https://doi.org/10.7840/kics.2023.48.2.241

짐벌 카메라 각도 제어와 객체 인식을 사용한 UAV 자동착륙 시스템

강 호 현◆, 신 수 용°

UAV Automatic Landing System Using Gimbal Camera Angle Control and Object Detection

Ho Hyun Kang◆, Soo Young Shin°

요 약

본 논문은 짐벌 카메라의 각도 제어 및 딥러닝 기반 객체 인식을 활용한 UAV 자동 착륙 시스템을 제안한다. UAV에 장착된 카메라를 사용하여 딥러닝 기반 객체 인식을 통해 목적지 착륙 패드를 둘러싸는 표시 영역(Bounding box)을 기준으로 운용 중인 UAV의 위치를 제어하도록 한다. 제안된 방식에서는 별도의 카메라가 필요 없이 기존의 UAV 임무 장비 중의 하나인 짐벌 카메라를 사용해 카메라 각도를 하단을 바라보게끔 제어하고, UAV에 장착된 컴패니언 컴퓨터를 사용해 UAV을 제어한다. 제안된 방식에서 딥러닝 기반 객체인식은 Yolo v4 tiny를 사용했고, 드론제어를 위해 ROS를 사용하여 구현되었고, 성능 평가를 위해 착륙 지점과 착륙 패드 간의 거리를 측정하였고 모의실험과 실제 실험을 통해 효율성을 입증하였다.

키워드 : 무인 이동체, 자동 착륙, GPS, 객체 인식, 딥러닝
Key Words : UAV (Unmmand Aerial Vehicle), Auto landing, GPS (Global Positioning System), Object Detection, Deep Learning

ABSTRACT

In this paper, we propose a UAV autonomous landing system using the angle control of a gimbal camera and object detection based on deep learning. It controls the position of the UAV in operation based on the bounding box surrounding the destination landing pad through deep learning-based object detection using the camera mounted on the UAV. In the proposed method, without the need for a separate camera, a gimbal camera, one of the existing UAV mission equipment, is used to control the camera angle to look downward, and a companion computer mounted on the UAV is used to control the UAV. In the proposed method, deep learning-based object detection used Yolo v4 tiny, implemented using ROS for drone control, and measured the distance between the landing point and the landing pad for performance evaluation. proved.

※ "본 연구는 과학기술정보통신부 및 정보통신기획평가의원의 ICT혁신인재4.0 사업의 연구결과로 수행되었습니다. (IITP-2022-RS-2022-00156394)"
◆ First Author : Kumoh National Institute of Technology, 20216106@kumoh.ac.kr, 학생회원
° Corresponding Author : Kumoh National Institute of Technology, wdragon@kumoh.ac.kr, 종신회원
논문번호 : 202211-276-D-RU, Received November 9, 2022; Revised December 2, 2022; Accepted December 9, 2022

Ⅰ. 서 론

최근 무인 이동체 (Unmanned Aerial Vehicle, UAV)를 활용한 자율 임무 수행에 대한 연구 (재난 지역 탐사, 인명 구조, 객체 추종, 특정지역 감시 등)들이 많이 진행되고 있다[1,2]. 특히, 이러한 연구는 사람이 진입할 수 없거나 인력이 부족한 상황에서 많이 활용되고 있다. UAV는 취미용, 농업용, 군사용 등 다양한 분야에서 사용되고 있는데 아직까지는 사람이 직접 제어하여 운용하고 있다. 중앙관제소 (Ground Control System, GCS)로 제어하더라도 GPS (Global Positioning System)에서 발생하는 오류로 인하여 약 1m 이상의 착륙 오차가 발생한다[3]. 그렇기 때문에 대부분 수동으로 목표 지점에 착륙을 시도한다. 조종사 대부분의 의견에 따르면 이 또한 날씨의 영향 혹은 개인 조종 능력의 영향을 많이 받는 것으로 보고 된다. 정밀한 착륙을 위한 안정성 확보에 대한 시간이 많이 투입돼야 하므로 작업 효율이 낮다.

[4]에서는 마커 기반 자동 착륙 시스템이 제안되었는데 이러한 방식의 경우 UAV과 착륙 패드 간 거리가 일정 이상 벌어지면 마커를 감지하지 못하는 문제가 발생한다. 이를 개선하기 위해 딥러닝 기반 자동 착륙을 위해 착륙 패드를 감지하기 위한 착륙용 카메라를 하단에 별도로 UAV에 장착하는 방식이 제안되었으나, 이로 인한 페이로드 증가, 배터리 소모, 운용 시간 감소 등의 추가적인 문제가 발생할 수 있다[5].

본 논문에서는 이러한 문제를 해결하기 위해 기존 UAV에 장착되어 있는 임무용 카메라 짐벌을 사용하여 객체 인식을 하는데 사용하였다. 장착된 각도 변경이 자유로운 카메라를 통해서 딥러닝 (Deep Learning, DL) 기반 객체 인식을 사용하여 GPS 단독 착륙 시스템 대비 오차를 줄일 수 있는 시스템을 제안한다. 마커를 이용한 착륙 시스템과 대비하여 정확도 부분에서 손해가 있을 수 있지만 장거리에서부터 착륙 패드를 인식하여 GPS 대비 오차를 줄일 수 있다. 추가적으로 카메라의 줌 기능을 통해 정확도를 보완한다. 본 논문에서는 복잡도가 낮고 편의성을 높인 CNN (Convolution Neural Network) 기반 YOLO (You Only Look Once)를 사용한다[6,7]. UAV의 배터리 및 탑재 한계로 인하여 컴패니언 컴퓨터(Companion Computer)을 사용하고 이에 따라 YOLO의 v4 tiny를 이용하여 딥러닝을 접목시킨다.

논문 구성은 다음과 같다. 2장에서는 본 논문의 시스템 시나리오 및 구성, 알고리즘에 대하여 정리하고 3장에서는 실험 환경 및 실험 방법을 설명하고 각 오차를 비교 및 분석한다. 마지막으로 4장에서는 논문의 결론을 맺는다.

Ⅱ. 본 문

2.1 시스템 시나리오

$$Altitude.z \times \tan\theta \tag{1}$$

본 논문에서는 제안하는 자동 착륙 시스템의 시나리오를 그림 1과 같이 구성한다. 카메라의 기본 각도는 하단 60도를 바라보게끔 설정하였다. 그림 1. a를 보면 UAV에 탑재된 카메라가 착륙 패드를 감지하고 이를 향해 대각선 방향 직선으로 착륙을 수행한다. 시스템에서 제안하는 특정 고도에 도달할 경우 그림 1. c 그림에서처럼 드론이 위치하게 되는데 이때 착륙 패드 위쪽으로 이동한다. 착륙 패드와 드론과의 직선 평면 거리를 추정하기 위해 수식 (1)의 삼각함수를 활용하여 착륙 패드 위쪽으로 이동한다. 이와 동시에 그림 1. b처럼 카메라의 각도를 90도 즉 수직으로 자동 제어되며 착륙을 진행한다.

2.2 시스템 모델

본 논문의 시스템 모델은 그림 2과 같이 GCS, VPN 서버, UAV로 구성한다. 기존의 5G 망에서 GCS와 UAV의 통신을 내부적으로 구성하기 위해 VPN (Virtual Private Network) 서버를 사용하여 영상을 송수신하고 임무 수행 명령을 전송한다[8]. 탑재된 컴패니언 컴퓨터를 통해 UAV를 제어하고 카메라를 통해 얻은 영상 정보를 기준으로 착륙 패드를 식별하여 자동 착륙을 수행한다.

2.3 VPN 서버

UAV의 고도가 높아짐에 따라 Wi-Fi 방식으로는 연결성의 한계점이 명확하므로 5G 망에서 VPN 서버를 구축하여 사용한다. 점대점 연결을 가능하게 하기 위하여 외부망이 아닌 내부망을 사용할 필요가 있다. 이에 따라 VPN 서버를 이용하여 하나의 내부망으로 연결하여 UAV를 제어하고 영상은 송수신한다. VPN 서버를 사용하기 위하여 데스크톱에 오픈소스 VPN 프로토콜인 OpenVPN을 사용하여 GCS와 UAV를 연결한다[9].

이처럼 5G를 사용하여 Wi-Fi 통신보다 연결 안정성을 확보하여 GCS와 UAV의 통신이 가능하지만, 이 또한 기지국의 송신 범위에 따라 제한될 수 있다.

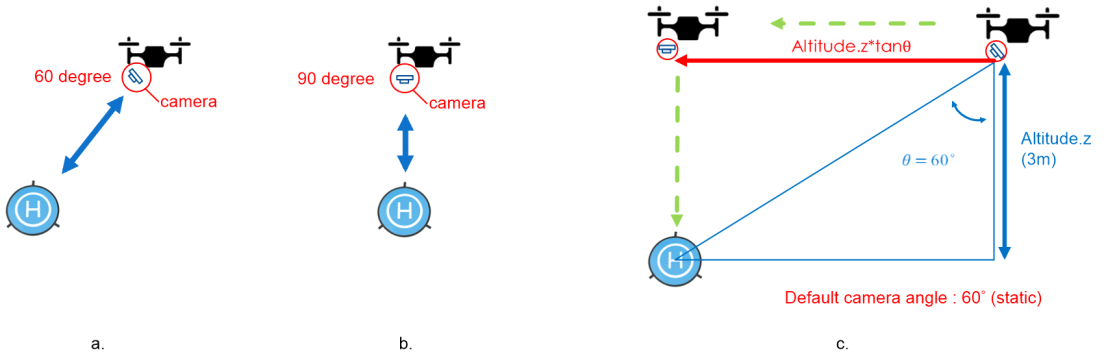

그림 1. 착륙 시나리오
Fig. 1. Landing scenario

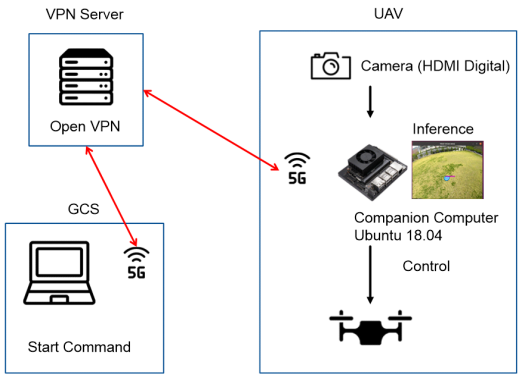

그림 2. 시스템 모델
Fig. 2. System Model

2.4 시스템 알고리즘

착륙 알고리즘은 그림 4에서 Pseudo 코드로 서술한다. 그림 3에서 선언된 변수를 확인할 수 있다. 카메라의 프레임과 Bounding box는 각각 0,0 / 0,0으로 지정하고 이를 기준으로 알고리즘을 수행한다. 객체 인식을 통하여 감지할 확률(decP)이 50% 이상일 경우 알고리즘이 시작되도록 초기 설정한다.

알고리즘은 카메라 프레임 중심점을 기준으로 바운딩 박스 중심점의 X, Y 좌표 위치에 따라 드론의 움직임을 제어한다.

```
AYmin = yMiddle <yframeMiddle × minV
AYmax = yMiddle >yframeMiddle × maxV
AXmin = xMiddle <xframeMiddle × minV
AXmax = xMiddle >xframeMiddle × maxV
conG = gimbal_check == False && position.z >= H
```

그림 3. 알고리즘 변수 선언
Fig. 3. Algorithm variable declaration

Algorithm 1 Landing Algorithm

```
while Box.probability >= decP do
    if position.z < LandH then
        land()
    end if
    if XDifferenceAbsolute < xframeMiddle × framV then
        if conG && AYmin then
            move(disU, 0, 0)
        else if conG && AYmax then
            move(0, 0, -disU)
        else if conG then
            move(position.z × tan(θ)/decV, 0 , position.z/decV)
        else if position.z <=H then
            move(position.z × tan(θ))
            gimbal_check = True
            change to gimbal angle
        else
            if AYmin then
                move(disU, 0, 0)
            else if AYmax then
                move(-disU, 0, 0)
            else
                move(0, 0, -disU)
            end if
        end if
    else
        if gimbal_check == False && AXmax then
            move_angle(-angU)
        else if AXmax then
            move(0, disU, 0)
        else if gimbal_check == False && AXmin then
            move_angle(angU)
        else if AXmin then
            move(0, -disU, 0)
        end if
    end if
end while
```

그림 4. 착륙 알고리즘
Fig. 4. Landing Algorithm

감지 이후 화면 중앙으로부터의 XDifferenceAbsolute (카메라 화면 x축 중심으로부터 바운딩 박스의 x축 중심이 얼마나 떨어져 있는지에 대한 절대값) 값이 xframeMiddle(카메라 화면 기준 X축 중심값) 값으로부터 frameV(오차 값) 값을 포함한 범위보다 작을 경우 바운딩 박스는 현재 그림 5의 X축 중심 붉은 점선 안쪽에 위치하게 되므로 카메라 화면 중심점과 바운

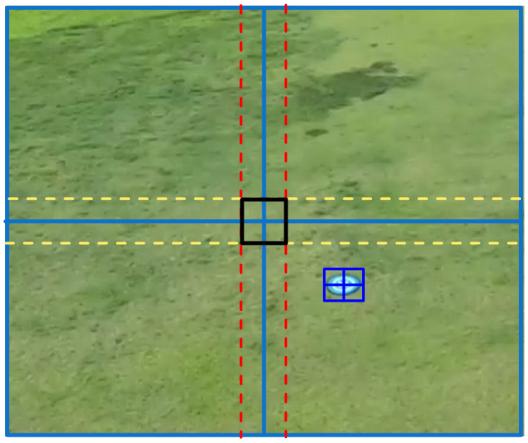

그림 5. UAV 카메라 화면
Fig. 5. UAV Camera Scene

딩 박스 중심점이 X축 중앙 정렬 완료된 것으로 판단하여 Y축 제어를 진행한다. 그렇지 않을 경우에는 X축 제어를 진행한다.

Y축 제어를 하기 위해서 conG 값을 통해 gimbal_check(짐벌 각도 변경 유무 확인, 각도가 변경되었을 경우 True) 값을 확인하여 값이 False인 것과 동시에 positon.z(현재 UAV의 고도)가 H (특정 고도) 값 이상일 때 AYmin 인 경우 yMiddle (바운딩 박스 Y 중심점) 값이 minV(최소 오차 값) 값을 포함한 yframeMiddle(카메라 화면 기준 Y축 중심값) 값보다 작으므로 그림 5에서 바운딩 박스 중심점이 노란색 점선보다 위쪽에 위치하게 된다. 이때 UAV는 disU (지정한 거리) 값만큼 앞으로 이동하여 Y축 정렬을 시도한다.

반대로 conG 값을 충족하고 이와 동시에 AYmax 값일 때에는 yMiddle 값이 maxV(최대 오차 값) 값을 포함한 yframeMiddle 값보다 크므로 그림 5에서 바운딩 박스 중심점이 노란색 점선보다 아래쪽에 위치하게 된다. 이 경우 UAV는 disU 값만큼 하강하여 Y축 정렬을 시도한다.

또한 conG 값만 충족할 경우에는 UAV는 수식 (1) 값에서 decV(한 번에 이동 할 수 있는 거리 제한을 위한 값) 값을 나눈 값만큼 앞으로 이동하는 것과 동시에 position.z 값에서 동일한 decV 값을 나눈 값만큼 이동하게 되므로 즉 UAV와 착륙 패드의 최단거리인 대각선 아래로 이동하게 된다.

그림 5의 검은색 박스 안에 바운딩 박스 중심점을 유지하면서 즉 X축과 Y축을 정렬하면서 착륙 진행 도중 position.z 값이 H 값 이하가 될 경우 UAV는 수식 (1) 값만큼 앞으로 이동하여 착륙 패드 바로 위로 이동하게 된다. 이와 동시에 gimbal_check 값은 True 값으로 재정의되고 카메라 짐벌 각도가 90도 즉 수직으로 하단을 향하게 된다.

이후 gimbal_check 값이 True이므로 현재 드론의 위치는 그림 1의 b와 같은 모습으로 착륙패드 위쪽에 위치하게 된다.

이때 AYmin 값을 충족하면 기존과 같이 disU 값만큼 앞으로 이동하게 되지만, AYmax 값을 충족할 때에는 UAV가 disU 값만큼 뒤로 이동하게 된다. AYmin 값과 AYmax 값 둘 다 충족하지 않는 경우 그림 5의 검은색 상자 안에 착륙 패드 중심점이 위치하고 있다고 판단하여 UAV는 disU 값만큼 하강하게 된다.

X축 정렬은 XDifferenceAbsolute 값이 xframeMiddle 값으로부터 frameV 값을 포함한 범위보다 클 경우 바운딩 박스 중심점은 그림 5의 붉은 점선 바깥쪽에 위치하게 되므로 X축 정렬이 되지 않았다고 판단하여 X축 제어를 진행한다.

gimbal_check 값이 False이며 AXmax 값일 경우 xMiddle(바운딩 박스 X 중심점) 값이 maxV(최대 오차 값) 값을 포함한 xframeMiddle(카메라 화면 기준 x축 중심값) 값보다 크므로 그림 5에서 바운딩 박스 중심점이 붉은 점선 보다 우측에 있다고 판단하여 UAV를 angU 값만큼 시계방향으로 회전시킨다.

gimbal_check 값이 True일 경우 UAV는 disU 값만큼 우측으로 이동한다.

반대로 AXmin 값일 경우 xMiddle 값이 minV(최소 오차 값) 값을 포함한 xframeMiddle 값보다 작으므로 그림 5에서 바운딩 박스 중심점이 붉은 점선 보다 좌측에 있다고 판단하여 UAV를 angU 값만큼 반시계방향으로 회전시킨다.

gimbal_check 값이 True일 경우 UAV는 disU 값만큼 좌측으로 이동한다.

그림 5의 검은색 박스 안에 바운딩 박스 중심점을 유지하며 X축과 Y축을 정렬하면서 착륙을 진행한다. 이때 poston.z 가 LandH 보다 작을 경우 착륙 명령을 보내어 착륙을 마무리한 뒤, 알고리즘을 종료한다.

Ⅲ. 실 험

3.1 실험환경 및 실험장비

본 논문에 사용된 UAV 기체는 그림 6과 같이 구성하였다. 기체 프레임은 Holybro 사의 X500 v2를 사용하였고, 착륙 패드를 감지하고 추론하기 위해 사

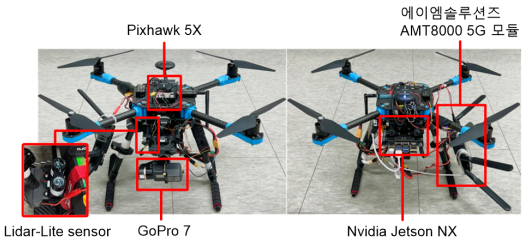

그림 6. UAV 구성
Fig. 6. Configuration of drone

용된 카메라는 GoPro7을 선택하여 장착하였다. UAV의 핵심인 FC(Flight Controller)는 Pixhawk 5X를 사용하였다. 정확한 고도를 측정하기 위해 광학 원거리 측정 센서인 Lidar-Lite 센서를 장착하여 고도를 측정하였으며, UAV를 제어하기 위해 사용된 컴패니언 컴퓨터는 Nvidia 사의 Jetson NX를 선택하여 사용하였다. 5G 통신을 하기 위해 사용된 모듈은 에이엠 솔루션즈 의 AMT8000 모듈을 사용하였다.

본 실험에 사용된 데이터 세트는 UAV에서 촬영한 영상을 프레임 단위로 분할하여 2,700여 장의 사진으로 나눈 다음에 augmentation을 진행하여 22,000여 장의 데이터 세트를 확보한 뒤 학습하고 진행하였다. 이후 학습된 가중치 파일을 Yolo v4 tiny를 사용하여 착륙 패드를 식별하는 데 사용하였다.

실험에 사용된 착륙 패드 사이즈는 지름이 약 76cm의 착륙 패드를 사용하였다.

비교 대상인 GPS 정확도 테스트는 UAV가 착륙 패드에서 비행을 시작하여 일정 거리를 이동한 뒤 내장되어 있는 GPS를 사용하여 fail safe 모드 중 하나인 return home 모드를 사용하여 측정하였다. 객체 인식 기반 자동 착륙 시스템은 UAV가 일정 거리 이상부터 착륙 패드를 감지하여 알고리즘을 수행하고, 착륙 오차는 UAV가 착륙한 후 착륙 패드 정 중앙점으로부터 UAV 중앙점까지 직선거리를 측정하였다.

표 1. 실험 환경
Table 1. Experimental Environment

Spec.	Configuration
GCS Laptop	I7-10750H
Drone	X500 v2
Companion Computer	Nvidia Jetson Xavier NX
FC	Pixhawk 5X
model	Yolo v4 timy
camera	GoPro 7

그림 7. 실험 환경
Fig. 7. Experiment environment

이렇게 각각 10번씩 측정하여 착륙 오차, 걸린 시간을 측정하여 각각 평균값을 구해 비교하였다. 표 1은 본 논문에 사용된 시스템을 정리한 표이다.

그림 7에서 실제 비행 실험 중인 환경 모습을 확인할 수 있다.

3.2 실험방법

GPS 착륙 패드 위에서 UAV를 이륙한 뒤 랜덤한 거리로 이동한다. 고도는 약 7미터에서 9미터 사이로 지정하였으며, 거리도 착륙 패드로부터 약 10미터 정도 떨어진 랜덤한 위치로 이동한 뒤, return home 모드를 사용하여 처음 이륙한 위치로 착륙하는 방식으로 진행하였다. 이후 착륙 패드 정 중앙점부터 UAV 중앙점까지 직선거리 오차를 측정한다.

3.3 실험결과 비교 및 분석

3.3.1 GPS 기반 자동 착륙

표 2는 GPS만 사용하여 자동 착륙을 시도했을 때 걸린 시간과 착륙 패드와의 직선거리 오차를 측정한 표이다. 착륙 패드와 UAV까지의 최소 거리는 40cm

표 2. GPS 착륙 결과
Table 2. GPS land result

Count	Error(cm)	Time(sec)
1	76	19
2	188	20
3	137	23
4	203	21
5	703	20
6	90	21
7	40	20
8	260	24
9	170	24
10	250	23

그림 8. GPS 착륙 결과
Fig. 8. GPS Land Result

이고 최대 703cm까지 오차가 발생하는 것을 확인할 수 있었다. 걸린 시간은 최소 19초 최대 24초로 균일한 시간을 보여주고 있다.

그림 8은 GPS를 사용한 착륙 결과 모습이다.

3.3.2 객체인식을 통한 자동착륙

표 3은 객체 인식 기반 자동 착륙 이후 착륙 패드 정 중앙점과 UAV 정 중앙점의 직선거리 오차 결과표이다. 최소 8cm 최대 60cm의 오차가 측정되었으며, 시간은 최소 50초 최대 251초의 시간이 측정되었다.

그림 9는 객체 인식 기반 자동 착륙한 결과 모습이다.

표 3. 객체 인식 기반 자동 착륙
Table 3. Auto land base on object detection

Count	Error(cm)	Time(sec)
1	25	140
2	15	251
3	20	140
4	25	112
5	23	119
6	8	50
7	21	139
8	22	92
9	17	137
10	8	53

그림 9. 객체 인식 기반 자동 착륙
Fig. 9. Auto Land base on object detection

3.3.3 비교 및 분석

GPS를 사용한 착륙과 객체 인식을 사용한 자동 착륙거리 오차 및 시간을 표 4에 정리하였다. GPS를 사용하여 착륙했을 때 착륙 패드 정 중앙점과 UAV 정 중앙점의 직선거리 오차 평균은 약 211cm가 발생하였으며, 객체 인식을 사용하여 자동 착륙했을 때 오차 평균은 약 18.4cm로 매우 큰 차이를 보여주고 있다. 착륙하는 시간을 측정하면 GPS는 약 21초, 제안하는 기법은 약 123초로 차이가 있으나 정밀도 면에서 제안하는 기법이 뛰어난다. 또한 제안하는 기법은 GPS를 사용하지 않기 때문에 실내 UAV에도 적용이 가능하다.

그림 10에서는 제안하는 기법의 운용기록을 확인할 수 있다. 이를 통해 제안하는 기법을 통해 예상한 움직임으로 UAV가 비행하는지 확인할 수 있다. 먼저 감지한 시점부터 대각선 방향으로 이동한다. 그리고 수식 1번의 움직임으로 착륙 패드 바로 위로 이동한다. 마지막으로 착륙 패드와 UAV 위치를 맞추며 착륙을 마무리하는 것을 확인 할 수 있다. 수식 1번의 움직임 이후 착륙 패드 바로 위에서 UAV 위치를 맞추는 과정에서 감지 정확도가 떨어져 전체적인 속도가 오래 걸리는 것을 확인 할 수 있었다.

표 4. 착륙 오차 및 소요시간
Table 4. Landing accuracy and required time

	GPS	Object Detection
Average (cm)	211	18.4
STD	187	6.3
Time (sec)	21	123

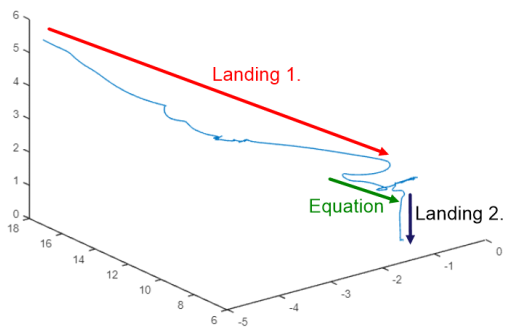

그림 10. 자동 착륙 경로
Fig. 10. Audo Landing trajectory

Ⅳ. 결 론

본 논문에서는 기존의 주로 사용되는 자동 착륙 방식인 GPS를 사용하여 자동 착륙하였을 때 발생하는 오차를 줄이기 위해 딥러닝 기반 객체 인식을 사용한

착륙 기법을 제안하였다. 제안된 방식은 별도의 추가 카메라 없이 하단을 바라보게끔 카메라 각도를 변경하여 자동 착륙을 구현할 수 있는 장점이 있다. 본 논문에서는 더 넓은 시야를 통한 착륙 패드 감지를 위해 기본 카메라 각도를 하단 60도로 설정하였다. 이후 착륙 시점에 따라 하단을 바라보게끔 변경하였다. 인식한 바운딩 박스를 기준으로 UAV를 제어하면서 착륙하는 알고리즘을 제안 및 구현하였다. 실제 비행 테스트를 진행하면서 착륙 정확도, 착륙에 소요되는 시간을 확인하였다.

본 논문에서는 착륙 패드를 감지한 것을 가정하여 실험을 진행하였으나 GPS 기반 경로 탐색 알고리즘 등과 결합하여 확장이 가능하다. 또한 객체 인식 정확도를 높이거나 마커를 결합하여 긴 착륙 소요 시간을 해결하고 착륙 정확도를 높일 수 있다. 이를 추후 연구로 진행하여 개선할 예정이다.

References

[1] A. Valsan, B. Parvathy, G. H. Vismaya Dev, R. S. Unnikrishnan, P. K. Reddy, and A. Vivek, “Unmanned aerial vehicle for search and rescue mission,” *ICOEI*, pp. 684-687, 2020. (https://doi.org/10.1109/ICOEI48184.2020.9143062)

[2] K. Nonami, “Drone technology, cutting-edge drone business, and future prospects,” *J. Robotics and Mechatronics*, vol. 28, no. 3, pp. 262-272, 2016. (https://doi.org/10.20965/jrm.2016.p0262)

[3] M. G. Wing, A. Eklund, and L. D. Kellogg, “Consumer-grade global positioning system (GPS) accuracy and reliability,” *J. Forestry*, vol. 103, no. 4, pp. 169-173, 2005. (https://doi.org/10.1093/jof/103.4.169)

[4] H. H. Kang and S. Y. Shin, “Precise drone landing system using aruco maker,” *J. KICS*, vol. 47, no. 01, pp. 145-150, 2022. (https://doi.org/10.7840/kics.2022.47.1.145)

[5] Y. K. Kim, D. Y. Choi, S. H. Paik, S. W. Jung, and D. N. Kim, “Implementation of deep learning based automatic landing system for docking on rotary wing drone contact charging stations,” *J. KIIT*, vol. 18, no. 10, pp. 45-53, 2020. (https://doi.org/10.14801/jkiit.2020.18.10.45)

[6] J. Redmon, et al., “You only look once: Unified, real-time object detection,” *IEEE Conf. CVPR*, pp. 27-30, 2016. (https://doi.org/10.1109/CVPR.2016.91)

[7] R. Iyer, et al., “Comparison of YOLOv3, YOLOv5s and MobileNet-SSD V2 for real-time mask detection,” *IRJET*, vol. 8, no. 7, pp. 1156-1160, 2021. (https://www.irjet.net/)

[8] P. Likhar, R. S. Yadav, and M. K. Rao, “Performance evaluation of transport layer VPN on IEEE 802.11 g WLAN,” *Trends in Netw. and Commun.: Int. Conf., NeCOM, WeST, WiMoN 2011,* vol. 197, pp. 407-415, Chennai, India, Jul. 2011. (https://doi.org/10.1007/978-3-642-22543-7_41)

[9] M. Feilner, *OpenVPN: Building and integrating virtual private networks*, Packt Publishing Ltd., 2006. (dl.acm.org/doi/abs/10.5555/1202604)

강 호 현 (Ho Hyun Kang)

2021년 2월 : 국립금오공과대학교 전자it융합과 졸업

2021년 2월~현재 : 국립금오공과대학교 IT융복합공학과 석사과정

<관심분야> 드론 응용, 로봇 제어

[ORCID:0000-0003-3601-3690]

신 수 용 (Soo Young Shin)

1999년 2월 : 서울대학교 전기공학부 졸업

2001년 2월 : 서울대학교 전기공학부 석사

2006년 2월 : 서울대학교 전기공학부 박사

2010년~현재 : 국립금오공과 대학교 전자공학부 교수

<관심분야> 차세대 무선통신 기술, 드론 응용, 블록체인, 머신러닝, 딥러닝, 혼합현실

[ORCID:0000-0002-2526-2395]