제목 : 객체인식을 활용한 자율주행 드론

I. 서론

1.1 작품 요약 소개

이 프로젝트는 DJI Tello EDU 드론을 활용하여 객체 인식을 통해 자율 주행 기능을 구현하는 것을 목표로 한다. Python을 사용하여 훈련데이터 세트를 이용한 객체 인식, 드론의 기본적인 조작, 카메라 스트리밍기능을 구현하였다.

1.2 기존 드론 대비 독창성 및 차별성, 강점 소개

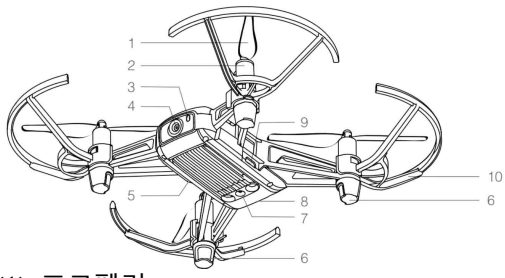
기존의 드론은 기본적인 조종 기능만을 제공하는 반면, 이 프로젝트에서는 객체 인식과 스트리밍 기능을 통해 드론을 더욱 직관적이고 편리하게 제어할 수 있는 기능을 추가하였다. 이를 통해 사용자와의 상호작용이 증가하고, 자율 주행의 가능성을 높였다.

II. 본론

- 구체적인 드론 강점 소개 (드론 및 제작에 사용된 부품 등의 사진 및 설계도면)

- **강점** : 기본적인 주행 드론 방식은 입력값을 조절하여 장애물을 통과하는 기본 방식이 아닌, 객체 인식을 활용한 자율주행 기능을 통해 더 높은 수준의 장애물 회피 능력을 갖추고 있다. 이를 통해 드론은 장애물을 인식하고 파악하여, 도착지점에 안전하고 정확하게 도달할 수 있다. 이러한 기능은 드론의 효율성과 안전성을 크게 향상시켜 다양한 응용 분야에서 활용 가능성을 높인다.

- **설계도면**



(1) 프로펠러
(2) 모터
(3) 항공기 상태 표시기
(4) 카메라
(5) 전원버튼
(6) 안테나
(7) 비전 포지셔닝 시스템
(8) 비행 배터리
(9) 마이크로 USB 포트
(10) 프로펠로 가드

(1) 프로펠러
- 드론의 비행을 위해 양력을 생성한다.
- 프로펠러의 회전 속도와 방향을 조절하여 드론의 상승, 하강, 이동, 회전을 제어한다.

(2) 모터
- 프로펠러를 회전시키는 역할을 한다.
- 전기 에너지를 기계적 에너지로 변환하여 드론이 비행할 수 있도록 한다.

(3) 항공기 상태 표시기
- 드론의 상태를 사용자에게 시각적으로 표시한다.
- 배터리 상태, GPS 신호, 연결 상태 등 다양한 정보를 제공하여 안전하고 효율적인 비행을 지원한다.

(4) 카메라
- 객체 인식, 영상 분석, 실시간 모니터링 등의 기능을 통해 다양한 응용 분야에서 활용한다.

(5) 전원버튼
- 드론의 작동을 시작하거나 종료할 때 사용한다.

(6) 안테나
- 드론과 조종기 간의 통신을 담당한다.
- 신호를 송수신하여 원격 조종과 데이터 전송을 가능하게 한다.

(7) 비전 포지셔닝 시스템
- 드론의 위치를 파악하고 안정적인 비행을 지원한다.
- 바닥에 있는 특정 패턴이나 물체를 인식하여 실내외에서 GPS 신호가 없는 환경에서도 정확한 위치를 유지할 수 있게 한다.

(8) 비행 배터리
- 드론의 모든 전자 장치와 모터에 전원을 공급한다.
- 비행 시간과 성능에 큰 영향을 미치며, 주기적인 충전이 필요하다.

(9) 마이크로 USB 포트
- 드론의 소프트웨어 업데이트, 데이터 전송, 배터리 충전 등의 목적으로 사용한다.
- 외부 기기와의 연결을 통해 다양한 기능을 수행할 수 있다.

(10) 프로펠로 가드
- 프로펠러를 보호하여 충돌이나 사고 시 손상을 최소화한다.
- 사람이나 물체와의 접촉으로 인한 위험을 줄여 안전성을 높인다.

2.1 드론 구성 내용 및 특징 소개

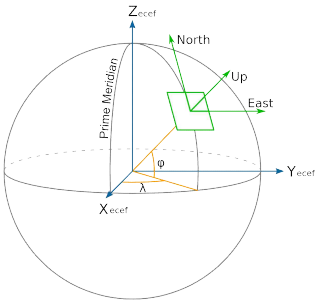
- 드론 본체 : DJI Tello EDU
- 프로그래밍 언어 : Python
- 구성 언어 시스템 : 파이참
- 사용된 라이브러리 : djitellopy, OpenCV, Mediapipe, Pygame

2.2 드론 구동을 위한 센서 종류 및 활용법 소개

- **IMU 센서** : 드론의 자세 제어 및 안정화를 위해 사용된다.

□ 텔로 드론의 경우 자신의 자세(롤,피치,요)를 알 수 있지만 위치는 알 수 없다. (고도 까지만)

□ 보통 사용하는 드론의 경우 GPS 센서로부터 위도경도를 계산하고, 이를 x,y,z의 지역좌표계로 변환하여 나타낸다. (지구는 둥글지만, 우리가 인식하는 범위는 평면이라 지역좌표계를 활용하면 좀 더 쉽게 인식할 수 있다. 앞으로 5m 가라는 명령 vs 위도 경도 명령, 당연히 전자가 편하다. NED 혹은 ENU 좌표계를 사용한다.)



(1) xyz 각각 East, North, Up 방향으로 좌표계를 나타낸 것을 ENU
(2) xyz 각각 North, East, Down 방향으로 좌표계를 나타낸 것을 NED
(3) IMU 센서로 자세를 알고, MAG센서로 헤딩방향을 안다. 텔로의 경우 기압계와 기체 바닥에 단방향 라이다로 고도정보를 알 수 있다.

- **카메라 센서**: 장애물을 객체인식 하여 코스를 통과한다.

□ 기체가 현재 위치를 유지하도록 도와줍니다. 비전 포지셔닝 시스템의 도움으로 Tello는 더 정확하게 제자리에서 호버링하고 바람이없는 환경에서 실내 또는 실외로 비행 할 수 있다.

□ 주요 구성 요소는 항공기 밑면에 위치한 카메라와 3D 적외선 모듈이다.

□ 성능은 비행하는 표면의 영향을 받는다. 비전 포지셔닝 시스템을 사용할 수없는 경우 기체는 자동으로 자세 모드로 변경된다. 자세 모드에서는 기체가 자체 위치를 지정할 수 없습니다. 기체가 자세 모드로 전환 될 수있는 다음 상황에서 기체를 매우주의하여 작동해야 한다.

* 성능 *

(1) 사진: 5MP (2592x1936)

(2) FOV: 82.6°

(3) 동영상: HD720P30

(4) 형식: JPG(사진); MP4(동영상)

(5) EIS: 지원

- **비젼(VISION) 포지셔닝 센서** : 장애물을 객체인식 하여 코스를 통과한다.

□ 기체가 현재 위치를 유지하도록 도와줍니다. 비전 포지셔닝 시스템의 도움으로 Tello는 더 정확하게 제자리에서 호버링하고 바람이없는 환경에서 실내 또는 실외로 비행 할 수 있다.

□ 주요 구성 요소는 항공기 밑면에 위치한 카메라와 3D 적외선 모듈이다.

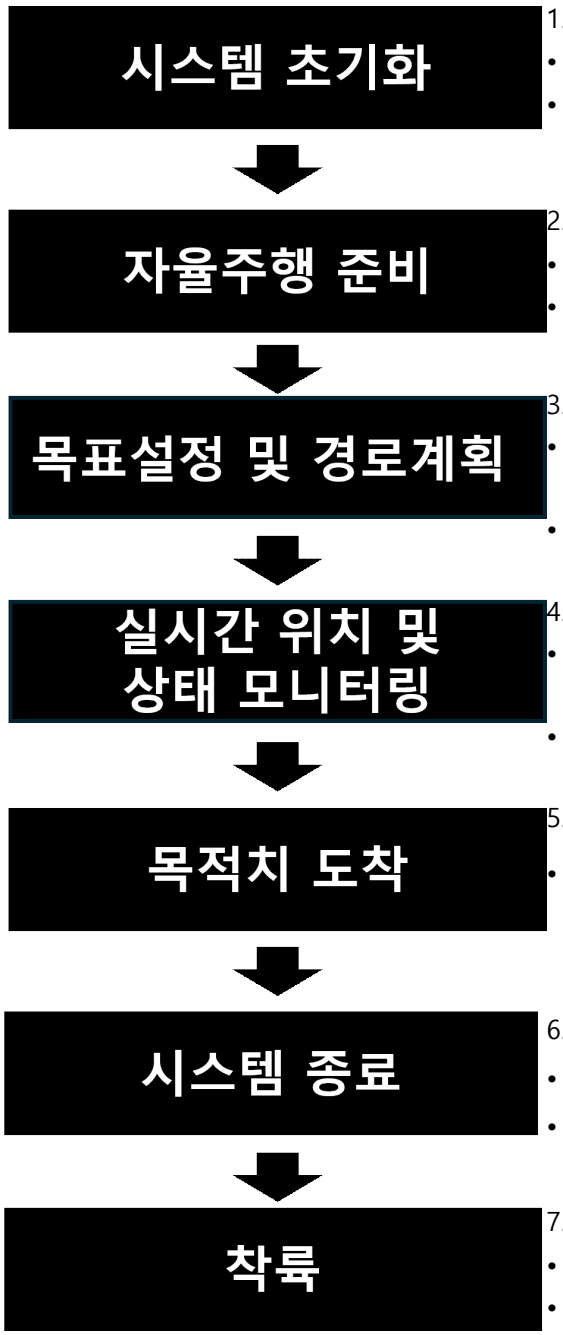
□ 성능은 비행하는 표면의 영향을 받는다. 비전 포지셔닝 시스템을 사용할 수없는 경우 기체는 자동으로 자세 모드로 변경된다. 자세 모드에서는 기체가 자체 위치를 지정할 수 없습니다.

* 비전 포지셔닝 센서가 작동하는 경우에도 호버링이 되지 않는 이유 *

(1) 정확한 패턴의 모양과 반사되지 않는 재질일 때

(2) 자기장 간섭에 의한 경우

2.3 자율 주행 드론을 위한 프로그램 알고리즘 소개 (알고리즘 흐름도 포함)



1. 시스템 초기화
- 드론의 시스템 및 센서 초기화를 수행
- GPS, IMU, 카메라 등의 센서 초기화 및 보정을 진행

2. 자율주행 준비
- 자율주행 모드로 진입하고, 비행준비 상태로 드론을 전환
- 자율주행 알고리즘 및 제어 알고리즘 초기화를 수행

3. 목표 설정 및 경로 계획
- 사용자로부터 목표 지점의 GPS 좌표를 입력받거나, 내부 목표 지점 데이터베이스에서 목표를 선택
- 선택된 목표로부터 경로를 계획하고, 적합한 경로 계획 알고리즘 (예: A* 알고리즘)을 사용하여 최적경로를 계산

4. 실시간 위치 및 상태 모니터링
- GPS 및 IMU 데이터를 사용하여 드론의 현재 위치와 자세를 실시간으로 추정하고 모니터링
- SLAM 기술을 사용하여 자율적으로 환경 매핑 및 위치 추정을 업데이트

5. 목적지 도착
- 목적지 근처에 도착하면, 정확한 착륙 위치를 계산하고 목표에 도달했음을 시스템에 통보

6. 착륙
- 안전하고 정확한 착륙을 위해 착륙 절차를 수행
- 안전 거리와 충돌 방지를 고려하여 착륙 공간을 선택하고 착륙 진행

7. 시스템 종료
- 임무가 완료되면 시스템을 종료
- 드론 시스템 및 알고리즘을 정리하고 비행 모드를 종료

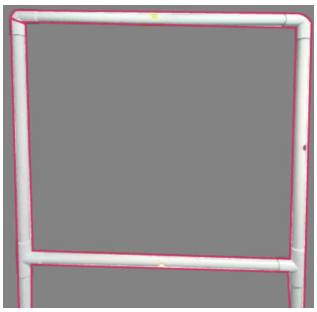
Ⅲ. 구현

3.1 대회 참가를 위한 기본 제작과정 소개

• 카메라 : 드론에 장착된 카메라는 실시간 객체인식을 위해 사용됩니다.

(장애물 누끼 따기)

네모 사진(초록색 배경)

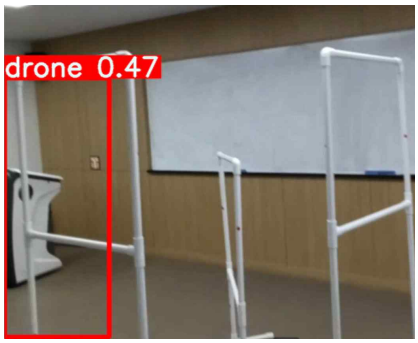


네모 사진(하얀색 배경)

장애물 전체 구조도

(실제 구동 모습)

장매물 사진(정면)



장애물 사진(측면)

3.2 작성 코드

```
import cv2
import torch
import pathlib
import sys
from utils.general import non_max_suppression, scale_boxes
from time import sleep
import threading
from djitellopy import Tello
# PosixPath를 WindowsPath로 재정의
pathlib.PosixPath = pathlib.WindowsPath
```

2. YOLOv5 모델 설정

```
# YOLOv5 경로 설정
yolov5_path = 'C:/Users/skwnd/PycharmProjects/adrone/yolov5'
sys.path.append(yolov5_path)
# YOLOv5 모델 경로 설정
model_path = 'C:/Users/skwnd/best_pt/best.pt'
```

```
# YOLOv5 모델 정의 및 가중치 로드
class YOLOv5Model:
    def __init__(self, model_path):
        # 모델 구조 정의
        self.model = torch.hub.load(yolov5_path, 'custom', path=model_path, source='local')
        self.model.eval()
    def __call__(self, img):
        with torch.no_grad():
            return self.model(img)
# 모델 초기화
model = YOLOv5Model(model_path)
```

3. 드론 설정 및 카메라 스트리밍 시작

```
# Tello 드론 객체 생성 및 연결
tello = Tello()
tello.connect()
# 드론 카메라 스트리밍 시작
tello.streamon()
# 카메라 프레임을 계속 가져오면서 CV 창을 열고, 프레임이 제대로 수신될 때까지 기다림
cv2.namedWindow('Tello Camera')
```

4. 경로 및 객체 인식 함수 정의

```
def plot_one_box(x, img, color=(128, 128, 128), label=None, line_thickness=3):
    tl = line_thickness or round(0.002 * (img.shape[0] + img.shape[1]) / 2) + 1
    color = [int(c) for c in color]
    c1, c2 = (int(x[0]), int(x[1])), (int(x[2]), int(x[3]))
    cv2.rectangle(img, c1, c2, color, thickness=tl, lineType=cv2.LINE_AA)
    if label:
        tf = max(tl - 1, 1)
        t_size = cv2.getTextSize(label, 0, fontScale=tl / 3, thickness=tf)[0]
        c2 = c1[0] + t_size[0], c1[1] - t_size[1] - 3
        cv2.rectangle(img, c1, c2, color, -1, cv2.LINE_AA)
        cv2.putText(img, label, (c1[0], c1[1] - 2), 0, tl / 3, [225, 255, 255], thickness=tf,
                    lineType=cv2.LINE_AA)
```

```
def drone_path():
    tello.takeoff()
    time.sleep(1)
    tello.move_up(34)#규격에 맞춰서 변경
    time.sleep(1)
    tello.move_forward(197)#규격에 맞춰서 변경
    time.sleep(1)
    tello.move_down(20)#규격에 맞춰서 변경
    time.sleep(1)
    tello.rotate_clockwise(94)#규격에맞춰서
                              변경
    time.sleep(1)
    tello.move_forward(40)#규격에 맞춰서 변경
    time.sleep(1)
    tello.move_up(35)#규격에 맞춰서 변경
    time.sleep(1)
    tello.set_speed(20)#규격에 맞춰서 변경
    tello.move_forward(168)#규격에 맞춰서 변경
    time.sleep(1)
    tello.land()
# 드론 경로를 따라가는 스레드 시작
thread = threading.Thread(target=drone_path)
thread.start()
```

5. 드론 비행 중 객체 인식 및 영상 출력

```
while True:
    frame = tello.get_frame_read().frame
    if frame is not None and frame.size != 0:
        # 이미지 크기 조정
        frame = cv2.resize(frame, (640, 480))

        # BGR을 RGB로 변환
        rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

        # 이미지를 tensor로 변환하고 차원 추가
        img = torch.from_numpy(rgb_frame).to(torch.device('cpu')).permute(2, 0, 1).unsqueeze(0)
        img = img.float() / 255.0  # 0 - 255를 0.0 - 1.0으로 변환

        # 객체 인식
        pred = model(img)

        # 결과 표시
        pred = non_max_suppression(pred, 0.25, 0.45, None, False, max_det=1000)
        for i, det in enumerate(pred):  # 이미지당 탐지 결과
            if len(det):
                det[:, :4] = scale_boxes(img.shape[2:], det[:, :4], frame.shape).round()
                for *xyxy, conf, cls in reversed(det):
                    label = f'{model.model.names[int(cls)]} {conf:.2f}'
                    plot_one_box(xyxy, frame, label=label, color=(255, 0, 0), line_thickness=3)

        # RGB를 다시 BGR로 변환하여 올바른 색상으로 표시
        bgr_frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        cv2.imshow('Tello Camera', bgr_frame)

        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        print("No frame received, waiting...")
        sleep(1)
```

6. 종료 및 정리

```
# 카메라 스트리밍 중지
tello.streamoff()
cv2.destroyAllWindows()

# 드론 스레드가 끝날 때까지 기다림
thread.join()
```

3.3 드론 제작 중 애로기술 및 해결과정 소개

- 비행 안정성 문제

문제점 :
초기 드론 테스트 단계에서 비행 중 드론이 흔들리거나, 바람에 의해 쉽게 영향을 받는 문제를 겪었다. 이는 특히 실외 비행에서 큰 문제로 작용했다.
해결과정 :
고급 자이로스코프와 가속도계를 통합하여 드론의 위치와 자세를 실시간으로 모니터링하고, 이를 바탕으로 비행 안정화 알고리즘을 개발했다. 또한, 비전 포지셔닝 시스템을 활용하여 드론이 주변 환경을 인식하고 안정적인 비행을 유지할 수 있도록 했다.

- 객체 인식 정확도

문제점 :
드론의 객체 인식 기능이 다양한 환경에서 일관되게 작동하지 않아, 장애물 회피나 목표물 탐지에 어려움을 겪었다.
해결과정 :
객체 인식 알고리즘을 개선하고, 다양한 환경에서의 데이터셋을 수집하여 딥러닝 모델을 재학습시켰다. 이를 통해 인식 정확도를 높였고, 드론이 다양한 조명 조건과 배경에서도 안정적으로 객체를 인식 할 수 있도록 했다.

- 통신 거리 및 신호 간섭

문제점 :
드론과 조종기 간의 통신 거리가 제한적이었고, 특히 도심지에서 신호 간섭으로 인해 조종이 어려운 상황이 발생했다.
해결과정 :
통신 모듈을 개선하여 더 긴 거리에서도 안정적인 통신이 가능하도록 했다. 또한, 주파수 대역을 다변화하고, 신호 간섭을 최소화하는 기술을 도입하여 통신의 신뢰성을 높였다.

- 바람에 의한 드론의 흔들림

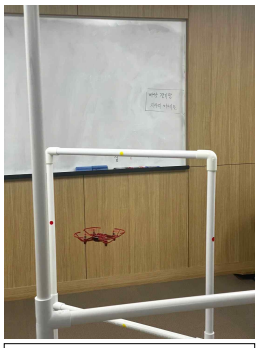
문제점 :
드론이 비행 중 바람에 의해 흔들리거나, 비행 경로가 벗어나는 문제가 발생했다. 특히 강풍이 불거나 불규칙한 바람이 부는 환경에서 드론의 안정성이 크게 떨어져 정확한 비행이 어렵고, 제어가 어려워질 수 있다. 이는 특히 촬영용 드론이나 정밀 작업을 수행하는 드론에서 큰 문제로 작용했다.
해결과정 :
통신 모듈을 개선하여 더 긴 거리에서도 안정적인 통신이 가능하도록 했다. 또한, 주파수 대역을 다변화하고, 신호 간섭을 최소화하는 기술을 도입하여 통신의 신뢰성을 높였다.

3.4 대회 참가 전 연습 주행 시 드론의 작업 능력 및 주행시간

- 연습 주행

(연습장 제작 후 주행 모습)

주행 중 - 호버링



주행 중 - 회전

주행 완료 - 도착

주행시간 : 약 28초

Ⅳ. 결론 및 향후 연구 방향

4.1 참가 팀만의 고유한 자랑거리 소개

a. 혁신적인 기술 개발 능력

- 최첨단 객체 인식 알고리즘 :

월등한 성능을 보이는 고급 객체 인식 알고리즘을 개발했다. 이를 통해 드론의 자율비행 성능을 크게 향상시켰다.

- 실시간 데이터 처리 시스템 :

실시간으로 대용량 데이터를 처리하고 분석할 수 있는 시스템을 구축하여, 드론의 비행 중 실시간 의사 결정을 가능하게 했다.

b. 다양한 드론 코드 작성

- 키보드를 사용한 드론 주행 :

키보드의 여러 가지 키를 사용하여 컨트롤러의 조종뿐만이 아닌 새로운 주행 방법 개발

- 얼굴인식을 이용한 드론 주행 :

코드 실행 시 화면에 모니터링이 실행되고 얼굴을 인식하여 인식된 사람이 움직이는 곳으로 이동하여 주행 가능하게 개발

- 모션인식을 이용한 드론 주행 :

손을 활용한 모션을 학습시키고 코드 실행 시 화면에 모니터링이 실행되고 미리 입력해 놓은 손동작으로 주행을 가능하게 개발

c. 뛰어난 성과

드론 부스 운영 :

- 키보드를 사용한 드론 주행, 모션인식, 얼굴인식을 활용하여 부스 체험하는 사람들에게 드론의 기능을 설명할 수 있는 경험을 얻었다.

4.2 대회를 통해 얻게 된 지식 및 향후 참가팀원들의 계획

a. 최신 기술 동향 파악

대회를 통해 드론 기술의 최신 동향과 발전 방향을 파악할 수 있었다. 특히, 객체 인식, 자율 비행 알고

리즘, 실시간 데이터 처리 등 다양한 최신 기술을 접하고 학습했다. 이를 통하여 드론뿐만이 아닌 자율주행이 가능한 여러 가지 이동 수단을 활용하여 훈련시켜, 다른 대회에 참가할 수 있다.

b. 팀워크와 협업의 중요성
대회 준비 과정에서 팀원들 간의 협업과 커뮤니케이션의 중요성을 다시 한번 느끼게 되었고, 각자의 역할을 명확히 하며, 효율적인 협업을 통해 최상의 결과를 도출하는 방법을 배웠다.

c. 기술적 한계와 개선점 파악
대회를 통해 기술의 현재 한계와 개선이 필요한 부분을 명확히 파악할 수 있었다. 이를 바탕으로 향후 개발 방향을 구체적으로 설정할 수 있는 계기가 되었다.

4.3 연구 방향

a. 프로토타입 개선 및 실용화
대회에서 사용한 프로토타입을 개선하여 상용화 가능한 제품으로 발전시킬 계획이다. 심사위원분들의 피드백을 반영한 디자인 개선과 안정성 테스트를 통해 실용성을 높일 것이다.

b. 새로운 도전과 프로젝트
대회를 통해 얻은 경험을 바탕으로 새로운 프로젝트에 도전할 계획이다. 특히, 농업, 물류, 인프라 점검 등 다양한 응용 분야에서 드론 기술을 적용할 수 있는 프로젝트를 발굴하고 추진할 것이다.

c. 차기 대회 준비
다음 대회를 대비하여 더욱 체계적으로 준비할 계획이다. 대회에서 얻은 경험을 바탕으로 전략을 수립하고, 부족한 부분을 보완하여 더 나은 성과를 목표로 할 것이다.

V. 참고문헌

5.1 대회준비를 위해 참고한 교재 혹은 사이트 소개

[1] 드론 AI 알고리즘, 센서, 프로그램 관련 교재 - 텔로에듀 TT 드론코딩 AtoZ
텔로 드론은 사용자가 비행을 즐기는 동안 프로그래밍의 기초를 쉽게 배울 수 있도록 SCRATCH, 앱인벤터, SWIFT, PYTHON 등의 프로그래밍 언어를 통하여 코딩을 할 수 있게끔 만들어졌다. 텔로 드론은 또한 간단한 블록기반 비주얼 프로그래밍도 가능하다.
앞으로 우리가 관심 가져야 할 AI(인공지능) 기반 사물인식 등 머신비전 활용법과 기체 모니터링, 컨트롤 및 군집비행 구현 등의 드론 프로그래밍 기술에 대한 매우 구체적인 내용들을 기술하고 있다.
[2] YouTube 등 기타 웹 사이트를 통해 참고한 사이트 소개
장애물 누끼 따기 사이트 – roboflow
https://roboflow.com/
장애물 객체 인식 훈련 사이트 – colab(코랩)
https://colab.research.google.com/?hl=ko
Tello 드론 사용 설명서 사이트
https://manuals.plus/ko/ryze/tello-drone-manual
RYZE – 드론 사양 설명서 사이트
https://www.ryzerobotics.com/kr/tello/specs