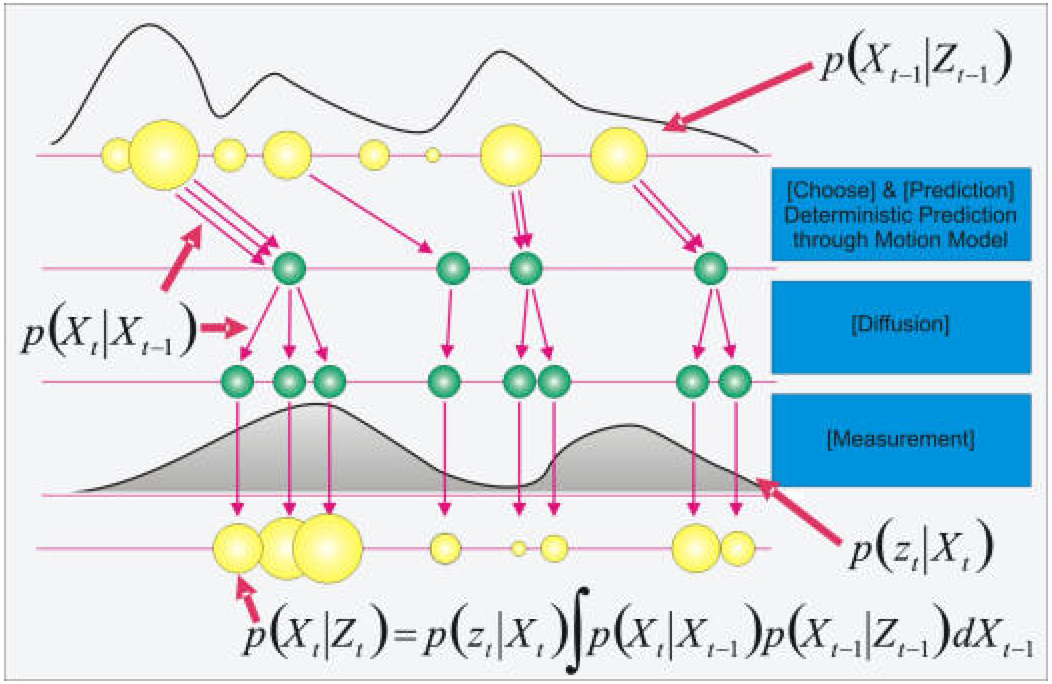
CS 4495 Computer Vision
Tracking 2: Particle Filters

Aaron Bobick
School of Interactive Computing



Administrivia

- PS5 due on Sunday Nov 10, 11:55pm
- PS6 out Thurs Nov 14, due Nov 24th
- EXAM: Tues before Thanksgiving, Nov 26th Covers concepts and basics
- Problem set resubmission policy:
  - Full questions only
  - You get 50% credit to replace whatever you got last time on that question.
  - Must be submitted by: DEC 1. NO EXCEPTIONS.

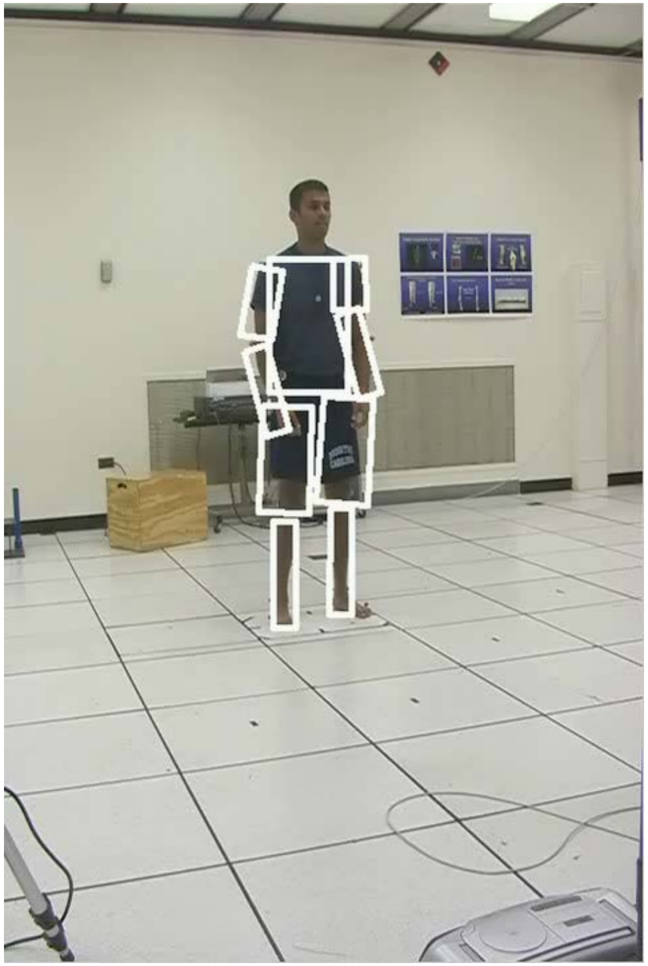
First some Matlab flow…

Tracking

- Slides still “adapted” from Kristen Grauman, Deva Ramanan, but mostly from Svetlana Lazebnik
- And now more adaptations from Sebastian Thrun, Dieter Fox and someone who did great particle filter illustrations but whose name is lost to web thievery….

Recall: Some examples

- http://www.youtube.com/watch?v=InqV34BcheM



Detection vs. tracking

t=1

t=2

…

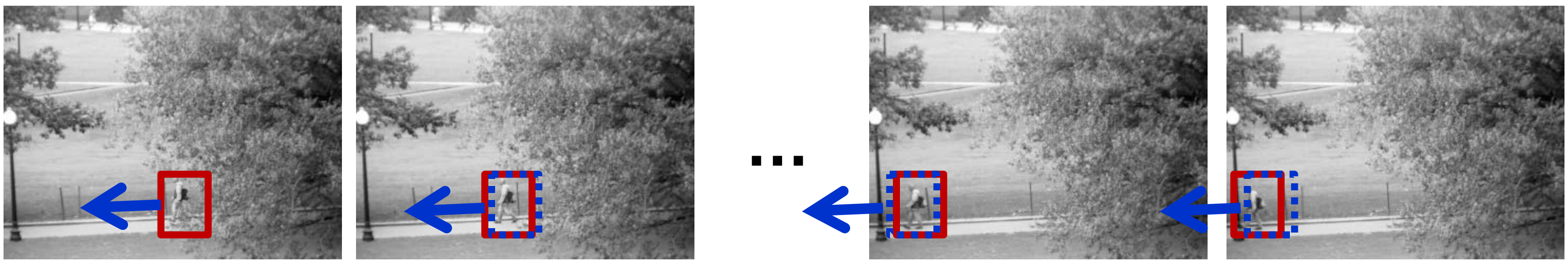
t=20

t=21

Detection vs. tracking

Detection: We detect the object independently in each frame and can record its position over time, e.g., based on blob's centroid or detection window coordinates

Detection vs. tracking



Tracking with *dynamics*: We use image measurements to estimate position of object, but also incorporate position predicted by dynamics, i.e., our expectation of object's motion pattern.

Tracking with dynamics

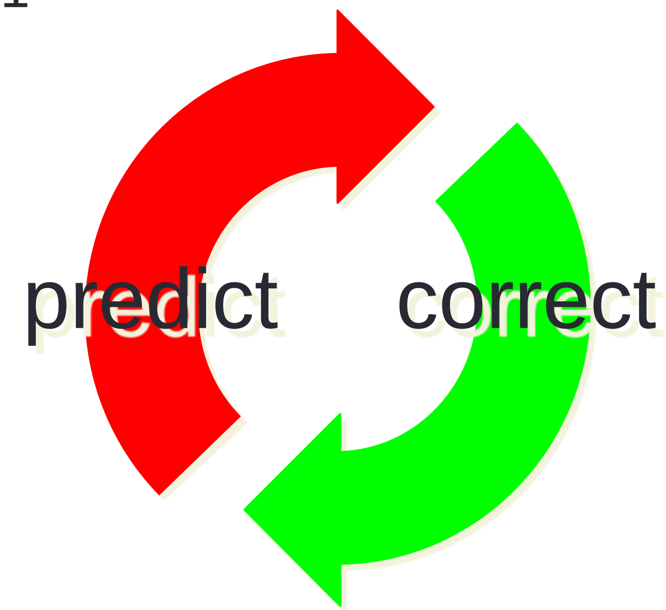
- Use model of expected motion to predict where objects will occur in next frame, even before seeing the image.

- **Intent**:
  - Do less work looking for the object, restrict the search.
  - Get improved estimates since measurement noise is tempered by smoothness, dynamics priors.

- **Assumption**: continuous motion patterns:
  - Camera is not moving instantly to new viewpoint
  - Objects do not disappear and reappear in different places in the scene
  - Gradual change in pose between camera and scene

Tracking as induction

- Base case:
  - Assume we have initial prior that predicts state in absence of any evidence: $P(X_0)$
  - At the first frame, *correct* this given the value of $Y_0=y_0$
- Given corrected estimate for frame t:
  - Predict for frame t+1
  - Correct for frame t+1



Steps of tracking

- Prediction: What is the next state of the object given past measurements?

$$P\left(X_t \middle| Y_0 = y_0, \ldots, Y_{t-1} = y_{t-1}\right)$$

- Correction: Compute an updated estimate of the state from prediction and measurements

$$P\left(X_t \middle| Y_0 = y_0, \ldots, Y_{t-1} = y_{t-1}, \boxed{Y_t = y_t}\right)$$

- Tracking can be seen as the process of propagating the ***posterior*** distribution of state given measurements across time

Simplifying assumptions

- Only the immediate past matters

$$P\left(X_t \middle| X_0, \ldots, X_{t-1}\right) = \boxed{P\left(X_t \middle| X_{t-1}\right)}$$

dynamics model

- Measurements depend only on the current state

$$P\left(Y_t \middle| X_0, Y_0 \ldots, X_{t-1}, Y_{t-1}, X_t\right) = \boxed{P\left(Y_t \middle| X_t\right)}$$

observation model

Linear Dynamic Models

- Dynamics model: state undergoes linear tranformation plus Gaussian noise
- $$X_t \sim N\left(D_t x_{t-1}, \Sigma_{d_t}\right)$$
- Observation model: measurement is linearly transformed state plus Gaussian noise

$$Y_t \sim N\left(M_t x_t, \Sigma_{m_t}\right)$$

Example: Constant velocity (1D)

- State vector is position and velocity

$$x_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix} \qquad \begin{aligned} p_t &= p_{t-1} + (\Delta t) v_{t-1} + \varepsilon \\ v_t &= v_{t-1} + \xi \end{aligned}$$

(greek letters denote noise terms)

$$x_t = D_t x_{t-1} + noise = \begin{bmatrix} 1 & \Delta t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \end{bmatrix} + noise$$

- Measurement is position only

$$y_t = M x_t + noise = \begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \end{bmatrix} + noise$$

Example: Constant acceleration (1D)

- State vector is position, velocity, and acceleration

$$x_t = \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} \qquad \begin{aligned} p_t &= p_{t-1} + (\Delta t) v_{t-1} + \varepsilon \\ v_t &= v_{t-1} + (\Delta t) a_{t-1} + \xi \\ a_t &= a_{t-1} + \zeta \end{aligned}$$

(greek letters denote noise terms)

$$x_t = D_t x_{t-1} + noise = \begin{bmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_{t-1} \\ v_{t-1} \\ a_{t-1} \end{bmatrix} + noise$$

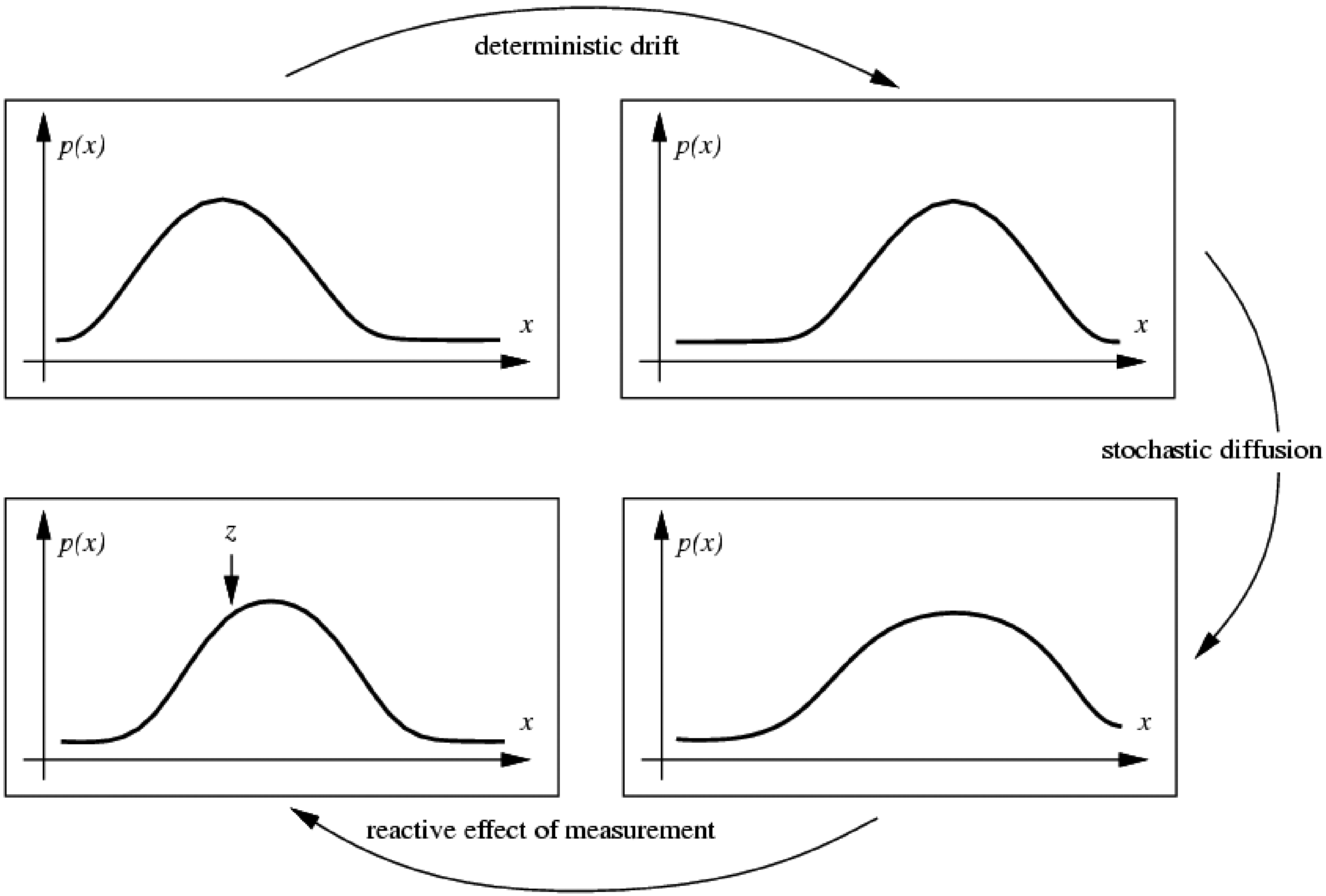
- Measurement is position only

$$y_t = M x_t + noise = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_t \\ v_t \\ a_t \end{bmatrix} + noise$$

The Kalman filter

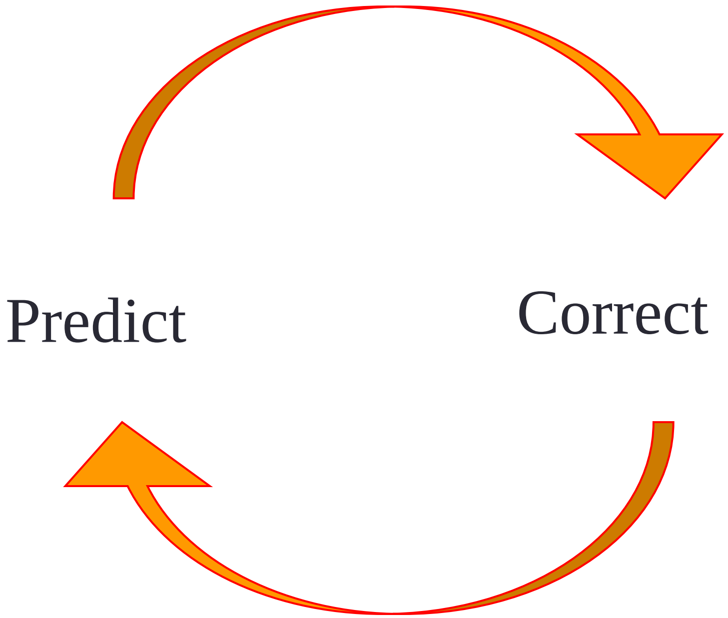
- Method for tracking linear dynamical models in Gaussian noise
- The predicted/corrected state distributions are Gaussian
  - You only need to maintain the mean and covariance
  - The calculations are easy (all the integrals can be done in closed form)

Propagation of Gaussian densities



The Kalman Filter: 1D state

Make measurement



Given corrected state from previous time step and all the measurements up to the current one, predict the distribution over the current step

$$P\left(X_t \middle| y_0,\ldots,y_{t-1}\right)$$

Mean and std. dev. of predicted state:

$$\mu_t^-,\sigma_t^-$$

Time advances (from *t*–1 to *t*)

Given prediction of state and current measurement, update prediction of state

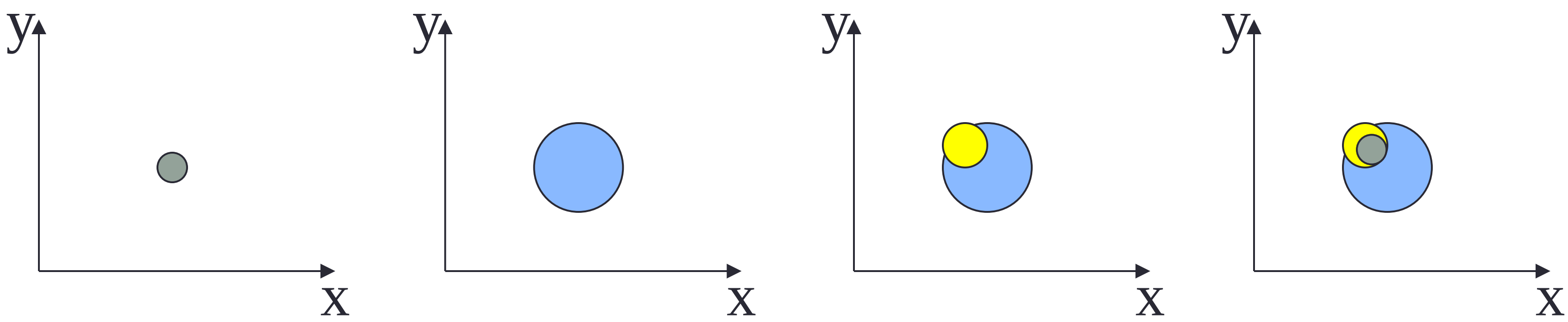
$$P\left(X_t \middle| y_0,\ldots,y_t\right)$$

Mean and std. dev. of corrected state:

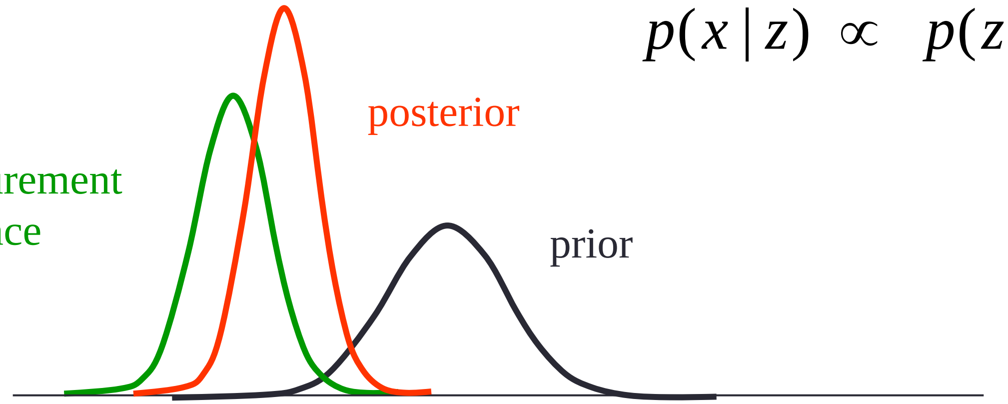
$$\mu_t^+,\sigma_t^+$$

Tracking with KFs: Gaussians!

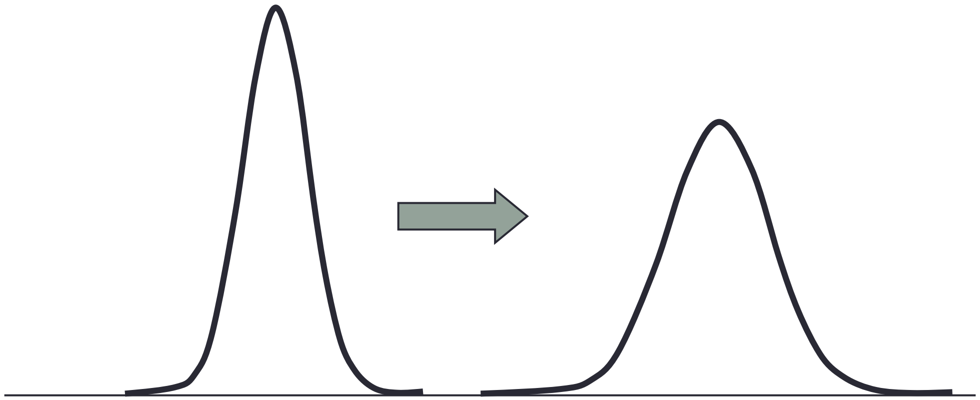
initial estimte | prediction | measurement | update



Kalman Filters



$$p(x \mid z) \propto p(z \mid x)\, p(x)$$



$$p(x') = \int p(x' \mid x)\; p(x)\; dx$$

1D Kalman filter: Prediction

- Linear dynamic model defines predicted state evolution, with noise

$$X_t \sim N\left(dx_{t-1}, \sigma_d^2\right)$$

- Want to estimate distribution for next predicted state

$$P\left(X_t \middle| y_0, \ldots, y_{t-1}\right) = N\left(\mu_t^-, (\sigma_t^-)^2\right)$$

- Update the mean: $\mu_t^- = d\mu_{t-1}^+$
- Update the variance: $(\sigma_t^-)^2 = \sigma_d^2 + (d\sigma_{t-1}^+)^2$

1D Kalman filter: Correction

- Mapping of state to measurements: $Y_t \sim N\left(mx_t, \sigma_m^2\right)$
- Predicted state: $P\left(X_t \middle| y_0, \ldots, y_{t-1}\right) = N\left(\mu_t^-, (\sigma_t^-)^2\right)$
- Want to estimate corrected distribution

$$P\left(X_t \middle| y_0, \ldots, y_t\right) = N\left(\mu_t^+, (\sigma_t^+)^2\right)$$

- Update the mean:

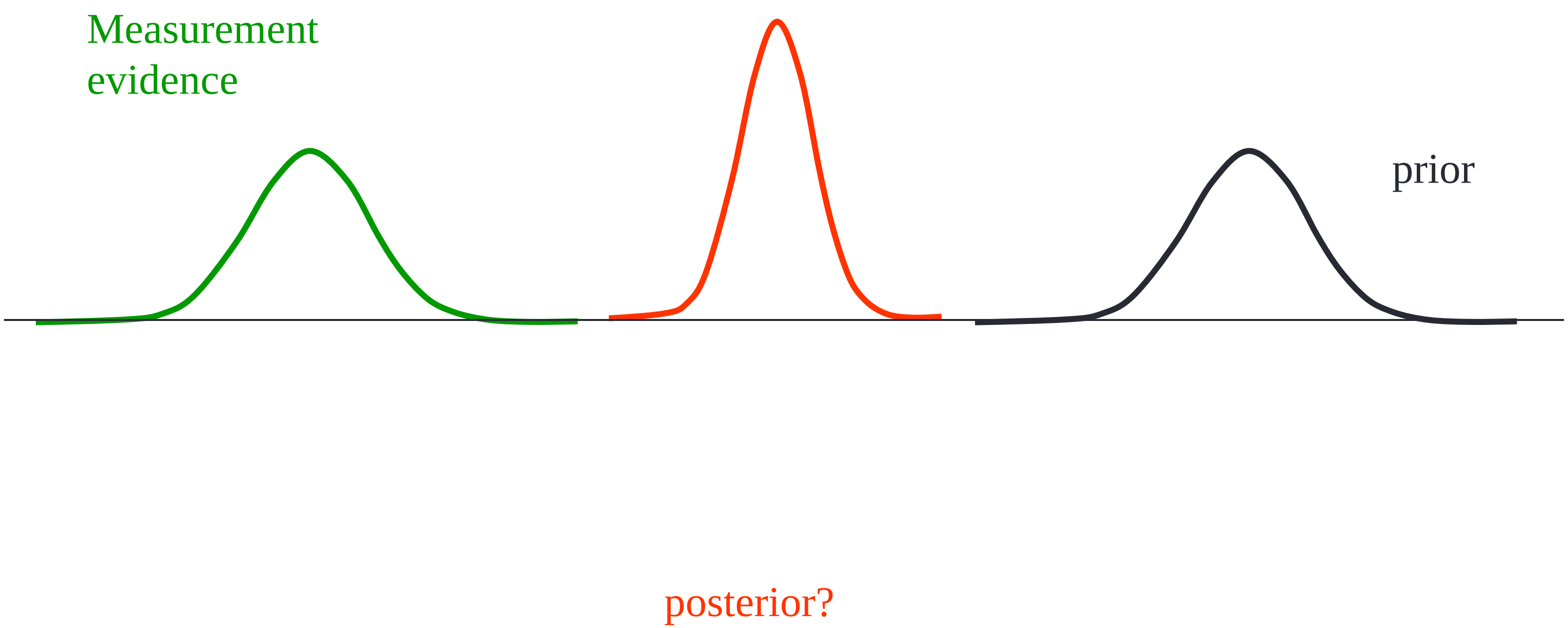
$$\mu_t^+ = \frac{\mu_t^- \sigma_m^2 + m y_t (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2}$$

- Update the variance:

$$(\sigma_t^+)^2 = \frac{\sigma_m^2 (\sigma_t^-)^2}{\sigma_m^2 + m^2 (\sigma_t^-)^2}$$

A Quiz

$$p(x \mid y) \propto p(y \mid x)\, p(x)$$



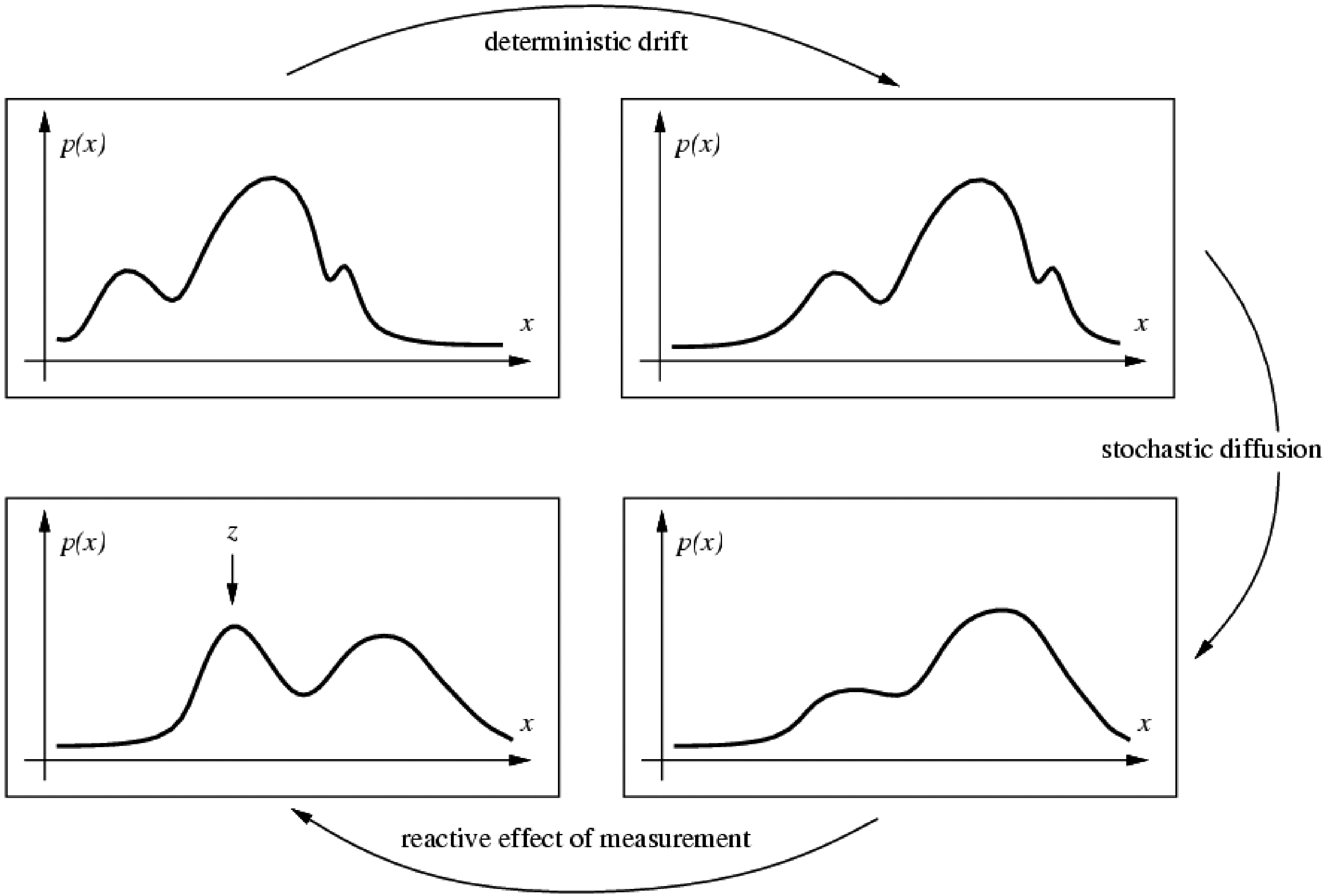
Does this agree with your intuition?

Kalman filter pros and cons

- Pros
  - Simple updates, compact and efficient
- Cons
  - Unimodal distribution, only single hypothesis
  - Restricted class of motions defined by linear model

- So what might we do if not Gaussian?
  - E.g. Object seems to be detected (noisy) in two locations in the next image that are both plausible in terms of dynamics and belief where the object was in the last image.

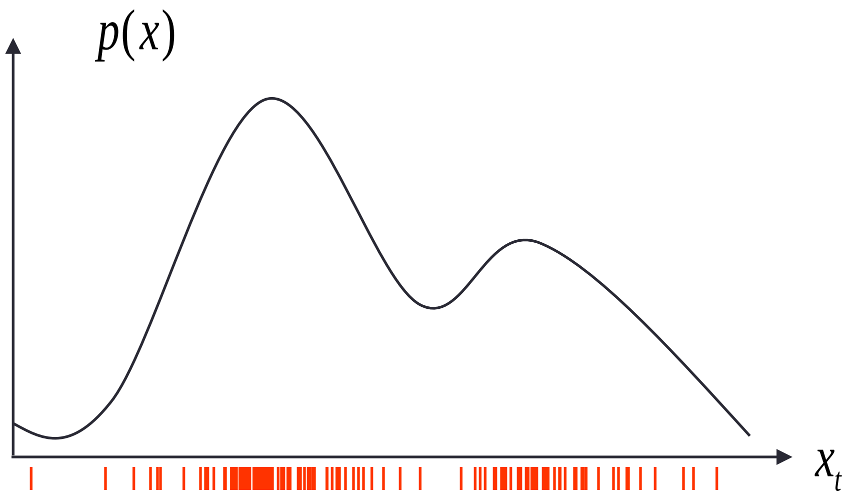
Propagation of general densities



Before we go any further…

- In particle filtering the measurements are written as z_t and not as y_t
- So we'll start seeing z's

Particle Filters: Basic Idea



set of n (weighted) particles X_t

Density is represented by both **where** the particles are and their **weight**. $p(x = x_0)$is now probability of drawing an x with value x_0.

Goal: $p(x_t \in X_t) \approx p\left(x_t \middle| z_{\{1\ldots t\}}\right)$ with equality when $n \to \infty$

A slight shift…

- For just *tracking* we had the notion of a dynamics model – and for the Kalman filter a *linear* dynamics model.
  - At each time step the prediction was based upon linear transform of state so could easily include velocity, acceleration, etc.
  - The linear form was necessary to make the math work out. To do non-linear dynamics need to linearize about the current state (remember Jacobians?).

- In general, the state transition matrix represented what was known or expected to change from t to $t+1$.

- We can generalize that notion of action or input u_t at time t.

Bayes Filters: Framework

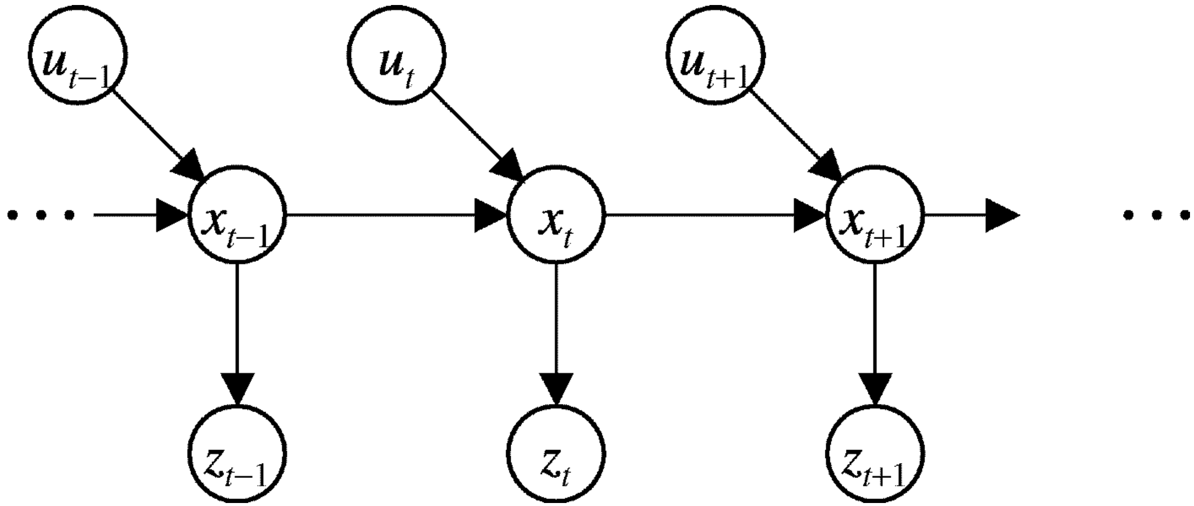
- **Given:**
  - Stream of observations *z* and action data *u:*

$$data_t = \{u_1, z_2 \ldots, u_{t-1}, z_t\}$$

  - Sensor model $p(z|x)$.
  - Action model $p(x_t|u_{\{t-1\}}, x_{\{t-1\}})$
  - Prior probability of the system state *p(x).*
- **Wanted:**
  - Estimate of the state *X* of a dynamical system.
  - The posterior of the state is also called **Belief**:

$$Bel(x_t) = P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

Graphical Model Representation



$$p(z_t \mid x_{0:t}, z_{1:t-1}, u_{1:t}) = p(z_t \mid x_t)$$
$$p(x_t \mid x_{1:t-1}, z_{1:t-1}, u_{1:t}) = p(x_t \mid x_{t-1}, u_t)$$

Underlying Assumptions

- Static world
- Independent noise

Bayes Rule reminder

Prior before measurement

$$p(x \mid z) = \frac{p(z \mid x)\,\boxed{p(x)}}{p(z)}$$

$$= \eta\, p(z \mid x)\, p(x)$$

Bayes Filters

z = observation
u = action
x = state

$$\boxed{Bel(x_t) =} P(x_t \mid u_1, z_2 \ldots, u_{t-1}, z_t)$$

Bayes
$$= \eta \, P(z_t \mid x_t, u_1, z_2, \ldots, u_{t-1}) \, P(x_t \mid u_1, z_2, \ldots, u_{t-1})$$

Markov
$$= \eta \, P(z_t \mid x_t) \, P(x_t \mid u_1, z_2, \ldots, u_{t-1})$$

Total Probability of "Prior"
$$= \eta \, P(z_t \mid x_t) \int P(x_t \mid u_1, z_2, \ldots, u_{t-1}, x_{t-1}) \cdot P(x_{t-1} \mid u_1, z_2, \ldots, u_{t-1}) \, dx_{t-1}$$

Markov
$$= \eta \, P(z_t \mid x_t) \int P(x_t \mid u_{t-1}, x_{t-1}) \, P(x_{t-1} \mid u_1, z_2, \ldots, u_{t-1}) \, dx_{t-1}$$

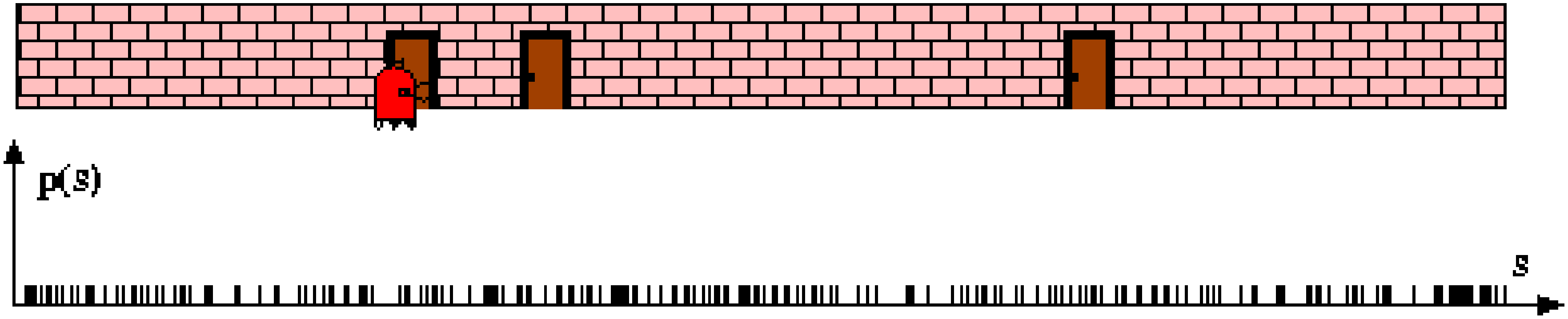
$$= \boxed{\eta \, P(z_t \mid x_t) \underbrace{\int P(x_t \mid u_{t-1}, x_{t-1}) \, Bel(x_{t-1}) \, dx_{t-1}}_{\text{Prediction } \textit{before} \text{ taking measurement}}}$$

To illustrate…

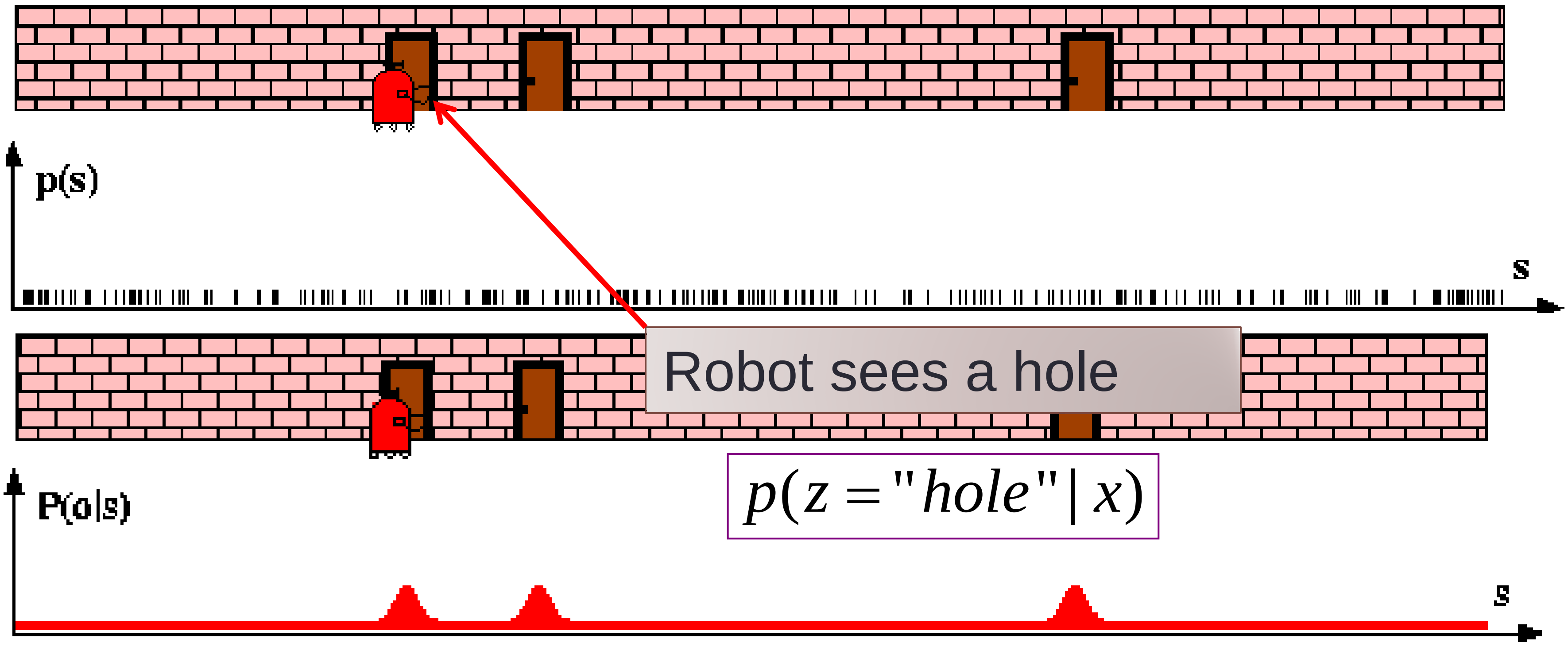
- Assume a simple robot with a noisy range sensor looking to its side…

Particle Filters

When density is not easily represented analytically (ie a couple of Gaussians), represent by a set of (possibly weighted) *samples*

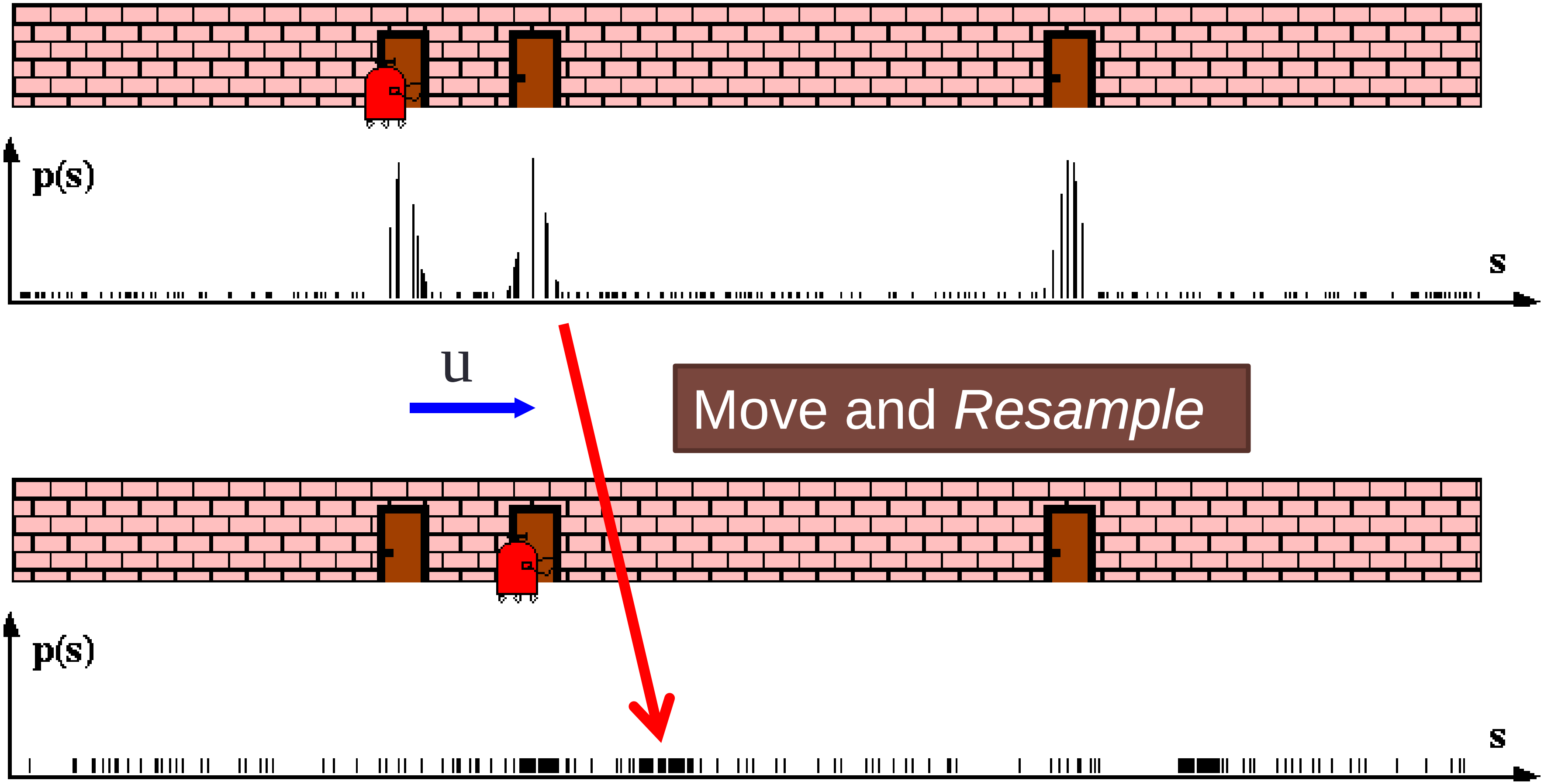


Sensor Information (aka Importance Sampling)

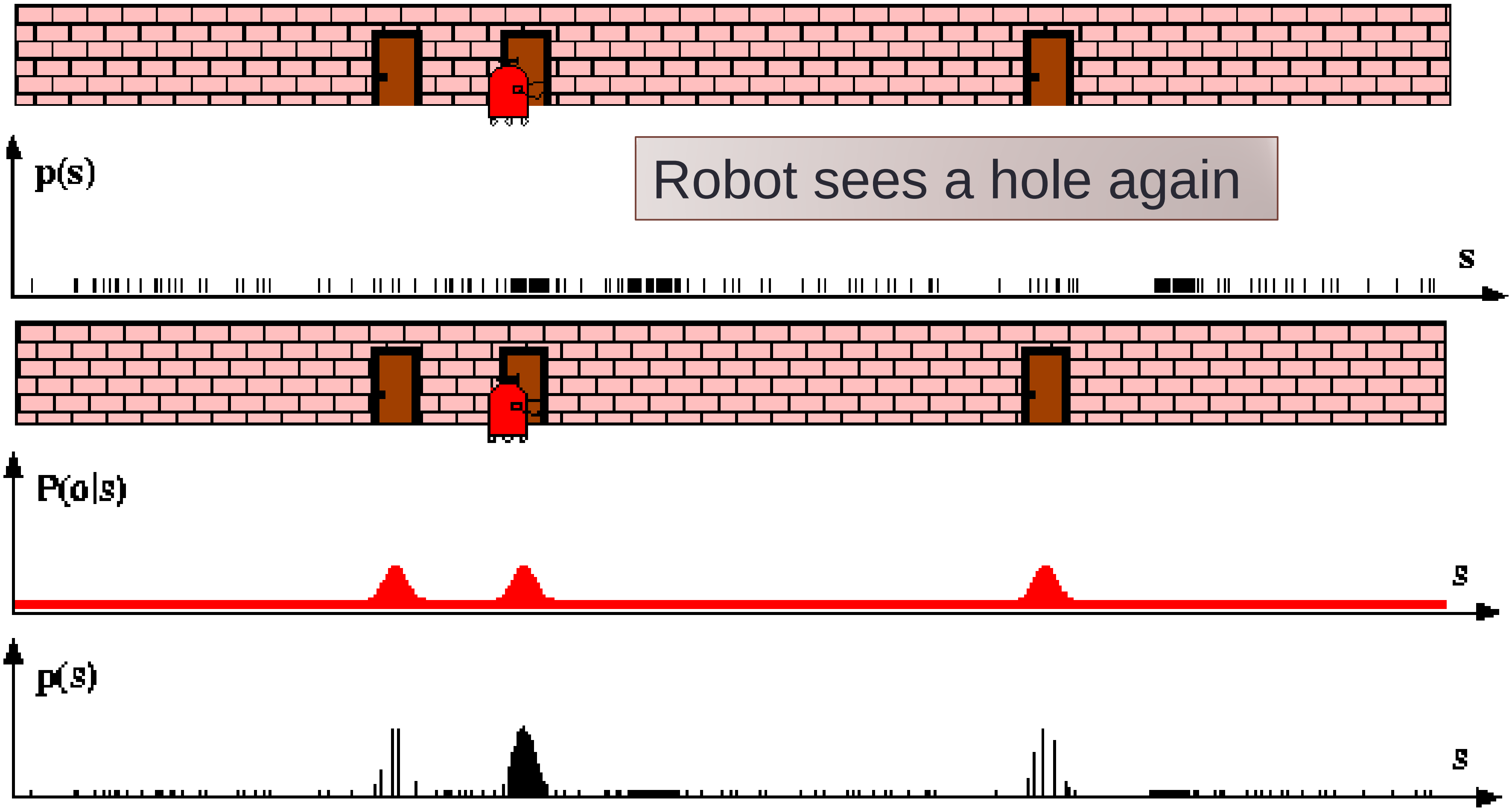


Robot Motion

$$Bel^{-}(x) \quad \leftarrow \quad \int p(x \mid u, x')\, Bel(x')\, \mathrm{d}x'$$

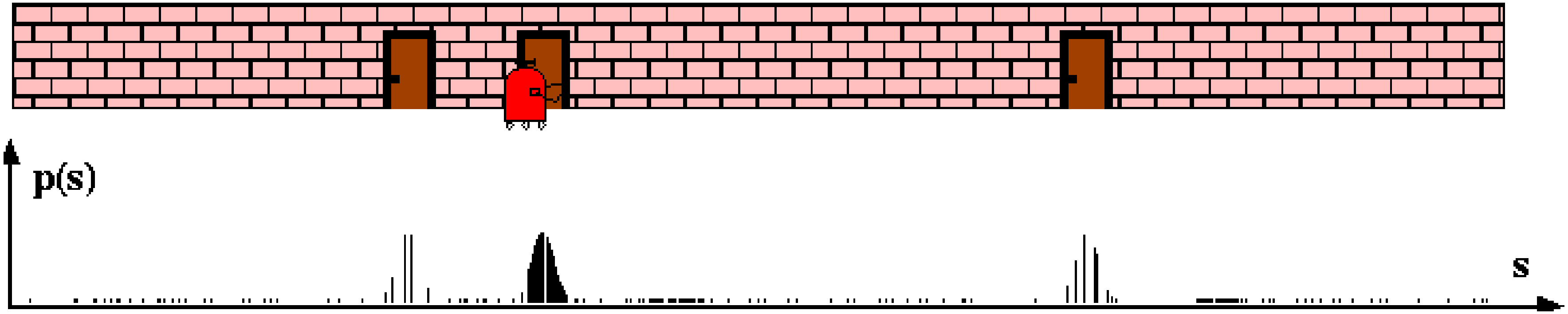


Next Sensor Reading

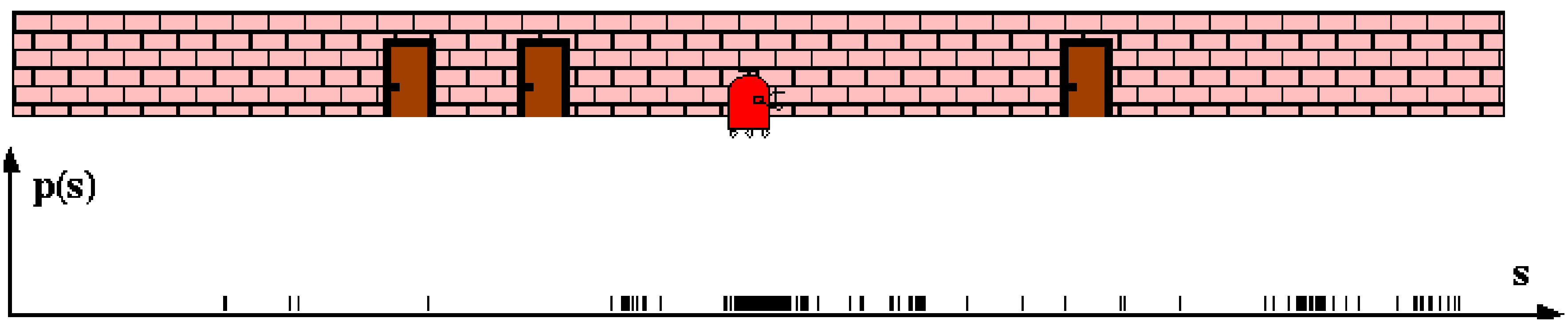


Robot sees a hole again

Robot Moves Again



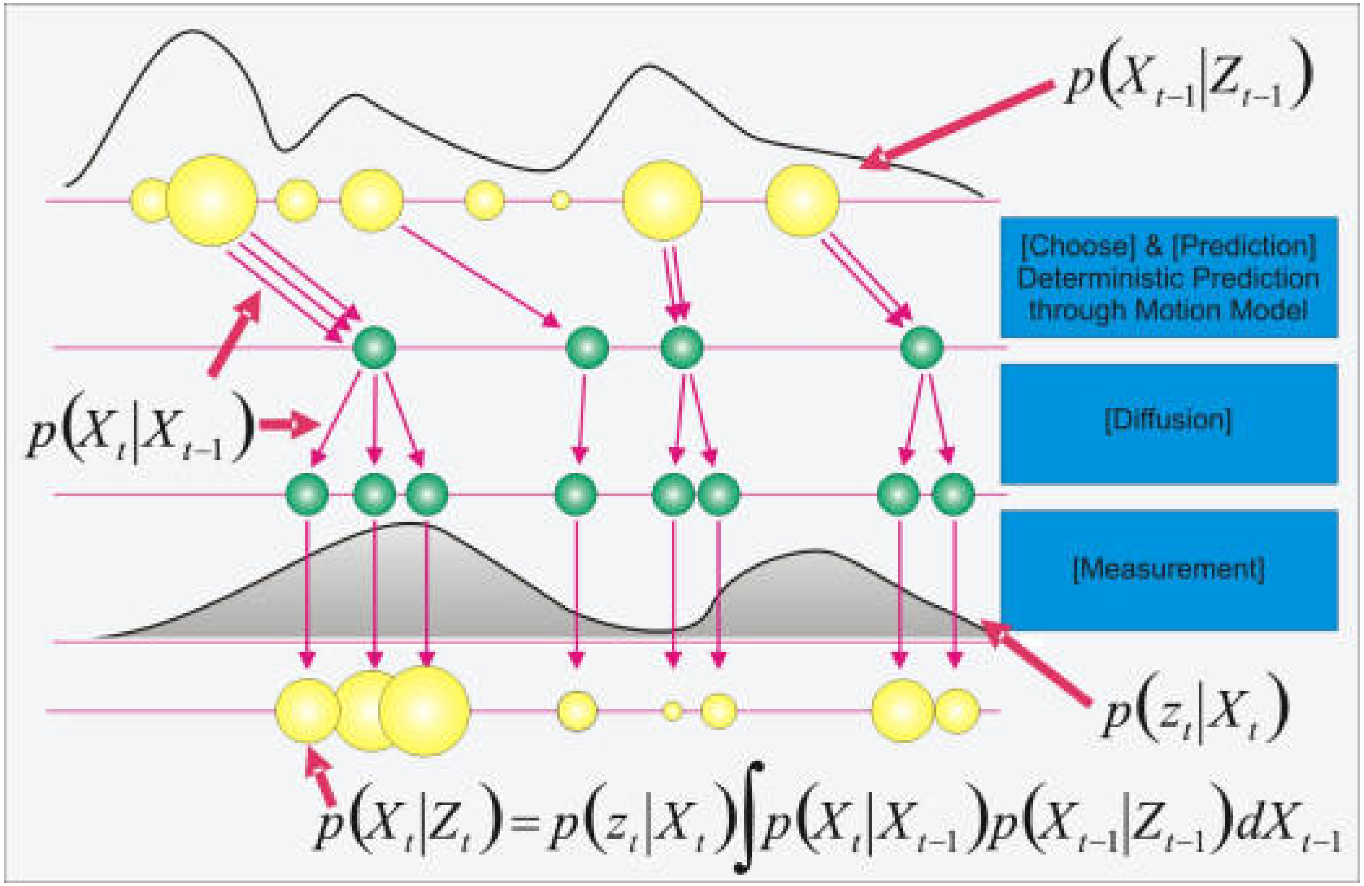
Move and *Resample again*



Particle Filter Algorithm (Sequential Importance Resampling)

1. Algorithm **particle_filter** $\{S_{t-1} = < x_{t-1}^{j}, w_{t-1}^{j} >, u_t, z_t\}$
2. $S_t = \emptyset, \quad \eta = 0$
3. **For** $i = 1 \ldots n$ ***Resample (generate i new samples)***
4. Sample index *j(i)* from the discrete distribution given by w_{t-1}
5. Sample x_t^i from $p(x_t \mid x_{t-1}, u_t)$ using $x_{t-1}^{j(i)}$ and u_t ***Control***
6. $w_t^i = p(z_t \mid x_t^i)$ ***Compute importance weight (or reweight)***
7. $\eta = \eta + w_t^i$ ***Update normalization factor***
8. $S_t = S_t \cup \{< x_t^i, w_t^i >\}$ ***Insert***
9. **For** $i = 1 \ldots n$
10. $w_t^i = w_t^i / \eta$ ***Normalize weights***

Graphical steps particle filtering

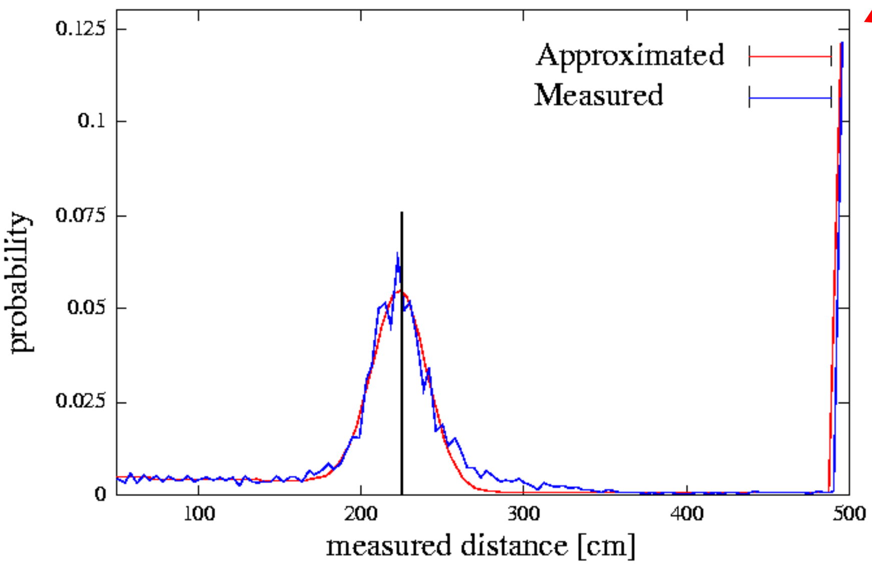


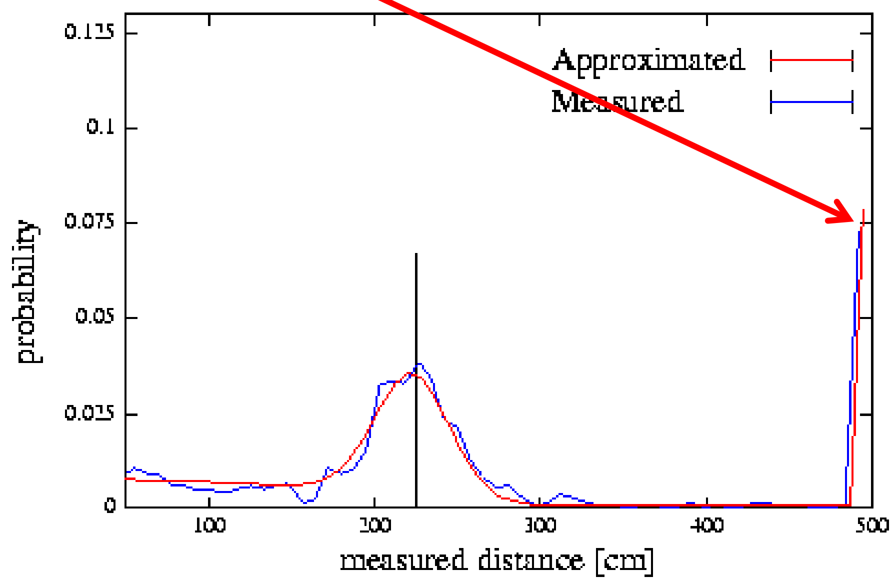
A real robot sensing problem

- Assume a robot **knows** a 3D map of its world
- It has a noisy depth sensor or sensors whose sensing uncertainty is **known**.
- It moves from frame to frame.
- How well can it know where it is *(x,y,θ) ?*

Proximity Sensor Model

No return!

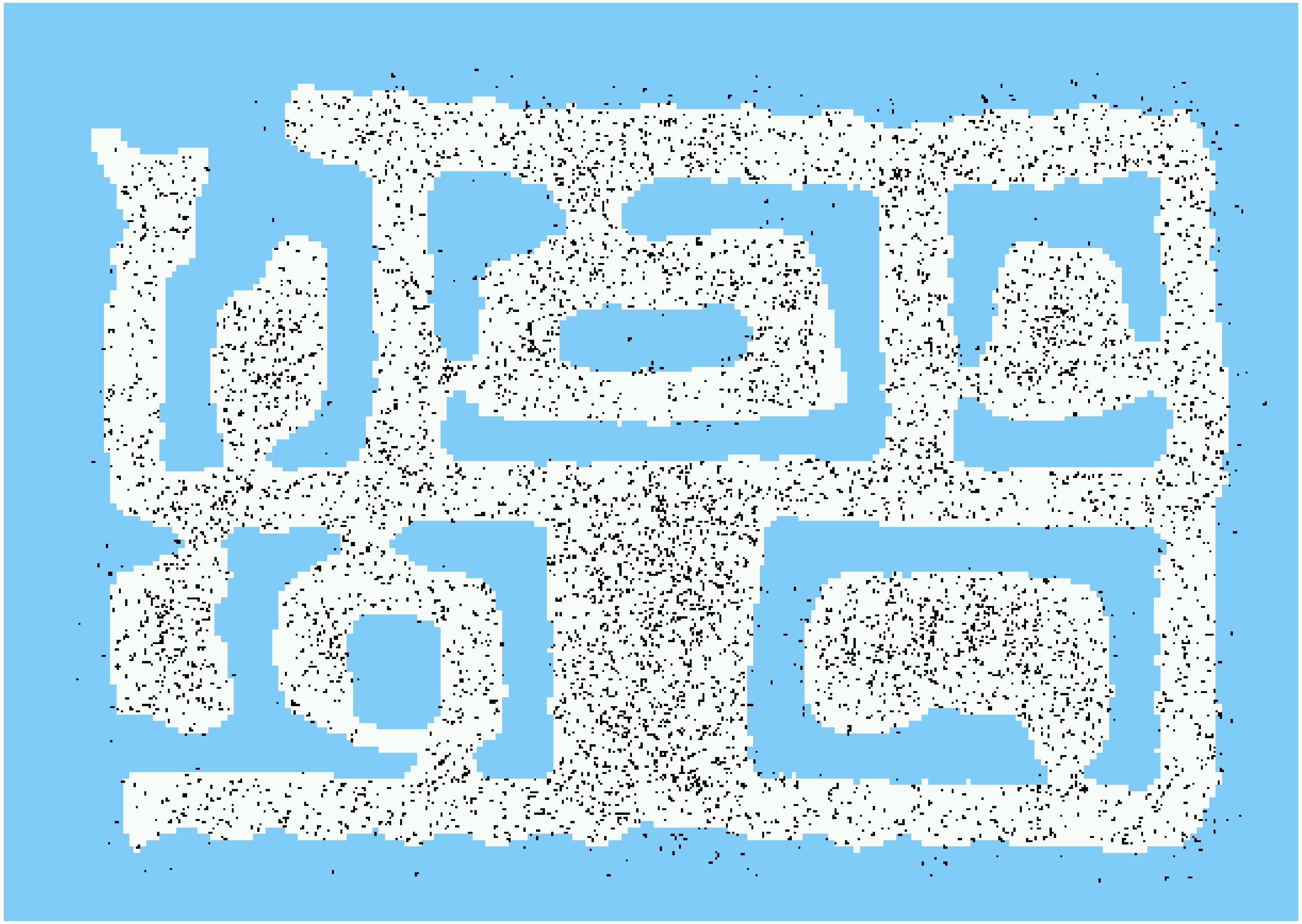




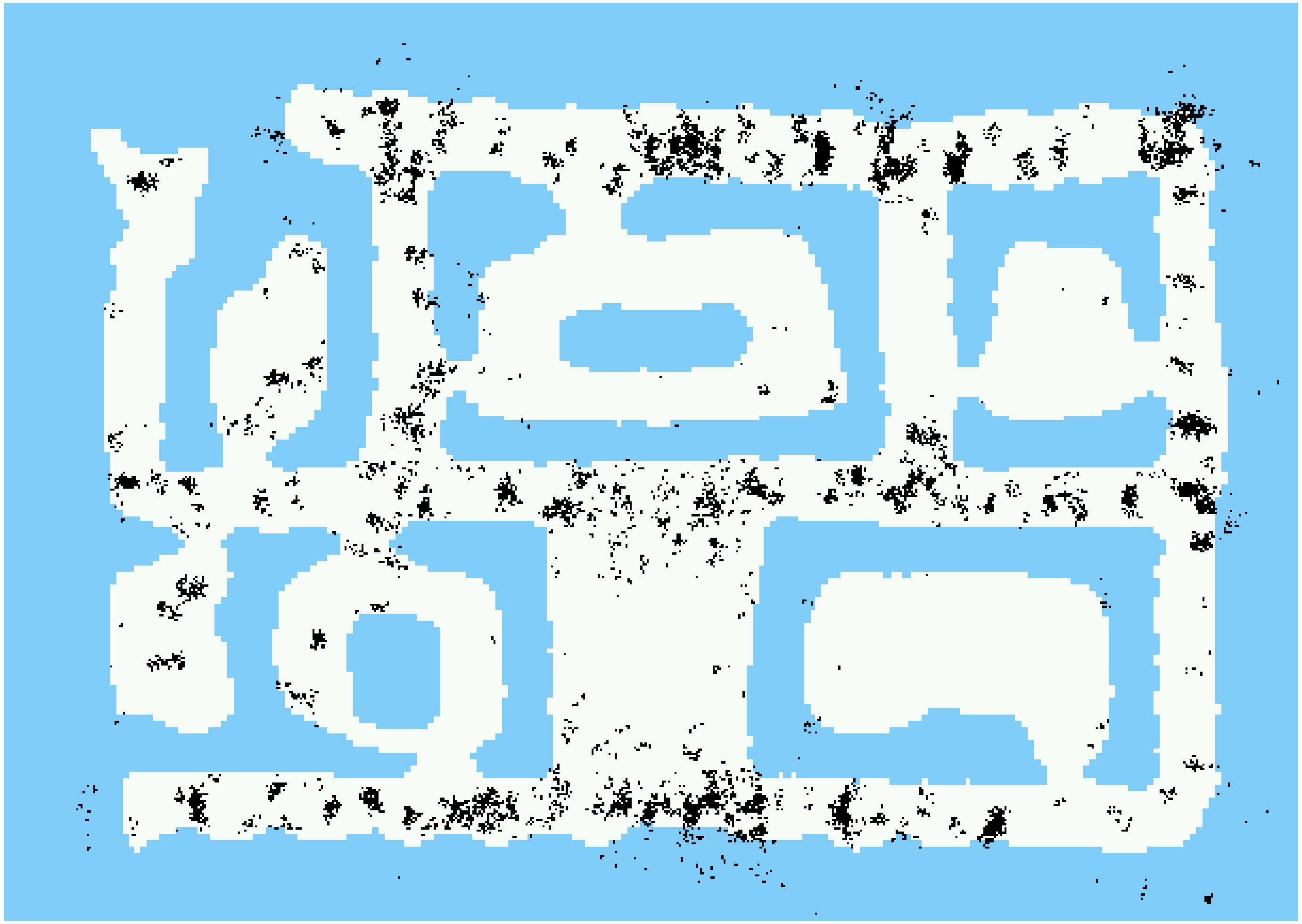
Laser sensor

Sonar sensor

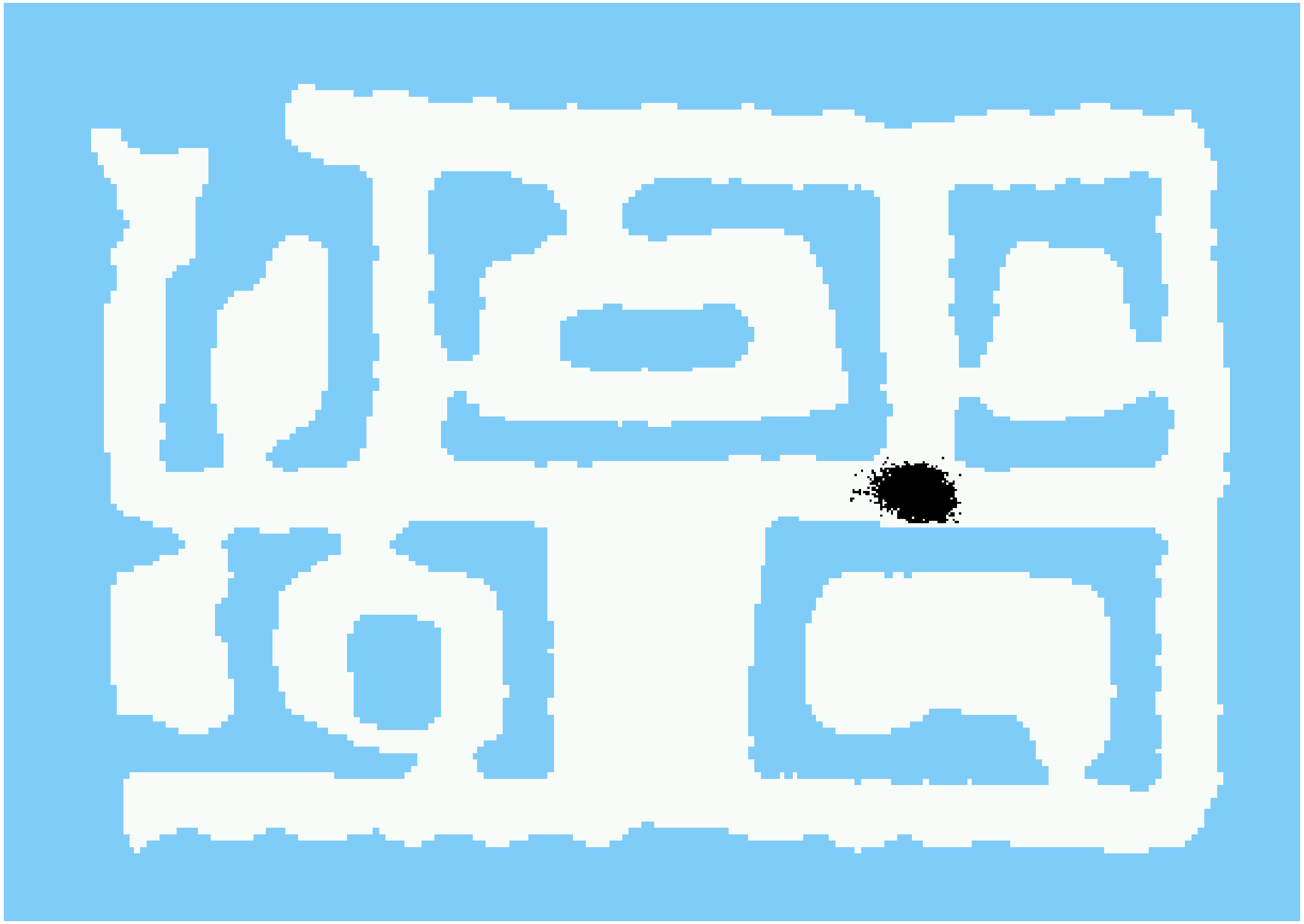
Sonar Example: Initial Distribution



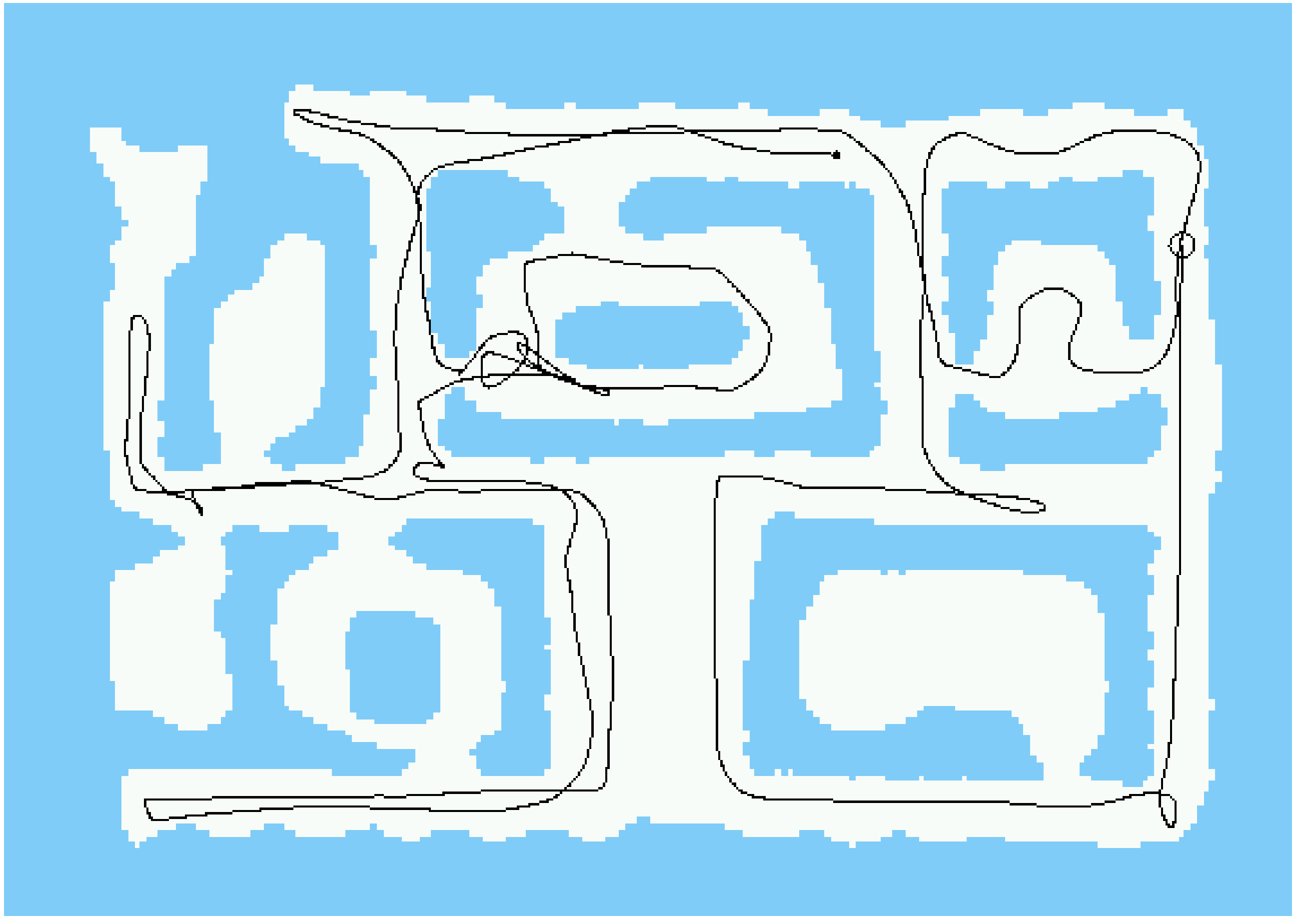
After Incorporating Ten Ultrasound Scans



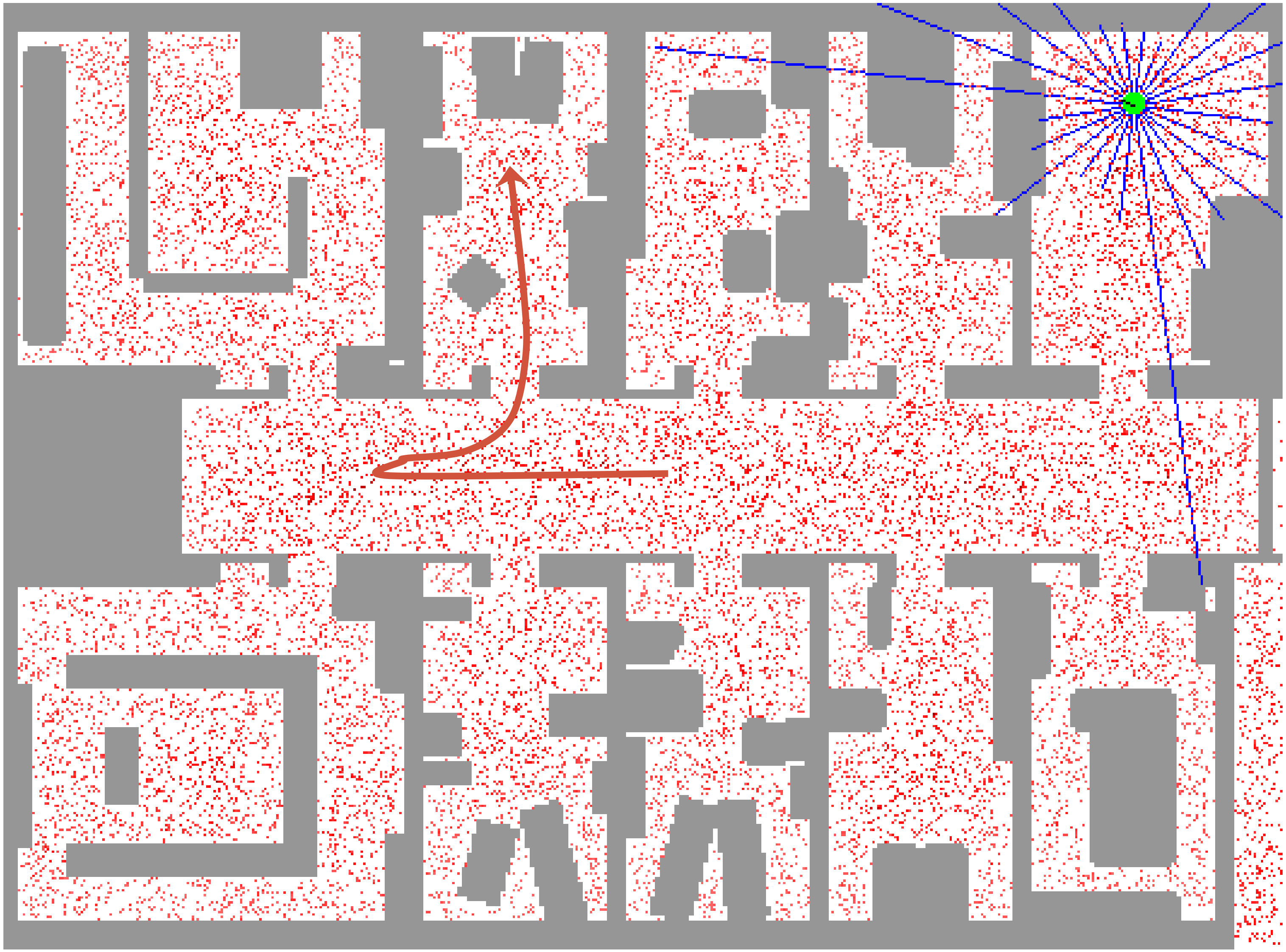
After Incorporating 65 Ultrasound Scans



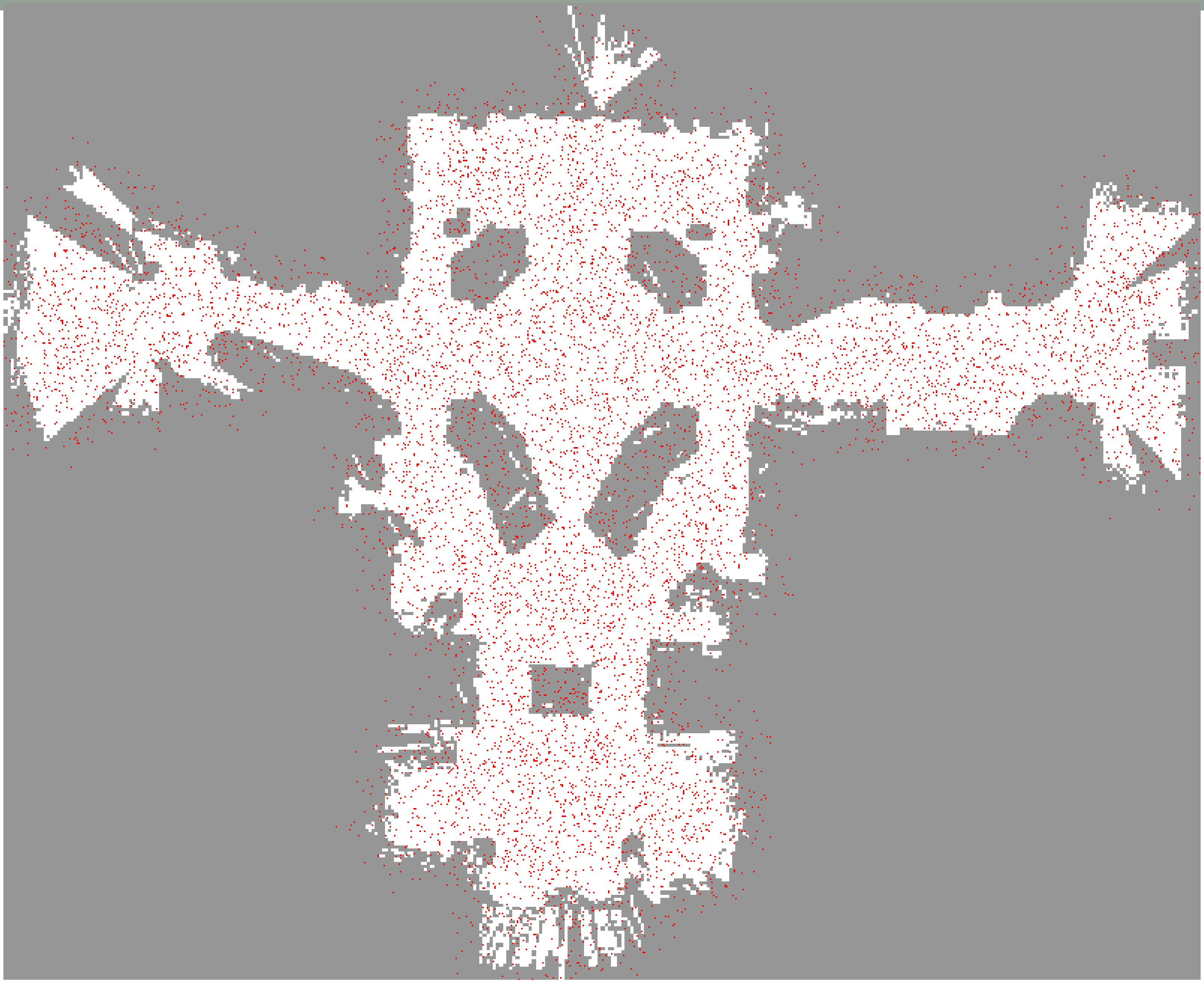
Estimated Path

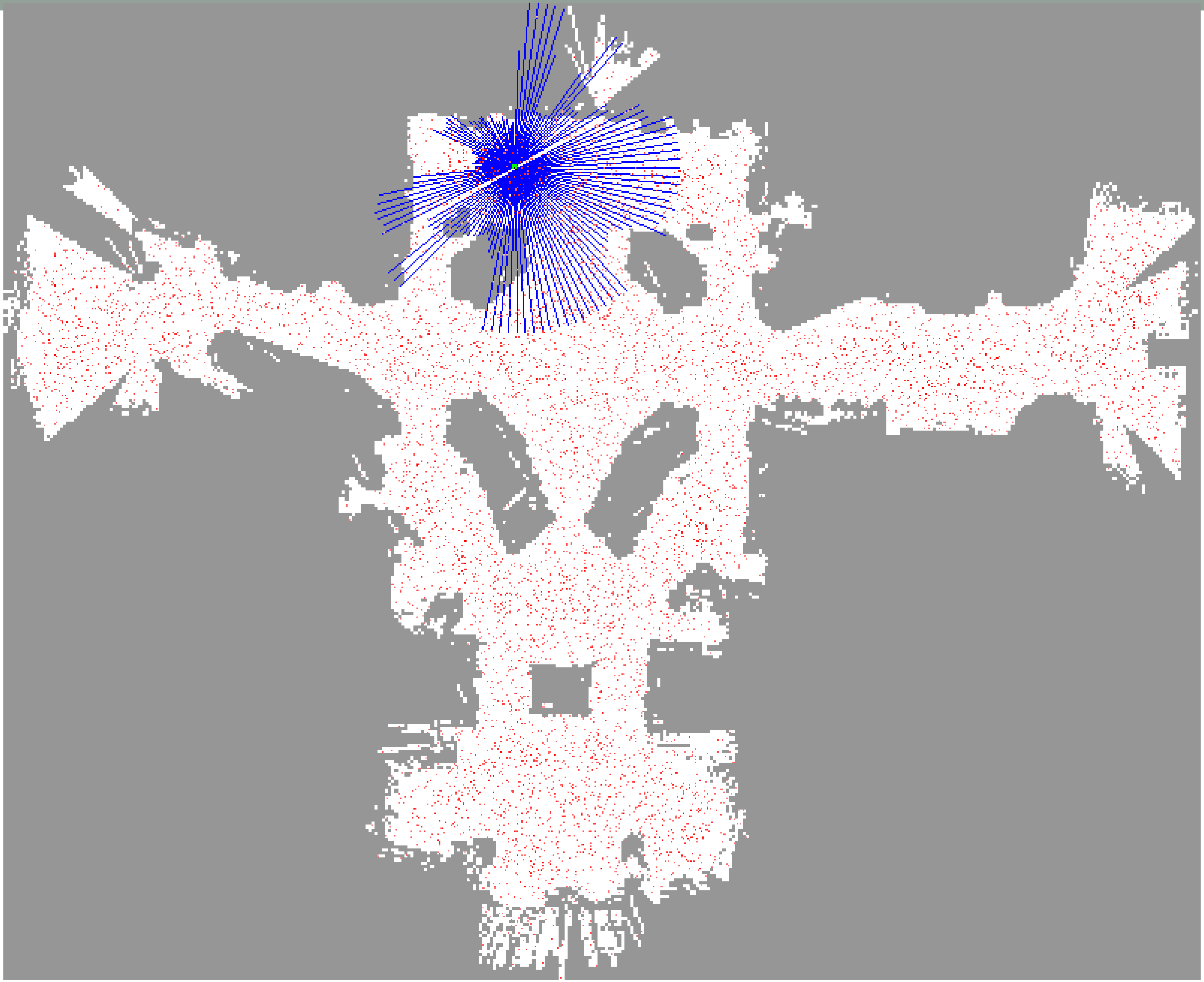


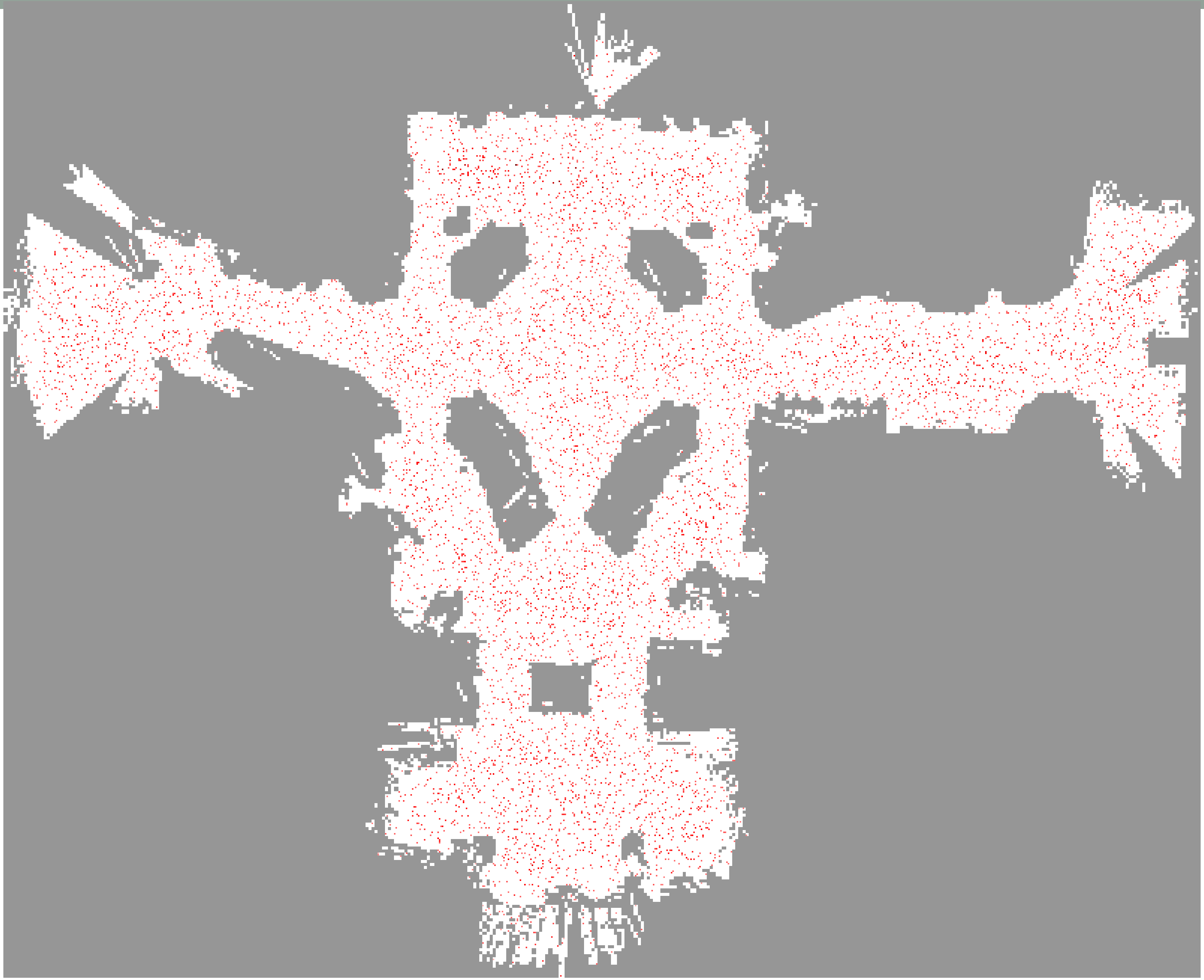
Sample-based Localization (sonar)

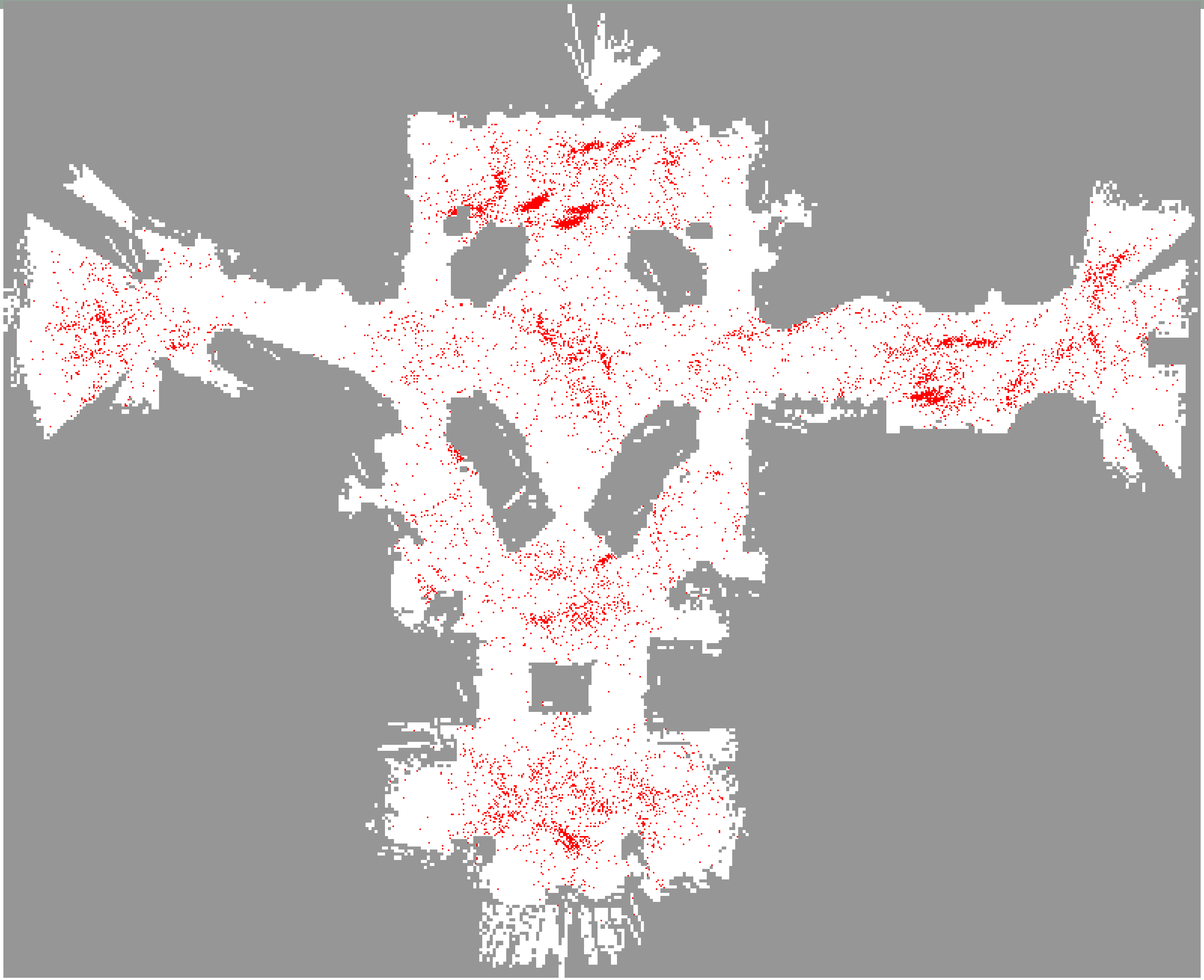


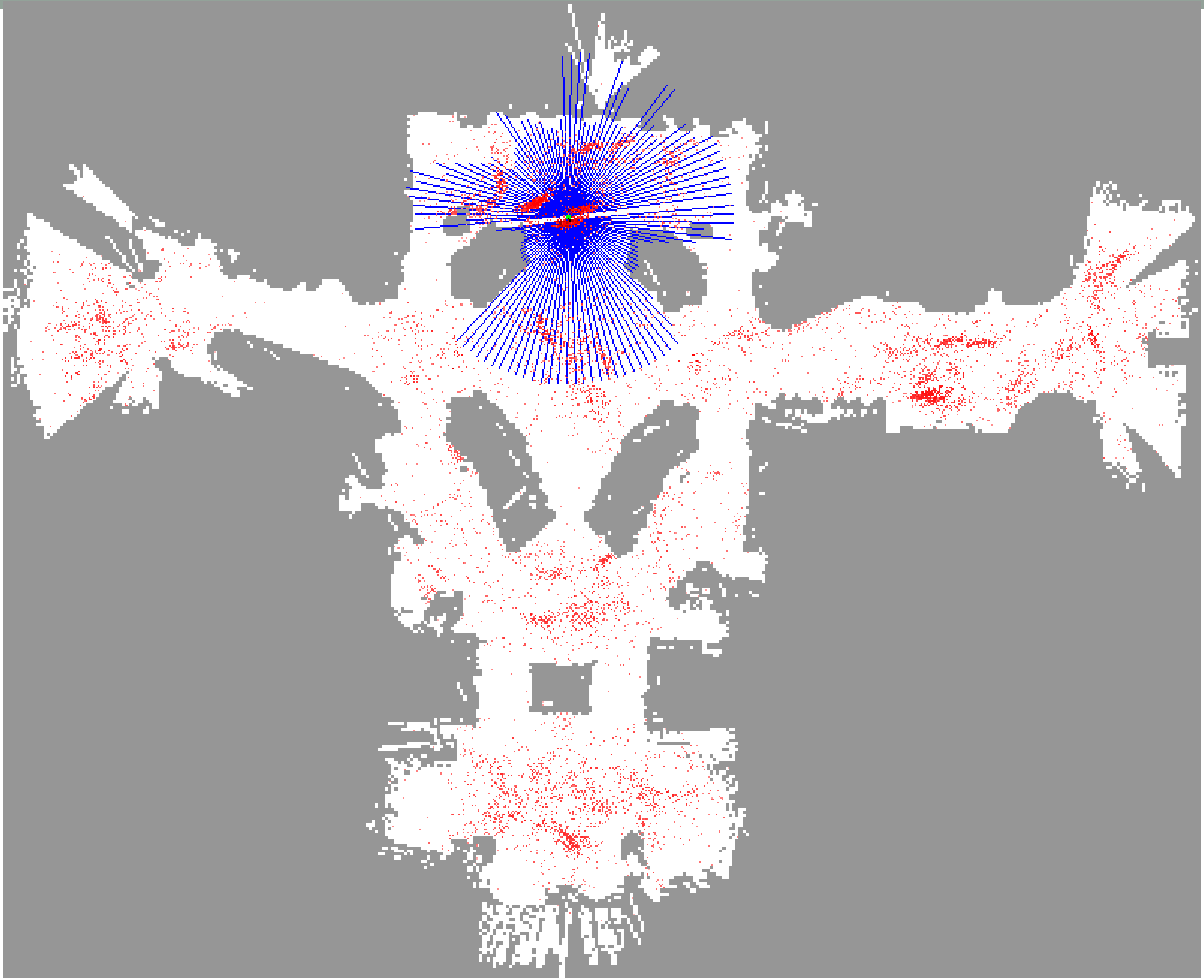
Smithsonian Museum of American History…

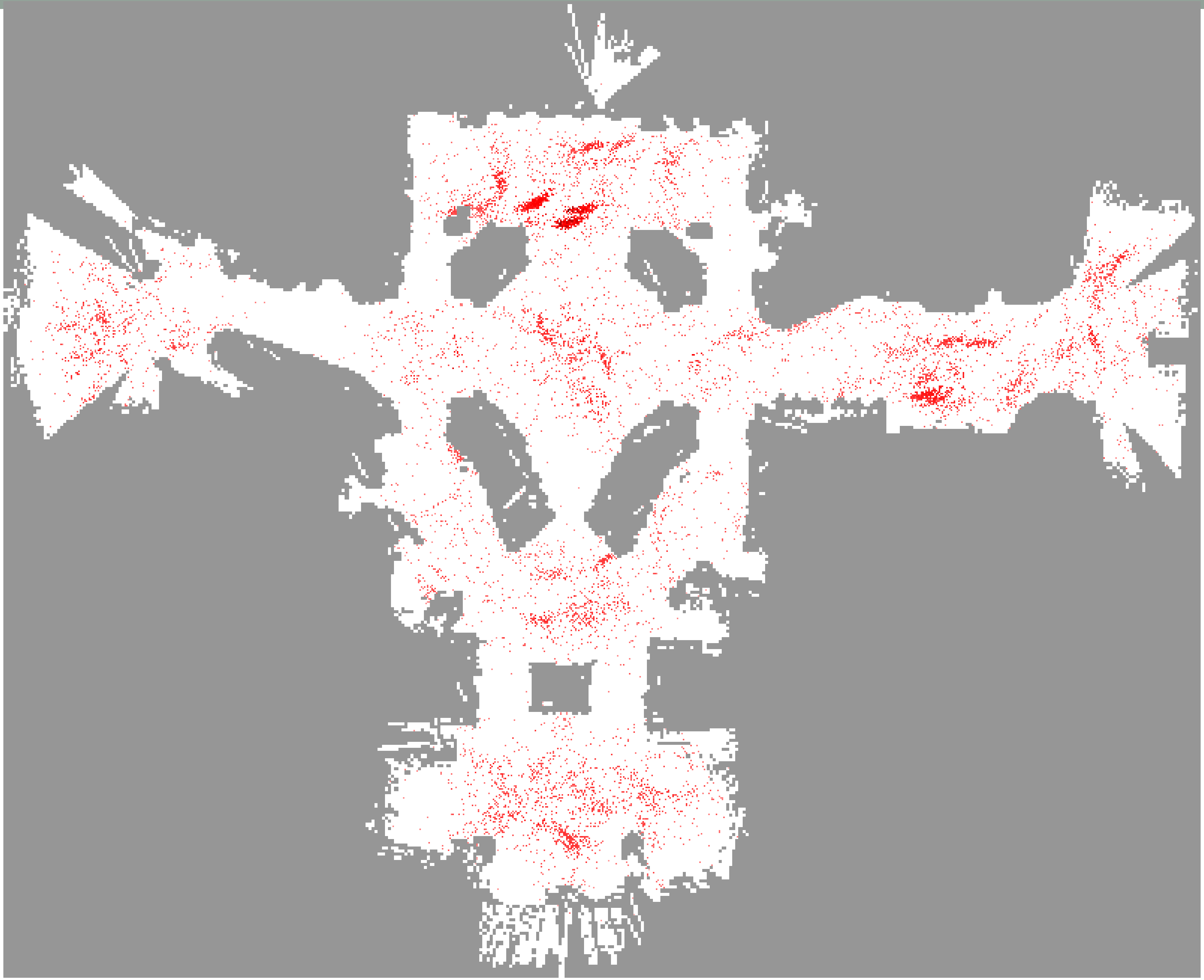


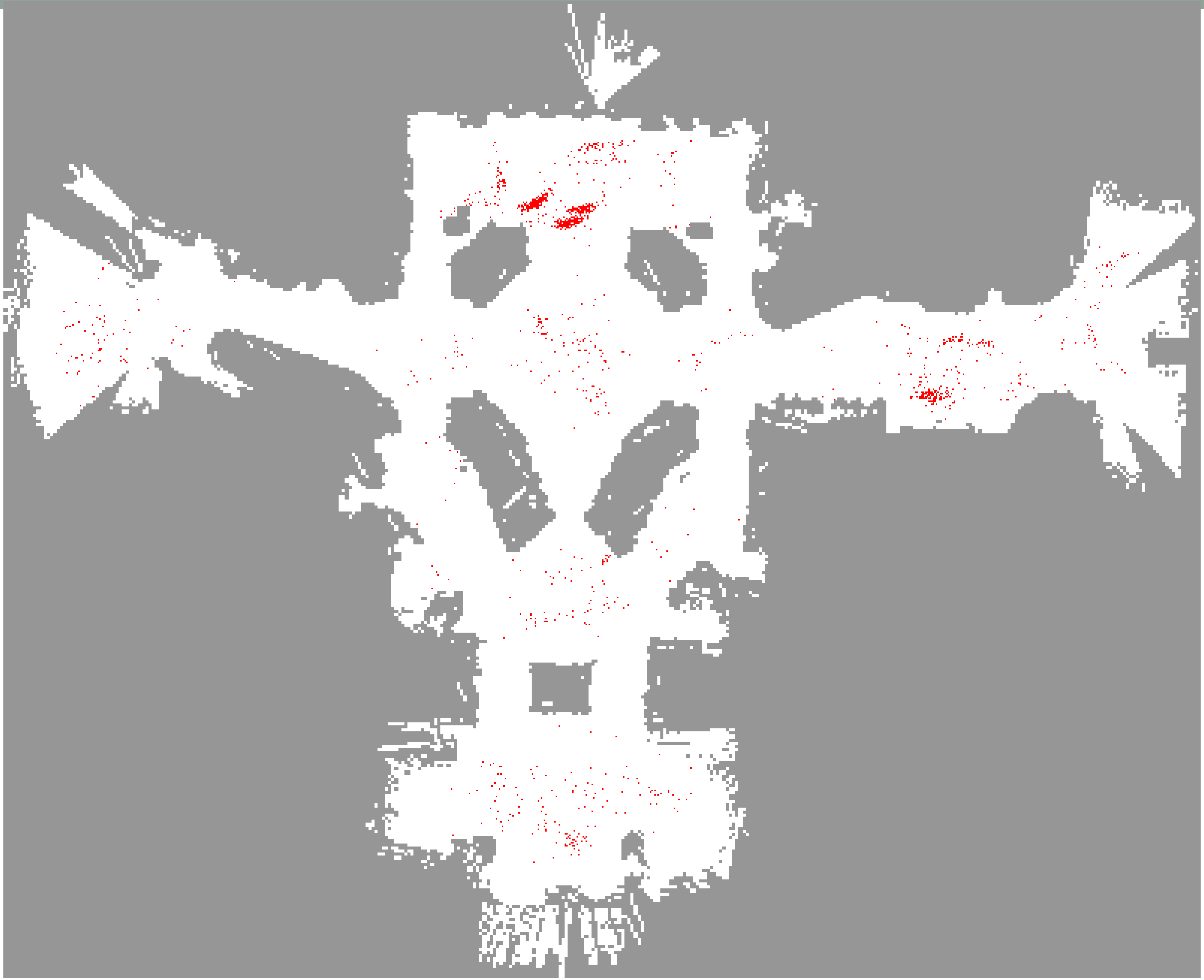


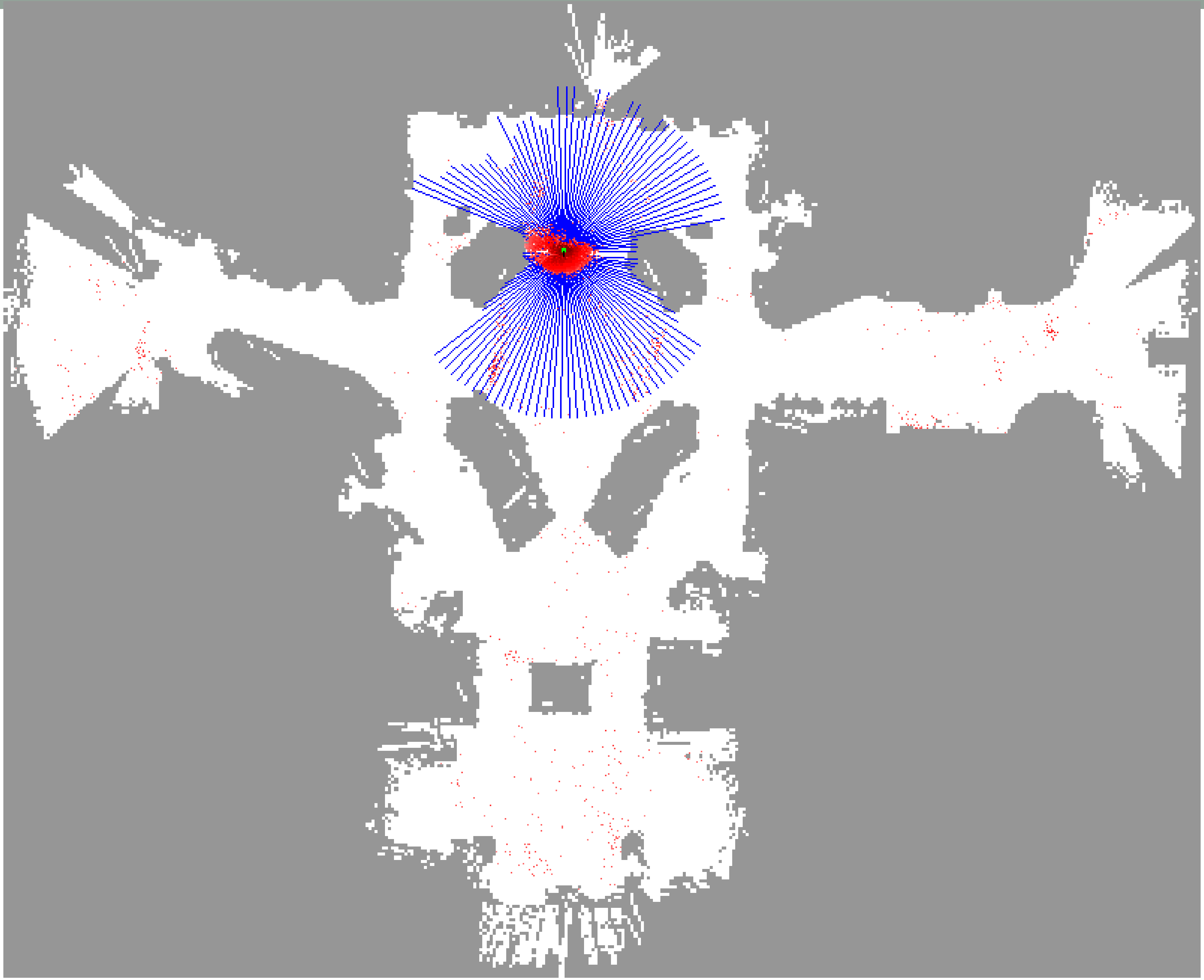


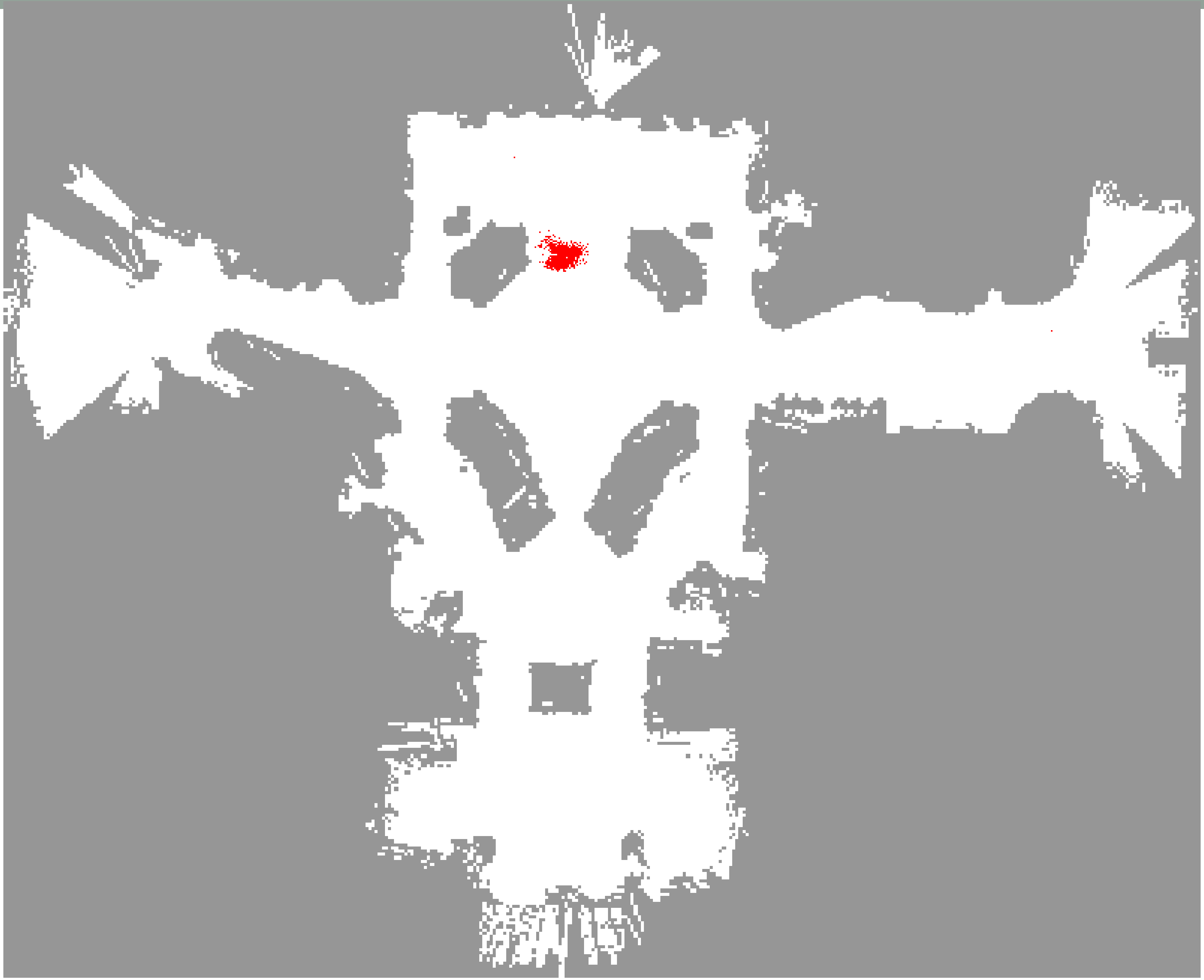


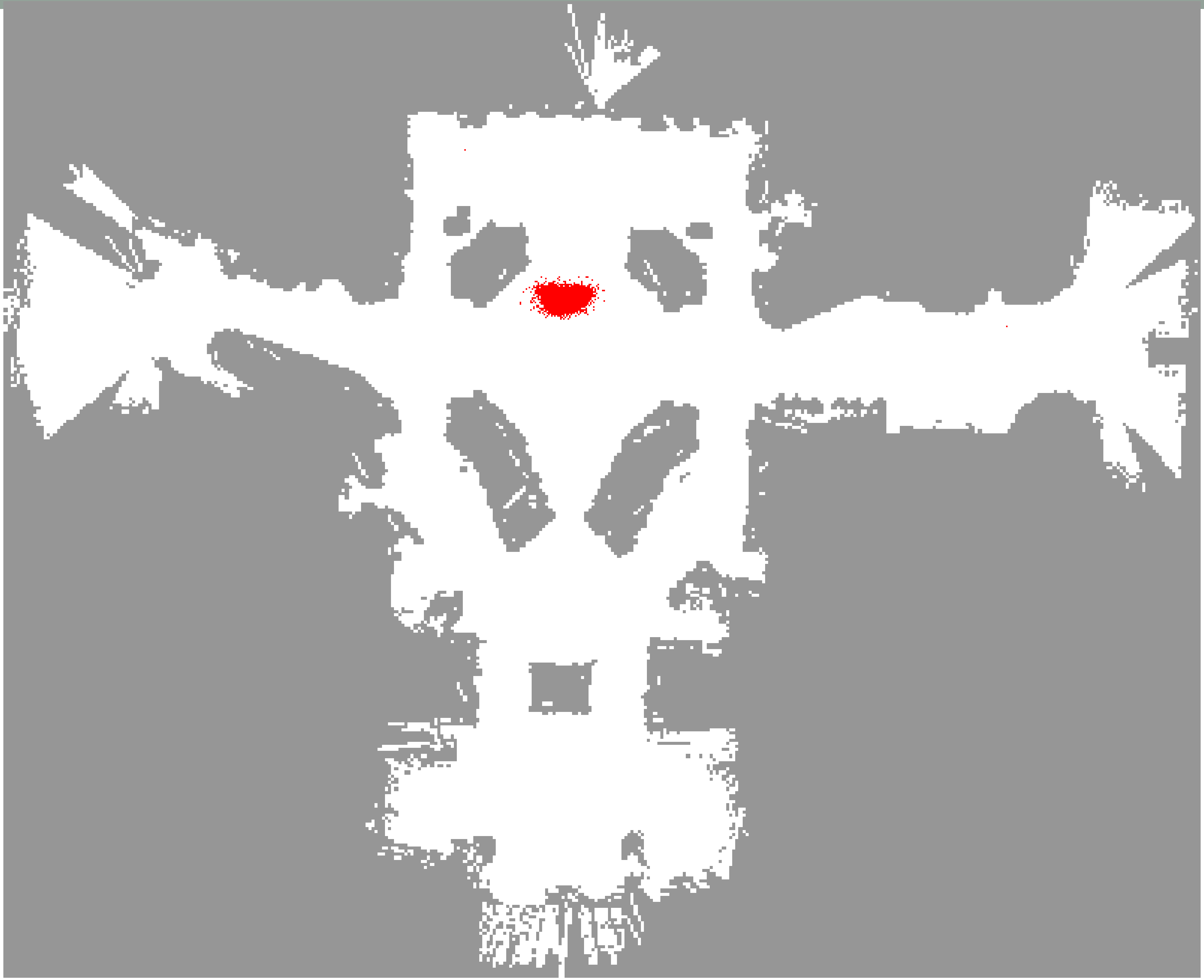


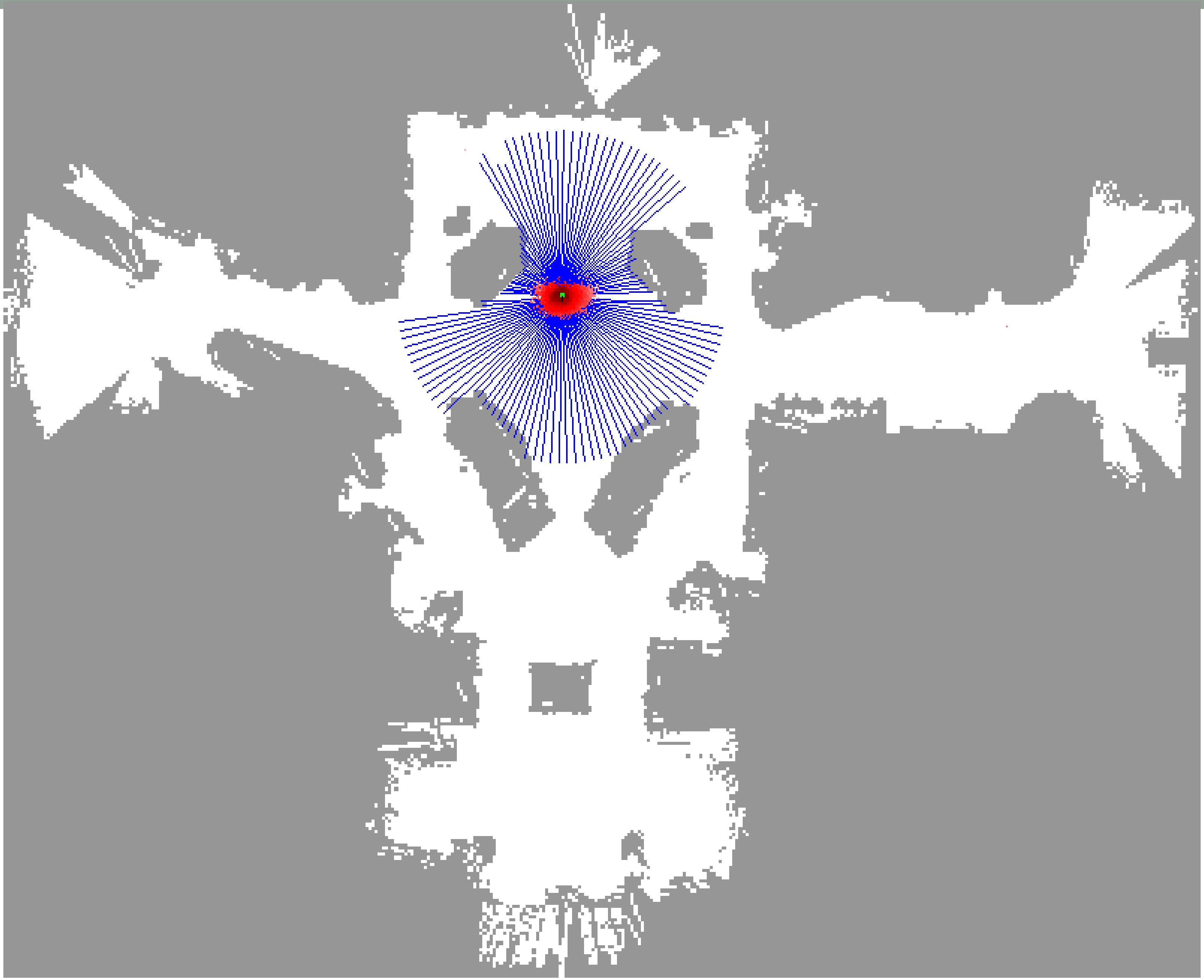


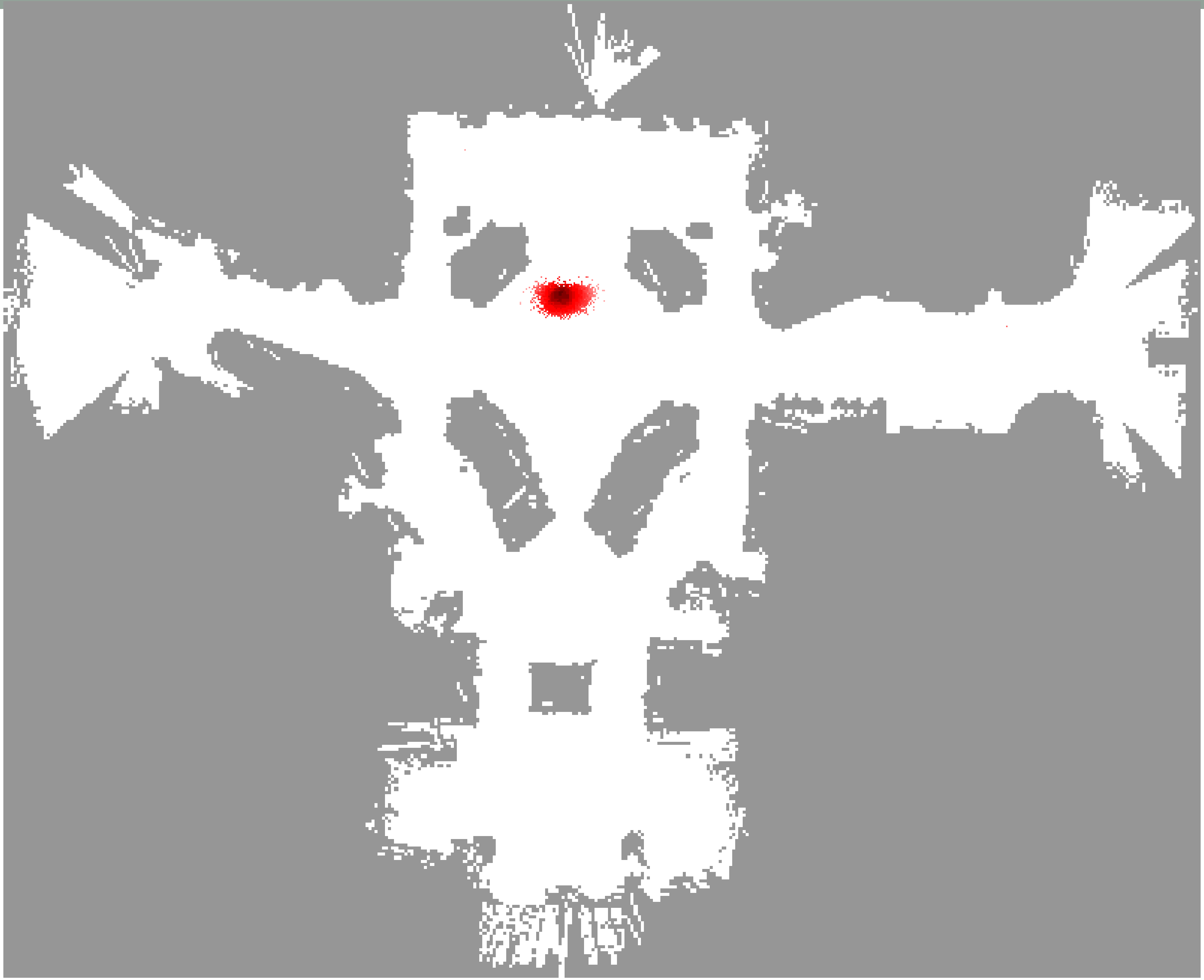


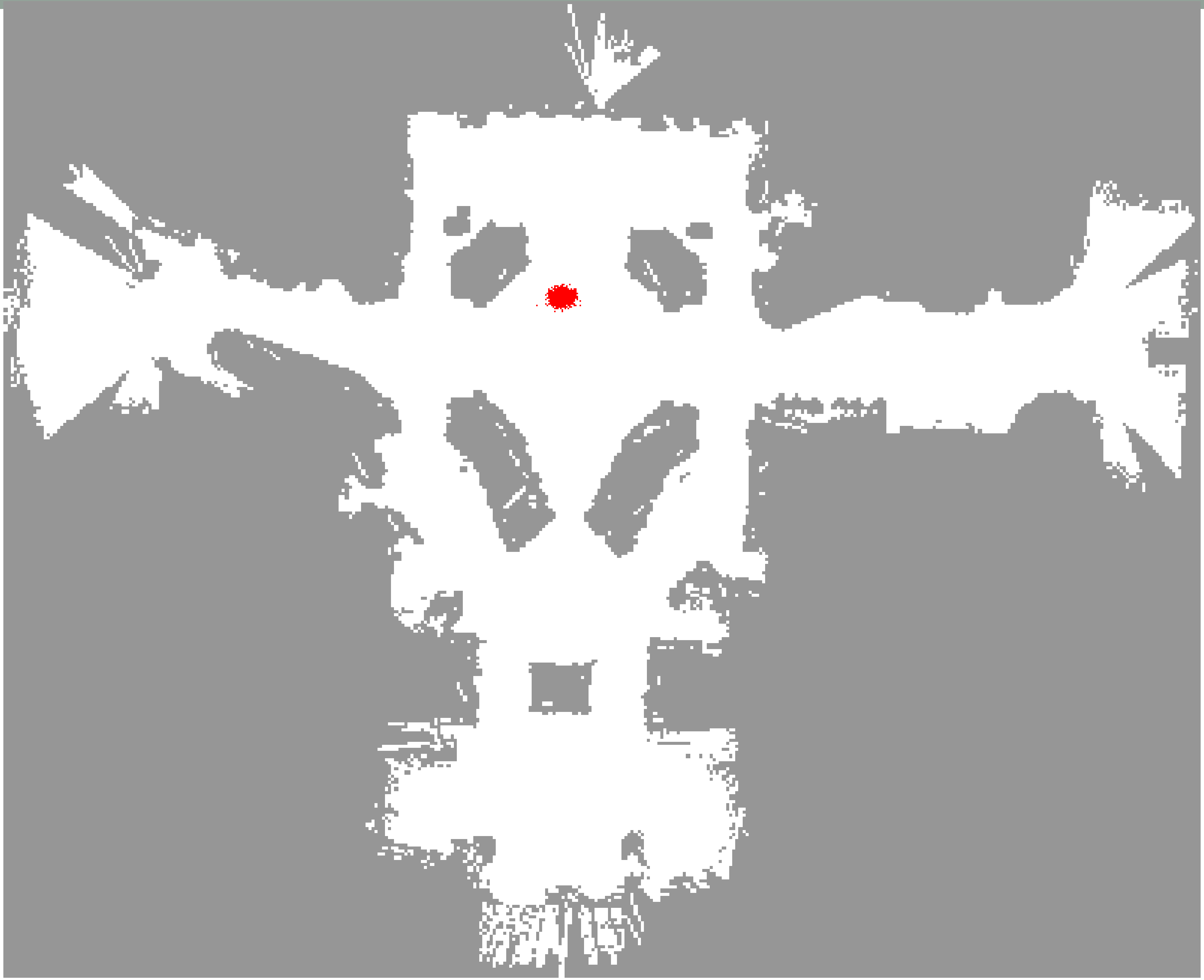


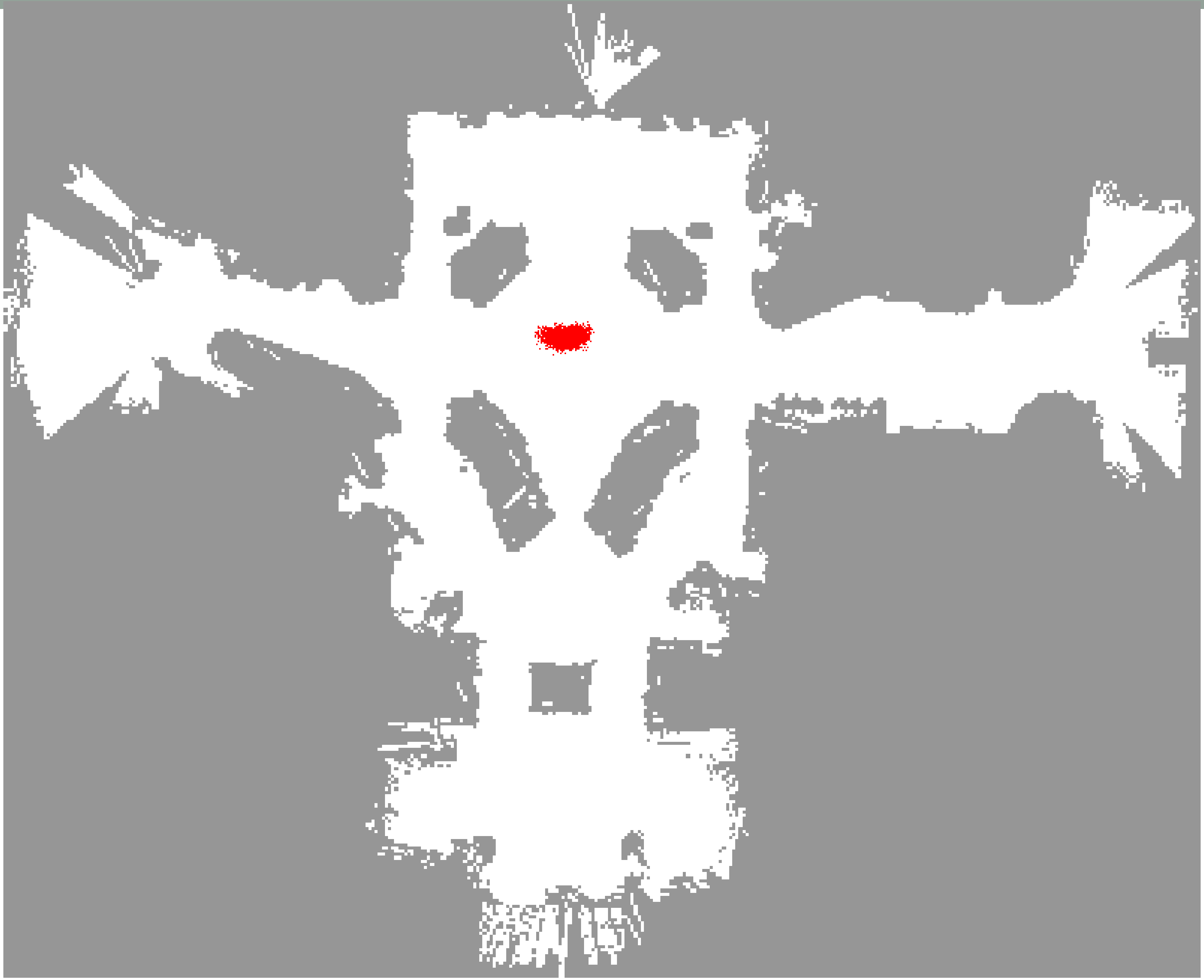


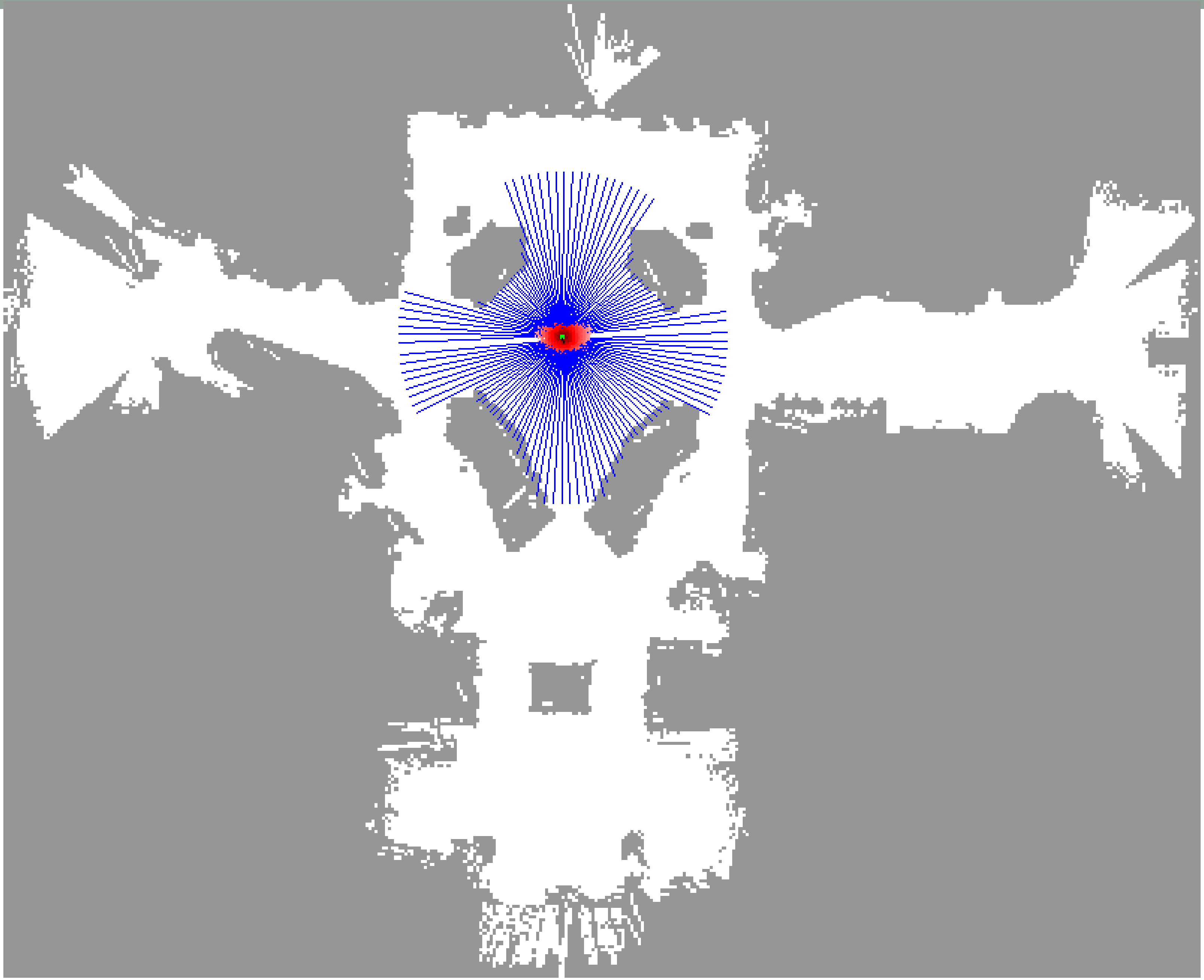






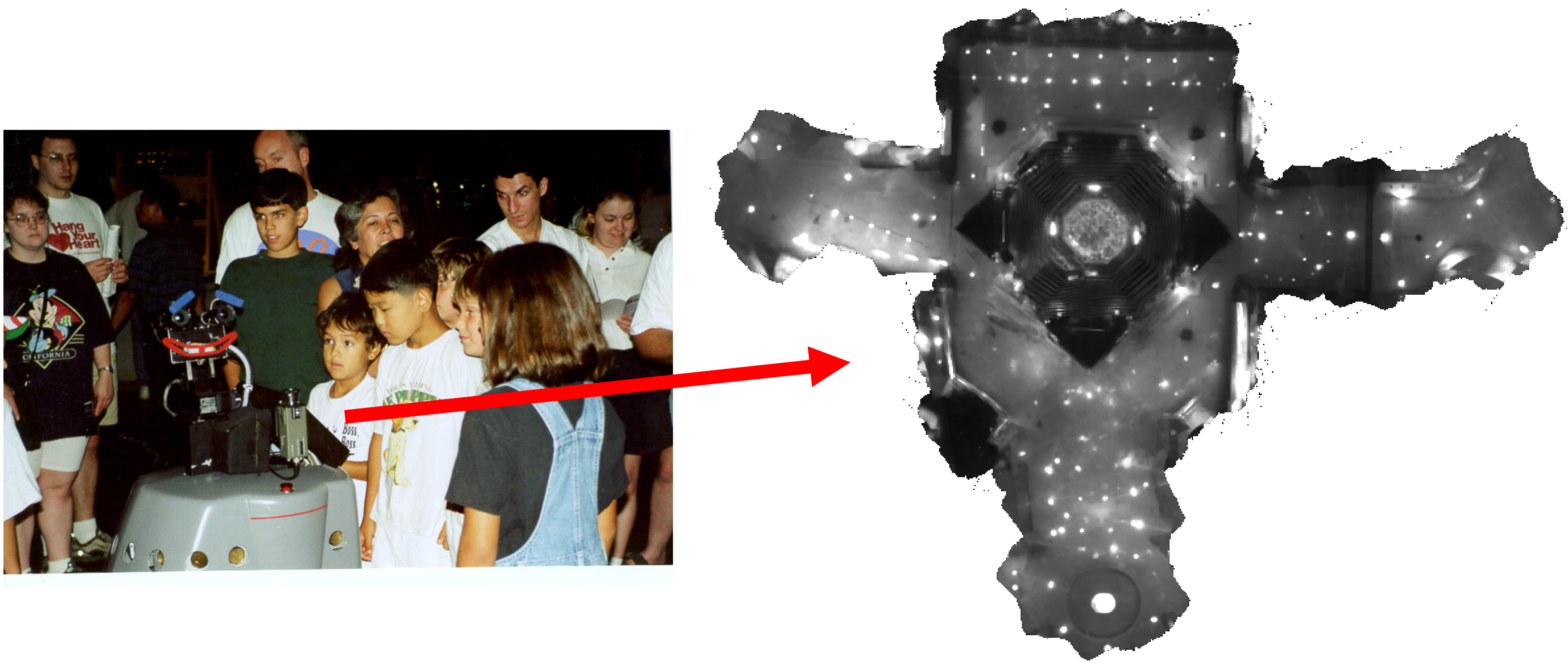






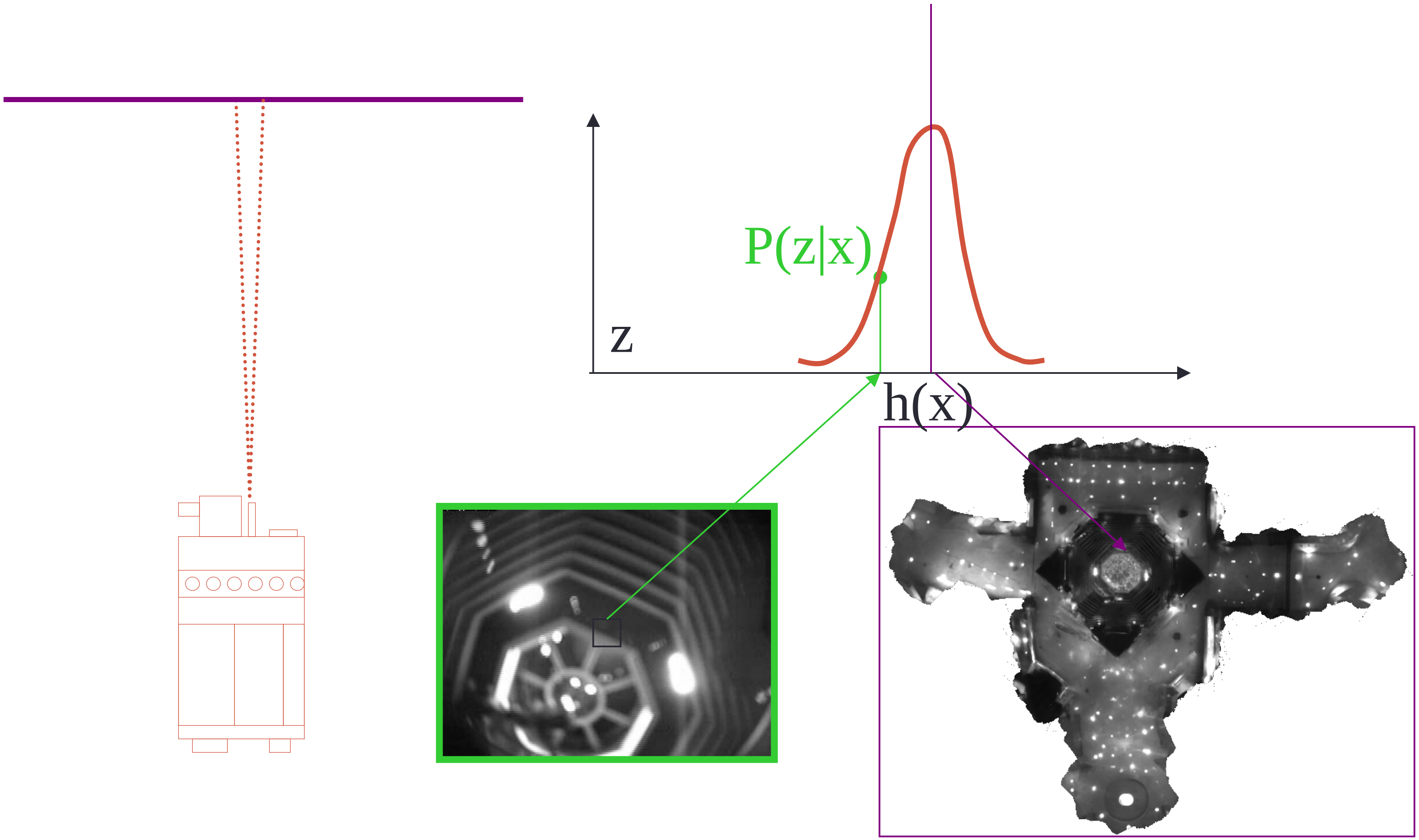
How about simple vision….

Using Ceiling Maps for Localization



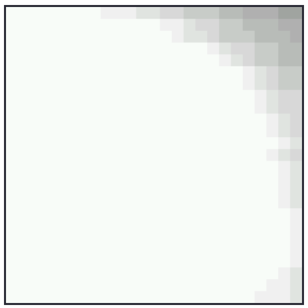
Dellaert, et al. 1997

Vision-based Localization

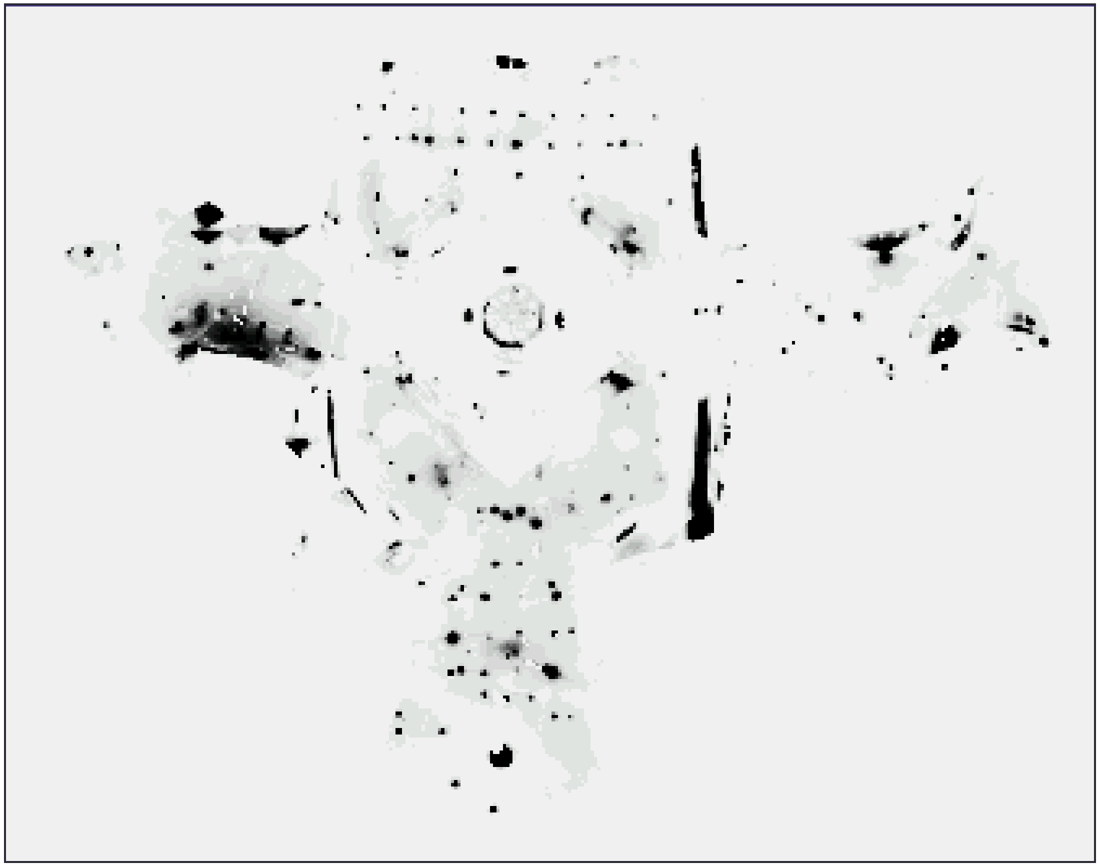


Under a Light

Measurement z:

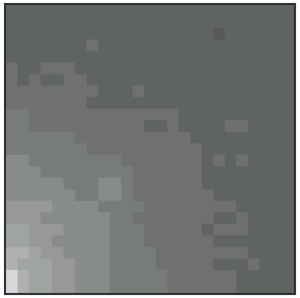


***P(z|x)*:**

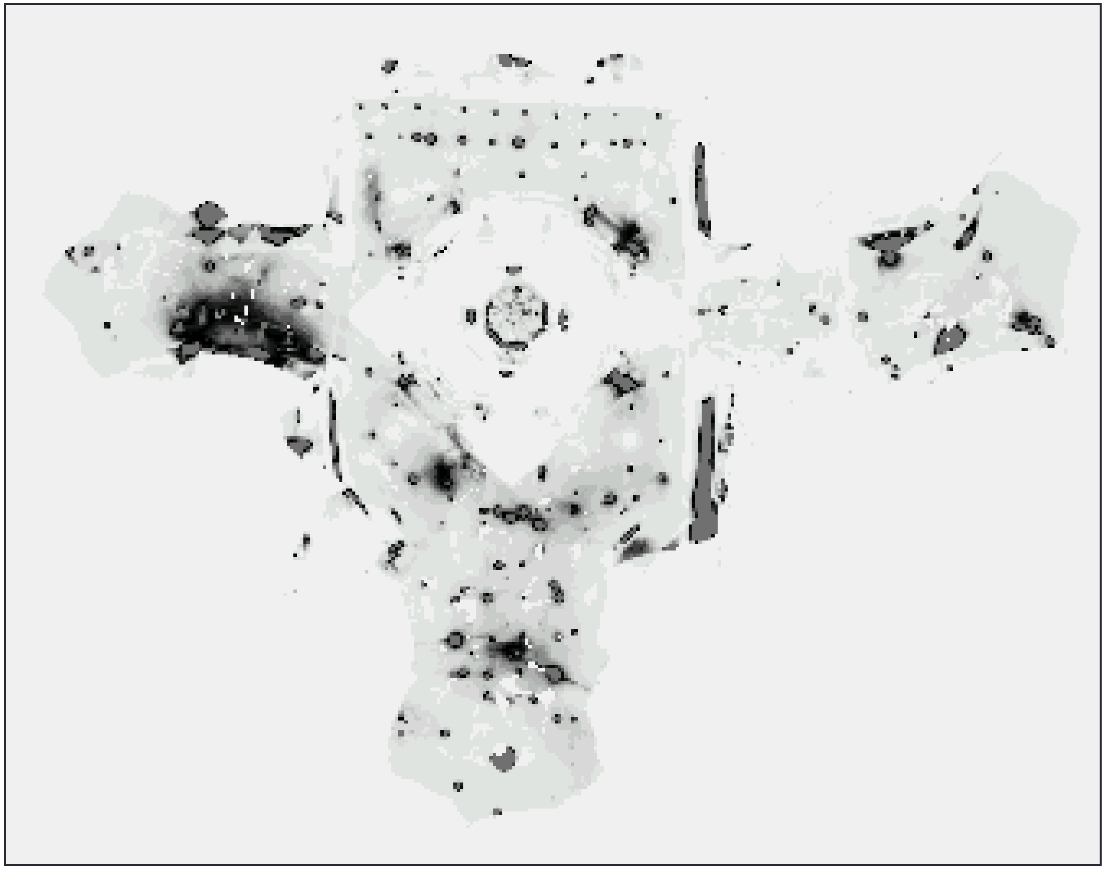


Next to a Light

Measurement z:



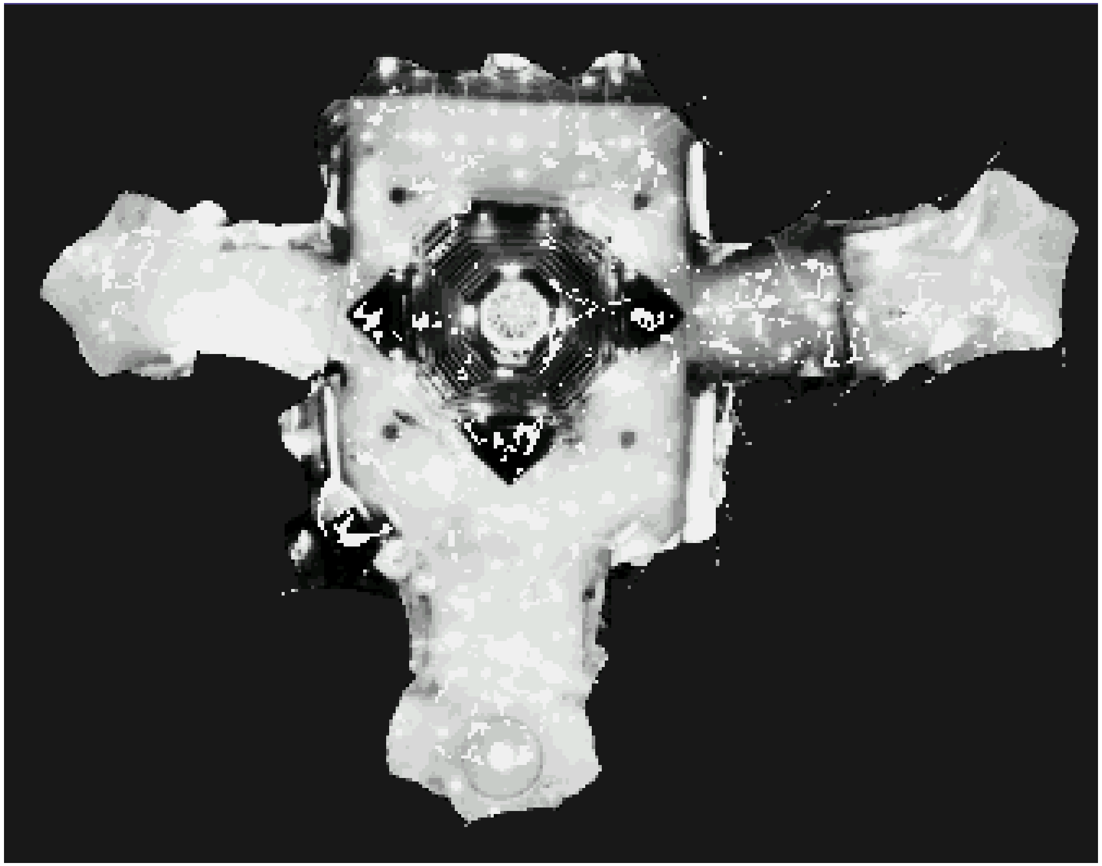
***P(z|x)*:**



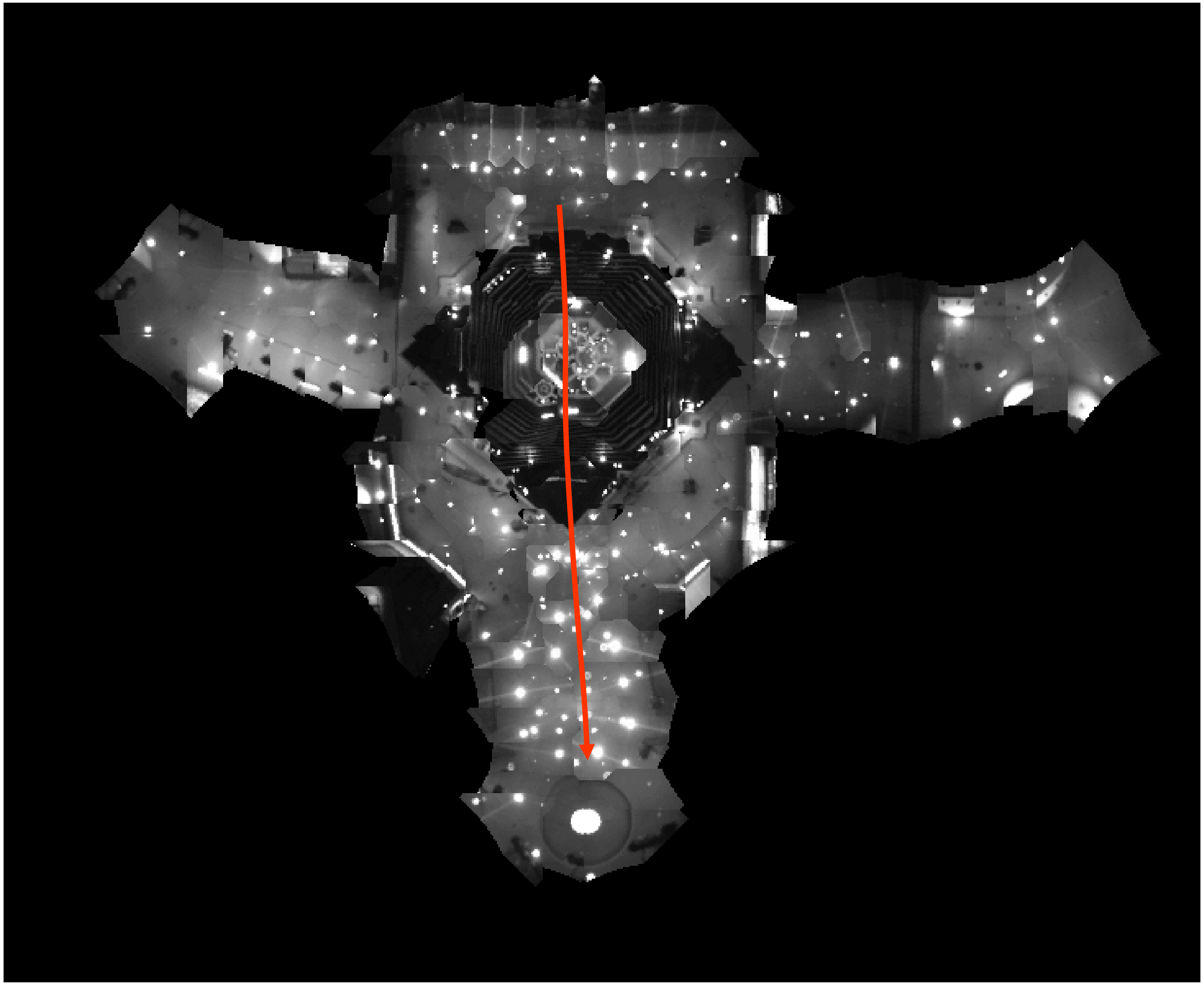
Elsewhere

Measurement z:

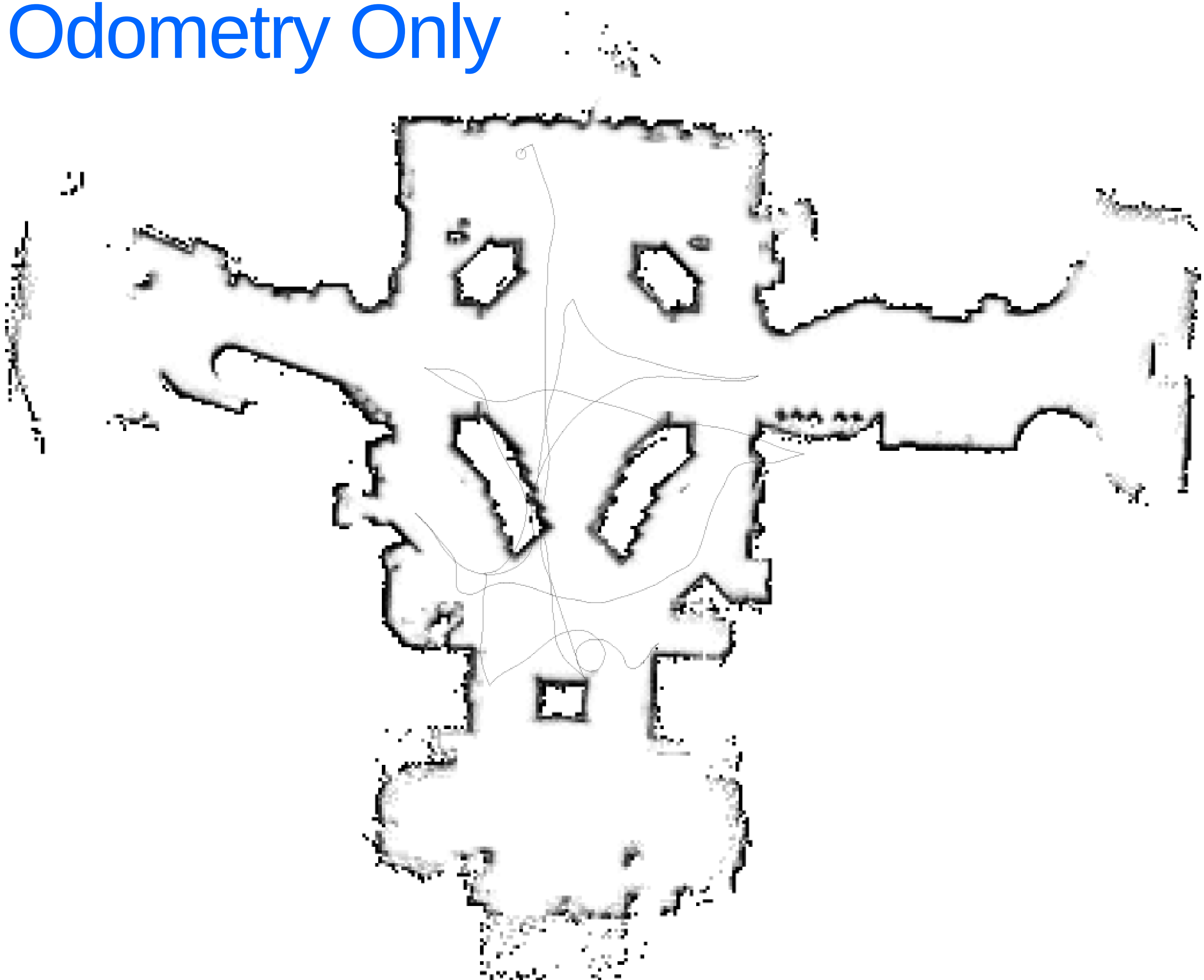
***P(z|x)*:**



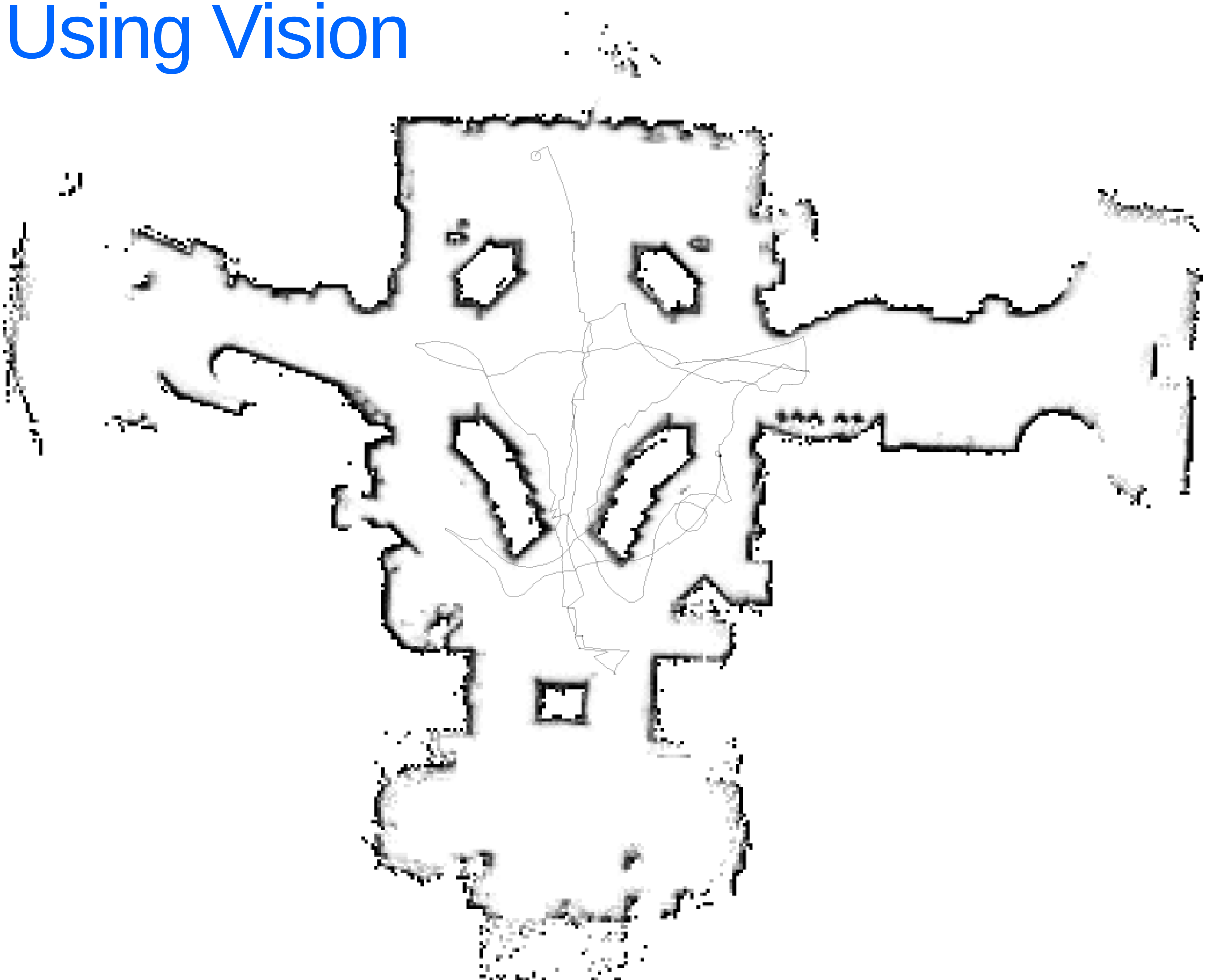
Global Localization Using Vision



Odometry Only



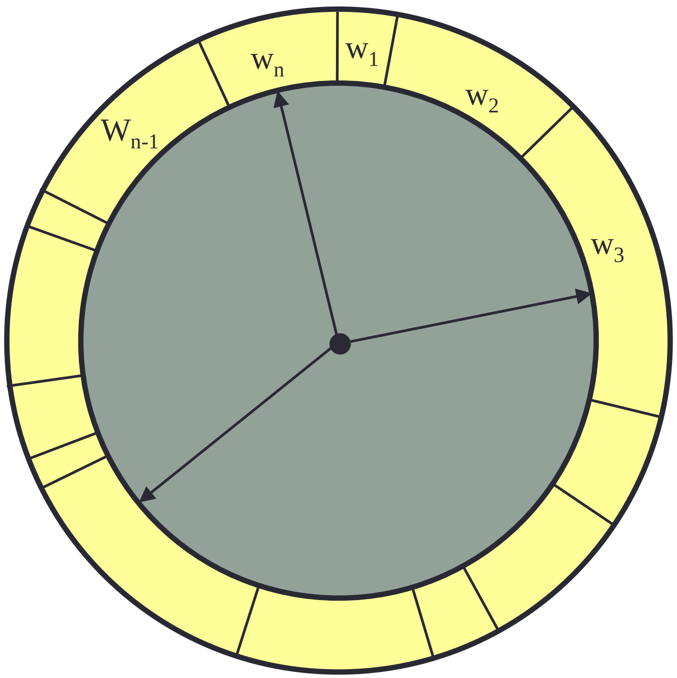
Using Vision



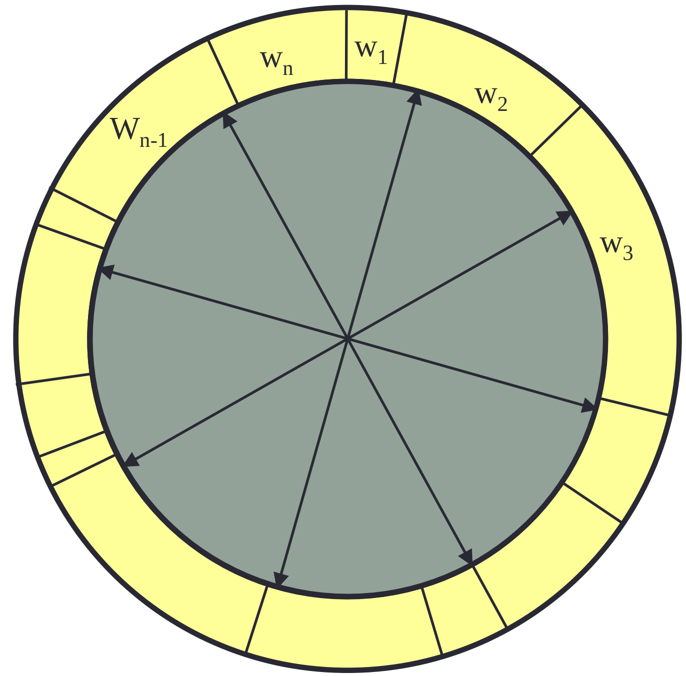
A detail: Resampling method can matter

- **Given**: Set *S* of weighted samples.
- **Wanted** : Random sample, where the probability of drawing x_i is given by w_i.
- Typically done *n* times with replacement to generate new sample set *S'*.
  - *Or even not done except when needed…too many low weight particles.*

Resampling



- Roulette wheel
- Binary search, n log n



- Stochastic universal sampling
- Systematic resampling
- Linear time complexity
- Easy to implement, low variance

Resampling Algorithm

1. Algorithm **systematic_resampling**(S,n):

2. $S' = \emptyset, c_1 = w^1$
3. **For** $i = 2 \ldots n$ — ***Generate cdf***
4. $c_i = c_{i-1} + w^i$
5. $u_1 \sim U[0, n^{-1}], i = 1$ — ***Initialize threshold***

6. **For** $j = 1 \ldots n$ — ***Draw samples …***
7. **While** ($u_j > c_i$) — ***Skip until next threshold reached***
8. $i = i + 1$
9. $S' = S' \cup \{< x^i, n^{-1} >\}$ — ***Insert***
10. $u_{j+1} = u_j + n^{-1}$ — ***Increment threshold***

11. **Return** S' *(Also called* ***stochastic universal sampling)***

PF: Practical Considerations

- If dealing with highly peaked observations
  - Add noise to observation and prediction models
  - Better proposal distributions: e.g., perform Kalman filter step to determine proposal
- Overestimating noise often reduces number of required samples
- Recover from failure by selectively adding samples from observations
- Recover from failure by uniformly adding some samples
- Can Resample only when necessary (efficiency of representation measured by variance of weights)

Tracking issues

- Next time?

Tracking issues

- Initialization
  - Manual
  - Background subtraction
  - Detection

Tracking issues

- Initialization
- Obtaining observation and dynamics model
  - Dynamics model: learn (very difficult) or specify using domain knowledge
  - Generative observation model: “render” the state on top of the image and compare

Tracking issues

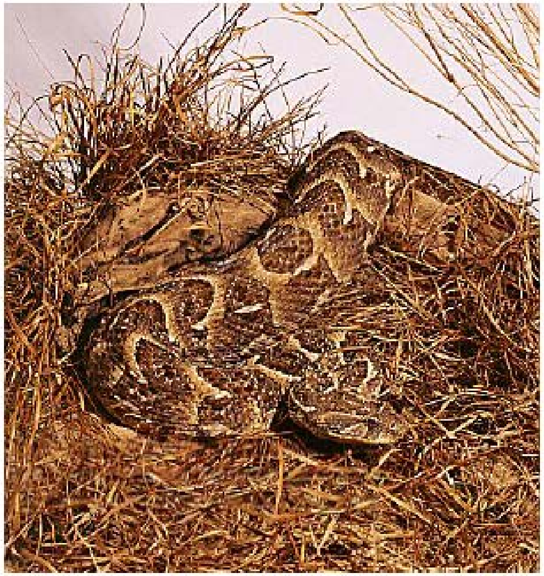
- Initialization
- Obtaining observation and dynamics model
- Prediction vs. correction
  - If the dynamics model is too strong, will end up ignoring the data
  - If the observation model is too strong, tracking is reduced to repeated detection

Tracking issues

- Initialization
- Obtaining observation and dynamics model
- Prediction vs. correction
- Data association
  - What if we don’t know which measurements to associate with which tracks?

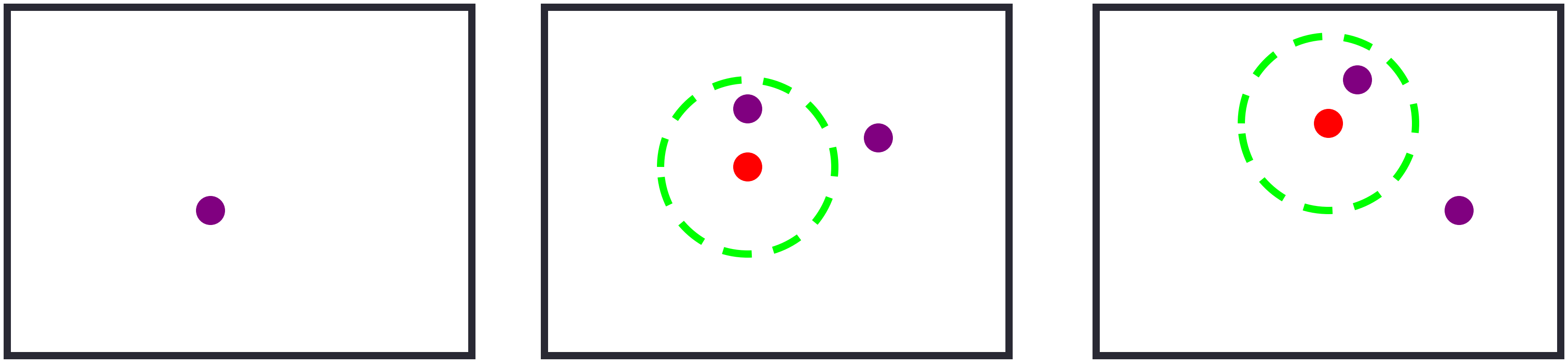
Data association

- So far, we've assumed the entire measurement to be relevant to determining the state
- In reality, there may be uninformative measurements (clutter) or measurements may belong to different tracked objects
- **Data association**: task of determining which measurements go with which tracks



Data association

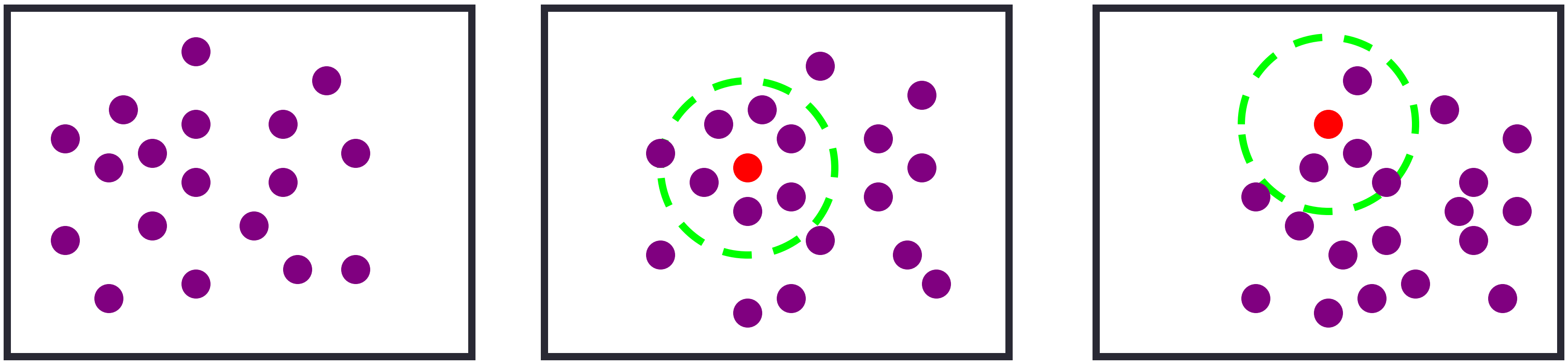
- Simple strategy: only pay attention to the measurement that is “closest” to the prediction



Source: Lana Lazebnik

Data association

- Simple strategy: only pay attention to the measurement that is “closest” to the prediction



Doesn’t always work…

Alternative: keep track of **multiple hypotheses** at once…

Source: Lana Lazebnik

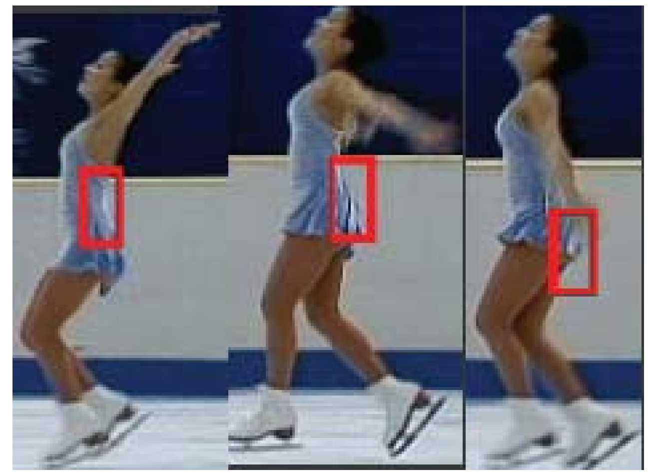
Data association

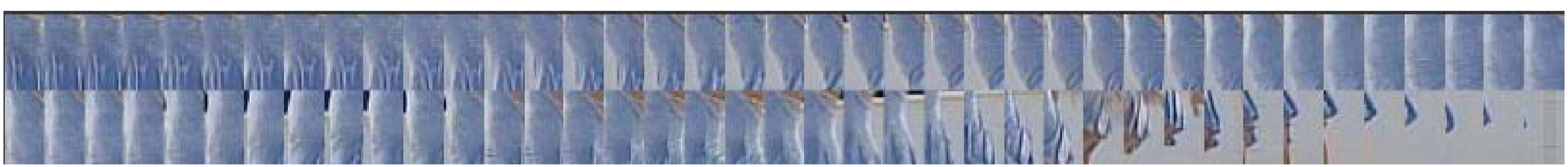
- Simple strategy: only pay attention to the measurement that is “closest” to the prediction
- More sophisticated strategy: keep track of multiple state/observation hypotheses
  - Can be done with particle filtering
- This is a general problem in computer vision, there is no easy solution

Tracking issues

- Initialization
- Obtaining observation and dynamics model
- Prediction vs. correction
- Data association
- Drift
  - Errors caused by dynamical model, observation model, and data association tend to accumulate over time

Drift





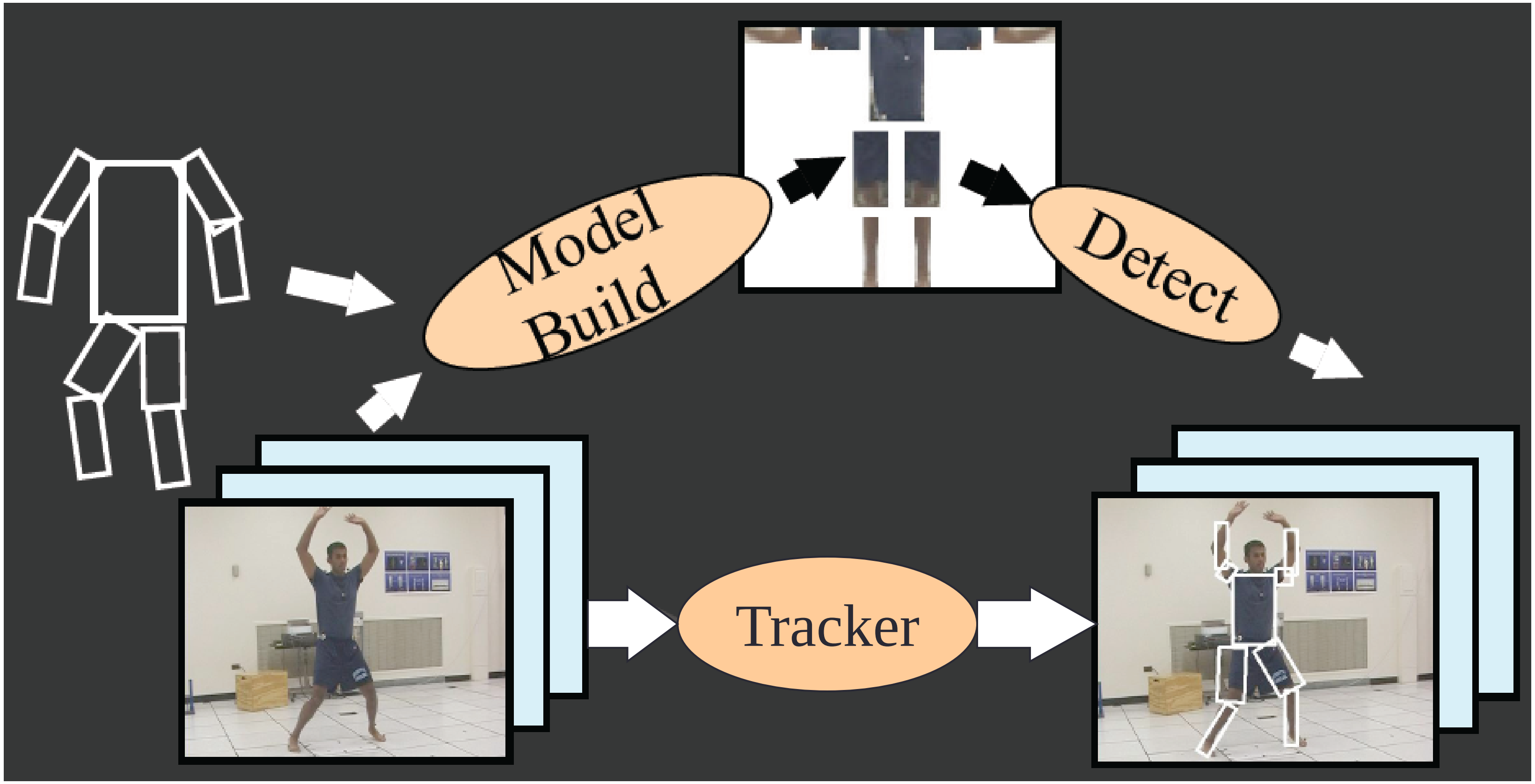
D. Ramanan, D. Forsyth, and A. Zisserman. [Tracking People by Learning their Appearance](). PAMI 2007.

Tracking people by learning their appearance

- Person model = appearance + structure (+ dynamics)
- Structure and dynamics are generic, appearance is person-specific
- Trying to acquire an appearance model “on the fly” can lead to drift
- Instead, can use the whole sequence to initialize the appearance model and then keep it fixed while tracking
- Given strong structure and appearance models, tracking can essentially be done by repeated detection (with some smoothing)

D. Ramanan, D. Forsyth, and A. Zisserman. [Tracking People by Learning their Appearance](). PAMI 2007.

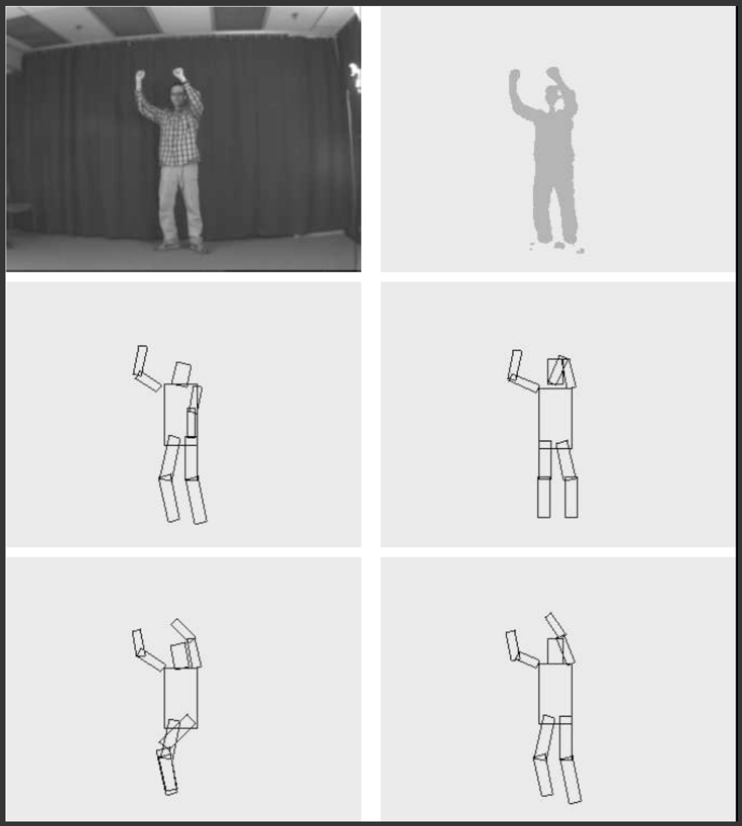
Tracking people by learning their appearance

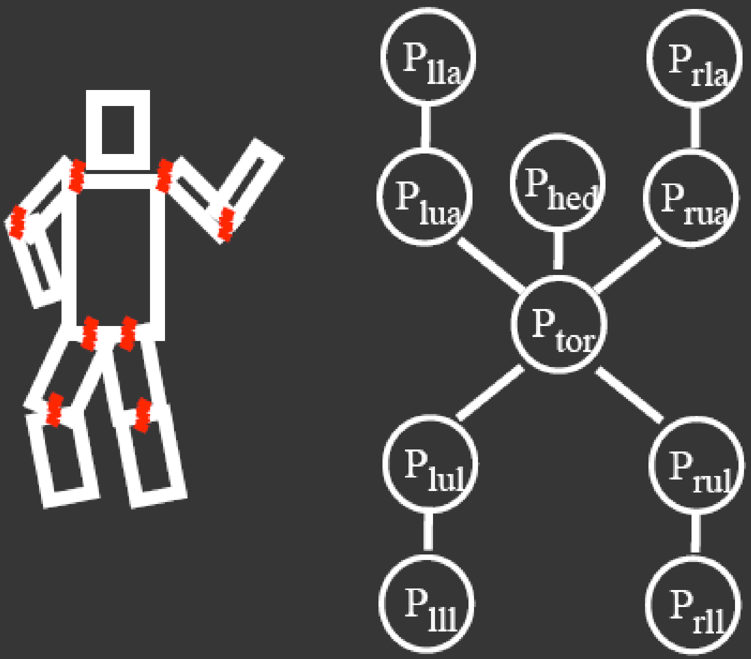


D. Ramanan, D. Forsyth, and A. Zisserman. [Tracking People by Learning their Appearance](). PAMI 2007.

Pictorial structure model

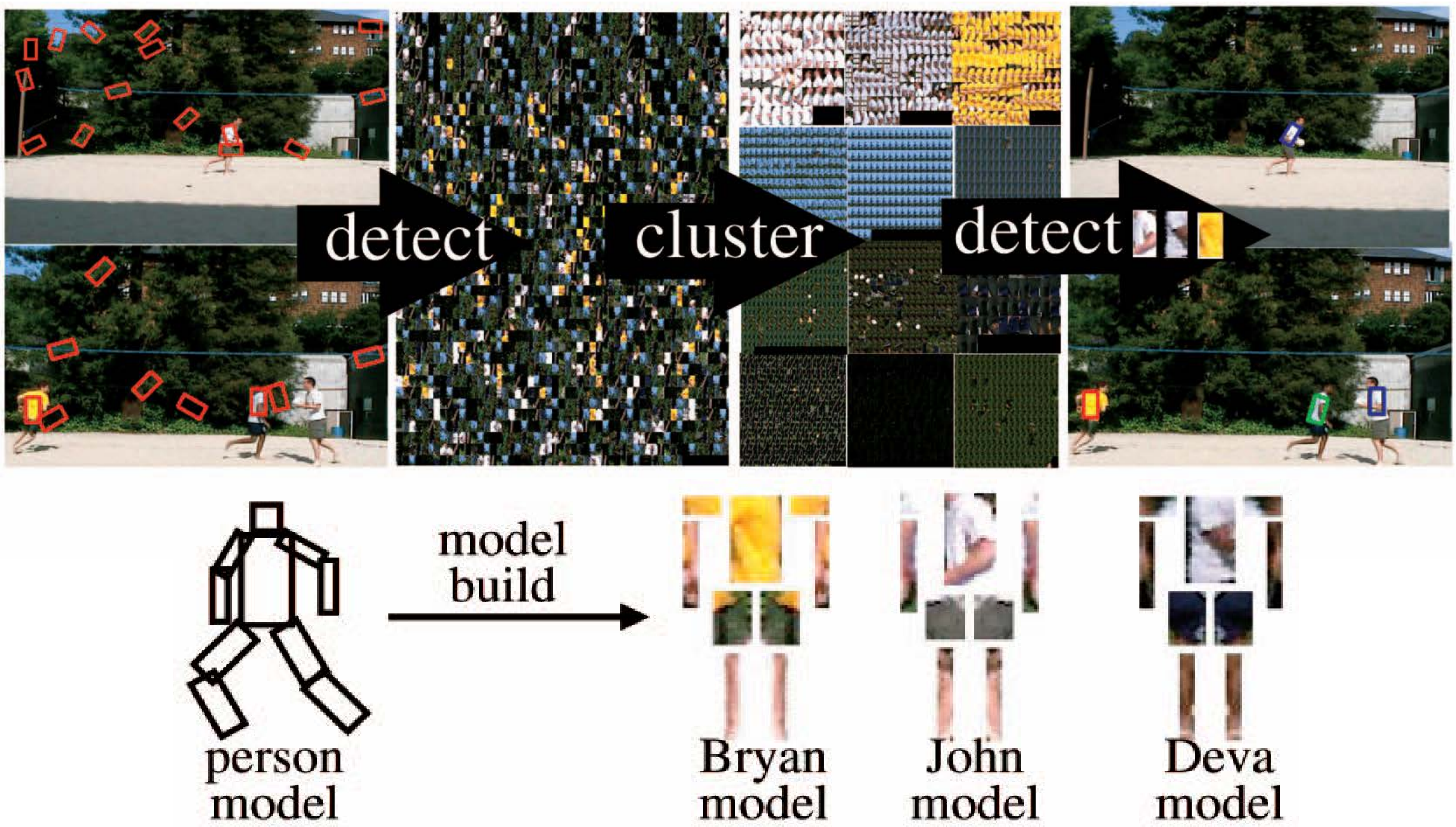
Fischler and Elschlager(73), Felzenszwalb and Huttenlocher(00)





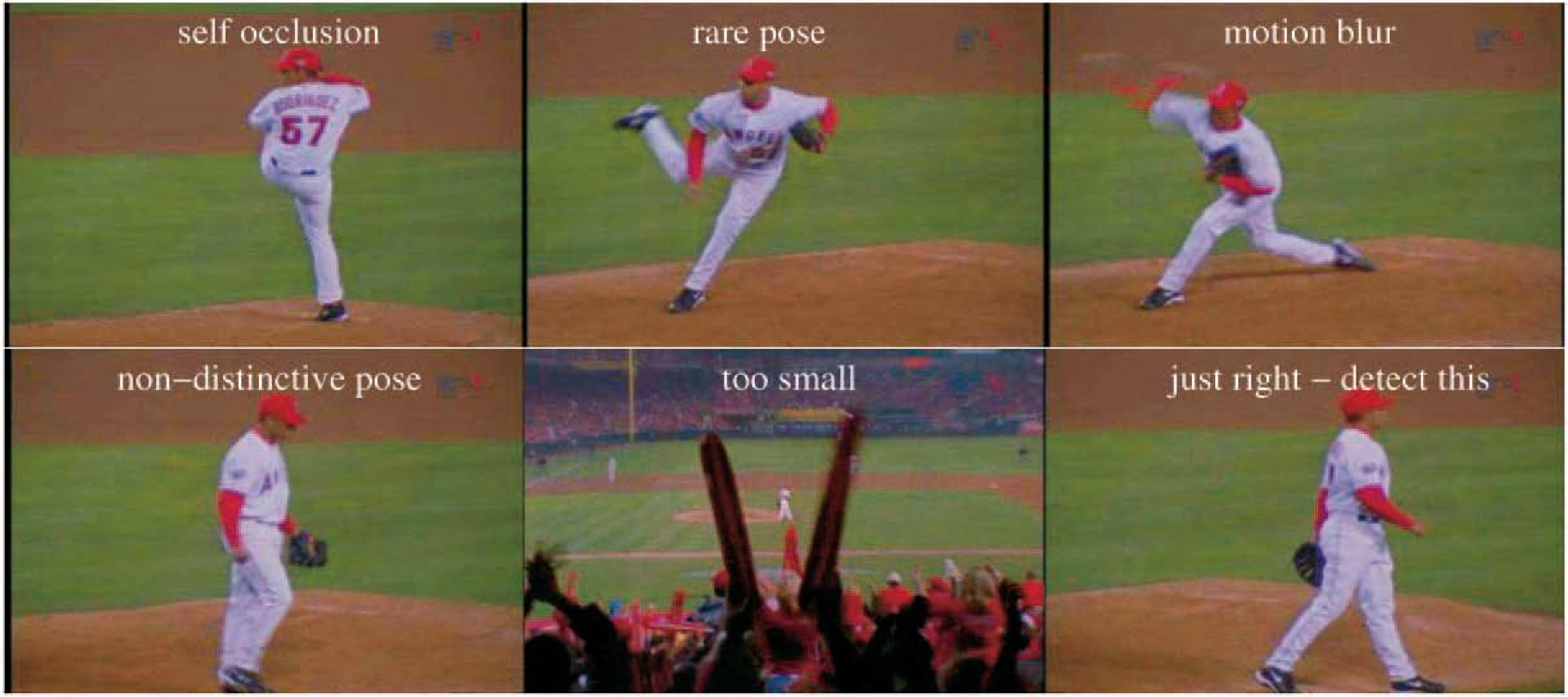
$$\Pr(P_{tor}, P_{arm}, \ldots | Im) \propto \prod_{i,j} \underbrace{\Pr(P_i \mid P_j)}_{\text{part geometry}} \prod_{i} \underbrace{\Pr(Im(P_i))}_{\text{part appearance}}$$

Bottom-up initialization: Clustering



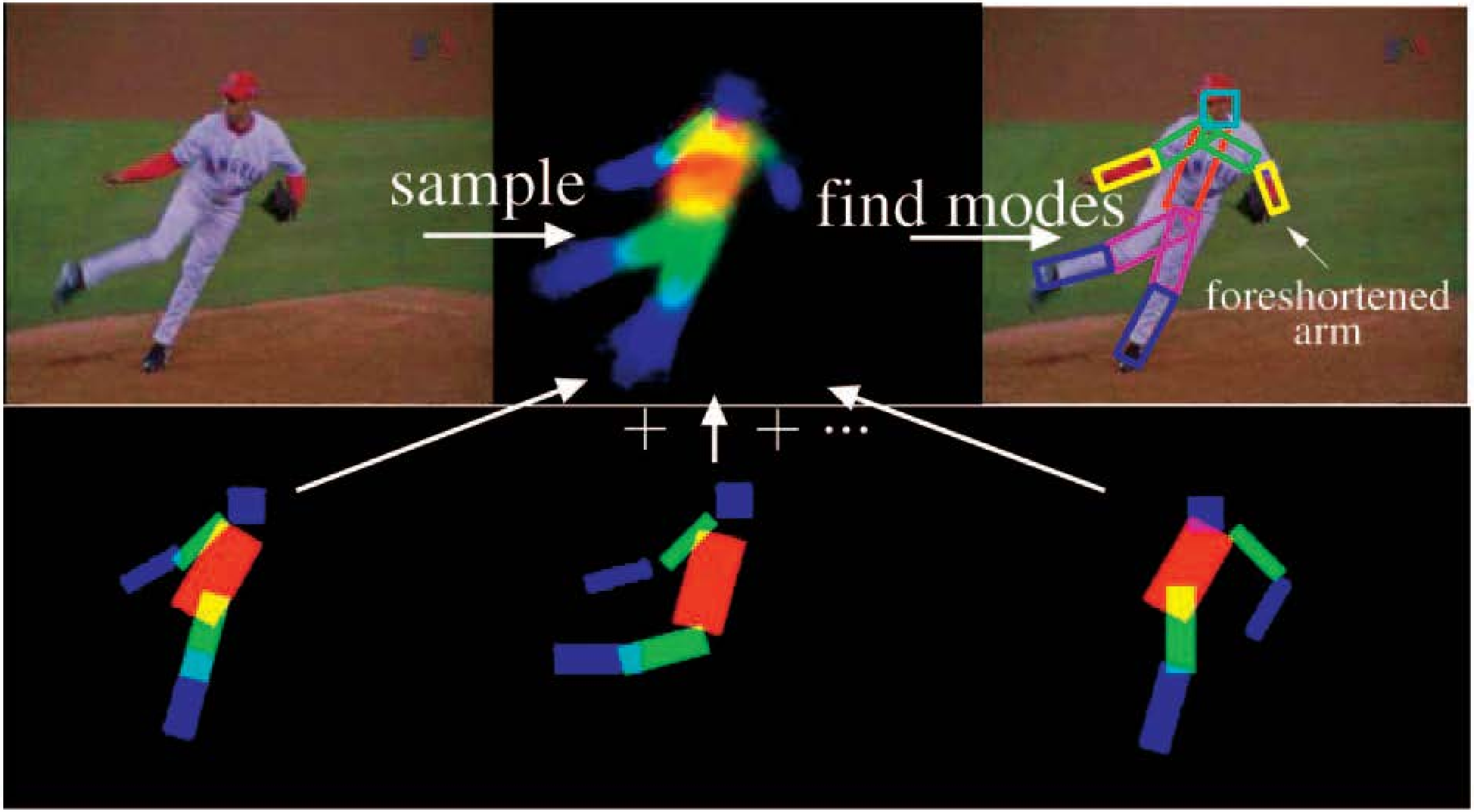
D. Ramanan, D. Forsyth, and A. Zisserman. [Tracking People by Learning their Appearance](). PAMI 2007.

Top-down initialization: Exploit "easy" poses



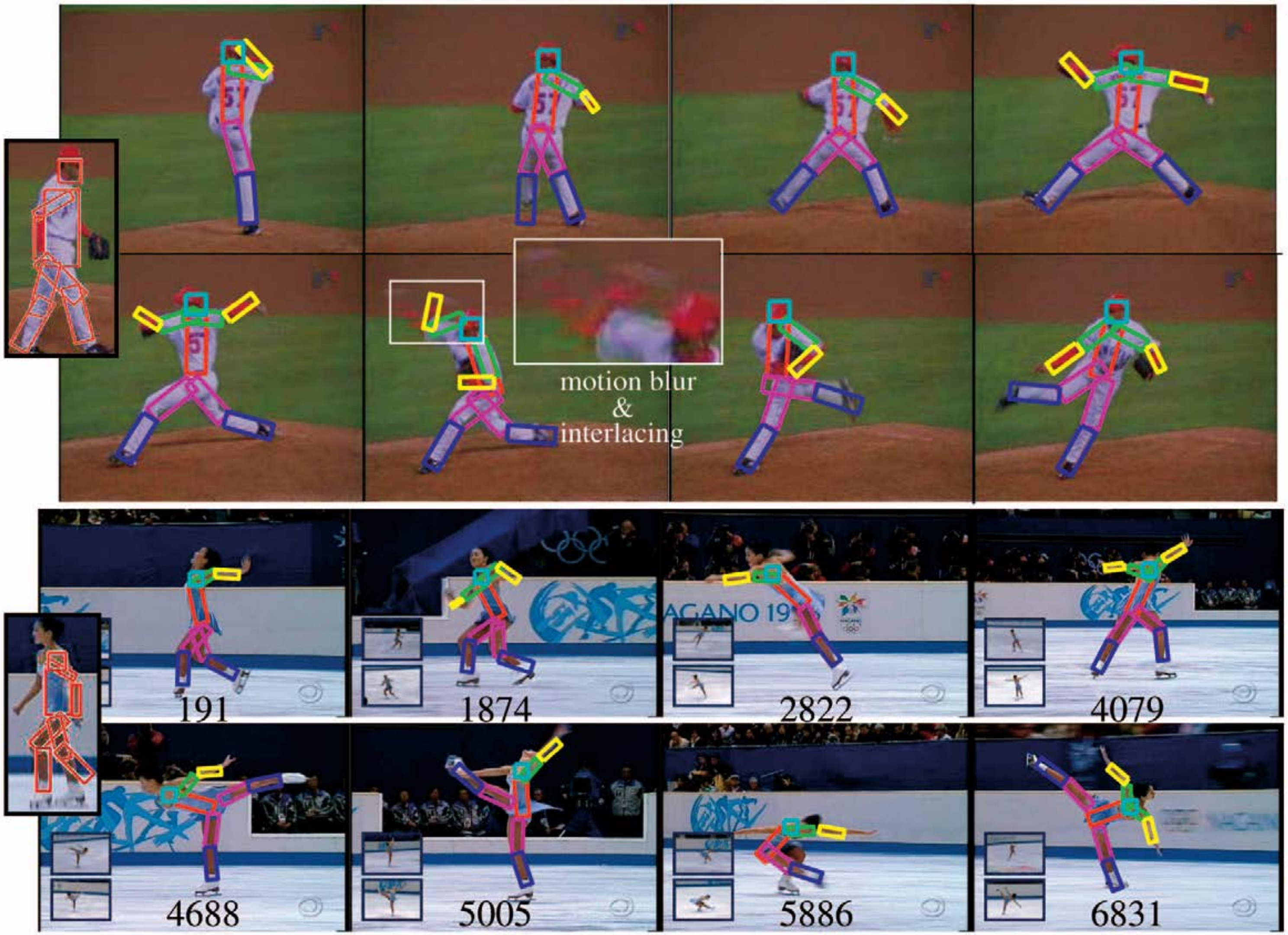
D. Ramanan, D. Forsyth, and A. Zisserman. [Tracking People by Learning their Appearance](). PAMI 2007.

Tracking by model detection



D. Ramanan, D. Forsyth, and A. Zisserman. [Tracking People by Learning their Appearance](). PAMI 2007.

Example results



http://www.ics.uci.edu/~dramanan/papers/pose/index.html