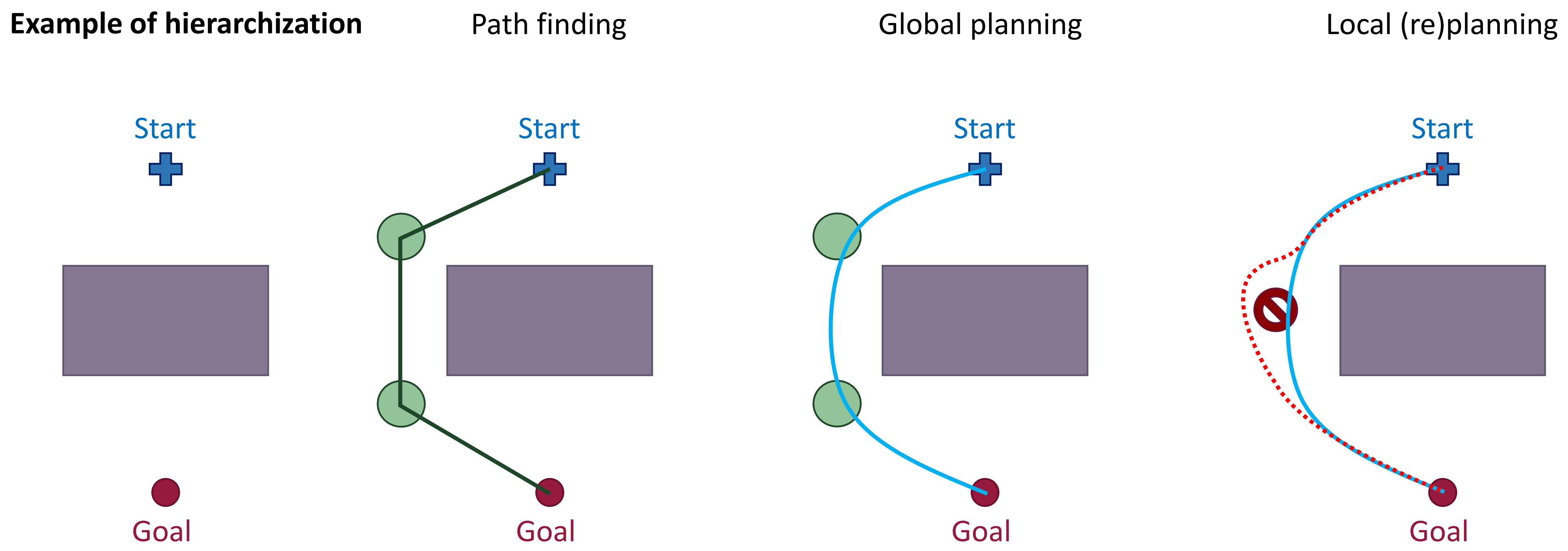
C 2.3 Path finding & following

Gauthier ROUSSEAU
gauthier.rousseau.cs@gmail.com

Global to local planning

Motion planning hierarchization



Case study 2

Mission details

Fire in a chemical plant

Toxic hazard prevent human intervention

Location of fire seats unknown

➡ **Locate and extinguish fire seats!**

Case study 2

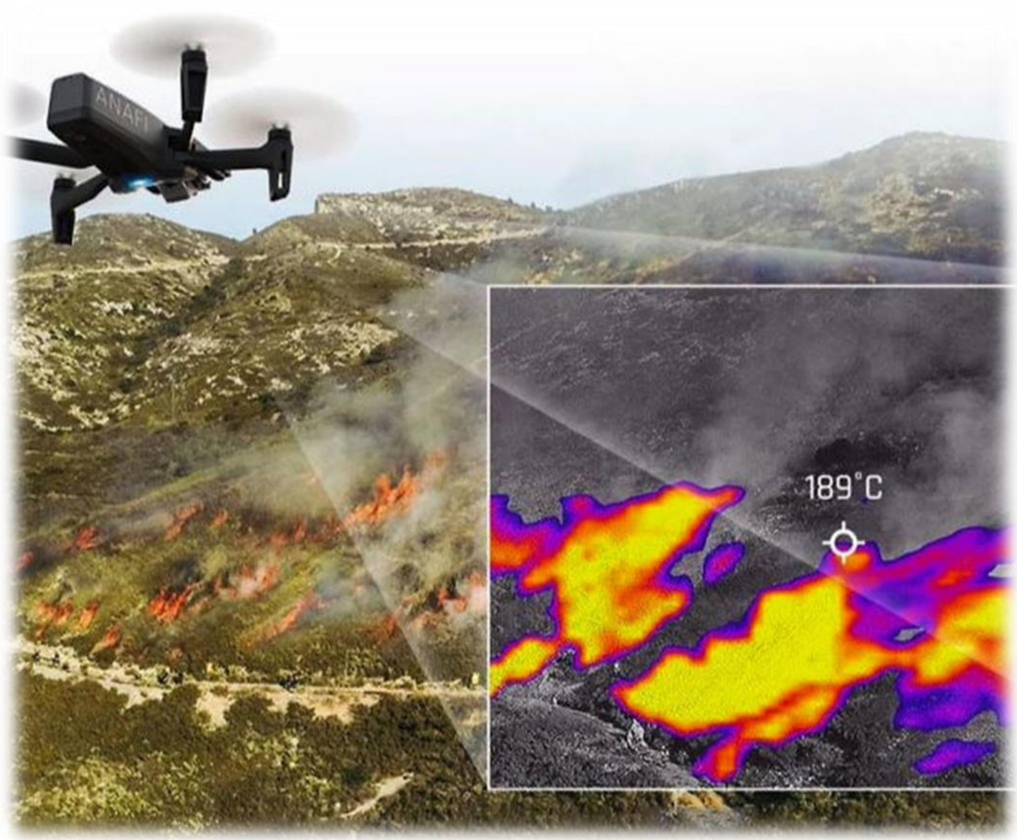
Equipment

Anafi thermal quadrotor

Quadrotor UAV

Dual camera visible/infrared

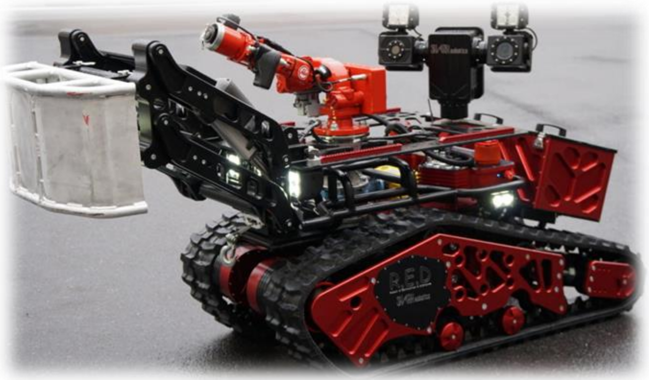
➡ Remote temperature measurements



Case study 2

Equipment

Colossus firefighting robot



Dubins car UGV

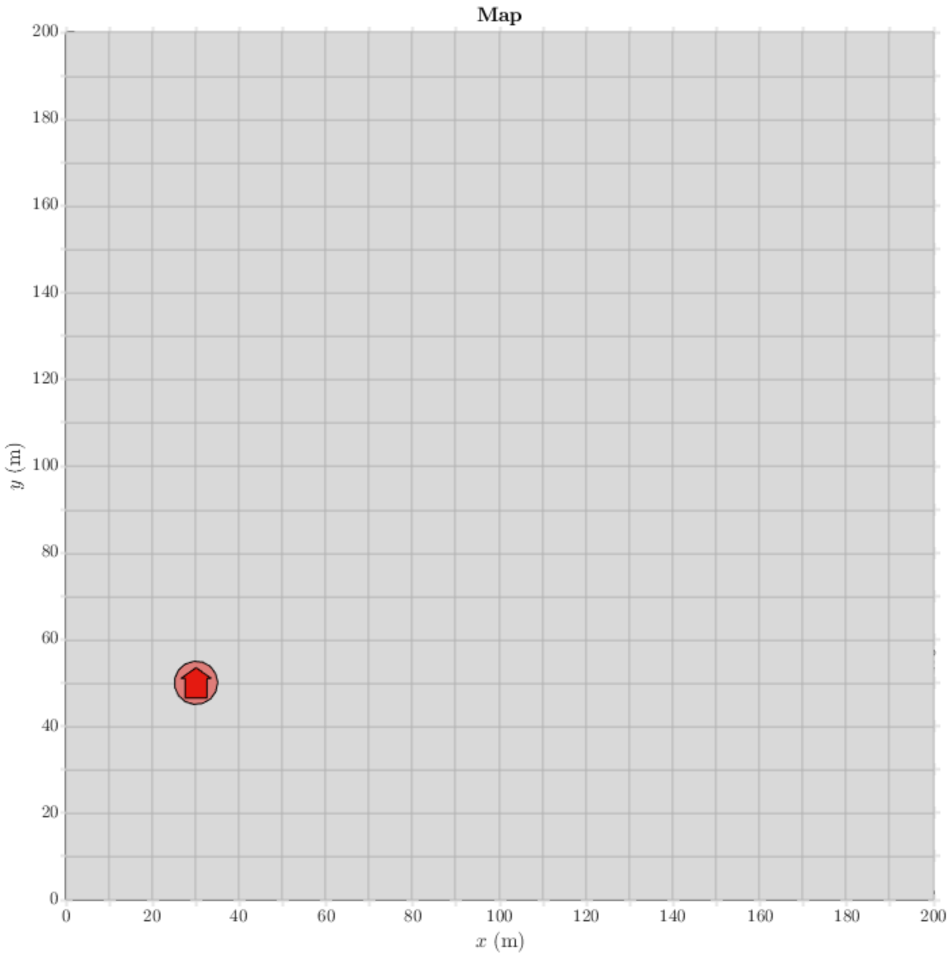
Water jet

➡ Remote fire seat management

Tasks

Task 1: find fire seats

➡ Scout map with UAV

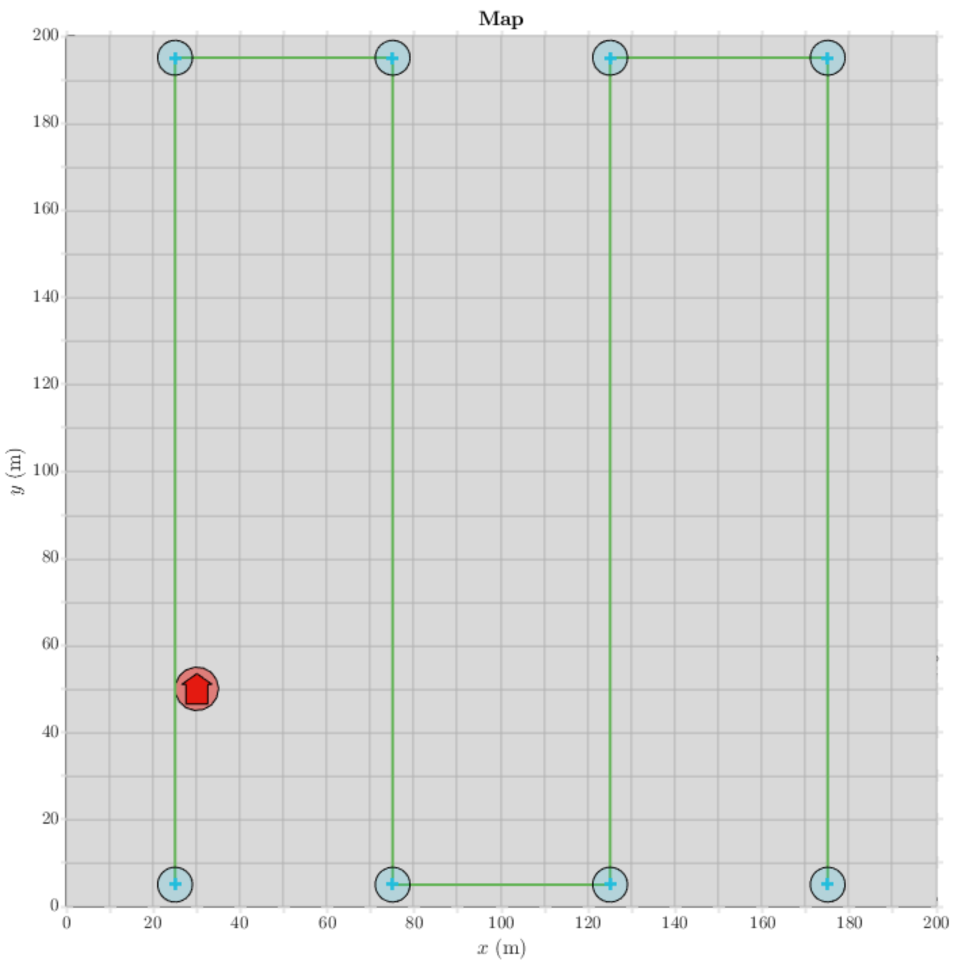


Mission

Tasks

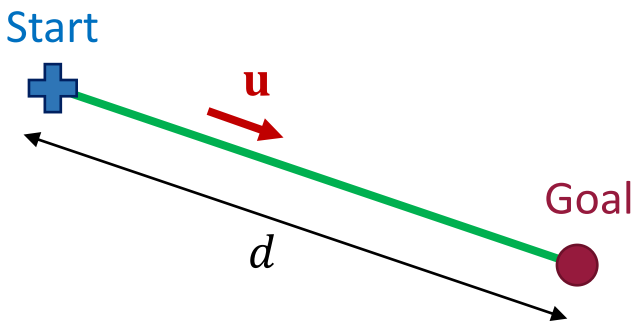
Task 1: find fire seats

➡ Scout map with UAV



Trajectory planning

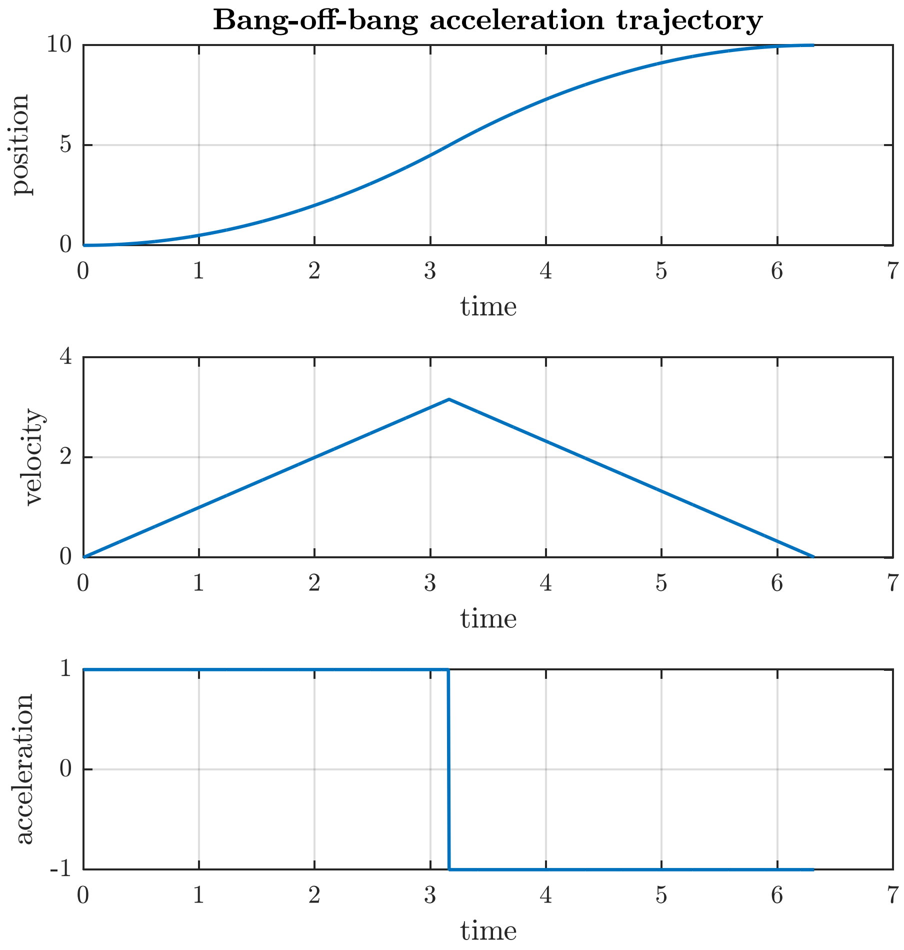
Bang-off-bang acceleration trajectory



$$\mathbf{a}_{\text{acc}} = a_{\max}\mathbf{u} \qquad \mathbf{a}_{\text{cruise}} = \mathbf{0} \qquad \mathbf{a}_{\text{brake}} = -a_{\max}\mathbf{u}$$

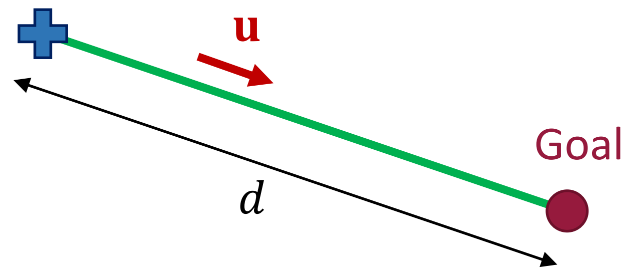
$$d \leq \frac{v_{ref}^2}{a_{\max}}$$

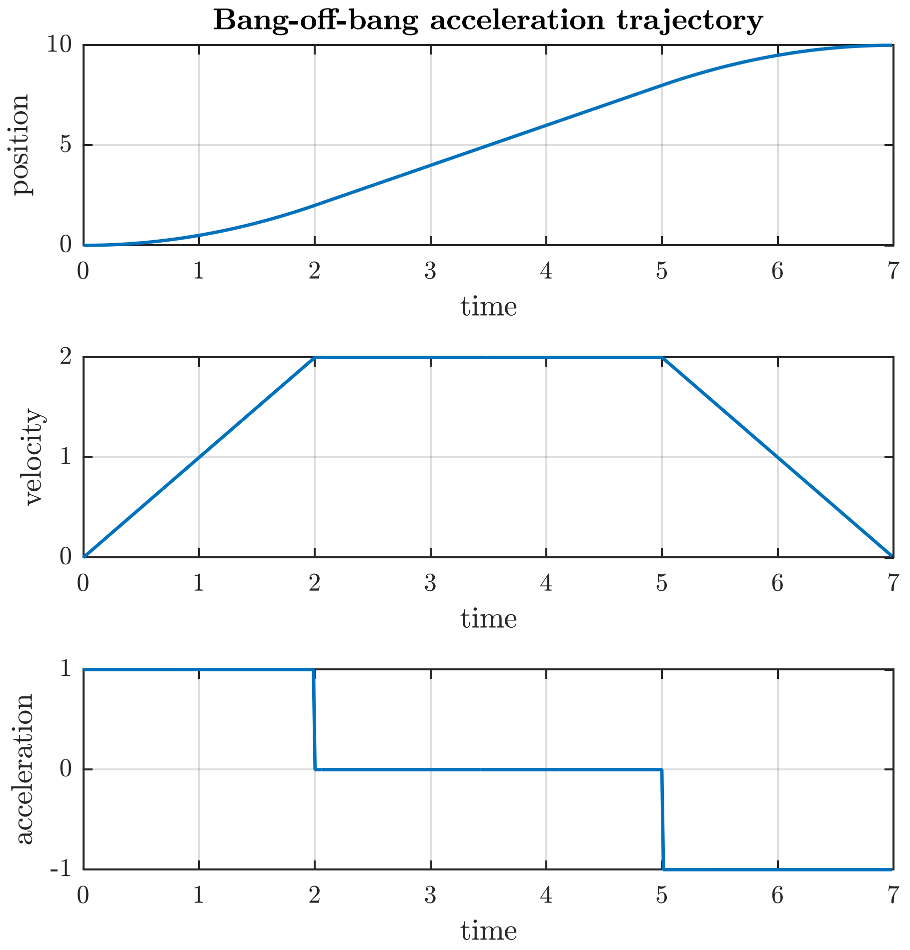
$$\Delta t_{\text{acc}} = \sqrt{\frac{d}{a_{\max}}} \qquad \Delta t_{\text{cruise}} = 0 \qquad \Delta t_{\text{brake}} = \sqrt{\frac{d}{a_{\max}}}$$



Trajectory planning

Bang-off-bang acceleration trajectory





$$\mathbf{a}_{acc} = a_{max}\mathbf{u} \qquad \mathbf{a}_{cruise} = \mathbf{0} \qquad \mathbf{a}_{brake} = -a_{max}\mathbf{u}$$

$$d > \frac{v_{ref}^2}{a_{max}}$$

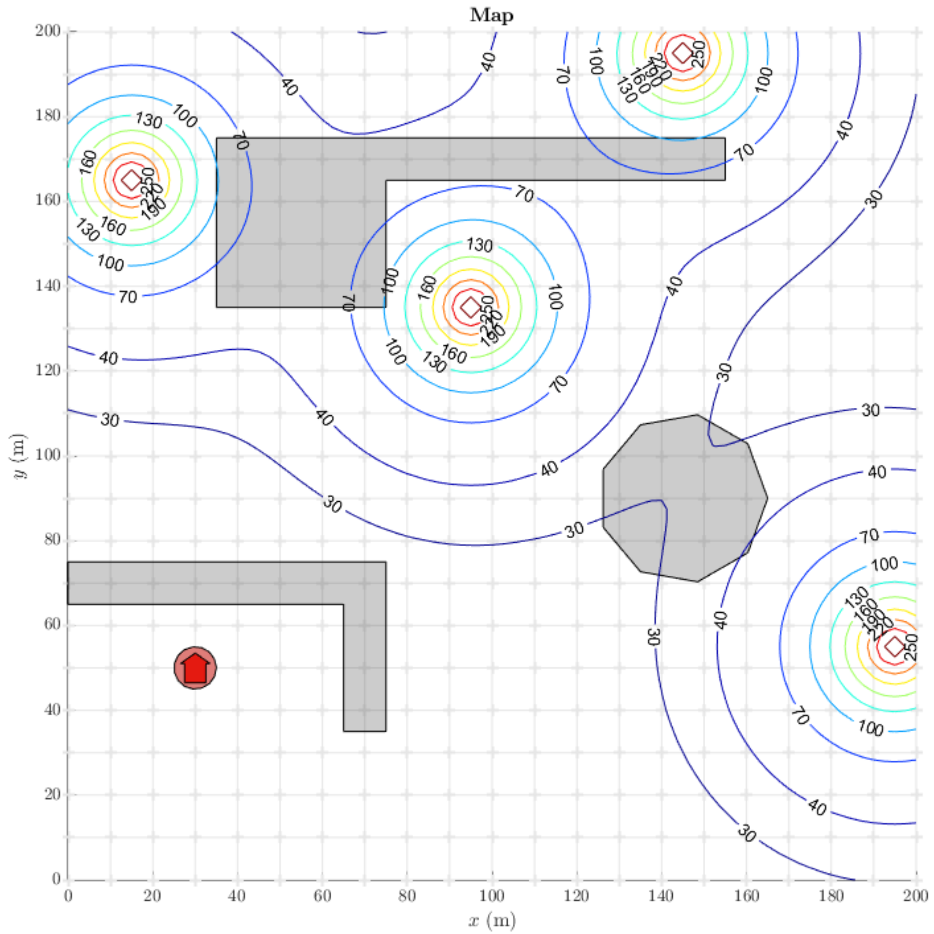
$$\Delta t_{acc} = \frac{v_{ref}}{a_{max}} \qquad \Delta t_{cruise} = \frac{d}{v_{ref}} - \frac{v_{ref}}{a_{max}} \qquad \Delta t_{brake} = \frac{v_{ref}}{a_{max}}$$

Mission

Tasks

Task 2: find path to fire seats

➡ Obstacle free path for UGV

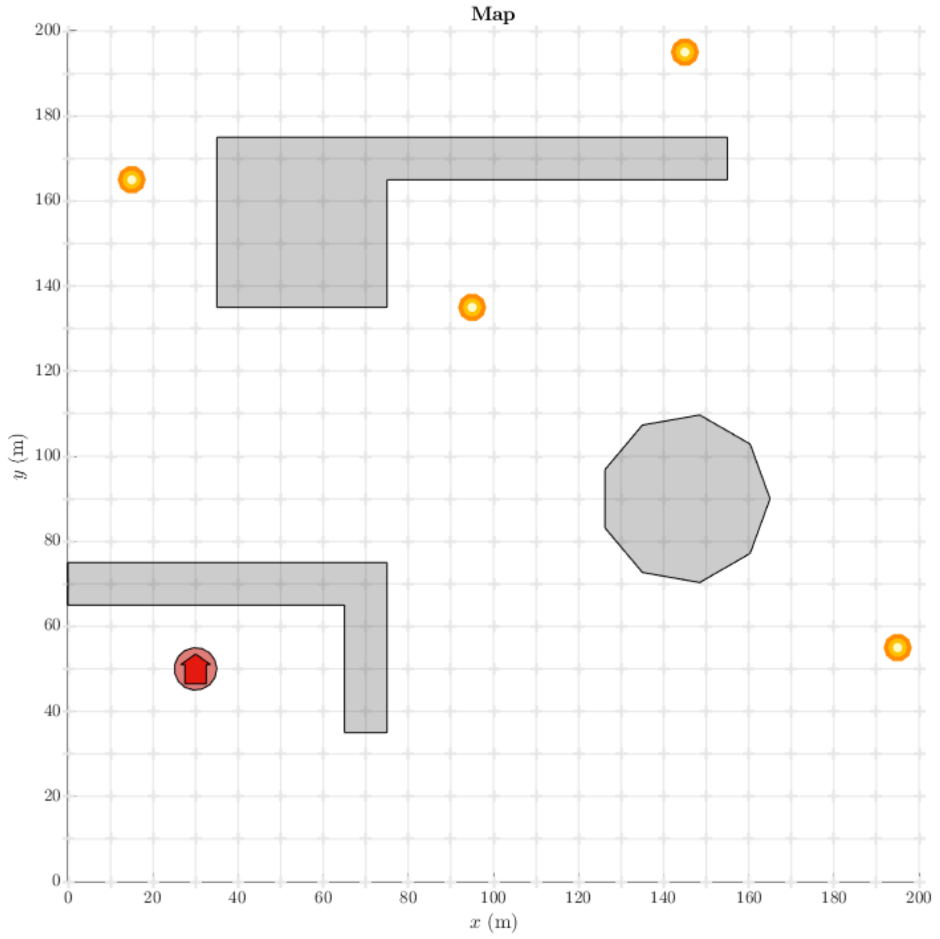


Mission

Tasks

Task 2: find path to fire seats

➡ Obstacle free path for UGV



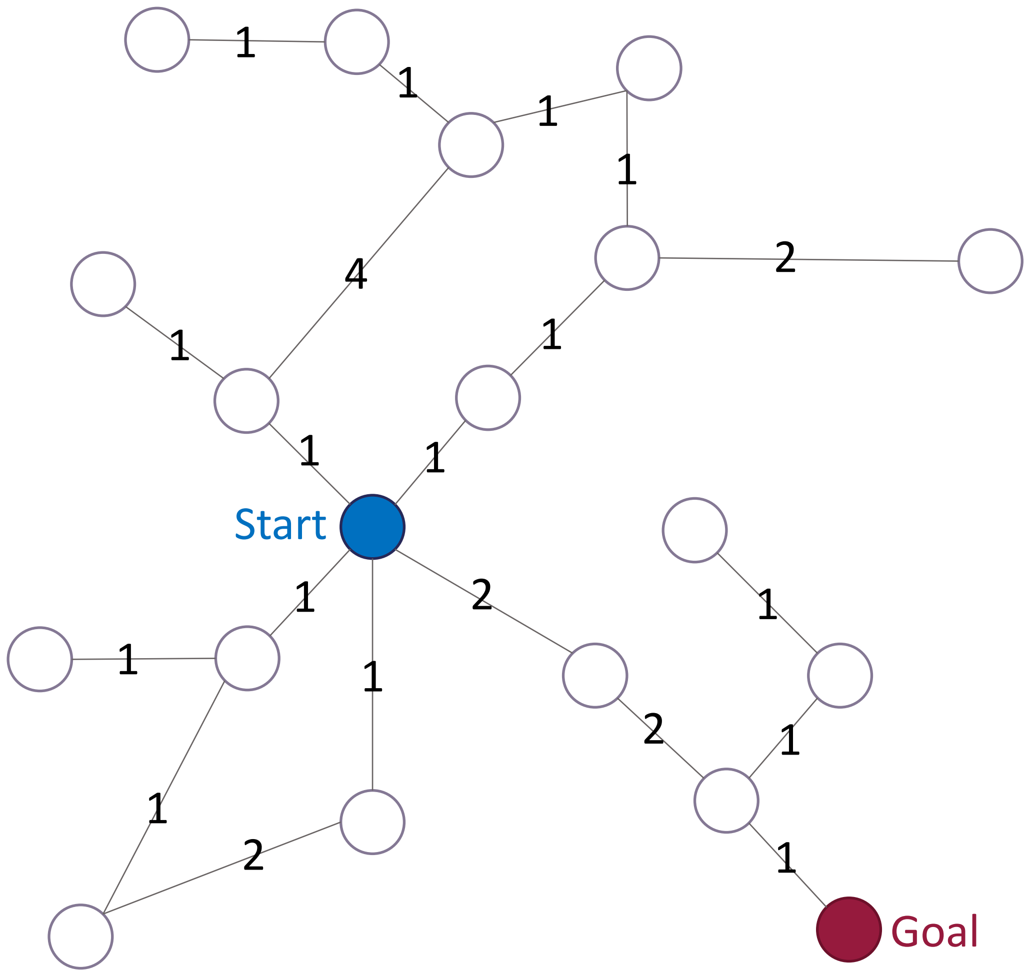
Path finding

Dijkstra path search

Node cost = sum of weights from start node

From start node ➡ Visit “closest” (cheapest) unexplored

Stop when goal is reached



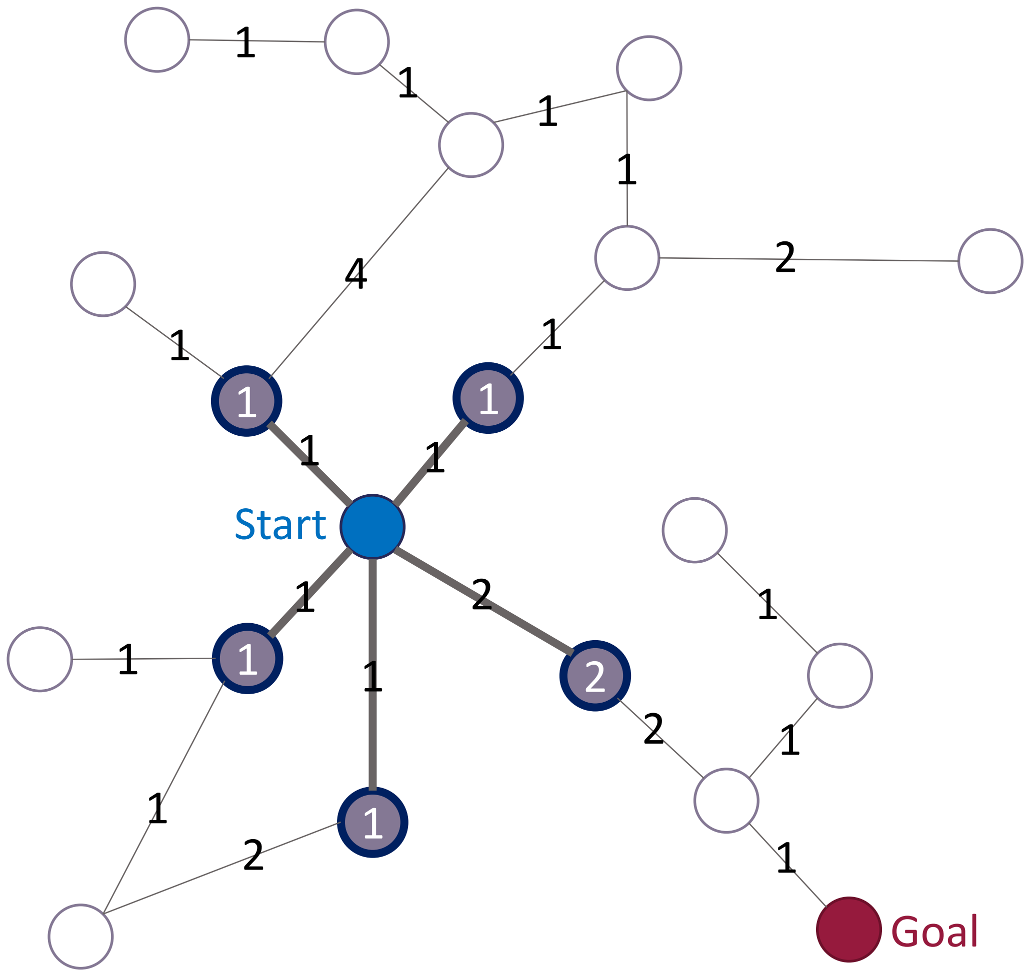
Path finding

Dijkstra path search

Node cost = sum of weights from start node

From start node ➡ Visit "closest" (cheapest) unexplored

Stop when goal is reached



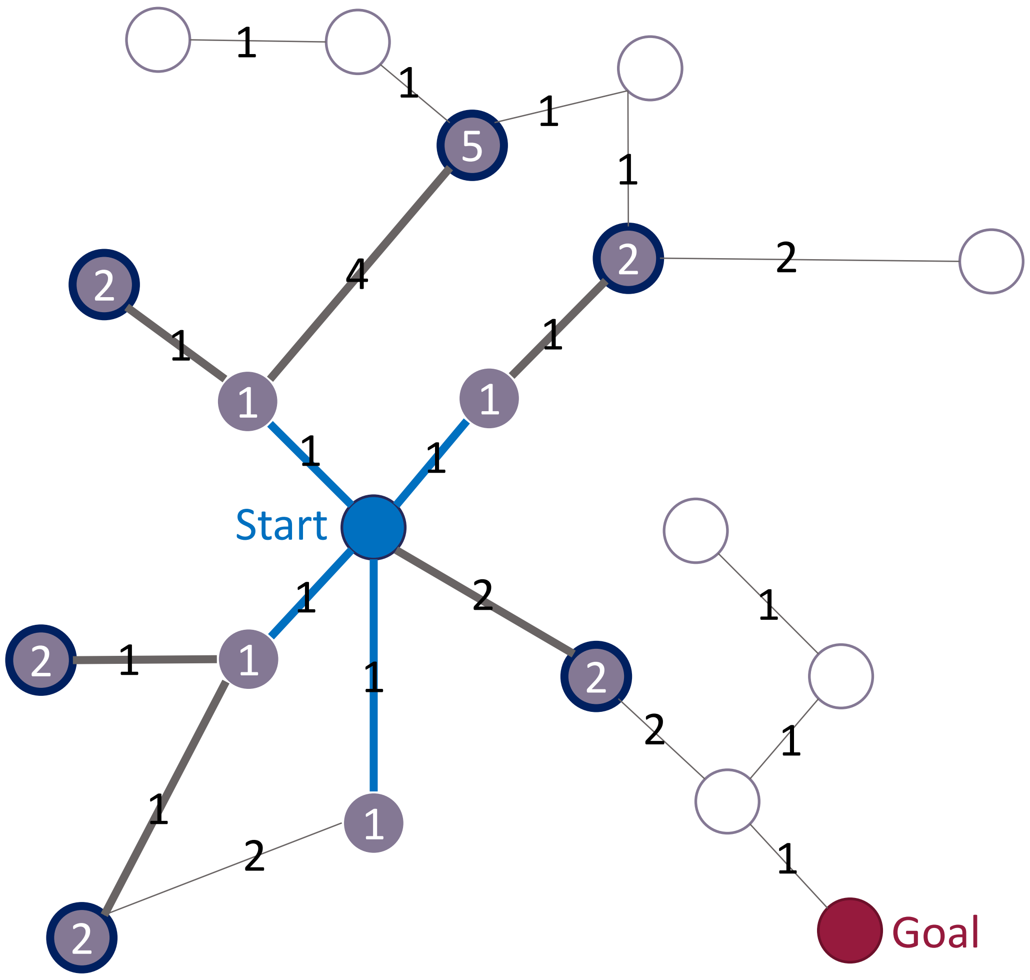
Path finding

Dijkstra path search

Node cost = sum of weights from start node

From start node ➡ Visit “closest” (cheapest) unexplored

Stop when goal is reached



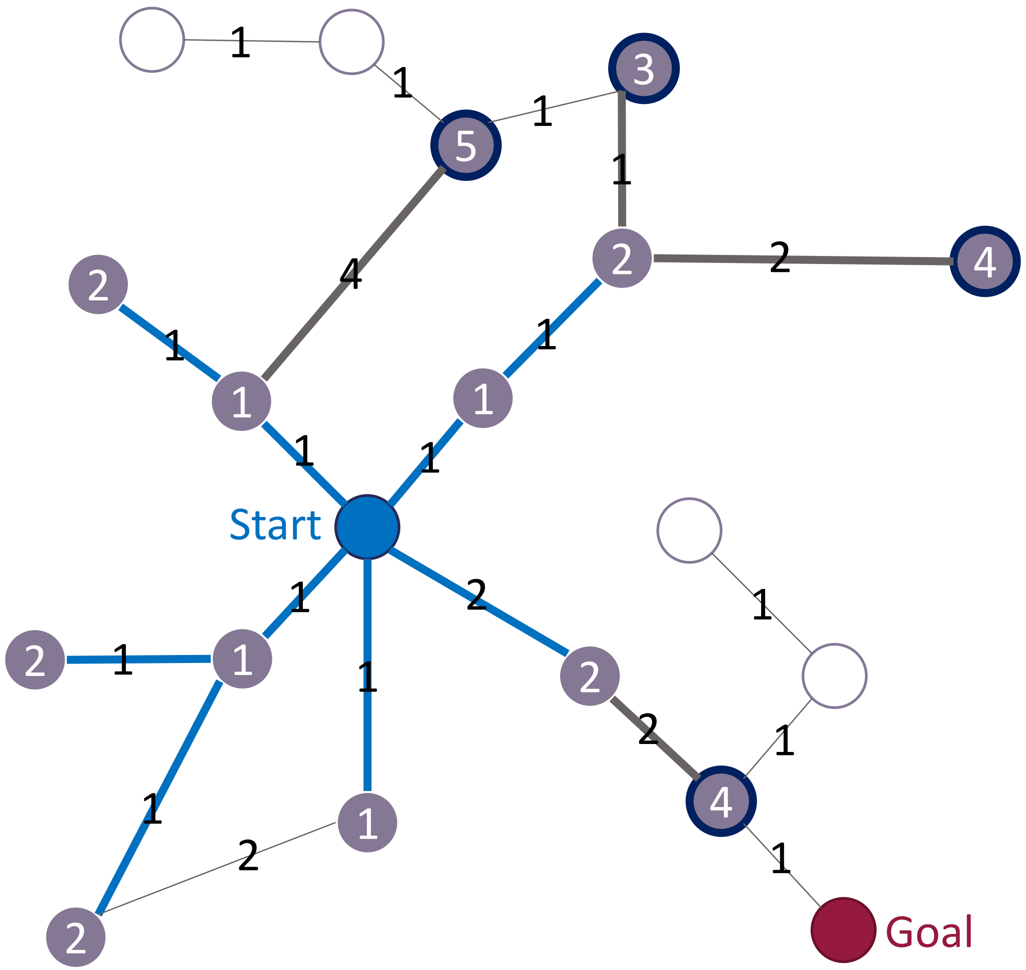
Path finding

Dijkstra path search

Node cost = sum of weights from start node

From start node ➡ Visit “closest” (cheapest) unexplored

Stop when goal is reached



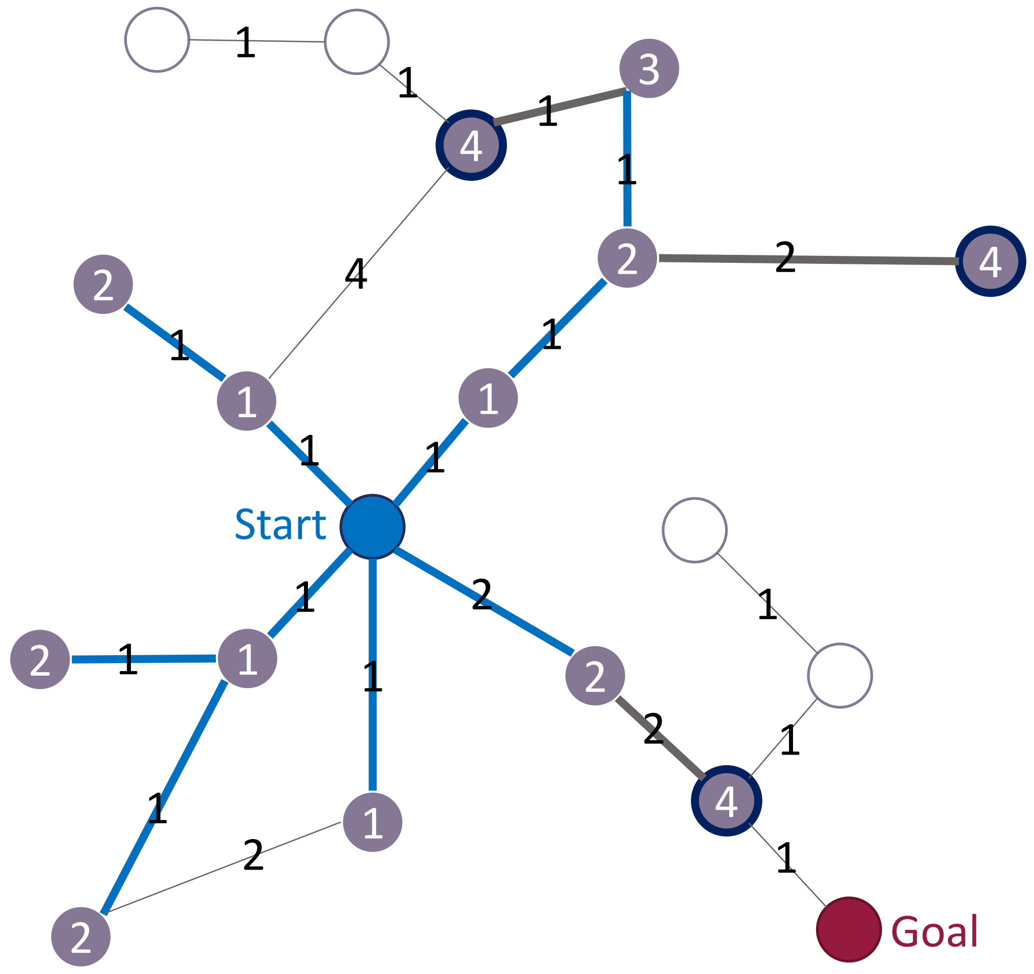
Path finding

Dijkstra path search

Node cost = sum of weights from start node

From start node → Visit “closest” (cheapest) unexplored

Stop when goal is reached



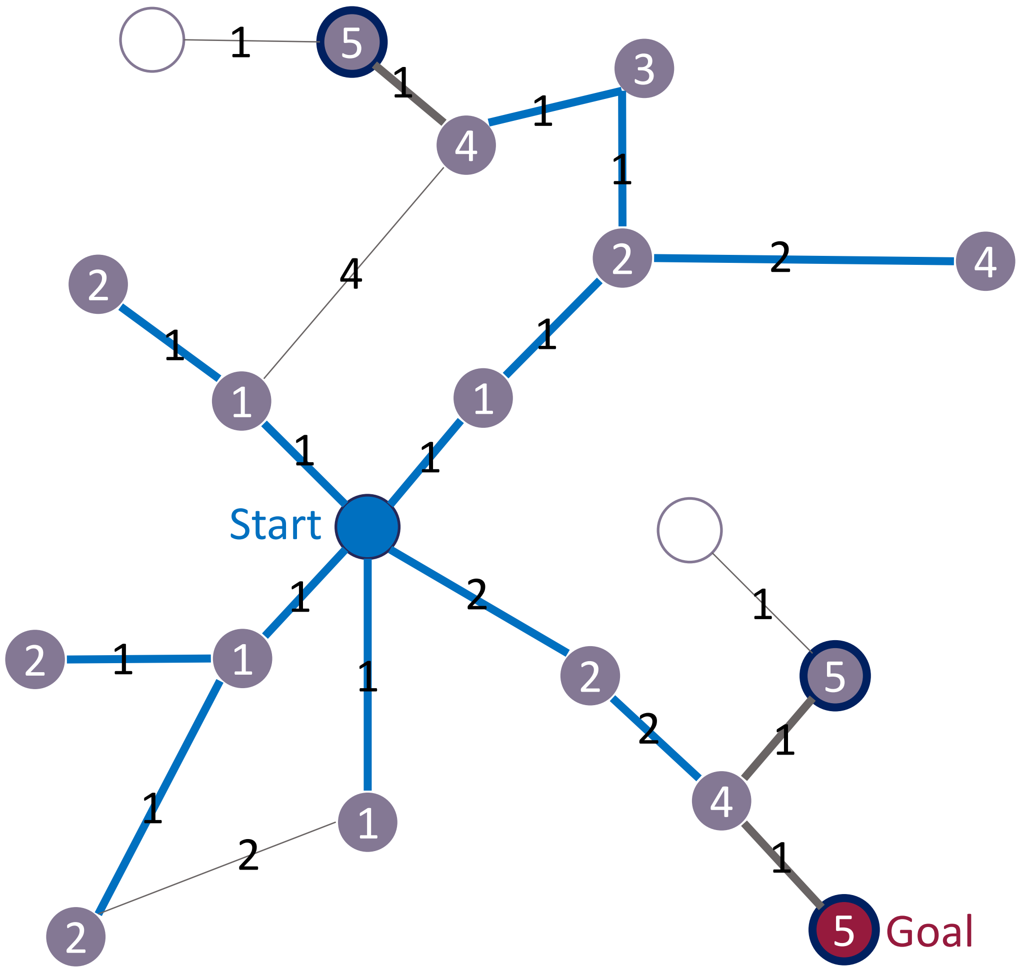
Path finding

Dijkstra path search

Node cost = sum of weights from start node

From start node ➡ Visit "closest" (cheapest) unexplored

Stop when goal is reached



Path finding

Dijkstra path search

Node cost = sum of weights from start node

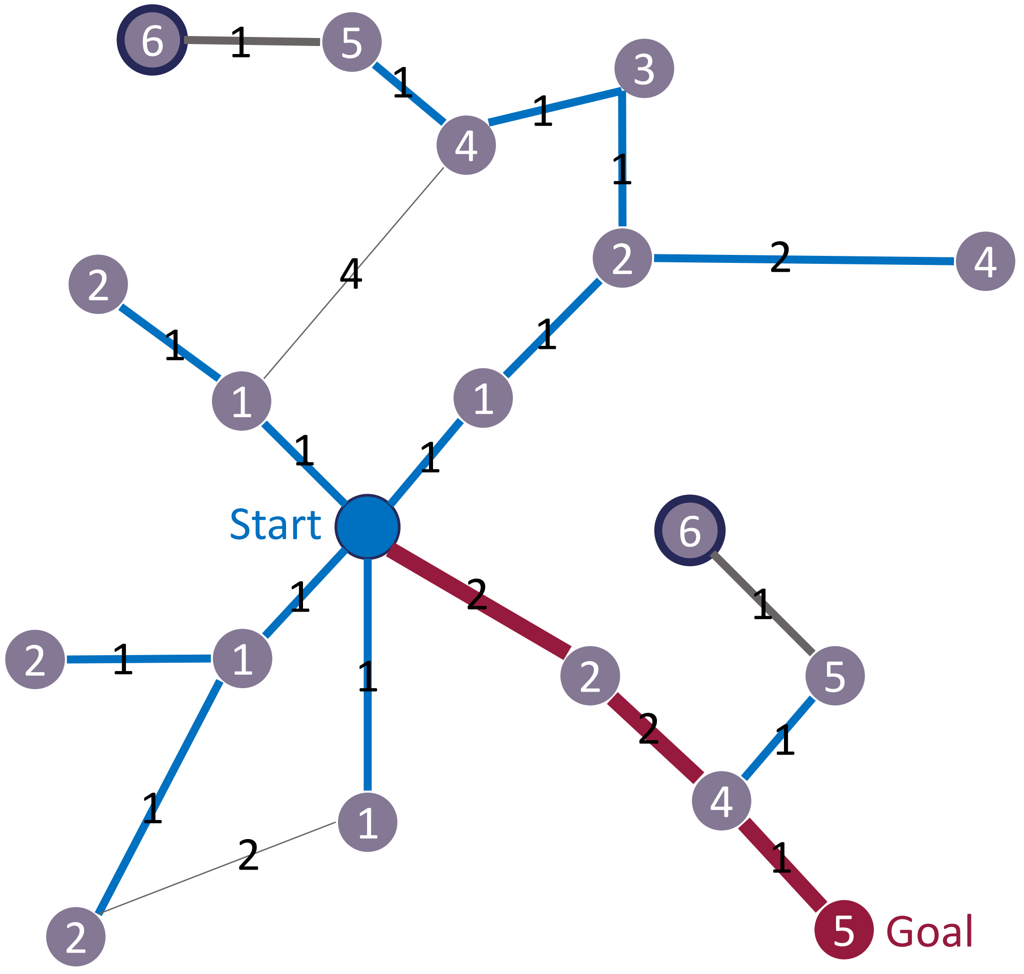
From start node ➡ Visit "closest" (cheapest) unexplored

Stop when goal is reached

Complete (find a path if there exist one)

Admissible (optimal path)

Require positive weights



Path finding

Grid distance functions

In path finding ➡ Graph = discretized space (grid)

Cost = distance

Chessboard

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

$$d_{1,2} = \max(|x_2 - x_1|, |y_2 - y_2|)$$

Manhattan

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

$$d_{1,2} = |x_2 - x_1| + |y_2 - y_2|$$

Euclidean

2.8	2.2	2	2.2	2.8
2.2	1.4	1	1.4	2.2
2	1	0	1	2
2.2	1.4	1	1.4	2.2
2.8	2.2	2	2.2	2.8

$$d_{1,2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Path finding

Grid distance functions

In path finding ➡ Graph = discretized space (grid)

Cost = distance

Same cost

Same cost

Chessboard

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

$$d_{1,2} = \max(|x_2 - x_1|, |y_2 - y_2|)$$

Manhattan

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

$$d_{1,2} = |x_2 - x_1| + |y_2 - y_2|$$

Euclidean

2.8	2.2	2	2.2	2.8
2.2	1.4	1	1.4	2.2
2	1	0	1	2
2.2	1.4	1	1.4	2.2
2.8	2.2	2	2.2	2.8

$$d_{1,2} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

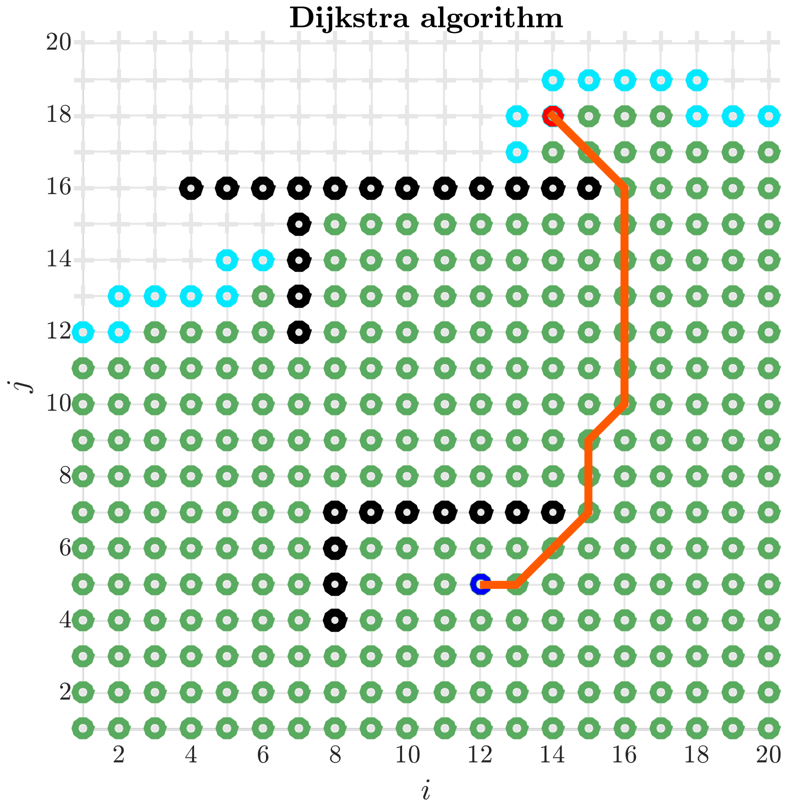
Path finding

Dijkstra path search

Optimal **graph** path cost

➡ **Map** path cost conditioned by grid resolution

Significant portion of grid is explored



Path finding

A* path search

Extension of Dijkstra ➡ "Push" exploration toward goal

Node cost = (cost from start) + (estimate coast to goal)

Manhattan

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

+ goal ↑

Heuristic

0	0	0	0	0
1	1	1	1	1
2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

= goal ↑

Cost

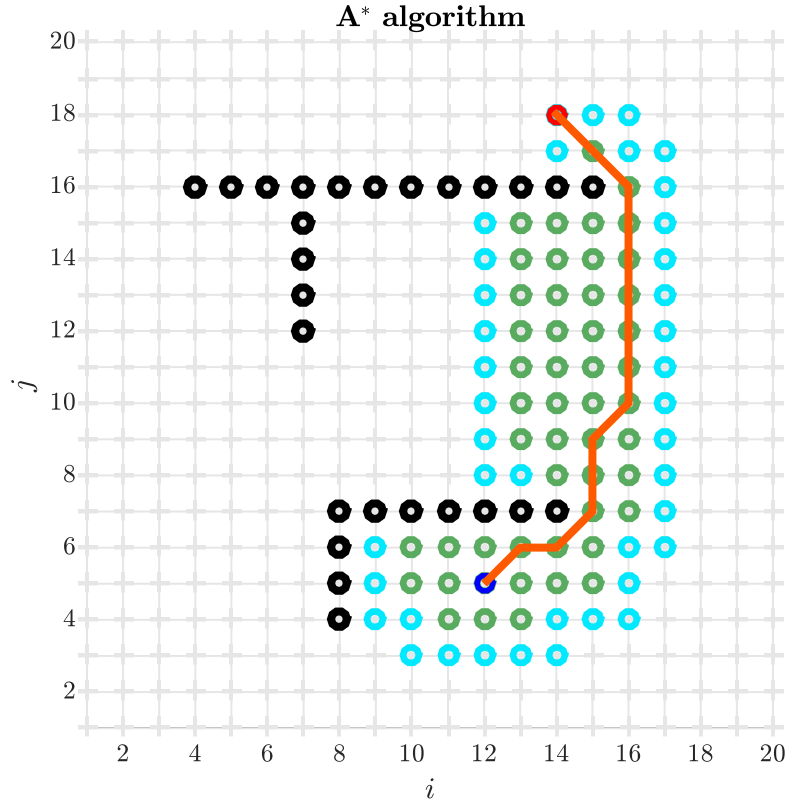
4	3	2	3	4
4	3	2	3	4
4	3	2	3	4
6	5	4	5	6
8	7	5	7	8

Path finding

A* path search

Extension of Dijkstra ➡ "Push" exploration toward goal

Node cost = (cost from start) + (estimate coast to goal)



Complete (find a path if there exist one)

Admissible (optimal path) if heuristic is admissible

Require positive weights

Dijkstra = A with null heuristic*

Heuristic underestimate cost

Path finding

A* path search

Extension of Dijkstra ➡ "Push" exploration toward goal

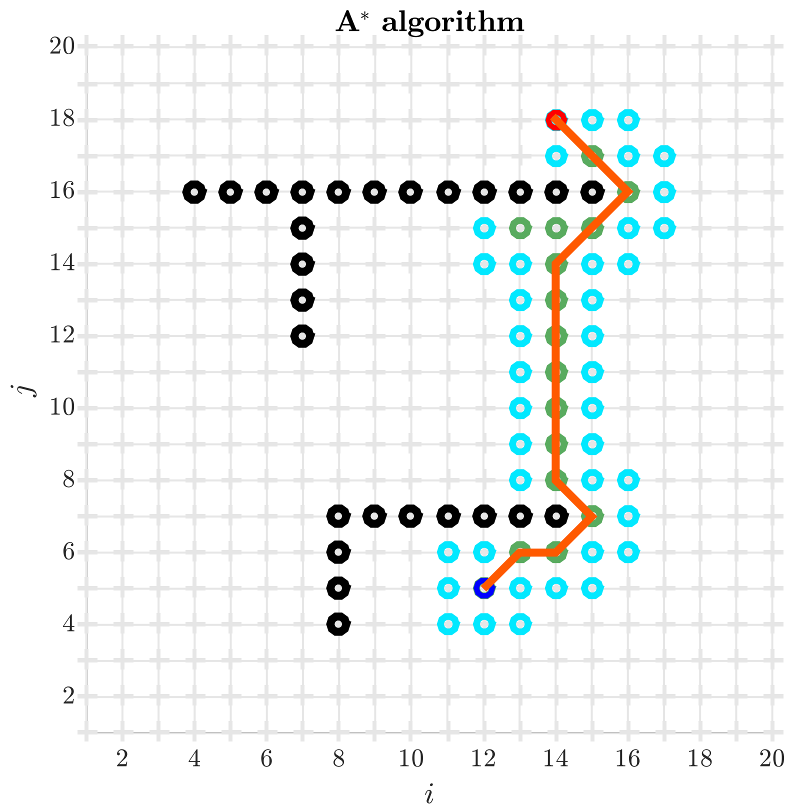
Node cost = (cost from start) + (estimate coast to goal)

Complete (find a path if there exist one)

Admissible (optimal path) if heuristic is admissible

Require positive weights

Dijkstra = A with null heuristic*



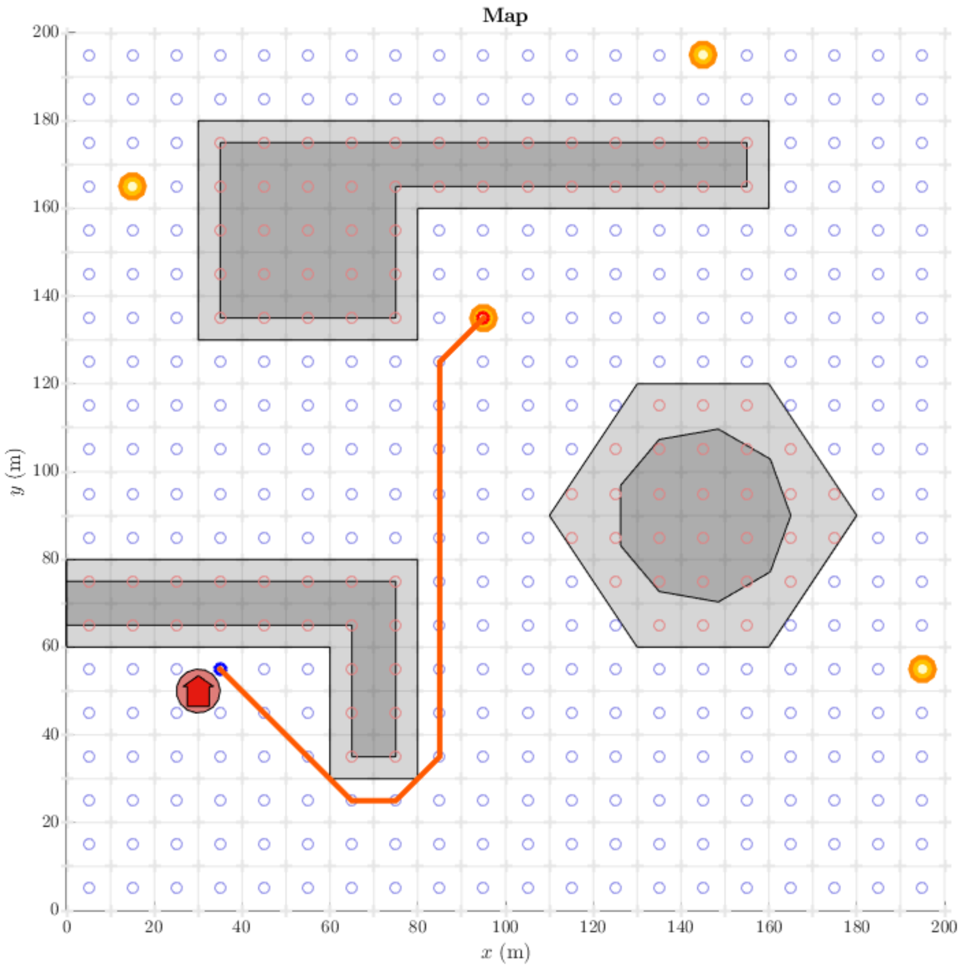
Heuristic overestimate cost

Mission

Tasks

Task 3: extinguish fire seats

➡ Send UGV to fire seats

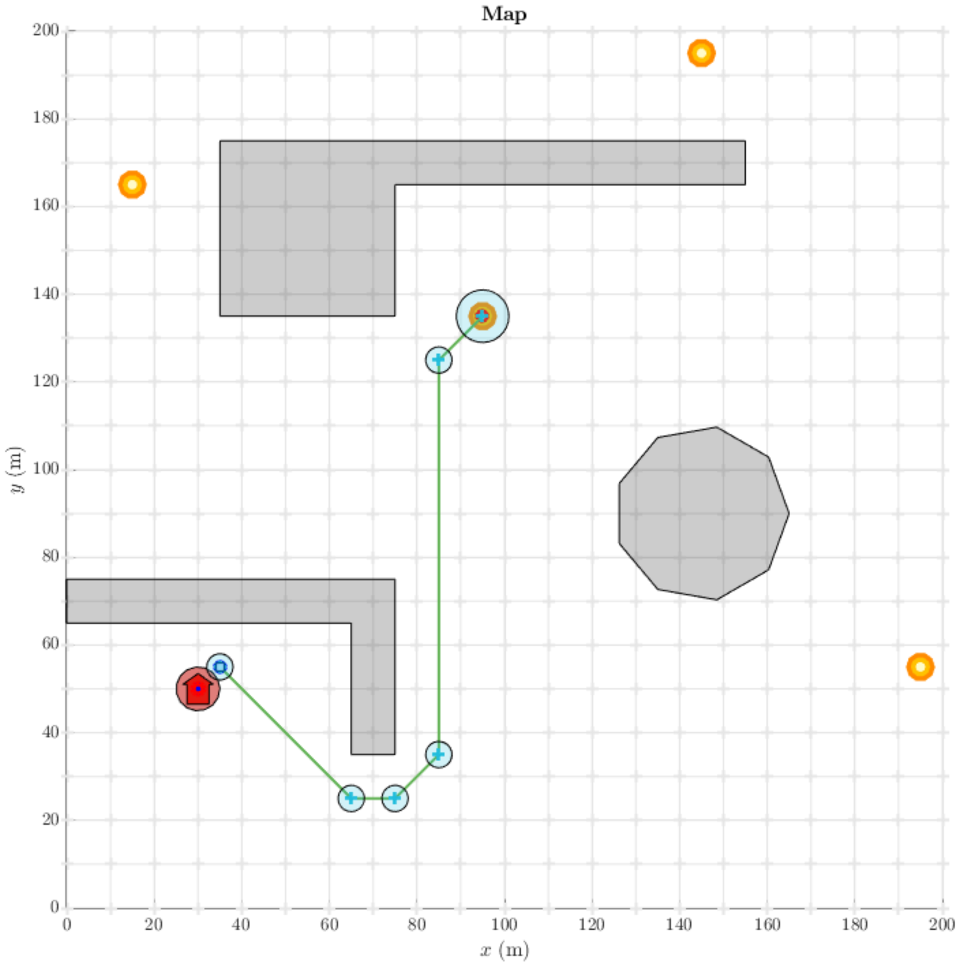


Mission

Tasks

Task 3: extinguish fire seats

➡ Send UGV to fire seats



Position control

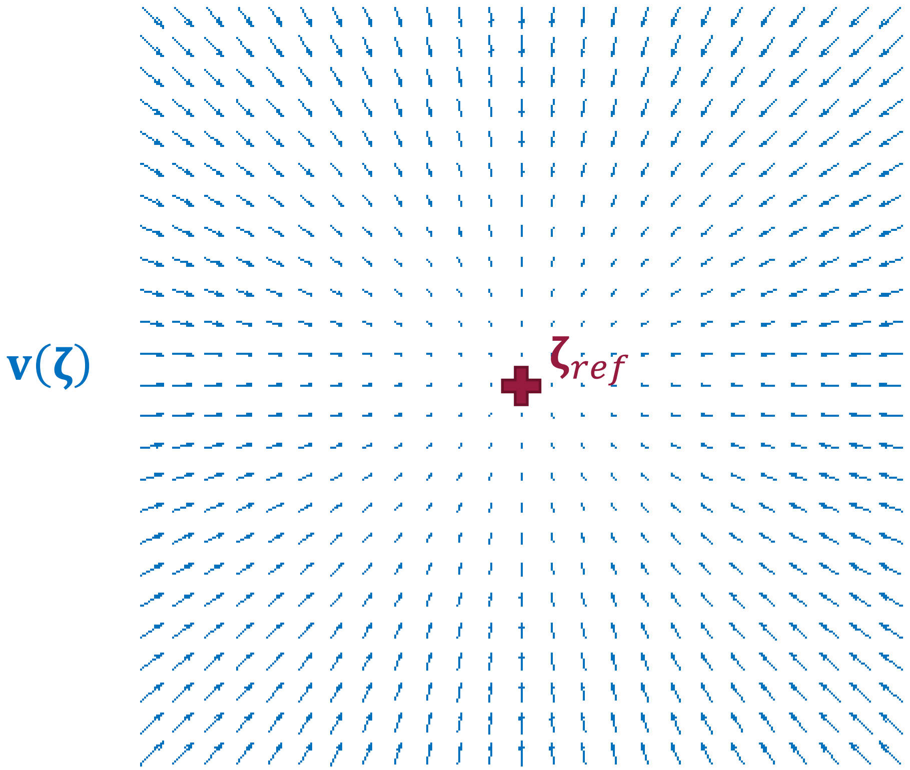
Command vector field

Join fix waypoints

➡ Proportional position control with velocity command

$$\mathbf{v}(t) = k\left(\boldsymbol{\zeta}_{ref} - \boldsymbol{\zeta}(t)\right)$$

➡ **Induced vector field**



Time dependence can be removed

Position control

Attractive point potential field

$$\mathbf{p} = (0,0)$$

Point

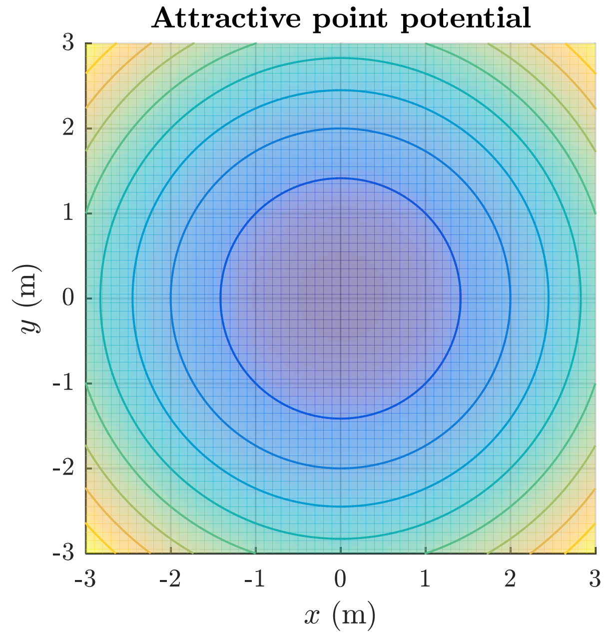
$$\mathbf{p}$$

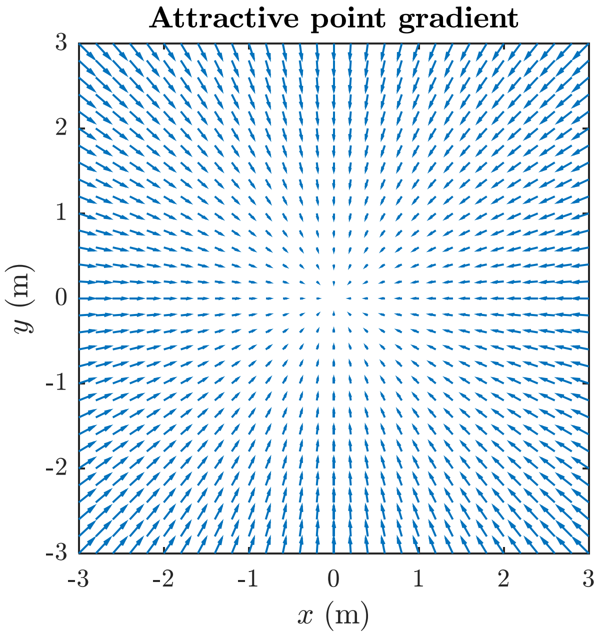
Potential

$$V(\boldsymbol{\zeta}) = \frac{1}{2}(\boldsymbol{\zeta} - \mathbf{p})^{\intercal}(\boldsymbol{\zeta} - \mathbf{p})$$

Gradient

$$-\nabla V(\boldsymbol{\zeta}) = \mathbf{p} - \boldsymbol{\zeta}$$





Potential field for path following

Attractive line potential field

Point, unit vector

$$\mathbf{p}, \mathbf{u}$$

Potential

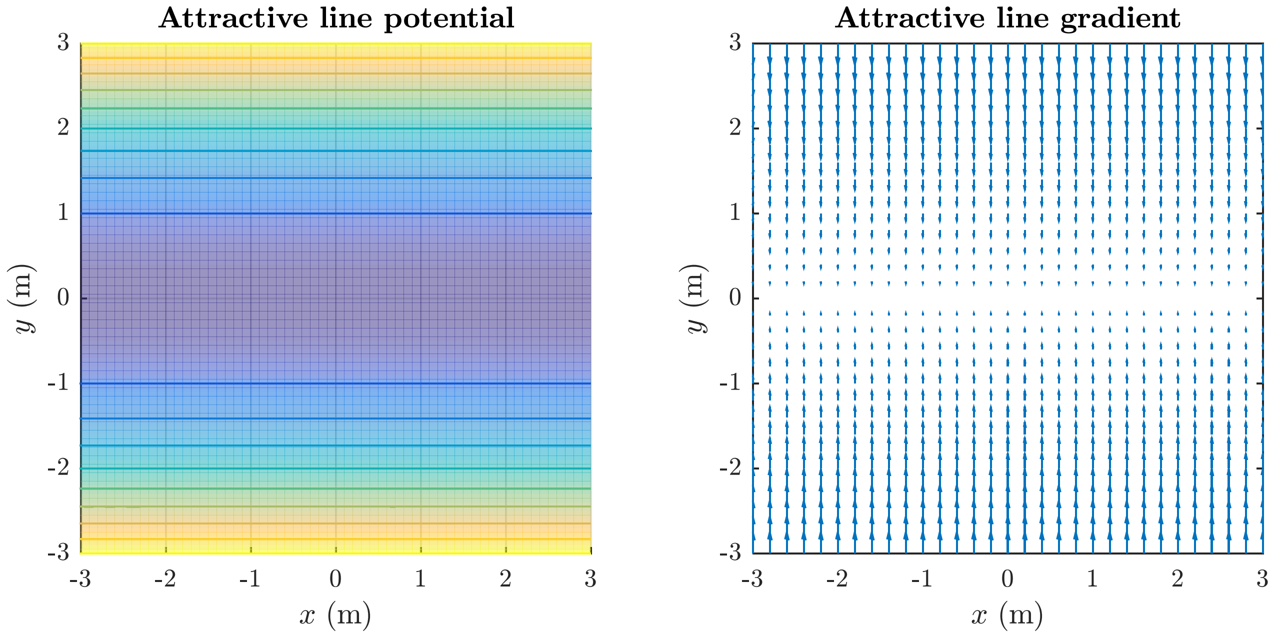
$$\mathbf{n} = (-u_y, u_x)$$

$$V(\boldsymbol{\zeta}) = \frac{1}{2}(\boldsymbol{\zeta} - \mathbf{p})^\top \mathbf{n}^\top \mathbf{n}(\boldsymbol{\zeta} - \mathbf{p})$$

Gradient

$$-\nabla V(\boldsymbol{\zeta}) = \mathbf{n}^\top(\mathbf{p} - \boldsymbol{\zeta})\mathbf{n}$$

$$\mathbf{p} = (0,0), \mathbf{u} = (1,0)$$



Potential field for path following

Repulsive point potential field

$\mathbf{p} = (0,0)$

Point

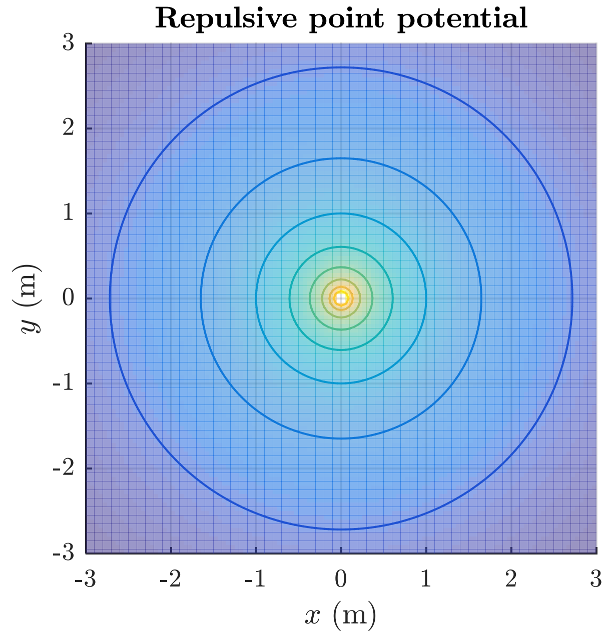
$$\mathbf{p}$$

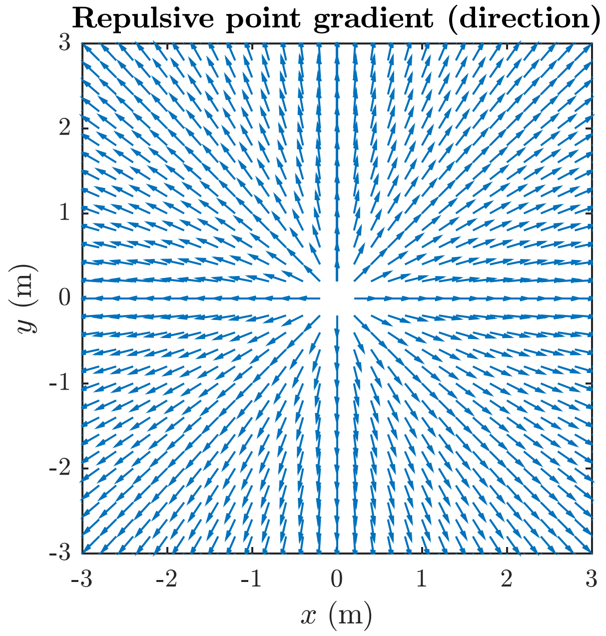
Potential

$$V(\boldsymbol{\zeta}) = \frac{1}{\|\boldsymbol{\zeta} - \mathbf{p}\|}$$

Gradient

$$-\boldsymbol{\nabla} V(\boldsymbol{\zeta}) = \frac{\boldsymbol{\zeta} - \mathbf{p}}{\|\boldsymbol{\zeta} - \mathbf{p}\|^3}$$





Potential field for path following

Repulsive disc potential field

$$\mathbf{p} = (0,0), r = 1$$

Center, radius

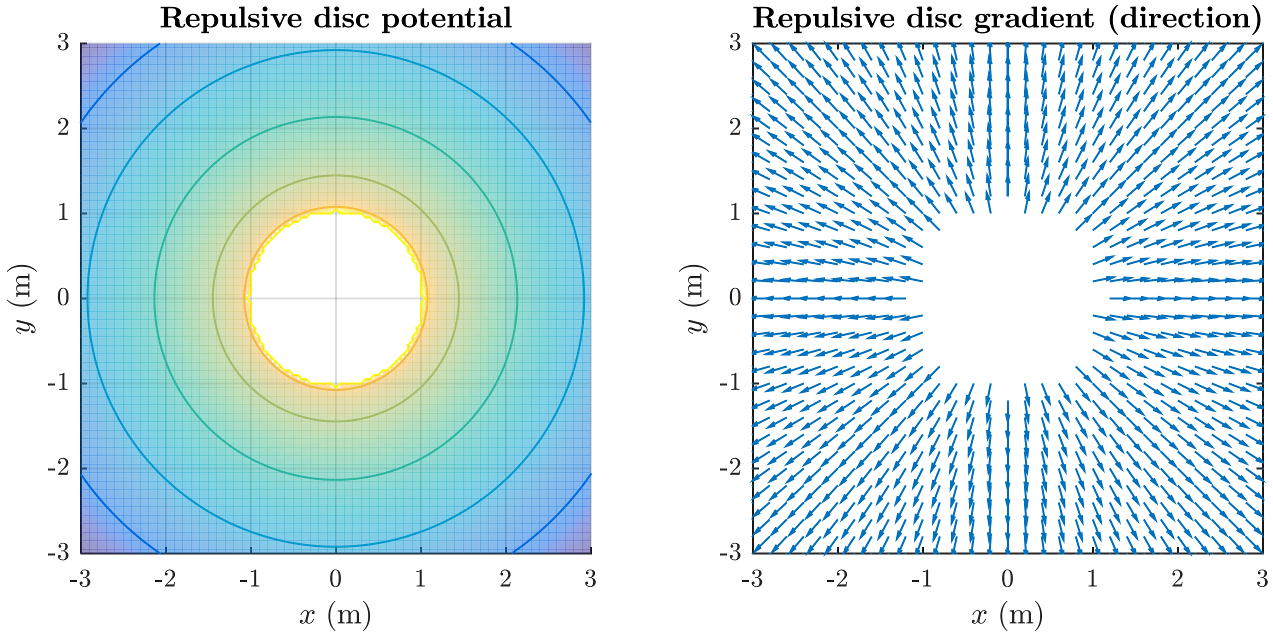
$$\mathbf{p}, r$$

Potential

$$V(\boldsymbol{\zeta}) = \begin{cases} \infty & \text{if } \|\boldsymbol{\zeta} - \mathbf{p}\| \leq r \\ \dfrac{1}{\|\boldsymbol{\zeta} - \mathbf{p}\| - r} & \text{else} \end{cases}$$

Gradient

$$-\boldsymbol{\nabla} V(\boldsymbol{\zeta}) = \begin{cases} \mathbf{0} & \text{if } \|\boldsymbol{\zeta} - \mathbf{p}\| \leq r \\ \dfrac{\boldsymbol{\zeta} - \mathbf{p}}{\|\boldsymbol{\zeta} - \mathbf{p}\| \left(\|\boldsymbol{\zeta} - \mathbf{p}\| - r\right)^2} & \text{else} \end{cases}$$



Potential field for path following

Uniform potential field

$$\mathbf{v} = (1,0)$$

Gradient

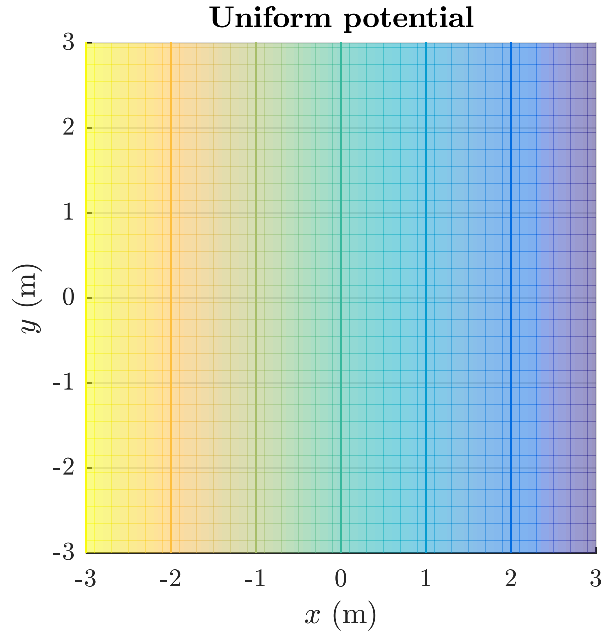
$$\mathbf{v}$$

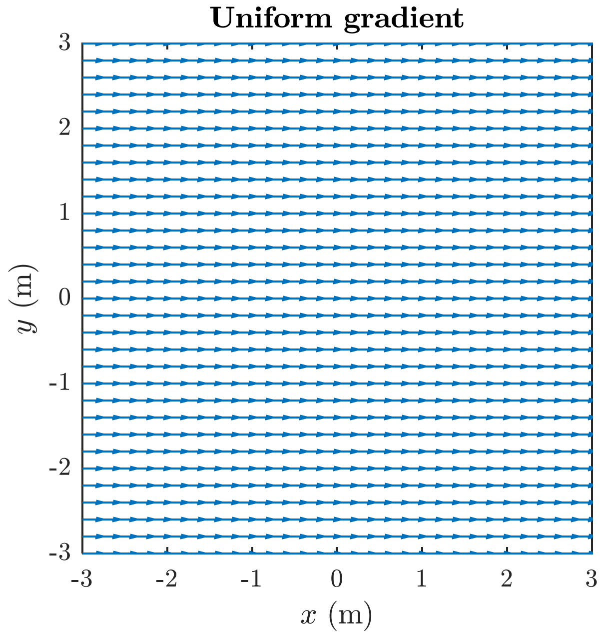
Potential

$$V(\boldsymbol{\zeta}) = -\mathbf{v}^\top \boldsymbol{\zeta}$$

Gradient

$$-\boldsymbol{\nabla} V(\boldsymbol{\zeta}) = \mathbf{v}$$





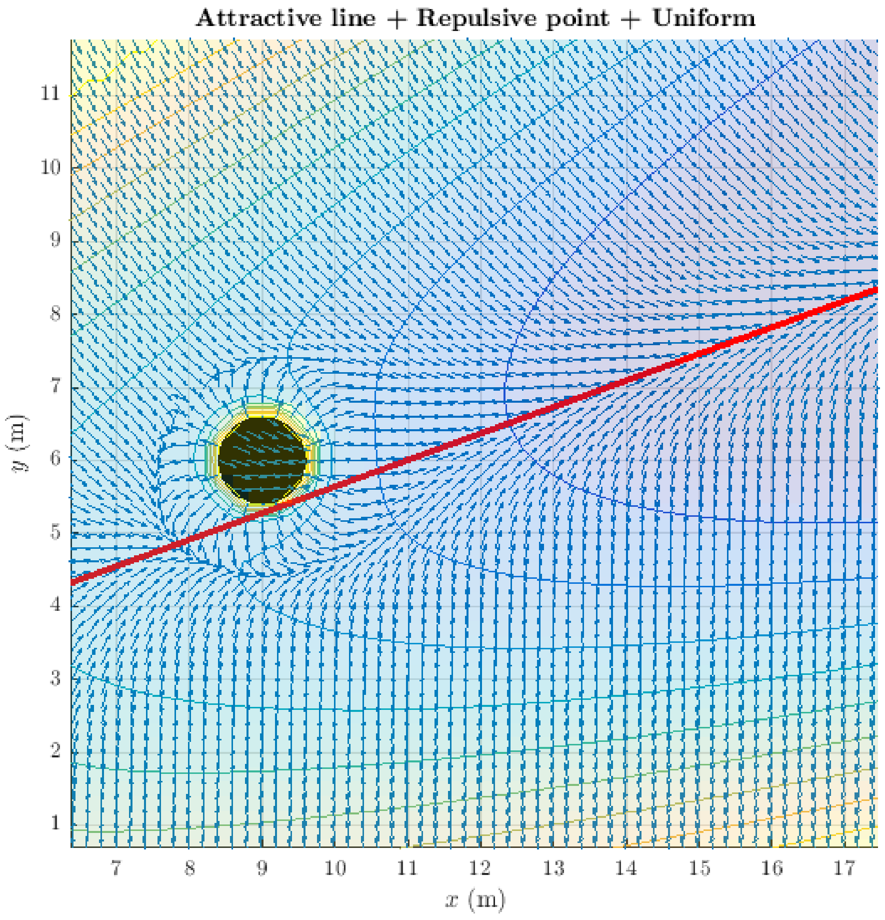
Potential field for path following

Potential fields combination

Command = gradient descent

Potential can be combined

➡ Path following **emerges** from the obtained control law

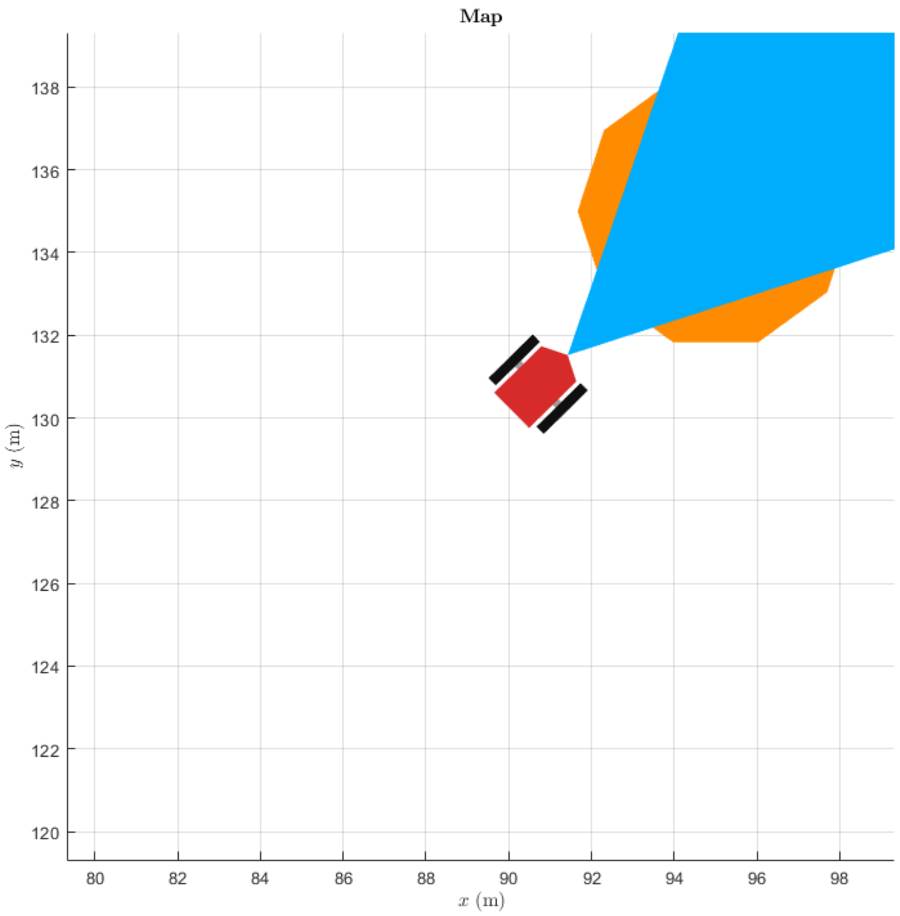


Mission

Tasks

Task 3: extinguish fire seats

➡ Proceed to the next fire seat

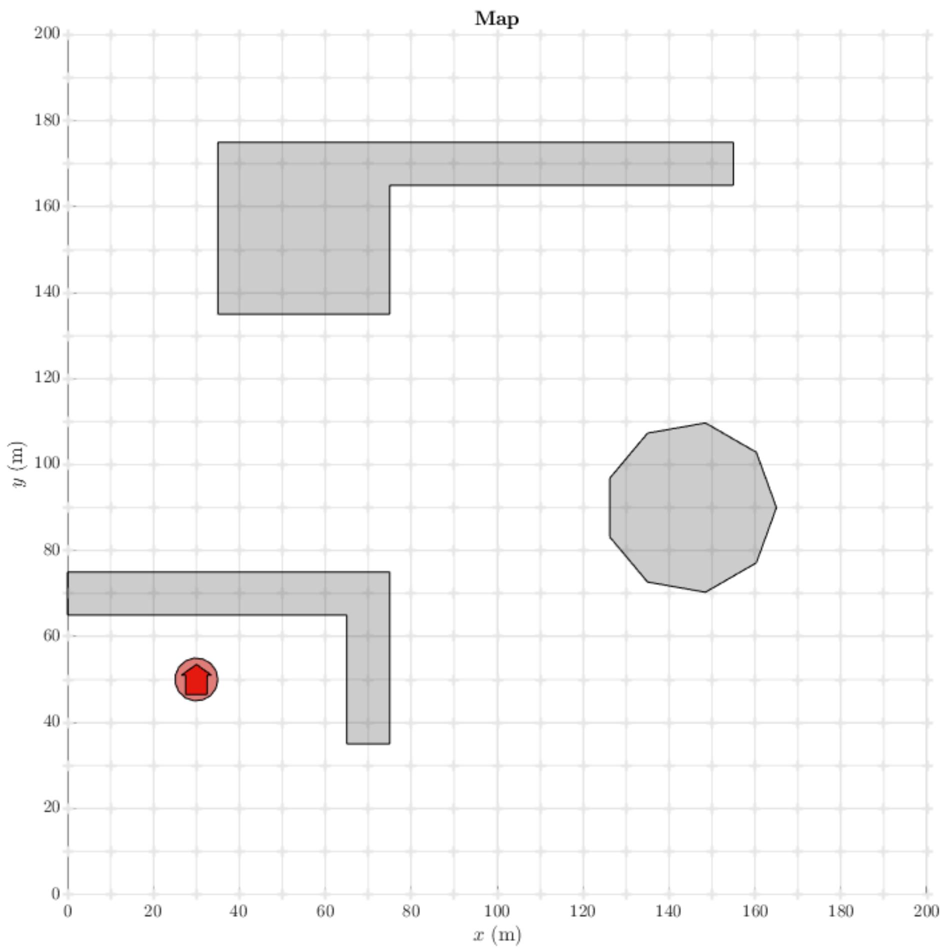


Mission

Tasks

Task 3: extinguish fire seats

➡ Proceed to the next fire seat



Mission

Tasks

Mission complete!

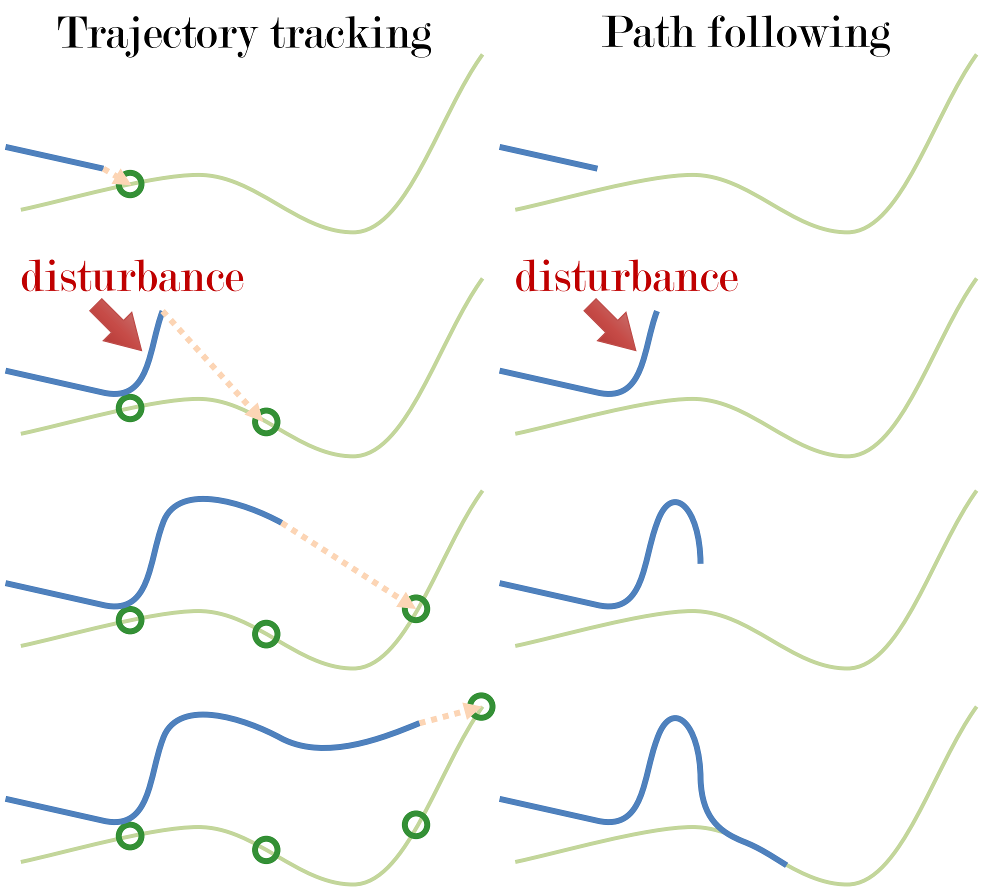
➡ You deserve a cookie

Trajectory tracking vs Path following

Robustness

Path following: no time dependence

➡ Reference trajectory cannot be "lost"

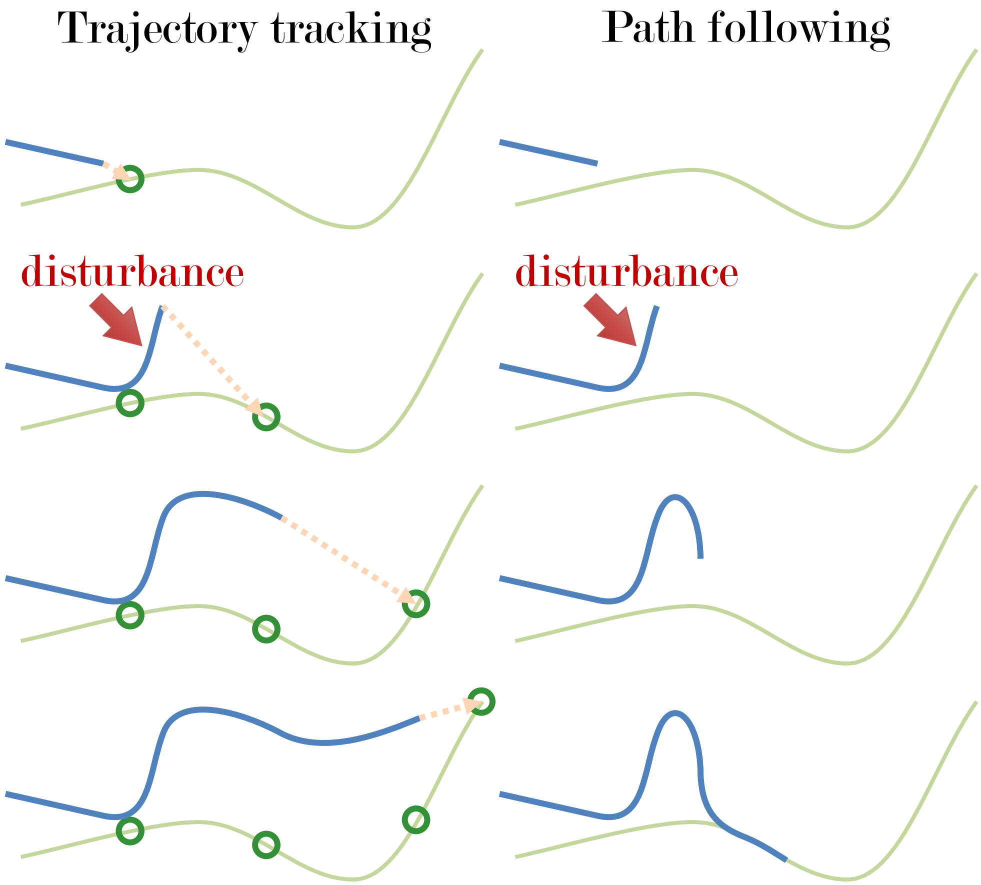


Trajectory tracking vs Path following

"Fine" motion control

Path following: no time dependence

➡ Less control on trajectory and duration



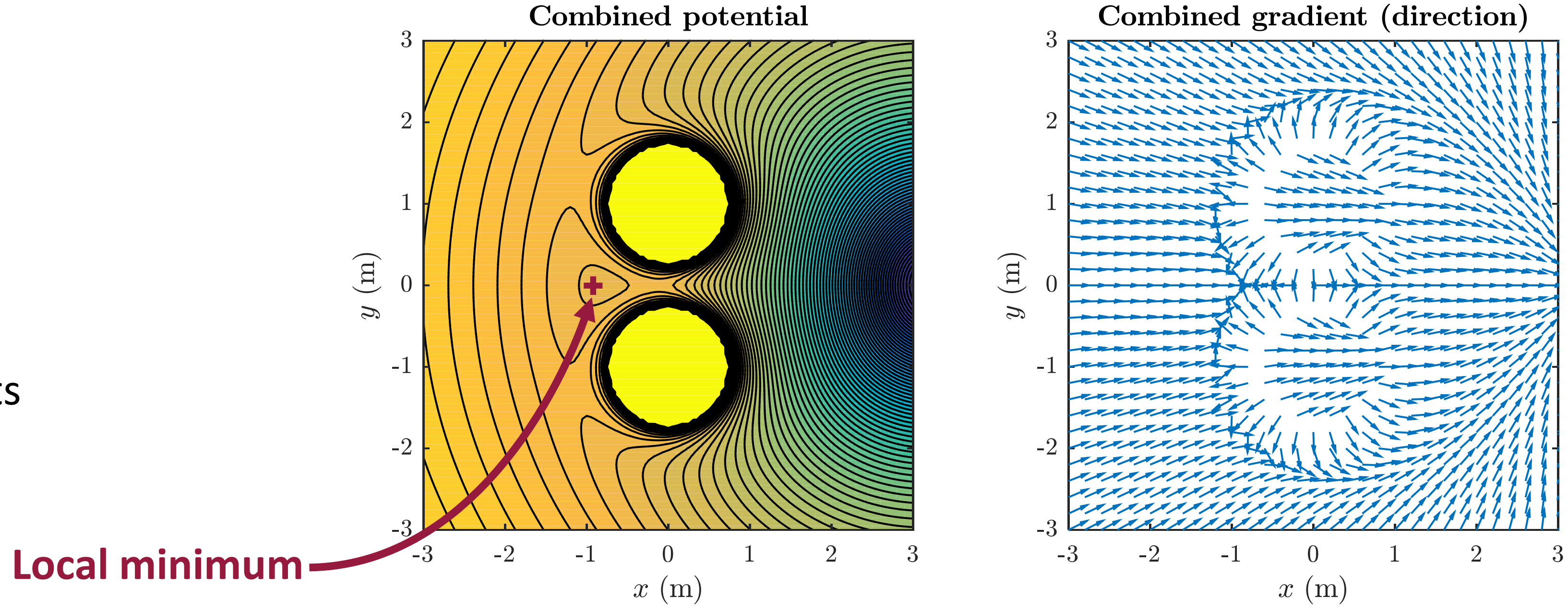
Trajectory tracking vs Path following

Complexity

Path following law can be hard to synthesize for complex systems

Require care with boundary effects

Attractive point + repulsive polygon



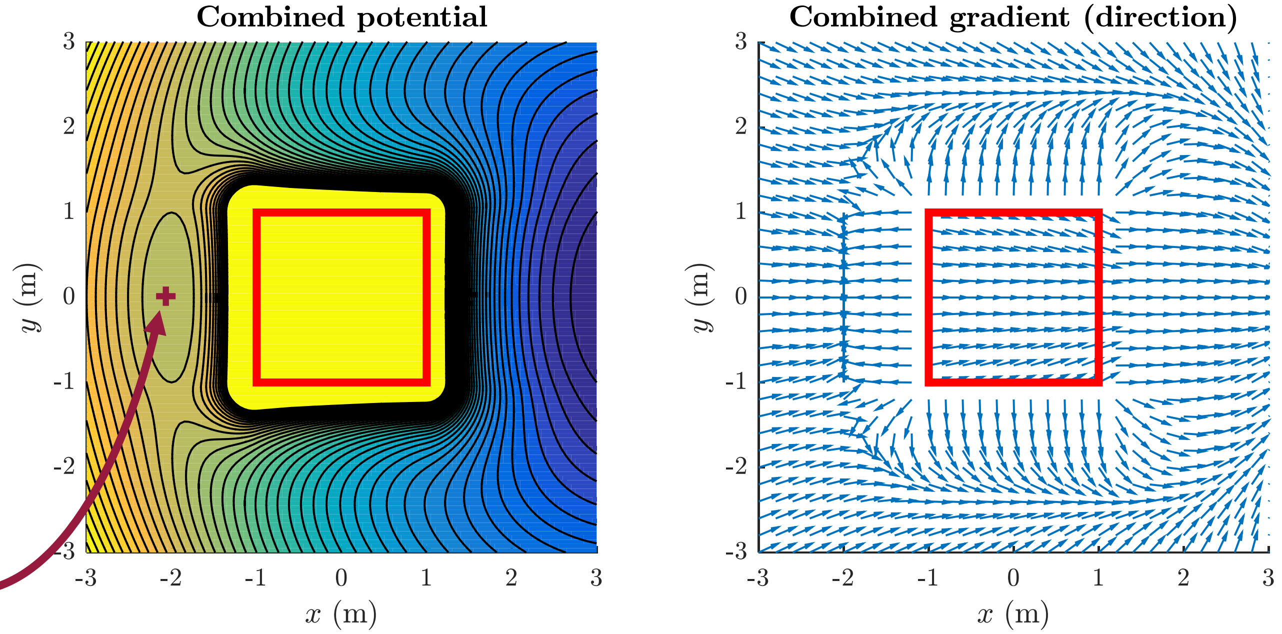
Trajectory tracking vs Path following

Complexity

Path following law can be hard to synthesize for complex systems

Require care with boundary effects

Attractive point + repulsive polygon



Local minimum

Fails to complete case study 1