고가치 표적의 타격 효율 극대화를 위한 정찰 및 감시용 드론 체계 연구

-실시간 적 표적 3D 좌표 송신 및 실시간 자율주행 타격능력 개발-

육군사관학교 정용한
해군사관학교 황서진
포항공과대학교 이세규
포항공과대학교 신용희
대구경북과학기술원 이석민

Abstract

전투 상황에서 드론을 활용한 적군 고가치 표적의 파괴는 경제적 측면에서 효율적인 행위이며 전술적 우위를 달성하는 데에 도움을 준다. 본 연구는 이러한 전술적 중요성을 바탕으로 탱크나 적 지휘관과 같은 고가치 표적의 효율적인 타격을 위한 드론 체계를 개발하였다. 먼저 드론은 적의 핵심 지역을 탐색하여 고가치 표적을 정밀하게 정찰한다. 표적이 탐지되면, 드론은 해당 표적의 3D 좌표를 실시간으로 본부에 전달하며, 필요한 경우 자율주행 기능을 활용하여 직접 표적에 돌진, 자폭 공격을 수행한다. 실시간으로 본부에 전송된 3D 좌표 정보는 아군 포병 전력을 비롯한 원거리 무기체계의 작전 수행 능력 향상과 지휘관의 상황판단 및 결심 보조자료로서 활용될 수 있다. 이 드론 체계를 통해, 적 고가치 표적에 대한 정찰 및 감시뿐만 아니라, 자폭 공격 및 원거리 공격을 통해 고가치 표적을 효율적으로 제거하여 아군의 작전 수행에 크게 기여할 수 있을 것이다.

1. 서론

미래전 양상과 미래 전장 환경 변화에 따라 전쟁에서 인명 손실을 최소화하고 전투의 효율성을 높일 수 있는 드론의 필요성이 증대되고 있다. [1]에 따르면, 드론은 전투 요원들이 직접 수행해야 할 위험한 작전을 대신할 수 있는 미래의 무인전투체계로써 감시 및 정찰, 전투, 통신, 화생방 정찰 및 탐지, 보급품 수송 등 군사적으로 다양하게 활용할 수 있다.

2022년에 발발한 우크라이나와 러시아의 전쟁은 전장에서 드론의 중요성을 강조하기에 충분한 예시일 것이다. 우크라이나가 러시아의 탱크와 핵심 시설들을 스위치블레이드, TB2 등의 드론으로 파괴한 일례는 드론의 전장에서의 파괴력을 입증한 명백한 사례이다.

본 연구에서는 적지에 투입되어 고가치 표적을 정찰 및 감시하고, 더 나아가 자율주행을 통해 표적에 자폭 공격을 수행할 수 있는 드론을 개발하였다. 고가치 표적을 정찰하기 위해 YOLO v8 알고리즘을 이용하여 탱크, 차량, 사람 등을 인식할 수 있도록 하였고, ByteTrack 알고리즘을 이용해서 특정 객체만을 표적으로 설정해 추적할 수 있도록 설계하였다. 표적의 3D좌표는 [2]에서 소개한 알고리즘을 참고해 변형했으며, 표적을 추격하기 위해 PID 제어를 이용했다.

본 연구의 기여점은 다음과 같다.

- 감시·정찰 임무를 수행하는 전투 요원의 접근이 제한되는 지형, 지역에서의 연속성 있는 임무 수행이 가능하므로 양질의 첩보를 지속해서 획득할 수 있게 된다.
- 더불어 원격제어를 통한 전투원의 생존 가능성 향상에 크게 기여할 수 있다.
- 실시간 3D 좌표 송신을 통해 적 고가치 표적의 정보를 전달함으로써 아군 화력 체계의 정확성과 신속성을 향상하여 전술적 우위를 확보할 수 있다.
- 화력 체계뿐만 아니라 적 고가치 표적의 실시간 3D 좌표를 본부로도 전파하여 지휘관의 상황인식과 결심을 보조할 수 있으며 더 나아가 지휘관의 전장가시화에 기여할 수 있다.
- 자율주행 타격 능력을 활용한 드론의 정밀타격은 적 고가치 표적을 처리하는 기존의 자산보다 매우 높은 수준의 경제성을 지니므로 효율적인 전투에 기여할 수 있다.

2. 본론

2.1 이론적 배경

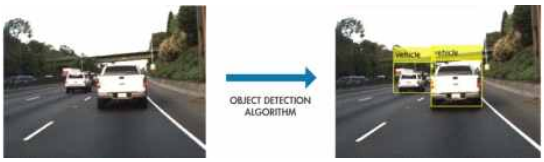
2.1.1 Tello Edu Drone

<그림 1 : tello edu drone>

본 연구에서 사용한 드론은 DJI사의 Tello Edu 드론이다. 드론이 만드는 와이파이에 컴퓨터를 연결하기만 한다면, 파이썬을 이용하여 드론을 조종함과 동시에 드론의 전방 영상을 실시간으로 볼 수 있는 것이 큰 장점이다. 다양한 파이썬 API를 제공하고 있기 때문에 복잡한 기능을 다른 드론 기종들에 비해 쉽게 구현할 수 있다. 또한 개당 가격이 약 20만원 정도로 저렴하여 경제적이다. 구체적인 제원은 다음과 같다.

- 비행: 최대거리 100m, 최대 속력 8m/s, 최대 비행 시간 13분, 최대 고도 30m
- 카메라: 최대 5백만 화소, 시야각: 82.6°, 동영상 최대 720p30

2.1.2 객체 탐지(YOLO 알고리즘)



<그림 2 : YOLO 탐지 Algorithm>

객체 탐지란, 하나의 입력 사진에서 사용자가 원하는 객체에 직사각형 모양의 선을 추가하고, 객체의 종류를 알려주는 것이다. 본 연구에서는 고가치 표적 탐지를 위한 알고리즘으로 YOLO 알고리즘(참고문헌[4])을 채택하였다.

YOLO 알고리즘은 Joseph Redmon 등에 의해 2016년 CVPR에서 발표되었으며, 객체 탐지를 위한 실시간 end-to-end 접근법을 처음으로 제시하였다. YOLO는 "You Only Look Once"의 약자이고 이는 네트워크를 한 번만 통과하여 탐지 작업을 수행할 수 있음을 의미한다. YOLO는 입력된 이미지를 일정 분할로 그리드한 다음, 신경망을 통과하여 바운딩 박스와 클래스 예측을 생성하여 최종 감지 출력을 결정한다. 복잡한 파이프라인을 다루지 않기 때문에 매우 빠른 모델이며, 따라서 실시간 의사 결정을 필요로 하는 분야에서 특히 두각을 드러내므로 드론의 객체 판별에 굉장히 적합하다.

<그림 3 : Non-Maximum Suppression (NMS) >

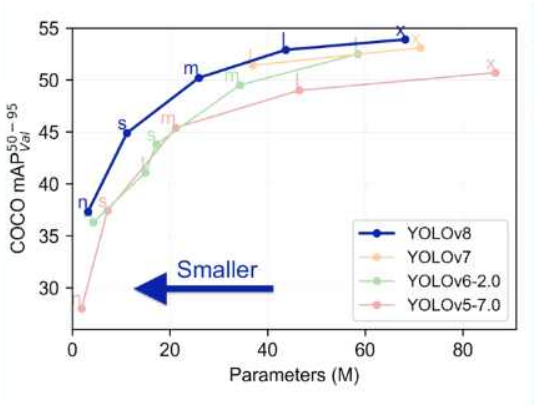
YOLO의 핵심적인 기술 중 하나는 후처리 기술로 이용되는 Non-Maximum Suppression (NMS)이다. 객체 탐지 알고리즘은 일반적으로 동일한 객체 주위에 다양한 신뢰도 점수를 가진 여러 바운딩 박스를 생성하는데, NMS는 중복되거나 관련 없는 바운딩 박스를 필터링하여 가장 정확한 것만을 유지한다.

YOLO는 2016년에 발표된 이후로 지속적으로 성능과 속도 면에서 개선되었고 2023년에는 YOLO v8이 등장하였다. YOLO v8은 C2f 모듈을 사용하여 높은 수준의 특징과

문맥 정보를 결합하여 탐지 정확도를 향상하였고, 분리된 헤드를 사용하여 객체성, 분류, 회귀 작업을 독립적으로 처리한다.

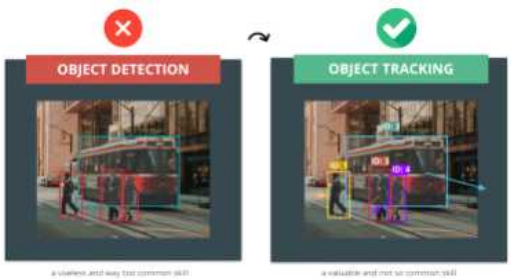
본 연구에서는 탐지 algorithm으로 API가 잘 제공되고 있는 YOLO 모델을 사용하였고, YOLO 모델들 중 mAP지표가 다른 모델들에 비해 10이상 우수한 YOLOv8x를 사용하였다.

YOLOv8x의 성능지표는 다음과 같다. 출처는 ultralytics 공식 웹사이트(참고문헌[4])를 참고하였다.



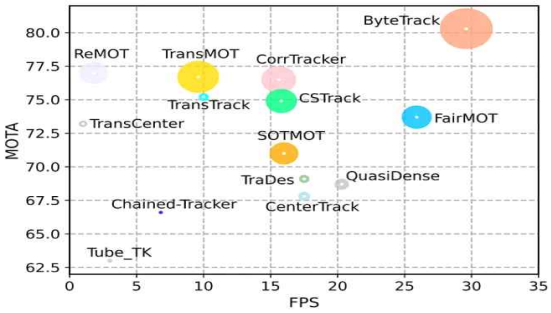
<그림 4 : YOLO v8 성능지표>

2.1.3 객체 추적(ByteTrack)



<그림 5 : 객체 추적>

객체 추적이란, 동영상을 입력으로 객체 탐지를 수행할 때, 즉 여러 사진을 sequential하게 객체 탐지를 할 때, 이전 사진에서 탐지 했던 물체가 현재 사진에서 탐지 한 물체와 동일한지의 여부를 고유번호(id)를 부여해 줌으로써 알려주는 것이다. 본 연구에서는 객체 추적 알고리즘으로 MOTA 지표가 다른 모델들에 비해 제일 우수한 ByteTrack 알고리즘(참고문헌[5])을 사용하였다. ByteTrack의 성능지표는 그림 6에 첨부하였다.

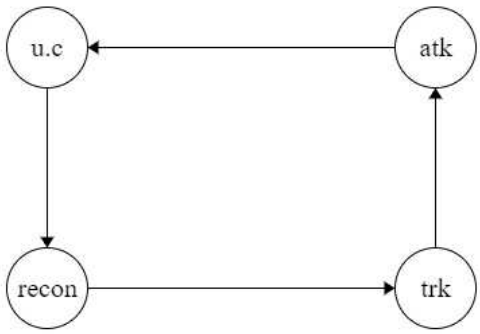


<그림 6 : ByteTrack 성능 지표>

2.2 연구의 전체적인 알고리즘과 구현

2.2.1 전체적인 알고리즘 설명

본 연구의 전체적인 알고리즘은 아래 그림과 같다.



<그림 7 : 알고리즘 순서도>

먼저, 프로그램을 실행시키게 되면 드론은 u.c(user control) 상태에 진입한다. u.c 상태란, 본부(드론과 연결된 컴퓨터)에서 드론을 직접 조종하여 적의 요충지까지 이동시키는 것을 의미한다. 이후 상태를 바꾸면, 드론은 recon (reconnaissance) 상태로 진입하고, 미리 입력된 적을 탐지할 때까지 회전

하며 적을 정찰한다. 만약 적이 화면상에 나타나면, 본부에 알리고, 추적 id를 통해 표적을 입력받는다. 표적을 입력받은 드론은 trk 상태로 진입하며 표적을 추격하며 표적의 3D 좌표를 실시간으로 본부에 전송한다. 마지막으로 본부에서 공격 명령을 내리면, atk 상태로 바뀌며 표적에 돌진해 자폭한다. 단, 이때 공격을 중지하고 싶다면 u.c 모드로 돌아가면 된다.

2.2.2 u.c 상태 구현

u.c 상태는 user control 상태로써, 본부가 직접 드론을 조종하여 적의 요충지까지 드론을 이동시키는 과정이다.
본 연구에서는 djitellopy API를 통해 드론의 속도를 3개의 축으로 나누어 제어하고, z축을 회전축으로 회전도 가능하게 하였다. 또한, 키보드 입력을 연속적으로 받기 위해서 pygame 모듈을 이용하였다. 이러한 모듈들을 이용하여 키보드 입력에 따라 드론의 움직임을 제어할 수 있었다. pseudo-code는 다음과 같다.

Algorithm 1 Pseudocode for Keyboard Input and Drone Control

```
while true do
    ki ← getKeyboardInput()
    drone.send_control(ki)
end while
```

<그림 8 : pseudo-code>

2.2.3 recon 상태 구현

<그림 9 : recon 상태>

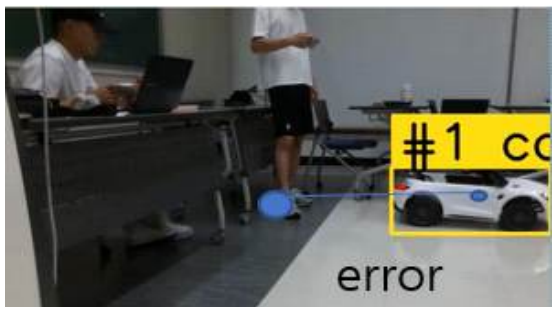
recon 상태는 reconnaissance 상태 즉 정찰 상태이다. 이 상태에서는 미리 코드에 입력된 적을 탐지할 때까지 회전하며 적을 정찰한다. 만약 적이 화면상에 나타나면, 본부에 알리고, 추적 id로 표적을 입력받아야 한다.
본 연구에서는 이 기능을 구현하기 위해서 YOLO 객체 탐지 알고리즘과 ByteTrack 추적 알고리즘을 사용하였다. pseudo-code는 다음과 같다.

Algorithm 2 Pseudocode for reconnaissance mode

```
while true do
    frame ← get_frame()
    frame ← YOLO(frame)
    if is_detected(frame) then
        tracking(frame)
        show_tracking_id(frame)
        target ← input("What's the target num?")
    else
        drone.spinning()
    end if
end while
```

<그림 10 : pseudo-code>

2.2.4 trk 상태 구현



<그림 11 : trk 상태>

trk mode는 드론이 표적을 추격하며 표적의 3D 좌표를 본부에 송신하는 상태이다. 먼저 표적 추격에 관해 설명하겠다.

그림 11과 같이 객체 추적을 하는 중에는, ByteTrack 알고리즘을 통해 특정 id를 가진 객체의 스크린 내의 좌표를 얻을 수 있다. 이 좌표와 화면 중심의 좌표 사이의 차이를

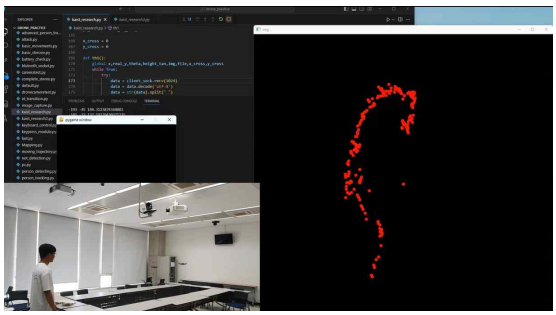
오차로 지칭하고, 오차가 0이 되도록 드론의 yaw 회전과 z축 방향 움직임에 대해 PID 제어를 수행하면 표적을 화면 중앙에 올 수 있도록 드론을 제어할 수 있다. 또한 드론에서 표적 객체가 멀리 떨어질수록 직사각형의 넓이가 작아지므로, default 직사각형의 넓이와 현재 직사각형의 넓이의 차이를 오차로 지정하고, 드론의 앞-뒤 움직임에 대해 PID 제어를 수행하면, 표적과의 일정 거리를 유지하며 추격할 수 있다. 구현 디테일과 pseudo-code는 다음과 같다.

Algorithm 3: Pseudocode for Object Tracking and Control

```
while true do
    frame ← get_frame()
    frame ← YOLO(frame)
    if is_detected(frame) then
        result ← tracking(frame)
        x, y ← result.get_coordinate(target)
        area ← result.get_area(target)
        actuator_yaw_z ← PID(x, y)
        actuator_forward_backward ← PID(area)
        drone.send_control(actuator_yaw_z, actuator_forward_backward)
    end if
end while
```

<그림 12 : tkr 상태 - 드론 제어>

다음으로는 3D좌표를 본부에 송신하는 방법에 관해서 설명하겠다.



<그림 13 : 표적의 3D 좌표>

표적의 3D좌표를 구하기 위해서는 드론이 두 대 필요하고, 드론 두 대가 같은 표적을 추적하고 있어야 한다.

이러한 상황에서, 본부의 좌표를 (0,0,0)이라고 했을 때, 표적의 3D좌표를 얻기 위해서는 두 가지 단계를 거치면 된다.

단계 1: Drone 2개의 좌표 구하기.
단계 2: Drone의 좌표정보와 각 드론에서의 표적 추적 정보를 활용한 표적의 좌표 구하기.

이 두 가지 단계를 설명해 보도록 하겠다.

먼저 단계 1이다. 드론의 좌표를 구하기 위해서는 가장 간단하게 GPS를 이용하면 될 것이다. 하지만, Tello edu 드론에 GPS가 내장되어 있지 않기 때문에, 다른 방법을 택해야 했다. 방법은 다음과 같다.

드론을 제어할 때, 드론에 각 축에 따른 이동 속도로 loop마다 명령을 주게 된다.

그렇다면, 각 축에 대한 이동 거리는 $\sum(speed * looptime)$로 계산할 수 있다. 여기에 회전운동을 고려하여 수식을 만들면 된다. 수식을 만들 때 주의할 점은, open cv로 출력하기 위해 y축이 뒤집어져 있는 것에 주의해야 한다. 구체적인 수식은 수식이 복잡한 관계로 연구 결과물[1]에 설명해 두었다. pseudo-code는 다음과 같다.

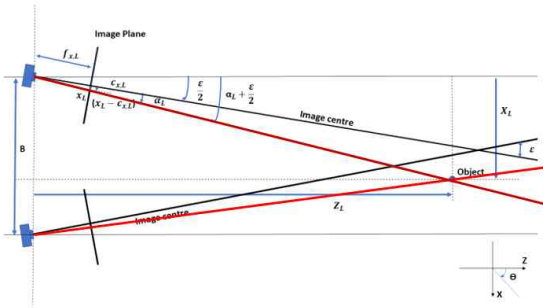
Algorithm 4: Pseudocode for calculating coordinate of drone

```
x_coor, y_coor, z_coor ← 0, 0, 0
while true do
    frame ← get_frame()
    frame ← YOLO(frame)
    if is_detected(frame) then
        result ← tracking(frame)
        x, y ← result.get_coordinate(target)
        area ← result.get_area(target)
        actuator_yaw_z ← PID(x, y)
        actuator_forward_backward ← PID(area)
        drone.send_control(actuator_yaw_z, actuator_forward_backward)
        x_coor ← x_coor+looptime×calculate_x_speed(actuator_yaw_z, actuator_forward_backward)
        y_coor ← y_coor+looptime×calculate_y_speed(actuator_yaw_z, actuator_forward_backward)
        z_coor ← z_coor+looptime×calculate_z_speed(actuator_yaw_z, actuator_forward_backward)
    else
        drone.spinning()
    end if
end while
```

<그림 14 : pseudo-code>

pseudo-code에는 u.c 상태일 때의 드론의 좌표 추적법이 담겼다. 같은 method로 표적을 추격할 때 역시 쉽게 드론의 좌표를 추적할 수 있다.

다음은 단계 2이다. 이번 단계에서는 드론의 좌표정보와 각 드론에서의 표적 추적 정보를 활용해 표적의 좌표를 구하게 된다. 방법은 다음과 같다.



<그림 15 : 3D 좌표 추적 알고리즘>

드론 2대가 공동 표적을 추격하고 있는 모습을 (x,y) 평면에 정사영 한다면, 위와 같은 모습일 것이다. 여기서 중요한 점은, [2]에 따르면, 만약 드론이 찍고 있는 전방 실시간 영상의 Image Plane 내에서의 표적의 좌표를 안다면, 카메라의 정중앙에서 카메라 렌즈 평면에 수직으로 나오는 Image centre 직선과 카메라의 정중앙에서 표적으로 뻗는 빨간색 직선 사이의 사잇각 α를 알 수 있다는 것이다.

이 사잇각 α를 안다면, 우리는 이미 Step 1을 통해 각 드론들의 좌표와 각 드론이 바라보고 있는 각도 $\frac{\epsilon}{2}$를 알고 있으므로, 다음과 같은 두 일차함수의 교점이 표적의 x,y 평면상의 좌표가 될 것이다.

$$y = \tan(\frac{\epsilon_L}{2} + \alpha_L)(x - x_L) + y_L$$

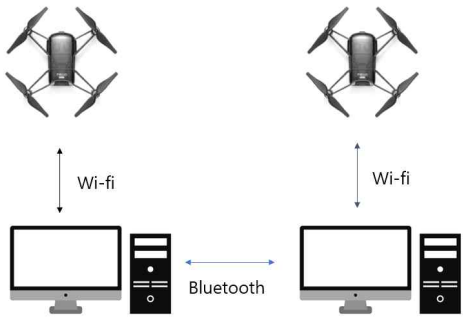
$$y = \tan(\frac{\epsilon_R}{2} + \alpha_R)(x - x_R) + y_R$$

where x_L,y_L is coordinate of left drone
x_R,y_R is coordinate of right drone

이후 표적의 z 좌표는 [2]논문에서 소개한 방법을 이용하여 x와 y, α_z를 바탕으로 도출 할 수 있다.

위의 알고리즘을 이용한다면 표적의 3D좌표를 실시간으로 추적할 수 있다.

한편, 위 알고리즘을 사용하여 표적의 좌표를 추출하기 위해서는 드론 2대가 서로의 정보를 주고받아야 한다.

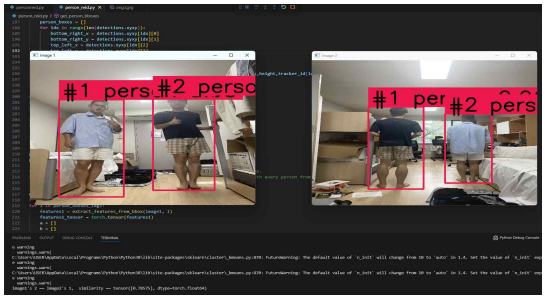


<그림 16 : setting>

각각의 컴퓨터는 각 컴퓨터와 연결된 드론이 만드는 와이파이에 연결되어 있으므로, 서로 다른 와이파이에 연결되어 있게 된다. 따라서 컴퓨터끼리는 와이파이를 이용한 통신이 불가능했다. 이를 극복하고자 pybluez 모듈을 이용하여 각 컴퓨터끼리 블루투스 통신을 실시하였다.

실시간 경로 plot의 경우에는, 서버 역할을 하는 컴퓨터에서 threading을 이용하여 표적의 좌표를 실시간으로 계산하고 plot 하였다. 파이썬에서는 연속적인 3D Plot이 불가능하므로, 파이썬 프로그램 내에서는 x,y축으로 정사영한 2D Plot의 실시간 plot을 진행하고, 3D 좌표 정보는 실시간으로 result.txt라는 파일에 저장하여 매트랩을 통해 실시간 3D plot을 진행하였다.

trk mode에서 또 하나의 이슈는 두 드론이 같은 표적을 추적할 때만 표적의 3D좌표를 얻을 수 있다는 것이다. 각각의 드론에서 찍은 영상은 각각 별도의 추적 알고리즘을 통과하기 때문에, 실제로 같은 물체이더라도 id가 다르게 출력된다. 때문에, 표적의 3D좌표를 추적하기 위해서는 본부에서 두 드론에 각각 표적의 아이디를 입력해야 한다.

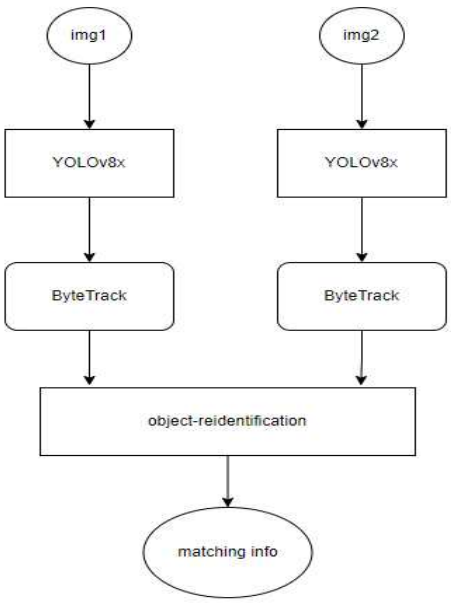


<그림 17 : different id>

위의 그림 17을 보면, 실제로는 같은 사람이지만 별도의 추적 알고리즘을 통과하면 id가 다른 것을 확인할 수 있다.

본 연구팀은 두 드론에 각각 표적의 아이디를 입력해야만 하는 한계를 해결하고자, 하나의 드론에 표적의 아이디를 입력하면, 다른 드론에서는 자동으로 표적의 아이디를 찾도록 객체 re-identification 알고리즘을 구현하였다.

객체-reidentification 알고리즘이란, 여러 카메라로부터 취득된 다양한 이미지나 비디오에서 동일한 개체나 사람을 자동으로 매칭시키는 알고리즘이다. 알고리즘의 순서도는 다음과 같다.



<그림 18 : 알고리즘 순서도>

먼저 서로 다른 두 각도에서 찍은 사진은 각각 객체 탐지, 객체 추적 알고리즘을 차례로 통과한다. 이후 각각의 이미지에서 탐지에 성공한 객체를 전체 사진에서 잘라내고, 객체의 외형, 색 정보 등을 담고 있는 특징 벡터를 추출한다. 특징 벡터 추출은 torchreid 모듈(참고문헌[6])에서 제공하는 osnet_x1_0 pre trained 모듈과 직접 만든 객체 색깔 특징 추출기를 ensemble 하였다. 마지막으로, 서로 다른 사진에 있는 객체들의 특징 벡터의 similarity를 비교하여 매칭을 완료한다. 그림 17에 대해 아래 콘솔 창을 보면 성공적으로 matching에 성공한 것을 확인할 수 있다. 왼쪽 그림의 2번 id를 가진 사람이 오른쪽 그림의 1번 id를 가진 사람과 같은 사람이라는 의미이다.

```
image1's 2 == image2's 1,  similarity == tensor([0.78575], dtype=torch.float64)
```

2.2.5 atk 상태 구현

atk 상태는 trk 상태의 표적 추격에서 forward backward speed를 200cm/s로 고정시킴으로써 구현했다.

2.2.6 구동 영상

해당 section에서는 위 시스템을 이용하여 장난감 차에게 자폭 공격을 시행하는 영상을 1초간격으로 캡쳐한 연속적인 사진을 소개한다. 각 사진을 보면 사진 위에 색으로 모드를 표시해 놓았다. (검은색 : u.c | 갈색 : recon | 빨간색 : trk | 파란색 : atk)

<그림 20 : 구동 영상>

2.3 정확성 및 성능 실험 결과

2.3.1 객체 탐지

객체 탐지를 크게 3가지 class에 대해서 실행하였다. 사람, 차, 탱크에 대하여 시행하였다. 사람과 차는 YOLOv8x 기본 모델을 사용하였고, 탱크는 직접 4000개의 훈련 데이터, 800개의 검증 데이터를 epoch 100으로 훈련했다. 1000개의 테스트 데이터에 대해 실험한 결과, 사람과 차는 99% 이상의 매우 높은 정확도를 보였고, 탱크는 전체적으로 성능이 아쉬웠는데, 탱크 데이터의 경우 웹 상의 공개된 데이터가 매우 적어 충분한 훈련 데이터를 얻을 수 없었기 때문이다. 추가적인 학습 데이터만 확보할 수 있다면 충분히 좋은 성능을 보일 것으로 기대된다.

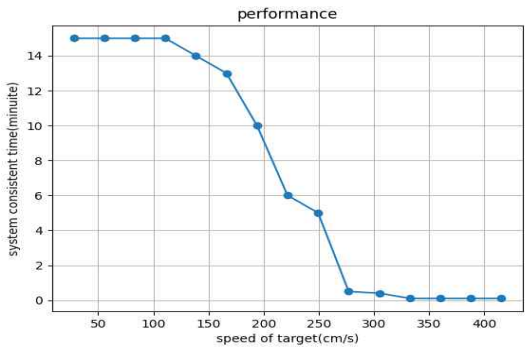
2.3.2 객체 추적

추적 시도(횟수)	추적 성공(횟수)
1,440,000	1,440,000

<표 1 : 추적 정확도>

객체 추적은 탐지의 정확도가 100%라고 가정했을 때, 정확도가 100%였다. 본 연구팀은 약 200회 정도 시험 비행을 실시했는데, 객체 탐지가 성공 중이었을 때, 단 한 번도 id transition이 발생한 것을 관찰하지 못했다. {추적 시도 횟수 = 시험 비행 횟수 × 평균 추적 지속 시간 × FPS}

2.3.3 추적된 표적 추격



<그림 21 : 추격 지속시간 - 표적 속도 그래프>

위의 그림 21은 표적의 속도에 따른 추격 지속시간 그래프이다. 표적 추격 지속시간이 표적의 속도에 따라 차이가 발생하는 이유는 표적의 속도가 빠를수록, 표적을 드론이 따라갈 수 없을 만큼 표적이 빨라 화면에서 사라지는 경우가 빈번하게 발생하기 때문이다. 결과를 보면, 150cm/s 이하의 속도로 자유롭게 움직이는 표적을 추격할 때에는 시스템이 정상 작동하는 반면, 200cm/s 이상의 속도로 움직이는 표적에 대해서는 평균 시스템 지속 시간이 5분 이하인 것을 알 수 있다.

하지만 이는 PID 제어 내부 변수들을 사람 표적에 맞춰 최적화 한 것이기에, 탱크나 차에 맞춰 재 최적화를 한다면, 표적의 속도가 높더라도 시스템의 지속 시간이 높게 유지될 것이다.

2.4 발생한 문제들과 극복 방안

이번 section에서는 본 연구를 하면서 발견했던 문제들과 그에 대한 극복 방안에 대해서 다뤄보고자 한다.

먼저 첫 번째 문제는 객체 탐지가 1~3프레임 정도 끊겼을 때, 추적 id가 변하는 문제이다.

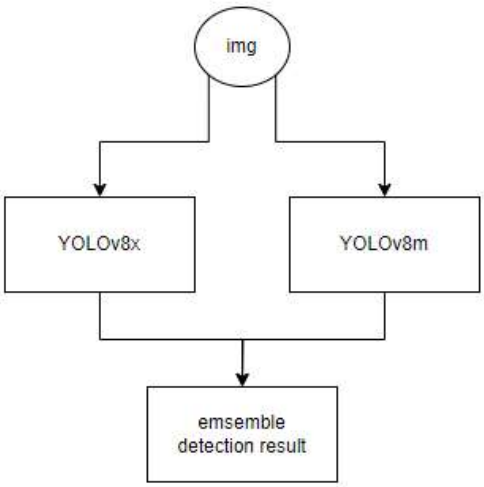
frame 1134th frame 1137th

<그림 22 : id transition 문제>

그림 22를 보면 표적의 추적 id가 1에서 6으로 변한 것을 볼 수 있다. 위 문제는 3분에 한 번 정도의 주기로 1~3 frame 정도 객체 탐지가 되지 않을 때 발생한다. 본 연구팀은 2가지 방안으로 위 문제점을 해결하였다.

첫 번째 방안은 ensemble model을 활용한 객체 탐지 정확도 향상이다. 위 문제가

발생한 상황들을 분석해 보면, 사람이 아닌 객체를 사람으로 탐지 하는 경우는 전혀 없었고, 사람을 3분에 1~3 frame 정도 탐지하지 못해서 발생하는 문제였다. 따라서 본 연구팀은 YOLOv8x model과 YOLOv8m model을 이용하여 간단한 앙상블 모델을 만들어 사람이 가끔 탐지되지 않는 문제를 해결하였다. 앙상블 모델의 세부 디자인은, 입력 이미지를 두 모델에 넣고, 탐지된 개수가 많은 모델의 결과를 탐지 결과로 사용하는 간단한 아이디어이다. 사람이 아닌 객체를 사람으로 탐지하는 경우가 없기 때문에 가능한 앙상블 모델이다. 아래 그림23은 앙상블 모델의 구현도이다.



<그림 23 : 앙상블 기반 객체 탐지 모델>

두 번째 극복 방안은 위 그림 18에서 소개한 객체 - reidentification을 이용하는 것이다. 만약 그림 22와 같이, id transition 문제가 발생하면, 프로그램에 입력되어 있는 표적 id = 1인데, 표적 id = 1인 객체가 없어, 드론이 정지하게 된다.

이때 유의할 점은 객체 탐지가 잘 되지 않았던 1~3 프레임이 지나면, 표적의 id만 바뀌었을 뿐, 객체 탐지 자체는 제대로 작동한다는 것이다. 1~3 프레임은 0.1초 정도이므로, 표적이 그동안 화면 밖으로 사라졌을 가능성은 매우 낮다. 그러므로, loop마다 표적의 사진을 변수에 저장해 두고, id transition이 일어나면, 표적의 사진과 현재 탐지된 객체들의 similarity를 계산하여 표적과 현재 탐지된 객체의 similarity가 0.9 이상일 경우 표적의 id를 해당 객체의 id로 자동으로 바꿔주는 것이다. 간단하게 말하면, 간단한 객체 추적 알고리즘(reidentification)을 만들어서 원래의 ByteTrack과 앙상블 했다고 생각할 수 있다.

	기존	방법 1	방법 2	방법1+2
id transition 발생 빈도	1%	0.6%	0.8%	0.34%

<표 2 : id transition 발생 빈도>

위 두 방법을 이용해서 id transition 문제의 빈출 빈도를 이용하기 전에 비해 30%가량으로 줄이는 데 성공하였다. 완벽히 막을 수 없었던 이유는, 탐지 ensemble 모델에서 사용한 객체 탐지 알고리즘들이 종류는 다르지만 비슷한 net 형태를 띠고 있어, 사람을 같이 탐지하지 못하는 경우가 많았고, reidentification을 할 때, 혹시 다른 객체를 원래의 표적으로 잘못 오해하는 것을 막기 위해서 similarity를 0.9로 높게 설정했기 때문이다. 따라서 id transition 문제를 완벽히 막는 것은 아직 과제로 남아있다. 이에 대한 기대 극복 방안은 밑의 3. 추후 연구 제안 section에서 소개하겠다.

두 번째 문제는, 바람이 강할 경우 각 드론의 위치를 정확하게 알아낼 수 없다는 것이다. 드론이 바라보고 있는 각도인 yaw 각도 같은 경우에는 하드웨어 상에서 자기장을 기반으로 측정값을 제공해 줘서 오차가 1° 정도로 매우 작지만, 드론의 3D 위치의 경우에는 드론에 내린 속도 command를 기반으로 계산하기 때문에, 만약 드론이 강한 바람 등에 의해 command만큼의 속도를 내지 못하면, 오차가 크게 발생하였고, 오차가 누적

되는 문제가 발생하였다. 이를 극복하기 위해 Tello edu drone에 GPS module을 추가하여 드론의 3D위치 오차를 시간과 관계없이 1~2m 내로 줄일 수 있었다.

<그림 24 : GPS 모듈 추가>

3. 추후 연구 제안

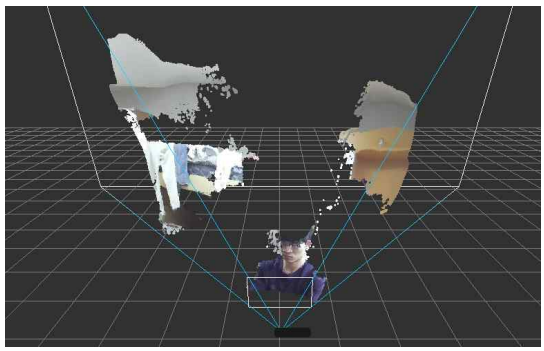
<그림 25 : zed 2i>

본 연구의 id transition 문제와 표적의 3D 좌표 오차 문제를 해결할 수 있는 방법으로 본 연구팀은 스테레오 카메라를 내장하는 드론을 생각하였다.

이는 한 드론에 카메라 2대가 내장되어있는 하드웨어 환경을 말한다. 아직 이러한 하드웨어 환경을 가지며 프로그래밍 언어와 호환이 되는 드론이 존재하지 않기 때문에, 본 연구팀은 파이썬과 호환되는 Zed 2i 스테레오 카메라를 구입하여 이러한 하드웨어가 드론에 탑재된다면 id transition 문제와 표적의 3D 좌표 오차 문제를 어떻게 해결할 수 있으며, 추가적으로 어떠한 연구가 수행될 수 있을지를 분석하였다.

실험 결과, 스테레오 카메라를 이용하면,

<그림 26 : 3D depth map>

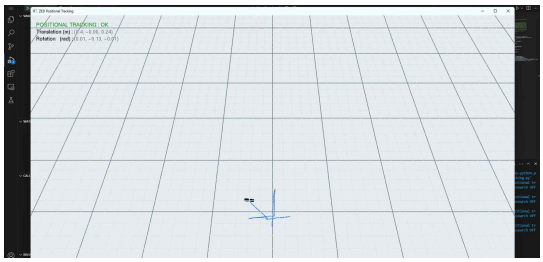


<그림 27 : 3d mapping>

1. depth map을 얻음으로써 객체 탐지 및 추적에 도움을 줄 수 있고, 표적의 3D 좌표 추출 정확성을 향상시킬 수 있다. 그림 26, 27을 보면, python 환경에서 depth map을 visualization 하는 code가 성공적으로 실행되는 것을 볼 수 있다.

<그림 28 : 3d 객체 탐지>

2. 3D 환경에서 객체 탐지와 추적을 수행함으로써, 객체 탐지와 추적 정확도를 향상해 id transition 문제를 해결할 수 있을 것이다. 2D에서는 화면의 같은 좌표에 포착되더라도, 같은 객체가 아닐 가능성이 있지만, 3D에서는 같은 좌표라면 무조건 같은 객체이기 때문이다. 이 특징이 추적 알고리즘의 성능을 향상시킬 수 있다고 생각한다. 그림 28을 보면, zed 2i 스테레오 카메라를 이용하여 python 환경에서 객체 탐지를 수행한 것이다.



<그림 29 : 위치 추적>

3. gps를 이용한 드론의 위치 추적과 함께, 영상 정보를 토대로 위치 추적을 수행하면서, 드론의 위치 추적을 보다 정확하게 수행해, 표적의 3D 좌표 오차를 줄일 수 있을 것이다. 그림 29는 영상 정보만을 기반으로 카메라를 움직이며 카메라의 위치를 추적한 것이다. 오차는 5cm 이내로 발생하였다.

이 외에도 표적이 장애물 등에 의해 갑자기 화면에서 사라졌을 때, 최신 표적의 3D위치 정보를 토대로 최신 위치로 이동하여 재탐색을 수행할 수 있는 등 많은 추가 연구를 수행할 수 있을 것이다.

위의 모든 실험은 파이썬 환경에서 코드의 excution을 통해 진행된 것이다. 만약 zed 2i 와 같은 스테레오 카메라가 내장된 드론 하드웨어를 만들 수 있다면, 본 연구팀의 id transition 문제와 표적의 3D 좌표 오차 문제를 해결하면서, 파이썬과 함께 무궁 무진한 발전을 이룰 수 있다고 기대하며, 이를 추후 연구과제로 추천한다.

4. 본 연구의 기술적 기여

- Python 3.8.10, YOLO 8.0.10, torch 1.13.1+cu116, numpy 1.22.4 환경에서 본 연구의 모든 코드는 한 코드에서 실행된다. 객체 탐지, 객체 추적, 객체 re-identification, PID 제어등의 알고리즘이 한 코드에서 실행됨으로써, 코드들의 호환성을 바탕으로 시뮬레이션 환경이 아닌, 실제 환경에서 동작이 가능하다. 또한, 하나의 프로그램만 실행하면 되기 때문에, 군의 사용자가 이용하기에 편리하다.

- GPU만 있는 컴퓨터에서, Python 3.8.10, YOLO 8.0.10, torch 1.13.1+cu116, numpy 1.22.4 환경만 구현한다면, 복잡한 환경설정이 필요하지 않기 때문에 연구를 재현하기 쉽다.

- 표적의 3D 좌표추적을 구현함으로서, 다른 연구들의 가정 및 전제조건을 만족시켰다. 일례로, 밀리테크 연구 “강화학습 기반의 비대칭 드론 전력 대응을 위한 격추 기법 개발”의 전제조건은 표적의 3D좌표와 속도를 아는 것인데, 이를 표적의 3D 좌표추적을 구현함으로서 만족시켰다.

- ZED 2i를 이용해 프로그래밍 언어와 호환이 가능하며 스테레오 카메라를 내장한 드론 하드웨어의 개발 가능성과 그 이점을 제시하였다.

5. 참고문헌

[1] 드론의 군사적 효용성(활용성) 및 발전전략 - 조선대학교 군사학 연구소

[2] Near-Parallel_Binocular-Like_Camera Pair_for_Multi-Drone_Detection_and 3D_Localization. IEEE.

[3] https://github.com/stereolabs/zed sdk - zed 2i code

[4] https://github.com/ultralytics/ultralytics - YOLOv8 reference

[5] https://github.com/ifzhang/ByteTrack- ByteTrack reference

[6] https://github.com/KaiyangZhou/deep-person-reid - torchreid, osnet_x1_0 reference

6. 연구 결과물

[1] https://github.com/seokm/militech - our research code

[2] https://www.youtube.com/@lsm9434 - experiment video