Collision Cones, Velocity Obstacles, Time Scaling and Trajectory Optimisation

Introduction

Meet Gera, Jun 10 2025

Overview

Learning Objectives

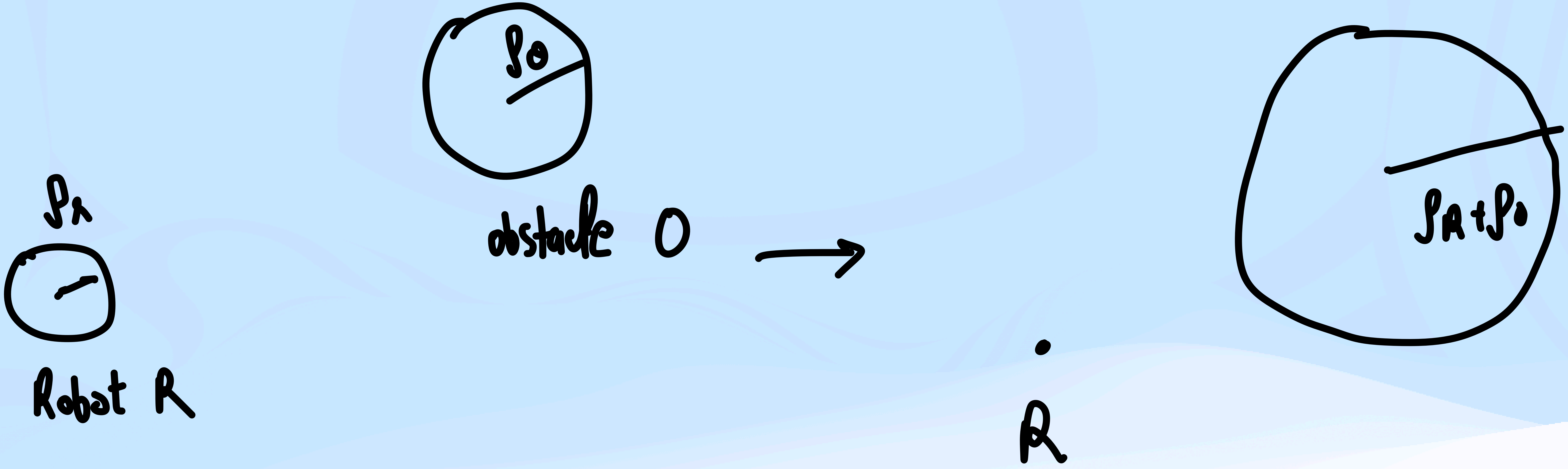
- By end of lecture, students will understand :
  - Collision cones and their geometric interpretation
  - Velocity Obstacles for collision avoidance
  - Constraint handling in motion planning
  - Time scaling techniques for trajectory execution
  - Trajectory Optimisation principles and methods

Collision Cones and Velocity Obstacles

What is a collision cone ?

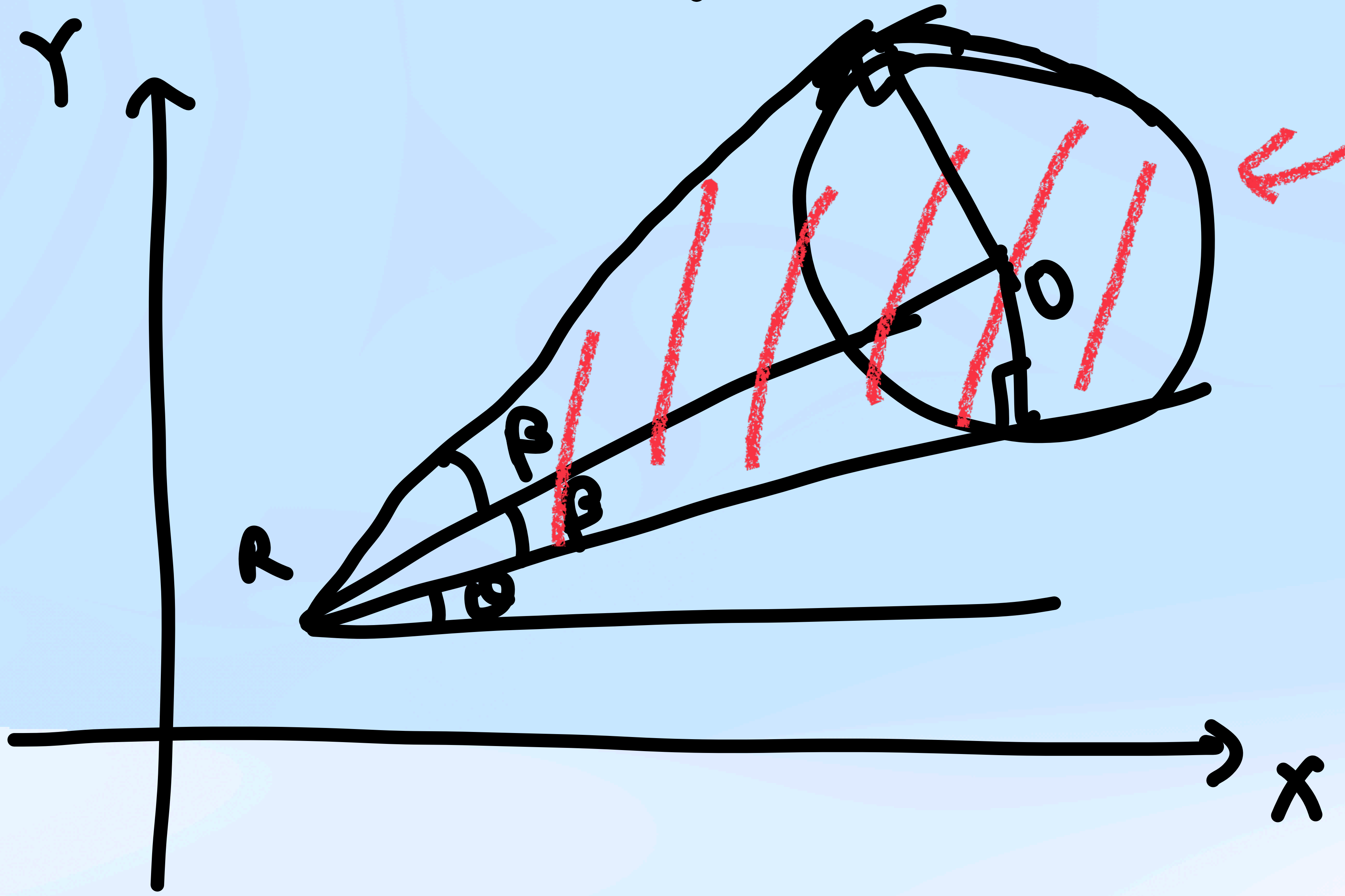
- It is a set {}
- Set of all relative velocities that would lead to collision
- Over a given time horizon

- Assumptions : Constant velocities over time.



We can collapse the size of the robot and inflate the size of all the obstacles present in the environment.

$$CC(O) = \{ v \mid \exists \lambda,\ q + \lambda v \cap R(O) \neq \phi$$



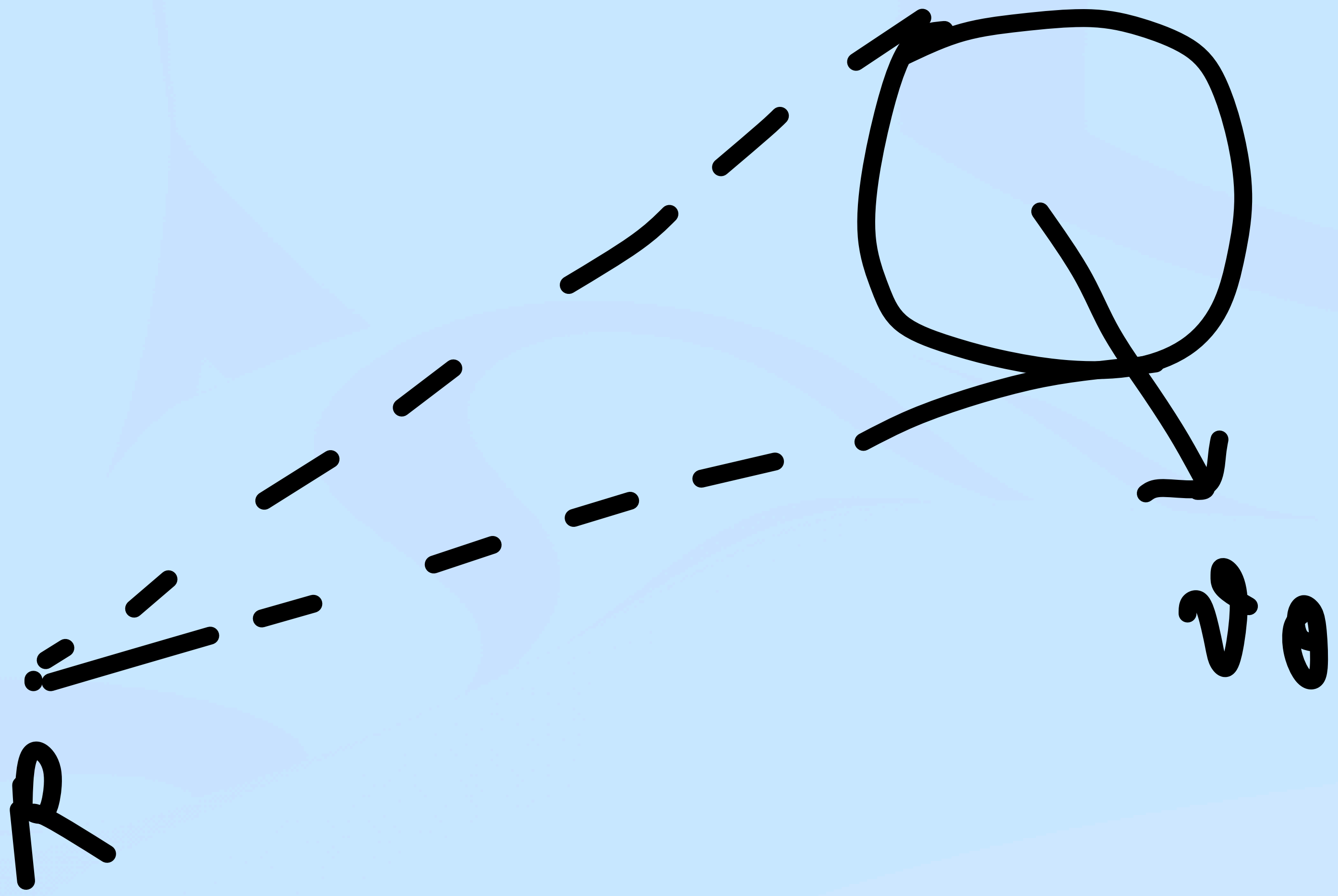
$$\sin\beta = \frac{\rho_o + \rho_R}{d(R,O)}$$

$$\theta = \operatorname{atan2}(O_y - R_y,\ O_x - R_x)$$

$$R(O) = \{ q : d(q,O) < \rho_o + \rho_R \}$$

occupancy of obstacle O.

What if the obstacle was moving ?



$$VO(O) \ominus v_O = CC(O)$$

$$VO(O) = CC(O) \oplus v_O$$

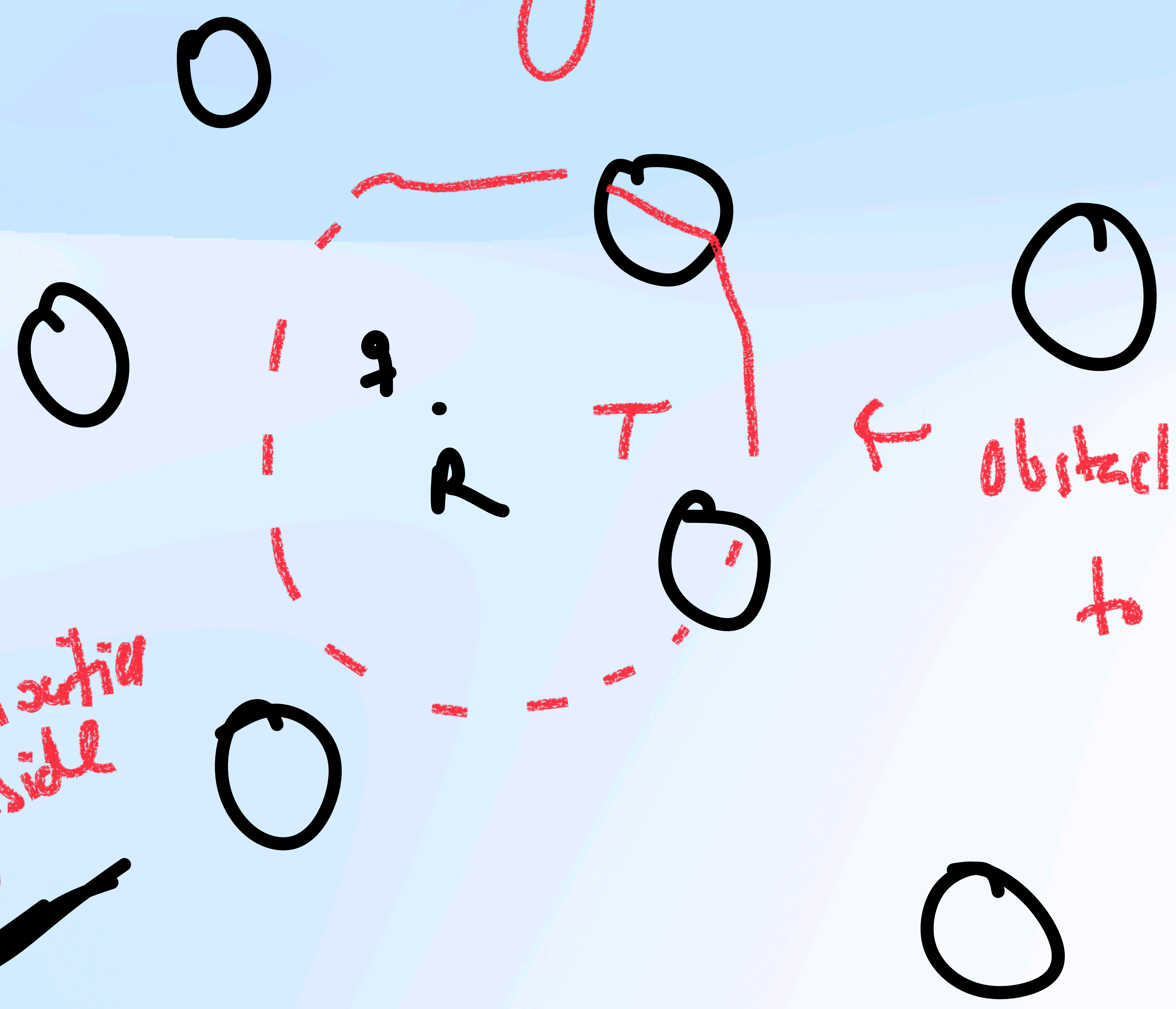
Velocity obstacle

minkowski addition

Set of all velocity (absolute) of R that would lead to collision with obstacle (O).

$V_O = \bigcup_{o} VO(o)$ ← won't work → given enough obstacles no direction will be viable.

→ Decision Making Horizon based on time



← obstacles likely to collide in time T

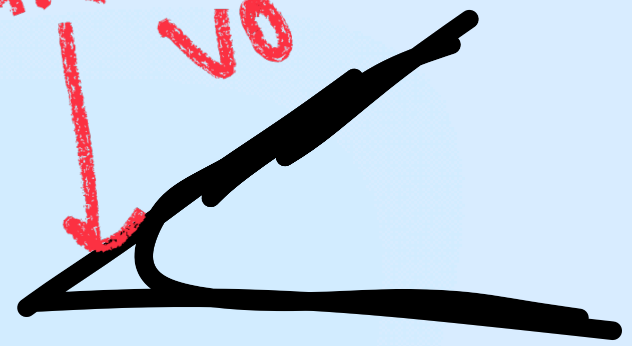
$$T \geq \frac{d(q, q_o) - (\rho_o + \rho_R)}{\|v\|}$$

+

$$\underline{\underline{VO}} = \{ v + v_o : \exists \lambda, \; q + \lambda v \cap R(o) \neq \varnothing, \; \|v\| \leq \frac{d(q, q_o)}{T}$$

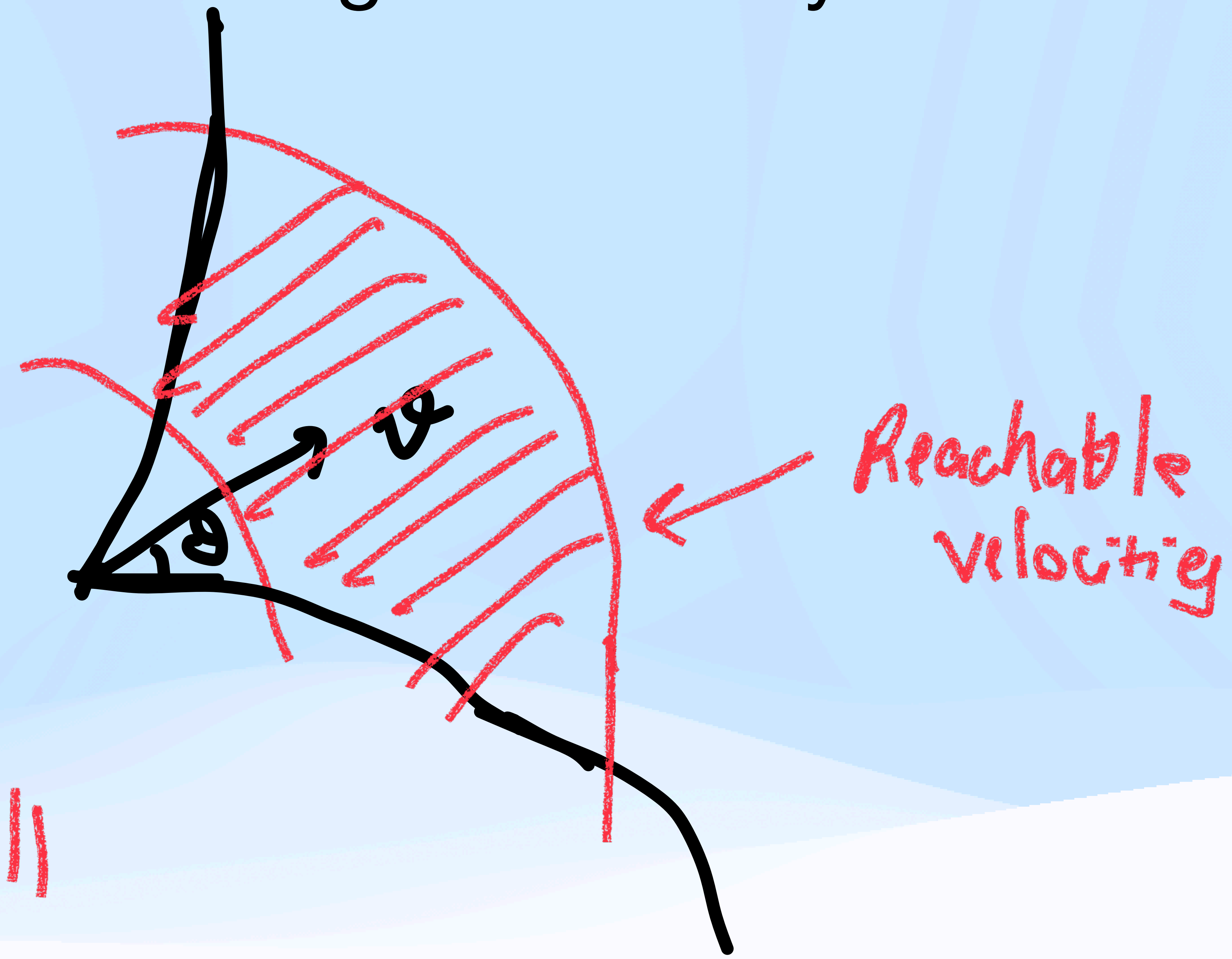
↑ final velocity obstacle.

lower velocities are outside VO



Constraints

For every robot there is a certain range of velocity and acceleration available.



$$0 \le \|v\| \le v_{max}$$

$$a_{min} \le acc \le a_{max}$$

$$\|v(t) + a_{min}\Delta t\| \le \|v(t+\Delta t)\| \le \|v(t) + a_{max}\Delta t\|$$

$$\theta + \omega_{min}\Delta t \le \theta(t+\Delta t) \le \theta + \omega_{max}\Delta t$$

RAV = RV − VO

reachable avoiding (collision) velocities.

set diff.

Time Scaling

What is the difference between Path And Trajectory ?

What is Path ?

Path : A geometric curve in space, independent of time.

1. Describes where the robot goes
2. Parameterised by arc length q(s) where s lies in [0,1]
3. Examples : A circular arc from point A to B

What is Trajectory ?

Trajectory : A path with timing information

1. Describes where and when the robot goes
2. Parameterised by time $q(t)$ where t lies in $[0,T]$
3. Example : Following the circular arc in 3 seconds

What is the time scaling function ?

The time scaling function s(t) connects path parameter to real time :

$q(t) = q(s(t))$

1. Essentially it tells how fast we move along the path
2. When we speed up or slow down
3. Total time to complete the path

$s(t) : [0,T] \longrightarrow [0,1]$

essentially it tells what point on path the robot is at time t.

Velocity and Acceleration Relationships :

Tangent to path

$$\text{velocity} = \dot{\bar{q}}(t) = \frac{d\bar{q}}{dt} = \frac{d\bar{q}}{ds} \cdot \frac{ds}{dt} = \bar{q}'(s) \cdot \dot{s}(t)$$

speed

$$\text{Acceleration} = \ddot{\bar{q}}(t) = \bar{q}''(s)\,\dot{s}^2(t) + \bar{q}'(s)\,\ddot{s}(t)$$

curvature

acceleration along path.

Examples :

Problem 1 :

Robot moves along straight line (0,0) to (10,0) in 2D plane.

$q(s) = \begin{bmatrix} 10s \\ 0 \end{bmatrix} \quad s \in [0,1]$

$q'(s) = \begin{bmatrix} 10 \\ 0 \end{bmatrix}$

$q''(s) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Scenario A

constant speed

speed

Want $|v| = 2m/s$

$\Rightarrow \quad t \in [0, 5]$

$\Rightarrow \quad s(t) = \dfrac{t}{5}$

$\Rightarrow \quad \bar{q}(t) = \begin{bmatrix} 2t \\ 0 \end{bmatrix},$

velocity

$\dot{q}(t) = q'(s) \cdot \dot{s} = \begin{bmatrix} 10 \\ 0 \end{bmatrix}(0.2) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}$

Scenerio B : Accelaration

Start from rest, end at rest

Constraints:

max. velocity = 4 m/s

max. acc. = 2 m/s^2

$$S(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

$$\dot{s}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

$s(0) = 0$, $\dot{s}(0) = 0$

$s(T) = 1$ $\dot{s}(T) = 0$

$\Rightarrow a_0 = 0, a_1 = 0, a_2 = \frac{2}{T^2}, a_3 = -\frac{2}{T^3}$

To satisfy n constraints a polynomial of n degree is required.

$\boxed{\text{Higher order poly. can be used.}}$

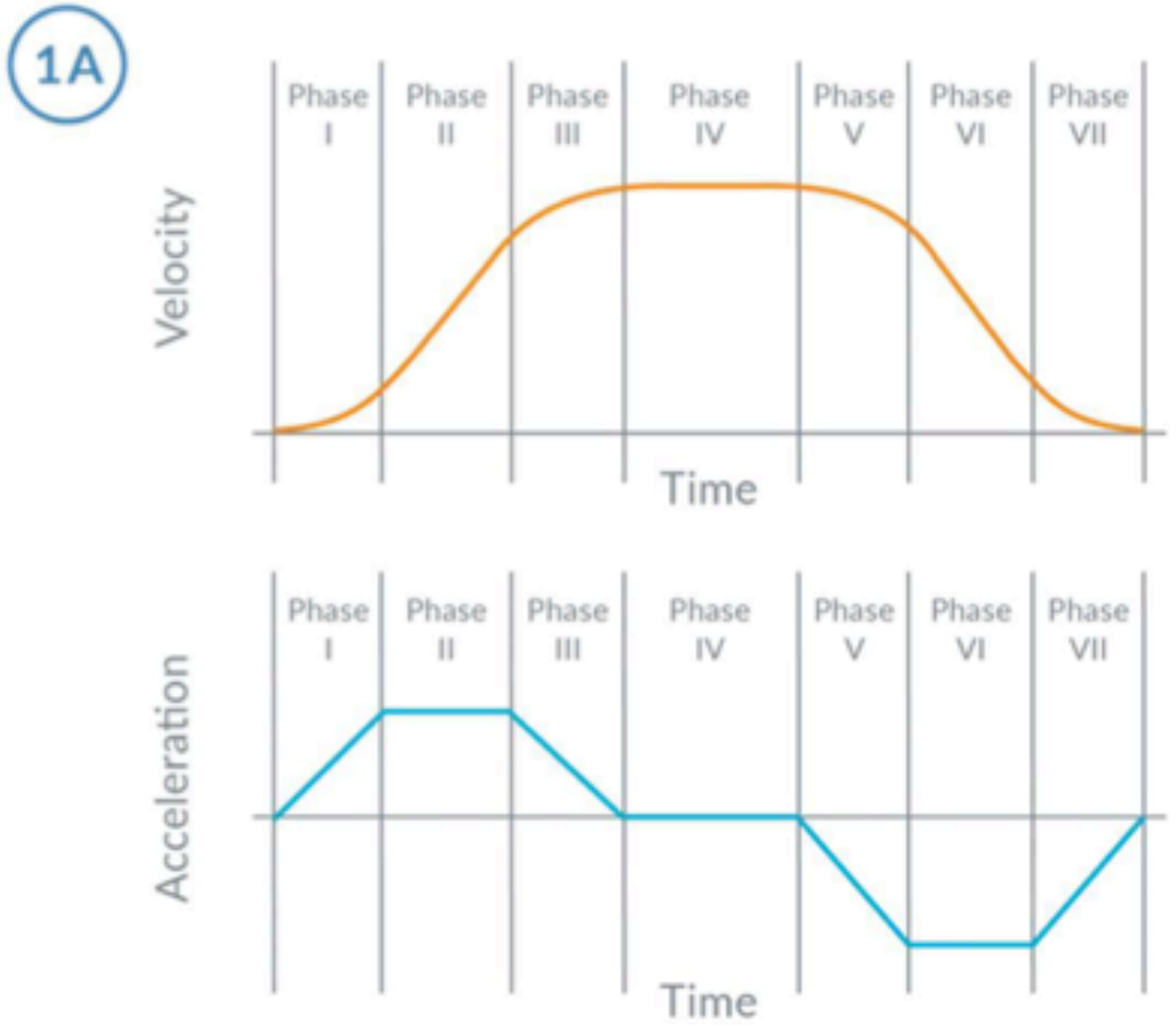
$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 1 & t_0 & t_0^2 & t_0^3 \\ 0 & 1 & 2t_0 & 3t_0^2 \\ 1 & t_f & t_f^2 & t_f^3 \\ 0 & 1 & 2t_f & 3t_f^2 \end{bmatrix} \begin{bmatrix} s(t_0) \\ \dot{s}(t_0) \\ s(t_f) \\ \dot{s}(t_f) \end{bmatrix}$$

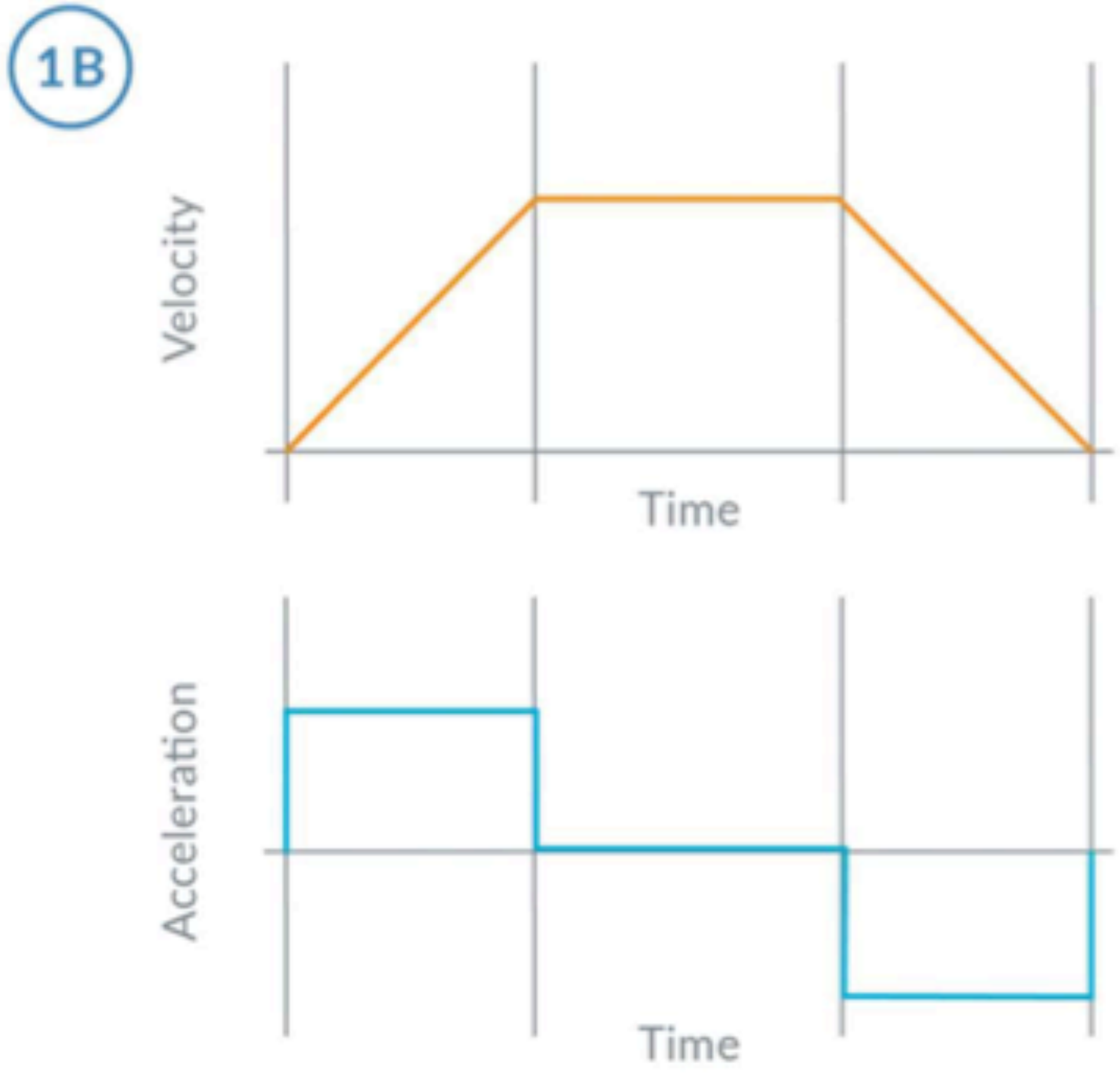
$$a = M s$$

$\mathrm{Det}(M) = (t_f - t_0)^4 \rightarrow$ unique sol^n as long as $t_f \neq t_0$

S-curve profile and trapezoidal profile

- The rapid acceleration induces powerful vibrations.
- Increased settling time and reduced accuracy.





Advanced Considerations :

1. Jerk Constraints
    1. For Smoother motion limit rate of acceleration change
2. Task Space Constraints
    a. When operating in cartesian space, joint velocities depend on robot config.
    b. Time scaling must account for varying Jacobian along the path.

In Essence :

The key insight is separating geometric planning (where to go) from temporal planning (when to go), allowing independent optimization of each aspect while ensuring the final trajectory respects all physical constraints.

What is Trajectory Optimization?

Trajectory optimization answers:
"What is the best way for my robot to move from A to B?"

Compared to:

- **Path Planning:** Only geometry
- **Time Scaling:** Fixed path

Trajectory Optimization determines:

- Where to go (spatial trajectory)
- When to be there (temporal profile)
- How to get there (control inputs)

Mathematical Framework

Objective:

$$\min_{\vec{x}(\cdot),\vec{u}(\cdot)} J = \underbrace{\phi(\vec{x}(T))}_{\text{Terminal Cost}} + \underbrace{\int_0^T L(\vec{x}(t),\vec{u}(t))dt}_{\text{Running Cost}}$$

Subject to:

$$\begin{aligned} \dot{\vec{x}}(t) &= f(\vec{x}(t),\vec{u}(t)) \\ g(\vec{x}(t),\vec{u}(t)) &\leq 0 \\ \psi(\vec{x}(0),\vec{x}(T)) &= 0 \end{aligned}$$

Analogical explanation for Objective function

The Big Picture Analogy

Imagine you're planning a road trip from your home to a destination. You want to minimize your total "cost"—which might include fuel expenses, tolls, and time. The optimal control framework breaks this down into:

The Objective Function (What You're Minimizing)

$$J = \underbrace{\phi(\vec{x}(T))}_{\text{Terminal Cost}} + \underbrace{\int_0^T L(\vec{x}(t), \vec{u}(t))\, dt}_{\text{Running Cost}}$$

Terminal Cost $\phi(\vec{x}(T))$: Think of this as a "parking fee" you pay when you arrive at your destination. It depends on where exactly you end up at time T. Maybe you prefer to arrive at a specific location, and there's a penalty for being far from that target.

Running Cost $\int_0^T L(\vec{x}(t), \vec{u}(t))\, dt$: This is like your ongoing expenses during the trip—fuel consumption, tolls, wear-and-tear on your car. The function L represents the instantaneous cost rate at each moment, which depends on both where you are ($\vec{x}(t)$) and what you're doing ($\vec{u}(t)$, like how hard you're pressing the gas pedal).

The Constraints (Rules You Must Follow)

System Dynamics:

$$\dot{\vec{x}}(t) = f(\vec{x}(t), \vec{u}(t))$$

This is like the "physics" of your car. Your state $\vec{x}(t)$ might include your position, velocity, and fuel level. Your control $\vec{u}(t)$ might be your steering angle and throttle position. The equation says: "Given where you are now and what you're doing, here's how your state will change in the next instant."

Example: If you press the gas pedal (control), your velocity increases (state change). If you're moving forward, your position changes accordingly.

Path Constraints:

$$g(\vec{x}(t), \vec{u}(t)) \leq 0$$

These are ongoing restrictions throughout your journey. Think of them as:

- Speed limits: "Your velocity must not exceed 70 mph"
- Fuel constraints: "Your fuel level must stay positive"
- Physical limits: "Your steering angle can't exceed the car's maximum turn radius"
- Safety margins: "You must stay a certain distance from other vehicles"

Boundary Conditions:

$$\psi(\vec{x}(0), \vec{x}(T)) = 0$$

These connect your starting and ending conditions. Examples:

- "You must start at your home and end at your destination"
- "You must start with a full tank and end with at least 10% fuel remaining"
- "Your final velocity should be zero (you need to stop)"

The Mathematical Challenge

The beauty and difficulty of this problem is that you're not just optimizing at one point in time—you're finding the best possible trajectory (path through time) for both your system state $\vec{x}(t)$ and your control actions $\vec{u}(t)$. Every decision you make now affects your future options, creating a complex web of trade-offs.

The solution typically involves advanced techniques like:

- Calculus of Variations
- Pontryagin's Maximum Principle
- Dynamic Programming

These methods systematically explore this trade-off space to find the optimal balance between immediate costs and future consequences.

Examples

Example 1: Point Mass Robot - Minimum Time Problem

Problem Setup

A point mass robot needs to move from $(0, 0)$ to $(10, 5)$ in minimum time.

State: $\vec{x} = [x, y, \dot{x}, \dot{y}]^T$
Control: $\vec{u} = [u_x, u_y]^T$
Dynamics:

$$\dot{\vec{x}} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ u_x \\ u_y \end{bmatrix}$$

Constraints:

- Control limits: $|\vec{u}| \leq u_{\max} = 2\ \mathrm{m/s^2}$
- Velocity limits: $|\dot{\vec{x}}|, |\dot{\vec{y}}| \leq 5\ \mathrm{m/s}$

Boundary Conditions:

$$\vec{x}(0) = [0, 0, 0, 0]^T$$
$$\vec{x}(T) = [10, 5, 0, 0]^T$$

Cost Function: For minimum time: $J = T$ (equivalently, $L(\vec{x}, \vec{u}) = 1$)

Analytical Insight

This is a bang-bang control problem. Optimal solution:

- Accelerate at maximum toward target
- Decelerate at maximum to arrive at rest

Phase 1:

$$\vec{u} = u_{\max} \cdot \frac{\vec{d}}{|\vec{d}|}, \quad \vec{d} = [10, 5]^T$$

Phase 2:

$$\vec{u} = -u_{\max} \cdot \frac{\vec{v}}{|\vec{v}|}, \quad \vec{v} = \text{current velocity}$$

Numerical Solution

Normalized direction:

$$\hat{d} = \frac{[10, 5]}{\sqrt{125}} = [0.894, 0.447]$$

Phase 1 (Acceleration):

$$\vec{u} = 2 \cdot [0.894, 0.447] = [1.789, 0.894]$$

$$t_1 = \sqrt{\frac{|\vec{d}|}{|\vec{u}|}} = \sqrt{\frac{\sqrt{125}}{2}} \approx 2.5 \text{ seconds}$$

Phase 2 (Deceleration):

$$\vec{u} = -[1.789, 0.894], \quad t_2 = t_1 = 2.5 \text{ seconds}$$

Total time: $T = 5$ seconds

Example 2: Inverted Pendulum - Energy Optimal Swing-Up

Problem Setup

Swing up an inverted pendulum from hanging to upright with minimum energy.

State: $\vec{x} = [\theta, \dot{\theta}]^T$

Control: u (torque)

Dynamics:

$$\dot{\vec{x}} = \begin{bmatrix} \dot{\theta} \\ \frac{1}{I}(u - mgl\sin\theta - b\dot{\theta}) \end{bmatrix}$$

Parameters:

- $m = 1$ kg
- $l = 1$ m
- $I = ml^2 = 1$ kg·m^2
- $g = 9.81$ m/s^2
- $b = 0.1$ (damping)

Boundary Conditions:

$$\vec{x}(0) = [\pi, 0]^T$$
$$\vec{x}(T) = [0, 0]^T$$

Cost Function:

$$J = \int_0^T u^2 \, dt$$

This penalizes large control efforts, leading to smooth, efficient motion.

Solution Approach Using Direct Collocation

Step 1: Discretize Time

Divide $[0, T]$ into N segments:

$$t_k = k \cdot \frac{T}{N}, \quad k = 0, 1, \ldots, N$$

Step 2: Discrete Variables

- States: $\vec{x}_k \approx \vec{x}(t_k)$
- Controls: $u_k \approx u(t_k)$
- Time step: $h = T/N$

Step 3: Discrete Dynamics

Using Euler integration:

$$\vec{x}_{k+1} = \vec{x}_k + h \cdot f(\vec{x}_k, u_k)$$

Step 4: Optimization Problem

$$\min_{\{\vec{x}_k, u_k\}} \sum_{k=0}^{N-1} h \cdot u_k^2$$

Subject to:

$$\begin{aligned} \vec{x}_{k+1} - \vec{x}_k - h \cdot f(\vec{x}_k, u_k) = 0 \quad \text{(Dynamics)} \\ \vec{x}_0 = [\pi, 0]^T \\ \vec{x}_N = [0, 0]^T \\ |u_k| \leq u_{\max} \end{aligned}$$

Thank You