CPSC 8810

Motion Planning

03 – Velocity Obstacles

January 15, 2020

Announcements

- TA office hours: Tuesdays, 2-3pm
  - Dial 736765509@clemson.webex.com
  - Or connect via this link
- Instructor office hours: Thursdays, 3-4:30pm
- Resources
  - Lecture pdfs available on the website under “Schedule”
  - Lecture recording available on Canvas

Today

- Local Navigation (Part II)
  - Velocity Obstacles
- What you need for the first programming assignments
- Hand in the 1st assignment

Recap

Local navigation

- We are given an environment containing static and dynamic obstacles and n agents
- The task for each agent A_i is to compute a *collision-free velocity that is as close as possible to its goal velocity*

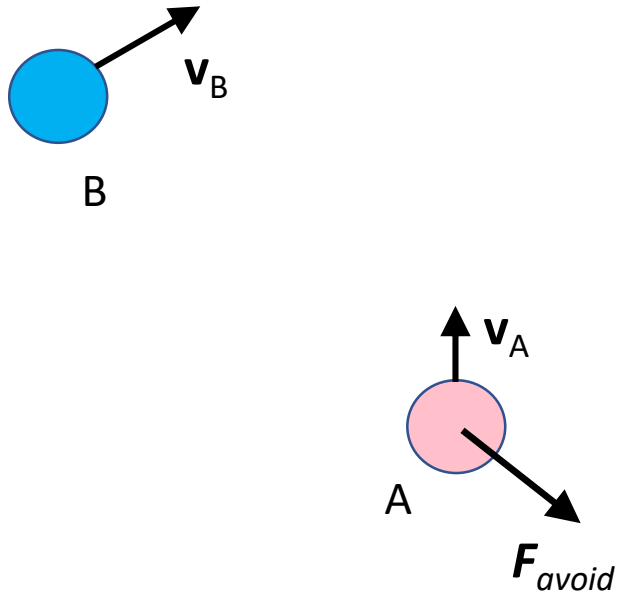
Local navigation methods

- Implementation-wise, most models for local collision avoidance can be classified as
  - Force-Based
    - The behavior of each agent is modelled as a collection of forces

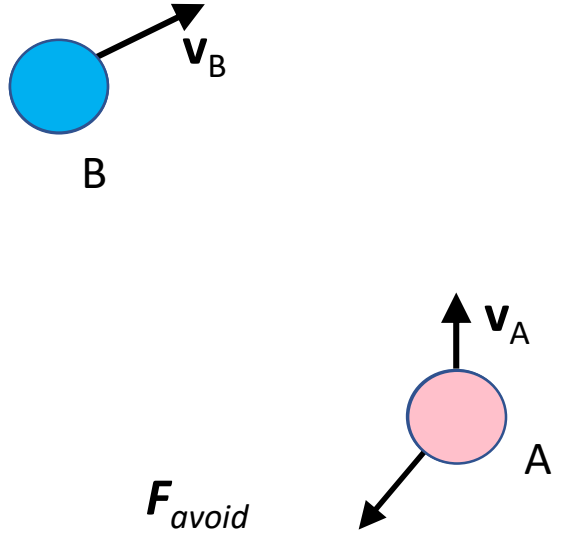
      [Reynolds 1987, Helbing et al. 2000, Karamouzas et al. 2014, …]
  - Velocity-Based
    - Directly plan a new velocity for each agent

      [Fiorini and Schiller 1998, Pettré et al. 2009, van den Berg et al 2011, Guy et al. 2011, …]

Reactive vs Predictive

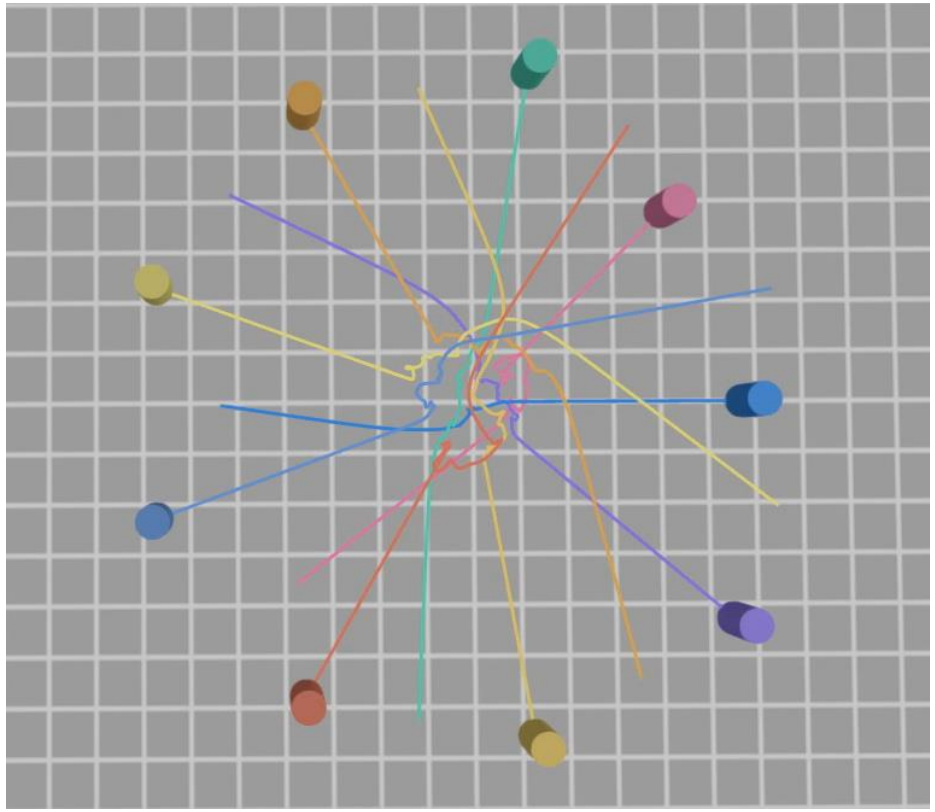


Distance-based forces

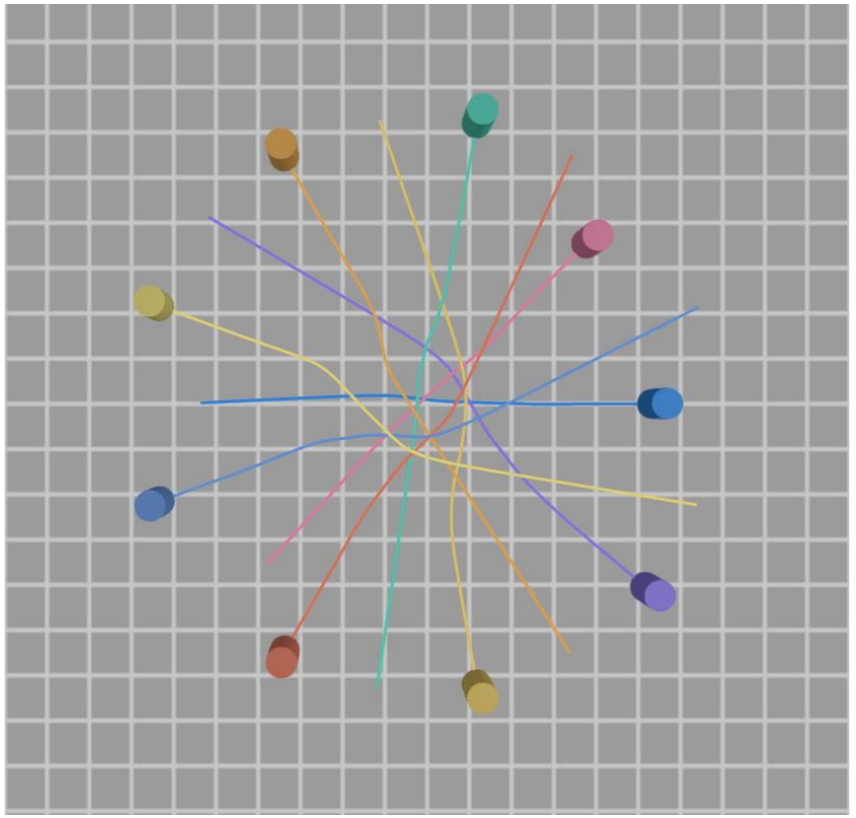


TTC forces

Reactive vs Predictive



Distance-based force



TTC force

Force-based approaches

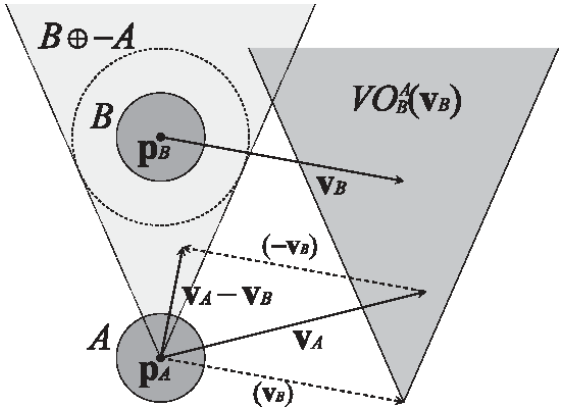
- Pros
  - Easy to implement
  - Fast to compute

- Cons
  - Hard to make collision-free guarantees
  - Requires parameter tuning
  - May not be intuitive to think about forces

Velocity-Based Navigation

Velocity-based approaches

- Plan directly in the velocity space
  - Can be more intuitive than force-based approaches
  - Typically relies on the concept of Velocity Obstacles
- Formulate collision avoidance as an optimization problem, which can be solved using, e.g.,
  - Sampling [van den Berg et al. 2008, Moussaïd et al. 2011, ….]
  - Linear programming [van den Berg et al. 2009]
  - Numerical methods [Dutra et al. 2017, Karamouzas et al. 2017, ..]



Velocity-Based Navigation

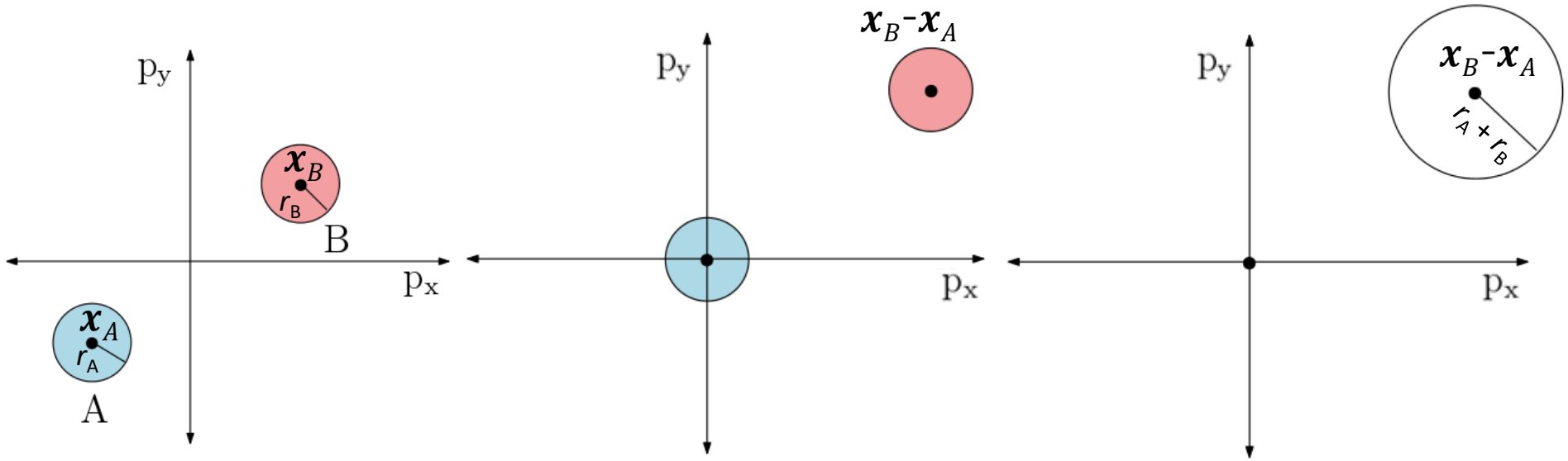
Velocity Obstacles

Velocity Obstacles

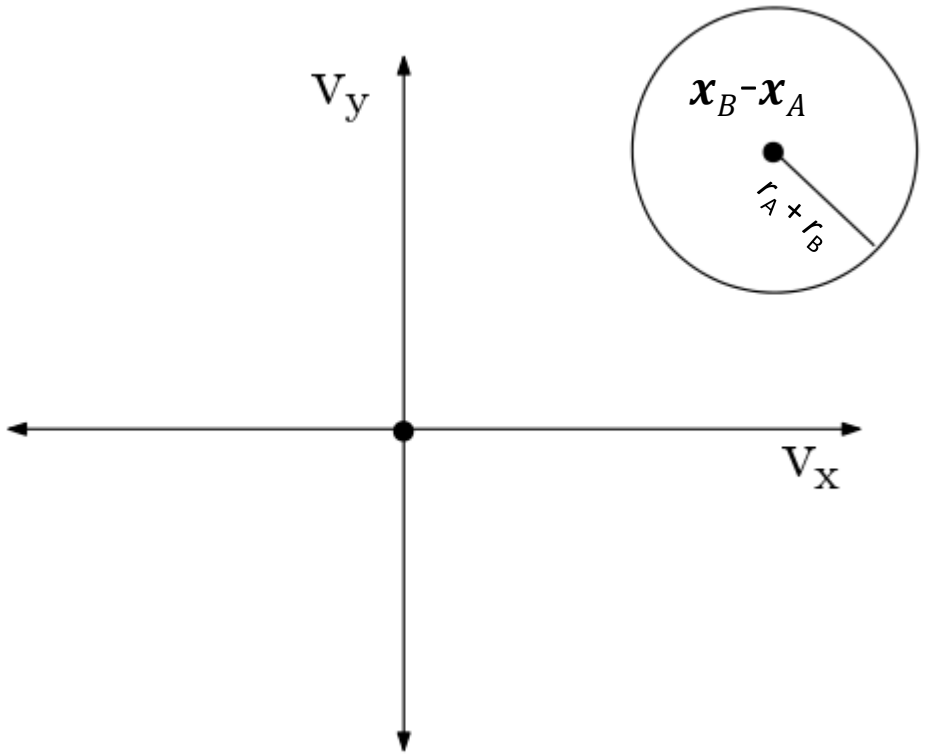
- Introduced by [Fiorini and Shiller ‘98]

  Given two agents, A and B, the velocity obstacle $VO_{A|B}$ (read: the velocity obstacle of A induced by B) is the set of all relative velocities that will lead to a collision between the two agents at some moment in time, $\tau >= 0$.

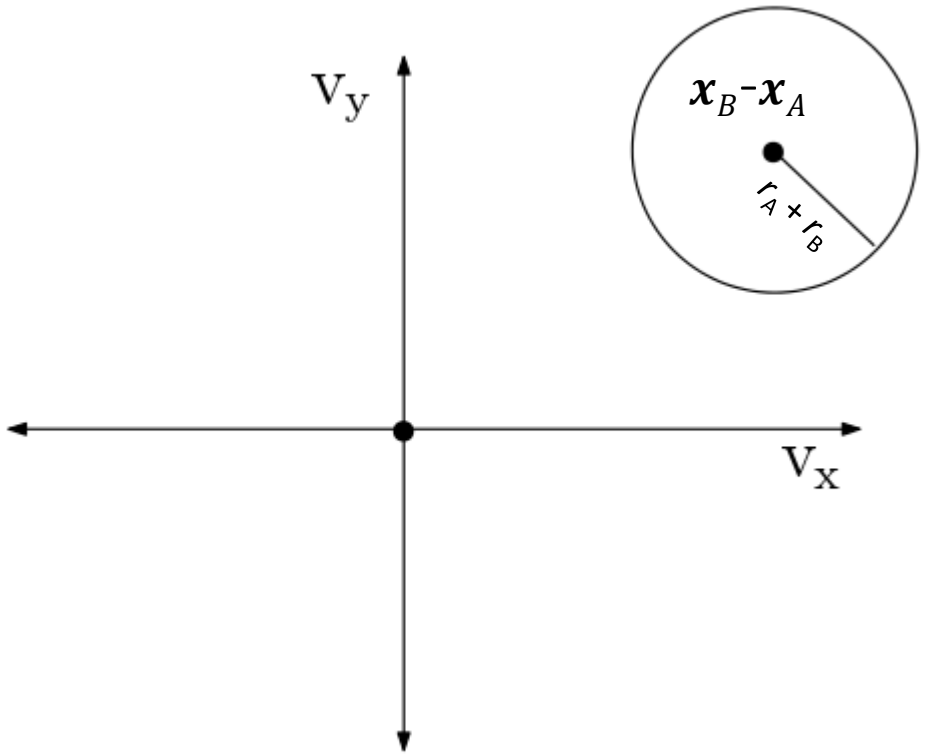
From Cartesian to velocity space



From Cartesian to velocity space

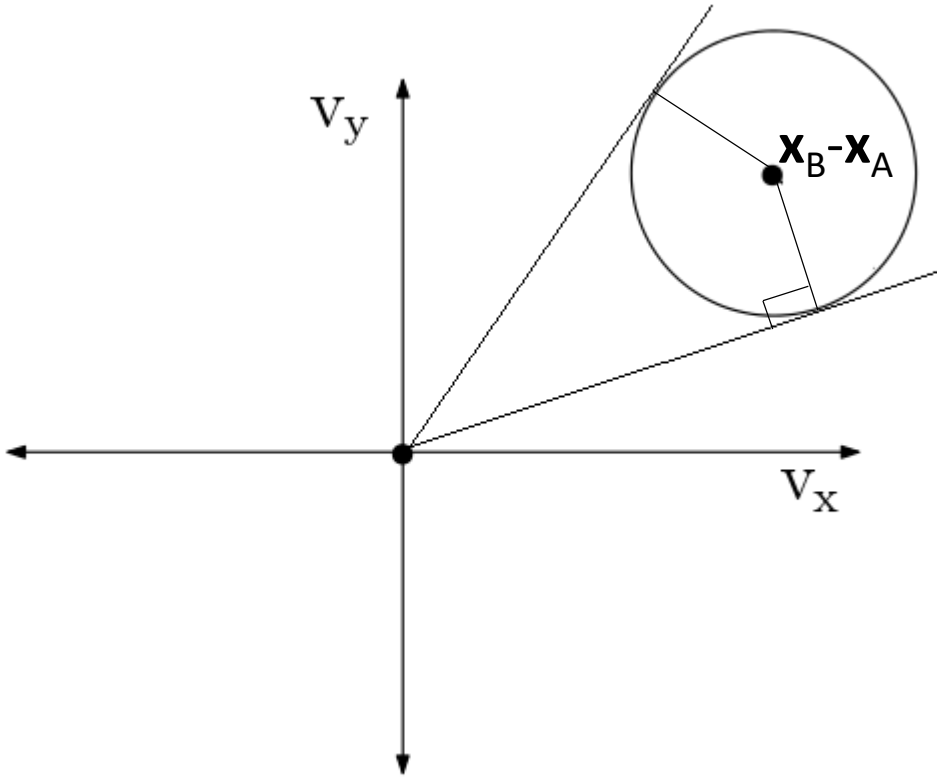


What is the VO?



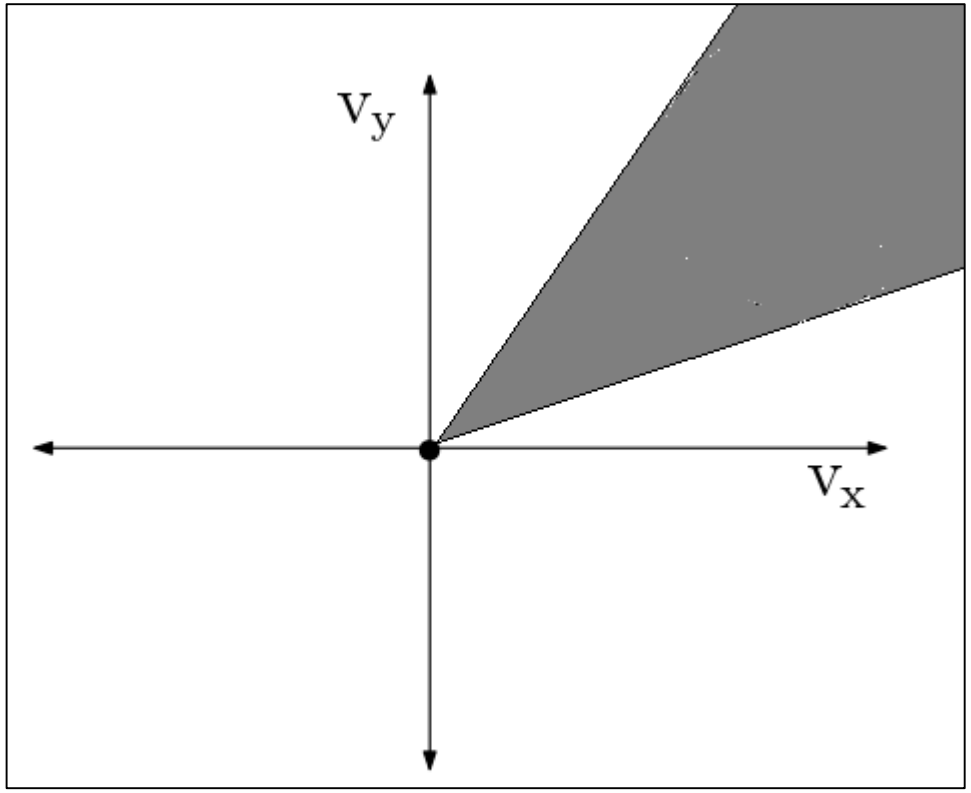
Geometric definition

- VO is a cone with its apex at the origin and its legs tangent to the disk of radius $r_A + r_B$ centered at $\boldsymbol{x}_B - \boldsymbol{x}_A$



Geometric definition

- VO is a cone with its apex at the origin and its legs tangent to the disk of radius $r_A + r_B$ centered at $\boldsymbol{x}_B - \boldsymbol{x}_A$



Revising the VO definition

- *Given two agents, A and B, the velocity obstacle $VO_{A|B}$ is the set of all relative velocities that will lead to a collision between the two agents at some moment in time, $\tau >= 0$.*
- *Assuming A and B are approximated as disks of radius r_A and r_B, respectively, the VO is defined as:*

$$VO_{A|B} = \{\mathbf{v} \,|\, \exists \tau \geq 0 : \tau \mathbf{v} \cap \in D(\mathbf{x}_B - \mathbf{x}_A, r_A + r_B) \neq \emptyset\}$$

where D($\boldsymbol{x}, r$) denote a disk centered at $\boldsymbol{x}$ having radius r

Velocity Obstacles

$$VO_{A|B} = \{\mathbf{v} \,|\, \exists\, \tau \geq 0 : \tau\mathbf{v} \cap \in D(\mathbf{x}_B - \mathbf{x}_A, r_A + r_B) \neq \emptyset\}$$

- This is basically a ray-disk intersection problem

$$\|\tau\boldsymbol{v} - (\boldsymbol{x}_B - \boldsymbol{x}_A)\| < r_B + r_A$$

- which is ….

Time to collision (τ)

- Formally, a collision exists if

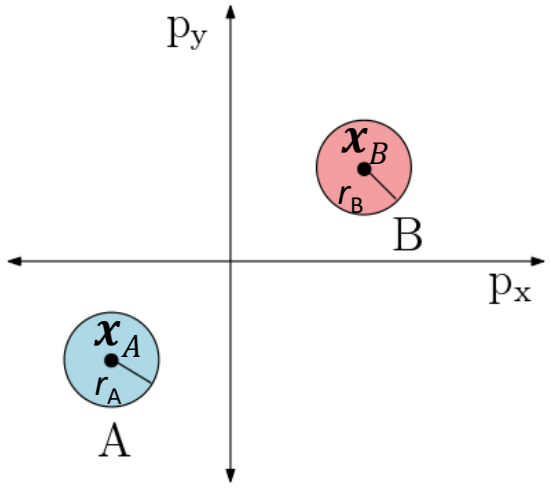
$$\|\boldsymbol{x} + \boldsymbol{v}\tau\| = r$$

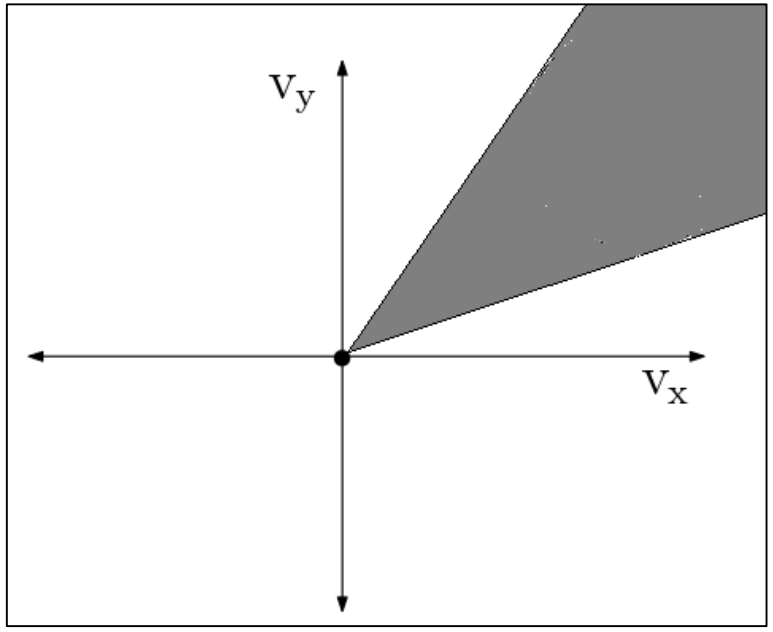
- Squaring and expanding, leads to the following quadratic equation for τ

$$(\boldsymbol{v} \cdot \boldsymbol{v})\tau^2 + 2(\boldsymbol{x} \cdot \boldsymbol{v})\tau + \boldsymbol{x} \cdot \boldsymbol{x} - r^2 = 0$$

Collision-free velocities

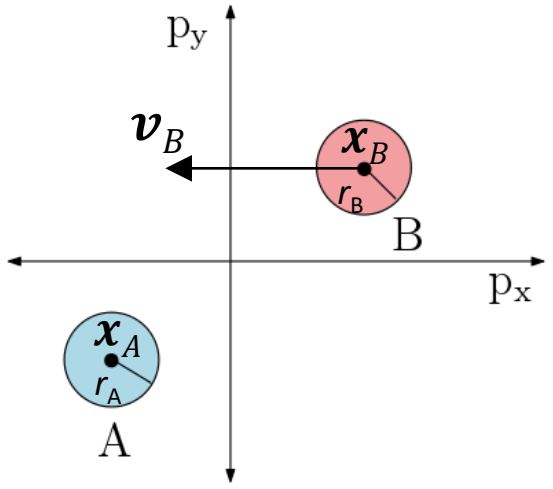
- Let $CFV_{A|B}$ denote the set of collision-free velocities for agent A induced by B
- What is $CFV_{A|B}$ if B stands still?

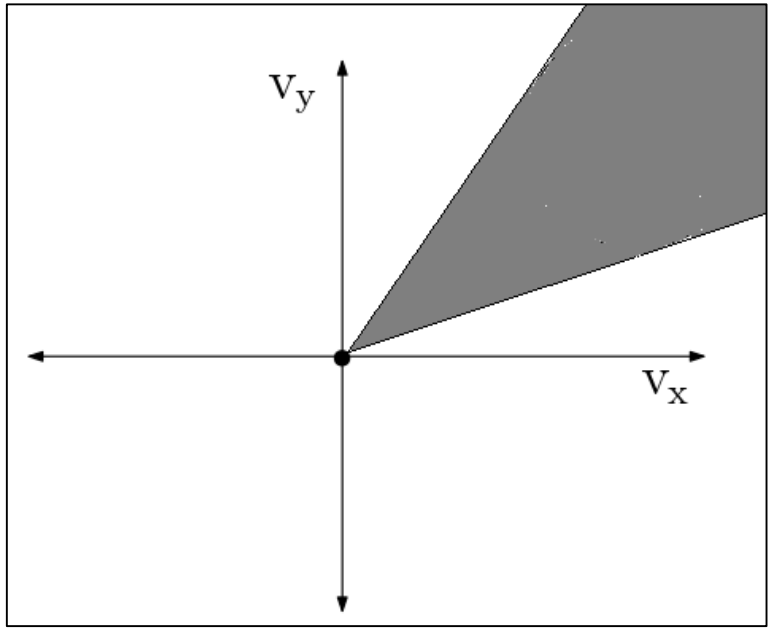




Collision-free velocities

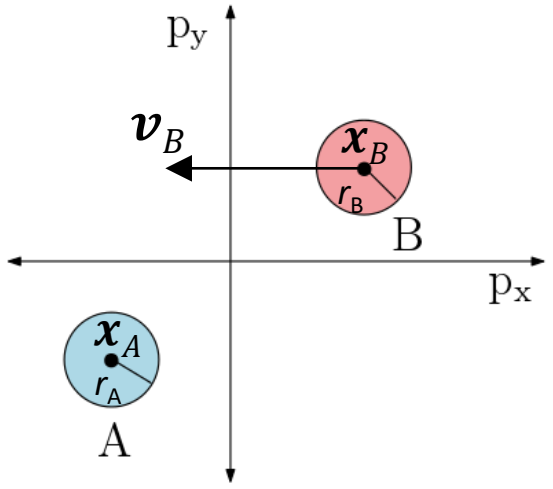
- What is $CFV_{A|B}$ if B is moving at velocity $\boldsymbol{v}_B$?

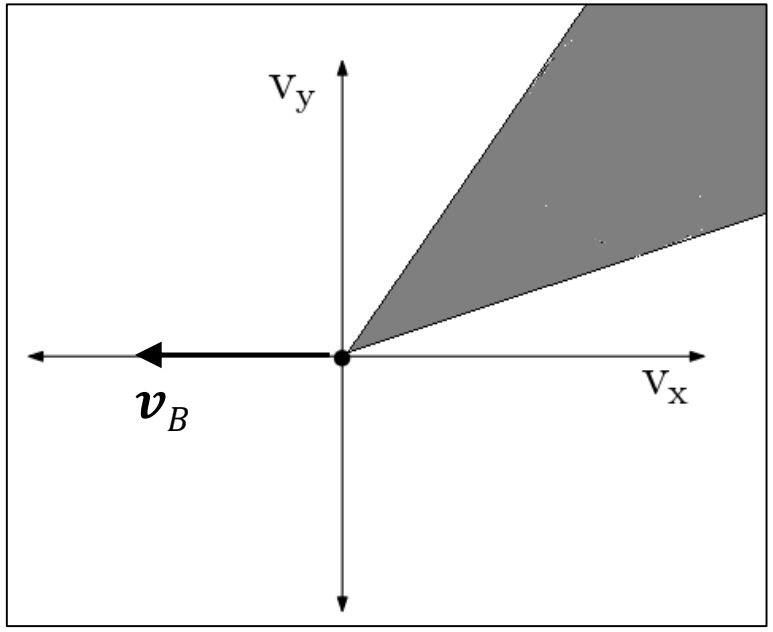




Collision-free velocities

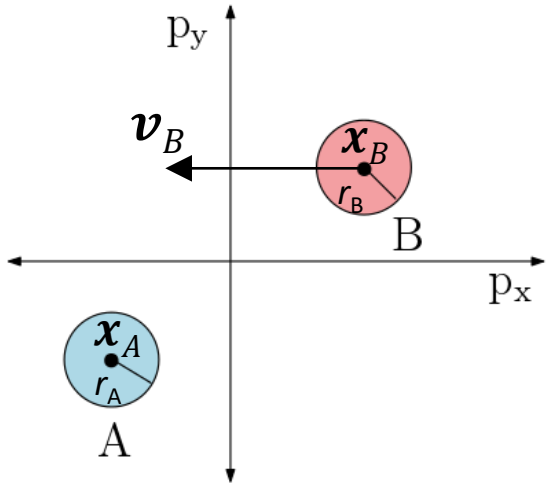
- What is $CFV_{A|B}$ if B is moving at velocity $\boldsymbol{v}_B$?

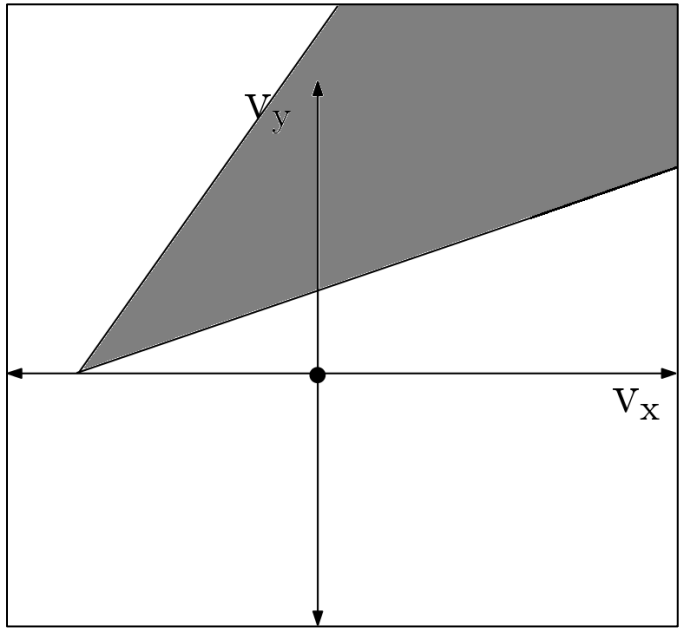




Collision-free velocities

- What is $CFV_{A|B}$ if B is moving at velocity $\boldsymbol{v}_B$?





Collision-free velocities

- *The set of collision-free velocities $CFV_{A|B}$ for agent A given an agent B is the set of all velocities that lie outside the $VO_{A|B}$ translated by B's velocity $\boldsymbol{v}_B$.*
- *Formally:*

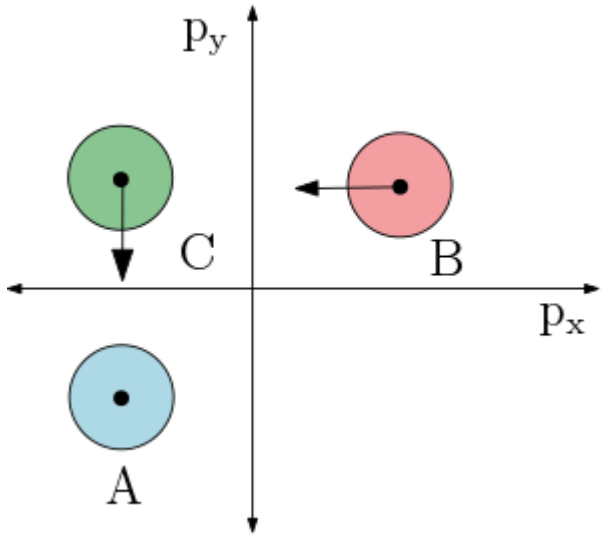
$$CFV_{A|B} = \{\mathbf{v} \,|\, \mathbf{v} \notin VO_{A|B} \oplus \mathbf{v}_B\}$$, *where* $\oplus$ *denotes the Minkowski sum*

- *Given two sets X and Y, the Minkowski sum is*

$$X \oplus Y = \{x + y \,|\, x \in X, y \in Y\}$$

Collision-free velocities

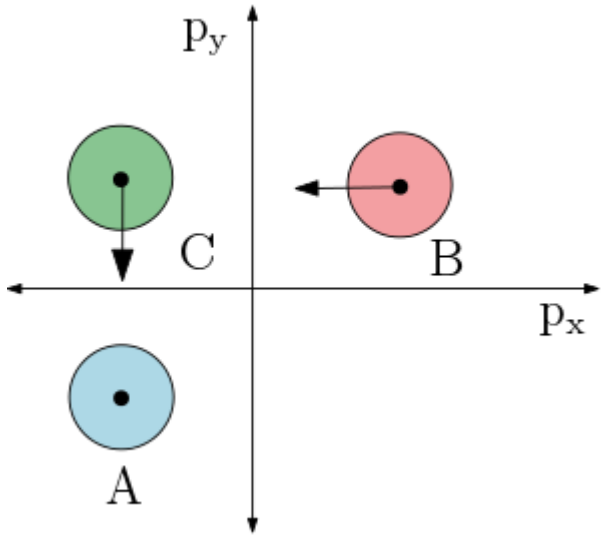
- What if agent A has to avoid multiple agents?

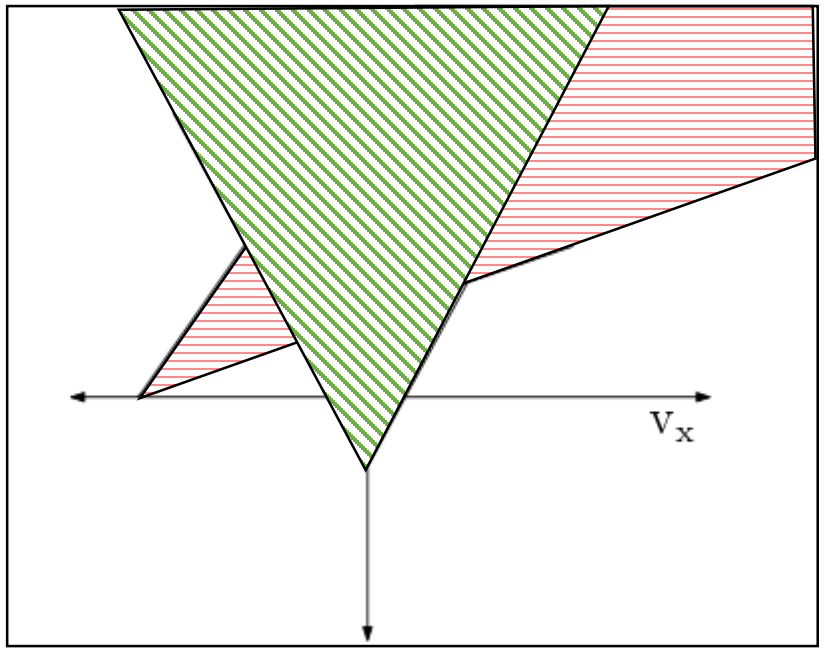


Collision-free velocities

- What if agent A has to avoid multiple agents?

$$CFV_A = \bigcap_{B \neq A} CFV_{A|B}$$





Admissible velocities

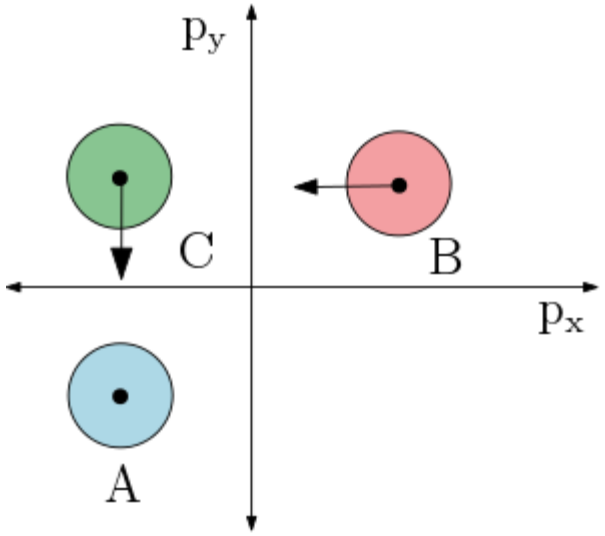
- At each time step, let AV_A denote the set of *admissible* velocities for agent *A*
- E.g., the agent is subject to a maximum speed $v^{\max}$ and a maximum acceleration $\alpha^{\max}$

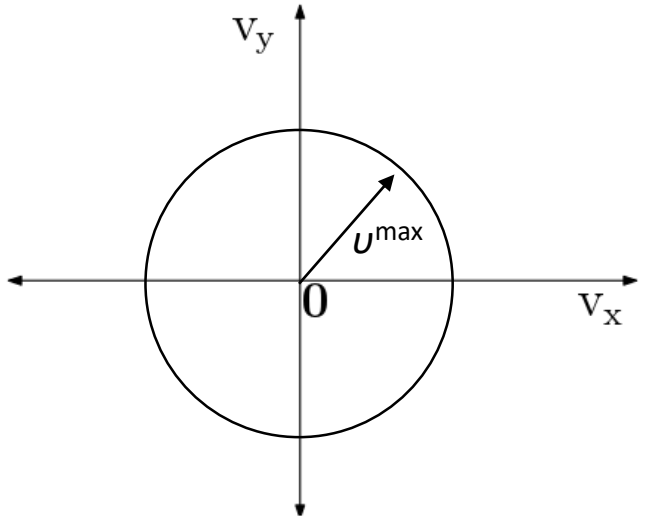
$$AV_A = \{\mathbf{v}' \mid \|\mathbf{v}'\| < v^{\max} \wedge \|\mathbf{v}' - \mathbf{v}\| < a^{\max}\Delta t\}$$

Admissible velocities

- Let's assume that A is subject to maximum speed $v^{\max}$, i.e.

$$AV_A = \{\boldsymbol{v}' \mid \|\boldsymbol{v}'\| \leq v^{max}\}$$

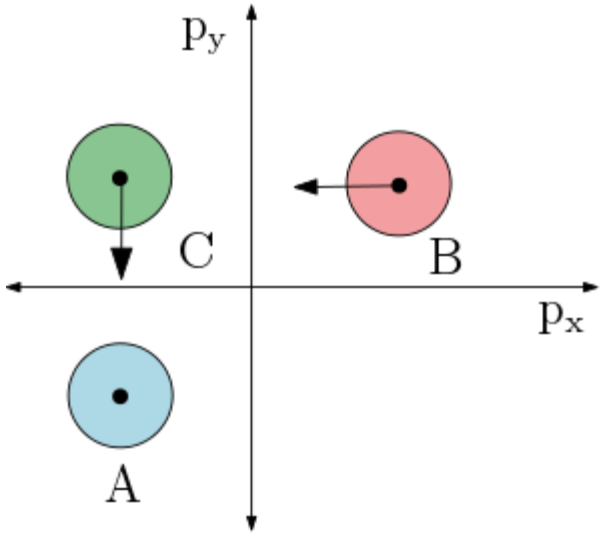


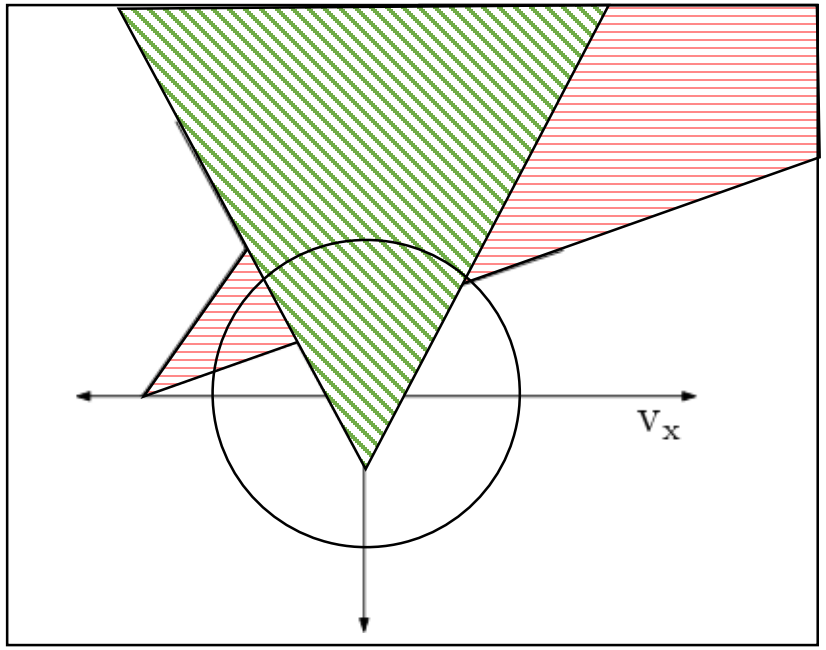


Admissible velocities

- Let's assume that A is subject to maximum speed $v^{\max}$, i.e.

$$AV_A = \{\boldsymbol{v}' \mid \|\boldsymbol{v}'\| \leq v^{max}\}$$



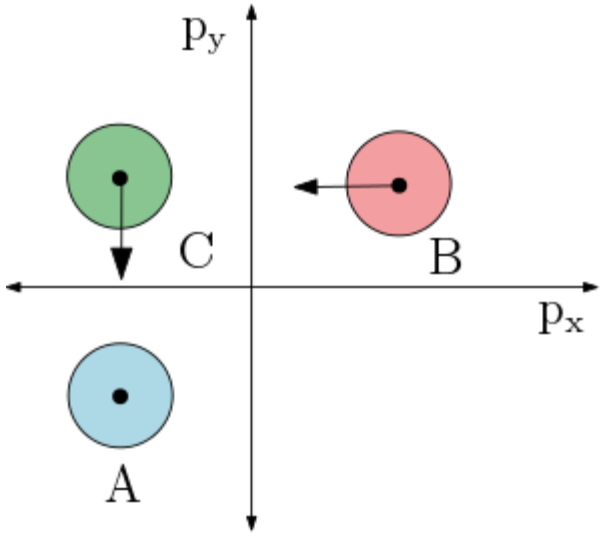


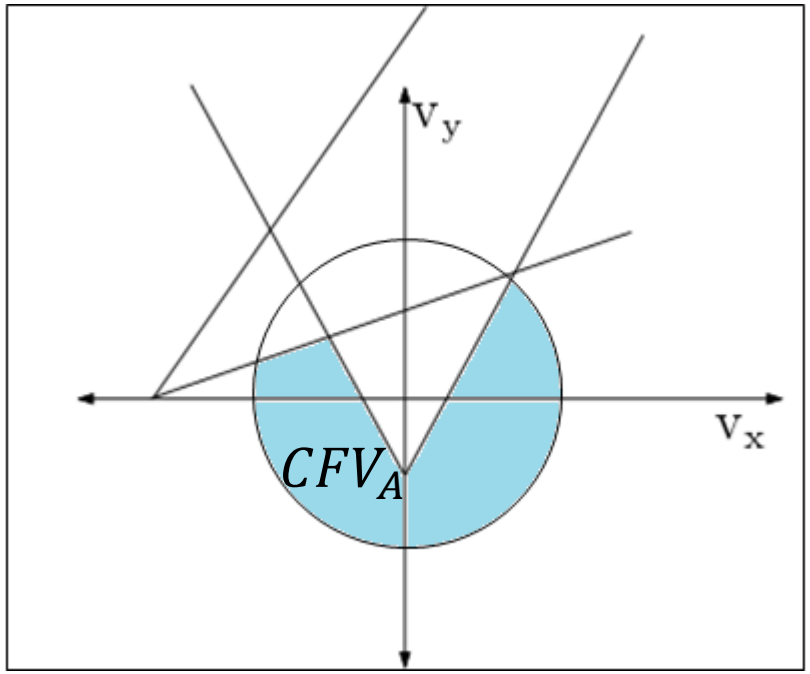
Collision-free velocities

- Let's assume that A is subject to maximum speed $v^{\max}$, i.e.

  $AV_A = \{\boldsymbol{v}' \mid \|\boldsymbol{v}'\| \leq v^{max}\}$

- Then the set of collision-free velocities becomes $CFV_A = \bigcap_{B \neq A} CFV_{A|B} \bigcap AV_A$





How to select a new velocity?

- Recall: We need to compute a *collision-free velocity that is as close as possible to the agent's goal velocity*
- Formulate this as an optimization problem

$$\mathbf{v}^n = \underset{\mathbf{v} \in CFV_A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}_g\|$$

- How to solve this?
  - Exactly

    See [Guy et al, 2009] for details

How to select a new velocity?

- Recall: We need to compute a *collision-free velocity that is as close as possible to the agent's goal velocity*
- Formulate this as an optimization problem

$$\mathbf{v}^n = \underset{\mathbf{v} \in CFV_A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}_g\|$$

- How to solve this?
  - Exactly

    See [Guy et al, 2009] for details
  - Approximately
    - Sampling [van den Berg et al., 2008]
    - Linear programming [van den Berg et al., 2011]

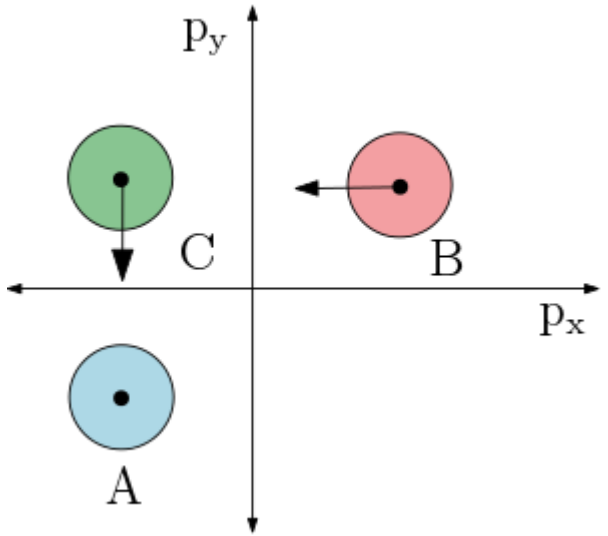
Velocity-Based Navigation

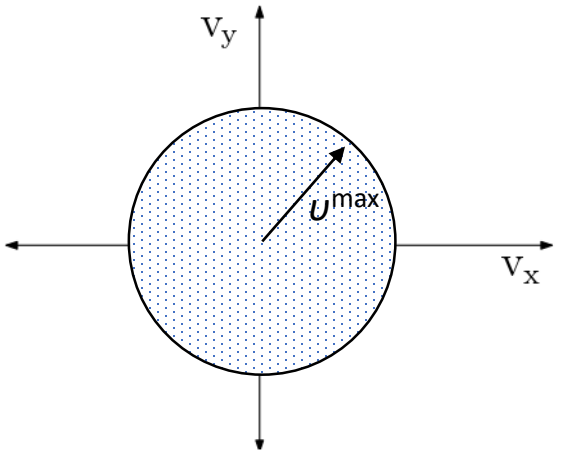
Sampling-Based VOs

Approximate solution via sampling

- To determine a new velocity $\mathbf{v}^n$ for agent A
  - Uniformly sample N candidate velocities from AV_A
  - *S*elect an optimal one according to a specific cost function *f:* $\mathbf{v}^n = \underset{\mathbf{v}_{i=1\ldots N}^{cand} \in AV_A}{\operatorname{argmin}} f$

What is a good f?



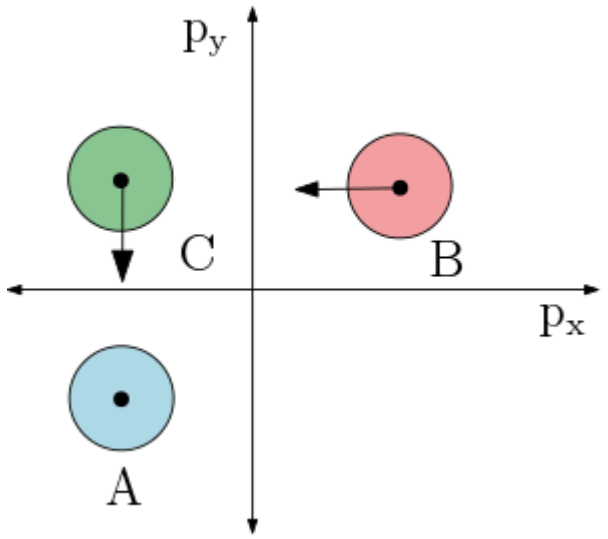


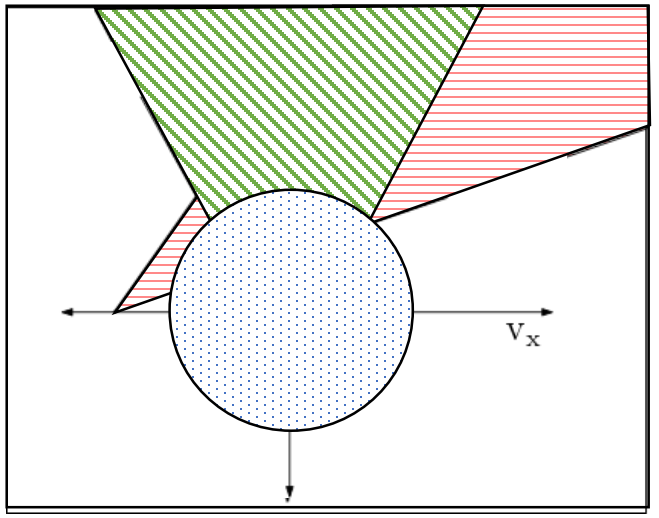
Approximate solution via sampling

- Recall: The optimization problem to solve
  - Compute a collision-free velocity that is as close as possible to the agent's $\boldsymbol{v}_g$

$$\mathbf{v}^n = \underset{\mathbf{v} \in CFV_A}{\operatorname{argmin}} \|\mathbf{v} - \mathbf{v}_g\|$$

- But we cannot directly sample CFV_A, only AV_A





Approximate solution via sampling

- Sample admissible velocities and select

$$\mathbf{v}^n = \underset{\mathbf{v}_{i=1..N}^{cand} \in AV}{\arg\min} \left\{ \frac{\gamma}{tc} + ||\boldsymbol{v}_i^{cand} - \boldsymbol{v}_g|| \right\}$$

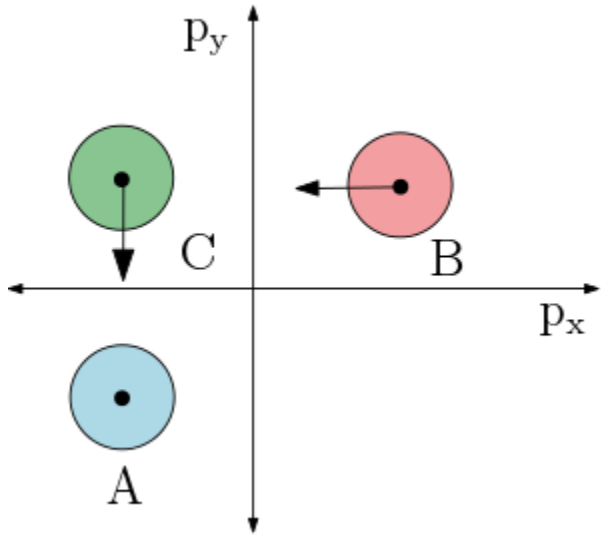
  - Here, tc denotes the minimum time that will take for the agent to collide with any of the other agents assuming that it selects a velocity $\boldsymbol{v}_i^{cand}$
  - The constant γ controls the relative importance of the two cost terms
  - Other terms can be added or variants of the above cost function

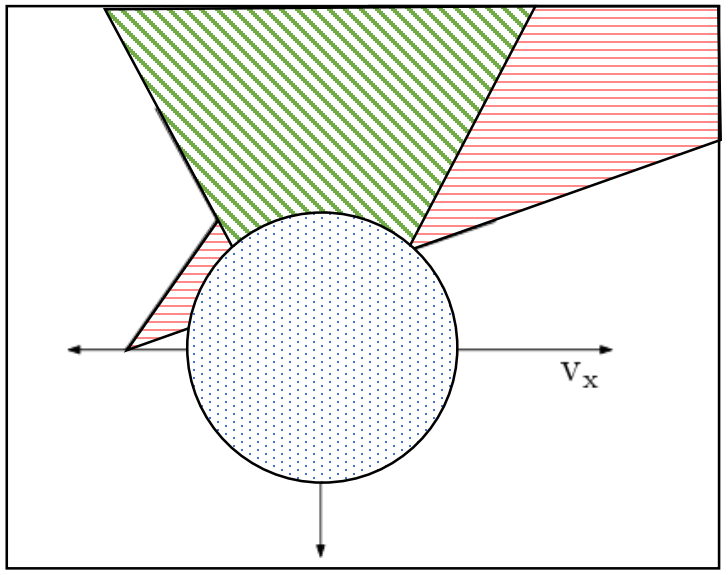
Interpretation: tradeoff between safety and closeness to the goal velocity

Approximate solution via sampling

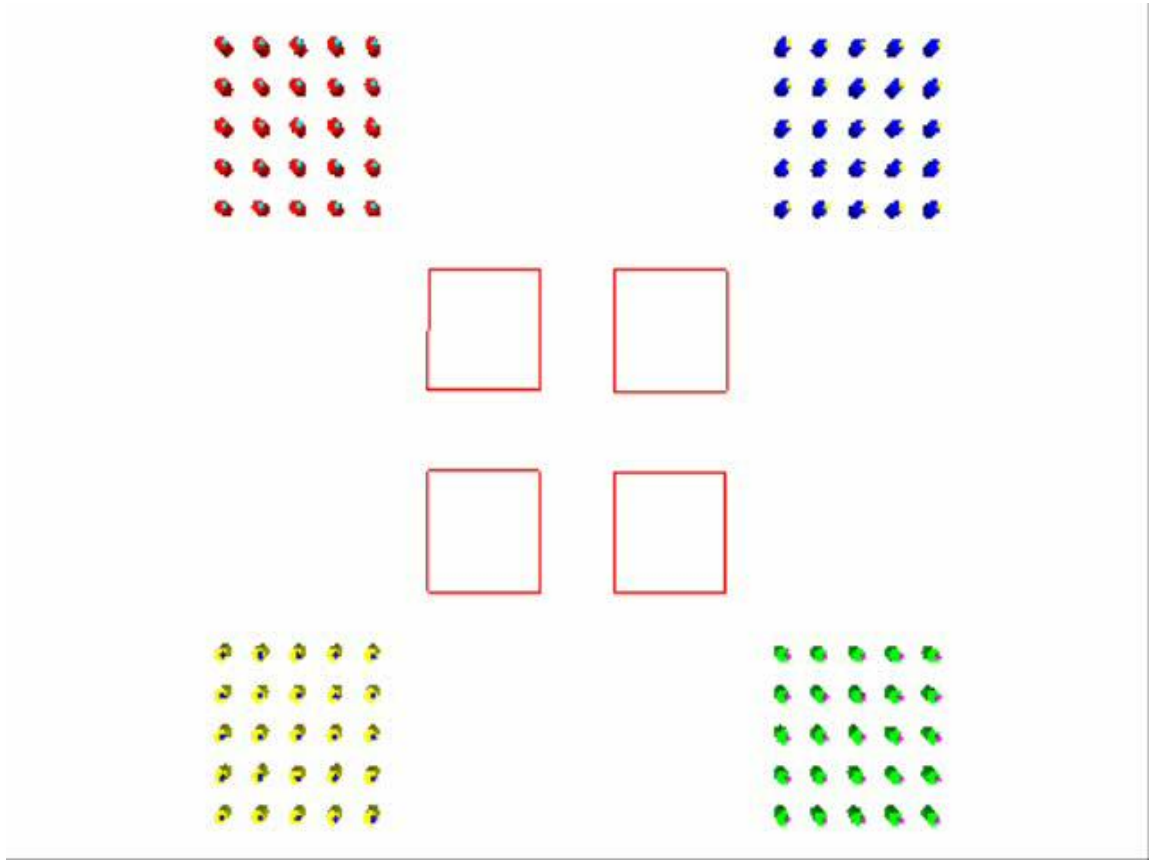
- Sample admissible velocities and select

$$\mathbf{v}^n = \underset{\mathbf{v}_{i=1..N}^{cand} \in AV}{\arg\min} \left\{ \frac{\gamma}{tc} + \|\mathbf{v}_i^{cand} - \mathbf{v}_g\| \right\}$$



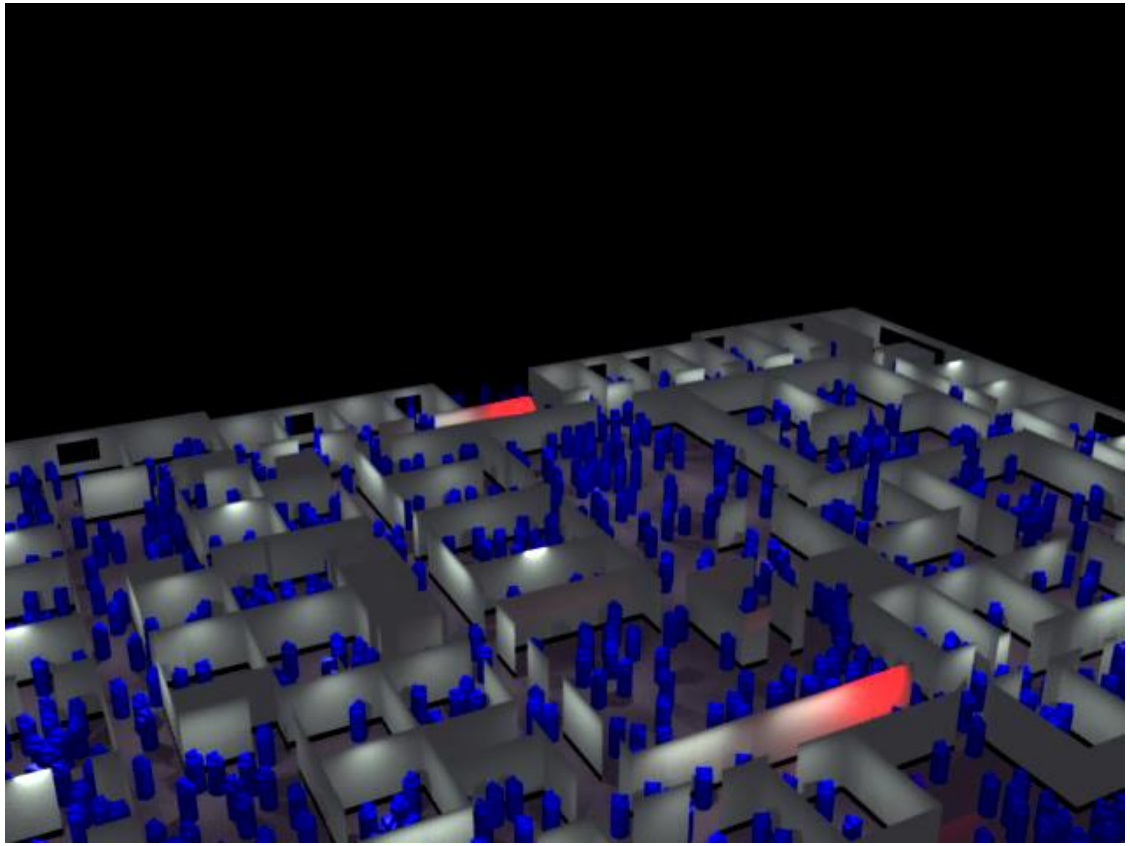


Some results



http://gamma.cs.unc.edu/RVO

Some results



http://gamma.cs.unc.edu/RVO

Assignment 1

- Implement the sampling-based approach
- Submit using the Canvas system
- Can work in pairs (please let us know in advance)
- Sample code in Python 3.6

Python installation

- If you do not have Python 3.6, I suggest you install Conda
  - Easy way to manage different environments, each with its own Python versions and dependencies
  - Can be installed through Anaconda/Miniconda
- Once Conda is installed, create a virtual environment for the class

```
conda create --name cpsc8810 python=3.6
```

- See here for more details on how to set up your machine for the assignments

Basic Requirements

Your program should work as follows. It should first load a simulation scenario and set the simulation time step, Δt, to 0.1 s by modifying the corresponding parameters in `simulator.py`. Then, for each simulation step, for each of the n agents you need to compute a new velocity by modifying the `computeNewVelocity` function in `agent.py` as follows:

1. Determine all neighbors that are less than d_H m away from the agent. A typical value for the sensing radius, d_H, to consider is 5-10 m.

2. Uniformly sample N velocities from the admissible velocity space, AV, of the agent. The agent is subject to a maximum speed, so AV is a disc centered at (0,0) having radius equal to the agent's maximum speed. Try different number of candidate velocities between 100 and 1000 in order to find a balance between acceptable navigation behavior and computational performance.

3. Evaluate the fitness of each of the N candidate velocities using the following cost function:

$$\alpha\|\mathbf{v}^{vcand} - \mathbf{v}^{goal}\| + \beta\|\mathbf{v}^{vcand} - \mathbf{v}\| + \frac{\gamma}{tc}.$$

   This is a modified version of the function covered in class that accounts for the distance between the candidate and goal velocity, the distance between the candidate and the agent's current velocity, and the risk of the agent to collide with nearby neighbors. Here, α, β, γ are scaling constants that control the relative importance of the three cost terms. You should experiment with different scaling values starting with $\alpha = 1$, $\beta = 1$, and $\gamma = 2$. Regarding the collision cost term, tc denotes the minimum time that it will take for the agent to collide with any of its sensed neighbors if it moves at velocity $\mathbf{v}^{vcand}$. To compute tc, you must first compute the time to collision, τ, between the agent and each of its neighbors as discussed in lecture 2 (though, you should use the candidate rather than the agent's actual velocity).

4. Select as new velocity for the agent the candidate velocity that has the minimum cost. The code in the `update` function will use the selected velocity to update the velocity and position of the agent. It also computes the goal velocity of the agent for the next simulation step by simply taking the unit vector pointing from the current position of the agent to its goal position scaled by the agent's preferred speed.

5. In case an agent is close to its goal (e.g., less than 1 m), you should stop computing new velocities and updating the agent's position, as well as considering this agent for the nearest neighbor computations of the other agents.