Semantic Localization

Matthew Deyo, Michael Traub, Anying Li,
Nicole Glabinski, David Stingley, Roxana Mata

Overview

1. **Motivation for Semantic Localization**
2. Particle Filters
3. Semantic Localization Implementation

Motivation

Orienteering Grand Challenge

What would you do?

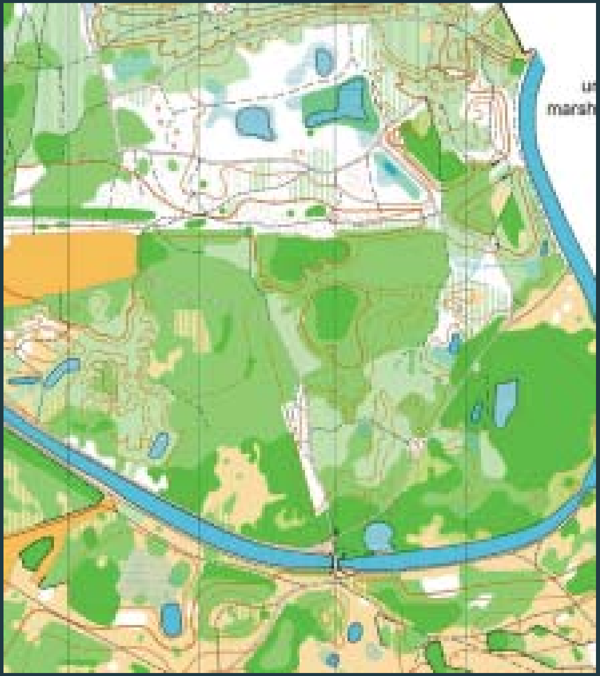
- You’re dropped in the wild
- You have a compass
- You have a map

Orienteering Relocation Tips

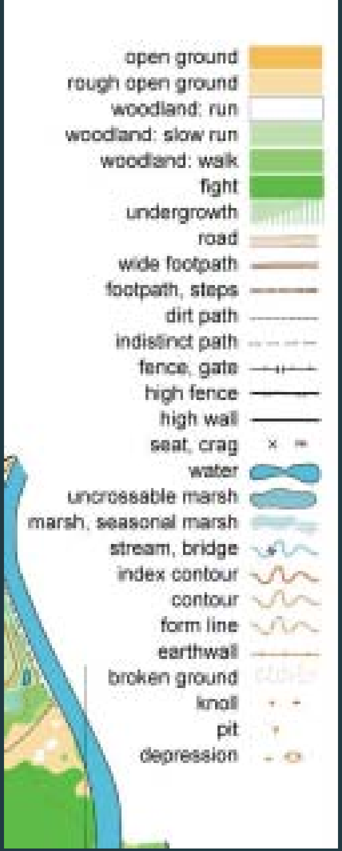
Tips from orienteering experts!:

- “Relocate: everyone gets disoriented from time to time.”
- “Stop, locate your last known location on the map, think about what you've seen and what direction you were moving, and how far you have gone.”
- “Look around you for any feature large or unique enough to be mapped.”

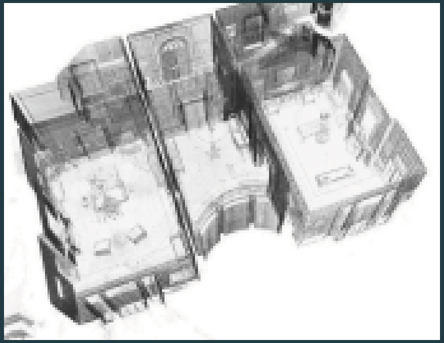
Orienteering Maps

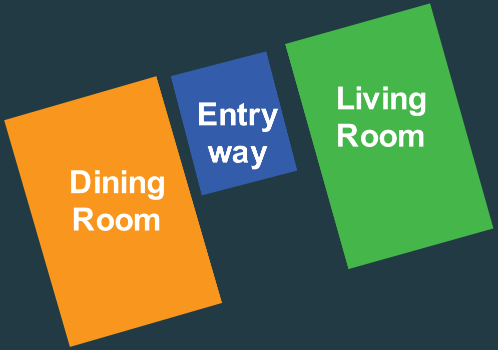


Rough Wood
Scale 1:5000 2.5m contours
Orienteering map
PEEL



What do we want in our map?





	Robot	Human
Encodes	distances, surfaces	rooms, objects, relationships
Memory	dense	sparse
Useful for	motion planning	activity planning

Semantic Information

“Signs and symbols that contain meaningful concepts for humans”

Semantic Information: Why is it important?

Human-robot interaction

Function-driven navigation and planning

Performance and memory optimization

Cheaper hardware

Semantic Localization

The problem of localizing based on semantic information

For the Grand Challenge, we have a map with labeled objects and their coordinates

How can we localize based on what objects we see?

Overview

1. Motivation for Semantic Localization

2. **Particle Filters**

3. Semantic Localization Implementation

Localization

Simple question: Where am I?

Not so simple answer

The answer depends on the map used

Metric Localization

If you want quantitative pose description:

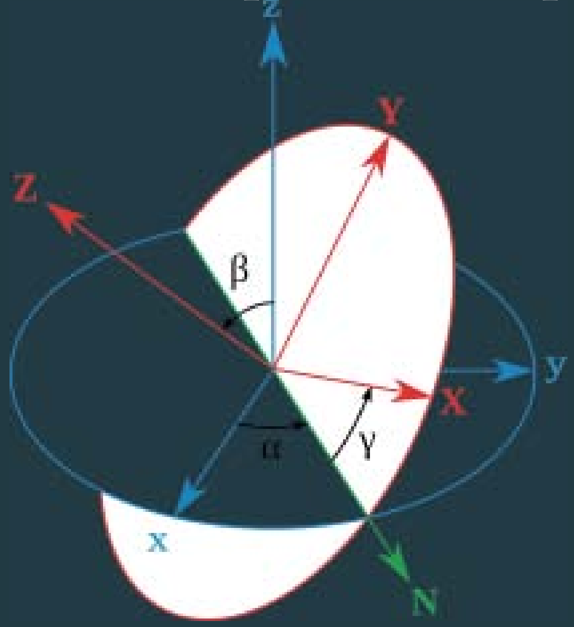
You need metric map for localization

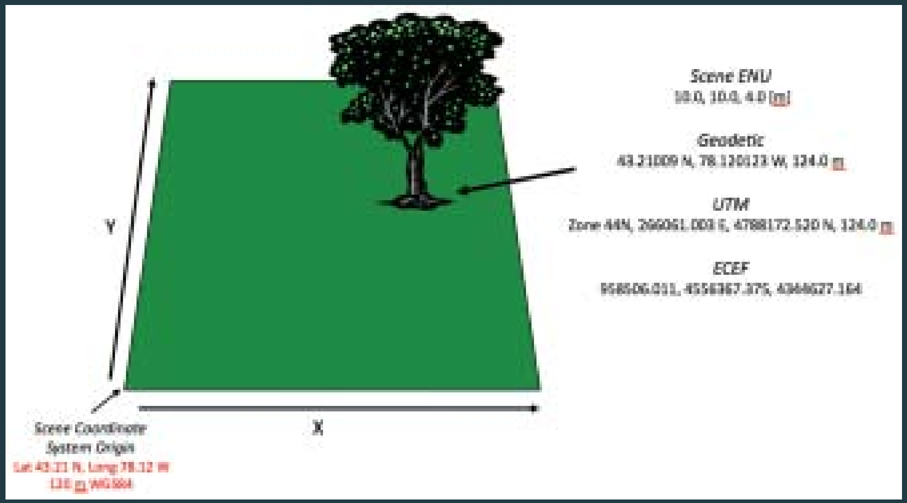
X, Y, Z coordinates in space

Angles for orientation

Metric Localization

Quantitative pose descriptions





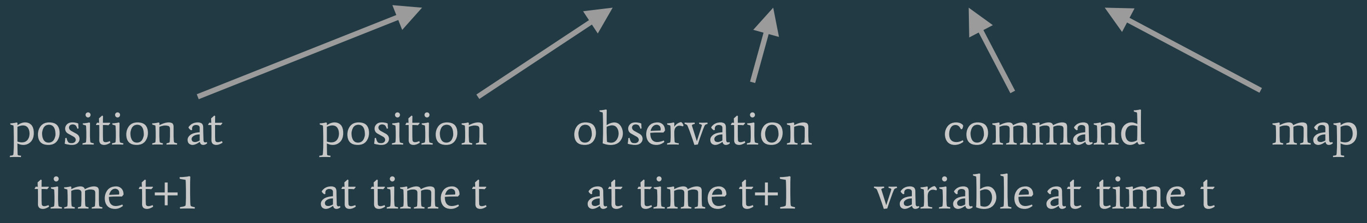
Review of Localization

- Localization problem statement

  Suppose that the control u_t is applied to the robot and, after moving, the robot obtains a random observation z_{t+1}. Given a prior belief over x_t and the map Y, what is the posterior belief of x_{t+1} after taking takes z_{t+1} and u_t into account?

- When we translate the localization question into probabilistic terms, we aim to find the distribution

$$p(x_{t+1} | x_t, z_{t+1}, u_t, Y)$$



position at time t+1

position at time t

observation at time t+1

command variable at time t

map

Review of Localization

- The Bayesian expansion of this posterior decomposes into

$$p(z_{t+1}|x_{t+1})\ p(x_{t+1}|x_t, u_t)\ p(x)$$

Observation noise model

Actuation model

Belief representation

- Our representation of the map limits what models we can use:
  - Topological map: actuation model to be transition probabilities
  - Laser scan observations: noise model over $\mathcal{R}^n$
  - Object detection observations: noise model over sets, or boolean variables
- Efficient semantic localization requires designing observation and actuation

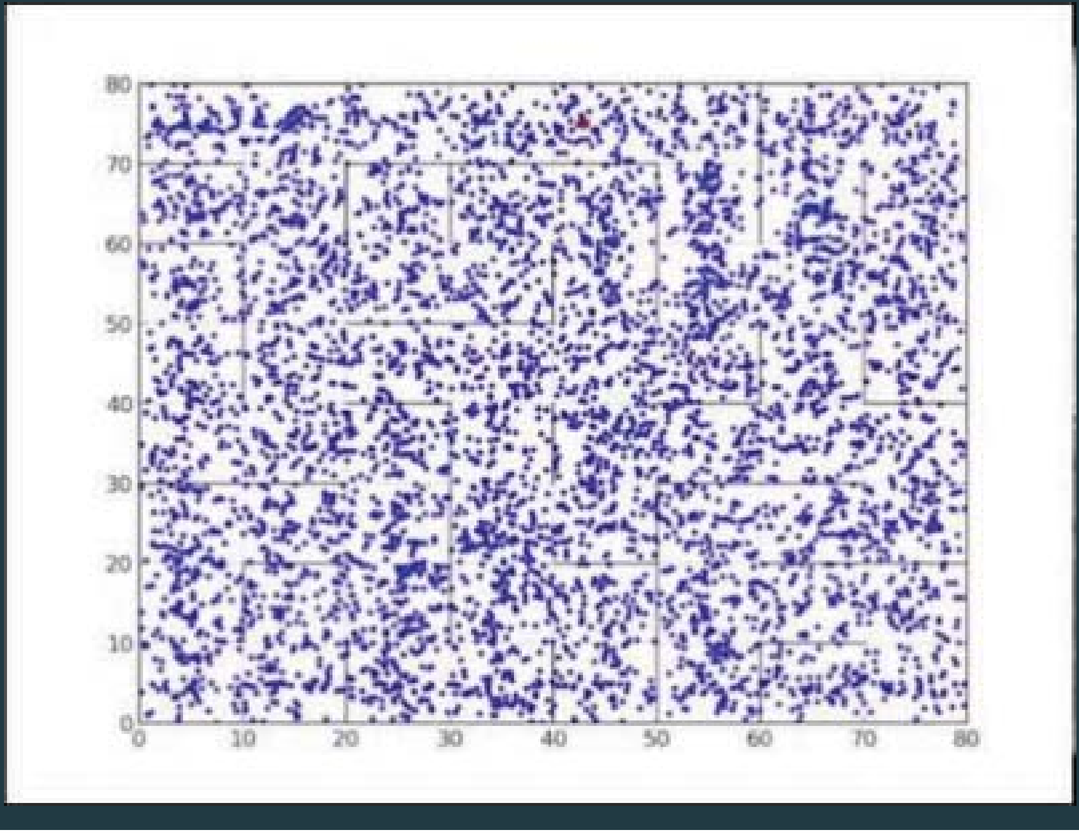
Particle Filters

- Representing our posterior over poses can be difficult

$$p(z_{t+1}|x_{t+1})\ p(x_{t+1}|x_t, u_t)\ \boxed{p(x)}$$

- Kalman filter → p(x) is a Gaussian
- Particle filter → p(x) is approximated by a set of points

Localization demo



Particle Filter

Sequential Importance Sampling Technique

Algorithm Steps:

0. Sample (using Initial Belief)

1. Update Weights

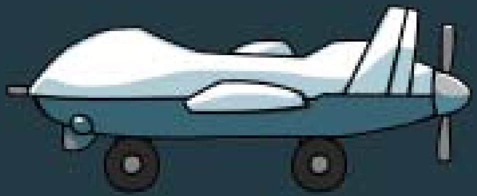
2. Resample

3. Propagate

Particle Filter - Example

Focus on problem with only one dimension

Aircraft

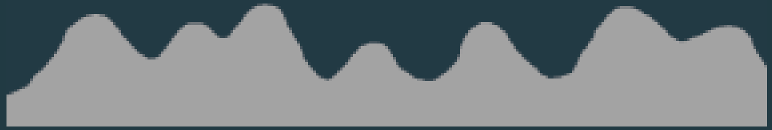


- Constant altitude
- Unknown x location
- Noisy forward velocity

Sensor

- Measures distance to ground below
- Noisy measurements

Map

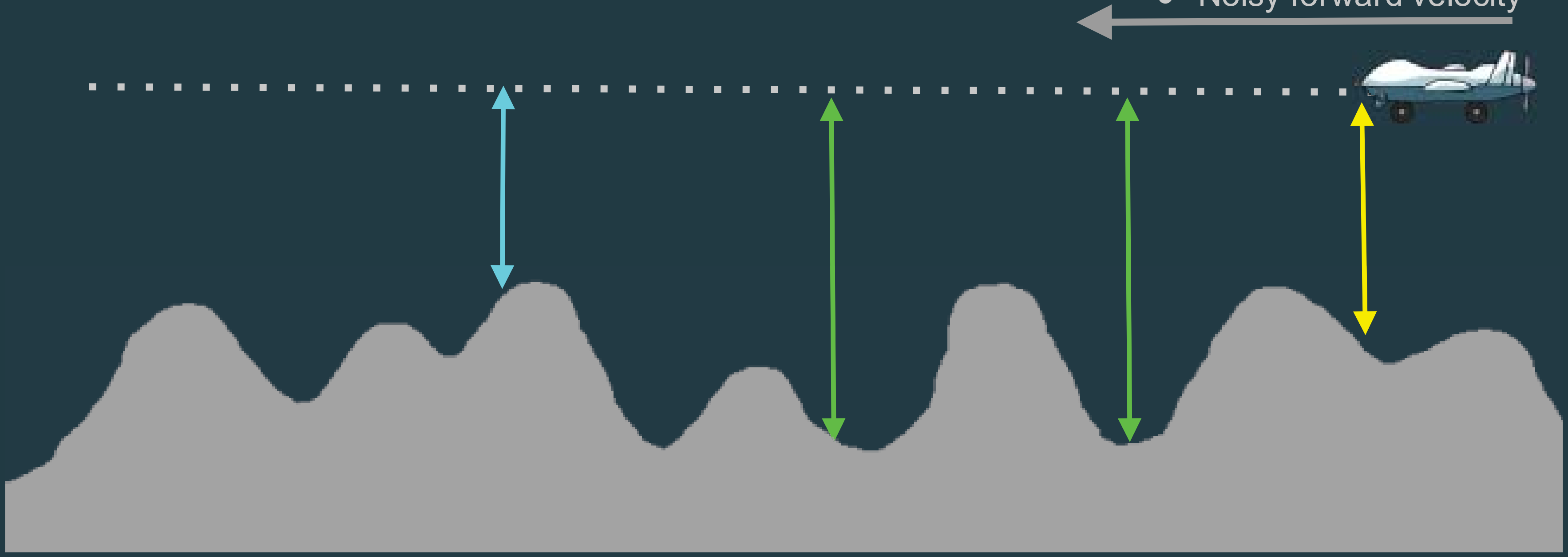


- Known mapping of x location to ground altitude

Goal: Determining unknown state - our location

Particle Filter - Example

- Constant altitude
- Unknown x location
- Noisy forward velocity



Particle Filter

Algorithm Steps:

0. Sample (using Initial Belief)

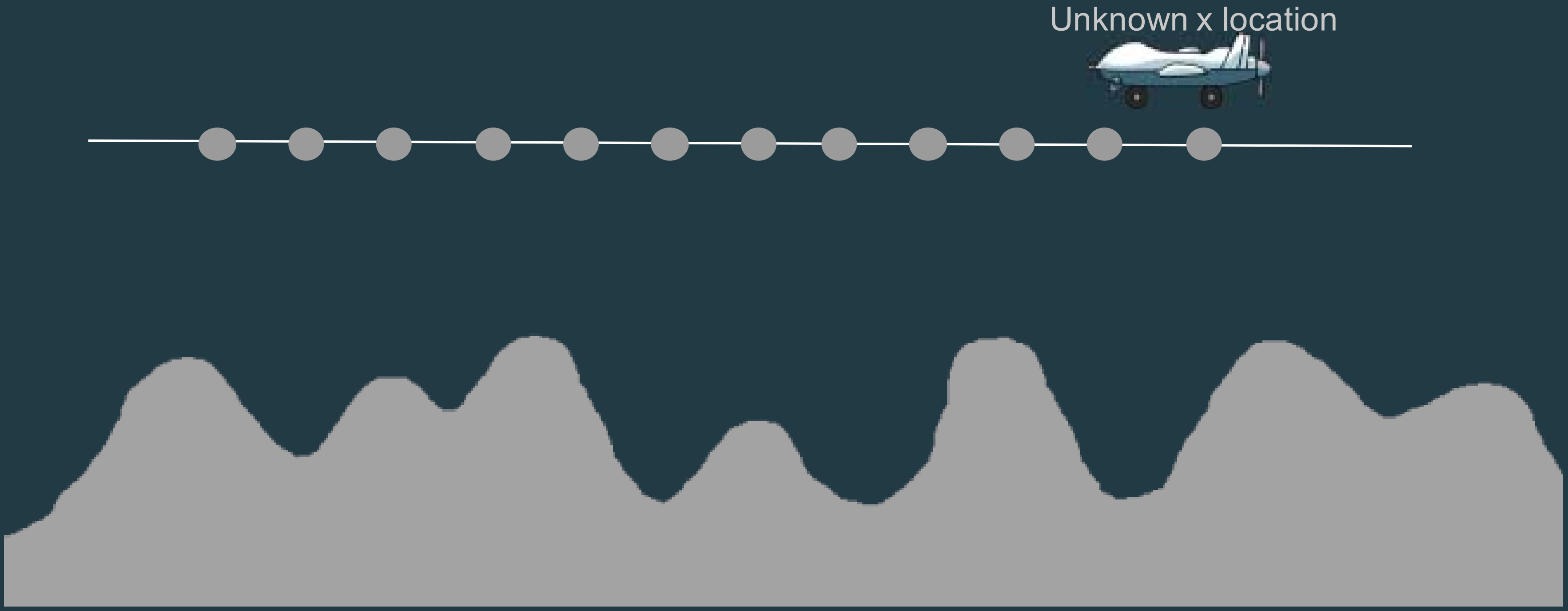
   If completely unknown initial state -> N samples from uniform distribution

1. Update Weights

2. Resample

3. Propagate

Initial Sampling with Unknown State



Particle Filter

Algorithm Steps:

0. Sample (using Initial Belief)

   If completely unknown initial state -> N samples from uniform distribution

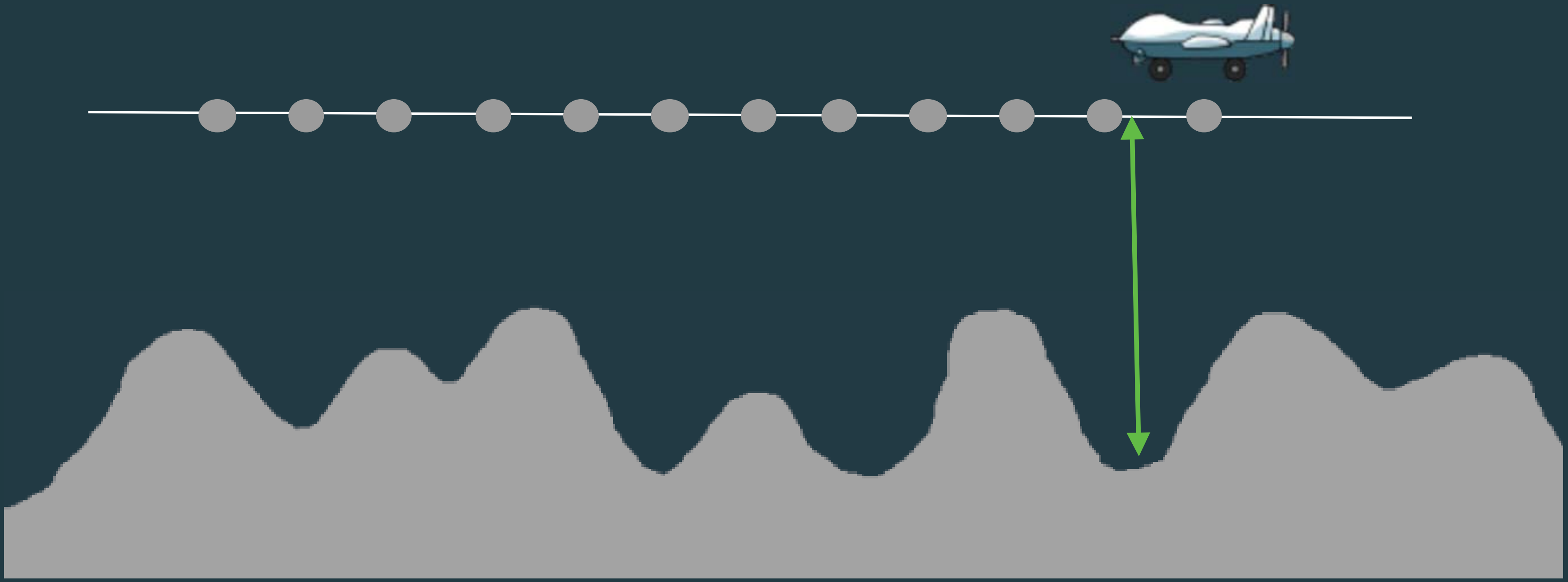
1. **Update Weights**

   Compare observations to expectations of each particle

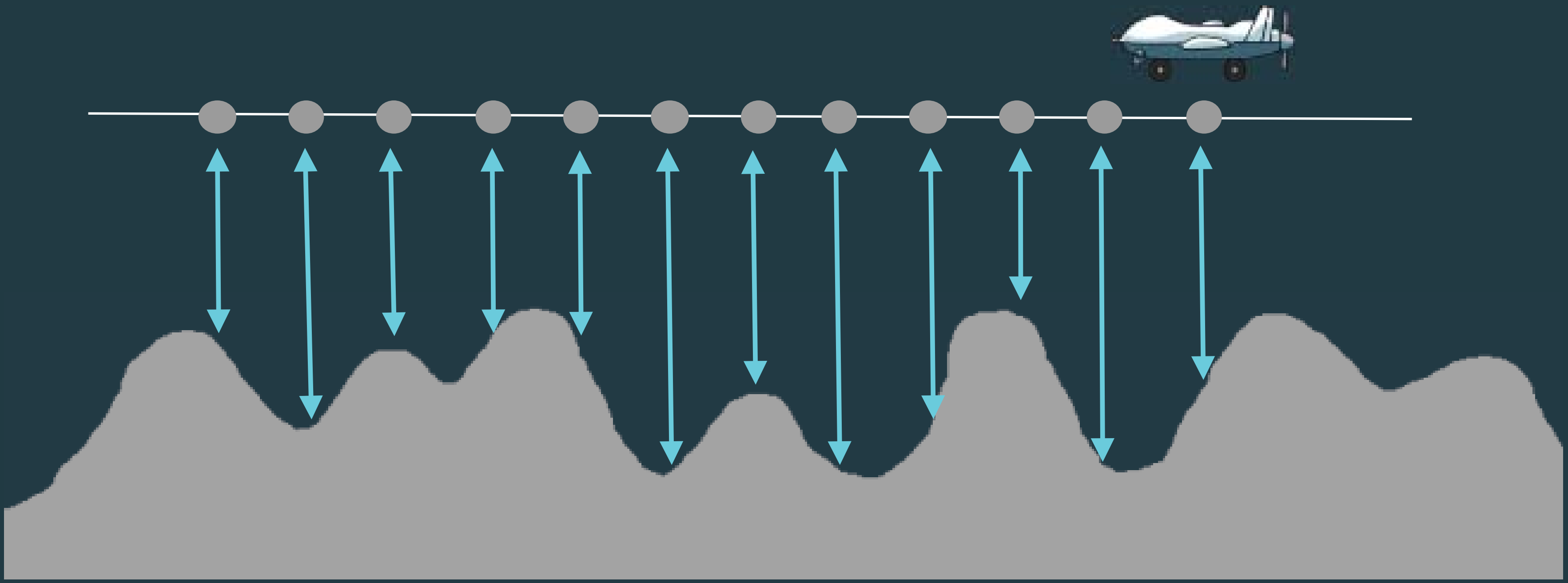
2. Resample

3. Propagate

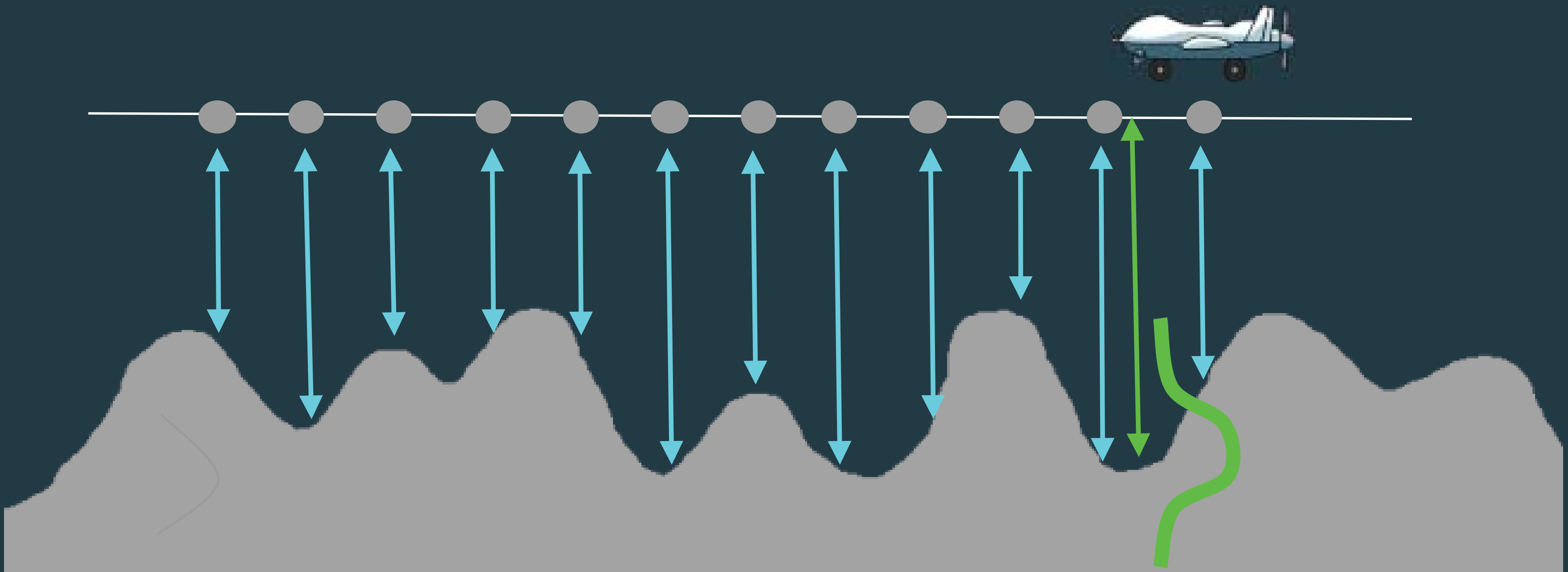
Measured value from our noisy sensor



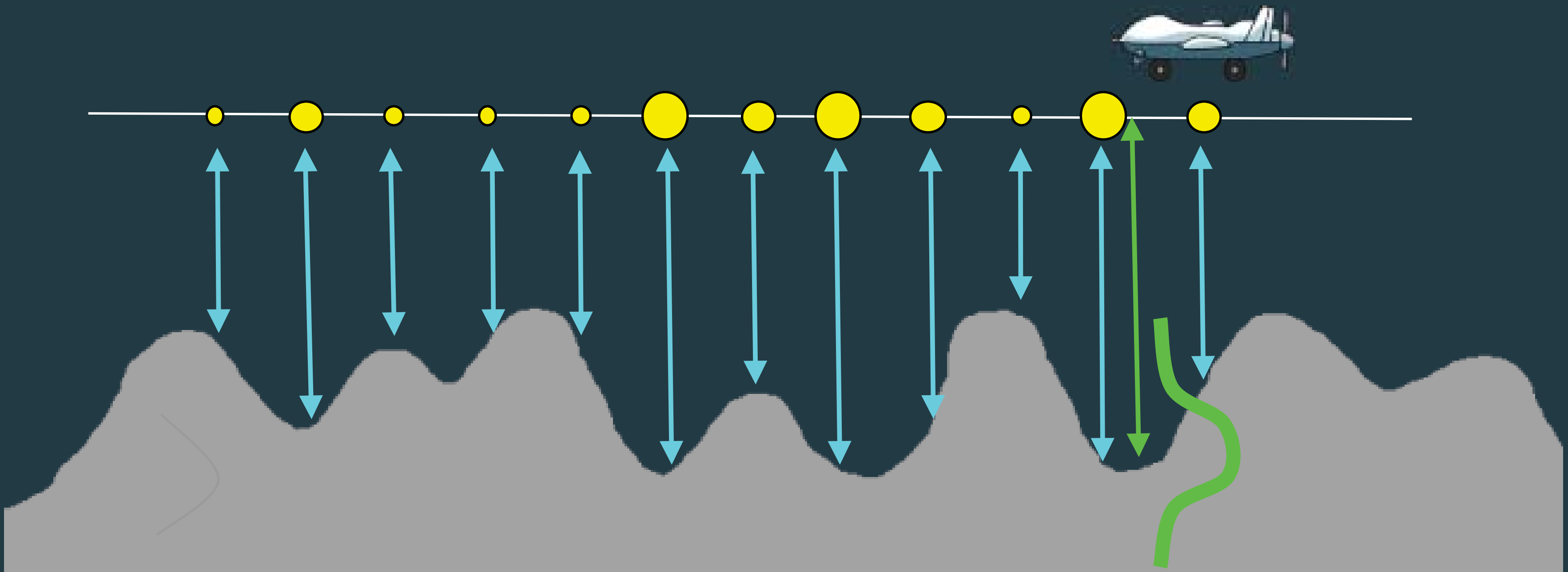
Expected height values of each particle



Likelihood that particle explains measurement



Particle weights based on likelihood



Particle Filter

Algorithm Steps:

0. Sample (using Initial Belief)

   If completely unknown initial state -> N samples from uniform distribution

1. Update Weights

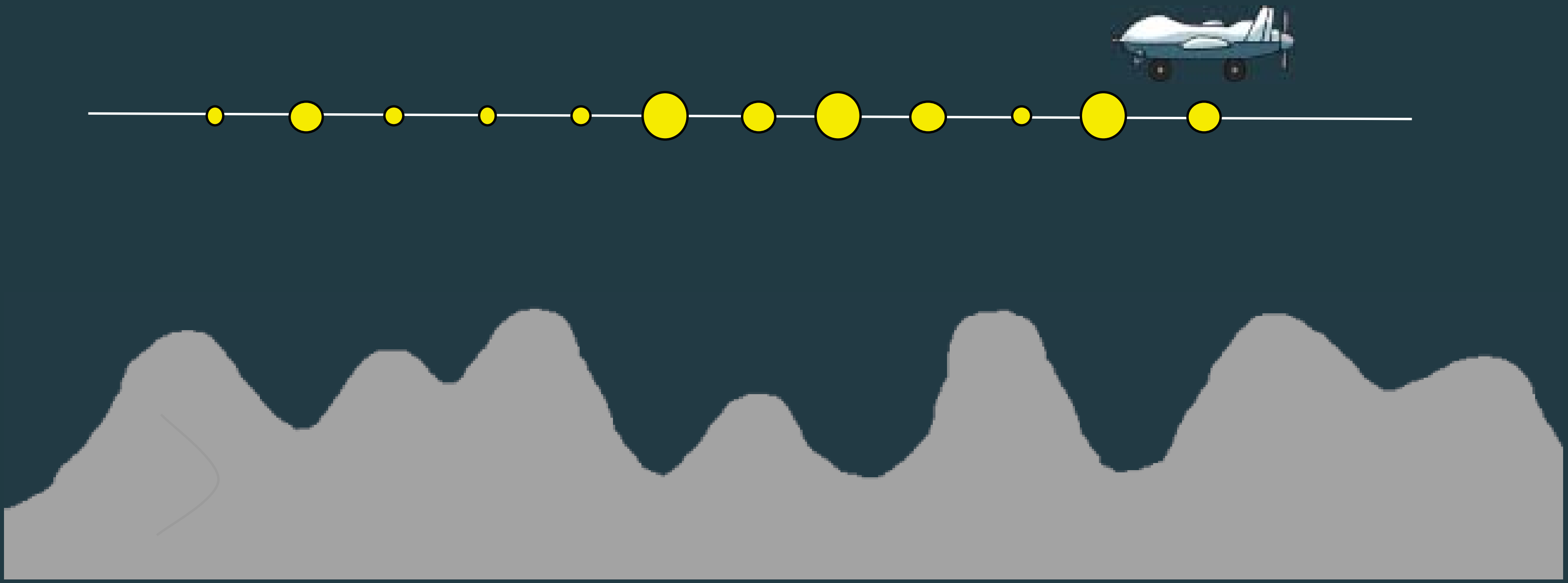
   Compare observations to expectations of each particle

2. **Resample**

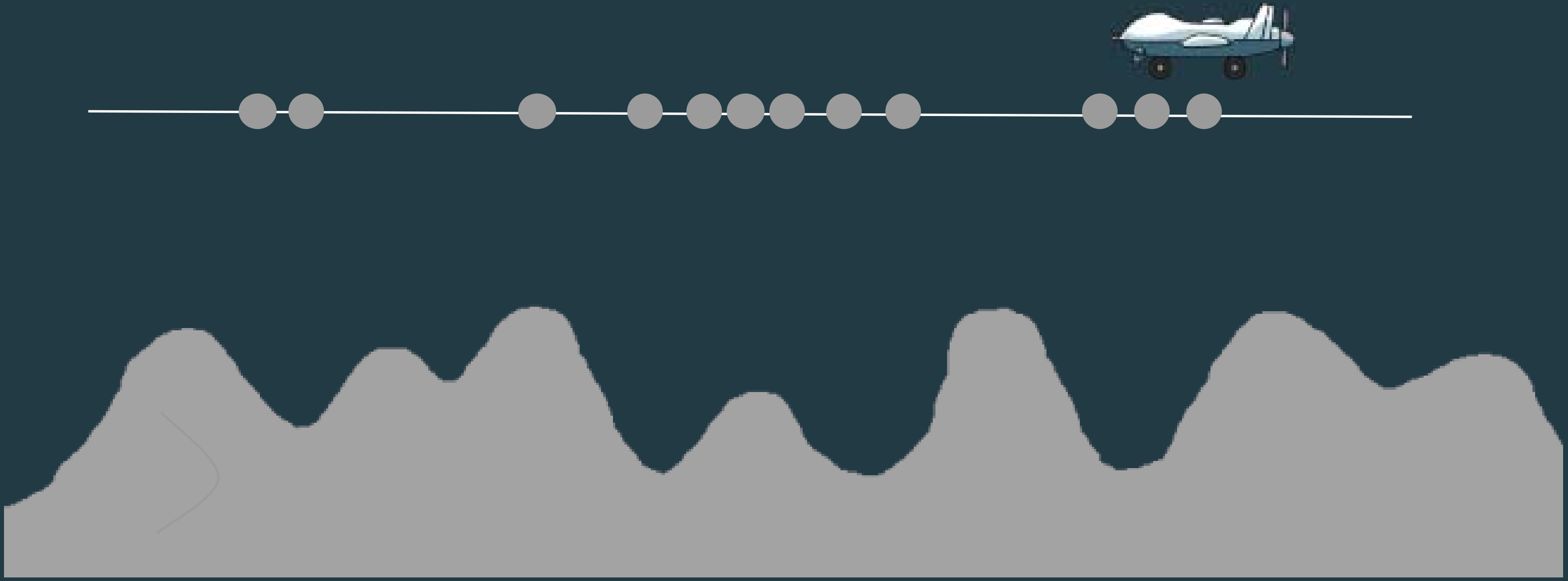
   Create N new samples based on weight distribution calculated

3. Propagate

Resample from measurement distribution



Resample from measurement distribution



Particle Filter

Algorithm Steps:

0. Sample (using Initial Belief)

   If completely unknown initial state -> N samples from uniform distribution

1. Update Weights

   Compare observations to expectations of each particle

2. Resample

   Create N new samples based on weight distribution calculated

3. Propagate

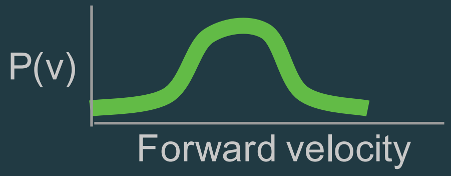
   Use dynamics model or inputs to propagate particles

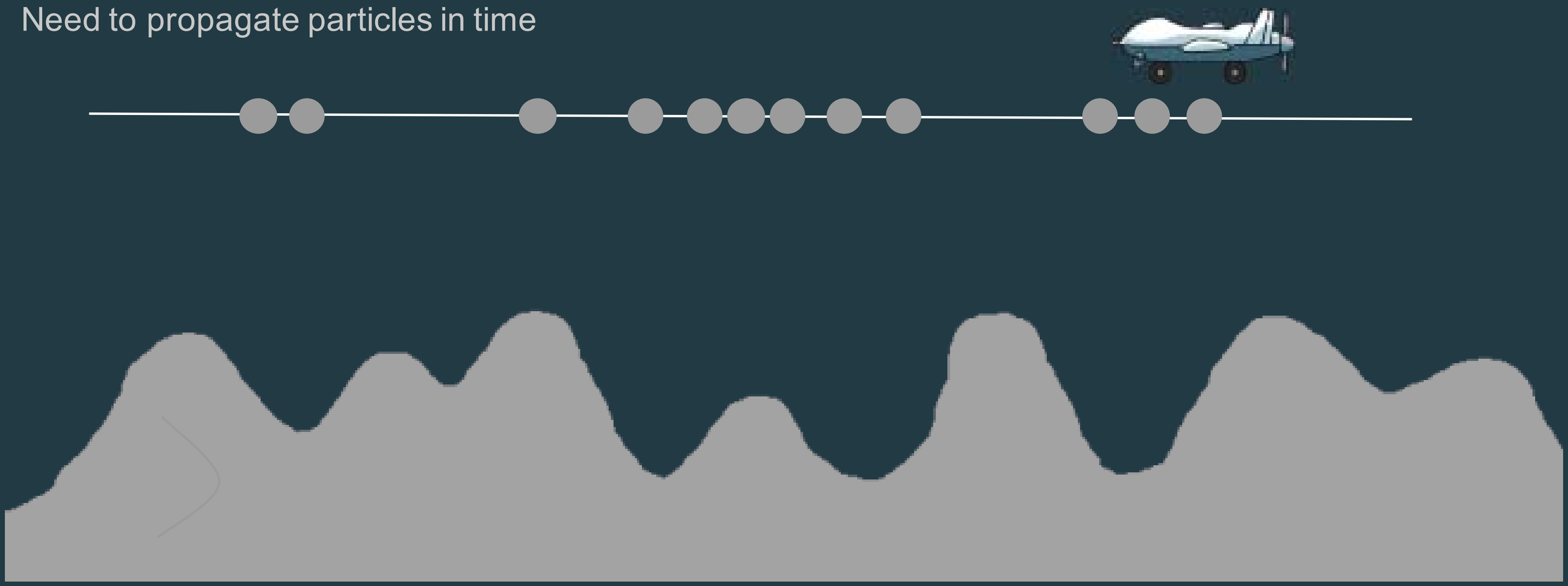
   Take into account uncertainty with new weight calculations

Dynamics Model

Delta t between sensor measurements

Need to propagate particles in time

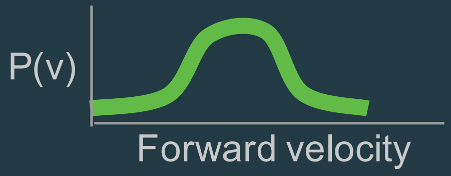


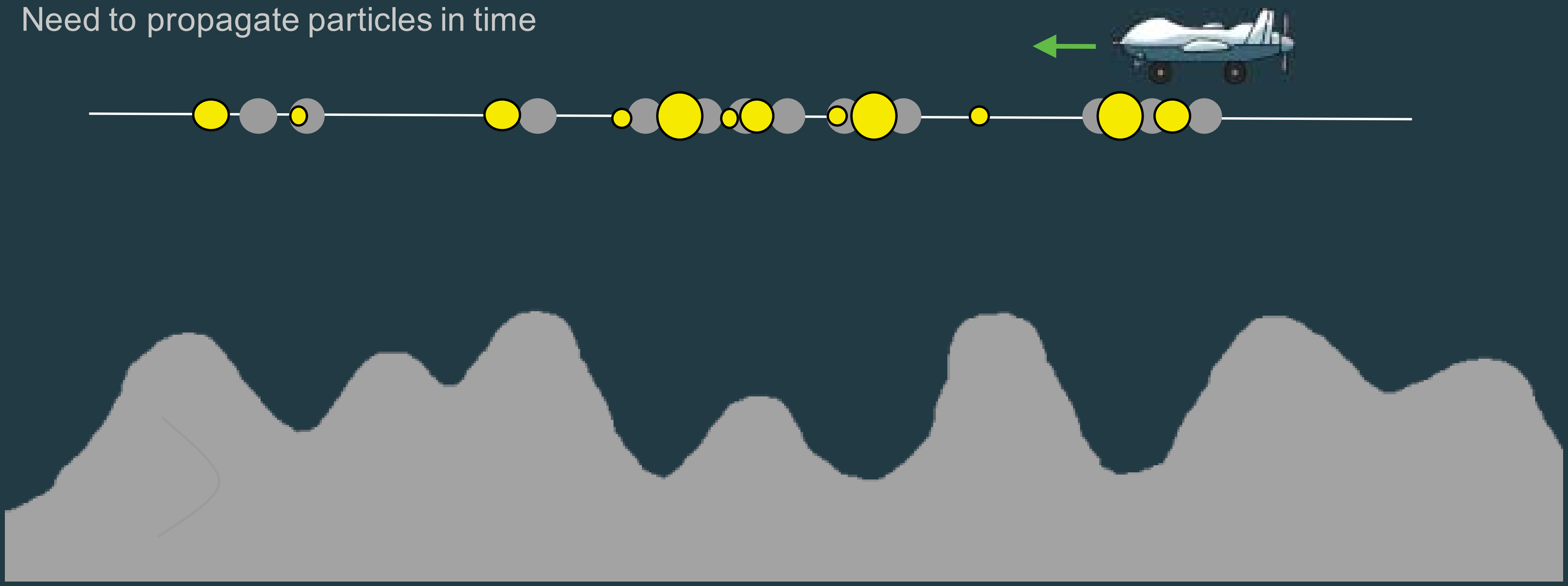


Dynamics Model

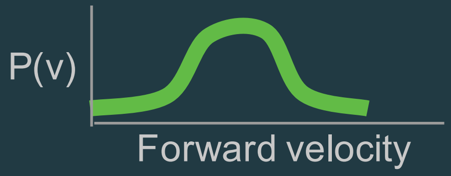
Delta t between sensor measurements

Need to propagate particles in time



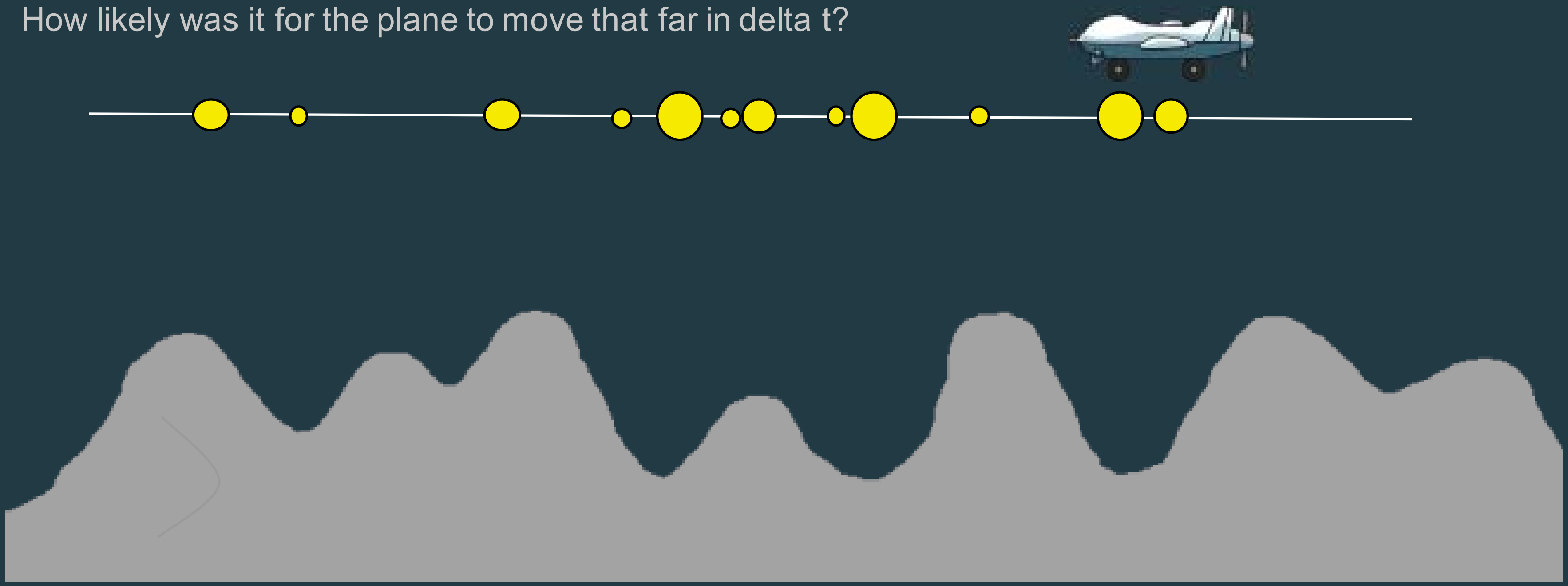


Dynamics Model



New weights based on probability of particle transition

How likely was it for the plane to move that far in delta t?



Particle Filter

Algorithm Steps:

0. Sample (using Initial Belief)

   If completely unknown initial state -> N samples from uniform distribution

1. Update Weights

   Compare observations to expectations of each particle

2. Resample

   Create N new samples based on weight distribution calculated

3. Propagate

   Use dynamics model or inputs to propagate particles

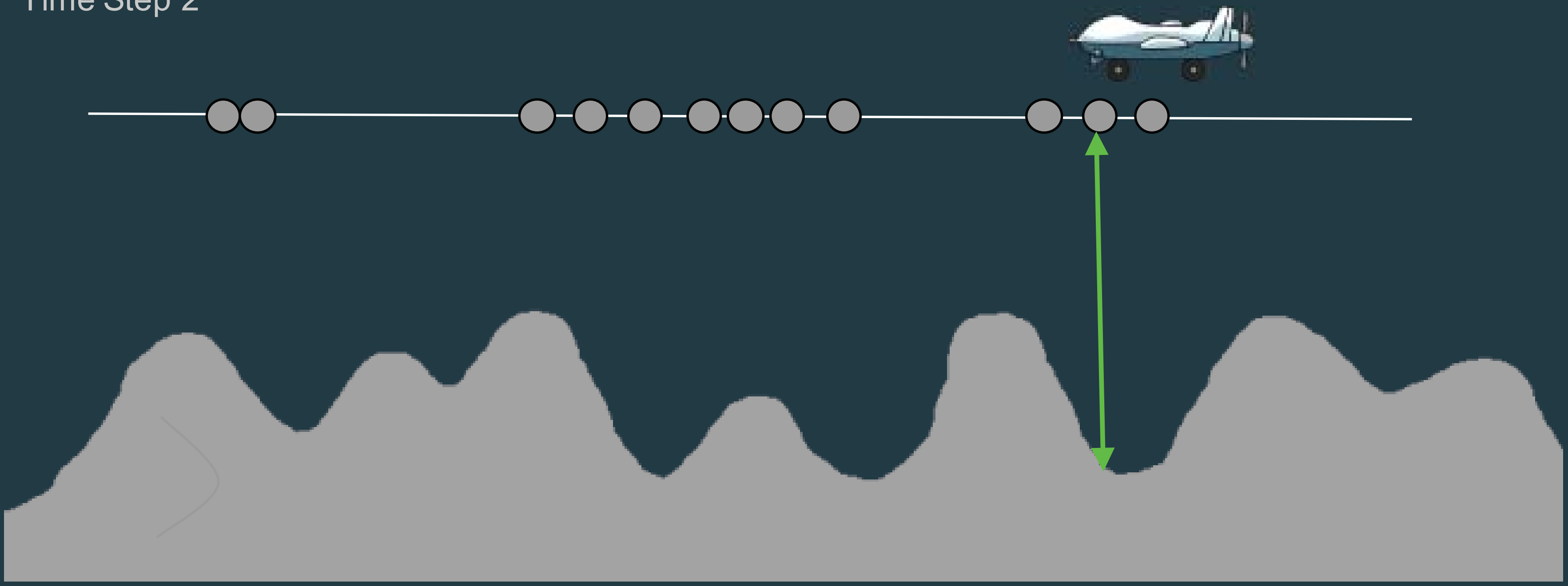
   Take into account uncertainty with new weight calculations

Repeat Steps 1 - 3

Keep filtering

Using new measurements and propagating through time

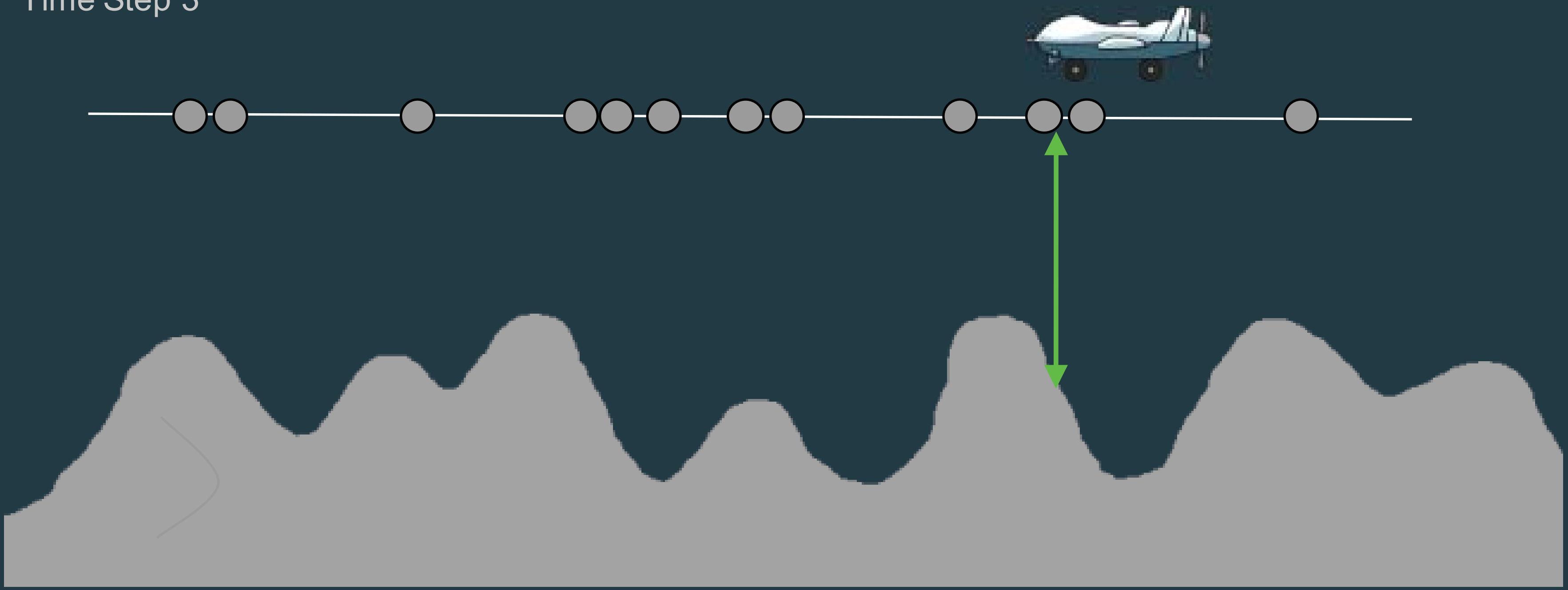
Time Step 2



Keep filtering

Using new measurements and propagating through time

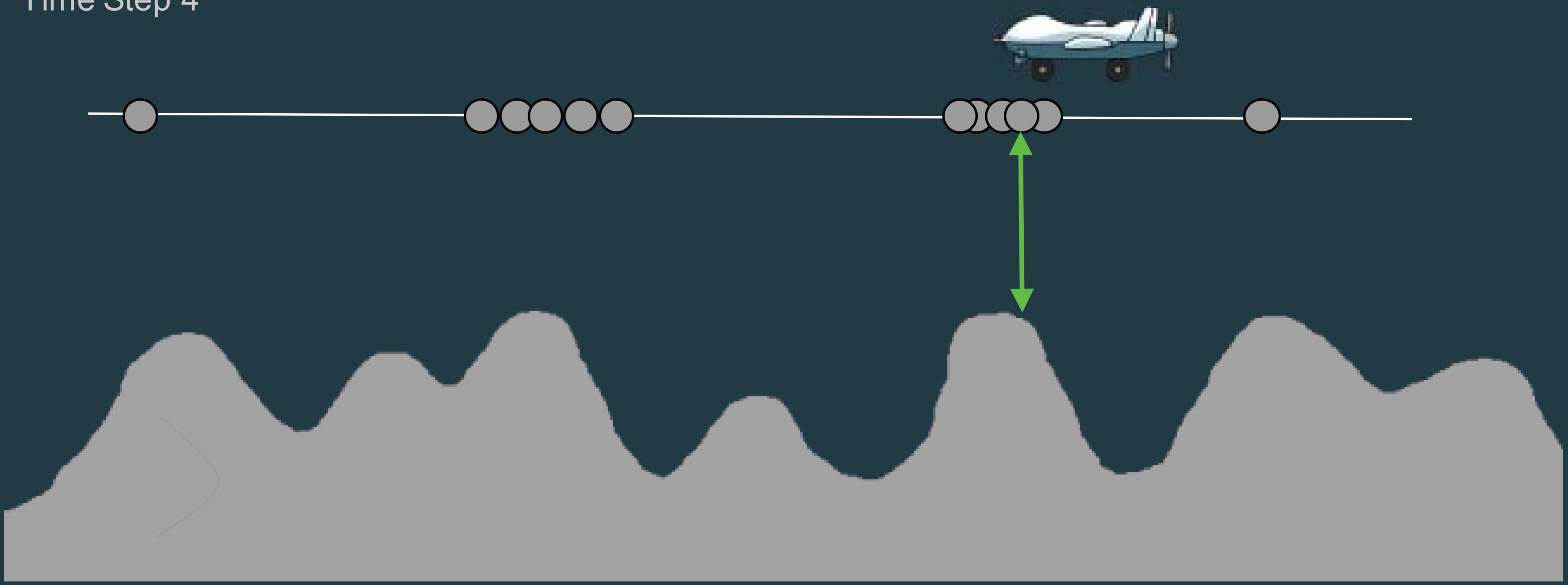
Time Step 3



Keep filtering

Using new measurements and propagating through time

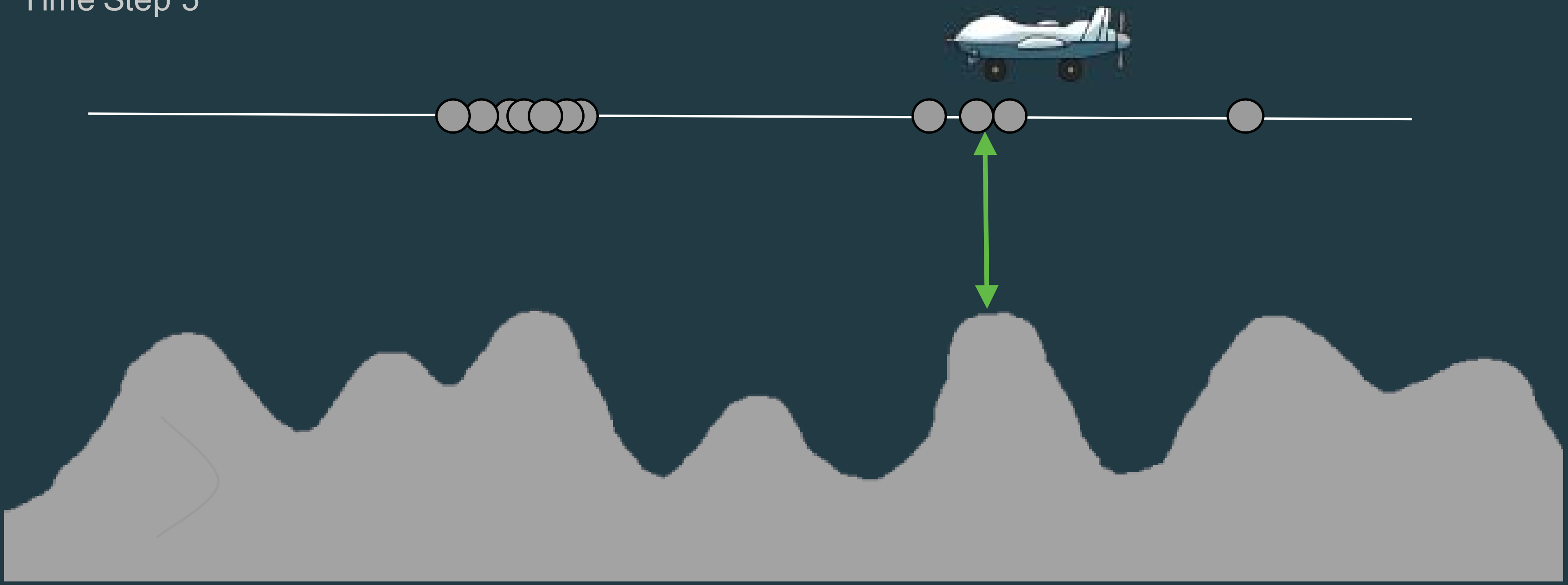
Time Step 4



Keep filtering

Using new measurements and propagating through time

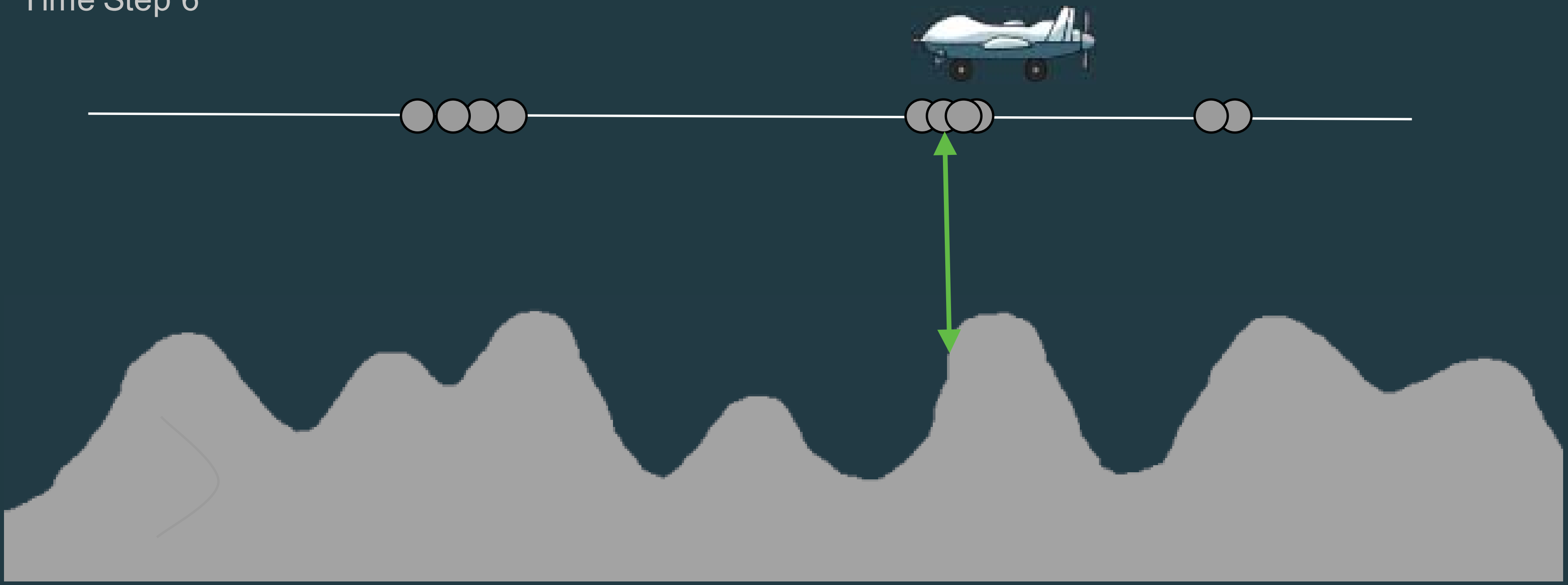
Time Step 5



Keep filtering

Using new measurements and propagating through time

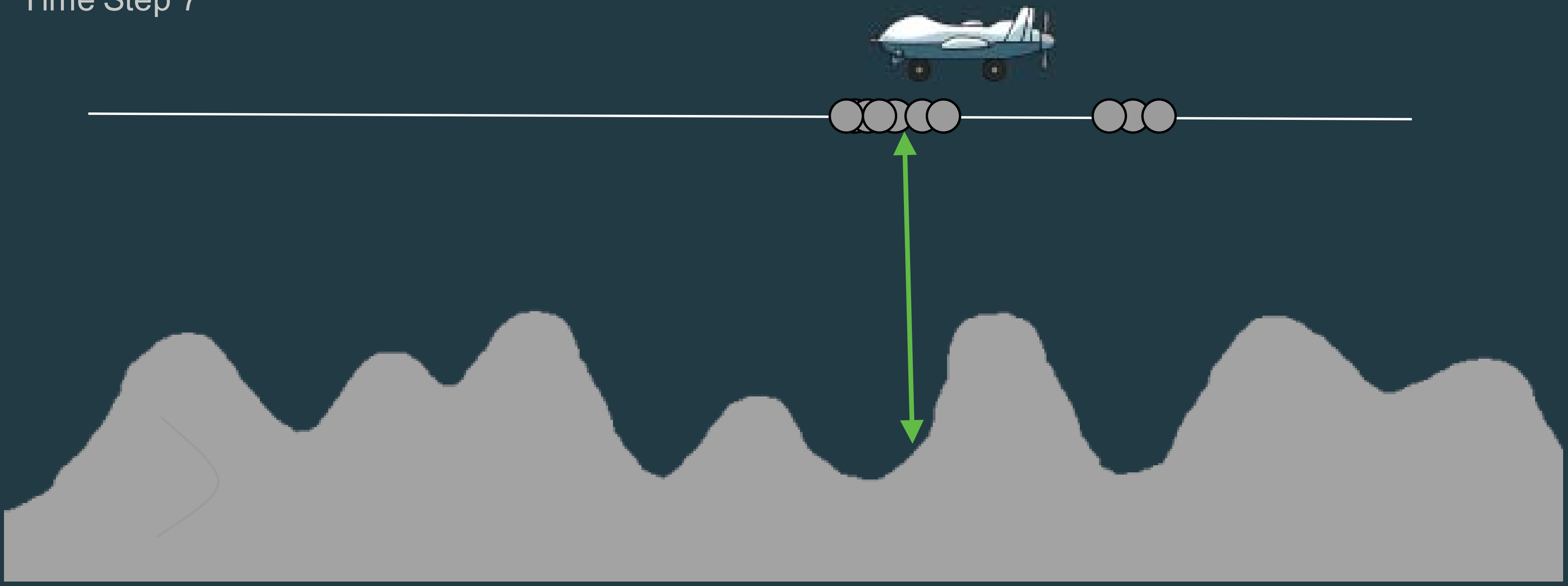
Time Step 6



Keep filtering

Using new measurements and propagating through time

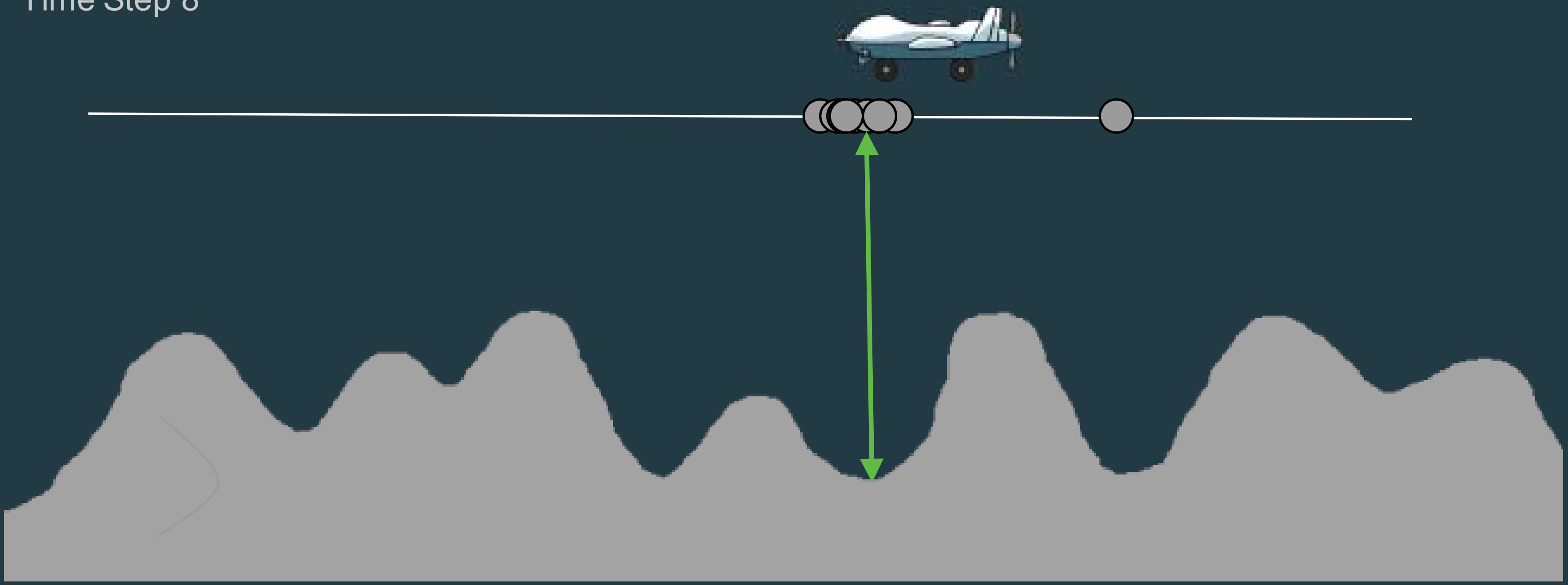
Time Step 7



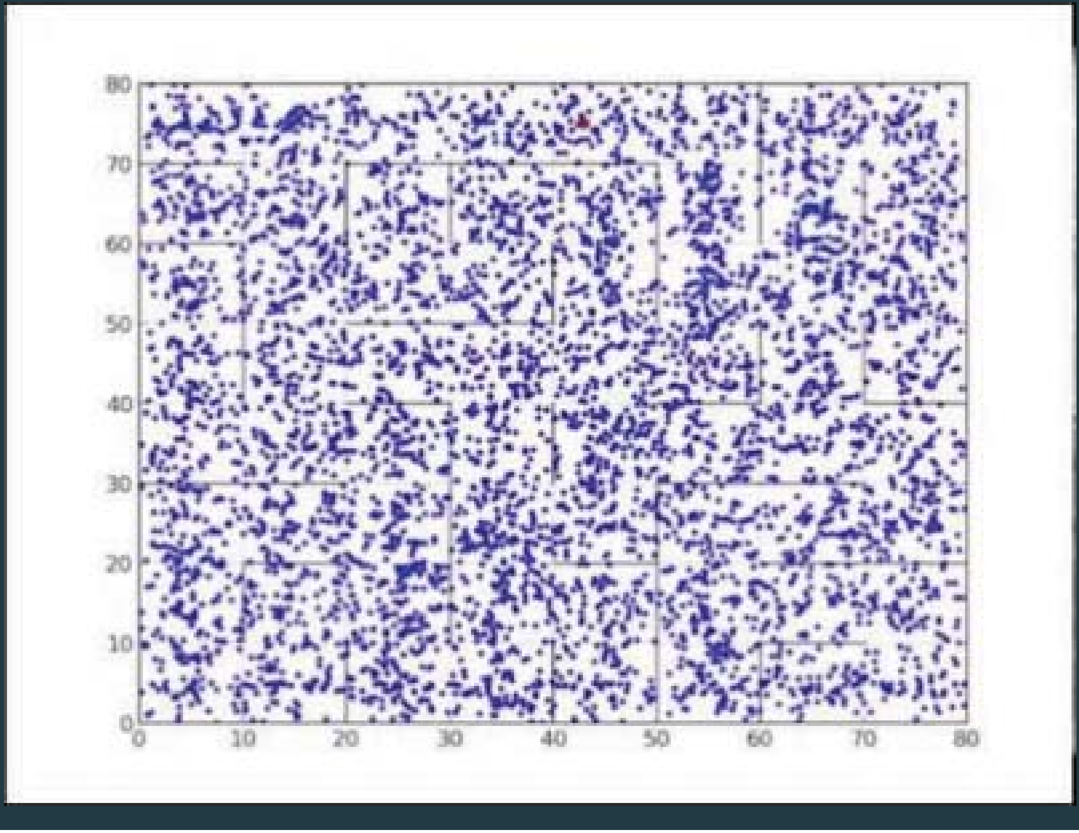
Keep filtering

Using new measurements and propagating through time

Time Step 8



Localization demo



Overview

1. Motivation for Semantic Localization
2. Particle Filters
3. **Semantic Localization Implementation**

Implementation

$$p(z_{t+1}|x_{t+1})\ p(x_{t+1}|x_t, u_t)\ p(x)$$

Observation
noise model

Actuation model

Belief
representation

Continuously Solve for most probable x

Thats our location

Psuedo Code

```
While the robot is moving
        Make observations                                           $z_{t+1}$
                Generate a probable location                        $P(x)$
                Update that location based on actuation             $P(x_{t+1} | x_t , u_t )$
                Simulate the observations at that location   }
                                                             }      $P( z_{t+1} | x_{t+1} )$
                Compare expected and actual                  }
                Update our location estimates based on comparison
```

Observation model selection

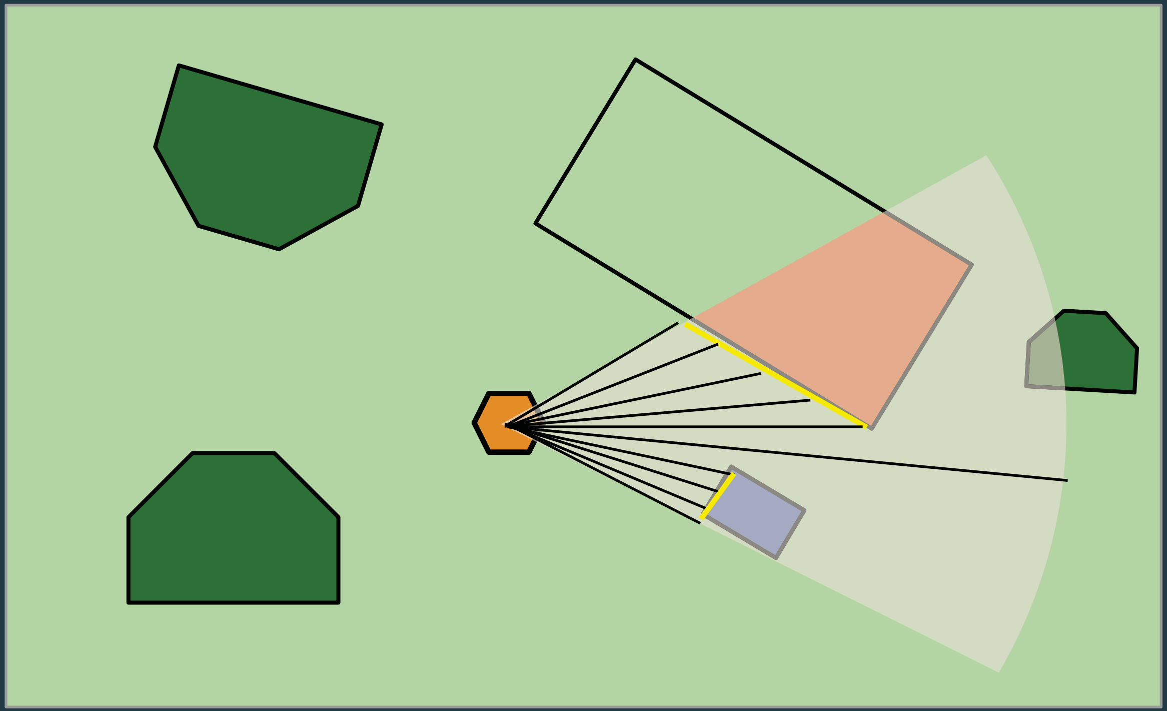
We needt o define **z** (our observation)

AL abeled Laser Scan

AS cene with Objects at Locations

AS et ofO bjects

Field-of-view with laser scanner

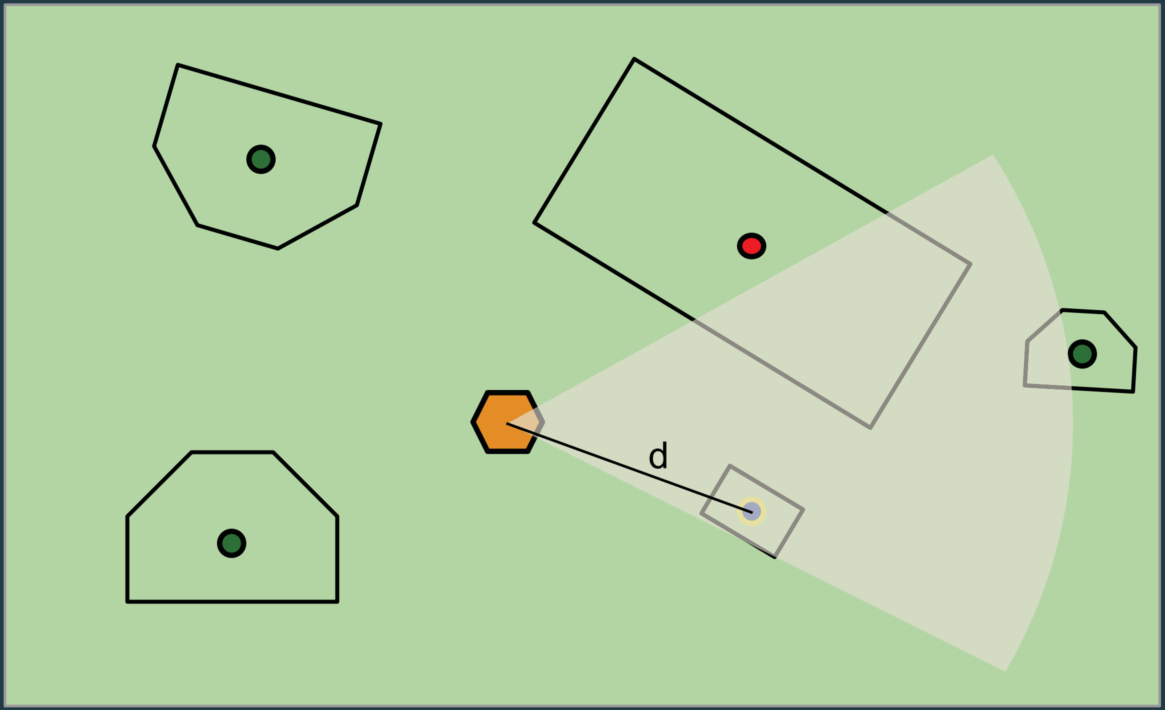


Legend

- mailbox
- tree
- house

Check each line segment for intersection at each θ. What counts as a detection?

Object-Point Assumption



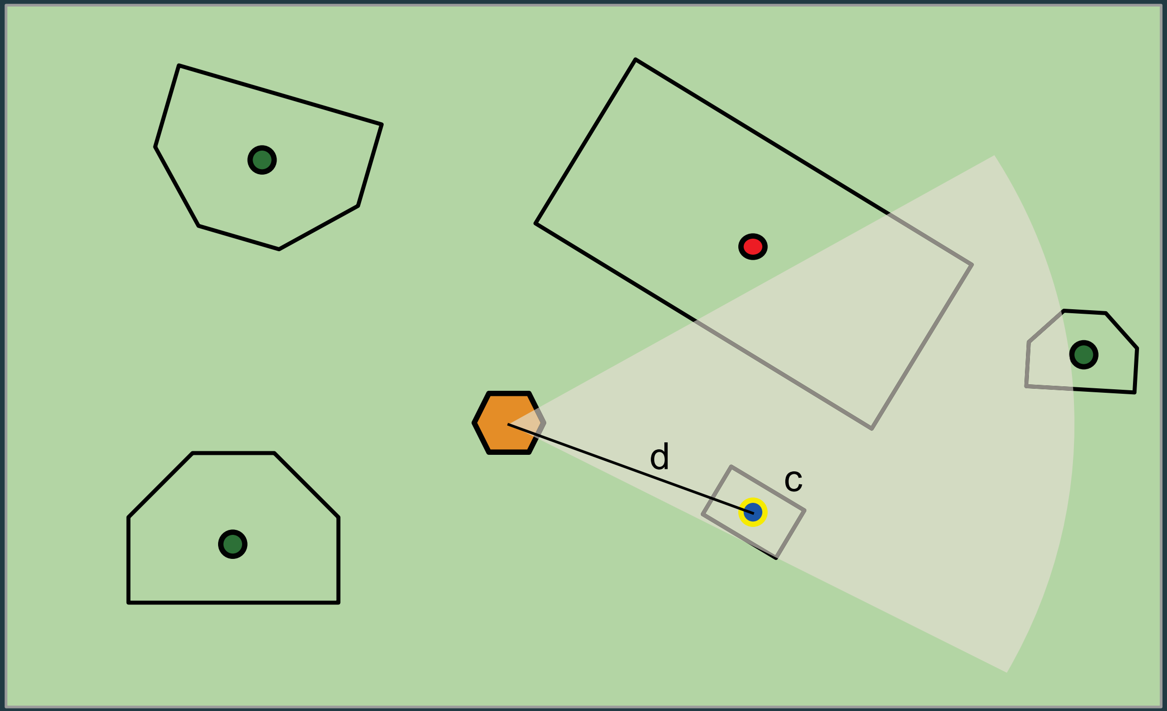
Legend

🟦 mailbox

🟩 tree

🟥 house

Field-of-view with point objects



Legend

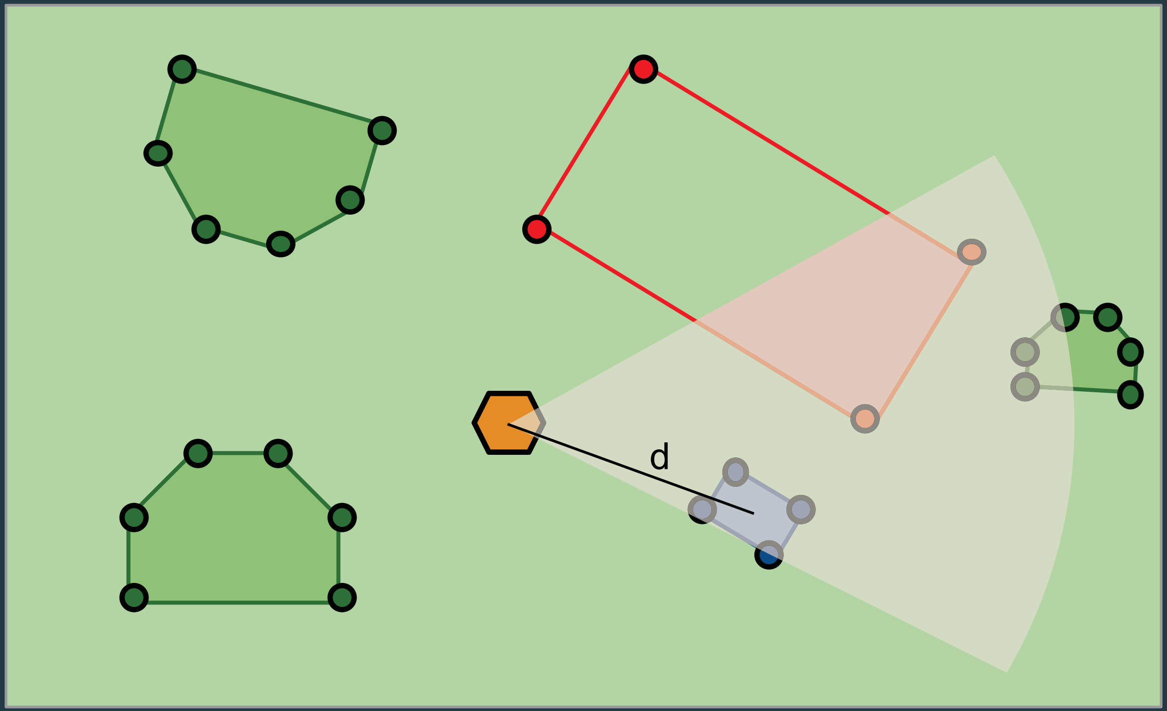
mailbox

tree

house

Check each point for intersection with FOV

Field-of-view with polygon objects



Legend

mailbox

tree

house

New observation type means new error types

Depending on what we characterize the observation as, there are different opportunities to get it wrong

Observation	Potential Errors
Distance & Bearing	Noise, Sensor Limitations
Object Class	Classification Error
Sets of Objects	Equality under Permutations

$$P(\, z_{t+1} \mid x_{t+1} \,) \Rightarrow P(\, Z \mid Y(x), x \,)$$

Z = Set of Observed Objects

$\{$ House, Mailbox $\}$

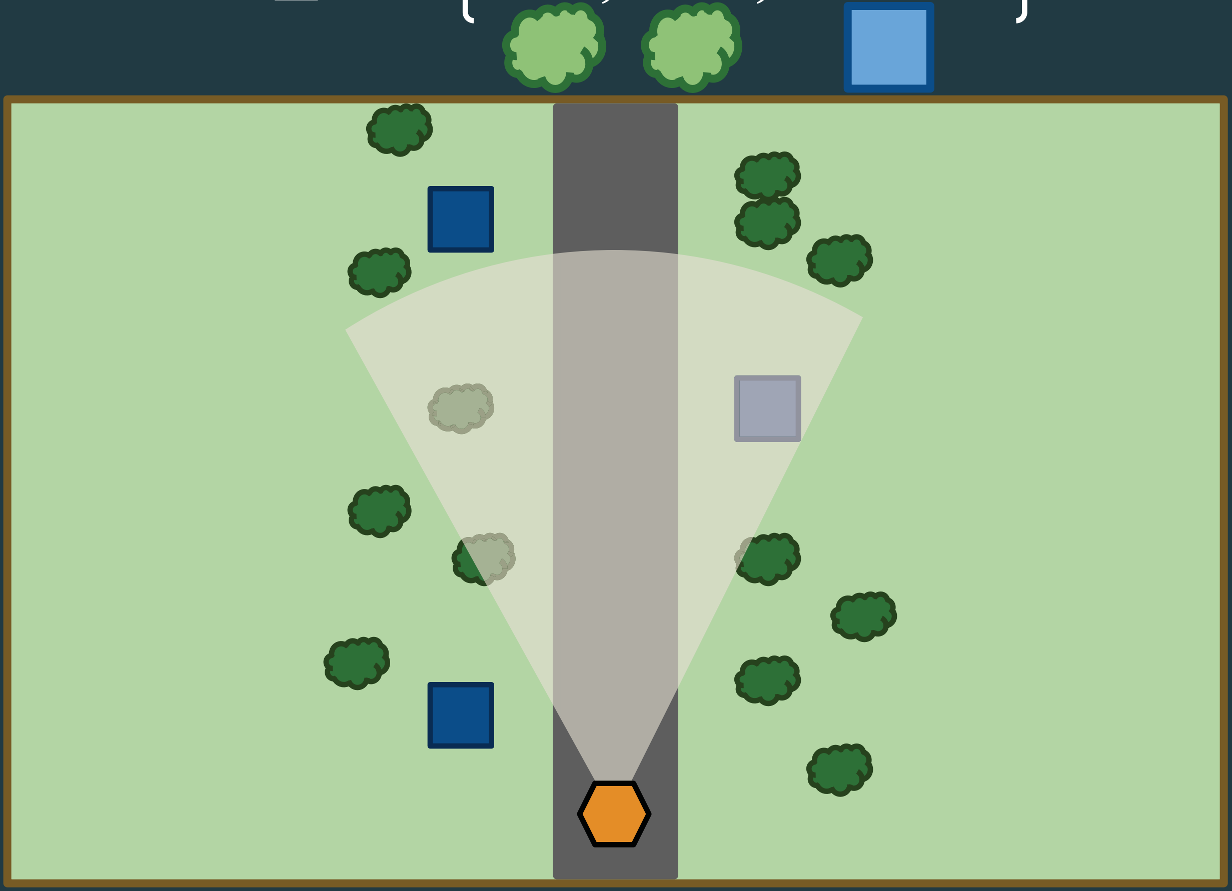
$Y(x)$ = Set of Expected Objects for a given position

X = Position

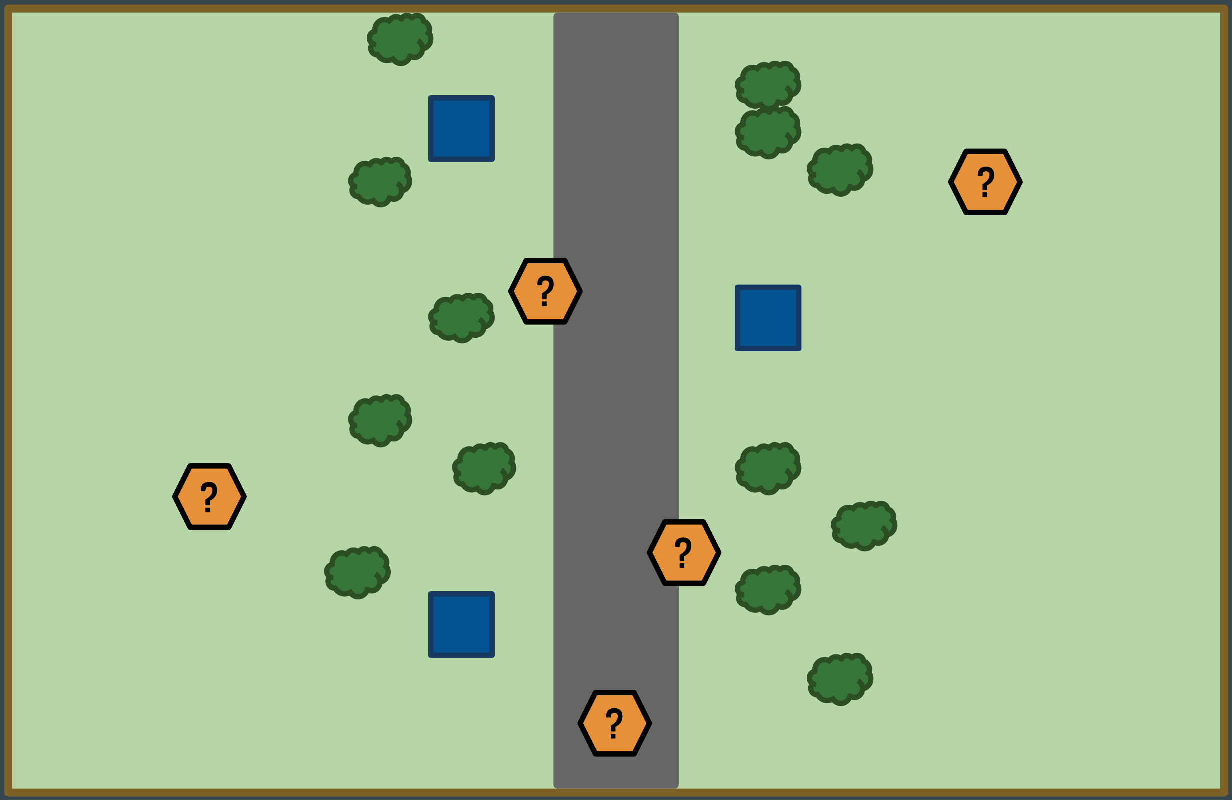
Example

Trees & Mailboxes

$$Z = \{\text{Tree, Tree, Mailbox}\}$$



$$Y = ?$$



What Do We Need To Consider?

Did we classify our observations correctly?

Did we observe everything in our FoV?

Did we interpret nothing as something?

~~Did we interpret two things as one thing?~~

Key Assumption 1: Each observation corresponds to exactly 1 object

Did we classify correctly?

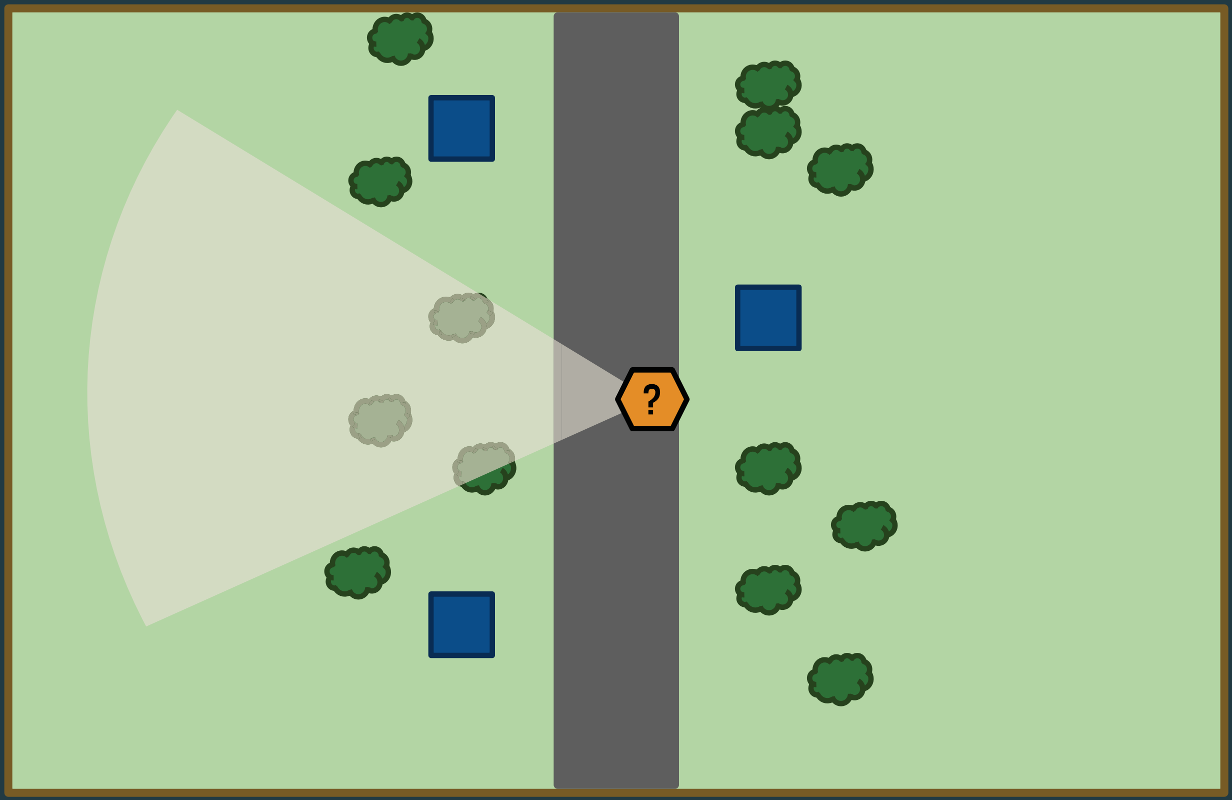
Assume: We see everything in our FoV

Assume: We never see something that doesn't exist

Solve

$$P(\,Z \mid Y(x), x\,)$$

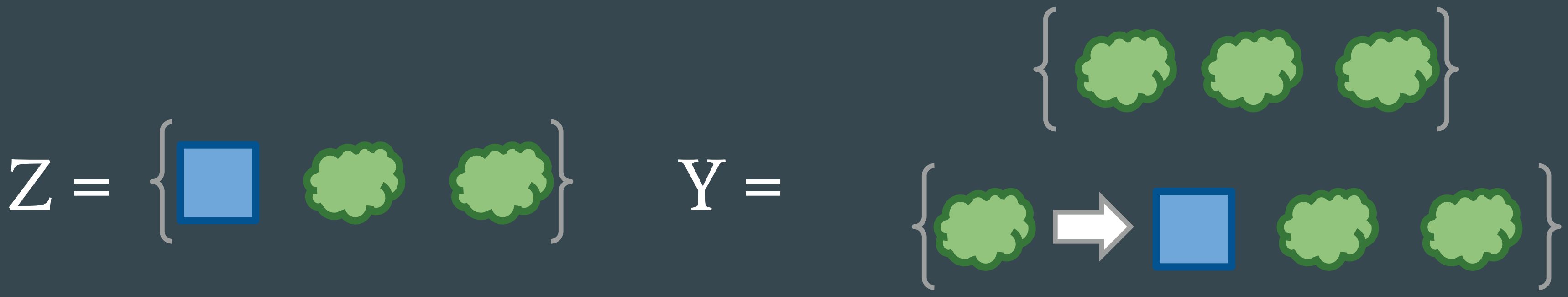
$$Y = ?$$



Did we classify correctly?

Assume: We see everything in our FoV

Assume: We never see something that doesn't exist



$$\underline{\boldsymbol{Pi}} = \left\{ Z \Rightarrow Y \right\}$$

Did we classify correctly?

Assume: We see everything in our FoV

Assume: We never see something that doesn't exist

This can keep expanding in relevant terms depending on the structure that detects objects

$$P(z_i \,|\, y_i , x) = P(c \,|\, y^{class}) \; P(s \,|\, c , y^{class}) \; P(b \,|\, y , x)$$

How often do we miss classify

If classifications have a score, is that score statistically likely

If we know the bearing we are viewing the object, does that effect classification?

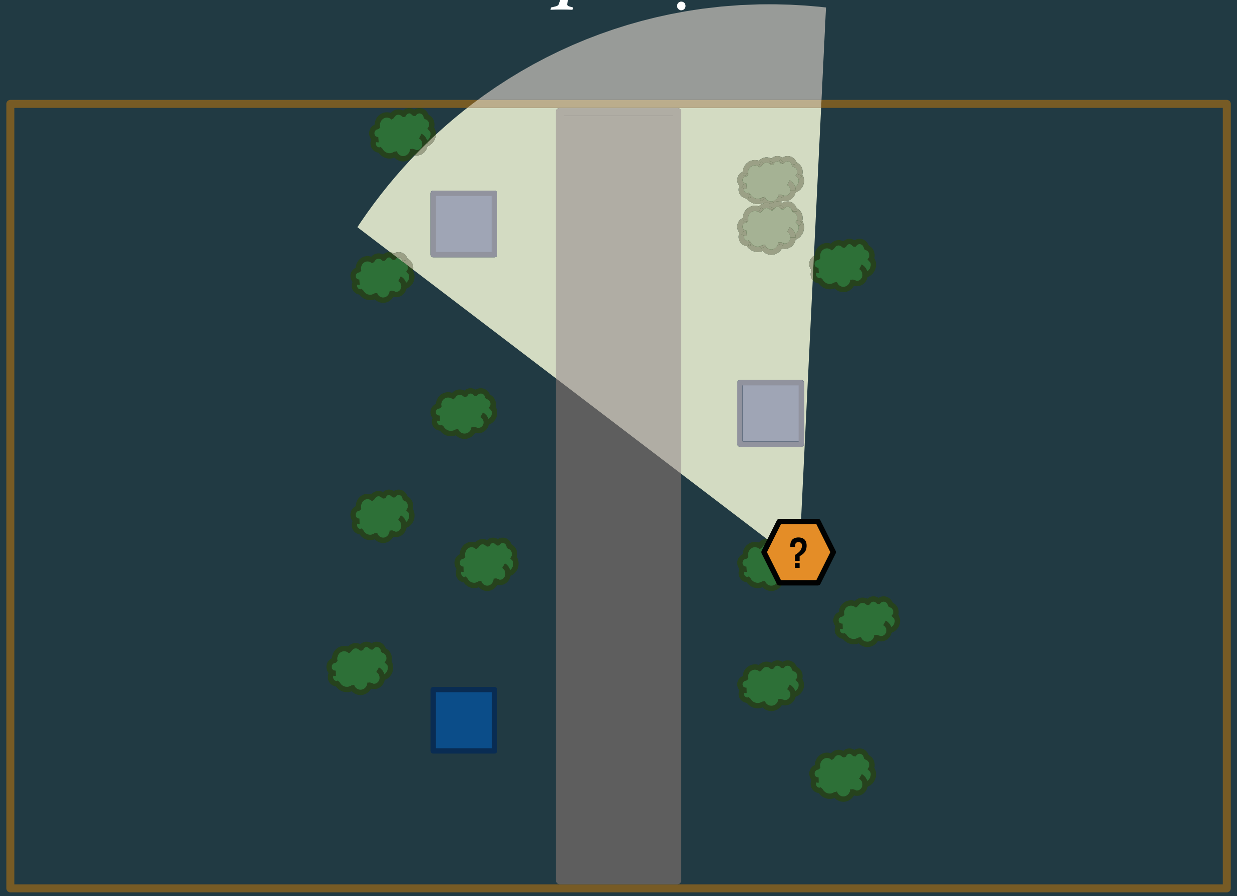
Did we classify correctly?

Assume: We see everything in our FoV

Assume: We never see something that doesn't exist

$$P(Z \,|\, Y(x) , x) = \sum^{pi} \prod_{i=0}^{|Y|} P(z_{i\, ,\, pi} \,|\, y_i , x)$$

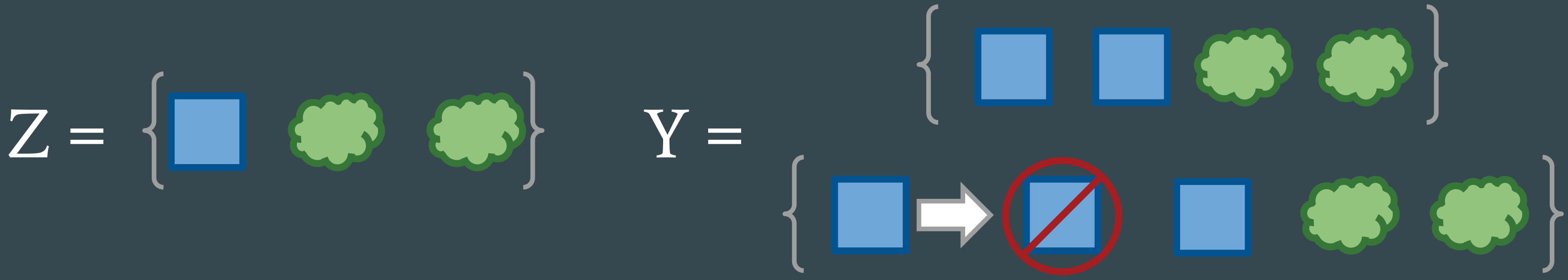
$Y = ?$



Did we see everything?

~~Assume: We see everything in our FoV~~

Assume: We never see something that doesn't exist



Did we see everything?

~~Assume: We see everything in our FoV~~

Assume: We never see something that doesn't exist

What if we see nothing

$$P(\, \emptyset \mid Y(x), x \,)$$

Did we see everything?

~~Assume: We see everything in our FoV~~

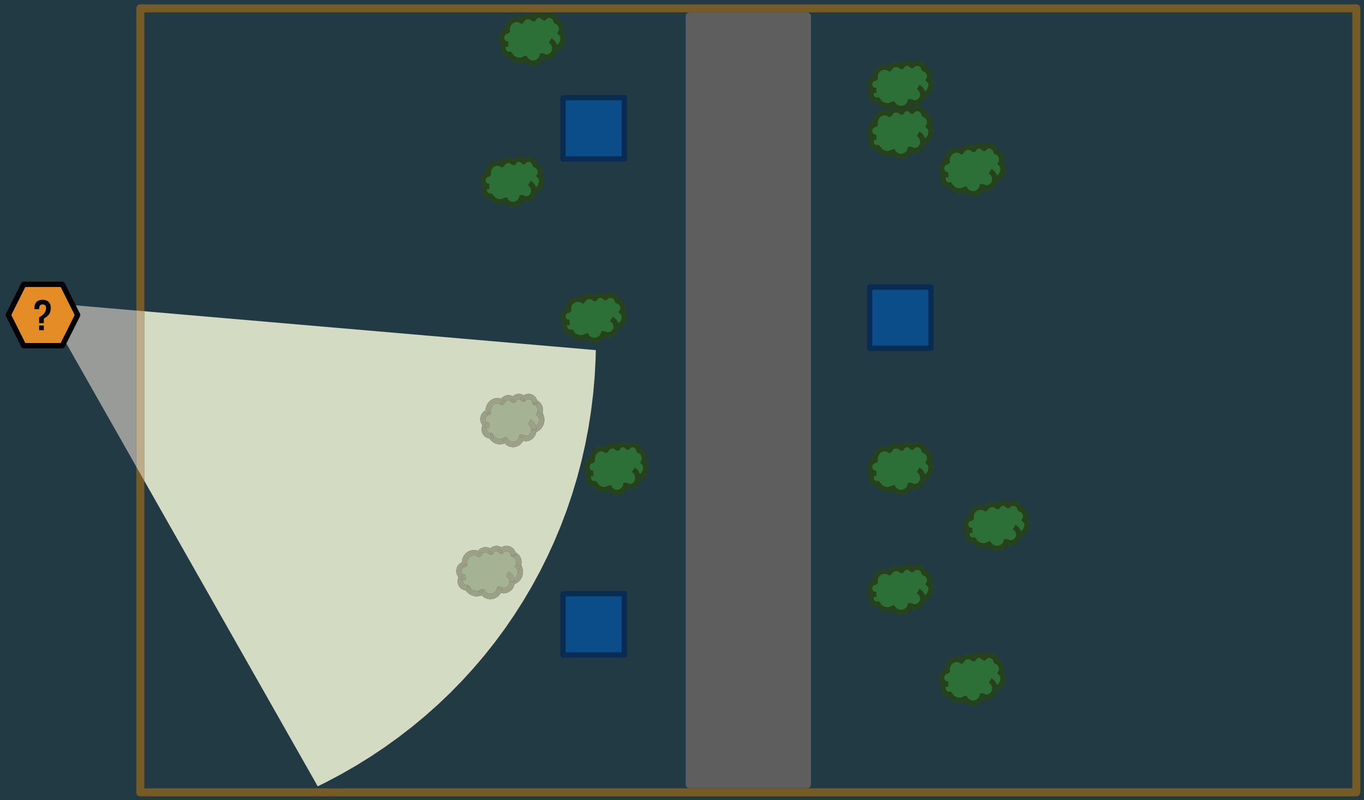
Assume: We never see something that doesn't exist

$$P(\emptyset \mid Y(x), x) = \prod_{i=0}^{|Y(x)|} (1 - P(y_i \mid x))$$

Key Assumption 2: An object is observed with some probability $P(y_i \mid x)$, and not with probability $1 - P(y_i \mid x)$

Key Assumption 3: For a given position x and map, any two object detections are independent

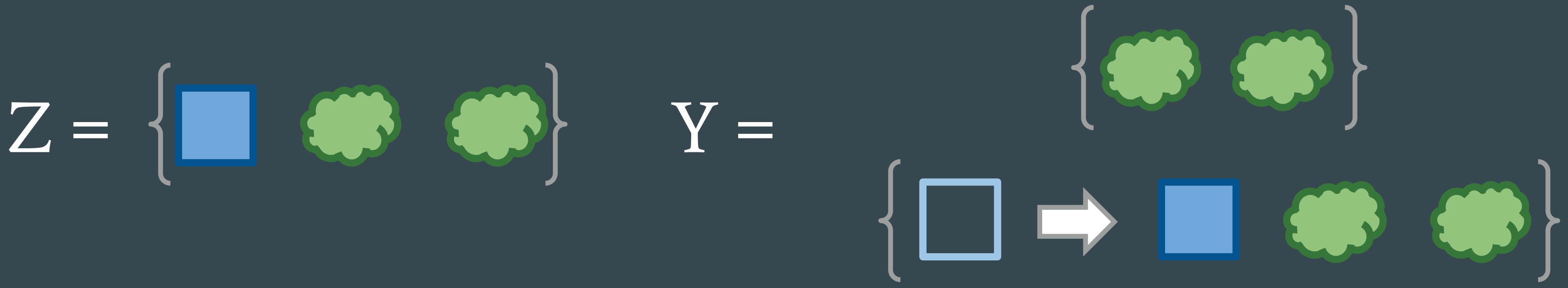
$$Y = ?$$



Did we see nothing as something?

~~Assume: We see everything in our FoV~~

~~Assume: We never see something that doesn’t exist~~



Did we see nothing as something?

~~Assume: We see everything in our FoV~~

~~Assume: We never see something that doesn't exist~~

What if there is nothing

$$P(\, Z \mid \cancel{\emptyset},\, x \,)$$

Did we see nothing as something?

~~Assume: We see everything in our FoV~~

~~Assume: We never see something that doesn't exist~~

$$P(\, Z \,|\, \emptyset \,,\, x \,) = \quad e^{\lambda} \prod^{|Z|} (\, \lambda \times K(z) \,)$$

Key Assumption 4: Noise is poisson distributed in time according to λ and spatially according to *K*(z)

Did we see nothing as something?

~~Assume: We see everything in our FoV~~

~~Assume: We never see something that doesn't exist~~

So what is *K(z)* ?

$$\frac{1}{|\,C\,|} \times \frac{1}{|\,S\,|} \times \frac{1}{|\,B\,|}$$

Possible Classifications

Possible Scores of Classifications

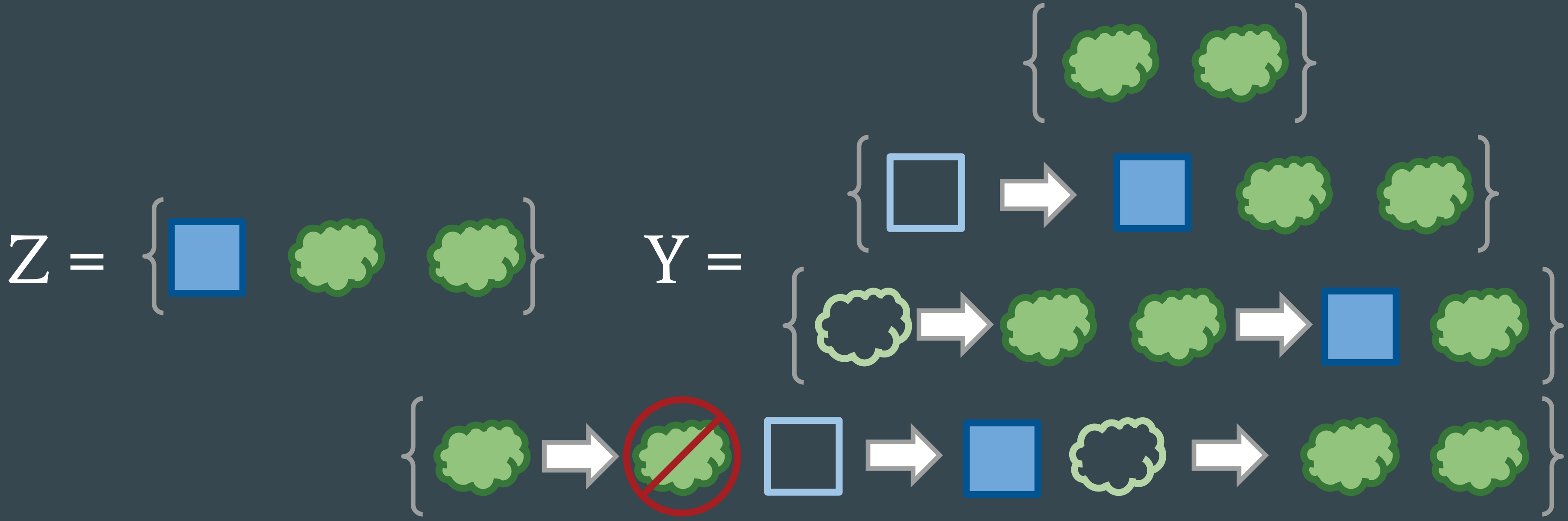
Possible Bearings

These correspond to the categories for classifying

Putting it all together

Solve

$P(\, Z \,|\, Y(x), x \,)$



Z =

Y =

Putting it all together

Solve

$P(\, Z \,|\, Y(x), x \,)$

Let

$|\, Z \,| = \,|\, \mathrm{Y} \,| - n + o$

Where *n* is missed detections and *o* is false detections

Putting it all together

Solve

$P(\, Z \,/\, Y(x), x \,)$

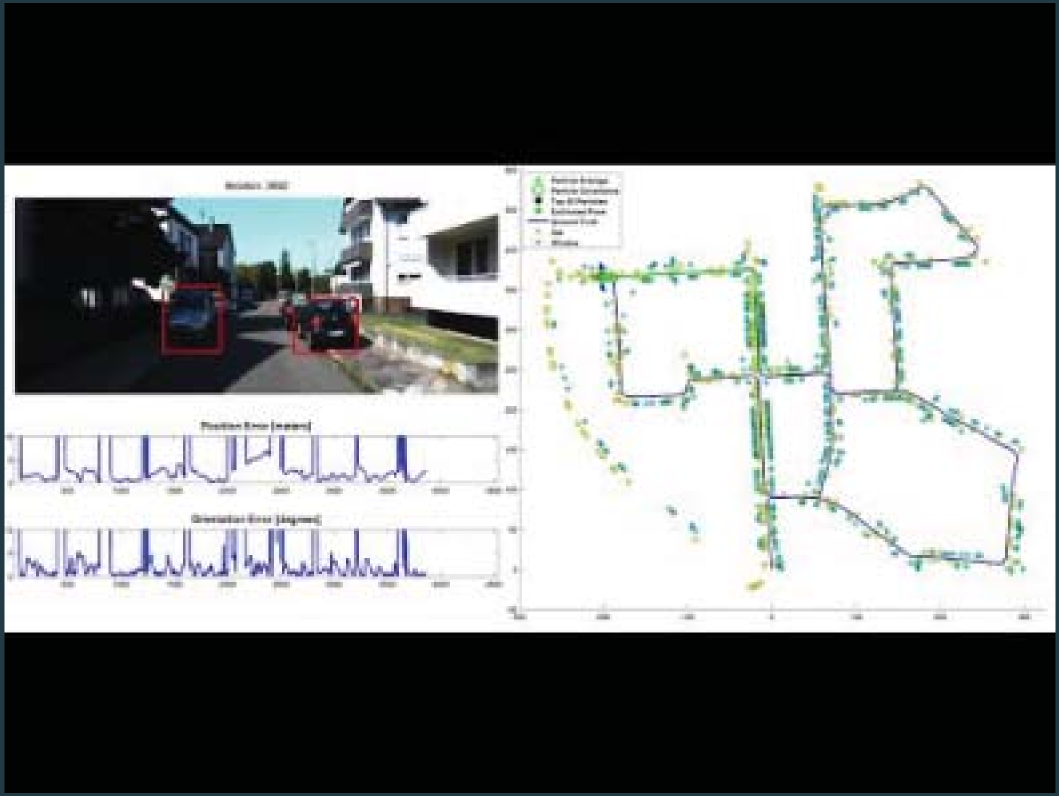
$P(\, Z \,/\, Y(x), x \,) =$

Pi now maps both actual and false detections

$$\sum^{pi} \prod_{i=0}^{|\,Y\,|} P(\, z_{i\,,\,pi} \,/\, y_i \,,\, x \,) * P(\, y_i \,/\, x \,)$$

$$\times \prod^{n} (\, 1 - P(\, y_i \,/\, x \,)\,) \quad \times \quad e^{\lambda} \prod^{o} (\lambda \times K(z_{pi}\,)\,)$$

Semantic Localization Video



Why?

Humans can’t walk into a room and reproduce an exact map, but we can store the most important aspects of the room and reason about what they’re used for.

Robots can store a pixel-perfect map of a room, but have no intuitive understanding.

This means we’re better at actually doing tasks with the environment.

How can we make robots localize and think more like humans?

Conclusion

1. Motivation for Semantic Localization

2. Particle Filters

3. Semantic Localization Implementation

References

F. Gustafsson, “Particle Filter Theory and Practice with Positioning Applications”, IEEE A&E Systems Magazine Vol. 25, No. 7, July 2010

O. Cappe, S. Godsill and E. Moulines, “An overview of existing methods and recent advances in sequential Monte Carlo”, IEEE Proceedings, Vol. 95 No. 5 pp. 899–924 2007

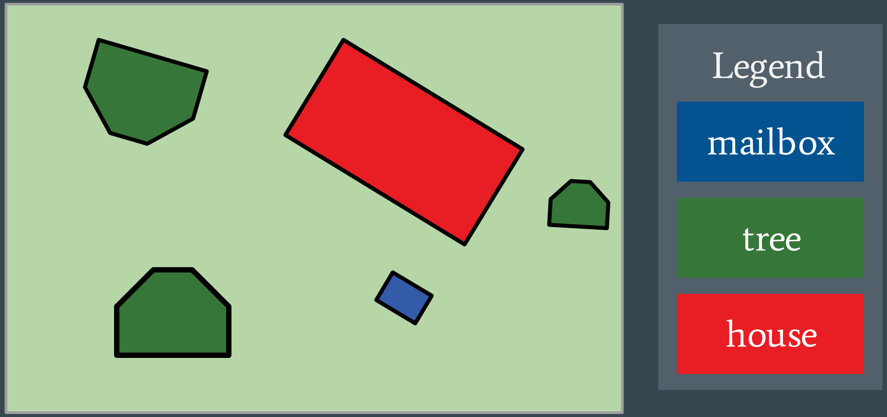
N. Atanasov, M. Zhu, K. Daniilidis, and G. Pappas, “Localization from Semantic Observations via the Matrix Permanent ”, The International Journal of Robotics Research, vol. 35 no. 1-3, pp.73-99, January 2016

http://www.us.orienteering.org/orienteers/training/getting-started

Various YouTube videos embedded in slides

Appendix: Our Semantic Map Definition

- We will use a labeled object map $\mathcal{M}$ which is a set of labeled N objects $< P_i , c_i >$ for $i = 1...N$
- P_i is an ordered list of vertices $<x, y>$ of the polygon boundary



- c_i is the class of the object, e.g. tree
- Our robot pose x_t will be a position and orientation $<x, y, \theta>$
- The actuation model can be any continuous dynamical probability model
- Must define the observation noise model