Robot Intelligence Documentation

リリース *1.0*

NAGASAKA Yasunori

2020 年 09 月 23 日

Contents:

第 1 章

はじめに

ロボットインテリジェンスは学科専門教育科目の「制御・信号処理」の区分に属する科目である。本科目は「ロボットモーション」、「センサ工学」、「ロボットビジョン」に続く科目であり、センサやビジョンにより周囲の状況を認識した結果に基づき、ロボットが適切な行動を選択できるようにする情報処理の手法やアルゴリズムを学ぶ。

ロボット理工学科では、ロボットを「外界のデータを取り込み (**感覚**)、その意味を理解し (**認識**)、何をすべきかを判断し (**判断**)、結果として人に役立つように外界に働きかける (**行動**) システム」と定義している。ロボットインテリジェンスはこの中で主として「判断」を実現するための理論を学ぶ。具体的には、ロボットを構成する様々な技術要素の中の一つであり、制御・ソフトウェアによりごく単純な動作から複雑、高度な動作までロボットがどのように動くかを規定する、ロボット特有の知能を実現する技術や手法を取り上げる。その結果、知能ロボットを実現する基礎技術を理解することができる。目標は、ロボットインテリジェンスに関する様々な事柄を正しく理解して、現実にロボットを開発、製作する場面で活きた知識として活用、応用できる力を身につけることである。

ロボットインテリジェンスはまだ新しい学問領域であり、関連する技術の進歩に伴ってこれから発展していくことが予想される。よって確立された理論はまだなく、必要な場面に応じて情報処理や人工知能の手法が選択、適用されている段階である。そのような状況で、2005 年に書籍「ロボットインテリジェンス」が発行された。そこではロボットインテリジェンスに関係する刊行時までに提唱された概念、手法、開発された技術をまとめて分類、整理し、統一的に記述する取り組みがなされた。この講義では、その内容に沿って学習を進める。

人工知能に関する研究は長い歴史があり、多くの理論や技術の蓄積がある。その中の一部がロボットで使える人工知能として利用されてきた。一般に人工知能と呼ばれる理論、技術と、ロボットインテリジェンスにおける理論、技術との違いは、

- 身体性 : （例外はあるとしても）ロボット本体という物理的な存在がある点
- 走性 : ロボットは動く、移動するという点
- 相互作用 : ロボットは外界やそこにある「もの」に働きかけるという点。逆に外界やものから働きかけられることもある。

である。

ロボットにおける知能として、ロボットに知的な動作をさせるためにどのような機能が必要か、生物の知能や人工

知能と比較してロボットに特有の知能とは何かについて学ぶ。ロボットの知能が、低次の働きから順に **反応行動** 、 **計画行動** 、 **適応行動** 、 **協調行動** に分類できることを示す。何らかの行動を実現するには、周囲の環境を正しく **認識** できる必要があり、そのために画像やセンサからの情報を処理する能力が不可欠である。しかし、この講義では行動を実現する知能を主として取り上げる。また講義の進行に合わせて、ロボットの知能に関連する様々な概念を紹介する。

上記 4 種類の行動に関するそれぞれの理論を学んだあとで、各行動に関する ROS 上のシミュレータを用いた実習を行う。そこでは、指定された行動を実現する課題のプログラムを作成して動作を確認することで、理論を現実の問題に応用する力、ロボットを適切に動作させるプログラムを作成する力を身に付ける。最後の協調行動の課題では、それまでに学んできた様々な行動を組み合わせて実現する複合的な課題をプログラムとして作成する。

この後のロボットインテリジェンスの講義では主要な内容として、

- ロボットの知能とは
- 反応行動
- 計画行動
- 適応行動
- 協調行動
- その他

を取り上げる。

参考文献： 「ロボットインテリジェンス」(岩波講座 ロボット学〈4〉), 浅田稔, 國吉康夫, 岩波書店, ISBN-13: 978-4000112444, 2005 年 10 月

1.1 まとめ

この章の目標

- ロボットがどのようなシステムと定義されているかわかる、説明できる。
- 一般的な人工知能とロボットの知能は何が異なるかわかる。
- ロボットの行動が低次元のものから高次元のものに段階的に分類、表現できることがわかる。

練習問題

1. ロボットがどのようなシステムと定義されているか、簡単に説明しなさい。
2. 一般的な人工知能とロボットの知能は何が異なるか、簡単に説明しなさい。
3. 反応行動とはどのような行動か、想像して簡単に説明しなさい。
4. 計画行動とはどのような行動か、想像して簡単に説明しなさい。

5. 適応行動とはどのような行動か、想像して簡単に説明しなさい。

6. 協調行動とはどのような行動か、想像して簡単に説明しなさい。

第 2 章

ロボットの知能は行動を規定する

2.1 知能が実現する様々な行動

ロボットにおける知能として、ロボットに知的な動作をさせるためにどのような機能が必要か、生物の知能や人工知能と比較してロボットに特有の知能とは何かについて考えてみる。この後の章で順次取り上げるが、ロボットの知能が実現する行動として、低次の働きから順に **反応行動** 、**計画行動** 、 **適応行動** 、 **協調行動** に分類できることを示す。

反応行動 とは、自分 (ロボット) の周囲のごく狭い範囲の環境から入力される外部の情報（様々なセンサからの情報）に対して引き起こされる単純な行動である。その行動は一定で変化することはない。このレベルでは、自分と自分の周囲のごく狭い範囲の環境（からの入力）しか存在しない。反応行動では、走性、反射、感覚運動写像、行動単位の重ね合わせ、人工ポテンシャル法、運動のスキーマ、行動の分解およびモジュール化、有限オートマトンによるモデル化などの概念が関係する。

計画行動 とはロボットを取り巻く環境内で自分（ロボット）の位置、および達成目標に関係する物体の位置を把握して、目標達成のために計画する移動や運動を表す。このレベルでは、自分と自分の周囲の広い範囲の環境（からの入力）が存在する。環境は静的で変化しない。計画行動では、空間記憶と認知地図、座標系について学ぶ。 次に応用として、自己位置同定とそれを実現する手法、空間表現、コンフィグレーション空間、状態空間モデル などの概念が関係する。

適応行動 とは環境や外部からの入力（センサ情報）が変化する場合に、それぞれの環境や入力に対して自分の行動を変化させ、望ましい行動を実現することである。このレベルでは、自分と自分の周囲の環境（からの入力）が存在する。環境は動的で変化する。適応行動を実現するには環境の変化に対応して自分の行動を変化させる何らかの学習の仕組みが必要となる。その手法として、強化学習、人工神経回路網、ニューラルネットワークなどが広く利用されている。

協調行動 とは複数台のロボットが協調、協力して全体の目標を達成するような行動を表す。単独の個体で取り組むよりも複数の個体が協調、または役割分担を決めて取り組んだ方が目標を達成しやすい場合に協調行動が必要となる。このレベルでは、自分と自分の周囲の環境（からの入力)、自分以外の個体が存在する。協調行動に関係する概念として、コミュニケーション、観察による協調、群れ行動、サブゴールの生成、マルチエージェント、共進化 などがある。

2.2 未来のロボットが家にいると

ここでは高度な知能を備えた未来のロボットを想定して、それがどのように与えられた課題を達成するか想像してみる。課題を達成する過程で様々な行動が現れるが、場面毎にどのような行動が必要とされるか書き出してみる。後でそれらの行動が、前述した 4 種類の行動のどれに相当するか分類する。

前提として、ロボットは家に 2 台いていろいろなことを手伝ってくれる。名前を RX, RY とする。

なお、単純な反応行動を除いて高次の知能による行動は、周囲の状況の高度な認識機能を必要とするので、行動とあわせてどのような認識機能が必要とされるかも記述する。

2.2.1 シナリオ 1

課題： お父さんに道具を取ってきて渡す。

前提は、過去に倉庫から道具を取ってきたことがある、とする。状況は、庭でお父さんの近くにいる、とする。

ある日曜日、お父さんが庭に出た時、先日植えた木が支柱と一緒に倒れそうになっているのを見つけた。そこで支柱を打ち込み直そうと思って近くにいた RX に声をかけた。「おーい、RX、支柱を直すので何か叩くものを持ってきてくれ」RX はお父さんと斜めになった支柱を見て返事をした。「はい、わかりました」

- 注視反応、声のする方に注意を向ける、注目する
- 音声認識
- 自然言語理解
- 視覚認識
- 情景理解、お父さんが支柱を叩こうとしていることがわかる

推論

RX は庭の隅にある倉庫に行くことを考えた。以前お父さんに教えてもらって倉庫に道具を取りに行ったことがあって、そこに役に立ちそうな道具が置いてあると推論した。

- 必要とされる道具の機能の理解、推論
- 道具がある場所の推論
- 過去の経験の記憶、記憶からの推論
- 行動計画、まず倉庫に行き、そこで道具を探す
- 経路計画、現在地から倉庫までどのように行くか

庭を倉庫に向かって移動していく。途中、植えてある木や花を踏まないように避けて移動する。以前倉庫に行った時の経路を思い出して参考にする。石に引っかかってしまってバランスを崩しそうになり、転倒しないように急い

で姿勢を立て直す。

- ナビゲーション
- 地図生成
- 経路計画、倉庫まで
- 障害物回避
- 学習
- 反射

倉庫の前に着いたが運悪く扉の前に自転車が置いてある。自分だけでは移動させられない。周囲を見渡すと RY が近くで別の作業をしている。自転車をどかすのを手伝ってもらおうと、RY にこちらに来るように依頼した。自転車の前の方を RX が持ち、後ろの方を RY が持ち、タイミングをあわせて移動させた。

- 行動計画、扉を開けるには自転車を移動させる必要がある
- コミュニケーション
- 協調行動、２台で協力して自転車を移動させる
- 行為理解、RX が前の方を持っているので、RY は後ろを持つのが適切だと推論した。

倉庫の扉を開け、中を見渡す。多くの道具があったがどの道具が適切か、見ただけではわからなかったので、過去の経験から考える。以前お父さんに金槌を持っていったことがあり、釘を叩いていたことを思い出した。今回も金槌を持っていくことにした。金槌を持ったら扉がうまく閉められなかったのでそのまま行くことにした。開け放しの扉を見つけたら RY が閉めてくれると推論した。

- 物体認識
- 物体の機能理解
- 過去の経験の記憶、記憶からの推論
- 行動予測
- 協調行動、相手の行動を推論する

お父さんのところに戻り、金槌を渡す。お父さんは、「遅かったな」とちょっと機嫌が悪そう。「自転車をどかすのに時間がかかりました」と説明した。「金槌か、まあいいだろう」と受け取って、支柱を真っ直ぐに直して打ち込み始めた。

- 感情認識
- 行為理解、自己の行為の説明
- 自然言語による対話

お父さんはしばらく打ち込む作業をしていたが、「疲れたから続きはやってくれ」と、RX に金槌を渡した。金槌を使うのは初めてだったが、お父さんの作業を横で見ていたので、どのように使うかは理解した。何回か支柱の頭を叩くうちに上手にできるようになった。

- 行為理解、お父さんの作業を理解する
- 模倣学習、お父さんの真似をする
- 運動学習

2.2.2 シナリオ 2

課題： 買い物袋を運ぶ。

状況は、家の中にいる、とする。

買い物から帰ってきたお母さんに玄関から呼ばれた。「重いから持っていって」と袋を渡された。中にはお米と卵と、牛乳が数パック入っていた。どこに持っていくか考えたら、飲食物なので台所が適当だろうと推論した。卵は割れやすいのでやさしく運んだ。台所に着いてどこに置くか考えた。牛乳と卵はお母さんが冷蔵庫から出し入れしているのを以前に見て覚えていたので、冷蔵庫にしまった。米は置く場所がわからなかったので冷蔵庫の横に置いた。

- 注視反応、声のする方に注意を向ける、注目する
- 音声認識
- 自然言語理解
- 物体認識
- 物体の機能理解、飲食物、脆さ
- 行為理解、お母さんの意図を推論する
- 過去の経験の記憶、記憶からの推論
- 経路計画、台所まで
- ナビゲーション

2.2.3 シナリオ３

課題：　新聞を取ってきて渡す。

状況は、家の中にいる、とする。

日曜日の朝、リビングにいるお父さんに呼ばれた。「新聞取ってきて」と頼まれた。お父さんが朝新聞を読んでいるのを何回か見たことがあるのを思い出した。お父さんは新しい新聞を読みたいのだろうと推論して、玄関脇の郵便受けから新聞を取ってきてお父さんに渡した。

- 注視反応、声のする方に注意を向ける、注目する
- 音声認識
- 自然言語理解
- 地図生成、家の中、リビング、玄関
- 物体認識、郵便受けにある新聞
- 行為理解、相手の行動を推論する
- 過去の経験の記憶、記憶からの推論
- 経路計画、玄関まで
- ナビゲーション

月曜日の昼間、お風呂場で掃除をしているお母さんに呼ばれた。「新聞取ってきて」と頼まれた。掃除の途中だから、お母さんは新聞を読みたいのではないだろう。以前、風呂場で拾った髪の毛やゴミを新聞に包んで捨てる時に呼ばれたのを思い出して今回も同じことをしたいのだろうと推論した。読み終わった新聞をしまってある場所から新聞を取ってきてお母さんに渡した。

- 注視反応、声のする方に注意を向ける、注目する
- 音声認識
- 自然言語理解
- 物体認識、しまってある新聞
- 行為理解、相手の行動を推論する
- 過去の経験の記憶、記憶からの推論
- 地図生成、家の中、お風呂
- 経路計画、新聞が置いてある場所まで
- ナビゲーション

月曜日の夕方、リビングで遊んでいる小学生のお兄ちゃんと幼稚園児の弟に呼ばれた。「新聞取ってきて」と頼まれた。兄弟はそれぞれフニャフニャした棒のようなものを振り回して戦っている。お侍の真似をしてチャンバラごっこをしているのだろうと推論した。新聞を読みたいわけではなく、遊びに使いたいのだろうと推論して、読み終わった新聞をしまってある場所から新聞を取ってきて二人に渡した。

- 注視反応、声のする方に注意を向ける
- 音声認識
- 自然言語理解
- 物体認識、しまってある新聞
- 行為理解、相手の行動を推論する
- 過去の経験の記憶、記憶からの推論
- 地図生成、家の中
- 経路計画、新聞が置いてある場所まで
- ナビゲーション

2.2.4 シナリオ、練習問題 1

課題： 椅子に座る。

前提は、ヒト型である、過去に実行したことがある、とする。状況は、目の前に椅子が置いてある、とする。

- 物体認識、椅子の位置を認識する
- 経路計画、座れる位置まで移動する
- 座る際にバランスを崩した時は転倒を防ぐ

補足

- 物体の機能理解、座り方
- 過去の経験の記憶、記憶からの推論、椅子に座る
- 運動学習、倒れないように、滑らかに座る

2.2.5 シナリオ、練習問題 2

課題： 服を着る。

前提は、ヒト型である、過去に実行したことがある、とする。状況は、目の前に服（T シャツ）が置いてある、とする。

- 物体認識、服
- 地形認識
- 地図生成
- 自己位置推定
- ナビゲーション
- 移動
- 障害物回避
- 物体の形状認識、どこに頭を通し、どこに腕を通すか
- 把持、着られるように持つ
- 着る

補足

- 過去の経験の記憶、記憶からの推論、服を着る
- 運動学習、引っかからないように頭、腕を通す

2.2.6 シナリオ、練習問題 3

課題： 冷蔵庫にしまってある牛乳（パック）を出してきてコップに注ぐ.

前提は、過去に実行したことがある、とする。状況は、テーブルの上にコップが置いてある、近くに冷蔵庫がある、とする。

- 物体認識、冷蔵庫から牛乳を探す
- 位置測定
- 物体の把持、牛乳を掴み、持ち上げる
- 冷蔵庫を閉める
- コップを出す

- 物体の機能理解、コップに入る量を判断
- 過去の経験の記憶、記憶からの推論、コップに牛乳を入れる
- 牛乳を元の場所にしまう

補足

- 物体認識、冷蔵庫
- 地図生成、部屋の中
- 経路計画、冷蔵庫がある場所まで
- ナビゲーション
- 障害物回避
- 物体の機能理解、冷蔵庫を開ける、閉める
- 物体の把持、注げられるように持つ
- 運動学習、滑らかに注ぐ

2.2.7 シナリオ、練習問題 4

課題： ペットボトルの蓋を開ける。

前提は、過去に実行したことがある、とする。状況は、目の前にペットボトルが置いてある、とする。

- 物体認識
- 位置測定
- 物体の把持、ペットボトルを掴む
- 物体の把持、ペットボトルの蓋を掴む
- ペットボトルの蓋を回す

補足

- 物体の機能理解、蓋を回すと開く
- 過去の経験の記憶、記憶からの推論、蓋の回し方を過去の経験から思い出す
- 運動学習、蓋の回し方が滑らかになる

2.3 ロボットの行動を分類する

上記の３種類のシナリオで未来のロボットが見せてくれた行動について、それを実現するのに必要な要素を反応行動、計画行動、適応行動、協調行動に分類して、それぞれの内容を簡単に解説する。生物の機能を例として挙げて説明することもある。

また認識機能も数多く活用されていたが、それらについても同様に分類して内容を確認する。

2.3.1 反応行動

外界で起きたことをセンサからの入力（刺激）として受け取り、それに反応して何らかの行動を起こす。生物の場合は、単純な生物ではこの行動しかしないものもある。高等生物であっても、一部の行動は反応行動に分類される。そのような高等生物の行動は、多くの場合考えた結果の行動ではなく、刺激に反応して体が勝手に動く類の行動である場合が多い。

- 反射行動

刺激とそれに対する行動は固定されており、常に一定の行動が呼び出される。行動の内容は定型で変化や調節はされない。例として、熱いものや尖ったものに触ったときに手がサッと引っ込むのは反射行動と考えられる。この時脳で熱いとか考える前に手は引っ込んでいる。刺激が消えると行動も止まる。次のような行動が含まれる。

障害物回避：他の物体との衝突を回避する運動。生物が移動する機能は単細胞生物でも備えている基本的な機能であり、生存のために必要なものである。

注視反応：刺激のある方向に注意を向けること。刺激を放置しないで、その内容に応じて逃げる、近づく、その他の対応をする、などの行動が必要となるので、これも生存のために必要となる。

- 定型行動

刺激の内容に応じて起動される一定の動作の組合せ、系列。例として、人が歩く動作は細かい動きの組合せをいちいち考えなくても両足をうまくタイミングを合わせて動かすことで実現されている。石に躓いて転びそうになると、転倒を防ぐように手や足がサッと出るのは反射行動と定型行動の組合せと考えられる。

- リズム動作生成

歩く、金槌でものを叩く、縄跳びを跳ぶ、など一定のリズムで繰り返される動作を円滑に行うための脳と運動器官を合わせた調整機能。いずれも最初はぎこちないが、習熟するに従ってスムースに実行できるようになる。動作を繰り返すうちに脳内で運動器官に指示を送るタイミングや強度が適切に調整される。

2.3.2 計画行動

前提として、周囲の環境を広い範囲で認識することができ、環境中に存在する物体の配置なども知ることができる。その結果自分の周囲の地図を作成することができ、環境内の自分の位置も知ることができる。地図、自分の位置、他の物体の位置を組合せて判断することで、目的の物体がある位置まで移動して、それに働きかけるといった行動の組合せを実現できる。ここではそのような目的を持った一連の行動の組合せを計画行動と定義している。

- 自己位置同定

自分が移動するときに、移動した距離をセンサ等で計測して知ることができ、環境内の物体を目印、ランドマークとして利用することで環境内での自分の位置を知ることができる。その場合は事前に環境の地図を知っていることが前提となる。

- 経路計画

自己位置同定ができている場合、目標地点と自己位置を地図上で参照することで目標地点まで移動できるようになる。目標地点までどのような経路を通って移動するか、その計画を立案する。

- ナビゲーション

経路計画に沿って移動中は、随時進行方向を確認して必要に応じて修正しながら移動する。経路計画に沿って正しく目標地点まで到達するように、移動に関する動作の内容を決める。また移動中は前述した障害物回避の機能を利用する。

- 地図生成

未知の環境では地図が存在しないので、地図情報を得られない場合は自ら探索しながら地図を作成する。

2.3.3 適応行動

外界で起きたことをセンサからの入力（刺激）として受け取り、それに反応して何らかの行動を起こす。この時、刺激に対する行動を何度も繰り返す間に、意識的であるか無意識であるかに関わらず、行動を変化させることで望ましい行動を獲得していく。

例として、動物が獲物を狩るときに、たまたまその時の状況によって、ある時は風上から追いかけて、別の時は風下から追いかけて、何度も繰り返した結果、風下から追いかけた方が獲物を捕まえる確率が高かったとすると、次第に意識的に風下から追いかけるようになる。

このような、経験に基づいて望ましい結果を得やすい行動に変化させていくことを、適応行動と定義している。人工知能において **学習** と呼ばれる過程である。最初は失敗することがあっても、試行錯誤を重ねるうちに徐々に成功する確率が高くなる。

- 運動学習

行動を繰り返すうちに徐々にうまくなっていくこと。直接結果の成功、失敗から直前の行動を評価することができる。それ以外にも、同じ結果を得るときでも、より円滑な動きや効率の良い動き、正確な動きを獲得することも含

む。何らかの評価基準が必要となる。

- 教師付き学習

学習を行う際に、外部から明示的に望ましい結果、正しい結果、それらを得るための評価基準（教師データ）が与えられている場合の学習である。ロボットにおいては外部からの入力に対して何らかの行動を行った時に、入力に対して対になる正しい行動が与えられる。自分の選択した行動と正しい行動を比較して、正しい行動を選択するための入力ー行動間の関係を獲得していく。

この時、入力ー行動間の関係を、固定された 1 対 1 の関係として記憶してしまうと、ちょっとした状況の変化にも対応できず、応用が効かなくなる。多少入力が変化した場合でも、それに惑わされずに正しい行動を選択できるように学習するのが望ましい。これを **汎化** と呼ぶ。

- 教師なし学習

パターン認識では入力データを統計的性質やその他の基準、制約などに基づいて分類する手法を表す。教師データは与えられないので、入力データがもともと備えている性質だけを用いて、プログラムが自発的に分類や抽象化ができるようになる。

ロボットへの応用としては、様々な入力、入力ー行動間の関係、行動による結果などが数多く蓄積された場合に、それらを分類したり、そこから何らかの法則を発見することができるようになると考えられる。多くのデータが与えられて、これらの間に法則が何かあるか見つけなさい、これらを分類してみなさい、という作業をすることに近い。

- 強化学習

行動の結果を評価する基準が事前に与えられる。様々な状況において、そこで実行可能な行動を列挙しておき、それぞれの行動の生起確率を定めておく。試行錯誤で様々な行動の組合せ（行動系列）を生成して、その結果を基準により評価して、その評価が高くなるように行動系列を生起させる確率を調整していく手法。ある状況において、どの行動を選択するのが最も良い結果に繋がるかを経験により学習していく。複数の行動の時間的な並びに対して適切な学習が行えるので、ロボットの複雑な行動の学習に適用されることが多い。

2.3.4 協調行動

達成したい課題に対して、単一の個体で取り組むよりも、複数の個体で取り組んだ方が良い結果（短時間で終わる、成功する確率が高くなる等）が得られる場合の、複数の個体による行動と定義できる。ある種の社会性を備えると言える。

例として、以下に示す行動が該当する。

- 協調運搬

荷物が大きくて単一の個体では運べない場合に、2 体以上で運ぶ。

- 行動予測

ある個体の手が塞がっていてドアを開けられない場合に、近くに別の個体がいればドアを開けてくれるだろうと期待する。この時のある個体の予測が行動予測となる。

- 観察に基づく協調

ある個体の手が塞がっていてドアを開けられない場合に、近くにいる別の個体が、手が塞がっていて開けられないだろうと推論して代わりにドアを開ける。この時の別の個体の推論と行動が観察に基づく協調となる。

- 群れ行動

部屋の中でものを探す場合に、1 体で探すよりも 2 体以上で場所を分担して探す。同質の作業を並行して行う。

- 役割分担

工場の流れ作業を 1 体ずつが担当するような場合。異質の作業を分担して行い、全体として効率を上げる。

2.3.5 外界と他者の認識、理解

計画行動、適応行動、協調行動を実現する高度な知能では、外界と他者の高度な認識、理解が必要とされる。

まず外界と他者の認識、理解のためには次に示す機能が必要となる。なお前提として、自分と他者はロボットまたは生物で、何らかの知能を持ち能動的に行動できる、物体は知能を持たず行動しない、とする。

- 外界の状況、物体を認識する。
- 物体の機能を理解する。
- 自分、他者の置かれている状況を認識、理解する。
- 自分、他者の振舞いを認識、理解する。
- 自分、他者の意図を認識、理解する。
- 自分、他者の感情を認識、理解する。(生物の場合)
- 自分、他者の間で意思の疎通を行う。

ここでは、そのような認識、理解の種々の機能において、どのような認識、理解の機能要素によりそれらの機能が実現されるかを示す。

- 行為理解

自己の行動の分析、および他者の行動の観察により、意図と行動と結果の関係を理解、予測する。行動の内容を直接理解する段階と、その後の展開を予測する段階がある。「大きなものを運ぼうとしている」、「大きなものを運ぼうとしているが、運べないだろう」、そして「手伝う必要がある」と推論する。

- 模倣

他者の行動を観察して、同じように行動できるように試してみる。動きを真似する単純な見まねや、意図を理解した上で同じ結果を得られるように行動する段階がある。見まねと試行錯誤による運動学習を組み合わせると、見まねを教師データとする学習（模倣学習）となる。

- 身体像

自分の物理的な存在を考えることで、自分の姿勢や動きの過程をシミュレートできる。運動学習に取り組む前段階として、身体像に基づくシミュレーションを内部で行うことで、身体各部をどのように動かすとよいか計画を立てることができる。身体性と同様の概念を表す。

- 物体認識

画像センサや他のセンサからの情報に基づいて、ものが何であるか、形状、位置、姿勢を認識する。物体に働きかける際には、まず物体認識が必要となり、基本的な認識機能の要素となる。

- 物体の機能理解

物体が何であるかが理解できたら、次の段階として、その物体全体でおよび構成要素の各部分でどのような機能を持つか、何に活用できるか、等を理解する必要がある。例として、物体が金槌だと認識した後には、金槌は何に使うか、何ができるか、どこを持ってどのように動かすと機能するか、といった様々な事柄についての理解ができないと、使うことができない。

- 音声認識、合成

人とコミュニケーションする場合の一つの方法として音声による会話がある。人にとって自然で使いやすい、わかりやすいコミュニケーション方法として重要である。音声を聞いて何を言っているかを認識する音声認識と、自分の考えを相手に伝えるための音声合成がある。

- 自然言語理解

音声認識で何を言っているか、単語のレベルまで認識できた後は、その意味を理解する「構文解析、意味解析」が必要となる。この段階が実現できると、相手の意図を正しく理解できる。

- 心の理論、感情の理解、表現

相手が人の場合、心の状態や感情を推定してコミュニケーションに活用することも考えられる。推定した相手の状態に応じて、言葉を選んだり、コミュニケーションの仕方を変化させることで、円滑なコミュニケーションを実現できる。

- 個人適応

特定の個人とコミュニケーションを繰り返すと、相手の特徴、考え方や行動の傾向、癖を認識できる可能性がある。その結果に基づいて、相手にあわせたコミュニケーションを実現できるようになるかもしれない。

2.4 知能の構成論的科学

単純な反応行動を除けば、**知能とは、変動する複雑な環境中で安定に目標を達成する行動能力である。**と定義できる。

これまで取り上げてきた知能の実現方法として、次のような方法が考えられる。

1. **表層的構成論** 、特定の状況を想定してそこで正しく振る舞う知能を作る。そのような個別の知能を数多く作り、その組合せで知能を構成する。**振舞い作り込み** とも呼ばれる。
2. **創発的構成論** 、現在までにわかっている／解明されている、知能を構成する基本的な要素と思われるものを用意して、それらを組合せて環境に投入して、環境との相互作用で新たな知能を生み出す。要素の単純な和ではなく複雑な知能が現れる。例、センサ＋走性＝逃避行動、１＋１＝３。
3. 知能の原理をまず解明し、それに基づいて知能を構成する。(**原理主導構成論、本質的構成論** と呼べよう。※ この 2 語は筆者の造語である。)

ただし、この中のどれか一つだけの取り組みでは高度な知能は実現されていない。

- 1 では特定の状況では正しく振る舞えるが、状況が変化した時の対応ができない。すなわち振舞い作り込みは、見かけ上正しい振舞いができているだけで知能ではない、本質的な解決ではない。
- 2 では反応行動、単純な知能は実際に生み出せることが示されているが、高度な知能は実現されていない。
- 3 では知能の原理を解明するのは困難な作業で、現在までに断片的な知見は得られているが、いつになったら全体像を解明できるかわからない。

高度な知能を実現している現実の存在として生物があるが、生物を分析して得られる知能に関する知識は限定的、局所的であり、それらをどのように組み合わせたら高度な知能を実現できるのかわかっていない。よって、上で示した 3 種類の方法の考え方、手法をうまく組合せながら知能の開発を進める必要がある。

すなわち、知能の原理の解明に継続して取り組みつつ、現在までに局所的、限定的な知能に関する知識しか得られてないとしても、まずそれに基づき知能を構成する個別の機能やそれによるロボットを構成して、現実の環境に投入し、どのような場合に正しく機能するか評価する。また他の機能と組み合わせたらどうなるか、どのような組合せがよいかという点も評価する。評価結果に基づいて、新たな機能やロボットを構成して評価する。このような (試作ー評価ー改善) のループを繰り返すことで、全体として高度な知能に近づけていく。

このような「完全でなくてもまずは作ってみて、それで試しながら改良していく」というアプローチが現在は最も有効であると考えられる。この方法は **進化的構成論** と呼ばれる。

参考文献では、次のように進化的構成論がまとめられている。

「ロボットインテリジェンスの最も基本となる最小単位を見極め、それに基づくロボットを構成し、環境に投入する（創発的構成論)。それがどの程度知的に振舞うかを観測し、何ができないかを明らかにする（結果的振舞いの評価)。そして不足した能力を補う必要最小限の構成要素を付加し（あるいは自律的に獲得させ)（ここで再び創発的構成論)、次のレベルのロボットを構成し、その振舞いを観測する。これを繰り返し、徐々に高度な知的ロボットを構成していく。」

創発的構成論と進化的構成論は、単純な機能の組合せからより複雑、高度な知能を生み出すという点で似ている部分があるが、前者が自発的に知能が生み出されるという立場であるのに対して、後者は評価、改善のプロセスに開発者が積極的に関与する点が異なる。

ここまで述べてきた「知能をどのように実現するか」という問題に対して、様々な方法を比較検討する取り組みは、 **知能の構成論的科学** と呼ばれる。

2.5 フレーム問題

人工知能の分野で古くからある問題で、現実世界で使える高度な人工知能を開発しようとすると、この **フレーム問題** が壁となる。フレーム問題の解決法はまだ見つけられていない。ロボットは現実世界で行動するので、高度な知能の実現を目指すときには同様に問題となる。

フレーム問題とは、ものの性質や状態の記述、行動前の前提条件の記述、行動後のものや外界の状態変化の記述などを厳密に行おうとすると、記述量が膨大になり、実行できなくなることを表す。

例として、少し前に取り上げたシナリオに指定された「冷蔵庫から牛乳（パック）を取り出してきてコップに注ぐ」という行動を、現実世界で行うには様々な制約や問題があることを示す。人間は過去の経験から抽出、蓄積した暗黙の知識を活用して、そのような制約が問題となる状況を無意識のうちに避けている。しかしロボットはそのような知識を持っていないので、事前に用意して記憶させておくか、多くの経験を通して事例ベースで覚えていくしかない。

上記のシナリオに対して考えられる制約、問題として次のような事柄がある。「コップの上で牛乳パックを傾けると注ぐことができる」、という事前の知識があったとしても、以下のような問題が起こり得る。

- コップは口が上を向いていないと牛乳が入らない。
- コップの口が上を向いていても、ものを載せてあって塞がっていたら入らない。
- コップの口が上を向いていて、ものを載せてなくても、誰かがいたずらして急に口を塞いだら入らない。
- コップの口が上を向いていて、ものを載せてなくて、周りに誰もいなくても、扇風機の風で紙が飛んできて口を塞いだら入らない。
- コップの口が上を向いていて、ものを載せてなくて、周りに誰もいなくて、扇風機がなくても、窓が開いていて風が吹いて飛んできた紙が口を塞いだら入らない。
- コップが斜めになっていたら、注ぐときに倒れるかもしれない。
- コップがテーブルの縁に置いてあったら、落ちるかもしれない。
- 牛乳が入ったコップは重くなるので、持ち上げたり動かすには以前より大きな力がいる。
- 注ぎ口を開けていない牛乳パックは、横にしてもこぼれない。取り出し方、運び方、仕舞方に影響する。
- 注ぎ口を開けた牛乳パックは、横にするとこぼれる。取り出し方、運び方、仕舞方に影響する。

- 注ぎ口が開いてないと牛乳を注げない。
- 注ぎ口が開いていても注ぎ口を下にして傾けないと牛乳を注げない。
- 注ぎ口が開いていて注ぎ口を下にしても、注ぎ口が下がるように傾けないと注げない。
- 急に傾けると、一気に牛乳が出てこぼれる。
- パックの中の残りが少ない場合、ある程度以上傾けないと出てこない。

上記をすべて理解してうまくこなせるようになっても、次にコーヒーのパックで同じようにできるか、オレンジジュースのパック、りんごジュースのパックではどうか、となる。これら液体が入っているパックは同じように扱えるという、一般化された知識がないと対応できない。

牛乳パックとコップだけでも上記のような様々な問題があるが、その他のすべてのものに対しても、適切に扱うには多くの暗黙の知識が必要となる。フレーム問題は、そのような知識を作成することが現実的にはできないということを示している。

2.6 生物における知能の進化と階層

生物については少し前に次の 2 点を述べた。

- 高度な知能を実現している現実の存在として生物がある。
- 生物を分析して得られる知能に関する知識は限定的、局所的であり、それらをどのように組み合わせたら高度な知能を実現できるのかわかっていない。

ロボットの知能を実現する際に、既に高度な知能を実現している実装の例として生物があるので、生物の知能を参考にするのは有効である。そのために生物の知能がどう実現されているかを理解できれば有効である。

ここでは生物の知能が進化と共にどのように発達してきたか調べてみる。

2.6.1 生物の構成論的理解

生物を分析して得られる知識は限定されており、そのような解析的な方法だけでは生物の知能を十分に理解することはできない。そこで、生物を調べるだけではなく、これまでにわかった知識から生物と同じように振る舞うもの（ロボット）を作ってその振舞いを調べて、生物と比較することで、生物の知能の理解に役立てようという方法論がある。この方法は、 **生物の構成論的理解** と呼ばれる。創発的構成論、進化的構成論を生物の理解に応用するものと考えられる。

参考文献では、次のように生物の構成論的理解が定義されている。

「生物に関してわかっていることおよび仮説に基づきシステムを合成、構成し、そのシステムを作るだけではなく実際に動かし、その動作を現実の現象と比較し、あるいは未知の事実を探ることで、生物的振舞いに必要な条件を知ったり、仮説をテストする。」

2.6.2 生物の進化、知能の進化

- 化学物質受容体、運動機能、信号伝達機能：　ある種の単細胞生物は、細胞表面に化学物質受容体、センサを持つ。鞭毛モータによる運動機能を持つ。細胞内に化学的信号伝達機能を持つことで、両者を接続し、感覚運動システムを持つ。栄養分に接近し、有害物質を避ける走性を実現する（ゾウリムシ）。
- コミュニケーションと社会性：　進化した単細胞生物は、細胞間化学的信号伝達機能、すなわちコミュニケーションの機能を持つ。これにより同種の生物が集まって集団を形成し、集団として一斉に行動できるようになる。信号伝達物質の濃度勾配により、集団内での自分の位置を知り、それによって行動を変化させる（アメーバ）。
- 多細胞生物、機能の分化：　単細胞生物の集団が多細胞生物に進化していき、大型化する。多細胞生物は最初は同質の細胞が集まっているだけであった（カイメン）。信号伝達物質の濃度勾配による行動の変化は、多細胞生物の身体における機能の変化に繋がる。
- 反射行動、消化管、神経：　多細胞生物は、次第に体内における位置によって異なった機能を持つようになった。効率的に栄養分を取り込むために消化管が生まれた。また消化管に食物を取り込むために口が生まれた（クラゲ、イソギンチャク）。食物を効率的に口に取り込むために、筋肉による運動機能が生まれた（ミミズ）。大型化した身体各部に信号を送るには化学的伝達物質の拡散では遅いため、電気信号による長距離の信号伝達機能（神経系）が生まれた。一方、近距離での信号伝達は化学的伝達物質で行われる。神経により、身体各部の信号伝達が効率よく行われるようになり、単細胞生物の反応行動よりも高度な反応、反射行動が実現できる。
- 運動機能の発達、中枢神経系：　効率的に食物を取り込むために運動機能が発達し、複数の運動器官を持つようになる。それらを適切に調整、制御するために、神経系が局所的に集中するようになり中枢神経系になっていく。食物を口に確実に取り込めるように、触覚、嗅覚、視覚等の感覚器官が口の周囲に集まり、頭部が形成される。中枢神経系は頭部で特に発達し、脳となる。
- 骨格：　外骨格と内骨格

2.7 まとめ

この章の目標

- 反応行動が何を表しているかわかる、説明できる。
- 計画行動が何を表しているかわかる、説明できる。
- 適応行動が何を表しているかわかる、説明できる。
- 協調行動が何を表しているかわかる、説明できる。
- 現実世界で複雑な課題を達成するときに、どのような行動の要素が含まれているか説明できる。
- 反応行動にどのような行動の要素が含まれているかわかる。

- 計画行動にどのような行動の要素が含まれているかわかる。
- 適応行動にどのような行動の要素が含まれているかわかる。
- 協調行動にどのような行動の要素が含まれているかわかる。
- 外界と他者の認識、理解にどのような要素が含まれているかわかる。
- 知能の構成論的科学が何を表しているかわかる、説明できる。
- 表層的構成論が何を表しているかわかる、説明できる。
- 創発的構成論が何を表しているかわかる、説明できる。
- 原理主導構成論が何を表しているかわかる、説明できる。
- 進化的構成論が何を表しているかわかる、説明できる。
- フレーム問題が何を表しているかわかる、説明できる。
- 生物の構成論的理解が何を表しているかわかる、説明できる。

練習問題

1. 椅子に座るという行動に、どのような行動の要素が含まれているか考えなさい。
2. 服を着るという行動に、どのような行動の要素が含まれているか考えなさい。
3. 冷蔵庫にしまってある牛乳を出してきてコップに注ぐという行動に、どのような行動の要素が含まれているか考えなさい。
4. シナリオ４を考えて、どのような行動の要素が含まれているか考えなさい。
5. 反応行動とはどのような行動か、簡単に説明しなさい。
6. ゾウリムシを想定して、反応行動の例を考えなさい。
7. ヒトを想定して、反応行動の例を考えなさい。
8. 計画行動とはどのような行動か、簡単に説明しなさい。
9. 適当な動物を想定して、計画行動の例を考えなさい。
10. 適応行動とはどのような行動か、簡単に説明しなさい。
11. 適当な動物を想定して、適応行動の例を考えなさい。
12. 協調行動とはどのような行動か、簡単に説明しなさい。
13. 適当な動物を想定して、協調行動の例を考えなさい。
14. 外界と他者の認識、理解が何を表しているか、簡単に説明しなさい。
15. 適当な動物を想定して、外界と他者の認識、理解の例を考えなさい。

16. 知能の構成論的科学が何を表しているか、簡単に説明しなさい。
17. 表層的構成論の内容を簡単に説明しなさい。
18. 創発的構成論の内容を簡単に説明しなさい。
19. 原理主導構成論の内容を簡単に説明しなさい。
20. 進化的構成論の内容を簡単に説明しなさい。
21. フレーム問題の内容を簡単に説明しなさい。
22.「冷蔵庫から牛乳（パック）を取り出してきてコップに注ぐ」行動に関して、フレーム問題に該当する事柄を挙げなさい。
23. 何らかの行動を想定して、それに関してフレーム問題に該当する事柄を挙げなさい。
24. 生物の構成論的理解が何を表しているか、簡単に説明しなさい。

参考資料

ゾウリムシは単細胞生物で、細胞の長さは 90-150Âţm、幅は 40Âţm 程度である。形状は円筒形に近く、中腹には細胞口というくぼみがある。細胞表面には約 3500 本の繊毛を持っており、その繊毛を使って泳ぐ。浸透圧を調整するための収縮胞という組織がある。細胞口の奥には細胞内へ餌を取り込む細胞咽口があり、餌はここを通って食胞に取り込まれる。食胞内で消化が行われ、有用な成分は細胞内へ吸収されながら、食胞は細胞内を回るように移動する。排泄物は細胞後方の細胞肛門から放出される。生殖機能も有する。（出典：「ゾウリムシ」『フリー百科事典　ウィキペディア日本語版』。2016 年 7 月 31 日 (日) 07:27　UTC、URL: http://ja.wikipedia.org から引用、要約した）

第 3 章

反応行動

反応行動 とはロボットに対して入力される外部の情報（様々なセンサからの情報）に反応して行われる、比較的単純な行動である。反応行動のレベルで利用できる外部からの入力は、

- 接触、力、空気や水の流れ、光、匂い、音、温度、湿度、化学物質などが該当する。
- 自分の周囲の状況は非常に限られた狭い範囲でしか知ることができない。
- 自分から能動的に情報を得ることができず、外部からの刺激を受動的に受け取るのみ。

などの特徴があり、これらの入力が反応行動を引き起こす手がかりとなる。また入力に対しての反応は常に一定で変化することはない。

3.1 反応行動に関する実習課題

課題の名称は ri1 とする。自分の周囲にある食物や毒物を探索可能なセンサを持つゾウリムシの行動を規定する知能のプログラムを作成する。ここで目指すのは、なるべく多くの食物を摂取し、毒物は避け、少しでも長い期間生存し続ける行動戦略の実現である。

ROS 上のシミュレータ world 内で行動するゾウリムシの行動を規定するプログラム er14000.cpp を作成する。センサの探索可能範囲は距離において 0, 5, 20 の 3 種類とする。詳細はこの後のインストールと ri1 に関する説明文書を参照する。

3.2 実習で使うソフトウェア開発キットのインストール (ver.1.1 20190927)

3.2.1 インストールの手順

以下の説明で %__WORKSPACE, %__SOMEWHERE, %__PACKAGE は適切な内容で読み替える。これらの文字列をそのままコピー、ペーストしてはいけない。

- %__WORKSPACE は新規作成、または既存のワークスペースの名前 (/home/username/rosws など、任意に決められる）
- %__SOMEWHERE はパッケージをダウンロードして展開したディレクトリ、(/home/username/ダウンロード など)
- %__PACKAGE はパッケージの名前、 (ri1, ri2, ri3 など) %__SAMPLE は例として付属するプログラムの名前、（er14000, er14000a など）
- %__SOURCE は作成するプログラムの名前、（通常自分の学籍番号 er19099 など）で置き換える。

以下のコマンドの例でコマンド文字列の前の４桁の数字は入力しない。

パッケージをダウンロード、%__SOMEWHERE に展開しておく。その結果、%__PACKAGE のディレクトリができる。

ワークスペースの作成から始める人はここから

```
1944  $ mkdir %__WORKSPACE
1946  $ cd %__WORKSPACE
1947  $ mkdir src
1948  $ cd src
1950  $ catkin_init_workspace
1951  $ cd ..
1953  $ catkin_make
1954  $ source devel/setup.bash
1955  $ echo $ROS_PACKAGE_PATH        // PATH が正しく設定されたか確認
```

そのまま次へ続く。

既存のワークスペース %__WORKSPACE があって、パッケージの追加から始める人はここから

（正しく作られているパッケージをインストールする場合は、ワークスペースの作成後、src の下にパッケージをコピーして、catkin_make でビルドできる。）

ワークスペースの src に移動してから、そこにパッケージをコピーする。

```
1959  $ cd %__WORKSPACE/src
1960  $ cp %__SOMEWHERE/%__PACKAGE .
```

3.2.2 プログラムのビルド、実行

例としてサンプルプログラム %__SAMPLE.cpp をビルド、実行してみる

ROS の version melodic に対応させる過程で、設定ファイルの内容が一部変更されたので、version に合った設定ファイルをコピーする。

```
1961  $ cd %__WORKSPACE/src/%__PACKAGE (%__PACKAGE に移動)

(1) ROS の version が indigo, kinetic の場合は以下を実行する。

1962  $ cp CMakeLists.txt.kinetic CMakeLists.txt
1963  $ cp package.xml.kinetic package.xml

(2) ROS の version が melodic の場合は以下を実行する。

1964  $ cp CMakeLists.txt.melodic CMakeLists.txt
1965  $ cp package.xml.melodic package.xml
```

ビルドする。

```
1969  $ cd ../..  または cd %__WORKSPACE (%__WORKSPACE に移動)
1970  $ catkin_make
```

エラーがあれば赤色のメッセージが表示されるので、それを参考にして修正する。成功すれば緑色のメッセージと 100% が表示される。

実行時に必要な設定ファイルを読み込む。

```
1971  $ source devel/setup.bash
```

実行時は roscore, world, %__SAMPLE の 3 個を別々のターミナルから起動する。world (ri2 では world2, ri3 では world3)　は環境を実現する。%__SAMPLE はゾウリムシやロボットの行動を決める知能である。

```
1980  $ roscore
1981  $ rosrun %__PACKAGE world
1982  $ rosrun %__PACKAGE %__SAMPLE
```

3.2.3 プログラムの編集

ソースファイル (.cpp) を作成 (または既存のファイルをコピー)、編集する。通常はサンプルプログラムを自分の学籍番号のファイル名でコピーしてそれを編集する。

```
1961  $ cd %__PACKAGE/src
1962  $ cp %__SAMPLE.cpp %__SOURCE.cpp
1963  $ vim %__SOURCE.cpp
```

プログラムで変更すべき点は以下の部分である。////////// で囲まれた部分以外を書き換えるとプログラムが動かなくなるので注意する。

```
//////////////////////////////////////////////

// Write your intelligence here. の部分を自分で考えた内容に変更する。

//////////////////////////////////////////////
```

- 独自のデータ構造や関数が必要な場合は作成してもよい。それらはすべて %__SOURCE.cpp　の中に記述すること。ファイルを分割してはいけない。

CMakeLists.txt に自分の実行ファイルの名前を登録する。

```
$ cd ..    (%__WORKSPACE/src/%__PACKAGE/ に移動)
$ vim CMakeLists.txt
```

ファイルの下の方に以下と類似の 2 行があるので、これを真似して自分の実行ファイルの設定を追加する。CMakeLists.txt は 2 箇所にあるので場所に注意する。下記の 2 行が含まれていない場合は場所が間違っている。

```
add_executable(%__SAMPLE src/%__SAMPLE.cpp)
target_link_libraries(%__SAMPLE ${catkin_LIBRARIES})
```

以降は上に書いたビルド、実行の手順と同じ。

```
$ cd ../..    (%__WORKSPACE に移動)
$ catkin_make
...
```

3.2.4 設定ファイルの読み込みについて

設定ファイルの読み込みを端末起動時に実行するように登録しておくと毎回手動で実行する必要がなくなる。

```
1985  $ cd
1986  $ vim .bashrc
```

開いたファイルの末尾に次の 2 行のコマンドを書き加えて改行して変更した内容を保存する。

```
source /opt/ros/VERSION/setup.bash
source %__WORKSPACE/devel/setup.bash
```

VERSION (indigo, kinetic 等) と %__WORKSPACE は自分の環境に合わせて書き替える。

3.2.5 その他

テレポート、ワープの機能は開発中の確認作業で使うことは問題ないが、競技中は使ってはいけない。

3.2.6 トラブルシューティング

1. 赤色のエラーメッセージが出て roscore の起動に失敗する場合

turtlebot の実習で、ROS の通信をネットワークを介して外部の PC と行うように設定した場合に赤色のエラーメッセージが出て roscore の起動に失敗する。対策は、~/.bashrc の末尾に ROS の通信に関する設定が以下のように追加されているので、

```
export ROS... ...
```

その部分を次のようにコメントアウトして端末を起動し直す。

```
# export ROS... ...
```

turtlebot の実習をするときには元に戻す。

2. ディレクトリ名が日本語、全角文字になっていると、ビルドや実行時に様々なファイルが見つからないエラーとなる。

対策は、ディレクトリ名、ファイル名は半角英数字にする。

3. ビルド時に、message_generation.cmake がないというエラーが出る。

対策は、動いている人の /opt/ros/VERSION/share/ から message_generation のディレクトリを自分の PC の同じ場所にコピーする。アクセス権でエラーとなる場合は chmod で修正する。

4. ビルド時に、turtlesim/Pose.h がないというエラーが出る。

対策は、動いている人の /opt/ros/VERSION/include/ から turtlesim のディレクトリを自分の PC の同じ場所にコピーする。アクセス権でエラーとなる場合は chmod で修正する。

5. サンプルプログラム (%__SAMPLE.cpp など) をコンパイルするときに#include <ri1/Pose.h> が見つからなくてエラーとなる場合がある。

対策は、この行をコメントアウトする。調べた結果、このヘッダの内容は現在参照していないので読み込まなくてもよい。

6. rosrun で実行しようとすると、rospack で package not found. のエラーが出る。

~/.bashrc に source %__WORKSPACE/devel/setup.bsah を書き忘れている。

3.2.7 備忘録

- ROS のコマンドの例

```
2001  $ rosservice list
2002  $ rosservice type /turtle1/teleport_absolute | rossrv show
2003  $ rosservice call /turtle1/teleport_absolute 20 20 0
```

- ROS melodic に対応させる過程で、turtle_frame.cpp の８０行付近で以下の変更がなされた。if (FALSE) → if (false)
- ri1 の新しいパッケージで修正されたので現在下記の変更は不要。

※インストール直後、最初の１回だけ次の間違い（（２箇所））を直す。

```
1967  $ vim turtle_frame.cpp
```

６０行目付近、３００行目付近の

```
QString images_path = (ros::package::getPath("turtlesim") + "/images/").c_str();
```

を

```
QString images_path = (ros::package::getPath("ri1") + "/images/").c_str();
```

に変更する。

3.3 ROS を使った実習課題 1 (ver.1.0.6, 20190927)

- パッケージのインストールは 「3.2 実習で使うソフトウェア開発キットのインストール」 を参照する。
- パッケージのファイルは http://edu.isc.chubu.ac.jp/naga/ri106.tgz をダウンロードする。
- %__PACKAGE は ri1
- %__SAMPLE は er14000, er14000a
- ビルドや実行時にエラーがあって動かない場合は、「3.2 実習で使うソフトウェア開発キットのインストール」 のトラブルシューティングを参照する。

- ROS melodic に対応するため、設定ファイルと手順を一部変更した。詳細は 「3.2 実習で使うソフトウェア開発キットのインストール」 を参照する。

3.3.1 目的

仮想環境 World 上で行動するゾウリムシの知能を規定するプログラムを作成して、反応行動（と、計画行動の一部）の実装を学ぶ。

ゾウリムシは環境中の栄養を摂取し、毒物を摂取しないように回避し、一定時間経過後になるべく多くの生命ポイント (life) を保持することを目指す。

3.3.2 環境とゾウリムシの仕様

- 環境には１０１個の栄養（食物)、１０個の毒物がランダムな場所に配置される。
- ゾウリムシと栄養または毒物との距離が２未満になると摂取される。
- ゾウリムシは初期値 10000 の生命ポイントを持ち、行動によって増減する。
- 栄養を摂取すると生命ポイントが２００増える。
- 毒物を摂取すると生命ポイントが１０００減る。
- ゾウリムシは生命を維持するために一定時間経過するごとに、生命ポイントが減る。
- ゾウリムシが移動すると、エネルギーを消費するため生命ポイントが減る。減り方は速度の二乗に比例する。
- 回転運動では生命ポイントは減らない。
- ゾウリムシはセンサを持ち、栄養または毒物までの距離と種別を知ることができる。

3.3.3 World の機能

- 座標は縦横１００，単位はない
- 左上が原点 (0, 0), 右下が (99, 99) (※描画機能の修正により原点は左下から変更された)
- 角度は画面に向かって右が 0 度、上が 90 度、左が 180 度、下が 270 度となる。変数には 0 -- 2*PI radian で代入される。
- センサで検出した物体は赤くなる
- 環境からゾウリムシに向けて、１秒間に 20 回、座標や検出したものの情報が伝えられる。
- ゾウリムシは伝えられた情報をもとに、次の行動を決めて環境に返信する。

- １回の返信では、速度の変更と回転速度の変更を各１回のみ送信できる。
- 起動時のオプションは次のとおり

```
$ rosrun ri1 world [-r 整数] [-s 整数] [-t 秒数]
```

- -r は乱数の種を指定する、０を指定すると毎回異なる。デフォルト値は０
- -s はセンサの計測可能範囲、デフォルト値は０
- -t は実行時間（秒)、デフォルト値は１２０

実行例は以下のとおり。

```
$ rosrun ri1 world -r 123     // 乱数の種が 123
$ rosrun ri1 world -s 10      // センサの探索範囲が 10
$ rosrun ri1 world -t 60      // world を 60 秒間実行する
$ rosrun ri1 world -r 456 -s 20 -t 60   // 同時に複数指定可能、順不同
```

3.3.4 サンプルプログラムの動かし方

インストールマニュアルにある手順に従って、インストールとビルドができると以下のサンプルプログラムを動かすことができる。

1. er14000 はセンサを使わない単純な動作をする

```
1980  $ roscore
1981  $ rosrun ri1 world
1982  $ rosrun ri1 er14000
```

er14000 が終了した後、world を終了したら次のプログラムを試すことができる。

2. er14000a はセンサを使って見つけたものを避ける

```
1983  $ rosrun ri1 world -s 5
1984  $ rosrun ri1 er14000a
```

3.3.5 プログラムの説明

- 実行ファイル world は Window の管理、描画、ゾウリムシの管理、食物の管理、生命ポイントの管理、er14000 への情報の送信を行う。
- 実行ファイル er14000 はゾウリムシの知能部分だけを実現し、速度と角速度を world に送信する。センサは使っておらず、周期的に角速度を変更している。

- 実行ファイル er14000a はゾウリムシの知能部分だけを実現し、速度と角速度を world に送信する。センサを使って周囲の物体の位置を測定し、衝突しないように角速度を変更している。

er14000.cpp において、Write your intelligence here. の前後の部分がサンプルの行動を規定している。

- 以下は World から受け取った情報を画面に表示している。
- 不要であればコメントアウト可

```
ROS_INFO("x:%.1f y:%.1f o:%.1f v:%.1f a:%.1f life:%.1f num:%d",
         x, y, orient, v, a, life, numfinds);
for (int i = 0; i < numfinds; i++) {
    ROS_INFO("fx[%d]%.1f fy[%d]%.1f fkind[%d]%d ",
             i, fx[i], i, fy[i], i, fkind[i]);
}
```

- 変数が表す内容、x はゾウリムシの x 座標, y はゾウリムシの y 座標, orient はゾウリムシの向いている方向, v はゾウリムシの速度, a はゾウリムシの角速度, life はゾウリムシの生命ポイント, numfinds はゾウリムシが見つけたものの個数となっている。fx[i] は見つけたもの i 番目の x 座標, fy[i] は見つけたもの i 番目の y 座標, fkind[i] は見つけたもの i 番目の種類（栄養:0 か 毒物:1）を表す。

```
static int timing = 0;    // timing control
static float apub = 8.0; // angular velocity for publish
static float vpub = 3.0; // linear velocity for publish
```

- 変数が表す内容、timing は速度、角速度を変更するタイミングの調整用カウンタ, apub は world への送信用の角速度, vpub は同じく送信用の速度,

```
if (0 == (timing++ % 10)) {
    twist.angular.z = apub;
    apub -= 0.1;
}
else {
    twist.linear.x = vpub;
}
```

- この if 文で、10 回に 1 回だけ角速度を変化させ、残りの 9 回は一定の速度を指定している。

サンプルのプログラム er14000 では見つけたものの情報を活用していないので、er14000a を参考にして見つけたものの位置や種類にあわせて行動を変化させられるような知能を考える。

3.3.6 課題

２年生は 1 に、３年生は 2 に取り組む。

1. センサで距離 5 未満のものを検出できる環境 world -s 5 におけるゾウリムシの行動を実装する。ソースファイル名は er?????a.cpp とする。

2. センサで距離 20 未満のものを検出できる環境 world -s 20 におけるゾウリムシの行動を実装する。ソースファイル名は er?????b.cpp とする。

1, 2 のそれぞれで、実現したい機能を箇条書きの項目で書き表し、個々の機能を順次プログラムで記述する。

それぞれのプログラムを１２０秒間実行して最後に表示された生命ポイントを記録する。５回実行して平均を求める。

3. 考案した知能（行動戦略）において、移動速度を速くした場合、遅くした場合のプログラム終了時の残存エネルギーの量を比較してみる。

以下の課題は古い内容で、現在は取り組まなくてもよい。興味のある人は取り組んでみてもよい。

0. センサが使えない環境 world -s 0 におけるゾウリムシの行動を実装する。ソースファイル名は er?????c.cpp とする。

3.3.7 レポートについて

レポートは、上記課題 1, 2 について、

- 実現しようとした機能を箇条書きで表す。
- 「自分が書いた部分」のソースリスト
- プログラムの説明
- 結果、５回実行したときの生命ポイントと平均値
- 結果についての考察、感想

を含むように、なんらかの文書作成ソフトを使用して作成し、PDF ファイルを作成する。ファイル名は er?????ab.pdf とする。

自分が書いていない部分も含めてプログラム全体を

- er?????a.cpp (1) のソースファイル
- er?????b.cpp (2) のソースファイル

とする。

レポートの PDF ファイル１個、ソースリストのファイルを Google Classroom のこの科目のクラス、ロボットインテリジェンスの課題提出のページから提出する。提出期限は 202?年 ?? 月 ?? 日 23:59。

3.3.8 その他

- センサで検出したものの座標等が１０個しか正しく送信されないバグが見つかったので、検出したものすべてについて正しく送信されるように turtle.cpp を修正した。(2016/11/03)

3.4 まとめ

この章の目標

- 反応行動が何を表しているかわかる、説明できる。
- 課題 ri1 の目標を実現するプログラムを作成して、反応行動に相当する知能で何ができるかを理解する。

練習問題

1. 反応行動とはどのような行動か、簡単に説明しなさい。

第 4 章

計画行動

計画行動 とはロボットを取り巻く環境内で自分（ロボット）の位置および達成目標に関係する物体の位置を把握して、目標達成のために計画する移動や運動を表す。計画行動レベルでは周囲の状況をより広く知ることができるようになり、外界がどうなっていて、それに対して次の行動をどうするか決めることができる。

4.1 SLAM, 環境地図作成と自己位置同定

移動ロボットが広い環境内で目標達成のために適切に行動できるためには、自分の周囲の状況を把握、認識している必要がある。環境を表現する完全な地図情報と現在位置を事前に与えられていなければ、行動しながら **環境地図作成** と **自己位置同定** を並行して進めることになる。環境地図作成と自己位置同定を並行して行う処理を、英語では **SLAM** (Simultaneous localization and mapping) と呼ぶ。

環境地図の作成では、現在位置から計測できる物体を地図に記録していく。移動の際に物体は障害物となりそこには移動することができないので、物体の位置を把握することは移動できる場所、できない場所を区別するのに重要な情報となる。また特徴のある物体は、自己位置同定の際の目印、 **ランドマーク** として活用できる。

自己位置同定では、作成している環境地図中の物体との相対的な距離、角度から、地図中で自分が現在いる位置を同定することができる。

4.1.1 座標系

環境に関する情報や環境地図が事前に与えられない場合にロボットが行動する場合は、基準となる座標系がないので最初は **自己中心座標系** から行動を開始するしかない。自己中心座標系では自分の現在位置が座標の原点、自分の向いている方向が基準の方向、角度（例えば０度）となる。自己中心座標系は自分が移動、回転するのにあわせて原点や基準の方向が変化するので、自分以外の物体との相互の位置関係を統一して扱うのが難しい。

行動しながら環境地図の作成を行い、ランドマークがいくつか登録され、ある程度環境地図の情報が充実してくると、**世界中心座標系** に移行することができる。世界中心座標系では、基準となるランドマークを一つまたは複数決めて、それらの座標と相互の位置関係から、基準となる座標（原点）と方向を定めることができる。世界中心座

標系は変化せず一定となる。

世界中心座標系において移動するには次の情報が必要となる。

1. 環境内のランドマークの位置の推定
2. ランドマークの位置の相互関係を把握し、環境地図を作成する
3. ランドマークの配置から世界中心座標系を設定する
4. 世界中心座標系と自己中心座標系との間の相互変換の式を求める
5. 世界中心座標系における自己位置推定

移動経路を決める際には次の情報が必要となる。

1. 自身の位置
2. 目的地の位置
3. ランドマークの位置
4. 環境内でロボットが移動可能、存在可能な自由領域と、存在できない障害物領域に分類する

4.1.2 デッドレコニング

デッドレコニング (Dead Reckoning) とは、自己位置同定において外部からの情報を参照しないで、ロボットが内部に持つセンサの情報のみを用いて位置を推定する手法である。使用するセンサの種類によって、 **オドメトリ** と **慣性航法装置**（Inertial Navigation System, INS）に分類することができる。

オドメトリは車輪で走行する種類のロボットや自動車、その他の移動体で用いられる。車輪の回転速度を測定するエンコーダを装着しておいて、各車輪のエンコーダの出力から移動した距離を算出する。左右の車輪の回転数の差によって旋回する場合は、その差から旋回角度を算出する。ステアリングで旋回する場合は、ステアリングの回転状態から旋回角度を算出する。これらの結果から、走行中の各瞬間の移動距離と旋回角度が得られる。出発点における位置と方向にこれらの値を累積して加算していくことで、移動距離と角度、すなわち現在位置を推定することができる。

身近な例では、カーナビゲーションシステムはオドメトリを備えており、トンネル内部などで GPS 信号が受信できない場合はオドメトリで現在位置を推定する。

車輪を持たないロボット、航空機、船舶、潜水艦などでは慣性航法装置が使用される。慣性航法装置では、前後、左右、上下の 3 方向の加速度センサにより 3 方向の並進加速度を計測することができる。また **ジャイロスコープ** 、**ジャイロセンサ** を備えており、方角を計測することができる。加速度センサで検出した加速度を積分することで速度が得られ、速度を積分することで移動距離を求めることができる。この結果と方角を合成することで、出発点からの移動距離や現在位置を推定することができる。慣性航法装置のように外部の補助を受けずに独立して航法が行える装置を**自立航法装置** と呼ぶ。

オドメトリと慣性航法装置のどちらのシステムも誤差が累積されていくので、長時間経過後は推定誤差が生じてしまう。可能であれば一定時間経過する毎に他の情報を用いて誤差を補正するとよい。

4.2 計画行動に関する実習課題

課題の名称は ri2 とする。自分の前方にある物体とそこまでの距離を探索可能なセンサを持つロボットの行動を規定する知能のプログラムを作成する。ここで目指すのは、自機の周囲にある物体の種類と距離から、現在位置を推定して目標地点（ゴール）まで移動する行動戦略の実現である。

ROS 上のシミュレータ world2 内で行動するロボットの行動を規定するプログラム er14000d.cpp を作成する。ロボットは次に示す 3 種類のセンサを備える。

- 前方にある物体の種類とそこまでの距離が分かる距離センサ
- 自機の絶対座標がわかる GPS センサ
- 自機の向いている方向が分かるジャイロセンサ

詳細はこの後のインストールと ri2 に関する説明文書を参照する。

4.3 ROS を使った実習課題 2 (ver.1.0.6, 20191010)

- パッケージのインストールは 「3.2 実習で使うソフトウェア開発キットのインストール」 を参照する。
- パッケージのファイルは http://edu.isc.chubu.ac.jp/naga/ri206.tgz をダウンロードする。
- %__PACKAGE は ri2
- %__SAMPLE は er14000d
- ビルドや実行時にエラーがあって動かない場合は、「3.2 実習で使うソフトウェア開発キットのインストール」 のトラブルシューティングを参照する。

4.3.1 目的

仮想環境 World2 上で行動するロボットの知能を規定するプログラム er14000d を作成して、計画行動（と、自己位置推定、環境地図作成、デッドレコニング）の実装を学ぶ。

ロボットは前方に距離センサを 1 個備えており、前方正面にある物体までの距離と種別を計測することができる。目的は、位置と種別が既知の物体との相対的な位置を計測し、その結果から自己位置を推定し、環境地図を作成して出力することである。その後環境の中心座標 (50, 50) まで移動する。ここまでの作業をなるべく短時間で達成することを目指す。可能であれば外部から与えられる自己位置の情報を参照しないで、デッドレコニングを実現する。

4.3.2 環境とロボットの仕様

- 環境には黄色のゴール (大きさは縦 10、横 1、中心座標 (0, 50)) と青色のゴール (大きさは縦 10、横 1、中心座標 (100, 50)) がある。障害物 (四角形、大きさは縦横 1 以上、中心座標は不定) が 0 個以上ある。
- ロボットは初期値 10000 のエネルギーを持ち、行動によって増減する。
- ロボットは稼動状態を維持するために一定時間経過するごとに、エネルギーが減る。
- ロボットが移動すると、エネルギーを消費するためエネルギーが減る。減り方は速度の二乗に比例する。
- 回転運動ではエネルギーは減らない。
- なるべくエネルギーを減らさないように目的を達成することを目指す。

4.3.3 World2 の機能

- 座標は縦横１００，単位はない
- 左上が原点
- センサで検出した物体の座標は赤く表示される。
- 環境からロボットに向けて、１秒間に２０回、検出した座標やものの種別の情報が伝えられる。
- ロボットは伝えられた情報をもとに、次の行動を決めて環境に返信する。
- １回の返信では、速度の変更と回転速度の変更を各１回のみ送信できる。
- 起動時のオプションは次のとおり

```
$ rosrun ri2 world2 [-fxyon/fxyoff] [-gpson/gpsoff] [-gyroon/gyrooff]
  [-mapon/mapoff] [-n 整数] [-o 実数] [-r 整数] [-s 実数] [-t 秒数]
  [-xyo X 座標 Y 座標 方向]
```

- -fxyon/fxyoff は距離センサで検出した座標の送信の可否を指定する。デフォルト値は fxyoff で送信しない。
- -gpson/gpsoff は GPS で検出したロボットの座標の送信の可否を指定する。デフォルト値は gpsoff で送信しない。
- -gyroon/gyrooff はジャイロセンサで検出した角度の送信の可否を指定する。デフォルト値は gyrooff で送信しない。
- -mapon/mapoff は 障害物の位置を描いた環境地図の記録の可否を指定する。デフォルト値は mapon で記録する。
- -n は障害物の個数を指定する、デフォルト値は 1。
- -o は障害物の辺の長さを指定する、0 を指定すると 5--20 の範囲で乱数で決まる。デフォルト値は 10。

- -r は乱数の種を指定する、0 を指定すると時刻を乱数の種として毎回異なる乱数が得られる。デフォルト値は 0。乱数の種を同一の値にするとロボットや障害物を同じ配置にすることができるので、再現性のあるシミュレーションが可能。
- -s はセンサの計測可能範囲、デフォルト値は 142 で, 対角線も届き、直線上で見えているものは壁を含めてすべての物体を計測可能。
- -t は実行時間（秒)、デフォルト値は 120
- -xyo はロボットの初期座標と向き、このオプションの後に 3 個の実数を指定する。特定の位置から再現性のあるシミュレーションが可能。

実行例は以下のとおり。

```
$ rosrun ri2 world2 -fxyon -gpson -gyroon // 距離センサとジャイロと GPS を ON
$ rosrun ri2 world2 -gpson // GPS は ON (距離センサとジャイロは OFF)
$ rosrun ri2 world2 -mapoff // 環境地図を記録しない
$ rosrun ri2 world2 -r 123    // 乱数の種が 123
$ rosrun ri2 world2 -s 10     // センサの探索範囲が 10
$ rosrun ri2 world2 -r 456 -s 20 -t 60   // オプションは同時に複数指定可能
$ rosrun ri2 world2 -n 10 -o 10  // 障害物の大きさが 10、個数が 10 個
$ rosrun ri2 world2 -xyo 10 10 315 // 座標 (10,10)、315 度の方向に配置する
```

4.3.4 サンプルプログラムの動かし方

インストールマニュアルにある手順に従って、インストールとビルドができると以下のサンプルプログラムを動かすことができる。

```
1980  $ roscore
1981  $ rosrun ri2 world2 -fxyon -gpson -gyroon -mapoff -n 0 // 課題 (1)
1982  $ rosrun ri2 er14000d

1983  $ rosrun ri2 world2 -mapoff -n 0 // 課題 (4)
1984  $ rosrun ri2 world2              // 課題 (5)
```

4.3.5 プログラムの説明

- ゾウリムシでは、計測、通信、移動の順で処理されていたが、今回は移動、計測、通信の順で処理される。この変更により通信の内容と現在位置のずれがなくなる。
- 実行ファイル world2 は Window の管理、描画、ロボットの管理、障害物の管理、エネルギーの管理、er14000d への情報の送信を行う。
- 実行ファイル er14000d はロボットの知能部分だけを実現し、速度と角速度を world2 に送信する。

er14000d.cpp において、88 行目付近、Write your intelligence here. の前後の部分がサンプルの行動を規定している。

- 以下は World2 から受け取った情報を画面に表示している。
- 不要であればコメントアウト可

```
//  ROS_INFO("x:%.1f y:%.1f o:%.1f v:%.1f a:%.1f life:%.1f num:%d",
//         x, y, orient, v, a, life, numfinds);
    for (int i = 0; i < numfinds; i++) {
        ROS_INFO("cross point theta:%.1f fx[%d]%.1f fy[%d]%.1f fd[%d]%.1f
                fkind[%d]%d ", orient*180.0/PI, i, fx[i], i, fy[i], i,
                fd[i], i, fkind[i]);
    }
```

- 以下の部分を自分のプログラムで置き換える。ri1 と同様。

```
// Write your intelligence here.
```

- 変数が表す内容

（入力）
x **はロボットの** x **座標** （GPS **に相当、※設定により有無を決める**）
y **はロボットの** y **座標** （GPS **に相当、※設定により有無を決める**）
orient **はロボットの向いている方向** （**ジャイロに相当、※設定により有無を決める**）
v **はロボットの速度**
a **はロボットの角速度**
life **はロボットのエネルギー**
numfinds **はロボットが見つけたものの個数となっている。この課題では** 1 **か** 0
fx[i] **は見つけた座標** i **番目の** x **座標** （**※設定により有無を決める**）
fy[i] **は見つけた座標** i **番目の** y **座標** （**※設定により有無を決める**）
fd[i] **は見つけた座標** i **番目までの距離**
fkind[i] **は見つけたもの** i **番目の種類（壁，青ゴール，黄ゴール，障害物）を表す。**

（出力）
apub **は** world2 **への送信用の角速度**
vpub **は** world2 **への送信用の速度**

fkind[i] の参照時など、種別の数値は KINDOBST など、以下の「名前」を使うこと。数値を直接書かない。数値は場合によって変更される可能性があるため。

```
#define KINDWALL        0       // 壁
#define KINDYELLOWGOAL  1       // 黄色ゴール
#define KINDBLUEGOAL    2       // 青ゴール
#define KINDOBST        3       // 障害物
```

- 他の変数が表す内容、timing は速度、角速度を変更するタイミングの調整用カウンタ

```
static int timing = 0;   // timing control
static float apub = 1.0; // angular velocity for publish
static float vpub = 1.0; // linear velocity for publish
```

- // ????? Q1 ????? と // ????? Q2 ?????　の 2 個の if 文はある動きを実現する。読み解こう。コメントを外して実行してみるとわかる。

```
if ((0 == timing++ % 5) &&
    (x < 4 || 100 - 4 < x || y < 4 || 100 - 4 < y || fd[0] < 4)) {
    twist.angular.z = 113 * 20 * PI /180.0;
}
else {
    twist.linear.x = vpub;
}
```

- このプログラムでは壁または障害物にぶつかりそうになったら、5 回に 1 回 113 度回転する。5 回に 1 回の理由は間に 4 回直進が入ることで衝突予想地点から離れられるため。

※　このプログラムはロボットの座標の情報を使っているため、GPS が OFF の設定ではくるくる回ってしまって正しく動かない。GPS を ON にすると障害物を避けて移動する。

- 以下は環境地図を作成、記録するプログラムのサンプルである。作成したファイルはワークスペースの下に置かれる。地図を記録する配列 mymap[100][100] が予め用意され初期化されているので、そこに結果を記録して、地図が完成したら下記のようにファイルに出力する。

```
static int mapsw = ON; // 通常は OFF、地図データを記録する場合に一度だけ
                       // ON にする。ON にし続けると HDD が地図データの
                       // ファイルで埋め尽くされるので注意する。
if (ON == mapsw) {
    mapsw = OFF; // 一回だけ記録して OFF にする処理。
    int sx = 20; // 障害物の左上の座標、x 座標
    int sy = 30; // 障害物の左上の座標、y 座標
    int hsize = 10; // 障害物の横のサイズ
    int vsize = 10; // 障害物の縦のサイズ
    for (int y = sy; y < sy + vsize; y++) {
        for (int x = sx; x < sx + hsize; x++) {
            mymap[y][x] = 255; // 障害物の位置に 255 を書く。1 ではなくなった。
        }
    }
    const char prefix[] = "er14000-"; // ファイル名の先頭文字列
    savemap(mymap, prefix); // あとはこの関数が記録してくれる。
}
```

4.3.6 課題

実現したい機能を箇条書きの項目で書き表し、個々の機能を順次プログラムで記述する。ソースファイル名は er?????d.cpp とする。

1. GPS, ジャイロセンサの情報を活用して、中心座標 (50, 50) まで移動する。
2. 青ゴール、黄色ゴールまでの距離と角度から、現在位置を推定する。
3. apub, vpub の値から移動距離と角度を推定する。その結果を累積してデッドレコニングを実現する。
4. (2),(3) の機能を組み合わせて、GPS, ジャイロセンサの情報を活用しないで、中心座標 (50, 50) まで移動する。
5. 障害物が 1 個以上ある状況で、環境地図を作成して出力する。

最終的に、センサで計測した結果を用いて環境 world2 の環境地図を作成し、自己位置を推定するロボットの行動を実装する。その後中心座標 (50,50) に戻る。

4.3.7 結果の評価

上記 1 から 5 の課題において、クライアントがミッションを達成したと判断して、中心座標 (50,50) に戻って、速度、角速度を 0 と指定した時点で、サーバは実行時間を確定する。

中心座標 (50,50) から距離１．０（※暫定値）以内に静止することで戻ったと判定する。開始から１２０秒（※暫定値）経過して戻っていない場合、その回のミッションは終了する。

作成した環境地図は、プログラム pgmcmp で正解したピクセル数を数える。

（使い方）全体をビルドすると実行ファイル pgmcmp ができる。先に用意されていた正解のファイルとプログラムが出力したファイルをそれぞれ file1.pgm, file2.pgm とすると、次のコマンドで正解したピクセル数がわかる。

```
$ pgmcmp file1.pgm file2.pgm
```

（１）から（４）の課題では、次の (１) 式でポイントを求める。

（１２０ － 実行にかかった時間（秒）） × 100 　　－ － － **（１）**

（５）の課題では、次の (２) 式でポイントを求める。

（１）式の値 ＋ 作成した環境地図の正解のピクセル数 　　－ － － **（２）**

なおエネルギーは今回評価に含めないことになった。よってエネルギーを消費してもなるべく早くゴールに到達するのがよい。

4.3.8 レポートについて

レポートは、上記課題 1 〜 5 で、完成したものについて、

- 実現しようとした機能を箇条書きで表す。
- 「自分が書いた部分」のソースリスト
- プログラムの説明
- 結果
- 結果についての考察、感想

を含むように、なんらかの文書作成ソフトを使用して作成し、PDF ファイルを作成する。ファイル名は er?????d.pdf とする。

自分が書いていない部分も含めてプログラム全体を er?????d.cpp とする。課題毎に個別のプログラムを作成した場合は、次のように学籍番号の次のアルファベットを順に変えて命名する。

```
er?????d.cpp      1 のソースファイル
er?????e.cpp      2 のソースファイル
er?????f.cpp      3 のソースファイル
...
```

レポートの PDF ファイル 1 個、ソースリストのファイルを Google Classroom のこの科目のクラス、ロボットインテリジェンスの課題提出のページから提出する。提出期限は 202?年 ?? 月 ?? 日 23:59。

4.3.9 既知のバグ、未実装の機能

（済）と付いている項目は修正済を表す。

- **（済）センサで計測した結果が、本来検出されるべき位置ではなく、** 物体の反対側の辺になったり、物体の向こうの壁になったりする。
- （済）ロボットと障害物の衝突判定をしていない。
- （済）乱数の初期値の設定が配置に反映されない。
- （済）障害物の位置、大きさ、個数が固定されている。
- （済）ロボットや障害物は初期配置時にお互いに重ならないようにする。
- （済）ロボットの座標、角度、検出点の座標を送信するか否かが指定できない。
- （済）ロボットや障害物は初期配置時に中心座標 (50,50) を空けておく必要がある。
- （済）作成した環境地図の出力ができない。

- **（済）ミッション達成時の総合評価において、実行にかかった時間、エネルギーの** 残り、環境地図の精度をどのような重み付けで加算するかが決まっていない。
- （済）実行時間を計測する仕組みがない。
- （済）ミッション終了の判定の仕組みがない。
- （済かも）ある状況（起動直後？）で、センサで検出した結果の値が正しくない。
- （済かも）動作の指示に対して、全く動いていないかのような値が送られてくる。
- 障害物の形状が四角形に固定されている。
- ロボットがまれに障害物の中に入ってしまう。

4.3.10 その他

- 障害物の初期配置時にロボットや他の障害物と重なる場合は再配置を試みるが、累積で 10000 回失敗すると配置不可能と判断して強制終了する。（仕様）

4.4 まとめ

この章の目標

- 計画行動が何を表しているかわかる、説明できる。

練習問題

1. 計画行動とはどのような行動か、簡単に説明しなさい。

第 5 章

適応行動

適応行動 とは環境や外部からの入力（センサ情報や刺激等）が変化する場合に、それぞれの環境や入力に対して望ましい行動を実現することである。望ましい行動を事前にすべて指定することはできないので、ロボットがなんらかの仕組みで望ましい行動を生成、記憶する「学習」が取り入れられる。反応行動が常に一定であったのに対して、適応行動では入力の変化に合わせて行動を変えることができるので、柔軟により良い行動を実現できる。

5.1 強化学習

強化学習とは、ロボットに実現させたい行動（を適切に選択するための基準）を、多くの試行を通して獲得する学習方法の一つである。強化学習では、学習の一連の過程で様々な概念やパラメータ、計算法が現れる。以下ではそれらを順に示す。

- 環境モデル

強化学習では、ロボットと、ロボット以外のすべてを表す環境の、２種類のものの間の相互作用により状態が変化していくと考える。環境からはセンサ等により検知した情報がロボットに与えられ、ロボットは環境から得た情報に基づき行動を選択し、実行する。注意する点として、ロボットが得られない情報は環境に属すると考える点である。例としてロボットアームの関節を１０度回転させる行動を選択して実行したとする。実際には、モータの回転角度制御の精度から９度になってしまったり、検知していなかった障害物に衝突して５度で止まってしまうことがあり得る。よって、ロボットは、モータを含めたハードウェア全体ではなく、行動を選択するソフトウェアのみをロボットと見なす。そしてモータは不確定な要素を含んだ環境の一部として扱う。これに付随して、強化学習では、モータや車輪の回転、行動によって生じた状態の変化などは不確定性を含んだ確率的な要素として扱われる。

- 時間変化

時間変化は離散的に起こる。時刻 $t, t+1, t+2, ...$ のように表現される。

- 状態空間

環境が取り得る状態は、有限個数に離散化されて表現される状態の集合として表現する。$S = (s1, s2, s3, ..., sm)$

個々の状態をどのように定義するかは決まっていない。細かく場合分けして多くの状態を定義することが可能であ

るし、反対に大まかに場合分けして少数の状態とすることもできる。状態を多く定義することが必ずしも良い結果に繋がるわけではない。

- 行動空間

ロボットが選択できる行動は、有限個数に離散化されて表現される行動の集合として表現する。$A = (a1, a2, a3, ..., an)$

状態空間と同様に、個々の行動をどのように定義するかは決まっていない。行動を多く定義することが必ずしも良い結果に繋がるわけではない。

- マルコフ性、マルコフ決定過程

上記に示した離散化された有限個の要素の集合からなる状態空間、行動空間からなる環境での状態変化は **マルコフ性** を持つ。マルコフ性とは、現在の状態から次の状態への遷移が、過去の状態や状態遷移の影響を受けないことを表す。またマルコフ性を持つ環境での状態遷移の過程は **マルコフ決定過程** と呼ばれる。例として、サッカーロボットがシュート可能な位置にいるという状態を考える。そこに到達するまでに、前進してきたのか、後退してきたのか、旋回してきたのか、速度が速かったか遅かったかという過程は関係なく、次の状態変化には現在の位置、角度、速度、加速度等の現在の状態のみが関係するということである。環境をマルコフ決定過程の条件を満たすようにモデル化することは、強化学習を行う上で必要な条件となる。

- 制御方策

ある状態にいる時に、どの行動を選択するか決める基準があって、その基準に沿って行動を決めること。学習前は基準がないが、学習が進むに連れて各状態毎に適切な行動を選択できる **方策** を獲得することが学習の目的である。方策は f で表す。状態 s において行動 a は方策 f により選択される。 $a = f(s)$ と表す。

- 報酬、報酬関数

ロボットは望ましい状態に達すると **報酬** が与えられる。状態と報酬の対応付けはある関数により規定されるとして扱われるが、現実の問題に適用する場合には、複雑な関数としても有効性はわからないので、最終的に到達して欲しい望ましい状態に 1，その他を 0 とする単純な対応でよい場合が多い。時刻 t で与えられる報酬を r(t) で表す。

- 強化学習の手順

1. ロボットは環境から情報を得て現在の状態を検知する。具体的にはセンサの値を読み取ったり、自分の位置、角度、速度等を得る。その結果、状態空間の中で現在の状態を認識する。環境をシミュレーションで実現している場合は、環境の方で状態を認識してそれをロボットに通知してもよい。

2. 現在の状態に基づいて、行動空間から選択可能な行動を選択し実行する。選択可能な状態と行動の組合せの個数は、最大でそれぞれの要素の個数の積となる。上記 S と A では $m \times n$ となる。状態によっては選択できない行動があり得る。

3. 環境はロボットの行動の結果次の状態に遷移する。一般にこの遷移は確率的に起こる。すなわち、同じ状況であっても検知できない不確定要素により次状態は異なる。シミュレーションでは遷移後の状態に応じて報酬 $r(t)$ をロボットに与える。学習の各回で与えられる報酬を即時報酬と呼ぶ。具体的には報酬の数値が通知される。

4. ロボットは受け取った報酬から積算報酬を求める。また報酬に基づいて、行動選択の方針を調整する。学習が収束していないと判断したら、時刻 t = t + 1 として 1 に戻る。

- 積算報酬

報酬は上記の手順を 1 回実行する毎に与えられる。学習の目的は、将来の報酬の合計（積算報酬）に関して、なるべく多くの報酬を得られるような制御方策を見つけることである。すなわち、個々の状態で望ましい行動を選択できるようになることである。学習が適切に行われると、瞬間的には報酬が低い場合があっても、その後の行動によって高い報酬が見込めるような、長期的に見て最も高い報酬が得られるような行動を選択するようになる。

時刻 t における積算報酬 return(t) に関しては、単純な合計とする代わりに、次の計算式を用いることが多い。r(t) はその回の即時報酬、 γ (Gamma) は減衰係数で 0.0 -- 1.0 とする。

$$return(t) = \sum_{n=0}^{\infty} \gamma^n r(t+n)$$

γ の意味は、将来得られる報酬をどの程度小さく見積もるかの度合い（割引率）を表す。状態遷移は確率的に起こるので、遠い未来になるほど環境変化やロボットの故障など、予期できない問題で報酬が得られなくなる可能性が高くなる。このリスクを現時点での評価に反映させていることになる。

- 価値関数

価値関数 V(f(s)) は f(s) によって得られる return の期待値を表す関数とする。学習の目的は価値関数で表すと、S 全体で価値関数 V(f(s)) を最大にする最適な f (これを fb とする) を求めることである。

$$V(fb(s)) = \max_{f} V(f(s)),\ \ \forall s \in S$$

環境モデルが離散時間、離散状態で表現されていると、fb(s) が必ず存在することがわかっている。

- 行動価値、Q(s,a)

ある方策 f が最適方策 fb であることの必要十分条件が次の式で表現される。

$$Q(s, f(s)) = \max_{a \in A} Q(s, a),\ \ \forall s \in S$$

Q(s, a) は行動価値、 **Q 値** と呼ばれ、状態 s で行動 a を選択した時の積算報酬 return の値を表す。よって各状態における Q 値がわかれば、大きな Q 値に対応する行動 a を選択することで大きな積算報酬が期待できる。次に示す **Q-学習** は逐次的な学習過程で Q 値を更新しながら正しい値に近づけ、それにより適切な行動を選択できるようにする学習手法である。

5.1.1 Q-学習, Q-learning

Q-学習では次の式に基づき Q 値を更新していく。学習が進むに連れて Q 値は正しい値に近づいていき、それに基づき行動を選択すると適切な行動が徐々に選択されるようになることである。Q-学習の特徴は、Q 値を用いて将来の報酬を予測して、その値を現在の Q 値に反映させられることである。

$$Q(s, a) = (1 - \eta)Q(s, a) + \eta(r(s) + \gamma \max_{a'} Q(s', a')),\ \ \forall a' \in A$$

各記号は次の意味を持つ。

- η : Eta, 学習率　0.1 程度にする
- γ : Gamma, 減衰係数 0.9 程度にする
- s' : 選択した行動 a により遷移した次の状態
- a' : s' で選択可能な行動

学習率 η は今までの Q 値と新しく学習した結果（右辺の後半）をどのような重み付けで新しい Q 値に反映させるかを決める。例として、$\eta = 0$ の場合は、新しい学習結果は反映されず、Q 値は旧値のままで更新はされない。$\eta = 1$ の場合は、今までの Q 値は破棄され、新しい結果のみが記録される。実際は 0.0 -- 1.0 の間で設定する。

減衰係数 γ は将来見込まれる行動価値を Q 値の更新に反映させる係数で、この項があることにより、将来の報酬を見越した長期的な行動の系列を獲得することができる。

Q 値は通常、状態数 x 行動数の要素を持つ 2 次元配列で表現、管理する。

- 学習の手順

以下に 1 ステップ Q 学習アルゴリズムの標準的な手順を示す。

1. Q 値の配列を初期化する。通常は全て 0 でよい。
2. 現在の状態 s に対して、方策 f により行動を選択、実行する。方策は後述する。
3. 環境から遷移後の状態、報酬を受け取ったら、それらの値を用いて上で示した Q 値の更新式に従い、更新する。
4. 方策 f を更新する。すなわち状態 s において最適な行動として最大 Q 値に対応する行動が選択されるように方策を調整する。
5. 各 Q 値が収束していない、および方策が収束していないと判断したら 2 に戻る。

行動選択の方策 f は、現在の状態においてどの行動を選択するかの基準を定める。以下の方策が考えられる。

1. 選択可能な行動の中からランダムに選ぶ。
2. 選択可能な行動の中から最も行動価値の高い行動を選ぶ。
3. その他

1. のランダムに選ぶ方策では、それまでの行動選択の結果報酬がどのように得られたかという経験が全く活用されず、学習ができない。

直感的には、2. のその時点で選択可能な行動の中から最も行動価値の高い（高い報酬が見込める）行動を選択するのが良いように思える。これは **greedy (貪欲) 方策** と呼ばれる。greedy 方策は学習が十分進んだ後で、各行動に対して見込める報酬が正しく推定されている場合は有効に機能する。しかし学習が進んでいない段階では、他の行動が選択される前にたまたま選択された行動で報酬が得られていると、他の行動を試すことなく常に同じ行動が選択されてしまい、より高い報酬が得られる行動を探索できない。

そこで、よく用いられるのが 1. と 2. の方策を組み合わせた ϵ **-greedy 方策** と呼ばれる手法である。これはある確率 ϵ（0.1 から 0.2 程度）でランダムに行動を選択し、それ以外では最も行動価値の高い（高い報酬が見込める）行動を選択する。多くの場合は確実に高い報酬が見込める行動を選択することで報酬の平均を高く保ち、その一方で新しい方策も一定の割合で探索することで、早い段階でたまたま報酬を得た行動に固定されるのを防ぐ効果がある。

5.2 計画行動に関する実習課題

課題の名称は ri3 とする。自分の前方にある物体の種類を検出可能な視覚センサを持つロボットの行動を規定する知能のプログラムを作成する。ここで目指すのは、フィールド内にあるボールとゴールを認識して、ゴールに向けてボールを蹴る行動を実現する戦略の実現である。

ROS 上のシミュレータ world3 内で行動するサッカーロボットの行動を規定するプログラム er14000k.cpp を作成する。行動を決める制御方策、ポリシーは強化学習により獲得する。ロボットの視覚センサは次の機能を備える。

- 視覚センサは 21 点の画素を持ち、前方の 20 度の範囲を一度に観察できる。
- 前方にある物体の種類が画素毎に分かる

詳細はこの後のインストールと ri3 に関する説明文書を参照する。

5.3 ROS を使った実習課題 3 (ver.1.0.4, 20191110)

- パッケージのインストールは 「3.2 実習で使うソフトウェア開発キットのインストール」 を参照する。
- パッケージのファイルは http://edu.isc.chubu.ac.jp/naga/ri304.tgz をダウンロードする。
- %__PACKAGE は ri3
- %__SAMPLE は er14000k, er14000j
- ビルドや実行時にエラーがあって動かない場合は、「3.2 実習で使うソフトウェア開発キットのインストール」 のトラブルシューティングを参照する。

5.3.1 目的

仮想環境 world3 上で行動するロボットの知能を規定するクライアントプログラム er14000k を作成して、サッカーに関連した以下の行動

- 最初は（どの方向でもよいので）ボールを蹴る
- ボールを黄色ゴールにシュートする

などを実現する行動系列を強化学習で獲得する。

目的は、センサでゴールとボールを見つけてキックやシュートする行動を試行錯誤により自律的に獲得することである。

5.3.2 新機能、変更点

- 学習した Q 値をセーブ、ロードする機能が追加された。これにより、中断した状態から学習を再開できるようになった。
- （★重要な変更）world3 からは報酬ではなく、ゴールに入った、ボールに触ったという情報が通知され、報酬はクライアント側で決めるようになった。これにより目標とする様々な行動に対して独自に報酬を決められるようになった。
- ボールが壁に衝突すると速度が遅くなるようになった。
- ゴールしたこと、ロボットがボールに触ったことをサーバ側で画面に表示するようにした。
- ロボットが高速度でボールを蹴るとピンボールのようになってしまうので、ボールの速度の上限を 100.0 に制限し、壁の反発係数を 0.75 にした。

5.3.3 環境とロボットの仕様

- 環境には黄色のゴール (大きさは縦 20、横 1、中心座標 (0, 50)) と青色のゴール (大きさは縦 20、横 1、中心座標 (100, 50)) がある。直径 5 のサッカーボールが 1 個ある。
- ロボットは前方に距離センサを 21 個備えており、前方正面から ±10 度の範囲にある物体までの距離と種別を 1 度刻みで計測することができる。
- ロボットがボールを蹴る（衝突する、接触する）と、ロボットの速度と方向をボールが継承し、転がる。時間が経過すると転がり抵抗により速度は遅くなる。
- ロボットやボールは壁に衝突すると跳ね返る。
- ボールが壁に衝突すると速度は遅くなる。
- ボールが黄色ゴールに衝突するとゴールに入ったと認識され、クライアントに通知される。
- ロボットがボールを蹴る（触れる、衝突する）と、クライアントに通知される。
- （★重要な変更）クライアントでは、通知されるゴールやボールを蹴った情報から、強化学習で使用する報酬を独自に定めることができる。
- なるべく短時間でシュートを決められるようになることを目指す。

5.3.4 World3 の機能

- 座標は縦横１００，単位はない
- 左上が原点
- センサで検出した物体の座標は紫色で表示される。
- 環境からロボットに向けて、１秒間に２０回、検出した座標やものの種別の情報、ゴールやボールとの接触の情報が伝えられる。
- 起動時のオプションは次のとおり

```
$ rosrun ri3 world3 [-fxyon/fxyoff] [-gpson/gpsoff] [-gyroon/gyrooff]
  [-r 整数] [-s 実数] [-t 秒数] [-xyo X 座標 Y 座標 方向]
```

- -fxyon/fxyoff は距離センサで検出した座標の送信の可否を指定する。デフォルト値は fxyoff で送信しない。
- -gpson/gpsoff は GPS で検出したロボットの座標の送信の可否を指定する。デフォルト値は gpsoff で送信しない。
- -gyroon/gyrooff はジャイロセンサで検出した角度の送信の可否を指定する。デフォルト値は gyrooff で送信しない。
- -r は乱数の種を指定する、0 を指定すると時刻を乱数の種として毎回異なる乱数が得られる。デフォルト値は 0。乱数の種を同一の値にするとロボットや障害物を同じ配置にすることができるので、再現性のあるシミュレーションが可能。
- -s はセンサの計測可能範囲、デフォルト値は 142 で, 対角線も届き、直線上で見えているものは壁を含めてすべての物体を計測可能。
- -t は実行時間（秒)、デフォルト値は 120
- -xyo はロボットの初期座標と向き、このオプションの後に 3 個の実数を指定する。特定の位置から再現性のあるシミュレーションが可能。

実行例は以下のとおり。

```
$ rosrun ri3 world3 -fxyoff -gyrooff // 距離センサとジャイロ（と GPS も）は OFF
$ rosrun ri3 world3 -gpson // GPS は ON（距離センサとジャイロは OFF)
$ rosrun ri3 world3 -r 123    // 乱数の種が 123
$ rosrun ri3 world3 -s 10     // センサの探索範囲が 10
$ rosrun ri3 world3 -r 456 -s 20 -t 60   // オプションは同時に複数指定可能
$ rosrun ri3 world3 -xyo 10 10 315 // 座標 (10,10)、３１５度の方向に配置する
```

5.3.5 er14000k の機能

- ロボットは伝えられた情報をもとに、次の行動を決めて環境に返信する。
- １回の返信では、速度の変更と回転速度の変更を各１回のみ送信できる。
- 起動時のオプションは次のとおり

```
$ rosrun ri3 er14000k [-lq filename] [-p 秒数] [-sq]
```

- -lq は Q 値の表を記録してあるファイルから読み込ませることができる。ファイルは事前に-sq オプションで記録しておく必要がある。デフォルト値は OFF で読み込まない。
- -p は -sq オプションで Q 値を記録する場合に何秒間隔で記録するかを指定する。デフォルト値は 600 で 10 分毎に記録する。
- -sq は Q 値の表をファイルに記録する。ファイル名は Q から始まる名前が自動的に付けられる。記録される場所はワークスペースである。デフォルト値は OFF で記録しない。

実行例は以下のとおり。

world3 を起動しておいて

```
$ rosrun ri3 er14000k  -sq -p 60
```

とすると 60 秒毎に Q 値の表がファイルに記録される。記録されたファイルが例えば Q20170108200000.dat とすると er14000k を終了して次回起動する時に

```
$ rosrun ri3 er14000k  -lq Q20170108200000.dat
```

とすると記録された Q 値が読み込まれ、そこから学習が行われる。

```
$ rosrun ri3 er14000k  -lq Q20170108200000.dat -sq
```

のようにロード、セーブを両方指定することも可能。

5.3.6 サンプルプログラムの動かし方

インストールマニュアルにある手順に従って、インストールとビルドができると以下のサンプルプログラムを動かすことができる。

サンプルプログラムを自分の学籍番号のファイル名でコピー、編集する。

```
$ cp er14000k.cpp er1????k.cpp // 強化学習
$ cp er14000j.cpp er1????j.cpp // 通常のプログラム
```

(次のページに続く)

(前のページからの続き)

```
$ vim er1????k.cpp
$ vim er1????j.cpp
```

以下のように起動する。

```
1980  $ roscore
1981  $ rosrun ri3 world3
1982  $ rosrun ri3 er14000k
```

er14000j は内部でロボットやボールの位置、方向などを参照するので、world3 の起動時には以下のオプションが必要になる。

```
1983  $ rosrun ri3 world3 -fxyon -gpson -gyroon
1984  $ rosrun ri3 er14000j
```

5.3.7 プログラムの説明

- ri1 のゾウリムシでは、計測、通信、移動の順で処理されていたが、今回は移動、計測、通信の順で処理される。この変更により通信の内容と現在位置のずれがなくなる。
- 実行ファイル world3 は Window の管理、描画、ロボットの管理、ボールの管理、er14000k への情報の送信を行う。
- 実行ファイル er14000k はロボットの知能部分だけを実現し、速度と角速度を world3 に送信する。どのように行動（速度と角速度）を決めるかは強化学習で獲得する。強化学習の計算処理は er14000k に組み込まれている。

er14000k.cpp において、100 行目付近、Write your intelligence here. の前後の部分がサンプルの行動を規定している。

- 変数が表す内容

```
（入力）
x はロボットの x 座標　（GPS に相当、※設定により有無を決める）
y はロボットの y 座標　（GPS に相当、※設定により有無を決める）
orient はロボットの向いている方向 （ジャイロに相当、※設定により有無を決める）
v はロボットの速度
a はロボットの角速度
life はロボットのエネルギー
numfinds はロボットが見つけたものの個数となっている。この課題では 1 か 0
fx[i] は見つけた座標 i 番目の x 座標　（※設定により有無を決める）
fy[i] は見つけた座標 i 番目の y 座標　（※設定により有無を決める）
fd[i] は見つけた座標 i 番目までの距離
fk[i] は見つけたもの i 番目の種類（壁，青ゴール，黄ゴール，障害物）を表す。
```

(次のページに続く)

(前のページからの続き)

```
goal は環境から与えられるゴールに関する情報、（1：ゴール、0：ゴールしていない）を表す。
balltouch は環境から与えられるボールに関する情報、（1：蹴った、0：蹴っていない）を表す。
```

```
（出力）
返信用の角速度、選択した行動により変わる
返信用の速度　、選択した行動により変わる
```

fk[i] の参照時など、種別の数値は KINDBALL など、なるべく以下の「名前」を使うこと。数値を直接書かない。数値は場合によって変更される可能性があるため。

```
#define KINDWALL        0     // 壁
#define KINDYELLOWGOAL  1     // 黄色ゴール
#define KINDBLUEGOAL    2     // 青ゴール
#define KINDBALL        6     // ボール
```

5.3.8 強化学習 er14000k

bp, glp, grp はボール、ゴールが見えた座標、初期値は存在しない値にしてある

```
static unsigned long timing = 0;   // timing control, for display & filesave
timing++;

int bp = -1, glp = -1, grp = -1; // ball, goal left, goal right position
```

Q 値に関する変数、Q 値の表は 2 次元配列

```
float Qmax;
static float Q[NSTATE][NACT] = {{0, 0, 0, 0, 0, 0}, // 0 から始めると学習の
                                {0, 0, 0, 0, 0, 0}, // 効果がわかりやすい
                                {0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0},
                                {0, 0, 0, 0, 0, 0}};
```

```
////////// 記録した Q 値を読み込んで再開する場合はここで読み込む
if (ON == lqflag && (char)0 != qfname[0]) {
    loadqtable(Q, qfname);
    lqflag = OFF; // 一度読み込んだらその後読まないように
    ROS_INFO("Load Qtable from %s", qfname);
}

static float gamma = 0.9; // Q 学習のパラメータ
static float eta = 0.2; // Q 学習のパラメータ
static float reward = 0.0; // 報酬
```

(次のページに続く)

(前のページからの続き)

```
static int state = -1; // 状態
static int ls = -1; // 前回の状態
static int action = -1; // 行動
static int la = -1; // 前回の行動

ls = state; // 前回の状態を記憶
la = action; // 前回の行動を記憶
```

カメラからの情報の処理、何が見えているか

```
////////// まずボールやゴールの見え方を調べる
for (int i = 0; i < 21; i++) { // ボールが見えているかチェック
  if (KINDBALL == fk[i]) {bp = i; break;}
}
for (int i = 0; i < 21; i++) { // 右にゴールが見えているかチェック
  if (KINDYELLOWGOAL == fk[i]) {grp = i; break;}
}
for (int i = 0; i < 21; i++) { // 左にゴールが見えているかチェック
  if (KINDYELLOWGOAL == fk[20-i]) {glp = 20-i; break;}
}
```

物体の見え方から状態を決める

```
////////// 次にそれぞれの物体の見え方から状態を決める
if (-1 == bp && -1 == glp && -1 == grp ) state = STATENONE; // 0 何も見えない
if (-1 != bp && -1 == glp && -1 == grp ) state = STATEBALL; // 1 ボールが見える
if (-1 == bp &&(-1 != glp || -1 != grp)) state = STATEGOAL; // 2 ゴールが見える
if (-1 != bp &&(-1 != glp || -1 != grp)) {
  if (bp < grp) state = STATEBALLGOALLEFT; // 3 ボールの左にゴール
  if (glp < bp) state = STATEBALLGOALRIGHT; // 4 ボールの右にゴール
  if (grp < bp && bp < glp)
                state = STATEBALLGOALLR; // 5 ボールの左右にゴール
}
```

直前の行動の結果から報酬を決めて与える、その後 Q 値の更新

```
////////// 報酬を決める、現在はボールに触ったら（蹴ったら）報酬を与える
reward = 0.0;
//  if (1.0 == goal) reward = 1.0; // 報酬の設定、ゴールしたら 1.0
if (1.0 == balltouch) reward = 1.0; // 報酬の設定、ボールに触ったら 1.0
//  if (YOURORIGINALCONDITION) reward = 1.0; // 独自の報酬を設定することも可

////////// Q 値の更新
Qmax = 0.0;
for (int i = 0; i < NACT; i++) { // 現在の状態から最大 Q 値の行動を選択する
    if (Qmax < Q[state][i]) Qmax = Q[state][i];
}
```

(次のページに続く)

(前のページからの続き)

```
Q[ls][la] = (1.0 - eta) * Q[ls][la] + eta * (reward + gamma * Qmax);
```

以下で次の行動を決める、action に選択した行動の値が入る

```
////////// 次の行動を決める
if (rand() % 10 < 3) { // ある確率でランダムに行動を選択
    action = rand() % NACT;
}
else { // それ以外は Q 値が最大の行動を選択
  if (0.0 == Qmax) action = rand() % NACT; // 最大 Q 値が 0 ならランダム
  else {
      for (int i = 0; i < NACT; i++) {
          if (Qmax == Q[state][i]) action = i;
      }
  }
}
```

選択した行動から速度、角速度を決める。決めた値により次にロボットが動く。

```
////////// 選択した行動に対応する速度、角速度を設定する
if (ACTSTRAIGHT == action) { // 直進
    twist.linear.x = 10.0 * CYCLE;              // length/sec
    twist.angular.z = 0.0;                      // degree/sec
}
else if (ACTRIGHT5 == action) { // 右に 5 度曲がる
    twist.linear.x = 10.0 * CYCLE;              // length/sec
    twist.angular.z = -5.0 * PI / 180.0 * CYCLE; // degree/sec
}
else if (ACTRIGHT30 == action) { // 右に 30 度曲がる
    twist.linear.x = 0.0 * CYCLE;              // length/sec
    twist.angular.z = -30.0 * PI / 180.0 * CYCLE; // degree/sec
}
else if (ACTLEFT5 == action) { // 左に 5 度曲がる
    twist.linear.x = 10.0 * CYCLE;              // length/sec
    twist.angular.z = 5.0 * PI / 180.0 * CYCLE; // degree/sec
}
else if (ACTLEFT30 == action) { // 左に 30 度曲がる
    twist.linear.x = 0.0 * CYCLE;              // length/sec
    twist.angular.z = 45.0 * PI / 180.0 * CYCLE; // degree/sec
}
else if (ACTBACK == action) { // 後退
    twist.linear.x = -10.0 * CYCLE;             // length/sec
    twist.angular.z = 0.0;
}
```

5.3.9 通常のプログラム er14000j

通常のプログラムは、強化学習は行わず、状態に合わせて決められた行動を行う。

```
static float bx = 0.0, by = 0.0;
float tx = 0, ty = 0;
int tn = 0;

for (int i = 0; i < 21; i++) { // ボールが見えているかチェック
  if (KINDBALL == fk[i]) {tx += fx[i]; ty += fy[i]; tn++;}
}
if (1 < tn) {tx /= tn; ty /= tn;} // ボールの座標の平均
if (0.0 != tx && 0.0 != ty) { // tx,ty が 0 でなければ
    bx = tx;  // ボール位置をセット
    by = ty;
}
```

これより下が行動を決める部分

```
if (0.0 == bx && 0.0 == by) { // ボールが見えていなければ
    twist.linear.x = 0.0;
    twist.angular.z = 20 * PI / 180.0; // 回転して探す
}
else { // ボールが見えていれば
    float b2g = arctan(bx, by, 0.0, 50.0); // ボールからゴールへの方向
    float r2g = arctan( x,  y, 0.0, 50.0); // ロボットからゴールへの方向
    float r2b = arctan( x,  y,  bx,   by); // ロボットからボールへの方向
    float kpx, kpy;
    kpx = bx + std::cos(b2g + PI) * 5.0;   // ボールを蹴るための位置
    kpy = by + std::sin(b2g + PI) * 5.0 * -1.0;
    float r2kp = arctan(x, y, kpx, kpy); // ロボットから蹴る位置への方向

    if ((fabs(b2g - r2g) < (1.0 * PI / 180.0)) &&
        (fabs(kpx - x) + fabs(kpy - y) < 2.0)) { // 差が 1 度以内、後ろにいたら
        twist.linear.x = 10.0; // ボールを蹴りに行く
        twist.angular.z = (r2b - orient) * 1.0;
    }
    else { // 角度の差がそれ以上なら
      if (fabs(kpx - x) + fabs(kpy - y) < 2.0) { // 位置に来たら
          twist.angular.z = (r2b - orient) * 1.0; // 角度を合わせる
          twist.linear.x = 0.0;
      }
      else { // まだ来てなければ向かう
          twist.angular.z = (r2kp - orient);
          if ((kpx-x)*(kpx-x) + (kpy-y)*(kpy-y) < 10.0) {
                  twist.linear.x = 3.0; // 近ければゆっくり
          }
          else {
```

(次のページに続く)

(前のページからの続き)

```
                    twist.linear.x = 10.0; // 遠ければ速く
            }
        }
    }
}
```

5.3.10 課題

実現したい機能を箇条書きの項目で書き表し、個々の機能を順次プログラムで記述する。今回は状態空間、行動空間の要素の変更、追加をすることになる。サンプルプログラムではそれぞれ 6 種類しか定義していないので、もう少し細分化して増やすと結果が変わる可能性がある。ソースファイル名は er1????k.cpp とする。

1. 配布されたプログラムで、ボールを蹴る行動が学習により獲得できることを確認する。最初はなかなか蹴ることができないが、数分で成功することが多い。
2. 配布されたプログラムで、ゴールに向かってシュートする行動が獲得できるか確認する。
3. 1, 2 の行動を配布したプログラムより効率よく獲得させることを目指す。具体的には、
   - 状態空間の要素を変更、追加してみる。
   - 行動空間の要素を変更、追加してみる。

   を組合せる。
4. 独自に定めた行動を強化学習により獲得させることを目指す。具体的には、
   - 獲得させたい行動を定め、それに対して適切な報酬を定める。
   - 状態空間の要素を目的に合わせて変更、追加する。
   - 行動空間の要素を目的に合わせて変更、追加する。

   を組合せる。

番外編

0. er14000j.cpp を参考にシュートをする行動のプログラムを書いてみる。ソースファイル名は er1????j.cpp とする。

5.3.11 結果の評価

学習の初期と、ある程度学習が進んだ段階で、キックやシュートなどの獲得させたい行動を行った時間間隔の平均を求める。学習後に時間間隔が短くなっていれば、より高い確率で獲得させたい行動を実現できるようになったと言える。

5.3.12 レポートについて

レポートは、上記課題中で取り組んだものについて、

- 実現しようとした機能を箇条書きで表す。状態空間や行動空間を変更した場合はその内容がわかるように書く。
- 「自分が書いた部分」のソースリスト
- プログラムの説明
- 結果、学習前後で獲得させたい行動の平均時間間隔がどうなったか
- 結果についての考察、感想

を含むように、なんらかの文書作成ソフトを使用して作成し、PDF ファイルを作成する。ファイル名は er140??k.pdf とする。

状態空間や行動空間を変更して作成したプログラムがあれば、自分が書いていない部分も含めて作成したプログラム全体を

```
er1????k.cpp
er1????l.cpp
(er1????j.cpp)
...
```

というファイル名で記録する。

レポートの PDF ファイル 1 個、作成したプログラムのソースリストのファイルを Google Classroom のこの科目のクラス、ロボットインテリジェンスの課題提出のページから提出する。提出期限は 202?年 ?? 月 ?? 日 23:59。

5.3.13 既知のバグ、未実装の機能

- 蹴られたボールが静止することが稀にある。ロボットの速度が 0 の時に接触したためと考えられるが、正確な理由は不明である。

5.3.14 その他

- 現在特記事項は無い。

5.4 まとめ

この章の目標

- 適応行動が何を表しているかわかる、説明できる。

練習問題

1. 適応行動とはどのような行動か、簡単に説明しなさい。

第 6 章

ロボットシミュレータ上でのロボットモデルの作成と行動生成

6.1 まとめ

この章の目標

- ロボットモデルの作成法を理解して、実際に作成できる。
- 作成したロボットモデルを用いて、目的とする行動を生成できる。

練習問題

1. ロボットモデルの作成法を、簡単に説明しなさい。
2. 作成したロボットモデルの構造、機能を説明しなさい。
3. 目的とする行動をどのように生成したか説明しなさい。

第 7 章

協調行動

協調行動 とは複数台のロボットが協調、協力して全体の目標を達成するような行動を表す。単独の個体で取り組むよりも複数の個体が協調、または役割分担を決めて取り組んだ方が目標を達成しやすい場合に協調行動が必要となる。このレベルでは、自分と自分の周囲の環境（からの入力）、自分以外の個体が存在する。

なお協調行動の実現には、複数台のロボットが必要になるので、現在実習課題は設定していない。

7.1 まとめ

この章の目標

- 協調行動が何を表しているかわかる、説明できる。

練習問題

1. 協調行動とはどのような行動か、簡単に説明しなさい。

第 8 章

マークアップのサンプル レベル１

8.1 マークアップのサンプル　レベル２

8.1.1 サンプル　レベル３

装飾のサンプル

太字 、 **イタリック** 、

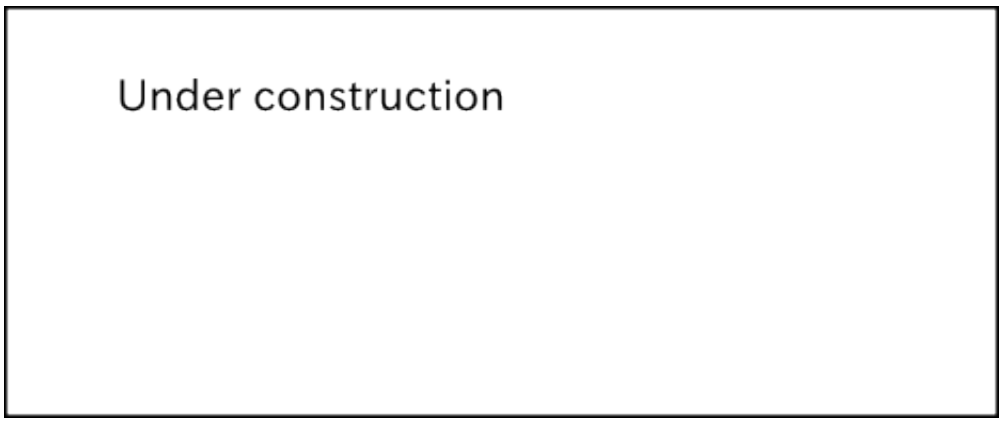

図 8.1　図のサンプル

本文中での参照は、図のサンプル 図 8.1 を御覧ください。

数式のサンプル $ABC + A\overline{B}C$

$$\overline{A}B\overline{C} + A\overline{B} + AC =$$
$$\overline{A}B\overline{C} + A\overline{B}(1) + A(1)C =$$

例題 3-2. 例題のサンプルを示しなさい。

答 3-2.　答のサンプル

リストのサンプル

箇条書き

- 順序回路の構成と動作原理が理解できる。
- ミーリー型順序回路とムーア型順序回路の違いがわかる。
- 状態遷移図、状態遷移表が理解できる、作成できる。
- 宇都宮市
- 那須塩原市
- 真岡市

数字付きリスト

1. まさし
2. みんみん
3. 夢餃子 (#を使うと、自動で数字が割り当てられます)

定義リスト

餃子　宇都宮の名物として有名。餃子の像もある。静岡の浜松がライバル。

ジャズ　宇都宮はジャズの町としても売り出し中。楽器メーカーを多数抱える静岡の浜松がライバル

焼きそば　知る人ぞ知る宇都宮の名物。専門店多数。なぜかビニール袋で持ち帰る。

表のサンプル

入力			出力					
A	0	1	A + 0	A + 1	A + A	A • 0	A • 1	A • A
0	0	1	0	1	0	0	0	0
1	0	1	1	1	1	0	1	1

これは **強調** のサンプルである。

例題 2-1. $(13)_{10}$ を 2 進数に変換しなさい。

答 2-1.

```
2 ) 13
  ------
2 )  6 ... 1
  ------
2 )  3 ... 0
  ------
     1 ... 1
```

13 [10] で上付文字になる。

コードサンプルの表示

リスト 8.1　2-2.cpp

```
#include <cstdio>
#include <cstdlib>

class a {
    int a;
} b;

int main(void)
{
    a c;

    printf("Hello world.\n");

    exit(EXIT_SUCCESS);
//  exit(EXIT_FAILURE);
}
```

リスト 8.2　2-2.cpp

```
// g++ 2-2.cpp

#include <cstdio>
#include <cstdlib>

int main(void)
{
    printf("Hello world. (2-2.cpp)\n");

    exit(EXIT_SUCCESS);
//  exit(EXIT_FAILURE);
}
```

第 9 章

Indices and tables

- genindex
- modindex
- search