ECE276A: Sensing & Estimation in Robotics
Lecture 1: Introduction

Nikolay Atanasov

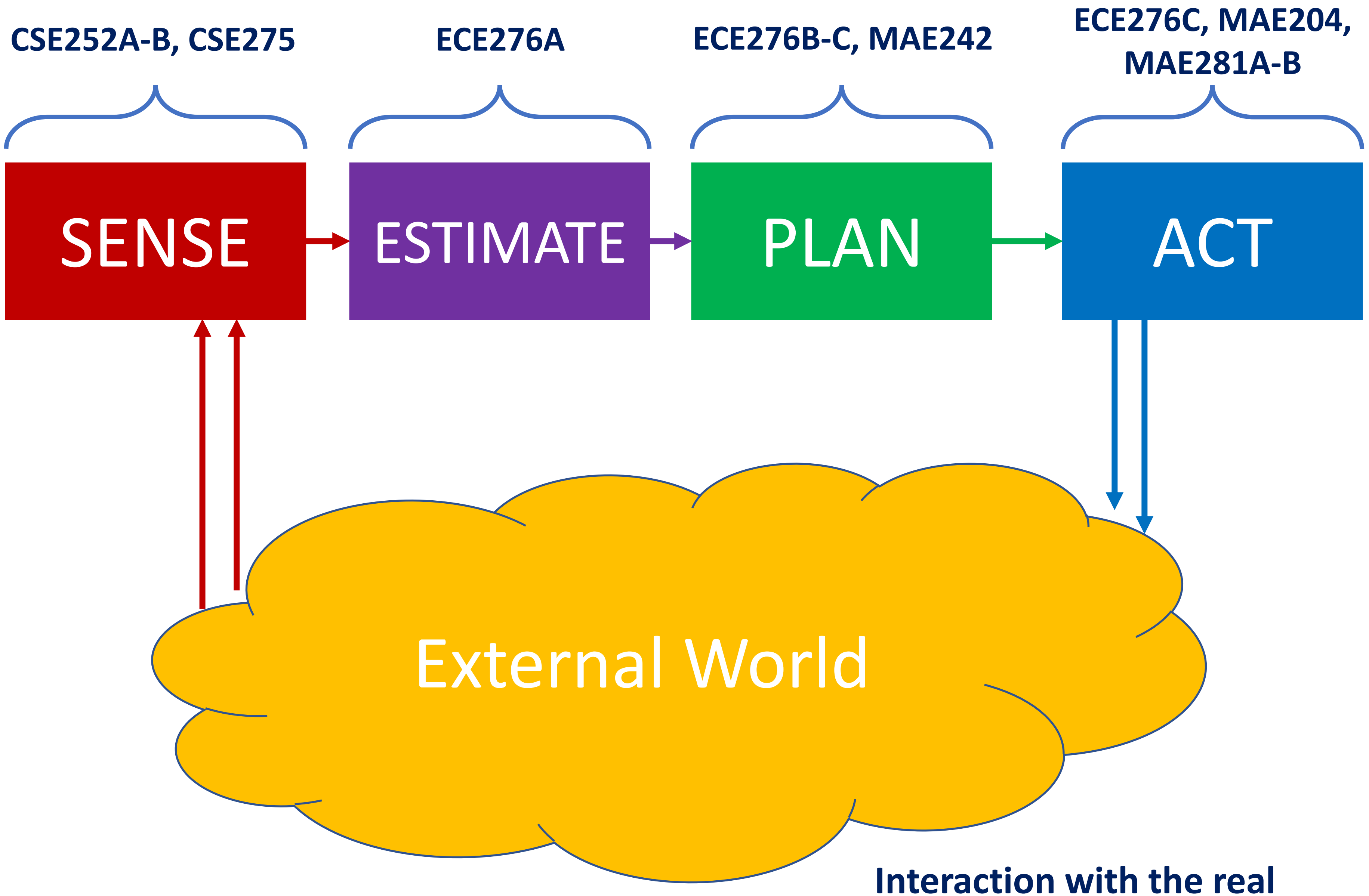
natanasov@ucsd.edu

UC San Diego
JACOBS SCHOOL OF ENGINEERING
Electrical and Computer Engineering

Mobile Robot Autonomy

- Mobile robot autonomy is a research area relying on tools from:
  - **Computer Vision & Signal Processing:** to deal with real-world signals in real time, e.g., filtering sound, convolving images, recognizing objects
  - **Probability Theory & Estimation Theory**: to deal with uncertainty caused by sensor and actuator noise, computation and communication delays, and environment changes and estimate robot and world states
  - **Optimization Theory**: to plan the best robot behavior according to a suitable performance criterion
  - **Control Theory**: to execute the planned robot behavior
  - **Machine Learning**: to improve the models and performance based on data (supervised, self-supervised, unsupervised, and reinforcement learning)

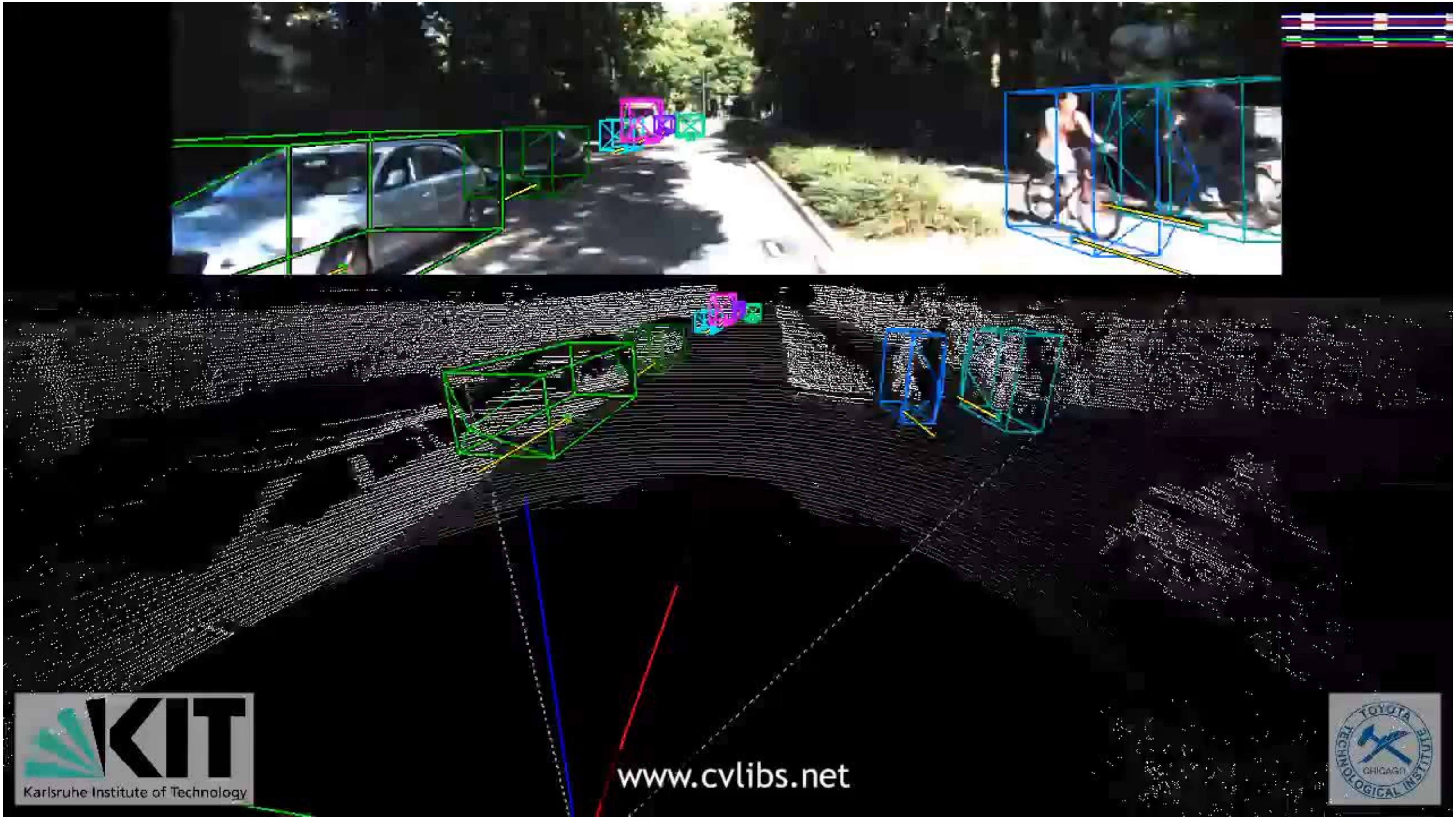
Robot Autonomy



Interaction with the real world introduces uncertainty!

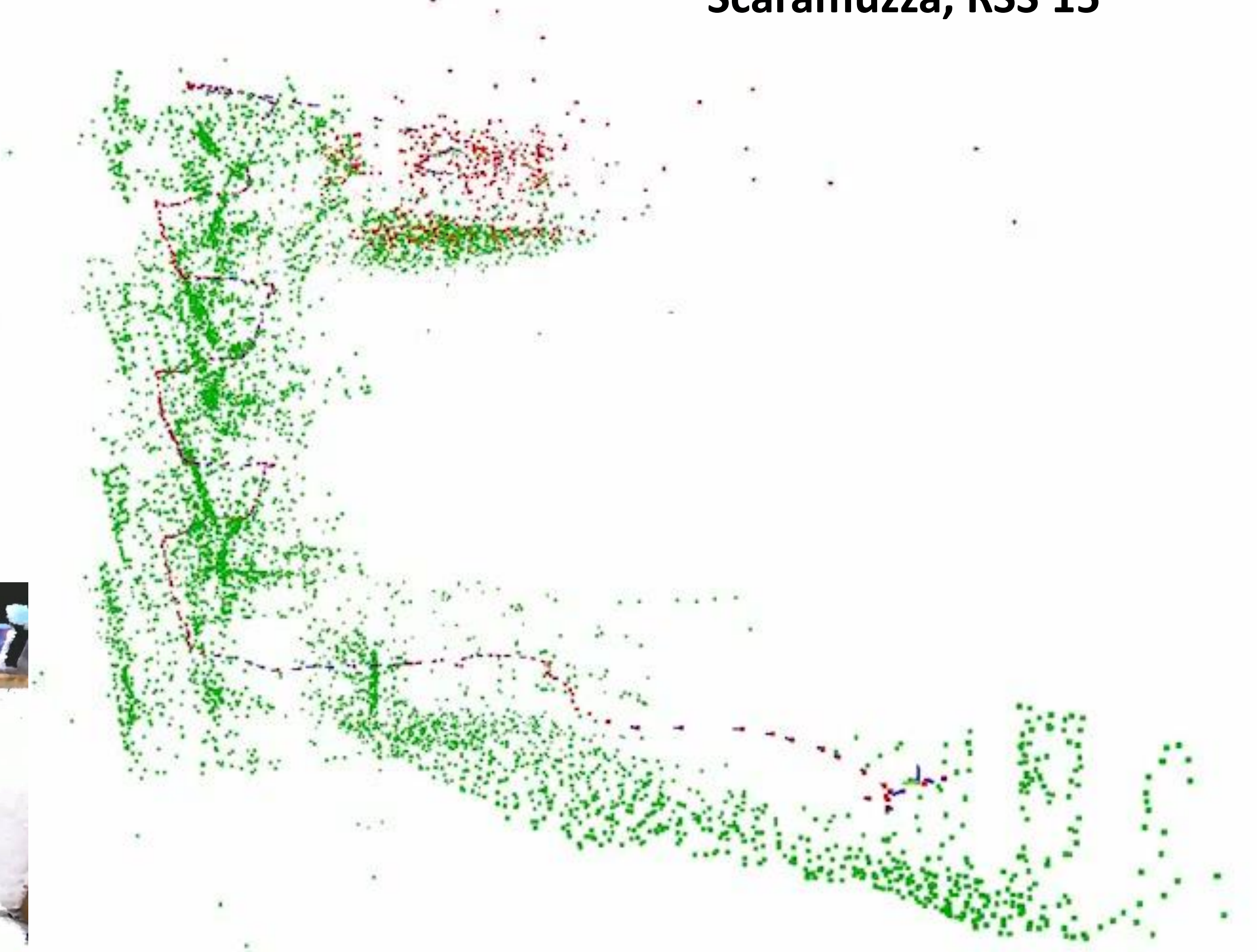
SENSE

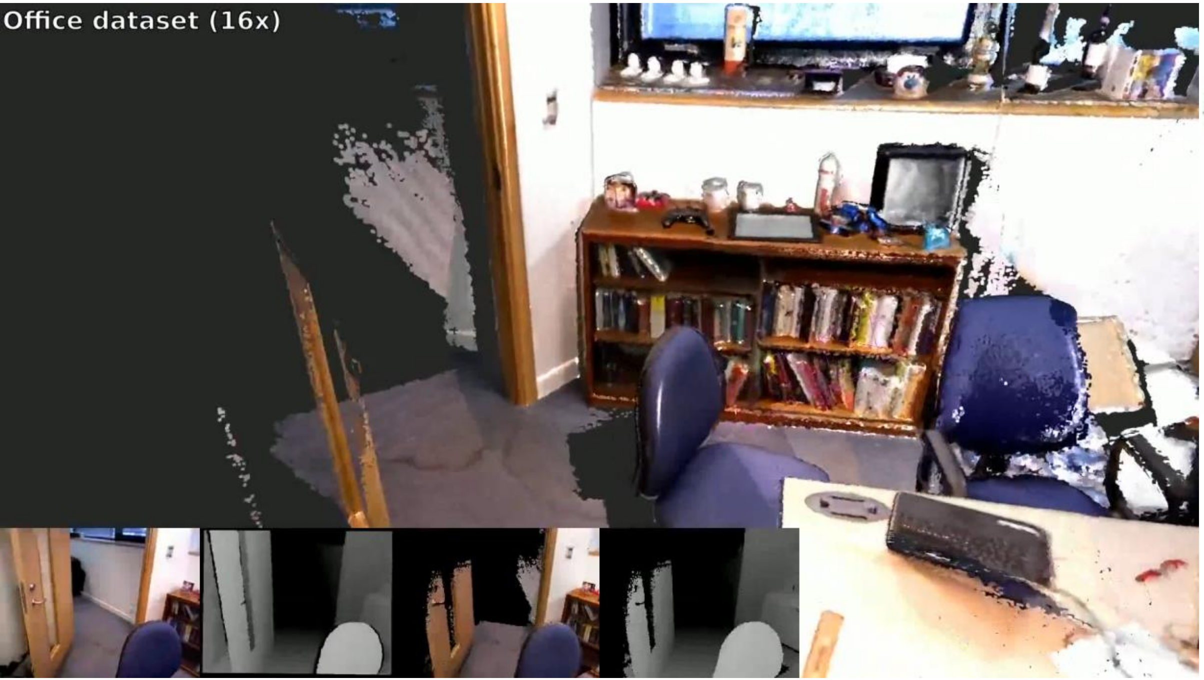
KITTI Dataset, Geiger, Lenz, Urtasun, CVPR'12



ESTIMATE

Forster, Carlone, Dellaert, Scaramuzza, RSS'15

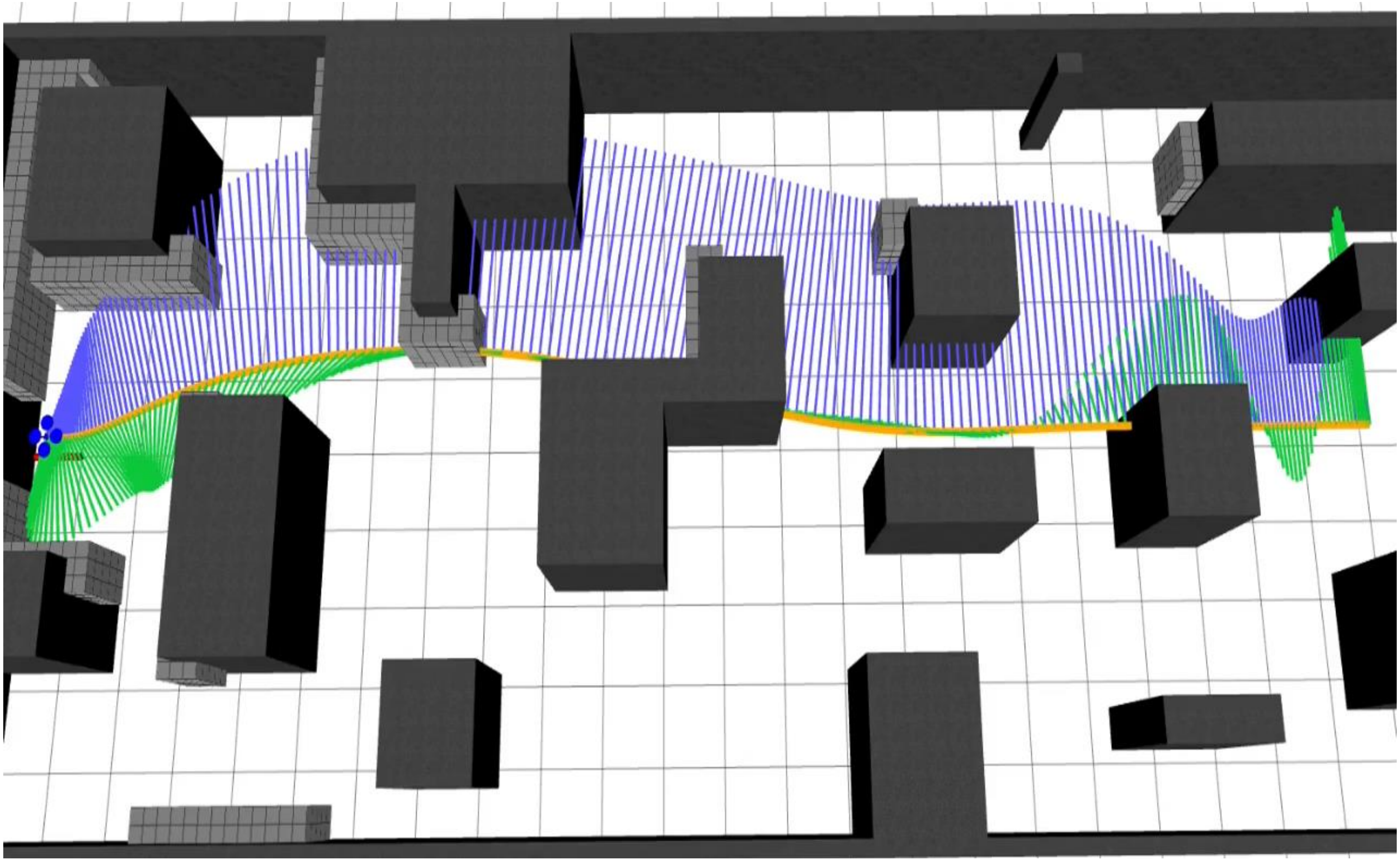




Whelan, Leutenegger, Salas-Moreno, Glocker, Davison, RSS'15

PLAN

Liu, Atanasov, Mohta, Kumar, IROS'17



ACT

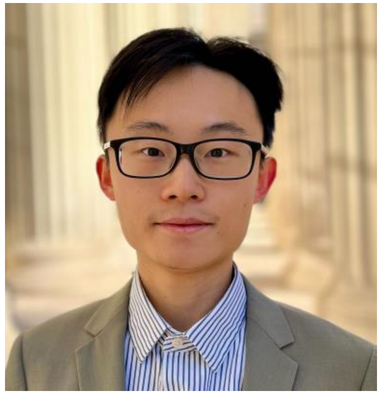
Mellinger, Michael, Kumar, IJRR'12

ECE 276A: Sensing & Estimation in Robotics

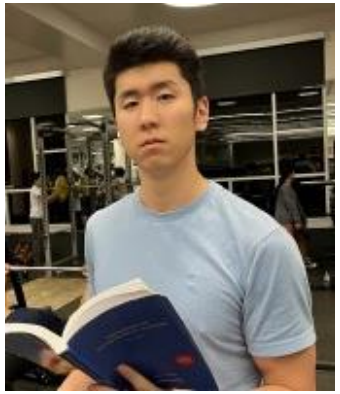
- The course will cover:
  - **Sensing**: image formation, projective geometry, rotations, features, optical flow
  - **Estimation**: unconstrained optimization, probabilistic models, maximum likelihood estimation (MLE), Bayesian filtering, simultaneous localization and mapping (SLAM)
- Course website: https://natanaso.github.io/ece276a
  - Schedule, reading materials, and assignments
  - Grades: **GradeScope** (**SIGN UP!**)
  - Discussion: **Piazza** (**SIGN UP!**)
  - Office hours/TA sessions
- **Piazza**:
  - Great place for discussion, I encourage you to use it!
  - In addition to asking questions, responding is a great way to strengthen your understanding!
- References (**optional**):
  - State Estimation for Robotics: Barfoot
  - Probabilistic Robotics: Thrun, Burgard & Fox
  - An Invitation to 3-D Vision: Ma, Kosecka, Soatto & Sastry
  - Bayesian Filtering and Smoothing: Sarkka

Teaching Team

- Instructor
  - **Nikolay Atanasov**
  - Associate Professor, ECE Department
  - Email: natanasov@ucsd.edu



- Teaching Assistant:
  - **Yulun Tian**
  - Postdoc, ECE Department
  - Email: yut034@ucsd.edu



- Teaching Assistant:
  - **Jay Paek**
  - MS Student, ECE Department
  - Email: jpaek@ucsd.edu

- Teaching Assistant:
  - **Yinzhuang Yi**
  - PhD Student, ECE Department
  - Email: yiyi@ucsd.edu

Warning About Prerequisites

- This is a challenging graduate course
- I want everyone to learn about robotics, so the prerequisites are not strictly enforced
- As graduate students, **I expect you to be mature and carefully evaluate whether you are prepared to take the course**
- Prerequisites:
  - **Probability Theory**: if you have not had a good course on probability theory, it is too early to take ECE276A
  - **Linear Algebra**: if you have not had a good course on linear algebra, it is too early to take ECE276A
  - **Programming experience**: if you have not done a programming project of reasonable complexity before, it is too early to take ECE276A
- You will enjoy this course and learn a lot more if you have the right background
- Every year some students ignore this, overestimate their prior preparation or available time, and have an unpleasant experience

Grading

- Assignments:
  - 1 academic integrity quiz (**required**) – **due Jan 17th on Canvas**
  - 3 theoretical homework assignments (16% of the grade total)
  - 3 programming assignments in **python** with project reports (18% of the grade each)
  - Final exam (30% of the grade): calculator + double-sided cheat sheet

- There is sufficient time to complete every assignment if you start **early**

- Late submissions and deadline extensions will not be possible because our schedule is tight (1 week background review, 3 weeks per project & homework, final exam)

- Letter grades will be assigned based on the class performance, i.e., you do not need to and will not be able to get everything right in order to get a good grade
  - Tentative rubric: 85+: A; 80-85: A-; 75-80: B+; 65-75: B; 60-65: B-; 55-60: C+; ...
  - The rubric may be adjusted at the instructor's discretion

Collaboration and Academic Integrity

- Every assignment in this course is **individual**
- You are encouraged to discuss the assignments with other students in general terms but the **work you do and turn in should be completely your own**
- An important element of academic integrity is fully and correctly acknowledging any materials taken from the work of others – provide references for papers and **acknowledge in writing people you discuss the assignments with**
- **Cheating will not be tolerated**
- Instances of academic dishonesty will be penalized via grade reduction and may be referred to the Office of Student Conduct for adjudication

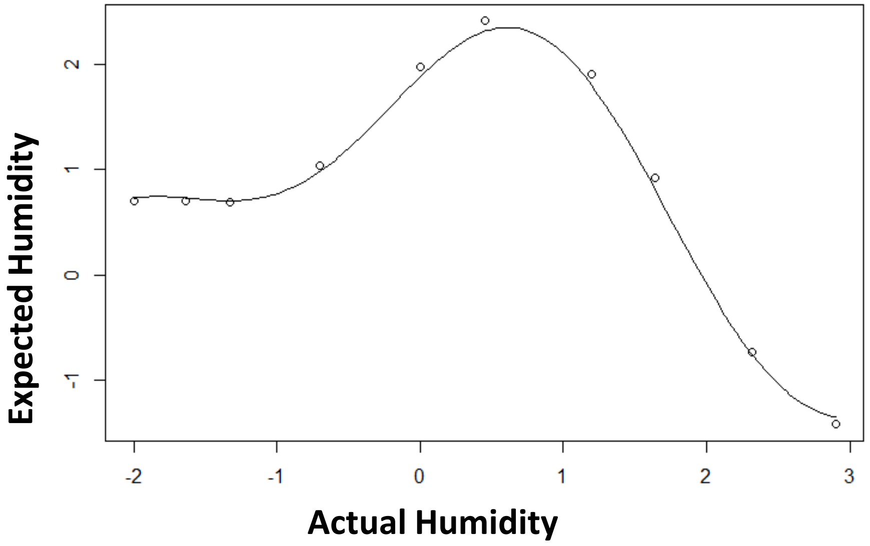
Project Report: Suggested Structure

1. **Introduction**: brief discussion of what the problem is and why it is important
   It is important to monitor the humidity of plants and choose optimal watering times. In this paper, we present an approach to select the best watering time in the week from given historical humidity data.

2. **Problem Formulation**: brief rigorous mathematical statement of the problem, not the solution!
   Let $f: \mathbb{R} \to \mathbb{R}$ be the average historical weakly humidity.
   ***Problem**: Find a watering time $t^* \in \mathbb{R}$ such that $t^* = \arg\min_t f(t)$*

3. **Technical Approach**: description of the ideas, equations, algorithms used to solve the problem
   The minimum of a function appears at one of its critical points $\{s \in \mathbb{R} \mid f'(s) = 0\}$. We find all the roots of f' and select the smallest one as the optimal watering time.

4. **Results:** figures showing qualitative and quantitative performance supported by discussion of what was successful and what fails
   The method performs well as shown in Fig. 1. The performance could be improved if real-time humidity measurements are used to update f.



Project Report: Examples

https://natanaso.github.io/ref/Wang_LearningNavigation_SCR19.pdf

https://natanaso.github.io/ref/Schlotfeldt_AdversarialInfoAcquisition_RSS18_Workshop.pdf

https://natanaso.github.io/ref/Lauri_MonteCarloTreeSearch_ICRA15_Workshop.pdf

Learning Navigation Costs from Demonstration via Differentiable Planning

Tianyu Wang
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, U.S.A.
tiw161@eng.ucsd.edu

Nikolay Atanasov
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, U.S.A.
natanasov@eng.ucsd.edu

***Abstract*—This paper focuses on learning cost functions that capture desirable behavior in tasks demonstrated by an expert. In a task such as autonomous navigation in unknown environments, it is possible to obtain sequences of observations (e.g., images), states (e.g., poses), and control inputs but not direct queries of the underlying task-specific cost function. Hence, it is necessary to design a planning algorithm that, depending on the current cost representation, computes a control policy and propagates its error with respect to the demonstrations back to the cost representation. Our contribution is a probabilistic environment representation for local observation updates and cost function design, and a differentiable planning algorithm that performs state expansions only on a subset of promising states. Our complete model can be trained end-to-end and improves upon Value Iteration Networks and the Dyna-Q algorithm.**

***Index Terms*—Inverse Reinforcement Learning, Differentiable Planning, Cost Learning**

I. Introduction

Autonomous robot systems increasingly require operation in unstructured, partially known, and dynamically changing environments. One core challenge for safe and robust navigation is that the *true* cost function of a navigation task, requiring safe, dynamically feasible, and efficient behavior, is generally not known while expert demonstrations can be utilized to uncover the underlying cost function [1], [2]. In addition, humans and animals can navigate successfully with partial knowledge of the environment and adapt when facing new obstacle configuration based on prior experience. Motivated by this observation, we focus on learning a cost function from demonstration that is not universally accurate over the state and control space but rather captures task-relevant information and leads to desirable behavior.

Our main contribution is an end-to-end differentiable model that combines a cost function representation and an efficient planning algorithm (see Fig. 1). The novelty of our approach is that the proposed model is fully differentiable, which allows using gradient-based optimization to improve the parameterized cost function. Our experiments show that the end-to-end differentiable model learns task-specific cost functions and improves upon Value Iteration Networks (VIN) [3] and the Dyna-Q algorithm [4] by handling partial and noisy observations. In summary, we offer the following contributions:

We gratefully acknowledge support from NSF CRII RI IIS-1755568.

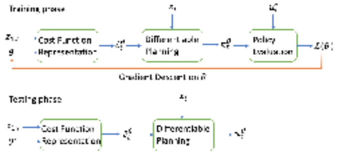


Fig. 1: Architecture for learning cost function representations from demonstrations via differentiable planning. During training, the goal is to learn a cost function parameterization θ based on demonstrations, consisting of states $x_{1:t}$, controls $u^*_{1:t}$, and partial observations $z_{1:t}$, so that a control policy generated based on the learned cost incurs minimum loss $\mathcal{L}(\theta)$. Both the cost function representation c^θ_t and the control policy π^θ_t need to be differentiable with respect to θ for error backpropagation. During testing in a new environment, online observations $z_{1:t}$ and the trained parameters θ^* provide the cost function necessary to generate a control policy.

- A cost function representation that incorporates a log-odds occupancy representation of the environment, updatable using a parameterized observation model.
- An efficient planning algorithm, which performs local convolutional operations encoding Bellman backups *only* on a subset of promising states. We guarantee that the output policy is differentiable with respect to the input cost function.
- An end-to-end differentiable model that learns task-specific cost functions from expert demonstrations by backpropagating policy performance loss through the planning algorithm and the cost representation.

II. Problem Formulation

Consider a robot navigating in an unknown environment with the task of reaching a goal state $x_g \in \mathcal{X}$. Let $x_t \in \mathcal{X}$ be the discrete time robot state. For a given control input $u_t \in \mathcal{U}$, the robot state evolves according to known deterministic dynamics: $x_{t+1} = f(x_t, u_t)$. Let m^* be a function

Adversarial Information Acquisition

Brent Schlotfeldt
GRASP Laboratory
University of Pennsylvania
Philadelphia, PA 19104
Email: brentsc@seas.upenn.edu

Nikolay Atanasov
Dept. of Electrical and Computer Engineering
University of California San Diego
La Jolla, CA 92093
Email: natanasov@ucsd.edu

George J. Pappas
GRASP Laboratory
University of Pennsylvania
Philadelphia, PA 19104
Email: pappasg@seas.upenn.edu

***Abstract*—This paper considers a non-cooperative two-player game modeling the problem of adversarial information acquisition in robotics. Each robot is equipped with a sensor, and is tasked with choosing control inputs that maximize an information (e.g. entropy) about the other player, while keeping its own state as uncertain to the other player as possible, subject to the available sensors. This adversarial information gathering problem has applications in surveillance, or search-and-rescue missions where the agent whose state is to be estimated may try to actively avoid the sensing robot. We formulate the problem of adversarial information acquisition, and provide an initial solution based on a variant of Monte Carlo Tree Search for simultaneous action games.**

I. Introduction

In this paper, we consider the problem of two robots interacting in an adversarial game where each robot attempts to estimate the state of its adversary, while keeping its own state hidden. This problem can have applications in search-and-rescue applications, where the agent to be found is mobile, and actively evades the sensing robot. In these problems, it is important to both accurately localize the target agent, while keeping one's own state hidden so that the target's ability to actively evade is reduced.

There is much prior work in the literature concerning the dynamics of pursuit-evasion in mobile robotic settings[3]. Approaches to the pursuit-evasion problem are split between considering the worst-case evader (adversary), and using probabilistic frameworks which consider the expected case. A common theme in the pursuit-evasion literature is the objective of reducing the distance to the evader to zero, or forcing the evader into a sensing footprint. In contrast, our problem is formulated using a probabilistic approach which optimizes an information theoretic quantity, namely entropy, about the distribution of the target to be tracked. Rather than closing distance to the target, our approach aims to produce the best estimate of the target's state, subject to the sensors available. Our previous work considers the information acquisition problem for target tracking [1], however this work assumes that the target being tracked moves independently of the sensing robot, and crucially is not trying to actively evade the sensing robot. In this work, the problem formulation is symmetric in the sense that the adversary is trying to maximize information gained about us, and also minimize the information we can gain about it.

We also draw attention to some biological inspiration for this problem. Motion camouflage [9] is a strategy utilized by dragonflies, which enables them to capture their prey by minimizing the optical flow of their motion. Mischiati and Krishnaprasad [8] consider the problem of mutual motion camouflage, where two agents each pursue each other, but attempt to maintain a constant bearing to avoid detection by the other agent. Our problem is related, but rather than considering pursuit-evasion, we consider the dynamics of an *adversarial information gathering game*.

In the most general case, the information acquisition game proposed is a stochastic game and is difficult to solve. McEneaney [7] discusses a class of stochastic games with finite-dimensional solutions and dynamic programming algorithms to solve them. With some assumptions on the motion and observation models of the agents in our game, the problem can be simplified to a deterministic game. McEneaney [6] introduces a curse-of-dimensionality free max-plus method for deterministic game problems, which is likely to be very applicable to the linear Gaussian version of the information-theoretic game introduced here. Additionally, Grünwald and Dawid [4] present a game-theoretic argument that maximizing entropy and minimizing worst-case expected loss are duals of each other. A comprehensive treatment on adversarial reasoning, is provided in the book by Kott and McEneaney [5]. The approach taken in this work is a variant of Monte Carlo Tree Search [11], for simultaneous action games. We present the details of this approach in Sec. III.

II. Problem Formulation

Consider a two-player partial information game with simultaneous moves. Each player $i \in \{1, 2\}$ has a state $x_{i,t}$ that evolves according to the following motion model:

$$x_{i,t+1} = f_i(x_{i,t}, u_{i,t}, w_{i,t}) \tag{1}$$

where $u_{i,t} \in \mathcal{U}_i$ is a *finite* space of admissible moves (control inputs) and $w_{i,t}$ is a random variable specifying the motion noise. Player i can observe its own state $x_{i,t}$ and chooses its moves with the objective of tracking the evolution of the state of the other player. Each player is equipped with a sensor used to collect information about the other player according to the following observation model:

$$z_{i,t} = h_i(x_{i,t}, x_{j,t}, v_{i,t}) \tag{2}$$

Active Object Recognition via Monte Carlo Tree Search

Mikko Lauri, Nikolay Atanasov, George J. Pappas, and Risto Ritala

***Abstract*—This paper considers object recognition with a camera, whose viewpoint can be controlled in order to improve the recognition results. The goal is to choose a multi-view camera trajectory in order to minimize the probability of having misclassified objects and incorrect orientation estimates. Instead of using *offline* dynamic programming, the resulting stochastic optimal control problem is addressed via an *online* Monte Carlo tree search algorithm, which can handle various constraints and provides exceptional performance in large state spaces. A key insight is to use an active hypothesis testing policy to select camera viewpoints during the rollout stage of the tree search.**

I. Introduction

The goal of this paper is to choose a sequence of views for an RGB-D camera in order to identify the class and orientation of an object of interest (see Fig. 1). Unlike many existing approaches, which consider a next-best-view problem [1], [2], [3], we plan a multi-view camera trajectory to minimize the probability of having misclassified objects and incorrect orientation estimates. In previous work [4], we addressed a similar stochastic optimal control problem by casting it as a partially-observable Markov decision process. A point-based approximate solver [5] was used to obtain a non-greedy policy offline. Since repeated observations of the object from the same viewpoint provide redundant information, it is desirable to disallow viewpoint revisiting. The drawback of computing a policy offline is that revisiting and occlusion constraints are hard to incorporate and if the environment were to change, the computed policy would no longer be useful. The idea of this paper is to apply Monte Carlo tree search (MCTS, [6], [7]) to the active object recognition problem. MCTS is a best-first *online* planning approach which can handle various constraints and has exceptional performance in large challenging domains such as game solving [8], [9] and belief-space planning in robotics [10], [11], [12].

II. Problem Formulation

Let the camera pose at time t be $x_t \in \mathcal{X} \subset SE(3)$, where $\mathcal{X}$ is a finite set of viewpoints on a sphere centered at the object's location (see Fig. 1). At time t, the camera can move to any of the viewpoints in $\mathcal{X}$ and pays a cost $g(x_{t-1}, x_t)$ which captures the energy expenditure. Let the true (unknown) class of the observed object be $c \in \mathcal{C}$. We formulate hypotheses about the class and orientation of the object:

$H(c, r)$: the object class is $c \in \mathcal{C}$ with orientation $r \in \mathcal{R}(c)$,

M. Lauri and R. Ritala are with the Department of Automation Science and Engineering, Tampere University of Technology, P.O. Box 692, FI-33101, Tampere, Finland, {mikko.lauri, risto.ritala}@tut.fi.
N. Atanasov and G. Pappas are with GRASP Lab, University of Pennsylvania, Philadelphia, PA 19104, USA, {atanasov, pappasg}@seas.upenn.edu. This work was supported by TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

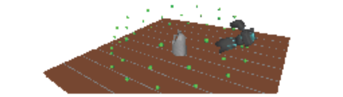


Fig. 1: Setup for the active object recognition problem. The camera position is restricted to a set of viewpoints (green) on a sphere centered at the object's location. The task is to choose a camera control policy, which minimizes the movement cost and the probability of misclassification.

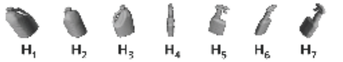


Fig. 2: An example set of hypotheses about the class and orientation of an unknown object

where $\mathcal{R}(c) \subset SO(3)$ is a small finite set of discrete[1] orientations for each class $c \in \mathcal{C}$. For notational convenience, let $i = 1, \ldots, M$ be an enumeration of the set $\{(c, r) \mid c \in \mathcal{C}, r \in \mathcal{R}(c)\}$ and denote the hypotheses by H_i (see Fig. 2).

Offline, a 3-D model database is used to train a viewpoint-pose tree [4] by extracting point clouds from views on a sphere around each model. A set of Fast Point Feature Histograms [13] is extracted from each point cloud and the clouds are arranged in a tree structure according to their feature similarity (see [4] for details). Given a query point cloud, the best-matching cloud from the tree carries information about the class and orientation of the observed object and about the quality of the feature match. Thus, the tree provides an observation $z_t \in \mathcal{Z}$, consisting of the class, orientation, and confidence score of the top match. The model database is used to learn the probability density function (pdf) $q(\cdot \mid x, H_i)$ of z conditioned on any camera viewpoint $x \in \mathcal{X}$ and any hypothesis H_i, $i = 1, \ldots, M$.

Problem. *Given a camera pose $x_0 \in \mathcal{X}$, a prior $p_0 \in [0,1]^M$ on the true hypothesis H_i, and a planning horizon $T < \infty$, choose a sequence of functions $\mu_t : (\mathcal{Z} \times \mathcal{X})^{t+1} \to \mathcal{X}$ for $t = 0, \ldots, T-1$, which minimizes the average movement cost and the probability of an incorrect hypothesis:*

$$\begin{aligned} \min_{\mu_{0:T-1}} \ & \frac{1}{T}\sum_{t=1}^{T} g(x_{t-1}, x_t) + \lambda Pe(T) \\ \text{s.t. } & x_{t+1} = \mu_t(z_{0:t}, x_{0:t}), \quad t = 0, \ldots, T-1, \\ & x_{t+1} \notin \{x_0, \ldots, x_t\}, \quad t = 0, \ldots, T-1, \\ & z_t \sim q(\cdot \mid x_t, H_i), \quad t = 0, \ldots, T, \\ & p_t = b(p_{t-1}, z_t, x_t), \quad t = 1, \ldots, T, \end{aligned} \tag{1}$$

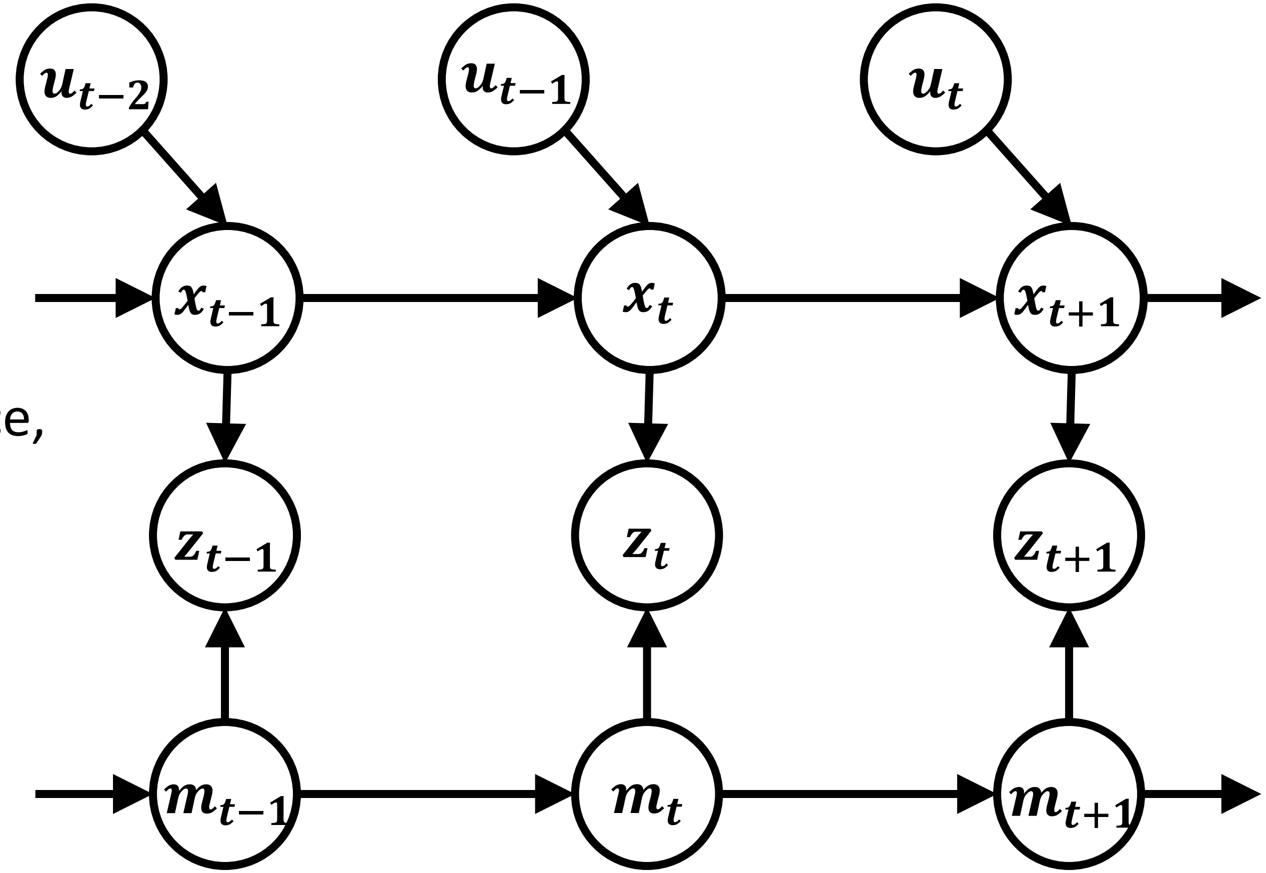
[1] After a hypothesis is chosen, the discrete orientation estimate can be refined by aligning the observed object surface to the corresponding model in the training database, e.g., by using the iterative closest point algorithm.

Tentative Schedule

Date	Lecture	Material	Assignment
Jan 06	Introduction	Matrix-calculus	
Jan 08	Unconstrained Optimization	Barfoot-Ch.4.3.1	
Jan 13	Catch up		
Jan 15	Rotations	Barfoot-Ch.6.1-6.3	HW1, PR1
Jan 20	Catch up		
Jan 22	Robot Motion and Observation Models	Barfoot-Ch.6.4	
Jan 27	Catch up		
Jan 29	Factor Graph SLAM	Dellaert-Kaess Ch.1, Ch.2	
Feb 03	Localization and Odometry from Point Features		
Feb 05	Catch up		HW2, PR2
Feb 10	Bayes Filter	Barfoot-Ch.4.2	
Feb 12	Particle Filter SLAM	Thrun-Burgard-Fox Ch.7, Ch.8, Ch.9	
Feb 17	Catch up		
Feb 19	Kalman Filter	Barfoot-Ch.3.3, Sarkka-Ch4	
Feb 24	Catch up		
Feb 26	EKF, UKF	Barfoot-Ch.4.2, Sarkka-Ch5	HW3, PR3
Mar 03	Matrix Lie Groups	Barfoot-Ch.7.1-7.2, Boumal-Ch.3	
Mar 05	Visual-Inertial SLAM		
Mar 10	Catch up		
Mar 12	Visual Features	Image-Features	
Mar 17	Final Exam		

Structure of Robotics Problems

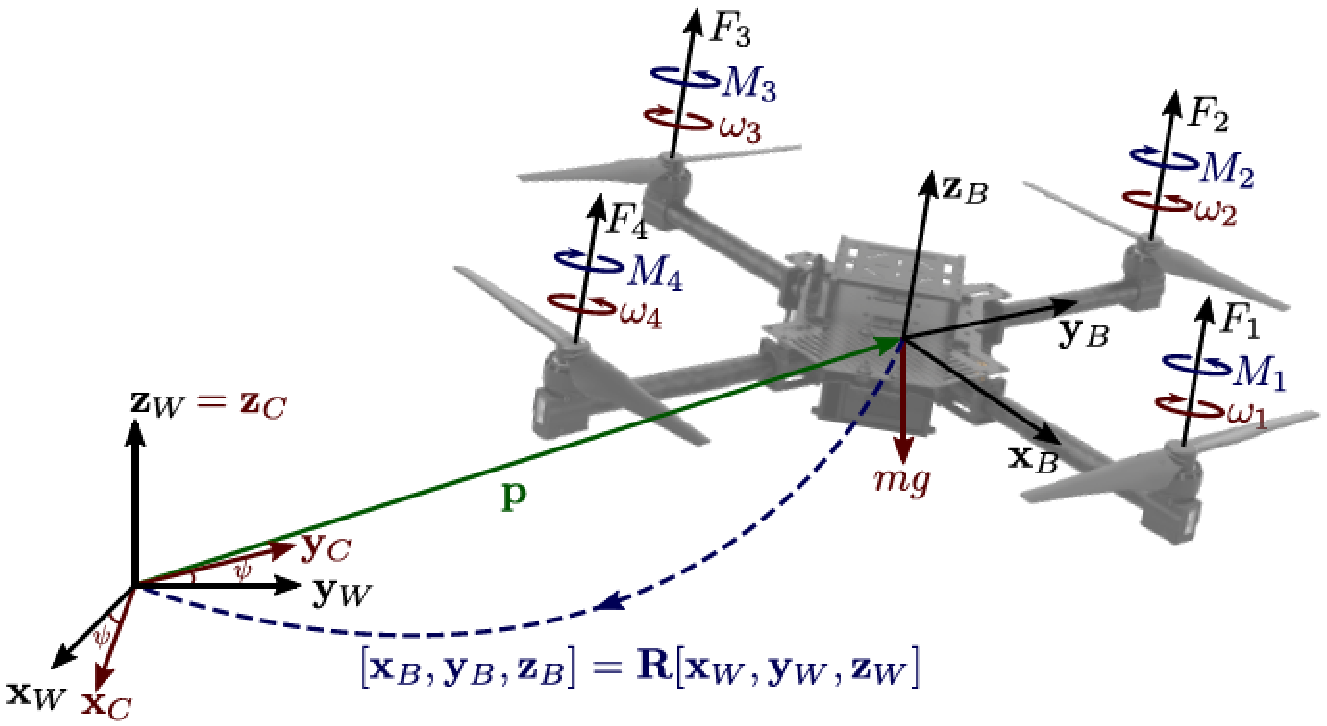
- **Time:** t (discrete or continuous)
- **Robot state:** x_t (e.g., position, orientation, velocity)
- **Environment state:** m_t (e.g., map of free space, locations of objects)
- **Control input:** u_t (e.g., force and torque)
- **Observation:** z_t (e.g., image, laser scan, radio signal, inertial measurement)

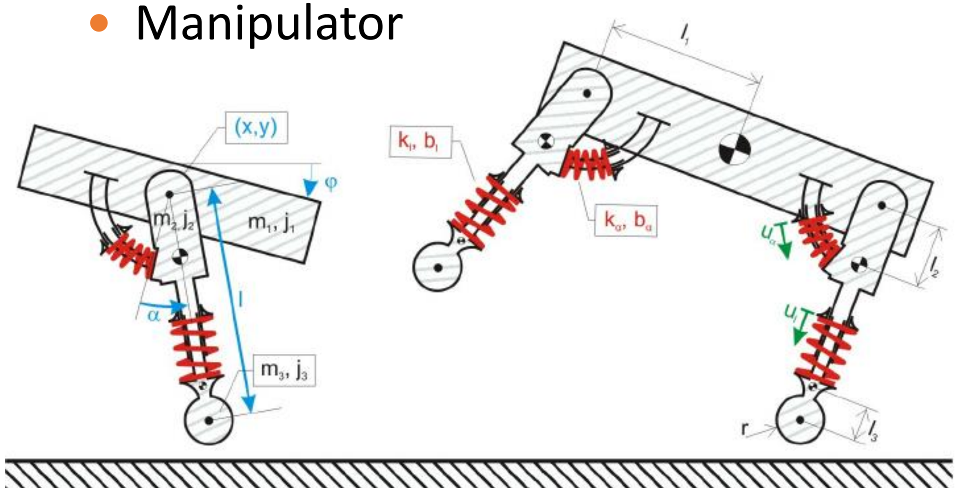


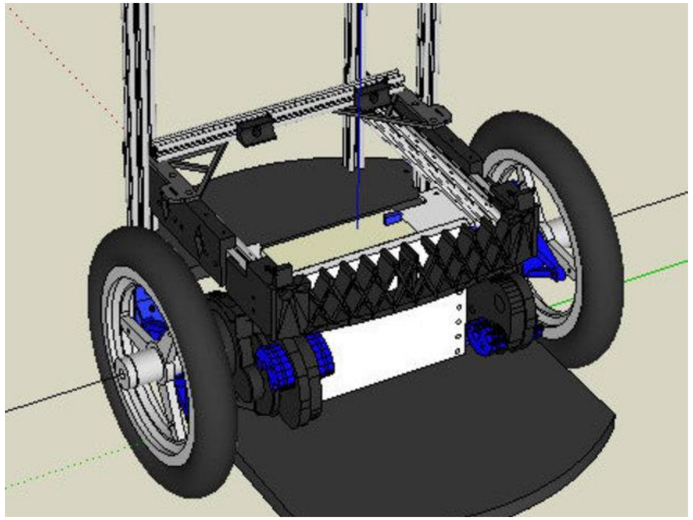
- **Motion Model:** $p(x_{t+1}|x_t, u_t)$ --- describes the motion of the robot to a new state x_{t+1} after applying control input u_t at state x_t
- **Observation Model:** $p(z_t|x_t, m_t)$ --- describes the observation z_t of the robot depending on its state x_t and the map m_t of the environment

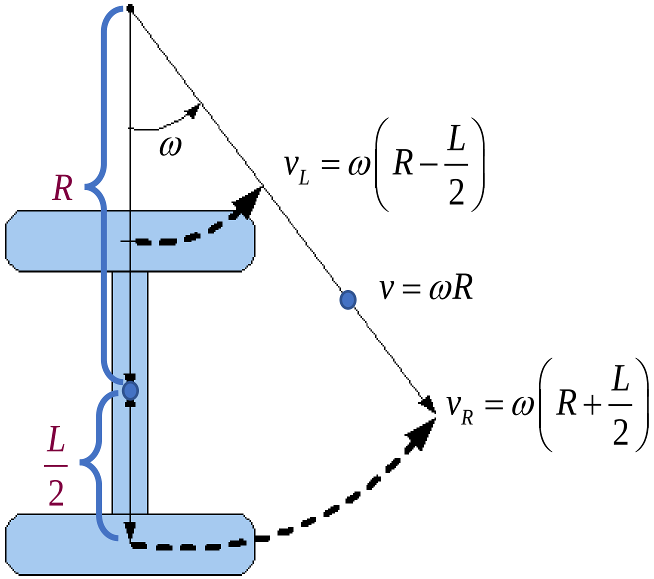
Motion Models

- A motion model describe the kinematics or dynamics of the robot state $\boldsymbol{x_t}$
- Wheeled robots:
  - Differential drive (roomba)
  - Ackermann drive (car, bicycle)
- Aerial robots:
  - Fixed-wing aerial vehicle
  - Quadrotor aerial vehicle
- Legged and humanoid robots:
  - Quadruped
  - Manipulator









Observation Models

- **Position Sensor:** directly measures position (e.g., GPS, laser scanner, IR sensor, RGBD camera)
- **Velocity/Acceleration/Force Sensor**: measures linear acceleration or angular velocity or pressure or force (accelerometer, gyroscope, inertial measurement unit (IMU), tactile sensor)
- **Bearing Sensor**: measures angles (e.g., magnetometer, camera, microphone)
- **Range Sensor**: measures distances (e.g., radio)

FLIR RGB Camera


VectorNav IMU

Ublox GPS and Compass


Beaglebone Radio


Intel RealSense RGBD Camera


Garmin Single-beam Lidar

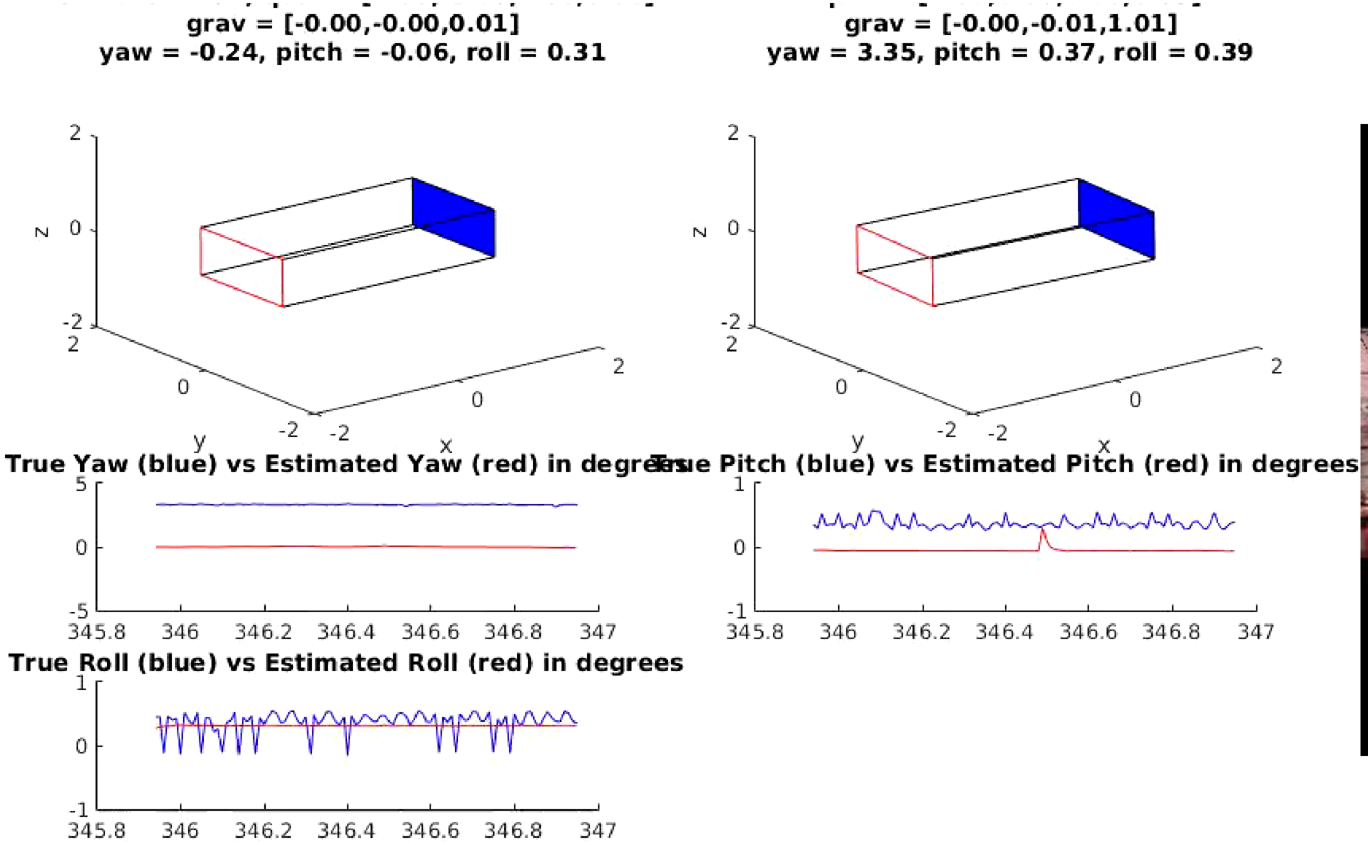

Hokuyo 2D Lidar

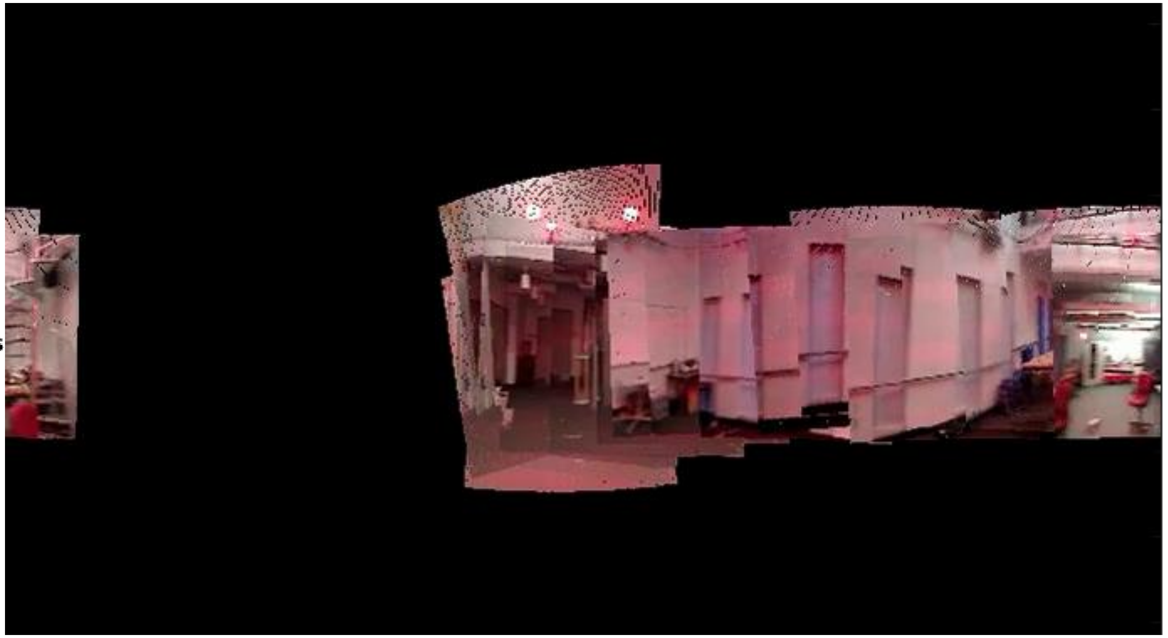
Velodyne 3D Lidar

Hex 6-Axis Force/Torque Sensor

Project 1: Orientation Tracking

- Use gradient descent to track the 3D orientation of a rotating body using IMU measurements and construct a panorama using RGB images

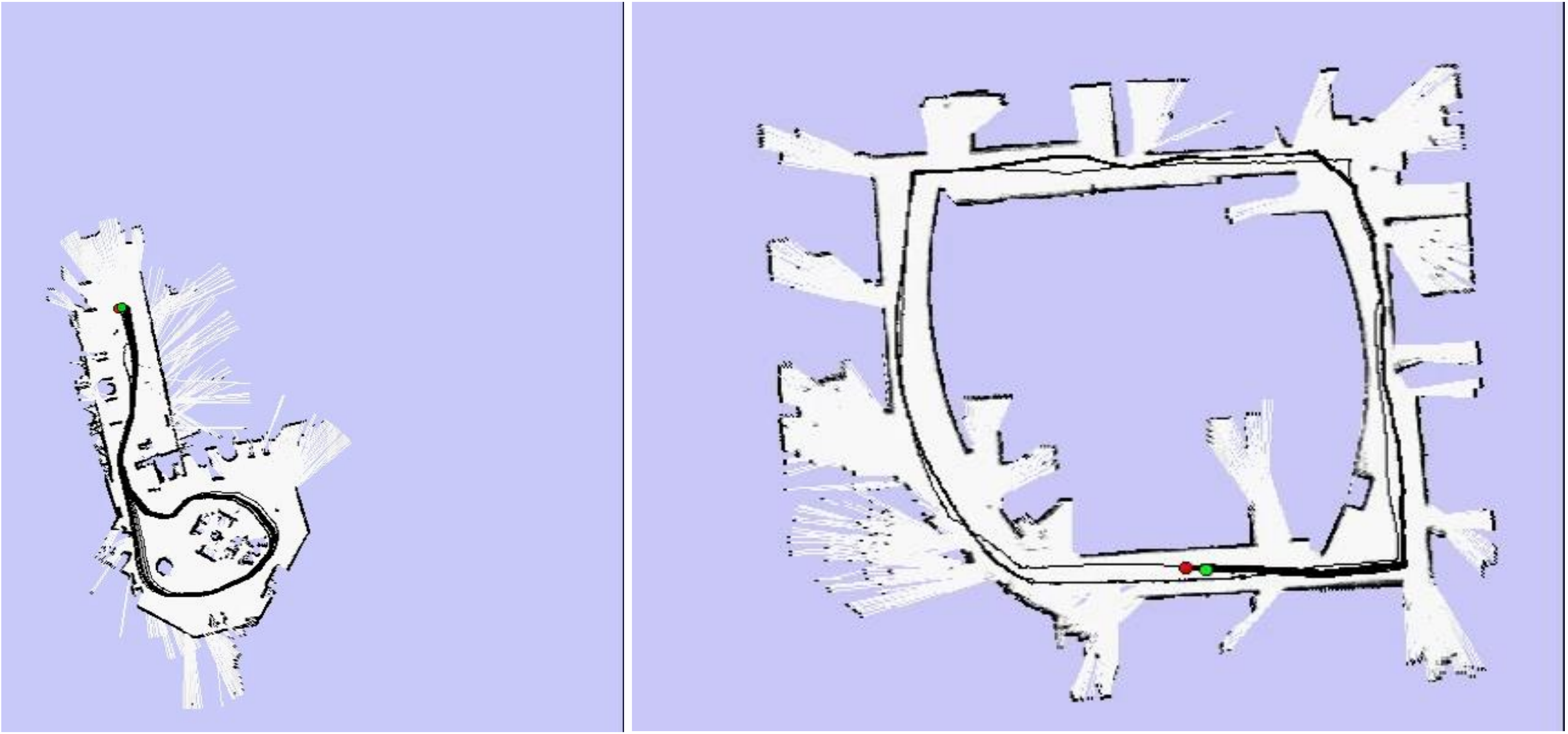




Project 2: Particle Filter SLAM

Montemerlo et al., FastSLAM, AAAI'02

- Simultaneous localization and mapping (SLAM) using a lidar scanner



Project 3: Visual Inertial SLAM

Mur-Artal et al., OrbSLAM, IEEE T-RO'15

- Kalman filter tracking of the 3D pose of a moving robot based on IMU and camera measurements