Robot Localization Using Kalman Filter

Yuri Balasanov

January, 2020

CT

Contents

I The First Part **1**

1 Key Concepts and Notations **3**

1.1 Environment state 3

1.2 Perception 4

1.3 Control 4

1.4 Distributions 4

1.5 Belief Distributions 5

2 Bayes Filters **7**

2.1 General Case 7

2.2 Linear Gaussian Filters 9

3 Motion Model **12**

3.1 Kinematic configuration 12

3.2 Velocity model 13

3.2.1 Ideal motion 15

3.2.2 Real motion 16

3.3 Odometry model 18

4 Localization Problem **21**

CT

Preface

This description is extracted from Probabilistic Robotics by Sebastian Thrun, Wolfram Burgard, Dieter Fox, MIT Press, 2006.

Part I

The First Part

CN # Chapter 1

CT # Key Concepts and Notations

A ## 1.1 Environment state

In 2D robot **pose** is its location and orientation $\langle x, y, \theta \rangle$, where x, y are location coordinates on the plane and θ is the orientation of the robot. For localization problem robot's pose is is the **state** of the environment

$$x_t = \langle x_t, y_t, \theta_t \rangle .$$

Note that notation x_t may be used in two different meanings: as state of the environment (pose) or as x-coordinate of robot's pose.

A

1.2 Perception

Perceptoion is the process by which the robot uses its sensors to obtain information about the state of the environment. The result of such perceptual interaction is called **measurement** and denoted z_t. Sequence of measurements between times t_{t_1} and t_2 is

$$z_{t_1:t_2} = z_{t_1}, z_{t_1+1}, z_{t_1+2}, \ldots, z_{t_2}$$

A

1.3 Control

In mobile robotics typical example of control data is velocity of robot. Alternative - odometers, or sensors that measure the revolution of robot's wheels. Even though, odometers are sensors, odometry is usually treated as control data.

Control variable u_t corresponds to change of state during time interval $(t-1, t]$.

Sequence of control data is $u_{t_1:t_2} = u_{t_1}, u_{t_1+1}, u_{t_1+2}, \ldots, u_{t_2}$.

A

1.4 Distributions

The key dprobability distributions are:

1. State transition probability distribution $p(x_t | x_{t-1}, u_t)$ specifying how environmental state evolves in time as a function of robot's control u_t

2. Measurement probability distribution $p(z_t|x_t)$ specifying how measurements z_t are generated from the environment state x_t.

The model described by these two distributions (including Markov condition) is known as Hidden Markov Model (HMM) of dynamic Bayes network (DBN).

A

1.5 Belief Distributions

A belief reflects the robot's internal knowledge about the state of the environment. State cannot be measured directly: pose might be

$$x_t = \langle 14.12, 12.7, 45^\circ \rangle$$

in some global coordinate system, but it is not known to the robot. Instead the robot has the belief based on perception.

Belief is represented through conditional distributions that assigns a probability (or density) value to each possible hypothesis about the true state.

Belief distributions are posterior distributions over state variables, conditional on the available data:

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}).$$

Note that $bel(x_t)$ is taken after the observation z_t.

Belief realized before the observation z_t, but after the control u_t is applied

will be annotated

$$BEL\left(x_t\right) = p\left(x_t \middle| z_{1:t-1}, u_{1:t}\right).$$

This probability distribution is referred to as **prediction distribution**.

Calculation of $bel\left(x_t\right)$ from $BEL\left(x_t\right)$ is called **correction** or **measurement update**.

CN # Chapter 2

CT # Bayes Filters

A ## 2.1 General Case

Bayes filter algorithm calculates beliefs distribution $bel(x_t)$ from measurement and control data. The algorithm is recursive: it calculates $bel(x_t)$ from $bel(x_{t-1})$.

The algorithm has 2 main steps:

1. Calculate current prior belief $BEL(x_t)$ from the previous posterior belief $bel(x_{t-1})$ using theorem of total probability.

   Theorem 1 *Theorem of total probability calculates unconditional probability distribution as*

$$p(x) = \sum_y p(x|y)\, p(y)$$

in discrete case and as

$$p(x) = \int p(x|y)\, p(y)\, dy$$

in continuous case

Applying this theorem prior belief at time t is calculated as

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

2. Calculate posterior belief $bel(x_t)$ through measurement update using as prior distribution and observation z_t as data. The formula for this step is Bayes rule

Theorem 2 *Bayes rule for discrete case is*

$$p(x|y) = \frac{p(y|x)\, p(x)}{\sum_{x'} p(y|x')\, p(x')}$$

and for continuous case

$$p(x|y) = \frac{p(y|x)\, p(x)}{\int p(y|x')\, p(x')\, dx'}$$

Since denominator in Bayes rule does not depend on x it is just a normalizer (constant with respect to x) and can be replaced with constant η as

$$p(x|y) = \eta p(y|x)\, p(x).$$

Then posterior belief is

$$bel(x_t) = \eta p(z_t|x_t)\, BEL(x_t),$$

where $p(z_t|x_t)$ is the measurement probability distribution.

A

2.2 Linear Gaussian Filters

In Gaussian Filters beliefs are represented by multivariate Gaussian distributions.

The simplest Gaussian Filter is **Kalman Filter** (KF) or **Linear Gaussian Filter.**

KF implements belief computation for continuous states.

Since KF is based on Gaussian distributions it represents belief by its vector of means μ_t and covariance matrix Σ_t.

In order for the belief distributions to be Gaussian, in addition to Markov property the following 3 conditions must hold:

1. State transition must be linear function with added Gaussian noise

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t,$$

where x_t is a column-vector of size n, u_t is a column-vector of size m, A_t is a square matrix of shape $n \times n$ and B_t is a matrix of shape $n \times m$. The noise ε_t is a column-vector of size n which has Gaussian distribution with zero mean and covariance matrix R_t.

From the state transition model it follows that state transition probability distribution $p\left(x_t \middle| x_{t-1}, u_t\right)$ is Gaussian with mean $A_t x_{t-1} + B_t u_t$ and covariance matrix R_t.

2. Measurement model is also linear:

$$z_t = C_t x_t + \delta_t,$$

where z_t is the measurement vector of size k, matrix C_t has shape $k \times n$ and δ_t is the vector of measurement noise distributed as Gaussian with mean 0 and covariance matrix Q_t.

Then the measurement probability distribution $p\left(z_t \middle| x_t\right)$ is Gaussian with mean $C_t x_t$ and covariance matrix Q_t.

3. The initial belief $bel\left(x_0\right)$ must be distributed as Gaussian with mean μ_0 and covariance matrix Σ_0.

Under these 3 assumptions the posterior belief is guaranteed to be Gaussian.

CHAPTER 2 — MANUSCRIPT

The KF algorithm is shown below:

$$
\begin{aligned}
\text{Kalman\_Filter}\left(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t\right) \quad &: \\
\bar{\mu}_t &= A_t \mu_{t-1} + B_t u_t \\
\bar{\Sigma}_t &= A_t \Sigma_{t-1} A_t^T + R_t \\
K_t &= \bar{\Sigma}_t C_t^T \left(C_t \bar{\Sigma}_t C_t^T + Q_t\right)^{-1} \\
\mu_t &= \bar{\mu}_t + K_t \left(z_t - C_t \bar{\mu}_t\right) \\
\Sigma_t &= \left(I - K_t C_t\right) \bar{\Sigma}_t \\
\text{return} \quad &: \quad \mu_t, \Sigma_t
\end{aligned}
$$

In the algorithm above $\bar{\mu}_t$ and $\bar{\Sigma}_t$ are the parameters of Gaussian distribution $BEL\left(x_t\right)$.

The variable K_t is called **Kalman gain**. Kalman gain specifies how much the observation z_t should be allowed to change the expected observation $C_t \bar{\mu}_t$.

The difference $\left(z_t - C_t \bar{\mu}_t\right)$ is called **innovation**. It shoes the difference between the actual measurement z_t and predicted (expected) measurement $C_t \bar{\mu}_t$.

CN # Chapter 3

CT # Motion Model

A ## 3.1 Kinematic configuration

As stated earlier state of the mobile robot on the plane is its pose

$$\begin{pmatrix} x \\ y \\ \theta \end{pmatrix}.$$

Orientation θ is also calle **bearing** or **heading direction**. When $\theta = 0$ the robot points in the direction of x axis and when $\theta = 0.5\pi$ robot points in the direction of y axis. Pose without the last element orientation is called **location**.

The control u_t can be either velocity command given to the robot's motors or odometry information (distance traveled, angle turned).

A

3.2 Velocity model

Under volocity model control is a vector of **translationsl velocity** v_t and **rotational velocity** w_t

$$u_t = \begin{pmatrix} v_t \\ \\ w_t \end{pmatrix},$$

where positive rotational velocity result in left turn (counterclock rotation) and positive trslational velocity results in moving forward.

Velocity model returns $p\left(x_t \middle| x_{t-1}, u_t\right)$, where $x_t = (x', y', \theta')^T$, $x_{t-1} = (x, y, \theta)^T$,

$u_t = (v, w)^T$:

$$
\begin{aligned}
Motion\_Velocity\_Model\,(x_t, u_t, x_{t-1}) \quad : & \\
\mu &= \frac{1}{2}\frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta} \\
x^* &= \frac{x + x'}{2} + \mu\,(y - y') \\
y^* &= \frac{y + y'}{2} + \mu\,(x' - x) \\
r^* &= \sqrt{(x - x^*)^2 + (y - y^*)^2} \\
\Delta\theta &= \arctan 2\,(y' - y^*, x' - x^*) \\
&\quad - \arctan 2\,(y - y^*, x - x^*) \\
\hat{v} &= \frac{\Delta\theta}{\Delta t} r^* \\
\hat{w} &= \frac{\Delta\theta}{\Delta t} \\
\hat{\gamma} &= \frac{\theta' - \theta}{\Delta t} - \hat{w} \\
&\quad \text{return } prob\left(v - \hat{v}, \alpha_1 v^2 + \alpha_2 w^2\right) \cdot \\
&\quad prob\left(w - \hat{w}, \alpha_3 v^2 + \alpha_4 w^2\right) \cdot \\
&\quad prob\left(\hat{\gamma}, \alpha_5 v^2 + \alpha_6 w^2\right),
\end{aligned}
$$

where $prob\,(a, b^2)$ returns either normal density with mean a and variance b^2

$$\frac{1}{\sqrt{2\pi b^2}} \exp\left(-\frac{1}{2}\frac{a^2}{b^2}\right)$$

or triangular density with the same moments

$$\max\left\{0, \frac{1}{\sqrt{6}b} - \frac{|a|}{6b^2}\right\}.$$

This function models the motion error.

B

3.2.1 Ideal motion

If the robot is ideal noise-free and control $u_t = (v, w)^T$ is kept constant for the period Δt from $t-1$ until t, the robot moves on a circle with radius $r = \left|\frac{v}{w}\right|$. This follows from the general relationship between translational and rotational velocities: $v = wr$. In particular, if $w = 0$ the robot does not turn at all and moves on a straight line corresponding to a circle with infinite radius.

The center of the circle on which the robot is moving is at $(x_c, y_c) = \left(x - \frac{v}{w}\sin\theta, y + \frac{v}{w}\cos\theta\right)$, where $x_{t-1} = (x, y, \theta)^T$.

After Δt time of motion the new pose $x_t = (x', y', \theta')$ is

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x_c + \frac{v}{w}\sin(\theta + w\Delta t) \\ y_c + \frac{v}{w}\cos(\theta + w\Delta t) \\ \theta + w\Delta t \end{pmatrix}$$

$$= \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{v}{w}\sin\theta + \frac{v}{w}\sin(\theta + w\Delta t) \\ \frac{v}{w}\cos\theta - \frac{v}{w}\cos(\theta + w\Delta t) \\ w\Delta t \end{pmatrix}.$$

Time interval Δt is usually small to make assumption about constant velocities more realistic.

B

3.2.2 Real motion

Now assume that the actual velocities are

$$\begin{pmatrix} \hat{v} \\ \hat{w} \end{pmatrix} = \begin{pmatrix} v \\ w \end{pmatrix} + \begin{pmatrix} \varepsilon_{\alpha_1 v^2 + \alpha_2 w^2} \\ \varepsilon_{\alpha_3 v^2 + \alpha_4 w^2} \end{pmatrix},$$

where ε_{b^2} is a zero-mean random variable with variance b^2. The standard deviation of the error is proportional to the required velocity. The parameters α_1,α_2,α_3,α_4 are robot-specific error parameters. Two common choices for distributions of these parameters are Gaussian and triangular.

Then a more realistic motion model is

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{w}} \sin\theta + \frac{\hat{v}}{\hat{w}} \sin\left(\theta + \hat{w}\Delta t\right) \\ \frac{\hat{v}}{\hat{w}} \cos\theta - \frac{\hat{v}}{\hat{w}} \cos\left(\theta + \hat{w}\Delta t\right) \\ \hat{w}\Delta t \end{pmatrix}.$$

However this model needs one more adjustment: the radius of the motion gets distorted by noise too.

The fix is the assumption that the robot makes additional rotation $\hat{\gamma}$ when it arrives at final pose. Thus instead of $\theta' = \theta + \hat{w}\Delta t$ the final orientation is

$$\theta' = \theta + \hat{w}\Delta t + \hat{\gamma}\Delta t,$$

where

$$\hat{\gamma} = \varepsilon_{\alpha_5 v^2 + \alpha_6 w^2}.$$

And the final model is

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{w}} \sin\theta + \frac{\hat{v}}{\hat{w}} \sin\left(\theta + \hat{w}\Delta t\right) \\ \frac{\hat{v}}{\hat{w}} \cos\theta - \frac{\hat{v}}{\hat{w}} \cos\left(\theta + \hat{w}\Delta t\right) \\ \hat{w}\Delta t + \hat{\gamma}\Delta t \end{pmatrix}$$

The final line of the algorithm calculates the distribution $p\left(x_t \middle| x_{t-1}, u_t\right)$ as the product of densities of the 3 errors, which are independent.

A

3.3 Odometry model

Odometry model usually is more accurate than velocity model, but information is only available after the moving period. This makes the odometry model preferrable for localization and planning, but not useful for planning.

Due to slippage and drift there is no fixed transformation of coordinates between the internal odometry and the physical world global coordinates.

The odometry model uses relative motion measured by the robot's internal odometry.

Between $t-1$ and t the robot moves from x_{t-1} to x_t, while the odometry reports the move from $\bar{x}_{t-1} = \left(\bar{x}, \bar{y}, \bar{\theta}\right)^T$ to $\bar{x}_t = \left(\bar{x}', \bar{y}', \bar{\theta}'\right)^T$.

Then control is

$$u_t = \begin{pmatrix} \bar{x}_{t-1} \\ \\ \bar{x}_t \end{pmatrix}.$$

To extract relative odometry the control is transformed into sequence of 3

steps: rotation δ_{rot1}, straight motion δ_{trans}, then another rotation δ_{rot2}.

$Motion\_Odometry\_Model\,(x_t, u_t, x_{t-1})$:

$$
\begin{aligned}
\delta_{rot1} &= \arctan 2\,(\bar{y}' - \bar{y}, \bar{x}' - \bar{x}) - \bar{\theta} \\
\delta_{trans} &= \sqrt{(\bar{x} - \bar{x}')^2 + (\bar{y} - \bar{y}')^2} \\
\delta_{rot2} &= \bar{\theta}' - \bar{\theta} - \delta_{rot1} \\
\hat{\delta}_{rot1} &= \arctan 2\,(y' - y, x' - x) - \theta \\
\hat{\delta}_{trans} &= \sqrt{(x - x')^2 + (y - y')^2} \\
\hat{\delta}_{rot2} &= \theta' - \theta - \hat{\delta}_{rot1} \\
p_1 &= prob\left(\delta_{rot1} - \hat{\delta}_{rot1}, \alpha_1 \hat{\delta}_{rot1}^2 + \alpha_2 \hat{\delta}_{trans}^2\right) \\
p_2 &= prob\left(\delta_{trans} - \hat{\delta}_{trans}, \alpha_3 \hat{\delta}_{trans}^2 + \alpha_4 \hat{\delta}_{rot1}^2 + \alpha_4 \hat{\delta}_{rot2}^2\right) \\
p_3 &= prob\left(\delta_{rot2} - \hat{\delta}_{rot2}, \alpha_1 \hat{\delta}_{rot2}^2 + \alpha_2 \hat{\delta}_{trans}^2\right) \\
&\text{return } p_1 \cdot p_2 \cdot p_3
\end{aligned}
$$

First, the algorithm recovers relative information from control: $(\delta_{rot1}, \delta_{trans}, \delta_{rot2})^T$, Then relative motion information $\left(\hat{\delta}_{rot1}, \hat{\delta}_{trans}, \hat{\delta}_{rot2}\right)^T$ is recovered from the previous pose $x_{t-1} = (x, y, \theta)$ and the hypothetical new pose $x_t = (x', y', \theta')$.

Note that all angular differences must be in $[-\pi, \pi]$, so they must be truncated

accordingly.

As before parameters α_1,α_2,α_3,α_4 are robot-specific parameters.

CN # Chapter 4

CT # Localization Problem