Proceedings of the
44th IEEE Conference on Decision and Control, and
the European Control Conference 2005
Seville, Spain, December 12-15, 2005

Autonomous Motion of a Driverless Vehicle operating among Dynamic Obstacles

Toby J. Myers, Tony Noël, Michel Parent, *MIEEE*, Ljubo Vlacic, *SMIEEE*

***Abstract*—This paper presents a new approach to the problem of dynamic obstacle avoidance. Our developed approach, the Gradient Velocity Obstacle algorithm, has been designed specifically to operate in real-time on a non-holonomic car-like vehicle with a Ladar sensor being used for obstacle detection. The algorithm has been demonstrated both through simulation tests and operation on a test vehicle platform.**

I. Introduction

Autonomous motion of intelligent vehicles, specifically in environments containing dynamic road obstacles, is an exciting field of research. In order for success to be achieved in this field a number of areas of research must be integrated including sensor design, data fusion, map building, localization, off-line optimized path planning methods and real-time high-speed obstacle avoidance methods.

Due to dynamic obstacle avoidance being just one small segment amongst this large group of topics it is often overlooked. Also, its close link to the path planning problem often causes the solution required for this complex and distinct area to be underestimated. This is exemplified by the fact that most researchers make obstacle avoidance methods that are designed for operating in an a priori environment amongst static obstacles. They then believe that in order to make these methods operate in real-time among dynamic obstacles it is only a matter of increasing processing power and applying small modifications to achieve the same success. This is strongly refuted by Kohout [1] who criticizes current path planning methods that are largely computed off-line and unsuitable for application to a real-time scenario with dynamic obstacles.

As the current focus on dynamic obstacle avoidance is from a traditional path planning view, terminology currently used is from the view point of the vehicle's surrounding environment [2]. For example an environment is described as *dynamic* if obstacle information becomes known over time. This paper will however adopt a new terminology taken from the view point of the vehicle platform. Therefore rather than describing a *dynamic* environment, an algorithm would be referred to as operating in *real-time* by gathering details about the environment on-the-fly (after the algorithm has begun execution). Alternatively an *offline* algorithm uses a priori information about the environment to determine its maneuver prior to execution. This allows the attributes *dynamic* and *static* to refer to moving and stationary obstacles respectively. Also, to describe the motion constraints of a vehicle, the term *holonomic* is used to describe a vehicle capable of unconstrained motion such as turning on its axis and near instant acceleration within a specified (normally low) velocity range.

The organization of this paper is as follows. In Section II, areas of related work are discussed. Section III summarizes the Dynamic window approach. Section IV gives a brief explanation of the Velocity Obstacle approach. Our proposed method, the Gradient Velocity Obstacle Algorithm, is detailed in Section V. In Section VI, experimental results using both a simulated environment and an autonomous vehicle are presented. Finally, the conclusions can be found in Section VII.

II. Related Work

The separation of obstacle avoidance into different problems is a very controversial subject in which there are many differing opinions. Most agree that there are two approaches, global and local (though not always with these titles). This however is where the agreement ends and, due to the differing definitions applied by different authors, it is common to find solutions to the problem classified as global by one author and local by another. This confusion is increased by the common practice of combining local and global solutions to the problem together in an attempt to form a complete system.

For the purposes of this paper the following definitions of global and local will be used. *Global* solutions to the dynamic obstacle avoidance problem operate in a static environment by computing offline an optimized path from start to finish that avoids all known static obstacles. *Local* solutions use only a small fraction of the world space and operate real-time in an unknown environment. Therefore they have an inherent advantage in avoiding dynamic obstacles. However, they may also have the disadvantage of not being able to produce an optimal solution and may get trapped in local minima (such as a large U shaped obstacle) [2].

The first techniques designed specifically for avoiding

Manuscript received March 1, 2005. This work was supported, in part, by the French Government through the IMARA research unit of INRIA Rocquencourt. Support was also provided by Griffith University, Australia and its Institute for Intelligent and Integrated Systems.

T. J. Myers and L. Vlacic are with the Intelligent Control Systems Laboratory of Griffith University, QLD Australia (email: toby.myers@student.griffith.edu.au, l.vlacic@griffith.edu.au)

T. Noël and M.Parent are with INRIA, Rocquencourt, France (email: t.noel@inria.fr, m.parent@inria.fr).

0-7803-9568-9/05/$20.00 ©2005 IEEE

dynamic obstacles used various techniques to determine the point where a collision would occur between the vehicle and the obstacles. The techniques that were developed include configuration-time space [3], path-velocity decomposition [4], and collision fronts [5]. Configuration-time space operates by searching for a path in configuration space over time by estimating new positions of obstacles assuming constant velocities and unchanging direction. Path-velocity decomposition separated the control of velocity and direction, allowing velocity to be altered along the path in order to avoid obstacles that have been found to cross the chosen path. Collision fronts consists of creating an accessibility graph of the environment that is the set of the points on obstacles which are achievable by the vehicle moving at maximum speed [5].

Another technique for avoiding dynamic obstacles is the Velocity Obstacle approach [6]. Unlike the previous techniques, the Velocity Obstacles approach directly uses velocity information to determine which velocities will cause the vehicle to collide with an obstacle. As this method uses the velocities of the obstacles directly when calculating possible collisions it is ideal for real-time applications.

Before we discuss the details of the Velocity Obstacle approach, we will firstly, in Section III, describe the Dynamic Window approach. Although the Dynamic Window approach has not been specifically designed for controlling a vehicle's motion amongst dynamic obstacles it consists of an elegant step structure concept which eliminates unsuitable velocities until the vehicles best possible velocity is chosen. This step structure is also deployed in our algorithm in order to enhance the structure of the original Velocity Obstacle approach.

III. Dynamic Window Approach

The dynamic window approach [7] is an obstacle avoidance method that is capable of operating in a real-time environment amongst dynamic obstacles using the position of obstacles relative to the mobile platform. The algorithm was originally designed for a non-holonomic vehicle, taking into account the kinematics and dynamics of syncro-drive robots. The kinematics of the vehicle is considered by searching the velocity space (v, ω) consisting of the translational velocities v and angular velocities ω that are achievable by the vehicle as represented in Figure 1.

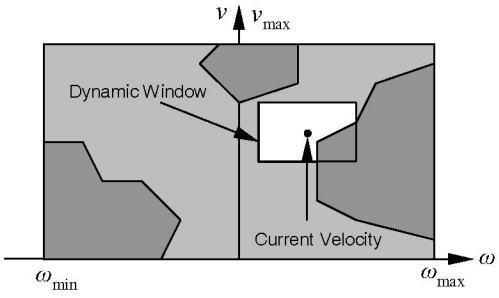

Figure 1: Dynamic Window Search Space [7]

The dynamic window approach is implemented in four steps each of which involves a further reduction of the vehicle's available velocities until the best possible translational and angular velocities have been chosen. The first step restricts the vehicle's velocity space to the achievable velocities, which are the set of translational and angular velocities achievable taking into account the kinematic constraints of the vehicle.

This set of achievable velocities is then reduced to those velocities that can safely avoid obstacles near the vehicle resulting in a set of velocities called admissible velocities. For the purposes of this method obstacles are considered avoided if the vehicle can modify its velocity or come to a complete stop to avoid a collision. This definition is far from ideal as it doesn't guarantee to keep the vehicle in motion and doesn't calculate future points of likely collision as is necessary to successfully avoid dynamic obstacles over a long driving time horizon.

The third step of the method is to create a dynamic window of the admissible velocities consisting of velocities that can be achieved within a certain time frame and within the acceleration constraints of the vehicle. This is normally a rectangular window centered on the current velocity of the vehicle and extended according to the vehicle's acceleration capabilities.

The fourth and final step of the dynamic window approach is to search the dynamic window using a cost function to find the best translational velocity v and angular velocity ω based on a set of heuristics. The heuristics of the cost function favor velocities in the direction of the goal, *angle(v,ω)*, that maintain a large distance from obstacles, *dist(v,ω)*, and that operate at faster speeds, *velocity(v,ω)*. This is all incorporated into a weighted function using coefficents σ, α, β and γ that allows the relative importance of each of the behaviors to be modified,

$$G(v,w) = \sigma(\alpha \cdot angle(v,w) + \beta \cdot dist(v,w) + \gamma \cdot velocity(v,w)), \quad (1)$$

which is then computed over the discretised set of translational and angular velocities located in the dynamic window.

IV. Velocity Obstacle Approach

The Velocity Obstacle approach [6], unlike the majority of other obstacle avoidance methods, is capable of operating amongst dynamic obstacles as it has been specifically designed for this purpose.

In order to calculate the Velocity Obstacle (VO), with reference to Figure 2, consider two circular objects, A and B, at time t_0, with velocities v_A and v_B where A represents the vehicle and B represents a moving obstacle. The first step is to reduce the vehicle circle, A, to a single point, $\hat{A}$ and to enlarge the object circle, B by the radius of A to $\hat{B}$. This now

minimizes computation by only having to calculate the intersection of a point and a circle.

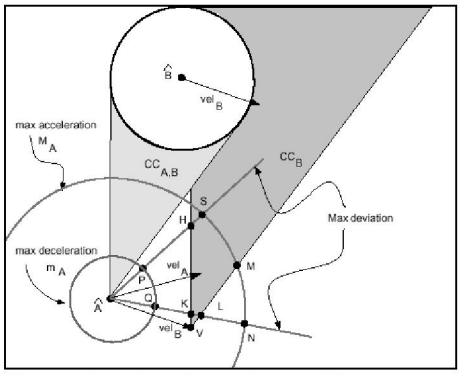

Figure 2: Construction of a Velocity Obstacle [6]

A *Collision Cone,* $CC_{A,B}$ is created using $\hat{A}$ and $\hat{B}$ which is the set of colliding *relative* velocities between $\hat{A}$ and $\hat{B}$ and is defined as:

$$CC_{A,B} = \left\{ v_{A,B} \mid \lambda_{A,B} \cap \hat{B} \neq 0 \right\}$$

where $v_{A,B}$ is the relative velocity of $\hat{A}$ with respect to $\hat{B}$, $v_{A,B} = v_A - v_B$, and $\lambda_{A,B}$ is the supporting line of $v_{A,B}$. Any relative velocity within $CC_{A,B}$ will cause A and B to collide.

In order to avoid multiple obstacles, absolute velocities must be used by adding the velocity of the obstacle, v_B, to each velocity in $CC_{A,B}$ thereby forming the Velocity Obstacle, VO, where $VO = CC_{A,B} \otimes v_B$ and $\otimes$ is the Minkowski vector sum operator, $A \otimes B = \bigcup_{\beta \in B} (A + \beta)$. The VO then is the set of colliding *absolute* velocities between $\hat{A}$ and $\hat{B}$. By finding the union of the individual velocity obstacles, $VO = \bigcup_{i=1}^{m} VO_{B_i}$, where m is the number of obstacles, multiple obstacles can be avoided using a combined velocity obstacle (VO). Therefore the vehicle can avoid a collision by selecting any velocity outside of the combined velocity obstacle; these velocities are known as *avoidance velocities.*

The dynamics of the vehicle are considered by computing the set of velocities that the vehicle can reach in the set time interval and are known as *achievable velocities*. By choosing avoidance velocities that are also achievable velocities a set of *achievable avoidance velocities* is formed. These achievable avoidance velocities can then be searched using a cost function in order to select the velocity that best fulfils a set of heuristics.

Recently some modifications have been suggested to the Velocity Obstacle method. One modification is the introduction of a short time horizon so that priority is given to avoiding obstacles that are directed towards an imminent collision [6]. Another modification has been to create a non-linear velocity obstacle capable of avoiding collisions with obstacles that do not move along a linear path [8].

V. Gradient Velocity Obstacle Approach

The Gradient Velocity Obstacle approach, as suggested by the name, is a modification of the Velocity Obstacle algorithm that uses a gradient rather than an absolute velocity obstacle. The motivation for the utilization of a gradient is that, in the case of a non-holonomic vehicle, it is possible that all of the achievable velocities may result in a collision. In this case it is necessary to take the best possible action to avoid the obstacle(s) in multiple steps by way of a gradient.

This approach has the benefit of using the well-structured steps of the Dynamic Window approach and the ability to operate amongst dynamic obstacles provided by the Velocity Obstacle approach. The combination of the Dynamic Window approach and the Velocity Obstacle approach has been investigated before, though independently, in [9], though this method fails to find the need for gradient velocity obstacles.

Similarly to the Velocity Obstacle approach the Gradient Velocity Obstacle method operates in Cartesian velocities V_x and V_Y rather than translational and angular velocity pairs. This decision was made due to the test vehicle platform – Cycab [10], using the IBEO Ladar sensor that provides velocity information on obstacles in Cartesian pairs.

In addition to the integration of the Dynamic Window approach and the Velocity Obstacle approach a number of other improvements were made, namely: the holonomic constraints of a car-like vehicle were incorporated; the velocity obstacle was discretised for improved performance and converted to a gradient; and a new cost function was created.

A. Holonomic Constraints

Most methods developed for obstacle avoidance are to be implemented on a holonomic vehicle with differential drive. Our method was required to be implemented on a car-like platform that required the consideration of non-holonomic constraints such as the inability of turning on the spot, i.e. to turn without forward motion.

The non-holonomic constraints of the vehicle were incorporated into the achievable velocities step of the dynamic window approach structure by limiting the maximum lateral velocity, V_x, based on the forward velocity, V_Y. By either limiting the maximum curvature of the vehicle, C, or by finding its rotational velocity, $d\beta/ds$, the maximum steering angle can be calculated by simplifying the vehicles motion to a two-wheel bicycle, as shown in Figure 3,

$$C = \frac{1}{R} = \frac{\tan\alpha}{L} = \frac{d\beta}{ds} \quad (3)$$

where α is the vehicle's heading direction. The maximum

steering angle is then used to limit the Cartesian velocity pairs based on the lateral velocity of the vehicle.

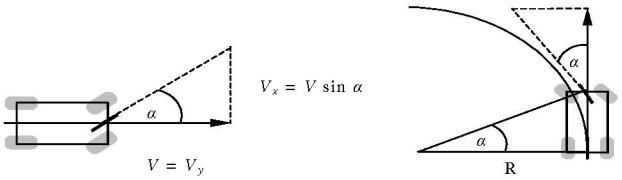

Figure 3: Simplification of Vehicle to Bicycle.

B. *Gradient Velocity Obstacle*

For the Velocity Obstacle method to interact with the Dynamic Window stage of the Dynamic Window approach, the Velocity Obstacle had to be converted from an absolute indicator of a collision to a gradient. The reason for this is when the Cost Function is used to select a velocity from the dynamic window it is possible that the whole dynamic window can contain collision velocities. In the case where an absolute velocity obstacle is used, and no avoidance is possible in the next iteration, the vehicle will no longer try to avoid a collision and will favor the other heuristics, in this case, speed and goal heading. By changing the velocity obstacle to a gradient, the vehicle can select velocities that will eventually place the vehicle along a safe path to avoid the obstacle that is essential in order for this method to be able to operate in real-time.

The Gradient velocity obstacles algorithm is as follows:

1. *Extend the left-most point (LMP) and right-most point (RMP) of the obstacle along LMPRMP by the vehicle width and a safety distance to LMP' and RMP'*
2. *For each point $P(x_0,y_0)$ inside the gradient velocity obstacle*
   - *Find $P_1(x_1,y_0)$ on OCP*
   - *Find $P_2(x_2,y_0)$ on OLMP'*
   - *The Gradient value at P_0 is calculated with:*

$$1-\frac{\left|\overrightarrow{P_0P_1}\right|}{2\times\left|\overrightarrow{P_1P_2}\right|} \quad (4)$$

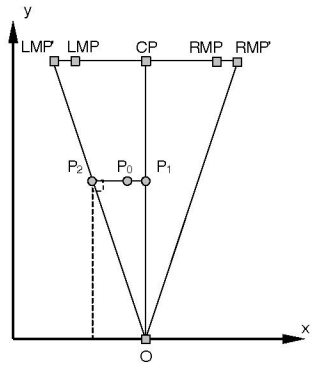

Figure 4: Implementation of the Gradient Velocity Obstacles

This algorithm results in a gradient with a maximum value of 1.0 at the center of the Velocity Obstacle and a value of 0.5 at its edge. A simplified example of the calculation of the Gradient Velocity Obstacle can be seen in Figure 5. Outside the steering constraints are blocks representing velocity pairs not possible due to holonomic constraints; inside the steering constraints blocks not covered by the gradient velocity obstacle represent safe velocity pairs; and boxes inside the gradient velocity obstacle represent velocity pairs that are predicted to result in a collision.

When more than one obstacle is present the individual gradient velocity obstacles are combined and averaged by the number of obstacles. This then encourages the vehicle to place higher priority on avoiding velocities that would cause collisions with multiple obstacles. As the gradient is currently calculated prior to combination of velocity obstacles, it is feasible that more than one local minimum could occur which may result in the vehicle maintaining an unsuitable velocity. In the future a new method of combining multiple velocity obstacles will be investigated which will remove the possibility of local minima.

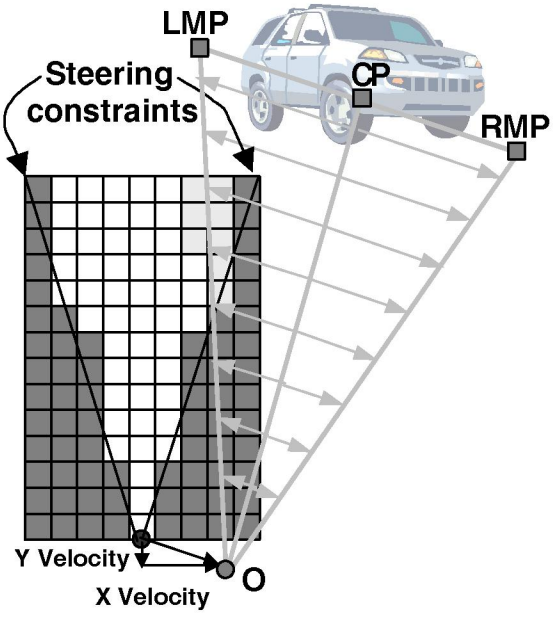

Figure 5: Gradient Velocity Obstacles

C. *Cost Function*

Similar to the cost function used in the Dynamic Window approach (1), apart from the change in coordinate systems; a function composed of three heuristics was used in this method to search for the next velocity of the vehicle. The three heuristics used were *angle*(V_x,V_y), *speed*(V_x,V_y), and *inhibition*(V_x,V_y) where V_x indicates lateral velocity and V_y indicates forward velocity relative to the vehicle platform.

The angle heuristic is used to provide the vehicle with a goal-directed behavior by finding the heading of the vehicle if it takes the current (V_x,V_y) velocity pair and then subtracting this from the goal heading and normalizing this value with π. This results in a value of 1 if the velocity pair is

directly towards the goal, and a value of 0 if in the totally opposite direction.

$$angle(V_x, V_y) = \frac{\pi - \left[\tan^{-1}\left(\frac{yGoal}{xGoal}\right) - \tan^{-1}\left(\frac{V_x}{V_y}\right) \right]}{\pi} \quad (5)$$

The speed heuristic makes the vehicle favor moving at faster velocities. The function for doing this is to normalize the current V_y velocity by the fastest velocity of the vehicle, $V_{y\max}$.

$$speed(V_x, V_y) = \frac{V_y}{V_{y\max}} \quad (6)$$

The inhibition heuristic converts the gradient of the velocity obstacles into a value that indicates the safety in choosing the velocity pair.

$$inhibition(V_x, V_y) = 1 - window(index_{V_x}, index_{V_y}) \quad (7)$$

Therefore if the dynamic window value (represented by window(x,y)) is 0, indicating the velocity will not cause a collision, the inhibition will be 1 and therefore maximized. However if the velocity is a collision velocity, the gradient at the point of the velocity pair will be subtracted from 1 making the velocity pair less likely to be chosen. This heuristic together with the gradient of the Velocity Obstacles, ensures that the vehicle will head in a direction that avoids a collision.

Combining these three heuristics the Cost Function becomes:

$$\begin{aligned} C(V_x, V_y) = {} & \delta \times angle(V_x, V_y) \\ & + \varepsilon \times speed(V_x, V_y) \\ & + \varphi \times inhibition(V_x, V_y) \end{aligned} \quad (8)$$

where δ, ε, and φ are the cost function parameters of the *angle*, *speed* and *inhibition* functions respectively.

V. Experimental Results

Testing of the method was first performed in a simulation environment and then on our experimental test platform. For simulation, a model of the Ladar sensor was used to provide data. The Ladar sensor data was simulated over an 180° field of view with any obstacle within a 20x20m radius becoming visible. In accordance with the Ladar sensors data the left-most, closest, and right-most points of the obstacle were then provided.

The aim of testing was to create simulated environments that would occur in a typical inner-city environment. Obstacles were modeled as circular figures, and by using different sizes were used to represent various moving obstacles such as pedestrians, cyclists, automobiles, and buses. Each obstacle was then given a starting point and a set velocity. Combinations of these obstacles were then used in the simulation of six possible scenarios. These scenarios were a static obstacle on a collision path with the vehicle, a static obstacle on a non-collision path with the vehicle, an intersection scenario, a head on collision scenario, a lane-merge scenario and a sidewalk/adjacent lane scenario.

All tests were performed using the cost function parameters of $\delta = 0.3$, $\varepsilon = 0.1$, $\varphi = 0.6$ with the simulated vehicle operating at 7m/s (25 km/h), and obstacles operated up to a maximum of 16.5m/s (60 km/h). Of the total 248 test cases performed the algorithm had an 89.1% success rate at avoiding all obstacles while maintaining a safety distance of a meter, and of reaching the goal. The exclusion of cases with obstacles operating at 16.5m/s raised this to 97.2% indicating that the current sensor setup is not yet adequate for operating with obstacles moving at velocities greater than 7m/s.

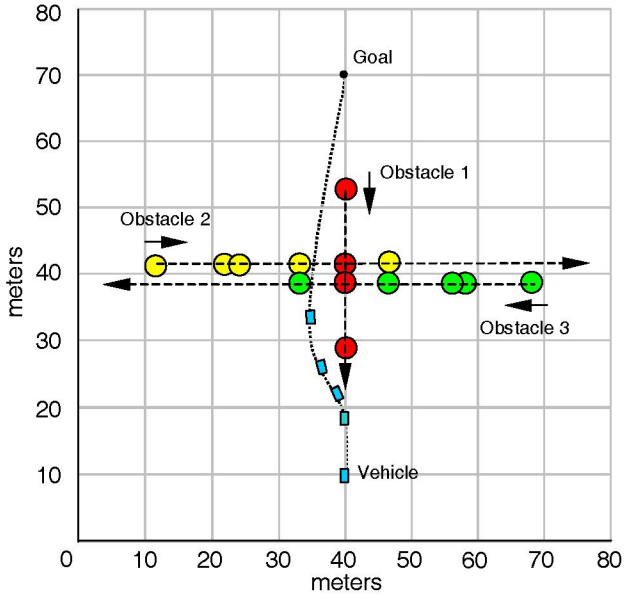

Figure 6: Intersection Scenario Test in Simulation Environment

In Figure 6 an example test of the simulation environment can be seen in which three obstacles are present. All of the obstacles and the vehicle are moving at 7m/s. Each instance of the vehicles and obstacles represents a critical stage of the simulation. The first stage shows the initial position of all the obstacles and the vehicle prior to the simulation starting. In the second stage the figure shows when the vehicle has just detected the first obstacle and is about to begin an evasive maneuver to the left. By the third stage the vehicle has detected obstacle two and using the combination gradient velocity obstacle continues to avoid to the left. The fourth stage shows the vehicle detecting the third obstacle and the gradient velocity obstacle of the third obstacle causes the vehicle to reduce its speed to avoid the collision. The fifth stage shows the vehicle maintaining a slow admissible velocity about to avoid the third obstacle, then increase

speed and reach the goal.

In addition to simulation, testing was also performed on the test vehicle platform. Initial tests were performed at low speeds (0.5m/s) with static and slow moving obstacles. The results were promising and gave initial success though more time needs to be spent performing on-road testing of the algorithm.

The algorithm was tested on our test vehicle platform, shown in Figure 7, which is a car-like vehicle operated using a 3 GHz Pentium 4 embedded computer, a second PC running a real-time Linux kernel and two MPC555 microcontrollers. The algorithm was implemented on the embedded computer that is used for high-level behaviors, while the secondary PC and the MPC555 controllers are used for low-level behavior consisting mostly of motor control. For the purposes of obstacle avoidance, the test vehicle platform was equipped with an IBEO Ladar Sensor [11] that performs scans at a frequency of 10Hz over a 270° area (though only 180° was utilized) at 0.25° degree intervals. The sensor can track 25 objects internally and transmits details of up to 20 objects that lie within a set object output area normally 6x6m, via the CAN bus. The range of the Ladar sensor is dependent on reflectivity but ranges from 100m at 90% reflectivity to 50m at 10% reflectivity.

During testing the cause of collisions was found to be local minima occurring in the gradient velocity obstacles upon the combination of multiple velocity obstacles. In the future calculating the gradient velocity obstacle after all velocity obstacles have been combined will extend this method. This would also allow the algorithm to follow a slow moving obstacle or stop to avoid a collision thereby giving success in the abortive maneuver of a similar method [9].

VI. Conclusion

This paper has outlined the current state of the art in research on the topic of autonomous motion of a driverless vehicle operating amongst dynamic obstacles. It has also presented a new algorithm, the Gradient Velocity Obstacle approach that is capable of avoiding a collision in a time-varying environment at high speeds.

The Gradient Velocity Obstacle algorithm has been shown through simulations to operate successfully at high speeds in a number of scenarios modeled on an inner-city environment. The gradient velocity obstacle used in this method allows for the vehicle to reach avoidable velocities in more than a single step.

Future work will be done on removing local minima from the gradient velocity obstacle, investigating the inclusion of recent modifications to the velocity obstacle as outlined previously, and on a method to constrain the path that the algorithm considers. Also, as the Gradient Velocity Obstacle method has only been tested at low motion, future work will involve testing the method using higher velocities of both the test vehicle platform and the dynamic obstacles.

Figure 7: The Cycab Platform

References

[1] B. Kohout, Challenges in Real Time Obstacle Avoidance, to appear in AAAI Spring Symposium on Real-Time Autonomous Systems, 2000, California, USA

[2] Y. Hwang, and N. Ahuja, Gross Motion Planning – A Survey. ACM Computing Surveys, vol. 24 (3), 1992, pp. 219-291.

[3] J. Reif, and M. Sharir, Motion Planning in the presence of moving obstacles. Journal of ACM, vol. 41 (4), 1994, pp. 764-790.

[4] T. Fraichard, and C. Laugier, Path-velocity decomposition revisited and applied to dynamic trajectory planning. In IEEE International Conference of Automation and Robotics, vol. 1, pp. 40-45, Atlanta, USA.

[5] K. Fujimura, and H. Samet, Time-minimal paths among moving obstacles. In IEEE International Conference on Robotics and Automation, pp 1110-1115.

[6] P. Fiorini, and Z. Shiller, Motion planning in Dynamic Environments using Velocity Obstacles, Int. Journal on Robotics Research, vol 17 (7), 1998, pp. 711-727.

[7] D. Fox, W. Burgard, and S. Thrun, The Dynamic Window Approach to Collision Avoidance, IEEE Robotics and Automation Magazine, vol 4 (1), 1997, pp. 23-33.

[8] Z. Shiller, F.Large and S. Sekhavat, Motion Planning in Dynamic Environments: Obstacles Moving Along Arbitary Trajectories, Int. Conf. On Robotics and Automation, 2001, Seoul, Korea.

[9] D. Castro, U. Nunes and A. Ruano, Reactive Local Navigation, Proceedings of the .28th Industrial Electronics Society Conference, 2002, Sevilla, Spain.

[10] User Needs Analysis and Analysis of Key Technologies: Report on Existing Technologies for vehicles. Retrieved 07 Feb. 2005, from http://www.cybercars.org/docs/D1Part1-UserNeed.doc

[11] IBEO LD Automotive. Retrieved 07 Feb. 2005, from http://www.ibeo-as.de/html/prod/prod_ld_autom.html.