주간연구일지 수정내역

1. 드론 인식 및 추적을 위해 openCV를 사용하기로 결정
2. 십자 마커를 이용해 드론 인식, 추적하기로 결정
   1. openCV의 findContours 함수를 이용해 윤곽선 추출
   2. openCV의 approxPolyDP 함수를 이용해 윤곽선 단순화, 꼭짓점이 12개인 윤곽선 선정
   3. 마커의 기준점은 이미지모멘트를 사용해 계산된 중심을 사용하기로 결정
3. 마커의 자세는 openCV의 solvePnP 함수로 추정하기로 결정
4. 마커가 카메라에서 멀리 떨어진 경우, 인식이 잘 되지 않는 문제 발생
   1. 십자 마커가 12각형이 아니라 8각형으로 인식됨을 발견
   2. approxPolyDP 함수는 Ramer-Douglas-Peucker 알고리즘을 사용해 윤곽선을 단순화
   3. Ramer-Douglas-Peucker 알고리즘은 두 점 사이의 직선에서 가장 멀리 떨어진 점의 거리가 기준거리보다 작을 경우 직선으로 단순화하는 알고리즘
   4. 마커가 카메라에서 멀리 떨어저 카메라 상 마커의 크기가 작아질 경우 십자의 오목 꼭짓점과 8각형 간의 거리가 기준거리보다 작아짐
   5. 마커의 형상을 변경하기로 결정
5. 십자 마커 대신 화살표 모양 마커를 사용하기로 결정
6. 화살표 형상이 단순해 마커 외의 다른 윤곽선이 인식되는 문제 발견
   1. 검은 사각형 안에 흰 화살표를 가지는 마커로 변경해 조건 강화
   2. 그럼에도 다른 윤곽선이 인식됨
   3. openCV의 convexHull 함수를 사용해 윤곽선에서 볼록선체 생성, convexityDefects 함수를 이용해 윤곽선과 볼록선체를 비교, 볼록결함 찾아냄
   4. 볼록결함의 개수가 2개이고, 각 볼록결함의 시작점과 끝점이 일치하지 않고, 각 볼록결함 간의 인덱스 거리가 2인 윤곽선만을 인식하도록 조건 강화
7. 마커 외 오인식이 많이 줄어들었지만, 그럼에도 가끔씩 오인식 발생
8. 자료 수집을 통해 자세 추정을 위해 사용되는 ArUco 마커 발견
   1. ArUco마커의 사용에는 ArUco 마커 단일, 마커 보드, ChArUco 보드 등의 다양한 파생형이 있음
   2. 실제로 인식, 자세 추정 해보니 Aruco 마커 보드의 성능이 가장 적합
      1. ArUco 마커 단일 사용의 경우
         1. 마커의 크기가 커 먼 거리에서 인식 가능
         2. 마커의 일부분이 가려지면 인식 불가능
      2. ArUco 마커 보드 사용의 경우
         1. ArUco 마커 여러 개를 묶어 하나의 마커로 간주
         2. 마커보드의 일부분이 가려저도 인식 가능
         3. 마커의 개수가 많아지면 먼 거리에서 인식 어려워지지만, 4개의 경우 먼 거리에서도 인식 가능
         4. 꼭짓점의 개수가 많아 solvePnP 함수로 자세 추정시 더욱 정확해짐
      3. ChArUco 보드의 경우
         1. 체스보드에 ArUco 마커를 결합한 마커
         2. 형상이 가장 복잡해 먼 거리에서 인식 어려움
9. solvePnP 함수를 이용한 자세추정에서 다음과 같은 문제 발견
   1. 마커가 카메라에 평행할 시 추정 자세 진동
   2. 인식된 마커 꼭짓점과 자세가 일치하지 않음
10. solvePnP 함수의 flag를 변경해 PnP 방법 변경
    1. 각 flag를 적용해본 결과, IPPE 방법이 가장 안정적이고 빨라 사용하기로 결정
       1. 다른 방법은 비평면 목표를 가정하고 있는 방면, IPPE 방법은 평면 목표만을 가정함
11. 자기 자세 추정을 위해 6축 자이로 센서 MPU6050을 사용하기로 결정
    1. 자이로 센서로 자세 추정시 오차 누적, 가속도 센서로 자세 추정시 부정확한 문제를 해결하기 위해 Kalman filter로 센서 융합하기로 결정
    2. python을 이용해 센서 값 획득, Kalman filter 함수 작성해 적용
12. 제어신호를 드론으로 전송하기 위해 nRF24L01 모듈을 통해 2.4GHz 통신을 사용하기로 결정
13. 제어신호를 해석하기 위해 수신기를 먼저 개발하기로 결정
    1. 자료 수집 결과 아두이노용 코드만 발견, 아두이노용 c++ 코드를 라즈베리파이용 python 코드로 포팅
14. 수신기로 수신한 조종기의 제어신호를 해석, 발신기를 개발
    1. 쓰로틀 신호는 0~255, 요, 롤, 피치 신호는 중립시 0, + 신호시 128~255, - 신호시 1~127이 됨을 발견, 발신기에 적용
    2. 드론은 드론 조종기에서 발신되는 바인딩 패킷을 통해 조종기 특정, 바인딩 패킷의 첫 5개가 조종기의 주소(161, 105, 1, 104, 204)가 됨을 발견, 발신기에 적용
    3. 바인딩이 성공적으로 진행되었지만, 드론이 제어신호에 응답하지 않음
       1. 자료 수집 결과 사용한 드론 syma x8은 주파수 호핑을 사용, 주파수 채널을 주기적으로 변경
       2. 수신기를 사용해 조종기의 신호를 모든 주파수 채널에서 수신, 각 채널에서 신호가 감지된 횟수를 파악해 가장 많이 감지된 4개의 채널(22, 38, 54, 70) 선정
15. 드론의 idle, 시동 신호를 파악, 적용(idle: [throttle=0, pitch: 255, yaw: 255, roll: 127], 시동:[ >127, 0, 0, 0])
16. 제어기 설계 시작
    1. 드론의 시스템을 파악, 드론은 MIMO(Multi Input, Multi Output) 시스템
    2. 드론을 제어하기 위해 상태 피드백 제어를 사용하기로 결정
    3. 자료조사 결과 드론의 상태공간모델 찾음
    4. LQR을 이용해 상태 피드백 제어 게인 계산
17. 마커 인식, 마커 자세 추정, 자기 자세 추정, 제어기, 제어신호 발신기를 한 프로그램에 통합
18. 드론에 다음과 같은 문제 발견
    1. 제어신호가 수렴하지 않고 진동함
       1. LQR 게인 K를 적절하게 튜닝할 필요
       2. LPF필터 등을 적용해 측정신호, 제어신호 잡음 제거 필요