LIACC

U.PORTO
FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

INESCTEC

Localization

Luís Paulo Reis

lpreis@fe.up.pt
Director/Researcher LIACC
Associate Professor at FEUP/DEI

Armando Sousa

asousa@fe.up.pt
Researcher INESC-TEC
Assistant Professor at FEUP/DEEC

Also, a very special Thank You to: ***Prof. Nuno Lau, IEETA, U. Aveiroca***

Background

- **Localization** – Where am I?

- **Mapping** – My (dynamic?) surroundings

- **Navigation** – How do I get where I want to go?

- **TREND**: Don’t separate them!

=> SLAM: Simultaneous Localization and Mapping

The Localization Problem

- **Also called Position Estimation**
- **Problem**
  - Inputs:
    - Map of the environment
    - Perceptions and actions of robot
  - The robot must determine its position relative to the map
    - Usually this can be expressed as the pose of the robot (x, y, θ)
- **The pose cannot be sensed directly**

The Localization Problem

- **The pose cannot be sensed directly**
- **The pose has to be inferred from data**
- **A single sensor measurement is usually insufficient to determine pose**
- **Robot has to integrate data over time**

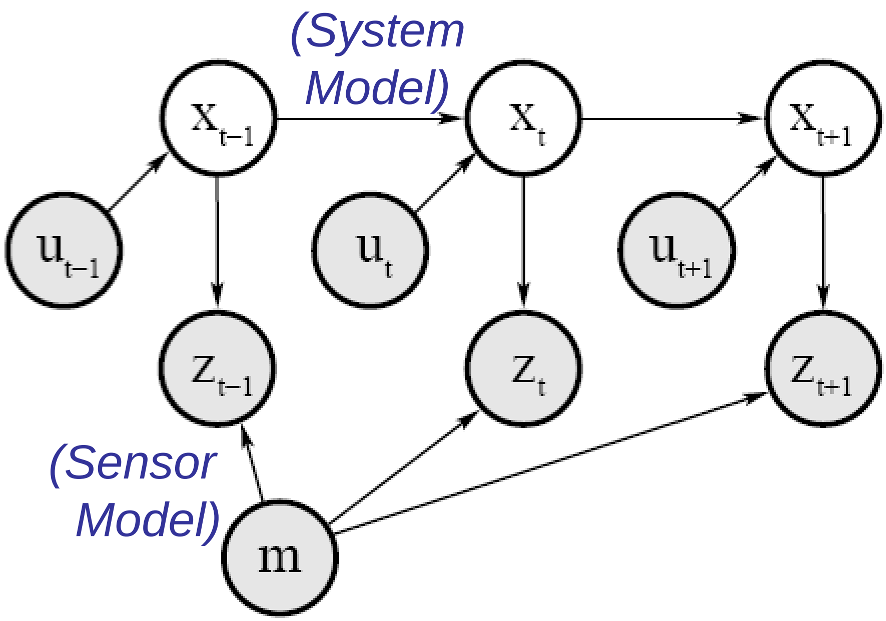
Taxonomy for Localization

- **Local vs. Global Localization**
  - Position tracking
    - Initial pose is known
    - Local uncertainty around robot's true pose
  - Global localization
    - Initial pose is unknown
  - Kidnapped robot
    - Robot can be teleported to different location

Taxonomy for Localization

- **Static vs. Dynamic Environments**
- **Passive vs. Active Approaches**
- **Single-Robot vs. Multi-Robot**

Localization Problem



- **x** are the estimated positions (path)
- **u** are the applied actions (controls)
- **z** are the sensor measures (observations)
- **m** is the map

Markov Localization

- **Probabilistic state estimation applied to localization problem**
- **Markov assumption**
  - Sensor measures do not depend on previous measures if position is known
  - Position is the only state
- **Belief function is the probability distribution of the estimated position of the robot for every possible position**

Markov Localization

- **Based on the Bayes Filter:**
  - Prediction

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

  - Correction

$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$$

Markov Localization

- **Based on the Bayes Filter:**
  - Prediction

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

  - Incorporates motion model
  - Input:
    - Previous position: $\mathbf{x_{t-1}}$
    - Action taken: $\mathbf{u_t}$
    - Previous Belief distribution: $\mathbf{bel(x_{t-1})}$
  - How does $\mathbf{u_t}$ changes **bel**?

Markov Localization

- **Based on the Bayes Filter:**
  - Prediction

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\, bel(x_{t-1})\, dx_{t-1}$$

  - How does $\mathbf{u_t}$ changes **bel**?
  - All possible x values have to be considered on their probability of transition to x_t
  - Discrete case: consider a grid world with 2 cells A,B

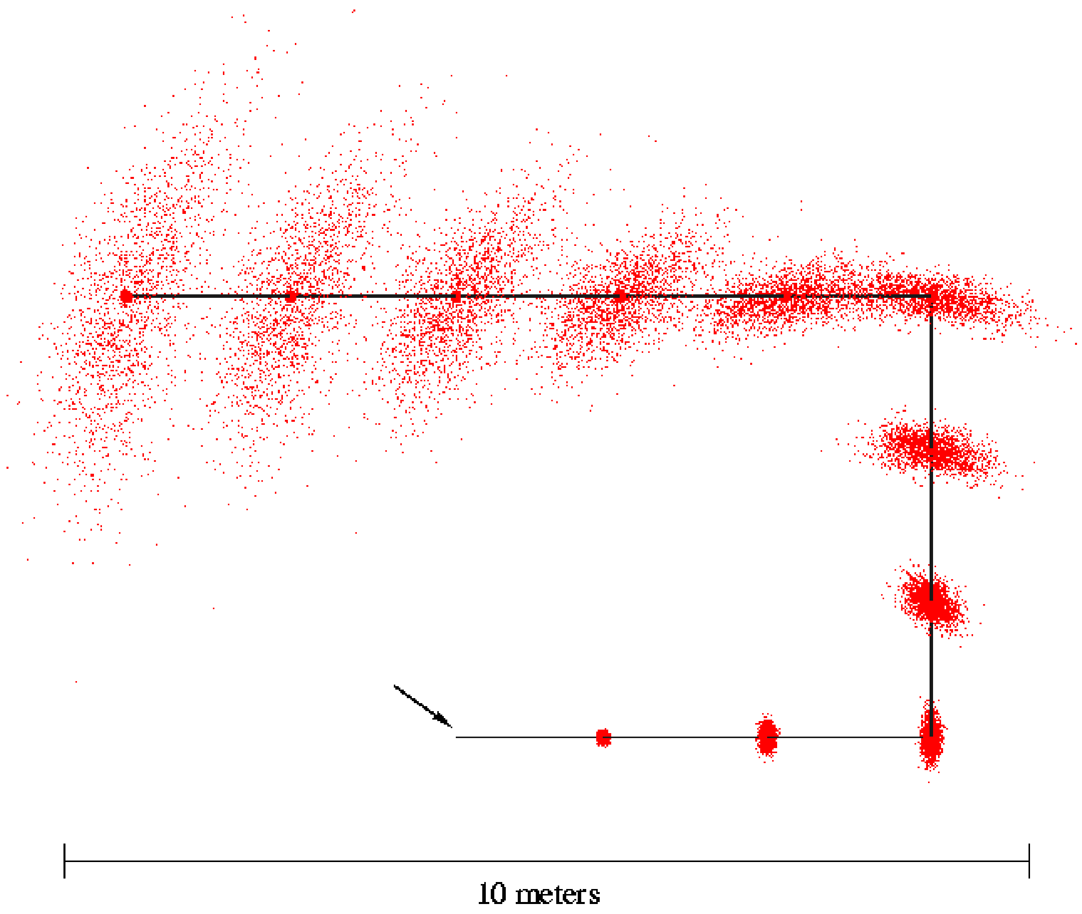
    P(A | A, left)=0,9 P(B | A, left) =0,1

    P(A | B, left)=0,8 P(B | B, left) =0,2

    if P(A) = 0,3 and P(B)=0,7 , which is P(B) after left?

Markov Localization

- **Motion model**



Markov Localization

- **Based on the Bayes Filter:**
  - Correction

$$bel(x_t) = \eta\, p(z_t \mid x_t)\, \overline{bel}(x_t)$$

  - Integration of sensor data

    $P(x_t \mid z_t)$
  - Using Bayes Formula

$$P(x_t \mid z_t) = \frac{P(z_t \mid x_t) * p(x_t)}{p(z_t)}$$

  - $P(z_t)$ does not depend on x and may be substituted by constant

Markov Localization

- **Algorithm:**

Algorithm Markov_localization($bel(x_{t-1}), u_t, z_t, m$**):**

for all x_t do

$$\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1}, m)\ bel(x_{t-1})\ \ dx_{t-1}$$

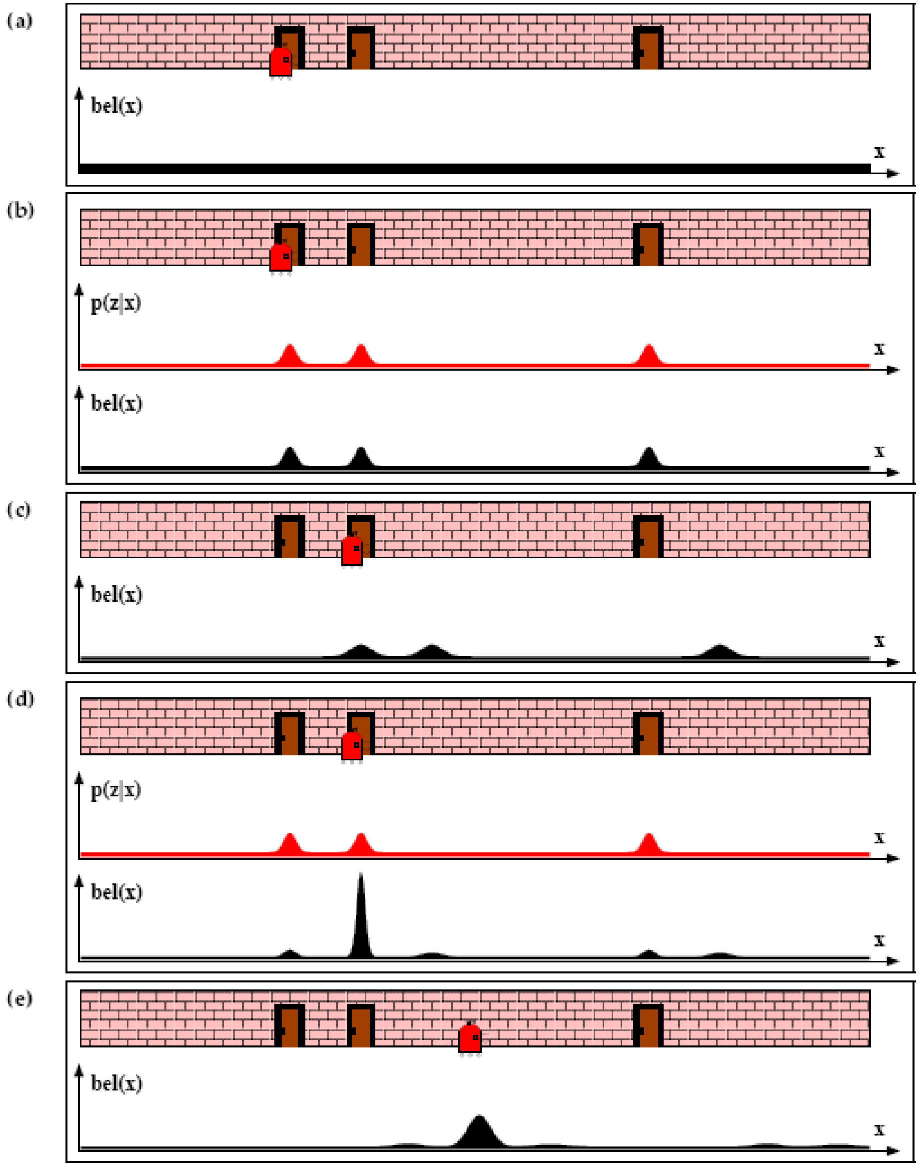
$$bel(x_t) = \eta\ p(z_t \mid x_t, m)\ \overline{bel}(x_t)$$

endfor

return $bel(x_t)$

Markov Localization

a) Belief is uniform
b) First integration of sensor data, result is multimodal
c) Convolution with motion model, shifts and flattens belief
d) Second integration of sensor data, robot localizes itself
e) Moving along



EKF Localization

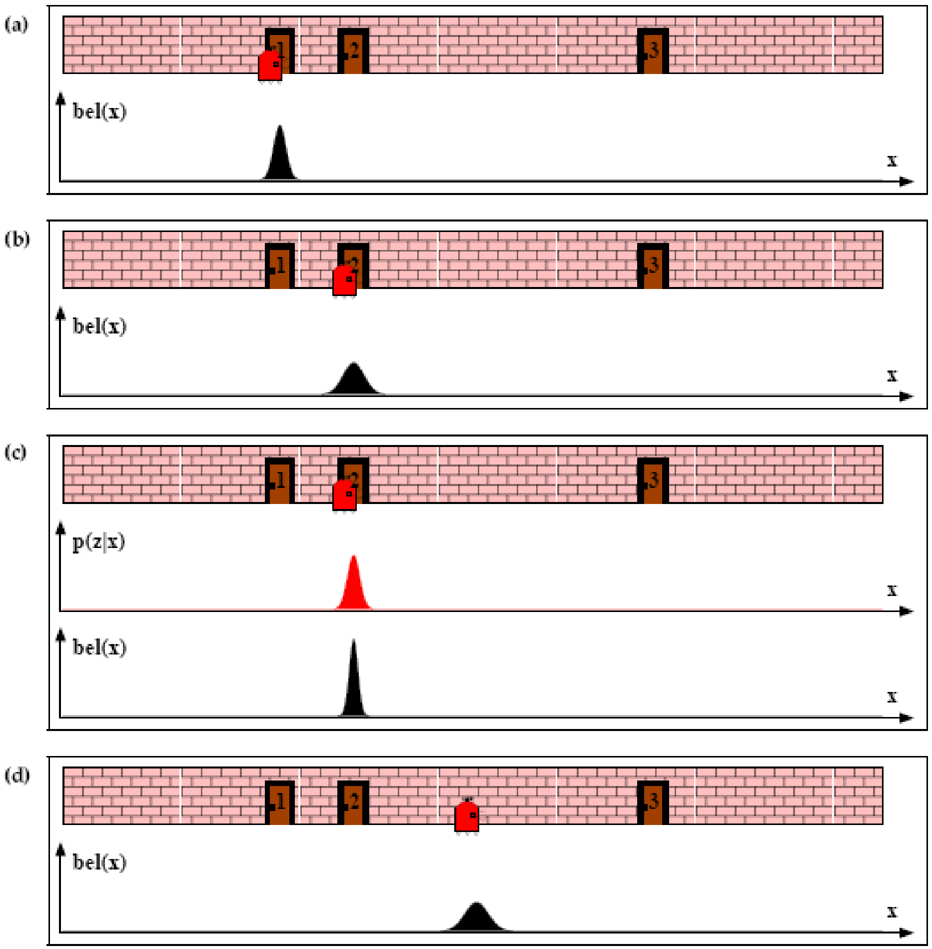
Extended Kalman Filter Localization

- **Special case of Markov Localization**
- **Beliefs are represented by mean and covariance (Gaussian)**
  - Belief shape is unimodal

- **Consider the problem of localization in a map in which features are identifiable**
  - Correspondence variables

EKF Localization

a) Initial belief is a Gaussian distribution
b) Motion model is applied
c) Sensor data is integrated resulting variance is smaller than variances of belief and sensor model
d) Motion uncertainty



Kalman Filter

- **Integration of measures over time**
- **Markovian assumption**
- **Considers physics model and action model**

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \qquad z_t = C_t x_t + \delta_t$$

- **Forecast step**

$$\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$$
$$\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t$$

- **Information**

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

Kalman Filter

A_t Matrix (nxn) that describes how the state evolves from t to $t\text{-}1$ without controls or noise.

B_t Matrix (nxl) that describes how the control u_t changes the state from t to $t\text{-}1$.

C_t Matrix (kxn) that describes how to map the state x_t to an observation z_t.

ε_t

δ_t

Random variables representing the process and measurement noise that are assumed to be independent and normally distributed with covariance R_t and Q_t respectively.

Extended Kalman Filter

- **Using Kalman Filter for nonlinear functions**
- **Linearization: Taylor expansion**
- **Considers physics model and action model**

$$x_t = g(u_t, x_{t-1}) \qquad z_t = h(x_t)$$

- **Forecast step**

$$\bar{\mu}_t = g(u_t, \mu_{t-1})$$
$$\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$$

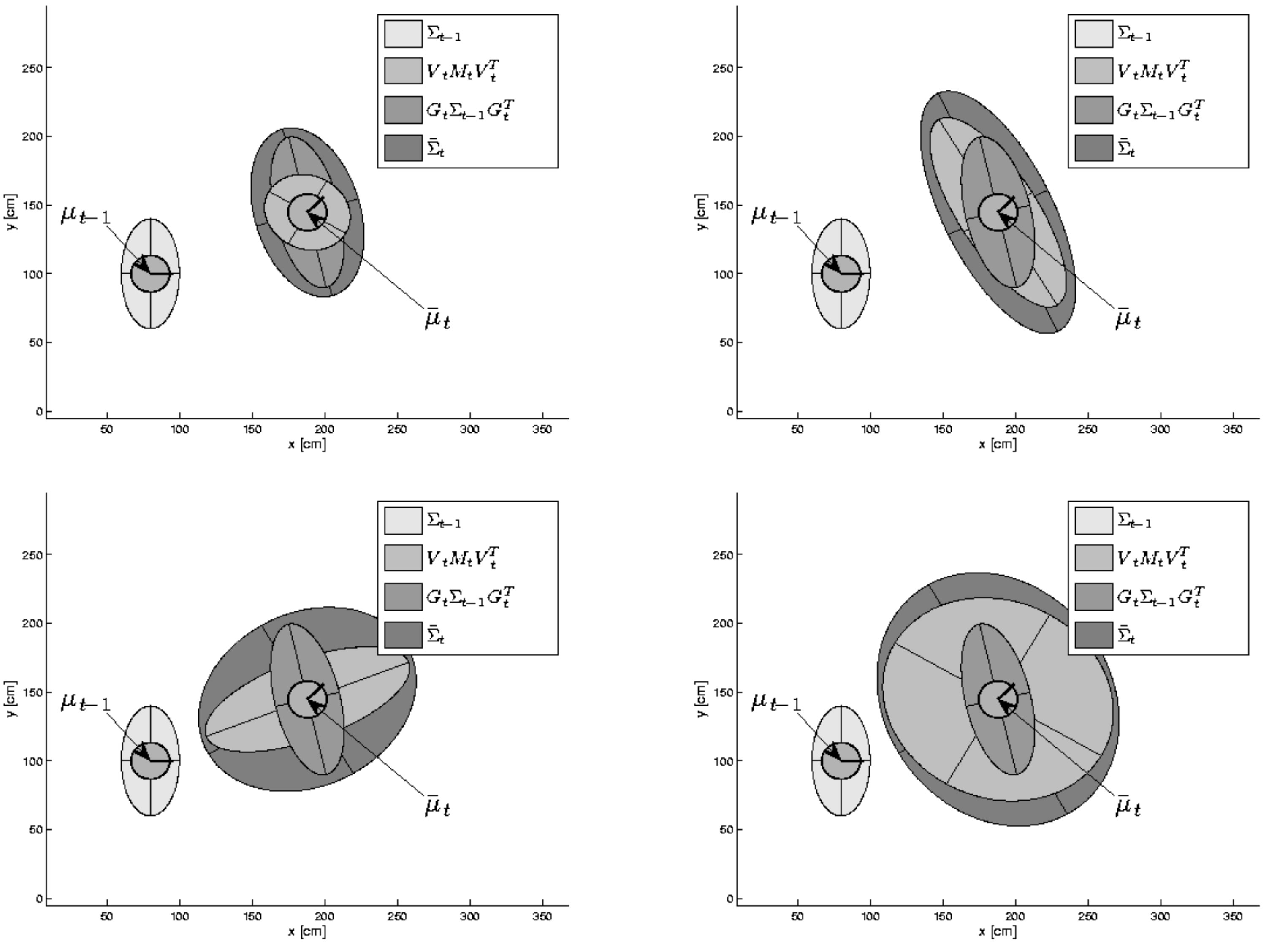
$$G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

- **Information**

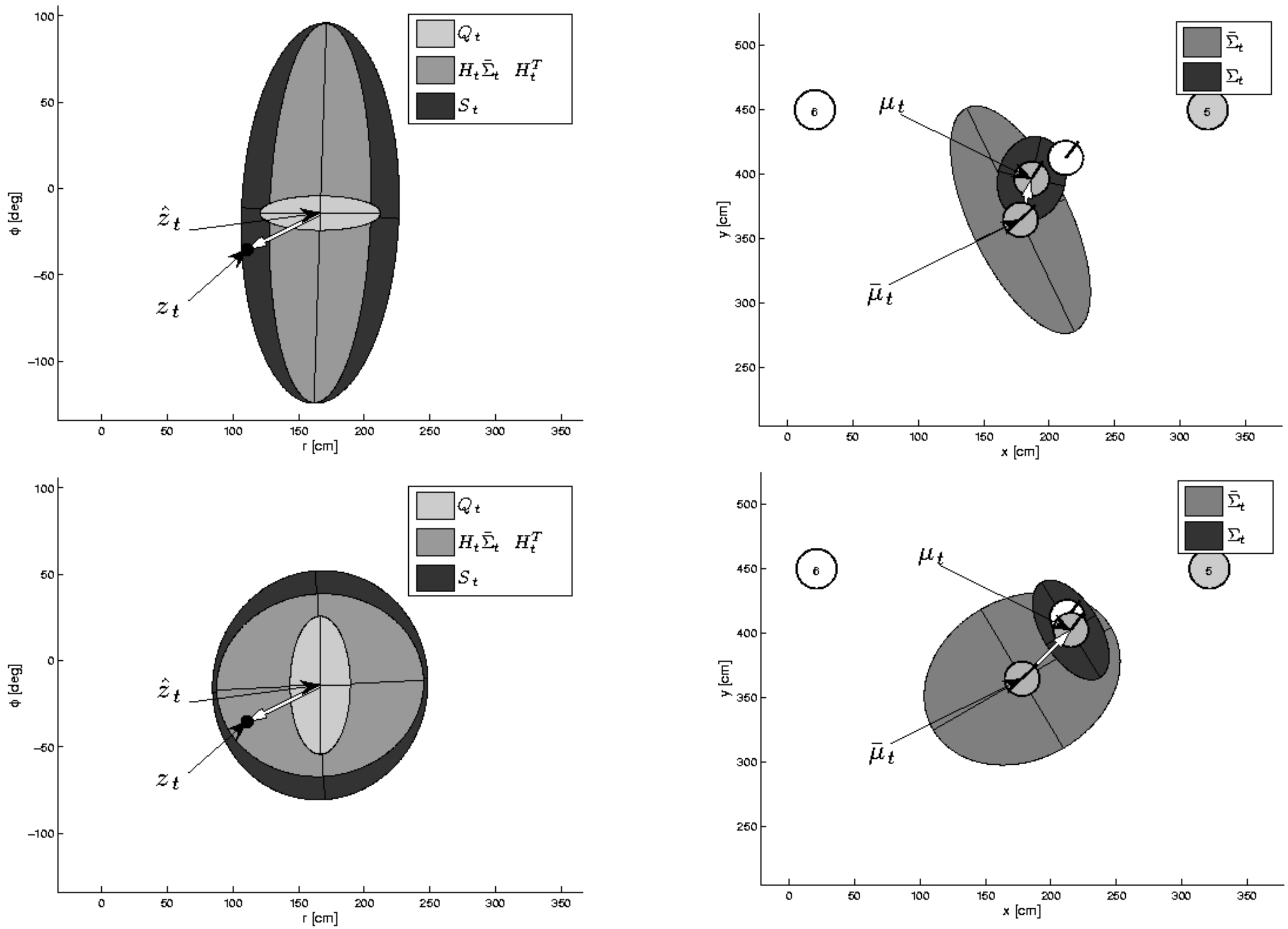
$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t}$$

$$K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$$
$$\mu_t = \bar{\mu}_t + K_t (z_t - h(\bar{\mu}_t))$$
$$\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$$

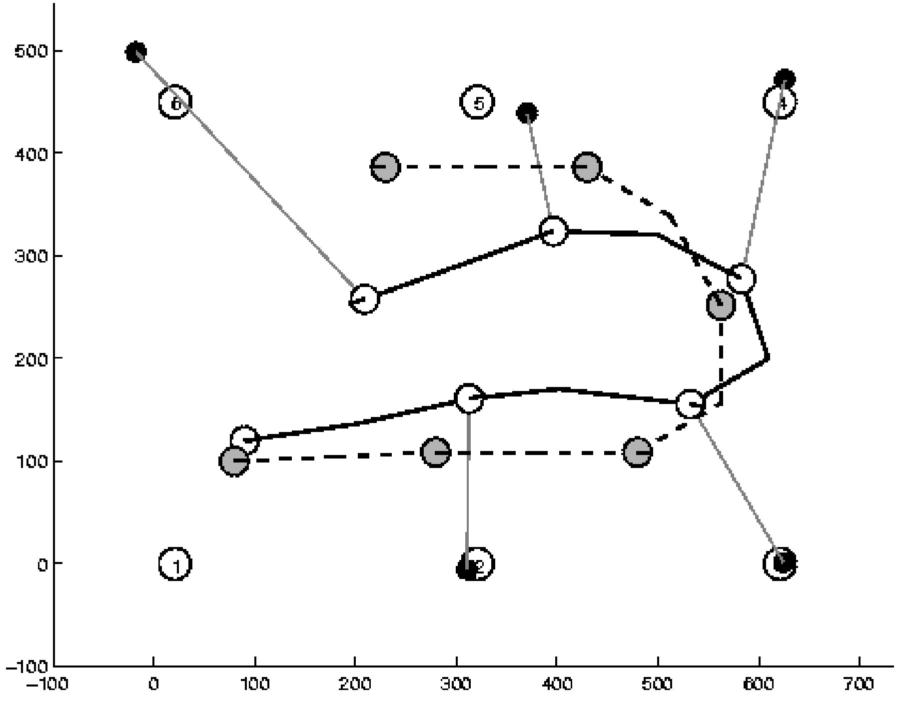
EKF Prediction Step



EKF Correction Step



Estimation Sequence



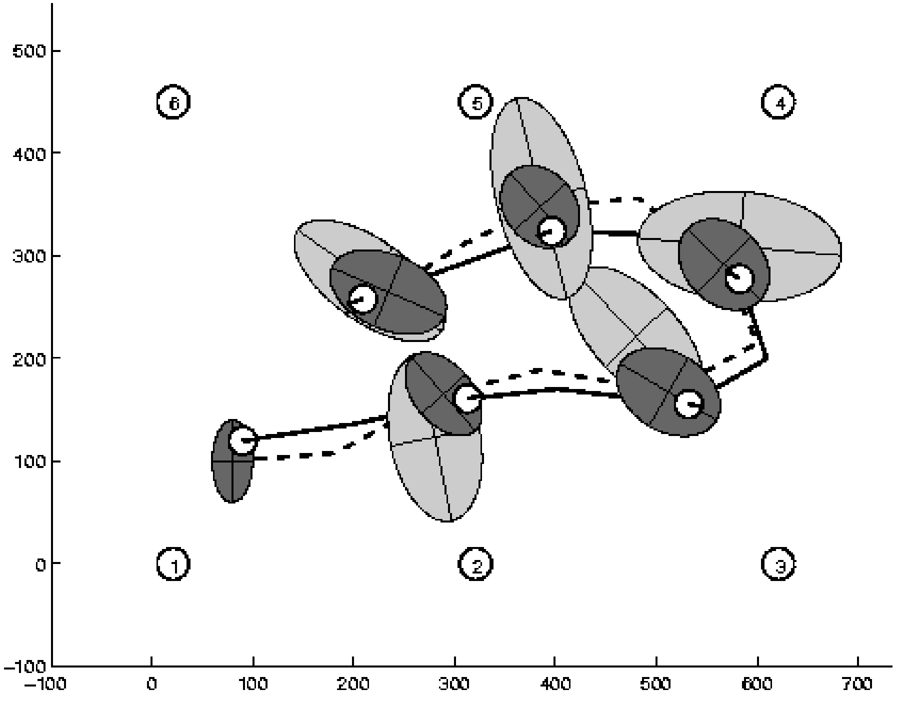


Tabela 5.1 Resumo da operação do Filtro de Kalman

Previsão	***Actualização***
Propagar o estado do sistema: $x(k^-) = \Phi\, x(k-1) + Bu(k-1)$ Propagar a covariância: $P(k^-) = \Phi\, P(k-1)\Phi^T + Q(k)$	Calcular o ganho de Kalman: $K(k) = P(k^-)H^T\left(HP(k^-)H^T + R(k)\right)^{-1}$ Actualizar estado com a medida: $x(k) = x(k^-) + K(k)\left(y(k) - H\,x(k^-)\right)$ Actualizar a covariância $P(k) = \left(I - K(k)H\right)P(k^-)$

Estimativas Iniciais

x_0 e P_0

www.fe.up.pt/~robosoc/sci/PhD_Armando_Sousa_lo_qual.pdf

Tabela 5.2 Resumo da operação do EKF

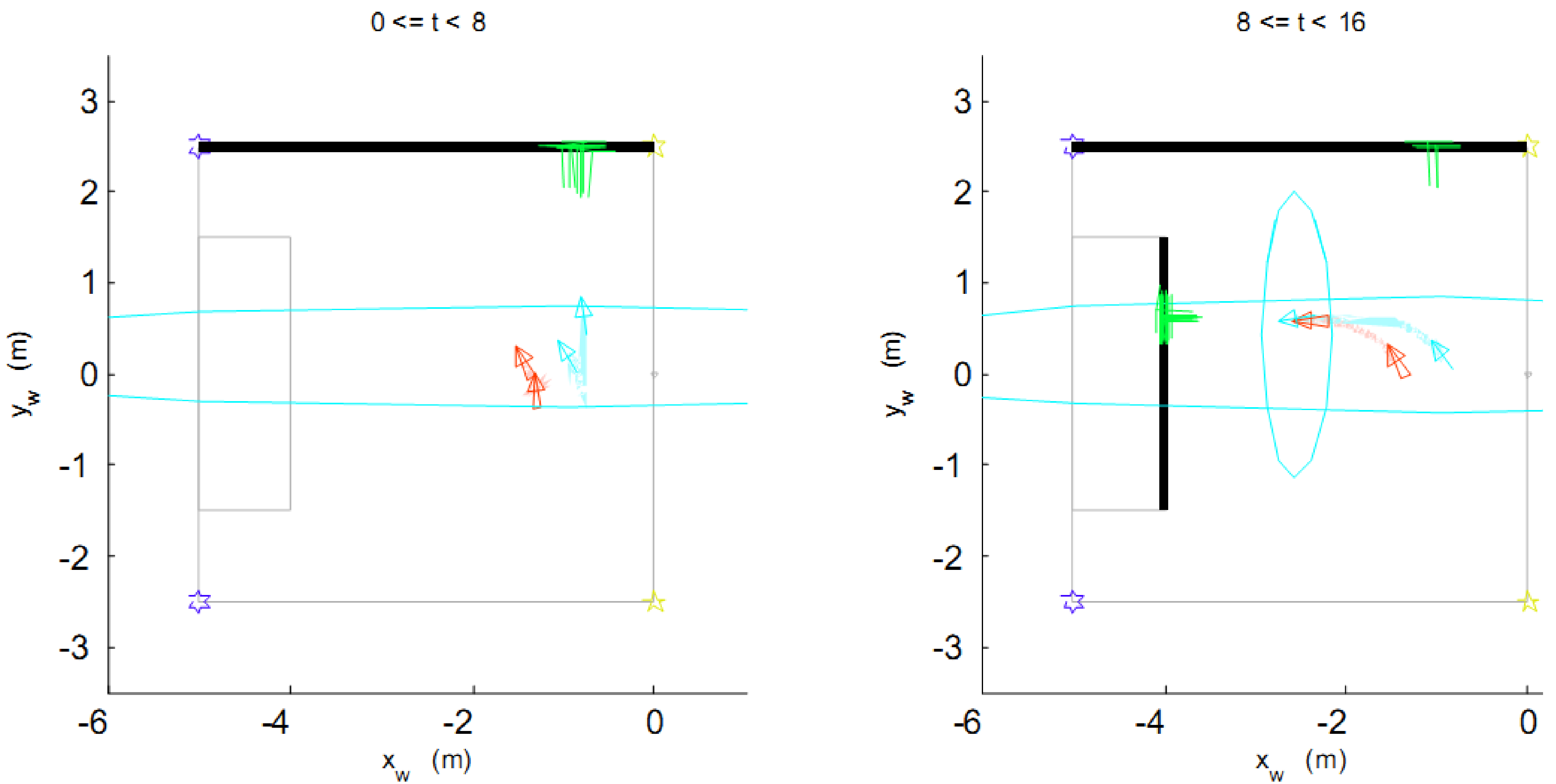
Previsão	***Actualização***
Propagar o estado do sistema: $\hat{x}(k^-) = f(\hat{x}(k-1), u(k), 0)$ Propagar a Covariância: $P(k^-) = (A(k)P(k-1)A(k)^T + W(k)Q(k-1)W^T(k))$	Calcular o Ganho de Kalman K(k): $L(k) = (H(k)P(k^-)H(k)^T + V(k)R(k)V(k)^T)^{-1}$ $K(k) = P(k^-)H(k)^T L(k)$ Actualizar estado com a medida: $x(k) = x(k^-) + K(k)(y(k) - h(x(k^-), 0))$ Actualizar Covariância $P(k) = (I - K(k)H(k))P(k^-)$

Estimativas Iniciais

x_0 e P_0

www.fe.up.pt/~robosoc/sci/PhD_Armando_Sousa_lo_qual.pdf

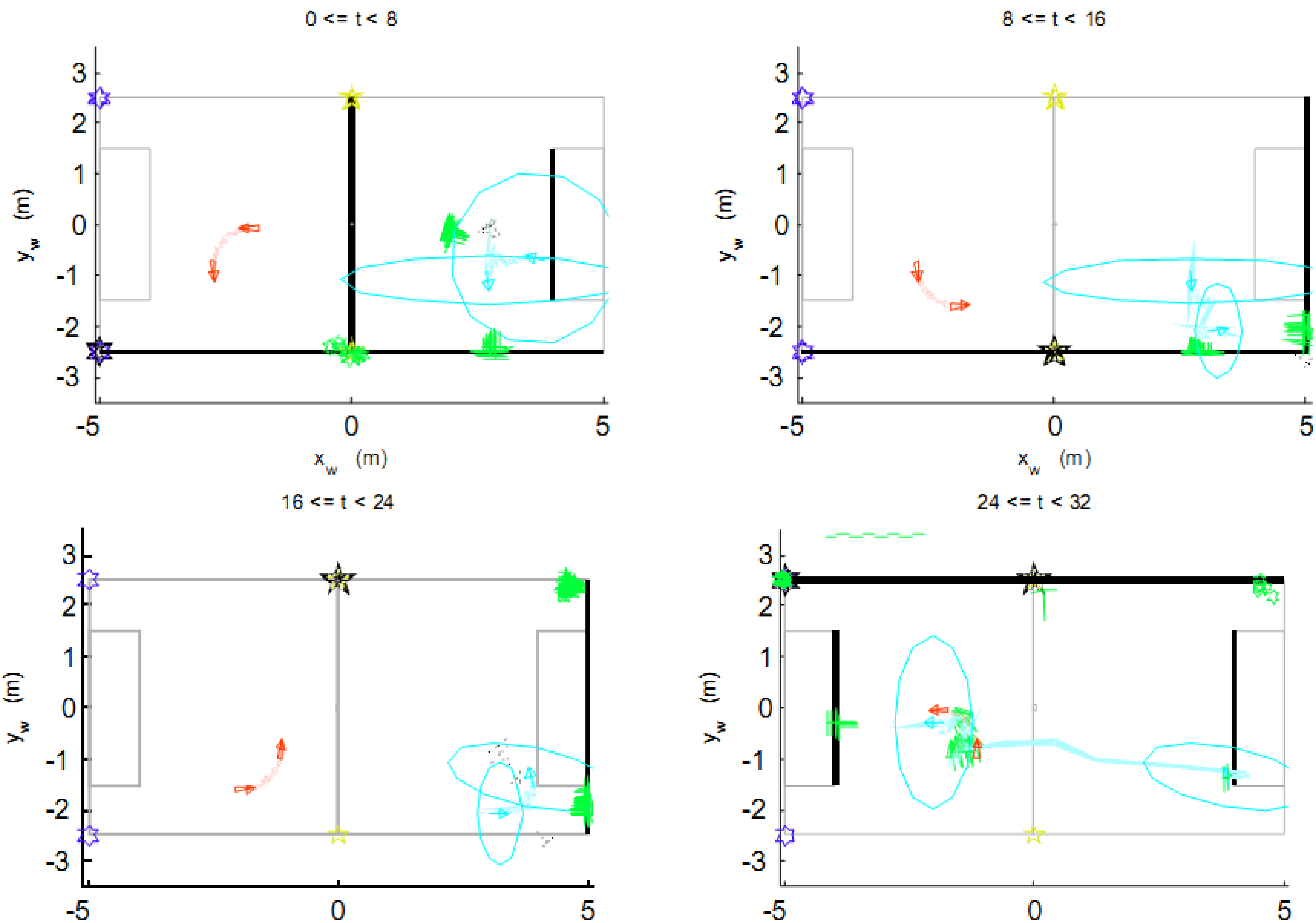
Exemplo EKF

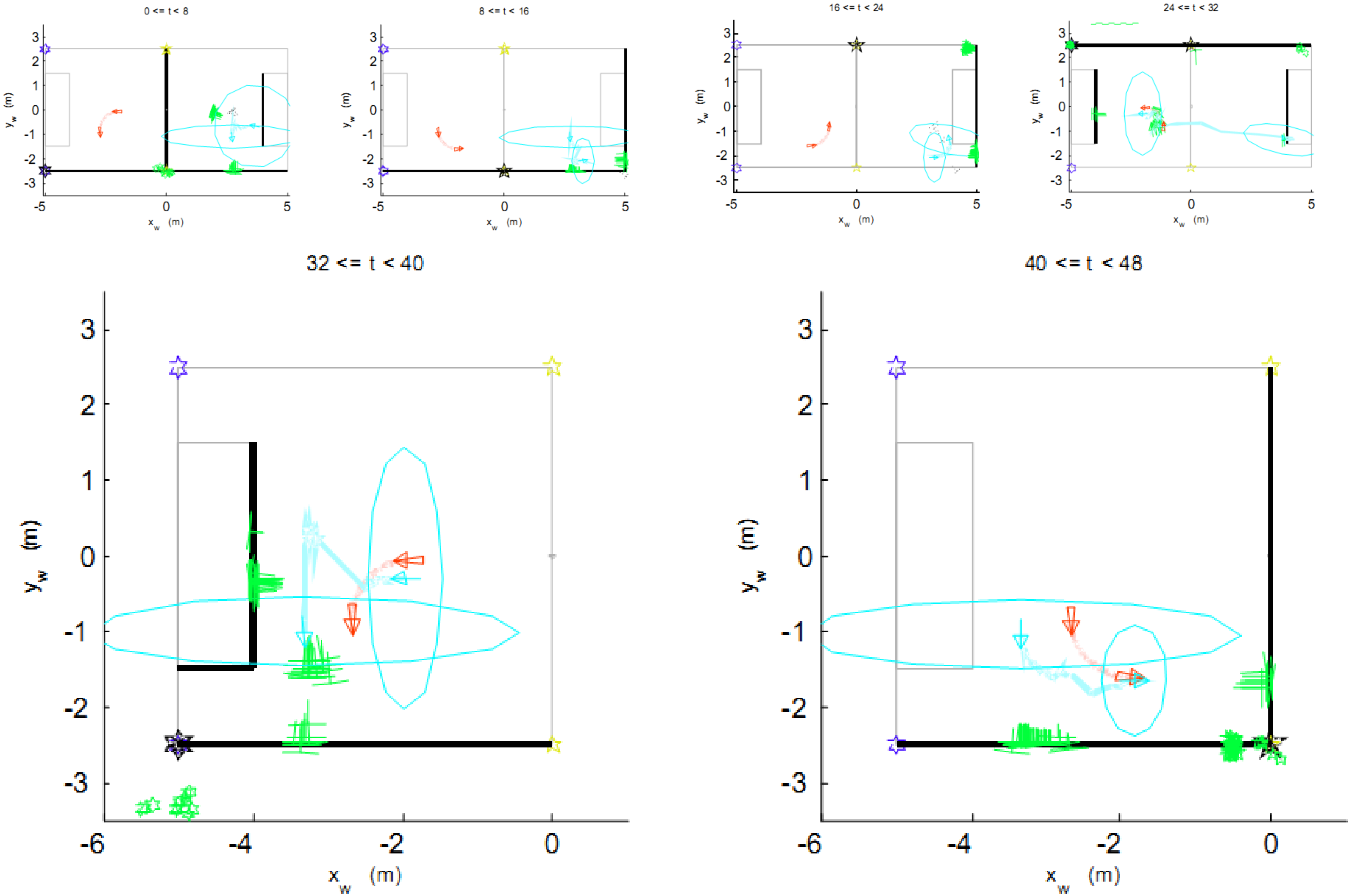


ura 8.72 Teste dinâmico na localização por Linhas (tempos em segundos)

www.fe.up.pt/~robosoc/sci/PhD_Armando_Sousa_lo_qual.pdf

Localiz EKF (Linhas + Postes + CB)

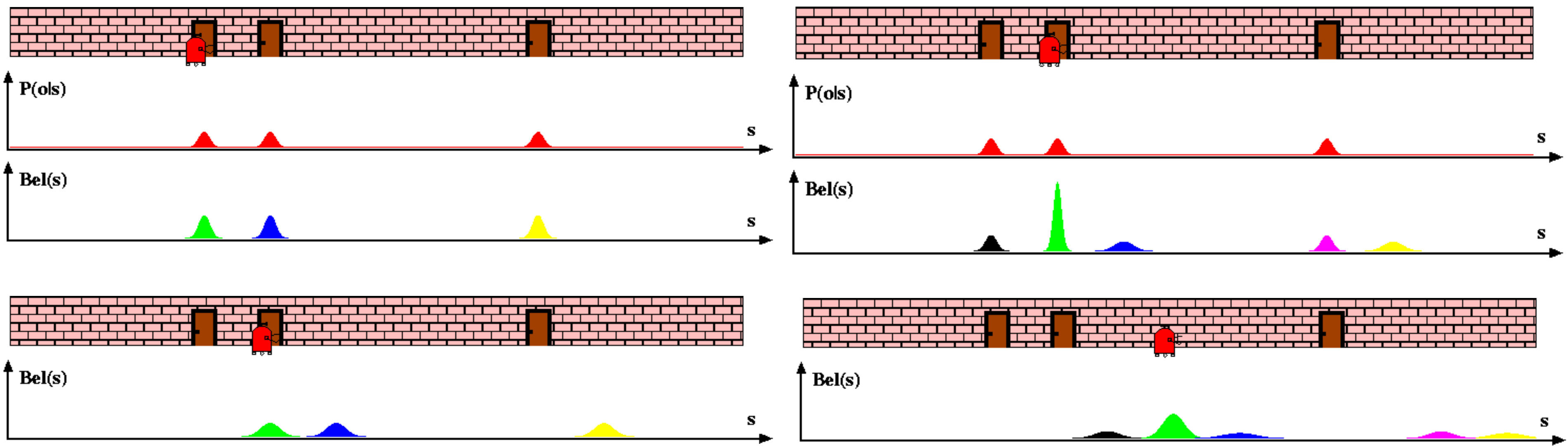




www.fe.up.pt/~robosoc/sci/PhD_Armando_Sousa_lo_qual.pdf

Multi-Hypothesis Tracking

- **Extension to basic EKF**
- **Belief is represented by multiple Gaussians**



Gaussian Localizations

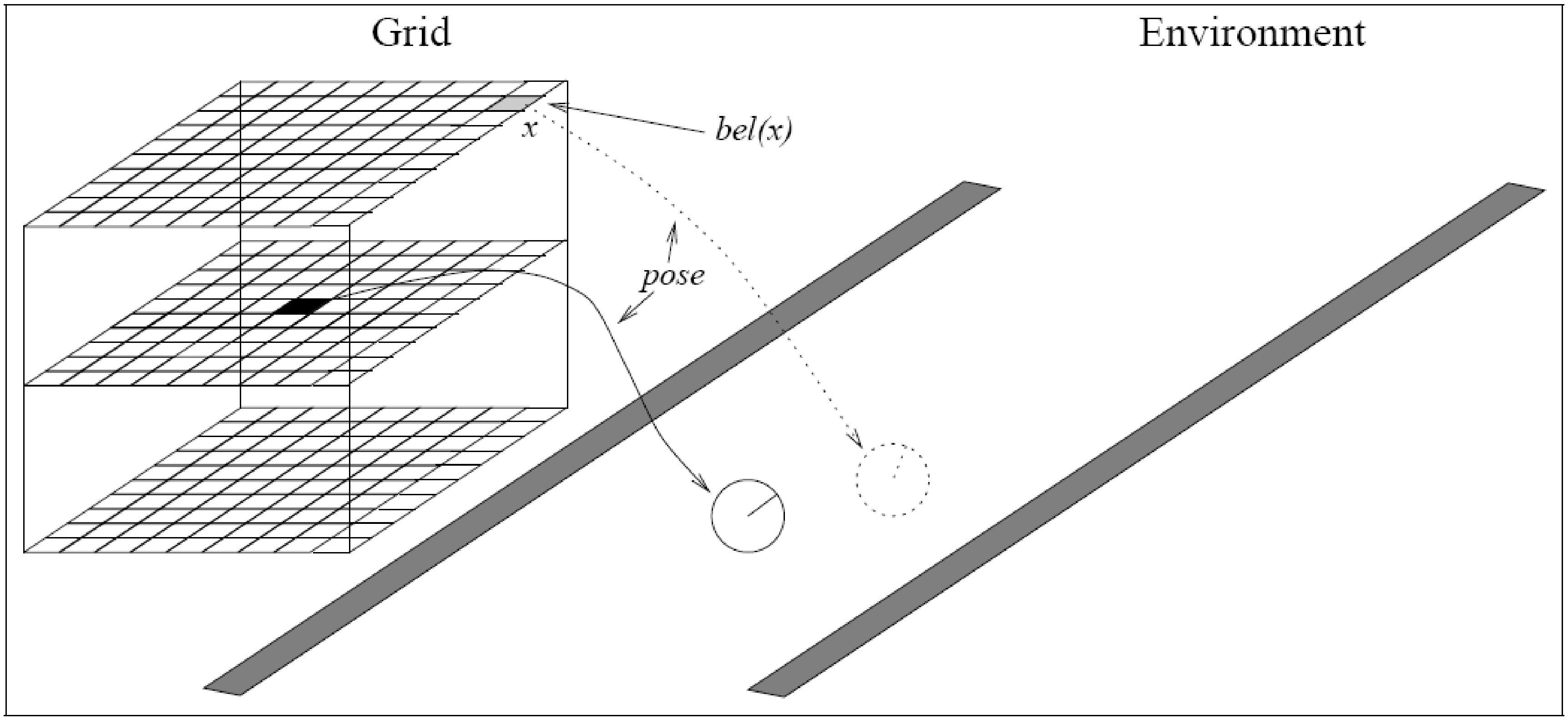
- **Unimodal Gaussian is a good uncertainty representation for tracking**
  - It is not for global localization
- **Not good for Hard Spatial Constraints**
  - “Close to wall, but not inside wall”
- **Linearization error increases with size of neighborhood**
- **Features must be sufficient and distinguishable**
- **Unable to process negative information**

Grid Localization

- **Can solve the global localization problem**
- **Not bound to unimodal distributions**
- **Can process raw sensor measurements**

- **Uses a histogram filter to represent posterior belief**
- **Choice of resolution is critical**
  - High resolution -> Slow
  - Low resolution -> information loss

Grid Localization



Grid Localization

- **Algorithm**

Algorithm Grid_localization($\{p_{k,t-1}\}, u_t, z_t, m$)**:**

for all k *do*

$$\bar{p}_{k,t} = \sum_i p_{i,t-1}\ \textbf{motion\_model}(\text{mean}(\mathbf{x}_k), u_t, \text{mean}(\mathbf{x}_i))$$

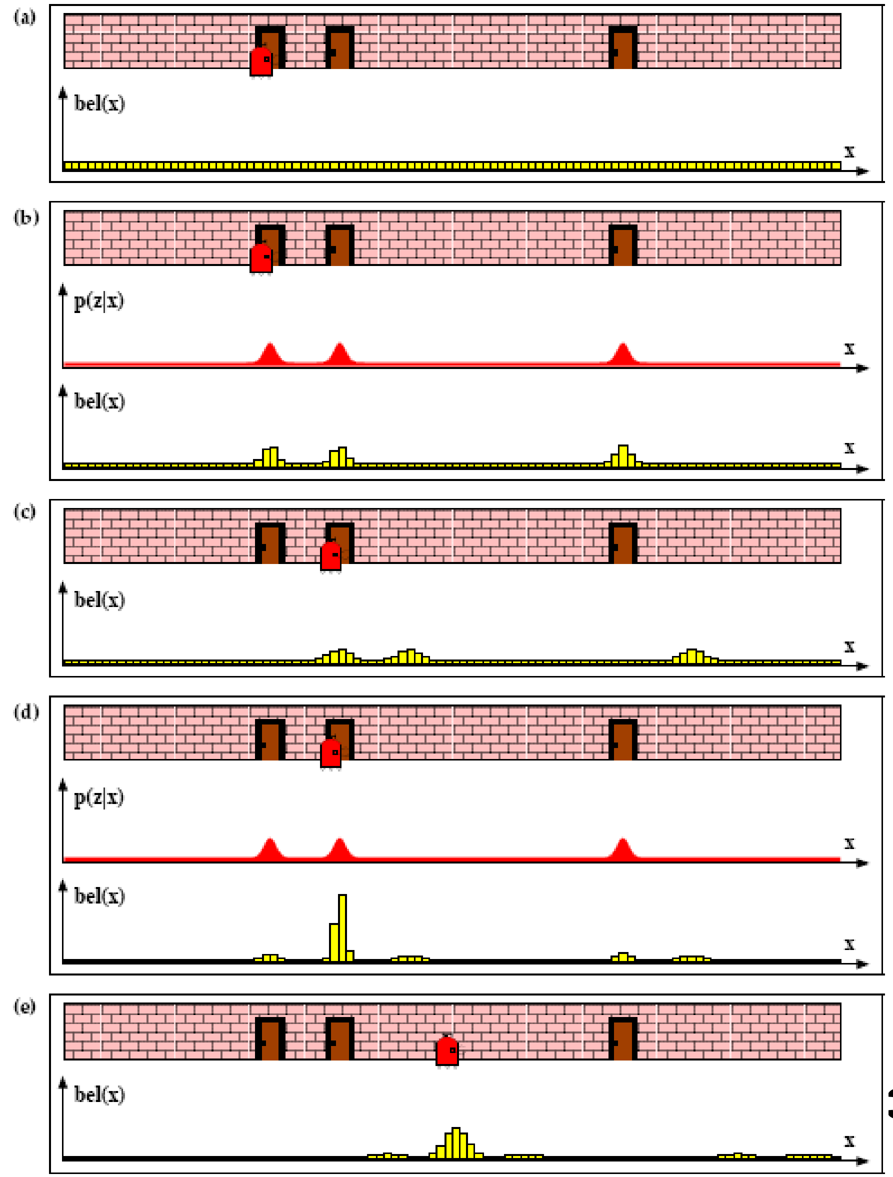
$$p_{k,t} = \eta\ \bar{p}_{k,t}\ \textbf{measurement\_model}(z_t, \text{mean}(\mathbf{x}_k), m)$$

endfor

return $\{p_{k,t}\}$

Grid Localization

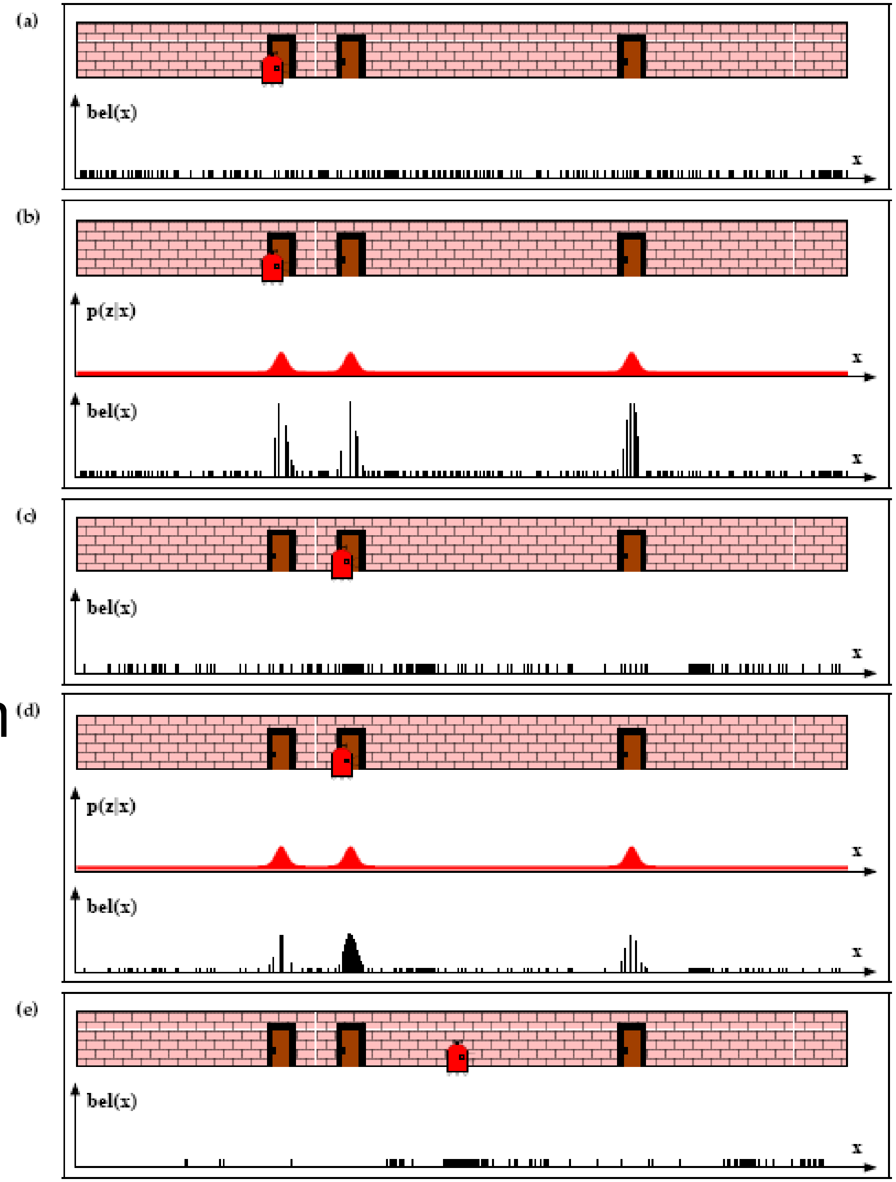
a) Belief is uniform
b) First integration of sensor data, result is multimodal
c) Convolution with motion model, shifts and flattens belief
d) Second integration of sensor data, robot localizes itself
e) Moving along



Monte Carlo Localization

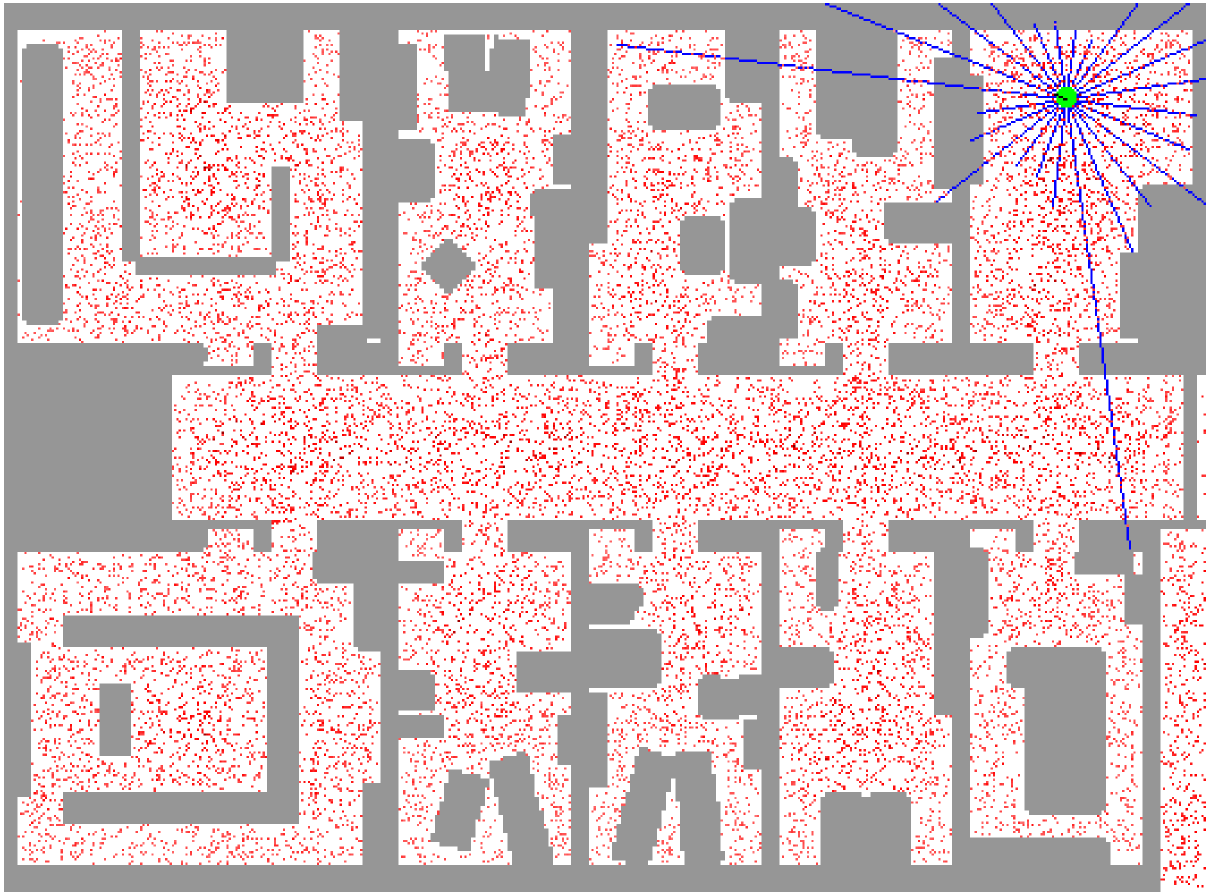
Algorithm MCL(X_{t-1}, u_t, z_t, m)**:**
$\overline{X}_t = X_t = \emptyset$
for $m = 1$ to M **do**
  $x_t^{[m]}$ =**sample_motion_model**$(u_t, x_{t-1}^{[m]})$
  $w_t^{[m]}$ =**sample_measurement_model**$(z_t, x_t^{[m]}, m)$
  $\overline{X}_t = \overline{X}_t + \left\langle x_t^{[m]}, w_t^{[m]} \right\rangle$
end for
for $m = 1$ to M **do**
  draw i with probability $\alpha\ w_t^{[i]}$
  add $x_t^{[i]}$ to X_t
end for
return X_t

Monte Carlo Localization



a) Pose particles drawn at random and uniformly
b) Importance factor assigned to each particle, set of particles hasn't changed
c) After resampling and incorporating robot motion
d) New measurement assigns new importance factors
e) New resampling and motion

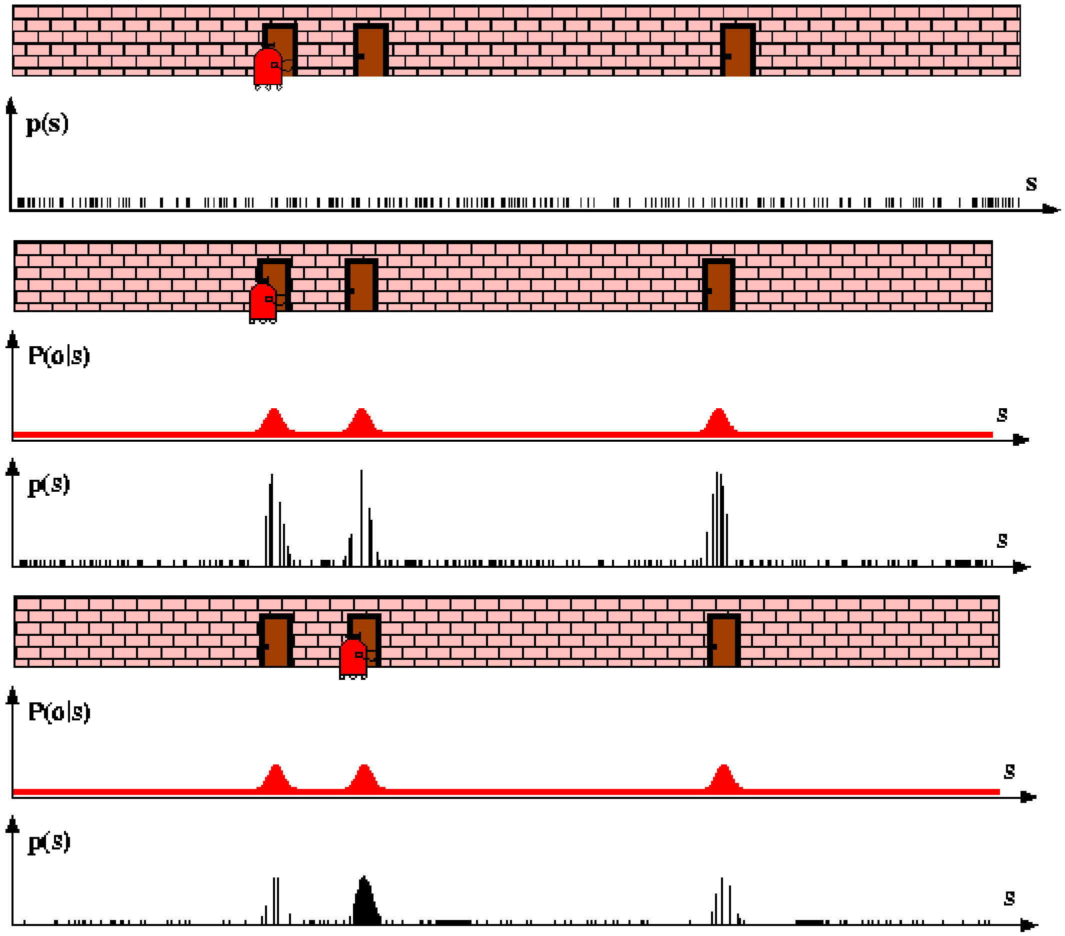
Monte Carlo Localization

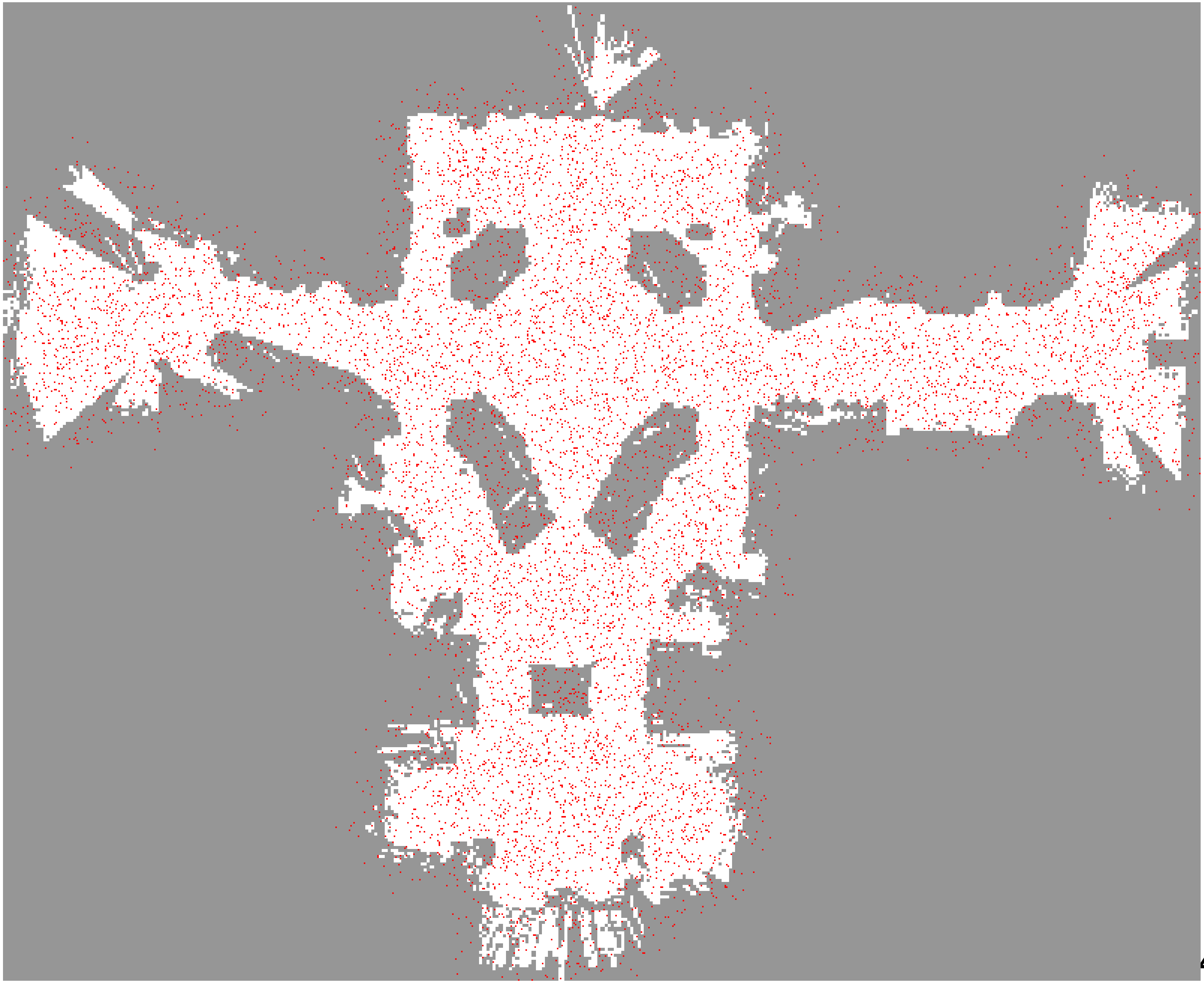


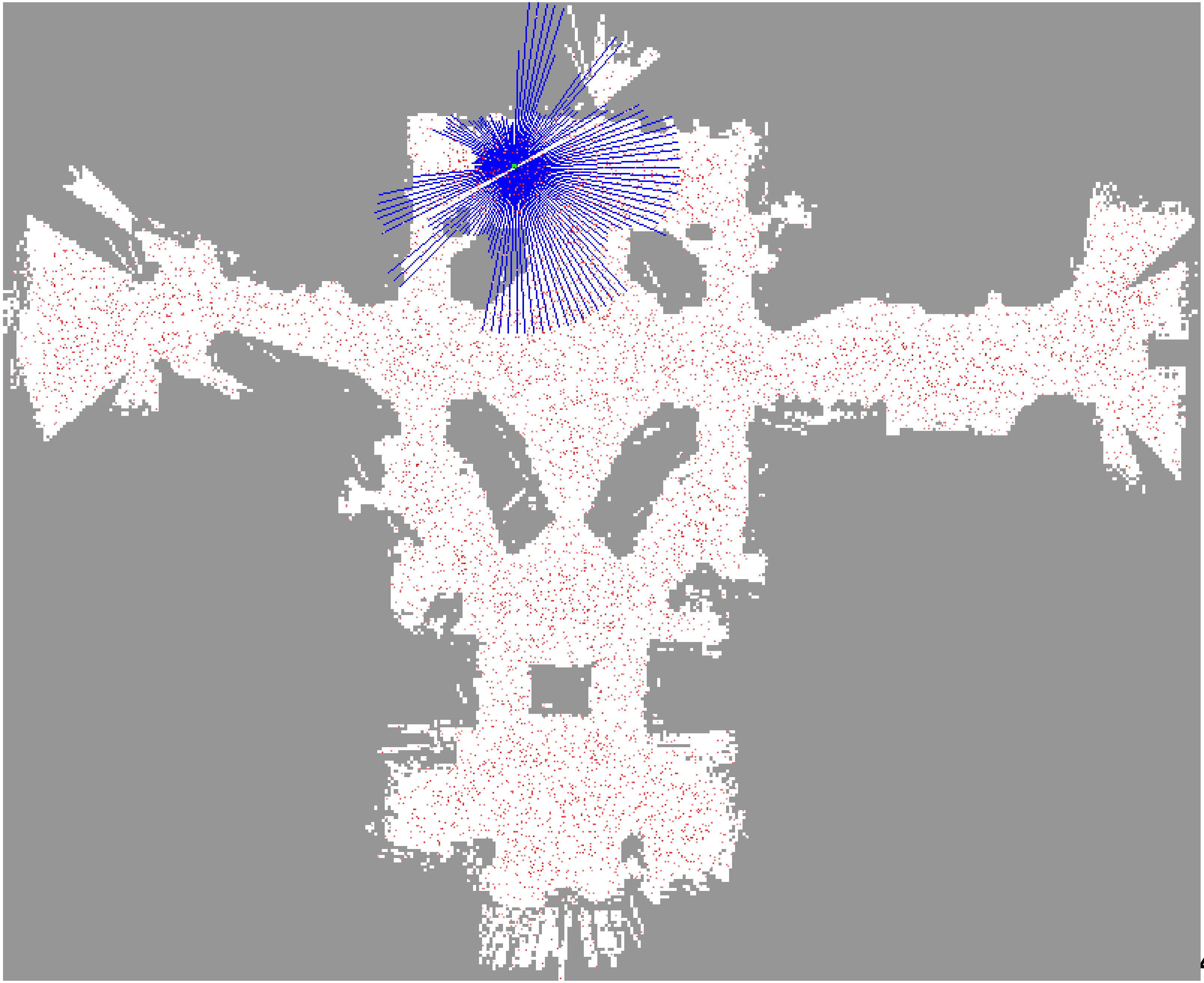
particle_filter (S_{t-1}, u_{t-1} z_t)

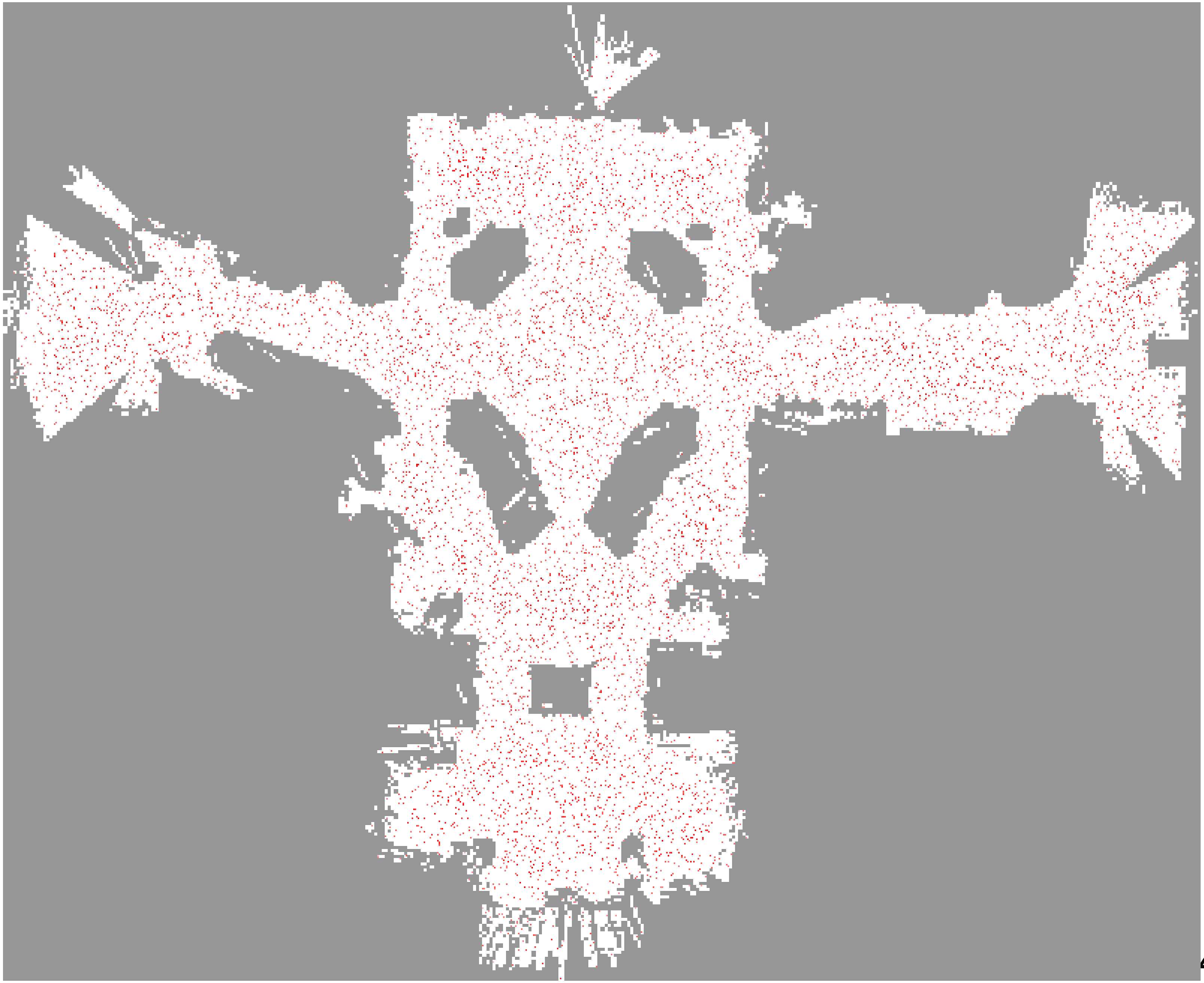
1. $S_t = \emptyset, \quad \eta = 0$
2. **For** $i = 1 \ldots n$ ***Generate new samples***
3. Sample index *j(i)* from the discrete distribution given by w_{t-1}
4. Sample x_t^i from $p(x_t \mid x_{t-1}, u_{t-1})$ using $x_{t-1}^{j(i)}$ and u_{t-1}
5. $w_t^i = p(z_t \mid x_t^i)$ ***Compute importance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

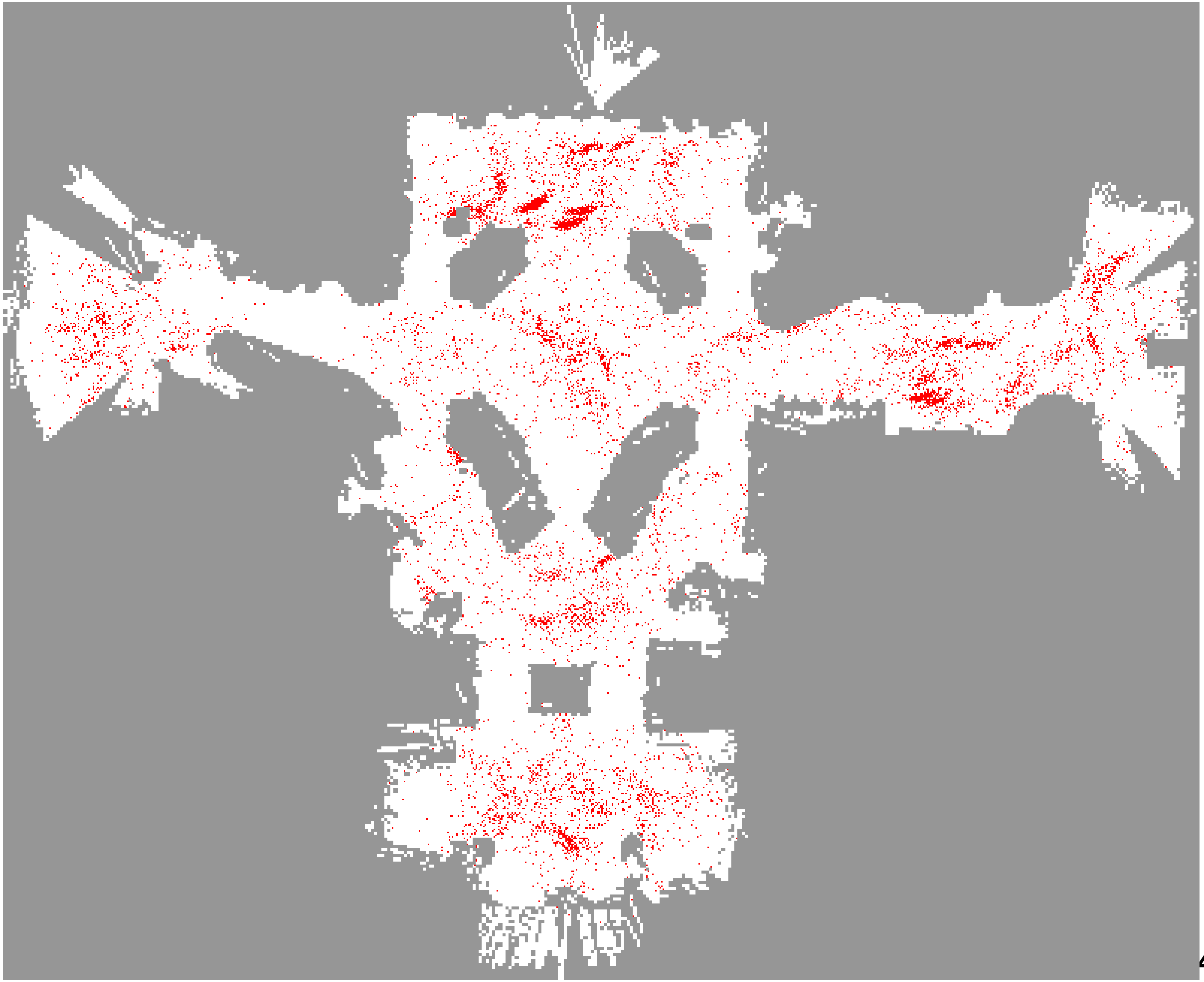
Particle Filters

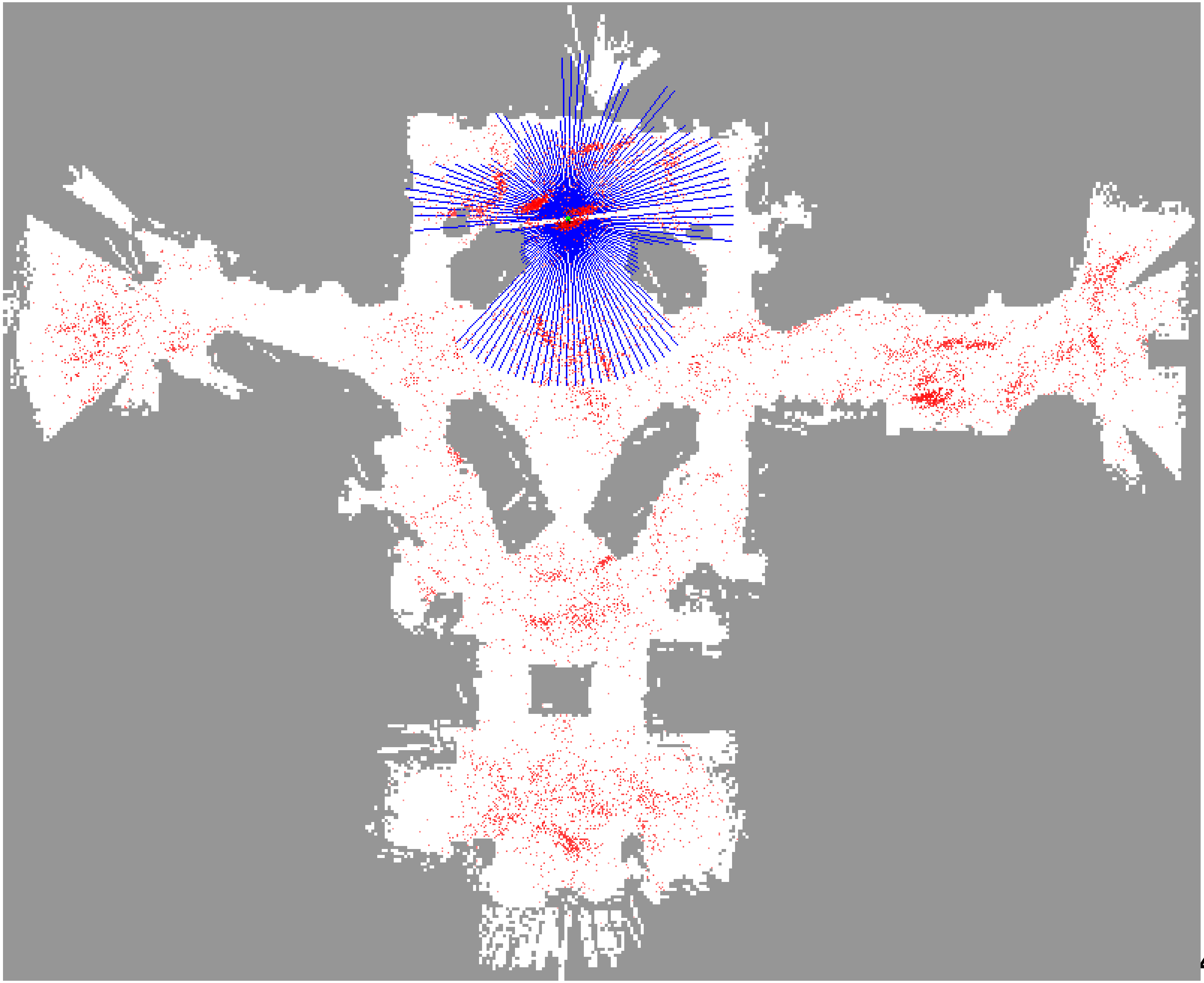


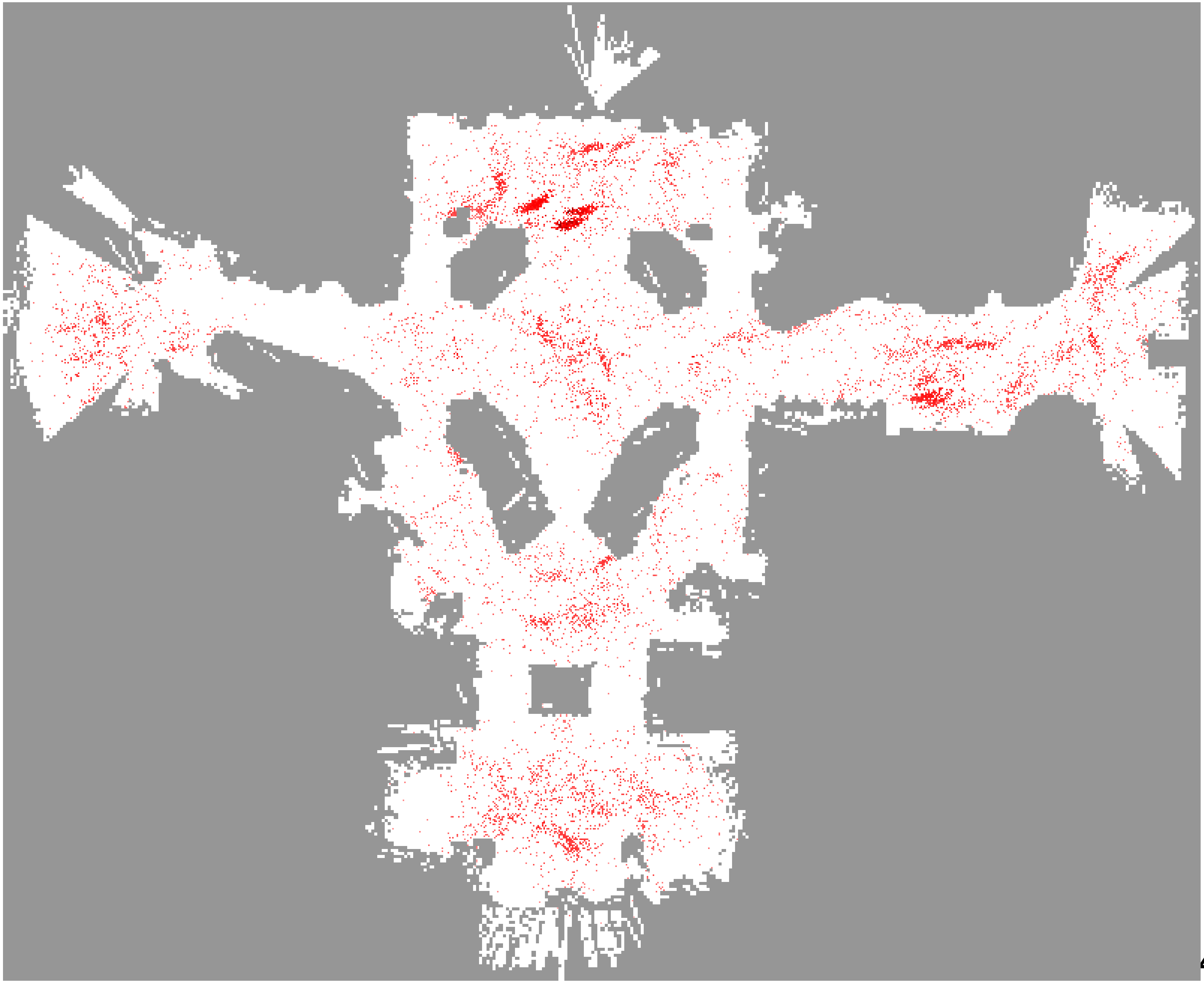


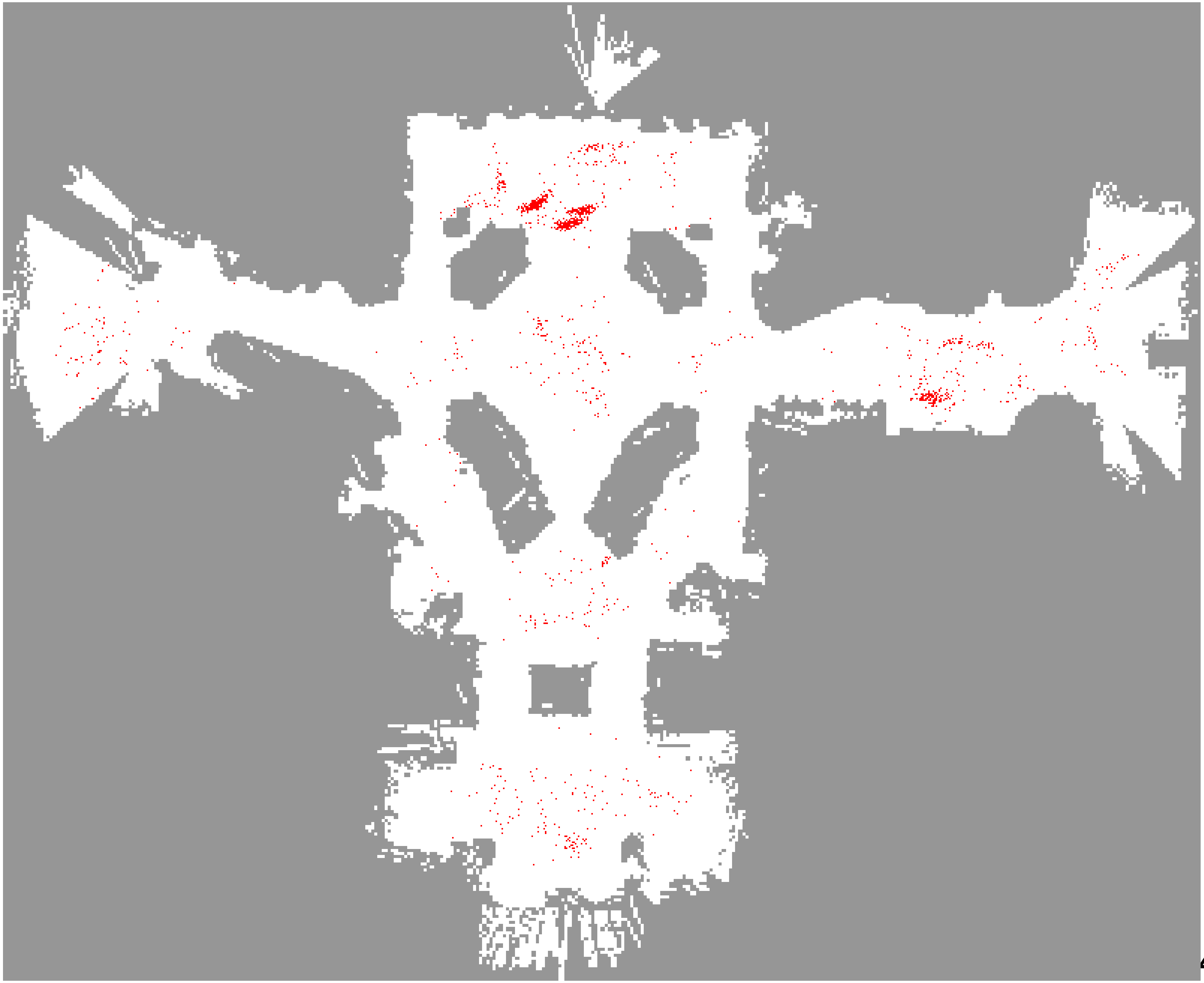




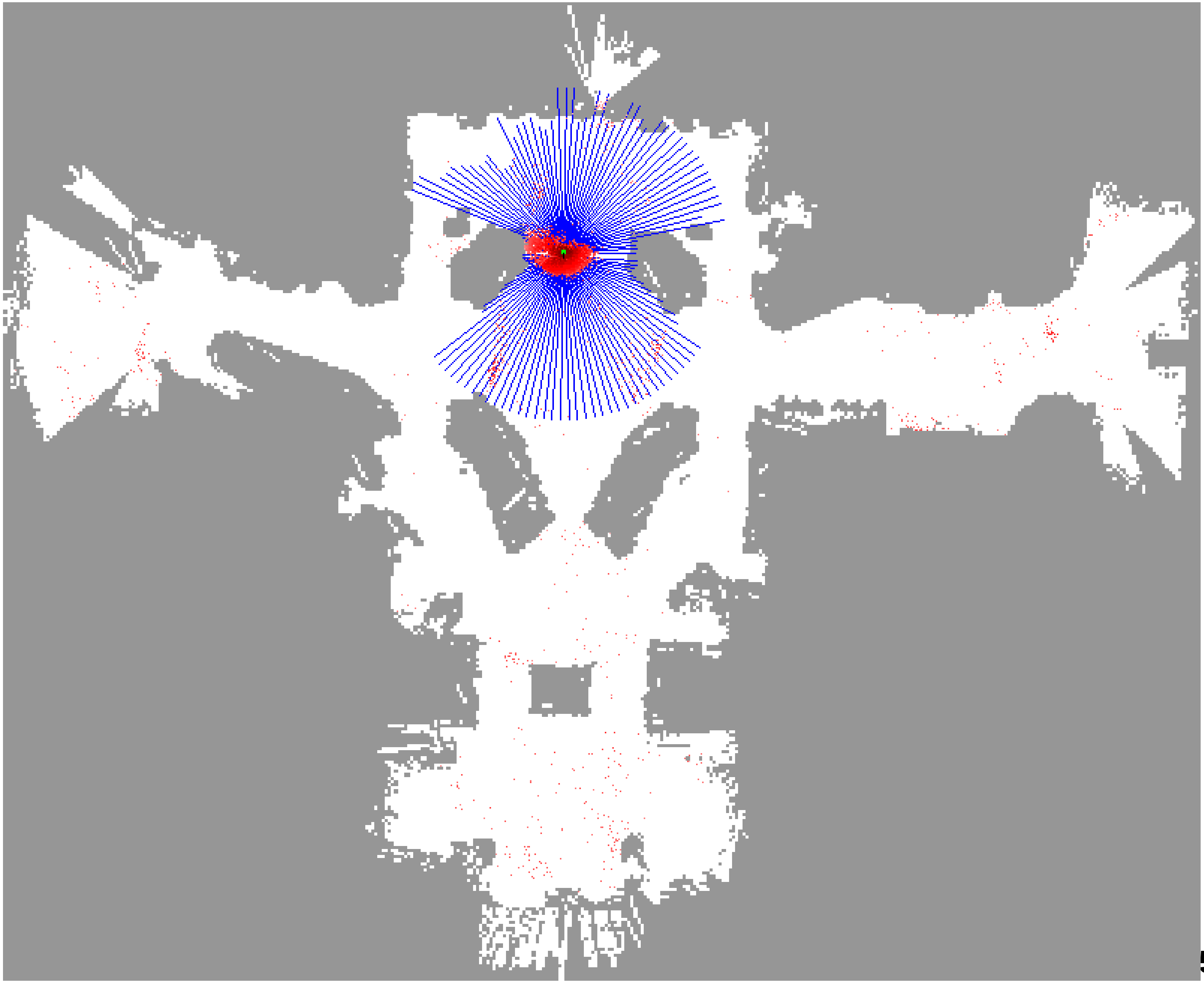


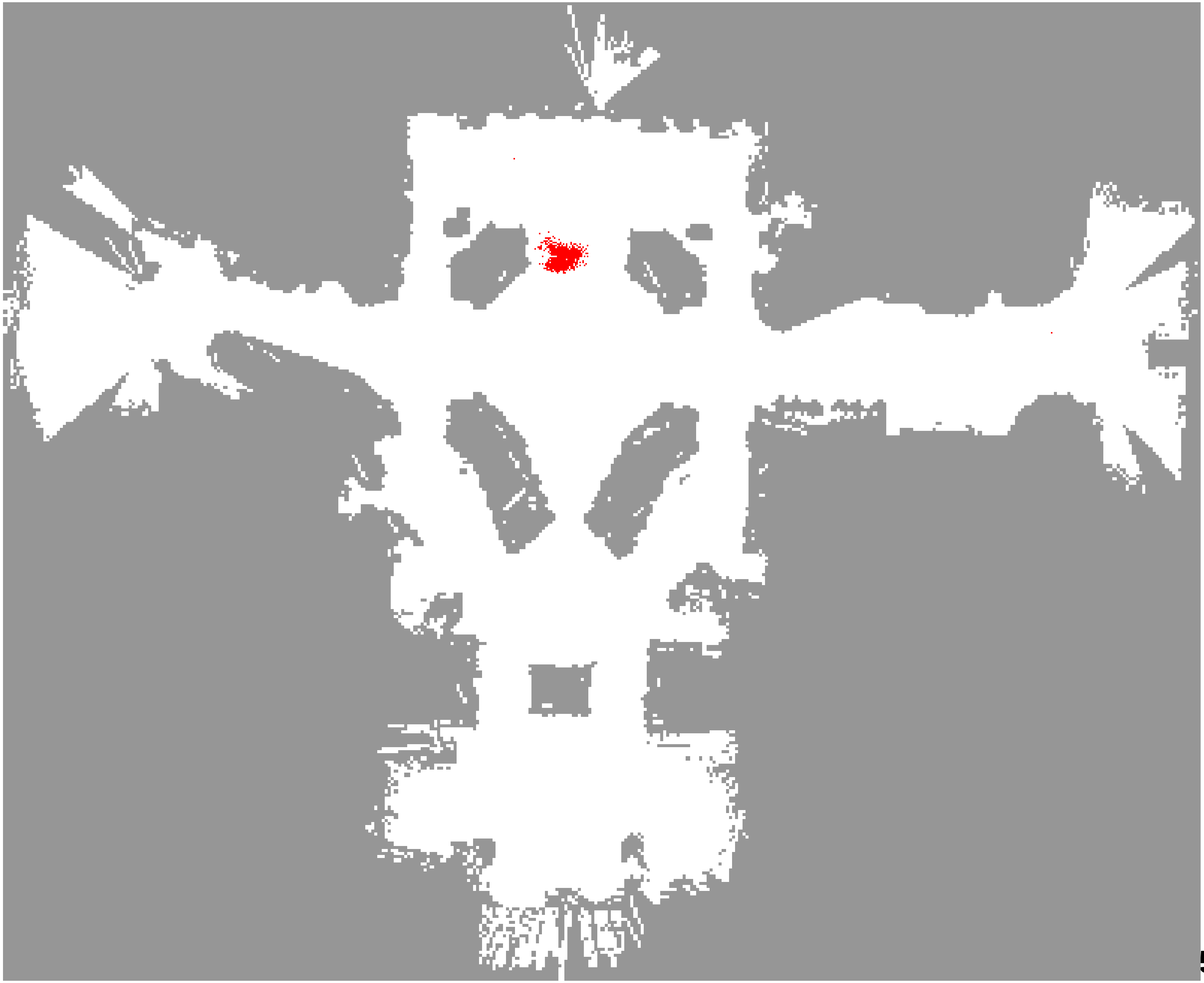


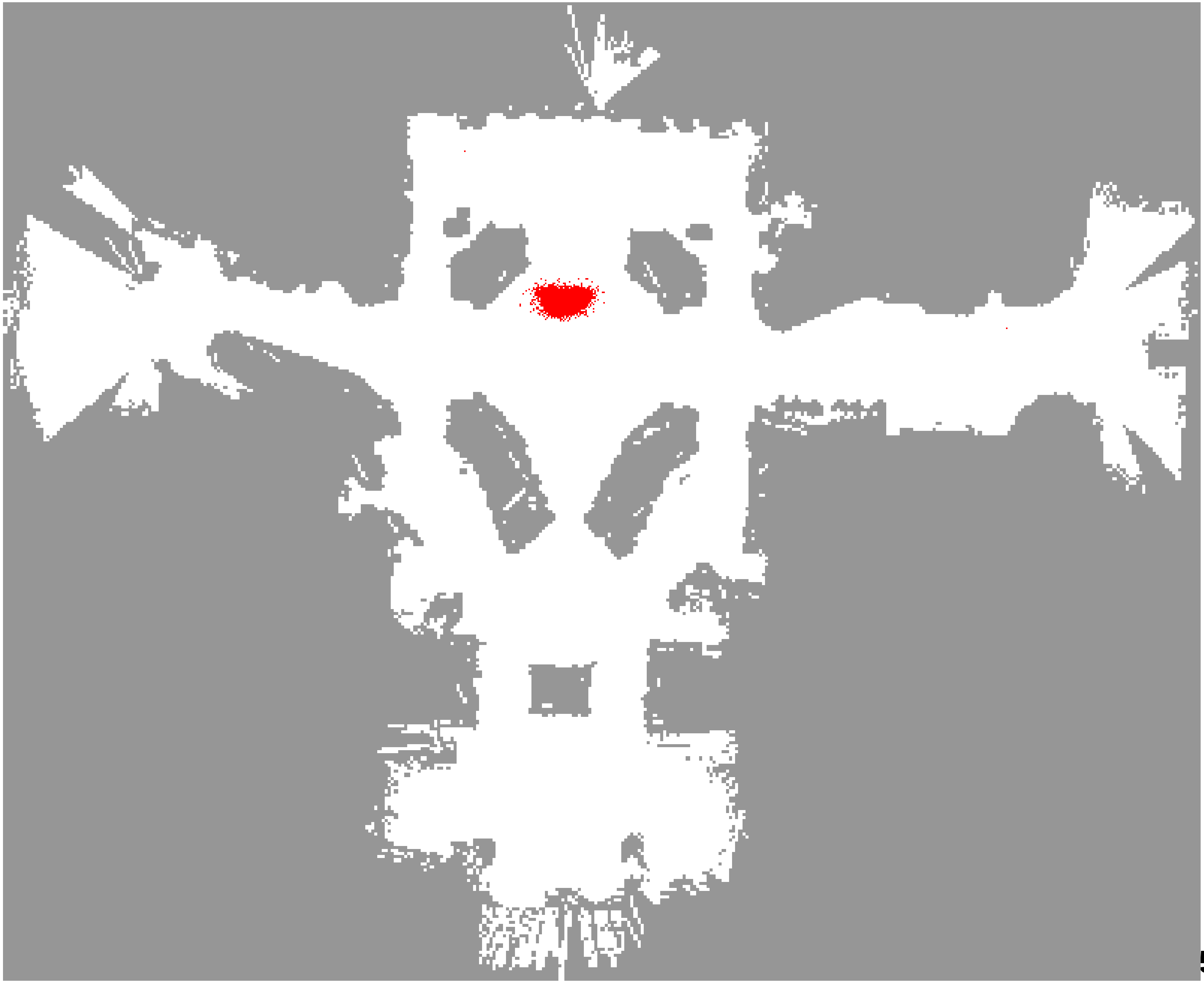


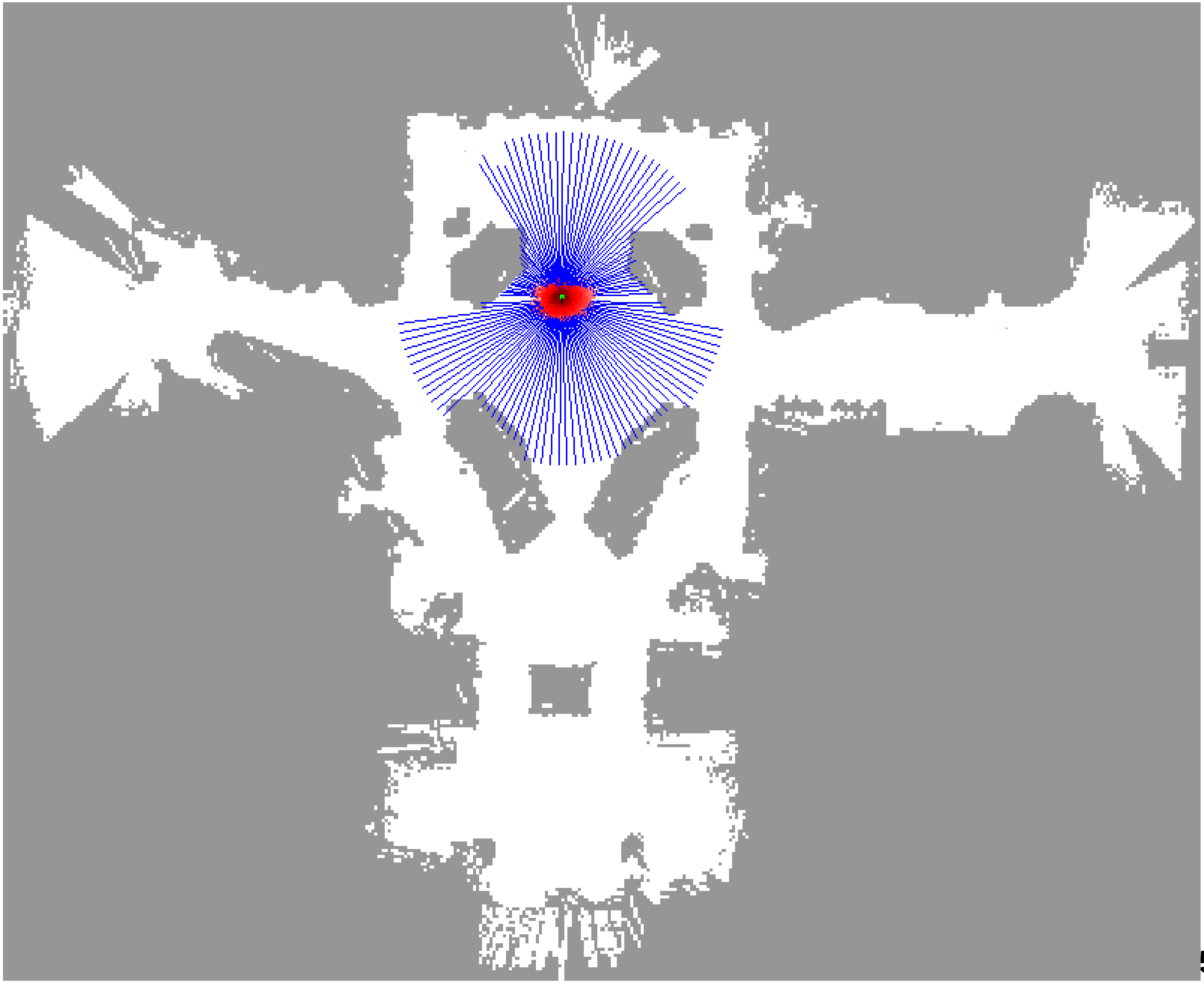


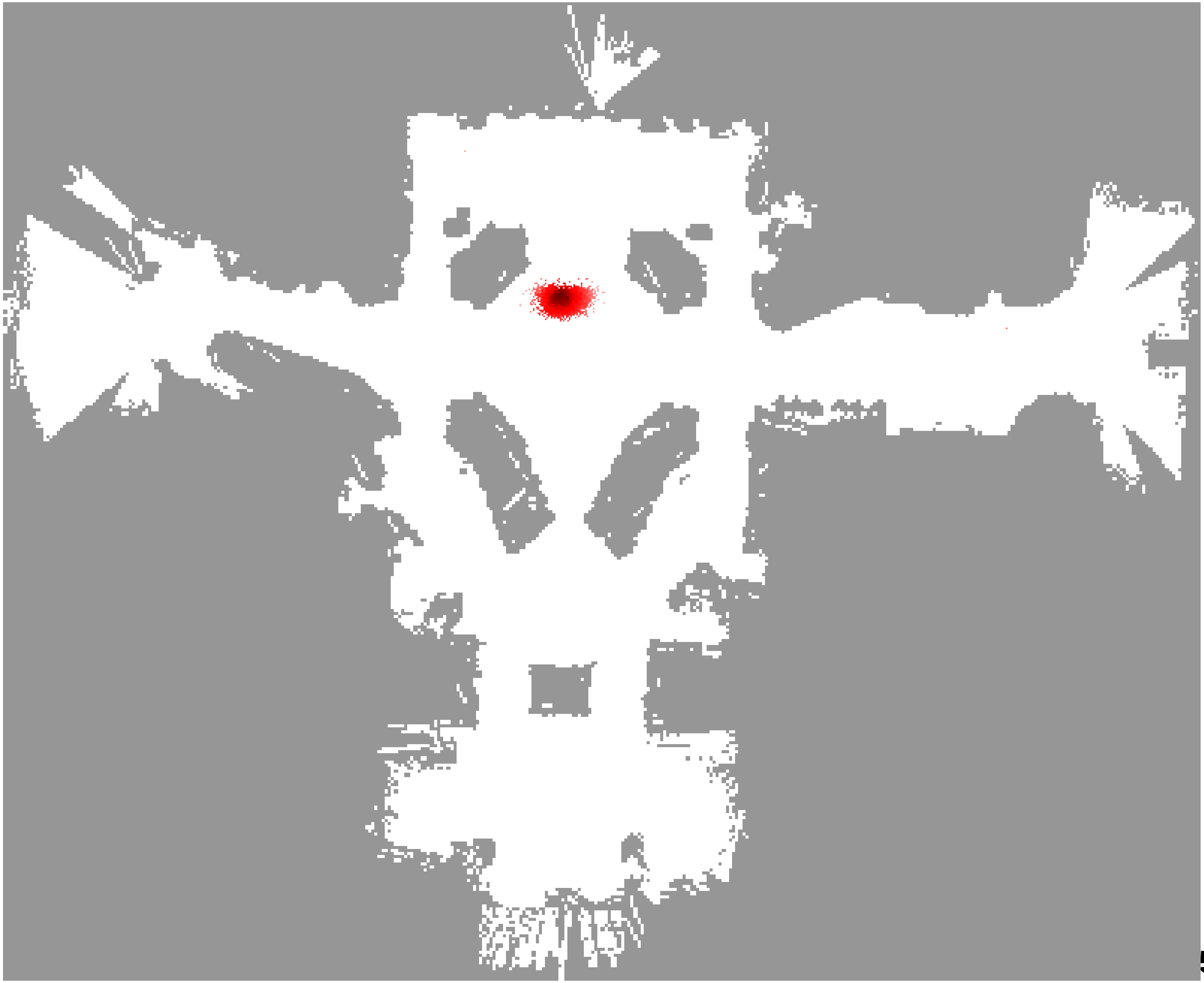
50

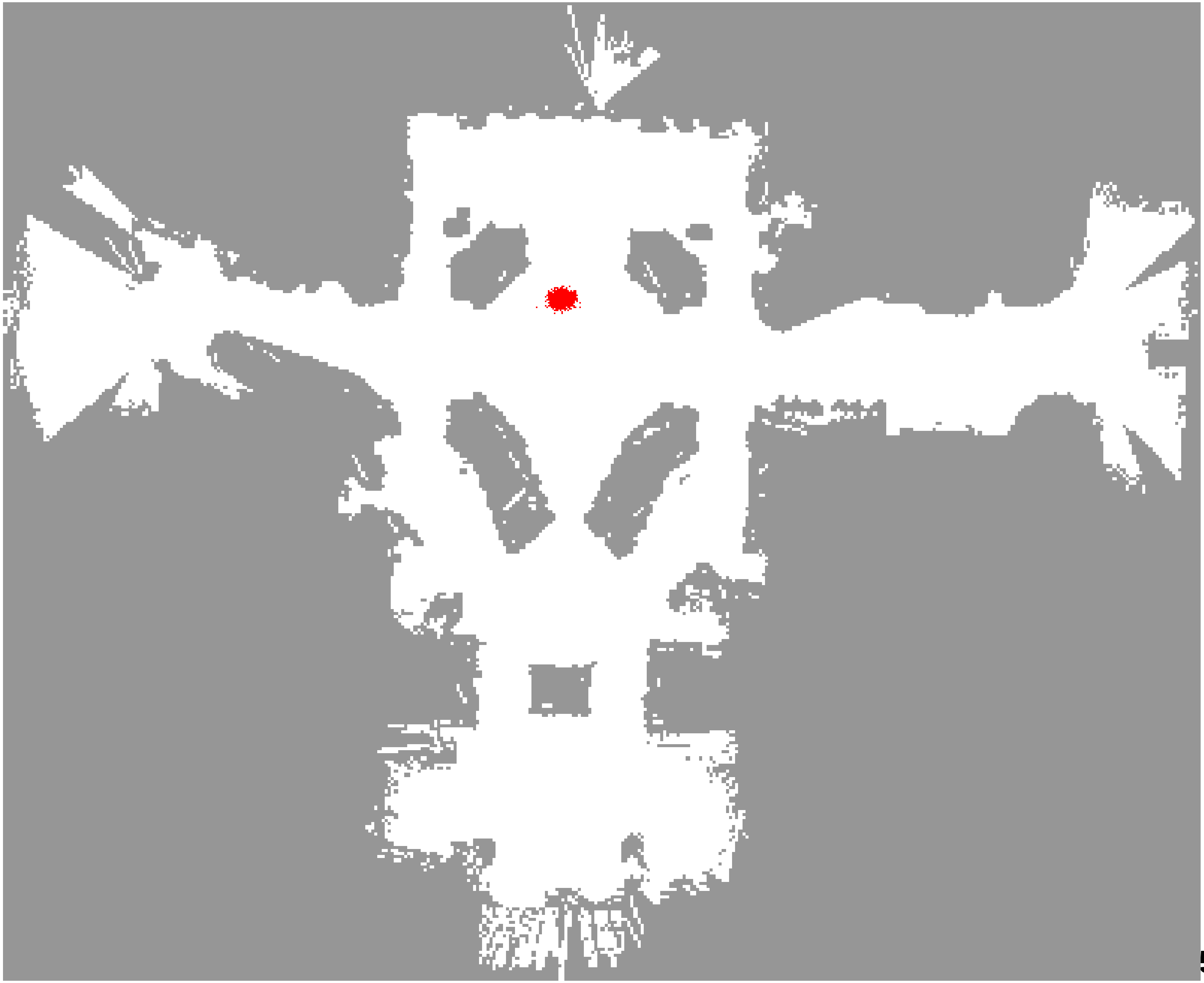


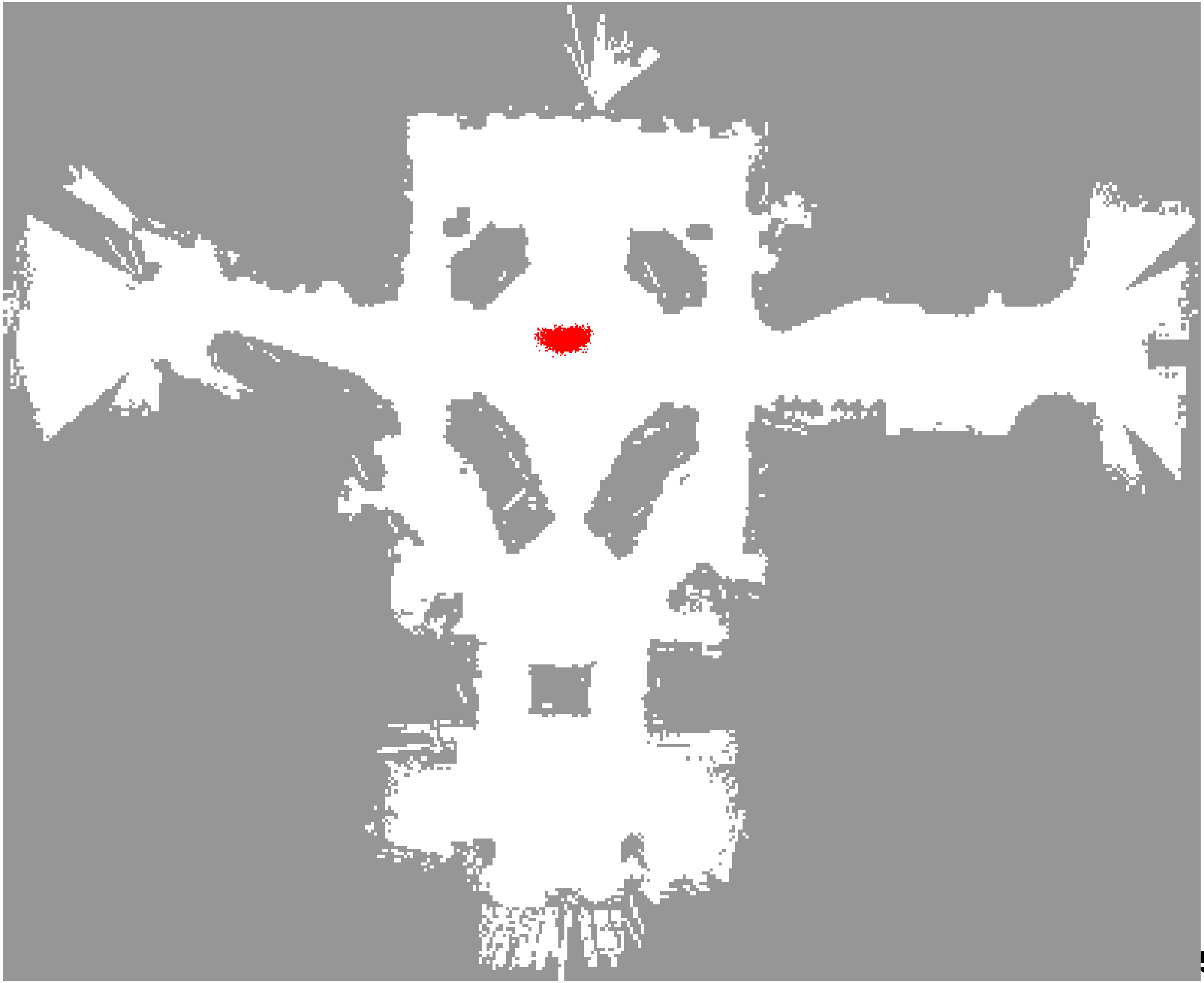


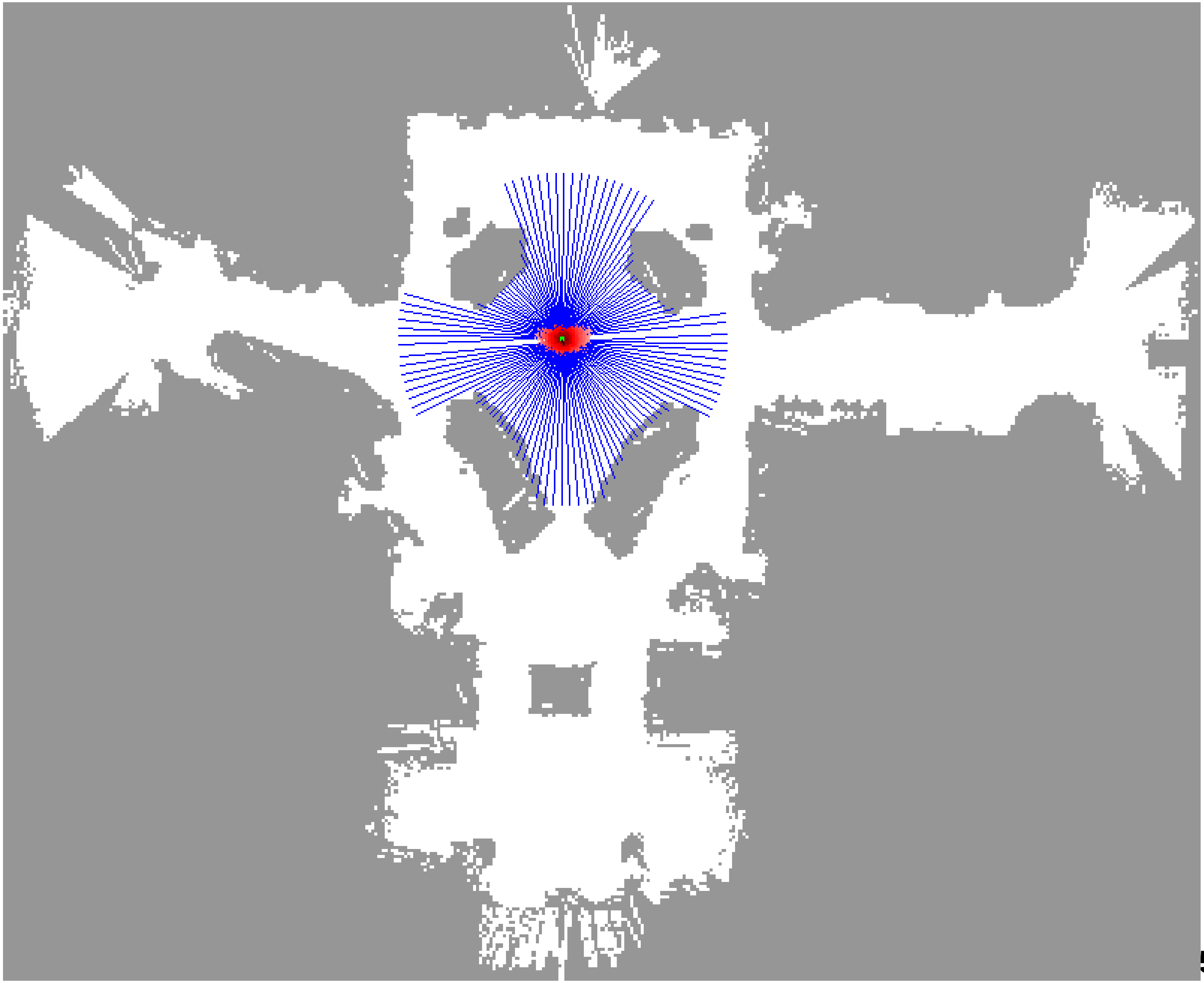


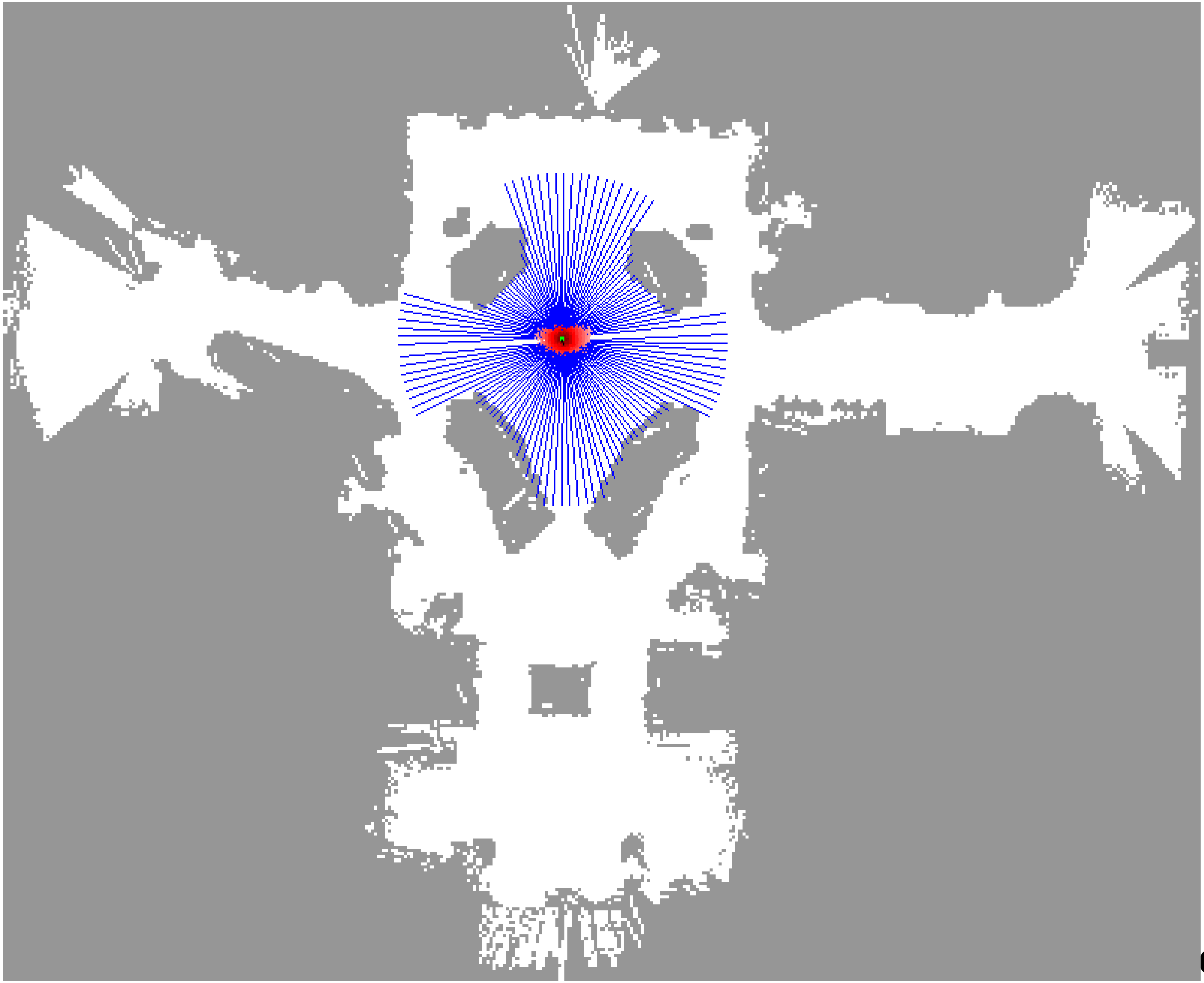






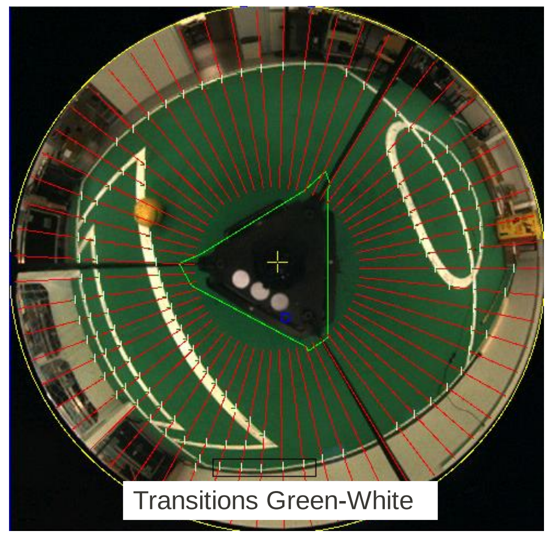


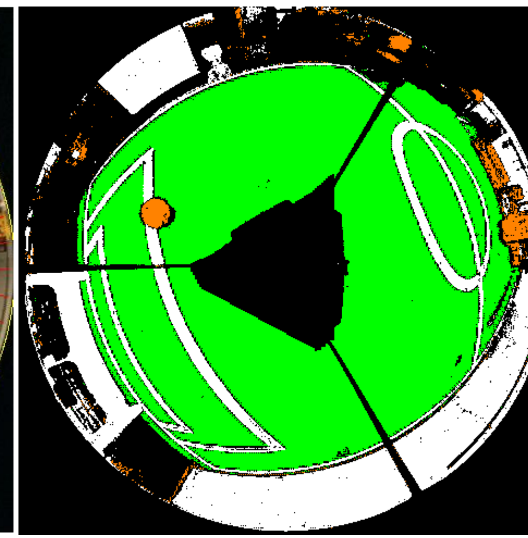




Localization
–
Tribots and Cambada
RoboCup MSL

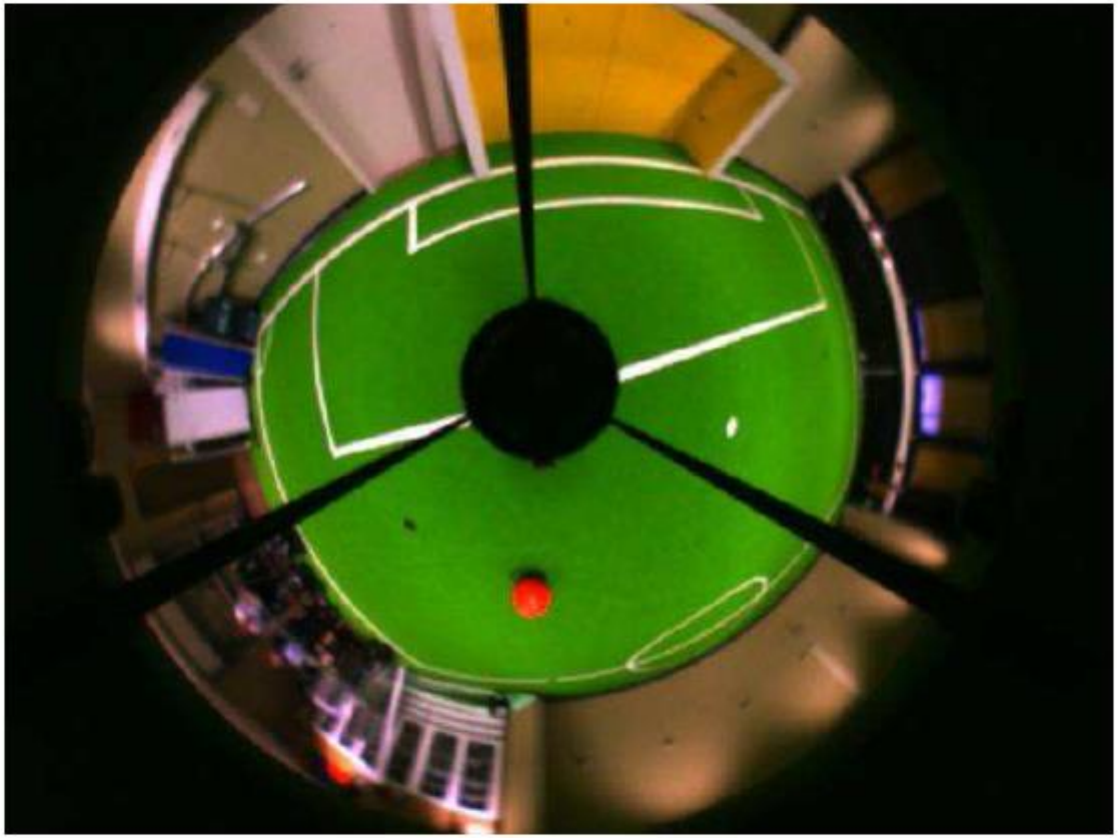
Localization in MSL





5dpo-2000 robots

Localization in Cambada



Tribots Localization

Optimization based

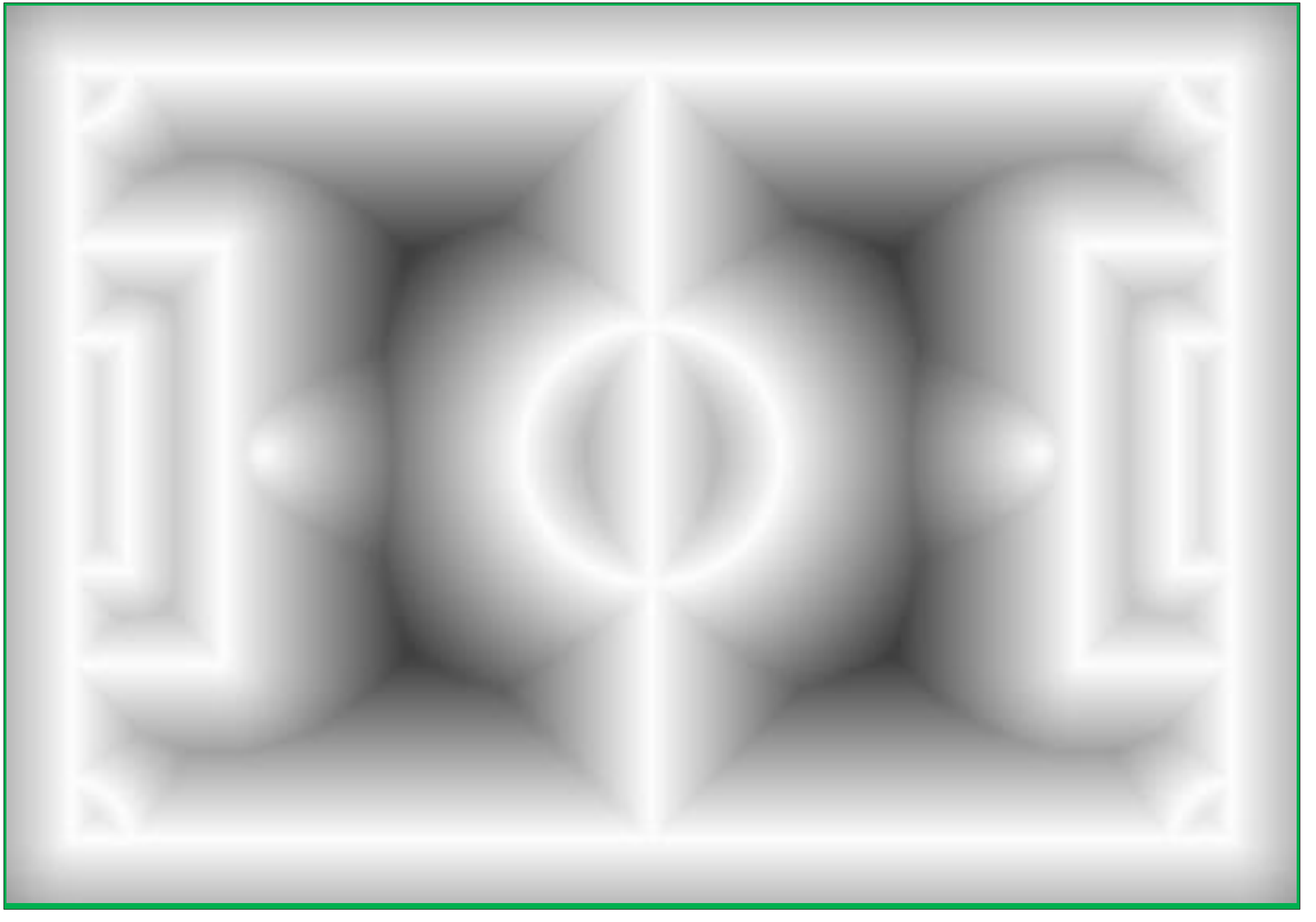
Uses white lines seen by the robot

Algorithm:

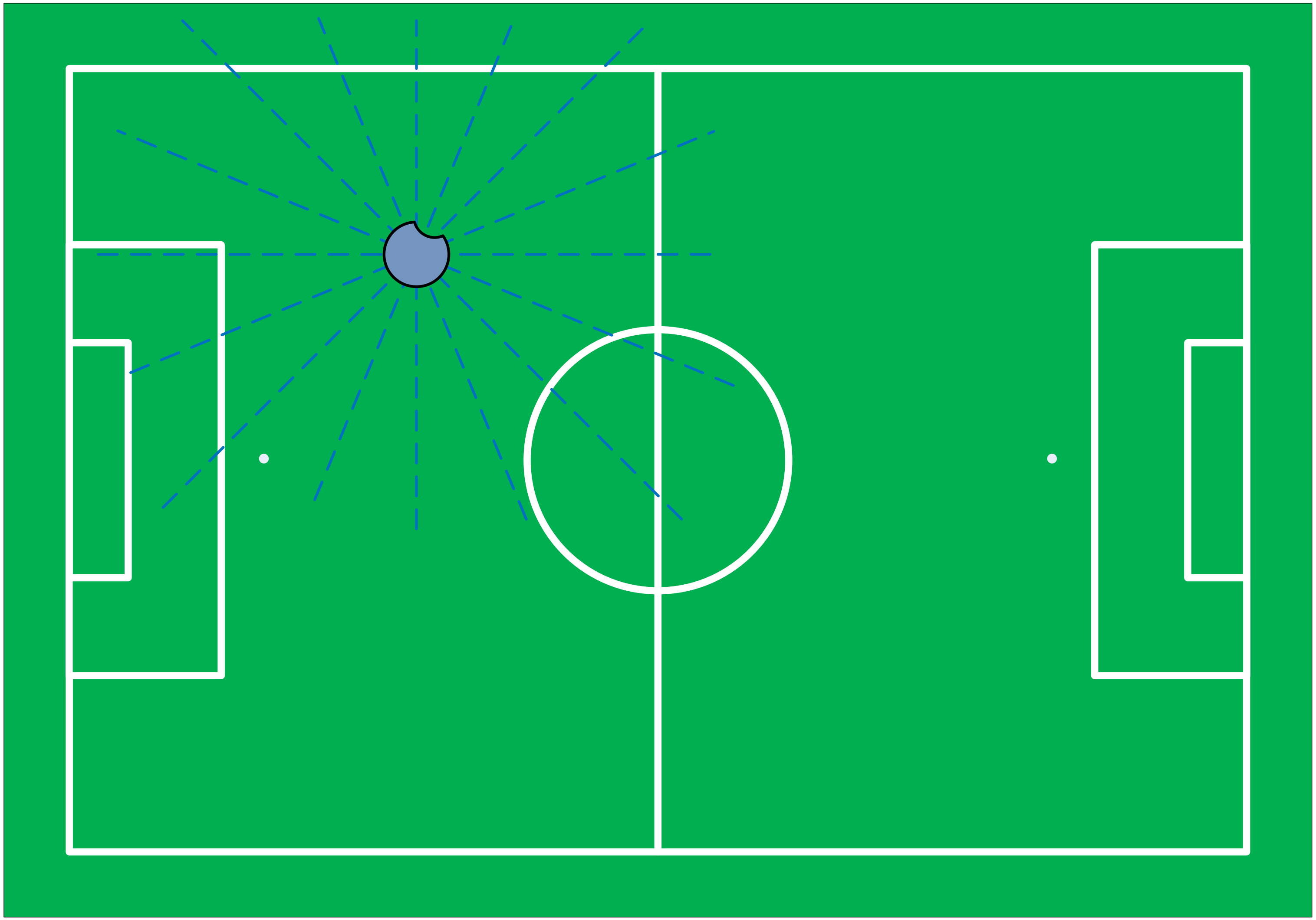
```
fieldLUT = build_fieldLUT()
For each vision frame
do
   whiteLines = getVisionLines()
   odometry = getOdometry()

   calculateDistanceWeights(whiteLines)
   trialPos = updateWithOdometry(curPos, odometry)
   optPos = optimize(trialPos, whiteLines)
   variances = analyse(optPos, whiteLines)
   pos = simpleKalman(curPos, odometry, optPos, variances)
endo
```

Lauer et al., Calculating the Perfect Match: an Efficient and Accurate Approach for Robot Self-Localization, RoboCup Symposium, 2005

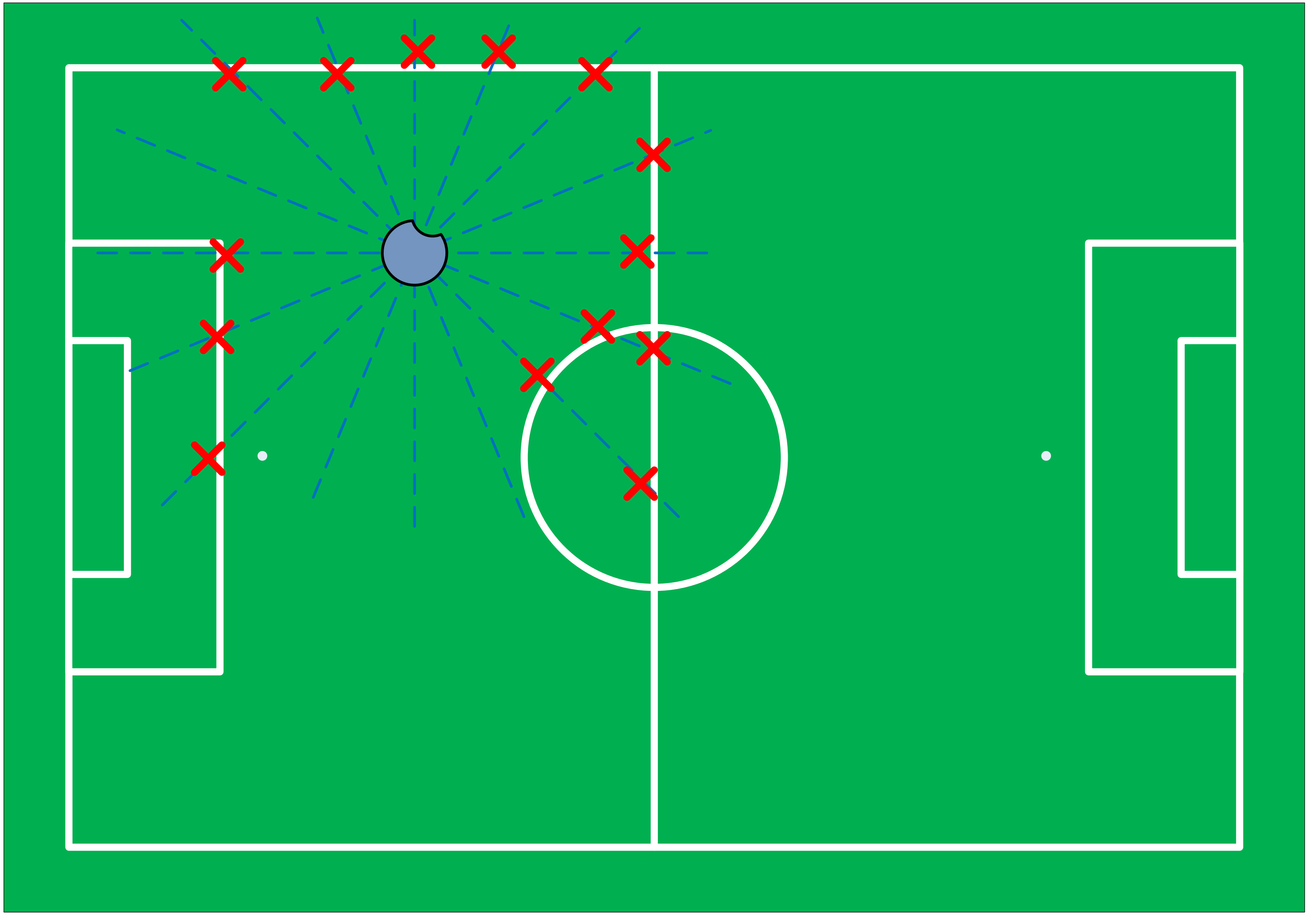
build_fieldLUT()



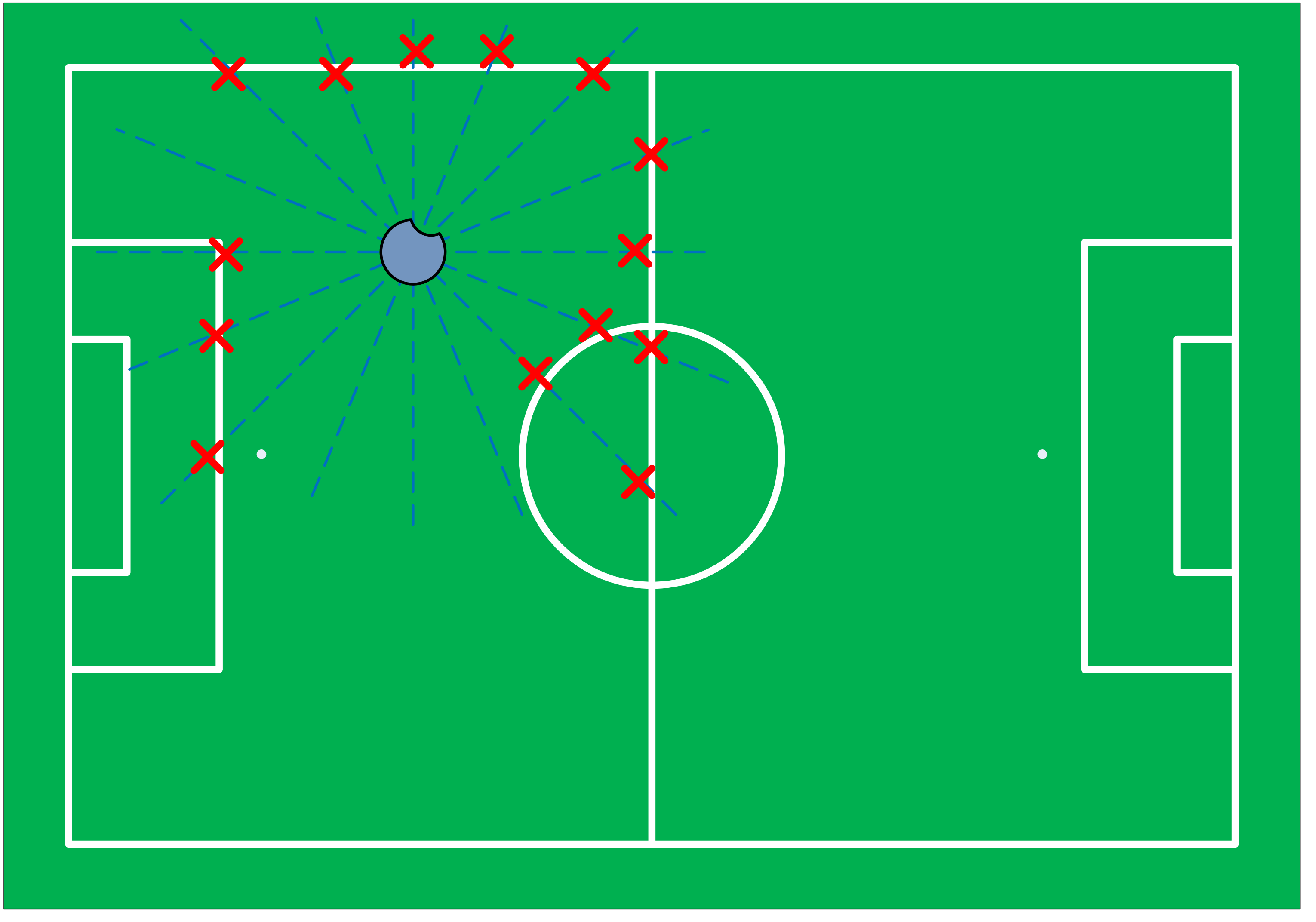
getVisionLines()



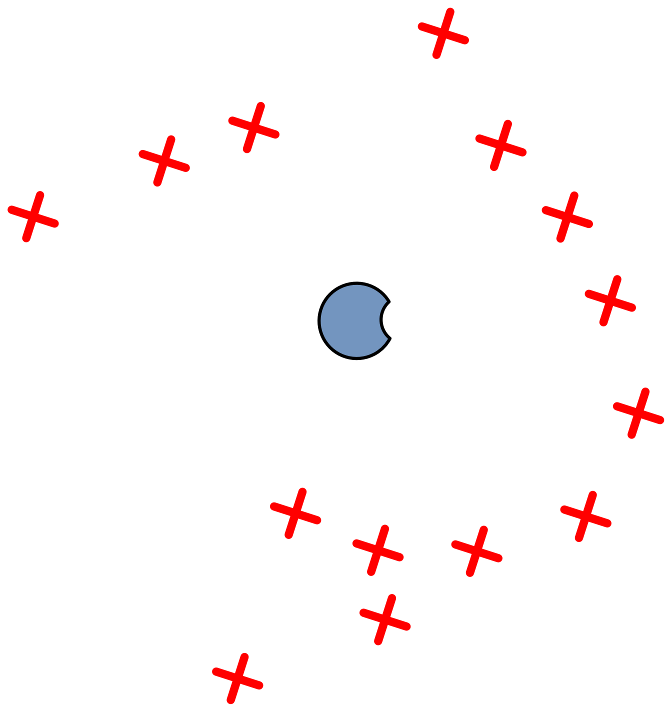
getVisionLines()



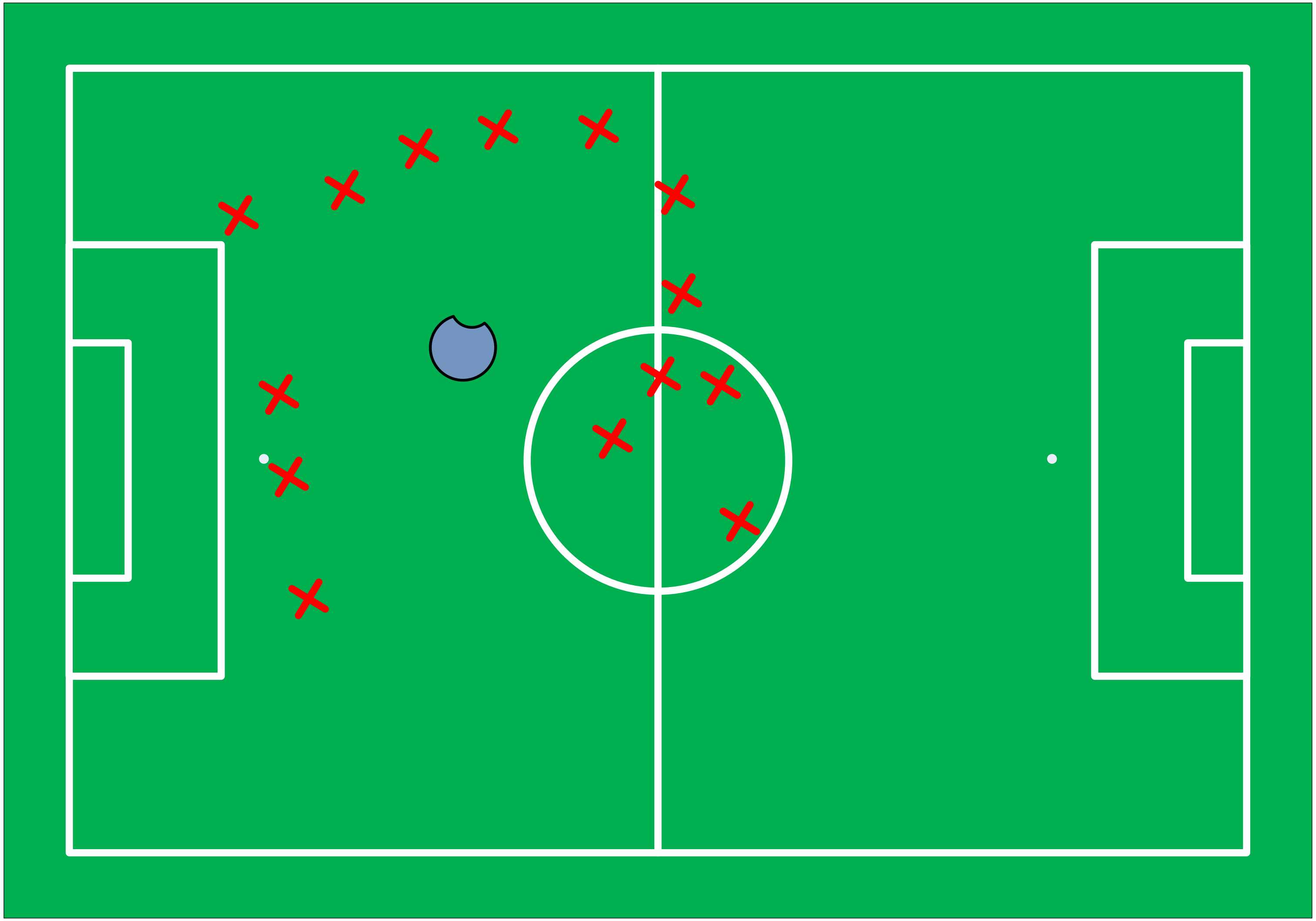
getVisionLines()



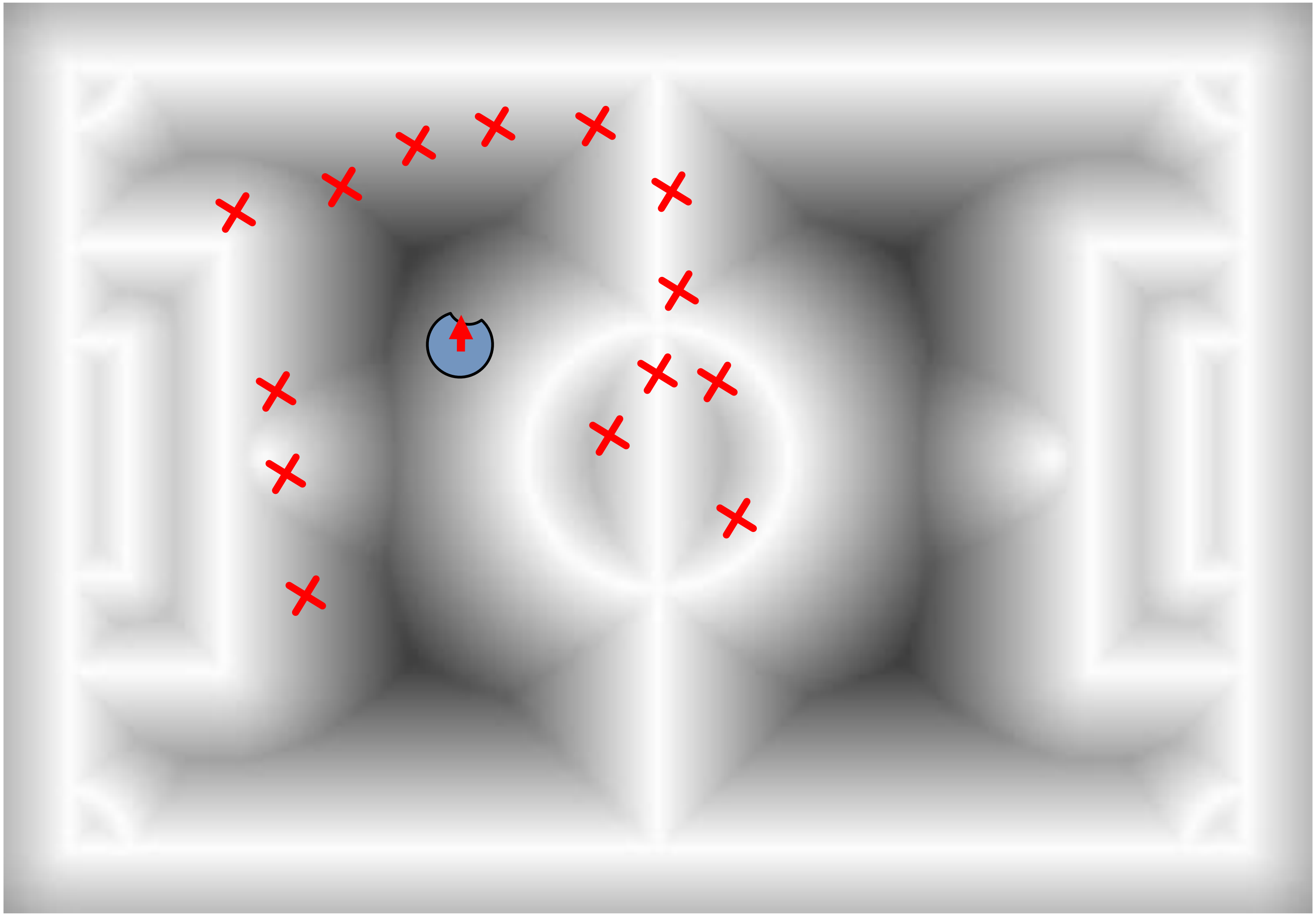
getVisionLines()



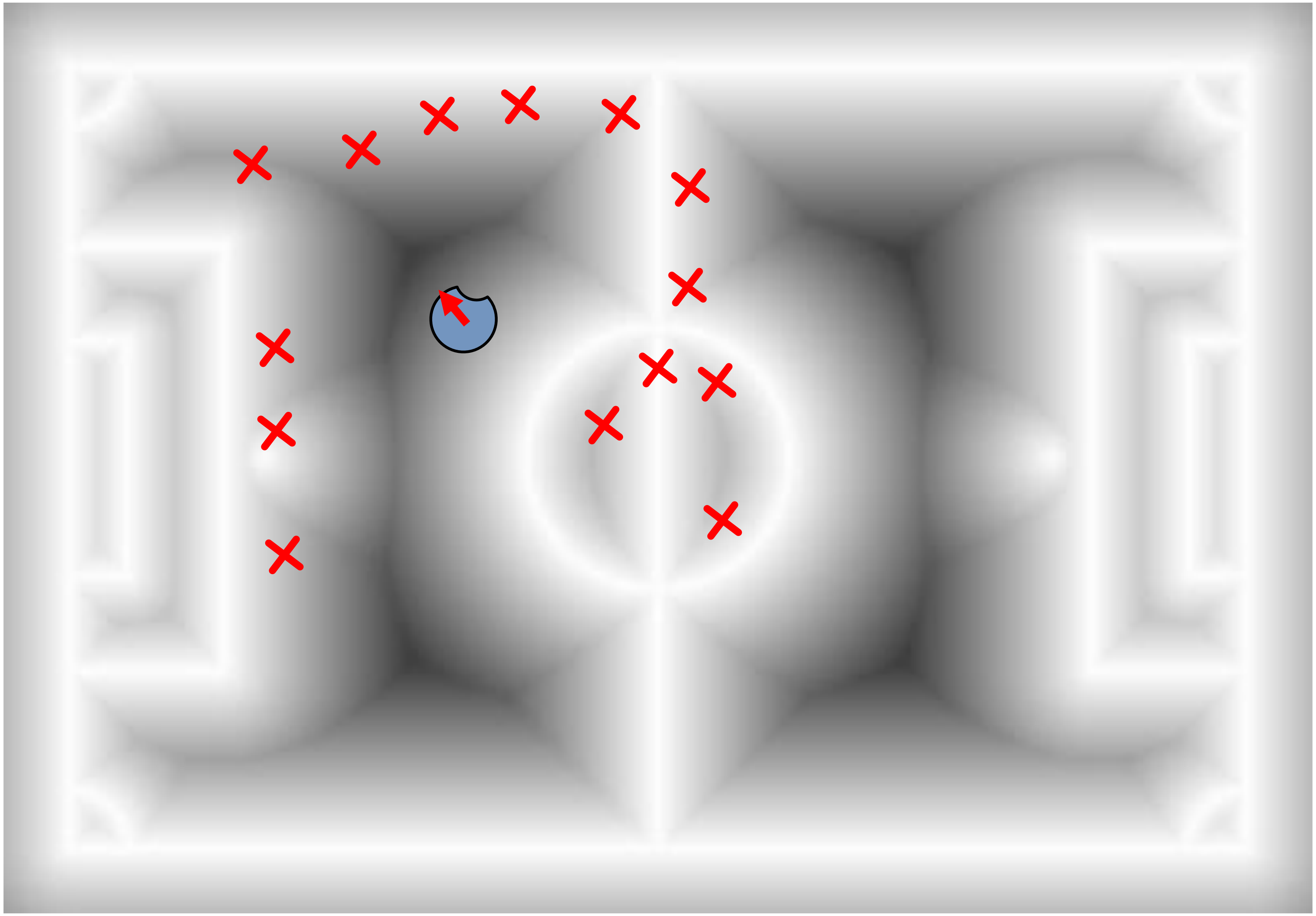
trialPos = updateWithOdometry(curPos, odometry)



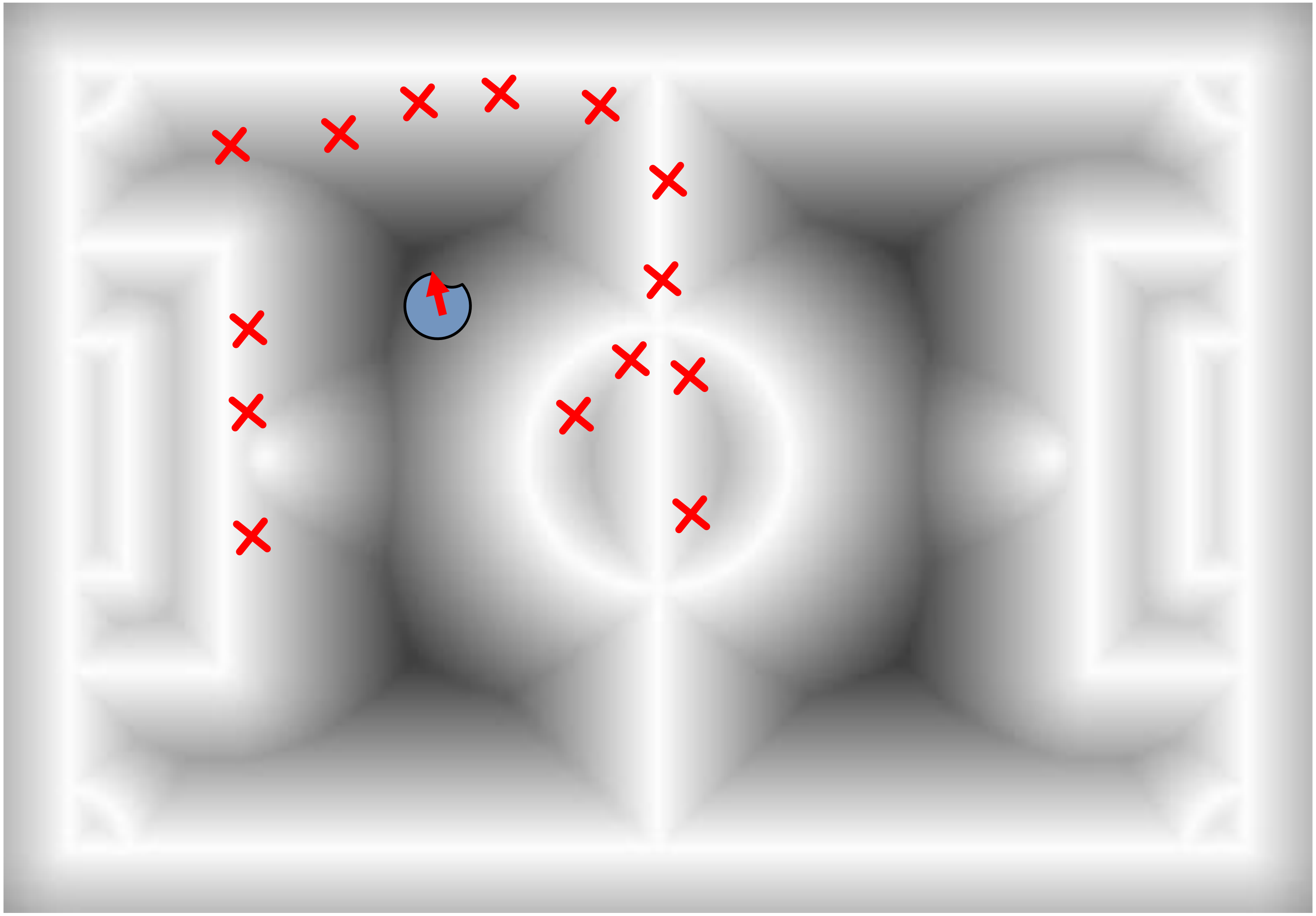
optPos = optimize(trialPos, whiteLines)



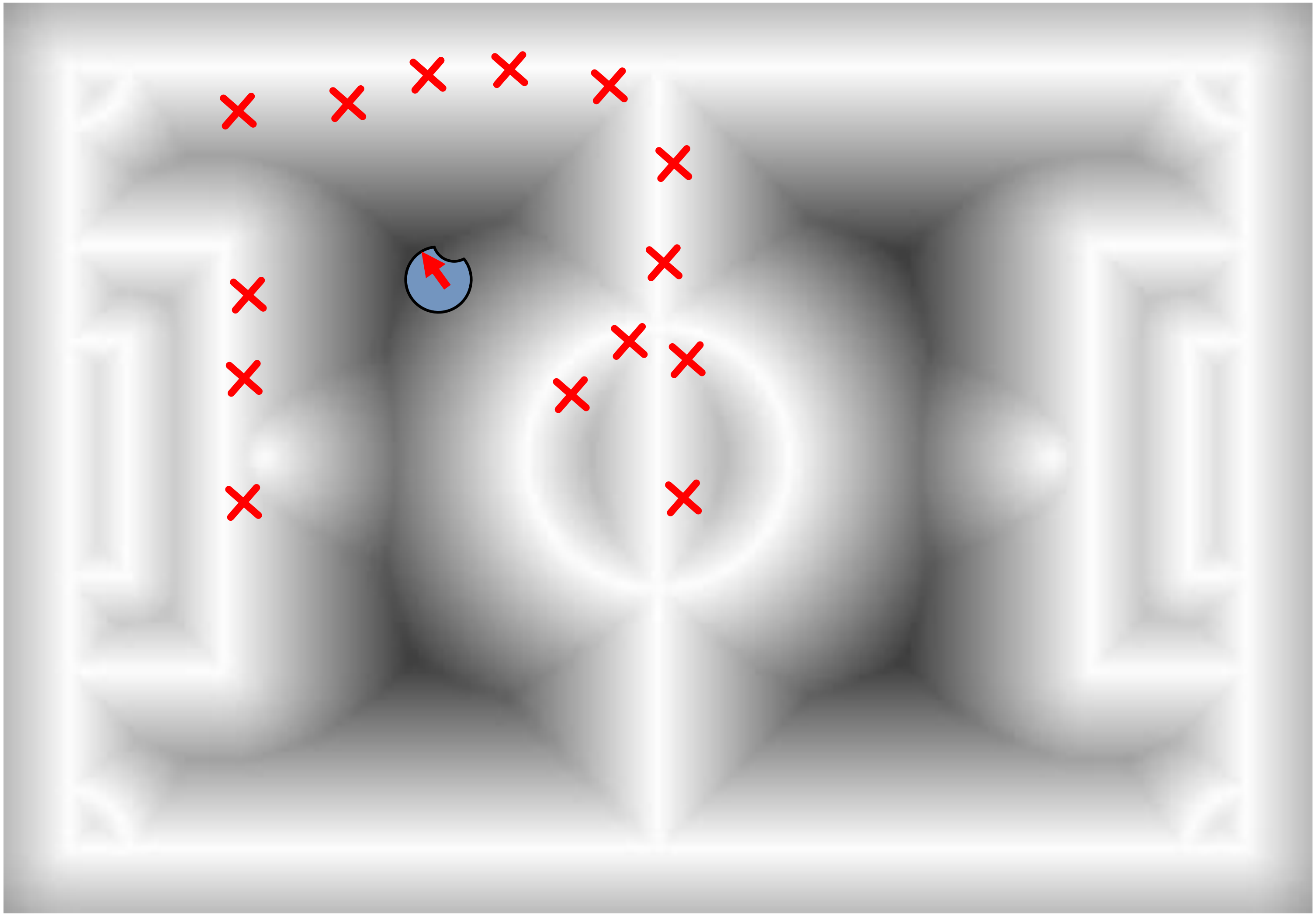
optPos = optimize(trialPos, whiteLines)



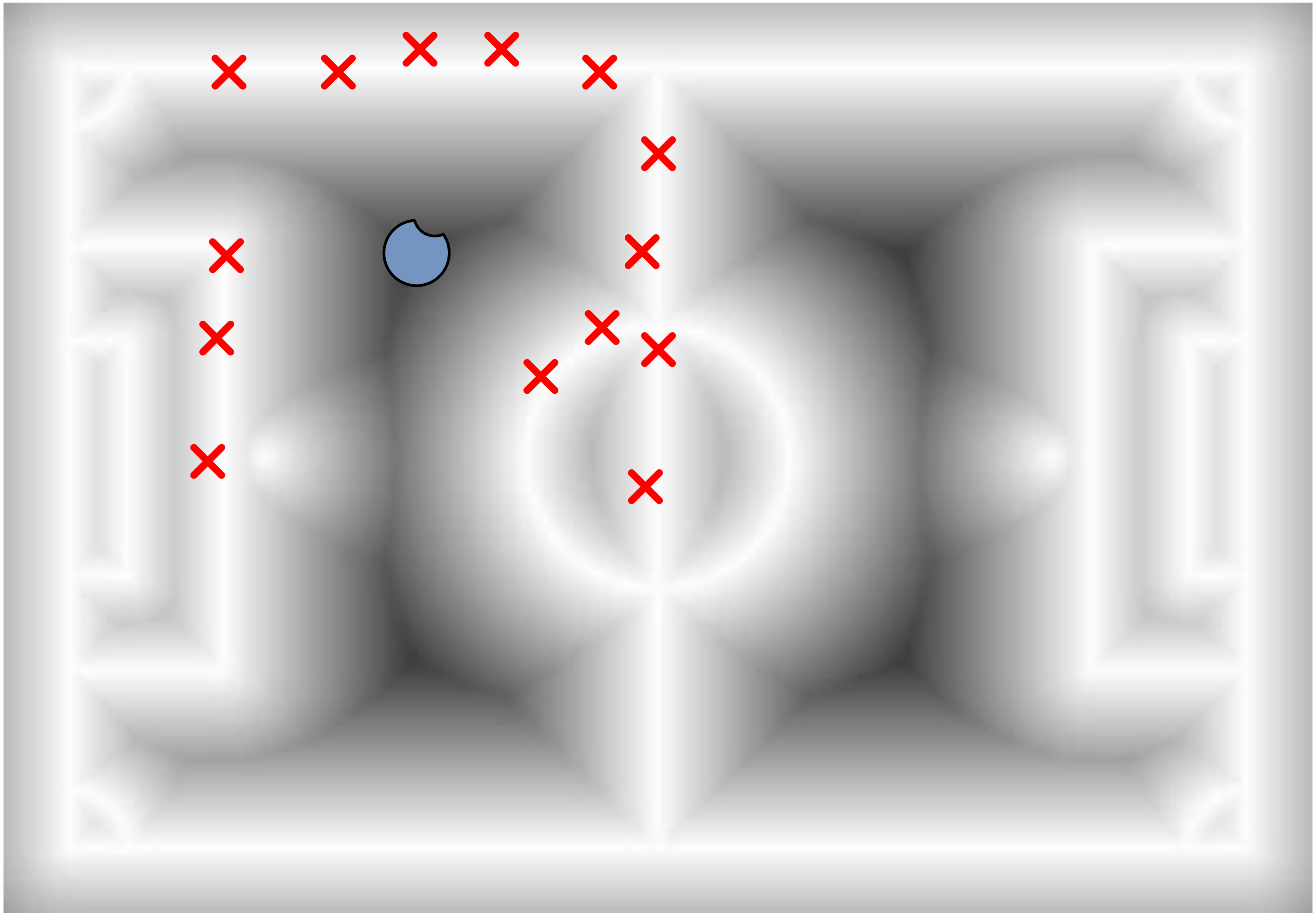
optPos = optimize(trialPos, whiteLines)



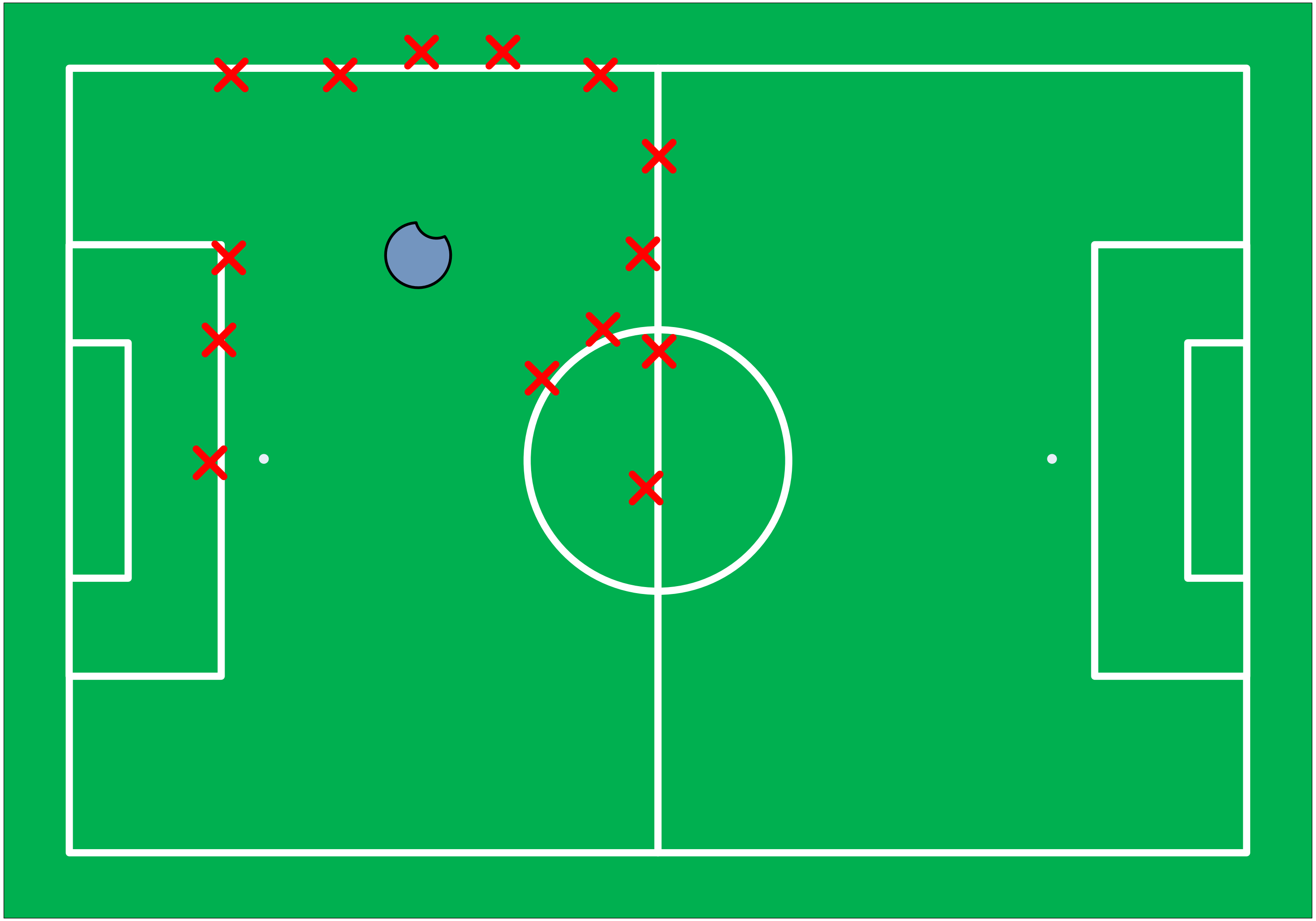
optPos = optimize(trialPos, whiteLines)



optPos = optimize(trialPos, whiteLines)



optPos = optimize(trialPos, whiteLines)



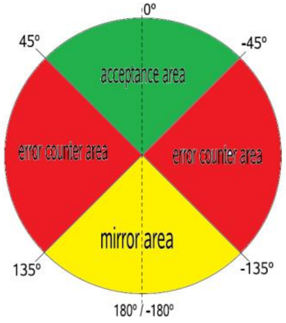
CAMBADA Localization - Tracking

Robot optimizes previous position (updated with odometry) and also 4 positions with:

fixed offsets of 60cm in xx and yy positive and negative dirs

small random heading offset

The optimized position with the smallest error is taken as the best estimate



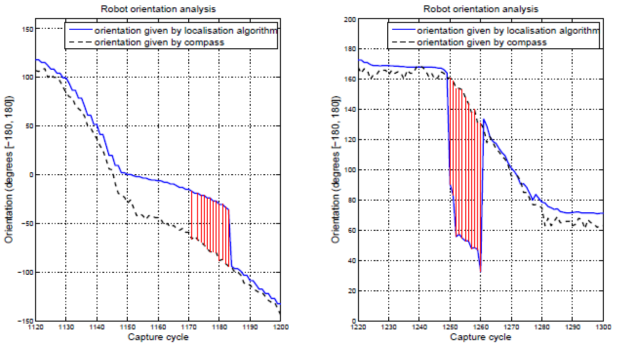
Detection of symmetric position

Compass based

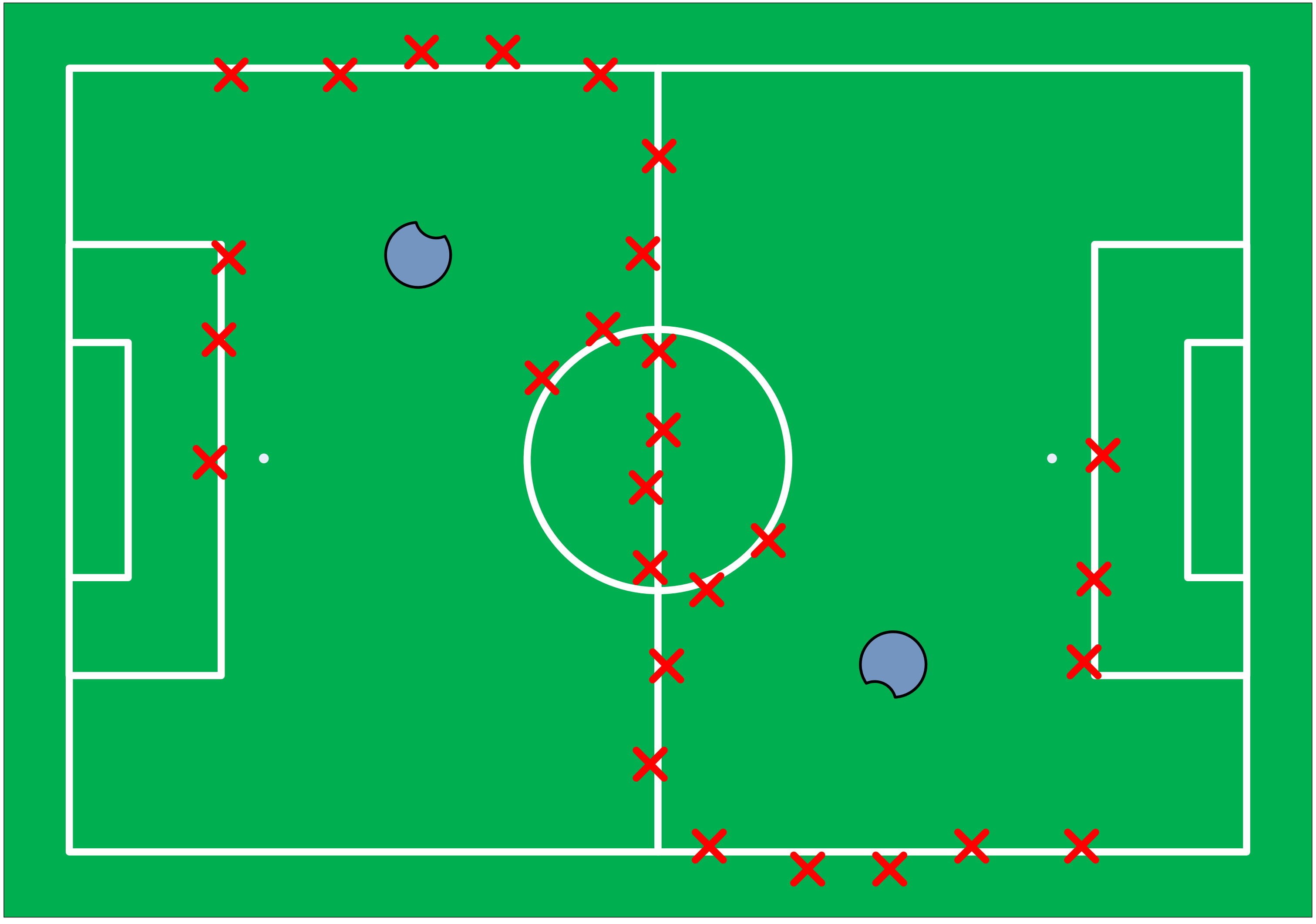
Detection of lost condition

Compass based

Forces global localization algorithm



Symmetric position problem



CAMBADA Localization – Global

A grid of trial points is used as candidate position for optimization

Grid spans one half of the field

Resolution of 1m over xx and yy

Initial heading may be:

- Based on compass (allows use without human intervention)
- Fixed, ex: robot oriented towards positive xx (for fatidic fields)

Optimized position with smallest error is chosen

A set of 4 neighbors of smallest error position (using 40cm offsets) are still checked for better precision

Takes about 1s to complete

Current problems

Compass is not usable in every environment

Method used for selecting best candidate may lead to instability

Lack of systematic evaluation of uncertainties in odometry, compass, white line distances, etc.

Proposed plan

Integration of inertial sensor information to improve localization

Use other methods (which?) for disambiguation of symmetric positions

Filter neighbor selection

Enable *Cambódromo* with a fixed vision system

Develop a localization debug tool

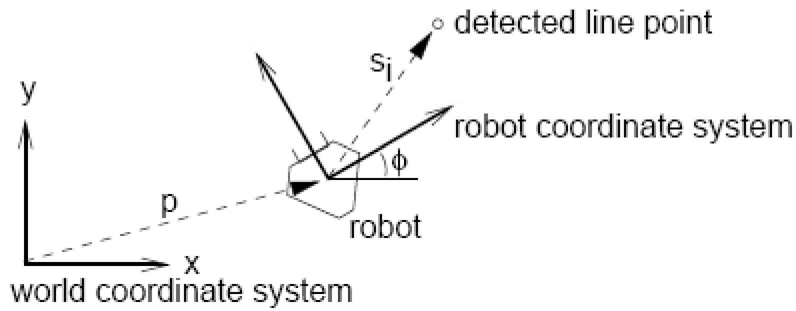
Use a more sophisticated Kalman filter

Test other localization methods

Localization in MSL

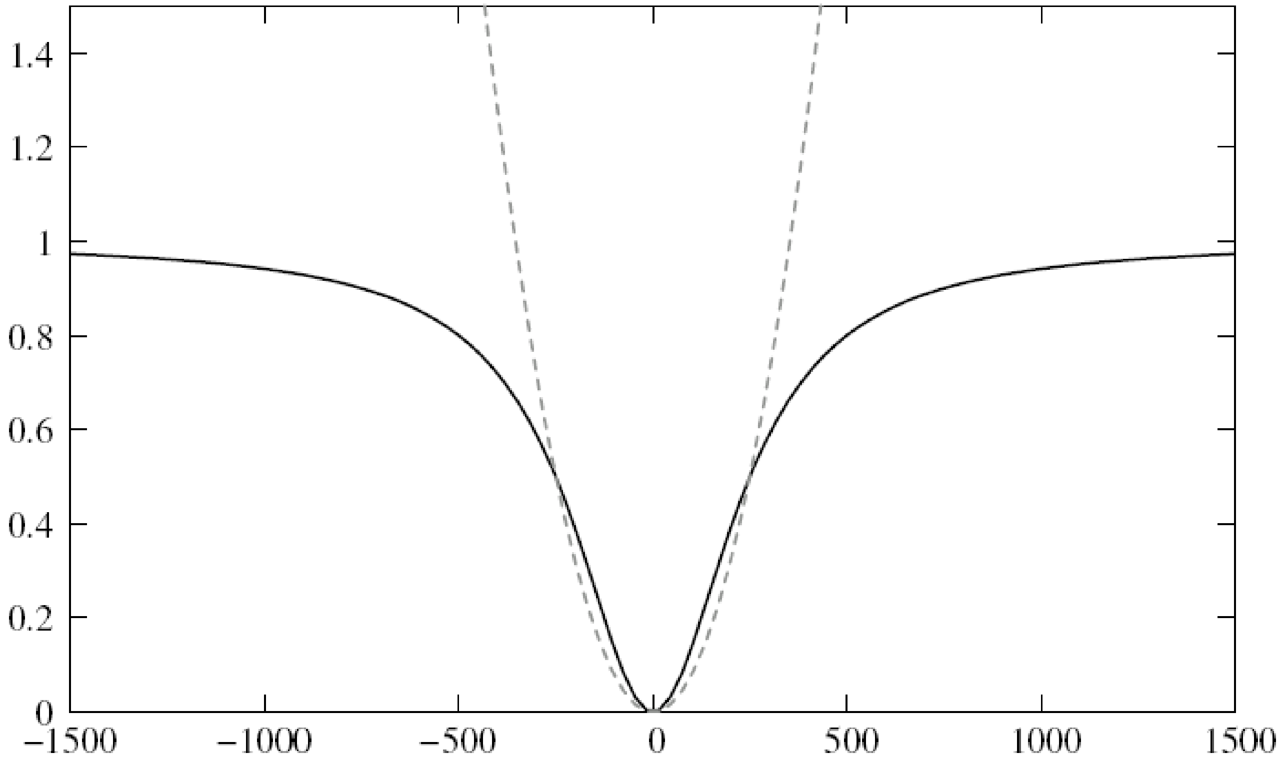
$$\underset{\boldsymbol{p},\phi}{minimize} \quad E := \sum_{i=1}^{n} err(d(\boldsymbol{p} + \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \boldsymbol{s_i}))$$

- p, θ are the position and heading
- s_i is the position of a detected white line
- Mapping d() gives the distance from a point in the field to the closest white line



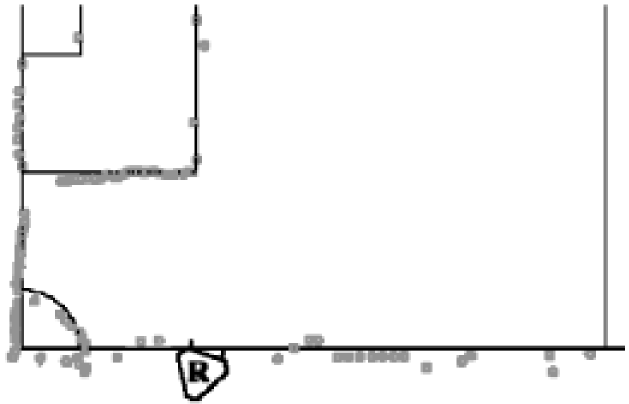
Localization in MSL

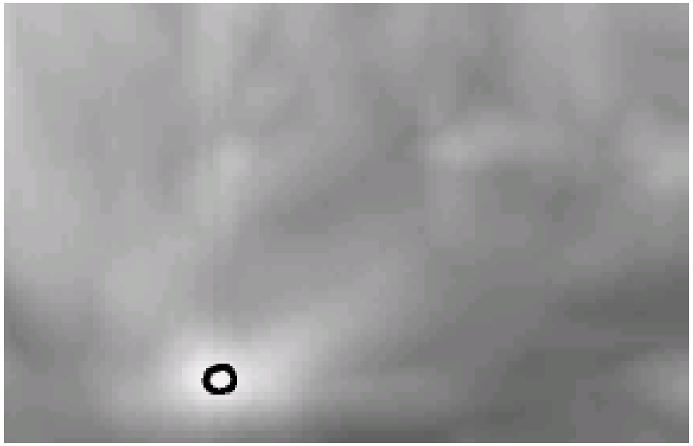
- Err function



Localization in MSL

- Position Estimation, error function

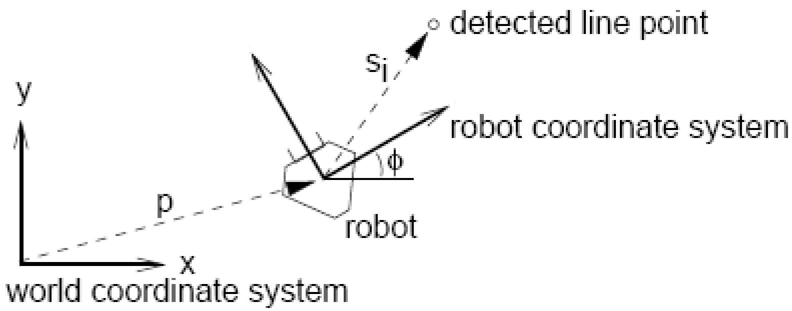




Localization in MSL

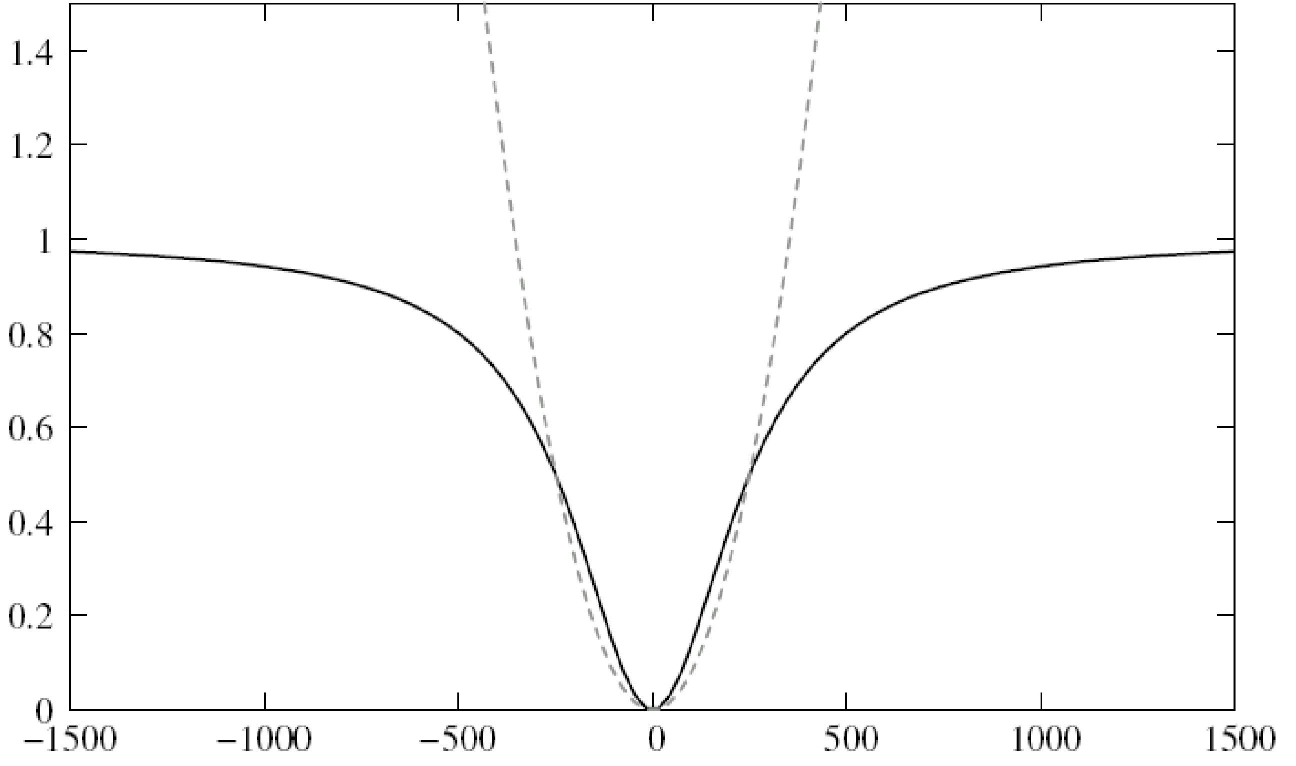
$$\underset{\boldsymbol{p},\phi}{minimize} \quad E := \sum_{i=1}^{n} err(d(p + \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix} \boldsymbol{s_i}))$$

- p, θ are the position and heading
- s_i is the position of a detected white line
- Mapping d() gives the distance from a point in the field to the closest white line



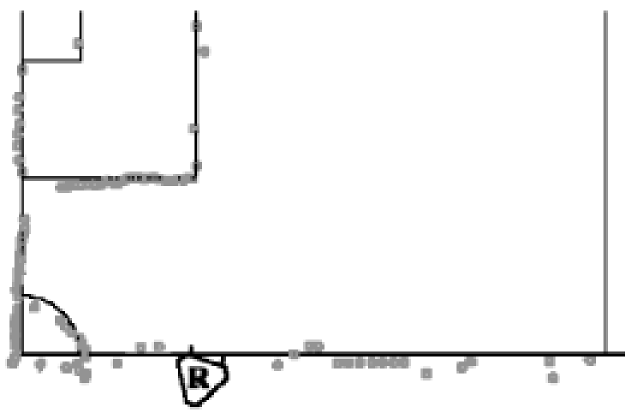
Localization in MSL

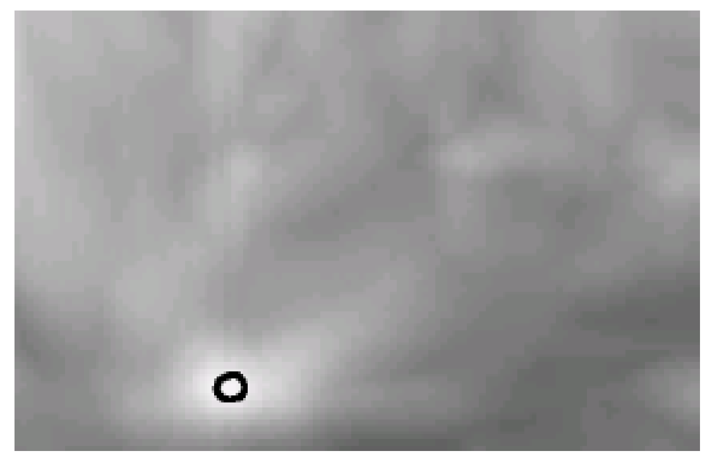
- Err function



Localization in MSL

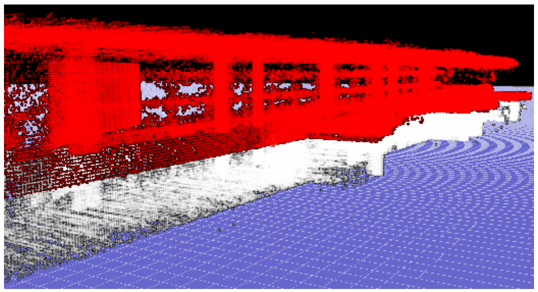
- Position Estimation, error function

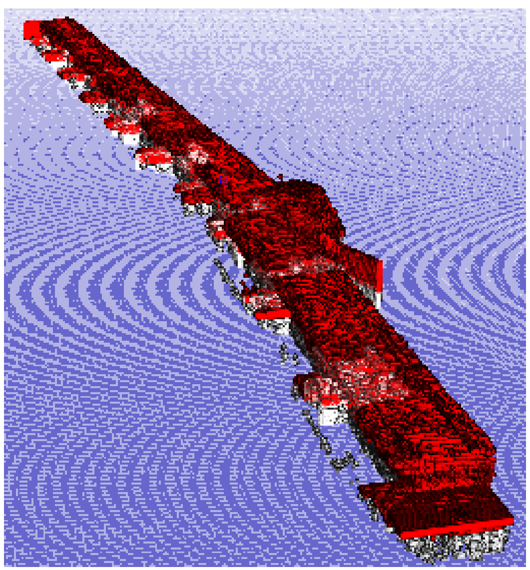


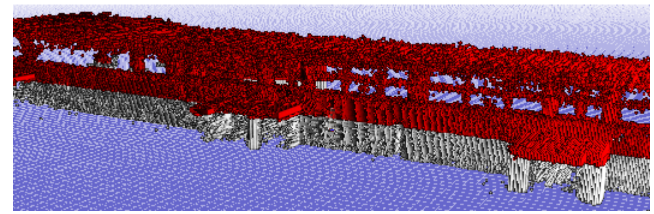


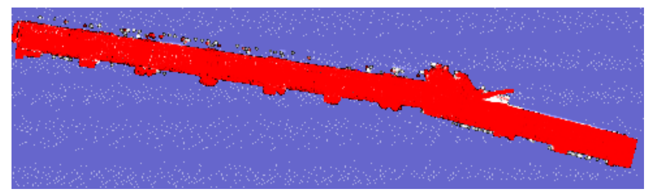
Localization in MSL

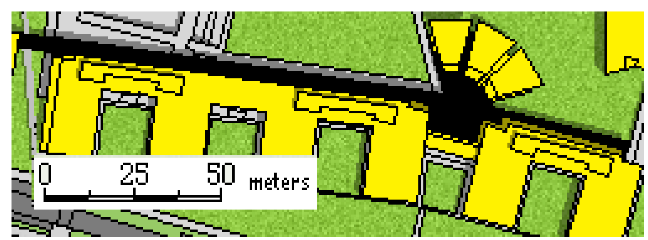
- Works very well for position tracking
- Global localization problem is solved by using several random seeds
- Detection of failure
  - Goal direction and position
    - Symmetrical and Complete failure
  - Compass










0
25
50
meters

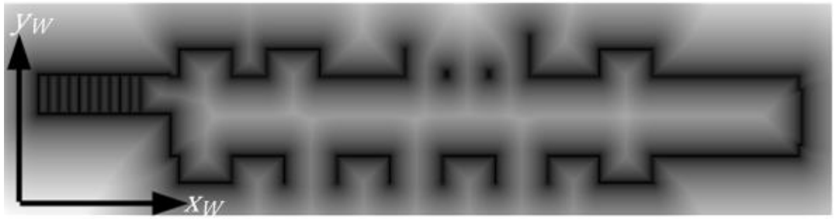


Fig. 9.1 Maps on a corridor where experiments were conducted. Distance Matrix.

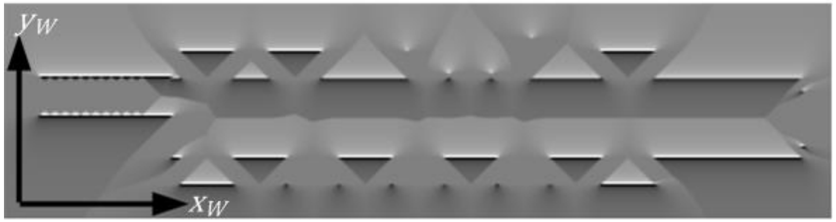


Fig. 9.2 Maps on a corridor where experiments were conducted. Gradient Matrix. Distance variation in direction y.

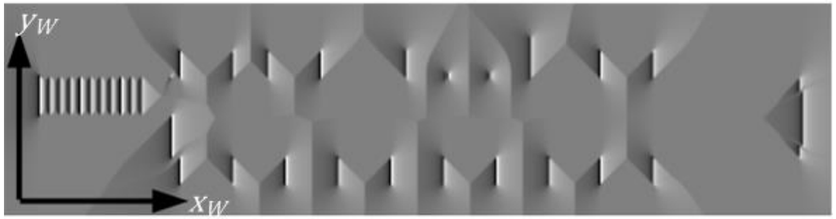


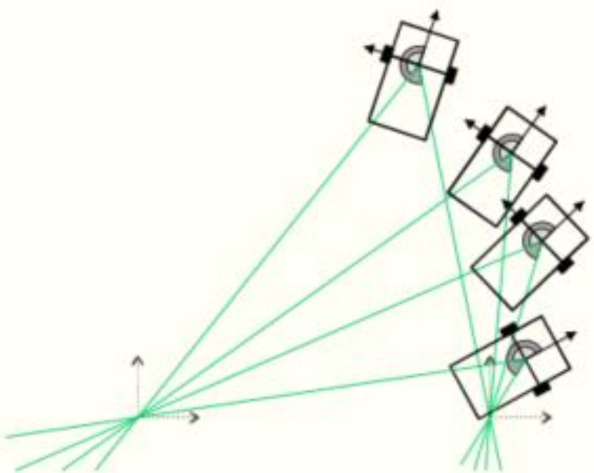
Fig. 9.3 Maps on a corridor where experiments were conducted. Gradient Matrix. Distance variation in direction x.

References

- Part II – Localization of *Probabilistic Robotics*, Sebastian Thrun, Wolfram Burgard, Dieter Fox, MIT Press, Cambridge, Massachusetts, London England, 2005. ISBN: 0-262-20162-3
- Martin Lauer, Sascha Lange and Martin Riedmiller: "Calculating the perfect match: An efficient and accurate approach for robot self-localisation", In A. Bredenfeld, A. Jacoff, I. Noda and Y. Takahashi, editors, RoboCup 2005: Robot Soccer World Cup IX, LNCS. Springer, 2006
- The Robotics Primer, Maja J. Mataric, The MIT Press, September 2007. ISBN: 0-262-63354-X

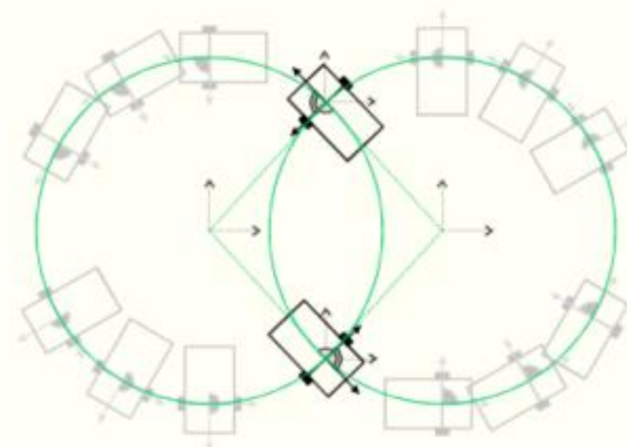
Practical

Two angles (internal to the robot) do not localize it (x,y,θ)



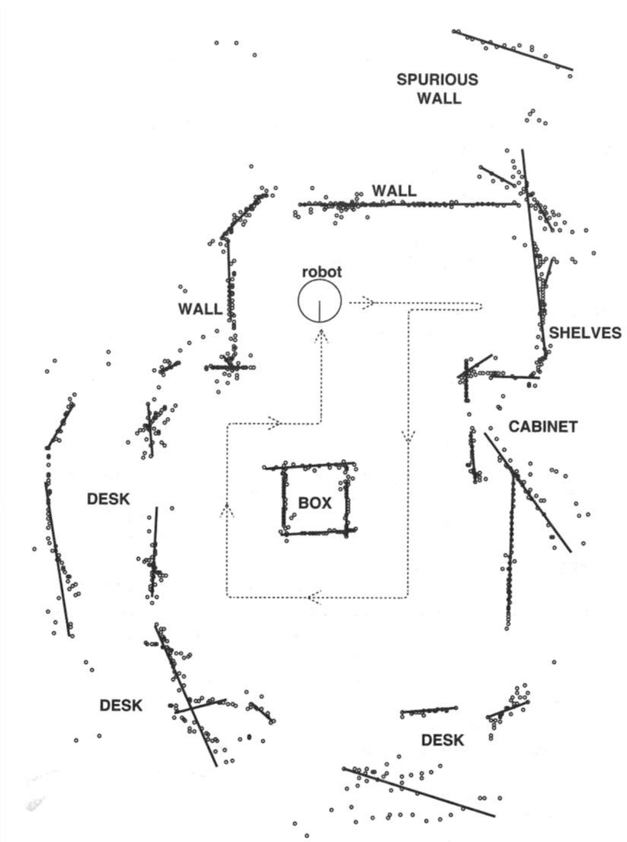
(shown are 4 positions of an infinite number of solutions)

Two measurements of (angle, distance) do not localize robot



(still two solutions eligible)

Practical



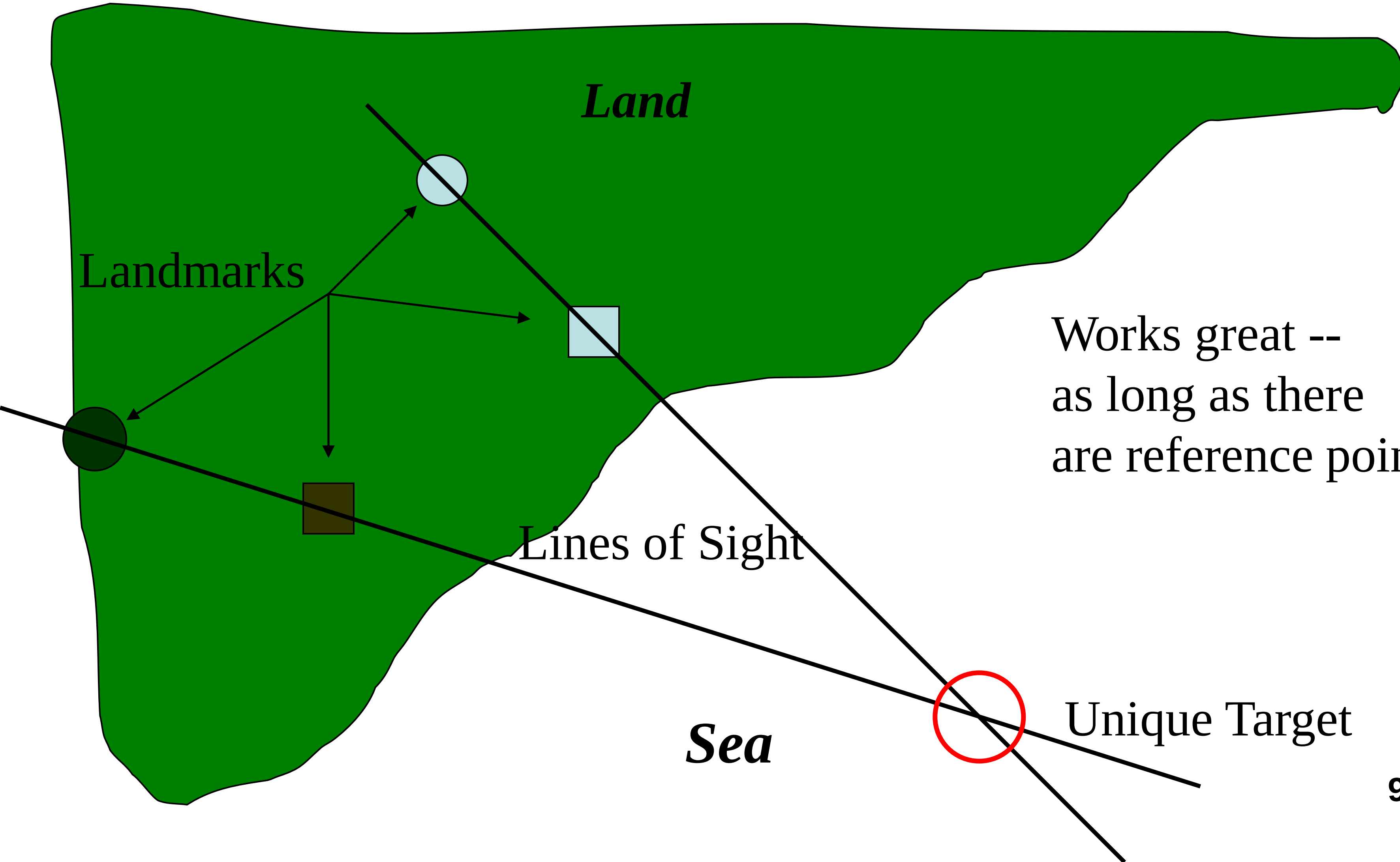
Includin stuff from
Intro Robotics from
New York City College
Slides

Localization - Practical

Plot the possible localizations for:

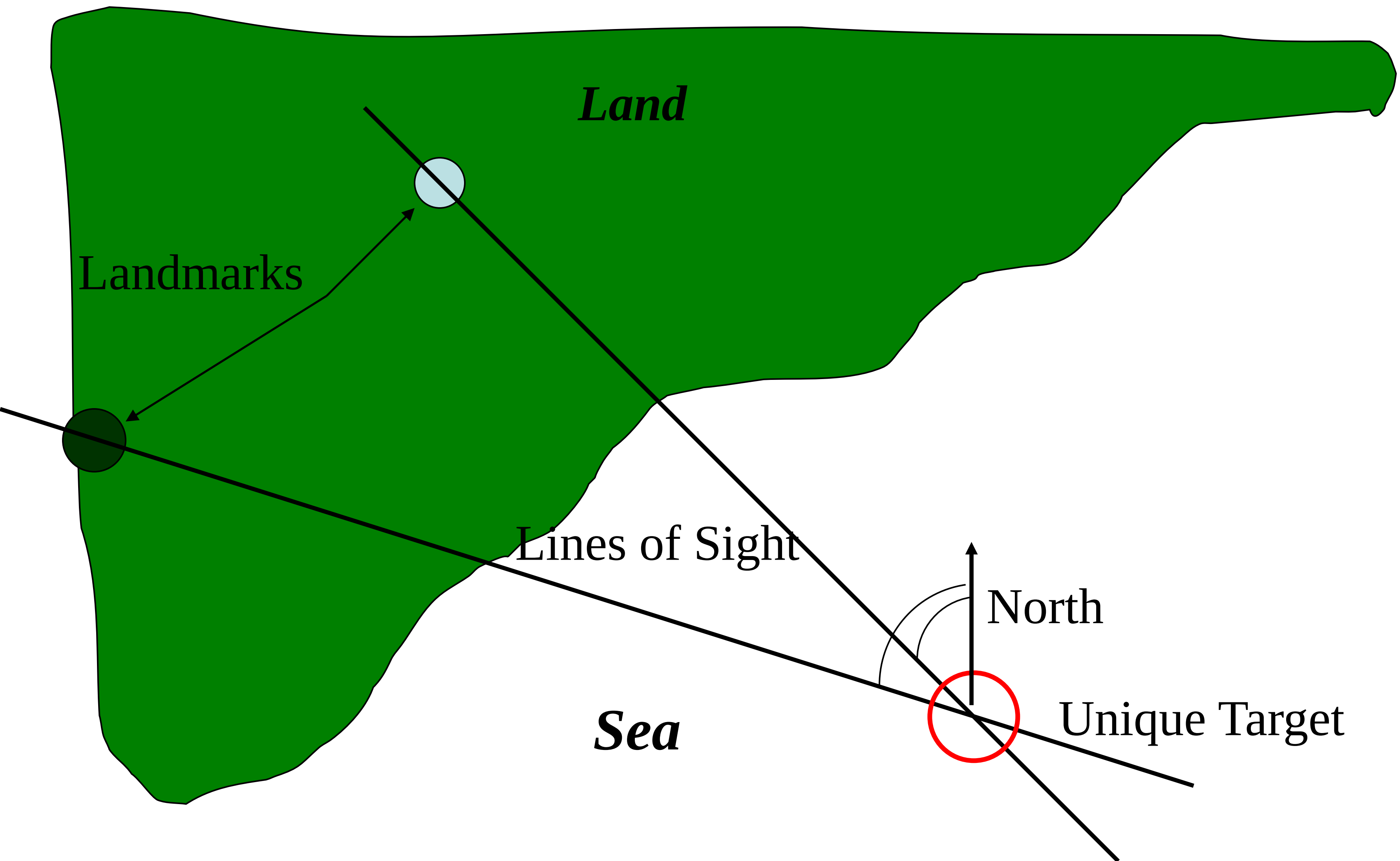
a) 1 distance to a known segment (wall) ☹
b) 1 distance and 1 angle to a segment ☹
c) 1 distance and 1 angle to a pt (corner) ☹
d) 2 dist. + 2 angles to 2 known “points” 😐
e) 3 dist. + 3 angles to 3 known “points” ☺
f) 1 dist. + 1 relative angles to beacon (non-round, determinable angle) ☺

Triangulation1 - practical



Works great --
as long as there
are reference points!

Triangulation2 - practical



Numeric Exercise

a) Robot at $(x,y,\theta)=(0,0,0)$
…moves straight front 1±0.1m… where is it?

b) Robot at $(x,y,\theta)=(0,0,[0\pm10^{\circ}])$
…moves 1 m… where is it?

c) Robot at $(x,y,\theta)=(0, [-.5,.5],[-.5,.5])$
Robot moves 1 m and skids $\theta=[-.5,.5]$
Where is it?

Exercise A

- Consider a grid of 0..20 by 0..20 cells
- Consider a beacon at position = (8,8)
- Consider a single range sensor that measures distance to beacon
- Do a grid search and display all possible positions for a measurement of 6

Exercise B

- Consider a grid of 0..20 by 0..20 cells
- Consider 2 beacons:
  Beacon1 at (8,8) and Beacon2 at (11,13)
- Consider a magical B1 Beacon1 range sensor and B2 Beacon2 range sensor
- Do a grid search and display all possible positions for a B1=6 and B2=4

Exercise C

Part 2-

[Grid 0..20 by 0..20 + 1 Beacon at (8,8)]

- The robot now has (x,y,θ) - pose
- The robot now has a sensor that returns heading and distance to beacon
- Augment the pose grid to 21x21x36;
  do a grid search to display all possible poses for the reading (0º, 6)

Localization

Luís Paulo Reis
lpreis@fe.up.pt
Director/Researcher LIACC
Associate Professor at FEUP/DEI

Armando Sousa
asousa@fe.up.pt
Researcher INESC-TEC
Assistant Professor at FEUP/DEEC

Also, a very special Thank You to: ***Prof. Nuno Lau, IEETA, U. Aveiro***