제 21회 로봇항공기 경연대회

가톨릭관동대학교 Victor
팀장: 이준희
지도교수: 조인제

INDEX

Part 1,

시스템 구성

Part 1,

하드웨어 구성

LIDAR SENSOR

RPLIDAR A2

IMAGE SENSOR

Realsense D435

MICRO CONTROLLER

OpenCR

USB

USB

Serial

Jetson Nano

Serial

Cube Orange+

Telemetry

Part 1,

드론 제원

자체 중량

4.8 kg

95g

190g

16g

240g

50g

60g

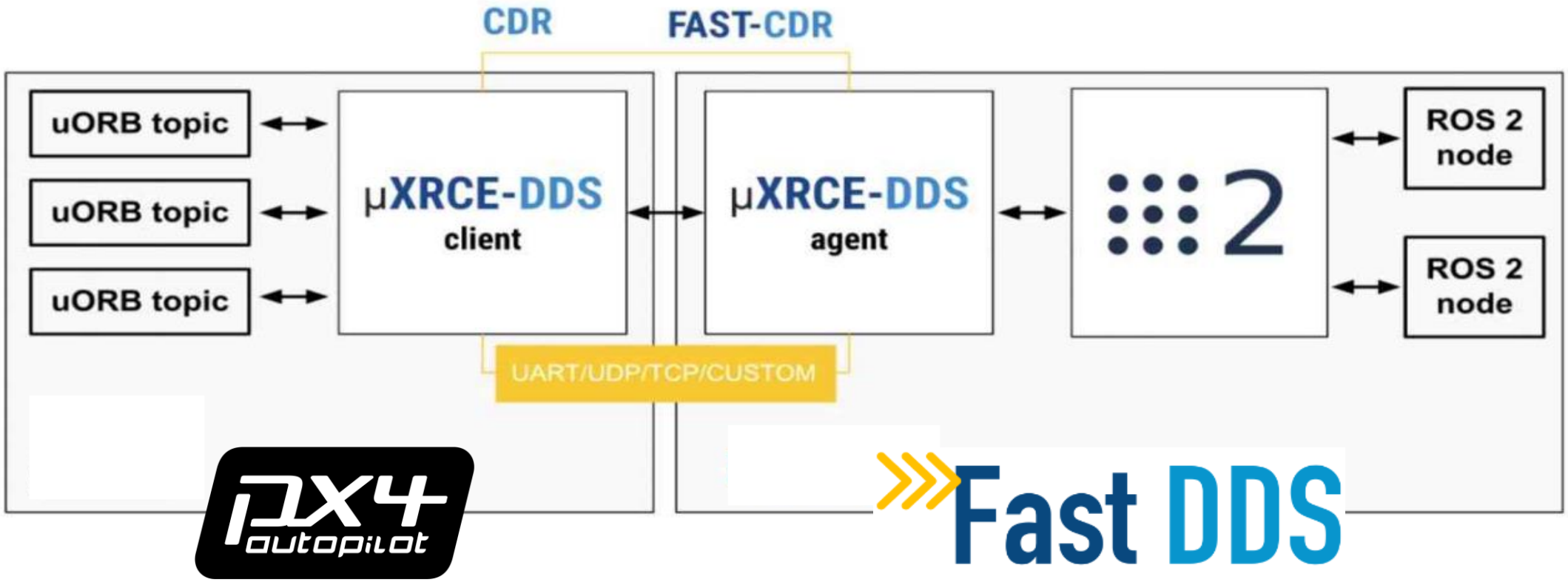
700g

탑재 중량

6kg

Part 1,

드론 제어 시스템 (1)



Part 1,

드론 제어 시스템 (2-자체 제작 로직)

- bridge.py
- circle_realworld.py
- forward_realworld.py
- forward.py
- get_gps_sub.py
- get_gps.py
- lidar_key.py
- lidar.py
- multy_square_simple.py
- multy_square.py
- multy_waypoint.py
- multy.py
- offboard.py
- outside_test.py
- outside.py
- point_one.py
- point_two.py
- speed.py
- standing_pid.py
- standing.py
- test_developing.py
- test_gps_point.py
- test.py
- trans.py
- waypoint_pid.py
- waypoint.py

```
class OffboardControl(Node):

    def __init__(self):
        super().__init__('minimal_publisher')
        qos_profile = QoSProfile(
            reliability=QoSReliabilityPolicy.RMW_QOS_POLICY_RELIABILITY
            durability=QoSDurabilityPolicy.RMW_QOS_POLICY_DURABILITY_TR
            history=QoSHistoryPolicy.RMW_QOS_POLICY_HISTORY_KEEP_LAST,
            depth=1
        )

        self.local_position_sub = self.create_subscription(
            VehicleLocalPosition,
            '/fmu/out/vehicle_local_position',
            self.local_position_callback,
            qos_profile)

        self.status_sub = self.create_subscription(
            VehicleStatus,
            '/fmu/out/vehicle_status',
            self.vehicle_status_callback,
            qos_profile)
        self.start_pub = self.create_publisher(Int32, "/start_detection
        self.start_pub2 = self.create_publisher(Int32, "/start_detectio
        self.error_sub = self.create_subscription(Int32, "/error_status
        self.cmd_vel_sub = self.create_subscription(Twist, 'cmd_vel', s
        self.publisher_vehicle_command = self.create_publisher(VehicleC
        self.publisher_offboard_mode = self.create_publisher(OffboardCo
        self.publisher_trajectory = self.create_publisher(TrajectorySet
        timer_period = 0.02  # seconds
        self.timer = self.create_timer(timer_period, self.cmdloop_callb

        self.takeoff_altitude = -4.0
        self.current_altitude = 0.0
        self.takeoff_complete = False
        self.armed = False
        self.nav_state = VehicleStatus.NAVIGATION_STATE_MAX
        self.dt = timer_period
        self.land_state = False
        self.stop_duration = 5.0  # seconds
```

```
jun@jun-Lenovo-Y520-15IKBN:~$ ros2 topic list
/fmu/in/obstacle_distance
/fmu/in/offboard_control_mode
/fmu/in/onboard_computer_status
/fmu/in/sensor_optical_flow
/fmu/in/telemetry_status
/fmu/in/trajectory_setpoint
/fmu/in/vehicle_attitude_setpoint
/fmu/in/vehicle_command
/fmu/in/vehicle_mocap_odometry
/fmu/in/vehicle_rates_setpoint
/fmu/in/vehicle_trajectory_bezier
/fmu/in/vehicle_trajectory_waypoint
/fmu/in/vehicle_visual_odometry
/fmu/out/failsafe_flags
/fmu/out/position_setpoint_triplet
/fmu/out/sensor_combined
/fmu/out/timesync_status
/fmu/out/vehicle_attitude
/fmu/out/vehicle_control_mode
/fmu/out/vehicle_global_position
/fmu/out/vehicle_gps_position
/fmu/out/vehicle_local_position
/fmu/out/vehicle_odometry
/fmu/out/vehicle_status
/parameter_events
/rosout
```

in offboard_mode, control what drone wants

drone move based on NED coordinate

arming or landing command

checking of real-time gps coordinate as latitude, longitude

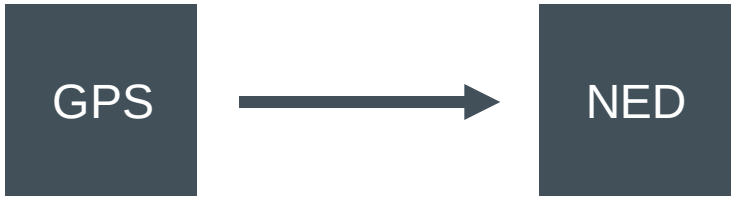
checking the real-time NED coordinate of drone

checking the real-time status of drone

Part 1,

드론 제어 시스템 (3)

```
---
timestamp: 1690367809307166
timestamp_sample: 1690367210727166
device_id: 0
lat: 473977508
lon: 85456074
alt: 488102
alt_ellipsoid: 488102
s_variance_m_s: 0.25
c_variance_rad: 0.5
fix_type: 3
eph: 1.0
epv: 1.0
hdop: 0.0
vdop: 0.0
noise_per_ms: 0
automatic_gain_control: 0
jamming_state: 0
jamming_indicator: 0
vel_m_s: 0.1264152526855468 8
vel_n_m_s: 0.0
vel_e_m_s: 0.0
vel_d_m_s: 0.0
cog_rad: 0.0
vel_ned_valid: true
timestamp_time_relative: 1015102721
time_utc_usec: 4355311139832201216
satellites_used: 16
heading: 5.515076353058539e-40
heading_offset: .nan
heading_accuracy: .nan
rtcm_injection_rate: .nan
selected_rtcm_instance: 0
```



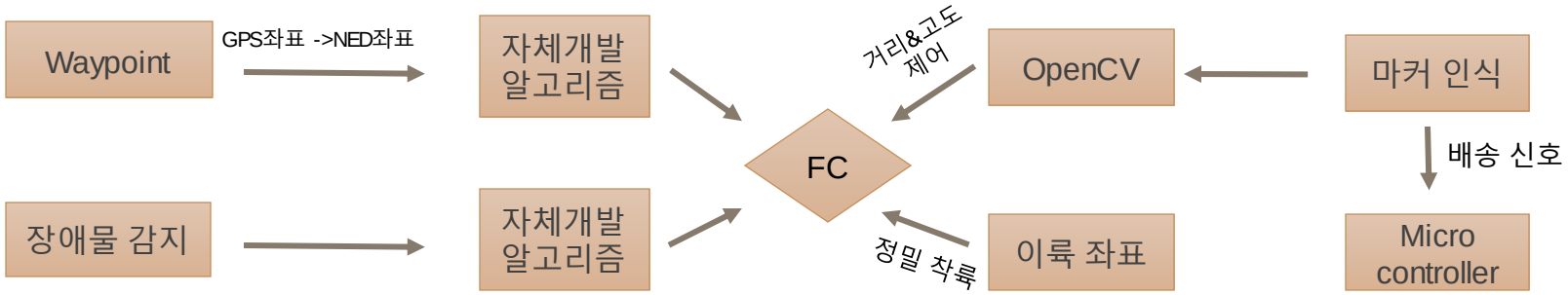

```
현재위치
47.3977773
8.5456078
목표지점
wp1
47.397 degree
8.545 degree
-86.28030000031828 m
-67.46580000010383 m
wp2
47.397121 degree
8.545111 degree
-72.84930000031409 m
-55.14480000057614 m
```

```
_gps
WP1, 위도(lat)를 입력하세요: 47.397000
WP1, 경도(lon)를 입력하세요: 8.545000
WP2, 위도(lat)를 입력하세요: 47.397121
WP2, 경도(lon)를 입력하세요: 8.545111
```

Part 1,

임무 시나리오

1. Waypoint 통과
2. 베란다 마커 인식
3. 택배 사출 및 자동착륙



1 시스템 구성

2 연구 개발 내용

3 시뮬레이션 & 비행 테스트 결과

4 개선방안 및 향후 계획

Part 2,

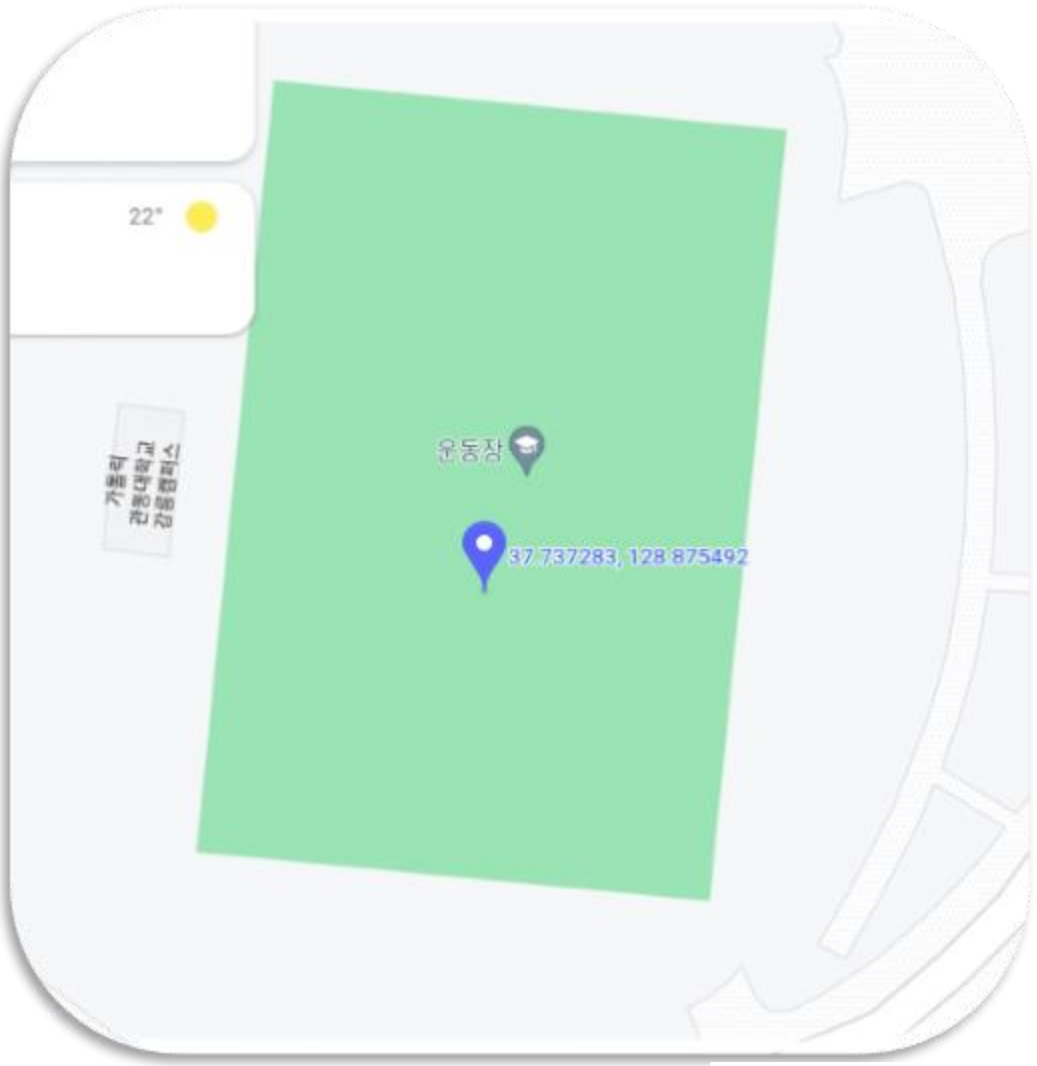
연구 개발 내용

Part 2,

장애물 회피(1)

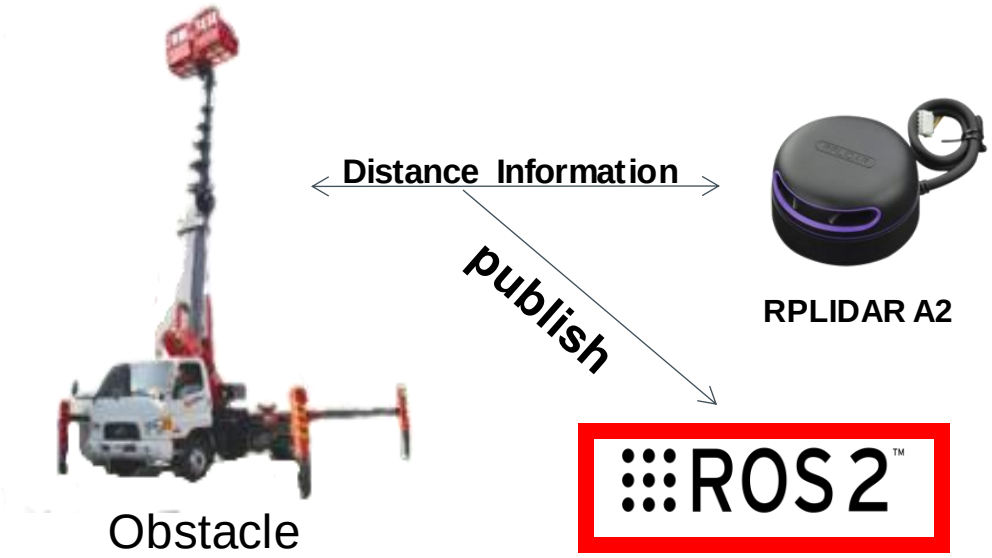
자체 개발 알고리즘

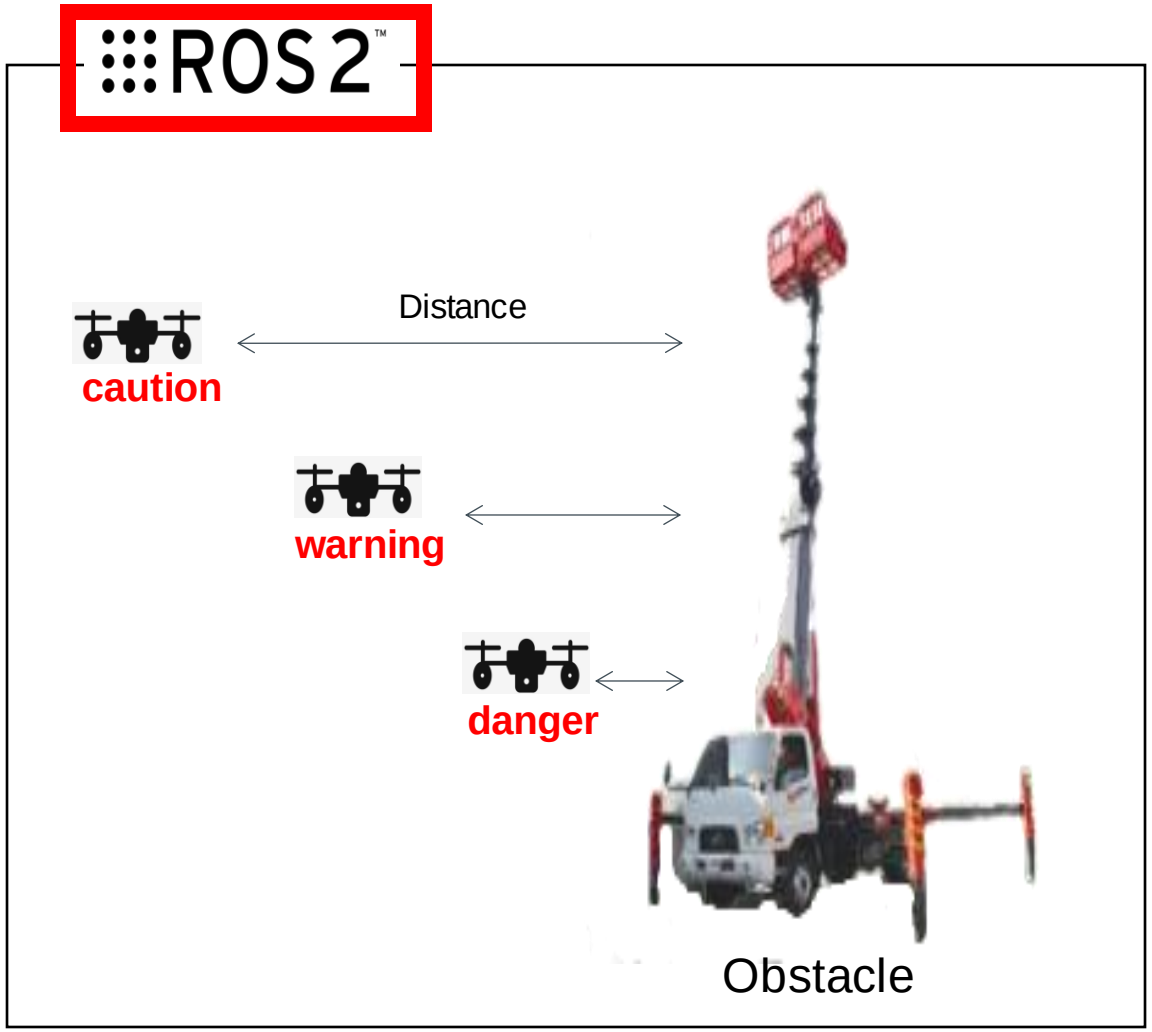
- Waypoint로 이동 중 장애물을 감지하게 되었을 때 회피 비행 실시
- 회피 후 드론의 위치를 기반으로 waypoint 좌표에 대한 최적의 경로(최소 비용)로 이동
- Waypoint는 GPS좌표를 NED 좌표로 변환하여 수행



Part 2,

장애물 회피(2)





Part 2,

장애물 회피(3)

```python
left_distances = msg.ranges[15:135]
right_distances = msg.ranges[225:344]

if 5.0 < left_distances < 7.0:
    self.obstacle_closed1 = True
    self.obstacle_detected1 =False
    self.status_dangered1 = False
    self.status_dangered2 = False
    self.obstacle_detected2 = False
    self.obstacle_closed2 = False
elif 4.0 < left_distances < 5.0:
    self.obstacle_detected1 =True
    self.obstacle_closed1 = False
    self.status_dangered1 = False
    self.status_dangered2 = False
    self.obstacle_detected2 = False
    self.obstacle_closed2 = False
elif left_distances < 3.5:
    self.status_dangered1 = True
    self.obstacle_detected1 =False
    self.obstacle_closed1 = False
    self.status_dangered2 = False
    self.obstacle_detected2 = False
    self.obstacle_closed2 = False
elif 7.0 < left_distances or 3.5 < left_distances < 4.0:
    self.status_dangered1 = False
    self.obstacle_detected1 = False
    self.obstacle_closed1 = False
    self.status_dangered2 = False
    self.obstacle_detected2 = False
    self.obstacle_closed2 = False

if 5.0 < right_distances < 7.0:            #0.50 < front
    self.obstacle_closed2 = True
    self.obstacle_detected2 =False
    self.status_dangered2 = False
    self.status_dangered1 = False
    self.obstacle_detected1 = False
    self.obstacle_closed1 = False
elif 4.0 < right_distances < 5.0:          #0.40 < front
    self.obstacle_detected2 =True
    self.obstacle_closed2 = False
    self.status_dangered2 = False
    self.status_dangered1 = False
    self.obstacle_detected1 = False
    self.obstacle_closed1 = False
elif right_distances < 3.5:                #front < 0.35
    self.status_dangered2 = True
```

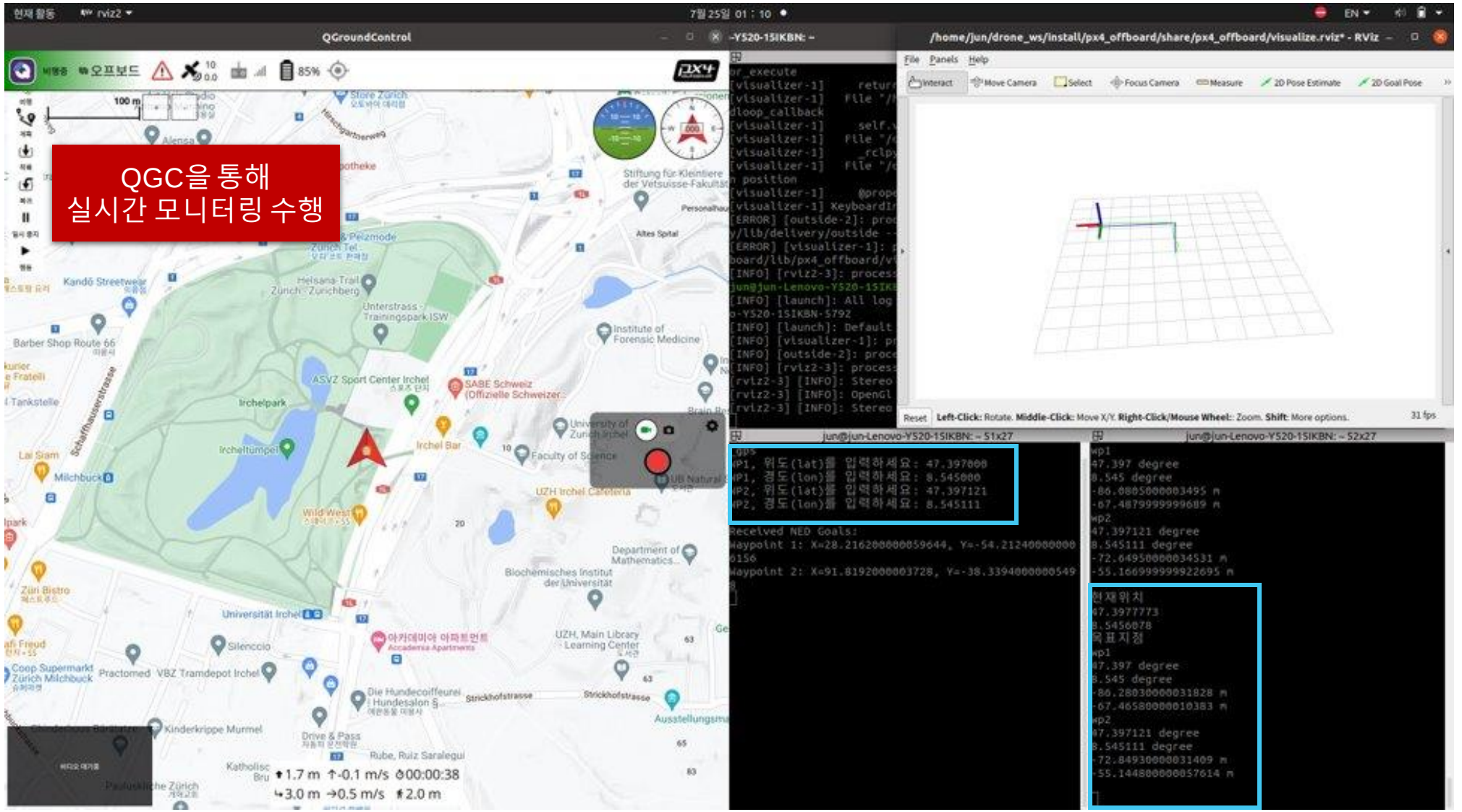
장애물회피 알고리즘

```
NAV_STATUS:  14
  - offboard status:  14
[INFO]: Obstacle closed. Starting avoidance.
[INFO]: Obstacle closed. Starting avoidance.
[INFO]: Obstacle closed. Starting avoidance.
[INFO]: Obstacle detected. Avoiding obstacle.
[INFO]: Obstacle detected. Avoiding obstacle.
[INFO]: Obstacle detected. Avoiding obstacle.
NAV_STATUS:  14
  - offboard status:  14
[INFO]: Obstacle detected. Avoiding obstacle.
[INFO]: Obstacle detected. Avoiding obstacle.
[INFO]: Obstacle detected. Avoiding obstacle.
[INFO]: Obstacle detected. Avoiding obstacle.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
NAV_STATUS:  14
  - offboard status:  14
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
[INFO]: Vehicle is dangerous. Moving back.
```

장애물 감지시 터미널

Part 2,

실시간 모니터링 (장애물회피)



Part 2,

마커 인식 및 접근 (1)

1

OpenCV를 이용한 색상 기반 탐지 수행

-해당 색상에 대한 마스크 생성

2

형태 기반 탐지 수행

-원하는 형태 기반을 탐지하도록 알고리즘 적용

3

필터링을 통한 후처리

-실제 환경에서 마커를 효과적으로 인식할 수 있도록 수행

4

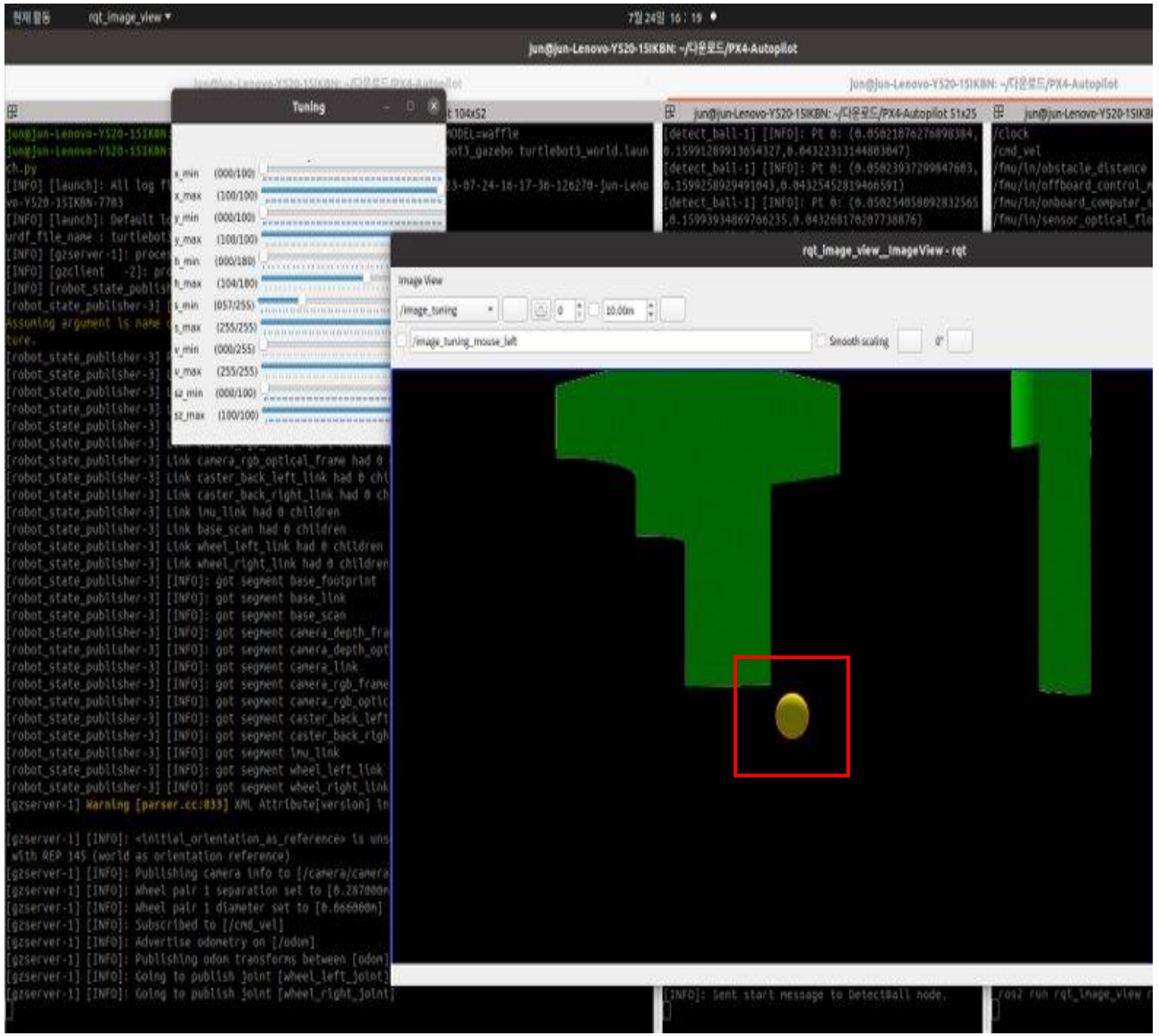
이미지 인식을 위한 Rudder의 증감 및 고도 하강

-이미지를 인식하도록 수색 작업 수행

5

이미지 좌표를 통한 드론의 위치 중앙화

-마커의 위치가 이미지의 중앙에 위치하도록 드론을 제어



Part 2,

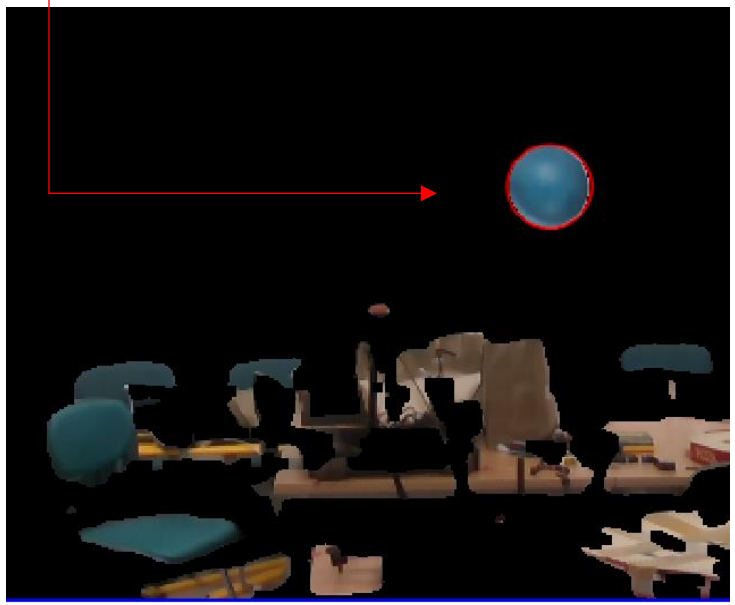
마커 인식 및 접근 (2)

approach

```
[INFO]: Target: -0.00021593850205601714
0.09010339914940227
hmm
[INFO]: Target lost
[INFO]: Target lost
[INFO]: Target lost
[INFO]: Target: -0.01677849365487928
0.0902310875581124
hmm
[INFO]: Target: -0.01677849365487928
0.0902310875581124
hmm
[INFO]: Target: -0.01677849365487928
0.0902310875581124
hmm
[INFO]: Target: -0.01677849365487928
0.0902310875581124
hmm
[INFO]: Target: -0.01677849365487928
0.0902310875581124
hmm
[INFO]: Target: -0.01677849365487928
0.0902310875581124
```

stop

```
[INFO]: Target: 0.029488201900479242
0.10125959899109635
Stop!!
[INFO]: Target: 0.09197004076614115
0.10120886477305323
Stop!!
[INFO]: Target: 0.09197004076614115
0.10120886477305323
Stop!!
[INFO]: Target: 0.09197004076614115
0.10120886477305323
Stop!!
[INFO]: Target: 0.09197004076614115
0.10120886477305323
Stop!!
[INFO]: Target: 0.09197004076614115
0.10120886477305323
```



Part 2,

마커 인식 및 접근 (3)

Detect

```
elif (self.target_dist >= self.max_size_thresh):
    msg.linear.x = 0.0
    print('Stop!!')
    self.state = 0
    # Check if this is the first time this condition is met
    if self.last_large_dist_time is None:
        self.last_large_dist_time = time.time()
    # Check if 5 seconds have passed since the condition first became true
    elif (time.time() - self.last_large_dist_time > 5.0):
        fire_msg = Int32()
        fire_msg.data = 1
        self.pub.publish(fire_msg)
    # Check if 10 seconds have passed since the condition first became true
    elif (time.time() - self.last_large_dist_time > 10.0):
        ending_msg = Int32()
        ending_msg.data = 3
        self.error_pub.publish(ending_msg)
```

Delivery

Part 2,

마커인식 및 접근 지상 테스트

Part 2,

화물 배송 방법 (1)

Robot arm

드론 하단부에 ROS로 구동되는 Robot arm을 부착한 배송 방식

장점 : 정밀 배송 가능, 화물 형태 유지 가능

단점 : Robot arm의 길이가 제한적이므로 사정거리가 짧음

Part 2,

화물 배송 방법 (2 -1)

고무줄 밴드

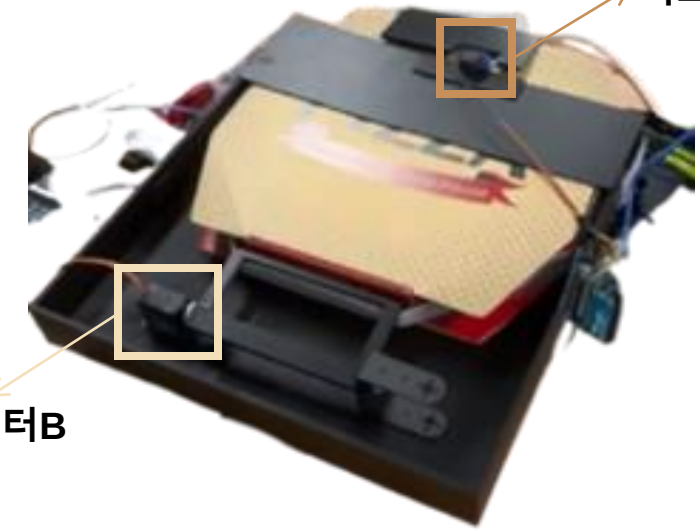
3D프린터 제작물인 발사장치에 고무줄 밴드를 부착,
탄성을 이용하여 화물을 사출하는 방식

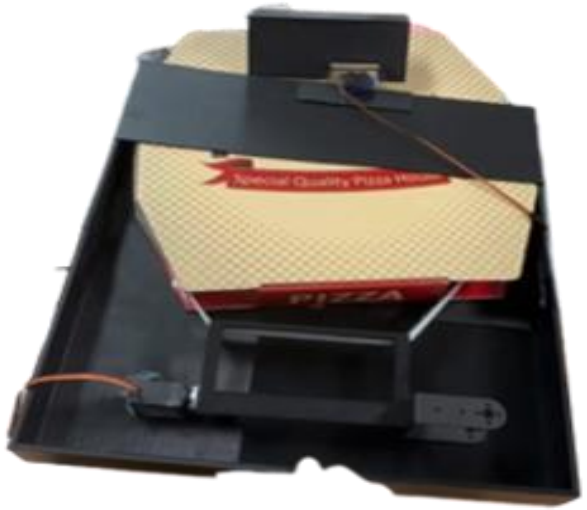
장점 : 긴 사정거리

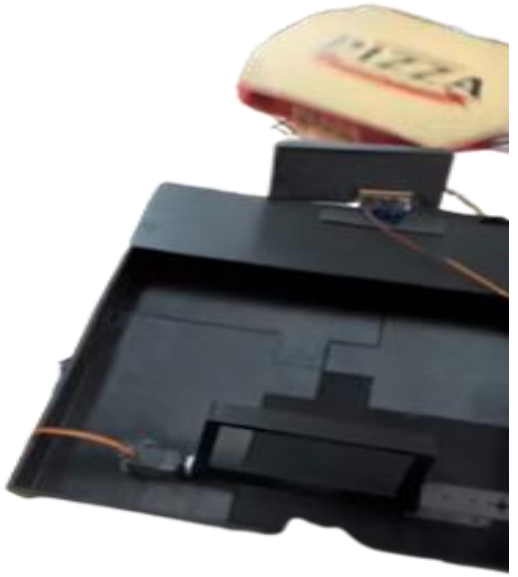
단점 : 구조물 무게 감량 필요

Part 2,

화물 배송 방법 (2-2)







1	OpenCV를 통해 이미지 인식 후 자동으로 배송 명령 전달
2	전면에 부착된 서보모터A를 이용하여 화물을 고정하는 가드를 개방
3	서보모터B로 고무줄 밴드를 고정하는 고리를 개방해 고무줄의 장력을 이용하여 발사

1 시스템 구성

2 연구 개발 내용

3 시뮬레이션 & 비행 테스트 결과

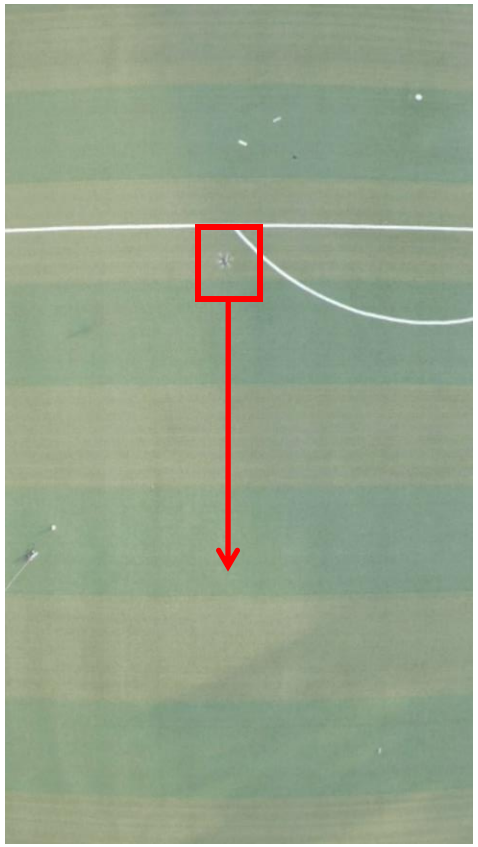
4 개선방안 및 향후 계획

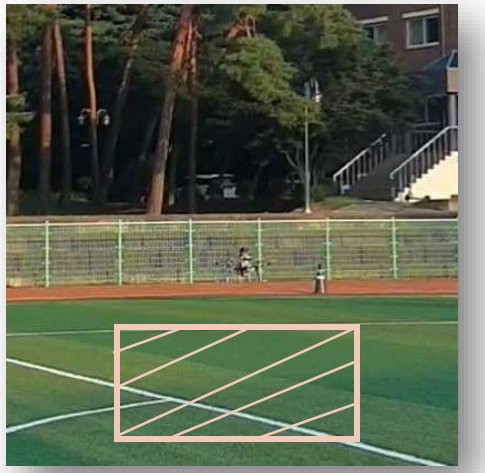
Part 3,

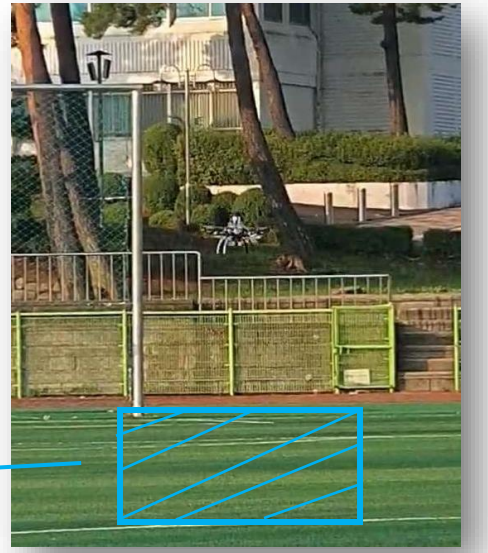
시뮬레이션 & 비행테스트 결과

Part 3,

GPS 좌표 이동 테스트



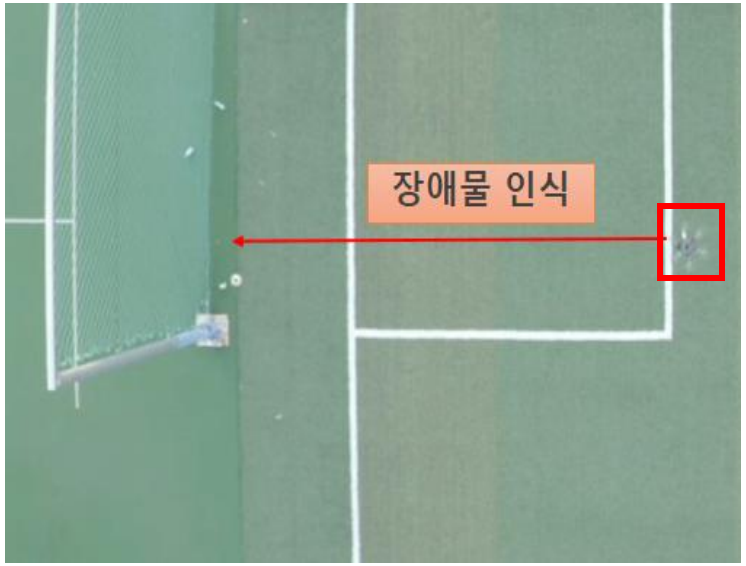


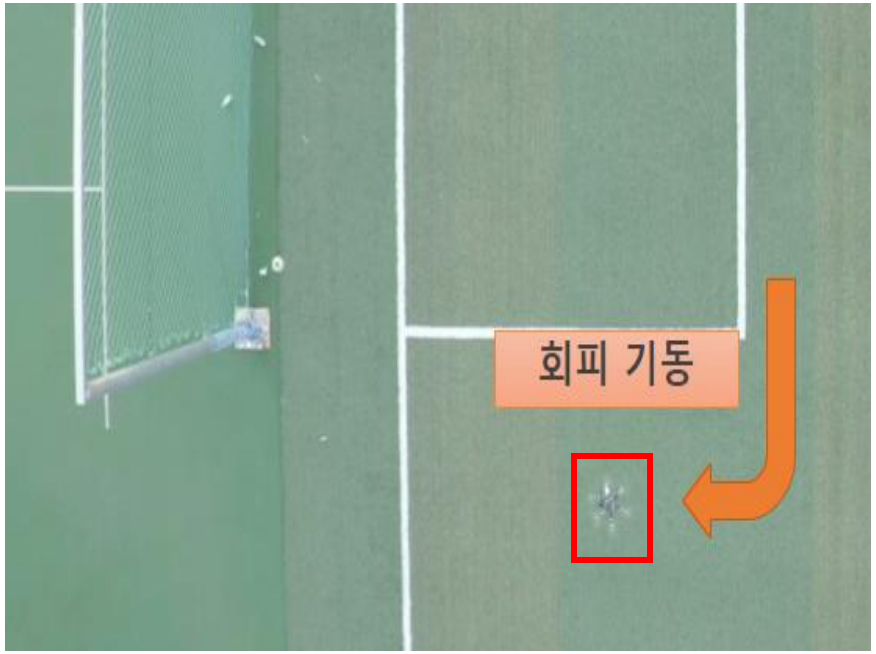


착륙지점 Gps 좌표
(37.737283, 128.875492)

Part 3,

장애물 회피 테스트





Part 3,

시뮬레이션 & 비행테스트 결과

1. 드론의 waypoint 이동 중 장애물 감지시 효과적인 회피 기동
2. 마커 탐색 및 로봇의 접근 가능
3. Offboard 비행 도중 안전한 조종권 인계 가능 등 안전성 검증
4. 시뮬레이션 환경과 동일하게 실제 드론에 적용 가능

Part 4,

개선방안 및 향후 계획

Part 4,

개선방안 및 향후계획

1. 드론의 임무 수행 전 과정을 한 번의 비행으로 모두 수행
2. 원형 마커에 대한 인식 부분을 십자형 마커로 변경 진행
3. 임무장비 무게 극복 위한 기체 형태 변경 진행(Quad --> Hexa)
4. 임무수행과 관련하여 안정성 지속 검토

Q&A

t