

SLAM: Simultaneous Localization and Mapping

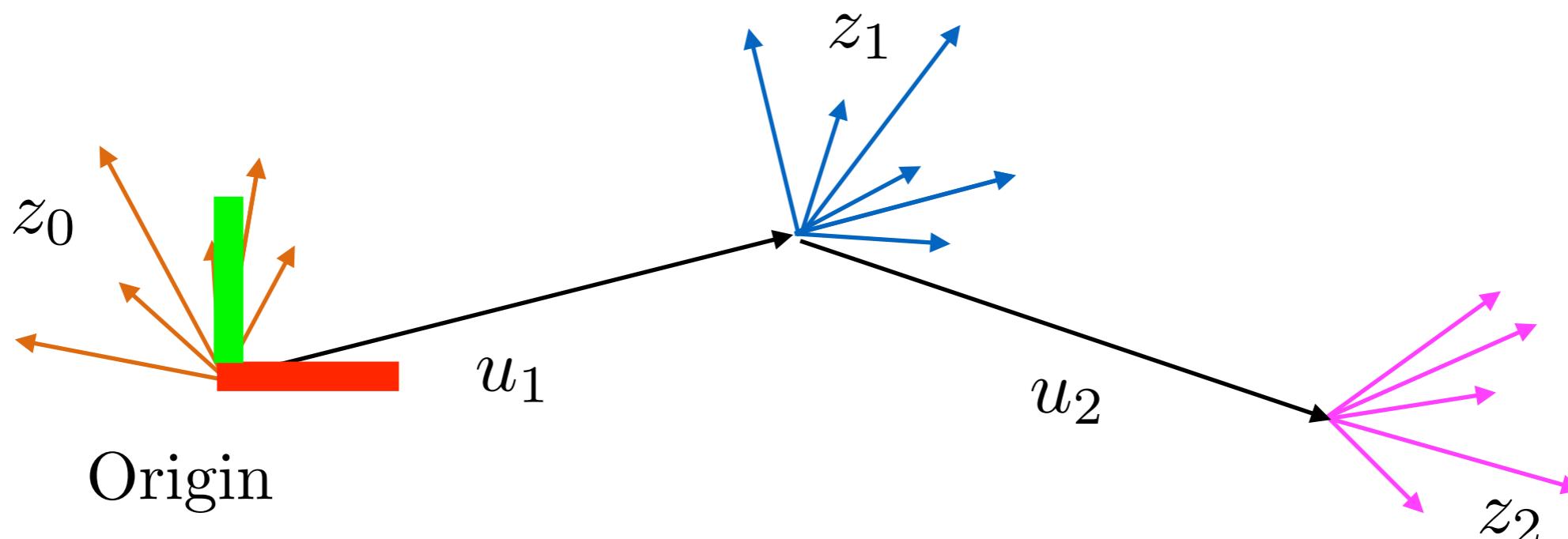
Instructor: Chris Mavrogiannis

TAs: Kay Ke, Gilwoo Lee, Matt Schmittle

*Slides based on or adapted from Sanjiban Choudhury, Dieter Fox, Michael Kaess

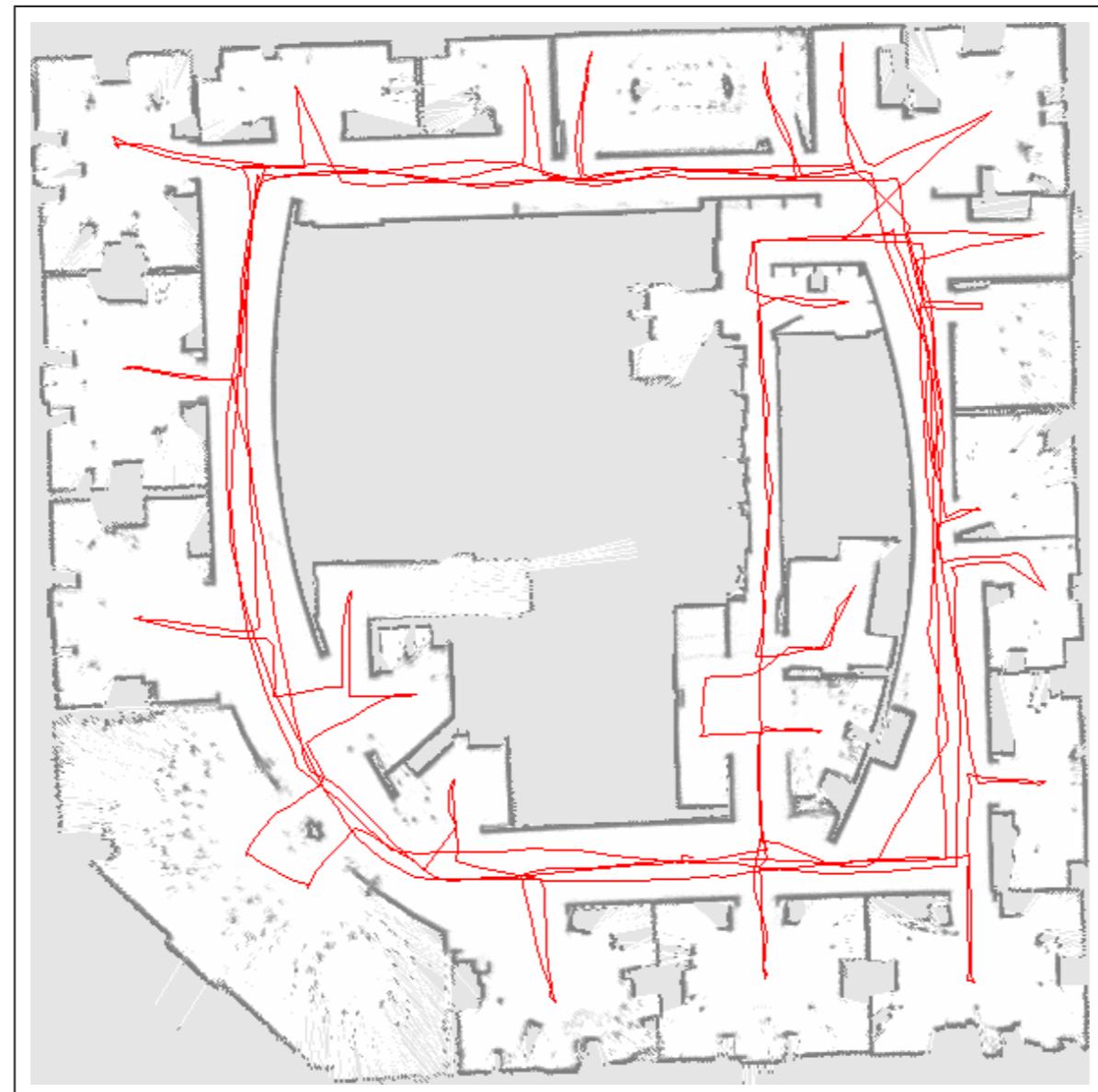
The SLAM problem

Robot is moving through a static unknown environment



Given a series of **controls** and **measurements**,
estimate **state** and **map**

What we want ...



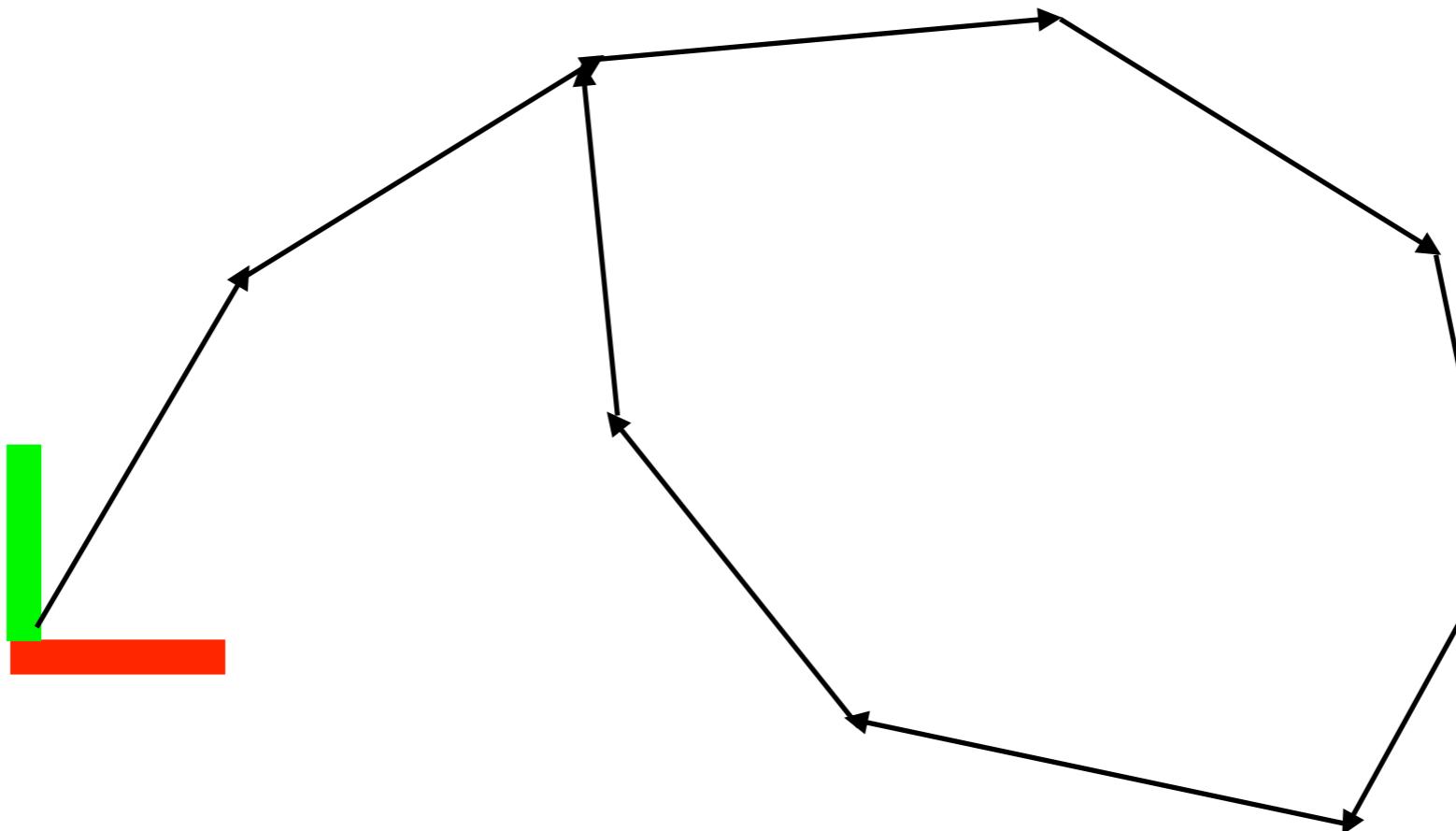
Need to figure out two things:

Correct relative movements
between successive measurements

Closing the loop
globally

Spilling the beans on SLAM

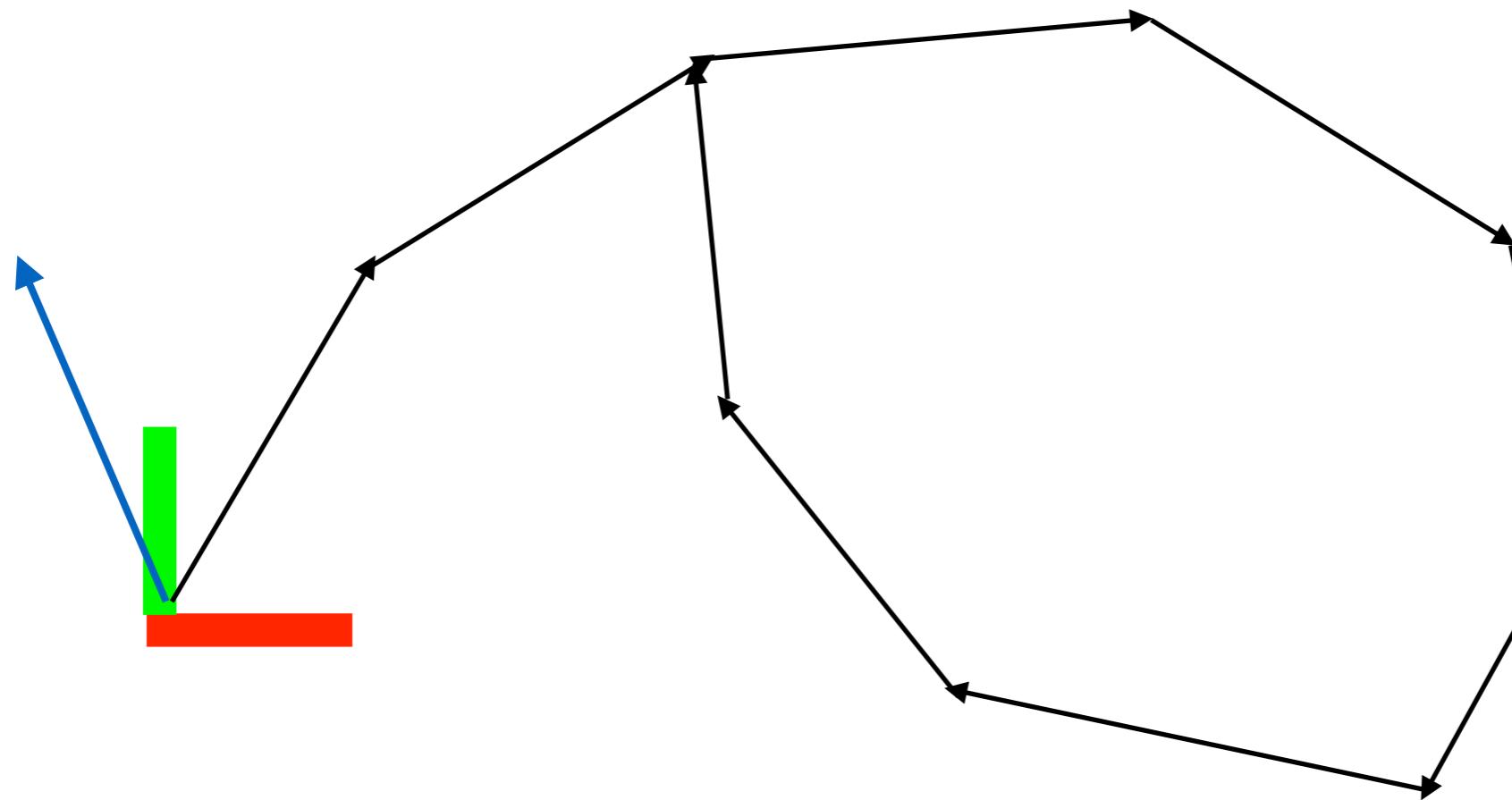
Let's assume this was the ground truth....



→ Ground truth

Spilling the beans on SLAM

Odometry is really really noisy!

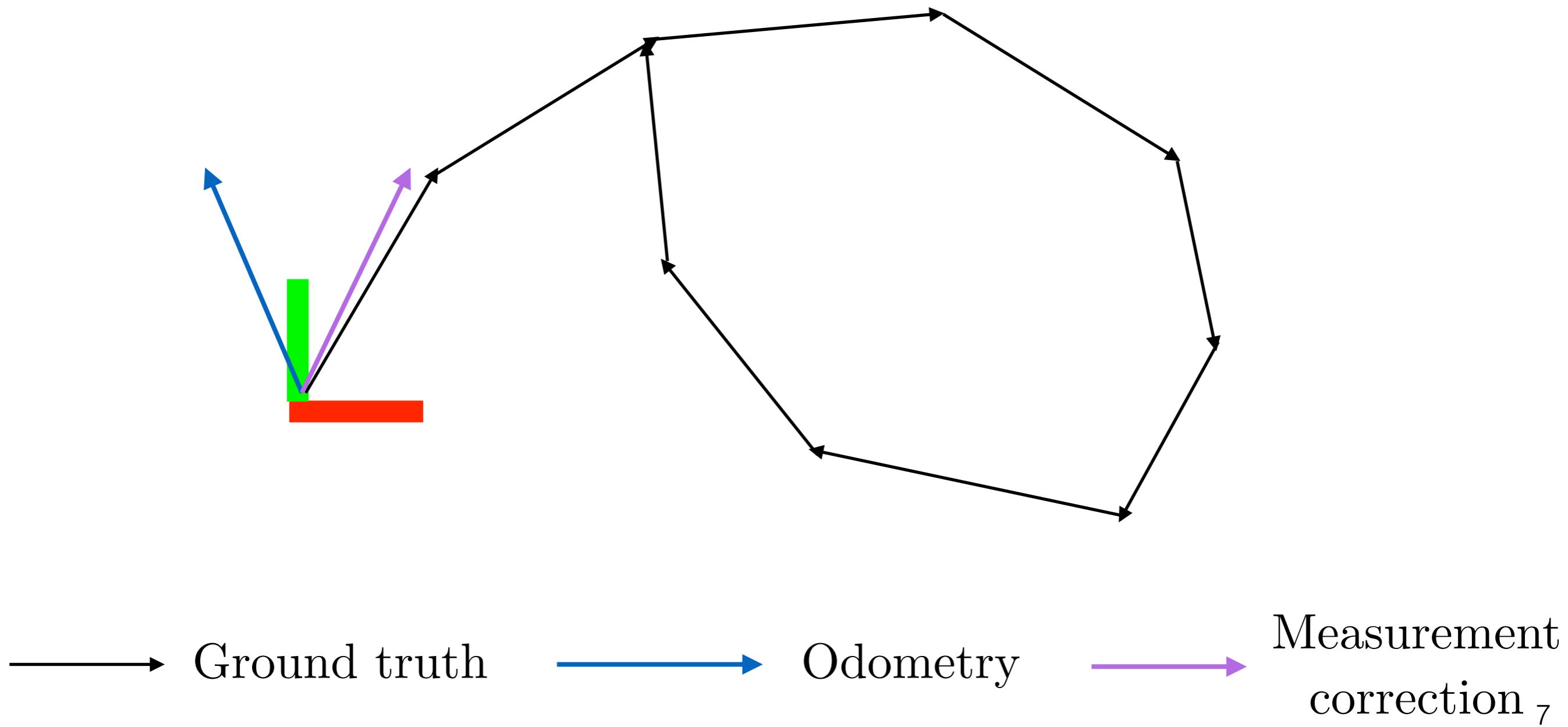


→ Ground truth

→ Odometry

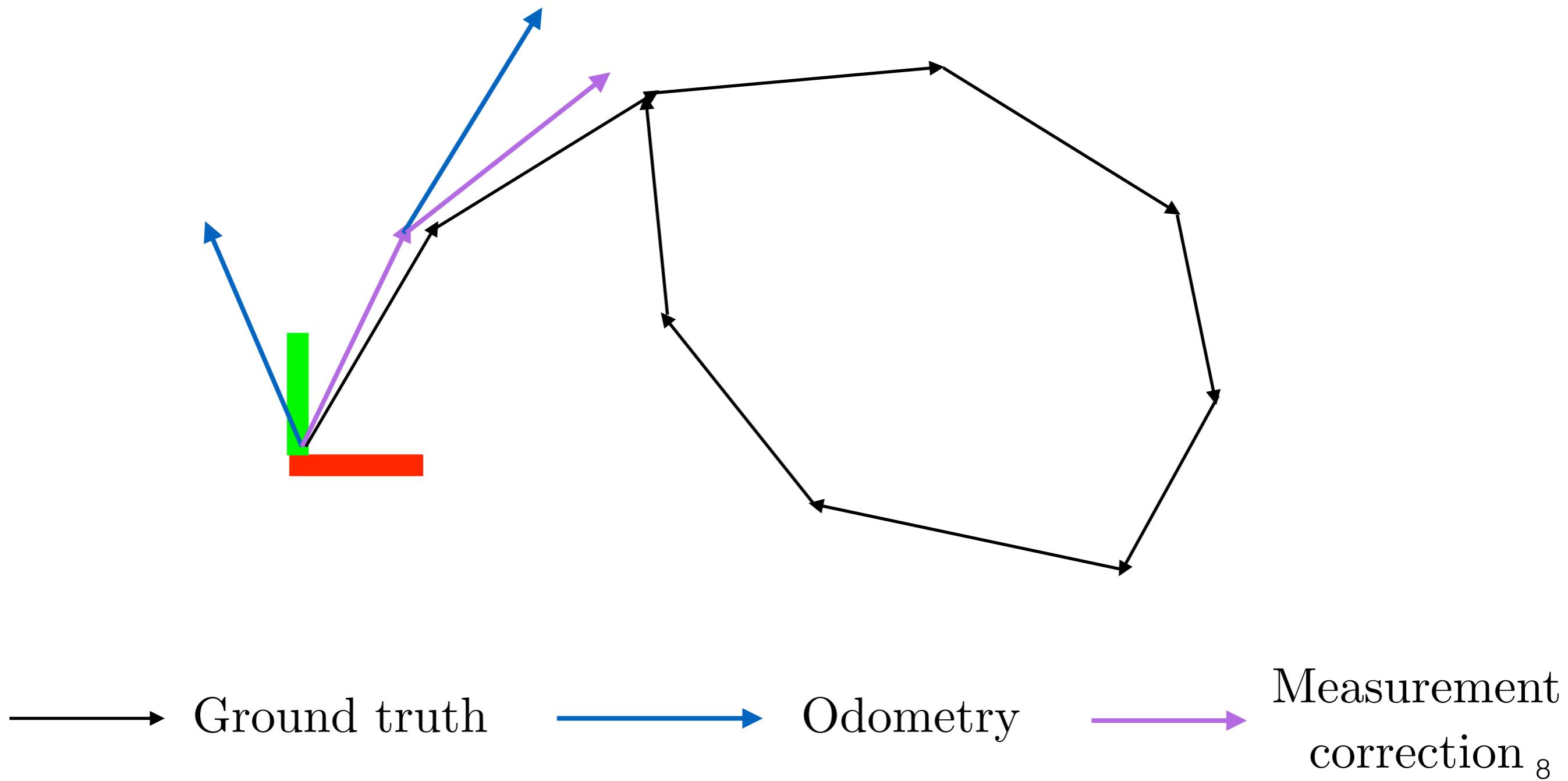
Spilling the beans on SLAM

Measurements can help correct this somewhat



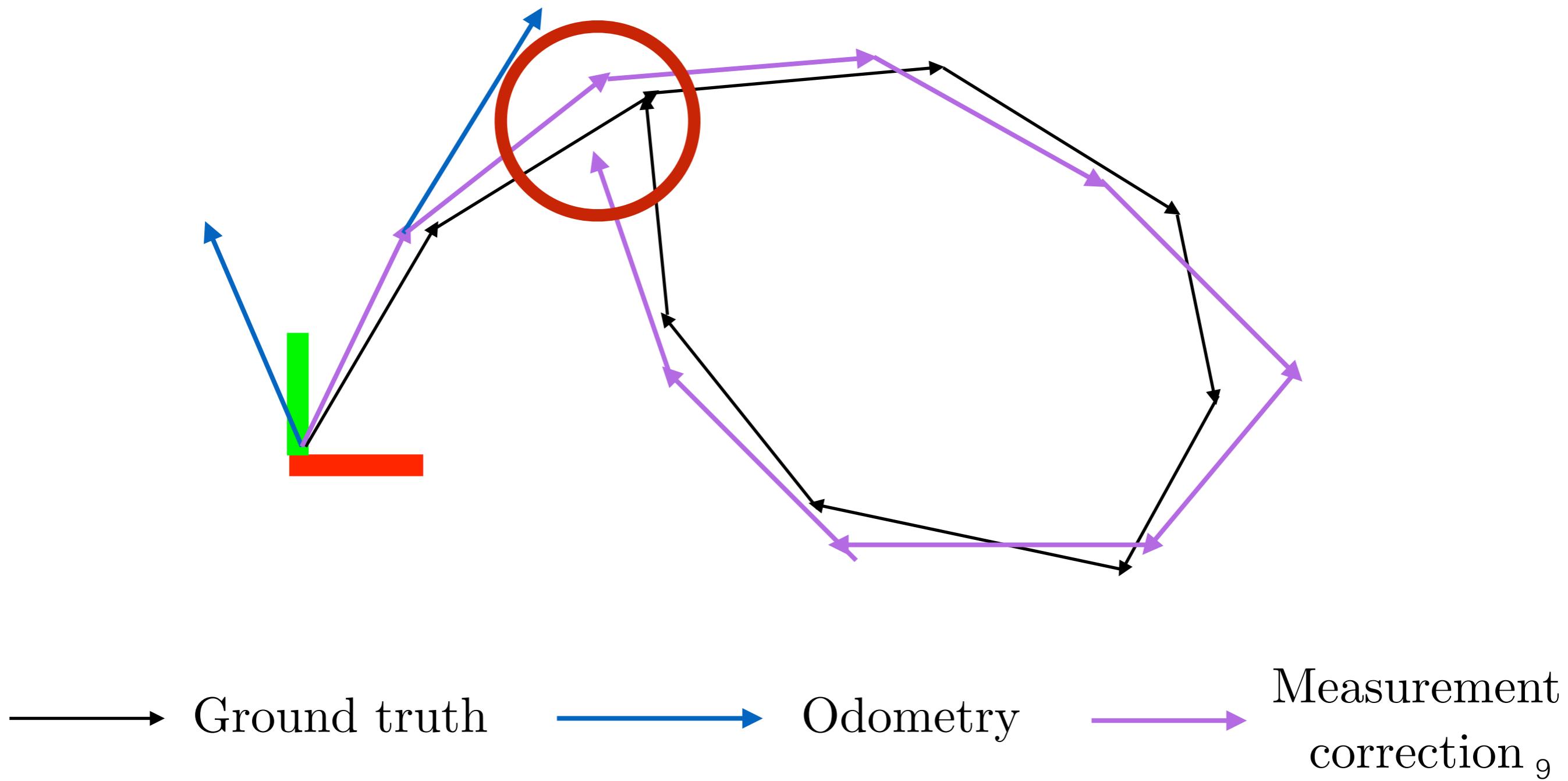
Spilling the beans on SLAM

Relative error accumulates ...



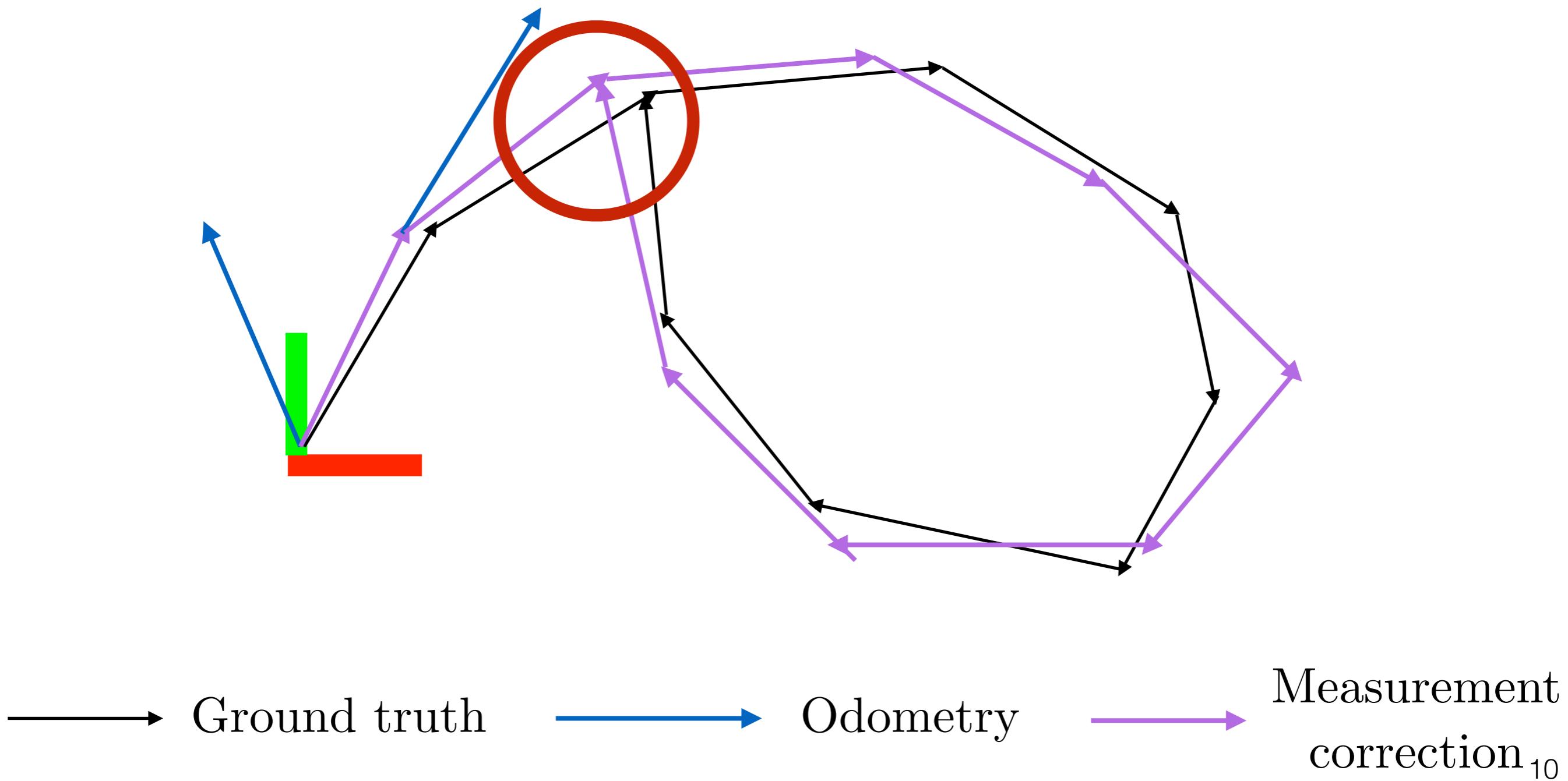
Spilling the beans on SLAM

Eventually robot comes back to a familiar place loop closure!



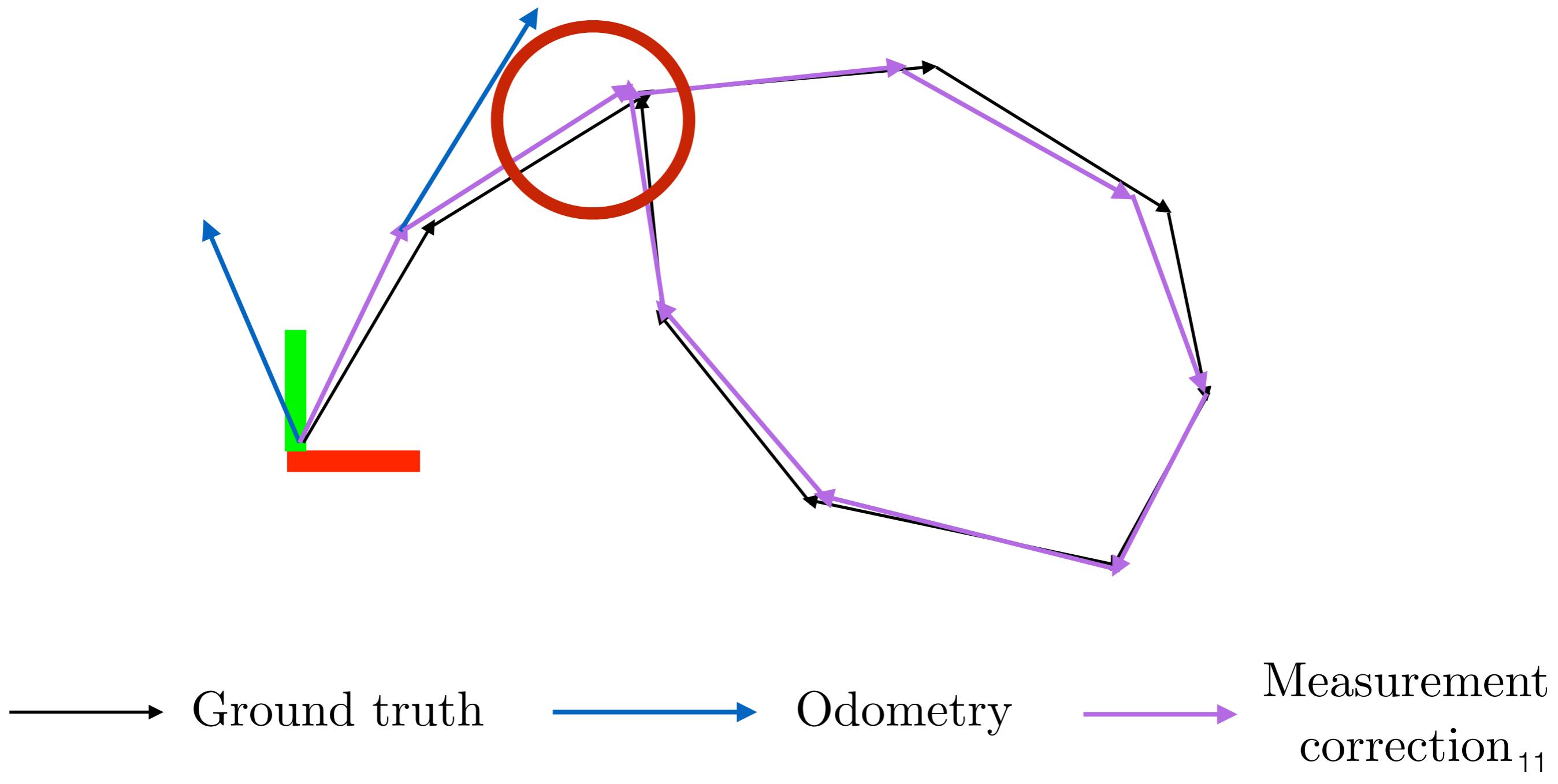
Spilling the beans on SLAM

If we are filtering, we can only fix the last pose estimate



Spilling the beans on SLAM

If we are **optimizing**, we can backprop errors in time!



Today's objective

1. How do we solve the chicken-or-egg problem in SLAM?
2. SLAM as an optimization instead of filtering

Bayes filter is a powerful tool



Localization



Mapping

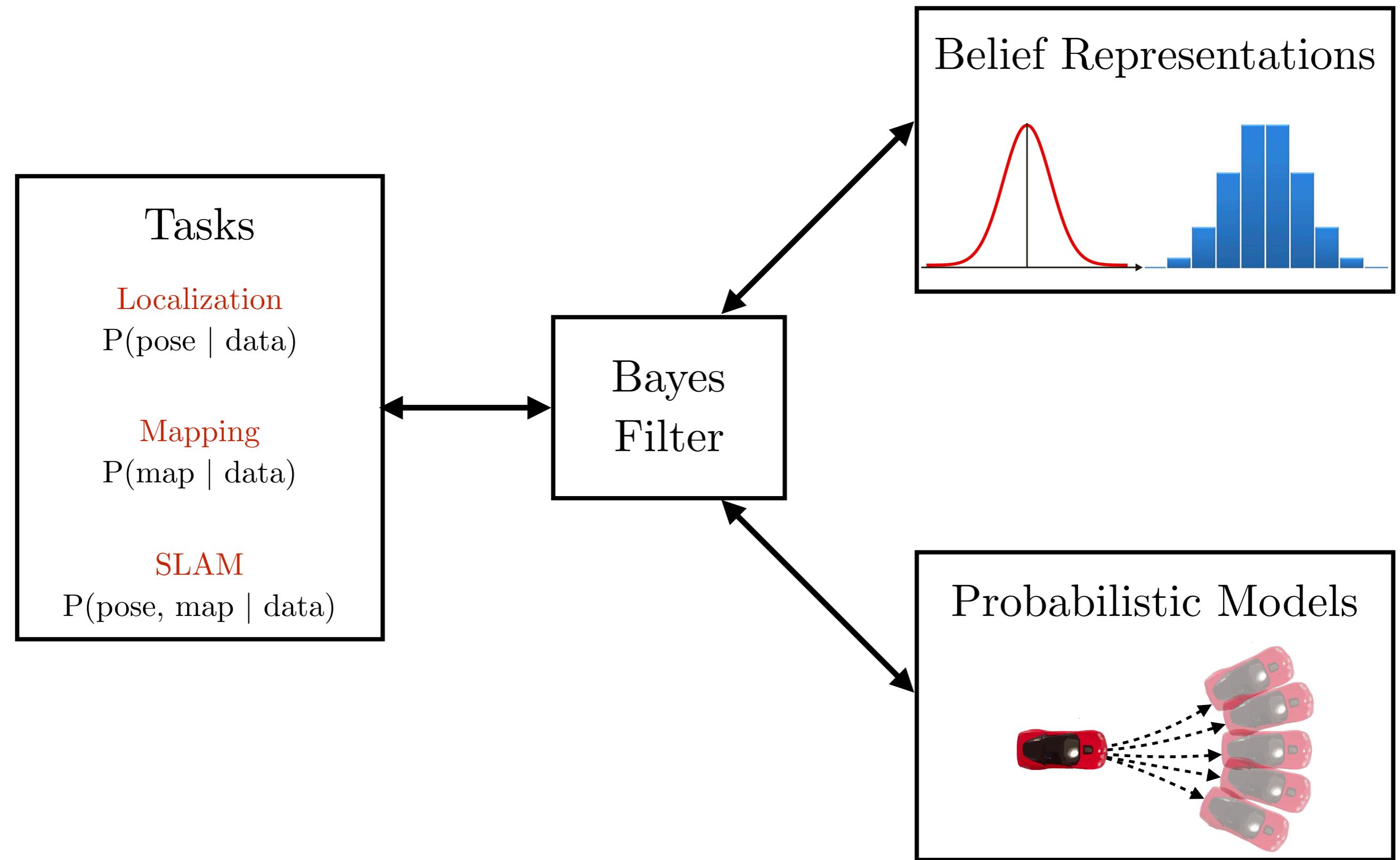


SLAM



POMDP

Assembling Bayes filter



Tasks that we will cover

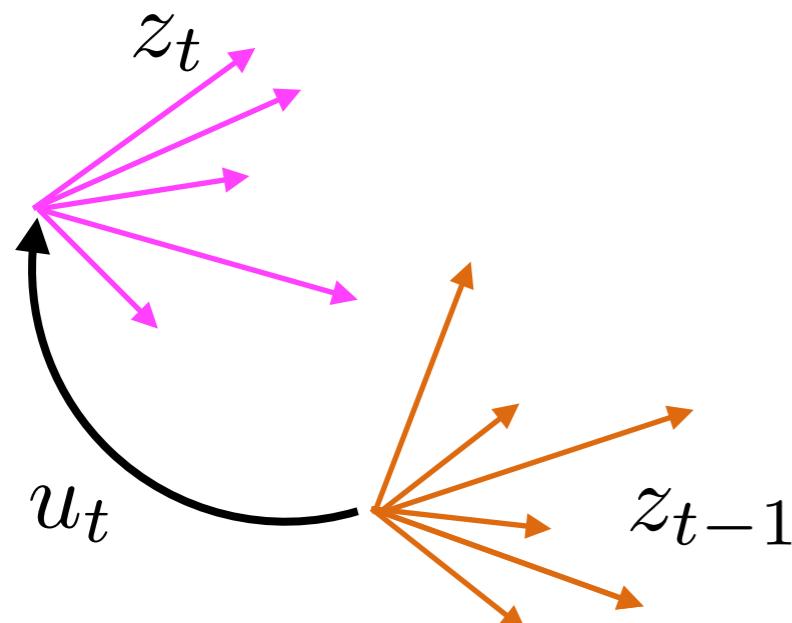
| Tasks | Belief Representation | Probabilistic Models |
|---|----------------------------------|--|
| Localization $P(\text{pose} \mid \text{data})$ (Week 3) | Gaussian / Particles | Motion model Measurement model |
| Mapping $P(\text{map} \mid \text{data})$ (Week 4) | Discrete (binary) | Inverse measurement model |
| SLAM $P(\text{pose, map} \mid \text{data})$ (Week 4) | Particles+GridMap (pose, map) | Motion model, measurement model, <i>correspondence model</i> |

SLAM as just another Bayes filtering problem

Task:

$$\text{SLAM} \\ P(\text{map, pose} \mid \text{data})$$

What is the data?

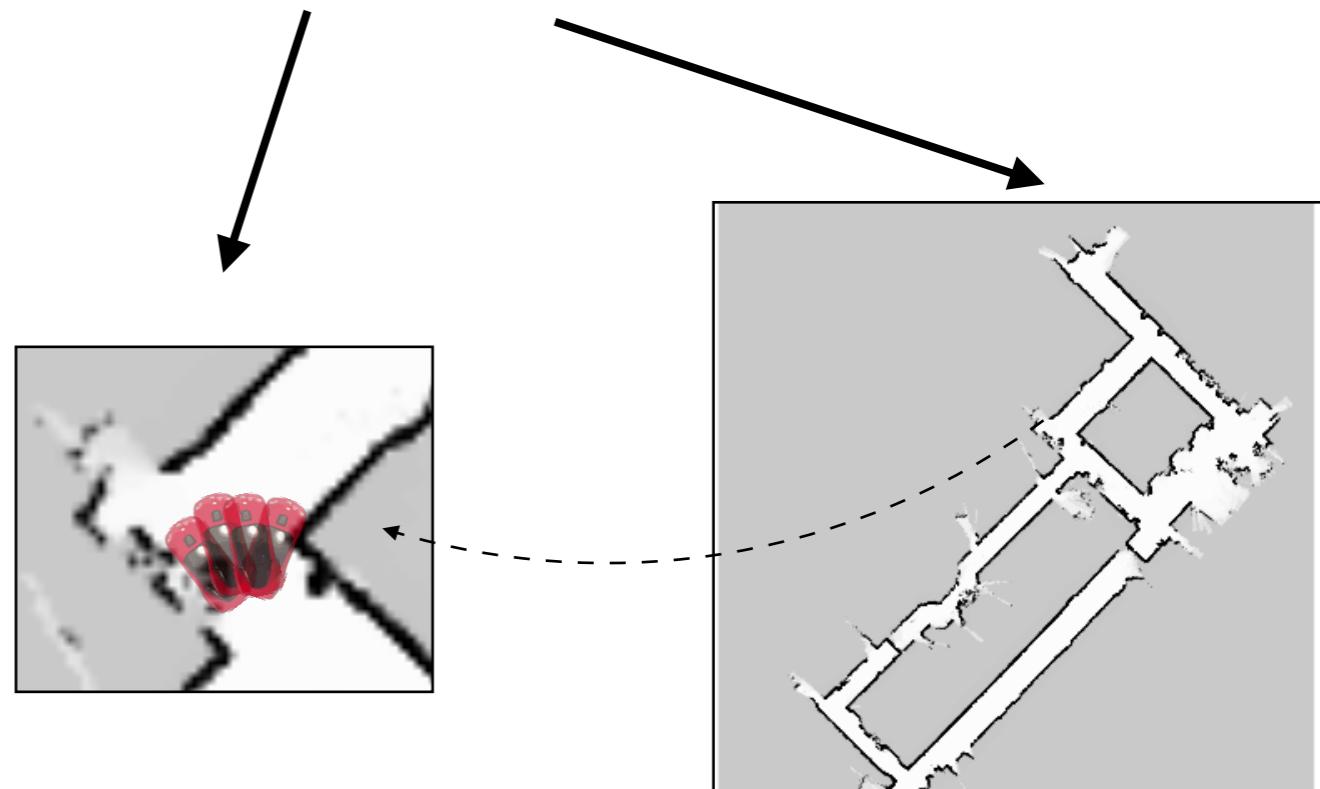


Stream of controls
and measurements

$$u_{1:t}, z_{1:t}$$

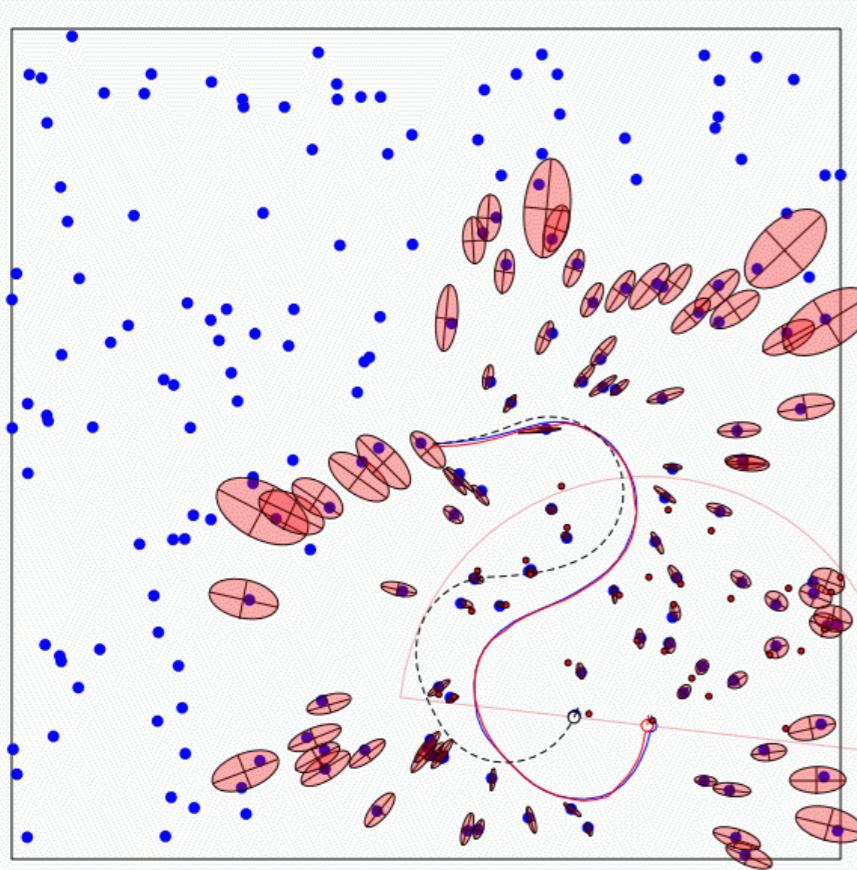
What is the belief representation?

$$P(x_t, m \mid u_{1:t}, z_{1:t})$$



Different map representations

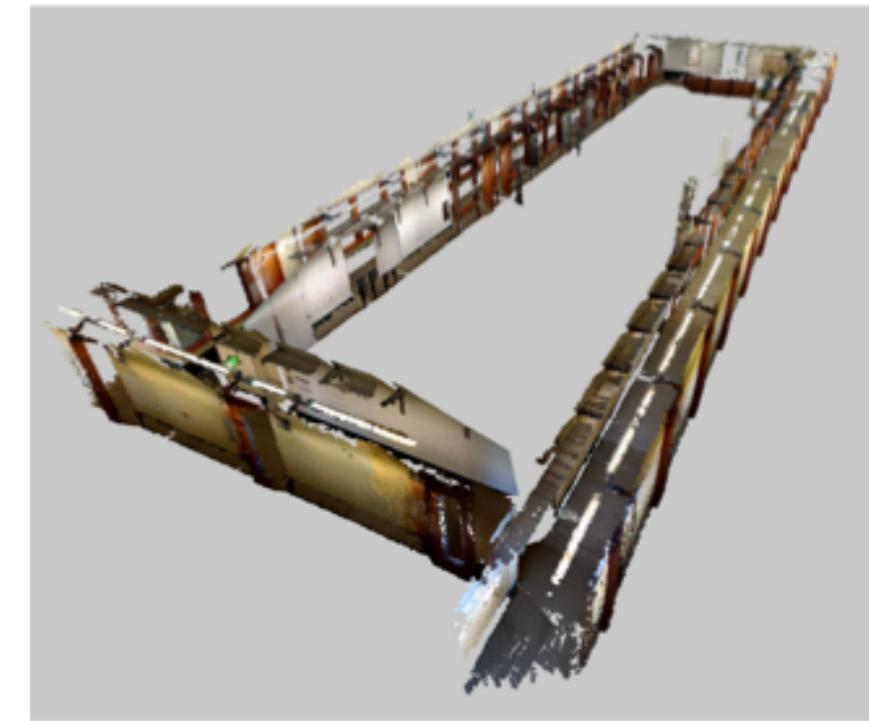
We are free to use any of the map representations we discussed



Feature maps



Grid maps



Surface maps

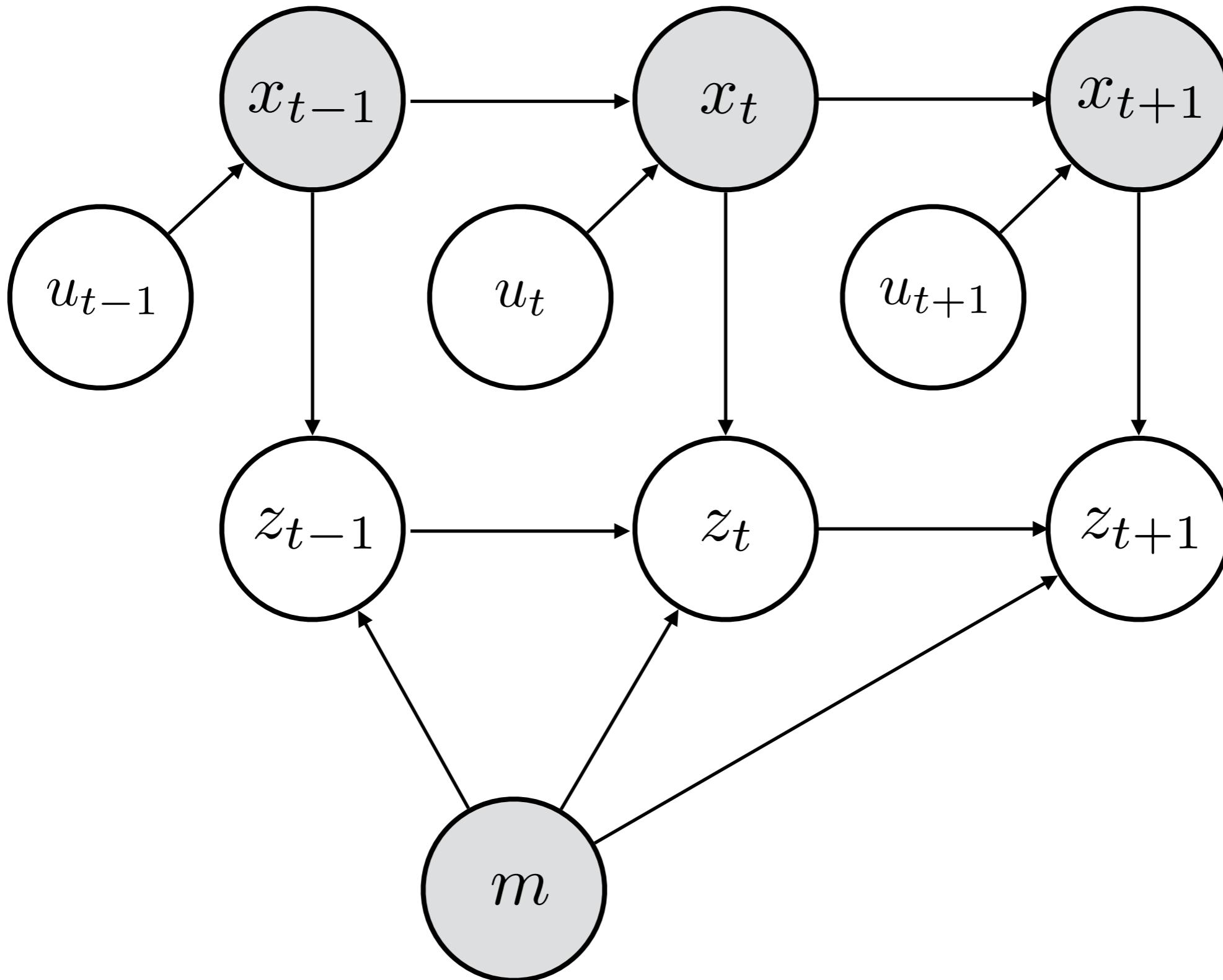
Why is SLAM hard?

Chicken-or-egg problem:

Given a map, we can localize

Given the pose, we can build map

Graphical model of SLAM



If SLAM can be expressed as a
Bayes filtering problem,
let's use our favorite Bayes filter...

Particle Filter!

Generalized particle filters

Step 0: Start with a set of particles

$$bel(x_{t-1}) = \{x_{t-1}^1, x_{t-1}^2, \dots, x_{t-1}^M\}$$

Step 1: Sample particles from proposal distribution

$$\overline{bel}(x_t) = \{\bar{x}_t^1, \dots, \bar{x}_t^M\}$$

Step 2: Compute importance weights

$$w_t^k = \frac{bel(x_t)}{\overline{bel}(x_t)} \left(= \eta P(z_t|x_t) \right)$$

Step 3: Resampling

Draw M samples from weighted distribution

Problem: Space of maps too large!!

$$P(x_t, m | u_{1:t}, z_{1:t})$$

How big is this space?

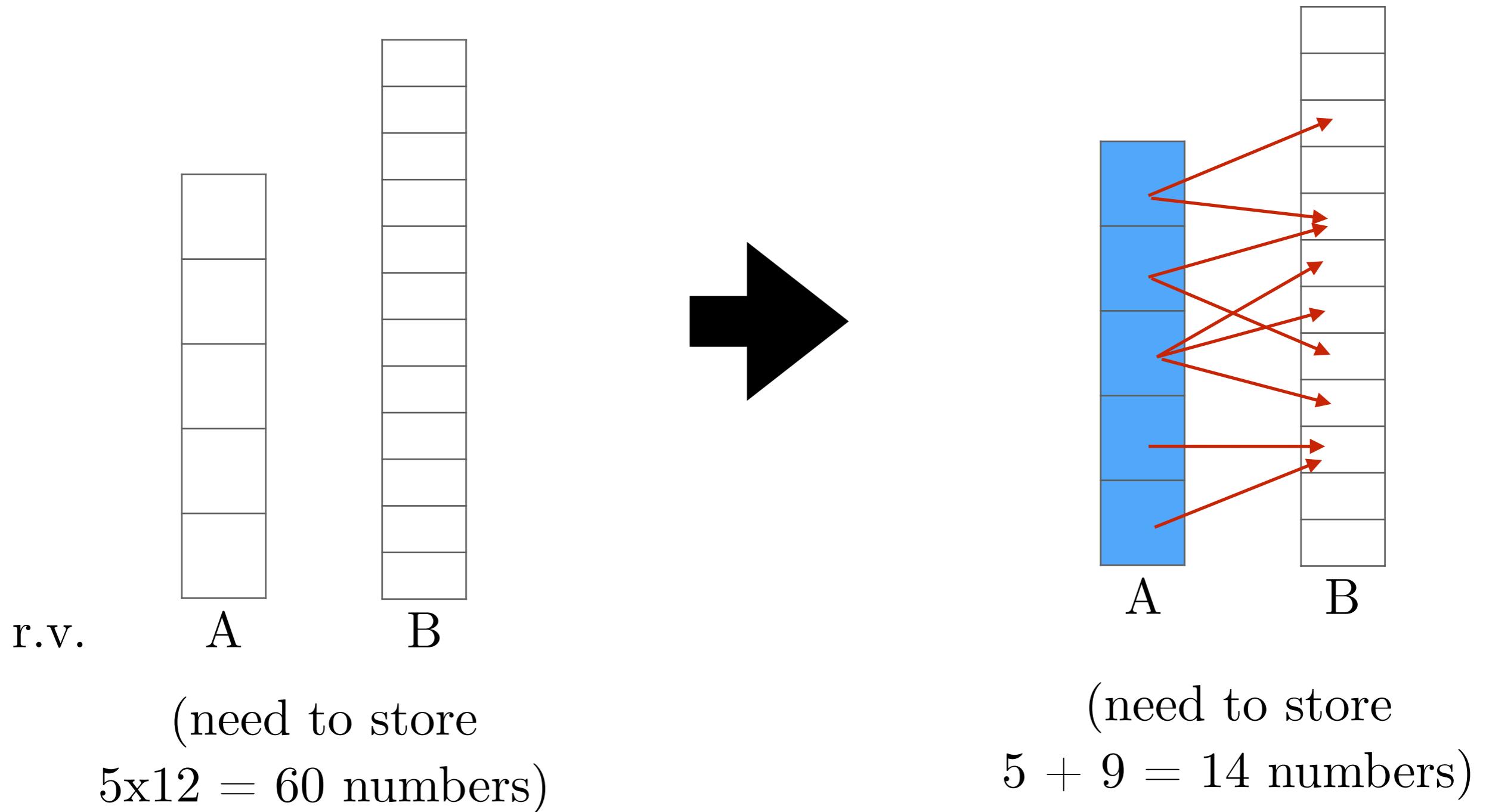
$$\left(2^M \right)$$

If we were to sample particles, what is the likelihood that they would explain the measurements??

$$P(z_t | m, x_t)$$

Key idea: Exploit dependencies

Even if a space is absurdly large, dependencies within the space can significantly shrink the space of possibilities we should consider.



Key idea: Exploit dependencies

Is there a dependency in this gigantic combined state space?

Yes!

The map depends on the history of poses of the robot

Rao-Blackwellization

Factorization to exploit dependencies between variables

$$P(a, b) = P(b|a)P(a)$$

If $P(b|a)$ can be computed efficiently

Represent

$$P(a)$$

with samples

Compute

$$P(b|a)$$

for every sample

Applying the factorization trick

$$P(x_{1:t}, m | z_{1:t}, u_{1:t})$$

state map data
history

$$= P(x_{1:t} | z_{1:t}, u_{1:t}) \ P(m | x_{1:t}, z_{1:t}, u_{1:t})$$

state
history data map state
 history data

$$= P(x_{1:t} | z_{1:t}, u_{1:t}) \prod_{i=1}^N P(m_i | x_{1:t}, z_{1:t})$$

state
 history data

(Particle filter to estimate this)

(Occupancy map)

How do we compute a PF over state history?

For simplicity, each particle only stores the current state at timestep t
(just like you did in your assignment)

However the weights of the particle are computed based on
its path through time.

How do we compute a PF over state history?

Let's jump straight to the importance sampling step

$$w_t = \frac{bel(x_{1:t})}{\overline{bel}(x_{1:t})} = \frac{P(x_{1:t} | \textcolor{red}{z}_t, z_{1:t-1}, u_{1:t})}{P(x_{1:t} | z_{1:t-1}, u_{1:t})}$$

$$\propto \frac{P(\textcolor{red}{z}_t | x_{1:t}, z_{1:t-1}, u_{1:t}) P(x_{1:t} | z_{1:t-1}, u_{1:t})}{P(x_{1:t} | z_{1:t-1}, u_{1:t})}$$

$$\propto P(\textcolor{red}{z}_t | x_t, x_{1:t-1}, z_{1:t-1}, u_{1:t})$$

$$\propto \sum_m P(\textcolor{red}{z}_t | x_t, m) P(m | x_{1:t-1}, z_{1:t-1}, u_{1:t})$$

$$\approx P(\textcolor{red}{z}_t | x_t, \hat{m})$$

(meas) (most likely map from prev timestep)

where $\hat{m} = \arg \max_m P(m | x_{1:t-1}, z_{1:t-1}, u_{1:t-1})$

FastSLAM

Proposed by Montemerlo 2002

**Particle
1**

x, y, θ

Occupancy grid map

**Particle
2**

x, y, θ

Occupancy grid map

⋮

**Particle
 N**

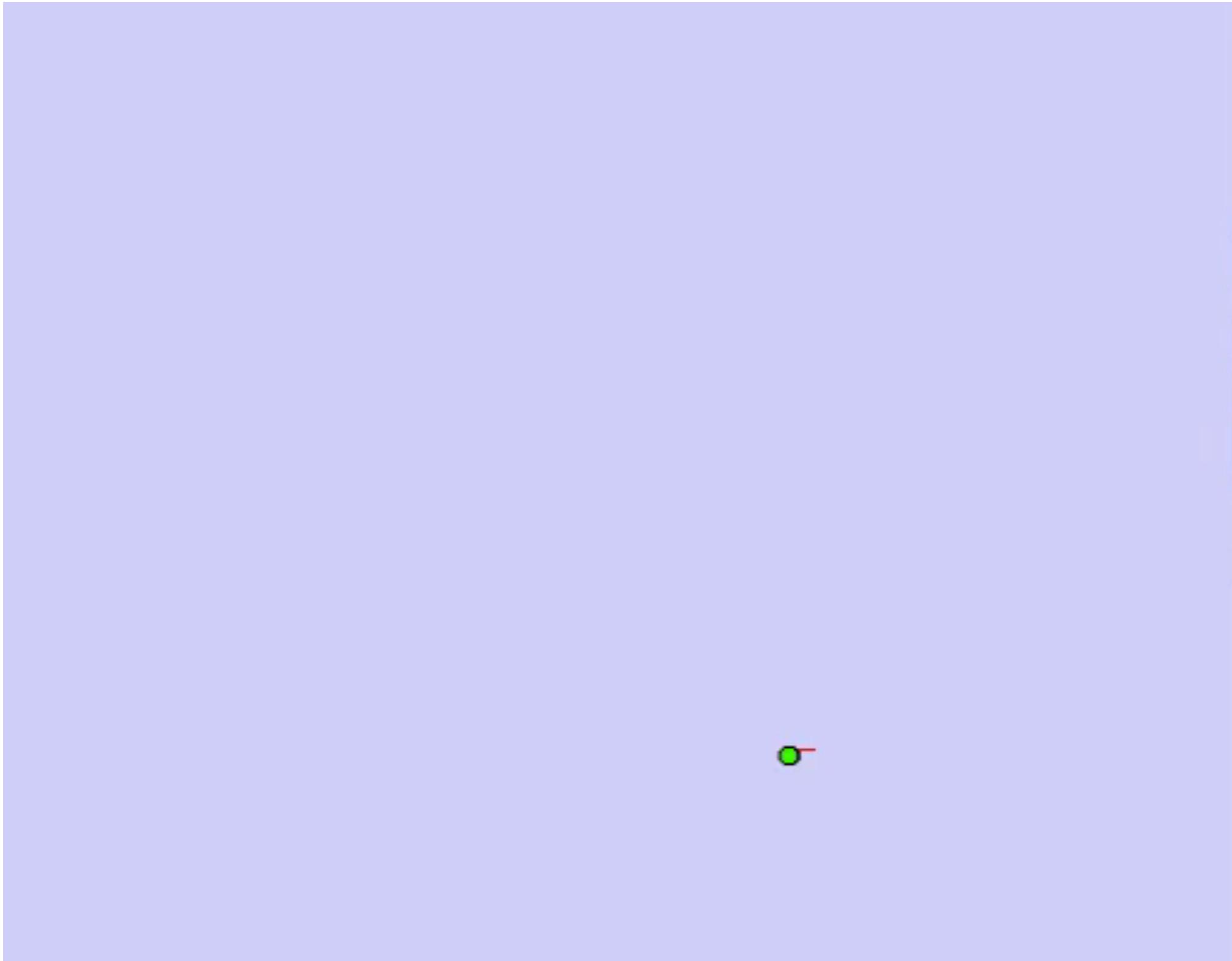
x, y, θ

Occupancy grid map

FastSLAM Algorithm

```
1:   Algorithm FastSLAM_occupancy_grids( $\mathcal{X}_{t-1}, u_t, z_t$ ):  
2:      $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$   
3:     for  $k = 1$  to  $M$  do  
4:        $x_t^{[k]} = \text{sample\_motion\_model}(u_t, x_{t-1}^{[k]})$   
5:        $w_t^{[k]} = \text{measurement\_model\_map}(z_t, x_t^{[k]}, m_{t-1}^{[k]})$   
6:        $m_t^{[k]} = \text{updated\_occupancy\_grid}(z_t, x_t^{[k]}, m_{t-1}^{[k]})$   
7:        $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[k]}, m_t^{[k]}, w_t^{[k]} \rangle$   
8:     endfor  
9:     for  $k = 1$  to  $M$  do  
10:      draw  $i$  with probability  $\propto w_t^{[i]}$   
11:      add  $\langle x_t^{[i]}, m_t^{[i]} \rangle$  to  $\mathcal{X}_t$   
12:    endfor  
13:    return  $\mathcal{X}_t$ 
```

FastSLAM in action!



Haehenl et al.

SLAM resources

OpenSLAM
Give your algorithm to the community

<https://openslam-org.github.io/>

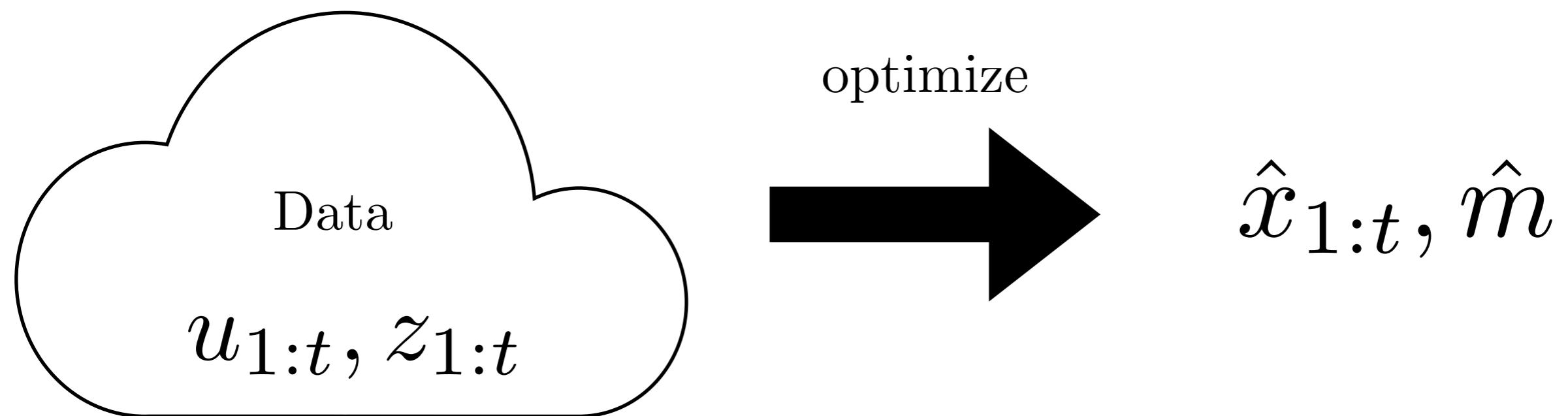
Do we really need a probability distribution?

$$P(x_{1:t}, m)$$

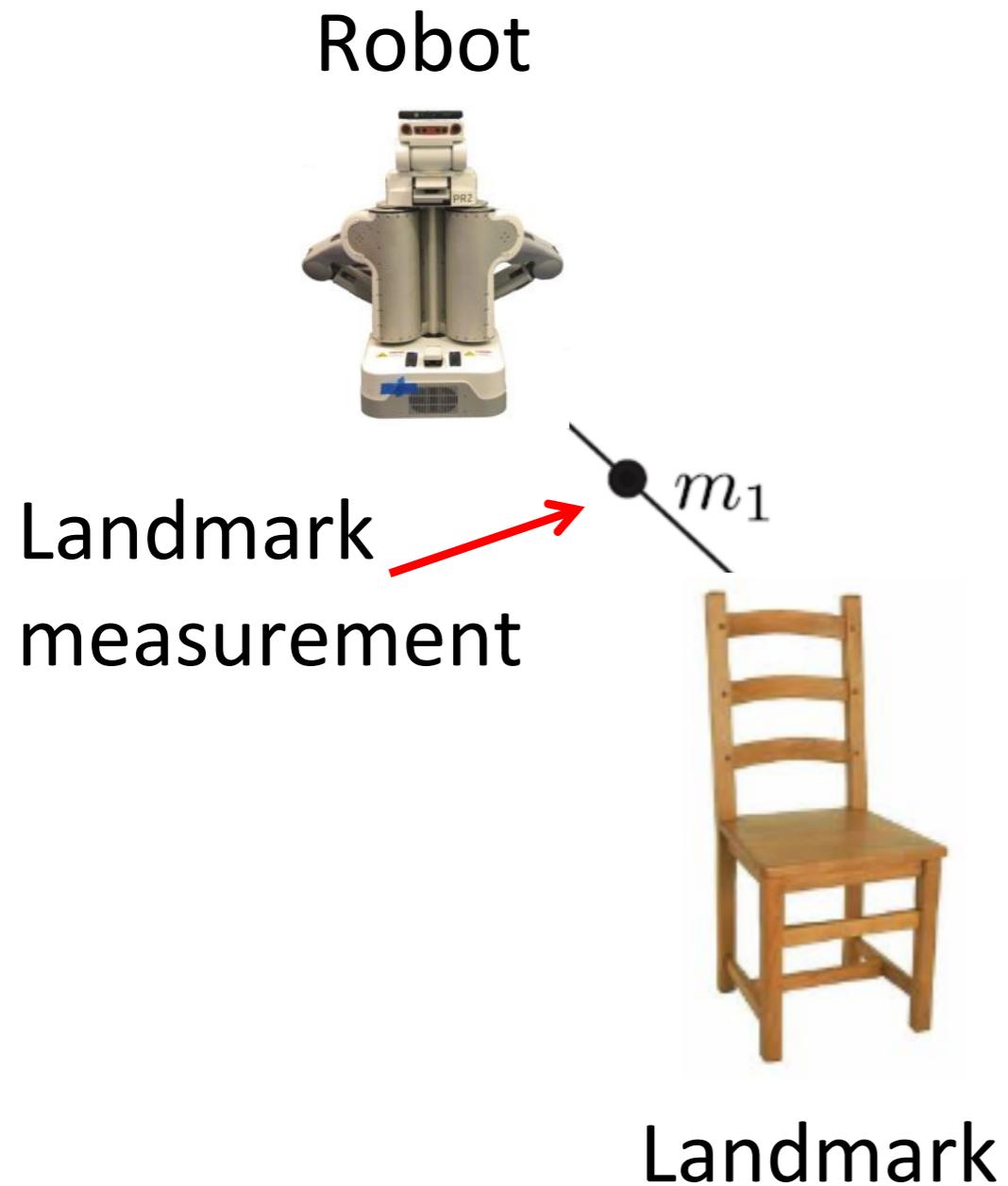
Or are does a maximum a posteriori estimate suffice?

$$\hat{x}_{1:t}, \hat{m} = \arg \max_{x_{1:t}, m} P(x_{1:t}, m)$$

SLAM as a pure optimization problem

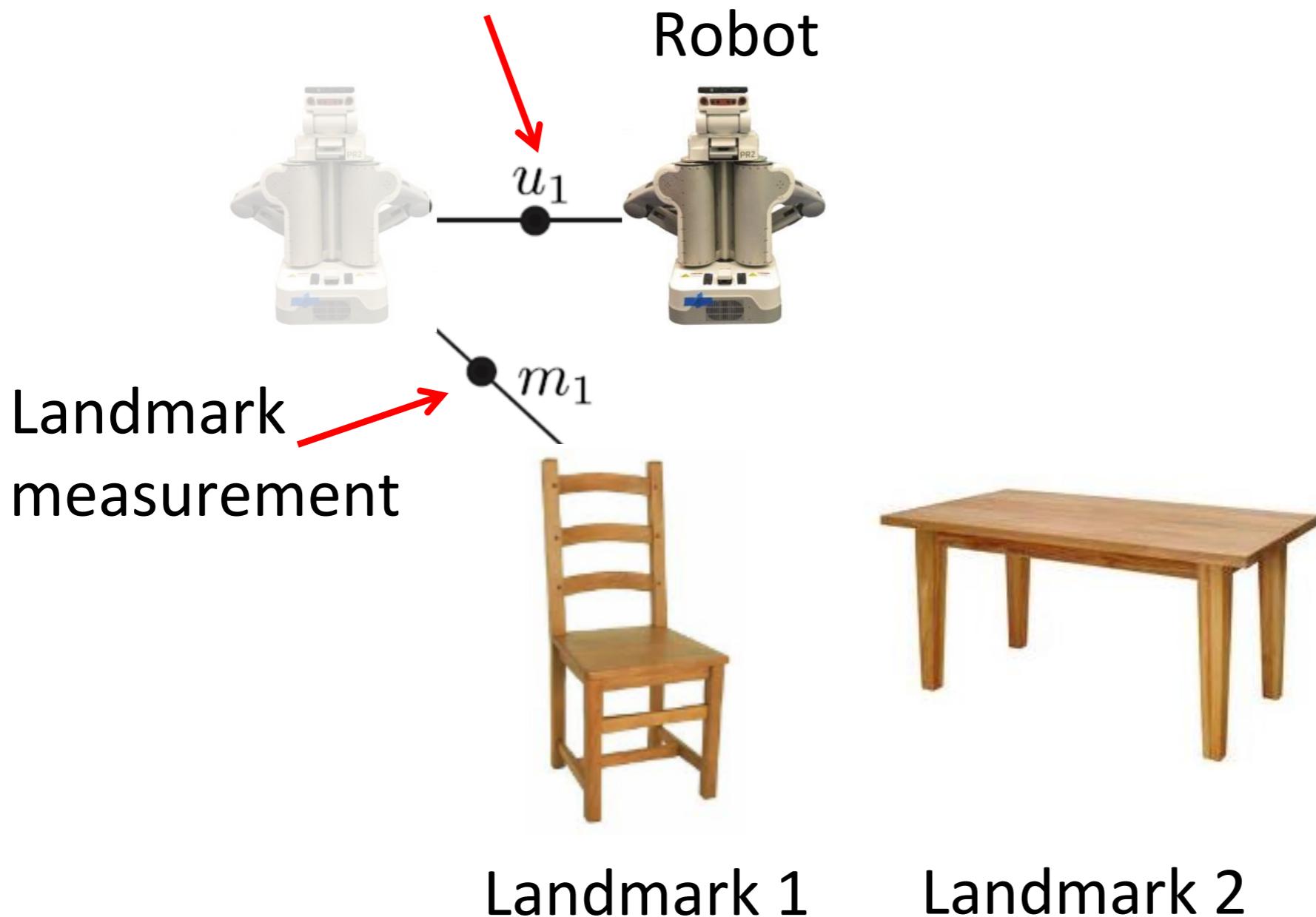


SLAM as a set of relationships



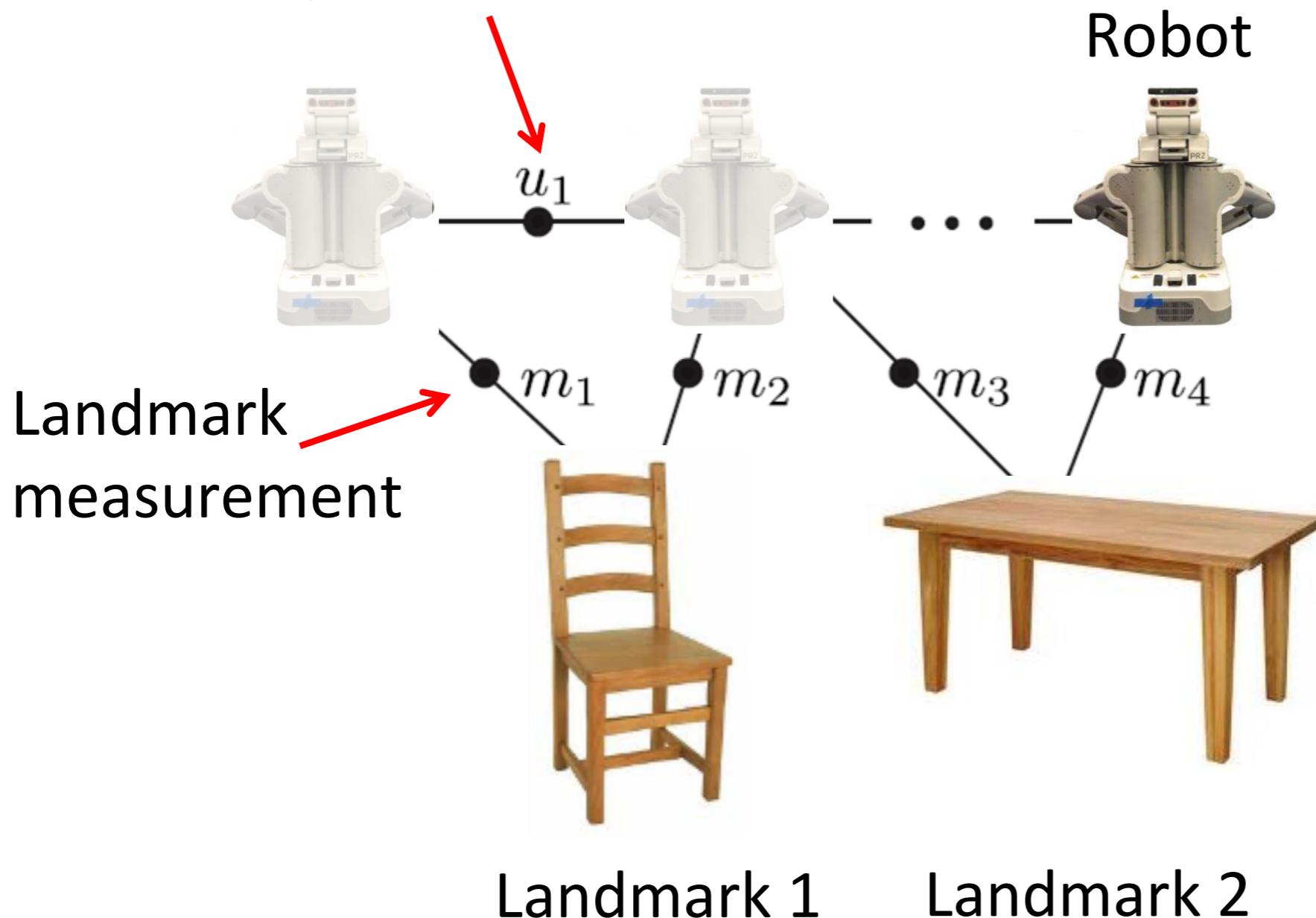
SLAM as a set of relationships

Odometry measurement



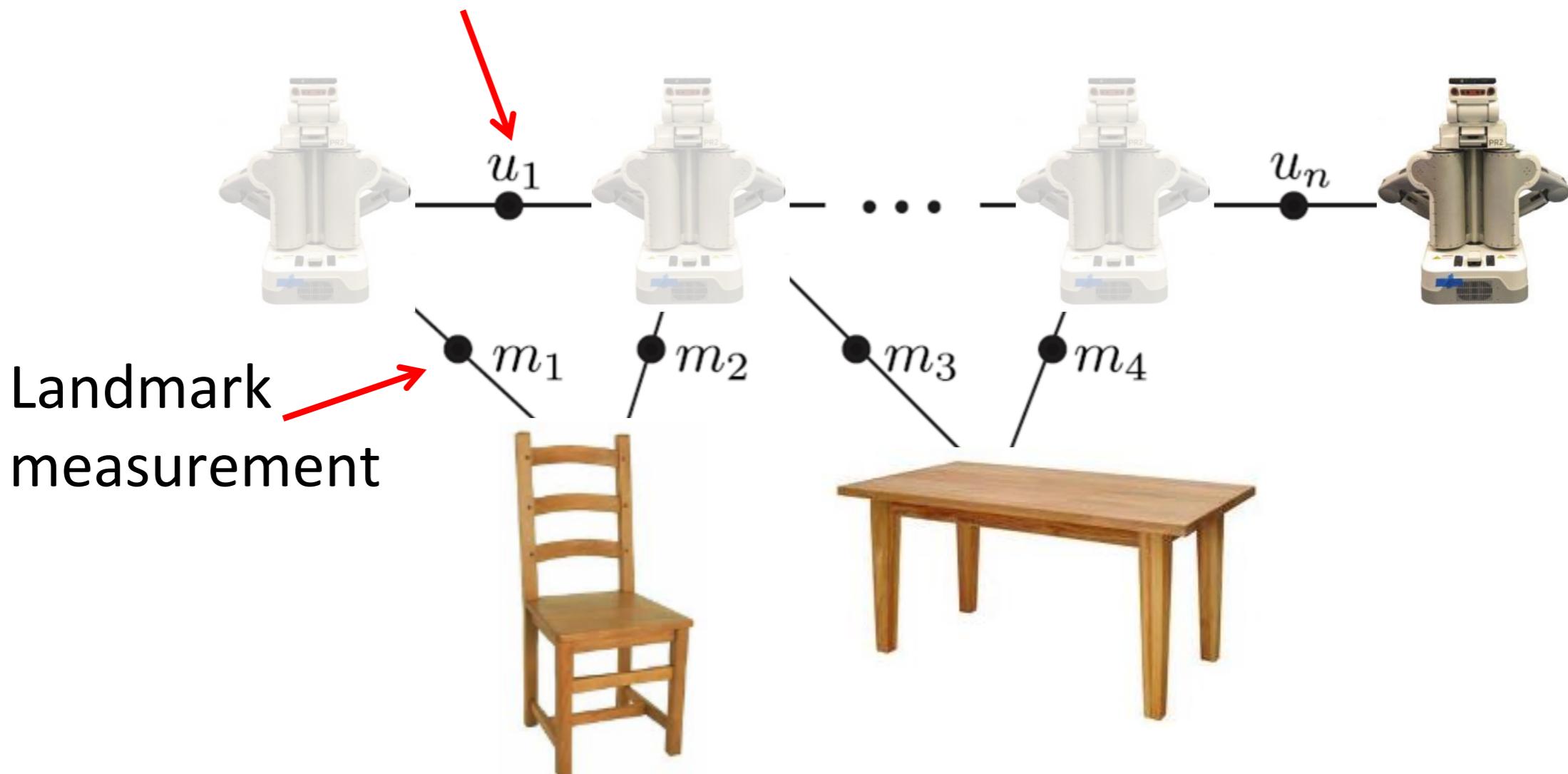
SLAM as a set of relationships

Odometry measurement



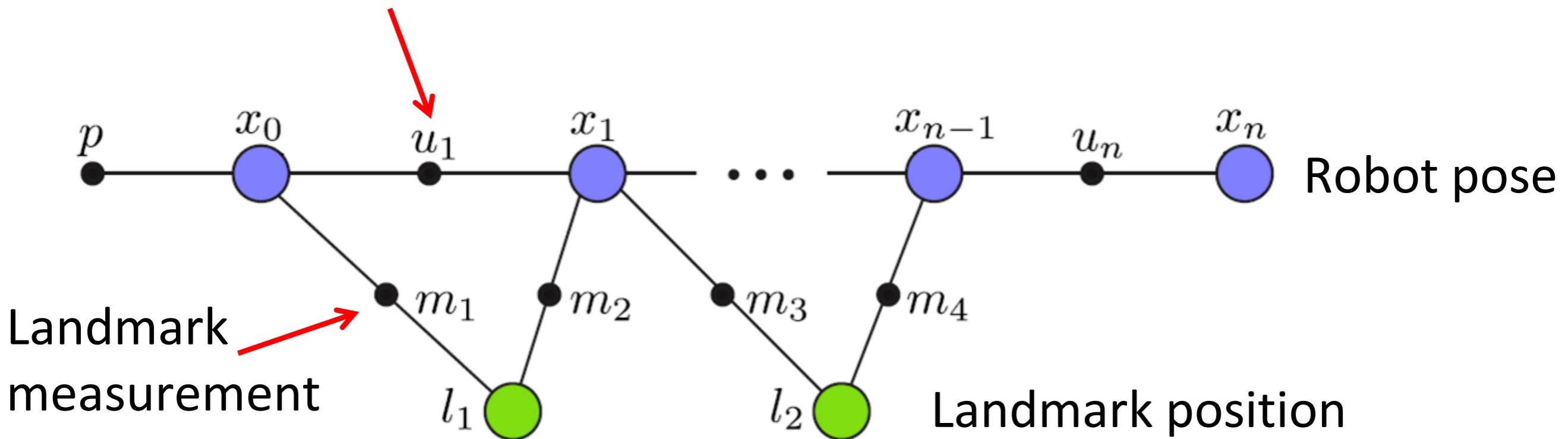
SLAM as a set of relationships

Odometry measurement

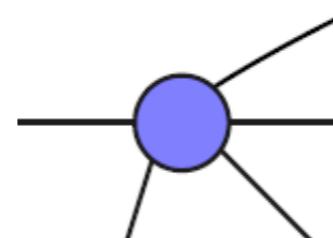
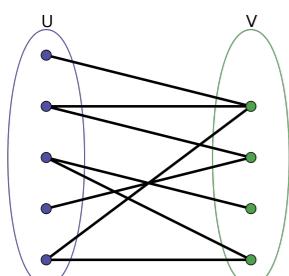


Factor Graph representation of SLAM

Odometry measurement



Bipartite graph with *variable nodes* and *factor nodes*



Courtesy M.Kaess

Factor Graph representation of SLAM

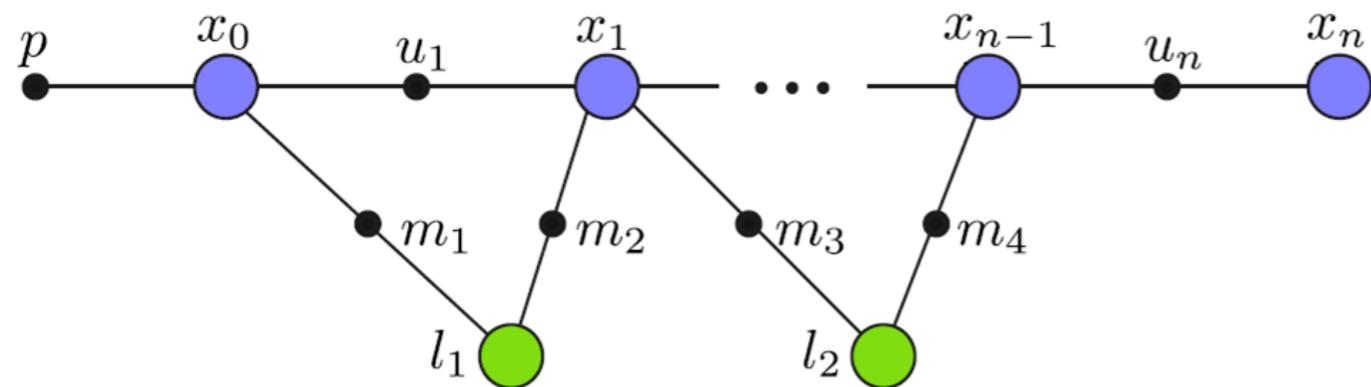
The variables in the optimization are poses of robot at all time steps and all landmarks

$$\theta = [x_1, x_2, \dots, x_n, l_1, \dots, l_m]$$

Factorization of probability density

- Conditional independence:

$$p(z_1 z_2 | \Theta) = p(z_1 | \Theta) p(z_2 | \Theta)$$



$$\operatorname{argmax}_{\Theta} \prod_{z \in Z} p(z | \Theta)$$

$$\operatorname{argmax}_{\Theta} p(p | \Theta) p(u_1 | \Theta) \cdots p(u_n | \Theta) p(m_1 | \Theta) \cdots p(m_4 | \Theta)$$

How do we solve such optimization?

Large scale
sparse
non-linear
least squares
optimization

+

incrementally
growing
factors

Option 1: Using sparse matrix algebra

(Kaess et al. 2008)

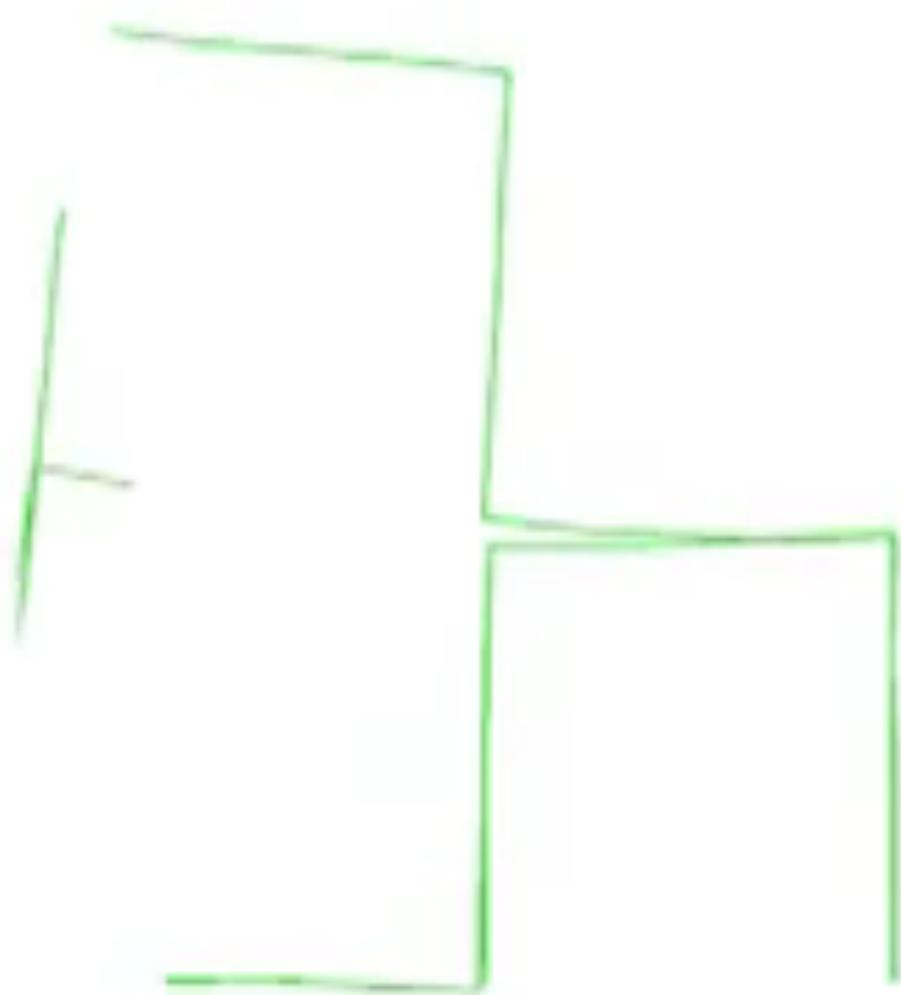
Option 2: Using probabilistic graphical models

(Kaess et al. 2011)

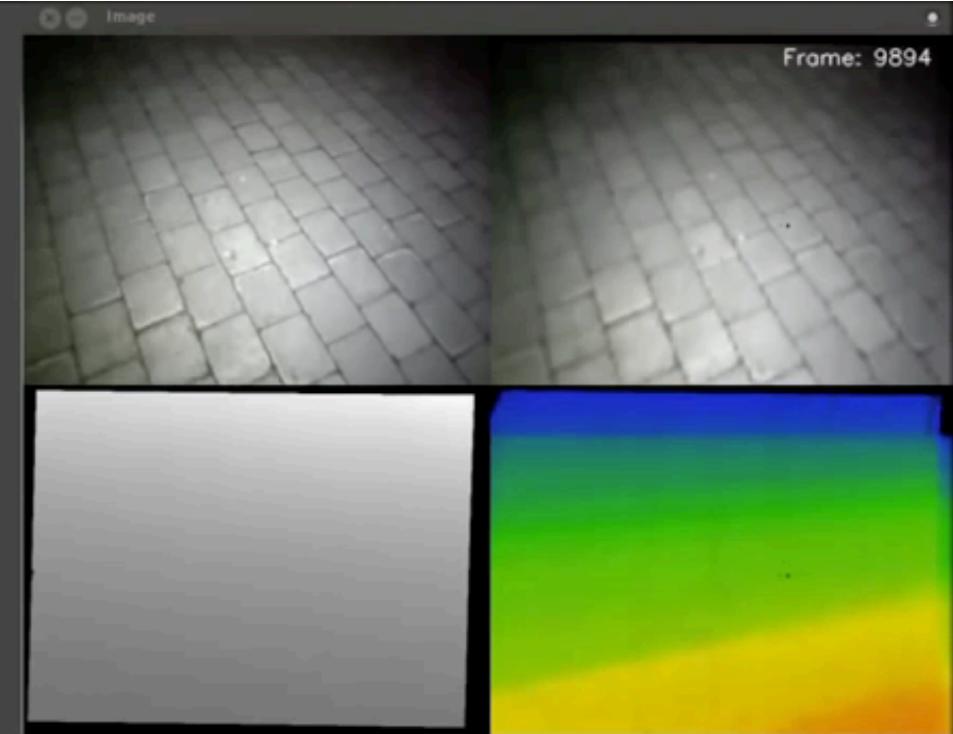


Bottom: Color coded trajectory.
Green corresponds to a low number
of variables, red to a high number.

Top: The Bayes tree.
The parts affected by the new
variables are colored in red.



Application: Kintinuous 2.0 (Whelan et al.)



32x