移動ロボットの確率的な自己位置推定と地図構築

概要

参考文献:『移動ロボットのための確率的な自己位置推定と地図構築』（友納昌弘, ロボット学会誌, vol. 29, No. 5, pp. 423-426, 2011）など

- 自己位置推定と地図構築は、移動ロボットにとってもっとも基本的な機能で、表裏一体をなす
  - 不確実性のあるセンサデータ ⇒ 確率に基づく手法
  - ベイズフィルタによる自己位置推定・地図構築が主流
- 記法・仕様
  - 車輪型移動ロボット、センサはロボットに搭載、GPSなし
  - ロボットの位置: $r_t = (x, y, \theta)^T$ は3自由度で表現
    ここで、x, y は地図上の座標値、θ はロボットの向き
  - 地図 m はランドマーク m_j の集合
    ここで、m_jの位置は二次元の点$(m_{j,x}, m_{j,y})^T$で表現

自己位置推定(1)

- 地図が与えられている場合、自分の地図上の位置をセンサデータから推定
  - 自己位置推定の状態ベクトル: $r_t = (x_t, y_t, \theta_t)^T$
  - 位置追跡と大域的自己位置推定(ここでは前者)
- 位置追跡(position tracking)
  - 初期値を外部から与え、それを始点に現在の位置を推定
  - センサの時系列データ $z_{1:t} = z_1, z_2, \cdots, z_t$、ロボットの動作の時系列データ $u_{1:t} = u_1, u_2, \cdots, u_t$ ときのロボットの位置の確率密度 $p(r_t \mid z_{1:t}, u_{1:t}, m)$ を推定

$$p(r_t \mid z_{1:t}, u_{1:t}, m) = \alpha\, p(z_t \mid r_t, m) \qquad \text{——(4)}$$

$$\times \int p(r_t \mid r_{t-1}, u_t)\, p(r_{t-1} \mid z_{1:t-1}, u_{1:t-1}, m)\, dr_{t-1}$$

自己位置推定(2)

ロボット位置の確率密度

$$p(r_t \mid z_{1:t}, u_{1:t}, m) = \alpha\, p(z_t \mid r_t, m) \times \int p(r_t \mid r_{t-1}, u_t)\, p(r_{t-1} \mid z_{1:t-1}, u_{1:t-1}, m)\, dr_{t-1}$$

- ベイズ定理とマルコフ性を仮定して変形
- 漸化式：時刻 $t-1$ のロボット位置の確率密度関数がわかれば、時刻 t の関数を求めることができる
- 重要：状態ベクトル r_t は直接観測されず、観測可能なデータ $z_{1:t}$ と動作 $u_{1:t}$ から推定する

動作モデル $p(r_t \mid r_{t-1}, u_t)$ と計測モデル $p(z_t \mid r_t, m)$ がそれを担う

ベイズフィルタアルゴリズム参照

参考：ベイズフィルタ（1）

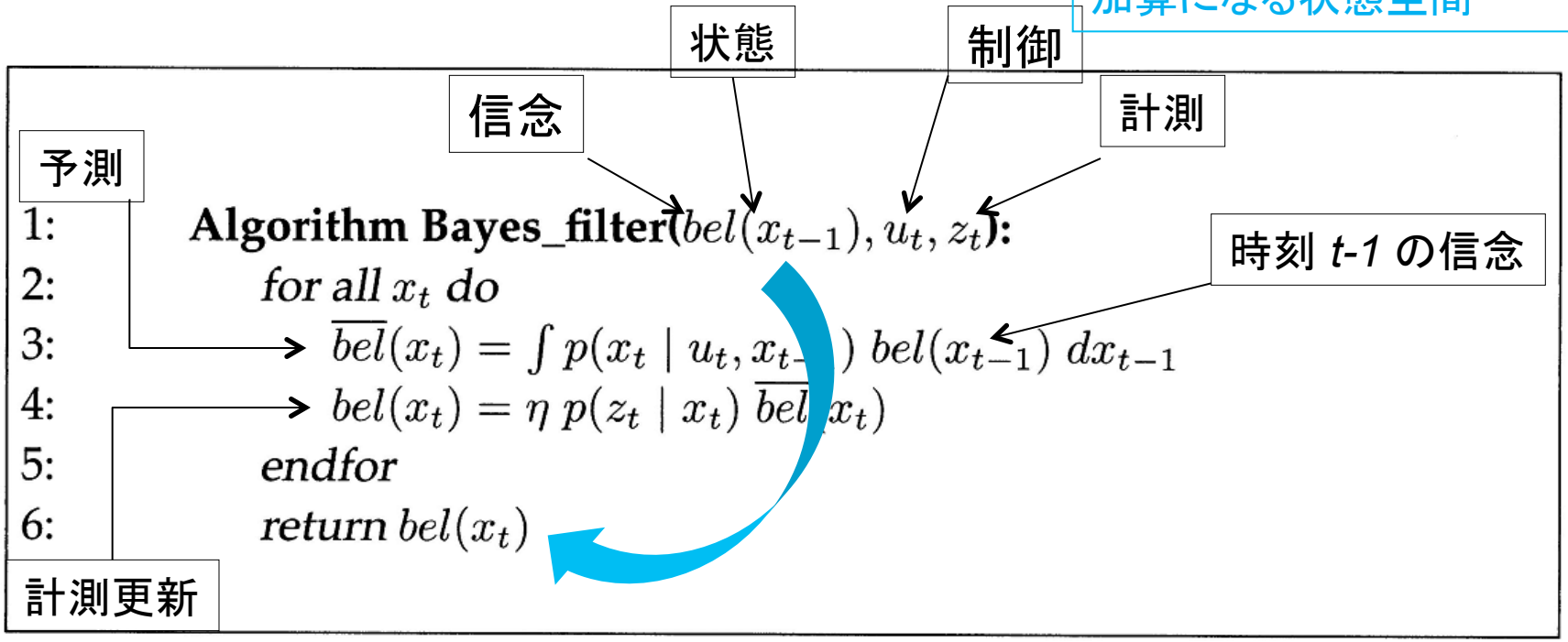
- ベイズフィルタアルゴリズム（BFA）
  - 信念を計算する一般的なアルゴリズム
  - 時刻 $t-1$ における信念 $bel(x_{t-1})$ から、時刻 t における信念 $bel(x_t)$ を計算

1: **Algorithm Bayes_filter**($bel(x_{t-1}), u_t, z_t$):
2: *for all* x_t *do*
3: $\overline{bel}(x_t) = \int p(x_t \mid u_t, x_{t-1})\ bel(x_{t-1})\ dx_{t-1}$
4: $bel(x_t) = \eta\ p(z_t \mid x_t)\ \overline{bel}(x_t)$
5: *endfor*
6: *return* $bel(x_t)$

参考：ベイズフィルタ（2）

- 3行目：制御更新、もしくは予測（cf. 全確率の定理）
- 4行目：計測更新、正規化変数η（積分しても1にならないので）
- 単純な推定問題にのみ適用可能

・3, 4行目が厳密に実行可能
・3行目の積分が有限回の加算になる状態空間



動作モデル

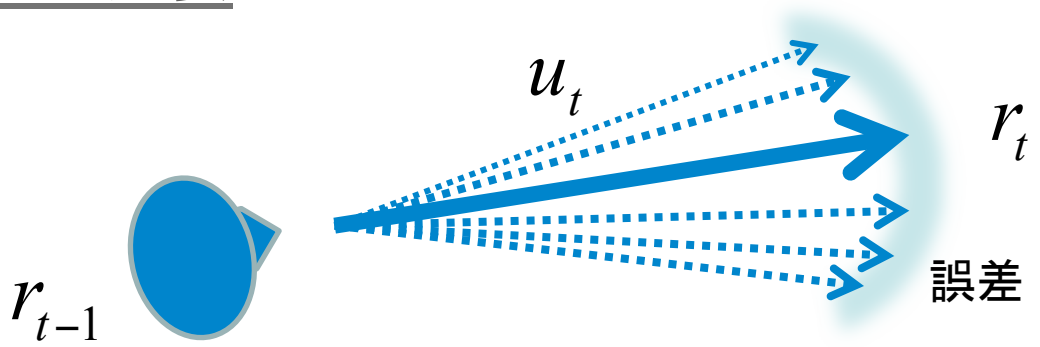
動作モデル（概要）

- ロボットの速度 v_t 、角速度 ω_t とすると $u_t = (v_t, \omega_t)^T$
- 実際には誤差を含むため、$\hat{u}_t = (\hat{v}_t, \hat{\omega}_t)^T$ （$u_t = \hat{u}_t + \varepsilon_t$ ）
- ロボットの位置 r_t は以下のように計算できる

微小変位を累積して位置推定

$$g(r_t, u_t) = \begin{pmatrix} x_{t-1} \\ y_{t-1} \\ \theta_{t-1} \end{pmatrix} + \begin{pmatrix} v_t \cos\theta_{t-1} \\ v_t \sin\theta_{t-1} \\ \omega_t \end{pmatrix} \Delta t \qquad \text{——(5)}$$

- 誤差が累積する可能性 ⇒ 計測モデルによる位置修正が必要



動作モデルと計測モデル(概要)

- 動作モデル:状態遷移確率 $p(x_t \mid u_t, x_{t-1})$
- 計測モデル:計測確率 $p(z_t \mid x_t)$

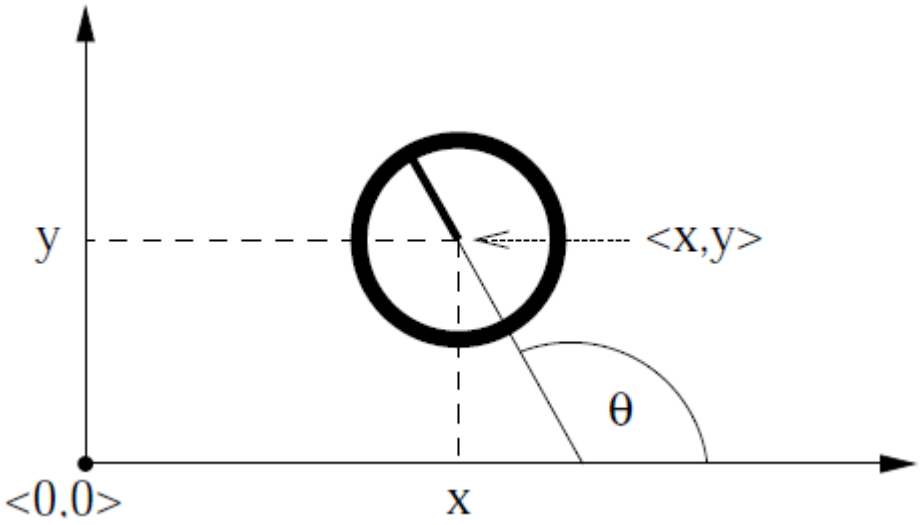
- 従来のロボット運動学 ⇒ 決定論的
- 「確率ロボティクス」⇒ モデルが確率的

- [重要!]
  厳密な形状の確率モデルはそれほど重要ではない
  ⇒ 不確かな動作結果への対策が重要

事前準備

- コンフィギュレーション：6変数で記述
  - デカルト座標（三次元）、オイラー角（三次元；ロール、ピッチ、ヨー）
- 二次平面を動くロボットの姿勢（平面上に限定）：

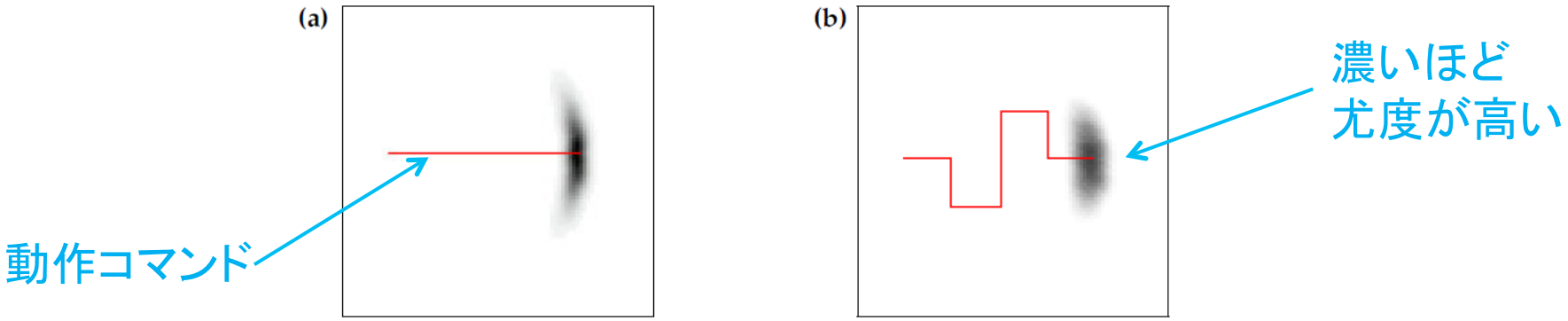
  x, y：座標、θ：方向のとき、ロボットの姿勢は以下



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

確率的運動学

- 条件付き確率密度関数　$p(x_t \mid u_t, x_{t-1})$
- ロボットの運動学モデル:
  - 最初の姿勢　x_{t-1}
  - 分布(グレースケール)　$p(x_t \mid u_t, x_{t-1})$



- 2種類の確率的動作モデル
  - 速度動作モデル: u_t はモータに与えられる速度指令
  - オドメトリ動作モデル: u_t は移動距離と回転速度(正確?)

速度動作モデル(1)

- 解析的な計算とアルゴリズム
  - 制御 u_t : 並進速度 v_t (前進方向が正)、回転速度ω_t (反時計回りが正)

$$u_t = \begin{pmatrix} v_t \\ \omega_t \end{pmatrix}$$

  - 解析的な計算:
    - 入力:初期姿勢 $x_{t-1} = (x\ y\ \theta)^T$ 、制御 $u_t = (v\ \omega)^T$
      次の姿勢の仮説 $x_t = (x'\ y'\ \theta')^T$
    - 出力: $p(x_t \mid u_t, x_{t-1})$
    - パラメータ: $\alpha_1, \ldots, \alpha_6$
    - 誤差がないと仮定した値: $\hat{v}$ 、$\hat{\omega}$
    - 関数 $\mathrm{prob}(x, b)$:原点を中心とする標準偏差 b の分布に従う確率変数 xを計算

速度動作モデル(2)

入力

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

2: $\mu = \frac{1}{2} \frac{(x - x')\cos\theta + (y - y')\sin\theta}{(y - y')\cos\theta - (x - x')\sin\theta}$

3: $x^* = \frac{x + x'}{2} + \mu(y - y')$

4: $y^* = \frac{y + y'}{2} + \mu(x' - x)$

5: $r^* = \sqrt{(x - x^*)^2 + (y - y^*)^2}$

6: $\Delta\theta = \text{atan2}(y' - y^*, x' - x^*) - \text{atan2}(y - y^*, x - x^*)$

7: $\hat{v} = \frac{\Delta\theta}{\Delta t} r^*$

8: $\hat{\omega} = \frac{\Delta\theta}{\Delta t}$

9: $\hat{\gamma} = \frac{\theta' - \theta}{\Delta t} - \hat{\omega}$

10: *return* $\textbf{prob}(v - \hat{v}, \alpha_1 v^2 + \alpha_2 \omega^2) \cdot \textbf{prob}(\omega - \hat{\omega}, \alpha_3 v^2 + \alpha_4 \omega^2)$
$\cdot \textbf{prob}(\hat{\gamma}, \alpha_5 v^2 + \alpha_6 \omega^2)$

誤差がないものとして計算

円弧を描いて移動
その中心と半径

方向の変化量

並進速度

制御は固定時間
Δt で実行

回転速度

回転方向の誤差
のパラメータ

並進方向の誤差パラメータ

出力

速度動作モデル(3)

- 動作の誤差をモデル化 $\mathrm{prob}(x, b)$
  - 正規分布について確率密度を計算するアルゴリズム
  - 三角分布についての確率密度を計算するアルゴリズム

1: **Algorithm prob_normal_distribution(a, b^2):**

2: *return* $\dfrac{1}{\sqrt{2\pi\, b^2}} \exp\left\{-\dfrac{1}{2}\dfrac{a^2}{b^2}\right\}$

3: **Algorithm prob_triangular_distribution(a, b^2):**

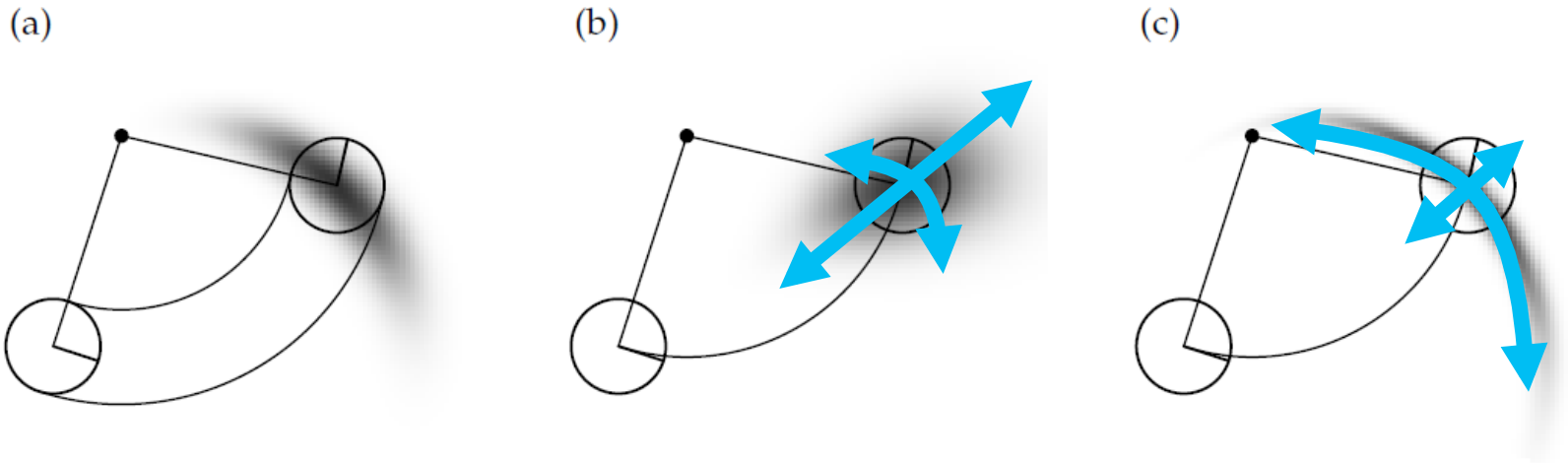
4: *return* $\max\left\{0, \dfrac{1}{\sqrt{6}\, b} - \dfrac{|a|}{6\, b^2}\right\}$

確率変数aの確率密度を計算

標準偏差b(分散b^2)の分布に従う

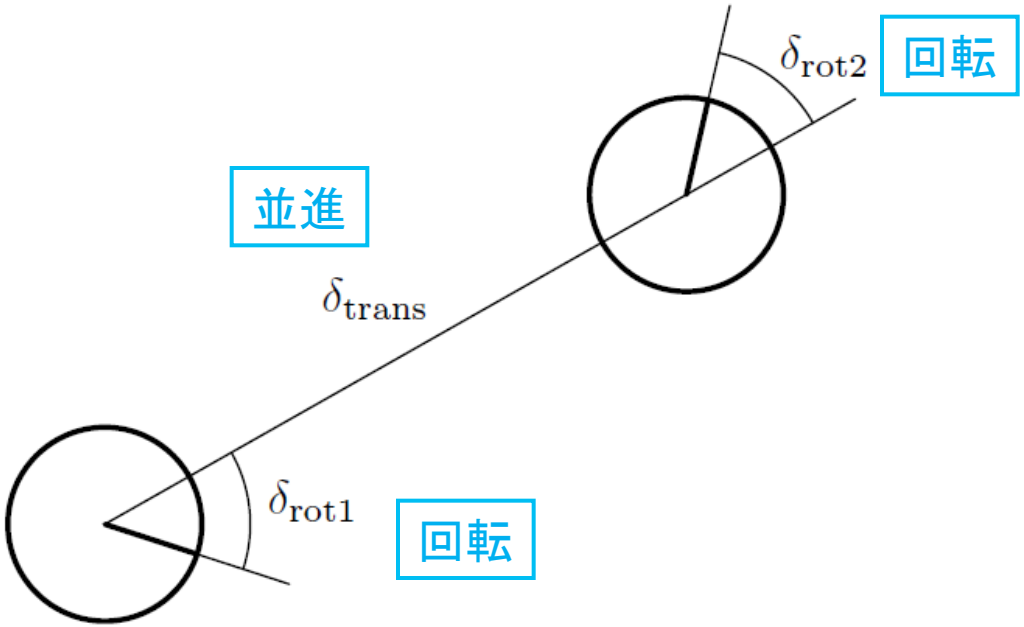
速度動作モデル（4）

- 異なる雑音パラメータの設定で得られる速度動作モデル（3例とも同じ並進速度・回転速度）
  - (a) 誤差パラメータ $\alpha_1, \ldots, \alpha_6$ を中程度に設定
  - (b) 回転方向の誤差パラメータ（α_3, α_4）を小さく、並進方向の誤差パラメータ（α_1, α_2）を大きく設定
  - (c) 逆に、回転方向の誤差を大きく、並進方向の誤差を小さく設定



オドメトリ動作モデル（1）

- 特徴：
  - オドメトリ：車輪のエンコーダの情報を積分して取得
  - オドメトリ動作モデルの特徴（速度動作モデルとの比較）：
    - オドメトリの方が情報はより正確
    - オドメトリは動作後でなければわからない
    - オドメトリはセンサ情報（通常は制御信号とみなす）



オドメトリ動作モデル（2）

- 解析的な計算：
  - 動作情報u_tは以下、ここで、バーは内部座標系で表現

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

グローバル座標系との対応は不明

  - 入力：
    - 初期姿勢 x_{t-1}
    - オドメトリから得られる姿勢の対 $u_t = (\bar{x}_{t-1}\ \bar{x}_t)^T$
    - 終端姿勢の仮説 x_t
  - 出力：
    - $p(x_t \mid u_t, x_{t-1})$ の数値

オドメトリ動作モデル（3）

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

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

オドメトリから
相対的な動作
パラメータ計算

回転
移動
回転

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

姿勢x_{t-1} とx_t から
相対的な動作
パラメータ計算

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

二通りのパラメータ
の違いに関する
発生確率計算

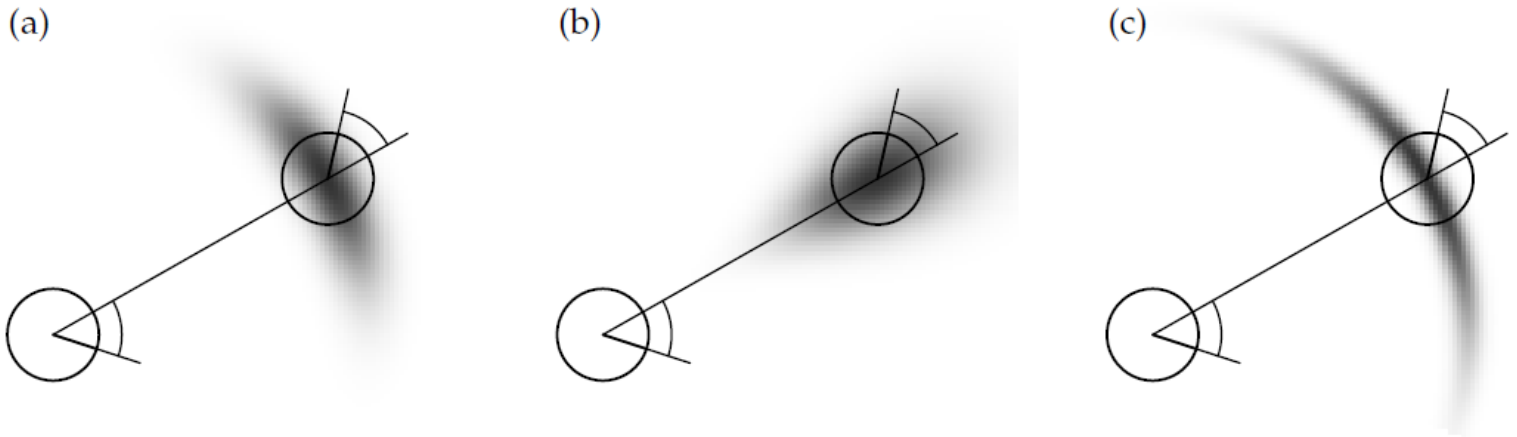
11: *return* $p_1 \cdot p_2 \cdot p_3$

各確率密度の積で
結合確率密度が出力

各誤差要因が互いに
独立であること

オドメトリ動作モデル（4）

- 誤差パラメータによる変化：
  - (a) 典型例
  - (b) 並進の誤差大
  - (c) 回転の誤差大
- 「速度動作モデル（3）」の図と比較
  - 計測時間の間隔が短い（信念が頻繁に更新される）ほど両者の差は小さくなる



動作と地図（1）

- 地図 m が考慮された動作モデル
- m が姿勢推定に意味のある情報を含む場合

$$p(x_t \mid u_t, x_{t-1}) \neq p(x_t \mid u_t, x_{t-1}, m)$$

- マップベースト動作モデル（上式の右辺）はより正確な推定結果をもたらす
- しかし、解析的な計算は困難（経路中に障害物がなくロボットが通れることを計算する必要あり）
- 近似する有効な方法：2つの成分に分解

$$p(x_t \mid u_t, x_{t-1}, m) = \eta \frac{p(x_t \mid u_t, x_{t-1}) p(x_t \mid m)}{p(x_t)}$$

地図と矛盾のない姿勢x_tのみに
確率が割り当てられた確率分布を得る

動作と地図（2）

– $p(x_t \mid u_t, x_{t-1}, m)$ を計算するアルゴリズム
  - 障害物に占有された地図 m 中の空間にはロボットが存在できない処理を付加

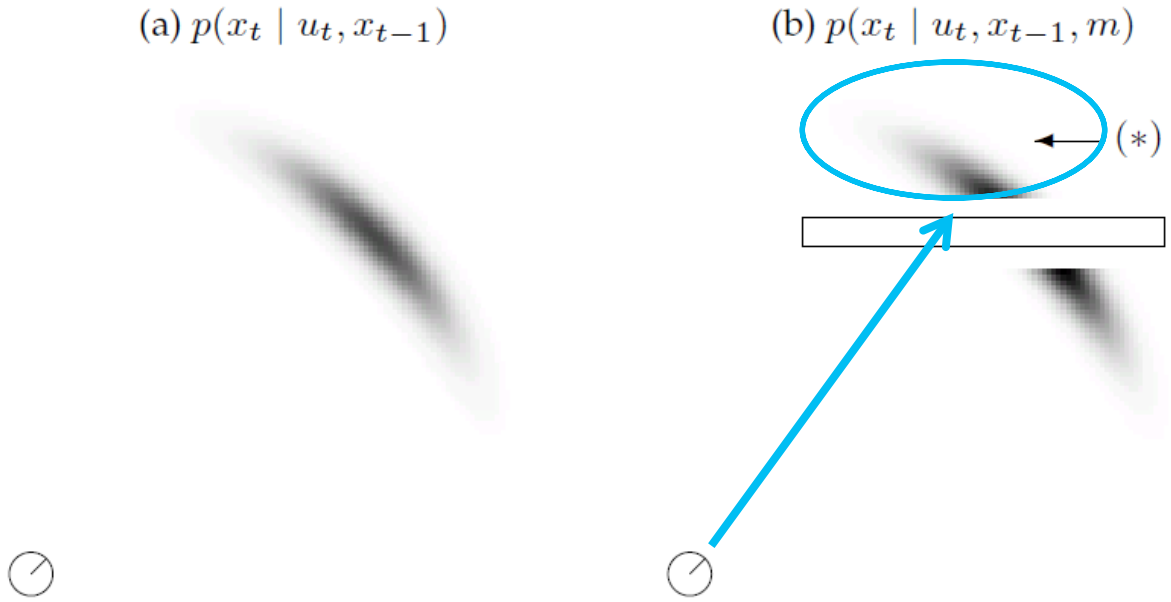
1: **Algorithm motion_model_with_map**(x_t, u_t, x_{t-1}, m)**:**
2: *return* $p(x_t \mid u_t, x_{t-1}) \cdot p(x_t \mid m)$

1: **Algorithm sample_motion_model_with_map**(u_t, x_{t-1}, m)**:**
2: *do*
3: $x_t =$ **sample_motion_model**(u_t, x_{t-1})
3: $\pi = p(x_t \mid m)$
4: *until* $\pi > 0$
5: *return* $\langle x_t, \pi \rangle$

地図が占有されているか？

動作と地図（3）

- 速度動作と地図：
  - (a)地図を考慮しない場合、(b) 地図m を考慮する場合
- 青い枠の領域の尤度はゼロではない
  $p(x_t \mid m)$と$p(x_t \mid u_t, x_{t-1})$のどちらもゼロではないから
- ロボットは壁をすり抜けるのか？



計測モデル

重要

計測モデル（概要）

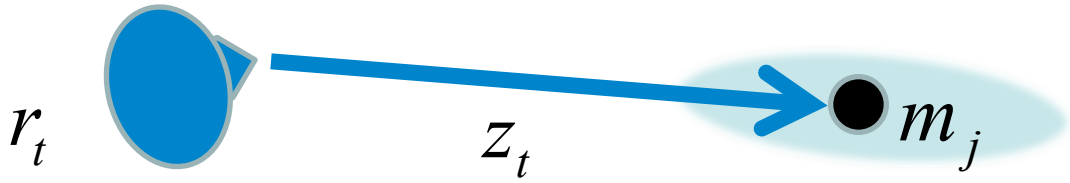
- 計測モデルを厳密に書くには、ランドマークと計測データの対応づけ関数 c_t として、$p(z_t \mid r_t, c_t, m)$ となる
- 計測モデルはセンサの能力に大きく依存する
  - レーザスキャナの場合、対象物までの距離と方向が得られる
- いまランドマーク m_j の位置 $(m_{j,x}, m_{j,y})^T$ とすると、ロボットの位置 r_t で得る計測データは以下となる

$$h(r_t, m_j) = \begin{pmatrix} \sqrt{(m_{j,x} - x_t)^2 + (m_{j,y} - y_t)^2} \\ \tan^{-1}\left(\frac{m_{j,y} - y_t}{m_{j,x} - x_t}\right) - \theta_t \end{pmatrix} \quad \text{——(6)}$$

ランドマークからの距離

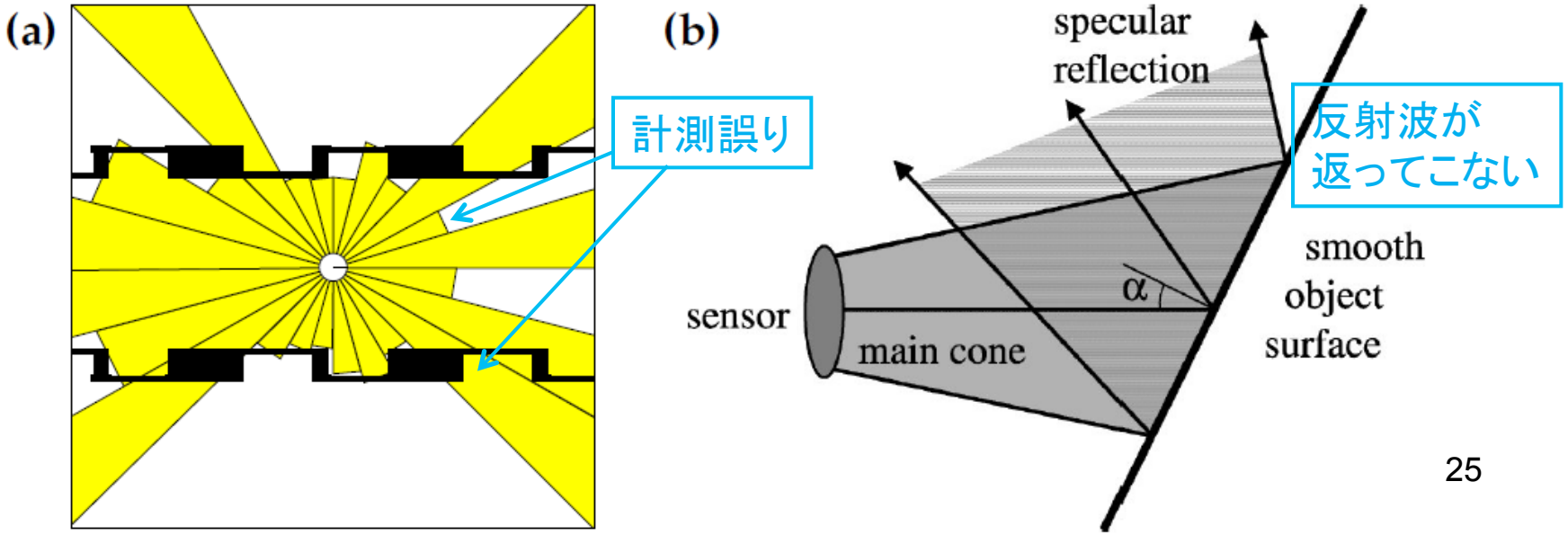
ランドマークとの角度

- 制約関係 $z_t = h(r_t, m_j)$ は成り立つがランドマーク1個から r_t を一意に求めることができない ⇒ 動作モデルで補う



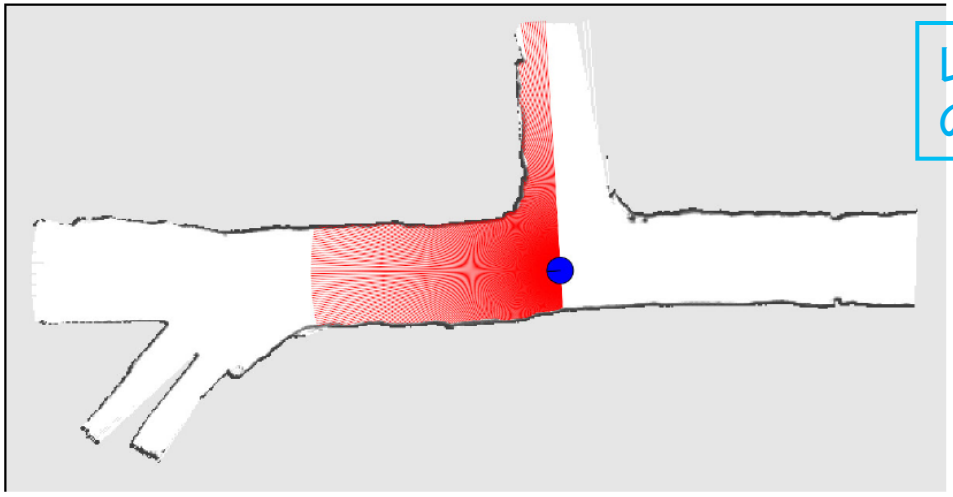
計測モデル（1）

- 計測モデル：センサ計測値の生成過程を記述
- 確率ロボティクス：センサ計測値の雑音を明示的にモデルに組み込む
- 計測モデル $p(z_t \mid x_t, m)$
  - ここでは、z_t は距離センサ（ソナーレンジスキャン）



計測モデル(2):確率モデルの特徴

- 典型的なレーザレンジスキャン
  - 照射範囲が音波よりずっと絞られている(狭い)
  - センサの応答特性 ⇒ 扱いたくない変数を含む
- 確率ロボティクス:決定論的な関数 $z_t = f(x_t)$ の代わりに条件付き確率密度 $p(z_t \mid x_t)$ でモデル化
  - センサモデルの不確かさ ⇒ モデルの非決定論的な性質で吸収できる



レーザレンジスキャンの典型例(炭坑内)

計測モデル(3):定式化

- 計測 z_t :計測値の数K

$$z_t = \left\{ z_t^1, \ldots, z_t^K \right\}$$

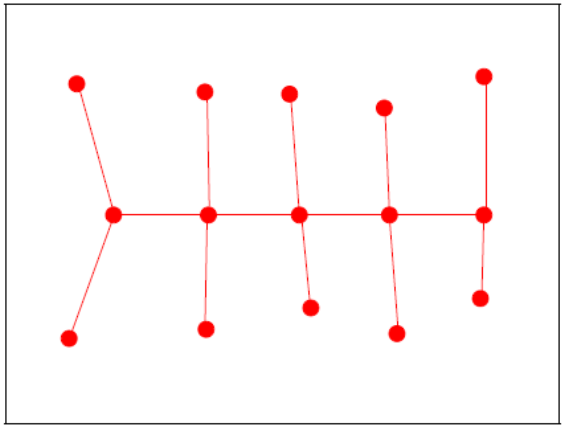
- 確率 $p(z_t \mid x_t, m)$は、個々の計測値に対する尤度の積

$$p(z_t \mid x_t, m) = \prod_{k=1}^{K} p(z_t^k \mid x_t, m)$$

  - 上の式が成り立つのは独立仮定(個々の計測ビームに生じる雑音が独立)の場合

計測モデル(4):地図

- 地図 $m = \{m_1, m_2, \ldots, m_N\}$ （Nは物体の個数）
- 地図の構築方法:
  - 特徴ベース:
    - インデックスは特徴と対応づけられる
    - 物体位置を調節することが容易
    - センサ情報から地図を作成するロボット ⇒ 特徴ベース採用
  - 位置ベース:
    - インデックスは特定の位置と対応づけられる
    - ボリューメトリック(容積測定的) ⇒ 物体のない場所も記述



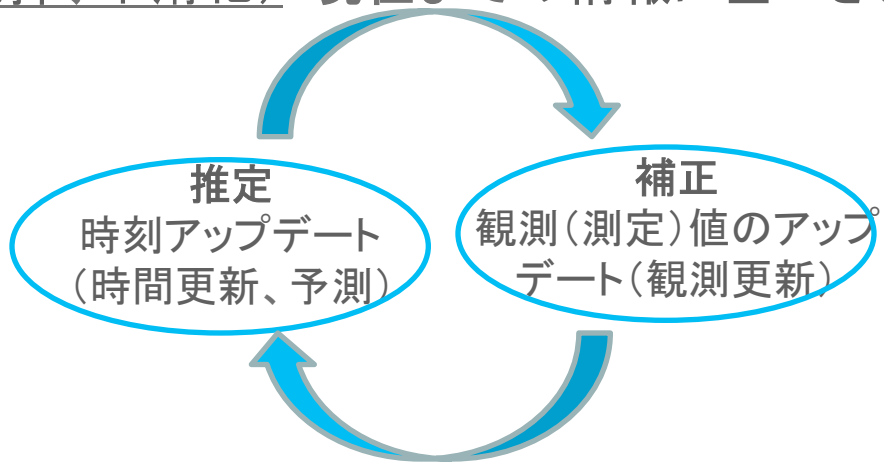
カルマンフィルタによる位置追跡

カルマンフィルタとは？

参考文献：“*An Introduction to the Kalman Filter*”, G. Welch and G. Bishop）

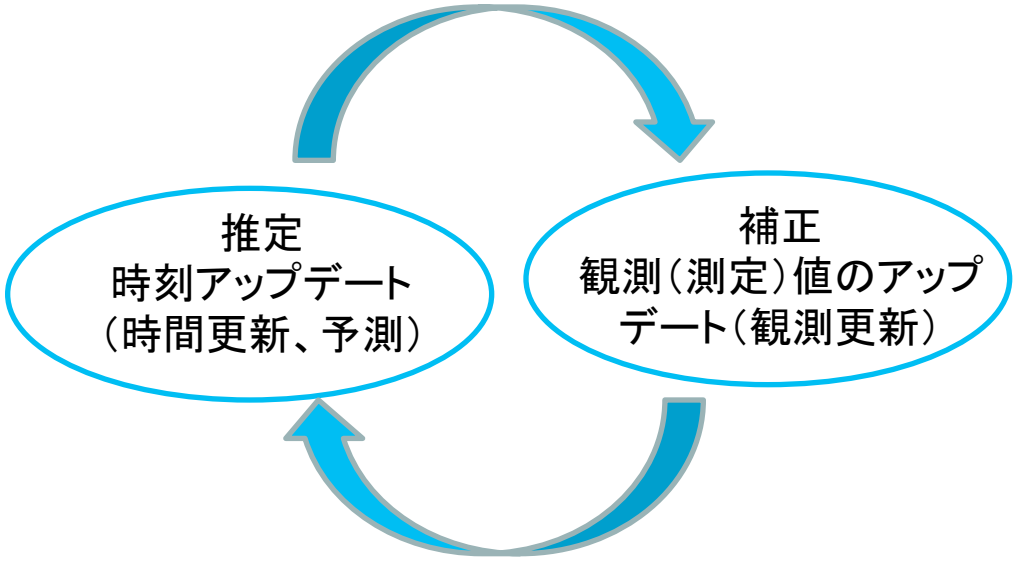
- 離散的な誤差のある観測から、時々刻々と時間変化する量（位置、速度など）を推定する
  - 応用：レーダー、コンピュータビジョンなど
  - 特徴：目標物の時間変化を支配する法則を活用して、目標物の位置を推定（現在：$t = t_0$、未来：$t < t_0$、過去：$t > t_0$
    - 現在（フィルタ）：現在の情報に基づき、現在の状態を推定
    - 未来（予測）：現在までの情報に基づき、未来の状態を推定
    - 過去（内挿、平滑化）：現在までの情報に基づき、過去の状態を推定

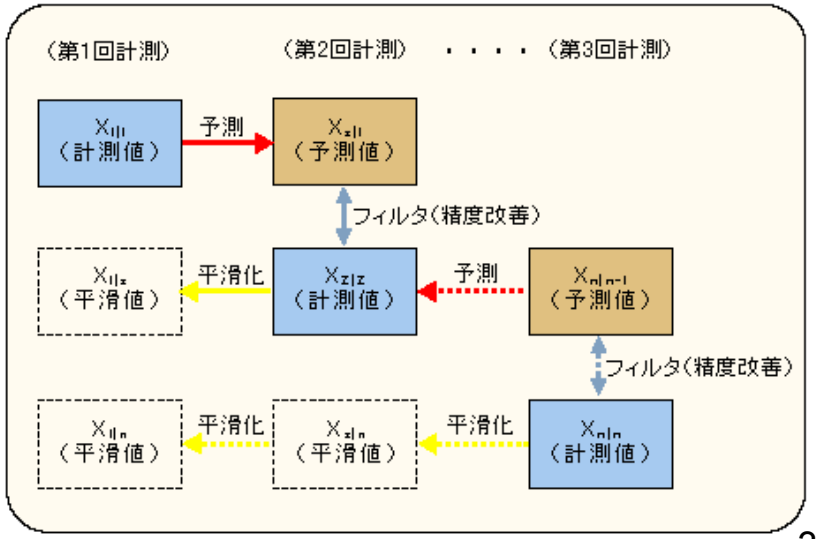
現在の状態は必ずしも直接観測できない



カルマンフィルタとは？

- 離散的な誤差のある観測から、時々刻々と時間変化する量（位置、速度など）を推定する
  - 応用：レーダー、コンピュータビジョンなど
  - 特徴：目標物の時間変化を支配する法則を活用して、目標物の位置を現在（フィルタ）、未来（予測）、過去（内挿あるいは平滑化）において推定可能





カルマンフィルタ(1)

- カルマンフィルタの特徴:
  - 連続状態に対して信念に関する計算が実装される
  - 離散系やハイブリッドな状態空間には適用できない
- カルマンフィルタにおいて、事後信念がガウス分布となる条件:

  いま、時刻 t において信念は平均 μ と共分散 Σ で表現

1. 状態遷移確率 $p(x_t \mid u_t, x_{t-1})$ がガウス雑音を足した線形関数である必要がある. ここで、A_t と B_t は行列

$$x_t = A_t x_{t-1} + B_t u_t + \varepsilon_t \quad -(1)$$

2. 計測確率 $p(z_t \mid x_t)$ も線形であり、雑音はガウス雑音

$$z_t = C_t x_t + \delta_t \quad -(2)$$

ガウス分布に従う乱数ベクトル

線形ガウス型

カルマンフィルタ(2)

3. 初期信念 $bel(x_0)$ は正規分布でなければならない.

共分散

平均

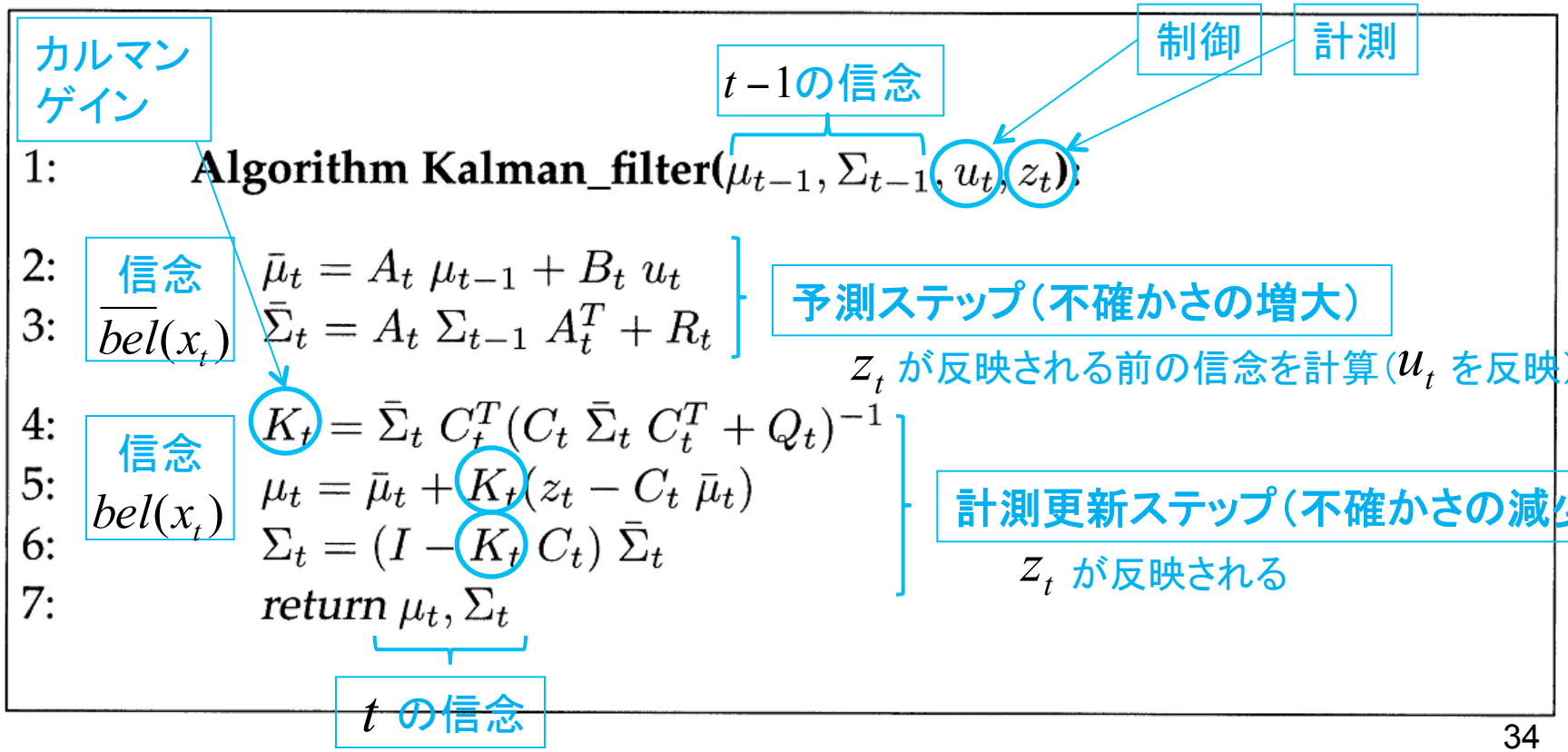
$$bel(x_0) = p(x_0) = \det(2\pi\Sigma_0)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\right\}$$

多変量正規分布

—(3)

カルマンフィルタ(3)

- カルマンフィルタのアルゴリズム
  - 時刻 t の信念 $bel(x_t)$ を平均μ_t と共分散Σ_t で表現



カルマンフィルタ(4):詳細な処理

- 2行目:平均値 $\overline{\mu}$ は式(1)の状態 x_{t-1} を μ_{t-1} で置き換えて、決定論的な部分のみ計算
- 3行目:共分散 $\overline{\Sigma}$ は、事後状態が A_t の作用で事前状態から遷移することを反映して更新. R_t は乱数ベクトルの共分散
- 4行目:カルマンゲイン K_t は、計測を新たな状態推定にどの程度反映させるかを決定
- 5行目:平均値の修正(カルマンゲイン K_t、実際の計測値 z_t と計算確率の式(2)によって予測された計測値の差
- 6行目:事後信念の共分散を計算

カルマンフィルタ(5):構成要素

A_t n×nの正方行列. 制御やノイズなしに、$t-1$から t へ状態 x_t がいかに発展するかを記述

B_t n×m行列. 制御 u_tが $t-1$から tにおいて状態 x_tをいかに変化させるかを記述

C_t k×n行列. 状態 x_tを計測 z_tにどのように位置づけるかを記述

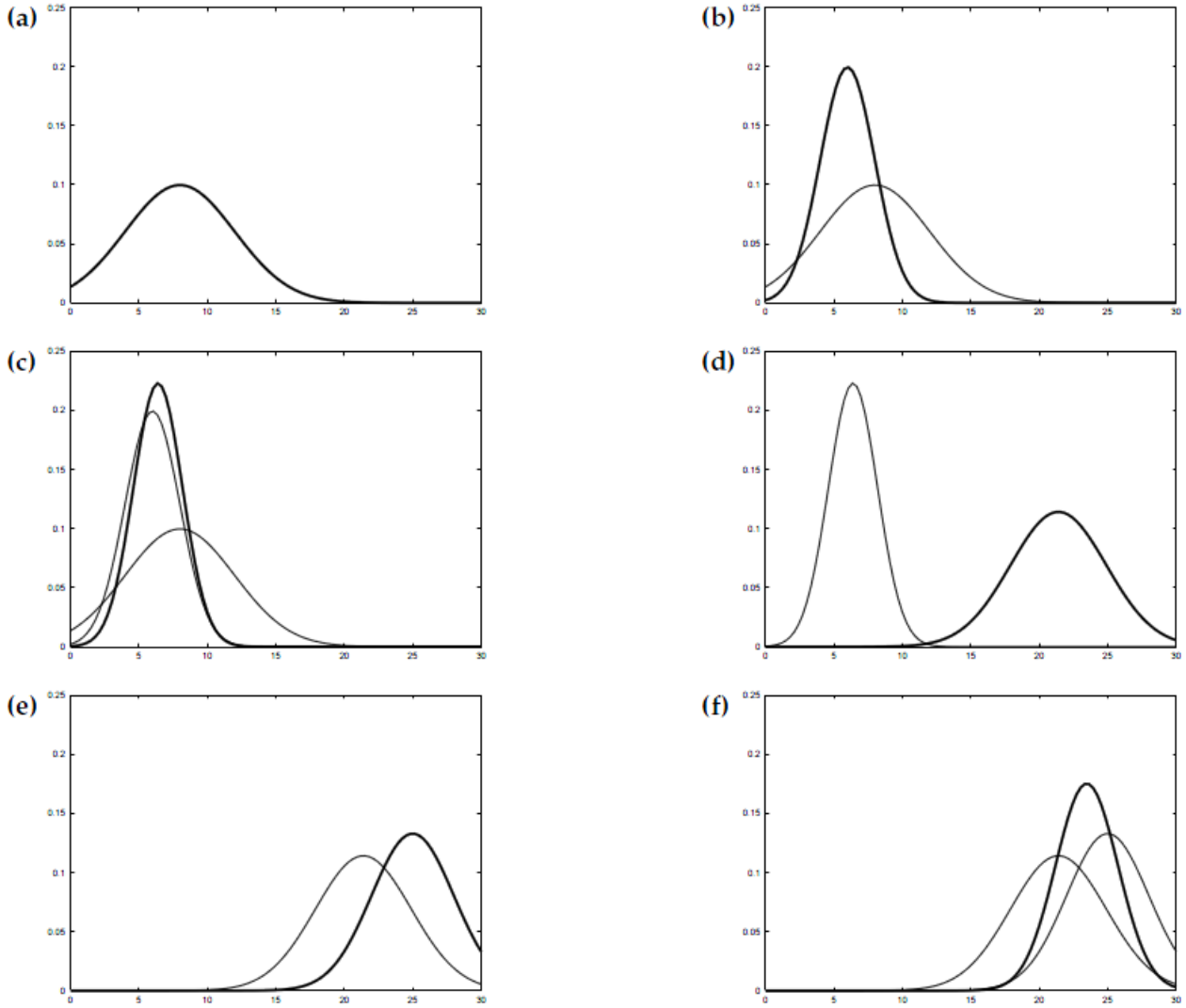
ε_t 乱数ベクトル. 共分散 R_t, Q_tからは独立で、正規分布すると仮定される、処理と計測の誤差を表現

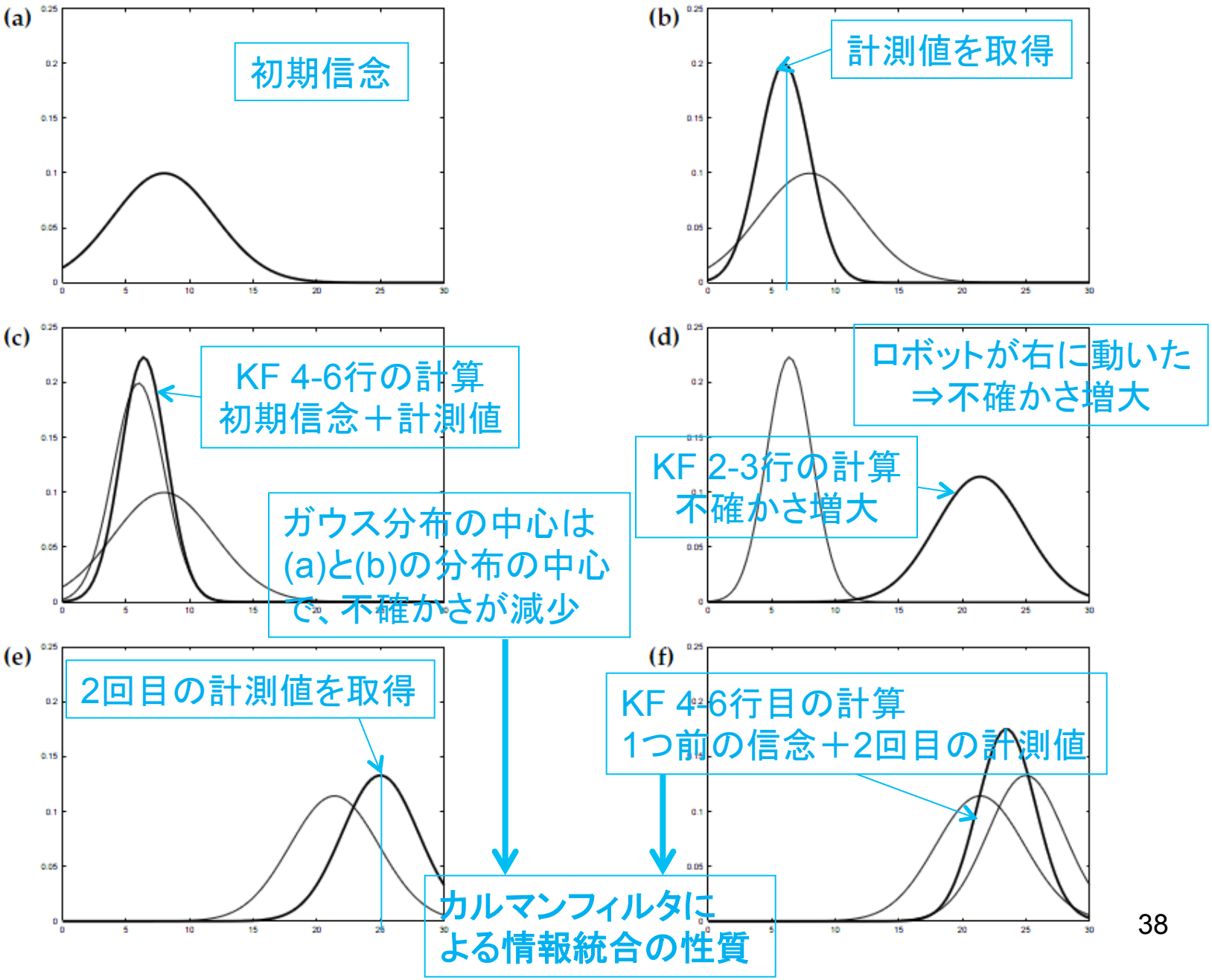
δ_t

n:状態ベクトル x_t の次元、m:制御ベクトル u_t の次元、
k:計測ベクトル z_t の次元

カルマンフィルタ(5)

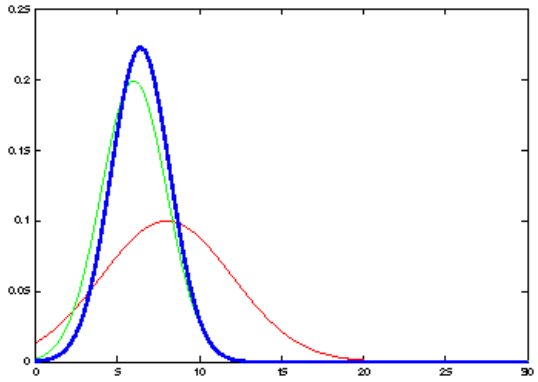
- 一次元での位置推定の例(ロボットは横軸を移動)




(a)
初期信念
(b)
計測値を取得
(c)
KF 4-6行の計算
初期信念＋計測値
(d)
ロボットが右に動いた
⇒不確かさ増大
KF 2-3行の計算
不確かさ増大
ガウス分布の中心は
(a)と(b)の分布の中心
で、不確かさが減少
(e)
2回目の計測値を取得
(f)
KF 4-6行目の計算
1つ前の信念＋2回目の計測値
カルマンフィルタに
よる情報統合の性質

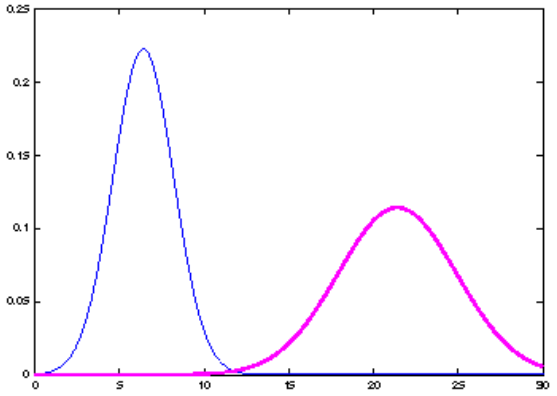
カルマンフィルタの予測ステップ

Prediction

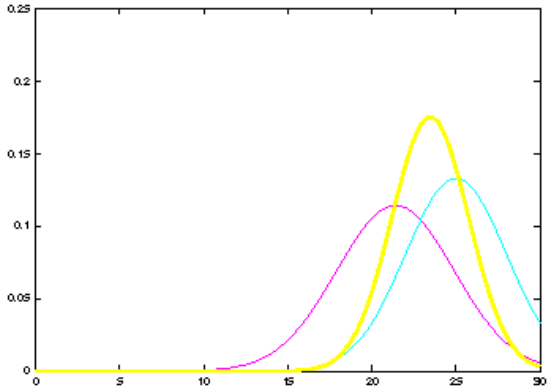


$$\overline{bel}(x_t) = \begin{cases} \overline{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \overline{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

$$\overline{bel}(x_t) = \begin{cases} \overline{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \overline{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$

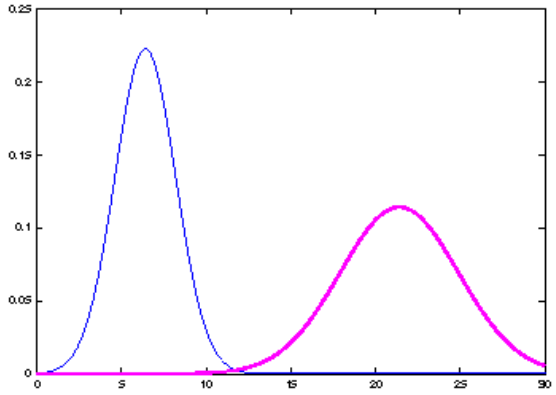


カルマンフィルタの計測更新ステップ



$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t)\bar{\sigma}_t^2 \end{cases}, K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t\bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases}, K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$



Correction

カルマンフィルタ(7):
予測・計測更新サイクル

Prediction

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = a_t \mu_{t-1} + b_t u_t \\ \bar{\sigma}_t^2 = a_t^2 \sigma_t^2 + \sigma_{act,t}^2 \end{cases}$$

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - \bar{\mu}_t) \\ \sigma_t^2 = (1 - K_t)\bar{\sigma}_t^2 \end{cases}, K_t = \frac{\bar{\sigma}_t^2}{\bar{\sigma}_t^2 + \bar{\sigma}_{obs,t}^2}$$

$$\overline{bel}(x_t) = \begin{cases} \bar{\mu}_t = A_t \mu_{t-1} + B_t u_t \\ \bar{\Sigma}_t = A_t \Sigma_{t-1} A_t^T + R_t \end{cases}$$

$$bel(x_t) = \begin{cases} \mu_t = \bar{\mu}_t + K_t(z_t - C_t \bar{\mu}_t) \\ \Sigma_t = (I - K_t C_t)\bar{\Sigma}_t \end{cases}, K_t = \bar{\Sigma}_t C_t^T (C_t \bar{\Sigma}_t C_t^T + Q_t)^{-1}$$

Correction

カルマンフィルタの動作例

- カルマンフィルタを用いた慣性計測ユニット（Inertial Measurement Unit ）のデモ（比較：IMUの加速度計からのデータを融合しただけの場合）

カルマンフィルタ(6):まとめ

- カルマンフィルタは、センサデータを信念に反映する計測更新ステップ(5～7行目)と、行動によって信念を変形させる予測ステップ(あるいは制御更新ステップ)を交互に実行する
- 計測更新ステップはロボットの信念の不確かさを減少させ、予測ステップは増加させる

– 拡張カルマンフィルタとは？

- カルマンフィルタは、計測が状態の線形関数で、状態が直前の状態の線形関数であることが不可欠
- しかし、状態遷移や計測が線形であることは、現実にはほとんどない
- 拡張カルマンフィルタ(extended kalman filter, EKF)は線形性の仮定を緩和する

拡張カルマンフィルタ

拡張カルマンフィルタによる位置追跡(1)

- 動作モデルと計測モデルをベイズフィルタで融合することにより、上記のそれぞれの短所を補完できる
  - ロボットの位置、動作モデル、計測モデルをそれぞれ一つの正規分布で表す
- 動作モデル
  - 関数 g は非線形なので、テーラー展開で線形化
  - G_t は r_t に関する g のヤコビ行例、V_t は u_t に関する g のヤコビ行例
  - $\hat{u}_t$ を動作指令値(オドメトリ値)とし、u_t は正規分布 $N(\hat{a}_t, M_t)$ に従う

一変数スカラ関数の接線の傾きを、多変数ベクトル値関数に対して拡張、高次元化

変化の比率を符号付きで表現

$$r_t = g(r_{t-1}, u_t)$$
$$\approx g(\bar{r}_{t-1}, \hat{u}_t) + G_t(r_{t-1} - \bar{r}_{t-1}) + V_t(u_t - \hat{u}_t) \quad \text{——(6)}$$

拡張カルマンフィルタによる位置追跡(2)

- 計測モデル
  - h も非線形関数なので、テーラー展開で線形化
  - H_tは r_t に関する h のヤコビ行例
  - β_t は誤差で正規分布 $N(0, Q_t)$ に従う

$$z_t = h(r_t, m_j) + \beta_t$$
$$\approx h(\bar{r}_t, m_j) + H_t(r_t - \bar{r}_{t-1}) + \beta_t \quad \text{——(7)}$$

変化の比率を符号付きで表現

誤差

拡張カルマンフィルタによる位置追跡(3)

- ロボットの位置の分布を $N(\bar{r}_t, \Sigma_t)$ とすると、拡張カルマンフィルタにより次のように計算できる

(1) 動作モデルによる r_t の予測

$$\tilde{r}_t = g(\bar{r}_{t-1}, \hat{u}_t)$$

$$\tilde{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T \quad \text{——(8)}$$

(2) 計測モデルによる r_t の更新

$$K_t = \tilde{\Sigma}_t H_t^T (H_t \tilde{\Sigma}_t H_t^T + Q_t)^{-1}$$

$$\bar{r}_t = \tilde{r}_t + K_t (z_t - h(\tilde{r}_t, m_j)) \quad \text{——(9)}$$

$$\Sigma_t = (I - K_t H_t) \tilde{\Sigma}_t$$

K_t はカルマンゲイン
動作モデルと計測モデルの精度に応じた重みをつけて最適値を得る
また、実測値をセンサ座標系から地図座標系に変換する

動作モデルの値

実測値

計測モデルの値

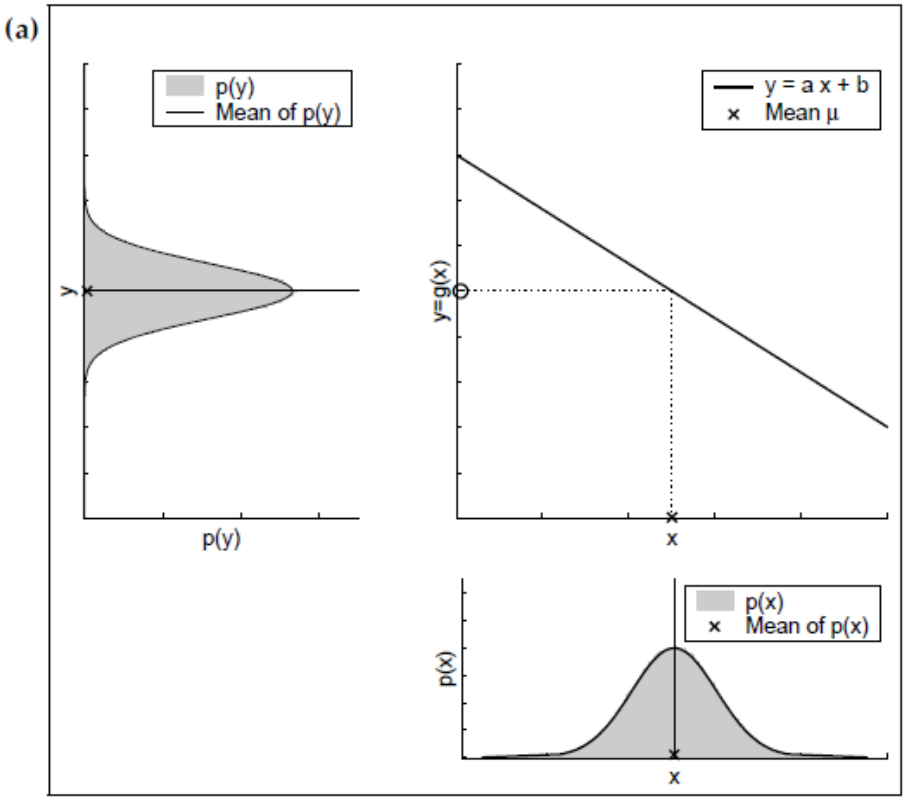
拡張カルマンフィルタによる位置追跡(4)

- カルマンフィルタは、動作モデル $\bar{r}_t$ と、実測値 z_t と計測モデルの値 $h(\tilde{r}_t, m_j)$ の差とを K_t で重みづけした線形和
- カルマンゲイン K_t は、動作モデルと計測モデルの精度に応じた重みをつけて最適値を得る働きをする

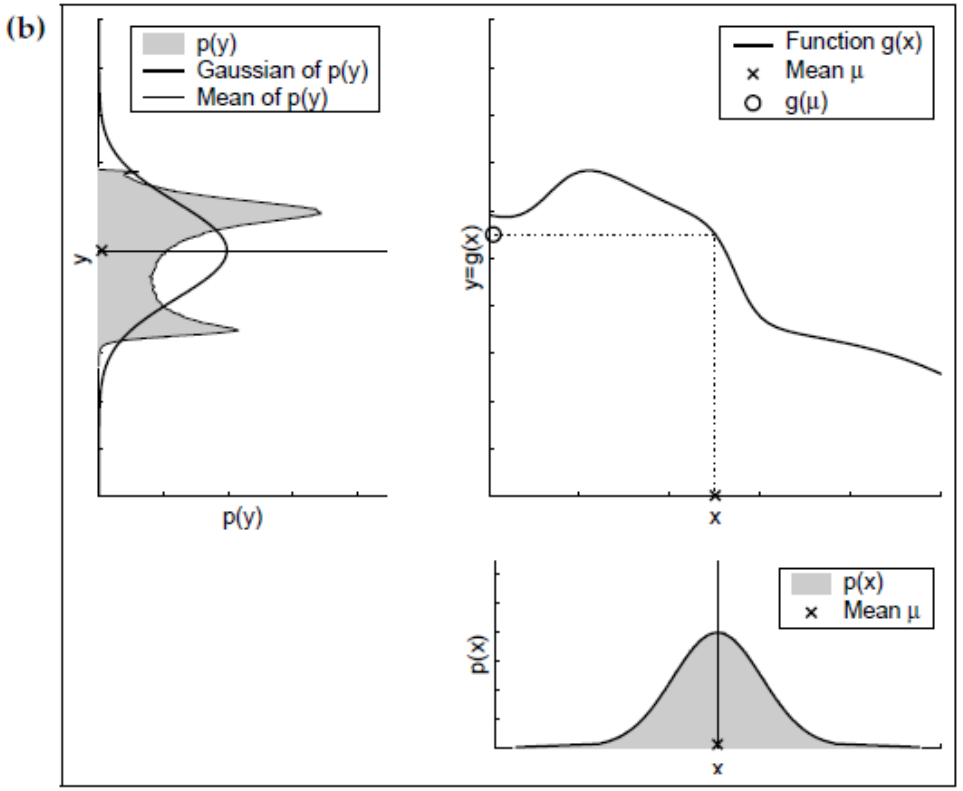
カルマンフィルタにより、複数のデータを融合して
最適解を得ることができる

拡張カルマンフィルタ(1):
線形変換と非線形変換

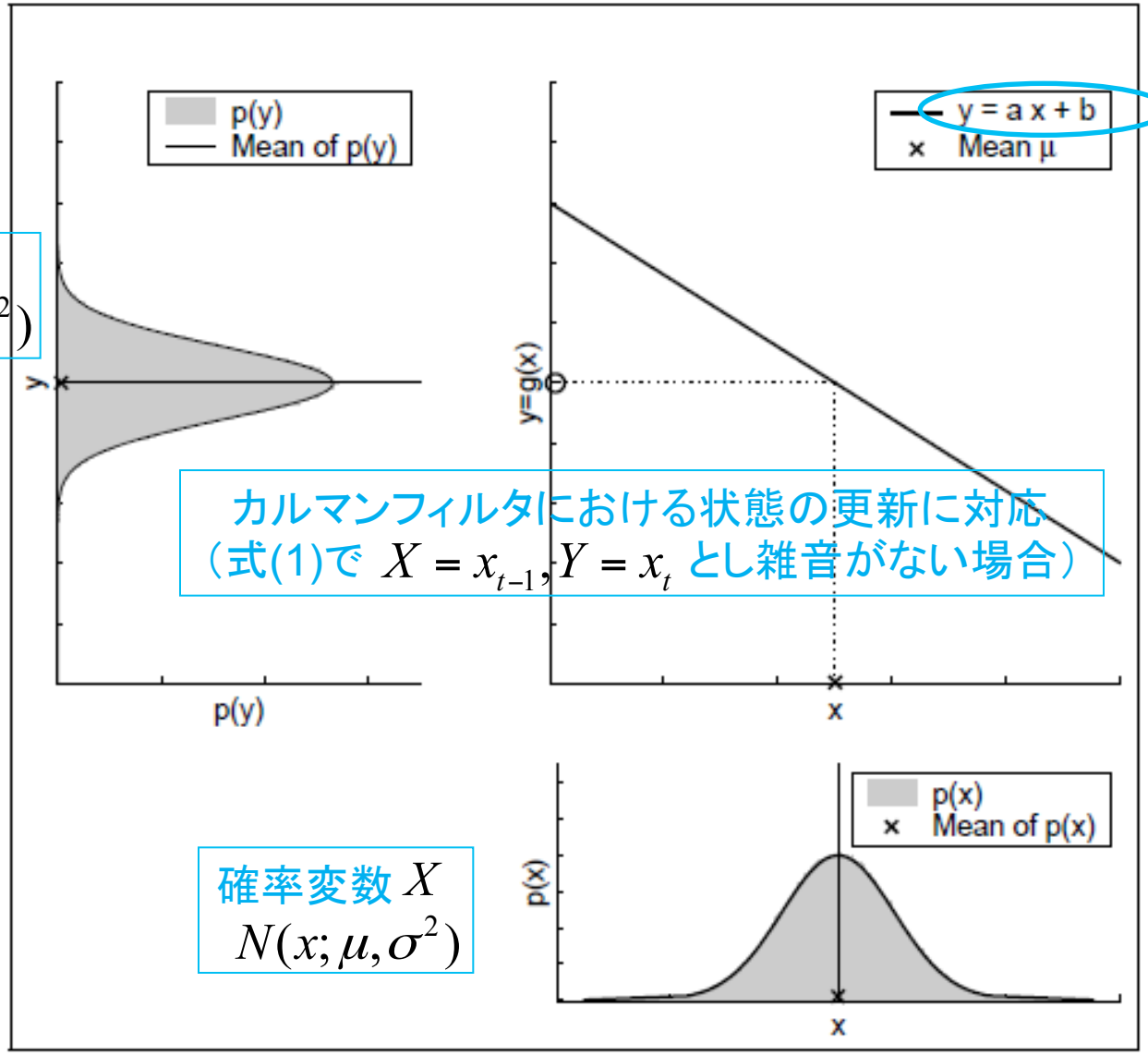
線形変換



非線形変換



線形変換



確率変数 Y
$N(y; a\mu + b, a^2\sigma^2)$

カルマンフィルタにおける状態の更新に対応
（式(1)で $X = x_{t-1}, Y = x_t$ とし雑音がない場合）

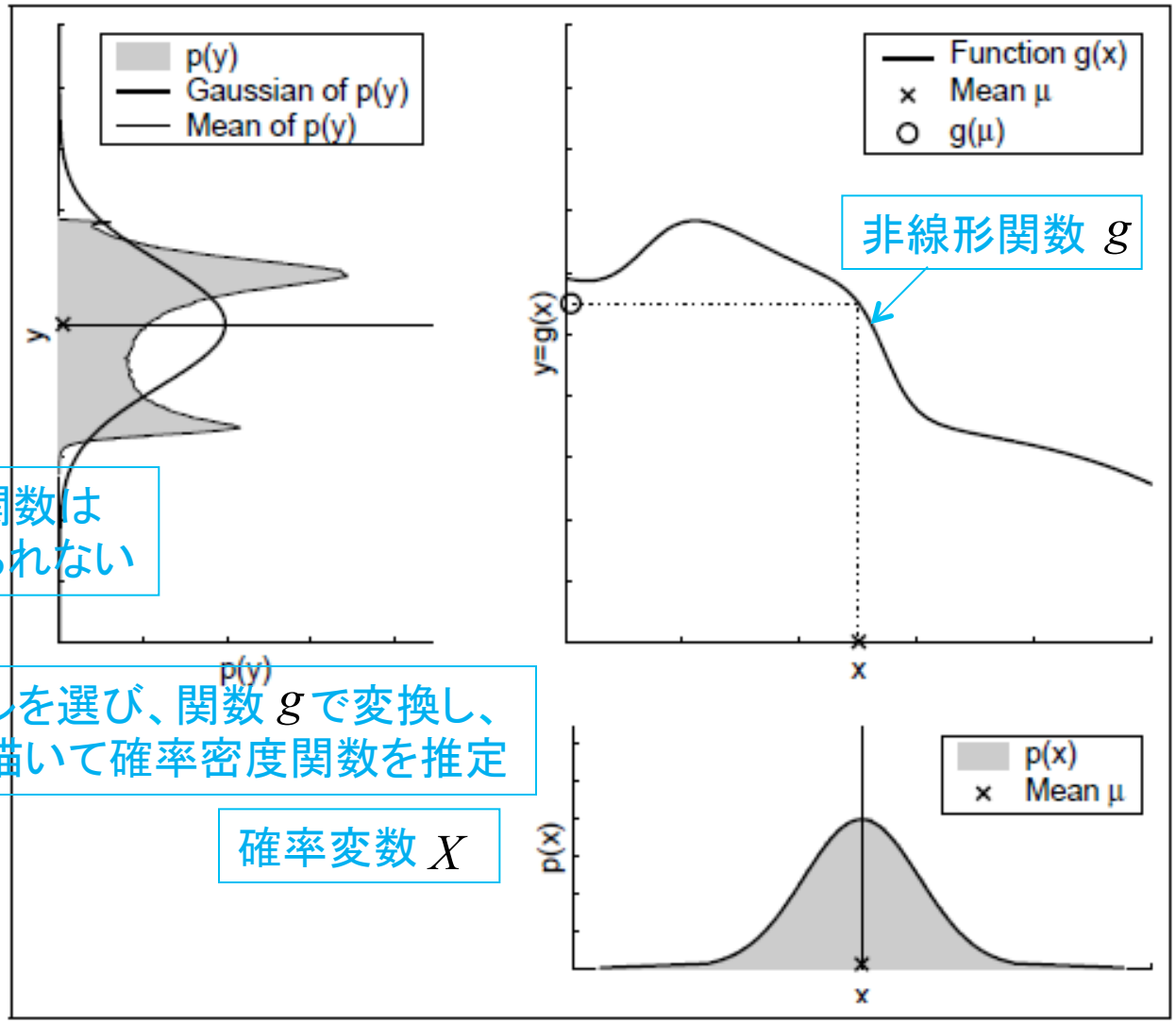
確率変数 X
$N(x; \mu, \sigma^2)$

非線形変換

確率密度関数
$Y = g(X)$

この確率密度関数は
数式では求められない

$p(x)$ のサンプルを選び、関数 g で変換し、
ヒストグラムを描いて確率密度関数を推定



確率変数 X

拡張カルマンフィルタの位置推定アルゴリズム（参考）

Algorithm EKF_localization_known_correspondences$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t, m)$:

$\theta = \mu_{t-1,\theta}$

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

$$V_t = \begin{pmatrix} \frac{-\sin\theta+\sin(\theta+\omega_t\Delta t)}{\omega_t} & \frac{v_t(\sin\theta-\sin(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\cos(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ \frac{\cos\theta-\cos(\theta+\omega_t\Delta t)}{\omega_t} & -\frac{v_t(\cos\theta-\cos(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\sin(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ 0 & \Delta t \end{pmatrix}$$

$$M_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2\omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4\omega_t^2 \end{pmatrix}$$

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

$$\bar{\Sigma}_t = G_t\,\Sigma_{t-1}\,G_t^T + V_t\,M_t\,V_t^T$$

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

9: *for all observed features* $z_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$ *do*

10: $j = c_t^i$

11: $q = (m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$

12: $\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(m_{j,y} - \bar{\mu}_{t,y}, m_{j,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \\ m_{j,s} \end{pmatrix}$

13: $H_t^i = \begin{pmatrix} -\frac{m_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y} - \bar{\mu}_{t,y}}{q} & -\frac{m_{j,x} - \bar{\mu}_{t,x}}{q} & -1 \\ 0 & 0 & 0 \end{pmatrix}$

14: $S_t^i = H_t^i \; \bar{\Sigma}_t \; [H_t^i]^T + Q_t$

15: $K_t^i = \bar{\Sigma}_t \; [H_t^i]^T [S_t^i]^{-1}$

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

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

18: *endfor*

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

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

21: $p_{z_t} = \prod_i \det\left(2\pi S_t^i\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} \; (z_t^i - \hat{z}_t^i)^T [S_t^i]^{-1} (z_t^i - \hat{z}_t^i)\right\}$

22: *return* μ_t, Σ_t, p_{z_t}

拡張カルマンフィルタ(2):特徴

- 拡張カルマンフィルタは、線形性の仮定を緩和する
  - 状態遷移確率と計測確率が非線形関数g, hに従うと仮定し、以下のようにモデル化する

$$x_t = g(u_t, x_{t-1}) + \varepsilon_t \quad \cdots (4)$$

$$z_t = h(x_t) + \delta_t \quad \cdots (5)$$

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

$z_t = C_t x_t + \delta_t$

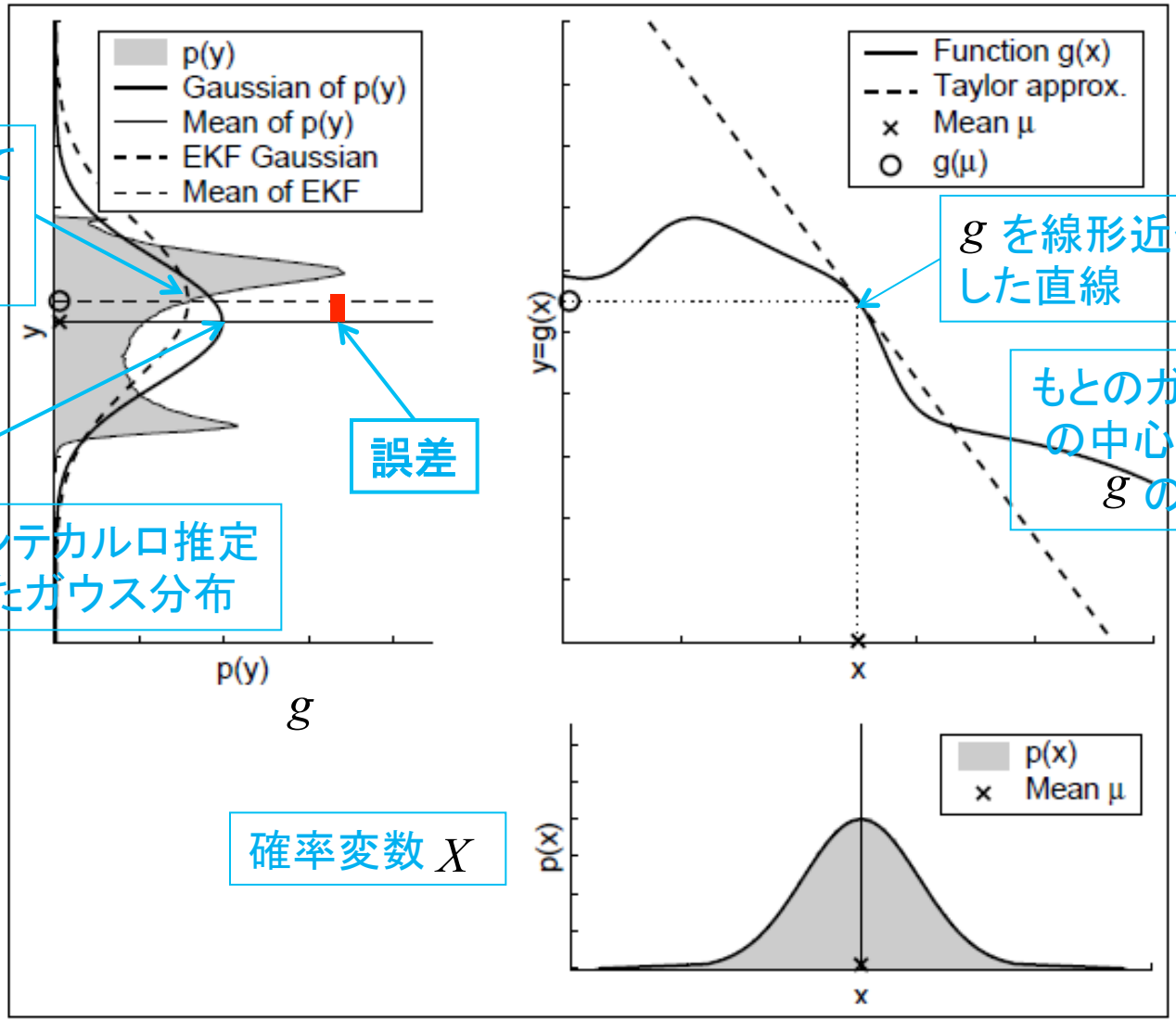
  - 上記の式は、式(1)(2)を一般化したもの
  - しかし、g, h は任意であるため、信念はガウス分布に従わない
  - 非線形関数g, hに対して信念を正確に更新することは不可能 ⇒ ベイズフィルタは数式解を持たなくなる

拡張カルマンフィルタ(3):近似

- EKFでは推定対象の状態に対してガウス分布の近似が計算される
  - EKFも時刻 t の信念 $bel(x_t)$ を平均 μ_t 、共分散 Σ_t で表す ⇒ カルマンフィルタの基本的な方法を受け継ぐ
  - しかし、EKFの信念は単なる近似

  - 複雑な分布形状の事後信念を厳密に計算しようとせず、平均と共分散を計算効率良く推定すること

拡張カルマンフィルタ(4):線形化



線形化によって得られるガウス分布

誤差

高精度のモンテカルロ推定から得られたガウス分布

g を線形近似した直線

もとのガウス分布の中心における g の接線

確率変数 X

拡張カルマンフィルタ(5):効率・手法

- 線形近似の効率の良さ
  - モンテカルロ推定:サンプルの変換を50万回繰り返し、平均と分散を計算
  - EKF:線形関数を選び、その関数からガウス分布の式を求めるだけ(gが線形化されるとKFと同じ)

    計測関数hに関するガウス分布の乗算にも適用可能
- 線形化する手法
  - テイラー展開:関数gの値と傾きから、gの線形近似を行う

$$g'(u_t, x_{t-1}) := \frac{\partial g(u_t, x_{t-1})}{\partial x_{t-1}}$$

  - 最尤な状態を選ぶ ⇒ ガウス分布では事後信念の平均値μ_{t-1}、計測関数hでは$\bar{\mu}_{t-1}$

拡張カルマンフィルタ(6):アルゴリズム

– KFとの違い(KF ⇒ EKF) 線形予測 ⇒ 非線形予測

- 2行目(予測): $A_t \mu_{t-1} + B_t u_t \Rightarrow g(u_t, \mu_{t-1})$
- 5行目(計測): $C_t \bar{\mu}_t \Rightarrow h(\bar{\mu}_t)$

Algorithm Extended_Kalman_filter$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t)$**:**

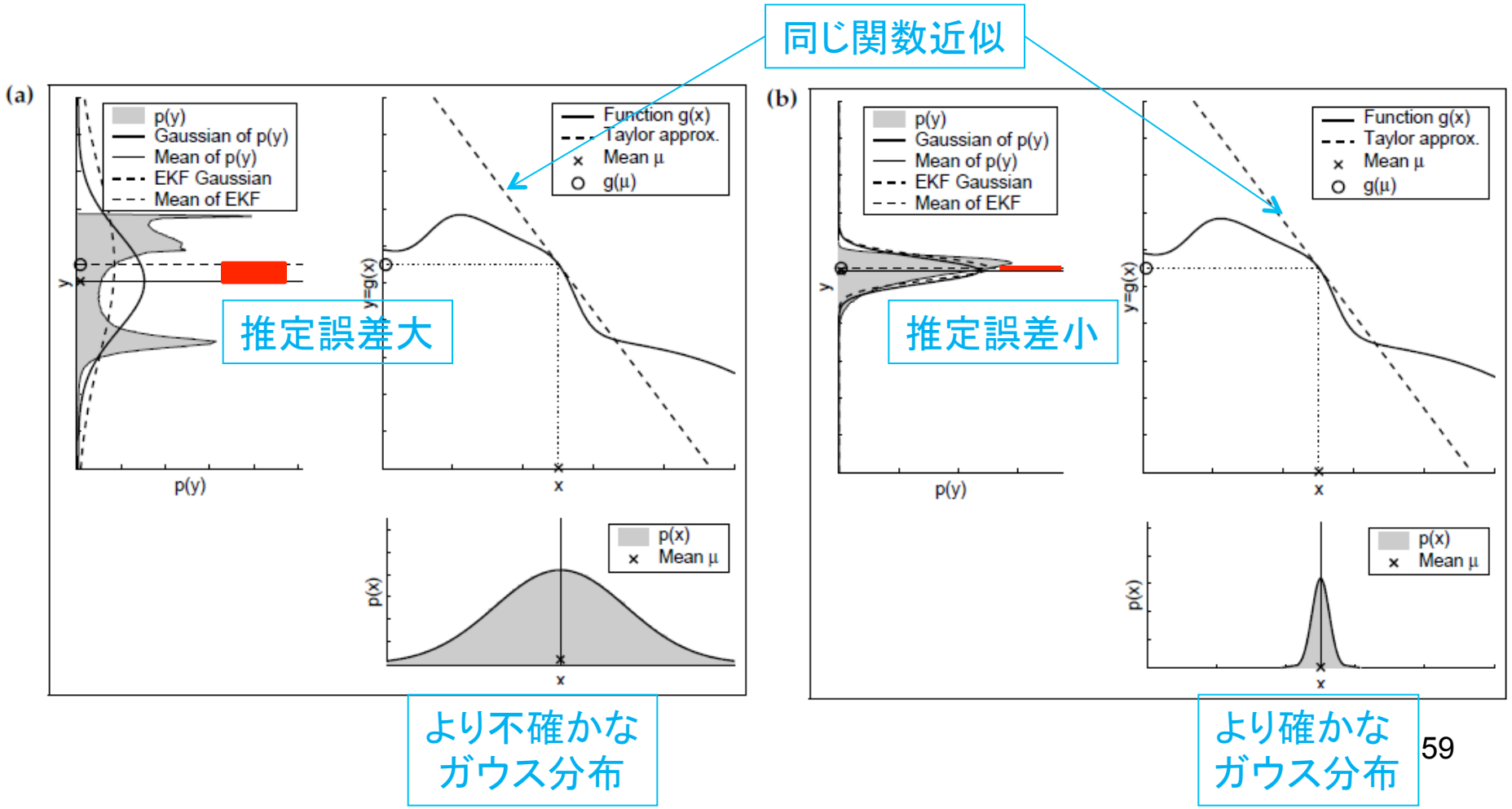
1: $\bar{\mu}_t = g(u_t, \mu_{t-1})$
2: $\bar{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + R_t$
3: $K_t = \bar{\Sigma}_t H_t^T (H_t \bar{\Sigma}_t H_t^T + Q_t)^{-1}$
4: $\mu_t = \bar{\mu}_t + K_t(z_t - h(\bar{\mu}_t))$
5: $\Sigma_t = (I - K_t H_t) \bar{\Sigma}_t$
6: return μ_t, Σ_t
7:

KFの A_t

KFの C_t

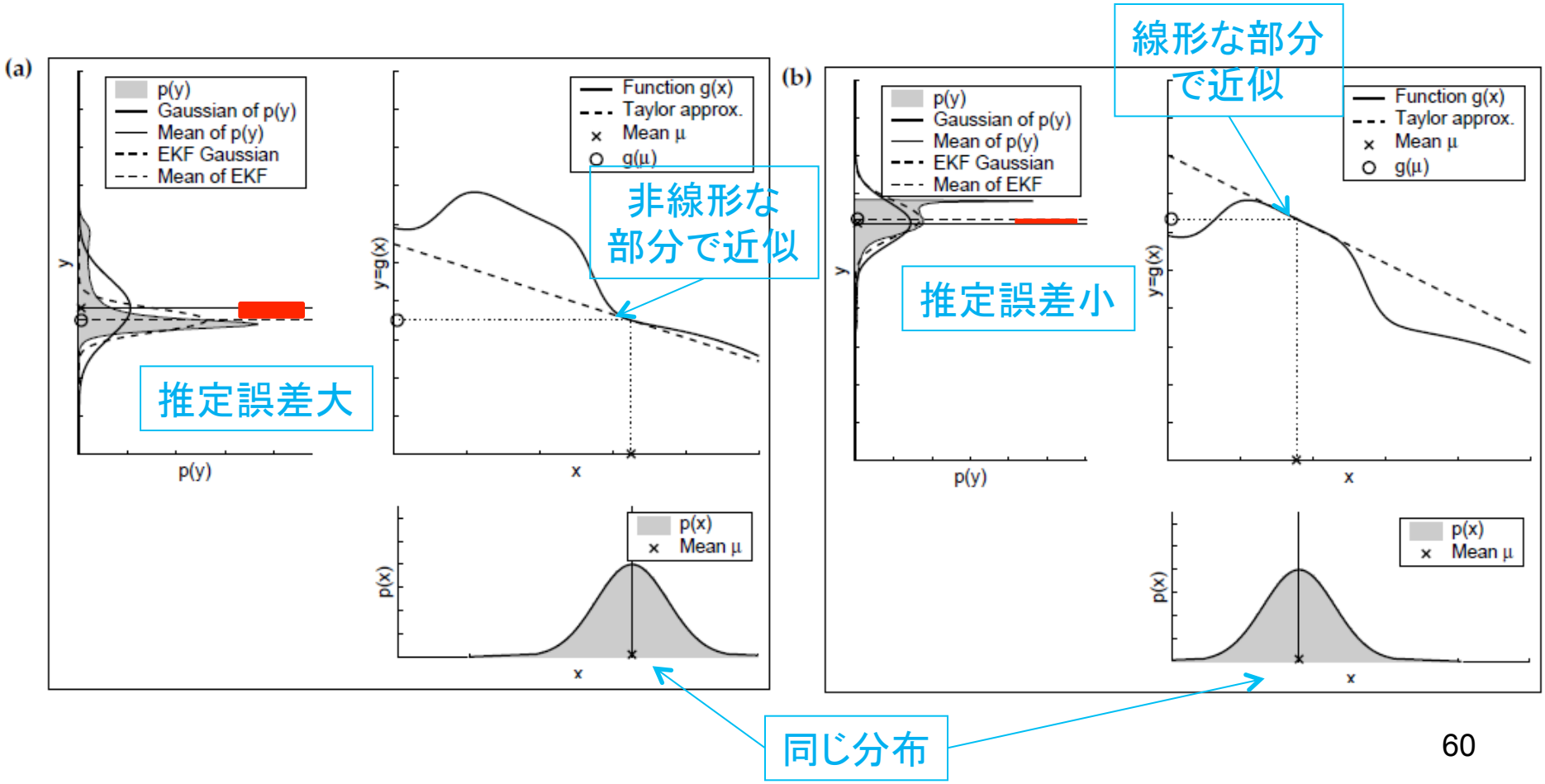
拡張カルマンフィルタ(7):注意点

– 不確かさの程度の違いで変わる近似性能



拡張カルマンフィルタ(8):注意点

– 関数 g の局所的非線形性に左右される近似性能



拡張カルマンフィルタ位置推定

EKFによる位置推定（1）

- 動作モデルと計測モデルをベイズフィルタで融合することにより、上記のそれぞれの短所を補完できる
  - ロボットの位置、動作モデル、計測モデルをそれぞれ正規分布で表す
- 動作モデル
  - 関数 g は非線形なので、テーラー展開で線形化
  - G_t は x_t に関する g のヤコビ行例、V_t は u_t に関する g のヤコビ行例
  - $\hat{u}_t$ を動作指令値（オドメトリ値）とし、u_t は正規分布 $N(\hat{u}_t, M_t)$ に従う

一変数スカラ関数の接線の傾きを、多変数ベクトル値関数に対して拡張、高次元化

変化の比率を符号付きで表現

$$x_t = g(u_t, x_{t-1}) \qquad \text{——(6)}$$

$$\approx g(\hat{u}_t, \bar{x}_{t-1},) + G_t(x_{t-1} - \bar{x}_{t-1}) + V_t(u_t - \hat{u}_t)$$

EKFによる位置推定(2)

- 計測モデル
  - h も非線形関数なので、テーラー展開で線形化
  - H_tはx_t に関するh のヤコビ行例
  - β_t は誤差で正規分布 $N(0, Q_t)$ に従う

$$z_t = h(x_t, m_j) + \beta_t$$
$$\approx h(\bar{x}_t, m_j) + H_t(x_t - \bar{x}_{t-1}) + \beta_t \quad \text{——(7)}$$

変化の比率を符号付きで表現

誤差

EKFによる位置推定(3)

- ロボットの位置の分布を$N(\bar{x}_t, \Sigma_t)$とすると、拡張カルマンフィルタにより次のように計算できる

(1) 動作モデルによるx_tの予測

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

$$\tilde{\Sigma}_t = G_t \Sigma_{t-1} G_t^T + V_t M_t V_t^T \quad \text{——(8)}$$

(2) 計測モデルによるx_tの更新

$$K_t = \tilde{\Sigma}_t H_t^T (H_t \tilde{\Sigma}_t H_t^T + Q_t)^{-1}$$

$$\bar{x}_t = \tilde{x}_t + K_t (z_t - h(\tilde{x}_t, m_j)) \quad \text{——(9)}$$

$$\Sigma_t = (I - K_t H_t) \tilde{\Sigma}_t$$

K_tはカルマンゲイン
動作モデルと計測モデルの精度に応じた重みをつけて最適値を得る
また、実測値をセンサ座標系から地図座標系に変換する

動作モデルの値

実測値

計測モデルの値

EKFによる位置推定(4)

- カルマンフィルタは、動作モデル $\bar{x}_t$ と、実測値 z_t と計測モデルの値 $h(\tilde{x}_t, m_j)$ の差とを K_t で重みづけした線形和
- カルマンゲイン K_t は、動作モデルと計測モデルの精度に応じた重みをつけて最適値を得る働きをする

カルマンフィルタにより、複数のデータを融合して
最適解を得ることができる

EKFによる位置推定(5):アルゴリズム

予測ステップ

1: **Algorithm EKF_localization_known_correspondences**$(\mu_{t-1}, \Sigma_{t-1}, u_t, z_t, c_t, m)$:

2: $\theta = \mu_{t-1,\theta}$

3: $$G_t = \begin{pmatrix} 1 & 0 & -\frac{v_t}{\omega_t}\cos\theta + \frac{v_t}{\omega_t}\cos(\theta+\omega_t\Delta t) \\ 0 & 1 & -\frac{v_t}{\omega_t}\sin\theta + \frac{v_t}{\omega_t}\sin(\theta+\omega_t\Delta t) \\ 0 & 0 & 1 \end{pmatrix}$$

4: $$V_t = \begin{pmatrix} \frac{-\sin\theta+\sin(\theta+\omega_t\Delta t)}{\omega_t} & \frac{v_t(\sin\theta-\sin(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\cos(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ \frac{\cos\theta-\cos(\theta+\omega_t\Delta t)}{\omega_t} & -\frac{v_t(\cos\theta-\cos(\theta+\omega_t\Delta t))}{\omega_t^2} + \frac{v_t\sin(\theta+\omega_t\Delta t)\Delta t}{\omega_t} \\ 0 & \Delta t \end{pmatrix}$$

5: $$M_t = \begin{pmatrix} \alpha_1 v_t^2 + \alpha_2\omega_t^2 & 0 \\ 0 & \alpha_3 v_t^2 + \alpha_4\omega_t^2 \end{pmatrix}$$

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

7: $$\bar{\Sigma}_t = G_t\,\Sigma_{t-1}\,G_t^T + V_t\,M_t\,V_t^T$$

計測更新ステップ

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

9: *for all observed features* $z_t^i = (r_t^i \;\; \phi_t^i \;\; s_t^i)^T$ *do*

10: $j = c_t^i$

11: $q = (m_{j,x} - \bar{\mu}_{t,x})^2 + (m_{j,y} - \bar{\mu}_{t,y})^2$

計測予測

12: $\hat{z}_t^i = \begin{pmatrix} \sqrt{q} \\ \text{atan2}(m_{j,y} - \bar{\mu}_{t,y}, m_{j,x} - \bar{\mu}_{t,x}) - \bar{\mu}_{t,\theta} \\ m_{j,s} \end{pmatrix}$

13: $H_t^i = \begin{pmatrix} -\frac{m_{j,x} - \bar{\mu}_{t,x}}{\sqrt{q}} & -\frac{m_{j,y} - \bar{\mu}_{t,y}}{\sqrt{q}} & 0 \\ \frac{m_{j,y} - \bar{\mu}_{t,y}}{q} & -\frac{m_{j,x} - \bar{\mu}_{t,x}}{q} & -1 \\ 0 & 0 & 0 \end{pmatrix}$

14: $S_t^i = H_t^i \; \bar{\Sigma}_t \; [H_t^i]^T + Q_t$

15: $K_t^i = \bar{\Sigma}_t \; [H_t^i]^T [S_t^i]^{-1}$

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

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

推定更新

18: *endfor*

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

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

21: $p_{z_t} = \prod_i \det\left(2\pi S_t^i\right)^{-\frac{1}{2}} \exp\left\{-\frac{1}{2} \; (z_t^i - \hat{z}_t^i)^T [S_t^i]^{-1} (z_t^i - \hat{z}_t^i)\right\}$

22: *return* μ_t, Σ_t, p_{z_t}

EKFによる位置推定(6):詳細

1行目:ロボットの姿勢推定のガウス分布(μ_{t-1}, Σ_{t-1})、制御u_t、地図m、計測z_tと対応変数c_t

22行目:更新された推定μ_t, Σ_t、特徴観測の尤度p_{z_t}

3～4行目:動作モデルを線形化(ヤコビ行列計算)

5行目:制御から、動作に生じる雑音の共分散特定

6～7行目:動作更新(6行目:予測される姿勢$\bar{\mu}_t$、7行目:それに対応する誤差楕円)

9～18行目:時刻tに観測されたすべての特徴iに対するループ

EKFによる位置推定(7):詳細

10行目:計測ベクトル中における i 番目の特徴の対応変数 j に代入

12行目:予測される計測 $\hat{z}_t^i$ を計算

13行目:計測モデルのヤコビ行列 H_t^iを計算

14行目:予測される計測 $\hat{z}_t^i$ の不確かさ S_t^i を計算

16～17行目:推定が一つの特徴に対して一回ずつ更新

19～20行目:新たな推定姿勢の計算

21行目:計測の尤度を計算

地図構築

地図構築(1)

- 地図構築は自己位置推定と分離してできず、SLAM(Simultaneous Localization and Mapping)という枠組みが提案された
  - 計測データはロボットからの相対値で取得 ⇒ 座標変換し、地図座標系でのランドマーク位置を求め、地図を構築
  - しかし、ランドマークを載せた地図はこれから作られる
  - ゆえに、ロボット位置と地図は同時に推定しなければならない ⇒ SLAMを適用

地図構築(2)

- SLAM
  - ポイント: 同じランドマークを複数回観察すること
    - これまで観測したランドマークを使ってロボットの位置推定
      ⇒ そのロボットの位置に従って新たなランドマークを配置
      ⇒ 地図を成長させる
  - SLAMにもベイズフィルタが使われる
    - ロボット位置と地図の同時確率密度の推定問題 ⇒ 状態ベクトルの次元は非常に大きくなる

カルマンフィルタを用いたSLAM

位置と地図の同時推定

- ロボット位置 r_t と地図 m の同時確率密度を正規分布で表し、自己位置推定と同様に漸化式を構成する

$$p(r_t, m \mid z_{1:t}, u_{1:t}) = \alpha\, p(z_t \mid r_t, m) \times \int p(r_t \mid r_{t-1}, u_t)\, p(r_{t-1}, m \mid z_{1:t-1}, u_{1:t-1})\, dr_{t-1}$$

カルマンフィルタアルゴリズム参照

- いま $u_t = (r_t, m_1, \ldots, m_n)$ とすると、求める同時確率は $p(a_t \mid z_{1:t}, u_{1:t})$ となり、自己位置推定と同様に拡張カルマンフィルタで実装できる
  - 動作モデル は r_t に対して計算するが、計測モデルは u_t に対して計算する（次元が大きいので、計算量は増える）

フィルタアプローチの特徴

- 自己位置推定のように状態ベクトルの次元が小さい場合 ⇒ ベイズフィルタは実用性が高く有効
- 地図構築は状態ベクトルの次元が膨大（数千～数百万）⇒ ベイズフィルタの適用は困難が伴う
- 最適化アプローチによる地図構築では、動作モデルと計測モデルから評価関数を構成し、その最適解を求めることにより、ロボット位置とランドマーク位置を同時に求める ⇒ 地図構築に有効

まとめ

- 移動ロボットの確率的な自己位置推定と地図構築の方法
  - ベイズフィルタによるアプローチが有効
  - 利点
    - データ融合を統一的に扱えること
    - ロボット位置などの不確実性を表せること

おわり