「主要情報記憶−細部制約付き探索」による軌道計画の高速化*

−学習ベース手法による環境ロバストな多関節ロボットの軌道生成−

Learning Embodied Optimal Point and Restricted Deft Search
− Learning Method for Robot Manipulator on changing Environment −

蓑谷 顕一　Kenichi MINOYA
尾崎 智章　Tomoaki OZAKI

We propose Learning Embodied Optimal Point and Restricted Deft Search (LEOPA): a learning based hierarchical motion planning algorithm which achieves a faster learning convergence and better solution through reasonable state and action spaces. LEOPA breaks down the motion planning problems into the learning of the end effector position and exploration of the remained posture. We evaluate LEOPA on a motion planning environment which consists of finding a valid, collision-free path for a 6 DOF robot manipulator from a start configuration to a goal configuration. The results show that LEOPA demonstrates a 100% success rate consistently in all 200 unseen environments. Also the average computation time of the results is less than 1 second, which is significantly lower than sampling-based methods.

Key words :
Motion Planing, Samplling-Based Method, Deep Neural Networks, Hierarchical Reinforcement Learning

特集

1. はじめに

かつてものづくりは少品種大量生産が主流であったが，近年では「CASE（コネクテッド，自動運転，シェアリング，電動化）」の台頭などによって多様化した顧客のニーズに柔軟に対応するために，多品種少量生産向けの生産システムの必要性が高まってきている．品種毎での作り込みが必要な従来のライン生産システムでは品種増大に対応しきれないため，今後はセル生産システムの導入が増えていくことが予想される．セル生産システムとは複数の作業員がひとまとまりの作業工程を担当する方式で，同時に異なった品種を複数生産できる多品種少量生産向けの方式である．現在は十分な教育を受けた多能工が作業を担当しているが，今後はコスト面や将来の労働者不足等の理由により，人からロボットへと移行していくことが予想される．セル生産システムでは，限られた狭いスペースの中で，ロボット同士の干渉回避や役割分担を考慮しなければならず，従来のような if-then ルールでは，ロボットのプログラム工数が激増してしまう．以上のような背景から，ロボット動作軌道を自動生成する技術への期待が高まっている．

ロボット軌道生成の代表的な手法としては Table 1 に示す探索ベース手法と学習ベース手法が挙げられる．

＊計測自動制御学会の了解を得て，「第 25 回ロボティクスシンポジア (2020 年)」講演番号 4B3 より一部加筆して転載

探索ベース手法[1)-3)]では，ロボットの動作制約を考慮しながら実行時にランダム（確率的）に探索を行う．探索ベース手法は時間を掛けることで確実に解が求まる一方，処理時間と軌道の最適性にトレードオフの関係があり，複雑な環境下では長い処理時間（数秒～数十分）を要してしまうことがある．一方，学習済みニューラルネットモデルを用い，実行時は決定論的に出力する学習ベース手法[4)-9)]は，超高速（10 ～ 100ms）に近似解を出力可能である．この為，探索ベースでは試行回数が増大しがちな複雑レイアウトや環境が変化する度に再計算が必要なマルチロボットのような動的環境では欠かせない技術となることが予想される．(Fig. 1) しかしながら，学習ベース手法ではロボットの膨大な動作パターンをニューラルネットワークに学習させることが困難な為，研究レベルでは活性化している一方で実用化に至った例は極めて少ないのが現状である．これは第一にロボットの行動出力が連続値であることに加え，軸が複数存在する為，解空間が膨大となる事が要因として考えられる．第二の要因は多様なレイアウトパターンに対応させる為に，障害物との相対的な位置関係も含めてニューラルネットワークに学習させる必要があるという点である．特に多関節ロボットでは障害物とロボットの座標系が異なる為，難解な座標変換規則（3 次元直交座標系から関節座標系への変換）を学習させなければならない．

そこで，本研究では Fig. 2 に示すよう階層構造の導入による状態数削減と，幾何計算と探索による衝突回避機能の外部実装により上述した問題を解決可能な主要情報記憶ー細部制約付き探索を提案する．一般的に階層構造を導入する場合，個々の階層で扱う状態空間は小さくなるが，上位階層での抽象化が適切でない場合必要な情報が失われてしまう事がある．そこで本研究では，上位階層ではロボット動作生成にとって重要な情報が縮約された手先位置を生成し，下位階層で残りの姿勢を求める構成とすることで，解の質を損ねることなく状態数削減が可能な階層化方法を提案する．なお，ニューラルネットワークで当たりを付け，細部は解析的に求めるという考え方はロボットの軌道生成だけでなく，膨大な状態空間を持つその他多くの問題に幅広く適用していけると考えている．

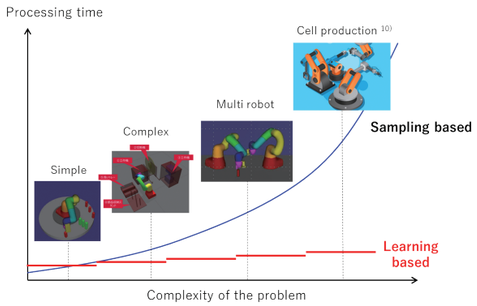


Fig. 1　Roadmap of the autonomous control technology

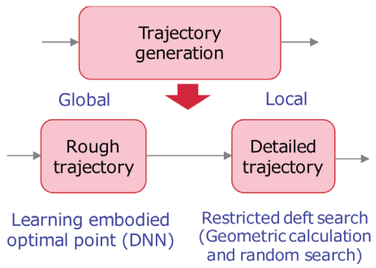


Fig. 2　Proposed method

Table 1　Conventional method for motion planning problems

	(a) Sampling-Based Method	(b) Learning-Based Method	(c) Hybrid Mehtod
Example methods	RRT[1], PRM[3]	GPS[4], HER[6]	LSBMP[11], MpNet[12]
Overview	They randomly sample valid robot configurations and form a graph of optimal motions.	They deterministically generate optimal robot configurations using deep neural networks.	They first generate the candidate of robot configurations using deep neural networks and form a graph of optimal motions.
Feature	A solution can be surely obtained by taking time, however; there is a trade-off between processing time and optimality.	Although it is possible to output an approximate solution at very-high speed, it is difficult for DNN to learn the enormous motion patterns of the robot.	A method that combines sampling-based reliability and learning-based speed.

2. 関連研究

学習ベース手法は価値ベース手法（DQN[7)] 等）と方策ベース手法（GPS[4)] 等）に大別できる．価値ベース手法は将棋やチェス等，行動出力が離散的に表現可能な問題でよく使用される．価値ベース手法は各状態で取るべき行動の評価値（Q 値）を学習する必要がある為，連続値をとる行動方策の学習には不向きである．一方，方策ベース手法では制御則を確率分布で表現する為，行動空間が連続的な多関節ロボット等の軌道生成に用いられることが多い．しかし，方策ベース手法では目的とするタスク達成にかかわる運動軌道まわりに注力して制御則を導出する為，局所解に陥りやすく，単純な動作獲得に留まっていた．最近では学習を効率よく行う為のテクニックも多数提案されており[5) 6) 8)]，複雑な動作獲得も可能になってきた．ただし，そのほとんどは周辺環境が固定されたレイアウトを対象としており，多様なレイアウト環境に対応している事例は少ない．環境ロバストな学習ベース手法として VIN[9)] 等が存在するが，アルゴリズムの構成上多関節ロボットに応用することは難しい．

一方近年，DNN（Deep Neural Networks）に確率要素を導入し，非干渉の最短軌道を探索的に求めるハイブリッド手法[11) 12)] 等が提案されている．ハイブリッド手法は，DNN が学習した範囲から多少離れたデータであっても，到達と衝突回避を満たす軌道を生成可能な信頼性と高速性を兼ね備えた手法である．多関節ロボットへの適用も一部検討されているが，両手法[11) 12)] ともにロボットの膨大な状態空間を階層化する事無く，一度に軌道生成問題を解いている為，レイアウトパターンが膨大に存在するような工場の実シーンでは学習収束性が課題となる事が予想される．そこで本研究では，ハイブリッドプランナー[12)] に階層構造を導入することで，状態数を削減し，環境ロバストな多関節ロボットの学習器を構築できないか試みる．

3. 階層化の課題

階層構造の導入にあたっては，その階層の区切り方が学習の成否に大きく左右される点に留意する必要がある．ここでは Fig. 3 に示す 2 次元迷路を例に説明する．Fig. 3 では迷路問題を解くにあたりグローバルな経路を出力する上位階層とローカルな経路を出力する下位階層に分けることで問題を簡単化している．しかし，上位階層で抽象化の仕方を誤ると遠回りの経路を出力してしまう問題（解の最適性）や，解が存在しない場所にサブゴールを生成してしまう問題（解の完全性）が発生する．このように上位階層で問題分割の粒度が大きいと，必要な情報が排除されてしまうために不完全知覚問題が生じ，最適なふるまいが獲得できない恐れがある．逆に問題分割の粒度が小さいと学習速度が落ちてしまう．よって上位階層では問題を簡単化しつつも，重要な情報は極力損なわれない抽象化方法を考える必要がある．このような条件に適っている中間状態の一つとして考えられるのは手先位置（エンドエフェクタ）であろう．なぜなら，手先位置は全関節の影響を受ける為，ロボット動作生成をする上で重要な情報が縮約された主要情報である為である．

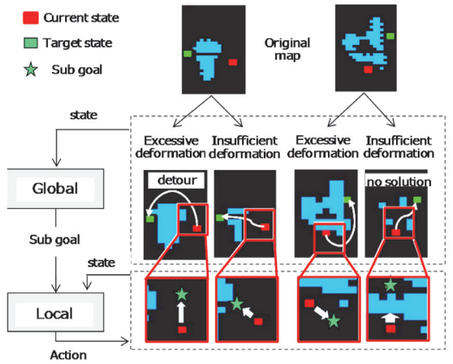
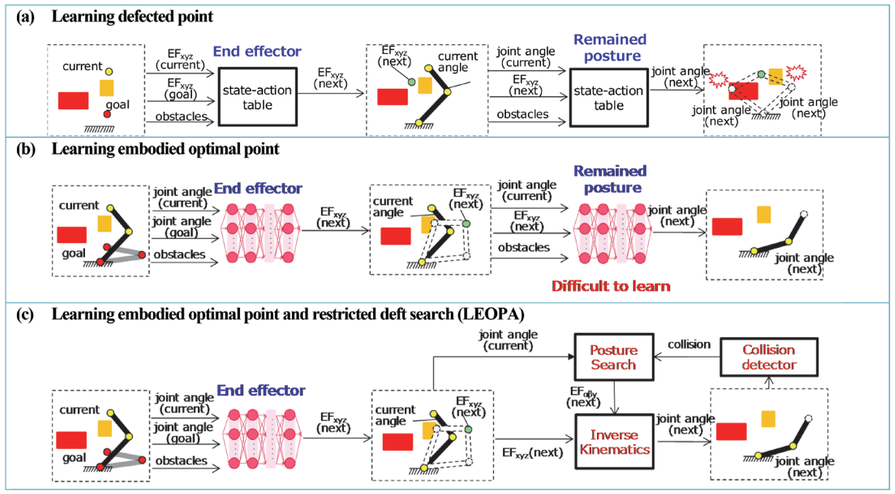


Fig. 3　Problem of the hierarchical method

4. 提案手法

手先位置を中間状態に持つ階層化方法としてまず考えられるのが Table 2 (a) に示す部分情報記憶型である．ここでは古典的な強化学習のように状態ー行動テーブルを方策として持つ事を想定している．上位階層で手先位置を定め，下位階層で手先位置を取る為の姿勢を生成することで状態数を削減可能である．本構成の問題は，上位階層でロボットの身体性（姿勢）が考慮されていない（不完全知覚問題）為，下位階層で解が得

Table 2 Hierarchical methods which take an end-effector position as an intermediate state. (EF_{xyz}: end-effector position, $EF_{\alpha\beta\gamma}$: end-effector posture)



られない可能性がある点である．次に，Table 2 (b) に示す主要情報記憶型では，各階層は DNN で表現された方策に従って行動決定をするものとする．ここでは上位階層の入力にロボットの関節をそのまま入力しているので，前述したような不完全知覚問題は解消される．DNN は多次元入力を内部でより少ない次元に抽象化することが可能な為，本構成との相性は良いと考えられる．なお，本構成は Hollerback [13] が提案するヒトの随意運動の計算理論のモデルの中で，効果器レベル（関節角度から筋運動への変換）を除いたものを DNN で実装したものに相当する．本構成の問題点は下位階層の学習収束性である．手先位置を取るための姿勢は無数存在する上に，ロボットと障害物との相対的な位置関係も含めて DNN に学習させなければならない為である．情報が縮約された手先位置との相対関係のみを記憶すればよい上位階層に比べ，ロボット全体を考慮する必要がある下位階層の学習は難易度が高く，実際，学習が収束せず満足する解を得られなかった．そこで今回我々は，Table 2 (c) に示すように学習難易度の高い下階層の姿勢を探索によって反復的に求める主要情報記憶－細部制約付き探索（LEOPA）を提案する．LEOPA ではまず RRT 等の探索ベース手法によってタイムステップ毎の環境状態とロボットの関節状態を生成する．そして，上位階層の DNN では生成した教師データのうち，手先位置のみを覚えこませる．ここではロボットの動作制約を考慮して生成した軌道を教師としているので，到達と衝突回避を満たすロボットの軌道と姿勢を非明示的に覚えさせることが可能である．次に，下位階層では，手先位置を取る為の非干渉の手先姿勢をランダムに生成する．手先位置と手先姿勢が求まれば，逆運動学によって幾何学的に関節角度を求めることができる為，非干渉の姿勢を反復的に探索することが可能である．ここでのポイントは，手先姿勢を直前にとった姿勢の近辺から探索する点である．このような構成にすることで，ロボット動作の連続性を考慮できるだけでなく，現実的な探索範囲に落とし込むことが可能である．

5. 全体処理フロー

Algorithm 1 にハイブリッドプランナーの概要を示す．Algorithm 1 の 1 行目で MPNet [12] を選択した場合の処理フローを Algorithm 2 に示す．また，LEOPA を選択した場合の処理フローを Algorithm 3 に示す．両者の相違点については Algorithm 2, 3 に太字で示した．

Algorithm 1: HybridPlanner(obs,x_{init},x_{goal})

```
τ←MPNet[1] (obs,x_init,x_goal) or LEOPA[2] (obs,x_init,x_goal)
if τ then
  τ←LazyStatesContration[3] (τ);
  if IsFeasible[4] (τ) then
    return τ
  else
    τ_new←Replanning[5] (τ, obs)
    τ←LazyStatesContration[3] (τ_new)
    If IsFeasible[4] (τ_new) then
      return τ_new
  return ∅
```

Algorithm 2: MPNet[1] (obs,x_{start}, x_{goal})

```
τ^a←{x_start};τ^b←{x_goal};
τ←∅;
Reached←False;
for i ←0 to N do
  x_now←τ^a(end); x_goal←τ^b(end);
  x_new←JOINT_DNN[6] (obs, x_now, x_goal)
  τ^a←τ^a ∪ {x_new}
  Connect←steerTo[7] (τ^a(end), τ^b(end))
  if Connect then
    τ←concatenate(τ^a, τ^b)
    return τ
  SWAP(τ^a, τ^b)
return ∅
```

Algorithm 3: LEOPA[2] (obs,x_{start}, x_{goal})

```
τ^a←{x_start};τ^b←{x_goal};
τ←∅;
Reached←False;
for i ←0 to N do
  x_now←τ^a(end); x_goal←τ^b(end);
  h_xyz←E_EFFECTOR_DNN[8] (obs, x_now, x_goal)
  for j ←0 to M do
    h_αβγ←PostureSearch[9] (x_now)
    x_new←IK[10] (h_xyz, h_αβγ)
    if CollisionCheck[11] (x_new) then
      break
  τ^a←τ^a ∪ {x_new}
  Connect←steerTo[7] (τ^a(end), τ^b(end))
  if Connect then
    τ←concatenate(τ^a, τ^b)
    return τ
  SWAP(τ^a, τ^b)
return ∅
```

Algorithm 4: Replanning(τ, obs)

```
τ_new←∅;
for i ←0 to τ.length() do
  if steerTo[7] (τ_i , τ_i+1) then
    τ_new←τ_new ∪ {τ_i , τ_i+1}
  else
    τ_mini←MPNet[1](obs,τ_i,τ_i+1) or LEOPA[2] (obs,τ_i ,τ_i+1)
  if τ_mini then
    τ_new←τ_new ∪ τ_mini
  else
    return ∅
return τ_new
```

1) MPNet：障害物情報 obs とスタート x_{start} とゴール x_{goal} を入力状態として持ち，RRT-Connect[2] と同様にスタートとゴール双方向から非干渉の経路を求める．RRT-Connect では候補点を探す際ランダムサンプリングをするのに対しここでは DNN で学習した結果を適用する．

2) LEOPA: 1) と同様にスタートとゴール双方向から非干渉の経路を求める．MPNet では関節角度を DNN で直接求めるのに対し，LEOPA では手先位置を DNN で求めた後，残りの非干渉な姿勢を反復的に探索する．

3) LazyStatesContration: 得られた経路 $\tau = \{x_0, x_1, \cdots, x_T\}$ についてノード間をショートカットできる部分については経路を短縮する．

4) isFeasible: 得られた経路 $\tau = \{x_0, x_1, \cdots, x_T\}$ に衝突する箇所がないかチェックする．

5) Replanning: 指定されたパス $\tau = \{x_0, x_1, \cdots, x_T\}$ のすべての隣り合うノード（x_i, x_{i+1}: i= [0,T -1] ⊂ N）について，それらが接続可能かどうか（2 点間を線形移動した際，障害物と一度も干渉しない事）を確認する．隣り合うノードが接続できない場合，ノード x_i, x_{i+1} をそれぞれスタート及びゴール状態とし，MPNet あるいは LEOPA モジュールによって，非干渉の軌道を生成する．

6) JOINT_DNN: 障害物座標値 obs，時刻 t における関節状態 x_t，及び目標関節状態 x_T を入力状態として持ち，時刻 t+1 の関節状態 xt+1 を出力する DNN を指す．MPNet [12] と同様，サンプリング機能を持たせるために，各層に確率 p:[0,1] ∈ R でドロップアウトを追加しているが，本研究ではさらに局所解回避効果を高めるため，DNN の出力側に対しても乱数を加えることで，より広範囲なサンプリングを可能とした．

7) steerTo: 2 つのノード間を線形補完で移動した場合に，障害物と干渉するか否かをチェックする．

8) E_EFFECTOR_DNN: 障害物座標値 obs，時刻 t における関節状態 x_t，目標関節状態 x_T を入力状態として持ち，時刻 t+1 の手先位置 h_{xyz} を生成する DNN を指す．ドロップアウトや出力側の乱数生成については JOINT_DNN と同様である．

特集

9） PostureSearch: 関節角度 x_{now} に対し各関節± θ 度の一様乱数を加えた関節角度 x'_{now} に対し順運動学で手先姿勢 $h_{\alpha\beta\gamma}$ を求める．

10） IK: 手先位置 h_{xyz} と $h_{\alpha\beta\gamma}$ を入力状態として持ち，逆運動学によって関節角度を求める．

11） CollisionCheck: 関節状態 x_{now} を入力状態として持ち，干渉しているか否かをチェックする．

6. 実験設定

A. 検証環境

ニューラルネットワークはPyTorchで実装し，学習時，実行時共にGPUを使用した．ハイブリッドプランナーの実装については MPNet [12] の著者が公開中のオープンソース [14] を参考にした．ベンチマーク対象の探索ベース手法は python で実装された RRT-Connect を用いた．ロボットはデンソーウェーブ製の 6 軸小型ロボット COBOTTA を用い，シミュレーションは Mujoco を使った．実験で使用した PC は 4.00GHz の Intel Core i7-6700k で GPU は GeForce GTX 1080 である．

B. データ収集

8000 個の異なるレイアウトそれぞれに対し，スタートとゴールが異なる 20 種類の環境を用意した．レイアウトはサイズが等しい立方体 8 つをロボット本体と重ならないようワークスペース上にランダムに配置し，スタートとゴールは障害物と非干渉のポーズが取れる関節状態をランダムに生成した．教師データは RRT-Connect で生成した軌道を用いた．学習時は 8000 レイアウトのうち 7800 レイアウトを使用し，評価時は未学習の残り 200 レイアウトを使用した．

C. ニューラルネットワークアーキテクチャ

中間層には 1280, 1024, 896, 768, 512, 384, 256, 128, 64, 32 の 10 層を設け，各層の活性化関数には Parametric Rectified Linear Unit（PReLU）を用い，各層 PReLU の処理後に Dropout を確率 50% で実装した．Optimizer は Adagrad を使用した．JOINT_DNN，E_EFFECTOR_DNN 共に出力層で生成された角軸の関節角度に対し平均 0，標準偏差 2 の正規分布の乱数を加えた．学習時間は MPNet が 1 週間程度，LEOPA が 3 日程度で打ち切った．

D. その他パラメータ

PostureSearch における逆運動学はヤコビアンベースのものを使用し，姿勢を求める際のパラメータ θ は 10 とした．また，MPNet, LEOPA 両者ともに探索上限 N は 500 とし，これを超えると未達（失敗）とした．また，LEOPA での姿勢生成回数の上限 M は 15 とした．

7. 実験結果

Fig. 4 に達成率を示す．Fig. 4 に示すように，階層化無しの MPNet では達成率が 31% 程度に留まっていたが，LEOPA では 100% を達成できた．Fig. 5，Fig. 6 に平均処理時間と平均移動コストを示す．ここで，平均移動コストとはスタートからゴールまでのロボットの関節移動量の平均値を意味する．Fig. 5，Fig. 6 に示すよう LEOPA では RRT-Connect とほぼ同程度の平均移動コストで，処理時間を大幅に（約 83%）低減できていた．また Fig. 7 に RRT-Connect の最適化レベルを変化させた場合の処理時間と移動コストの関係を LEOPA の結果と合わせて示す．Fig. 7 に示すように RRT-Connect では処理時間と移動コスト（最適性）の間にトレードオフの関係があるが，LEOPA では高速に準最適解を得られていることが分かる．参考までに Fig. 8 に LEOPA で得られた軌道の例を示す．

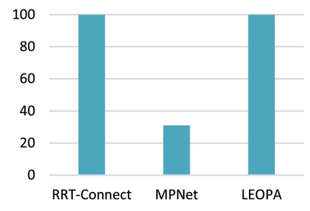


Fig. 4 Achievement ratio (%)

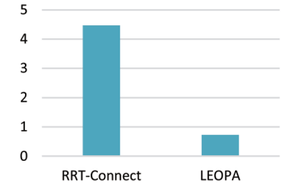


Fig. 5 Average processing time (s)

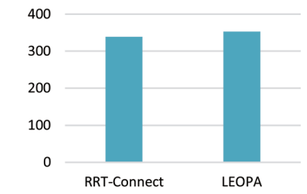


Fig. 6 Average cost (degree)

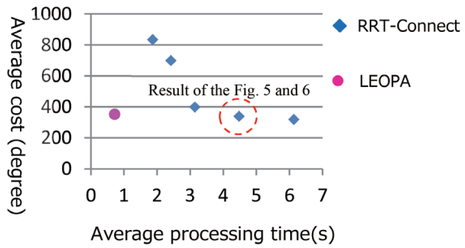


Fig. 7 Processing time and cost

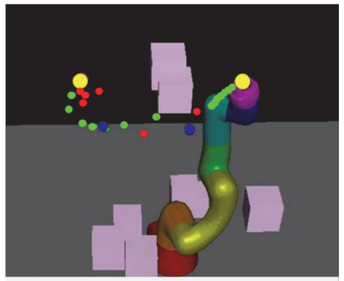


Fig. 8 Example trajectory

8. おわりに

本研究では軌道生成問題を手先位置の学習とその他の姿勢を探索する問題に分割することで，未知のレイアウトパターンに対しても再学習不要な主要情報－記憶細部制約付き探索を提案し，6 軸ロボットでその有効性を確認した．今後は製品化に向け，より幅広いシーンへの対応を検討していきたいと考えている．特に今回は Fig. 8 に示すように単純形状の障害物を対象としていたが，今後は多様な形状の障害物が存在する実用シーンのテストにも耐えうるように手法をブラッシュアップしていきたいと考えている．

参考文献

1） S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.
2） J. J. Kuffner Jr and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in ICRA, vol. 2, 2000.
3） L. E. Kavraki and J.-C. Latombe, “Probabilistic roadmaps for robot path planning,” 1998.
4） S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” The Journal of Machine Learning Research, vol. 17, no. 1, pp. 1334–1373, 2016.
5） A. Levy, R. Platt, K. Saenko, Hierarchical reinforcement learning with hindsight, arXiv:1805.08180, 2018.
6） M. Andrychowicz et al., “Hindsight experience replay,” in Advances in neural information processing systems, 2017.
7） V. Mnih, K. Kavukcuoglu, D. Silver, et,al., “Human-level control through deep reinforcement learning Nature”, 2015. Vezhnevets et al. FeUdal Networks for Hierarchical Reinforcement Learning. ICML, 2017.
8） C. Colas, P. Fournier, O. Sigaud et. al., “CURIOUS: Intrinsically Motivated Multi-Task Multi-Goal Reinforcement Learning”, 2018.
9） A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, “Value iteration networks,” in Advances in Neural Information Processing Systems, 2016, pp. 2154–2162.
10） https://jp.freepik.com/free-vector/industrial-robot-set_2869637.htm（macrovector / Freepik によるデザイン）
11） B. Ichter, M. Pavone, Robot motion planning in learned latent spaces, “IEEE Robotics and Automation Letters 4”, 2019.
12） A. H. Qureshi et. al., Motion Planning Networks: Bridging the Gap Between Learning-based and Classical Motion Planners, CoRR, 2019.
13） J. Hollerback, Computers, brains and the control of movement. Trends in Neuroscience, 1982.
14） https://github.com/ahq1993/MPNet

特集

著者

蓑谷 顕一
みのや けんいち

AI 研究部　博士 (情報科学)
自律制御ロボット開発に従事

尾崎 智章
おざき ともあき

AI 研究部
自律制御ロボット開発に従事