
THE UNIVERSITY *of*
NEW ORLEANS

# CSCI6650: Planning Algorithm

Instructor: Abdullah Al Redwan Newaz, PhD

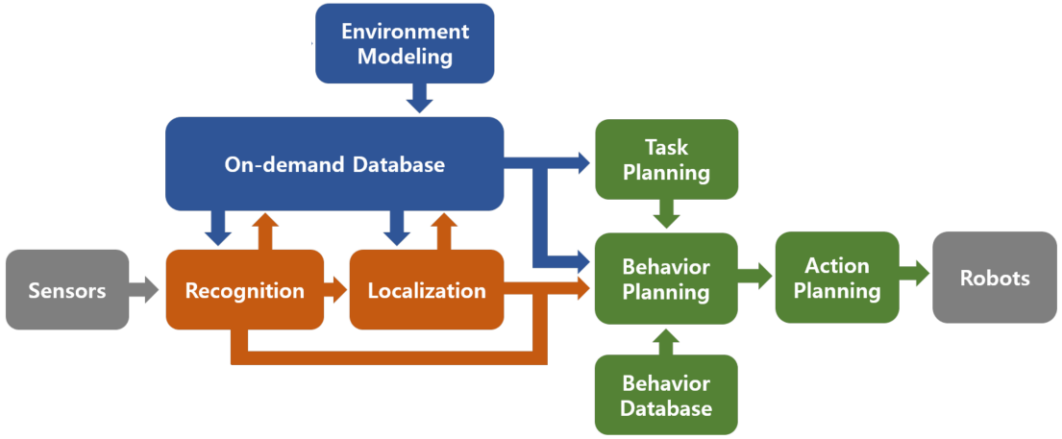
*Assistant Professor*

Department of Computer Science
Mathematics Building #345

# Intelligent Behaviors

- A **behavior** is a collection of actions or maneuvers that the agent (a.k.a the robot) performs to achieve a goal
- **Autonomous** means capable of operating without operator input for extended periods of time.
- **Autonomous behaviors** (e.g., track a thermocline, follow waypoints) are on-board capabilities that provide actions for known situations and events.
- **Intelligent behaviors** are capabilities that provide a structure to reason about unexpected or unknown situations or events and allow the agent to adapt and complete a mission where possible

# Intelligent Autonomy



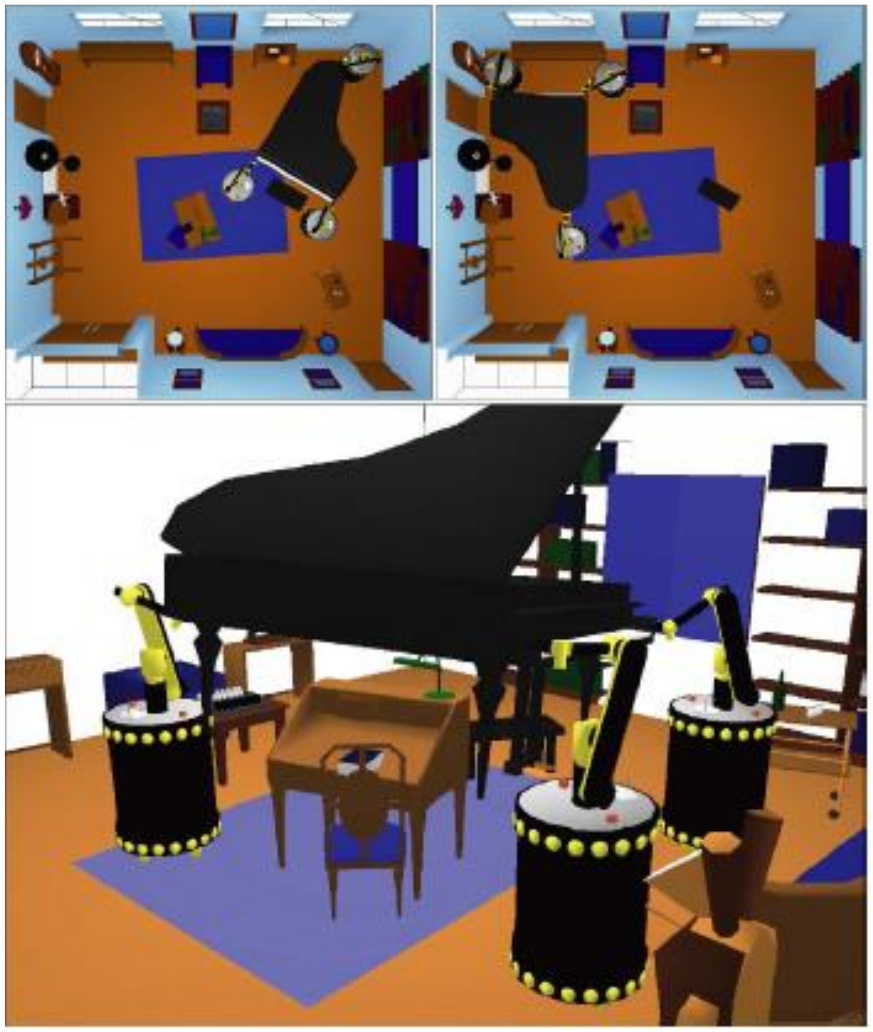

- **Intelligent autonomy** provides a framework for deliberation and reasoning with intelligent behaviors
- Intelligent autonomy can specifically include the ability to fulfill goals by:
  - making decisions given a set of automated planned options and, if these plans are inadequate, to make new plans;
  - analyzing and interpreting in situ sensor data to create information;
  - diagnosing and recommending solutions for robot or mission problems detected through sensing and monitoring; and
  - learning and adaptation; and
  - collaborating with other agents, systems, or humans through point-to-point or network communications.

# Type of Autonomy Framework

- **Reactive architectures:** use the sense-plan-act schema and are suited to structured and predictable environments and missions.
- **Deliberative architectures:** base their planning on a world model. They take on-board sensor data and process it into information to build and update that world model.
- **Hybrid architectures:** combine the advantage of fast reactive responses based on a behavior-based layer coupled with a deliberative layer capable of decisions and planning.

# Planning Problem

- Planning is a fundamental problem in artificial intelligence and robotics that involves determining a sequence of actions to achieve a desired goal state.
- How to make good decisions and act effectively in complex environments.
- Common types of planning problem in Robotics:
  - **Navigation planning**
  - **Manipulation planning**
  - **Task planning**
  - **Coordination planning**
  - **Information gathering planning**
  - **Contingency planning**
  - **Planning under uncertainty**
  - **Planning with constraints**
  - **Continuous motion planning**



*How to move a piano from one room to another in a house without hitting anything?*

# AI Agents Vs Robots

Robotics planning algorithms need to reason about at least four types of constraints:

- physical constraints (e.g., robot dynamics),
- sensing constraints (e.g., limited visibility),
- resource constraints (e.g., battery life), and
- computational constraints (e.g., decision-making in real-time speed).

*All Robots are AI agents, but all AI agents are not Robots!*

# Basic Ingredients of Planning

- **State:** Planning problems involve a state space that captures all possible situations that could arise. The state could, for example, represent the position and orientation of a robot, the locations of tiles in a puzzle, or the position and velocity of a helicopter. Both discrete (finite, or countably infinite) and continuous(uncountably infinite) state spaces will be allowed.

- **Time:** All planning problems involve a sequence of decisions that must be applied over time. Time might be explicitly modeled, as in a problem such as driving a car as quickly as possible through an obstacle course. Alternatively, time may be implicit, by simply reflecting the fact that actions must follow in succession, as in the case of solving the Rubik's cube

# Basic Ingredients of Planning

- **Actions:** A plan generates actions that manipulate the state. The terms actions and operators are common in artificial intelligence; in control theory and robotics, the related terms are inputs and controls

- **Initial and goal states:** A planning problem usually involves starting in some initial state and trying to arrive at a specified goal state or any state in a set of goal states. The actions are selected in a way that tries to make this happen.

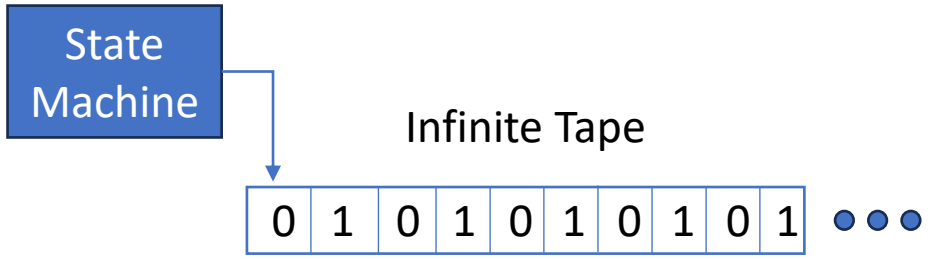
# Basic Ingredients of Planning

- **A criterion:** This encodes the desired outcome of a plan in terms of the state and actions that are executed. There are generally two different kinds of planning concerns based on the type of criterion:
  - **Feasibility:** Find a plan that causes arrival at a goal state, regardless of its efficiency.
  - **Optimality:** Find a feasible plan that optimizes performance in some care-fully specified manner, in addition to arriving in a goal state.

- **A plan:** In general, a plan imposes a specific strategy or behavior on a decision maker. A plan may simply specify a sequence of actions to be taken; however, it could be more complicated. If it is impossible to predict future states, then the plan can specify actions as a function of state. In this case, regardless of the future states, the appropriate action is determined.

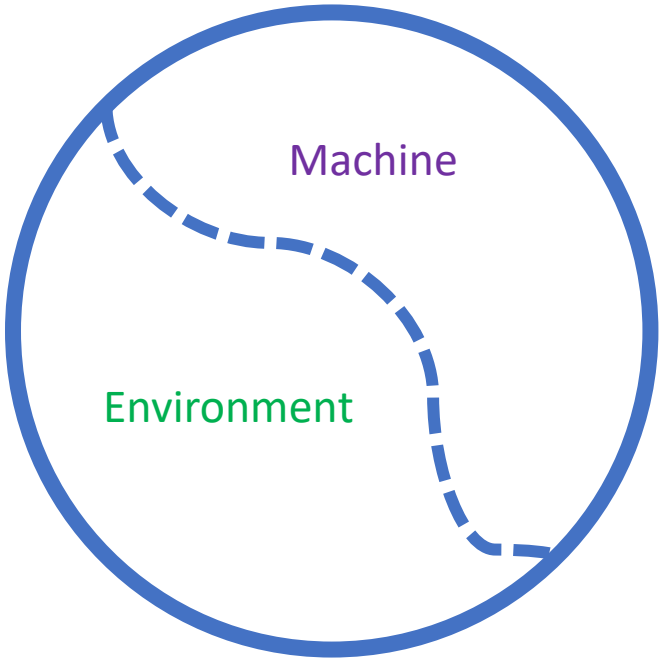
# Algorithms, Planners, and Plans

- **Algorithm** - A step-by-step procedure for solving a problem or accomplishing a task. In planning, algorithms refer to the specific search strategies used to explore the state space, such as depth-first search, A*, etc.
- **Planner** - A software program that implements a planning algorithm. The planner takes as input a domain model that defines states, actions, and goals. It uses the algorithm to search for a solution plan. Popular planners include GraphPlan, LPSAT, etc.
- **Plan** - The output of the planning process, consisting of a sequence of actions that will achieve the desired goal when executed from the initial state. A plan provides a policy for the agent to reach the goal.



# Relationships among Algorithms, Planners, and Plans

- Algorithms provide the strategy for exploring the state space to find plans. Different algorithms have tradeoffs in completeness, optimality, speed, memory use, etc.
- Planners are programs that combine algorithms with domain-specific representations and heuristics to produce plans as outputs.
- Plans are the end product that planners generate after using algorithms to search the state space defined by the domain model.
- The same planning domain can be solved with different planners using different algorithms, potentially producing different plans.
- Planners rely on algorithms, but also implement techniques like heuristic guidance to improve plan quality and search efficiency.

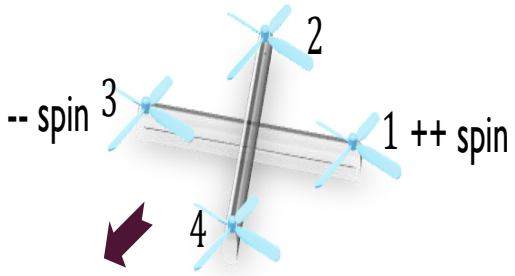



*The distinction between the machine and its environment is not absolute, but rather depends on perspective and goals*

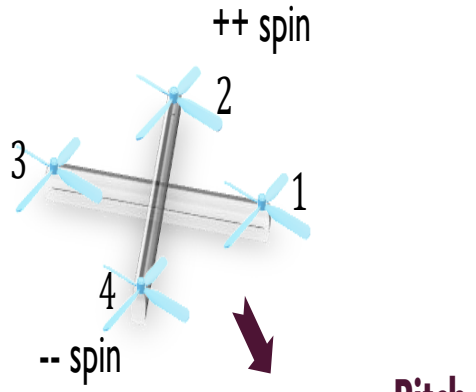
# Case Study: Unmanned Aerial Vehicle

## Quadrotors

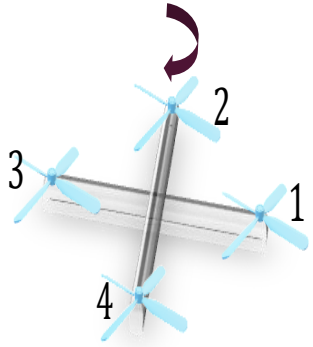
- Inherently unstable
- Motion is generated by balancing forces
- C-Space: $\{x_t, y_t, z_t, \theta_t, \phi_t, \psi_t\}$
- Controllable parameter: $\{x_t, y_t, z_t, \psi_t\}$
- Underdamped system



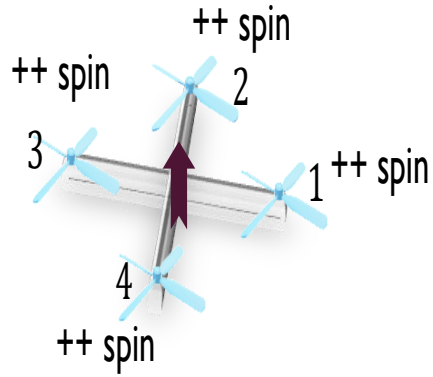

Roll




Pitch




Yaw




Throttle

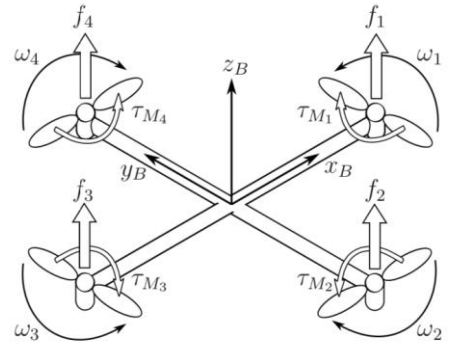
# Dynamic Model

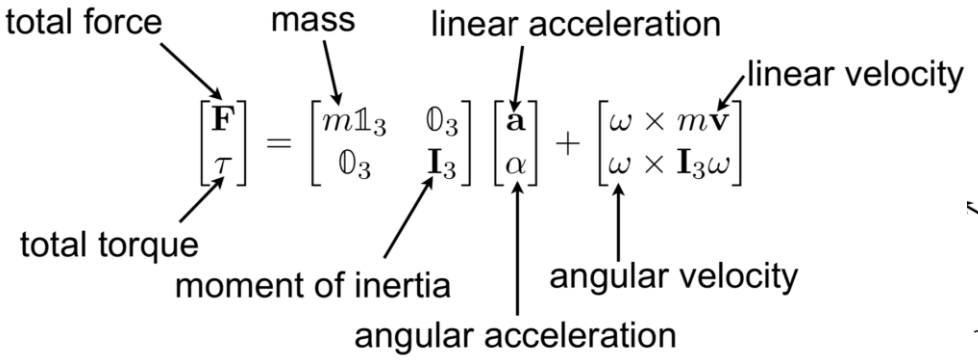
## Empirical Model

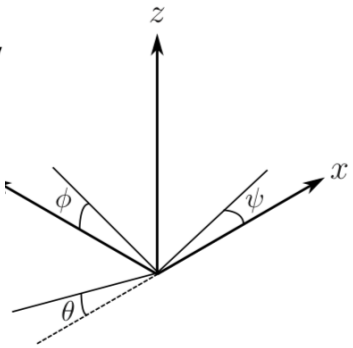
- Based on Input-Output relationship
- Sensitive to
  - Sensor noise
  - Experimental condition
- Error prone

## Mathematical Model

- Rigid-body dynamics
- Newton-Euler equation of motion

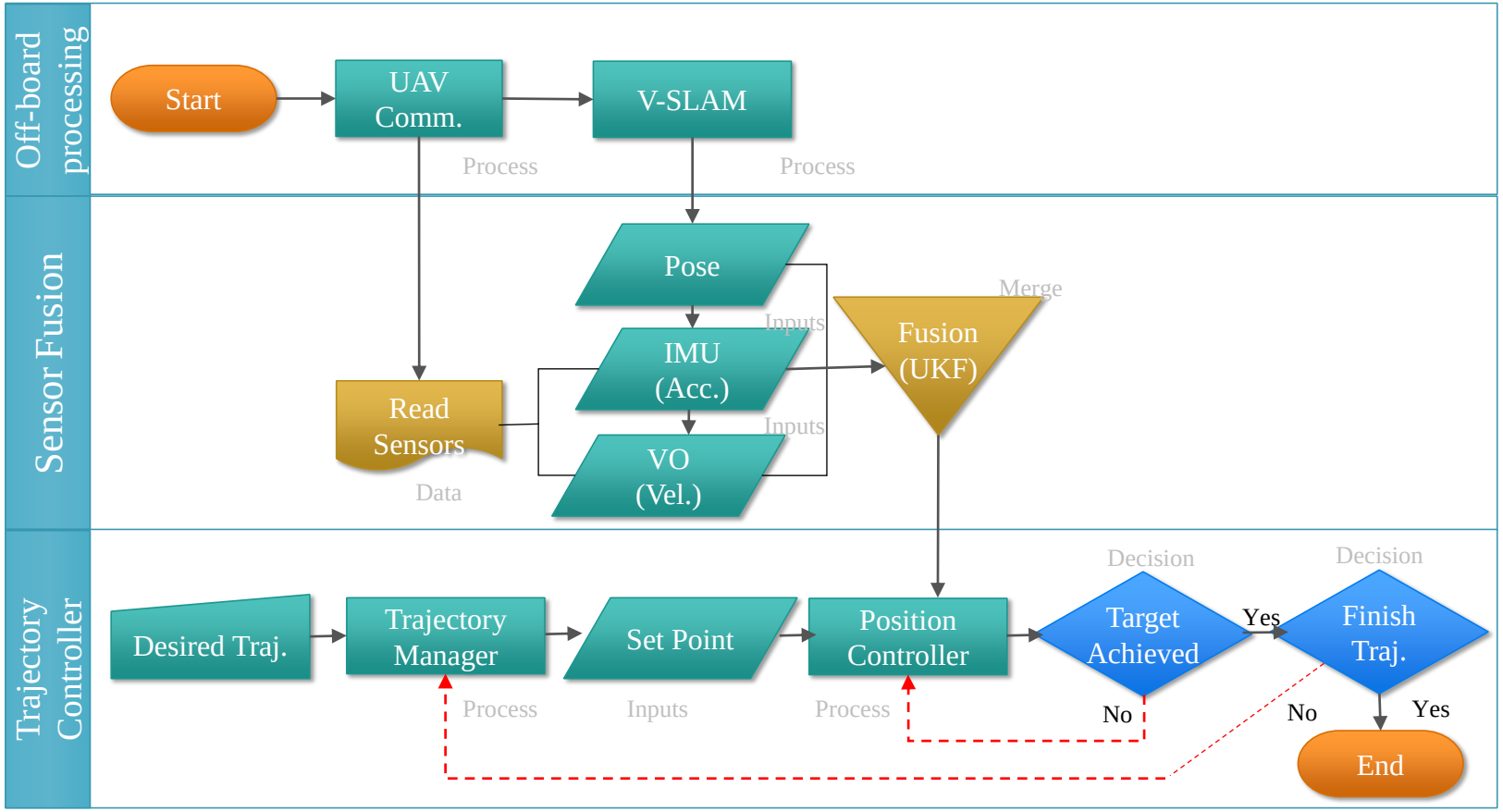


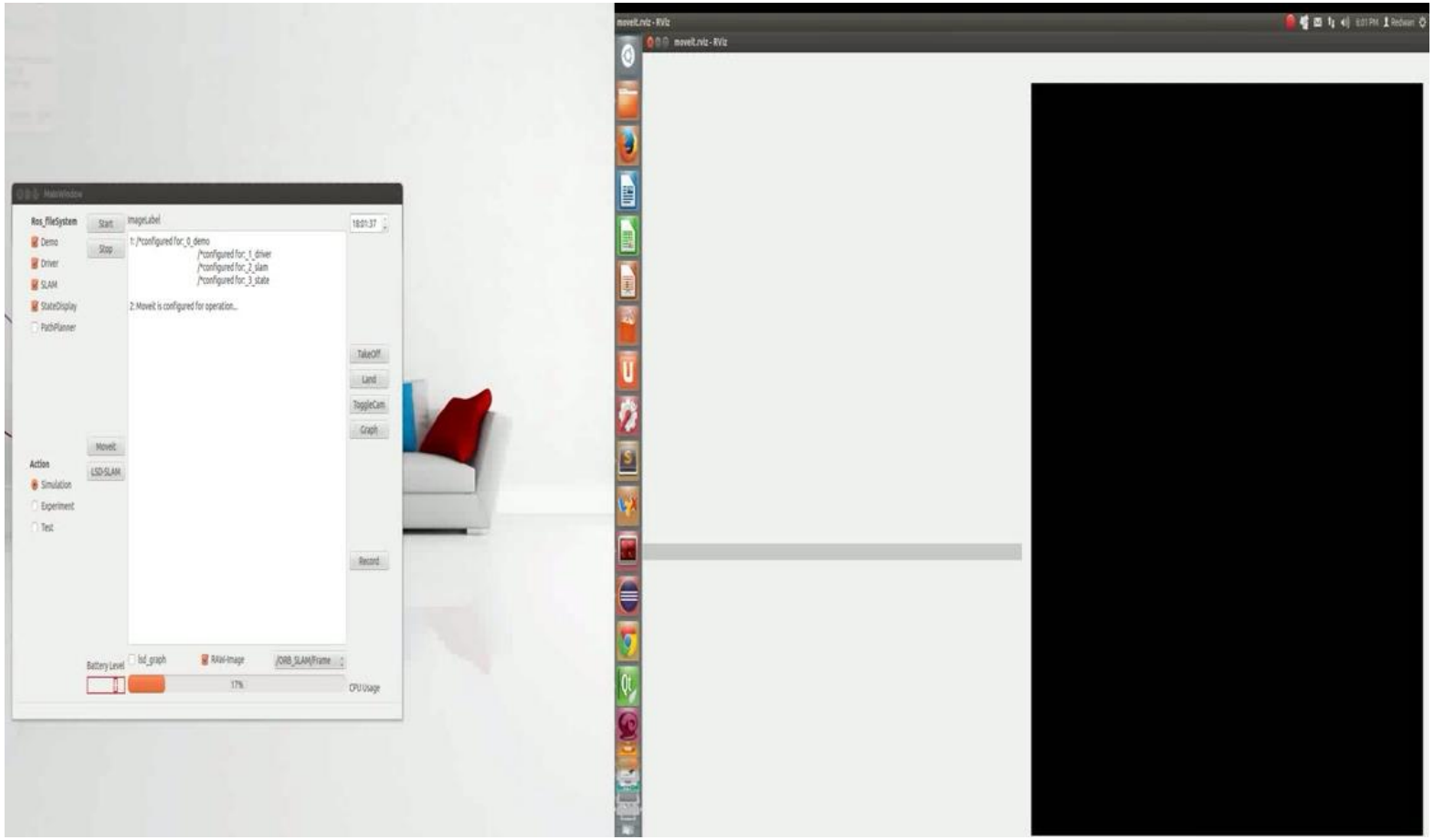




D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. Intl. J. Robot. Research, 2012.

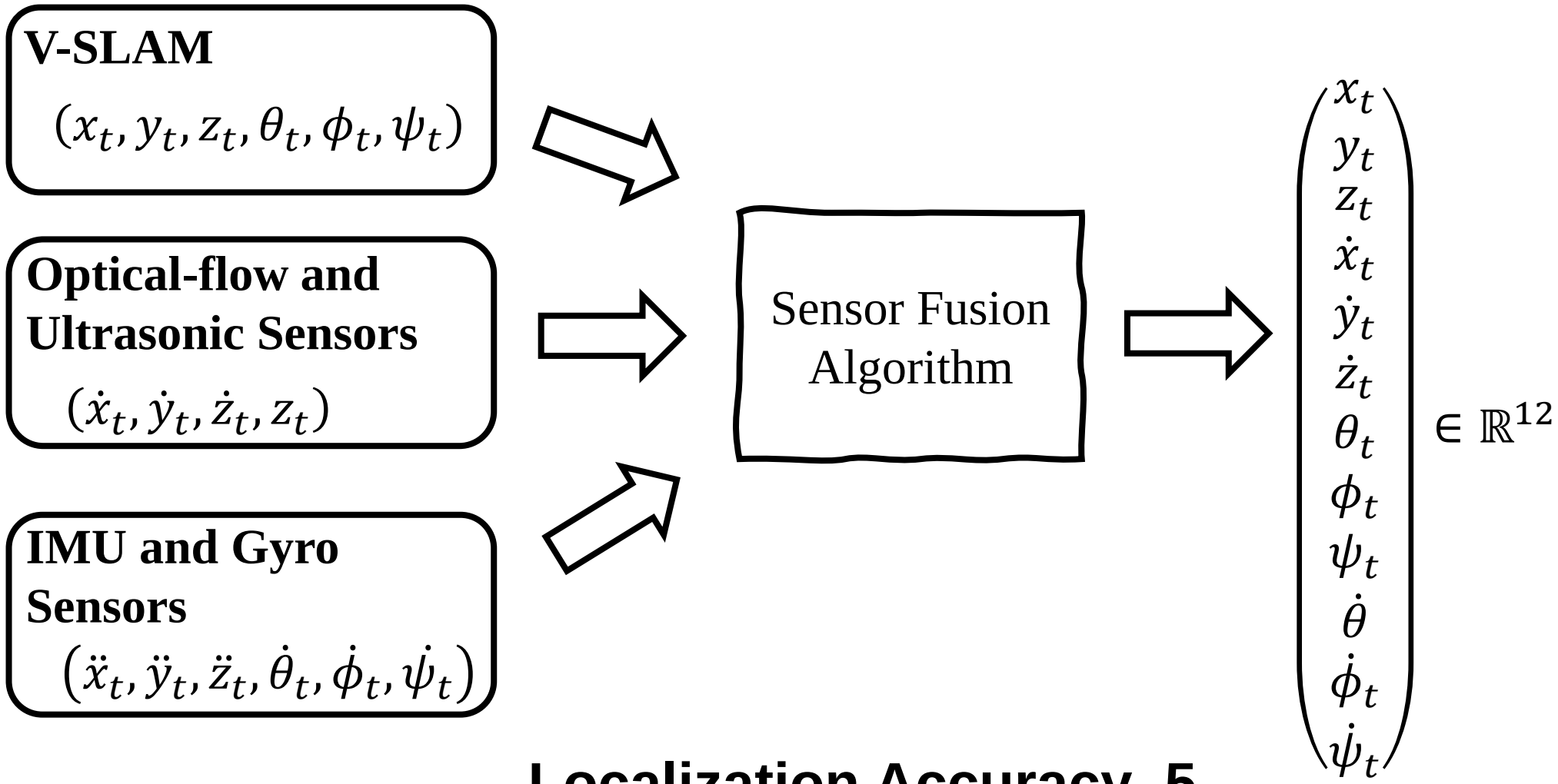
# Perception & Control Modules



# Simultaneously Localization & Mapping



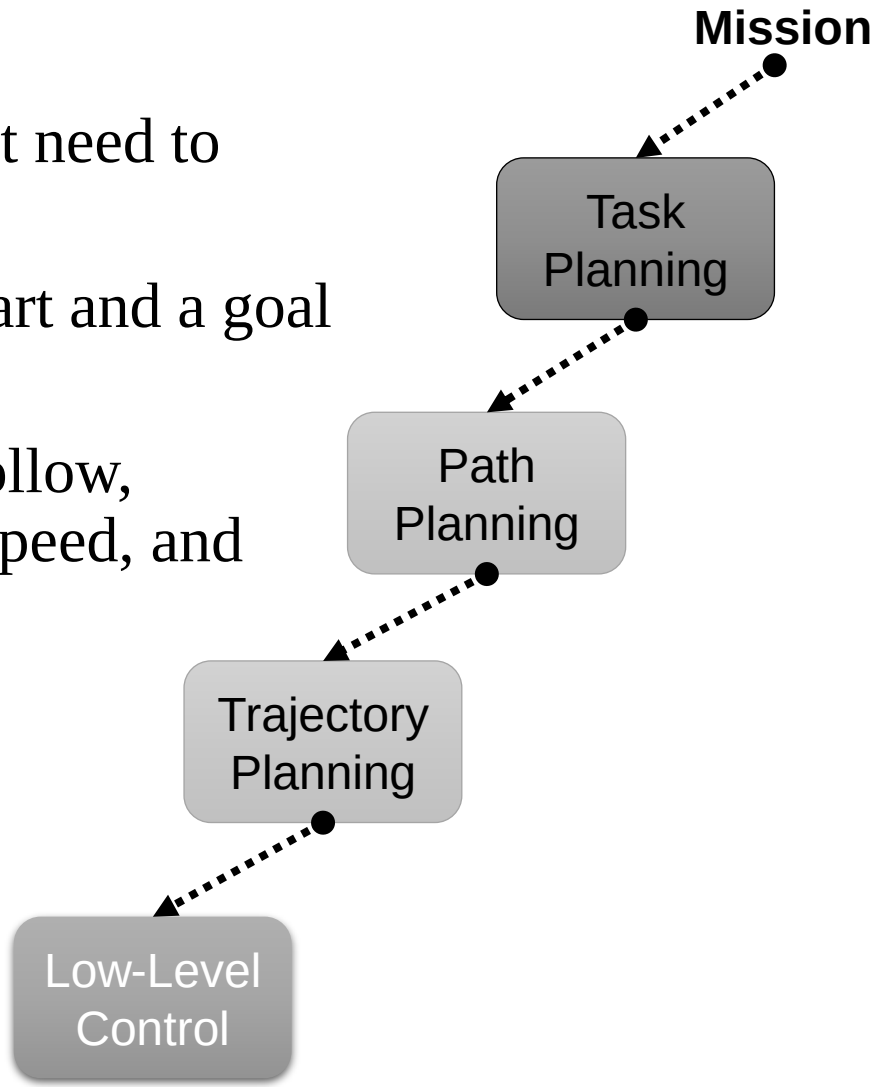
# State Estimation




**Localization Accuracy 5 cm**

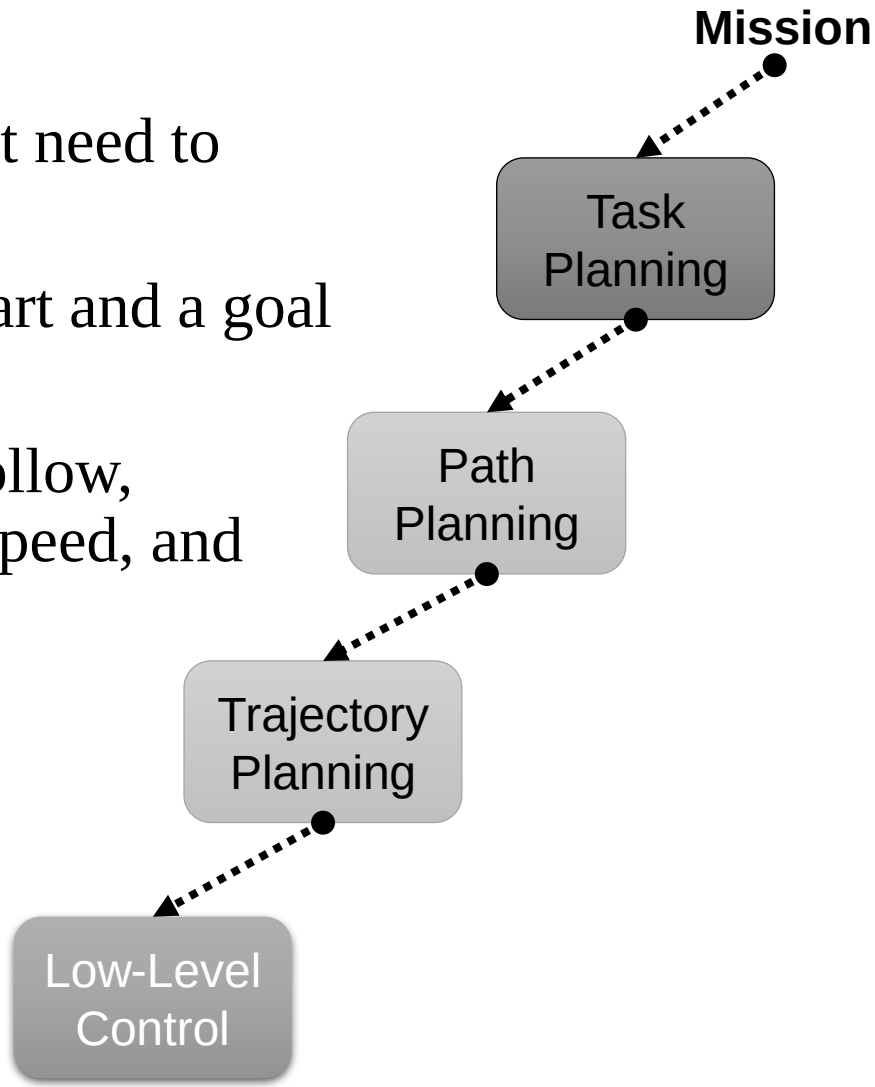
# Planning Hierarchy

- **Task:** a sequence of tasks that robot need to complete its mission
- **Path**: a set of points that links a start and a goal pose only
- **Trajectory**: scheduled motion to follow, including time information (pose, speed, and acceleration)



# Planning Hierarchy

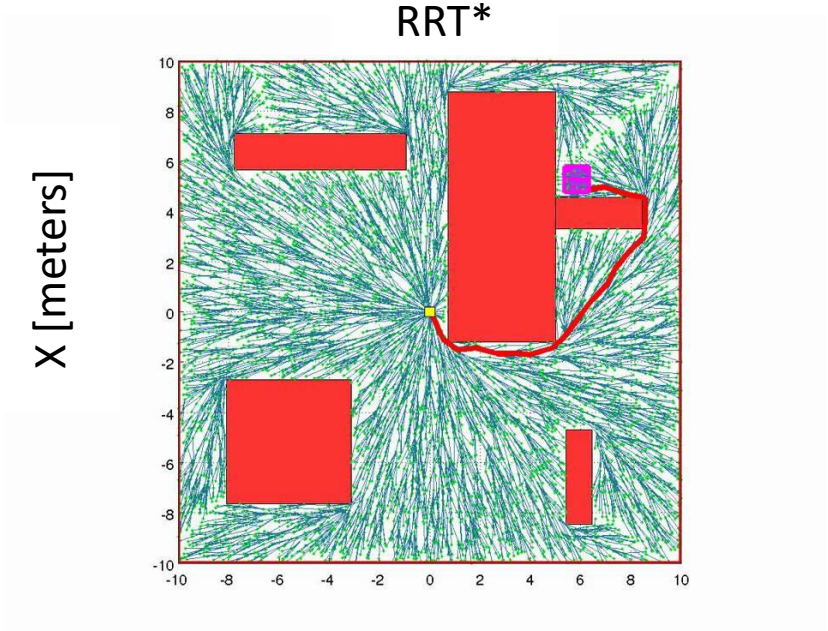
- **Task:** a sequence of tasks that robot need to complete its mission
- **Path**: a set of points that links a start and a goal pose only
- **Trajectory**: scheduled motion to follow, including time information (pose, speed, and acceleration)



# Path Planning: Sampling-based approach

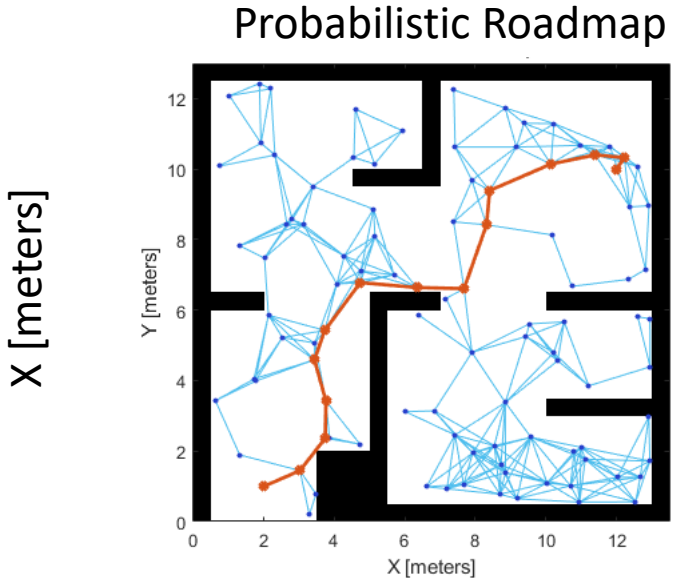
## Single Query Planner

RRT*




## Multiple Query Planner

Probabilistic Roadmap



# Motion Planner types

**Task-Space Planner**

1. Methodology
   1. Task-Space waypoints
   2. Task-Space trajectory
   3. Dynamics simulation
   4. Actuation command
2. Less work at trajectory generation
3. More work at trajectory following
4. Better collision avoidance guarantee

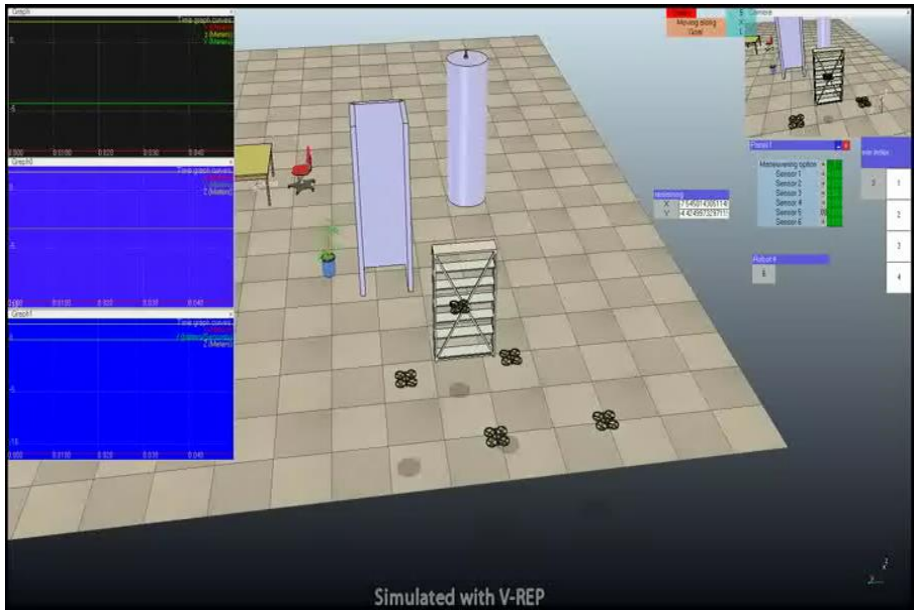
**C-Space Planner**

1. Methodology
   1. Task-Space waypoints
   2. Dynamics simulation
   3. C-Space trajectory
   4. Actuation command
2. More work at trajectory generation
3. Less work at trajectory following
4. Poor collision avoidance guarantee

# Task-space Planner

## Single UAV


## Flock of UAVs




v-rep

# Trajectory Generation Problem

- General setup
  - Start and goal positions
  - Waypoints
  - Smoothing criteria
  - Order of the system (n)
- Problem formulation
  - $x^*(t) = \underset{x(t)}{\operatorname{argmin}} \int_0^T \mathcal{L}(x^n)^2 \, dt$
  - Smoothing constraint

$$\frac{d}{dt}\left(\frac{d\mathcal{L}}{dx}\right) - \frac{d\mathcal{L}}{dx} = 0$$

Euler-Lagrange Equation

Minimum Snap Trajectory (4th order derivatives)

$$x^{*(t)} = \underset{x(t)}{\operatorname{argmin}} \int_0^T (\ddddot{x})^2 \, dt$$

Minimum Jerk Trajectory (3rd order derivatives)

$$x^{*(t)} = \underset{x(t)}{\operatorname{argmin}} \int_0^T (\dddot{x})^2 \, dt$$

Minimum Acceleration Trajectory (2nd order derivatives)

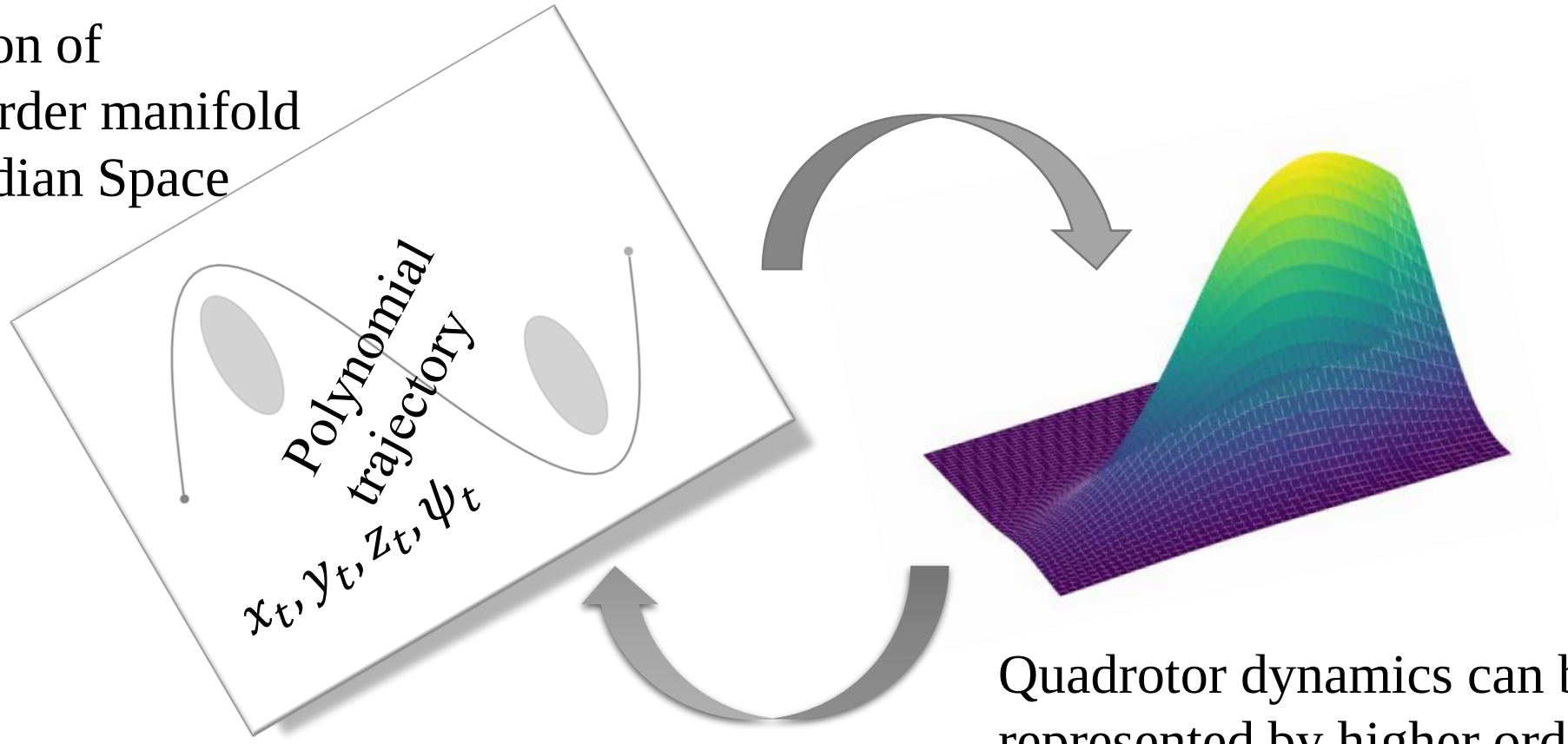
$$x^{*(t)} = \underset{x(t)}{\operatorname{argmin}} \int_0^T \ddot{x}^2 \, dt$$

Minimum Velocity Trajectory (1st order derivatives)

$$x^{*(t)} = \underset{x(t)}{\operatorname{argmin}} \int_0^T \dot{x}^2 \, dt$$

# Trajectory generation

Projection of
higher order manifold
to Euclidian Space

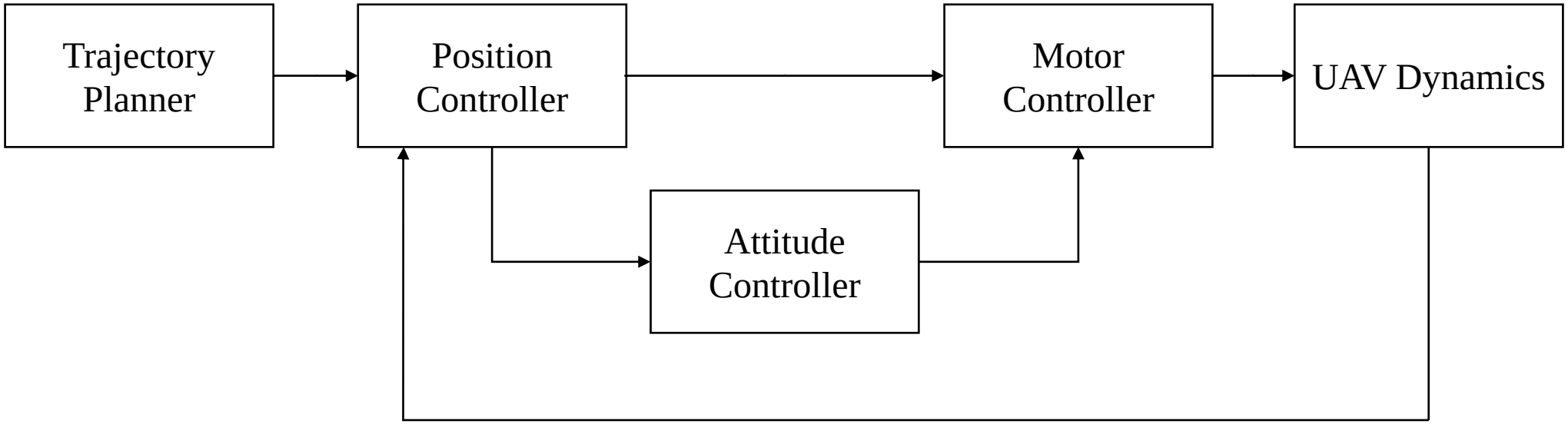



Quadrotor dynamics can be represented by higher order manifold (12 D)

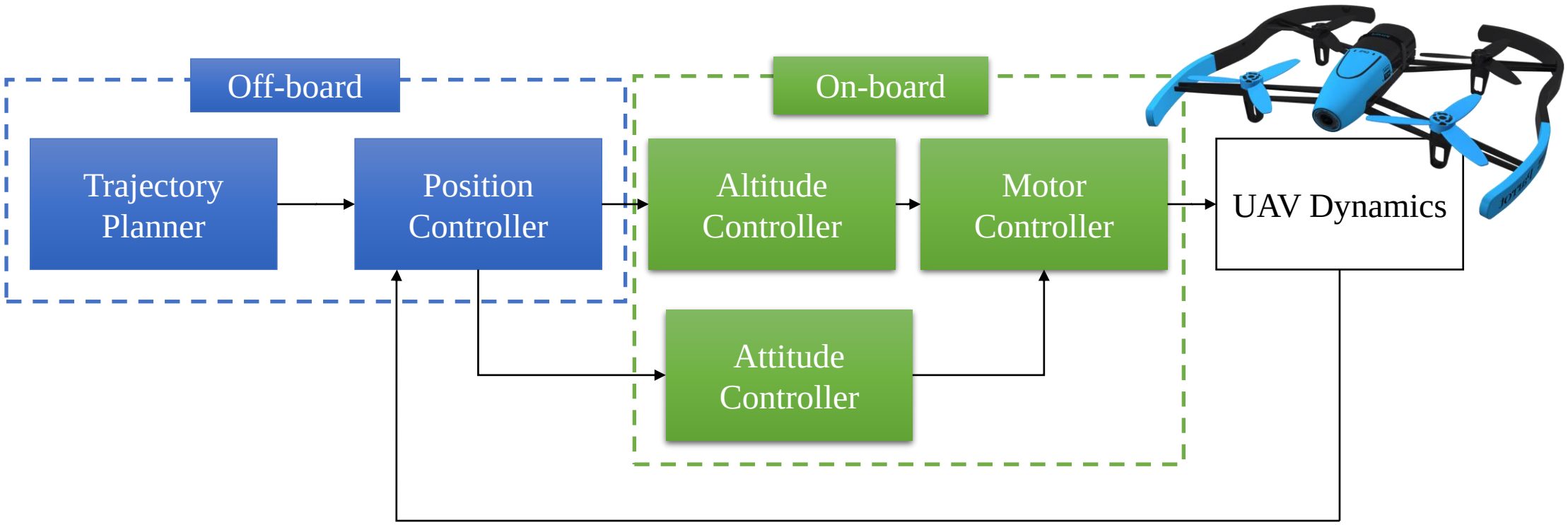
# Polynomial Trajectory Planner

- Is natural choice for highly dynamic vehicles and robots
- Can be efficiently obtained as the solution to a Quadratic Program (QP)
- Minimizes cost function of path derivatives
- Polynomials can be jointly optimized while maintaining continuity of path derivatives
- Ensure smooth motions
- Don’t require step inputs to the vehicle’s actuators

# Trajectory Following

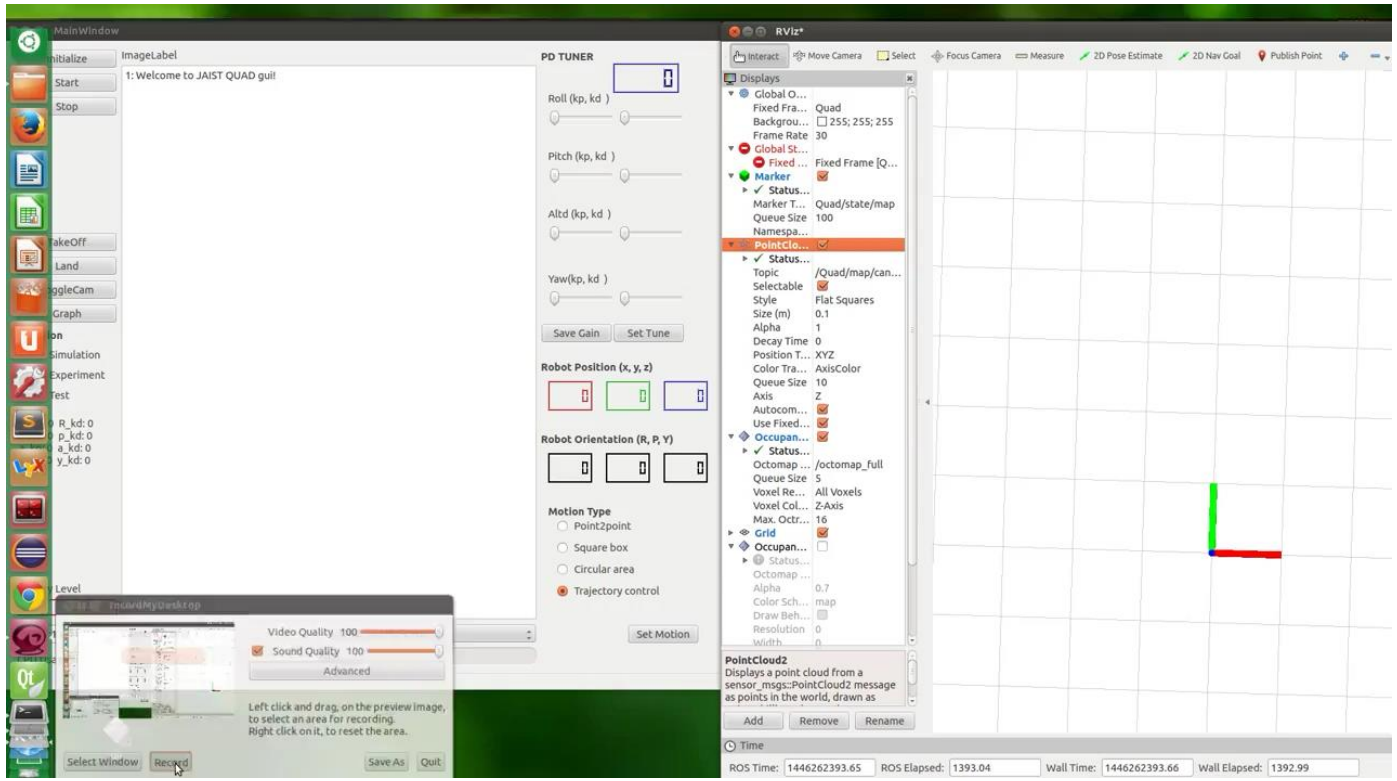


# Trajectory Following



# Polynomial Trajectory Planner

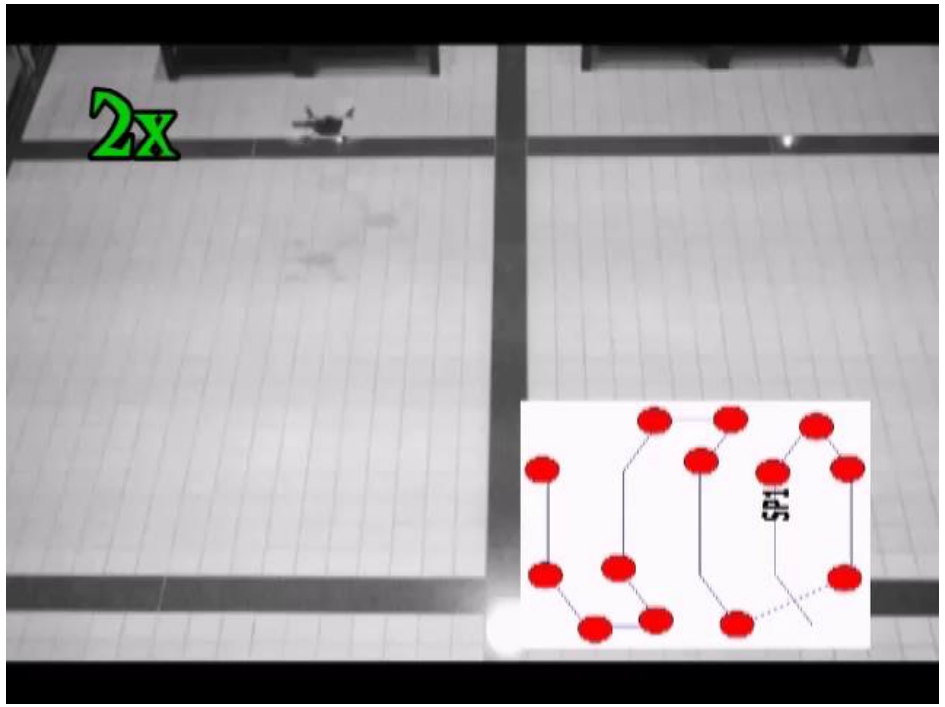
- Cubic polynomial
  - Define position and velocity at start and end path for each segment
  - 4 coefficients
  - Minimum acceleration trajectory planner
- Quantic polynomial
  - Define position, velocity and acceleration at start and end path for each segment
  - 6 coefficients
  - Minimum jerk trajectory planner

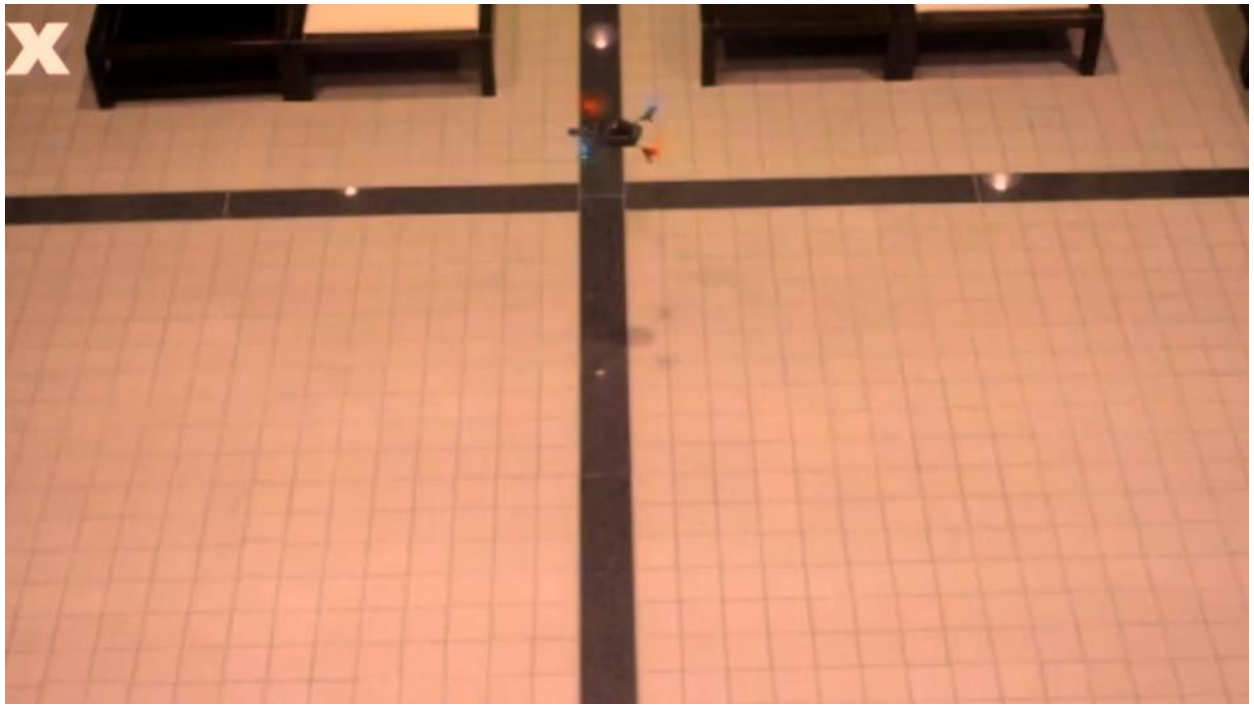



Higher polynomial order provides smooth motion

ROS

# Autonomous Navigation



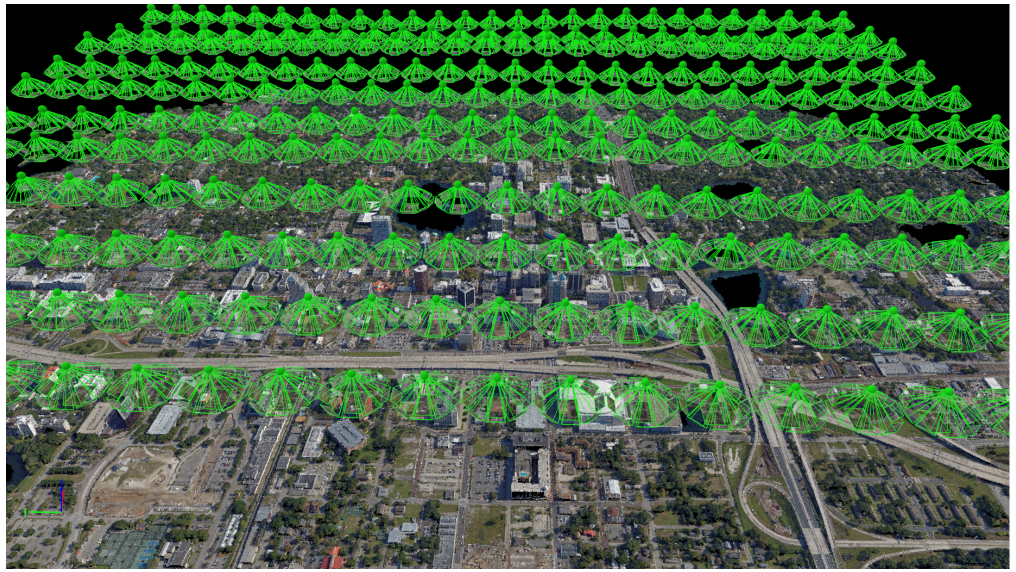
Minimum acceleration trajectory

Trajectory tracking accuracy  15 cm

ROS

# Multi-robot Planning for Smart Agriculture

**Coverage using multi-UAV system:** a cooperative team of UAVs coordinate their locations to achieve optimal coverage of a given area.



**Coverage metric:** Coverage is measured in terms of the UAVs' ability to collectively capture images for analyzing yields across an agricultural field.

# Challenges in Farmland Coverage

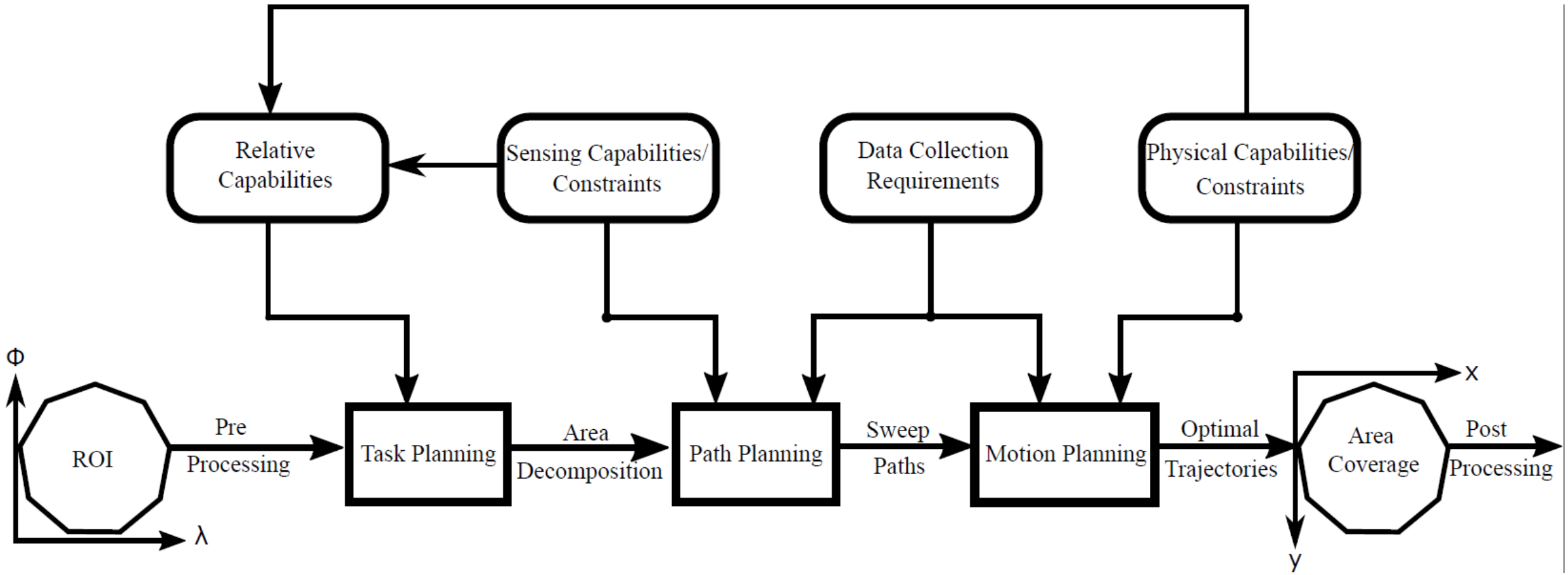
**Single-UAV system:**

- ❑ does not have enough resources (e.g., battery) to cover a large farmland
- ❑ takes long time to cover the farmland

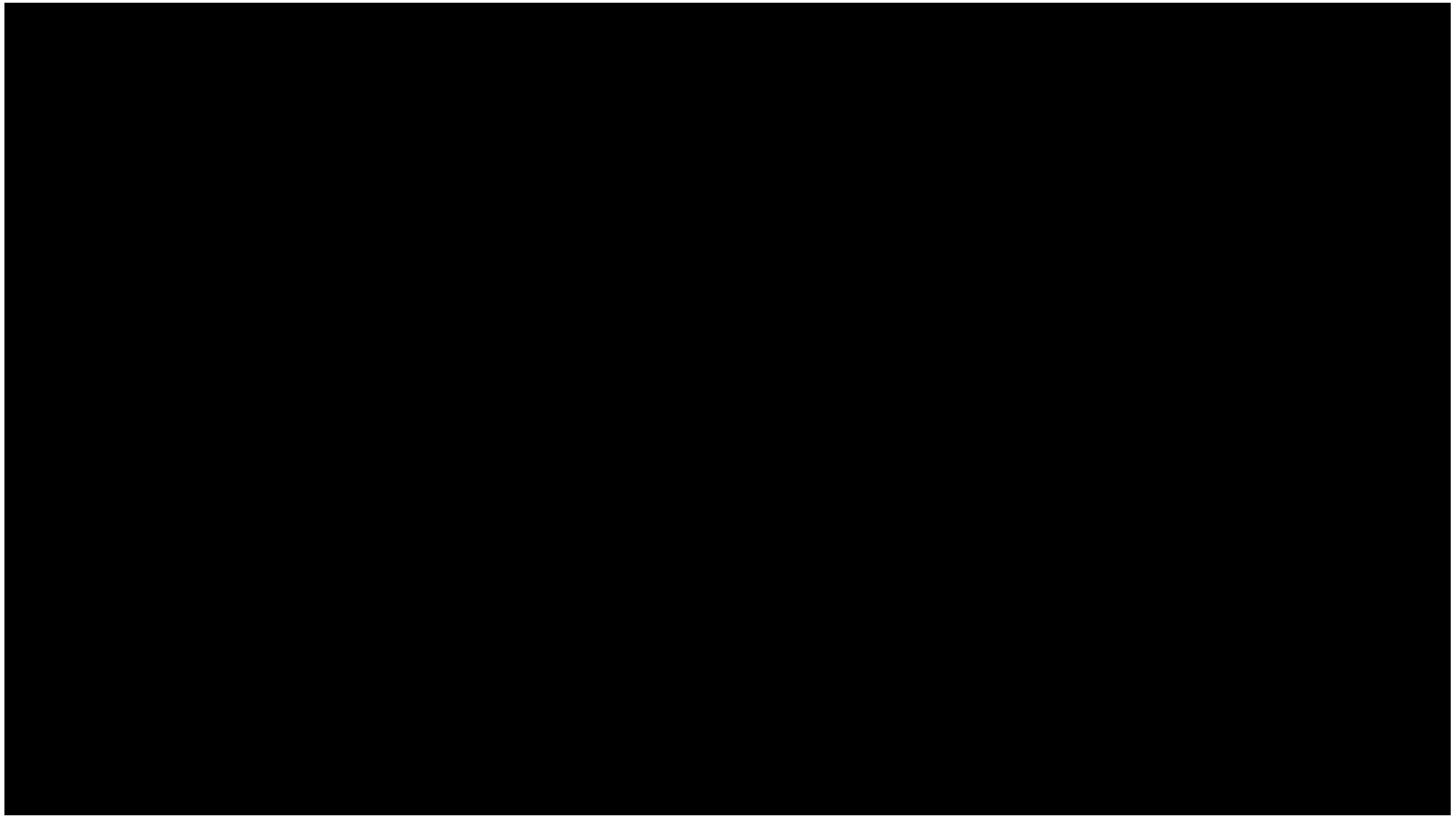
**Multi-UAV System:**

- ❑ demands an efficient tasking and coordination mechanism
- ❑ needs to consider relative capabilities, sensing capabilities, data collection requirements, and physical capabilities
- ❑ conventional methods are not computationally efficient for solving large-scale coverage planning problem.

# Proposed Framework



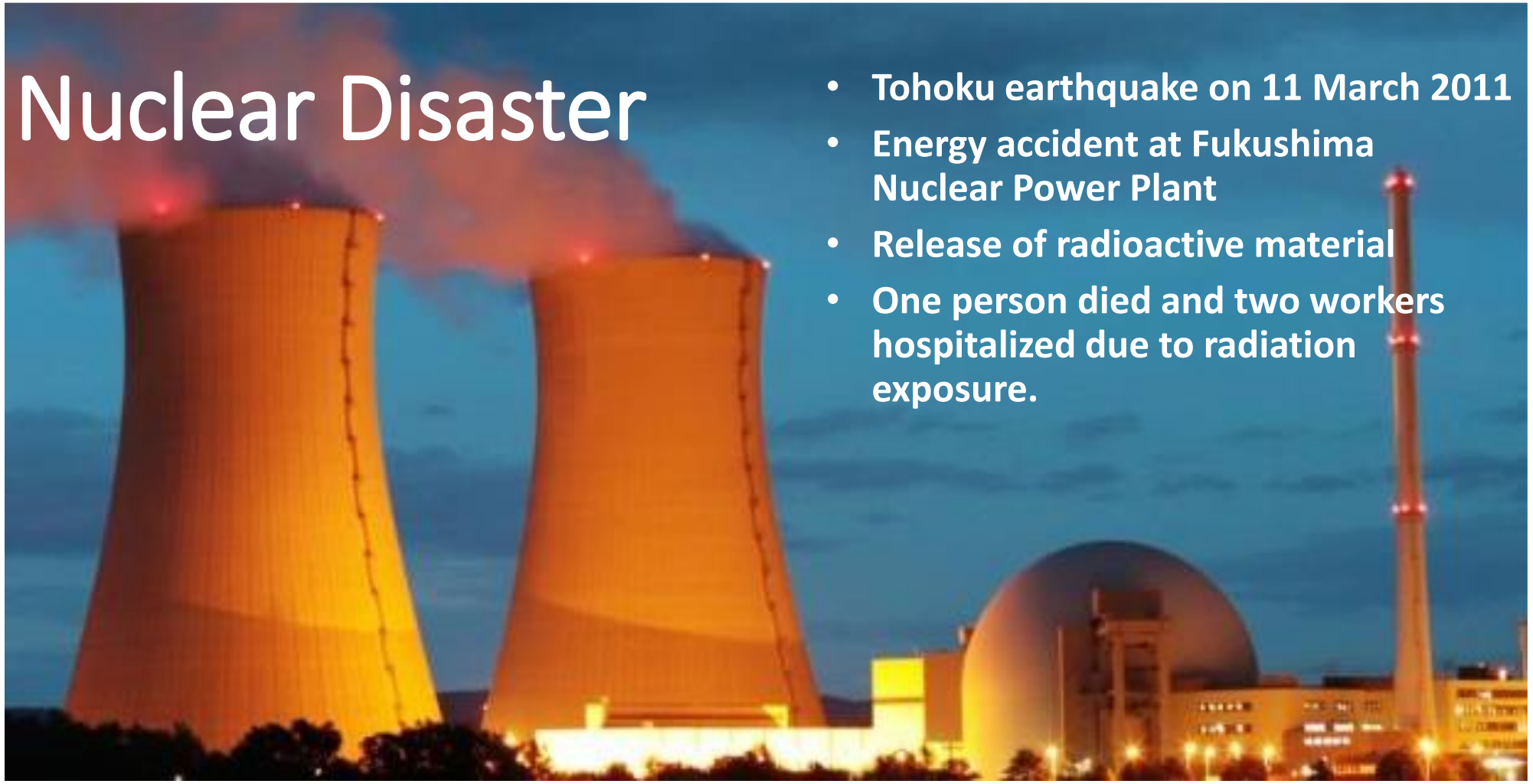
# Simulation Results



# Goal Free Planning

- Radioactive source localization using unmanned aerial vehicles
- Optimal multi robot planning for port facility monitoring
- Distributed decentralized multi-robot planning for monitoring large-scale environments

# Nuclear Disaster



- Tohoku earthquake on 11 March 2011
- Energy accident at Fukushima Nuclear Power Plant
- Release of radioactive material
- One person died and two workers hospitalized due to radiation exposure.

*"If you can reduce the initial response by one day, you can reduce the overall recovery by thousand days"*

*-- Robin Murphy on May 2015*

# UAVs as fast responders

- Search and rescue
- Preliminary radiation map
- Helping rescue workers to save more lives

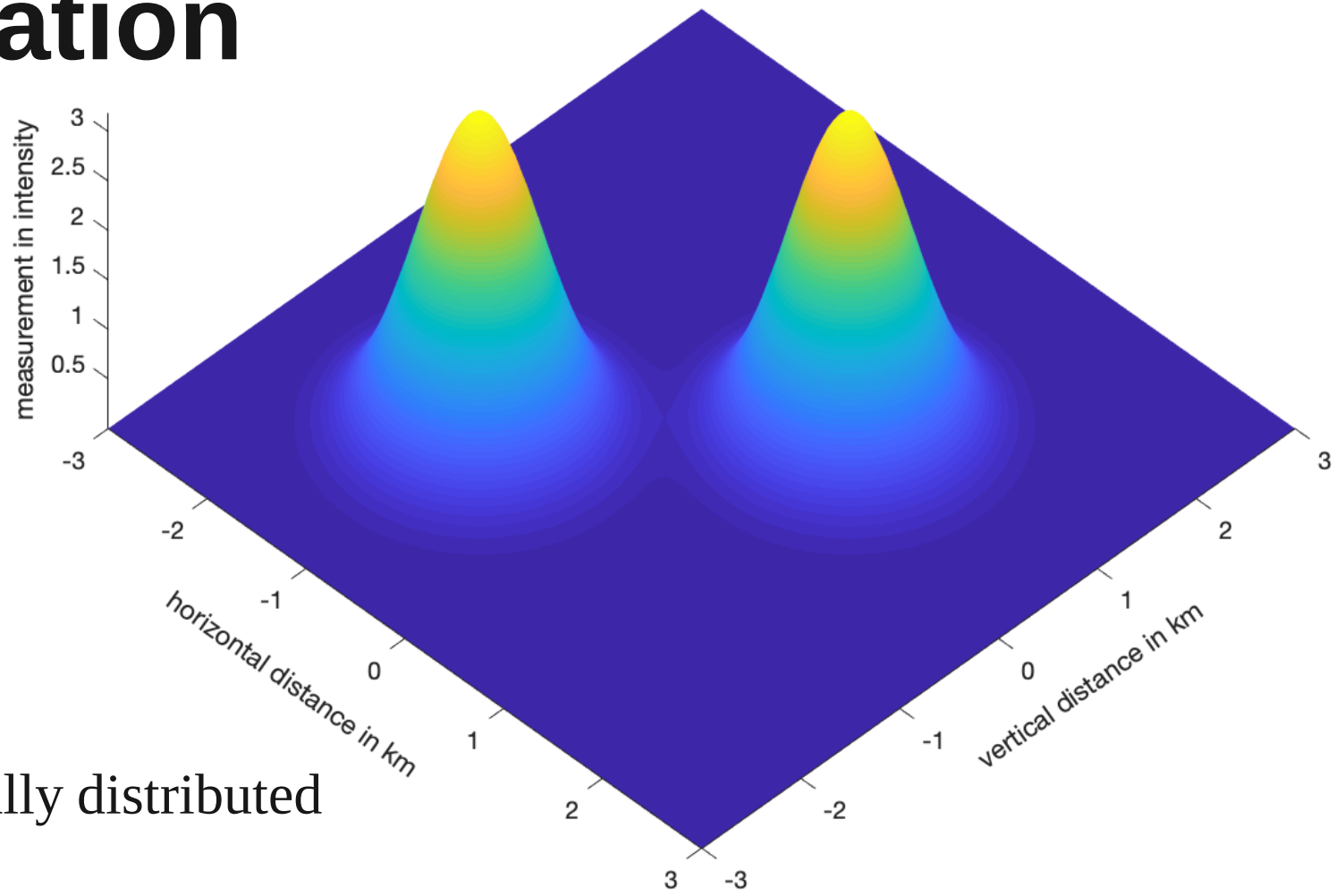

People

Infrastructure

Environment

# Source Localization

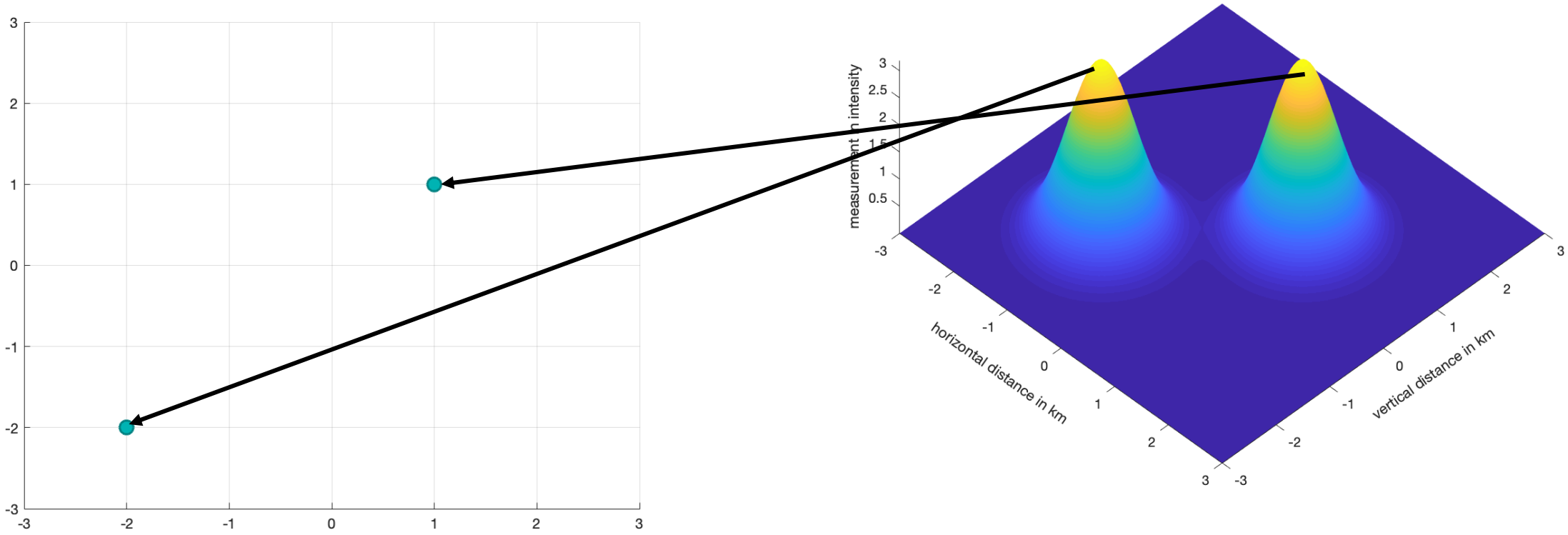
1. How many sources?
2. Where are the sources?



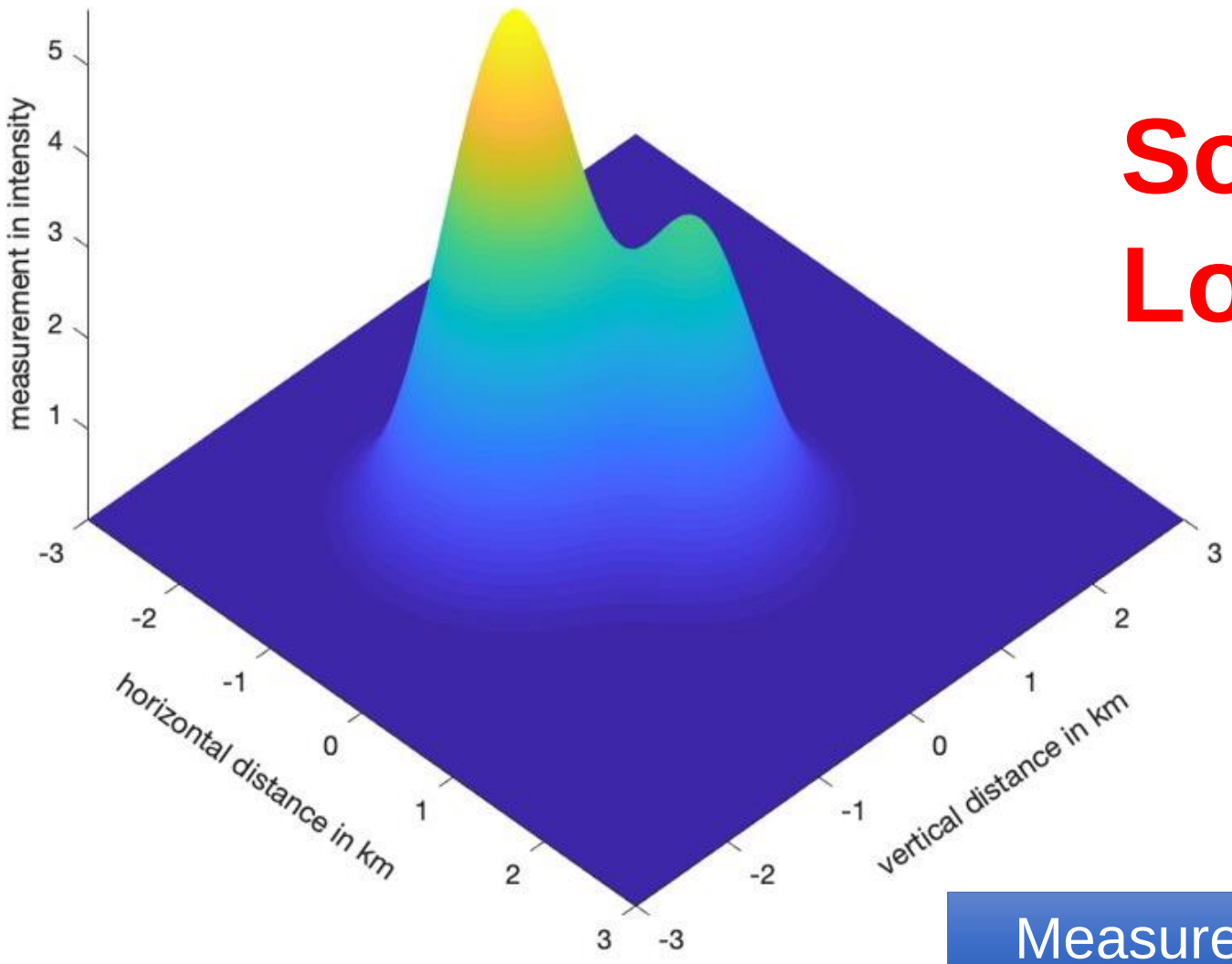

**Challenges:**

- Measurements are spatially distributed
- Require in-situ sensing

# Point Source




Point sources can be mapped uniquely under Idealize conditions

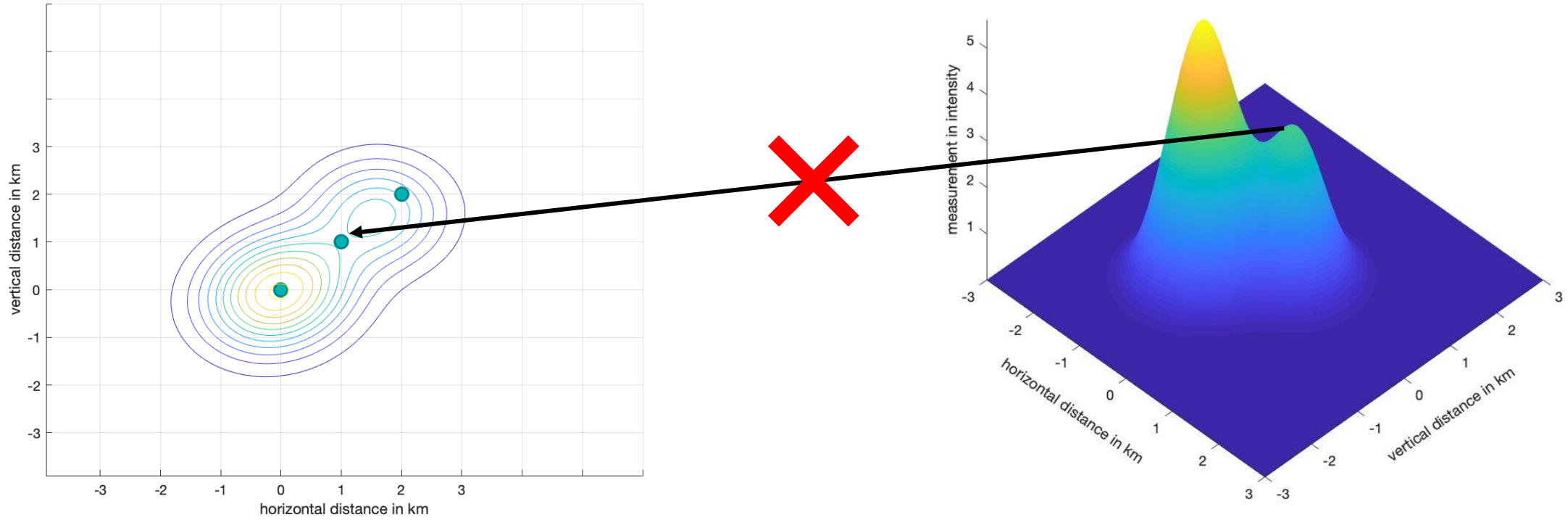



# Source Localization

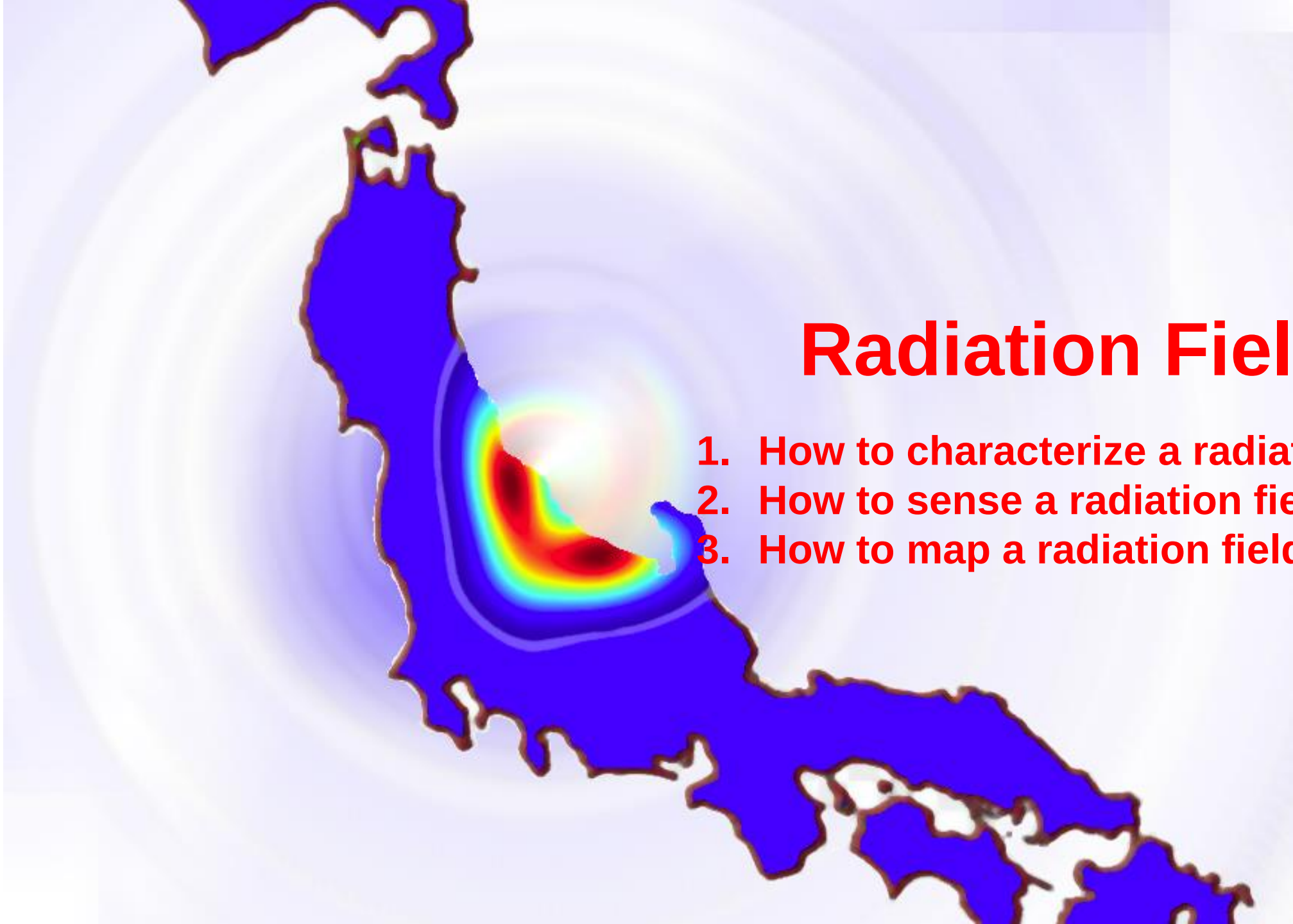
1. How many sources?
2. Where are the sources?

Measurement of the radiation field is

# Cumulative Radiation Effects



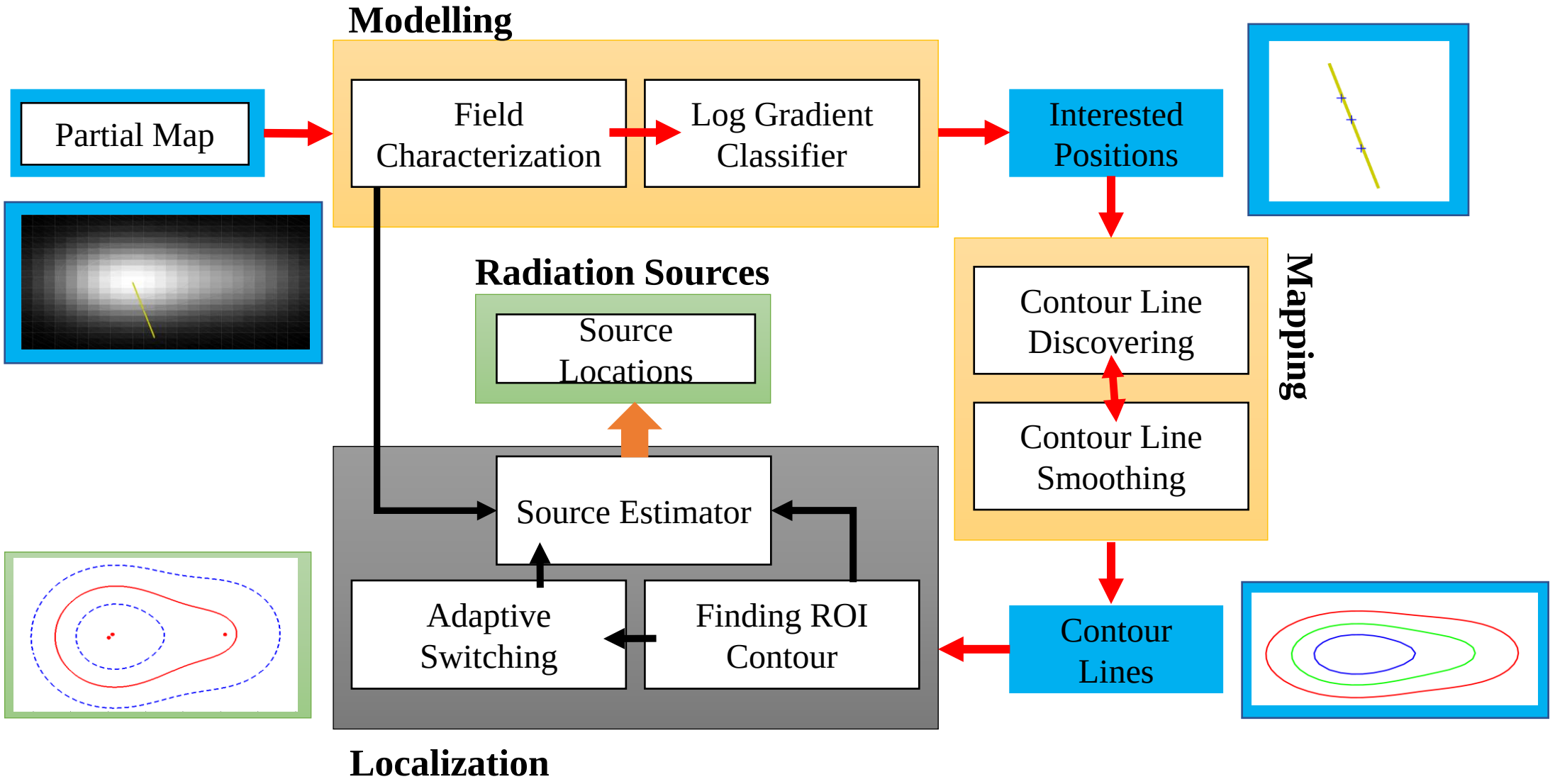

How to separate sources when cumulative radiation effects exist?



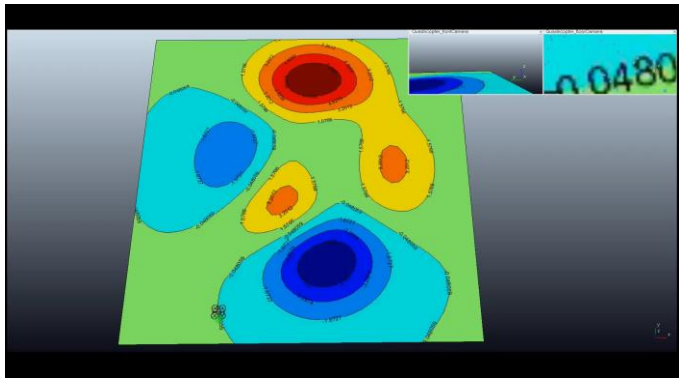
# Radiation Fields

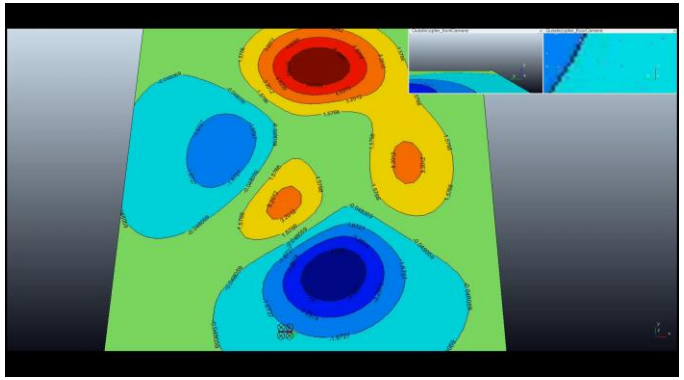
1. How to characterize a radiation field?
2. How to sense a radiation field?
3. How to map a radiation field?

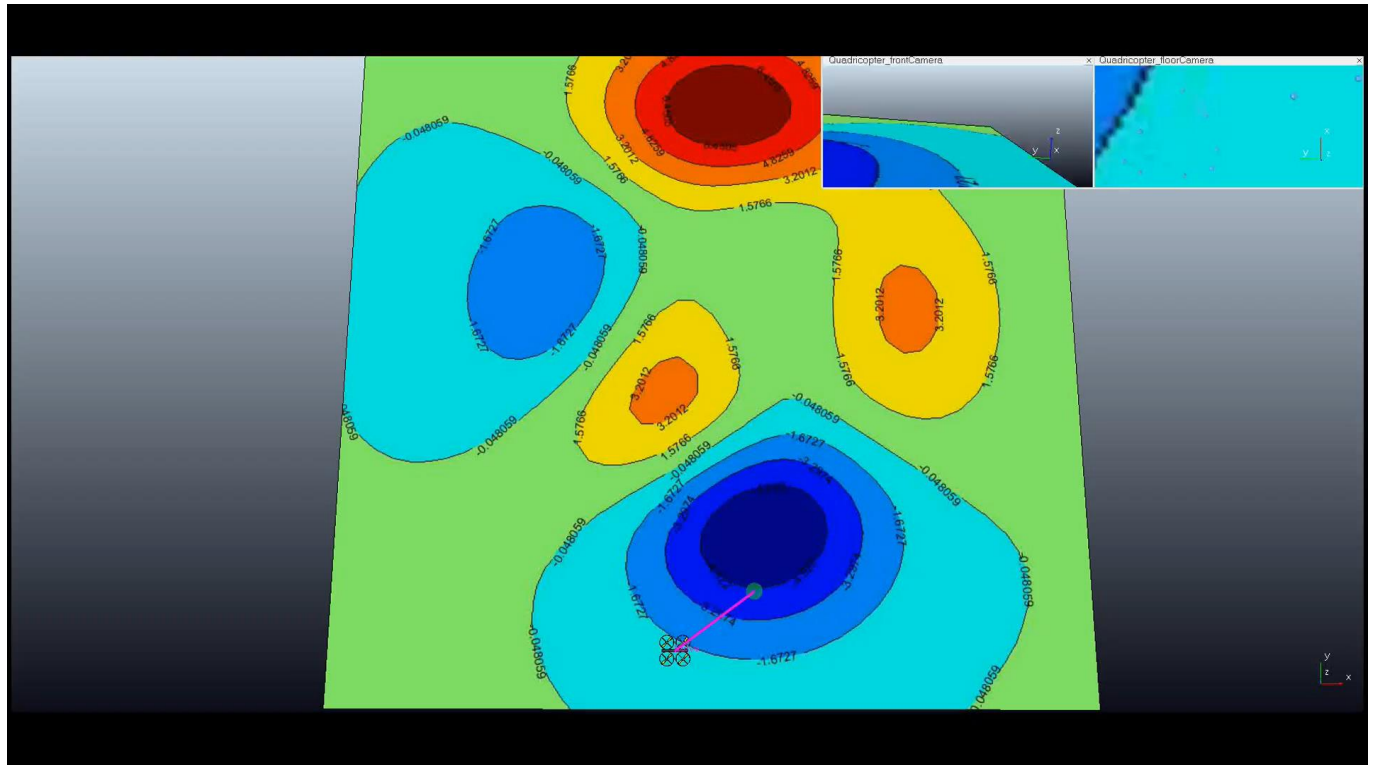
# Radioactive Source Localization



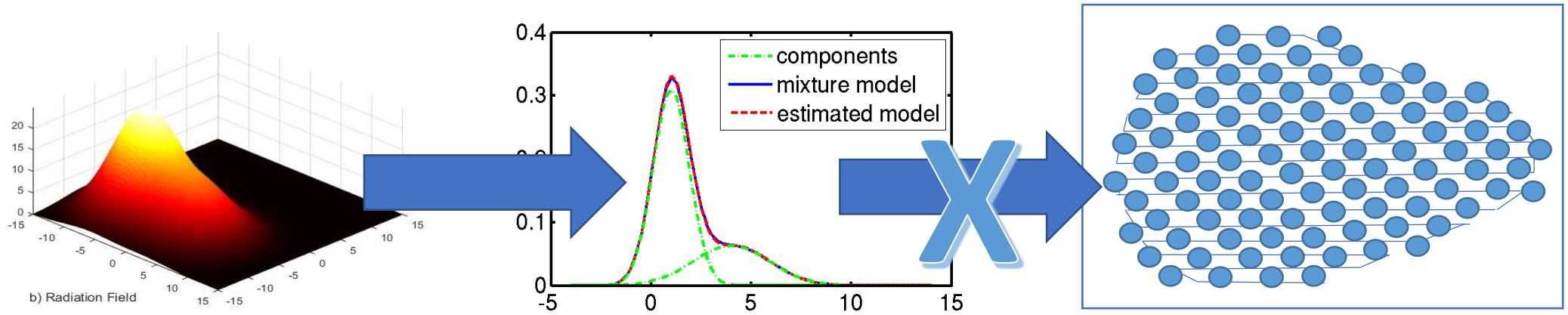
# Field Exploration Strategy







# Density Estimator

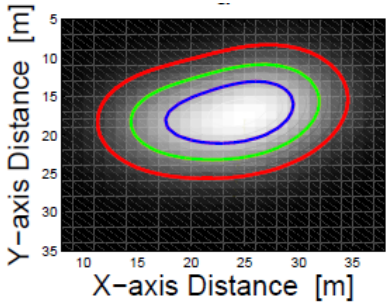



Measurement Domain

Probability Domain

Spatial Domain

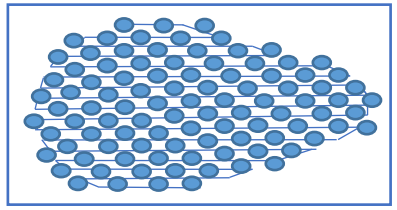
## Solving Inverse Problem




Input

Sample points

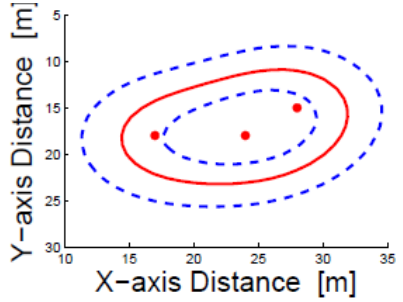
$x_{si} \in \mathbb{R}^2$



Variational Bayesian

Cluster Mean

$x_{\mu}^{j} \in \mathbb{R}^2$




Output

# Port Facility Monitoring

- Enhancing security in ports demands periodic infrastructure monitoring.
- Port infrastructure monitoring involves detecting spatial events.
- Autonomous multi-robot navigation is useful for efficient data collection about events.

**Multi-Robot Information Gathering Problem**

# Multi-Robot Information Gathering

Given a GMM model parameters , n number of robots, m number of information gathering stages, the informative trajectory planning for n robots under energy budgets and kinematic constraints needs solving the following maximization problem:

$$\arg\max_{\tau_{i,j}} \sum_{j=1}^{m} \sum_{i=1}^{n} I(\tau_{i,j}|\Theta)$$

subject to

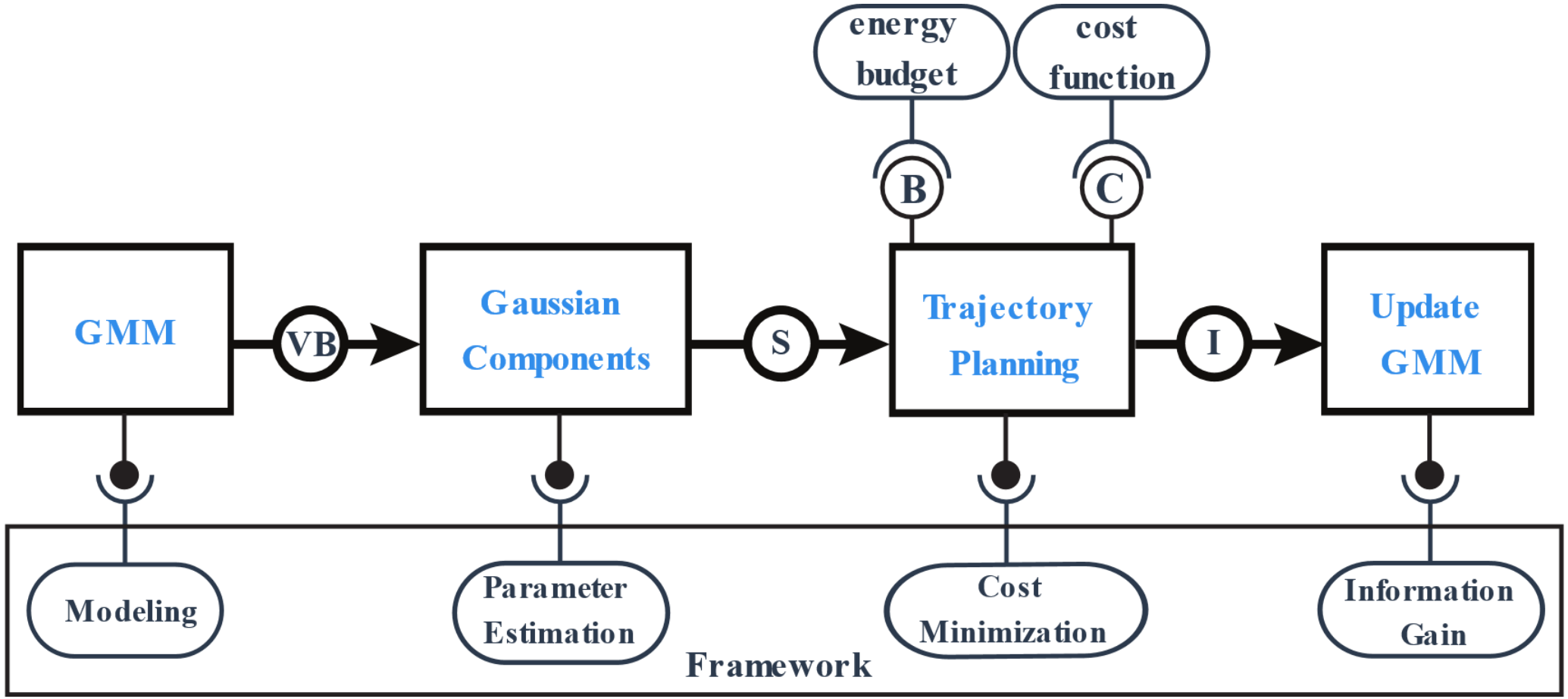
$$x_{t+1} = f(x_t, u_t), \longleftarrow \text{Kinematic constraint}$$

$$\forall t,\ \forall i : c(\tau_{i,j}) \leq B \text{ where } c : \Psi \to \mathbb{R}^+, \longleftarrow \text{Resource constraint}$$

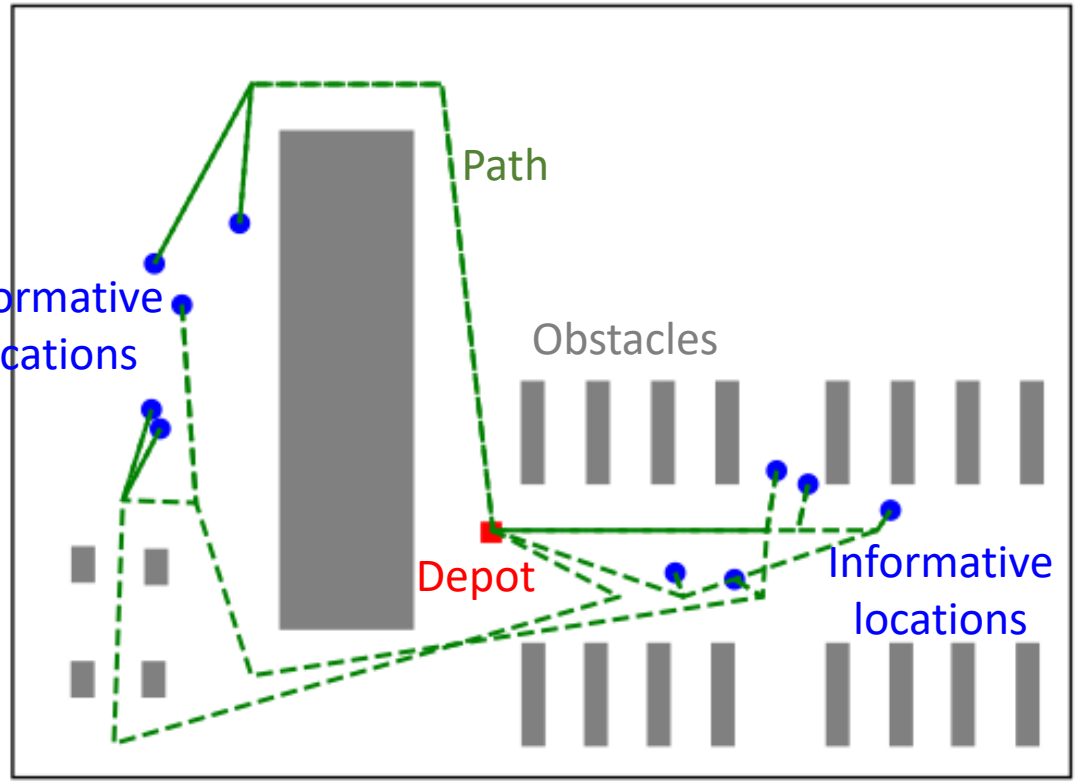
$$\forall t,\ \forall (i,k),\ \forall j : ||\tau_{i,j}(t) - \tau_{k,j}(t)||_2 \geq \delta \text{ and } i \neq k, \longleftarrow \text{Dynamic collisions}$$

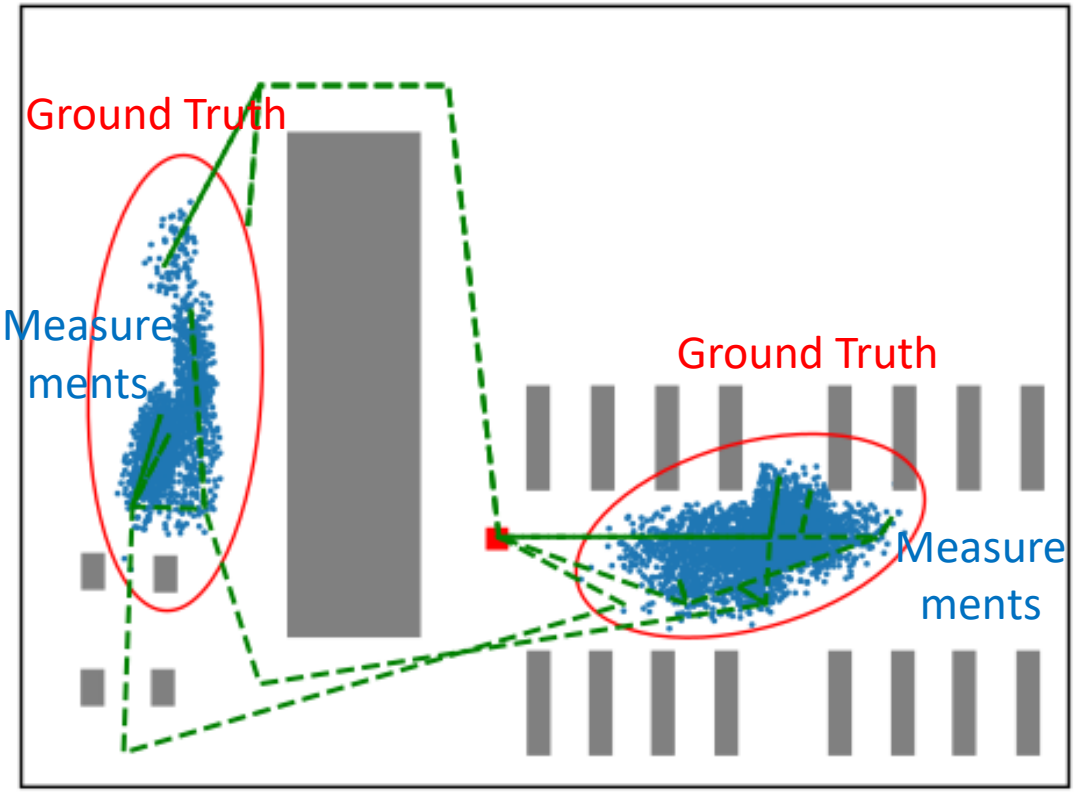
$$\forall p,\ \forall t : ||\tau_{i,j}(t) - \mathcal{O}_p||_2 \geq \delta, \longleftarrow \text{Static collisions}$$

# Proposed Framework

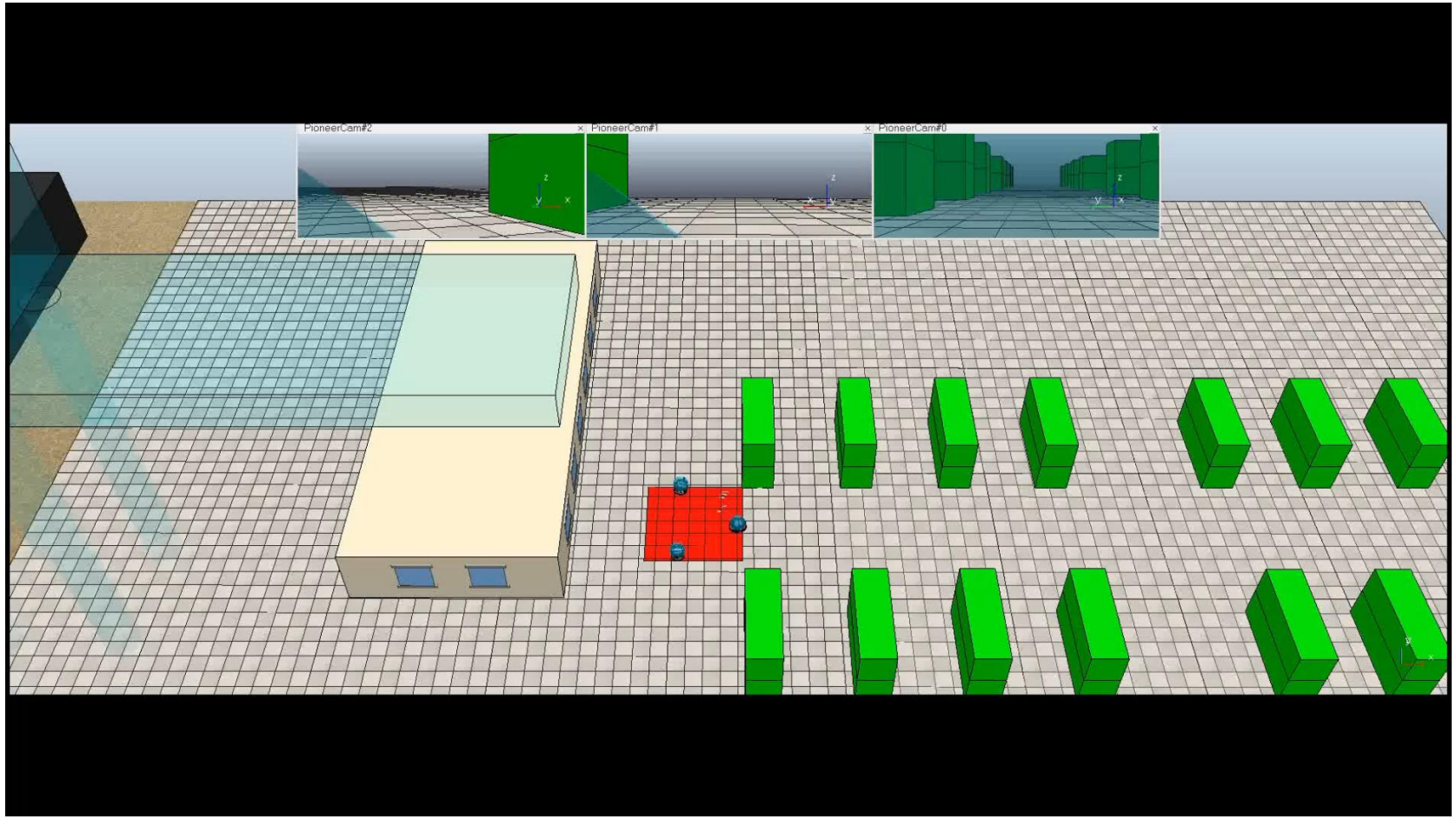


# Simulation Results

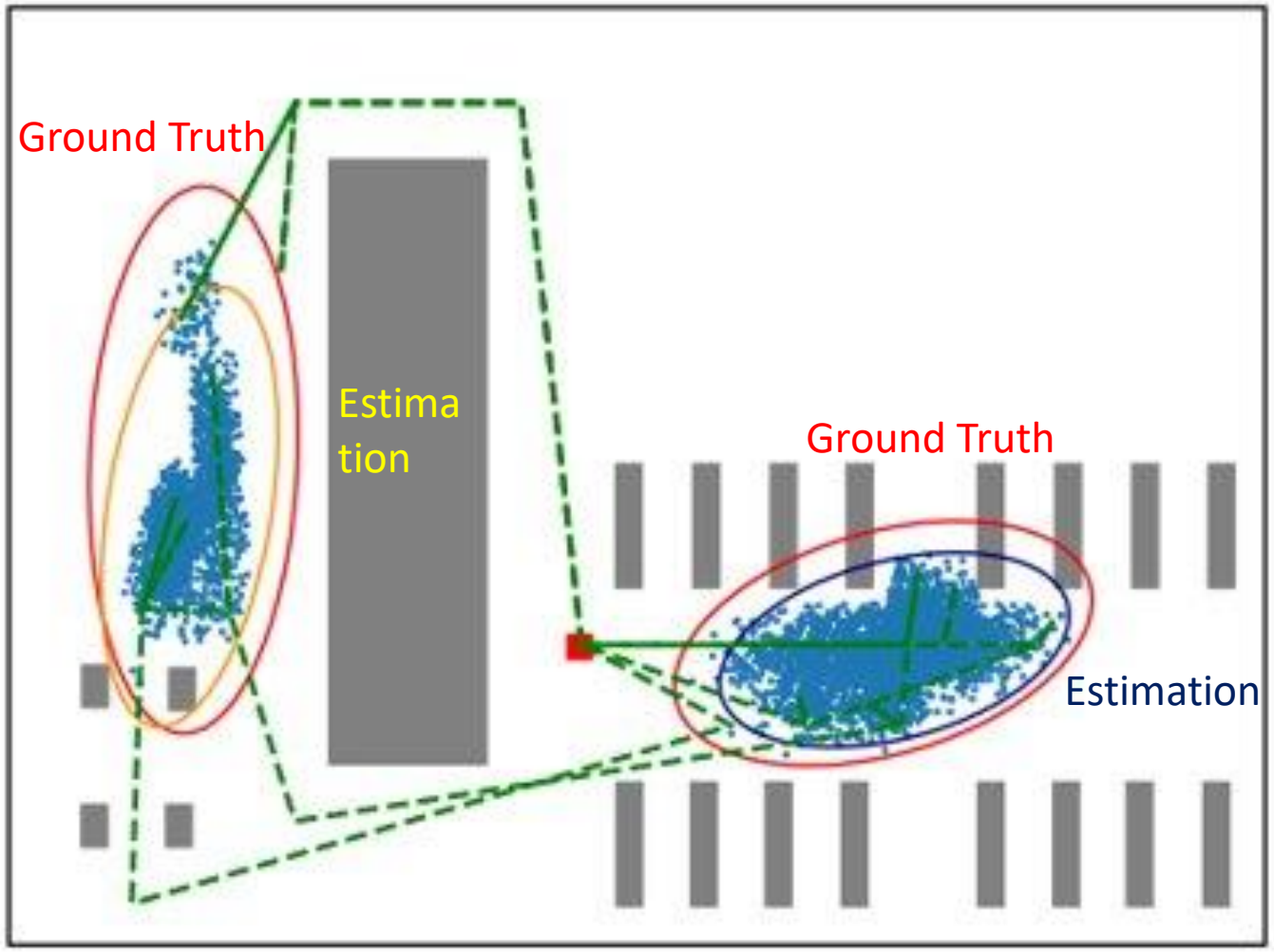


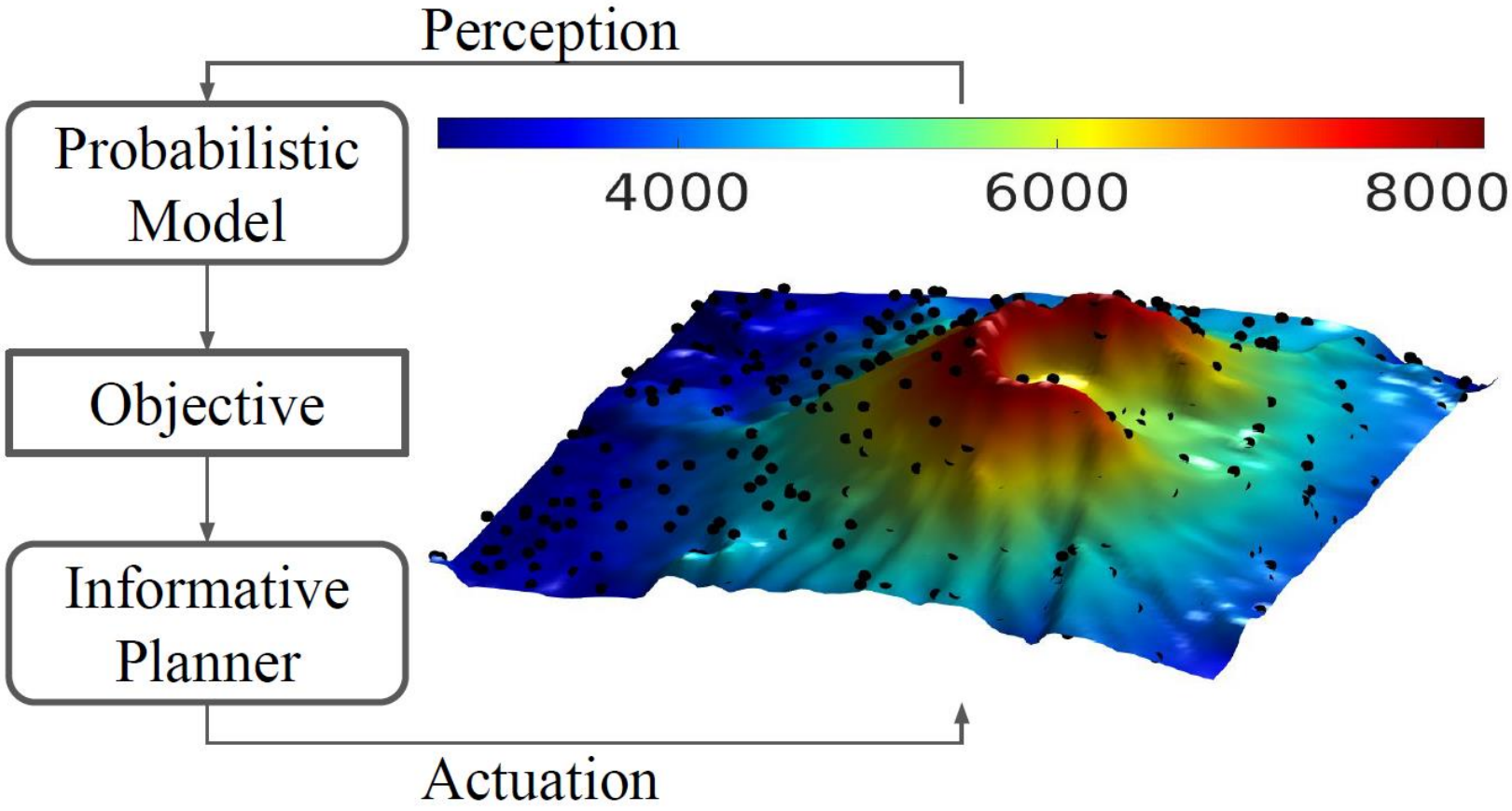
# Realistic Simulation



# Simulation Results (Cont..)



# A Generic Information Gathering Framework



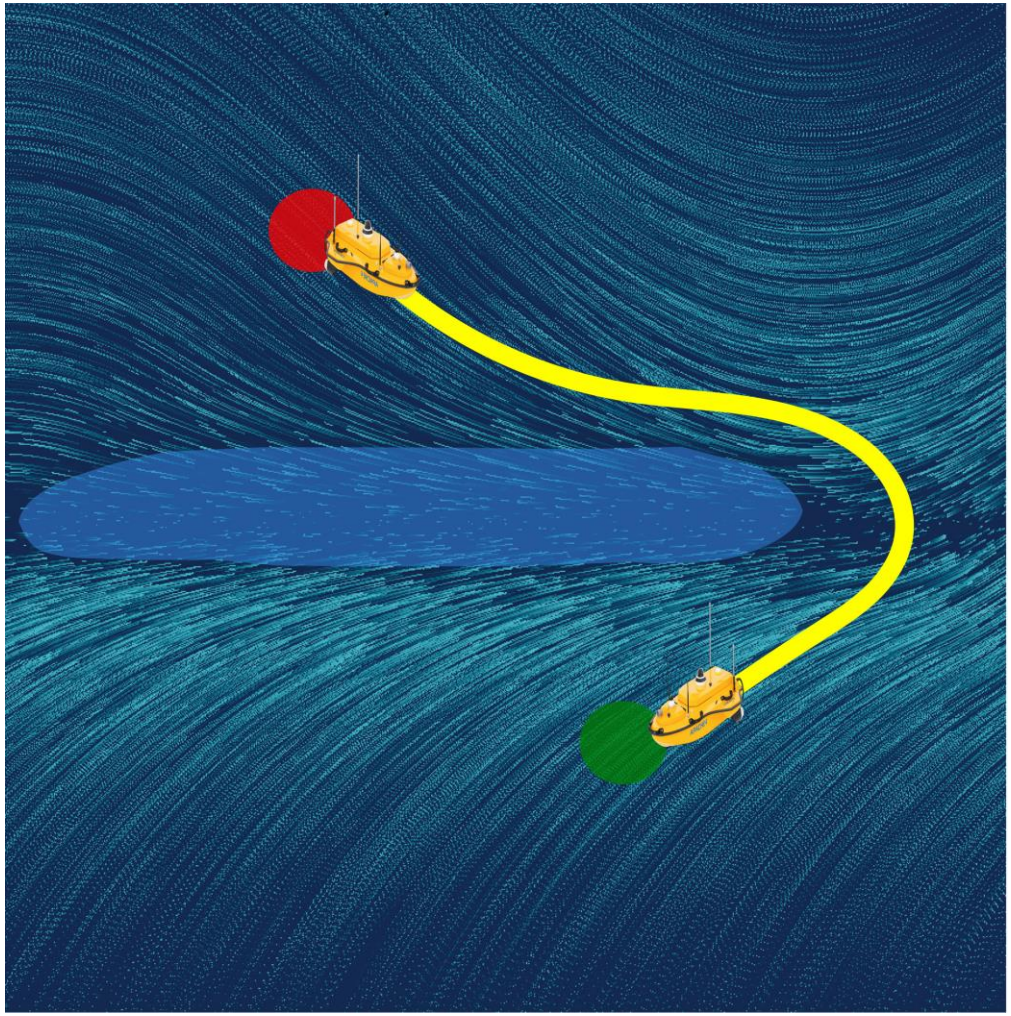
# Path planning on predictive vector fields

Informative path can be found by moving along a vector field

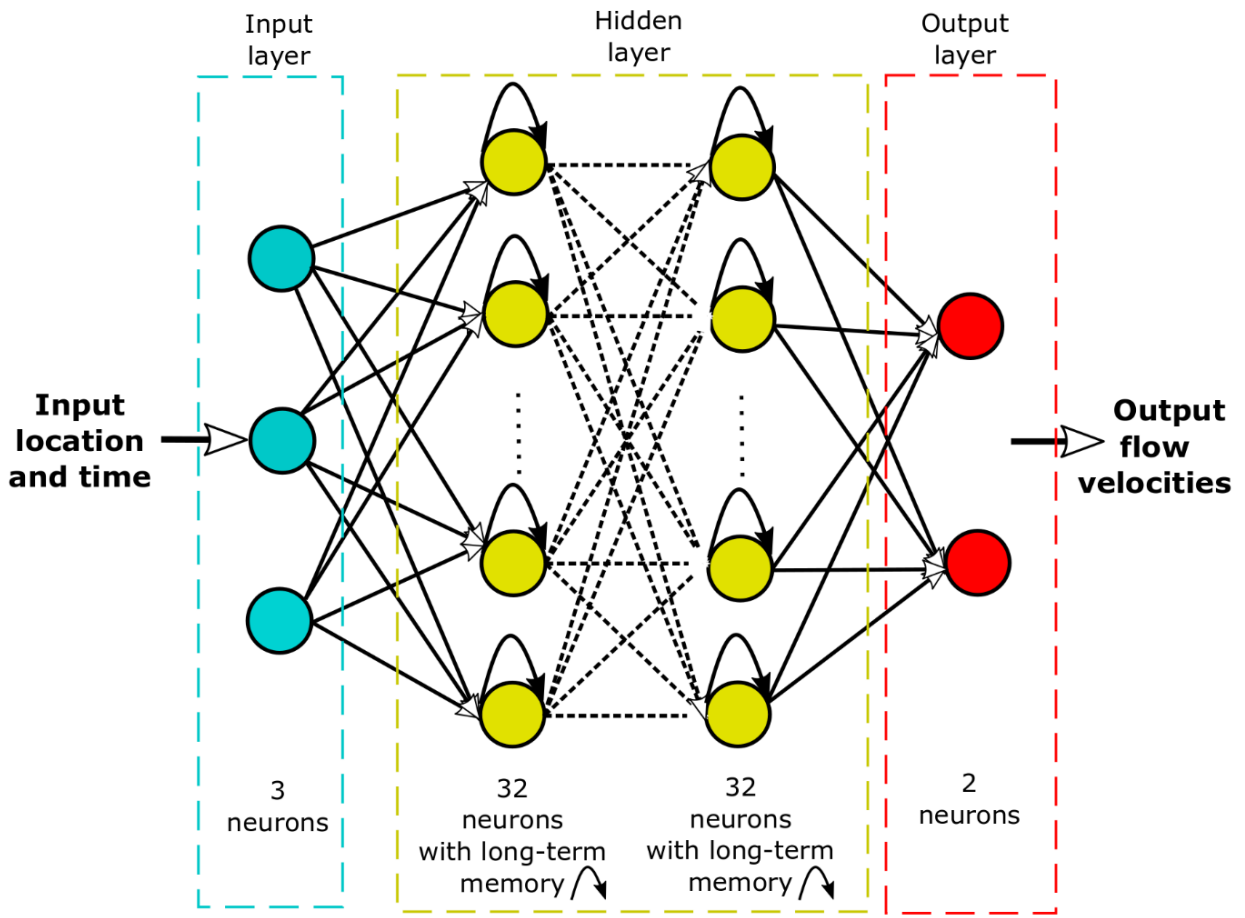
# DVF-RRT: Randomized Path Planning on Predictive Vector Fields

**Problem Statement:** Given a vector field that characterizes a marine environment, our goal is to solve the problem of path planning for Autonomous Surface Vehicles (ASVs)

Challenges: Uncertain nature of environments with obstacles and external forces or disturbances of vector fields



# LSTM Neural Network Model for Predicting Ocean Dynamics



# DVF-RRT Algorithm

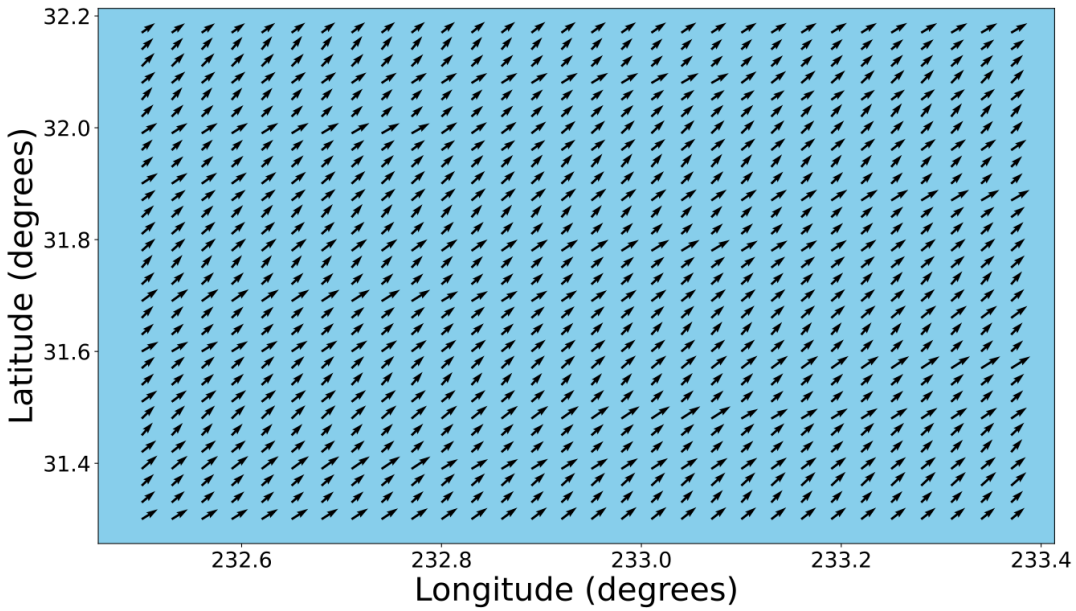
**Algorithm 1:** DVF-RRT $(\mathbf{x}_I, \mathbf{x}_G, F)$

**Input:** $\mathbf{x}_I, \mathbf{x}_G, F$ – Initial configuration, Goal Configuration, and Predicted Vector Field

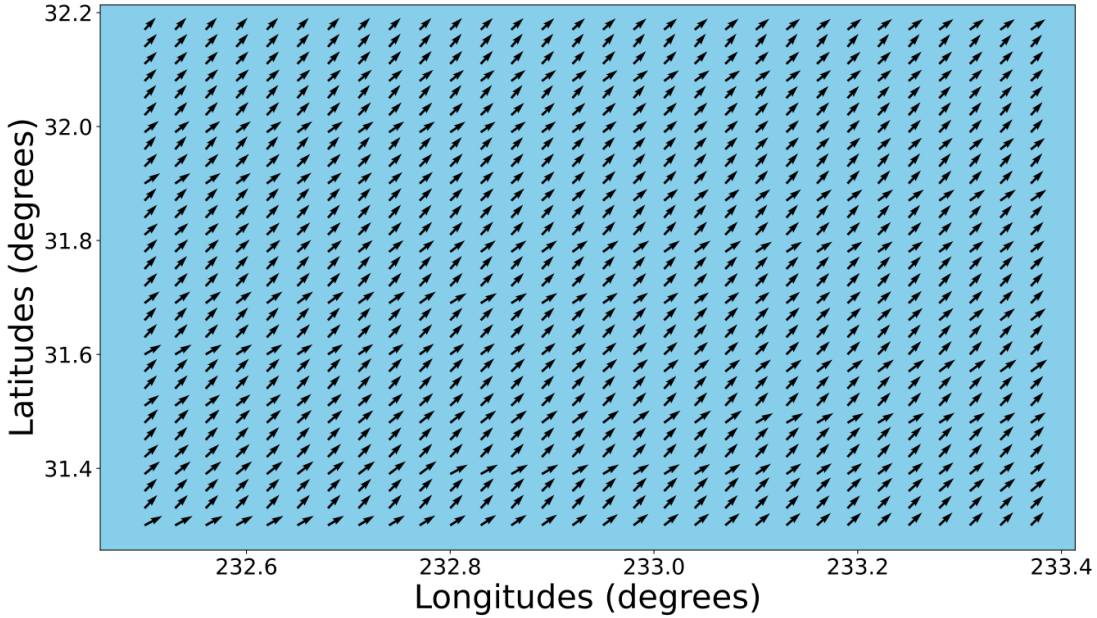
**Output:** $\mathcal{P}$ – Path

```
1  Tree ← ∅
2  Tree.AddVertex(x_I)
3  while i < I_max do
4      i ← i + 1
5      x_rand ← RandomSampling()
6      x_near ← FindNearestNodeOnTree(Tree, x_rand)
7      v̂_new ← GetNewDirection(x_near, x_rand, F, Tree)
8      x_new ← Extend(Tree, x_near, v̂_new)
9      if Connect(x_new, x_G) then
10         return P
11 return NULL
```

# Simulation Results: Learning Ocean Dynamics

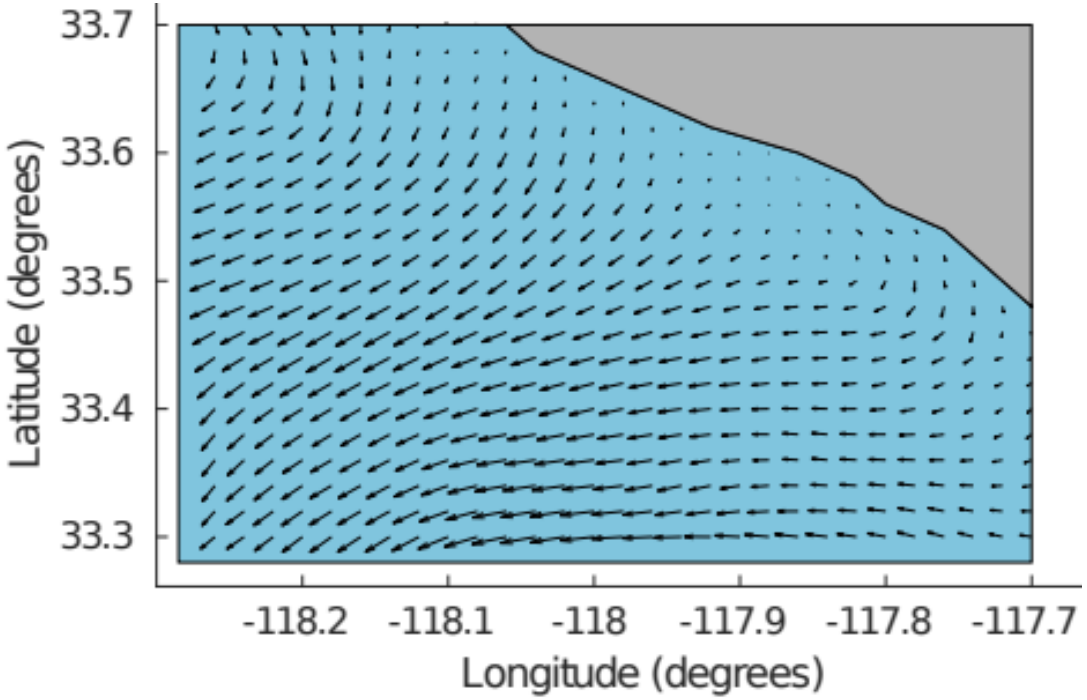



Ground Truth Vector Field

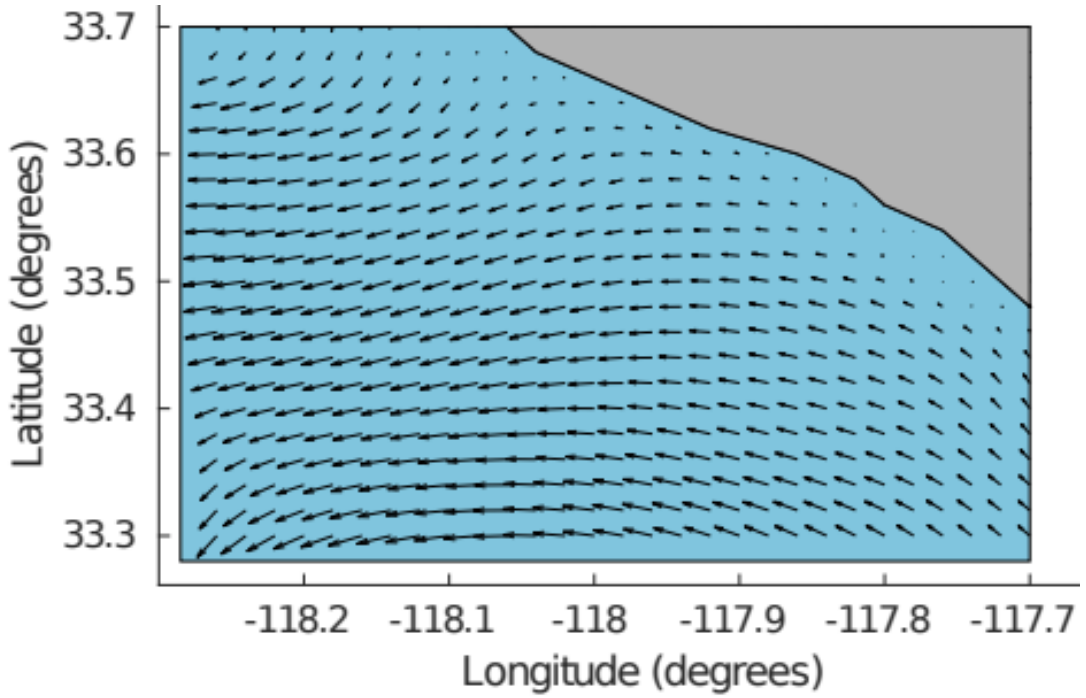



Predicted Vector Field

# Simulation Results: Learning Ocean Dynamics

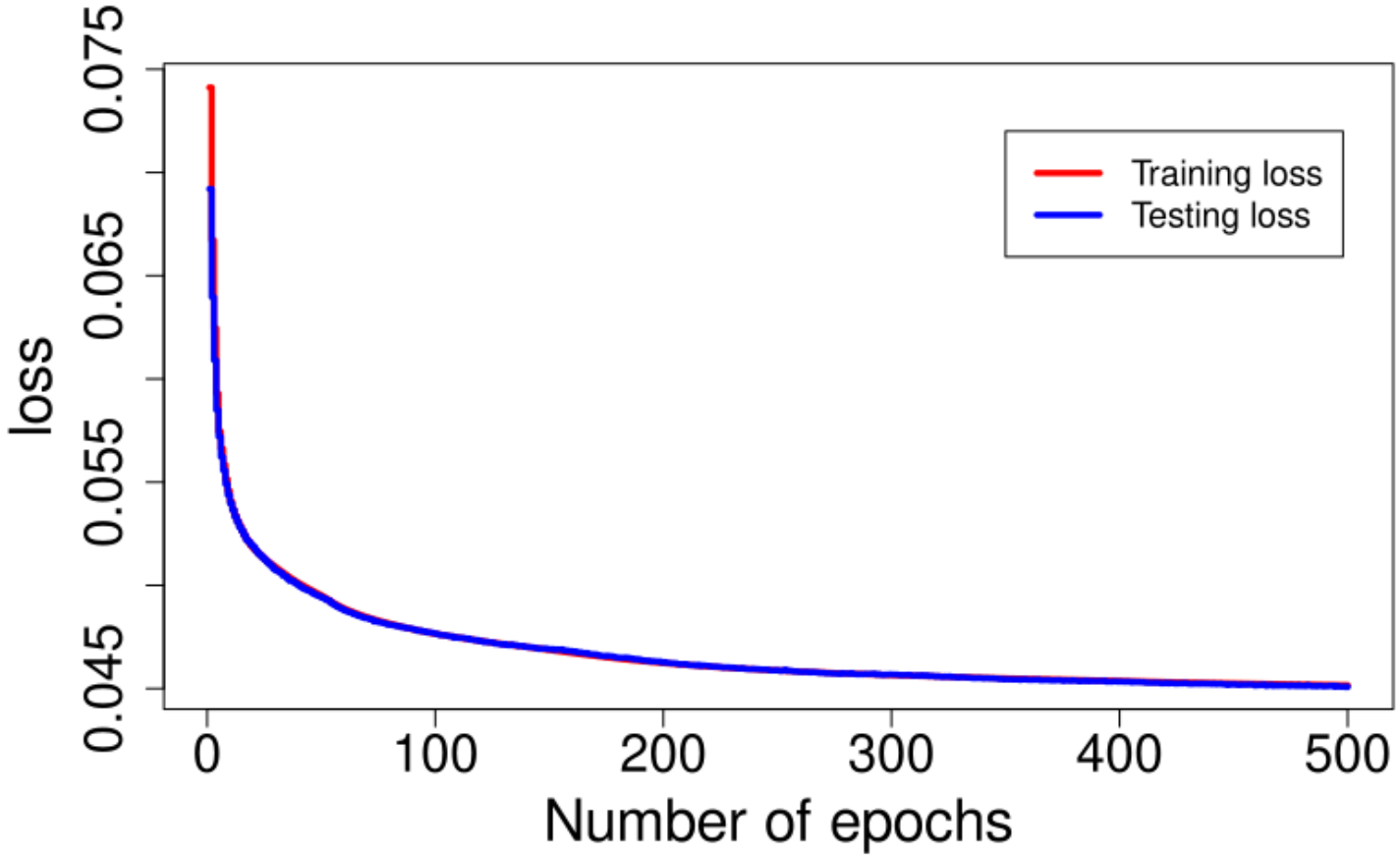



Ground Truth Ocean Dynamics

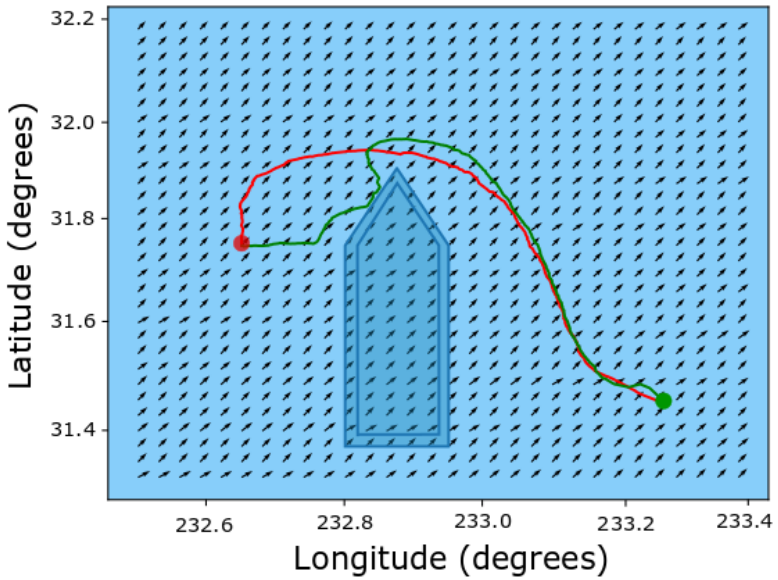



Predicted Ocean Dynamics

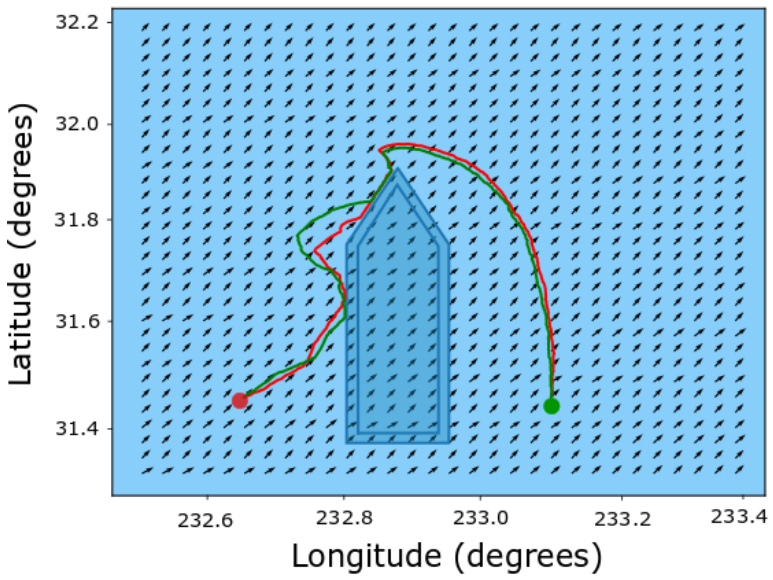
# Simulation Results: Loss Function



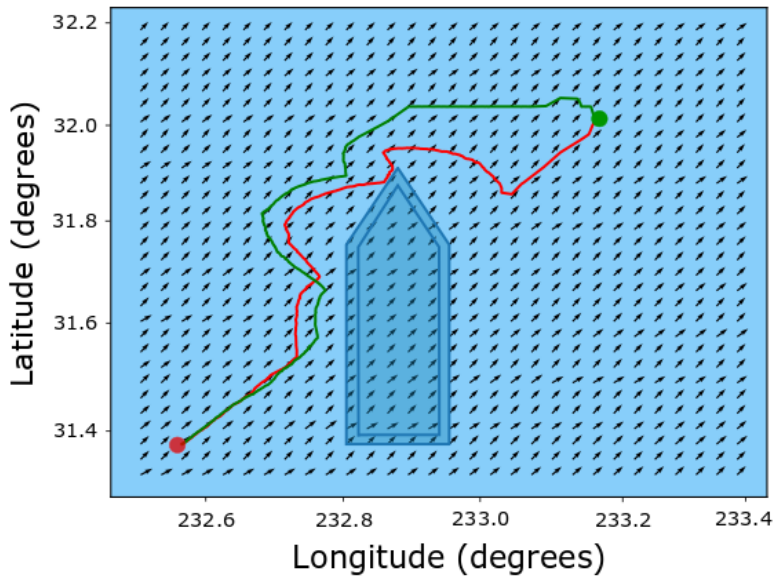
# Simulation Results: Navigation Paths




Downstream



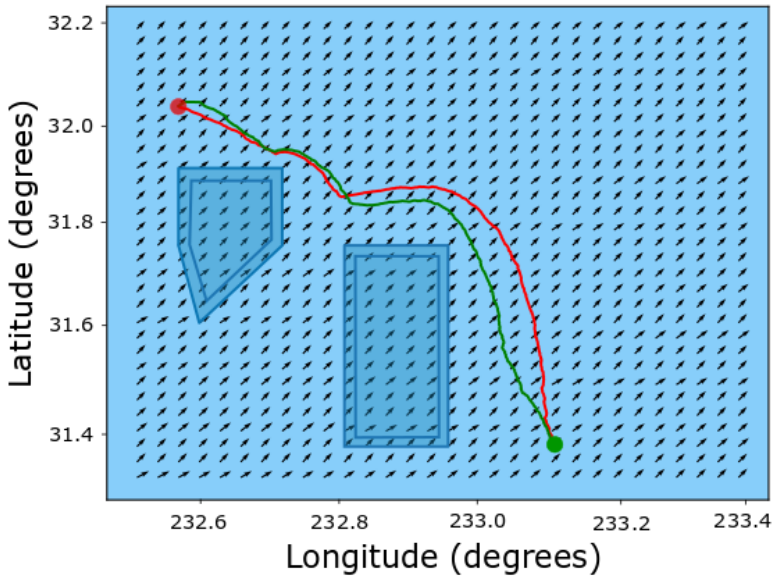

Linear



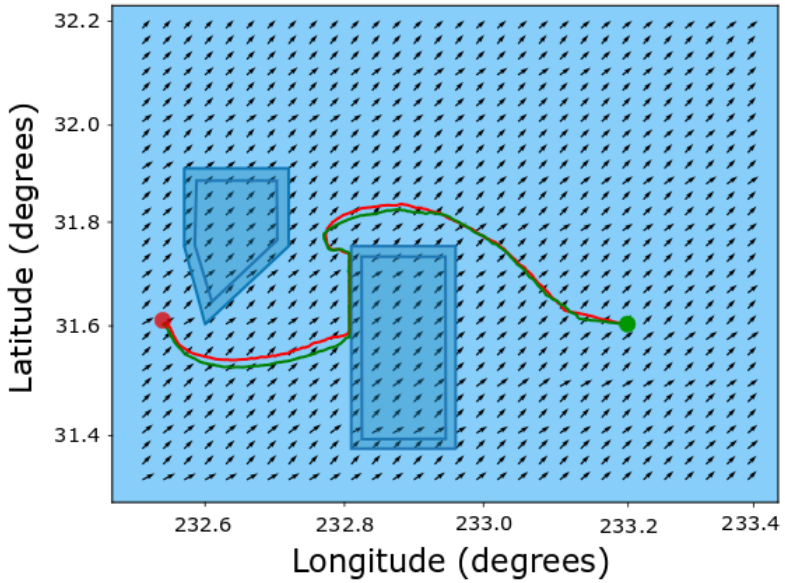

Upstream

Paths on Flow Field 1

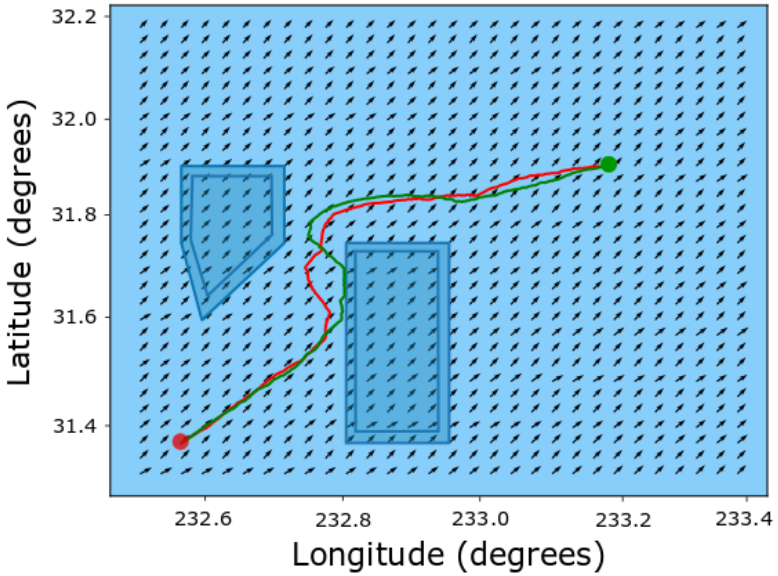
# Simulation Results: Navigation Paths (Contd.)




Downstream



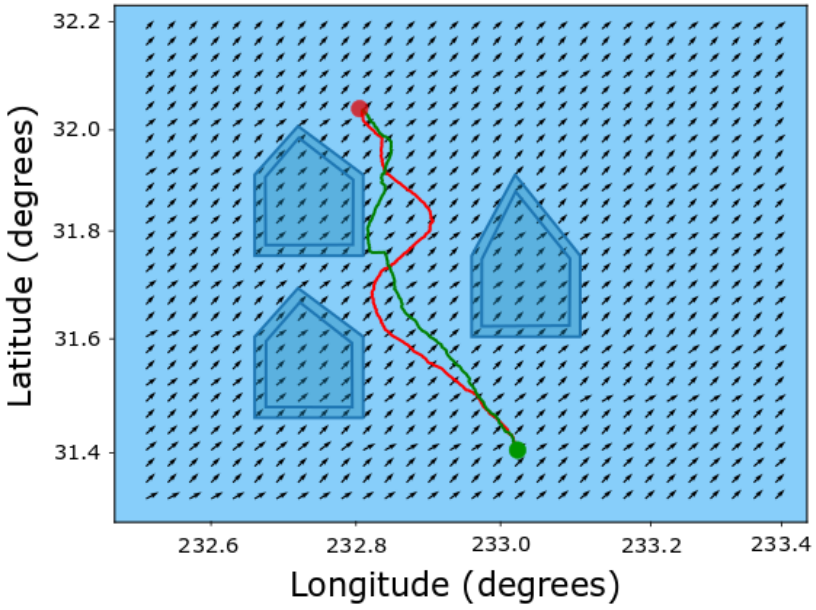

Linear



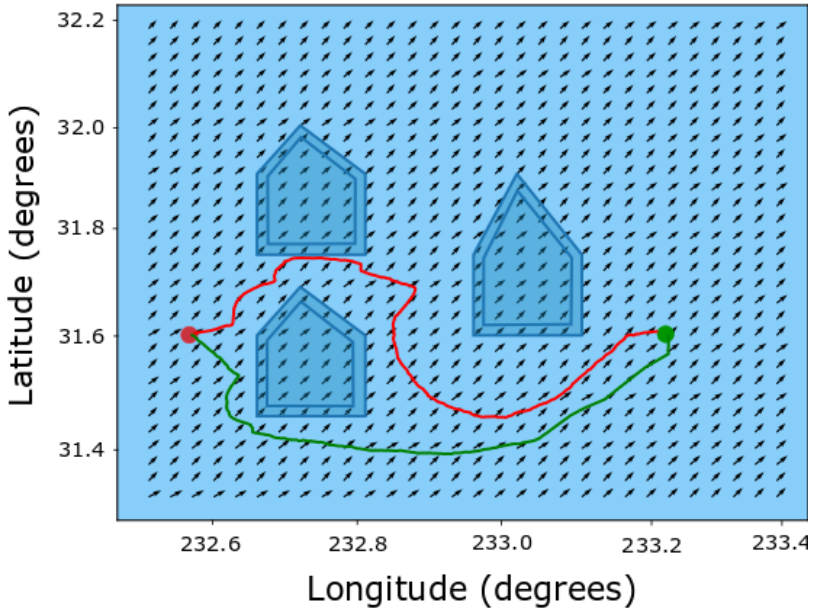

Upstream

Paths on Flow Field 2

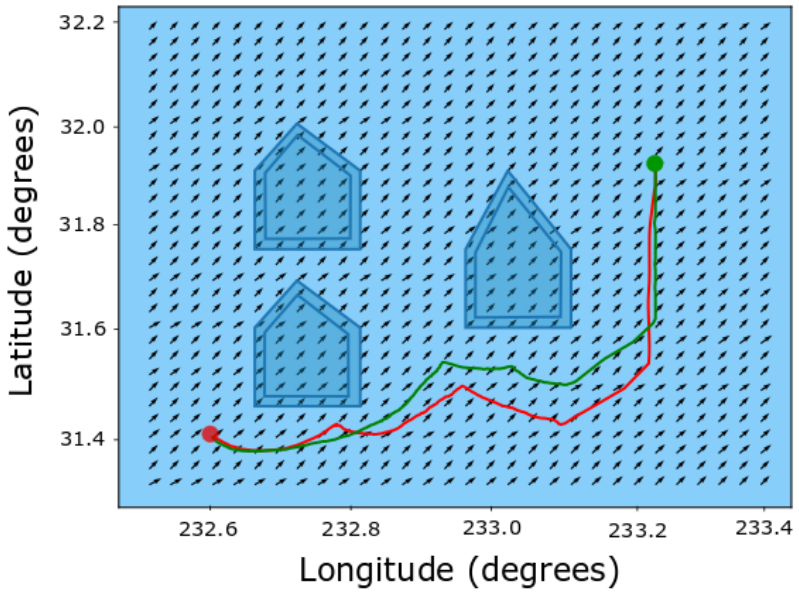
# Simulation Results: Navigation Paths (Contd.)




Downstream



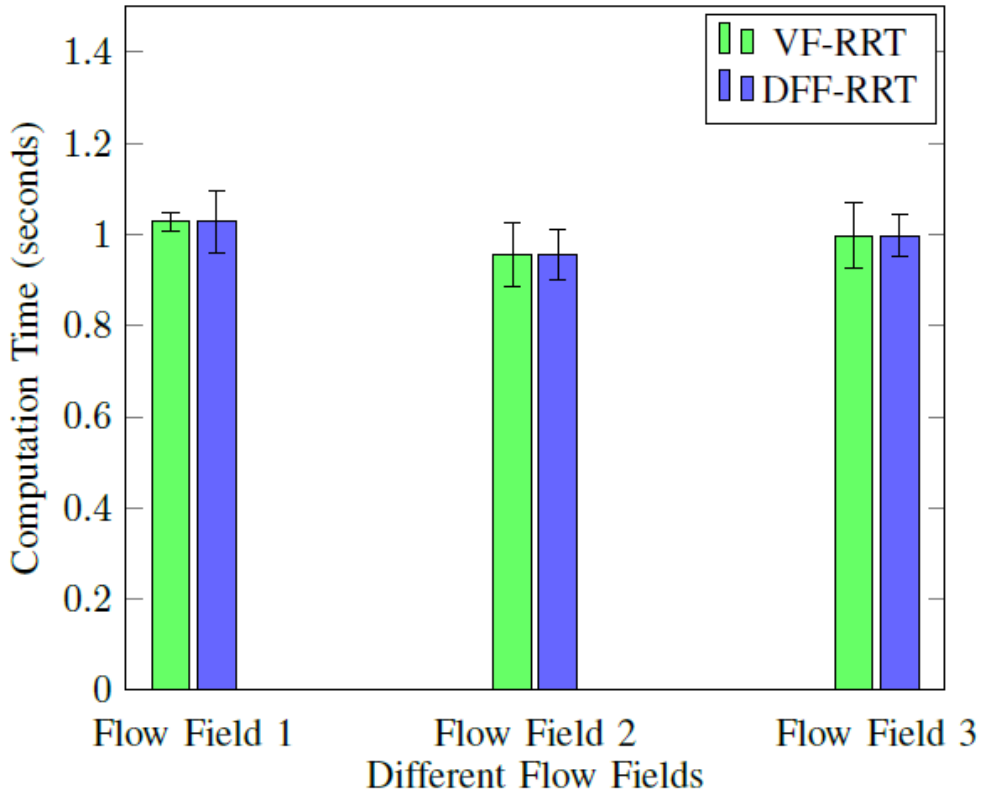

Linear




Upstream

Paths on Flow Field 3

# Simulation Results: Performance Comparison




Path computation time comparison between VF-RRT and DFF-RRT algorithms across several types of paths

# Simulation Results: Performance Comparison (Contd.)

| Vector Field | Upstream VF-RRT | Upstream DVF-RRT | Downstream VF-RRT | Downstream DVF-RRT | Linear VF-RRT | Linear DVF-RRT |
|---|---|---|---|---|---|---|
| Vector Field 1 | $31.35 \pm 5.89$ | $33.33 \pm 4.67$ | $\mathbf{47.79 \pm 4.02}$ | $\mathbf{47.44 \pm 4.41}$ | $22.57 \pm 17.37$ | $19.28 \pm 17.63$ |
| Vector Field 2 | $30.5 \pm 8.01$ | $36.26 \pm 2.13$ | $28.39 \pm 8.57$ | $25.18 \pm 9.38$ | $36.37 \pm 9.07$ | $36.36 \pm 9.02$ |
| Vector Field 3 | $\mathbf{29.94 \pm 5.64}$ | $\mathbf{30.71 \pm 3.78}$ | $35.23 \pm 10.55$ | $33.23 \pm 9.09$ | $\mathbf{31.94 \pm 2.24}$ | $\mathbf{31.19 \pm 4.43}$ |

Comparison of path lengths for proposed DVF-RRT and existing VF-RRT algorithms

| Vector Field | Upstream VF-RRT | Upstream DVF-RRT | Downstream VF-RRT | Downstream DVF-RRT | Linear VF-RRT | Linear DVF-RRT |
|---|---|---|---|---|---|---|
| Vector Field 1 | $3.21 \pm .45$ | $3.35 \pm .39$ | $\mathbf{2.48 \pm .45}$ | $\mathbf{2.53 \pm .47}$ | $2.27 \pm .21$ | $1.84 \pm .21$ |
| Vector Field 2 | $2.72 \pm .60$ | $3.12 \pm .19$ | $1.87 \pm 1.27$ | $1.35 \pm 1.61$ | $\mathbf{3.27 \pm .93}$ | $\mathbf{3.3 \pm .93}$ |
| Vector Field 3 | $\mathbf{2.97 \pm .64}$ | $\mathbf{3.02 \pm .61}$ | $4.02 \pm .56$ | $3.9 \pm .49$ | $2.52 \pm .11$ | $2.29 \pm .41$ |

Comparison of vehicle heading angle differences in generated paths for proposed DVF-RRT and existing VF-RRT algorithms

# Discussion

- We present a method that integrates learning an unknown vector field with planning a path on the predicted vector field for ASV navigation in marine environments
- High-level specification of marine environments -> predicted vector fields
- An LSTM network for learning an unknown vector field from real ocean current data in a continuous space instead of a discrete space. Our trained model predicts a vector field that is accurate when compared to the ground truth vector field.
- Low-level planner -> computes randomized paths that follow vector fields and avoid static obstacles
- The validation and benchmarking of our proposed method demonstrate its potential applicability

# Motion planning Under External Disturbances

# Robot Planning and Control Under External Disturbances

- Robot motion planning and control is challenging in dynamic environments with external disturbances (e.g., wind and water currents) as well as static and moving obstacles
- Existing Dynamic Window Approaches [1, 2] provide an elegant online collision avoidance strategy for mobile robots but do not reason about the effect of external disturbances in their object functions.
- Closely related work [3]: A feedback motion plan approach for a single unicycle robot by combining attractive and repulsive vector fields in environments with static circular obstacles. This approach does not consider external disturbances affecting system dynamics.

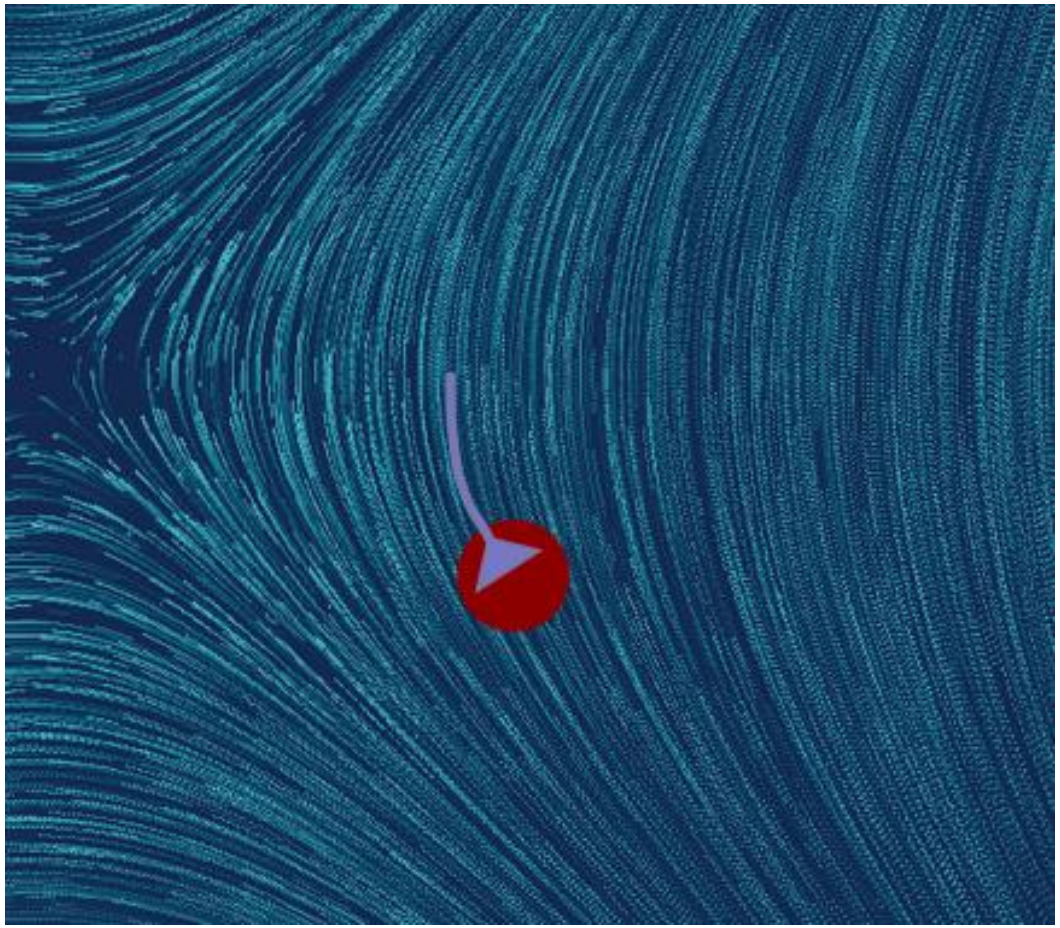
**Objective**

*Generate collision-free and resilient local trajectories for a mobile robot navigation in environments with external disturbances*

[1] M. Dobrevski, and D. Skočaj, "Adaptive dynamic window approach for local navigation," in IROS, pp. 6930-6936. IEEE, 2020.
[2] M. Missura, and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in IEEE ICRA, pp. 8620-26. 2019.
[3] D. Panagou, "Motion planning and collision avoidance using navigation vector fields," in IEEE ICRA, pp. 2513–2518, 2014.

# Our Contributions

1. Model external disturbances of an environment through a parametric vector field with unknown parameters
2. Propose a learning method to estimate these unknown parameters
3. Modify the robot motion model to include the effect of the vector field
4. Extend the objective function of the DWA to incorporate the environmental flow factor for efficiently navigating through a vector field while avoiding unknown static obstacles



# The Dynamic Window Approach to Motion Planning and Collision Avoidance Under Vector Fields

$$\begin{aligned}\mathbf{x}_t &= f(\mathbf{x}^*_{t-1}, \mathbf{u}^*_{t-1}, \mathbf{v}^*_{t-1}) + \mathbf{A}_t(\mathbf{x}_{t-1} - \mathbf{x}^*_{t-1}) \\ &\quad + \mathbf{B}_t(\mathbf{u}_{t-1} - \mathbf{u}^*_{t-1}) + \mathbf{D}_t(\mathbf{v}_{t-1} - \mathbf{v}^*_{t-1})\end{aligned}$$

Learn the unknown parameters from sample dataset over the workspace

$$G(\bar{\mathbf{u}}) = \sigma(\alpha \cdot heading(\bar{\mathbf{u}}), +\beta \cdot dist(\bar{\mathbf{u}}) + \gamma \cdot vel(\bar{\mathbf{u}}) + \zeta \cdot flow(\bar{\mathbf{u}})$$

**Robot Motion Model under Vector Fields**

**Learn Unknown Parameters of Vector Fields**

**DWA Under Vector Fields (DWAVF)**

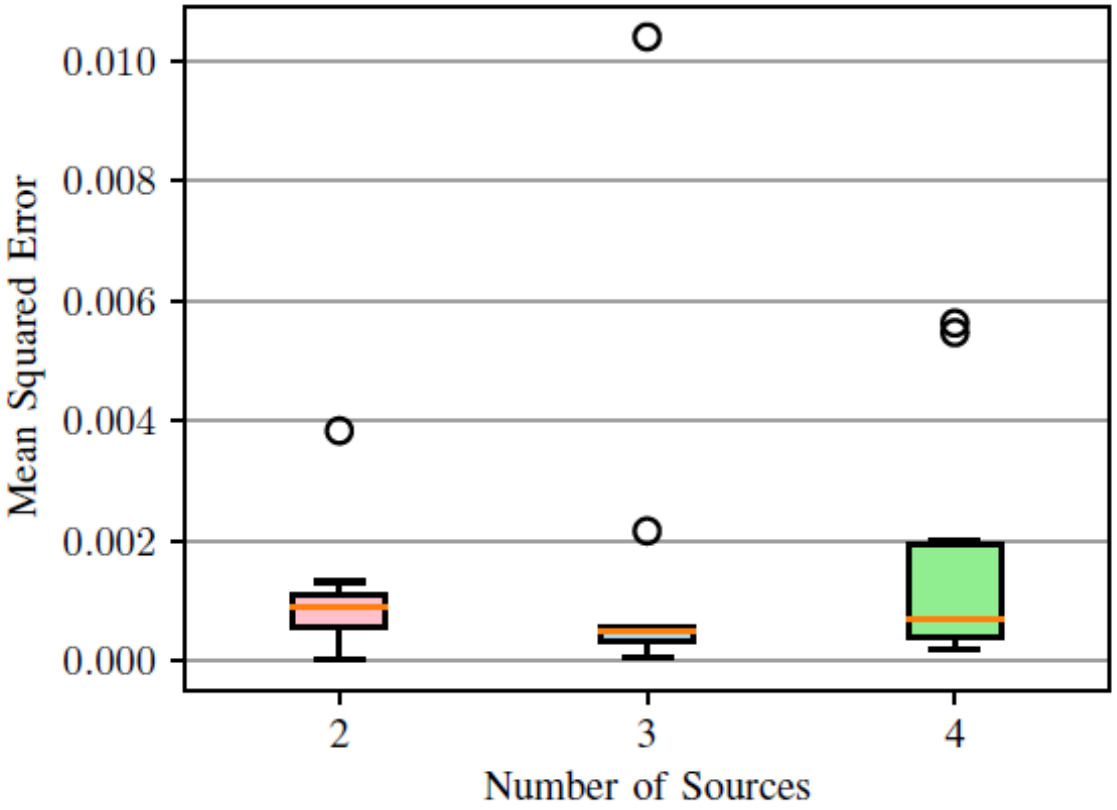
Consider the effect of vector fields which can lead to a non-linear motion model (approximately linearized)

$$\chi^2(\beta) = \sum_{j=1}^{|D|} \left(\mathbf{F}(\mathbf{r}_j) - \mathbf{F}_\beta(\mathbf{r}_j)\right)^2,$$

Estimate four factors, including our introduced environmental flow factor

- Target Heading factor
- Safety factor
- Velocity factor
- Flow factor

# Learn the Unknown Parameters of Vector Fields



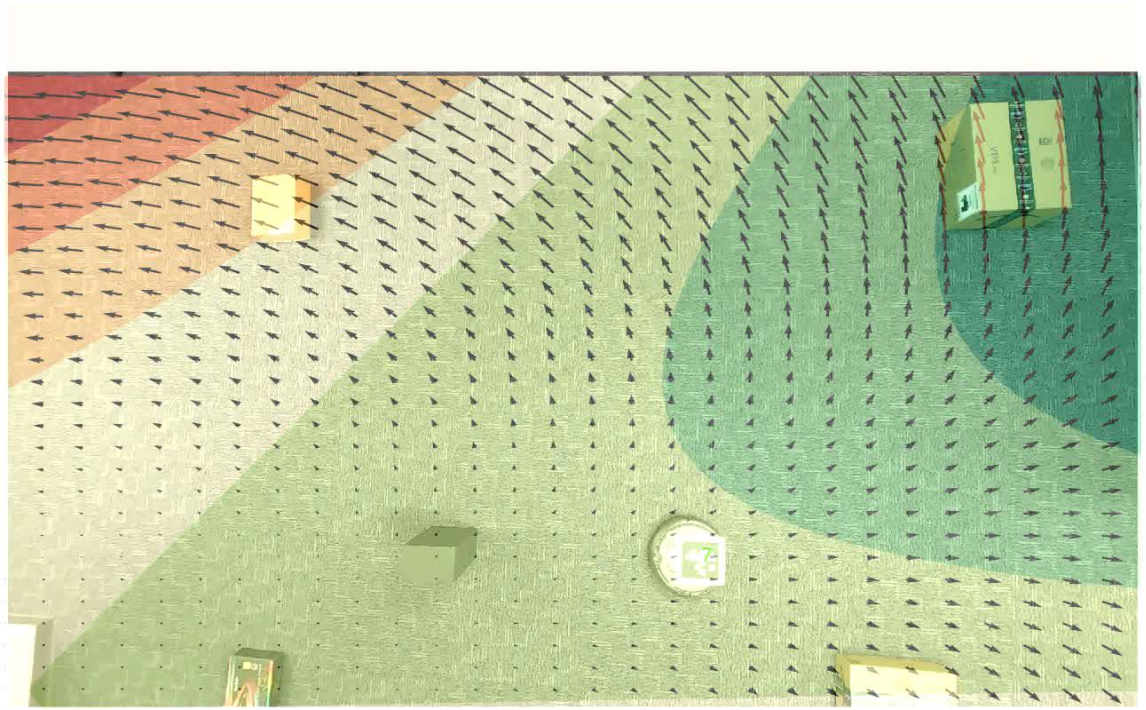

The Gauss-Newton algorithm can accurately estimate our vector field parameters with a few samples.

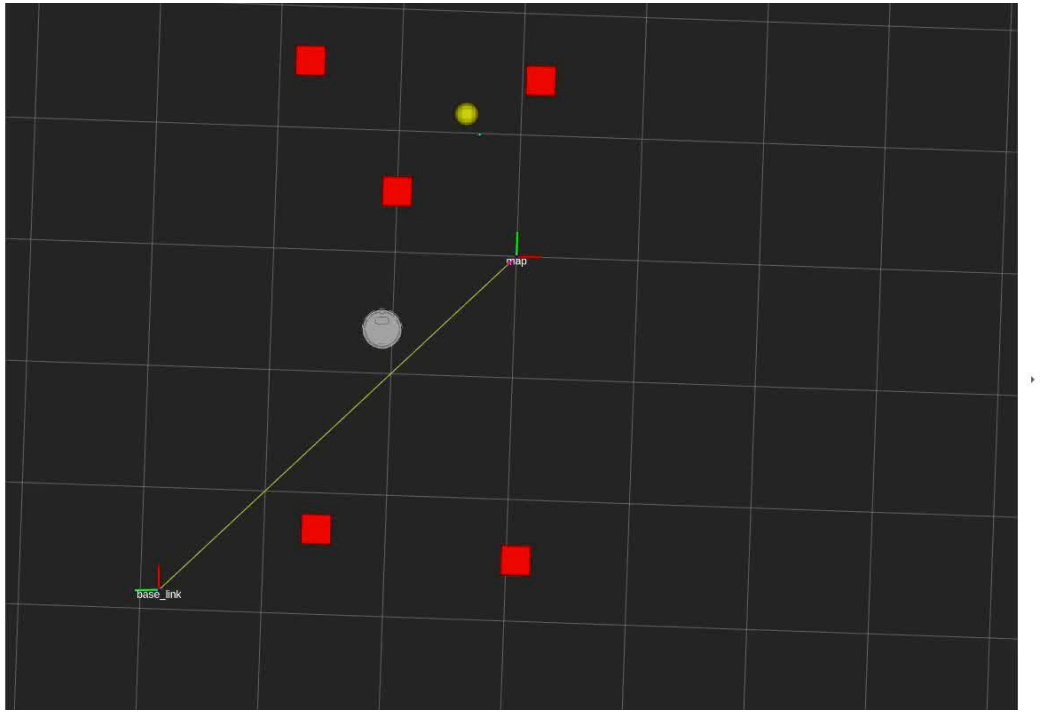
# Trajectory Planning for DWA and DWAVF Planners in an Aquatic Environment

DWA Planner

DWAVF Planner

# Physical Experiment in a Windy Environment






Physical experiment with a ground robot under simulated wind disturbances.

# Performance Analysis

| Exp | Trajectory Length (m) | | Total Workdone (N.m) | |
|---|---|---|---|---|
| | DWA | VFDWA | DWA | VFDWA |
| 1 | 679.471 | 530.153 | 336.137 | 303.899 |
| 2 | 463.362 | 530.153 | 739.361 | 303.899 |
| 3 | 679.471 | 457.063 | 336.137 | 611.941 |

Trajectory analysis between the DWA and DWAVF planners.

# Multi-Robot Information Gathering

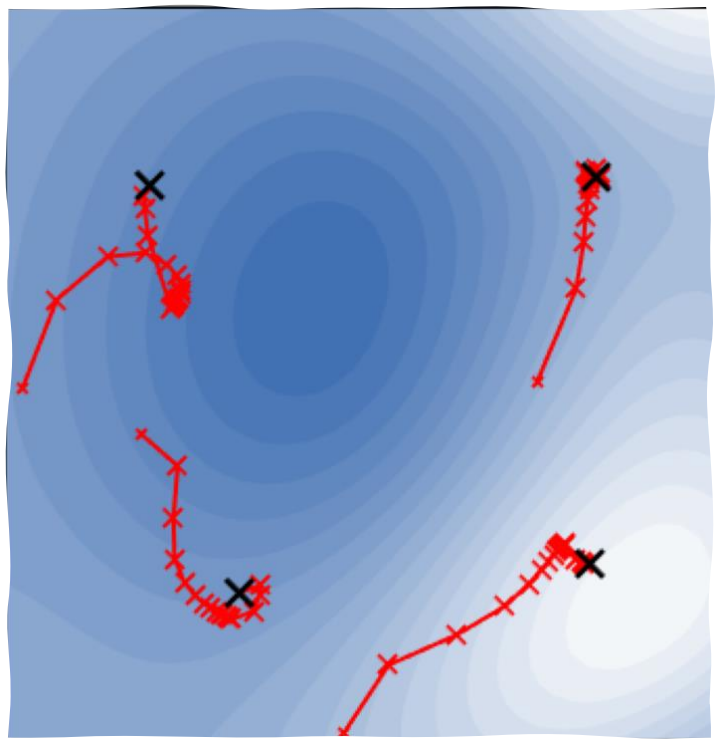
# Multi-Robot Information Gathering

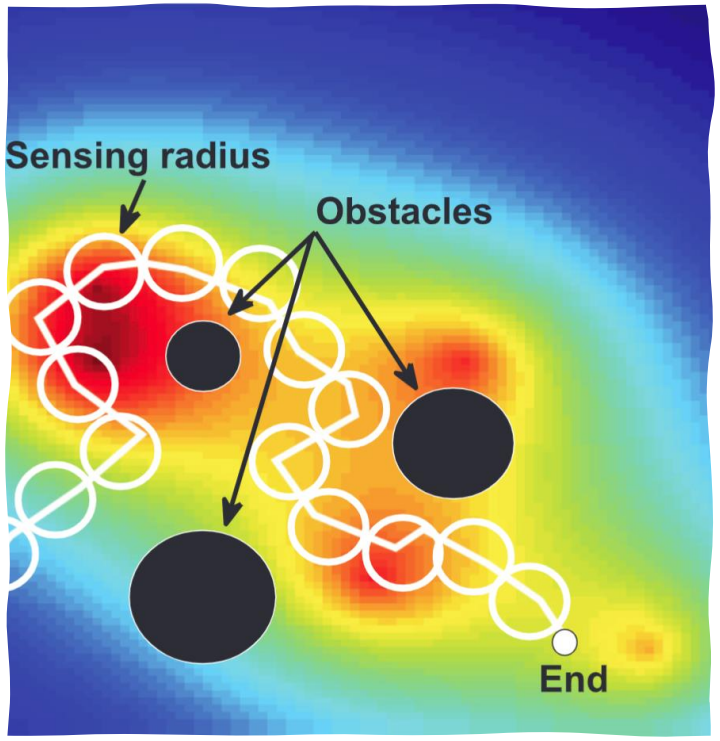
- Multi-robot information gathering plays pivotal roles in many application domains:
  - nuclear radiation,
  - greenhouse gas emission,
  - soil parameters (e.g., nutrient levels and pH) in precision agriculture,
  - chemical or oil spill in coastal areas
- These phenomena are generally referred to as spatial fields.

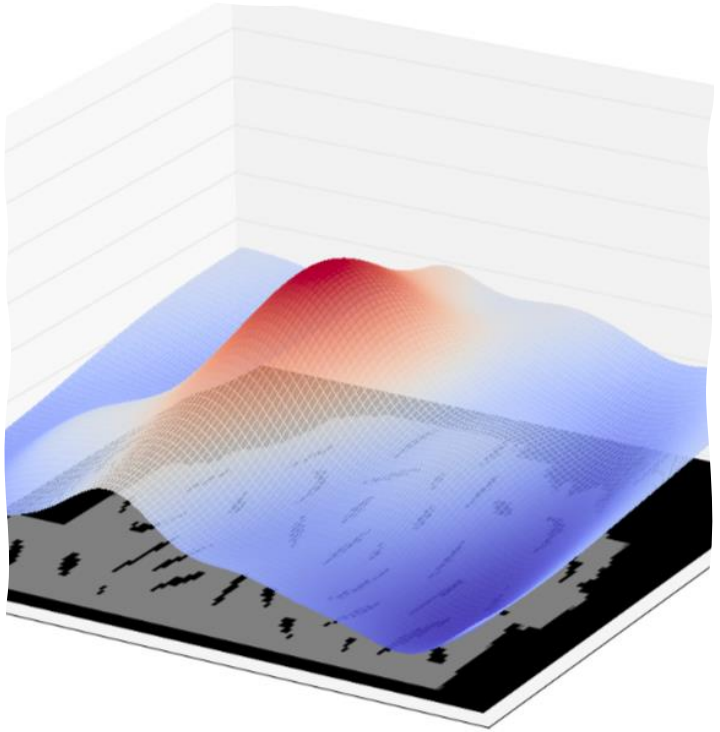
**Objective**

Estimate an unknown spatial field specified as a density function while satisfying robot motion and resource constraints

# Existing Methods Not Scalable!



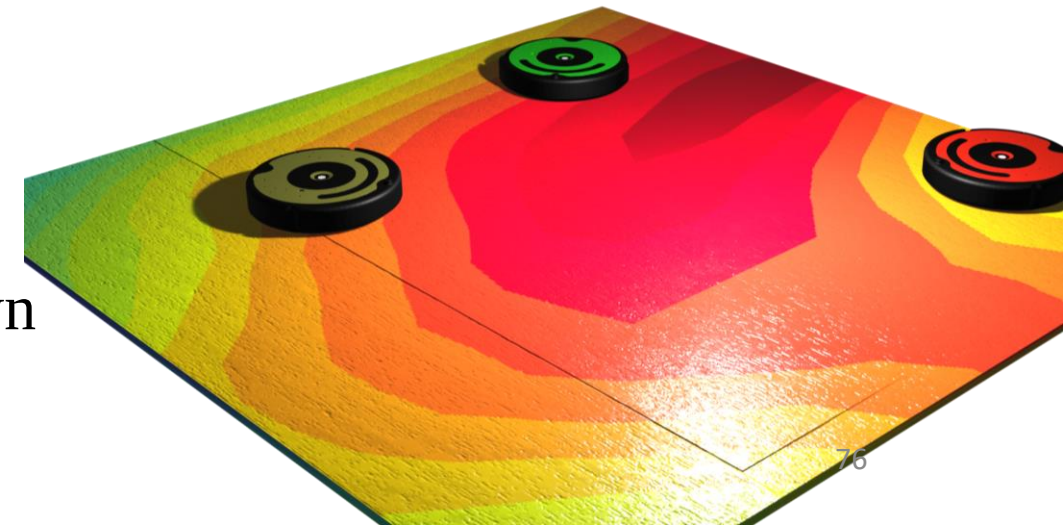


[1] A. Benevento, M. Santos, G. Notarstefano, K. Paynabar, M. Bloch, and M. Egerstedt, "Multi-robot coordination for estimation and coverage of unknown spatial fields," in *IEEE International Conference on Robotics and Automation*, pp. 7740–7746, 2020.
[2] G. A. Hollinger and G. S. Sukhatme, "Sampling-based robotic information gathering algorithms," *International Journal of Robotics Research*, vol. 33, no. 9, pp. 1271–1287, 2014.
[3] M. Santos, U. Madhushani, A. Benevento, and N. E. Leonard, "Multi-robot learning and coverage of unknown spatial fields," in *International Symposium on Multi-Robot and Multi-Agent Systems*, pp. 137–145, 2021.

# Our Contributions

1. Propose an Asynchronous Information Gathering with Bayesian Optimization (AsyncIGBO) algorithm to choose the next informative samples for robots subject to their observed knowledge
2. Estimate the unknown density function from gathered samples with independently exploring multiple robots
3. Generate a distributed control law in continuous domains while considering realistic robotic resource and motion constraints
4. Theoretically analyze the asymptotic no-regret properties of our algorithm with respect to a known spatial field



# Proposed Framework

$$x_i^{\text{goal}} = \arg\max_{x_i \in \mathcal{X}_{\text{neigh}}^i} \left\{ \alpha\left(x_i|D\right) \prod_{j=1}^{n} \psi(x_i|D) \right\}$$

Generate collision-free trajectories toward informative samples

$$(\rho, \theta) = \arg\max_{\rho,\theta} \prod_{i=1}^{n} \prod_{l=1}^{b} \mathbb{P}(y_i, y_l | x_i, x_l, D, D_b, D_i; \rho, \theta)$$

**Informative Sampling**

**Local, reactive navigation**

**Behavior Tree-based Mechanism**

A penalization-based asynchronous sampling strategy to guide a team of cooperative robots toward the most informative locations

$$VO_{i|j} = \{\dot{\mathbf{x}} \mid \exists t > 0 : t(\dot{\mathbf{x}}_i - \dot{\mathbf{x}}_j) \in \mathcal{D}(\mathbf{x}_j - \mathbf{x}_i, \ell_i + \ell_j)\}$$

Capture the teaming and coordination of decentralized robots

# Theoretical Analysis

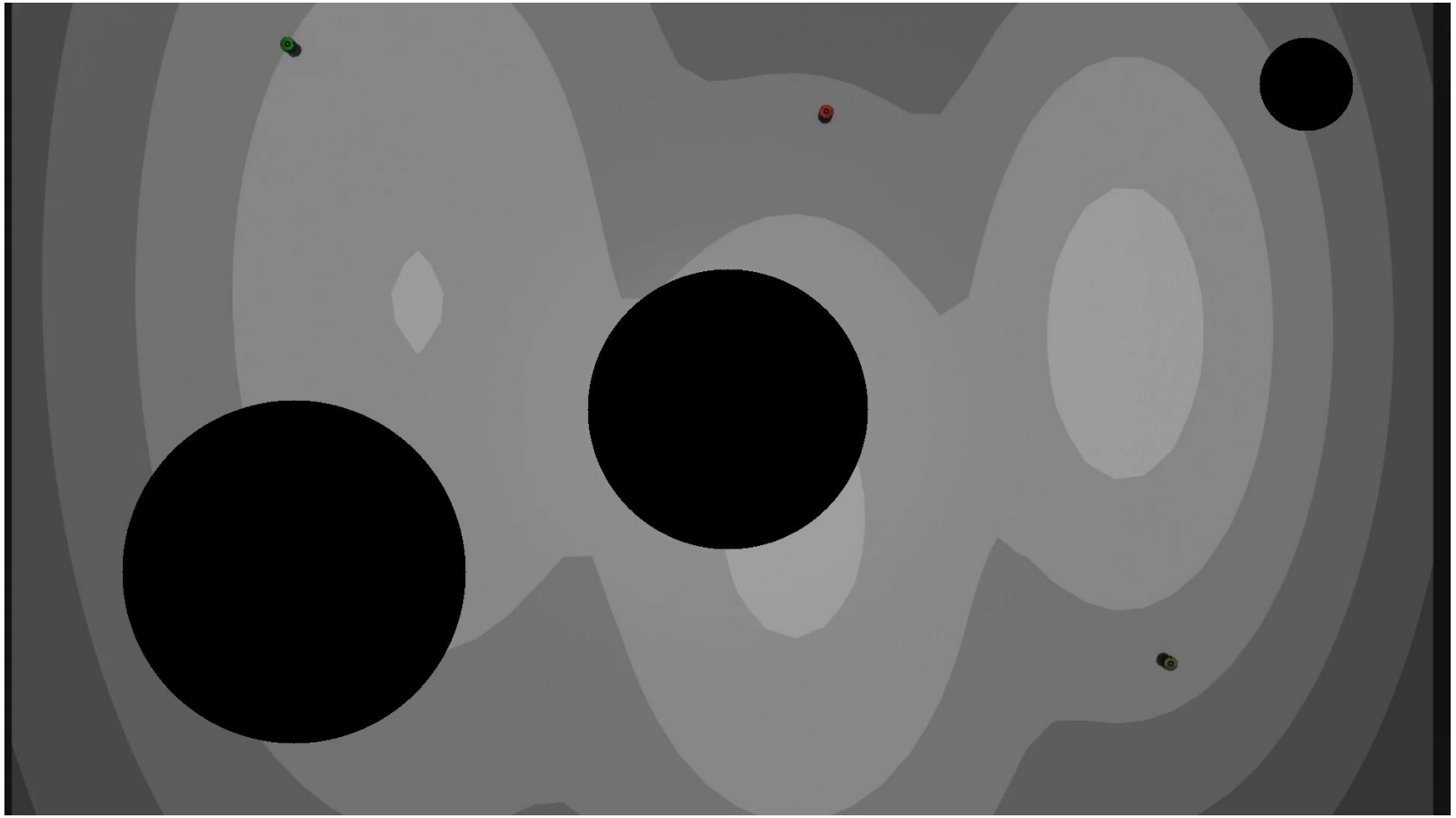
1. Probabilistic error bound on the penalization region

$$\mathbb{P}\left[r_i \leq \mathbb{E}(r_i) + \epsilon\right] \leq \exp\left(-\frac{2\epsilon^2 L^2}{\sigma^2}\right)$$

2. Asymptotic no-regret with respect to a known field

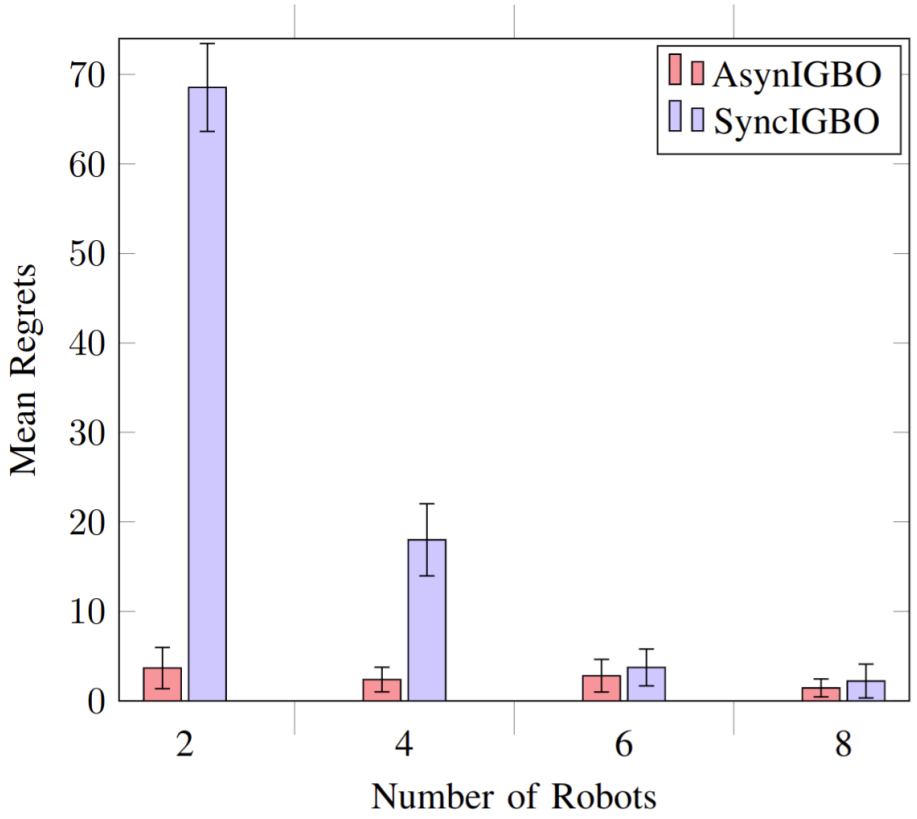
$$\int_{t=1}^{T} R(t)dt \leq \int_{t=1}^{T} c_0 \max \frac{\sqrt{\beta(t)}\sigma(t-1)(x_i(t))}{\exp\left(-\frac{2\epsilon^2 L^2}{\sigma(t-1)^2}\right)}dt.$$

# Simulation

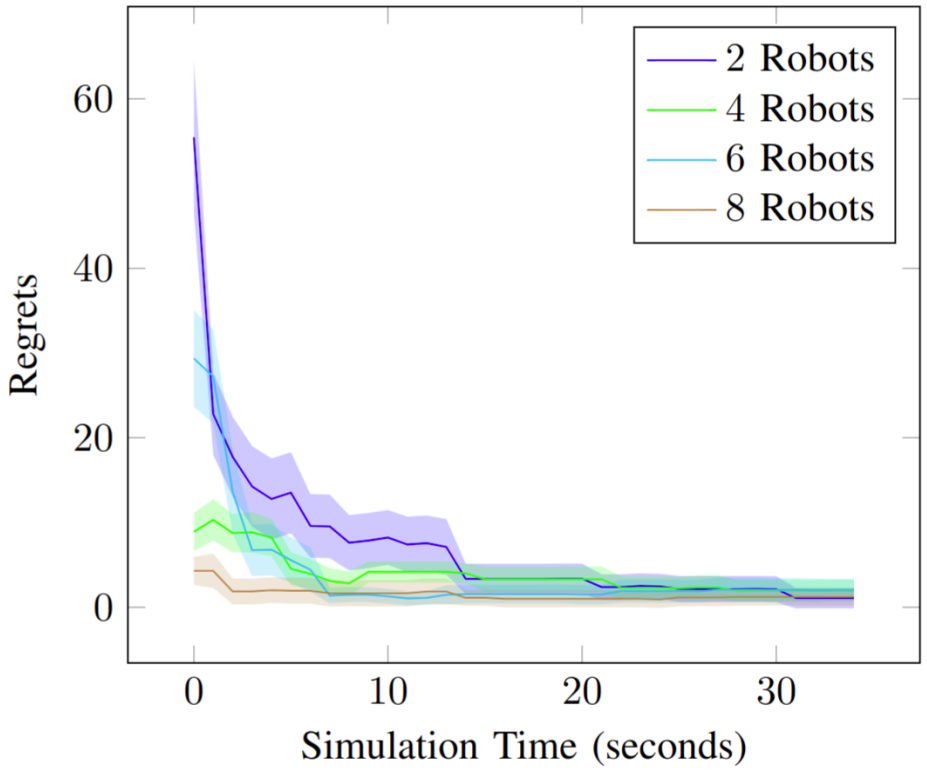


# Physical Experiment

# Performance Analysis




Our asynchronous algorithm outperforms the synchronous algorithm




Our algorithm is scalable in the size of problem

# Dealing with Non-stationary Fields

# Related Work

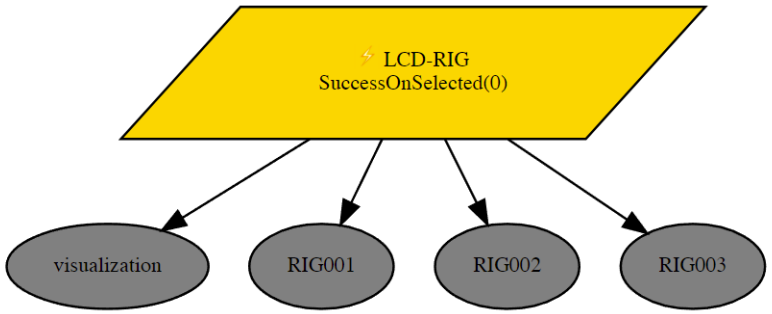
- The state-of-the-art Robotic information gathering algorithm [1] improves the uncertainty quantification of a probabilistic model using a nonstationary kernel but its application is limited to a single robot scenario and fails to utilize a team of cooperative robot.
- Centralized methods [2, 3] decompose environments for multiple robots but do not consider any coordination mechanism for decentralized robots to select the next informative locations.

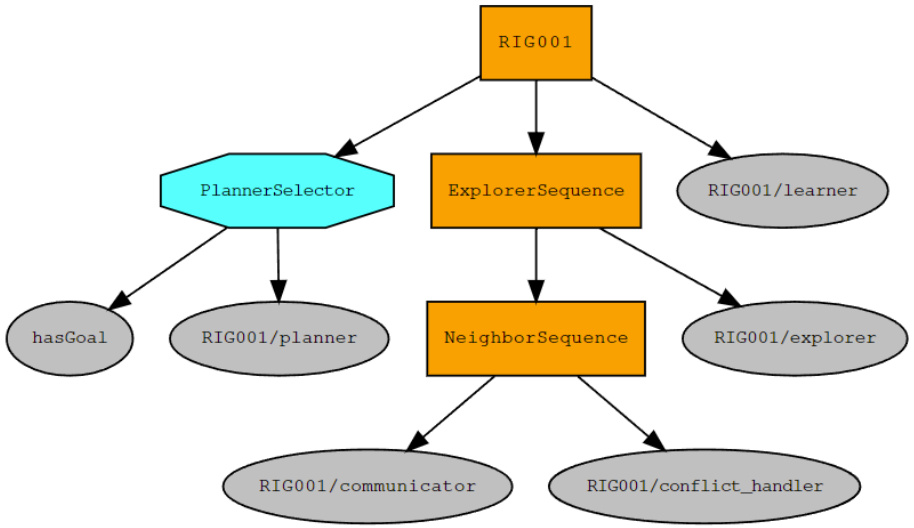
[1] W. Chen, R. Khardon, and L. Liu, "AK: Attentive kernel for information gathering," in RSS, 2022.
[2] A. Benevento et al., "Multi-robot coordination for estimation and coverage of unknown spatial fields," in ICRA, pp. 7740–7746, 2020.
[3] M. Santos et al., "Multi-robot learning and coverage of unknown spatial fields," in MRS, pp. 137–145, 2021.

# Objective

*LCD-RIG: Limited Communication Decentralized Robotic Information Gathering Systems*

# Proposed Method



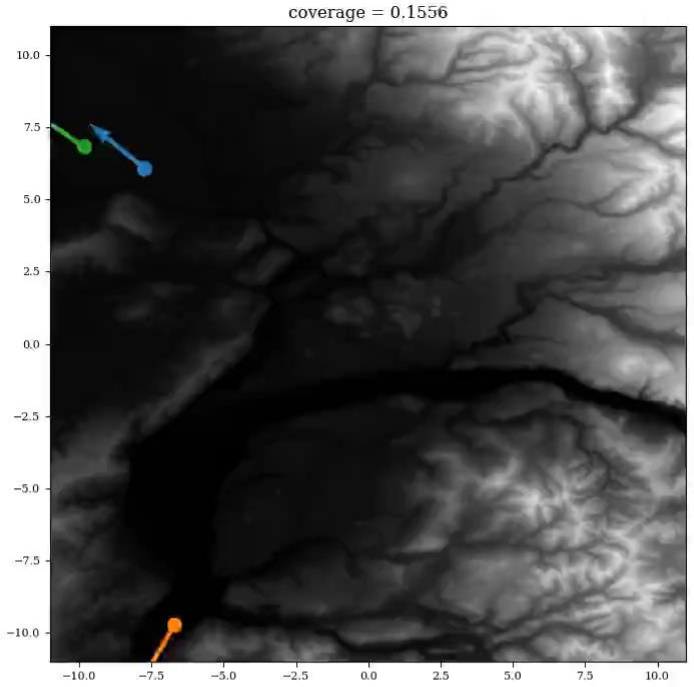

**Algorithm 1** LCD-RIG Algorithm for Robot $i$

Randomly initiate $\mathcal{A}_i, \ldots, \mathcal{A}_n$ robots with pilot data $D_{i,0}$
**for** $t = 1, 2, \ldots, T$ **do**
  **/* Sensing and Marking Visited Locations */**
  Sample independent training point $(x_{i,t}, y_{i,t})$
  $\xi_{i,t} = \{\xi_{i,t-1}, (x_{i,t}, y_{i,t})\}$,
  Update local quadtree $Q_i(t)$ with $\forall x_{i,t} \in \xi_{i,t}$
  **/* Communication and Conflict Handling */**
  **if** Distance$(\mathcal{A}_i, \mathcal{A}_j) \leq \delta$ **then**
    Receive $\xi_{j,t}$ from neighbors $j$
    Recompute intermediate goal states
  **end if**
  **/* Distributed Data Compression and Update */**
  Aggregate data points $S_K \leftarrow \xi_{i,t} \cup \xi_{j,t}$
  **if** DataSize $|S_K| > \lambda$ **then**
    **for** $k = K$ to $0$ **do**
      **for** $x_\ell \in S_K$ **do**
        $H_{s_\ell} \leftarrow H(s|S_k) = \frac{1}{2} \ln \left(2\pi e \sigma^2_{x_\ell|S_k|}\right)$
      **end for**
      $(x, y) \leftarrow \arg\min_{(x_\ell, y_\ell)} H_{s_\ell}$
      $S_k \leftarrow S_k \setminus \{(x, y)\}$
    **end for**
  **end if**
  Update training set $D_{i,t} \leftarrow D_{i,t-1} \cup S_K$
  Update local Gaussian process predictions:

$$\rho_{i,t} = P(\phi(x^*)|D_{i,t}, x^*) = \mathcal{N}(\mu_{i,t}, \Sigma_{i,t})$$

**end for**

# LCD-RIG Demo




```
/communicator]: From (RIG002/communicator samples = 2)
/conflict_handler] : [modifying goal states]  [array([ 3.04, -5.21])]
/communicator]: -> (1) RIG002
/communicator]: From (RIG001/communicator samples = 1)
/conflict_handler] : [modifying goal states]  [array([2.66, 0.35])]
/communicator]: -> (1) RIG001
/communicator]: From (RIG002/communicator samples = 2)
/conflict_handler] : [modifying goal states]  [array([-0.27, -3.19])]
/communicator]: -> (1) RIG002
/communicator]: From (RIG001/communicator samples = 1)
/conflict_handler] : [modifying goal states]  [array([4.24, 0.64])]
e = 0.1562
```

# Decision-making under Uncertainty

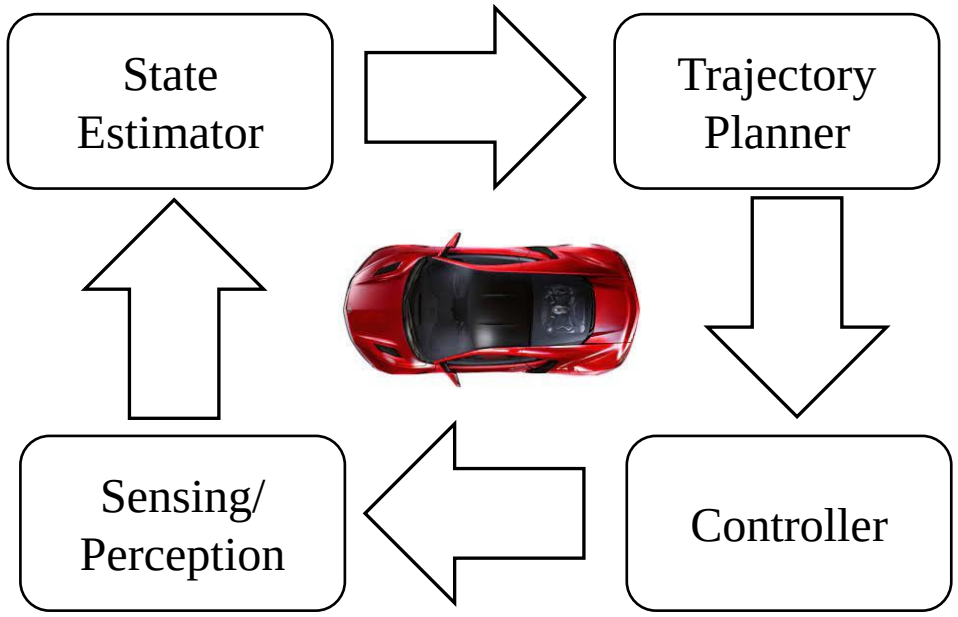
- Markov Decision Process (MDP)
- Partially Observable Markov Decision Process  (POMDP)
- Decision-making under Qualitative objective
- Decision-making under Qualitative and Quantitative objectives
- Decision making under cost, energy constraints
- Integrated task-and-motion planning under uncertainty

# Motion Planning for Robots

**Planning**: Given a robot with finite degrees of freedom ➡ find a trajectory through its configuration space

Commonly, robot motion planners have four building blocks

- ❖ Sensing/ Perception
- ❖ Localization / State Estimator
- ❖ Trajectory Planner
- ❖ Trajectory Tracker/ Controller

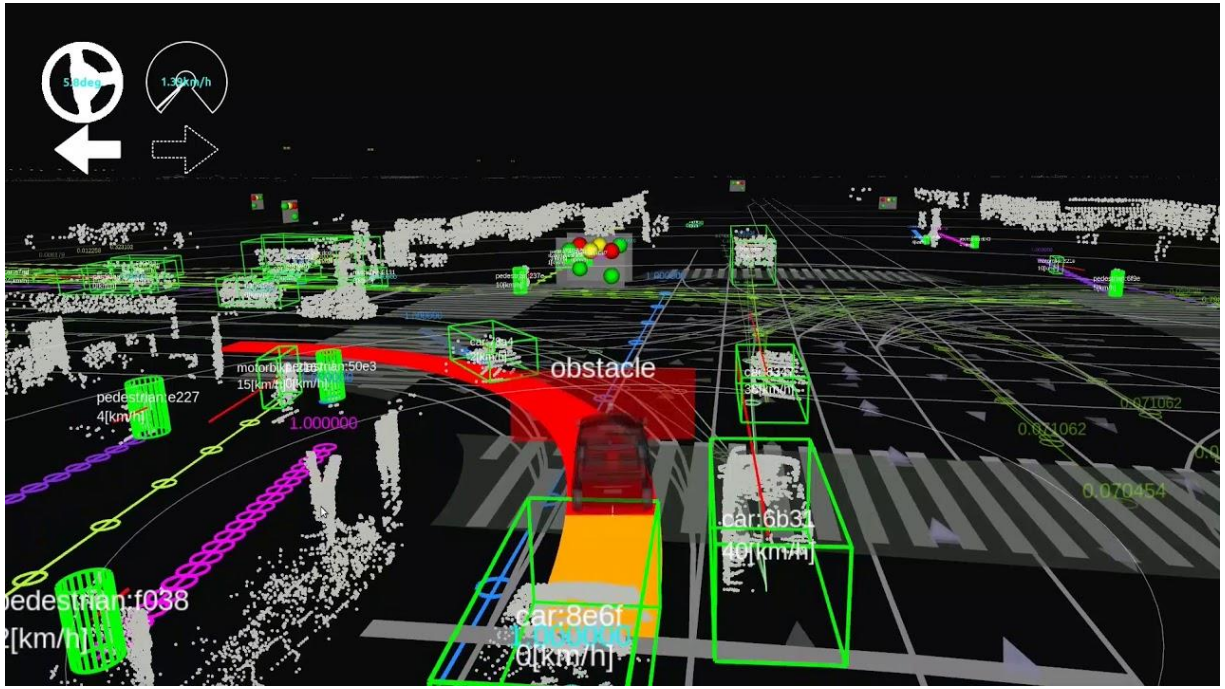



**Challenge**: *Uncertainty is an unavoidable part of each building block!*

# How to Deal With Uncertainties?

Three main sources of uncertainties

- **Motion uncertainty:** from the noise that affects systems dynamics.
- **Sensing uncertainty:** noisy sensory measurements, which also refer to imperfect state information, and
- **Environment uncertainty:** uncertain static obstacle locations, pedestrians, dynamic obstacles, or uncertain locations of features in the environment.

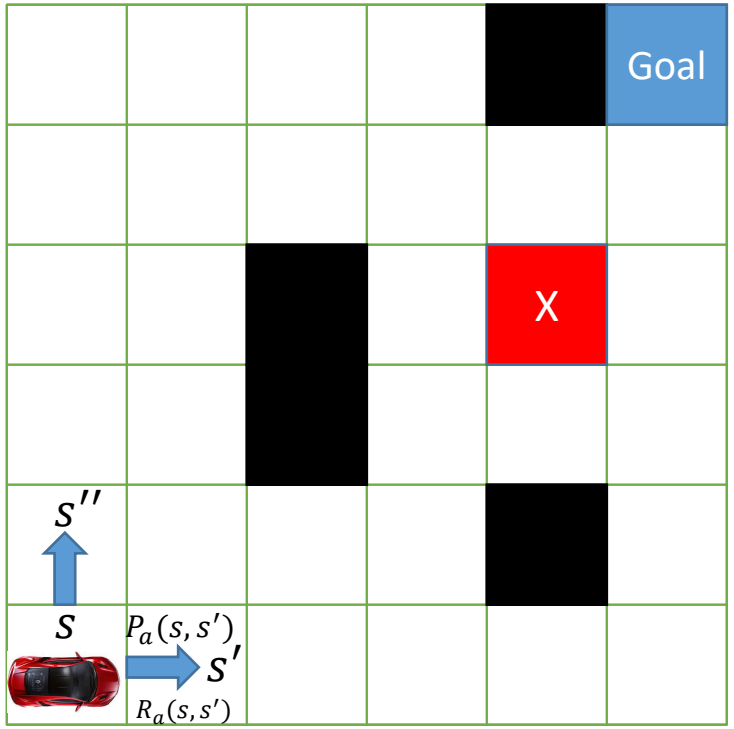


# Markov Decision Process (MDP)

Markov Decision Process (MDP) is a 4 tuple:

$$\mathbf{M} = (X, U, P, R)$$

- $X$ is a set of states, i.e., *the state space*
- $U$ is a set of actions, i.e., *the action space*
- $P_u(x, x')$ is the state *transition probability*
- $R_u(x, x')$ is the immediate *reward*




**How to find a policy for the car to get from a starting location to a goal location?**

# Why MDP is Not Sufficient?

In MDP, $\mathbf{M} = (X, U, P, R)$

- States are fully observable
- We can capture uncertainty in decision-making in the form of transition probabilities

However,

MDP *Does Not* capture sensing uncertainty

Challenges with MDP-based planning:

- Sensor noise gives rise to drift in the state estimation.
- Sensor noise makes the system **partially observable**.

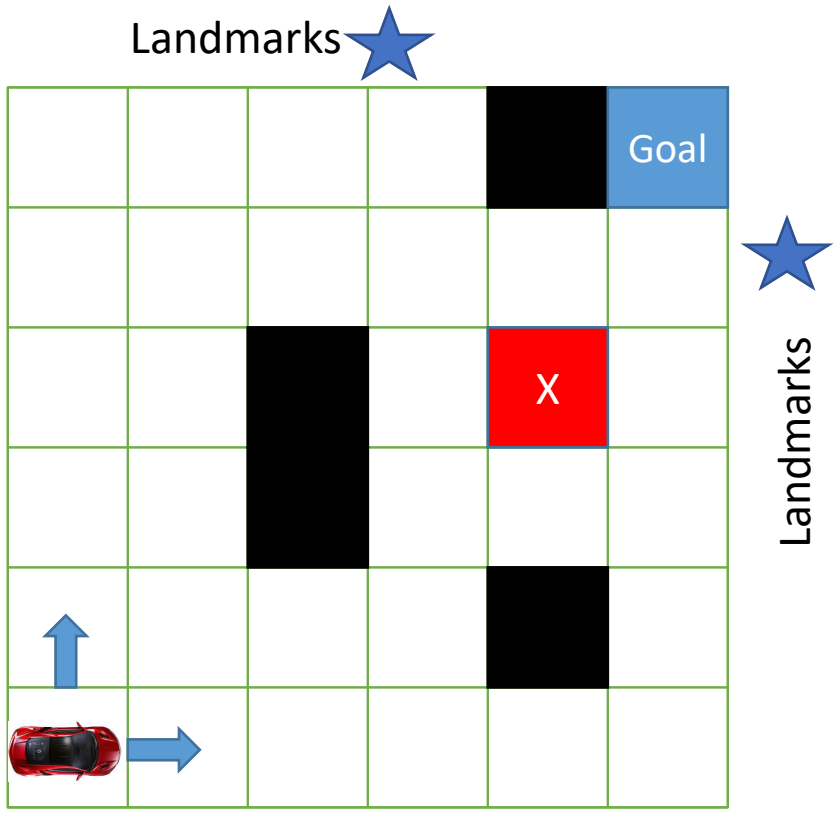
**Detection is noisy and frequently produces false positives or negatives**

# Partially Observable Markov Decision Process (POMDP)

Partially Observable Markov Decision Process (POMDP) is a 6 tuple:

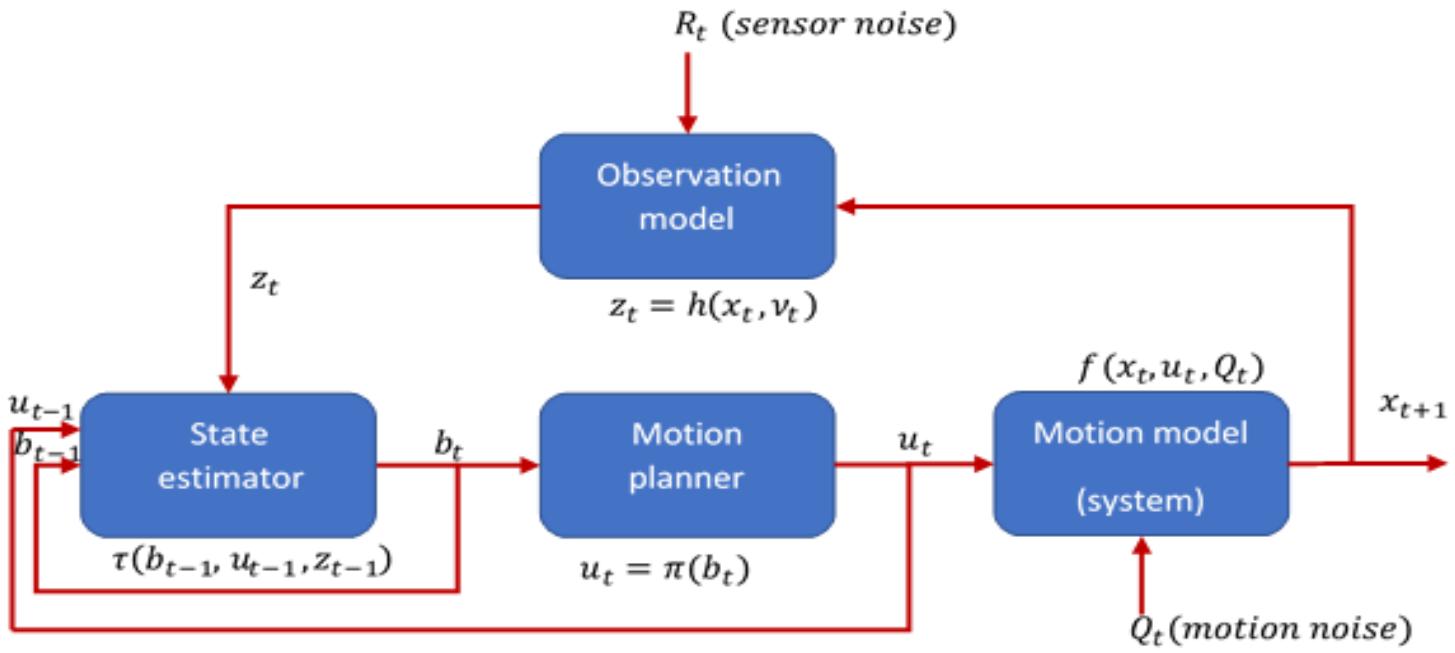
$$\mathbf{PM} = (X, U, \Omega, P, O, R)$$

- $X$ is a set of states, i.e., *the state space*
- $U$ is a set of actions, i.e., *the action space*
- $\Omega$ *is a set of observations,* i.e., *the observation space*
- $P_u(x, x')$ is the state *transition probability*
- $O_a(o, x')$ is the *observation probabilities*
- $R_a(x, o, x')$ is the immediate *reward*




**How to find a policy from a starting location to a goal location under sensing and motion uncertainties?**

# POMDP-based Motion Planning




- Car current state $x_t$ depends on
  - current control $u_t$ and
  - sensor readings $z_t$
- Car current state $x_t$ is affected by
  - sensor noise $R_t$ and
  - motion noise $Q_t$,
- Need to maintain a belief $b_t$ using the state estimator τ.
- Motion planner generates policy π based on the current belief $b$

**Challenge**: Belief space is *much larger* than the state space.

# Needs for Scalable Algorithms

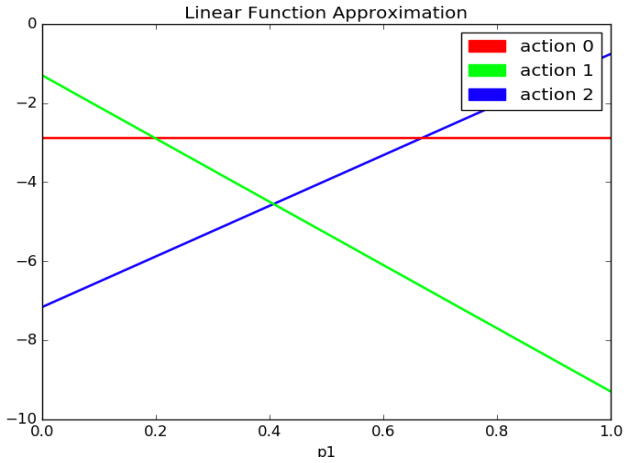
Infinite horizon POMDP-based planning is undecidable!

Finite-horizon POMDP-based planning has a P-SPACE complete complexity!

We need scalable algorithms that strike a balance between exploring a partially observable world and acting in a goal-directed manner

# Point-based Value Iteration

**Hint**: The value function of POMDPs is a piecewise-linear and convex (PWLC)




POMDP value function is PWLC

❖Sampling to find important parts of the belief space and

❖Dynamic-programming updates on PWLC vector representation

Point-based method

The goal-constrained belief space only contains beliefs visited by desired executions

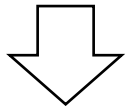
Next Big Idea

[*] Yue Wang, Abdullah Al Redwan Newaz, Juan David Hern´andez, Swarat Chaudhuri, and Lydia E Kavraki. “Online Partial Conditional Plan Synthesis for POMDPs with Safe-Reachability Objectives: Method and Experiments.” *IEEE Transactions on Automation Science and Engineering*, 2021.

# Offline vs Online Policy Synthesis

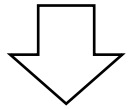
**Offline Synthesis** method that computes an optimal policy before execution.

- **Pros:** the execution is extremely fast when a desired policy is synthesized
- **Cons:** a lot of computation might unnecessarily be done during synthesis



Practical Approach

**Online Synthesis** interleaves planning and execution



Take Away

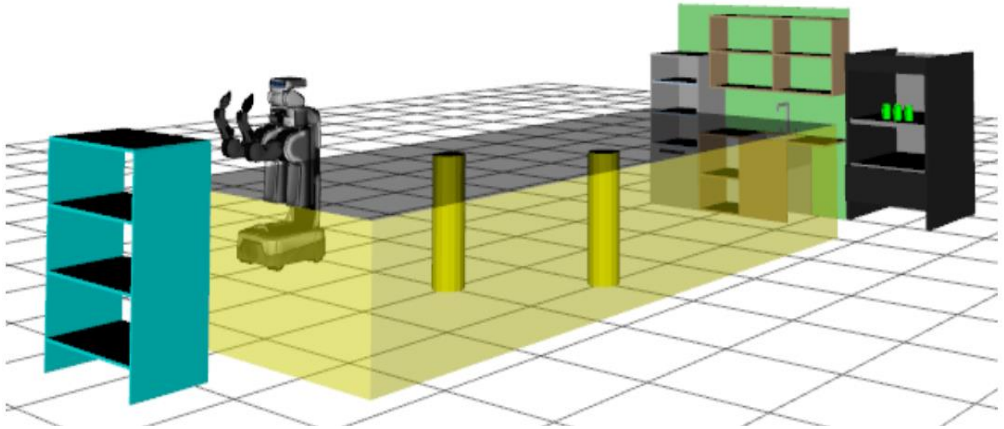
Online planning can take advantage of **fast approximate updates** which often result in a more useful value function than performing a few exact updates

# Online Partial Conditional Plan Synthesis for POMDPs with Safe-Reachability Objectives*

**Problem**: *How to synthesize a valid policy to achieve a qualitative objective which is characterized as a safe-reachability requirement?*

**Proposed approach:**

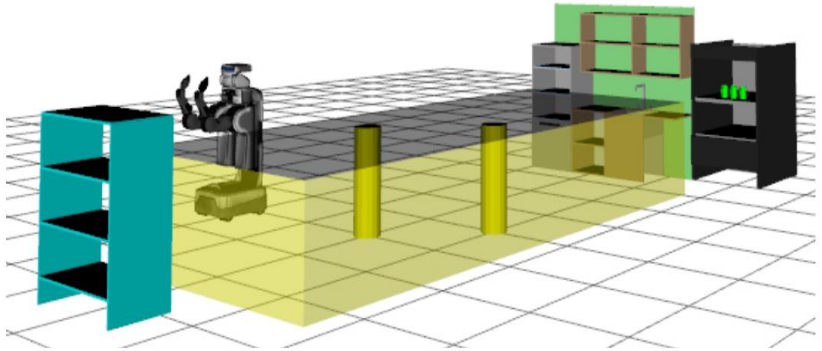
- Construct the **goal-constrained belief space**
- Apply the **point-based value iteration** for exploring the belief space efficiently
- Synthesize the policy in an online way
- Employ the Satisfiability Modulo Theories (SMT) based Model Checker for guaranteeing the safe-reachability property



[*] Yue Wang, Abdullah Al Redwan Newaz, Juan David Hern´andez, Swarat Chaudhuri, and Lydia E Kavraki. "Online Partial Conditional Plan Synthesis for POMDPs with Safe-Reachability Objectives: Method and Experiments." *IEEE Transactions on Automation Science and Engineering*, 2021.

# Problem Statement

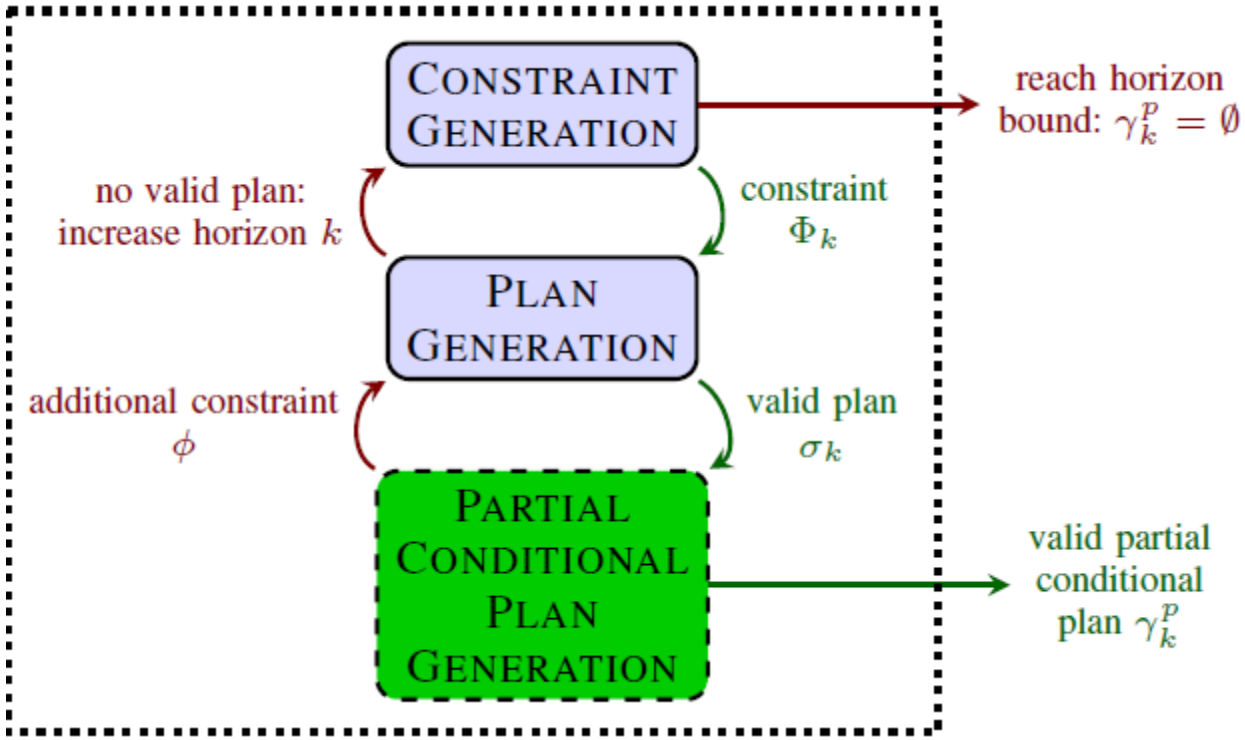
- Given
  - POMDP as an environment
  - Initial Belief as robot initial position
  - Replanning probability bound
  - Safe-reachability objective
  - Horizon bound
- Goal
  - Synthesize a valid k-step partial conditional plan subject to a replanning probability bound



# Online Partial Conditional Plan Synthesis

**Input:** POMDP $P = (\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{O}, \mathcal{Z})$, Initial Belief $b_{init}$, Replanning Probability Bound $\delta_{p_{\text{replan}}}$, Safe-Reachability Objective $\mathcal{G} = (Dest, Safe)$, Horizon Bound $h$
**Output:** A boolean: true - success, false - failure

/* Generate partial conditional plan */
$\gamma_k^p \leftarrow$ PartialContionalPlanSynthesis($P$, $b_{init}$, $\mathcal{G}$, $\delta_{p_{\text{replan}}}$, $h$)
**if** $\gamma_k^p = \emptyset$ **then**
  /* No partial conditional plan: failure */
  **return** false
**repeat**
  $(a, \mathcal{O}_k^p, \nu_k^p) \leftarrow \gamma_k^p$
  Execute action $a$
  Receive observation $o$
  $b_{init} \leftarrow \mathcal{T}_{\mathcal{B}}(b_{init}, a, o)$ /* Update belief */
  **if** $b_{init} \in Dest$ **then**
    /* reach a goal belief: success */
    **return** true
  /* Get next partial conditional plan */
  $\gamma_k^p \leftarrow \nu_k^p(o)$
  $h \leftarrow h - 1$ /* Reduce horizon bound */
**until** $\gamma_k^p = \emptyset$
/* recursively perform OPCPS on new branch */
**return** OPCPS($P$, $b_{init}$, $\mathcal{G}$, $h$)



# Real-world Deployment

# Point-based Value Iteration*


Works well in small-scale POMDPs

⇩

Dynamic Programming to update the belief space

⇩

**Does not scale well in large-scale POMDPs**

[*] Pineau, Joelle, Geoff Gordon, and Sebastian Thrun. "Point-based value iteration: An anytime algorithm for POMDPs." *International Joint Conference on Artificial Intelligence*. 2003.

# Monte-Carlo Policy Synthesis

POMCP proposed by Silver et. al. in 2010

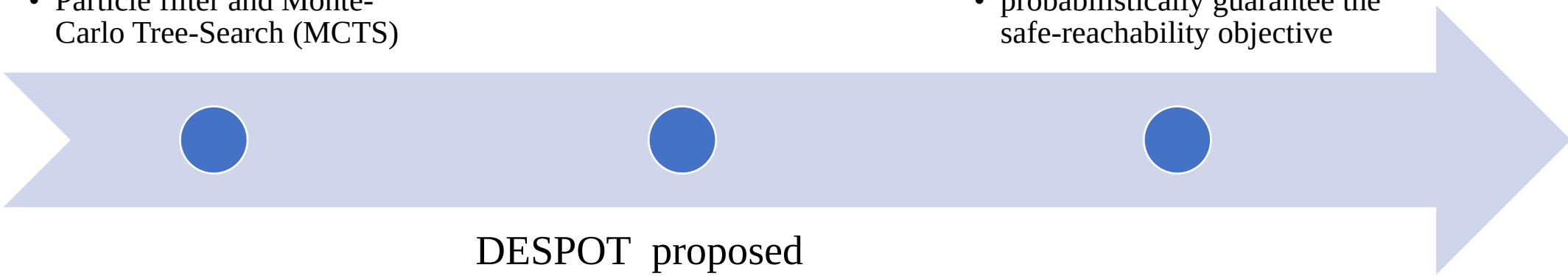
- Particle filter and Monte-Carlo Tree-Search (MCTS)

DESPOT proposed by Somani et. al. in 2013

- reduces variance on MCTS with theoretical guarantee

POSMC proposed By Newaz et. al in 2019

- probabilistically guarantee the safe-reachability objective

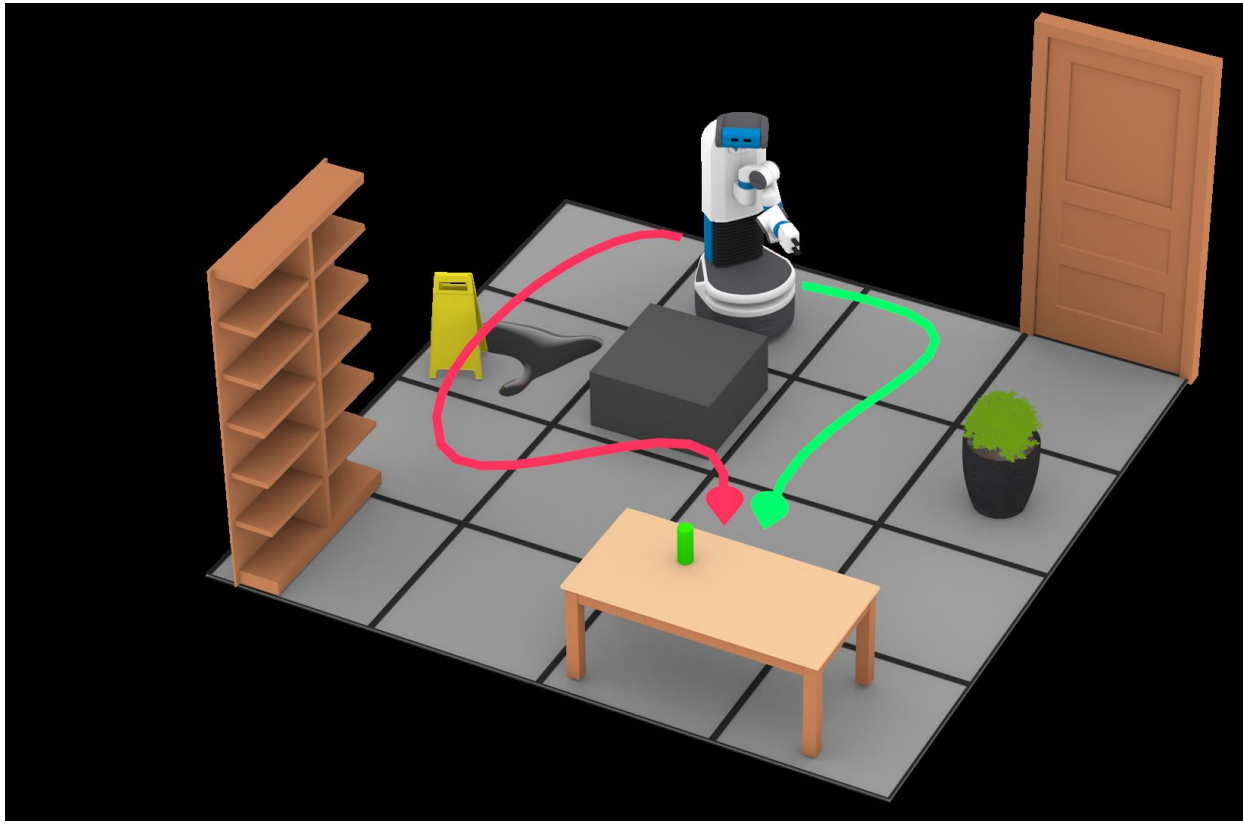


# Monte-Carlo Policy Synthesis in POMDPs with Quantitative and Qualitative Objectives*

**Problem**: *How to synthesize a valid policy to achieve a quantitative objective (e.g., maximizing the reward function) and a qualitative objective which is characterized as a safe-reachability requirement?*

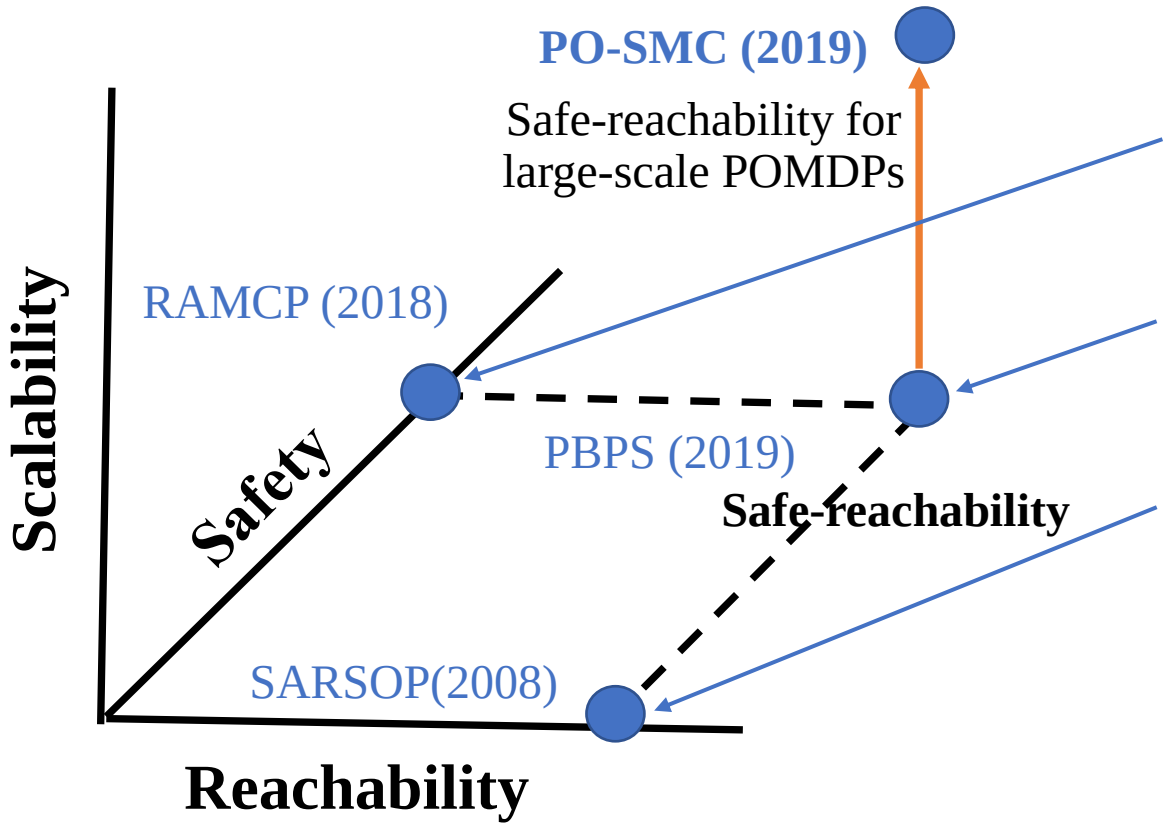
**Proposed approach:**

- Sampling-based method for *large-scale POMDPs*
- Apply the **particle filter** for approximating the belief space
- Monte-Carlo tree search for exploring the belief space efficiently
- Statistical Model Checker for *probabilistically guarantee* the safe-reachability property



[*] Abdullah Al Redwan Newaz, Swarat Chaudhuri, and Lydia E Kavraki. Monte-Carlo policy synthesis in POMDPs with quantitative and qualitative objectives. Robotics: Science and Systems, 2019

# Scalable Constrained POMDP Solver





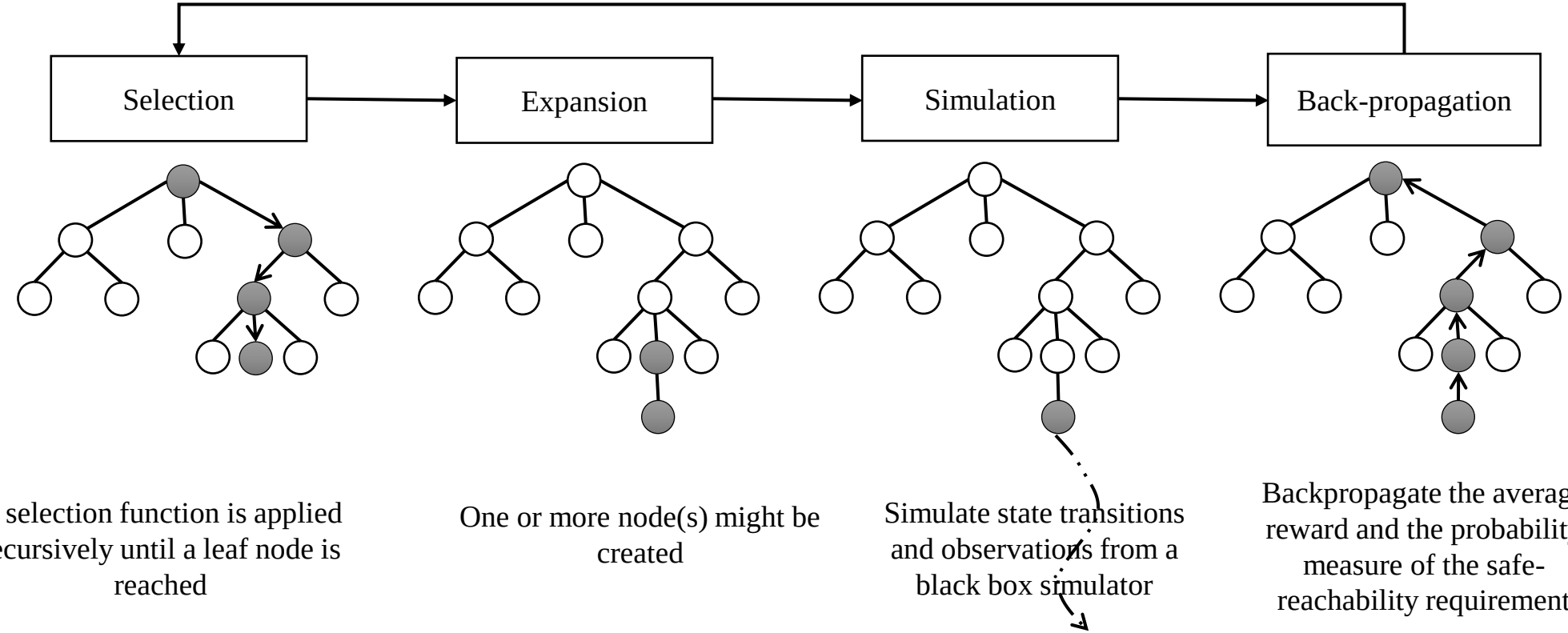
Chatterjee, K., Elgyütt, A., Novotný, P., & Rouillé, O., "Expectation optimization with probabilistic guarantees in POMDPs with discounted-sum objectives." *International Joint Conference on Artificial Intelligence*, 2018.

Y. Wang, S. Chaudhuri, and L. E. Kavraki, "Point-based policy synthesis for POMDPs with boolean and quantitative objectives." *IEEE Robotics and Automation Letters*, 2019.

Kurniawati, H., Hsu, D., & Lee, W. S. "SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces." In *Robotics Science and Systems*, 2008.

# POSMC Algorithm

Run continuously in the allocated time



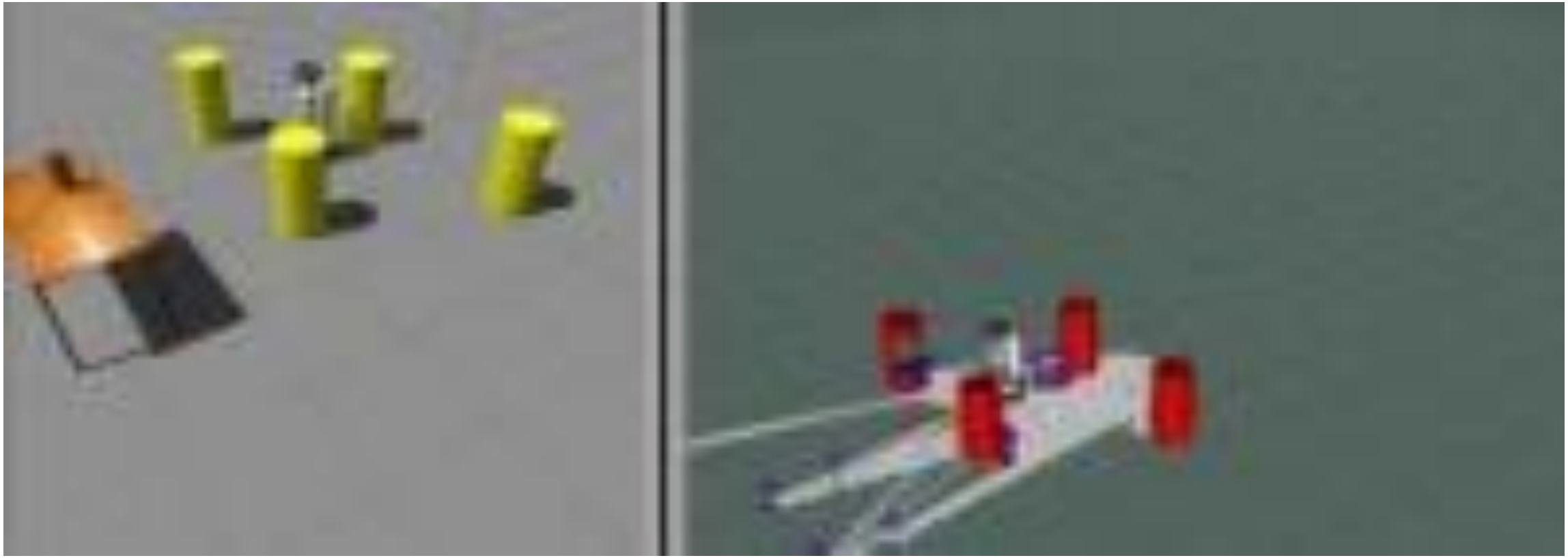

A selection function is applied recursively until a leaf node is reached

One or more node(s) might be created

Simulate state transitions and observations from a black box simulator

Backpropagate the average reward and the probability measure of the safe-reachability requirement

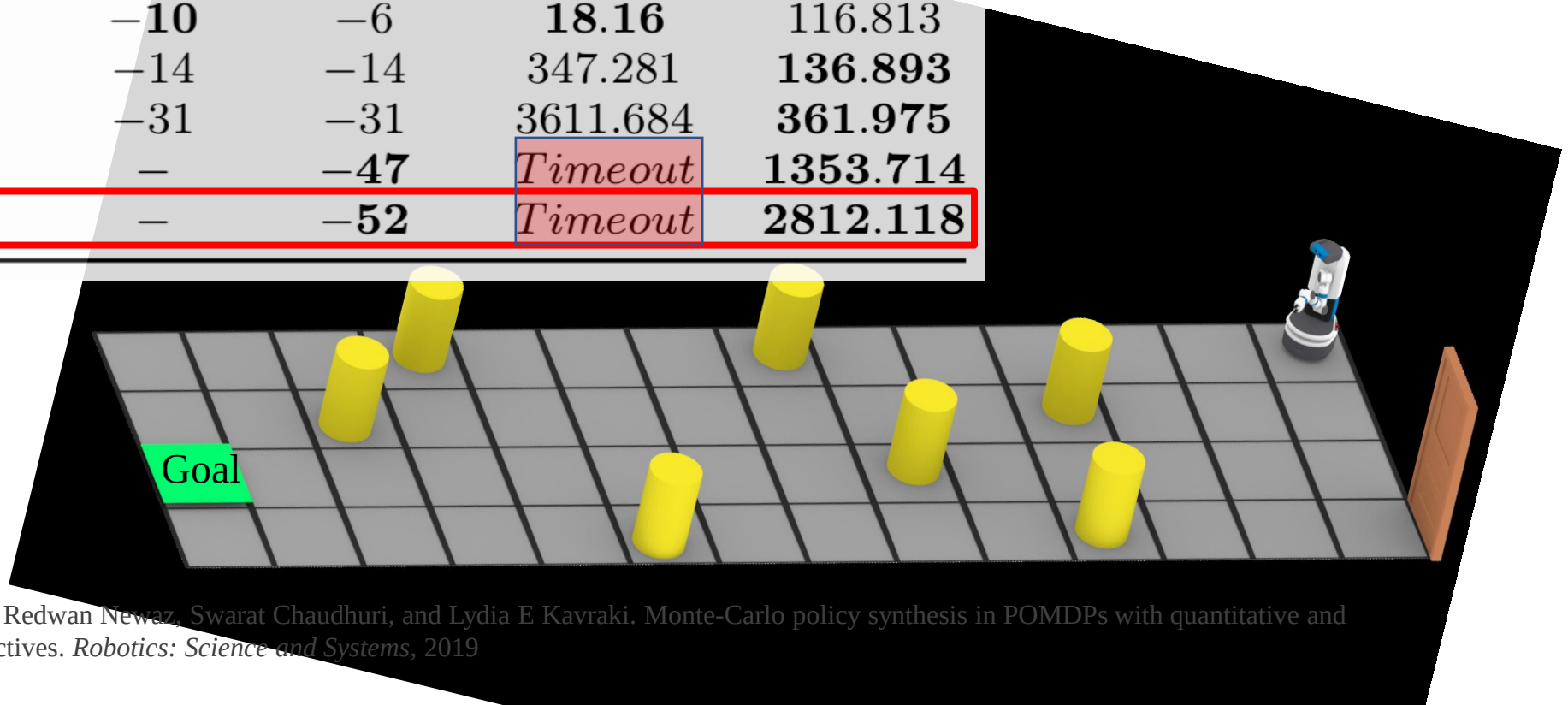
# Simulation Results



[*] Abdullah Al Redwan Newaz, Swarat Chaudhuri, and Lydia E Kavraki. Monte-Carlo policy synthesis in POMDPs with quantitative and qualitative objectives. *Robotics: Science and Systems*, 2019

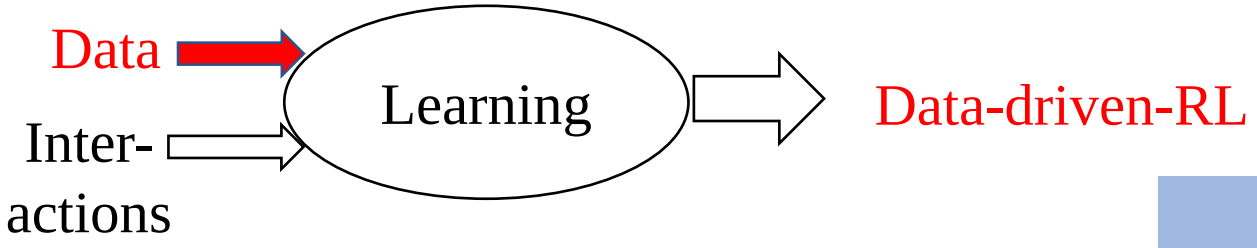
# Benchmarking

| Grid size | Obstacles | PBPS reward | PO-SMC reward | PBPS time (s) | PO-SMC time (s) |
|---|---|---|---|---|---|
| $13 \times 3$ | 1 | $-\mathbf{10}$ | $-6$ | **18.16** | 116.813 |
| $13 \times 3$ | 2 | $-14$ | $-14$ | 347.281 | **136.893** |
| $13 \times 3$ | 3 | $-31$ | $-31$ | 3611.684 | **361.975** |
| $14 \times 4$ | 6 | – | $-\mathbf{47}$ | *Timeout* | **1353.714** |
| $14 \times 4$ | 7 | – | $-\mathbf{52}$ | *Timeout* | **2812.118** |



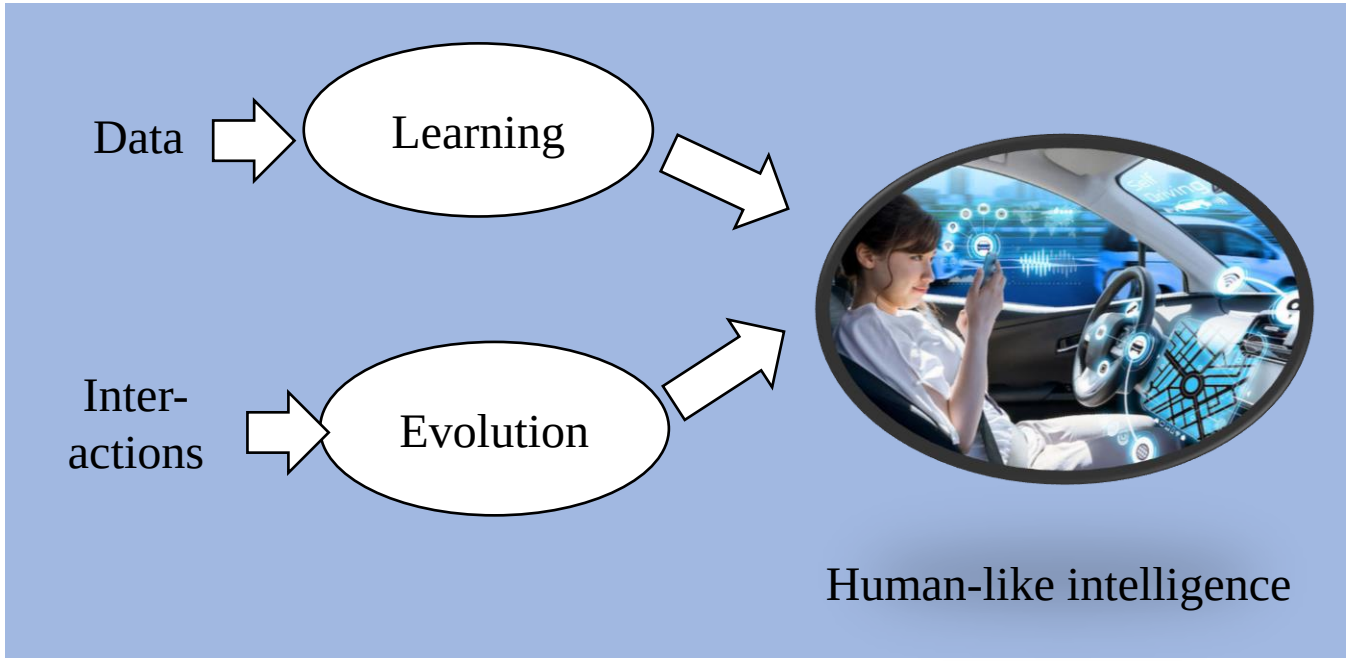

[*] Abdullah Al Redwan Newaz, Swarat Chaudhuri, and Lydia E Kavraki. Monte-Carlo policy synthesis in POMDPs with quantitative and qualitative objectives. *Robotics: Science and Systems*, 2019

# Learning from Data and Interaction




**Data-driven-RL leverages**

- learn from interacting from the environment
- looking at past interactions which is data

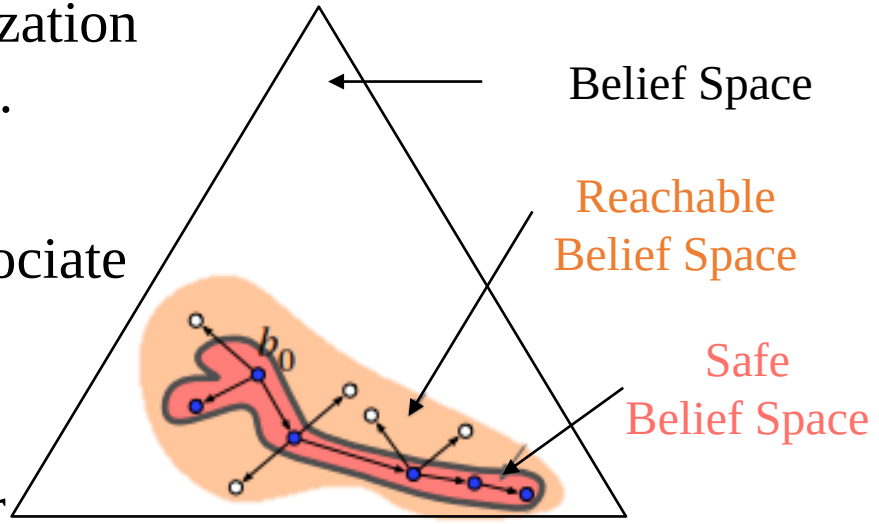


# Constrained data-driven RL

**Challenge:** The standard formulation of RL-based reward maximization cannot be applied to achieve an objective with multiple constrains.

In standard RL, the action incurs only rewards, but we need to associate them with different types of costs to handle multiple constraints.

**Solution:** The constrained POMDP is an appealing framework for dealing with multi-objective sequential decision-making problem




We can use the *constrained POMDP-based RL* to find an optimal policy that maximizes the expected cumulative rewards while bounding each of expected cumulative costs below certain levels

# Long-Term Autonomy for AUVs Operating Under Uncertainties in Dynamic Marine Environments*

**Problem**: *How to synthesize optimal policies with cost and energy constraints in dynamic marine environments which are modeled as large-scale continuous POMDPs?*


**Proposed approach:**

- Combination of interactive learning and data-driven prediction
- Deep Neural Network to predict the time-varying ocean dynamics
- Proposed an Energy Cost-Constrained Partially Observable Monte-Carlo Planner
- Optimization of the trade-off among the rewards, the energy costs, and the collision costs in a continuous state space

[*] Abdullah Al Redwan Newaz, Tauhidul Alam, Murad Reis Gregory, Leonardo Bobadilla, and Ryan N Smith. Long-term autonomy for AUVs operating under uncertainties in dynamic marine environments. Robotics and Automation Letters, 2021

# Energy Cost-Constrained POMDP

- ECC-POMDP problem entails to optimize multiple objectives such as
  - goal-driven rewards,
  - costs for dynamic obstacle avoidance,
  - and energy awareness using ocean currents.
- Formally defined as $\langle B, U, T, R, C, \hat{c}, \xi, \hat{e}, \gamma \rangle$

where,

- $C$ cost function
- $\xi$ energy function
- $\hat{c}$ cost threshold
- $\hat{e}$ energy threshold

- Our objective is to compute an optimal policy
  - maximizes the expected cumulative reward

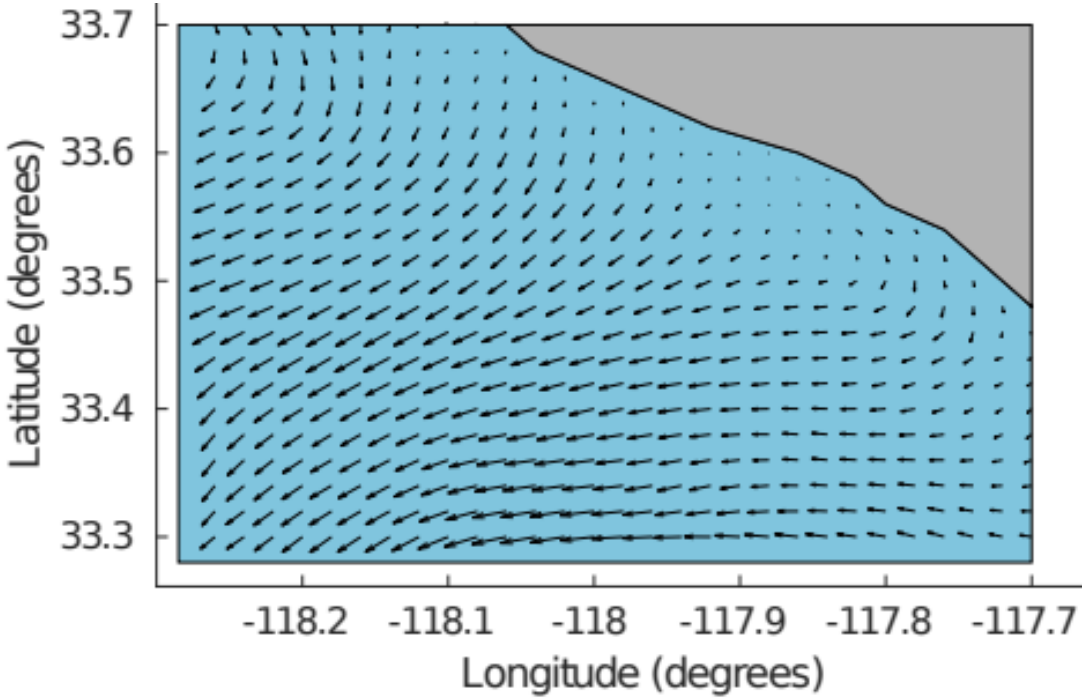
$$\max_{\pi} V_R^{\pi}(b) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \gamma^t R(b_t, \pi(b_t)|b_0)\right]$$

  - while bounding the expected cumulative costs and energies:

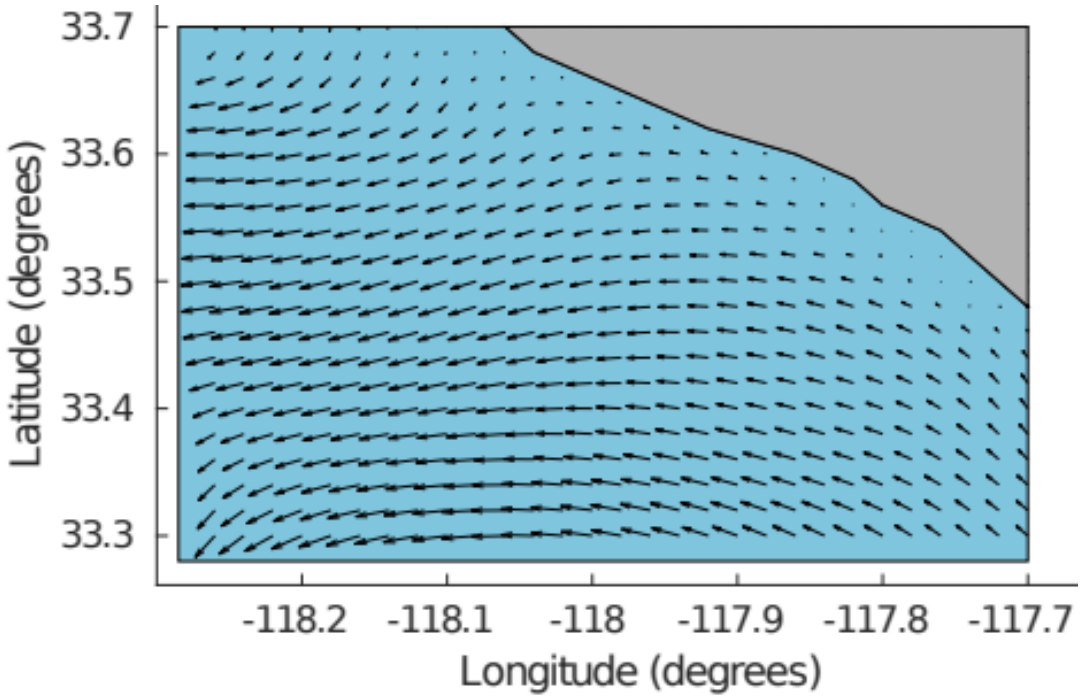
$$V_C^{\pi}(b) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} C(b_t, \pi(b_t)|b_0)\right] \leq \hat{c},$$

$$V_{\xi}^{\pi}(b) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{\infty} \xi(b_t, \pi(b_t)|b_0)\right] \leq \hat{e}.$$

# LSTM Neural Network to Predict Ocean Dynamics




Ground Truth Ocean Dynamics




Predicted Ocean Dynamics

# Algorithm and Simulation

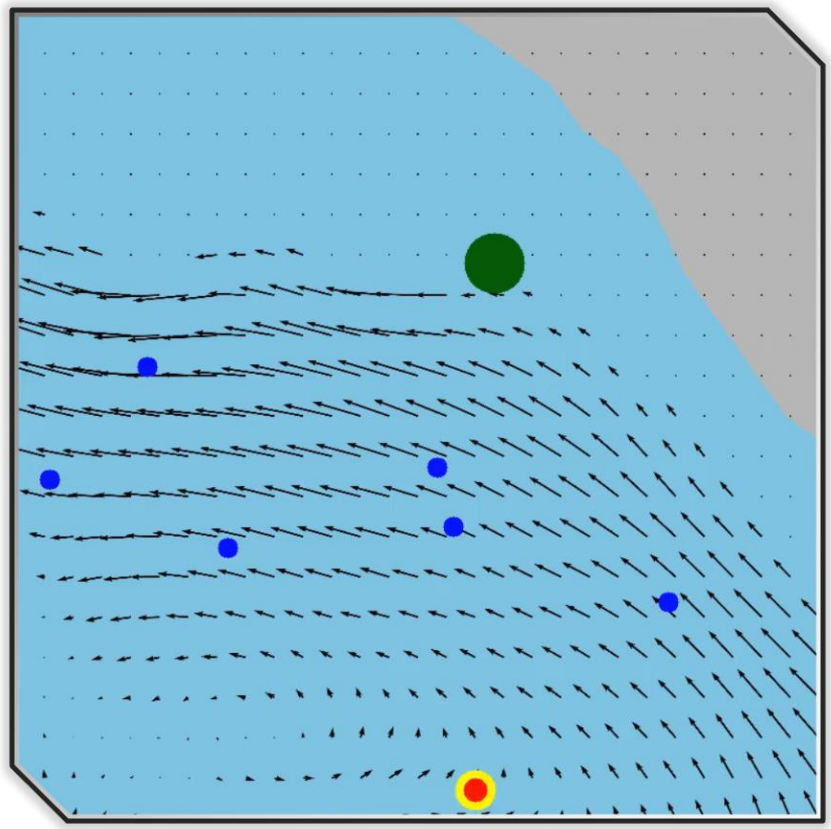
**Input:** Belief-state CMDP, $\langle B, U, T, R, C, \hat{c}, \xi, \hat{e}, \gamma \rangle$ and learned ocean model $\mathcal{F}$

**Output:** Optimal Policy, $\pi^*_\lambda$

1: Initialize $\lambda_c$ and $\lambda_\xi$
2: **repeat**
3: Policy Evaluation for $\pi(u \mid b)$
4: For a given belief $b$ and an action $u$, compute $Q_R(b,u), Q_C(b,u)$, and $Q_\xi(b,u;\mathcal{F})$
5: Obtain the joint belief-action value $Q^{\oplus}_\lambda(b,u)$
6: Policy Improvement on $\langle B, U, T, R - \lambda_c \hat{c} - \lambda_\xi \hat{e}, \gamma \rangle$
7: Update the policy based on the joint action-value as:

$$\pi^*_\lambda = \arg\max_{u \in U(h)} Q^{\oplus}_\lambda(b,u)$$

8: **until** Converge()



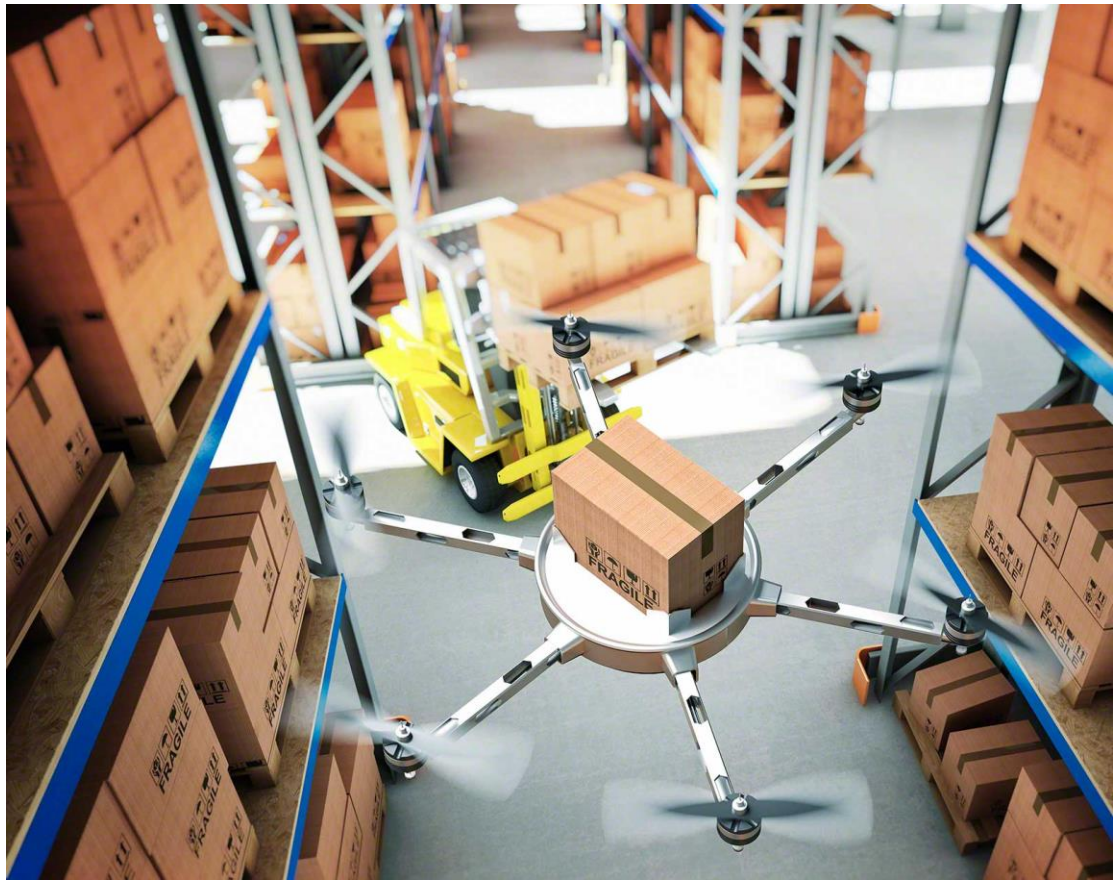
Our ECC-POMCP Solution

# Experiments

| Exp. | Traj. Length (m) | | Avg. Velocity (m/s) | | Mission Time (min) | |
|---|---|---|---|---|---|---|
| | **ECC** | CC | **ECC** | CC | **ECC** | CC |
| 1 | 469.75 | 456.76 | 0.547 | 0.452 | 14.30 | 16.82 |
| 2 | 685.38 | 456.76 | 0.624 | 0.375 | 18.28 | 20.28 |
| 3 | 523.60 | 456.76 | 0.559 | 0.440 | 15.59 | 17.29 |

# Task and Motion Planning (TAMP) under Uncertainties

- TAMP integrates the generation of high-level tasks in a discrete space and the execution of low-level actions in a continuous space
- TAMP often used in the logistics and object transportation industries
- TAMP be framed as a hybrid search problem that requires a robot to reason about its actions in both discrete and continuous spaces
- TAMP is a challenging problem under uncertainties as it handles the interaction between task and planning modules



# Problem Overview

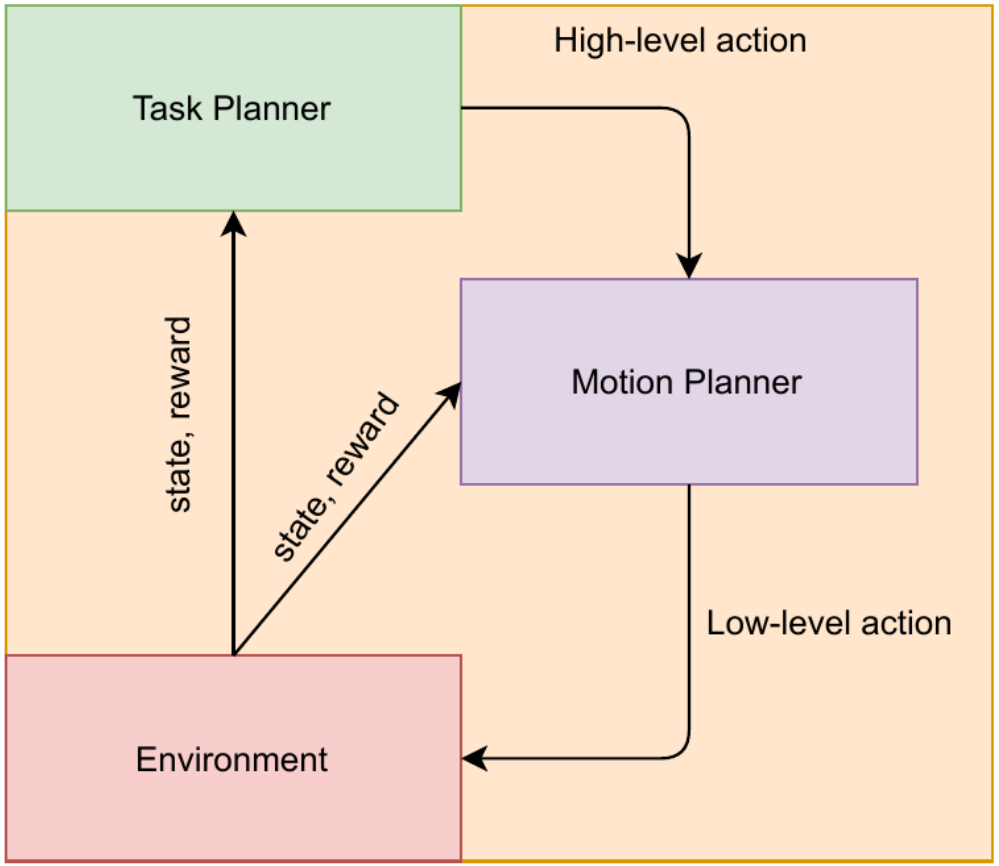
The TAMP under uncertainties is decomposed into two interconnected planning problems:

**High-level Task Planning:** it provides discrete actions over the task domain

**Low-level Motion Planning:** it provides continuous actions over the motion domain

**Challenges:**

- ❑ Stochastic environment
- ❑ Sequential planning
- ❑ High-level task actions need to be validated by low-level motion planner



# Problem Formulation

**High-level Task Planning**
Modeled as a Multi-agent Markov Decision Process

An MMDP is a tuple $\mathcal{M} = \langle S, s_0, \alpha, A, P, R \rangle$

Where,

- $S$ is a finite set of states
- $s_0$ is a set of initial locations
- $\alpha$ is a set of robots
- $A$ is the joint action space
- $P$ is the transition probability function
- $R$ is the real-valued reward function

**Low-level Motion Planning**
Modeled as a Markov Decision Process

An MDP is a tuple $M = \langle X, U, p, r \rangle$

Where,

- $X$ is a set of states
- $U$ is the set of control (action)
- $p$ is the transition probability function
- $r$ is the real-valued reward function

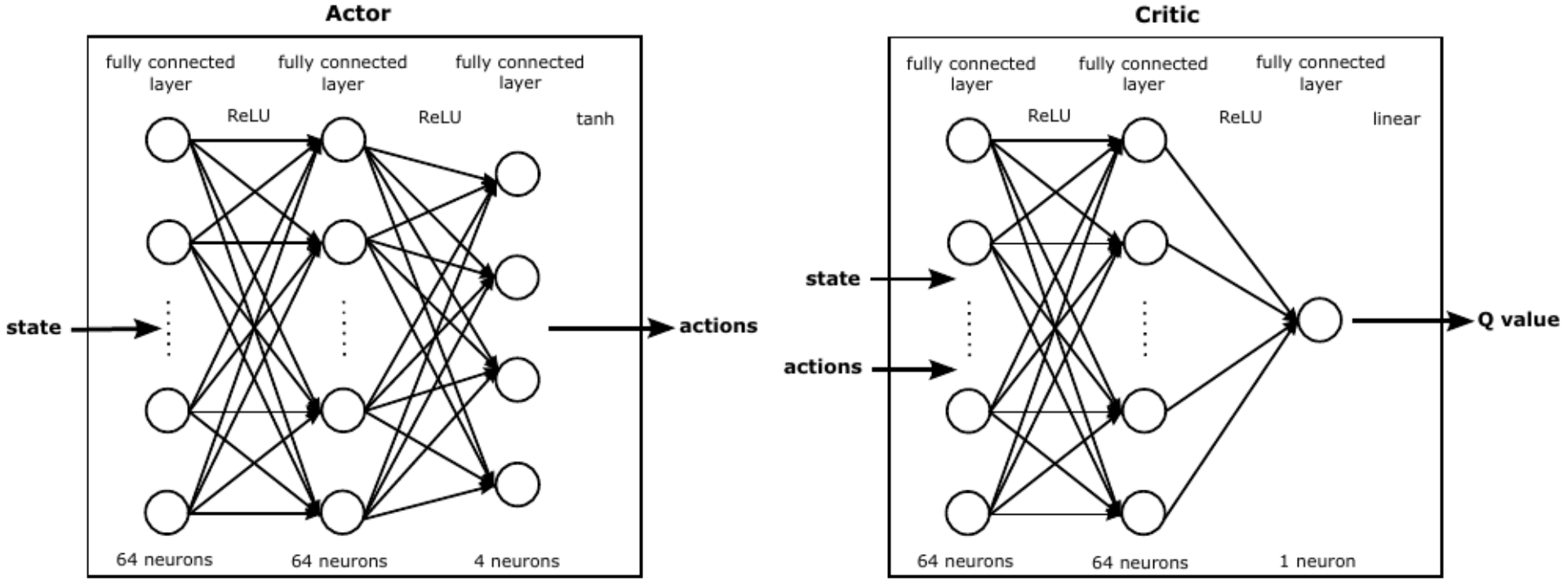
***How to find an optimal control policy that prescribes motions for a robot based on its current state and high-level task requirements?***

$$\pi^* = \arg\max_{a_t \in A, u_t \in U} \mathbb{E}\left\{ \int_{t=0}^{\infty} \gamma^t r\left(x_t, u_t, x_{t+1} | a_t, s_t\right) \right\}$$

# Method: Learning Motion Policy

**Motion Planner**

- Deep Deterministic Policy Gradient for learning an optimal control policy
- DDPG algorithm that consists of two models to learn the policy in a continuous state space setting: Actor and Critic.

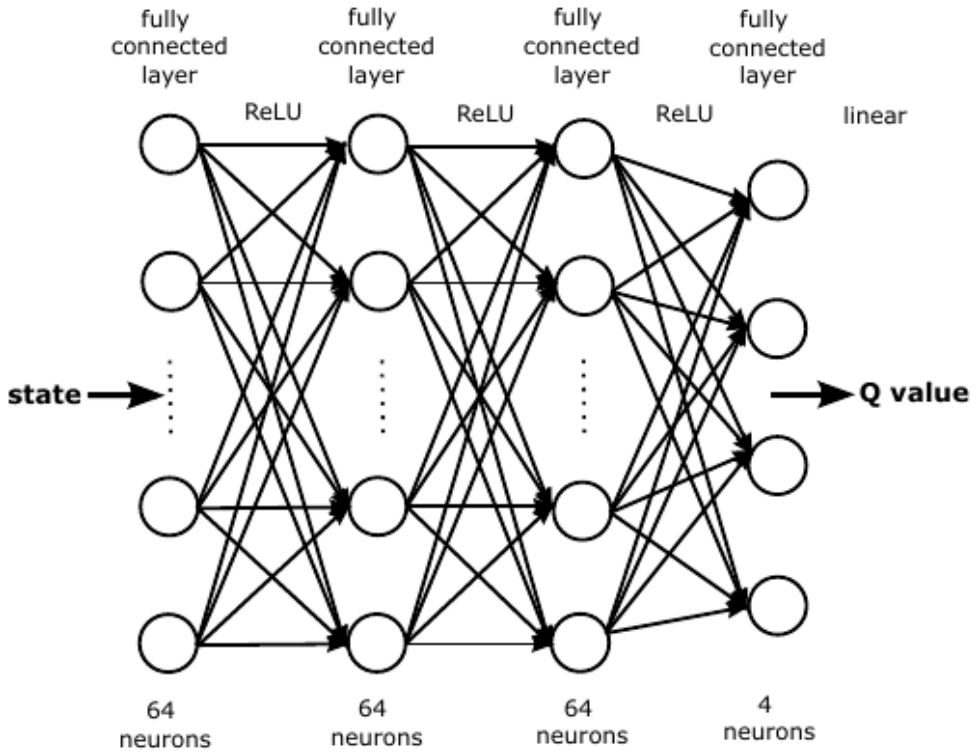



- The Actor is a policy network that takes the state as input and outputs actions in the continuous space.
- The Critic is a Q-value network that takes a state and actions as input and outputs the Q-value.
- Finally, we use curriculum learning for improving control policy

# Method: Learning Task Policy

**Task Planner**

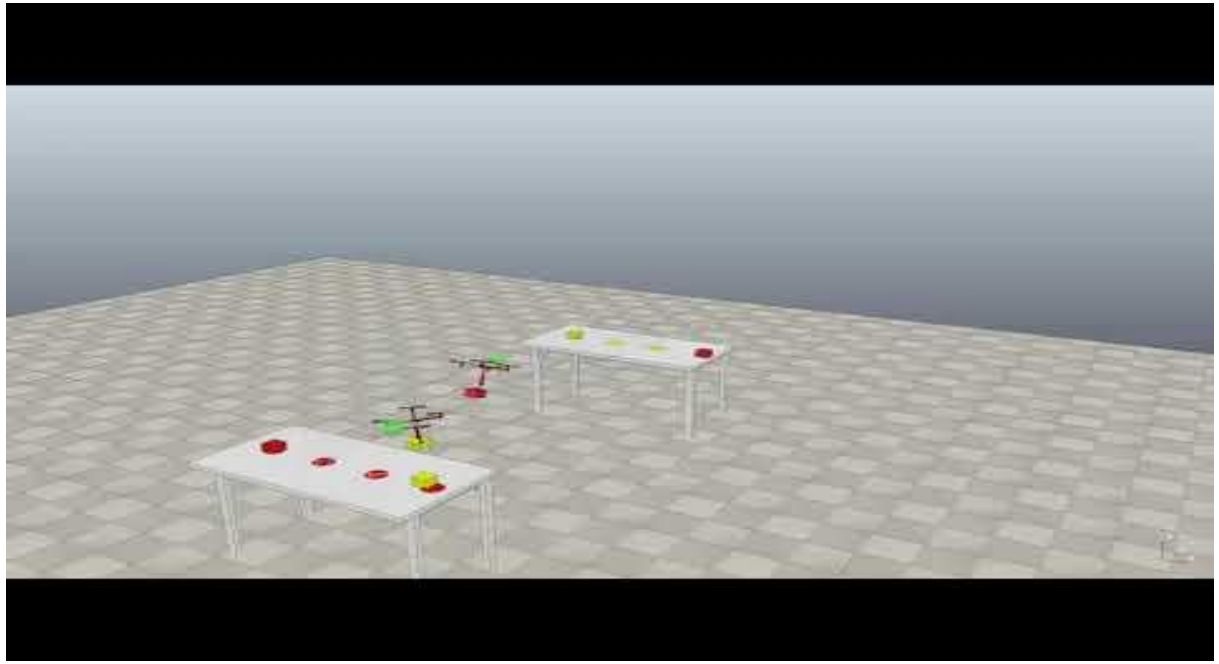
- Deep Q-learning Network (DQN) for learning a discrete task policy




- DQN leverages task planning under uncertainties to learn a policy from abstract states and actions.
- DQN can jointly evaluate actions from multiple robots to compute the high-level tasking policy.

# Learning to pick and place objects

Two flying robots pick and place objects from one table to another while avoiding robot-robot collisions. All the red objects from the first table are transferred to the second table and all the yellow objects are transferred from the second table to the first table.



**Pick and place tasks in a warehouse environment**