Motion Planning in Dynamic Environments using Velocity Obstacles

Paolo Fiorini* and Zvi Shiller†

* Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109 USA

† Department of Mechanical, Nuclear and Aerospace Engineering
University of California, Los Angeles
Los Angeles, CA 90024 USA

Abstract

This paper presents a new approach for robot motion planning in dynamic environments, based on the concept of Velocity Obstacle. A velocity obstacle defines the set of robot velocities that would result in a collision between the robot and an obstacle moving at a given velocity. The avoidance maneuver at a specific time is thus computed by selecting robot's velocities out of that set. The set of all avoiding velocities is reduced to the dynamically feasible maneuvers by considering the robot's acceleration constraints. This computation is repeated at regular time intro vals to account for genera] obstacle trajectories.

The t rajectory from start to goal can be com puted by searching a tree of feasible avoidance maneuvers computed at discrete time int ervals. An exhaustive search of the tree yields near-optima] trajectories that either minimize distance or motion time. A

heuristic search of the tree yields trajectories that satisfy a prioritized list of objectives, such as reaching the goal, maximizing speed, and achieving a desired trajectory structure.

The heuristic approach is computationally efficient, applicable to on-line planning of industrial robots, performing assembly tasks cm moving conveyers, and to intelligent vehicles negotiating freeway traffic. The method is demonstrated for planning the trajectory of an automated vehicle in an Intelligent Vehicle Highway System scenario.

1. Introduction

This paper addresses the problem of motion planning in dynamic environments. Typical examples of dynamic. environments include manufacturing tasks in which robot manipulators track and retrieve parts from moving conveyers, and air, sea, and land traffic, where aircraft, vessels and vehicles avoid each other while moving towards their destination.

Motion planning in dynamic environments is considerably more difficult than the widely studied static problem, since it requires the simultaneous solution of the path planning and of the velocity planning problems. Path planning involves the computation of a collision free path from start to goal without considering robot dynamics. Velocity planning, on the other hand, involves the computation of the velocity profile along a given path, satisfying system dynamics and act uator constraints. In addition, motion planning in static environments can be guaranteed to find a solution if one exists at time t_0, whereas motion planning in dynamic environments is essentially intractable, [29], [7], i.e. the solution at t_0 may not mist at a later time because of the evolution of the environ In ent.

The problem of planning in a time-varying environment has been previously addressed for obstacles moving at known speeds [28] [9] [18] [15]. Erdmann [9, 10] computed a collision-free trajectory using a position-time representation of the environment, thus extending the concept of Configuration Space [22--24], to dynamic motion planning by using time as an additional dimension. The time axis was discretized, and the configuration space of the robot was computed at successive time intervals.

in [18,19], Kant decomposed the planning problem into two steps: first, a path is selected to avoid the static obstacles; then, the velocity profile along that path is selected to avoid the moving obstacles. The latter step was carried out on a space-time plane in which the abscissa was the path arc length, and the ordinate was time. A similar approach was developed by Lee, [21], who computed a velocity profile that reached the goal within a prescribed time limit. Approximate dynamic constraints were satisfied by limiting slope, direction and curvature of the trajectory in the space-time plane.

Canny [8] presented an algorithm for computing a minimal-time trajectory of a point mass among fixed obstacles that satisfied acceleration constraints. The result was a polynomial time algorithm computing a trajectory on a Non-linear fixed-time grid in the phase space of the point-mass robot. A general method for motion planning is provided by the roadmap algorithm [6], that can consider moving obstacles. in [26], Ó'Dúnlaing presented an exact algorithm for computing the optimal velocity profile on a one-dimensional path subject to acceleration bounds.

In [15,16], Fujimura uses dynamic constraints to define the collision front, i.e. the locus of collision points between two objects moving on arbitrary paths with piecewise constant velocities. With this approach, a minimum time trajectory can be computed

by selecting the maximum feasible speeds at every moment, and the dynamic constraints can be included by limiting the path curvature. However, the minimum time result is only guaranteed when robots are faster than the obstacles. Recently, in [13,14], Fujimura introduced the concept of transient obstacles, such as objects placed and removed by a manipulator, that could also be applicable to dynamic planning.

In [28], Reif presented a solution to the two- dimensional *asteroid* problem, where a velocity-limited robot is required to reach its destination while avoiding any number of moving polygonal obstacles. The computation is carried out in the configuration space-time, and it is similar to a visibility graph search. The algorithm computes a tree of possible motions joining two asteroids, until it reaches the goal,

In [2–5], Cameron modeled solid objects in motion by representing them in a four-dimensional space, thus reducing the collision detection problem to the test for intersection among four-dimensional objects. By constructing the four-dimensional representation of the objects, the computation of a non colliding path is reduced to finding a position resulting in no intersections among the moving objects.

Hayward [17] recently presented an approach to manage the computation burden of detecting the future collisions of a robot with many moving obstacles. Relative coordinates were used to compute the time to collision for every pair robot-obstacle, subject to velocity and acceleration constraints. These times were used as a measure of urgence for ordering the obstacles, and prioritize the collision avoidance.

On-line planning in dynamic environments has been so far treated mostly as an AI problem, emphasizing reasoning and decision making, with no concern to robot dynamics. In [29], Sanborn develop reaction rules for a robot moving in a Traffic World. The moving

robot reacts to the environment by predicting the space-time intervals in which belligerent and apathetic objects would intersect its specified path. While this approach reasons from the physical properties of the environment, it neglects the physical limitations of the moving robot.

Although this problem has been extensively addressed in the literature, the current methods are either computationally too expensive, or do not consider robot's dynamics.

In this paper, we develop an efficient method to compute the trajectories of a robot moving in a time-varying environment that considers robot dynamics aud actuator constraints. It utilizes the concept of Velocity Obstacle (VO), which maps the dynamic environment into the robot velocity space. The velocity obstacle is the set of robot's velocities that would cause a collision with an obstacle if both maintain their current velocities. Thus, a collision avoidance 'maneuver is computed by selecting velocities that are outside of the velocity obstacle, if such velocities exist. To ensure that the maneuver is dynamically feasible, robot dynamics and actuator constraints are transformed into constraints on the robot acceleration which are then mapped into the robot velocity space. A complete trajectory consists of a sequence of such avoidance maneuvers, selected to achieve some desired objective.

The avoidance trajectory that minimizes motion time is computed by searching over a tree of avoidance maneuvers generated at discrete time intervals. For on-line applications, the tree is pruned using a heuristic search designed to achieve a prioritized set of objectives, such as avoiding collisions, reaching the goal, maximizing speed, or computing trajectories with desirable topology.

The exhaustive search and the heuristic planning have been implemented for an

autonomous vehicle negotiating freeway traffic, moving from the fast lane to the exit ramp.

The advantages of this approach are multi-fold: *i*) it permits an efficient geometric representation of potential avoidance maneuvers of the moving obstacles, *ii*) any number of moving obstacles can be avoided by considering the union of their VO'S, *iii*) it unifies the avoidance of moving as well as stationary obstacles, and *iv*) it allows for the simple consideration of robot dynamics and actuator constraints.

The solutions computed with this method are conservative, since they exclude trajectories that, although feasible, include avoidance maneuvers violating the velocity obstacle. However, such trajectories may also be considered by excluding obstacles with a long time to collision, or by optimizing; the trajectory using a dynamic optimization [11].

This paper is organized as follows, Section 2 defines the Velocity Obstacle and its basic properties. Section 3 define the avoidance velocities, and presents the procedure for computing; the feasible avoidance maneuvers, Complete trajectories are computed in Section 4 using both global and heuristic, search methods. Examples of a single avoidance maneuver and of complete trajectories are presented in Section 5.

2. The Velocity Obstacle

The Velocity Obstacle (VO) is an extension of the Configuration Space Obstacle [24] to a time-varying environment. It consists of the velocities of the robot that will cause a collision between the robot and the obstacles at some future time. Although this concept is valid for general robots and obstacles, in this paper we restrict our analysis to circular robots and obstacles in the plane.

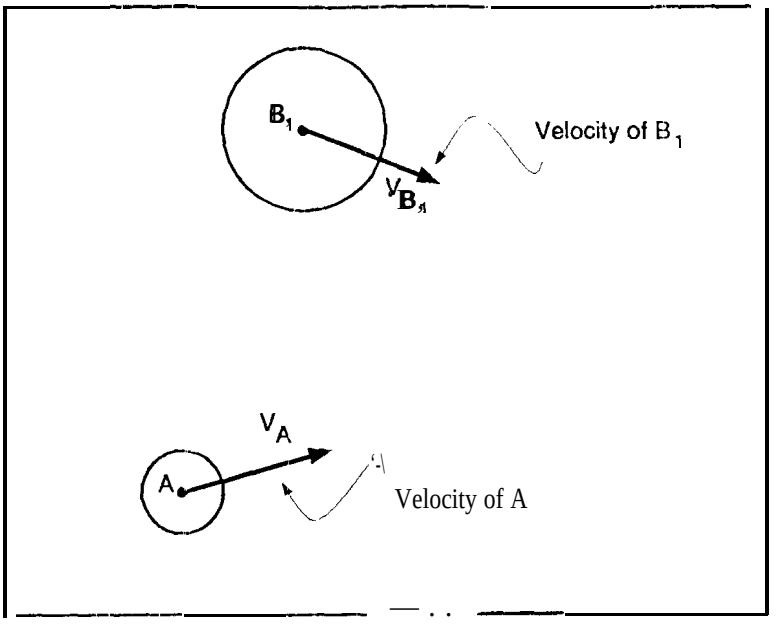

Figure 1: The robot and a moving obstacle.

The VO is illustrated using the scenario shown in Figure 1, where two circular objects, A and B_1, are shown at time t_0 with velocities $\mathbf{v}_A$ and $\mathbf{v}_{B_1}$ [12]. Circle A represents the robot, and circle B_1 represents the obstacle. The velocities and positions of A and B_1 were chosen so that A and B_1 will collide at some time $t_1, (t_1 > t_0)$, provided that $\mathbf{v}_A$ and $\mathbf{v}_{B_1}$ do not change. The VO is then used to modify the velocity of A so that the collision is avoided.

To compute the VO, we first represent, B_1 in the *Configuration Space* of A by reducing A to the point $\hat{A}$, and enlarging B_1 by the radius of A to the circle $\widehat{B_1}$ [22], We then represent the state of the moving robot and of the obstacle by attaching their velocity vectors to the position of $\hat{A}$ and to the center of $\widehat{B_1}$, respectively. This representation, called the *Velocity Space* of the underlying configuration space [1], allows the direct use of computational geometry tools for computing the avoidance maneuvers in a dynamic environ ment.

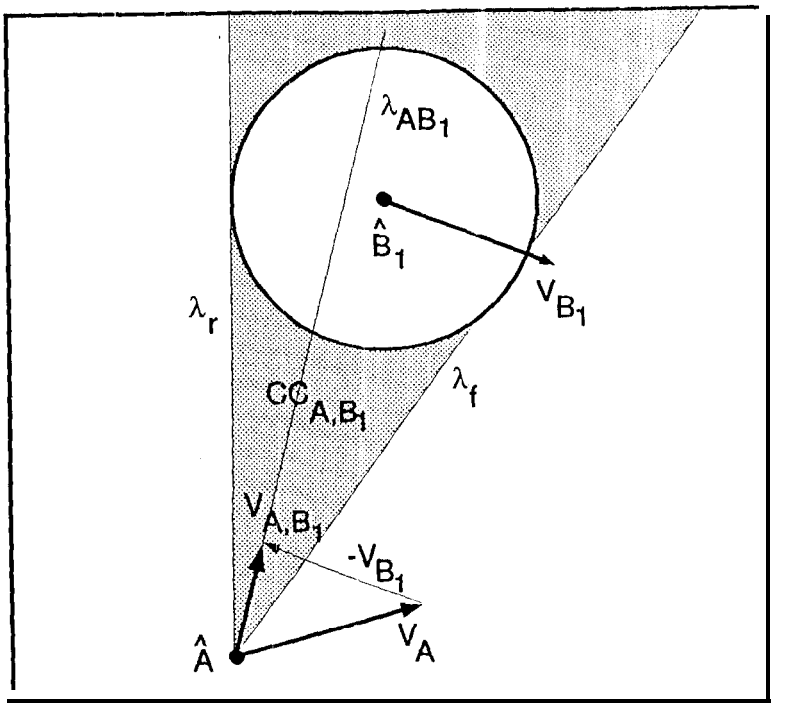

Figure 2: The Relative Velocity $\mathbf{v}_{A,B_1}$ and the Collision Cone CC_{A,B_1}.

The analysis is further simplified by considering the relative velocity $\mathbf{v}_{A,B_1}$:

$$\mathbf{v}_{A,B_1} = \mathbf{v}_A - \mathbf{v}_{B_1} \tag{1}$$

Assuming that A and B_1 maintain their current velocities, a collision between $\hat{A}$ and $\widehat{B_1}$ will occur at some future time $t_1 > t_0$ if the line λ_{A,B_1} of the relative velocity $\mathbf{v}_{A,B_1}$ intersects B_1, as shown in Figure 2, or:

$$\lambda_{A,B_1} \cap \widehat{B_1} \neq \emptyset \tag{2}$$

In fact, any relative velocity that lies between the two tangents to $\widehat{B_1}$ λ_f and λ_r will muse a collision between æ and l?$_1$. Therefore, we define the *Collision Cone*, CC_{A,B_1}, as the set of colliding relative velocities between $\hat{A}$ and $\widehat{B_1}$ satisfying equation (2):

$$CC_{A,B_1} = \{\mathbf{v}_{A,B_1} \mid \lambda_{A,B_1} \cap \widehat{B_1} \neq \emptyset\} \tag{3}$$

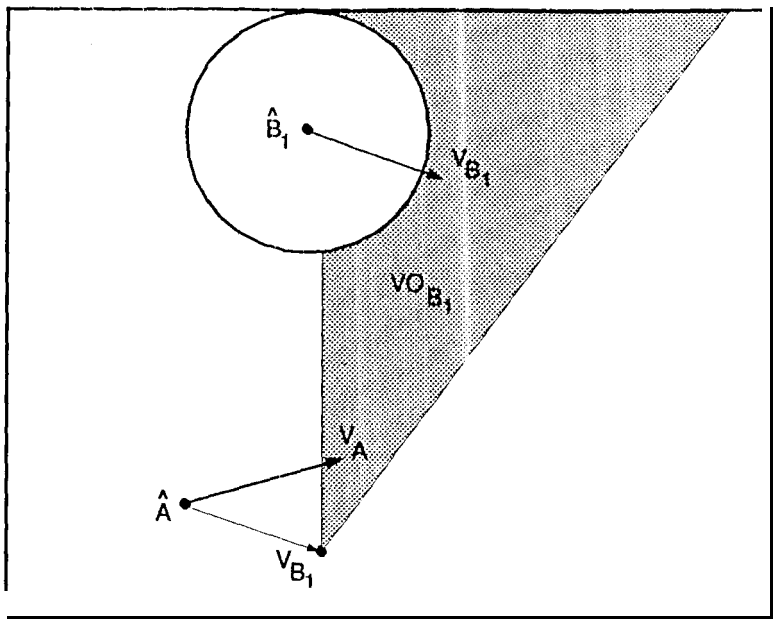

Figure 3: *The* velocity obstacle VO.

'This cone is the planar sector with apex in $\hat{A}$, bounded by the two tangents λ_f and λ_r from $\hat{A}$ to $\widehat{B_1}$, as shown in Figure 2.

Clearly, any relative velocity outside CC_{A,B_1} is guaranteed to be collision free, provided that the obstacle $\widehat{B_1}$ retains its current shape and spew]. The collision cone thus partitions the space of *relative* velocities into colliding and avoiding; velocities. The relative velocities, $\mathbf{v}_{A,B_1}$, lying on the boundaries of CC_{A,B_1} represent *tangent* maneuvers that would graze the obstacle B_1, as discussed later in Section 3.2.

The collision cone is specific to a particular pair of robot/obstacle. To consider multiple obstacles, it is useful to establish au equivalent partition of the *absolute* velocities of A. This is done simply by adding the velocity of B_1, V_{n}, , to each velocity in CC_{A,B_1} or, equivalently, by translating the collision cone CC_{A,B_1} by $\mathbf{v}_{B_1}$, as shown in Figure 3

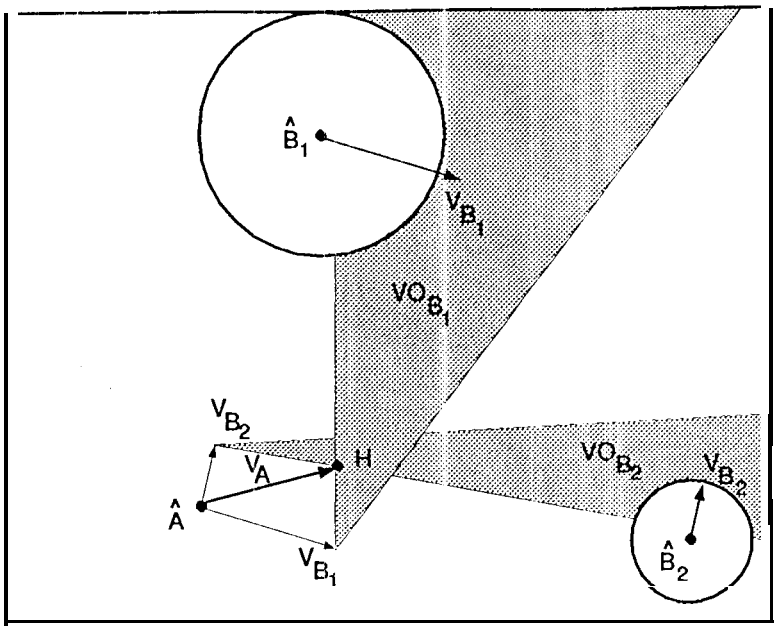

Figure 4: Collision cones and velocity obstacles for B_1 and B_2.

[12]. The *Velocity Obstacle* VO is then defined as:

$$VO = CC_{A,B_1} \oplus \mathbf{v}_{B_1} \tag{4}$$

where $\oplus$ is the Minkowski vector sum operator.

'Thus, the VO partitions the absolute velocities of A into *avoiding* and colliding velocities. Selecting $\mathbf{v}_A$ outside of VO would avoid colliding; with B_1, whereas selecting $\mathbf{v}_A$ so that its tip is inside VO would eventually result in a collision if both A and B_1 maintain their current velocities, or

$$A(t) \cap B_1(t) = \emptyset \quad \text{if } \mathbf{v}_A(t_0) \notin VO(t_0),\ t > t_0 \tag{5}$$

Velocities on the boundaries of VO *would* result in A grazing B_1, since the corresponding relative velocities lie on the boundary of the collision cone CC_{A,B_1}. Note that the VO of a stationary obstacle is identical to its relative velocity cone, since then $\mathbf{v}_{B_1} = 0$.

To avoid multiple obstacles, the, VO's of each obstacle are combined into a single velocity obstacle:

$$VO = \cup_{i=1}^{m} VO_i \tag{6}$$

where m is the number of obstacles. The avoidance maneuvers, then, consist of those velocities $\mathbf{v}_A$, that are outside the union of the VO's, as shown in Figure 4. The VO assumes that the velocity of B_1 remains constant. '1'0 account for variable velocities, VO is recomputed at specified time intervals.

The assumption of circular robot and obstacles reduces the dimension of the configuration space, and thus greatly simplifies the computation of the VO. It also fixes the shape of the configuration space obstacles, which are generally functions of their positions in the robot's work space. For general manipulators, the VO must be periodically recomputed to account for the time varying configuration space obstacles [1 O].

The maneuvers generated by velocities outside the velocity obstacle VO are guaranteed to avoid collision with the obstacles, independently of tile expected collision times. This excludes maneuvers that are on a collision course with obstacles with a long time to collision. Such maneuvers may not necessarily result in collision, since they can be appropriately modified at a future time. However, this can be remedied by considering only obstacles with imminent collision.

3. The Avoidance Maneuvers

In this Section, we first define the reachable avoidance velocities, which are a subset of all avoidance velocities, satisfying the robot's dynamic constraints. We then identify the

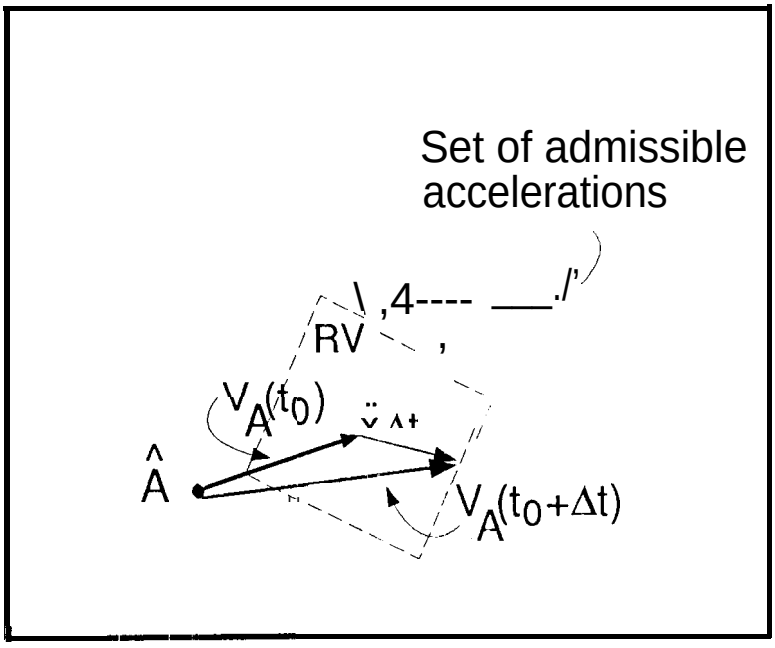

Figure 5: The Feasible Accelerations

topological properties of the avoidance maneuvers, and present a procedure for computing the set of reachable avoidance velocities.

3.1. Definition of Avoidance Velocities

The velocities reachable by robot A at a given state over a given time interval Δt are computed by transforming the dynamic constraints of the robot into bounds on its acceleration. The set of *feasible accelerations* at time t_0, $FA(t_0)$, is defined as;

$$FA(t_0) = \{\ddot{\mathbf{x}} \mid \ddot{\mathbf{x}} = f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}), \mathbf{u} \in U\} \tag{7}$$

where $f(\mathbf{x}, \dot{\mathbf{x}}, \mathrm{u})$ represents the dynamics of the robot, u are the actuator efforts, U is the set of admissible controls, and x is the position vector defined earlier. Note that the feasible acceleration range of a two degree-of-freedom system with decoupled actuator limits is a parallelogram [3(1].

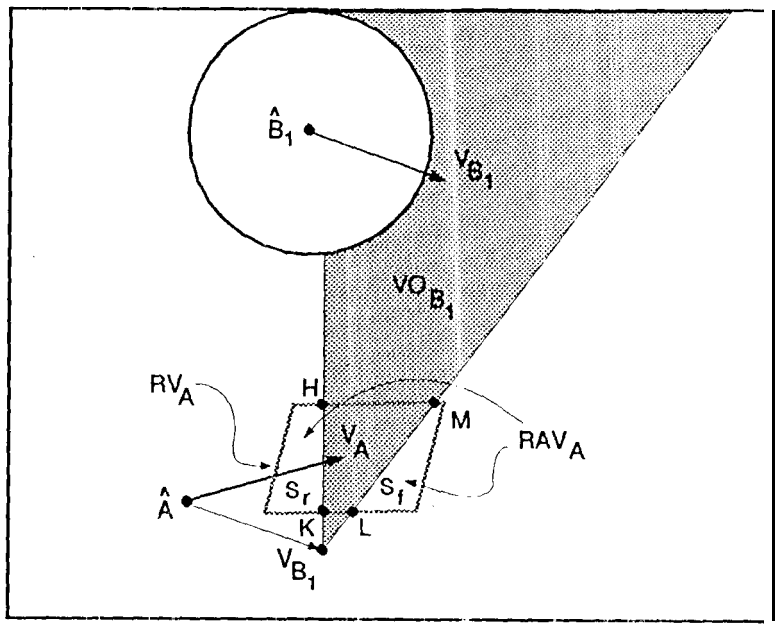

Figure 6: The reachable avoidance velocities RAV.

The set of *reachable velocities,* $\mathrm{RV}(t_0 + \mathrm{At})$, over the time interval Δt is thus defined as:

$$\mathrm{RV}(t + \Delta t) = \{\mathrm{v} \mid \mathrm{v} - \mathbf{v}_A(t_0) \oplus \Delta t \cdot FA(t_0)\} \tag{8}$$

Figure 5 shows a schematic construction of the reachable velocity set for robot *A,* by adding the change in velocity $(\ddot{\mathbf{x}}\Delta t \mid \mathrm{x} \in FA)$ to the current velocity of A.

The set of *reachable* avoidance velocities, RAV, is defined as the difference between the reachable velocities and the velocity obstacle:

$$\mathrm{RAV}(t_0 + \mathrm{At}) = \mathrm{RV}(t_0 + \Delta t) \ominus VO(t_0) \tag{9}$$

where $\ominus$ denotes the operation of set difference. A maneuver avoiding obstacle B_1 is thus computed by selecting any velocity in RAV.

Figure 6 shows schematically a RAV set, consisting of two disjoint closed sets, S_f and S_r. For multiple obstacles, the RAV may consist of multiple disjoint subsets. The selection

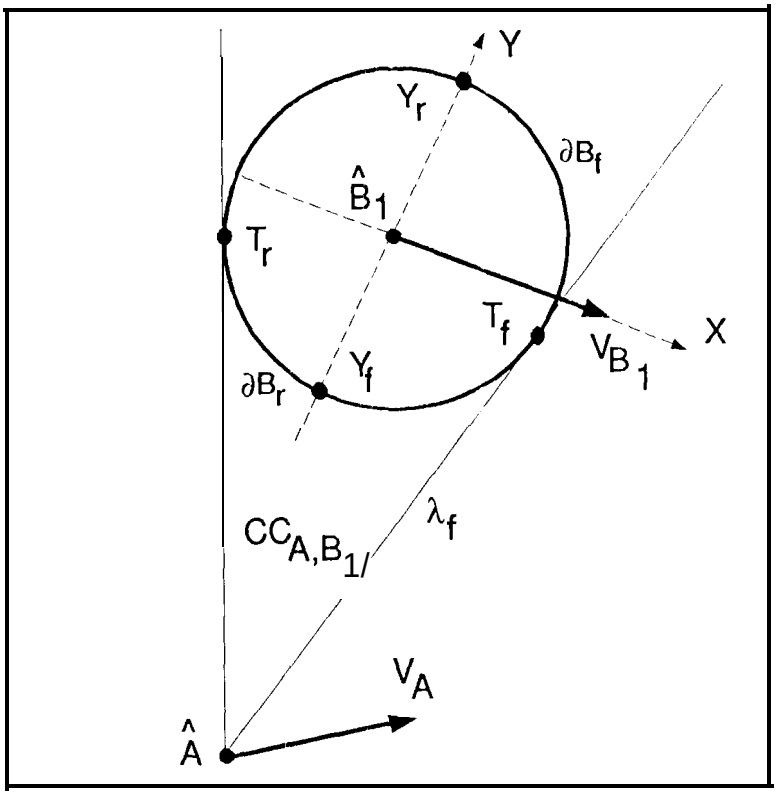

Figure 7: Grazing arcs in an avoidance maneuver.

of a specific avoidance velocity determines the structure of the avoidance maneuver, i.e. the side of the obstacle that will be avoided by the robot, as discussed next.

3.2. Structure of the Avoidance Maneuvers

In this section, we first compute the maneuvers *tangent* to the moving obstacle, and their corresponding tangency points. These maneuvers are then used to identify the *front* and *rear* avoid ante maneuvers, i.e. those that would pass in front or behind the moving obstacle.

The front and rear sides of $\widehat{B_1}$ are identified by a coordinate frame (X,Y) with its origin at the center of B_1, and its X axis coinciding with the velocity $\mathbf{v}_{B_1}$. The Y axis then partitions the boundary of $B_1, \partial(B_1)$, into the *front* semi-circle, $\partial(B_{1f})$, which intersects

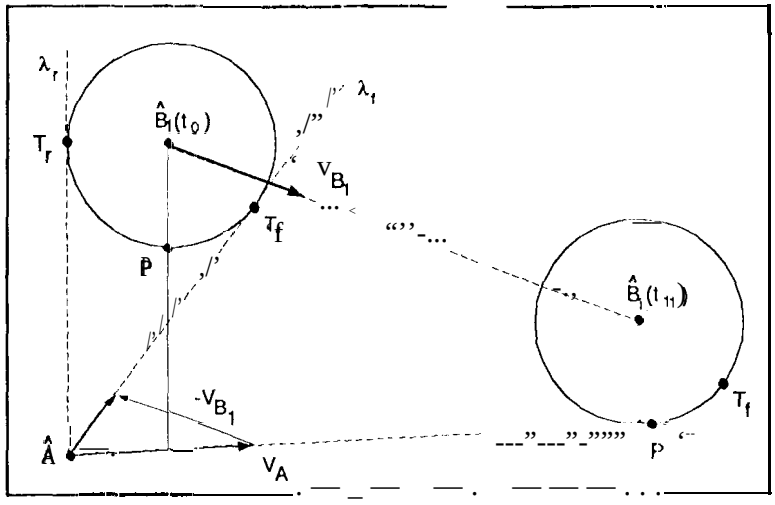

Figure 8: Trajectory tangent to B_1.

the positive X axis, and the *rear* semi-circle, $\partial(B_{1_r})$, as shown in Figure 7.

As stated earlier, any relative velocity $\mathbf{v}_{A,B_1}$ that coincides with the boundary of the Collision Cone $CC_{A,B_1}, \{\lambda_f, \lambda_r\}$, would result in A grazing B_1. We now show that: i) these relative velocities correspond to absolute velocities generating trajectories that are tangent to $\widehat{B_1}$ at some point $P \in \partial(B_1)$, and ii) that point P is different from points T_f and T_r, at which λ_f and λ_r are tangent to $\widehat{B_1}$.

Figure 8 shows obstacle $\widehat{B_1}$ at time t_0 when the velocity obstacle and the avoidance maneuver are computed, and at t_1, when $\widehat{B_1}$ is tangent to the trajectory generated by an absolute velocity $\mathbf{v}_A$ chosen so that $\mathbf{v}_{A,B_1}$ coincides with λ_f. Assuming that B_1 does not rotate during its motion, the tangency point P between the trajectory of A and the obstacle can be found by intersecting the straight line perpendicular to $\mathbf{v}_A$ and passing through the center of $\widehat{B_1}$ at time t_0, as shown in Figure 8. Clearly $P \neq T_f$, since P depends on the *absolute* velocities of A and B_1.

'Therefore, to choose au appropriate avoidance maneuver, it is necessary to establish

a mapping between the reachable avoidance velocities and the tangency points on $3?_1$. The following Lemma states that the set of *tangent* relative velocities consists only of the semi-infinite lines λ_f and λ_r, and that the tangency points can occur only on at most two, non-overlapping, subsets of $\partial(B_1)$.

Lemma 1: Robot A can be tangent to obstacle B_1 at, some point $P \in \partial(B_1)$ *iff* it follows a trajectory generated by $\mathbf{v}_A$ corresponding to $\mathbf{v}_{A,B_1} \in \{\lambda_f, \lambda_r\}$. The tangency set in $\partial(B_1)$ consists of the shortest segment connecting $T_f = \lambda_f \cap \partial(B_1)$ to Y_f, and of the shortest segment connecting $T_r = \lambda_r \cap \partial(B_1)$ to Y_r. ❑

This Lemma is proved in the Appendix.

Figure 7 shows the tangency sets between A and B_1, i.e. $T_f Y_f \in \partial(B_{1f})$ and $T_r Y_r \in \partial(B_{1r})$. Points Y_f and Y_r are defined by the two tangents to $\widehat{B_1}$ parallel to $\mathbf{v}_{B_1}$.

The conditions on the relative velocities map into conditions on the absolute velocities of A, recognizing that the boundaries of the collision cone, CC_{A,B_1}, transform to the boundaries of the velocity obstacle VO. Therefore, the boundary of the velocity obstacle VO, $\{\delta_f, \delta_r\}$, represents all absolute velocities generating trajectories tangent to $\widehat{B_1}$, since their corresponding relative velocity lies on λ_f and λ_r. In the example shown in Figure 6, the only reachable velocities that would result in A grazing $\widehat{B_1}$ are represented by the segments KH and LM of the reachable avoidance velocity set RAV.

We use the tangent velocities to subdivide the set RAV into subsets, each containing velocities generating a single type of avoidance maneuver, The set RAV in Figure 9 is subdivided into the three subsets S_f, S_r, and S_d by the boundary of VO and by the straight line passing through $\dot{A}$ and the apex of VO_{B_1}. Each of these subsets corresponds

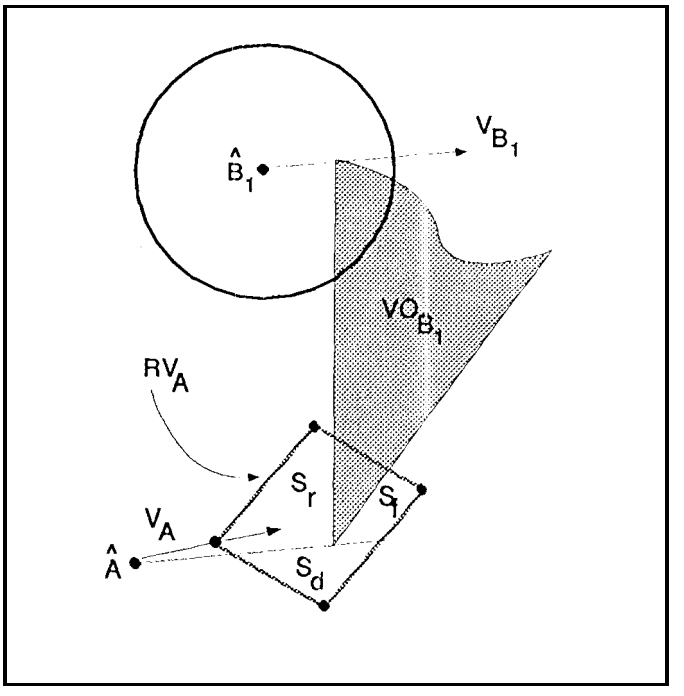

Figure 9: General structure of the reachable avoidance velocities.

to *rear*, *front*, or *diverging* avoidance maneuvers, respectively, as stated in the following Lemma. A diverging maneuver is one that takes the robot A away from the obstacle B_1.

Lemma *2:* The reachable avoidance velocities RAV due to a single obstacle consists of at most three non-overlapping subsets S_f, S_r, and S_d, each representing velocities corresponding to front, rear or diverging maneuvers, respectively. □

The proof of this Lemma is given in the Appendix.

Lemma 2 states that RAV due to one obstacle may consist of *at most* three subsets. Clearly, there can be cases where the RAV consists of fewer sets, one, two or none, as was shown in Figure 6.

For many moving; obstacles, we define the type of the avoidance maneuver according to the avoidance of each obstacle. We represent each avoidance set by S_{ij}, where the index

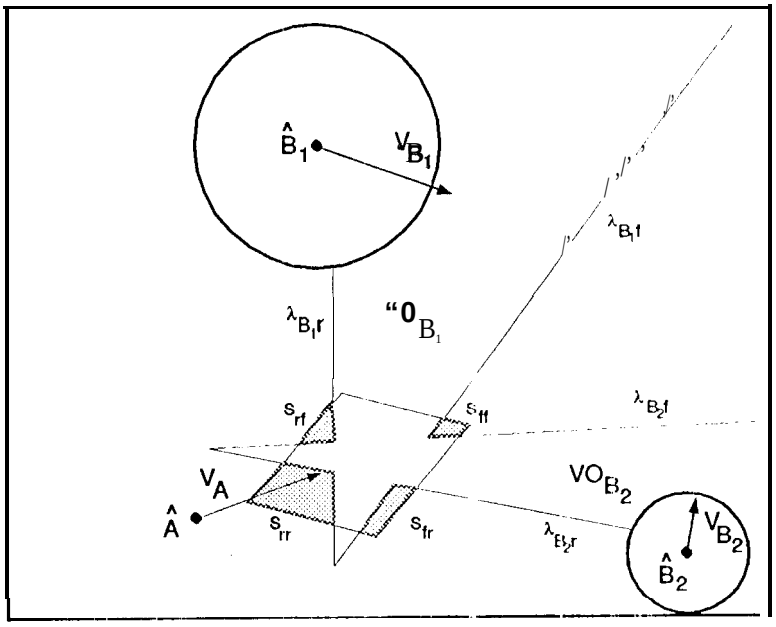

Figure 10: Classification of the reachable avoidance velocities.

i represents the type of avoidance velocity, i.e. $(i = f,r,d)$, aud j is the index of the obstacle. For example, the set S_{ff} shown in Figure10 includes velocities generating maneuvers that avoid both obstacles B_1 and B_2 with a front maneuver. The set S_{fr} corresponds to the front avoidance of B_1 and rear avoidance of B_2, respectively.

The properties of the RAV generated by multiple obstacles are summarized in the following Theorem, whose proof is given in the Appendix.

Theorem 1: Given a robot A and m moving obstacles $B_j (j = 1,..., m)$, the reachable avoid anceset RAV consists of at most $3m$ subsets, each including velocities corresponding to a unique type of avoidance maneuver. m

The Theorem states that it is possible to subdivide the avoid ance velocities RAV into subsets, each corresponding to a specific avoidance maneuver. This allows 11s to select avoidance maneuvers with a desired structure. For example, a car avoiding a big

truck may choose a rear avoidance maneuver, even though a front maneuver might also be feasible, to avoid a potential collision, due to uncertainties in the measurement of the truck's velocity.

3.3. Generating the Set of Avoidance Velocities

'This section describes the procedure used for generating the Set of Reachable Avoidance Velocity, S_i, for robot A. 'The procedure is illust rated in Figure 11, using the set of reachable velocities RV, the velocity obstacle VO_{B_1} and the line l_{aa} passing through A and the apex of VO_{B_1}.

Since the RV set is a polygon, and the open set VO_{B_1} is represented with a planar triangle, it is convenient to represent each set by an ordered list of their vertices. A typical vertex V_i is specified by its position and by the supporting lines intersected at that vertex: $V_i = \{(x_i, y_i), (l_{i-1}, \text{ii})\}$. The vertices are ordered counter-clockwise so that the interior of the polygon is on the left of its boundary. This representation facilitates the set operations needed to compute RAV.

The first step of the procedure is to intersect RV with the line l_{aa} to form the set of diverging velocities, S_d, and the set of non-diverging velocities, S_{nd}, as shown in Figure 11-a. In this case, the set S_d is represented by vertices *(5, 2, 3, 6),* and S_{nd} is represented by (1, 5,6, 4). By construction, the set S_d is not affected by the velocity obstacle VO_{B_1}, whereas S_{nd} is, generally, subdivided by VO_{B_1} into su bsets of colliding and avoiding velocities, as shown in Figure 1 1-1).

'The computation of these subsets is carried out as illustrated, for one obstacle, in Figure 1 l-c.. First, the intersections between S_{nd} and the velocity obstacle VO_{B_1} are computed. For the case shown, these are points (7, *8,9)* that are inserted in the proper

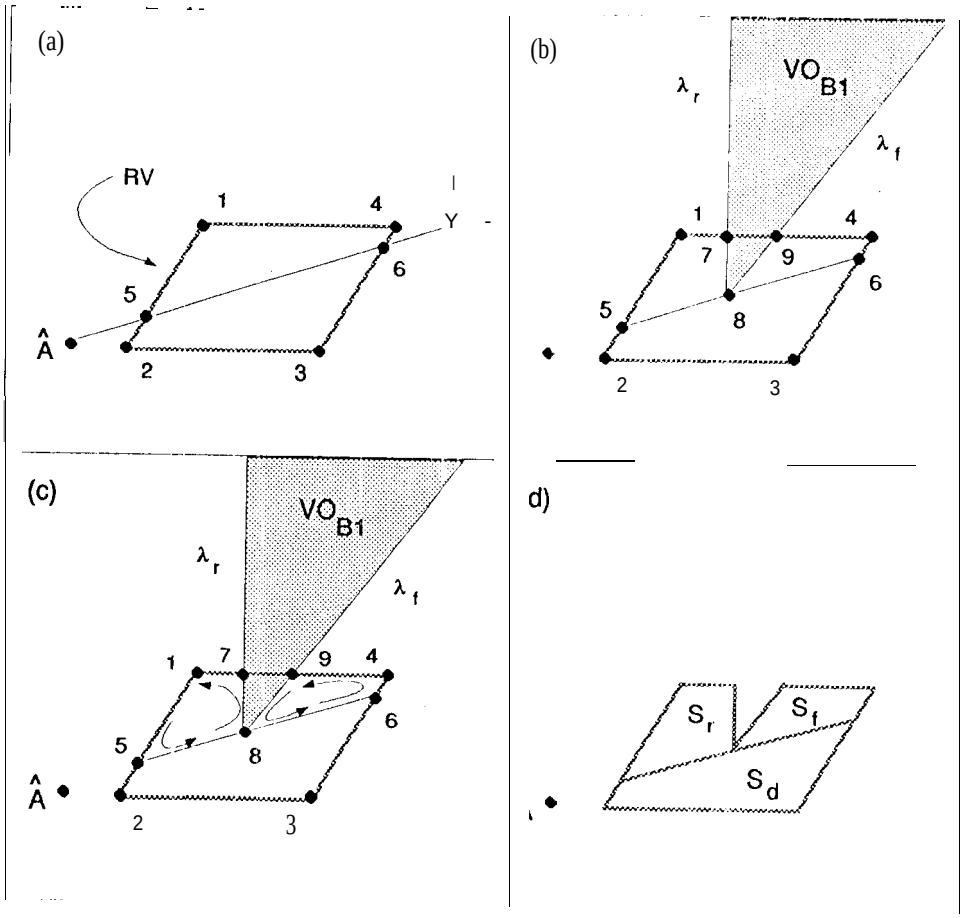

Figure 11: The computation of the Reachable Avoidance Velocities RAV.

order in the list of S_{nd}. The new list of S_{nd} consists of points (1,5,8,6,4,9,7). This list is then scanned starting from the first vertex that is external to VO_{B_1}, in this case vertex 1. The subsets of avoidance velocities are created by subtracting VO_{B_1} from S_{nd} marching counter-clock wise along the boundary of S_{nd} and clockwise along the boundary of VO_{B_1}, as shown schematically in Figure 1 l-c. This corresponds to assigning a positive value to S_{nd} and a negative value to VO_{B_1}. *The* set whose boundary includes a segment of the rear boundary of $VO_{B_1}\lambda_f$, is the set of rear avoidance velocities S_r, and the set whose boundary includes a segment of the front boundary λ_r is the set of front, avoidance velocities S_f. in this case, set S_f is represented by (9,8,6,4) and S_r by(1 ,5,8, 7). Finally,

Figure 11-d, shows the reachable avoidance velocities RAV = RV $\ominus VO_{B_1}$ consisting of the three subsets, Sf, S_r, and S_d, corresponding to front, rear aud diverging velocities, respectively.

To consider multiple obstacles, this procedure is applied recursively to each subset of RAV, generating smaller subsets S_{ij}, where the index i represents the type of avoidance velocity, i.e. (i=f,r,d), and j is the index of the obstacle. For example, three obstacles might generate up to 27 avoidance velocity sets, such as S_{frd} representing a front avoidance maneuver of the first obstacle, a rear avoidance of the second, and a diverging maneuver from the third obstacle.

The upper bound on the number of sets S_{ij} is 3^m, where m is the number of obstacles in the environment, However, the actual number is much smaller, because the lines l_{oa} from A to the apex of each VO do not intersect each other. Also, many of the potential subsets S_{ij} may be covered by some of the VO_{B_j}. Generally, we observed that the higher the number of obstacles, the fewer the sets S_{ij} remaining to be computed. 'To ensure that the set RAV is not empty, it might be useful to consider only obstacles with a short expected collision time. For example, the m obstacles $B_j (j = 1, \ldots . m)$, can be sorted according to their expected collision times t_j [1 7], and only those with time to collision $(t_j$ -to< $t_h)$, with t_h a suitable horizon, can be included in the computation of RAV at t_0.

4. Computing the avoidance trajectories

This section presents a method for computing complete trajectories that, avoid static, and moving obstacles, reach the goal, and satisfy the robot's dynamic constraints. A

trajectory can be computed as a sequence of elementary avoidance maneuvers, selected by a global search over the tree of all feasible maneuvers at specified time intervals. Alternatively, the global search may be reduced to a heuristic search for on-line applications, where the trajectories of the moving obstacles are not known a-priori, but are rather acquired in real-time. These two approaches are discussed next.

4.1. Global Search

To facilitate an efficient global search, we represent the state space of the robot by a tree of avoidance maneuvers at discrete time intervals. The *nodes* on this tree correspond to the positions of the robot at discrete times t_i. The *operators* expanding a node at time t_i into its successors at time $t_{i+1} = t_i + T$ are the velocities in the reachable avoidance velocity set RAV. The *edges* correspond to the avoidance maneuvers at those positions [27,25]. The search tree is then defined as follows:

$$n_{i,j} = \{\mathbf{x}_{i,j} = (x_j(t_i), y_j(t_i)), \mathbf{v}_{i,j} = (v_{j,x}(t_i), v_{j,y}(t_i))\} \tag{10}$$

$$o_{i,j,l} = \{\mathbf{v}_l \mid \mathbf{v}_l \in \mathrm{RAV}_j(t_i)\} \tag{11}$$

$$e_{j,k} = \{(n_{i,j}, n_{i+1,k}) \mid n_{i+1,k} = n_{i,j} + (o_{i,j,l}\,\Delta t)\} \tag{12}$$

where $n_{i,j}$ if tile jth node at time t_i, $\mathrm{RAV}_j(t_i)$ is tile reachable velocity set computed for node $n_{i,j}$, $o_{i,j,l}$ is the lth operator on node j at time t_i, and $e_{j,k}$ is the edge between node n_j at time t_i, and node n_k at time t_{i+1}.

The tree of all feasible avoidance maneuvers is constructed by the following operations. At time t_i, the avoidance set RAV, corresponding to node $n_{i,j}$, is discretized by a grid. Then, the velocities corresponding to each node on the grid are used to compute

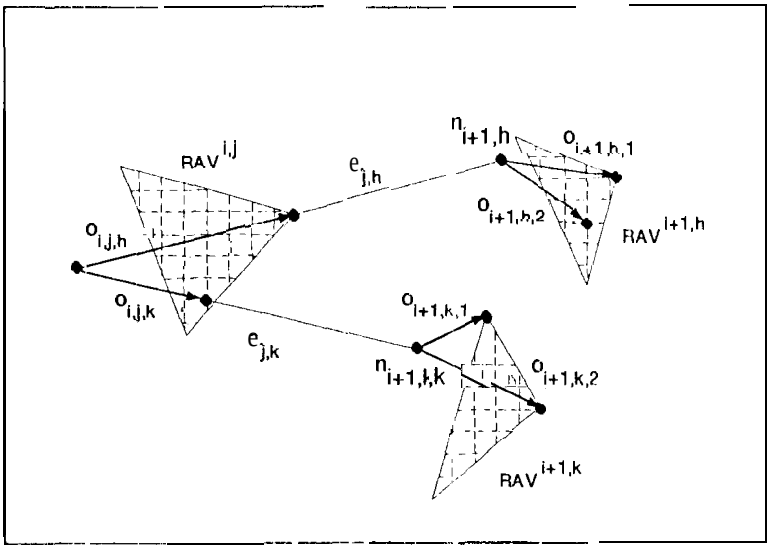

Figure 12: 'Tree representation for the global search.

the edges emanating from node n_{ij}, i.e. the avoidance maneuvers from the position corresponding to node $n_{i,j}$ over the interval T. The positions reached by the robot at the end of each maneuver are the successors of node $n_{i,j}$. A node $n_{i,j}$ is completely expanded when all the operators $o_{i,j,l}$ have been applied and all tile edges emanating from $n_{i,j}$ have been examined. The resulting tree has a constant time interval between nodes, and a variable branch number that is a function of the shape of each RAV. This tree is searched for the trajectory that maximizes any objective function. such as as motion time, distance traveled or energy, using standard search techniques [27,25]. Figure12 shows schematically a subtree of some avoidance maneuvers.

Since the avoidance maneuvers are selected based on the velocity obstacles VO, they are never on a collision course with any of the considered obstacles, as discussed earlier. 'This excludes trajectories that might, part of the time, be on a collision course with some of the obstacles. However, such trajectories may be generated by either considering only obstacles with imminent collisions, or by optimizing the trajectory using dynamic,

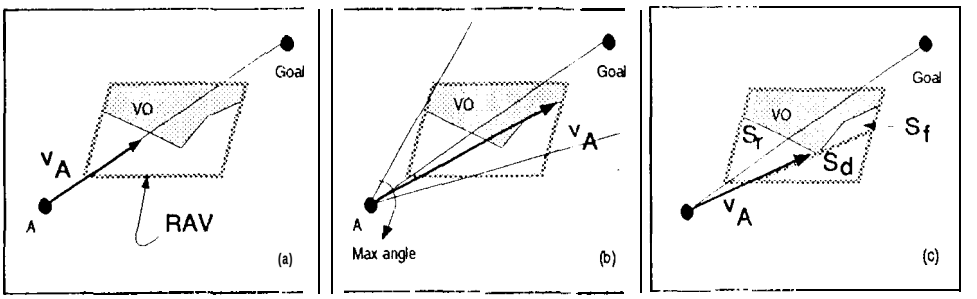

Figure 13: a: TG strategy. b: MV strategy. c: ST strategy.

optimization [11].

4.2. Heuristic Search

The search for an avoidance trajectory can be reduced to a heuristic search for on-line applications, or when only incomplete information about the environment is available. The heuristics can be chosen to satisfy a prioritized series of goals, with survival of the robot being the primary goal, and other goals including reaching the desired target, minimizing a performance index, and implementing a desired trajectory structure. The survival of the robot can be guaranteed by selecting the avoidance velocities RAV; the target can be reached by selecting velocities that point towards the destination; the motion time can be minimized by choosing the highest velocity available; and the desired trajectory structure can be selected by choosing an appropriate sequence of front and rear avoid ante maneuvers.

We thus propose the following basic heuristics:

(a) Choose the highest avoidance velocity along the line to the goal, as shown in Figure 13-a, so that the trajectory takes the robot towards its target. This strategy is denoted in the following by TG (to goal).

(b) Select the maximum avoidance velocity within some specified angle α from the line to

the goal, as shown in Figure 13-b, *so* that the robot moves at high speed, even if does not aim directly at the goal. This strategy is henceforth called MV (maximum velocity).

(*c*) Select the velocity that avoids the obstacles according to their perceived risk. This strategy is called ST (structure). In Figure 13-c the chosen velocity is the highest among the rear avoidance velocities.

Other heuristics may combine some or all of the above strategies. For example, the avoidance velocity could be chosen as the fastest velocity pointing towards the target and generating a rear avoidance maneuver. Furthermore, it might be advantageous to switch between heuristics, in order to yield trajectories that better satisfy the prioritized goals.

5. Examples

5.1. A Single Avoidance Maneuver

This example demonstrates a single avoidance maneuver computed using the velocity obstacle method, as shown in Figure 14. The environment consists five circles, representing a robot and four moving obstacles. The velocities are represented by bars attached to the centers of the robot and of the obstacles. The position and velocity of each object are as follows:

robot: $(x = 5.0,\ y = 5.0), (v_x = 8.0,\ v_y = 5.0),\ r = 5.0,$

obstacle 1: $(x = 90,\ y = 40),\ (v_x = -12{,}0,\ v_y = -1.0),\ r = 5.0,$

obstacle 2: $(x = 60,\ y = -5),\ (v_x = -5.0,\ v_y = 5.5),\ r = 5.0,$

obstacle 3: $(x = -30,\ y = -20),\ (v_x = -5.0,\ v_y = 3{,}0),\ r = 5.0,$

obstacle 4: $(x = -10,\ y = 40),\ (v_x = 4.0,\ v_y = -1.0),\ r = 5.0.$ These velocities result in the trajectory shown in Figure 15. The trajectories are represented by circles displayed

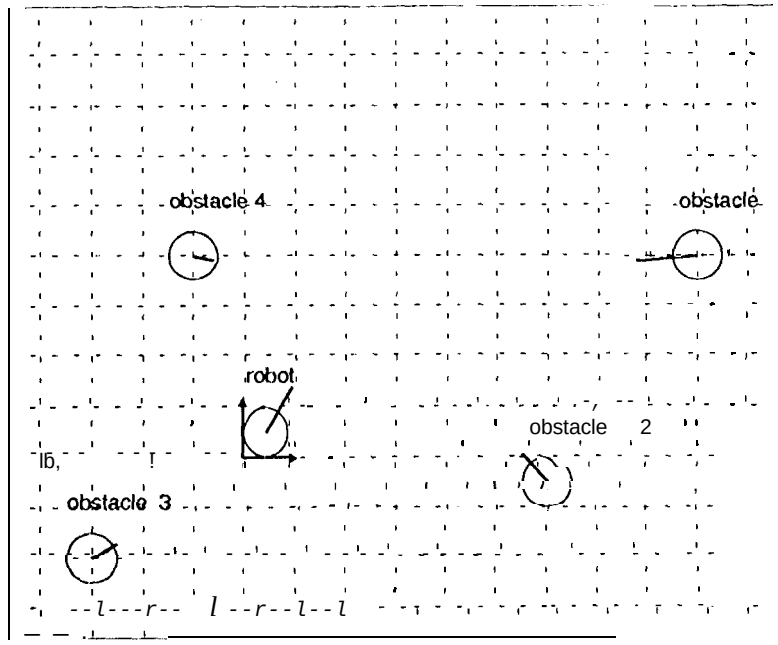

Figure 14: Initial configuration of the robot aud the obstacles.

every one second. Obstacle 2 is on a collision course with the robot with which it collides approximately at $t = 5.\$$, as shown by the grey circle in Figure 15.

Computing a single avoidance maneuver resulted in the trajectory shown in Figure 16, where the robot avoids obstacle 4 with a front maneuver, obstacle 2 with a rear maneuver, and obstacle 3 with a right maneuver

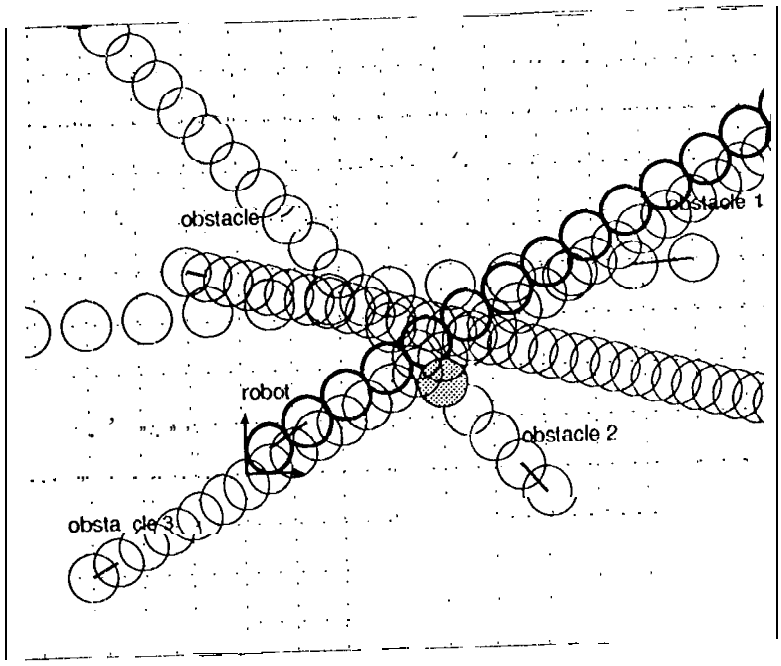

Figure 15: Simulation with the initial velocities.

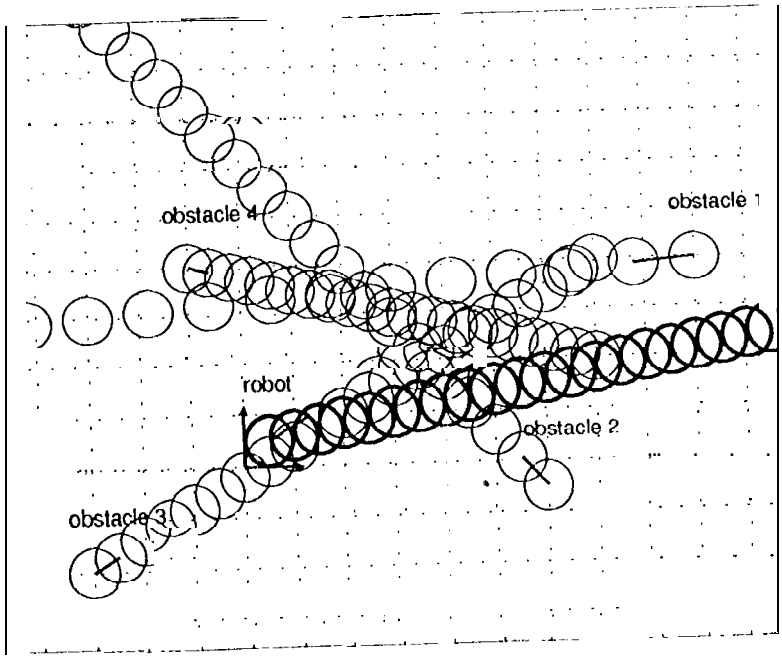

Figure 16: Simulation of an avoidance maneuver.

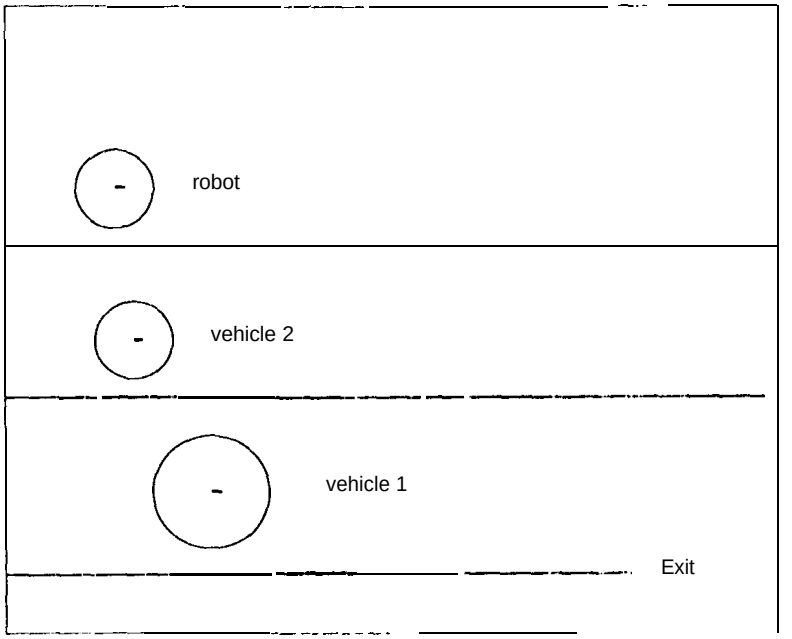

Figure 17: Example of trajectory problem.

5.2. Complete Avoidance Trajectories

In this example, the computation of a complete trajectory is demonstrated for the Intelligent Highway scenario shown in Figure 17. The goal of the robotic vehicle in the leftmost lane is to reach the exit to the right without colliding with the other two vehicles. 'The initial velocity of the robot and of vehicle 2 is $(v_x = 30.0\ m/s, v_y = 0.0\ m/s)$, and the initial velocity of vehicle 1 is $(v_x = 23\ m/s, v_y = 0.0\ m/s)$. The velocities are represented as bars attached to the center of each circle.

For the global search used in the example of Figure 17, the time interval T between nodes is chosen arbitrarily as $T = 1$s.

The trajectory shown in Figure 18 is computed using the Depth First Iterative Deepening algorithm [20]. 'This algorithm returns the fastest path to the target, the first time it reaches the target. In the example, the RAV sets have been discretized by considering four points on each side of the boundary, and the maximum feasible velocity in the di-

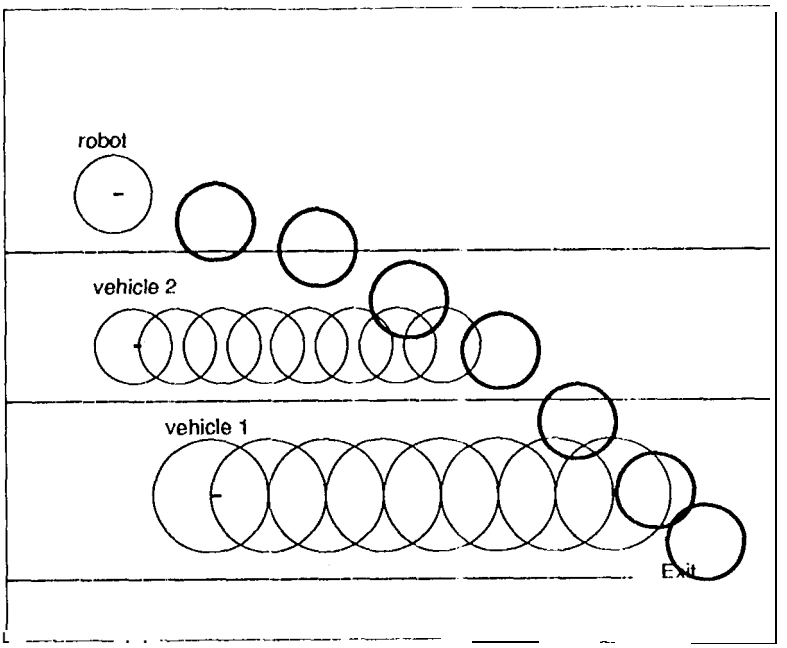

Figure 18: Trajectory computed by global search.

rection to the goal. The search has been expanded to **4.5** s and a depth of **5** levels. The total motion time of this solution is of 3.5 s.

Using the TG strategy resulted in the trajectory shown in Figure 19. Along this trajectory, the robot slows down and lets vehicle 2 pass, and then speeds up towards the exit, behind vehicle 1. 'he total motion time for this trajectory is 6.07 s.

Using the MV heuristics resulted in the trajectory shown in Figure 20. Along this trajectory, the robot speeds up and passes in front of both vehicles 1 and 2. The total motion time along this trajectory is 3.56 s.

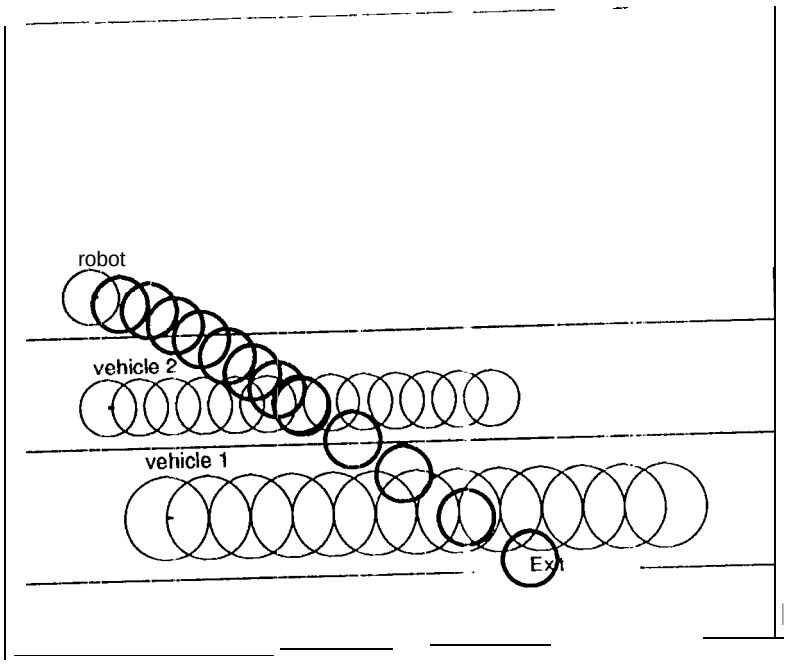

Figure 19: Trajectory computed with the TG strategy.

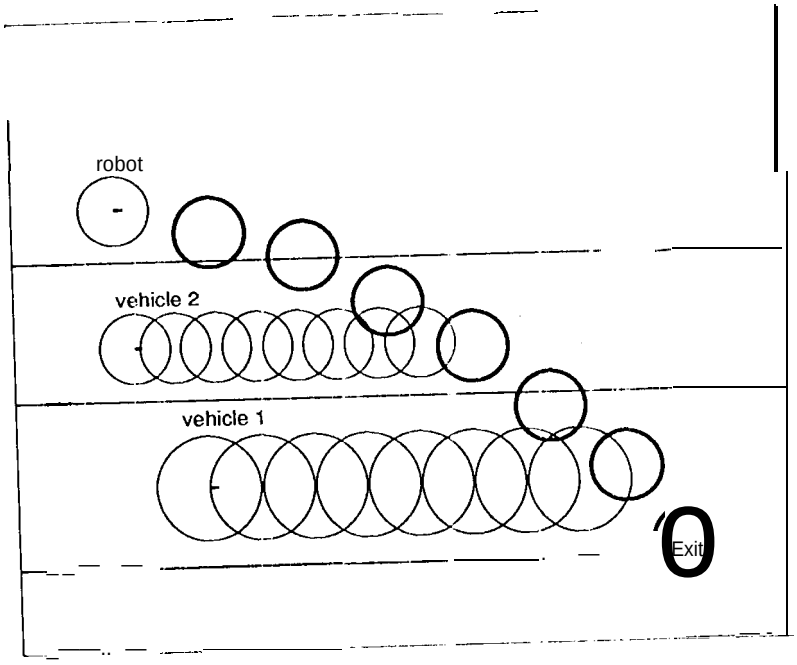

Figure 20: Trajectory computed with MV strategy.

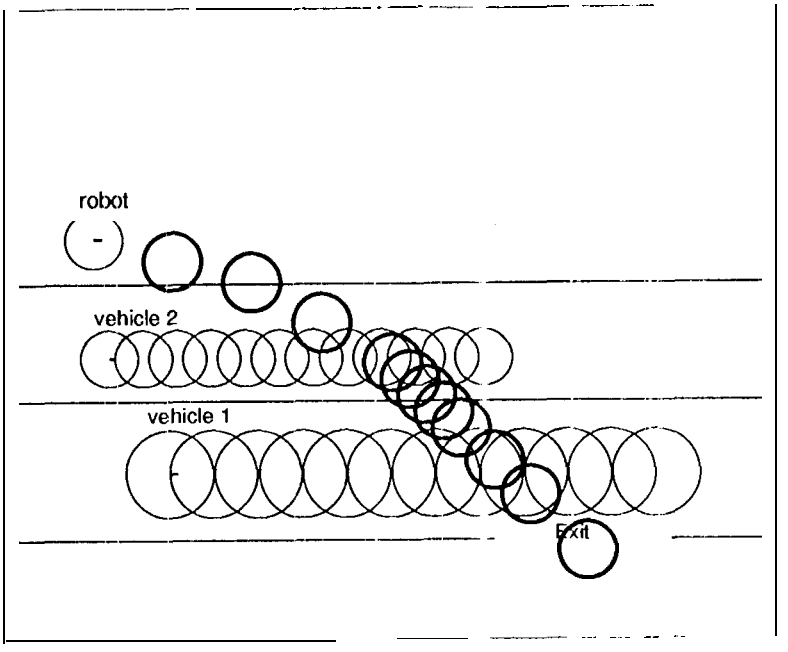

Figure 21: Trajectory computed with both TG and MV strategies.

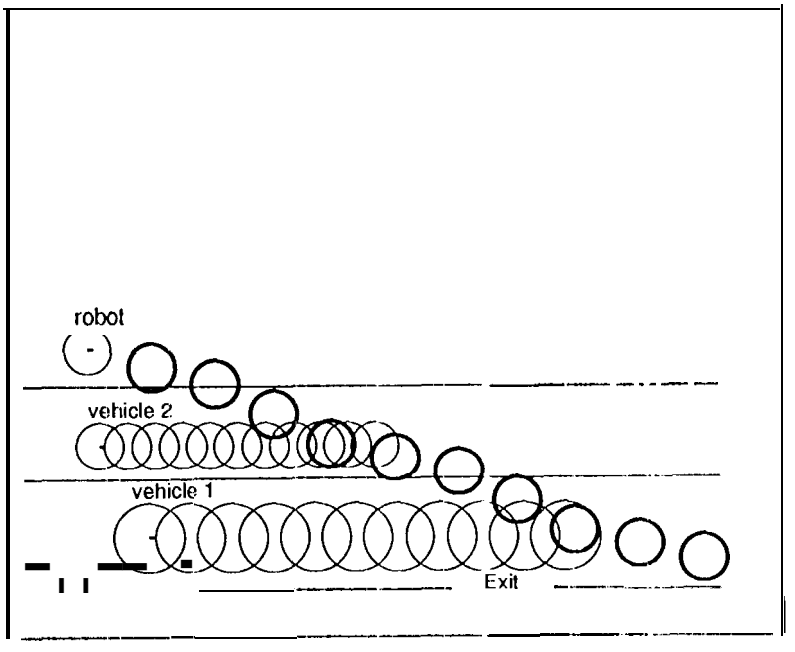

Figure 22: Failed trajectory computed with the MV heuristics,

In the case of Figure 20, the MV heuristics managed to reach the goal. The trajectory computed using both MV and TG strategies is shown in Figure 21. Along this trajectory, the robot first speeds up to pass vehicle 2, then slows down to let vehicle 1 pass on, and then speeds up again towards the goal. The motion time for this trajectory is 5.31 s. This trajectory was computed by using the MV heuristics for $0 \leq t \leq 2.0$ s and the TG heuristics afterwards.

Figure 22 shows a trajectory computed with the MV strategy that missed the exit target because of a small increase in the velocity of vehicle 1 to $(v_x = 33\ m/s, v_y = 0.0\ m/s)$. Nevertheless, the trajectory is collision free.

These examples show that both global and heuristic search strategies perform well in computing the complete trajectories. The motion time of the MV solution compares well with that of the global search. Although this result cannot be generalized, it shows the potential of the heuristic strategies for on-line applications.

6. Conclusion

A new method for planning the motion of a robot moving in a time-varying environment, the Velocity obstacle approach, has been presented. It is significantly different from currently available planning algorithms, since it simultaneously computes the path and velocity profile that avoid all static and moving obstacles and satisfy the robot's dynamic constraints.

The method consists of computing, for every obstacle, its corresponding velocity obstacle, which is the set of colliding velocities between the obstacle and the robot. Then, by subtracting it from the! reachable velocities of t he robot, the set of reachable avoidance

velocities is formed, which consists of all the velocities that avoid the obstacles and satisfy the robot's dynamic, constraints. A search space is then formed by representing the state space of the robot by a tree of avoidance maneuvers. A global search over the tree yields trajectories that minimize a selected performance index, such as motion time or traveled distance. For on-line applications, the tree is pruned rising one of several heuristic strategies, aimed at satisfying a prioritized list of goals, such as the survival of the robot, reaching the target, and minimize motion time. 'Tile method is demonstrated for planning the trajectory of an automated vehicle in an Intelligent Vehicle Highway System scenario.

The main advantages of the velocity obstacle approach include the efficient geometric representation of maneuvers avoiding any number of moving and static obstacles, and the simple consideration of robot dynamics and actuator constraints. The solutions computed with this method are conservative, since each maneuver avoids all obstacles, irrespectively of their expected collision time, and they may exclude trajectories that, although feasible, violate the velocity obstacle in some interval.

REFERENCES

1. W. Burke. *Applied Differential Geometry.* Cambridge University Press, 1985.

2. S. Cameron. A study oft he clash detection problem in robotics. In *IEEE International Conference 071 Robotics and Automation,* pages *488--493,* St. Louis, MO, March *25-28 1985.*

3. S. Cameron. Efficient intersection tests for objects defined constructively. *The International Journal of Robotics Research,* 8(1):3–25, February 1989.

4. S. Cameron. Collision detection by four-dimensional intersection testing. *IEEE Journal oj Robotics and Automation,* 6(3):291–302, June 1990.

5. S.A. Cameron. *Modelling Solids in Motion.* PhD thesis, Edinburgh, UK, 1984.

6. J. Canny. The *Complexity of Robot Motion Planning.* MIT Press, Boston, MA, 1988.

7. J. Canny and J. Reif. New lower bound techniques for robot motion planning problems. In *28th IEEE Symposium on Foundation of Computer Science, 1987.*

8. J. Canny, J. Reif, B. Donald, and P. Xavier. On the complexity of kinodynamic planning. In *29th IEEEE Symposium* on *Foundation of Computer Science, 1988.*

9. M. Erdmann and T. Lozano-Perez. On multiple moving objects. 1 n *IEEE International Conference on Robotics and Automat ion,* pages 1419--1424, San Francisco, CA, April 7-101986.

10. M. Erdmann and T. Lozano-Perez. On multiple moving objects. *Algorithmica,* (2):477--521, 1987.

11. P. Fiorini. *Robot Motion Planning among Moving Obstacles.* PhD thesis, University of California, Los Angeles, January 1995.

12. P. Fiorini and Z. Shiller. Motion planning in dynamic environments using' the relative velocity paradigm. In *IEEE International Conference of Automation and Robotics, volume 1,* pages .*560 --566,* May 1993.

13. K. Fujimura. On motion planning amidst transient obstacles. In *IEEE International Conference on Robotics ana! Automation,* pages *1488--1493,* Nice, France, May 1992.

14. K. Fujimura. Motion planning amid transient obstacles. *International Journal of Robotics Research,* 13(5):395–407, October 1994.

15. K. Fujimura and H. Samet. Time-minimal paths among moving obstacles. In *IEEE International Conference on Robotics and Automation,* pages 1110-1115, Scottsdale, AZ, May 1989.

16. K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *IEEE International Conference on Robotics and A utomation,* pages *324-330,* Cincinnati, 011, May 1990.

17. V. Hayward, S. Aubry, A. Foisy, and Y. Ghallab. Efficient collision prediction among m auy moving obstacles. *International Journal oj Robotics Research, 14(2): 129-143,* April 1995.

18. K. Kant and S.W. Zucker. Towards efficient trajectory planning: the path-velocity decomposition. The *International Journal of Robotic Research,* 5(3):72–89, Fall 1986.

19. K. Kant and S.W. Zucker. Planning collision free trajectories in time-varying environments: a two level hierarchy, In *IEEE International Conference on Robotics* and *Automation,* pages 1644--1649, Raleigh, NC, 1988.

20. R. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *A rtificial Intelligence,* (27):97–1 *09, 1985.*

21. B.H.Lee and C.S.G.Lee. Collision-free motion planning; of two robots. *IEEE* Transactions *on System Man and Cybernetics*, SMC-1 7(1):21- 32, January-February 1987.

22. 'T. Lozano-Pérez. Spatial planning: a configuration space approach. *IEEE Transaction on Computers, C-32(2):108--120,* February 1983.

23. T.Lozano-Pérez. A simple motion-planning algorithm for genera] robot manipulators. *IEEE Journal of Robotics and Automation,* RA-3(3):224- *238,* June 1987.

24. T.Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the A CM,* 22(10):560–570, October 1979.

25. N Nilsson. *Principles of Artificial Intelligence.* Tioga, Palo Alto, CA, 1980.

26. C.Ó'Dúnlaing. Motion planning with inertial constraints, Technical Report TR-230, NYU Robotics Laboratory, 1986.

27. J. Pearl. *Heuristics.* Addison Wesley Publishing, Co., Reading, MA, 1985.

28. J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. in *25th IEEE Symposium on the Foundation of Computer Science,* pages 144-] 53, 1985.

29. J.C.Sanborn and J.A.Hendler. A model of reaction for planning in dynamic environments. *International Journal of Artificial Intelligence in Engineering,* 3(2):95–1 01, April 1988.

30. Z. Shiner and Dubowsky. Time optimal paths and acceleration lines of robotic manipulators. In *The 26th IEEE Conference on Decision and Control,* Los Angeles, CA, December 1987.

Appendix

Proof of **Lemma** 1:

The proof is carried out for $\widehat{B_1}$ since there is a one-to-one correspondence between $\widehat{B_1}$ and B_1. Furthermore, only the front side of $\widehat{B_1}$, $\partial(B_{1f})$, is analyzed, since the rear side, $\partial(B_{1r})$, is completely analogous. First, the necessary aud sufficient conditions of the lemma are proven by contradiction. Then a geometric construction is used to define the map between the tangent velocities and the corresponding points on $\partial(B_{1f})$.

To prove the necessary condition, let us assume that there exists a velocity $\mathbf{v}_A$ tangent to obstacle $\widehat{B_1}$ such that its corresponding relative velocity, $\mathbf{v}_{A,B_1}$ is not on the boundary of the collision cone CC_{A,B_1}, or:

$$\exists\, \mathbf{v}_A \text{ tangent to } \widehat{B_1} \mid \mathbf{v}_{A,B_1} = (\mathbf{v}_A - \mathbf{v}_{B_1}) \notin \lambda_f$$

Then $\mathbf{v}_{A,B_1}$ would be either inside or out Side CC_{A,B_1}. In the first case, by definition of CC_{A,B_1}, it would correspond to a collision maneuver, whereas in the second case, it would correspond to an avoidance maneuver, thus contradicting the hypothesis that $\mathbf{v}_A$ is tangent to B_1.

'The sufficient condition can be proved similarly, assuming; that there exists a relative velocity $\mathbf{v}_{A,B_1}$ on the boundary of the collision cone CC_{A,B_1}, that does not generate a maneuver tangent to $\overline{\widehat{B_1}}$, or:

$$\exists\, \mathbf{v}_{A,B_1} \in \lambda_f \text{ such that } \mathbf{v}_A = (\mathbf{v}_{A,B_1} - \mathbf{v}_{B_1}) \text{ not tangent to } \widehat{B_1}$$

Then, $\mathbf{v}_A$ would be either colliding or avoiding, and thus it will be either inside or outside the velocity obstacle VO. This would contradict the definition of VO, which is computed

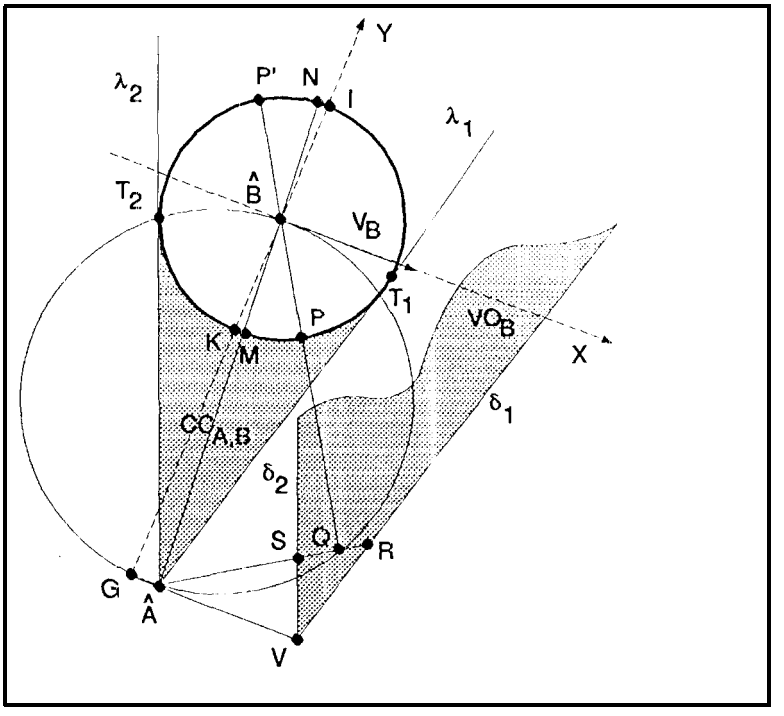

Figure 23: Contact points of tangent trajectories.

by translating the collision cone CC_{A,B_1} by the velocity of B_1, $\mathbf{v}_{B_1}$. In particular, all velocities on λ_f become velocities with the tip on δ_f.

To prove the second part of the Lemma, we compute the tangency points that correspond to each tangent velocity $\mathbf{v}_{A,B_1} \in \lambda_f$. The tangent relative velocities are represented by the semi-infinite line $\{\hat{A}, \lambda_f\}$, with minimum and maximum relative velocities given by $|\mathbf{v}_{A,B_1}| \rightarrow$ O, and $|\mathbf{v}_{A,B_1}| \rightarrow \infty$, respectively. 'The former velocity corresponds to the limit point Y_f, when A and B_1 move on parallel trajectories. The latter velocity corresponds to the limit point T_f.

The correspondence between any other velocity O < $\mathbf{v}_{A,B_1}$ < cm, and a tangency point $P \in (T_f, Y_f)$ is established by using the orthogonality between a tangent trajectory t, due to $\mathbf{v}_A = (\mathbf{v}_{A,B_1} - \text{VII},)$, and the line 1 through i he center of $\widehat{B_1}$ and the tangency

point,. The right angle between t and 1 can be embedded into the. circle C of diameter $(\hat{A},\widehat{B_1})$, as shown in Figure 23. Thus, given a desired tangency point $P \in \partial(B_{1f})$, circle C maps it into the direction of the corresponding tangency trajectory t, represented by the line through $\hat{A}$ and Q, as shown in Figure 23. Similarly, given a desired tangency direction t, circle $\mathcal{C}$ maps it into point P, represented by the intersection of $\partial(B_{1f})$ with the line through the center of $\widehat{B_1}$ and Q.

The magnitude of the velocity $\mathbf{v}_A$ tangent to $\widehat{B_1}$ in P is found using the velocity obstacle VO, *whose* boundary δ_f represents all front tangent velocities. Then, the intersection R of the trajectory line t with δ_f gives the magnitude of $\mathbf{v_A}$. The magnitude Of the corresponding relative velocity follows from the definition $\mathbf{v}_{A,B_1} = \mathbf{v}_A - \mathbf{v}_{B_1}$.

Note that the magnitude of the velocity $\mathbf{v}_A$ tangent to $\widehat{B_1}$ in $P' \in \partial(B_{1r})$ is simply given by the intersection S oft and δ_r.

■

Proof of **Lemma 2:**

The subsets S_f and S_r of the reachable avoidance velocities RAV, include in their boundary a segment of the boundary of the velocity obstacle VO_{B_1}, $\partial(VO_{B_1})$. Subset S_d, instead, does not include in its boundary any segment from the boundary of VO_{B_1}. From Lemma 1, velocities with tip on these segments generate trajectories grazing B_1, and therefore only subsets S_f and S_r include tangent velocities.

Tangent velocities separate collision velocities from avoidance velocities, and therefore, a velocity $\mathbf{v}_A \in \{S_f, S_r\}$, whose tip is away from the boundary $\partial(VO_{B_1})$ will generate a trajectory passing at a certain distance from B_j, on the same side of the tangent trajectory.

Similarly, trajectories generated by velocities $\mathbf{v}_A \in S_d$ cannot be tangent to B_1 in a finite time, since the boundary of S_d is the velocity parallel to $\mathbf{v}_{B_1}$. Therefore, if this subset exists in RAV, it contains all the velocities diverging from B_1.

■

Proof of **Theorem** 1:

From Lemma 2, each velocity obstacle VO_{B_1} divides the set of reachable avoidance velocities RAV into three non-overlapping subsets. Then, m obstacles, $B_j (j = 1,..., m)$, generate $3m$ subsets, that are partially overlapping. Let us assume that RAV is divided into 1 non-overlapping subsets $RAV_i i = 1, \ldots . 1.$

The boundary of each subset, $\partial(RAV_i)$, may include segments from the boundaries of some velocity obstacles $VO_{B_j}, \partial(VO_{B_j})$. From Lemma **2,** all velocities in the same subset RAV_i will generate the same type of avoidance maneuver. Then, if the boundary of a $RAV_i \partial(RAV_i)$, includes segments from the boundary of n different velocity obstacles VO_{B_j} $(j = 1, \ldots, n; n \leq m)$, all velocities $\mathbf{v}_A \in RAV_i$ will generate trajectories avoiding each of the n obstacles $B_j (j = 1, \ldots . n; n \leq m)$ with the maneuver determined by the portion of $\partial(VO_{B_j})$ included in $\partial(RAV_i)$. *Then,* each subset RAV_i corresponds to a specific maneuver type, represented by indices i^j, where the index i represents the type of avoidance velocity, i.e. (i= f,r,d), and j, $(j = 1, \ldots . m)$ is the index of the obstacle.

■