



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

공 학 석 사 학 위 논 문

RGB-D 카메라 기반 드론의 장애물 회피  
알고리즘 연구

A Study on Obstacle Avoidance  
Algorithm of Multi-Copter Drone  
using an RGB-D Camera

울산대학교 일반대학원  
기계공학과 항공우주공학전공  
권 용 길

# RGB-D 카메라 기반 드론의 장애물 회피 알고리즘 연구


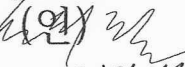

지도교수 신지철

이 논문을 공학석사학위 논문으로 제출함

2024년 2월

울산대학교 일반대학원  
기계공학과 항공우주공학전공  
권용길

권 용 길 의 공학석사학위 논문을 인준함

심사위원	하	철	근 
심사위원	신	지	철 (인) 
심사위원	이	병	룡 (인) 

울 산 대 학 교 대 학 원  
2024년 2월

# RGB-D 카메라 기반 드론의 장애물 회피 알고리즘 연구

울산대학교 일반대학원  
기계공학과 항공우주공학전공  
권 용 길

## 초 록

드론의 장애물 회피 알고리즘 기초 연구로써, PX4 오픈소스와 Depth Camera 를 이용하여 3DVFH\* 장애물 회피 및 경로 생성 알고리즘을 분석 및 검증하였다. Gazebo 시뮬레이션 환경을 구성하여 장애물 회피 알고리즘을 분석하였고, 실외 비행시험을 통해 장애물 회피 알고리즘을 검증하였다. Gazebo 시뮬레이션 에서 복잡하게 장애물이 있는 환경과 긴 벽이 있는 환경 등 다양한 비행 환경을 구성하여 회피 경로와 연관된 파라미터 조합을 통한 회피 성능 변화를 분석하였고, 실외 비행에서 시뮬레이션 환경과 유사한 환경을 조성하여 비행 테스트를 수행하였다. 대부분의 장애물에서 안정적인 회피 및 경로 생성이 되었지만, 빛 반사가 되는 재질이거나 얇은 나뭇가지에서는 실험에 사용된 RGB-D 카메라의 한계로 인해 장애물이 탐지되지 않았고, 회피하지 못하였다. 장애물 회피 알고리즘 성능 개선을 위해 시뮬레이션에서 RGB-D 카메라 3개를 이용한 실험을 진행하였고, 카메라 1개에 비해 빠른 회피 및 경로 생성 능력을 보여주었다.

## 목 차

국 문 요 약 .....	iv
목 차 .....	v
그 림 목 차 .....	vii
1. 서 론 .....	1
1.1 연구 배경 .....	1
1.2 연구 동향 .....	3
1.2.1 Geometrical methods .....	3
1.2.2 Optimized trajectory method .....	4
1.2.3 Potential field methods .....	5
1.2.4 Vector Field Histogram (VFH) .....	6
1.3 연구 목적 .....	7
1.4 장애물 탐지 및 회피 알고리즘 개요 .....	8
1.4.1 Octomap .....	8
1.4.2 3DVFH+ 알고리즘 .....	9
1.4.2.1 Octomap Exploring .....	9
1.4.2.2 2D Primary Polar Histogram .....	9
1.4.2.3 물리적 특성 .....	11
1.4.2.4 2D Binary Polar Histogram .....	12
1.4.2.5 경로 탐지 및 선택 .....	13
1.4.3 3DVFH* 알고리즘 .....	14
2. 본 론 .....	17
2.1 실험 구성 .....	17
2.1.1 시뮬레이션 환경구성 .....	17
2.1.2 실외 비행 환경구성 .....	20
2.1.3 실외 자율 비행 시스템 .....	23
2.2 실험 결과 .....	24
2.2.1 단일 카메라 시뮬레이션 비행 실험 결과 .....	24
2.2.1.1 복잡한 블록 사이 회피 비행 결과 .....	24
2.2.1.2 긴 블록 회피 비행 결과 .....	27
2.2.1.3 블록 2개 사이 회피 비행 결과 .....	30
2.2.2 단일 카메라 실외 비행 실험 결과 .....	31
2.2.2.1 사다리차 회피 비행 결과 .....	31
2.2.2.2 건물 및 나무 회피 비행 결과 .....	32
2.2.2.3 큰 건물 회피 비행 결과 .....	34
2.2.2.4 큰 나무 회피 비행 결과 .....	34

2.2.3	다중 카메라 시뮬레이션 비행 실험 결과	35
2.2.3.1	카메라 1개 실험 결과	35
2.2.3.2	카메라 3개 실험 결과	36
4.	결론	38
참고문헌		39
영문 요약		41

## 그림 목 차

1. Fig. 1 연도별 신규등록 현황 .....	1
2. Fig. 2 드론 충돌 사고 .....	1
3. Fig. 3 장애물 회피 드론 시스템 .....	2
4. Fig. 4 장애물 인식 센서 비교 .....	2
5. Fig. 5 Path planning 구분 .....	3
6. Fig. 6 장애물 거리에 따른 궤적 변화 .....	4
7. Fig. 7 전역 경로 및 로컬궤적 최적화 .....	5
8. Fig. 8 드론에 작용하는 힘 .....	5
9. Fig. 9 2D 데카르트 히스토그램 격자 생성 .....	6
10. Fig. 10 Active Window와 Polar Histograms .....	7
11. Fig. 11 Octomap Octree 데이터 구조 .....	8
12. Fig. 12 point cloud (left), volumetric (voxel) representation (right) .....	8
13. Fig. 13 2D Polar Histogram .....	9
14. Fig. 14 voxel 위치를 나타낸 그림 .....	10
15. Fig. 15 voxel 확대 그림 .....	10
16. Fig. 16 물리적 특성 단계에서의 회전 .....	11
17. Fig. 17 Moving window of the fifth stage .....	13
18. Fig. 18 장애물 회피 최적 경로 선택 .....	14
19. Fig. 19 3DVFH* algorithm 구조 .....	15
20. Fig. 20 A* 알고리즘 cost function .....	15
21. Fig. 21 visualization of cost terms .....	16
22. Fig. 22 시뮬레이션 테스트 기체 .....	17
23. Fig. 23 복잡하게 블록이 있는 비행 환경 .....	18
24. Fig. 24 긴 블록이 있는 비행 환경 .....	18
25. Fig. 25 높은 기둥 2개가 있는 비행 환경 .....	19
26. Fig. 26 dynamic_reconfigure node parameter .....	20
27. Fig. 27 실외 비행 테스트 기체 (사다리차 회피 임무) .....	20
28. Fig. 28 실외 비행 테스트 기체 (건물 및 나무 회피 임무) .....	21
29. Fig. 29 사다리차 사이를 통과하는 임무 환경 .....	21
30. Fig. 30 경로상 나무 및 건물이 있는 임무 환경 .....	22
31. Fig. 31 경로상 큰 건물이 있는 환경 .....	22
32. Fig. 32 경로상 큰 나무가 있는 임무 환경 .....	23
33. Fig. 33 실외 자율 비행 시스템 구성도 .....	24
34. Fig. 34 yaw_cost_parameter = 1 .....	25
35. Fig. 35 yaw_cost_parameter = 5 .....	25
36. Fig. 36 yaw_cost_parameter = 10 .....	25
37. Fig. 37 yaw_cost_parameter = 19 .....	25



38. Fig. 38 yaw_cost_parameter = 1	26
39. Fig. 39 yaw_cost_parameter = 5	26
40. Fig. 40 yaw_cost_parameter = 10	26
41. Fig. 41 yaw_cost_parameter = 19	26
42. Fig. 42 pitch_cost_parameter = 25	27
43. Fig. 43 pitch_cost_parameter = 5	27
44. Fig. 44 yaw_cost_parameter = 1	27
45. Fig. 45 yaw_cost_parameter = 5	27
46. Fig. 46 yaw_cost_parameter = 10	28
47. Fig. 47 yaw_cost_parameter = 19	28
48. Fig. 48 velocity_cost_parameter = 14000	28
49. Fig. 49 velocity_cost_parameter = 49000	28
50. Fig. 50 obstacle_cost_parameter = 1.5	29
51. Fig. 51 obstacle_cost_parameter = 5	29
52. Fig. 52 obstacle_cost_parameter = 1.5	29
53. Fig. 53 obstacle_cost_parameter = 5	29
54. Fig. 54 yaw_cost_parameter = 1	30
55. Fig. 55 yaw_cost_parameter = 5	30
56. Fig. 56 yaw_cost_parameter = 10	30
57. Fig. 57 yaw_cost_parameter = 19	30
58. Fig. 58 첫 번째 장애물 인식 및 회피 경로	31
59. Fig. 59 두 번째 장애물 인식 및 회피 경로	32
60. Fig. 60 나무 인식 및 회피 경로	33
61. Fig. 61 나무, 건물 인식 및 회피 경로	33
62. Fig. 62 건물 인식 실패	34
63. Fig. 63 작은 나뭇가지 인식 실패	35
64. Fig. 64 카메라 1개 복잡한 블록	36
65. Fig. 65 카메라 1개 긴 블록	36
66. Fig. 66 카메라 3개 복잡한 블록	36
67. Fig. 67 카메라 3개 긴 블록	36

# 1. 서론

## 1.1 연구 배경

드론 시장이 증가함에 따라 다양한 드론들이 시중에 판매되고 있으며, 드론 운용자 수 또한 나날이 증가하고 있다.

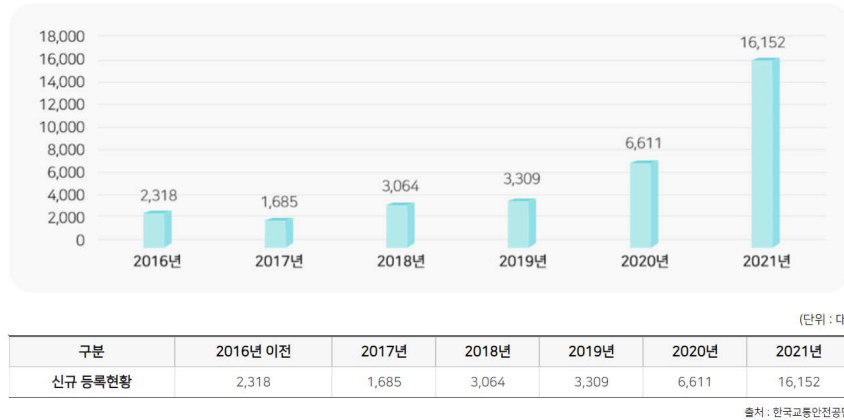


Fig. 1 연도별 신규등록 현황

그에 따른 드론 관련 사고들도 급증하고 있다. 드론 사고 감소를 위해서는 드론이 정적 장애물 또는 동적 장애물과의 충돌을 회피하는 시스템이 필요하다. 또한 현재 주목받고 있는 드론 택시, 드론 택배와 같은 자율 비행 드론이 도심 하늘을 비행하기 위해서는 동적 및 정적 장애물과의 충돌을 피하는 것이 중요하다.



Fig. 2 드론 충돌 사고

드론이 자체 센서를 이용하여 환경을 인식하고 스스로 경로를 만든 후 장애물을 회피해 목표지점까지 무사히 비행하기 위해서는 드론 위치 및 자세 제어 시스템, 장애물 회피 경로 계획 시스템, 장애물 인식 시스템이 반드시 구현되어야 한다.

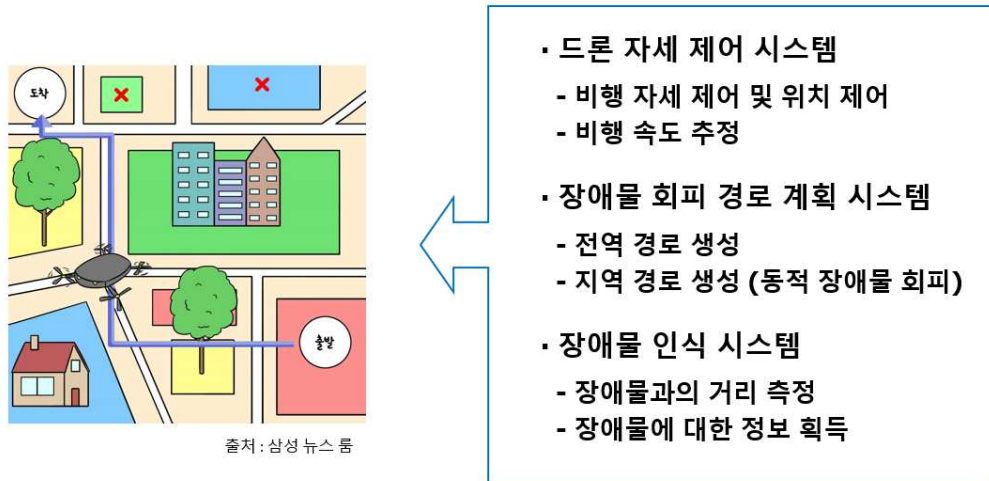


Fig. 3 장애물 회피 드론 시스템

시중에 판매되고 있는 대부분을 드론들도 장애물과의 충돌 방지를 위한 여러 종류의 센서를 탑재하고 있다.

		
<b>라이다 센서</b>	<b>이미지 센서 (RGB-D 카메라)</b>	<b>초음파 센서</b>
- 장점 : 고속, 높은 정확도	- 장점 : 높은 정확도, 색상 및 형상 정보	- 장점 : 저렴한 가격
- 단점 : 비싼 가격, 색상 및 형상 정보 x	- 단점 : 실외 역광에서 오류 발생	- 단점 : 낮은 정확도, 색상 및 형태 정보 x

Fig. 4 장애물 인식 센서 비교

충돌 방지를 위한 센서의 종류로는 대표적으로 라이다 센서, 이미지 센서, 초음파 센서가 있다. 라이다 센서의 경우 고속에서 사용 가능하고 실내외에서 정확도가 높다는 장점이 있지만, 색상 및 형상 정보를 제공하지 못한다는 단점이 있다. 이미지 센서의 경우 라이다와 비슷한 수준의 정확도를 보이며 색상 및 형상 정보도 얻을 수 있다는 장점이 있지만, 실외에서 역광에 따른 오류가 종종 발생한다. 초음파 센서의 경우 저렴한 가격에 가볍다는 장점이 있지만, 낮은 정확도와 색상 및 형상 정보를 제공하지 못한다는 단점이 있다. 추후 다른 임무와의 호환 가능성을 염두 하여 본 연구에

서는 이미지 센서와 적외선 센서가 합쳐진 Intel 사의 RGB-D 카메라인 리얼센스 D455 제품을 이용하여 장애물 회피 및 경로 생성 알고리즘 연구를 수행하였다.

## 1.2 연구 동향

다음 논문은 드론의 장애물 회피 및 경로 생성 알고리즘 연구에 관한 것이다.

드론이 장애물을 회피하는 방법은 크게 두 가지로 정의가 가능하다. 비행 시작 전 장애물에 대한 정보를 모두 알고 있는 경우에 장애물을 회피하는 방법으로, Trajectory Planning(TP)이라 하며, Global path planning이라고도 한다. 이와는 반대로 장애물에 대한 정보를 알지 못한 상태로 비행 중 장애물을 회피하는 방법으로, Reactive Obstacle Avoidance(ROA)라고 하며 Local path planning이라고도 한다.

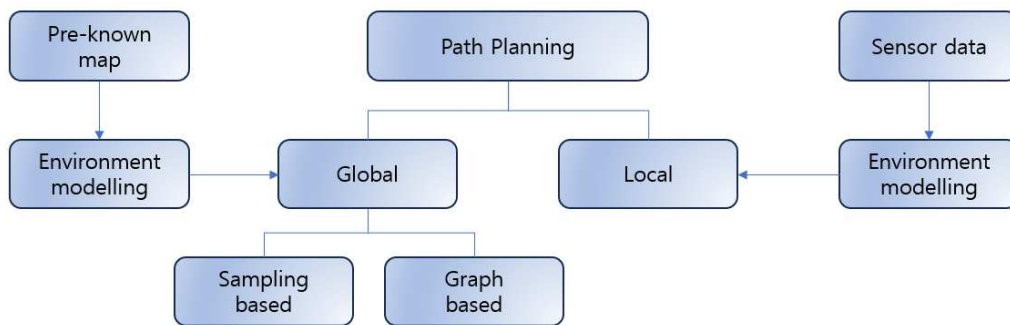


Fig. 5 Path planning 구분

자율 비행에 활용되는 충돌회피 알고리즘으로는 대표적으로 Geometrical methods, Potential field methods, Vector Field Histogram(VFH) 등으로 분류된다.

### 1.2.1 Geometrical methods

기하학적 접근법 바탕인 회피 방법은 기하학적으로 정의 된다. 기하학적 접근법의 이용하기 위해서는 드론과 장애물 둘 다의 위치, 헤딩 및 속도와 같은 파라미터를 알고 있어야 한다.

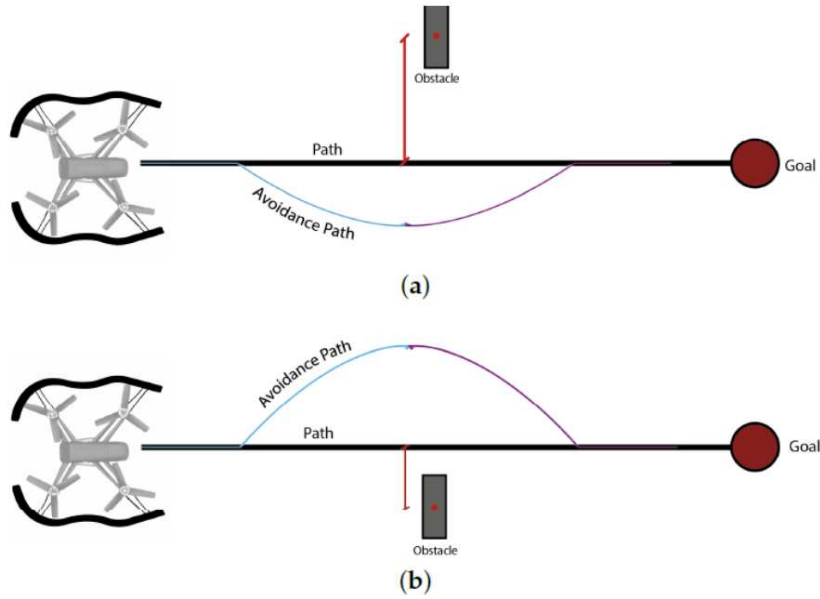


Fig. 6 장애물 거리에 따른 궤적 변화 [8]

Fig. 6의 경우 기하학적 접근법 장애물 회피에 대한 예시를 보여주고 있다. 설정된 장애물과의 최소거리에 들어오게 되면 드론은 장애물과의 충돌을 방지하기 위해 최단 거리 벡터를 계산할 것이다. 장애물 가까울수록 궤적을 변화는 더 크게 나타난다. [2] 이러한 방법은 계산이 복잡하지 않고 간단하다는 장점을 가지고 있다. 하지만 시스템이 장애물의 파라미터를 알고 있어야 하며, 정확한 계산이 필수이다. 또한 복잡한 장애물에서 정확성이 매우 떨어진다.

### 1.2.2 Optimized trajectory method

궤적 최적화 방법은 기하학적 접근법을 기초로 하는 방법이다. 장애물을 회피하기 위해 모든 장애물에 대한 최적의 경로를 계산하는 방법이다.

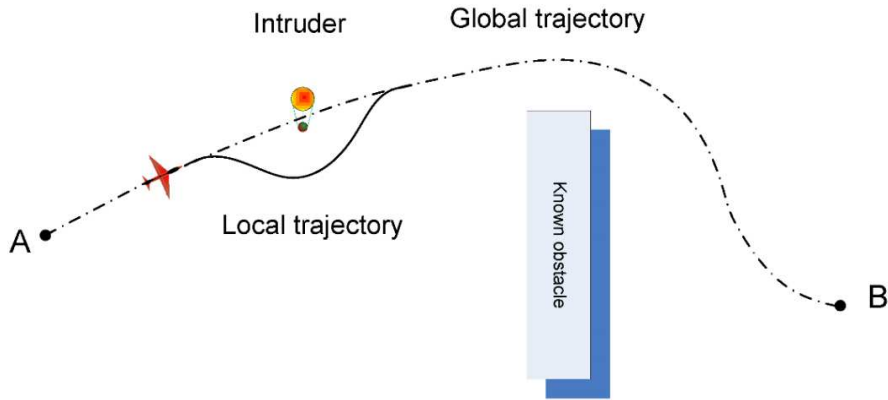


Fig. 7 전역 경로 및 로컬궤적 최적화 [8]

Fig. 7의 경우 궤적 최적화 방법에 대한 예시를 보여주고 있다. 비행 시작 전 알려진 장애물에 대한 전역 경로를 형성한다. 드론은 목적지를 향해 전역 경로를 따라 이동합니다. 경로에 발생하는 로컬 장애물에 대해서는 다른 알고리즘과 접근법들을 이용한다. [3]

### 1.2.3 Potential field methods

포텐셜 필드 방법은 1985년 Khatib에 의해 처음 제안되었다. [4] 목표 포인트를 인력으로 정의하고, 장애물을 반발력으로 정의한다. 이를 이용해 장애물을 회피하는 경로를 생성한다.

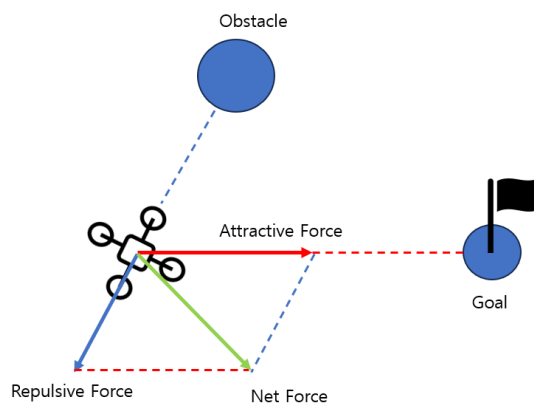


Fig. 8 드론에 작용하는 힘

비교적 빠른 시간에 경로 생성이 가능하고, 복잡한 장애물에서 경로 생성이 가능하다. [5] 포텐셜 필드 방법은 로컬영역에서 장애물의 회피하기 유용하다. 하지만 회피 비행 중 일정 구간을 반복하는 지역 극소점(local minima) 문제에 빠질 수 있다. [6]

### 1.2.4 Vector Field Histogram (VFH)

벡터 필드 히스토그램 방법은 반응형 실시간 장애물 회피 방법으로, Borenstein과 Koren에 의해 처음 제안되었다. [7] 연구는 로봇이 목표 포인트로 이동하는 동안 알려지지 않은 장애물을 감지하고 회피하는 방법의 설명하고 있다. VFH 방법은 2D 데카르트 히스토그램 격자를 이용한다. 시스템에 통합된 센서들이 수신하는 데이터는 2 단계의 데이터 축소 과정을 거칩니다. 먼저 1D 폴라 히스토그램에서 로봇 위치 주변의 2D 히스토그램 격자의 일정한 크기 부분집합을 축소한 다음 폴라 히스토그램에서 가장 적합한 섹터를 선택하여 원하는 제어 명령을 계산합니다. 방법은 크게 세 단계로 나눌 수 있습니다. [8]

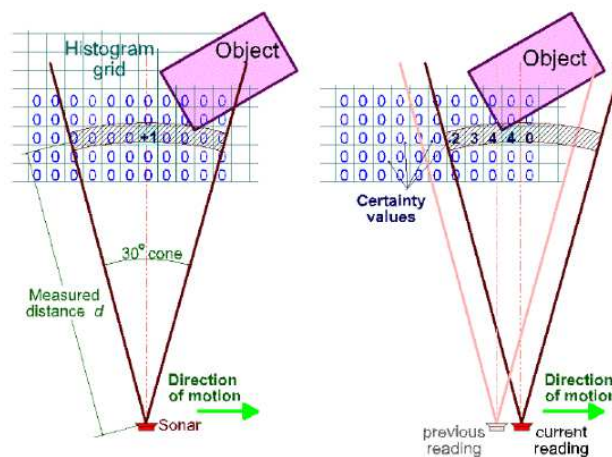


Fig. 9 2D 데카르트 히스토그램 격자 생성 [8]

첫 번째로 2D 데카르트 히스토그램의 생성은 다음과 같다. 각 레인지 센서 측정으로 찍은 모든 좌표를 고려하여 격자지도에 넣는다. Fig. 9에서 나타난 것처럼 각 범위에 중심축이 놓여 있고 거리 d에 대응하는 셀이 증가합니다. 이는 해당 셀의 certainty value의 증가를 의미합니다. certainty value의 경우 차량이 움직이는 동안 계속 업데이트된다. [9]

다음으로 2D 격자지도를 1D 구조로 변환하는 것이다. 전체 격자지도를 처리하는 것보다 장애물에 대한 위협 정보를 개선하기 위해, Active Window 개념을 도입한다. 차량 위치를 중심으로 일정한 차원으로 영역을 제한한다. 이 영역은 로봇의 이동과 함께 변하고, 그 주변의 로컬영역으로 나타낸다.

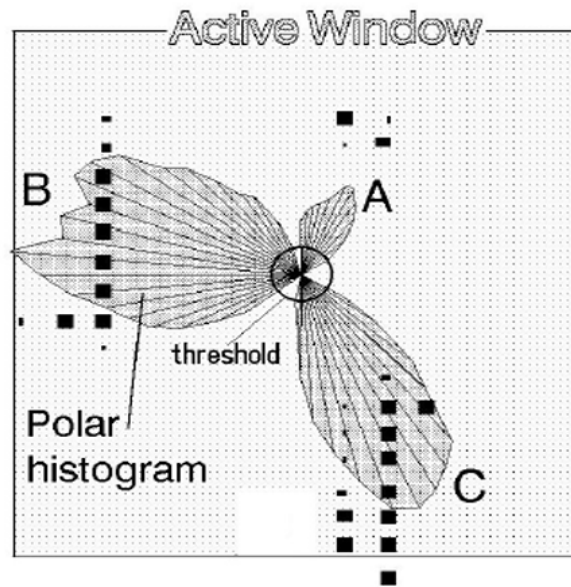


Fig. 10 Active Window와 Polar Histograms [8]

새로운 격자는 폴라 히스토그램이라 불리는 1차원 구조로 지도가 생성된다. Fig. 10은 로봇 주변에 3개의 장애물이 있는 상황에서 폴라 히스토그램을 나타낸 것이다.

마지막으로, 로봇의 이동 방향이 결정된다. 방향은 장애물의 극성 밀도에 따라 결정된다. VFH 방법은 센서의 잘못된 정보로 발생할 수 있는 문제를 최소화할 수 있고, 포텐셜 필드 방법에서 발생한 지역 극소 점(local minima) 문제를 해결하였다. 장애물 경계 값을 이용해 회피 경로를 생성하는 방법으로 실시간 동작 및 구현 가능하다는 장점이 있다.

### 1.3 연구 목적

본 연구는 드론이 비가시권 자율 비행 시 드론에 부착된 이미지 센서에 장애물이 탐지되면 자율적으로 회피 경로를 생성하여 안전하게 비행 임무를 수행하는 것을 목적으로 하며, RGB-D 카메라 기반 드론의 장애물 회피 비행 시스템 구축 및 성능 검증하고자 한다.

주요 연구 내용은 다음과 같다. 장애물 회피 및 경로 생성에 있어서는 Local Path Planning 방법을 이용하고, 충돌회피 알고리즘은 3D Vector Field Histogram (3DVFH\*)을 사용한다. Gazebo 시뮬레이션을 통해 다양한 환경에서 장애물 회피 알고리즘 분석하고, 실외 비행을 통한 장애물 회피 성능을 검증하는 연구를 수행한다.

테스트 결과를 기반으로 실제 드론 운용환경에서의 회피 알고리즘 이용가능성에 대해 분석하고, 알고리즘 문제점에 대한 하드웨어적 개선 방안을 제시하고 이를 시뮬레이션 환경에서의 비행 결과를 통해 증명한다. 또한 실험에 사용된 RGB-D 카메라의 장애물 탐지 성능을 분석한다.



## 1.4 장애물 탐지 및 회피 알고리즘 개요

### 1.4.1 Octomap

Octomap 데이터 구조는 3차원 환경을 표현하는 효율적인 방법이다. [11] Octomap은 Octree Structure를 기반으로 하는 3D grid이다. 모든 voxel (물체)는 8개의 자식 voxel을 포함한다. 8개의 자식 voxel은 가득 차게 되면 하나의 부모 voxel로 감소하고, 8개의 자식 voxel이 모두 비었다면, 이 voxel은 비어있는 하나의 부모 voxel로 감소한다. 이 계층적 구조에 있는 레이어의 수는 Octree의 정밀도와 크기를 결정한다. 장애물 회피 관점에서 보게 되면 차 있는 voxel은 장애물 유무를 나타낸다. 하지만 드론에 장착 되어있는 센서들은 종종 노이즈를 발생시키기 때문에 각 voxel에는 장애물 유무를 확률적으로 표현하고 있다.

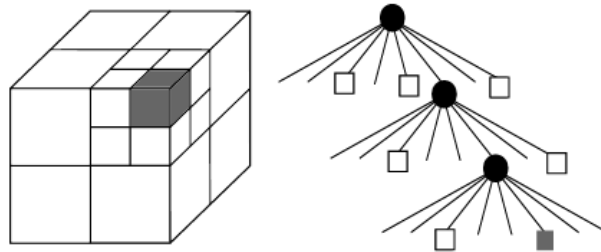


Fig. 11 Octomap Octree 데이터 구조 [12]

센서로부터 들어온 point cloud 정보를 통해 빈 공간에 대한 점유 확률을 계산하고, 점유 확률이 일정 값 이상 커지면 해당 영역에 장애물이 있다고 판단하여 3차원 공간 지도를 구성한다.

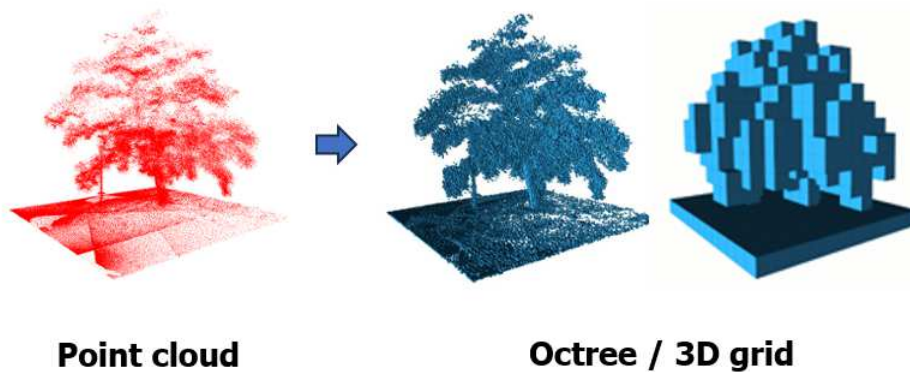


Fig. 12 point cloud (left), volumetric (voxel) representation (right) [13]

### 1.4.2 3DVFH+ 알고리즘

Ulrich et al. [10]은 3DVFH+ 알고리즘의 기본이 되는 VFH+ 알고리즘을 제시하였다. VFH+ 알고리즘은 2D 1차 폴라 히스토그램, 2D 이진 폴라 히스토그램, 마스크 폴라 히스토그램을 계산하는 과정을 거친다. 이 알고리즘은 2차원에서의 드론 경로를 계산할 수 있으며, 실시간으로 매끄러운 궤적을 생성하고 드론의 크기 및 회전속도와 같은 물리적 특성을 고려한다. 3DVFH+ 알고리즘은 3차원 환경에서의 드론 운용을 위해 2차원의 VFH+ 알고리즘 향상 버전이다. 3DVFH+ 알고리즘에서 드론이 경로를 계산하기 위해서는 아래 5단계의 계산이 필요하다. [12]

#### 1.4.2.1 Octomap Exploring

드론이 넓은 환경에서 움직일 때 모든 환경에 대해 장애물 여부를 분석하는 것은 컴퓨터 성능으로 인해 무리가 있다. 따라서 드론에 노출된 환경에서 특정 부분 (bounding box)만의 Octomap을 이용하여 분석한다. bounding box는  $w_w \times w_h \times w_d$  (폭, 높이 그리고 깊이) 크기를 가지며, Vehicle Center Point (VCP)를 중심으로 한다. VCP는 드론의 중심점이며, 드론 전체가 이 점으로 표시된다. 어떠한 voxel이 bounding box 안에 있다면, 그 voxel은 3D 좌표  $i,j,k$ 에 위치한 active cell이라고 부른다. 탐색단계에서는 bounding box 내에 있는 특정 voxel을 찾아서 정보 분석하면 된다. 특정 voxel을 정보 분석이 필요 없는지를 파악하기 위해서 위치를 알아야 하지만 위치정보를 Octomap data structure에 넣기에는 메모리에 무리가 가기 때문에 Octomap tree를 움직이며 위치를 계산하게 된다. 또한 거리가 먼 bounding box는 탐색하지 않는다. 첫 번째 단계에서 bounding box가 발견되면 두 번째 단계 bounding box의 정보를 이용하여 2D primary polar histogram을 만든다.

#### 1.4.2.2 2D Primary Polar Histogram

첫 번째 단계에서 얻은 voxel 정보를 2D primary polar histogram에 추가한다. 히스토그램은 활성 voxel의 위치를 나타내는 극 히스토그램이며, 2개의 각도에 의해 결정된다. 각도는 voxel과 VCP 위치에 의해 결정된다. voxel을 기준으로 계산되는 가중치는 2개의 각도를 좌표로 하여 2D primary histogram에 삽입된다.

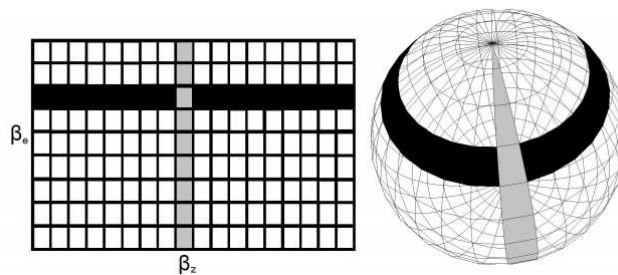


Fig. 13 2D Polar Histogram [12]

먼저 bounding box 내에 bounding sphere를 만들어 활성 셀의 목록을 줄인다. 이것은 VCP와 각 voxel 사이 euclidean distance  $d_{i,j,k}$  계산을 통해 해결할 수 있으며, euclidean distance가 bounding sphere의 반지름보다 클 때 (즉,  $d_{i,j,k} > w_s/2$ ) 해당 voxel은 무시된다.

다음으로 2D primary polar histogram  $H^p$ 내의 좌표를 결정하기 위해 2개의 각도를 계산한다. 각도는 azimuth angle  $\beta_z$ (2D primary polar histogram의 x축)과 elevation angle  $\beta_e$  (2D primary polar histogram의 y축) 이다.

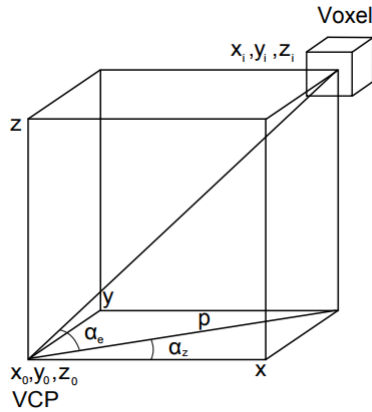


Fig. 14 voxel 위치를 나타낸 그림 [12]

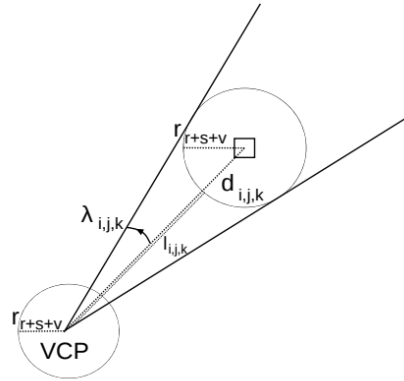


Fig. 15 voxel 확대 그림 [12]

azimuth angle  $\beta_z$ 과 elevation angle  $\beta_e$ 는 각각 식 1.1과 식 1.2로 계산된다.

$$\beta_z = \text{floor}\left(\frac{1}{\alpha} \arctan \frac{x_i - x_0}{y_i - y_0}\right) \quad (1.1)$$

$$\beta_e = \text{floor}\left(\frac{1}{\alpha} \arctan \frac{z_i - z_0}{p}\right) \quad p = \sqrt{(x_i - x_0)^2 + (y_i - y_0)^2} \quad (1.2)$$

정리하면 다음과 같다.

$$\beta_e = \text{floor}\left(\frac{1}{\alpha} \arctan \frac{z_i - z_0}{\sqrt{(x_i - x_0)^2 + (y_i - y_0)^2}}\right) \quad (1.3)$$

이전 계산은 VCP와 voxel의 중심을 이용하여 두 각도를 계산하였다. 드론의 크기는 모든 활성 voxel을 크기를 확대하여 구현한다. 이것은 Fig. 15에 나타나 있다. 확대 계수는 드론의 반경  $r_r$ , 안전 반경  $r_s$  그리고 voxel의 크기  $r_v$ 를 더하여 계산된다. 계산식은 식 1.4에 나타나 있다.

$$\lambda_{i,j,k} = \text{floor}\left(\frac{1}{a} \arcsin \frac{r_{r+s+v}}{d_{i,j,k}}\right) \quad (1.4)$$

Voxel과 VCP 사이의 최소거리는 다음과 같이 계산할 수 있다.

$$l_{i,j,k} = d_{i,j,k} - r_{r+s+v} \quad (1.5)$$

히스토그램에 들어가는 정보는 voxel을 통해 계산한 확률정보이다. 이때 확률은  $l_{i,j,k}$  (euclidean distance)와  $o_{i,j,k}$  (occupancy certainty)로 계산된다.

$$H_{z,e}^p = \sum_{i,j,k \in C_a} \begin{cases} (o_{i,j,k})^2 (a - bl_{i,j,k}) & \text{if } e \in \left[ \beta_e - \frac{\lambda}{a}, \beta_e + \frac{\lambda}{a} \right] \\ & \text{and } z \in \left[ \beta_z - \frac{\lambda}{a}, \beta_z + \frac{\lambda}{a} \right] \\ 0 & \text{otherwise} \end{cases} \quad (1.6)$$

상수 a, b는 다음과 같이 구할 수 있으며, 상대적인 값이다.

$$a - b\left(\frac{w_s - 1}{2}\right)^2 = 1 \quad (1.7)$$

#### 1.4.2.3 물리적 특성

두 번째 단계 후에는 드론의 물리적 특징들을 이용해 추가적인 정보를 2D primary polar histogram에 추가하게 된다. 여기에는 드론 움직임의 한계점들을 고려해야 한다. 아래 그림과 같이 voxel이 위치할 때 드론의 회전궤적을 계산하고자 한다.

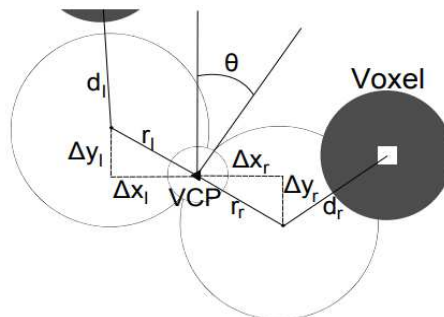


Fig. 16 물리적 특성 단계에서의 회전 [12]

VFH+ 알고리즘에 의해  $d_r$ 과  $d_l$ 은 다음과 같이 정의 된다. [11]

$$\begin{aligned} d_r &= \sqrt{(\Delta x_r - \Delta x(i))^2 + (\Delta y_r - \Delta y(j))^2} \\ d_l &= \sqrt{(\Delta x_l - \Delta x(i))^2 + (\Delta y_l - \Delta y(j))^2} \end{aligned} \quad (1.8)$$

$$d_r < (r_r + r_{r+s+v}) \quad d_l < (r_l + r_{r+s+v}) \quad (1.9)$$

또한 드론의 클라이밍 동작을 고려할 필요가 있다. 클라이밍 운동상수  $f$ 와 선회거리  $t$ 는 다음과 같이 정의된다.

$$t = \frac{2\pi r(2\alpha_z)}{360} \quad (1.10)$$

$$f = \frac{z_i - z_0}{t} \quad (1.11)$$

모든  $\alpha_z$ 각에 대한 도달 불가능 고도  $z_{az}$ 에서의 선회반경  $t_{az}$ 는 다음과 같이 정의된다.

$$t_{az} = \frac{2\pi r(2\beta_z\alpha)}{360} \quad (1.12)$$

$$z_{az} = f t_{az} \quad (1.13)$$

마지막으로 도달할 수 없는 elevation angle  $\beta_e$ 는 고도 차이와 euclidean distance  $l_{\beta_z}$ 에 의하여 계산가능 하며 다음과 같이 정의된다.

$$l_{az} = \sqrt{r^4 - 2r^2 \cos(270 - 2\alpha_z)} \quad (1.14)$$

$$\beta_e = \frac{1}{\alpha} \arctan\left(\frac{z_{az}}{l_{az}}\right) \quad (1.15)$$

이러한 계산은 알고리즘이 주어진 전진 속도 및 최대 상승 속도로 도달할 수 없는 셀을 계산하도록 확장할 수 있다. [12]

#### 1.4.2.4 2D Binary Polar Histogram

이전 과정을 통해 생성한 2D primary polar histogram을 2D binary polar histogram으로 바꾸는 과정을 통해 정보를 더 감소시킨다. 2가지 threshold  $\tau_{low}$ ,  $\tau_{high}$ 를 기준으로  $\tau_{high}$ 보다 크면 1,  $\tau_{low}$ 보다 작으면 0, 이 두 값 사이면 옆 voxel을 이

용한다.

$$H_{\beta_z, \beta_e}^b = \begin{cases} 1 & \text{if } H_{\beta_z, \beta_e}^b > \tau_{\beta, high} \\ 0 & \text{if } H_{\beta_z, \beta_e}^b < \tau_{\beta, low} \\ H_{\beta_{z-1}, \beta_p}^b & \text{otherwise} \end{cases} \quad (1.16)$$

이 방법을 통해 더 명확하게 장애물과 노이즈를 구분할 수 있다.

#### 1.4.2.5 경로 탐지 및 선택

마지막 단계에서는 2D binary polar histogram에서 이동 가능한 경로를 검색하고 경로 가중치가 가장 낮은 경로를 선택한다. 어떤 경로로 이동 가능한지를 알아보기 위해 2D binary polar histogram에서 window를 이동시켜 window 안에 모든 셀이 0인 지점들을 찾아 경로를 만든다.

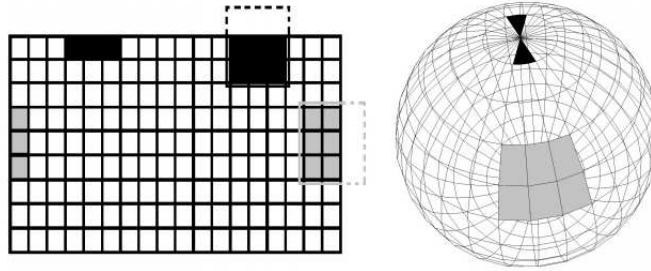


Fig. 17 Moving window of the fifth stage [12]

window가 히스토그램을 위에서 아래로 가로지를 때 (Fig. 17에서 검은색 부분) 같은 elevation angle에서 azimuth angle을 180도 돌리면 이어지는 셀들을 찾을 수가 있다. window가 왼쪽 또는 오른쪽 경계를 가로지르는 경우 window는 히스토그램의 반대쪽 셀들을 확인할 것이다. 이렇게 가능한 경로들을 어떠한 경로가 가장 효율적인지를 찾아야 한다. 이를 위해 경로에 대한 weight 값을 이용한다. weight 값이 작을수록 적은 움직임으로 목표에 도달할 수 있다. 경로 weight는 아래 식으로 계산한다.

$$k_i = \mu_1 \Delta(v, k_i) + \mu_2 \Delta(v, \frac{\theta}{\alpha}) + \mu_3 \Delta(v, k_{i-1}) \quad (1.17)$$

위의 식을 보면 총 3개의 weight가 있다. 첫 번째 weight는 특정 경로를 가기 위해 만들어져야 하는 target angle  $k_i$ 와 드론이 가는 방향  $v$ 의 차이를 기준으로 만들어진 것이다. 두 번째 weight는 rotation of robot  $\theta$ 와 드론이 가는 방향  $v$ 의 차이이다. 마지막 weight는 기존에 선택된 방향과 현재 방향의 차이이다. 모든 후보 방향의 경로 가중치가 계산되면 알고리즘은 가중치가 가장 낮은 방향을 선택한다.

### 1.4.3 3DVFH\* 알고리즘

VFH\* 알고리즘은 Ulrich와 Borenstein에 의해 처음 제안되었다. [14] VFH\* 알고리즘은 로컬 장애물 회피 알고리즘에 문제가 되는 상황을 해결하기 위해 개발되었다. 예를 들어 Fig. 18 그림에서 VFH+ 알고리즘의 경우 경로를 선택할 때 활성영역만 고려하기 때문에 경로 A, B 둘 다를 선택하게 된다. 경로 A는 막다른 길로 이어지는 반면, 경로 B를 따라간다면 점 p에서 경로 C로 바뀐다. 만약 사전 정보가 없다면 드론은 50%의 확률로 경로 B를 선택한다. 이러한 문제는 VFH+ 알고리즘에 A\* 알고리즘을 결합하여 여러 이동단계를 사전에 예측하고 평가하는 방식으로 해결 가능하다. 따라서 VFH\* 알고리즘의 경우 경로 B만의 선택하게 된다.

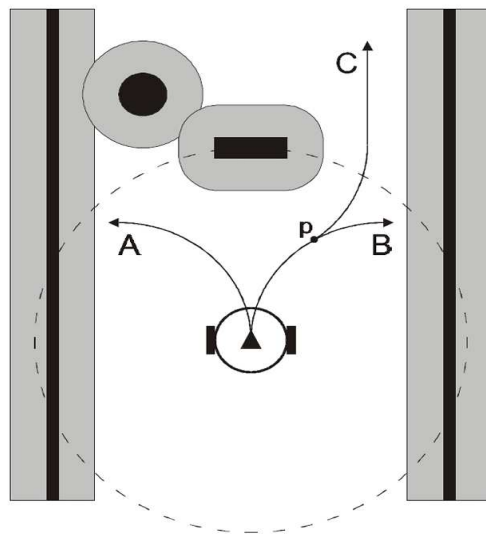


Fig. 18 장애물 회피 최적 경로 선택 [17]

3DVFH\* 장애물 회피 알고리즘은 2D 기반의 VFH\* 알고리즘을 3D 환경으로 확장한 것이다. [15] 3DVFH\* 알고리즘 구조는 Fig. 19에 나타나 있습니다.

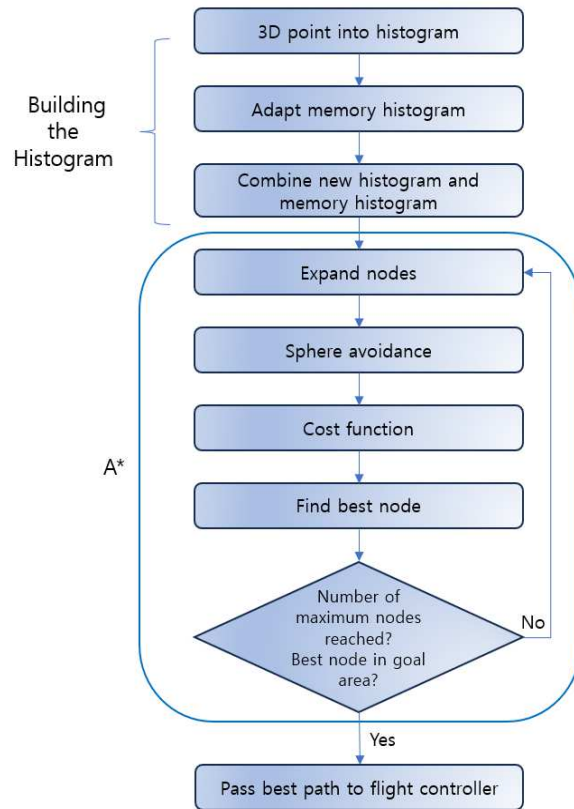


Fig. 19 3DVFH\* algorithm 구조

3DVFH\*의 핵심은 cost function이다. 이 cost function은 이동 가능한 경로들을 평가하여 가장 최적의 경로를 찾도록 해준다. cost function은 4가지 cost parameter인 yaw\_cost, pitch\_cost, velocity\_cost 그리고 obstacle\_cost와 heuristic cost로 구성되어 있다.

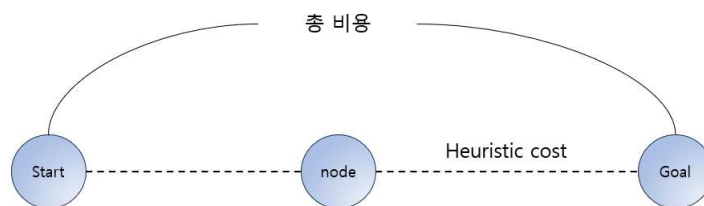


Fig. 20 A\* 알고리즘 cost function

heuristic cost는 고려된 노드에서 목표까지의 거리를 의미한다. 모든 cost parts에는 가중 계수 k가 있습니다. 각 노드는 pitch angle과 yaw angle을 포함하고 있으며 pitch angle은 elevation angle, yaw angle은 azimuth angle이다. [16]



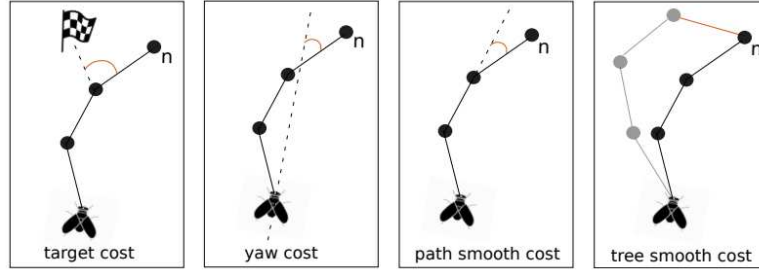


Fig. 21 visualization of cost terms [15]

yaw\_cost 값은 현재 노드와 목표 노드 사이의 yaw 각도 차이에 비례해 결정된다.

$$cost_{yaw} = k_{yaw} \times (yaw_{node} - yaw_{goal})^2 \quad (1.18)$$

pitch\_cost 값은 현재 노드와 목표 노드 사이의 pitch 각도 차이에 비례해 결정된다.

$$cost_{pitch} = k_{pitch} \times (pitch_{node} - pitch_{goal})^2 \quad (1.19)$$

velocity\_cost 값은 다음 노드의 normalized position과 현재 노드에서의 드론 속도 간의 내적이 고려된다. 위치 벡터와 속도 벡터 사이의 각도가 클수록 velocity cost 값이 커짐을 의미한다.

$$cost_{velocity} = k_{vel} \times (|v| - \frac{p_{node}}{|p_{node}|} \cdot v) \quad (1.20)$$

obstacle\_cost는 식 1.21과 같이 계산된다.  $d_{histogram}$ 은 노드와 다음 장애물 간의 거리를 의미하며, 가중 계수  $k_{obst}$ 는 distance value로 inflection point 역할을 한다.

$$cost_{obst} = 5000 \times (1 + \frac{d}{\sqrt{1 + d \times d}}) \quad (1.21)$$

$$d = k_{obst} - d_{histogram} \quad (1.22)$$

노드에 대한 비용 계산식은 식 1.23과 같이 간단히 표현된다.

$$cost_{total} = cost_{yaw} + cost_{pitch} + cost_{velocity} + cost_{obst} \quad (1.23)$$

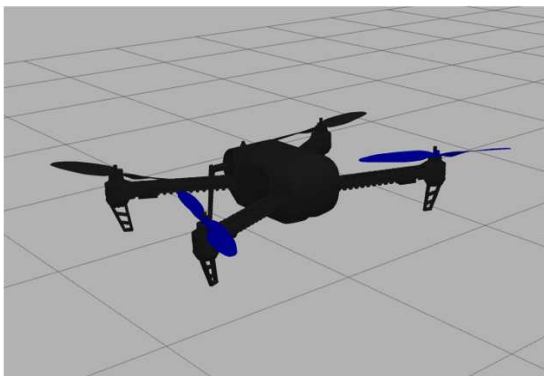
모든 노드에 대해 cost function을 적용한 후에 가장 낮은 값을 가지는 노드에 heuristic cost 항을 추가 한다.

## 2. 본 론

### 2.1 실험 구성

#### 2.1.1 시뮬레이션 환경구성

장애물 회피 시뮬레이션은 로봇 항공기 경연대회 임무인 사다리차 사이 통과를 가정한 높은 기둥 2개가 있는 환경, 끝이 보이지 않을 만큼 큰 건물 회피 비행을 가정한 긴 블록이 있는 환경 그리고 도심 속 높은 빌딩 사이 회피 비행을 가정한 복잡한 블록이 있는 환경으로 구성하여 실험을 진행하였다. 시뮬레이션에 사용될 기체 제원과 형상은 Fig. 22에 정리하였다.



항목	내용
최대이륙중량	3kg
모터 축간 거리	약 550mm
Battery	3s 3500mah
Endurance	약 15분
Camera	Realsense D435

Fig. 22 시뮬레이션 테스트 기체

시뮬레이션 환경에서 장애물 위치 및 크기는 다음과 같다.

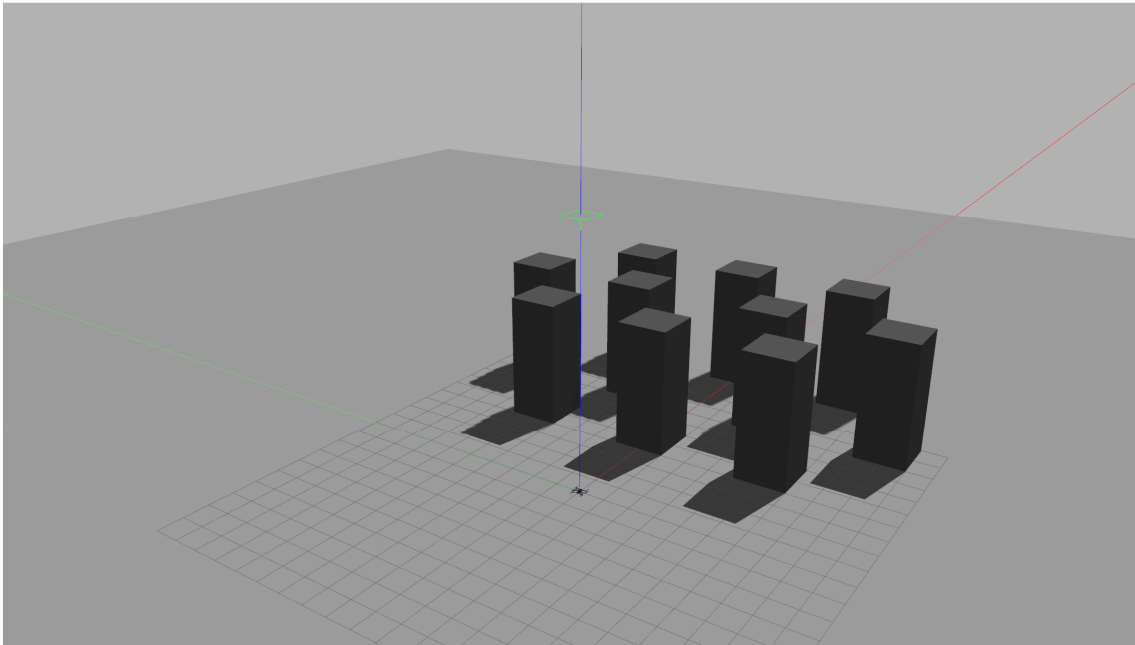


Fig. 23 복잡하게 블록이 있는 비행 환경

복잡하게 블록이 있는 비행 환경에서의 장애물 크기는 모두 동일하며  $2\text{m} \times 2\text{m} \times 5\text{m}$  ( $L \times W \times H$ ) 크기를 가지고 있다. 장애물 사이 간격은 앞뒤 2m, 좌우 3m 간격으로 구성되어 있다.

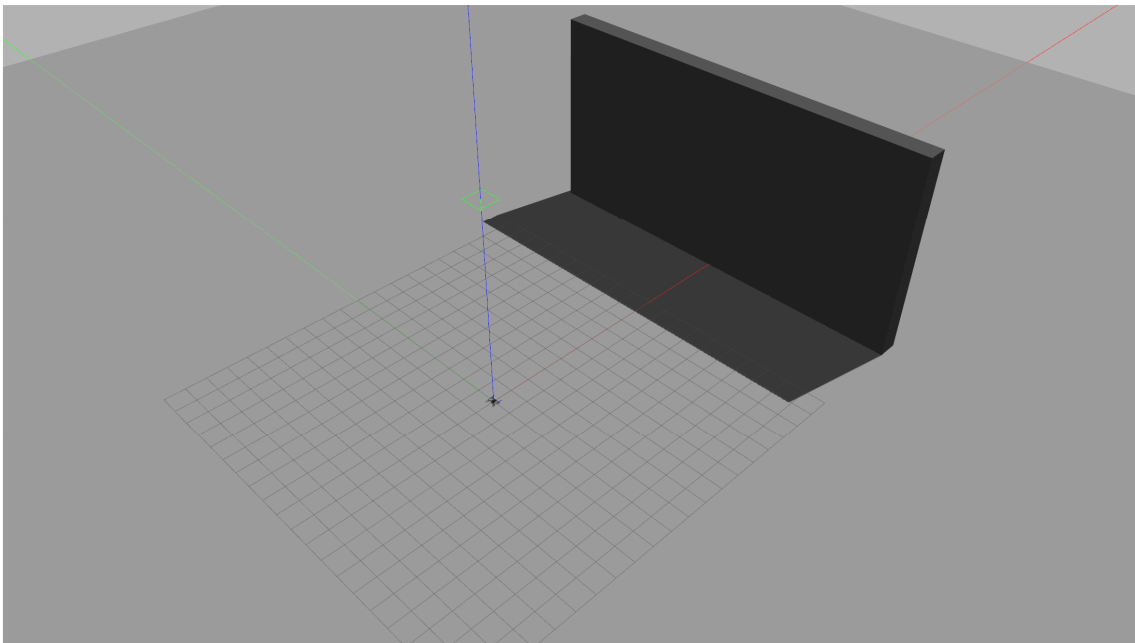


Fig. 24 긴 블록이 있는 비행 환경

긴 블록이 있는 비행 환경에서의 장애물 크기는  $20\text{m} \times 1\text{m} \times 10\text{m}$  ( $L \times W \times H$ ) 크기를 가

지고 있다.

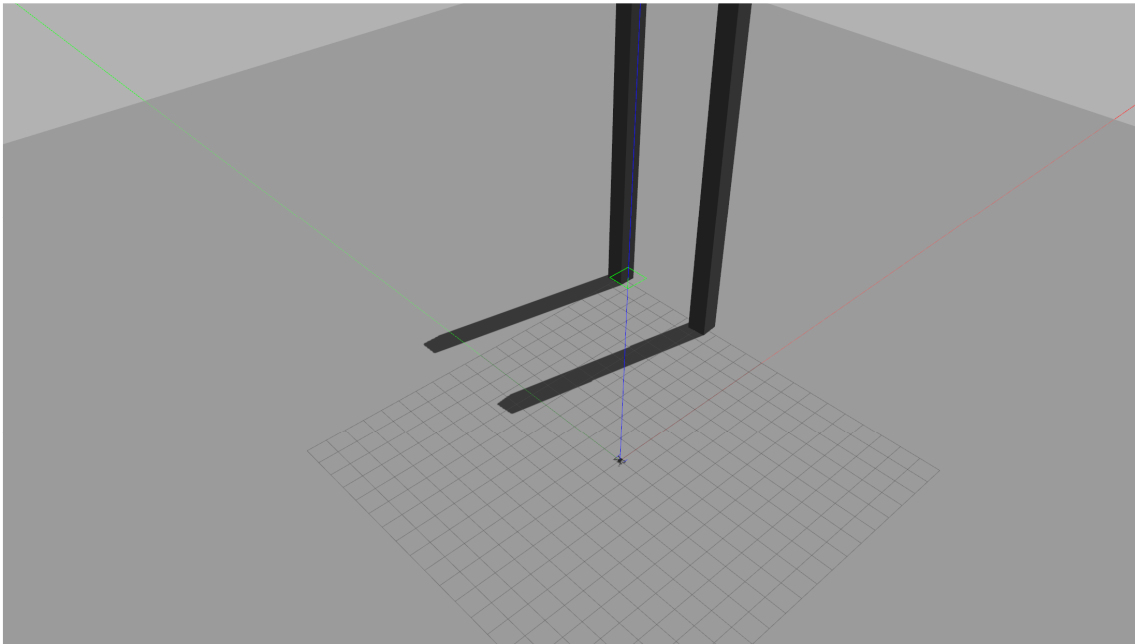


Fig. 25 높은 기둥 2개가 있는 비행 환경

높은 기둥 2개가 있는 비행 환경에서의 장애물 크기와 간격은 로봇 항공기 경연대회 임무에서의 사다리차 배치와 유사하게 구성하였다. 장애물 크기는  $1\text{m} \times 1\text{m} \times 20\text{m}$  ( $L \times W \times H$ )이고, 장애물 사이 간격은 6m이다.

시뮬레이션 비행 테스트에서는 동일 waypoint 비행에서 node parameter를 변경해가며 진행하였다. 아래 Fig. 26은 장애물 회피 비행경로 생성과 연관된 파라미터들이다. 테스트에서는 obstacle\_cost\_parameter, pitch\_cost\_parameter, yaw\_cost\_parameter, velocity\_cost\_parameter 변경 및 조합을 통한 충돌회피 성능 변화에 초점을 두고 실험을 진행하였다.

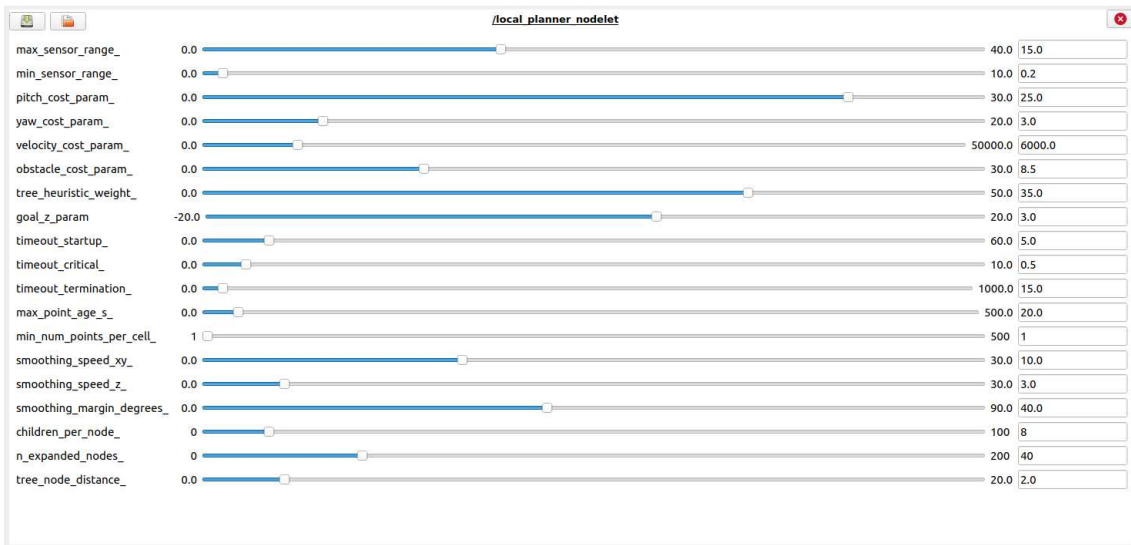


Fig. 26 dynamic\_reconfigure node parameter

### 2.1.2 실외 비행 환경구성

실외 비행 테스트는 2대의 사다리차를 회피하여 사이를 통과하는 임무, 건물 및 나무를 회피하여 돌아가는 임무 등을 선정하여 실험을 진행하였다. 실외 비행 테스트에 사용한 기체들을 제원과 형상은 Fig. 27과 Fig. 28에 정리하였다.



항목	내용
최대이륙중량	18kg
모터 축간 거리	약 1200mm
비행제어장치	Pixhawk 6C
GPS	Holybro RTK
Battery	12s 16000mah
Endurance	약 30분
Camera	Realsense D455
Onboard PC	Jetson Xavier NX

Fig. 27 실외 비행 테스트 기체 (사다리차 회피 임무)



항목	내용
최대이륙중량	10kg
모터 축간 거리	약 730mm
비행제어장치	Pixhawk 6C
GPS	Holybro RTK
Battery	6s 22000mah
Endurance	약 30분
Camera	Realsense D455
Onboard PC	Jetson Xavier NX

Fig. 28 실외 비행 테스트 기체 (건물 및 나무 회피 임무)

실험 환경은 아래와 같은 환경에서 진행하였다.

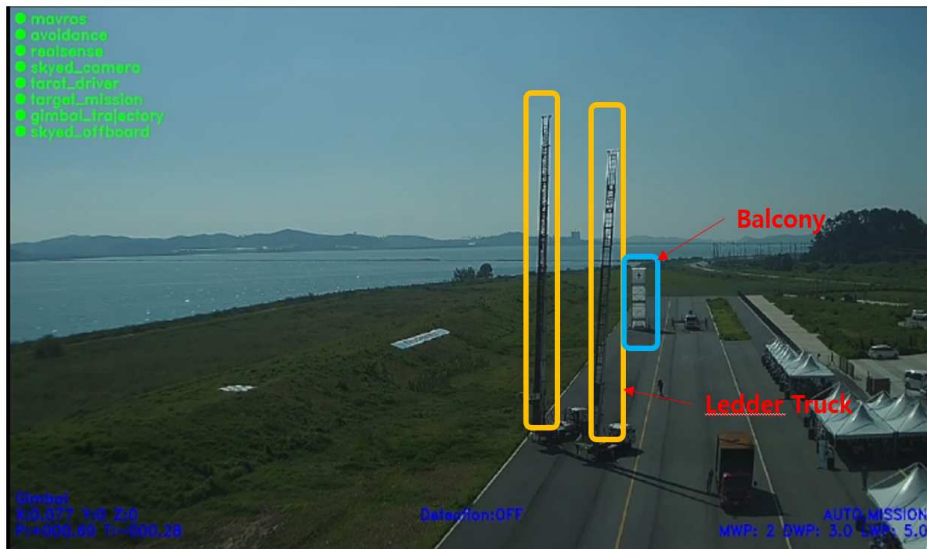


Fig. 29 사다리차 사이를 통과하는 임무 환경

첫 번째 실외 비행 테스트는 비행 경로상 있는 사다리차를 회피하여 사이를 통과하는 임무이다. 장애물인 사다리차의 사이 간격은 6m이고, 높이는 약 38m이다. 비행 임무의 경우 고도는 15m이고, 비행 속도는 5m/s로 설정하였다.





Fig. 30 경로상 나무 및 건물이 있는 임무 환경

두 번째 실외 비행 테스트는 비행 경로상 있는 건물 및 나무를 회피하는 임무이다. 나무를 먼저 회피한 후 뒤에 있는 건물도 회피하여 돌아가는 임무이다. 비행 임무의 경우 고도는 3m이고, 비행 속도는 5m/s로 설정하였다.



Fig. 31 경로상 큰 건물이 있는 임무 환경

세 번째 실외 비행 테스트는 비행 경로상 있는 큰 건물을 회피하는 임무이다. 장애물의 크기는 가로는 약 10m, 세로는 약 8m이다. 비행 임무의 경우 고도는 3m이고, 비행 속도는 5m/s로 설정하였다.



Fig. 32 경로상 큰 나무가 있는 임무 환경

네 번째 실외 비행 테스트는 비행 경로상 있는 큰 나무를 회피하는 임무이다. 나무를 통과하며 작은 나뭇가지를 감지하는지에 대한 테스트이다. 비행 임무의 경우 고도는 3m이고, 비행 속도는 5m/s로 설정하였다.

### 2.1.3 실외 자율 비행 시스템

실험은 Mission mode와 Offboard mode로 진행하였으며, 실외 자율 비행을 위한 시스템은 아래 Fig. 33과 같이 구성하였다. 기체의 비행 제어 장치는 상위(High-level) 제어기인 Jetson Xavier NX와 하위(Low-level) 제어기인 Pixhawk로 이루어져 있다. 기본자세를 제어하는 하위(Low-level) 제어기 Pixhawk(PX4)와 이미지 정보처리 및 경로에 관한 실질적인 비행 제어를 담당하는 상위(High-level) 제어기 Jetson Xavier NX는 Robot Operation System (ROS)을 기반으로 통합되어 있다.



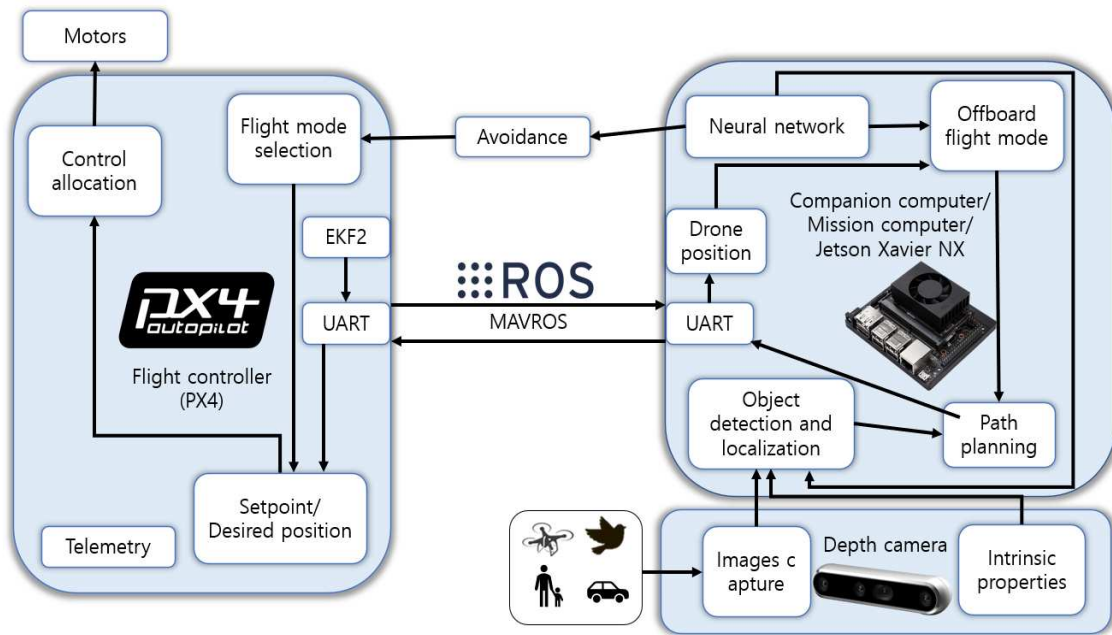


Fig. 33 실외 자율 비행 시스템 구성도

## 2.2 실험 결과

### 2.2.1 단일 카메라 시뮬레이션 비행 실험 결과

#### 2.2.1.1 복잡한 블록 사이 회피 비행 결과

obstacle\_cost\_parameter의 경우 장애물 간격에 의존한다. 장애물 간격보다 크게 설정하게 되면 장애물 사이를 통과하지 못하고, 반대로 너무 작게 설정하게 되면 장애물 한쪽 끝에 붙게 되어 충돌 위험이 커진다. 따라서 기체의 크기와 장애물 사이 간격을 고려하여 중간값을 설정하여야 장애물 중간 지점으로 안전하게 회피하게 된다. 테스트 환경에서의 장애물 간의 간격은 3m이며, 기체의 크기가 작아 기체 크기는 고려하지 않았다. 따라서 1.5로 설정하였다.

yaw\_cost\_parameter 변경 및 조합에 따른 충돌회피 성능 변화를 분석하기 위해 pitch\_cost\_parameter는 25, velocity\_cost\_parameter는 49000, obstacle\_cost\_parameter는 1.5로 고정하고 테스트를 진행하였다.

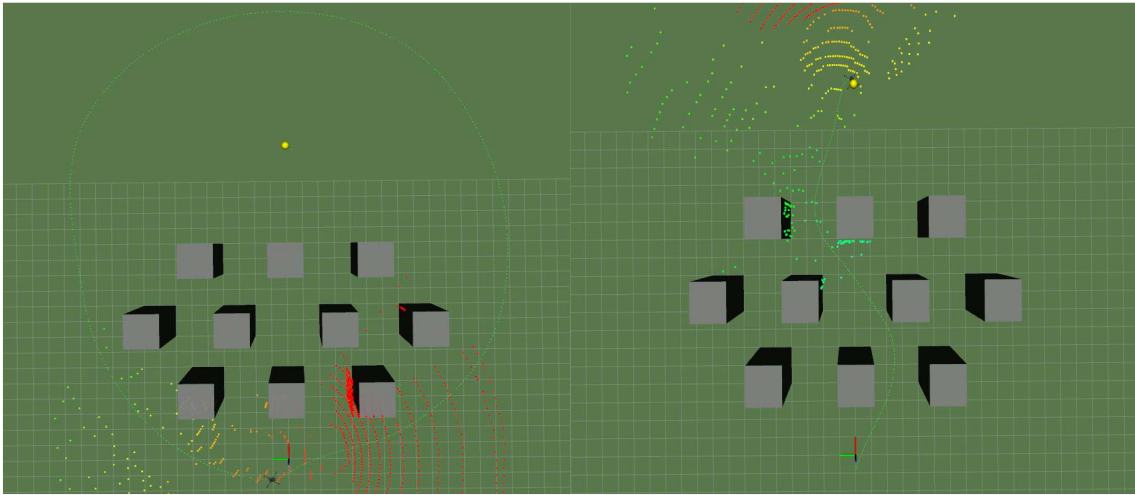


Fig. 34 yaw\_cost\_parameter = 1

Fig. 35 yaw\_cost\_parameter = 5

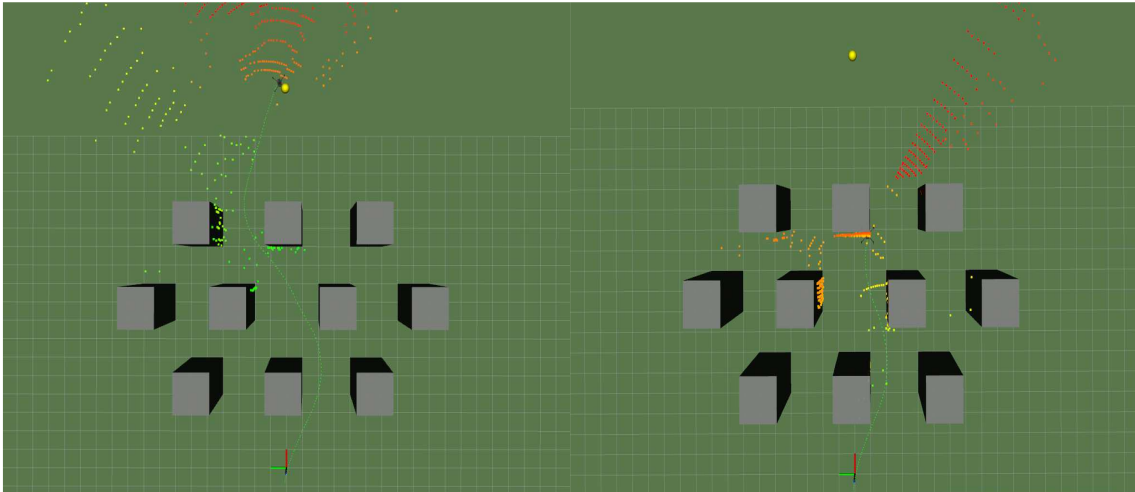


Fig. 36 yaw\_cost\_parameter = 10

Fig. 37 yaw\_cost\_parameter = 19

위 그림들은 동일 조건에서 yaw\_cost\_parameter를 변경 해가며 테스트한 결과이다. 파라미터값이 1인 경우 장애물 사이를 통과하지 못하고, 목적지도 찾아가지 못하는 결과가 나왔다. 또한 파라미터값이 19인 경우에는 장애물을 제대로 회피하지 못하고 충돌하는 결과가 나왔다.

velocity\_cost\_parameter 변경 및 조합에 따른 충돌회피 성능 변화를 분석하기 위해 앞선 yaw\_cost\_parameter 테스트 조건에서 velocity\_cost\_parameter를 1000으로 변경하여 테스트를 진행하였다.

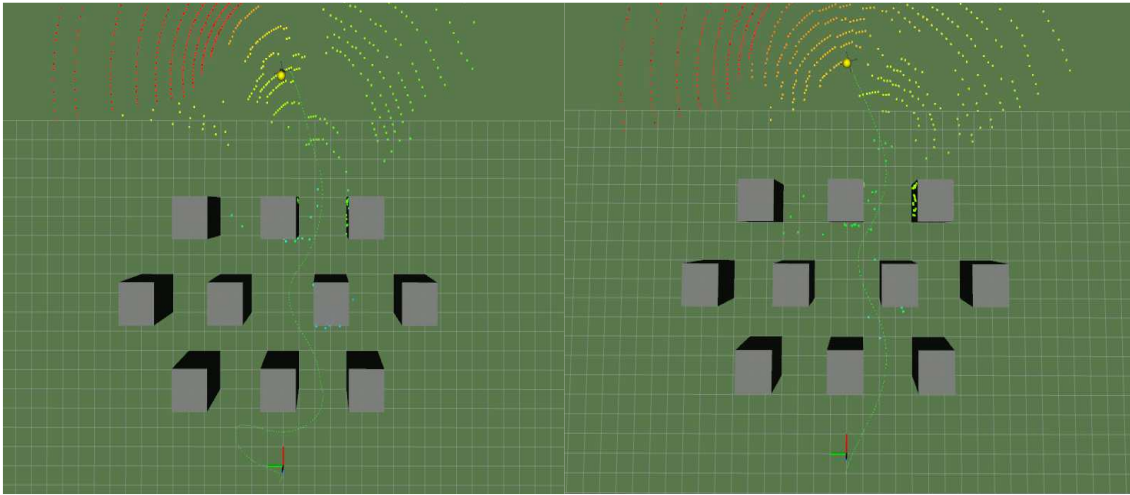


Fig. 38 yaw\_cost\_parameter = 1

Fig. 39 yaw\_cost\_parameter = 5

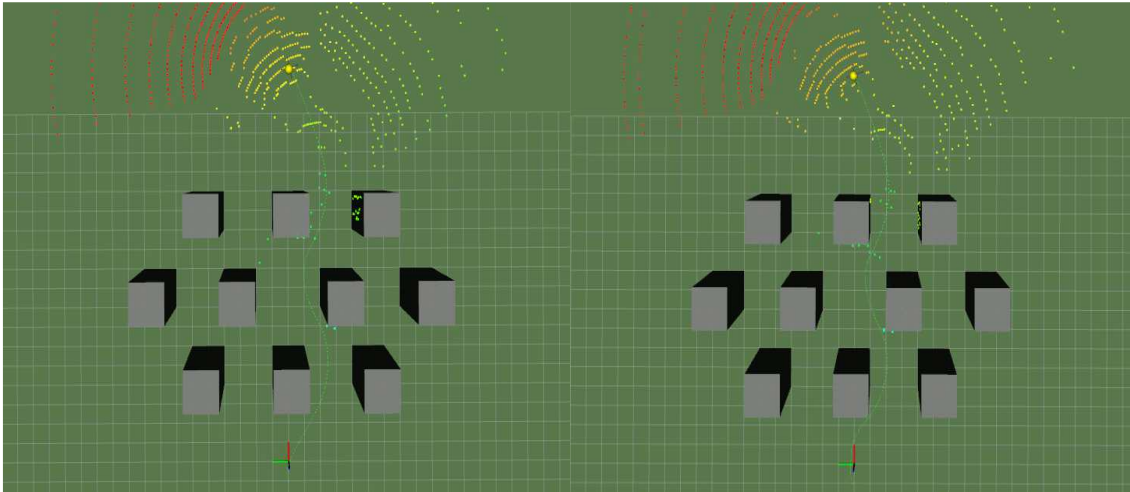


Fig. 40 yaw\_cost\_parameter = 10

Fig. 41 yaw\_cost\_parameter = 19

위 그림은 velocity\_cost\_parameter를 1000으로 변경 후 테스트를 진행한 결과이다. 파라미터값 49000에서 통과하지 못했거나, 충돌했던 문제는 파라미터값 1000에서는 발생하지 않았다.

pitch\_cost\_parameter 변경 및 조합에 따른 충돌회피 성능 변화를 분석하기 위해 yaw\_cost\_parameter를 5, obstacle\_cost\_parameter를 1.5, velocity\_cost\_parameter를 1000으로 고정하고 테스트를 진행하였다.

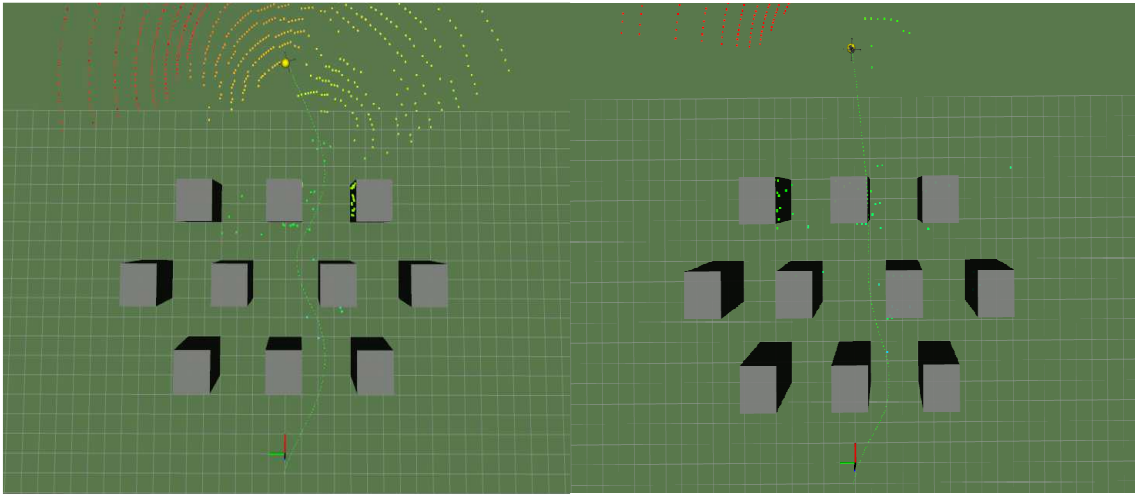


Fig. 42 pitch\_cost\_parameter = 25

Fig. 43 pitch\_cost\_parameter = 5

위 그림은 pitch\_cost\_parameter에 따른 비행경로 변화 결과를 보여주는 그림이다. 같은 조건에서 pitch\_cost\_parameter가 낮을수록 장애물 회피할 때 좌우로만이 아닌 고도 상승을 통한 회피가 이루어지는 것을 볼 수 있다.

### 2.2.1.2 긴 블록 회피 비행 결과

이전 테스트와 마찬가지로 yaw\_cost\_parameter 변경 및 조합에 따른 충돌회피 성능 변화 테스트를 먼저 진행하였다. 해당 파라미터를 제외한 나머지 값들은 고정하였다. 고정된 파라미터값은 obstacle\_cost\_parameter 1.5, pitch\_cost\_parameter 25, velocity\_cost\_parameter 49000이다.

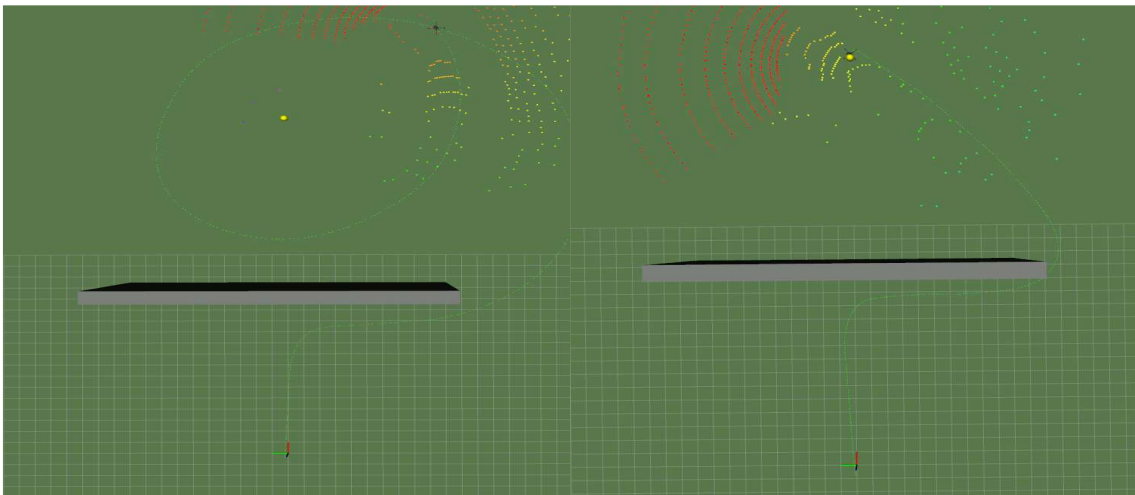


Fig. 44 yaw\_cost\_parameter = 1

Fig. 45 yaw\_cost\_parameter = 5

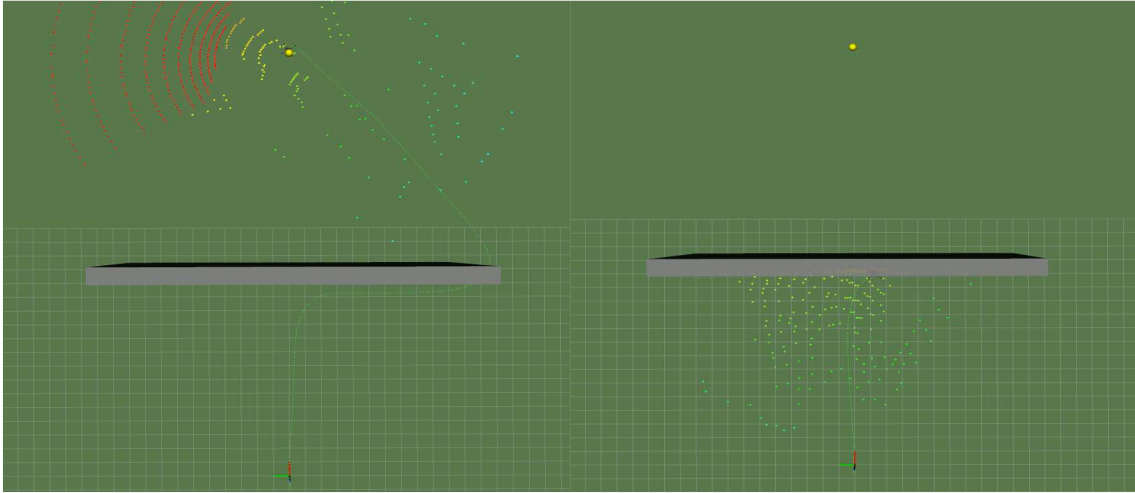


Fig. 46 yaw\_cost\_parameter = 10

Fig. 47 yaw\_cost\_parameter = 19

위 그림들은 동일 조건에서 yaw\_cost\_parameter를 변경 해가며 충돌회피 성능 변화를 테스트한 결과이다. 파라미터값이 1인 경우 장애물은 통과하였지만, 목적지를 찾아가지 못하는 결과가 나왔다. 또한 파라미터값이 19인 경우에는 장애물을 제대로 회피하지 못하고 충돌하는 결과가 나왔다. 이는 복잡한 장애물에서 진행한 테스트와 동일한 결과를 보인다.

velocity\_cost\_parameter 변경 및 조합에 따른 충돌회피 성능 변화 테스트를 진행하였다. 마찬가지로 해당 파라미터값을 제외한 나머지 값들은 고정하였다. 고정된 파라미터 값은 obstacle\_cost\_parameter 1.5, pitch\_cost\_parameter 25, yaw\_cost\_parameter 5이다.

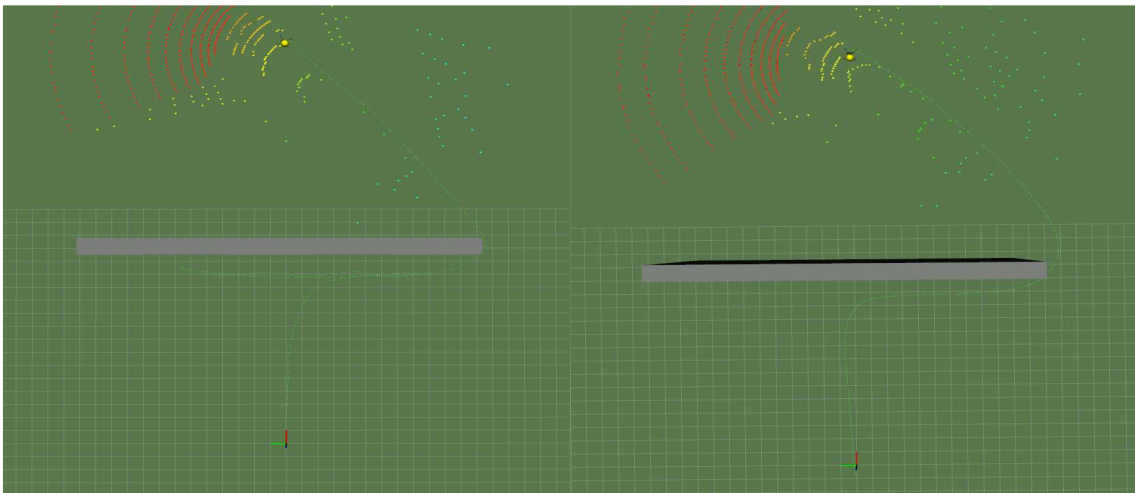


Fig. 48 velocity\_cost\_parameter = 14000

Fig. 49 velocity\_cost\_parameter = 49000

위 그림들은 동일 조건에서 velocity\_cost\_parameter를 변경 해가며 테스트한 결과이다. 파라미터값이 14000 아래일 경우 장애물을 통과하지 못하는 결과가 나왔고,

14000에서도 여러 번 반복 끝에 통과하는 모습을 보였다.

obstacle\_cost\_parameter 변경 및 조합에 따른 충돌회피 성능 변화 테스트도 진행하였다. 테스트는 yaw\_cost\_parameter 5, pitch\_cost\_parameter 25, velocity\_cost\_parameter 14000으로 고정하고 obstacle\_cost\_parameter만 변경하여 테스트를 진행하였다.

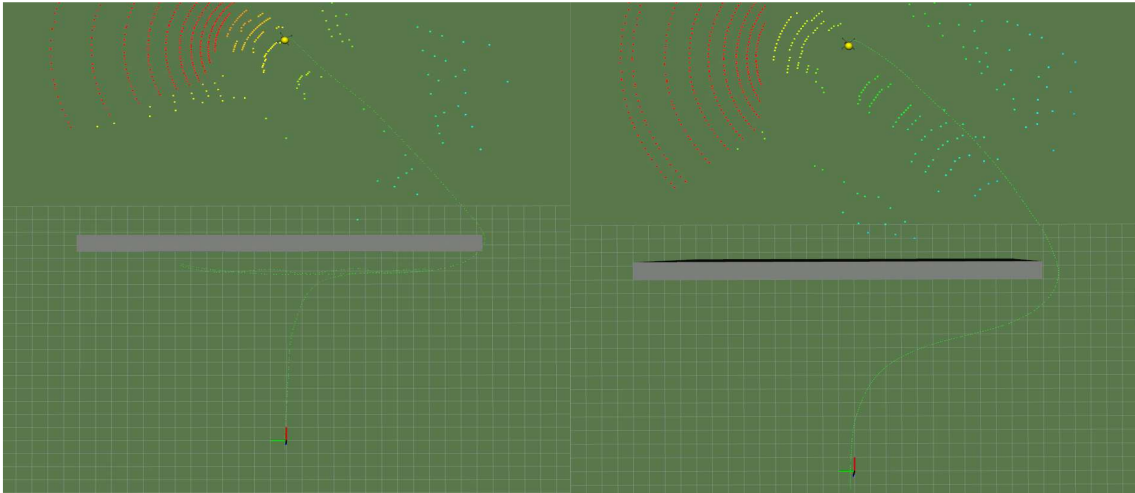


Fig. 50 obstacle\_cost\_parameter = 1.5

Fig. 51 obstacle\_cost\_parameter = 5

위 결과를 보면 동일 조건에서 obstacle\_cost\_parameter 증가가 단일 장애물 회피에 있어서는 더 안정적인 회피를 보여준다. 또한 동일 조건에서 pitch\_cost\_parameter만 5로 수정하여 추가 테스트를 진행하였다.



Fig. 52 obstacle\_cost\_parameter = 1.5

Fig. 53 obstacle\_cost\_parameter = 5

위 결과를 보면 obstacle\_cost\_parameter를 1.5로 하였을 때는 드론이 장애물을 넘지 못하였고 충돌하였다.



### 2.2.1.3 블록 2개 사이 회피 비행 결과

이전 테스트 결과를 통해 `obstacle_cost_parameter`는 2.5, `pitch_cost_parameter` 25, `velocity_cost_parameter` 49000으로 고정하고 `yaw_cost_parameter`만 수정하며 테스트를 진행하였다.

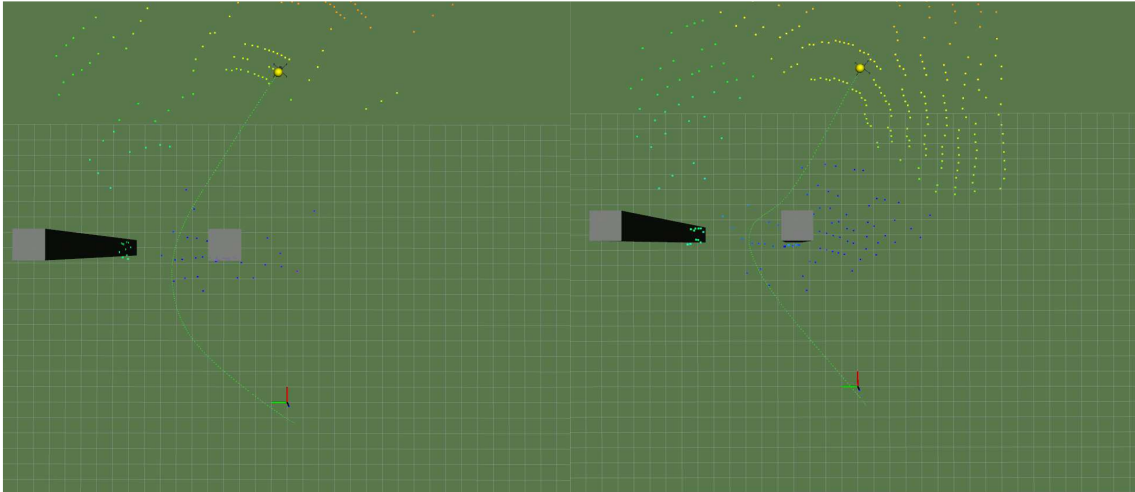


Fig. 54 `yaw_cost_parameter` = 1

Fig. 55 `yaw_cost_parameter` = 5

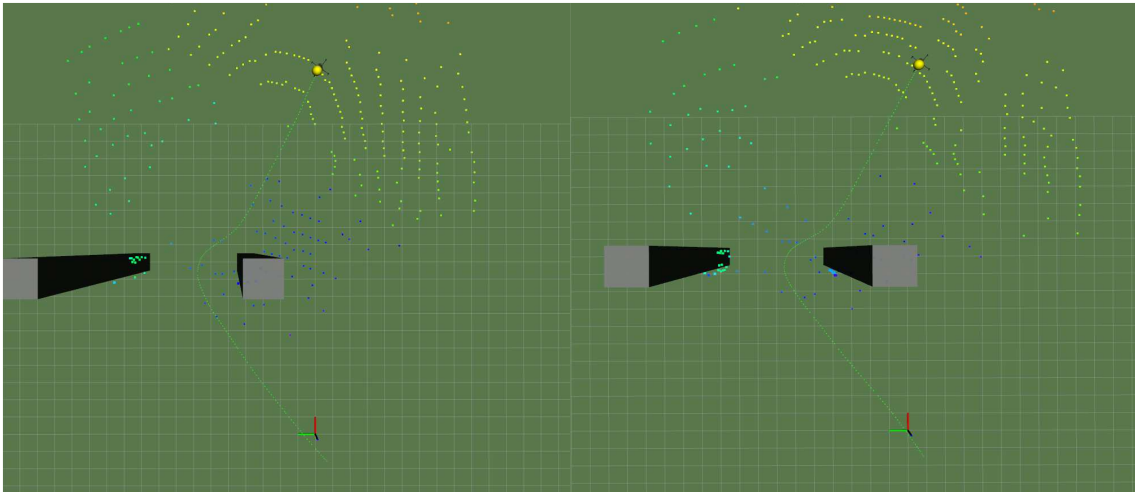


Fig. 56 `yaw_cost_parameter` = 10

Fig. 57 `yaw_cost_parameter` = 19

시뮬레이션 결과를 정리하면 다음과 같다.

`obstacle_cost_parameter`의 경우 장애물로부터의 거리를 의미하며, 높을수록 단일 장애물에 대해서는 안정적인 경로 생성이 가능하지만, 다중 장애물에서는 장애물 사이 경로 생성이 불가능해진다.

`pitch_cost_parameter`의 경우 이전 노드에서의 `pitch` 각도를 유지하는 것에 대한 가중치를 의미하며, 낮을수록 고도 상승을 통한 회피 경로 생성을 시도한다. 파라미터

설정에 있어 기체의 상승 속도를 고려 하여야 한다.

yaw\_cost\_parameter의 경우 이전 노드에서의 yaw 각도를 유지하는 것에 대한 가중치를 의미하며, 낮을수록 안정적인 회피 경로를 생성하지만, 2 이하의 값을 가지게 되면 목적지를 찾지 못하는 현상을 보인다.

velocity\_cost\_parameter의 경우 path smoothness에 대한 가중치를 의미하며, 높을수록 단일 장애물에 대해서는 비행시간이 감소하였지만, 다중 장애물에서는 낮을수록 안정적인 회피 경로 생성 및 비행시간이 감소하였다.

## 2.2.2 단일 카메라 실외 비행 결과

### 2.2.2.1 사다리차 회피 비행 결과

사다리차 사이 통과를 위한 파라미터의 경우 블록 2개 사이 회피 비행 시뮬레이션 결과를 바탕으로 설정하였다. pitch\_cost\_parameter는 25, yaw\_cost\_parameter는 5, velocity\_cost\_parameter는 25000, obstacle\_cost\_parameter는 2.5로 설정하였다. 임무 특성상 동일 장애물에 대해 2번의 회피를 진행하였고, 비행은 mission mode로 임무를 진행하였다. 아래 사진은 장애물 인식과 실제 비행경로를 보여주고 있다.

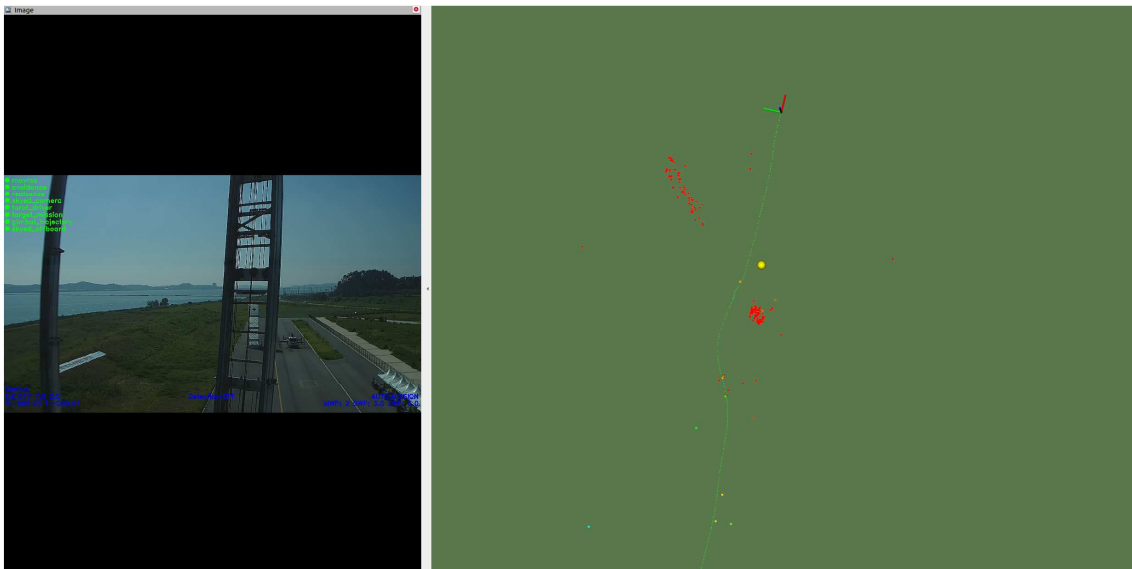


Fig. 58 첫 번째 장애물 인식 및 회피 경로



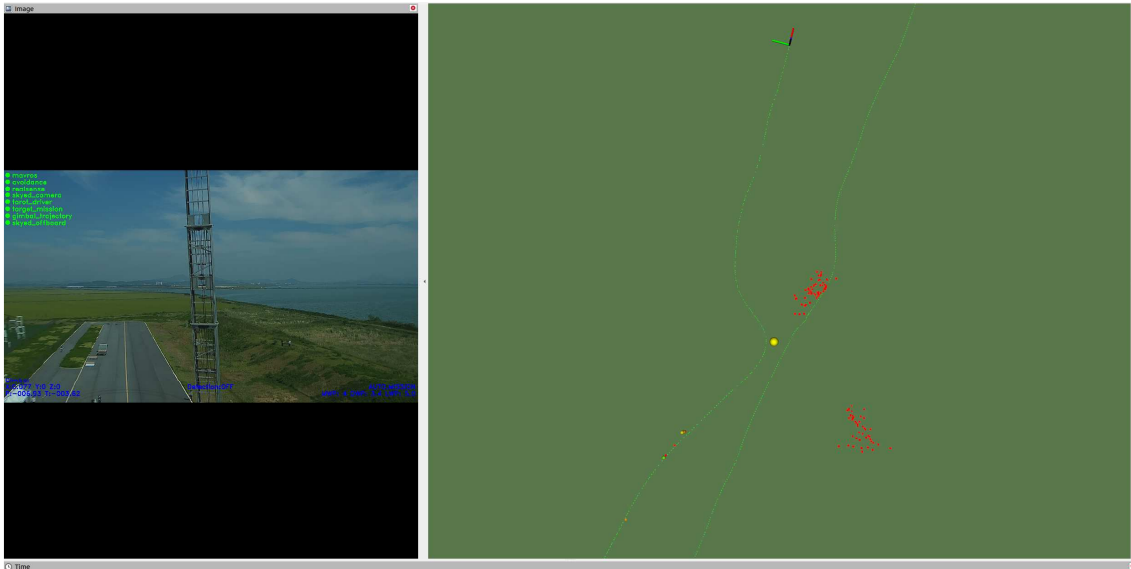


Fig. 59 두 번째 장애물 인식 및 회피 경로

결과를 보면 첫 번째 통과인 경우 시뮬레이션처럼 두 장애물(사다리차) 사이로 회피 경로를 생성하여 통과한 것을 볼 수 있다. 하지만 두 번째 통과에서 장애물 사이가 아닌 바깥쪽으로 회피 경로를 생성하여 통과하였다. 이는 시뮬레이션의 경우 정확히 장애물 중간에 경로점을 주었지만, 실제 대회에서의 경로점은 장애물 중간이 아닌 한 쪽 장애물에 치우쳐 있었고, 그로 인해 최소경로를 생성하려는 알고리즘 특성상 시뮬레이션과는 다른 경로를 생성하였다.

#### 2.2.2.2 건물 및 나무 회피 비행 결과

건물 및 나무 회피를 위한 파라미터는 시뮬레이션 결과를 바탕으로 설정하였다. pitch\_cost\_parameter는 25, yaw\_cost\_parameter는 5, velocity\_cost\_parameter는 25000, obstacle\_cost\_parameter는 5로 설정하였다. 비행은 offboard mode로 임무를 진행하였다. 아래 사진은 장애물 인식과 실제 비행경로를 보여주고 있다.

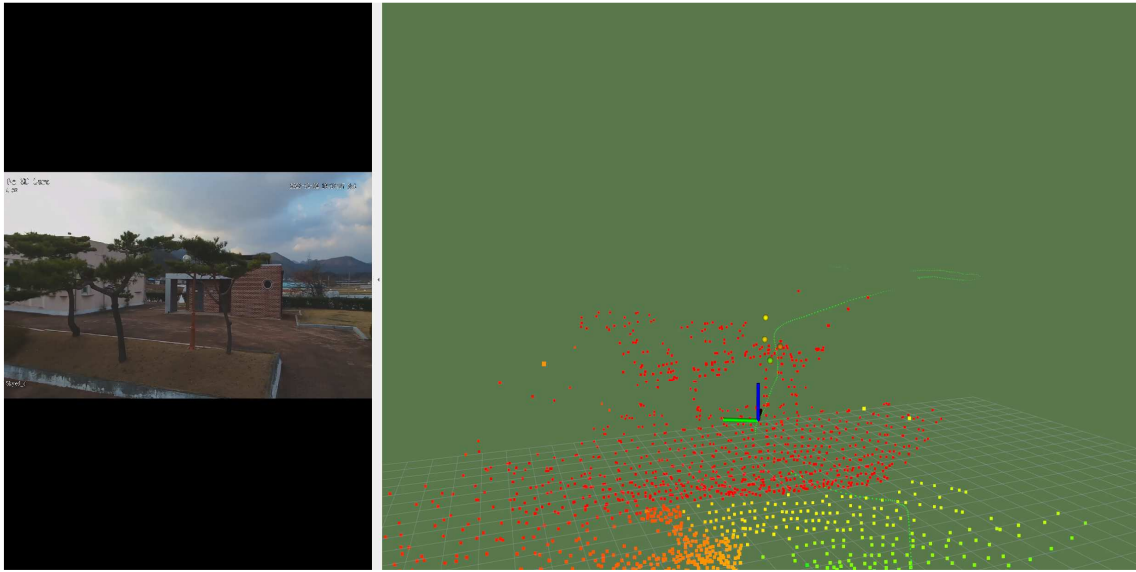


Fig. 60 나무 인식 및 회피 경로

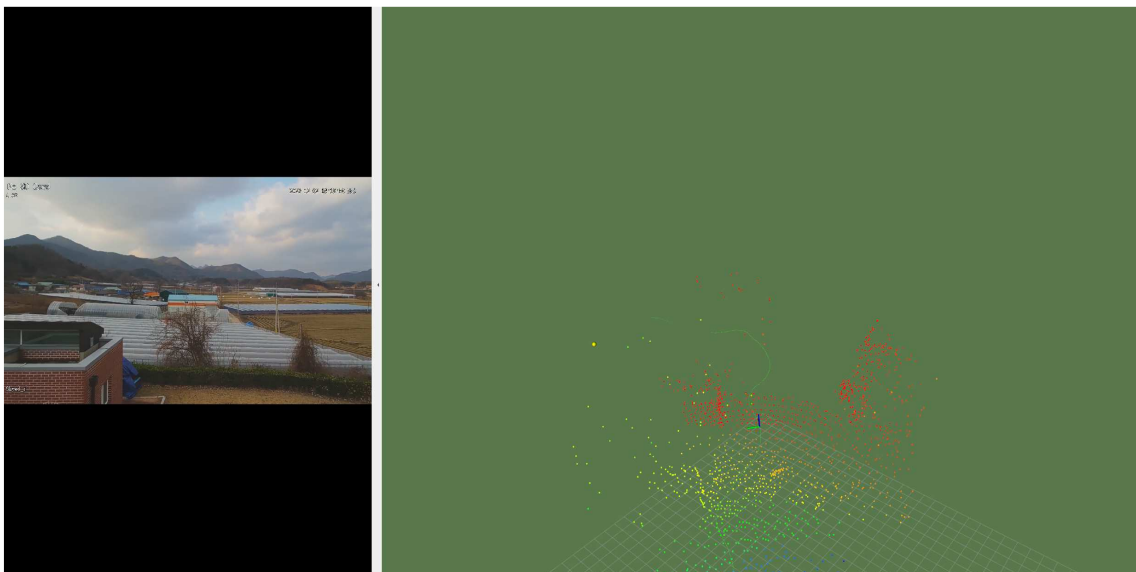


Fig. 61 나무, 건물 인식 및 회피 경로

결과를 보면 시뮬레이션과 유사한 장애물 회피 경로를 생성하는 모습을 보여주고 있다. obstacle\_cost\_parameter를 5로 설정하였고, 장애물을 비교적 먼 거리에서부터 인식하고 회피 경로를 생성하였다. 또한 지면을 높이 변화를 인식한 드론은 고도를 상승하며 회피 경로를 생성하였다. 고도 상승과 더불어 좌우로 회피 경로를 생성하였다. 시뮬레이션의 경우 지면의 고도 변화에 따른 테스트는 수행하지 않았지만, 이번 실외 비행 테스트를 통하여 지면의 고도 변화에 따라 회피 경로 생성 변화를 볼 수 있었다.

### 2.2.2.3 큰 건물 회피 비행 결과

큰 건물 회피를 위한 파라미터는 시뮬레이션 결과를 바탕으로 설정하였다. `pitch_cost_parameter`는 25, `yaw_cost_parameter`는 5, `velocity_cost_parameter`는 25000, `obstacle_cost_parameter`는 5로 설정하였다. 비행은 `offboard mode`로 임무를 진행하였다.

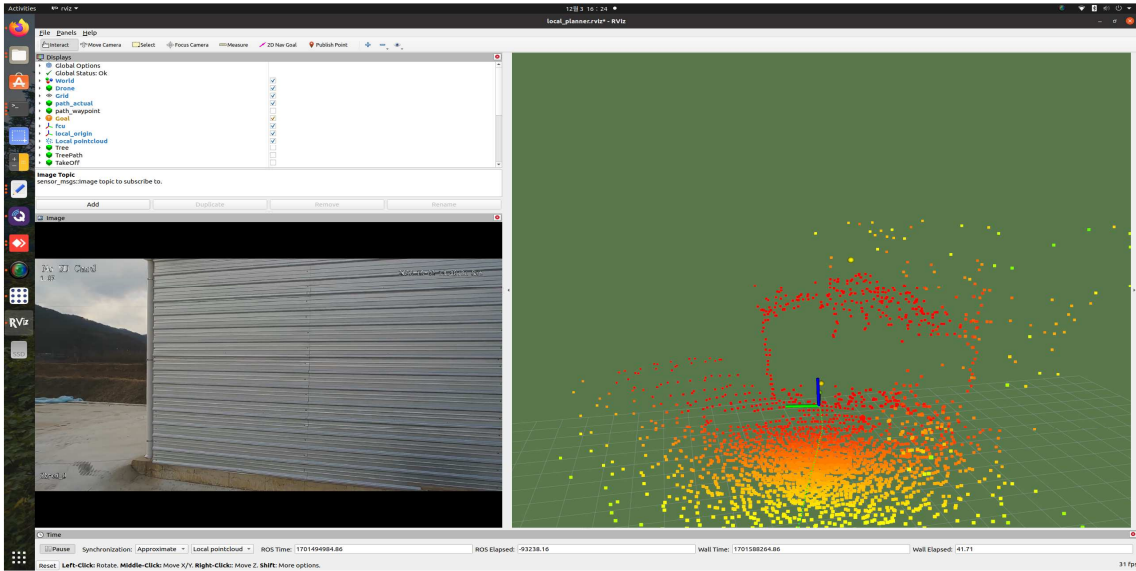


Fig. 62 건물 인식 실패

결과를 보면 경로상 장애물인 건물을 전혀 인식하지 못하고 있는 것을 확인할 수 있다. 건물 외벽 특성으로 인해 적외선 빛이 굴절되었고, RGB-D 카메라는 장애물을 인식하지 못하여 회피 경로를 생성하지 못하였다. 또한 적외선 빛이 투과되는 유리 또는 비닐 재질인 경우에도 RGB-D 카메라는 장애물로 인식하지 못하고 회피 경로를 생성하지 못한다.

### 2.2.2.4 큰 나무 회피 비행 결과

큰 나무 회피를 위한 파라미터는 시뮬레이션 결과를 바탕으로 설정하였다. `pitch_cost_parameter`는 25, `yaw_cost_parameter`는 5, `velocity_cost_parameter`는 25000, `obstacle_cost_parameter`는 5로 설정하였다. 비행은 `offboard mode`로 임무를 진행하였다.

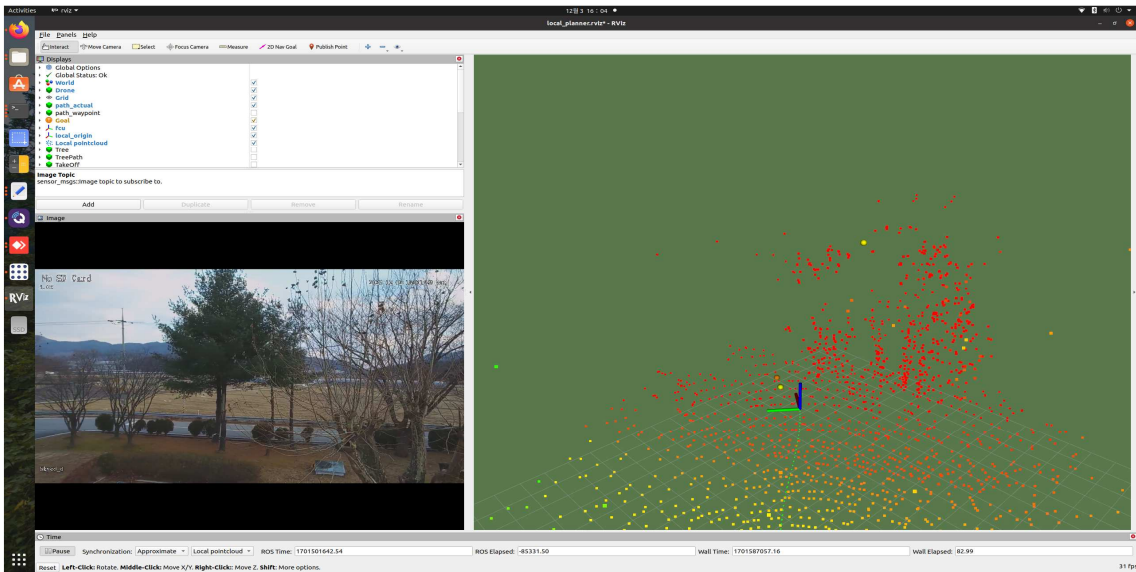


Fig. 63 작은 나뭇가지 인식 실패

결과를 보면 작은 나뭇가지가 듬성듬성 있는 경우 장애물로 인식하지 못하고 회피 경로를 생성하지 못한다. 반면 작은 나뭇가지들이 뭉쳐져 있거나 나뭇잎이 있는 경우에는 장애물로 인식하여 회피 경로를 생성하였다.

### 2.2.3 다중 카메라 시뮬레이션 비행 실험 결과

테스트 조건은 다음과 같다. 복잡한 블록을 경우 pitch\_cost\_parameter는 25, yaw\_cost\_parameter는 3, obstacle\_cost\_parameter는 2, velocity\_cost\_parameter는 1000으로 고정 후 비교하였다. 긴 블록을 경우 pitch\_cost\_parameter는 25, yaw\_cost\_parameter는 3, obstacle\_cost\_parameter는 2, velocity\_cost\_parameter 12000으로 고정 후 비교하였다. 목표까지의 직선거리는 25m로 모두 동일하다.

#### 2.2.3.1 카메라 1개 실험 결과

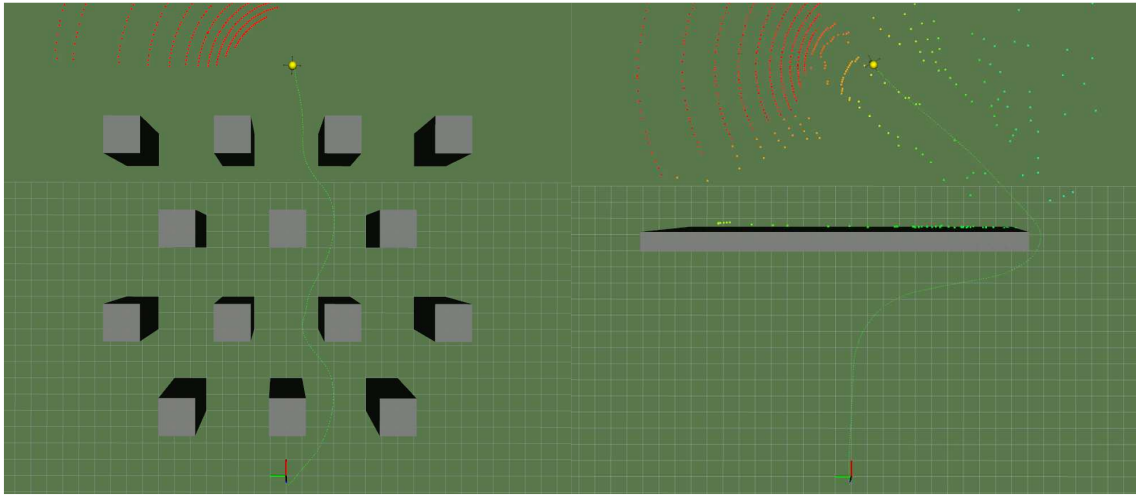


Fig. 64 카메라 1개 복잡한 블록

Fig. 65 카메라 1개 긴 블록

카메라 1개의 경우 복잡한 블록 환경에서 목표지점까지 도달하는데 약 35초, 긴 블록 환경에서는 목표지점까지 도달하는데 약 41초의 시간이 소요되었다.

### 2.2.3.2 카메라 3개 실험 결과

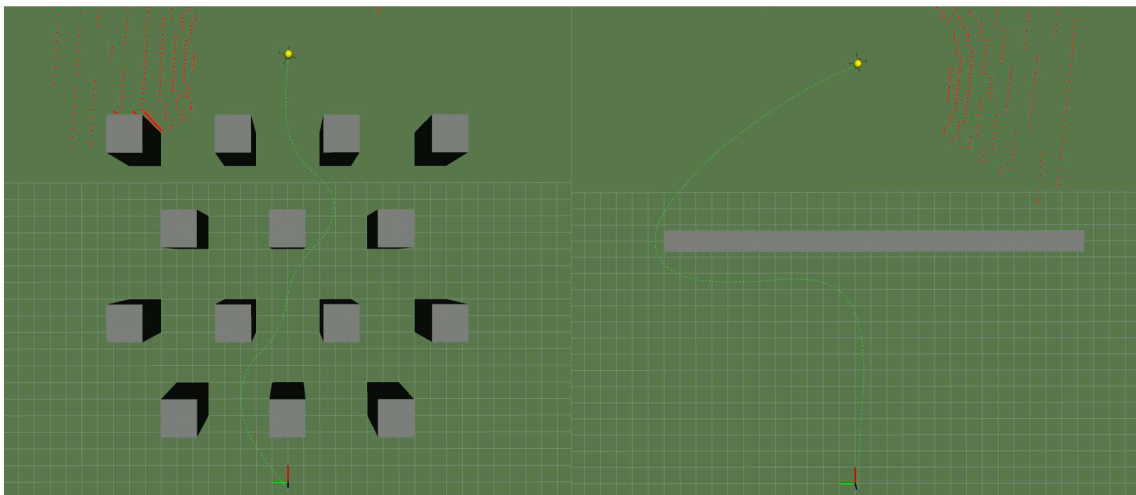


Fig. 66 카메라 3개 복잡한 블록

Fig. 67 카메라 3개 복잡한 블록

카메라 3개의 경우 복잡한 블록 환경에서 목표지점까지 도달하는데 약 28초, 긴 블록 환경에서는 목표지점까지 도달하는데 약 34초의 시간이 소요되었다.

결론적으로 카메라 1개일 때보다 3개일 때가 더 안정적인 회피 경로와 빠른 비행시간을 보여주었다. 카메라가 1개인 경우 드론이 비행하면서 일정 간격마다 속도를 줄여 좌우를 살피며 장애물 유무를 파악하거나, 경로에 대한 cost를 계산한다. 또한 장

애물 통과 시 장애물 사이 간격을 파악하는 등을 행동으로 인해 빠른 속도 비행이 어렵다. 하지만 3개의 카메라가 전방 180도를 볼 수 있어 1개일 때와 같은 행동이 필요하지 않고 비교적 빠른 속도 비행이 가능하다. 또한 장애물 사이 거리를 정확하게 알 수 있어 더 효율적인 경로 생성이 가능하다. 하지만 카메라가 늘어난 만큼 포인트 클라우드 개수도 늘어나게 되고 회피 경로 생성을 위한 계산량이 늘어난다. 드론에 탑재된 상위제어기인 Jetson Xavier NX의 경우 그래픽 성능은 좋지만, cpu 성능이 부족하다는 단점이 있고, 회피 경로 계산을 cpu로 하는 시스템 특성상 회피 경로를 안정적으로 생성하지 못한다. 그로 인해 실외 비행 테스트는 진행하지 못하고 시뮬레이션 결과만을 이용하였다.

## 4. 결 론

본 논문에서는 드론의 비행 중에 발생할 수 있는 장애물에 대한 회피 및 경로 생성 알고리즘을 분석하고 검증하는 내용을 주로 다루었다. 회피 알고리즘을 분석하기 위해서 Gazebo 시뮬레이션 환경에서 파라미터 변경 및 조합에 따른 충돌회피 성능 변화 테스트를 진행하였고, 실외 비행 테스트를 통한 알고리즘 검증 테스트를 수행하였다. Gazebo 시뮬레이션 테스트에서 회피 경로 생성에 연관된 node parameter 조합에 따른 충돌회피 성능 변화를 분석하였고, 장애물 별로 적절한 node parameter 최적값을 도출하였다. 또한 실외 비행 테스트에서 시뮬레이션 테스트를 통해 도출한 node parameter 조합에 따른 충돌회피 성능을 검증하였고, 단일 카메라 회피 시스템 한계점을 도출하였다.

장애물 인식 및 회피를 위한 센서로는 적외선 센서와 이미지 센서가 결합된 Intel사의 RGB-D 카메라를 이용하였다. 대부분의 장애물에 대해서는 탐지 및 회피가 안정적으로 이루어진 것을 확인할 수 있다. 하지만 실험에 사용된 RGB-D 카메라의 한계로 인해 장애물이 빛의 흡수하는 재질이거나 얇은 나뭇가지일 경우에는 장애물을 탐지하지 못하여 회피 경로를 생성하지 못하였고 충돌로 이어졌다.

여러 테스트를 통해서 장애물 회피 알고리즘을 분석 및 검증한 결과 도심 속에서의 장애물 탐지 및 회피는 충분히 가능하다고 생각한다. 또한 이미지 센서를 이용함에 따라 추가적인 센서를 부착하지 않아도 객체 인식과 같은 다양한 용도로도 활용이 될 수 있다. 하지만 3DVFH\* 알고리즘은 회피 경로 생성을 하기 위해 고사양의 온보드 PC가 탑재되어야 하고, 비행 속도가 느리다. 그리고 카메라 1개만 탑재했을 경우 장애물 사이 회피에 있어서 양쪽 장애물에 거리정보를 동시에 얻지 못해 정확히 중앙으로 회피하기 힘들고, 비행 중 속도를 줄여가며 좌우를 살펴 주변 장애물들을 탐지하고 경로 cost를 계산해야 하는 단점이 존재한다.

본 논문에서는 이러한 단점을 해결하기 위해 RGB-D 카메라 3개를 전방 180도로 배치하여 테스트를 진행하였다. Gazebo 시뮬레이션을 통한 테스트를 진행하였다. RGB-D 카메라 3개를 탑재했을 경우 1개에 비해 빠른 비행 속도와 안정적인 회피가 이루어지는 것을 확인하였다. 하지만 카메라가 3개로 늘어난 만큼 데이터양도 동시에 늘어났고, 현재 드론에 탑재된 온보드 PC의 CPU 코어 수 한계로 인해 2개 이상의 RGB-D 카메라를 연결했을 경우 경로 생성 알고리즘이 정상적으로 동작하지 않는 것을 확인하였다. 따라서 실외 비행 테스트는 진행하지 못하였다.

## 참고 문헌

1. Titterton, David, John L. Weston, and John Weston, "Strapdown inertial navigation technology", United Kingdom : The Institution of Electrical Engineers, 2004.
2. Jung-Woo Park, Hyon-Dong Oh, Min-Jea Tahk, "UAV collision avoidance based on geometric approach", s.l. : SICE Annual Conference, 2008, pp. 2122-2126.
3. Hamid Alturbek, James F. Whidborne, "Visual Flight Rules-Based Collision Avoidance Systems for UAV Flying in Civil Aerospace", s.l. : robotics, MDPI, 2020.
4. O. Khatib, "Real time obstacle avoidance for manipulators and mobile robots" Proc. IEEE Int. Conf. on Robotics and Automation, vol. 2, pp. 500-505, 25-28 March 1985, St. Louis, MO, USA.
5. 김필준, "포텐셜 필드 기법을 이용한 항공기 위협 회피 궤적 설계", 서울대학교 대학원, 2007
6. 안승규. "장애물의 상대속도를 고려한 포텐셜필드 기반 무인항공기 충돌회피 기법 연구", 한서대학교 대학원, 2019.
7. J. Borenstein, Y. Koren. 3, "The vector field histogram - fast obstacle avoidance for mobile robots", s.l. : IEEE Transaction on Robotics and Automation, 1991, Vol. 7, pp. 278 - 288.
8. Erika Mai, "Obstacle Detection and Avoidance Techniques for Unmanned Aerial Vehicles", Polytechnic University of Turin, 2020
9. J. Borenstein, Y. Koren. 4, "Histogramic in-motion mapping for mobile robot obstacle avoidance", s.l. : IEEE Transaction on Robotics and Automation, 1991, Vol. 7, pp. 535 - 539.
10. I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robotics", in Robotics and Automation, 1988. Proceedings. 1998 IEEE International Conference on, vol. 2. IEEE, 1998, pp. 1572-1577.
11. A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilities 3d mapping framework based on octrees",



- Autonomous Robots, vol. 34, no. 3, pp. 198-206, 2013.
12. Simon Vanneste, Ben Bellekens, Maarten Weyn. "3DVFH+: Real-Time Three-Dimensional Obstacle Avoidance Using an Octomap", MORSE@ STAF, Vol. 1319, pp. 91-102, Jul. 2014.
  13. Kai M. Wurm, Armin Hornung, Maren Bennewitz, Cyrill Stachniss, Wolfram Burgard. "OctoMap : A Probabilistic, Flexible, and Compact 3D Map Representation for Robotics Systems", ICSRT 2022: 2022 the 3rd International Conference on Service Robotics Technologies, pp. 48-53, March 2022.
  14. I. Ulrich and I. Borenstein, "VFH\*: Local obstacle avoidance with look ahead verification," IEEE International Conference on Robotics and Automation (ICRA), vol. 2, pp. 1572-1577, 2000.
  15. T. Baumann, "Obstacle Avoidance for Drones Using a 3DVFH\* Algorithm", Masterarbeit, ETH Zürich, Zürich, 2018
  16. K. Thomessen, "A bio-inspired local path planning algorithm based on the 3DVFH\*," Bachelor's Thesis, FH Aachen, Aachen, 2022.
  17. Ruan Jacobus van Breda, "Vector Fiel Histogram Star Obstacle Avoidance System for Multicopters", Stellenbosch University, 2016

# A Study on Obstacle Avoidance Algorithm of Multi-Copter Drone using an RGB-D Camera

Yonggil Kwon

*Department of Mechanical and Aerospace Engineering*

*University of Ulsan, Korea*

## ABSTRACT

As a basic study of drone's obstacle avoidance algorithm, 3DVFH\* obstacle avoidance and path generation algorithms were analyzed and verified using PX4 open source and Depth camera. The obstacle avoidance algorithm was analyzed by configuring a Gazbo simulation environment, and the obstacle avoidance algorithm was verified through outdoor flight tests. In the Gazbo simulation, various flight environments such as an environment with complicated obstacles and an environment with a long wall were configured to analyze the change in avoidance performance through the combination of parameters related to the avoidance path, and flight tests were performed by creating an environment similar to the simulation environment in outdoor flight. Stable avoidance and path creation were achieved in most obstacles, but no obstacles were detected and could not be avoided in light-reflective materials or thin branches due to the limitations of the RGB-D camera used in the experiment. To improve the performance of the obstacle avoidance algorithm, an experiment was conducted using three RGB-D cameras in the simulation, and it showed faster avoidance and path generation ability than one camera.