CSE P590a
Robotics

Recap and Discussion

Goal of this course

- Provide an overview of fundamental problems / techniques in robotics

- Understanding of estimation and decision making in dynamical systems
  - Probabilistic modeling and filtering
  - Deterministic and non-deterministic planning
  - Learning for perception and modeling

- Augment model-based understanding with hands-on experience in deep learning

Bayesian Filtering, Models

ESTIMATION

Bayes Filters

z = observation
u = action
x = state

$$\boxed{Bel(x_t)} = P(x_t \mid u_1, z_1 \ldots, u_t, z_t)$$

Bayes
$$= \eta \; P(z_t \mid x_t, u_1, z_1, \ldots, u_t) \; P(x_t \mid u_1, z_1, \ldots, u_t)$$

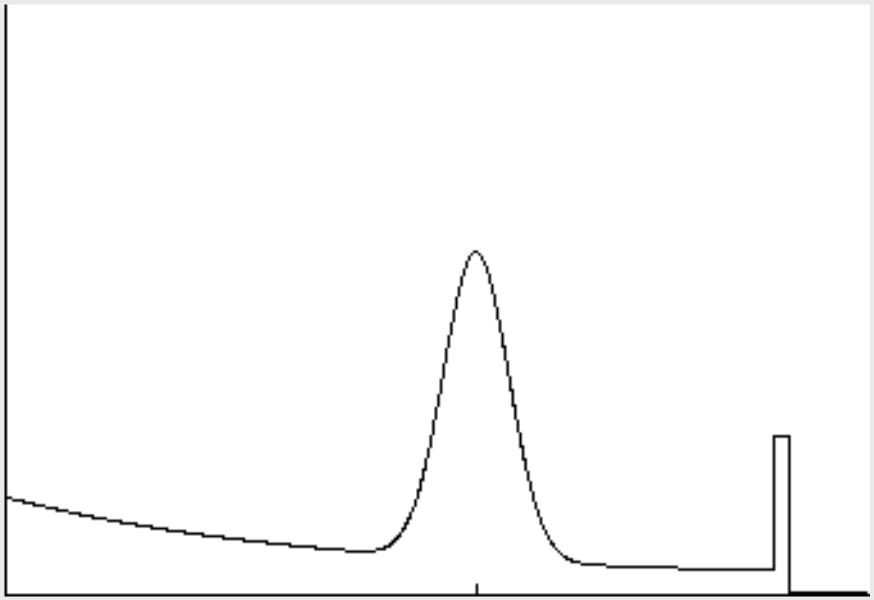
Markov
$$= \eta \; P(z_t \mid x_t) \; P(x_t \mid u_1, z_1, \ldots, u_t)$$

Total prob.
$$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1}) \; P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \; dx_{t-1}$$

Markov
$$= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; P(x_{t-1} \mid u_1, z_1, \ldots, u_t) \; dx_{t-1}$$

$$= \boxed{\eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1}) \; Bel(x_{t-1}) \; dx_{t-1}}$$

Parametric Sensor Model



$$P(z \mid x,m) = \begin{pmatrix} \alpha_{\text{hit}} \\ \alpha_{\text{unexp}} \\ \alpha_{\text{max}} \\ \alpha_{\text{rand}} \end{pmatrix}^T \cdot \begin{pmatrix} P_{\text{hit}}(z \mid x,m) \\ P_{\text{unexp}}(z \mid x,m) \\ P_{\text{max}}(z \mid x,m) \\ P_{\text{rand}}(z \mid x,m) \end{pmatrix}$$

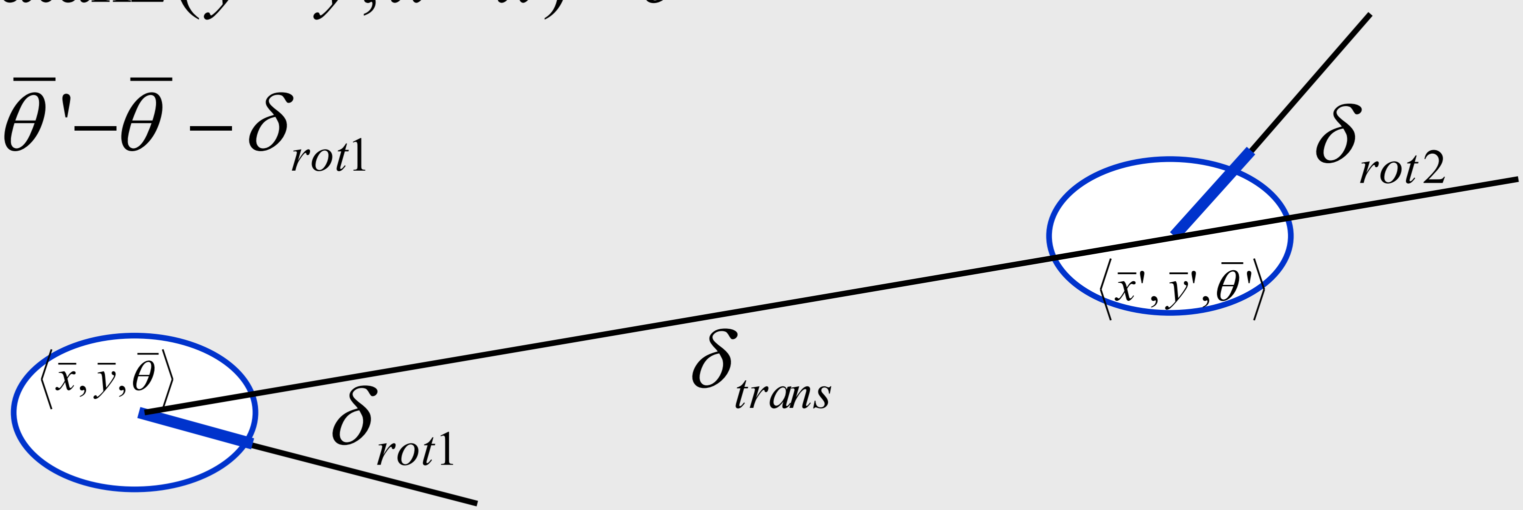
Parametric Kinematics Model

- **Robot moves from** $\langle \bar{x}, \bar{y}, \bar{\theta} \rangle$ **to** $\langle \bar{x}', \bar{y}', \bar{\theta}' \rangle$**.**
- **Odometry information** $u = \langle \delta_{rot1}, \delta_{rot2}, \delta_{trans} \rangle$**.**

$$\delta_{trans} = \sqrt{(\bar{x}' - \bar{x})^2 + (\bar{y}' - \bar{y})^2}$$

$$\delta_{rot1} = \operatorname{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$

$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$$



The Prediction-Correction-Cycle of Kalman Filters

Prediction

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

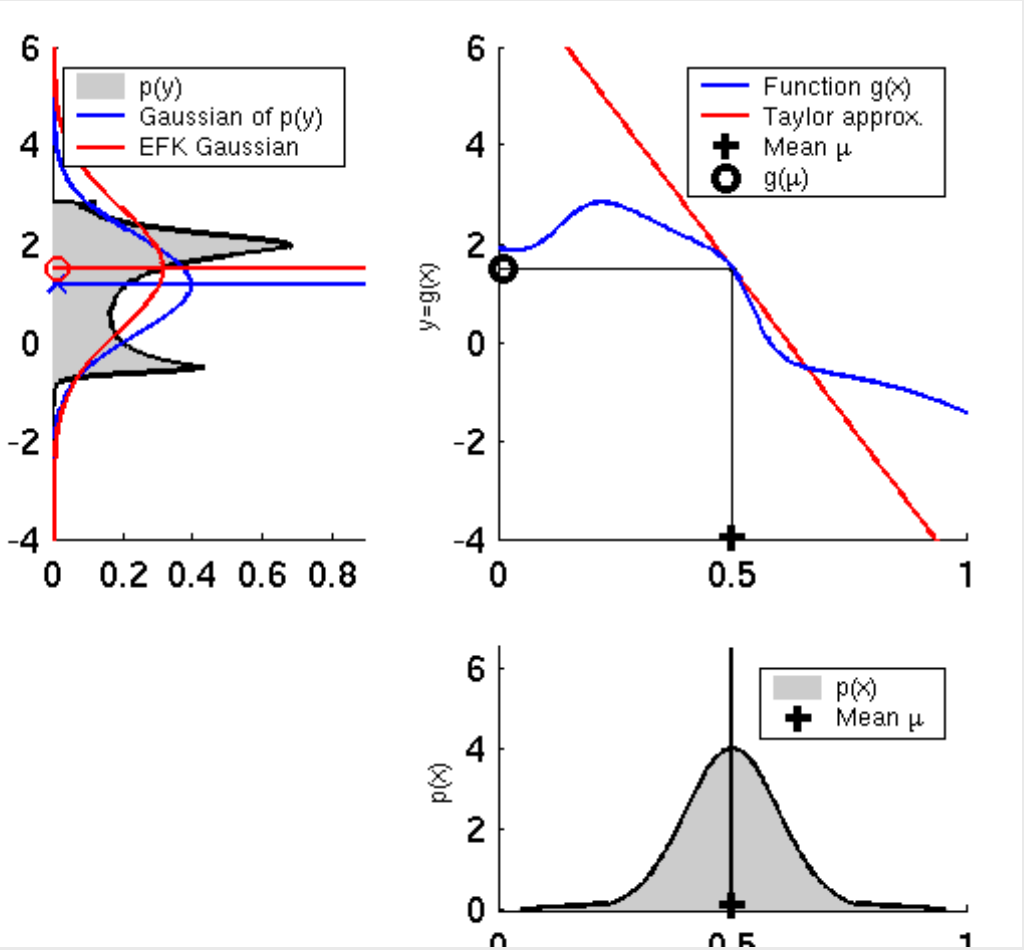
$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$

Correction

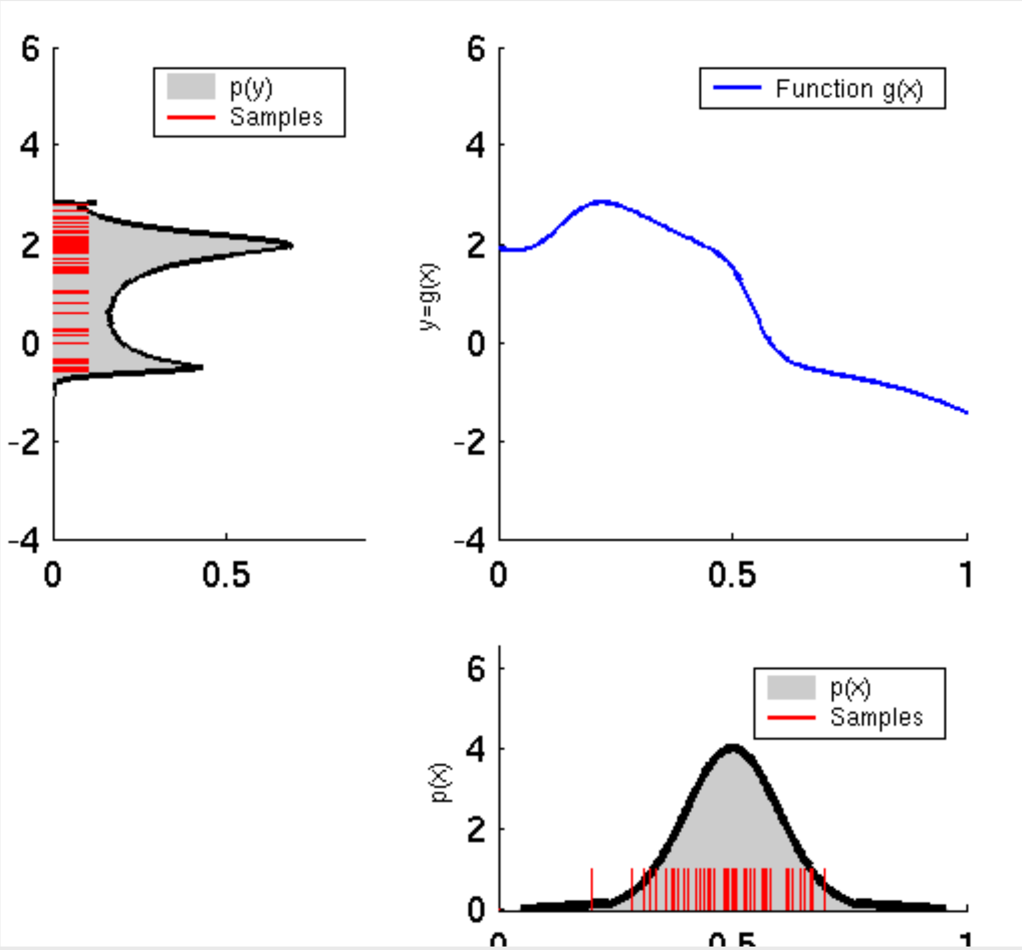
$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t (z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t) \bar{\sigma}_t^2 \end{cases}, K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t (z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t) \bar{\Sigma}_t \end{cases}, K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

EKF Linearization



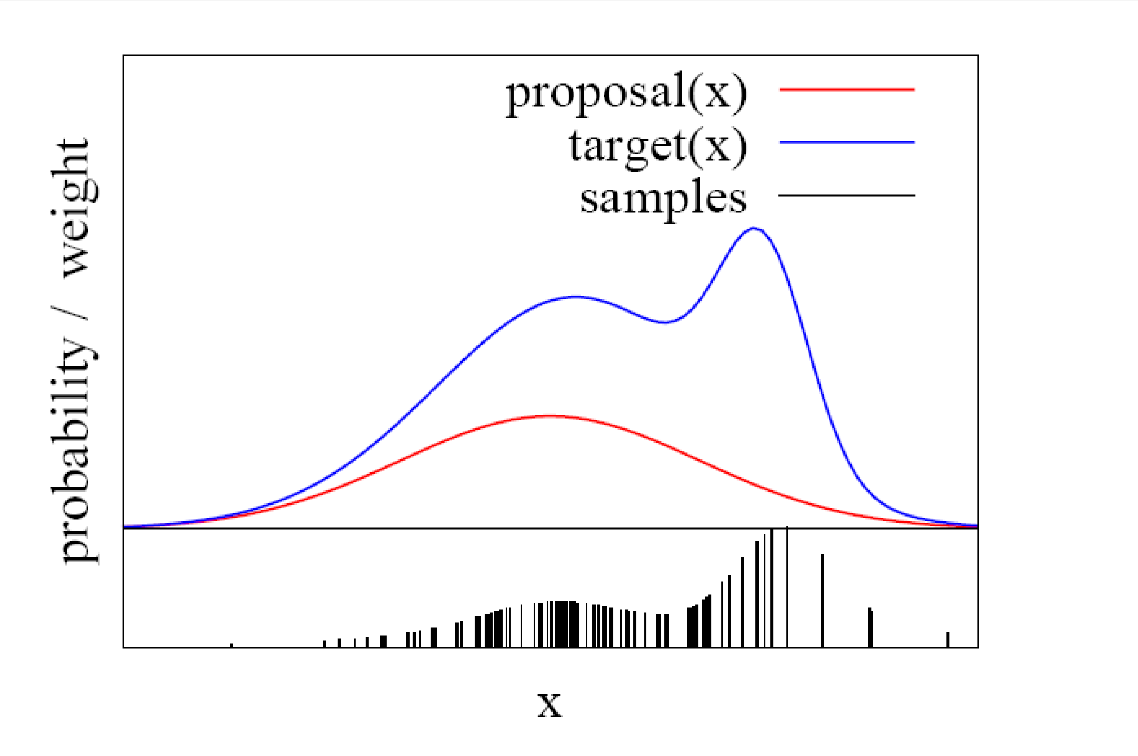
Particle Filter Projection



Importance Sampling Principle

- We can use a different distribution g to generate samples from f
- By introducing an importance weight w, we can account for the "differences between g and f"

$$w = f / g$$

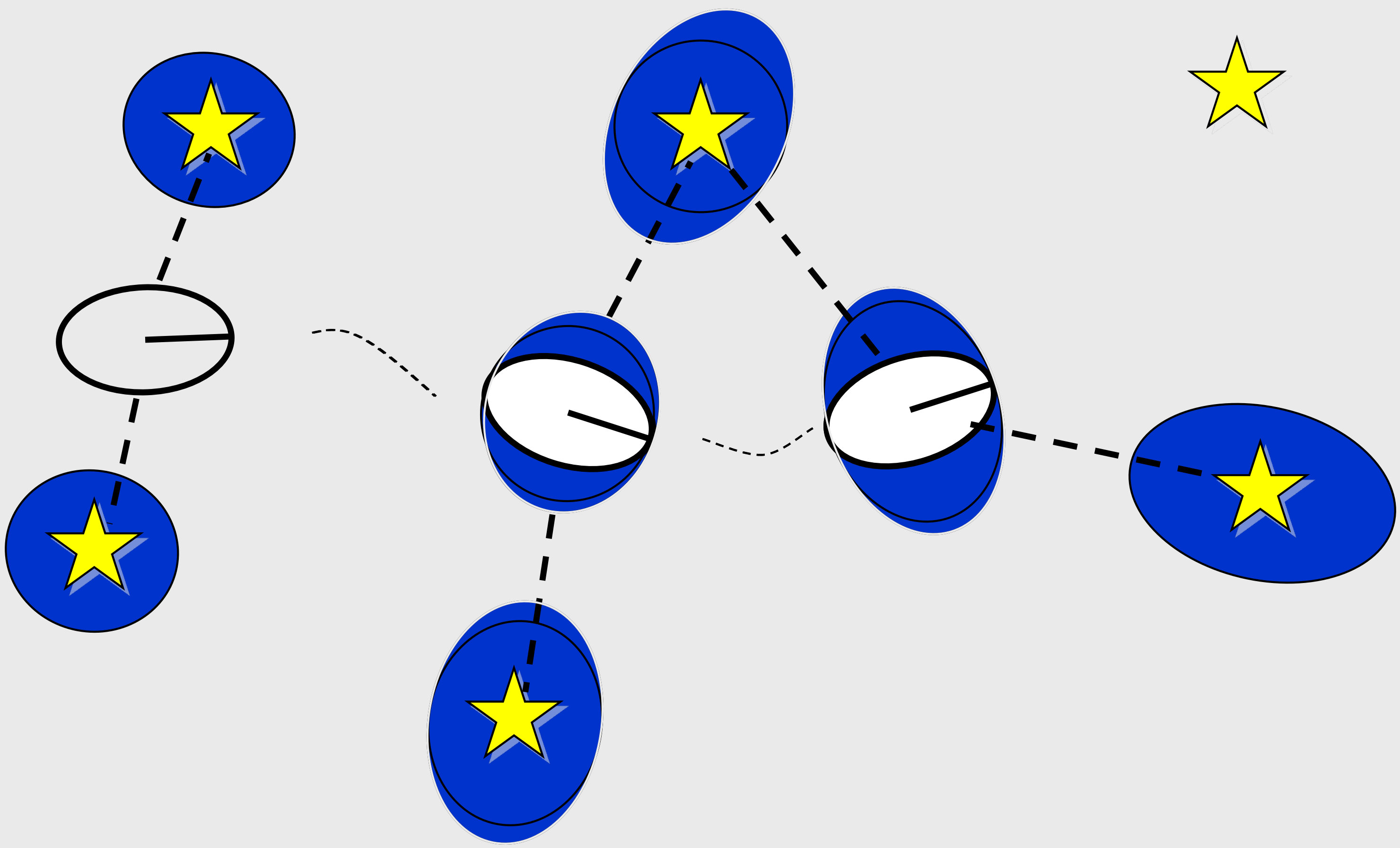


SLAM

ESTIMATION

Why is SLAM a hard problem?

- SLAM: robot path and map are both unknown



- Robot path error correlates errors in the map

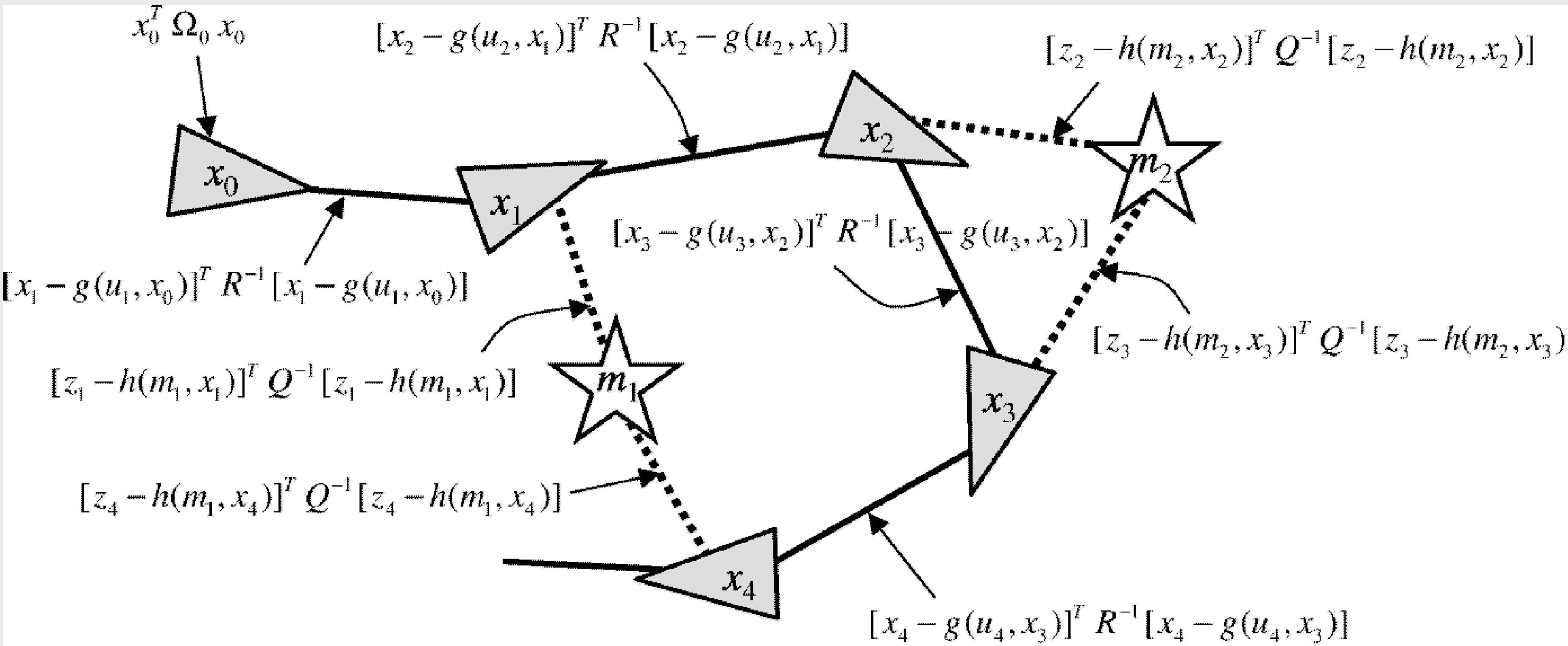
EKF-SLAM

- Map with N landmarks:(3+2N)-dimensional Gaussian

$$Bel(x_t, m_t) = \left\langle \begin{pmatrix} x \\ y \\ \theta \\ l_1 \\ l_2 \\ \vdots \\ l_N \end{pmatrix}, \begin{pmatrix} \sigma_x^2 & \sigma_{xy} & \sigma_{x\theta} & \sigma_{xl_1} & \sigma_{xl_2} & \cdots & \sigma_{xl_N} \\ \sigma_{xy} & \sigma_y^2 & \sigma_{y\theta} & \sigma_{yl_1} & \sigma_{yl_2} & \cdots & \sigma_{yl_N} \\ \sigma_{x\theta} & \sigma_{y\theta} & \sigma_\theta^2 & \sigma_{\theta l_1} & \sigma_{\theta l_2} & \cdots & \sigma_{\theta l_N} \\ \sigma_{xl_1} & \sigma_{yl_1} & \sigma_{\theta l_1} & \sigma_{l_1}^2 & \sigma_{l_1 l_2} & \cdots & \sigma_{l_1 l_N} \\ \sigma_{xl_2} & \sigma_{yl_2} & \sigma_{\theta l_2} & \sigma_{l_1 l_2} & \sigma_{l_2}^2 & \cdots & \sigma_{l_2 l_N} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \sigma_{xl_N} & \sigma_{yl_N} & \sigma_{\theta l_N} & \sigma_{l_1 l_N} & \sigma_{l_2 l_N} & \cdots & \sigma_{l_N}^2 \end{pmatrix} \right\rangle$$

- Can handle hundreds of dimensions

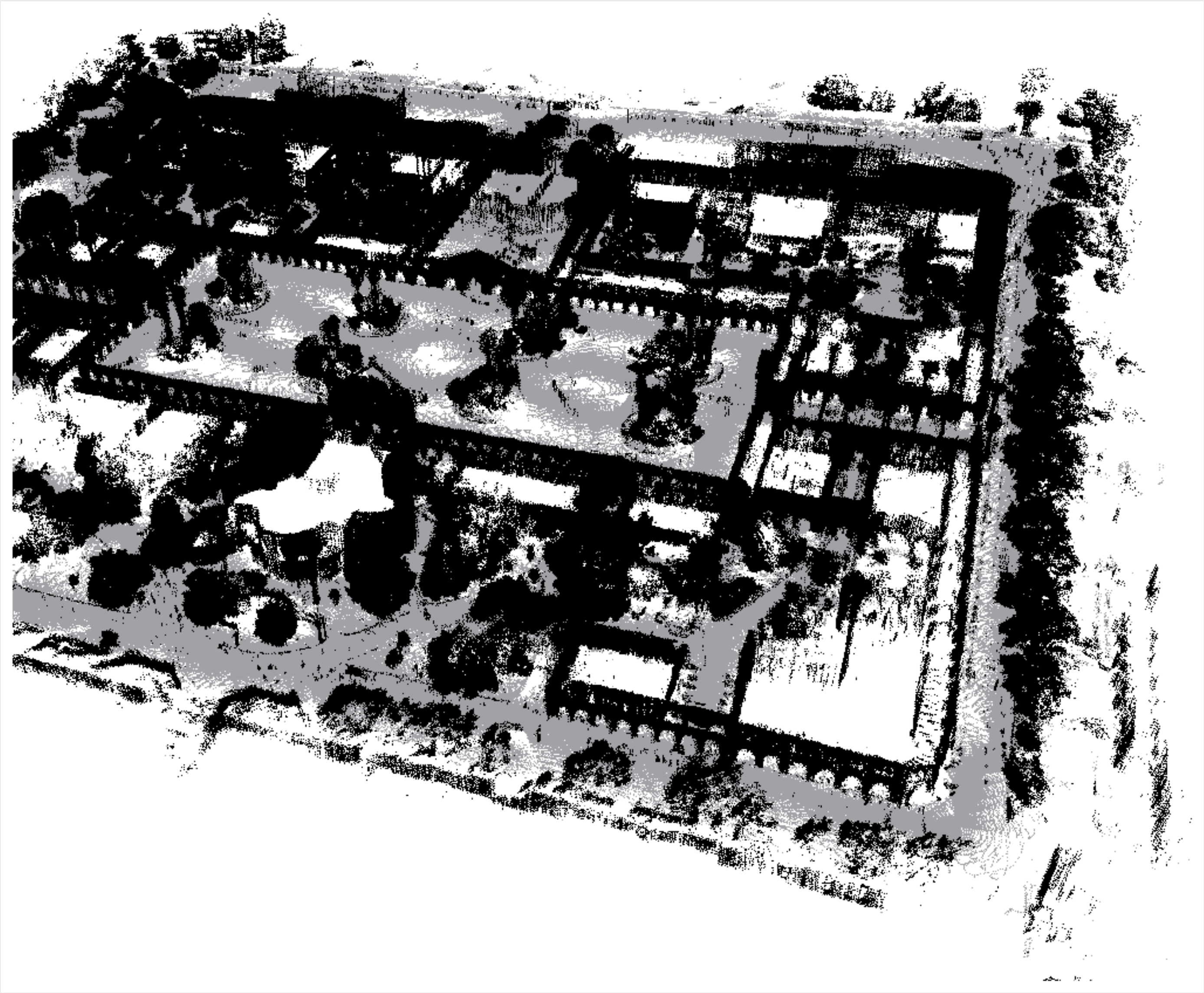
Graph-SLAM Idea



Sum of all constraints:

$$J_{\text{GraphSLAM}} = x_0^T \Omega_0 x_0 + \sum_t [x_t - g(u_t, x_{t-1})]^T R^{-1} [x_t - g(u_t, x_{t-1})] + \sum_t [z_t - h(m_{c_t}, x_t)]^T Q^{-1} [z_t - h(m_{c_t}, x_t)]$$

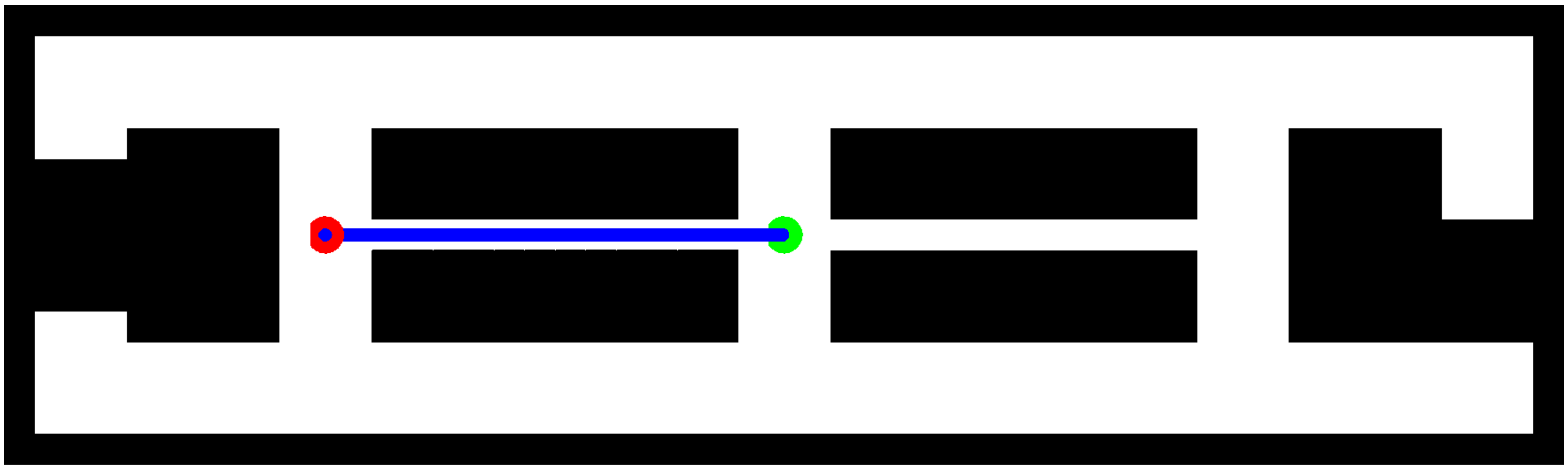
3D Outdoor Mapping



10^8 features, 10^5 poses, only few secs using cg.

PLANNING / CONTROL

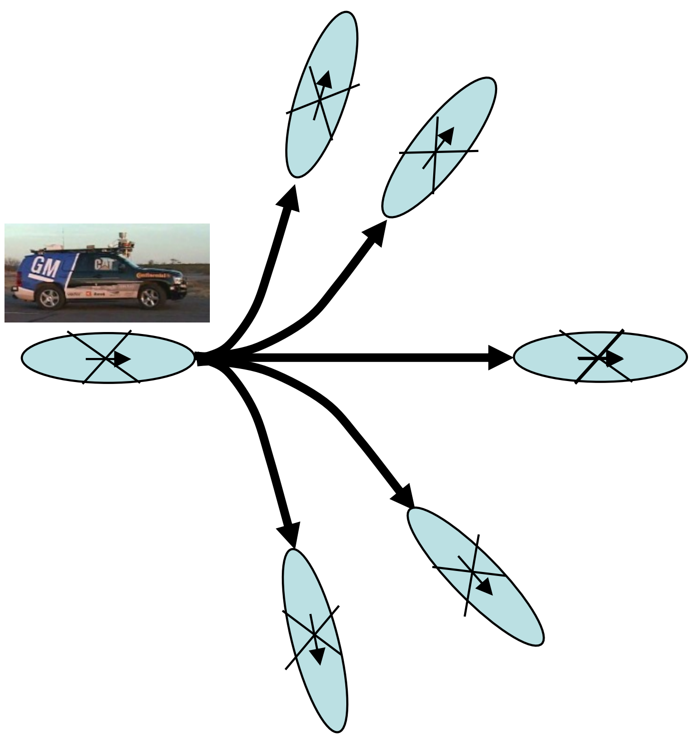
Deterministic, fully observable



Planning via Cell Decomposition

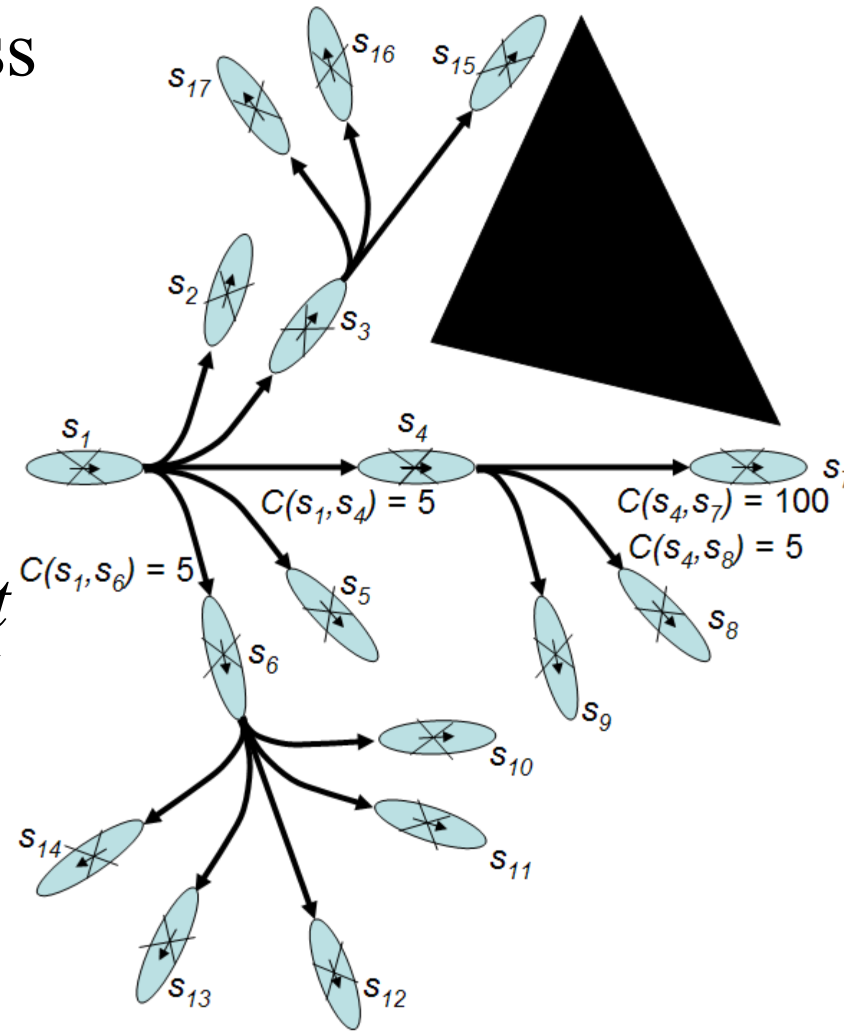
- Graph construction:
  - lattice graph
  - pros: sparse graph, feasible paths
  - cons: possible incompleteness

action template

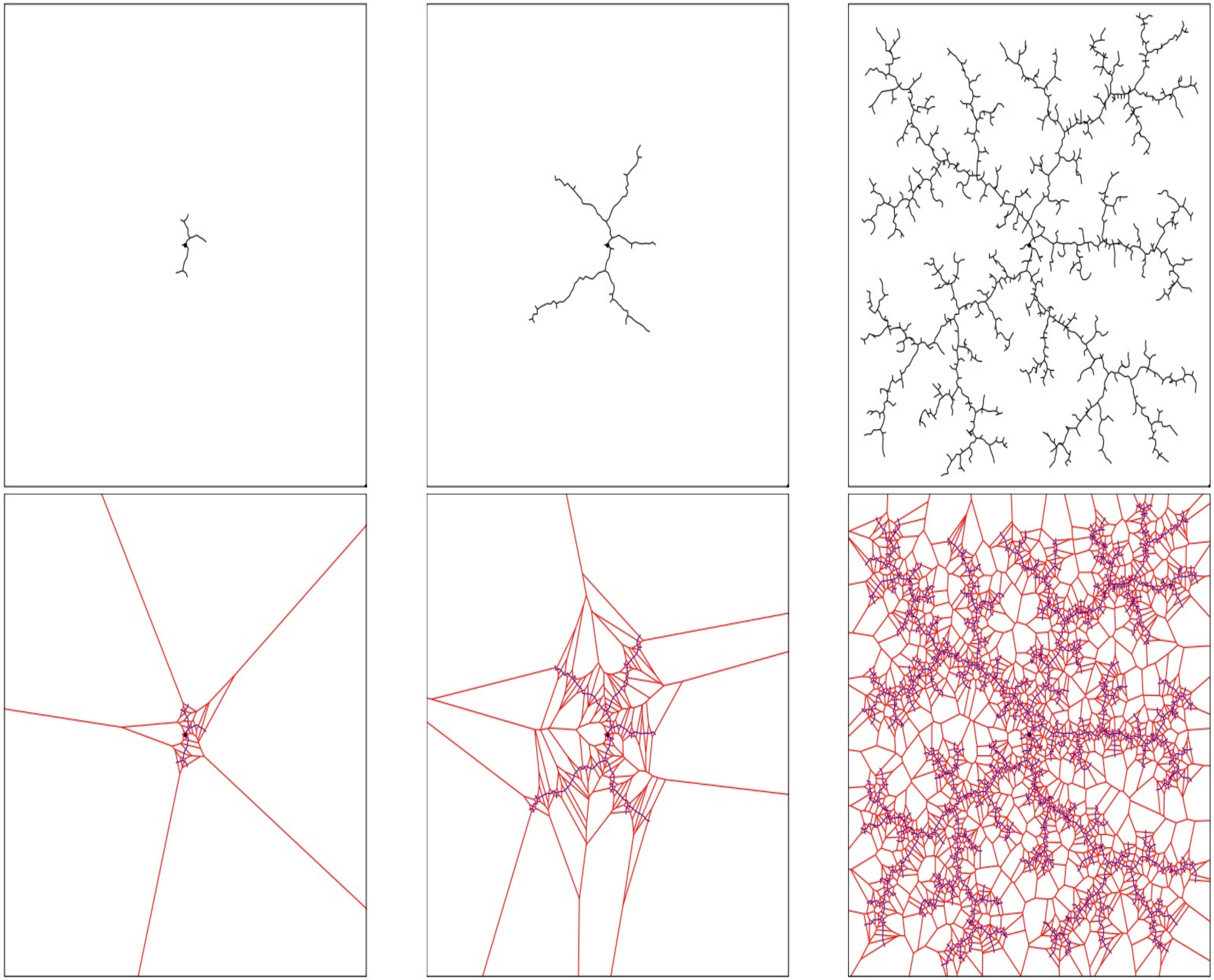


replicate it

online

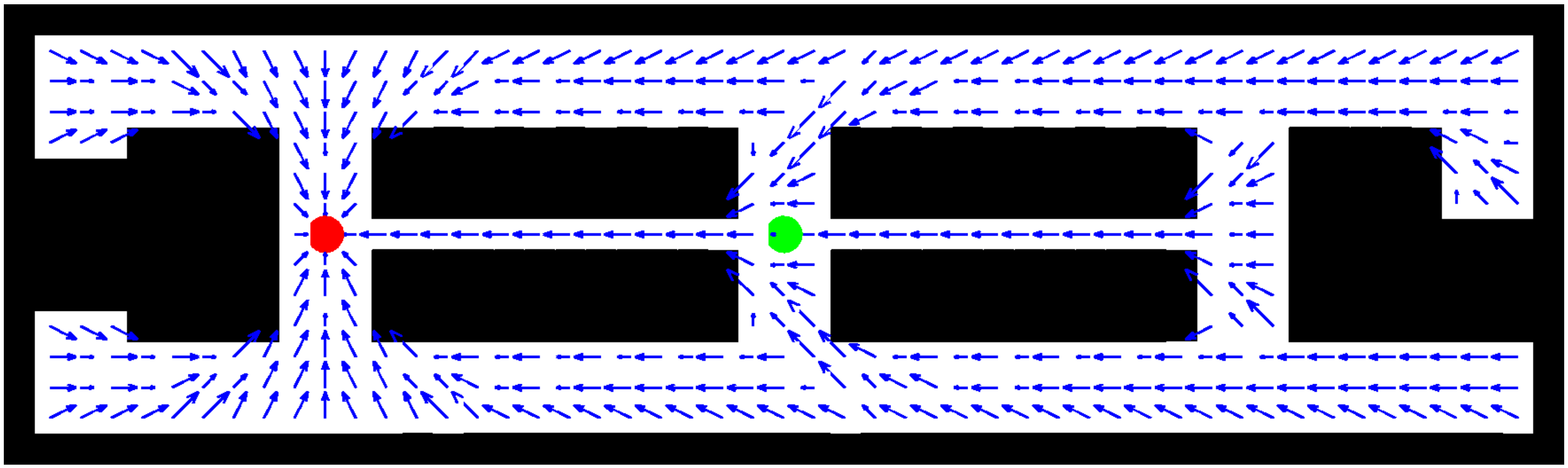


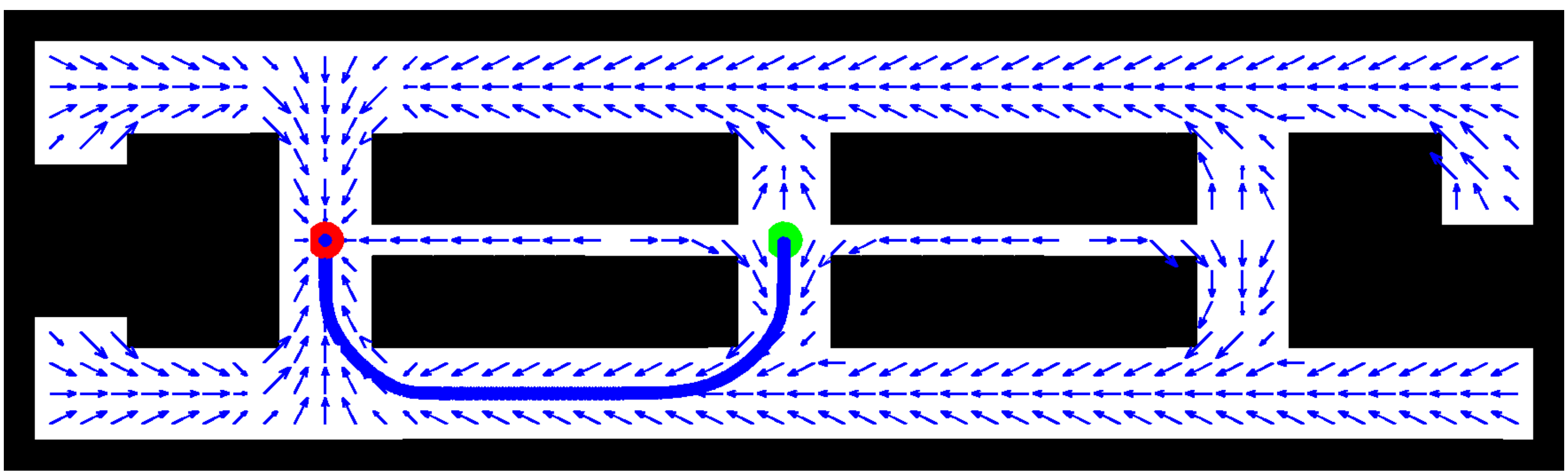
Rapidly exploring Random Tree (RRT)



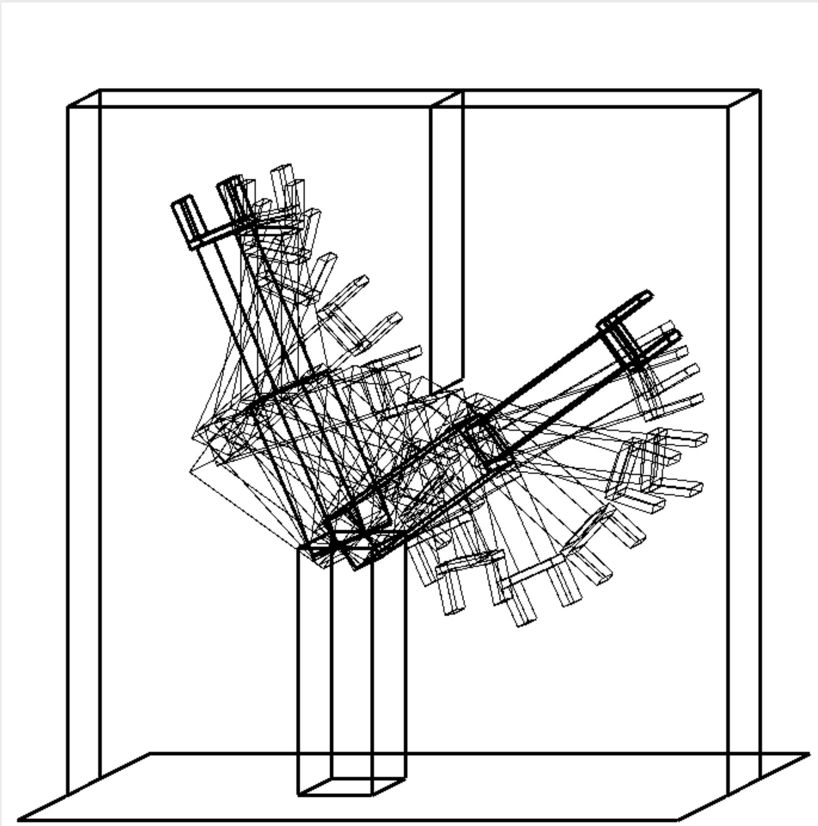
Source: LaValle and Kuffner 01

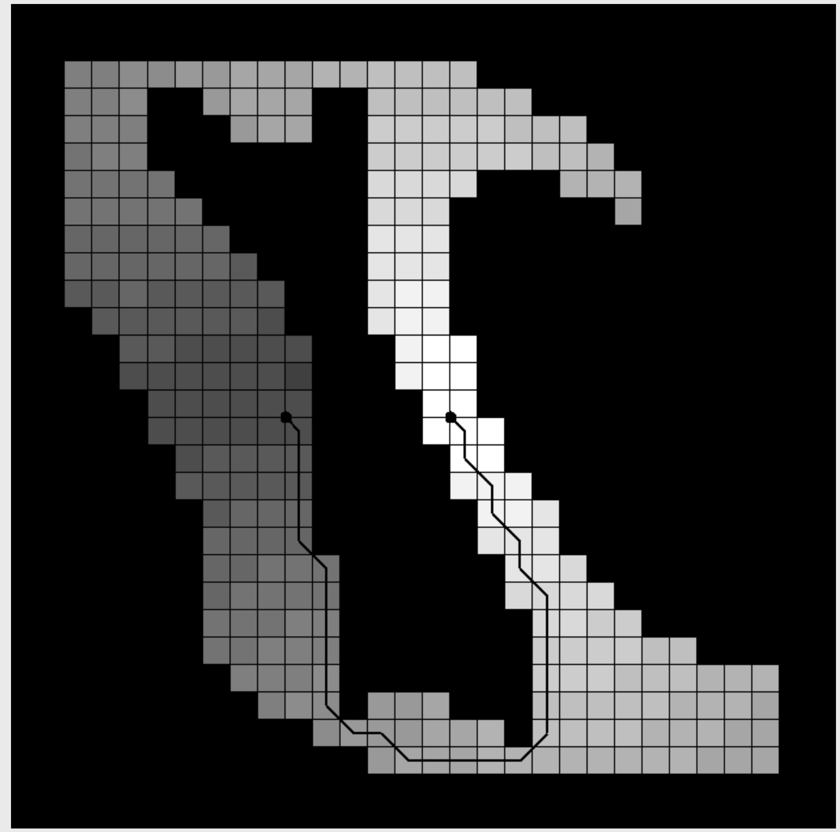
Stochastic, Fully Observable





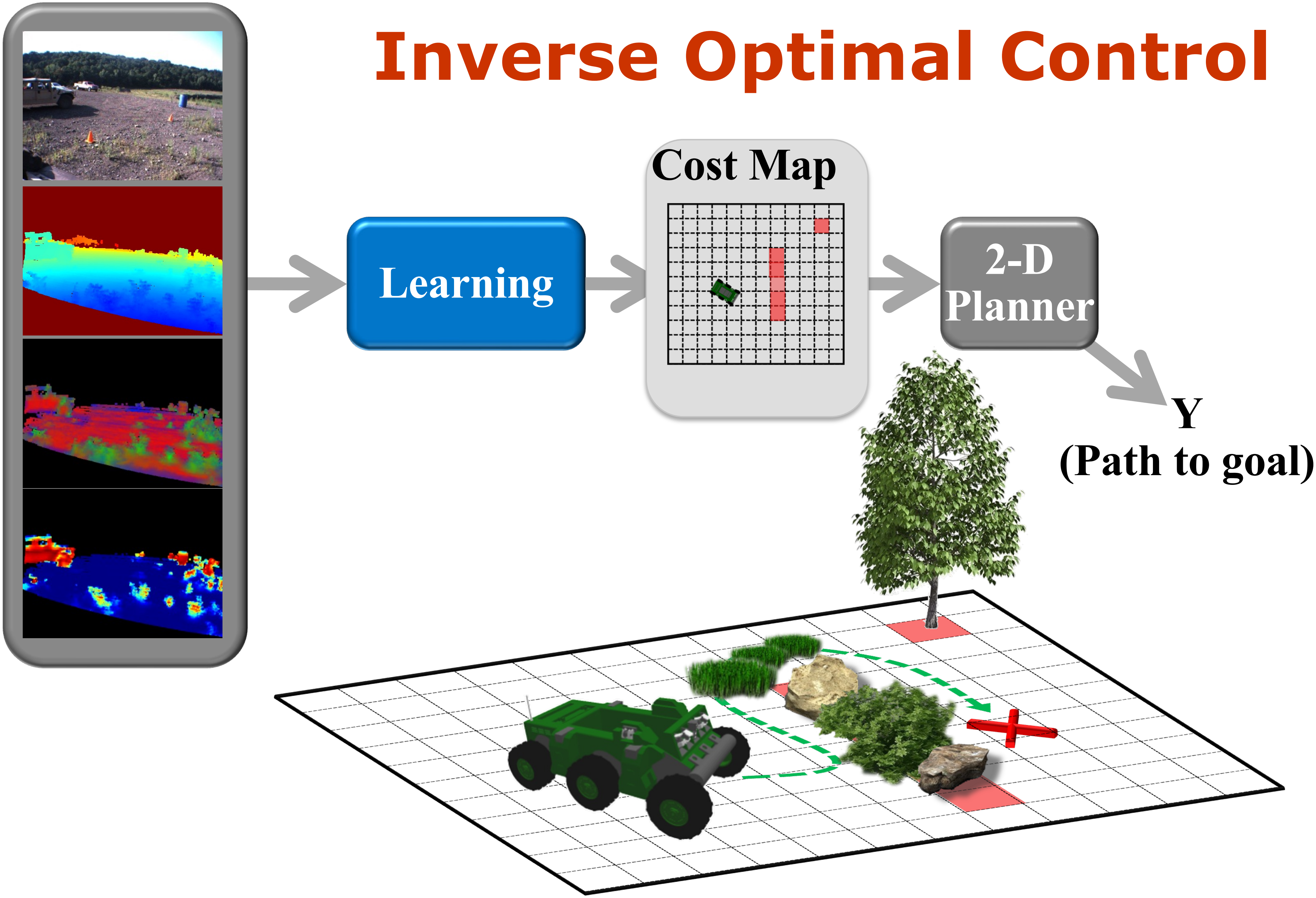
Manipulator Control Path





Work space

Configuration space


Inverse Optimal Control
Learning
Cost Map
2-D
Planner
Y
(Path to goal)

Beyond Model-Based Reasoning
Cooking with Julia

Gravity and Onions

Intuitive Physics

- People have intuitive understanding of how things evolve over time, and how to achieve desired change
- Qualitatively related to physics underlying a scene: gravity, forces, friction, mass, size, persistence, rigid and non-rigid motion, …
- Good enough for control since tightly coupled to perception --> closed loop control

- Physics based models in robotics generalize well but are not tightly coupled to perception

- Can we learn intuitive physics models for robots?
  - Ideally suited for closed-loop control since fully grounded in perceptual experience
  - Applicable across a wide range of tasks

Deep learning for robotics

- Extremely flexible and expressive framework for learning from raw data
  - Will dominate many recognition / control tasks, especially well suited for closed-loop control with complex perception and state spaces
  - In robotics, future data provides supervisory signals
- Challenges
  - How to get training data (scalability, safety, overfitting, simulation)?
  - How to best combine models and deep learning?
  - How to extract / model uncertainty and guarantees?
  - Understanding of network structures, training regimes, generalization capabilities
- Risks
  - Students degraded to network and data engineers
  - Company or lab with most GPU’s wins
- A toolbox to try new things and revisit tasks from new perspectives