

飛行ロボットに搭載した360度カメラによる自己位置認識

Self-location recognition by a 360-degree camera mounted on a flying-robot

大谷 元気^{*1} 本田 泰^{*2}
Genki Ohtani Yasushi Honda

^{*1}室蘭工業大学 工学研究科博士前期課程 情報電子工学系専攻
Muroran Institute of Technology Faculty of Engineering Division of Information and Electronic Engineering

^{*2}室蘭工業大学 しゅくみ情報系領域
College of Information and Systems, Muroran Institute of Technology

In this research, we try to derive the three-dimensional coordinates and orientation of a flying robot equipped with a 360-degree camera. There are various means such as GPS, direction sensor, etc. as means for knowing the coordinates and orientation. However, it is impossible to know exact orientation and coordinates in places where indoors or magnetic fields are unstable. In this research, we aimed to acquire the direction and coordinates of the flying robot using a 360 degree camera. As a concrete method, we installed two color objects at the place we wanted to be a reference, aimed at achieving the purpose by acquiring the relative coordinates of the color object and the flying robot. Currently the average of the errors of x, y and z coordinates is less than 400 mm, but the average of the error of the direction is about 30 degrees, and in the place where the largest error is most, it is close to 120 degrees.

1. はじめに

近年、ドローンと呼ばれる無人航空機は人々の生活に馴染み深いものとなっている。写真や動画撮影に用いられるだけではなく、農薬の散布や宅配など様々な分野での活躍が期待されている。

一方で、その安全性が懸念されている。地上で動作するロボットとは違い、空中で動作するため落下による事故の危険性も高い。そして、操縦の難しさによるヒューマンエラーも少なくない。そのため、ドローンは環境の変化に適応する自律行動を取ることが求められている。

本研究で扱う飛行ロボットは4回転翼自律飛行ロボット [1](図1)である。この飛行ロボットは、知的行動の一つである反応行動 [2] によって飛行する。人間による操作は行わず自律的に飛行することから、ドローンと区別するために飛行ロボットと呼ぶ。

飛行ロボットは自律飛行するため、自分の位置や向きを把握する必要がある。ロボットの座標や向きを知る手段は様々な存在する。GPS(Global-Positioning-System:全地球測位システム)や方位センサーなどの小型で安価なものが多数市販されている。

また、複数のドローン間で直接通信を行い、衝突などの危険を回避し安全性を高める研究も行われている [3]。これらの研究に置いて、GPSなどの外部からドローンの位置情報を得ることを前提としている。

しかし、GPSを受信できないような場所や磁場の安定しないような場所では座標や向きを知ることはできない場合も存在する。そこで我々は、飛行ロボット自身に搭載したカメラによる画像情報を使い、座標と向きを取得できないかと考えた。

本研究の目的は、飛行ロボットに搭載したカメラによる色認識でカラーオブジェクトと飛行ロボットの3次元座標と相対的

な向きを取得することである。基準としたい点の床に2つのカラーオブジェクトを配置することで、飛行ロボットの座標と向きの認識を目指した。目的達成の手法として、順伝播型ニューラルネットワークの教師付き学習を用いた。

モーションキャプチャの位置情報データと比較することにより、飛行ロボット自身に搭載したカメラ情報から、自身の3次元座標と向きを認識することが可能となった。また、飛行ロボットに搭載可能なボードコンピュータを用いて、飛行に応用可能な速度でそれらを認識可能であることを示した。

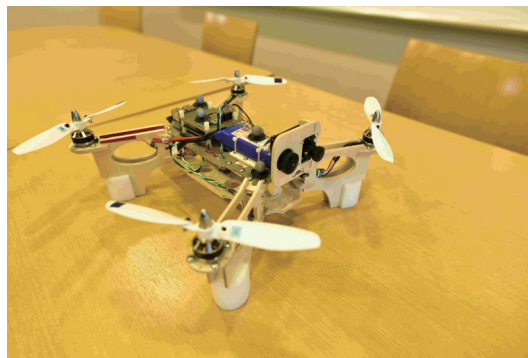


図1: 実験に使用した飛行ロボット

2. 4回転翼自律飛行ロボット

本実験で使用するロボットは4回転翼自律飛行ロボット [2]である。この飛行ロボットにはボードコンピュータ RaspberryPiを2台搭載しており、1台が機体の制御、もう1台がカメラの制御を行っている。

2.1 飛行ロボットに搭載しているカメラ

本研究で使用したカメラは「Pixy CMUcam5(以下、Pixyと呼ぶ)」である。Pixyはカラーオブジェクトの2次元座標

連絡先: 大谷元気, 室蘭工業大学工学研究科博士前期課程情報電子工学系専攻, 18043011@mmm.muroran-it.ac.jp, bfb46mm@gmail.com

を Pixy 内部のプロセッサで取得することができる。Pixy は飛行ロボットの下部に下向きに搭載しており、地面に設置したカラーオブジェクトを捉えることができるようになっている。さらに、Pixy に魚眼レンズを取り付けることで画角を 75 度から 360 度に広げている。

Pixy は図 2 のように捉えたカラーオブジェクトを四角形の枠で囲い、その枠の左上角の座標と枠の幅と高さを取得する。その座標データを保存し、学習用のデータとして使用している。Pixy が取得できるデータは 1 秒間に最大 50 データとなっている。

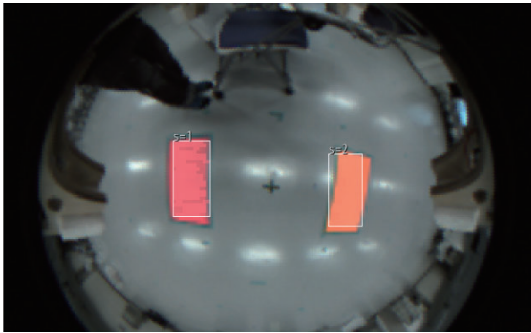


図 2: Pixy によるカラーオブジェクトの識別の様子

3. ニューラルネットワーク

本研究では順伝播型のニューラルネットワークを使用する。カメラから得られるカラーオブジェクトの 2 次元座標を入力とし、飛行ロボットの 3 次元座標と向きを出力するように学習を行う。教師データとしてモーションキャプチャで取得した飛行ロボットの 3 次元座標と向きを用いる。本研究では、python2.7.13, chainer5.1.0 を使用した。

3.1 ニューラルネットワークの入力データ

本研究では、入力データとして 2 つのカラーオブジェクトのデータを使用する。カラーオブジェクトとして、ピンク色とオレンジ色のフェルトの 2 枚を用いる。2 枚のカラーオブジェクトを使用することで、飛行ロボットの向きを表現できると考えた。そして図 3 のようにピンク色のフェルトの左上角の x 座標, y 座標, 幅, 高さ, そしてオレンジ色のフェルトの左上角の x 座標, y 座標, 幅, 高さの 8 つのデータを入力データとした。

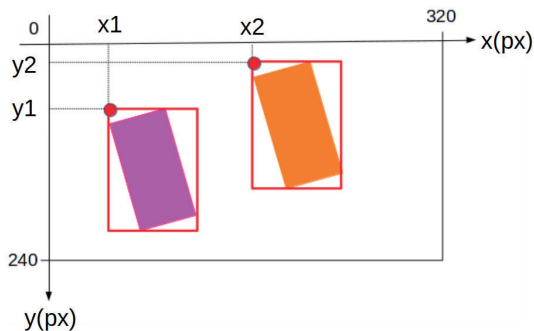


図 3: Pixy が取得する 2 次元座標

3.2 ニューラルネットワークの教師データ

学習における教師データとして、モーションキャプチャで取得した実際の飛行ロボットの 3 次元座標と向きの 4 つのデータを使用する。飛行ロボットには 5 つのマーカがついており、モーションキャプチャではそれぞれのマーカの 3 次元座標を取得することができる。この 5 つのマーカの平均を取り、その値を飛行ロボットの 3 次元座標としている。

飛行ロボットの向きは 2 つのマーカを基準として計算している。2 つのマーカの高さを除く 2 次元座標をそれぞれ (x_1, y_1) , (x_2, y_2) と置くと、

$$angle = \tan^{-1}\left(\frac{y_1 - y_2}{x_1 - x_2}\right) * \frac{180}{\pi} \quad (1)$$

となる。向きの変化を視覚的にわかりやすくするために度数法を用いた。そのため、向きの範囲は $-180 < angle < 180$ となる。

3.3 ニューラルネットワークの構成

本研究では順伝播型のニューラルネットワークを使用した。入力層は 8 ユニット、中間層は 1 層で 100 ユニット、出力層は 4 ユニットとする (図 4)。

中間層の活性化関数 σ_1 は $\tanh$ とする。

$$\sigma_1(x) = \tanh(x) \quad (2)$$

出力層の活性化関数 σ_2 は恒等写像を用いる。

$$\sigma_2(x) = x \quad (3)$$

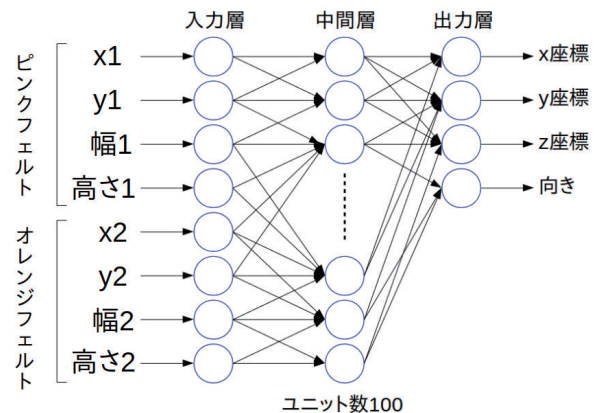


図 4: 本研究におけるニューラルネットワークの構成

4. 飛行実験

4.1 実験手法

モーションキャプチャにより原点とする位置を決める。そして地面に二枚のカラーフェルトを原点から x 軸方向に +400mm, -400mm の位置 (図 5) に貼り付ける。モーションキャプチャのレートとカメラの fps(frames-per-second) を 50 に指定し飛行ロボットを飛行させる。飛行中の飛行ロボットの 3 次元座標をモーションキャプチャで取得し、同時にカメラによりカラーフェルトの 2 次元座標を取得する。その後、飛行ロボットの 3 次元座標とカラーフェルトの 2 次元座標のデータを同期させ訓練データ及びテストデータを作成する。

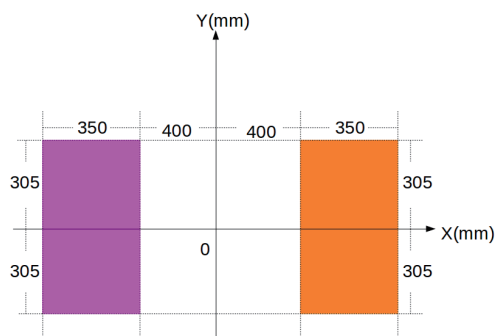


図 5: カラーオブジェクトの位置関係図

4.2 実験結果

モーションキャプチャの範囲に制限があるため、モーションキャプチャのデータ取得の範囲外に出てしまうことがしばしば見られた。そして、同様にカメラの画角にも制限があるため、カラーオブジェクトがカメラの範囲外に出てしまうことも起きた。このように取得できなかったデータを除き、訓練データ及びテストデータを作成した。約 100 回の飛行により、25000 ほどのデータを取得した。実際の飛行の様子を図 6 に示す。



図 6: 飛行ロボットが実際に飛行している様子

5. ニューラルネットワークによる学習

5.1 学習条件

飛行実験で入手したデータのうち約 18000 データを訓練データ、約 6000 データをテストデータとして学習を行った。過学習を防ぐため、テストデータの平均自乗誤差が最小になるところで学習をとめる。学習を行う際、すべてのデータが+1 から-1 の範囲に収まるように正規化する。

5.2 学習結果

学習結果の確認として、学習に使用していない未知のデータを用意しモーションキャプチャによる飛行ロボットの 3 次元座標とニューラルネットワークを使用した飛行ロボットの 3 次元座標を比較した。そして平方平均自乗誤差 (RMSE) を計算した。ニューラルネットワークを利用して得られた x, y, z, 向きデータを O, その教師データとなる x, y, z, 向きのデータを T, 訓練データの総数を n とすると、

$$RMSE = \sqrt{\frac{1}{n} \sum_{k=1}^n (O_k - T_k)^2} \quad (4)$$

となる。

5.3 ニューラルネットワークとモーションキャプチャの比較

テストデータを入力とした時に得られた x, y, z 座標と向きとその時の実際の飛行ロボットの x, y, z 座標と向きを比較した図を図 7, 8, 9, 10 に示す。このテストデータでの RMSE は x 座標は 356.43mm, y 座標は 245.40mm, z 座標は 129.39mm, 向きは 31.03° となった。

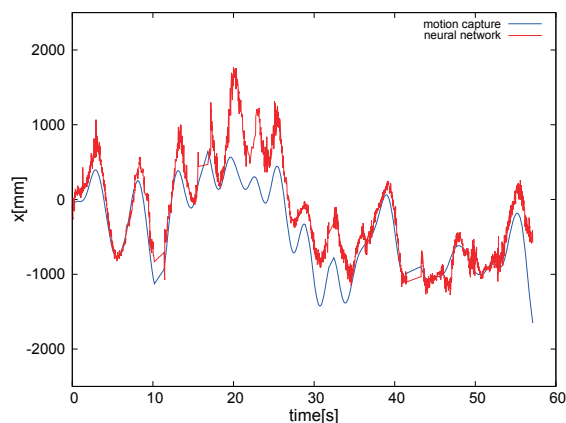


図 7: モーションキャプチャで取得した飛行ロボットの x 座標 (青) とニューラルネットワークを使用して得られた x 座標 (赤) の比較

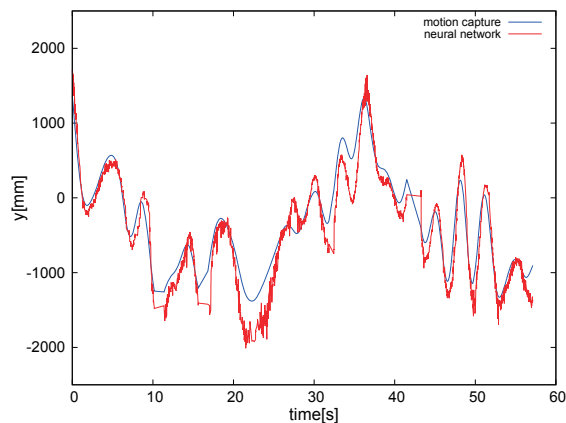


図 8: モーションキャプチャで取得した飛行ロボットの y 座標 (青) とニューラルネットワークを使用して得られた y 座標 (赤) の比較

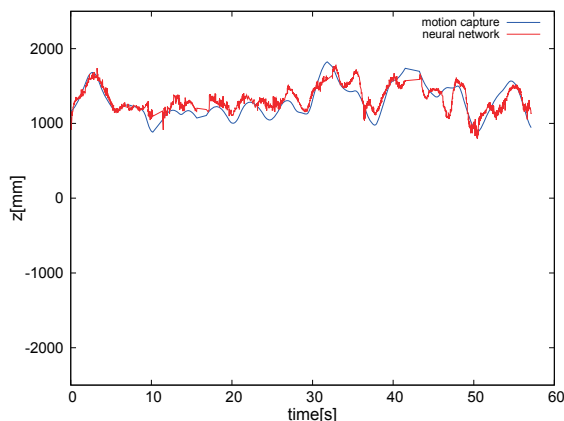


図 9: モーションキャプチャで取得した飛行ロボットの z 座標 (青) とニューラルネットワークを使用して得られた z 座標 (赤) の比較

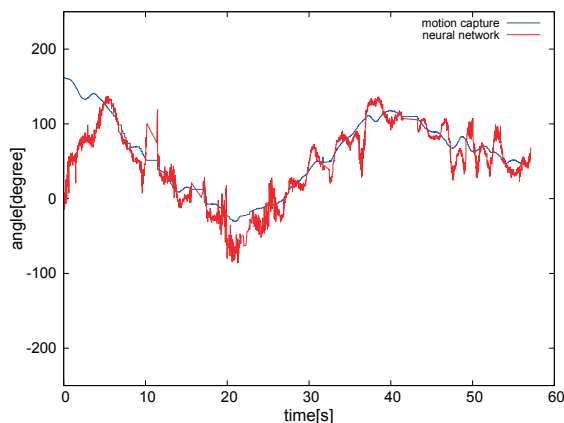


図 10: モーションキャプチャで取得した飛行ロボットの向き (青) とニューラルネットワークを使用して得られた向き (赤) の比較

5.4 考察

飛行ロボットの x, y, z 座標はモーションキャプチャに近い値を出す結果となった。その中でも x 座標は誤差が大きく出ていたが、これはカラーオブジェクトの形と配置が原因であると考えられる。本実験では長方形のカラーオブジェクトを 2 枚配置していたが、カラーオブジェクトを丸のように角度による見え方の変化が小さいものに変更したり、カラーオブジェクト間のスペースを変更することで誤差を減らすことができるのではないかと考えている。また、学習データを増やすことで全体の誤差も減らすことができると考えている。

向きの出力に関しては、大きいところで 180 度近い誤差がうまれた。本実験では、向きを -180° から $+180^\circ$ の範囲で出力するように学習を行っている。そのため、 -180° と $+180^\circ$ の付近ではニューラルネットワークの入力は近い値になるが、出力は大きく変わることで誤差が生じると考えられる。この問題は角度の出力方法を変更することで改善されようと考えている。

6. RaspberryPi によるニューラルネットワークの計算

学習は外部 PC により行っているが、実際に飛行ロボットに搭載する際は RaspberryPi で処理を行う。RaspberryPi は cpu の処理能力が高くないため、実際にどのくらいの速度でニューラルネットワークの計算を行えるのか検証する必要がある。そこで RaspberryPi を使いニューラルネットワークの計算を行い、どのくらいのフレームレートが出るのかの実験を行った。Pixy のフレームレートが最大 50fps のため 50fps を 1 つの基準とする。

ニューラルネットワークに同じ値を 5000 回入力し、その時の計算時間を測定する。そして、5000 を秒数で割ることで平均の fps を算出した。

その結果、RaspberryPi でのニューラルネットワークの計算のフレームレートは約 180fps となった。

7. まとめ

ニューラルネットワークによる飛行ロボットの 3 次元座標と向きは、多少の誤差があるものの概ね出力することができた。また、RaspberryPi によるニューラルネットワークの計算時間の検証により、ニューラルネットワークの計算量を増加させられることも示された。中間層やユニット数を増やすことで誤差の減少につながると考えている。

8. 今後の課題

カラーオブジェクトの形や配置場所を見直し誤差の減少を目指す。また、角度の表し方を見直し、 -180° 、 $+180^\circ$ 付近の誤差の減少を目指す。さらに、ニューラルネットワークの中間層やそのユニット数を増やすことで、誤差がどのように変化するかの実験も必要となる。そして、今現在は飛行ロボットで取得したデータを外部 PC で計算しているため、実際に飛行ロボットに搭載して飛行実験も行っていきたい。

参考文献

- [1] 第 19 回交通流のシミュレーションシンポジウム論文集, p53-56, 2014 本田 泰
- [2] ロボットインテリジェンス, 浅田 稔, 國吉 康夫
- [3] ドローン同士の直接通信でニアミスを自動的に回避する実験に成功, <https://www.nict.go.jp/press/2019/01/24-1.html> (参照日:2019/02/07)
- [4] 第 22 回交通流のシミュレーションシンポジウム論文集, p17-20, 2016 本田 泰