システム制御特論

21P2033 山田竜輝

**学術論文テーマ**

　シミュレーション環境でのLidarセンサを用いたUAVの自動衝突回避システムの実装と評価

**論文詳細**

　この論文では、シミュレーション環境（Gazeboソフト）で、UAV modelを設定し、外部LidarセンサをUAV modelに統合し、UAVの自動衝突回避システムの実装を行う。また、シミュレーション環境で、無事UAVが障害物の衝突を回避できたかの検証を行う。

**実験補足情報**

　実験補足情報として、今回の論文内容を説明するにあたり、説明するべき単語や補足情報を下記項目ごとにまとめた。

* UAV（搭載されている機器含め）

今回シミュレーション環境（Gazebo）で使用するUAVは図1である。図1のUAVに関しては、Gazeboソフトが提供しているUAVであり、Irisクアッドコプターモデルと呼ばれている。

　Irisクアッドコプターモデル構成に関して、フライトコントローラはPixhawk(図2)を使用しており、加速度センサ、ジャイロセンサ、地磁気センサ、気圧センサが内蔵されている。また、外部センサとして、GPSが含まれているモデルとなっている。重量やプロペラ、モーターに関して、詳細不明である。（\*論文で確認出来ず。）

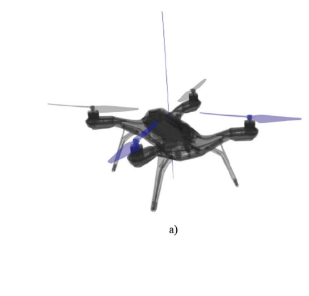


図1 Irisクアッドコプターモデル



図2 フライトコントローラ(Pixhawk)

* **Lidarセンサ**

今回使用するLidarセンサは、Hokuyo URG-04LXである。Hokuyo URG-04LXの詳細に関して、箇条書きとして、以下にまとめた。

* + レーザークラス1 λ=785nm
  + 範囲、20〜10,000 mm
  + 精度、60～1000mm；±30mm
  + 1000mmから10000mmまで ±3
  + スキャンタイム、100ms
  + 電源、DC5V + - 5
  + 重量, 160 g

使用する理由は、信頼性の高さ、ペイロード（UAV重量）のバランスが取れるからである。

　　　上記の詳細パラメータをGazeboソフトで、パラメータ設定を行い、Iris クアッドコプターの外部センサ（Lidar）として、フライトコントローラ（Pixhawk）に統合する。

* **シミュレーション環境（Gazeboについて）**

　今回シミュレーション環境として、Gazeboを使用した。

　使用した理由に関して、フライトコントローラ（Pixhawk）をサポートしているからである。将来現実環境で、UAVの衝突回避システムを実装するにあたり、フライトコントローラ（Pixhawk）を使用するためである。

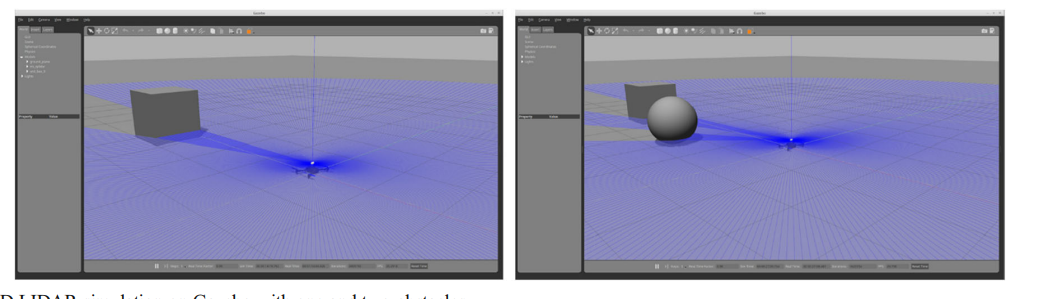


図3 シミュレーション環境（Gazebo）

* **Q Ground Control**

　Q Ground Controlに関して、サポートされているフライトコントローラを搭載しているUAVの飛行経路を設定、自立飛行することができ、設定情報をフライトコントローラに書き込むことができるソフトである。また、UAVの飛行経路をMAPとして可視化することが可能である。今回使用するPixhawkはサポートされているフライトコントローラである。

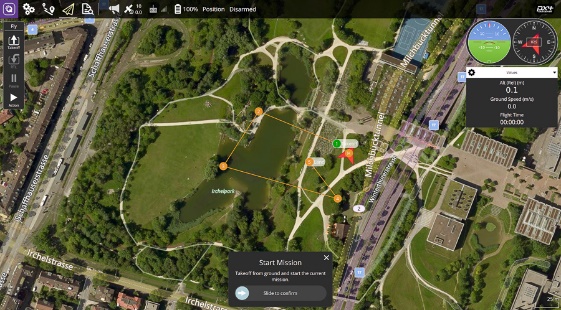


図4 Q ground Control

* **MATLAB/Simulink**

　MATLAB/Simulinkに関して、センサ情報などをグラフとして表示し、センサ情報の分析、衝突回避システムの評価をするためのツールで使用する。

* **Pixhawkによる衝突回避システムの処理内容について**

　Pixhawkによる衝突回避システムの処理内容に関して、図5を見ていただきたい。衝突回避システムに関して、Iris クアッドコプターモデルのPixhawkフライトコントローラ内部で処理される。

　今回の衝突回避システムのプログラム場所は、図5のObstacle Avoidance Moduleである。シミュレーション環境のもと、Lidarセンサで障害物との距離をObstacle Avoidance Moduleで管理し、UAVと障害物で衝突しそうな距離になる際に、Obstacle detection＆trackingブロックでトリガーを生みだし、回避させるための提案をEvasion logicにする。Evasion logicで、EKFの処理後のUAVの位置情報や姿勢情報を入力値として、回避処理を行う流れとなっている。

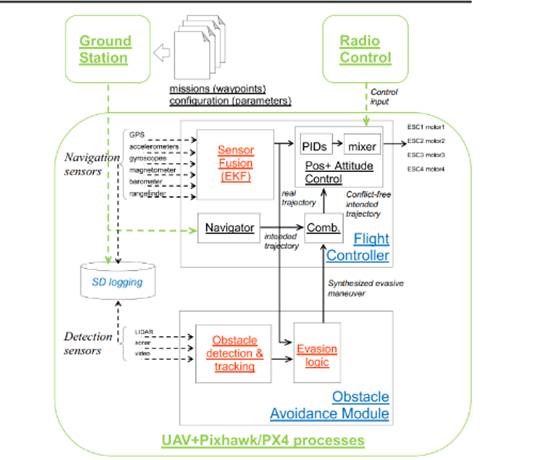


図5 Pixhawkの内部処理に関して

**実験手順**

　初めに、GazeboソフトでIrisクアッドコプターモデルを選択。選択後、衝突回避システム実装のために、今回使用するLidarセンサ情報をGazeboソフトで入力し、フライトコントローラ（Pixhawk）に統合する。そしてLidarセンサによる衝突回避システムをプログラミングし、統合する。次に、統合がうまくいっているかどうかをGazebo simulatorで確認（図5）。その後、衝突回避システムのための障害物（家）をGazebo Simulatorで用意する。(図6)

　次にQ ground Controlソフトを立ち上げ、Irisクアッドコプターの飛行経路を生成する。（図7）今回一番左側を出発点として、直線経路を作成している。飛行経路をPixhawkに書き込みを行う。

そしてGazeboシミュレーターで実行し、衝突回避の実装確認を行う。

　最後に、Gazeboシミュレーターでの実行終了後、シミュレーター環境で得られたセンサデータをMATLAB/Simulinkソフトに反映させ、グラフ化し、衝突回避システムの性能評価を行う。

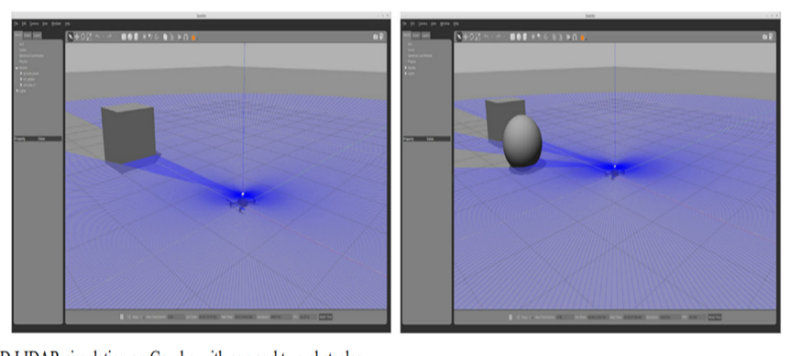


図6 Gazebo simulatorによるLidarセンサの実装確認

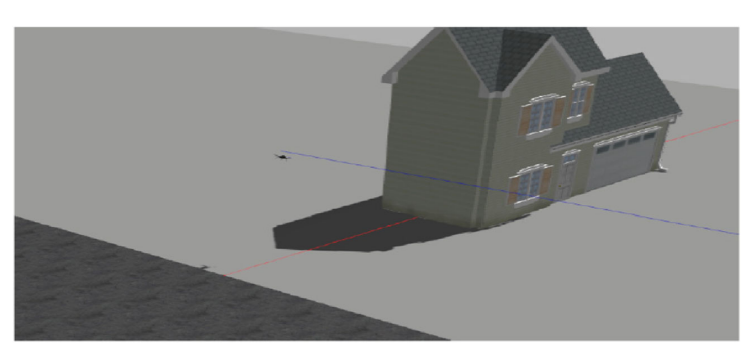


図7 Gazebo Simulatorによる障害物生成



図8 Irisクアッドコプターの飛行経路

**実検結果**

　初めにシミュレーション終了後のQ groud controlでの飛行経路結果が図9である。赤丸印が障害物（家）である。図9からわかるように障害物を回避したことがわかる。



図9 Q groud controlでの飛行経路結果（障害物回避）

次に直交座標系での衝突回避時の機体の速度状況をMATLAB/Simulinkソフトでグラフにした結果である。横軸は時間、縦軸がx,y,z速度となっている。（図10(a),(b)）比較対象として、障害物がない時のグラフも記載する。

　図10(a)に関して、障害物がない場合、離陸からミッション終了までの速度が一定である。速度の鉛直成分では、最初にUAVが上昇する際に変化しており、それ以降は一定である。

図10(b)に関して、UAVがLidarセンサで障害物を検知すると、衝突回避のためにブレーキとして、機体の速度を0もしくはマイナスになる。衝突回避後、機体の速度が0以上となり飛行目標経路に向かう。

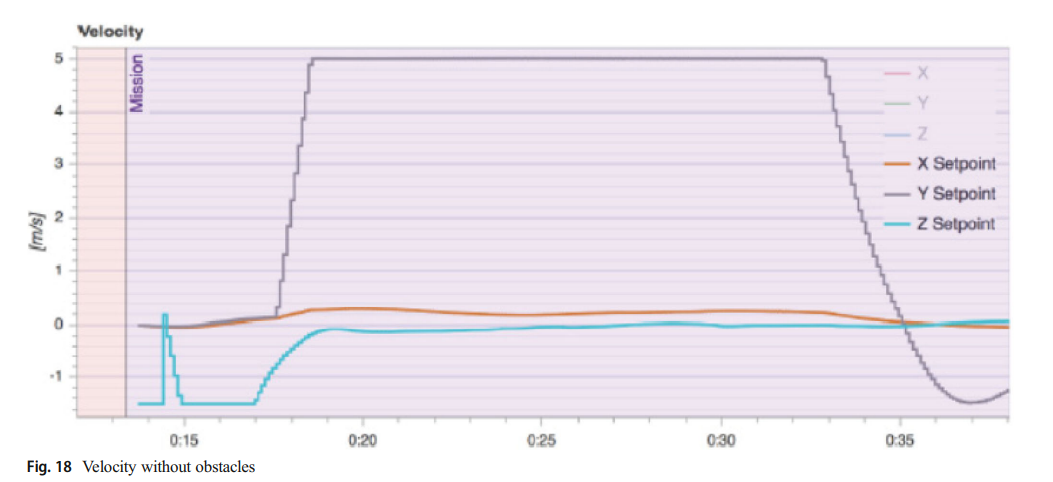


図10(a) 障害物がない時の機体の速度

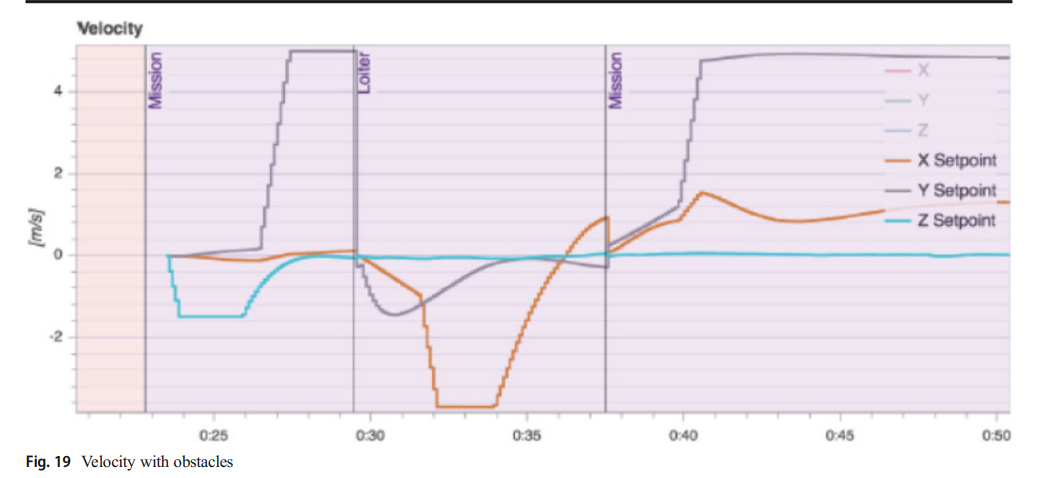


図10(b) 障害物（家）がある時の機体の速度

最後に機体のピッチに関してのグラフが図11(a),(b)である。横軸は時間、縦軸が角度となっている。赤線は拡張カルマンフィルタで推定されたピッチである。図11(a),(b)の初めの時間、UAVの鉛直離陸のためピッチが一定になる時間がある。図11(b)では、衝突回避システムが起動すると、正の値までピッチが増加し、ドローンを停止させ、0で安定し、再びピッチ上昇し、障害物を回避するようなグラフとなった。

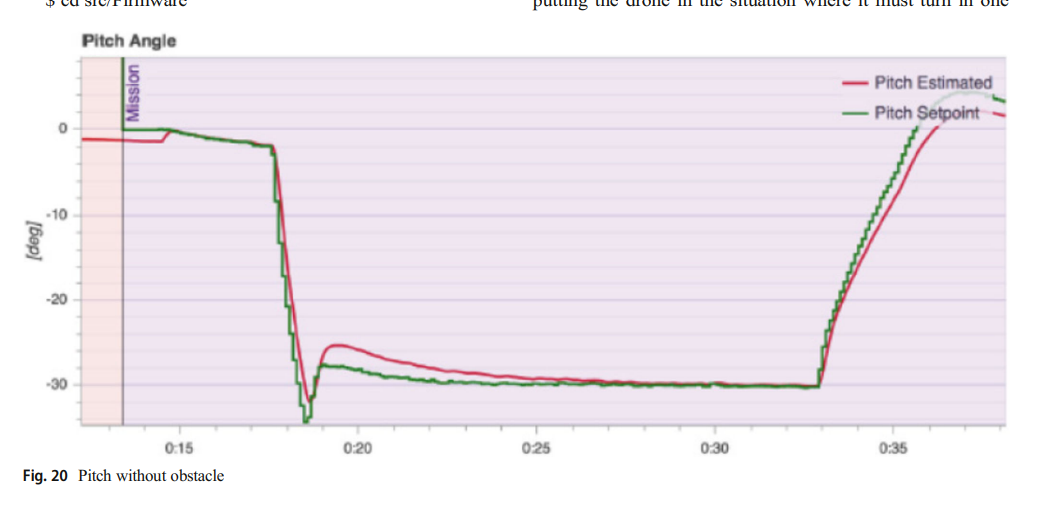


図11(a) 機体のピッチ（障害物なし）

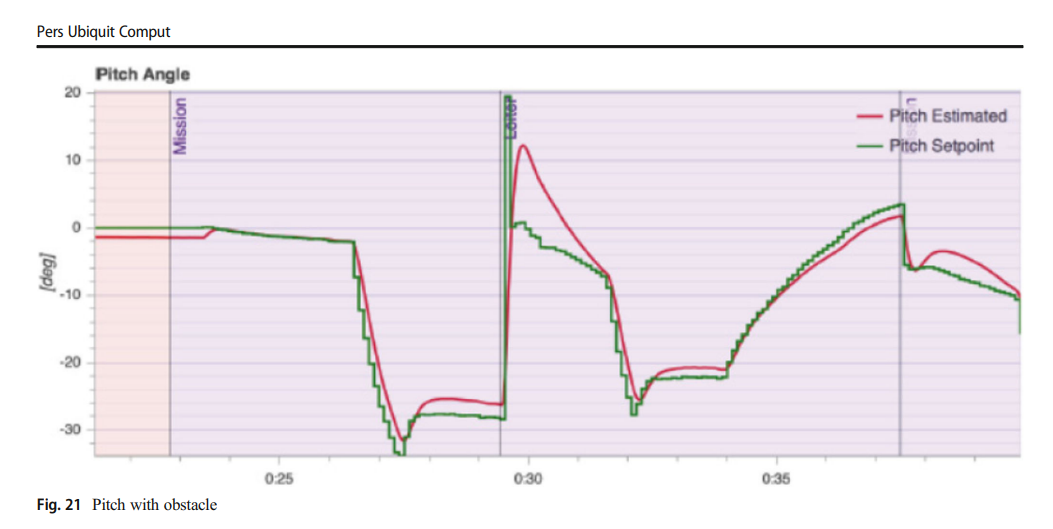


図11(b) 機体のピッチ（障害物あり）

**自分の考察と思ったこと**

　考察として、衝突回避システムを実現するにあたり、UAVの高度情報がかなり重要になってくると考えられる。

　理由として、図12のLidarセンサの検出範囲を拝見した際に、Lidarセンサの縦軸の検出範囲がかなり短いのではないかと考えたからである。

今回の実験ではLidarセンサをUAVの上に搭載されていた。（図12）そして、UAVの高度に関して、今回障害物の高さ以下であり、障害物の衝突回避を行っていた。もし、UAVの高度が、図12の赤線を少し上回った高度の場合、Lidarセンサは障害物を認識せず、UAVの下側部分が障害物にあたる問題が発生する可能性がある。

　上記の解決方法として、Lidarセンサの位置を変更するか、Lidarセンサをもう一つ増やし、UAVの下側に取り付ける方法があるのではないかと思った。もう一つ取り付けるにあたり、上側のLidarセンサの光の干渉が起きないように気を付ける。

　また、UAVにミラーを搭載し、Lidarセンサの光の一部を屈折させ、Lidarセンサの縦軸の検出範囲を増やすことが可能になるのではないかと考えられる。

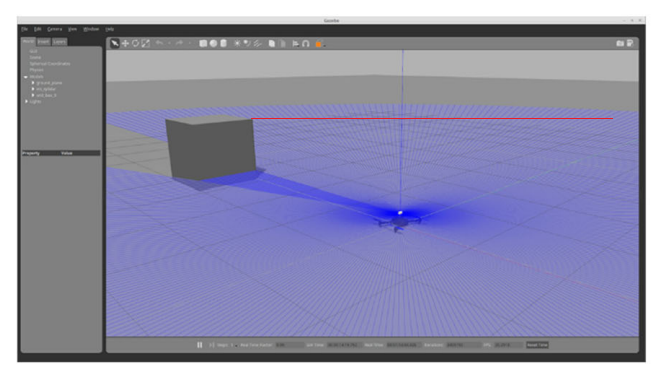


図12 Lidarセンサ取り付け位置・センサ範囲