深層強化学習と活用するためのコツ

中部大学
機械知覚&ロボティクスグループ

山下隆義　　平川翼

CHUBU UNIVERSITY

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

講演内容

- 強化学習について
- 深層強化学習のアルゴリズム
- 深層強化学習のフレームワーク
- 深層強化学習のコツ
- 仮想環境を用いた強化学習

強化学習について

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

強化学習で何ができる？

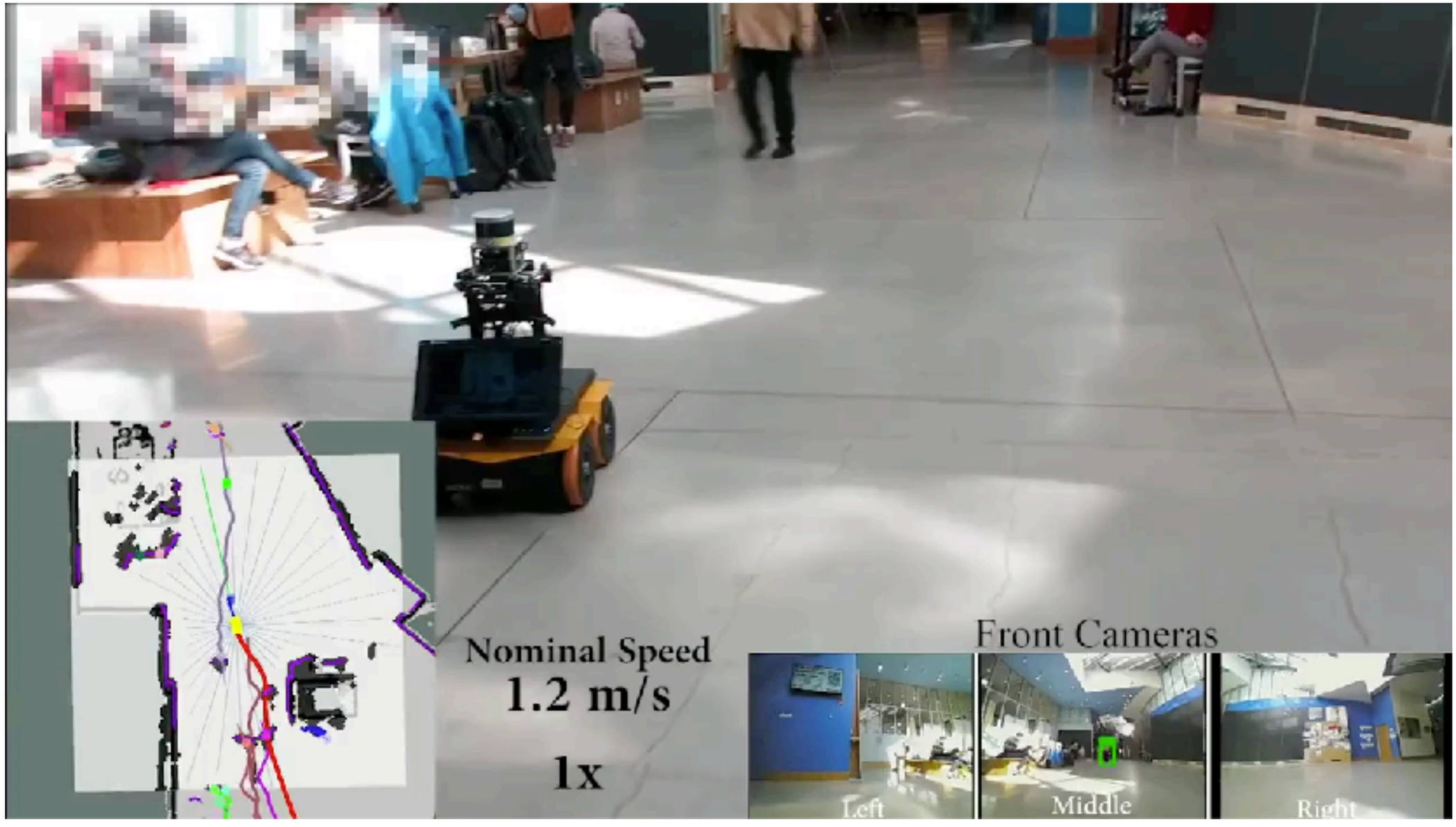
- ゲーム
  - AlphaGo [Silver, et al., 2016]

http://y2u.be/WCfgaqfRwZY

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

強化学習で何ができる？

- ロボットの制御
  - 自律走行 [Chen, et al., 2017]



http://y2u.be/CK1szio7PyA

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

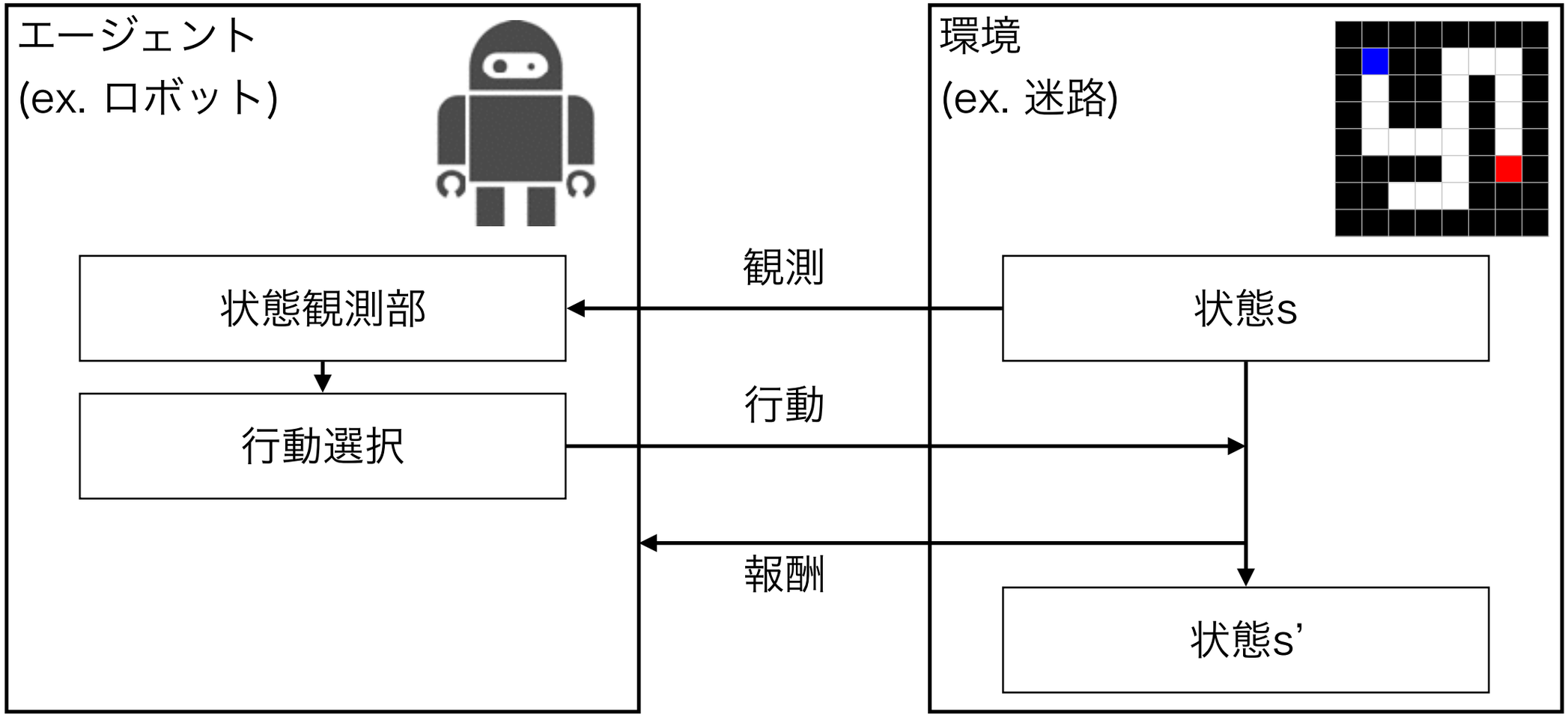
強化学習で何ができる？

- ロボットの制御
  - 物体把持 [Levine, et al., 2016]

http://y2u.be/iaF43Ze1oel

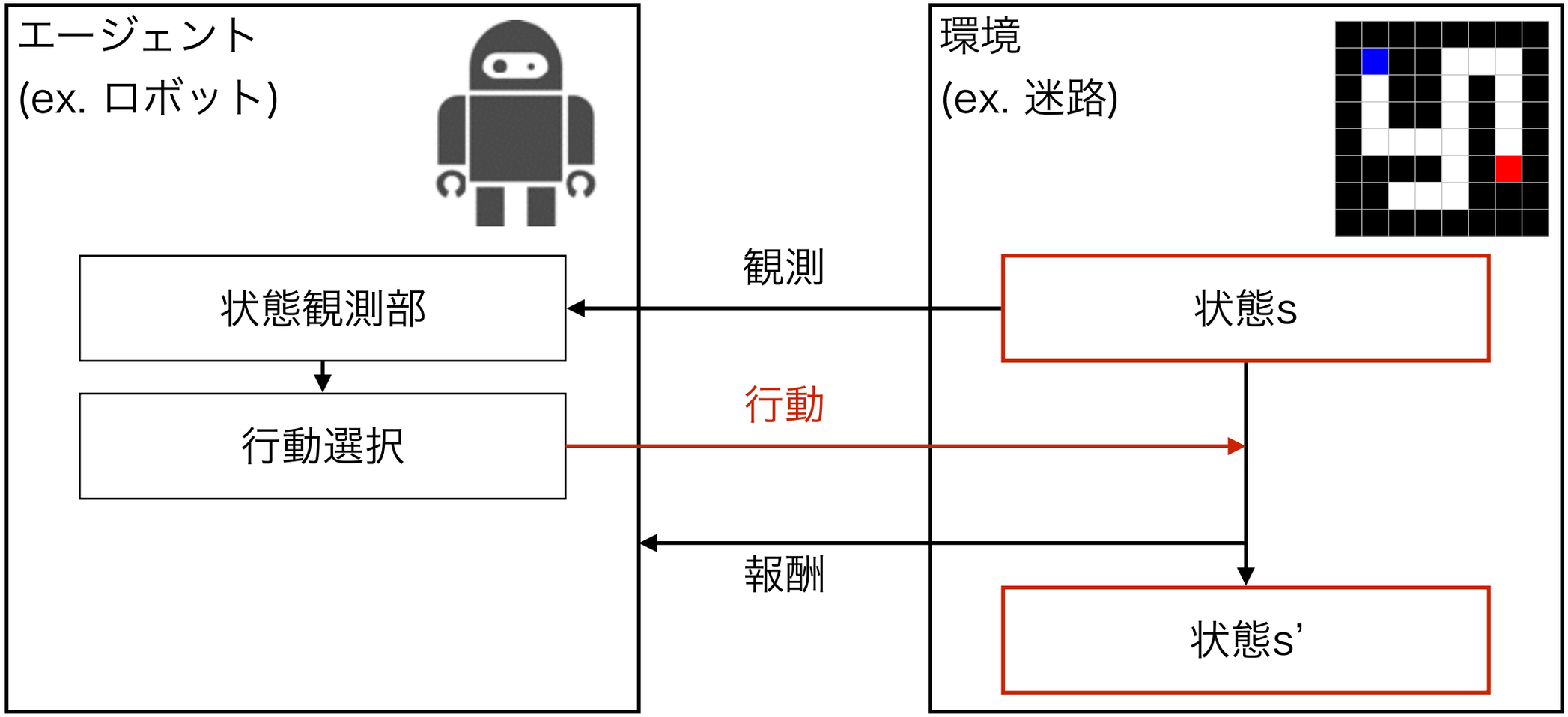
強化学習の概念

- 試行錯誤を繰り返すことで動作を獲得する
  1. 現在の状態を観測
  2. 観測を踏まえて行動を選択
  3. エージェントが行動
     - 状態が変化
  4. 報酬を受け取る



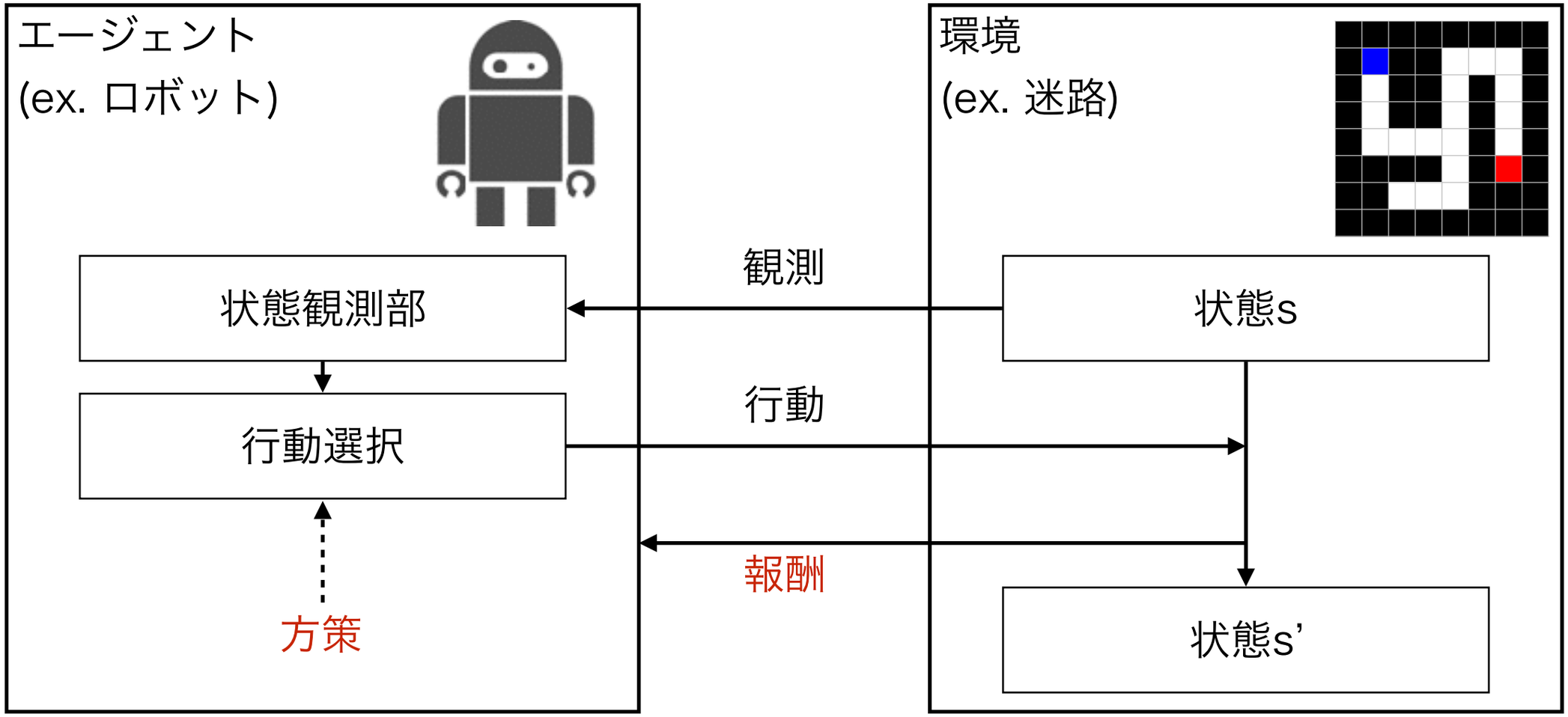
強化学習とは ~用語~

- 状態 … エージェントが置かれている環境
  - 迷路の中での位置
- 行動 … エージェントの環境に対する働きかけ
  - 隣接するマスへの移動



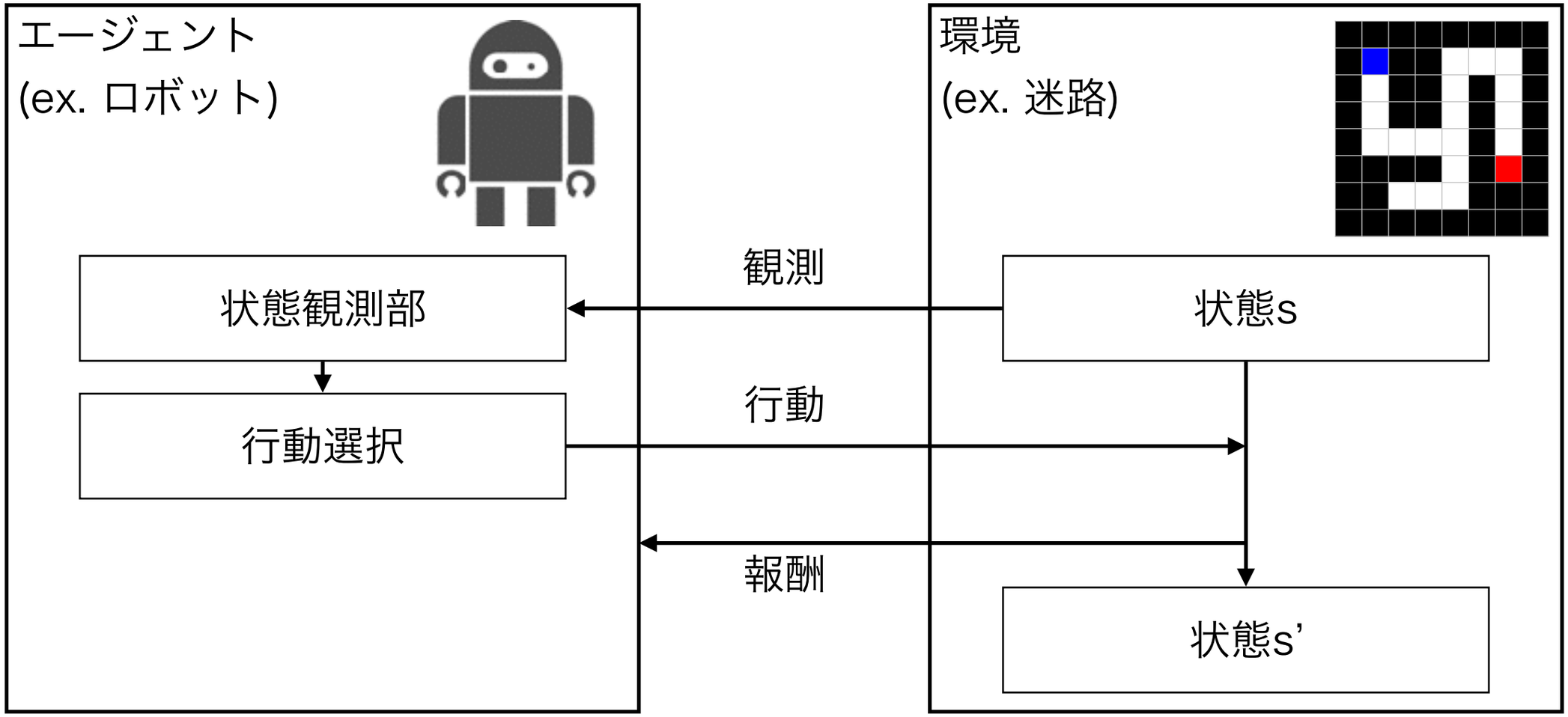
強化学習とは ~用語~

- 方策 … エージェントの行動規範
  - 今の状態でどこへ移動すれば良いか
- 報酬 … 行動の結果の良さを表す値
  - 壁にぶつかると-1点，ゴールすると+100点



強化学習の目的

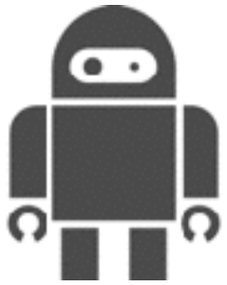
獲得できる報酬を
最大化するような行動を学習



報酬の考え方

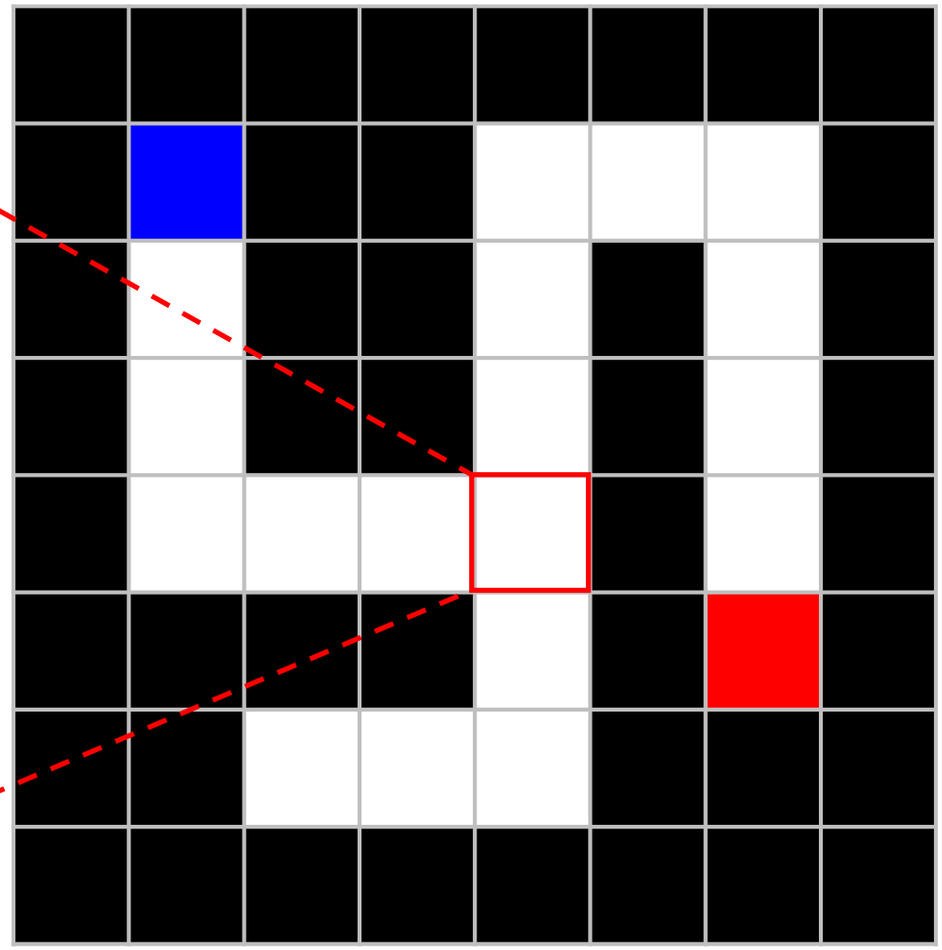
- （今は低くても）後に高い報酬が得られる可能性を考慮
  - 即時報酬　　… ある行動をとった直後に得られる報酬
  - 収益（遅延報酬）… その後に得られる報酬を合計したもの

エージェント



即時報酬
「ゴールまで近づくから,
下に行こう」

遅延報酬
「上の方が遠くなるけど,
ゴールへ辿り着きそう」



MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

報酬の定式化

- （今は低くても）後に高い報酬が得られる可能性を考慮
  - 即時報酬　　… ある行動をとった直後に得られる報酬
  - 収益（遅延報酬）… その後に得られる報酬を合計したもの
  - 割引報酬和 … 未来の不確実性を報酬から割り引いた収益

$$R_t = \sum_{\tau=0}^{\infty} \gamma^{\tau} r_{t+\tau} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \cdots$$

※ γ：割引率

γ=0 の場合（現在の状態・行動だけを考慮）

$$R_t = r_t$$

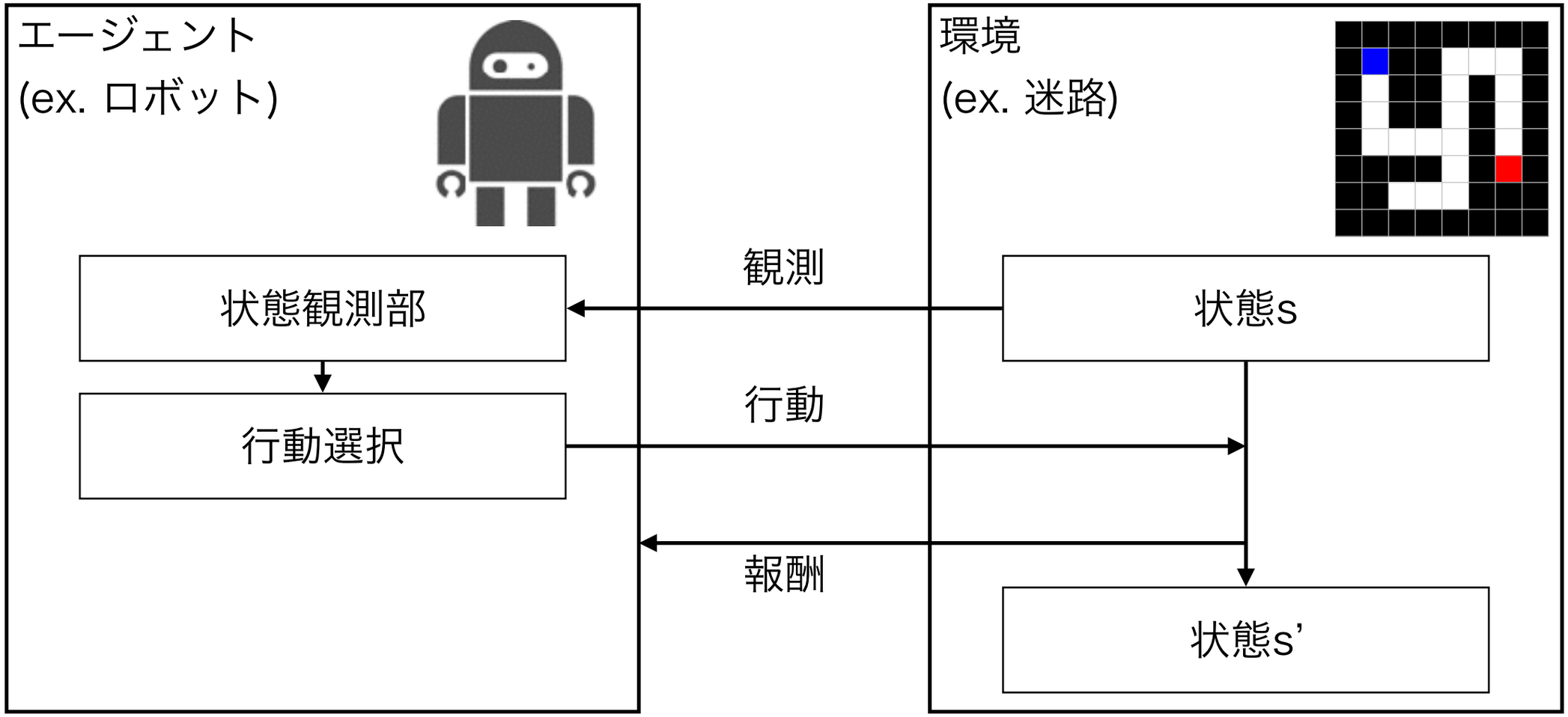
γ=0.9 の場合（ある程度近い未来を考慮）

$$R_t = r_t + 0.9 r_t + 0.81 r_{t+2} + \cdots$$

強化学習の目的

将来にわたって得られる報酬（収益）を

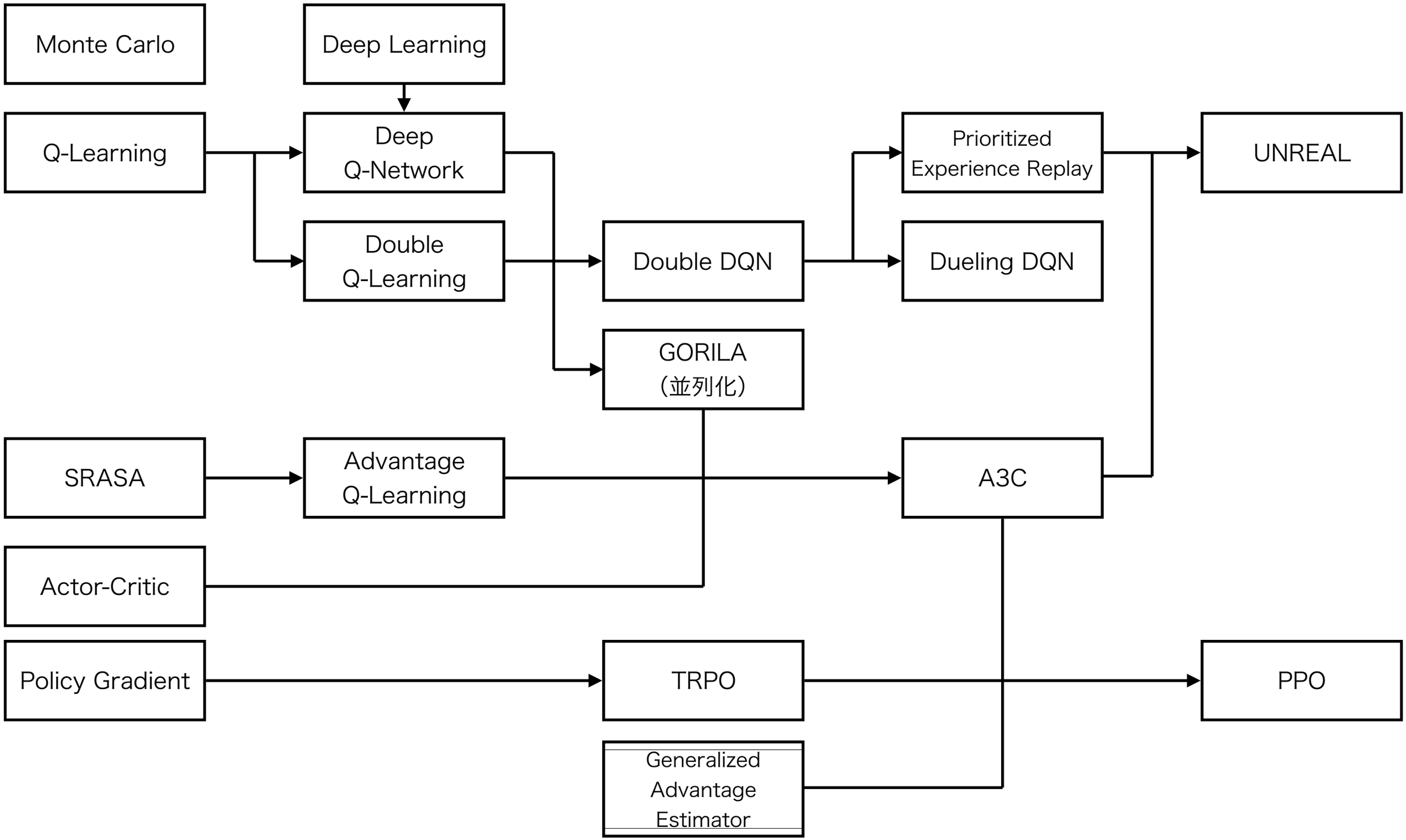
最大化するような行動を学習



深層強化学習のアルゴリズム

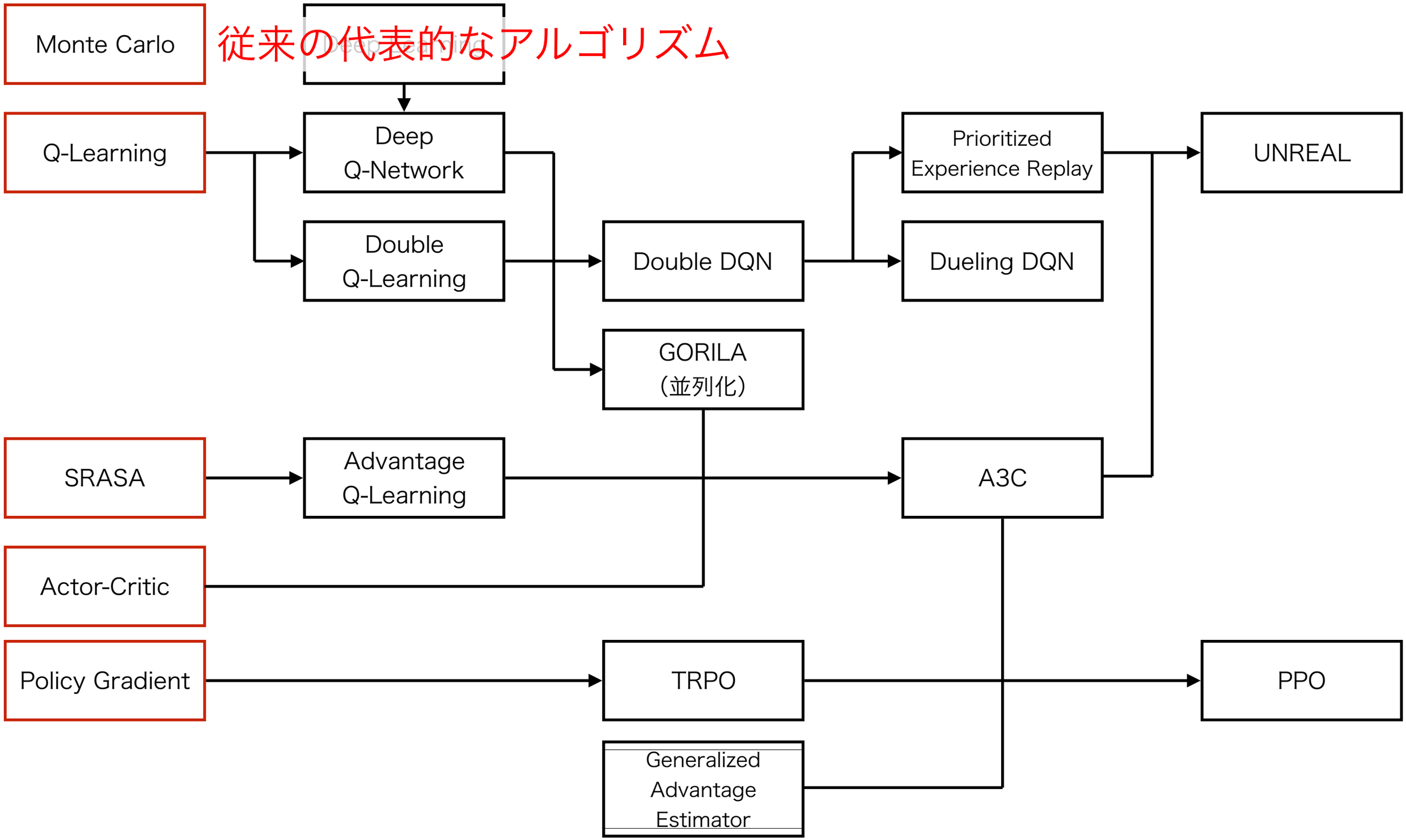
MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

強化学習アルゴリズムマップ



これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

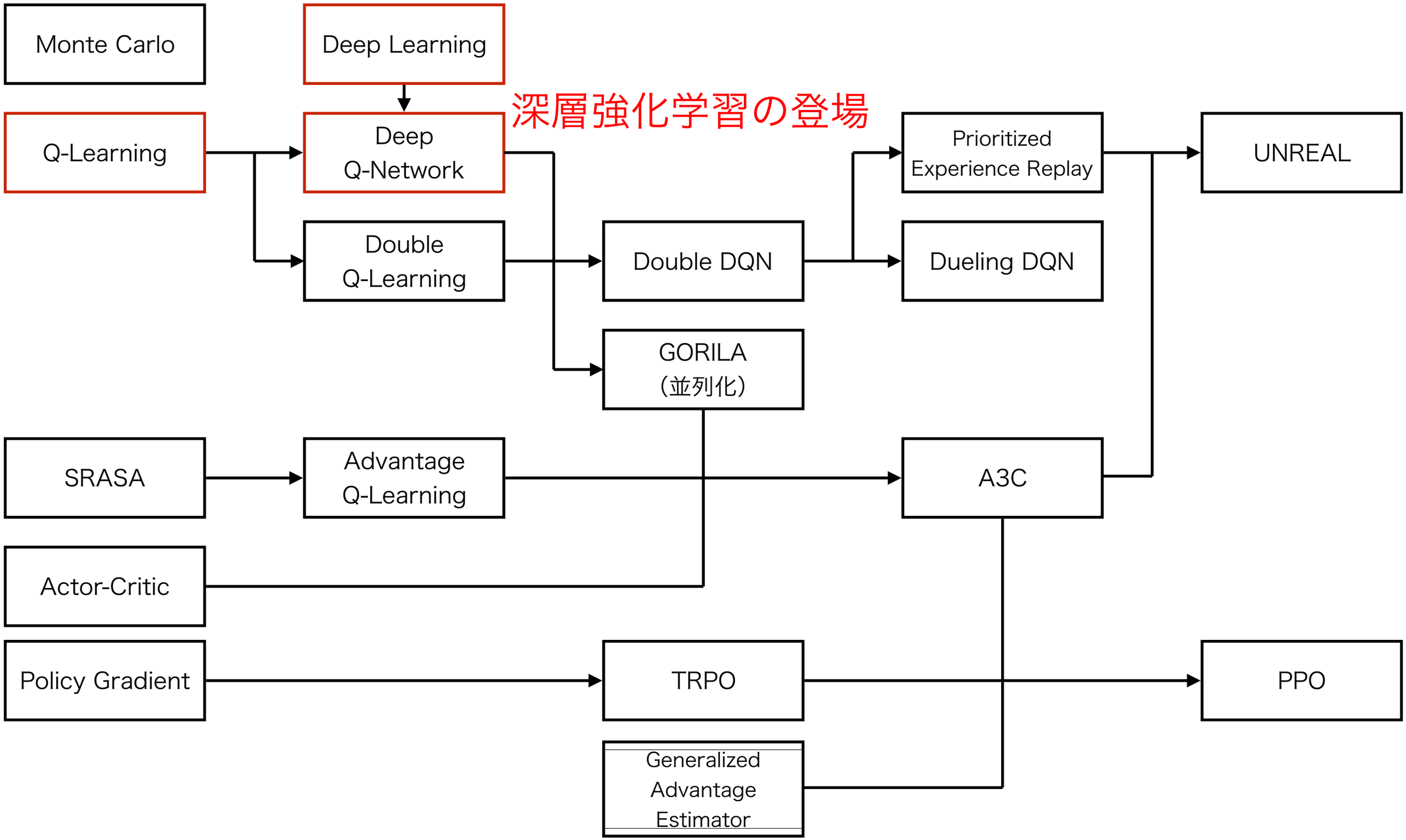
強化学習アルゴリズムマップ



これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

強化学習アルゴリズムマップ

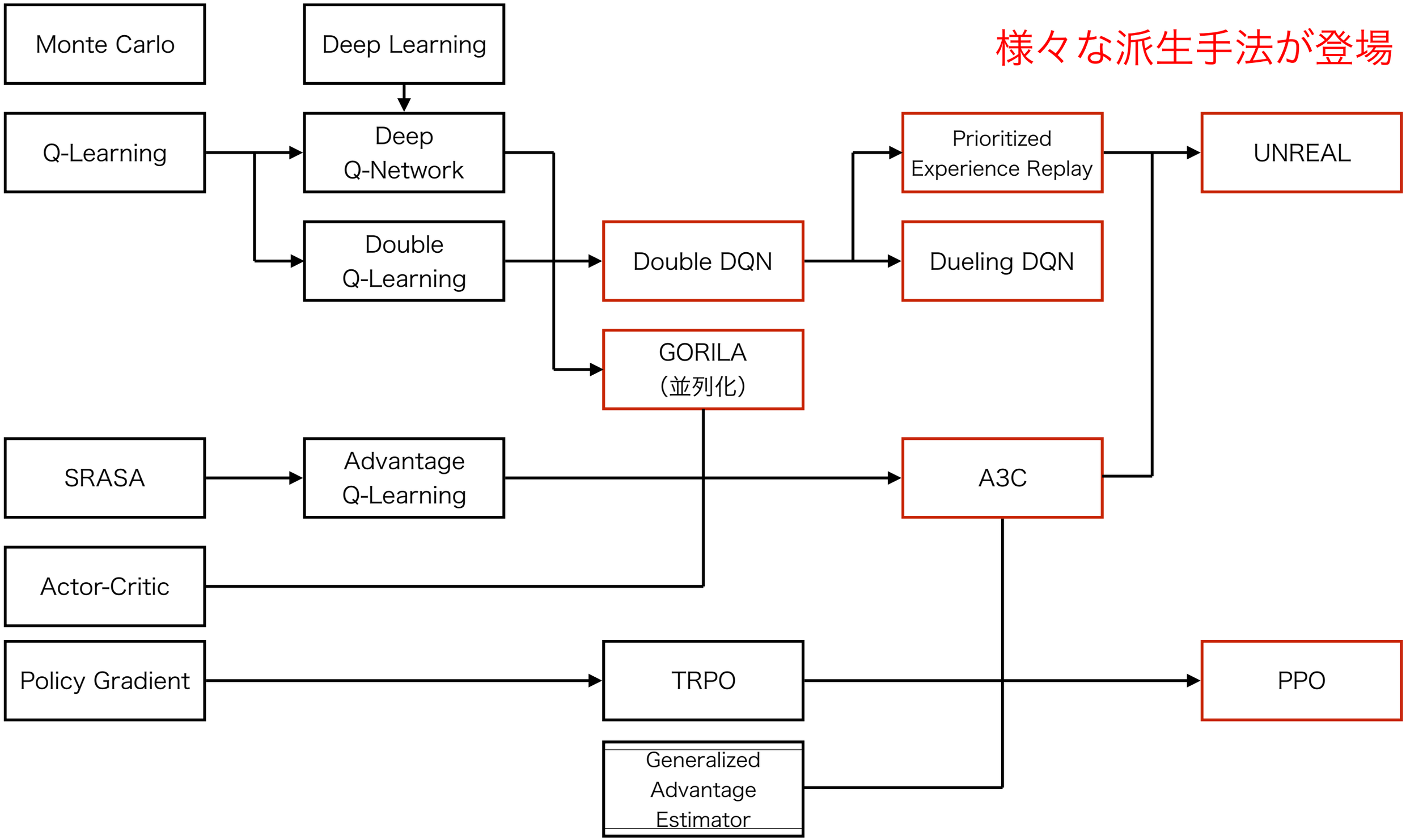


これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

強化学習アルゴリズムマップ

様々な派生手法が登場



これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

Q-Learning [Watkins and Dayan, 1992]

- 行動価値関数（Q関数）を学習
  - Q関数 … 状態sで行動aを取った時の良さを表す関数
  - Q関数を更新しながら最適な関数を推定

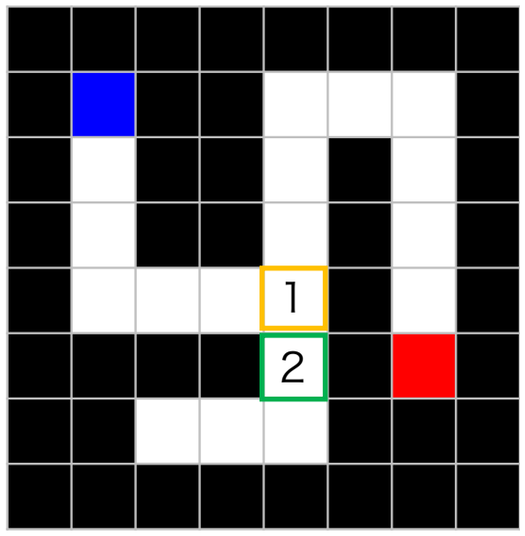
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right)$$

※ α：学習率

更新前のQ値と割引報酬和の差分を求めて，Q値を更新

Temporal Difference (TD) 誤差

- 最もシンプルな方法 => 表（Q-Table）に保存

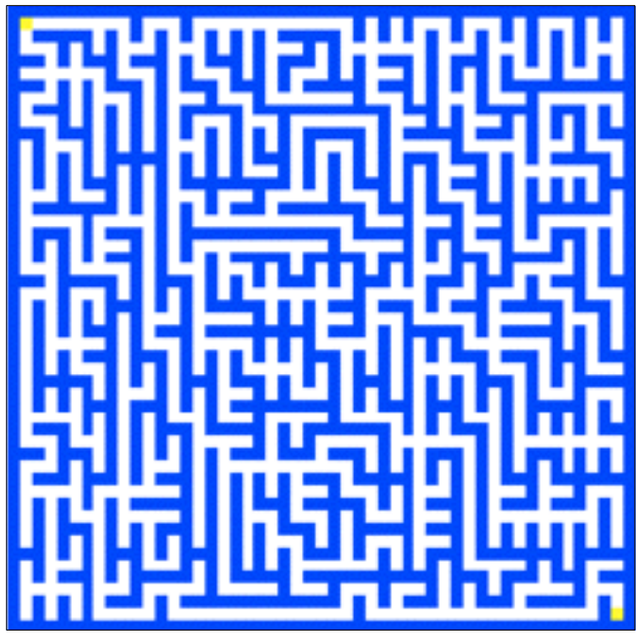


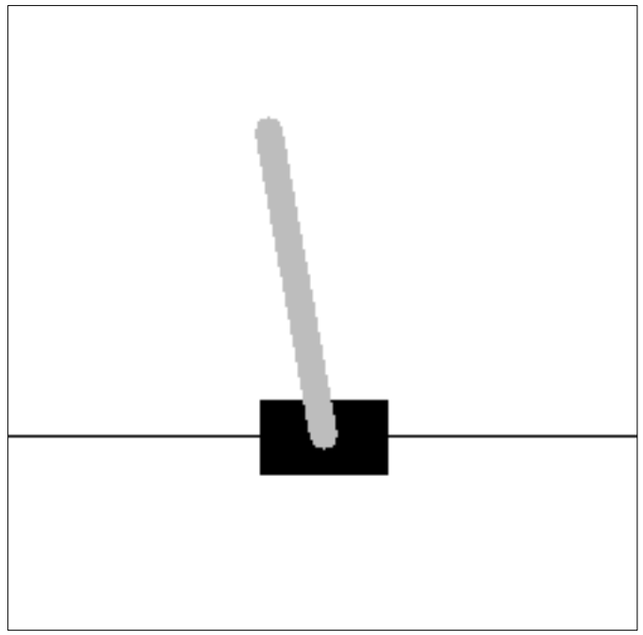
状態	行動↑	行動↓	行動←	行動→
s = 1	7	9	3	0
s = 2	4	7	0	0

Q-Table

Q-Learningの問題点

- 大規模な状態空間や連続空間での学習が困難
  - 迷路
    - 8x8の迷路の状態数 … 64
    - 100x100の迷路の状態数 … 10^4
  - 画像を入力する場合（CartPole）
    - 84x84のグレースケール画像の状態数 … 256^{84x84}





高次元な状態空間では，現実的に計算することが難しい

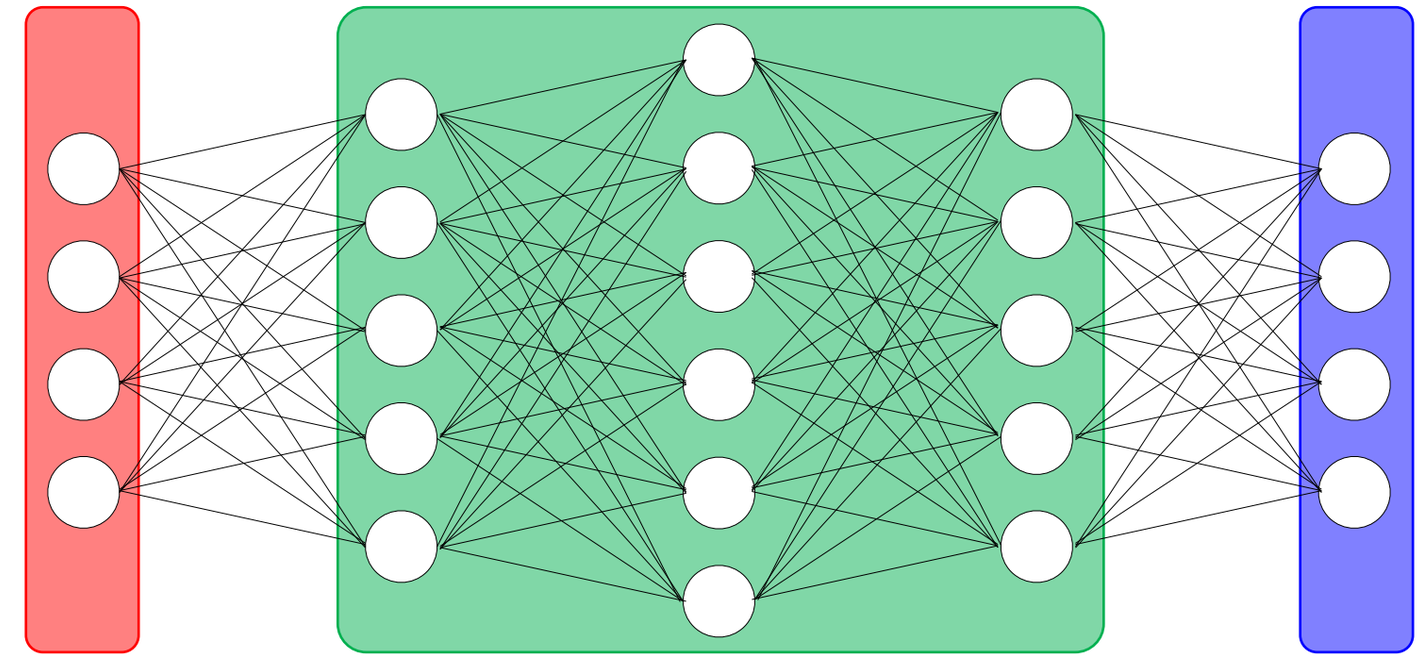
Q-Network [Tesauro, 1995]

- ニューラルネットワーク（NN）で関数近似

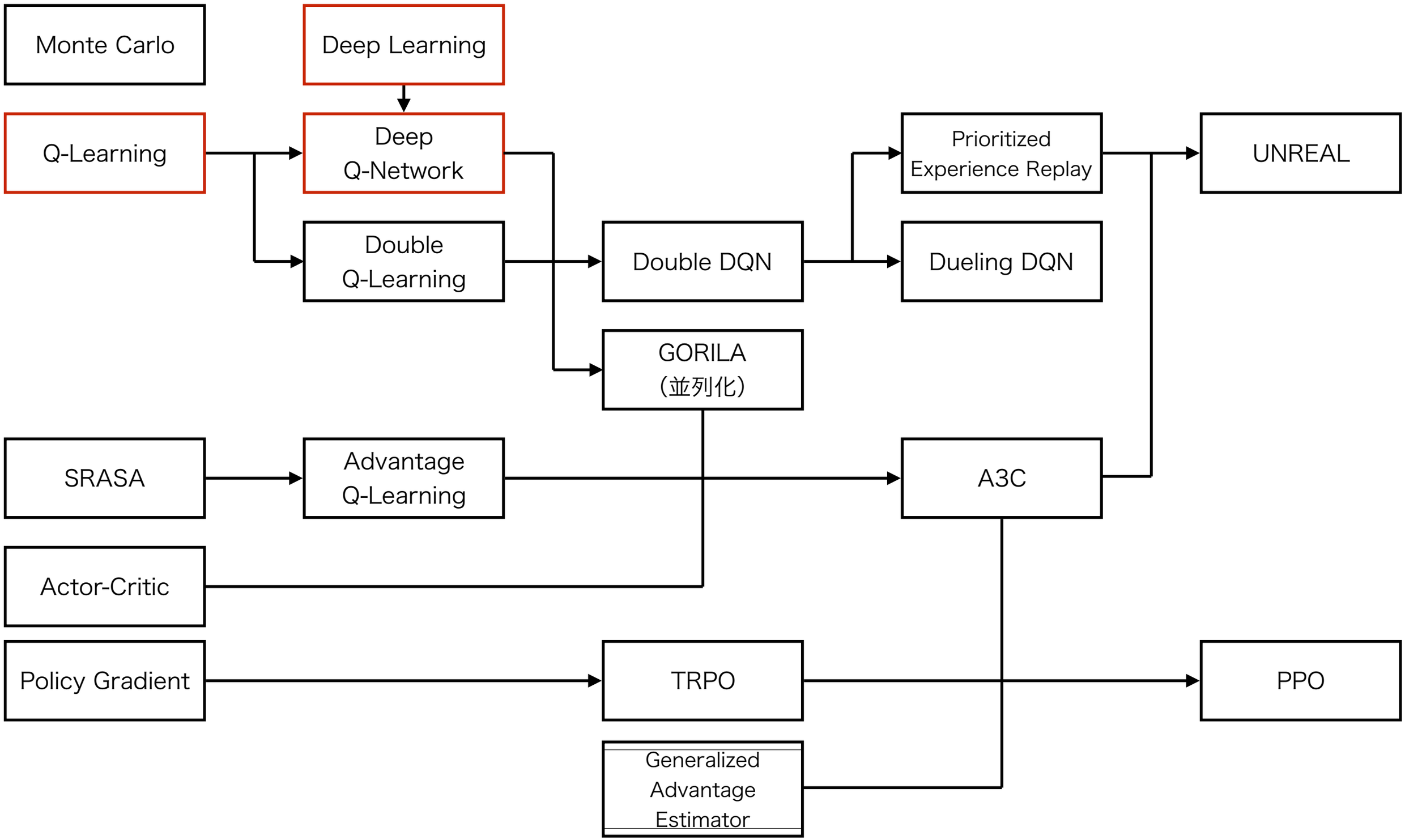
$$Q_\theta(s,a) \approx \theta^{\mathrm{T}}\phi(s,a)$$

- NNで近似（非線形）

$$Q_{\theta_1,\theta_2,\ldots,\theta_n}(s,a) \approx \theta_n^{\mathrm{T}}\phi_n(\ldots\theta_2^{\mathrm{T}}\phi_2(\theta_1^{\mathrm{T}}\phi_1(s,a))\ldots)$$



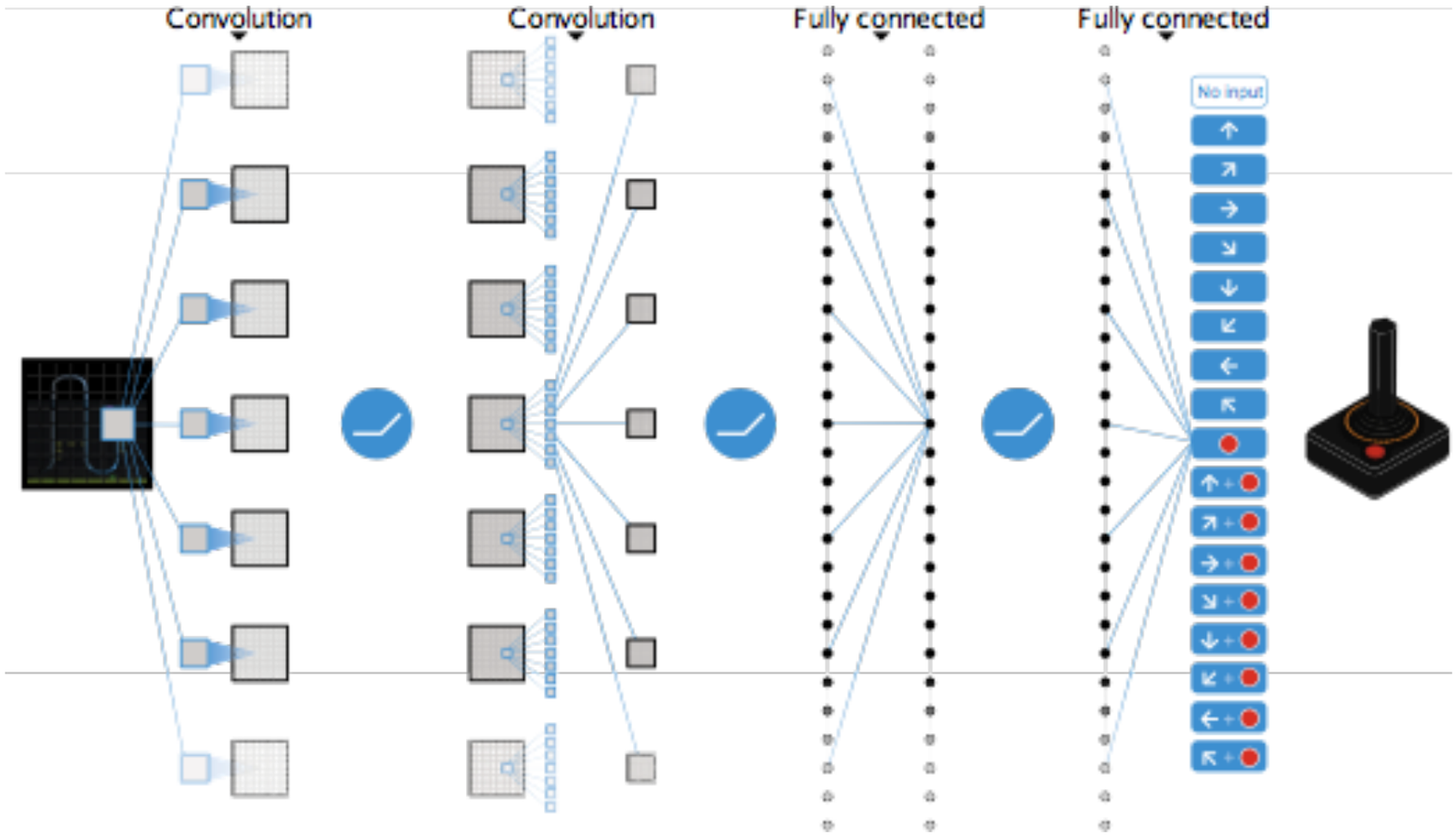
強化学習アルゴリズムマップ



これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

Deep Q-Network (DQN) [Mnih, et al., 2013 and 2015]

- Q-Networkに畳み込みニューラルネットワークを適用
  - 局所解に陥らないように学習するテクニックを導入



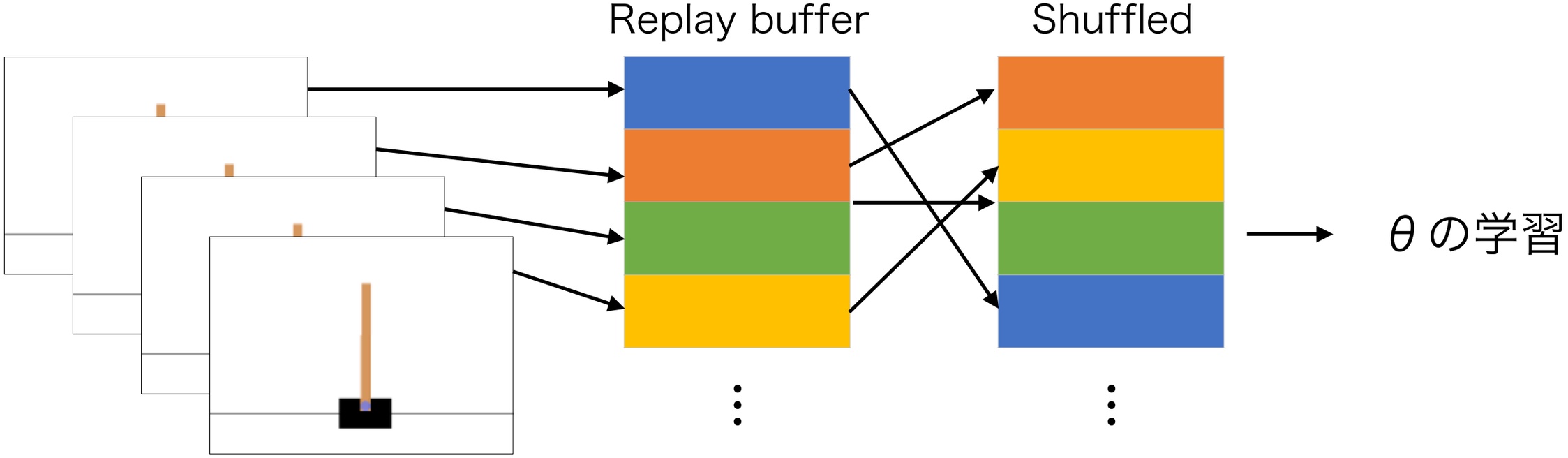
学習の安定化

- 単純な学習アルゴリズムでは学習が不安定
  - パラメータの値が振動・発散する

- 様々な工夫を取り入れることで安定化
  - Experience replay
    - 試行により獲得した行動をランダムに学習に使用
  - Target Q-Network
    - ネットワークのパラメータを固定
  - Reward Clipping
    - 報酬のスケールを一定

Experience replay [Lin, 1992]

- 過去の行動遷移（s_t, a_t, r_t, s_{t+1}）をreplay bufferに保存
  - ランダムに選択して学習に使用
    - 過学習を抑制
  - ミニバッチで学習



Target Q-Network

- ネットワークのパラメータを固定して学習
  - 従来の価値関数では目標値が変動
    - 少ない更新でもパラメータが発散してしまう
  - 目標値のQ関数を固定して学習
    - 発散が抑制される

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_{t+1} + \gamma \max_{a'} \boxed{Q(s_{t+1}, a')} - Q(s_t, a_t))$$

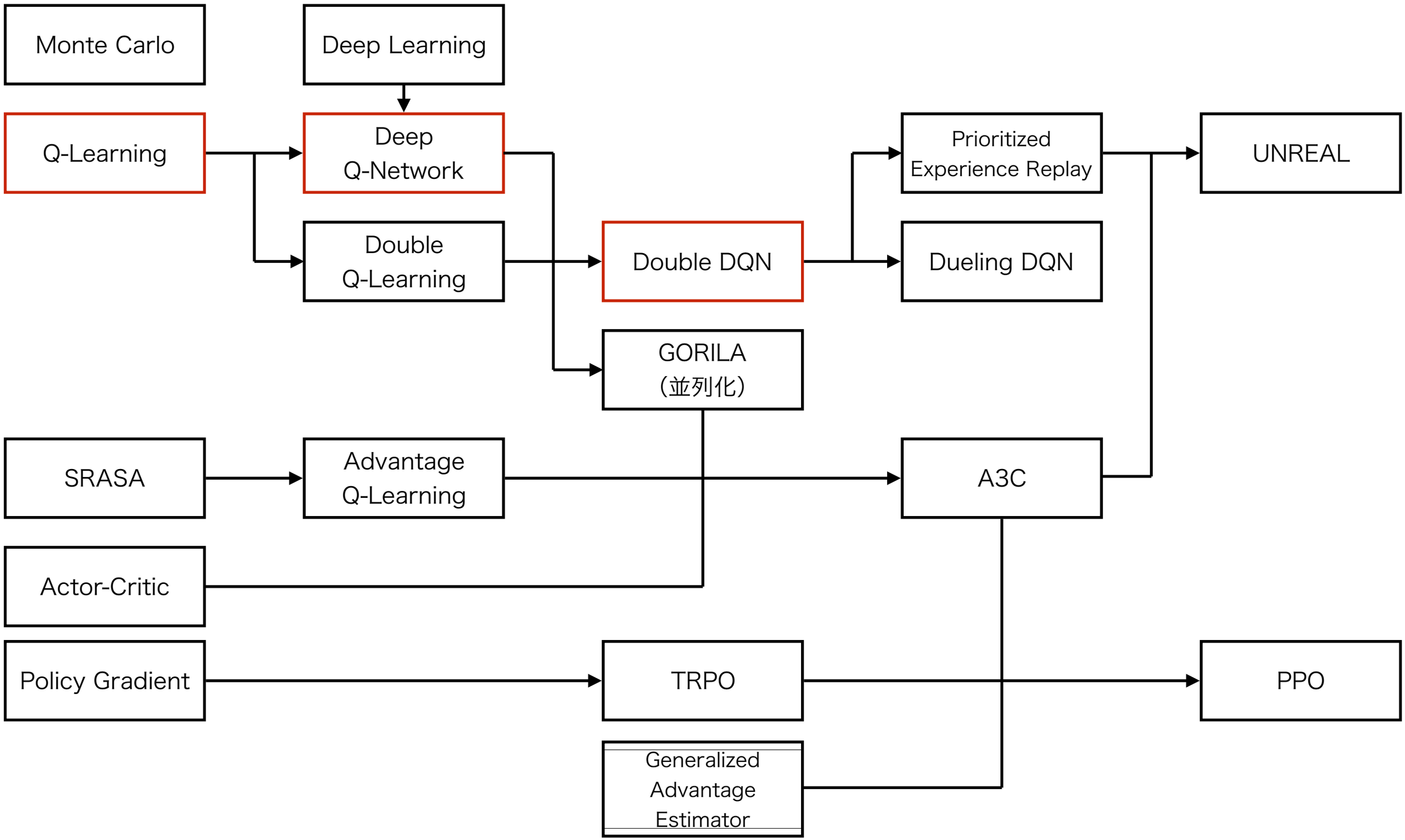
ここを固定

- 一定の更新回数で固定値を同期
  - 10000回に1回 など

Reward clipping

- 報酬を[-1，1]の範囲にクリップ
  - 報酬が負 … -1
  - 報酬が正 … 1
  - Q値の急激な増大を抑制
    - 勾配が安定する
- デメリット
  - 報酬の大小を区別できなくなる

強化学習アルゴリズムマップ



これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

Double DQN [Hasselt, et al., 2016]

- 2つのネットワークを用いて学習
  - 一つは行動選択
  - もう一つは行動価値の推定

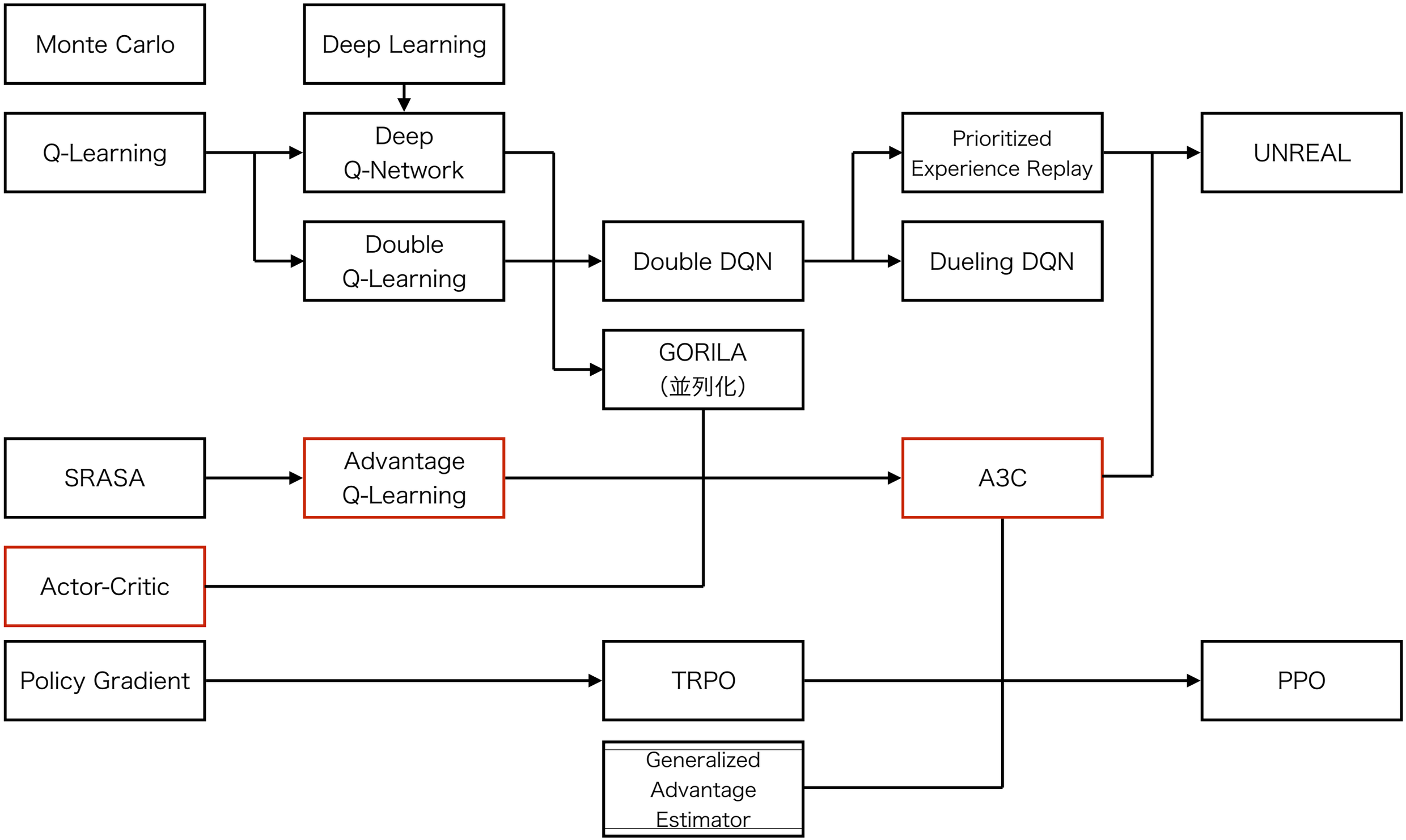
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_{t+1} + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t)\right)$$

$$r_{t+1} + \gamma Q(s_{t+1}, \mathrm{argmax}_{a'} Q(s_{t+1}, a; \theta_t), \theta_t^-)$$

- 過大評価を抑制可能

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

強化学習アルゴリズムマップ



これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

Advantage

- 2ステップ以上先まで行動して学習
  - 従来のQ関数 … 一つ先の行動まで考慮

$$Q(s,a) \rightarrow r_t + \gamma \max Q(s_{t+1}, a)$$

  - Advantage
    - 2ステップ以上先の報酬まで考慮
    - 現在の状態がより確からしくなる
    - 効率的に学習可能

$$Q(s,a) \rightarrow r_t + \gamma r_{t+1} + \gamma^2 \max Q(s_{t+2}, a)$$

Value-based と Policy-based

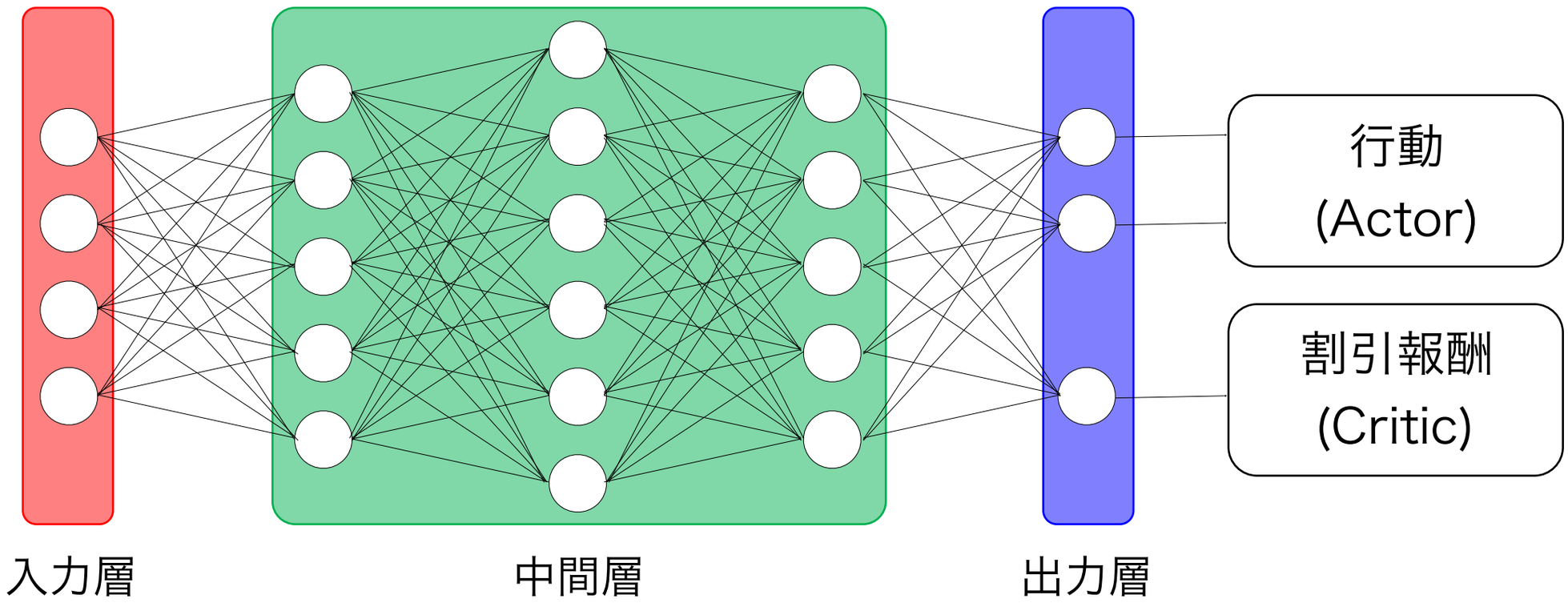
- Value-based
  - 状態sにおいて行動aをとった時に得られる行動価値をもとに学習
  - Q関数を用いた強化学習
    - Q-Learning, DQN, Double DQN, etc.
  - デメリット
    - 行動価値の変化が行動選択に大きく影響

- Policy-based
  - 状態sから行動aを直接決定する
  - デメリット
    - 学習が進みにくい（微分が難しい）

Actor-Critic [Konda and Tsitsiklis, 2000]

- Policy (actor) とValue (critic) を同時に学習する手法
  - Actor … 行動を出力
  - Critic … 割引報酬を出力

  - 双方のデメリットを補うことが可能
  - ActorとCriticの両方を更新して学習

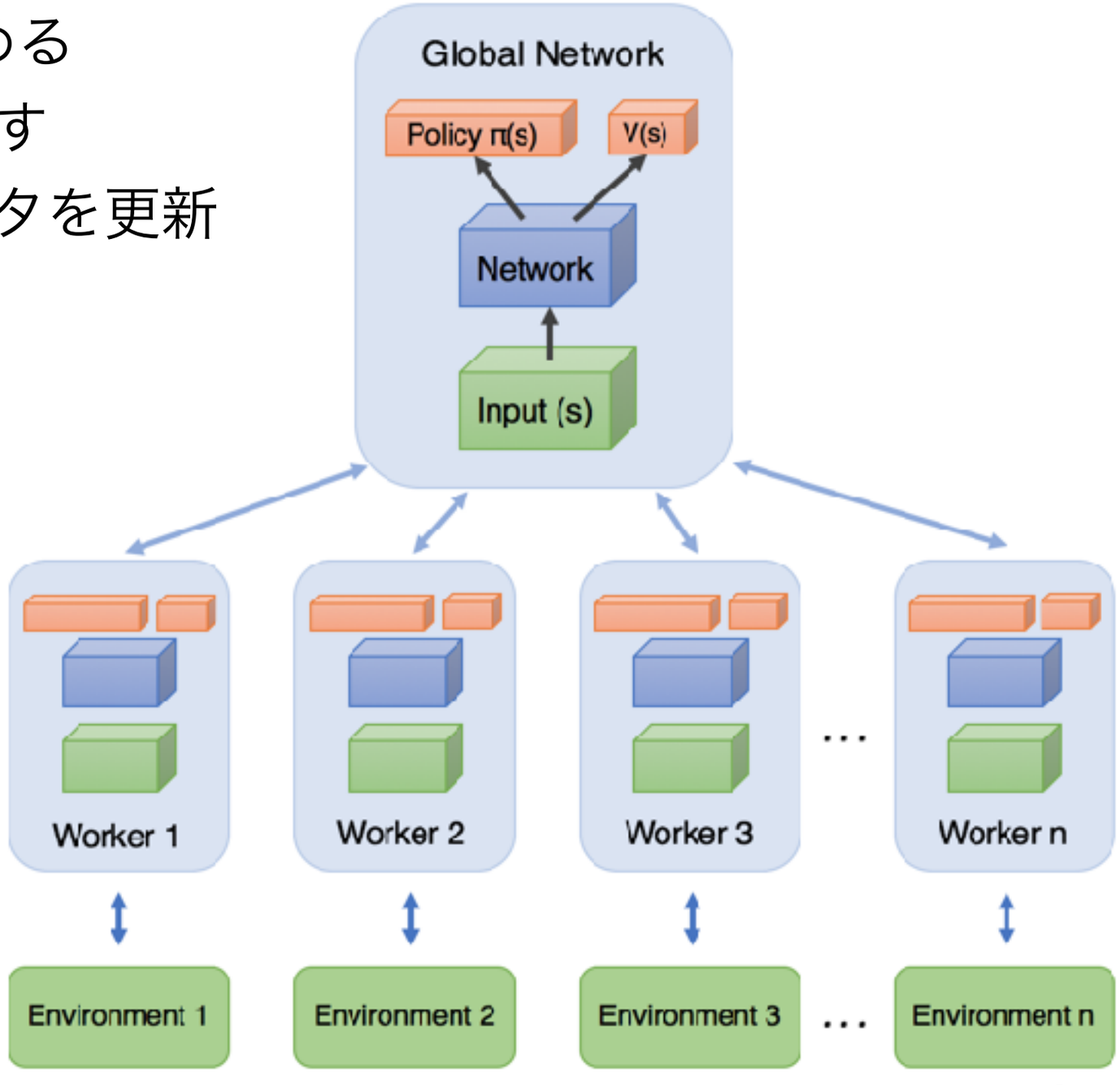


A3C [Mnih, et al., 2016]

- Asynchronous
  - 複数の学習環境を用意
  - 各環境のエージェントが経験を蓄積（分散学習）
- Advantage
  - 少し先の報酬まで考慮
- Actor-Critic
  - Actor-Criticネットワークを使用

A3Cの処理の流れ

1. Global Network (Parameter server) からパラメータをコピー
2. スレッドで行動を獲得・蓄積
3. 蓄積した行動から勾配を求める
4. 勾配をGlobal Networkに渡す
5. Global Networkのパラメータを更新
6. 1. へ戻る

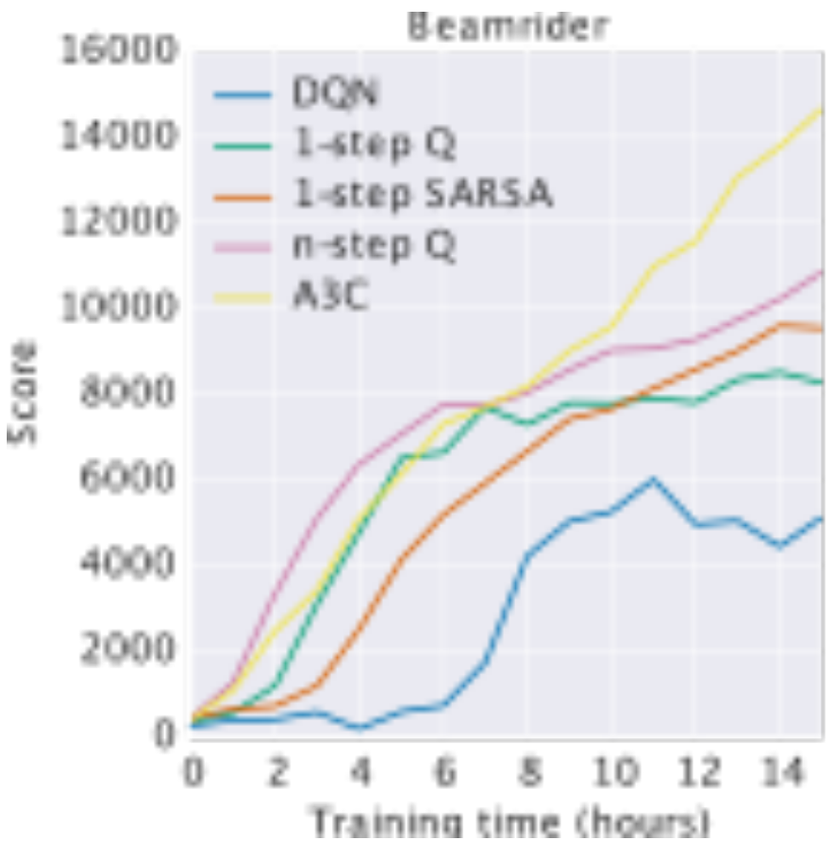


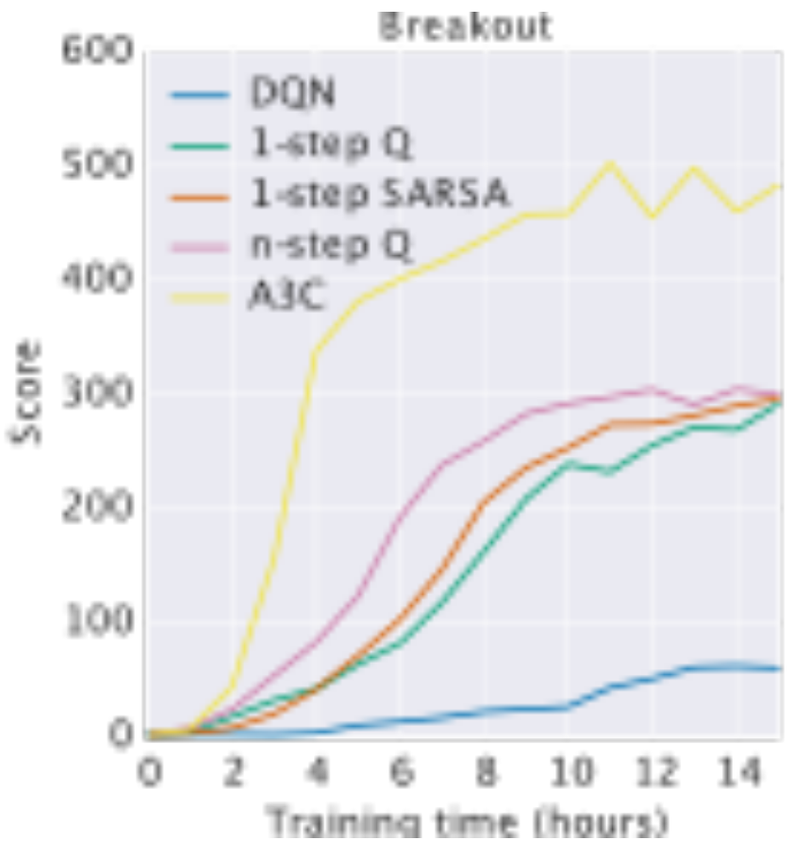
https://medium.com/emergent-future/

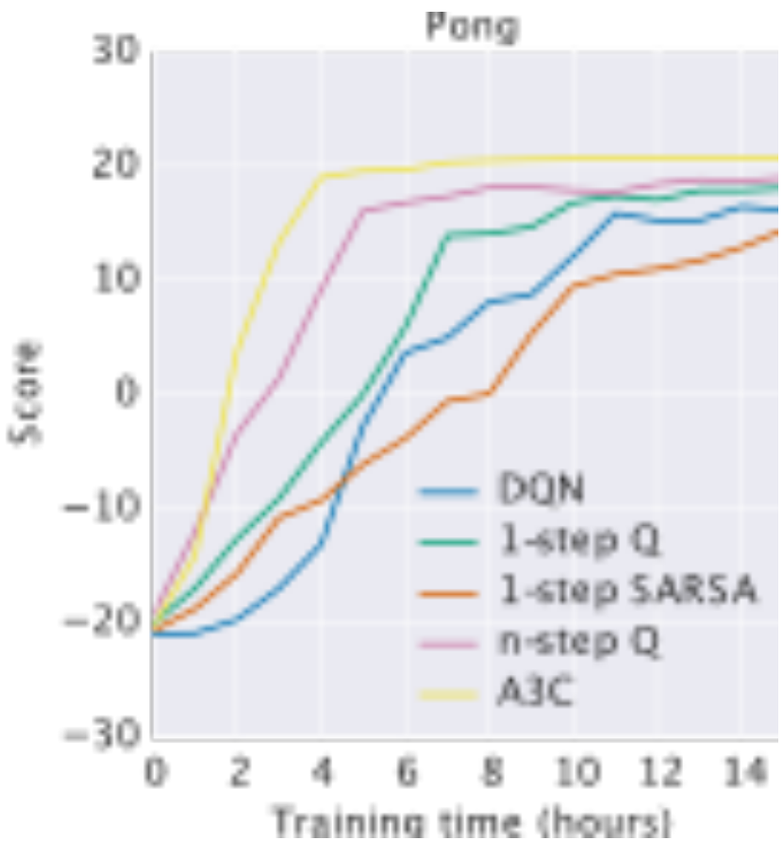
MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

A3C ~実験結果~

- Atari2600で実験
  - 高い性能を獲得

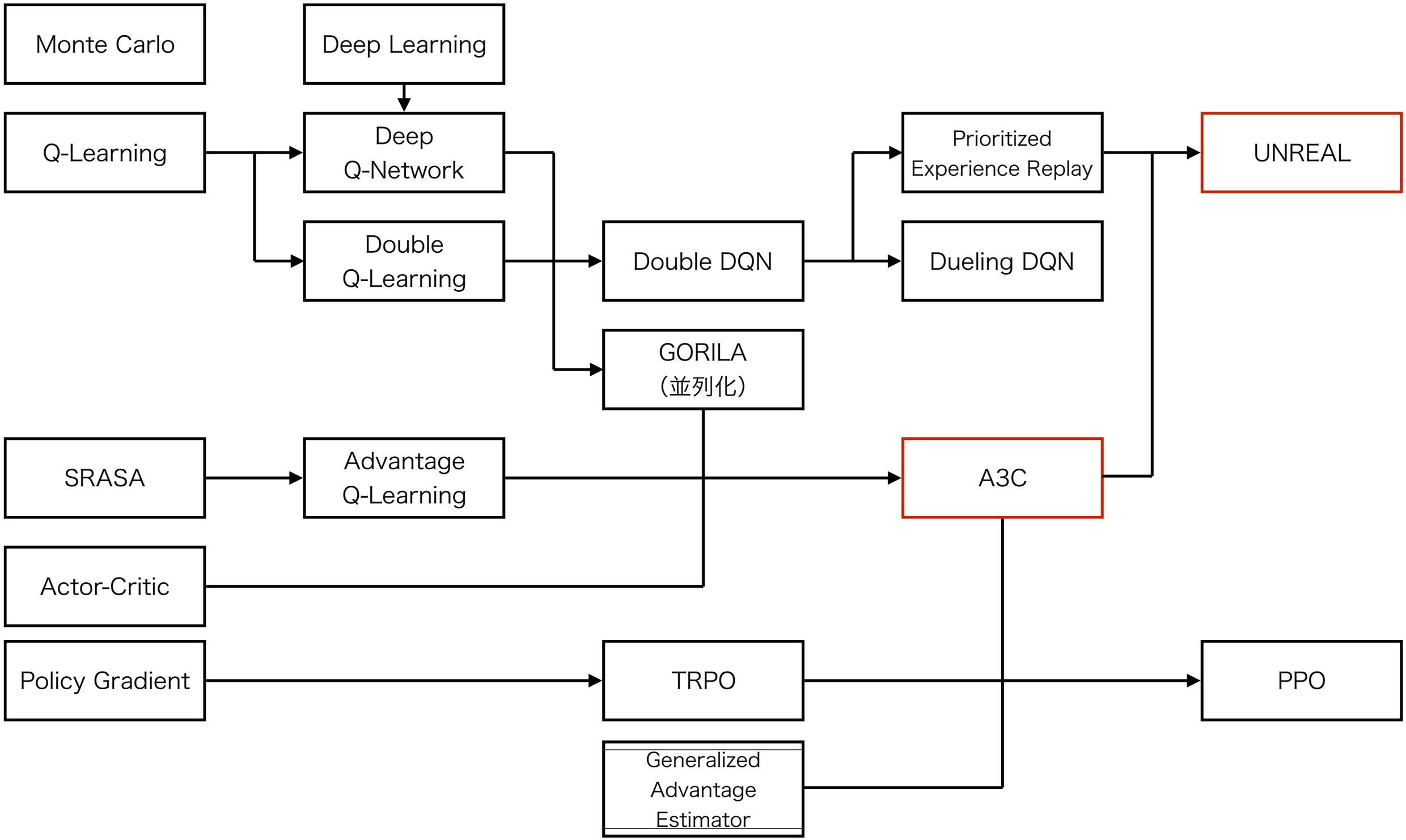






MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

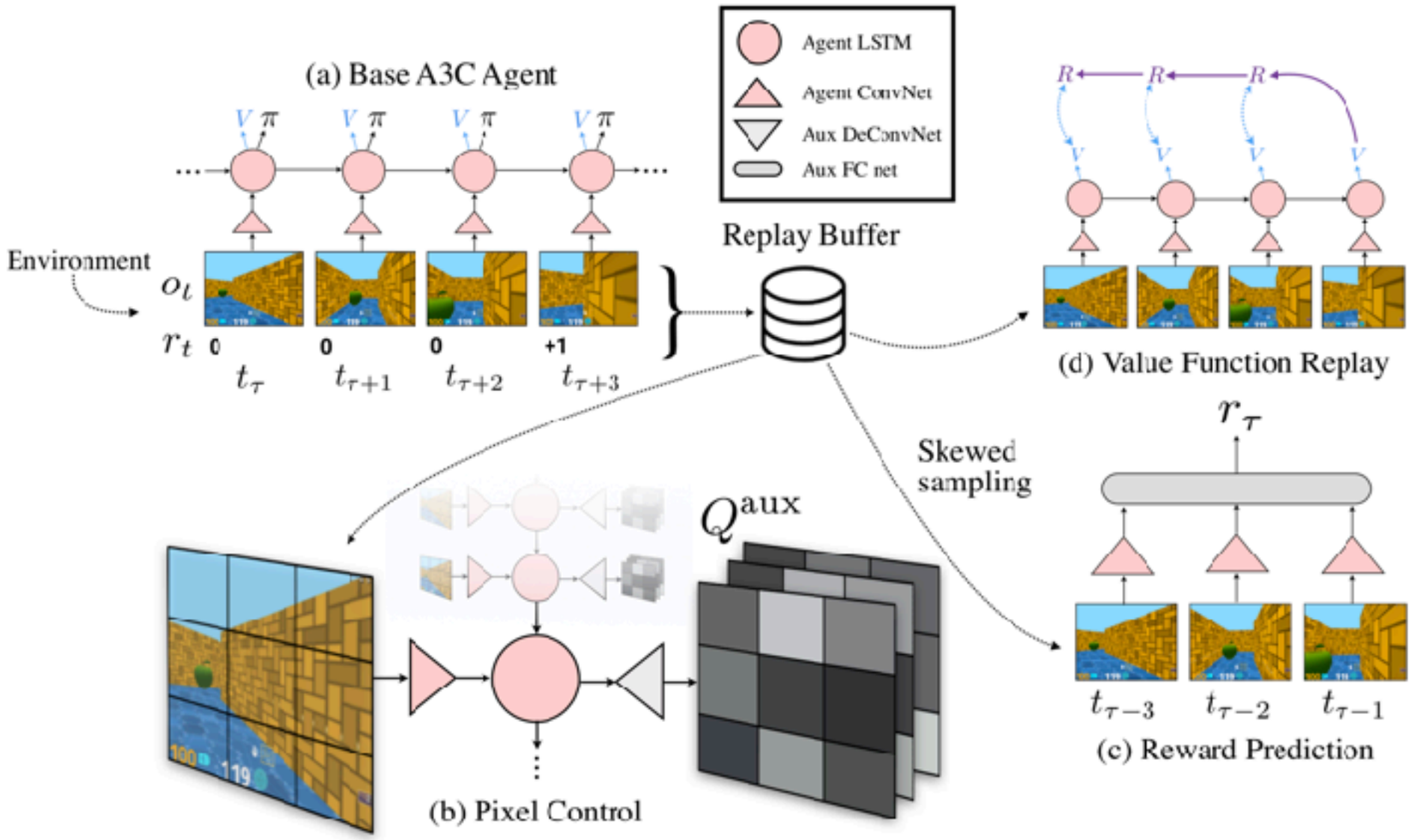
強化学習アルゴリズムマップ



これから強化学習を勉強する人のための「強化学習アルゴリズムマップ」と、実装例まとめ
https://qiita.com/sugulu/items/3c7d6cbe600d455e853b

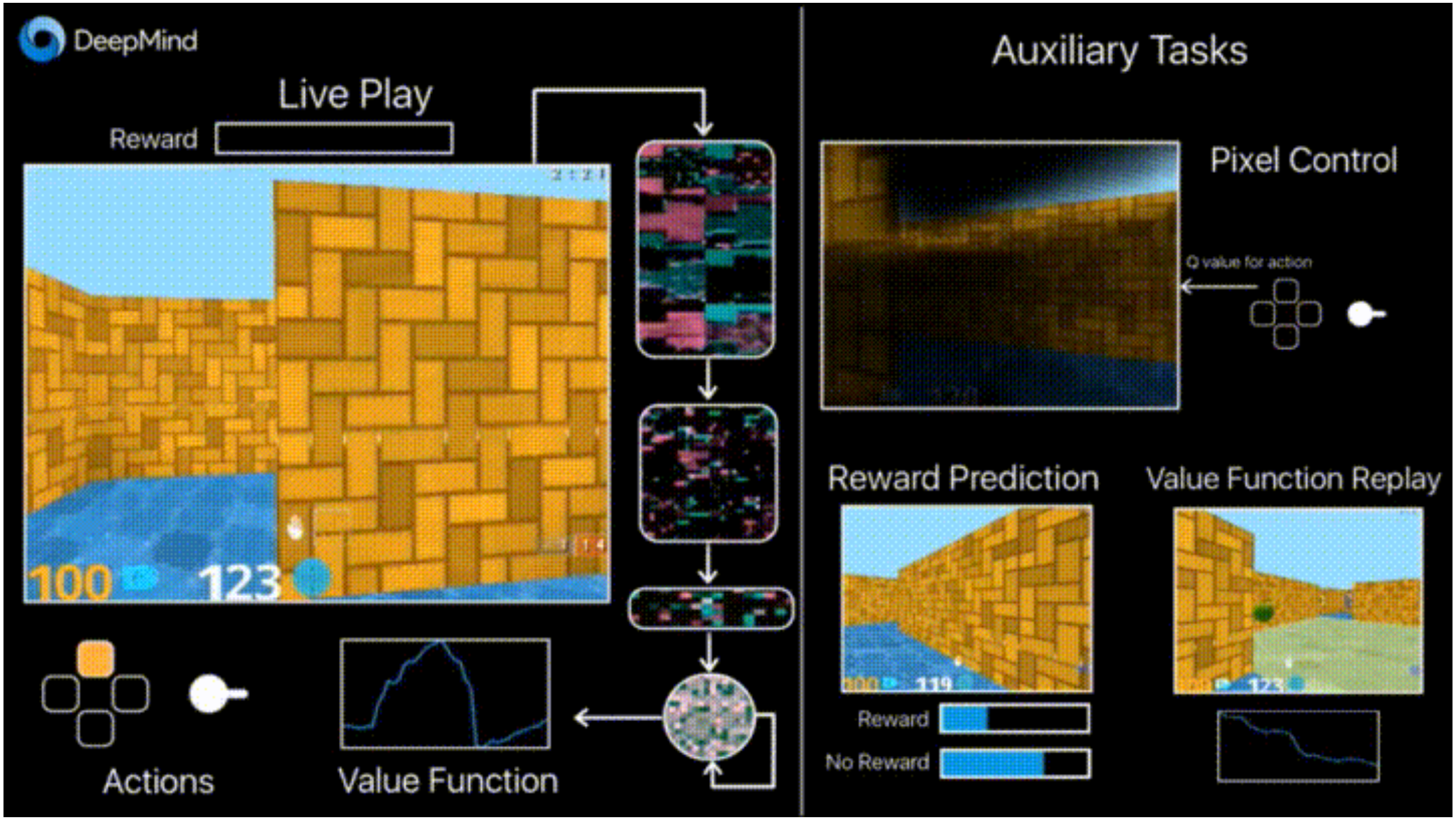
UNREAL [Jaderberg, et al., 2016]

- A3Cをベースに補助タスクを同時に学習
  - Pixel Control … 画像の画素値が大きく変動する動きを学習
  - Value Function Replay … 過去の経験をシャッフルして学習
  - Reward Prediction … 現在の状態から報酬を予測する学習



UNREAL ~実験結果~

- 3次元の迷路タスクで高い性能を達成
  - 本来のタスクと共有している部分が学習される
    - 本来のタスクの性能も向上

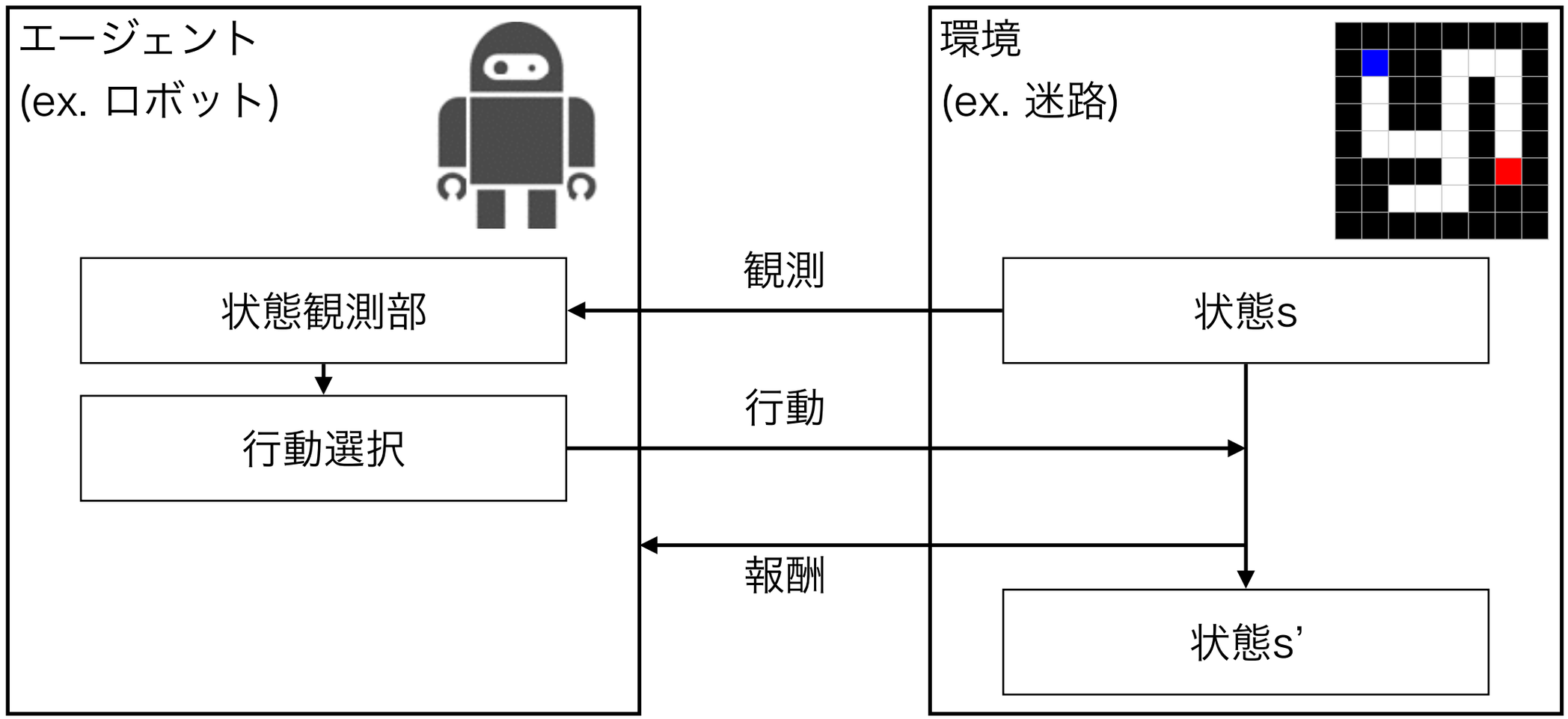


https://deepmind.com/blog/reinforcement-learning-unsupervised-auxiliary-tasks/

強化学習のフレームワーク

強化学習を行うために

- エージェントと行動する環境が必要



代表例

- ChainerRL

代表例

- OpenAI Gym
- MuJoCo

OpenAI Gym

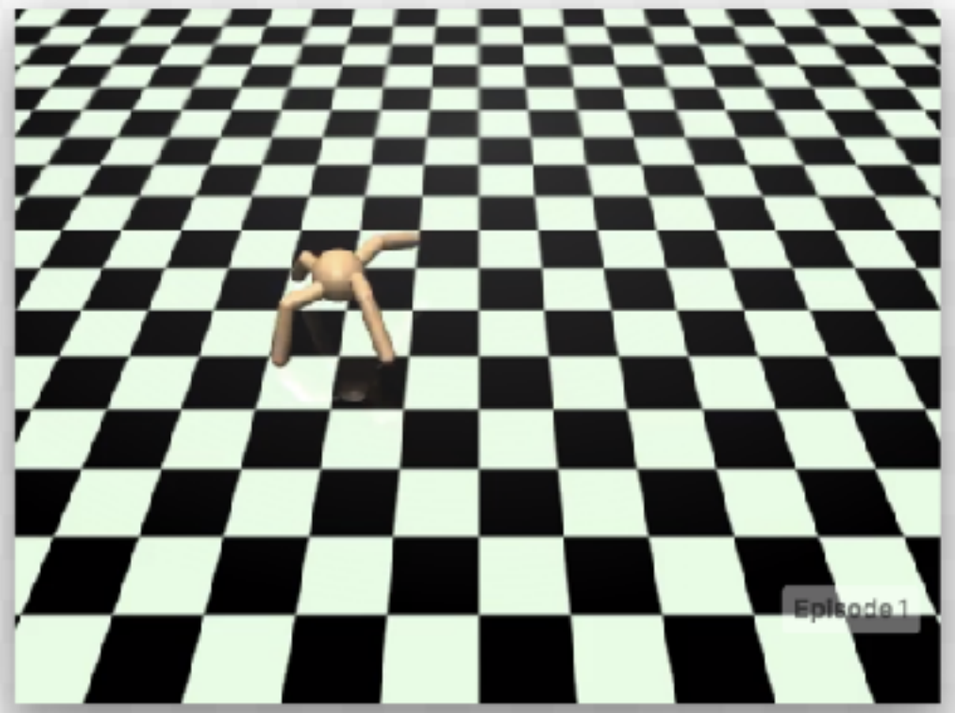
- 強化学習用シミュレーションプラットフォーム
  - OpenAIが開発・公開
    - AI研究を行う非営利団体
  - オープンソース
  - Pythonライブラリとして公開

Gym

Gym is a toolkit for developing and comparing reinforcement learning algorithms. It supports teaching agents everything from walking to playing games like Pong or Pinball.

View documentation ›

View on GitHub ›



RandomAgent on Ant-v2

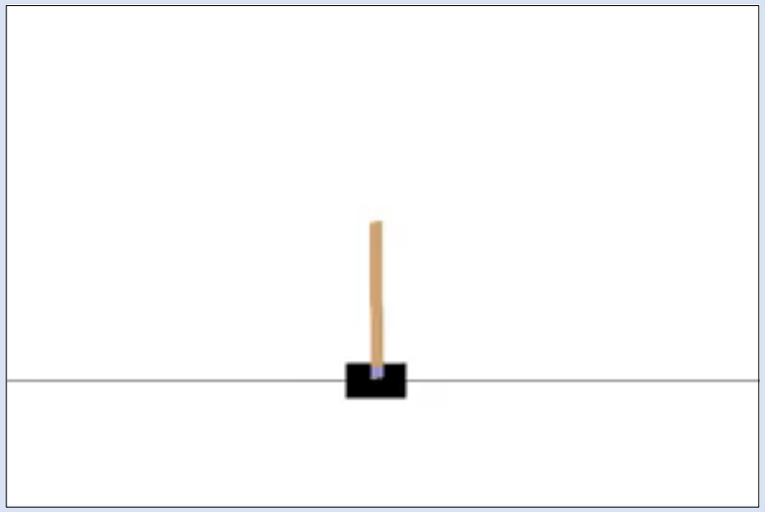
https://gym.openai.com/

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

OpenAI Gym

- 様々な環境が用意されている

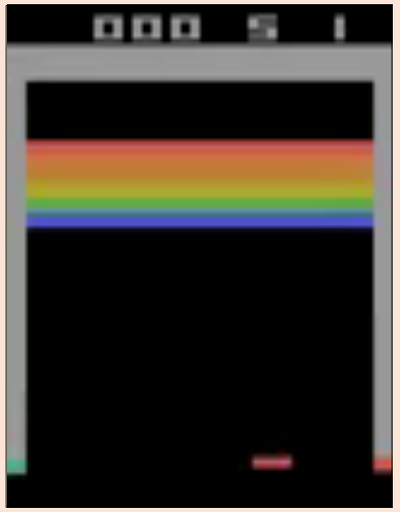
Classic control



CartPole

Atari

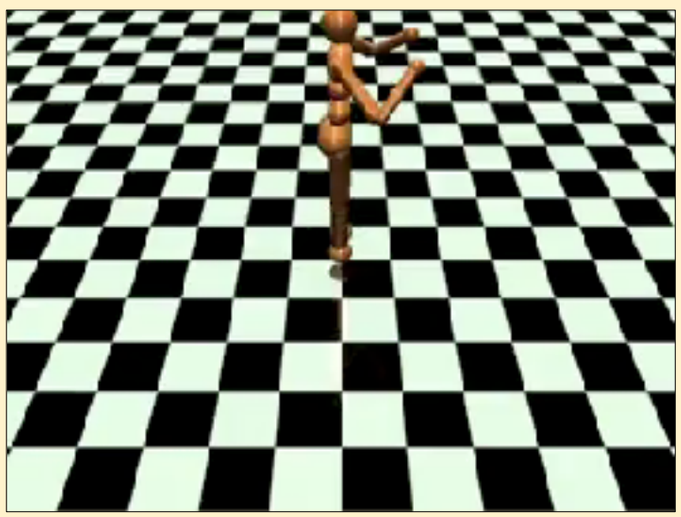
Pong



Breakout

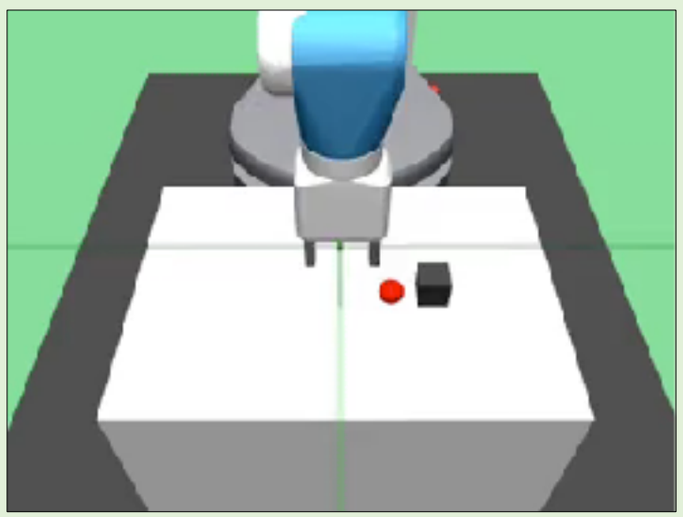
Boxing

MuJoCo

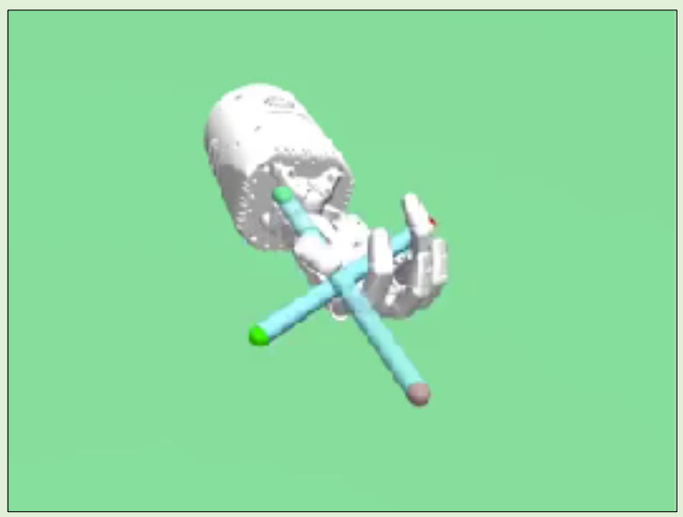


Humanoid

Robotics



Picking



Hand manipulate

https://gym.openai.com/

OpenAI Gymの使い方 ~インストール~

- 環境
  - Ubuntu 14.04
  - Python 2.7 または 3.5
  - pip (最新版)
  - Cmake (フルインストール時)

- インストール
  - GitHubからパッケージをクローン
  - pipでインストール

```
git clone https://github.com/openai/gym.git
cd gym
// 簡易インストール

pip install -e .
// フルインストール

pip install -e .[all]
```

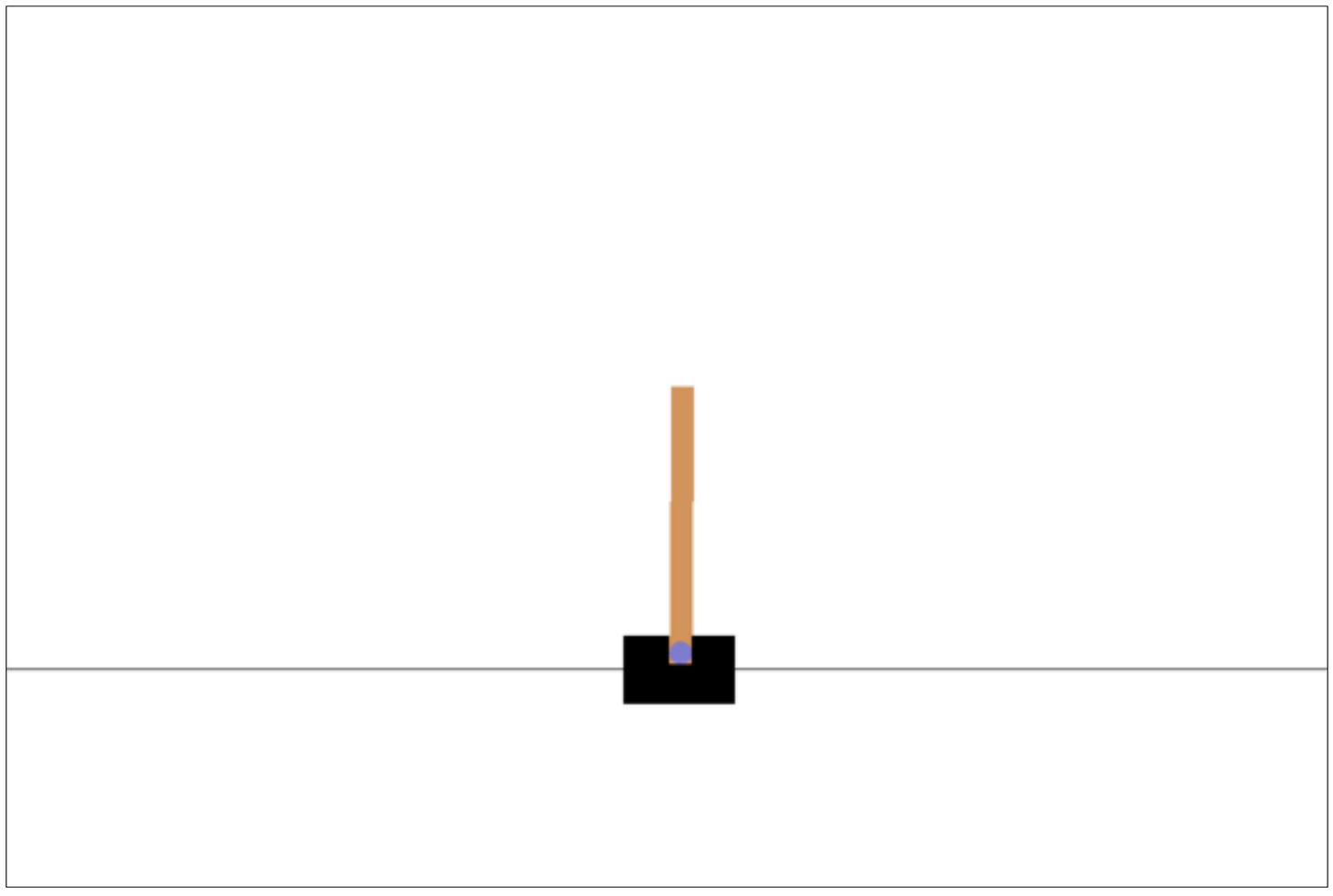
MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

OpenAI Gymの使い方 ~CartPoleの実行~

```
import gym
env = gym.make('CartPole-v0')
env.reset()  // 初期化
env.render() // 表示
```

←使用する環境名に設定

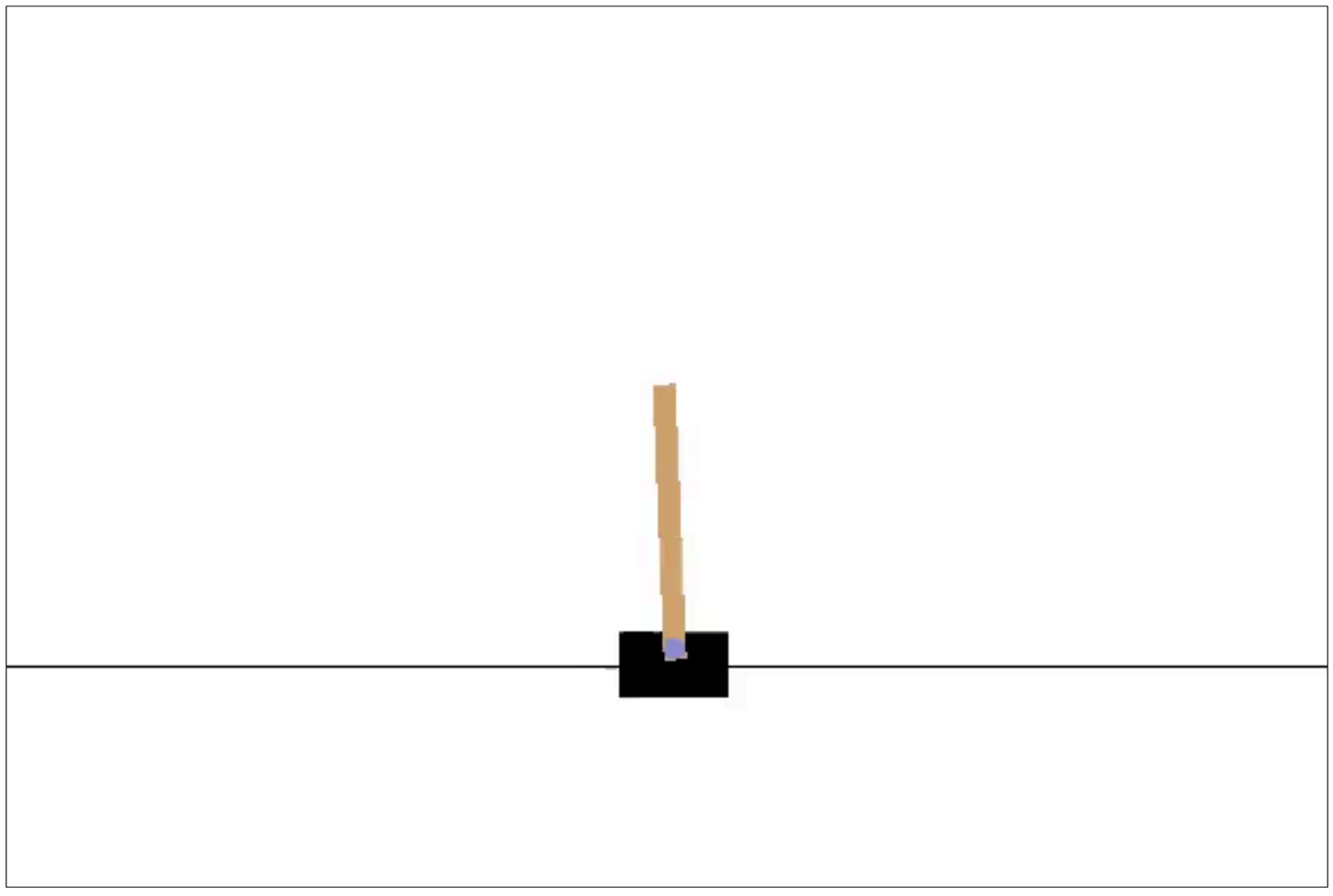
出力



OpenAI Gymの使い方 ~CartPoleの実行~

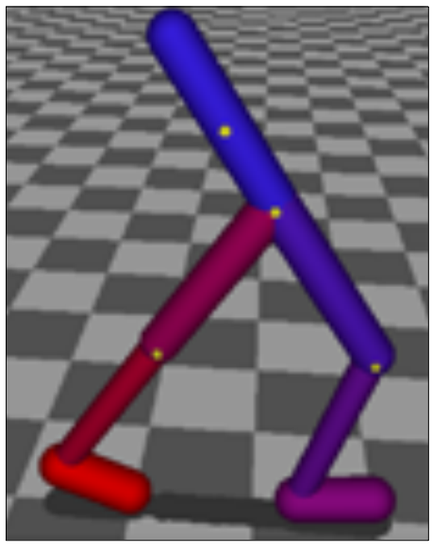
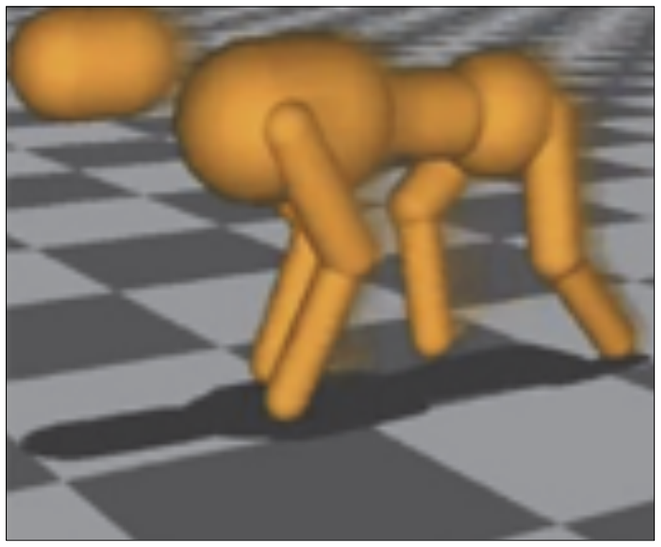
```
import gym
env = gym.make('CartPole-v0')
env.reset()                                          // 初期化

for _ in range(1000):
    env.render()
    action = env.action_space.sample()               // 適当に行動を選択

    env.step(action)                                 // 行動を入力
```

出力



MuJoCo

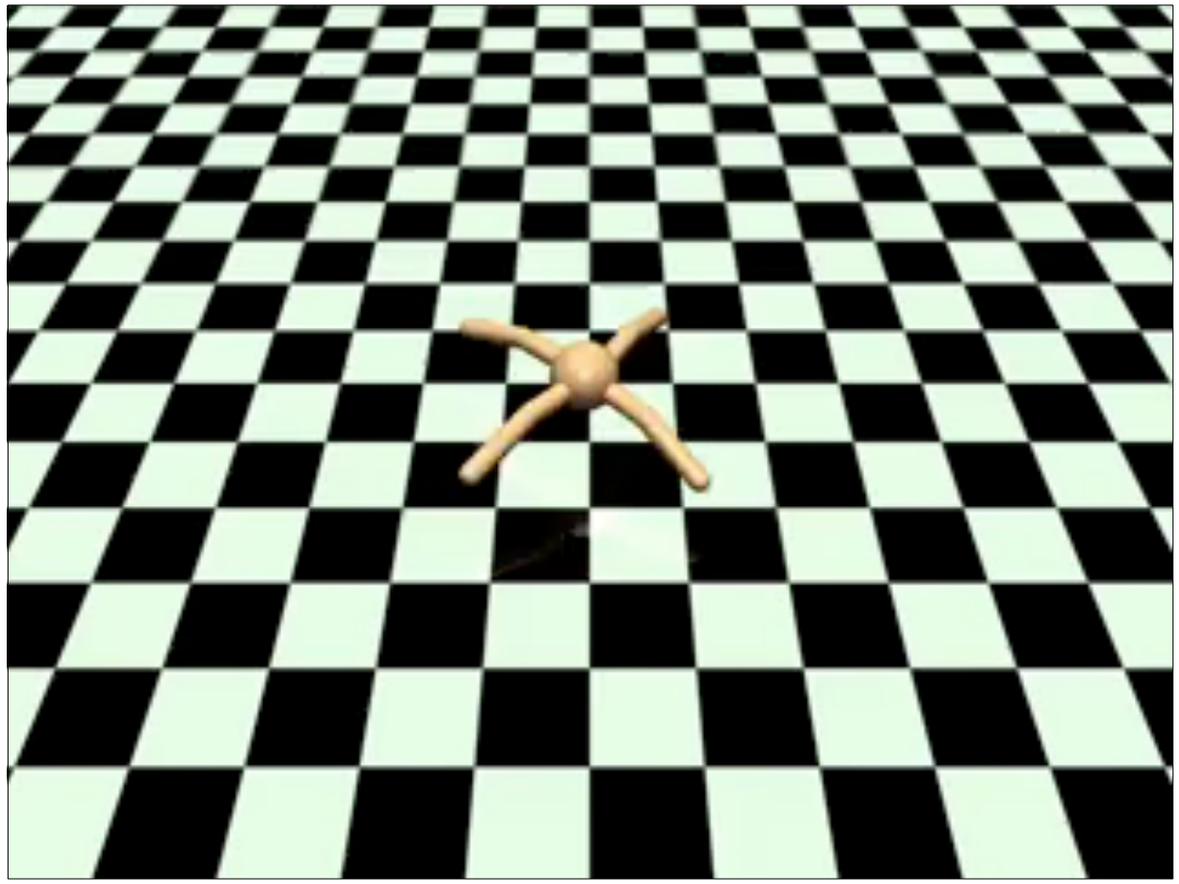
- 物理演算エンジン
  - 3次元シミュレーション等に使用

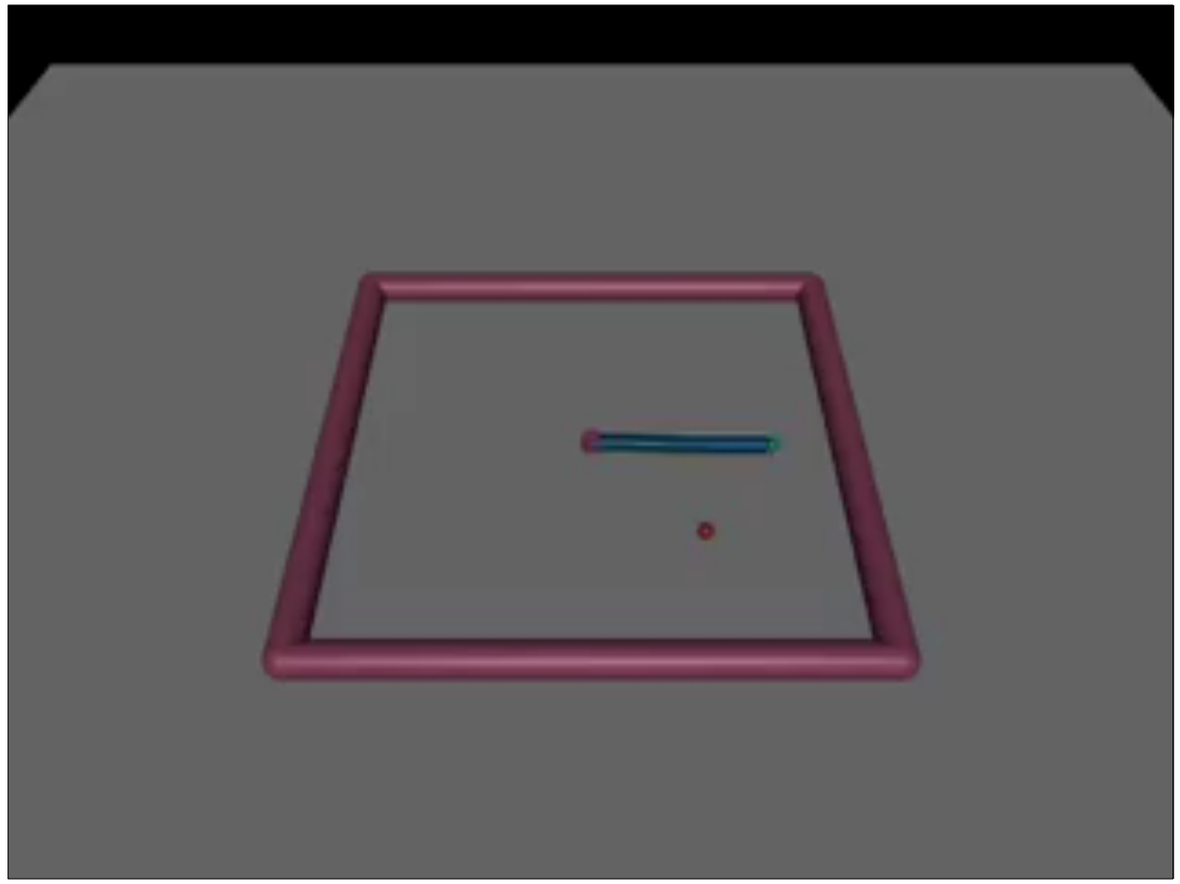



http://www.mujoco.org/

MuJoCo

- 物理演算エンジン
  - 3次元シミュレーション等に使用
  - OpenAI Gym内で利用可能
    - Python 3.* 推奨





https://gym.openai.com/

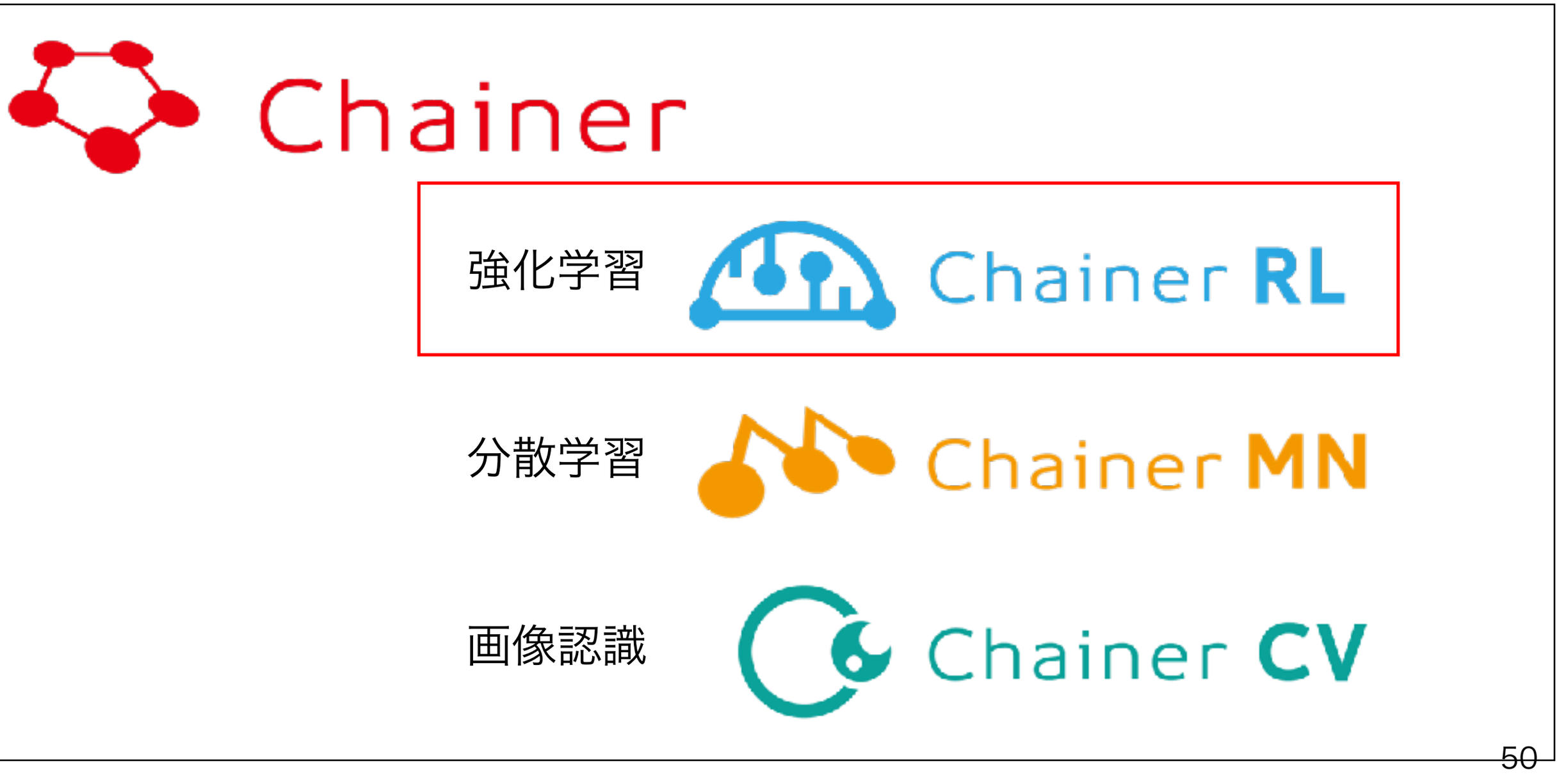
MuJoCoの使用方法

- ライセンス登録が必要
  - 基本的に有料
    - 学生は1年間無料
    - 30日間の無料トライアル

Personal Student	Free	Available to full-time students for use in personal projects and class work only. The software may not be used as part of employment, or in projects receiving financial support (see license text). Requires school email address.
Personal Non-commercial	$500	Available to anyone for use in personal projects. Also available to students, faculty and staff at academic institutions for use in research and education.
Personal Commercial	$2,000	Available to anyone for use in any project.

ChainerRL

- 深層強化学習用ライブラリ
  - Preferred Networksが開発
  - Chainerの追加パッケージの一つ



MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

ChainerRLの使い方 ~CartPole~

1. 必要なパッケージをインポート

```
import chainer
import chainer.functions as F
import chainer.links as L
import chainerrl
import gym
import numpy as np
```

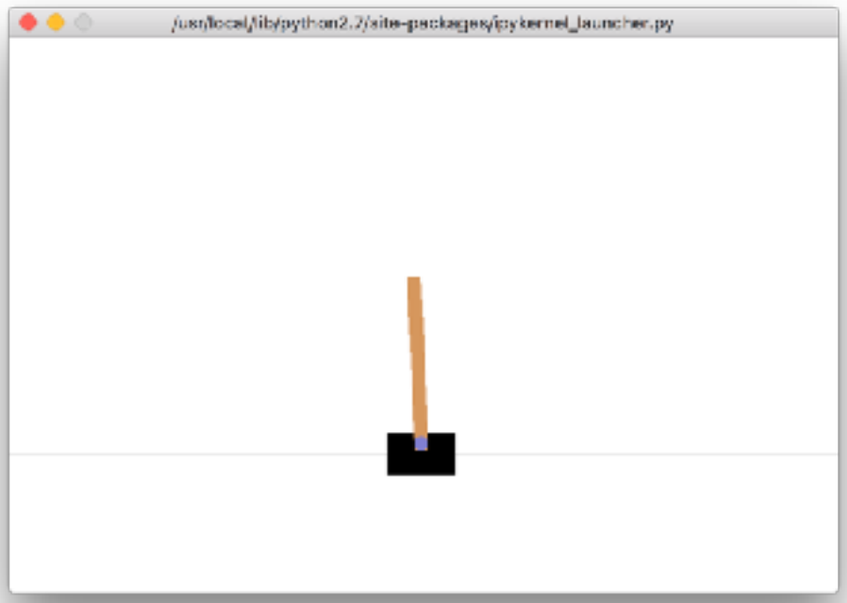
ChainerRLの使い方 ~CartPole~

2. CartPoleの環境を作成・初期化

```
env = gym.make('CartPole-v0’)
print("observation space   : {}".format(env.observation_space))
print("action space        : {}".format(env.action_space))

obs = env.reset()
env.render()
print(“observation : {}”.format(obs))
```

出力

```
observation space   : Box(4,)
action space        : Discrete(2)
observation : [-0.0169323 -0.0251642 -0.039872   0.0498410]
```



MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

ChainerRLの使い方 ~CartPole~

3. Q関数 (ネットワーク) の作成

```
class QFunction(chainer.Chain):
    def __init__(self, obs_size, n_actions, n_hidden_channels=50):
        # For Python 2.*
        super(QFunction, self).__init__(
        # For Python 3.*
        super(self).__init__(
            l0=L.Linear(obs_size, n_hidden_channels),
            l1=L.Linear(n_hidden_channels,n_hidden_channels),
            l2=L.Linear(n_hidden_channels, n_actions))

    def __call__(self, x):
        h = F.tanh(self.l0(x))
        h = F.tanh(self.l1(h))
        return chainerrl.action_value.DiscreteActionValue(self.l2(h))

obs_size = env.observation_space.shape[0]
n_actions = env.action_space.n
q_func = QFunction(obs_size, n_actions)

# GPUを使いたい人
q_func.to_gpu(0)
```

ChainerRLの使い方 ~CartPole~

4. 学習の設定

```
# 最適化手法
optimizer = chainer.optimizers.Adam(eps=1e-2)
optimizer.setup(q_func)

# 割引率
gamma = 0.95

# epsilon greedy法
explorer = chainerrl.explorers.ConstantEpsilonGreedy(
    epsilon=0.3, random_action_func=env.action_space.sample)

# experience replay
replay_buffer = chainerrl.replay_buffer.ReplayBuffer(capacity = 10**6)

phi = lambda x:x.astype(np.float32, copy=False)
agent = chainerrl.agents.DQN(
    q_func, optimizer, replay_buffer, gamma, explorer,
    replay_start_size=500, update_interval=1,
    target_update_interval=100, phi=phi)
```

ChainerRLの使い方 ~CartPole~

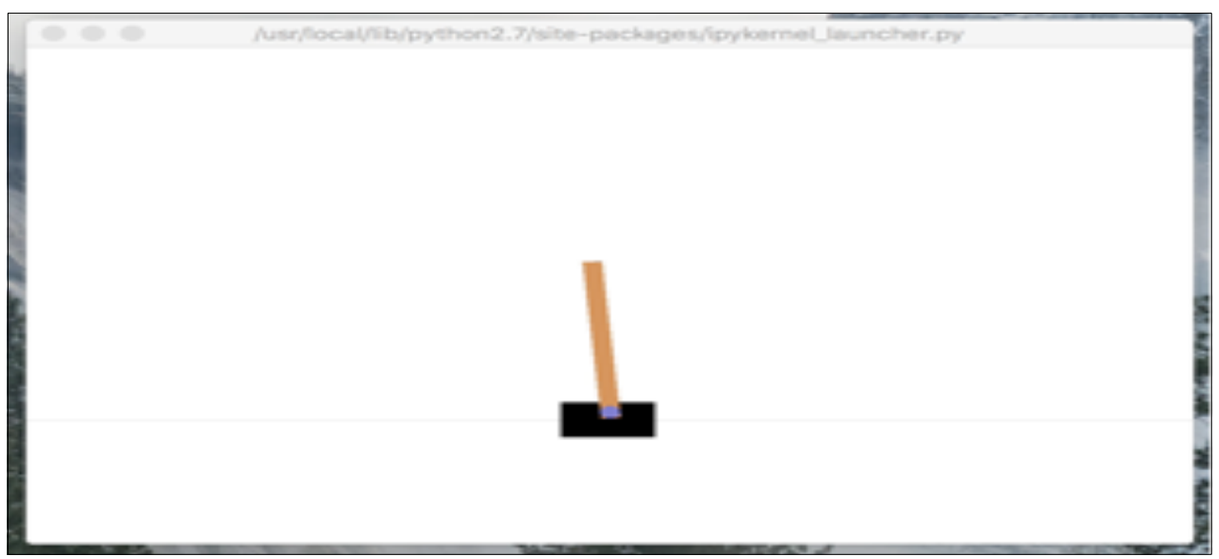
5. 学習

```
for i in range(1, 200 + 1):
    obs = env.reset()
    reward = 0
    done = False
    R = 0
    t = 0
    while not done and t < 200:
        env.render()
        action = agent.act_and_train(obs.astype(np.float32), reward)
        obs, reward, done, _ = env.step(action)
        R += reward
        t += 1
    agent.stop_episode_and_train(obs, reward, done)

# 保存
agent.save(“filename”)
```

ChainerRLの使い方 ~CartPole~

6. テスト

```
for i in range(10):
    obs = env.reset()
    done = False
    R = 0
    t = 0
    while not done and t < 200:
        env.render()
        action = agent.act(obs.astype(np.float32))
        obs, r, done, _ = env.step(action)
        R += r
        t += 1
    print('test episode:', i, 'R:', R)
    agent.stop_episode()
```



学習前

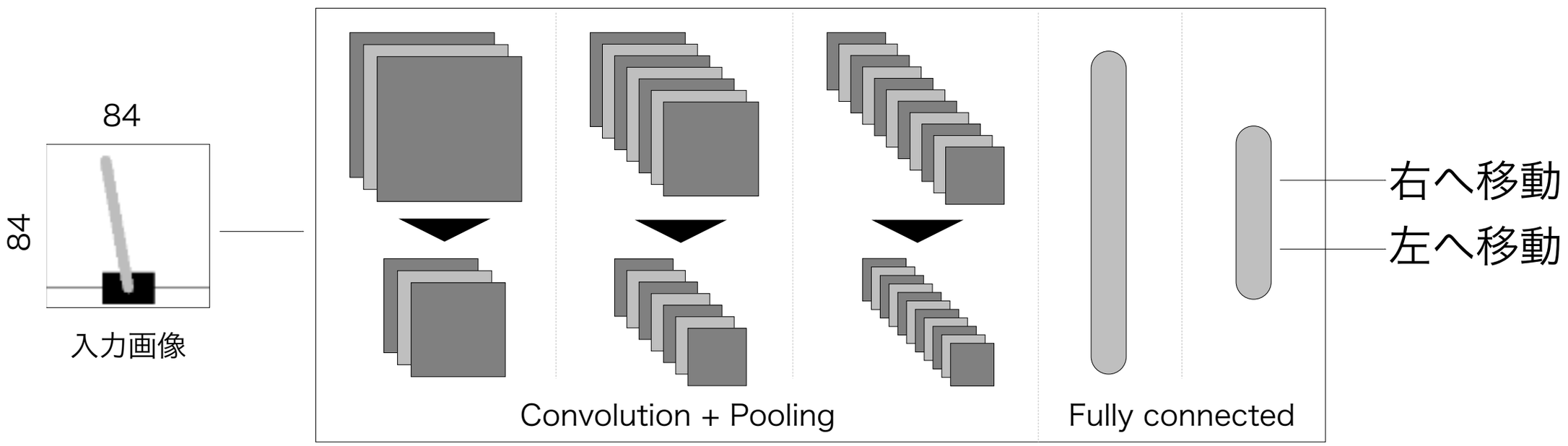
200episode学習後

深層強化学習のコツ

画像を入力としたCartPole制御

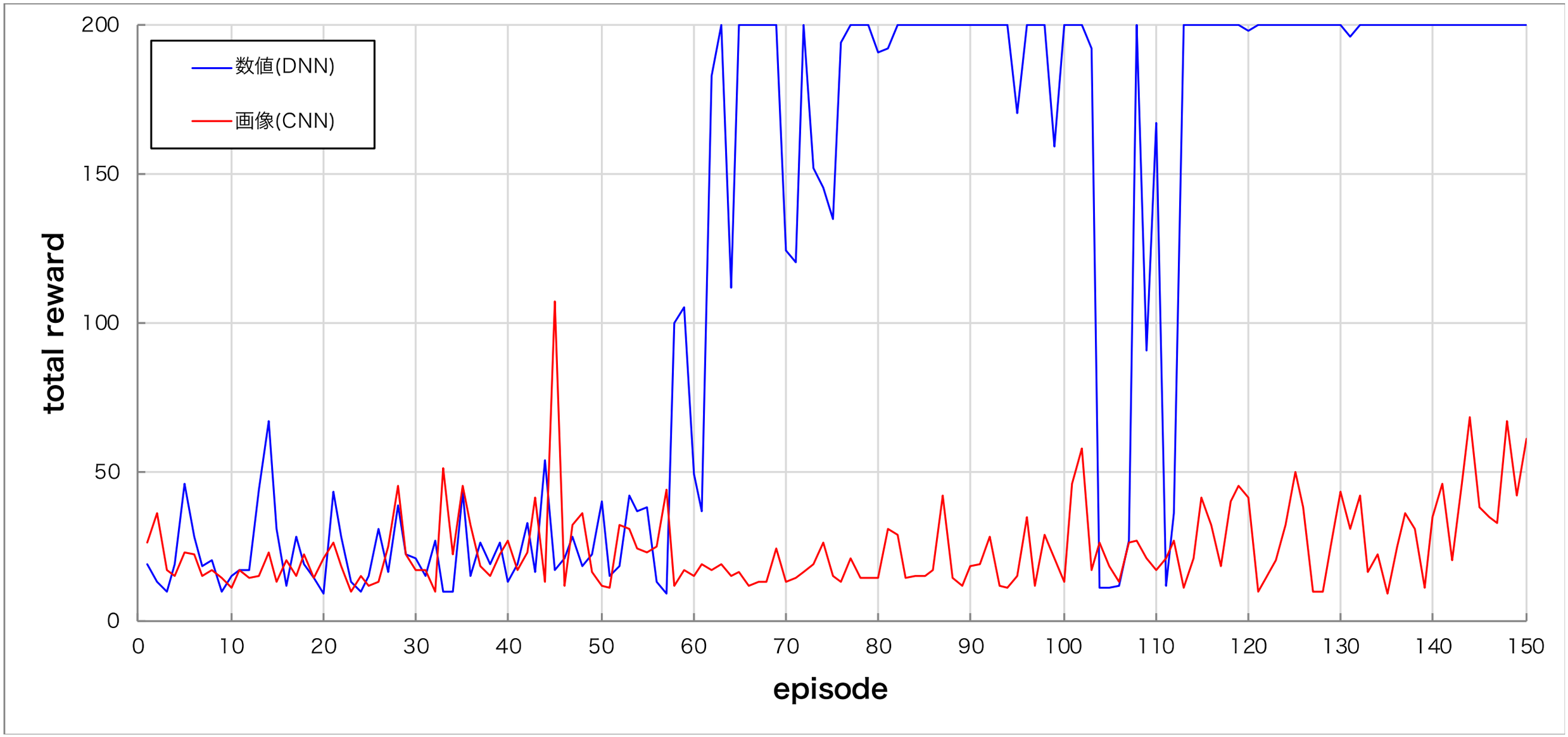
- 前の実験では数値情報を入力
  - カートの位置
  - カートの速度
  - 棒の速度
  - 棒の角速度

- グレースケール画像をCNNに入力
  - 入力画像サイズ: 84x84 [pixels]



画像を入力としたCartPole制御 ~結果~

- 報酬が向上しない
  - 学習が進まない



深層強化学習のメリット・デメリット

- メリット
  - 高次元な状態空間を扱うことが可能
  - 複雑な行動を獲得可能

- デメリット
  - 収束しない
  - 不安定な動作を獲得

- デメリットを軽減させるために
  - 学習時の工夫
    - Experience replay・Target network・Reward clipping
  - 手法の改良
    - Double DQN・A3C・UNREAL…

↑それでもうまくいかない

おさらい

- そもそも強化学習とは…

将来にわたって得られる報酬（収益）を

最大化するような行動を学習

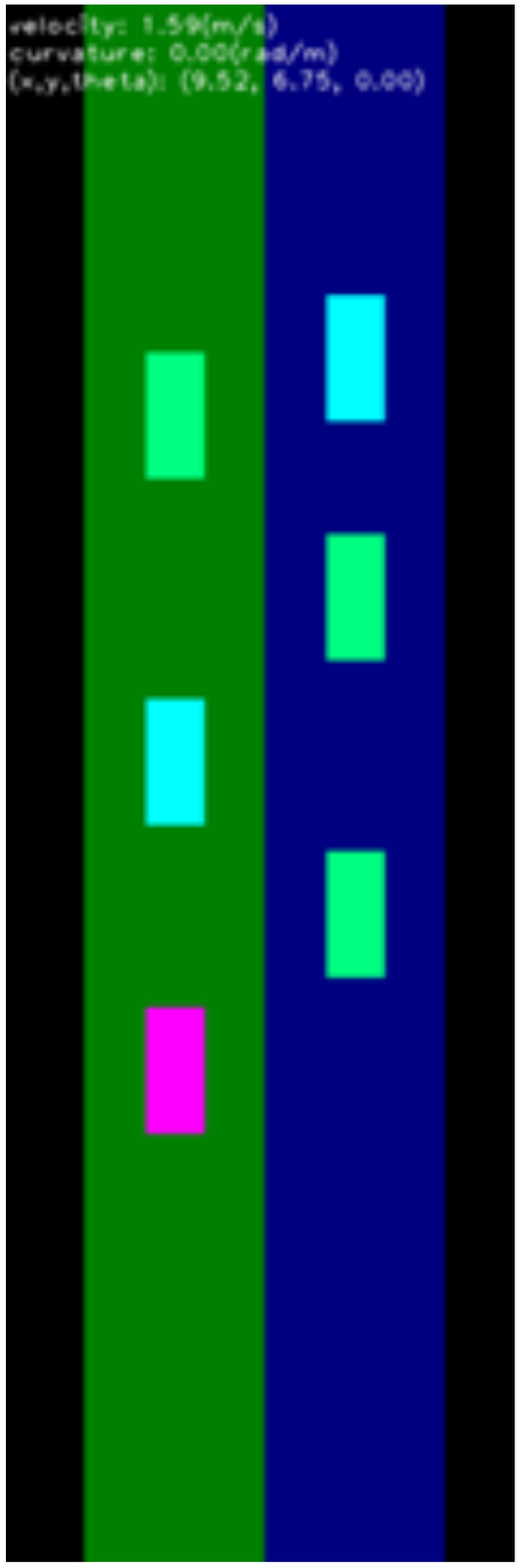
手法・アルゴリズム以上に

報酬の設計が重要！

車線変更のタスクで実験

- 2種類の報酬で実験
  - 入力: 自車の周辺領域 84x84 [pixels]
  - 出力: 9種類（速度3種類 x 曲率3種類）
  - 報酬

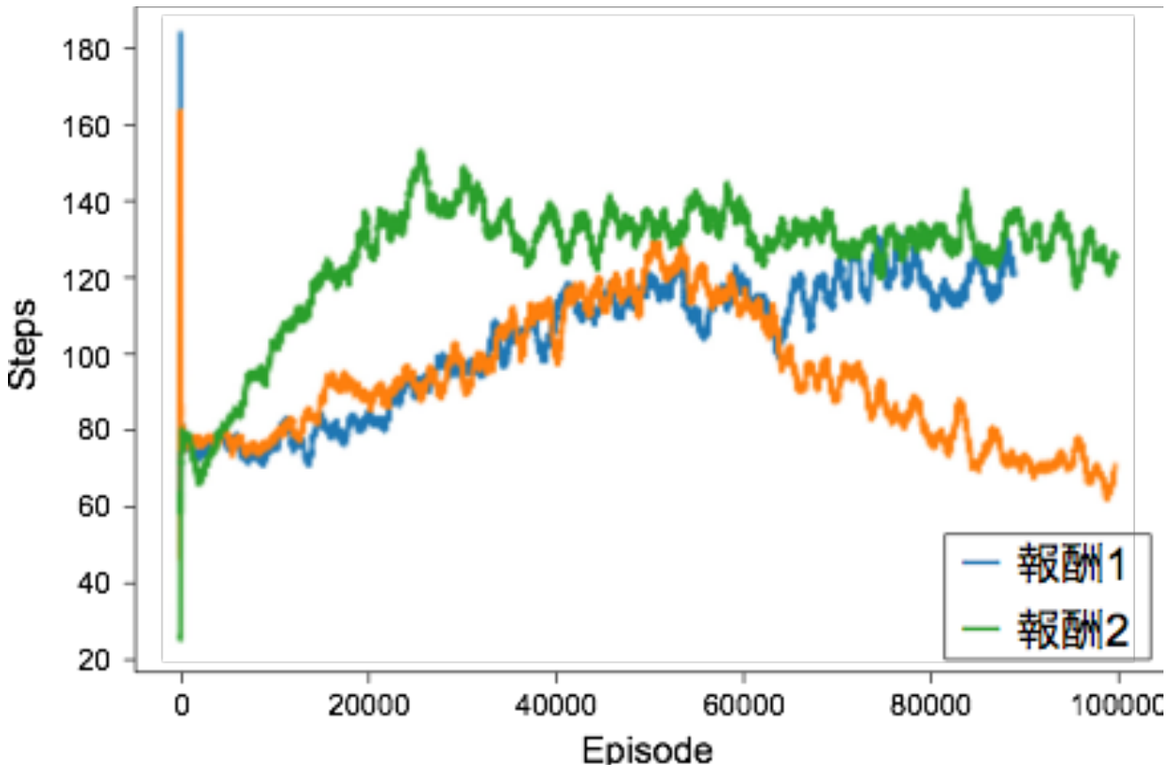
状態	報酬1	報酬2
壁または他車に衝突	-5	-100
停止	-1	0
走行中	0	+1
レーン中央	+1	+1
目標レーン	+1	+3
非目標レーン	-2	0



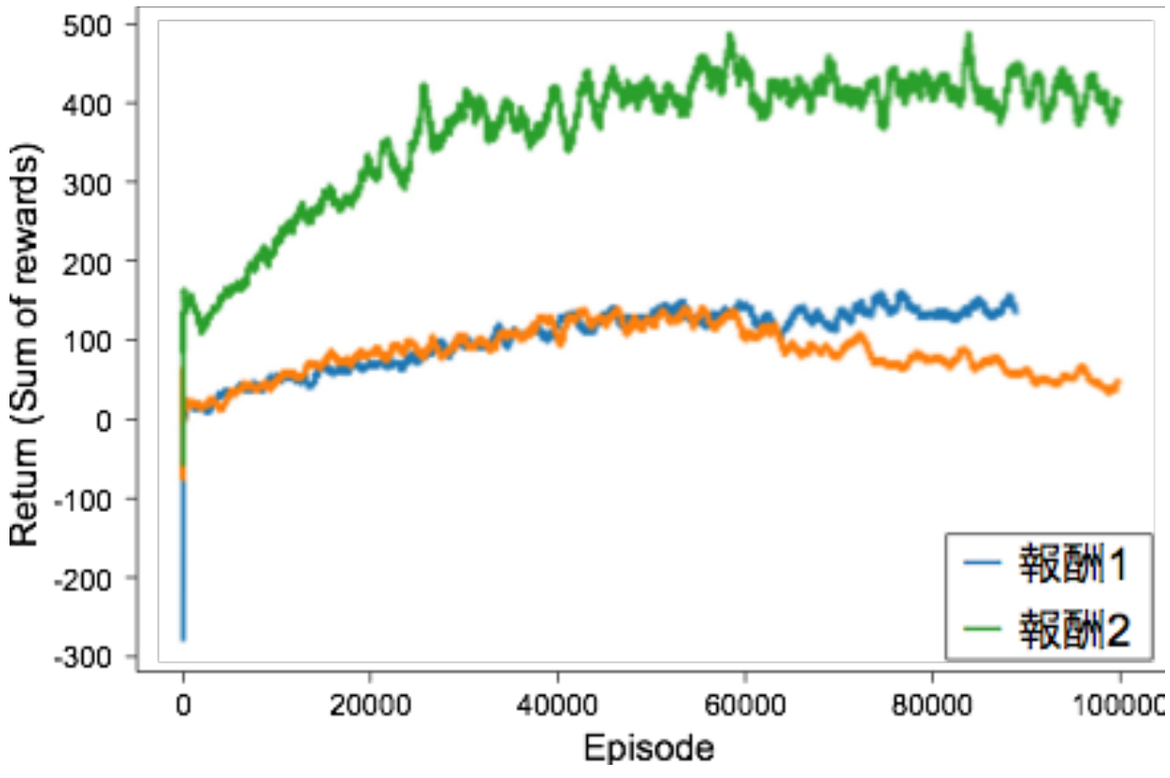
自動運転車の車線変更 ~結果~

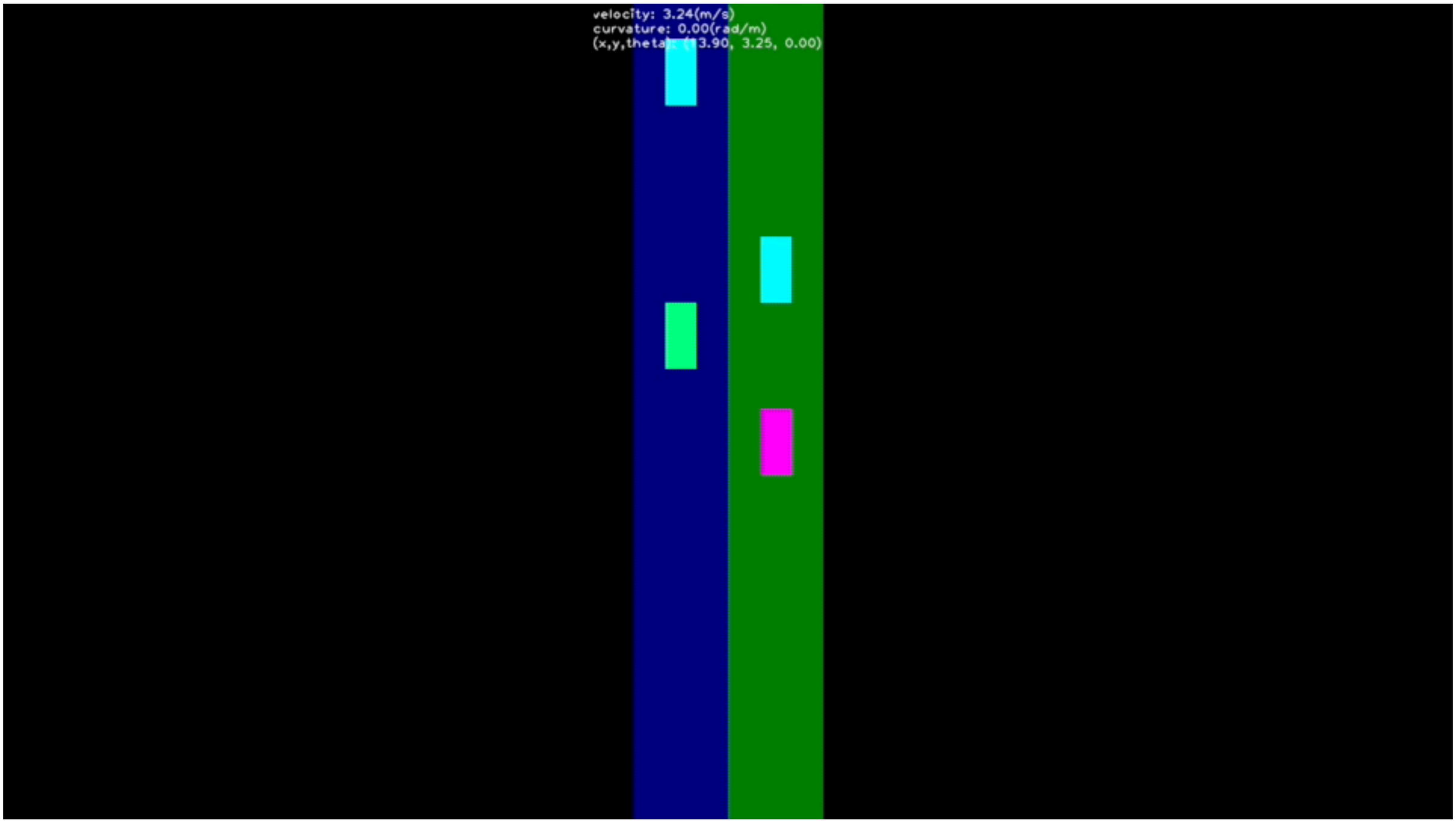
- 報酬を変更することで性能が大幅に変化
  - 報酬2: 行動回数が上昇・平均収益が高い

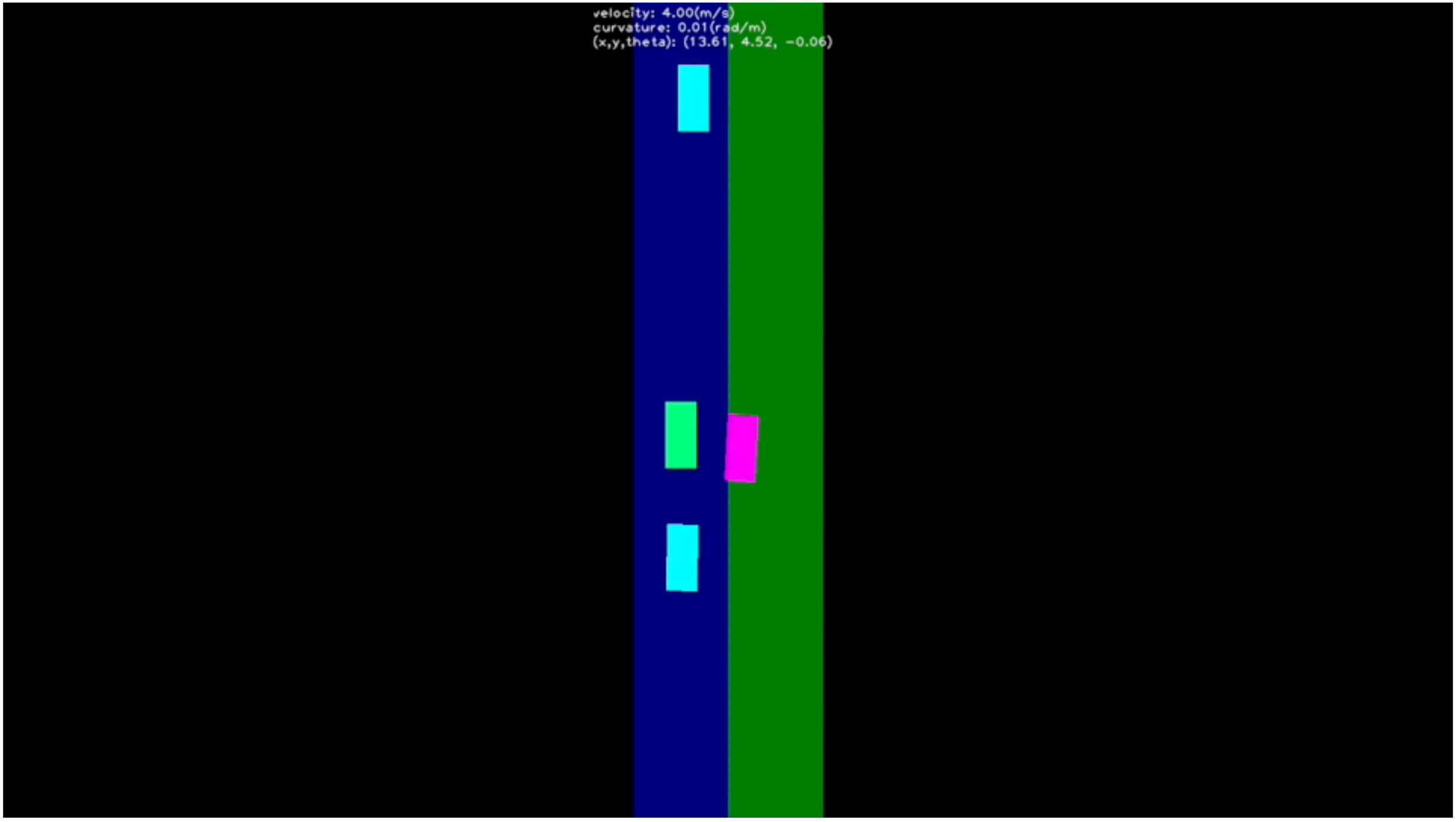
平均行動ステップ数



平均収益




velocity: 3.24(m/s)
curvature: 0.00(rad/m)
(x,y,theta): (13.90, 3.25, 0.00)


velocity: 4.00(m/s)
curvature: 0.01(rad/m)
(x,y,theta): (13.61, 4.52, -0.06)

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

深層強化学習のコツ

- 獲得したい行動ができた時 (ex. 車線変更成功)
  - かなり高めの報酬を与える (ex. +100点)

- 行って欲しくない行動をした時（ex. 衝突）
  - かなり低めの報酬を与える (ex. -100点)

- デメリット
  - Reward clippingは使えなくなります…

仮想環境を活用した強化学習

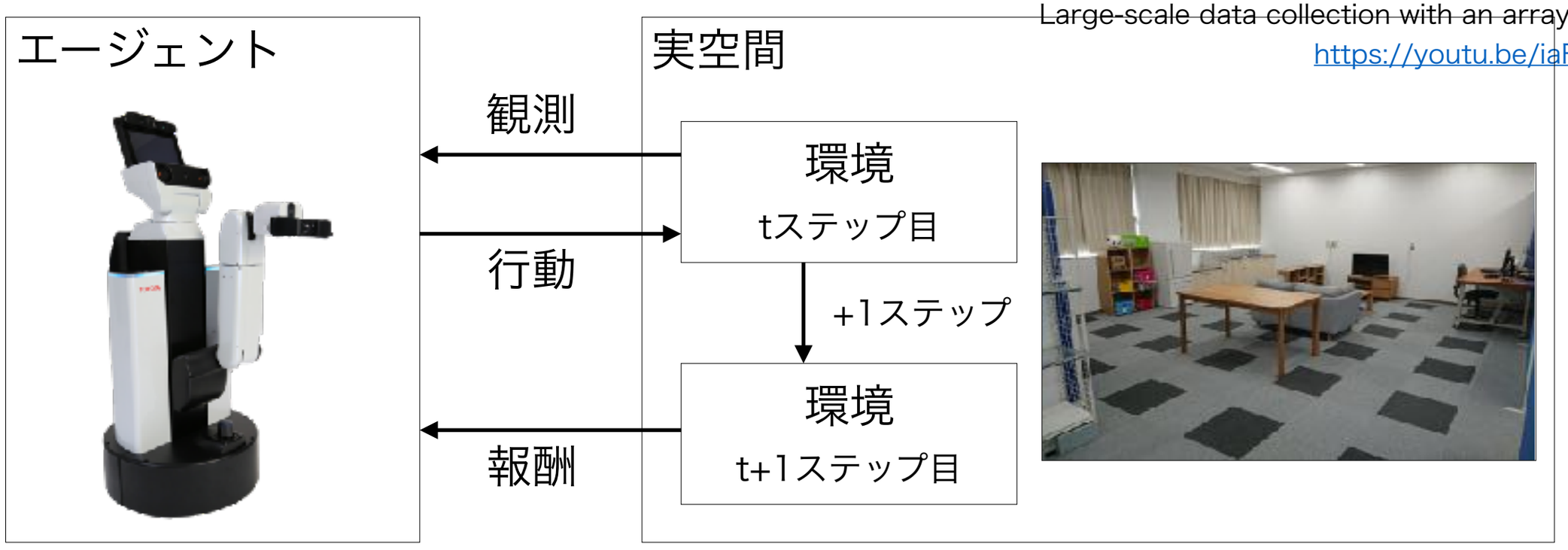
セマンティックセグメンテーションを用いた
深層強化学習による自律移動の獲得

[丸山ら，日本ロボット学会2017]

強化学習によるロボットの動作獲得

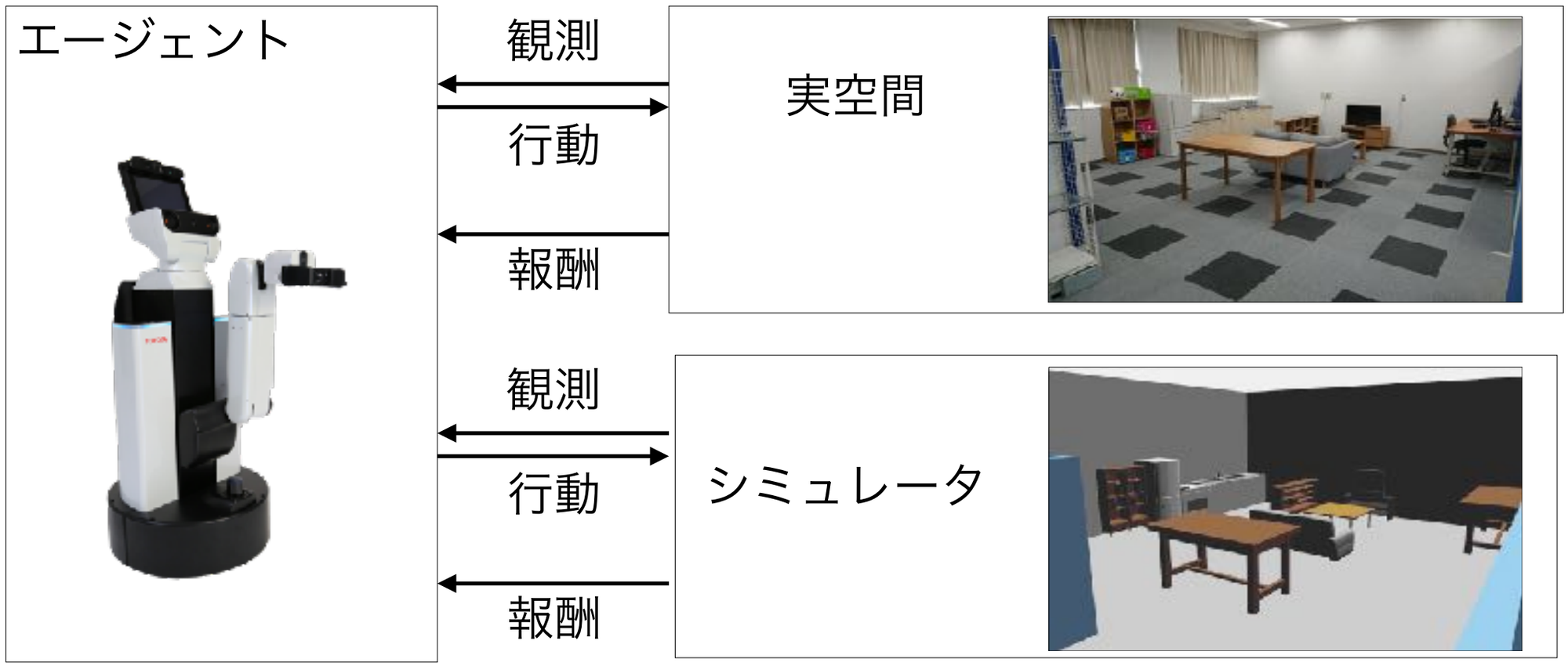
- 実空間の学習コストが大きい
  - 学習時間
  - 大量のロボット
  - 衝突への対応

Large-scale data collection with an array of robots
https://youtu.be/iaF43Ze1oel



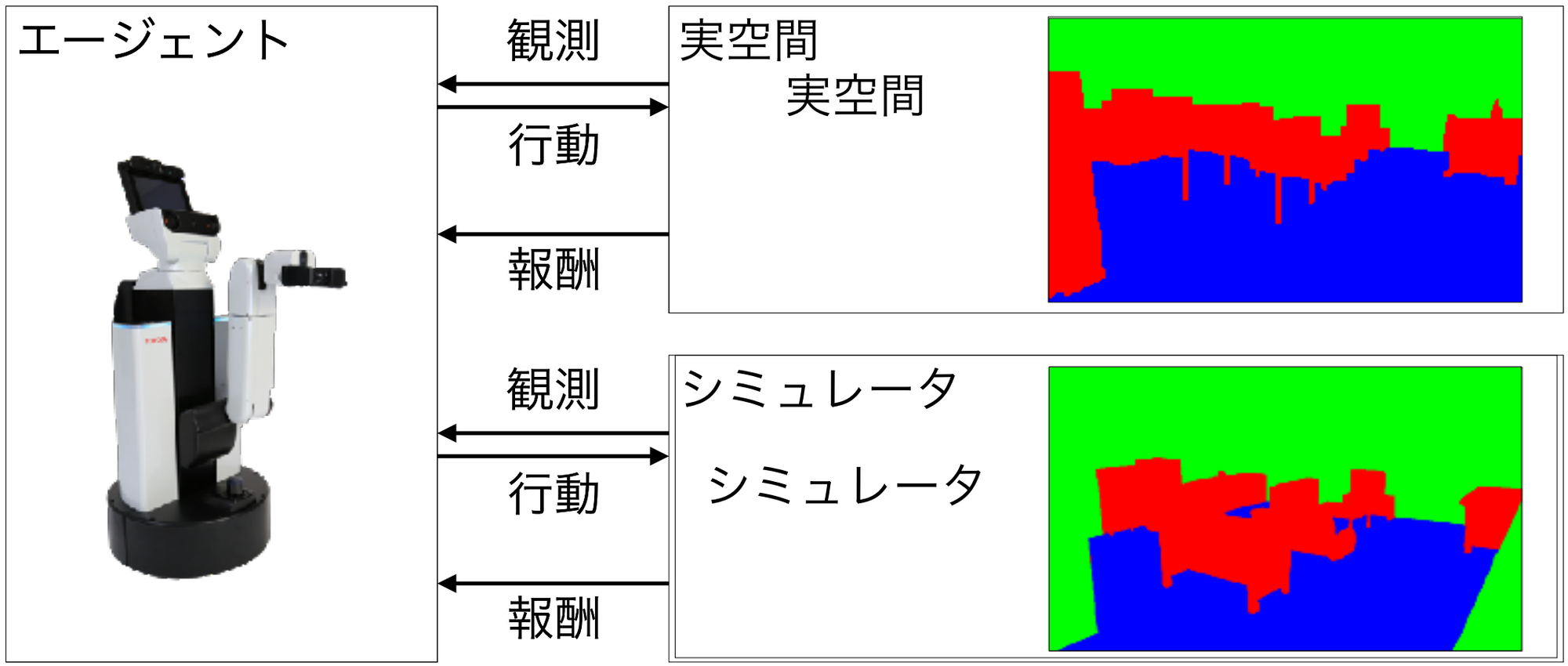
シミュレータを用いた強化学習

- 学習コストを大きく低減
  - 実空間よりも高速に動作
  - ロボットの破損等が発生しない
- 問題点
  - 実空間とシミューレータでは入力データに隔たりが存在
  - 入力データが異なると正しい行動を選択できない



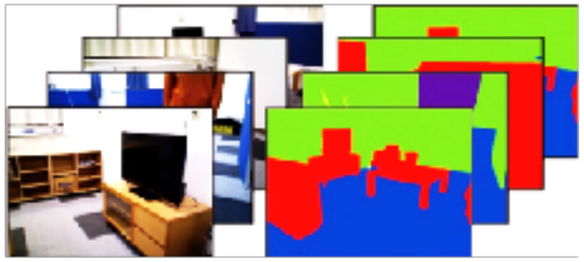
解決方法

- セマンティックセグメンテーションの活用
  - 各画素にクラスラベルを割り当てる
  - RGB画像に比べて実空間に近いデータを取得可能
    - シミュレータで学習した結果を実空間で利用可能



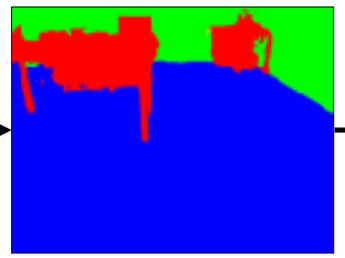
アルゴリズム

学習用画像



入力画像

セマンティック
セグメンテーション



セグメンテーション結果

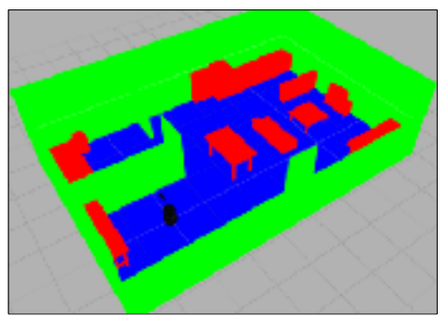
強化学習

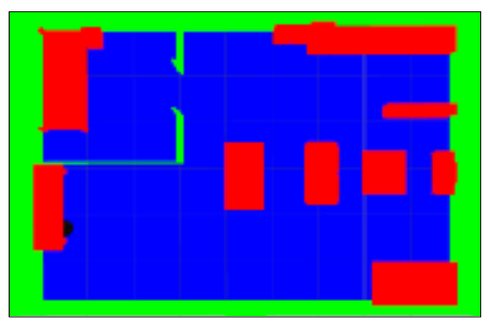
制御コマンド

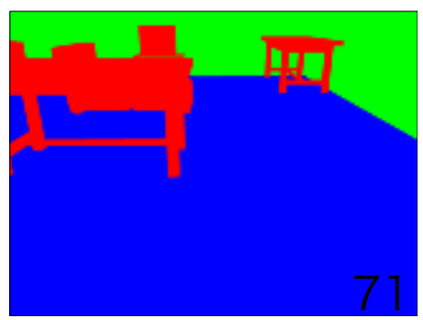
- 前進
- 右旋回
- 左旋回

ロボット

シミュレータ



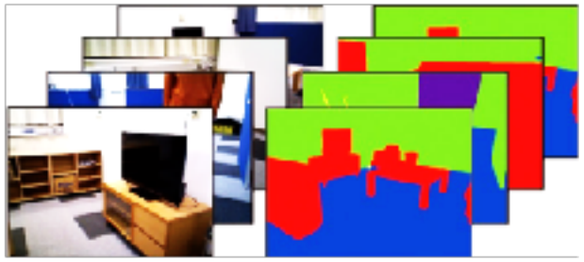




アルゴリズム

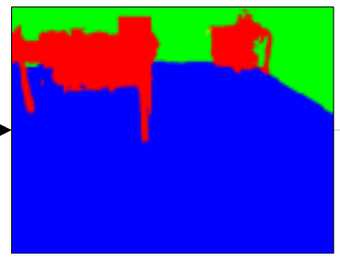
1. セマンティックセグメンテーションの学習

学習用画像



入力画像

セマンティック
セグメンテーション



セグメンテーション結果

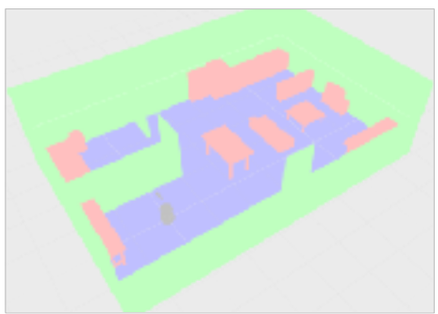
ロボット

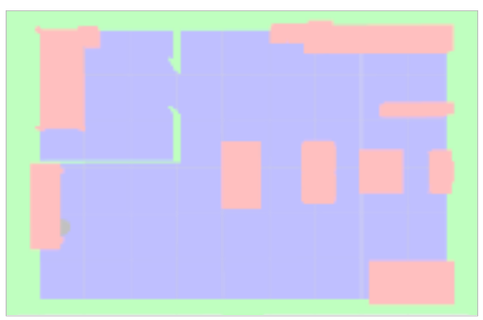
強化学習

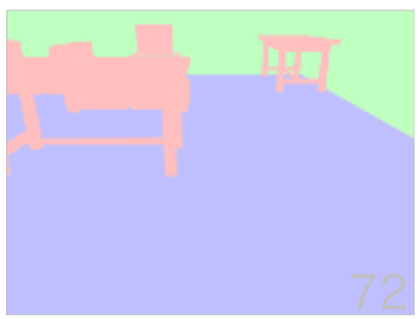
制御コマンド

- 前進
- 右旋回
- 左旋回

シミュレータ







アルゴリズム

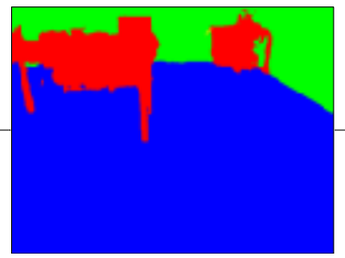
1. セマンティックセグメンテーションの学習
2. 強化学習 (Deep Q-Network)

アルゴリズム

1. セマンティックセグメンテーションの学習
2. 強化学習 (Deep Q-Network)
3. セグメンテーション結果を入力とした
   実空間での自律移動

入力画像

セマンティック
セグメンテーション



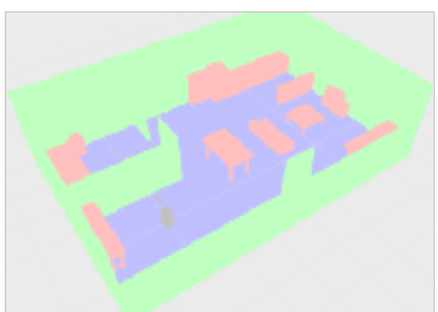
セグメンテーション結果

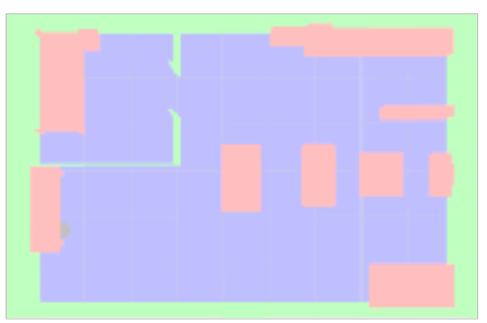
強化学習

ロボット

制御コマンド
- 前進
- 右旋回
- 左旋回

シミュレータ



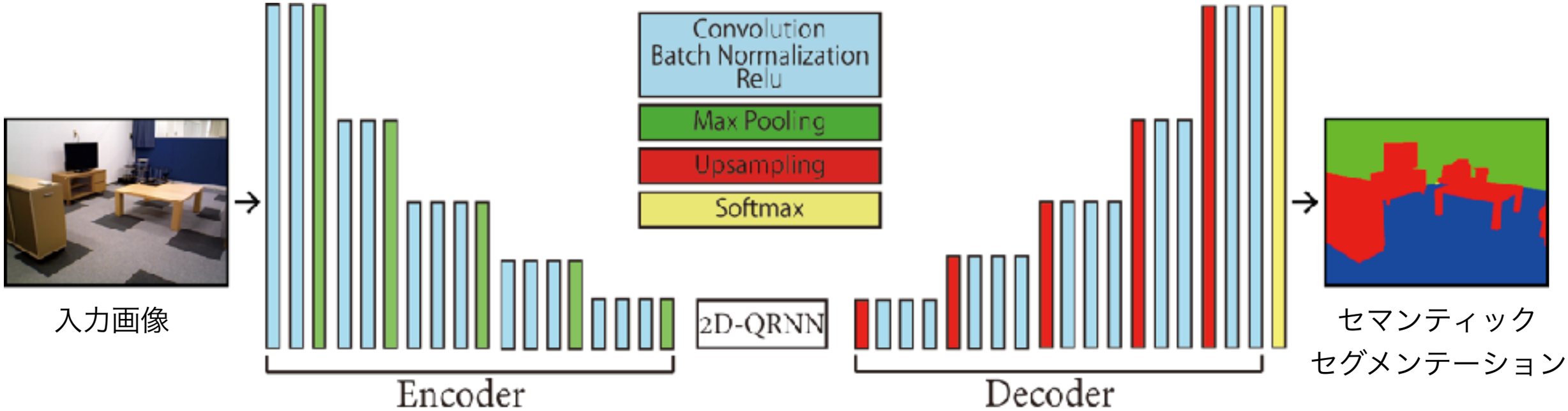


セマンティックセグメンテーションの学習

- 2D-QRNNによるセマンティックセグメンテーション

[古川ら, MIRU2017]

  - CNNと2次元拡張したQRNNを組み合わせた手法
  - 局所的 + 大域的な領域を考慮した識別が可能



セマンティックセグメンテーションの学習

- 2D-QRNNによるセマンティックセグメンテーション

[古川ら, MIRU2017]

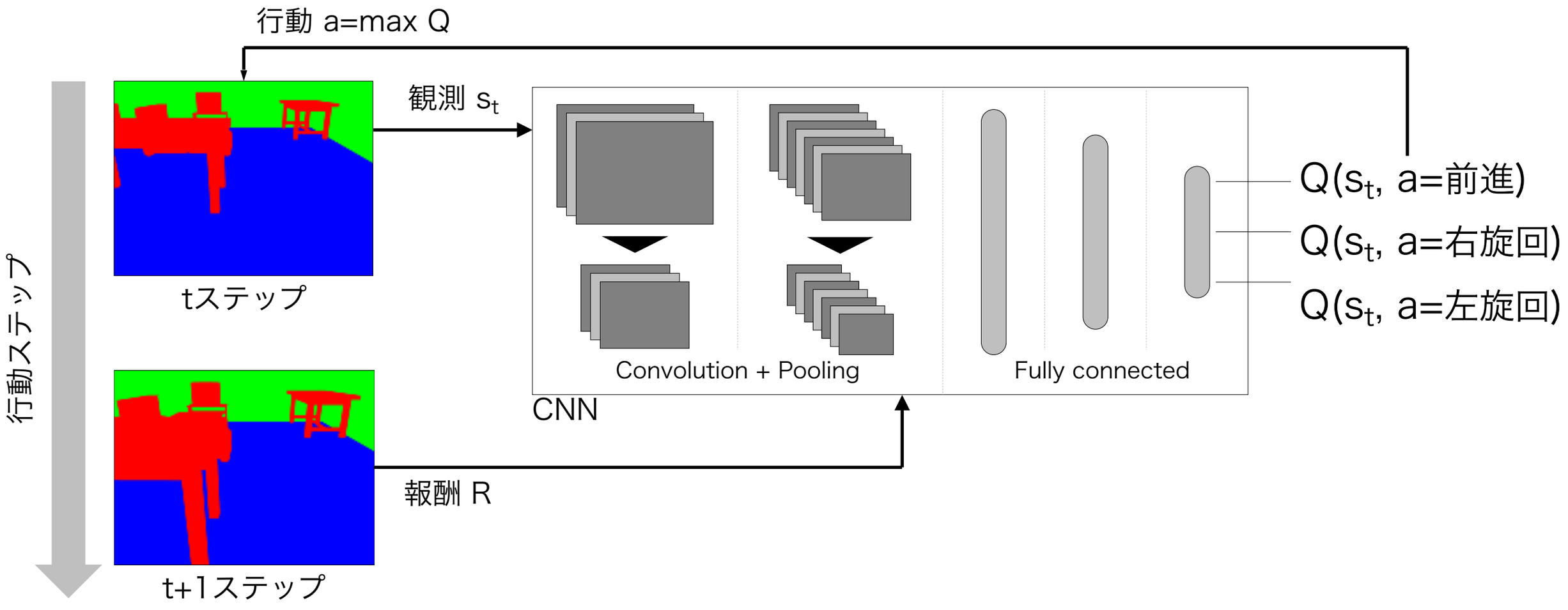
  - CNNと2次元拡張したQRNNを組み合わせた手法
  - 局所的 + 大域的な領域を考慮した識別が可能

■床　■壁　■家具

入力画像　　セグメンテーション結果

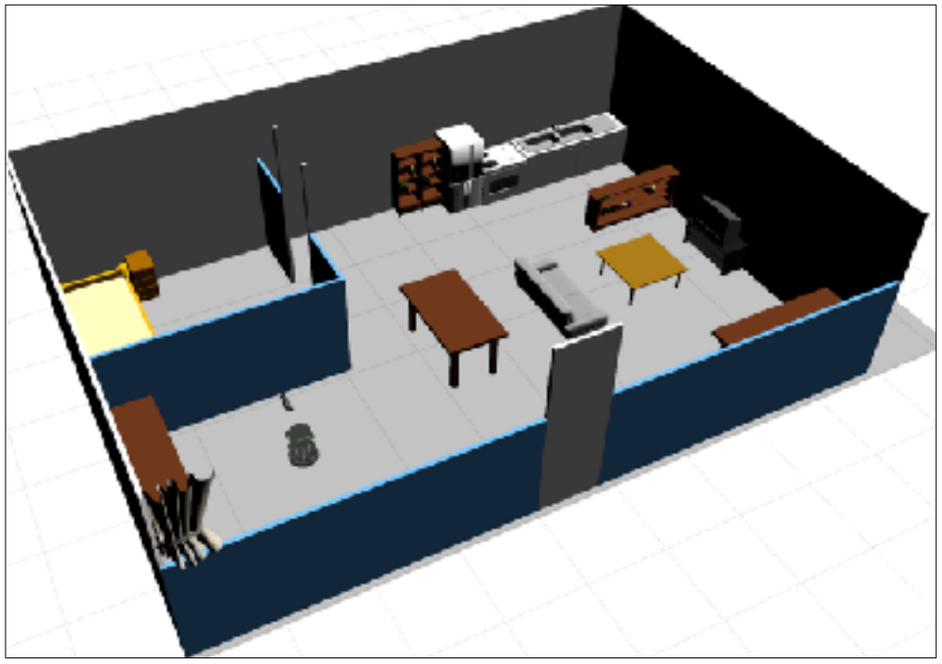
強化学習

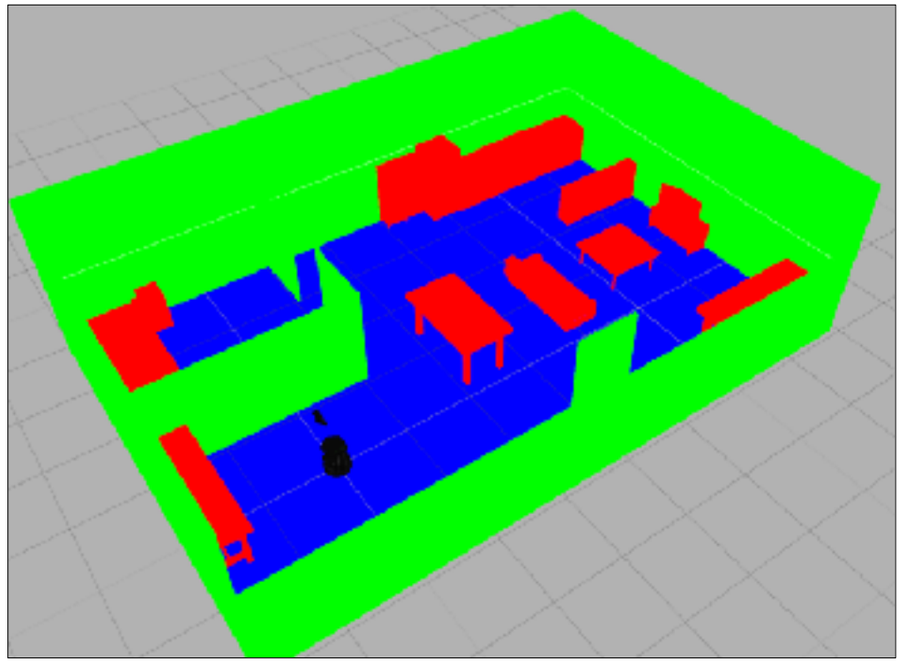
- Deep Q-Network (DQN) [Mnih, et al., NIPS2013]
  - 入力にセグメンテーション画像
    - ラベルをRGBで表現

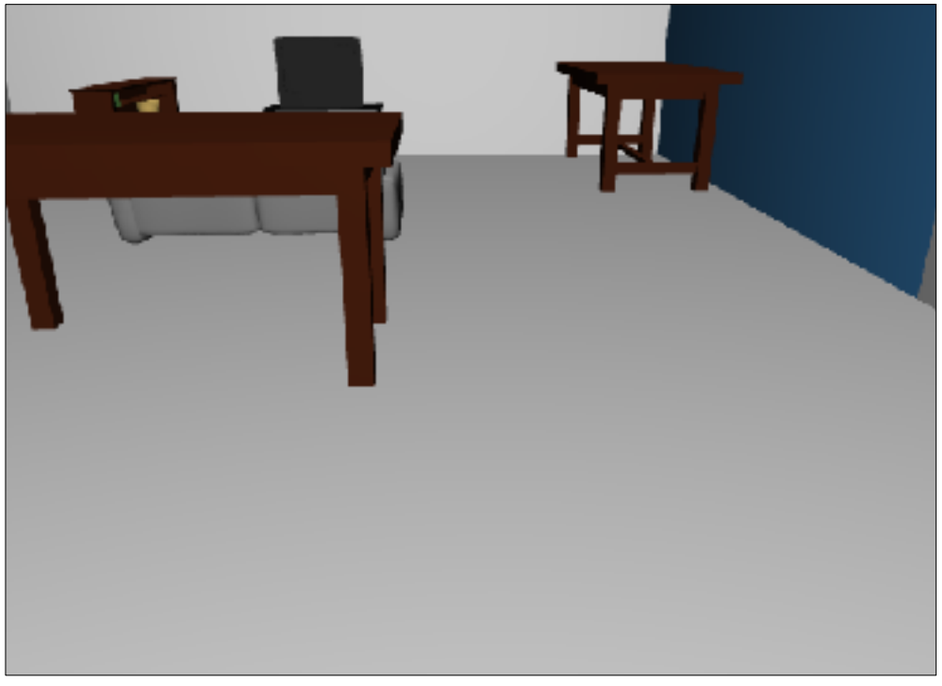


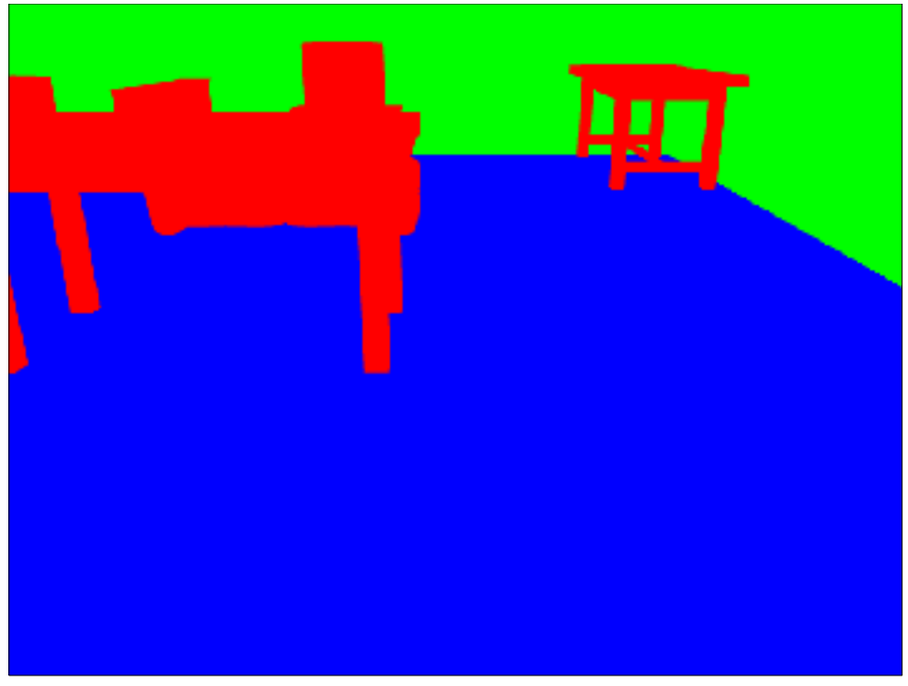
シミュレータでのクラスラベル作成

- モデルのテクスチャを単色に変更









シミュレータでの学習条件

- 報酬
  - +15 - 衝突回数　（ゴールに到達）
  - -1　　（衝突）
  - 0　　（上記以外）

- リセットの条件
  - 累積報酬が-30以下
  - ゴールに到達

- リセットの処理
  - リセットまでのステップを1エピソードとして学習
  - ランダムなスタート位置から再スタート

実験: 生活支援ロボット (HSR) のナビタスク

- 環境
  - 人が生活する空間を模した実験室

リビング

キッチン

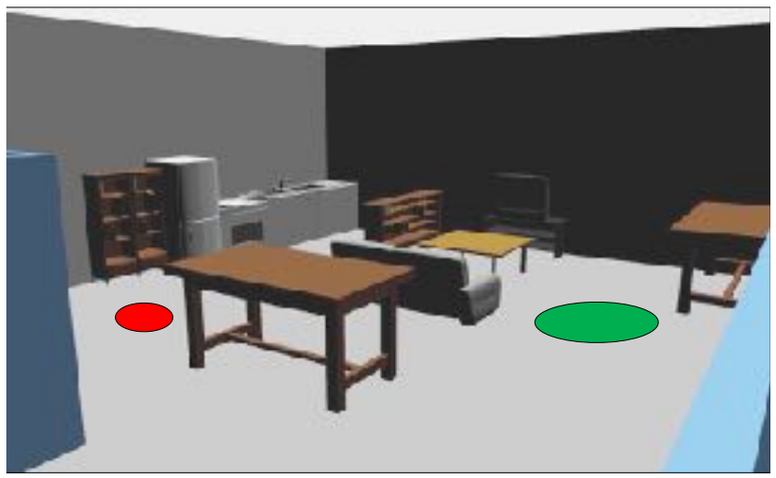
ベッドルーム

- 目標
  - 4つのスタート地点からゴールへ向かう動作の獲得

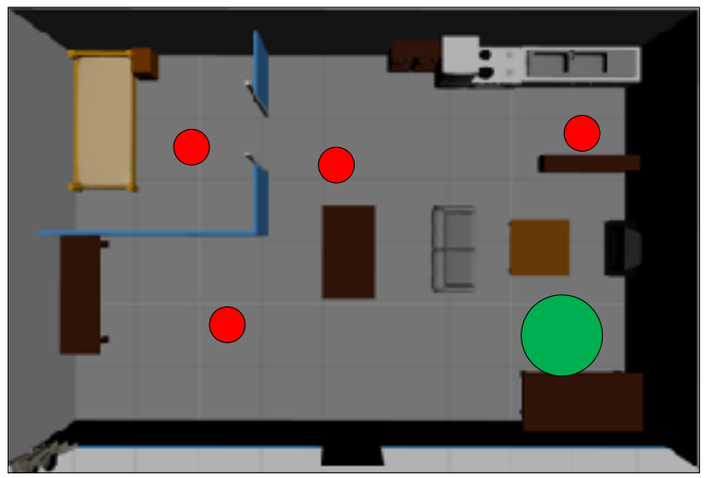
スタート

ゴール

実空間



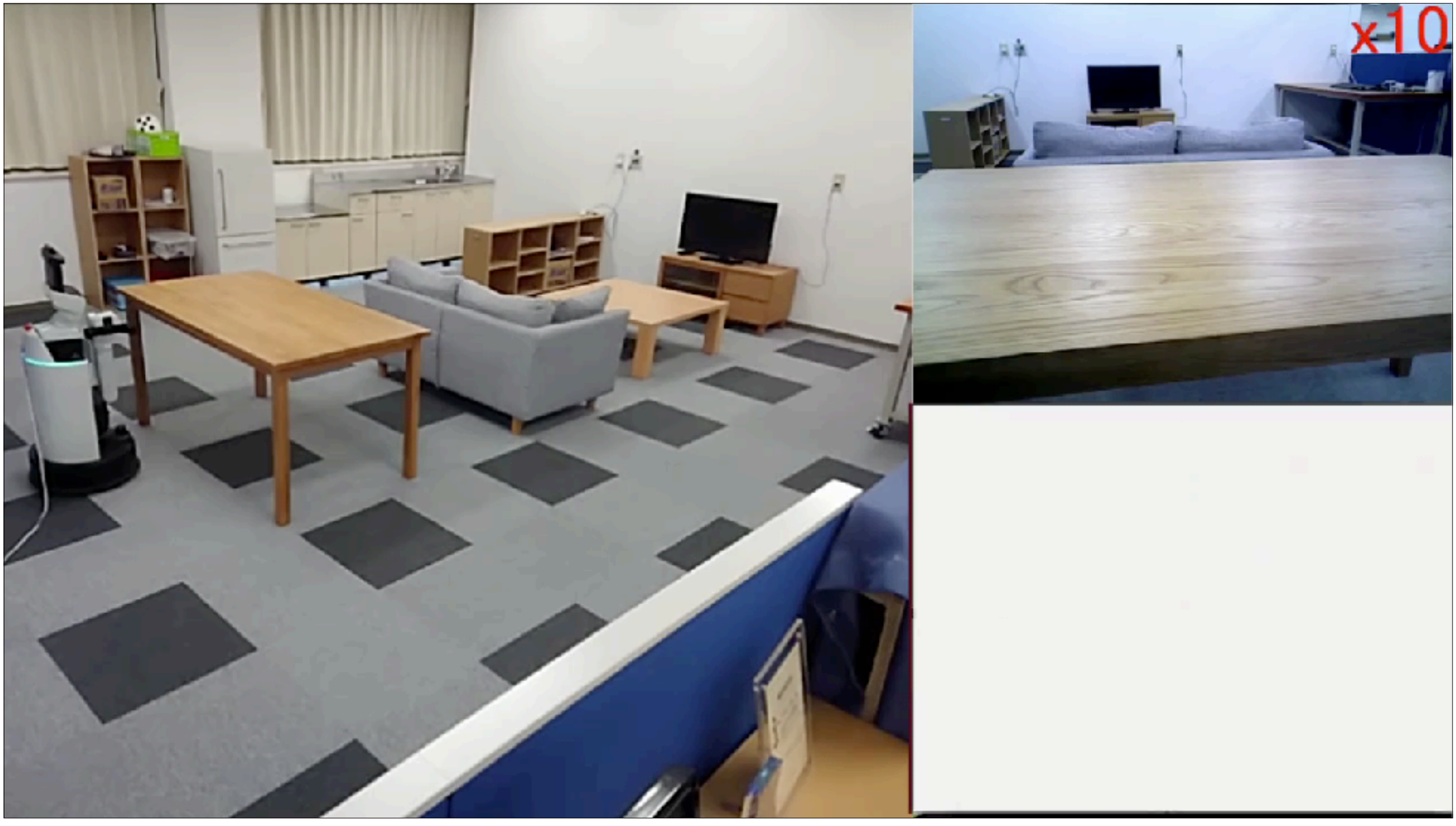
シミュレータ



鳥瞰図

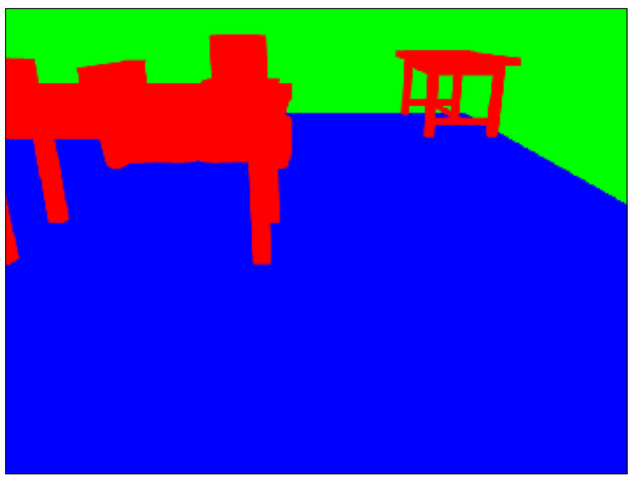
MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

実環境での動作結果

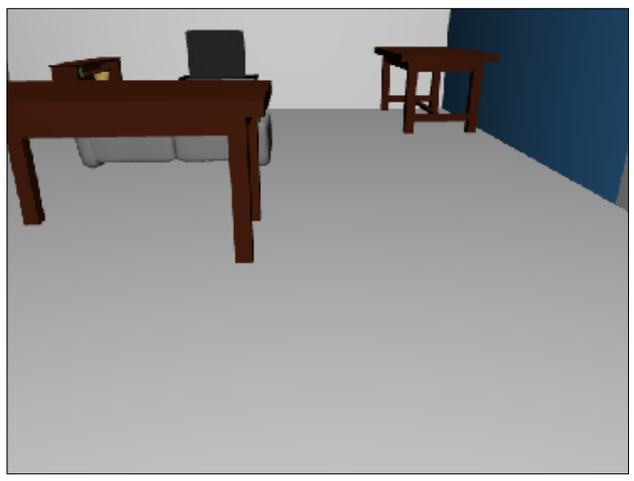


評価実験

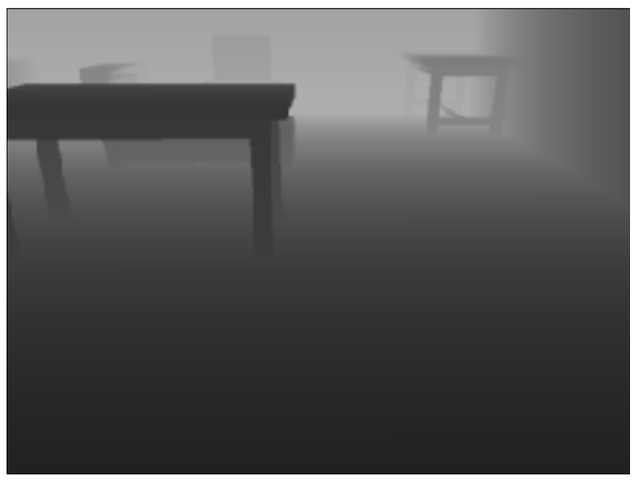
- 条件
  - 入力画像サイズ: 128×96 [pix.]
  - HSRの行動: 前進・右旋回・左旋回
- 入力画像を変更して比較
  - セグメンテーション画像（提案手法）
  - RGB画像
  - 距離画像



セグメンテーション画像
（提案手法）

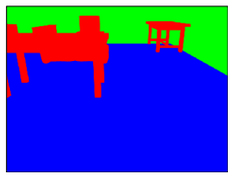


RGB画像

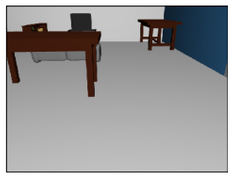


距離画像

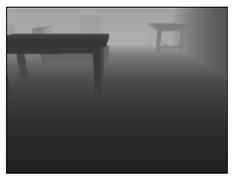
シミュレータでの学習結果



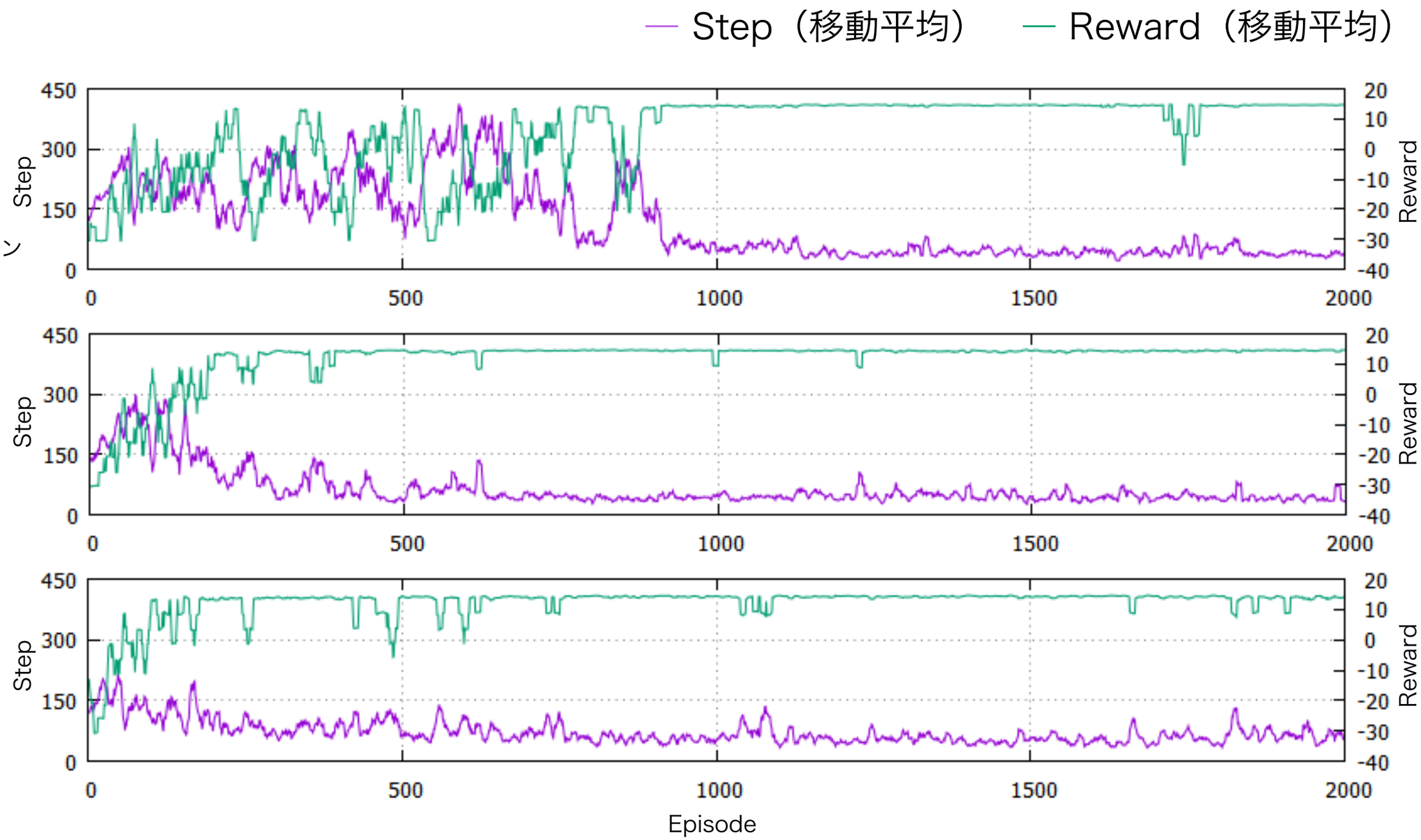
セグメンテーション画像



RGB画像



距離画像



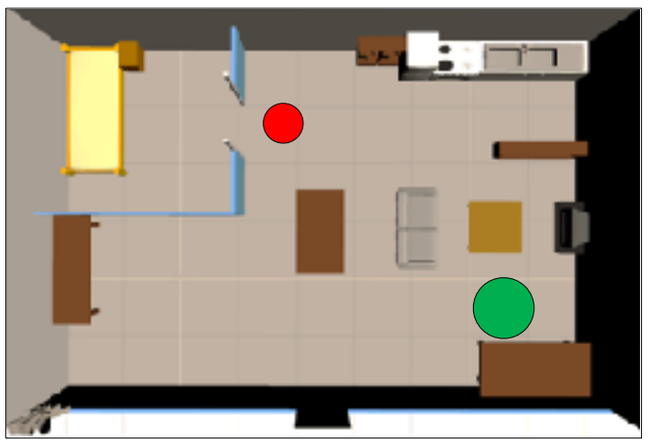
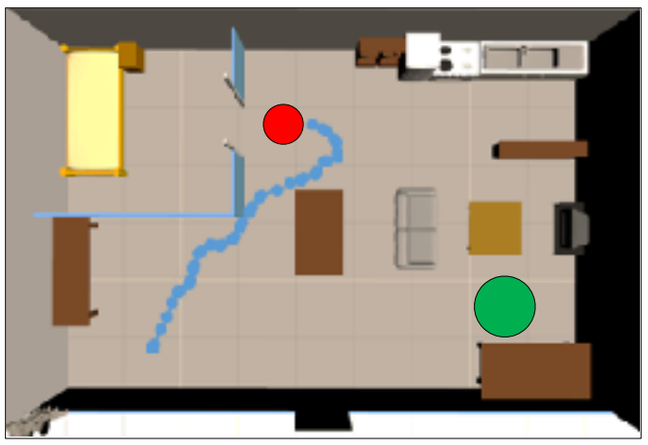
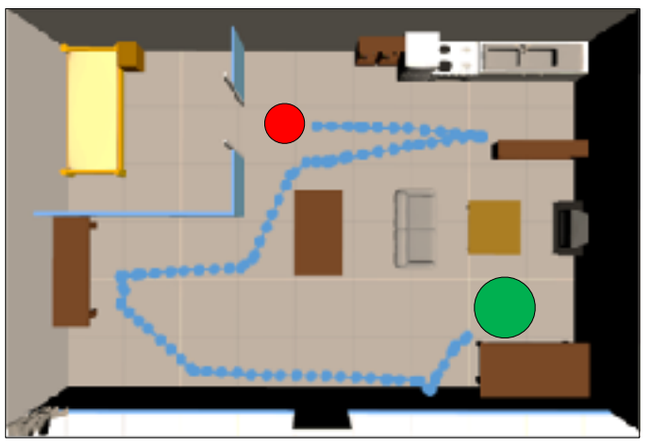
シミュレータではどの入力画像でも学習可能

実空間での評価実験

- 4箇所のランダムなスタート地点から10回実験
  - 成功条件
    - ゴールの設定地点から1m^{2}に到達
  - 失敗条件
    - 衝突または移動不可能になった場合
- 評価指標
  - 10回の実験中にゴールできた回数
  - 失敗した地点からゴールまでの平均距離

実空間での評価実験の結果

- セグメンテーション画像のみゴールへ到達
  - 他の入力画像では到達できず
  - RGB画像では移動せずその場で旋回を繰り返した

入力	RGB画像	距離画像	セグメンテーション画像
ゴール回 [回]	0/10	0/10	5/10
失敗時 平均距離値 [m]	——	4.81	2.28
移動の軌跡例			

MPRG
MACHINE PERCEPTION AND ROBOTICS GROUP

まとめ

- 深層強化学習について紹介
  - 近年の動向
  - 代表的なアルゴリズム
  - フレームワーク