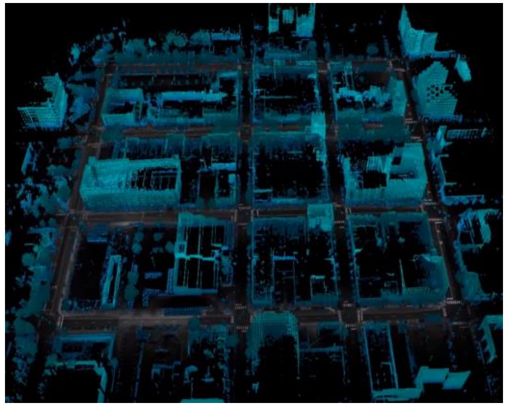
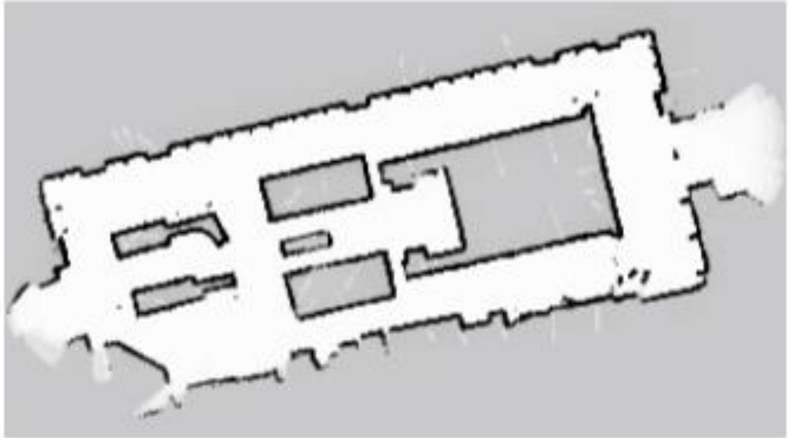
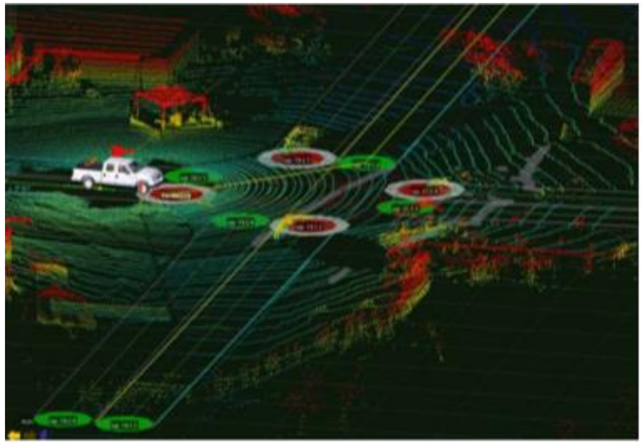
BYU Electrical & Computer Engineering





LAB4: EKF-UKF LOCALIZATION NOTES

ECEN 633: Robotic Localization and Mapping

Odometry Motion Model – Prob. Robotics

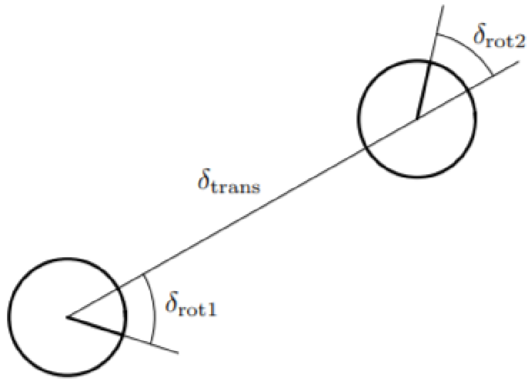


Figure 5.7 Odometry model: The robot motion in the time interval $(t-1, t]$ is approximated by a rotation $\delta_{\text{rot}1}$, followed by a translation δ_{trans} and a second rotation $\delta_{\text{rot}2}$. The turns and translations are noisy.

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

$$\delta_{\text{rot}1} = \text{atan2}(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta}$$
$$\delta_{\text{trans}} = \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2}$$
$$\delta_{\text{rot}2} = \bar{\theta}' - \bar{\theta} - \delta_{\text{rot}1}$$

$$\hat{\delta}_{\text{rot}1} = \delta_{\text{rot}1} - \textbf{sample}(\alpha_1 \delta_{\text{rot}1}^2 + \alpha_2 \delta_{\text{trans}}^2)$$
$$\hat{\delta}_{\text{trans}} = \delta_{\text{trans}} - \textbf{sample}(\alpha_3 \delta_{\text{trans}}^2 + \alpha_4\ \delta_{\text{rot}1}^2 + \alpha_4\ \delta_{\text{rot}2}^2)$$
$$\hat{\delta}_{\text{rot}2} = \delta_{\text{rot}2} - \textbf{sample}(\alpha_1 \delta_{\text{rot}2}^2 + \alpha_2 \delta_{\text{trans}}^2)$$

$$x' = x + \hat{\delta}_{\text{trans}}\ \cos(\theta + \hat{\delta}_{\text{rot}1})$$
$$y' = y + \hat{\delta}_{\text{trans}}\ \sin(\theta + \hat{\delta}_{\text{rot}1})$$
$$\theta' = \theta + \hat{\delta}_{\text{rot}1} + \hat{\delta}_{\text{rot}2}$$

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

Table 5.6 Algorithm for sampling from $p(x_t \mid u_t, x_{t-1})$ based on odometry information. Here the pose at time t is represented by $x_{t-1} = (x\ y\ \theta)^T$. The control is a differentiable set of two pose estimates obtained by the robot's odometer, $u_t = (\bar{x}_{t-1}\ \bar{x}_t)^T$, with $\bar{x}_{t-1} = (\bar{x}\ \bar{y}\ \bar{\theta})$ and $\bar{x}_t = (\bar{x}'\ \bar{y}'\ \bar{\theta}')$.

Motion Model – MATLAB Code (prediction.m)

prediction.m

```
%--------------------------------------------------------
% predicts the new state given the current state and motion
% motion in form of [drot1,dtrans,drot2]
%--------------------------------------------------------
function state = prediction(state, motion)

state(3)=state(3)+motion(1);
state(1)=state(1)+motion(2)*cos(state(3));
state(2)=state(2)+motion(2)*sin(state(3));
state(3)=state(3)+motion(3);
state(3)=minimizedAngle(state(3));

```

EKF Motion Model – State Prediction

- Additive Gaussian Noise

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t) + \epsilon_t$$

$$\mathbf{x}_{t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$$
$$\mathbf{u}_t \text{ known}$$
$$\epsilon_t \sim \mathcal{N}(\mathbf{0}, R_t)$$

EKF Prediction:

$$\mu_t = g(\mu_{t-1}, \mathbf{u}_t)$$
$$\Sigma_t = G_t \Sigma_{t-1} G_t^\top + R_t$$

- Non-additive Gaussian Noise

$$\mathbf{x}_t = g(\mathbf{x}_{t-1}, \mathbf{u}_t)$$

$$\mathbf{x}_{t-1} \sim \mathcal{N}(\mu_{t-1}, \Sigma_{t-1})$$
$$\mathbf{u}_t \sim \mathcal{N}(\hat{\mathbf{u}}_t, R_t)$$

EKF Prediction:

?

Landmark Bearing Measurement Model

$$\hat{z}_t = \begin{pmatrix} \sqrt{(m_x - \bar{\mu}_{t,x})^2 + (m_y - \bar{\mu}_{t,y})^2} \\ \boxed{\operatorname{atan2}(m_y - \bar{\mu}_{t,y}, m_x - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta}} \end{pmatrix}$$

Range

Bearing

Landmark Bearing Meas. Model – (observation.m)

observation.m

```
%-------------------------------------------------------
% returns the observation of the specified marker given the current state
%-------------------------------------------------------
function obs = observation(state, id)
global FIELDINFO;

% Compute expected observation.
dx = FIELDINFO.MARKER_X_POS(int32(id)) - state(1);
dy = FIELDINFO.MARKER_Y_POS(int32(id)) - state(2);
dist = sqrt(dx^2 + dy^2);

obs = [ minimizedAngle(atan2(dy, dx) - state(3));
        id ];
```