History and Current Challenges in the field of Robotics

Christian Siagian
(adapted from lectures by
Dr. Maja Mataric &
Dr. Gaurav Sukhatme)
Univ. Southern California, CA, USA

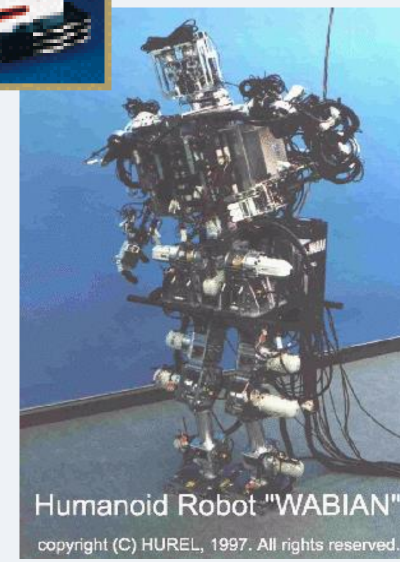
Outline

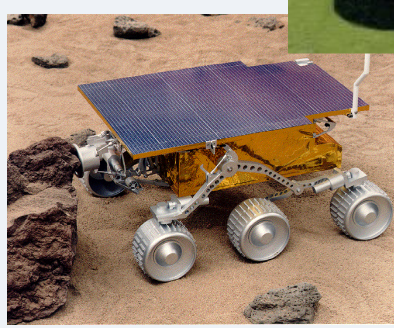
- Definition of a robot
- History of robotics
- Sub-field in Robotics

Defining “robot”

- What makes a robot
- Sensors, sensor space
- State, state space
- Action/behavior, effectors, action space
- The spectrum of control

Robots Today

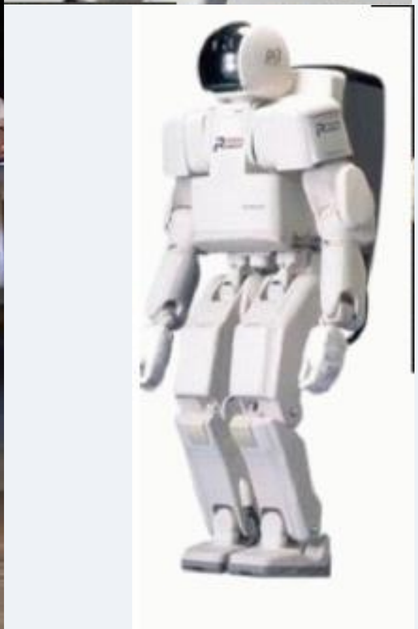




iLab

USC
UNIVERSITY OF SOUTHERN CALIFORNIA

Robots Today: Anthropomorphic



Robots today: Animal-like robots

Why "robot"?

- The term "robot" comes from Karel Capek's 1921 play RUR (Rossum's Universal Robots)
- "robotics" first introduced by Isaac Asimov in his science fiction writing
- It is most likely a combination of "rabota" (obligatory work) and "robotnik" (serf).
- The kind of robotics we will talk about will move far beyond such "obligatory work."

Alternative terms

- UAV: Unmanned Aerial Vehicle
- UGV: Unmanned Ground Vehicle
- UUV: Unmanned Undersea (underwater) Vehicle
- AUV: Autonomous Underwater Vehicle

What is a Robot?

- A robot is a system which exists in the *physical* world and *autonomously* *senses* its environment and *acts* in it to achieve some goals.

Other Definitions

- A robot is a re-programmable, multi-functional, manipulator designed to move material, parts, or specialized devices though variable programmed motions for the performance of a task (Robotics Industry Association)

- Robotics is the intelligent connection of perception to action (M. Brady)

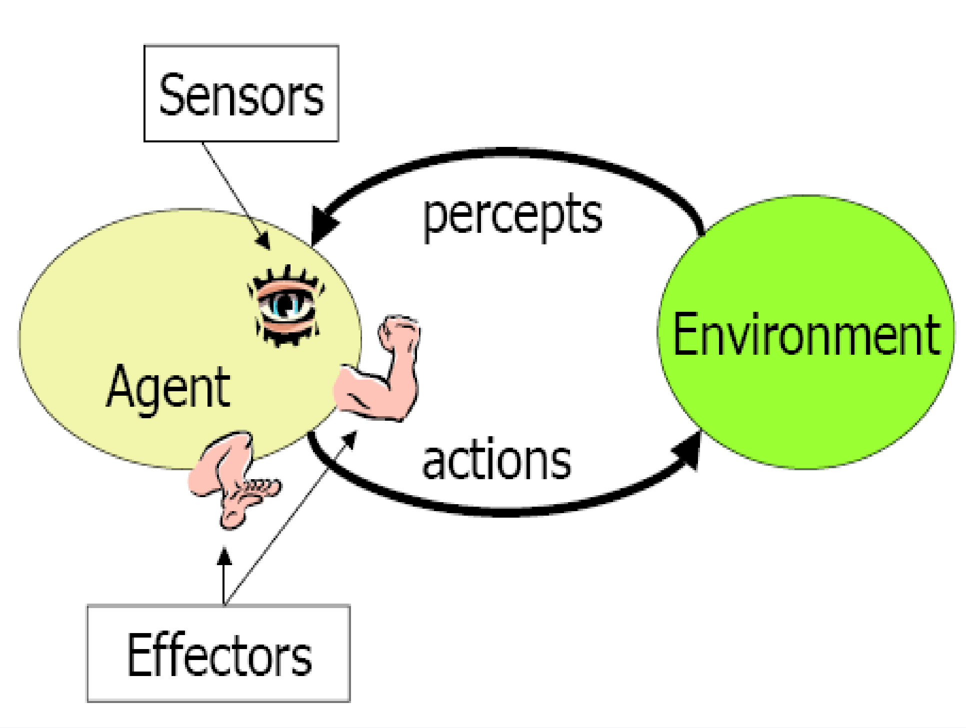
What Makes a Robot?

A robot consists of:

- sensors
- effectors/actuators
- (communication)
- controller

A robot is capable of:

- acting autonomously
- achieving goals



Sensors: what can be sensed?

- Depends on the sensors on the robot
- The robot exists in its *sensor space*: all possible values of sensory readings
- Also called *perceptual space*
- Robot sensors are very different from biological ones
- A roboticist has to try to imagine the world in the robot's sensor space

State: what can be known?

- A sufficient description of the system
- can be:
  - *Observable*: robot always knows its state
  - *Hidden/inaccessible/unobservable*: robot never knows its state
  - *Partially observable*: the robot knows a part of its state
  - *Discrete* (e.g., up, down, blue, red)
  - *Continuous* (e.g., 3.765 mph)

Types of State

- *External state*: state of the world
  - Sensed using the robot's sensors
  - E.g.: night, day, at-home, sleeping, sunny
- *Internal state:* state of the robot
  - Sensed using internal sensors
  - Stored/remembered
  - E.g.: velocity, mood
- The robot's state is a <u>combination</u> of its external and internal state.

State and Intelligence

- *State space*: all possible states the system can be in
- *A challenge*: sensors do not provide state!
- *How intelligent a robot appears is strongly dependent on how much it can sense about its environment and about itself.*

Internal Models

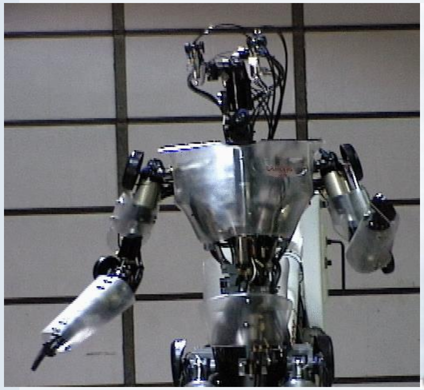
- Internal state can be used to remember information about the world (e.g., remember paths to the goal, remember maps, remember friends v. enemies, etc.)
- This is called a *representation* or an *internal model*.
- *Representations/models have a lot to do with how complex a controller is!*

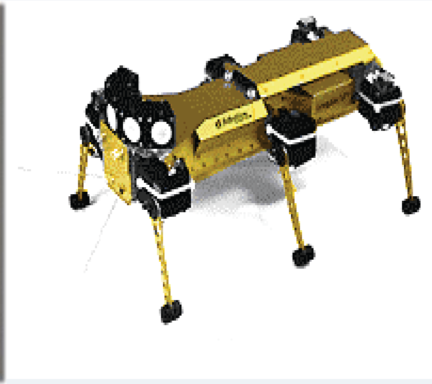
Action/Actuation

- A robot acts through its *actuators* (e.g., motors), which typically drive *effectors* (e.g., wheels)
- Robotic actuators are very different from biological ones, both are used for:
  - *locomotion* (moving around, going places)
  - *manipulation* (handling objects)
- This divides robotics into two areas
  - mobile robotics
  - manipulator robotics

Actuators and DOF

- Mobile robots move around using wheels, tracks, or legs
- Mobile robots typically move in 2D (but note that swimming and flying is 3D)
- Manipulators are various robot arms
- They can move from 1 to many D
- Think of the dimensions as the robot's *degrees of freedom (DOF)*






Action vs. Behavior

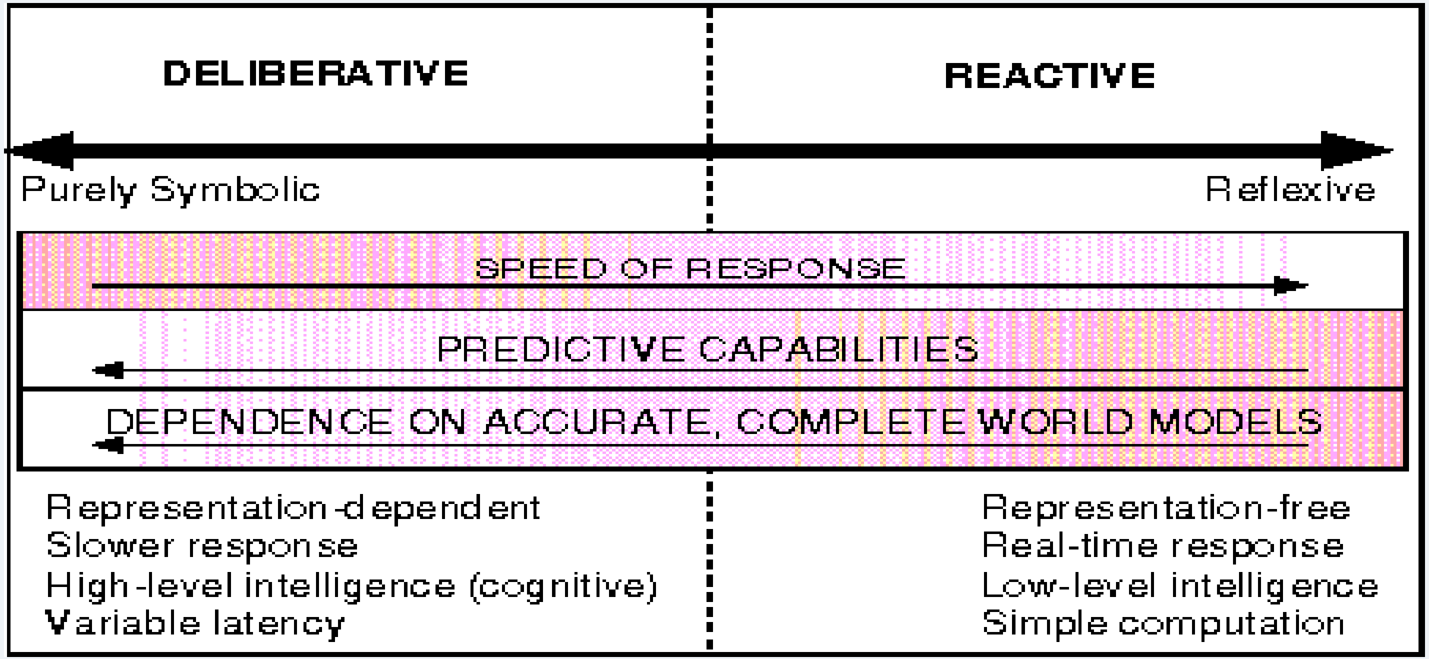
- *Behavior* is what an external observer sees a robot doing.
- Robots are programmed to display desired behavior.
- Behavior is a result of a sequence of robot actions.
- Observing behavior may not tell us much about the internal control of a robot. Control can be a black box.

Autonomy

- *Autonomy is the ability to make one's own decisions and act on them.*
- For robots, autonomy means the ability to sense and act on a given situation appropriately.
- Autonomy can be:
  - *complete (e.g., R2D2)*
  - *partial* (e.g., tele-operated robots)

Control

- *Robot control* refers to the way in which the sensing and action of a robot are coordinated.
- The many different ways in which robots can be controlled all fall along a well-defined *spectrum of control*.



A brief history of robotics

- Feedback control
- Cybernetics
- Artificial Intelligence (AI)
- Early robotics
- Robotics today
  - Why is robotics hard?

Feedback Control

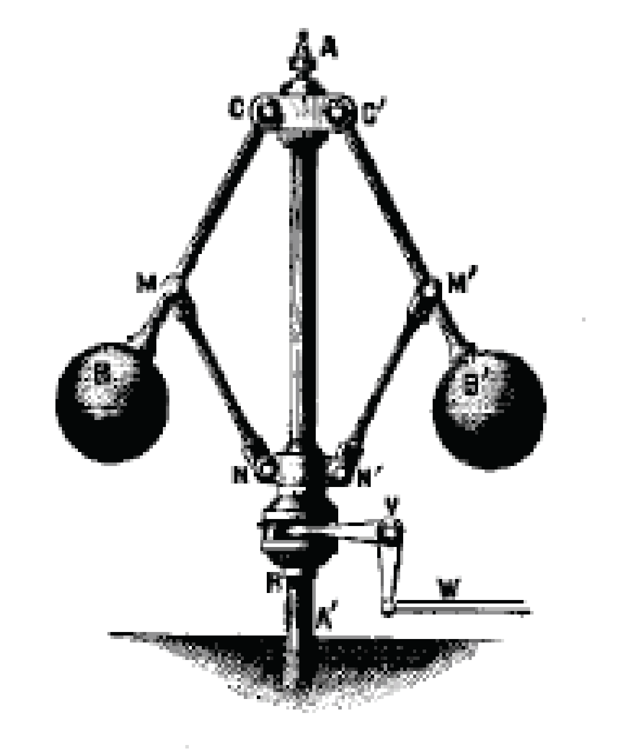
- ***Feedback***: *continuous monitoring of the sensors and reacting to their changes.*
- Feedback control = self-regulation
- Two kinds of feedback:
  - Positive
  - Negative
- The basis of control theory

− and + Feedback

- *Negative feedback*
  - acts to regulate the state/output of the system
  - e.g., if too high, turn down, if too low, turn up
  - thermostats, toilets, bodies, robots...
- *Positive feedback*
  - acts to amplify the state/output of the system
  - e.g., the more there is, the more is added
  - lynch mobs, stock market, ant trails...

Uses of Feedback

- Invention of feedback as the first simple robotics (does it work with our definition)?
- The first example came from ancient Greek water systems (toilets)
- Forgotten and re-invented in the Renaissance for ovens/furnaces
- Really made a splash in Watt's steam engine

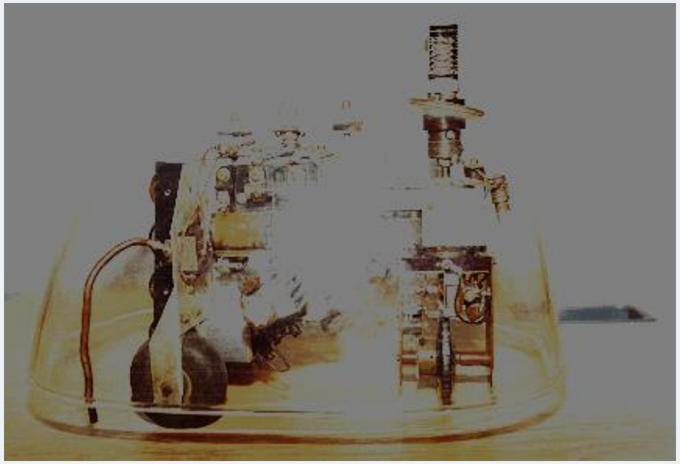


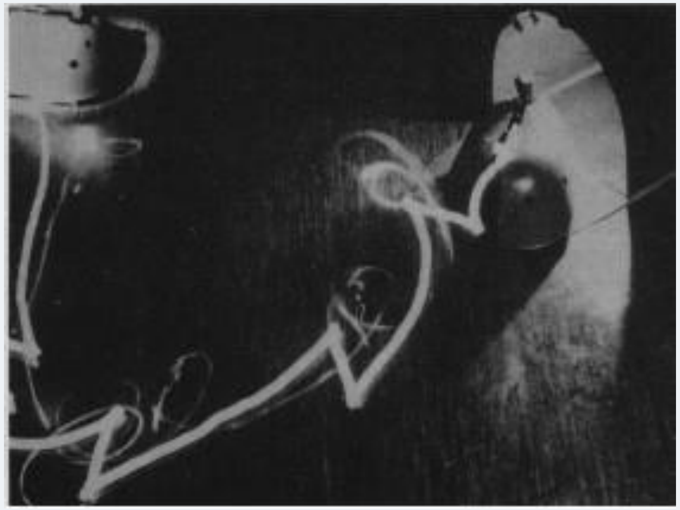
Cybernetics

- Pioneered by Norbert Wiener (1940s)
  - (From Greek "steersman / governor" of steam engine)
- Marriage of control theory (feedback control), information science and biology
- Seeks principles common to animals and machines, especially for control and communication
- Coupling an organism and its environment (*situatedness*)

W. Grey Walter's Tortoise (1950's)

- *Machina Speculatrix*
- 1 photocell & 1 bump sensor, 1 motor
- Behaviors:
  - seek light
  - head to weak light
  - back from bright light
  - turn and push
  - recharge battery
- Reactive control





The Walter Turtle in Action

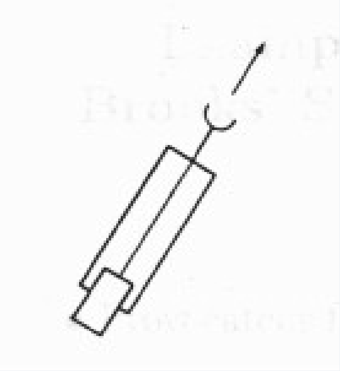
Braitenberg’s Vehicles

- Valentino Braitenberg (early 1980s)
- Extended Walter’s model in a series of thought experiments
- Also based on analog circuits
- Direct connections (excitatory or inhibitory) between light sensors and motors
- Complex behaviors from very simple mechanisms

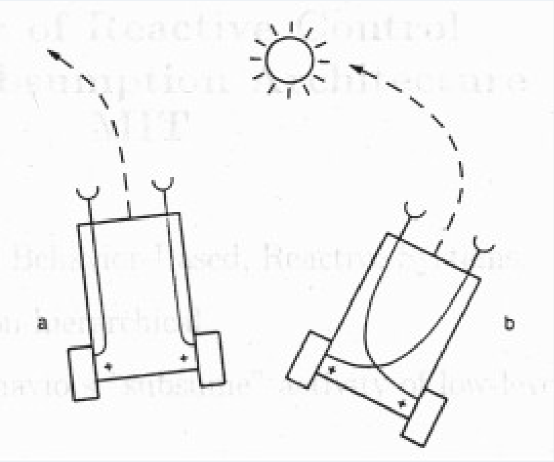
Braitenberg’s Vehicles

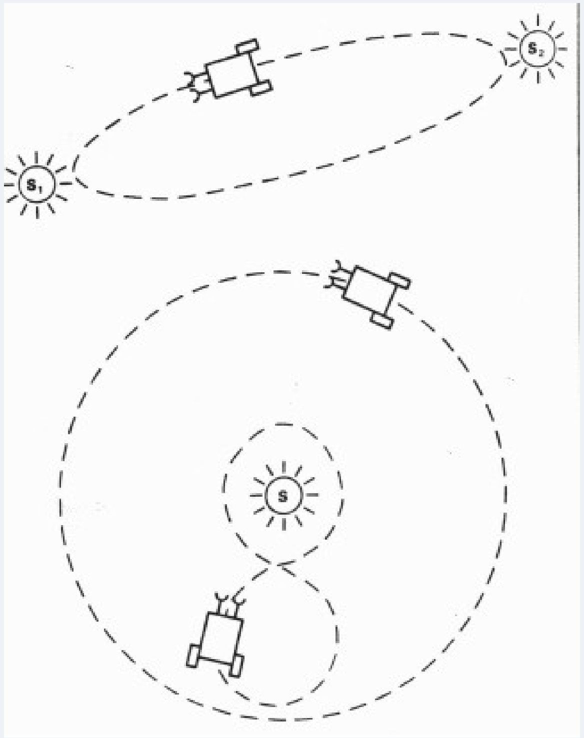
- Examples of Vehicles:

V1:



V2:





Braitenberg's Vehicles

- By varying the connections and their strengths, numerous behaviors result, e.g.:
  - "fear/cowardice" - flees light
  - "aggression" - charges into light
  - "love" - following/hugging
  - many others, up to memory and learning!

- Reactive control
- Later implemented on real robots

Early Artificial Intelligence

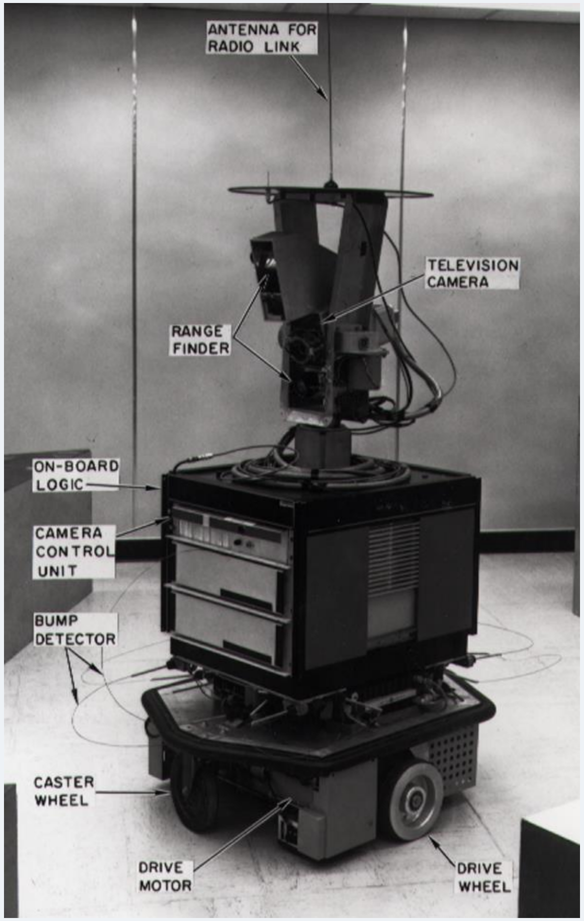
- "Born" in 1955 at Dartmouth
- "Intelligent machine" would use internal models to search for solutions and then try them out (M. Minsky) => deliberative model!
- *Planning* became the tradition
- Explicit symbolic representations
- Hierarchical system organization
- Sequential execution

Artificial Intelligence (AI)

- Early AI had a strong impact on early robotics
- Focused on knowledge, internal models, and reasoning/planning
- Basis of deliberative control in early robots

Early Robots: SHAKEY

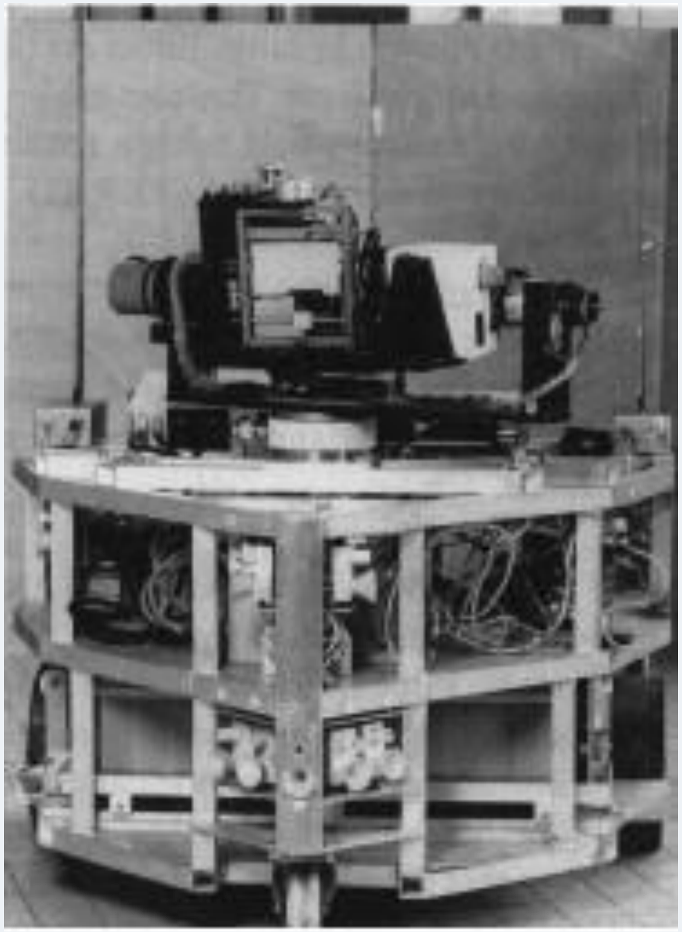
- At Stanford Research Institute (late 1960s)
- Vision and contact sensors
- STRIPS planner
- Visual navigation in a special world
- Deliberative



1972 model

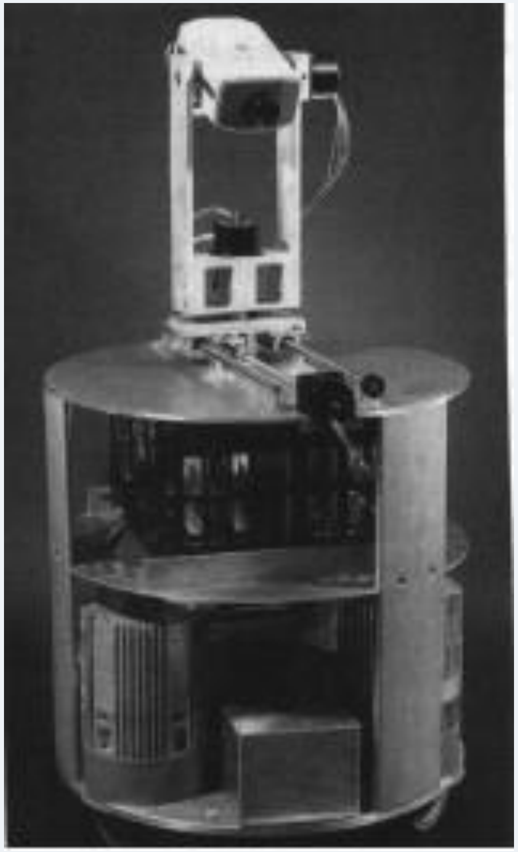
Early Robots: HILARE

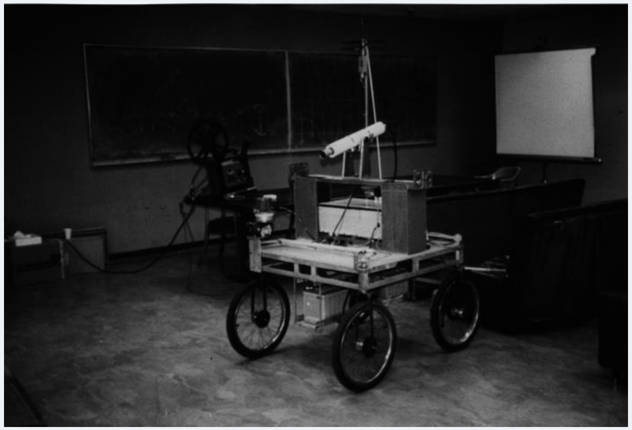
- LAAS in Toulouse, France (late 1970s)
- Video, ultrasound, laser range-finder
- Still in use!
- Multi-level spatial representations
- Deliberative -> Hybrid Control



Early Robots: CART/Rover

- Hans Moravec
- Stanford Cart (1977) followed by CMU rover (1983)
- Sonar and vision
- Deliberative control





What’s the problem?

- Robot is slow in deliberation
  - Based on the *sense->plan->act* (SPA) model
  - Inherently sequential
  - Planning requires search, which is slow
  - Search requires a world model
  - World models become outdated
  - Search and planning takes too long

Reactive Systems

- Collections of *sense-act (stimulus-response) rules*
  - Inherently *concurrent (parallel)*
  - No/minimal state
  - No memory
  - Very fast and reactive
  - Unable to plan ahead
  - Unable to learn
- Sentinel papers:
  - Brooks, R. A., Intelligence Without Reason
  - Brooks, R.A., A robust layered control system for a mobile robot.

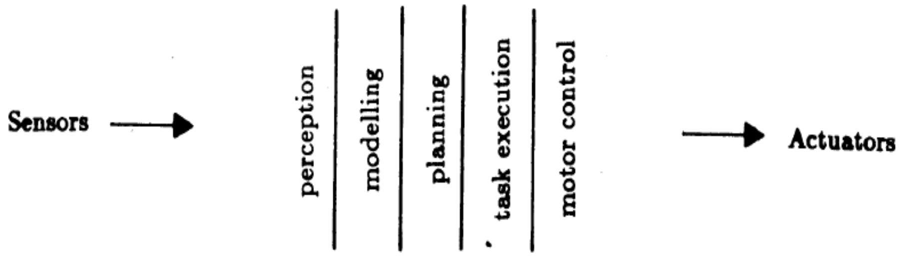


Figure 1. A traditional decomposition of a mobile robot control system into functional modules.

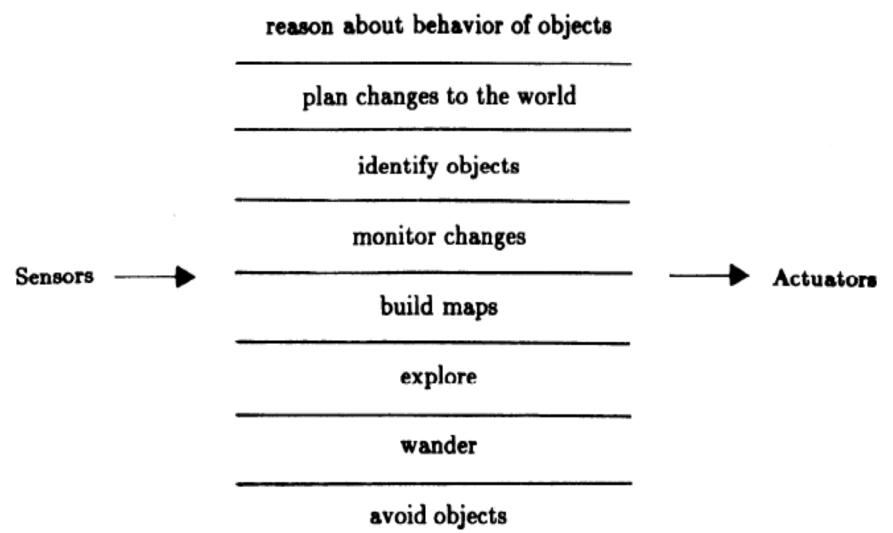
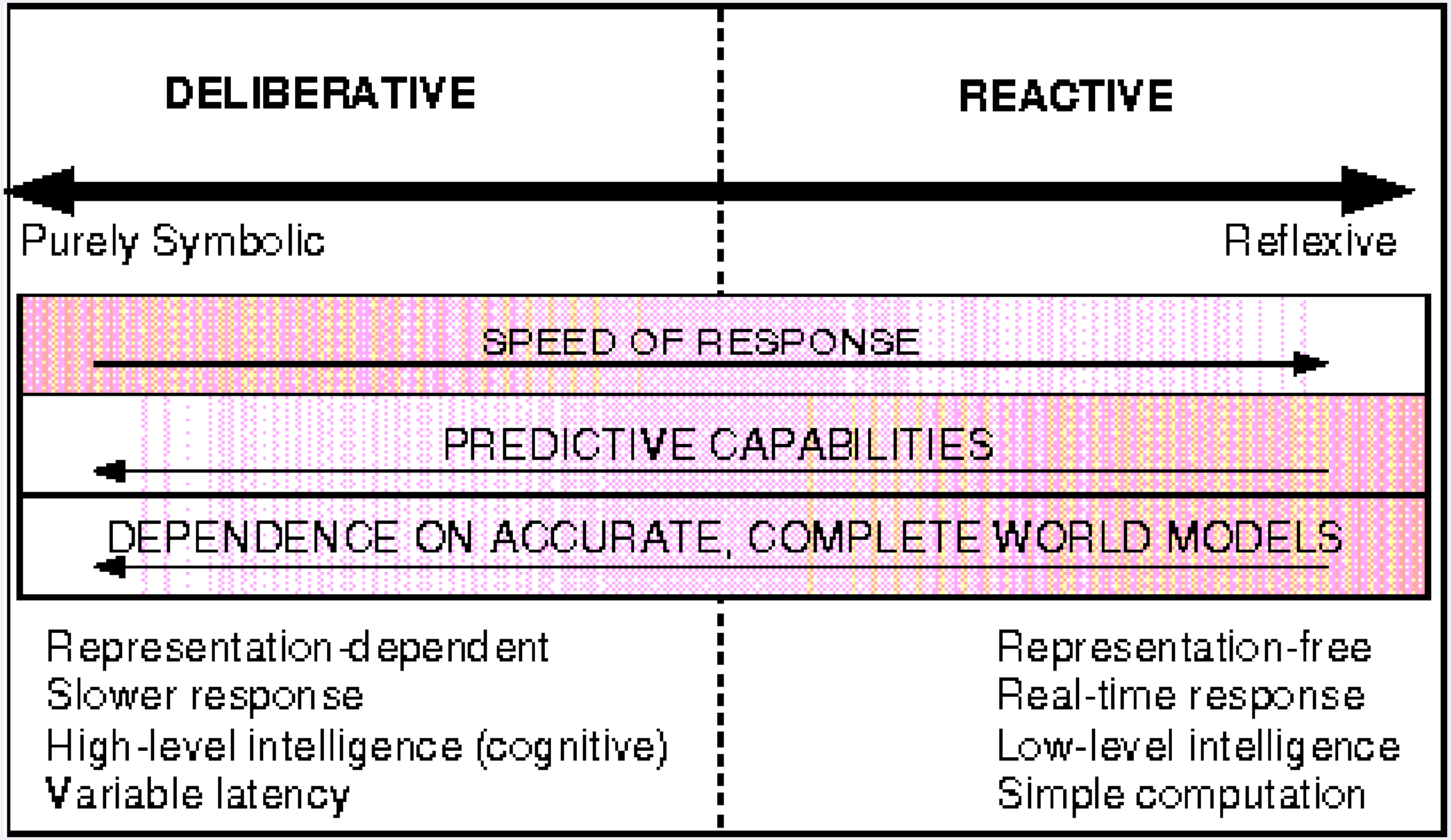


Figure 2. A decomposition of a mobile robot control system based on task achieving behaviors.

Spectrum of Control



Hybrid Systems

- Combine the two extremes
  - reactive system on the bottom
  - deliberative system on the top
  - connected by some intermediate layer
- Often called 3-layer systems
- Layers must operate *concurrently*
- Different *representations and time-scales* between the layers
- The best or worst of both worlds?

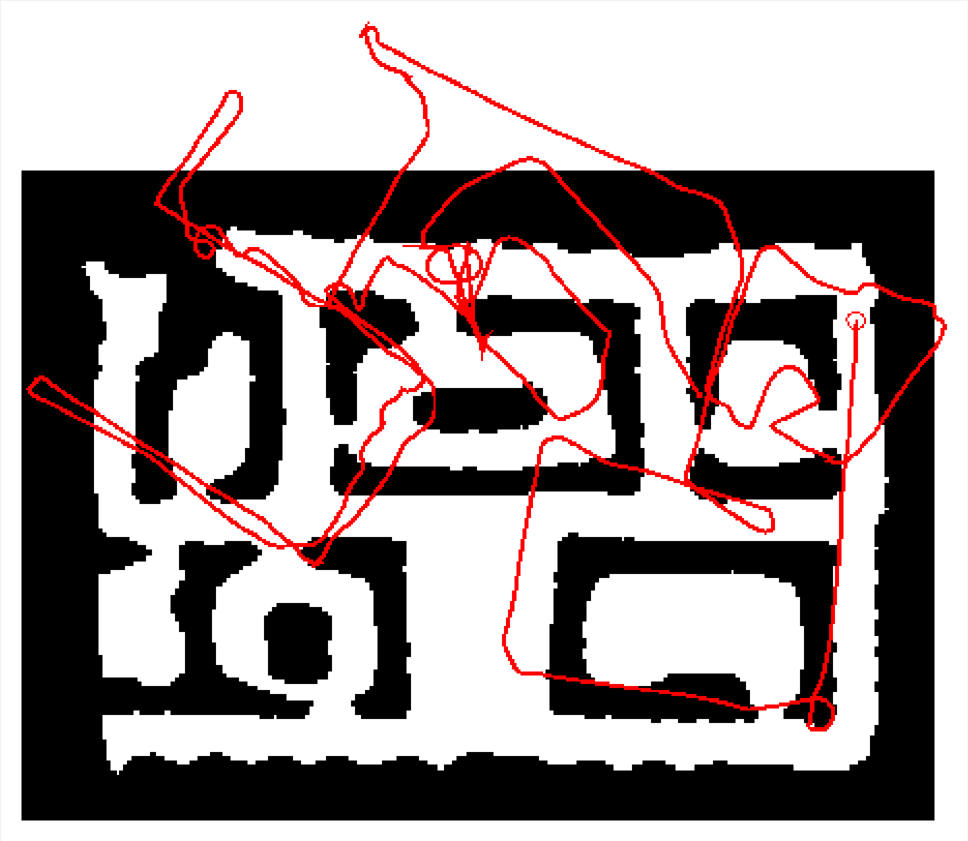
Behavior-Based Systems

- An alternative to hybrid systems
- Have the same capabilities
  - the ability to act reactively
  - the ability to act deliberatively
- There is no intermediate layer
- A unified, consistent representation is used in the whole system=> *concurrent behaviors*
- That resolves issues of time-scale

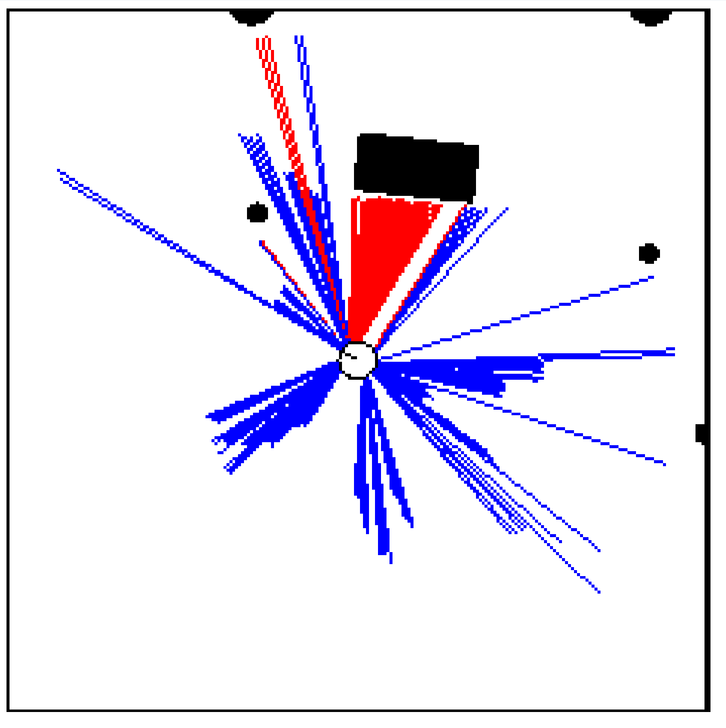
So, are we done?

- Sensors are limited and crude
- Effectors are limited and crude
- State (internal and external, but mostly external) is partially-observable
- Environment is dynamic (changing over time)
- Environment is full of potentially-useful (and useless) information

Nature of Sensor Data



Odometry Data



Range Data

Probabilistic Robotics

- Address the fundamental problem of robotics i.e. *how to combat uncertainty* using the tools of probability theory
- The need to model errors from sensors, actuators, and dynamic environment.

$$P(x, y) = P(x \mid y) P(y) = P(y \mid x) P(x)$$

$$\Rightarrow$$

$$P(x \mid y) = \frac{P(y \mid x)\ P(x)}{P(y)} = \frac{\text{likelihood} \cdot \text{prior}}{\text{evidence}}$$

Recursive Bayesian Updating

state observations

$$P(x \mid z_1, \dots, z_n) = \frac{P(z_n \mid x, z_1, \dots, z_{n-1}) \, P(x \mid z_1, \dots, z_{n-1})}{P(z_n \mid z_1, \dots, z_{n-1})}$$

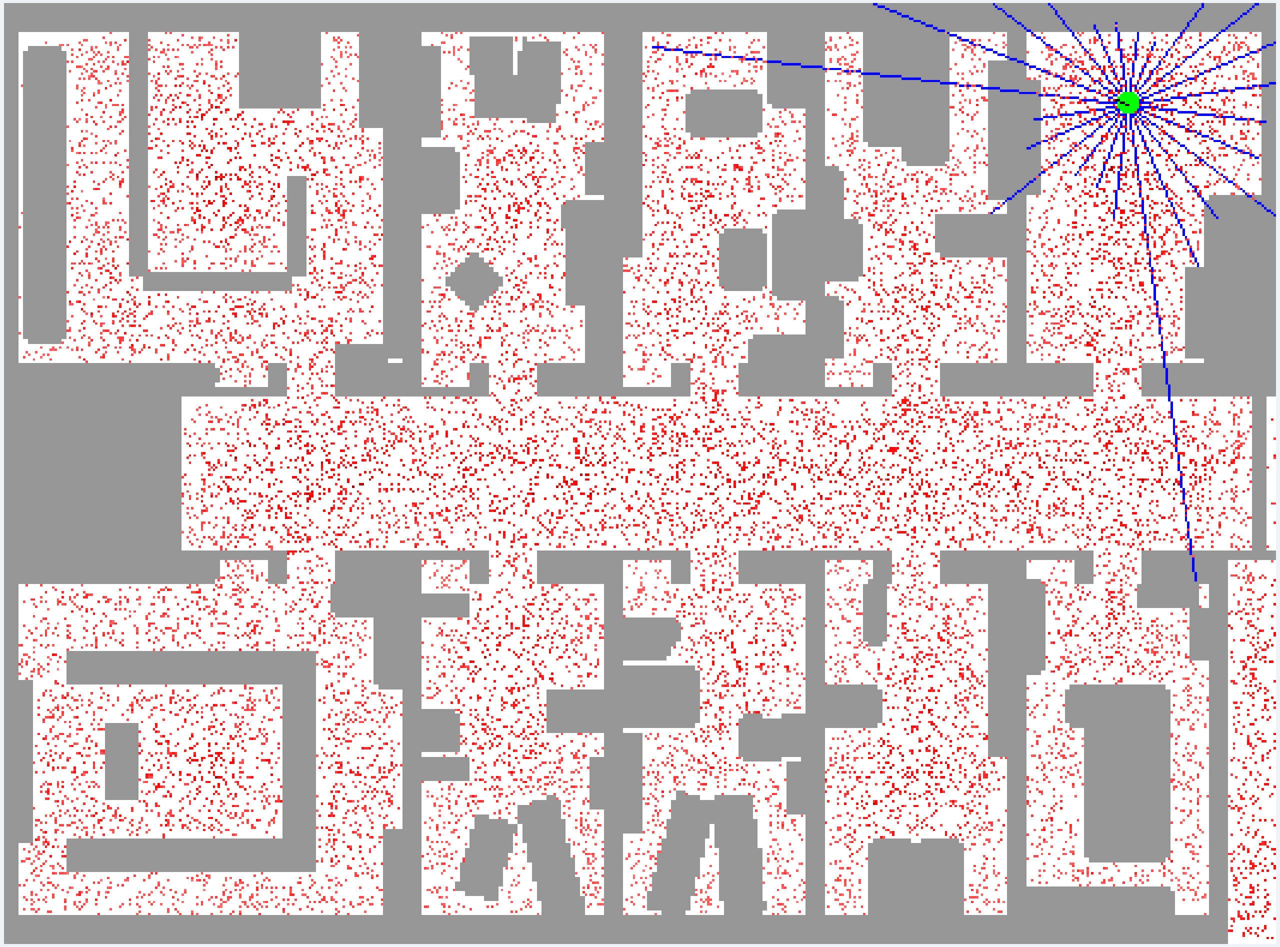
Markov assumption: z_n is independent of $z_1, \dots, z_{n-1}$ if we know x.

$$P(x \mid z_1, \dots, z_n) = \frac{P(z_n \mid x) \, P(x \mid z_1, \dots, z_{n-1})}{P(z_n \mid z_1, \dots, z_{n-1})}$$

$$= \eta \, P(z_n \mid x) \, P(x \mid z_1, \dots, z_{n-1})$$

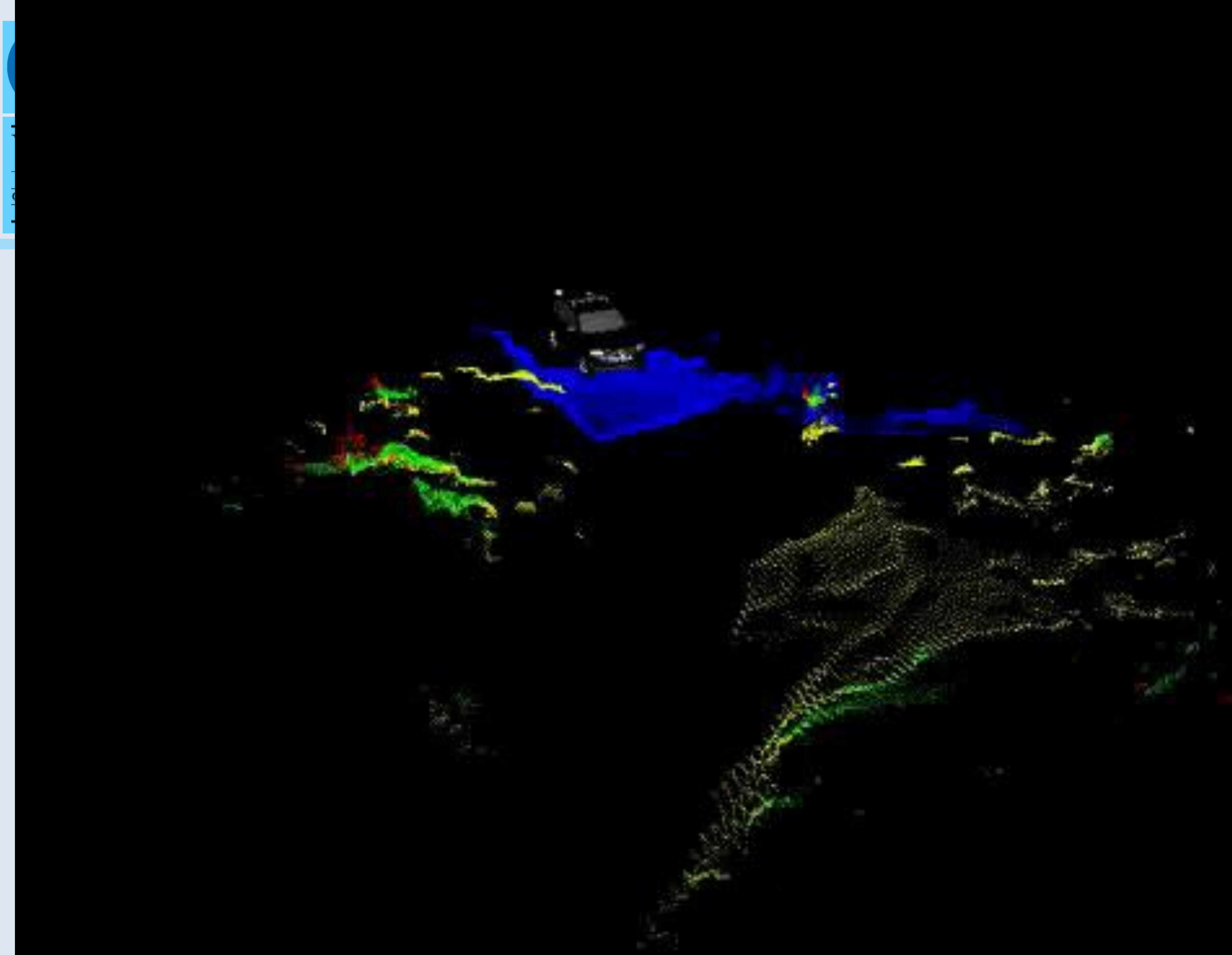
$$= \eta_{1 \dots n} \prod_{i=1 \dots n} P(z_i \mid x) \, P(x)$$

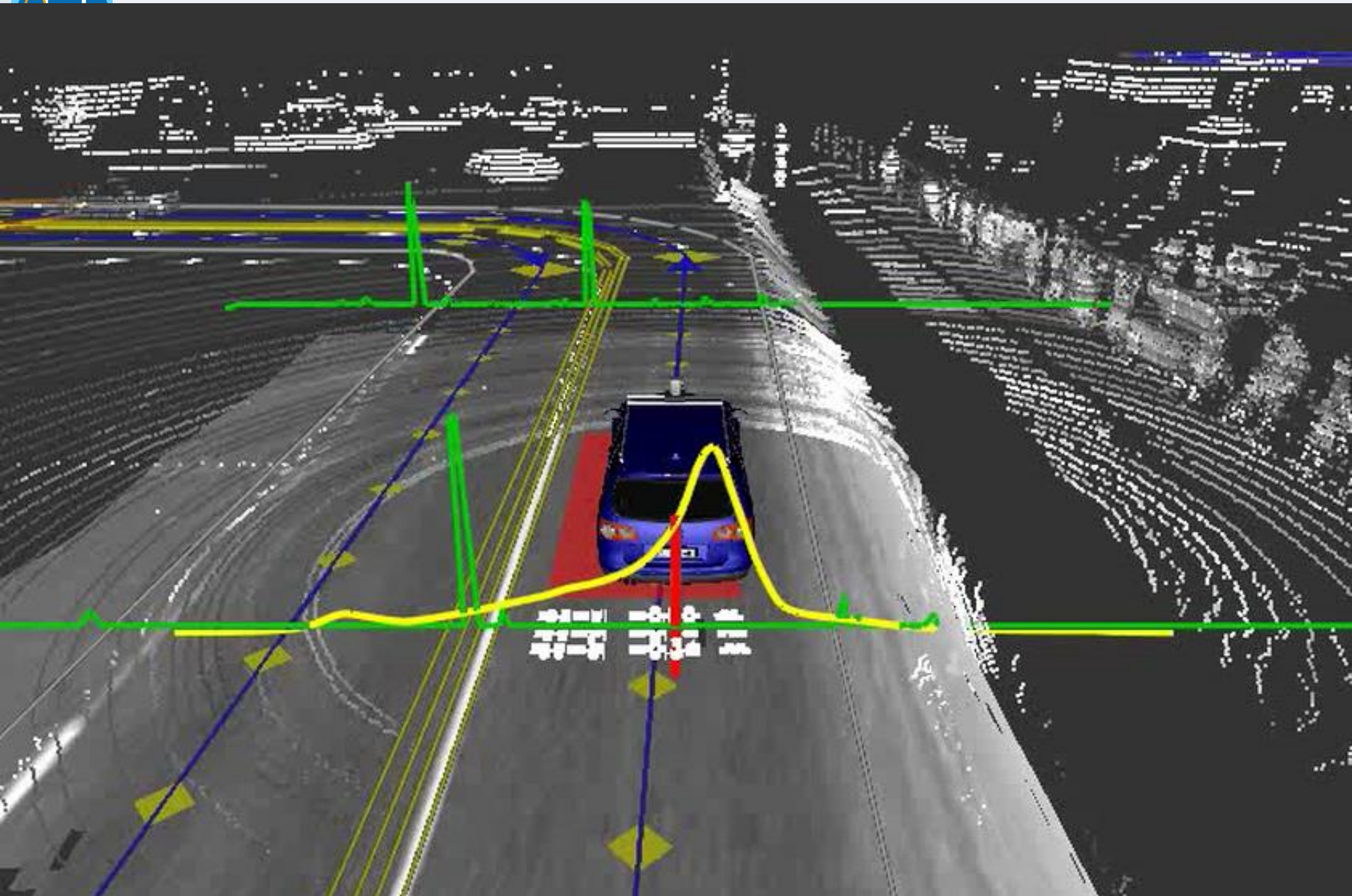
Localization



Navigation Challenges

- Moving from place to place
- Why is navigation hard?
  - Limitations and errors in sensing
  - Uncertainty in localization (where am I?)
  - Uncertainty in map or no map
  - Uncertainty in actuation





Navigation Involves

- Localization
- Path planning
- Search / coverage
- Mapping
- Simultaneous Localization and Mapping (SLAM)

Localization

- Have a map of the world, need to figure out where you are on the map.

- Can use:
  - Global knowledge - GPS
  - Dead-reckoning - odometry
  - Landmark detection
- Problems:
  - Estimation process is indirect
  - Measurements are noisy
  - Measurements are not always available

Landmark Detection

- Compare view of world as perceived by sensors to stored world model - try to find a match

- Examples:
  - Laser scans - scan matching
  - Vision - visual landmarks

- Problems:
  - Output from sensors needs to match the format of the world model representation. (eg difficult to use vision to match to a sonar-built map)
  - Data association - may parts of the environment look the same
  - Environment changes - world model becomes outdated (e.g., change in lighting -> shadows change -> vision problems)

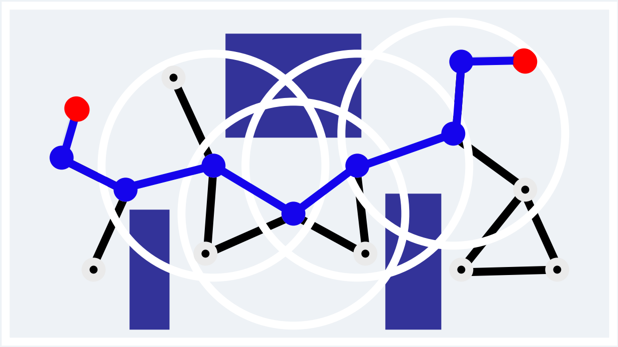
Path Planning

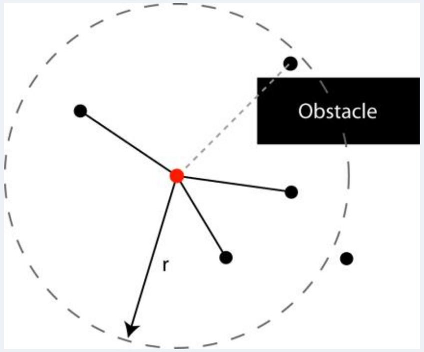
- Finding a route from start to goal location
- Need a map / world model
- Need localization
- World is often represented as a graph (topological map)
- Use graph search techniques to find path (e.g., A*)

- Problems:
  - Searching is slow
  - Dynamic environment -> world model outdated -> need to detect changes and re-plan new route.

Planning Example - PRM

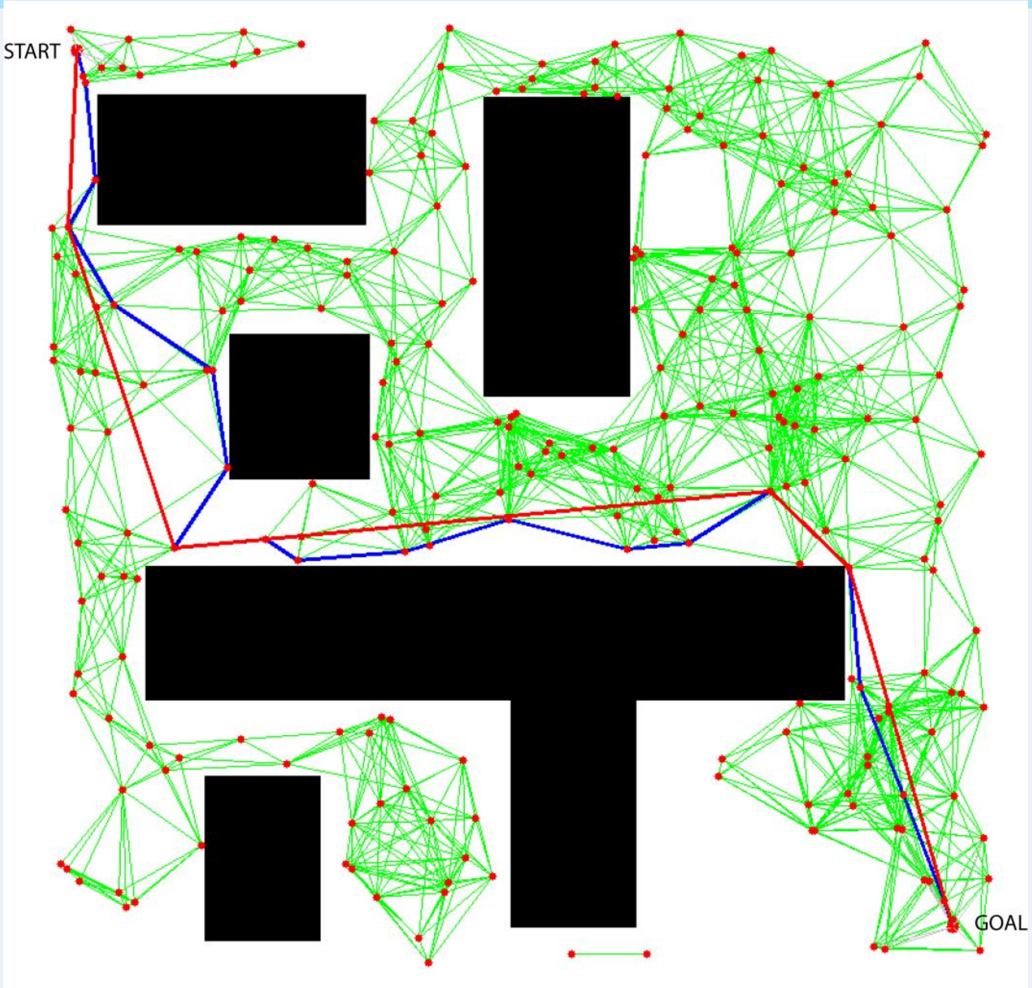
- Probabilistic Roadmaps
  - Map Building Phase
    - Randomly add nodes to C_{free}
    - Attempt to connect to other nodes (local planner)
  - Query Phase
    - Connect start and goal to map
    - Use A* to find path from start to goal





Local Planner

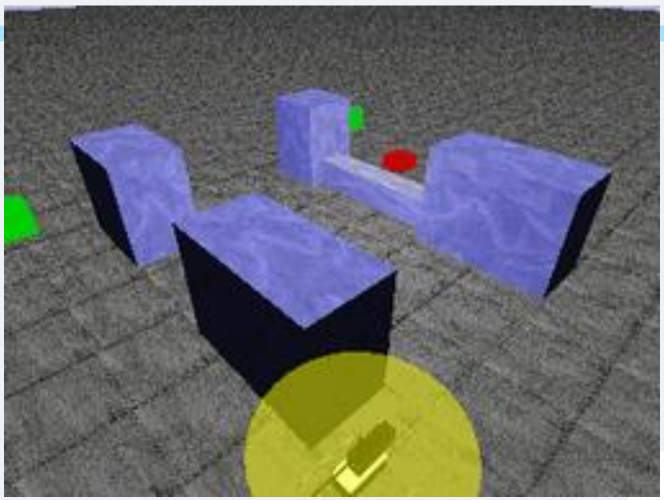
PRM Result



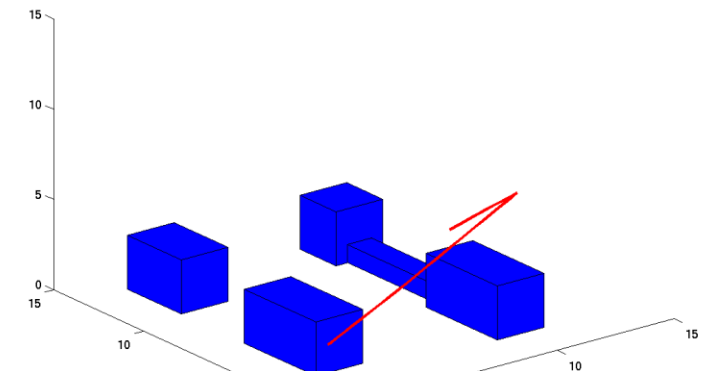
Original path

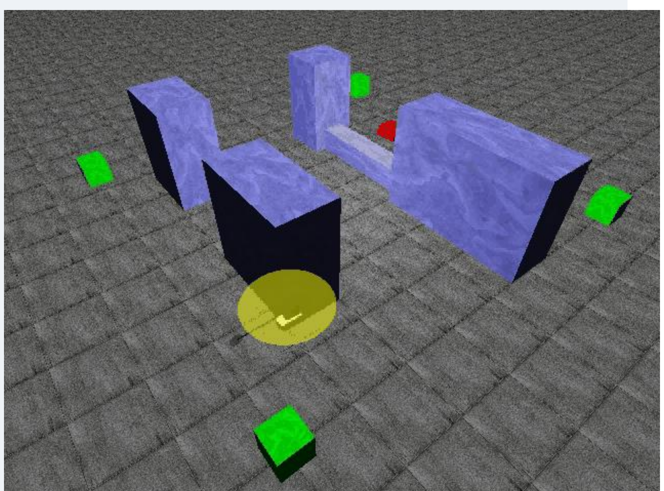
After path smoothing

3D PRM results

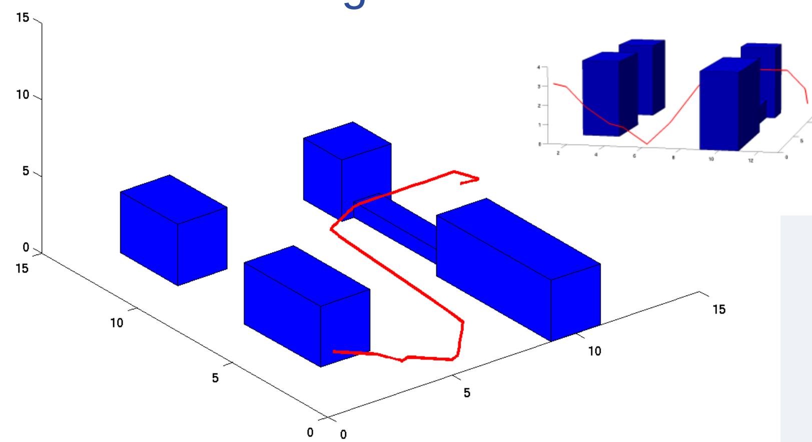


No Height Constraint





4m Height Constraint



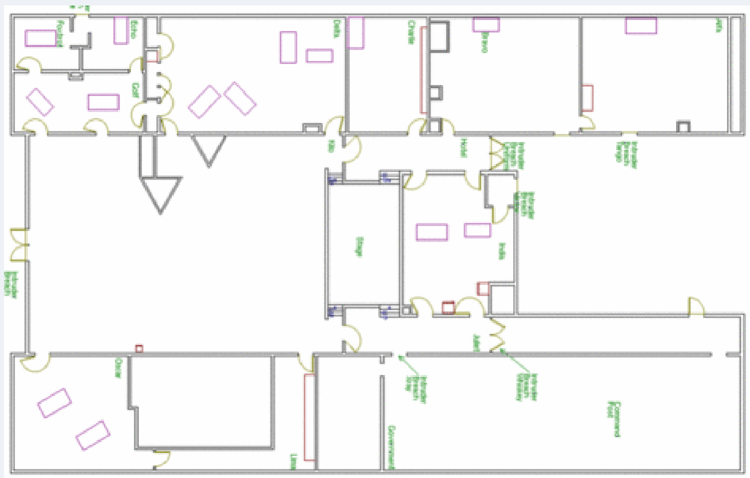
SLAM

- SLAM: simultaneous localization and mapping
- To start - no map and don't know where you are
- "Chicken and egg" problem: Need map to localize, but can't build a map if you aren't localized
- Solution: Run mapping and localization in parallel - as map accuracy improves, localization improves, which improves the map, and so on

Coverage and Exploration

- Try to explore an unknown area
- Try to find something in an area
- With map - can plan a path that covers the area
- Without map - try to move in a systematic manner
  - Follow boundaries
  - Spiral outward
  - Random motion
  - E.g. - Roomba

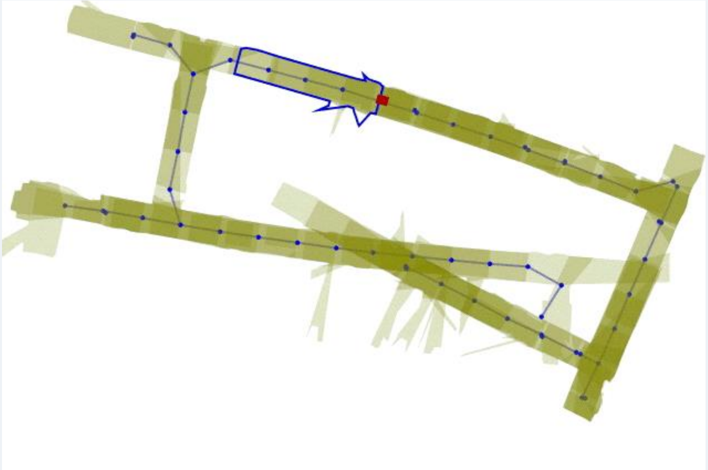
Example Exploration Results



Common Navigation Challenges

- Kidnapped Robot
  - Robot is localized, but is then moved (without being told)
  - Now it thinks it is still somewhere else - needs to re-localize

- Closing the loop
  - When building a map, robot returns to the same place via another route - needs to realize this is the same place, and not a new area.



Monte Carlo Localization

- the probability density function is represented by samples
- randomly drawn from it
- it is also able to represent multi-modal distributions, and thus localize the robot *globally*
- considerably reduces the amount of memory required and can integrate measurements at a higher rate
- state is not discretized and the method is more accurate than the grid-based methods
- easy to implement

"Probabilistic Robotics"

- Sebastian Thrun

- Bayes' rule

$$p(A \mid B) = \frac{p(B \mid A) \cdot p(A)}{p(B)}$$

- Definition of *marginal* probability

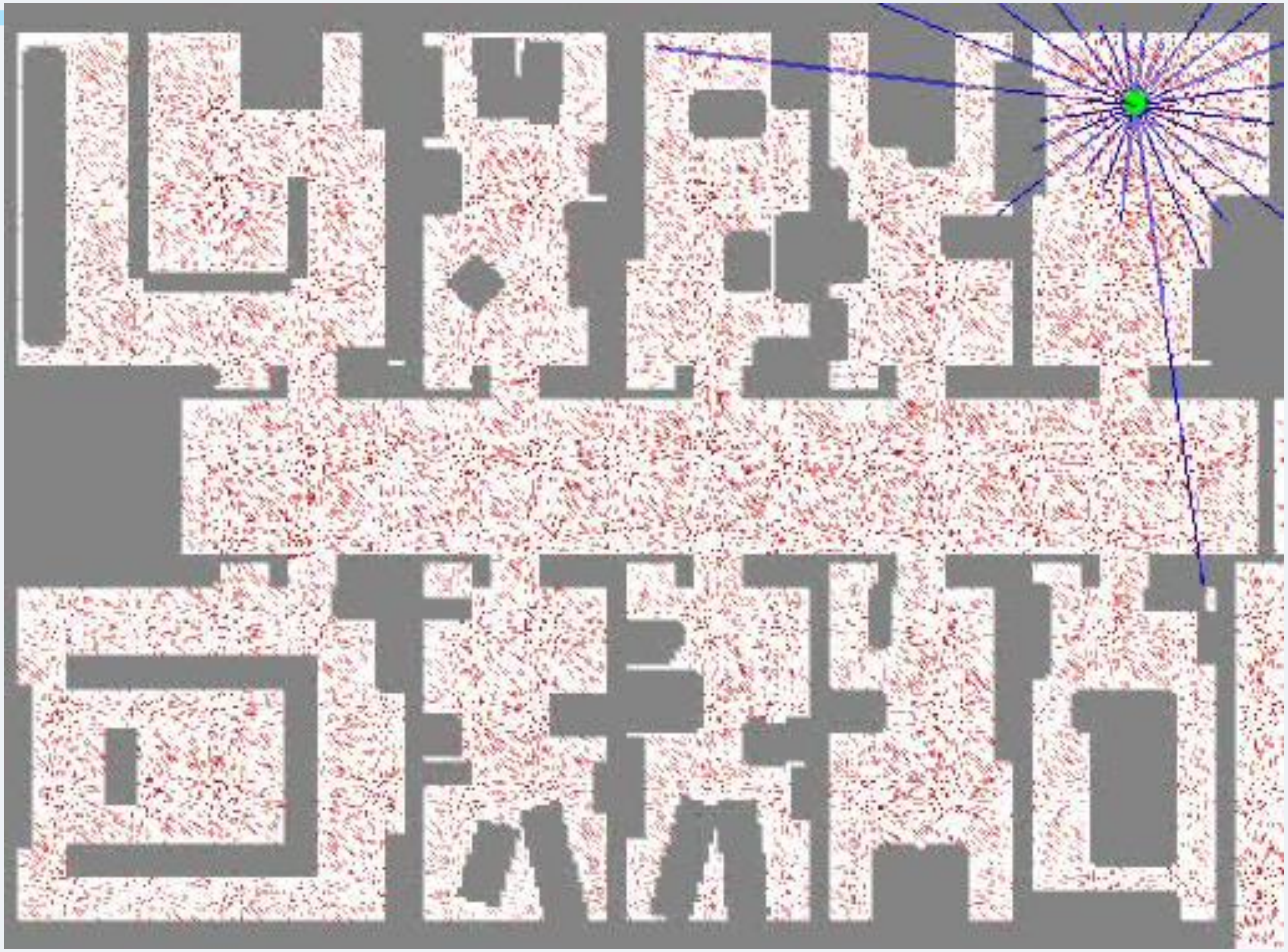
$$p(A) = \sum_{\text{all } B} p(A \wedge B)$$

$$p(A) = \sum_{\text{all } B} p(A \mid B) \cdot p(B)$$

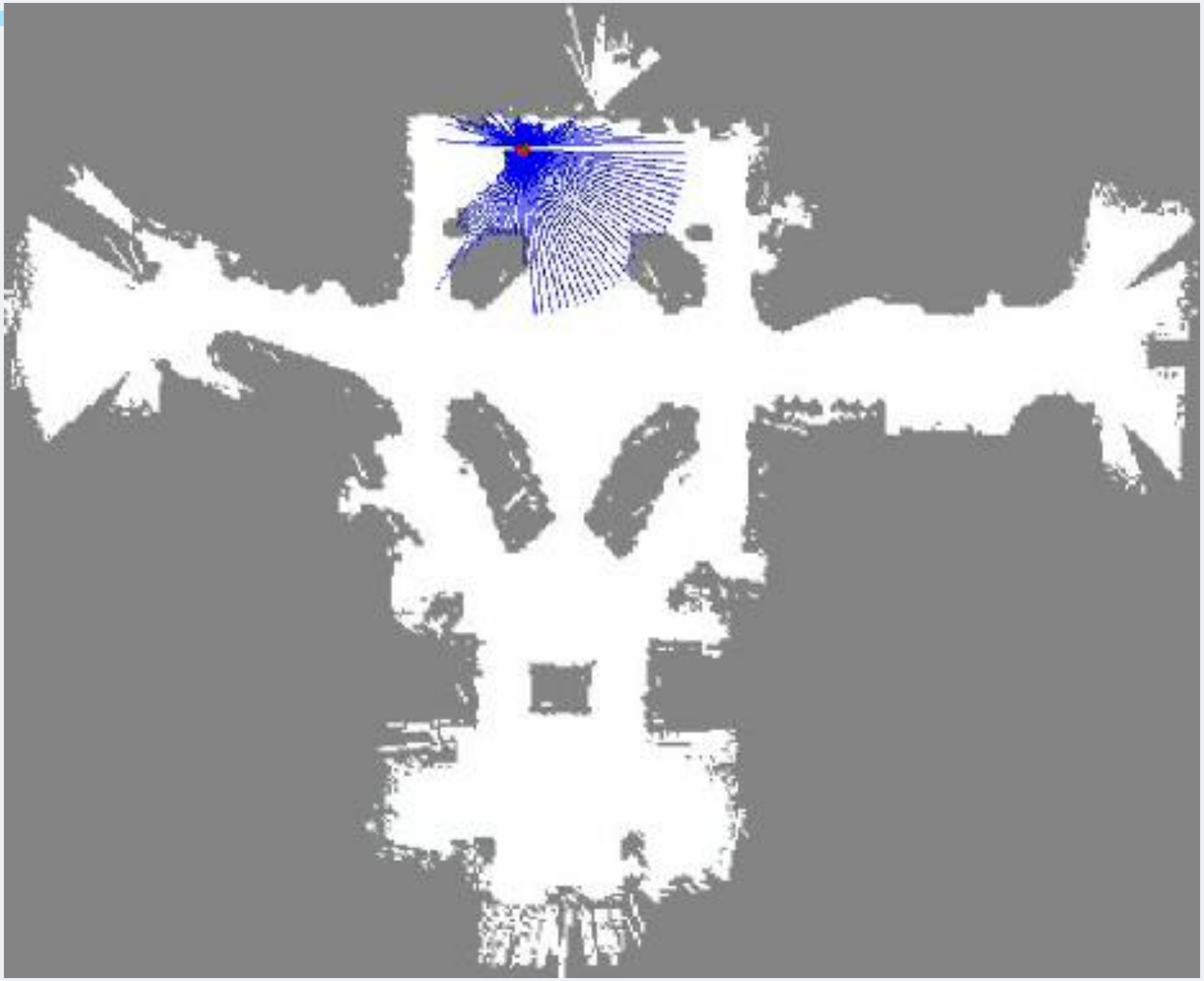
- Definition of *conditional* probability iLab

USC
UNIVERSITY
OF SOUTHERN
CALIFORNIA

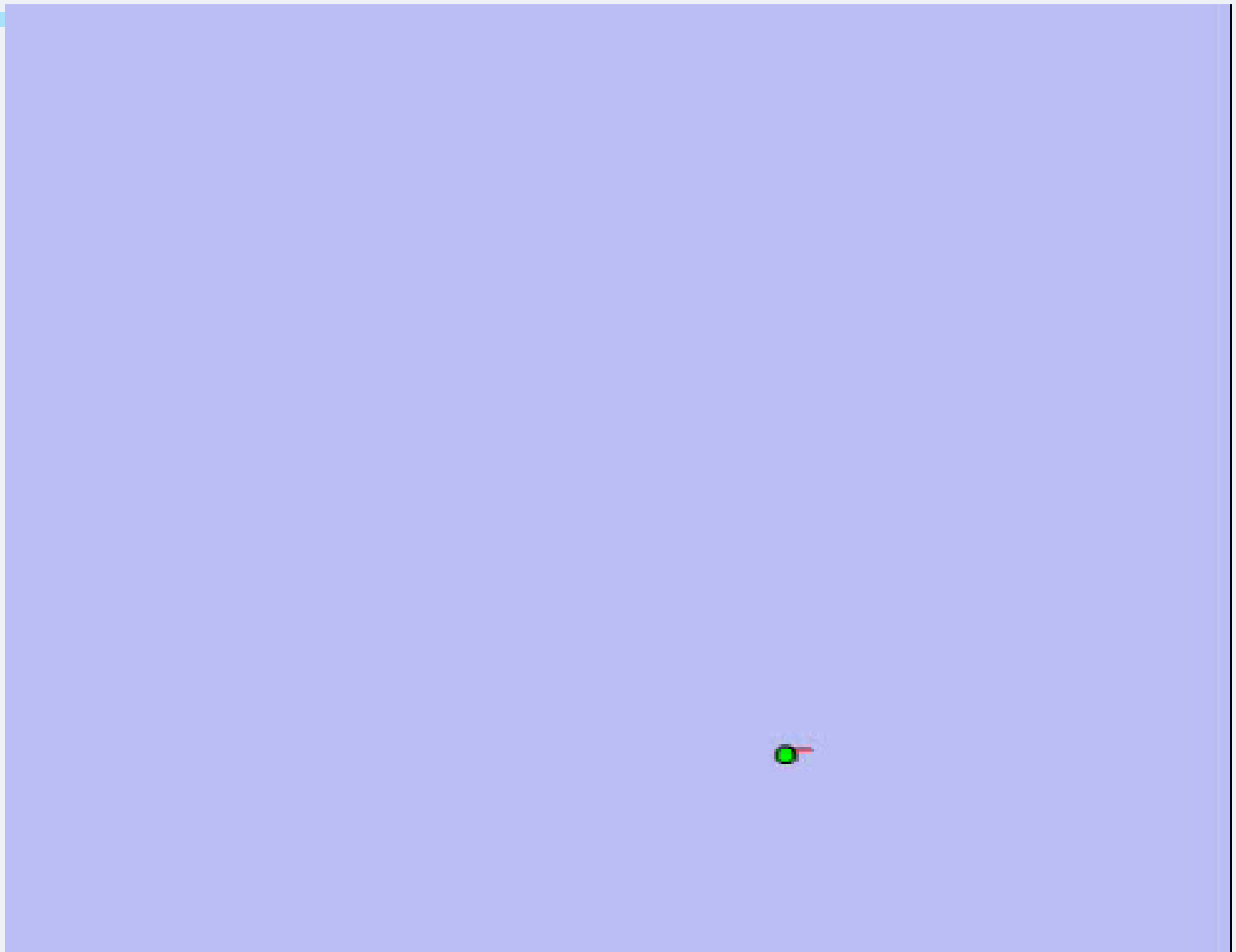
Monte-Carlo localization (Thrun)



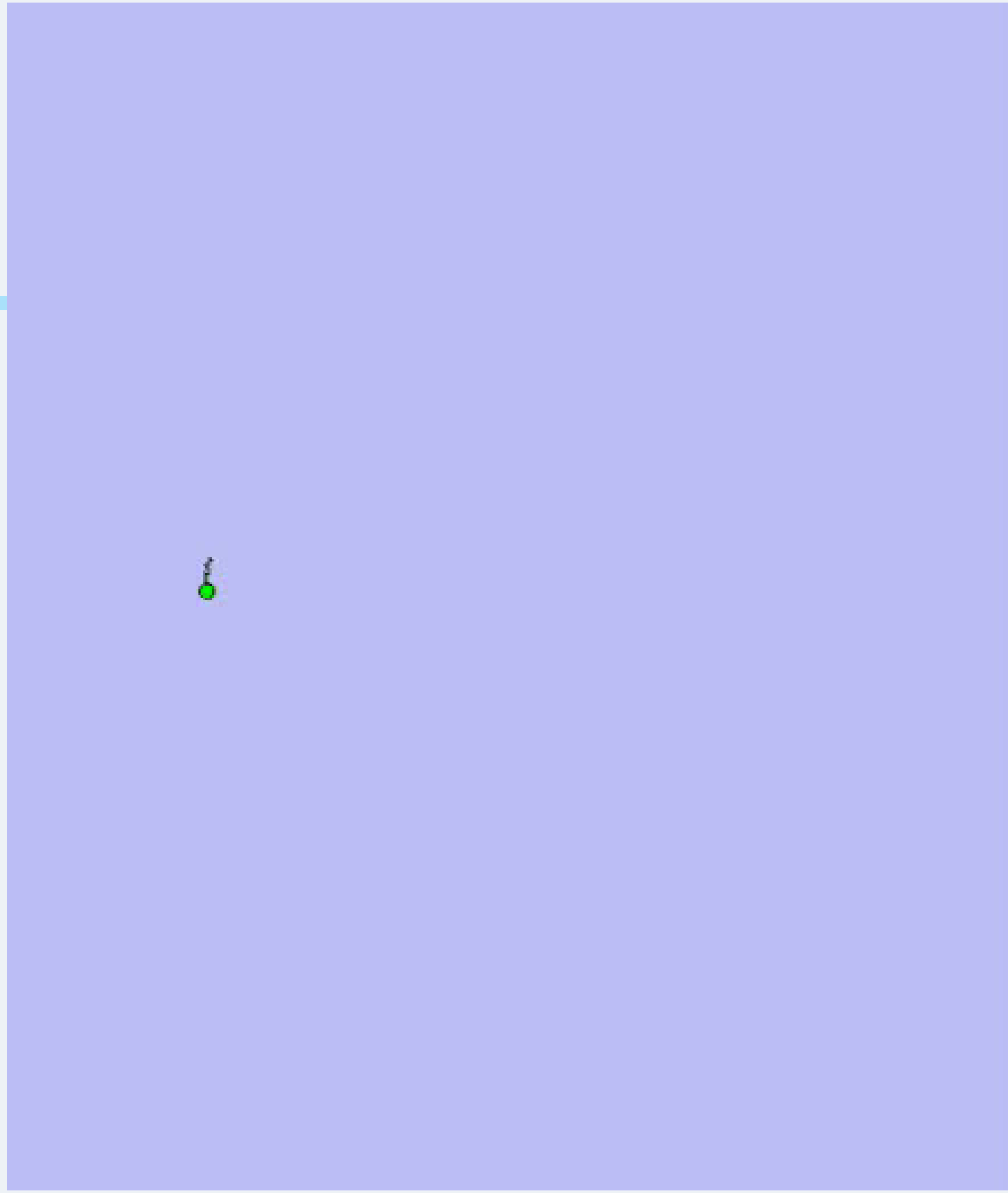
Monte-Carlo Localization



FASTslam (Thrun)



FASTslam

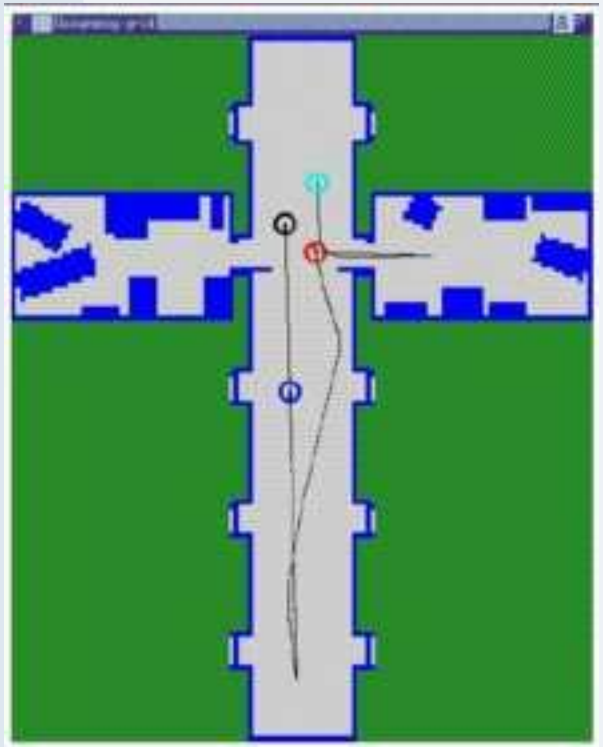


Setting up the problem

(following slides from Jizhong Xiao, CUNY)

The robot does (or can be modeled to) alternate between

- sensing -- getting range observations $o_1, o_2, o_3, \ldots, o_{t-1}, o_t$
- acting -- driving around (or ferrying?) $a_1, a_2, a_3, \ldots, a_{t-1}$



Setting up the problem

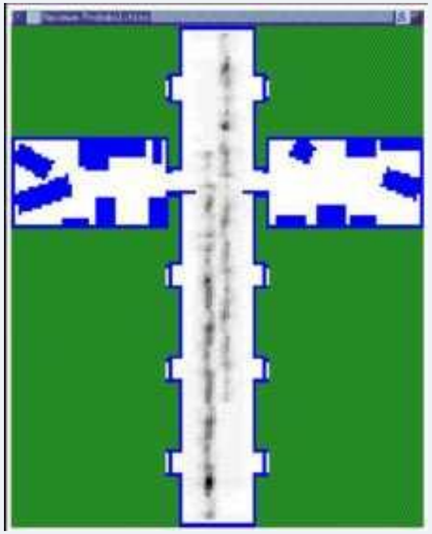
The robot does (or can be modeled to) alternate between

- sensing -- getting range observations $o_1, o_2, o_3, \ldots, o_{t-1}, o_t$
- acting -- driving around (or ferrying?) $a_1, a_2, a_3, \ldots, a_{t-1}$

We want to know r_t -- the position of the robot at time t

- but we'll settle for $p(r_t)$ -- the probability distribution for r_t

What *kind* of thing is $p(r_t)$?



Setting up the problem

The robot does (or can be modeled to) alternate between

- sensing -- getting range observations $o_1, o_2, o_3, \ldots, o_{t-1}, o_t$
- acting -- driving around (or ferrying?) $a_1, a_2, a_3, \ldots, a_{t-1}$

We want to know r_t -- the position of the robot at time t

- but we'll settle for $p(r_t)$ -- the probability distribution for r_t

! What *kind* of thing is $p(r_t)$?

We do know m -- the map of the environment

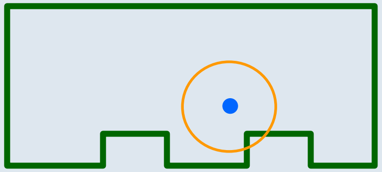
$p(o \mid r, m)$ -- the sensor model

$p(r_{new} \mid r_{old}, a, m)$ -- the accuracy of desired action a

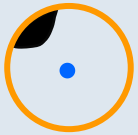
Robot modeling

p(o | r, m) sensor model

map m and location r



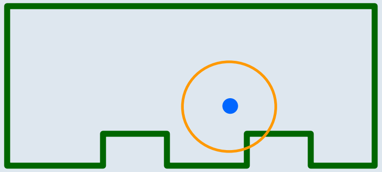
p( | r, m) = .75

p( | r, m) = .05

potential observations o

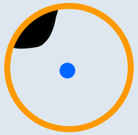
Robot modeling

$p(\, o \mid r, m\,)$ sensor model

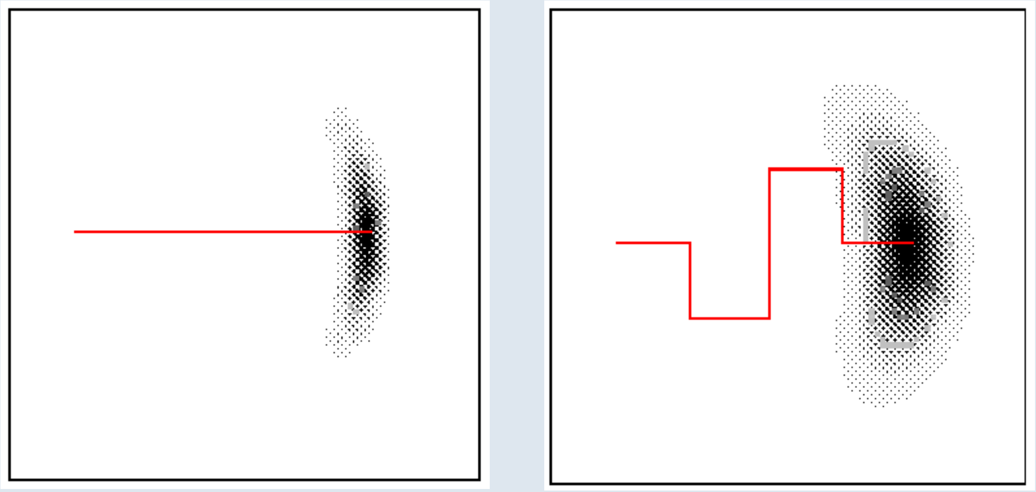


$p(\, r_{new} \mid r_{old}, a, m\,)$ action model

$p($  $\mid r, m\,) = .75$

$p($  $\mid r, m\,) = .05$

potential observations o



"probabilistic kinematics" -- encoder uncertainty

- red lines indicate commanded action
- the cloud indicates the likelihood of various final states

Robot modeling: how-to

p(o | r, m) sensor model

p(r_{new} | r_{old}, a, m) action model

(0) Model the physics of the sensor/actuators (with error estimates)

theoretical modeling

(1) Measure lots of sensing/action results and create a model from them

empirical modeling

- take N measurements, find mean (m) and st. dev. (σ) and then use a Gaussian model
- or, some other easily-manipulated model...

$$f(x) = \frac{e^{-(x-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

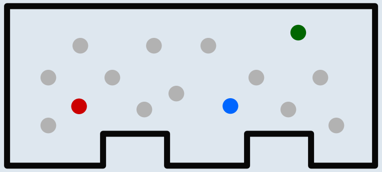
$$p(x) = \begin{cases} 0 & \text{if } |x-m| > \sigma \\ 1 & \text{otherwise} \end{cases}$$

$$p(x) = \begin{cases} 0 & \text{if } |x-m| > \sigma \\ 1- |x-m|/\sigma & \text{otherwise} \end{cases}$$

(2) Make something up...

Monte Carlo Localization

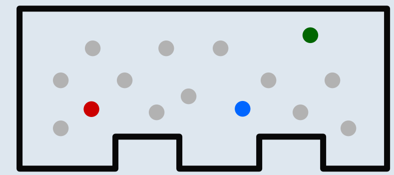
Start by assuming $p(r_0)$ is the uniform distribution.



take K samples of r_0 and weight each with an *importance factor* of $1/K$

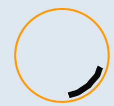
Monte Carlo Localization

Start by assuming $p(r_0)$ is the uniform distribution.

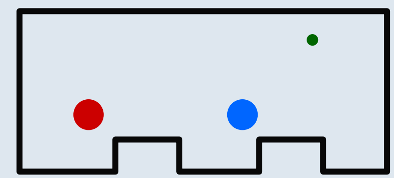


take K samples of r_0 and weight each with an *importance factor* of 1/K

Get the current sensor observation, o_1

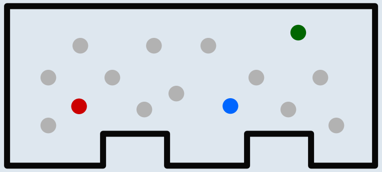


For each sample point r_0 multiply the importance factor by $p(o_1 \mid r_0, m)$



Monte Carlo Localization

Start by assuming $p(r_0)$ is the uniform distribution.

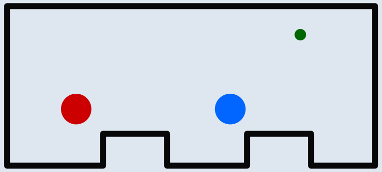


take K samples of r_0 and weight each with an *importance factor* of 1/K

Get the current sensor observation, o_1

For each sample point r_0 multiply the importance factor by $p(o_1 \mid r_0, m)$

Normalize (make sure the importance factors add to 1)

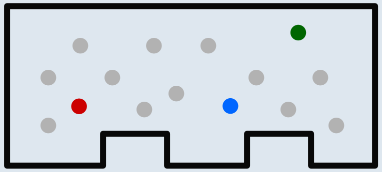


You now have an approximation of $p(r_1 \mid o_1, \ldots, m)$

and the distribution is no longer uniform

Monte Carlo Localization

Start by assuming $p(r_0)$ is the uniform distribution.

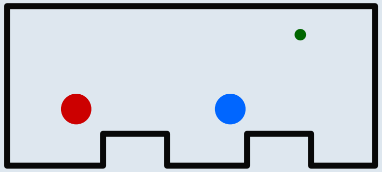


take K samples of r_0 and weight each with an *importance factor* of 1/K

Get the current sensor observation, o_1

For each sample point r_0 multiply the importance factor by $p(o_1 \mid r_0, m)$

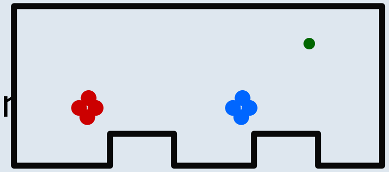
Normalize (make sure the importance factors add to 1)



You now have an approximation of $p(r_1 \mid o_1, \ldots, m)$

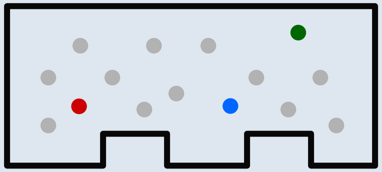
and the distribution is no longer uniform

Create r_1 samples by dividing up large clumps



each point spawns new ones in proportion to its importance factor

Monte Carlo Localization

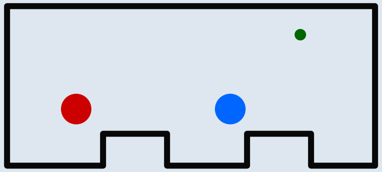


Start by assuming $p(r_0)$ is the uniform distribution.

take K samples of r_0 and weight each with an *importance factor* of 1/K

Get the current sensor observation, o_1

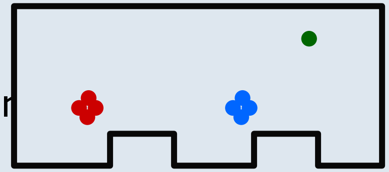
For each sample point r_0 multiply the importance factor by $p(o_1 \mid r_0, m)$



Normalize (make sure the importance factors add to 1)

You now have an approximation of $p(r_1 \mid o_1, \ldots, m)$

and the distribution is no longer uniform

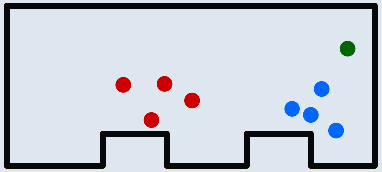


Create r_1 samples by dividing up large clumps

each point spawns new ones in proportion to its importance factor

The robot moves, a_1

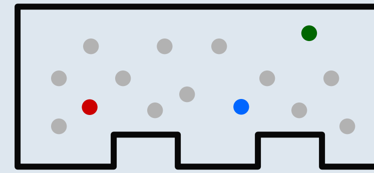
For each sample r_1, move it according to the model $p(r_2 \mid a_1, r_1, m)$



Monte Carlo Localization

Start by assuming $p(r_0)$ is the uniform distribution.

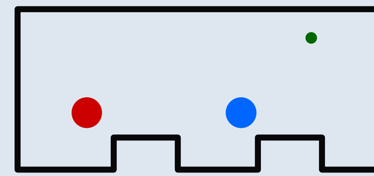
take K samples of r_0 and weight each with an *importance factor* of 1/K



Get the current sensor observation, o_1

For each sample point r_0 multiply the importance factor by $p(o_1 \mid r_0, m)$

Normalize (make sure the importance factors add to 1)

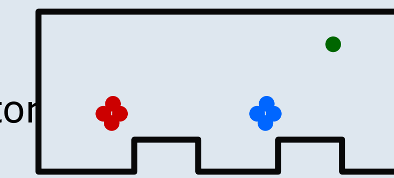


You now have an approximation of $p(r_1 \mid o_1, \ldots, m)$

and the distribution is no longer uniform

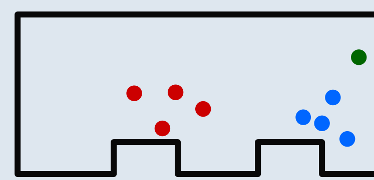
Create r_1 samples by dividing up large clumps

each point spawns new ones in proportion to its importance factor



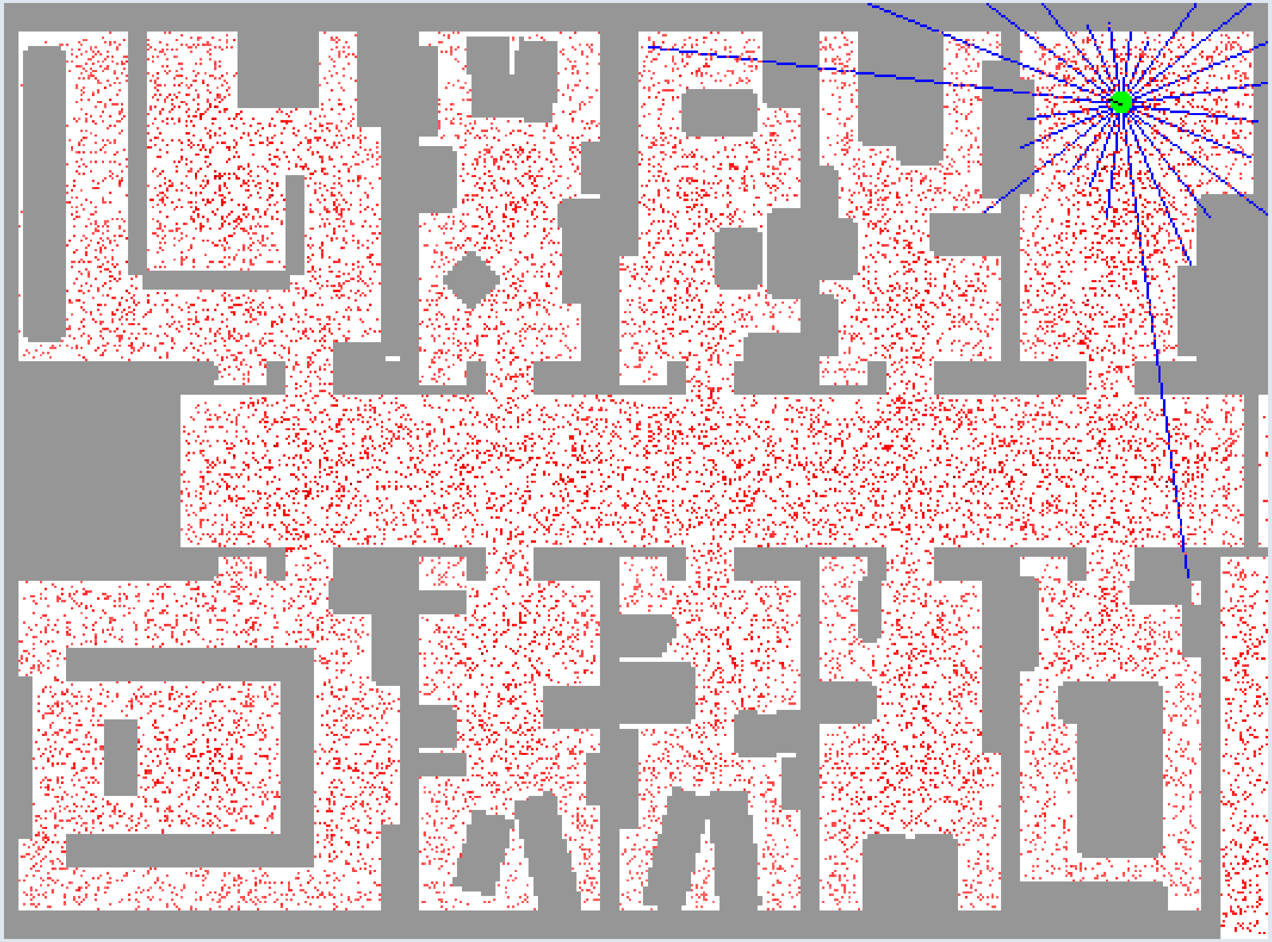
The robot moves, a_1

For each sample r_1, move it according to the model $p(r_2 \mid a_1, r_1, m)$



MCL in action

"Monte Carlo" Localization -- refers to the resampling of the distribution each time a new observation is integrated



References

1. Dieter Fox, Wolfram Burgard, Frank Dellaert, Sebastian Thrun, “Monte Carlo Localization: Efficient Position Estimation for Mobile Robots”, Proc. 16th National Conference on Artificial Intelligence, AAAI’99, July 1999

2. Dieter Fox, Wolfram Burgard, Sebastian Thrun, “Markov Localization for Mobile Robots in Dynamic Environments”, J. of Artificial Intelligence Research 11 (1999) 391-427

3. Sebastian Thrun, “Probabilistic Algorithms in Robotics”, Technical Report CMU-CS-00-126, School of Computer Science, Carnegie Mellon University, Pittsburgh, USA, 2000