FastSLAM

Hakan Aşık

12.01.2020

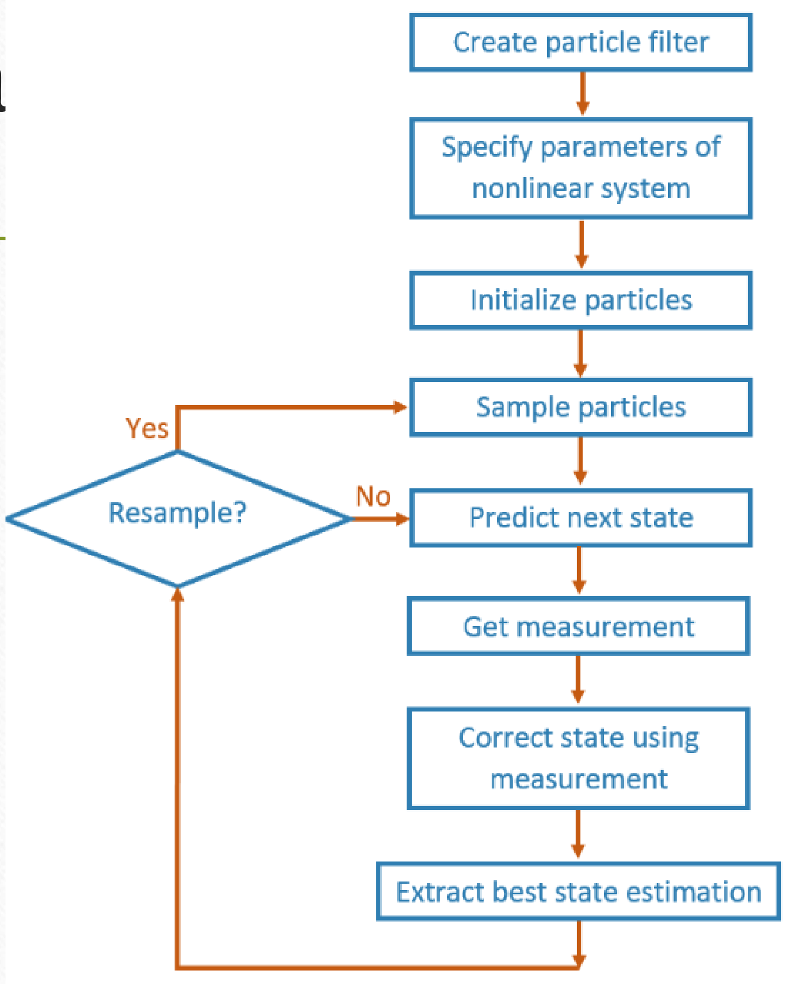
Outline

- What is FastSLAM?
- Particle Filter workflow
- FastSLAM workflow
- Demo

FastSLAM

- Particle filter based SLAM
- Decomposes the SLAM problem into :
  - a robot localization problem, plus
  - a collection of landmark estimation problems that are conditioned on the robot pose estimate.
- Stems from the basis : all individual landmark estimation problems are independent if one knew the robot's path s_t and the correspondence variables n_t
- Reduces running time to O(M*logK) time

Pa flow



Create particle filter
Specify parameters of nonlinear system
Initialize particles
Sample particles
Predict next state
Get measurement
Correct state using measurement
Extract best state estimation
Resample?
Yes
No

FastSLAM1.0_known_correspondence$(z_t, c_t, u_t, \mathcal{X}_{t-1})$:

 for $k = 1$ *to* N *do*   *// loop over all particles*

  Let $\left\langle x_{t-1}^{[k]}, \left\langle \mu_{1,t-1}^{[k]}, \Sigma_{1,t-1}^{[k]} \right\rangle, \ldots \right\rangle$ be particle k in $\mathcal{X}_{t-1}$

  $x_t^{[k]} \sim p(x_t \mid x_{t-1}^{[k]}, u_t)$ *// sample pose*

  $j = c_t$ *// observed feature*

  if feature j *never seen before*

   $\mu_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})$ *// initialize mean*

   $H = h'(\mu_{j,t}^{[k]}, x_t^{[k]})$ *// calculate Jacobian*

   $\Sigma_{j,t}^{[k]} = H^{-1}\, Q_t\, (H^{-1})^T$ *// initialize covariance*

   $w^{[k]} = p_0$ *// default importance weight*

  else

   $\hat{z}^{[k]} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ *// measurement prediction*

   $H = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ *// calculate Jacobian*

   $Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ *// measurement covariance*

   $K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ *// calculate Kalman gain*

   $\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}^{[k]})$ *// update mean*

   $\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ *// update covariance*

   $w^{[k]} = |2\pi Q|^{-\frac{1}{2}}\ \exp\left\{-\frac{1}{2}(z_t - \hat{z}^{[k]})^T\, Q^{-1}\, (z_t - \hat{z}^{[k]})\right\}$ *// importance factor*

  endif

  for all unobserved features j' *do*

   $\langle \mu_{j',t}^{[k]}, \Sigma_{j',t}^{[k]} \rangle = \langle \mu_{j',t-1}^{[k]}, \Sigma_{j',t-1}^{[k]} \rangle$ *// leave unchanged*

  endfor

 endfor

 $\mathcal{X}_t = \text{resample}\left(\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, w^{[k]} \right\rangle_{k=1,\ldots,N} \right)$

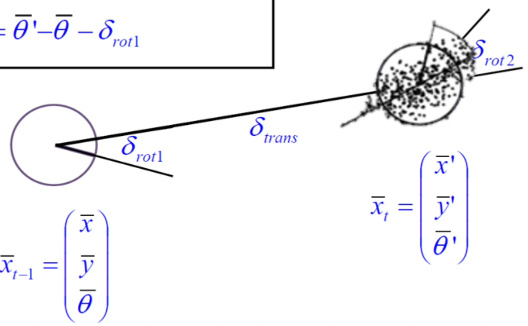
 return $\mathcal{X}_t$

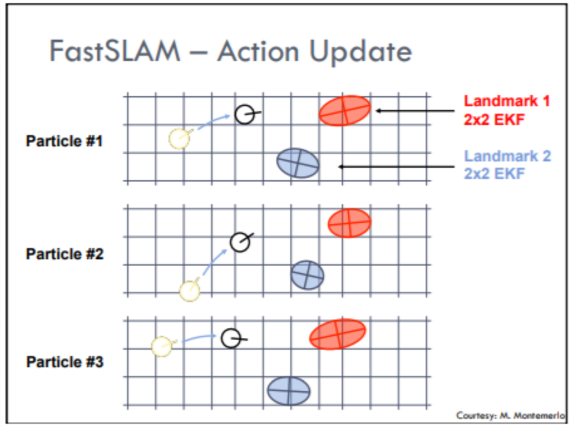
FastSLAM1.0_known_correspondence$(z_t, c_t, u_t, \mathcal{X}_{t-1})$:

```
for k = 1 to N do                                          // loop over all particles
    Let ⟨x_{t-1}^{[k]}, ⟨μ_{1,t-1}^{[k]}, Σ_{1,t-1}^{[k]}⟩, ...⟩ be particle k in X_{t-1}
    x_t^{[k]} ~ p(x_t | x_{t-1}^{[k]}, u_t)                // sample pose
    j = c_t                                                // observed feature
    if feature j never seen before
        μ_{j,t}^{[k]} = h^{-1}(z_t, x_t^{[k]})            // initialize mean
        H = h'(μ_{j,t}^{[k]}, x_t^{[k]})                   // calculate Jacobian
        Σ_{j,t}^{[k]} = H^{-1} Q_t (H^{-1})^T              // initialize covariance
        w^{[k]} = p_0                                      // default importance weight
    else
```

Odometry Model

$$\delta_{trans} = \sqrt{(\bar{x}'-\bar{x})^2 + (\bar{y}'-\bar{y})^2}$$
$$\delta_{rot1} = \operatorname{atan2}(\bar{y}'-\bar{y}, \bar{x}'-\bar{x}) - \bar{\theta}$$
$$\delta_{rot2} = \bar{\theta}' - \bar{\theta} - \delta_{rot1}$$





Sample Odometry Motion Model

```
Algorithm sample_motion_model(u_t, x_{t-1}):
    δ_rot1 = atan2(ȳ' - ȳ, x̄' - x̄) - θ̄
    δ_trans = sqrt((x̄' - x̄)^2 + (ȳ' - ȳ)^2)
    δ_rot2 = θ̄' - θ̄ - δ_rot1
    δ̂_rot1 = δ_rot1 - sample(α_1 δ_rot1^2 + α_2 δ_trans^2)
    δ̂_trans = δ_trans - sample(α_3 δ_trans^2 + α_4 (δ_rot1^2 + δ_rot2^2))
    δ̂_rot2 = δ_rot2 - sample(α_1 δ_rot2^2 + α_2 δ_trans^2)
    x' = x + δ̂_trans cos(θ + δ̂_rot1)
    y' = y + δ̂_trans sin(θ + δ̂_rot1)
    θ' = θ + δ̂_rot1 + δ̂_rot2
    return [x'  y'  θ']^T
```

else

$\hat{z}^{[k]} = h(\mu^{[k]}_{j,t-1}, x^{[k]}_t)$ *// measurement prediction*

$H = h'(\mu^{[k]}_{j,t-1}, x^{[k]}_t)$ *// calculate Jacobian*

$Q = H\ \Sigma^{[k]}_{j,t-1}\ H^T + Q_t$ *// measurement covariance*

$K = \Sigma^{[k]}_{j,t-1}\ H^T\ Q^{-1}$ *// calculate Kalman gain*

$\mu^{[k]}_{j,t} = \mu^{[k]}_{j,t-1} + K(z_t - \hat{z}^{[k]})$ *// update mean*

$\Sigma^{[k]}_{j,t} = (I - K\ H)\Sigma^{[k]}_{j,t-1}$ *// update covariance*

$w^{[k]} = |2\pi Q|^{-\frac{1}{2}}\ \exp\left\{-\frac{1}{2}(z_t - \hat{z}^{[k]})^T\ Q^{-1}\ (z_t - \hat{z}^{[k]})\right\}$ *// importance factor*

endif

for all unobserved features j' *do*

$\langle \mu^{[k]}_{j',t}, \Sigma^{[k]}_{j',t} \rangle = \langle \mu^{[k]}_{j',t-1}, \Sigma^{[k]}_{j',t-1} \rangle$ *// leave unchanged*

endfor

endfor

$\mathcal{X}_t = \text{resample}\left(\left\langle x^{[k]}_t, \left\langle \mu^{[k]}_{1,t}, \Sigma^{[k]}_{1,t} \right\rangle, \ldots, w^{[k]} \right\rangle_{k=1,\ldots,N} \right)$

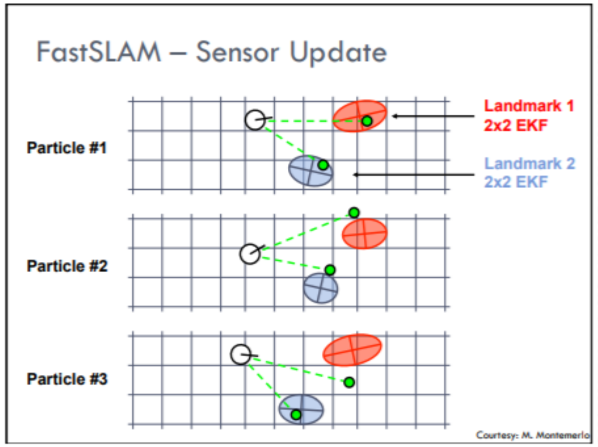
return $\mathcal{X}_t$

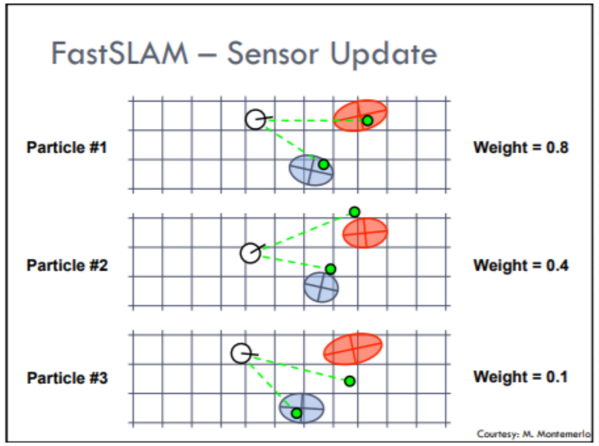
EKF Correction steps

$$K_t = \Sigma_t^- H_t^T (H_t \Sigma_t^- H_t^T + Q_t)^{-1}$$

$$\bar{x}_t^+ = \bar{x}_t^- + K_t(z_t - h(\bar{x}_t^-))$$

$$\Sigma_t^+ = (I - K_t H_t)\Sigma_t^-$$





else

$\hat{z}^{[k]} = h(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *measurement prediction*

$H = h'(\mu_{j,t-1}^{[k]}, x_t^{[k]})$ // *calculate Jacobian*

$Q = H\, \Sigma_{j,t-1}^{[k]}\, H^T + Q_t$ // *measurement covariance*

$K = \Sigma_{j,t-1}^{[k]}\, H^T\, Q^{-1}$ // *calculate Kalman gain*

$\mu_{j,t}^{[k]} = \mu_{j,t-1}^{[k]} + K(z_t - \hat{z}^{[k]})$ // *update mean*

$\Sigma_{j,t}^{[k]} = (I - K\, H)\Sigma_{j,t-1}^{[k]}$ // *update covariance*

$w^{[k]} = |2\pi Q|^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}^{[k]})^T Q^{-1}(z_t - \hat{z}^{[k]})\right\}$ // *importance factor*

endif

for all unobserved features j' do

$\langle \mu_{j',t}^{[k]}, \Sigma_{j',t}^{[k]} \rangle = \langle \mu_{j',t-1}^{[k]}, \Sigma_{j',t-1}^{[k]} \rangle$ // *leave unchanged*

endfor

endfor

$\mathcal{X}_t = \text{resample}\left(\left\langle x_t^{[k]}, \left\langle \mu_{1,t}^{[k]}, \Sigma_{1,t}^{[k]} \right\rangle, \ldots, w^{[k]} \right\rangle_{k=1,\ldots,N}\right)$

return $\mathcal{X}_t$

Resampling of particles enables to select particles based on the current state, instead of using the particle distribution given at initialization which helps more accurate tracking and long-term performance improvements.

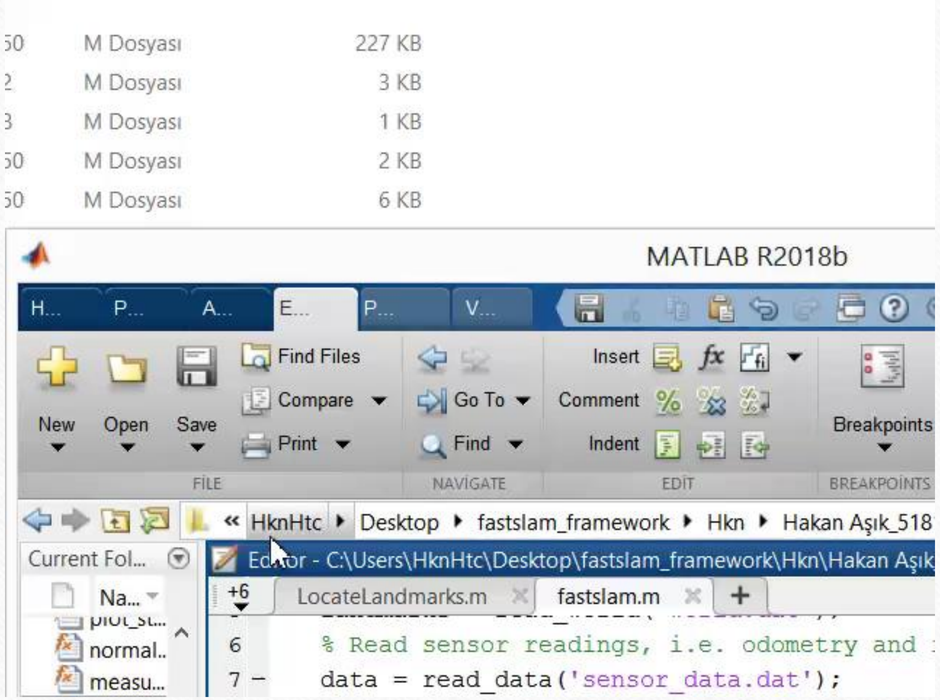
The minimum effective particle ratio(neff) is a measure of how well the current set of particles approximates the posterior distribution. The number of effective particles is calculated by

$$N_{eff} = \frac{1}{\sum_{i=1}^{N}\left(w^i\right)^2}$$

The lower the neff, less contribution to the state estimation.

Resampling help closer particles to be selected so as to contribute to the current state estimation and have higher weights.

Demo



M Dosyası 227 KB
M Dosyası 3 KB
M Dosyası 1 KB
M Dosyası 2 KB
M Dosyası 6 KB
MATLAB R2018b
Find Files
Compare
Print
New
Open
Save
Go To
Find
Insert
Comment
Indent
Breakpoints
FILE
NAVIGATE
EDIT
BREAKPOINTS
HknHtc
Desktop
fastslam_framework
Hkn
Current Fol...
LocateLandmarks.m
fastslam.m
% Read sensor readings, i.e. odometry and
data = read_data('sensor_data.dat');

References

1) FastSLAM, PhD thesis, M. Montemerlo, 2002
2) FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem, Montemerlo, Thrun, Koller and Wegbreit, 2002
3) Probabilistic Robotics, Thrun 2005


Q
&
A