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 method for robot motion planning in dynamic environments. It consists of selecting avoidance maneuvers to avoid static and moving obstacles in the velocity space, based on the *current* positions and velocities of the robot and obstacles. It is a first-order method since it does not integrate velocities to yield positions as functions of time.

The avoidance maneuvers are generated by selecting robot velocities outside of the Velocity Obstacles, which represent the set of robot velocities that would result in a collision with a given obstacle that moves at a given velocity, at some future time. To ensure that the avoidance maneuver is dynamically feasible, the set of avoidance velocities is intersected with the set of admissible velocities, defined by the robot's acceleration constraints. Computing new avoidance maneuvers at regular time intervals accounts for general obstacle trajectories.

The trajectory from start to goal is computed by searching a tree of feasible avoidance maneuvers, computed at discrete time intervals. An exhaustive search of the tree yields near-optimal trajectories that either minimize distance or motion time. A heuristic search of the tree is applicable to on-line planning. The method is demonstrated for point and disk robots among static and moving obstacles, and for an automated vehicle in an Intelligent Vehicle Highway System scenario.

1. Introduction

This paper addresses the problem of motion planning in dynamic environments, such as robot manipulators avoiding moving obstacles, and intelligent vehicles negotiating freeway traffic.

Motion planning in dynamic environments is a difficult problem, since it requires planning in the state-space, i.e. simultaneously solving the *path planning* and the *velocity planning* problems. Path planning is a kinematic problem, involving the computation of a collision-free path from start to goal, whereas velocity planning is inherently a dynamic problem, requiring the consideration of robot dynamics and actuator constraints. It was shown that dynamic motion planning for a point in the plane, with bounded velocity and arbitrary many obstacles, is intractable (NP-hard) (Canny and Reif, 1987). In addition to the computational difficulty, this problem is not guaranteed to have a solution due to the uncertain nature of the environment, since a solution computed at time t_0 may be infeasible at a later time (Sanborn and Hendler, 1988).

Motion planning in dynamic environments was originally addressed by adding the time dimension to the robot's configuration space, assuming bounded velocities and known trajectories of the obstacles (Reif and Sharir, 1985; Erdmann and Lozano-Perez, 1987; Fujimura and Samet, 1989a). Reif and Sharir (1985) solved the planar problem for a polygonal robot among many moving polygonal obstacles, by searching a visibility graph in the configuration-time space. Erdmann and Lozano-Pérez (1987) discretized the configuration-time space to a sequence of slices of the configuration space at successive time intervals. This method essentially solves the static planning problem at every slice and joins adjacent solutions. Fujimura and Samet (1989a) used a cell decomposition to represent the configuration-time space, and joined empty cells to connect start to goal.

Another approach to dynamic motion planning is to decompose the problem into smaller sub-problems: path planning and velocity planning. This method first computes a feasible path among the static obstacles. The velocity along the path is then selected to avoid the moving obstacles (Kant and Zucker, 1986; Lee and Lee, 1987; Fraichard, 1993; Fraichard and Laugier, 1993; Fujimura and Samet, 1993; Fujimura, 1995). Kant and Zucker (1986) selected both path and velocity profile using a visibility graph approach. Lee and Lee (1987) developed independently a similar approach for two cooperating robots, and compared the effects of delay and velocity reduction on motion time. Fraichard (1993) considered acceleration bounds, and used a search in a state-time space $(s, \dot{s}, t)$ to compute the velocity profile yielding a minimum-time trajectory. Fraichard and Laugier (1993) considered adjacent paths that could be reached from the nominal path when the nominal path becomes blocked by a moving obstacle. Fujimura (1995) considered the case of a robot moving on a fixed time-dependent network, whose nodes could be temporarily occluded by moving obstacles.

A different approach consists of generating the accessibility graph of the environment, which is an extension of the visibility graph (Fujimura and Samet, 1989b; Fujimura and Samet, 1990). Fujimura and Samet (1989b) defined it as the locus of points on the obstacles which are reachable by the robot moving at maximum speed. These points form the *collision front*, and can be linked together to construct a path from start to goal. The accessibility graph has the property that, if the robot moves faster than the obstacles, the path computed by searching the graph is time-minimal. This concept was extended in (Fujimura, 1994) to the case of slowly moving robots and transient obstacles, i.e. obstacles that could appear and disappear in the environment.

On-line planning in dynamic environments has mostly emphasized reasoning and decision making, with little concern to robot dynamics. In (Sanborn and Hendler, 1988),

reaction rules were developed 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. This approach considers the physical properties of the environment, but neglects the physical limitations of the moving robot. Bounds on the robot's velocity were added in (Tsubouchi and Arimoto, 1994), where a Cartesian-time representation was used to avoid the volumes swept by the moving obstacles. A problem related to on-line avoidance of many obstacles, namely predicting and ordering future collisions according to time to collision, was addressed in (Hayward et al., 1995).

Common to the previous work is the reliance on position information to test for collision between the robot and the moving obstacles. For example, the configuration-time space (Erdmann and Lozano-Perez, 1987; Fujimura and Samet, 1989a; Reif and Sharir, 1994) is essentially a time evolution of the configuration-space obstacles. The path-velocity decomposition (Kant and Zucker, 1986; Fujimura and Samet, 1989b; Fraichard and Laugier, 1993) is based on testing for collisions at various positions along the path. Similarly, the collision front (Fujimura and Samet, 1989b; Fujimura and Samet, 1990) is generated by integrating velocities to produce colliding positions between the robot and the obstacles. Thus, all current algorithms can be described as *zero order* methods, since they rely on position information to determine potential collisions.

In this paper, we present a *first order* method to compute the trajectories of a robot moving in a time-varying environment, using velocity information to directly determine potential collisions. This method utilizes the concept of Velocity Obstacle (VO) (Fiorini and Shiller, 1993; Fiorini, 1995), which maps the dynamic environment into the robot velocity space. The velocity obstacle is a first-order approximation of the robot's velocities that would cause a collision with an obstacle at some future time, within a given time horizon. With this representation, an avoidance maneuver can be computed simply by selecting velocities outside of the velocity obstacle. The maneuver is ensured to be dynamically feasible by selecting velocities that satisfy the actuator constraints, mapped into velocity constraints using forward dynamics.

A trajectory consists of a sequence of such avoidance maneuvers, computed by a global search 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 achieving trajectories with desired topologies. Both methods were previously implemented in computer simulations for an autonomous vehicle negotiating freeway traffic (Fiorini, 1995; Fiorini and Shiller, 1995), for a robotic manipulator avoiding moving obstacles (Fiorini and Shiller, 1996; Fiorini and Shiller, 1997), and extended to a 3-D environment (Fiorini and Shiller, 1993). The analysis and examples presented in this paper focus on a point robot among static and moving obstacles, and on on automated vehicle applications.

This paper is organized as follows. Section 2 presents the velocity obstacle. Section 3 describes the structure of the avoidance maneuvers. The generation of complete trajectories is described in Section 4 using both global and heuristic search methods. Examples of trajectories for a point robot avoiding static and moving obstacles, and for a circular robot navigating highway traffic, are presented in Section 5.

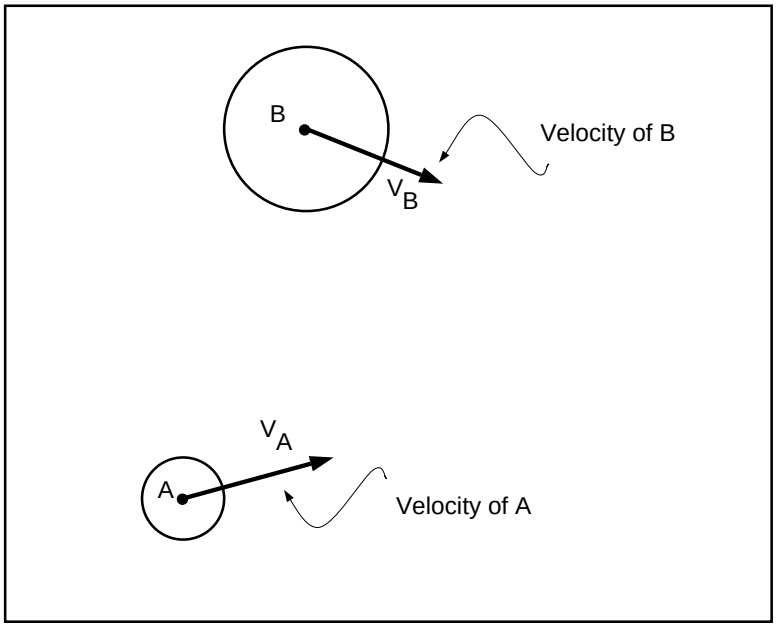

Figure 1: The robot and a moving obstacle.

2. The Velocity Obstacle

In this section, we present the concept of *Velocity Obstacle* (VO) for a single and multiple obstacles. For simplicity, we restrict our analysis to circular robots and obstacles, thus considering a planar problem with no rotations. This is not a severe limitation since general polygons can be represented by a number of circles (O'Rourke and Badler, 1979; Featherstone, 1990). We also assume that the obstacles move along arbitrary trajectories, and that their instantaneous state (position and velocity) is either known or measurable.

Consider the two two circular objects, A and B, shown in Figure 1 at time t_0, with velocities $\mathbf{v}_A$ and $\mathbf{v}_B$. Let circle A represent the robot, and circle B represent the obstacle. To compute the VO, we first map B into the *Configuration Space* of A, by reducing A to the point $\widehat{A}$ and enlarging B by the radius of A to $\widehat{B}$. The state of each moving object is represented by its position and a velocity vector attached to its center.

We define the *Collision Cone*, $CC_{A,B}$, as the set of colliding relative velocities between $\widehat{A}$ and $\widehat{B}$:

$$CC_{A,B} = \{\mathbf{v}_{A,B} \mid \lambda_{A,B} \cap \widehat{B} \neq \emptyset\} \tag{1}$$

where $\mathbf{v}_{A,B}$ is the relative velocity of $\widehat{A}$ with respect to $\widehat{B}$, $\mathbf{v}_{A,B} = \mathbf{v}_A - \mathbf{v}_B$, and $\lambda_{A,B}$ is the line of $\mathbf{v}_{A,B}$.

This cone is the planar sector with apex in $\widehat{A}$, bounded by the two tangents λ_f and λ_r from $\widehat{A}$ to $\widehat{B}$, as shown in Figure 2. Any relative velocity that lies between the two tangents to $\widehat{B}$, λ_f and λ_r, will cause a collision between A and B. Clearly, any relative velocity outside $CC_{A,B}$ is guaranteed to be collision-free, provided that the obstacle $\widehat{B}$ maintains its current shape and speed.

The collision cone is specific to a particular pair of robot/obstacle. To consider multiple obstacles, it is useful to establish an equivalent condition on the *absolute* velocities of A. This is done simply by adding the velocity of B, $\mathbf{v}_B$, to each velocity in $CC_{A,B}$ or,

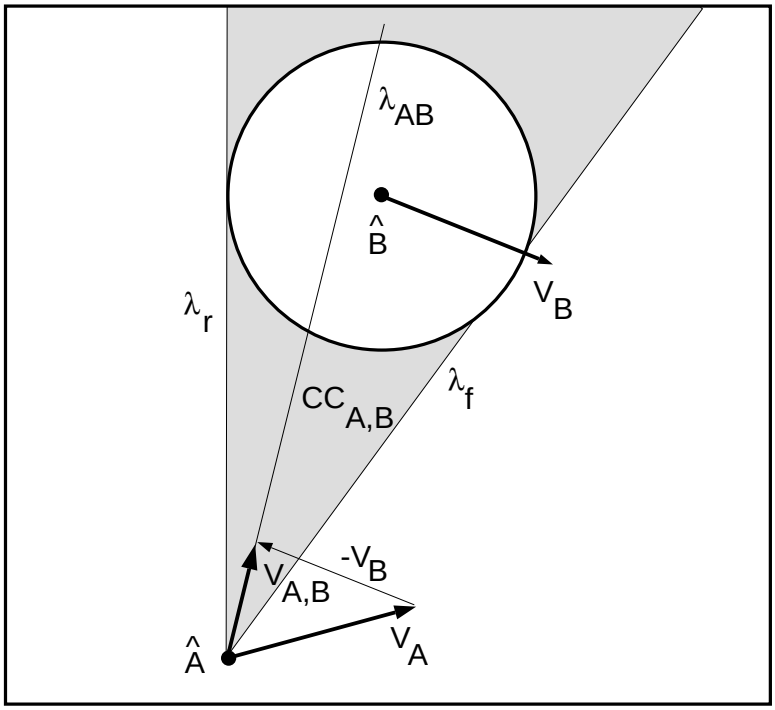

Figure 2: The Relative Velocity $\mathbf{v}_{A,B}$ and the Collision Cone $CC_{A,B}$.

equivalently, by translating the collision cone $CC_{A,B}$ by $\mathbf{v}_B$, as shown in Figure 3. The *Velocity Obstacle* VO is then defined as:

$$VO = CC_{A,B} \oplus \mathbf{v}_B \tag{2}$$

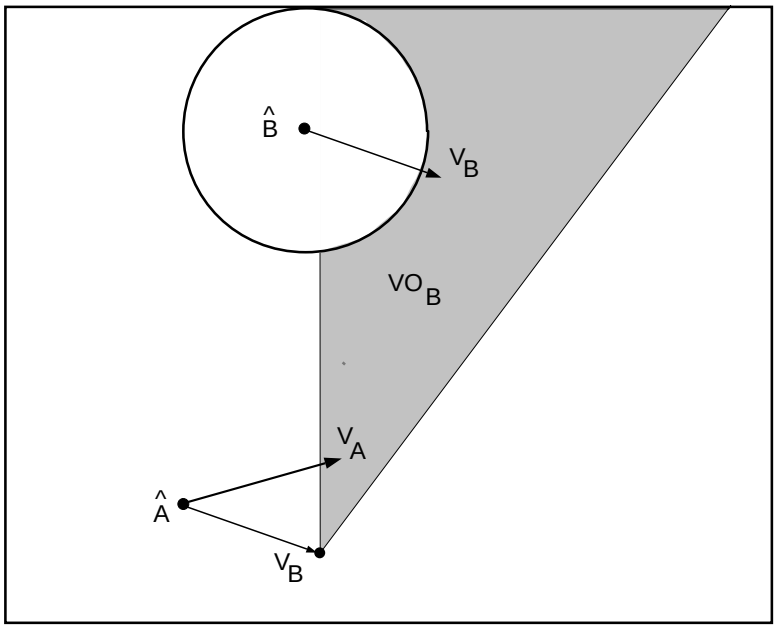

Figure 3: The velocity obstacle VO_B.

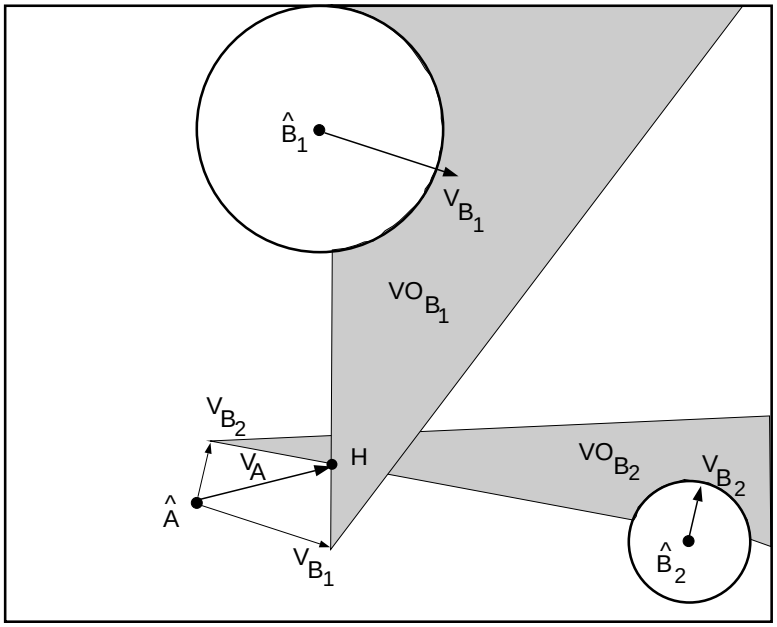

Figure 4: Velocity obstacles for B_1 and B_2.

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

The VO partitions the absolute velocities of A into *avoiding* and *colliding* velocities. Selecting $\mathbf{v}_A$ outside of VO would avoid collision with B, or:

$$A(t) \cap B(t) = \emptyset \quad \text{if } \mathbf{v}_A(t) \notin VO(t) \tag{3}$$

Velocities on the boundaries of VO would result in A grazing B. Note that the VO of a stationary obstacle is identical to its relative velocity cone, since then $\mathbf{v}_B = 0$.

To avoid multiple obstacles, we consider the union of the individual velocity obstacles:

$$VO = \cup_{i=1}^{m} VO_{B_i} \tag{4}$$

where m is the number of obstacles. The avoidance velocities, then, consist of those velocities $\mathbf{v}_A$, that are outside all the VO's, as shown in Figure 4.

In the case of many obstacles, it may be useful to prioritize the obstacles so that those with imminent collision will take precedence over those with long time to collision. Furthermore, since the VO is based on a linear approximation of the obstacle's trajectory, using it to predict remote collisions may be inaccurate, if the obstacle does not move along a straight line. We call *imminent* a collision between the robot and an obstacle if it occurs at some time $t < T_h$, where T_h is a suitable time horizon, selected based on system dynamics, obstacle trajectories, and the computation rate of the avoidance maneuvers.

To account for imminent collisions, we modify the set VO by subtracting from it the set VO_h defined as:

$$VO_h = \{\mathbf{v}_A \,|\, \mathbf{v}_A \in VO\,,\, \| \mathbf{v}_{A,B} \| \leq \frac{d_m}{T_h}\} \tag{5}$$

where d_m is the shortest relative distance between the robot and the obstacle. The set VO_h represents velocities that would result in collision, occurring beyond the time horizon.

Figure 5 shows the modified VO, where the velocity set VO_h has been removed.

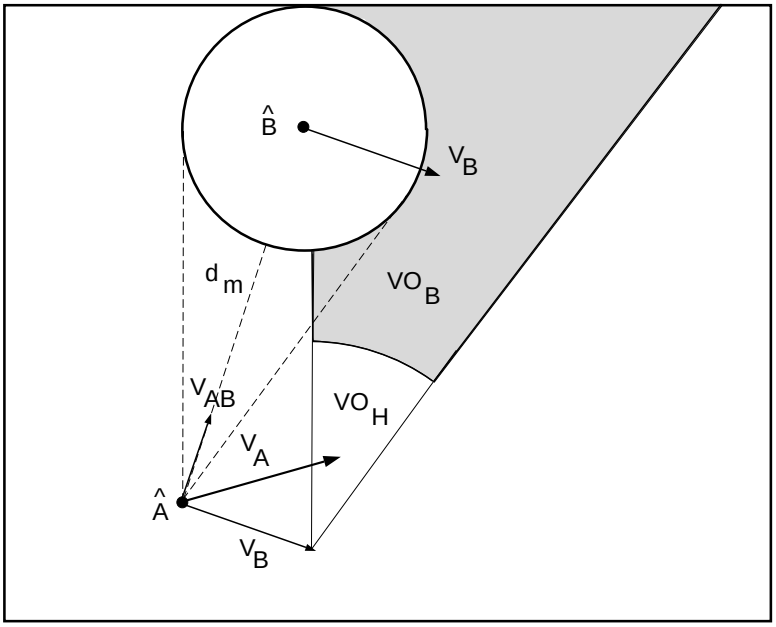

Figure 5: The velocity obstacle VO_B for a short time horizon.

3. The Avoidance Maneuver

In this Section, we describe the *avoidance maneuver*, consisting of a one-step change in velocity to avoid a future collision within a given time horizon. We start by discussing the set of reachable velocities that account for robot dynamics and actuator constraints.

3.1. The Reachable Avoidance Velocities

The velocities reachable by robot A at a given state over a given time interval Δt are computed by mapping the actuator constraints to acceleration constraints. The set

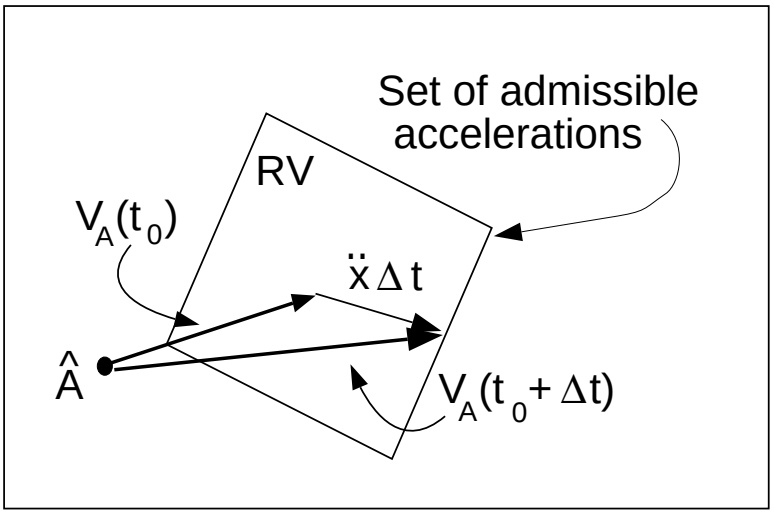

Figure 6: The Feasible Accelerations

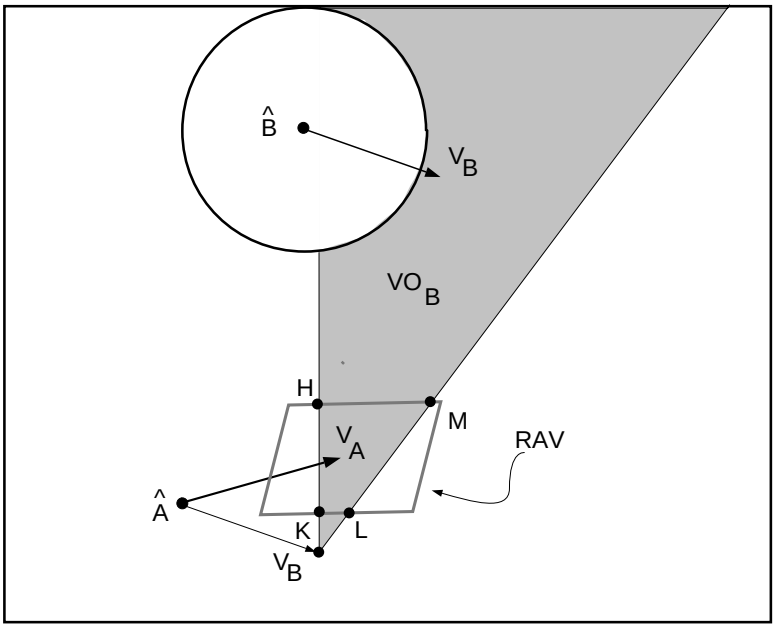

Figure 7: The reachable avoidance velocities RAV.

of *feasible accelerations* at time t, $\mathrm{FA}(t)$, is defined as:

$$\mathrm{FA}(t) = \{\ddot{\mathbf{x}} \,|\, \ddot{\mathbf{x}} = f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u}), \mathbf{u} \in U\} \tag{6}$$

where $\mathbf{x}$ is the position vector, $f(\mathbf{x}, \dot{\mathbf{x}}, \mathbf{u})$ represents the robot dynamics, $\mathbf{u}$ is the vector of the actuator efforts, and U is the set of admissible controls.

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

$$\mathrm{RV}(t + \Delta t) = \{\mathbf{v} \,|\, \mathbf{v} = \mathbf{v}_A(t) \oplus \Delta t \cdot \mathrm{FA}(t)\} \tag{7}$$

It is computed by scaling $\mathrm{FA}(t)$ by Δt and adding it to the current velocity of A, as shown schematically in Figure 6.

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

$$\mathrm{RAV}(t + \Delta t) = \mathrm{RV}(t + \Delta t) \ominus VO(t) \tag{8}$$

where $\ominus$ denotes the operation of set difference. A maneuver avoiding obstacle B can then be computed by selecting any velocity in RAV. Figure 7 shows schematically the set RAV consisting of two disjoint closed subsets. For multiple obstacles, the RAV may consist of multiple disjoint subsets.

It is important to note that the reachable set (7) accounts only for constraints on the robot accelerations, which we call *dynamic* constraints. In the context of mobile robots, non-holonomic kinematic constraints, which prevent the vehicle from moving sideways, need also be considered. However, the non-holonomic constraints are relevant only at low speeds, such as during parking, since at high speeds the dynamic constraints are generally more restrictive, as shown schematically in Figure 8. In other words, the bounded

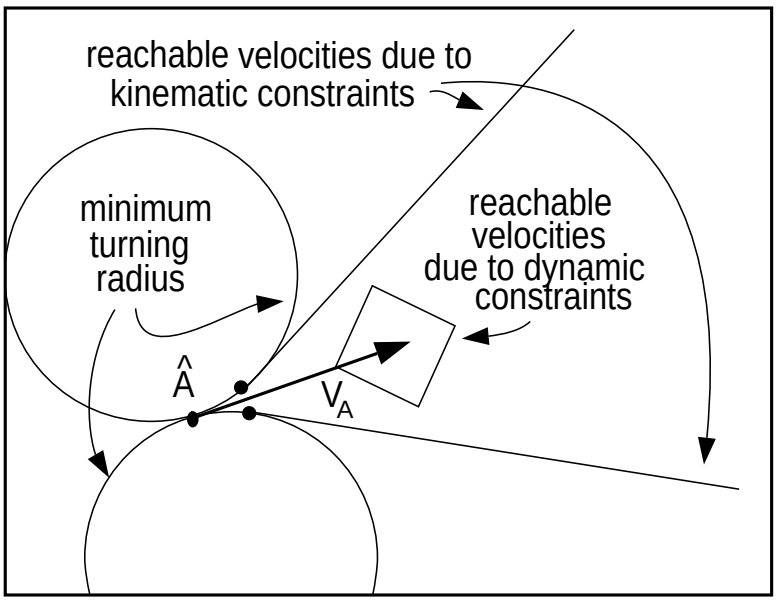

Figure 8: Dynamic versus kinematic constraints.

accelerations prevent the vehicle from moving sideways at non-zero forward speeds, thus automatically satisfying the non-holonomic constraints (Shiller and Sundar, 1996). For this reason, we will disregard the non-holonomic kinematic constraints in computing the feasible avoidance maneuvers.

3.2. Structure of the Avoidance Maneuvers

Avoidance maneuvers can be classified according to their position with respect to the moving obstacle. In general, the avoidance velocities form two disjoint sets separated by the set of colliding velocities, as was shown in Figure 7. The boundary between colliding and non-colliding velocities consists of the velocities laying on the two lines λ_f and λ_r, generating a maneuver tangent to $\hat{B}$ at some point $P \in \partial B$. To choose the structure of an avoidance maneuver, it is then necessary to determine on which side of the obstacle each maneuver will pass.

We identify the front and rear sides of $\hat{B}$ by attaching a coordinate frame (X,Y) to the center of $\hat{B}$, oriented with its X axis coinciding with the velocity $\mathbf{v}_B$. The Y axis then partitions the boundary of $\hat{B}$, ∂B, at points Y_f and Y_r into the *front* semi-circle, ∂B_f, which intersects the positive X axis, and the *rear* semi-circle, ∂B_r, as shown in Figure 9.

The following Lemma states that tangent maneuvers can be generated only by relative velocities laying on λ_f and λ_r, and that the actual tangency points of these maneuvers are different from the tangency points T_f and T_r of λ_f and λ_r, since they depend on the *absolute* velocities of A and B. Assuming no rotations, Figure 10 shows schematically the position of the tangency point P at time t_0, when the avoidance maneuver is computed, and at time t_1, when $\hat{A}$ reaches $\hat{B}$.

Lemma 1: Robot A is tangent to obstacle B at some point $P \in \partial B$ *iff* it follows a trajectory generated by $\mathbf{v}_A$ corresponding to $\mathbf{v}_{A,B} \in \{\lambda_f, \lambda_r\}$. The tangency sets in ∂B

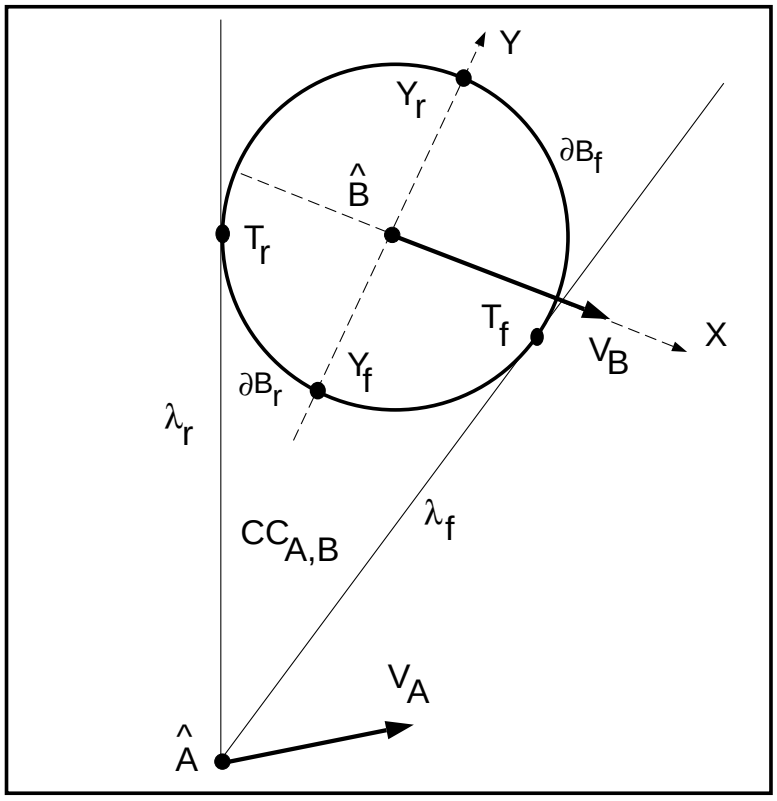

Figure 9: Grazing arcs in an avoidance maneuver.

consist of the shortest segment connecting $T_f = \lambda_f \cap \partial B$ to Y_f, and of the shortest segment connecting $T_r = \lambda_r \cap \partial B$ to Y_r. □

This Lemma is proved in Appendix A.

As discussed earlier, the boundary of the velocity obstacle VO, $\{\delta_f, \delta_r\}$, represents all

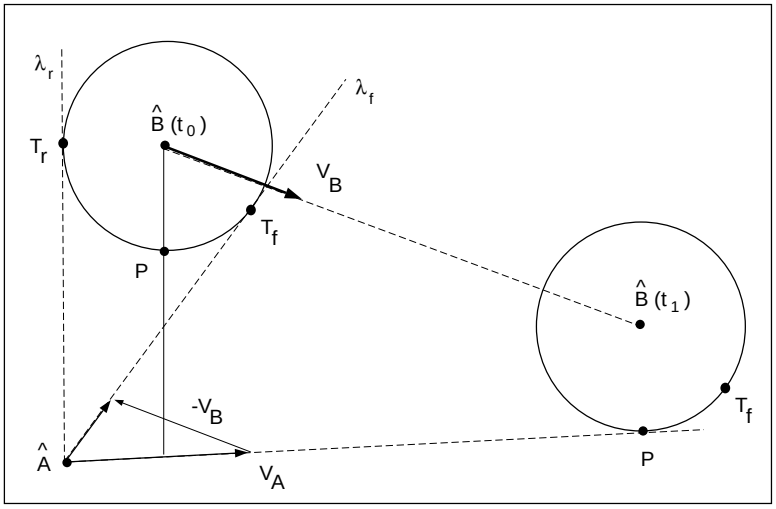

Figure 10: Trajectory tangent to B.

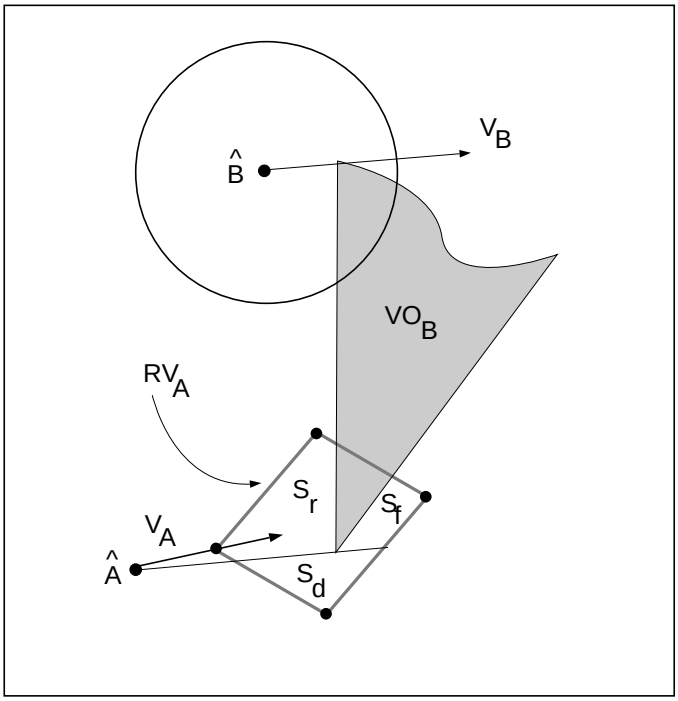

Figure 11: General structure of the reachable avoidance velocities.

absolute velocities generating trajectories tangent to $\widehat{B}$, since their corresponding relative velocities lay on λ_f and λ_r. For example, the only tangent velocities in Figure 7 are represented by the segments KH and LM of the reachable avoidance velocity set RAV.

Since the tangent velocities are uniquely determined by λ_f and λ_r, we use them to label the subsets in RAV containing velocities generating a single type of avoidance maneuver. The set RAV in Figure 11 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 $\widehat{A}$ and the apex of VO_B. Each of these subsets corresponds 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.

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 Appendix A.

The RAV set due to one obstacle may consist of *at most* three subsets. Clearly, there can be cases where RAV consists of fewer sets, one, two or none, as was shown in Figure 7. For the case of multiple obstacles, we represent each RAV set by S_s, where s is an ordered string of characters (f,r,d) representing the type of avoidance of each obstacle. For example, the set S_{ff} shown in Figure 12 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.

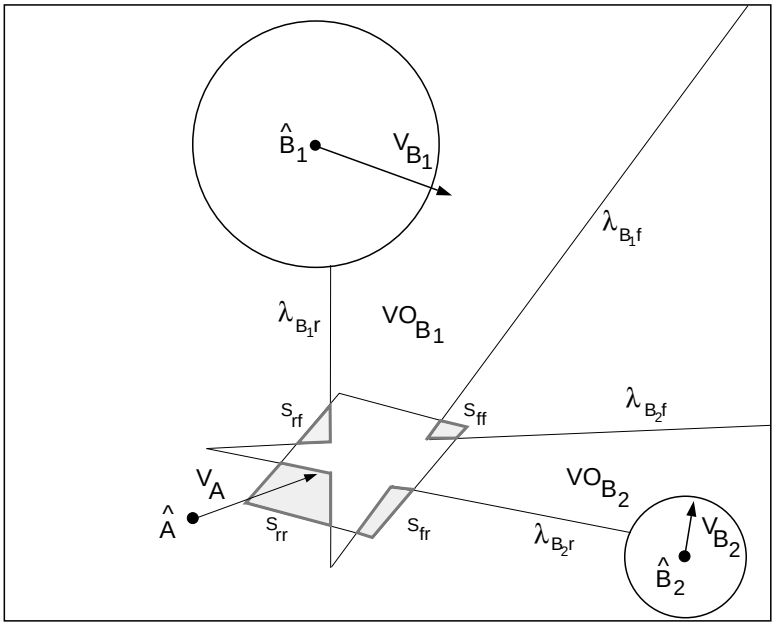

Figure 12: Classification of the reachable avoidance velocities.

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

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

The Theorem states that it is possible to subdivide the avoidance velocities RAV into subsets, each corresponding to a specific avoidance maneuver of an obstacle. For example, this knowledge about maneuver types can be used by an intelligent vehicle in avoiding a potentially dangerous obstacle: a car avoiding a big truck may choose a rear avoidance maneuver, even though a front maneuver might also be feasible, for the sake of safety. A procedure for computing the sets of avoidance velocities is described in Appendix B.

4. Computing the avoidance trajectories

This section presents a method for computing trajectories that avoid static and moving obstacles, reach the goal, and satisfy the robot's dynamic constraints. A trajectory consists of a sequence of avoidance maneuvers, selected by searching over a tree of feasible maneuvers computed at discrete time intervals. We propose a global search for off-line applications, and a heuristic search for on-line applications.

4.1. Global Search

The tree of avoidance maneuvers is generated by computing the reachable avoidance set RAV at discrete time intervals. The *nodes* on the tree correspond to the positions of the robot at times t_i, and the *branches* correspond to the avoidance maneuvers at those positions. The *operators* expanding each node into its successors at time $t_{i+1} = t_i + T$, i.e.

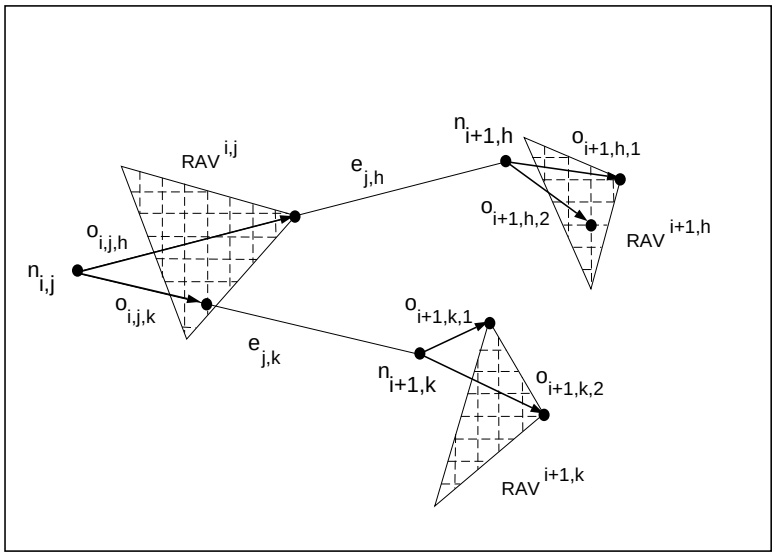

Figure 13: Tree representation for the global search.

generating the avoidance maneuvers, are the velocities computed by discretizing RAV. The search tree is formally defined as follows:

$$n_j(t_i) = \{\mathbf{x}_j, \text{RAV } (t_i))\} \quad (9)$$

$$o_{j,l}(t_i) = \{\mathbf{v}_l(t_i) \mid \mathbf{v}_l(t_i) \in \text{RAV}_j(t_i)\} \quad (10)$$

$$e_{j,k}(t_i) = \{(n_j(t_i), n_k(t_{i+1})) \mid n_k(t_{i+1}) = n_j(t_i) + (o_{j,l}\, T)\} \quad (11)$$

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

To keep the computation manageable, each avoidance set $\text{RAV}(t_i)$ is discretized by a grid. The positions reached by the robot at the end of each maneuver are the successors of node $n_j(t_i)$. A node $n_j(t_i)$ is completely expanded when all the operators $o_{j,l} \in \text{RAV}_j$ have been applied. The resulting tree has a constant time interval between nodes, and a variable branch number that depends on the shape of each RAV. By assigning an appropriate cost to each branch, this tree can be searched for the trajectory that maximizes some objective function, such as distance traveled, motion time, or energy, using standard search techniques (Nilsson, 1980; Pearl, 1985). Figure 13 shows schematically a subtree of a few avoidance maneuvers.

Since the avoidance maneuvers are generated using the velocities in RAV, each segment of the trajectory avoids all the obstacles that are reachable within the specified time horizon. This excludes trajectories that might, on some segment, be on a collision course with some of those obstacles, and that would avoid them on a later segment. Therefore, this method generates only conservative trajectories. Trajectories excluded from the computation can be generated by considering a shorter time horizon, or by computing the time-minimal trajectories (Fiorini, 1995).

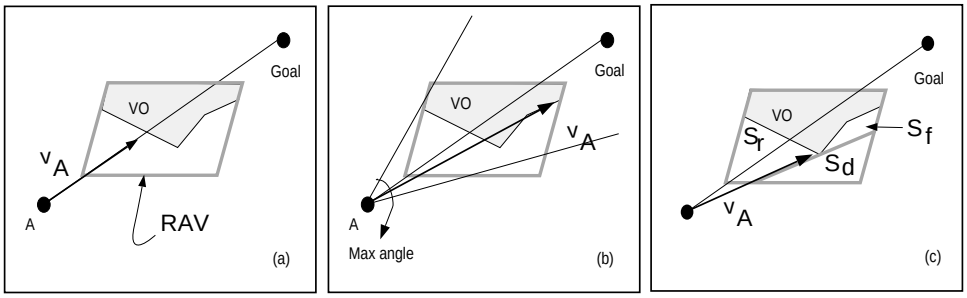

Figure 14: **a**: TG strategy. **b**: MV strategy. **c**: ST strategy.

4.2. Heuristic Search

For on-line applications, the trajectory can be generated incrementally by expanding only the node corresponding to the current robot position, and generating only one branch per node, using some heuristics to choose among all possible branches.

The heuristics can be designed to satisfy a prioritized series of goals, such as survival of the robot as the first goal, and reaching the desired target, minimizing some performance index, and selecting a desired trajectory structure, as the secondary goals. Choosing velocities in RAV (if they exist) automatically guarantees survival. Among those velocities, selecting the ones along the straight line to the goal would ensure reaching the goal. Selecting the highest feasible velocity in the general direction of the goal may reduce motion time. Selecting the velocity from the appropriate subset of RAV can ensure a desired trajectory structure (front or rear maneuvers). It is important to note that there is no guarantee that any objective is achievable at any time. The purpose of the heuristic search is to find a "good" local solution if one exist.

We have experimented with the following basic heuristics:

1. Choose the highest avoidance velocity along the line to the goal, as shown in Figure 14-a, so that the trajectory takes the robot towards its target. This strategy is denoted in the following by TG (to goal).

2. Select the maximum avoidance velocity within some specified angle α from the line to the goal, as shown in Figure 14-b, so that the robot moves at high speed, even if it does not aim directly at the goal. This strategy is henceforth called MV (maximum velocity).

3. Select the velocity that avoids the obstacles according to their perceived risk. This strategy is called ST (structure). In Figure 14-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. It might be advantageous to switch between heuristics in order to yield trajectories that better satisfy the prioritized goals. Examples of trajectories generated by these heuristics are presented in the following Section.

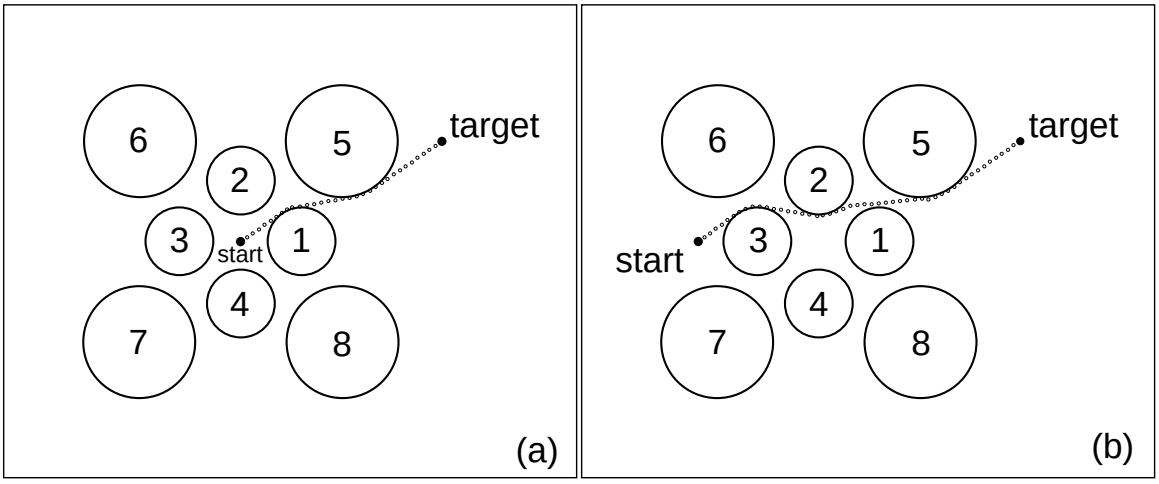

Figure 15: Trajectories avoiding static obstacles.

5. Examples

This Section presents several examples of navigation in static and dynamic environments using the Velocity Obstacle approach.

5.1. Avoidance of static obstacles

The first two examples demonstrate the applicability of this approach to static environments. Here, a point robot avoids eight circular obstacles, arranged as shown in Figure 15, starting from rest from two different points. The trajectories shown in Figure 15 were computed every $1\,s$ using the MV heuristics, and time horizon $T_h = 10\,s$. The trajectory in Figure 15-a first follows the boundary of the velocity obstacle of obstacle 1, then switches to obstacle 5. Similarly, the trajectory in Figure 15-b switches from obstacle 3, to 2, and finally to 5. In this case, obstacle 1 was not obstructing the target after avoiding obstacle 2. In both cases, setting the time horizon higher, to $T_h = 20\,s$, resulted in no solution since the reachable avoidance velocities RAV at the the initial positions were completely covered by the VO of the static obstacles.

5.2. Avoidance of fixed and moving obstacles

This example demonstrates avoidance of static and moving obstacles that move along circular trajectories. The robot starts from rest at the center, as shown in Figure 16-a, first avoiding the small static obstacles, and then avoiding the large moving obstacles, using the MV heuristics. The trajectory is shown in three snapshots in Figures 16-(b,c,d), corresponding to times $t = 24\,s$, $t = 29\,s$, and the final time $t = 37.6\,s$, respectively. After avoiding obstacle 1, the robot moves to avoid the incoming obstacle 8. It aims at being tangent to obstacle 8, which explains the initial distance from the obstacle in Figures 16-(b). The robot is eventually tangent to obstacle 8, as shown in Figure 16-c. From that point on, the robot continues towards the target, as shown in Figure 16-d.

Here the robot slows down while approaching obstacle 8, as represented by the short distance between the circles in Figure 16-b. This occurred due to the linear approximation of the obstacle's trajectory, which at those points made rear avoidance the only feasible

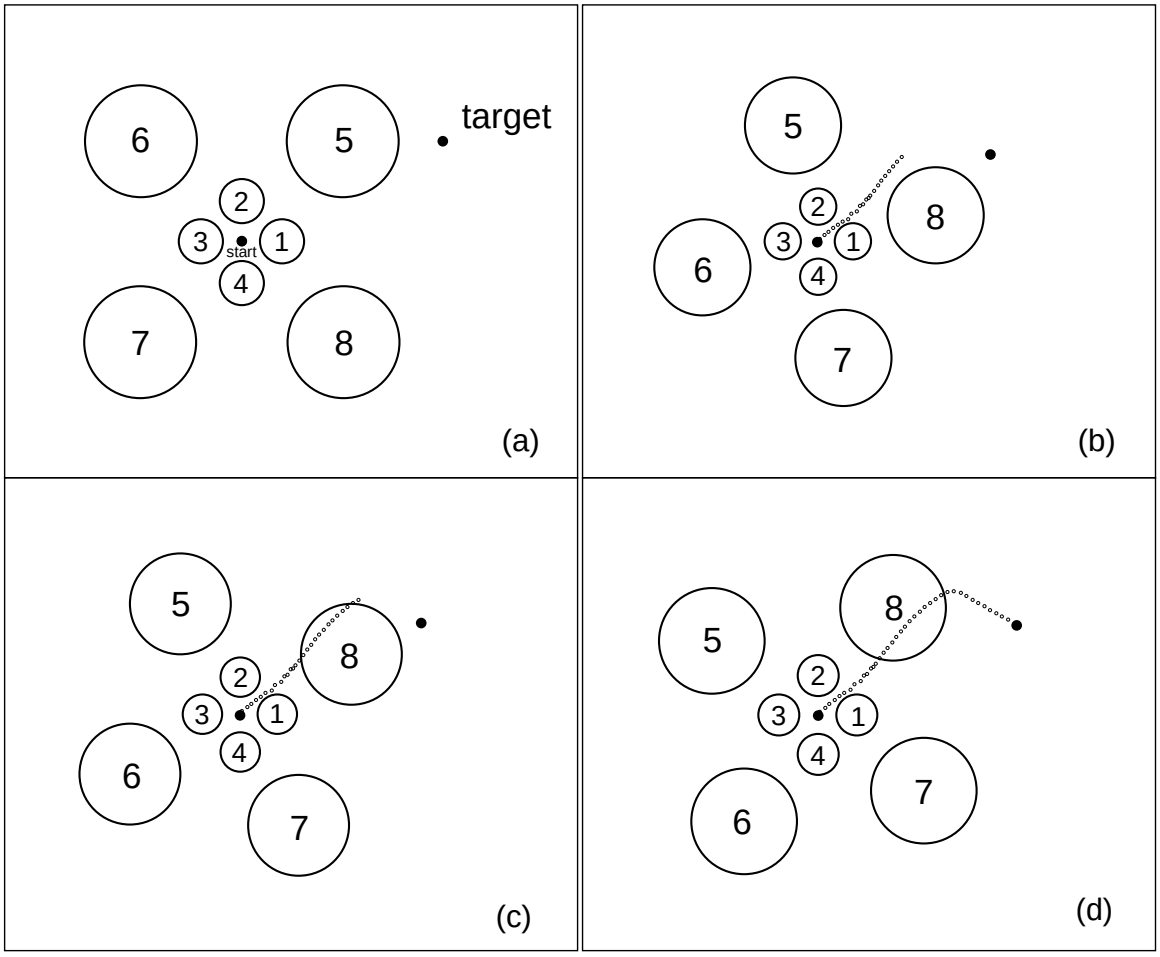

Figure 16: Snapshots of the trajectory avoiding static and moving obstacles.

maneuver. Later, due to the change in direction of the obstacle, the front avoidance maneuver became feasible, which explains the higher speeds at the distal end of the trajectory shown in Figure 16-b.

5.3. Intelligent Highway Examples

The following examples demonstrate the proposed approach for intelligent vehicles negotiating highway traffic.

5.3.1. Moving to the Exit Ramp

This example consists of a robotic vehicle in the leftmost lane, attempting to reach the exit ramp on the right, while avoiding other two vehicles that move at constant speeds. 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 initial velocities are represented by vectors attached to the center of each vehicle. The grey circles represent positions of the robot and of the obstacles when they cross each other's path.

First, we computed the time-optimal trajectory using a global search, as discussed in Section 4.1. The nodes of the tree were generated at $1s$ intervals. The RAV sets were

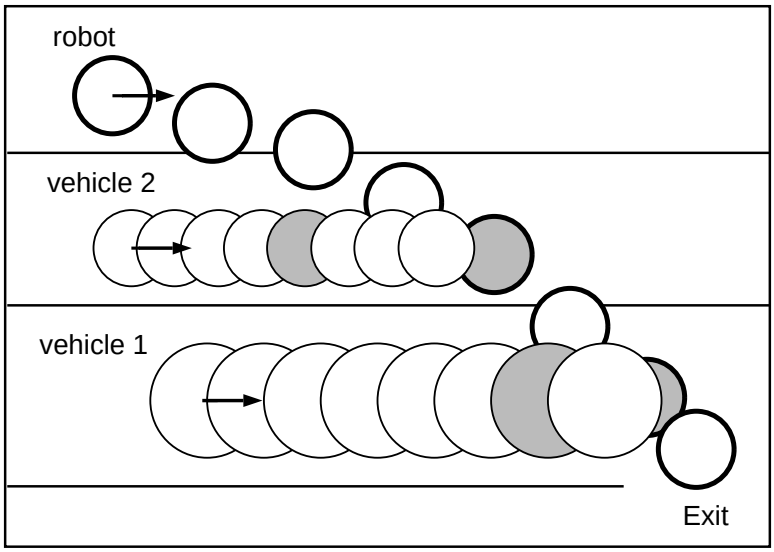

Figure 17: Trajectory computed by global search.

discretized on average by 12 points, and the tree was expanded to a depth of 5 levels .

Using the Depth First Iterative Deepening algorithm (Korf, 1985) resulted in the trajectory shown in Figure 17, with the motion time of 3.5 s.

Using the TG heuristic resulted in the trajectory shown in Figure 18. Along this trajectory, the robot slows down and lets vehicle 2 pass, and then speeds up towards the exit, behind vehicle 1. The total motion time for this trajectory is 6.07 s.

Using the MV heuristics produced the trajectory shown in Figure 19. Here, 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. It compares favorably with the time of 3.5 s computed by the global search.

Using both MV and TG strategies resulted in the trajectory shown in Figure 20. Here, the robot first speeds up to pass vehicle 2, then slows down to let vehicle 1 pass, and then speeds up again towards the goal. The motion time for this trajectory is 5.31 s.

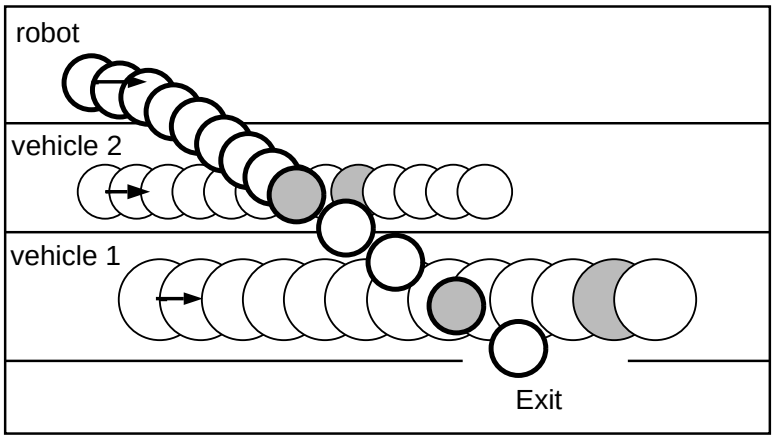

Figure 18: Trajectory computed with the TG strategy.

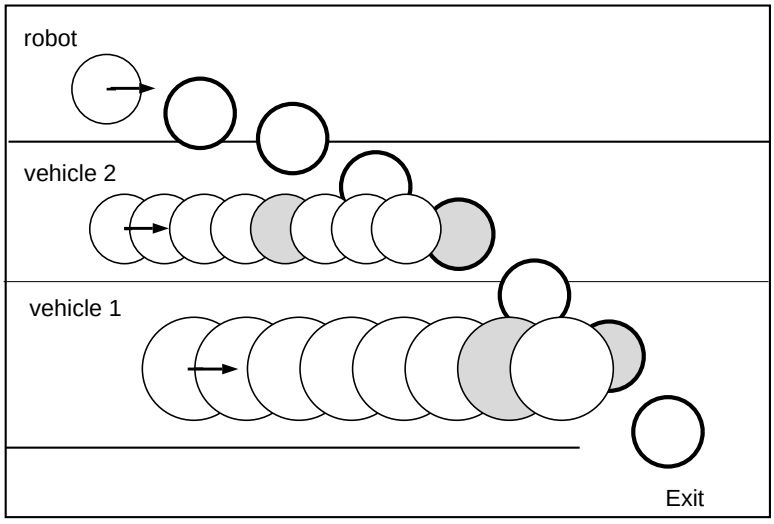

Figure 19: Trajectory computed with MV strategy.

This trajectory was computed using the MV heuristics for $0 \leq t \leq 2.0\ s$ and the TG heuristics afterwards.

5.3.2. Negotiating Highway Traffic

This example considers a "speeding" robot. First, the vehicles are represented by small circles, simulating motorcycles. Using the MV heuristic resulted in the trajectory shown in Figure 21. Here, the robot passes vehicle 1 without making a complete lane change, and thus is not affected by the presence of vehicles 2 and 5.

Considering larger vehicles, and keeping the initial velocities the same as before, resulted in the trajectory shown in Figure 22. Blocked by vehicle 2 from the left and vehicle 3 from the right, the robot first slows down to attempt a lane change to the left in order to pass vehicle 1. But then a higher velocity to the right becomes feasible, and the robot speeds up in front of vehicle 3, passes vehicle 1 on the right, and returns to the center lane.

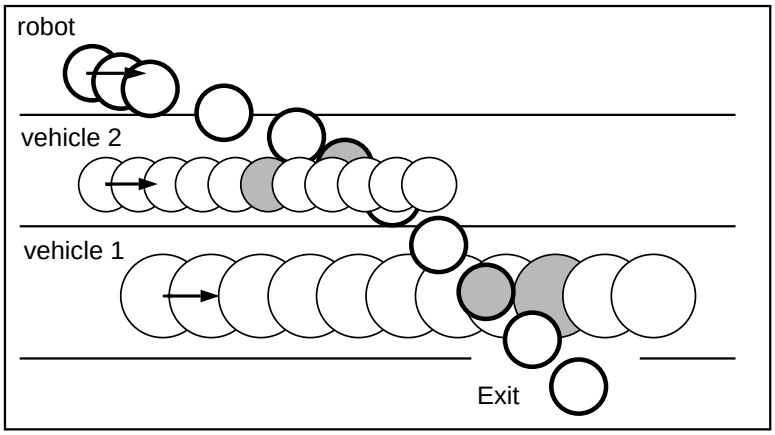

Figure 20: Trajectory computed with both TG and MV strategies.

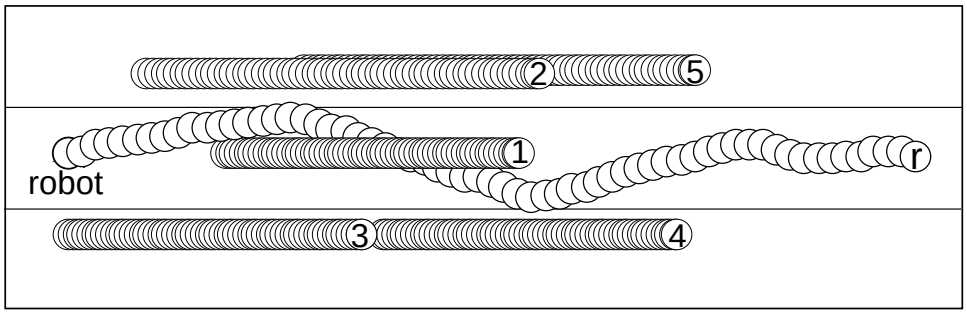

Figure 21: Trajectory computed for a motorcycle.

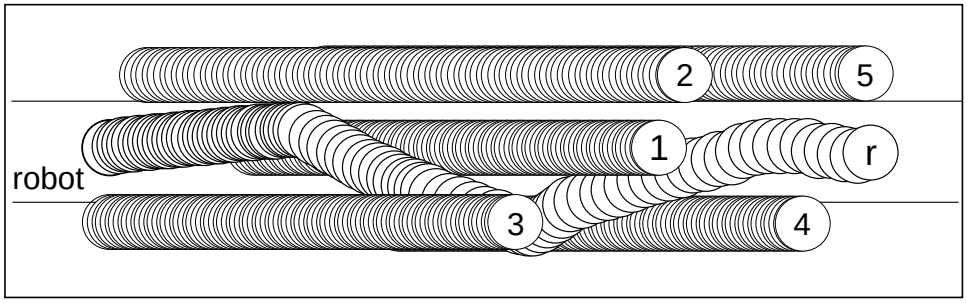

Figure 22: Trajectory computed for a car.

6. Conclusions

A method for planning the motion of a robot in dynamic environments has been presented. It is a first-order method since it relies on velocity information for planning an avoidance maneuver. Planning in the velocity space makes it possible to consider robot dynamics and actuator constraints. In contrast, most current planners are zero-order, generating avoidance maneuvers based on position information, and most do not consider robot dynamics.

Key features of this approach are: *i*) it provides a simple geometric representation of potential avoidance maneuvers; *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; *iv*) it allows for the simple consideration of robot dynamics and actuator constraints.

The avoidance maneuvers are generated using a representation of the obstacles in the velocity space, that we call the Velocity Obstacle. It represents the colliding velocities of the robot with a given obstascle, that moves at a given velocity. General trajectories are approximated by a sequence of piecewise constant segments. Robot dynamics are considered by restricting the set of potential avoidance velocities to those reachable by the set of admissible accelerations over a given time interval.

This approach was demonstrated numerically in several examples showing the applicability of the proposed approach to the avoidance of static and moving obstacles, and to obstacles moving along straight and curved trajectories. Also demonstrated was the potential use of this method to effectively negotiate highway traffic.

7. Acknowledgment

The authors are grateful to the anonymous reviewers for their suggestions and extensive reviews. The research described in this paper has been partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

References

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

Erdmann, M. and Lozano-Perez, T. (1987). On multiple moving objects. *Algorithmica*, 2(4):477–521.

Featherstone, R. (1990). Swept bubbles: A method of representing swept volume and space occupancy. Technical Report TR-90-024 and MS-90-069, Philips Laboratories, North American Philips Corporation, Briarcliff Manor, New York 10510.

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

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

Fiorini, P. and Shiller, Z. (1995). Robot motion planning in dynamic environments. In Girald, G. and Hirzinger, G., editors, *International Symposium of Robotic Research*, pages 237–248, Munich, Germany. Springer-Verlag.

Fiorini, P. and Shiller, Z. (1996). Time optimal trajectory planning in dynamic environments. In *IEEE International Conference of Automation and Robotics*, volume 2, pages 1553–1558, Minneapolis, MN.

Fiorini, P. and Shiller, Z. (1997). Time optimal trajectory planning in dynamic environments. *Journal of Applied Mathematics and Computer Science*, 7(2):101–126.

Fraichard, T. (1993). Dynamic trajectory planning with dynamic constraints: a state-time space approach. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1393–1400, Yokohama, Japan.

Fraichard, T. and Laugier, C. (1993). Path-velocity decomposition revisited and applied to dynamic trajectory planning. In *IEEE International Conference of Automation and Robotics*, volume 1, pages 40–45, Atlanta, GA.

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

Fujimura, K. (1995). Time-minimum routes in time-dependent networks. *IEEE Transaction on Robotics and Automation*, 11(3):343–351.

Fujimura, K. and Samet, H. (1989a). A hierarchical strategy for path planning among moving obstacles. *IEEE Transaction on Robotics and Automation*, 5(1):61–69.

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

Fujimura, K. and Samet, H. (1990). Motion planning in a dynamic domain. In *IEEE International Conference on Robotics and Automation*, pages 324–330, Cincinnati, OH.

Fujimura, K. and Samet, H. (1993). Planning a time-minimal motion among moving obstacles. *Algorithmica*, 10:41–63.

Hayward, V., Aubry, S., Foisy, A., and Ghallab, Y. (1995). Efficient collision prediction among many moving obstacles. *International Journal of Robotics Research*, 14(2):129–143.

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

Korf, R. (1985). Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, (27):97–109.

Lee, B. and Lee, C. (1987). Collision-free motion planning of two robots. *IEEE Transactions on System Man and Cybernetics*, SMC-17(1):21–32.

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

O'Rourke, J. and Badler, N. (1979). Decomposition of three-dimensional objects into spheres. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-1(3):295–305.

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

Reif, J. and Sharir, M. (1985). Motion planning in the presence of moving obstacles. In *25th IEEE Symposium on the Foundation of Computer Science*, pages 144–153.

Reif, J. and Sharir, M. (1994). Motion planning in the presence of moving obstacles. *Journal of ACM*, 41(4):764–790.

Sanborn, J. and Hendler, J. (1988). A model of reaction for planning in dynamic environments. *International Journal of Artificial Intelligence in Engineering*, 3(2):95–101.

Shiller, Z. and Sundar, S. (1996). Emergency maneuvers of autonomous vehicles. In *The 13th World Congress, IFAC96*, volume Q, pages 393–398, San Fancisco, CA.

Tsubouchi, T. and Arimoto, S. (1994). Behavior of a mobile robot navigated by an iterated forecast and planning scheme in the presence of multiple moving obstacles. In *IEEE Conference on Robotics and Automation*, pages 2470–2475, San Diego, CA.

A. Proofs of Lemmas and Theorem

Proof of Lemma 1:
The proof is carried out for $\widehat{B}$ since there is a one-to-one correspondence between $\widehat{B}$ and B. Furthermore, only the front side of $\widehat{B}$, ∂B_f, is analyzed, since the rear side, ∂B_r, is completely analogous. First, the necessary and 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 ∂B_f.

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

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

Then $\mathbf{v}_{A,B}$ would be either inside or outside $CC_{A,B}$. In the first case, by definition of $CC_{A,B}$, 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 $\widehat{B}$.

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

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

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

We prove the second part of the Lemma by construction, computing the tangency points that correspond to each tangent velocity $\mathbf{v}_{A,B} \in \lambda_f$. The tangent relative velocities are represented by the semi-infinite line $\{\widehat{A}, \lambda_f\}$, with minimum and maximum relative velocities given by $|\mathbf{v}_{A,B}| \to 0$, and $|\mathbf{v}_{A,B}| \to \infty$, respectively. The former velocity corresponds to the limit point Y_f, when A and B move on parallel trajectories. The latter velocity corresponds to the limit point T_f.

The correspondence between any other velocity $0 < \mathbf{v}_{A,B} < \infty$, and a tangency point $P \in (T_f, Y_f)$ is established by using the orthogonality between a tangent trajectory t, due to $\mathbf{v}_A = (\mathbf{v}_{A,B} - \mathbf{v}_B)$, and the line l through the center of $\widehat{B}$ and the tangency point. The right angle between t and l can be embedded into the circle $\mathcal{C}$ of diameter $(\widehat{A},\widehat{B})$, as shown in Figure 23. Thus, given a desired tangency point $P \in \partial B_f$, circle C maps it into the direction of the corresponding tangency trajectory t, represented by the line through $\widehat{A}$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 ∂B_f with the line through the center of $\widehat{B}$ and Q.

The magnitude of the velocity $\mathbf{v}_A$ tangent to $\widehat{B}$ 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 $\mathbf{v}_{A,B} = \mathbf{v}_A - \mathbf{v}_B$.

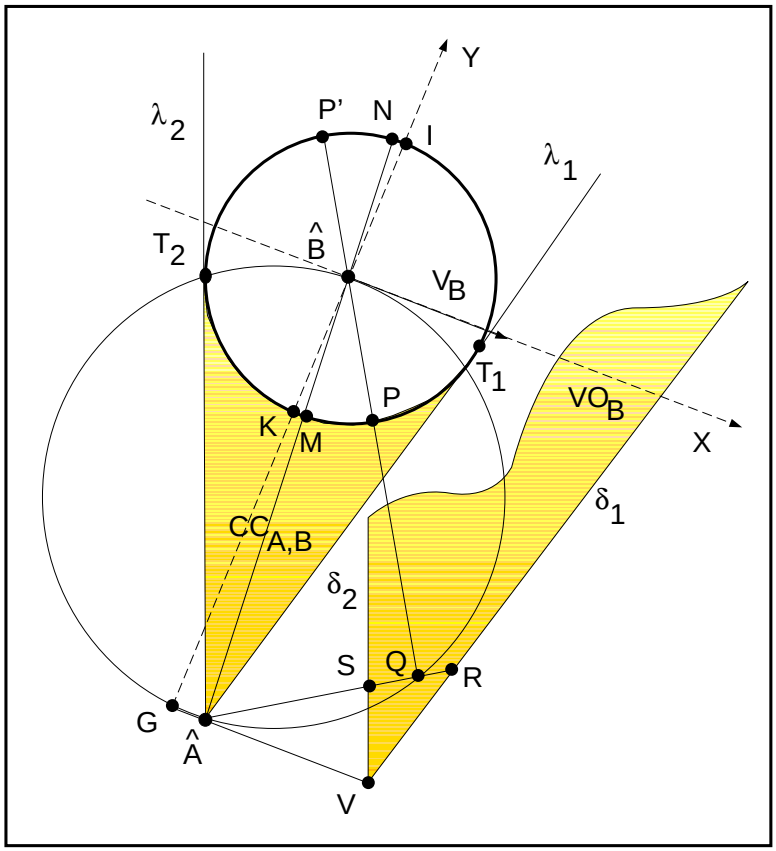

Figure 23: Contact points of tangent trajectories.

Note that the magnitude of the velocity $\mathbf{v}_A$ tangent to $\widehat{B}$ in $P' \in \partial B_r$ is simply given by the intersection S of t 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, $\partial(VO_B)$. Subset S_d, instead, does not include in its boundary any segment from the boundary of VO_B. From Lemma 1, velocities with tip on these segments generate trajectories grazing B, 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 $\partial(VO_B)$ 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 in a finite time, since the boundary of S_d is the velocity parallel to $\mathbf{v}_B$. Therefore, if this subset exists in RAV, it contains all the velocities diverging from B.

■

Proof of Theorem 1:
From Lemma 2, each velocity obstacle VO_{B_j} divides the set of reachable avoidance velocities RAV into three non-overlapping subsets. Then, m obstacles, B_j $(j = 1, \ldots, m)$,

generate $3m$ subsets, that are partially overlapping. Let us assume that RAV is divided into l non-overlapping subsets RAV_i $i = 1, \ldots, l$.

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.

Some obstacles do not contribute directly to the boundary of a RAV_i because a successive VO_{B_j} has covered the corresponding segment of the boudary of the RAV_i. In this case the correct sequence of avoidances is generated incrementally during the computation of the RAV_i.

■

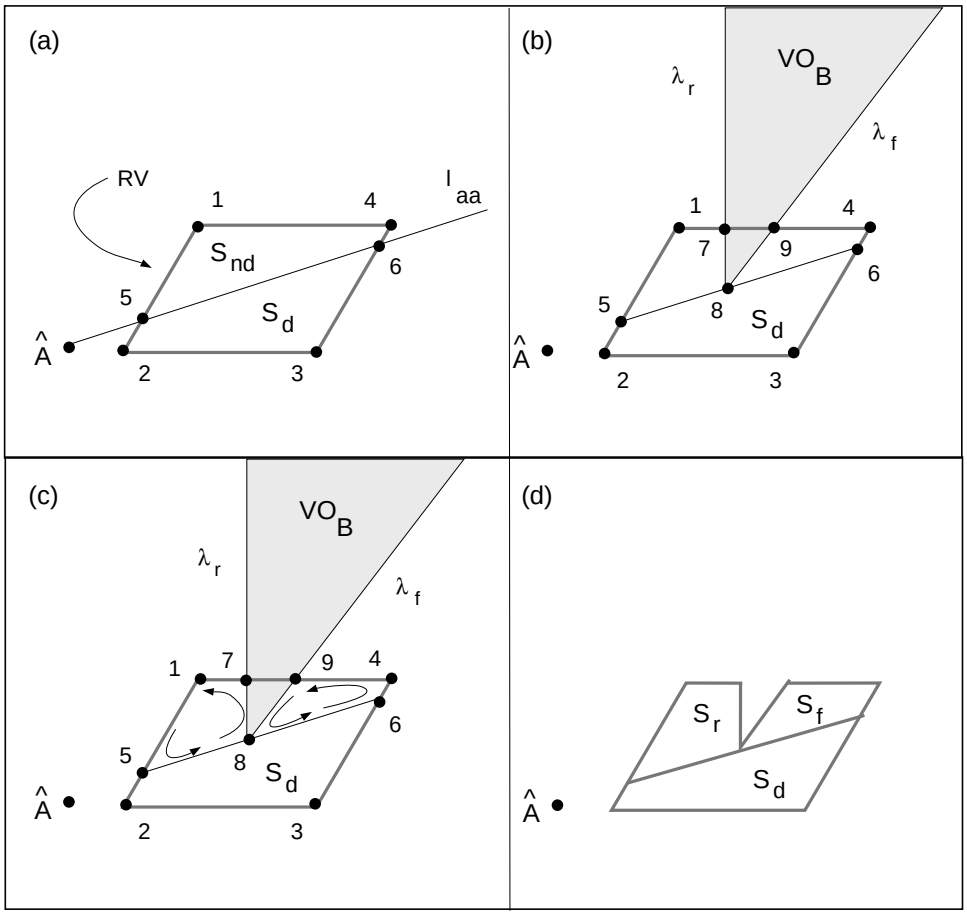

Figure 24: The computation of the Reachable Avoidance Velocities RAV.

B. Generating the Set of Avoidance Velocities

The procedure used for generating the sets of Reachable Avoidance Velocity RAV, S_i (i=d,f,r), for robot A is illustrated in Figure 24. The Figure shows the set of reachable velocities RV, the velocity obstacle VO_B for a long time horizon T_h, and the line l_{aa} passing through $\widehat{A}$ and the apex of VO_B. The S_i sets are represented by the ordered lists of their vertices $V_i = \{(x_i, y_i), (l_{i-1}, l_i)\}$, each specified by its position and by its supporting lines. The lists are ordered counter-clockwise so that the interior of the set is on the left of its boundary.

The procedure for computing S_i consists of the following steps:

1. Compute the sets of diverging velocities and of non-diverging velocities by intersecting RV with the line l_{aa}. Figure 24-a shows the two sets represented by vertices $(5, 2, 3, 6)$ and$(1, 5, 6, 4)$ respectively. Label the set of diverging velocities S_d and the set of non-diverging velocities S_{nd}.

2. Compute the intersections between the boundary of S_{nd} and of the velocity obstacle VO_B, and insert them into the list representing S_{nd}. Figure 24-b shows the intersections as points $(7, 8, 9)$. The new list representing S_{nd} becomes $(1, 5, 8, 6, 4, 9, 7)$.

3. Compute the sets of non-colliding velocities, i.e. $(S_{nd} \ominus VO_B)$ by marching counter-clockwise along the boundary of S_{nd} and clockwise along the boundary of VO_B, starting from the first vertex that is external to VO_B. This corresponds to assigning a positive value to S_{nd} and a negative value to VO_B. Figure 24-c shows schematically the march along the boundaries of S_f, starting from vertex 1 and generating the difference sets represented by $(9, 8, 6, 4)$ and by $(1, 5, 8, 7)$.

4. Label the non-colliding velocity sets according to the segments of VO_B included in their boundary. The set including a segment of the rear boundary of VO_B λ_r, is the set of rear avoidance velocities S_r, and the set including a segment of the front boundary λ_f is the set of front avoidance velocities S_f, as shown in Figure 24-d.

To consider multiple obstacles, this procedure is applied recursively to each subset S_i, (i=d,f,r), of RAV, generating smaller subsets S_s, where the index s is a string representing the type and sequence of avoidance maneuvers of each obstacle. For example, three obstacles might generate up to 9 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_s is $3m$, where m is the number of obstacles in the environment. However, the actual number is much smaller, because the lines l_{aa} from A to the apex of each VO_{B_j} do not intersect each other. Also, many of the potential subsets S_s 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_s remaining to be computed. To ensure that the set RAV is not empty, sometimes it is necessary to shorten the time horizon, thus reducing the size of some of the VO_{B_j}.