Reachability

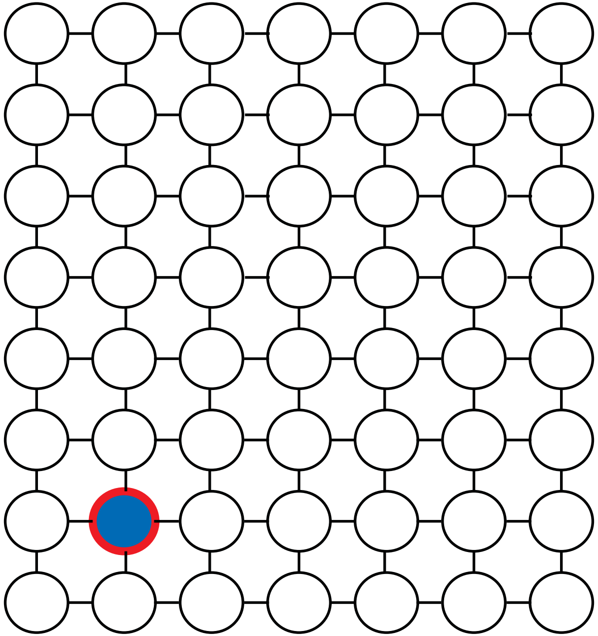
Outline

- **Reachability and representing reach sets**
- Applications - robust motion planning
- Computing reach sets
  - Flow Tubes
  - Funnels

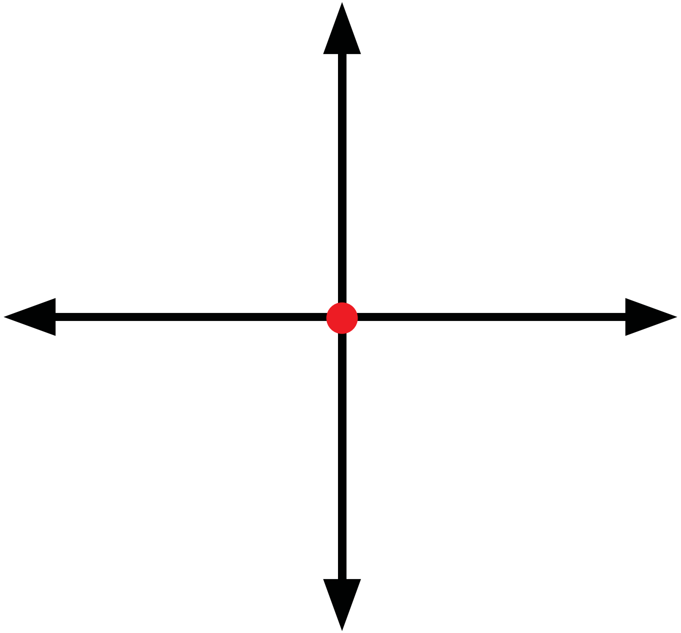
Definition

- Reachability is the task of figuring out what states a dynamical system could possibly reach.

Example, t = 0

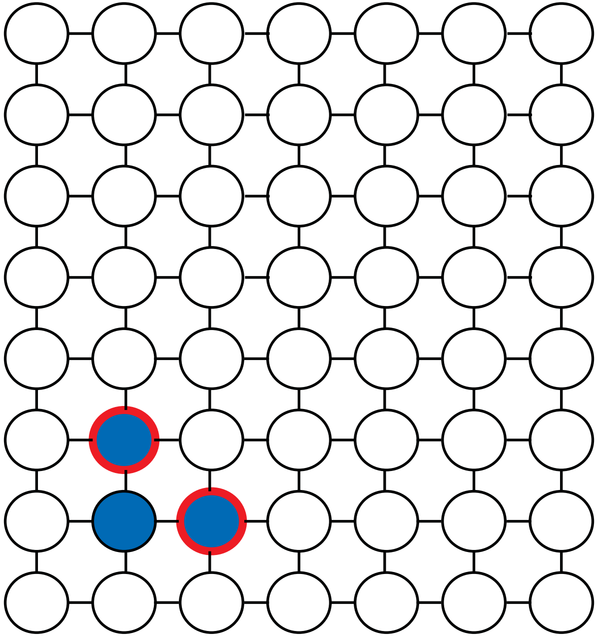


A = {E, N}

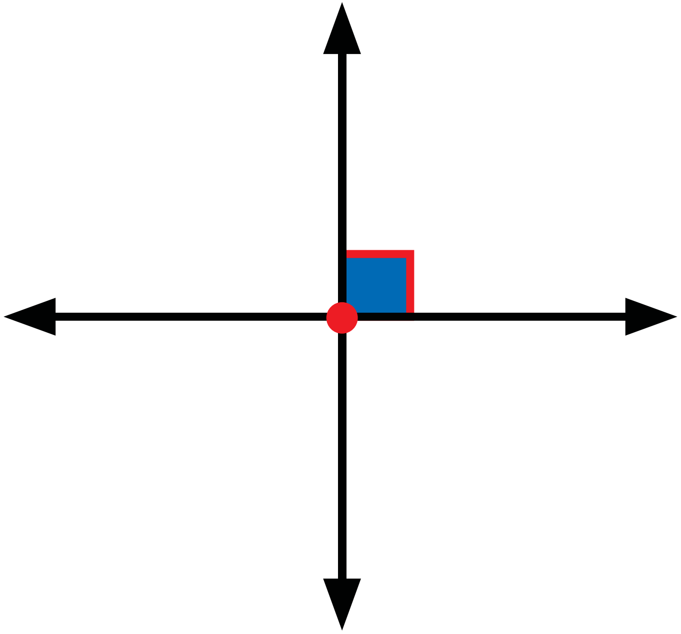


$$0 \le V_x \le 1$$
$$0 \le V_y \le 1$$

Example, t = 1

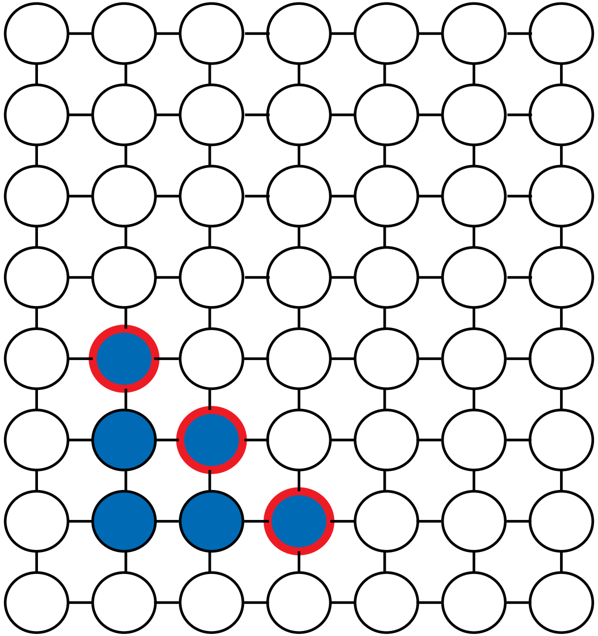


A = {E, N}

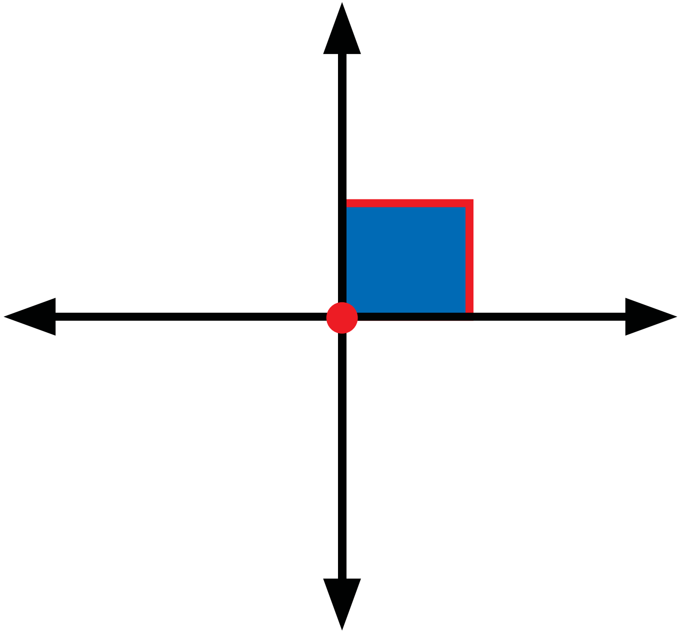


$$0 \le V_x \le 1$$
$$0 \le V_y \le 1$$

Example, t = 2

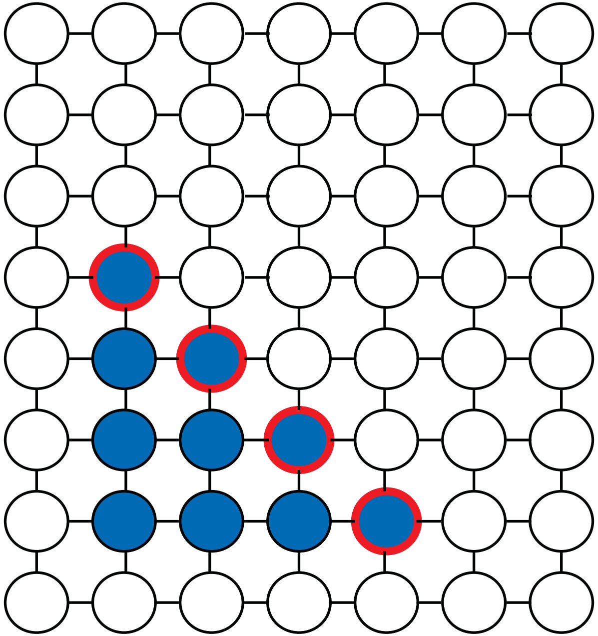


A = {E, N}

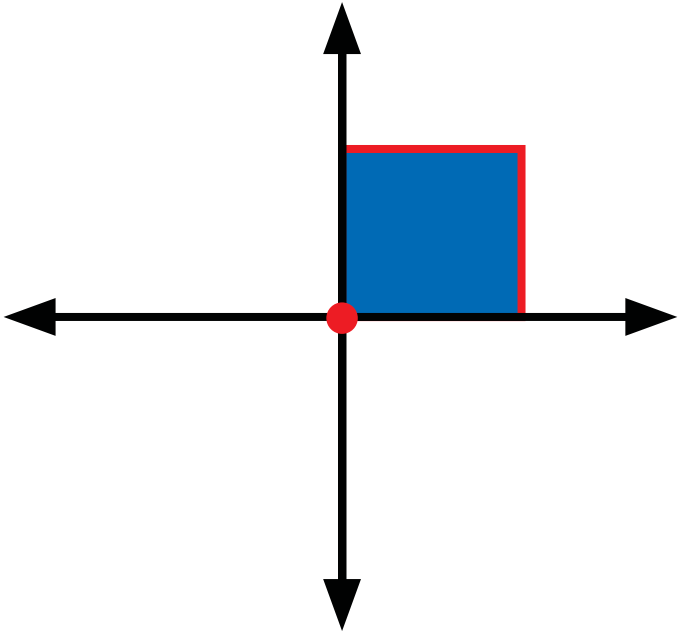


$$0 \le V_x \le 1$$
$$0 \le V_y \le 1$$

Example, t = 3

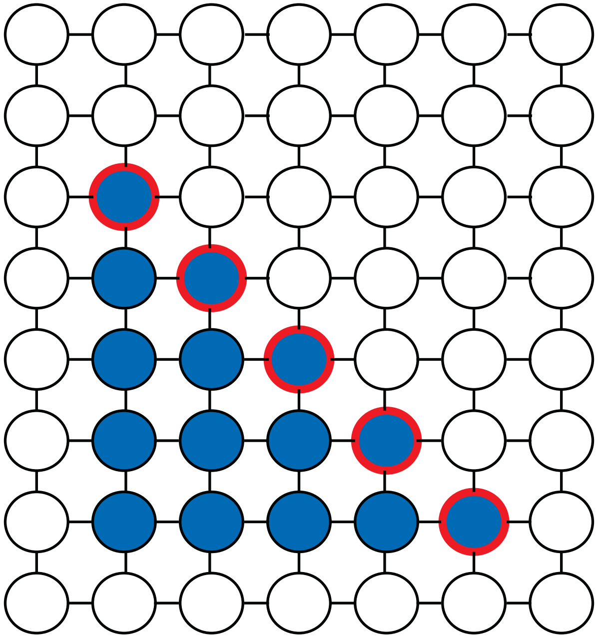


A = {E, N}

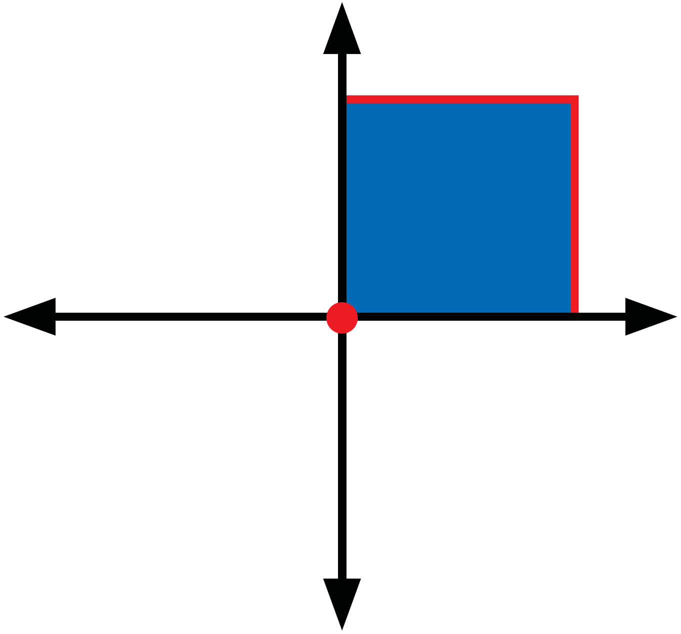


$$0 \leq V_x \leq 1$$
$$0 \leq V_y \leq 1$$

Example, t = 4



A = {E, N}



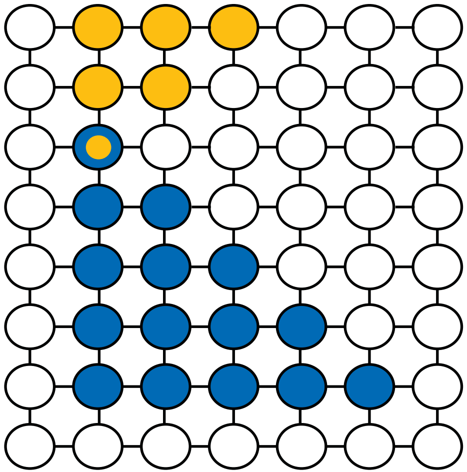
$$0 \leq V_x \leq 1$$
$$0 \leq V_y \leq 1$$

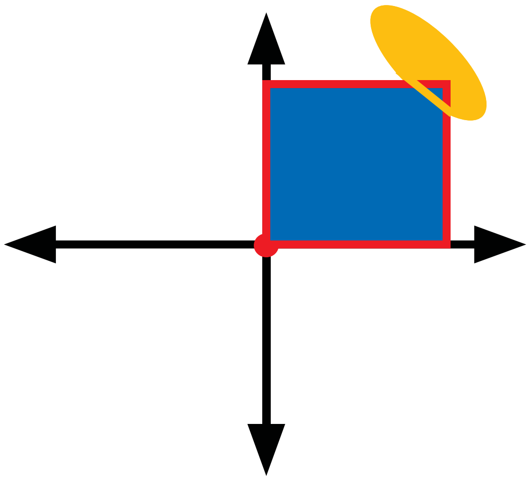
Motivation

- Reachability analysis is primarily used for verification.
- Generally, we test for intersection between the reachable set and a set of bad states.

Motivation

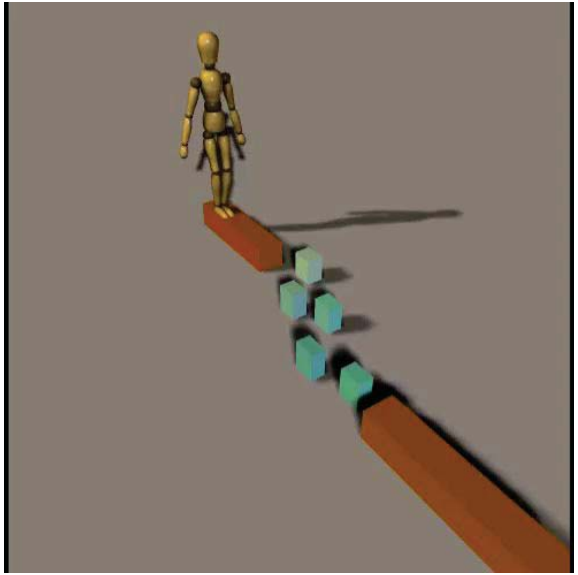
- Reachability analysis is often used for verification.
- Generally, we test for intersection between the reachable set and a set of bad states.





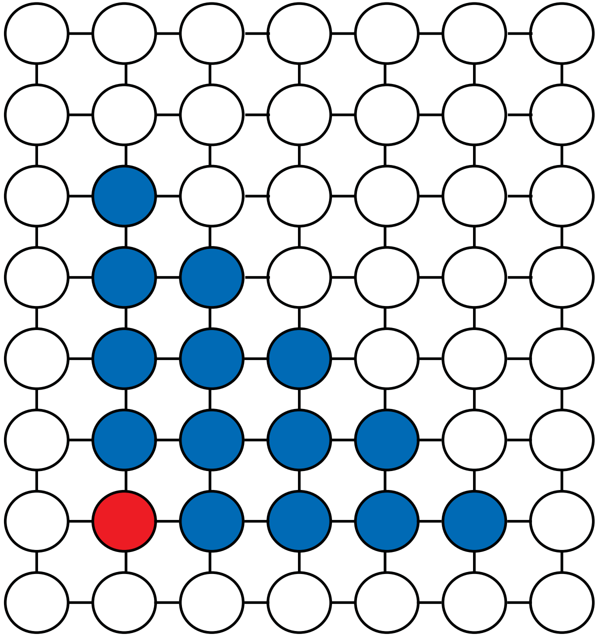
Motivation (continued)

- Reachability is used for robust motion planning.



Reachability on Finite State Machines

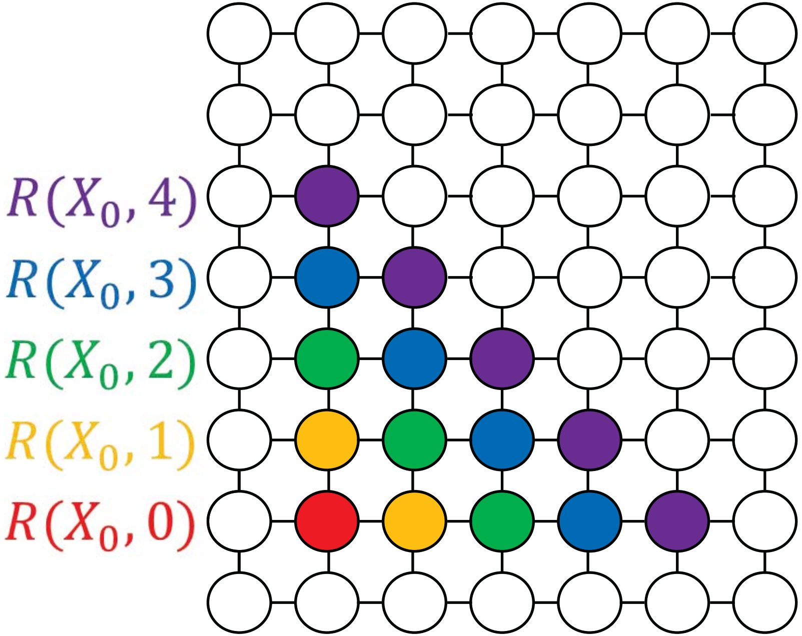
- $S = (X, U, T)$
  - X is the finite set of states
  - U is the finite set of control inputs
  - $T: X \times U \to X$ is the transition function
- X_0: set of initial states



U = {E, N}

Reachability on Finite State Machines

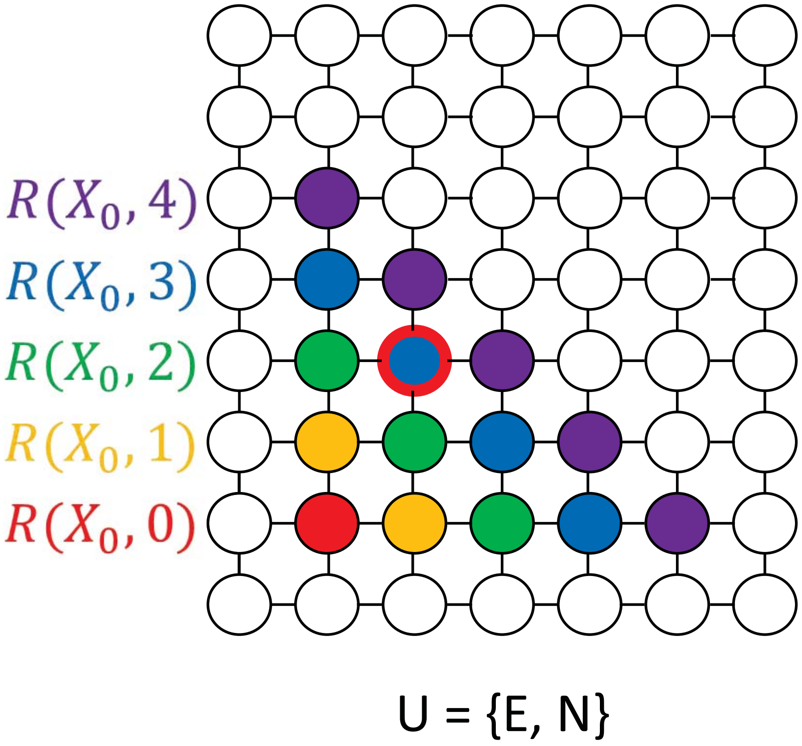
- $R(X_0, t)$: the **reach set** at time t is the set of states x for which there exists a sequence of control inputs $u_0, \ldots, u_{t-1}$ that would take us from a state $x_0 \in X_0$ to state x.



U = {E, N}

Reachability on Finite State Machines

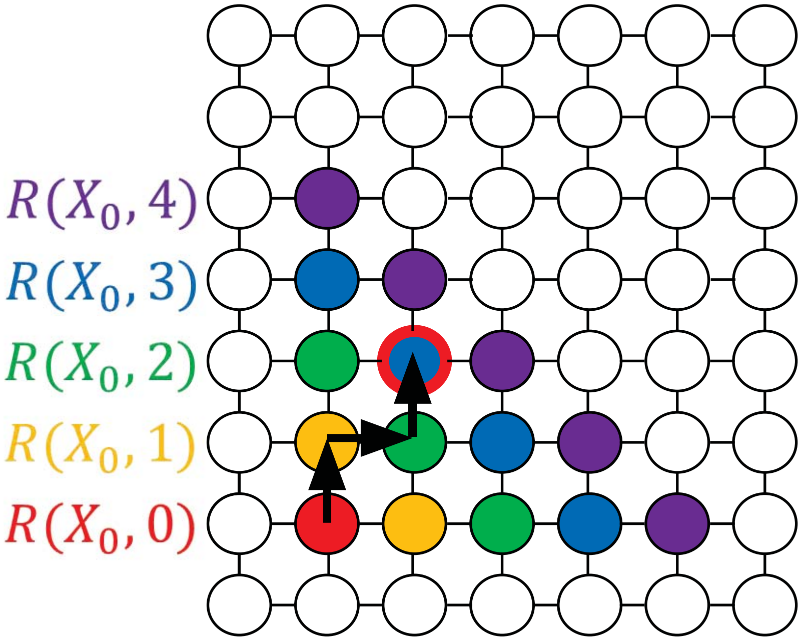
- $R(X_0, t)$: the **reach set** at time t is the set of states x for which there exists a sequence of control inputs $u_0, \ldots, u_{t-1}$ that would take us from a state $x_0 \in X_0$ to state x.



Reachability on Finite State Machines

- $R(X_0, t)$: the **reach set** at time t is the set of states x for which there exists a sequence of control inputs $u_0, \dots, u_{t-1}$ that would take us from a state $x_0 \in X_0$ to state x.

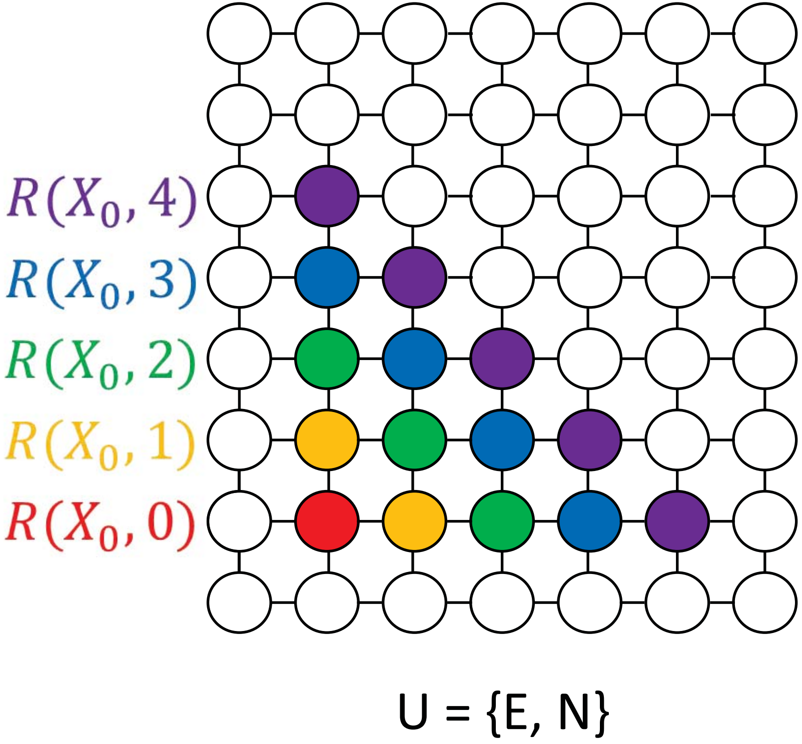
$$u_0, u_1, u_2 = N, E, N$$



U = {E, N}

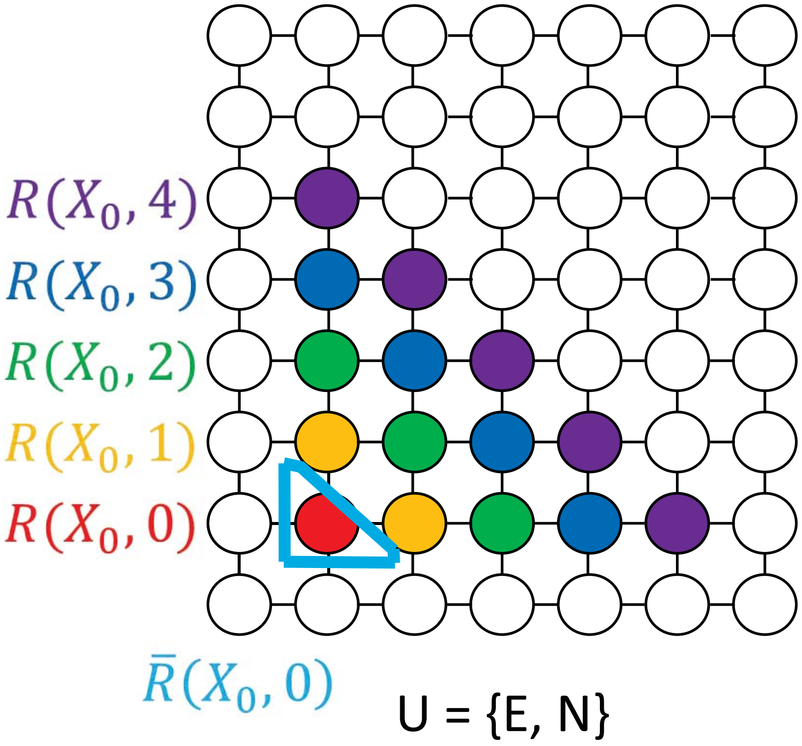
Reachability on Finite State Machines

- $\bar{R}(X_0, t) = \bigcup_{s \le t} R(X_0, s)$ is the **reachable set** at time t.



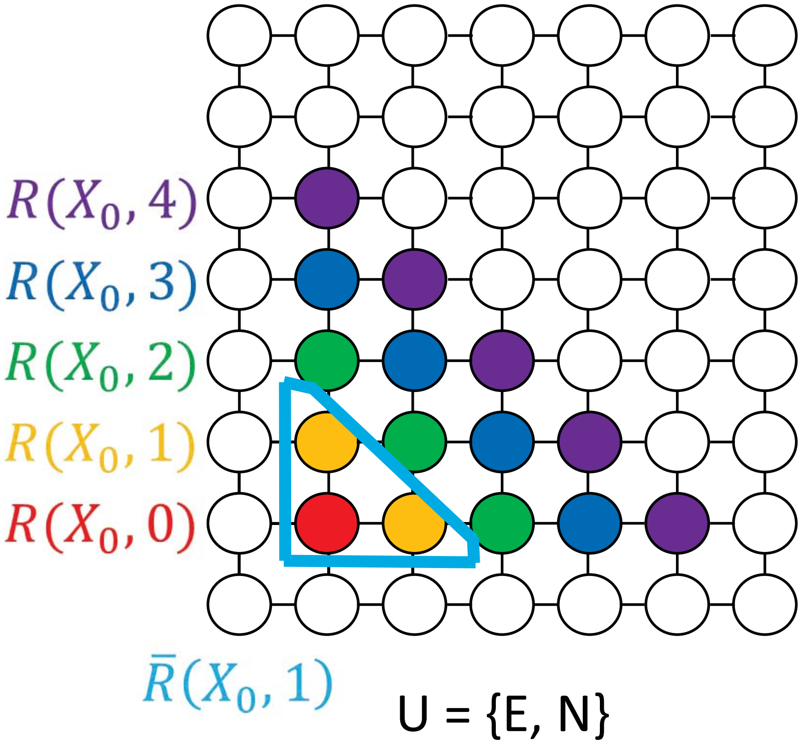
Reachability on Finite State Machines

- $\bar{R}(X_0, t) = \bigcup_{s \le t} R(X_0, s)$ is the **reachable set** at time t.



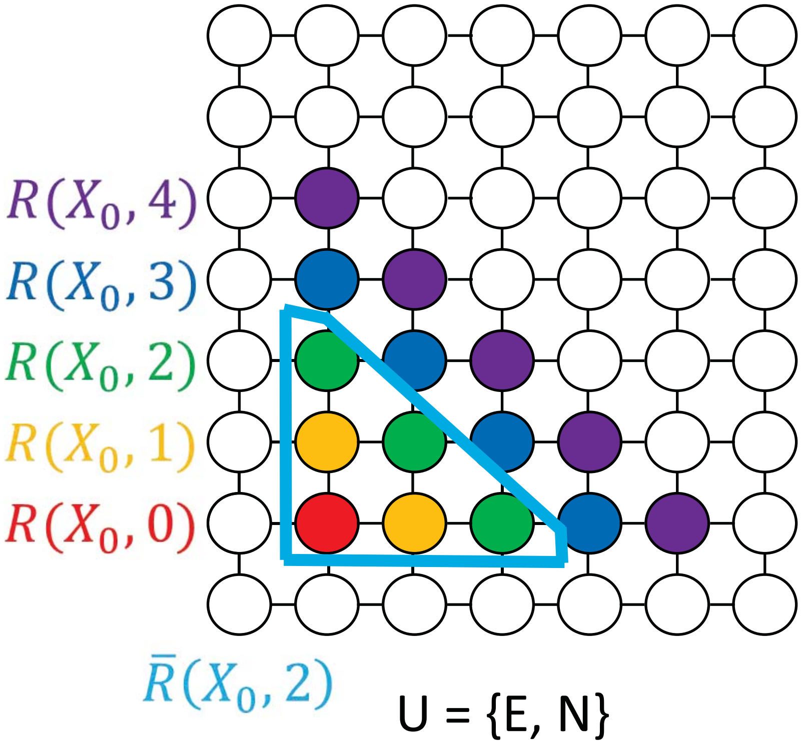
Reachability on Finite State Machines

- $\bar{R}(X_0, t) = \cup_{s \le t} R(X_0, s)$ is the **reachable set** at time t.



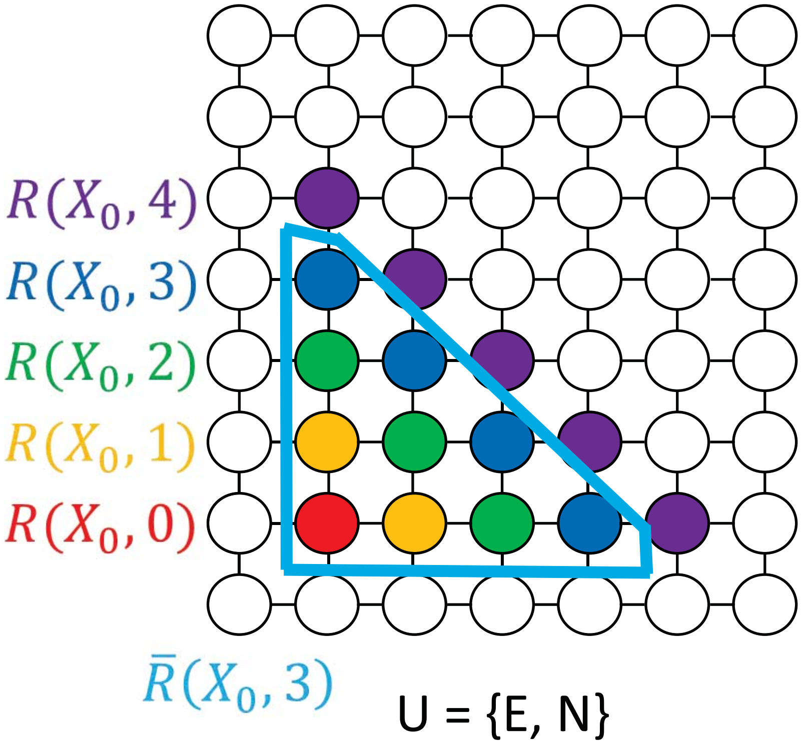
Reachability on Finite State Machines

- $\bar{R}(X_0, t) = \bigcup_{s \le t} R(X_0, s)$ is the **reachable set** at time t.



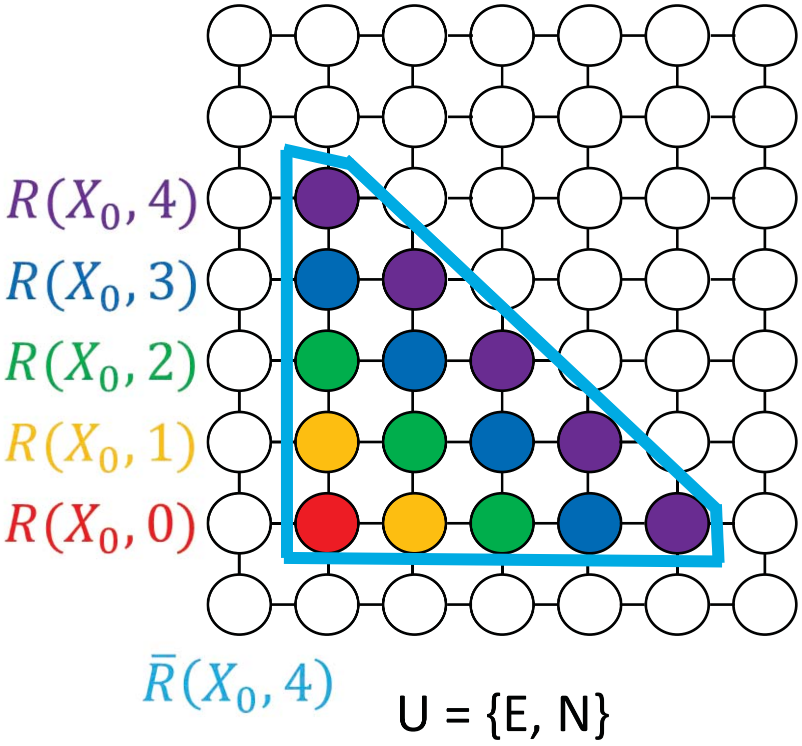
Reachability on Finite State Machines

- $\bar{R}(X_0, t) = \bigcup_{s \leq t} R(X_0, s)$ is the **reachable set** at time t.

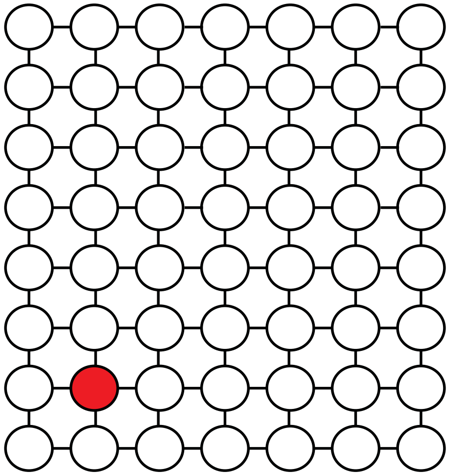


Reachability on Finite State Machines

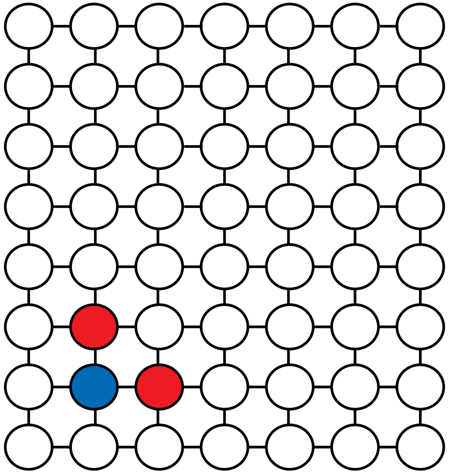
- $\bar{R}(X_0, t) = \cup_{s \leq t} R(X_0, s)$ is the **reachable set** at time t.



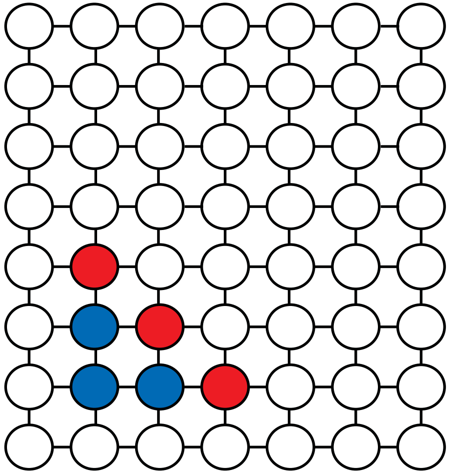
Computing reach sets



U = {E, N}
$R(X_0, 2) = ?$



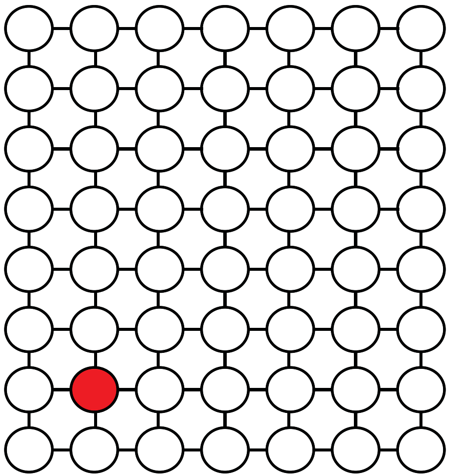
U = {E, N}
$R(X_0, 1)$



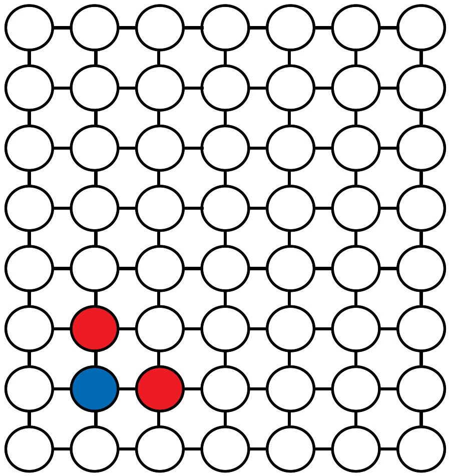
U = {E, N}
$R(X_0, 2) = R(R(X_0, 1), 1)$

Computing reach sets

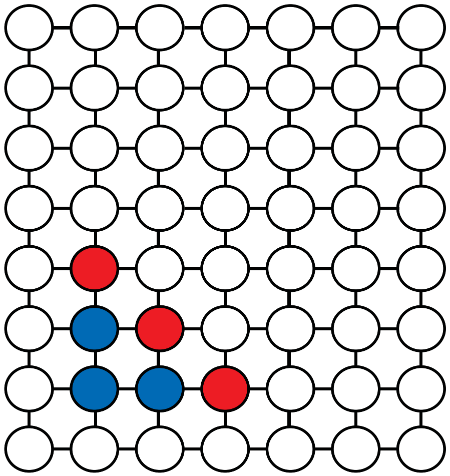
- This works because reach sets are semi-groups: $R(X_0, s+t) = R(R(X_0, s), t)$



U = {E, N}
$R(X_0, 2) = ?$



U = {E, N}
$R(X_0, 1)$



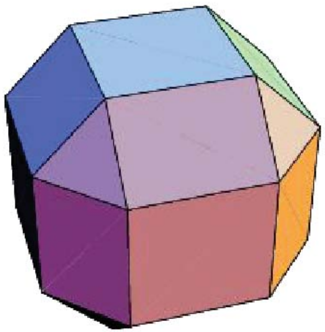
U = {E, N}
$R(X_0, 2) = R(R(X_0, 1), 1)$

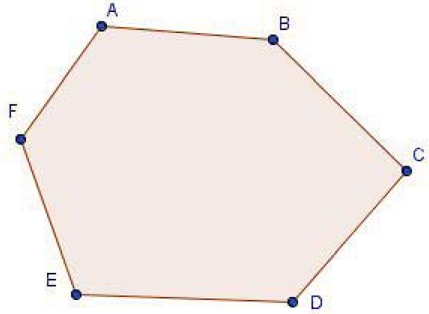
Continuous Systems

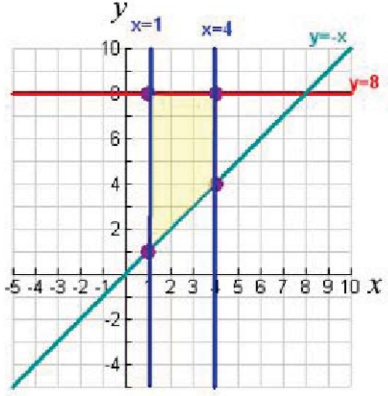
- In an FSM, we could represent reach sets as finite sets.
- In a continuous system, a reach set will be a region of the state space, so we need a symbolic representation.

Convex polytopes

- Two canonical representations:
  - Vertices: polytope = convex_hull(vertices)
  - Inequalities: polytope = $\bigcap(solutions\ to\ inequalities)$



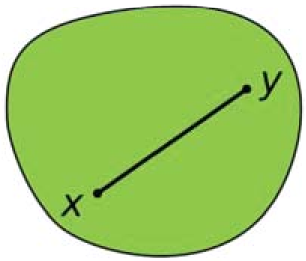




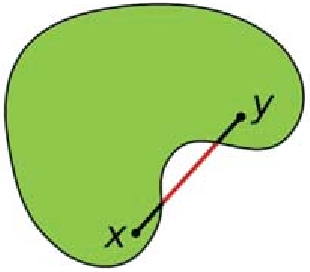
Convexity

- For every pair of points within the region, every point on the straight line segment that joins the pair of points is also within the region.

Convex

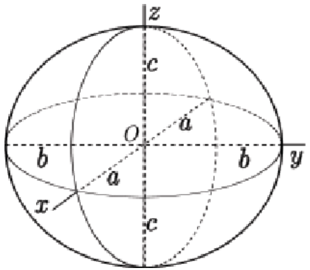


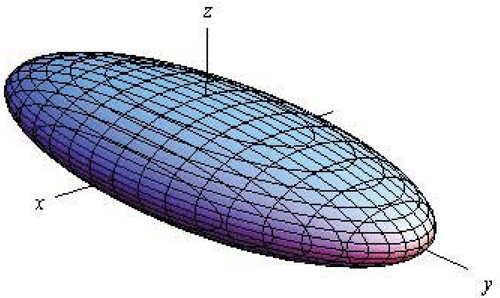
Non-convex



Ellipsoids

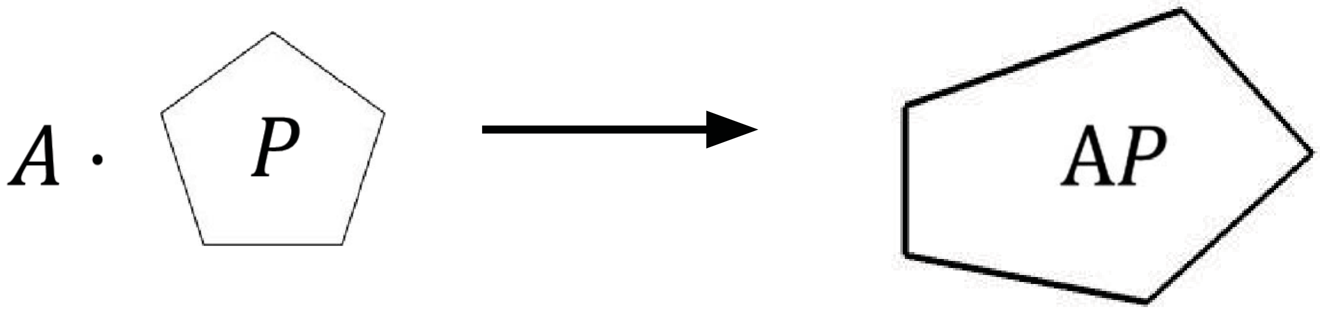
- An arbitrary ellipsoid can be represented with the following: $(x - v)^T A \, (x - v) \leq 1$





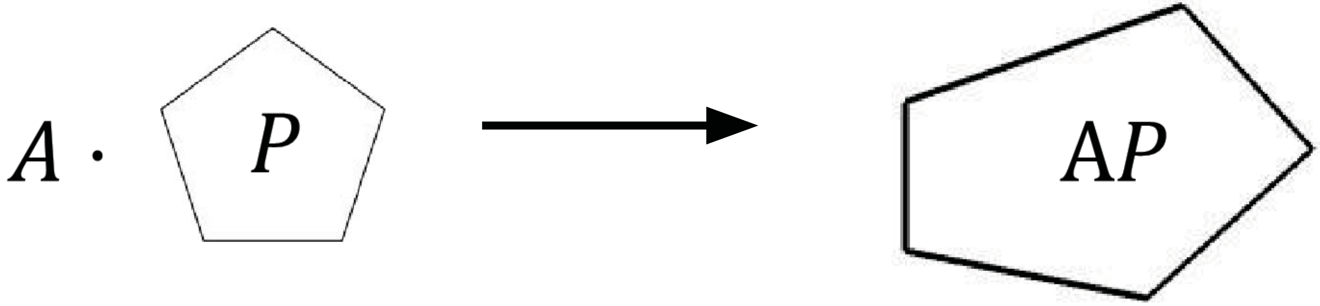
Closure under linear operators

- Let P be a convex polytope (resp. ellipsoid), then: $AP = \{Ax : x \in P\}$ is also a convex polytope (resp. ellipsoid).



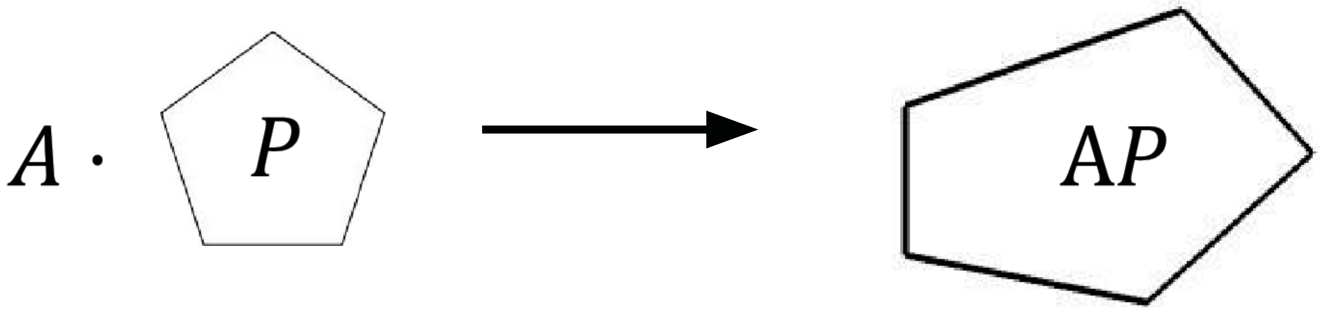
Closure under linear operators

- This means that for a system defined by $\boldsymbol{x_{i+1}} = \boldsymbol{Ax_i}$, or linear system defined by $\dot{\boldsymbol{x}}(\boldsymbol{s}) = \boldsymbol{Ax}(\boldsymbol{s}) + \boldsymbol{u}(\boldsymbol{s})$*, if we start with a convex X_0, then the $\mathrm{R}(X_0, t)$ will also be convex.



Closure under linear operators

- This means that for linear systems defined by $\boldsymbol{x_{i+1}} = \boldsymbol{Ax_i}$ or $\dot{\boldsymbol{x}}(\boldsymbol{s}) = \boldsymbol{Ax}(\boldsymbol{s}) + \boldsymbol{u}(\boldsymbol{s})*$, if we start with a convex X_0, then the $\mathrm{R}(X_0, t)$ will also be convex.



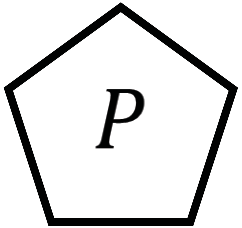
* Requires U and X_0 to be convex and compact, where $u(s) \in U$.

Closure under linear operators

- Even if the reach set is convex, the reachable set $\overline{\boldsymbol{R}}(\boldsymbol{X_0}, \boldsymbol{t})$ is not necessarily convex, but it will be a **union of the convex polytopes (resp. ellipsoid).**

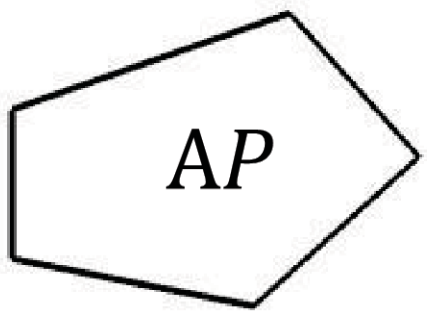
Closure under linear operators

- Even if the reach set is convex, the reachable set $\overline{\boldsymbol{R}}(\boldsymbol{X_0}, \boldsymbol{t})$ is not necessarily convex, but it will be a **union of the convex polytopes (resp. ellipsoid).**



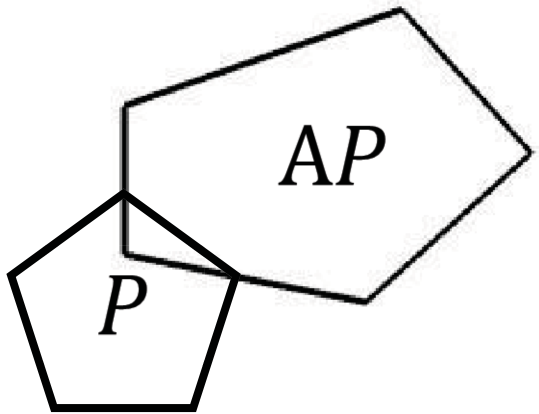
Closure under linear operators

- Even if the reach set is convex, the reachable set $\overline{\boldsymbol{R}}(\boldsymbol{X_0}, \boldsymbol{t})$ is not necessarily convex, but it will be a **union of the convex polytopes (resp. ellipsoid).**



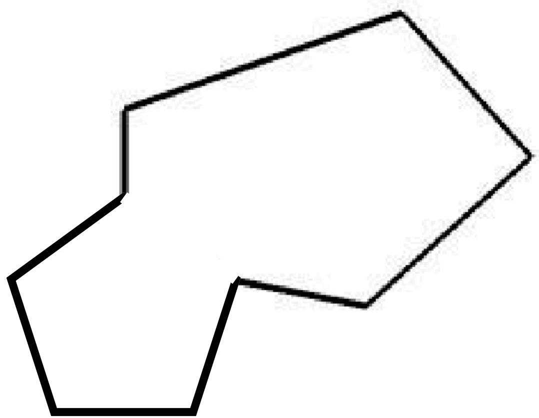
Closure under linear operators

- Even if the reach set is convex, the reachable set $\overline{\boldsymbol{R}}(\boldsymbol{X_0}, \boldsymbol{t})$ is not necessarily convex, but it will be a **union of the convex polytopes (resp. ellipsoid).**



Closure under linear operators

- Even if the reach set is convex, the reachable set $\overline{\boldsymbol{R}}(\boldsymbol{X_0}, \boldsymbol{t})$ is not necessarily convex, but it will be a **union of the convex polytopes (resp. ellipsoid).**



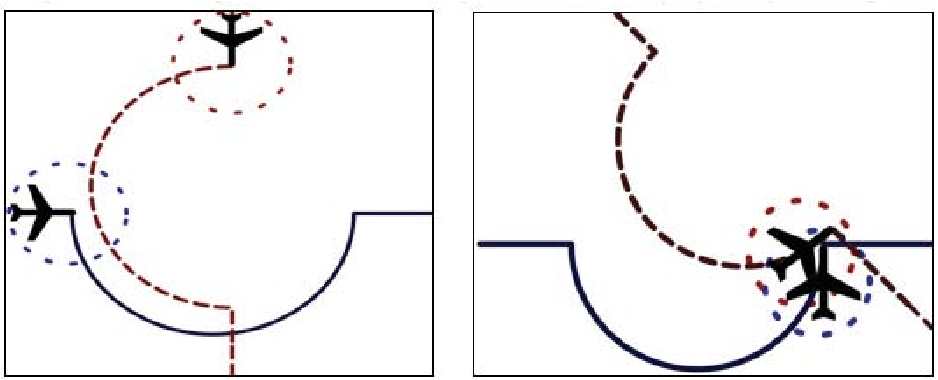
Outline

- Reachability and representing reach sets
- **Applications - robust motion planning**
- Computing reach sets
  - Flow Tubes
  - Funnels

Applications

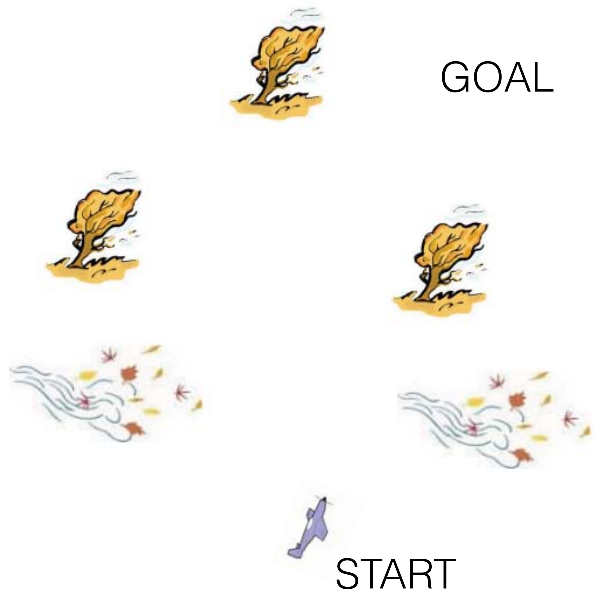
Robust motion planning

Aircraft collision avoidance

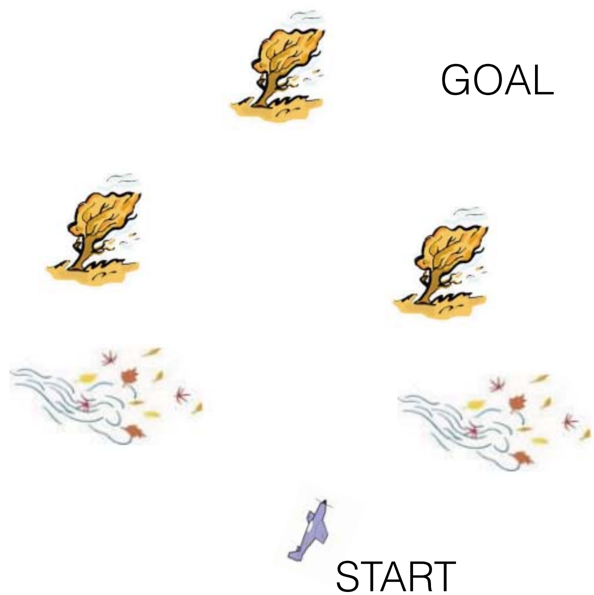


Controlling complex systems

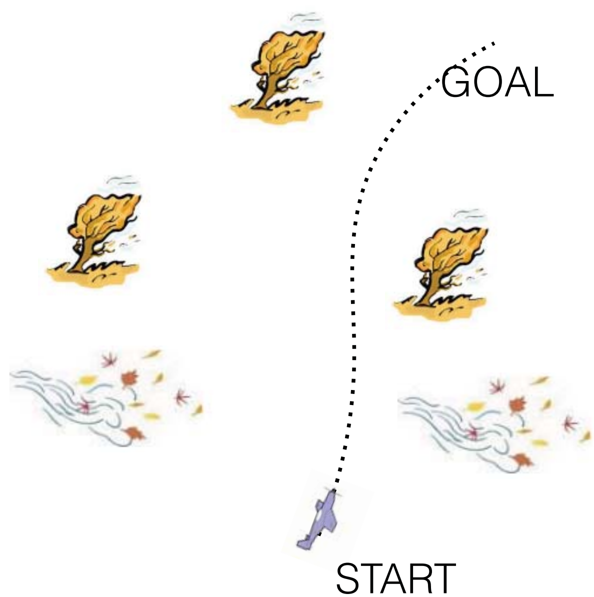
Robust motion planning



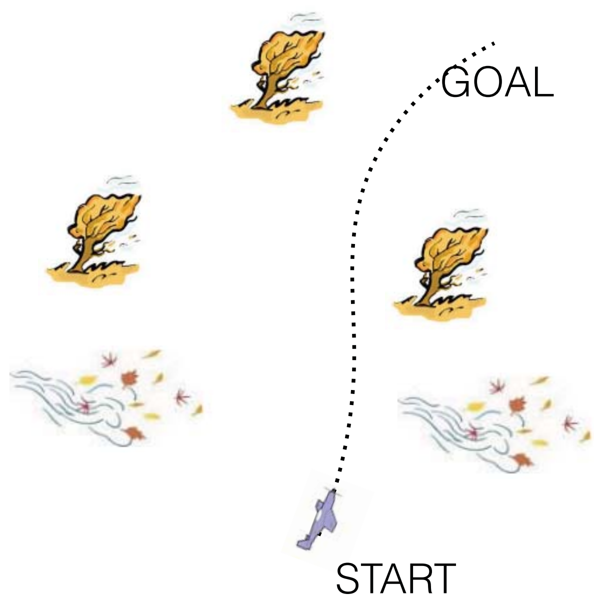
Robust motion planning



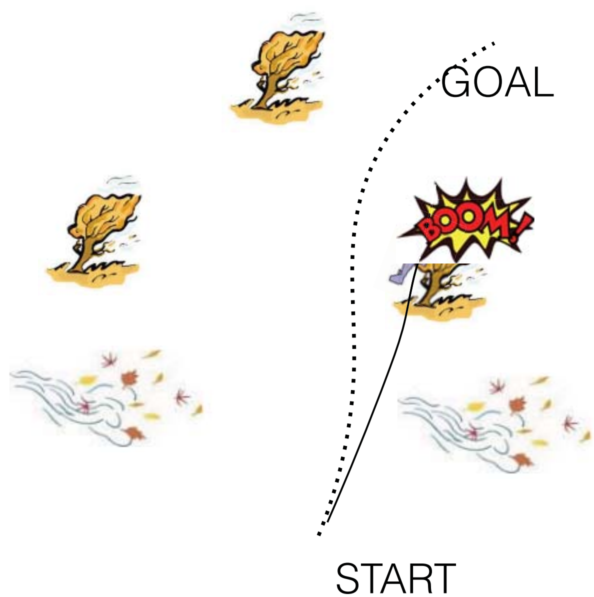
Robust motion planning



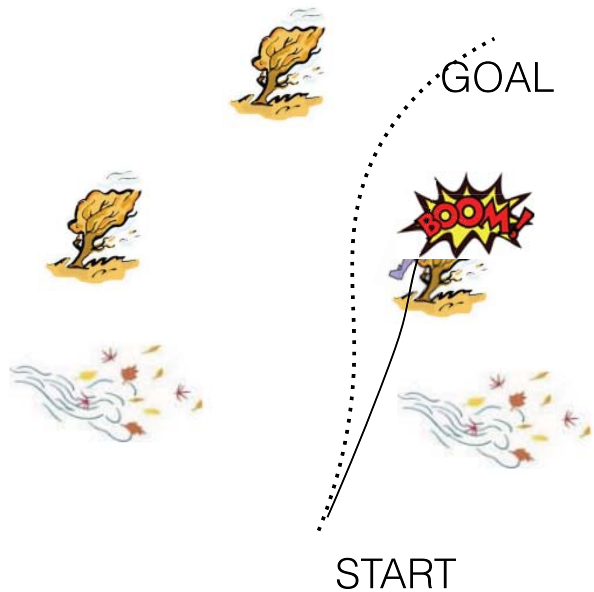
Robust motion planning



Robust motion planning

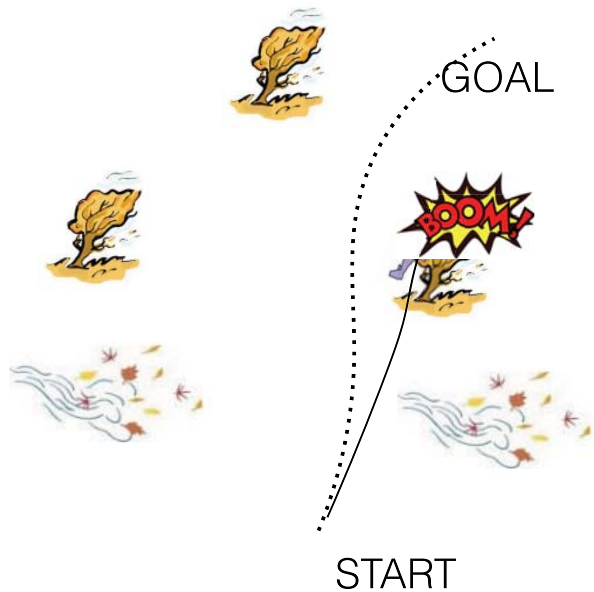


Robust motion planning



- Environmental disturbances (wind)
- Modeling errors
- State uncertainty
- Randomness in initial conditions

Robust motion planning

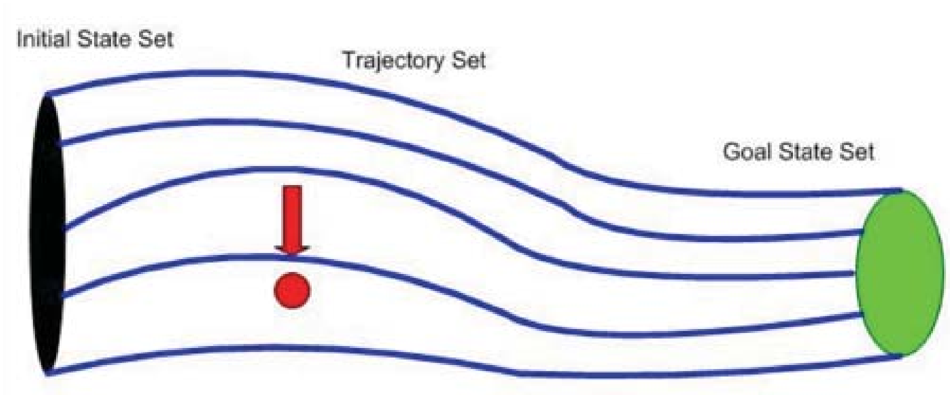


- Environmental disturbances (wind)
- Modeling errors
- State uncertainty
- Randomness in initial conditions

Robustness goal: Need to guarantee (with some confidence) that the system reaches a goal state and does not reach any states insides the obstacle sets under uncertainty

Timeline:

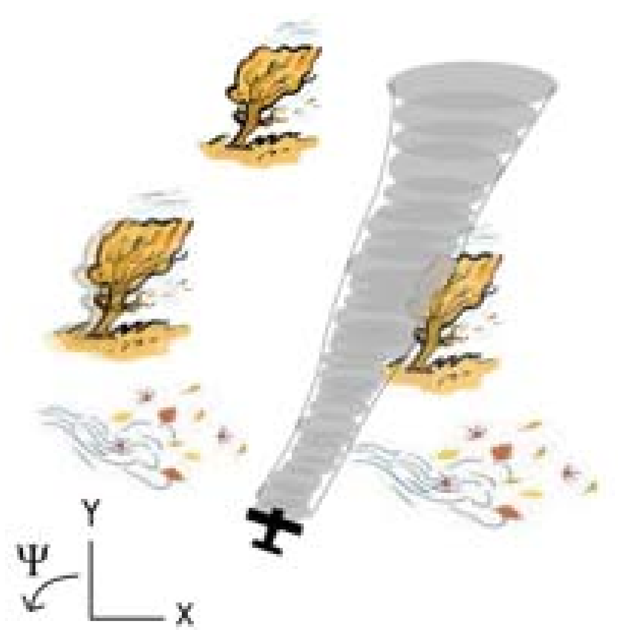
- Bradley and Zhao, 1993; Frazzoli, 2001 - Flow tubes



- Hoffman and Williams, 2006 - Flow tubes with temporal constraints
- Tobenkin, Manchester, and Tedrake, 2011; Majumdar and Tedrake, 2012 - Funnels

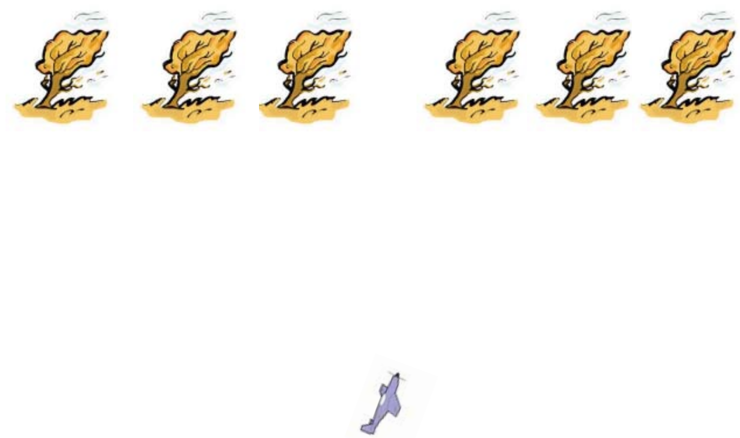
Motion planning with funnels

- Generate regions of finite time invariance (“funnels”) subjected to a general class of uncertainty



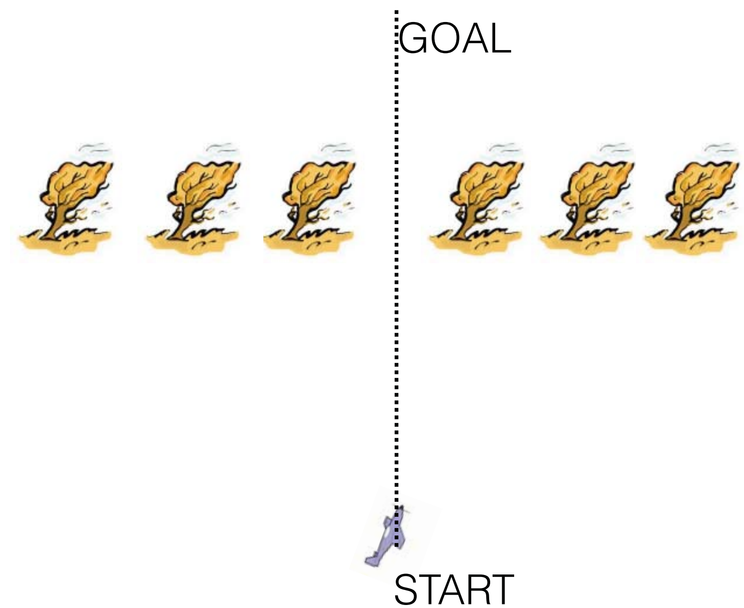
Motion planning with funnels

GOAL

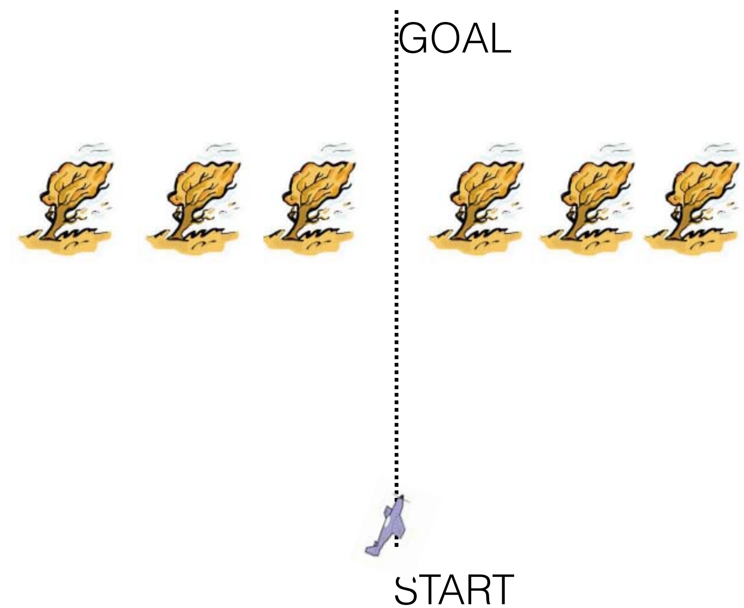


START

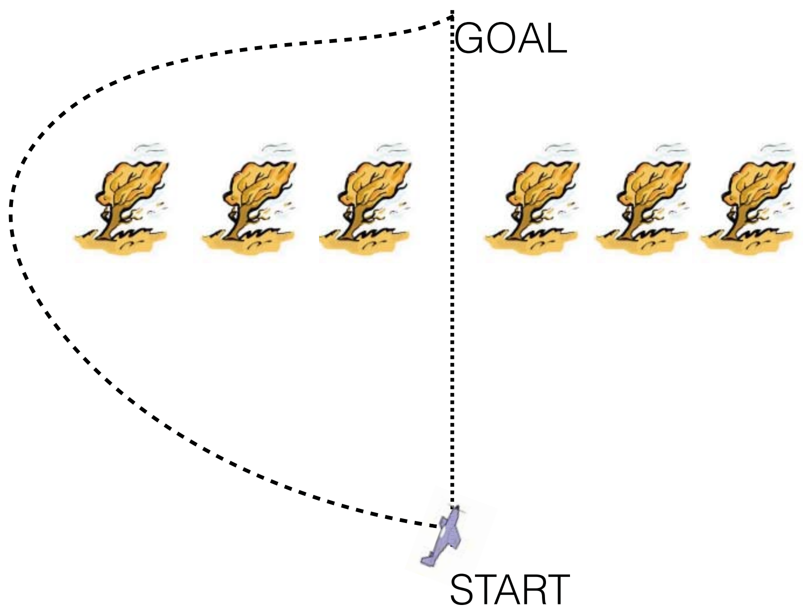
Motion planning with funnels



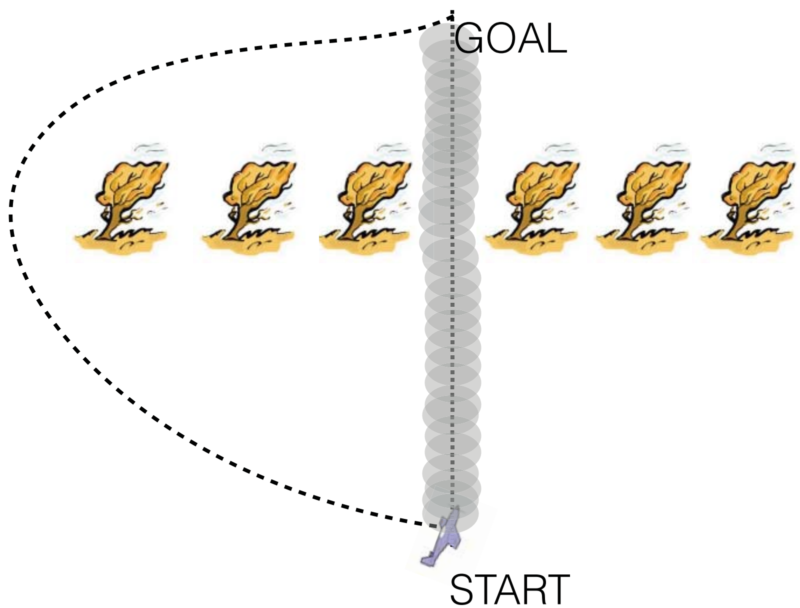
Motion planning with funnels



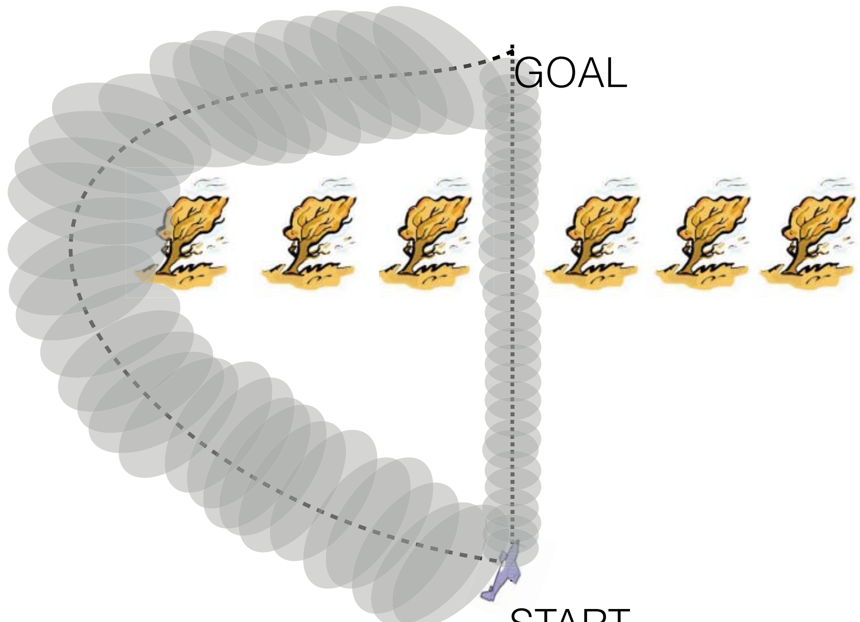
Motion planning with funnels



Motion planning with funnels



Motion planning with funnels

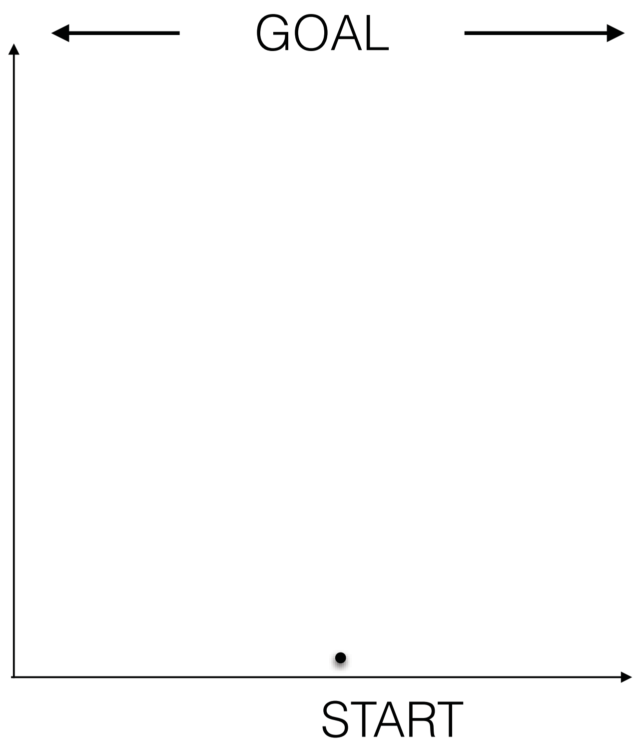


Reachability analysis can help distinguish between “intuitively less risky” paths from actual “safe” paths

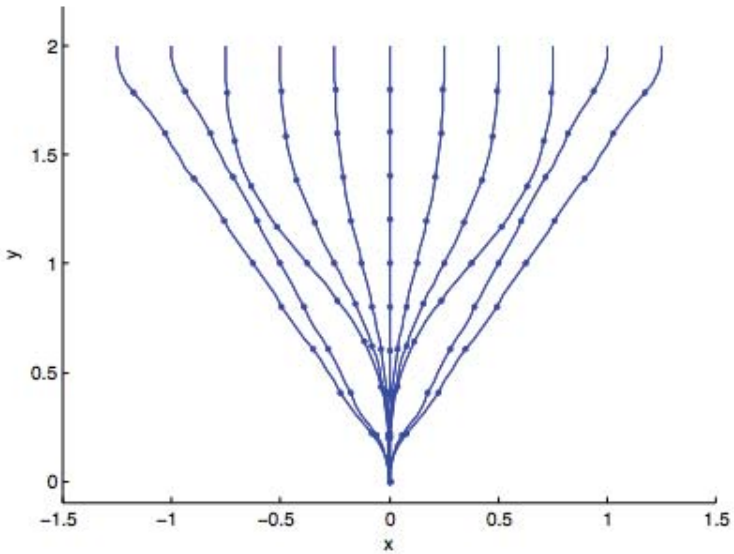
Online planning with funnel libraries

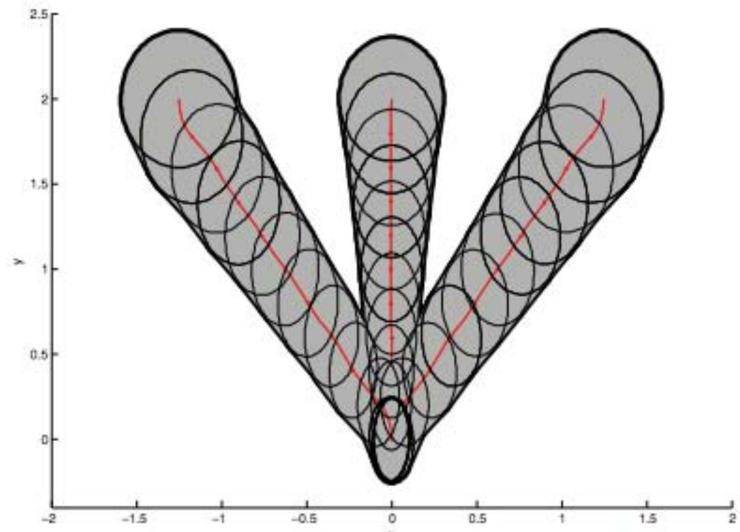
- Not all information about the environment is known before hand
- Cannot perform expensive computations during runtime
- Create libraries of funnels offline — one for each possible trajectory

Online planning with funnel libraries



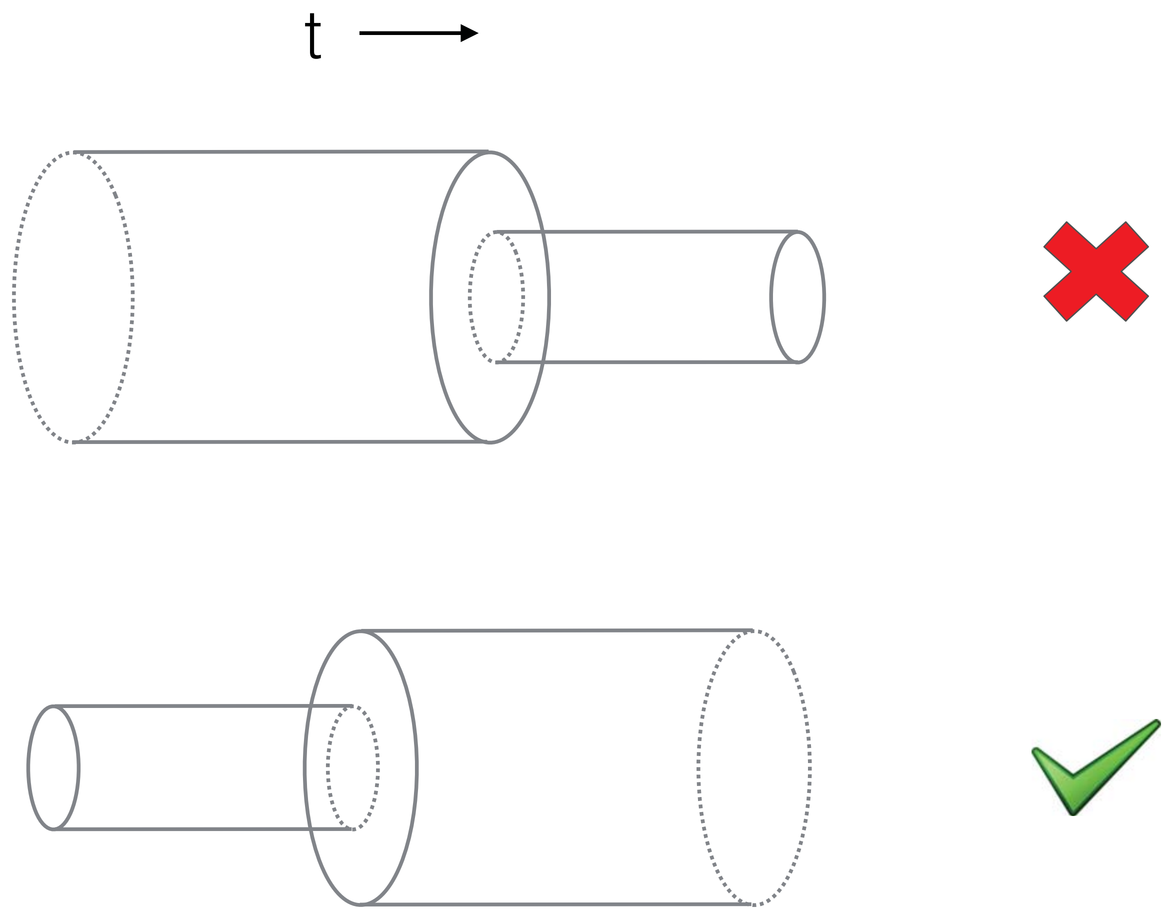
Online planning with funnel libraries



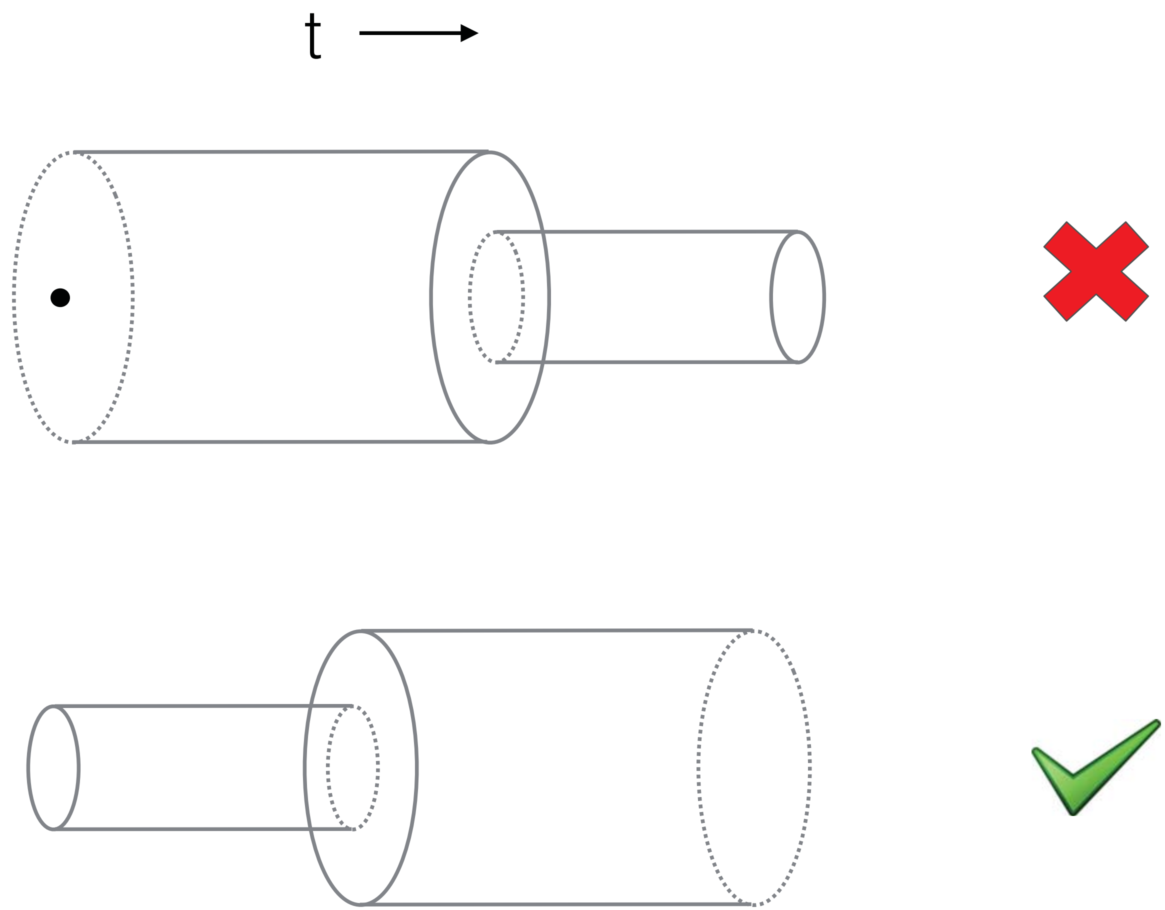


Problem reduces to finding a sequential composition of funnels to avoid obstacles

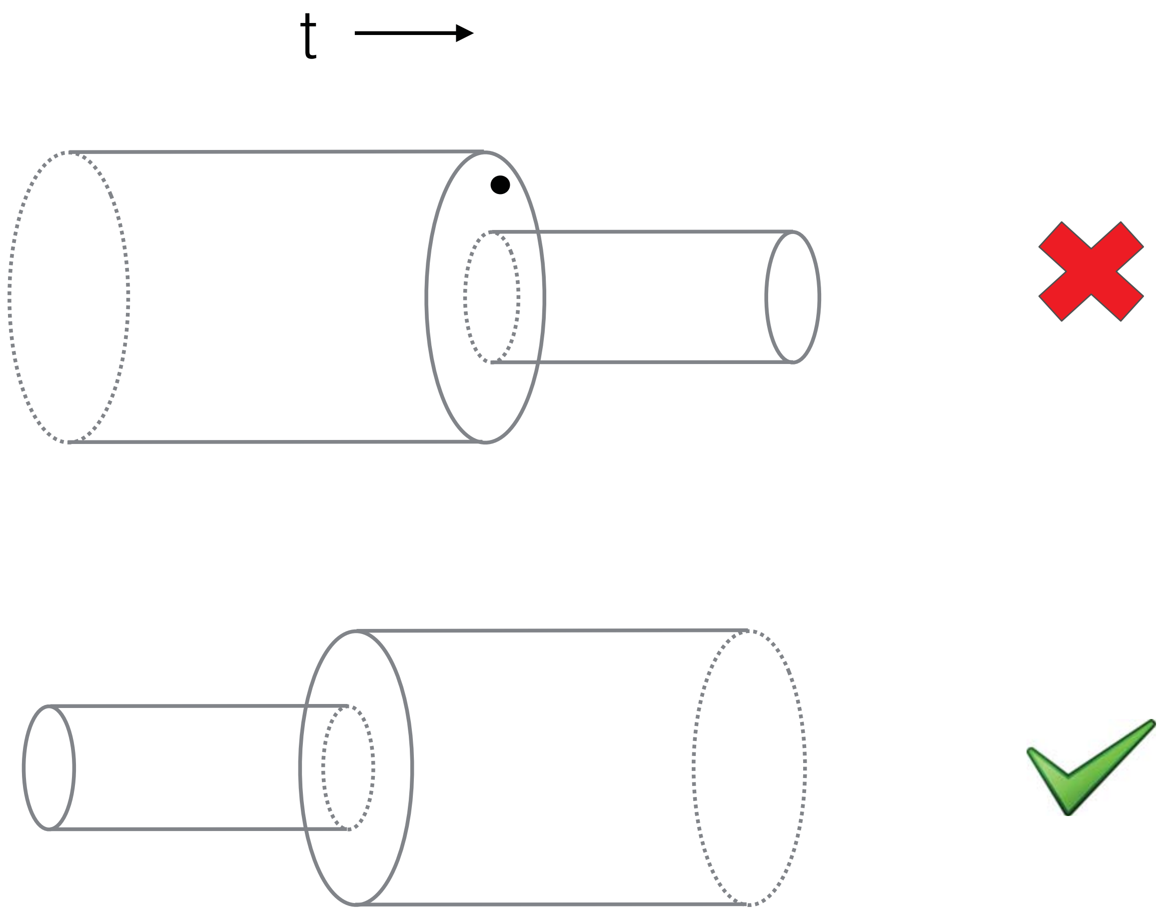
Sequential composition



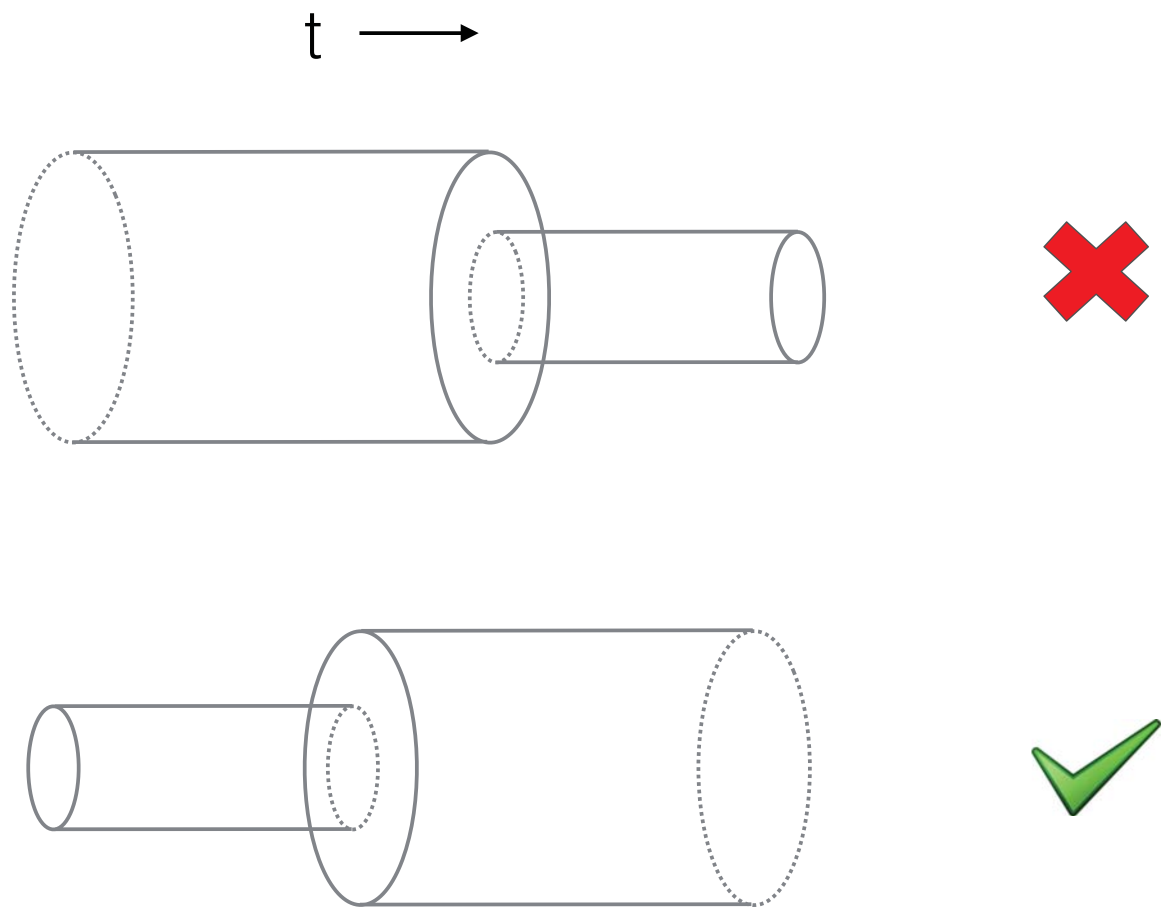
Sequential composition



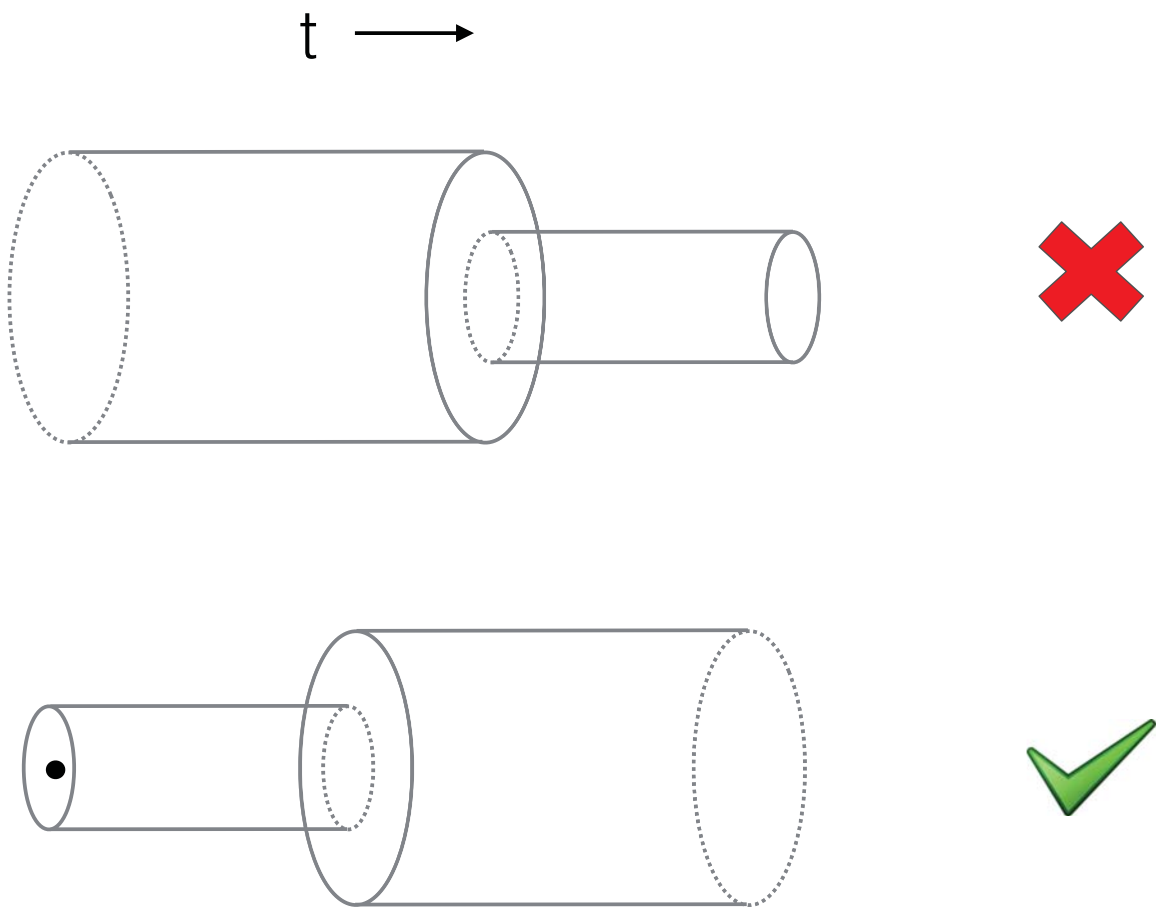
Sequential composition



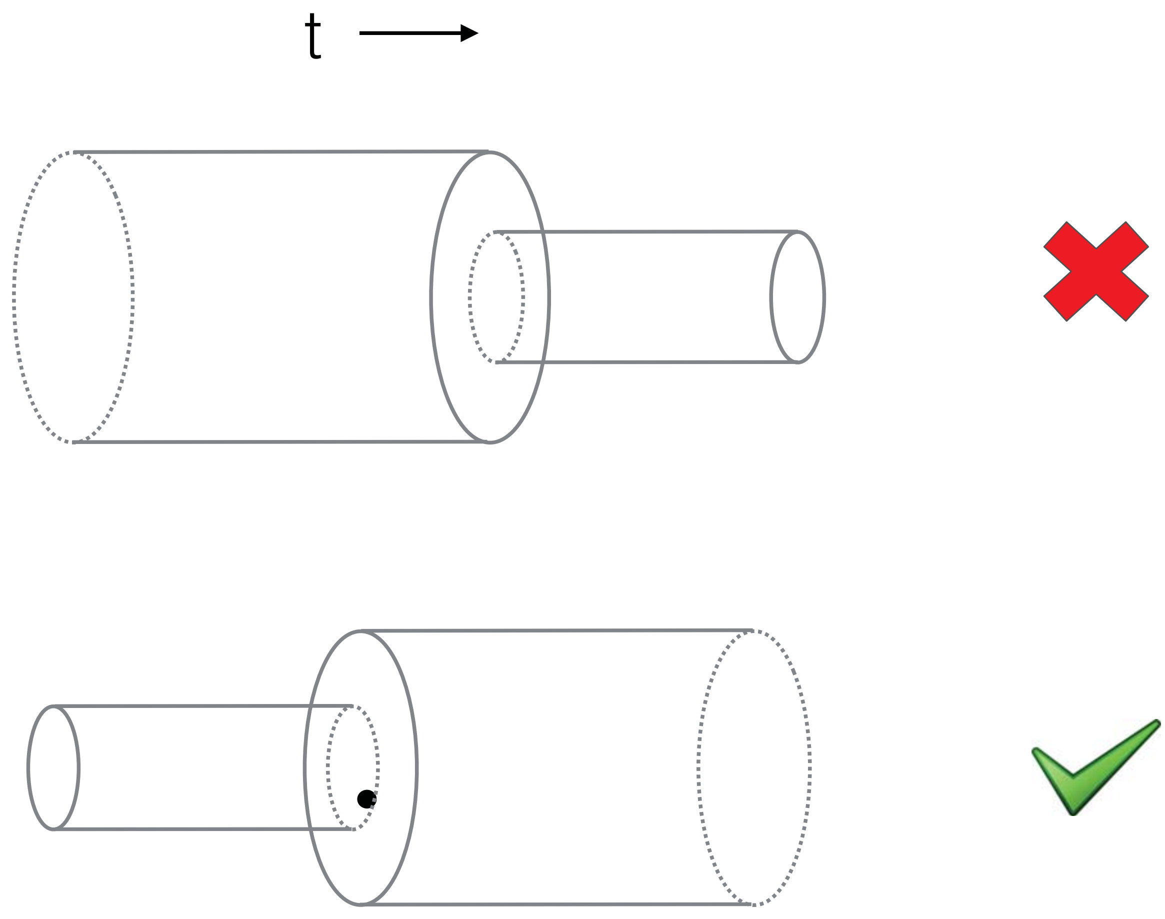
Sequential composition



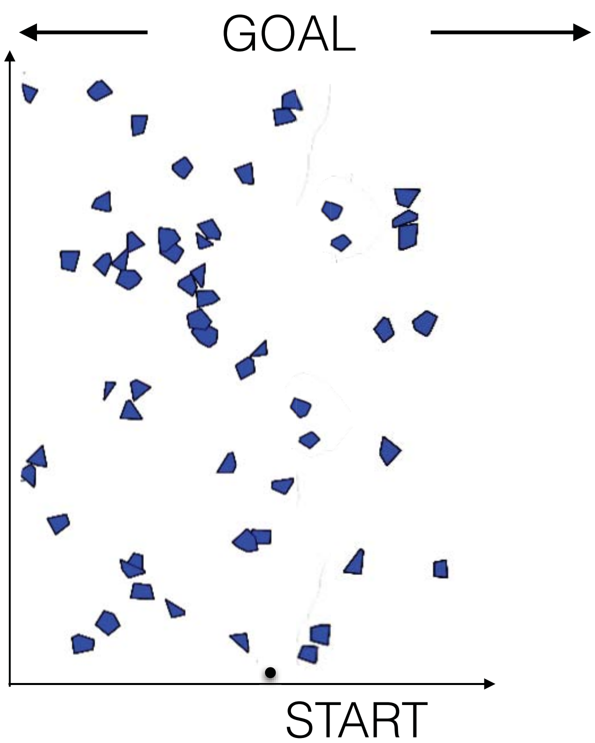
Sequential composition



Sequential composition

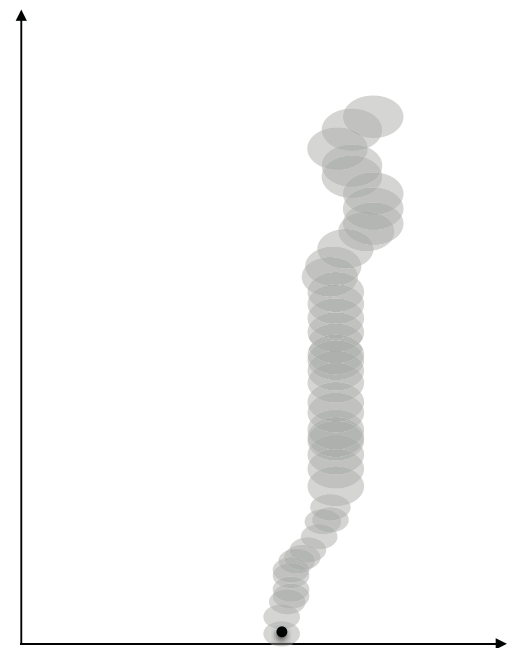


Online planning



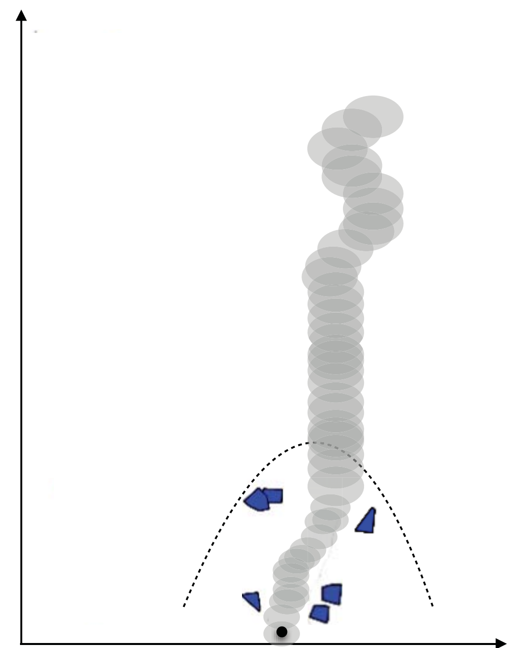
Online planning

1. **Initial planned funnel sequence, P**
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



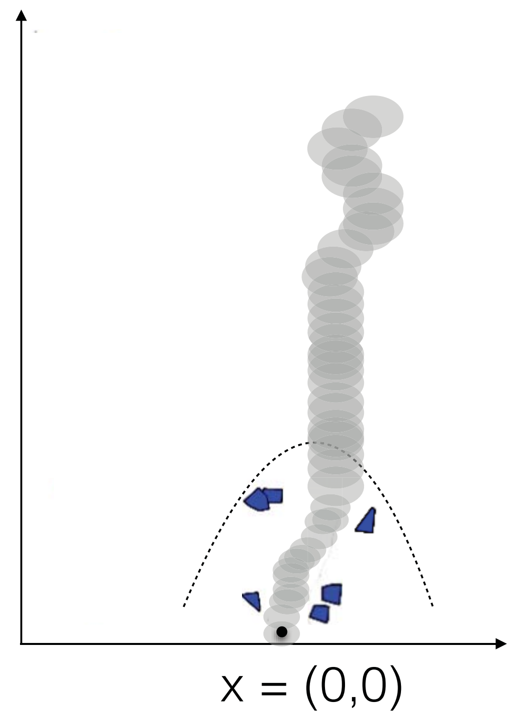
Online planning

1. Initial planned funnel sequence, P
2. **Update obstacles information, O**
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



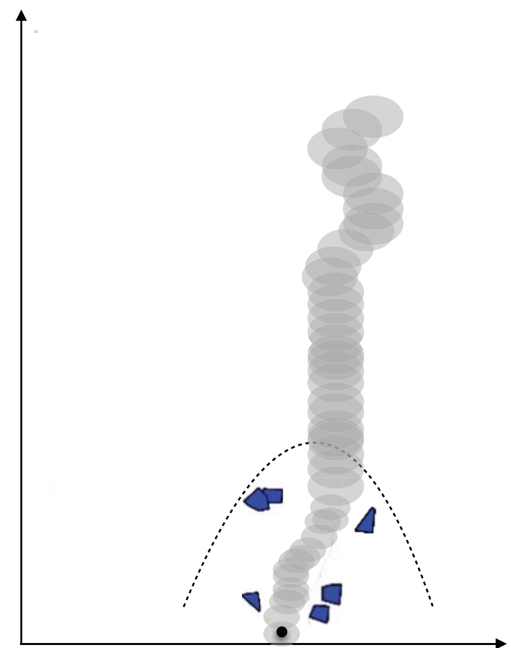
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. **Get current state of robot, x**
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



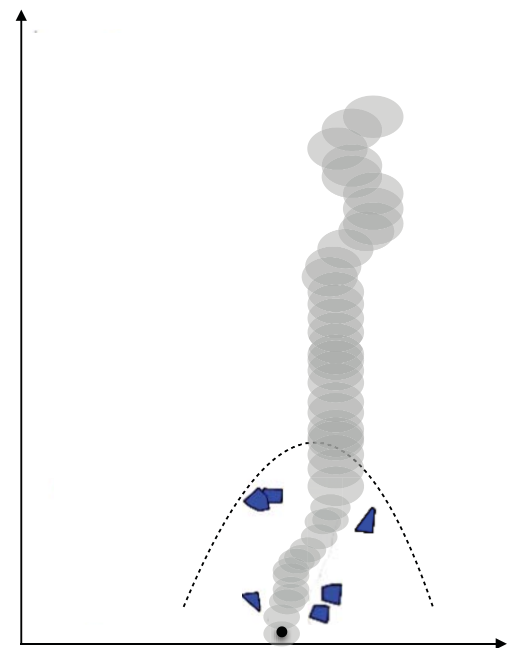
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. **Check if P collides with any of the obstacles**
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



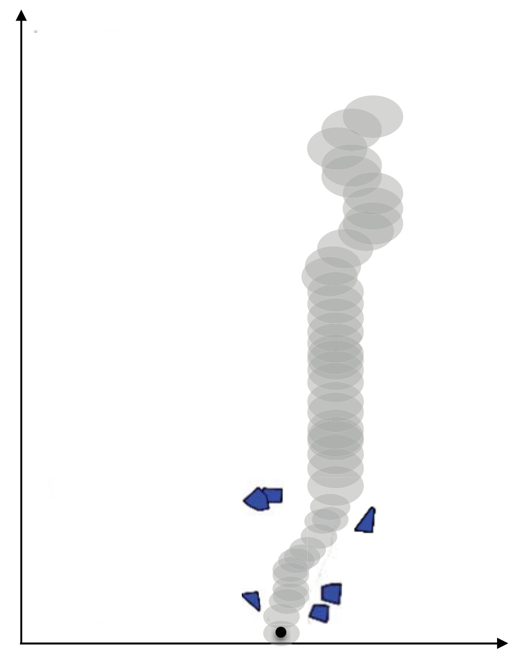
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. **Apply control corresponding to P and x**
7. Goto 2



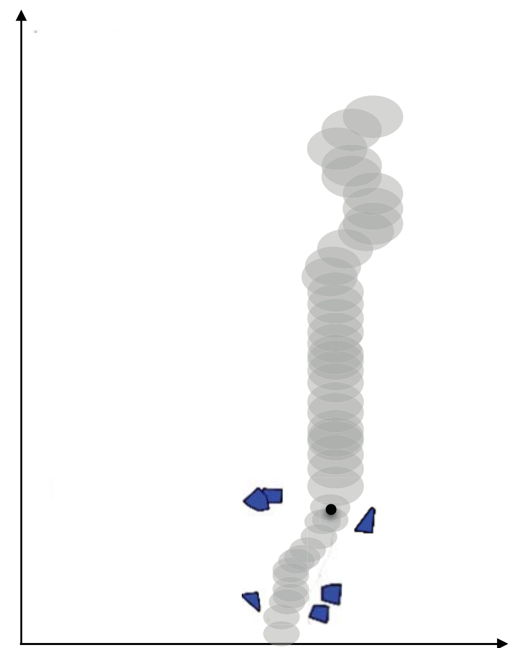
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - P = ReplanFunnels(x, O)
6. **Apply control corresponding to P and x**
7. Goto 2



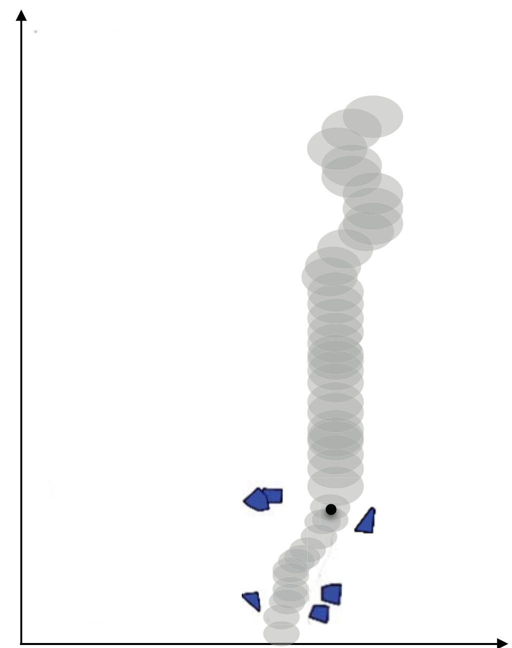
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. **Goto 2**



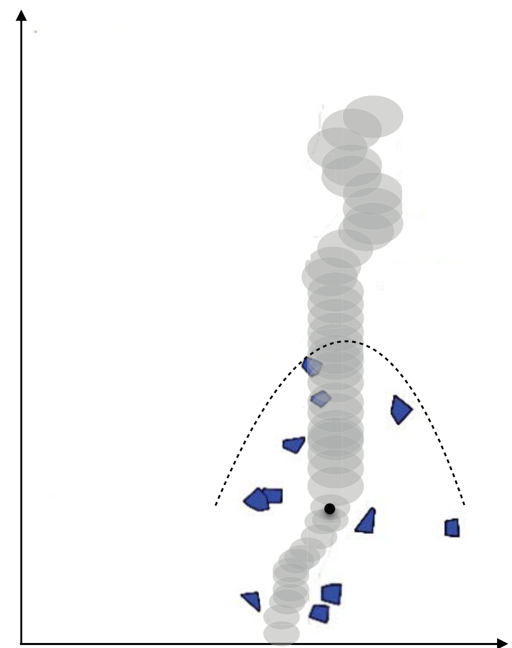
Online planning

1. Initial planned funnel sequence, P
2. **Update obstacles information, O**
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



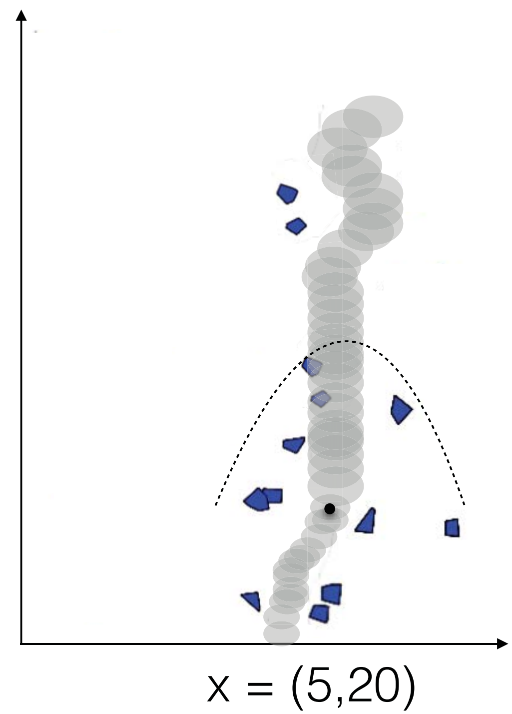
Online planning

1. Initial planned funnel sequence, P
2. **Update obstacles information, O**
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



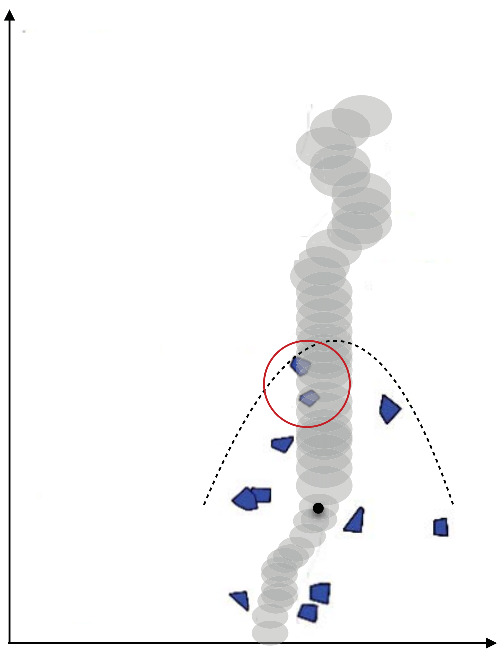
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. **Get current state of robot, x**
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



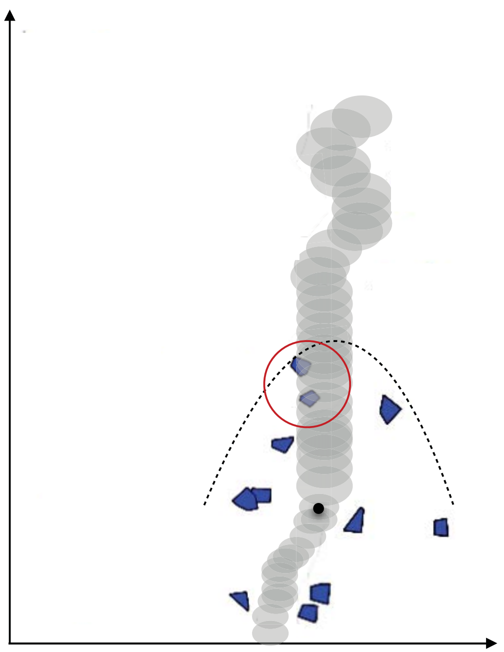
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. **Check if P collides with any of the obstacles**
5. If collision
    - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



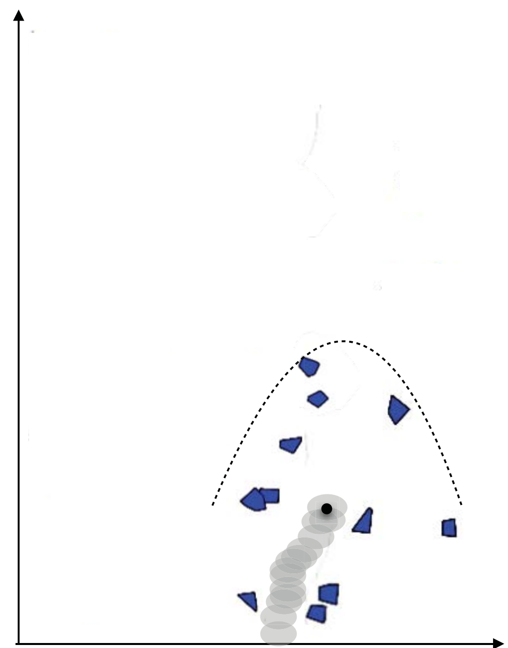
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



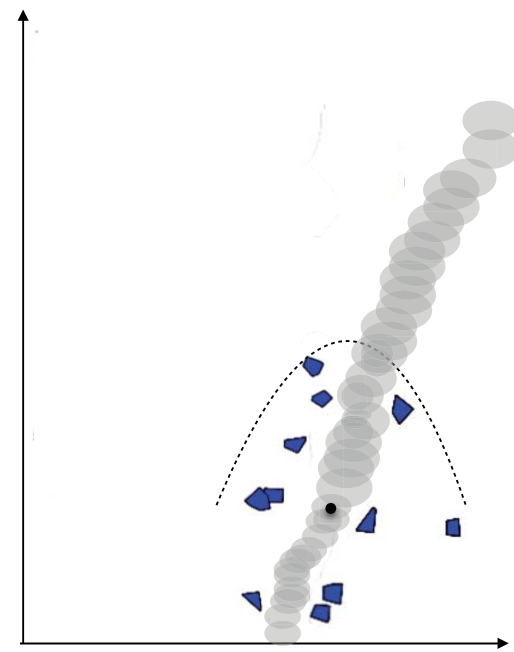
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



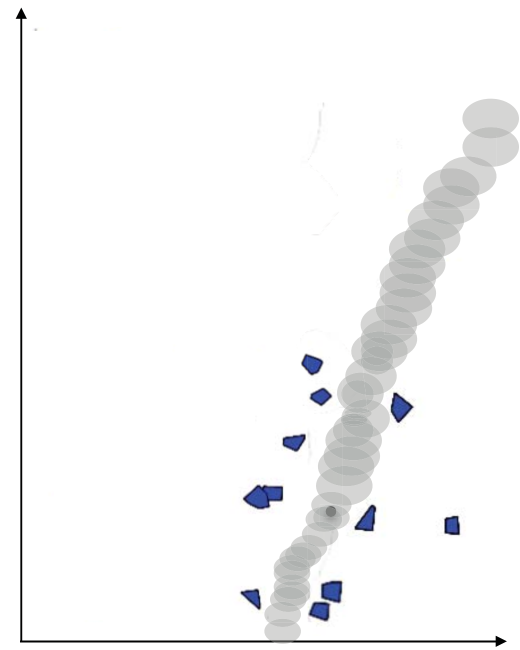
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



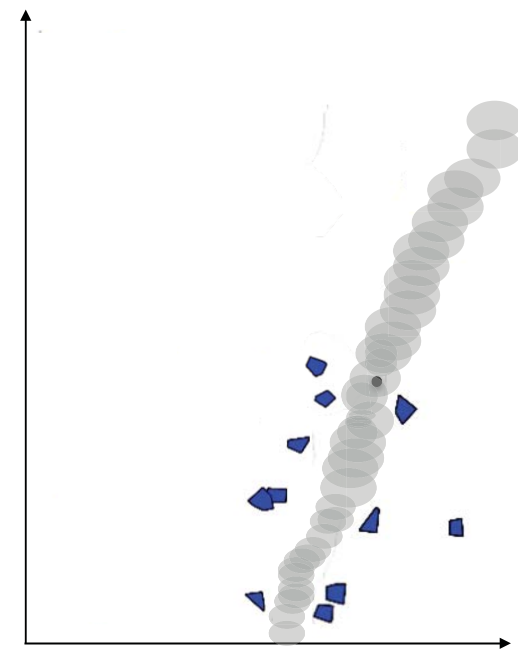
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. **Apply control corresponding to P and x**
7. Goto 2



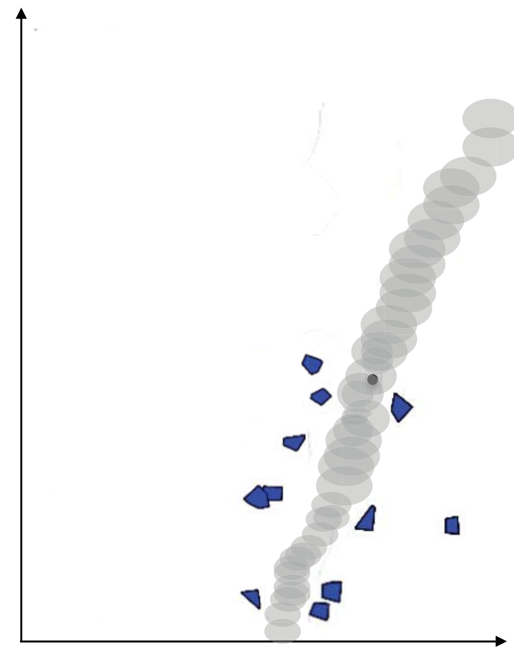
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. **Goto 2**



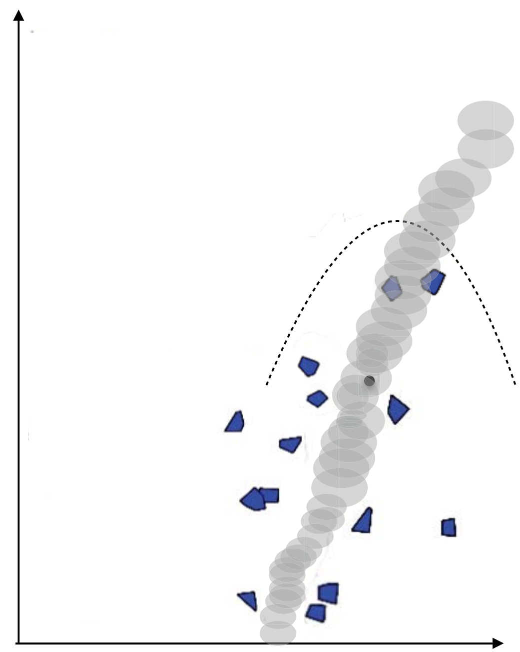
Online planning

1. Initial planned funnel sequence, P
2. **Update obstacles information, O**
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



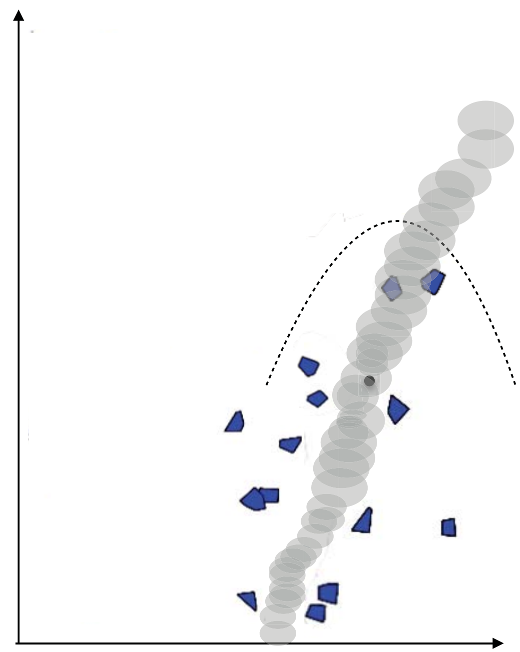
Online planning

1. Initial planned funnel sequence, P
2. **Update obstacles information, O**
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



Online planning

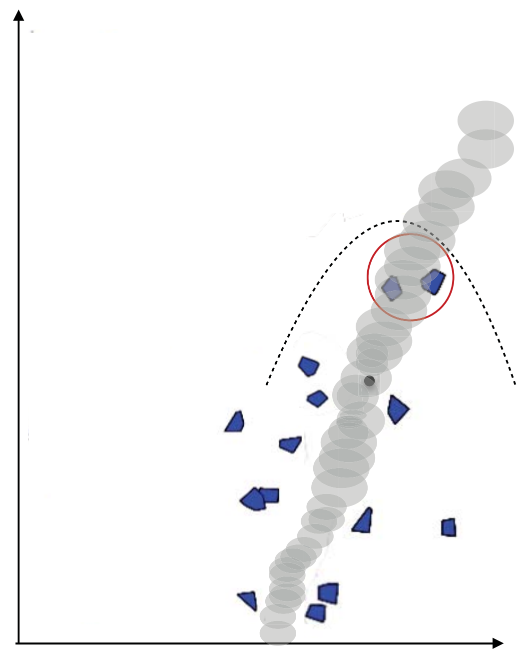
1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. **Get current state of robot, x**
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



x = (10,40)

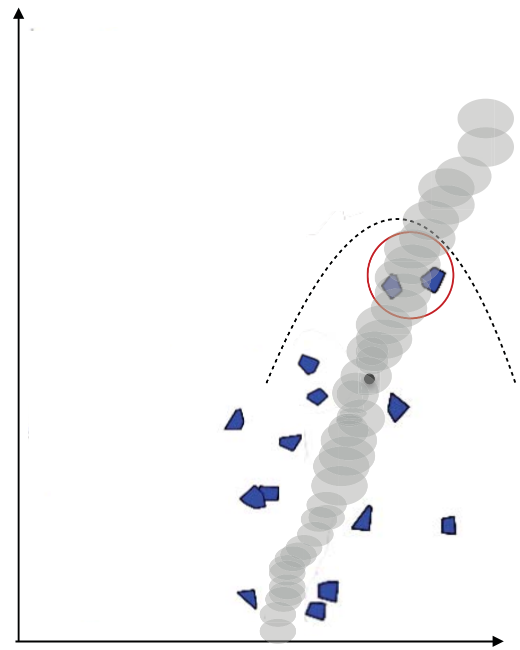
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. **Check if P collides with any of the obstacles**
5. If collision
    - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



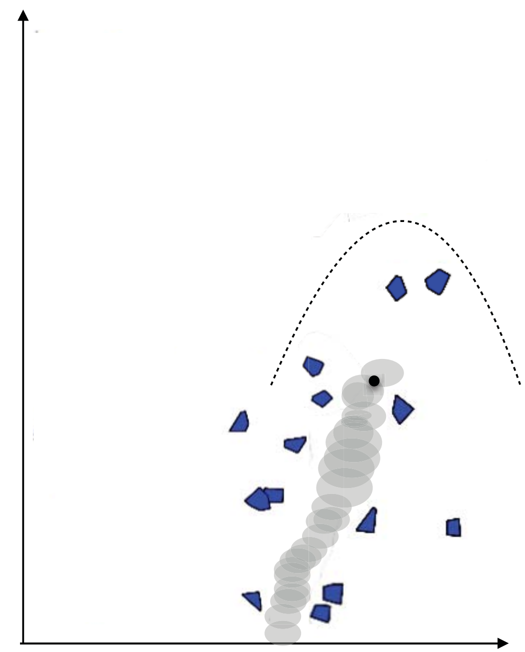
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



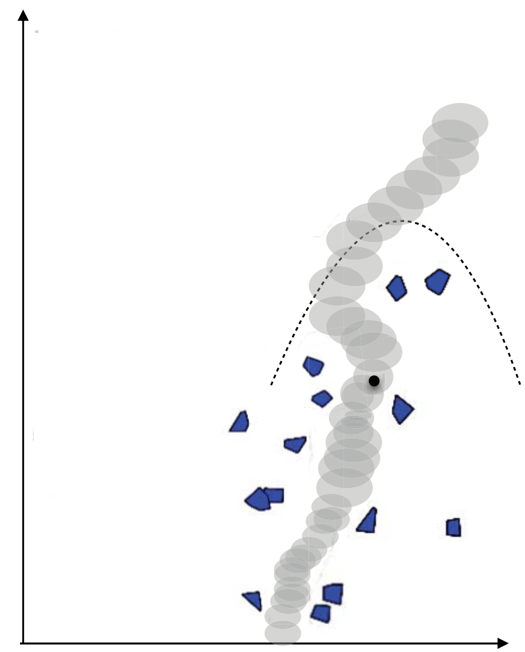
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



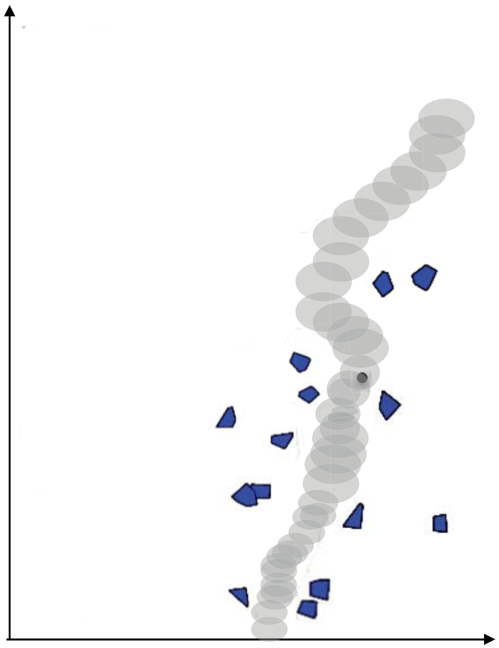
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



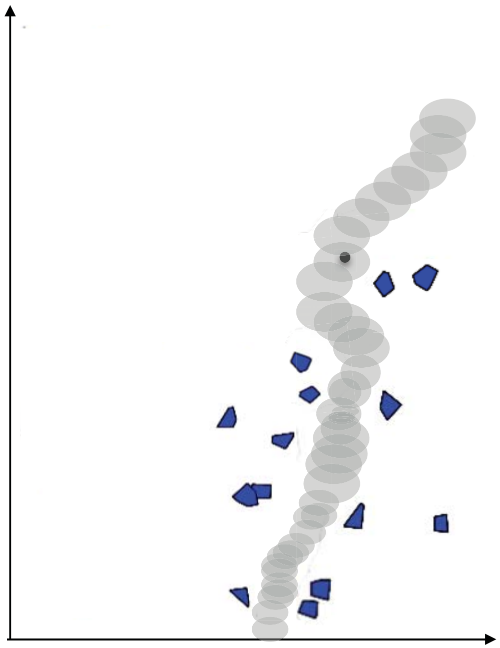
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. **Apply control corresponding to P and x**
7. Goto 2



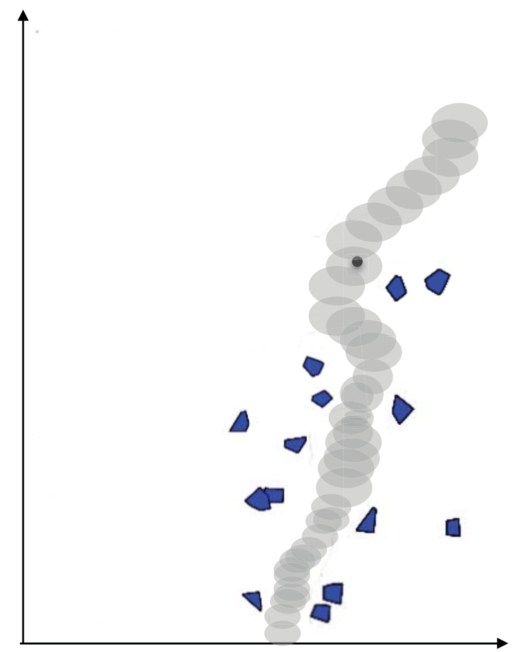
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. **Goto 2**



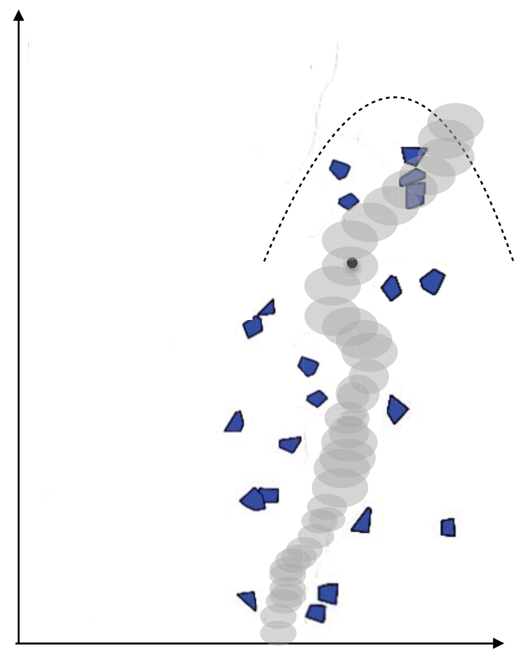
Online planning

1. Initial planned funnel sequence, P
2. **Update obstacles information, O**
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



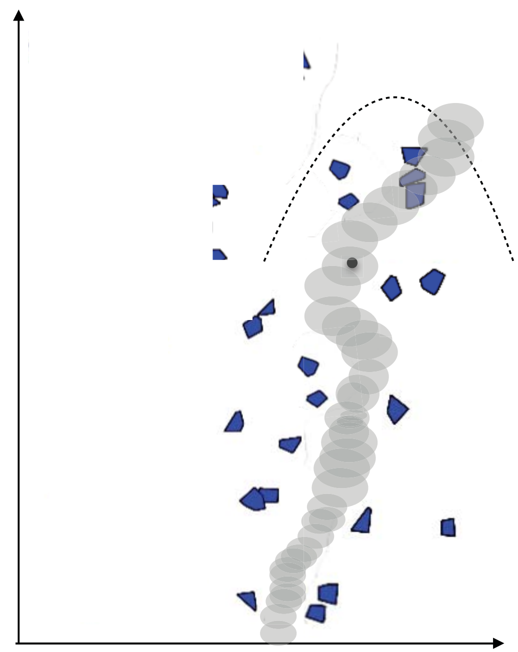
Online planning

1. Initial planned funnel sequence, P
2. **Update obstacles information, O**
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



Online planning

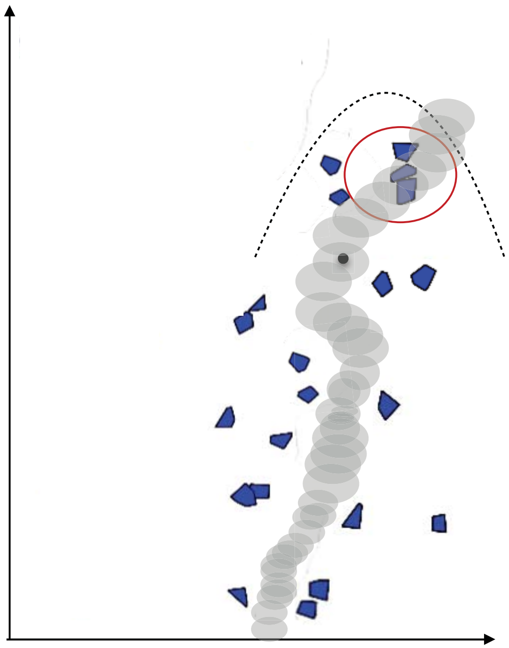
1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. **Get current state of robot, x**
4. Check if P collides with any of the obstacles
5. If collision
   - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



x = (8,60)

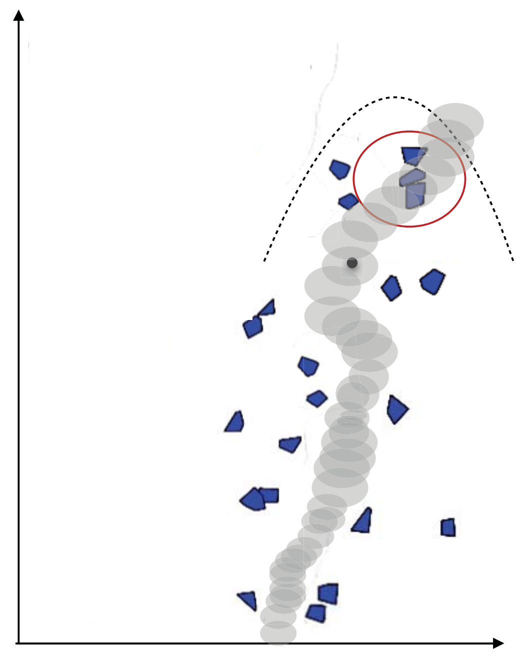
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. **Check if P collides with any of the obstacles**
5. If collision
    - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2



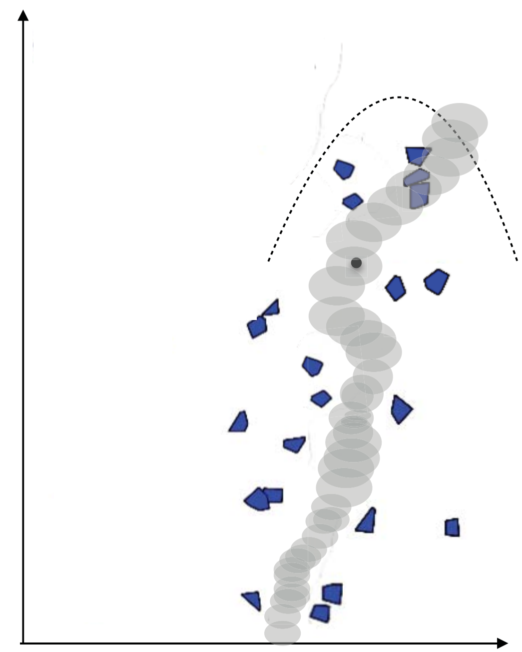
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



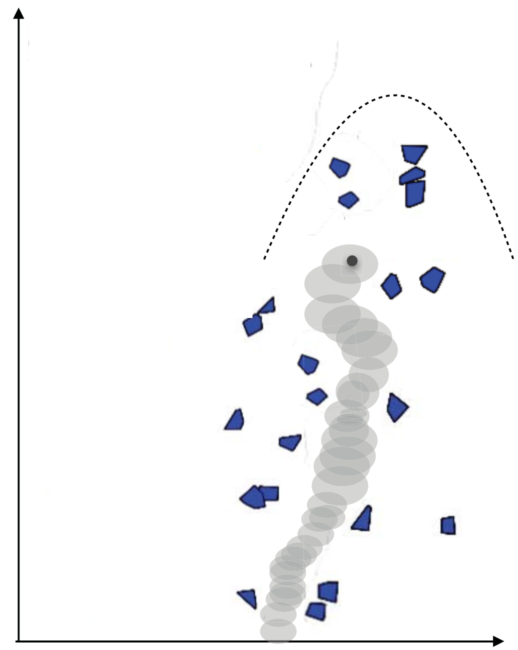
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
   - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



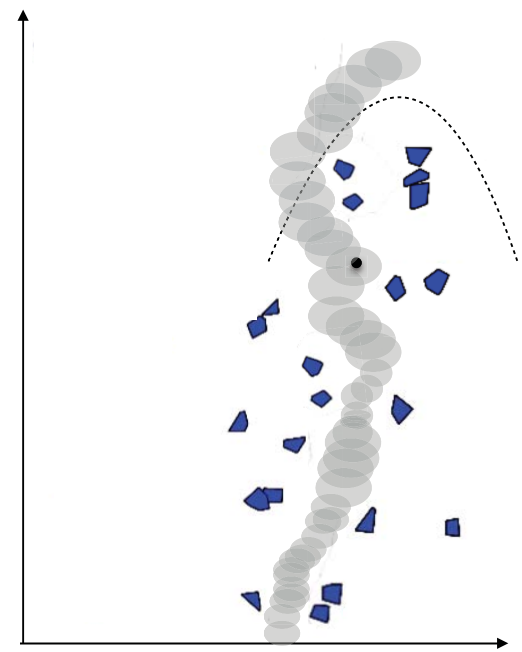
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



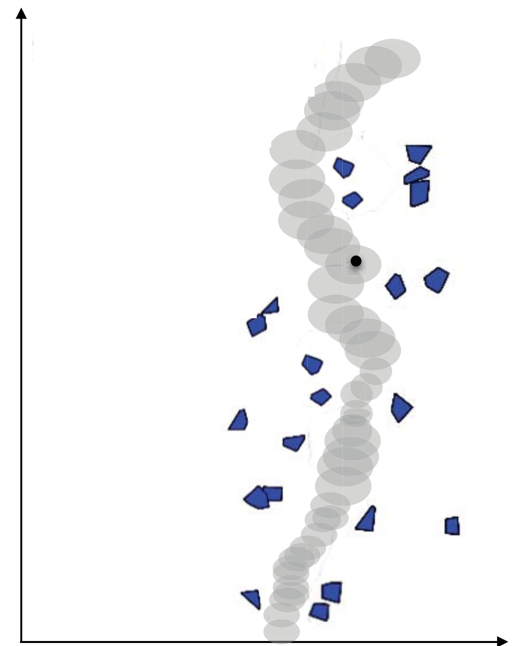
Online planning

1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - **P = ReplanFunnels(x, O)**
6. Apply control corresponding to P and x
7. Goto 2



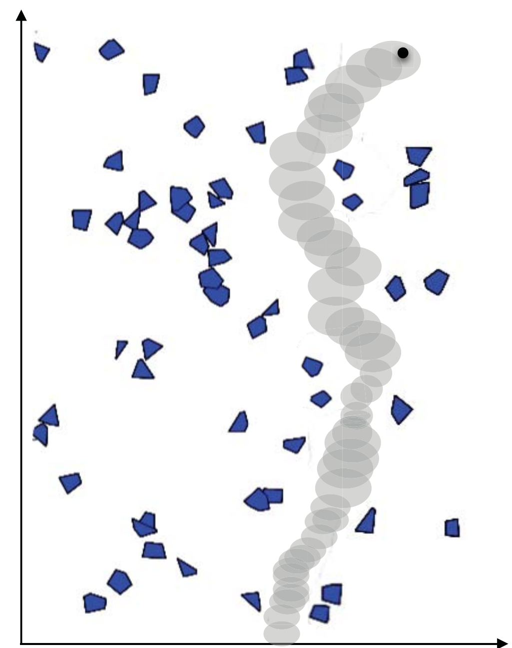
Online planning

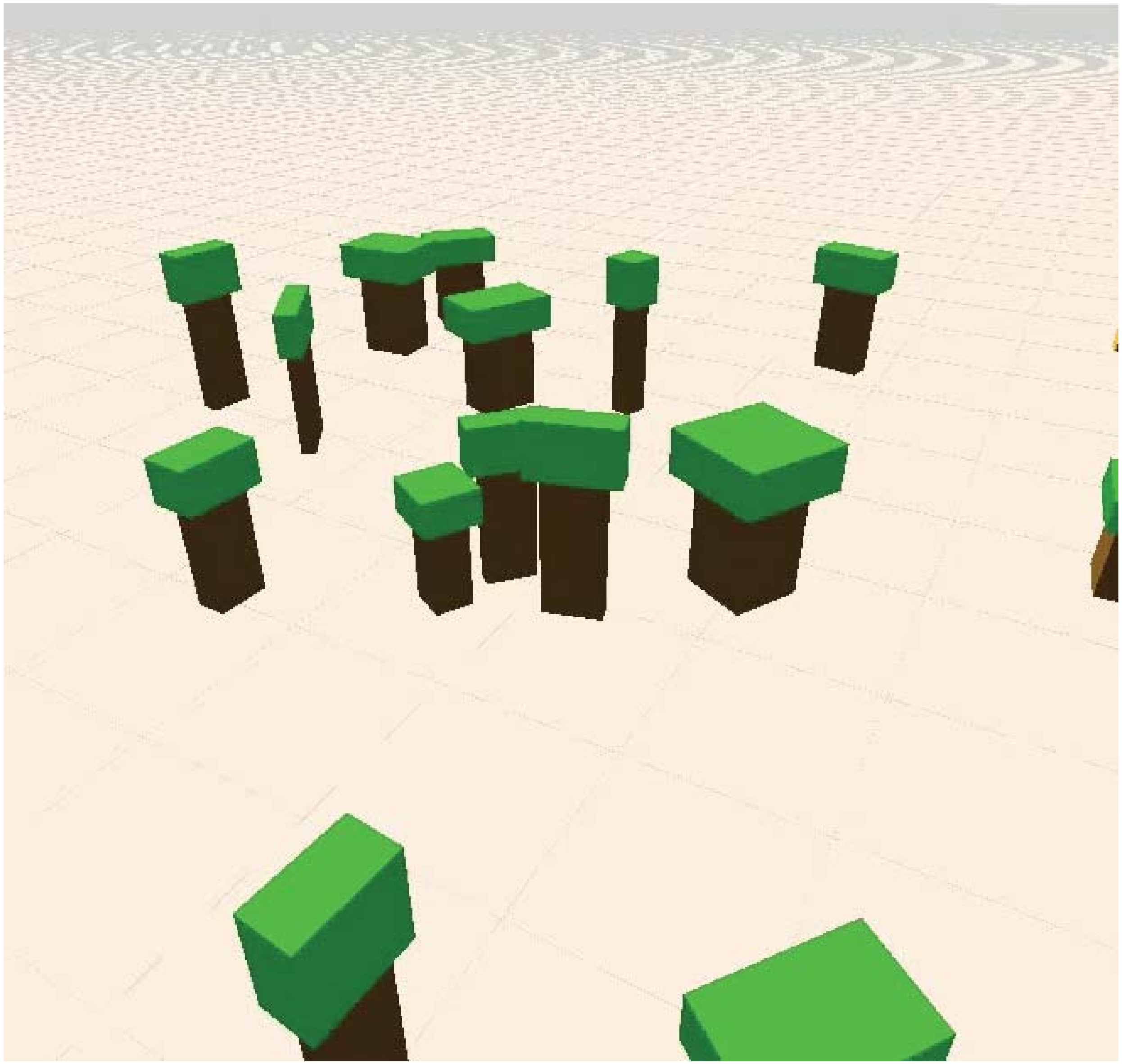
1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - P = ReplanFunnels(x, O)
6. **Apply control corresponding to P and x**
7. Goto 2

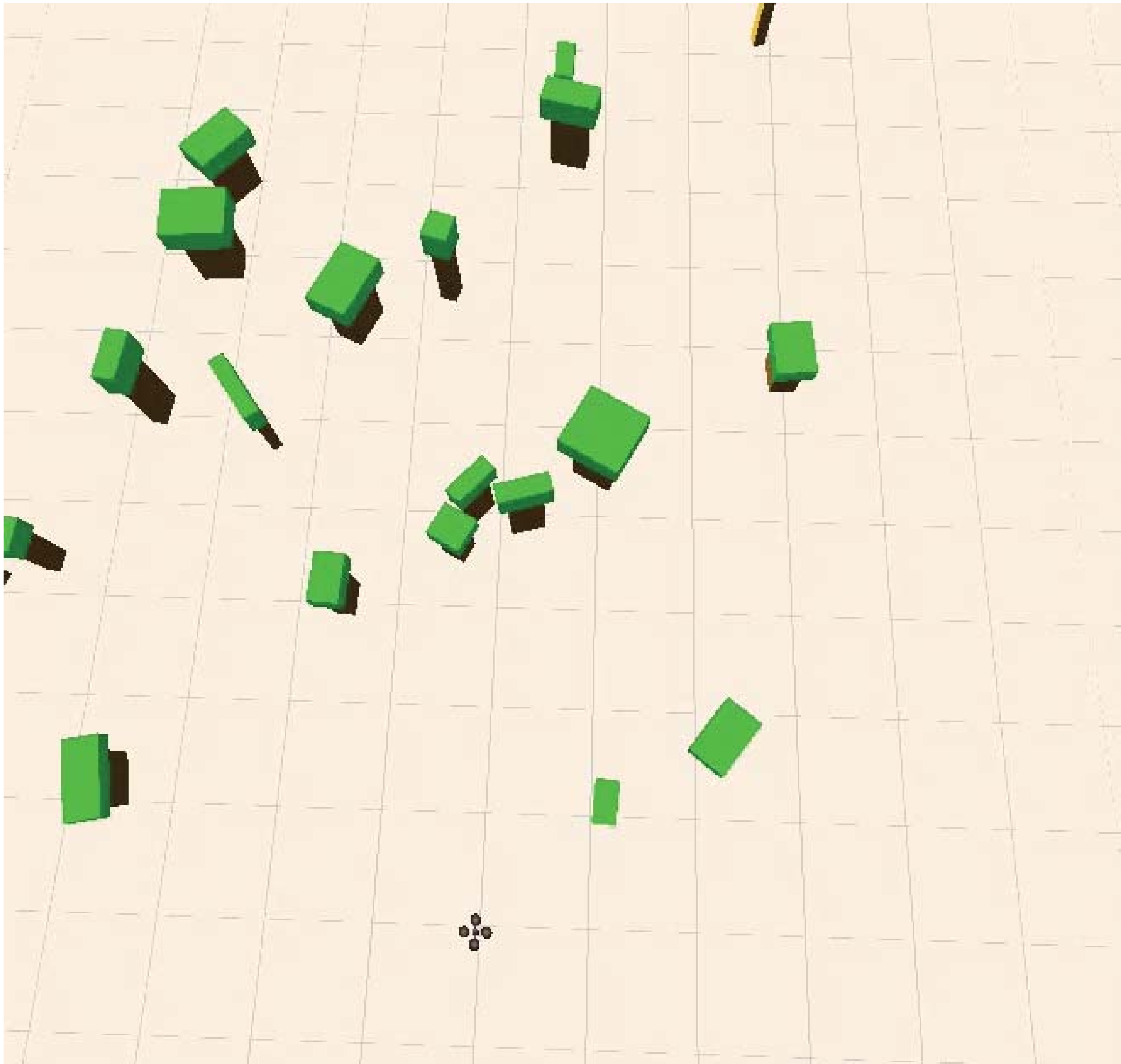


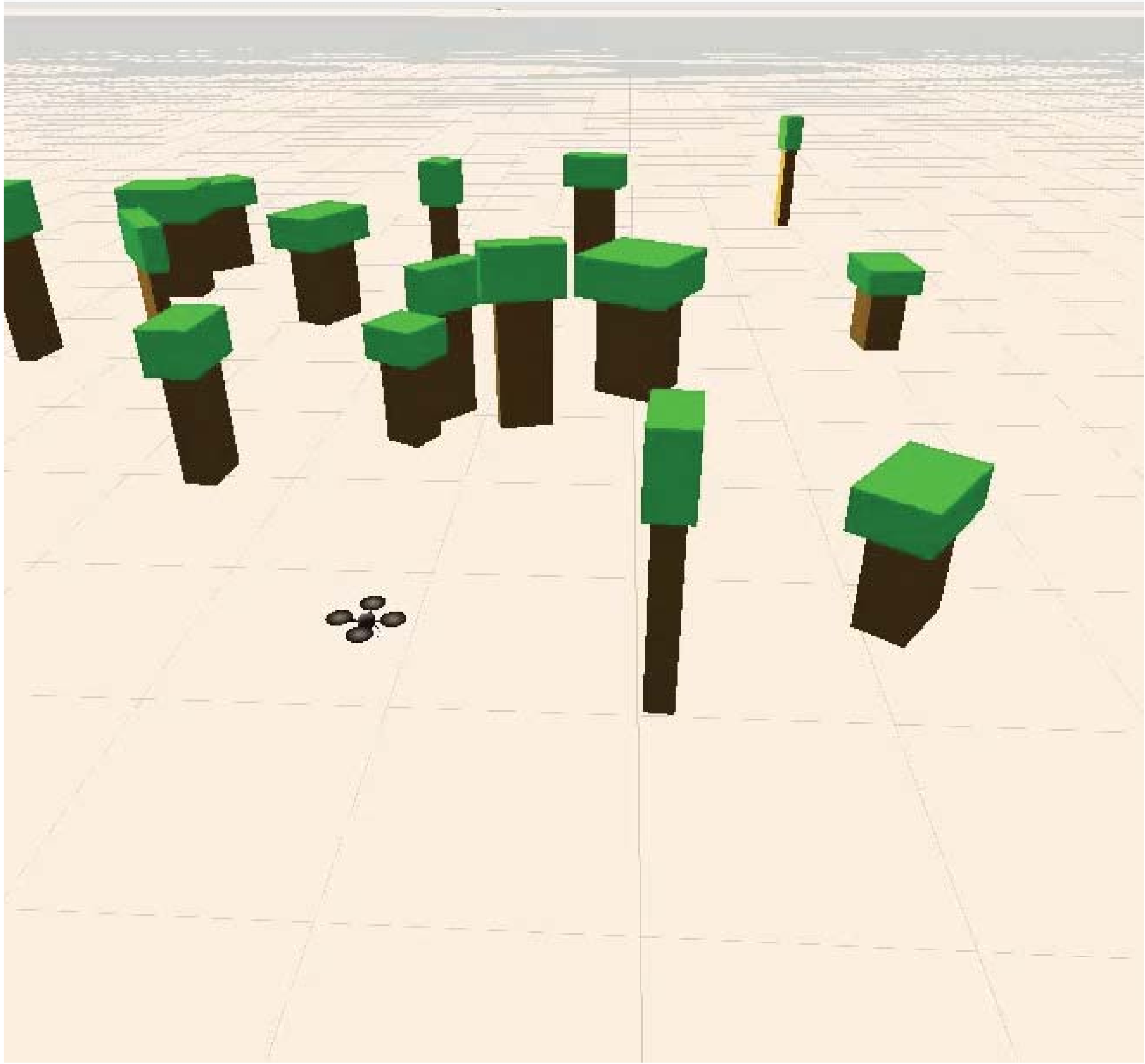
Online planning

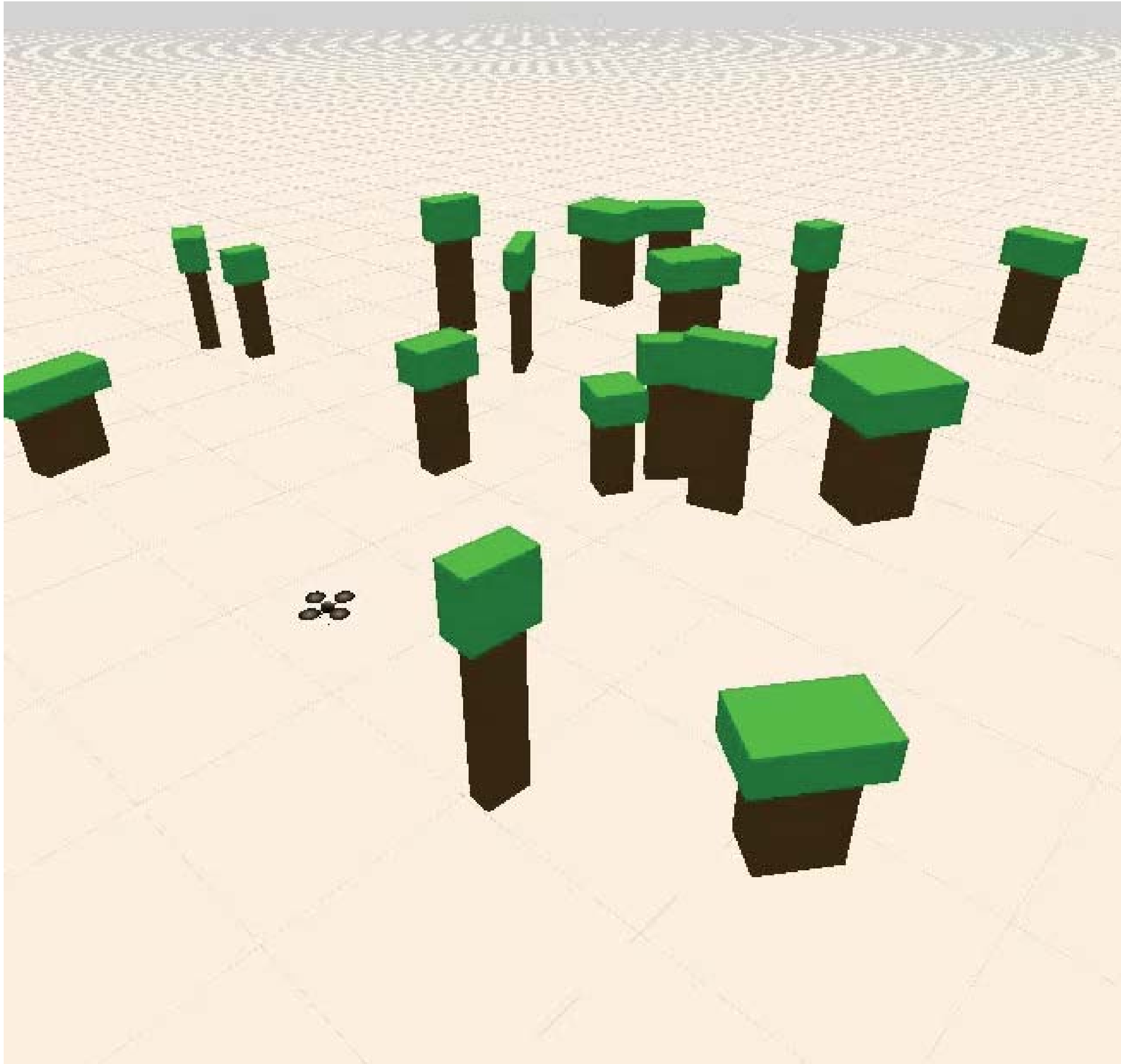
1. Initial planned funnel sequence, P
2. Update obstacles information, O
3. Get current state of robot, x
4. Check if P collides with any of the obstacles
5. If collision
    - P = ReplanFunnels(x, O)
6. Apply control corresponding to P and x
7. Goto 2







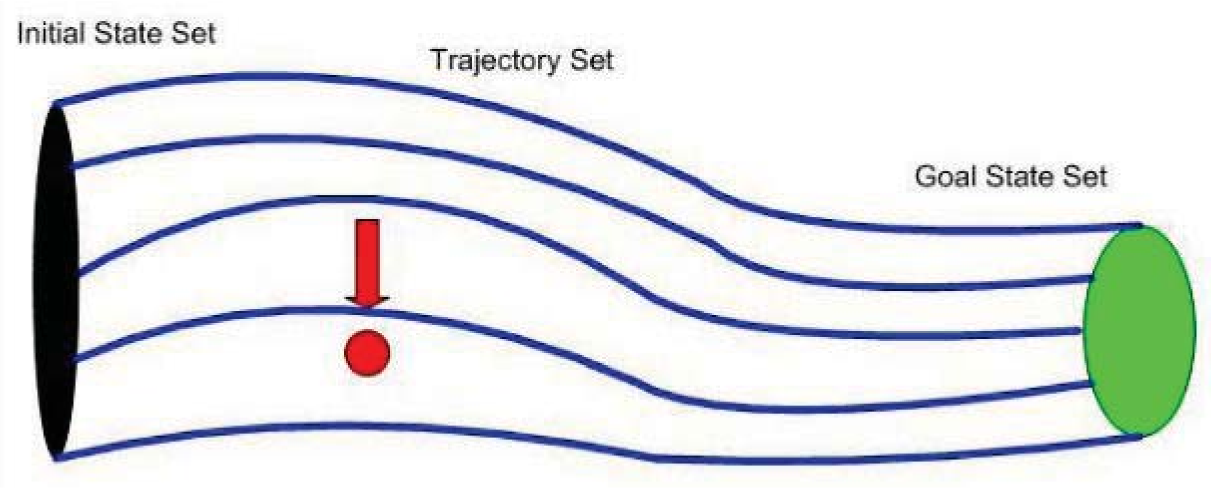




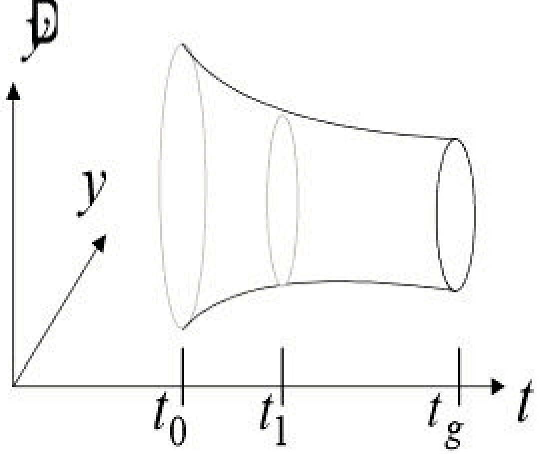
Outline

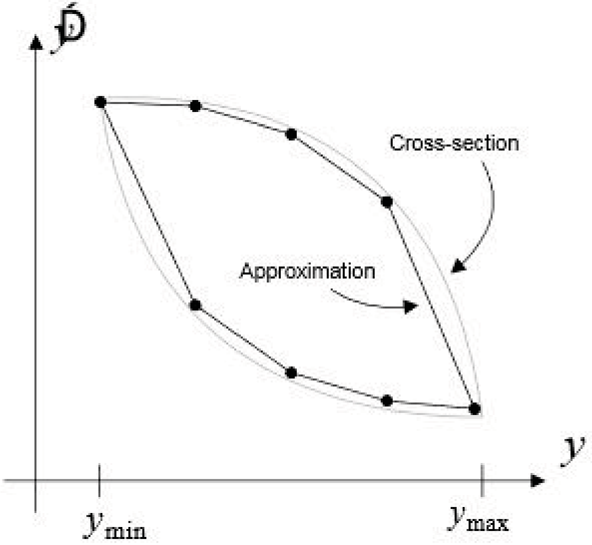
- Reachability and representing reach sets
- Applications - robust motion planning
- **Computing reach sets**
  - **Flow Tubes**
  - Funnels

Flow Tubes from Trajectories



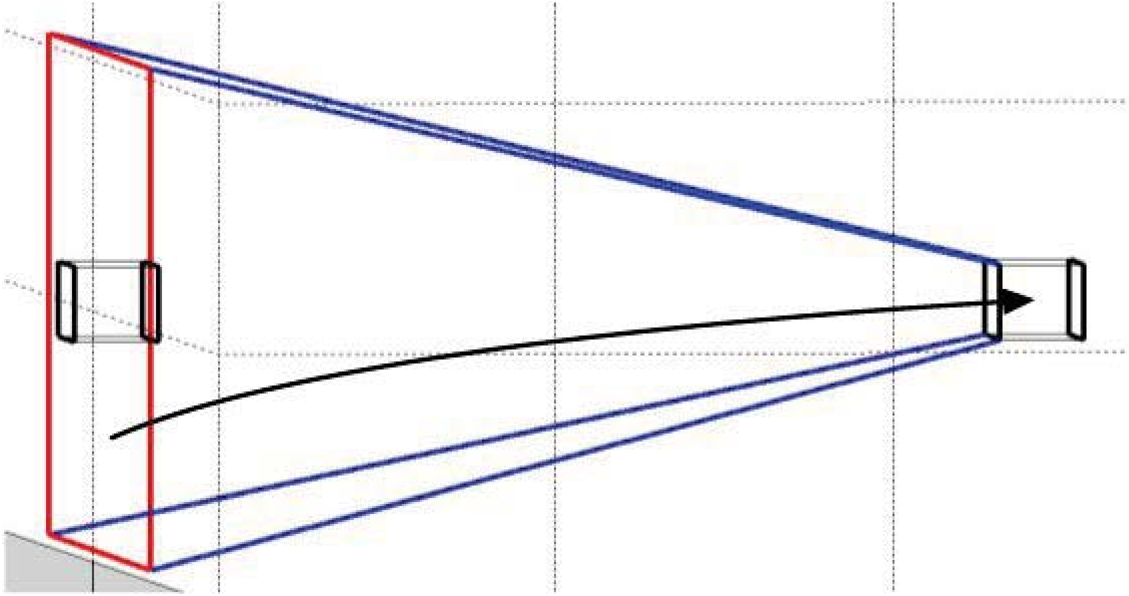
Flow Tube Approximations





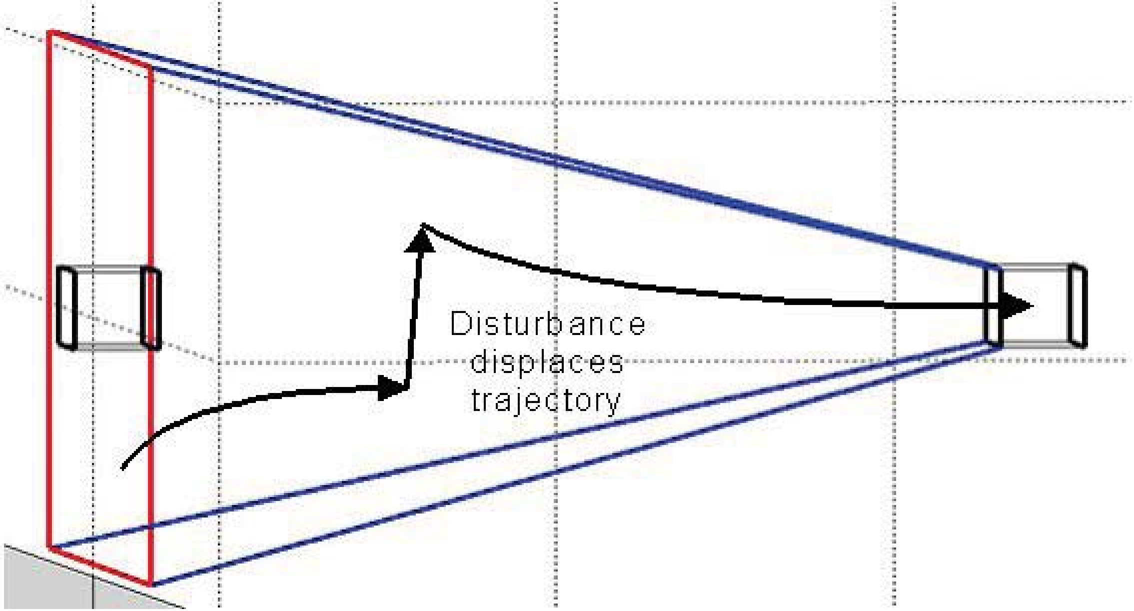
Polytopes, Ellipsoids, Rectangles used for cross section inner approximations

Robust Planning with Flow Tubes



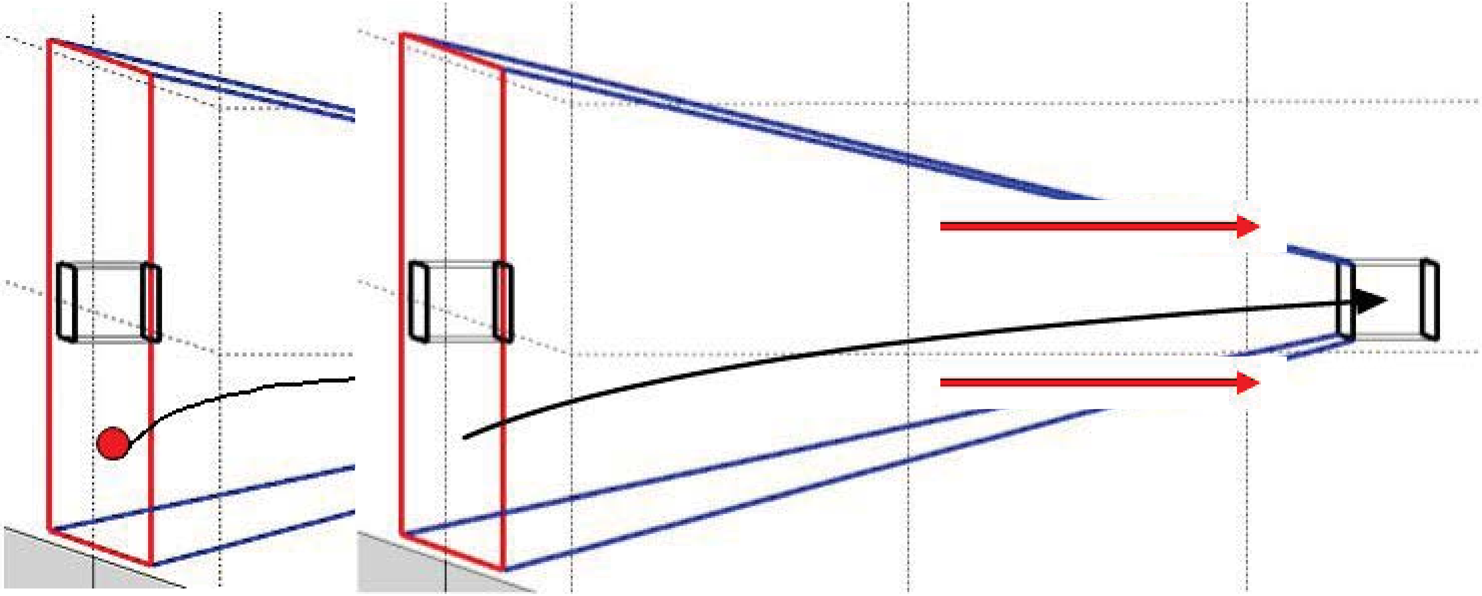
Plan a trajectory from initial to goal state

Robust Planning with Flow Tubes



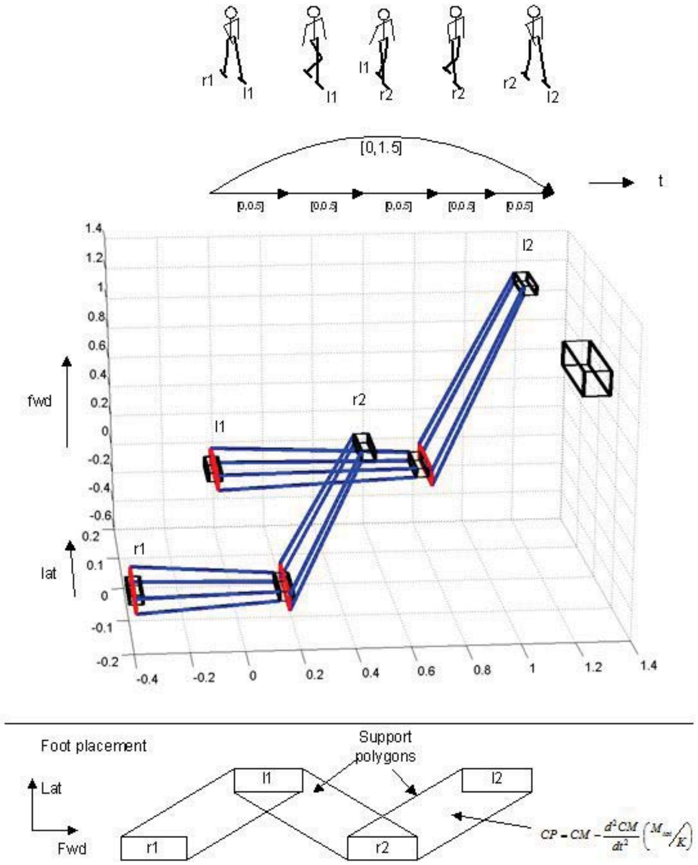
Robust to disturbances within the flow tube

Robust Planning with Flow Tubes

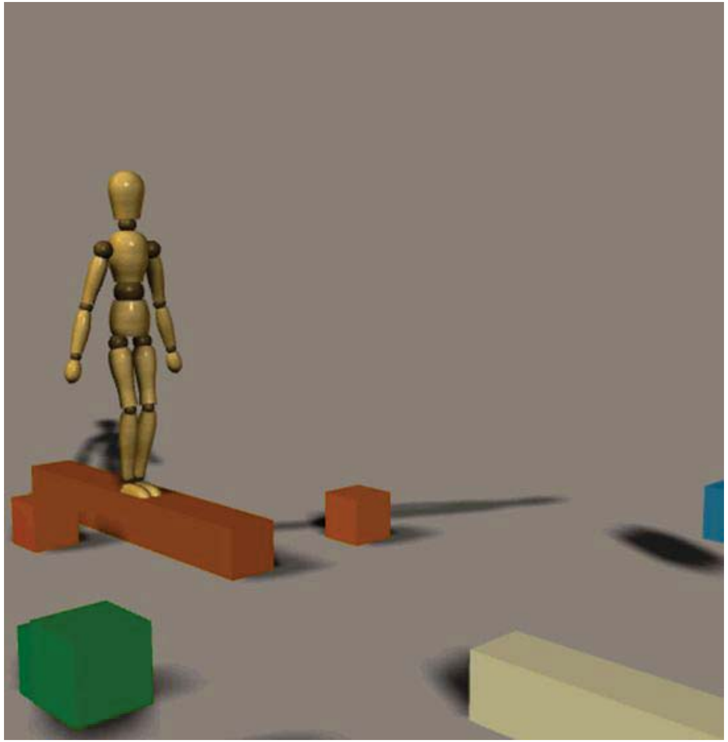


Framework allows temporal planning between flow tubes

Humanoid Footstep Planning with Flow Tubes



Humanoid Footstep Planning with Flow Tubes

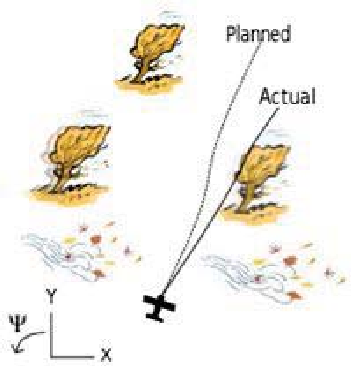


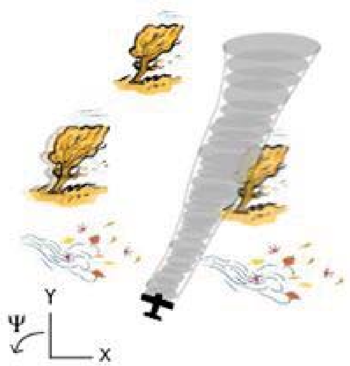
Outline

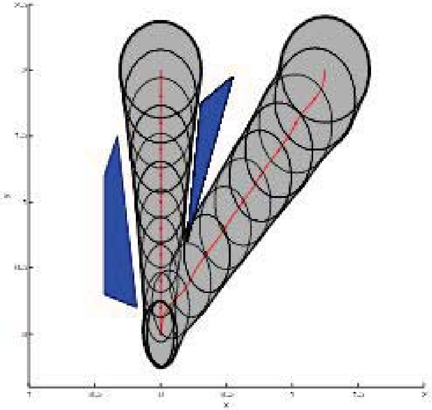
- Reachability and representing reach sets
- Applications - robust motion planning
- **Computing reach sets**
  - Flow Tubes
  - **Funnels**

Understanding Funnels

- **Goal:** find the region that guarantees safety under the given bounded uncertainty
- Funnels are composed regions of finite time invariance around a trajectory for all time.
- In practice, tradeoff between guarantees and computation time
- **Benefit:**





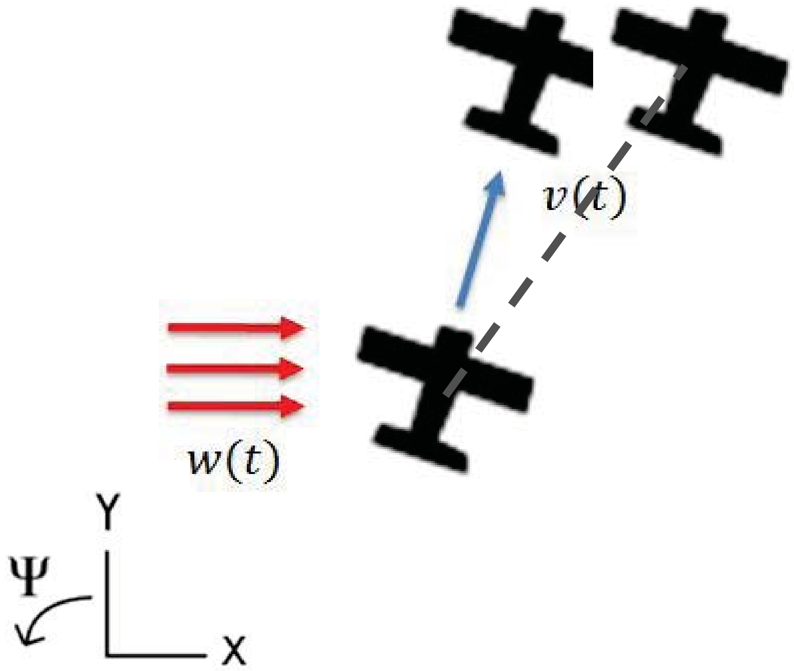


Funnel Computing Example: System Model

- System Model

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ \psi \\ \dot{\psi} \end{bmatrix}, \quad \dot{\mathbf{x}} = \begin{bmatrix} -v(t)\cos\psi \\ v(t)\sin\psi \\ \dot{\psi} \\ u \end{bmatrix} + \begin{bmatrix} w(t) \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

$$\dot{x} = f\big(x, t, w(x,t)\big)$$



- Bounded Uncertainty

$v(t) \in [9.5, 10.5] m/s$

$w(t) \in [-0.3, 0.3 m/s]$

Anirudha Majumdar and Russ Tedrake, "Robust Online Motion Planning with Regions of Finite Time Invariance"

Funnel Computing Example: Nominal Trajectory

- Nominal Trajectory

$x_i(0)$ and $x_i(T_i)$

$$J = \int_0^{T_i} [1 + u_0(t)^T R(t) u_0(t)]\, dt$$

- Optimal Control Law

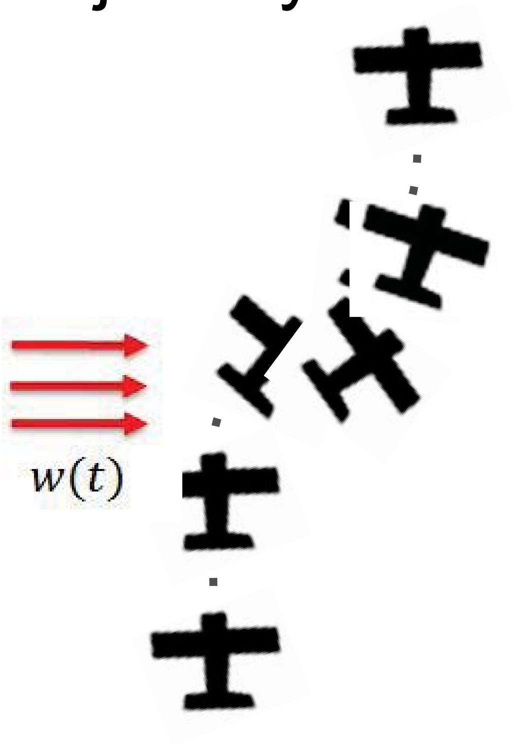
$$\dot{\bar{x}} \approx A_i(t)\bar{x}(t) + B_i(t)\bar{u}(t) + D_i(t)w(t)$$

$$\bar{x} = x - x_i(t)$$

$$\bar{u} = u - u_i(t)$$

$$\bar{u}^*(x,t) = -R^{-1}B_i(t)^T S_i(t)\bar{x}$$

$$-\dot{S}_i(t) = Q + S_i(t)A_i(t) + A_i(t)^T S_i(t) - S_i(t)[B_i(t)R^{-1}B_i(t)^T - \frac{1}{\gamma^2}D_i(t)D_i(t)^T]S_i(t)$$



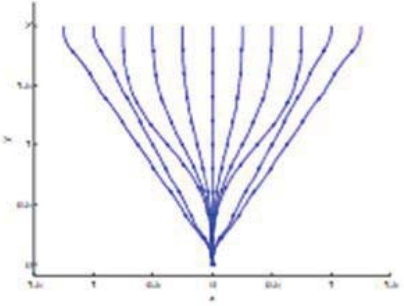
Funnel Computing Example: Ellipse

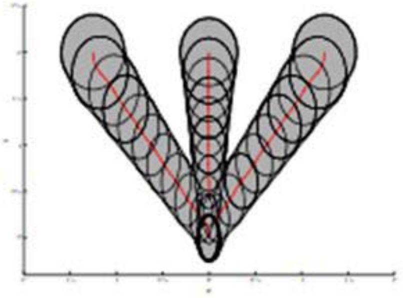
- Computing Funnel

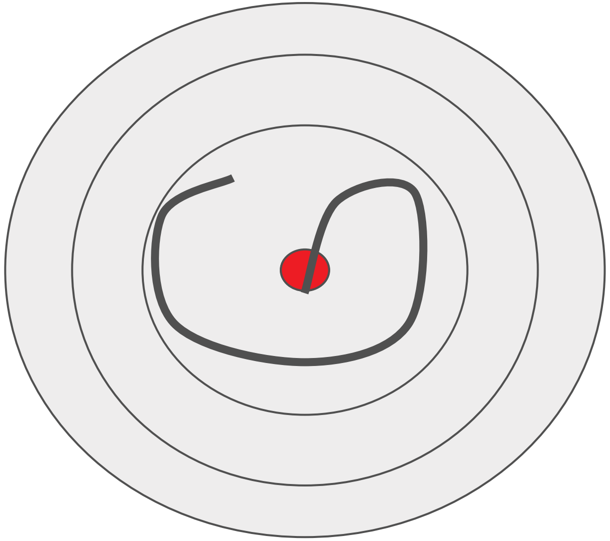
$$V_i(x,t) = (x - x_i(t))^T S_i(t)(x - x_i(t))$$

$$X_0 = \{x \mid V(x,t) \leq \rho(0)\}$$

$$\begin{aligned} \underset{\rho(0),\tau}{\text{minimize}} \quad & \rho(0) \\ \text{subject to} \quad & \rho(0) - V(x,0) + \tau(V_{des}(x) - 1) \geq 0 \\ & \tau \geq 0 \end{aligned}$$







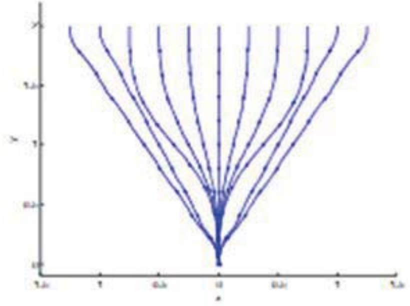
Funnel Computing Example: Ellipse

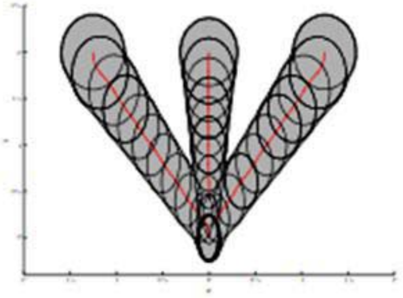
- Computing Funnel

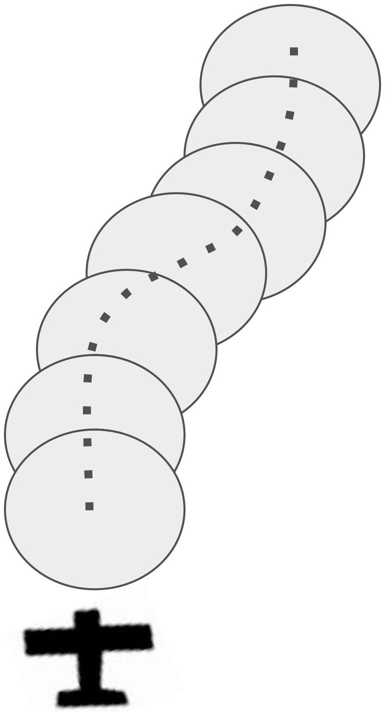
$$V_i(x,t) = (x - x_i(t))^T S_i(t)(x - x_i(t))$$

$$X_0 = \{x | V(x,t) \leq \rho(0)\}$$

$$\begin{aligned} \underset{\rho(0),\tau}{\text{minimize}} \quad & \rho(0) \\ \text{subject to} \quad & \rho(0) - V(x,0) + \tau(V_{des}(x) - 1) \geq 0 \\ & \tau \geq 0 \end{aligned}$$





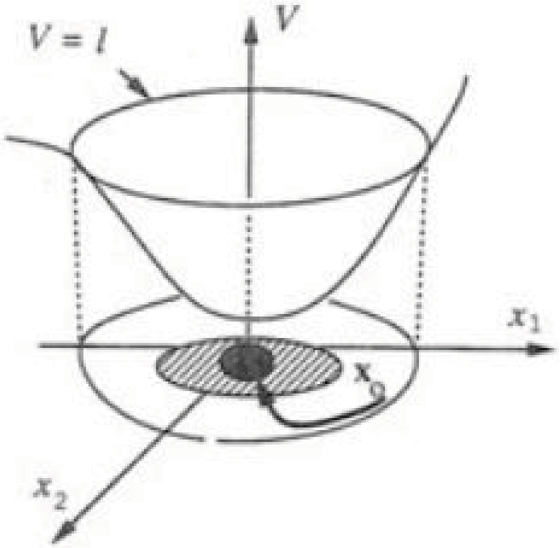


Importance of Lyapunov Functions

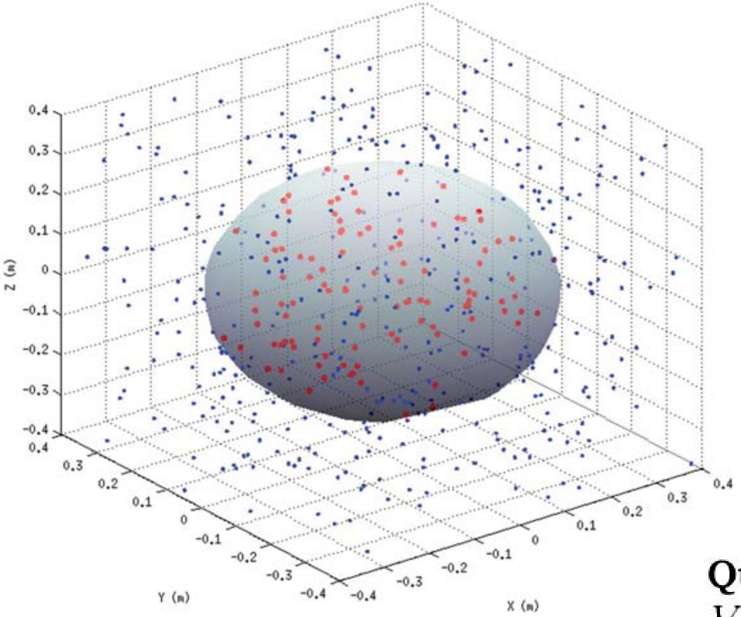
Used to verify stability of a system

V is positive definite

$\dot{V}(z) < 0$ for all $z \neq 0$, $\dot{V}(0) = 0$



Ellipsoid: Quadratic Lyapunov Functions



- Funnels defined by Quadratic Lyapunov Functions

- Evaluate the function at the points in the cloud

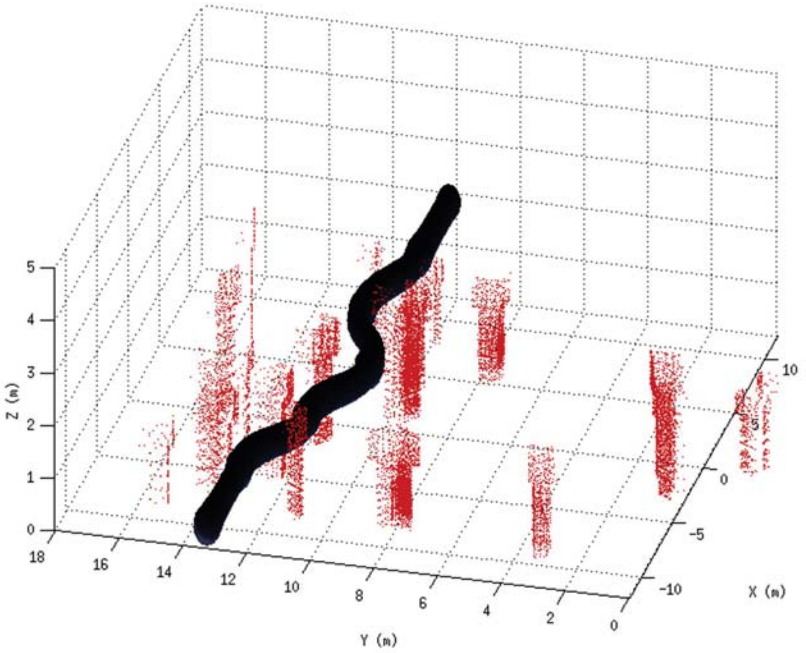
- To be out of collision, result must be greater than 1

Quadratic Lyapunov Function

$$V_p(\tilde{\mathbf{x}}) = \tilde{\mathbf{x}} S_p \tilde{\mathbf{x}} + S_{1p}\tilde{\mathbf{x}} + S_{2p} > 1$$

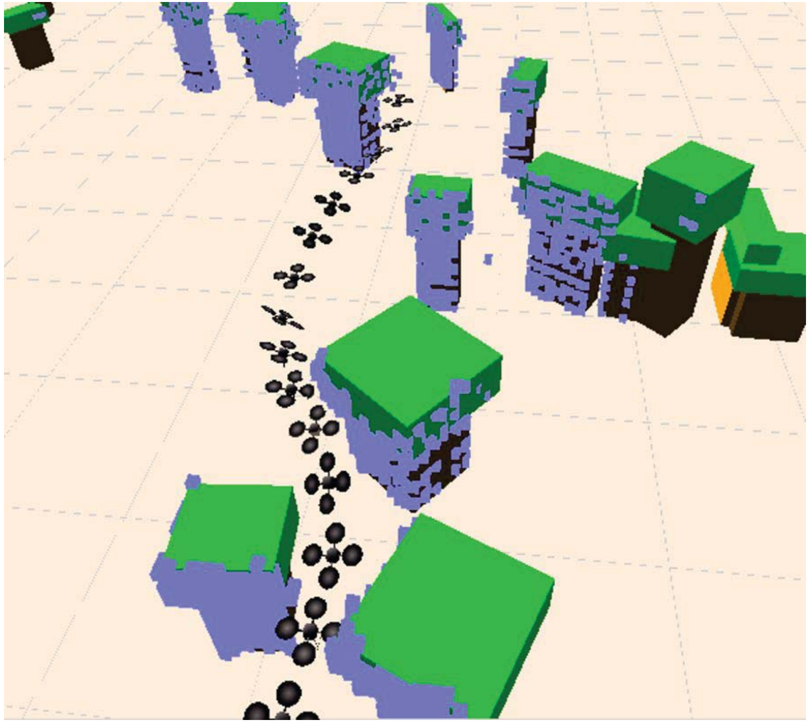
where $\tilde{\mathbf{x}} = \begin{Bmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{z} \end{Bmatrix} = \hat{\mathbf{x}} - \mathbf{x}_0$

Flying Through Forest: Path Planning



- Sequentially planned funnels in the sensed environment
- Path computed in increments of 5 meters, the length of each funnel

Flying Through Forest: Guaranteed Safety!



Algorithm 1 Online Planning

```
Initialize current planned funnel sequence, 𝒫 = {F_1, F_2, ..., F_n}
for t = 0, ... do
    𝒪 ⇐ Obstacles in sensor horizon
    x ⇐ Current state of robot
    Collision ⇐ Check if 𝒫 collides with 𝒪 by solving QPs (7)
    if Collision then
        𝒫 ⇐ ReplanFunnels(x, 𝒪)
    end if
    F.current ⇐ F_i ∈ 𝒫 such that x ∈ F_i
    t.internal ⇐ Internal time of F.current
    Apply control u_i(x, t.internal)
end for
```

References

- http://groups.csail.mit.edu/robotics-center/public_papers/Majumdar12a.pdf
- Bhatia, A. and Frazzoli E., Incremental Search Methods for Reachability Analysis of Continuous and Hybrid Systems
- Hoffman A. , Williams B., Exploiting Spatial and Temporal Flexibility for Plan Execution of Hybrid, Under-actuated Systems
- G. Bledt, A. Majumdar, A. J. Barry, R. Tedrake, "UAV Trajectory Planning and Optimization for High Speed Flight with Uncertainties Using Computer Vision Through Forests," Presented at the MIT Summer Research Program Poster Session, August 2014, Cambridge, MA.