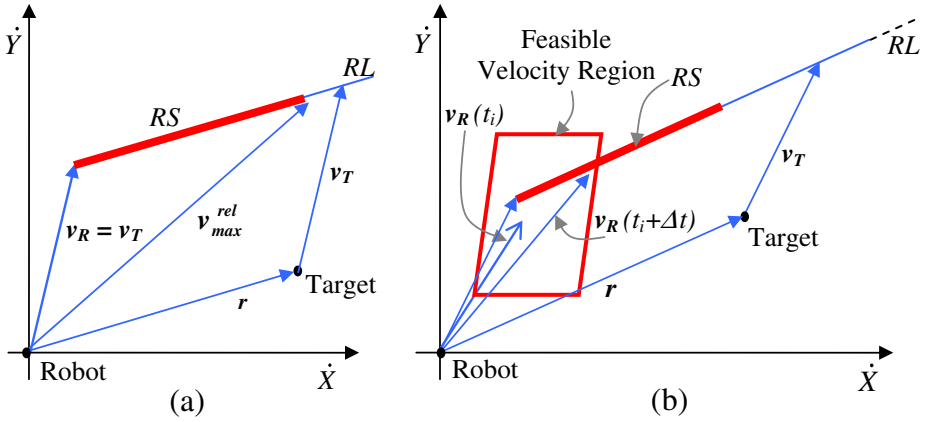

Figure 3 Rendezvous set and the generation of velocity commands.

kinematic and dynamic constraints on the robot, [29]: this region is depicted by the polygon in Figure 3b. The velocity selected by the robot for the time interval Δt is the component of the *RS* within the feasible velocity region with the maximum value, which is represented by $\boldsymbol{v_R}$ $(t_i+\Delta t)$ in Figure 3b. It is, thus, concluded that if the robot adopts velocity commands from within the *RS* with the largest allowable closing velocity component, then, a time-efficient interception can be achieved.

The overall generation of velocity commands from the rendezvous-guidance algorithm can be summarized as follows, Figure 3b: (1) Receive the instantaneous state of the object and the robot, (2) determine the feasible velocity region of the robot for the current state, (3) construct the *RL* obtain the maximum closing velocity, (4) construct the *RS*, and finally, (5) determine the robot velocity based on the intersection of *RS* and feasible velocity region. This process is repeated until both position and velocity matching errors are reduced to within the specified tolerances.

2.2 Obstacle Avoidance Using Cell Decomposition

The two most common static obstacle-avoidance methods that have been utilized in robotic applications are the Exact Cell Decomposition (ECD) and Potential Field (PF) methods. Both examine a single snapshot of the workspace and devise an optimal collision-free path to the target. Since obstacle and target movements are not factored into the calculations, these methods only perform well in finding a collision-free path based on the current positions of the robot, obstacles, and target. The planned path is usually the shortest one that is within the constraints of the algorithm. When used in a dynamic environment, the time-optimality of these methods quickly degrade. In this work, we consider only the ECD method, since it offers a more robust performance over the PF method by guaranteeing a solution when one exists [24]. However, since solutions provided by ECD may perform poorly under dynamic conditions, herein, the ECD method is first modified and, then, further augmented with a rendezvous guidance law for optimal-time interception and velocity matching.

2.2.1 Exact Cell Decomposition

Exact Cell Decomposition (ECD) divides the workspace into triangles (cells) and determines a collision-free path by connecting the midpoints of the edges of a series of cells, henceforth referred to as *nodes*, that link the robot to the target via the assignment of cost for travel between each node, Figure 4. Only the obstacles and the boundaries of the workspace are considered in the triangulation problem. The specific triangulation algorithm used in this work is based on the *bridge-building* operation first presented in [30]. The Dijkstra Algorithm [31] is subsequently used to search for the shortest path between the robot and the target from any node in the network.

2.2.2 Modified Exact Cell Decomposition

In conventional ECD, as noted above, the workspace is divided into triangles and nodes are placed at the midpoints of the edges of these triangles. One may note that if an obstacle were to be located far from the workspace boundaries, the edges would be very long. As a result, if the position of the robot or target were near an obstacle, the path generated would be unnecessarily long. Thus, the use of a Modified form of ECD (MECD) is proposed here.

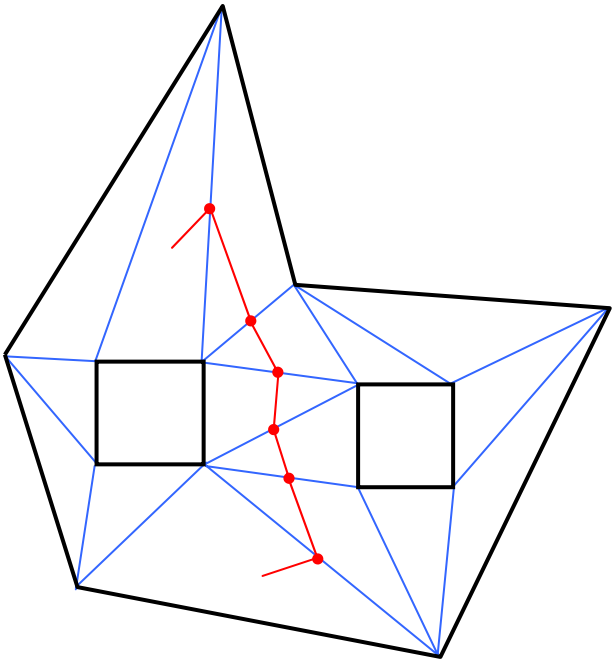

Figure 4 A possible path in a decomposed workspace using ECD.

MECD is similar to the Framed-Quadtree approach for the Approximate Cell Decomposition method [27, 28], where more than one node is made available on the edge of a cell. A *safety distance* is defined for MECD. Rather than having only one possible node on each line, at the midpoint of the line, another node is considered, at a safety distance from the obstacle, on the line. This is applied to both ends of the line. Therefore, on each line, there are, now, three available nodes that could be traversed. In the case where a edge is *short*, defined as being less than (3×*safety distance*), only the midpoint is used as a node. A robot path based on ECD is illustrated Figure 5a, while that of MECD is shown in Figure 5b.

3 Implementation

3.1 Trajectory Generation

The process of generating a robot trajectory using the proposed hybrid RG-MECD algorithm is depicted in Figure 6, where a target is moving with a velocity $\boldsymbol{v_T}$ in an environment having a single static obstacle that blocks the path of the robot.

In Figure 6, $\boldsymbol{v_R}$ represents the velocity of the robot at the current time instant t_i. First, in order to obtain the RG velocity component, $\boldsymbol{v_{RG}}$, the *Rendezvous Set* (*RS*) is constructed (line from $\boldsymbol{v_1}$ to $\boldsymbol{v}_{max}^{rel}$ as described in Section 2.1). The *Feasible Velocity Region* (*FVR*) for the robot is obtained next: this denotes all velocities achievable by the robot in the time interval $(t_i+\Delta t)$ shown by the parallelogram in Figure 6. As shown in Figure 6, the velocity $\boldsymbol{v}_{max}^{rel}$ lies outside the *FVR*, thus, this velocity cannot be achieved by the robot within the time interval $(t_i+\Delta t)$. Therefore, we have to select another velocity from within the *RS* for the time interval $(t_i+\Delta t)$. As shown in Figure 6, the *Rendezvous Line* (*RL*) intersects the *FVR* at two points designated by $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$. The set of points on the line bounded by $\boldsymbol{v_1}$ and

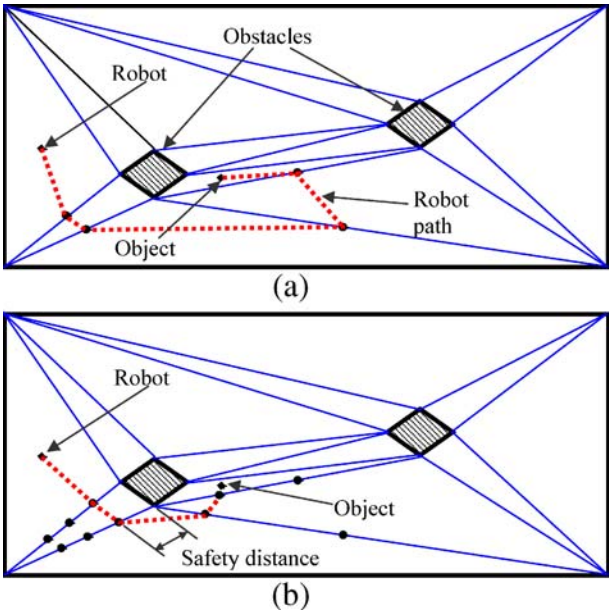

Figure 5 (a) Conventional ECD. (b) Modified ECD.

$\boldsymbol{v_2}$ represents the set of velocities that are both within the *FVR* as well as members of the *RS*, these set of points is referred to as the *Feasible Rendezvous Set* (*FRS*). By selecting a point from within *FRS* we would ensure a rendezvous with the target with the desired velocity component.

For time-optimal rendezvous, we select the maximum velocity from within *FRS* that takes the robot nearest to the target, which corresponds to $\boldsymbol{v_2}$ in this case. Thus, the required velocity to be achieved by the robot in the next time instant computed by the RG law is given by $\boldsymbol{v_{RG}} = \boldsymbol{v_R}(t_i + \Delta t) = \boldsymbol{v}_1$. The corresponding lateral acceleration command $\boldsymbol{a_{RG}}$ in order for the robot to achieve this velocity in the interval $(t_i+\Delta t)$ is expressed as

$$\boldsymbol{a_{RG}} = \frac{\boldsymbol{v_{RG}} - \boldsymbol{v_R}(t_i)}{\Delta t}. \tag{10}$$

However, as can be noted in Figure 6, if the acceleration command is computed purely via the RG law, the robot would collide with the stationary obstacle. Therefore, the acceleration command needed to avoid the stationary obstacle must be determined via the

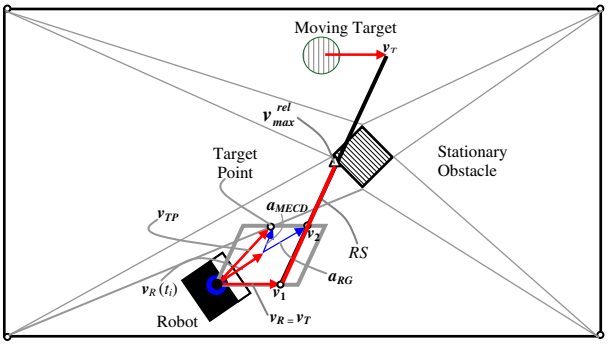

Figure 6 Desired velocities using MECD and without using MECD.

MECD method, as discussed in Section 2.2. First, we need to compute the velocity component based on the MECD method, $\boldsymbol{v}_{\boldsymbol{MECD}}$. As discussed in Section 2.2, the MECD algorithm provides a *target point* for the robot, defined here as the workspace location to which the robot is commanded to move during the next instant ($t_i+\Delta t$). This is represented by the velocity vector $\boldsymbol{v}_{\boldsymbol{TP}}$ in Figure 6. Thus, the velocity required in the next time interval Δt, in order to avoid the obstacle, is given by $\boldsymbol{v}_{\boldsymbol{TP}}$. It should noted that, in the case of static obstacles, the *target point* remains the same as long as the robot remains in a particular cell, since for static obstacles there is no change in the decomposition of the workspace with time. However, for dynamic obstacles, there would be a shift in the position of the target point for each time increment Δt. Therefore, for dynamic targets the *target point* needs to be tracked as well.

The lateral acceleration command, $\boldsymbol{a}_{\boldsymbol{MECD}}$, given to the robot to achieve $\boldsymbol{v}_{\boldsymbol{TP}}$ in the interval ($t_i+\Delta t$) is expressed as

$$\boldsymbol{a}_{\boldsymbol{MECD}} = \frac{\boldsymbol{v}_{\boldsymbol{TP}} - \boldsymbol{v}_{\boldsymbol{R}}(t_i)}{\Delta t}. \tag{11}$$

Thus, for each time instance, we have two acceleration commands: $\boldsymbol{a}_{\boldsymbol{RG}}$ that places emphasis on rendezvous with the target and $\boldsymbol{a}_{\boldsymbol{MECD}}$ that places emphasis on obstacle avoidance, respectively. The next issue to consider is how to obtain a feasible overall acceleration command which achieves both the above mentioned goals (rendezvous and obstacle avoidance) without compromising time optimality. If the path of the robot to the target computed by the RG Law is free of obstacles, one could proceed using just the $\boldsymbol{a}_{\boldsymbol{RG}}$ command, conversely, in close vicinity of obstacle one could use only the acceleration command $\boldsymbol{a}_{\boldsymbol{MECD}}$. Thus, herein, depending on the need for obstacle avoidance, a weight w is assigned to each acceleration command component in real-time, the procedure for assigning the weight w is discussed in succeeding paragraphs.

In order to achieve the best possible optimization, the path planner collects two pieces of information from the vision system: the distance and distance rate between the robot and obstacle(s). The *time-to-impact* is, then, determined by dividing the distance by its rate:

$$time - to - impact = \frac{d(t_i)}{d(t_{i-1}) - d(t_i)/\Delta t}, \tag{12}$$

where $d(t_i)$ is the distance between the robot and the obstacle at the current time instant and $d(t_{i\text{-}1})$ is the distance between the robot and the obstacle in the preceding time instant. Using the *time-to-impact* value, the acceleration command weight is calculated using a function such as the one in shown in Figure 7. During the implementation of the proposed algorithm, in order to ensure the safety of the robot, when the distance between the robot and obstacle was below 300 mm, w was forced to have a value of 1. Also, one can note that, the value of *time-to-impact* would be negative when the distance between the robot and the obstacle at the current time instant $d(t_i)$ is greater than the distance in the previous time instant $d(t_{i\text{-}1})$, namely, the robot is moving away from the obstacle. In such a case, the obstacle is no longer a threat and obstacle avoidance is not required. Therefore, the value of w is set to 0. By considering the above two cases, the final acceleration command is obtained via equation (13) by summing the weighted accelerations:

$$\boldsymbol{a}_{\boldsymbol{R}} = (1 - w) \times \boldsymbol{a}_{\boldsymbol{RG}} + w \times \boldsymbol{a}_{\boldsymbol{MECD}}. \tag{13}$$

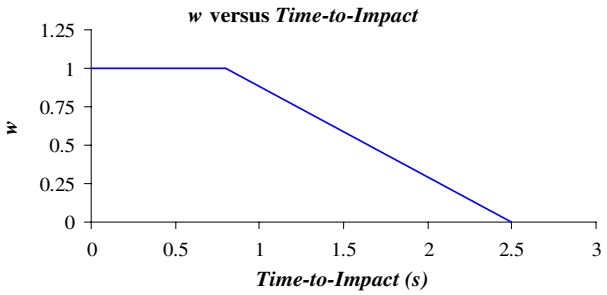

Figure 7 Acceleration command weight, *w*, as a function of *time-to-impact*.

The above procedure can be extended to multiple-obstacle scenarios, where the trajectory planner first computes the distance and distance rate for each obstacle in the environment. Based on these values of distance rate, the algorithm obtains the value of w for each obstacle. Finally, the acceleration command with the largest value for w is executed by the robot.

3.2 Robot/Interceptor Kinematics

In this sub-section, we briefly examine the kinematics of a mobile robot/interceptor with a differential drive system by considering a robot located at (x,y) turning to the right. Let $\boldsymbol{v_l}$, $\boldsymbol{v_r}$, and $\boldsymbol{v_R}$ represent the velocities of the left wheel, the right wheel, and the robot, respectively. Let d be the distance between the two wheels, and D be the distance between the right wheel and the Instantaneous Center of Curvature (*ICC*), Figure 8. The set linear velocity, $\boldsymbol{v_R}$, and angular velocity, $\omega_{\boldsymbol{R}}$, of the robot define the turning of the robot at a certain radius of curvature, R_C, specifying $\boldsymbol{v_l}$, and $\boldsymbol{v_r}$,

$$\omega_{\boldsymbol{R}} = \frac{\boldsymbol{v_R}}{d/2 + D}. \tag{14}$$

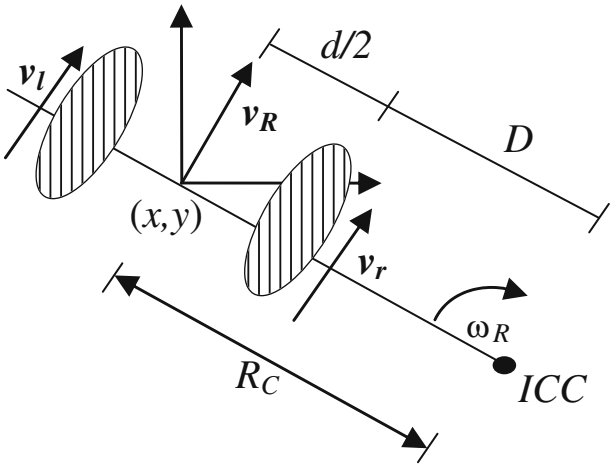

Figure 8 Robot kinematics.

The wheel velocities are then calculated as

$$\boldsymbol{v_R} = \frac{\boldsymbol{v_l} + \boldsymbol{v_r}}{2}, \tag{15}$$

$$\frac{\boldsymbol{v_r}}{D} = \frac{\boldsymbol{v_l}}{D+d}, \text{and} \tag{16}$$

$$\boldsymbol{v_l} = r_{wh}\omega_l, \boldsymbol{v_r} = r_{wh}\omega_r, \tag{17}$$

where r_{wh} is the radius of the wheel, and ω_l and ω_r are the angular velocities of the left and right wheel, respectively.

4 Simulations

A large number of computer simulations were carried out for interception with matching velocity using the RG-MECD algorithm. The maximum velocity and lateral acceleration of the robot was limited to 300 mm/s and 3000 mm/s^2, respectively. The robot is assumed to have no axial acceleration and is initially moving at 300 mm/s in the positive X direction. The criterion for successful interception was set as the achievement of <10 mm relative distance in both X and Y directions and achieving a relative velocity of <10 mm/s with that of the target. Results of two of simulations are shown in Table I.

The first simulation was carried out with static obstacles and a moving target. The motion of the target in this simulation is a straight line and the rendezvous trajectory of the robot is shown in Figure 9. The proposed trajectory planner uses MECD only when needed, therefore, the robot begins its trajectory by following the path generated by the RG law. As it, then, approaches the obstacle, the value of w increases and more emphasis is placed on the acceleration command given by MECD. Since obstacle 1 lies directly in the path generated by the RG law the value of w saturates to 1 as soon as the distance between the robot and obstacle becomes less than 300 mm, in this situation the entire acceleration command is obtained form the MECD method. As soon as obstacle 1 is navigated the distance between the robot and the obstacle begins to increase this causes the value of *time-to-impact* to become negative and w reduces to 0. Since obstacle 2 is not directly on the path generated by the RG Law a small weight w is given to the acceleration command generated by MECD for obstacle 2 during the entire manoeuvre. This example demonstrates by the ability of the proposed RG-MECD method of yielding an optimal path until obstacle avoidance is required.

The second simulation shows a highly dynamic environment in which both the obstacles as well as the target is moving, for this simulation, the motion of the obstacles is linear

Table I Interception data simulations

S/No.	Complexity	Obstacle velocity (mm/s)		Target velocity (mm/s)	Interception time (s)
		Obs. 1	Obs. 2		
1	Static obstacles – moving target	0	0	120	7.57
2	Moving obstacles – moving target	80	80	90	9.07

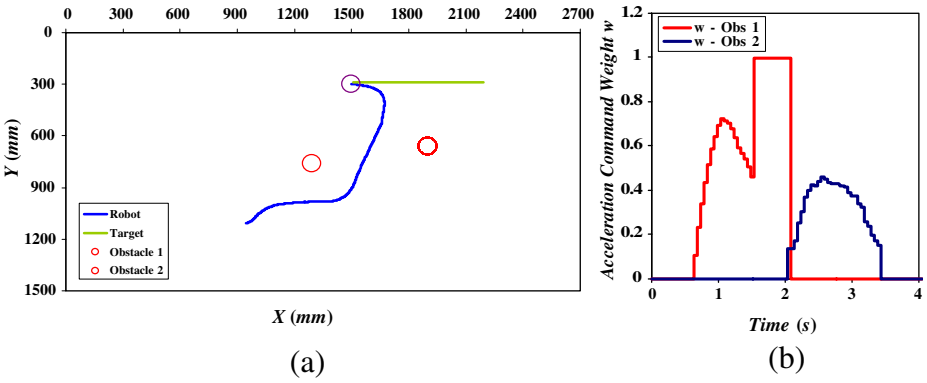

Figure 9 Simulation with static obstacles. (a) Robot trajectory. (b) Acceleration command weight *w*.

whereas the motion of target is circular as shown in Figure 10. In this figure, a time sequence of objects in the environment to show the time evolution of the avoidance manoeuvre is also shown. In this case, the position of each object at discrete time instants is shown. Each circle represents the location of the object after 1 s of motion. The shaded circles represent the location of each object at time instants of 0, 3, 6, 9 s of simulated time.

As shown in Figure 10, the robot initially moves directly towards the target based on the command generated by the RG Law, however, if this path is continued the robot will collide with obstacle 1. This is recognised by the planner and the value of *w* increases and obstacle 1 is avoided. Further the method also realizes that the optimal path lies by passing in front of obstacle 1 (instant 3). After obstacle 1 is negotiated the motion of obstacle 2 brings it directly between the target and the robot. This causes the value of *w* to increase gradually. At first the value of *w* is very small and the robot follows the command mainly generated by the RG law, however, as the robot approaches obstacle 2 the value of *w* increases up to the point where the distance between robot and obstacle reduces to less than 300 mm. At this point the value of *w* saturates and becomes 1. However, as soon as obstacle 2 is negotiated and the distance between the robot and the obstacle starts to increase the value of *w* reduces to 0. Here again the algorithm correctly assumes that the optimal path lies by passing behind the obstacle (instant 6). After avoiding the obstacles the robot matches the velocity with the target using the rendezvous guidance law until the desired position and

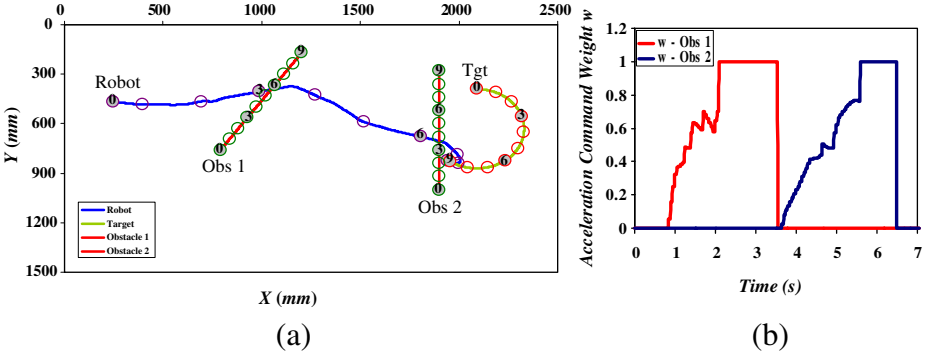

Figure 10 Simulation with dynamic obstacles. (a) Robot trajectory. (b) Acceleration command weight *w*.

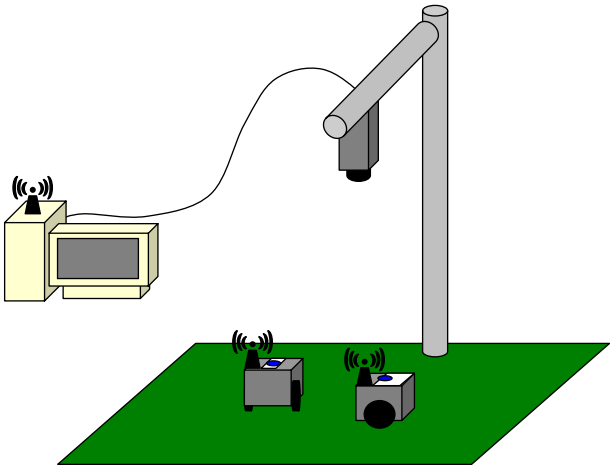

Figure 11 Physical layout of the setup.

velocity matching is achieved (instant 9). This example demonstrates the ability of the method to deal with a highly dynamic environment and to correctly follow the optimal path to rendezvous with a moving target.

5 Experiments

The physical layout of the experimental set-up is depicted in Figure 11 and the hardware specifications are given in Table II. The software for the experiments, running in a Pentium IV 1.6 GHz processor PC, consists of three modules: image acquisition and processing, trajectory planning, and communication modules, respectively. An analog CCD camera captures the image of the workspace and transfers it to the frame-grabber in the PC. The vision algorithm, then, extracts the positional information of all the objects in the workspace. This information is sent to the trajectory planner, where an acceleration command is calculated for the robot/interceptor. The communication module broadcasts this data to the mobile robots via a RF module connected to the PC through a serial port. Details of the vision system, communication system and mobile robots are included in Appendix A.

Table II Experimental hardware

Component	Characteristics
Two mobile robots	RF controlled
PC	Host computer with frame grabber and RF module
CCD camera	Resolution: 640×480 pixels
	Lens focal length: 6 mm
	Distance from floor: 3,000 mm
Floor workspace	2,740 mm×1,500 mm
	Surface material: felt

Table III Interception data experiments

E/No.	Complexity	Obstacle velocity (mm/s)		Target velocity (mm/s)	Interception time (s)
		Obs. 1	Obs. 2		
1	Static obstacle − moving target	0	0	120	5.04
2	Moving obstacle − static target	120	0	0	6.18

5.1 Experimental Results

In order to illustrate the real-time performance of the proposed algorithm, experiments were carried out with the aim to intercept a moving target without trying to rendezvous with it (for equipment safety reasons). A large number of experiments were carried out to verify the proposed implementation methodology. Each experiment was repeated three times under identical conditions (Table III and Figures 12 and 13). As shown in these figures, the behaviour of the (pursuing) robot in terms of performance and trajectory is very similar to the behaviour of the robot behaviour in the simulations. Experimental results (as did the simulations) reflect the ability of the algorithm to determine the optimal interception path.

In the first experiment reported herein, the object is moving in a straight line and the obstacles are static. In this case, as shown in Figure 12, the behaviour of the robot follows closely the behaviour observed in the simulations. In the second experiment, Figure 13, an obstacle is moving across the path of the robot and the target is static. This experiment shows the ability of the algorithm to correctly assume the optimal interception path in real-time. The robot starts of by proceeding directly towards the target using the RG Law; however, it realizes that this path will lead to a collision with the oncoming obstacle. Therefore, based on the weight w obtained form MECD the path of the robot is modified (only when required) to avoid the obstacle. The algorithm again correctly decides that the shorter path is obtained by passing in front of the obstacle rather than behind it.

6 Conclusions

A novel rendezvous-guidance method is proposed for autonomous robotic interception of moving targets in a dynamic environment with static and/or moving obstacles. The focus

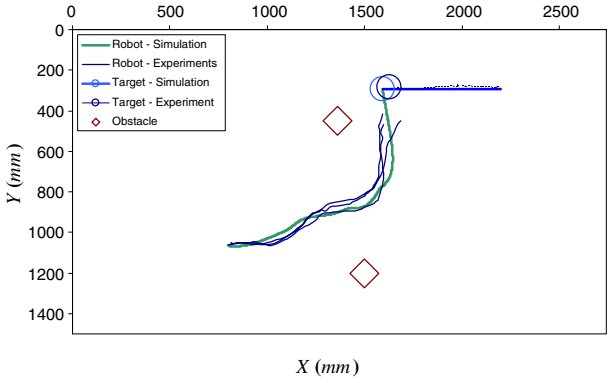

Figure 12 Experiment with static obstacles.

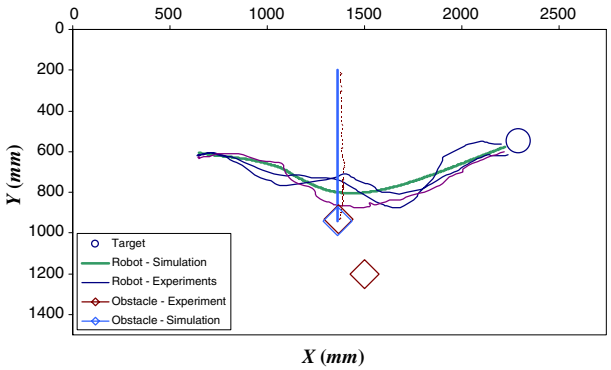

Figure 13 Experiment with a mobile obstacle.

has been on two autonomy aspects: (a) time-optimal interception and rendezvous with a moving target and (b) obstacle avoidance. The trajectory planner has two components: The first component is obtained from the RG law, which is used to track the target itself. The second component is obtained from the MECD method, which tracks the target points at teach time instant. Each component assigns a lateral acceleration command independent of the other. The final acceleration command, which is executed by the robot, is the sum of the two weighted acceleration commands based on *time-to-impact* with an obstacle.

Simulations and experiments have verified the system to be efficient and robust in regards to interception of moving targets with various different interception parameters and situations. It is to be noted that in order to reduce the errors between the perceived and actual state of the moving objects and further improve the robustness and performance of the algorithm, it can be enhanced with well established probabilistic methods for robot navigation e.g., use of particle filters.

Since the proposed method is based on well established guidance laws, it can be extended to multi-robot applications. An interesting extension to the proposed algorithm for future work could be to consider the application of the algorithm to intercept a set of multiple moving targets in which case acceleration commands for each target could be computed and executed based on a hierarchical framework depending on the tasks required to be performed by the interceptor. A similar method could also be used in a multi-robot intercepting system where a number of robots co-ordinate their movement based on the acceleration commands generated by the guidance law to surround and eventually intercept an evasive target.

Acknowledgement We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

Appendix A: Details of the Experimental Setup

A1 Vision System

The robot, the obstacles, and the target have markers that are colour-coded for identification. The raw image containing three channels of data, indicating the intensities of the Red (*R*), Green (*G*) and Blue (*B*) colours in each pixel is transformed into the *YCbCr* (luminance,

chrominance-blue, and chrominance-red) colour space. The transformation is performed by the following equations [31]:

$$Y = 0.299R + 0.587G + 0.114B \tag{18}$$

$$Cb = (B - Y)/1.772 \tag{19}$$

$$Cr = (R - Y)/1.402 \tag{20}$$

where Y has a range of [0, 255], and Cb and Cr have a range of [−127.5, 127.5].

When an image is examined, the weighted Euclidean distances, in the *YCbCr* colour space, between each pixel in the image and the predefined colour set are calculated:

$$D = \sqrt{0.15\left(Y_p - Y_c\right)^2 + 0.425\left(Cb_p - Cb_c\right)^2 + 0.425\left(Cr_p - Cr_c\right)^2} \tag{21}$$

where D is the weighted Euclidean distance, Y_p, Cb_p, and Cr_p, are measured *YCbCr* values of the pixel, and Y_c, Cb_c, and Cr_c are the values of the predefined colour set. It was noted that the pixels on the identification marker did not vary more than 18.0 in weighted Euclidean distance from the defined *YCbCr* value. This value was, therefore, set as the threshold distance: if a pixel is within this threshold distance of a certain colour in the predefined colour set, then it is considered to be that colour. After the image has gone through the thresholding operation, the positions of the mobile robot, the obstacles, and the target are determined.

A search is performed to find the markers on all the objects. To achieve the smallest sampling rate, the dimension of the smallest marker is used, denoted here as l pixels. Starting from the pixel location (0, 0), every $0.5l$ pixels are sampled along the X and Y directions. If the sampled pixel has the colour of the predefined set, a search frame is placed over that pixel. The size of the search frame is twice the diameter of the marker. If the number of pixels of a certain colour in the search frame exceeds a pre-determined threshold, then, a marker of that colour is considered to be located in that search frame. The centroid

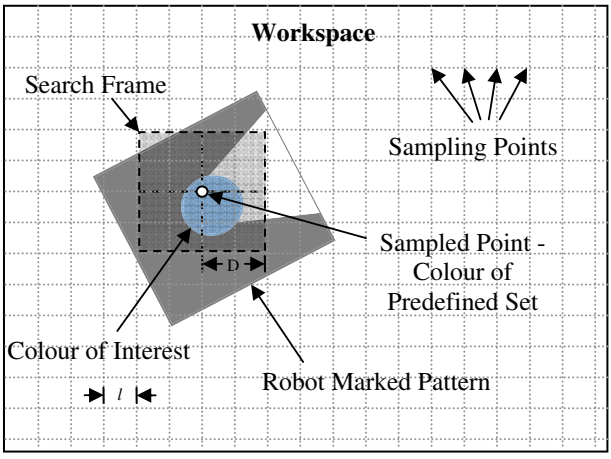

Figure 14 Colour marker search.

of that colour blob is, then, calculated to sub-pixel accuracy using the Centroid Method [32] (Figure 14).

With all the markers located, object identification can be performed. The vision program first searches for blue markers. Once a blue marker is found, the algorithm looks for a white marker within a distance of the radius of a robot. If a corresponding white marker is located, then, a robot has been successfully identified. Bearing of the object is indicated by an imaginary line drawn between the center of the blue circle to the centroid of the white pattern. The algorithm takes approximately 150 ms to execute (i.e., a frame-rate of 6.5 fps).

A2 Communication System

The communication system uses wireless transceivers to provide an asynchronous half-duplex link between the host PC and the mobile robots, [33]. The transmission baud rate operates at 19.2 kbps. Two carrier frequencies are available for this system: 866 MHz and 916 MHz. An independent wireless transceiver module is responsible for communication on the host PC. It is connected to the PC through the RS-232 serial port. The MAX232N chip converts the RS-232 signals into TTL signals which can, then, be transmitted by the transceivers. The transceiver modules include the TLP916 and RLP916 for transmitting and receiving, respectively.

A3 Mobile Robots

Two differential-drive mobile robots were used in the implementation of this paper: an interceptor and a moving obstacle or target. The mobile robots are powered by two Faulhaber 2842-012C DC-motors, rated at 12 V and provide 6.50 W of output power. Maximum torque is rated at 6.88 oz-in. A Faulhaber 38/3 spur gear-head provides a 5.42:1 reduction ratio to provide more torque to the wheels. This gear-head is in turn connected to the wheels through a 1:1 ratio gear assembly. Two ball casters provide balancing support to the robot. The top plate is marked with a coloured pattern.

A4 Robot Controller

The robot controllers utilize the Quanser QIC processor core. A baseboard was designed to house this processing unit and other auxiliary modules, such as the motor drivers and wireless transceivers. The controllers are powered by two 7.2 V Canon BP-511 Li-Ion batteries. The QIC Processor Core uses a Microchip PIC16F877 microprocessor and has a built-in RS-232 interface. It has 8 K of flash program memory, 10-bit A/D converter, and a built-in PWM controller. The processor is programmed using an embedded C++ language. The baseboard consists of three modules: transceiver unit, power unit, and motor drivers. Voltage regulators provide steady 12 V and 5 V for the transceiver unit and the motor drivers. The PWM driven motor drivers used were Allegro Microsystems 3959 series capable of delivering up to 3.0 A to each motor.

References

1. Yilmaz, A., Sami, O., Davis, R.P.: Flexible manufacturing systems: characteristics and assessment. Eng. Manage. Int. **4**(3), 209–212 (1987)

2. Shneydor, N.A.: Parallel navigation. In: Missile Guidance and Pursuit, pp. 77–99. Horwood, Chichester, England (1998)
3. Patrick, H.L., Seltzer, S.M., Warren, M.E.: Guidance laws for short-range tactical missiles. J. Guid. Control Dyn. **4**(2), 98–108 (1981)
4. Anderson, G.M.: Comparison of optimal control and differential game intercept missile guidance law. J. Guid. Control **4**(2), 109–115 (1981)
5. Ghose, D.: True proportional navigation with maneuvering target. IEEE Trans. Aerosp. Electron. Syst. **1**(30), 229–237 (1994)
6. Speyer, T.J., Kim, K., Tahk, M.: Passive homing missile guidance law based on new target manoeuvre models. Journal of Guidance **1**(13), 803–812 (1990)
7. Yang, C.D., Yang, C.C.: A unified approach to proportional navigation. IEEE Trans. Aerosp. Electron. Syst. **33**(2), 557–567 (1997)
8. Yuan, P.J., Hsu, S.C.: Rendezvous guidance with proportional navigation. J. Guid. Control Dyn. **7**(2), 409–411 (1993)
9. Guelman, M.: Guidance for asteroid rendezvous. J. Guid. Control Dyn. **14**(5), 1080–1083 (1990)
10. Jensen, D.L.: Kinematics of rendezvous manoeuvres. Journal of Guidance **7**(3), 307–314 (1984)
11. Piccardo, H.R., Hondered, G.: A new approach to on-line path planning and generation for robots in non-static environment. Robot. Auton. Syst. 187–201 (1991)
12. Mehrandezh, M., Sela, M.N., Fenton, R.G., Benhabib, B.: Robotic interception of moving objects using an augmented ideal proportional navigation guidance technique. IEEE Trans. Syst. Man Cybern. **30**(3), 238–250 (2000)
13. Borg, J.M., Mehrandezh, M., Fenton, R.G., Benhabib, B.: Navigation-guidance-based robotic interception of moving objects in industrial settings. J. Intell. Robot. Syst. **33**(1), 1–23 (2002)
14. Agah, F., Mehrandezh, M., Fenton, R.G., Benhabib, B.: On-line robotic interception planning using rendezvous-guidance technique. J. Intell. Robot. Syst.: Theory and Applications **40**(1), 23–44 (2004)
15. Pérez, T.L., Wesley, M.A.: An algorithm for planning collision-free paths among polyhedral obstacles. Commun. ACM **22**(10), 560–570 (1979)
16. Jacob, T.S., Micha S.: On the 'Piano Movers' problem. I: the case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. Commun. Pure Appl. Math. **36** (3), 345–398 (1983)
17. Jacob, T.S., Micha, S.: On the 'Piano Movers' problem. II: General techniques for computing topological properties of real algebraic manifolds. Adv. Appl. Math. **4**(3), 298–351 (1983)
18. Jacob, T.S., Micha, S.: On the 'Piano Movers' problem. III: Coordinating the motion of several independent bodies. The special case of circular bodies moving amidst polygonal barriers. Int. J. Rob. Res. **2**(3), 46–75 (1983)
19. Elmer, G.G., Daniel, W.J.: Distance functions and their applications to robot path planning in the presence of obstacles. IEEE J. Robot. Autom. **1**(1), 21–30 (1985)
20. Rosen, C.A., Nilsson, N.J.: Application of intelligent automata to reconnaissance (technical report). Stanford Research Institute (1967)
21. Parodi, A., Nitao, J., McTamaney, L.: An intelligent system for an autonomous vehicle. In: Proc. of IEEE Internat. Conf. on Robotics and Automation, San Francisco, CA, pp. 1657–1663, April 1986
22. Hu, H., Brady, M., Probert, P.: Navigation and control of a mobile robot among moving obstacles. In: Proc. of IEEE Conf. on Decision and Control, Brighton, England, pp. 698–703, December 1991
23. Cameron, S.: Obstacle avoidance and path planning. Ind. Rob. **21**(5), 9–14 (1994)
24. Latombe, J.C.: Robot Motion Planning. Kluwer, Boston (1991)
25. Hwang, Y. K., Ahuja, N.: Gross motion planning. ACM Comput. Surv. **24**(3), 219–291 (1992)
26. Baraquand, J., Langlois, B., Latombe, J.C.: Numerical potential field techniques for robot path planner. IEEE Trans. Syst. Man Cybern. **22**(2), 224–241 (1992)
27. Yahja, A., Stentz, A., Singh, S., Brumitt, B.L.: Framed-quadtree path planning for mobile robots operating in sparse environments. In: Proc. of IEEE Internat. Conf. on Robotics and Automation, Leuven, Belgium, pp. 650–655, May 1998
28. Chen, C., Danny, Z., Szczerba, R.J., Uhran, J.J. Jr.: A framed-quadtree approach for determining euclidean shortest paths in a 2-d environment. IEEE Trans. Robot. Autom. **13**(5), 668–681 (1997)
29. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. Int. J. Rob. Res. **17**(7), 711–727 (1998)
30. Seneviratne, L.D., Ko, W.-S., Earles, S.W.E.: Triangulation-based path for a mobile robot. Proc. Inst. Mech. Eng., C J. Mech. Eng. Sci. **211**(5), 365–371 (1997)
31. Noto, M., Sato, H.: A method for the shortest path search by extended Dijkstra algorithm. In: Proc. of IEEE Internat. Conf. on Systems, Man and Cybernetics, Nashville, TN, vol. 3, pp. 2316–2320, October 2000

32. Bourgin, D.: Color Space FAQ, http://www.neuro.sfc.keio.ac.jp/~aly/polygon/info/color-space-faq.html, August (2004)
33. Bose, C.B., Amir, J.: Design of fiducials for accurate registration using machine vision. IEEE Trans. Pattern Anal. Mach. Intell. **12**(12), 1196–1200 (1990)