Algorithmic Human-Robot Interaction

Task Planning III and Project Pitches

CSCI 7000

Prof. Brad Hayes

Computer Science Department

University of Colorado Boulder

Papers for Thursday:
Need 1 PRO (10m) and 1 CON (5m) speaker each

Designing Robot Learners that Ask Good Questions

Maya Cakmak and Andrea Thomaz

PRO presenter: Lakhan Kamireddy

CON presenter: Dhanendra Soni

Anticipating human actions for collaboration in the presence of task and sensor uncertainty

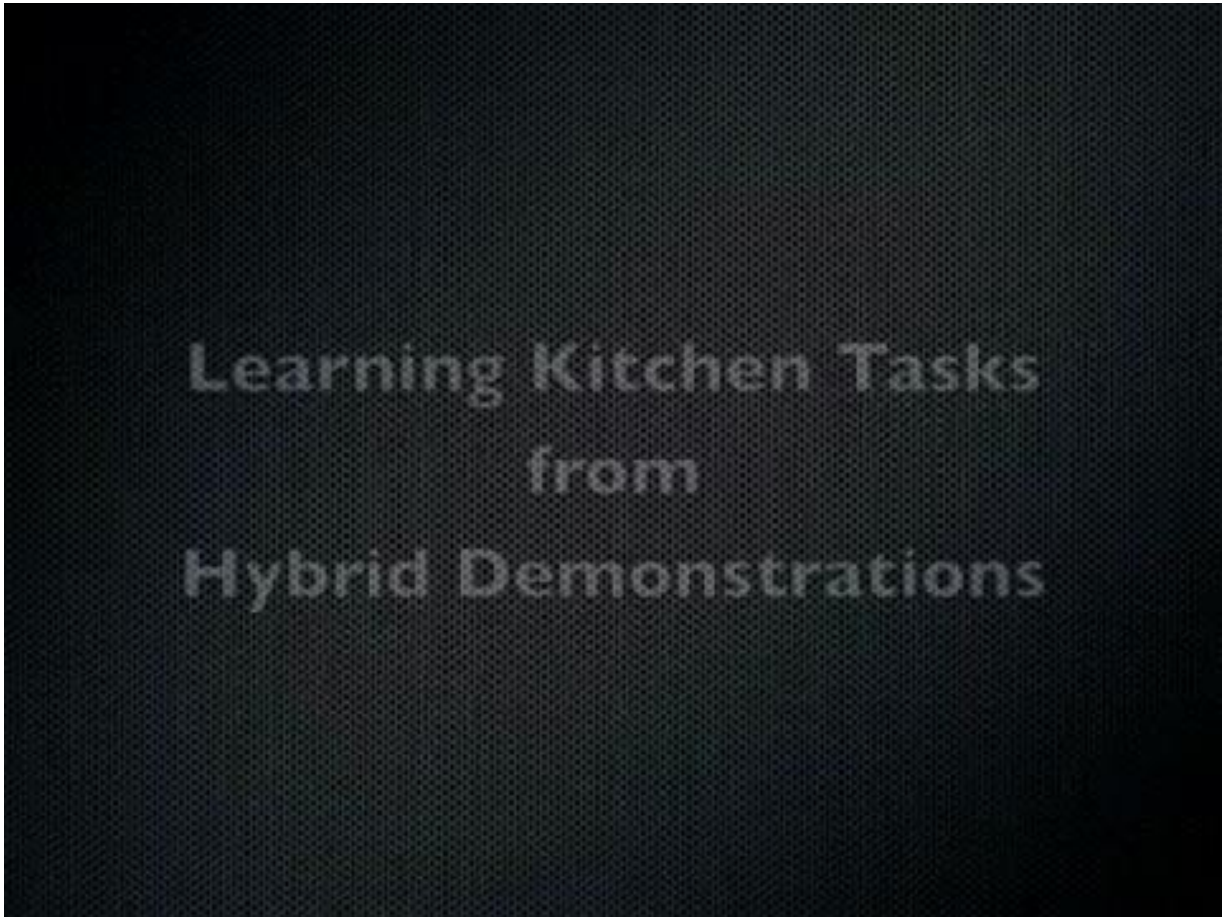
Kelsey Hawkins et al.

PRO presenter: Ian Loegfren

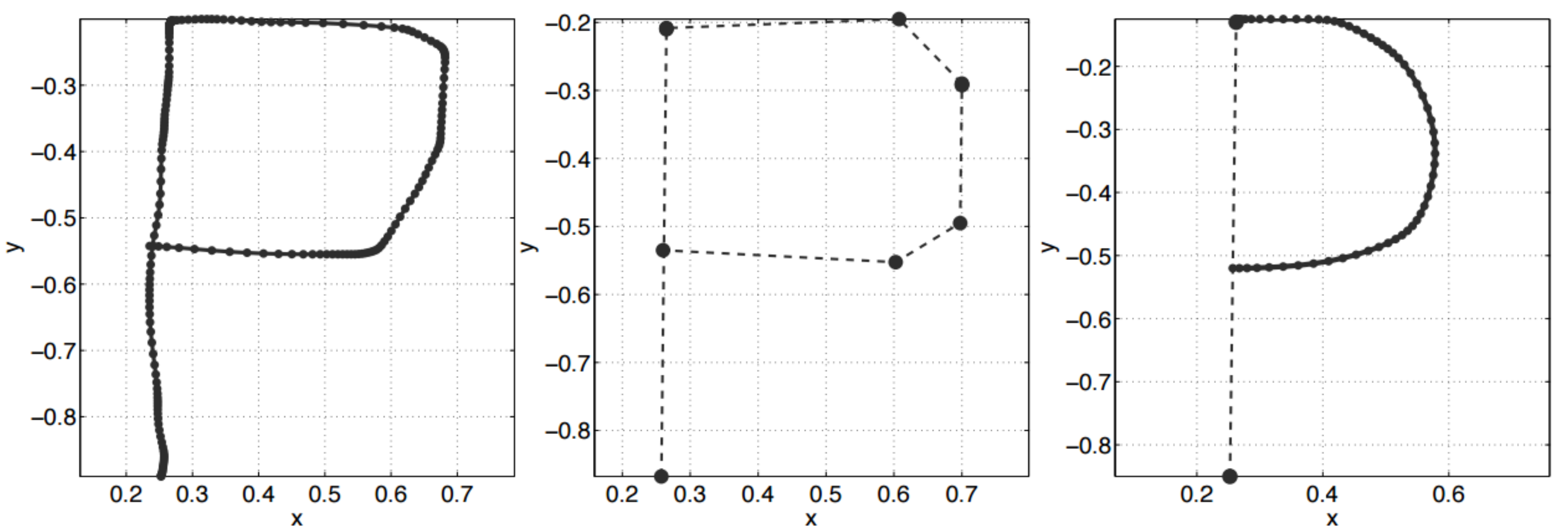
CON presenter: Nishank Sharma

Last Time…

Keyframe and Trajectory Demonstrations



Trajectories and Keyframes for Kinesthetic Teaching: A Human-Robot Interaction Perspective



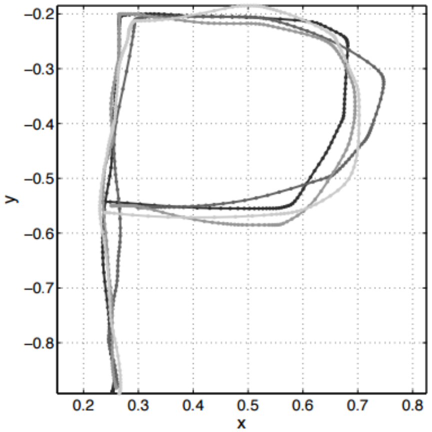
Trajectory Demonstration

Keyframe Demonstration

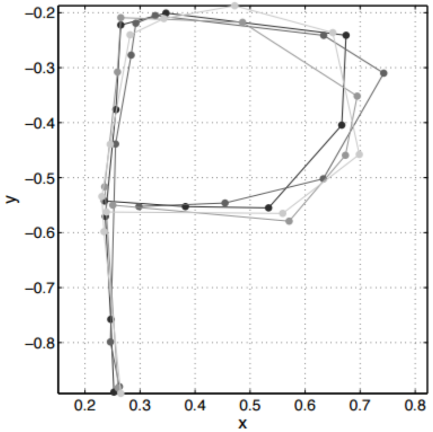
Hybrid Demonstration

Sample demonstrations of the letter P in 2D

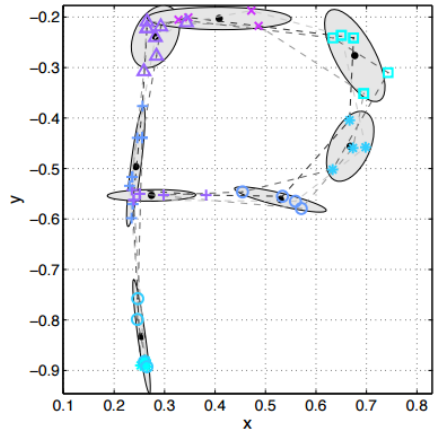
Trajectory Conversion



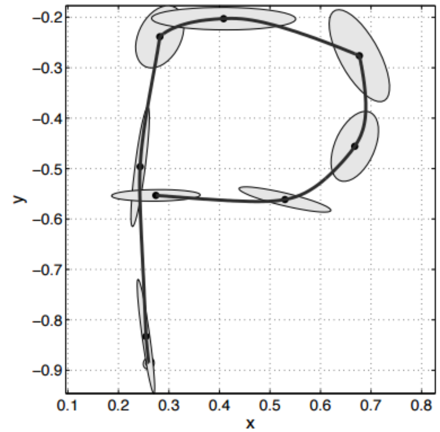
Continuous trajectories in 2D



Data converted to keyframes



Clustering of keyframes and the sequential pose distributions



Learned model trajectory

Trajectory Conversion: Forward-Inverse Relaxation Model

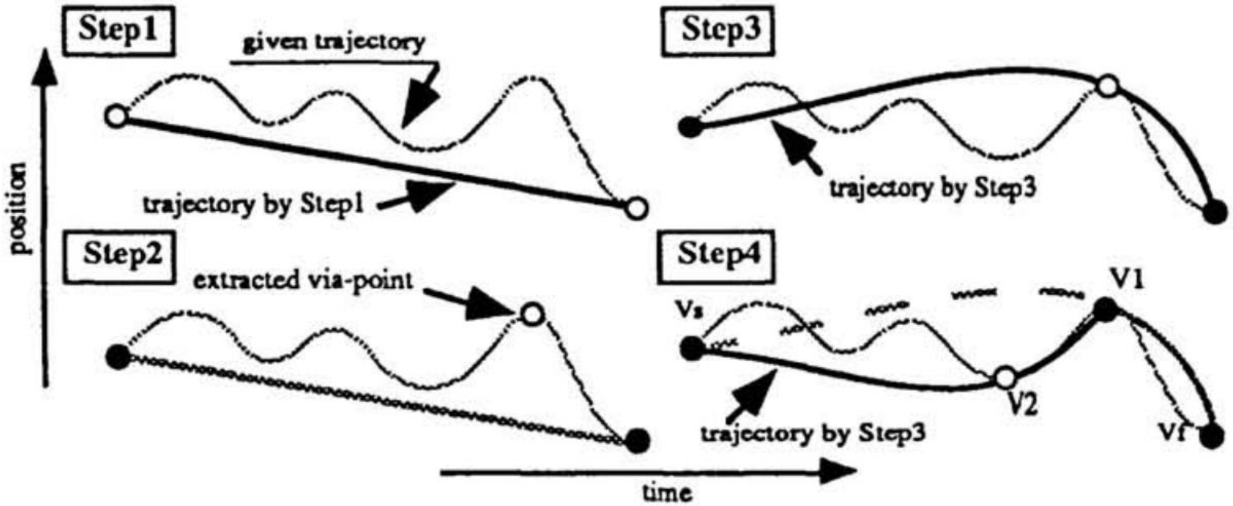
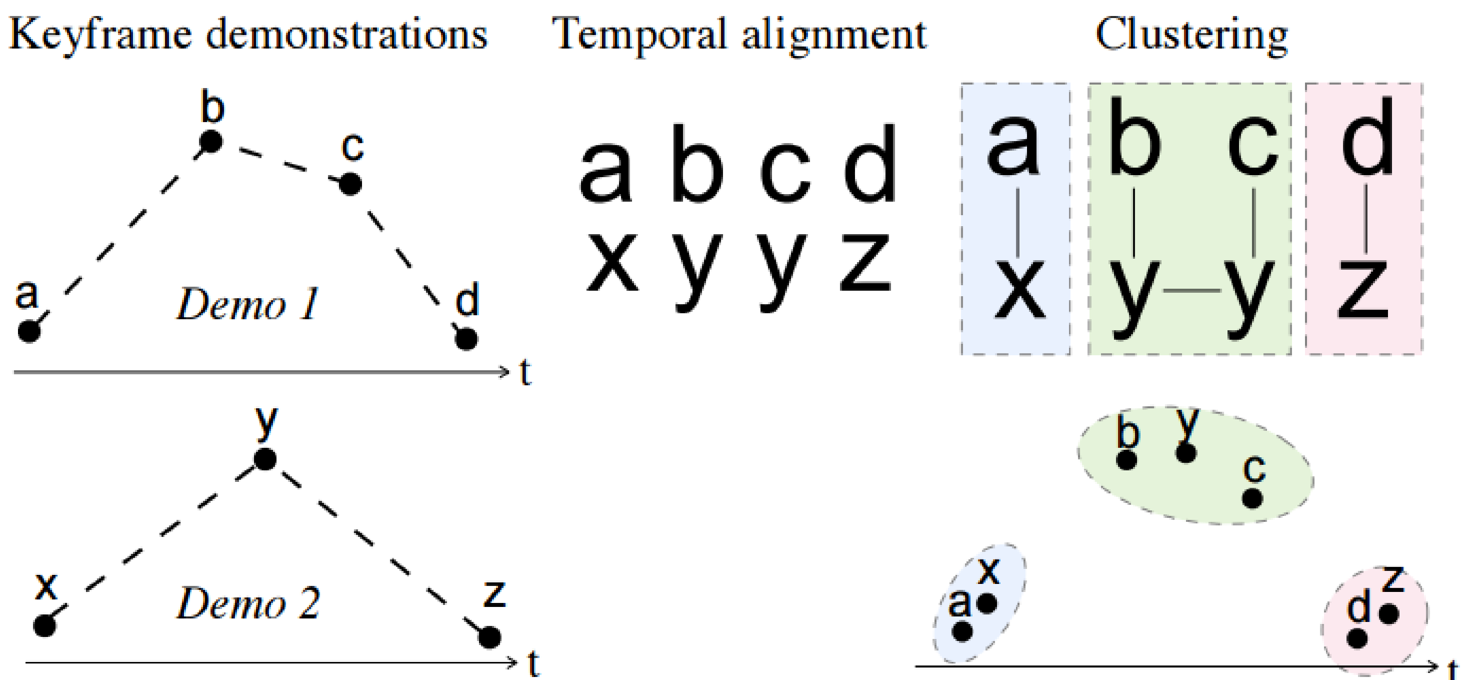


Figure 3: An algorithm for extracting via-points.

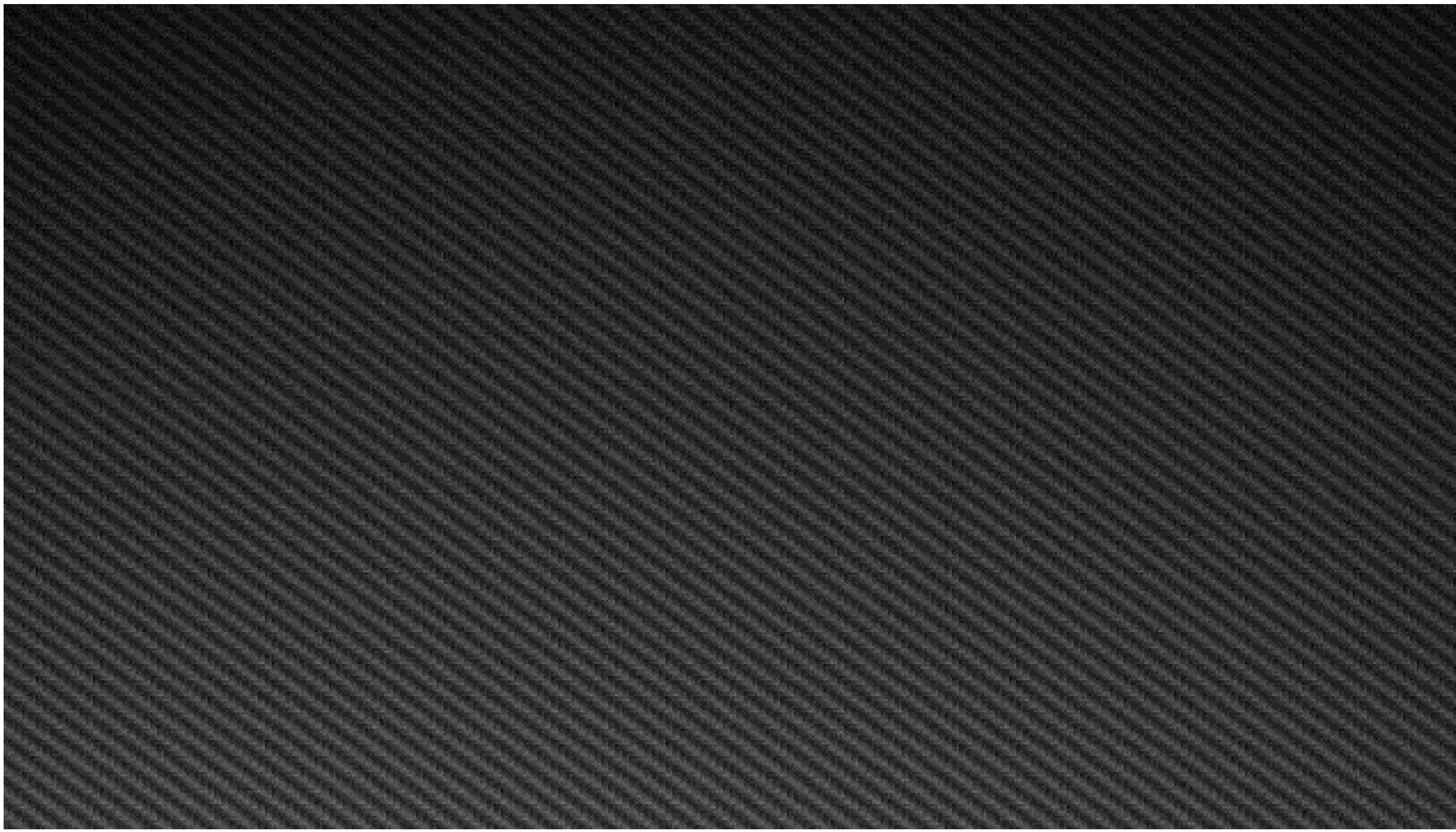
- Fifth order splines used between positions to minimize jerk, using position, velocity, and acceleration per keyframe to compute the spline unknowns.
- Keyframes assume zero velocity/acceleration per point
- Trajectory demonstrations use the means from cluster centers.

A Computational Model for Cursive Handwriting Based on the Minimization Principle – Wada et al.

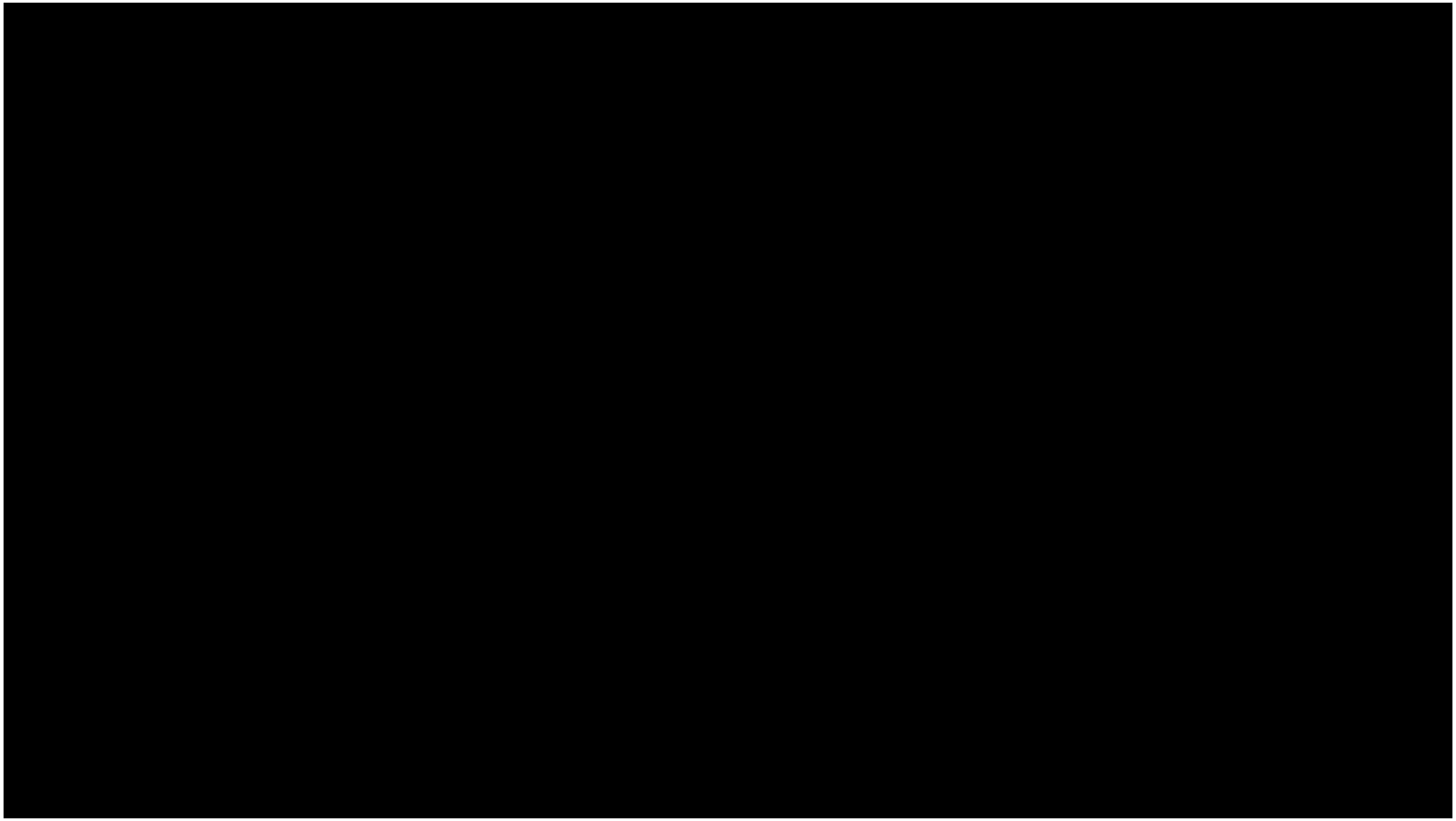
Aligning Multiple Demonstrations



Planning human-aware motions using a sampling-based costmap planner



Extending this work with human motion models



STRIPS: Language for Classical Planning

- A **problem** in Strips is a tuple $P = \langle F, O, I, G \rangle$:
  - F stands for set of all **atoms** (boolean vars)
  - O stands for set of all **operators** (actions)
  - $I \subseteq F$ stands for **initial situation**
  - $G \subseteq F$ stands for **goal situation**

- Operators $o \in O$ **represented** by
  - the **Add** list $Add(o) \subseteq F$
  - the **Delete** list $Del(o) \subseteq F$
  - the **Precondition** list $Pre(o) \subseteq F$

- Pickup(X)
  - P: grip(∅) ∧ clear(X) ∧ ontable(X)
  - A: grip(X)
  - D: onTable(X) ∧ grip(∅)

Planning with Markov Decision Processes

MDPs are **fully observable, probabilistic** state models:

- a state space S
- initial state $s_0 \in S$
- a set $G \subseteq S$ of goal states
- actions $A(s) \subseteq A$ applicable in each state $s \in S$
- **transition probabilities** $P_a(s'|s)$ for $s \in S$ and $a \in A(s)$
- action costs $c(a, s) > 0$

– **Solutions** are **functions (policies)** mapping states into actions

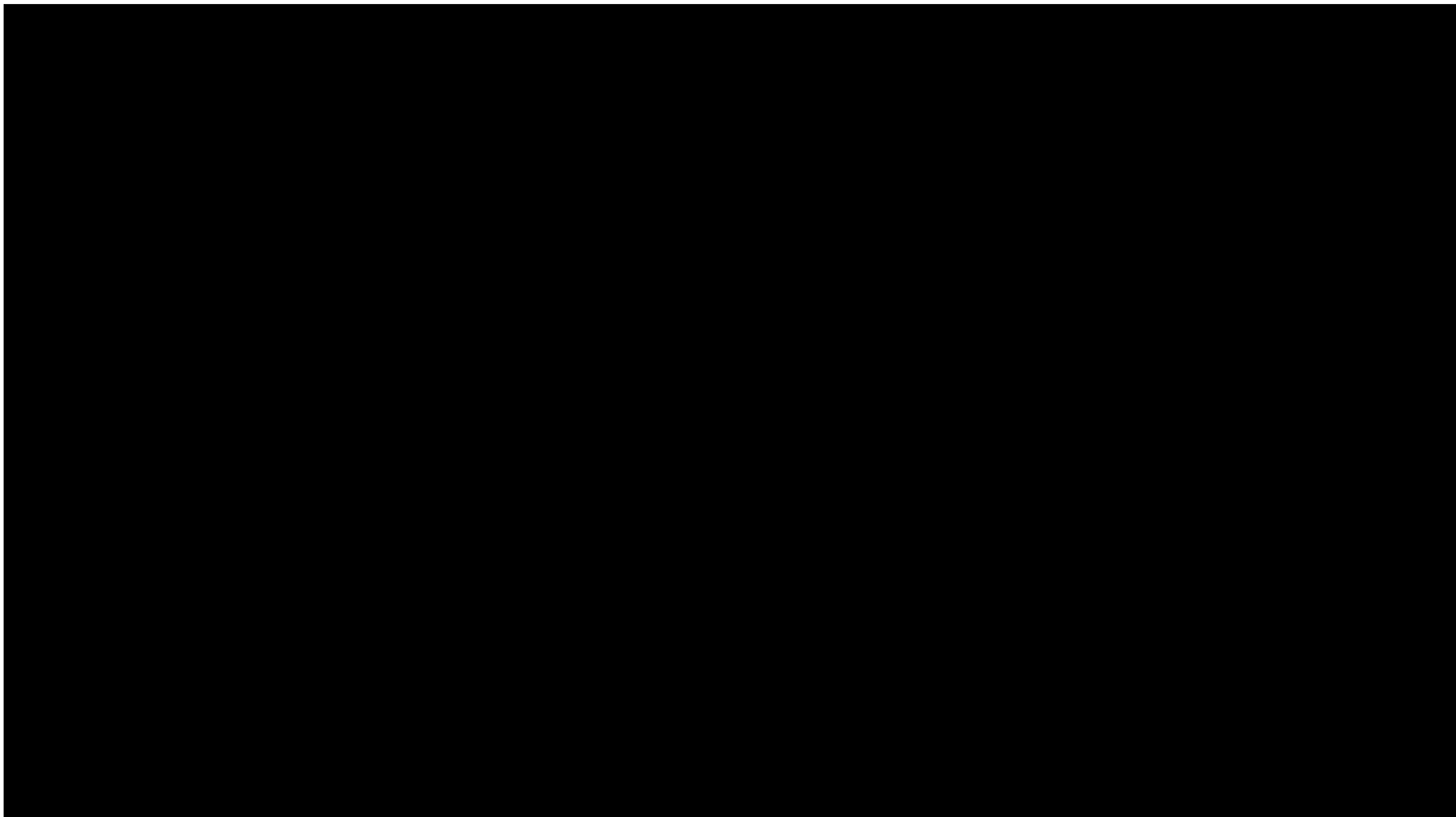
– **Optimal** solutions minimize **expected cost** to goal

Partially Observable MDPs (POMDPs)

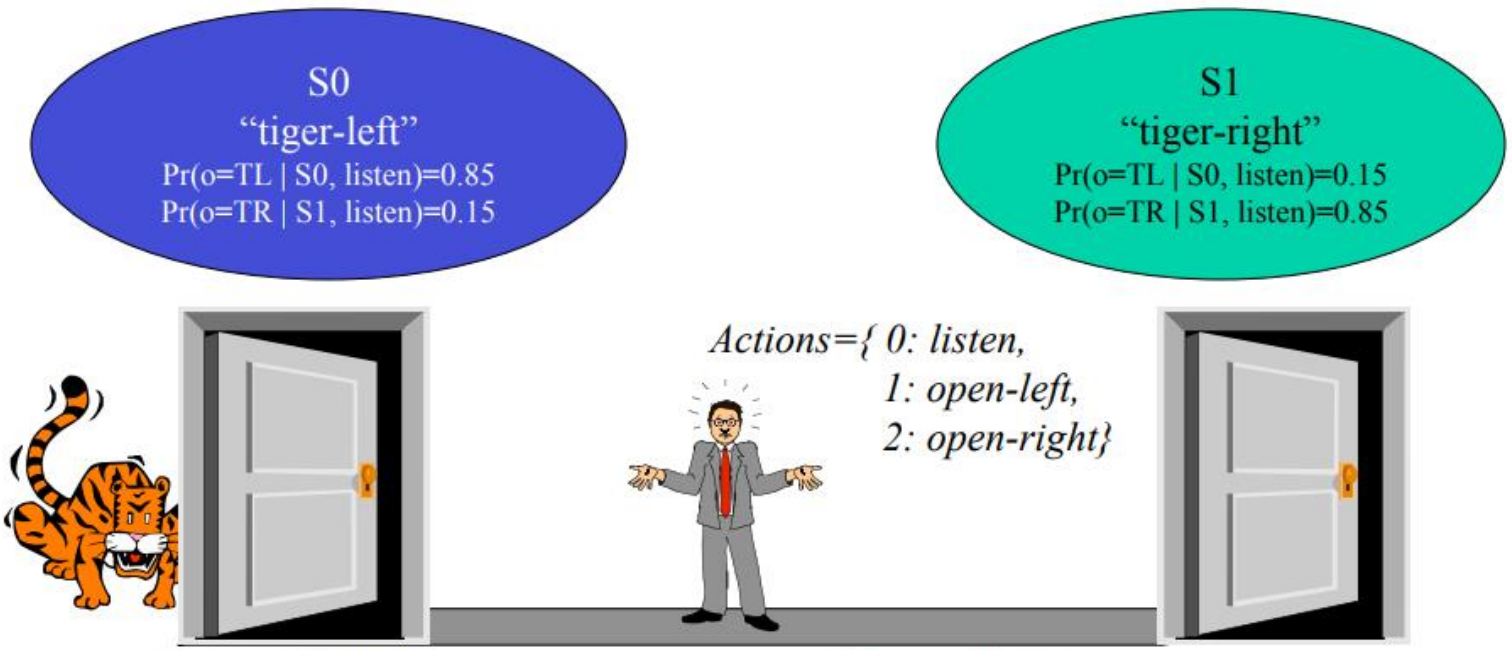
- Traditional MDPs are defined with:
  - States – S = {(0,0), (0,1), …}
  - Actions – A = {move_north, …}
  - Rewards – R(s,a,s’) -> Reward
  - Transition Probabilities – T(s,a,s’) -> P(s’| s, a)
- Now we have to add:
  - Observation set – $\mathrm{O} = \{o_1, o_2, \dots\}$
  - Observation prob. – $\Omega = P(o_1, \dots, o_i \mid S)$
- Also have to augment:
  - Current State (now *belief*) – B = [0.1, 0.6, 0.2, 0.1, ...]
  - Policy *(no longer $S \rightarrow A$)* – $\pi: B \rightarrow A$

POMDPs in Action

POMDPs in Action



POMDP: The Tiger Problem



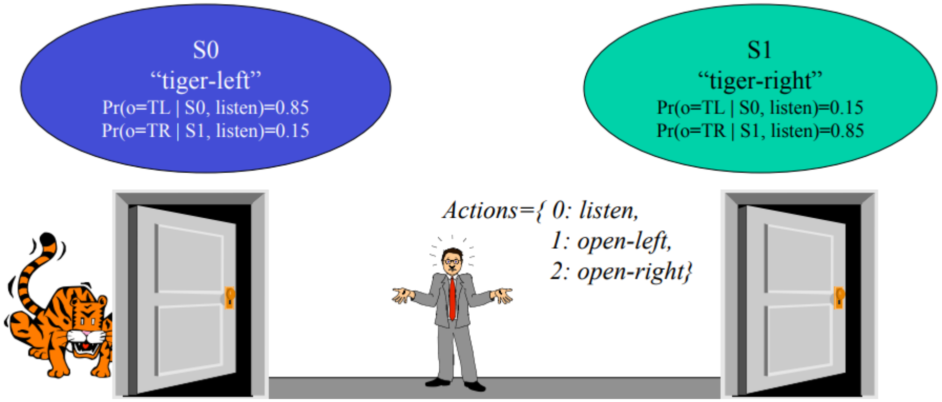
Reward Function

- *Penalty for wrong opening: -100*
- *Reward for correct opening: +10*
- *Cost for listening action: -1*

Observations

- *to hear the tiger on the left (TL)*
- *to hear the tiger on the right(TR)*

POMDP: The Tiger Problem



Prob. (LISTEN)	Tiger: left	Tiger: right
Tiger: left	1.0	0.0
Tiger: right	0.0	1.0

Prob. (LISTEN)	O: TL	O: TR
Tiger: left	0.85	0.15
Tiger: right	0.15	0.85

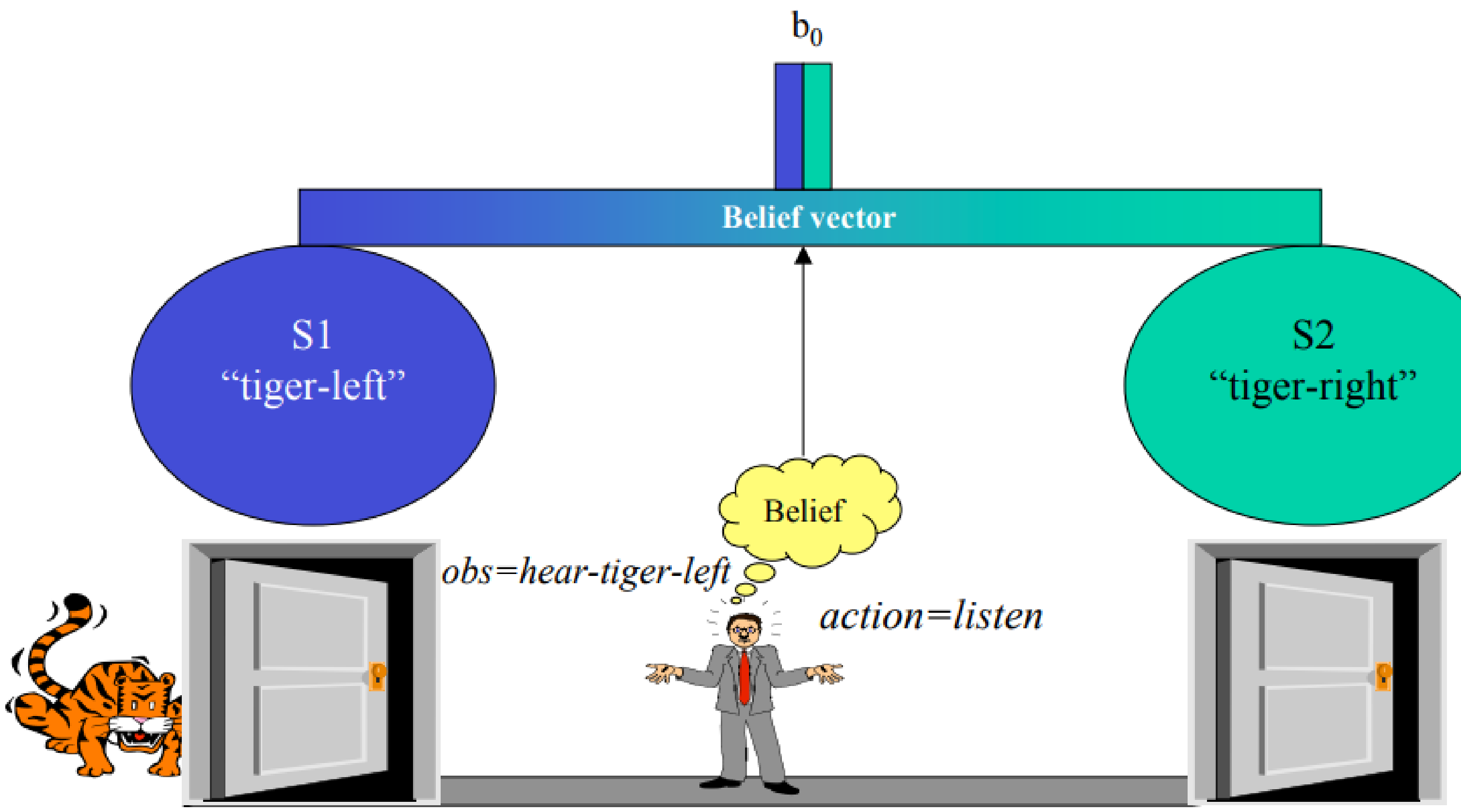
Prob. (LEFT)	Tiger: left	Tiger: right
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

Prob. (LEFT)	O: TL	O: TR
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

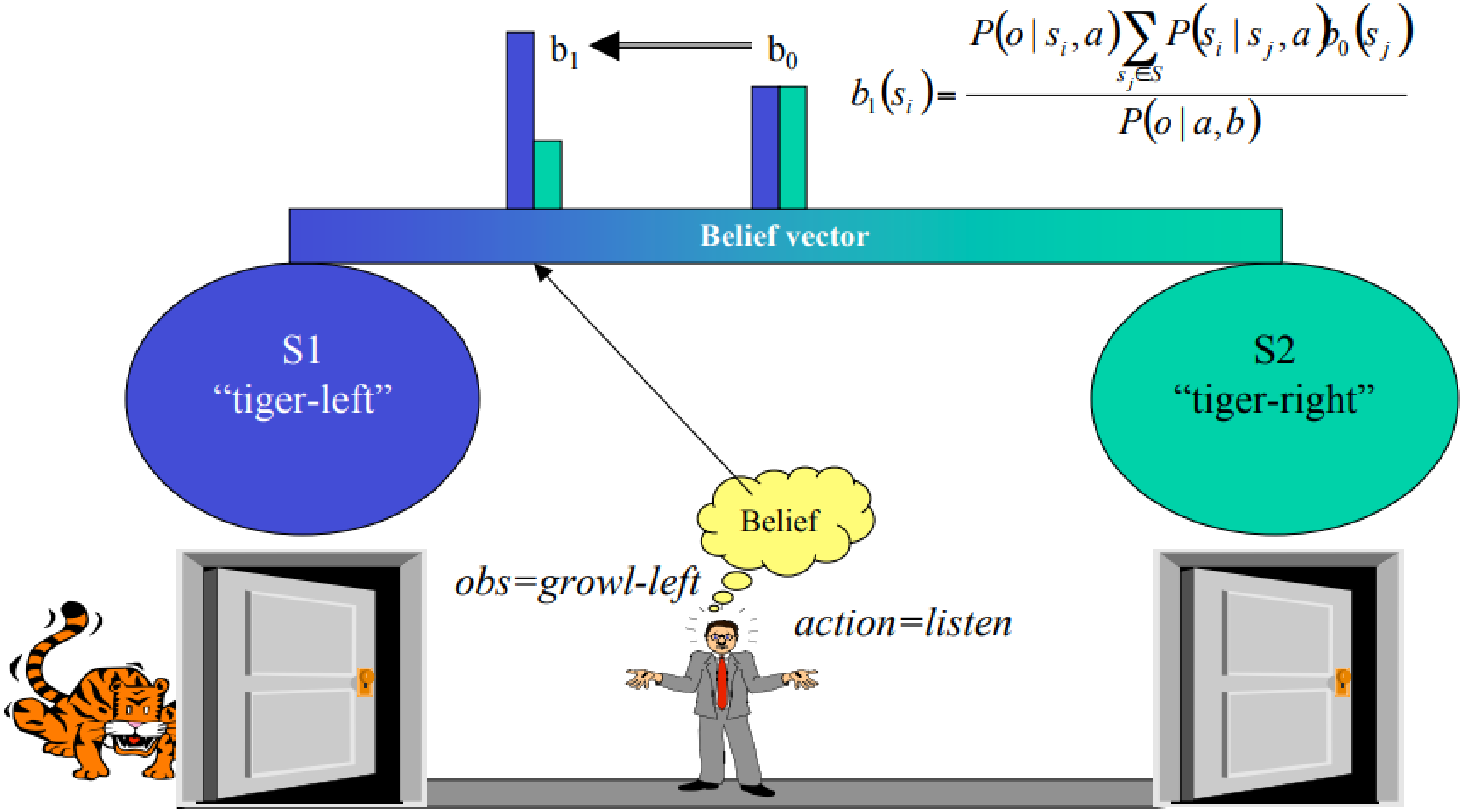
Prob. (RIGHT)	Tiger: left	Tiger: right
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

Prob. (LEFT)	O: TL	O: TR
Tiger: left	0.5	0.5
Tiger: right	0.5	0.5

POMDP: The Tiger Problem

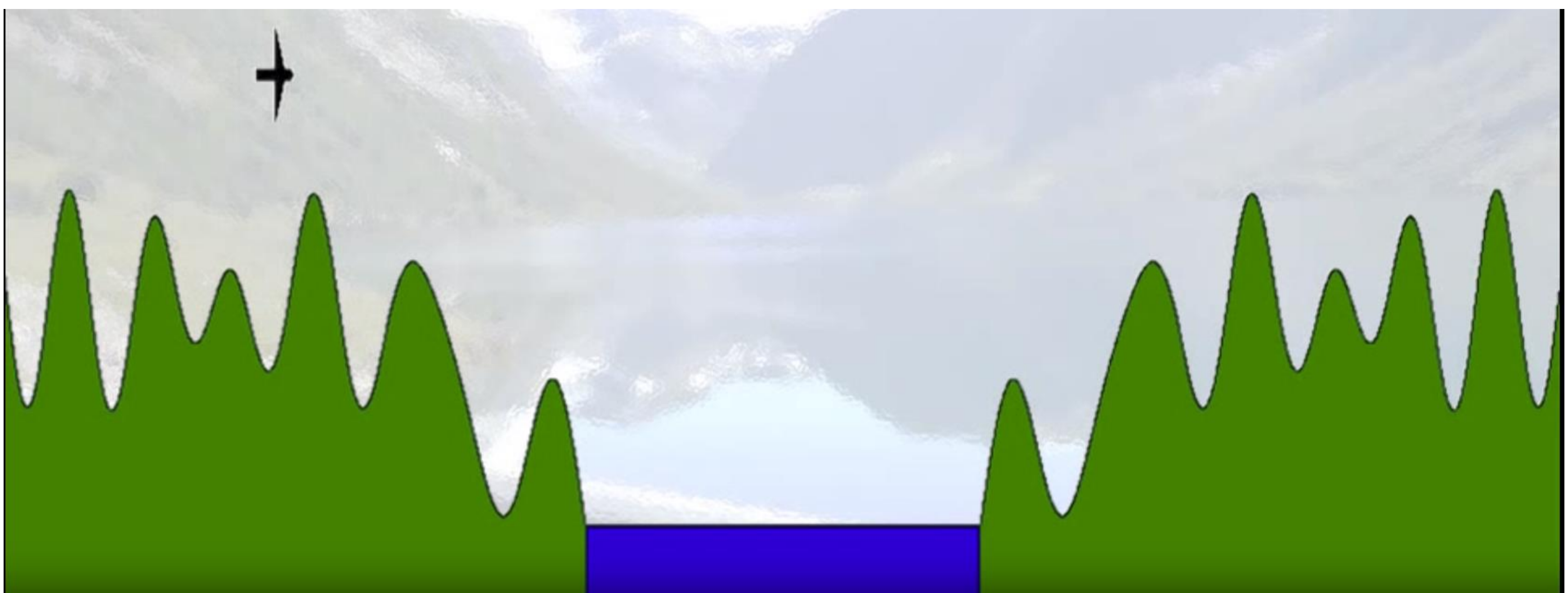


POMDP: The Tiger Problem



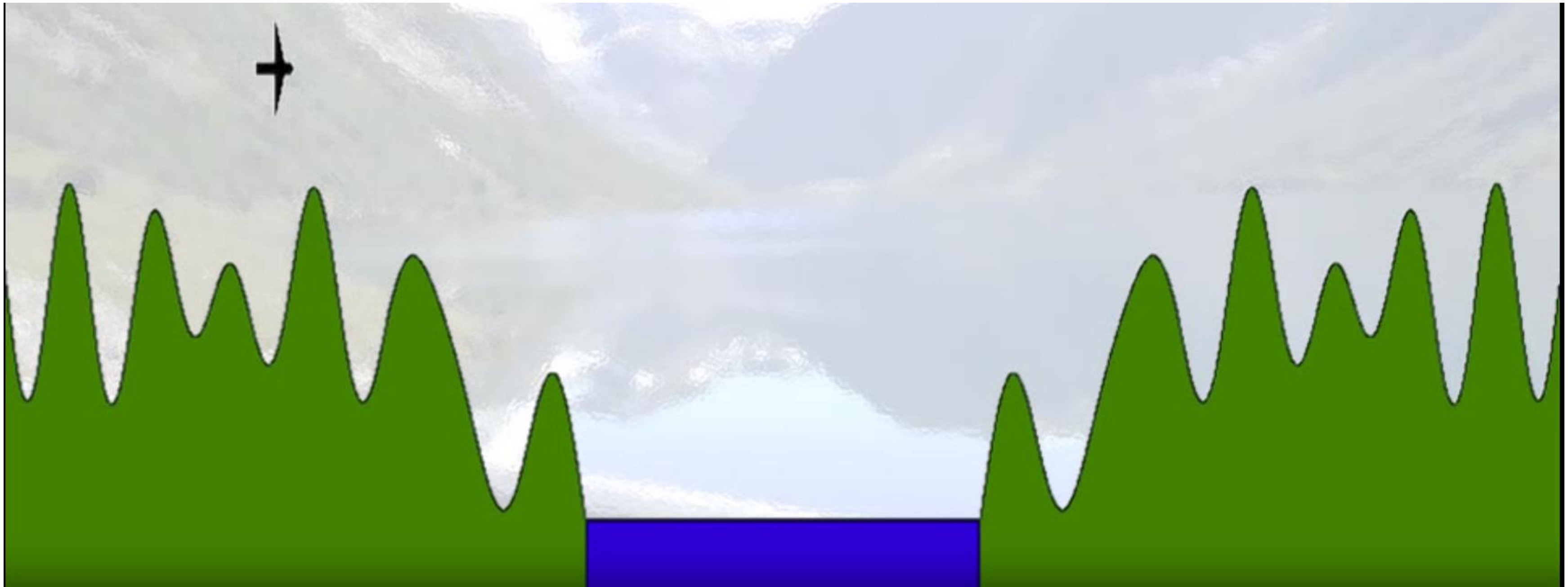
Still not feasible to exhaustively check all of our hypotheses!

- But we can still make progress!
- Need a mechanism to enable us to mostly pursue *reasonably likely* hypotheses.
- Example domain: Localization!



Localization Domain

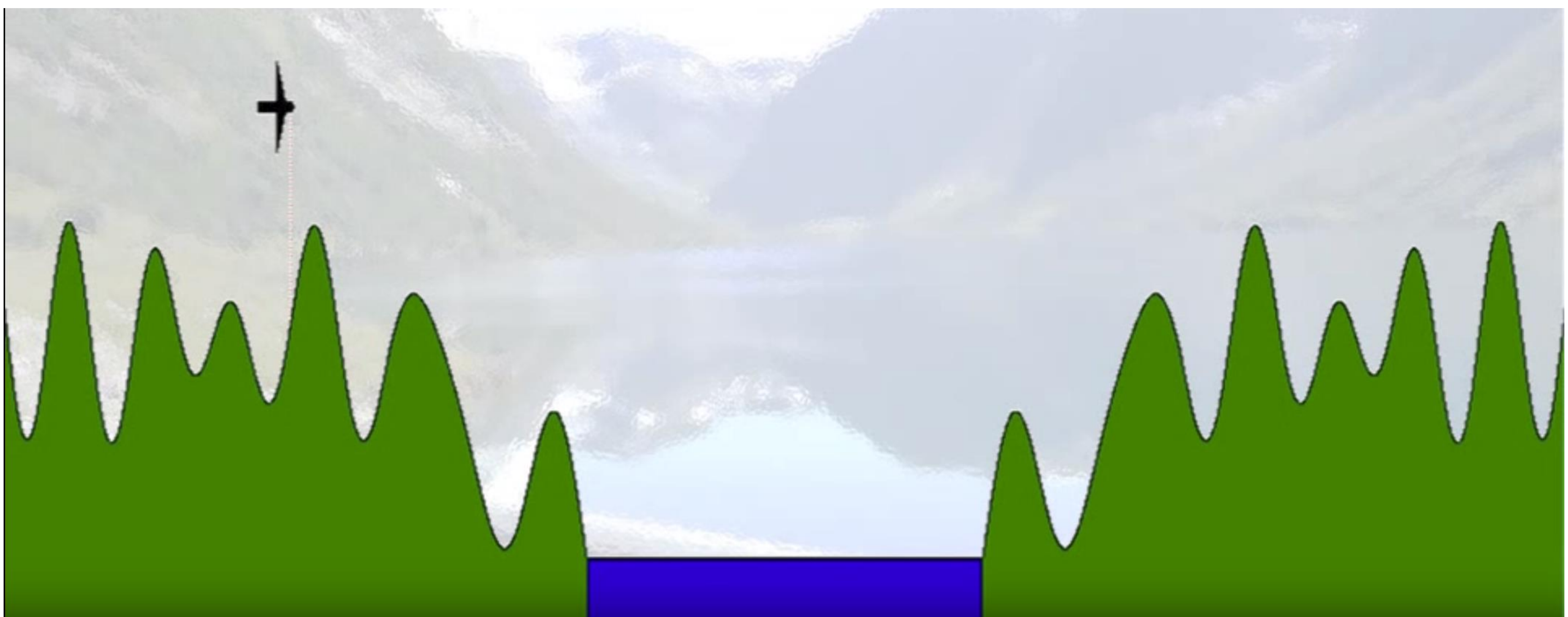
- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (with some error)
- **Known information:** Airspeed (with some error), Map
- **Goal:** Use repeated observations to find true horizontal position



Particle Filter

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

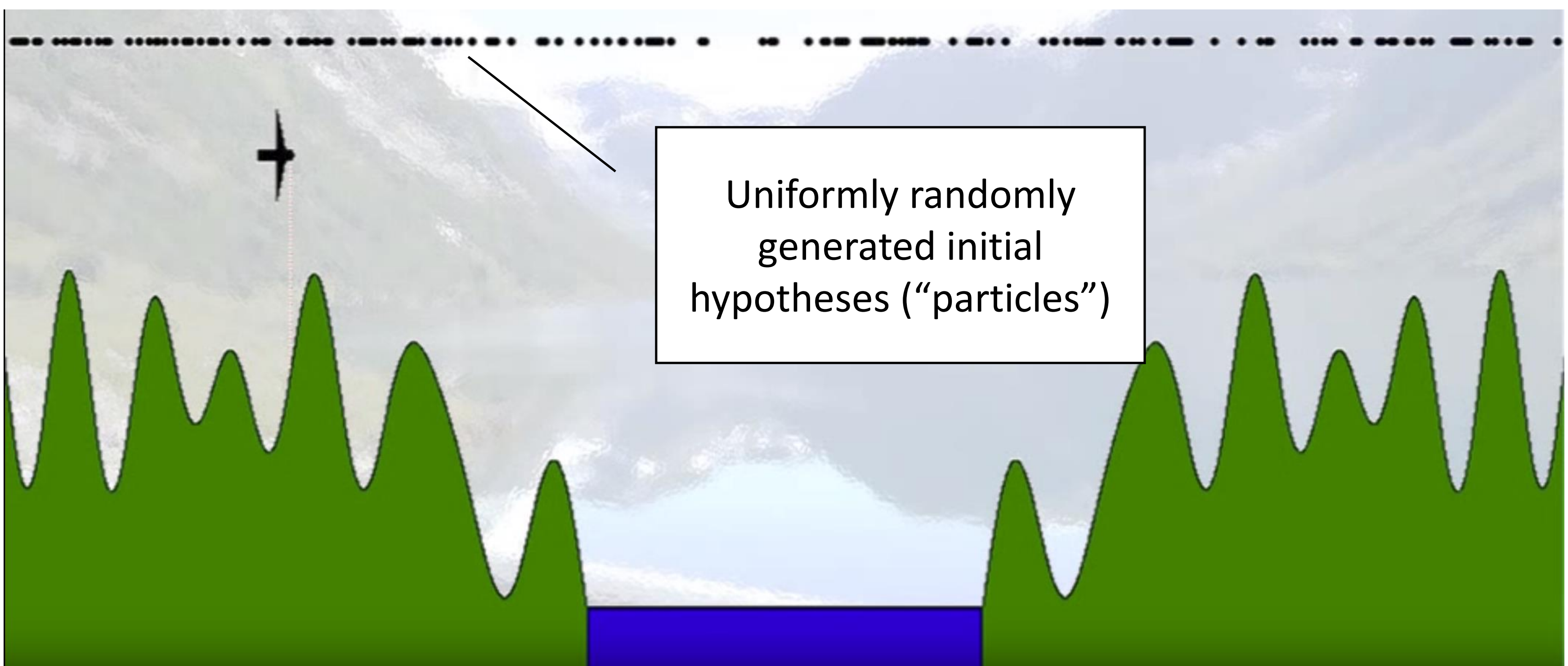
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Initial Hypotheses

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

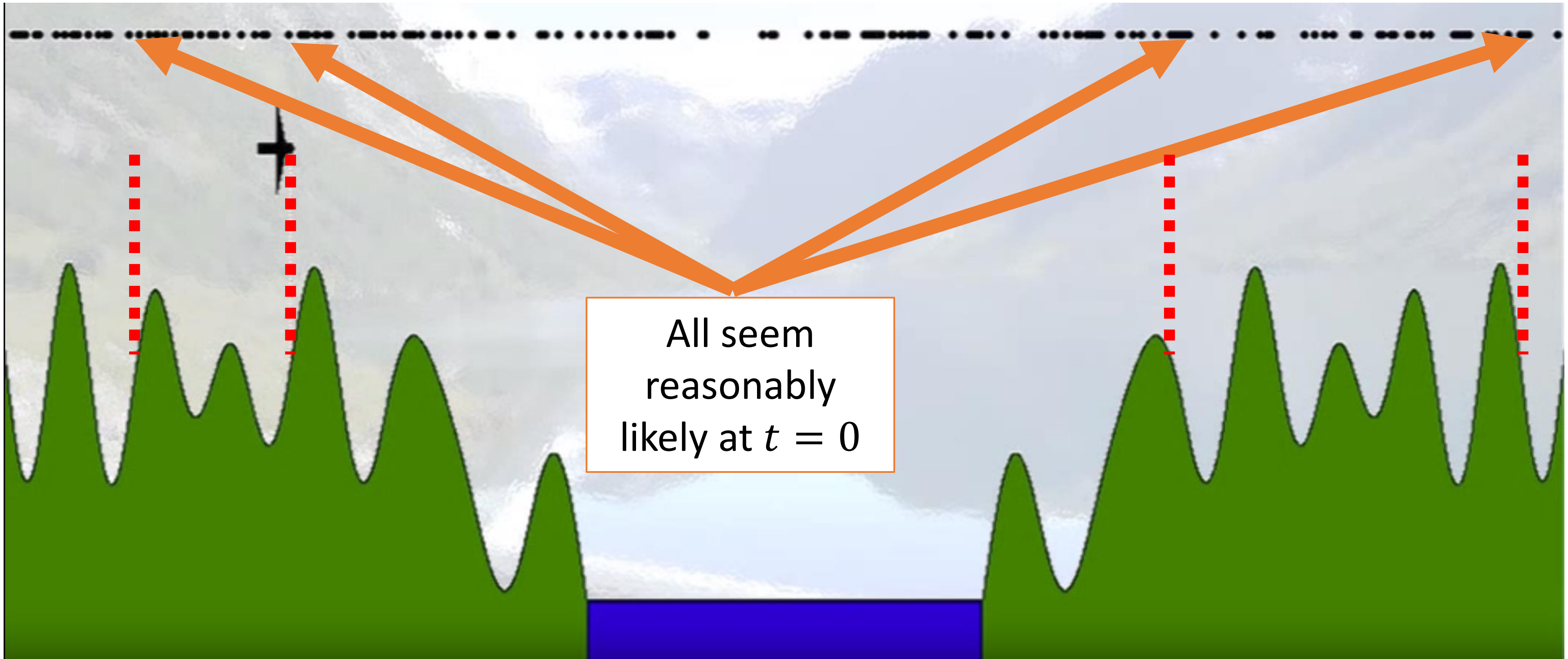
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Likelihood Evaluation

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

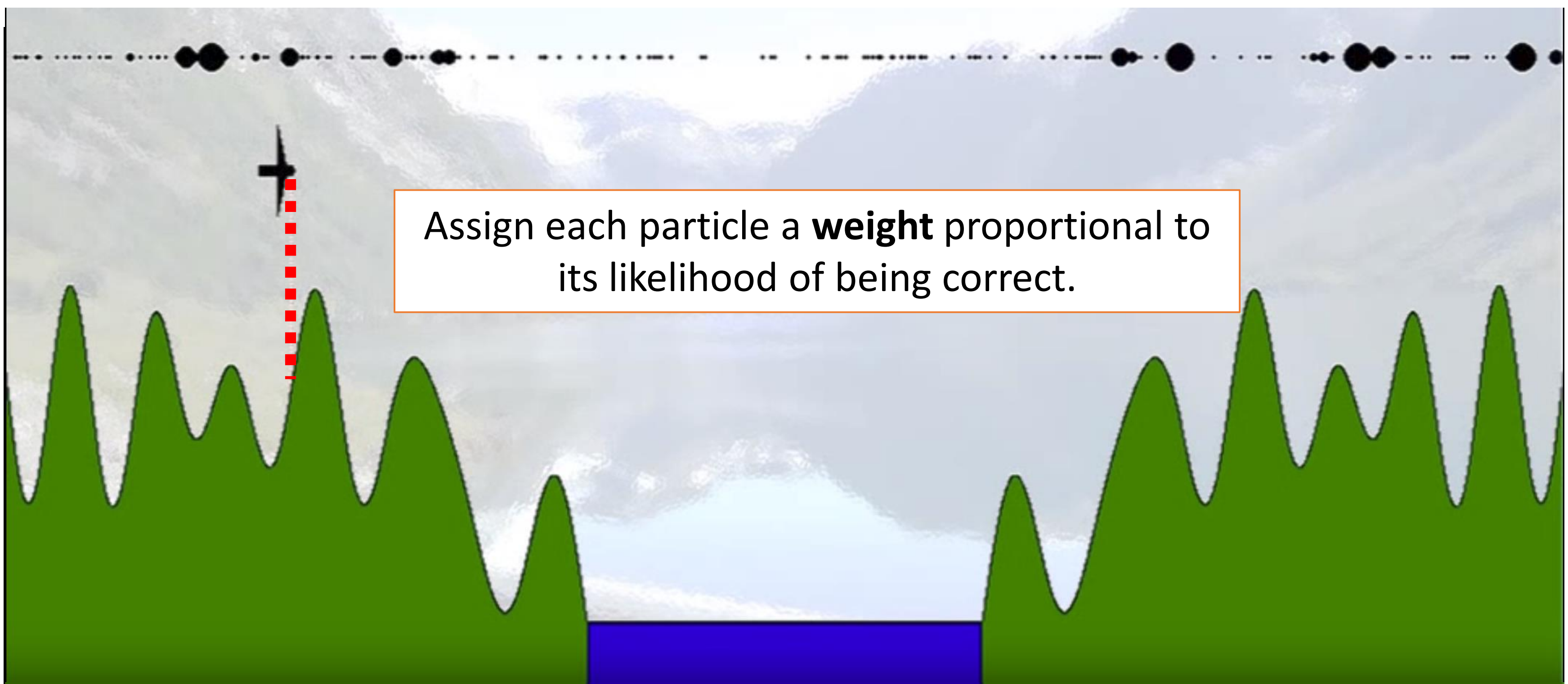
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Weighting Particles

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

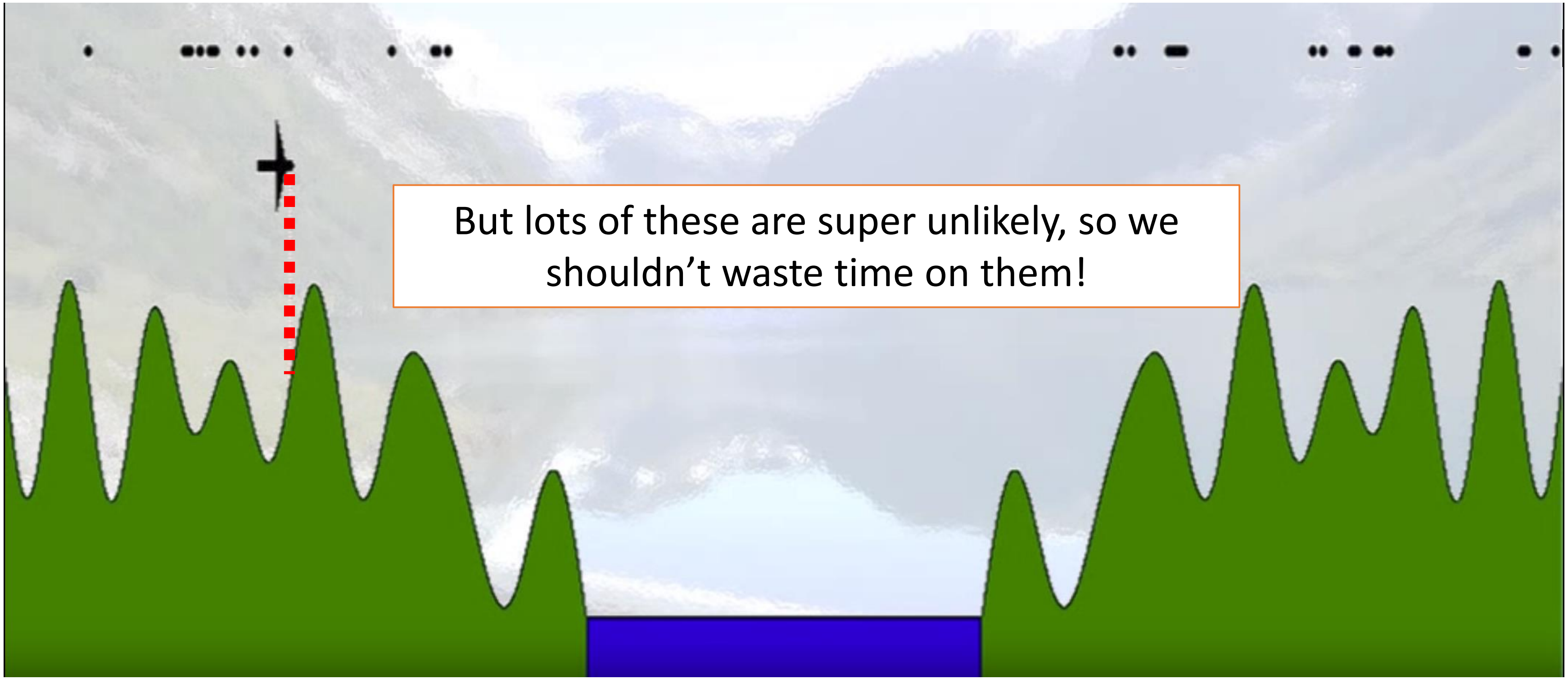
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Exhausting Particles

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.

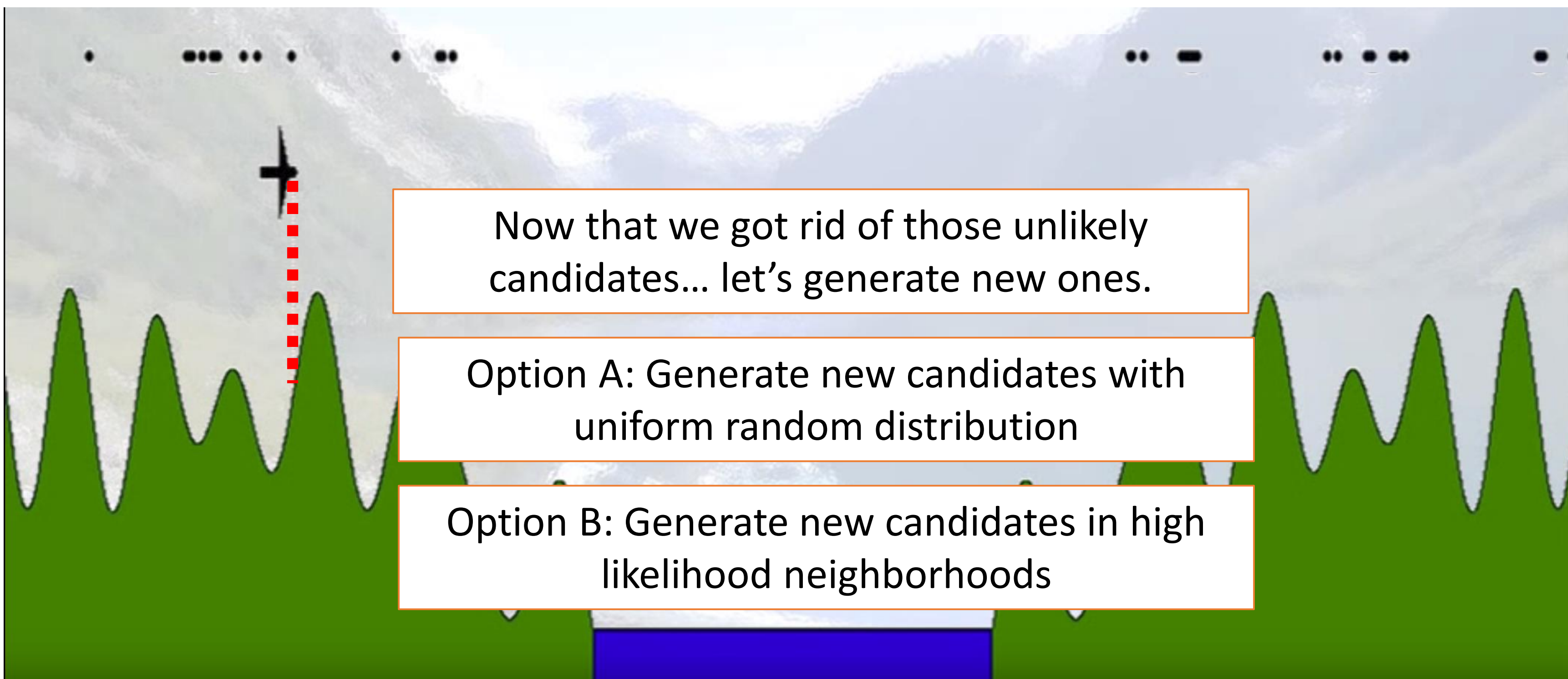


But lots of these are super unlikely, so we shouldn't waste time on them!

Particle Filter: Resampling Particles

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

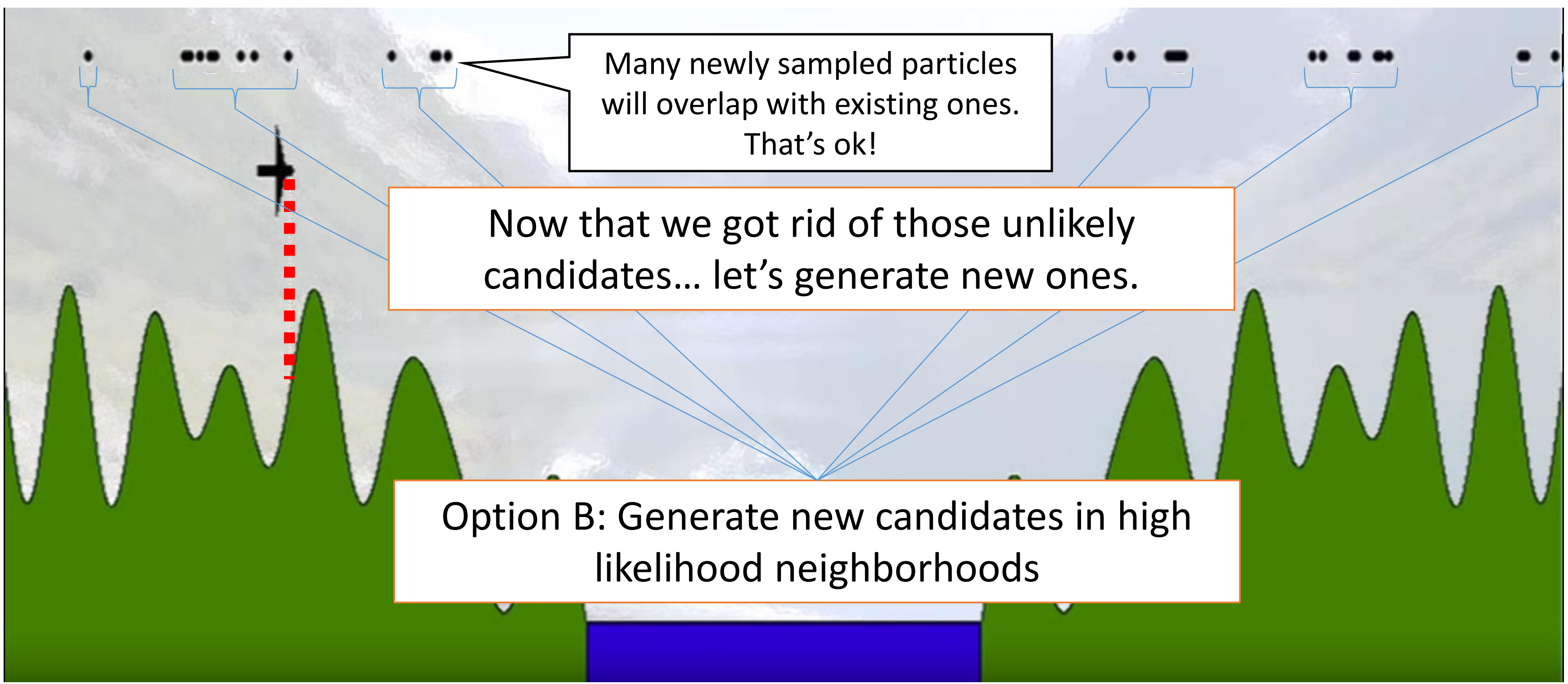
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Resampling Particles

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

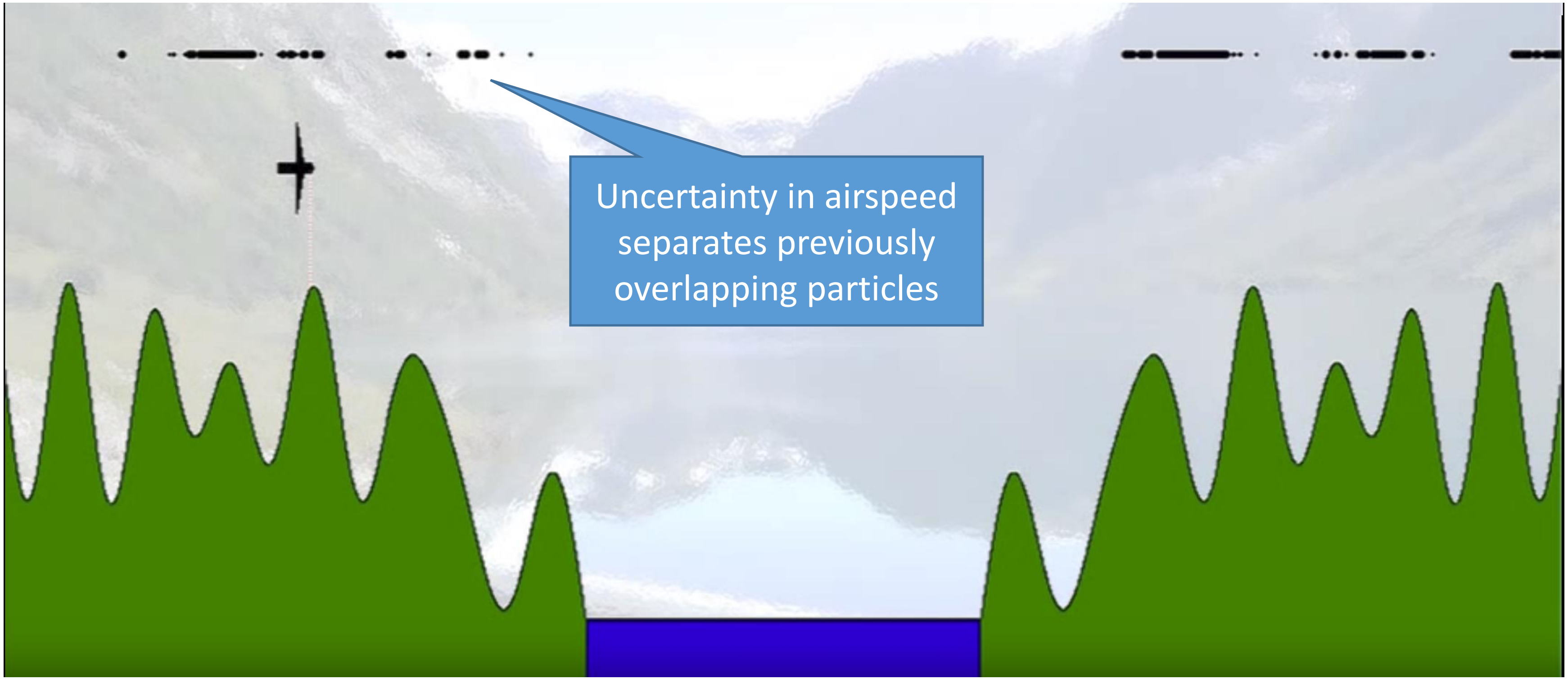
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Advance 1 Timestep

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

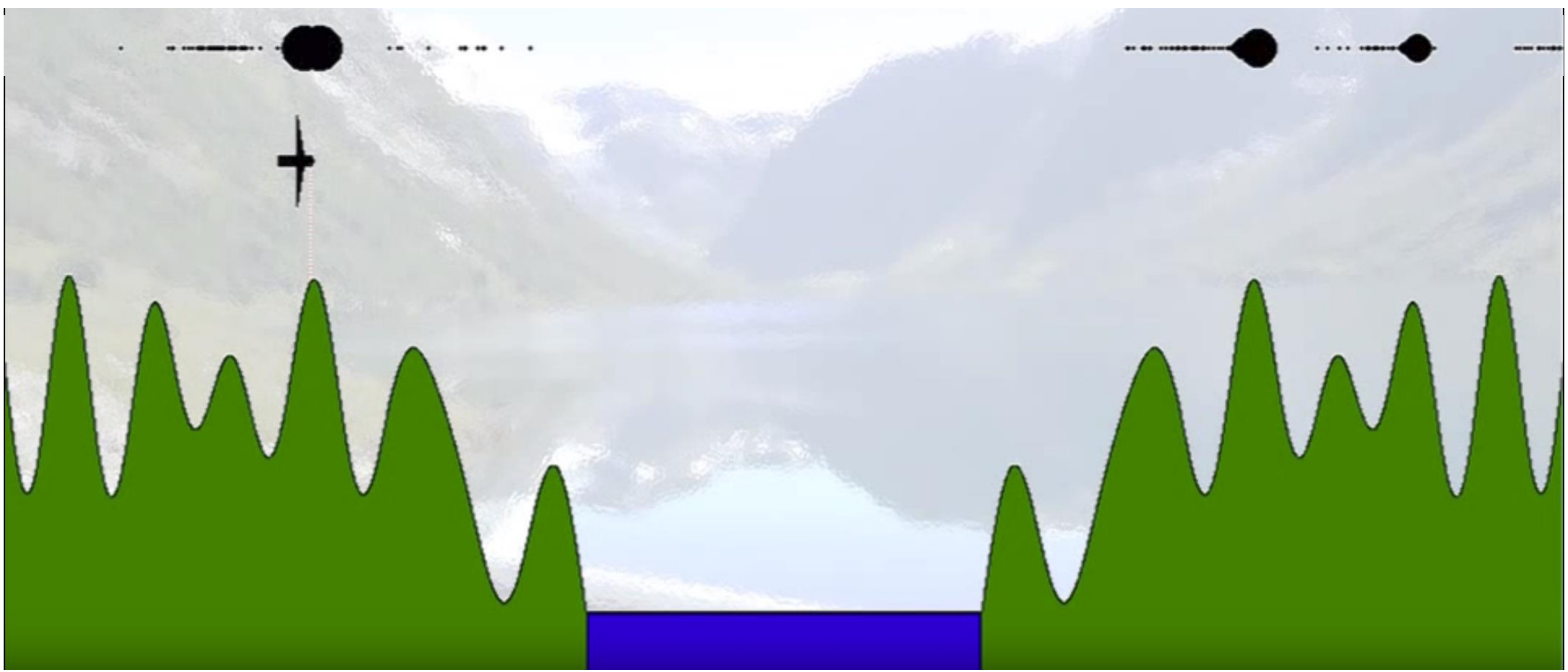
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Update Weights

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

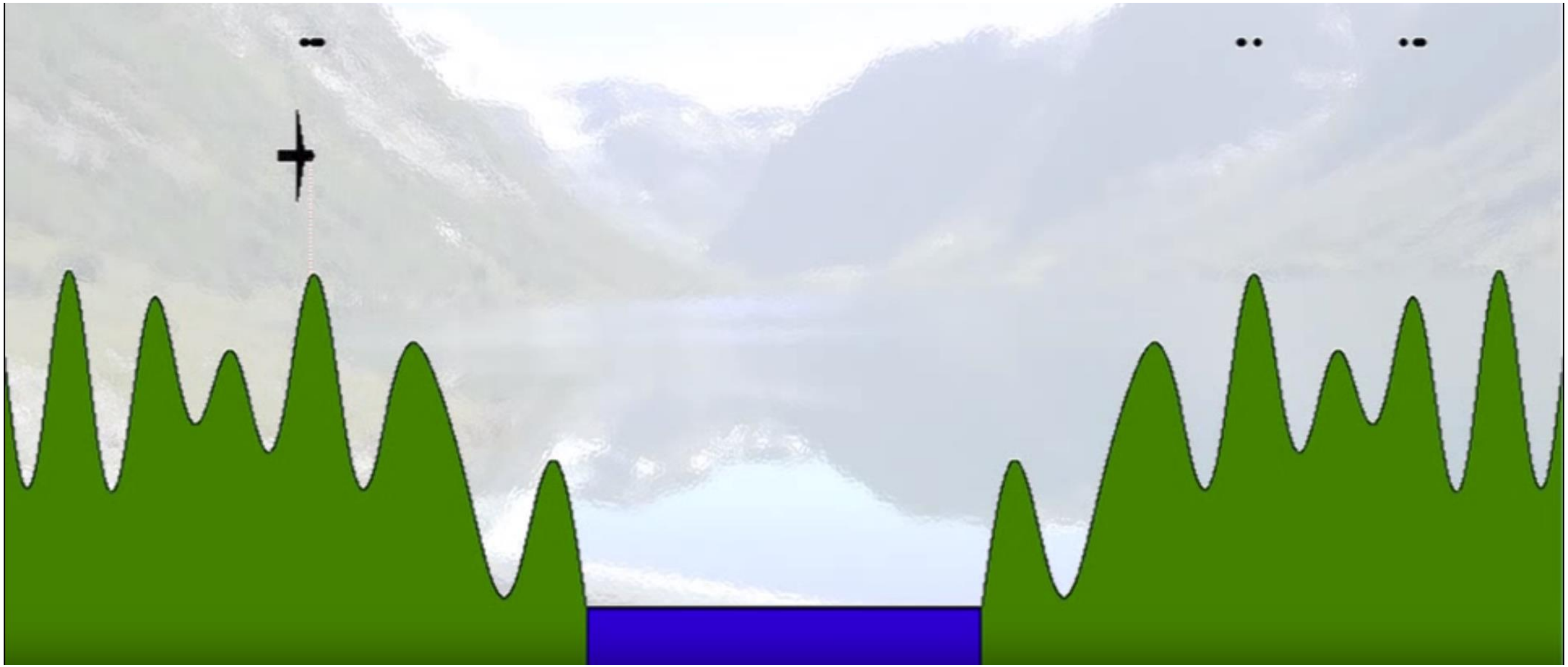
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Exhaust/Resample

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

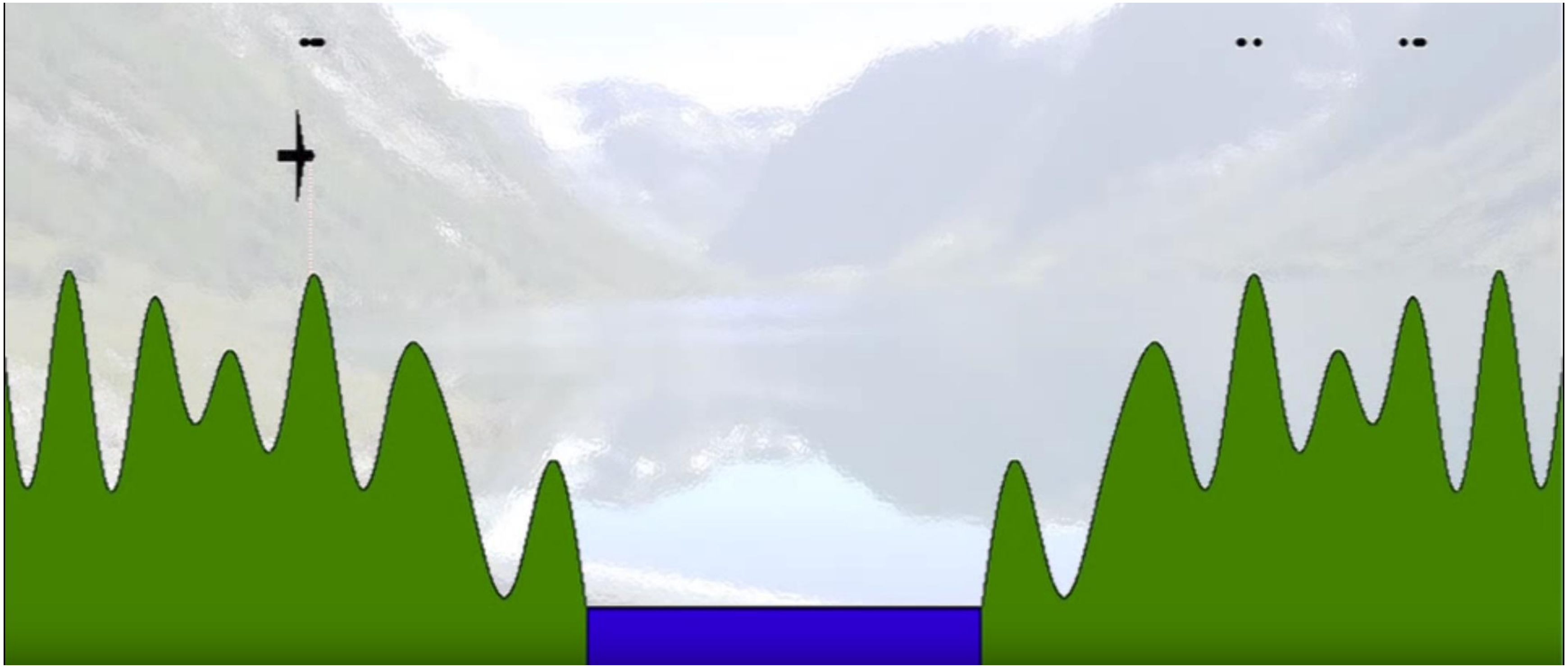
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Exhaust/Resample

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

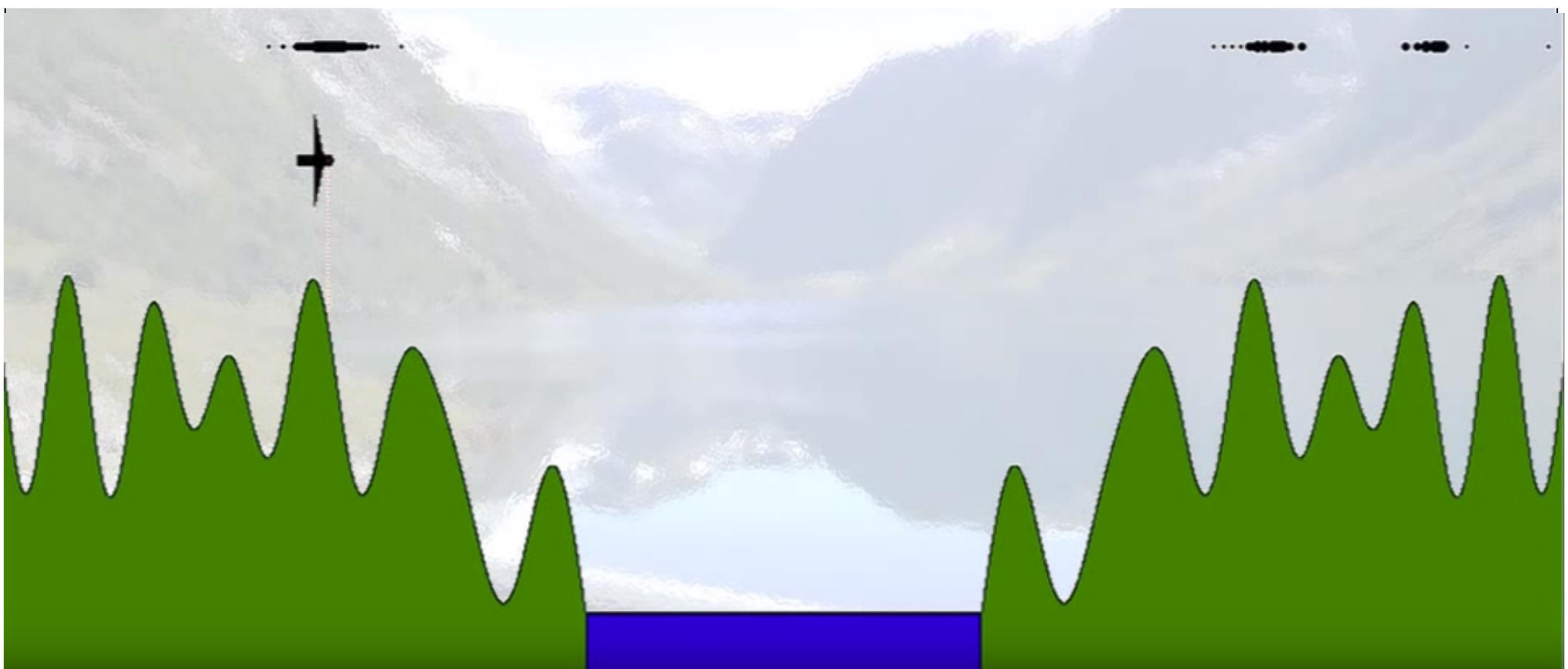
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



Particle Filter: Advance 1 Timestep

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

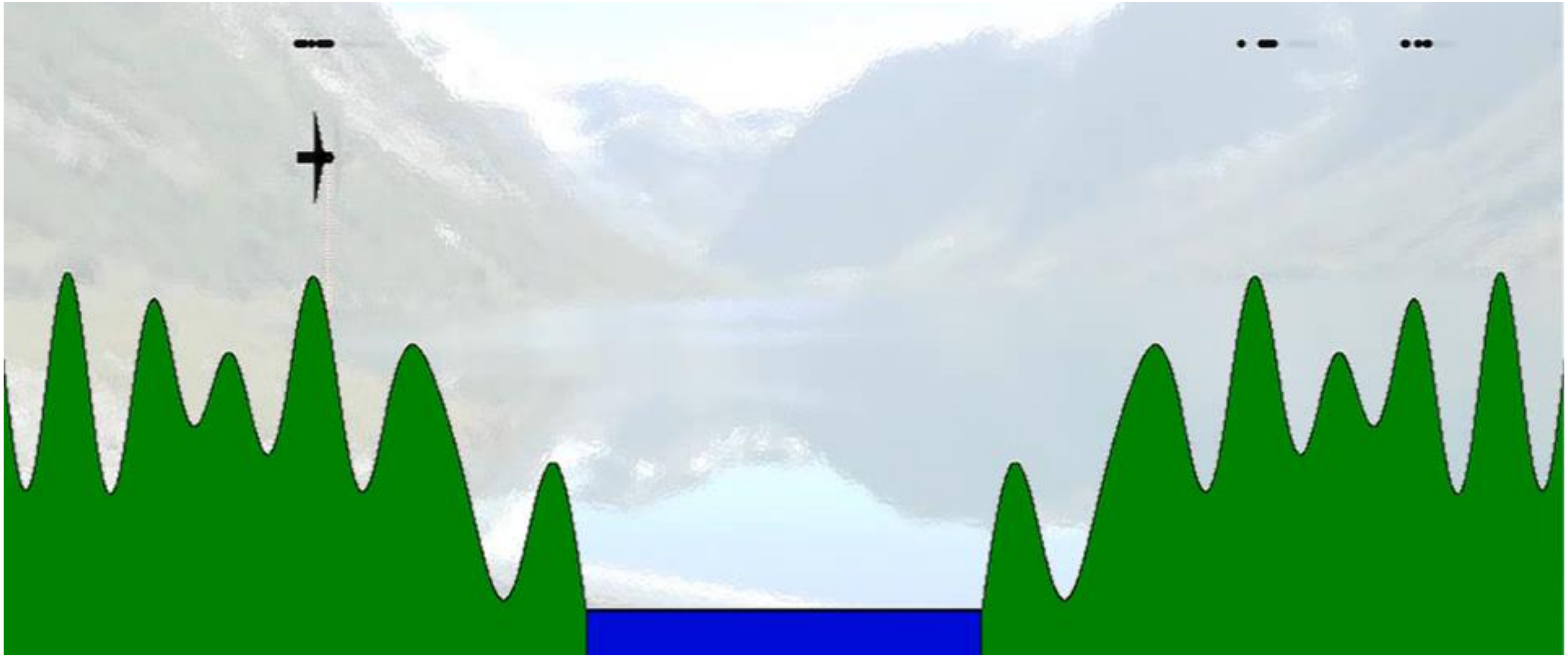
Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.



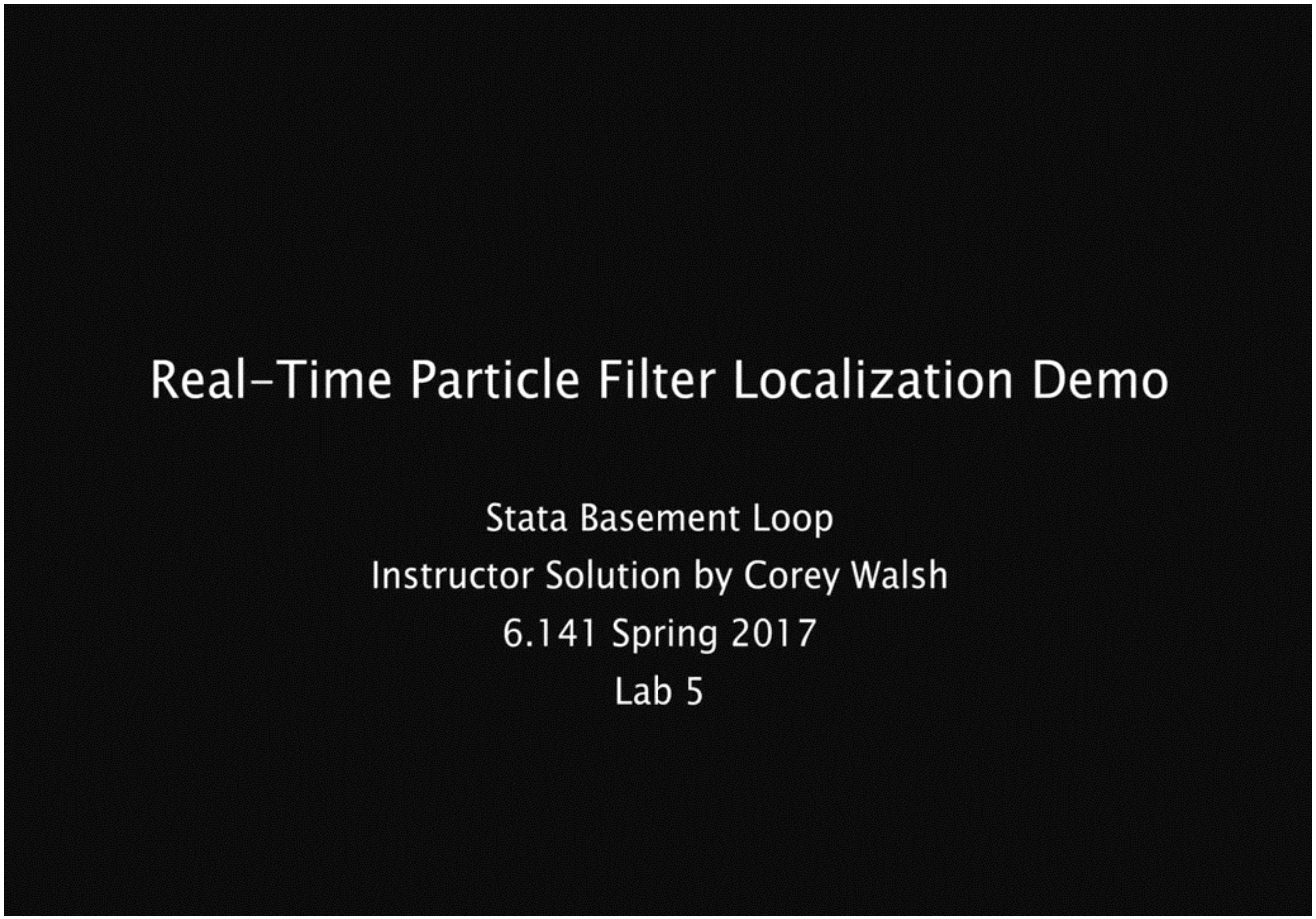
Particle Filter: Future Iterations…

- **Latent variable:** Horizontal aircraft position
- **Observable variable:** Vertical aircraft position (altitude)
- **Known information:** Airspeed, Map
- **Goal:** Use repeated observations to find true horizontal position

Main Idea:
Generate lots of hypotheses and let the observations determine their likelihood.

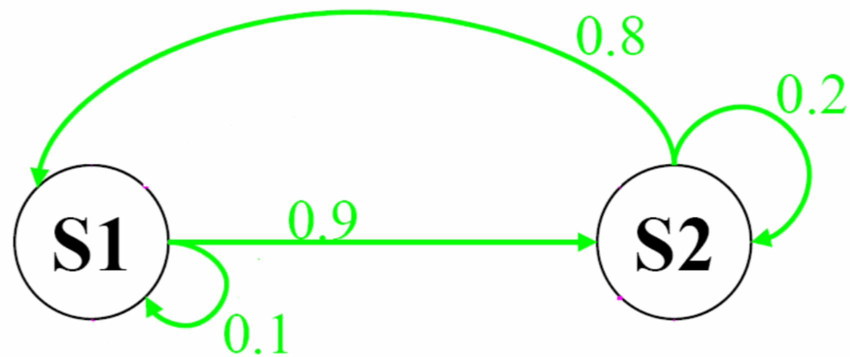


More Particle Filtering in Action!



Markov Chains

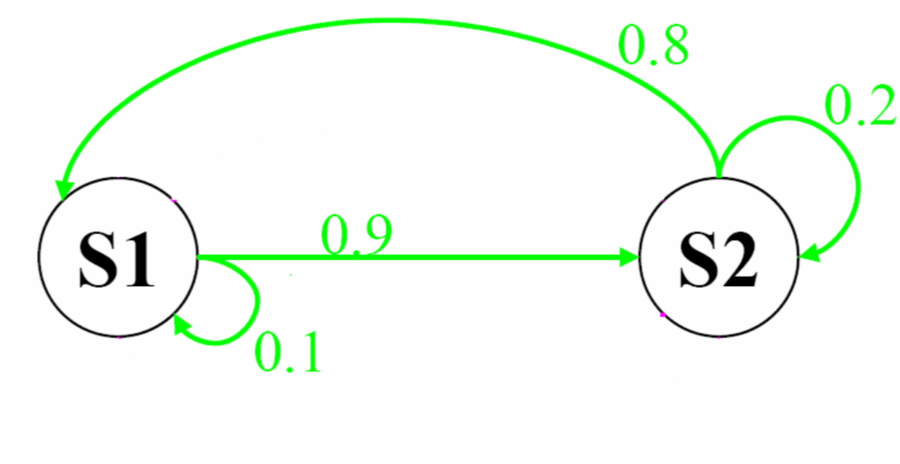
- Finite number of discrete states
- Probabilistic transitions between states
- Next state determined only by the current state
  - This is the Markov property



Rewards: S1 = 10, S2 = 0

Hidden Markov Model

- Finite number of discrete states
- Probabilistic transitions between states
- Next state determined only by the current state
- We're unsure which state we're in
  - The current states emits an observation

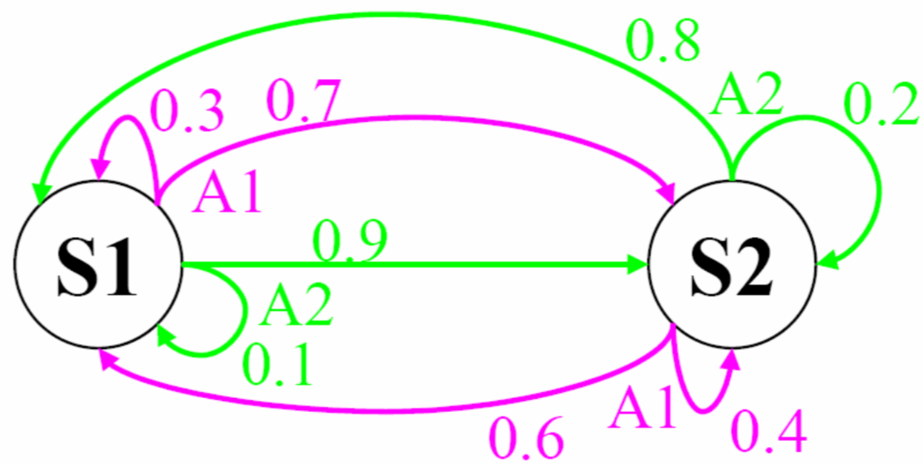


Rewards: S1 = 10, S2 = 0

Do not know state:
S1 emits O1 with prob 0.75
S2 emits O2 with prob 0.75

Markov Decision Process

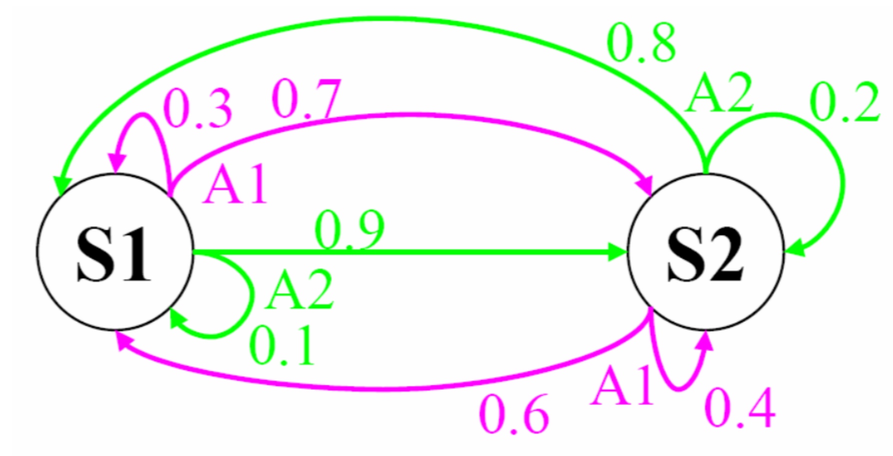
- Finite number of discrete states
- Probabilistic transitions between states ***and*** controllable actions in each state
- Next state determined only by the current state ***and*** current action
  - This is still the Markov property



Rewards: S1 = 10, S2 = 0

Partially Observable Markov Decision Process

- Finite number of discrete states
- Probabilistic transitions between states and controllable actions
- Next state determined only by the current state and current action
- We're unsure which state we're in
  - The current state emits observations



Rewards: S1 = 10, S2 = 0

Do not know state:
S1 emits O1 with prob 0.75
S2 emits O2 with prob 0.75

Markov Model Chart

Do we have control over the state transitions?
(Are we picking which actions are executed)

Are the states completely observable?

	NO	**YES**
YES	Markov Chain	MDP
NO	HMM	POMDP

Final Project Pitches

In two minutes or less:

- What new capability are you introducing / phenomenon are you exploring?
- What's hard about this now?
- How will you evaluate whether your approach works or not?