

## 1. UWB+オプティカルフローセンサの構成で位置制御

以前より推定アルゴリズム、位置制御アルゴリズムを構築していた UWB + オプティカルフローセンサを用いた構成にてドローンの位置を制御する実験を行った、内容は以前にも行った原点でのホバリングである。その結果を fig. 1 に示す。原点からの距離の標準偏差は 0.113 m, 最大誤差は 0.625 m となった。この数字は以前に UWB のみで飛行させた時 (0.083 m, 0.406 m) よりも悪くなってしまっている。理由は今の所わからないが、以前の飛行時間の 1 分に比べ今回は 4 分と長時間飛ばしたことやオプティカルフローセンサが搭載されたことで、今までは時間分解能が粗く、計測できていなかった移動距離まで計測できるようになったためなどが理由として考えている。

体感としては、以前 (オプティカルフローセンサ未搭載) は UWB の値が飛行中に揺らぐため、ある程度は位置をとどめてくれるが、ふらついた飛行であった。しかし、今回の実験ではそういったことが少なくなり、比較的安定したホバリングであったと感じる。カルマンフィルタの分散をもう少し調整し、観測値をオプティカルフローセンサ側に寄せても良い感じがする。

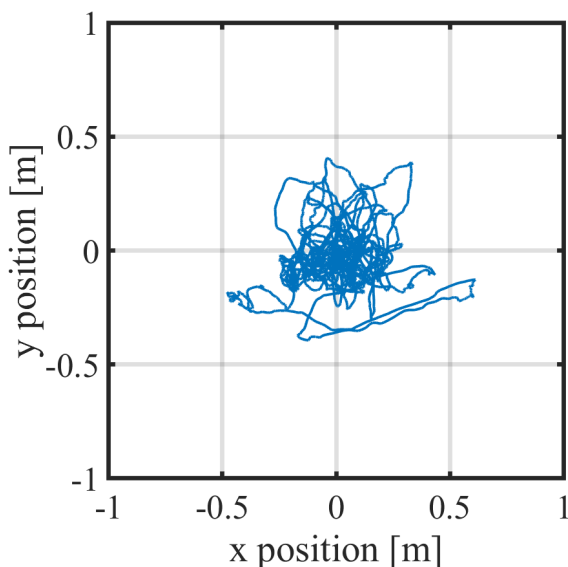


Fig.1: オプティカルフローセンサの値も用いた場合の原点ホバリング結果

## 2. ステレオカメラの件

前回、ステレオカメラを同期させるには同期用のケーブルが必要であると述べたが、そのケーブルの存在を確認した。それを用いて動画 (静止画) 撮影の同期を取れるようアプローチしてみた。手法としては、どちらかのカメラをプライマリカメラとして設定し、もう片方をセカンダリカメラとして設定する。そして、セカンダリカメラは同期モードにし、プライマリカメラから同期用の割り込み信号を出し、それをセカンダリカメラが拾う形で 2 台間の同期をとることが出来る。実際に行ったところ、確かに開始のタイミングは完全に合わせることが可能であったが、2 代とも 100fps で設定したにも関わらず、2 台間のフレームレートが全く合わず、プライマリカメラは 100fps で撮影できているが、セカンダリカメラは 60fps でしか撮影できないという結果を得た。恐らく CPU の処理の限界の問題だと考える。

次に 2 台間の同期は初めから諦め、静止画を 100fps で撮影できるようアプローチした。以前までは 100fps で撮影した動画を Python の機能で分割することで一秒間に 100 枚の静止画を得ていた。しかし、この静止画にする作業に非常に時間を要していた (1 分の動画で数 10 分) ため、100fps で静止画を撮影できるようにした。初め、両カメラで 100fps で撮影したところ、スキップされるフレームがとて多く (7 フレームに 1 まいくらい)、とても 100fps では撮影できなかった。しかし、これの原因は画像の解像度を高くしてしまっていたためであり、以前の画質を落とした設定に戻せば問題なくフレーム落ちすることなく、100fps にてカメラ 2 台で画像を撮影することができた。設定が変わってしまった原因はカメラ同期をするために設定をいじっている際に画質の設定まで変わってしまったためである。また、飛翔部屋の PC の Windows に Python 環境を構築したため、今までは別の PC の環境で行っていた動画の分割や位置座標を求める計算などを全て 1 つの PC 内で完結できるようになった。また、画像の記録を HDD から SSD へと保存先を改めたことでも大幅な作業効率アップができた。

ここで先の同期の話に戻るが、2 台のカメラで 100fps で問題なく撮影できたため、画質の設定を見直せば、きちんと同期が取れた状態で 100fps で撮影が可能かもしれない。同期が取れていなくても Python のプログラム側で同期が取れるよう組んであるため、これ以上の検証は修論が終わってからに持ち越したいと思う。

## 3. SII のスライド作り

来週も発表練習があるため、今週ご指摘頂いた点をきちんと修正した後臨みたいと思う。

## 4. 今後の予定

- ・ 真値をロギングした状態（ステレオカメラを用いて）で以前の制御アルゴリズム，開発した制御アルゴリズムで飛行実験
- ・ 修論のためのデータ集め，まとめ