2016 年度 卒業論文

方位情報を用いた自律ロボットの行為の失敗認識の効率化について

奈良女子大学 理学部 情報科学科 4 回生 新出研究室
13251472 小谷麻緒

平成 29 年 2 月

目 次

概 要

我々は、小型ロボットを用いて、目標物に到達する自律ロボットの研究を行っている。先行研究において、カメラによる物体追跡と目標物への移動能力を獲得した自律ロボットが実現された。しかし、このロボットは行動に失敗した場合にそのことを認識することができず、目標に到達することができなくなっていた。そこで我々はこのロボットに、従来定義されていなかった失敗の概念を加え、失敗後の行動パターンを追加した。また、ロボットの基本行為を改良し、ロボットが「目標物認識の失敗」を効率よく把握できるようにした。本論文では「目標物認識の失敗」の把握の効率化について述べる。

1 はじめに

1.1 研究背景

近年、自律的に目的を達成するロボットの実現が求められている。このようなロボットは一般的に、目標達成のために様々な能力を持つことが必要であり、その能力の一つとして目標物を認識してそこに到達するという行動がある。先行研究においてカメラによる物体追跡と目標物への移動能力を獲得した自律ロボットが実現された [1, 2]。しかし、実世界ではセンサーやモータの不正確さなどによって行為が正確に行えず、目的が達成できないことがあり、その場合には、行為に失敗したことを認識した上での行動をとる必要がある。先行研究でのロボットは行為に失敗した場合にそのことを認識することができず、目標に到達することができなくなっていた。そこで我々は、この先行研究における自律ロボットに、従来定義されていなかった失敗の概念を加えた。その際、従来のロボットで実現していた基本行動をそのまま失敗の認識に用いると、動作が冗長になり、「目標物認識の失敗」ではないにも関わらず、失敗という結果になってしまう可能性があった。そこで基本行動を改良し、方位情報を取得することで目標物再探索を行うことができるようにし、ロボットの「目標物認識の失敗」を効率よく把握するようにした。また、失敗後の行動パターンを追加した。

なお、本研究は、奈良女子大学理学部 4 回生石井、山本との共同研究である。失敗の概念の追加については [3]、失敗後の行動パターンについては [4] で述べる。

1.2 ロボットの行動

従来研究でのロボットの物体認識、物体追跡について述べる。物体認識が行われた際に、この物体検出の緑枠 (図 1) の左上の中心の x 座標 (カメラ画像の左端を 0 とする) が、カメラ画像の横幅に対して 30%未満のときは、ロボットがその位置から左回転を行うことで緑枠がカメラ画像の中心にくるようにする。反対に、緑枠の左上の中心の x 座標が、カメラ画像の横幅に対し

て 70%以上のときは、ロボットがその位置から右回転を行うことで緑枠がカメラ画像の中心にくるようにする。それ以外の場合は直進行動を行う。これらを繰り返し、目標物に接近する。物体追跡の結果、カメラ画像に対して物体検出の緑枠が 30%以上の大きさになった場合に、ロボットが目標物に十分に近づいたと判断する。表 1 は以上の物体認識、物体追跡をまとめたものである。

また、図 1 は実際にロボットが目標物探索を行い、その目標物に到達するまでを示したものであり、左はカメラ画像、右は接近行動、到達行動の様子を上から見たものである。この場合の目標物は絵に描かれた女性である。ロボットはカメラ画像の左側で目標物を認識したため、左回転と接近行動を行い、目標物に到達する。

カメラ画像における緑枠の場所	ロボットの行動
～30%	左回転
30%～70%	直進
70%	右回転

表 1: 物体認識、物体追跡

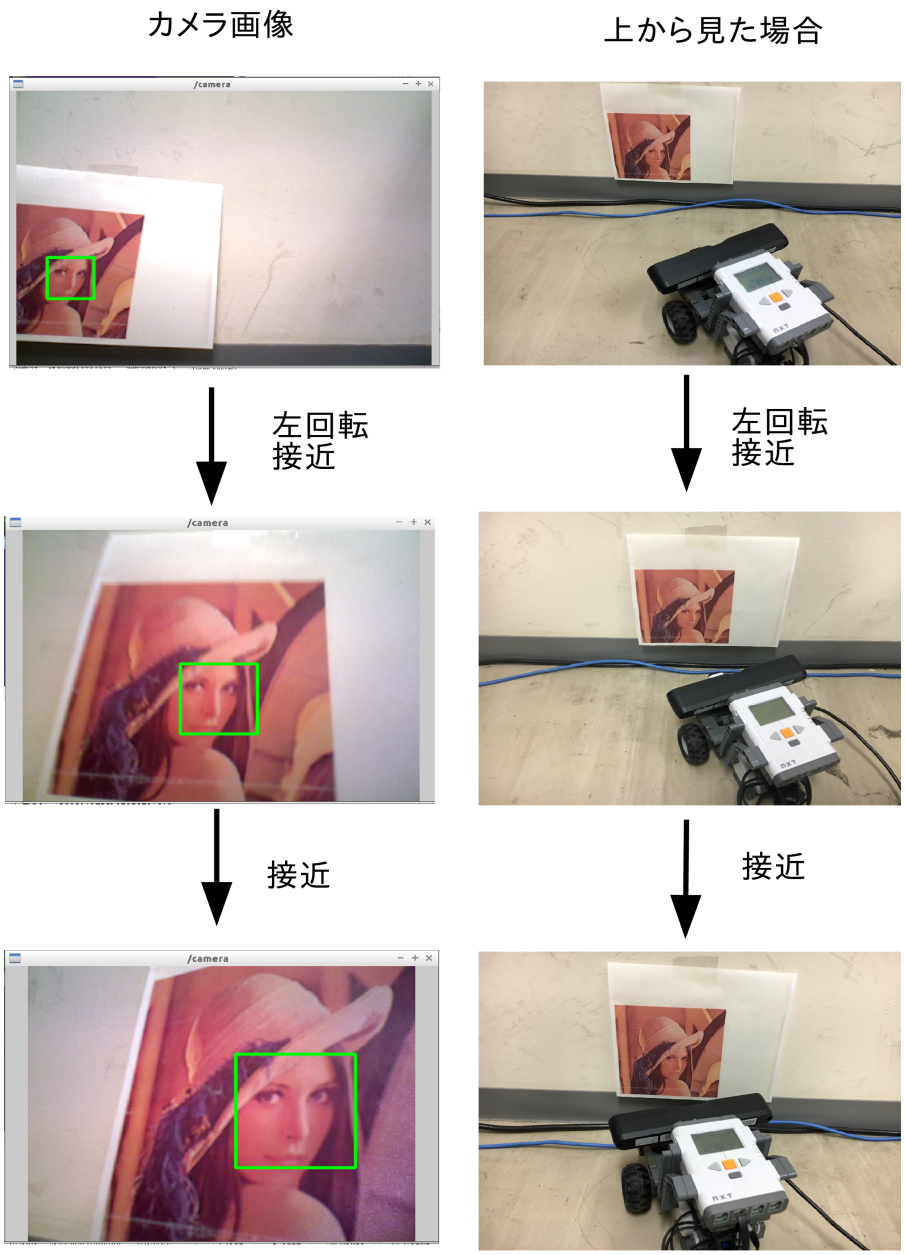


図 1: 到達行動

2　コンパスセンサー

2.1　コンパスセンサーとは

本研究では、ロボットが目標物探索行為に失敗したことの認識の効率化を行った。そのためには自身の向きを把握し、より多様な動きができるようにすることが必要である。そこでコンパスセンサーを使用した [5, p.3]。地磁気を読み取って方角を検出するセンサーであり、0 から 179 までの整数の値を出力する。実際のロボットの向きは 0 度〜360 度なので、コンパスセンサーの出力はその 1/2 の値である。

図 2: コンパスセンサー

2.2　実行

実際にコンパスセンサーを使用する場面は、ロボットを回転させる場合や向きが一定の範囲に入っているかを調べる場合である。図 3 はロボットを上から見た様子であり、黄色の円がロボットを表す。いま、北をコンパスセンサーの値で 0 とする。ロボットが向きを時計回りに変えながら回転するとコンパスセンサーの値も増え、179 に達した後は 0 に戻って一周する。

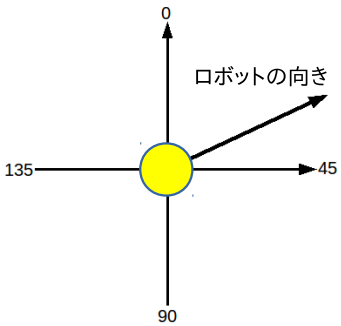


図 3: ロボットを上から見た図

3　実装

3.1　従来研究での問題点

従来の研究では、ロボットは目標物が見つからない場合、反時計回りに向きを変えながら目標物を探索していた。ロボットが一度目標物を認識すると、その目標物の方へ近づこうとする。しかし、画像認識の精度が悪い状況下においては一度認識した目標物を見失ってしまうことがあった。その場合、再び反時計回りに回転することになるが、同じ方向に回り続けたのではもう 1 周しないと目標物を再発見できない可能性が高い。このように、従来のロボットの基本行動では、目標物の再認識に時間がかかり、一度目標物を認識したものの「目標物認識」の失敗となってしまう可能性が高い。それを示したのが図 4 である。

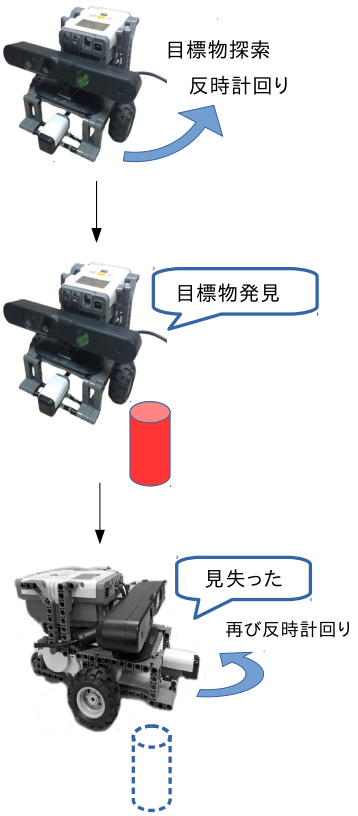


図 4: 問題点

3.2　改善策

ロボットが目標物を一度認識した後、その目標物を見失ってしまった場合、ロボットが一度目標物を認識した位置周辺を把握しておくことができれば、そのあたりを探索することで再び目標物に到達できる可能性が高いと考えられる。その際、目標物を見失ってしまった後の行動をただ反時計回りのみにするのではなく、場合分けをすることで再発見の効率を上げることができる。そこでロボットが目標物を見失った後の行動の改善を二点行った。

3.2.1　改善策１：再探索時の回転方向の適切な選択

改善策一点目は、ロボットが目標物を見失った後、以前目標物を認識した角度に戻るように行動するという方法である。ロボットが目標物を見失った際、目標物がロボットから見てどのあたりにあったかによって、ロボットの目標物再探索の行動は変わる。よって、目標物を見失うまでの間にロボットがとっていた行動によって場合分けを行った。それを示したのが図５である。例えば、ロボットが目標物を認識し、その目標物に近づこうと時計回りを行っていた際に、目標物を見失ってしまった場合は、その周辺に近づこうと反時計回りをする。この様子を左側の図で示す。同様に、ロボットが目標物を認識し、その目標物に近づこうと反時計回りを行っていた際に、目標物を見失ってしまった場合は、その周辺に近づこうと時計回りをする。この様子を右側の図で示す。

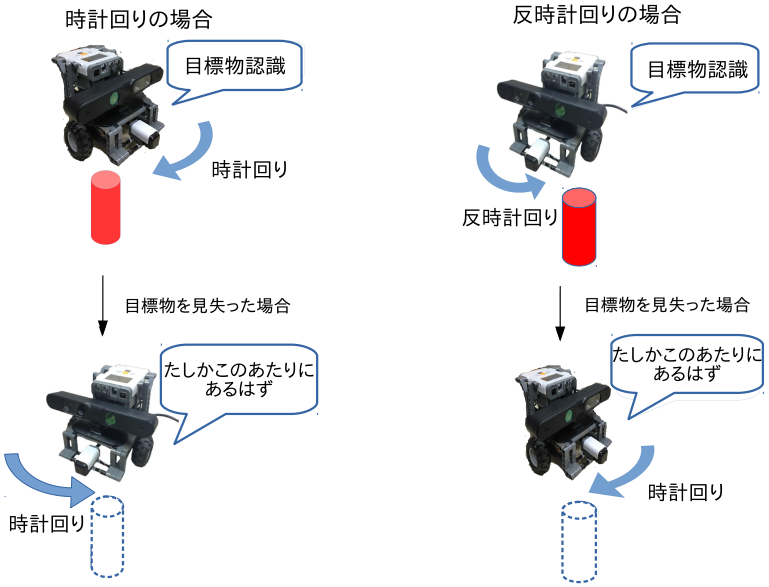


図 5: ロボットの目標物への近づき方

3.2.2　改善策２：再探索開始時の向きの調整

方針　改善策二点目は、2.1 のコンパスセンサーを使用し、ロボットが目標物を見失ってから、90 度回転させて再探索を行うという方法である。改善策１よりも、見失った目標物の位置周辺の探索により重点を置くことができるので、目標物再発見の効率を上げることができる。90 度は経験上、目標物再発見に適していると考えた値である。図 6 は、ロボットを上から見たときの図である。例えば目標物を認識し、反時計回りに目標物に近づいている際に目標物を見失ってしまった場合、その目標物があると考えられる場所の周辺を中心に探索することを考えた。そのため、90 度右回転を行い、回転した後の位置から再び反時計回りで向きを変えながら探索することで、図 6 の下部のようにピンクの楕円で表されている目標物の周辺を注意深く探索することができる。

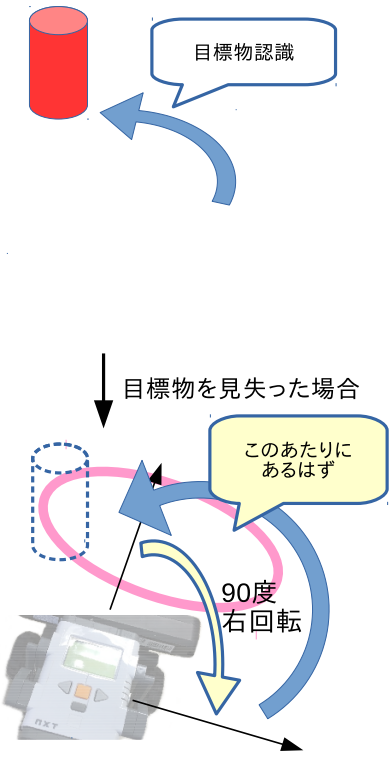


図 6: 90 度右回転

使用した関数とその問題点の解決　改善策２の実装においては、先行研究において構築された NXT Python+[6] にある action.py 内の関数 turn_right_angle、turn_left_angle を改良した。図７で示されているとおり、action.py は、ロボットの基本行動を制御するメソッド群である basic_action.py の上位ファイルとなっており、ロボットのより高度な行動を制御するファイルである。これらのファイルは Python で記述されている。以下、実装にあたって実際に action.py 内の関数 turn_right_angle、turn_left_angle を使用しコンパスセンサーを動かした結果判明した問題二点と、その解決として改良した点について述べる。

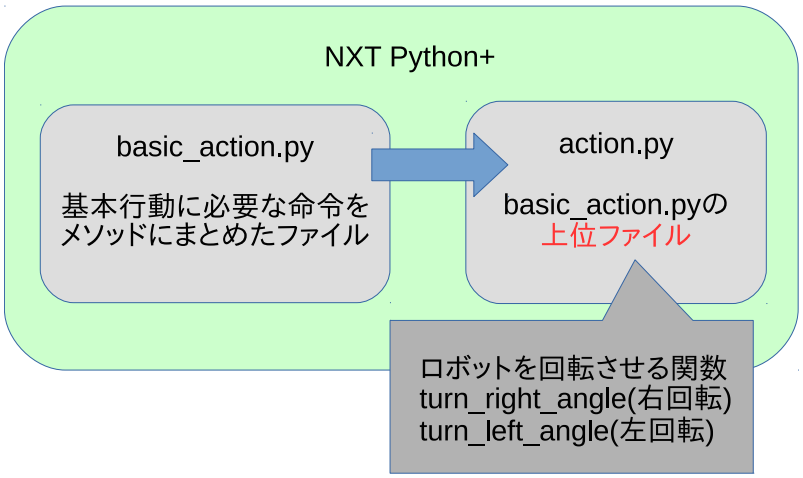
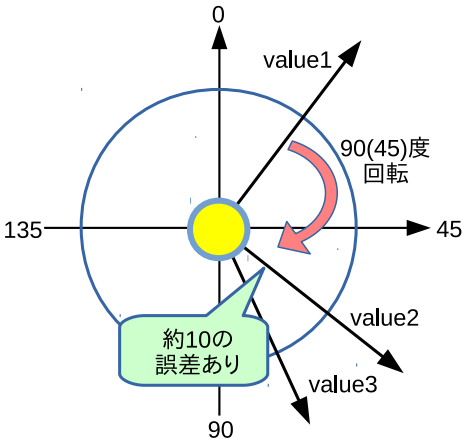


図 7: NXT Python+

問題点１：コンパスセンサーの認識速度　一つ目の問題点は、コンパスセンサーはロボットが回転している最中、その都度方位を返すことができないことである。原因として、ロボットの移動速度に対してコンパスセンサーの認識が遅いことが挙げられる。そこで、上記の関数に sleep 関数を組み込み、ロボットを少しずつ停止させながら回転させることで、コンパスセンサーの値を読み取るようにした。

問題点２：回転角度の大きな誤差　また、二つ目の問題点として、ロボットを指定の角度だけ回転させる時の目標位置と、実際の回転後の位置に大きな誤差が生じていたことが挙げられる。図 8 は、ロボットを上から見た時の図であり、黄色の円がロボットである。いま、ロボットを右に 90 度 (コンパスセンサーの値で表すと 45) 回転した場合を考える。ロボットが最初に向いている位置を value1 とする。ロボットは右に 90 度 (45) 回転するため、回転後の目標位置は value2 となる。しかし、実際にロボットを回転させると value3 まで回転してしまい、value2 と約１０の誤差が生じた。これはコンパスセン

サーの出力値での誤差であるので、実際には２倍の約２０度の誤差が生じていることとなる。これだけの誤差が生じた原因として、ロボットの回転中の動きが粗いため、十分に目的の角度に近づけないことが挙げられる。



value1	最初の位置
value2	90 度 (45) 回転後の目標位置
value3	実際に 90 度 (45) 回転した後の位置

表 2: 位置説明

図 8: ロボットを上から見た図

そこで、この問題を解決するために、関数 turn_right_angle、turn_left_angle 内のプログラムを二点修正した。まず、これらの関数においてロボットの回転を終了させるかどうかの判断の部分を修正した。turn_right_angle(ロボットが右回転する関数) を例にして図 9 で示す。compass_angle_right_of(value2,value3) は value3 が value2 から見て右半分にあるかどうかを調べる関数であり、修正前は if not compass_angle_right_of(value2,value3) であったが、not 文を除去することで、value3 が value2 を越えるとロボットが停止するという処理にした。

修正前

```
# value2を越えたら停止
#if value3 >= value2:
if not compass_angle_right_of(value2, value3):
   self.stop()
   break
```

修正後

```
# value2を越えたら停止
#if value3 >= value2:
if compass_angle_right_of(value2, value3):
   self.stop()
   break
```

図 9: turn_right_angle における改良

二つ目に、ロボットの回転後に位置を微調整する adjust_heading 関数についても修正を行った。今、上述の関数の修正により、ロボットが右回転を行うと value2 より value3 が右側に来るようになっている。そこで、この状態から value3 が value2 に近づくように左回転を少しずつ行うようにした。図 10 で示す。turn_left_angle についてもこれらと同様の修正を行った。

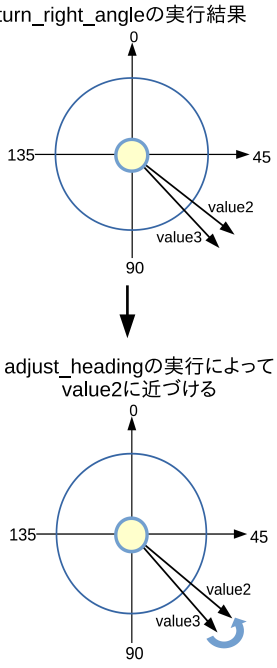


図 10: adjust_heading 関数の改良

3.3　実行結果

まず、改善策 1 の実行結果について述べる。以前目標物を認識した角度に戻るように行動するという方法を行うことにより、一度目標物の認識を行ったものの画像認識の結果によって目標物を見失ってしまった場合においても、すばやく再探索することができるようになり目標物再発見の可能性を上げることができた。

また、ロボットが失敗を効率よく認識するために改善策 2 を行った際、コンパスセンサーを使用する上で生じた二つの問題点を解決した結果について述べる。ロボットを左右にそれぞれ 5 回ずつ 90 度回転させた時の value2、value3 の値を出力したものと、それらの値の差を表 3 で示す。左回転時の value2 と value3 の誤差は 2、右回転時の value2 と value3 の誤差は 2.2 となった。これらは実際の角度にすると約 4 度となるので、ロボットが動作を続行するのにあたって十分な精度であるといえる。改善策 2 を行うことにより、一度認識した目標物があると考えられる範囲を重点的に探索できるようになったため、目標物再発見の可能性を上げることができた。

	左回転					右回転				
回数	1	2	3	4	5	1	2	3	4	5
value2	36	48	103	142	166	24	45	110	136	177
value3	38	51	104	142	170	26	50	112	138	177
誤差	2	3	1	0	4	2	5	2	2	0

表 3: 左回転、右回転した場合の誤差

4　まとめ

action.py 内の関数を改善したことで、ロボットがコンパスセンサーを使用して方位情報取得をするようになり、正確に回転できるようになった。また、ロボットの基本行為を改良することで、物体認識の精度が悪い状況下においても従来より効率よく再探索を行えるようになった。そのため、結果的に目標物の再認識に失敗した場合、そのことが従来より早く把握できることになり、すなわち、「目標物認識の失敗」をロボットが効率よく把握することができるようになった。

今後の課題を述べる。実世界において、より対応できるように様々な環境を想定することが必要である。たとえば、ロボットが走行中に小石に当たり倒れてしまった場合や、目標物への最短の道が閉ざされている場合にどのような行動をとるのが望ましいかといったことである。また、より緻密で高度な物体認識、物体追跡の開発が望まれる。

5　謝辞

本研究の遂行および本論文の執筆にあたり、いつも親身にご指導して下さった新出尚之准教授に深く感謝し、厚く御礼申し上げます。また、樽井先輩、兼松先輩をはじめ、新出研究室の皆様に感謝の意を表します。ありがとうございました。

参考文献

[1] 柿原怜奈. 自律ロボットの実現に向けた ROS による環境情報の取得. 2015 年度卒業論文, 奈良女子大学理学部情報科学科, 2016.

[2] 兼松明未. ROS によるロボットの目標地点への到達行動の実現について. 2015 年度卒業論文, 奈良女子大学理学部情報科学科, 2016.

[3] 山本千尋. 実世界における行為の失敗の概念を考慮した自律ロボットの実装について. 2016 年度卒業論文, 奈良女子大学理学部情報科学科, 2017.

[4] 石井あすか. 自律ロボットの目的達成における行動の柔軟化について. 2016 年度卒業論文, 奈良女子大学理学部情報科学科, 2017.

[5] HiTechnic 社のセンサ. `http://www.afrel.co.jp/cms/wp-content/uploads/2013/08/tech_hitech_091006.pdf`.

[6] 小島侑子. 小型ロボット制御のための汎用ライブラリの構築. 2011 年度修士論文, 奈良女子大学大学院人間文化研究科, 2012.