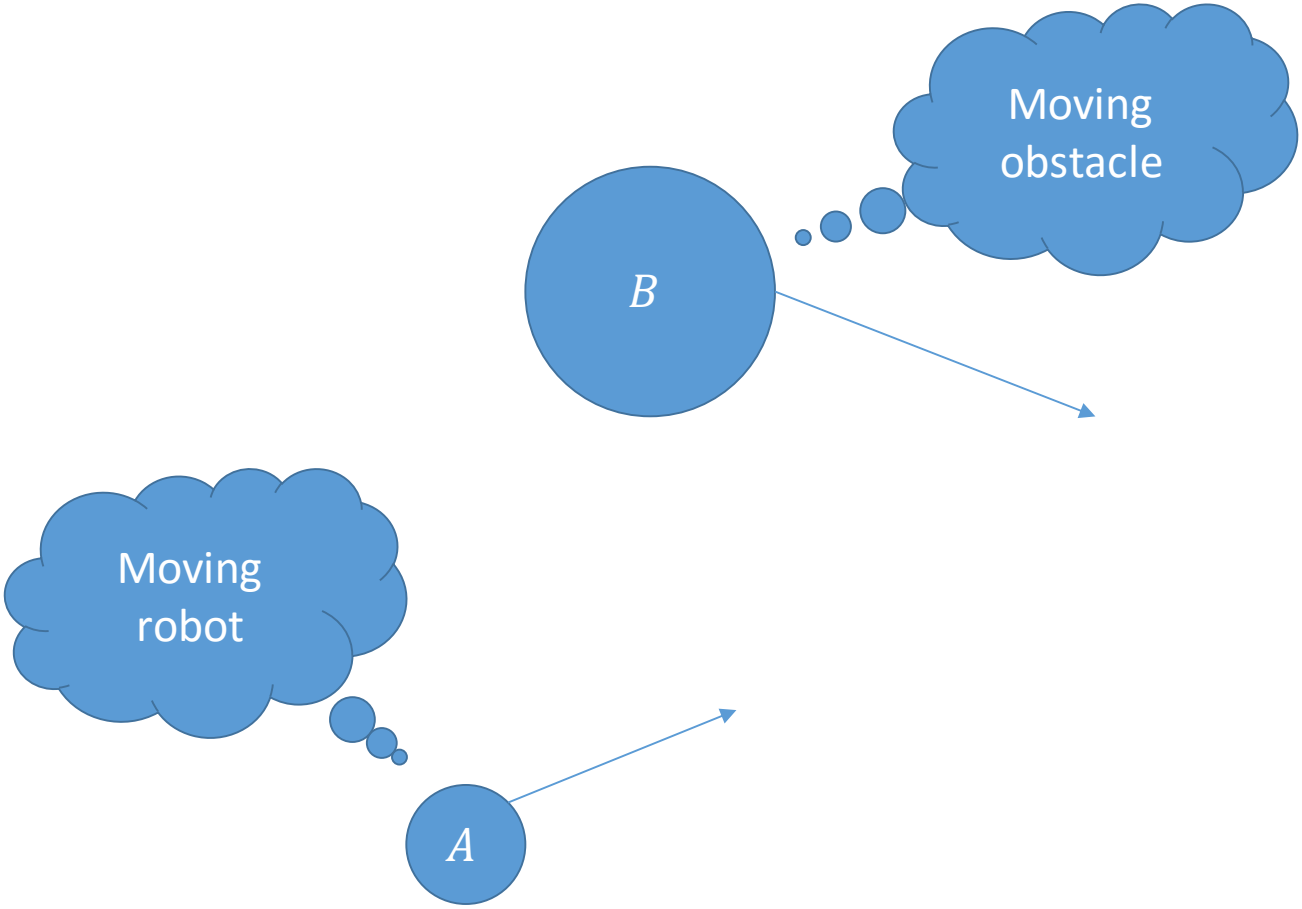
Autonomous Navigation on 2D Environments

based on
Motion Planning on Dynamic Environments
Using Velocity Obstacles (Fiorini and Shiller)

Paul Vincent Contreras

Problem Environment



A is the robot and B is the moving object that needs to be avoided.

Problem Environment

Given a robot and an obstacle each moving with a velocity v_A and v_B respectively, how do we prevent them from colliding with each other?

Step 1: Obtain the collision cone

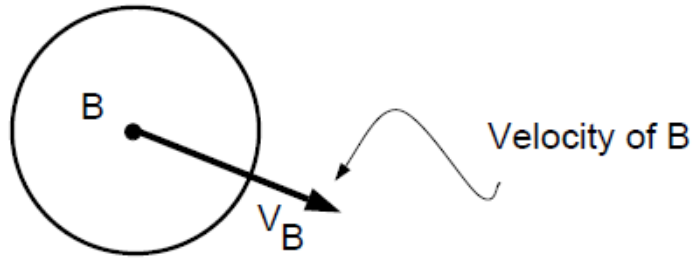
Collision Cone (Definition)

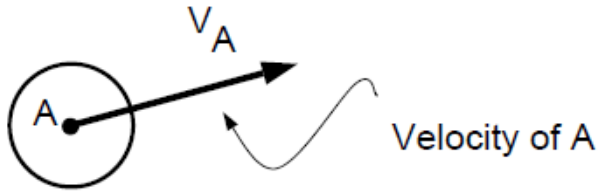
The collision cone $CC_{A,B}$ is the set of colliding relative velocities between $\hat{A}$ and $\hat{B}$:

$$CC_{A,B} = \left\{ \boldsymbol{v_{A,B}} \middle| \lambda_{A,B} \cap \hat{B} \neq \emptyset \right\}$$

where $v_{A,B}$ is the relative velocity of $\hat{A}$ with respect to $\hat{B}$, $\boldsymbol{v_{A,B}} = \boldsymbol{v_A} - \boldsymbol{v_B}$ and $\lambda_{A,B}$ is the line of $\boldsymbol{v}_{A,B}$. The cone's apex (summit) is $\hat{A}$.

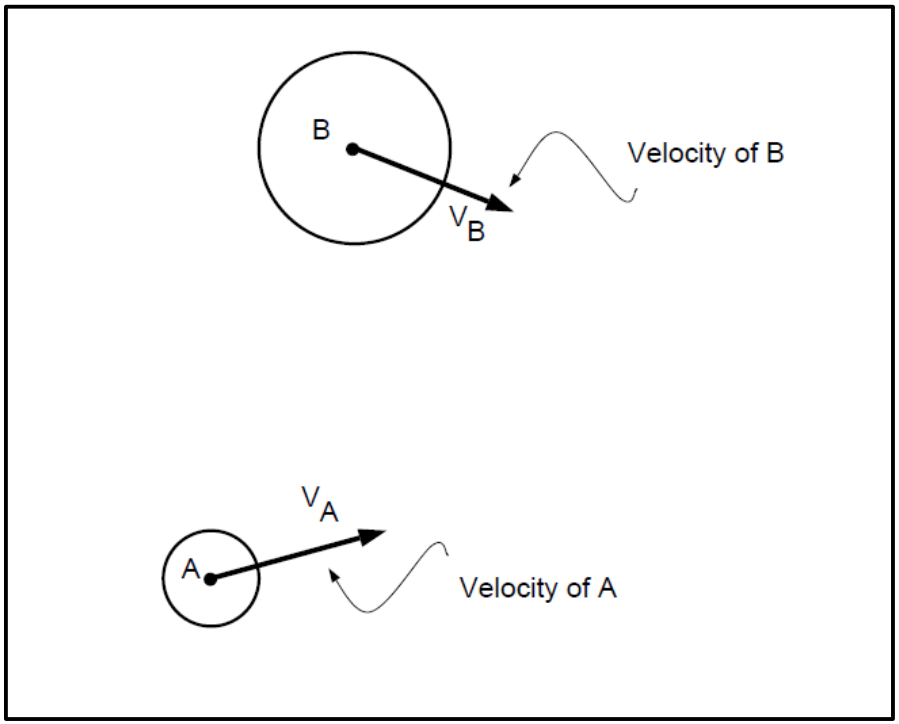
Step 1: Obtain the collision cone

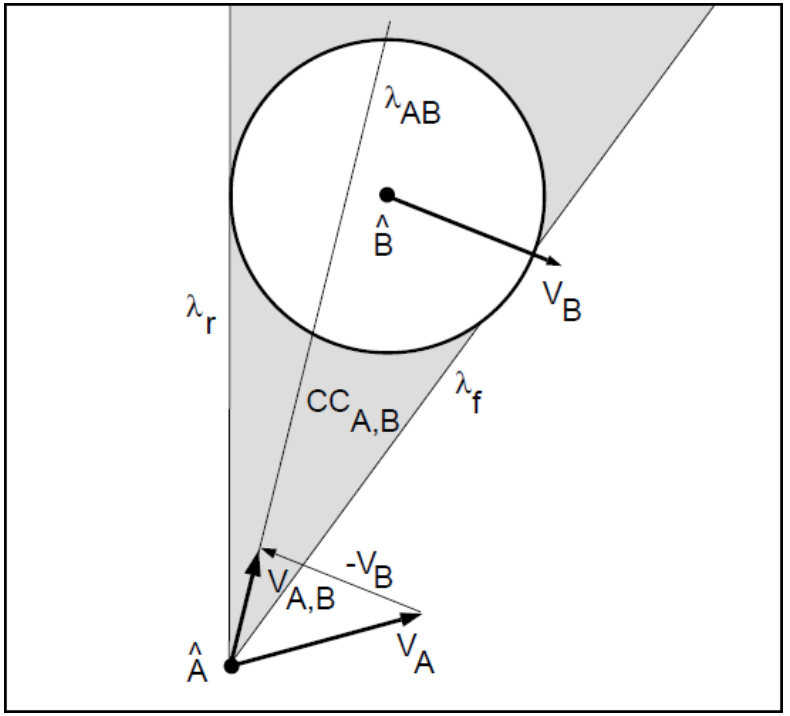




Step 1: Obtain the collision cone

- First map B to the *Configuration Space* of A, by reducing A to the point $\hat{A}$ and enlarging B by the radius of A to $\hat{B}$.





Collision cone

Notes

- Any relative velocity that lies between the two tangents to $\hat{B}, \lambda_f$ and λ_r will cause a collision between A and B.
- Any relative velocity that lies outside $CC_{A,B}$ is guaranteed to be collision-free provided that $\hat{B}$ maintains its current shape and speed.
- Collision cones are specific to a particular pair of robot/obstacle.
- If we want to consider the absolute velocities of A, we add the velocity $\boldsymbol{v}_B$ to each of the velocity in $CC_{A,B}$. The result is what we call the **velocity obstacle**.

Step 2: Add the absolute velocity of the obstacle

Vehicle obstacle (Definition)

The **velocity obstacle** is the set of all velocities of a robot that will result in a collision with another robot at some moment in time, assuming that the other robot maintains its current velocity.

Formally, it is defined as:

$$VO = CC_{A,B} \oplus \boldsymbol{v_B}$$

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

Minkowsky vector sum

- The **Minkowski sum** (also known as dilation) of two sets of position vectors *A* and *B* in Euclidean space is formed by adding each vector in *A* to each vector in *B*, i.e., the set

$$A + B = \{a + b | a \in A, b \in B\}$$

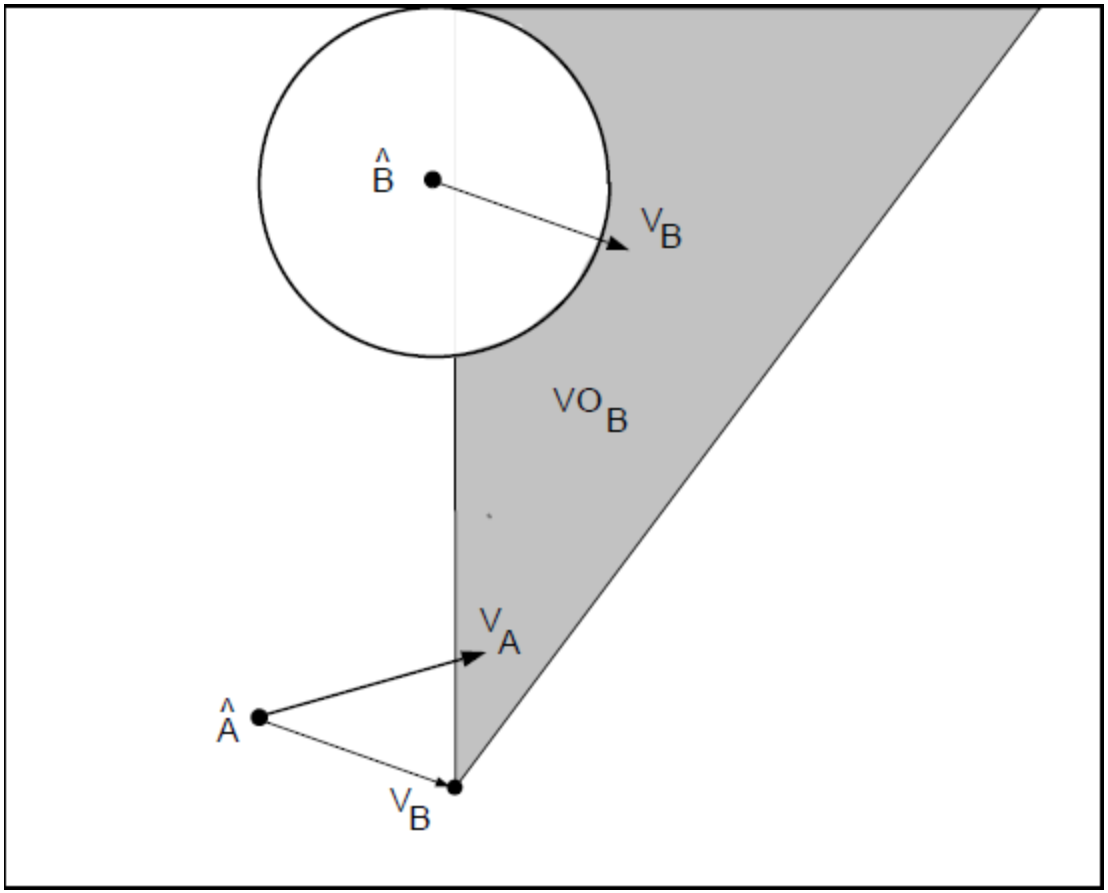
Example:

$A = \{(1,0), (0,1), (0,-1)\}$ and $B = \{(0,0), (1,1), (1,-1)\}$

The Minkowski sum is:

$A + B = \{(1,0), (2,1), (2,-1), (0,1), (1,2), (1,0), (0,-1), (1,0), (1,-2)\}$

Velocity obstacle



How do we avoid collisions?

- Generate an avoidance trajectory
  - Feasible accelerations
  - Reachable Velocities
  - Reachable Avoidance Velocities

What are the feasible movements of the robot?

The set of **feasible accelerations**, $FA(t)$, that a robot can take at time t is defined as:

$$FA(t) = \{\ddot{\boldsymbol{x}} | \ddot{\boldsymbol{x}} = f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u}), \boldsymbol{u} \in U\}$$

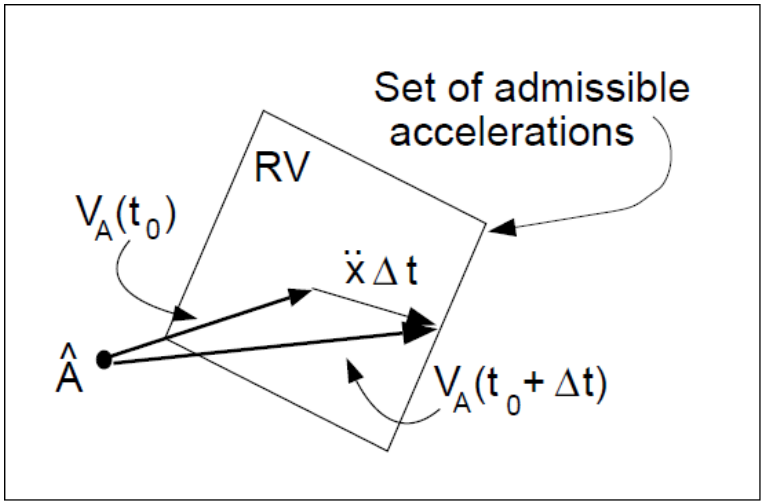
where $\boldsymbol{x}$ is the position vector, $f(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{u})$ represents the robot dynamics, $\boldsymbol{u}$ is the vector of actuator efforts and $\boldsymbol{U}$ is the set of admissible controls.

Reachable Velocities

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

$$RV(t + \Delta t) = \{\boldsymbol{v} | \boldsymbol{v} = \boldsymbol{v_A}(t) \oplus \Delta t \cdot FA(t)\}$$

RV is computed by scaling $FA(t)$ by Δt and adding it to the current velocity of A.

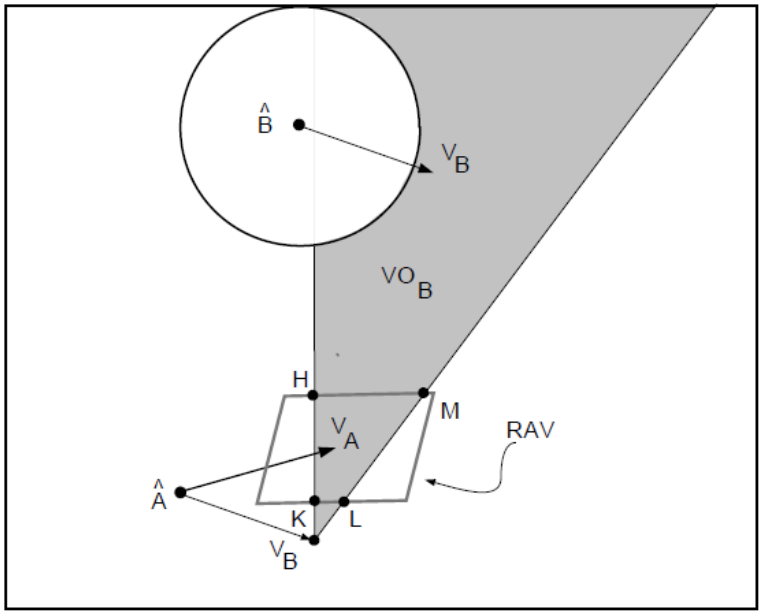


Reachable Avoidance Velocities

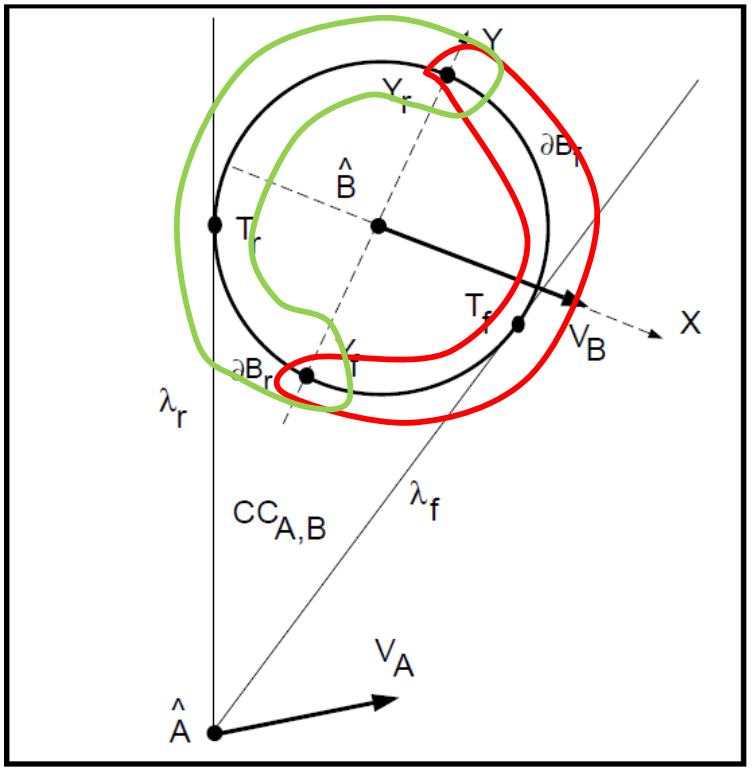
The set of reachable avoidance velocities, RAV, is defined as:

$$RAV(t + \Delta t) = RV(t + \Delta t) \ominus VO(t)$$

where $\ominus$ denotes the operation of set difference.



Parts of the obstacle



Using the collision cone diagram, if we attach a coordinate frame to the center of $\hat{B}$, we can specify which is the rear and front of the circular object:

GREEN = rear semicircle ∂B_r
RED = front semicircle ∂B_f

Structure of Avoidance Maneuvers

Lemma 1

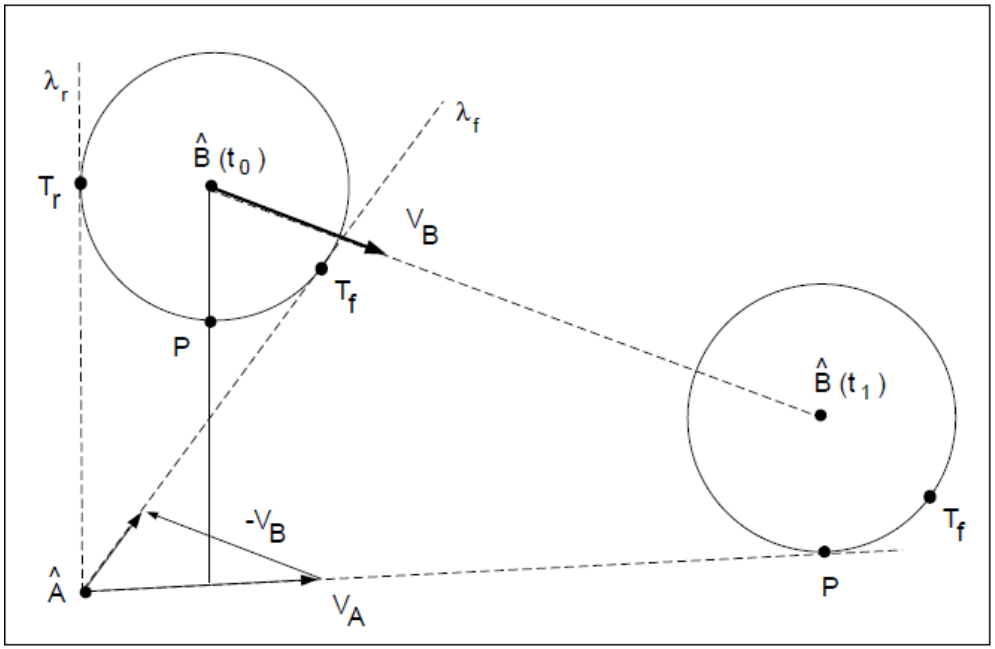
Robot A is tangent to obstacle B at some point $P \in \partial B$ if and only if it follows a trajectory generated by $\boldsymbol{v}_A$ corresponding to $\boldsymbol{v}_{A,B} \in \{\lambda_f, \lambda_r\}$. The tangency sets in ∂B consist of the shortest segment connecting $T_f = \lambda_f \cap \partial B$ to Y_f, and of the shortest segment connecting $T_r = \lambda_r \cap \partial B$ to Y_r.

Structure of Avoidance Maneuvers

What Lemma 1 means

Tangent maneuvers can only be generated only by relative velocities on λ_f and λ_r.

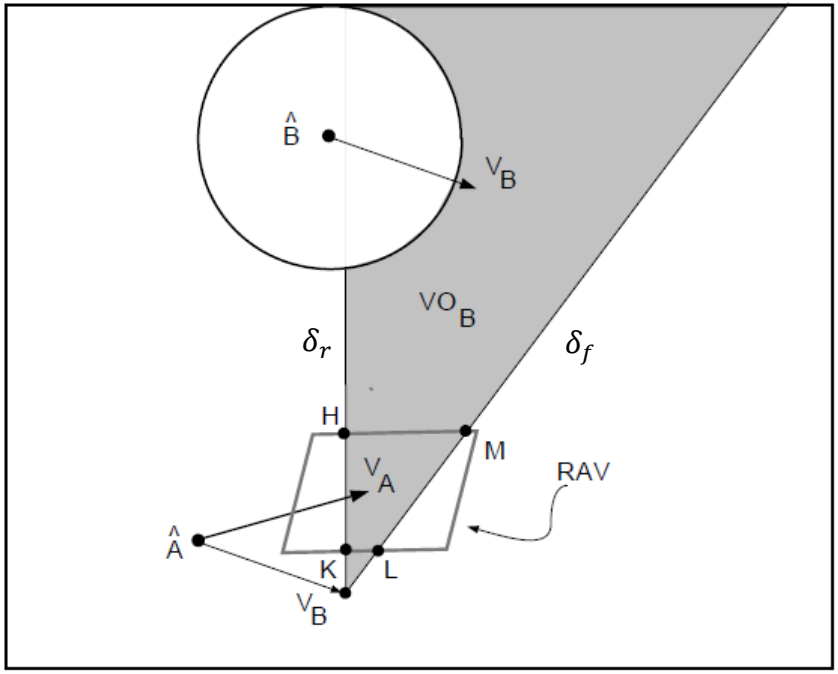
The actual points of tangency of these maneuvers are different from the tangency points T_f and T_r, since the points depend on the absolute velocities of A and B.



Structure of Avoidance Maneuvers

What Lemma 1 means

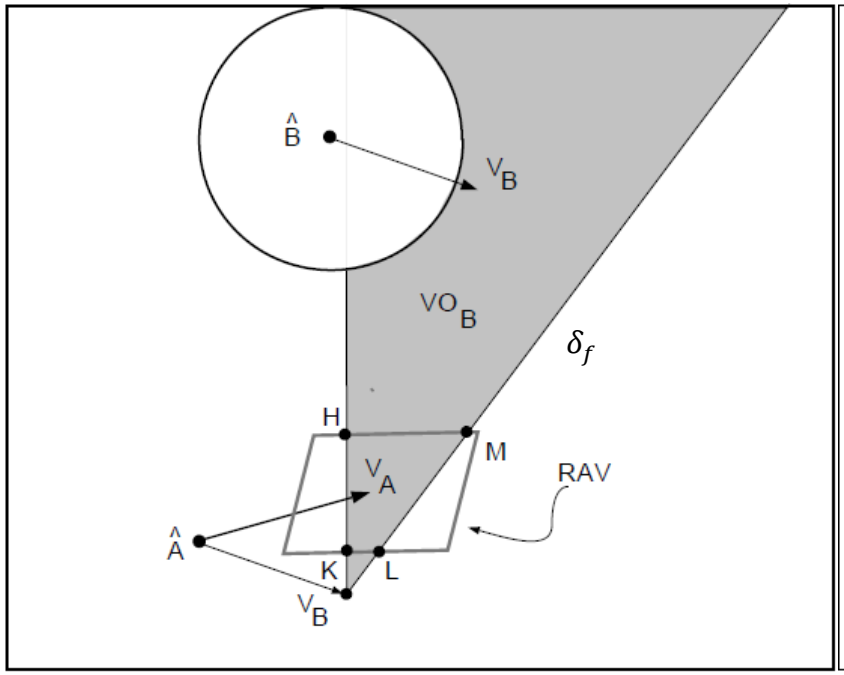
Velocities on δ_f or δ_r are tangent to $\hat{B}$ at some point in time

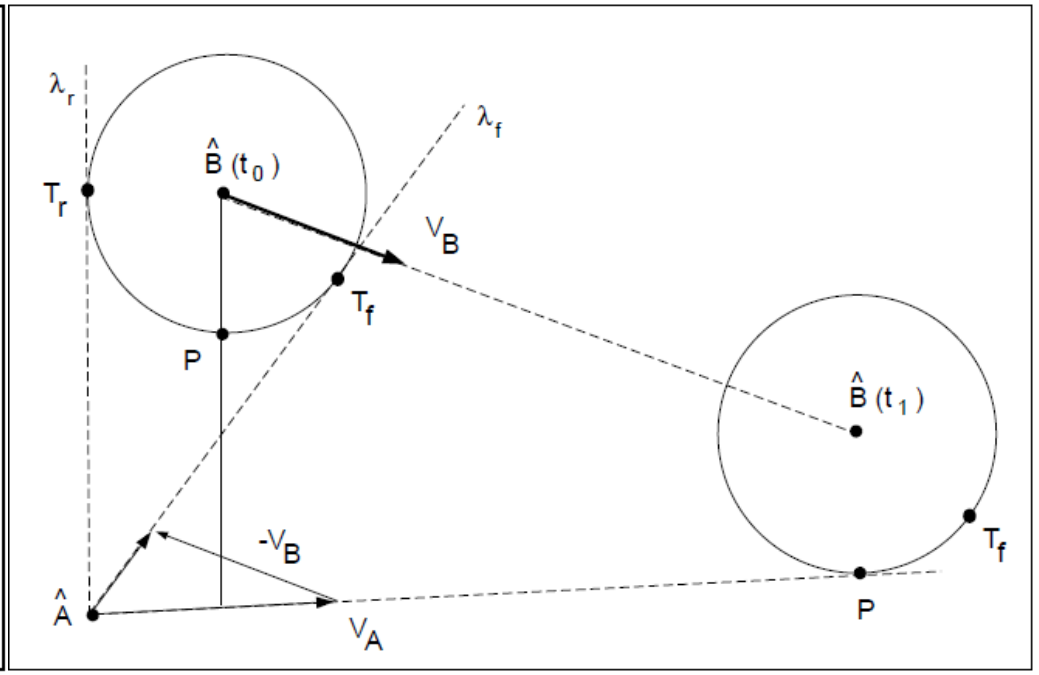


Structure of Avoidance Maneuvers

Lemma 1

Velocities on δ_f or δ_r are tangent to $\hat{B}$ at some point in time

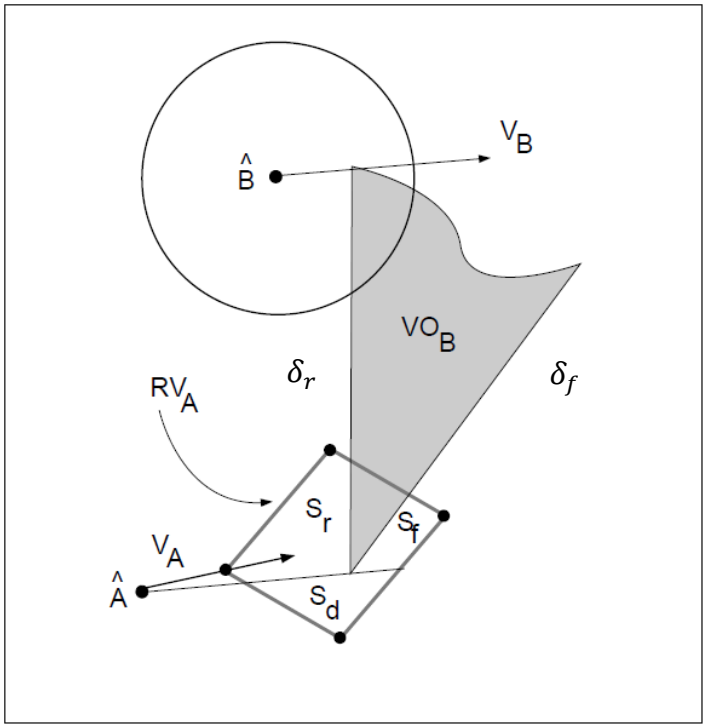




Structure of Avoidance Maneuvers

Lemma 2

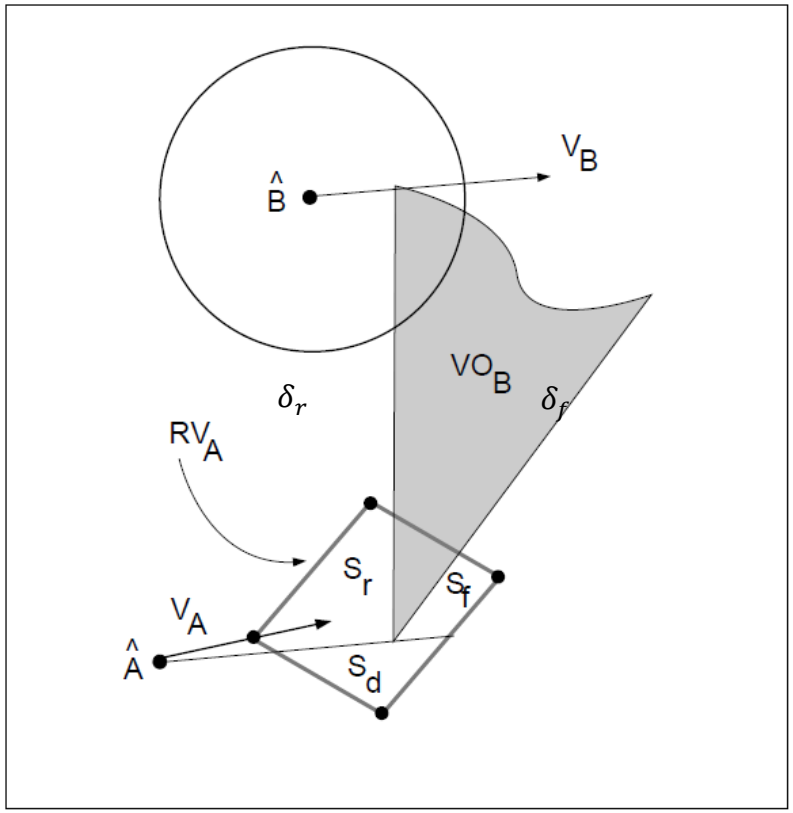
The reachable avoidance velocities RAV due to a single obstacle consists of at most three non-overlapping subsets S_f, S_r and S_d, each representing velocities corresponding to front, rear or diverging maneuvers respectively.



Structure of Avoidance Maneuvers

Lemma 2

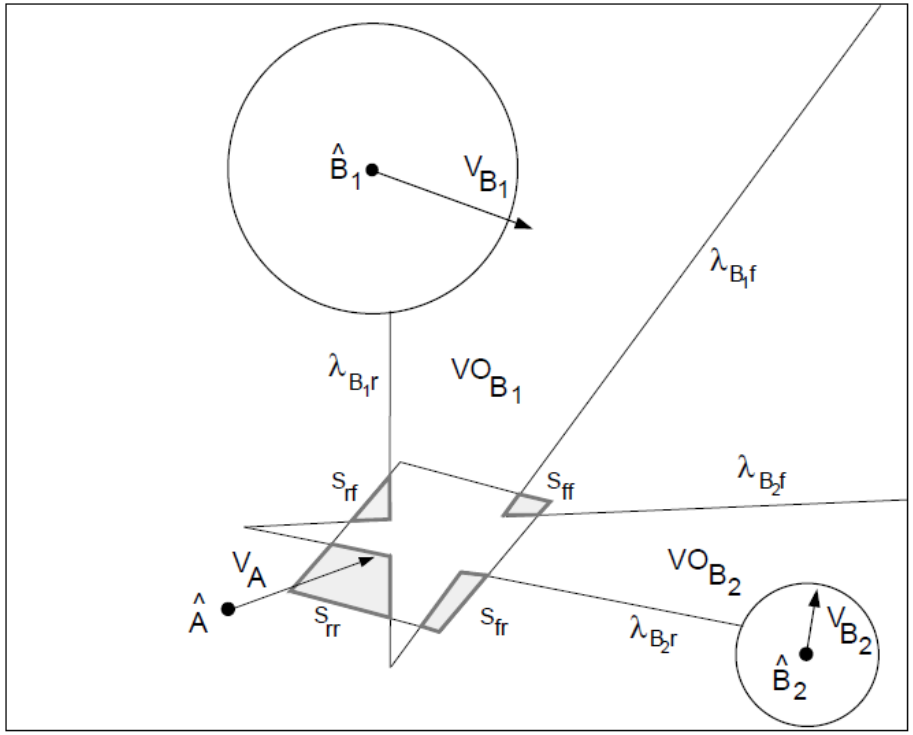
- Avoidance maneuvers are classified into three:
  - Front maneuvers (S_f)
  - Rear maneuvers (S_r)
  - Diverging maneuvers (S_d)



Structure of Avoidance Maneuvers

Theorem 1

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



How do we develop avoidance trajectories?

Generate a tree of feasible maneuvers computed at discrete time intervals.

At each time step, the robot should choose a velocity to take so that if the moving obstacle changes velocity, the robot would still be able to avoid collision.

Proposed Methods

- Global search
- Heuristic search

These proposed methods are tree generation algorithms.

Search tree

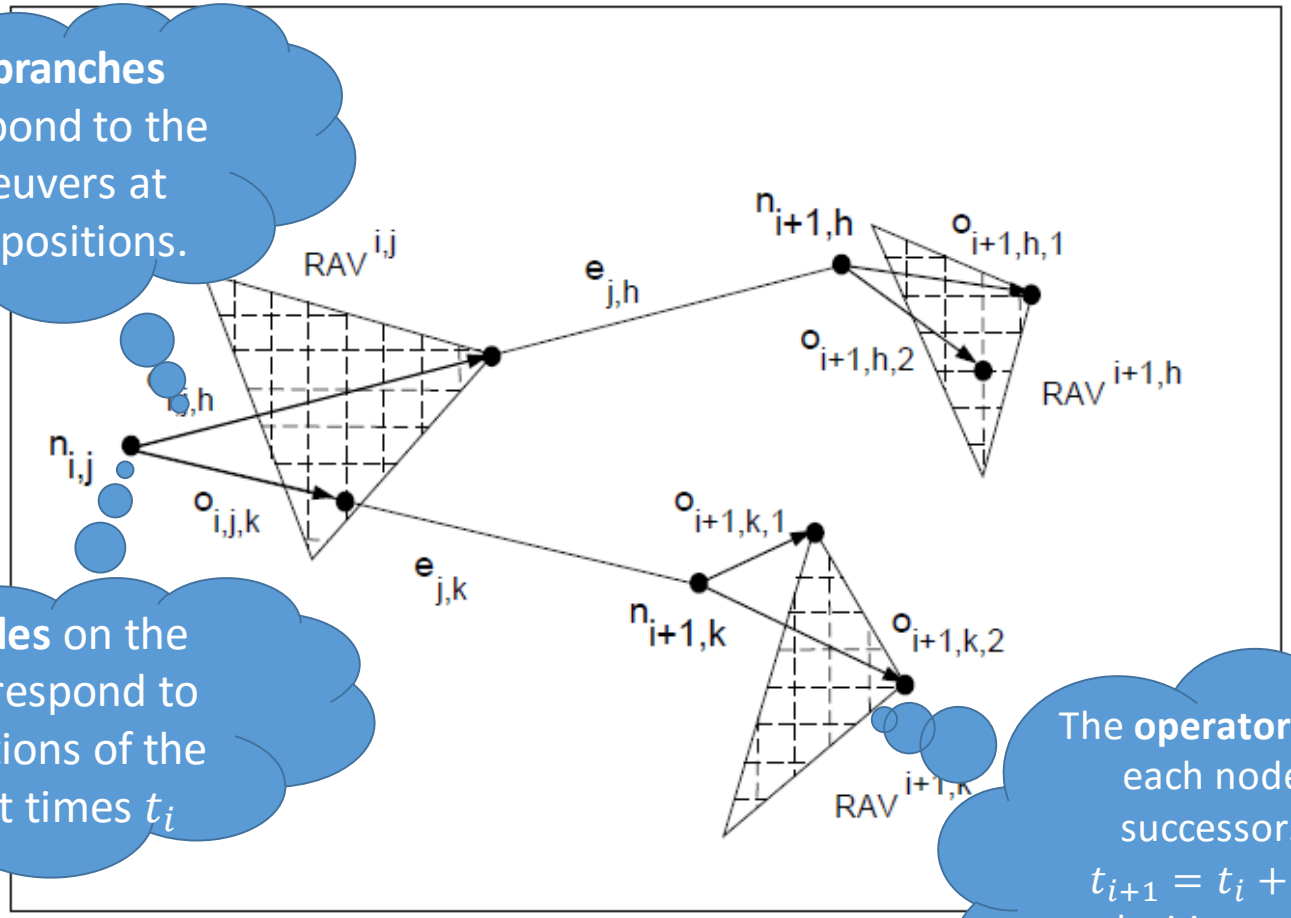
The search tree represents the available maneuvers a robot can take and its corresponding position at each discrete time step. Formally defined as:

$$n_j(t_i) = \left\{\left(\boldsymbol{x_j}, RAV(t_i)\right)\right\}$$
$$o_{j,l}(t_i) = \left\{\boldsymbol{v_l}(t_i)\middle| v_l(t_i) \in RAV_j(t_i)\right\}$$
$$e_{j,k}(t_i) = \left\{\left(n_j(t_i), n_j(t_{i+1})\right)\middle|\, n_k(t_{i+1}) = n_j(t_i) + \left(o_{j,l}T\right)\right\}$$

where

1. $n_j(t_i)$ is the jth node at time t_i
2. $RAV_j(t_i)$ is the reachable velocity set computed for the node n_j
3. $o_{j,l}$ is the lth operator on node j at time t_i
4. $e_{j,k}$ is the branch between node n_j at time t_i, and node n_k at time t_{i+1}.

Search tree



The **branches** correspond to the maneuvers at those positions.

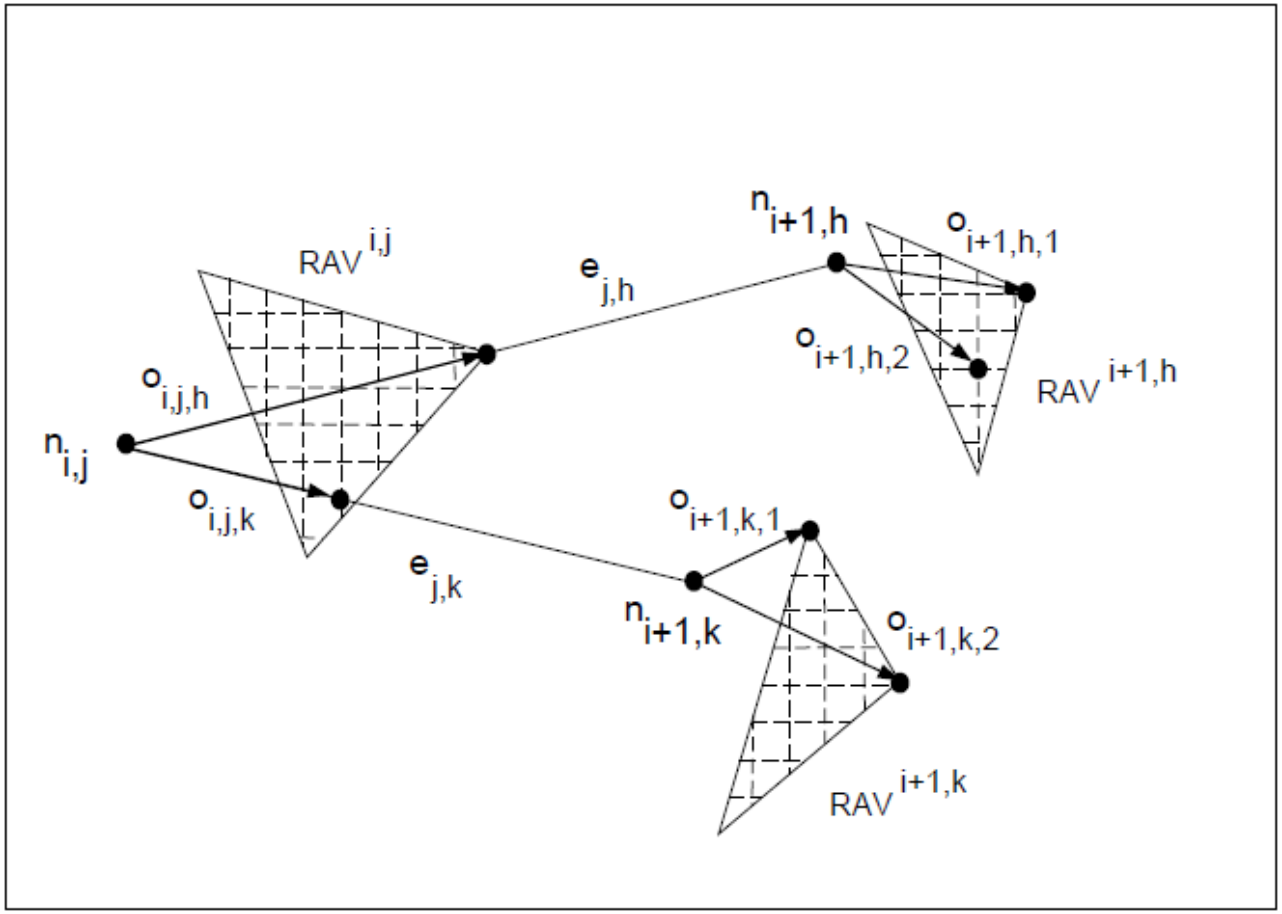
The **nodes** on the tree correspond to the positions of the robot at times t_i

The **operators** expanding each node into its successors at time $t_{i+1} = t_i + T$ are the velocities computed by discretizing RAV.

Global search

- Global search is an off-line method
- Generates reachable avoidance velocities at discrete time intervals.
  - RAVs are discretized by grids.
  - Positions reached by robots are the successors of the node.
- Trajectories generated avoid all obstacles at a specific time horizon.

Search tree

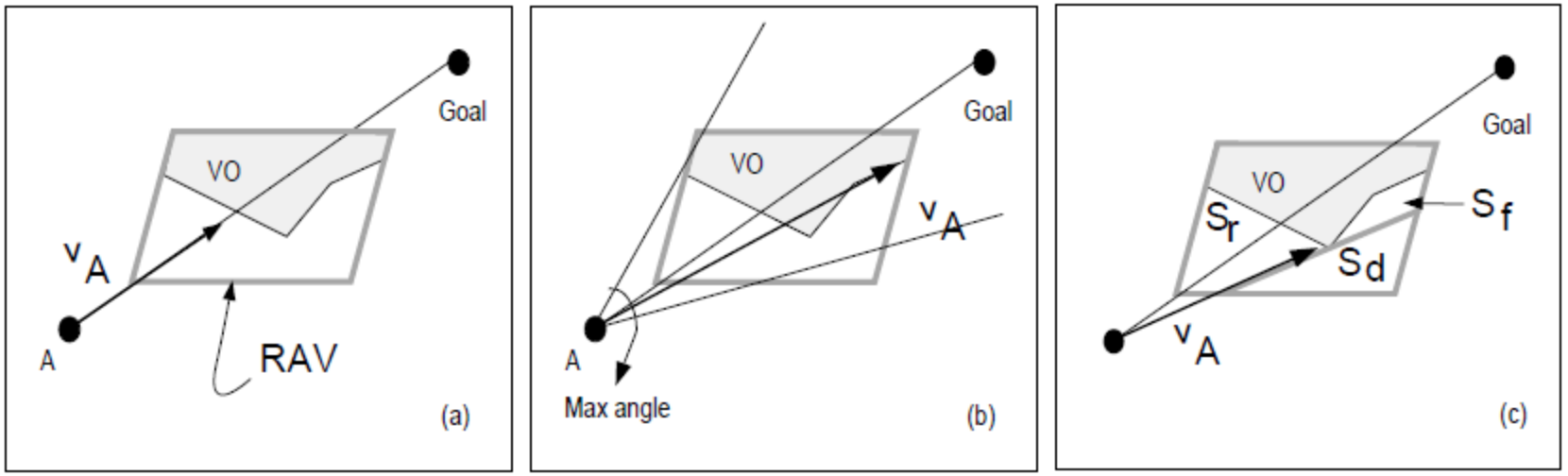

RAV i,j
e j,h
n i+1,h
o i+1,h,1
o i+1,h,2
RAV i+1,h
o i,j,h
n i,j
o i,j,k
e j,k
o i+1,k,1
n i+1,k
o i+1,k,2
RAV i+1,k

Heuristic search

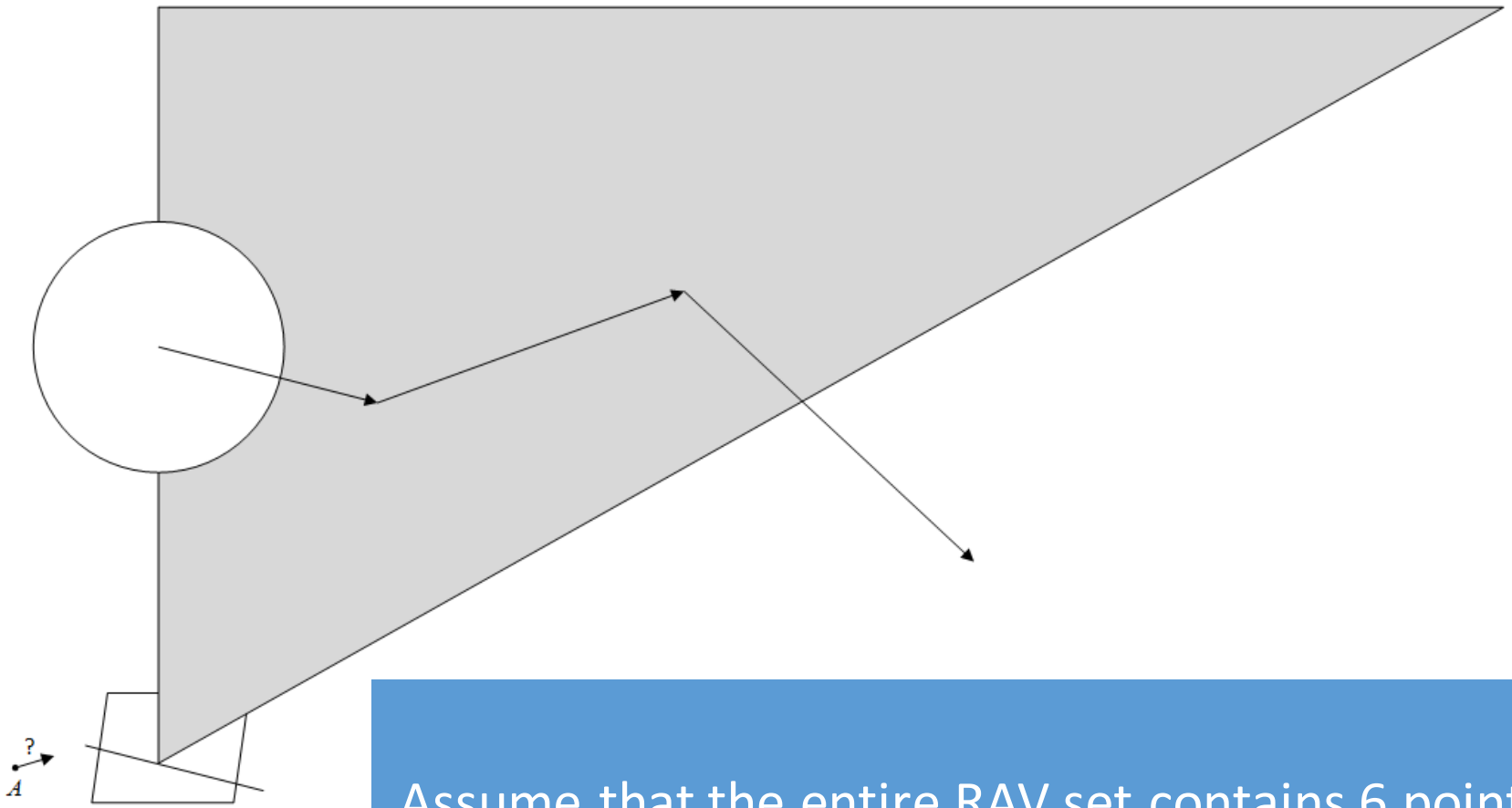
- Heuristic search involves the generation of trajectories on-line.
- Expand nodes that correspond to the robot's current position and generate only one branch per node, depending on a heuristic arbitrarily chosen by a path planner.
- Path planners arbitrarily define heuristics that would let the robot satisfy a goal.
  - Examples:
    - Survival of the robot
    - Reaching the desired target
    - Minimizing a performance criteria
    - Selecting a desired trajectory structure
- There is no guarantee that objectives set by the path planners will be achieved at any time.

Heuristic search

- In this paper, three kinds of heuristics were used:
  - Choosing the highest avoidance velocity along the line to the goal **(TG = to goal)**
  - Selecting the maximum avoidance velocity with some specified angle α from the line to the goal **(MV = maximum velocity)**
  - Selecting the velocity that avoids the obstacles according to their perceived risk. **(ST = structure)**

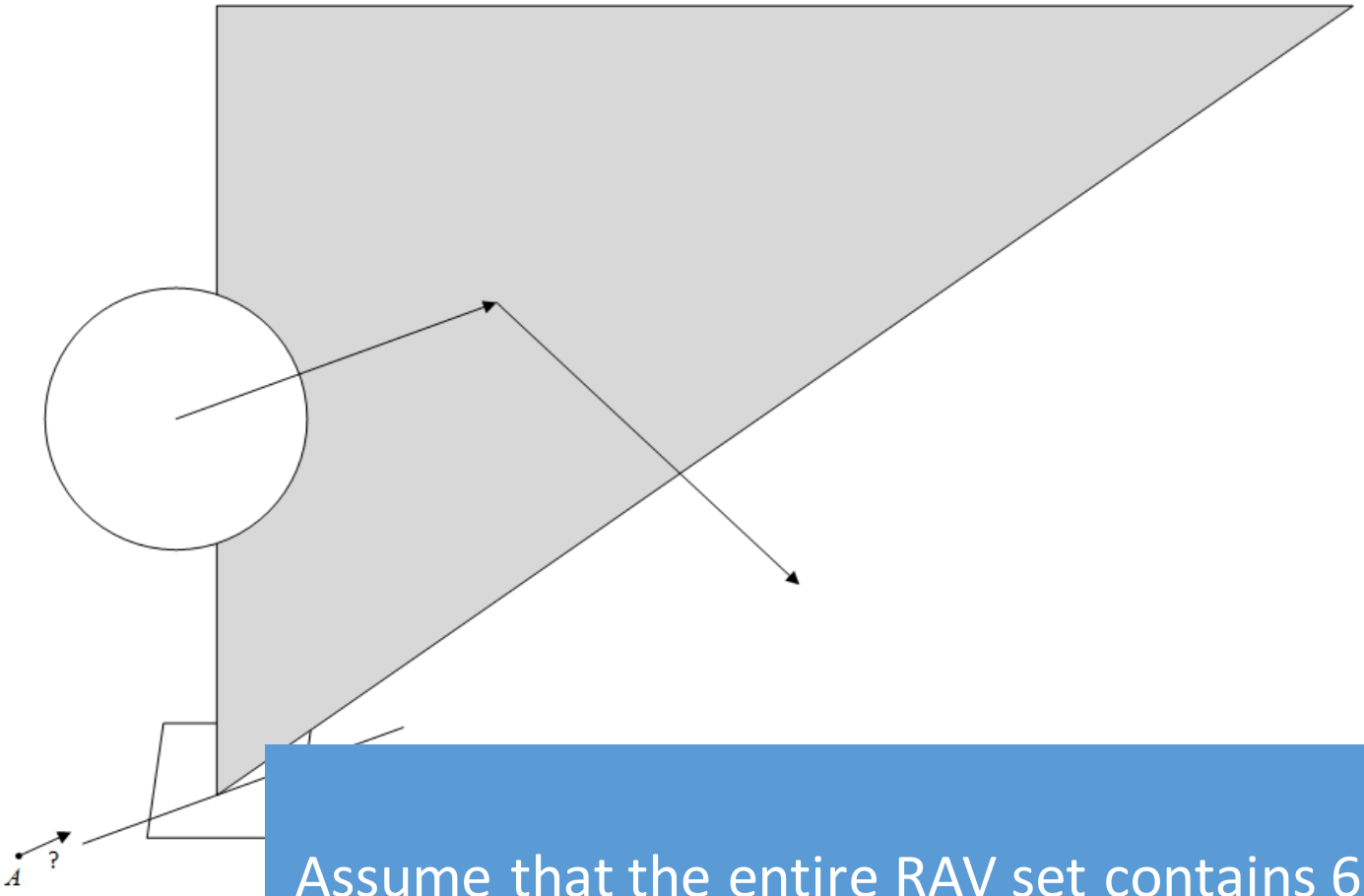


Example (step 1)



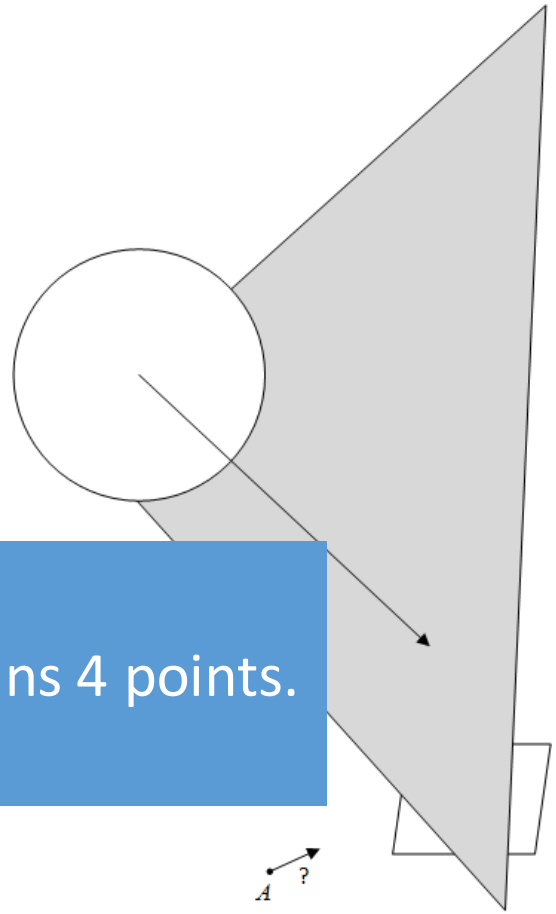
Assume that the entire RAV set contains 6 points.

Example (step 2)



Assume that the entire RAV set contains 6 points.

Example (step 3)



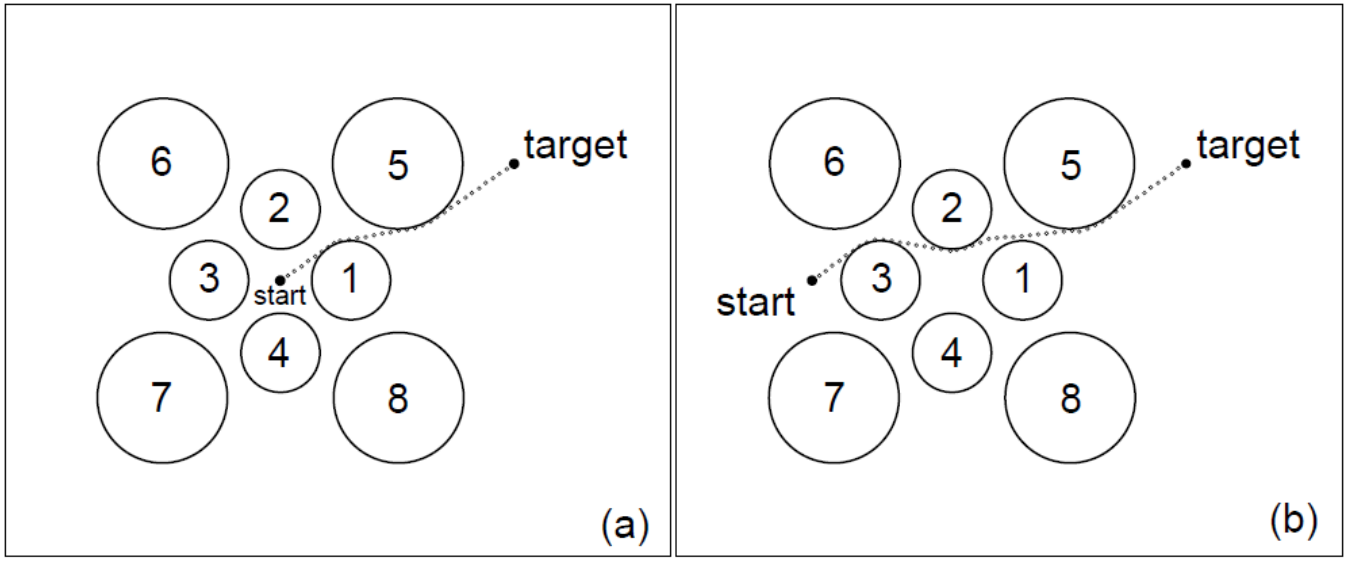
Assume that the entire RAV set contains 4 points.

Examples

- Avoidance of static obstacles
- Avoidance of fixed and moving obstacles
- Intelligent Highway

Example 1: Avoidance of static obstacles

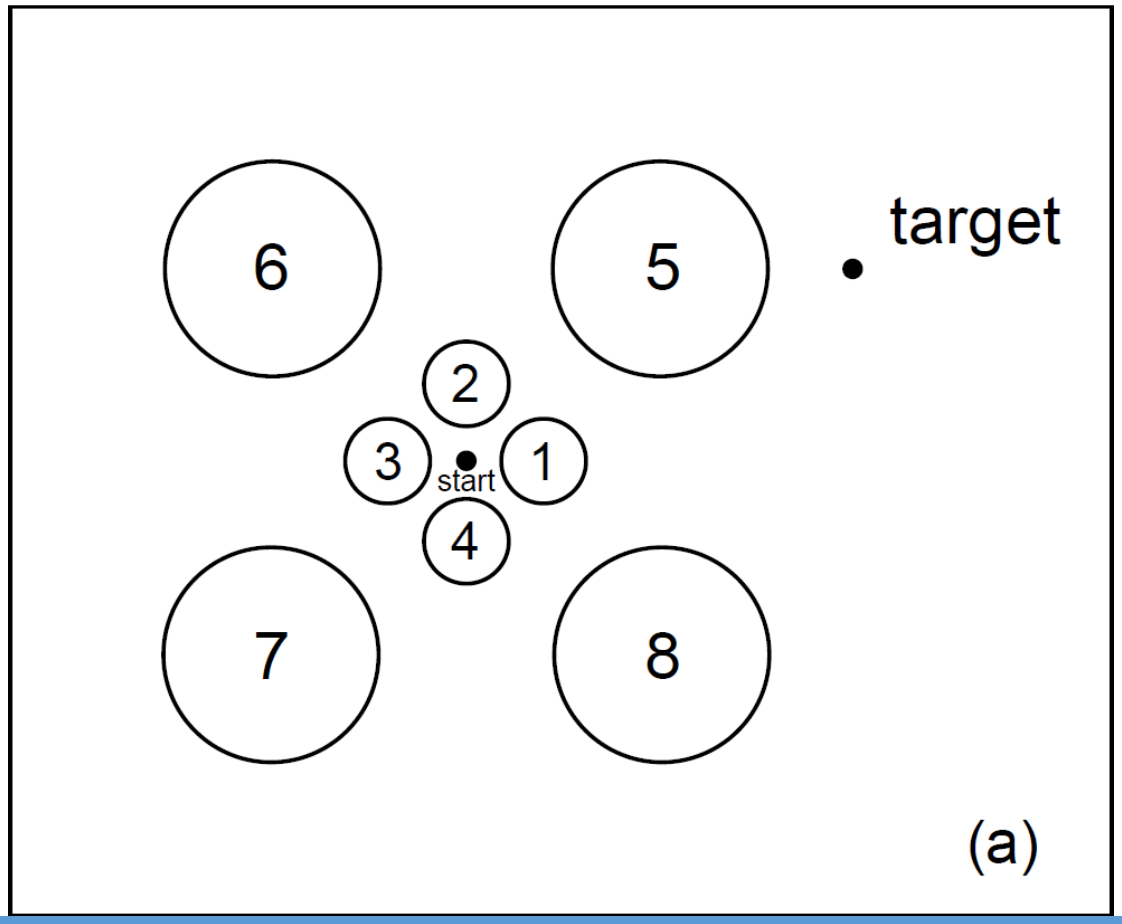
- A point robot avoids eight circular obstacles starting from rest.
- Used MV heuristics.
- Trajectories were computed every 1 s.



Example 2: Avoidance of fixed and moving obstacles

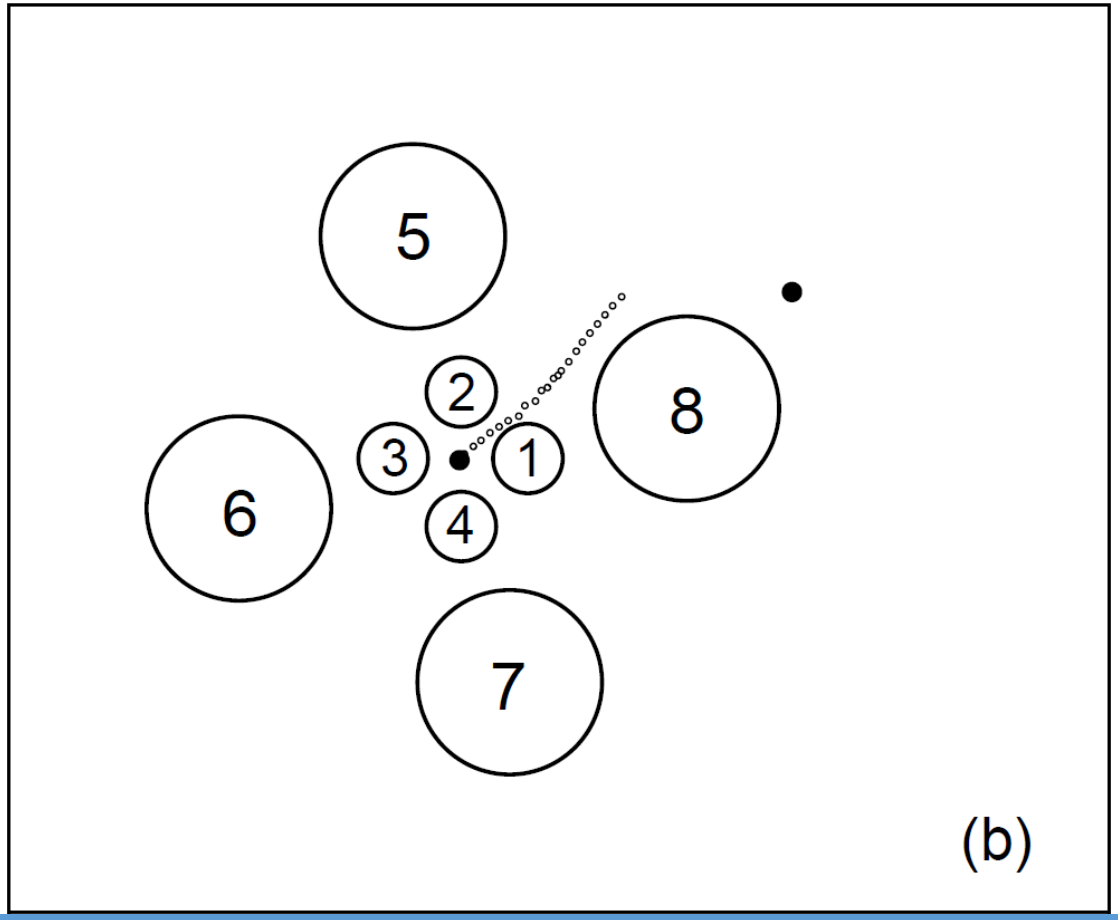
- Robot starts from rest at the center.
- The robot should first avoid static objects at the center and then large moving objects.
- The robot used MV heuristics.

Example 2: Avoidance of fixed and moving obstacles (1)



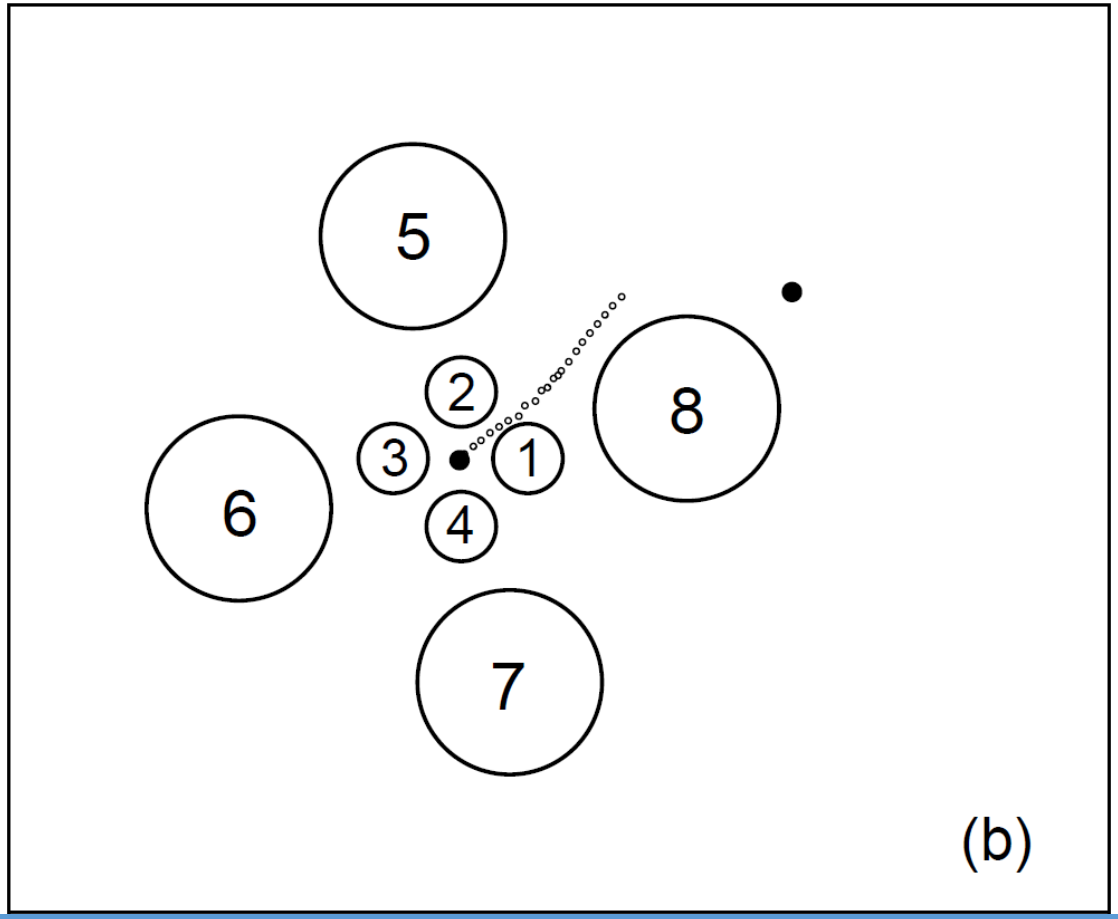
Robot is situated at the center. $(t = 0)$

Example 2: Avoidance of fixed and moving obstacles (2)



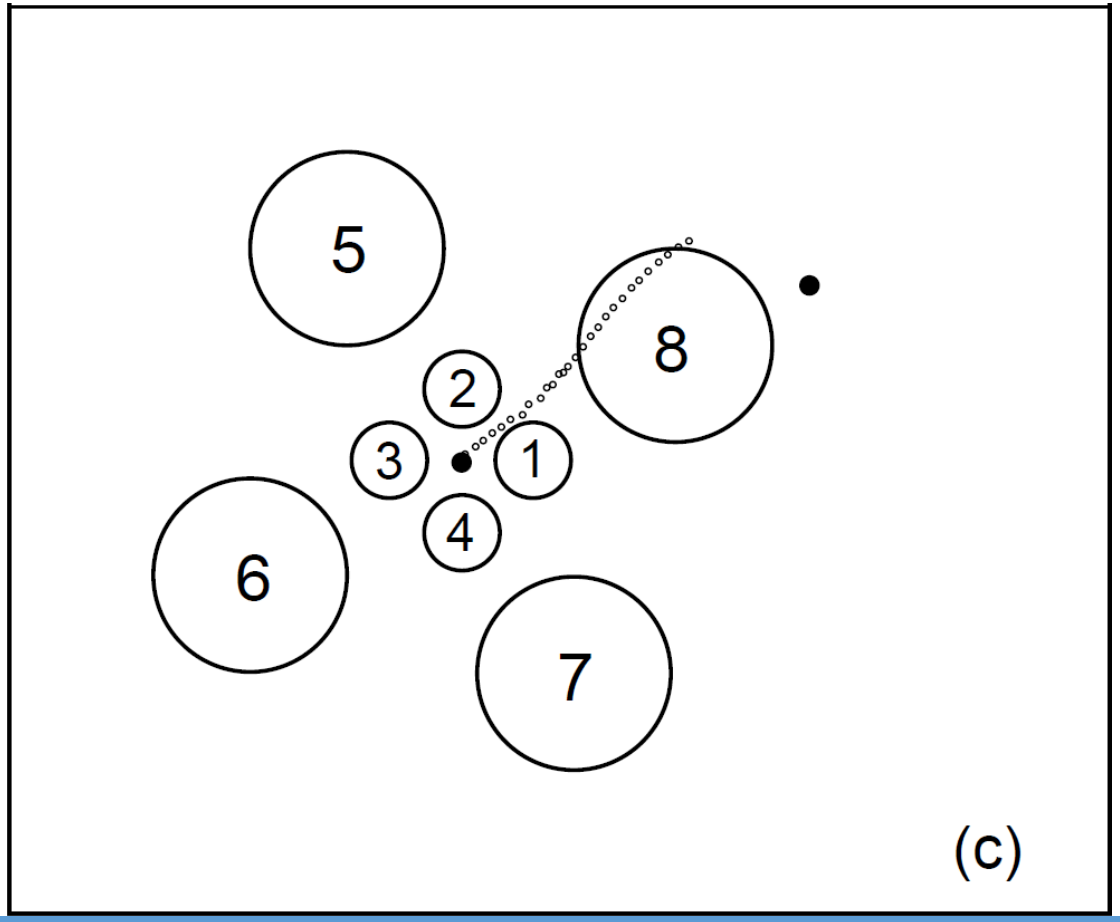
The robot first avoids the small static obstacles

Example 2: Avoidance of fixed and moving obstacles (2)



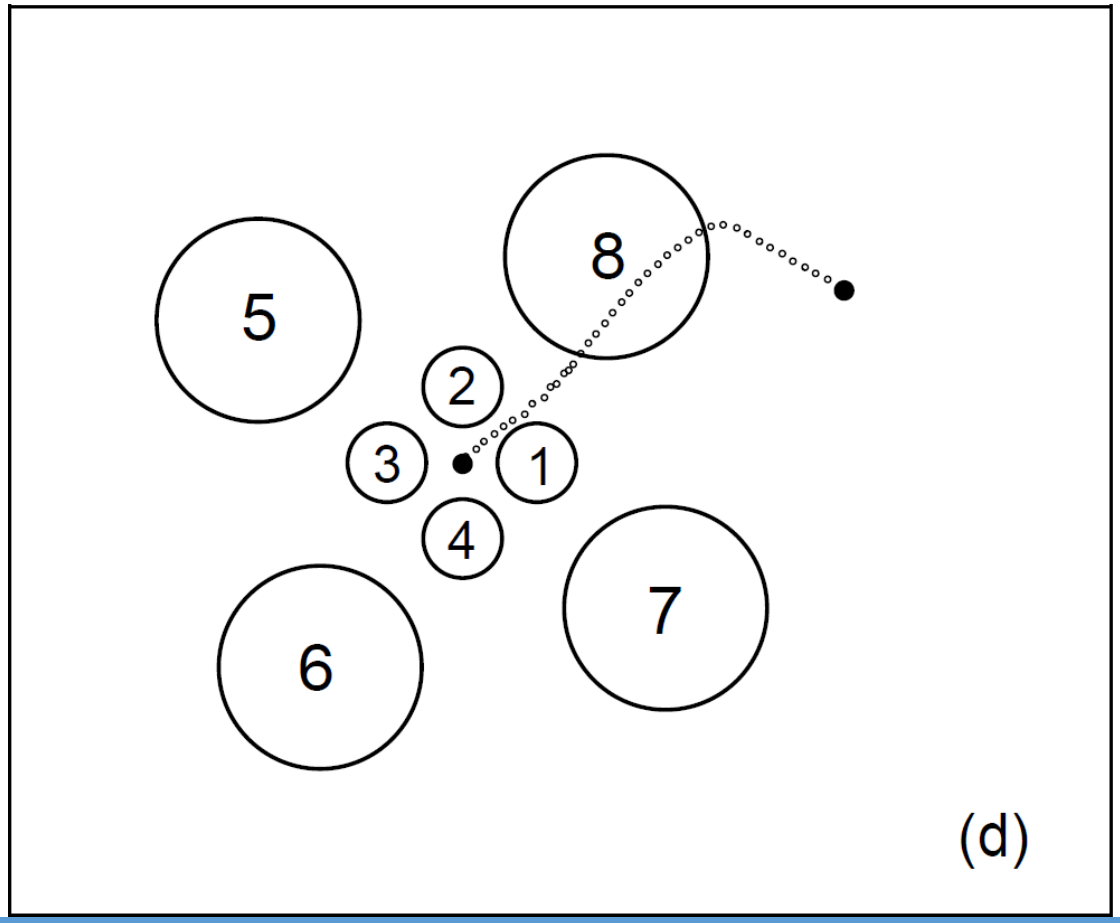
The robot moves to avoid the incoming obstacle 8, but it first slows down and accelerates. $(t = 24\ s)$

Example 2: Avoidance of fixed and moving obstacles (3)



The robot becomes tangent to obstacle 8. ($t = 29\ s$)

Example 2: Avoidance of fixed and moving obstacles (4)



Robot accelerates to target. ($t = 29\ s$)

Example 3a: Moving to an exit ramp

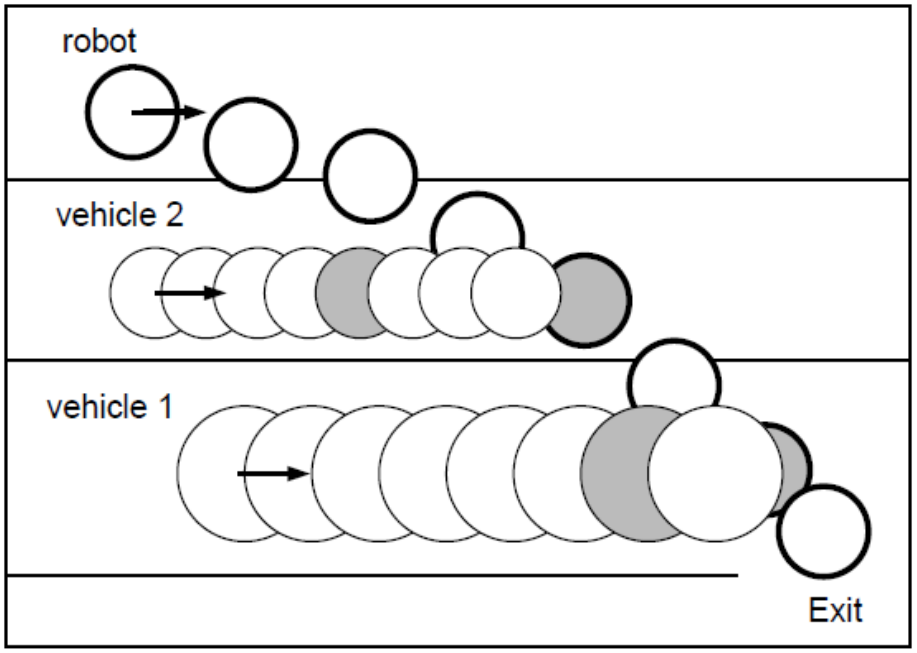
- A robotic vehicle is located at the leftmost lane trying to reach an exit ramp located on the right.
- There are two moving vehicles that move at constant speeds.
  - Vehicle 1 ($v_x = 30$ m/s, $v_y = 0$)
  - Vehicle 2 ($v_x = 23$ m/s, $v_y =$)
- Used Global Search and the Heuristic Search (TG and MV)

Example 3a: Moving to an exit ramp

- Used a global search (Depth First Iterative Deepening Algorithm)
- Nodes of the tree were generated every $1\ s$
- The RAV sets were discretized, on average, 12 points

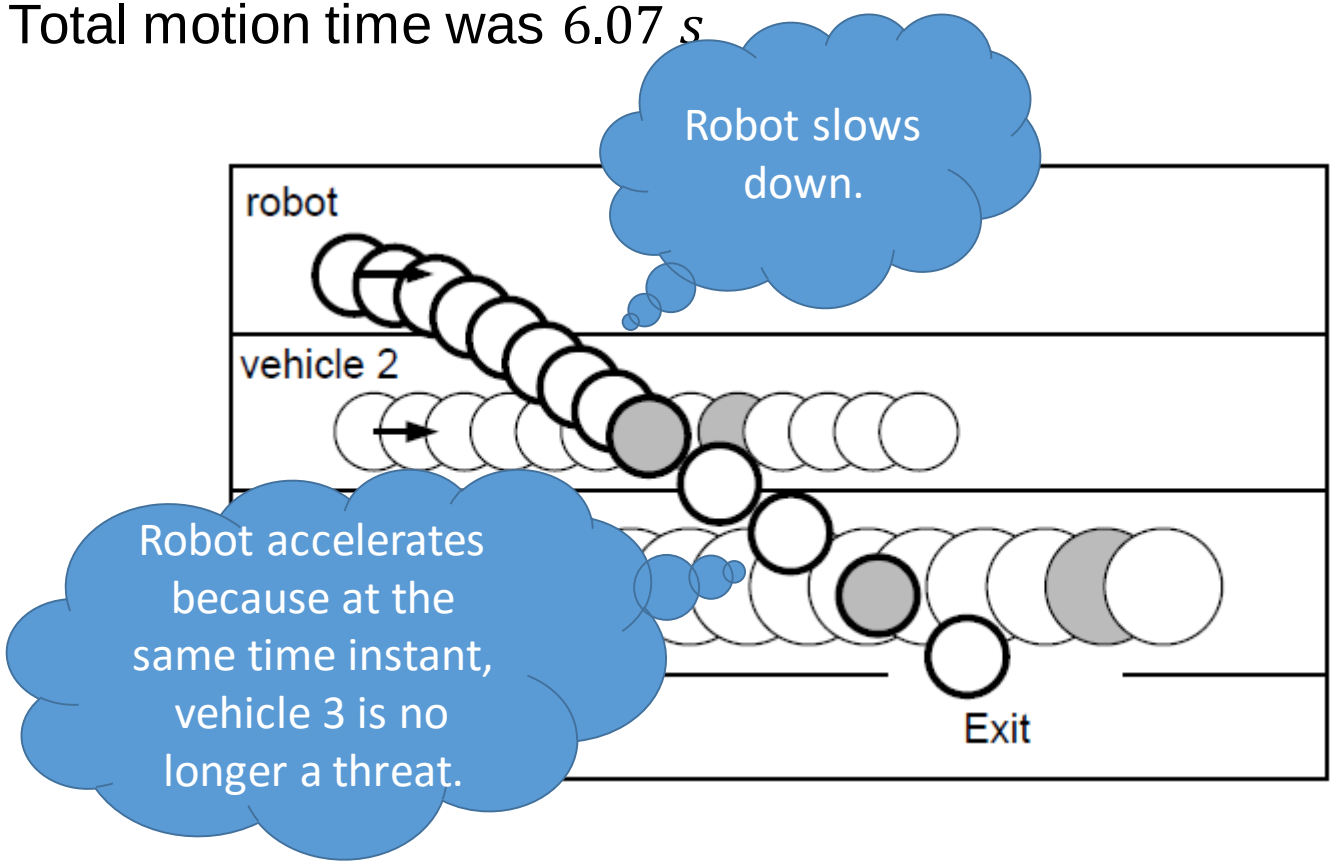
Example 3a: Moving to an exit ramp

- The total motion time was $3.5\ s$



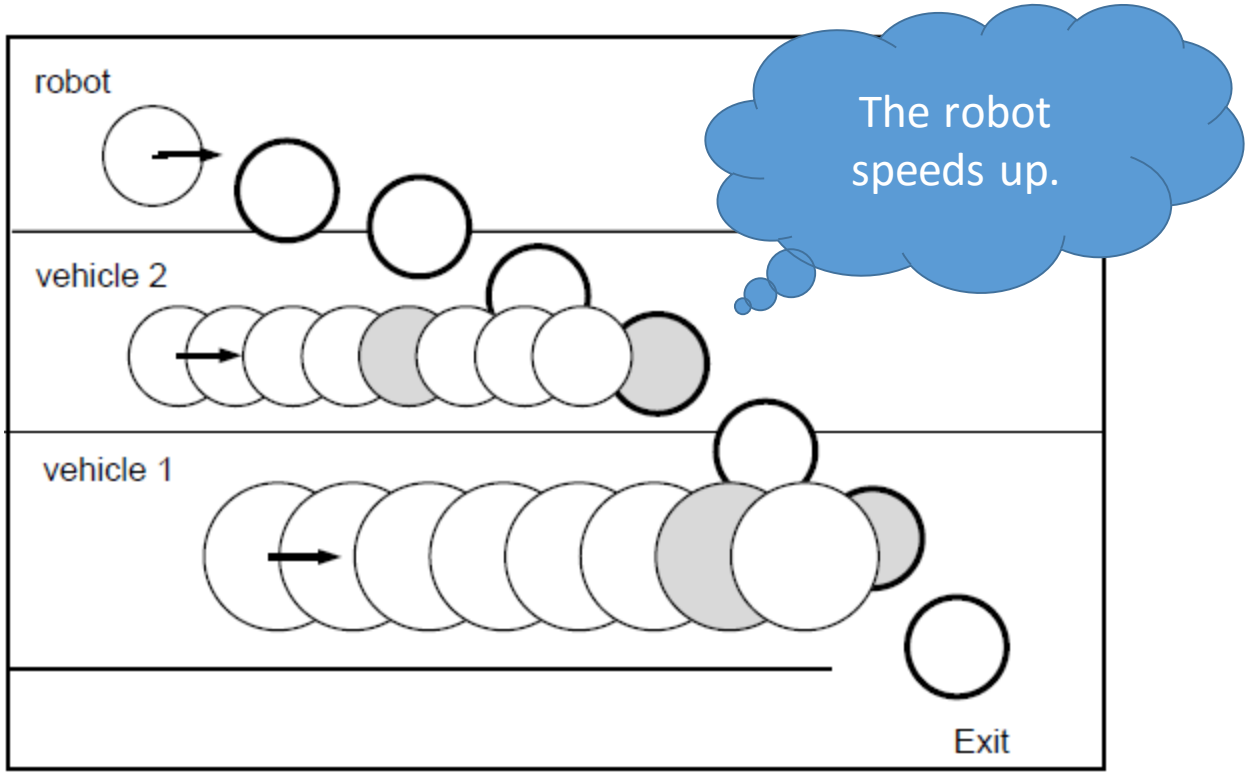
Example 3a: Moving to an exit ramp

- Used TG heuristics
- Total motion time was $6.07\ s$



Example 3a: Moving to an exit ramp

- Used MV heuristics
- The total motion time was $3.56\ s$

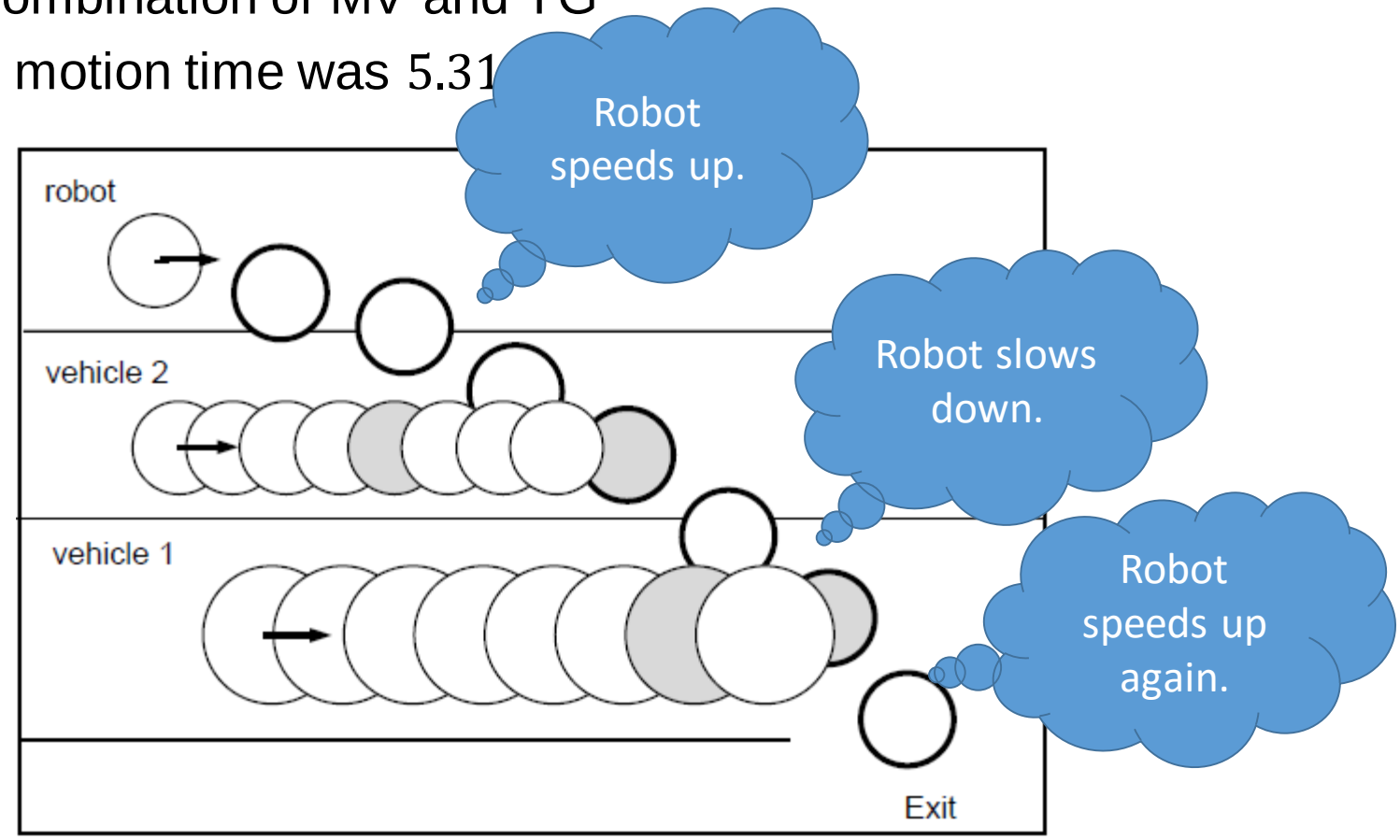


Example 3a: Moving to an exit ramp

- Used a combination of MV and TG
- MV heuristics was the basis for the robot's movement when $0 \leq t \leq 2.0\ s$
- TG heuristics was the basis for the robot's movement afterwards
- The total motion time was $5.31\ s$

Example 3a: Moving to an exit ramp

- Used a combination of MV and TG
- The total motion time was 5.31



Velocity obstacle

Notes

- Selecting a v_A **<u>outside</u>** of VO would **<u>avoid collision</u>**, or symbolically:

$$A(t) \cap B(t) = \emptyset \text{ if } \boldsymbol{v_A} \notin VO(t)$$

Velocities on the boundaries of VO would result in A grazing (or slightly touching) B.

- For multiple obstacles, we have

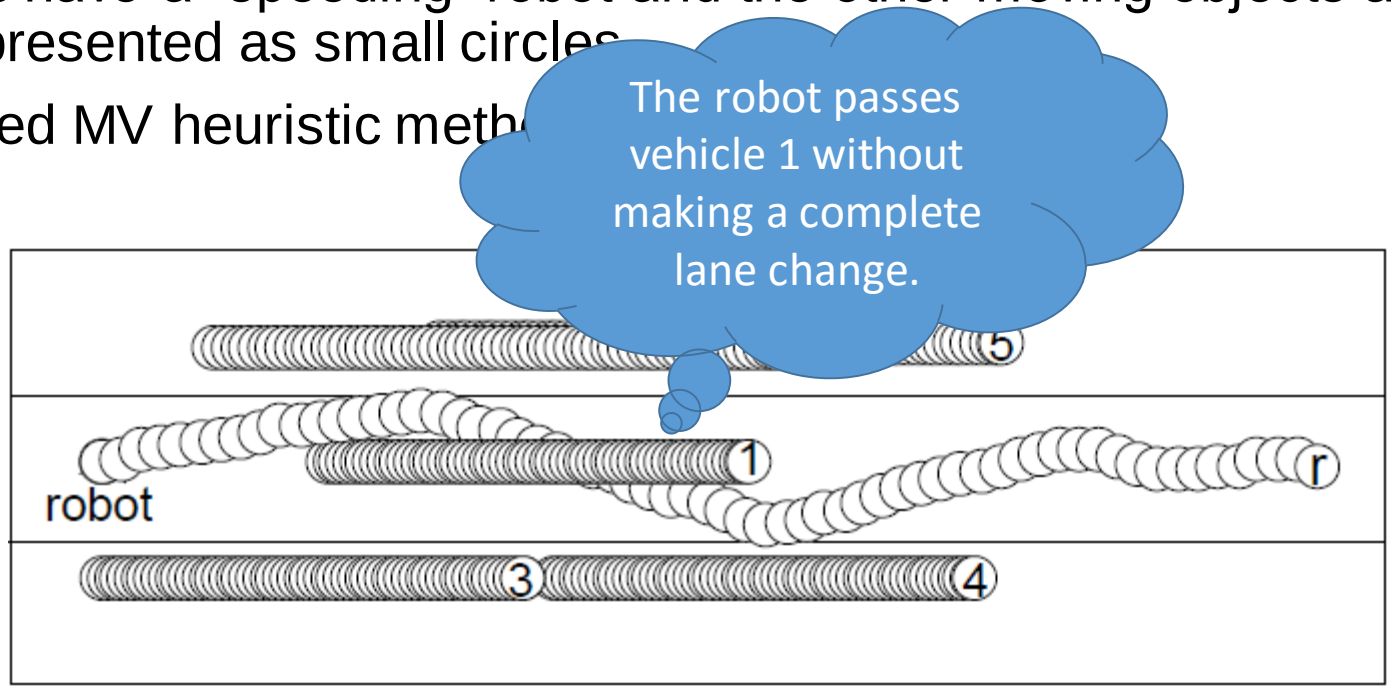
$$VO = \cup_{i=1}^{m} VO_{B_i}$$

where m is the number of obstacles.

- We assign priorities to each obstacle and the one with an imminent collision will have the greatest precedence.

Example 3b: Negotiating Highway Traffic

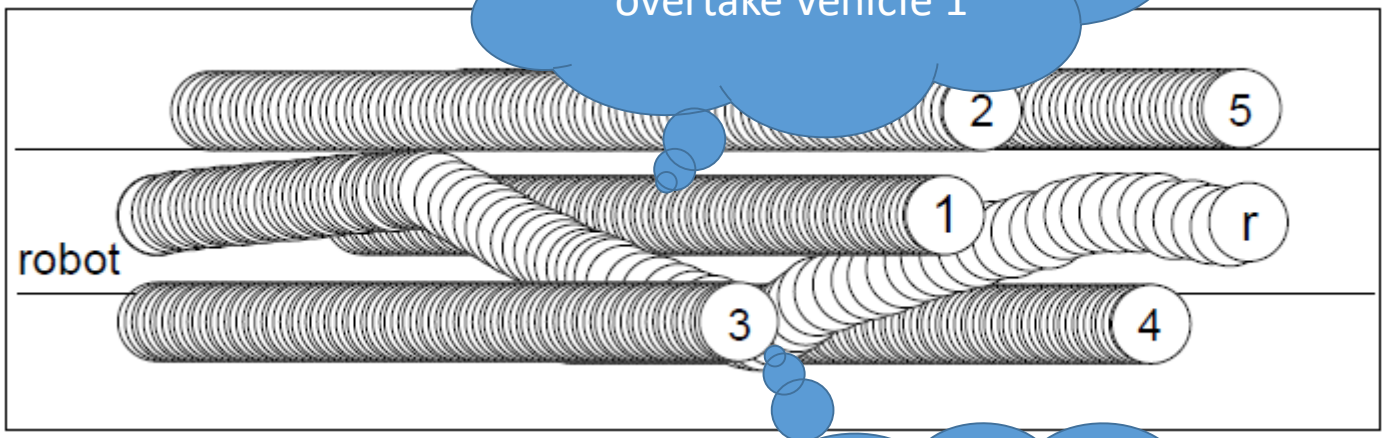
- We have a “speeding” robot and the other moving objects are represented as small circles
- Used MV heuristic meth



The robot overtook vehicle 1 and continued to take the same lane.

Example 3b: Negotiating Highway Traffic

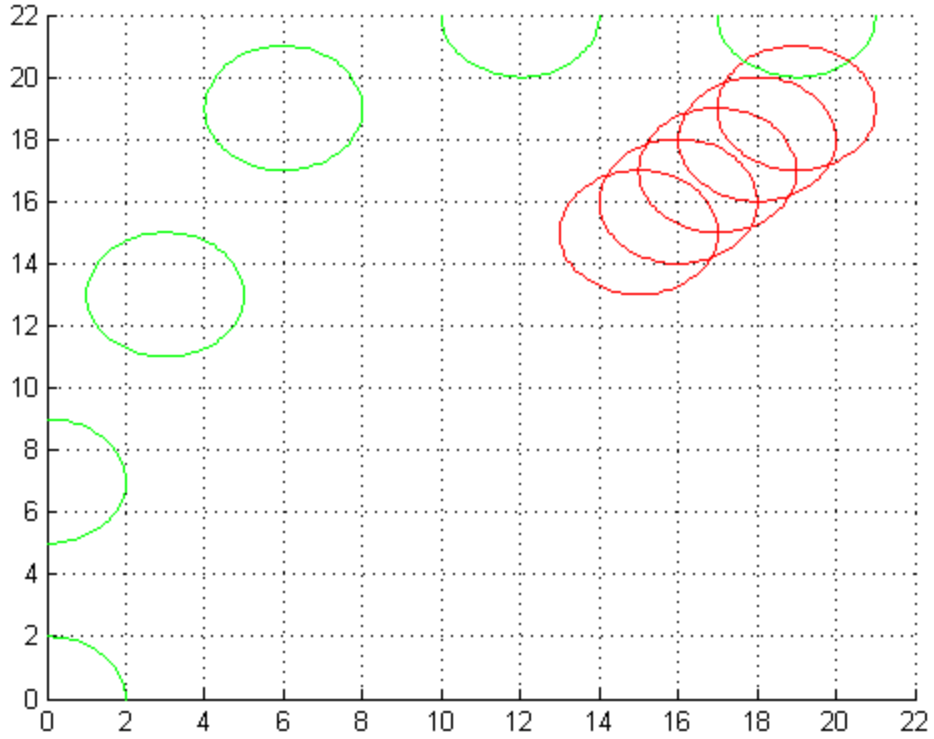
- We have a "speeding" robot and the other moving objects are represented as big circles
- Used MV heuristic meth



Because the moving vehicles were o d down to attempt a lane change to the left elocity to the right became feasible so the robo 1 on the right and returned to the center lane.

Example 4: Moving towards the start of a moving obstacle

- MV heuristics applied
- Robot's maximum velocity is 10 m/s
- Obstacle is moving at a speed of 3 m/s



Summary

- A first-order method for planning the motion of a robot in dynamic environments was presented.
  - Simple geometric representation
  - Avoids all kinds of obstacles (static and dynamic)
  - Scalable (possible to increase and decrease the number of obstacles)