For velocity model, it use this algorithm, here $x_{t-1} = (x, y, \theta)^T, x_t = (x', y', \theta')^T, u_t = (v, \omega)^T$

Algorithm motion_model_velocity(x_t, u_t, x_{t-1}):

$$\mu = \frac{1}{2} \frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$$
$$x^* = \frac{x + x'}{2} + \mu(y - y')$$
$$y^* = \frac{y + y'}{2} + \mu(x' - x)$$
$$r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2}$$
$$\Delta\theta = \operatorname{atan2}(y' - y^*, x' - x^*) - \operatorname{atan2}(y - y^*, x - x^*)$$
$$\hat{v} = \frac{\Delta\theta}{\Delta t}\, r^*$$
$$\hat{\omega} = \frac{\Delta\theta}{\Delta t}$$
$$\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}$$
$$\textit{return } \mathbf{prob}(v - \hat{v}, \alpha_1 v^2 + \alpha_2 \omega^2) \cdot \mathbf{prob}(\omega - \hat{\omega}, \alpha_3 v^2 + \alpha_4 \omega^2) \cdot \mathbf{prob}(\hat{\gamma}, \alpha_5 v^2 + \alpha_6 \omega^2)$$

The return value is the value of $p(x_{t+1}|x_t, u_t)$.

Detection probability:

Algorithm landmark_model_known_correspondence(f_t^i, c_t^i, x_t, m):

$$j = c_t^i$$
$$\hat{r} = \sqrt{(m_{j,x} - x)^2 + (m_{j,y} - y)^2}$$
$$\hat{\phi} = \operatorname{atan2}(m_{j,y} - y, m_{j,x} - x)$$
$$q = \mathbf{prob}(r_t^i - \hat{r}, \sigma_r) \cdot \mathbf{prob}(\phi_t^i - \hat{\phi}, \sigma_\phi) \cdot \mathbf{prob}(s_t^i - s_j, \sigma_s)$$
$$\textit{return } q$$

generate samples:

Algorithm sample_motion_model_velocity(u_t, x_{t-1}):

$$\hat{v} = v + \mathbf{sample}(\alpha_1 v^2 + \alpha_2 \omega^2)$$
$$\hat{\omega} = \omega + \mathbf{sample}(\alpha_3 v^2 + \alpha_4 \omega^2)$$
$$\hat{\gamma} = \mathbf{sample}(\alpha_5 v^2 + \alpha_6 \omega^2)$$
$$x' = x - \tfrac{\hat{v}}{\hat{\omega}} \sin\theta + \tfrac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega}\Delta t)$$
$$y' = y + \tfrac{\hat{v}}{\hat{\omega}} \cos\theta - \tfrac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega}\Delta t)$$
$$\theta' = \theta + \hat{\omega}\Delta t + \hat{\gamma}\Delta t$$
$$\textit{return } x_t = (x', y', \theta')^T$$

For odometry:

Algorithm sample_motion_model_odometry(u_t, x_{t-1}):

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

$$\hat{\delta}_{\text{rot1}} = \delta_{\text{rot1}} - \mathbf{sample}(\alpha_1 \delta_{\text{rot1}}^2 + \alpha_2 \delta_{\text{trans}}^2)$$
$$\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \mathbf{sample}(\alpha_3 \delta_{\text{trans}}^2 + \alpha_4\, \delta_{\text{rot1}}^2 + \alpha_4\, \delta_{\text{rot2}}^2)$$
$$\hat{\delta}_{\text{rot2}} = \delta_{\text{rot2}} - \mathbf{sample}(\alpha_1 \delta_{\text{rot2}}^2 + \alpha_2 \delta_{\text{trans}}^2)$$

$$x' = x + \hat{\delta}_{\text{trans}} \cos(\theta + \hat{\delta}_{\text{rot1}})$$
$$y' = y + \hat{\delta}_{\text{trans}} \sin(\theta + \hat{\delta}_{\text{rot1}})$$
$$\theta' = \theta + \hat{\delta}_{\text{rot1}} + \hat{\delta}_{\text{rot2}}$$

$$\textit{return } x_t = (x', y', \theta')^T$$

Similarity: Both of them are easy to implement.

Differenece: velocity reuire to calculate the inverse of the physical motion model; Odemetry side-steps the need for an inverse model.

Algorithm motion_model_odometry(x_t, u_t, x_{t-1}):

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

$$\hat{\delta}_{\text{rot1}} = \operatorname{atan2}(y' - y, x' - x) - \theta$$
$$\hat{\delta}_{\text{trans}} = \sqrt{(x - x')^2 + (y - y')^2}$$
$$\hat{\delta}_{\text{rot2}} = \theta' - \theta - \hat{\delta}_{\text{rot1}}$$

$$p_1 = \mathbf{prob}(\delta_{\text{rot1}} - \hat{\delta}_{\text{rot1}}, \alpha_1 \hat{\delta}_{\text{rot1}}^2 + \alpha_2 \hat{\delta}_{\text{trans}}^2)$$
$$p_2 = \mathbf{prob}(\delta_{\text{trans}} - \hat{\delta}_{\text{trans}}, \alpha_3 \hat{\delta}_{\text{trans}}^2 + \alpha_4 \hat{\delta}_{\text{rot1}}^2 + \alpha_4 \hat{\delta}_{\text{rot2}}^2)$$
$$p_3 = \mathbf{prob}(\delta_{\text{rot2}} - \hat{\delta}_{\text{rot2}}, \alpha_1 \hat{\delta}_{\text{rot2}}^2 + \alpha_2 \hat{\delta}_{\text{trans}}^2)$$

Here, control $u_t, = (\bar{x}_{t-1}, \bar{x}_t)^T, \bar{x}_{t-1} = (\bar{x}, \bar{y}, \bar{\theta})^T, u_t = (v, \omega)^T$.

the value of $p(x_{t+1}|x_t, u_t)$ is p1 * p2 * p3.

Similarity: for both motion models, we presented two types of implementations, one in which thr $p(x_t|u_t, x_{t-1})$ is calculated in closed form, and one that enables us to generate samples from $p(x_t|u_t, x_{t-1})$.

Difference: 1. velocity model add a third moise paremeter, expressed as a noisy "final votation".

2. Odemetry: the readings are technically not controls, use them just like controls.

Algorithm sample_landmark_model_known_correspondence(f_t^i, c_t^i, m):

$$j = c_t^i$$
$$\hat{\gamma} = \operatorname{rand}(0, 2\pi)$$
$$\hat{r} = r_t^i + \mathbf{sample}(\sigma_r)$$
$$\hat{\phi} = \phi_t^i + \mathbf{sample}(\sigma_\phi)$$
$$x = m_{j,x} + \hat{r}\cos\hat{\gamma}$$
$$y = m_{j,y} + \hat{r}\sin\hat{\gamma}$$
$$\theta = \hat{\gamma} - \pi - \hat{\phi}$$
$$\textit{return } (x \;\; y \;\; \theta)^T$$

Algorithm beam_range_finder_model(z_t, x_t, m):

$$q = 1$$
for $k = 1$ *to* K *do*

compute z_t^{k*} *for the measurement* z_t^k *using ray casting*
$$p = z_{\text{hit}} \cdot p_{\text{hit}}(z_t^k \mid x_t, m) + z_{\text{short}} \cdot p_{\text{short}}(z_t^k \mid x_t, m) + z_{\max} \cdot p_{\max}(z_t^k \mid x_t, m) + z_{\text{rand}} \cdot p_{\text{rand}}(z_t^k \mid x_t, m)$$
$$q = q \cdot p$$
return q

Scan-based sensors:

Algorithm likelihood_field_range_finder_model(z_t, x_t, m):

$$q = 1$$
for all k *do*

if $z_t^k \neq z_{\max}$
$$x_{z_t^k} = x + x_{k,\text{sens}} \cos\theta - y_{k,\text{sens}} \sin\theta + z_t^k \cos(\theta + \theta_{k,\text{sens}})$$
$$y_{z_t^k} = y + y_{k,\text{sens}} \cos\theta + x_{k,\text{sens}} \sin\theta + z_t^k \sin(\theta + \theta_{k,\text{sens}})$$
$$dist = \min_{x',y'} \left\{ \sqrt{(x_{z_t^k} - x')^2 + (y_{z_t^k} - y')^2} \,\middle|\, \langle x', y' \rangle \text{ occupied in } m \right\}$$
$$q = q \cdot \left(z_{\text{hit}} \cdot \mathbf{prob}(dist, \sigma_{\text{hit}}) + \frac{z_{\text{random}}}{z_{\max}} \right)$$
return q

Bean-bases sensors suffers two major drawbacks. (1) lack of smoothness. (2) computational involved. Scan-based sensors overcomes these limitations. It does not compute a conditional probability relative to any meaningful generative model of the physics of sensors.

1. Algorithm **Kalman_filter**(μ_{t-1}, Σ_{t-1}, u_t, z_t):
2. Prediction:
3. $\bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$
5. Correction:
6. $K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
9. Return μ_t, Σ_t

1. **Extended_Kalman_filter**(μ_{t-1}, Σ_{t-1}, u_t, z_t):
2. Prediction:
3. $\bar{\mu}_t = g(u_t, \mu_{t-1}) \longleftarrow \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t$
4. $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + Q_t \longleftarrow \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + Q_t$
5. Correction:
6. $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + R_t)^{-1} \longleftarrow K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + R_t)^{-1}$
7. $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t)) \longleftarrow \mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t)$
8. $\Sigma_t = (I - K_t H_t)\bar{\Sigma}_t \longleftarrow \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t$
9. Return μ_t, Σ_t

$$H_t = \frac{\partial h(\bar{\mu}_t)}{\partial x_t} \qquad G_t = \frac{\partial g(u_t, \mu_{t-1})}{\partial x_{t-1}}$$

Bayes Filter

z = observation
u = action
x = state

$$\boxed{Bel(x_t)} = P(x_t \mid u_1, z_1, \ldots, u_t, z_t)$$

Bayes $= \eta \; P(z_t \mid x_t, u_1, z_1, \ldots, u_t)\, P(x_t \mid u_1, z_1, \ldots, u_t)$

Markov $= \eta \; P(z_t \mid x_t)\, P(x_t \mid u_1, z_1, \ldots, u_t)$

Total prob. $= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_1, z_1, \ldots, u_t, x_{t-1})\, P(x_{t-1} \mid u_1, z_1, \ldots, u_t)\, dx_{t-1}$

Markov $= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1})\, P(x_{t-1} \mid u_1, z_1, \ldots, u_t)\, dx_{t-1}$

Markov $= \eta \, P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1})\, P(x_{t-1} \mid u_1, z_1, \ldots, z_{t-1})\, dx_{t-1}$

$$\boxed{= \eta \; P(z_t \mid x_t) \int P(x_t \mid u_t, x_{t-1})\, Bel(x_{t-1})\, dx_{t-1}}$$

Resampling Algorithm

Algorithm **systematic_resampling**(S,n):

$S' = \emptyset, c_1 = w^1$

For $i = 2 \ldots n$ — *Generate cdf*

$c_i = c_{i-1} + w^i$

$u_1 \sim U[0, n^{-1}],\ i = 1$ — *Initialize threshold*

For $j = 1 \ldots n$ — *Draw samples …*

While ($u_j > c_i$) — *Skip until next threshold reached*

$i = i + 1$

$S' = S' \cup \{< x^i, n^{-1} >\}$ — *Insert*

$u_{j+1} = u_j + n^{-1}$ — *Increment threshold*

Return S'

Algorithm **particle_filter**(S_{t-1}, $u_{t-1}\, z_t$):

$S_t = \emptyset, \quad \eta = 0$

For $i = 1 \ldots n$ — *Generate new samples*

Sample index $j(i)$ from the discrete distribution given by w_{t-1}

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}

$w_t^i = p(z_t \mid x_t^i)$ — *Compute importance weight*

$\eta = \eta + w_t^i$ — *Update normalization factor*

$S_t = S_t \cup \{< x_t^i, w_t^i >\}$ — *Insert*

For $i = 1 \ldots n$

$w_t^i = w_t^i / \eta$ — *Normalize weights*

Algorithm landmark_detection_model(z,x,m):

$z = \langle i, d, \alpha \rangle, x = \langle x, y, \theta \rangle$

$\hat{d} = \sqrt{(m_x(i) - x)^2 + (m_y(i) - y)^2}$

$\hat{a} = \operatorname{atan2}(m_y(i) - y, m_x(i) - x) - \theta$

$p_{\det} = \operatorname{prob}(\hat{d} - d, \varepsilon_d) \cdot \operatorname{prob}(\hat{\alpha} - \alpha, \varepsilon_\alpha)$

Return $z_{\det} p_{\det} + z_{\text{fp}} P_{\text{uniform}}(z \mid x, m)$

Occupancy Update Rule

- Recursive rule

$$\frac{p(m_i \mid z_{1:t}, x_{1:t})}{1 - p(m_i \mid z_{1:t}, x_{1:t})} = \underbrace{\frac{p(m_i \mid z_t, x_t)}{1 - p(m_i \mid z_t, x_t)}}_{\text{uses } z_t} \underbrace{\frac{p(m_i \mid z_{1:t-1}, x_{1:t-1})}{1 - p(m_i \mid z_{1:t-1}, x_{1:t-1})}}_{\text{recursive term}} \underbrace{\frac{1 - p(m_i)}{p(m_i)}}_{\text{prior}}$$

- The product turns into a sum

$$l(m_i \mid z_{1:t}, x_{1:t}) = \underbrace{l(m_i \mid z_t, x_t)}_{\text{inverse sensor model}} + \underbrace{l(m_i \mid z_{1:t-1}, x_{1:t-1})}_{\text{recursive term}} - \underbrace{l(m_i)}_{\text{prior}}$$

Refection Map

We assume that all cells m_j are independent:

$$m^* = \arg\max_m \left(\sum_{j=1}^{J} \alpha_j \ln m_j + \beta_j \ln(1 - m_j) \right)$$

If we set

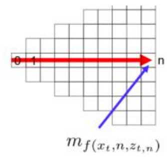
$$\frac{\partial m}{\partial m_j} = \frac{\alpha_j}{m_j} - \frac{\beta_j}{1 - m_j} = 0$$

we obtain

$$m_j = \frac{\alpha_j}{\alpha_j + \beta_j}$$

Computing the most likely map amounts to counting how often a cell has reflected a measurement and how often it was intercepted.

1. pose at time t: x_t
2. beam n of scan t: $z_{t,n}$
3. maximum range reading: $\varsigma_{t,n} = 1$
4. beam reflected by an object: $\varsigma_{t,n} = 0$



max range: "first $z_{t,n}$-1 cells covered by the beam must be free"

$$p(z_{t,n} \mid x_t, m) = \begin{cases} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t,n,k)}) & \text{if } \zeta_{t,n} = 1 \\ m_{f(x_t,n,z_{t,n})} \prod_{k=0}^{z_{t,n}-1} (1 - m_{f(x_t,n,k)}) & \zeta_{t,n} = 0 \end{cases}$$

otherwise: "last cell reflected beam, all others free"

Reflection v.s. Occupancy Maps

Reflection: $p(m \mid \rho) = \rho^m (1 - \rho)^{1-m}$

$p(N_{hit}, N_{mis} \mid \rho) = \rho^{N_{hit}} (1 - \rho)^{N_{mis}}$ ⇒ $\rho_{ML} = \frac{N_{hit}}{N_{hit} + N_{mis}}$

inverse sensor model: (Occupancy) $p(o \mid \alpha, z = 1{:}\, hit) = \alpha^o (1 - \alpha)^{1-o}$

$p(o \mid \beta, z = 0{:}\, mis) = \beta^o (1 - \beta)^{1-o}$

Occupancy: $p(o = 1 \mid \alpha, \beta, N_{hit}, N_{mis}) = \alpha^{N_{hit}} \beta^{N_{mis}}$

$p(o = 0 \mid \alpha, \beta, N_{hit}, N_{mis}) = (1 - \alpha)^{N_{hit}} (1 - \beta)^{N_{mis}}$

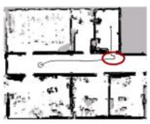
Occupancy ratio > 1   occupancy grid value will be 1

Dynamic Window Approach

- Reacts quickly.
- Low CPU power requirements.
- Guides a robot on a collision free path.
- Successfully used in a lot of real-world scenarios.
- Resulting trajectories sometimes sub-optimal.
- Local minima might prevent the robot from reaching the goal location.

Problems of DWAs

- Typical problem in a real world situation:



- Robot does not slow down early enough to enter the doorway.

Reachable Velocities

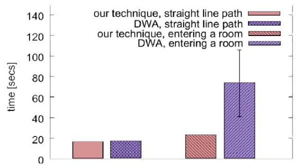
- Speeds are admissible if

$$V_d = \{(v, \omega) \mid v \in [v - a_{trans} t, v + a_{trans} t] \wedge \omega \in [\omega - a_{rot} t, \omega + a_{rot} t]\}$$

5D planning

1. Update (static) grid map based on sensory input.
2. Use A* to find a trajectory in the <x,y>-space using the updated grid map.
3. Determine a restricted 5d-configuration space based on step 2.
4. Find a trajectory by planning in the restricted <x,y,θ,v,ω>-space.

Comparison to the DWA (II)



The presented approach results in significantly faster motion when driving through narrow passages!

Markov Decision Process Setup

- Given

States x, Actions u

Transition probabilities $p(x' \mid u, x)$

Reward function $r(x, u)$

- Wanted

Policy $\pi(x)$ that maximizes the future expected reward

Typical Assumptions in A*

- A robot is assumed to be localized.
- Often a robot has to compute a path based on an occupancy grid.
- Often the correct motion commands are executed (but no perfect map).

Problems

- What if the robot is slightly delocalized?
- Moving on the shortest path guides often the robot on a trajectory close to obstacles.
- Trajectory aligned to the grid structure.

A* in Convolved Maps

- The costs are a product of path length and occupancy probability of the cells.
- Cells with higher probability (e.g., caused by convolution) are avoided by the robot.
- Thus, it keeps distance to obstacles.
- This technique is fast and quite reliable.

EKF_SLAM_Prediction($\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t, R_t$):

$$F_x = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 \end{pmatrix}$$

$$\bar{\mu}_t = \mu_{t-1} + F_x^T \begin{pmatrix} -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} - \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ \omega_t \Delta t \end{pmatrix}$$

$$G_t = I + F_x^T \begin{pmatrix} 0 & 0 & -\frac{v_t}{\omega_t} \cos \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \cos(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & -\frac{v_t}{\omega_t} \sin \mu_{t-1,\theta} + \frac{v_t}{\omega_t} \sin(\mu_{t-1,\theta} + \omega_t \Delta t) \\ 0 & 0 & 0 \end{pmatrix} F_x$$

$$\bar{\Sigma}_t = G_t\, \Sigma_{t-1}\, G_t^T + \underbrace{F_x^T\, R_t^x\, F_x}_{R_t}$$

EKF_SLAM_Correction

$$Q_t = \begin{pmatrix} \sigma_r^2 & 0 \\ 0 & \sigma_\phi^2 \end{pmatrix}$$

for all observed features $z_t^i = (r_t^i, \phi_t^i)^T$ do

$j = c_t^i$

if landmark j never seen before

$$\begin{pmatrix} \bar{\mu}_{j,x} \\ \bar{\mu}_{j,y} \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{t,x} \\ \bar{\mu}_{t,y} \end{pmatrix} + \begin{pmatrix} r_t^i \cos(\phi_t^i + \bar{\mu}_{t,\theta}) \\ r_t^i \sin(\phi_t^i + \bar{\mu}_{t,\theta}) \end{pmatrix}$$

endif

$$\delta = \begin{pmatrix} \delta_x \\ \delta_y \end{pmatrix} = \begin{pmatrix} \bar{\mu}_{j,x} - \bar{\mu}_{t,x} \\ \bar{\mu}_{j,y} - \bar{\mu}_{t,y} \end{pmatrix}$$

$$q = \delta^T \delta$$

$$\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \operatorname{atan2}(\delta_y, \delta_x) - \bar{\mu}_{t,\theta} \end{pmatrix}$$

$$F_{x,j} = \begin{pmatrix} 1 & 0 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 1 & 0 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 1 & 0 \cdots 0 & 0 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & 0 \cdots 0 & 1 & 0 & 0 \cdots 0 \\ 0 & 0 & 0 & \underbrace{0 \cdots 0}_{2j-2} & 0 & 1 & \underbrace{0 \cdots 0}_{2N-2j} \end{pmatrix}$$

$$H_t^i = \frac{1}{q} \begin{pmatrix} -\sqrt{q}\delta_x & -\sqrt{q}\delta_y & 0 & +\sqrt{q}\delta_x & \sqrt{q}\delta_y \\ \delta_y & -\delta_x & -q & -\delta_y & +\delta_x \end{pmatrix} F_{x,j}$$

$$K_t^i = \bar{\Sigma}_t\, H_t^{iT} (H_t^i\, \bar{\Sigma}_t\, H_t^{iT} + Q_t)^{-1}$$

$$\bar{\mu}_t = \bar{\mu}_t + K_t^i (z_t^i - \hat{z}_t^i)$$

$$\bar{\Sigma}_t = (I - K_t^i\, H_t^i)\, \bar{\Sigma}_t$$

endfor

$\mu_t = \bar{\mu}_t$

$\Sigma_t = \bar{\Sigma}_t$

return μ_t, Σ_t