

卒業論文 2019 年度 (令和元年度)

QR コードを用いた経路情報に関してステートレスなロボットナビゲーションシステムの開発

慶應義塾大学 環境情報学部環境情報学科

井手田悠希

| |
|---|
| QR コードを用いた経路情報に関してステートレスなロボットナビゲーションシステムの開発 |
|---|

人間が物品管理を行っている倉庫で物品管理の自動化を行おうとする場合二通りの方法が考えられる。一つ目が物品棚自体に機械を搭載したり、物品棚自体をロボットで移動させながら物品管理を行う方法。二つ目は倉庫内は何も手を加えずに巡回ロボットを用いて物品管理を行う方法である。本研究では2つ目の方法で物品管理に利用されることを想定したロボットのナビゲーションシステムの開発を目指す。ロボットには移動の自由度が高いドローン(小型無人航空機)を採用している。ロボットを自律的に倉庫内を移動させる場合ロボットの経路情報の管理は非常に重要であり、現状ロボットのナビゲーションシステムでは経路情報はソフトウェアによる生成及び管理が主である。しかしながら、倉庫での巡回という目的においては現状のナビゲーションシステムは経路情報の扱いを含めて少々冗長な部分が存在している。そこで今回は特に経路情報の扱いに注目し、経路情報に関してステートレスなナビゲーションシステムを考案し利用可能かを検証した。本研究ではQRコードを用いて経路情報に関してステートレスなナビゲーションシステムの実現をする。

キーワード:

1. ドローン, 2. ナビゲーションシステム, 3. 倉庫, 4. QRコード,

慶應義塾大学 環境情報学部環境情報学科
井手田悠希

| |
|---|
| Development of a robot navigation system with QR code |
|---|

There are 2 ways which realize automate item management system in the warehouse. The first way is using an automated rack or using robots to move a rack. The second way is using patrol robots without any warehouse reform. This research focuses on the robot which is used in a second way. We use a drone as a robot because the drone has many degrees of freedom. It is important that is route information and managed by software mainly. However, It is verbose a little for patrol in the warehouse that already exists methods. Thus, this research targeted the route control system and developed a stateless route control system. In this system, we use QRCode to develop a stateless route control system.

Keywords :

1. Drone, 2. Navigation System, 3. WareHouse, 4. QRCode,

Keio University Bachelor of Arts in Environment and Information Studies

Yuki Ideta

目次

| | | |
|------------|-----------------------|----------|
| 第1章 | 序論 | 1 |
| 1.1 | 本研究の背景 | 1 |
| 1.1.1 | 自己位置推定手法の発展 | 1 |
| 1.1.2 | 倉庫内での物品管理方法 | 1 |
| 1.2 | 本研究の問題と仮説 | 1 |
| 1.2.1 | 倉庫の巡回タスクにおける特徴 | 2 |
| 1.2.2 | 倉庫の巡回タスクにおける課題 | 2 |
| 1.3 | 本論文の構成 | 2 |
| 第2章 | 本研究の要素技術 | 3 |
| 2.1 | PnP 問題 | 3 |
| 2.2 | QR コード | 3 |
| 第3章 | 本研究における問題定義と仮説 | 5 |
| 3.1 | 本研究における問題定義 | 5 |
| 3.2 | 倉庫の巡回タスクで利用する場合の課題 | 5 |
| 3.2.1 | 経路情報の管理 | 5 |
| 3.2.2 | 環境地図の作成 | 5 |
| 3.2.3 | 専用機材の用意 | 6 |
| 3.3 | 本研究における仮説 | 6 |
| 3.3.1 | 提案手法 | 6 |
| 3.3.2 | 各アプローチとの比較 | 6 |
| 第4章 | 提案手法 | 7 |
| 4.1 | 概要 | 7 |
| 4.2 | コントローラ部 | 7 |
| 4.3 | 映像部 | 8 |
| 4.4 | トラッカー部 | 8 |
| 4.5 | ナビゲーション部 | 8 |
| 第5章 | 実装 | 9 |
| 5.1 | 実装環境 | 9 |
| 5.2 | コントローラ部 | 9 |
| 5.3 | 映像部 | 10 |

| | | |
|--------------|--|-----------|
| 5.4 | トラッカー部 | 11 |
| 5.5 | ナビゲーション部 | 12 |
| 第 6 章 | 評価 | 13 |
| 6.1 | 評価方法 | 13 |
| 第 7 章 | 関連研究 | 14 |
| 7.1 | Nanomap | 14 |
| 7.2 | Octomap | 14 |
| 7.3 | 実用性を備えた手のひらサイズ・完全オンボード処理 UAV | 15 |
| 7.4 | DeepDrone | 15 |
| 7.5 | TDOA を用いた UAV の自己位置推定 | 16 |
| 第 8 章 | 結論 | 17 |
| 8.1 | 本研究のまとめ | 17 |
| 8.2 | 本研究の課題 | 17 |
| 付 録 A | 付録 | 18 |
| A.1 | 付録内容 | 18 |
| | 謝辞 | 19 |

図 目 次

| | |
|--|----|
| 2.1 QR コードの仕様: https://en.wikipedia.org/wiki/QR_code | 3 |
| 4.1 システム概要図 | 7 |
| 5.1 QR コードの姿勢推定結果 | 11 |
| 7.1 圧縮率の違いによる生成物の違い | 15 |
| 7.2 シミュレータ上のゲート | 16 |

表 目 次

第1章 序論

本章では本研究の背景，課題及び手法を提示し，本研究の概要を示す．

1.1 本研究の背景

1.1.1 自己位置推定手法の発展

モバイルデバイスの高度化やセンサの小型化，計算能力の向上により高度な処理が可能となった．これにより自己位置推定の様々な手法が提案され，その精度も向上を続けている．しかしながら実運用を考慮すると運用に必要な金銭的費用や機体性能，環境による制約が多く要件に応じた適切な自己位置推定手法を取る必要がある．

1.1.2 倉庫内での物品管理方法

倉庫内の物品管理¹の方法は大別すると以下の二通りが挙げられる．一つ目が物品棚自体に機械を搭載したり，物品棚自体をロボットで移動させながら物品管理を行う方法．二つ目は倉庫内は何も手を加えずに巡回ロボットを用いて物品管理を行う方法である．自動型の物であれば物品棚自体に自動化用の機械を取り付ける場合や物品棚自体をロボットで移動させながら物品管理及び取得・移動を行う物が存在している．一方で人間が読み取り機を用いて物品管理や物品の配置/取得をする形式の倉庫も多く存在している．現在人間が物品管理を行っている倉庫で新たに形で自動化を行おうとすると倉庫自体の大きな改修が必要となる．ここで大規模な改修が不要な物品管理の自動化の方法として自律航行するロボットを巡回させる形での物品管理の自動化が考えられる．また，この際用いられるロボットとしては高さのある棚に対しても対応が容易な自由度の高いドローンを用いることが想定される．そこで本研究では将来的に物品管理に利用されることを想定したドローンのナビゲーションシステム²の開発を目指す．

1.2 本研究の問題と仮説

自律航行するロボットの制御において最も重要な要素のうちの 하나가自己位置推定である．自己位置推定とは，任意の系においてロボット自身がどこにいるのかを推定する事で

¹本論文では定期的に物品の位置の記録を行うことを指す

²本論文ではロボットの自己位置推定と経路選択を行うシステムを指す

ある。

現在自己位置推定には GPS(Global Positioning System) を用いる手法が非常に有力であり頻繁に用いられる。しかし、屋内等の理由により直上方向に遮蔽物がある場合にはその精度が著しく低下することや利用できないという状況が発生する。その為屋内では後述する Visual SLAM や LiDAR SLAM, TDOA 等様々な自己位置推定手法が考案され利用されている。これらの手法を利用したナビゲーションシステムで倉庫の巡回タスクを想定した実運用を考えると、いくつかの冗長な点と問題点が浮上する。

1.2.1 倉庫の巡回タスクにおける特徴

まず倉庫の巡回というタスクを考えた際に挙げられる特徴を以下にまとめる。

- 移動経路がほぼ固定化する
- 障害物の位置が既知である
- 機器の故障やメンテナンス等により機体の入れ替えが頻繁に発生する
- 収容率を上げる為にも通路は狭ければ狭い程良いとされる

1.2.2 倉庫の巡回タスクにおける課題

これらの特徴を踏まえた上で既存のナビゲーションシステムでは冗長であると考えられる部分や課題を以下にまとめる。詳細については 3 章にて説明する。

- 経路情報の管理 3.2.1
- 環境地図の作成 3.2.2
- 専用機材の用意 3.2.3

1.3 本論文の構成

本論文における以降の構成は次の通りである。

2 章では、本研究の要素技術となる PnP 問題と SLAM と QR コードに着いて整理する。

3 章では、本研究における問題の定義と、解決するための要件の整理を行う。

4 章では、本研究の提案手法を述べる。

5 章では、4 章で述べたシステムの実装について述べる。

6 章では、3 章で求められた課題に対しての評価を行い、考察する。

8 章では、本研究のまとめと今後の課題についてまとめる。

第2章 本研究の要素技術

本章では本研究の要素技術となる PnP 問題と GraphBasedSLAM と QR コードについて各々整理する.

2.1 PnP 問題

PnP(Perspective-n-Point) 問題は 3 次元空間上の n 個の点の情報が与えられた際にカメラ座標系に投影された 2 次元平面状の同じ点をを用いてカメラの外部パラメータ¹を求める. という問題を指す.

本論文上の実装では QR コードの左上の角を世界座標系原点として推定を行う. QR コードの四角を利用し P4P 問題として帰着させる.

2.2 QR コード

自動車メーカーのデンソーに発明された. 汚れによる耐障害性が非常に高く, 非対称なので, 向きの検出も可能. 英数字で最大 4296 文字まで記録が可能

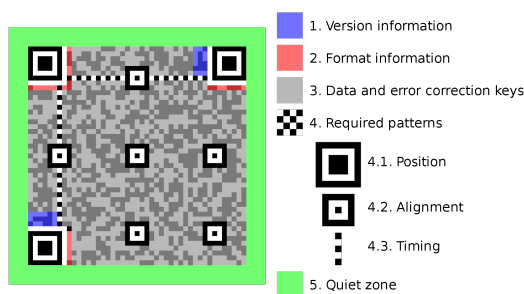


図 2.1: QR コードの仕様: https://en.wikipedia.org/wiki/QR_code

コンピュータ上で検出する際は以下の手順で検出を行う.

1. 水平方向に探索をし, 白黒の比が 1:1:3:1:1 になる箇所を探索しファインパターンの候補として記録する.

¹世界座標系におけるカメラ座標系の座標を求める為の回転と変換のパラメータ

2. 垂直方向に探索をし，ファイパターン候補の中心から上下にそれぞれ白黒の比が 3:2:2 になるか検証する．
3. 3 箇所ファイパターンが求まれば 3 点で構成される平行四辺形の頂点から残りの 1 点を推定し四角を推定します．

データ部分のデコードは右下から順に黒を 1 白を 0 として右下のマスを基点として左隣のマス，上のマス，左上のマス，と順に読み取っていく．

第3章 本研究における問題定義と仮説

本章では1章で述べた背景より、現状の自律航行システムを倉庫内で物品管理に用いる場合の問題点を列挙し、整理する。また、どのように解決すれば良いのかを定義する、

3.1 本研究における問題定義

現状の自律飛行システムを倉庫内での物品管理に用いる場合の問題点を列挙し、整理する

3.2 倉庫の巡回タスクで利用する場合の課題

3.2.1 経路情報の管理

既存の方法による経路情報の管理はいくつか方法が考えられる。

はじめに考えられるのは機体それぞれに飛行プログラムの一部として経路情報を書き込む方法である。外部との通信を必要とせず自律飛行が可能である事に加えて、倉庫の巡回タスクにおいては基本的に経路情報が変化しない為適していると考えられる。しかし飛行前に経路情報のチェックやアップデートの有無の確認、経路情報のバージョン管理はやはり必要であると考えられる。また、機器の故障やメンテナンスによる機体の入れ替えの度に飛行経路に応じた書き換えが必要となる。同じ経路のプログラムを書き込んだ予備機体を用意することも考えられるが飛行させたい経路が増えるにつれてその予備台数も増加する事になってしまう。

次に考えられるのがグラウンドコントロールシステムによる中央集権的な管理である。基本の飛行システムさえ書き込んでおけば経路情報と誘導はグラウンドコントロールシステム側で一元管理が可能になり飛行前のアップデートの確認の手間やバージョン管理は不要になると考えられる。機体側は外部通信機構が必須になる上グラウンドコントロールシステム側での複数機体の管理、グラウンドコントロールシステムと各機体とのコミュニケーションが必要となる上に機体数が増えた場合にそのネットワークの帯域も問題となる可能性があるというデメリットがある。

3.2.2 環境地図の作成

今回対象としている倉庫の巡回タスクにおいては

3.2.3 専用機材の用意

3.3 本研究における仮説

3.3.1 提案手法

今回上述した課題を解決する為に QR コードを用いた経路情報に対してステートレスなナビゲーションシステムを考案した。上述した課題をどのように解決するかをそれぞれ以下にまとめる。

3.3.2 各アプローチとの比較

ここに表を入れる。

第4章 提案手法

本章では提案手法について述べる.

4.1 概要

本研究では機材の都合上映像からの自己位置推定, 機体コントロールの計算処理を遠隔 PC 上で行う.

全体のシステム構成は以下の図 4.1 の通りである.

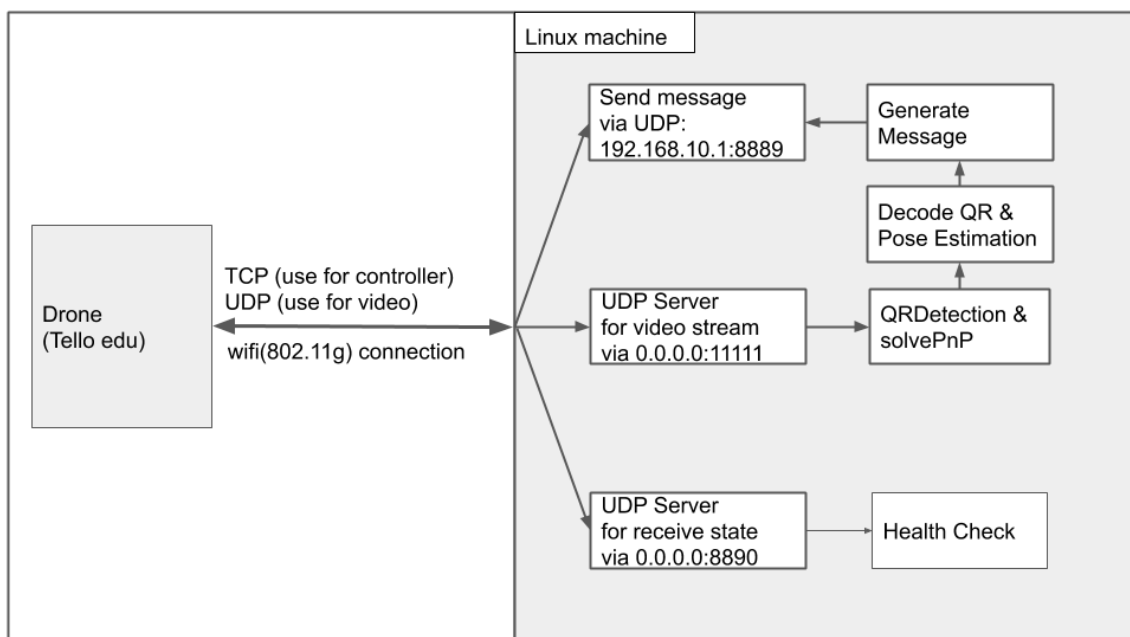


図 4.1: システム概要図

本研究では ROS の使用を前提としたシステム設計となっている.

4.2 コントローラ部

ナビゲーション部で生成された制御命令を実際にドローンに伝える部分. 機体の API のラッパーの役割を担う.

4.3 映像部

映像を取得し，扱い易い形式に変換する部分．今回は UDP Server を建ててドローンから送られてくる映像を受け取る．

4.4 トラッカー部

QR コードの検出とデコードには OpenCV に用意されている QRCodeDetector クラスを利用し，そこから得られた QR コードの四角に対して P4P による 3 次元再構成を行った．

これにより QR コードよりデコードされた次の QR コードの位置情報とカメラ座標系からみたモデルの回転ベクトル，並進ベクトルが得られるのでこれらの情報をナビゲーション部??にて処理をする

4.5 ナビゲーション部

QR コードから得られたカメラの情報と次点の QR コードの位置情報を元にドローンに対してメッセージを送信する．並進ベクトルを cm 単位に変換した後そのまま移動量として利用する．また，回転ベクトルのうち y 軸 (鉛直方向) 回転のデータのみ利用する．これは今回利用している機体が基本制御として機体を水平に保つ為である．

第5章 実装

本章では提案手法の実装について述べる.

5.1 実装環境

本研究では以下の環境で実装及び実行を行う.

- DJI Tello edu
- Ubuntu 18.04
- ROS melodic
- python 3.7, 2.7

5.2 コントローラ部

プログラム 5.1: controller.py

```
1 def go_to_target(self):
2     print('target_pos:', self.target_pos)
3     print('self_pos:', self.drone_pos)
4     move_x_val = int(self.target_pos["x"] - self.drone_pos["x"] -
5                     (self.QR_SIZE/2))
6     move_y_val = int(self.target_pos["y"] - self.drone_pos["y"] -
7                     (self.QR_SIZE/2))
8     move_z_val = int(self.target_pos["z"] - self.drone_pos["z"])
9     rotate_val = int(self.target_pos["rotate"] - self.drone_pos["
10         r"])
11     self.is_moving = True
12     if move_x_val > 0:
13         self.drone.move_right(move_x_val)
14         time.sleep(move_x_val/10 + self.COMMAND_WAIT_BUFFER)
15         print('sleep_x:', move_x_val/10 + self.
16             COMMAND_WAIT_BUFFER)
17     else:
18         self.drone.move_left(-move_x_val)
19         time.sleep(-move_x_val/10 + self.COMMAND_WAIT_BUFFER)
20         print('sleep_x:', -move_x_val/10 + self.
21             COMMAND_WAIT_BUFFER)
```

```

18     if move_y_val > 0:
19         self.drone.move_down(move_y_val)
20         time.sleep(move_y_val/10 + self.COMMAND_WAIT_BUFFER)
21         print('sleep_y: ', move_y_val/10 + self.
                COMMAND_WAIT_BUFFER)
22     else:
23         self.drone.move_up(-move_y_val)
24         time.sleep(-move_y_val/10 + self.COMMAND_WAIT_BUFFER)
25         print('sleep_y: ', -move_y_val/10 + self.
                COMMAND_WAIT_BUFFER)
26
27     if move_z_val > 0:
28         self.drone.move_forward(move_z_val)
29         time.sleep(move_z_val/10 + self.COMMAND_WAIT_BUFFER)
30         print('sleep_z: ', move_z_val/10 + self.
                COMMAND_WAIT_BUFFER)
31     else:
32         self.drone.move_backward(-move_z_val)
33         time.sleep(-move_z_val/10 + self.COMMAND_WAIT_BUFFER)
34         print('sleep_z: ', -move_z_val/10 + self.
                COMMAND_WAIT_BUFFER)
35
36     if rotate_val > 0:
37         self.drone.rotate_cw(rotate_val)
38     else:
39         self.drone.rotate_ccw(-rotate_val)
40     time.sleep(self.COMMAND_WAIT_BUFFER)
41     self.is_moving = False
42     self.is_standby = False
43     self.drone_pos_list = []
44     self.drone_rotate_list = []
45     return

```

5.3 映像部

今回は SDK として用意されていた libh264decoder という C++ のプログラムを利用した。また、そこから得られた画像を利用し易い形に以下プログラム 5.2 を用いて利用した controller では tello SDK で用意されている tello クラスを継承して利用している。

ここでは self.drone という変数で参照が可能となっている

プログラム 5.2: controller.py

```

1 def video_loop(self):
2     try:
3         time.sleep(0.5)
4         while not self.stopEvent.is_set():
5             frame = self.drone.read()
6             if frame is None or frame.size == 0:
7                 continue
8             self.pnp_qr(frame)
9     except (KeyboardInterrupt, SystemExit):
10         self.stopEvent.set()
11         print('exit_video_loop')

```

5.4 トラッカー部

トラッカーノードの実装には OpenCV を用いる。

QR コードの検出とデコードには OpenCV に用意されている DetectQR クラスを利用し，そこから得られた QR コードの四角に対して PnP による 3 次元再構成を行った．PnP には OpenCV に用意されている solvePnPRansac 関数を利用した。

これにより QR コードよりデコードされた次の QR コードの位置情報と QR コードに対するカメラの回転ベクトル，並進ベクトルが得られるのでこれらの情報をナビゲーション部 5.5 にて処理をする

実際にスマートフォン上に表示した QR コードに対して姿勢推定結果を描画したものを以下の図 5.1 に示す



図 5.1: QR コードの姿勢推定結果

ソースコードの該当部分をそれぞれ以下のプログラム 5.3 に示す．本コードでは QR コードの認識時にノイズが乗る事が多く影響を減らす為に 20 フレーム分バッファを保持するようにし，その平均値を用いて制御を行なっている．

プログラム 5.3: pnp_qr.py

```

1 def pnp_qr(self, frame):
2     img = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
3
4     qr = cv2.QRCodeDetector()
5     data, points, _ = qr.detectAndDecode(img)
6     if data:
7         _, origin_rvec, origin_tvec, inliers = cv2.
            solvePnPRansac(self.objp, points, self.mtx, self.
            dist)

```

```
8         if not self.is_moving:
9             self.target_pos = json.loads(data)
10            tvec = origin_tvec * self.UNIT_SIZE
11            rvec = origin_rvec * self.RAD_UNIT
12            self.drone_pos_list.append(tvec)
13            self.drone_rotate_list.append(rvec)
14            if len(self.drone_pos_list) > 20:
15                self.drone_pos_list.pop(0)
16                self.drone_rotate_list.pop(0)
17            drone_avg_pos = np.mean(self.drone_pos_list, axis
18                                   =(0))
19            drone_avg_rotate = np.mean(self.drone_rotate_list,
20                                       axis=(0))
21            self.drone_pos = {
22                'x': drone_avg_pos[0],
23                'y': drone_avg_pos[1],
24                'z': drone_avg_pos[2],
25                'r': drone_avg_rotate[1]
26            }
27
28            imgpts, jac = cv2.projectPoints(self.axis, origin_rvec,
29                                           origin_tvec, self.mtx, self.dist)
30            img = self.draw(img, points, imgpts)
31
32            cv2.imshow('img',img)
33            cv2.waitKey(10)
34
35            else:
36                cv2.imshow('img',img)
37                cv2.waitKey(10)
```

5.5 ナビゲーション部

QR コードから得られたカメラの情報と次点の QR コードの位置情報を元にドローンに対してメッセージを送信する。

第6章 評価

本章では，提案システムの評価について述べる．

6.1 評価方法

倉庫内を模した周回コースを作成し，QR コードの数を変化させた際の周回成功率と速度を計測する．QR コードの数を増やせばコース上の任意の QR コードから次点の QR コードまでの距離も短くなり，カメラの計測誤差等による経路誤差も小さくなり周回成功率も向上すると考えられる．一方で本システムでは QR コードを認識するとそこで一時停止する為，QR コードの数を減らせば次点へ向かうまでの時間が短くなり周回速度が向上すると考えられる．

第7章 関連研究

基本的にドローンが自律飛行を行う際に必要とされる処理がいくつかある。そのうちの2つが自己位置推定と姿勢推定である。自己位置推定とは、ドローンが飛行している空間の中で自分がどの位置にいるのかを認識する事である。姿勢推定とは自分が水平方向に対してどの程度傾いているのか、鉛直方向に対してどの程度回転しているかといった事を認識することある。

7.1 Nanomap

これらを行なっている関連研究として Nanomap[1] という飛行モデルがある。においては 2D-LIDAR を用いた飛行経路探索手法が取られており、取得した点群データを過去数フレーム分を記憶しておき、過去の数フレーム分のデータと現在のフレームとの点群の変量によっておおよその障害物の位置を認識している。こちらの手法では、物体を正確に捉えることは放棄し、不確実な部分を物体の存在している可能性のある範囲として捉えている。視野にある物体のある可能性も含めて1番物体が少ない方向を飛行経路として選定し、飛行している。

7.2 Octomap

他に、ドローンに限らずロボット工学全般で利用されてきた Octomap[2] がある。こちらも Nanomap と同様に LIDAR を用いて周辺的环境情報を扱うものであるが、こちらは LIDAR を用いて実際に周囲を点群からモデリングして周辺状況を把握する。

Octomap ではモデリングする際に近い点群同士を同じ物体としてまとめ1つのブロックにするということを繰り返し、樹構造的にブロックを生成する為、生成後のデータは繰り返す回数等のパラメータによっては小さく圧縮することもできる。しかし、実際に点群からモデルを生成するとなるとその計算量は非常に多く、高性能な小型コンピュータが現れている現在においても処理負荷が大きく、特にドローンにおいては積載量に制限がかかりやすい為扱いにくいものとなっている。



図 7.1: 圧縮率の違いによる生成物の違い

7.3 実用性を備えた手のひらサイズ・完全オンボード処理 UAV

また、これらのように LIDAR を使用せず、CMOS センサーのみで自律飛行を行なっている例もある。

実用性を備えた手のひらサイズ・完全オンボード処理 UAV のための 3 次元自己位置推定手法の提案と全自動飛行の実現では自己位置推定で使用するセンサーは CMOS センサーを利用した自律飛行を実現している。CMOS センサーからの映像を FPGA にストリーミングして FAST による特徴点抽出処理を行い、その結果を Brief を用いて表現し、MCU に流し込み処理している。この研究での目的は外部処理系に依存せず、ドローン上で全て完結する自律飛行可能な小型ドローンの開発で、限定条件下 (新聞紙を敷き詰めた床面) の上で飛行させ、特徴点を十分に検出できるようにした上で、飛行させ想定通りの飛行を実現している。

7.4 DeepDrone

他に今回の研究のベース物として DeepDrone が挙げられる。こちらではドローンレースを自律飛行で行う為に自律飛行モデルを提案している。ドローンレースとは基本的にゲートを順に潜りながら飛行することが条件となっているドローン競技である。このゲートを RGB カメラで撮影した映像を推定モデルに対し、 200×300 の画像を所与として $\{\vec{x}, v\}$ を推定結果として得る。 $\{\vec{x}$ は $\vec{x} \in [-1, 1]^2$ と定義され正規化された入力画像の中にある目標とするゲートへの方向を表していおり、 $v \in [0, 1]$ は飛行速度を正規化して表している。

DeepDrone では訓練にシミュレータでのデータと現実世界でのデータを使っている。両者とも理想とする軌道とのズレを計算しそのズレを損失関数に入力し学習を進めていく。シミュレータ上では直接理想とする軌道とのズレを取得し、実空間では実際に手動で実際のコース上を運び、ゲートを通過させるなどしてデータを収集していた。

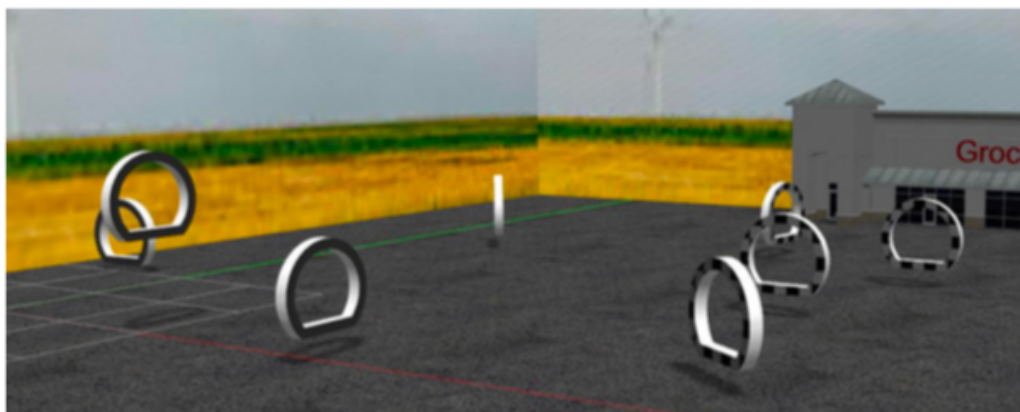


図 7.2: シミュレータ上のゲート

7.5 TDOA を用いた UAV の自己位置推定

他に屋内でドローンの位置測位を行っている例として Scalable and Precise Multi-UAV Indoor Navigation[3] が挙げられる．これは TDOA-UWB という方式を用いたものである．TDOA(Time Differences of Arrival) とは電波を用いた位置推定手法であり TOA(Time of Arrival) とよく比較される．Time of Arrival では電波を発信する送信機と位置推定を行う受信機との構成で行われる．送信機と受信機で時刻同期を行った上で、各送信機からの伝播時間を計測．この電波時間から距離を計算し、複数の送信機を用いると位置推定を行う事が出来る．

一方で TDOA では時刻同期の必要はなく、送信されてくる電波の時間差を用いて位置推定を行う．今回挙げているこの例では TDOA による位置測位を行うのに UWB を電波として用いている．

第8章 結論

本章では，本研究のまとめと今後の課題を示す．

8.1 本研究のまとめ

今回のこのシステム実際に使う上では間違っって他の QR コードを認識してしまったり，悪意ある人間によって QR コードがすり替えられていないかの検証システムが必要になってくる．その場合には jwt などの方式で QR コード内のデータに署名をする．という事で対応できるが，悪意ある人間がその環境に存在した場合 QR コードの場所が置き換えられる可能性や複製されてしまうと言う可能性は残る．

また，QR コードを LED パネルで表示する方法 [4] も考案されている為この方法を用いて QR コードを容易に変更・配置の記録をする事も可能になる．一方でどこにどの QR コードが配置されているかはシステム全体として保持しておきたい情報でもあり，LED パネルで QR コードを表示する手法は本研究との親和性が非常に高いと考えられる．

8.2 本研究の課題

付 録 A 付録

A.1 付録内容

謝辭

参考文献

- [1] Peter R. Florence¹ et al. Nanomap: Fast, uncertainty-aware proximity queries with lazy search over local 3d data. arXiv, 2018.
- [2] Hornung et al. Octomap: An efficient probabilistic 3d mapping framework based on octrees. A.Hornung and K.M.Wurm and M.Bennewitz and C.Stachniss, and W.Burgard, 2013.
- [3] Tiemann Janis and Wietfeld Christian. Scalable and precise multi-uav indoor navigation using tdoa-based uwb localization. International Conference on Indoor Positioning and Indoor Navigation, 2017.
- [4] Ukida Hiroyuki, Miwa Masafumi, Tanimoto Yoshio, Sano Tetsuya, and Hideki Yamamoto. Visual uav control system using led panel and on-board camera. IEEE International Instrumentation and Measurement Technology Conference (I2MTC), 2013.