1I5-3

能動的マーカ貼付により自己位置推定を行う自律飛行ロボット

Self Localization with Active Attachment of AR Markers by Autonomous UAV

三木 崇弘　　堀 浩一
Takahiro Miki　Koichi Hori

東京大学大学院 工学系研究科
School of Engineering The University of Tokyo

小型飛行ロボットの自律飛行のためには自己位置の推定が必要である。マーカをカメラで認識することで位置推定が可能となるが、移動はマーカが設置されている範囲に限定されてしまう。本研究では自律的にマーカを天井に設置していくことで、自ら活動範囲を広げていく飛行ロボットを提案する。マーカを設置する位置の状況に応じた判断には強化学習を用いた事前学習結果を利用する。

1. はじめに

1.1 研究背景と目的

現在無人空中機（Unmanned Aerial Vehicle UAV) に関する研究が多く行われており、中でもクアッドコプターに関する研究が盛んである。クアッドコプターはホバリングした状態で自律飛行することができることから、様々なミッションへの応用が期待されている。例として、ブロックを積み上げ構造物を作る研究 [Willmann 12], 案内ロボットを作成した研究 [sky] などがある。

ミッションを実行するためには自己位置の認識が必要である、本研究では、画像マーカを用いた位置推定について検討したい。マーカを元に自己位置を推定する研究はあるが,[Rudol 10]、これらは予め地面に設置してあるマーカを元に自己位置を推定するものである。

そこで、飛行ロボットが自らマーカを貼り、活動領域を広げていくことを考える。本研究の目的は、自らの判断でマーカを天井に貼り、行動できる範囲を広げていく飛行ロボットの実現である。

1.2 強化学習について

強化学習とは、エージェントが報酬を最大にする方策を見つけることを目標とするものである。エージェントはある環境の中で行動をすると、環境から行動に応じて報酬が得られる。エージェントは試行錯誤をしつつ、その時々の状況においてどの行動が最も良いのかを学習する。学習は行動価値関数 Q(s,a) の値の更新をすることによって行われる。Q(s,a) はある状況 s において、行動 a を行うことの価値を 1 つの数値で表すものでこの数値が高いほど良い行動だと判断される。この値は Q 学習という手法を用いる場合、ステップ毎に以下のように更新する。

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

2. マーカの貼り付け位置の学習

自律的にマーカを貼り付ける際にその位置を判断しなければならないが、ずれて貼ってしまった場合や、貼れなかった場合などに対応して判断しなければならない。そこで、本研究では強化学習を用いてマーカを貼る順番と位置を学習させた。オフラインで学習し、その結果を実機で利用するものとする。

連絡先: 三木 崇弘，東京大学大学院工学系研究科 航空宇宙工学専攻 知能工学研究室,miki@ailab.t.u-tokyo.ac.jp

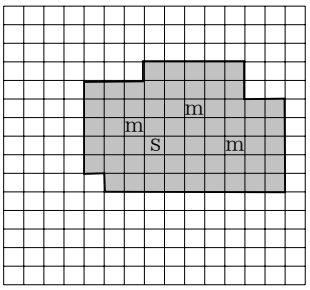
表 1: シミュレータの設定

マップサイズ	30 × 30
開始地点 (充電できる地点)	(15,15)
持っているマーカの数	15 枚
バッテリ容量	50
充電できる上限回数	5 回
視界の大きさ	5 マス

表 2: 想定と異なる行動を起こす確率

正常に貼れる確率	0.3
正常でない場合にずれる確率	0.7
正常でない場合に落とす確率	0.3

2.1 シミュレータの設定



s:開始地点（充電地点）
m:マーカ
灰色の領域:移動可能な領域

図 1: シミュレーションのマップ

1,1 のようなマップを考える。また、2 の確率でずれが生じるとした。状態、報酬、行動は以下のようにした。

- 状態
  自機の座標, バッテリの状態, マーカのマップ
- 報酬
  - 移動可能な面積の増加あたり 3
  - 充電を一回する 100
  - バッテリ消費 1 につき -1
- 行動
  - 移動可能なマス目へと移動する

この条件で、100 ステップ行い学習させた。

2.2 シミュレーションによる学習の結果

結果は、2,3 のようになった。

2 の左側はマップを表しており、緑の四角がマーカ、赤の四角が自機の位置である。右側はその状態の時の行動価値関数の値を表している。これを見ると、外側へ貼ることの価値が高くなっており、面積を広げようとしていることがわかる。

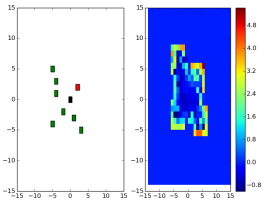


図 2: 学習の結果

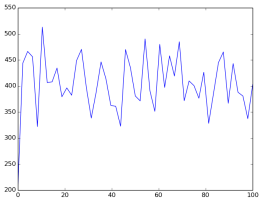


図 3: 報酬の推移

しかし、3 よりあまり学習が進んでいないことがわかる。これは状態数が多すぎたため、同じ状態になることが殆ど無く、価値の更新があまりされなかったためであると考えられる。学習を進ませるためには、より多くのステップ数で計算させると共に、マップの回転や反転などを同じとみなすなど状態数を減らす工夫や、学習手法の工夫が必要だと考えられる。

このシミュレーションでは想定した位置とずれた位置に貼ってしまうことを考えている。よって、学習は進まなかったが、マーカが貼る位置からずれる場合などに対応するマーカ貼り付け位置の判断は出来たと考えられる。

3. 作成したシステム

3.1 ハードウェア構成

本研究で作成したシステムは PC と AR.Drone、マーカ貼り付け装置、マーカから成っている。PC は Wifi を介して、AR.Drone と通信することができ、速度を与えることで操作することができる。AR.Drone のカメラは上向きにしてあり、天井を撮影することができる。また上にマーカ貼り付け装置を搭載した。マーカは 6,7 のようになっており、互い違いに重ねることで、下のマーカと貼りつかないようになっている。

図 4: AR.Drone

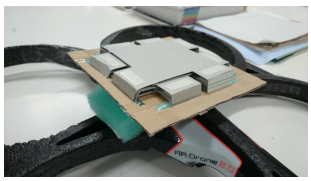


図 5: マーカ貼り付け装置

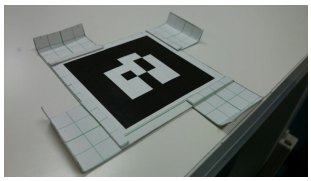


図 6: マーカ (表)

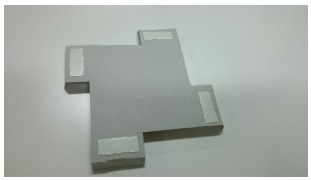


図 7: マーカ (裏)

3.2 ソフトウェア構成

本研究では、AR.Drone との通信に ROS(Robot Operating System)[WillowGarage. 12] を用いた。AR.Drone との通信には ardrone_autonomy[Monajjemi 12]、マーカの認識には ar_track_alvar[Niekum. 13]、コントローラの入力には joy[Morgan Quigley] を用いた。ROS のノードの構成は 8 のようになっている。ardrone_autonomy、ar_track_alvar、joy、send_cmd、marker_sub、brain はパッケージと呼ばれるものであり、それぞれが、並列的に実行される。本研究では、marker_sub,send_cmd,brain を作成した。

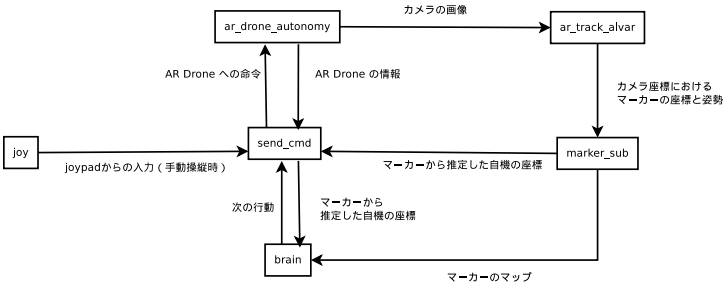


図 8: ROS のノード構成

3.2.1 marker_sub

このノードでは、カメラから見たマーカの情報を元に、自機の絶対座標での位置を求めると同時に、次々にマーカの天井における位置を推定し、登録していく。このノードが認識した位置は send_cmd へと送信され、機体の速度と合わせて位置推定に使われる。また、このノードが認識しているマーカのマップは brain へと送信され、現在の状態を決めるために用いられる。

3.2.2 send_cmd

このノードでは、マーカから推定された自己位置と、AR.Drone から得られる速度の情報からカルマンフィルタを用いて、自己位置を推定する。推定した位置は、brain へと送信される。また、brain から行動が送信されると、send_cmd はその命令に基づいて、マーカを貼る、充電するために最初の地点へと戻るなどの制御を行う。
AR.Drone への命令は自動でされるが、手動操縦に切り替えることができる。これは、実験中に予期しない動作をしてしまったり、危険な状況になった時に安全を確保するためである。

3.2.3 brain

このノードは、marker_sub からのマーカのマップ、send_cmd からの自機の座標とバッテリの残量から現在の状態を求め、強化学習を用いて、次の行動を決定し、send_cmd へと送信する。

3.3 マーカの天井座標への変換と自己位置推定

marker_sub ではマーカの天井での位置を計算して登録していく。また、そこから自己位置を推定する。ar_track_alvar からは、カメラから見たマーカの座標と姿勢が得られる。そこで、天井での位置をすでに位置のわかっているマーカより求め、登録していく必要がある。今、ひとつの位置が既知のマーカと、ひとつの位置が未知のマーカが見えているとする。ここで、

- l:マーカ間の長さ
- α:マーカの中心同士ををつなぐ線のカメラから見た角度
- θ_0:カメラから見た既知のマーカの角度
- θ:カメラから見た未知のマーカの角度
- θ_a:既知のマーカの登録されている絶対角度
- θ_m:未知のマーカの求めたい絶対角度
- x_a, y_a:既知のマーカの登録されている絶対座標
- x_m, y_m:未知のマーカの求めたい絶対座標

とすると、次の式より、未知のマーカの位置を求めることができる。

$$x_m = x_a + l\cos(\theta_a + \alpha + \theta_0) \quad (2)$$
$$y_m = y_a + l\sin(\theta_a + \alpha + \theta_0) \quad (3)$$
$$\theta_m = \theta_0 - \theta + \theta_a \quad (4)$$

次に既知のマーカを見ている状態から、自己位置を求めることを考える。

- x, y:カメラから見たマーカの座標
- θ:カメラから見たマーカの角度
- $l = \sqrt{x^2 + y^2}$
- $\alpha = \arctan(x/y)$
- x_a, y_a, θ_a:登録されているマーカの絶対座標と絶対角度

- $x_{my}, y_{my}, \theta_{my}$ 求めたい自機の絶対座標と絶対角度

この情報を元に自機の絶対位置を求める。以下の式より、自機の座標と Yaw 角を求めることができる。

$$\beta = \theta + \theta_a - \alpha - 180 \text{ として} \quad (5)$$
$$x_{my} = x_a + l\cos\beta \quad (6)$$
$$y_{my} = y_a + l\sin\beta \quad (7)$$
$$\theta_{my} = \theta + \theta_a \quad (8)$$

3.4 カルマンフィルタを用いた位置推定

send_cmd では、マーカから得られる位置と、AR.Drone から得られる速度からカルマンフィルタを用いて自己位置推定を行う。状態方程式、観測方程式は以下で表される。

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{w}_{k-1} \quad (9)$$
$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \quad (10)$$

である。ここで、

$$\mathbf{x} = \begin{pmatrix} x \\ y \\ z \\ \phi \\ \dot{x} \\ \dot{y} \\ \dot{z} \\ \dot{\phi} \end{pmatrix}, A = \begin{pmatrix} 1 & 0 & 0 & 0 & \Delta t & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$z = \begin{pmatrix} x \\ y \\ z \end{pmatrix}, H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \text{ とする。}$$

x, y, z は自機の座標、ϕ は自機の Yaw 角度である。予測は、AR.Drone から得られる速度を用い、マーカの観測をしたらその値を用いて更新した。AR.Drone からは約 200Hz で速度が得られ、マーカの観測は約 10Hz で行われた。本研究におけるカルマンフィルタのパラメータ設定は以下のようになった。Yaw 角に関しては、マーカからの値のみを使った。

誤差共分散行列 $\mathbf{Q}, \mathbf{R}$

$$\mathbf{Q} = \begin{pmatrix} 0.1 & 0 & \cdots & 0 \\ 0 & 0.1 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & 0.1 \end{pmatrix}, \mathbf{R} = \begin{pmatrix} 50 & 0 & 0 & 0 \\ 0 & 50 & 0 & 0 \\ 0 & 0 & 50 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

AR.Drone から得られるのは機体の座標での速度であるため、Yaw 角を用いて、絶対座標での速度を計算して用いた。

$$V_x = v_x\cos\phi - v_y\sin\phi \quad (11)$$
$$V_y = v_x\sin\phi + v_y\cos\phi \quad (12)$$

ここで、V は絶対座標での速度、v は AR.Drone の座標における速度、ϕ は Yaw 角度である。

3.5 PD 制御を用いた位置制御

send_cmd では位置制御も行う。目標の位置へと移動する制御は PD 制御を用いた。x 方向の速度は以下のように計算した。

$$u_{xi} = K_p\Delta x_i + K_d(\Delta x_i - \Delta x_{i-1})/\Delta t \quad (13)$$

本研究ではゲインを以下のように設定した。

$$K_p = 0.15, K_d = 0.4 \quad (14)$$

同様にして、y 方向、z 方向の計算も行う。また、絶対座標での速度の機体座標における速度への変換は以下のように変換した。

x 方向:$-u_x sin\theta + u_y\cos\theta$

y 方向:$-u_x\cos\theta - u_y\sin\theta$

4. 実験結果

4.1 マーカ認識の結果

天井にマーカを貼っておき、その下において AR.Drone を動かし、マーカの位置と自己位置を認識できることを確認した。マーカは 9 のように貼り付けた。このマーカ配置においてマーカを順番に認識していった結果は 10 のようになった。右側の図は登録されたマーカの位置を描画したものであり、左側は自己位置の軌跡を描画したものである。マーカの位置とその角度が認識されていることが確認できた。

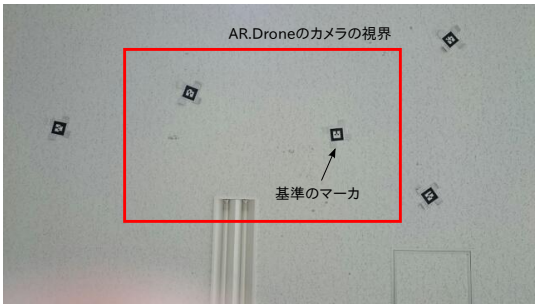


図 9: 貼り付けたマーカ配置

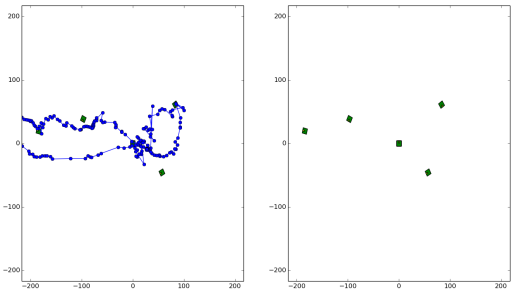


図 10: マーカ認識の結果

4.2 カルマンフィルタを用いた自己位置推定の確認

11 は飛行した軌跡を描いたものである。青い線はマーカより得られる位置の軌跡であり、緑の線はカルマンフィルタを適用した位置の軌跡である。青い線よりマーカを見失うと位置が跳んでいることがわかる。一方、カルマンフィルタ適用後はある程度推定できていることが確認できた。12 は、途中でマーカの認識の外に移動させた時のグラフ

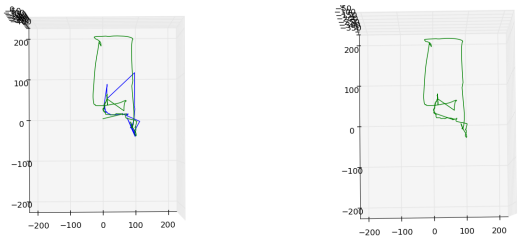


図 11: 上から見た軌跡

である。横軸が時間、縦軸が x,y,z それぞれの座標の値であり、実線で描かれているのがカルマンフィルタを適応した位置の軌跡、点で描かれているのが、マーカを認識した時に得られる、位置である。マーカを見失っている時でも、速度よりしばらくは位置の推定ができていることがわかった。そのため、マーカを見失ってももとの位置へ戻ることができる。

4.3 位置制御の確認

次に位置制御の実験を行った。目標座標は (0,0,-100) とした。13 にその結果を示した。離陸した後、図の制御開始点から目標座標へと制御を開始し、図の制御終了点まで制御した後、着陸した。マーカを貼り付けるというミッションを考えた時には十分な精度で制御が出来たと考えられる。

4.4 ミッションの実行

ミッションを実行した結果、5 枚のマーカを連続して自動で貼ることに成功した。また、実際の写真は 15 のようになった。固まっているマーカは同じ場所に貼りに行ってしまったマーカである。同じ所に貼りに行ってはしまったが、マーカを貼る位置の制御はできているとわかった。

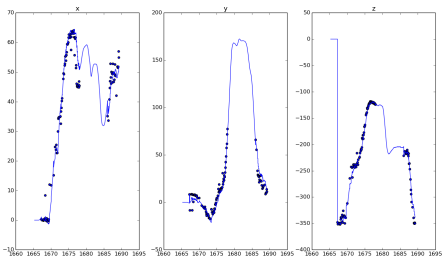


図 12: カルマンフィルタを適用した結果

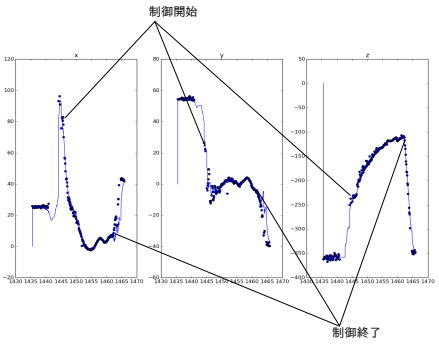


図 13: 位置制御の結果

図 14: ミッション中の様子

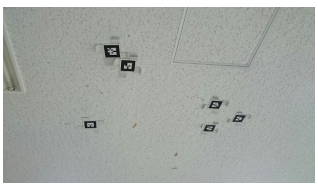


図 15: ミッション後の天井の写真

5. 結論

5.1 本研究の成果

まず、天井に貼ったマーカを順次認識していき、その位置を登録することが出来た。また、マーカから自己位置を推定し、目標座標へと位置制御を行うことが出来た。また、マーカ、自分の位置に応じてマーカの貼り付け位置の判断をずれた場合でも行うことが出来た。その結果目標としていた自律的にマーカを天井に貼り、移動可能な領域を広げることに成功した。

5.2 今後の課題

強化学習による判断の獲得では状態数が多く、学習が思うように進まなかった。そのため、状態数を減らす工夫や学習手法の工夫が必要である。また、マーカを天井に貼るときにずれが生じてしまった。この時、マーカを見ることのできている時間が短いほどずれが大きくなったと考えられる。そのため、マーカを貼る制御での工夫と、マーカをなるべく見ることのできる位置へと貼り付けるように学習させるなどの工夫により、より正確にマーカを貼り付けることができるようになると考えられる。

参考文献

[Monajjemi 12] Monajjemi, M.: ardrone_autonomy: a ROS Driver for ARDrone 1.0 & 2.0 (2012)

[Morgan Quigley] Morgan Quigley, K. W. B. G., Brian Gerkey: joy -ROS Wiki: `http://wiki.ros.org/joy`

[Niekum. 13] Niekum., S.: ar_track_alvar-ROS Wiki (2013), `http://www.ros.org/wiki/ar_track_alvar`

[Rudol 10] Rudol, P., Wzorek, M., and Doherty, P.: Vision-based pose estimation for autonomous indoor navigation of micro-scale unmanned aircraft systems, in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1913–1920IEEE (2010)

[sky] skycall`http://senseable.mit.edu/skycall/`

[Willmann 12] Willmann, J., Augugliaro, F., Cadalbert, T., D'Andrea, R., Gramazio, F., and Kohler, M.: Aerial robotic construction towards a new field of architectural research, *International journal of architectural computing*, Vol. 10, No. 3, pp. 439–460 (2012)

[WillowGarage. 12] WillowGarage., : Documentation - Robot Operating System (October 2012), `http://www.ros.org/wiki/`