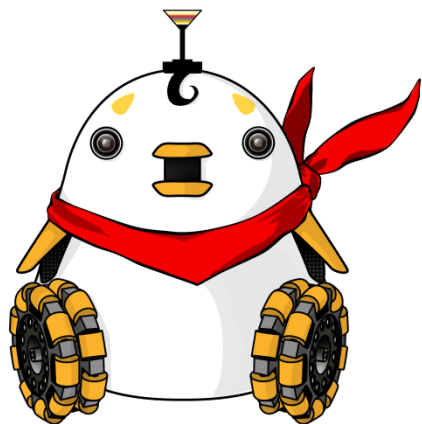


人工知能

第10章 状態推定(2)

粒子フィルタ



STORY 粒子フィルタ

- ベイズフィルタを実装されたホイールダック2号は思った。「何だか、これ、面倒くさくないだろうか?」。ベイズフィルタでは常に「自分があらゆる状態にいる確率」を考えなければならない。ところが、迷路は広いものの、自分がいる場所は1箇所だし、自分がいる可能性のある場所も正直なところ限られている。ほとんどの場所で自分の存在確率はほぼ0だ。
- こんな無駄な情報を持っているより、むしろ、「僕はここにいるかもしれない」という仮説をいくつか持っておいた方がいいのではないだろうか。粒子のように自分のいそうな場所の仮説を持ち、それらを更新していくことで効率的な状態推定をする。このアイデアを実現するのが粒子フィルタだったのだ。



仮定 粒子フィルタ

- ホイールダック2号は迷路の完全な地図を持っているものとする.
- ホイールダック2号は自分がどこにいればどんな観測が得られるか知っているものとする (ただし確率的に) .
- ホイールダック2号はそれぞれの状態で自分がどんな行動をとれば, どの状態へ移動するのかを知っているものとする (ただし確率的に) .



Contents

- ▣10.1 ベイズフィルタの問題点
- ▣10.2 モンテカルロ近似
- ▣10.3 粒子フィルタ
- ▣10.4 通路上のホイールダック 2 号の位置推定
(粒子フィルタ編)
- ▣10.5 SLAM: 自己位置と地図の同時推定*

10.1.1位置推定とメモリ管理

- 前章であつかったベイズフィルタでは、実装上の非効率な点がある。それは「全ての状態に関する存在確率を常に保持しなければならない」という点である。
- 膨大な状態空間が存在する場合には、ロボットが絶対に存在しないであろう場所における存在確率も含めて、その状態空間全てにおいて存在確率 $P(s_t | o_{1:t}, a_{1:t-1})$ の更新を行う必要がある。これは非効率である。

確率がゼロのところを陽に表現しない
効率的データ表現を！

Monte Carlo Localization (MCL) Particle Filter

だいたいこのへんにいる . . .

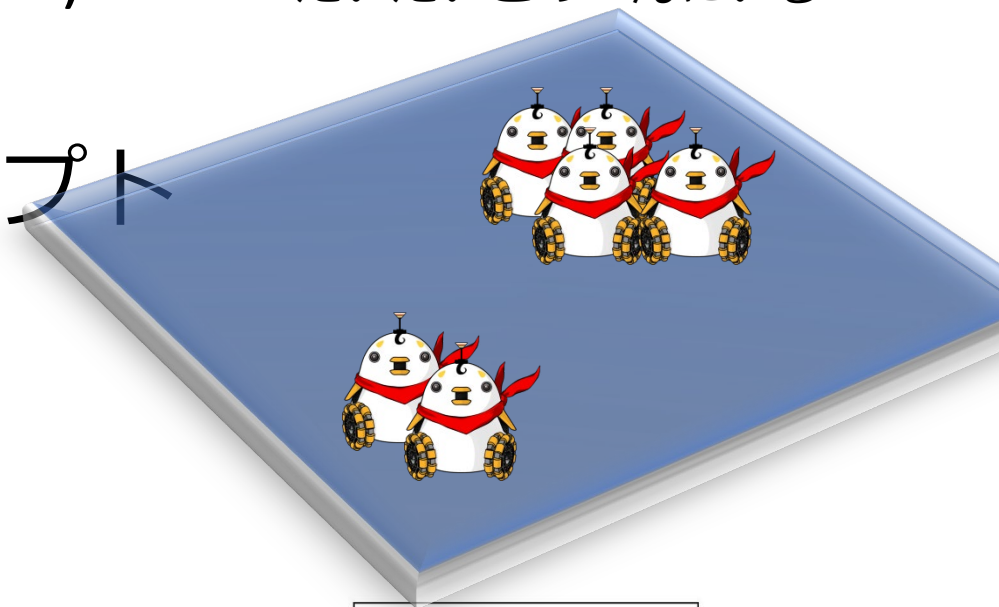
粒子フィルタのコンセプト

□これに対して、「ロボットはここにいるかもしれない」という仮説（候補）をいくつか持って、これを更新することで、自己位置推定を進めるのが**粒子フィルタ**

粒子フィルタ

= モンテカルロ近似

+ SIR



(1) 確率分布による表現

	1	2	3	4
1	0.10	0.01	0.01	0.01
2	0.02	0.01	0.01	0.22
3	0.30	0.01	0.01	0.22
4	0.04	0.01	0.01	0.01



(2) 候補の集合による表現

	1	2	3	4
1				
2				
3				
4				

	1	2	3	4
1				
2				
3				
4				

.....

	1	2	3	4
1				
2				
3				
4				

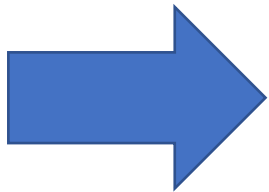
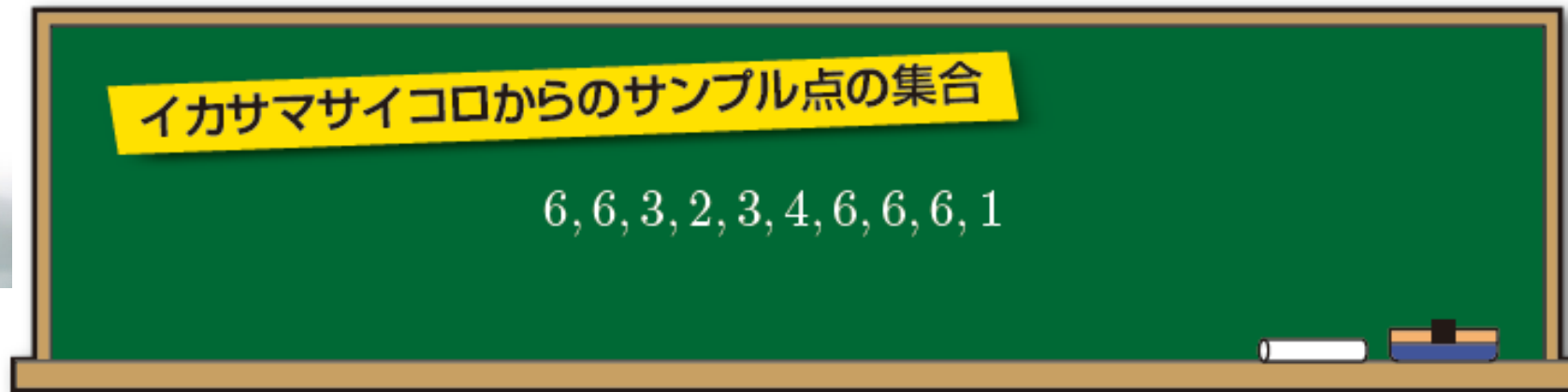
図 10.2 自己位置の確率分布による表現と候補の集合による表現

Contents

- 10.1 ベイズフィルタの問題点
- 10.2 モンテカルロ近似
- 10.3 粒子フィルタ
- 10.4 通路上のホイールダック 2 号の位置推定
(粒子フィルタ編)
- 10.5 SLAM: 自己位置と地図の同時推定*

10.2.1 サンプル点の集合による確率分布の近似

- 「『自らがいそうな状態』に関する候補をいくつか持つことで『すべての状態』についての情報を持つ代わりにする」



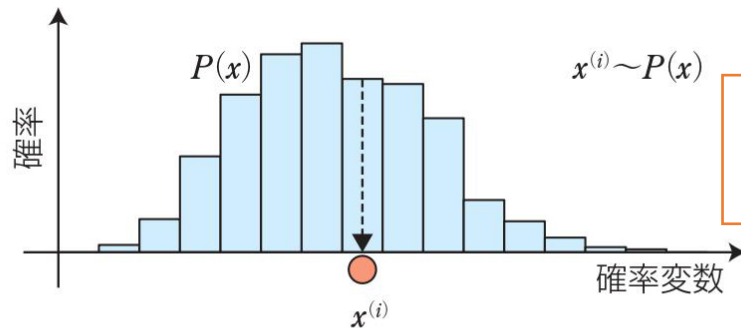
なんとなく裏に潜むイカサマサイコロの確率分布を想像できる！

サンプル点の集合を確率分布の近似として扱う

信号処理のサンプリングとは別物だから気をつけましょう。
統計学の「標本調査」の標本をサンプルと呼ぶのと同じ

“サンプリング”とは？

- 確率分布から具体的な値を抽出（サンプリング）すること。
- サイコロを振って値を出すことに相当。
- 機械学習等の分野では「確率分布からdrawする（引き出す，振り出す）」のような表現を使う。
- 例）確率分布 $P(x)$ から i 番目のサンプル $x^{(i)}$ をdrawする。



$$x^{(i)} \sim P(x)$$

確率分布からdrawする記号

図 10.3 確率分布 $P(x)$ からのサンプリング

※イメージとしてはC言語における `int x = rand()` をイメージするのがよい。

10.2.3 モンテカルロ法

□モンテカルロ法は一般的に、確率分布の式を直接扱うかわりに、その確率分布から生成されたサンプル群によって、その確率分布の代用とする方法である。

□期待値の評価によく用いられる。 $E(x) = \sum_x xP(x) \simeq \frac{1}{N} \sum_{i=1}^N x^{(i)}$

□より一般的に確率変数 x についての関数値 $f(x)$ の期待値を評価する。

$$E[f] = \sum_x f(x)P(x) \simeq \frac{1}{N} \sum_{i=1}^N f(x^{(i)})$$

※ $x^{(i)}$ は確率分布 $P(x)$ からサンプリングされる i 番目のサンプル値

10.2.2 モンテカルロ近似

□モンテカルロ法が前提にしているのは、 N 点のサンプル集合が元々の確率分布のよい近似になっているという性質である。

$$P(x) \simeq \frac{1}{N} \sum_{i=1}^N \delta(x, x^{(i)})$$

□として、 N 個のサンプル点によって確率分布を近似する。 δ はクロネッカーのデルタである。

$$\delta(x, x^{(i)}) = \begin{cases} 1 & (x = x^{(i)}) \\ 0 & (otherwise) \end{cases}$$

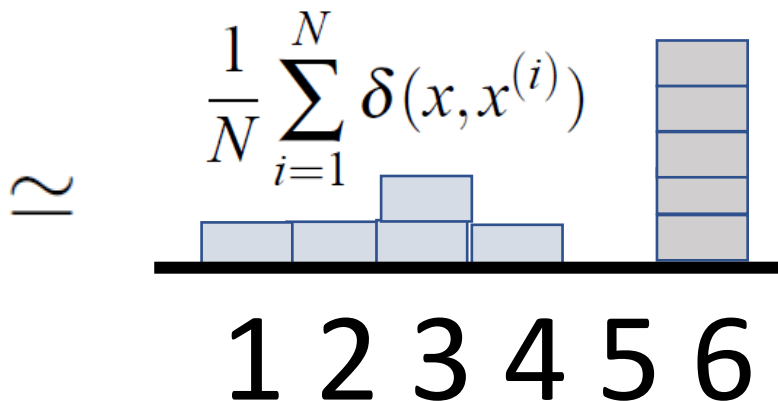
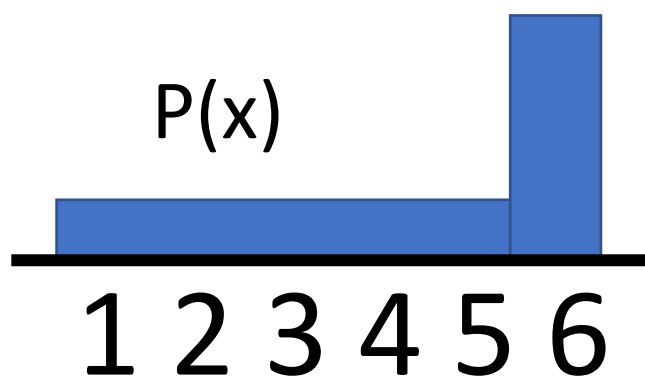
イカサマサイコロのモンテカルロ近似

\simeq

は近似のマークです.

$$P(x) \simeq \frac{1}{N} \sum_{i=1}^N \delta(x, x^{(i)})$$

- 例えば半分の確率で6がでるイカサマサイコロを10回振る.
- $x^{(i)} = \{6, 6, 3, 2, 3, 4, 6, 6, 6, 1\}$ と出たとする.



演習10-1 モンテカルロ法

- あるテストについてクラスの平均点を調べようと100 人のクラスでランダムに10 人から聞き取り調査をした．すると，10 人の回答の平均値は50 点であった．モンテカルロ法に基づいてこのクラスの平均点を求めよ．

Contents

- 10.1 ベイズフィルタの問題点
- 10.2 モンテカルロ近似
- 10.3 粒子フィルタ
- 10.4 通路上のホイールダック 2 号の位置推定
(粒子フィルタ編)
- 10.5 SLAM: 自己位置と地図の同時推定*

10.3 Sampling Importance Resampling (SIR)

- モンテカルロ近似を位置推定のベイズフィルタアップデートに用いる上で、さらに加えられた近似手法がSampling Importance Resampling (SIR) である.
- 非常に巧妙な手法であり、ベイズ更新自体にモンテカルロ近似を適用することによって、アルゴリズム上は常に、仮説となるサンプル点群を持てばいいことになり、非常に実装しやすいアルゴリズムとなる.

Sampling

サンプリングする

Importance

重み付けする

Resampling

もう一回
サンプリングする

第9章の導出を必ず復習！

SIRの導出(1)

- 第9章のベイズフィルタ

$$F_t(s_t) \propto P(o_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}, a_{t-1}) F_{t-1}(s_{t-1}) = G_t(s_t)$$

- 事後分布のモンテカルロ近似

$$F_t(s_t) \simeq \frac{1}{N} \sum_i \delta(s_t, s_t^{(i)})$$

モンテカルロ近似のもとになる
一個一個のサンプルのことを
粒子と呼ぶ.

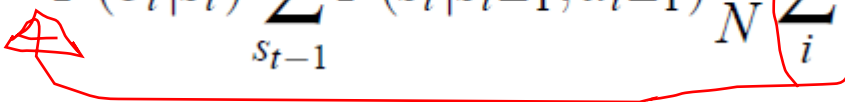
- 右辺のFへの適用


$$F_t(s_t) \propto P(o_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}, a_{t-1}) \frac{1}{N} \sum_i \delta(s_{t-1}, s_{t-1}^{(i)})$$

これじゃあ、ベイズフィルタの売りの漸化式にはならない！？

SIRの導出(2)


粒子iごとに式を分解してみる.

$$F_t(s_t) \propto P(o_t|s_t) \sum_{s_{t-1}} P(s_t|s_{t-1}, a_{t-1}) \frac{1}{N} \sum_i \delta(s_{t-1}, s_{t-1}^{(i)})$$


$$F_t(s_t) \simeq \frac{1}{N} \sum_i F_t^{(i)}(s_t)$$


(Point)

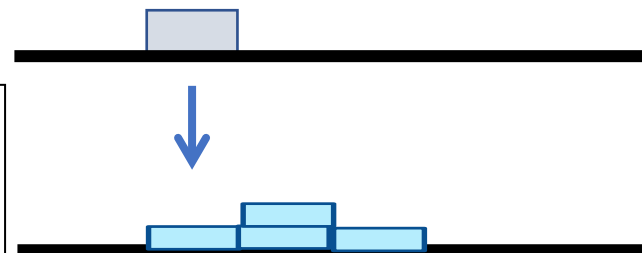
シグマの順番を変える！

$$F_t^{(i)}(s_t) \propto P(o_t|s_t) \bar{P}^{(i)}(s_t)$$


$$\bar{P}^{(i)}(s_t) = \sum_{s_{t-1}} P(s_t|s_{t-1}, a_{t-1}) \delta(s_{t-1}, s_{t-1}^{(i)})$$

分解した式の解釈

$$\bar{P}^{(i)}(s_t) = \sum_{s_{t-1}} P(s_t | s_{t-1}, a_{t-1}) \delta(s_{t-1}, s_{t-1}^{(i)})$$



t-1時刻の粒子iが状態遷移したものの確率分布

$$F_t^{(i)}(s_t) \propto P(o_t | s_t) \bar{P}^{(i)}(s_t)$$

粒子iが状態遷移後，観測を得て重み付けられたもの

$$F_t(s_t) \simeq \frac{1}{N} \sum_i F_t^{(i)}(s_t)$$

全ての粒子についての和

SIRの導出(3): 状態遷移後の確率分布の近似

$\bar{P}^{(i)}$ については, これを $s_{t-1}^{(i)}$ から出た一つのサンプルで近似する:

$$F_t(s_t) \simeq \frac{1}{N} \sum_i F_t^{(i)}(s_t)$$

$$\propto \frac{1}{N} \sum_i P(o_t | s_t) \bar{P}^{(i)}(s_t)$$

$$\simeq \frac{1}{N} \sum_i P(o_t | s_t) \delta(s_t, \bar{s}_t^{(i)})$$

$$= \frac{1}{N} \sum_i \underbrace{P(o_t | \bar{s}_t^{(i)})}_{\text{センサ情報による重み}} \underbrace{\delta(s_t, \bar{s}_t^{(i)})}_{\text{状態遷移確率に従い移動してきたサンプル点}}$$

$$= \frac{1}{N} \sum_i w_i \delta(s_t, \bar{s}_t^{(i)})$$

w_i をサンプル点の“重み”と見る.

$$\bar{s}_t^{(i)} \sim \bar{P}^{(i)}(s_t)$$

$$\bar{P}^{(i)}(s_t) \simeq \delta(s_t, \bar{s}_t^{(i)})$$

思い切ったモンテカルロ近似

以上により F_t が重み付けられた粒子群の和として表される.

SIRの導出(4) resampling!!!

$$F_t(s_t) \simeq \frac{1}{N} \sum_i w_i \delta(s_t, s_t^{(i)})$$

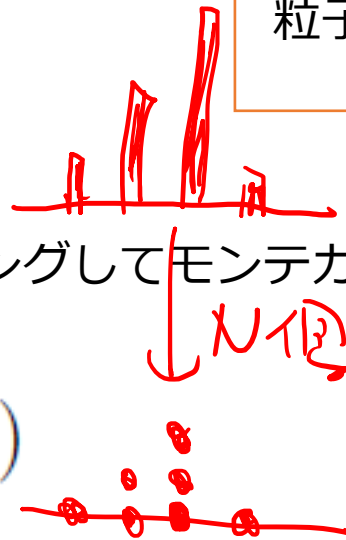


$$F_t(s_t) \simeq \frac{1}{N} \sum_i \delta(s_t, s_t^{(i)})$$

$s_t^{(i)}$ をサンプリングしてモンテカルロ近似

↓ N個のサンプル

N個の「重み付き」
粒子（サンプル）



また，粒子群になった！アルゴリズムミックな更新式が得られる！

- **リサンプリング**(resampling) するというアイデアにより，SIR では粒子群の効率的な更新のアルゴリズムを得ている．
- サンプル点の集合は粒子群のように振る舞うため，各サンプル点のことを**粒子**(particle) と呼ぶ．

Algorithm 10.1 粒子フィルタ

- ① 粒子の分布を初期化させる $t \leftarrow 1$
- ② repeat
- ③ ロボットの行動 a_{t-1} に従い, 粒子 $s_{t-1}^{(i)}$ ($1 \leq i \leq N$) ごとに次状態 $s_t^{(i)}$ を状態遷移確率を用いてサンプリングする.

$$\bar{s}_t^{(i)} \sim P(s_t | s_{t-1} = s_{t-1}^{(i)}, a_{t-1}) \quad (10.20)$$

- ④ o_t を観測し, 各粒子についてセンサ情報の観測確率 $w_i = P(o_t | \bar{s}_t^{(i)})$ を計算し, それぞれの粒子の重みとする.
- ⑤ 粒子の重みの比率 $\frac{w_i}{\sum_j w_j}$ に従って, 粒子をリサンプリングする. これらを $s_t^{(k)}$ ($1 \leq k \leq N$) とする.

$$s_t^{(k)} \sim \sum_i w_i \delta(s_t, \bar{s}_t^{(i)}) \quad (10.21)$$

- ⑥ $t \leftarrow t + 1$
- ⑦ until 停止させられる.

簡単な実装！
メモリ使用量も制御容易！

Contents

- 10.1 ベイズフィルタの問題点
- 10.2 モンテカルロ近似
- 10.3 粒子フィルタ
- 10.4 通路上のホイールダック 2 号の位置推定
(粒子フィルタ編)
- 10.5 SLAM: 自己位置と地図の同時推定*

10.4 例：通路上のホイールダック2号 (粒子フィルタ編)

- 8章と同じ問題を考える.
- ベイズフィルタと異なり、ロボットの位置についての確率分布を粒子の集合で表現される.
- 移動は80%の確率で成功する.
- 70%の確率で正しい観測が得られるが、誤認識が発生した場合、それはそれぞれ2%の確率で、その中から15個の選択肢の中から誤った観測が得られるものとする.

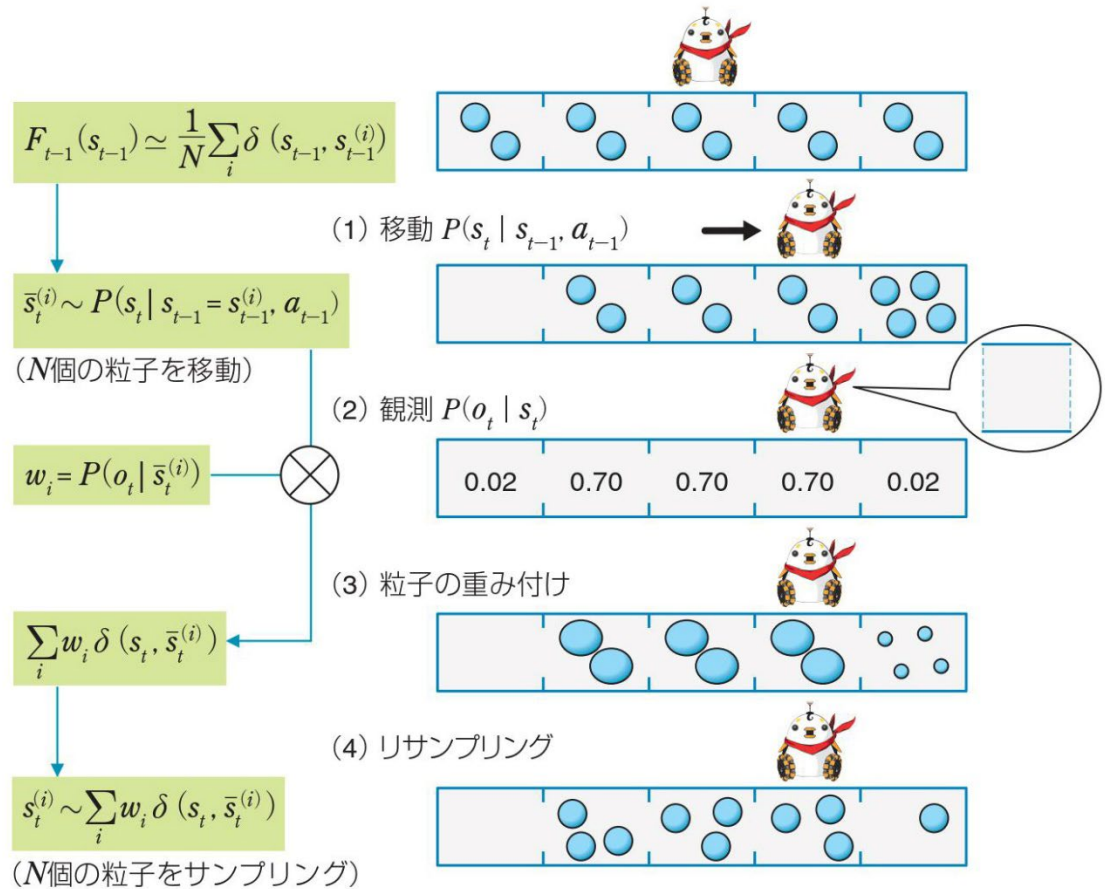


図 10.5 ホイールダック 2 号の位置推定 1 ステップ目

続き

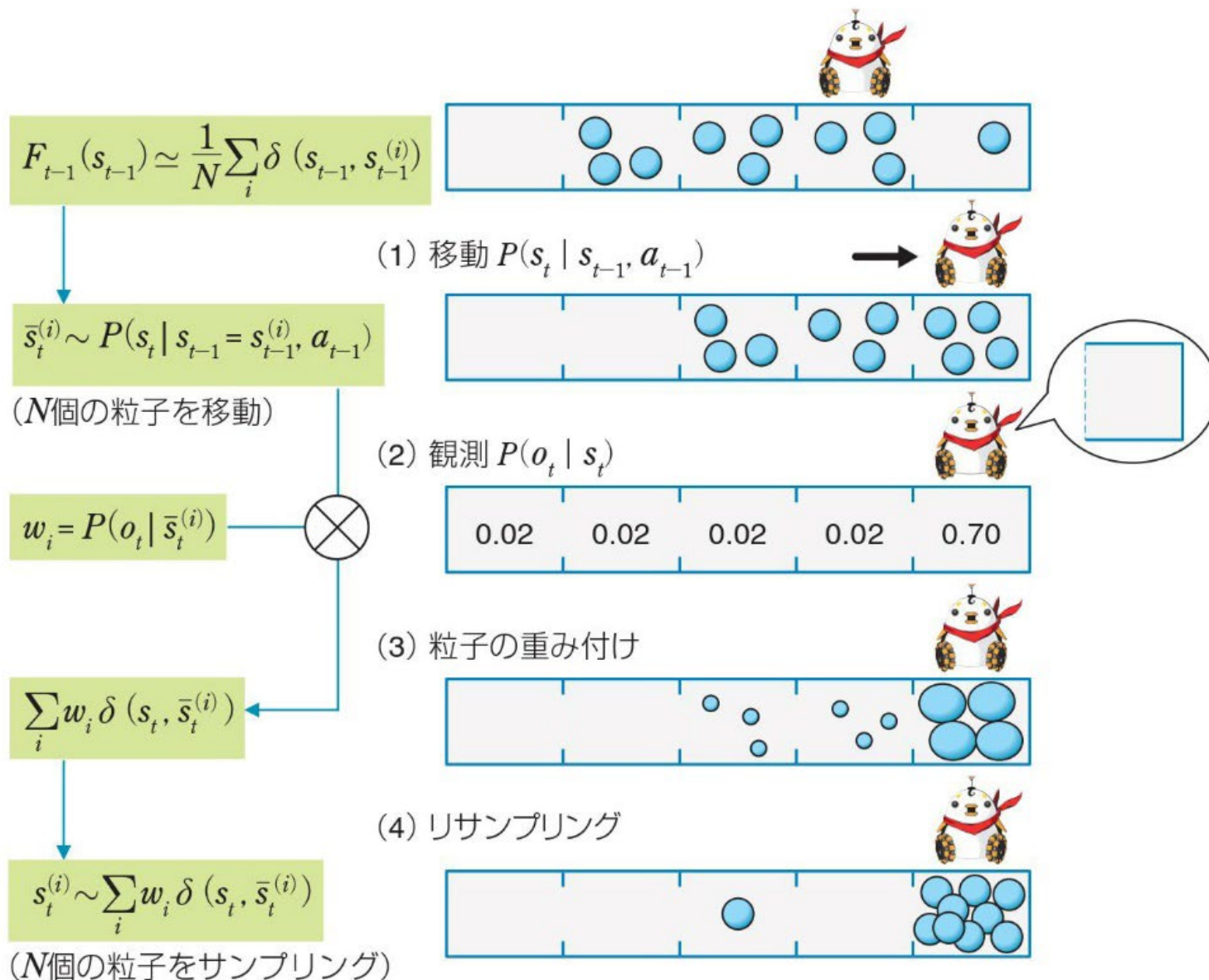


図 10.6

ホイールダック 2 号の位置推定 2 ステップ目

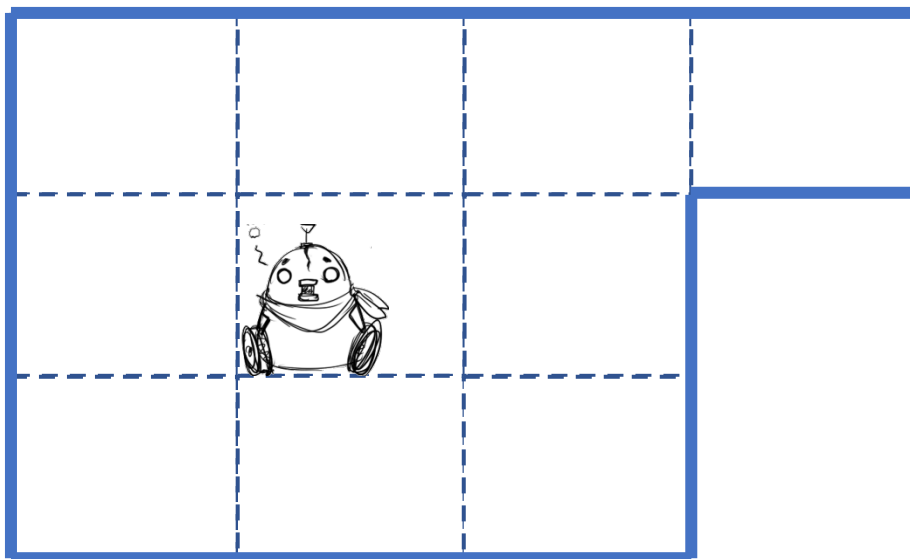
10.4.2 粒子フィルタの応用

- 移動ロボットの自己位置推定には粒子フィルタはMCL (Monte Carlo Localization, モンテカルロ位置推定) と呼ばれ、大変よく用いられている方法である.
- 実際には、連続の確率システムの状態方程式に置き換え、確率分布は離散分布ではなく、システムノイズにガウス分布を仮定することが多い.
- また、コンピュータビジョンの研究では、古くから物体追跡（オブジェクト・トラッキング）に粒子フィルタが用いられている.

(動画の例) Monte Carlo Localization demo. (Youtubeより)

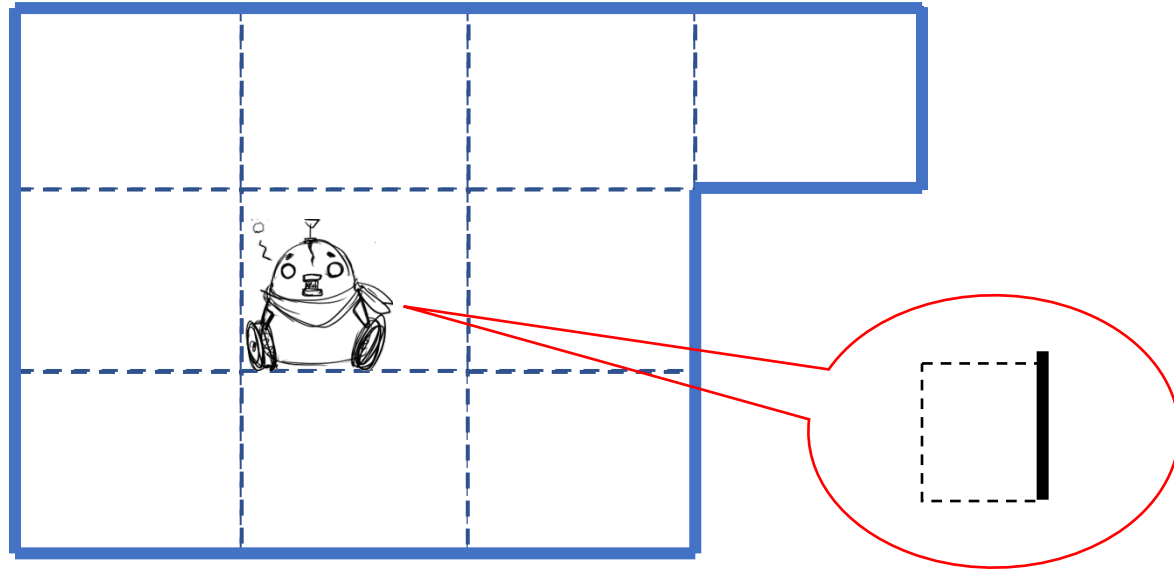
<https://www.youtube.com/watch?v=10tvdmZ7OqU>

演習10-2



ホイールダック2号はスタート時**無情報**である.
それぞれのマスにある粒子 (パーティクル) の分布の一例を
上記のセル内に書け. 粒子の合計数は20とする.

演習10-3



演習10-2の状況の後にホイールダック2号が「停止行動」をとった後に観測を行ったところ右のような観測を得た．この観測を得た後に，ホイールダック2号がいる場所をリサンプリングした結果の粒子数の一例を各セルに書け（乱数は適宜，各自で生成するものとする．）

Contents

- 10.1 ベイズフィルタの問題点
- 10.2 モンテカルロ近似
- 10.3 粒子フィルタ
- 10.4 通路上のホイールダック 2 号の位置推定
(粒子フィルタ編)
- 10.5 SLAM: 自己位置と地図の同時推定*

10.5 SLAM: 自己位置と地図の同時推定*

- 地図作成: 観測 o_t を得るための情報が地図 m でありこれを推定するのが地図作成。
- SLAM: 自己位置の状態系列 $s_{1:t}$ と地図 m の両方を繰り返し更新し同時に推定することで、自己位置と地図を同時に推定する手法。移動ロボットや自動運転技術の基礎となる。

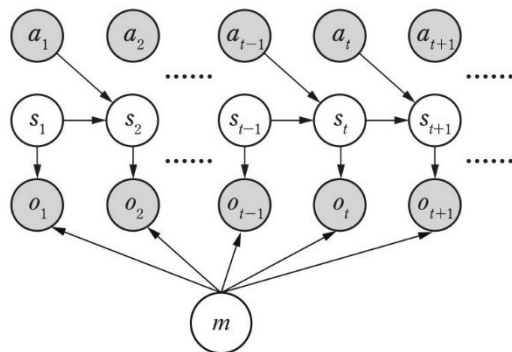


図 10.7 SLAM における部分観測マルコフ決定過程のグラフィカルモデル



13 S. Thrun, W. Burgard, D. Fox (著), 上田隆一 (訳): 確率ロボティクス (プレミアムブックス版), マイナビ出版, 2016.

まとめ

- 位置推定の問題においてベイズフィルタが持つ問題点をメモリ管理と関連して理解した.
- モンテカルロ法とモンテカルロ近似について学んだ.
- SIR のアルゴリズムを数学的に導出し, その近似の仕組みについて理解した.
- 粒子フィルタのアルゴリズムについて学んだ.
- 例を通して粒子フィルタの実行時の基本的な手続きについて確認した.
- SLAM とはどのような問題であるかを学んだ.