AMIRKABIR WINTER SCHOOL

Minimalism in Robotics: From Sensing to Filtering to Planning

PART 3: FILTERING

Steven M. LaValle

March 3, 2012

Overview of Topics

There are two general kinds of filters:

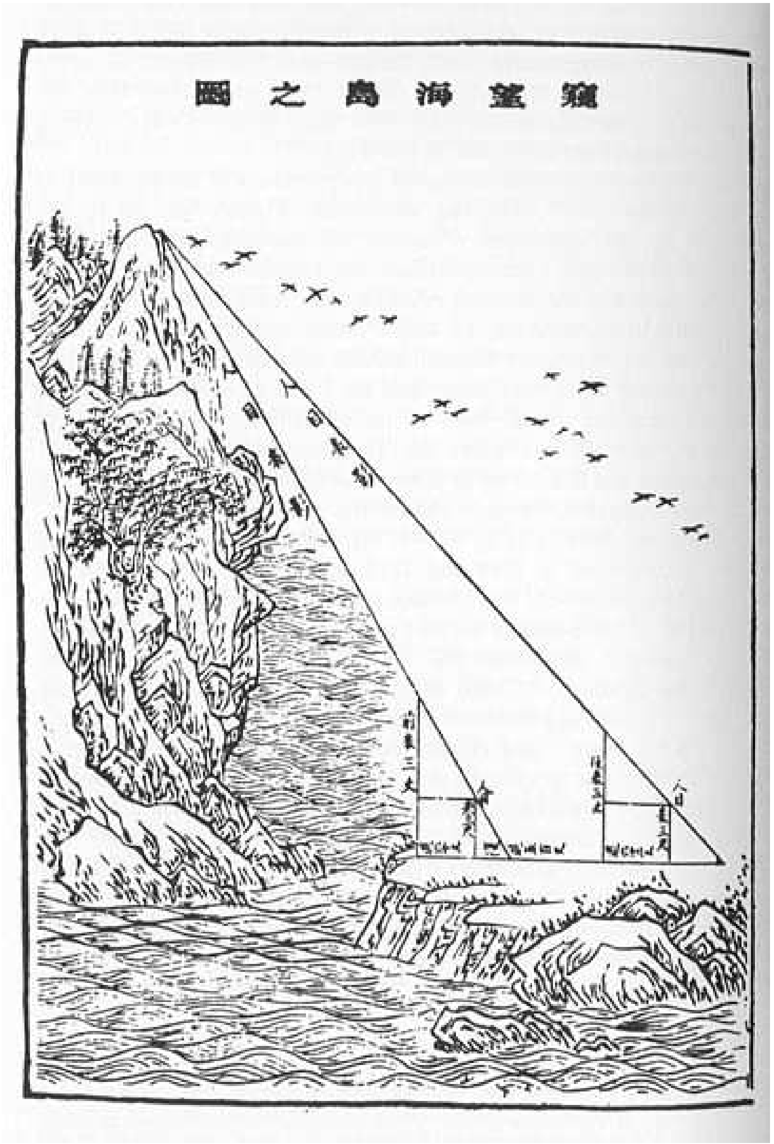
1. **Spatial:** Combining simultaneous observations from multiple sensors.

2. **Temporal:** Incrementally incorporating observations from a sensor at discrete stages.

Of course, we can make spatio-temporal filters.

Spatial filters

Triangulation: An Ancient Idea



Liu Hui, 263 A.D.

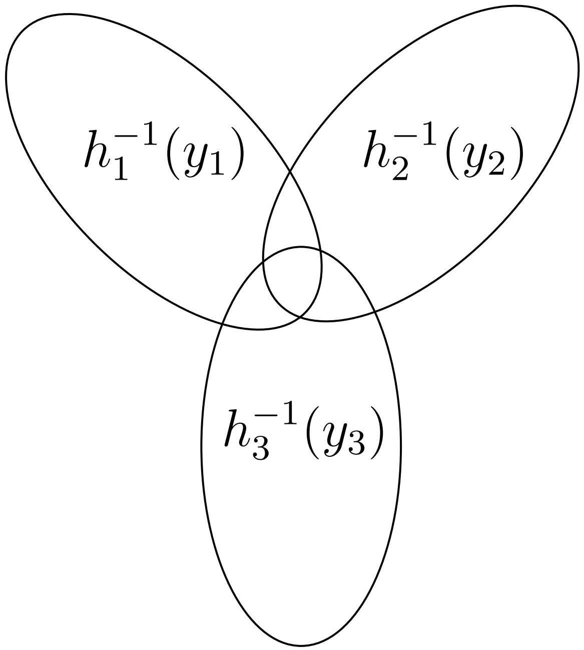
Triangulation: Intersection of Preimages

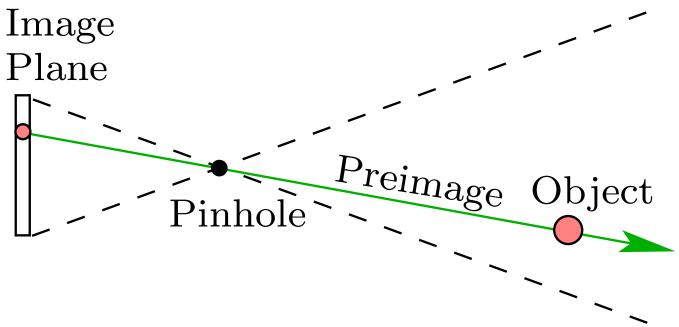
Consider any n sensor mappings $h_i : X \rightarrow Y_i$ for i from 1 to n.

The *triangulation* of a set of the observations y_1,. . .,y_n is:

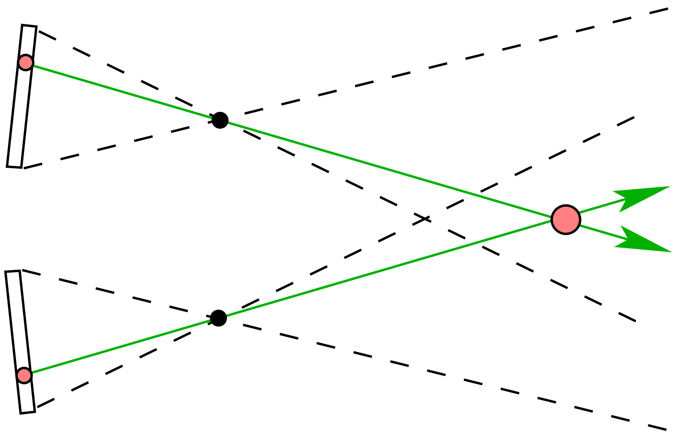
$$\Delta(y_1, \ldots, y_n) = h_1^{-1}(y_1) \cap h_2^{-1}(y_2) \cap \cdots \cap h_n^{-1}(y_n),$$

which is a subset of X.





One camera

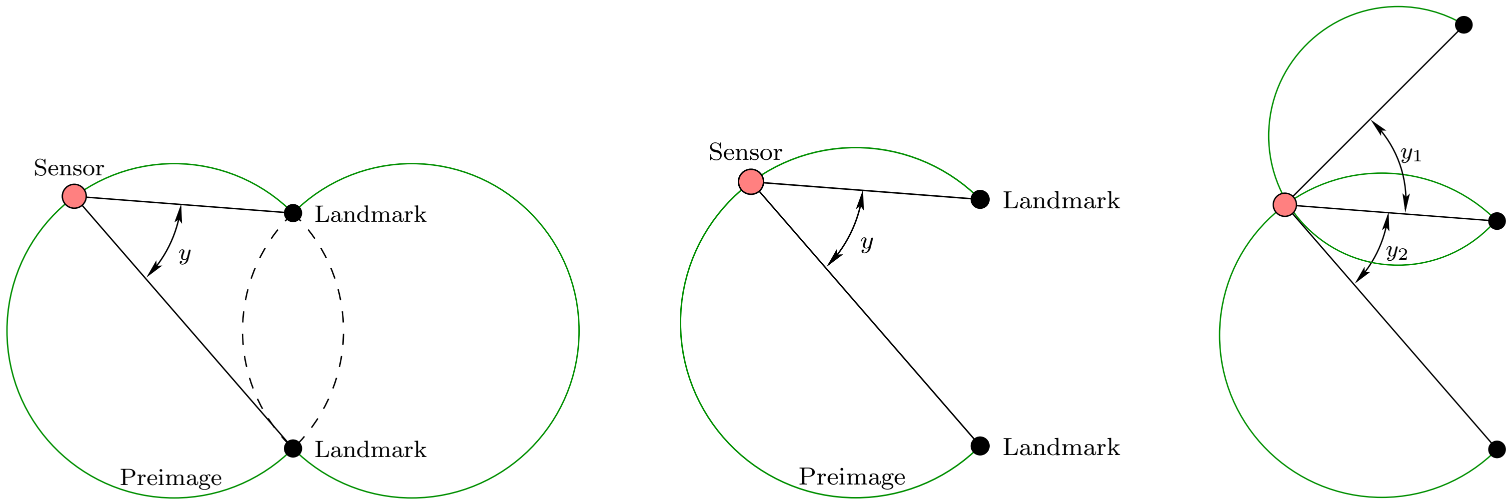


Triangulation

Observation: Object location in image plane

Preimages: Infinite rays

Triangulation: $\Delta(y_1, y_2)$ is a point.



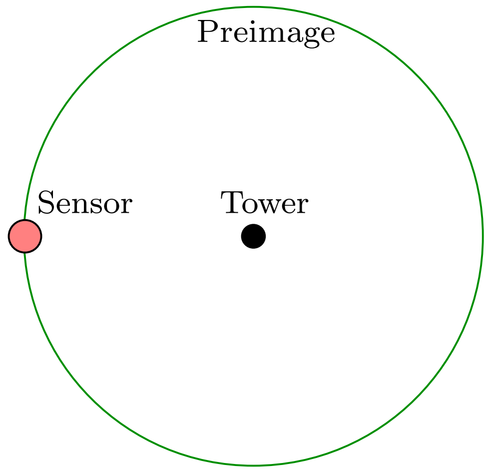
General preimage | Preimage with ordering | Triangulation

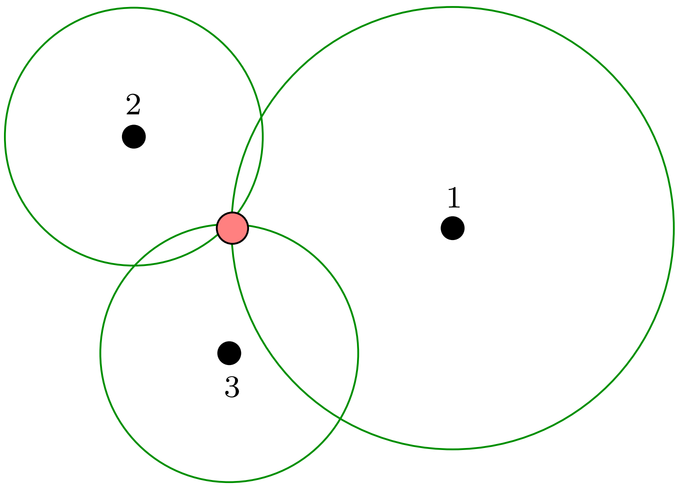
Observations: Angle between a pair of landmarks

Preimages: Circular arcs

Triangulation: $\Delta(y_1, y_2)$ is a point.

Trilateration

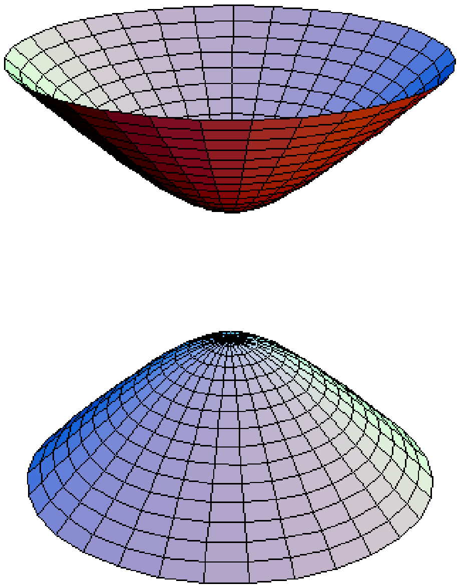




Observations: Distance to a landmark (based on TOA)

Preimages: Circles (or spheres in $\mathbb{R}^3$)

Triangulation: $\Delta(y_1, y_2, y_3)$ is a point.



Observations: Difference in distance to a pair of landmarks

Preimages: Hyperbolas

Triangulation: $\Delta(y_1, y_2, y_3)$ is a point.

Relation to Linear Algebra

Precisely how does information improve from multiple observations?

Linear case: $y_i = C_i x$, with $Y = \mathbb{R}^{m_i}$ and $X = \mathbb{R}^n$.
Assume C_i has rank k.

Each $h_i^{-1}(y_i)$ is a $n - k$-dimensional hyperplane through the origin of X.

$\Delta(y_1, \ldots, y_n)$ is the intersection of hyperplanes.

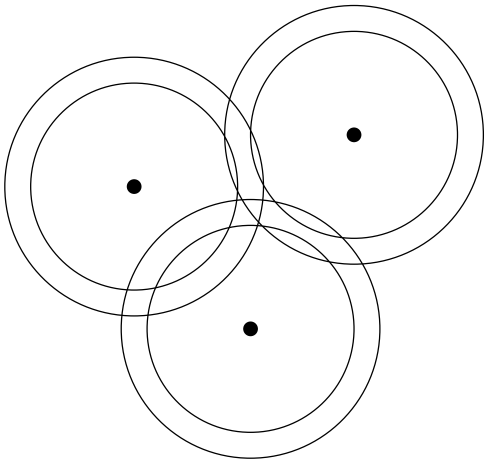
Preimage dimension and linear independent are crucial.

Nonlinear case: Similar, but tricky due to geometry.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Handling Disturbances

Nondeterministic disturbances:



Probabilistic disturbances:

$$p(x|y_1,\ldots,y_n) = \frac{p(y_1|x)p(y_2|x)\cdots p(y_n|x)p(x)}{p(y_1,\ldots,y_n)}$$

The *least squares* optimization problem:

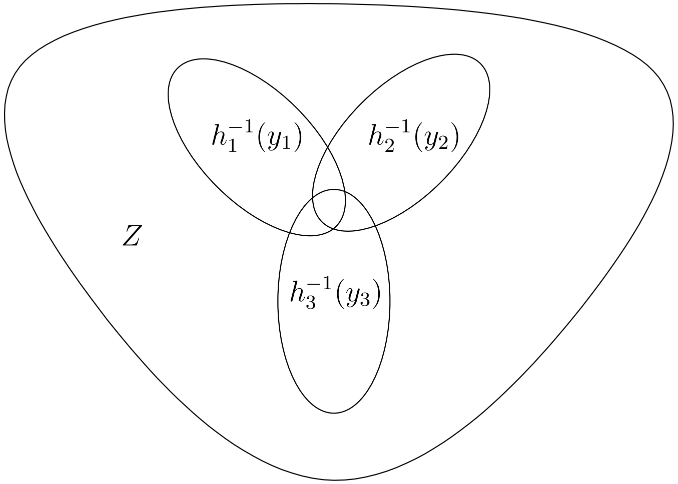
$$\min_{\hat{x}\in X} \sum_{i=1}^{n} d_i^2(\hat{x}, y_i)$$

Recall state-time space $Z = X \times T$.

A sensor is $h : Z \to Y$.

Triangulation intersects chunks of state-time space:

$$\Delta(y_1, \ldots, y_n) = h_1^{-1}(y_1) \cap h_2^{-1}(y_2) \cap \cdots \cap h_n^{-1}(y_n),$$



Important example: GPS simultaneously estimates position and time.

General temporal filters

Temporal Filters: Fundamental Questions

Given state space X and sensor $h : X \to Y$.

Let $\tilde{x} : [0, t] \to X$ be a state trajectory.

Let $\tilde{y} : [0, t] \to Y$ be an *observation history*.

When presented with $\tilde{y}$, there are two fundamental questions:

1. What is the set of state trajectories $\tilde{x} : [0, t] \to X$ that might have occurred?
2. What is the set of possible current states, $\tilde{x}(t)$?

Time Parameterized Sensor Mapping

Apply $h : X \to Y$ for every $t' \in [0, t]$.

Every $t' \in [0, t]$ yields some observation $\tilde{y}(t') = h(\tilde{x}(t'))$.

Let $\tilde{X}$ be all state trajectories.

Let $\tilde{Y}$ be all possible observation histories.

Applying h over $[0, t]$, we obtain the induced map:

$$H : \tilde{X} \to \tilde{Y}$$

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Answering the Fundamental Questions

This preimage answers 1st question:

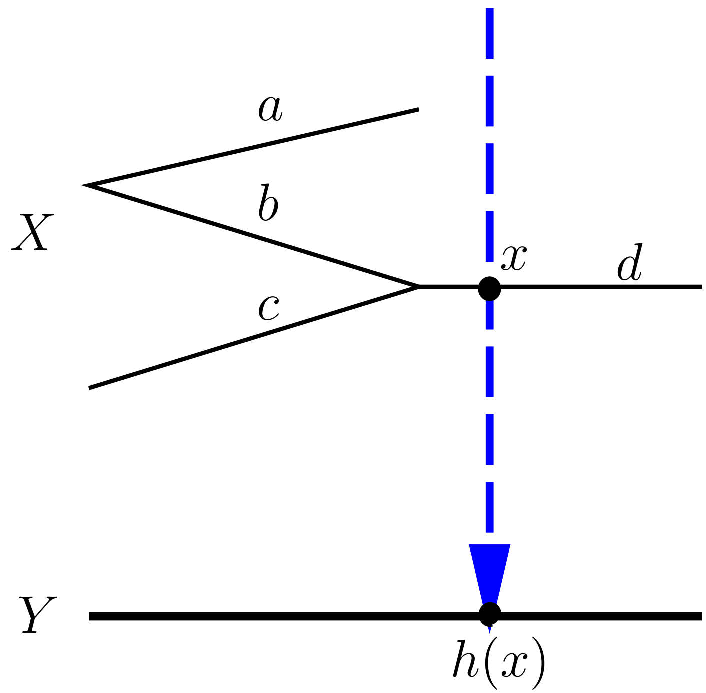
$$H^{-1}(\tilde{y}) = \{\tilde{x} \in \tilde{X} \mid \tilde{y} = H(\tilde{x})\}$$

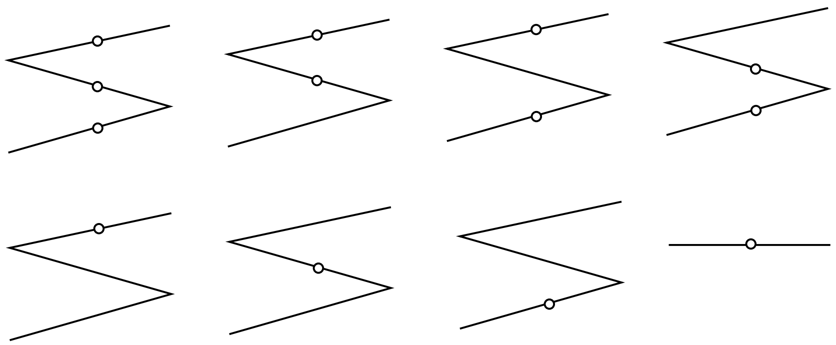
“all state trajectories that could have produced $\tilde{y}$”

Answer to 2nd question:

$$\{x \in X \mid \exists \tilde{x} \in H^{-1}(\tilde{y}) \text{ such that } \tilde{x}(t) = x\}$$

“all possible current states, considering the history $\tilde{y}$”





- $h^{-1}(y)$ is a finite set of points.
- On the left, the particular edge is unknown.
- Using $\tilde{y}$, the possible edges are narrowed down.
- Due to $\tilde{y}$, the precise timing is known.
- $H^{-1}(\tilde{y})$ becomes finite.

Discretely Indexed Histories

Rather than $\tilde{y} : [0, t] \rightarrow Y$, observations are obtained at discrete *stages*.

$h : X \rightarrow Y$ is a sequence $\tilde{y} = (y_1, \ldots, y_k)$.

Between stage i and $i + 1$, there are *no* observations.

For temporal filters:

1. Observations arrive incrementally; filter information is therefore updated incrementally.
2. Need to model how the state might change over time, when no observations are available.

The Structure of Temporal Filters

Let $\mathcal{I}$ be any set, and call it an *information space*.

Let ι_0 be called the *initial I-state*.

Transition function (filter):

$$\iota_k = \phi(\iota_{k-1}, y_k)$$

Sometimes it is shifted to $\iota_{k+1} = \phi(\iota_k, y_{k+1})$.

Operation of the Temporal Filter

- ι_0 is given.

Operation of the Temporal Filter

- ι_0 is given.
- y_1 is received.

Operation of the Temporal Filter

- ι_0 is given.
- y_1 is received.
- $\iota_1 = \phi(\iota_0, y_1)$ is computed.

Operation of the Temporal Filter

- ι_0 is given.
- y_1 is received.
- $\iota_1 = \phi(\iota_0, y_1)$ is computed.
- y_2 is received.

Operation of the Temporal Filter

- ι_0 is given.
- y_1 is received.
- $\iota_1 = \phi(\iota_0, y_1)$ is computed.
- y_2 is received.
- $\iota_2 = \phi(\iota_1, y_2)$ is computed.

Operation of the Temporal Filter

- ι_0 is given.
- y_1 is received.
- $\iota_1 = \phi(\iota_0, y_1)$ is computed.
- y_2 is received.
- $\iota_2 = \phi(\iota_1, y_2)$ is computed.
- y_3 is received.
- $\iota_3 = \phi(\iota_2, y_3)$ is computed.
- y_4 is received.
- $\iota_4 = \phi(\iota_3, y_4)$ is computed.
- y_5 is received.
- $\iota_5 = \phi(\iota_4, y_5)$ is computed.
- y_6 is received.
- $\iota_6 = \phi(\iota_5, y_6)$ is computed.
- y_7 is received.
- $\iota_7 = \phi(\iota_6, y_7)$ is computed.
- y_8 is received.
- $\iota_8 = \phi(\iota_7, y_8)$ is computed.

Some Generic Filter Examples

Sensor feedback: $\mathcal{I} = Y$

Stage counter: $\mathcal{I} = \{0, 1, 2, 3, \ldots\}$

History I-space transitions: $\mathcal{I} = \tilde{Y}$

State estimator: $\mathcal{I} = X$

Sensor Feedback Filter

I-space: $\mathcal{I} = Y$

Initial I-state: Not needed

Filter: $\iota_k = \phi(\iota_{k-1}, y_k) = y_k$

Reactive planning: Actions depend only on y_k.

Stage Counter Filter

I-space: $\mathcal{I} = \mathbb{N} \cup \{0\}$

Initial I-state: $\iota_0 = 0$

Filter: $\iota_k = \phi(\iota_{k-1}, y_k) = \iota_{k-1} + 1$

"open loop": Actions depend only on time or the stage index.

Tricky: Filter ignores observations, but are sensors need to know when the next stage occurs?

History I-Space Transition Filter

I-space: $\mathcal{I} = \tilde{Y}$

Initial I-state: $\iota_0 = ()$

Filter: $(y_1, \ldots, y_k) = \phi(\iota_{k-1}, y_k) = \phi((y_1, \ldots, y_{k-1}), y_k)$

This is simple concatenation onto the history.

State Estimator

I-space: $\mathcal{I} = X$

Initial I-state: $\iota_0 = x_0$

Generic filter: $\iota_k = \phi(\iota_{k-1}, y_k) = x_k$

"closed loop": Actions depend only on state

Problem: How did we determine x_k from ι_{k-1} and y_k?
Crucial issue: Must have enough information to compute transitions.

Simple State Estimator

How did the last filter work? Usually need a model of how X changes.
State space: $X = \mathbb{R}^2$

History-based sensor: $y_k = h(x_k, x_{k-1}) = x_k - x_{k-1}$

Filter: $\iota_k = \iota_{k-1} + y_k$

x_k is recovered from a telescoping sum.

This example is nice, but too simple.

We usually need a model of how X changes.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Life in the New I-Space

Once a filter ϕ is defined, we "live" in $\mathcal{I}$.

Given ι_0, ϕ, and $\tilde{y}_k = (y_1, \ldots, y_k)$ we can obtain ι_k by iterating ϕ:

$$\iota_k = \phi(\phi(\cdots \phi(\iota_0, y_1), y_2), \ldots, y_k)$$

We can always construct an *information mapping*:

$$\kappa : \mathcal{I} \times \tilde{Y} \rightarrow \mathcal{I}$$

Applying it:

$$\iota_k = \kappa(\iota_0, \tilde{y}_k)$$

State transition models

Ensuring Transition Functions

We want to make a filter:

$$\iota_k = \phi(\iota_{k-1}, y_k)$$

How do we know that ι_k can be computed from ι_{k-1} and y_k?

We can use every preimage $h^{-1}(y_k) \subseteq X$.

We also define *motion models* to model state change *between stages*.

Warning: Perhaps the mapping ϕ exists, but is not efficiently computable.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

How does the state change when not being observed?

Predictable state transitions:

$$x_{k+1} = f(x_k)$$

If the state is only known to be in $X_k \subseteq X$, then

$$X_{k+1}(X_k) = \{x_{k+1} \in X \mid x_k \in X_k \text{ and } x_{k+1} = f(x_k)\}.$$

This is a *forward projection*.

Simple enough, but states are usually not predictable.

Nondeterministic state transitions:

$$F : X \to \mathrm{pow}(X),$$

yielding $X_{k+1} = F(x_k) \subseteq X$.

The forward projection is

$$X_{k+1}(X_k) = \{x_{k+1} \in X \mid x_k \in X_k \text{ and } x_{k+1} \in F(x_k)\}.$$

Example: Bodies must move on a continuous path.

Probabilistic state transitions:

$$p(x_{k+1}|x_k)$$

The forward projection is

$$p(x_{k+1}) = \sum_{x_k \in X} p(x_{k+1}|x_k)p(x_k)$$

Filters with actions

Introducing Actions

Bodies may choose actions, which affect state transitions.

Example: Controlling a robot.

Passive: We do not choose actions, but receive them

Active: We get to chose the actions.

Whether passive or active, filtering is the same.

Let U be an *action space*.

Let $u_k \in U$ be the action applied at stage k.

Transition Models

Predictable state transitions:

$$x_{k+1} = f(x_k, u_k)$$

Nondeterministic state transitions:

$$F : X \times U \rightarrow \mathrm{pow}(X)$$

Probabilistic state transitions:

$$p(x_{k+1} | x_k, u_k)$$

These are all the same as before, but now depend on actions.

Expanding the History

In addition to $\tilde{y}$, we now have an *action history:*

$$\tilde{u}_k = (u_1, \ldots, u_k)$$

General filter template:

$$\iota_k = \phi(\iota_{k-1}, u_{k-1}, y_k)$$

The Full History I-Space Filter

History I-state: $\eta_k = (\tilde{y}_k, \tilde{u}_{k-1})$

History I-space: $\mathcal{I}_{hist}$ is all possible η_k for all k

A trivial filter:

$$\eta_k = \phi(\eta_{k-1}, u_{k-1}, y_k)$$

Simply concatenation, once again.

Two Important Generic Filters

Based on the type of uncertainty, we get two alternatives;

1. Nondeterministic filter, with $\mathcal{I}_{ndet} = \mathrm{pow}(X)$

2. Probabilistic filter (Bayesian filter), with $\mathcal{I}_{prob}$
   - Special case: Kalman filter, with $\mathcal{I}_{gauss} \subset \mathcal{I}_{prob}$

Bayesian (including Kalman) are extremely popular in robotics.

Localization, mapping, SLAM, ...

Nondeterministic filters

Nondeterministic Filters

Models: $h : X \rightarrow \mathrm{pow}(Y)$ and $F(x_k, u_k) \subseteq X$

The I-space: $\mathcal{I}_{ndet} = \mathrm{pow}(X)$

Initial I-state: $X_1 \subseteq X$

The filter:

$$X_{k+1}(\eta_{k+1}) = \phi(X_k(\eta_k), u_k, y_{k+1})$$

After first observation y_1:

$$X_1(\eta_1) = X_1(y_1) = X_1 \cap h^{-1}(y_1)$$

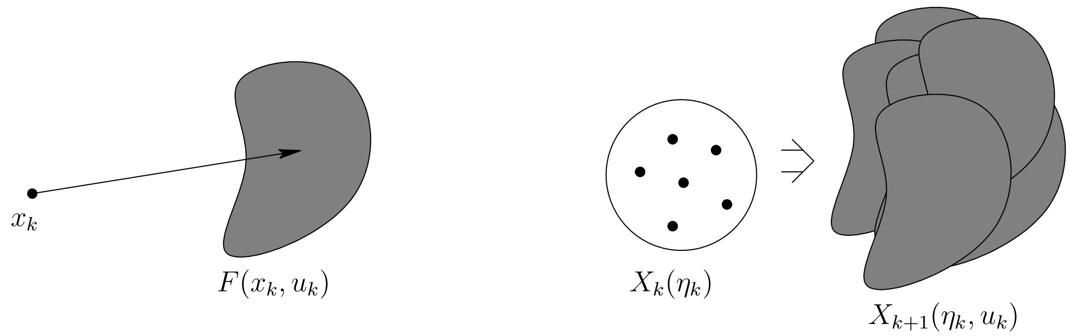
(Intersect initial constraint with observation preimage.)

Inductively, $X_k(\eta_k)$ is given.

Determine $X_{k+1}(\eta_{k+1})$ using $X_k(\eta_k)$, u_k, and y_{k+1}.

Using u_k,

$$X_{k+1}(\eta_k, u_k) = \bigcup_{x_k \in X_k(\eta_k)} F(x_k, u_k).$$



Using y_{k+1},

$$X_{k+1}(\eta_{k+1}) = X_{k+1}(\eta_k, u_k, y_{k+1}) = X_{k+1}(\eta_k, u_k) \cap h^{-1}(y_{k+1}).$$

Trajectory space filters

Nondeterministic Filter Over Trajectory Space

Let $Z = X \times T$ with $T = [0, t_f]$ and *final time* t_f.

A *complete trajectory* is $\tilde{x} : T \rightarrow X$.
A *partial trajectory* is $\tilde{x} : [0, t] \rightarrow X$ for any $t \in [0, t_f)$.

Let $\tilde{X}_c$ denote the set of complete trajectories.

Nondeterministic Filter Over Trajectory Space

Consider a set of sensors of the form $h_i : Z \to Y$.

Particularly, let $y_i = h_i(x_i, t_i) = (y'_i, t_i)$, in which $y'_i = h'_i(x)$ is a standard sensor mapping.

Suppose that n observations, $y_1, \ldots, y_n$ are obtained.
Each y_i is obtained from $y_i = h_i(\tilde{x}(t_i), t_i)$.

What is the set of possible trajectories?

Nondeterministic Filter Over Trajectory Space

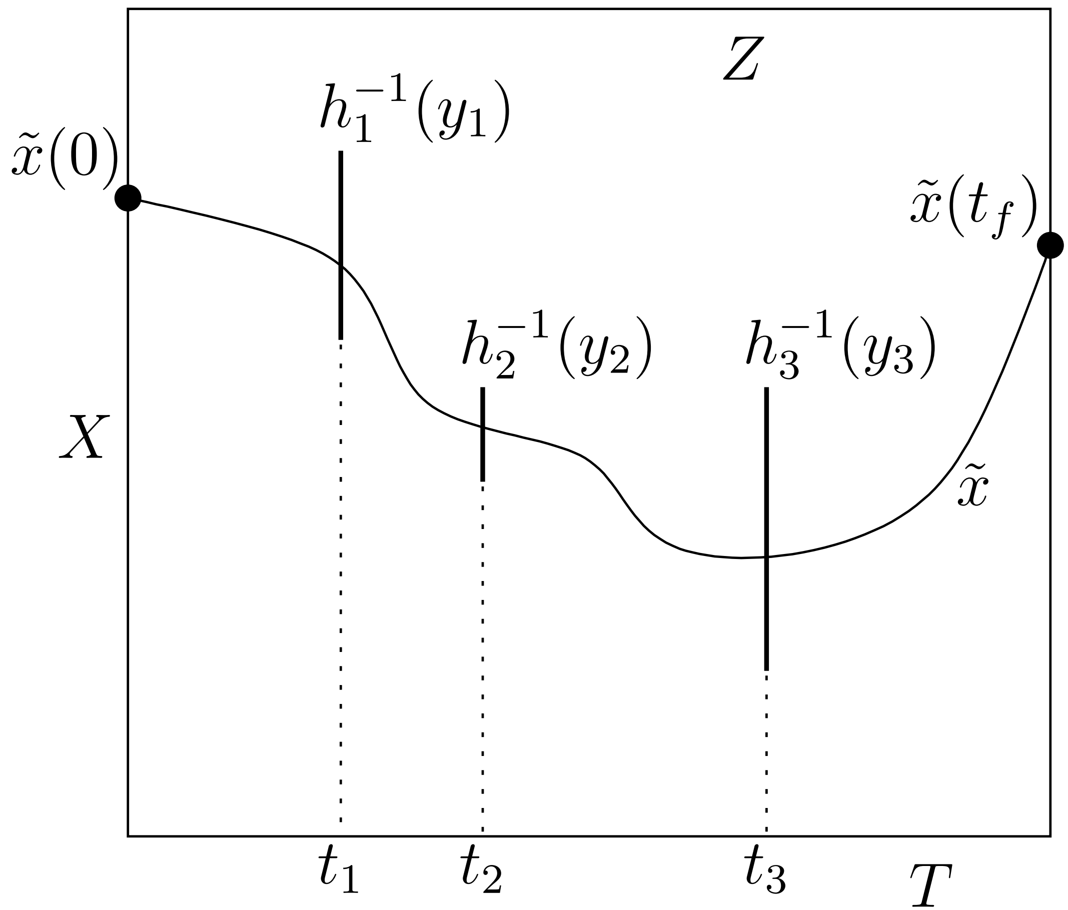
Consider the preimage over $\tilde{X}_c$:

$$\tilde{h}_i^{-1}(y_i) = \{\tilde{x} \in \tilde{X}_c \mid \tilde{x}(t_i) = h_i(x_i, t_i)\},$$

The filter is a form of triangulation on $\tilde{X}_c$:

$$\tilde{\triangle}(y_1, \ldots, y_n) = \tilde{h}_1^{-1}(y_1) \cap \tilde{h}_2^{-1}(y_2) \cap \cdots \cap \tilde{h}_n^{-1}(y_n),$$

Nondeterministic Filter Over Trajectory Space



Each observation is like a new “hoop” through the trajectory must “jump”.

Bayesian filters

Probabilistic Filters

Models: $p(y_k|x_k)$ and $p(x_{k+1}|x_k, u_k)$

The I-space: $\mathcal{I}_{prob}$, all pdfs over X

Initial I-state: $p(x_1)$, a prior pdf

The filter:

$$p(x_{k+1}|\eta_{k+1}) = \phi(p(x_k|\eta_k), u_k, y_{k+1}),$$

After first observation y_1:

$$p(x_1|\eta_1) = p(x_1|y_1) = \frac{p(y_1|x_1)p(x_1)}{\sum_{x_k} p(y_1|x_1)p(x_1)}$$

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Inductively, $p(x_k|\eta_k)$ is given.

Determine $p(x_{k+1}|\eta_{k+1})$ using $p(x_k|\eta_k)$, u_k, and y_{k+1}.

Using u_k,

$$\begin{aligned} p(x_{k+1}|\eta_k, u_k) &= \sum_{x_k \in X} p(x_{k+1}|x_k, u_k, \eta_k) p(x_k|\eta_k) \\ &= \sum_{x_k \in X} p(x_{k+1}|x_k, u_k) p(x_k|\eta_k). \end{aligned}$$

Using y_{k+1},

$$p(x_{k+1}|y_{k+1}, \eta_k, u_k) = \frac{p(y_{k+1}|x_{k+1}, \eta_k, u_k) p(x_{k+1}|\eta_k, u_k)}{\displaystyle\sum_{x_{k+1} \in X} p(y_{k+1}|x_{k+1}, \eta_k, u_k) p(x_{k+1}|\eta_k, u_k)}.$$

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Often it is intractible to compute the posteriors over X.
Sampling-based methods have been developed.

For some large number, m, of iterations, perform the following:

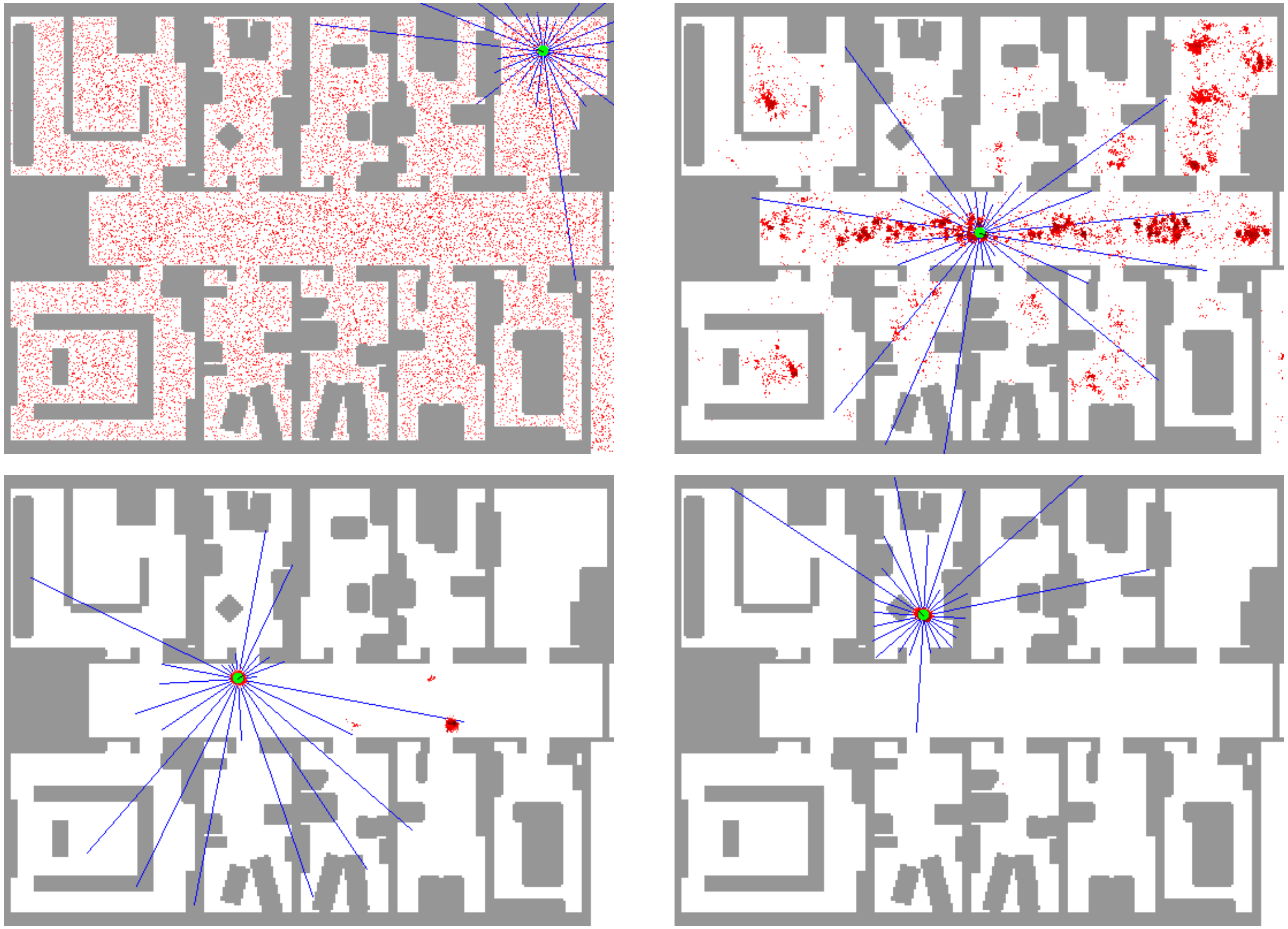
1. Select a state $x_k \in S_k$ according to the distribution P_k.
2. Generate a new sample, x_{k+1}, for S_{k+1} by generating a single sample according to the density $P(x_{k+1}|x_k, u_k)$.
3. Assign the weight, $w(x_{k+1}) = P(y_{k+1}|x_{k+1})$.

After the m iterations have completed, the weights over S_{k+1} are normalized to obtain a valid probability distribution, P_{k+1}.

Particle filters are used throughout robotics for localization and mapping.

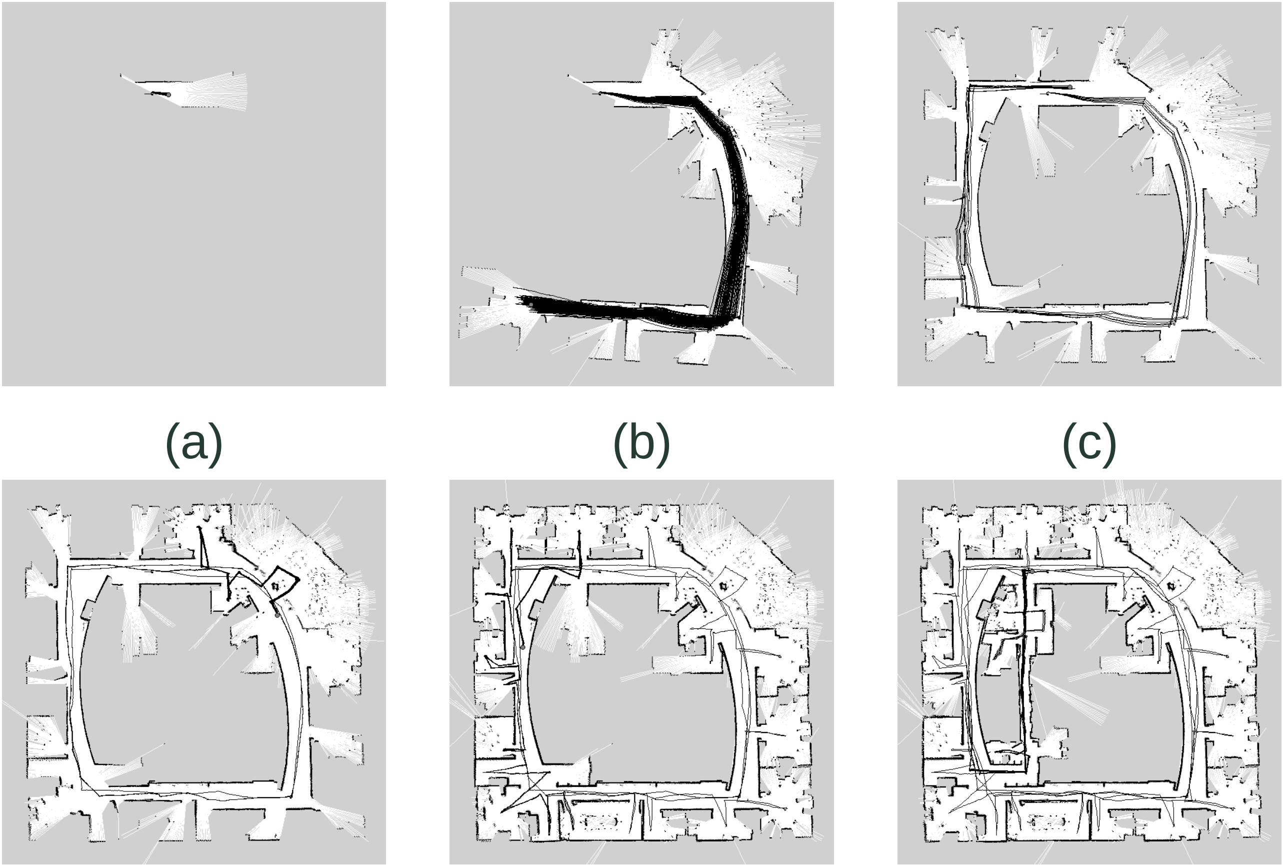
ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

Particle Filter For Localization



Fox, Thrun, Burgard, Delaert, 2001

SLAM Example



(a) (b) (c)

(d) (e) (f)

Hähnel, Fox, Burgard, Thrun, 2003

Kalman filter

Bayesian Special Case: Kalman Filter

State space: $X = \mathbb{R}^n$
Action space: $U = \mathbb{R}^m$
Disturbance space: $\Theta = \mathbb{R}^\ell$

Linear state transition equation:

$$x_{k+1} = A_k x_k + B_k u_k + G_k \theta_k$$

Example:

$$x_{k+1} = \begin{pmatrix} 0 & \sqrt{2} & 1 \\ 1 & -1 & 4 \\ 2 & 0 & 1 \end{pmatrix} x_k + \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} u_k + \begin{pmatrix} 1 & 1 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \theta_k$$

Observation space: $Y = \mathbb{R}^i$
Observation disturbance space: $\Psi = \mathbb{R}^j$

$$y_k = C_k x_k + H_k \psi_k$$

θ_k and ψ_k are zero-mean Gaussians with covariance matrices Σ_θ and Σ_ψ.

Kalman Filter: Linear Algebra Gore

First step (starting from μ_0, Σ_0):
$\mu_1 = \mu_0 + L_1(y_1 - C_1\mu_0)$ and $\Sigma_1 = (I - L_1C_1)\Sigma_0$
in which $L_1 = \Sigma_0 C_1^T \left(C_1\Sigma_0 C_1^T + H_1\Sigma_\psi H_1\right)^{-1}$

Mean update:
$$\mu_{k+1} = A_k\mu_k + B_ku_k + L_{k+1}(y_{k+1} - C_{k+1}(A_k\mu_k + B_ku_k))$$

Covariance update:
$\Sigma'_{k+1} = A_k\Sigma_kA_k^T + G_k\Sigma_\theta G_k^T$
$\Sigma_{k+1} = (I - L_{k+1}C_{k+1})\Sigma'_{k+1}$
in which $L_k = \Sigma'_k C_k^T \left(C_k\Sigma'_k C_k^T + H_k\Sigma_\psi H_k\right)^{-1}$

Kalman Filter Summary

I-space: $\mathcal{I}_{gauss}$, the set of all Gaussian pdfs

The linear algebra gore basically says:

$$(\mu_{k+1}, \Sigma_{k+1}) = \phi((\mu_k, \Sigma_k), u_k, y_{k+1})$$

Closure under Gaussians is a good thing:

Gaussian + action + sensor reading = Gaussian

The Kalman filter is used almost everywhere in engineering!

Extended Kalman filter: Keep approximating by Gaussians, even when the model is wrong.

Summary of General Filters

The updates expressions are closely related:

- Nondeterministic: Set union and set intersection
- Probabilistic: Marginalization and Bayes rule

Both involve considerable computational challenges in practice

Options:

- Get a bigger computer
- Resort to sampling-based, particle filtering techniques
- Compute approximations (for example, EKF)
- **Use the task and model structure to reduce complexity**

Combinatorial filters

Combinatorial Filters

Now we attempt to reduce filter complexity.

Introducing *combinatorial filters*

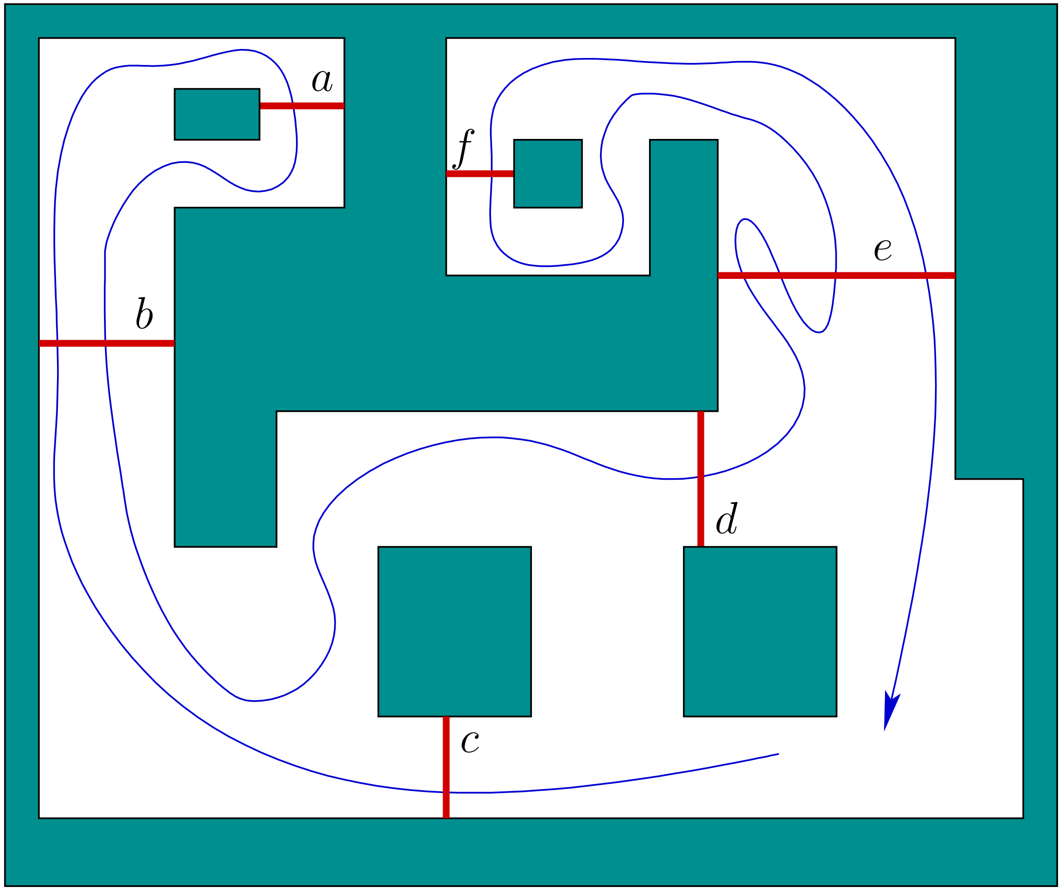
Three examples:

1. Obstacles and beams
2. Shadow information spaces
3. Gap navigation trees

Many, many more should be possible from the numerous virtual sensor models already given.

Obstacles and beams

Obstacles and Beams

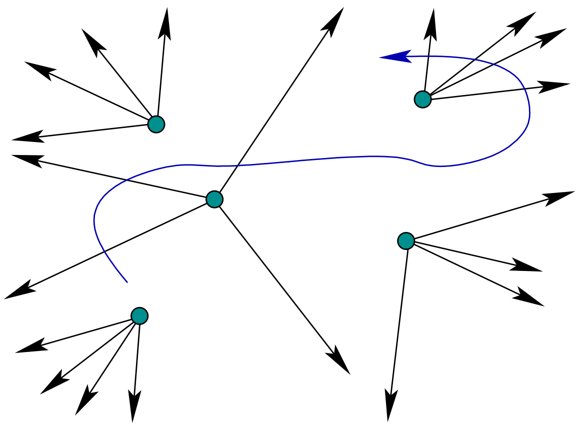


A point body moves in a known environment.

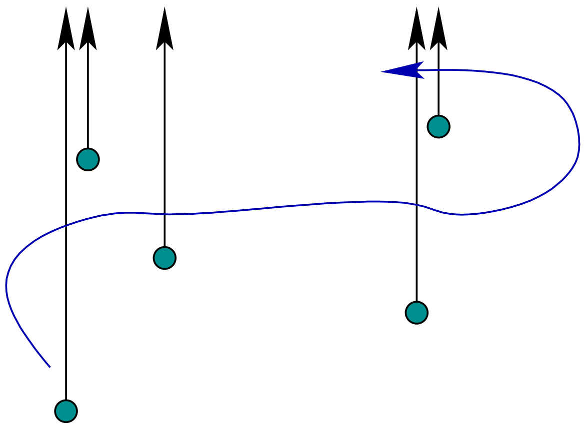
$X = E \subset \mathbb{R}^2$ and $\tilde{y} = cbabdeeefe$

What state trajectories are possible?

Remember: Virtual sensor models



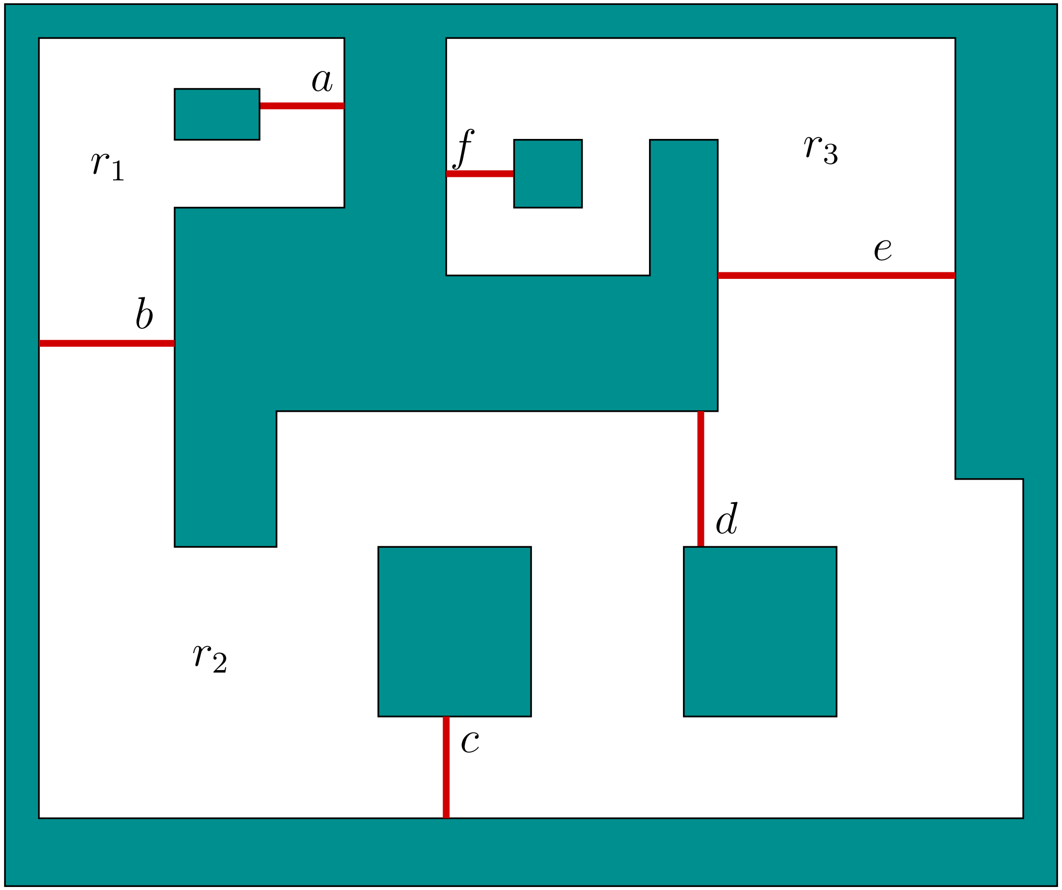
Crossing pairs of landmarks



Towers passing south

The obstacles and beams abstraction itself is important.

Beam Regions



A set of 3 two-dimensional regions $R = \{r_1, r_2, r_3\}$

A Simple Region Filter

Assumptions:

- Every beam either touches ∂E at each end or shoots off to infinity
- Every beam is uniquely labeled
- No pair of beams intersects

Let $\mathcal{I} = R$ and $\iota_0 = r_0$ (initial region known).

SIMPLE REGION FILTER:

$$r_k = \phi(r_{k-1}, y_k)$$

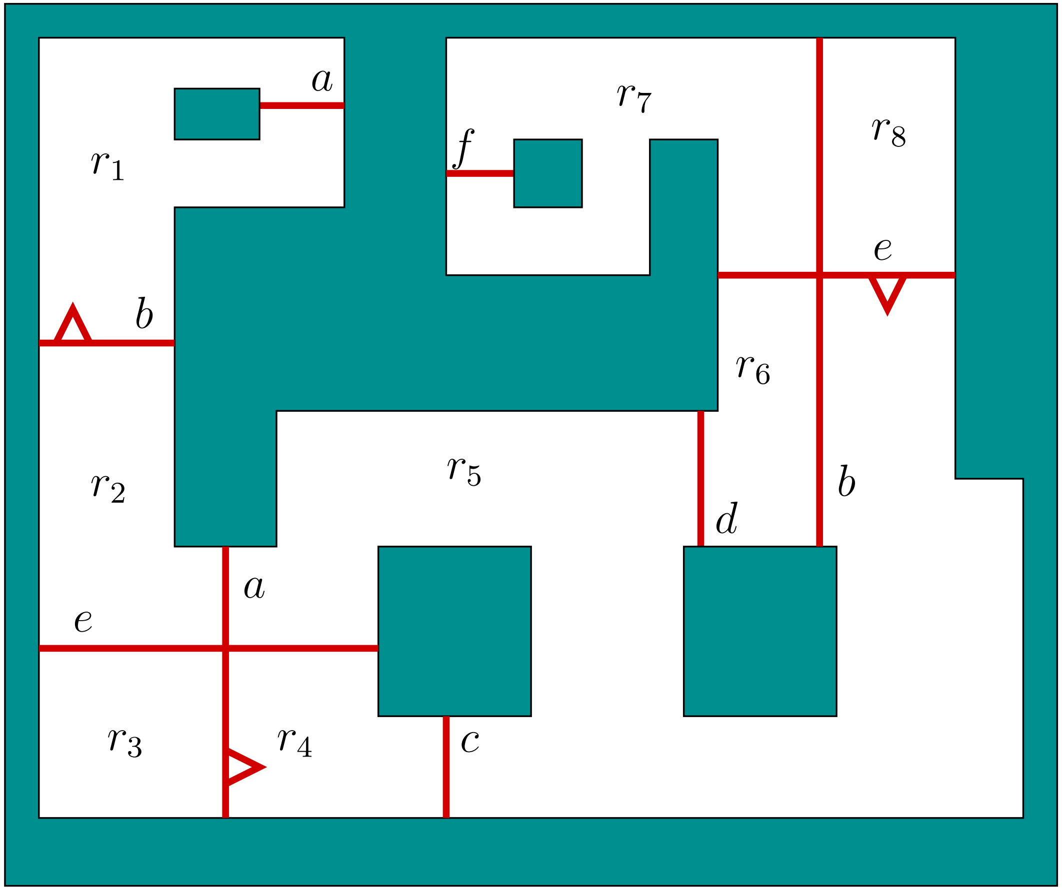
Using y_k and r_{k-1}, only one possibility exists for r_k.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

A more complicated scenario:

1. Beams may or may not be *distinguishable*.
2. Beams may or may not be *disjoint*.
3. Beams may or may not be *directed*.

With more complicated beams:

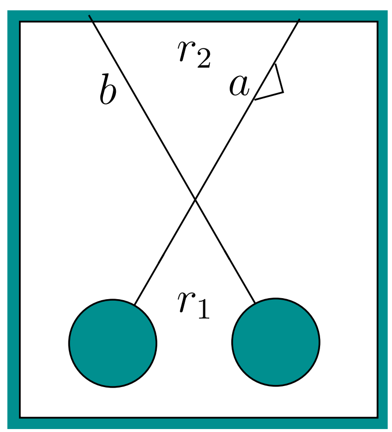


8 regions $R = \{r_1, \ldots, r_8\}$

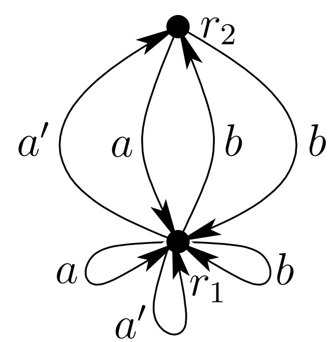
Construct a Multigraph

Let G be a multigraph:

- There is one *vertex* for every $r \in R$.
- A *directed edge* is made from $r_1 \in R$ to $r_2 \in R$ if and only if the body can cross a single beam to go from r_1 to r_2.
- Each edge is labeled with the beam label and the direction, if needed.



Two beams



The multigraph G

Nondeterministic Region Filter

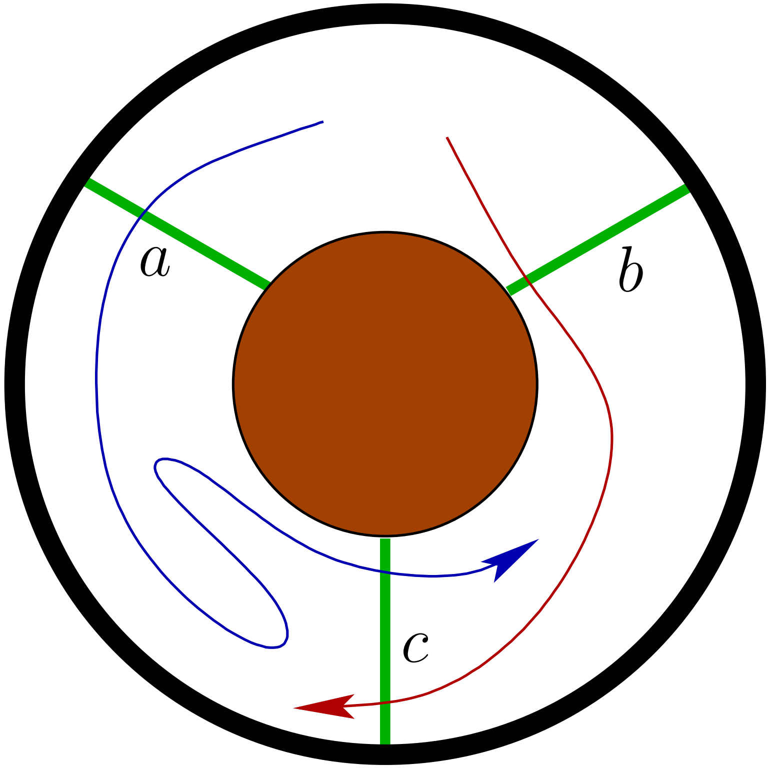
Let $\mathcal{I} = \mathrm{pow}(R)$ and $\iota_0 = R_0$, an initial region set.

Filter:

$$R_{k+1} = \phi(R_k, y_{k+1})$$

In particular:

1. Let $k = 0$ and $R_k = R_0$.
2. Let $R_{k+1} = \emptyset$.
3. For vertex in R_k and outgoing edge that matches y_{k+1}, insert the destination vertex/region into R_{k+1}.
4. Increment k, and go to Step 2.



In a given annulus E, we have two bodies, yielding $X = E^2 \subset \mathbb{R}^4$.

There are three disjoint, distinguishable, undirected beams a, b, c.

There are 3 regions, and nine combinations:
$(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 1)$, $(2, 2)$, $(2, 3)$, $(3, 1)$, $(3, 2)$, and $(3, 3)$

Multiple Body Filter

What if more than one body move around?
For n bodies, $X \subseteq \mathbb{R}^{2n}$.

Let $R^n = R \times R \times \cdots \times R$

I-space: $\mathcal{I} = \mathrm{pow}(R^n)$

Compute the multigraph G, and form a product G^n.

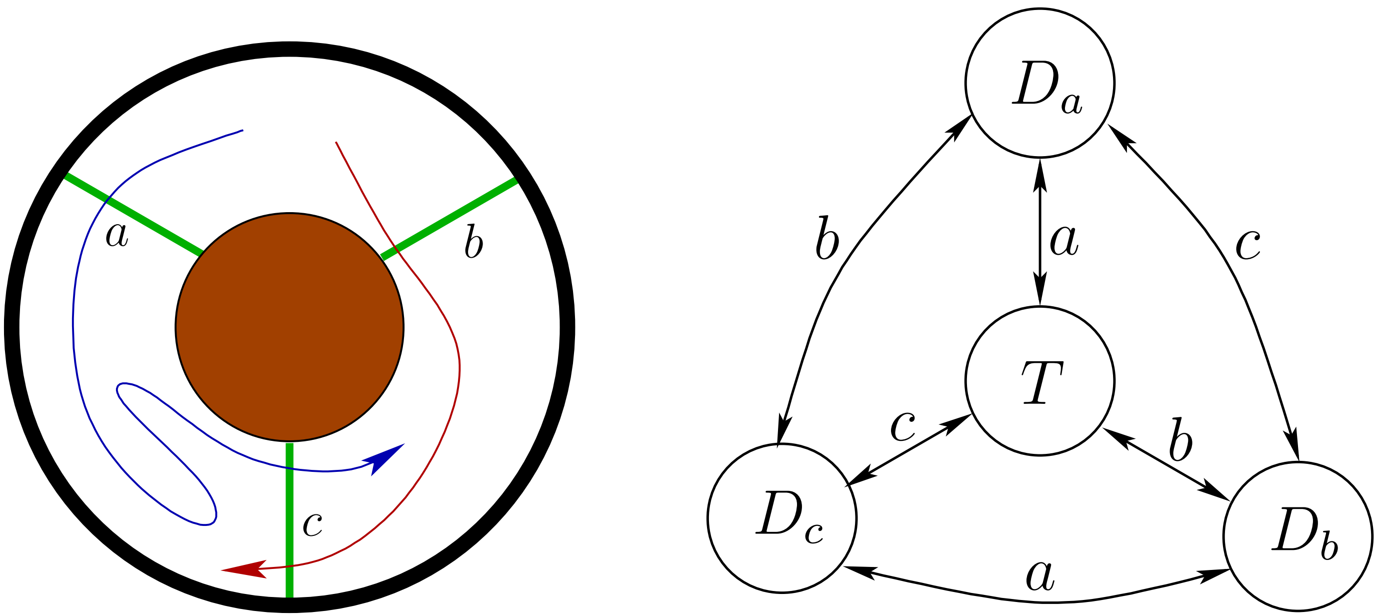
Vertices of G^n are region assignments $(r_1, \ldots, r_n)$.
Edges of G^n correspond to possible transitions.

Extend the one-body filter directly to G^n.
Problem: Number of vertices is exponential in n.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN

All of the region filters are special cases of nondeterministic filters. Can we simplify further?

Task: Determine whether the bodies in a room *together*?



The previous I-space would have 511 I-states.

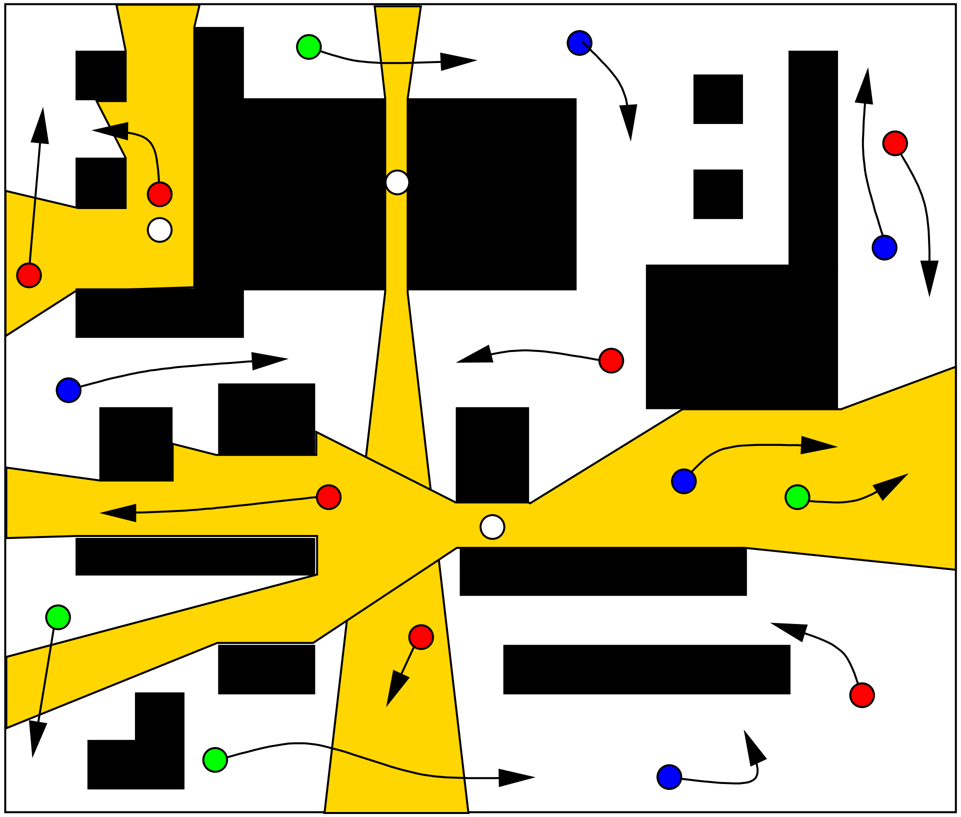
Here, the I-space is: $\mathcal{I} = \{T, D_a, D_b, D_c\}$

Filter: $\iota_k = \phi(\iota_{k-1}, y_k)$

Recall Myhill-Nerode and DFA minimization...

Shadow I-spaces

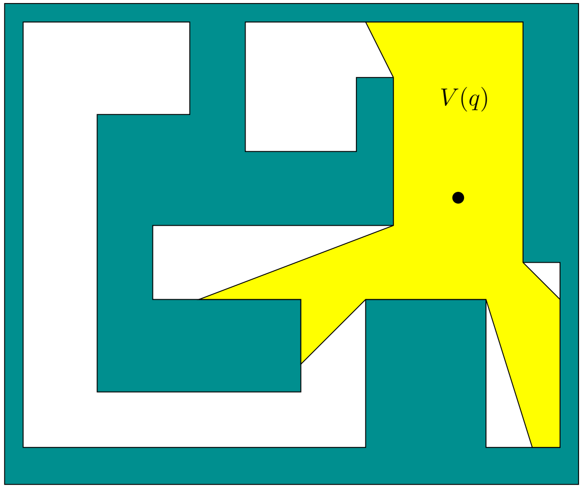
Shadow Information Spaces



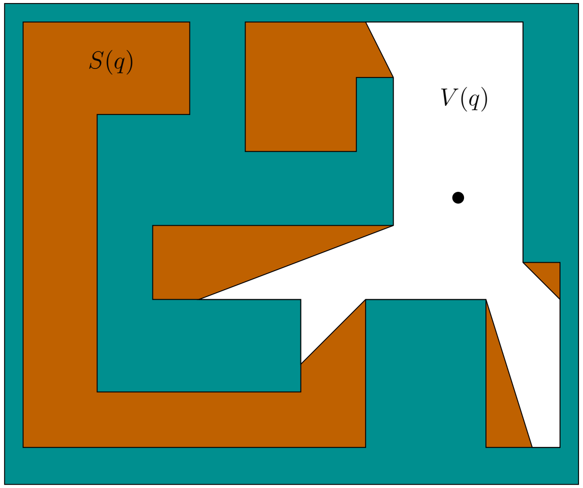
Keep track of bodies out of view–in the shadows.

How many are there? What kinds are there?

Detection and Shadow Regions



Detection region



Shadow region

$S(q)$ is the union of a finite number of *connected components*.

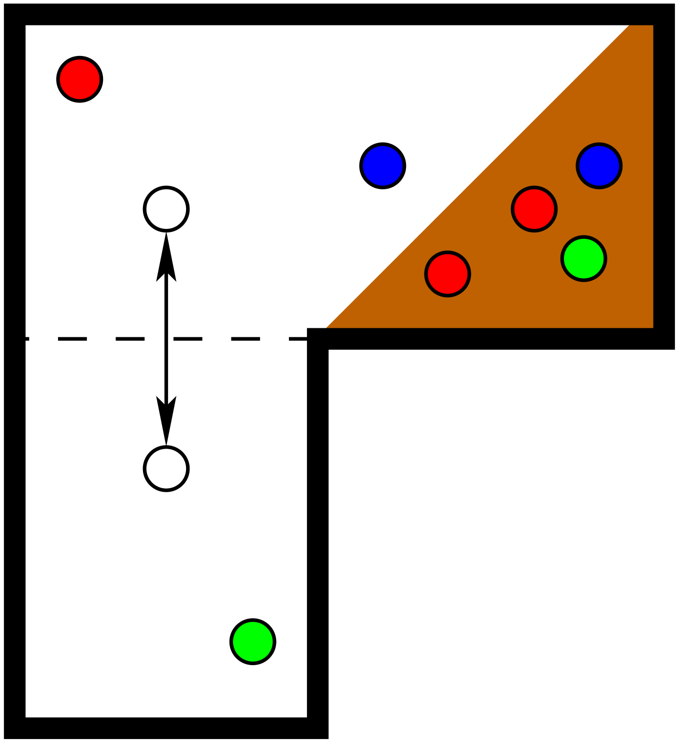
Crucial to pursuit-evasion algorithms.

Component Events

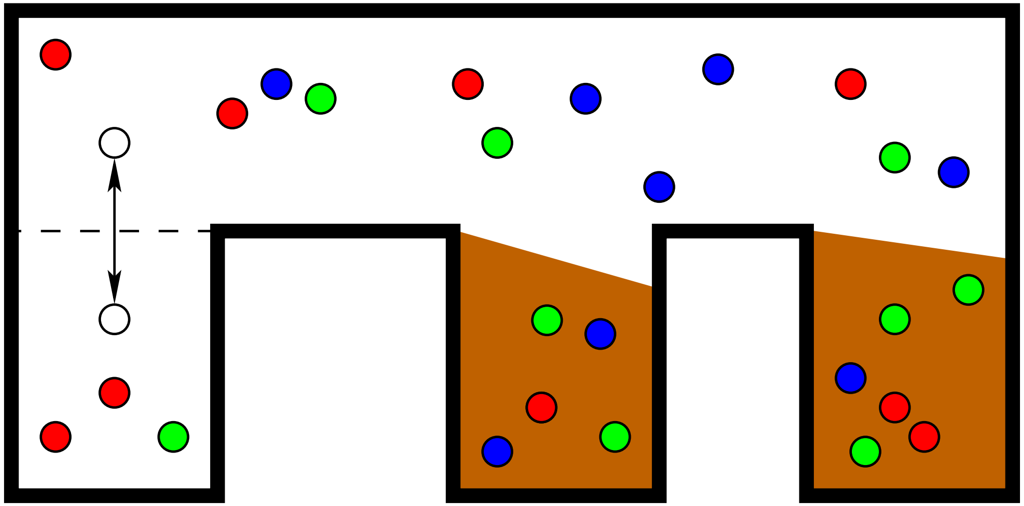
As q changes, there are *critical events* for $S(q)$:

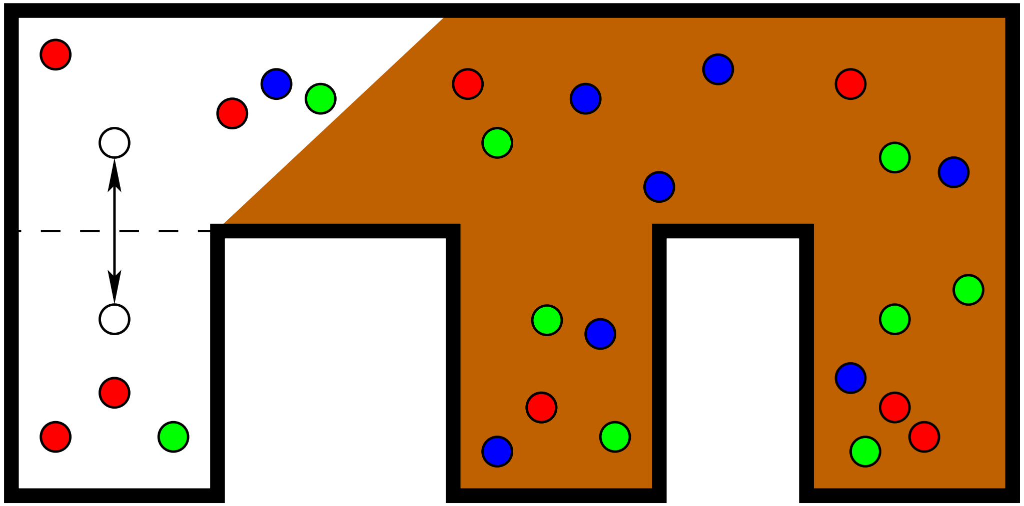
1. **Disappear:** A shadow component vanishes, which eliminates a hiding place for the bodies.
2. **Appear:** A shadow component appears, which introduces a new hiding place for the bodies.
3. **Split:** A shadow component splits into multiple shadow components.
4. **Merge:** Multiple shadow components merge into one shadow component.

We make appropriate general position assumptions.

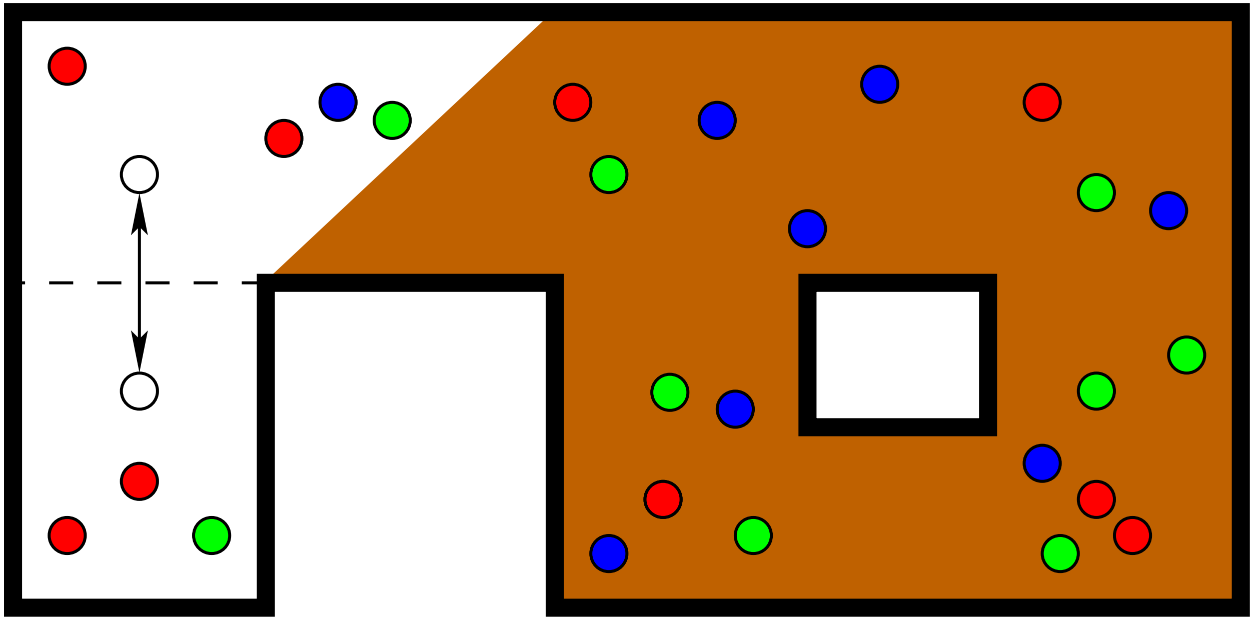


Split and Merge





Be Careful About Holes



What Information Do We Have?

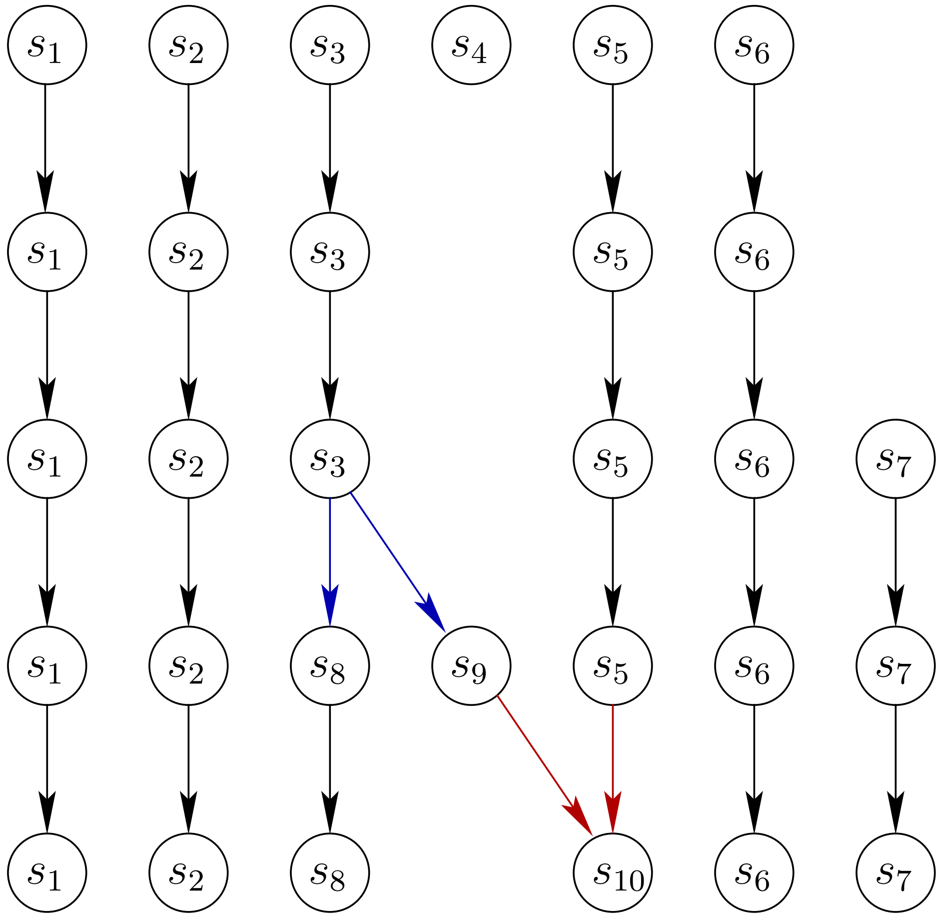
Set of shadows at stage k:

$$S_k = \{s_1, s_2, \ldots, s_n\}$$

Transition from S_k to S_{k+1}:

1. **Disappear:** $S_{k+1} = S_k \setminus \{s\}$ for some s.
2. **Appear:** $S_{k+1} = S_k \cup \{s\}$ for some new s.
3. **Split:** Split relation, $S(s, s', s'')$, meaning s splits to form s' and s''.
4. **Merge:** Merge relation, $M(s, s', s'')$, meaning s and s' merge to form s''.

The sequence over 5 stages:



s_4 disappears

s_7 appears

s_3 splits into s_8 and s_9

s_9 and s_5 merge into s_{10}

Each stage is the interval of time between events.

Pursuit-Evasion Filter

Is there an evader in $S(q)$?
Used in several visibility-based pursuit-evasion algorithms.

Keep a status bit for each component:

$$b_k : S_k \to \{0, 1\}$$

The filter needs only to maintain a single bit per component:

- "0" means that there is definitely no body in s_1
- "1" means that could be a body in s_1

Pursuit-Evasion Filter

Update rules when going from S_k to S_{k+1}:

1. **Disappear:** Nothing to update.
2. **Appear:** $b_{k+1}(s) = 0$.
3. **Split:** $b_{k+1}(s') = b_k(s)$ and $b_{k+1}(s'') = b_k(s)$.
4. **Merge:** $b_{k+1}(s) = 0$ if and only if $b_k(s') = 0$ and $b_k(s'') = 0$

Note: Split and merge relations are used.

Count-Bounding Filter

How many bodies in each component of $S_k(q)$?
Keep nonnegative integers or ∞ for each component.

Lower bound:

$$\ell_k : S_k \rightarrow \mathbb{N} \cup \{0, \infty\}$$

Upper bound:

$$u_k : S_k \rightarrow \mathbb{N} \cup \{0, \infty\}$$

Naive update rules when going from S_k to S_{k+1}:

1. **Disappear:** Nothing to update.
2. **Appear:** $\ell_{k+1}(s) = u_{k+1}(s) = 0$.
3. **Split:** $\ell_{k+1}(s') = 0$, $\ell_{k+1}(s'') = 0$, $u_{k+1}(s') = u_k(s)$, and $u_{k+1}(s'') = u_k(s)$.
4. **Merge:** $\ell_{k+1}(s'') = \ell_k(s) + \ell_k(s')$ and $u_{k+1}(s'') = u_k(s) + u_k(s')$.

Count-Bounding Filter

Let c, c', and c'' be the actual number of bodies in s, s', and s''.

If $S(s, s', s'')$, then $c = c' + c''$.

if $M(s, s', s'')$, then $c + c' = c''$.

Interpretation: The I-state is a polytope on an integer lattice.

Let $|S_k| = m$, and consider integer lattice $\mathbb{Z}^m$.

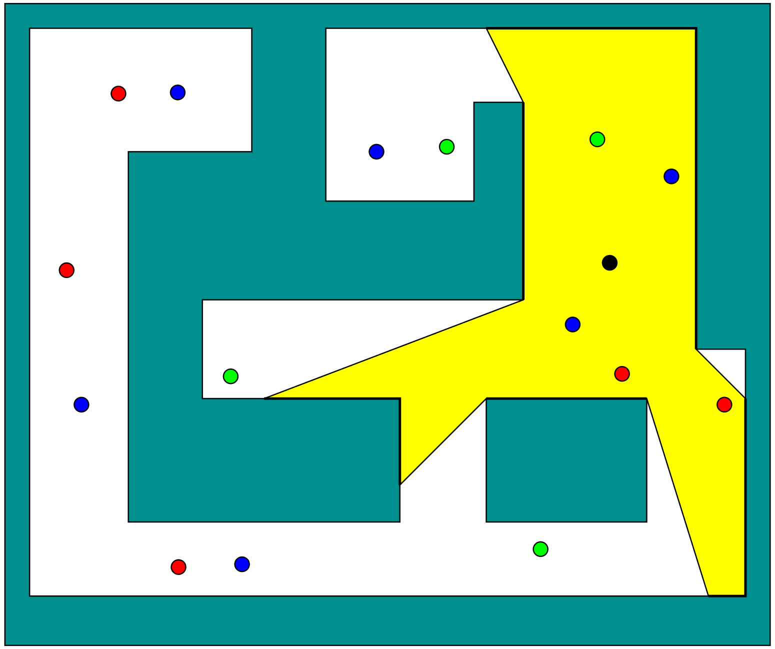
Consider all constraints due to

- $\ell_k(s)$ for all $s \in S_k$.
- $u_k(s)$ for all $s \in S_k$.
- All equations of the form $c = c' + c''$ and $c + c' = c''$.

The polytope can be efficiently queried to get count estimates.

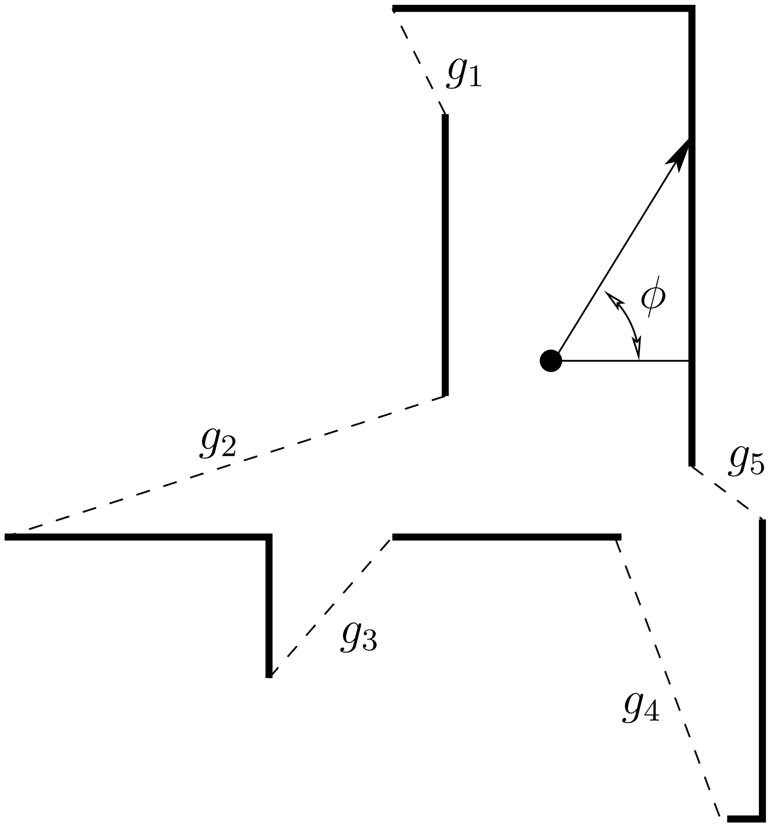
Count-Bounding For Teams

- Extend to teams of partially distinguishable bodies
- Efficient max-flow algorithms compute I-states
- See Yu, LaValle, ICRA 2008.
- Open problem: Planning using these filters.



Gap navigation trees

Recall the gap sensor:



$y = (g_1, g_2, g_3, g_4, g_5)$

What happens as q varies? The same 4 critical events!

Gap Navigation Trees

Set of gaps at stage k:

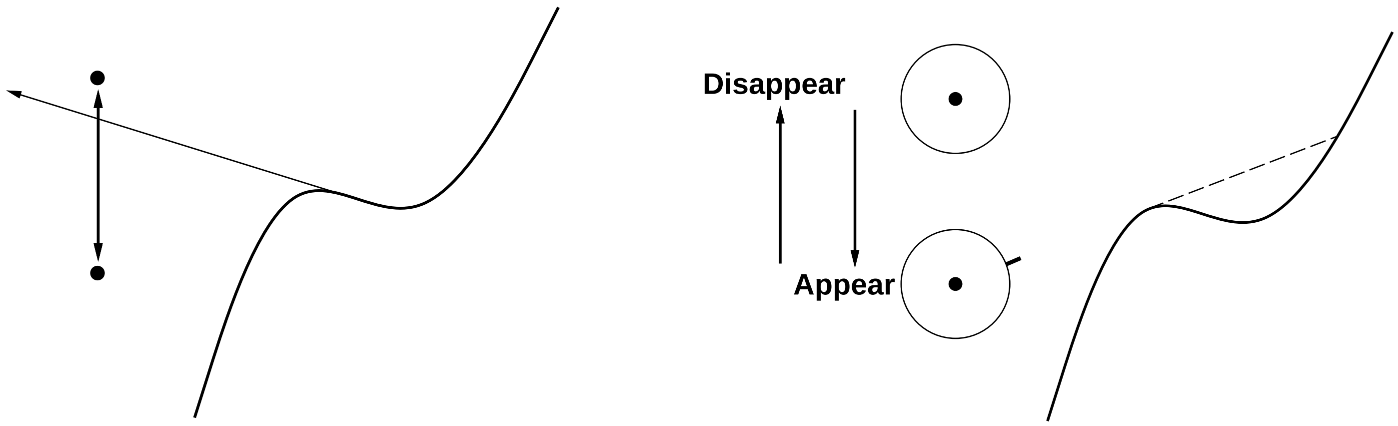
$$G_k = \{g_1, g_2, \ldots, g_n\}$$

I-space: A set of trees, $\mathcal{I}_{trees}$.

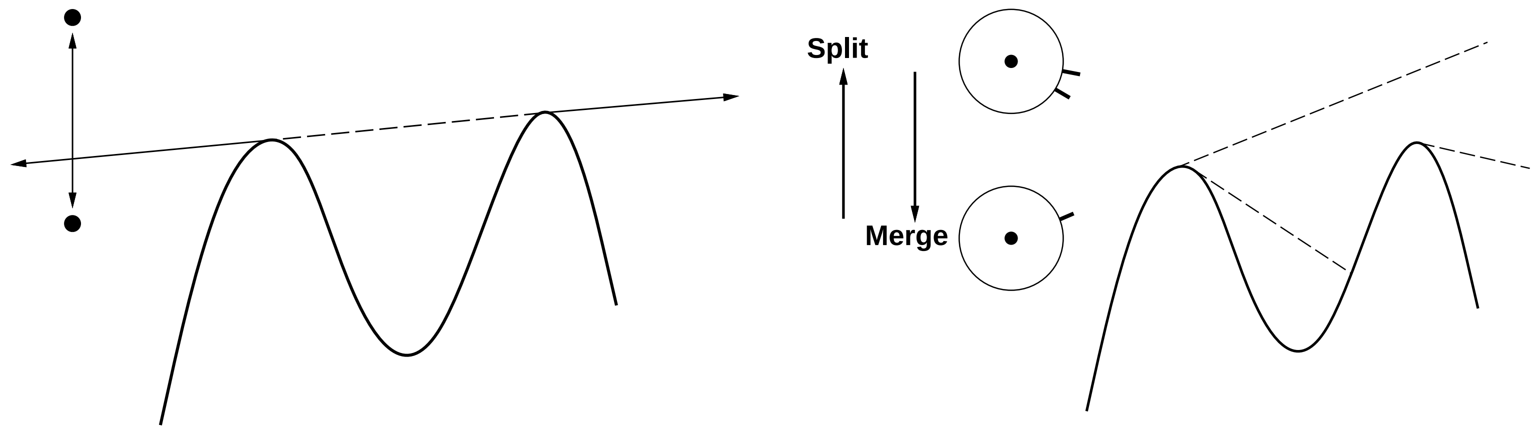
For each event, perform tree surgery:

1. **Disappear:** Delete corresponding leaf.
2. **Appear:** Insert new leaf from root.
3. **Split:** Delete child of root, raise children.
4. **Merge:** Insert child of root, lower children.

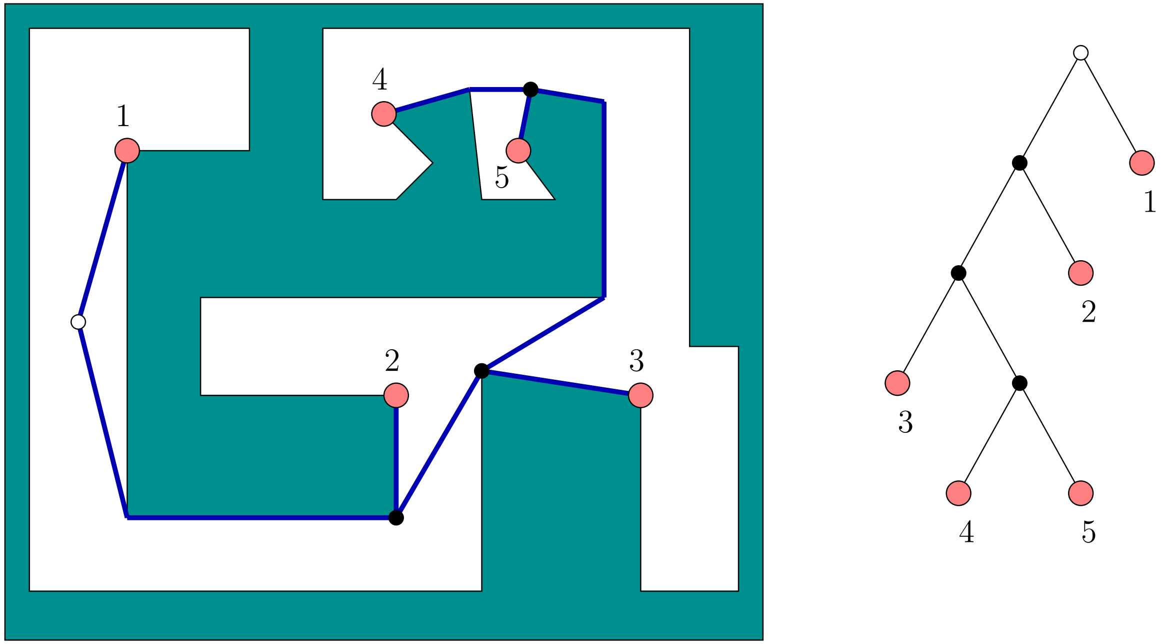
Appear and Disappear



Split and Merge

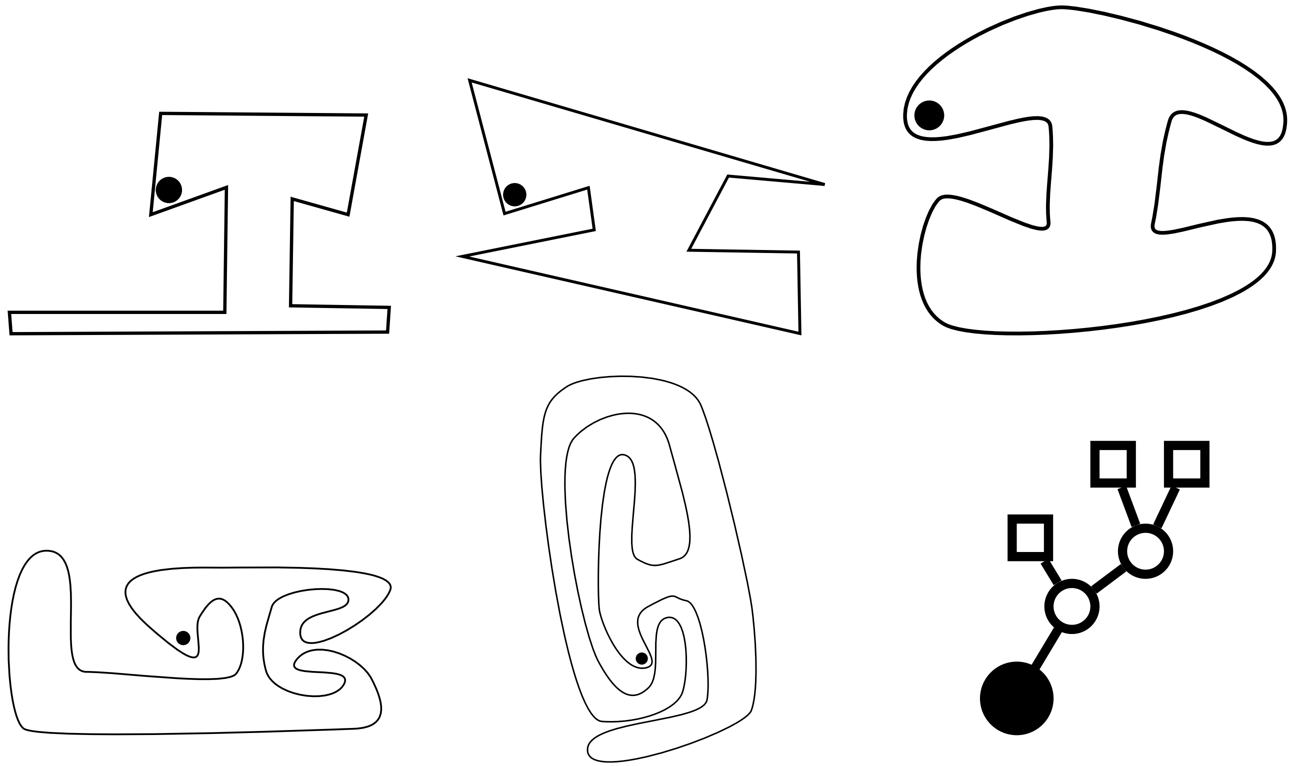


What the Filter Encodes



A piece of the shortest-path graph, as viewed from sensor position.
See shortest-path trees in Ghosh's 2007 book.

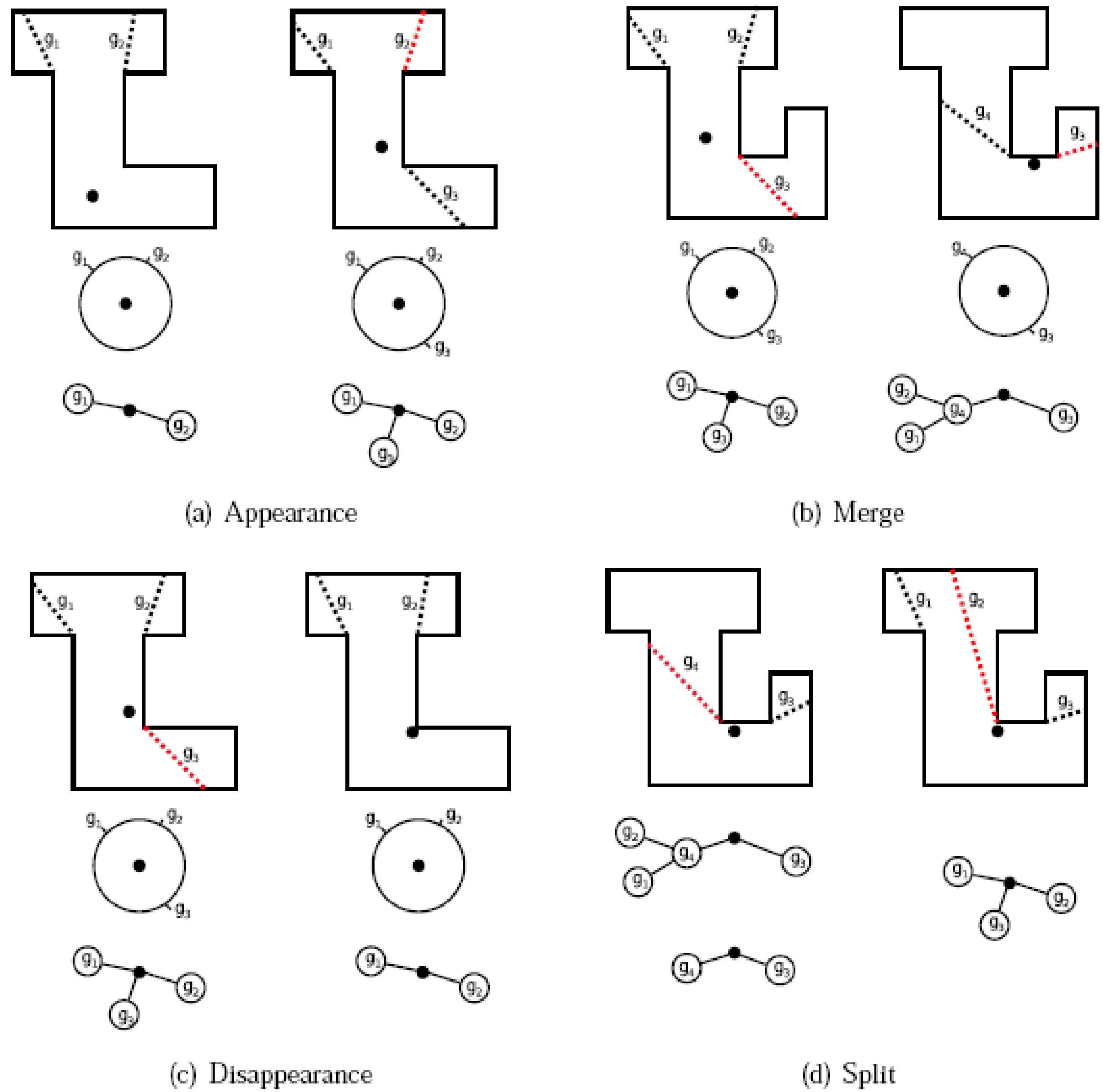
Possible Current States



Many configuraton-environment pairs have the same tree.

The robot does not have to distinguish!

How the Tree Is Updated



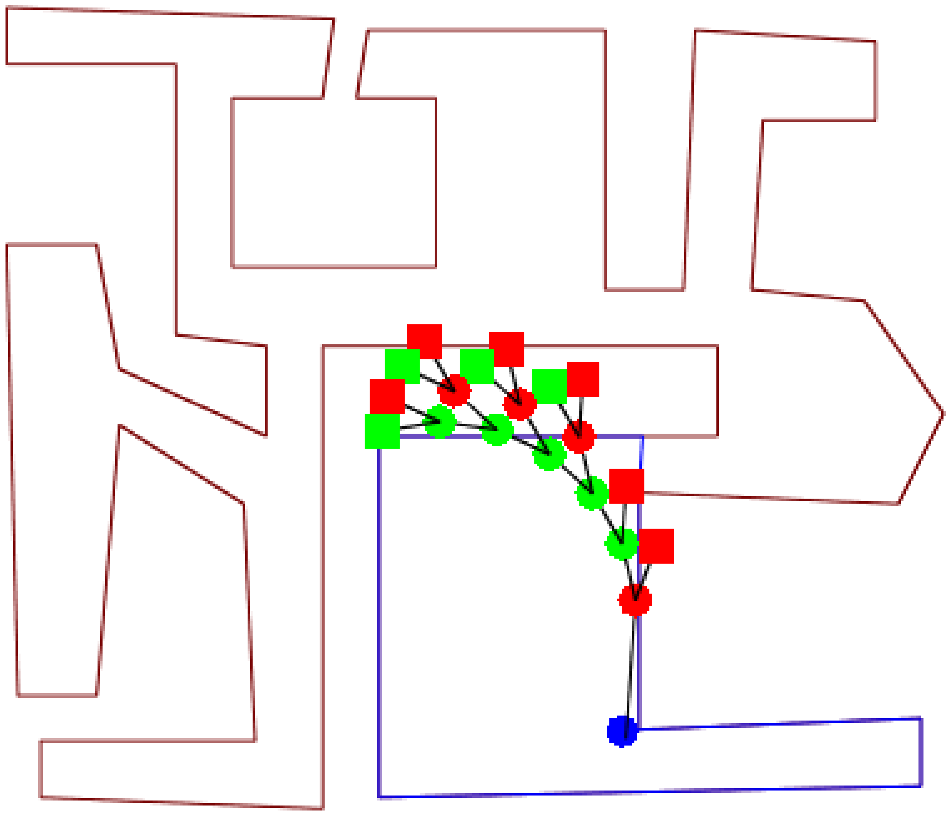
(a) Appearance

(b) Merge

(c) Disappearance

(d) Split

The Robot Can Learn a “Complete” Map



We prove that the shortest-path (visibility) graph is essentially recovered.

Part 3 Summary

- Spatial filters vs. temporal filters
- Generalized triangulaion principle: Intersect preimages
- Preimages from observation histories to state trajectory space
- Temporal filters generally walk through an I-space
- Nondeterministic vs. probabilistic vs. combinatorial filters
- Obstacles and beams, shadow I-spaces, gap trees

By defining virtual sensors and studying preimages carefully, reduced-complexity filters can be developed.

ILLINOIS
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN