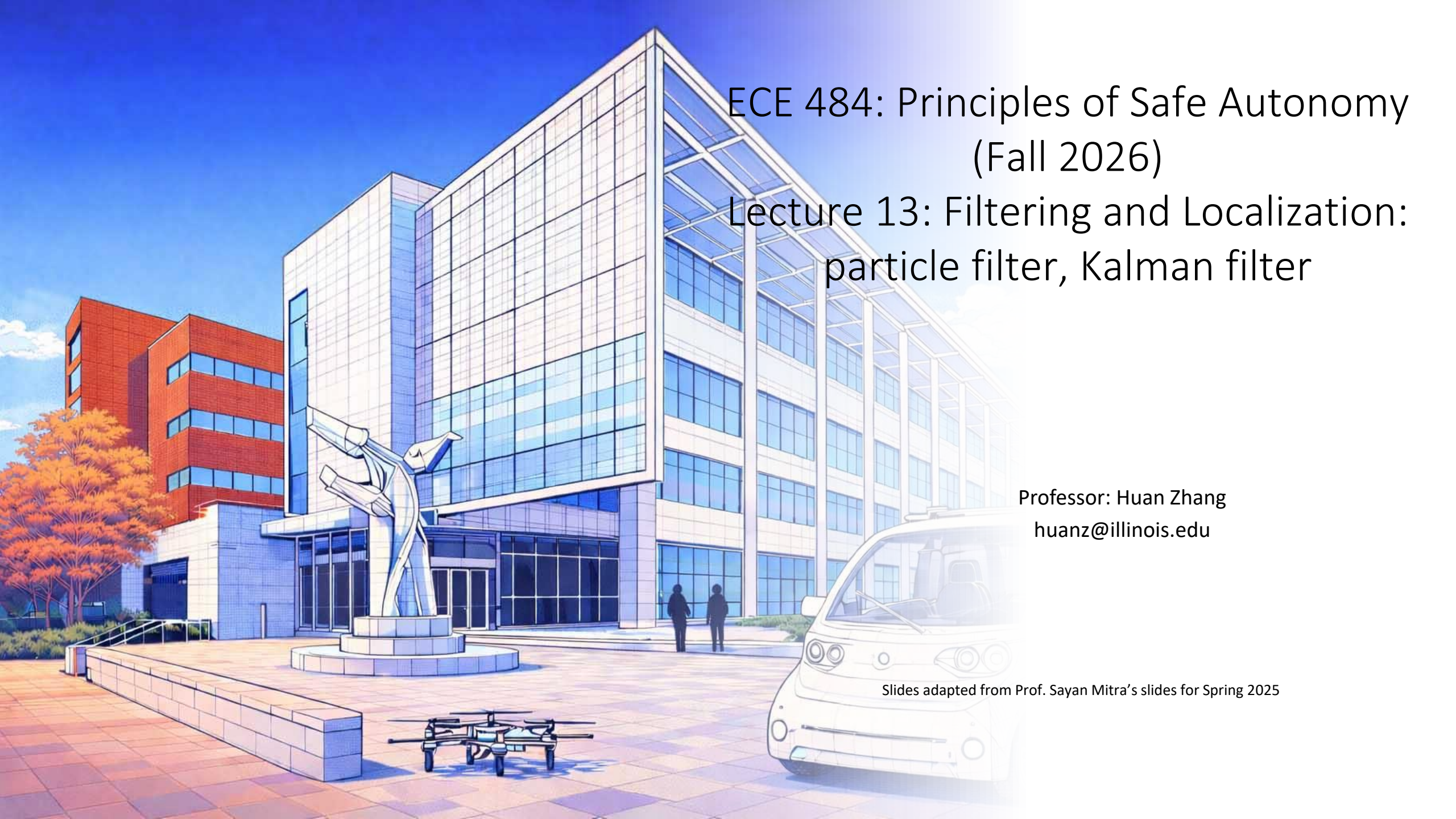


# ECE 484: Principles of Safe Autonomy (Fall 2026)
# Lecture 13: Filtering and Localization: particle filter, Kalman filter

Professor: Huan Zhang
huanz@illinois.edu

Slides adapted from Prof. Sayan Mitra's slides for Spring 2025

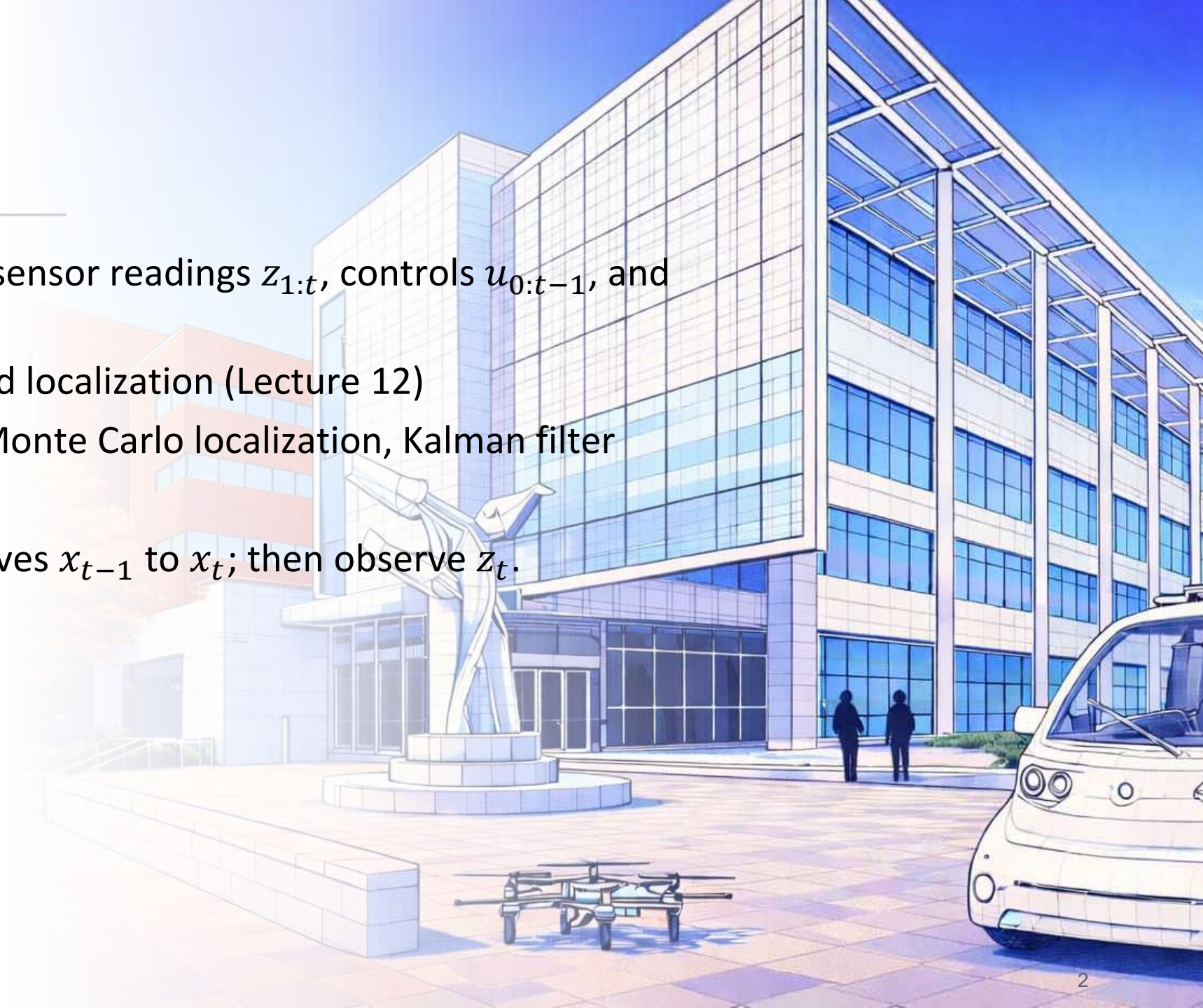
# Outline

Estimate the state $x_t$ from sensor readings $z_{1:t}$, controls $u_{0:t-1}$, and a known map $m$.

**Review:** Bayes filter and grid localization (Lecture 12)

**Today:** particle filters and Monte Carlo localization, Kalman filter

Recall the **Setting**: $u_{t-1}$ moves $x_{t-1}$ to $x_t$; then observe $z_t$.



# Review: Beliefs

**Belief:** a probability distribution over possible states.

The state is not directly measured; infer it from sensor readings.

$$bel(x_t) = p(x_t | z_{1:t}, u_{0:t-1})$$

Posterior from the available history; update in two steps:

Initially: $bel(x_0) = \pi_0$ (e.g., uniform distribution over all locations)

**Prediction:** $\overline{bel}(x_t) = p(x_t | z_{1:t-1}, u_{0:t-1})$ apply motion model

**Correction:** update $\overline{bel}(x_t)$ using the new reading $z_t$ to get $bel(x_t)$ apply measurement model

# Review: Bayes Filter

Bayes_filter$(bel(x_{t-1}), u_{t-1}, z_t)$ returns $bel(x_t)$.

Notation: $Q$: finite state set; $x$: old candidate; $x'$: new candidate.

**Prediction:** $\overline{bel}(x') = \sum_{x \in Q} p_D(x'|x, u_{t-1}) bel(x)$. motion model

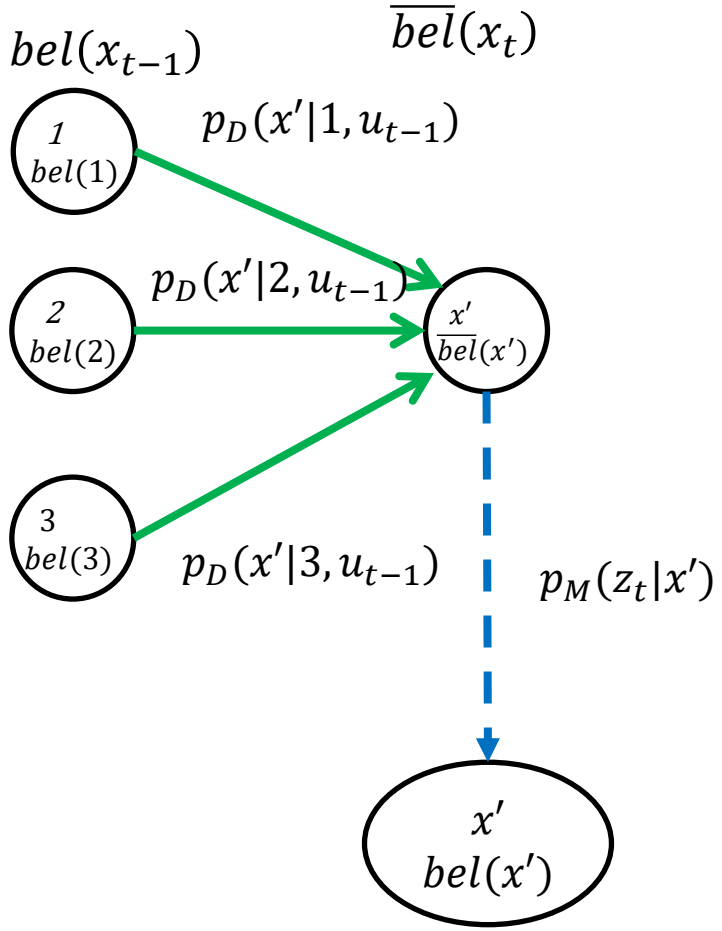
**Intuition**: all previous state $x$ may have a probability to reach $x'$, we must sum them up, weighted by probability $bel(x)$

**Correction:** $bel(x') = \eta p_M(z_t|x') \overline{bel}(x')$. measurement model

Normalization factor: $\eta^{-1} = \sum_{r \in Q} p_M(z_t|r) \overline{bel}(r) > 0$.

**Intuition**: if new observation align with believe, increase belief probability; otherwise decrease.

All old beliefs are at t − 1; predicted/current beliefs are at t.



# Review: Histogram Filter or Discrete Bayes Filter

Define: $Q = \{x_1, \dots, x_K\}$; $p_{k,t} = bel(X_t = x_k)$

**Discrete_Bayes_filter**$(\{p_{k,t-1}\}, u_{t-1}, z_t)$:

For each candidate $k$:

$\overline{p}_{k,t} = \sum_i p_D(x_k|x_i, u_{t-1}) p_{i,t-1}$

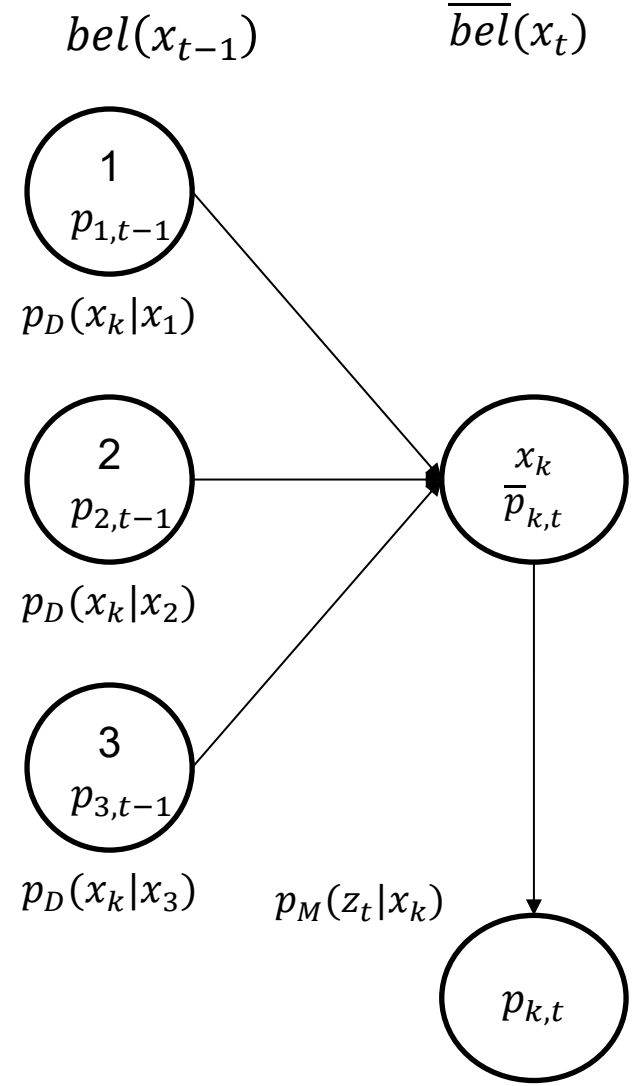
$w_{k,t} = p_M(z_t|x_k)\overline{p}_{k,t}$ **(unnormalized weights)**

End for

Compute $S = \sum_j w_{j,t}$; if $S > 0$, set $\eta = 1/S$. **(Normalization, shared by all)**

Return $p_{k,t} = \eta w_{k,t}$ for every $k$.

Fixed control $u_{t-1}$ in this diagram.



# Review: Grid Localization: Bayes Filter applied on grids

Same Bayes filter; each possible state is now a **pose cell** with discrete position and heading

**Discrete Bayes filter**




One probability per state



**Grid localization**




One heading layer; repeat for each heading bin.

One state = pose cell $C_k$
Center $c_k = (x_B, y_B, \theta_B)$

$p_{k,t}$ is the probability that the robot is in cell $k$

**Predict**

$$\overline{p}_{k,t} = \sum_i p_{i,t-1} T_{ik}(u_{t-1})$$

**Reweight**

$$w_{k,t} = \overline{p}_{k,t} \ell_{k,t}$$

**Normalize**

$$p_{k,t} = w_{k,t} / \sum_j w_{j,t}$$

$T_{ik}$: cell transition probability among the cells (motion model), $\sum_k T_{ik} = 1$
$\ell_{k,t} \approx p_M(z_t | c_k, m)$ (measurement model at the cell center).

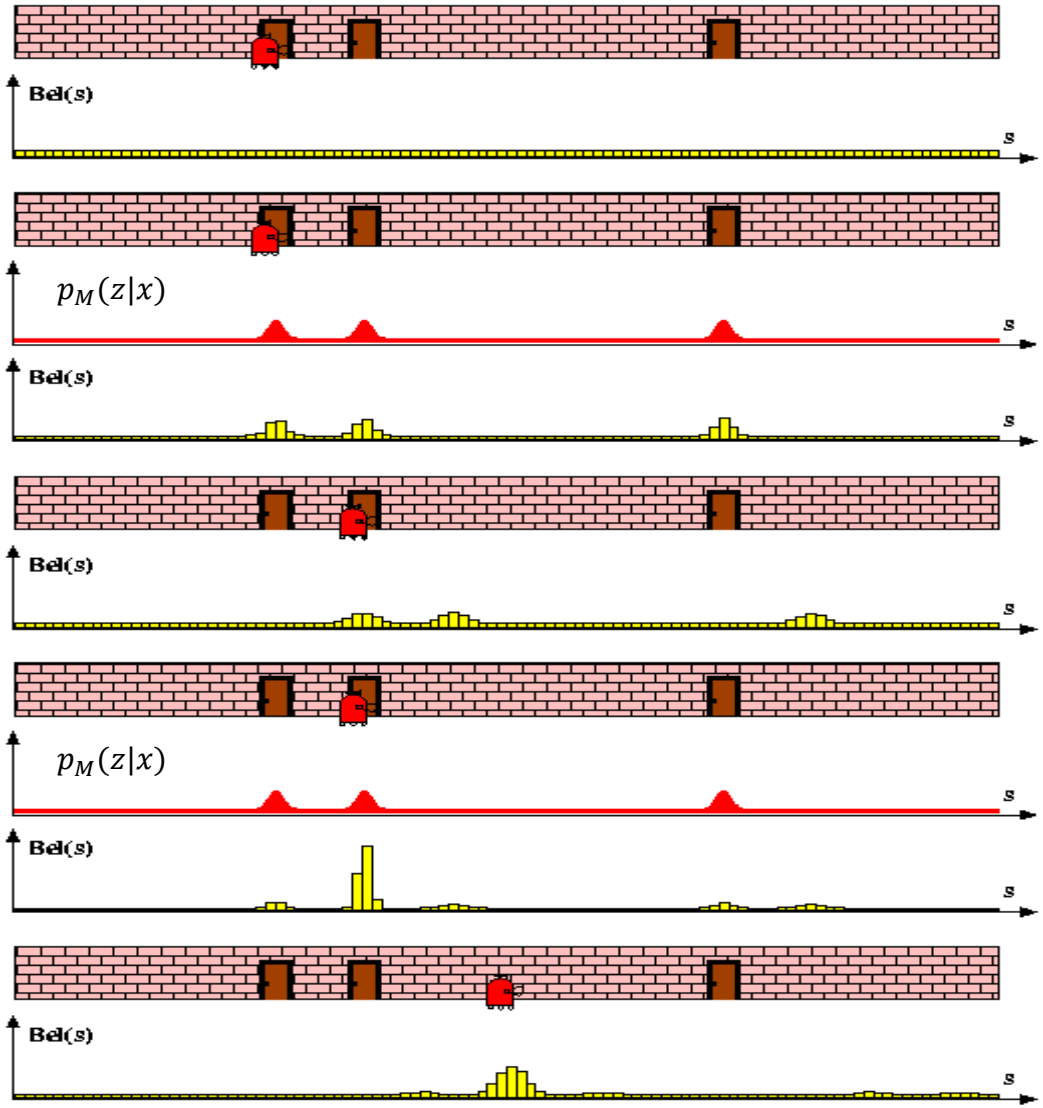
# Grid localization Example: one-dimensional corridor

**Correction:** compare with the sensed doors.

**Prediction:** move all plausible positions.

A second reading can distinguish the earlier alternatives.

Here $s$ is position along the corridor.




Initial

Correction

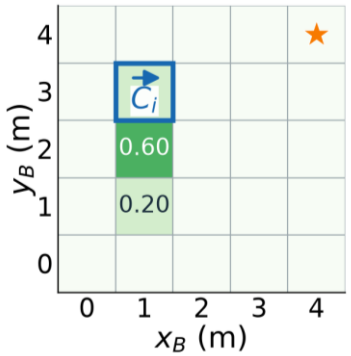
Prediction

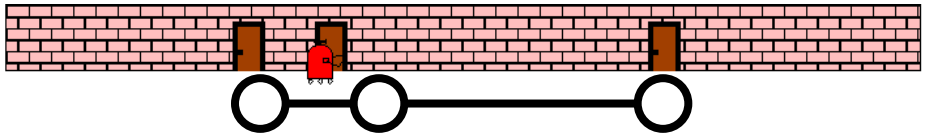
Correction

Prediction

Try it yourself: https://colab.research.google.com/drive/1YyBCjQeIQQDxzZ-crB3Ss_KM-qA2QF7N?usp=sharing

# Grid localization: practical designs



**Key choice: grid resolution.**

**Topological:** group meaningful places. **Metric:** partition pose into fine cells.

A finer grid can improve approximation, with more memory and computation.

**Coarse cells:** should not discard small motions by repeatedly rounding to a cell center.

Example: 10 cm/s × 1 s = 0.1 m, only one tenth of a 1 m cell.

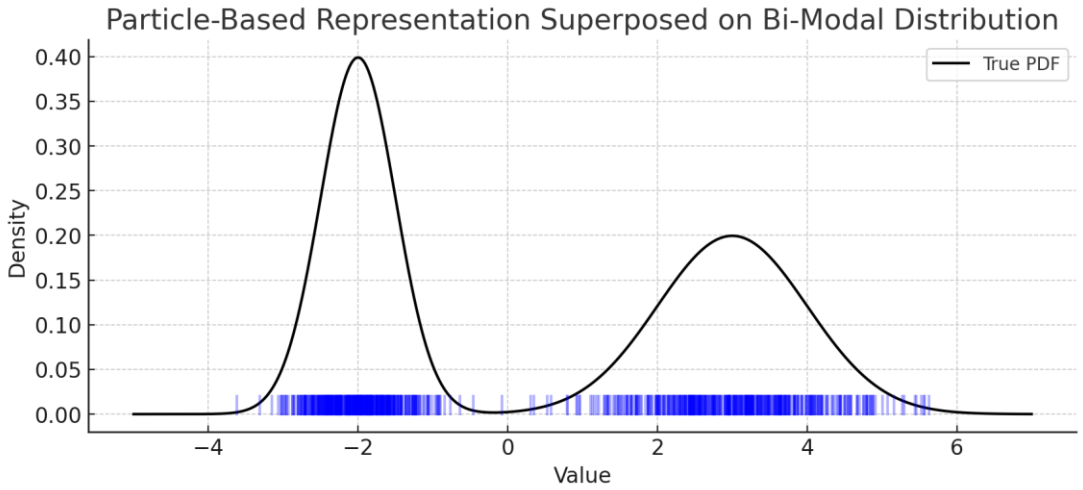
**Computation:** direct 3-D pose prediction sums over source/destination pairs: six coordinate loops.

With $K$ cells: dense prediction $O(K^2)$; measurement update $O(K)$.

Fine grids can be costly; sparse transitions and caching (e.g., ray-cast ranges) help.

Today: can we be “smarter” than using a grid?

# Today: Particle Filters




- Belief = a **finite set of particles**.

Each particle represents one possible location (position, heading, etc)

The density of particles reflects the probability distribution of the belief

- Advantages
  - Avoid the cost of the grid; no need to divide the space
  - Also allows **nonparametric** representation of the belief, and therefore can represent a broader set of distributions e.g., bimodal distributions
  - Can handle nonlinear transformations, e.g., under motion and measurements
- Related ideas: Monte Carlo filter, Survival of the fittest, Condensation, Bootstrap filter, Filtering: [Rubin, 88], [Gordon '93], [Kitagawa 96], Dynamic Bayesian Networks: [Kanazawa '95]

# Particle filtering algorithm

Initially: we have $M$ particles $\{x_t^{[1]}, \dots, x_t^{[M]}\}$

**Particle_filter**$(\mathcal{X}_{t-1}, u_{t-1}, z_t)$:

$\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$

For $i = 1, \dots, M$:

Draw $x_t^{[i]} \sim p_D(x_t | x_{t-1}^{[i]}, u_{t-1})$. **Move particles with motion model**

$w_t^{[i]} = p_M(z_t | x_t^{[i]}, m)$ **Reweight with measurement model**

Add $(x_t^{[i]}, w_t^{[i]})$ to $\overline{\mathcal{X}}_t$.

Draw $M$ indices **with replacement**:

sample particle $j$ with prob. $P(j) = w_t^{[j]} / \sum_k w_t^{[k]}$

Copy the sampled $x_t^{[j]}$ into $\mathcal{X}_t$

Return $\mathcal{X}_t$.

**Predict:** sample candidates using the motion model $p_D$.

**Weight:** evaluate how well each candidate explains $z_t$.

$\overline{\mathcal{X}}_t$: temporary candidates and raw weights.

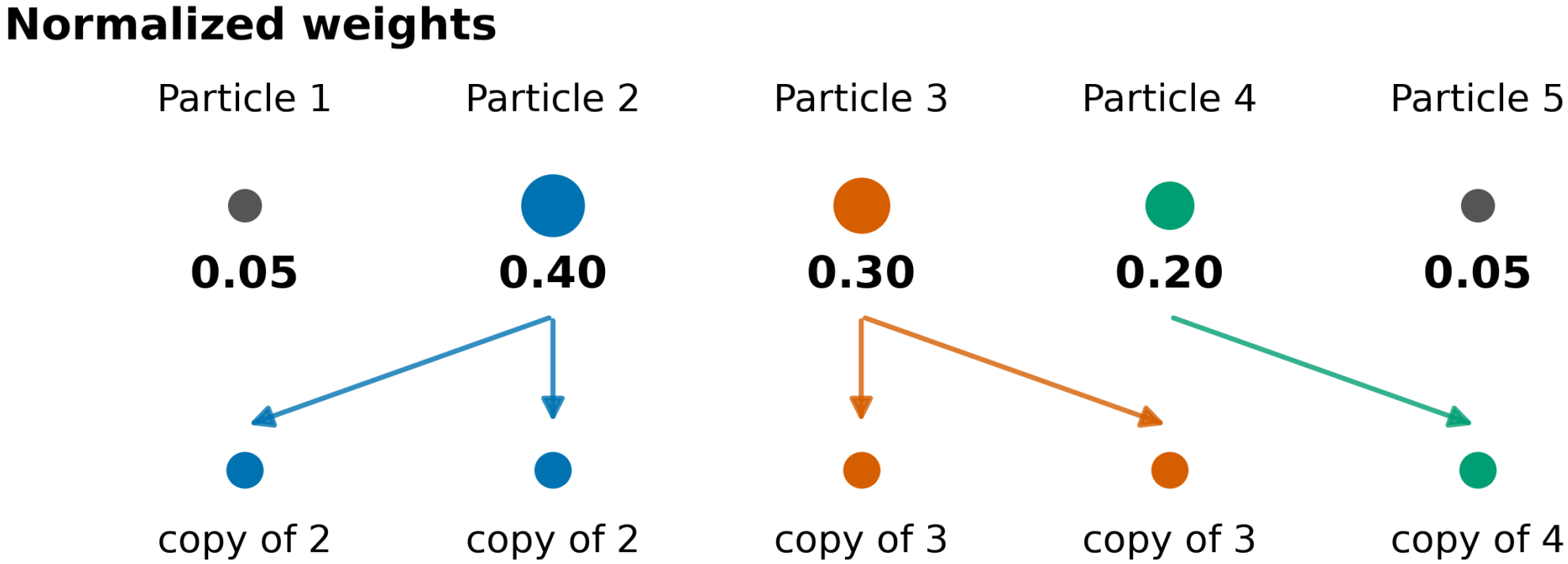
Before weighting, candidates approximate $\overline{bel}(x_t)$.

After weighting/resampling, candidates approximate $bel(x_t)$.

**Resample:** likely candidates can get multiple copies.

# Resampling: copies carry the probabilities

Five candidates: raw weights $0.10, 0.80, 0.60, 0.40, 0.10$ (sum $2.00$).
Normalize, then draw $M = 5$ indices **with replacement**.




**Question:** can resampling invent a pose that was absent?

# Monte Carlo Localization (MCL): particle filter for localization

Initially: we have $M$ particles $\{x_t^{[1]}, \dots, x_t^{[M]}\}$

**MCL**$(\mathcal{X}_{t-1}, u_{t-1}, z_t, m)$:
$\overline{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
For $i = 1, \dots, M$:
$x_t^{[i]} = \text{sample_motion_model}(u_{t-1}, x_{t-1}^{[i]})$
$w_t^{[i]} = \text{measurement_model}(z_t, x_t^{[i]}, m)$
Add $(x_t^{[i]}, w_t^{[i]})$ to $\overline{\mathcal{X}}_t$.
Draw $M$ indices **with replacement**:
$P(j) = w_t^{[j]} / \sum_k w_t^{[k]}$
Copy $x_t^{[j]}$ into $\mathcal{X}_t$.
Reset weights to $1/M$; return $\mathcal{X}_t$.

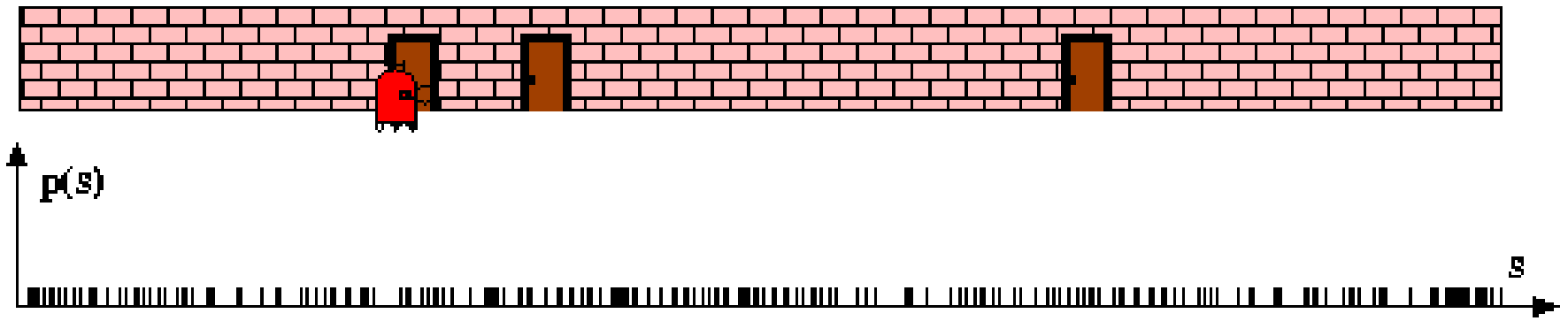
MCL = particle filtering for robot pose in a **known map**.

**Motion model:** draw a possible next pose.

**Measurement model:** score that pose against the observed sensor reading.

$m$: map; $M$: particle count; $[i]$: sample index.

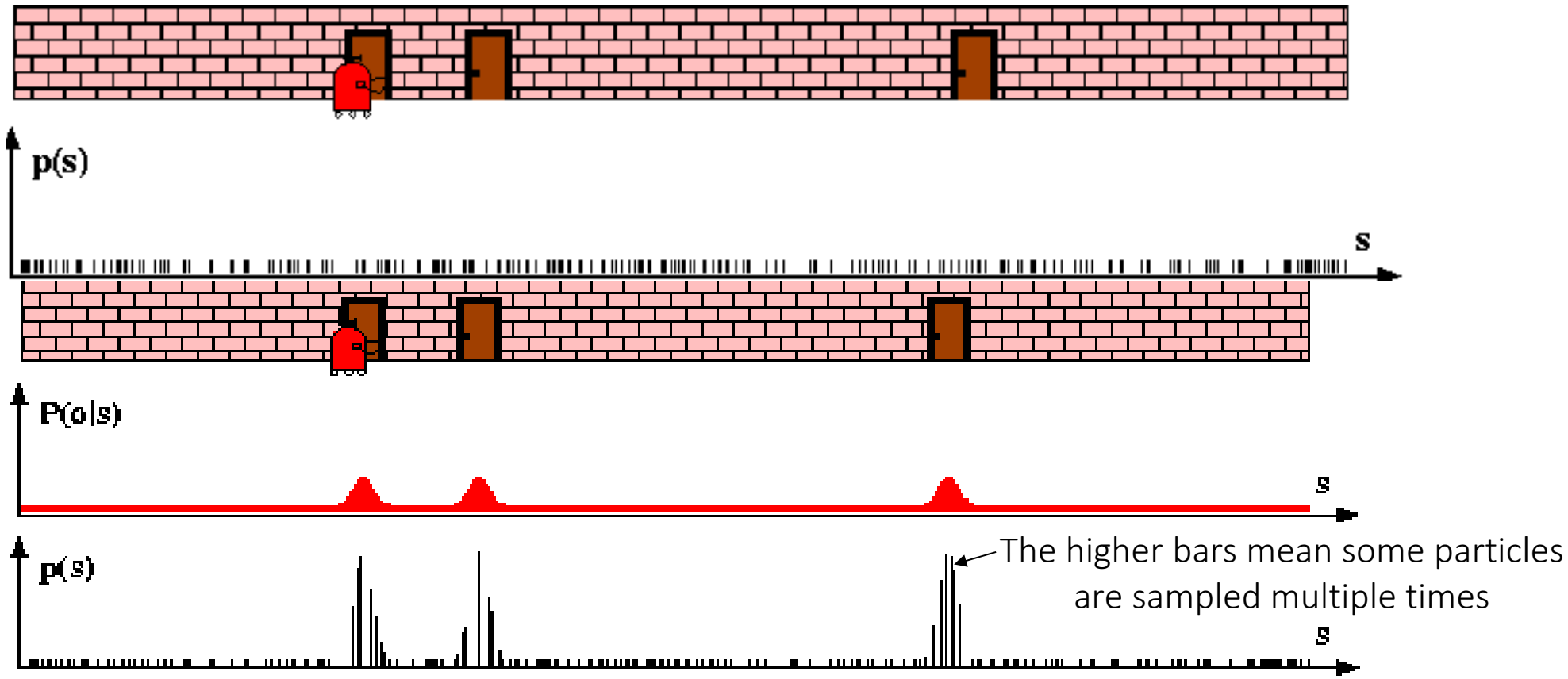
# Particle Filters




Initially, particles are randomly distributed over all locations

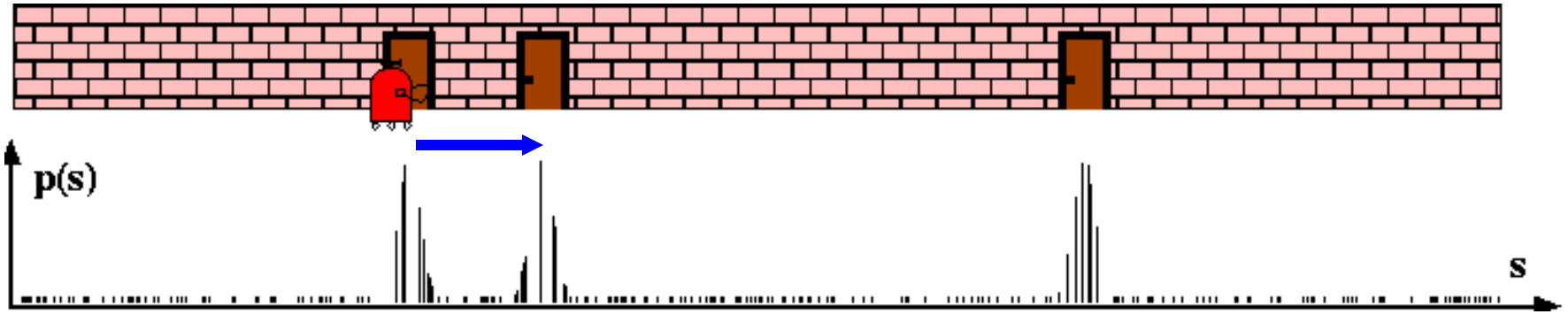
# Sensor Information

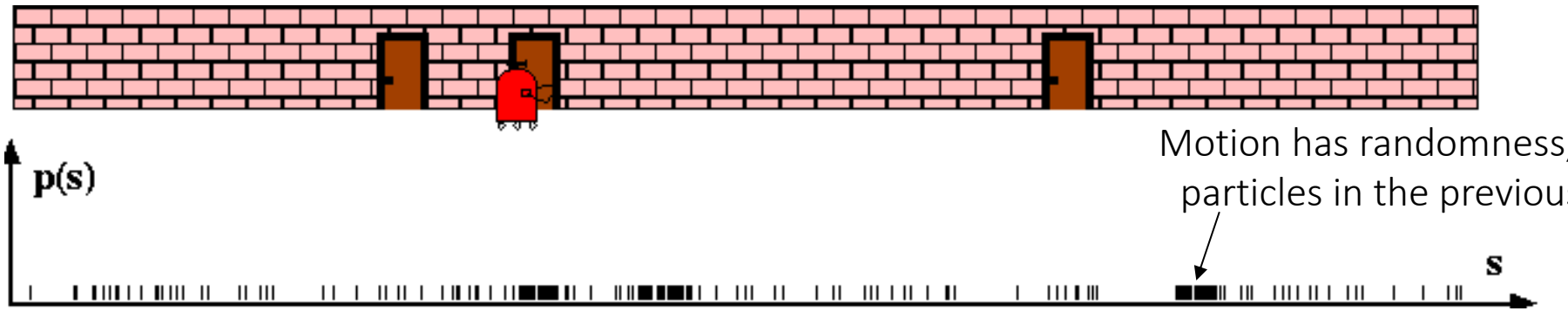
Resample the particles based on the measurement model:
particles near the doors have a higher chance of being sampled



# Robot Motion

Move every particle by one step using the motion model

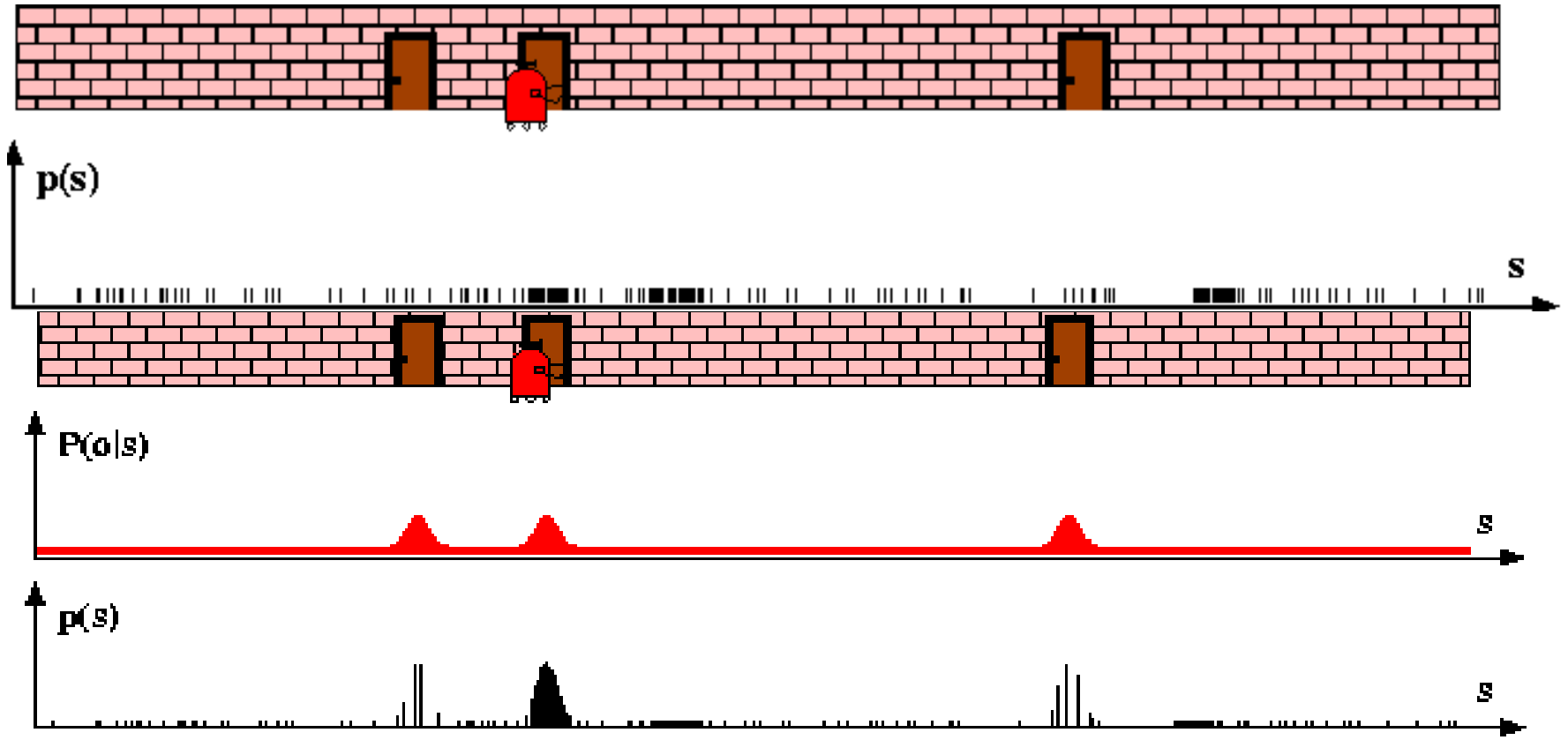



Motion has randomness, so the duplicated particles in the previous step spread out

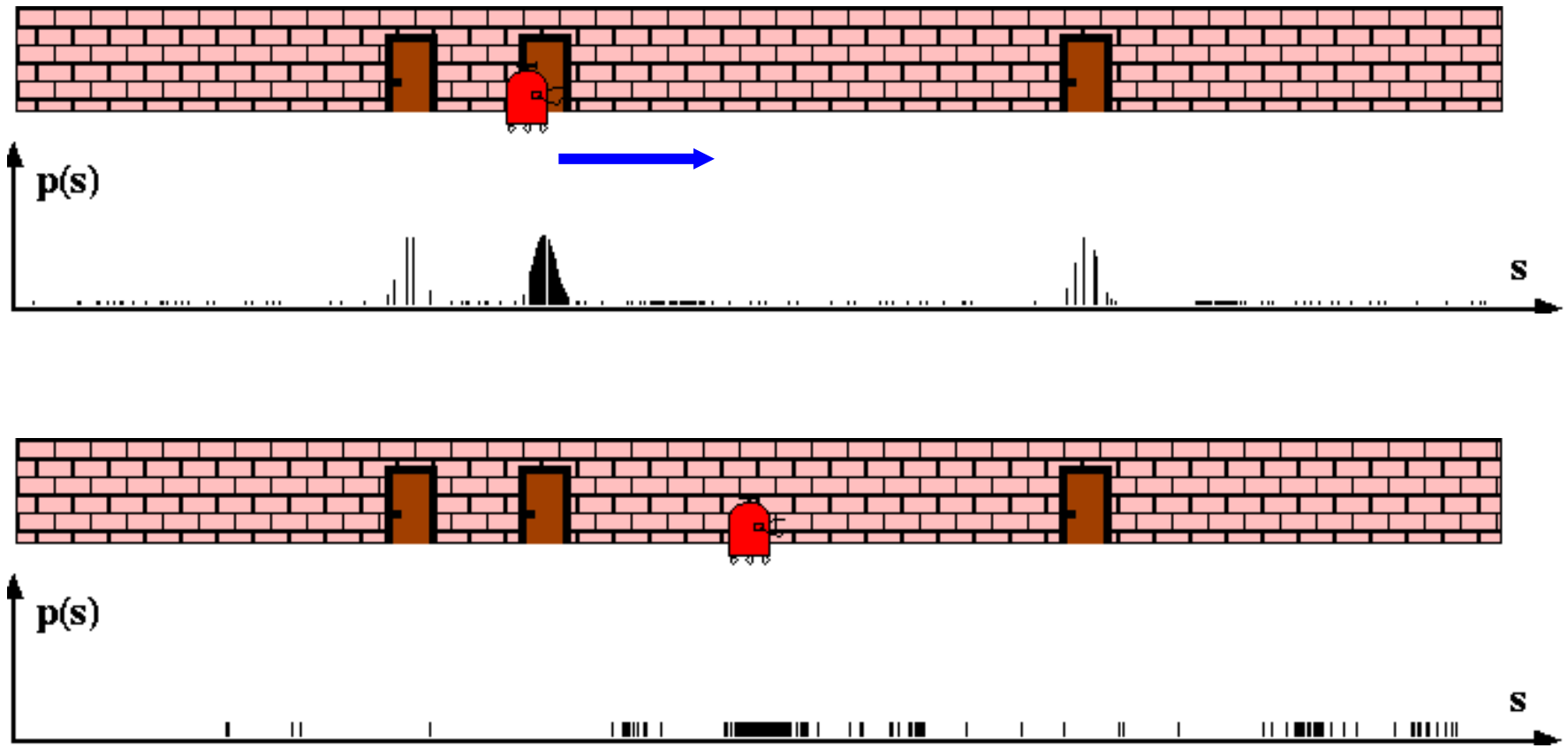
# Sensor Information

Resample the particles based on the measurement model:
particles near the doors have a higher chance of being sampled

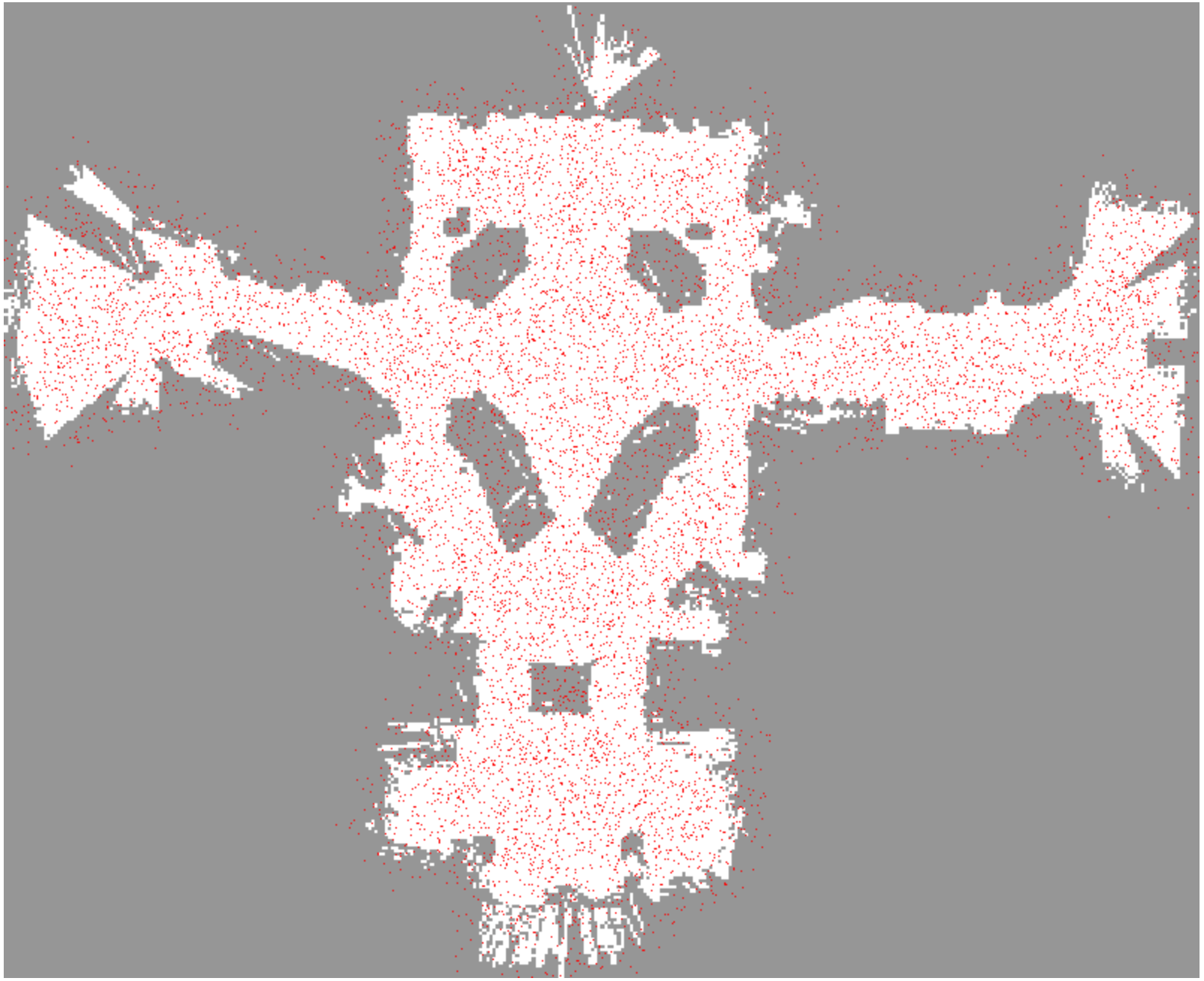


# Robot Motion

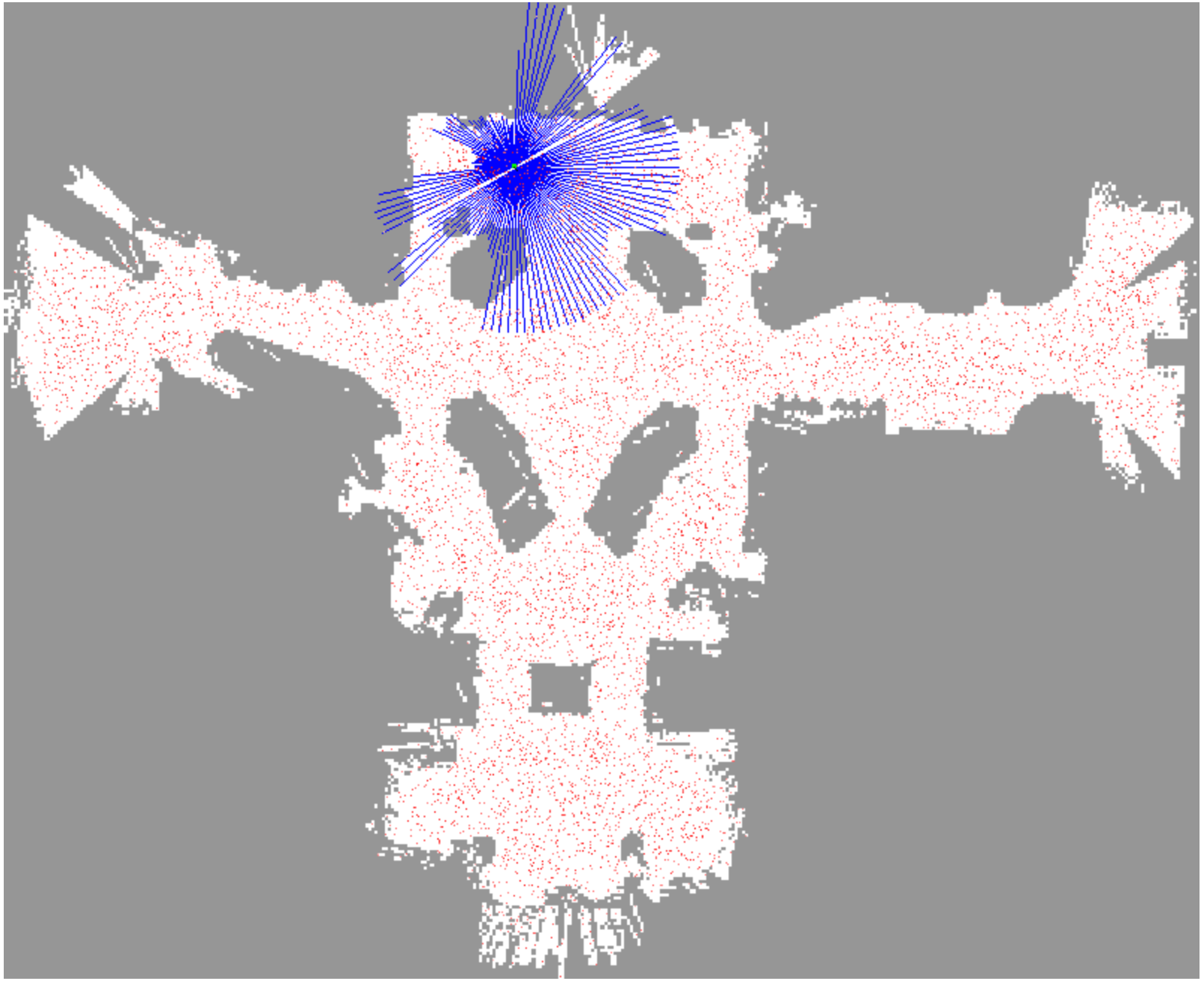
Move every particle by one step using the motion model



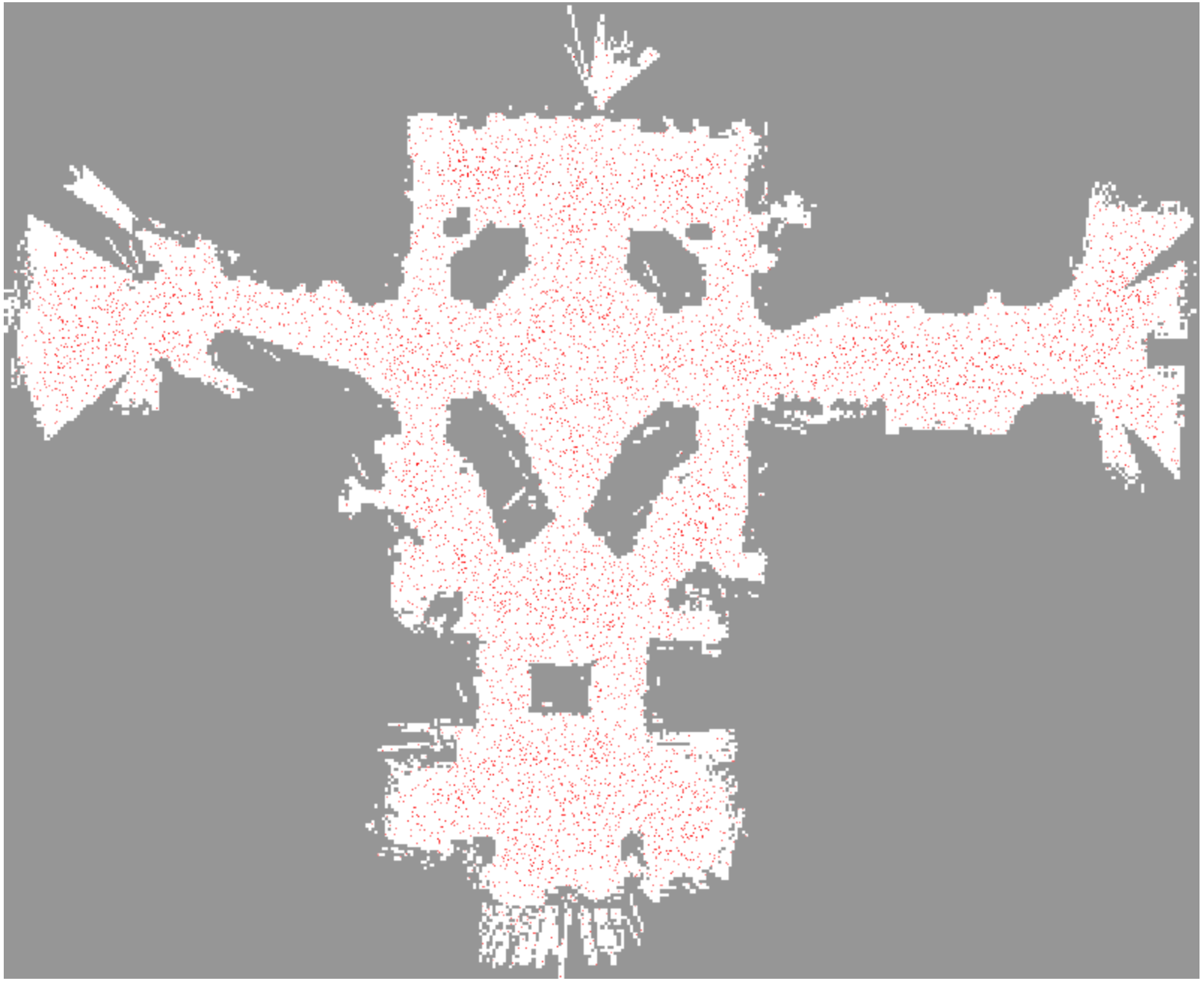
**Global localization**
Particles cover many places.
Red: candidate poses
Blue rays: current scan



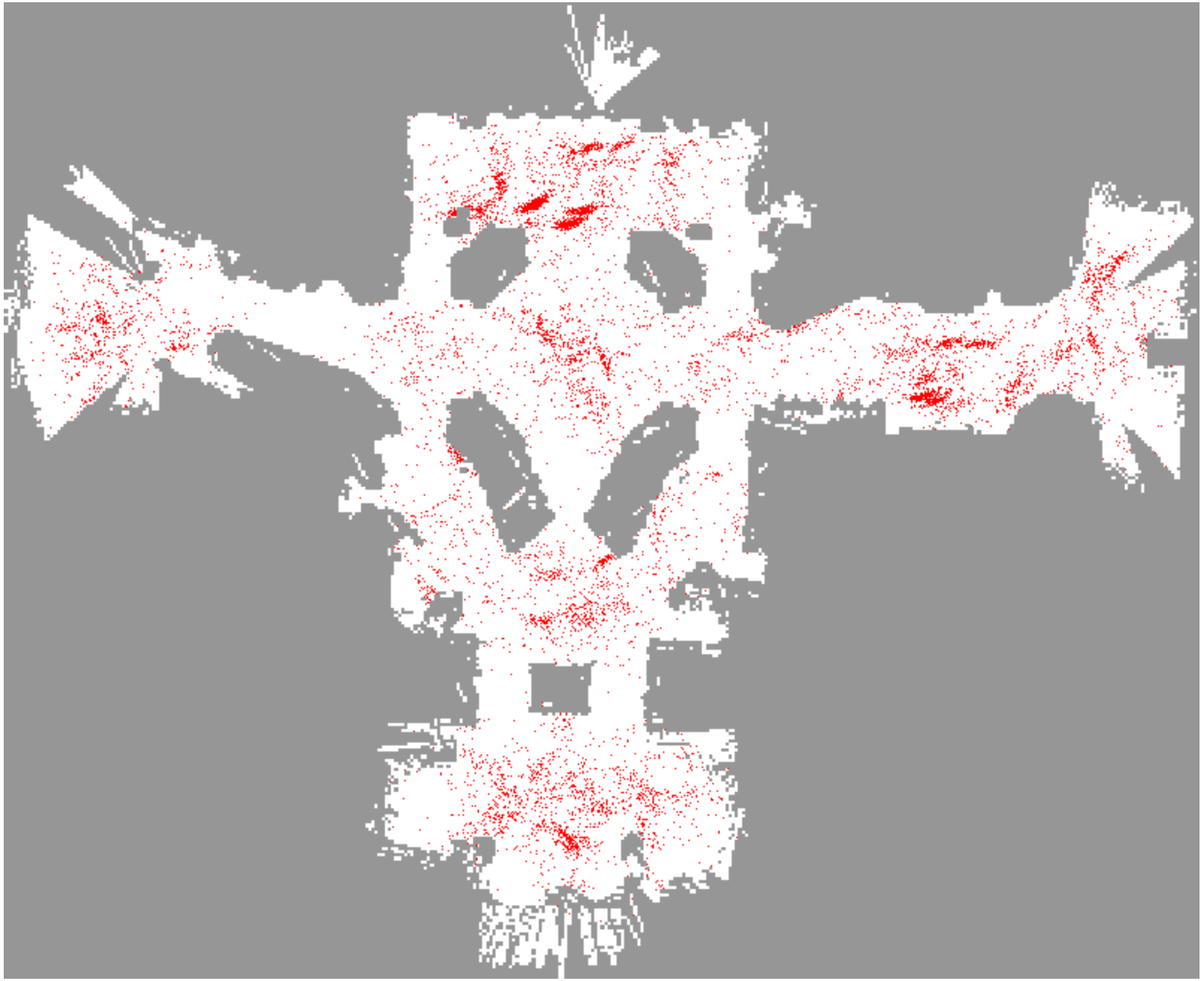
**Laser measurement**
Blue rays are the current scan.
Red: candidate poses
Blue rays: current scan



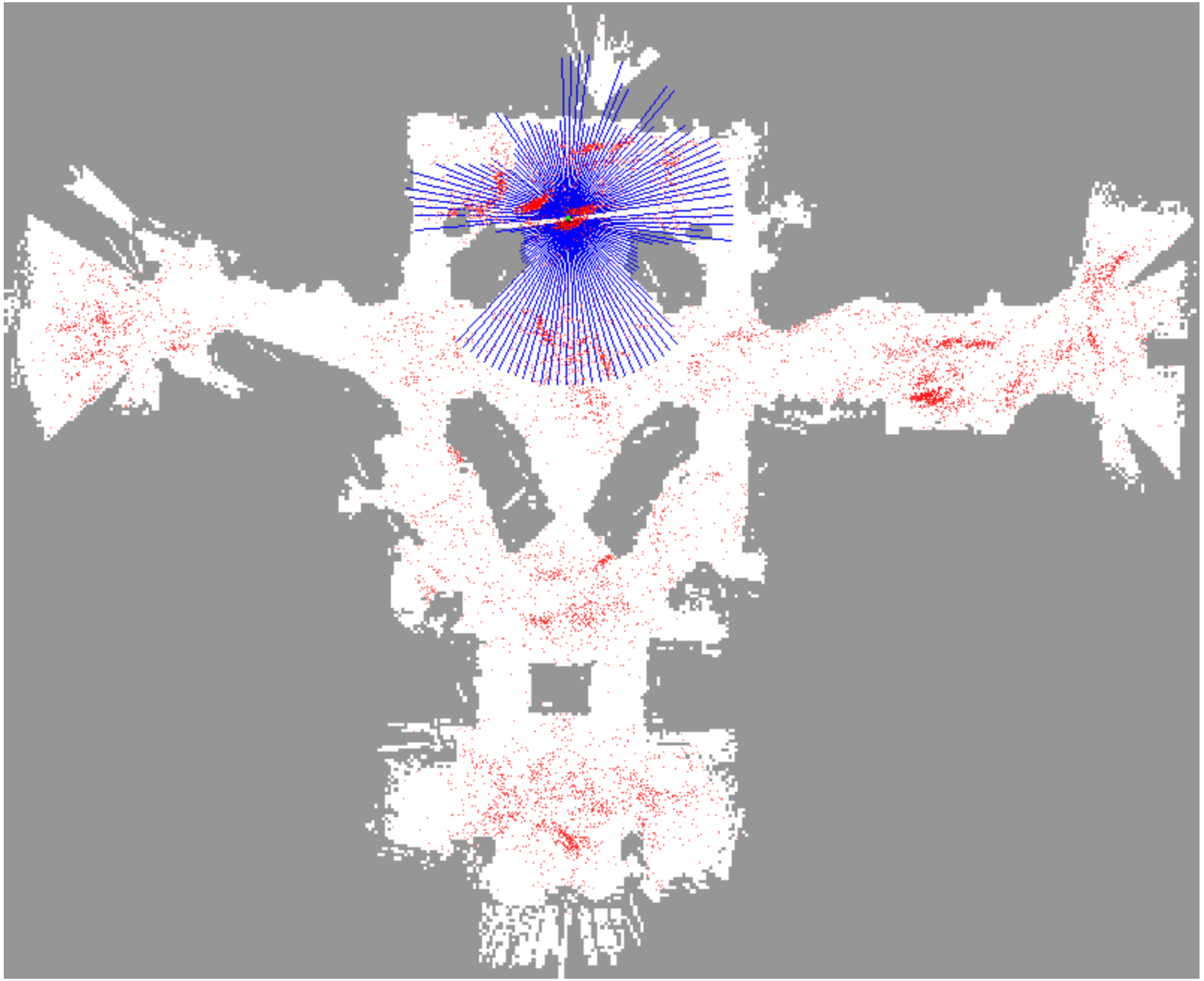
**Weight candidates**
Compare scan with each pose.
Red: candidate poses
Blue rays: current scan



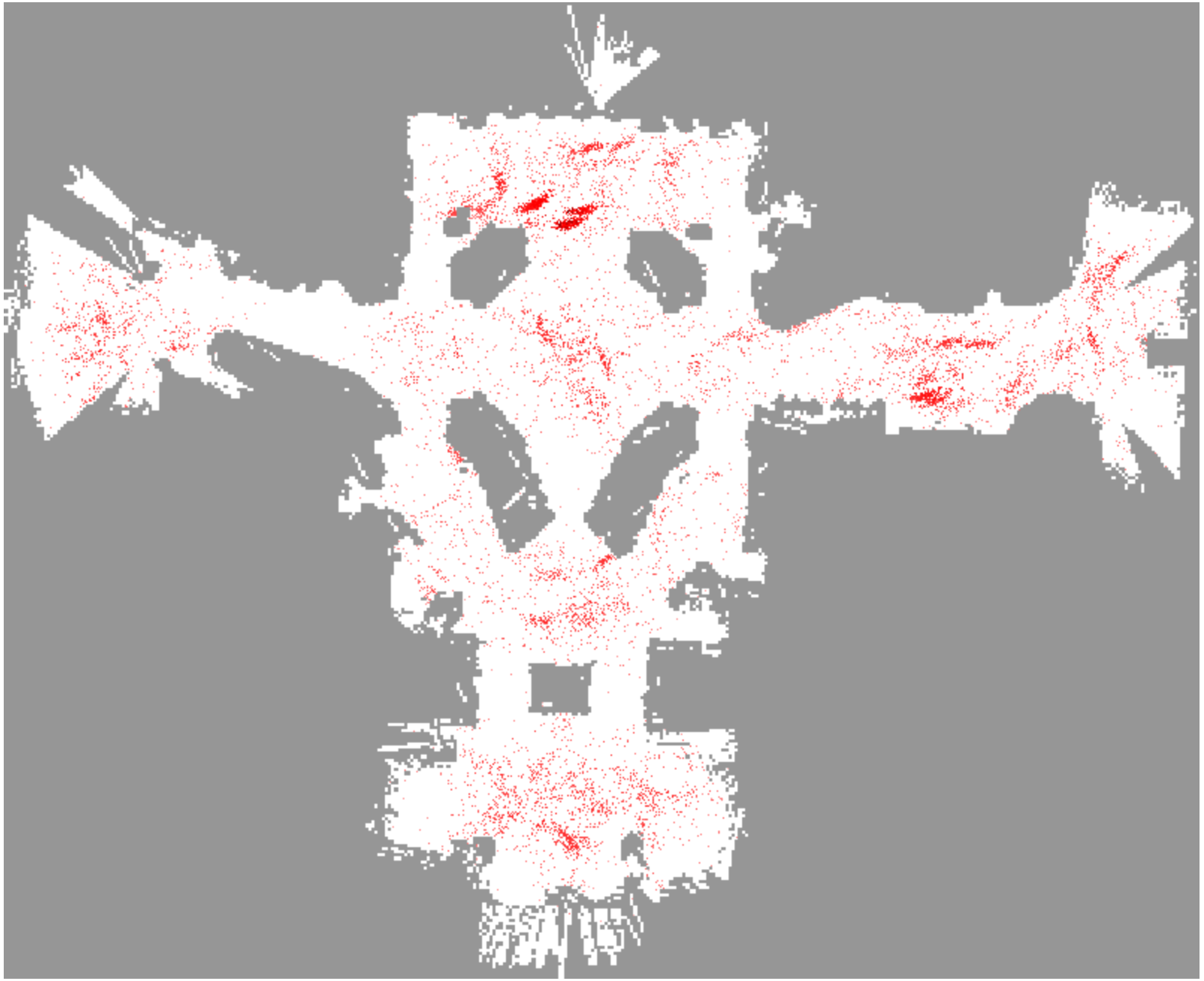
**Resample + move**
Plausible regions get more copies.
Red: candidate poses
Blue rays: current scan



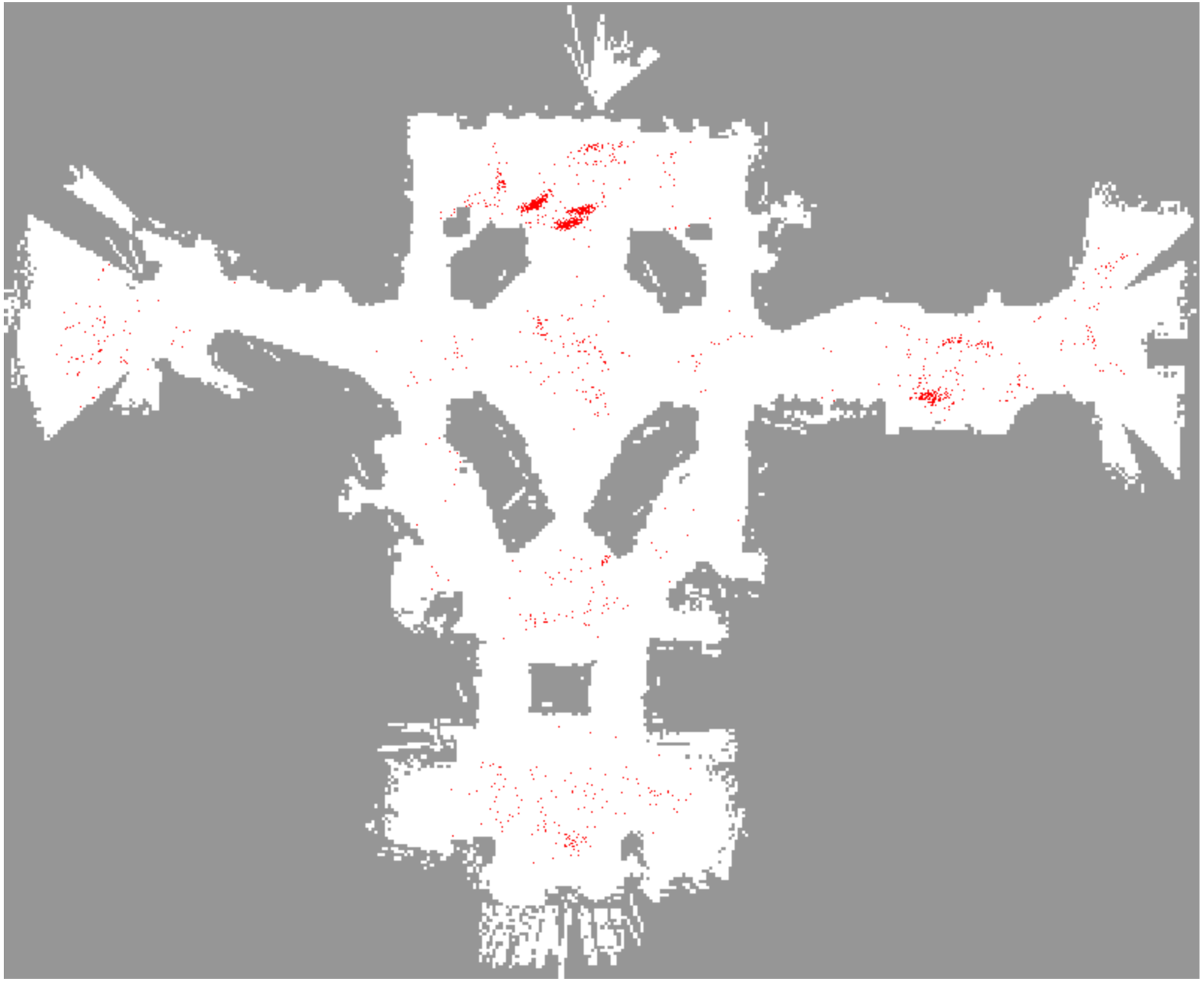
**Next laser scan**
Motion links the observations.
Red: candidate poses
Blue rays: current scan



**Update the weights**
Repeated geometry is still ambiguous.
Red: candidate poses
Blue rays: current scan



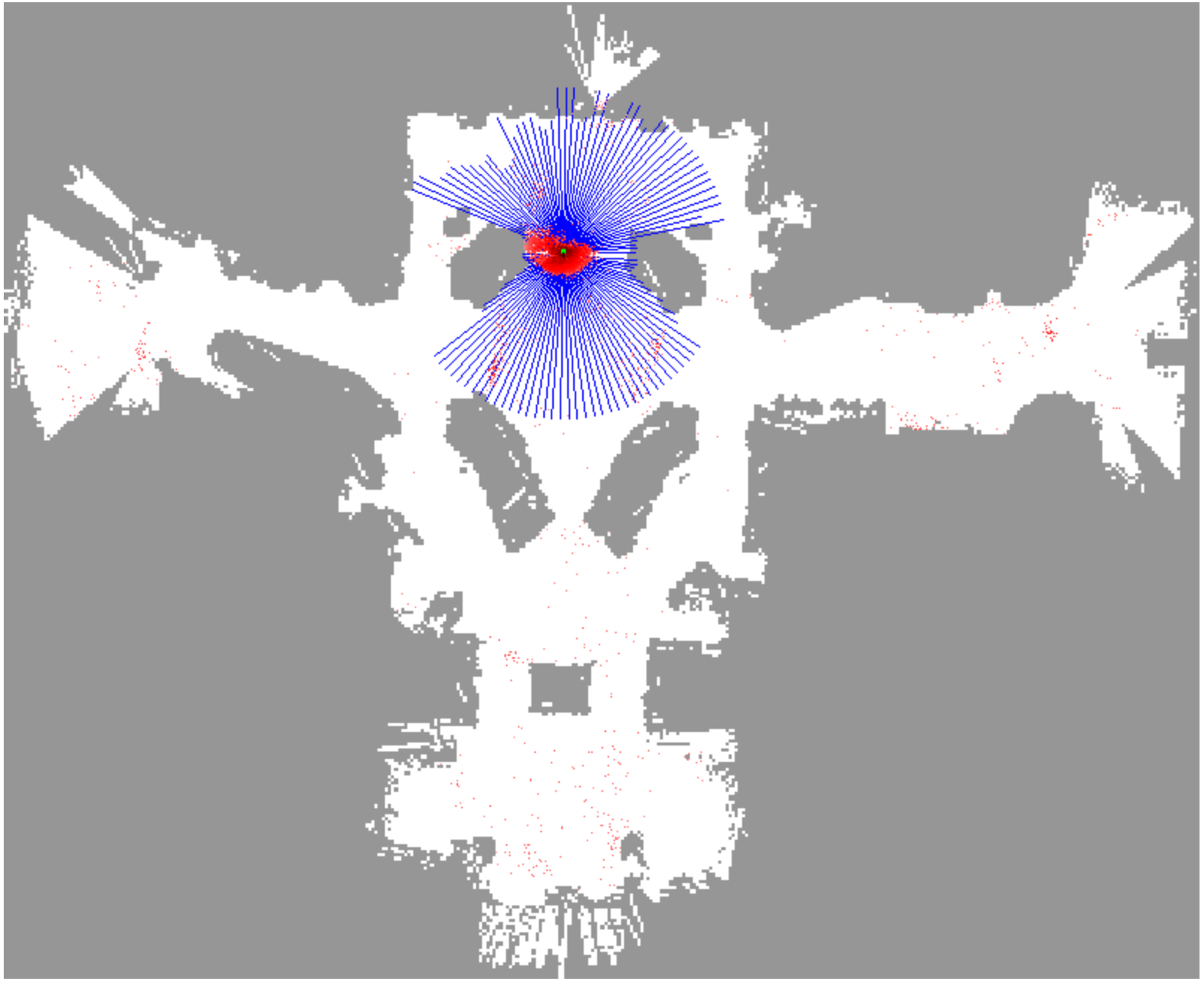
**Fewer hypotheses**
The particle cloud begins to contract.
Red: candidate poses
Blue rays: current scan



**Further contraction**
Several regions still survive.
Red: candidate poses
Blue rays: current scan



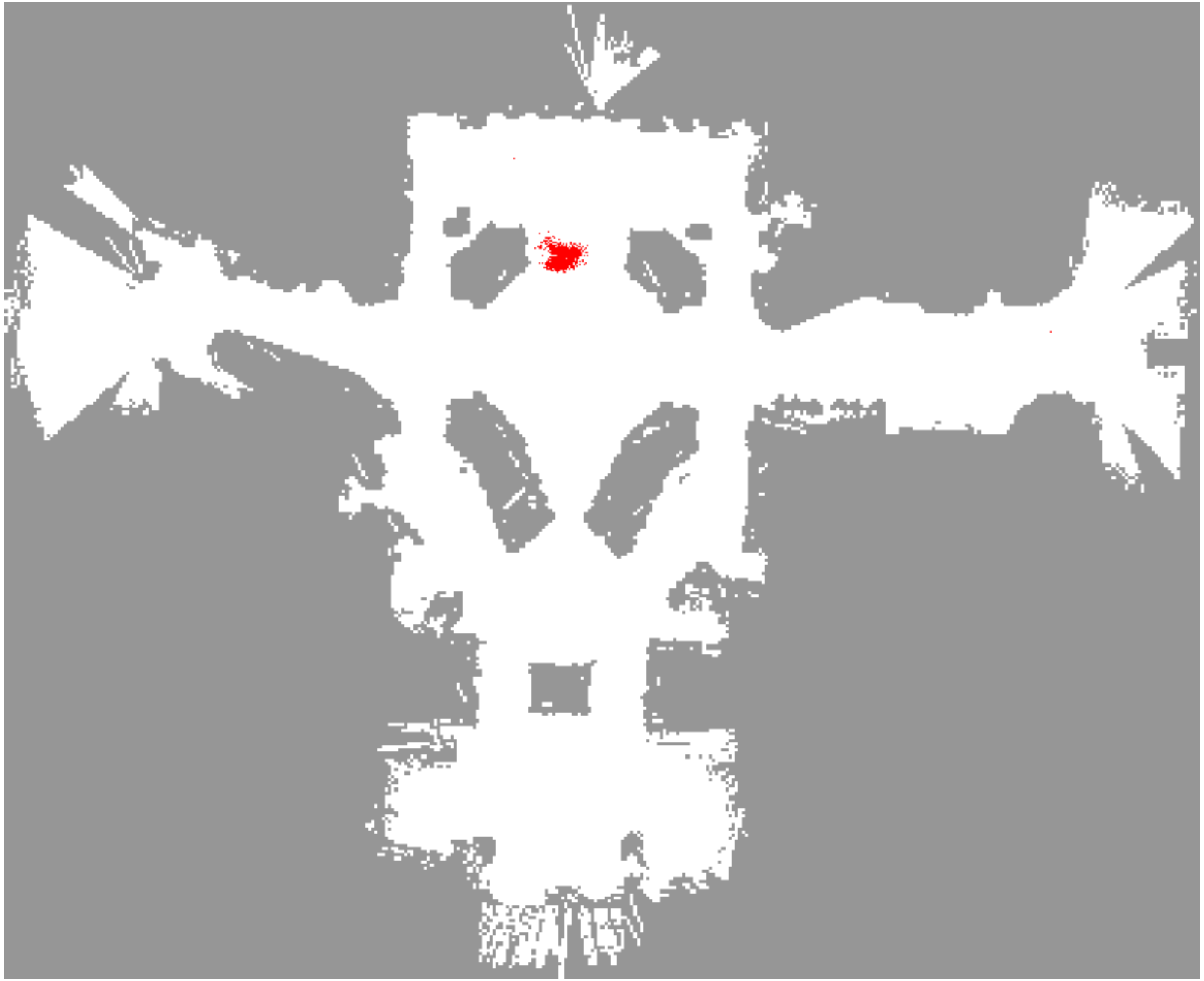
**Another scan**
Test the remaining hypotheses.
Red: candidate poses
Blue rays: current scan



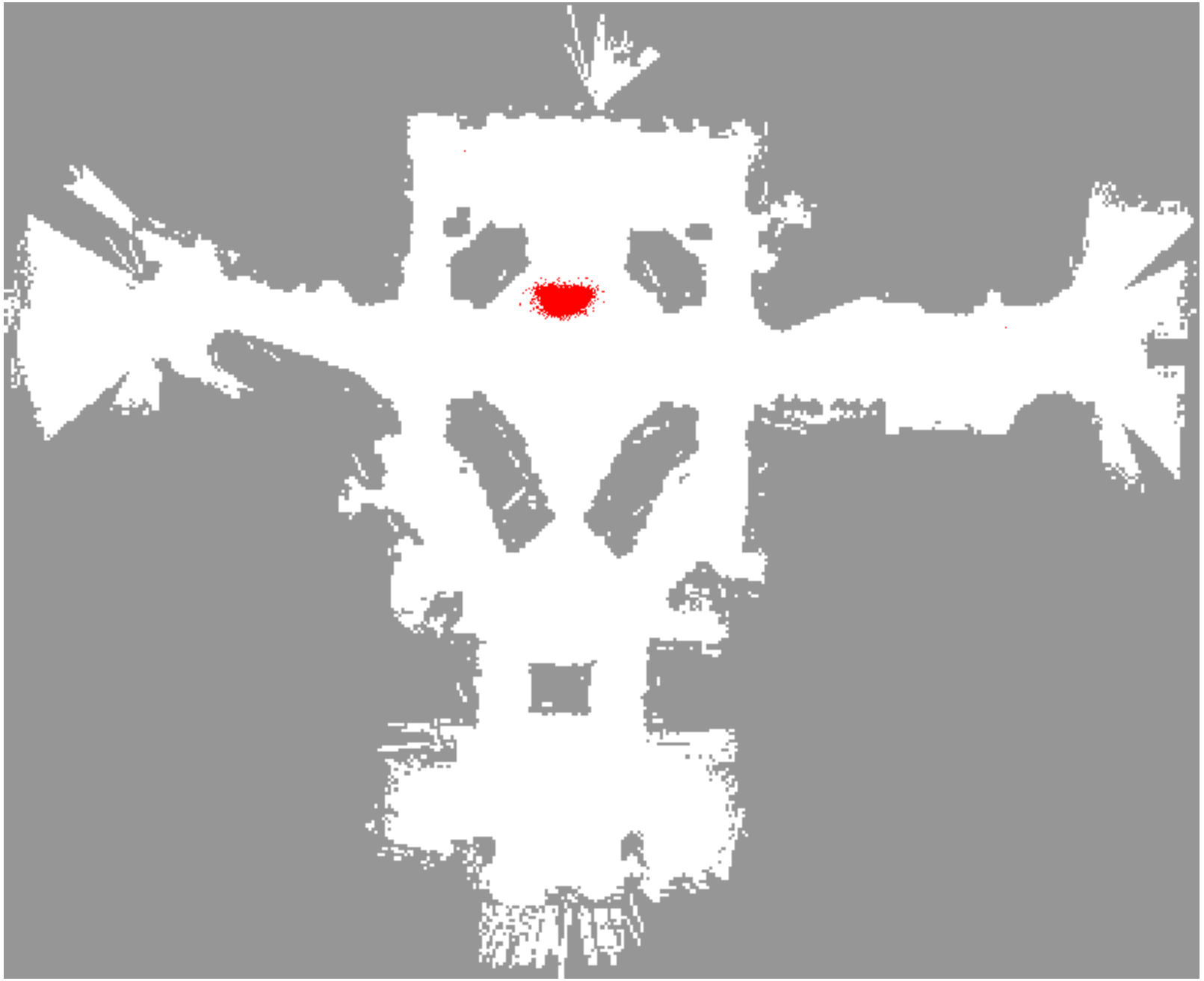
**Another weighting**
Weighting does not move particles.
Red: candidate poses
Blue rays: current scan



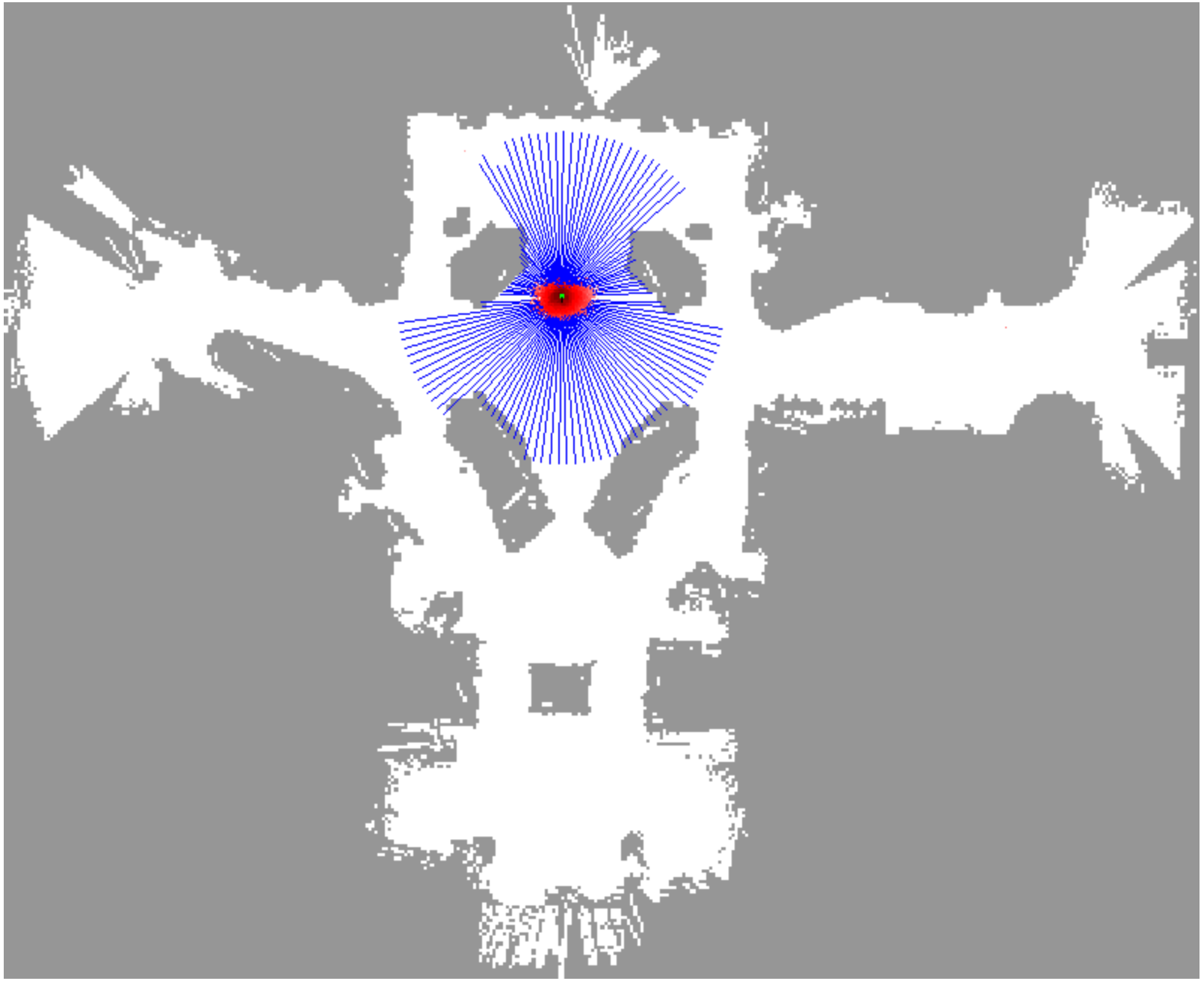
**One dominant region**
The cloud becomes localized.
Red: candidate poses
Blue rays: current scan



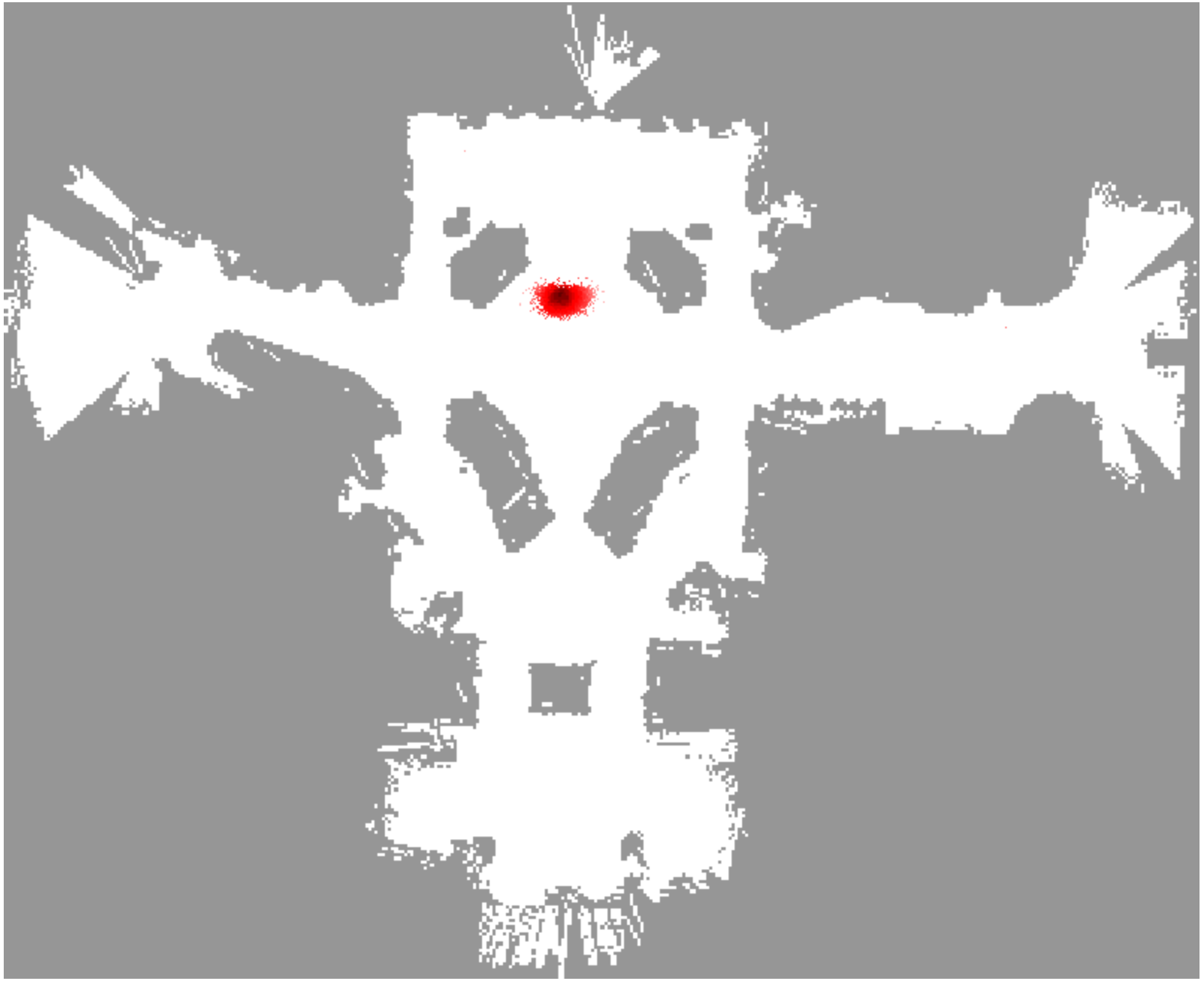
**Start tracking**
Maintain a distribution near that pose.
Red: candidate poses
Blue rays: current scan



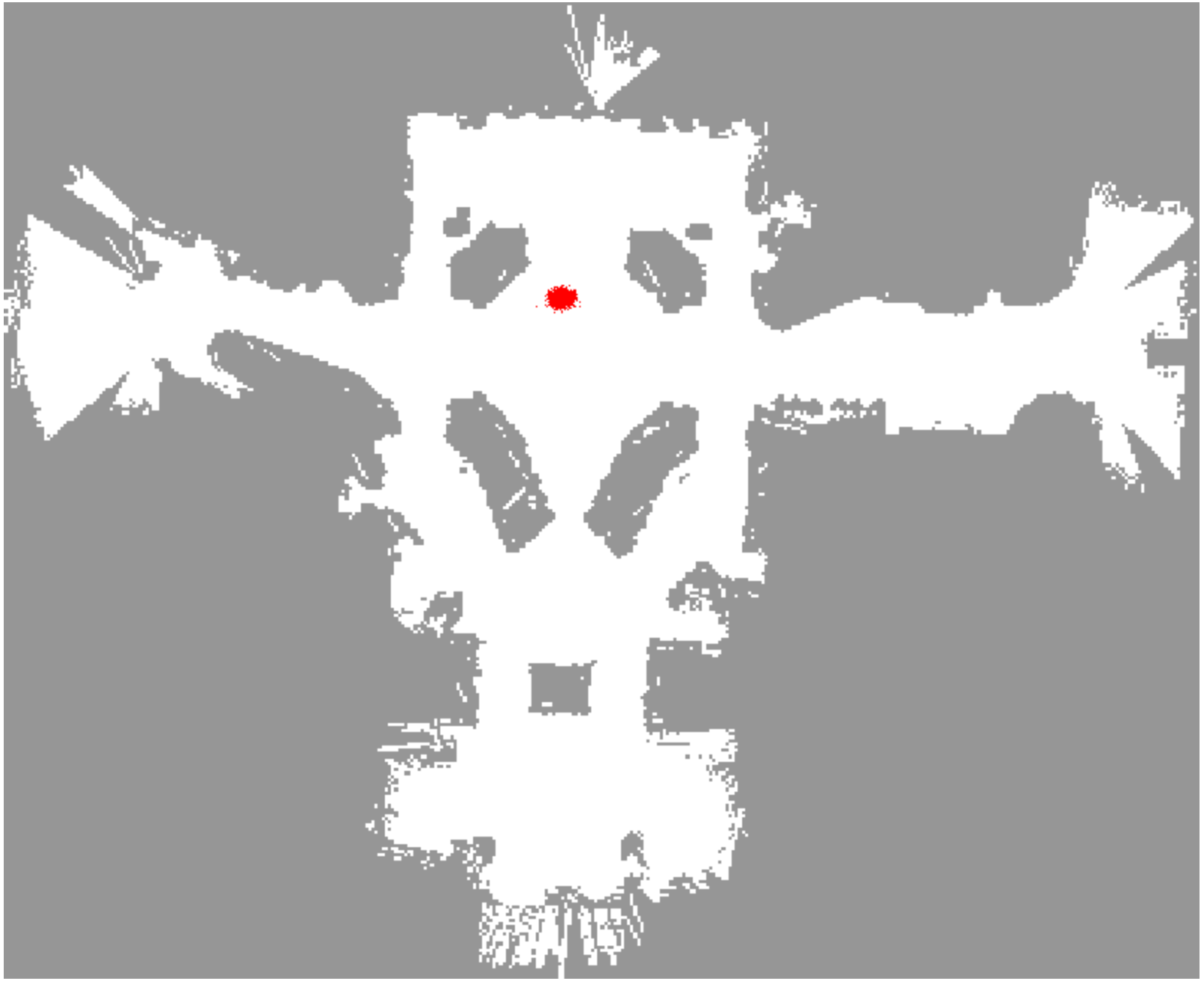
**Tracking scan**
Compare nearby candidate poses.
Red: candidate poses
Blue rays: current scan



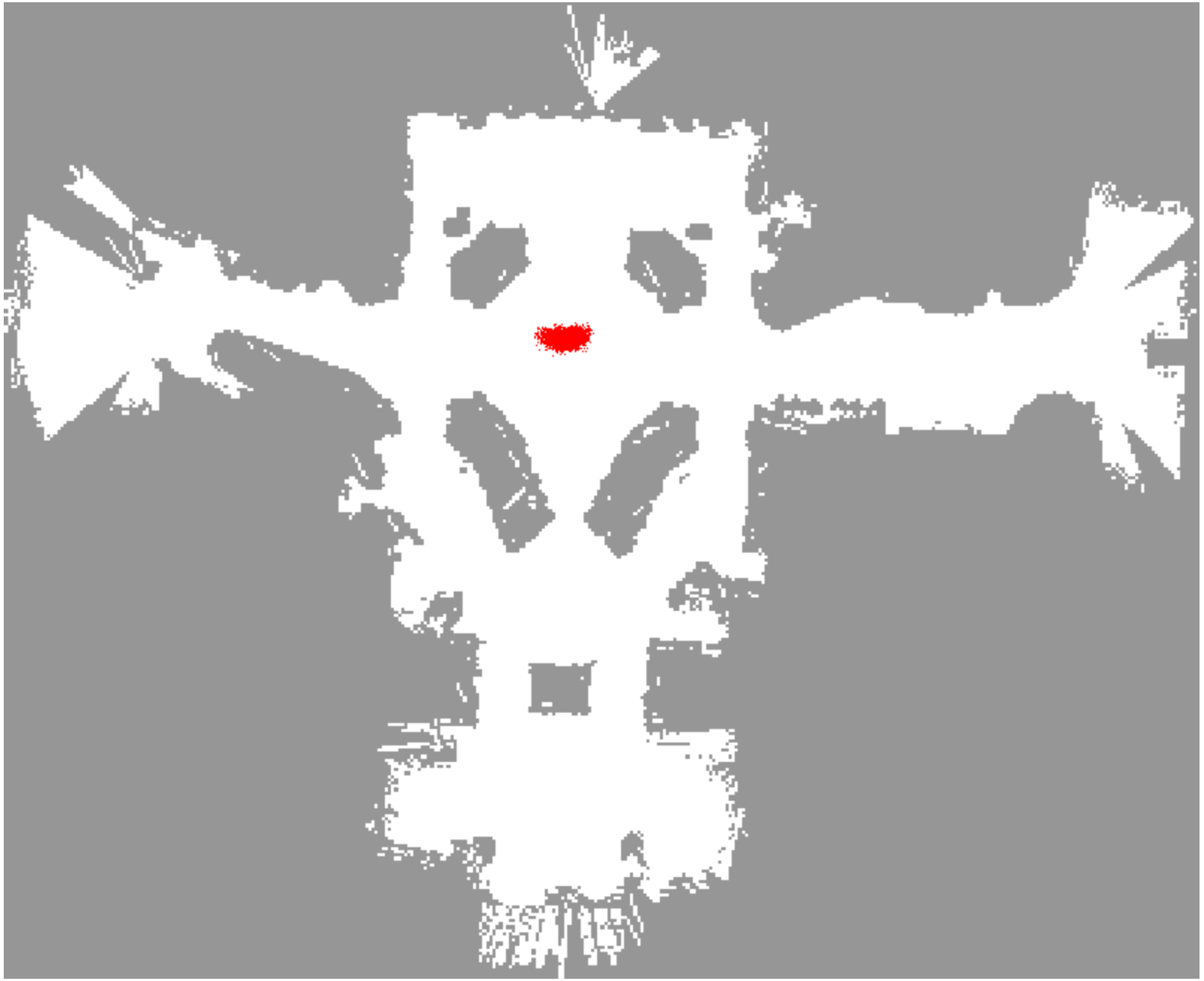
**Tracking update**
Retain candidates that fit the scan.
Red: candidate poses
Blue rays: current scan



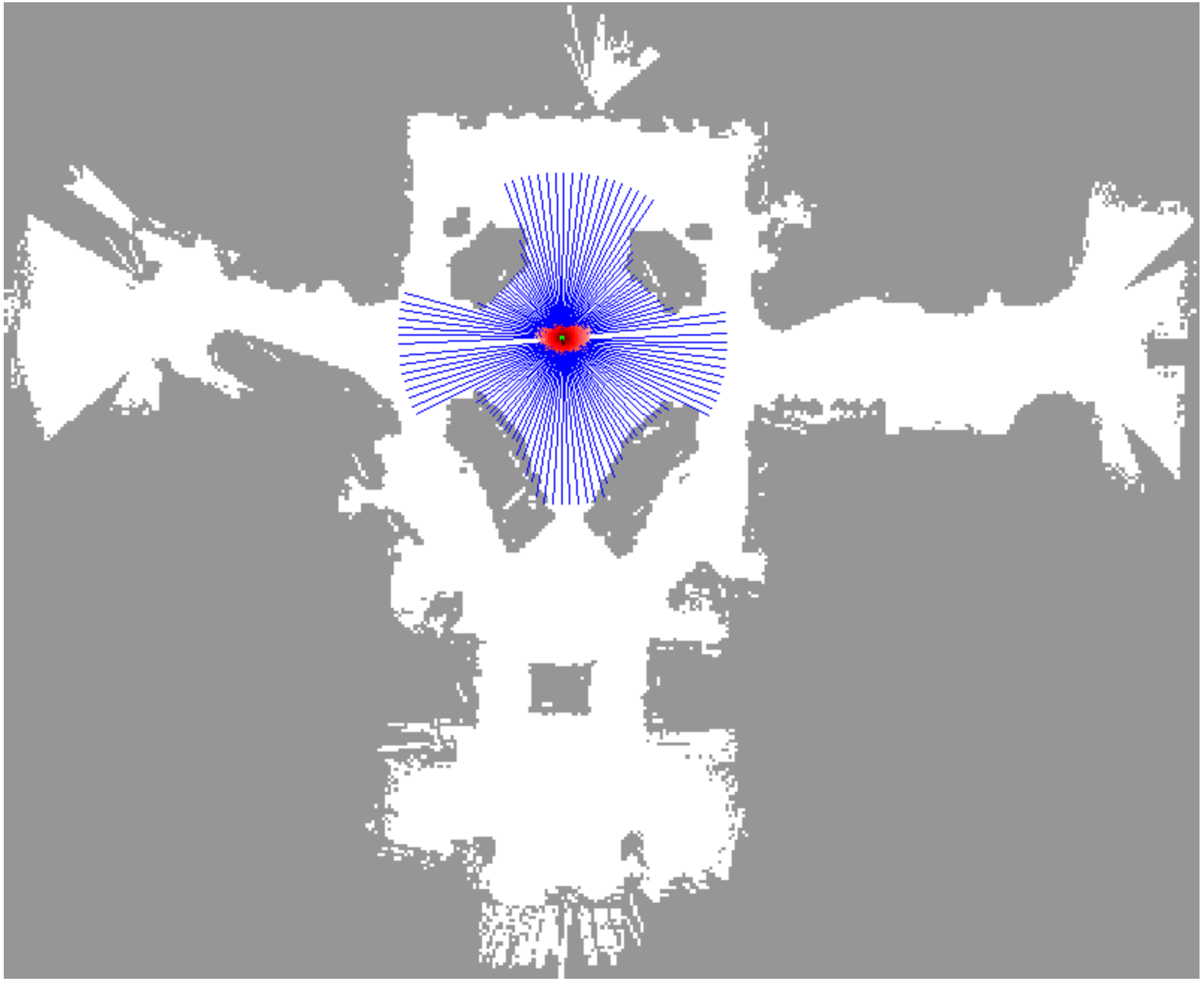
**Compact cloud**
Small spread does not prove correctness.
Red: candidate poses
Blue rays: current scan



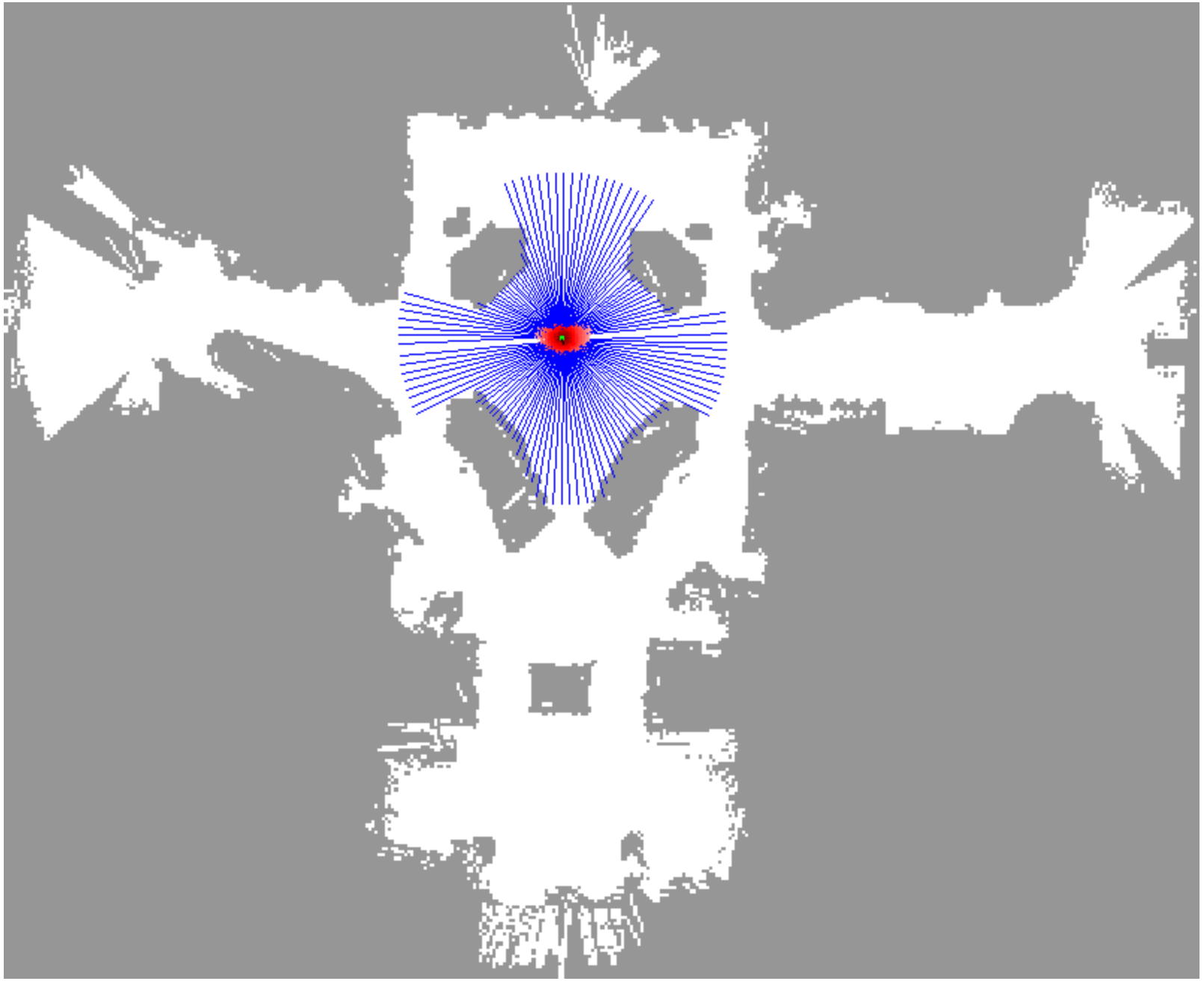
**Move while tracking**
Motion uncertainty remains.
Red: candidate poses
Blue rays: current scan



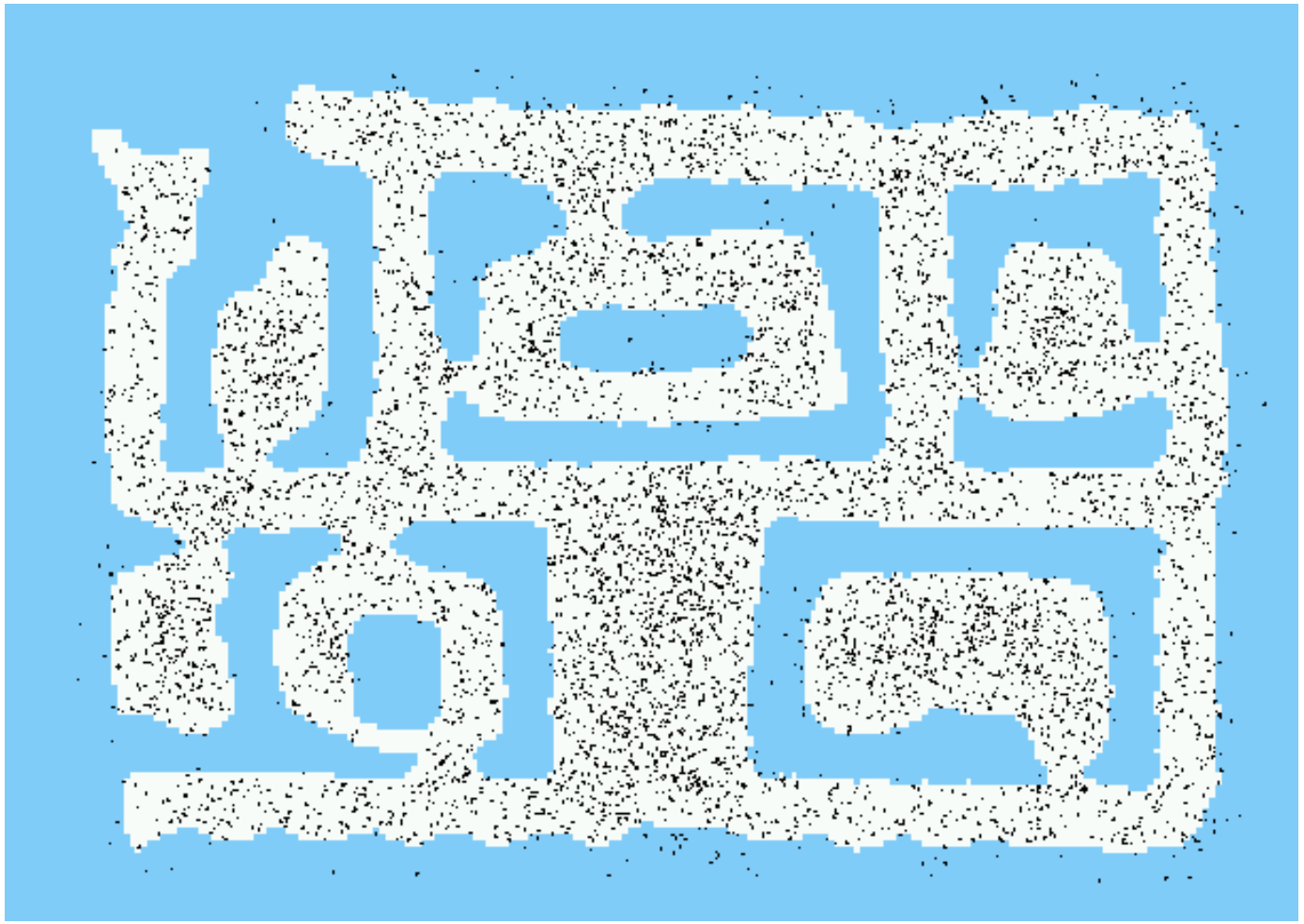
**A new scan**
Sensor evidence updates the cloud.
Red: candidate poses
Blue rays: current scan



**Tracking result**
Original video poster frame retained.
Red: candidate poses
Blue rays: current scan

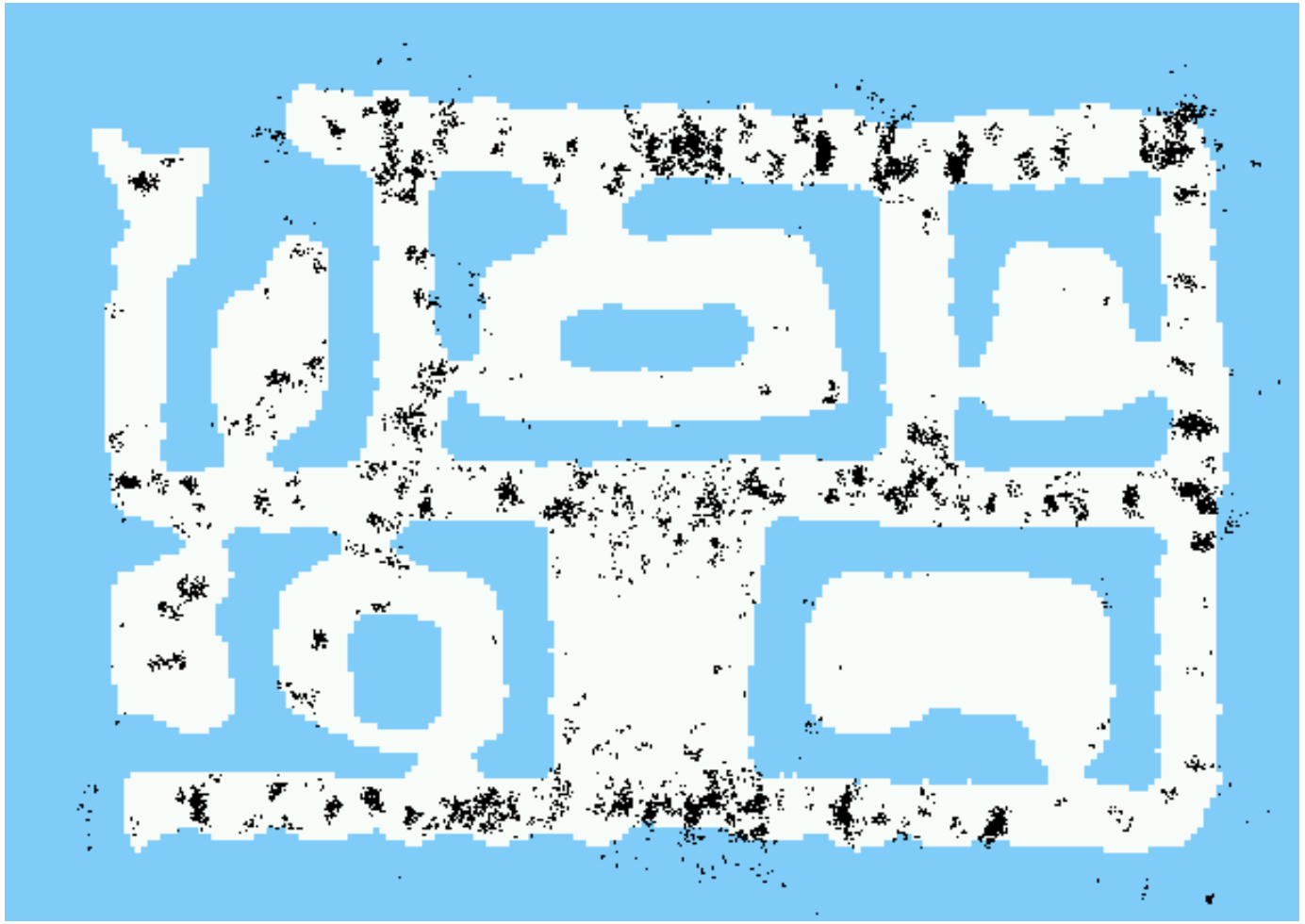


# Initial Distribution



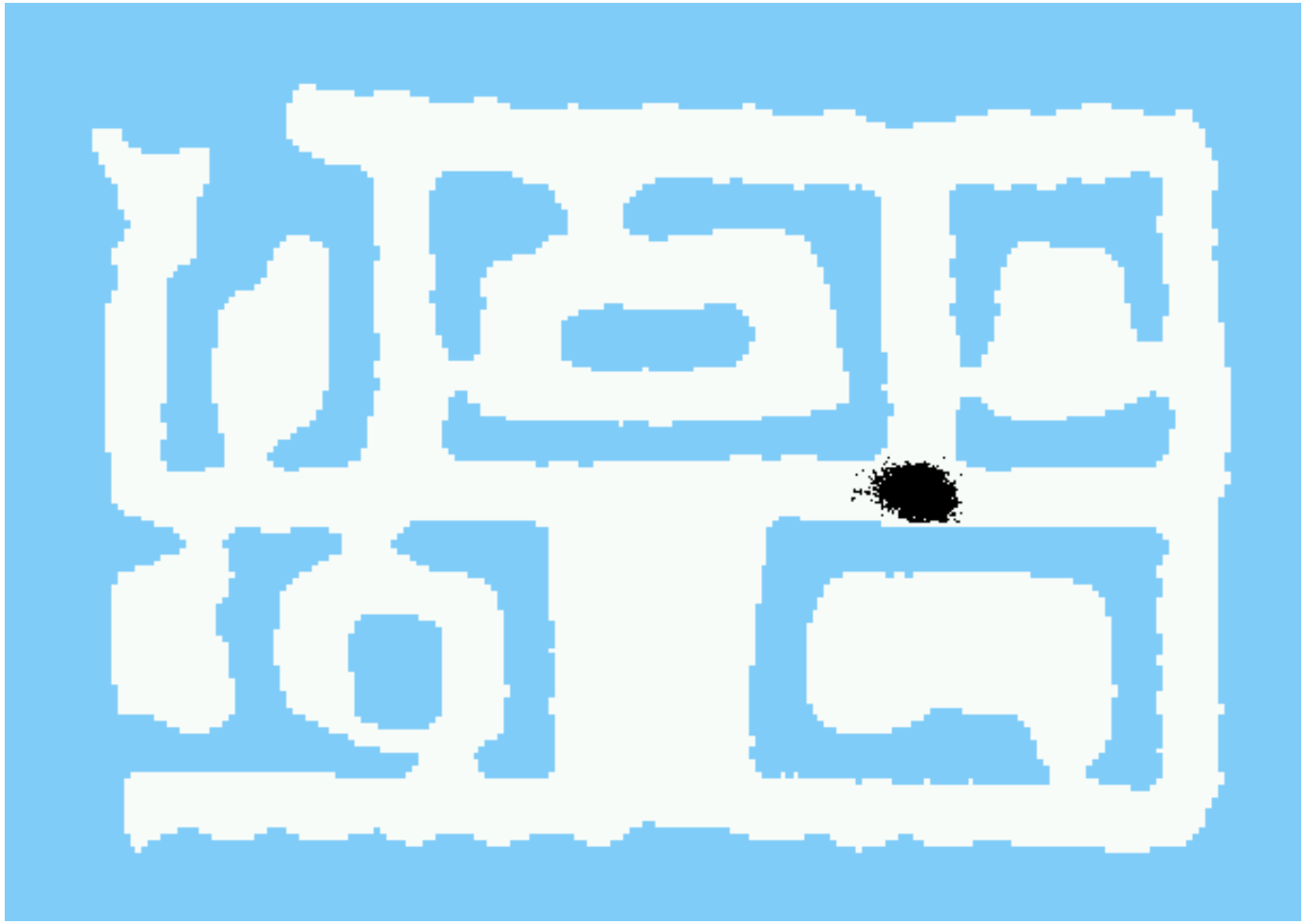
Global localization starts with candidates across free space.

# After Incorporating Ten Ultrasound Scans



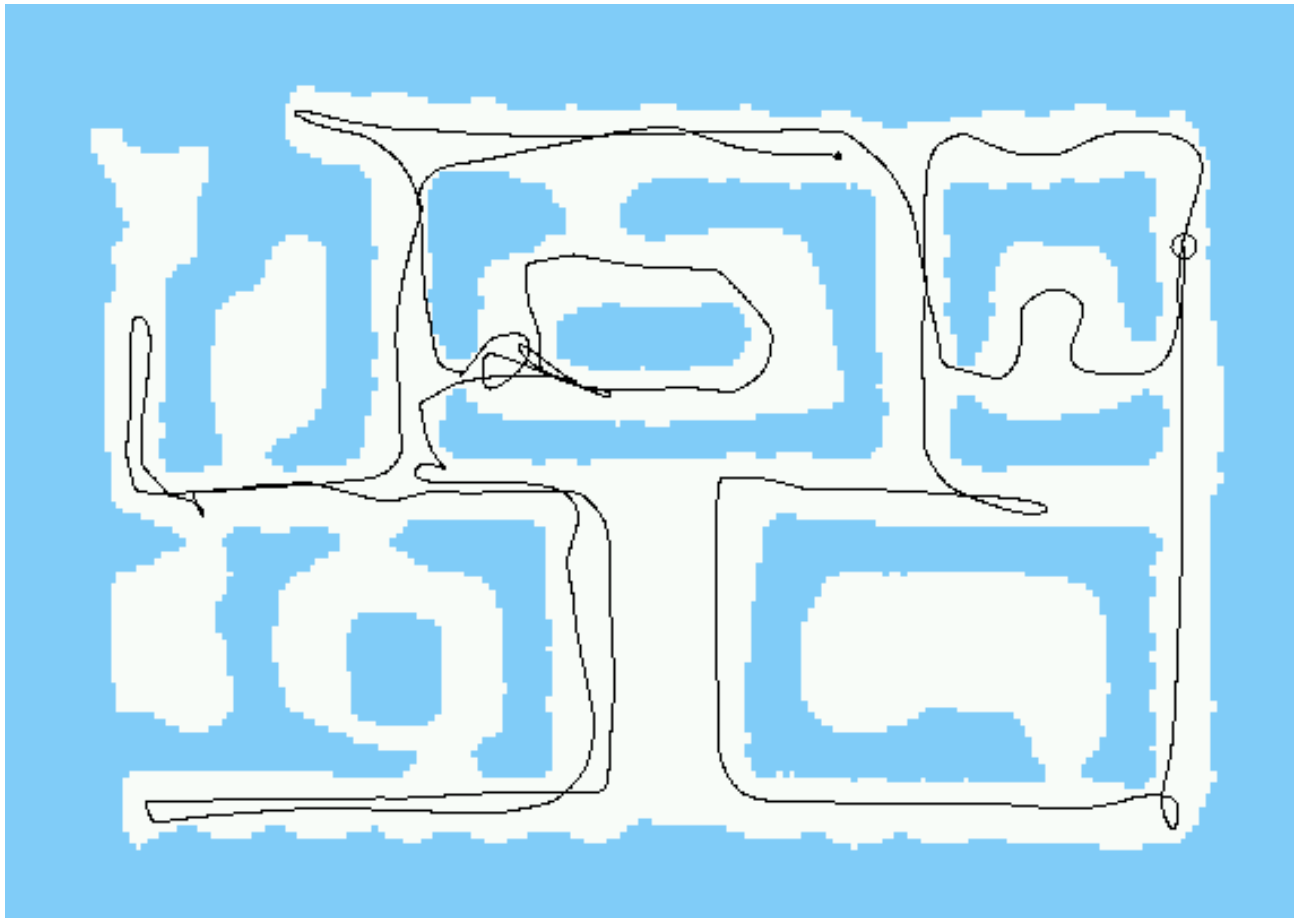
Ten scans: several separated pose hypotheses remain.

# After Incorporating 65 Ultrasound Scans



65 scans: one dominant pose cluster remains in this run.

# Estimated Path



The path is a point estimate; the filter also represents uncertainty.

# Summary & Limitations

**Particle filter:** approximate the posterior with sampled states.

- **Predict** each particle with the motion model.
- **Weight** it with the measurement likelihood.
- **Resample with replacement**.

Supports position tracking and **global localization**.

**Limitation:** finite samples can miss the correct hypothesis.

Particle approximations improve as the number of particles grows, under assumptions.

**Kidnapped robot?** Resampling alone cannot create a missing pose.

- Inject some globally drawn candidates to help recovery.

# Kalman filter: parametric distribution + linear model

Kalman Filter: State estimation algorithm for linear systems with Gaussian uncertainty

- linear model of system and measurement
- hard limit on computation
- Need an exact update **under linear–Gaussian assumptions**

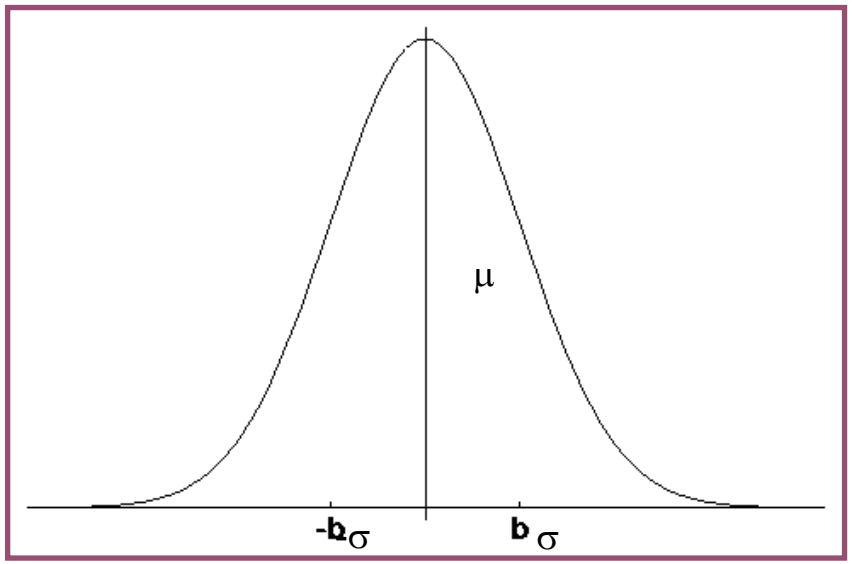
Particle filter:

- Nonparametric method
- Heavy computation
- Can handle nonlinear models
- Does not rely on analytical expression for distributions

# Gaussians

$$p(x) = N(x; \mu, \sigma^2)$$
$$= \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

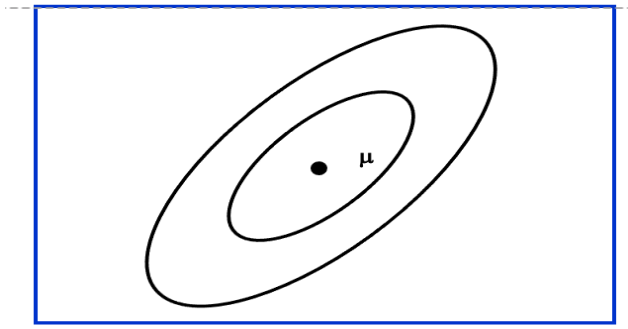
Univariate




Ticks: $\mu \pm b\sigma$ ($b > 0$: chosen multiplier).

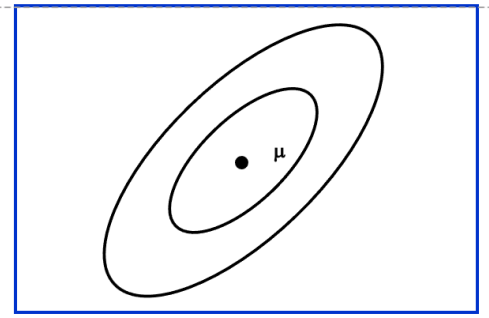
$$p(x) = N(x; \mu, \Sigma)$$
$$= \frac{\exp[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)]}{(2\pi)^{n/2}\sqrt{\det\Sigma}}$$

Multivariate



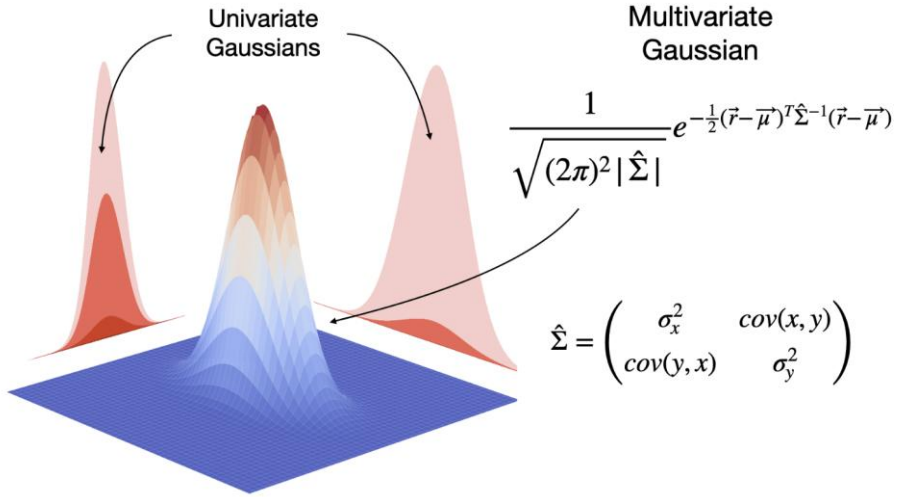

$\mu$: mean; $\sigma^2$: variance; $\Sigma$: covariance matrix.
$n$: number of components; this density needs $\Sigma$ positive definite.

# Multivariate Gaussians

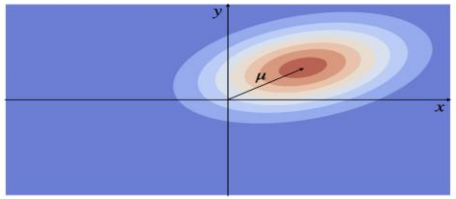



$$N(x;\mu,\Sigma) = \frac{\exp[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)]}{(2\pi)^{n/2}\sqrt{\det\Sigma}}$$

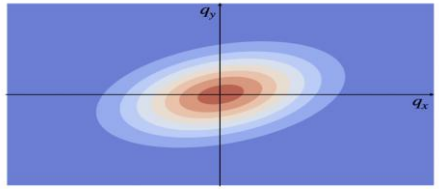
Each component $x_i$ has marginal $N(\mu_i, \sigma_i^2)$; $\sigma_i^2 = \Sigma_{ii}$.
**Uncorrelated components:** $\Sigma$ is diagonal with entries $\sigma_i^2$.



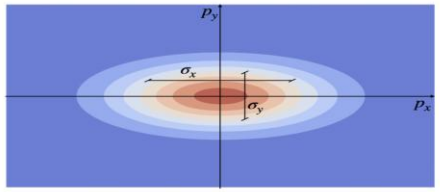

**Interactive demo**:
https://colab.research.google.com/drive/1Z6v83JRmJWKAVuniS48GKuro18Ky1qtt?usp=sharing



Random bivariate Gaussian distribution



Bivariate Gaussian distribution centered in zero



Bivariate Gaussian distribution diagonalised

# Properties of Gaussians

**Affine transformations** of Gaussians are Gaussian.
The mean shifts; the variance scales by $a^2$.

$$X \sim N(\mu, \sigma^2), \quad Y = aX + b \Rightarrow Y \sim N(a\mu + b, a^2\sigma^2)$$

Products of Gaussian densities are proportional to a Gaussian.

$$N(x; \mu_1, \sigma_1^2)N(x; \mu_2, \sigma_2^2) \propto N(x; \mu, \sigma^2)$$
$$\frac{1}{\sigma^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}, \quad \mu = \sigma^2\left(\frac{\mu_1}{\sigma_1^2} + \frac{\mu_2}{\sigma_2^2}\right)$$

$a, b$: constants; $\mu$: mean; $\sigma^2$: variance.

# Multivariate Gaussians: same properties apply

$$X \sim N(\mu, \Sigma), \quad Y = AX + b \Rightarrow Y \sim N(A\mu + b, A\Sigma A^T)$$

$$N(x; \mu_1, \Sigma_1)N(x; \mu_2, \Sigma_2) \propto N(x; \mu, \Sigma)$$
$$\Sigma = (\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}$$
$$\mu = \Sigma(\Sigma_1^{-1}\mu_1 + \Sigma_2^{-1}\mu_2)$$

Gaussian prediction: **affine map + independent Gaussian noise**.

Gaussian correction: **condition on a linear Gaussian measurement**.

$A$: matrix; $b$: constant vector; $\mu$: mean; $\Sigma$: covariance.

The product formulas assume invertible $\Sigma_1, \Sigma_2$.

# Discrete Kalman Filter

The Kalman filter estimates a **discrete-time linear system** with Gaussian noise.

Time is discrete; states and measurements can take continuous values.

Assume Gaussian initial belief and independent, zero-mean Gaussian noises.

motion model $x_t = A_{t-1}x_{t-1} + B_{t-1}u_{t-1} + \varepsilon_{t-1}$

measurement model $z_t = C_t x_t + \delta_t$

observation matrix

$x_t$: state vector; $u_{t-1}$: applied input; $z_t$: sensor reading.

$\varepsilon_{t-1} \sim N(0, Q_{t-1})$: **process** noise.

$\delta_t \sim N(0, R_t)$: **measurement** noise.

$$p_D(x_t|x_{t-1}, u_{t-1}) = N(x_t; A_{t-1}x_{t-1} + B_{t-1}u_{t-1}, Q_{t-1})$$

$$p_M(z_t|x_t) = N(z_t; C_t x_t, R_t)$$

# Kalman Filter Algorithm

Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_{t-1}, z_t$):

**Prediction:**

$$\overline{\mu}_t = A_{t-1}\mu_{t-1} + B_{t-1}u_{t-1}$$
$$\overline{\Sigma}_t = A_{t-1}\Sigma_{t-1}A_{t-1}^T + Q_{t-1}$$

**Correction:**

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + R_t)^{-1}$$
$$\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$$

Return $\mu_t, \Sigma_t$.

$bel(x_{t-1}) = N(x_{t-1}; \mu_{t-1}, \Sigma_{t-1})$

**Prediction:** affine motion plus independent noise.

Mean $\overline{\mu}_t$; covariance $\overline{\Sigma}_t$ includes $Q_{t-1}$.

**Correction:** a multiplication: prediction × measurement likelihood

$p_M(z_t|x_t) = N(z_t; C_t x_t, R_t)$

The posterior (product of two Gaussian) is Gaussian.

$bel(x_t) = N(x_t; \mu_t, \Sigma_t)$

$I$: identity; $R_t$: sensor-noise covariance; $C_t$: observation matrix

# Kalman Filter Algorithm

Kalman_Filter($\mu_{t-1}, \Sigma_{t-1}, u_{t-1}, z_t$):

**Prediction:**

$$\overline{\mu}_t = A_{t-1}\mu_{t-1} + B_{t-1}u_{t-1}$$
$$\overline{\Sigma}_t = A_{t-1}\Sigma_{t-1}A_{t-1}^T + Q_{t-1}$$

**Correction:**

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + R_t)^{-1}$$
$$\mu_t = \overline{\mu}_t + K_t(z_t - C_t\overline{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t)\overline{\Sigma}_t$$

Return $\mu_t, \Sigma_t$.

$\mu_t, \Sigma_t$: belief **mean and covariance**.

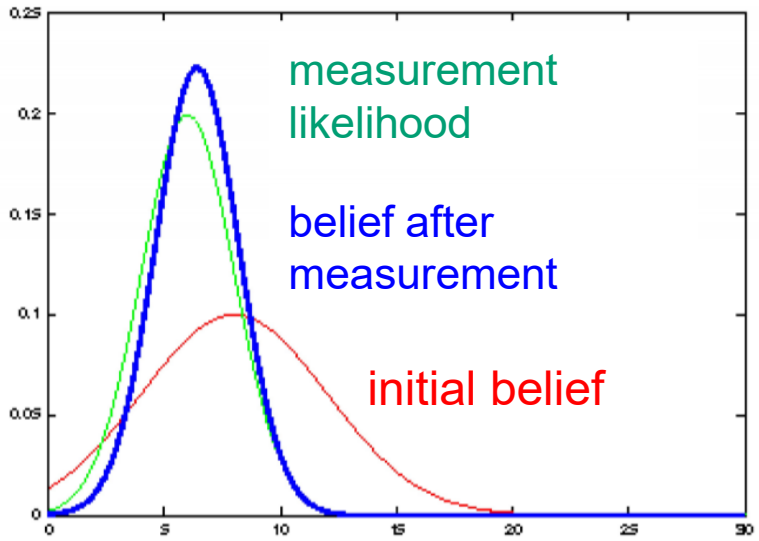
**Innovation:** $\nu_t = z_t - C_t\overline{\mu}_t$ (actual − predicted reading).
**Kalman gain** $K_t$ converts that residual into a state correction.

In 1D with $C_t = 1$:

$$K_t = \overline{\sigma}_t^2 / (\overline{\sigma}_t^2 + r_t)$$

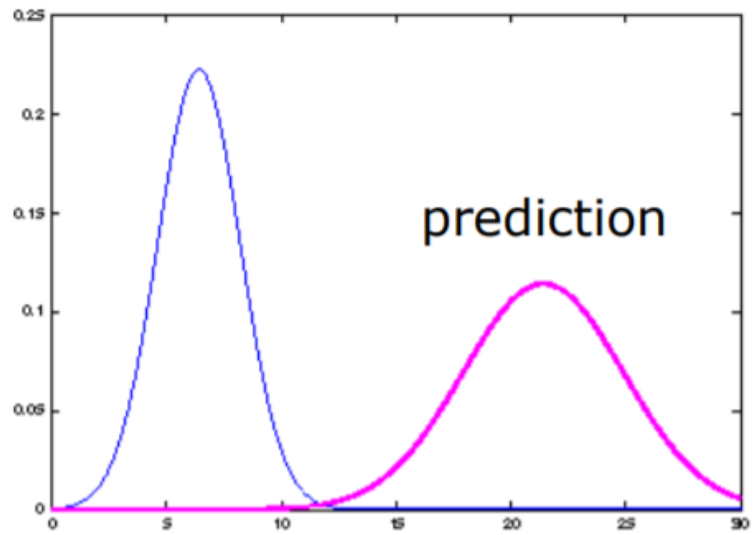
Larger sensor variance $r_t$ (more noisy sensor) gives a smaller gain (less impact on belief).

$r_t$: scalar sensor variance;
$\overline{\sigma}_t^2$: predicted variance.




**Apply control action**

Mean and variance moved according to the linear dynamics

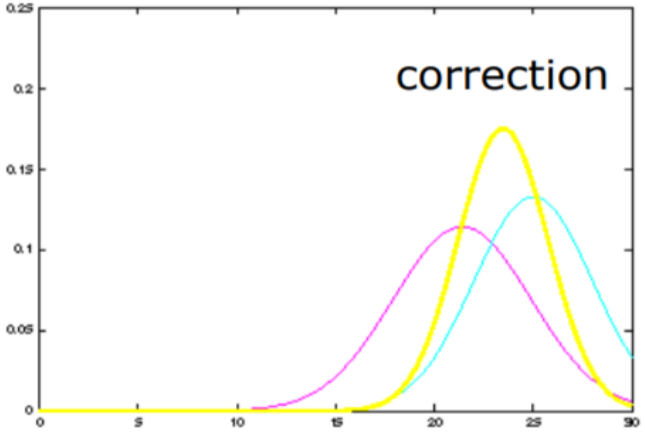



**Correction:**

$$K_t = \overline{\Sigma}_t C_t^T (C_t \overline{\Sigma}_t C_t^T + R_t)^{-1}$$
$$\mu_t = \overline{\mu}_t + K_t (z_t - C_t \overline{\mu}_t)$$
$$\Sigma_t = (I - K_t C_t) \overline{\Sigma}_t$$

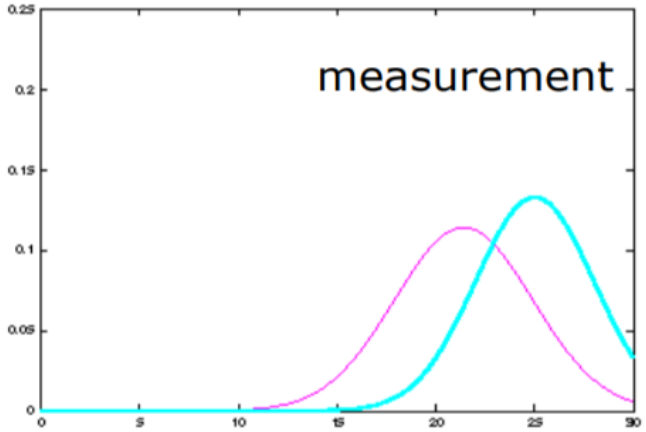
**Prediction:**

$$\overline{\mu}_t = A_{t-1} \mu_{t-1} + B_{t-1} u_{t-1}$$
$$\overline{\Sigma}_t = A_{t-1} \Sigma_{t-1} A_{t-1}^T + Q_{t-1}$$



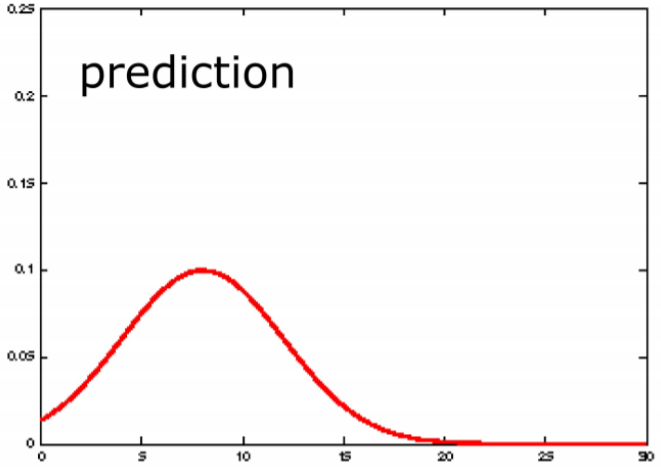

Product of two gaussians
prediction × measurement

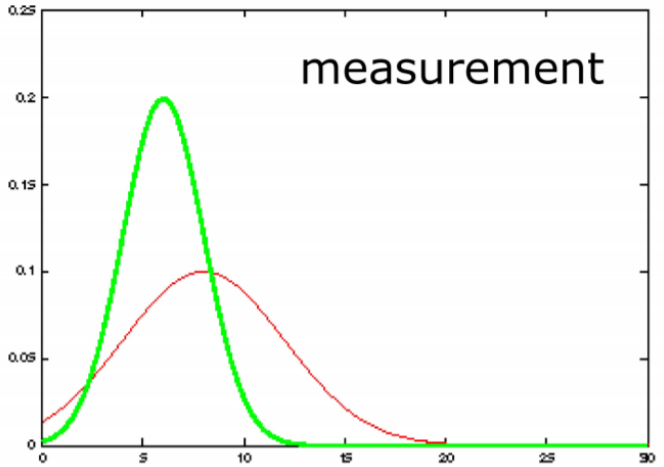
**Get sensor measurement**

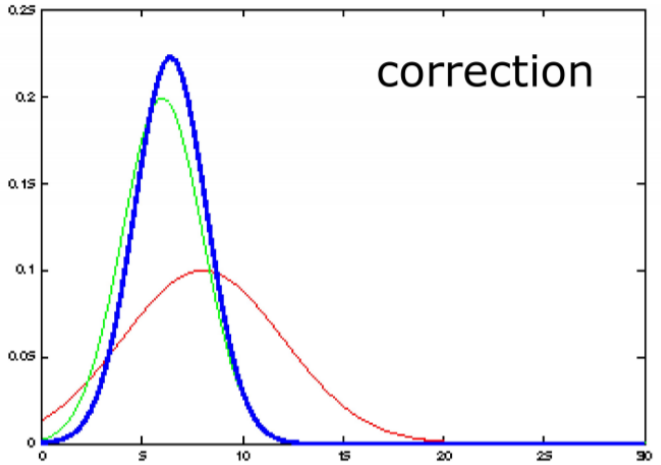


# Kalman Filter Example

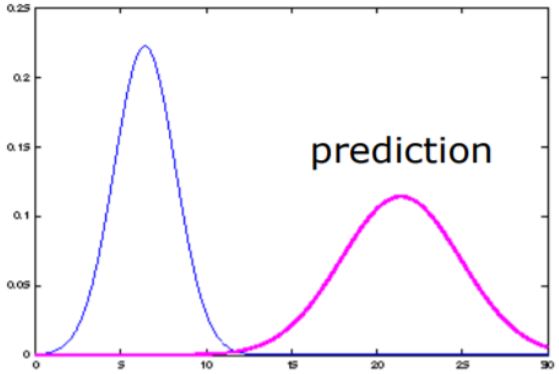
time = 1

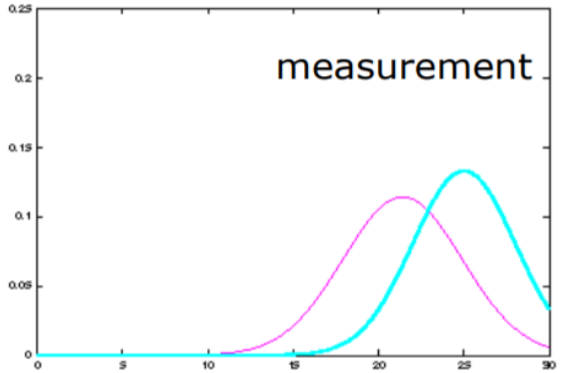



time = 2







**Demo**: https://colab.research.google.com/drive/1qclNZgx8ebwWtRQROh3z8cpvtmuE4Dt0?usp=sharing
The demo uses 2D Gaussian.
Try different sensor covariance (noise) and see what you observe for the correction step.