

Sensor Fusion

Robótica Móvel e Inteligente

José Luís Azevedo, Bernardo Cunha, Pedro Fonseca,
Nuno Lau e Artur Pereira

Ano Letivo 2021/2022
IRIS/IEETA – DETI – Universidade de Aveiro

- Introduction to Sensor Fusion
- Kalman Filter
- Particle Filter
- Conclusion

- **Act of combining sensory data** or data derived from sensory data from disparate sources
- **The resulting information is “better”** than it would be possible when the sources were used individually
- **“Better”** is defined according to the context. Can be **more accurate, more complete, a different view**, etc.
- Using a broader definition, we can speak of **Information Fusion**
- Sensor Fusion is usually considered a subset of information fusion, although the terms are often used with the same meaning
- Another very used term for this task is Multi-Sensor Data Fusion

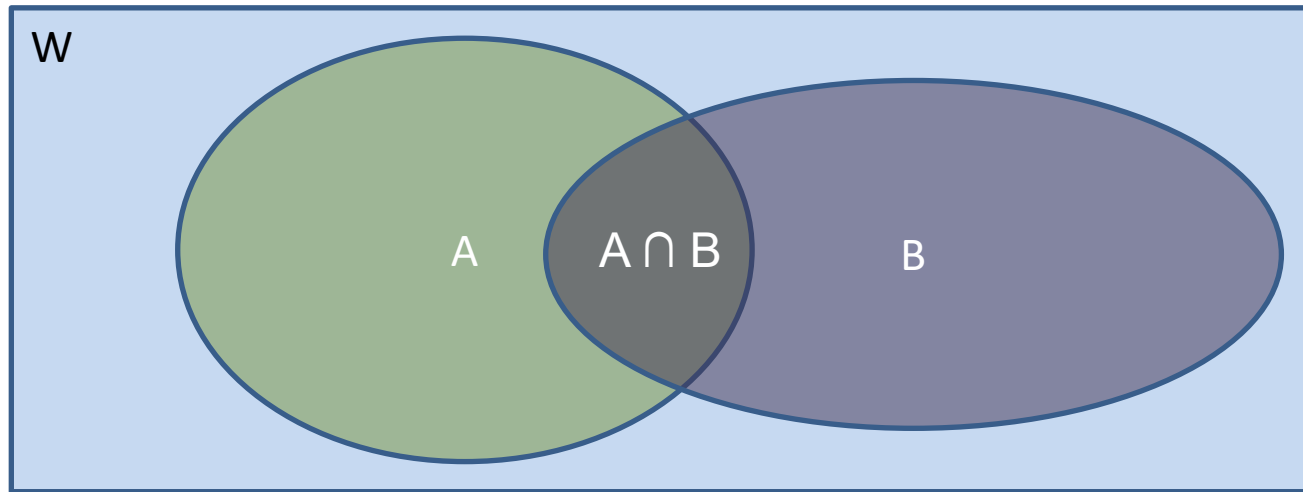
- Usually based on **modeling the sensors** and **modeling the system** being measured
- Most common methodologies are **probability based**
- However, some methodologies try to overcome some limitations of probability models (complexity, inconsistency, precision of models, uncertainty about uncertainty)
- Some of these methodologies are:
 - Interval calculus
 - Fuzzy logic
 - Theory of evidence (Dempster-Shafer methods)

- Determine the probability of a state/event, given the result of other states/events that are related.

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}$$

- $P(A | B)$ is the probability of A given that B is true, called **conditional probability**
- $P(B | A)$ is the probability of B given that A is true, called conditional probability
- $P(A)$ and $P(B)$ are the **marginal probabilities** of A and B

Bayes Rule



- Consider that the probability of a given event is related to its area in the above diagram, and that $area(W) = 1$
- $P(A) = area(A)$ $P(B) = area(B)$
- $$P(A | B) = \frac{area(A \cap B)}{area(B)} = \frac{\frac{area(A \cap B)}{area(A)} \cdot \frac{area(A)}{area(W)}}{area(B)} = \frac{P(B | A) \cdot P(A)}{P(B)}$$

- Applying Bayes Rule to determine the probability of being sick if the test for the disease is positive (+).
 - Assumptions:
 - Tests are correct 99% (test are positive with 99% probability if the person is sick, and negative with 99% if the person is not sick)
 - Disease is rare, happening only 1 in every 10000 people
 - If a person is tested and result is positive which is the probability of being sick?

<https://math.hmc.edu/funfacts/medical-tests-and-bayes-theorem/>

- Applying Bayes Rule to determine the probability of being sick if the test for the disease is positive (+).
 - Assumptions:
 - Tests are correct 99% (test are positive with 99% probability if the person is sick, and negative with 99% if the person is not sick)
 - Disease is rare, happening only 1 in every 10000 people
 - If a person is tested and result is positive which is the probability of being sick?

$$P(\text{sick} \mid +) = P(+ \mid \text{sick}) \cdot P(\text{sick}) / P(+)$$

$$P(+ \mid \text{sick}) = 0.99 \quad P(\text{sick}) = 1/10000$$

$$\begin{aligned} P(+) &= P(+ \mid \text{sick}) \cdot P(\text{sick}) + P(+ \mid \text{not sick}) \cdot P(\text{not sick}) \\ &= 0.99 \times 1/10000 + 0.01 \times 9999/10000 = 0.010098 \end{aligned}$$

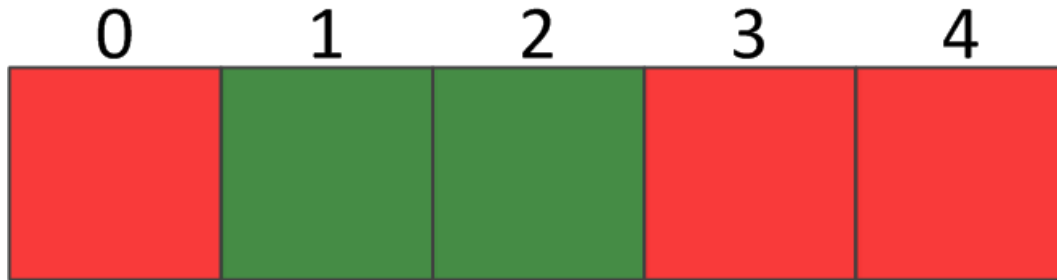
$$P(\text{sick} \mid +) = 0.99 \times 1/10000 / 0.010098 = 0.009804 \approx \mathbf{1\%}$$

<https://math.hmc.edu/funfacts/medical-tests-and-bayes-theorem/>

- Application of the **Bayes' Rule** when:
 - X_t is the state vector at time t .
 - U_t is the control vector used to drive from state X_{t-1} to X_t .
 - Z_t is the observation of the state at time t
- At an instant t an estimation of the current state can be achieved by application of Bayes' Rule:

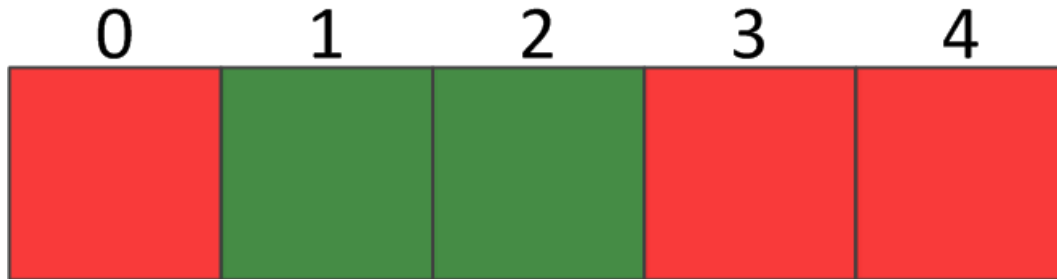
$$P(X_t|Z_t, U_t) = \frac{P(Z_t|X_t)P(X_t|Z_{t-1}, U_t)}{P(Z_t|Z_{t-1}, U_t)}$$

- Consider a grid world:

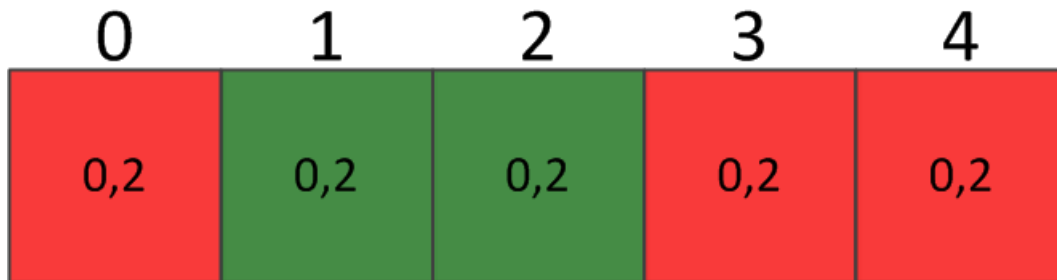


- Where is the robot?

- Consider a grid world:

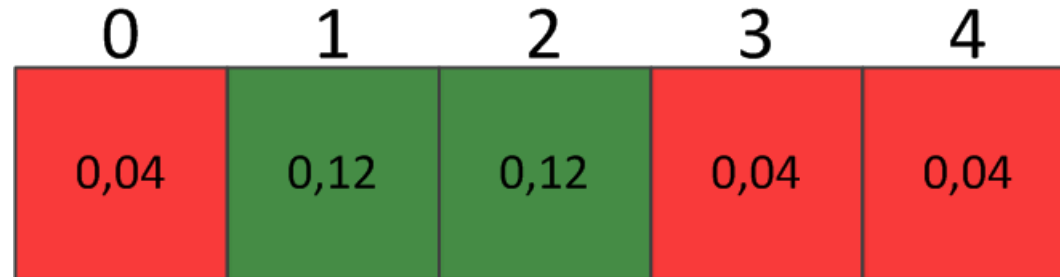


- Position probability distribution:



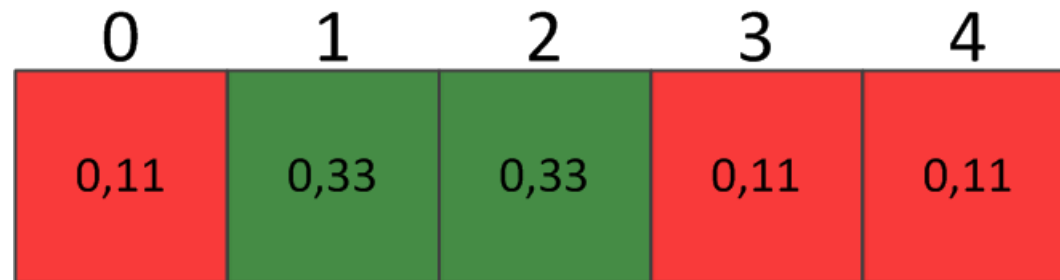
- The robot has a color sensor
- Sensor model
 - G_M : green measure; R_M : red measure
 - G_C : green cell; R_C : red cell
 - $P(G_M|G_C)=0.6$; $P(R_M|G_C)=0.4$
 - $P(G_M|R_C)=0.2$; $P(R_M|R_C)=0.8$
- The robot measures **green**
- Which is the new position estimate?

- Multiply cell values by $P(G_M|G_C)$ and $P(G_M|R_C)$



$$\sum P = 0,36$$

- Normalize



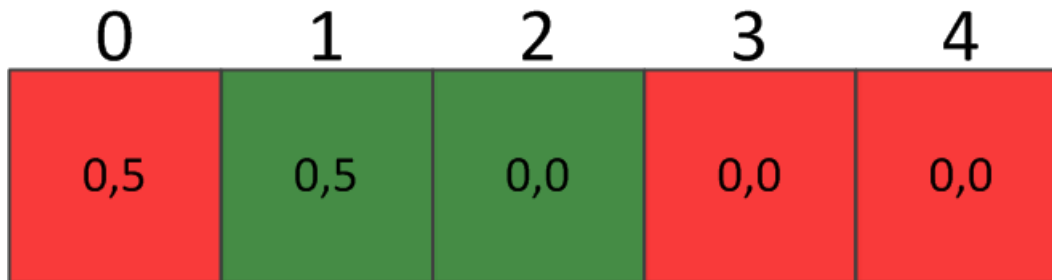
- **We have just applied Bayes' Rule!**

- Bayes' Rule

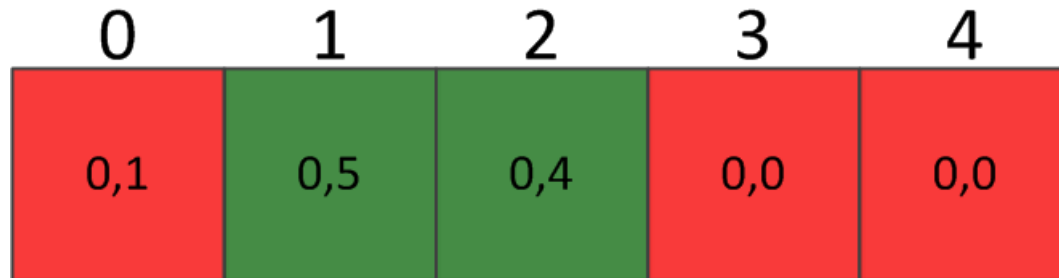
$$P(X_p|M) = \frac{P(M|X_p) * P(X_p)}{P(M)}$$

- $P(X_p)$ is the estimate before measurement integ.
- $P(X_p|M)$ is the posterior estimate
- $P(M)$ does not depend on X_p , so it can be considered as a constant that performs normalization
- **Measurement integration** is performed using the **product** of sensor model and previous estimate

- If the robot (tries) to move
- Action model
 A_R : Move Right action;
 $P(X+1 | A_R, X) = 0.8$; $P(X | A_R, X) = 0.2$
- Initial belief



- $P(1) = P(1 \mid A_R, 0) * P(0) + P(1 \mid A_R, 1) * P(1)$
- Predicted belief



- When the **robot moves** belief is updated through **convolution**

- Continuous environments
 - Measurement Integration (Product)

$$bel(X)_t = \eta p(Z_t|X) \hat{bel}(X)_t$$

- Motion update (Convolution)

$$\hat{bel}(X)_t = \int p(X|U_t, X') bel(X')_{t-1} dX'$$

- Assumptions
 - **Linear model** (transition, action and sensor)
 - **Every estimate is a gaussian**
 - Can be characterized by **mean** and **variance**
 - **Noise is gaussian** (mean=0)
- Integration of measures over time
- **Markovian assumption**
 - The next estimate only depends on the previous estimate
- Considers **action model** and **physics/observation model**

- **Product of gaussians distributions (measurement integration)**

$$\mu_P = \frac{\mu_1 \cdot \sigma_2^2 + \mu_2 \cdot \sigma_1^2}{\sigma_1^2 + \sigma_2^2} \quad \sigma_P^2 = \frac{\sigma_1^2 \cdot \sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

- **Convolution of gaussians distributions (motion forecast)**

$$\mu_C = \mu_1 + \mu_2 \quad \sigma_C^2 = \sigma_1^2 + \sigma_2^2$$

$$\hat{X}_t = F_t \hat{X}_{t-1} + B_t U_t + \omega_t$$

- \hat{X}_t is the **estimated state**
- F_t is the **state transition model**
- B_t is the **control-input model**
- U_t is the **control vector**
- w_t is the **process noise** with covariance

$$Q_t : \omega_t \sim N(0, Q_t)$$

$$\hat{Z}_t = H_t X_t + v_t$$

- \hat{Z}_t is the **measurement** taken at time t
- H_t is the **observation model** of the state/event
- v_t is the **observation noise** with covariance:

$$R_t : v_t \sim N(0, R_t)$$

- The filter state is represented by two variables:
 - X_t is the **estimate of the state** at time t
 - P_t is the **measure of estimated accuracy** of the process
- The filter works in **two steps**:

Forecast

$$\begin{aligned}\bar{X}_t &= F_t X_{t-1} + B_t U_t \\ \bar{P}_t &= F_t P_{t-1} F_t^T + Q_t\end{aligned}$$

Measurement integration

$$\begin{aligned}K_t &= \frac{\bar{P}_t H_t^T}{H_t \bar{P}_t H_t^T + R_t} \\ X_t &= \bar{X}_t + K_t (Z_t - H_t \bar{X}_t) \\ P_t &= (I - K_t H_t) \bar{P}_t\end{aligned}$$

Particle Filter (Monte Carlo)

- Integration of measures over time
- Is **non-parametric** and thus can cope with a several types of distributions, rather than just Gaussian
- The samples of the state are called **particles**
- **Each particle** is a **concrete instantiation of the state** at time t
- This filter also works in **two main steps**, applied to every single particle:
 - **Evolution** of the current state, **weighing** and **creation** of a dataset
 - **Resampling** of the dataset for the next evaluation

Particle Filter (Monte Carlo)

- Consider M the number of particles used and X_t the particle set at time t

Evolution and weighing

for $m = 0$ to M

$$x_t^{[m]} \sim p(x_t | u_t, x_{t-1}^{[m]})$$

$$w_t^{[m]} = p(z_t | x_t^{[m]})$$

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

endfor

Resampling

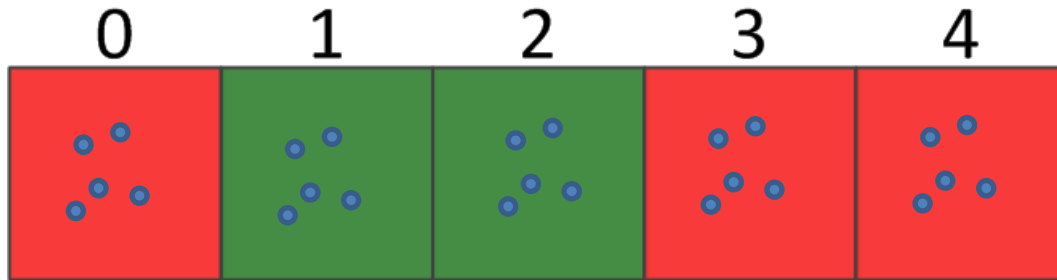
for $m = 0$ to M

draw i with prob $\propto w_t^{[i]}$

add $x_t^{[i]}$ to X_t

endfor

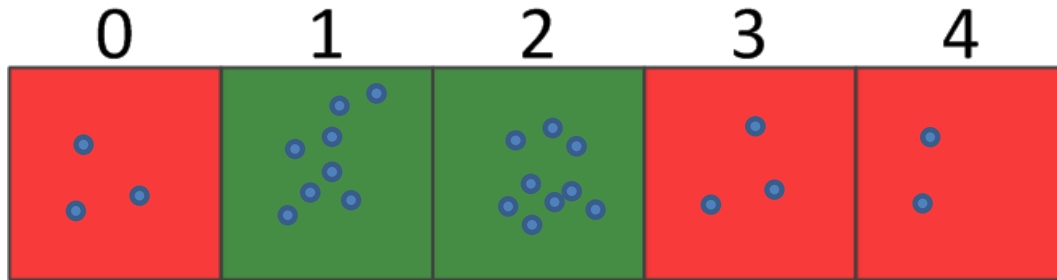
- Initial particle distribution:



- If green is seen, particles in green cells have higher weights
- They will have higher chances of getting into the new particle set

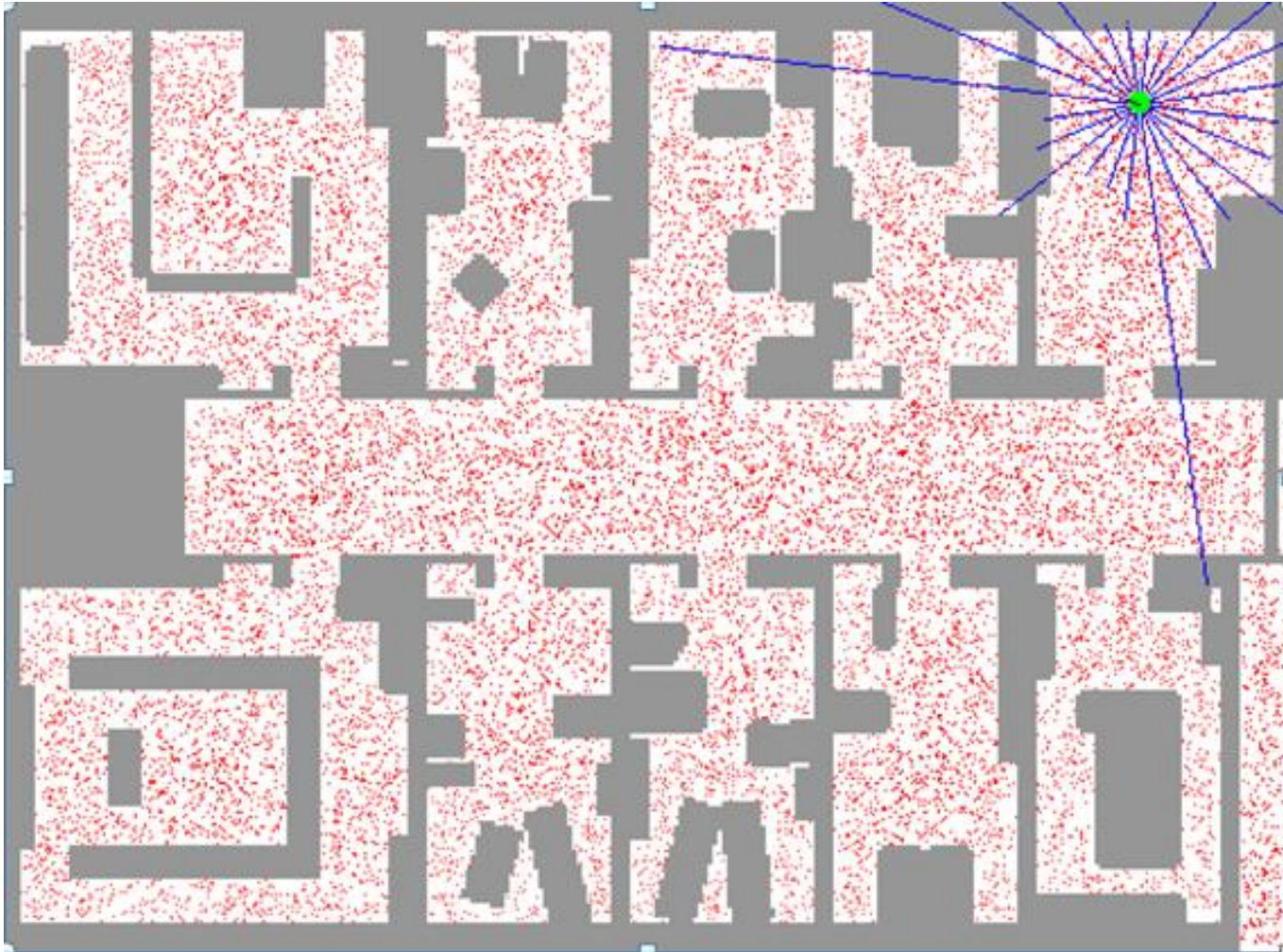
Grid World - Resampling

- After green is seen:

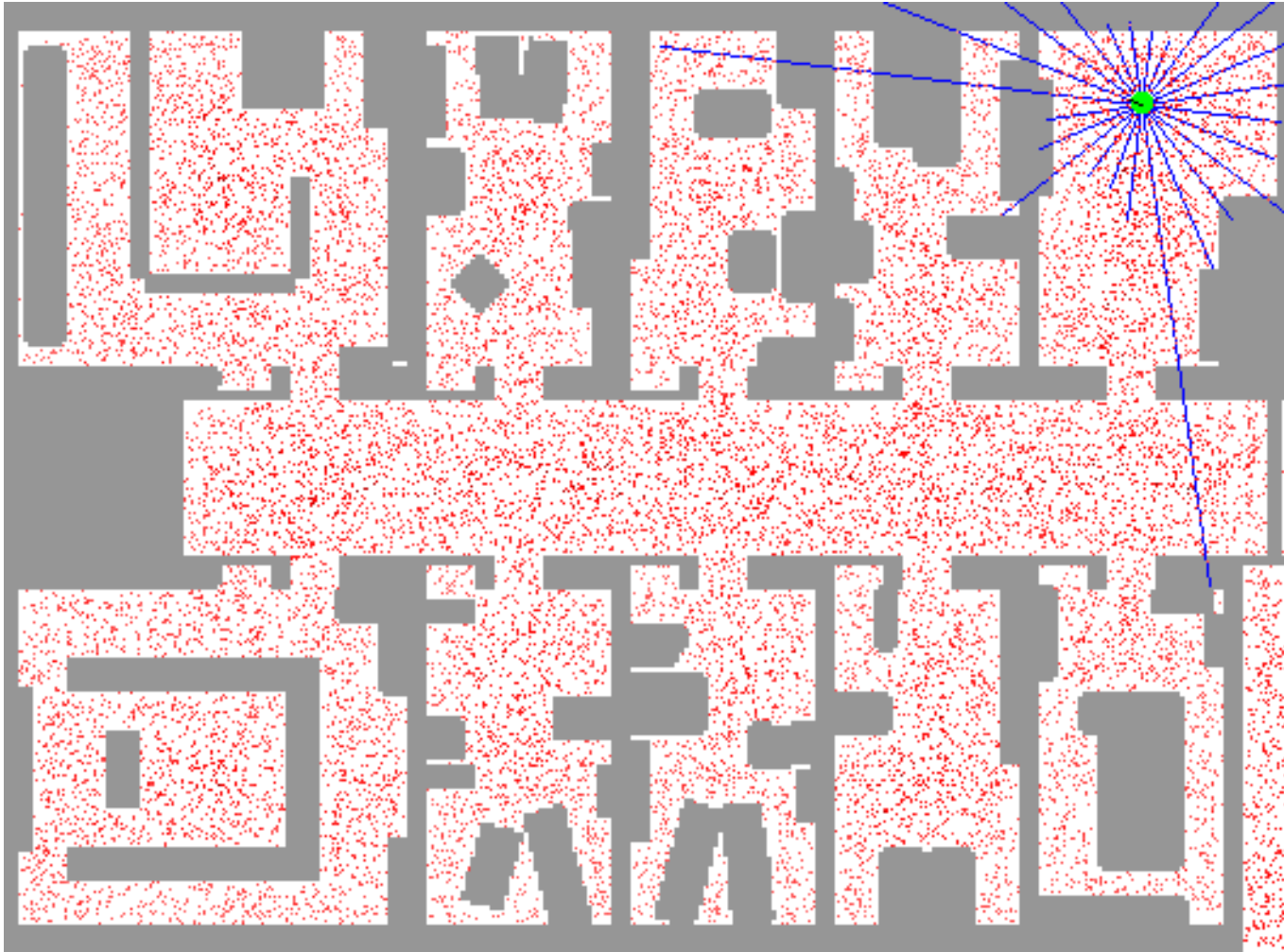


- The motion model may be applied directly to each particle

Particle Filter (Monte Carlo)



Particle Filter (Monte Carlo)



<https://rse-lab.cs.washington.edu/projects/mcl/animations/global-floor.gif>

- Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics. The MIT Press, 2005.
- Sebastian Thrun, Artificial Intelligence for Robotics, Udacity, www.udacity.com
- Hugh Durrant-Whyte and Tom Henderson. Multisensor data fusion. In Siciliano and Khatib, editors. Springer Handbook of Robotics. 2nd ed. Springer, 2016, pages 867–890.