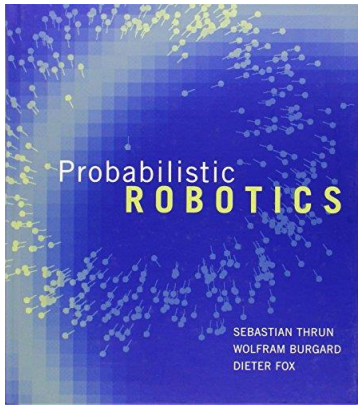
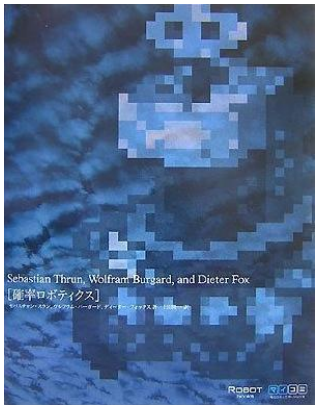
ロボットの計画と制御
マルコフ決定過程

確率ロボティクス　14章





http://www.probabilistic-robotics.org/

14.1　動機付け　ロボットの行動選択のための確率的なアルゴリズム

目的

予想される不確かさを最小化したい.

ロボットの動作につての不確かさ（MDPで考える）

決定論的な要素

ロボット工学の理論の多くは，動作の影響は決定論的であるという仮定のもとに成り立っている．しかしこれだけでは説明できない事象が混入するのが通常.

統計的な要素

ロボットや周囲の環境の確率論的な性質から生ずる.

ロボットの知覚についての不確かさ（MDP→POMDPへ）

完全可観測な系　vs. 部分観測可能な系

14.2　ロボットの行動選択における不確かさ

ロボットはどちらの経路を進むのが良いだろうか。

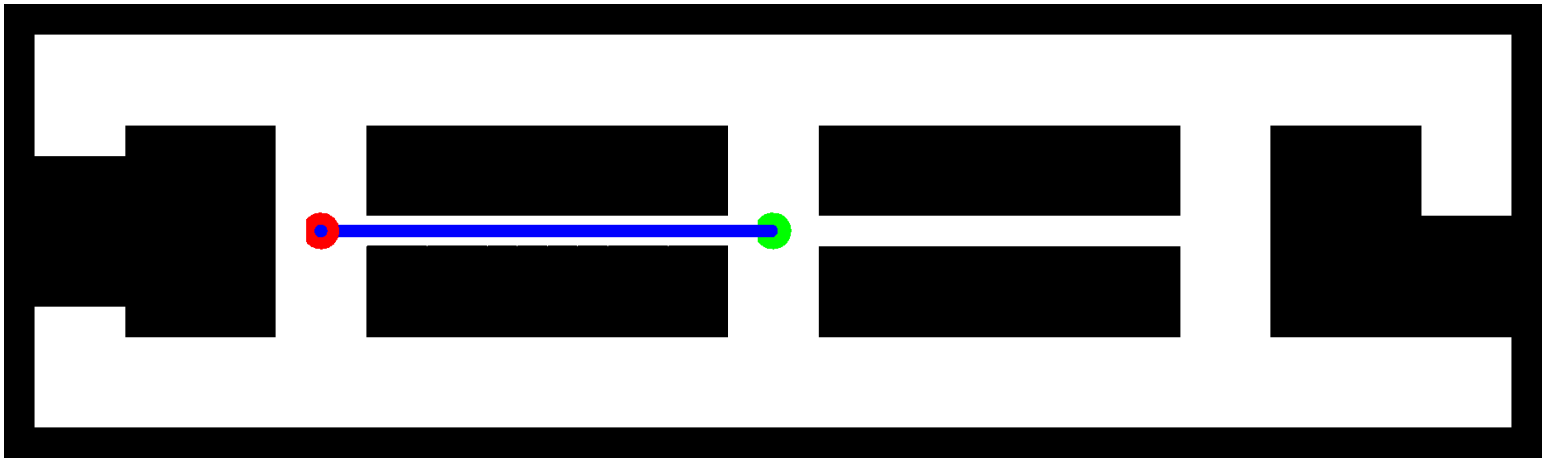
古典的な制御手法では①が選ばれる.

狭い経路を通ると，途中で壁をこするかもしれない。
ロボットは壁を検知するためにスピードを落とさなければならないかもしれない。

②を選ぶ方法はどのように構成すればよいだろう？

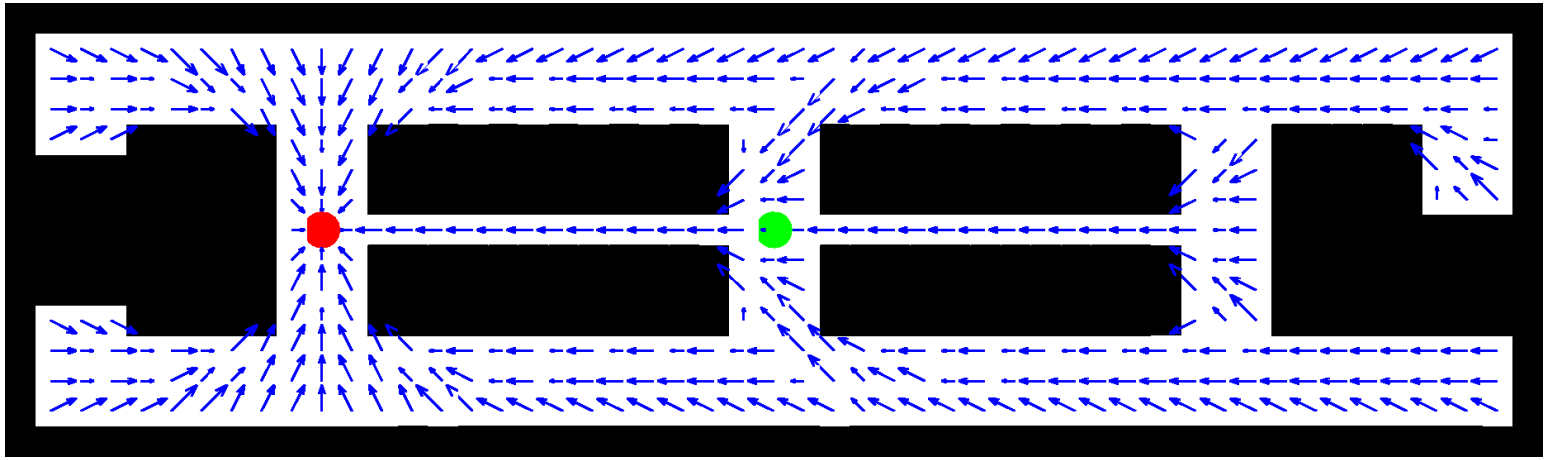
http://www.probabilistic-robotics.org/

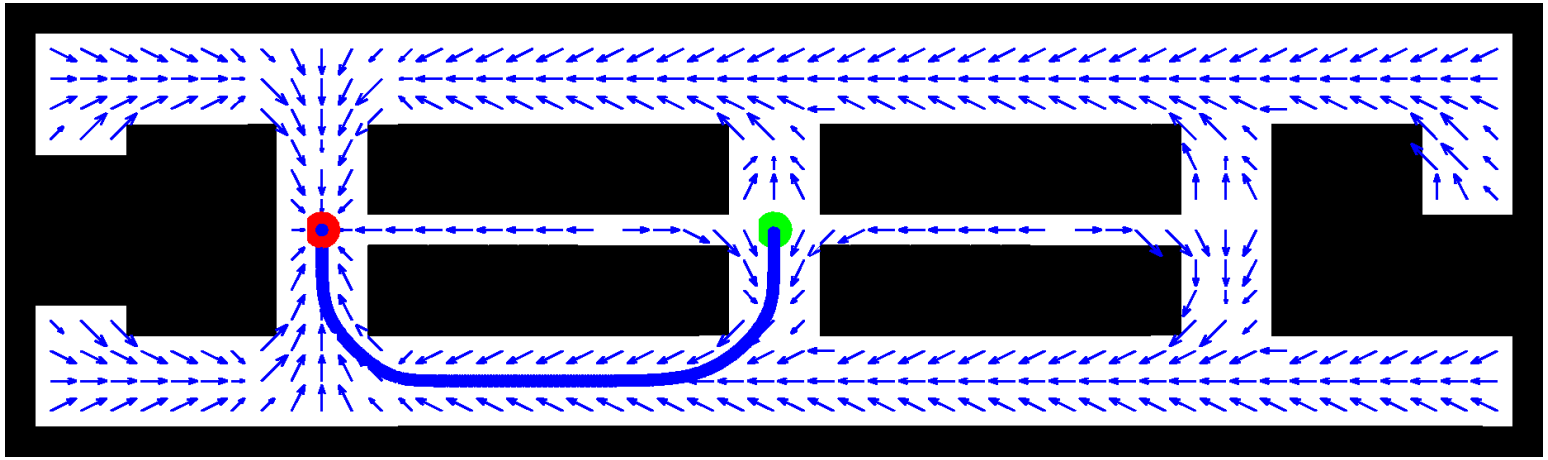
Deterministic, fully observable



http://www.probabilistic-robotics.org/

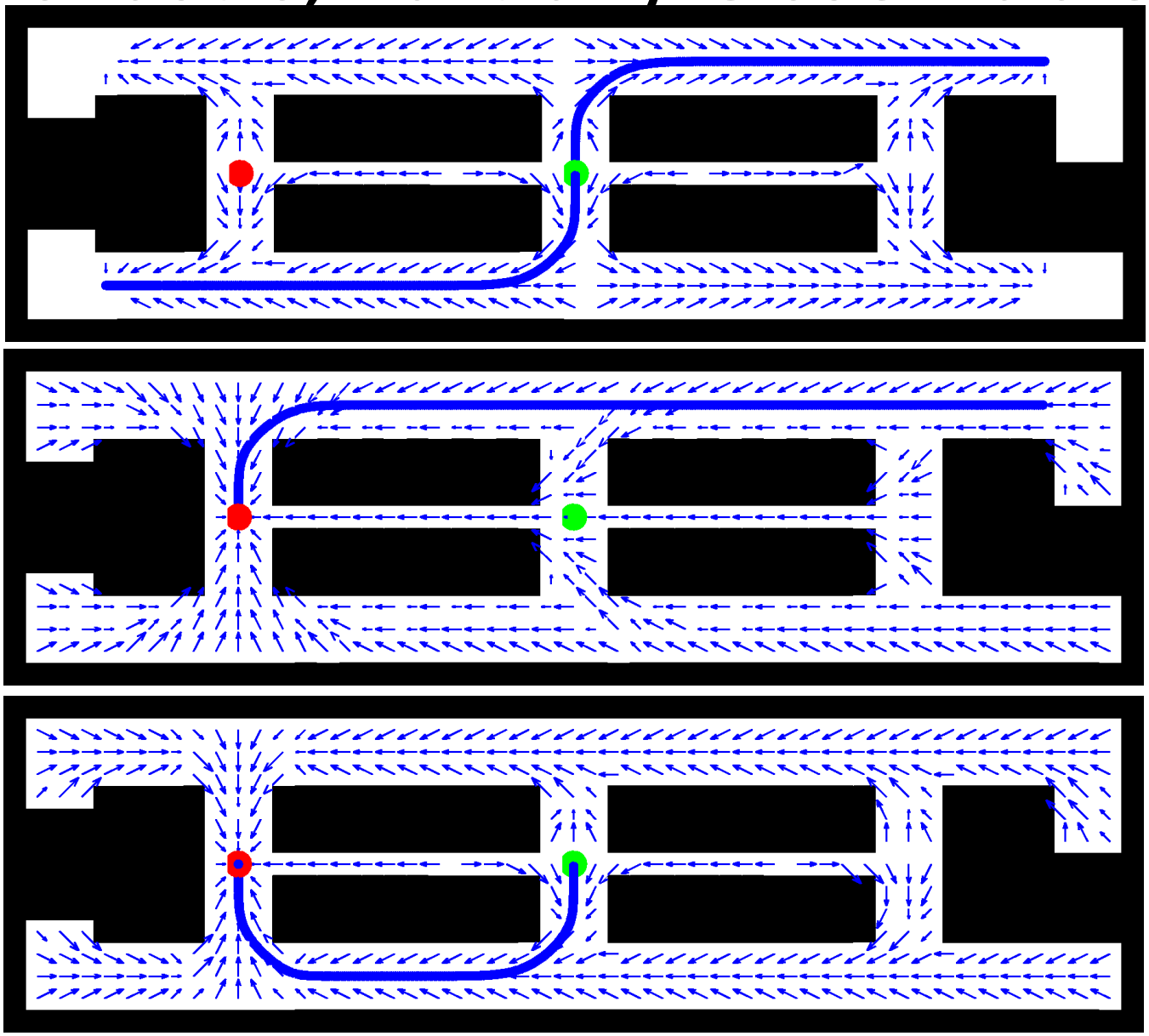
Stochastic, Fully Observable





http://www.probabilistic-robotics.org/

Stochastic, Partially Observable



マルコフ決定過程（MDPs: Markov decision processes）

ロボットの動作に関する不確かさのみを考慮する枠組み.
任意の時間において，環境の状態が完全に計測できることを仮定する.

$p(\boldsymbol{y}\|\boldsymbol{x})$	知覚モデルは決定論的で全単射を仮定
$p(\boldsymbol{x}_k\|\boldsymbol{u}_{k-1}, \boldsymbol{x}_{k-1})$	行動モデルは非決定論的. 単一の行動シーケンスを計画するのでは不十分であるので，複数の行動シーケンスを計画可能でなければならない.

ロボットが出あうすべての状態に対して
行動選択のための方策を生成する？

制御方策・ユニバーサルプラン・ナビゲーション関数　など

部分観測マルコフ決定過程（POMDPs: partially observable Markov decision processes）

ロボットモーションにおける不確かさを考慮する枠組み

$p(\boldsymbol{y}\|\boldsymbol{x})$	実用において計測には雑音が混入する
信念空間（情報空間）	ロボットが環境内で持つ可能性のある信念※すべてから構成される.

POMDPsでは信念空間において行動を割り当てる.
しかし計算複雑性の問題がつきまとう.

※信念：ベイズ統計の用語。ベイズ統計では、確率はあくまでも何らかの主観的な根拠に基づいて計算されるものであり、計算された確率分布を「信念」、ある事象に対する確率を「信念の度合い」と呼ぶ。

14.3　価値反復

価値反復：報酬関数に対する各行動の有効性を再帰的に計算する手法

仮定：環境の状態が各時刻において完全観測可能（MDPを考える）

14.3.1 終端状態と報酬

- 報酬関数の簡単な例

$$r(\boldsymbol{x},\boldsymbol{u}) = \begin{cases} 100 & (\boldsymbol{u}\text{により終端状態に到達する場合}) \\ -1 & (\text{その他の場合}) \end{cases}$$

- 報酬関数の導入の意義

  統一的にコストを評価する

  様々な要因のコストのトレードオフを表現するため
  「ゴールへ到達する確率を高めることが、余分な
  エネルギーや時間に見合うだろうか？」など

14.3　価値反復

- 制御方策の表現

$$\pi: \boldsymbol{y}_{1:k-1}, \boldsymbol{u}_{1:k-1} \rightarrow \boldsymbol{u}_k$$

制御方策：過去の制御入力系列と観測系列のデータを
現時刻の制御入力$\boldsymbol{u}_k$に写像する関数

完全観測可能な場合は状態量を知り得るので　$\pi: \boldsymbol{x}_{1:k} \rightarrow \boldsymbol{u}_k$

- 期待累積報酬

報酬を得る時刻には遅延が生ずるのが一般的.
将来の報酬の和が最大となるような行動を選択することが目標.

$$R_T = \mathrm{E}\left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{k+\tau}\right]$$

Tは計画区間

- ロボットが時刻kから$k+T$まで累積した報酬$r_{k+\tau}$
- γ^{τ}は割引率．$\gamma \in [0; 1]$
- この値が小さいと，未来の報酬が指数的に割り引かれる。

14.3　価値反復

期待累積報酬　$R_T = \mathrm{E}\left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{k+\tau}\right]$　Tは計画区間

$T = 1$の場合

グリーディアルゴリズム：すぐ次の報酬を最大化する．計算は早い．

$T > 1$の場合

有限区間：通常$\gamma = 1$として報酬を割り引かない．計算は複雑化する．

$T = \infty$の場合

無限区間：$\gamma < 1$かつ$|r| < r_{\max}$でR_∞を最大化する．

$$R_\infty \le r_{\max} + \gamma r_{\max} + \gamma^2 r_{\max} + \gamma^3 r_{\max} + \cdots = \frac{r_{\max}}{1-\gamma}$$

14.3　価値反復

累積報酬と状態との関連付けの表記

複数の制御方策毎の報酬を比較・選択する場合に便利

$$R_T(\boldsymbol{x}_k) = \mathrm{E}\left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{k+\tau} \,\middle|\, \boldsymbol{x}_k\right]$$

累積報酬と制御方策との関連付け（状態との依存関係を明示すると）

$$R_T^{\pi}(\boldsymbol{x}_k) = \mathrm{E}\left[\sum_{\tau=1}^{T} \gamma^{\tau} r_{k+\tau} \,\middle|\, \boldsymbol{u}_{k+\tau} = \pi(\boldsymbol{y}_{1:k+\tau-1}, \boldsymbol{u}_{1:k+\tau-1})\right]$$

各πについての$R_T^{\pi}(\boldsymbol{x}_k)$を比較し，将来の報酬が多いものを選べる．

14.3　価値反復

14.3.2 完全観測可能な場合の最適制御方策の発見

事後確率 $p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k},\boldsymbol{u}_{1:k})$ が期待値 $\mathrm{E}[p(\boldsymbol{x}_k|\boldsymbol{y}_{1:k},\boldsymbol{u}_{1:k})]$ で表現できる応用を想定
（運動モデルに誤差や外乱の混入がほとんどないシステム）

$$\pi_k : \boldsymbol{x}_k \rightarrow \boldsymbol{u}_k$$

制御方策：状態から最適入力への写像
（ある状態$\boldsymbol{x}$を実行するための最適入力$\boldsymbol{u}$を決める関数をπと呼ぼう）

$T = 1$の場合（1ステップ後の報酬を最大化する方策π_1に興味がある場合）

1ステップ後の報酬 $r(\boldsymbol{x}_{k+1},\boldsymbol{u}_{k+1})$ を最大化する制御方策を選ぶ

$$\pi_{k+1}(\boldsymbol{x}_{k+1}) = \underset{\boldsymbol{u}_{k+1}}{\operatorname{argmax}}\, r(\boldsymbol{x}_{k+1},\boldsymbol{u}_{k+1})$$

報酬$r(\boldsymbol{x}_{k+1},\boldsymbol{u}_{k+1})$は対応する価値関数を持つ

$$V_1(\boldsymbol{x}_{k+1}) = \gamma \max_{u_{k+1}} r(\boldsymbol{x}_{k+1},\boldsymbol{u}_{k+1})$$

（$\boldsymbol{x}$は固定で$\boldsymbol{u}$を様々に変更し，最大の報酬を探す．）
（γで割引された次の最大報酬を与える関数）

14.3　価値反復

$T > 1$の場合（Tステップ後の報酬を最大化する方策π_Tに興味がある場合）

$$\pi_{k+2}(\boldsymbol{x}_{k+2}) = \underset{\boldsymbol{u}_{k+2}}{\operatorname{argmax}} \left[r(\boldsymbol{x}_{k+2}, \boldsymbol{u}_{k+2}) + \int V_1(\boldsymbol{x}_{k+1}) p(\boldsymbol{x}_{k+1} | \boldsymbol{u}_{k+2}, \boldsymbol{x}_{k+2}) \mathrm{d}\boldsymbol{x}_{k+1} \right]$$

$$V_2(\boldsymbol{x}_{k+2}) = \gamma \max_{\boldsymbol{u}_{k+1}} r(\boldsymbol{x}_{k+1}, \boldsymbol{u}_{k+1})$$

⋮　再帰的に繰り返す

$$\pi_{k+T}(\boldsymbol{x}_{k+T}) = \underset{\boldsymbol{u}_{k+T}}{\operatorname{argmax}} \left[r(\boldsymbol{x}_{k+T}, \boldsymbol{u}_{k+T}) + \int V_{T-1}(\boldsymbol{x}_{k+T-1}) p(\boldsymbol{x}_{k+T-1} | \boldsymbol{u}_{k+T}, \boldsymbol{x}_{k+T}) \mathrm{d}\boldsymbol{x}_{k+T-1} \right]$$

$$V_T(\boldsymbol{x}_{k+T}) = \gamma \max_{\boldsymbol{u}_{k+T-1}} r(\boldsymbol{x}_{k+T-1}, \boldsymbol{u}_{k+T-1})$$

$\pi_{k+T}(\boldsymbol{x}_{k+T})$は計画対象区間$T$において最適

最適な制御方策を求めるアルゴリズムができそう！

14.3　価値反復

Algorithm MDP_discrete_value_iteration () :

for $i = 1$ to N do

$\hat{V}(\boldsymbol{x}_i) \leftarrow r_{\min}$

endfor

repeat until convergence

for $i = 1$ to N do

$$\hat{V}(\boldsymbol{x}_i) \leftarrow \gamma \max_{\boldsymbol{u}} \left[r(\boldsymbol{x}_i, \boldsymbol{u}) + \sum_{j=1}^{T} \hat{V}(\boldsymbol{x}_j) p(\boldsymbol{x}_j | \boldsymbol{u}, \boldsymbol{x}_i) \right]$$

endfor

endrepeat

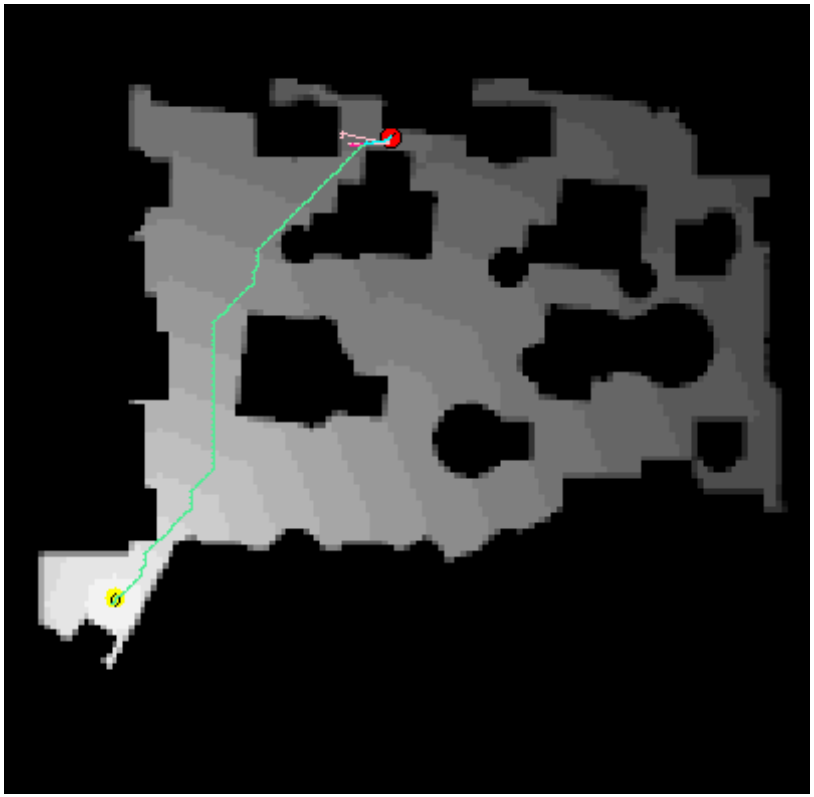
return $\hat{V}$

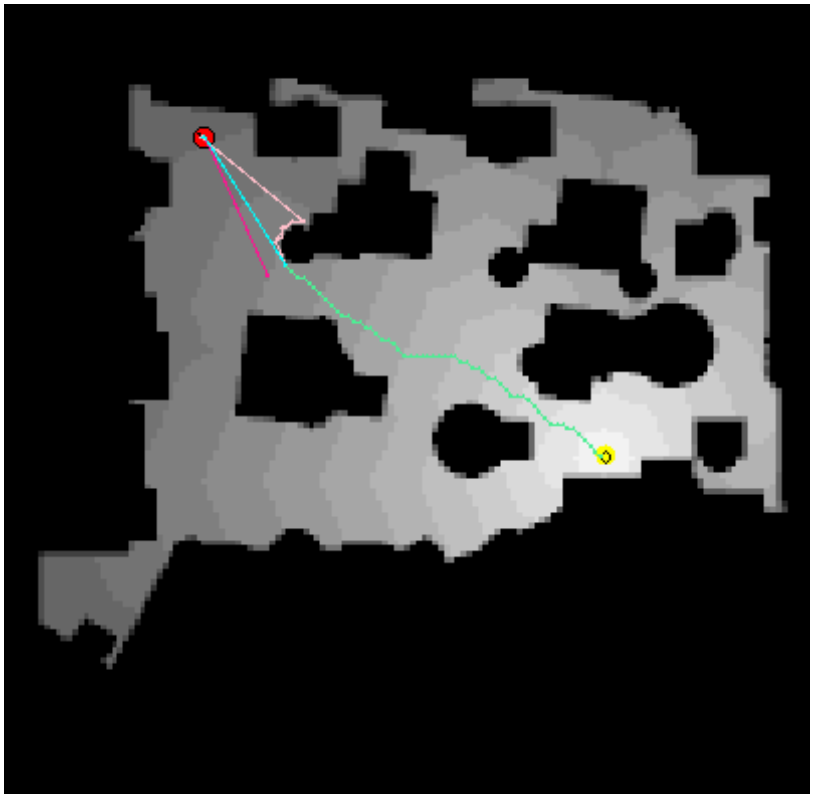
Algorithm policy_MDP $(\boldsymbol{x}, \hat{V})$:

$$\pi(\boldsymbol{x}) = \operatorname*{argmax}_{\boldsymbol{u}} \left[r(\boldsymbol{x}, \boldsymbol{u}) + \sum_{j=1}^{N} \hat{V}(\boldsymbol{x}_j) p(\boldsymbol{x}_j | \boldsymbol{u}, \boldsymbol{x}_i) \right]$$

http://www.probabilistic-robotics.org/

Value Iteration for Motion Planning





14.4　ロボット制御への応用

価値反復の特徴

ロボットが取りうるすべての状態空間全体（たとえば$[x, y, \theta, v, \omega]$）で定義される

ロボットがどこにいようとも行動を選択できる.

大域的な位置推定を行っている最中には実行できない.

実応用に利用するためには

状態空間と制御入力空間を離散化して求める.

関数$\hat{V}(\boldsymbol{x}_k)$はルックアップテーブルで実装する.

次元の呪いがあるので、離散化で分解表現が利用できるのは
低次元の状態空間表現に限られる.

高次元の場合には、価値関数を表現するために
学習アルゴリズムの導入が一般的.

15.1　部分観測マルコフ決定過程

部分観測マルコフ決定過程
（POMDPs: partially observable Markov decision processes）

- MDPsでは行動の不確かさのみを考慮していた．計測の不確かさも考慮に入れるにはPOMDPsを利用する．
- 計測の不確かさと制御の不確かさの両立を考慮する必要がある．
- 「部分」とは，環境の全状態を直接計測できないことを表す．
- 状態空間，行動空間，観測空間，計画対象区間Tがすべて有限であれば，最適制御方策を発見するアルゴリズムは存在する．
- しかし，計算量が大きくなるため，近似的なアルゴリズムになる．

15.1　部分観測マルコフ決定過程

$$\hat{V}_T(\boldsymbol{x}) = \gamma \max_u \left[r(\boldsymbol{x}, \boldsymbol{u}) + \sum_{j=1}^{T} V_{T-1}(\boldsymbol{x}') p(\boldsymbol{x}' | \boldsymbol{u}, \boldsymbol{x}) \mathrm{d}\boldsymbol{x}' \right] \qquad V_1(\boldsymbol{x}) = \gamma \max_u r(\boldsymbol{x}, \boldsymbol{u})$$

$\boldsymbol{x}$は部分観測可能

$\boldsymbol{x}$が部分的に観測できない

$$\hat{V}_T(\boldsymbol{b}) = \gamma \max_u \left[r(\boldsymbol{b}, \boldsymbol{u}) + \sum_{j=1}^{T} V_{T-1}(\boldsymbol{b}') p(\boldsymbol{b}' | \boldsymbol{u}, \boldsymbol{b}) \mathrm{d}\boldsymbol{b}' \right] \qquad V_1(\boldsymbol{b}) = \gamma \max_{\boldsymbol{u}} \mathrm{E}_x[r(\boldsymbol{x}, \boldsymbol{u})]$$

事後確率分布（信念空間$\boldsymbol{b}$）において価値評価を行う

制御方策の決定

$$\pi_T(\boldsymbol{b}) = \operatorname*{argmax}_u \left[r(\boldsymbol{b}, \boldsymbol{u}) + \sum_{j=1}^{T} V_{T-1}(\boldsymbol{b}') p(\boldsymbol{b}' | \boldsymbol{u}, \boldsymbol{b}) \mathrm{d}\boldsymbol{b}' \right]$$

15.1　部分観測マルコフ決定過程

制御方策の決定アルゴリズム

POMDP：

有限的なPOMDSアルゴリズムよりも正確だが計算量に問題がある.

ロボットが明快な信念状態から行動を開始するとき，その後取りうる信念状態の数はごく少数に限られることが多い.

価値関数が適切な信念状態だけではなく全ての信念状態に対して計算されるということは，価値反復アルゴリズムの欠点.

PBVI：
ポイントベースド価値反復

典型的な信念状態の組み合わせを考え，価値関数をその組み合わせ内で最大化するように制限する.

与えられる現状のどの信念状態とも対応しない関連しない拘束を生成しないことで価値計算を効率化する.

15.1　部分観測マルコフ決定過程

制御方策の決定アルゴリズム

QMDP：
MDPとPOMDPのハイブリッド手法．行動を一回行うと，状態が完全に観測可能になるという仮定のもと，信念空間における正確な価値関数を得ることができる．MDPと同じ計算複雑性を持つ．

拡張MDP：
AMDP: augmented MDP
信念状態を低次元な十分統計量に落として，その低次元空間で価値反復を行う方法．もっとも基本的な実装では，最尤な状態とエントロピーで計算された不確かさの度合いを組み合わせた表現を用いる．
MDPの計算量よりも効率は良くならないが，精度は向上する．

MC-POMDP：
モンテカルロPOMDP
POMDPのパーティクルフィルタバージョン．信念をパーティクルで近似する．信念を動的に生成することで，信念の数を比較的少なくできる．連続量の状態，行動，計測に適用可能である．ただし，パーティクルフィルタの一般的な手法と同等の問題や，MC-POMDP特有の問題が生ずる．