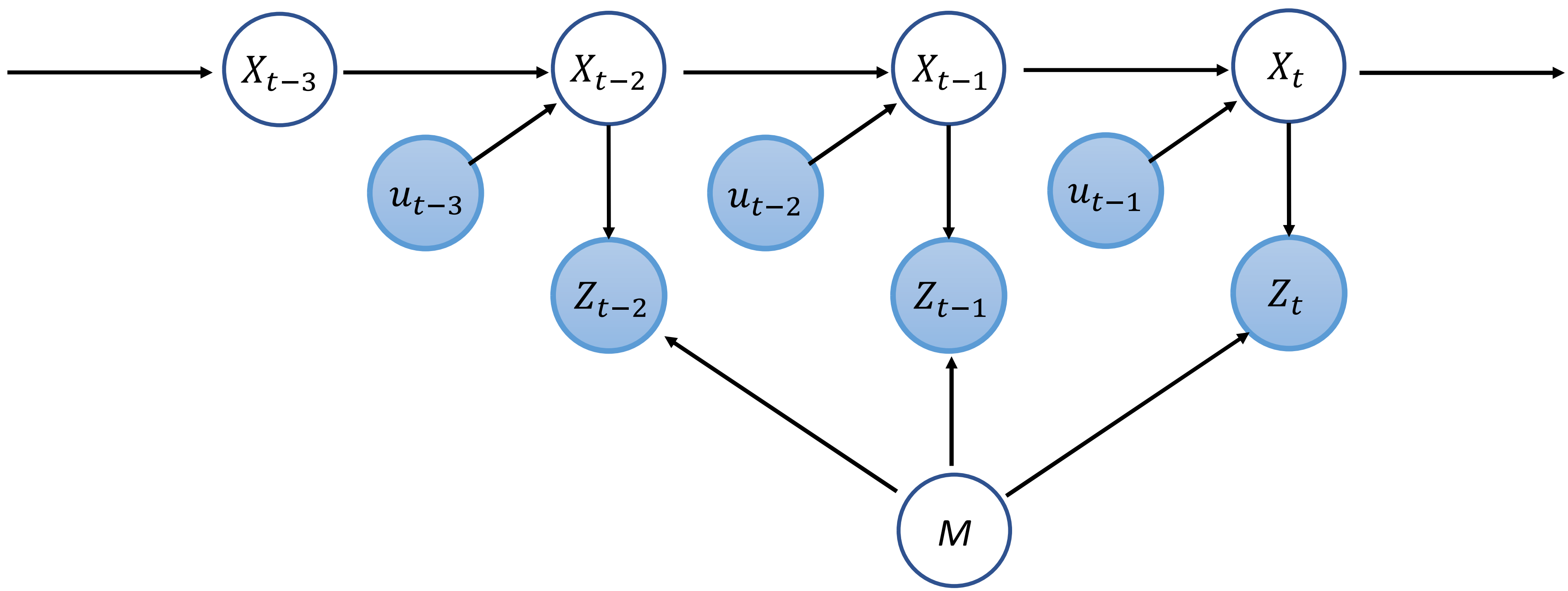
EECE 5550: Mobile Robotics



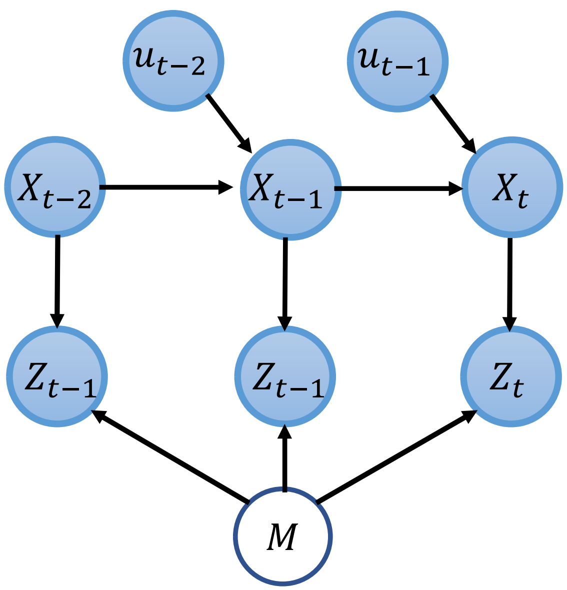
Lecture 12: Localization

Recap

Last time: mapping

Given: Robot poses $x_{0:t}$, measurements $z_{1:t}$

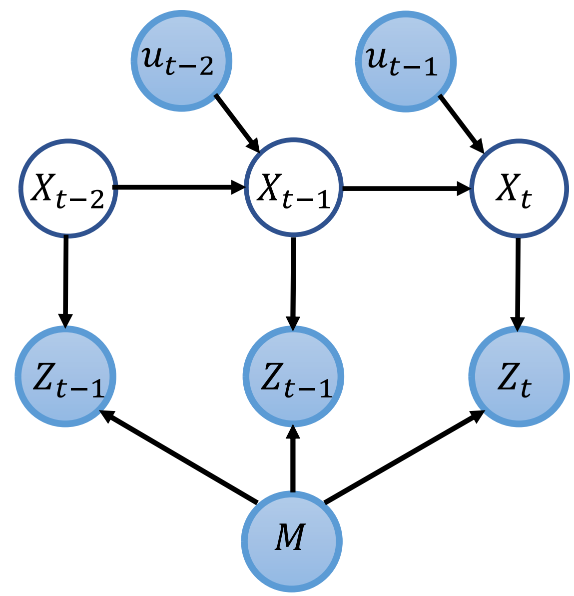
Estimate: Belief $p(m|x_{0:t}, z_{1:t})$ over the map M



This time: localization

Given: Prior $p(x_0)$, map m, controls $u_{0:t}$, obs. $z_{1:t}$

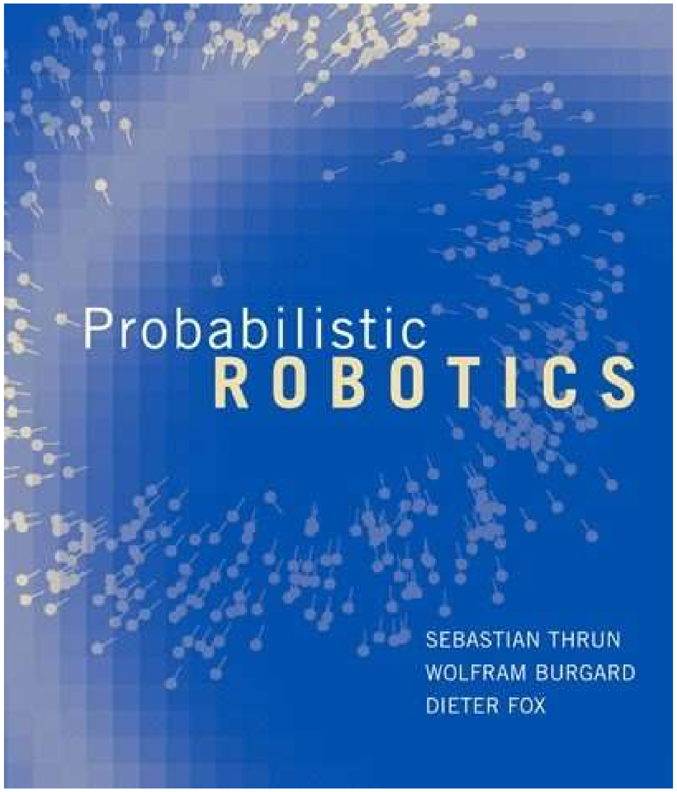
Estimate: Belief $p(x_t|\, m, z_{1:t})$ over the robot pose



Plan of the day

- Monte Carlo localization
- Models:
  - Probabilistic motion models
  - Forward beam sensor models
- Practicalities:
  - Particle diversity / depletion
  - Tracking localization performance
  - Adaptive sample sets: KLD-sampling

References



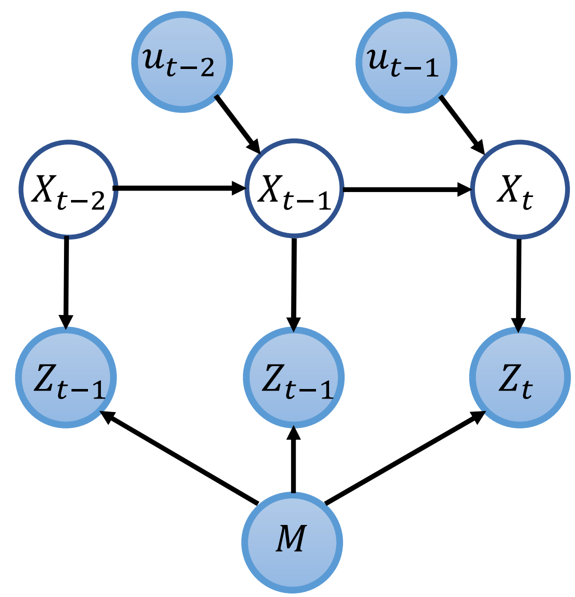
Chapters 5, 6, and 8 of “Probabilistic Robotics”

Monte Carlo Localization

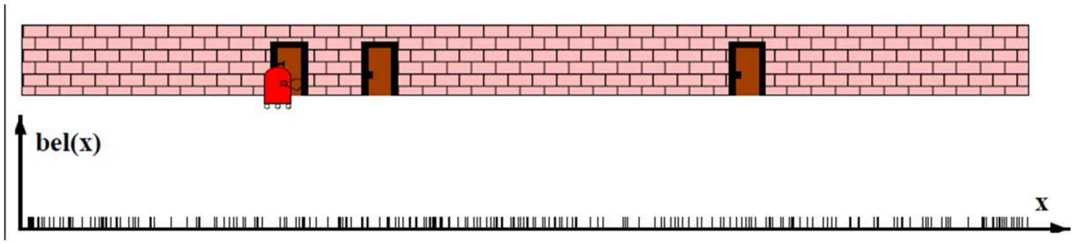
Main idea: Monte Carlo localization (MCL) is simply localization using the Particle Filter

Given: Prior $p(x_0)$, map m, controls $u_{0:t}$, observations $z_{1:t}$
Estimate: Belief $p(x_t|\, m, z_{1:t})$ over the robot pose

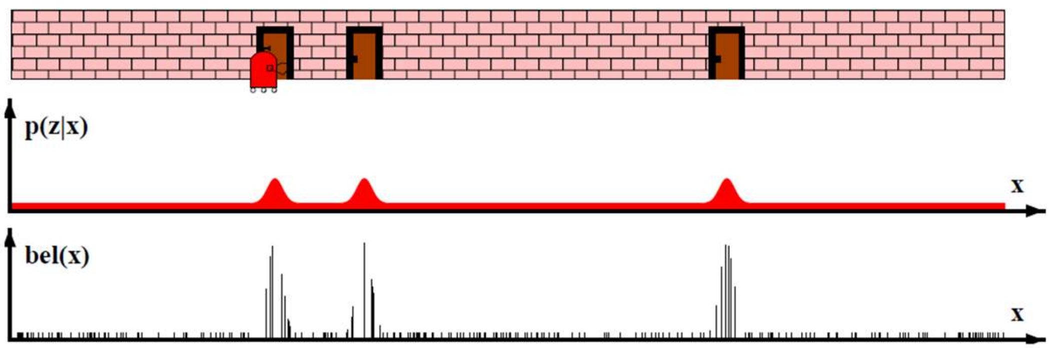
Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):
  $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
  for $m = 1$ to M do
    sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$
    $w_t^{[m]} = p(z_t \mid x_t^{[m]})$
    $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
  endfor
  for $m = 1$ to M do
    draw i with probability $\propto w_t^{[i]}$
    add $x_t^{[i]}$ to $\mathcal{X}_t$
  endfor
  return $\mathcal{X}_t$



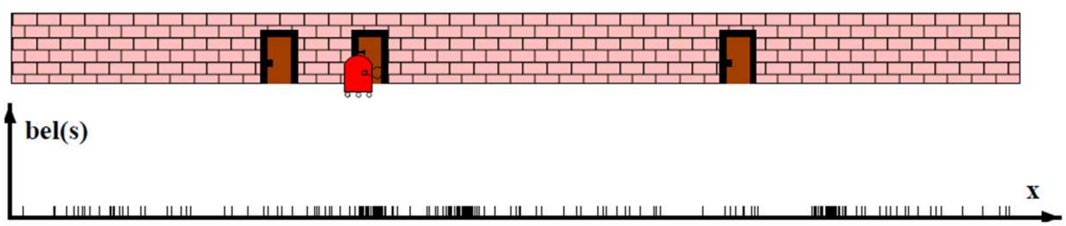
Example application



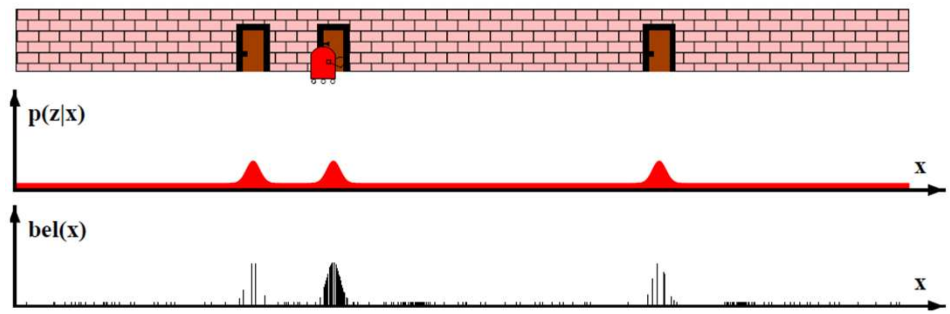
Initial belief



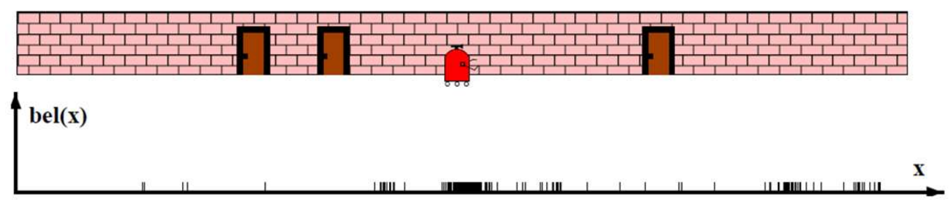
Measurement



Move right



Measurement



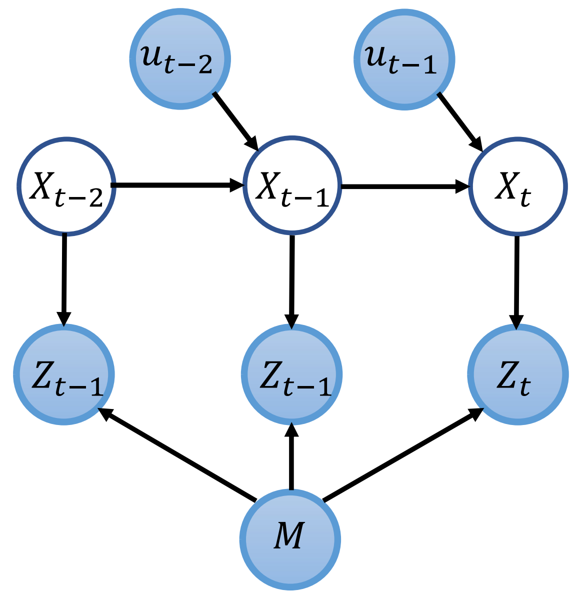
Move right

Monte Carlo Localization

Main idea: Monte Carlo localization (MCL) is simply localization using the Particle Filter

Given: Prior $p(x_0)$, map m, controls $u_{0:t}$, observations $z_{1:t}$
Estimate: Belief $p(x_t|\, m, z_{1:t})$ over the robot pose

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$)**:**
$\quad \bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
$\quad$ for $m = 1$ to M do
$\quad\quad$ sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ ⟵ Motion model
$\quad\quad$ $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ ⟵ Sensing model
$\quad\quad$ $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
$\quad$ endfor
$\quad$ for $m = 1$ to M do
$\quad\quad$ draw i with probability $\propto w_t^{[i]}$
$\quad\quad$ add $x_t^{[i]}$ to $\mathcal{X}_t$
$\quad$ endfor
$\quad$ return $\mathcal{X}_t$

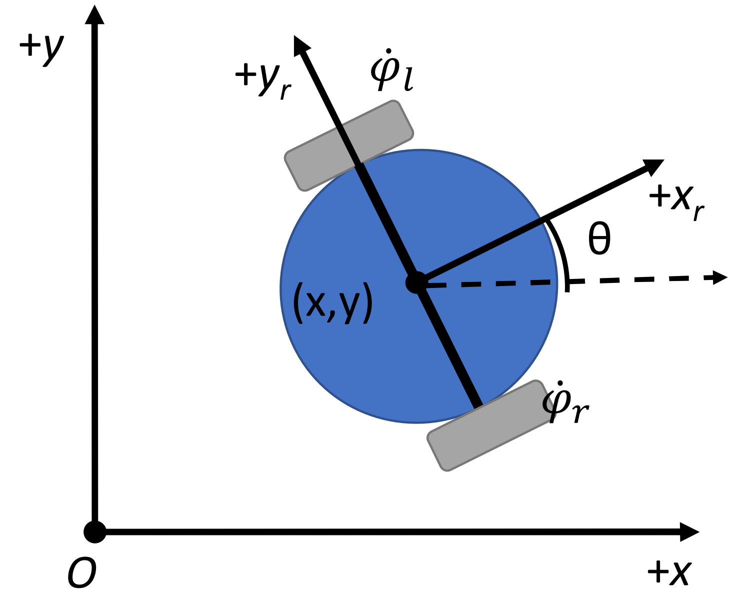


Requires:
- A *sampler* for the motion model
- *Likelihood function* for the sensor model

Robot motion models

We will consider two primary types of robot motion models:

- Velocity-based
- Odometry-based



Velocity-based motion models

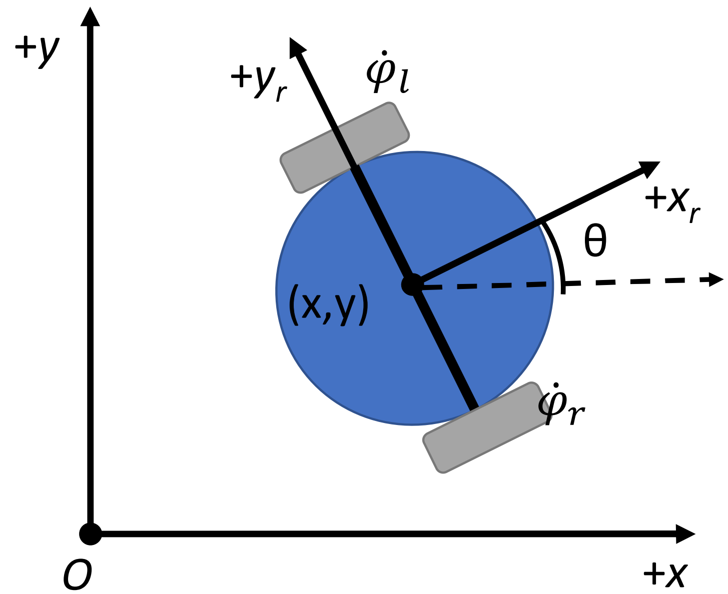
Goal: We need a procedure for drawing samples $x_{t+1} \sim p(x_{t+1}|x_t, u_t)$ from the robot's motion model

Recall: In Lecture 7 we derived the kinematic equations for a differential-drive robot. These give the robot's body-centric velocity $\dot{v}_r \triangleq \left(\dot{x}_r, \dot{y}_r, \dot{\theta}_r\right)$ as a function of its wheel speeds $(\dot{\varphi}_l, \dot{\varphi}_r)$.

In the *noiseless* case, we can integrate $\dot{v}_r$ to calculate the next pose x_{t+1} given the current pose x_t. (You will do this in Lab #2.)

We can model *noisy motion* by supposing that the robot's velocity $\dot{v}_r$ is subject to noise. This leads to a very natural sampling algorithm:

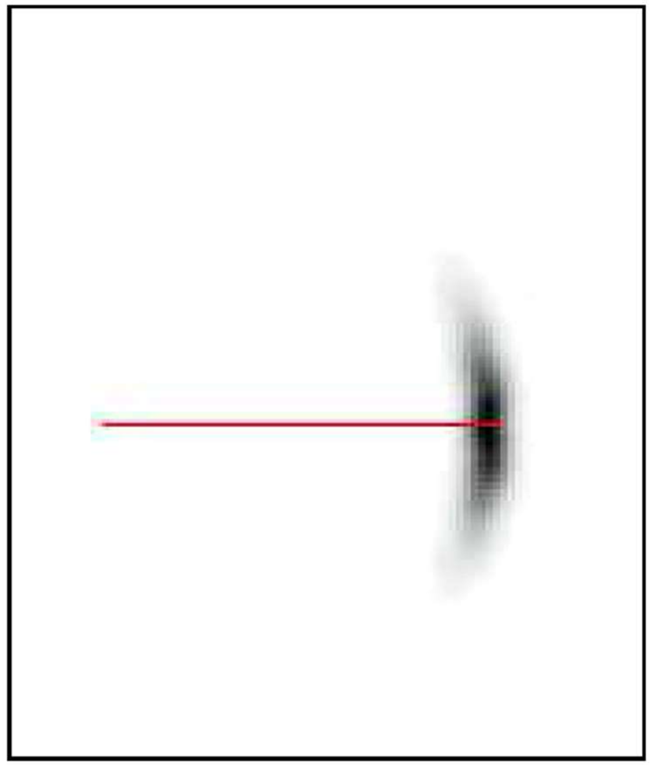
sample_velocity_motion_model($x_t, \dot{\varphi}_l, \dot{\varphi}_r$):
1. Calculate body-centric velocity $\dot{v}_r$ from $(\dot{\varphi}_l, \dot{\varphi}_r)$ using kinematic model
2. Add noise: $\tilde{v}_r \triangleq \dot{v}_r + \Delta v_r$, where Δv_r is sampled from a noise distribution (e.g. Gaussian)
3. Calculate next pose x_{t+1} by integrating *noisy* velocity $\tilde{v}_r$



$$\begin{pmatrix} \dot{x}_r \\ \dot{y}_r \\ \dot{\theta}_r \end{pmatrix} = \begin{pmatrix} \frac{r}{2}(\dot{\varphi}_r + \dot{\varphi}_l) \\ 0 \\ \frac{r}{w}(\dot{\varphi}_r - \dot{\varphi}_l) \end{pmatrix}$$

The Banana Distribution

This velocity-based motion model induces a “banana”-shaped marginal distribution over position



In robotics, this is often referred to (appropriately enough) as the “banana distribution”

Odometry-based motion models

Velocity-based motion models provide a simple means of simulating noisy robot motion.

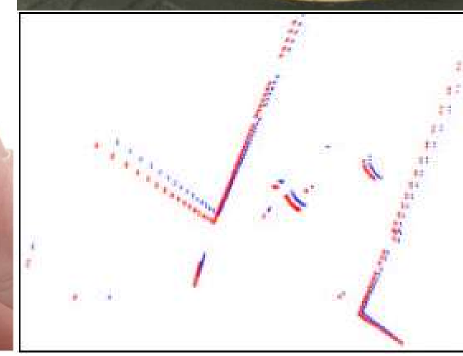
But: They are often grossly oversimplified (⇒not very accurate)

Alternative: Treat an odometry measurement $z_t \approx x_{t-1}^{-1} x_t$ between pose x_{t-1} and x_t *as if* it were a *motion command* u_{t-1}. That is, consider:

$$x_t \sim p(x_t \mid x_{t-1}, z_t)$$

Payoffs:

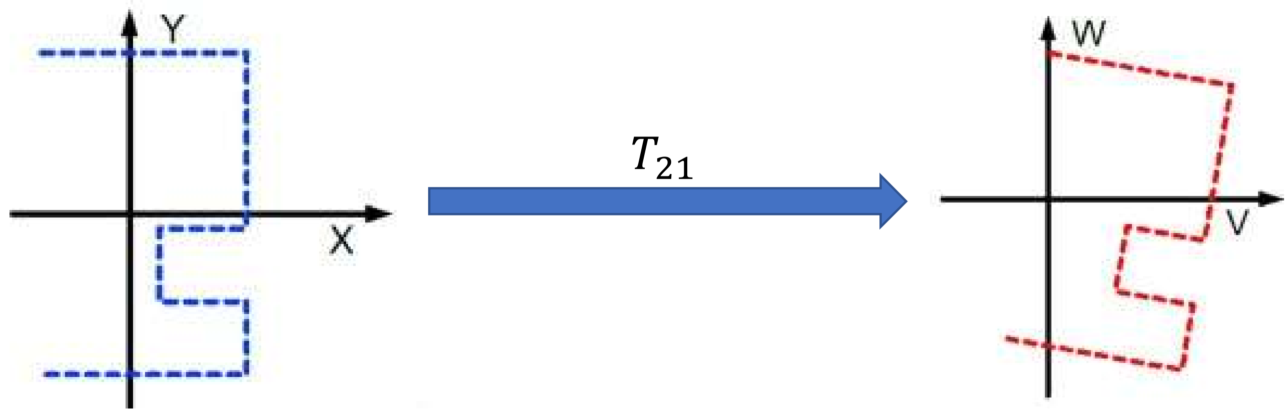
- Often significantly more accurate than velocity models
- Odometry can be gotten from several sources:
  - Wheel encoders
  - Inertial measurement units
  - Scan matching or visual odometry



Con: Only available *post hoc* – can't use for planning

Estimating odometry via scan matching

Suppose I have two scans of environment taken from nearby poses (so that they partially overlap)



Scan at pose 1

Scan at pose 2

Question: What is the coordinate transformation T_{21} that relates their overlap?

Recall: If T_{W1} and T_{W2} are poses of the scanner in the world frame *W*, and *p* is a point, then:

$$T_{W1}p_1 = p_W = T_{W2}p_2$$

$$\Rightarrow p_2 = \underbrace{T_{W2}^{-1}T_{W1}}_{= T_{21}} p_1$$

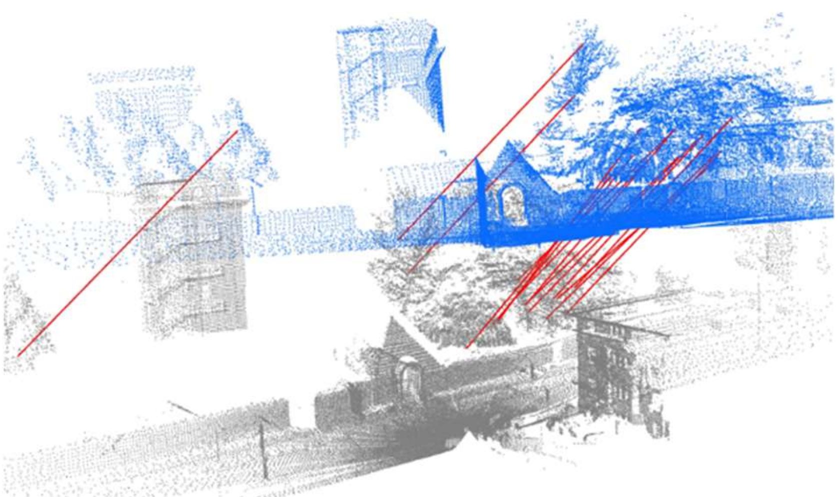
Punchline: Scan matching provides a means of *directly measuring odometry*

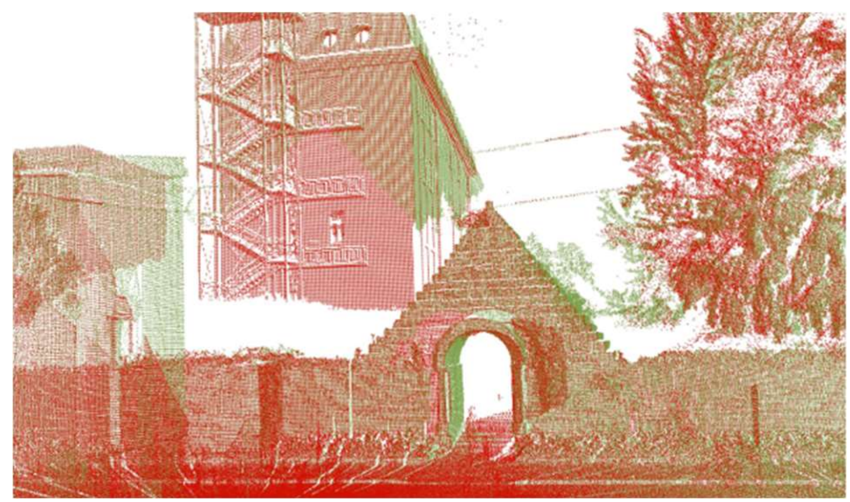
But: How can we actually *find* T_{21} given only the two scans?

Pointcloud registration

Problem: Given point sets $X = \{x_i\}_{i=1}^{n}$ and $Y = \{y_i\}_{i=1}^{n}$ in $\mathbb{R}^d$, find the transformation $T = (t, R) \in SE(d)$ that *optimally aligns* them:

$$(t, R) = argmin \sum_{i=1}^{n} \|y_i - (Rx_i + t)\|^2$$





Pointcloud registration

Given: particle sets $X = \{x_i\}_{i=1}^{n}$ and $Y = \{y_i\}_{i=1}^{n}$ in $\mathbb{R}^d$

Horn's method:

1. Calculate centroids:
$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i\,, \qquad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i\,,$$
2. Center the pointclouds: $x_i' \triangleq x_i - \bar{x},\ \ y_i' \triangleq y_i - \bar{y}$
3. Construct outer product matrix W:
$$W \triangleq \sum_i y_i' x_i'^T$$
4. Recover optimal rotation R:
$$R = U \cdot diag(1, \dots, 1, \det(UV^T)) \cdot V^T$$
where $W = U\Sigma V^T$ is the singular value decomposition of W
5. Recover optimal translation t: $\quad t = \bar{y} - R\bar{x}$

Iterative Closest Point

Catch: Horn's method assumes that we know the *point correspondences* $x_i \leftrightarrow y_i$. What if we don't have these?

One approach: Estimate these together with the registration $T = (t, R)$!

Iterative closest point (ICP): Given $X = \{x_i\}_{i=1}^{n}$ and $Y = \{y_j\}_{j=1}^{m}$, initial guess $T = (t_0, R_0)$, maximum association distance $d > 0$:

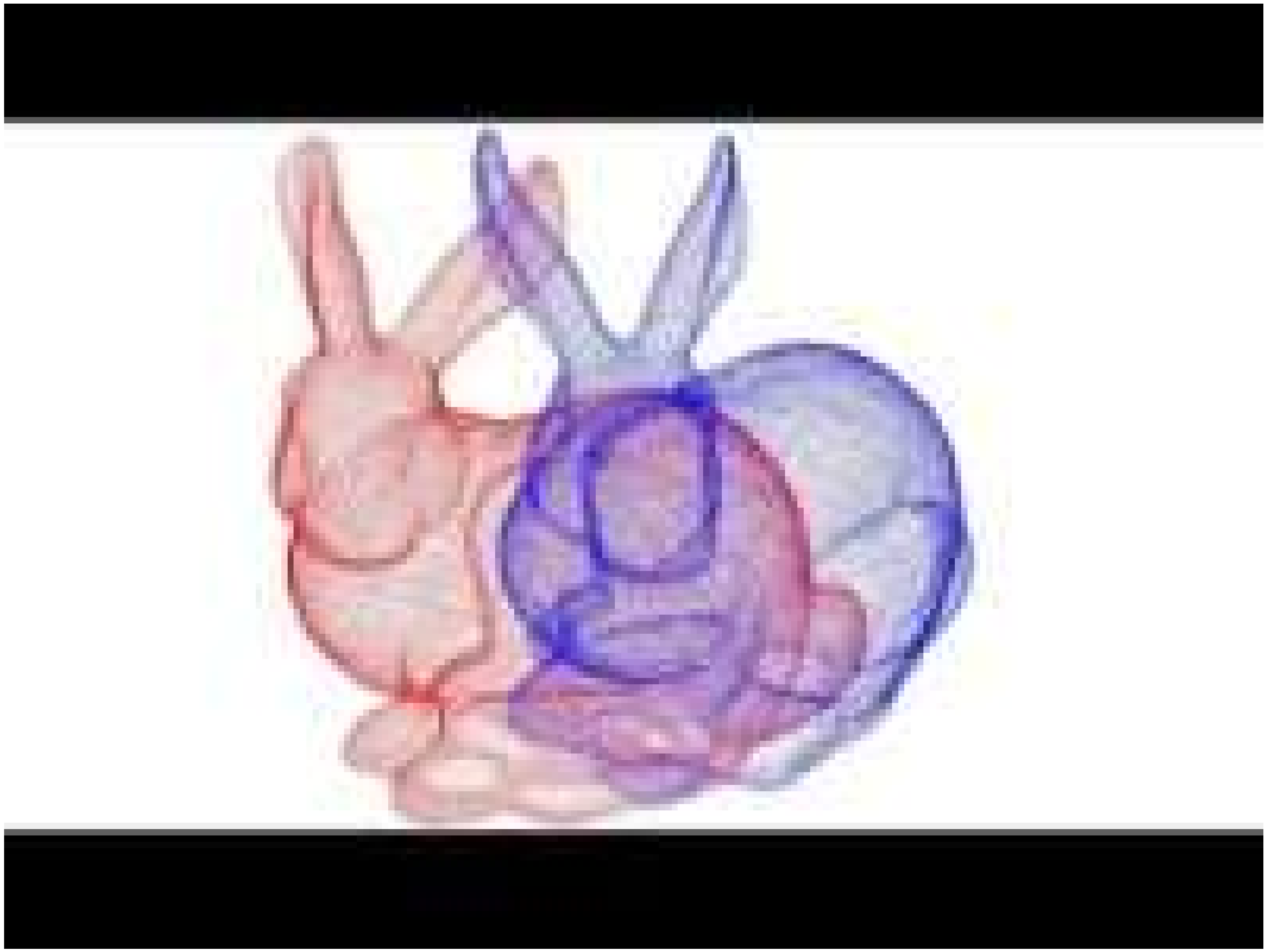
For k = 0, … until convergence:

1. **Estimate matches**: Let $y_{l_i} = argmin_j \left\| y_j - (R_k x_i + t_k) \right\|$ be the closest point in *Y* to the image of x_i under the current registration estimate $T_k = (t_k, R_k)$.

   If $\left\| y_{l_i} - (R_k x_i + t_k) \right\| \leq d$, accept $x_i \leftrightarrow y_{l_i}$ as a match

2. **Estimate registration**: Compute next registration estimate $T_{k+1} = (t_{k+1}, R_{k+1})$ by applying Horn's method to matches $\{x_i \leftrightarrow y_{l_i}\}$.

Iterative closest point



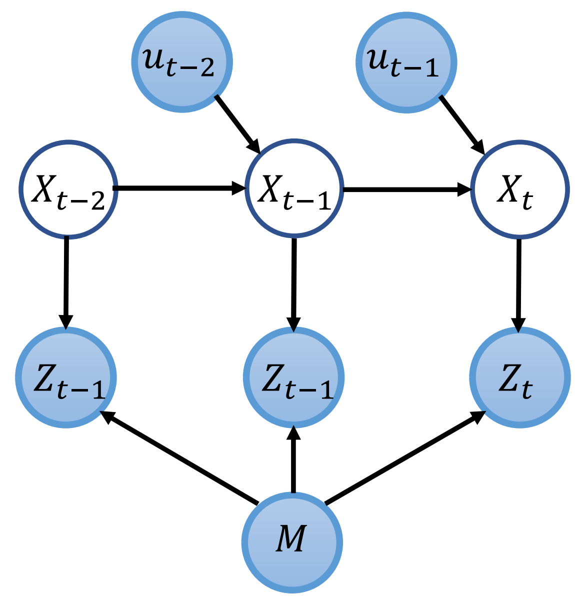
Odometry estimation using ROS's Canonical Scan Matcher

Monte Carlo Localization

Main idea: Monte Carlo localization (MCL) is simply localization using the Particle Filter

Given: Prior $p(x_0)$, map m, controls $u_{0:t}$, observations $z_{1:t}$
Estimate: Belief $p(x_t | m, z_{1:t})$ over the robot pose

Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):
- $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
- for $m = 1$ to M do
  - sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ ← Motion model
  - $w_t^{[m]} = p(z_t \mid x_t^{[m]})$ ← Sensing model
  - $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
- endfor
- for $m = 1$ to M do
  - draw i with probability $\propto w_t^{[i]}$
  - add $x_t^{[i]}$ to $\mathcal{X}_t$
- endfor
- return $\mathcal{X}_t$



Requires:
- A *sampler* for the motion model
- *Likelihood function* for the sensor model

Beam sensor likelihood model

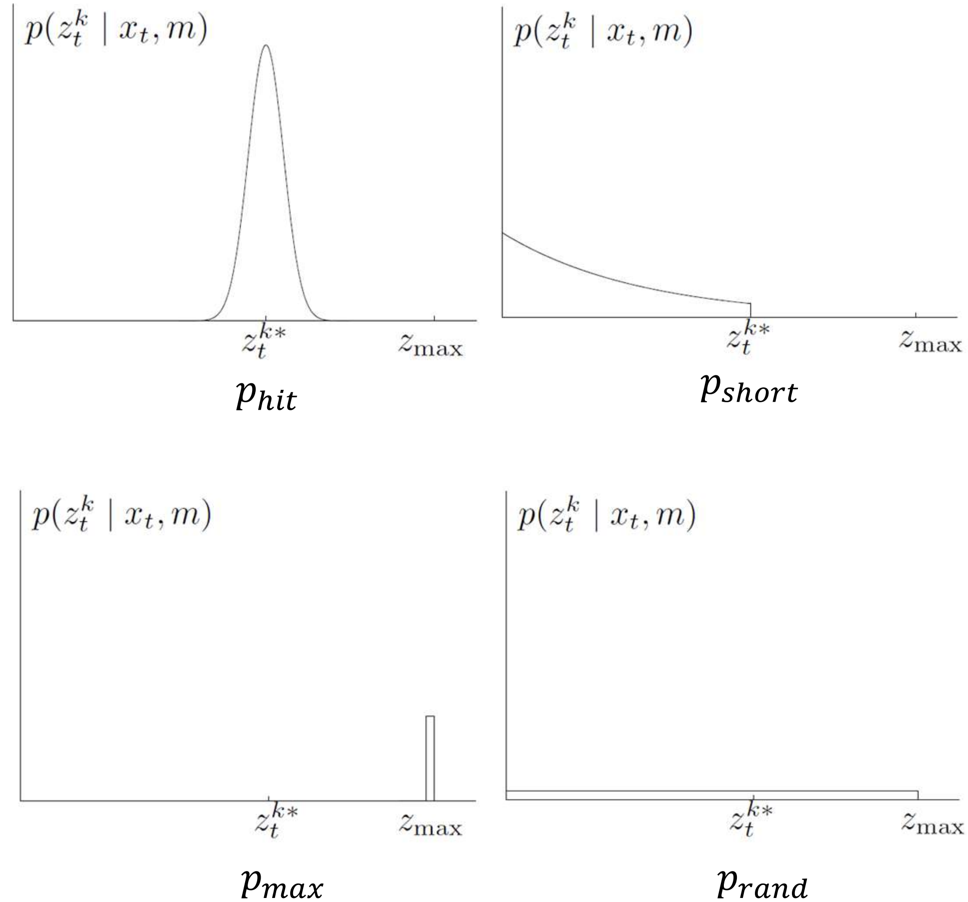
Last time: (Approximate) *inverse* sensor model: $p(m_i|x_t, z_t)$
⇒Approximate likelihood of *map* given *pose and measurement*

Today: *Forward* sensor model $p(z|x_t, m)$
⇒ Likelihood of *measurement* given *pose and map*

Beam sensor likelihood model

We consider a more physically-faithful sensor model that accounts for 4 distinct sources of error

- **Correct return w/ small measurement noise**
  - Nominal case for operation
  - Modeled as a Gaussian p_{hit} centered on true range
- **Unexpected / transient objects**
  - Corresponds to e.g. people moving in the scene
  - Causes a *short return*
  - Modeled as an *exponential* distribution p_{short} (one can justify this mathematically by appealing to e.g. survival analysis).
- **Failed detection**
  - Sensor does not detect a reflected beam – can be caused by e.g. specular reflection or absorption of the beam on target
  - Appears as a *maximum range* return
  - Can be modeled as a *point mass* (or very narrowly peaked distribution) p_{max} at the sensor's maximum range
- **Otherwise unexplained**
  - Catch-all category for other general weirdness
  - Modeled as a *uniform distribution* p_{rand} over the sensor's range



Beam sensor likelihood model

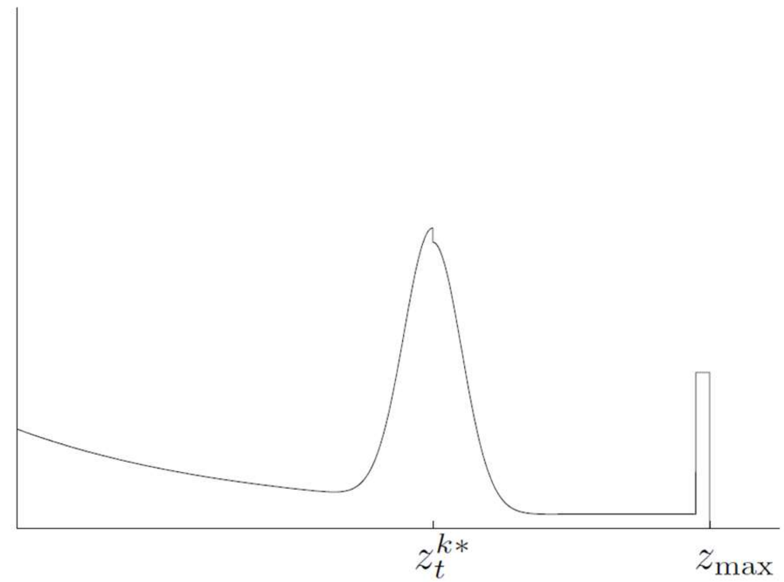
Our overall beam sensor model $p(z|x_t, m)$ is a *mixture* of these 4 components:

$$p(z|x_t, m) \\ = w_{hit}\, p_{hit} + w_{short} p_{short} + w_{max} p_{max} + w_{rand} p_{rand}$$

where w_{hit} , w_{short}, w_{max}, w_{rand} are the *mixture weights* (nonnegative and sum to 1).

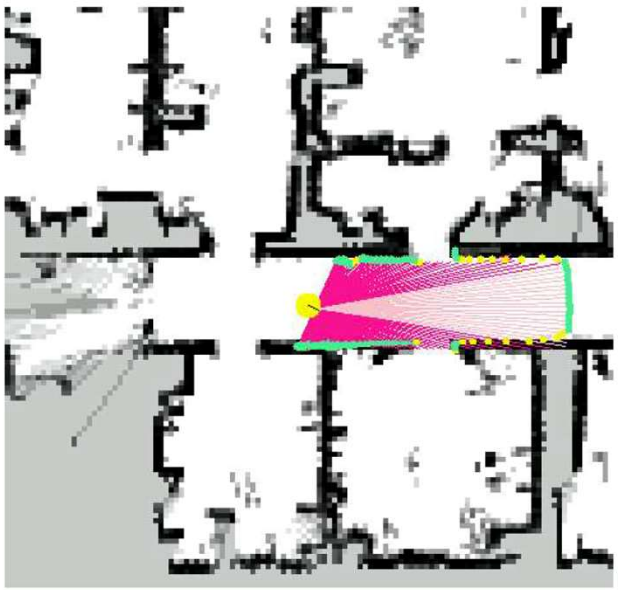
We also assume that each beam is sampled *independently*, so the joint probability for an entire (*n*-beam) scan is:

$$p(z_t|x_t, m) = \prod_{k=1}^{n} p(z_t^k|x_t, m)$$

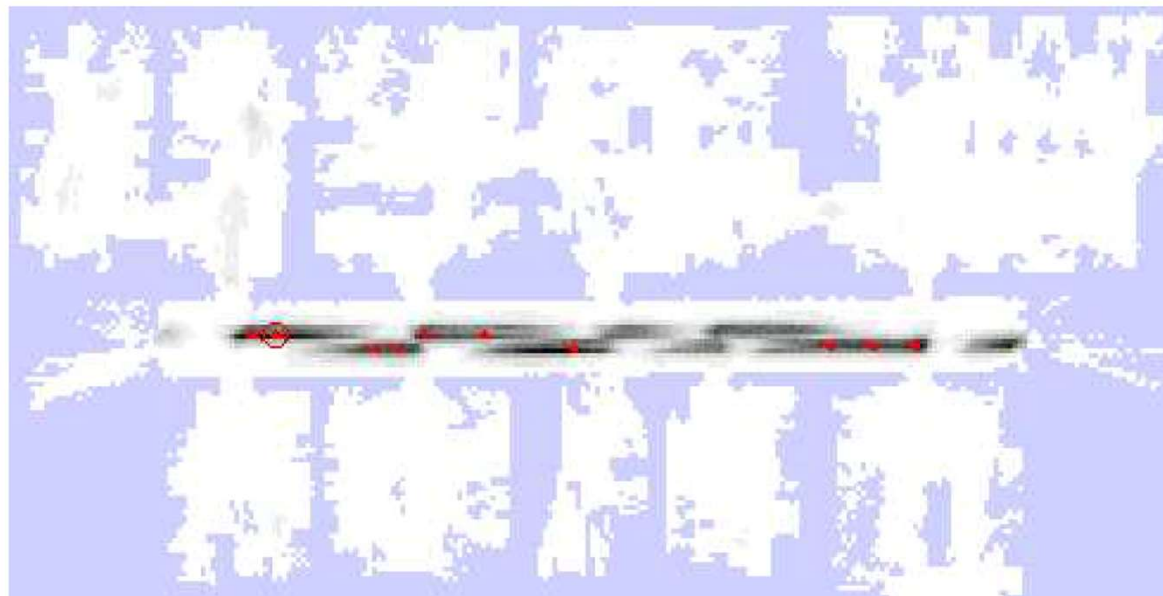


Complete (mixture) beam sensor model $p(z|x_t, m)$

Beam sensor measurement model



Laser scan (robot's true position)



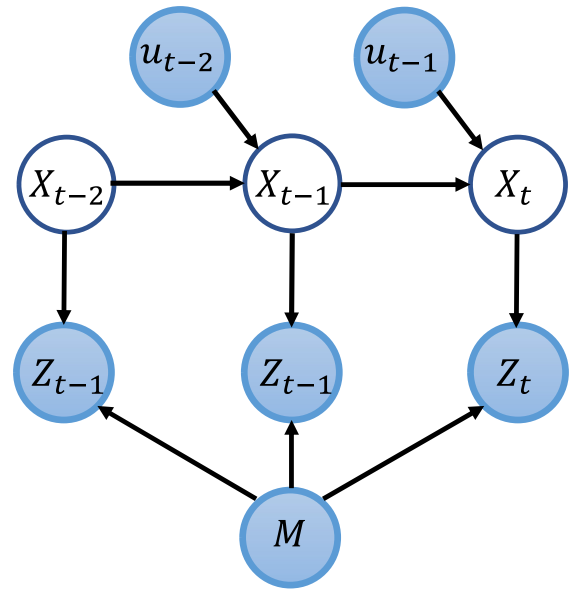
Heatmap for scan likelihood as a function of position

Monte Carlo Localization

Main idea: Monte Carlo localization (MCL) is simply localization using the Particle Filter

Given: Prior $p(x_0)$, map m, controls $u_{0:t}$, observations $z_{1:t}$
Estimate: Belief $p(x_t|\, m, z_{1:t})$ over the robot pose

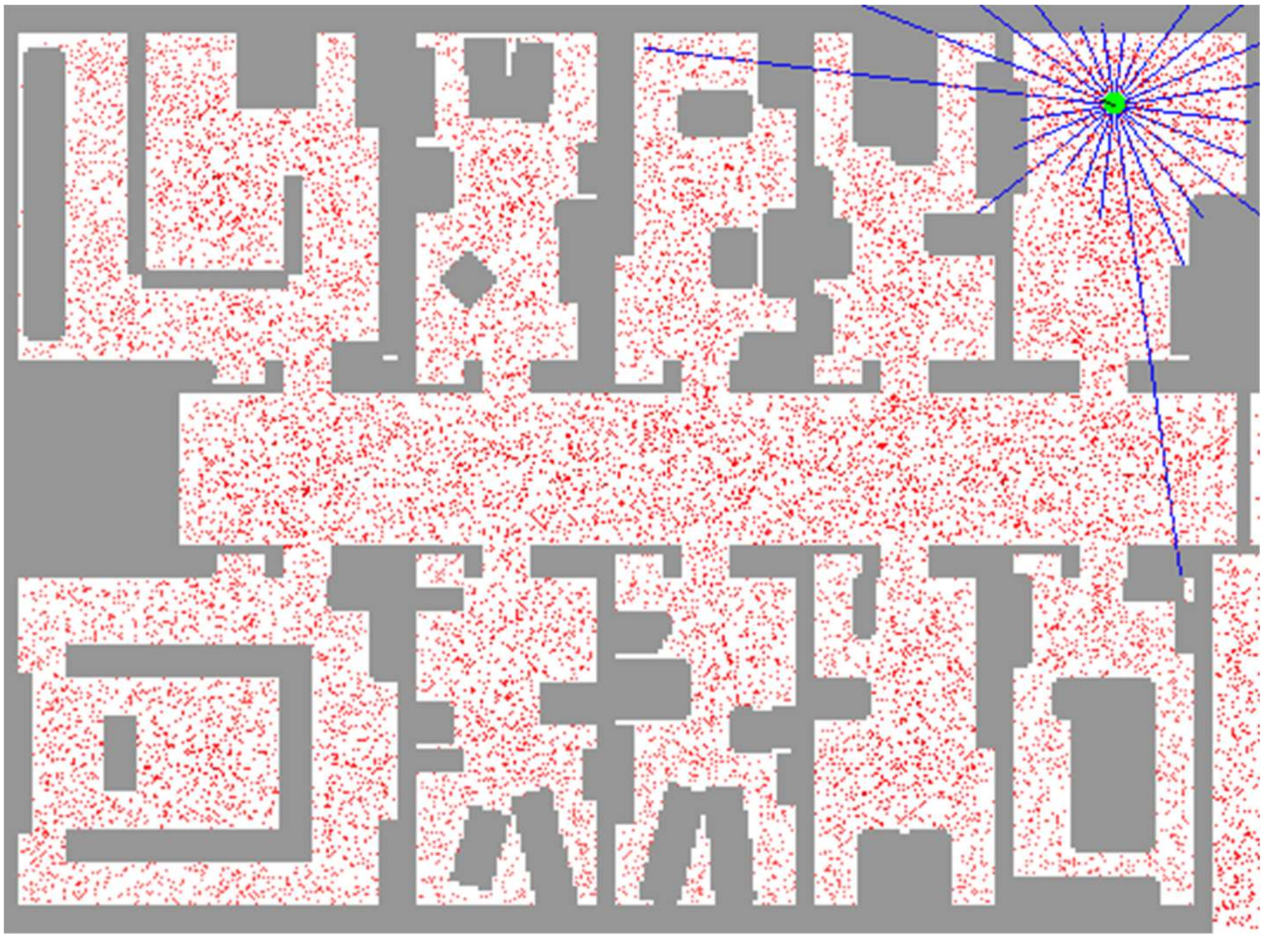
Algorithm Particle_filter($\mathcal{X}_{t-1}, u_t, z_t$):
$\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
for $m = 1$ to M do
sample $x_t^{[m]} \sim p(x_t \mid u_t, x_{t-1}^{[m]})$ ← Motion model
$w_t^{[m]} = p(z_t \mid x_t^{[m]})$ ← Sensing model
$\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
endfor
for $m = 1$ to M do
draw i with probability $\propto w_t^{[i]}$
add $x_t^{[i]}$ to $\mathcal{X}_t$
endfor
return $\mathcal{X}_t$



Requires:

- A *sampler* for the motion model
- *Likelihood function* for the sensor model

Monte Carlo localization



Monte Carlo localization on a RACECAR

Practicalities

Recall (from Lecture 10) two major issues with particle filters:

- **Particle depletion:** Need to preserve diversity in the particle set

⇒Augmented MCL: Method for increasing particle diversity

- **Computational cost:** Might need a **lot** of particles to model complex beliefs

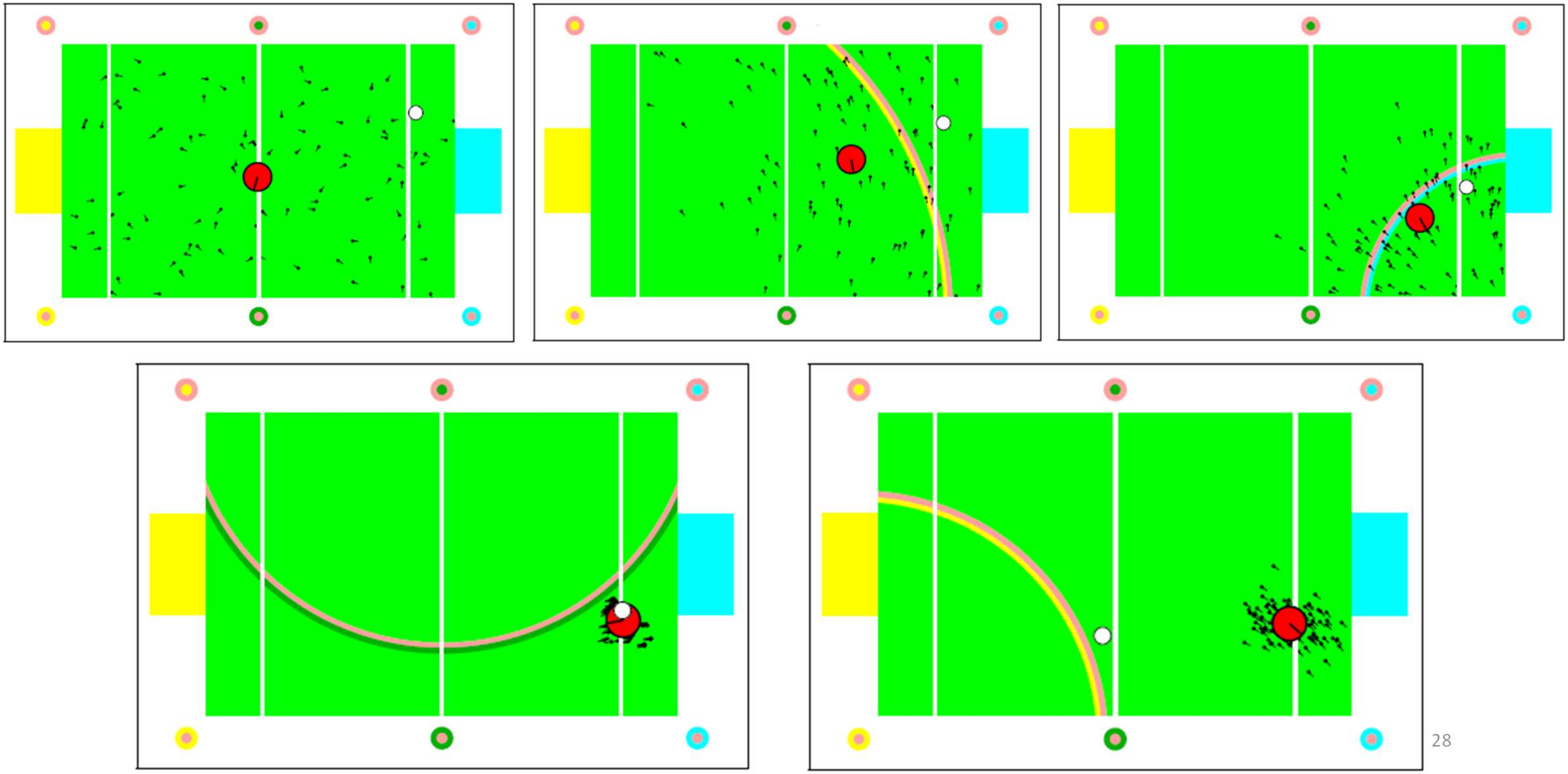
⇒Adaptive sampling: *Dynamically* adjust particle set size

The “kidnapped robot” problem

Def: A scenario in which a robot must relocalize itself after being (instantly) moved to an *arbitrary* location in a map

Ex: Repositioning a robot that is powered off (Moving a Roomba up / down a floor)

Particle depletion and the kidnapped robots



Preserving particle diversity: Augmented MCL

Problem: We need to preserve *diversity* in the filter's particle set

⇒Need to have particles near the true state to recover from localization failures

Idea: We can increase diversity by resampling a small fraction of (purely) *random* particles

Approach: Use *measurement likelihood* (weights) as a measure of localization quality

⇒Sudden decrease in avg particle likelihood vs. historical avg may indicate localization failure

⇒In that case, we should increase diversity of particle set

Algorithm Augmented_MCL($\mathcal{X}_{t-1}, u_t, z_t, m$):

static $w_{\text{slow}}, \ w_{\text{fast}}$

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

for $m = 1$ to M do

$\quad x_t^{[m]} = \textbf{sample\_motion\_model}(u_t, x_{t-1}^{[m]})$

$\quad w_t^{[m]} = \textbf{measurement\_model}(z_t, x_t^{[m]}, m)$

$\quad \bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$

$\quad w_{\text{avg}} = w_{\text{avg}} + \frac{1}{M} \, w_t^{[m]}$

endfor

$w_{\text{slow}} = w_{\text{slow}} + \alpha_{\text{slow}}(w_{\text{avg}} - w_{\text{slow}})$

$w_{\text{fast}} = w_{\text{fast}} + \alpha_{\text{fast}}(w_{\text{avg}} - w_{\text{fast}})$

for $m = 1$ to M do

$\quad$ with probability $\max(0.0, \ 1.0 - w_{\text{fast}}/w_{\text{slow}})$ do

$\quad\quad$ add random pose to $\mathcal{X}_t$

$\quad$ else

$\quad\quad$ draw $i \in \{1, \ldots, N\}$ with probability $\propto w_t^{[i]}$

$\quad\quad$ add $x_t^{[i]}$ to $\mathcal{X}_t$

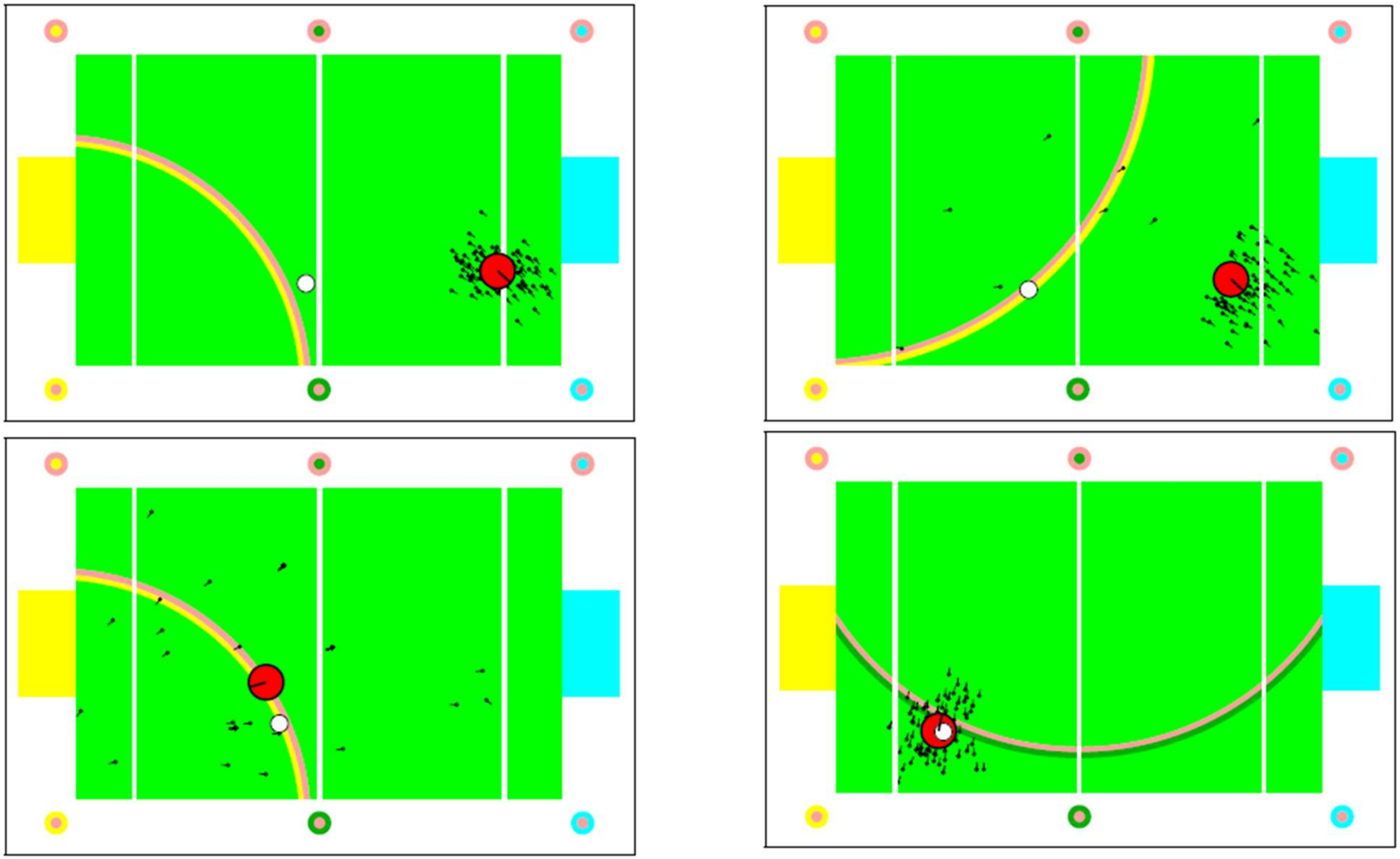
$\quad$ endwith

endfor

return $\mathcal{X}_t$

Table 8.3 An adaptive variant of MCL that adds random samples. The number of random samples is determined by comparing the short-term with the long-term likelihood of sensor measurements.

Example: Kidnapped robot revisited



KLD-sampling

KLD-sampling is a method for *dynamically adjusting* the number of particles that we maintain in the sample set

Main idea: The number of particles that we need is related to the *uncertainty* in our current belief:

More uncertainty ⇒ belief is more “spread out” ⇒ need more particles to model

Approach: *Dynamically adjust* the number of particles based upon an estimate of the Kullback-Leibler divergence between particle set and true posterior

KLD-Sampling

Algorithm KLD_Sampling_MCL($\mathcal{X}_{t-1}, u_t, z_t, m, \varepsilon, \delta$):
  $\mathcal{X}_t = \emptyset$
  $M = 0,\ M_\chi = \infty,\ k = 0$
  for all b in H do
    $b = empty$
  endfor
  do
    draw i with probability $\propto w_{t-1}^{[i]}$
    $x_t^{[M]}$ = **sample_motion_model**$(u_t, x_{t-1}^{[i]})$
    $w_t^{[M]}$ = **measurement_model**$(z_t, x_t^{[M]}, m)$
    $\mathcal{X}_t = \mathcal{X}_t + \langle x_t^{[M]}, w_t^{[M]} \rangle$
    if $(x_t^{[M]}$ *falls into empty bin* $b)$ *then*
      $k = k + 1$
      $b = non\text{-}empty$
      if $(k > 1)$ *then*
$$M_\chi := \frac{k-1}{2\varepsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}\, z_{1-\delta}\right\}^3$$
    endif
    $M = M + 1$
  while $(M < M_\chi)$
  return $\mathcal{X}_t$