Fundamental Problems In Robotics

- What does the world looks like? (mapping)
  - sense from various positions
  - integrate measurements to produce map
  - assumes perfect knowledge of position
- Where am I in the world? (localization)
  - Sense
  - relate sensor readings to a world model
  - compute location relative to model
  - assumes a perfect world model
- Together, these are SLAM (Simultaneous Localization and Mapping)

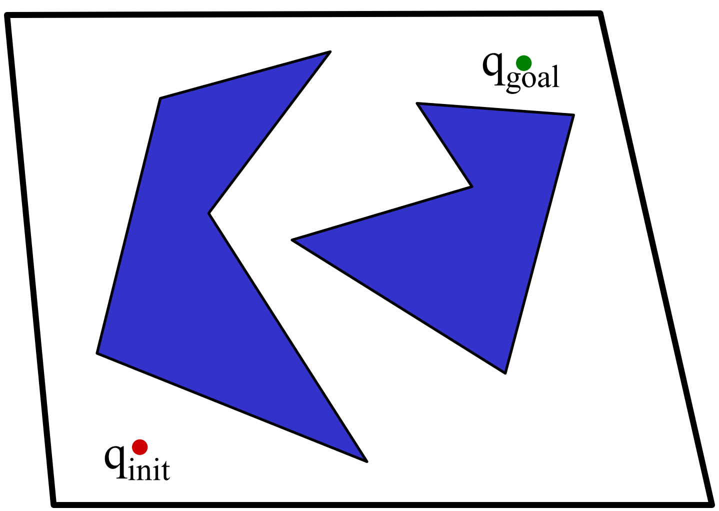
Path Planning

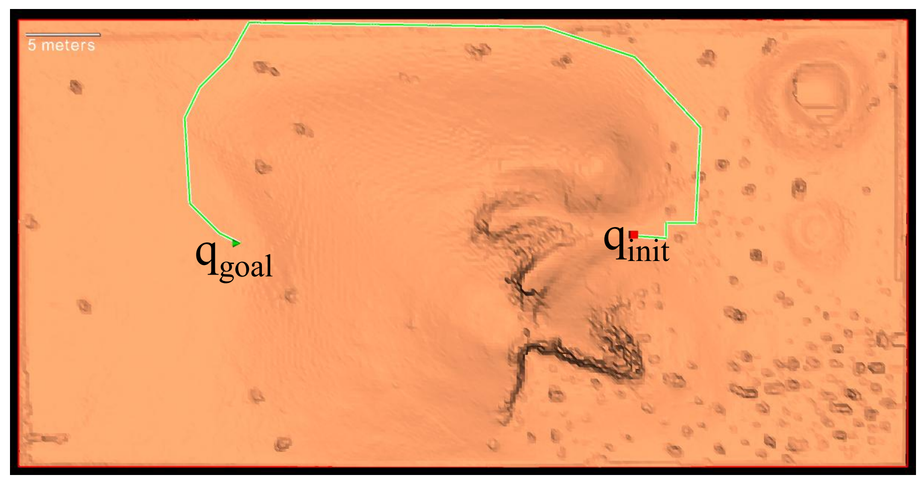
- Visibility Graph
- Bug Algorithms
- Potential Fields
- Skeletons/Voronoi Graphs
- C-Space
- PRM’s
- RRT’s

Motion Planning

- The ability to go from **A** to **B**
  - Known map – Off-line planning
  - Unknown Environment –Online planning
  - Static/Dynamic Environment

• q_{init} • q_{goal}





Path Planning

World

- Indoor/Outdoor
- 2D/2.5D/3D
- Static/Dynamic
- Known/Unknown
- Abstract (web)

Robot

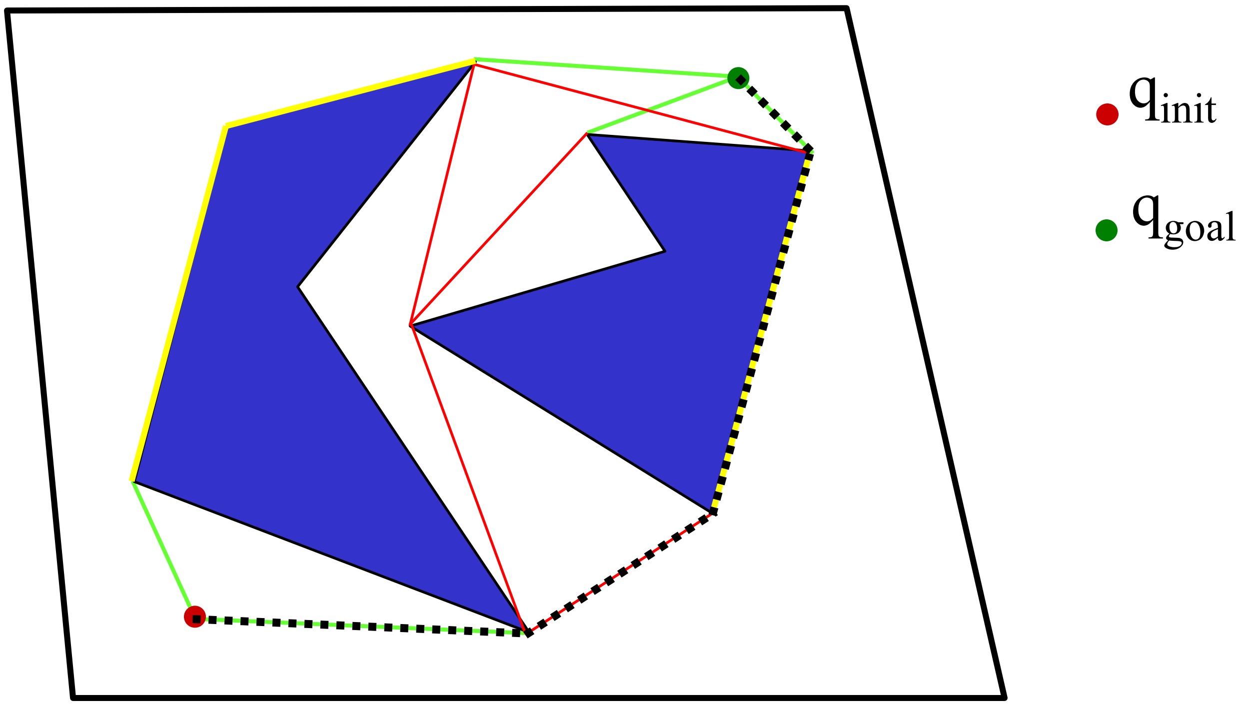
- Mobile
  - Indoor/Outdoor
  - Walking/Flying/Swimming
- Manipulator
- Humanoid
- Abstract

Map

- Topological
- Metric
- Feature Based
- 1D,2D,2.5D,3D

Visibility Graph

- Connect initial and goal locations with all the visible vertices
- Connect each obstacle vertex to every visible obstacle vertex
- Remove edges that intersect the interior of an obstacle
- Plan on the resulting graph

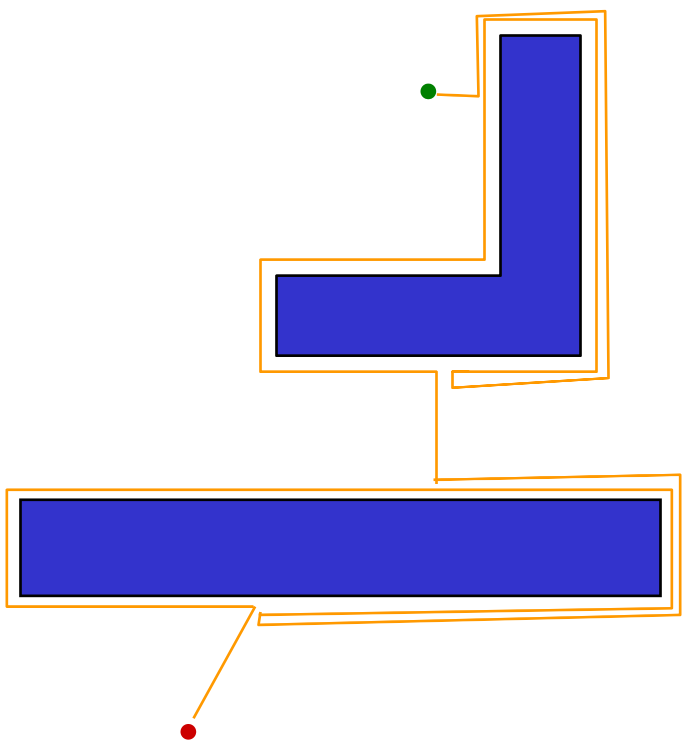


Bug 1

Insect-inspired "bug" algorithms

- known direction to goal
- otherwise only local sensing

walls/obstacles encoders



"Bug 1" algorithm

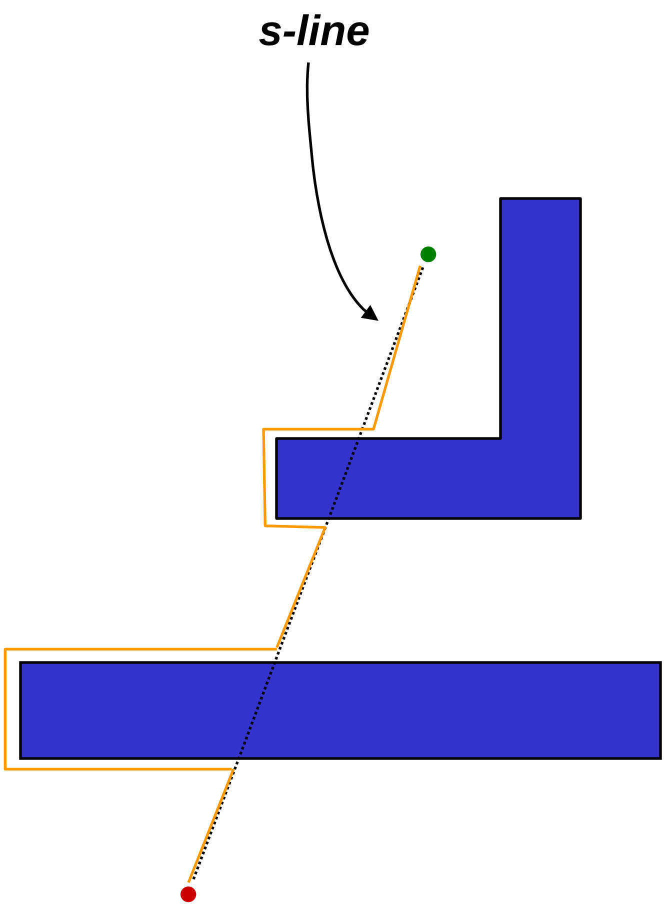
1) head toward goal

2) if an obstacle is encountered, circumnavigate it *and* remember how close you get to the goal

3) return to that closest point (by wall-following) and continue

Vladimir Lumelsky & Alexander Stepanov Algorithmica 1987

A better bug?



"Bug 2" algorithm

1) head toward goal on the *s-line*

2) if an obstacle is in the way, follow it until encountering the s-line again.

3) Leave the obstacle and continue toward the goal

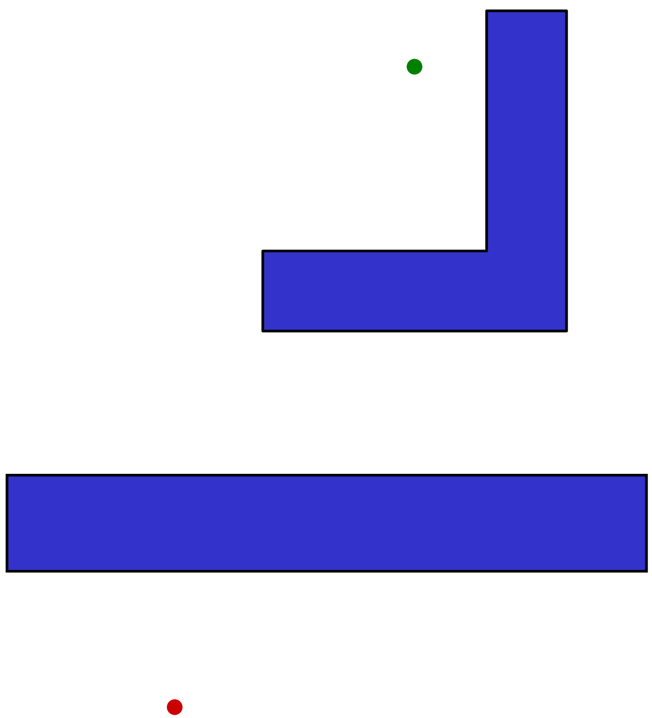
OK ?

head-to-head comparison

or thorax-to-thorax, perhaps

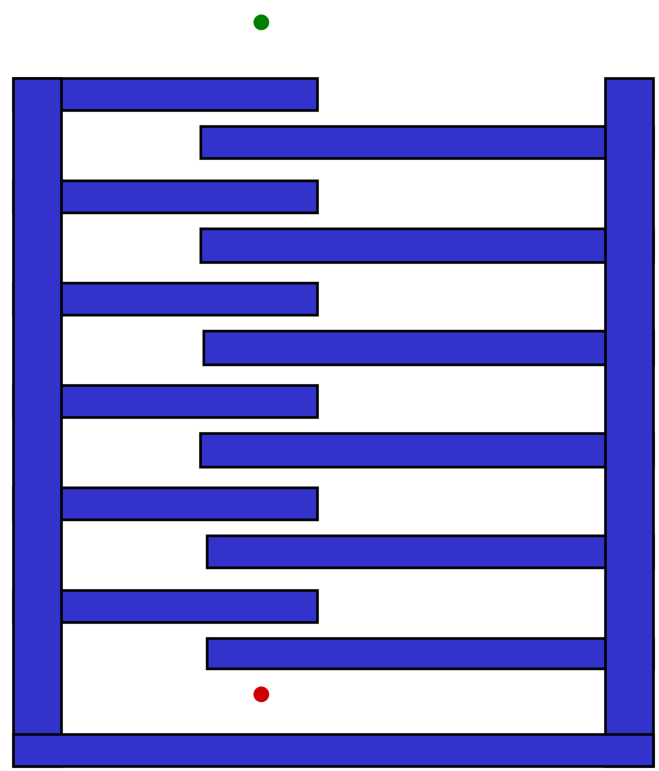
What are worlds in which Bug 2 does better than Bug 1 (and vice versa) ?

Bug 2 beats Bug 1

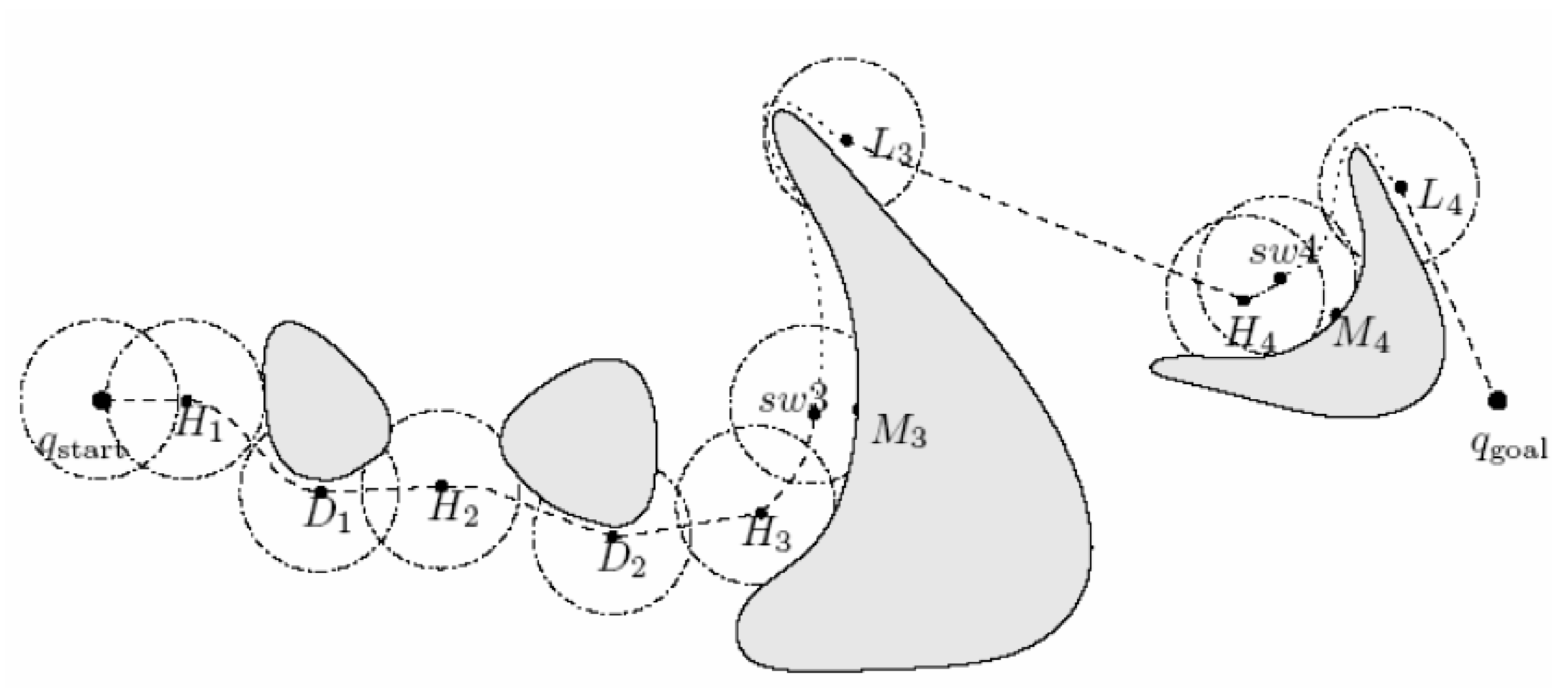


Bug 1 beats Bug 2

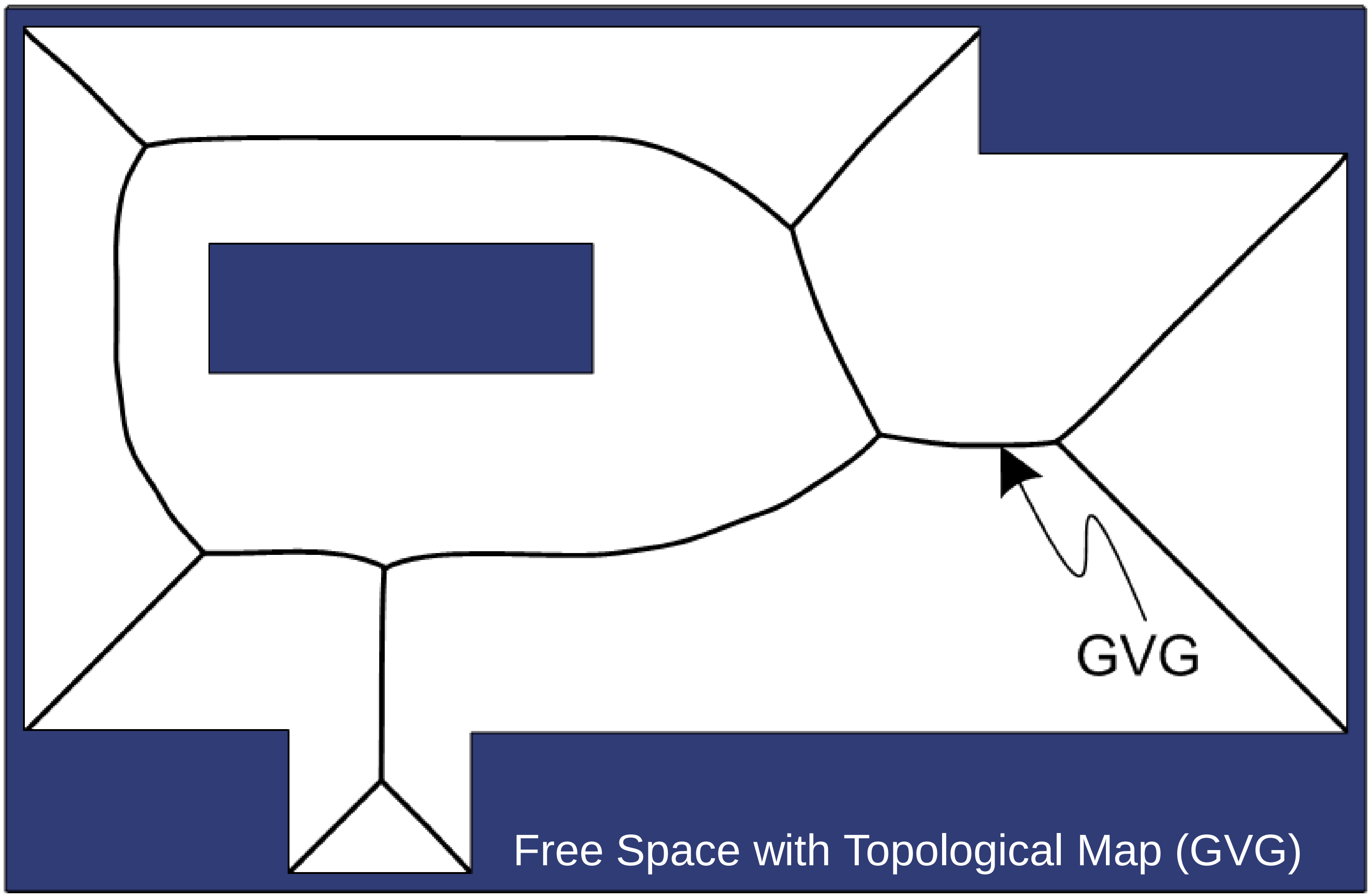
"zipper world"



Limited Sensor Range Tangent-Bug

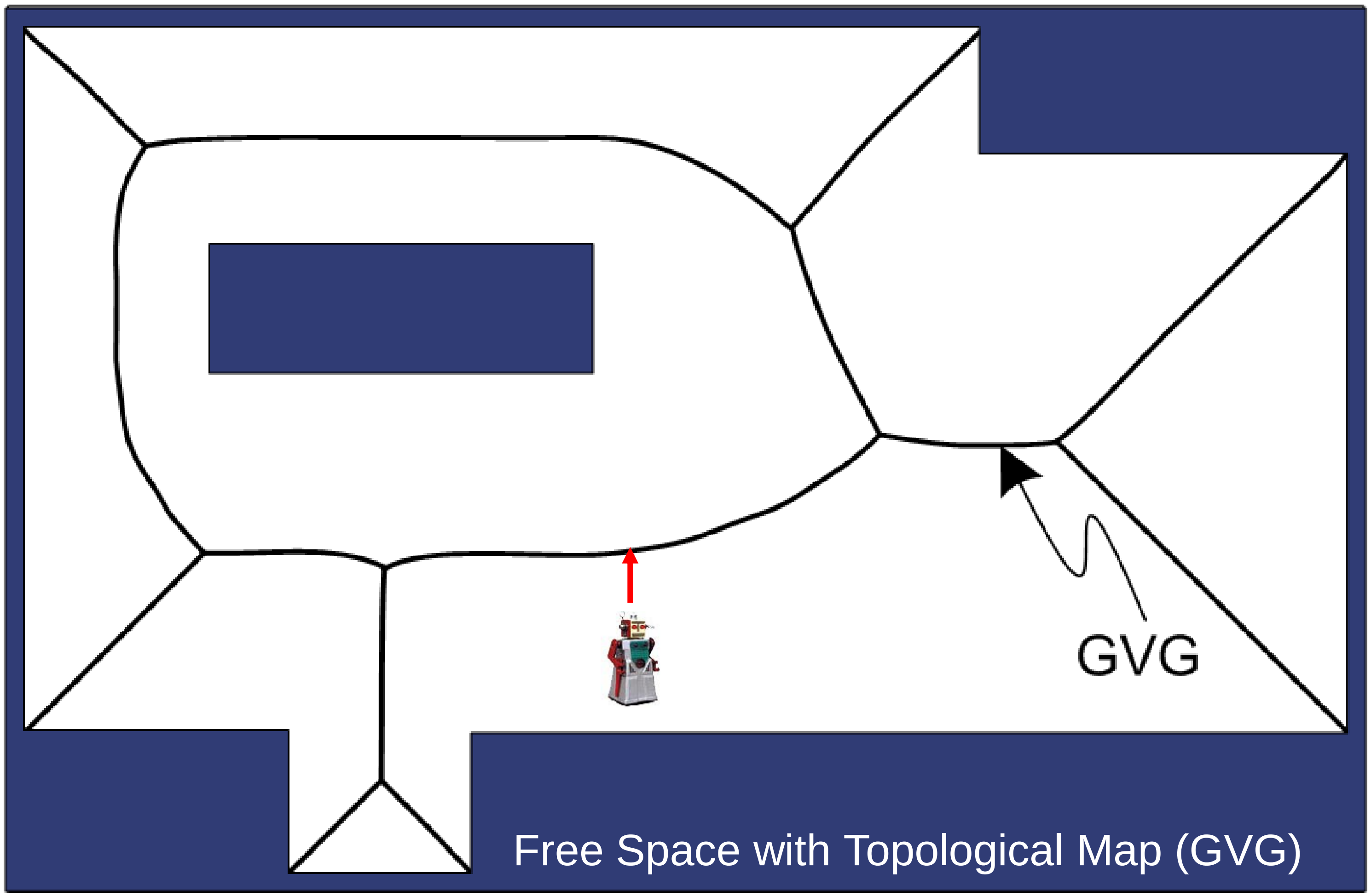


Generalized Voronoi Graph (GVG)



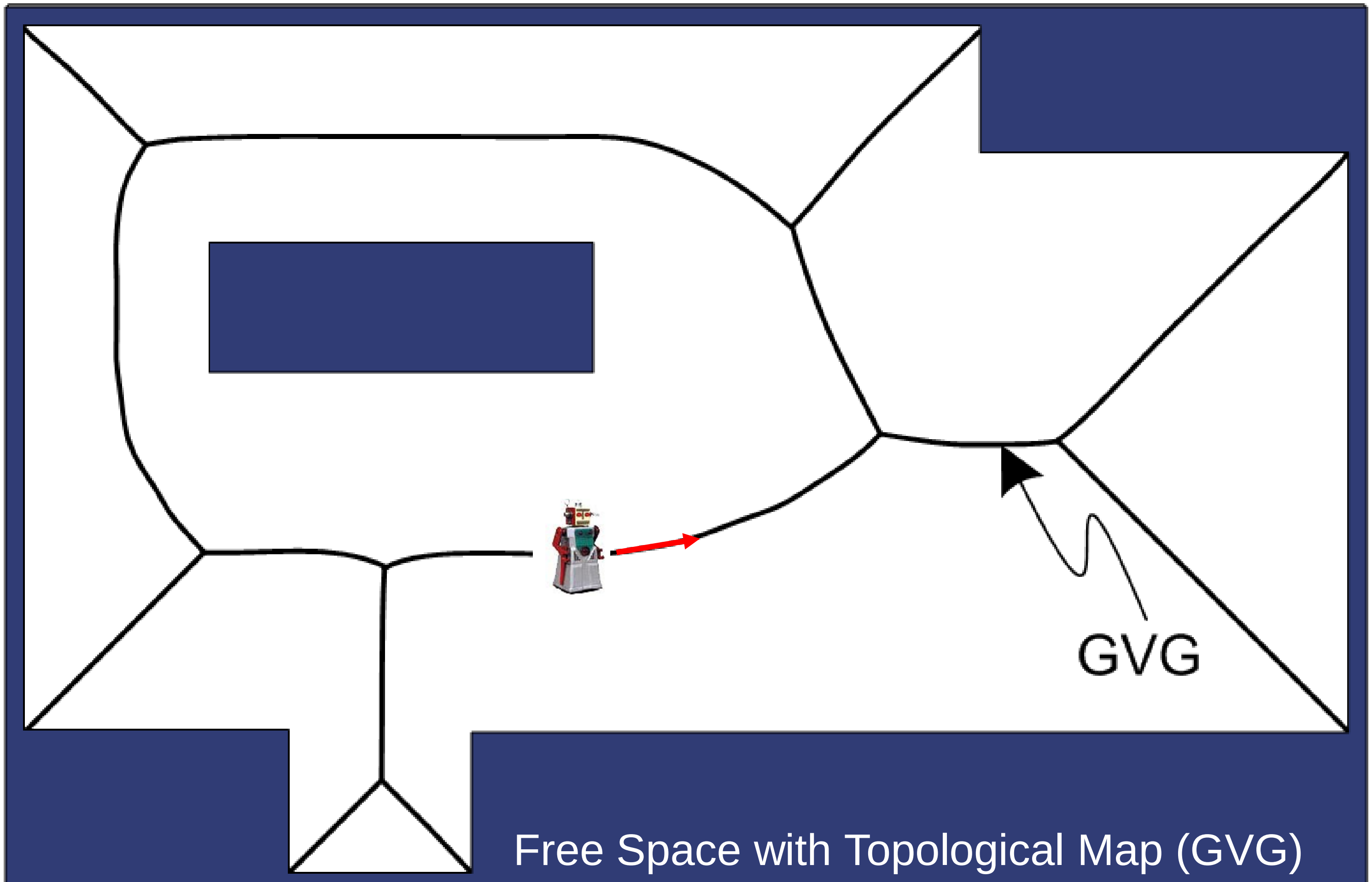
Generalized Voronoi Graph (GVG)

- Access GVG



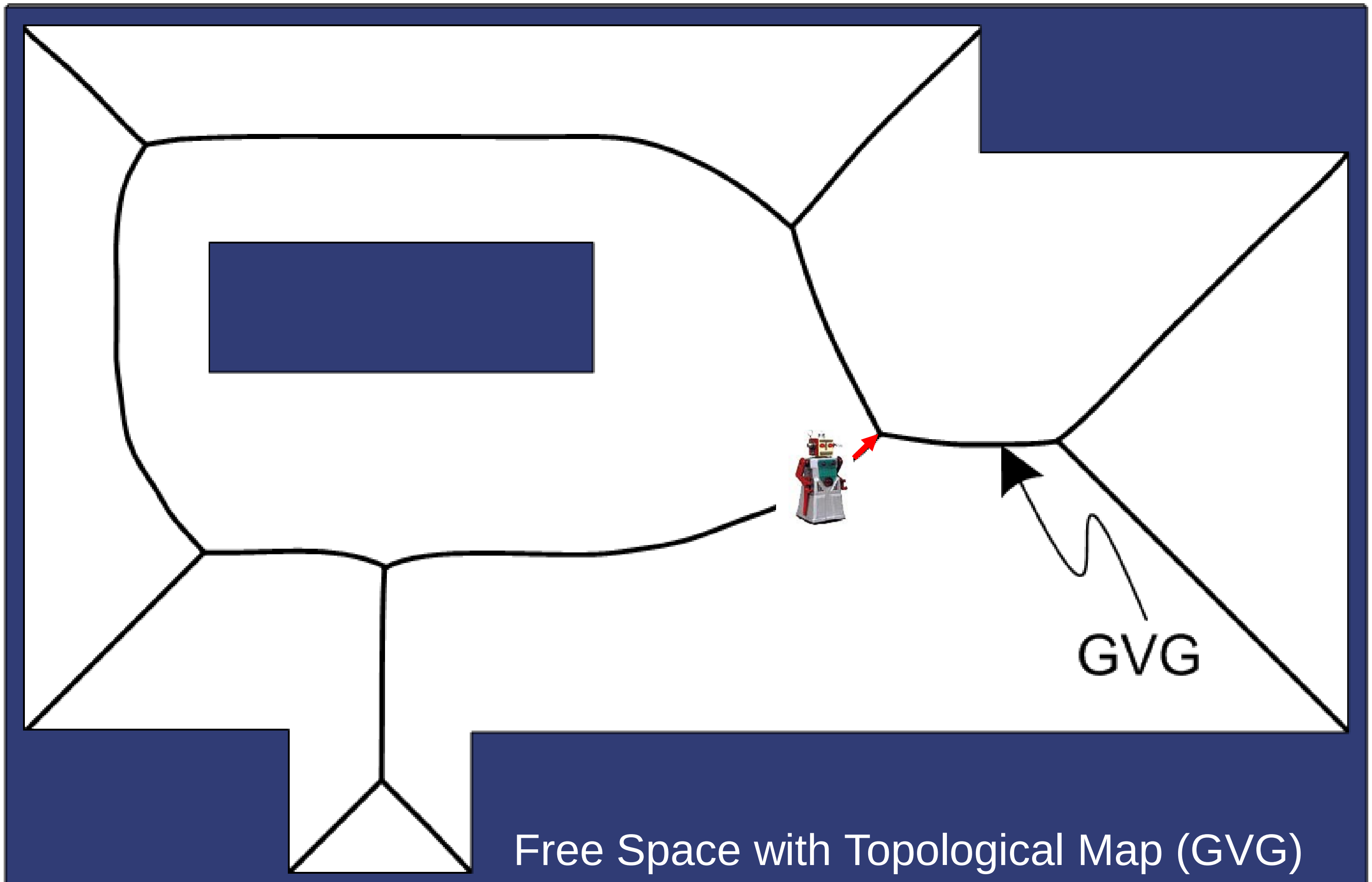
Generalized Voronoi Graph (GVG)

- Access GVG
- Follow Edge



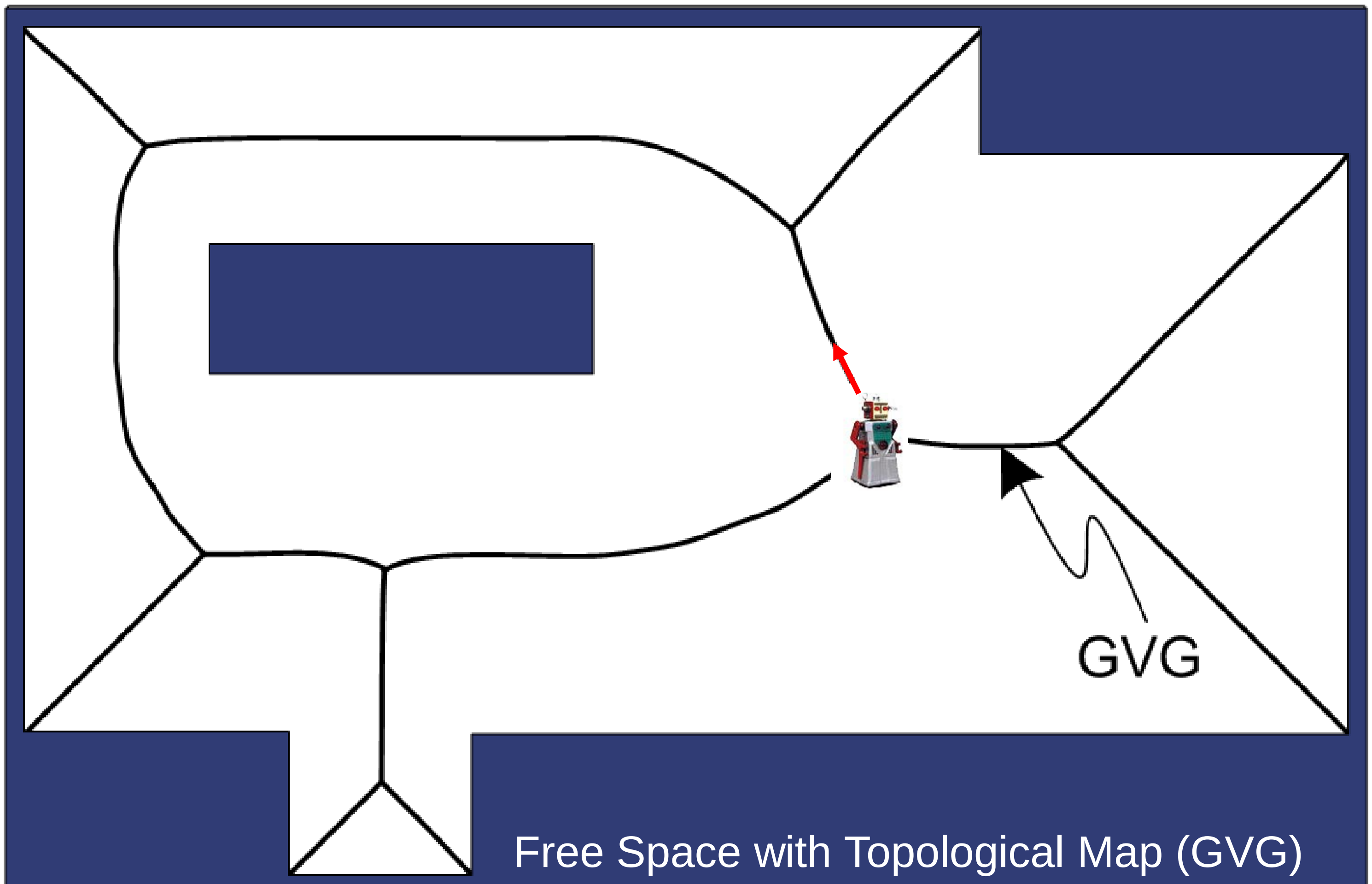
Generalized Voronoi Graph (GVG)

- Access GVG
- Home to the MeetPoint
- Follow Edge



Generalized Voronoi Graph (GVG)

- Access GVG
- Follow Edge
- Home to the MeetPoint
- Select Edge

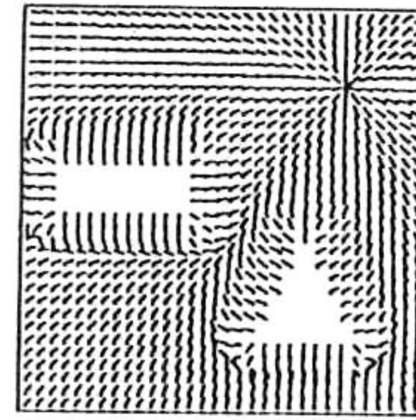


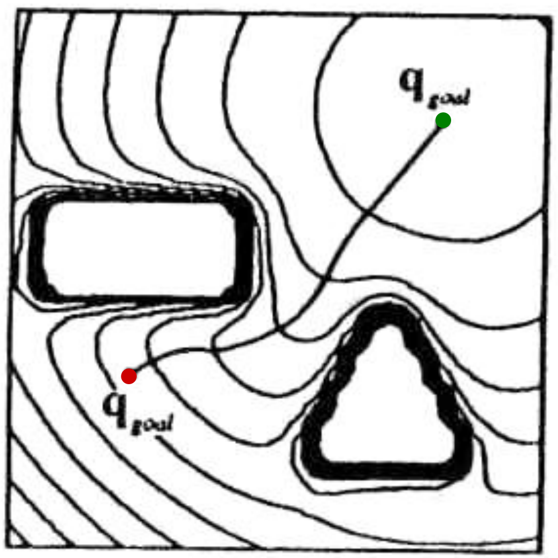
Local techniques

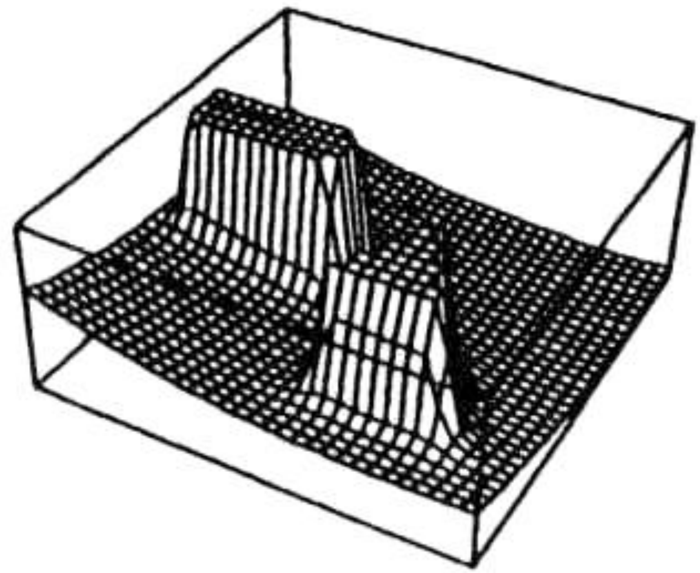
Potential Field methods

- compute a repulsive force away from obstacles
- compute an attractive force toward the goal

→ let the sum of the forces control the robot



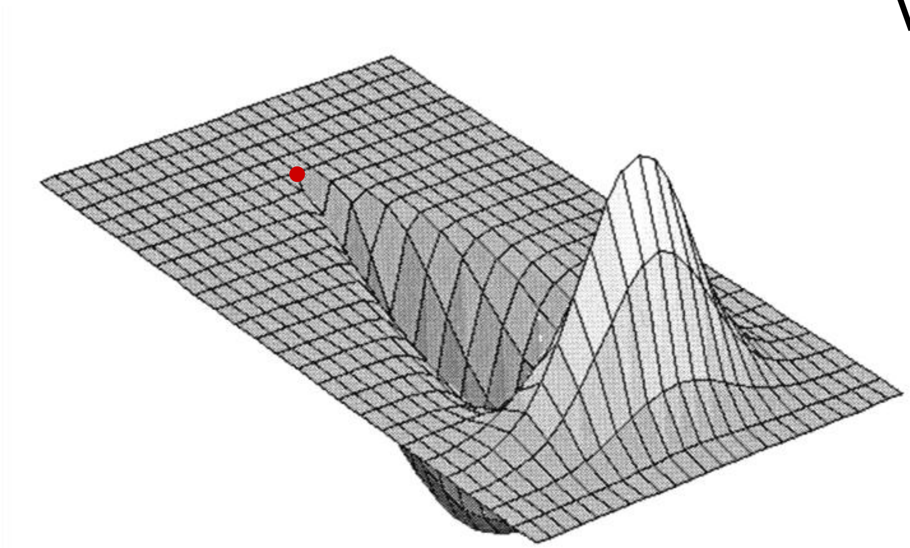




To a large extent, this is computable from sensor readings

SONAR modeling using Occupancy Grids

- The key to making accurate maps is combining lots of data.
- But combining these numbers means we have to know what they are !



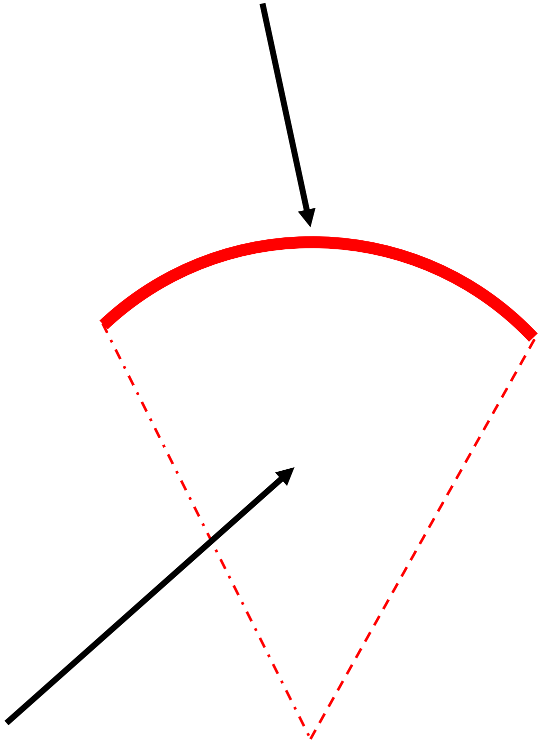
what is in each cell of this sonar model / map ?

What should our map contain ?

- small cells
- each represents a bit of the robot's environment
- larger values => obstacle
- smaller values => free

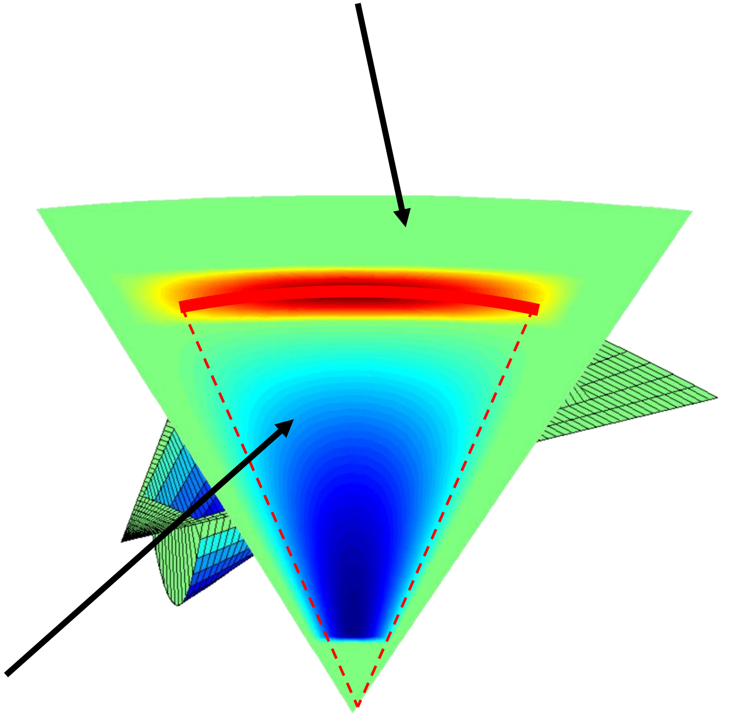
Arc Carving Sonar Model

- Represents a sonar return as a cone with an arc base
  - The arc approximates the sonar response
  - The interior of the cone represents a region of likely freespace

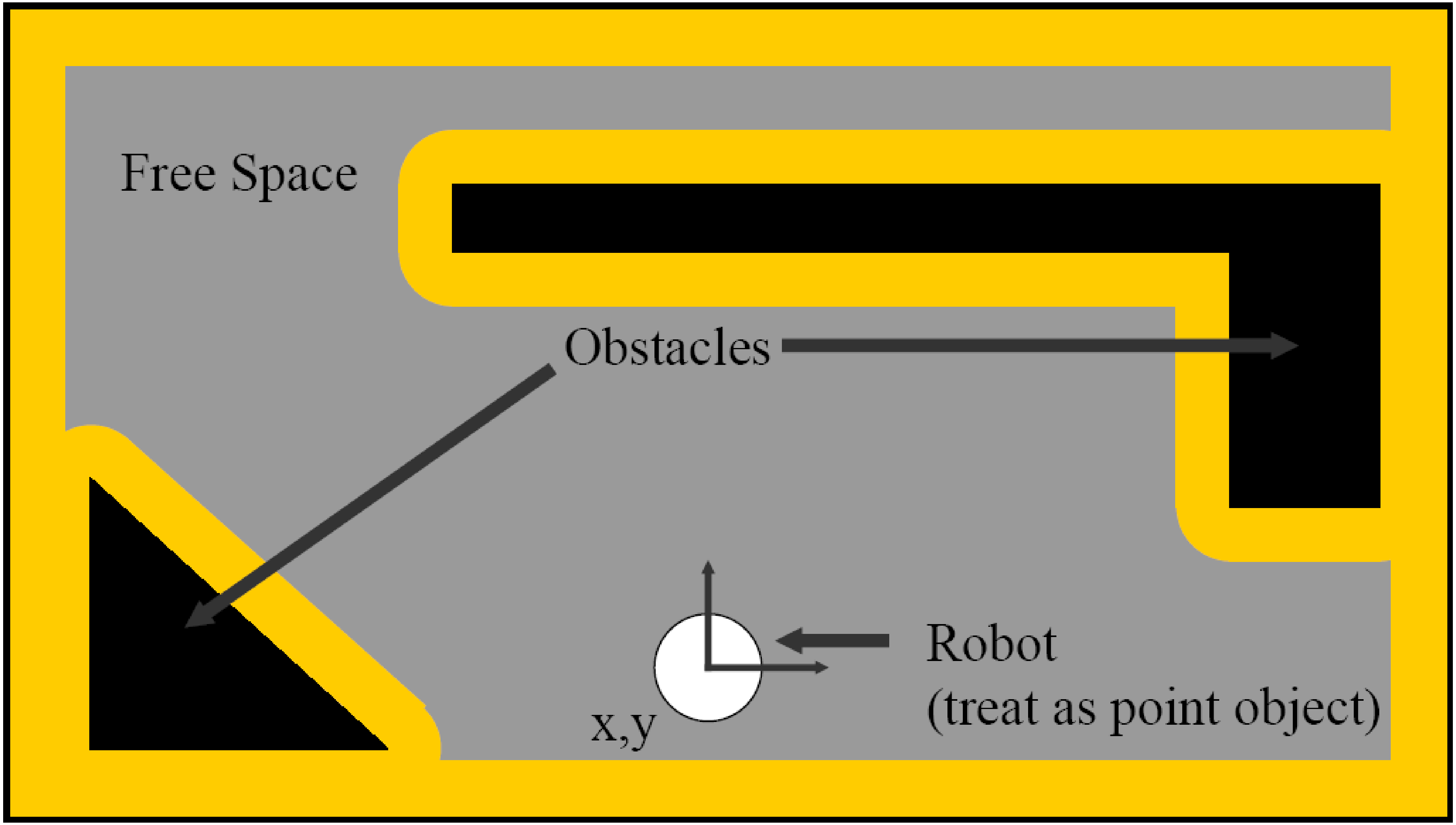


Occupancy Grid Sonar Model

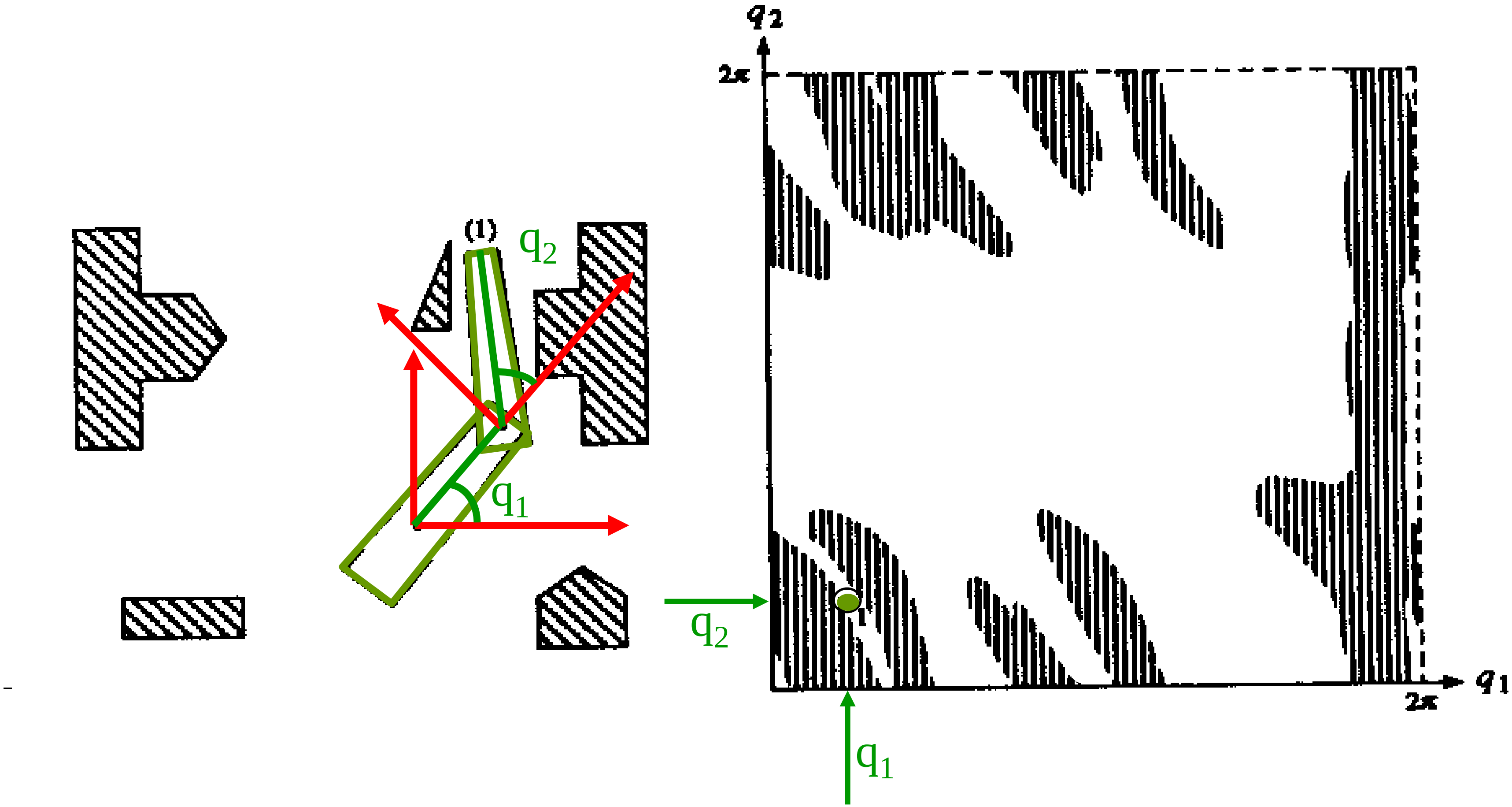
- The arc carving model may be viewed as a binary approximation of the model used by Moravec and Elfes
  - An Arc with nonzero probability of occupancy
  - A cone with nonzero probability of freespace



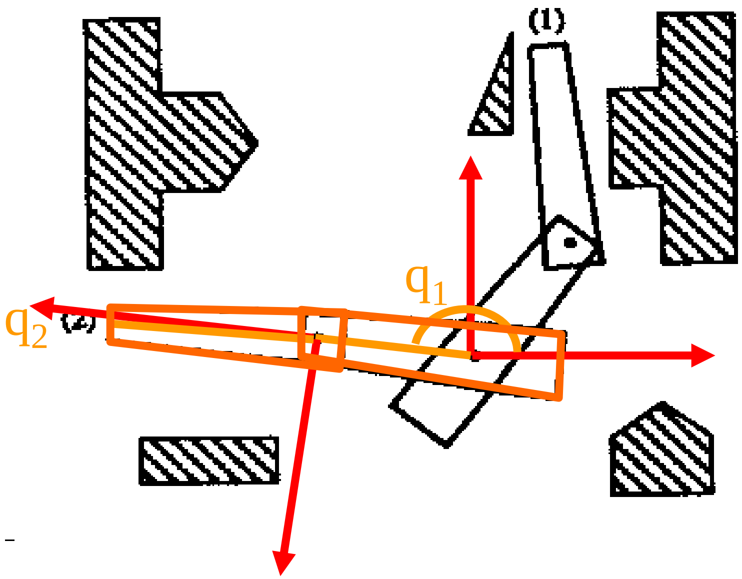
Configuration Space

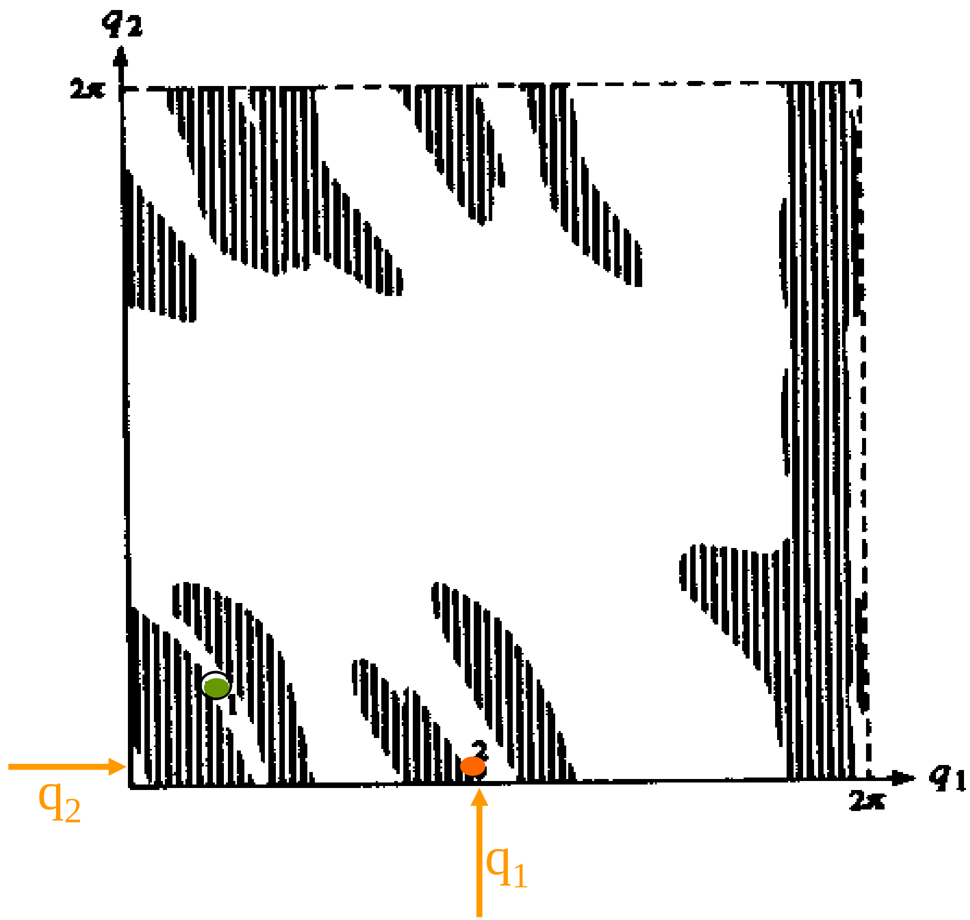


Tool: Configuration Space (C-Space C)

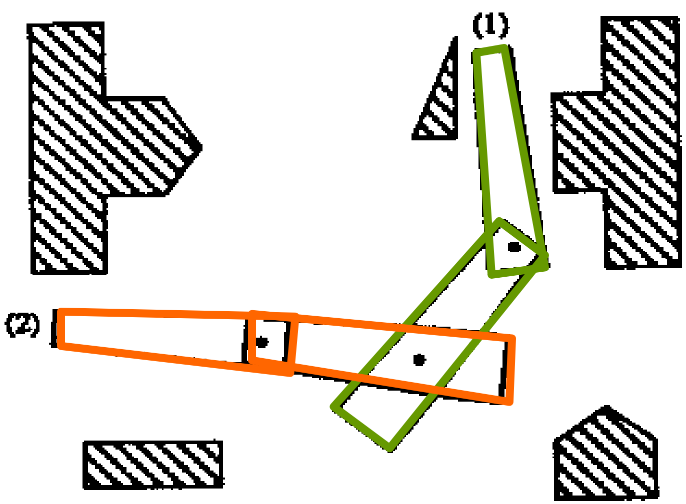


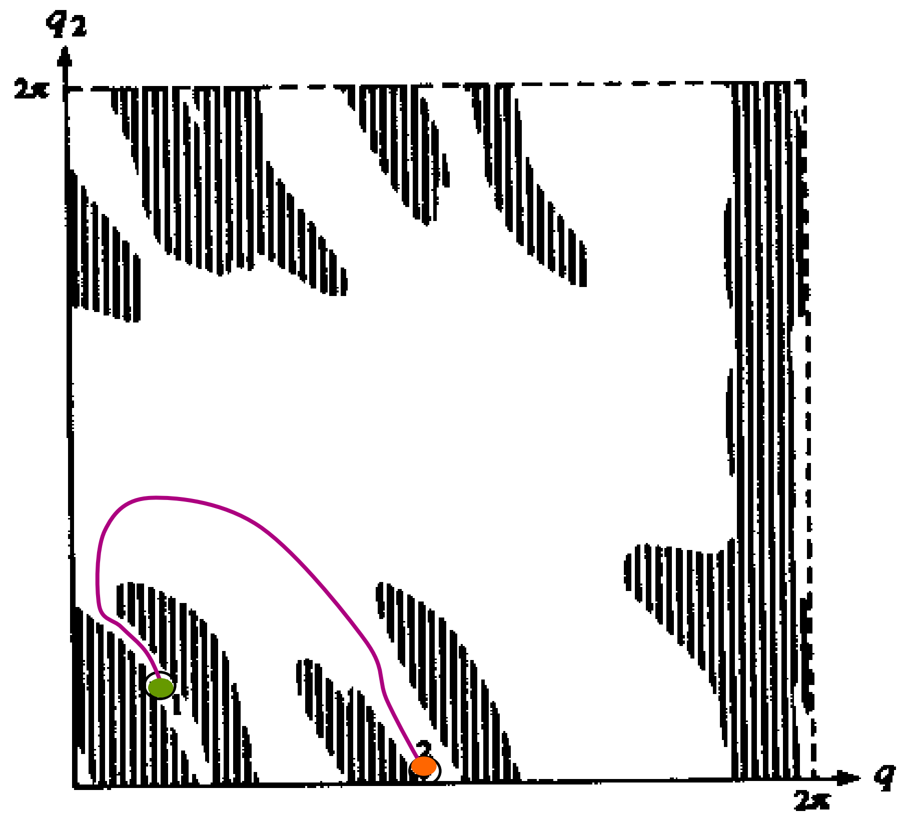
Tool: Configuration Space (C-Space C)





Tool: Configuration Space (C-Space C)

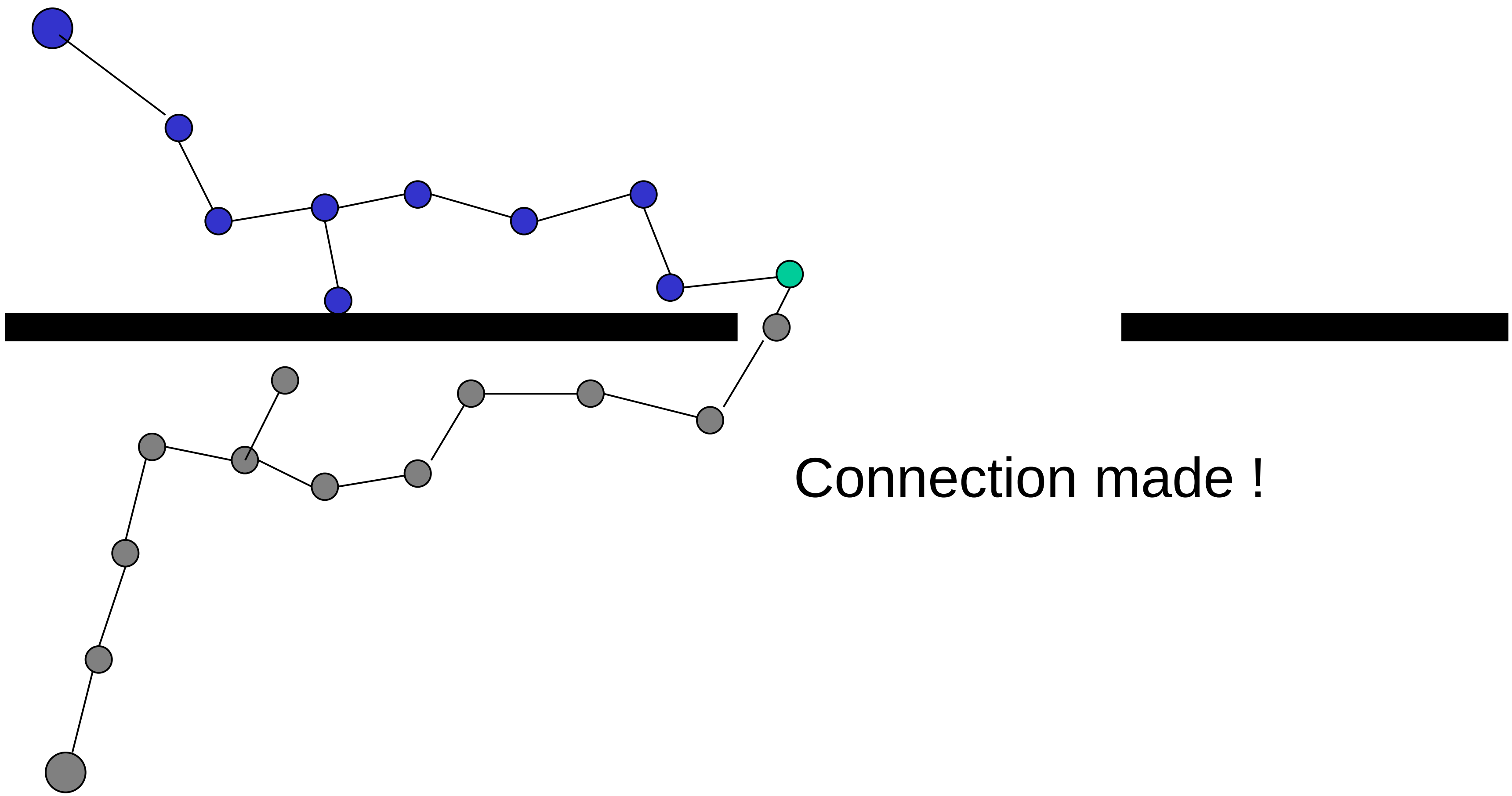




Road Maps

- PRMs
- RRTs

RRT-Connect: example

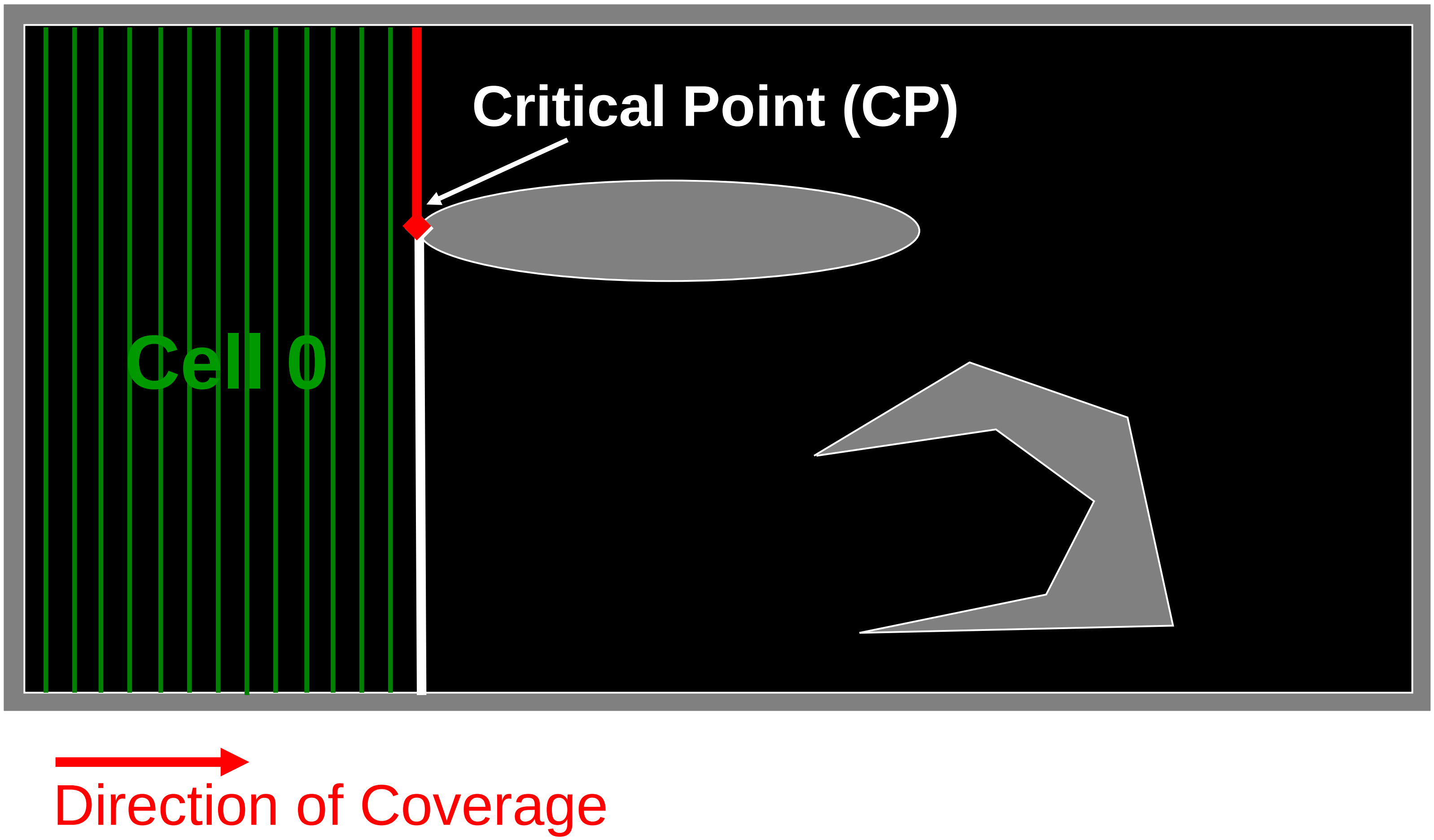


Connection made !

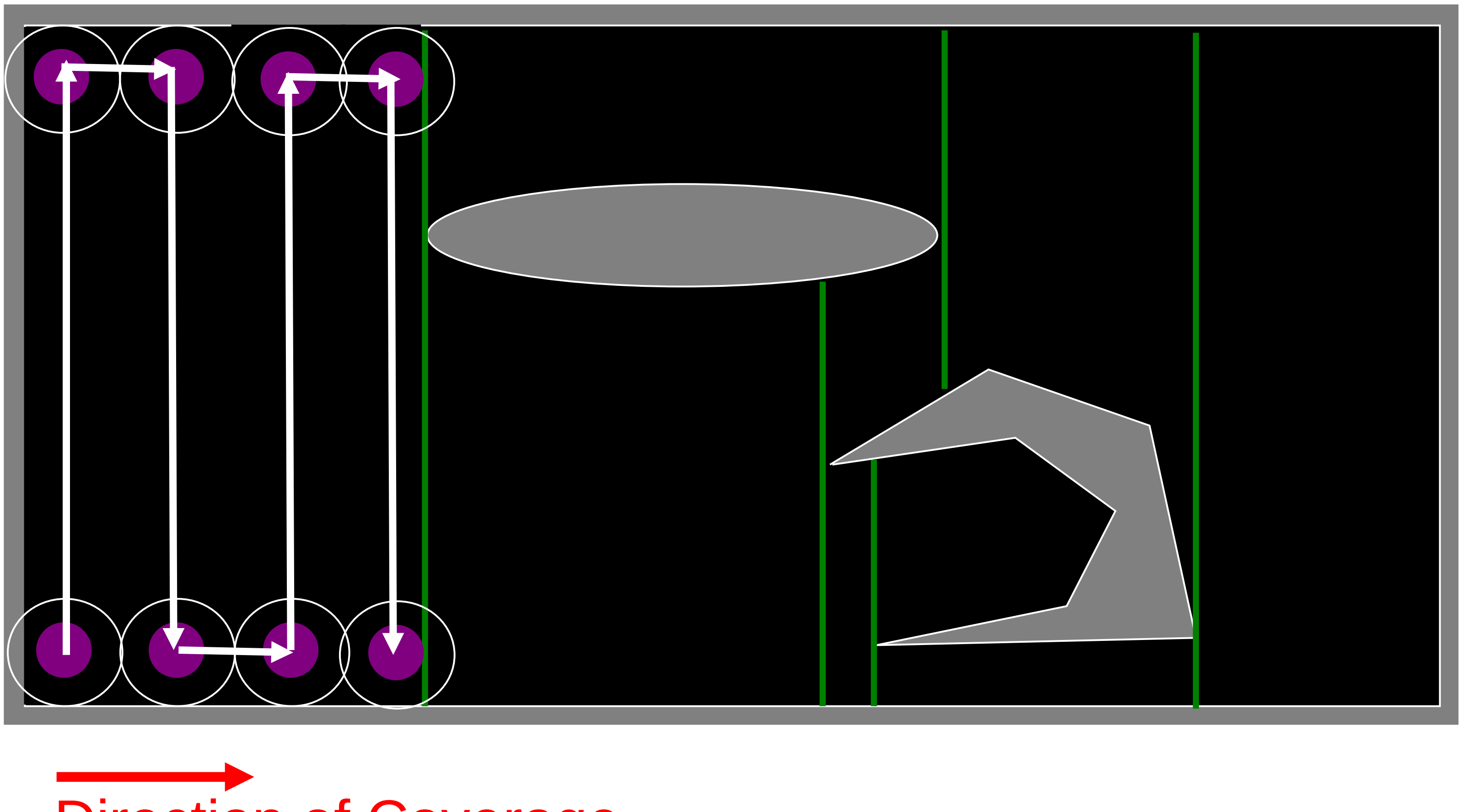
Coverage

- First Distinction
  - Deterministic **Demining**
  - Random **Vacuum Cleaning**
- Second Distinction
  - Complete
  - No Guarantee
- Third Distinction
  - Known Environment
  - Unknown Environment

Cellular Decomposition

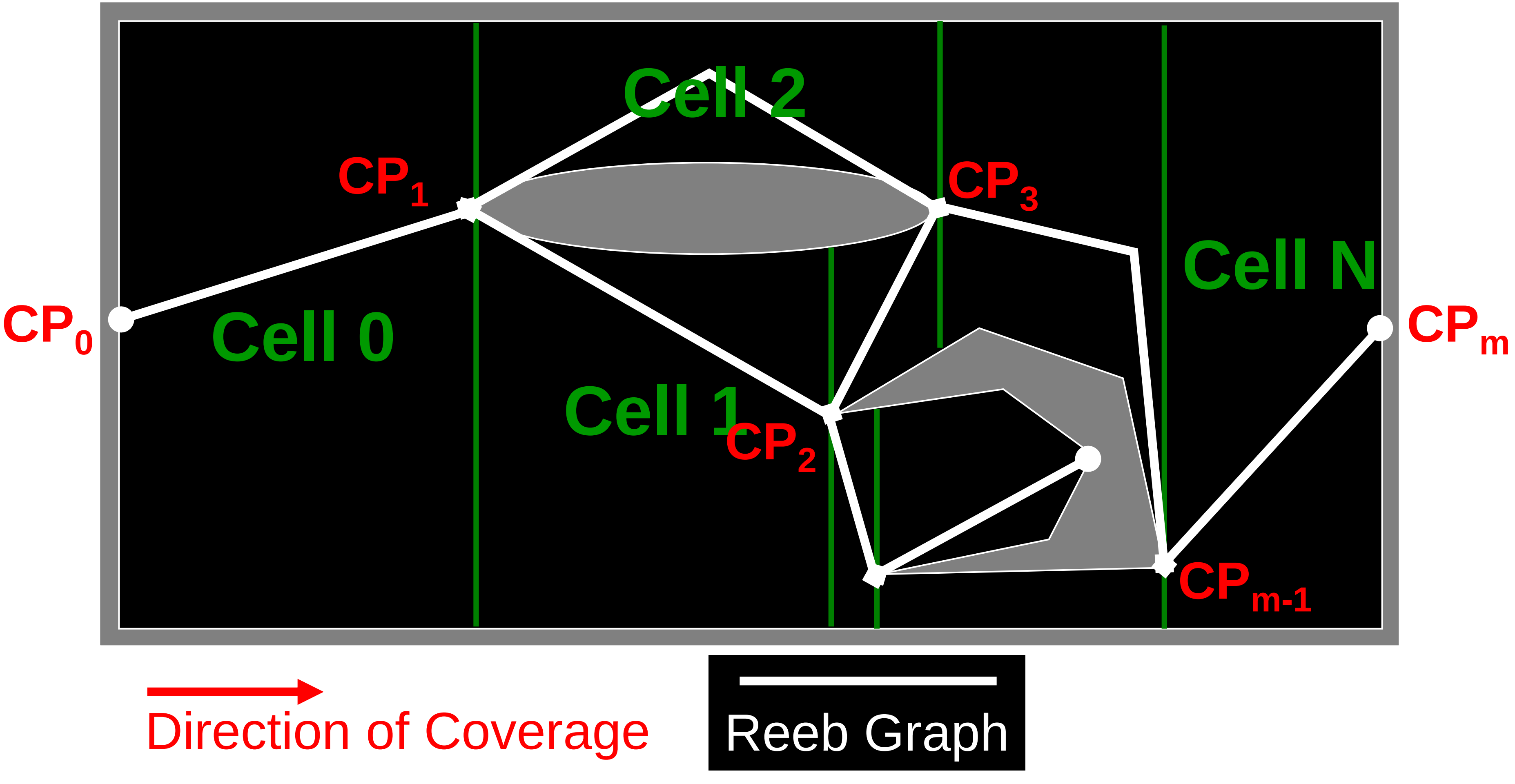


Single Cell Coverage



Direction of Coverage

Cellular Decomposition



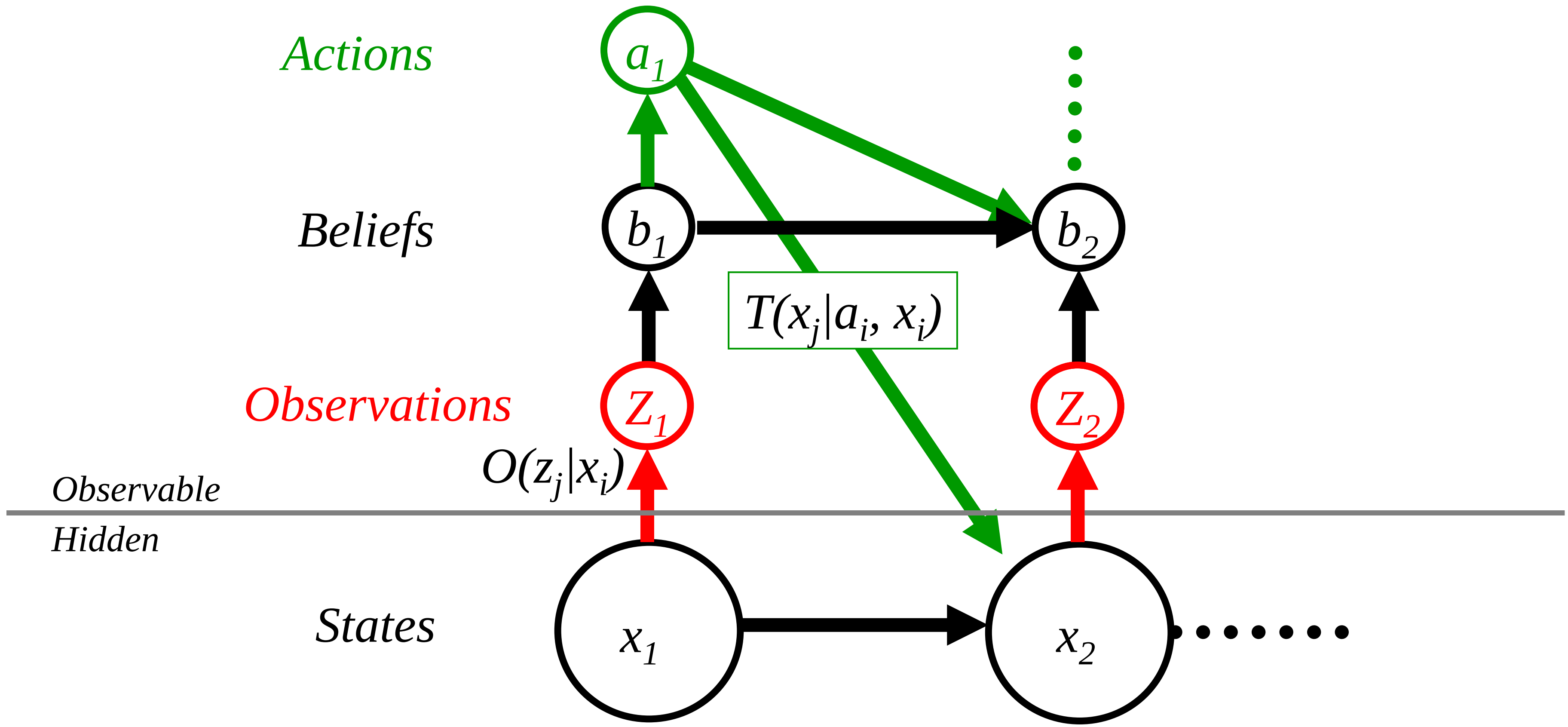
Multi-Robot Coverage

- Team based
- Distributed
  - Auctions

Localization

- Tracking: Known initial position
- Global Localization: Unknown initial position
- Re-Localization: Incorrect known position
  - (kidnapped robot problem)

Graphical Models, Bayes' Rule and the Markov Assumption



$$\text{Bayes rule}: p(x \mid y) = \frac{p(y \mid x)\, p(x)}{p(y)}$$

$$\text{Markov}: p(x_t \mid x_{t-1}, a_t, a_0, z_0, a_1, z_1, \ldots, z_{t-1}) = p(x_t \mid x_{t-1}, a_t)$$

Derivation of the Bayesian Filter

First-order Markov assumption shortens middle term:

$$Bel(x_t) = \eta \boxed{p(o_t \mid x_t)} \int p(x_t \mid x_{t-1}, a_{t-1}) p(x_{t-1} \mid a_{t-1}, \ldots, o_0) dx_{t-1}$$

Finally, substituting the definition of *$Bel(x_{t-1})$:*

$$Bel(x_t) = \eta p(o_t \mid x_t) \int p(x_t \mid x_{t-1}, a_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

The above is the probability distribution that must be estimated from the robot's data

Iterating the Bayesian Filter

- Propagate the motion model:

$$Bel\_(x_t) = \int P(x_t \mid a_{t-1}, x_{t-1}) Bel(x_{t-1}) dx_{t-1}$$

Compute the current state estimate before taking a sensor reading by integrating over all possible previous state estimates and applying the motion model

- Update the sensor model:

$$Bel(x_t) = \eta P(o_t \mid x_t) Bel\_(x_t)$$

Compute the current state estimate by taking a sensor reading and multiplying by the current estimate based on the most recent motion history

Different Approaches

Kalman filters (late-60s?)
- Gaussians
- approximately linear models
- position tracking

Extended Kalman Filter
Information Filter
Unscented Kalman Filter

Multi-hypothesis ('00)
- Mixture of Gaussians
- Multiple Kalman filters
- Global localization, recovery

Discrete approaches ('95)
- Topological representation ('95)
- uncertainty handling (POMDPs)
- occas. global localization, recovery
- Grid-based, metric representation ('96)
- global localization, recovery

Particle filters ('98)
- Condensation (Isard and Blake '98)
- Sample-based representation
- Global localization, recovery
- Rao-Blackwellized Particle Filter

The Kalman Filter

- Motion model is Gaussian…
- Sensor model is Gaussian…
- Each belief function is uniquely characterized by its mean μ and covariance matrix Σ
- Computing the posterior means computing a new mean μ and covariance Σ from old data using actions and sensor readings
- *What are the key limitations?*

1) Unimodal distribution
2) Linear assumptions

What we know…
What we don't know…

- We know what the control inputs of our process are
  - We know what we've told the system to do and have a model for what the expected output should be if everything works right
- We don't know what the noise in the system truly is
  - We can only estimate what the noise might be and try to put some sort of upper bound on it
- When estimating the state of a system, we try to find a set of values that comes as close to the truth as possible
  - There will always be some mismatch between our estimate of the system and the true state of the system itself. We just try to figure out how much mismatch there is and try to get the best estimate possible

Kalman Filter Components

(also known as: Way Too Many Variables…)

Linear discrete time dynamic system (motion model)

State — Control input — Process noise

$$x_{t+1} = F_t x_t + B_t u_t + G_t w_t$$

State transition function — Control input function — Noise input function with covariance Q

Measurement equation (sensor model)

Sensor reading — State — Sensor noise with covariance R

$$z_{t+1} = H_{t+1} x_{t+1} + n_{t+1}$$

Sensor function

Note: Write these down!!!

Computing the MMSE Estimate of the State and Covariance

What is the **minimum mean square error** estimate of the system state and covariance?

$$\hat{x}_{t+1|t} = F_t \hat{x}_{t|t} + B_t u_t$$ Estimate of the state variables

$$\hat{z}_{t+1|t} = H_{t+1} \hat{x}_{t+1|t}$$ Estimate of the sensor reading

$$P_{t+1|t} = F_t P_{t|t} F_t^T + G_t Q_t G_t^T$$ Covariance matrix for the state

$$S_{t+1|t} = H_{t+1} P_{t+1|t} H_{t+1}^T + R_{t+1}$$ Covariance matrix for the sensors

The Kalman Filter…

Propagation (motion model):

$$\hat{x}_{t+1/t} = F_t \hat{x}_{t/t} + B_t u_t$$

$$P_{t+1/t} = F_t P_{t/t} F_t^T + G_t Q_t G_t^T$$

Update (sensor model):

$$\hat{z}_{t+1} = H_{t+1} \hat{x}_{t+1/t}$$

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1}$$

$$S_{t+1} = H_{t+1} P_{t+1/t} H_{t+1}^T + R_{t+1}$$

$$K_{t+1} = P_{t+1/t} H_{t+1}^T S_{t+1}^{-1}$$

$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$$

$$P_{t+1/t+1} = P_{t+1/t} - P_{t+1/t} H_{t+1}^T S_{t+1}^{-1} H_{t+1} P_{t+1/t}$$

…but what does that mean in English?!?

Propagation (motion model):

$$\hat{x}_{t+1/t} = F_t \hat{x}_{t/t} + B_t u_t$$

- State estimate is updated from system dynamics

$$P_{t+1/t} = F_t P_{t/t} F_t^{\,T} + G_t Q_t G_t^{\,T}$$

- Uncertainty estimate *GROWS*

Update (sensor model):

$$\hat{z}_{t+1} = H_{t+1} \hat{x}_{t+1/t}$$

- Compute expected value of sensor reading

$$r_{t+1} = z_{t+1} - \hat{z}_{t+1}$$

- Compute the difference between expected and “true”

$$S_{t+1} = H_{t+1} P_{t+1/t} H_{t+1}^{\,T} + R_{t+1}$$

- Compute covariance of sensor reading

$$K_{t+1} = P_{t+1/t} H_{t+1}^{\,T} S_{t+1}^{\,-1}$$

- Compute the Kalman Gain (how much to correct est.)

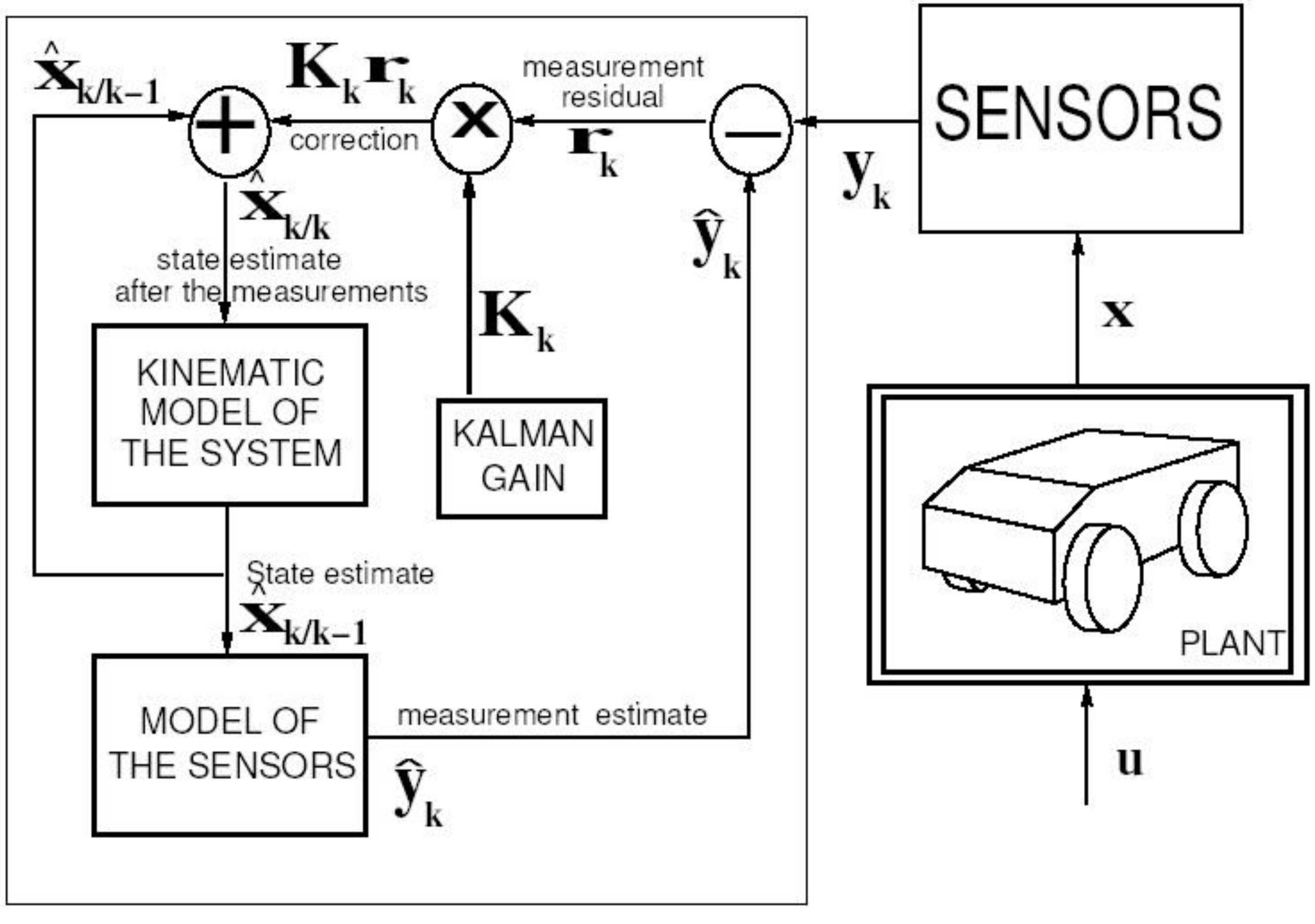
$$\hat{x}_{t+1/t+1} = \hat{x}_{t+1/t} + K_{t+1} r_{t+1}$$

- Multiply residual times gain to correct state estimate

$$P_{t+1/t+1} = P_{t+1/t} - P_{t+1/t} H_{t+1}^{\,T} S_{t+1}^{\,-1} H_{t+1} P_{t+1/t}$$

- Uncertainty estimate SHRINKS

Kalman Filter Block Diagram

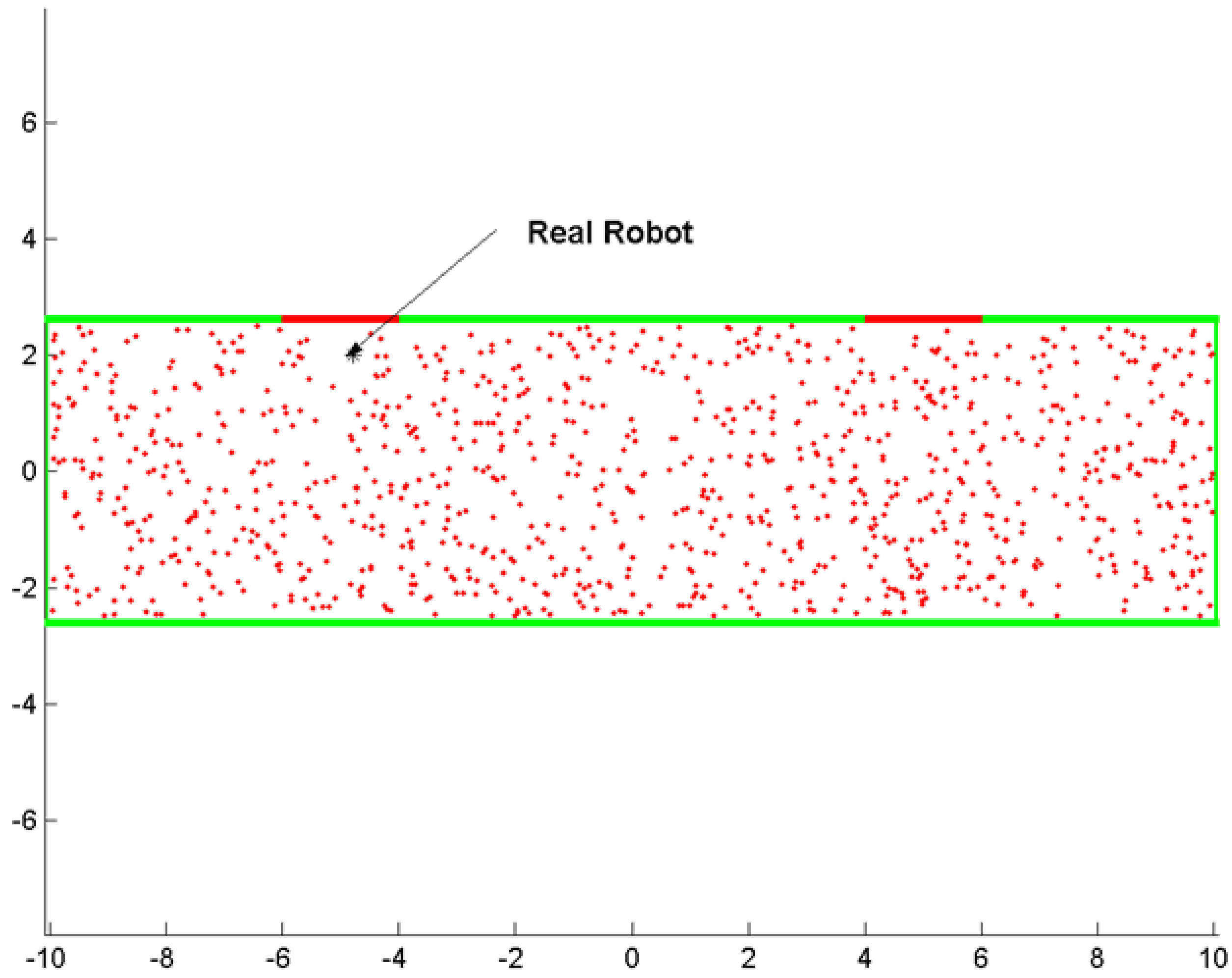


Some observations

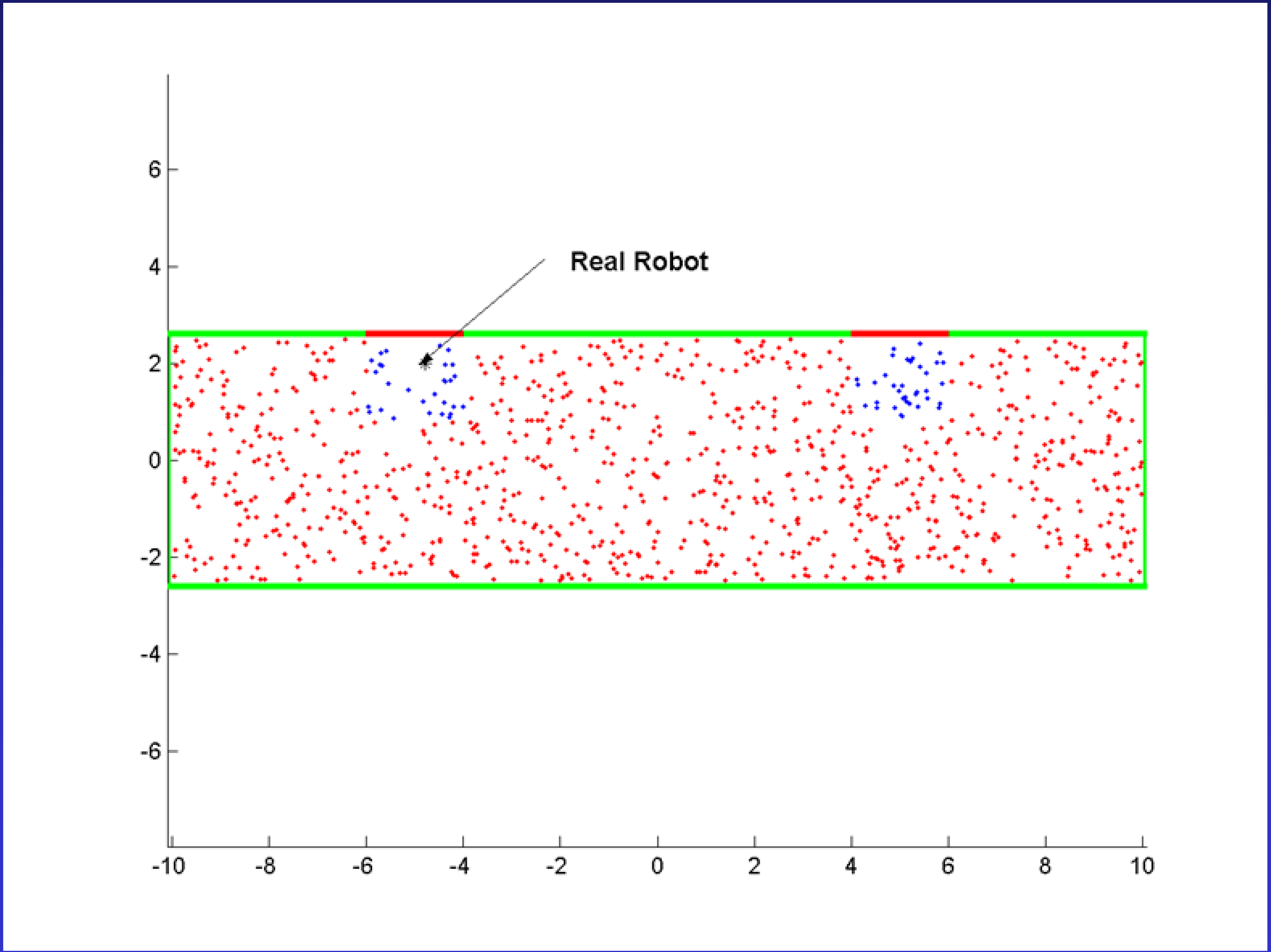
- The larger the error, the smaller the effect on the final state estimate
  - If *process* uncertainty is larger, *sensor* updates will dominate state estimate
  - If *sensor* uncertainty is larger, *process* propagation will dominate state estimate
- Improper estimates of the state and/or sensor covariance may result in a rapidly diverging estimator
  - As a rule of thumb, the residuals must always be bounded within a $\pm 3\sigma$ region of uncertainty
  - This measures the “health” of the filter
- *Many* propagation cycles can happen between updates

Particle Filters

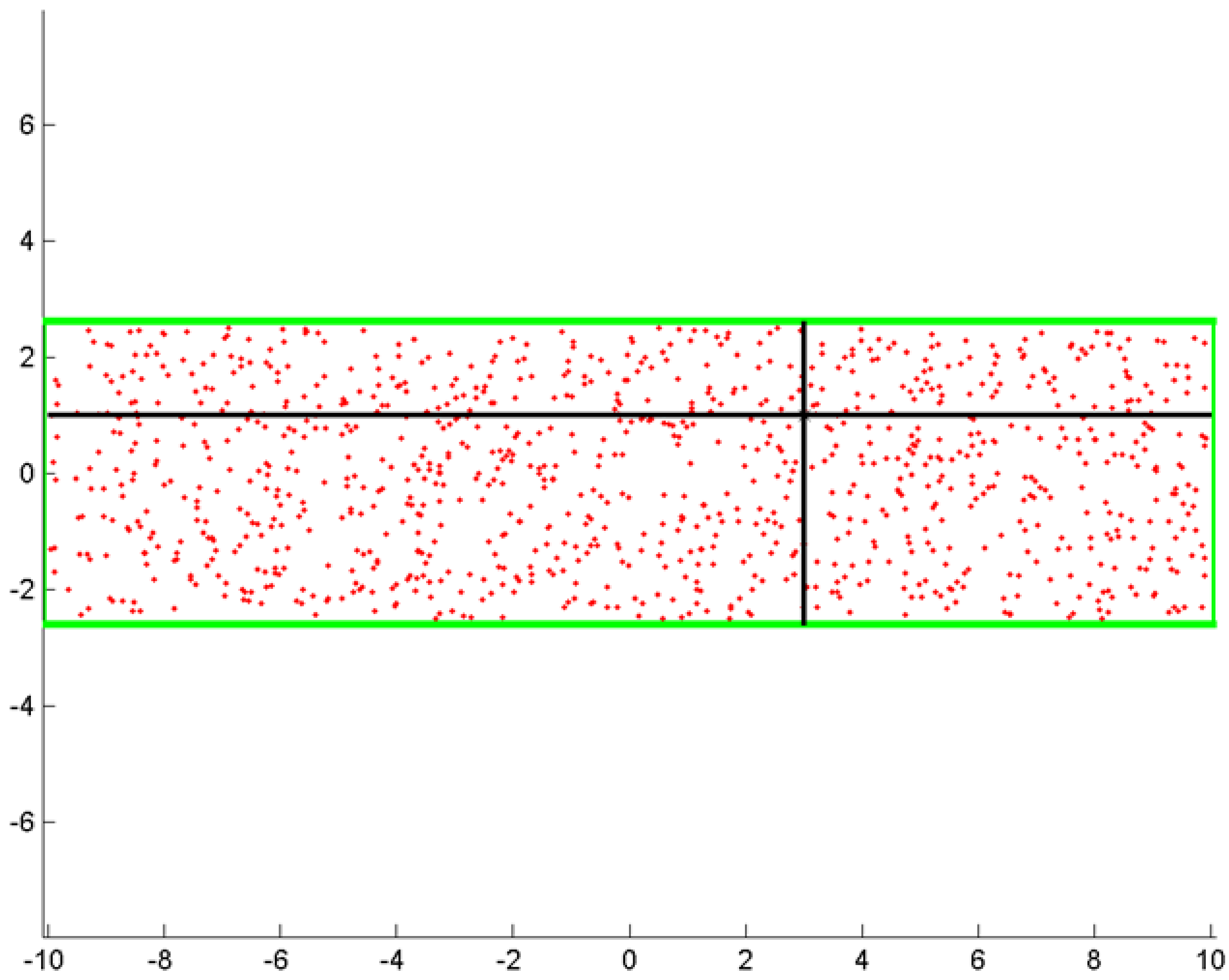
Environment with two red doors
(uniform distribution)



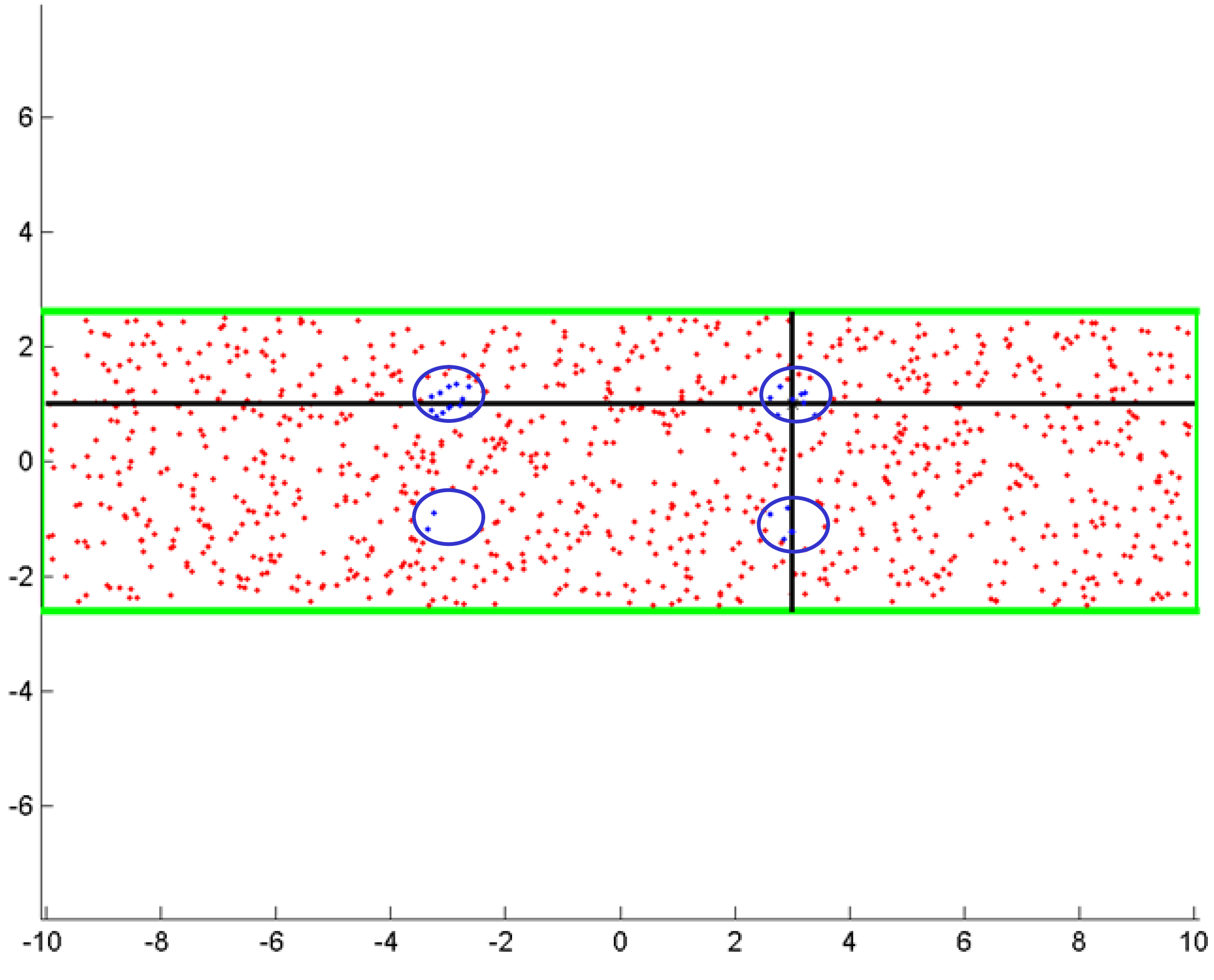
Environment with two red doors
(Sensing the red door)



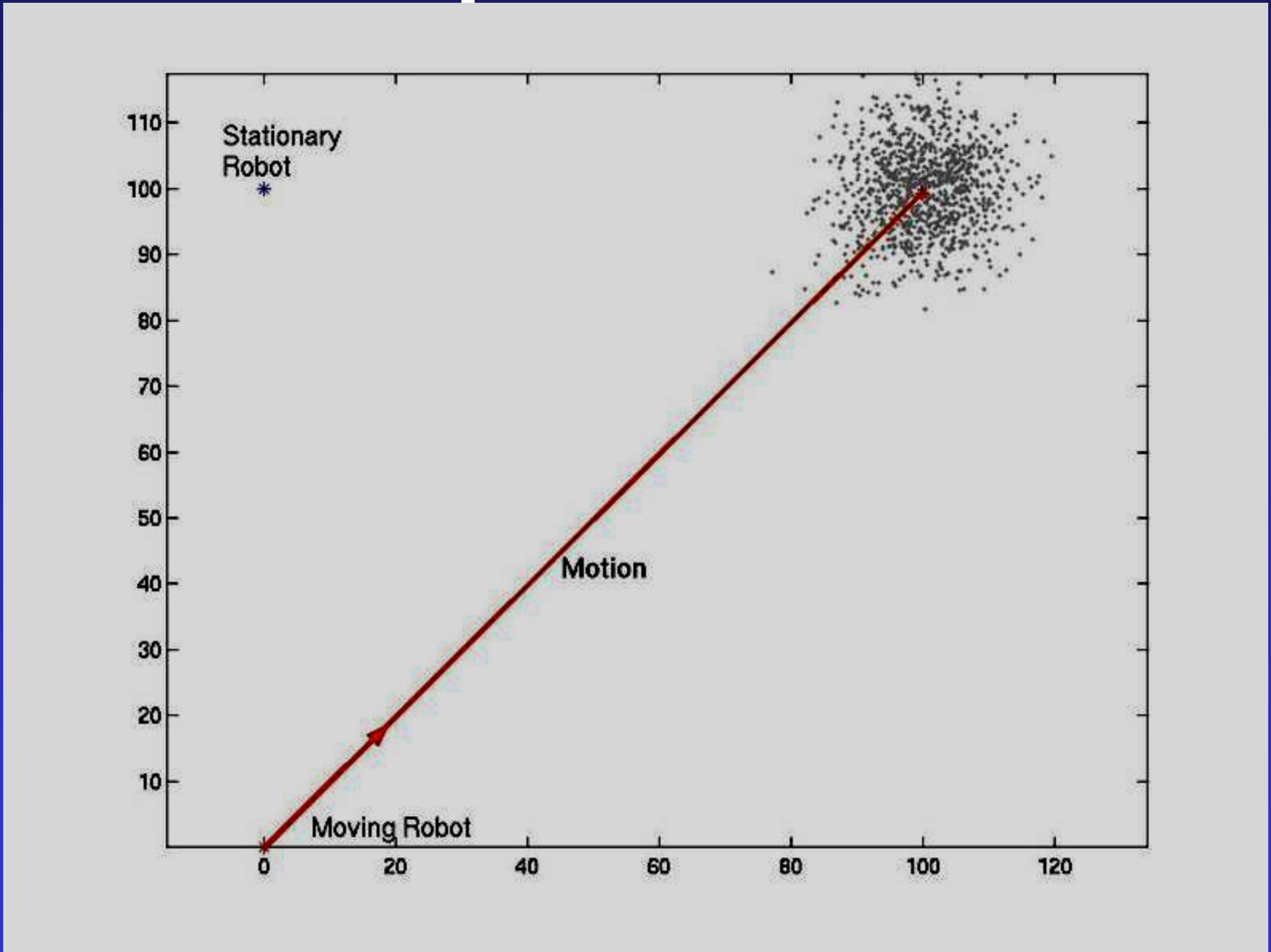
Sensing four walls



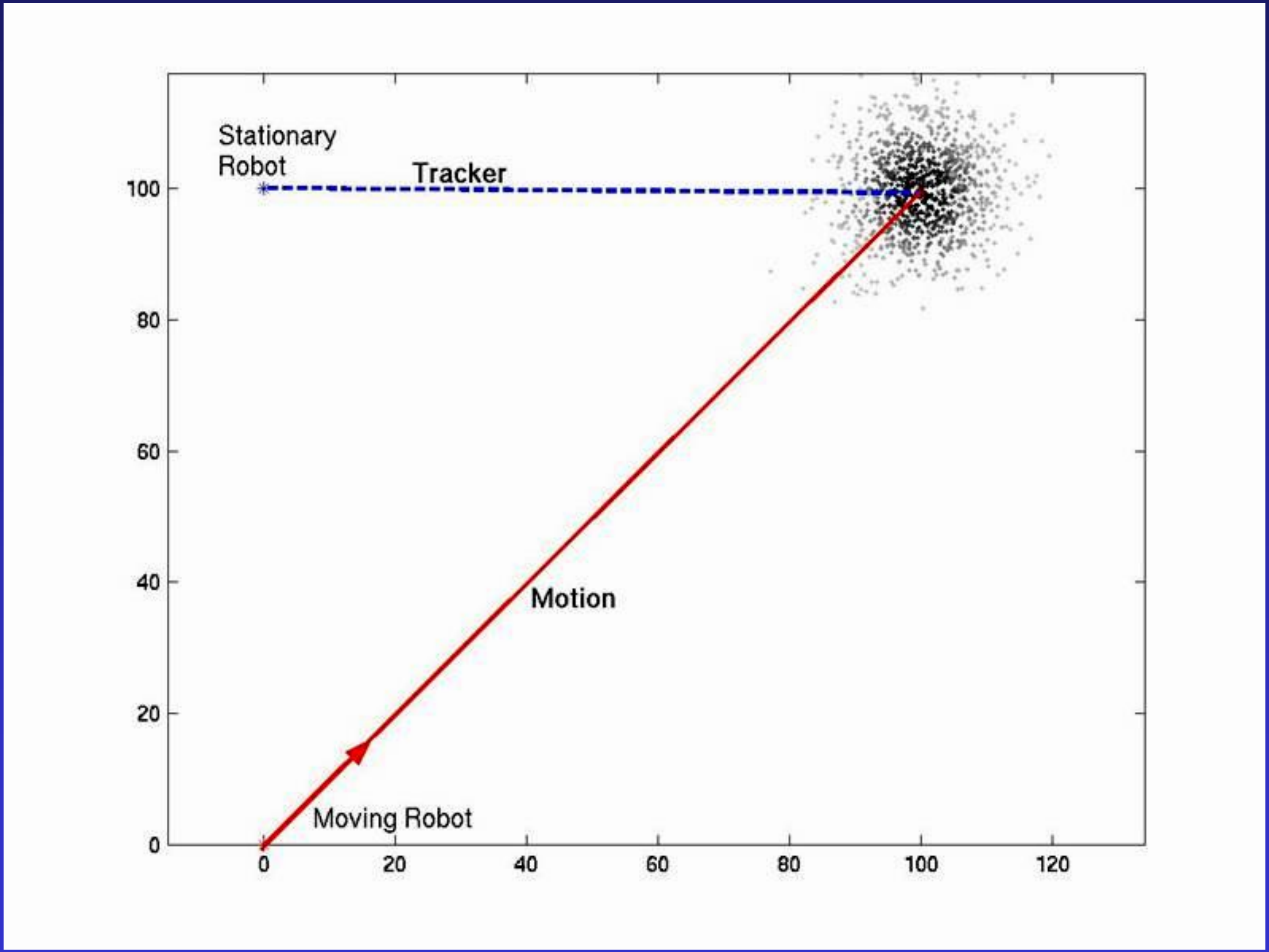
Four possible areas



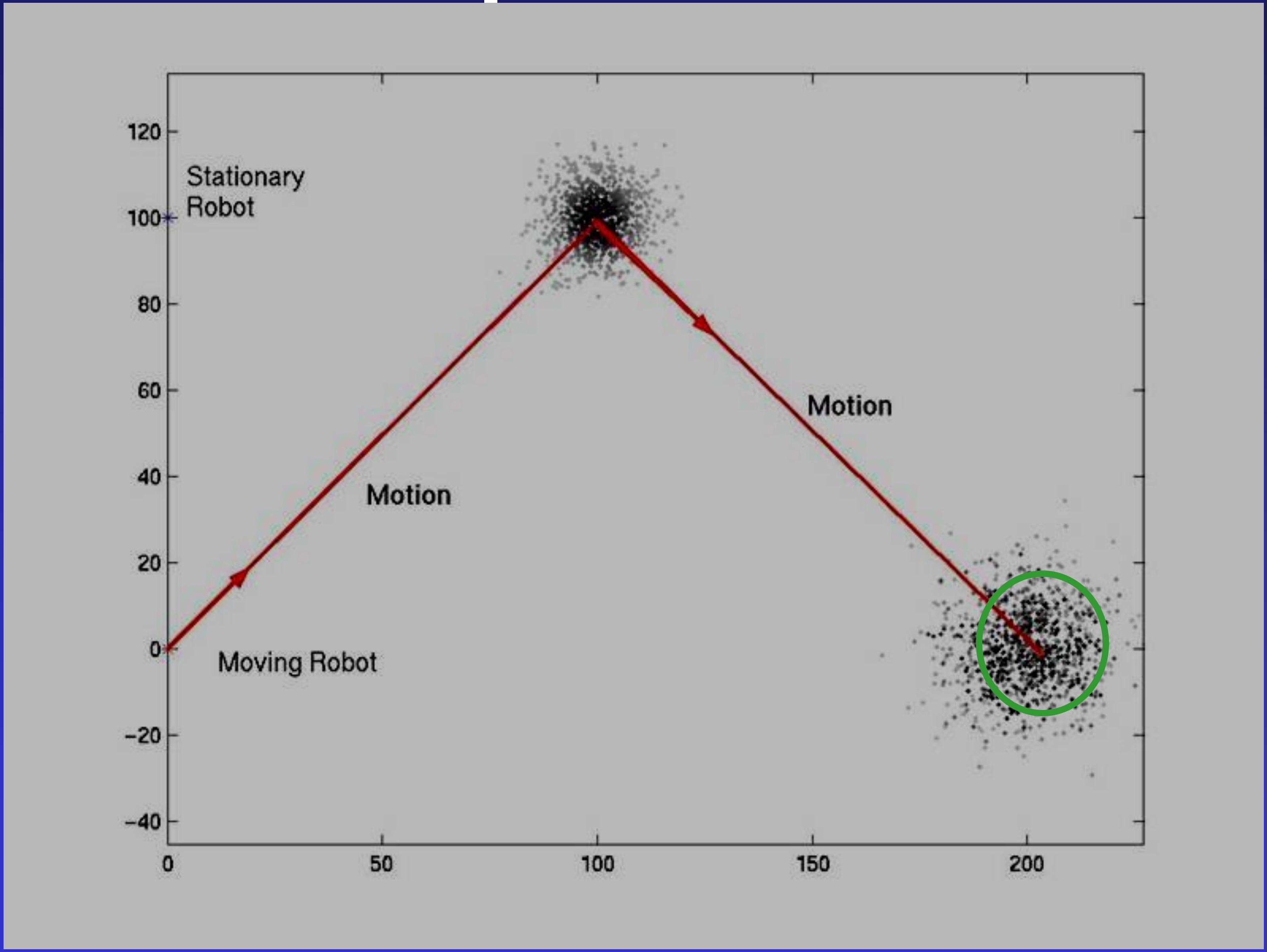
Example: Prediction



Example: Update



Example: Prediction



Example: Update