



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](#)

공학석사학위 논문

배속 시뮬레이션을 이용한 공항 내의 지상 교통
관리에 관한 연구

A Study on Ground Traffic Management in Airport
Using Fast-Time Simulator

2019년 2월

인하대학교 대학원

항공우주공학과

김 태 영

공학석사학위 논문

배속 시뮬레이션을 이용한 공항 내의 지상 교통
관리에 관한 연구

A Study on Ground Traffic Management in Airport
Using Fast-Time Simulator

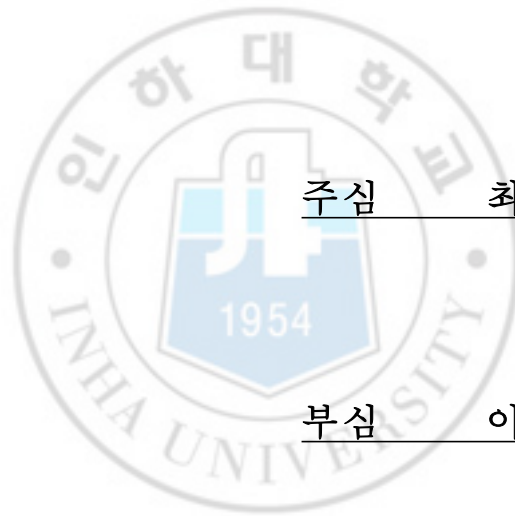
2019년 2월

지도교수 이 학 태

이 논문을 석사학위 논문으로 제출함

이 논문을 김태영의 석사학위논문으로 인정함.

2019년 2월



주심 최기영

부심 이학태

위원 유창경

목 차

1. 서론	1
1.1 연구 배경	1
1.2 연구 목표 및 진행방향	1
1.2.1 연구 목표	1
1.2.2 연구 진행 방향	2
1.3 논문의 구성	2
2. 시뮬레이터 구성	3
2.1 입력 데이터	3
2.1.1 노드	3
2.1.2 링크	4
2.1.3 스케줄	4
2.2 사용자 인터페이스 화면 구성	5
2.3 시뮬레이션 흐름도	6
3. 항공기 운동 모델	10
3.1 FCFS 스케줄러 항공기 지상 이동 운동 모델	10
3.2 2차원 질점 항공기 지상 이동 운동 모델	11
3.3 바퀴 저항력 모델	13
3.4 운동 제약 조건	14
3.5 항공기 제원	14
3.6 항공기 지상 이동 운동 모델링 결과	15
3.6.1 항공기 지상 이동 운동 모델 검증 - 속도	15
3.6.2 항공기 지상 이동 운동 모델 검증 - 최소 회전 반경	16

4. 항공기 지상 이동 운동 모델 제어기 구성	19
4.1 방위각 제어기 구성	19
4.2 속도 제어기 구성	23
4.3 분리 거리 유지 제어기 구성	28
5. 항공기 분리 유지 알고리즘	31
5.1 항공기 간의 거리 계산	31
5.2 분리 유지 알고리즘	32
5.2.1 항공기 간의 앞, 뒤 구별이 가능	32
5.2.1.1 Case 1: 동일 선상	32
5.2.1.2 Case 2: 다른 항공기가 사용 중인 다음 링크	33
5.2.1.3 Case 3: 활주로 양보	34
5.2.2 항공기 간의 앞, 뒤 구별이 불가능	35
5.2.2.1 Case 4: 서로 다른 링크에서 하나의 노드로 접근	35
5.2.2.2 Case 5: 링크 내에서 마주 오는 항공기	36
5.2.2.3 Case 6: 예외 상황	37
6. 시뮬레이션 결과	38
7. 결론	39
8. 참고문헌	40

표 목 차

[표 1] 속도에 따른 최소 회전 반경	9
[표 2] Wake Turbulence Category별 선정 항공기 제원	15
[표 3] 공력 계산에 필요한 상수	15
[표 4] 조향각과 속도에 따른 항공기의 회전 반경	18
[표 5] 방위각 PID 제어기 이득	20
[표 6] 방위각 입력 값에 따른 Boeing 738 항공기의 방위각 제어 비교	21
[표 7] 방위각 입력 값에 따른 Boeing 773 항공기의 방위각 제어 비교	21
[표 8] 속도 PID 제어기 이득	24
[표 9] 방위각 입력 값에 따른 Boeing 738 항공기의 속도 제어 비교	25
[표 10] 방위각 입력 값에 따른 Boeing 773 항공기의 속도 제어 비교	25
[표 11] DEP001 항공기의 스케줄과 모델의 링크 진출입 시간 비교	26
[표 12] ARR001 항공기의 스케줄과 모델의 링크 진출입 시간 비교	27
[표 13] 분리 거리 유지 PID 제어기 이득	28
[표 14] 항공기 정지 기능 비활성화 시 분리 거리 유지 실패 횟수	38
[표 15] 정지 알고리즘 별 지연 항공기 대수 및 평균 지연 시간	38

그림 목 차

[그림 1] 노드 모델 파일 예시	3
[그림 2] 링크 모델 파일 예시	4
[그림 3] 스케줄 파일 예시	4
[그림 4] 구동중인 프로그램의 화면	5
[그림 5] 시뮬레이션 구성 흐름도	6
[그림 6] 항공기 상태 업데이트 과정	7
[그림 7] 항공기 노드 도착 판정 기준	8
[그림 8] 속도에 따른 최소 회전 반경과 다항식 곡선 피팅 결과	9
[그림 9] 노드-링크 구조로 표현된 항공기 스케줄 예시	10
[그림 10] 항공기의 자유물체도 (수직 방향 힘 성분)	12
[그림 11] 항공기의 자유물체도 (수평 방향 힘 성분)	12
[그림 12] 최대 추력 입력 시 항공기의 속도 변화 그래프	16
[그림 13] 항공기 최소 회전 반경	16
[그림 14] 앞바퀴 조향각 명령 값 계산	19
[그림 15] 항공기 방위각 제어기 블록선도	20
[그림 16] 항공기의 스케줄 경로와 실제 궤적 비교	22
[그림 17] 속도 명령 계산	23
[그림 18] 항공기 속도 제어기 블록선도	23
[그림 19] 분리 거리 유지 제어기 블록선도	28
[그림 20] 분리 거리 유지 제어를 통한 Boeing 738의 속도 제어 및 분리 거리 변화	29
[그림 21] 분리 거리 유지 제어를 통한 Boeing 773의 속도 제어 및 분리 거리 변화	30
[그림 22] 동일 링크내의 항공기 거리 계산	31

[그림 23] 이웃한 링크의 항공기의 거리 계산	31
[그림 24] 항공기 분리 거리 유지 실패 Case 1-1	32
[그림 25] 항공기 분리 거리 유지 실패 Case 1-2	33
[그림 26] 항공기 분리 거리 유지 실패 Case 2	33
[그림 27] 항공기 분리 거리 유지 실패 Case 3	34
[그림 28] 항공기 분리 거리 유지 실패 Case 4	35
[그림 29] 항공기 분리 거리 유지 실패 Case 5	36
[그림 30] 복합적 상황에서의 예외 처리	37
[그림 31] 시나리오에서 발견된 Case 1-2	39
[그림 32] 시나리오에서 발견된 Case 2	40
[그림 33] 시나리오에서 발견된 Case 4	41



초 록

본 논문에서는 First-Time First-Served (FCFS) 스케줄러로부터 도출된 결과를 검증하기 위한 항공기 지상 이동 Fast-Time 시뮬레이터를 개발하였다. 시뮬레이터는 FCFS 스케줄러로부터 생성된 스케줄을 사용하여 항공기를 지상 이동시키는데, 이 결과는 링크 내에서 일정한 속도로 이동한다고 가정한 1차원 질점 항공기 운동 모델을 통해 도출되었다. 실제 항공기에 적용 가능 여부를 확인하기 위해 실제 항공기와 유사한 운동 특성을 보이는 2차원 질점 항공기 지상 이동 운동 모델을 개발 및 검증하였다.

항공기가 경로상의 링크를 자동으로 따라가기 위해 방위각 제어기와 속도 제어기의 구성을 설명하였다. 여러 방위각 명령 값을 입력하여 제어기의 요구 성능 만족 여부를 확인하여 방위각 제어기의 성능을 검증하였다. 또한 여러 속도 명령 값을 입력하여 제어기의 요구 성능 만족 여부를 확인하여 속도 제어기의 성능을 검증하였다. 최종적으로 항공기가 택시 및 유도로를 이탈하지 않으며 예정된 시간에 링크를 진출입하는 것을 확인하여 두 제어기를 통합하여 검증하였다.

앞 항공기와의 분리 거리를 유지하기 위해 필요에 따라 항공기의 속도를 인위적으로 늦춰야하는 상황이 발생하며 이때 필요한 분리 거리 유지 제어기를 구성 및 성능을 검증하였다.

이동 중인 항공기의 충돌 위험이 발생하는 상황에 대해 분석하였다, 안전 분리 거리 기준을 도입하여 충돌 감지 및 회피 알고리즘을 구현하여 분리 거리를 유지하고 교차로에서 항공기의 교착 상태를 방지하였다. 항공기간의 분리 거리가 기준을 위반하지 않기 위한 알고리즘을 개발하였다.

인천국제공항의 실제 운용상황을 모사한, 72대의 항공기가 포함된 시나리오에 대하여 테스트를 실시하였다. 충돌 감지 및 회피 기능을 사용하지 않은 경우, 다양한 위험 상황이 확인되었으며, 충돌 감지 및 회피 알고리즘을 사용하면 추가적인 지연이 발생함을 확인하였다. 또한 회피 알고리즘에서 3가지 통행 우선순위 부여 방식을 구현하여 이를 비교하였다.

Abstract

In this paper, Fast-Time simulator is developed to verify the results derived from the First-Time First-Served (FCFS) scheduler. The simulator uses schedules generated from the FCFS scheduler to move the aircraft on the ground, and this result is derived from one-dimensional mass point aircraft dynamic model assuming a constant velocity within the link. In order to confirm the applicability to actual aircraft, a two-dimensional mass point aircraft dynamic model showing similar dynamic characteristics to actual aircraft is developed and verified.

The bearing controller and the speed controller has been described to make the aircraft automatically follow the links on the path. The performance of the controllers are verified by confirming whether controller has meet the required performance with various command values. Finally, the two controllers were integrated and verified by confirming that the aircraft did not leave the taxiway, and would enter and exit the link at the scheduled time.

In order to maintain the separation distance between aircraft, the speed of the aircraft need to be artificially slowed down if necessary.

The collision detection and resolution algorithm was implemented by introducing the safety separation distance standard to maintain the separation distance and prevent deadlock at the intersection. Algorithm to prevent aircraft from violating the separation distance are discussed.

Tests were conducted on scenarios that included 72 airplanes at the Incheon International Airport. When collision detection and resolution functions were not used, various risk situations were identified, and additional collisions were detected using collision detection and avoidance algorithms. In addition, three different prioritization schemes are implemented in resolution algorithm and results are compared.

1. 서론

1.1 연구 배경

매년 항공 교통량이 크게 증가하는 추세이며, 많은 공항이 용량의 한계에 봉착하고 있다. 수요를 만족시키기 위해 공항의 용량이 증가되어야 하는데 기존 시설을 그대로 이용할 경우 공항 내에서 항공기의 움직임이 복잡해지고, 출도착 지연이 빈번하게 발생할 수 있다. 이러한 문제점의 해결을 위해 출도착 통합 관리 시스템에 대한 연구 [1],[2]와, 이에 필요한 효율적인 항공기 스케줄링 알고리즘에 대한 연구 개발이 진행되고 있다[2],[3].

출도착 관리 시스템의 연구 개발 과정에서 항공기 지상 이동 시뮬레이터의 개발이 필수적이다. 일반적으로 스케줄링 알고리즘은 상대적으로 단순한 항공기 모델을 이용하여 스케줄링 결과를 도출하게 되는데, 택시 및 유도로 내에서의 추월 등 비 정상적인 상황을 확인할 수 있는 지상 이동 시뮬레이터를 통한 검증이 필요하다. 또한 지상 이동 시뮬레이터를 통해 항공기의 지상 궤적을 예측하고 이를 다시 스케줄링에 반영할 수도 있다. 미국 NASA에서는 Spot and Runway Departure Advisor (SARDA)라는 출발 관리 시스템을 개발하여[1] 미국 내의 몇몇 주요 공항에서 테스트를 진행 중이며, 이를 위해 필요한 fast-time 시뮬레이터인 Surface Operations Simulator and Scheduler (SOSS)[4]를 개발하여 사용하고 있다.

1.2 연구 목표 및 진행 방향

1.2.1 연구 목표

본 논문에서는 기존의 First-Come First-Served (FCFS) 스케줄러와 연동이 가능한 fast-time 시뮬레이터를 개발한다. 시뮬레이터를 통해 스케줄을 시각적 방법과 수치적 방법으로 검증을 할 수 있으며, FCFS 스케줄러에 의해 생성된 스케줄이 정상적으로 작동하지 않을 시에도 항공기의 충돌을 방지하기 위한 알고리즘을 적용하여 충돌 위험을 제거하여 새로운 스케줄을 생성할 수 있다. 교차로에서 통행 우선순위를 부여하는 3가지 알고리즘을 제시하였으며 각 알고리즘을 적용하였을 때의 지연 항공기 대수와 평균 지연 시간을 비교한다.

1.2.2 연구 진행 방향

기존의 스케줄링 결과는 안전 분리 시간 개념을 도입하여 충돌을 방지한다. 실제 공항에서는 관제사와 조종사의 판단 하에 항공기 간의 적정 분리 거리를 유지한다. 항공기 간의 분리 거리 유지의 목적은 항공기 간의 충돌 위험 방지 및 앞선 항공기의 엔진의 후류로 인한 동체 손상 방지이다. 따라서 본 논문에서 소개하는 배속 시뮬레이터는 거리 개념을 도입하였다.

본 논문에서는 실제 항공기와 유사한 운동 특성을 가진 2차원 질점 항공기 운동 모델을 개발하여 단순한 항공기 운동 모델로 도출된 스케줄링 결과의 유효성 여부를 판단할 수 있는 도구를 개발한다. 항공기가 지상 이동 중 택시 및 유도로를 이탈하지 않으며, 스케줄의 링크 진출입 시간을 준수할 수 있도록 방위각 제어기와 속도 제어기의 개발한다.

FCFS 스케줄러에 적용된 확정적 알고리즘의 특성 상 항공기의 지연 및 항공기 모델의 한계로 인해 링크 진출입 시간에 작은 오차가 발생할 경우 정상적으로 작동하지 않게 되어 항공기 간의 분리 거리가 의도했던 것보다 가까워 질 수 있으며 심각한 경우 항공기 충돌로 이어질 수 있다. 이러한 상황을 방지하고자 항공기 간의 전후 분리 간격을 유지하고, 교차로에서의 통행 우선순위를 부여하는 알고리즘을 개발하여, 이를 위해 필요한 분리 거리 유지 제어기를 추가 구성한다.

1.3 논문의 구성

II 장에서는 전반적인 시뮬레이터의 구성에 대하여 설명하고, III 장에서는 FCFS 스케줄러에 적용된 항공기 운동 모델의 한계와 본 연구에서 개발한 2차원 질점 항공기 운동 모델을 설명하고, IV 장에서 항공기 운동 모델의 자동 운행에 필요한 3가지 제어기의 구성을 설명한다. V 장에서는 항공기 분리 유지 알고리즘을 설명한다. VI 장에서 시뮬레이션 구성을 설명하고 테스트 결과를 종합하고, VII 장에서 결론을 제시하였다.

2. 시뮬레이터 구성

2.1 입력 데이터

시뮬레이터를 구동하기 위해 공항의 노드-링크 모델과 스케줄 파일이 필요하다. 노드는 공항의 유도로 및 활주로의 교차점과 계류장을 의미하며, 링크는 두 개의 노드로 구성되어 유도로 및 활주로의 일부를 의미한다.

2.1.1 노드

* Nodes in RKS1			
* Type	ID	Latitude	Longitude
PUSHBACK	10001	37.4521	126.4567
TAXI	10002	37.4528	126.4561
TAXI	10003	37.4544	126.4548
PUSHBACK	10004	37.4523	126.4541
DEICE_PAD	10156	37.4854	126.4434
DEICE_PAD	10157	37.4859	126.4444
GATE	20000	37.4496	126.456
GATE	20001	37.4497	126.4565
GATE	20002	37.45	126.4568
RWY15R/33L	20576	37.4811	126.4369
RWY15R/33L	20577	37.4804	126.4375
RWY15R/33L	20578	37.4556	126.4596

그림 1 노드 모델 파일 예시

그림 1은 시뮬레이터 구동에 필요한 파일 중 하나인 노드 모델 파일의 예시로, 각 노드는 Type와 ID, 위경도 좌표로 구성되어 있으며, 공항의 모든 노드 정보는 각 열마다 나열되어 있다.

2.1.2 링크

* Links in RKS1			
* Type	ID	Node1	Node2
GATE	gate1	20000	20186
GATE	gate2	20001	20186
GATE	gate3	20002	20189
RAMP	Rp1	20186	20189
RAMP	Rp2	20189	10195
RAMP	Rp3	10195	10196
TAXI	Tx1	20591	10105
TAXI	Tx2	10105	10106
TAXI	Tx3	10106	20600
RWY15L/33R	Rwy1	20580	20610
RWY15L/33R	Rwy2	20610	20590
RWY15L/33R	Rwy3	20590	20591

그림 2 링크 모델 파일 예시

그림 2는 링크 모델 파일의 예시로, 각 링크는 Type, ID와 구성 노드로 이루어져 있으며, 각 열마다 공항의 링크 정보가 포함되어 있다.

2.1.3 Schedule

* Flight ID	Address	Type	Wake Category	State	Entry Time	Exit Time	Transit Time	Previous Link	Current Link	Next Link	Speed-Up	Slow-Down
ARR002	HL0002	B772	H	ARR	2208	2220	12	XXXX	20581	Rwy7	0	0
ARR002	HL0002	B772	H	ARR	2220	2232	12	Rwy7	Rwy6	Rwy5	0	0
ARR002	HL0002	B772	H	ARR	2232	2240	8	Rwy6	Rwy5	Rwy4	0	0
⋮												
ARR002	HL0002	B772	H	ARR	2661	2678	17	Rp124	Rp77	Rp78	0	0
ARR002	HL0002	B772	H	ARR	2678	2692	14	Rp77	Rp78	gate64	0	0
ARR002	HL0002	B772	H	ARR	2692	2735	43	gate64	20062	XXXX	0	0

그림 3 스케줄 파일 예시

그림 3은 FCFS 스케줄러에 의해 도출된 스케줄링 결과의 일부분이다. 항공기의 편명과 출발, 도착 노드가 있으며 경로상의 경유 링크가 순서대로 나열되어 있다. 또한 각 링크별 진출입 시간이 있다.

2.2 사용자 인터페이스 화면 구성

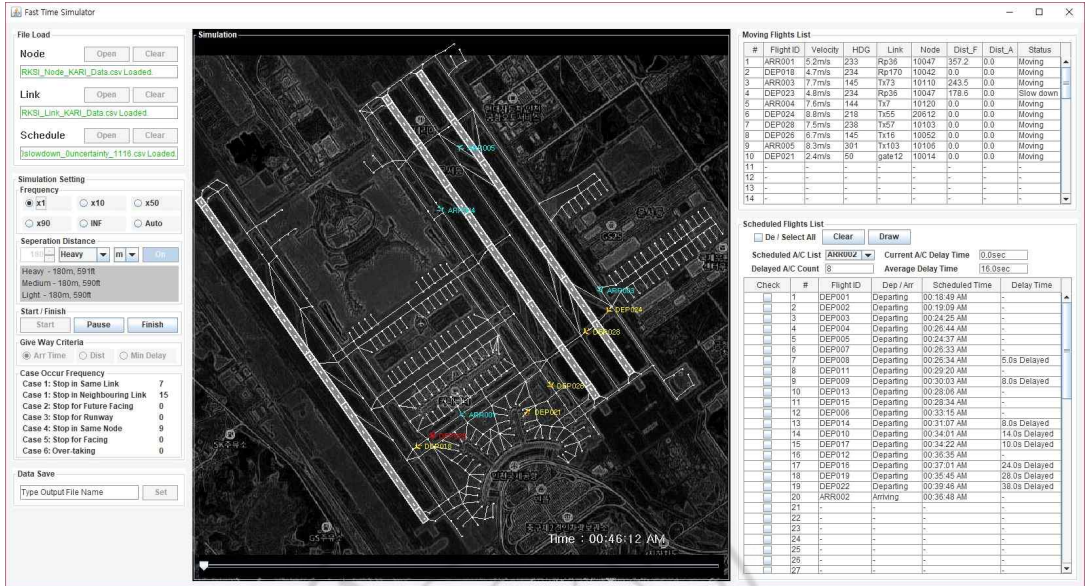


그림 4 구동중인 프로그램의 화면

그림 4는 본 논문의 연구를 위해 개발한 프로그램의 화면이다. 시뮬레이터의 가운데 창에서 공항의 노드-링크 모델과, 현재 이동 중인 항공기를 시각적으로 확인할 수 있다. 노란색과 시안 색을 사용하여 출발 항공기와 도착 항공기의 구별하였으며, 주변 항공기와 거리가 가까워져 정지가 필요한 항공기는 빨간색으로 표시하였다.

화면의 좌측에는 입력파일을 불러올 수 있는 창과 시뮬레이션 설정 창이 구성되어 있어 배속과 항공기 분리 거리 설정, 교차로에서 항공기 통과 순서 등을 지정할 수 있다. 화면의 오른쪽 위에서는 현재 이동 중인 항공기들의 상태 정보를 확인할 수 있으며, 현재 링크, 주변 항공기와의 거리 등을 파악할 수 있다. 화면의 오른쪽 아래에서는 이륙 또는 게이트에 도착한 항공기들의 리스트와, 입력 스케줄 대비 지연 시간을 확인할 수 있다.

2.3 시뮬레이션 흐름도

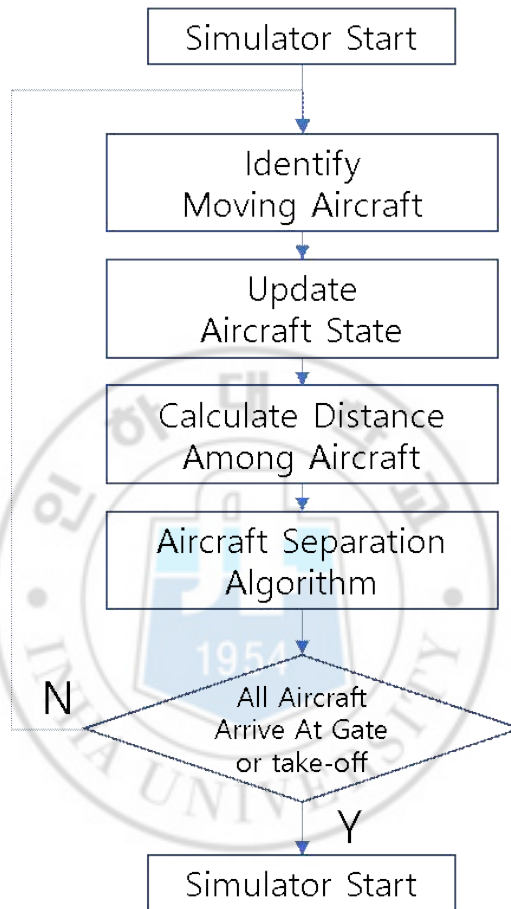


그림 5 시뮬레이션 구성 흐름도

위의 그림5에서 시뮬레이션의 구성을 확인할 수 있다. 시뮬레이션 시작 이후, 매 타임스텝에서 이동 중인 항공기를 파악한다. 이동 중인 항공기들의 상태 변수를 계산하여 업데이트하게 되며 그 구성은 아래의 그림에서 확인할 수 있다. 항공기의 위치 업데이트를 통해 항공기간의 거리 계산과 충돌 방지를 위해 정지 혹은 속도 제어가 필요한 항공기를 구별하게 된다. 이를 다음 타임스텝에서 고려하여 상태를 업데이트하게 된다. 이러한 일련의 과정을 모든 항공기의 시뮬레이션이 완료되는 순간까지 반복적으로 진행

하게 된다.

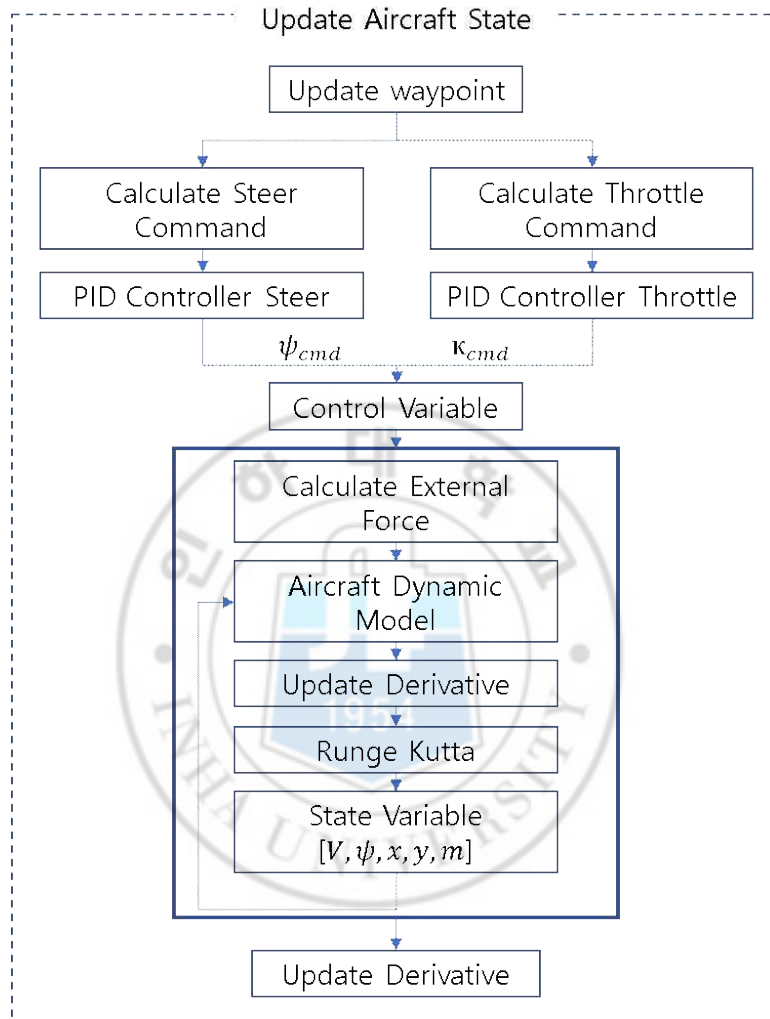


그림 6 항공기 상태 업데이트 과정

위의 그림에서는 항공기 상태를 업데이트하는 과정을 확인할 수 있다. 매 타임스텝마다 가장 먼저 항공기의 목적 노드의 변경 여부를 확인하여 $Node_{target}$ 을 업데이트 한다. 이후 $Node_{target}$ 의 위치와 항공기의 위치, $Node_{target}$ 에 도착 예정 시간과 현재 시간 정보를 사용하여 제어를 통해 조향각 명령 값과 엔진 추력 명령 값을 계산하게 된다. 주변의 항공기로 인해 정지 혹은 감속이 필요한 경우 속도 제어기에 의해 산출된 엔진

추력 명령 값을 사용하지 않고 분리 거리 유지 제어기에 의해 산출된 엔진 추력 명령 값을 사용한다. 두 가지 입력변수를 항공기의 모델에 입력하여 상태 변수의 미분 값을 계산한 후, Runge Kutta 적분을 통해 현재 시간에서의 상태 정보를 산출하게 된다. 이후 계산된 상태 정보 값을 저장하여 업데이트 과정을 마치게 된다.

항공기 상태 업데이트 과정의 초기 단계에서 $Node_{target}$ 을 업데이트는 다음과 같다.

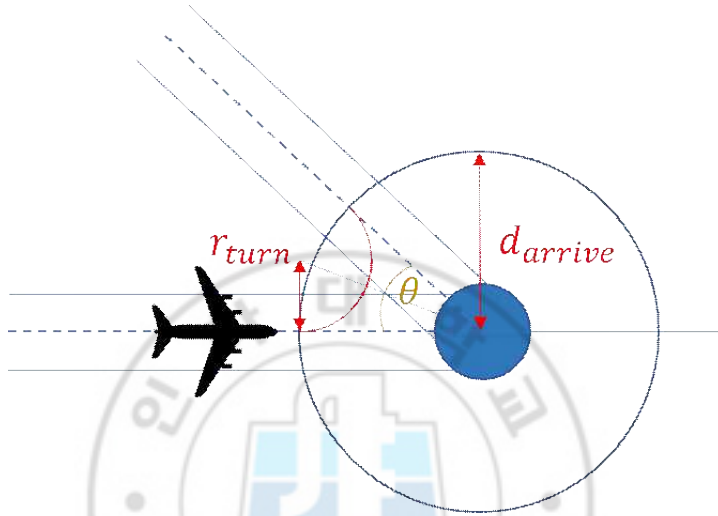


그림 7 항공기 노드 도착 판정 기준

항공기가 노드의 정확한 위치에 도착하기 어려울 뿐만 아니라 항공기의 경로 및 이동 시간을 줄이기 위해 항공기가 노드로부터 특정 거리 이내로 근접하면 도착한 것으로 간주하고 $Node_{target}$ 을 업데이트해야한다. 위의 그림은 항공기가 노드에 도착한 것으로 파악하게 되는 기준인 d_{arrive} 와 최소 회전 반경인 r_{turn} 과의 관계를 확인할 수 있다.

앞서 항공기 지상 이동 모델 구성 후 산출된 최소 회전 반경을 이용하여 아래의 식으로부터 d_{arrive} 를 계산한다.

$$\tan\left(\frac{\theta}{2}\right) = \frac{r_{turn}}{d_{arrive}} \rightarrow d_{arrive} = \frac{r_{turn}}{\tan\left(\frac{\theta}{2}\right)} \quad (2.1)$$

위의 식에서 사용된 최소 회전 반경은 항공기의 속도가 15kt일 때 계산된 값으로, 항공기의 속도가 다를 경우 최소 회전 반경이 달라질 수 있다. 항공기의 속도가 5kt, 10 kt, 15 kt, 20 kt일 때 최소 회전 반경을 계산한 결과 아래의 표로 정리할 수 있다.

표 1 속도에 따른 최소 회전 반경

속도	최소 회전 반경
5 kt	22 m
10 kt	32 m
15 kt	49 m
20 kt	75 m

위의 결과 값으로부터 다항식 곡선 피팅을 통해 속도에 대한 최소 회전 반경의 2차 함수를 구하였으며 그 식은 다음 식 (2.2)과 같다.

$$r_{turn} = (0.16 * velocity^2 - 0.48 * velocity + 20.5) \quad (2.2)$$

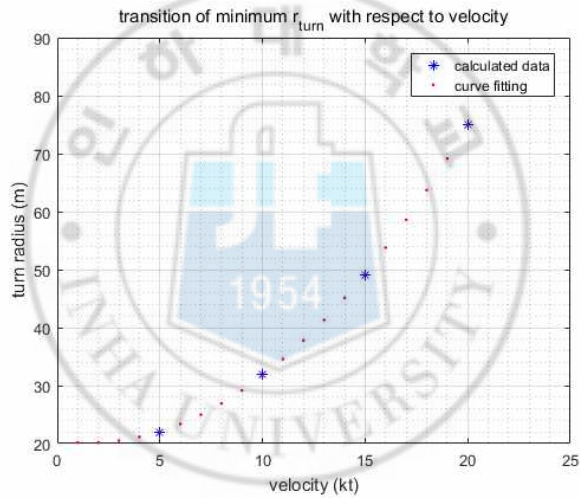


그림 8 속도에 따른 최소 회전 반경과 다항식 곡선 피팅 결과 그래프

위의 그림으로부터 식 3.2을 통해 계산되는 최소 회전 반경의 값이 유효함을 확인할 수 있다.

위의 결과로부터 항공기의 속도를 고려한 d_{arrive} 을 구하기 위한 식을 다음과 같이 적용하여 계산하였다.

$$d_{arrive} = \frac{(0.16 * velocity^2 - 0.48 * velocity + 20.5)}{\tan(\frac{\theta}{2})} \quad (2.3)$$

3. 항공기 운동 모델

3.1 FCFS 스케줄러 항공기 지상 이동 운동 모델

본 시뮬레이터의 입력파일 중 하나인 FCFS 스케줄러에 의해 도출된 결과는 비교적 단순한 1차원 질점 항공기 운동 모델을 통해 링크 진출입 시간이 계산되었다.

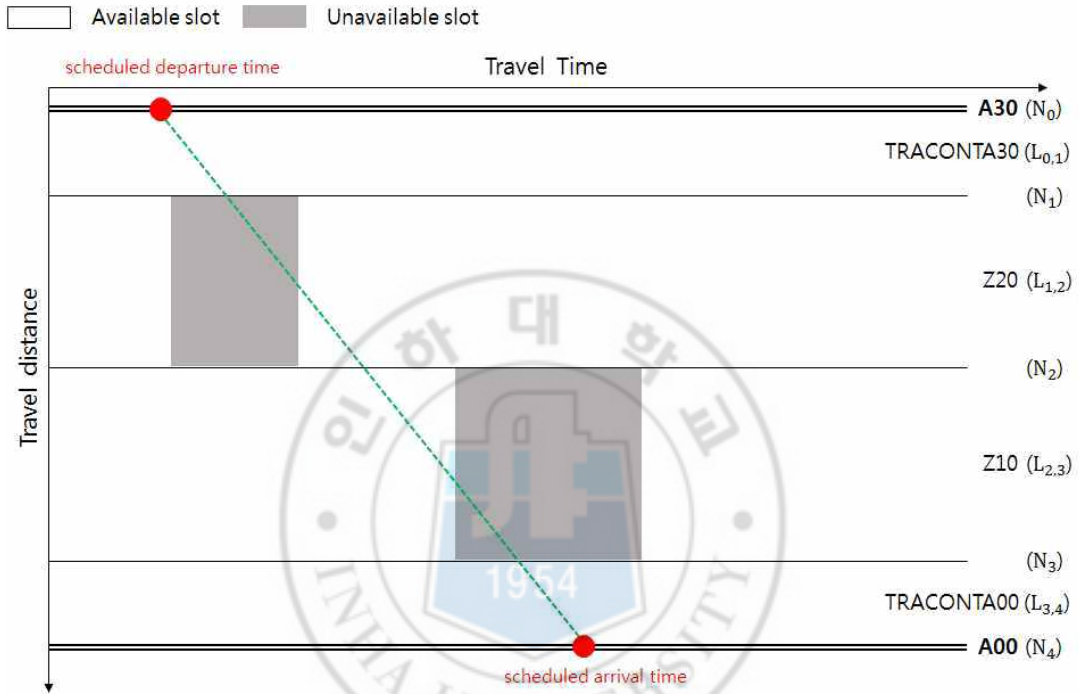


그림 9 노드-링크 구조로 표현된 항공기 스케줄 예시

위의 그림 9는 FCFS 스케줄링 알고리즘에 의해 생성된 항공기 스케줄 예시의 일부이며 스케줄링 알고리즘에 적용되어 있는 항공기 운동 모델을 볼 수 있다. 그림의 가로축은 시간을 의미하여 세로축은 이동거리를 의미한다. 우측의 N_i 는 항공기 경로상의 노드가 순서대로 나열되어 있으며, $L_{i,j}$ 는 N_i 와 N_j 로 구성된 링크를 의미한다. 여기서 초록색 점선의 기울기는 거리와 시간의 비로 항공기의 속도를 의미한다. 항공기의 속도는 링크 내에서 일정하다는 가정 하에 계산되었으며 속도 변화는 노드에서만 계단 함수의 형태로 이루어진다. 항공기 경로상의 이웃한 두 링크의 사잇각에 따른 추가 이동 거리에 의한 시간 증가는 고려되지 않았으며 사잇각에 따라 일정 시간을 추가하는 방식으로 계산되었다.

실제에는 항공기의 속도는 매 순간 변화 가능하며 궤적 또한 링크의 중심선에서 벗어날 수 있다. 따라서 실제 항공기가 스케줄러에 의해 생성된 시간 계획을 따라가지 못할 수 있다. 또한 FCFS 스케줄러에서는 항공기가 링크 중심선을 따라서 이동한다고 가정하였으므로 스케줄의 속도를 유지하여 회전할 때 택시 및 유도로를 이탈하지 않고 이동 가능 여부는 검증되어있지 않다. FCFS 스케줄러의 1차원 항공기 운동 모델을 통해 생성된 스케줄을 실제 항공기에 적용 가능 여부 판단 및 검증을 위해 실제 항공기의 이동을 모사할 수 있는 항공기 운동 모델 개발이 필요하다.

3.2 2차원 질점 항공기 지상 이동 운동 모델

항공기 운동 모델을 단순화하기 위해 구면을 이루고 있는 공항의 지상 영역을 람베르트 정각원추도법을 통해 평면으로 사영하였으며, 모든 지점의 고도가 동일하다는 가정하에 2차원 항공기 운동 모델을 구성하였다.

항공기 지상 이동 모델의 운동 방정식 구성을 위해 2개의 좌표계 정의가 필요하다. 첫 번째 좌표계는 관성좌표계이며, 공항의 노드-링크 모델과 항공기의 위치를 표현하기 위한 좌표계이다. 람베르트 정각원추도법에서 사용된 원점을 좌표계의 중심으로 하며, 좌표축은 단위벡터 $(\hat{I}, \hat{J}, \hat{K})$ 와 같다. \hat{I} 축은 원점을 기준으로 동쪽을 향하는 방향이며, \hat{J} 축은 원점을 기준으로 북쪽을 향하는 방향이고 \hat{K} 축은 \hat{I} 축과 \hat{J} 축 벡터의 오른손 법칙으로 방향을 결정하여 지상 위쪽을 향하는 방향이다. 두 번째 좌표계는 항공기 동체 좌표계로 항공기에 작용하는 모든 힘을 고려하여 속도와 방위각을 계산하기 위해 사용되는 좌표계이다. 동체 좌표계는 항공기의 무게 중심을 중심으로 하며, 좌표축은 $(\hat{i}_{ac}, \hat{j}_{ac}, \hat{k}_{ac})$ 으로 구성되어 있다. \hat{i}_{ac} 축은 항공기의 무게 중심을 기준으로 항공기의 기수 방향으로의 동체의 종축과 같으며, \hat{j}_{ac} 축은 항공기의 무게 중심을 기준으로 \hat{i}_{ac} 에 수직하며 왼쪽 날개 끝을 향하는 방향이고 \hat{k}_{ac} 축은 \hat{i}_{ac} 축과 \hat{j}_{ac} 축의 벡터의 오른손 법칙으로 방향을 결정하였다.

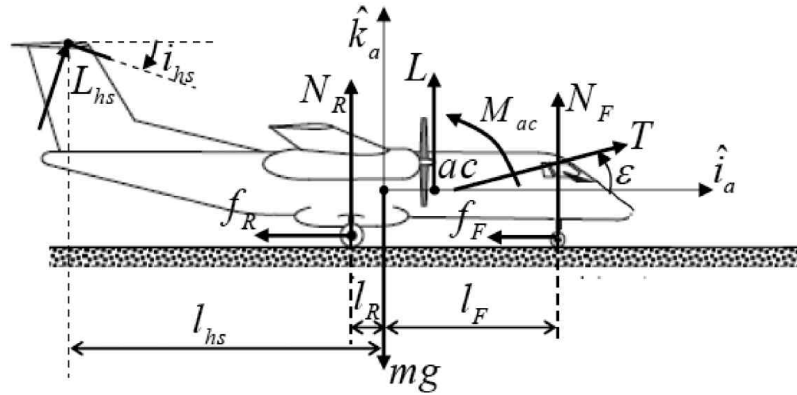


그림 10 항공기의 자유물체도 (수직 방향 힘 성분)

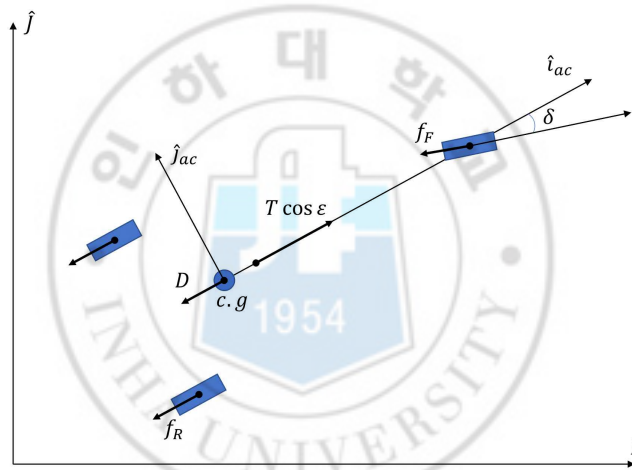


그림 11 항공기의 자유물체도 (수평 방향 힘 성분)

위의 그림10과 그림11에서는 항공기의 수직과 수평 방향의 자유 물체도를 볼 수 있다.

본 논문에서는 항공기는 앞바퀴로만 조향이 이루어진다고 가정하였다. 또한 낮은 속도에서는 동압이 낮아 꼬리날개에서 발생하는 양력의 영향이 미미하고, 힘 성분에 외부 바람의 영향을 고려하지 않았으므로 동압이 높은 속도에서는 방향키의 조작이 불필요하여 꼬리날개에서 발생하는 양력을 무시할 수 있다. 따라서 꼬리날개에서 발생하는 양력에 의한 모멘트는 없다고 가정하였으며, 꼬리날개에서 발생하는 양력은 전체 항력에 포함하여 계산하였다. 이와 같은 가정을 통해 항공기 지상 이동 모델의 운동 방정식은 아

래의 식 (3.1)–(3.4)와 같이 구성할 수 있다.

$$m\dot{V} = T \cos \epsilon - D - f_R - f_F \cos \delta - F_{brake} \quad (3.1)$$

$$mV\dot{\Psi} = -f_F \sin \delta \quad (3.2)$$

$$\dot{x} = V \sin \Psi \quad (3.3)$$

$$\dot{y} = V \cos \Psi \quad (3.4)$$

위의 식 (3.1)에서 D 는 항공기 동체에 작용하는 항력이며 항공기 동체에 작용하는 양력과 항력의 식은 각각 다음 식 (3.5)–(3.6)과 같다.

$$L = \frac{1}{2} \rho V^2 S C_L \quad (3.5)$$

$$D = \frac{1}{2} \rho V^2 S C_D \quad (3.6)$$

모델의 입력변수는 매순간 현재 상태변수로부터 계산되며 상태변수는 변화하는 입력 변수에 따라 결정된다. 항공기 지상 이동 모델의 입력변수는 추력과 앞바퀴 조향각 $[T, \delta]$ 이며, 상태변수는 속도와 방위각, 위치, 무게 $[V, \psi, x, y, m]$ 이다.

추력 설정 값은 입력 변수 추력을 계산하기 위해 사용되며 아래의 식 (3.7)과 같이 정의된다.

$$\kappa = \frac{T - T_{\min}}{T_{\max} - T_{\min}} \quad (3.7)$$

항공기가 지상에서 이동 중 연료 소모로 인해 무게가 변화하는 것을 고려하기 위해 연료 소모율을 고려하였으며 아래의 식 (3.8)을 적용하였다.

$$\dot{m} = -\dot{m}_{fuel} = -TSFC * Thrust \quad (3.8)$$

3.3 바퀴 저항력 모델

항공기에 작용하는 총 힘을 계산하기 위해 바퀴에 작용하는 힘을 고려하였으며 그 크기는 마찰계수와 수직항력의 곱과 같고, 방향은 바퀴의 진행방향과 반대이다. 항공기의

앞, 뒷바퀴의 수직항력과 무게는 동일하며 무게 중심에서의 키놀이 모멘트가 0이므로 앞, 뒷바퀴의 수직항력은 다음 식 (3.9)–(3.10)과 같다.

$$N_F = (mg - T \sin \epsilon - L) * \frac{l_R}{l_F + l_R} \quad (3.9)$$

$$N_R = (mg - T \sin \epsilon - L) * \frac{l_F}{l_F + l_R} \quad (3.10)$$

각 바퀴에 작용하는 저항력은 다음 식 (3.11)–(3.12)과 같으며 C_f 는 종축 마찰계수이다.

$$f_F = C_{f_F} N_F \quad (3.11)$$

$$f_R = C_{f_R} N_R \quad (3.12)$$

항공기의 지상 이동 속도가 빠르지 않다는 점을 감안하여 앞, 뒷바퀴 모두 미끄럼 각은 발생하지 않는다고 가정하였다.

3.4 운동 제약 조건

실제 공항에서 운용중인 항공기의 기동을 모사하기 위해 운동과 관련된 제약 조건이 필요하다. 가장 먼저 입력 변수에 대한 제약 조건이 필요하다. 앞바퀴의 조향각(δ)과 엔진 추력 설정(κ)의 제약 조건은 아래와 같으며 $\delta_{\max} = 50^\circ$ 이고 T_{\max} 는 항공기 엔진의 제원상의 최대 추력이다.

$$|\delta| \leq \delta_{\max} \quad (3.13)$$

$$0 \leq \kappa \leq 0.15 * T_{\max} \quad (3.14)$$

유도로와 및 유도로에서의 최대 이동 속도 $V_{\max} = 30kt$ 이며 속도의 제약 조건은 아래와 같다.

$$0 \leq V \leq V_{\max} \quad (3.15)$$

위의 엔진 추력과 속도에 대한 제약조건은 항공기가 택시 및 유도로에서 이동 중에만 적용되며 활주로에서는 제한이 없다.

3.5 항공기 제원

시뮬레이터에는 2가지 Wake Turbulence Category별 분리 거리를 입력할 수 있도록 되어 있으며, 항공기의 제원 차이에 따라 운동 특성 등이 다르므로 각 Wake Turbulence Category별 항공기를 설정하였으며 그 제원은 아래와 같다.

표 2 Wake Turbulence Category별 선정 항공기 제원

Category	Medium	Heavy
기종	Boeing 738	Boeing 773
최대 이륙 중량	79000 kg	299370 kg
항공기 엔진의 최대 추력	240 kN	880 kN
날개 면적	125 m ²	428 m ²
무게중심부터 노즈 기어까지의 거리	14 m	27.35 m
무게중심부터 메인 기어까지의 거리	1.6m	3.87m

항공기 공력 계산을 위한 상수는 아래의 표에 정리되어 있다.

표 3 공력 계산에 필요한 상수

공기 밀도	1.225 kg/m ³
유해 항력 계수, C_{D_0}	0.0123
유도 항력 계수, K	0.0435
양력 계수, C_L	1

항공기 전체에 작용하는 항력을 계산하기 위한 항력계수는 아래의 식을 통해 계산하였다.

$$C_D = C_{D_0} + KC_L^2 \quad (3.16)$$

3.6 항공기 지상 이동 운동 모델링 결과

항공기 이동 모델의 입력 변수는 추력과 앞바퀴 조향각이 있으며, 입력 변수에 따른 항공기의 운동 특성을 통한 모델 검증이 필요하다.

3.6.1 항공기 지상 이동 운동 모델 검증 - 속도

FCFS 스케줄러는 항공기가 활주로를 50초 이내에 탈출하도록 설정되어 있으므로 이

륙하는 항공기 모델에 최대 추력을 입력하였을 때 50초 이내에 이륙하여야 한다.

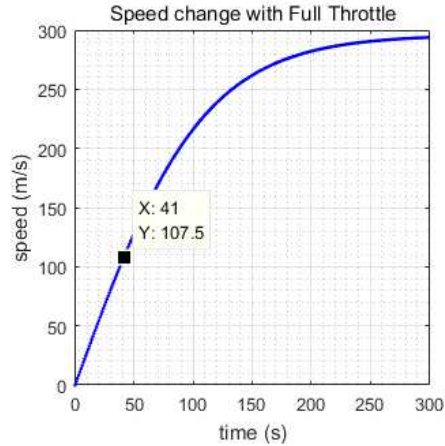


그림 12 최대 추력 입력 시 항공기의 속도 변화

위의 그림은 항공기가 최대 추력으로 이륙 시 속도의 변화를 볼 수 있는 그래프이다. 항공기의 무게보다 양력이 커지는 시점에 이륙을 한 것으로 간주하였으며 그 시점은 41초로 계산되었다. 최대 활주로 점유 시간인 50초 이내에 활주로를 탈출한 것을 확인할 수 있다.

3.6.2 항공기 지상 이동 운동 모델 검증 - 최소 회전 반경

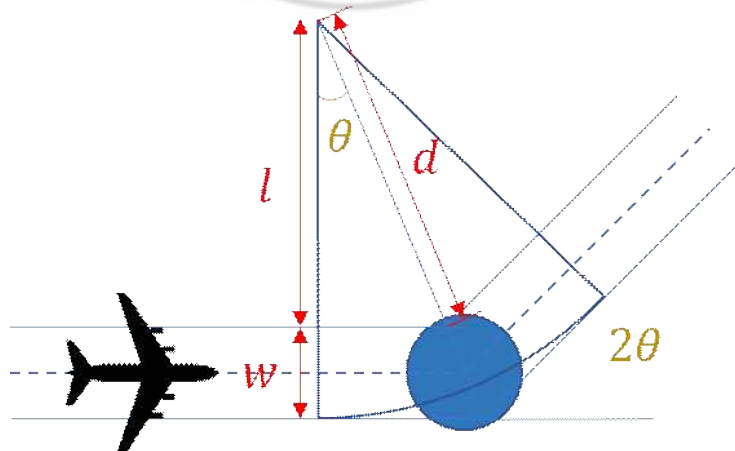


그림 13 항공기 최소 회전 반경

위의 그림은 항공기의 최소 요구 회전 반경을 구하기 위한 그림이다.

항공기의 회전 반경은 아래의 식으로 정의되며 항공기가 택시 및 유도로를 이탈하지 않기 위한 조건은 다음과 같다.

$$r_{turn} = l + w \quad (3.17)$$

$$r > d \quad (3.18)$$

두 링크의 사잇각이 2θ 일 때, l 과 d 의 관계는 다음 식을 통해 구할 수 있다.

$$d = \frac{l}{\cos(\theta)} \quad (3.19)$$

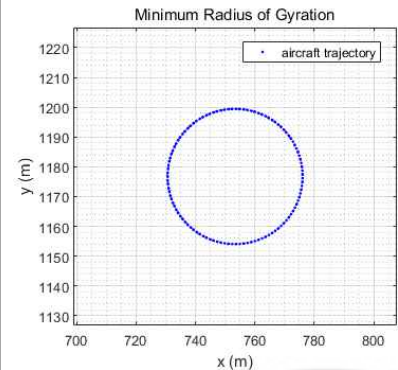
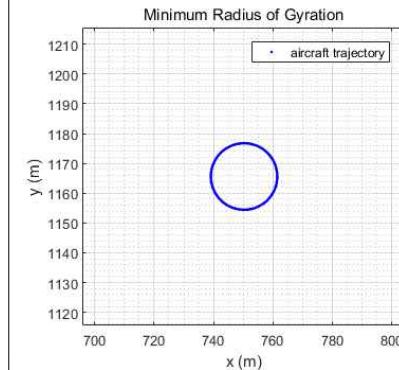
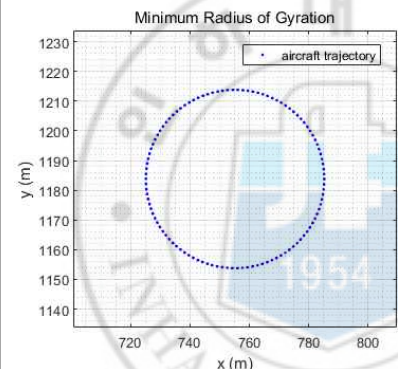
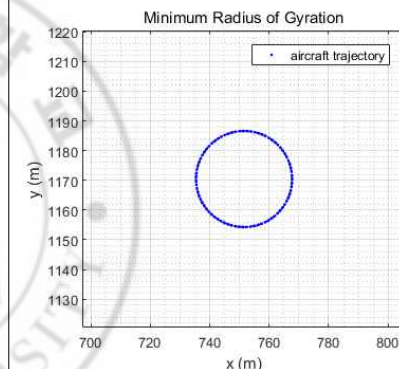
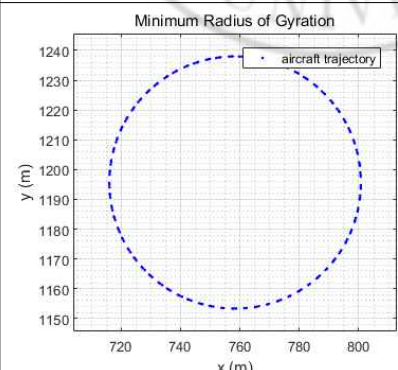
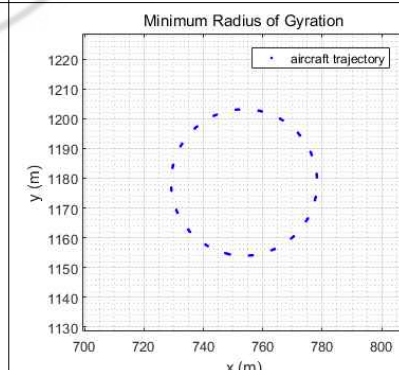
위의 식 (3.17)–(3.19)로부터 다음 식을 유도할 수 있다.

$$r < \frac{w}{1 - \cos(\theta)} \quad (3.20)$$

두 링크의 사잇각의 최댓값은 180° 로 택시 및 유도로의 폭인 $30m$ 를 적용하면 최소 회전 반경은 $30m$ 로 계산된다. 실제 공항의 노드 링크 모델 중 두 링크의 사잇각이 180° 인 경우는 없다.

항공기의 회전 반경은 항공기의 속도와 조향각에 의해 크게 좌우된다. 각 속도와 조향각에 의한 최소 회전 반경을 확인하였다. 속도는 $5kt$, $10kt$, $15kt$ 와 조향각은 30° , 50° 에 대해 계산하였다.

표 4 조향각과 속도에 따른 항공기의 회전 반경

조향각 속도	$\delta = 30^\circ$	$\delta = 50^\circ$
$v_{ac} = 5 kt$	 <p>회전 반경: 45.4 m</p>	 <p>회전 반경: 22.3 m</p>
$v_{ac} = 10 kt$	 <p>회전 반경: 60.0 m</p>	 <p>회전 반경: 32.3 m</p>
$v_{ac} = 15 kt$	 <p>회전 반경: 84.6 m</p>	 <p>회전 반경: 49.0 m</p>

Heavy급 항공기인 Boeing 773의 제원을 사용하여 항공기의 회전 반경을 계산하였으며 위의 표에 정리하였다.

4. 항공기 지상 이동 운동 모델 제어기 구성

실제 항공기는 주어진 경로를 따라가기 위해 매순간 조종사의 판단에 의해 입력변수가 결정된다. 본 시뮬레이터에서는 자동으로 경로를 추종하기 위해 조종사 모델을 대체할 제어기가 필요하다. 항공기가 링크의 중심선으로부터 벗어나지 않으며, 링크를 빠져나온 후 다음 링크로 진입하기 위해 항공기의 기수를 조종해야 한다. 이를 위해 앞바퀴 방위각 제어기를 구성하였다. 또한 스케줄링 결과의 링크 진출입 시간을 준수하기 위해 속도를 맞춰 이동해야하므로 속도 제어기가 필요하다. 본 시뮬레이터의 핵심 기능 중 하나인 전, 후 분리 거리 유지 알고리즘은 두 항공기의 분리 거리가 분리거리 기준보다 가까운 경우 뒤따라오는 항공기를 정지 혹은 속도 제어를 통해 분리거리를 유지한다. 따라서 분리 거리 유지 제어기를 추가적으로 구성하였다.

4.1 방위각 제어기 구성

항공기가 스케줄의 경로를 따라 이동하기 위해 항공기 방위각 제어기 개발이 필요하다. 제어기에 입력되는 방위각 계산 방법은 아래 그림에서 확인할 수 있다.

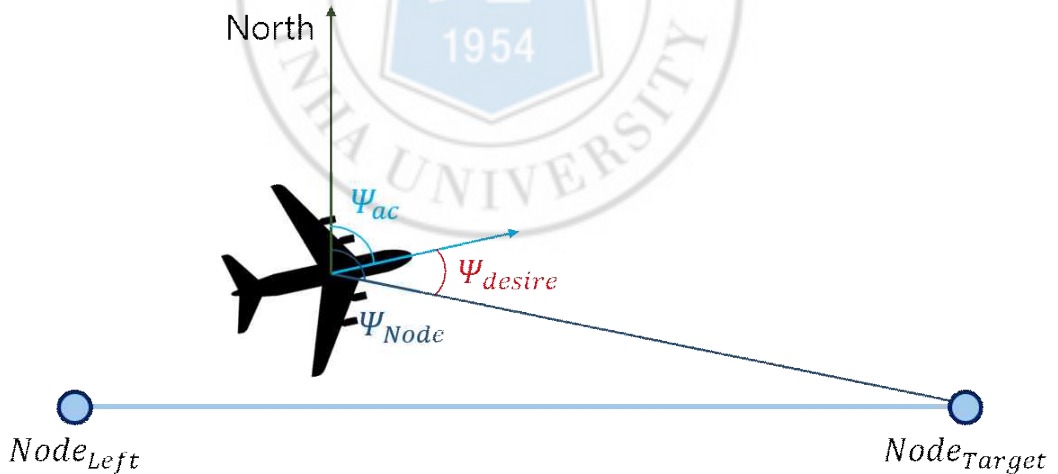


그림 14 앞바퀴 조향각 명령 계산

$$\psi_{Node} = \frac{\pi}{2} - \tan^{-1}\left(\frac{y_{Node} - y_{ac}}{x_{Node} - x_{ac}}\right) \quad (4.1)$$

$$\psi_{desire} = \psi_{Node} - \psi_{ac} \quad (4.2)$$

현재 항공기의 위치와 목적 노드의 위치를 통해 관성 좌표계 기준으로 노드의 방위각을 위의 식 (4.1)을 통해 계산할 수 있다. 노드의 방위각과 항공기의 방위각의 차이를 계산한 후 방위각 제어기에 입력하게 된다.

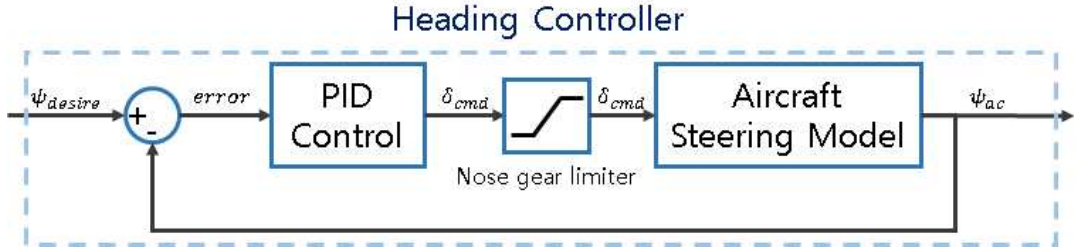


그림 15 항공기 방위각 제어기 블록선도

위의 그림에서는 방위각 제어기의 구성을 확인할 수 있으며, 매순간 식 (4.1)과 식 (4.2)을 통해 계산된 값이 입력된다. PID 제어기를 통해 앞바퀴의 조향각 명령 값 δ_{cmd} 를 계산하게 되고, 리미터를 거친 후 최종 명령 값을 얻게 된다. 앞바퀴 조향각 명령 값을 항공기 모델의 운동 방정식에 입력한 후 현재 시간에서의 방위각을 산출한 후 상태를 업데이트 하게 된다.

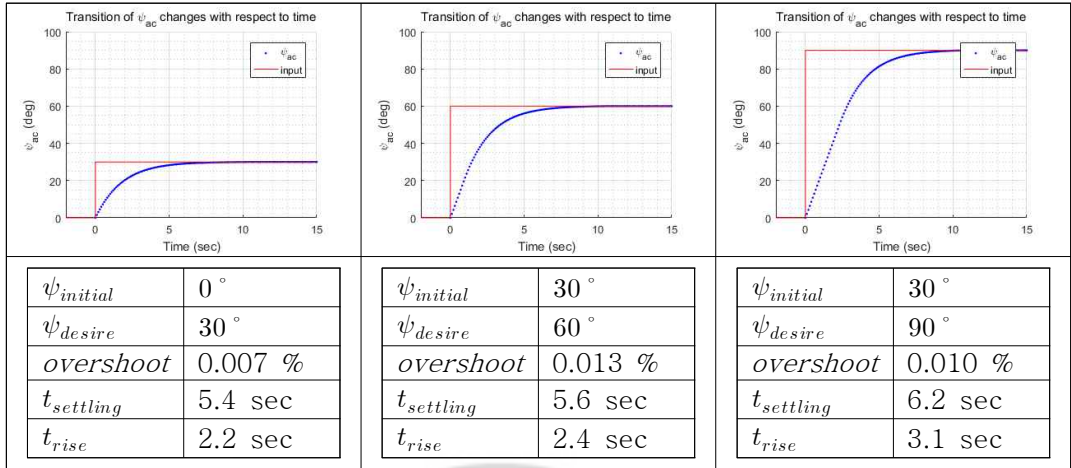
표 5 방위각 PID 제어기의 이득

Gain	Proportional	Integral	Derivative
값	70	0.1	1

실제 공항에서 이동하는 항공기의 경우 다음 링크로 이동할 때 링크 중심선을 기준으로 바깥쪽으로 밀려나는 길이가 미미하다는 점을 고려하여 제어기의 오버슈트 0에 가깝도록 구성하였으며 방위각 제어기의 이득은 위의 표에서 확인할 수 있다.

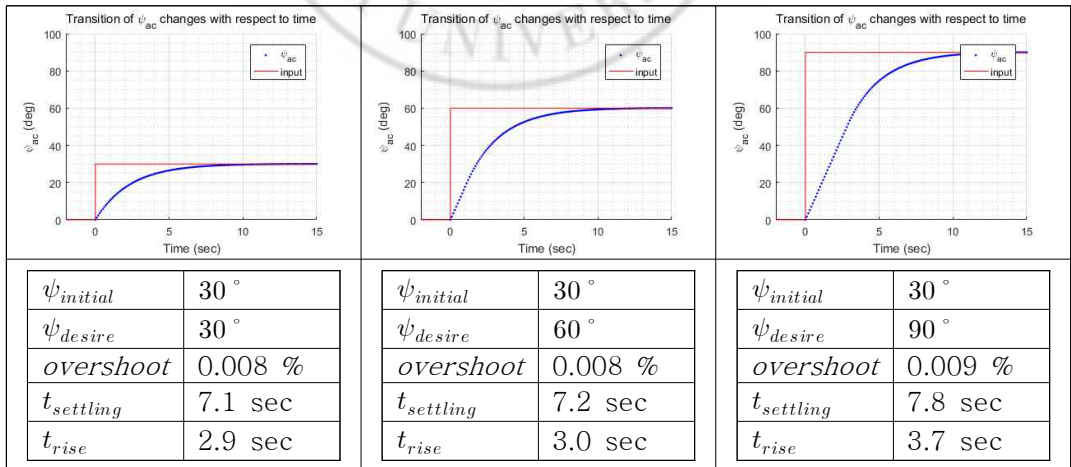
항공기 방위각 제어기에 리미터가 추가적으로 구성되어 있으므로 입력 값의 크기에 따라 제어기의 성능이 달라질 수 있다. 이때 입력 값에 따라 제어기의 성능의 변화 폭을 확인하기 위해 30° , 60° , 90° 세 값에 따른 결과를 정리하였다.

표 6 방위각 입력 값에 따른 Boeing 738 항공기의 방위각 제어 결과 비교



위의 표에서는 Medium급 항공기 Boeing 738의 모델을 적용하여 $v_{ac} = 15kt$ 으로 시뮬레이션을 수행하였을 때의 방위각 제어기의 성능을 확인할 수 있다. 모든 입력 값에 대해 오버슈트는 거의 없는 것을 확인할 수 있다. 수렴 시간은 약 5~6초 이었으며 상승 시간은 약 2~3초인 것을 확인할 수 있다.

표 7 방위각 입력 값에 따른 Boeing 773 항공기의 방위각 제어 결과 비교



위의 표에서는 Heavy급 항공기 Boeing 773의 모델을 적용하여 $v_{ac} = 15kt$ 으로 시뮬레이션을 수행하였을 때의 방위각 제어기의 성능을 확인할 수 있다. 모든 입력 값에 대해 오버슈트는 거의 없는 것을 확인하였다. 수렴 시간은 약 7~8초 이었으며 상승 시간은 약 3~4초인 것을 확인할 수 있다. Medium급 항공기에 적용하였던 제어기와 동일한 것을 적용하였으며 모델에 따른 성능 차이가 크지 않음을 볼 수 있다.

방위각 제어기 구성의 최종 목적은 항공기나 링크를 벗어나지 않고 스케줄의 경로를 추종하도록 하는 것이다.

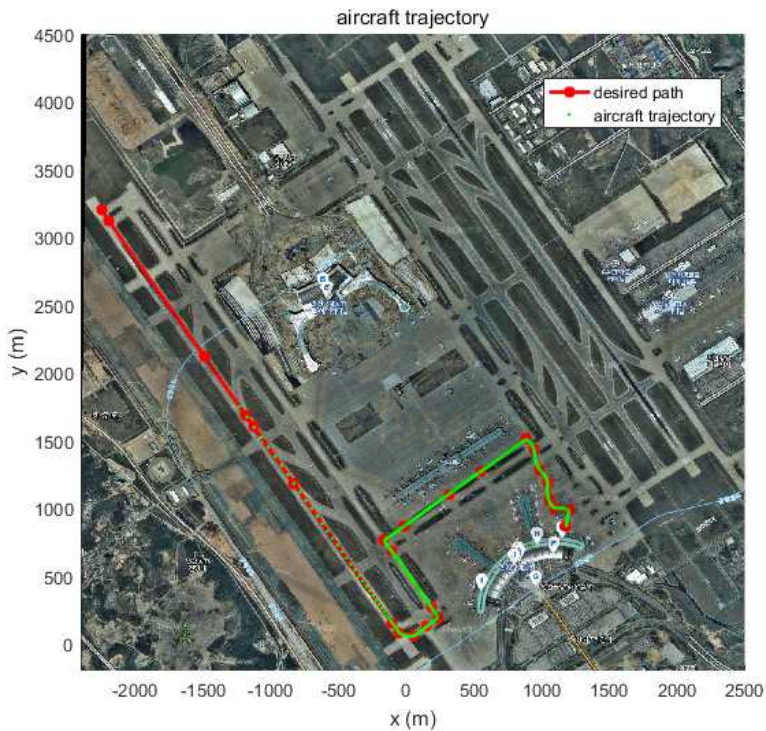


그림 16 항공기의 스케줄 경로와 실제 궤적 비교

위의 그림에서는 스케줄 된 경로와 항공기의 실제 궤적을 비교한 것을 확인할 수 있다. 방위각 제어기에 강조되었던 성능인 낮은 오버슈트와 빠른 수렴 시간으로부터 경로를 효과적으로 추종하는 것을 확인하였다.

4.2 속도 제어기 구성

입력된 스케줄의 링크 진출입 시간의 추종 가능 여부 판단을 위해 속도 제어기 개발이 필요하다. 제어기에 입력되는 속도 값 계산 방법은 아래 그림에서 확인할 수 있다.

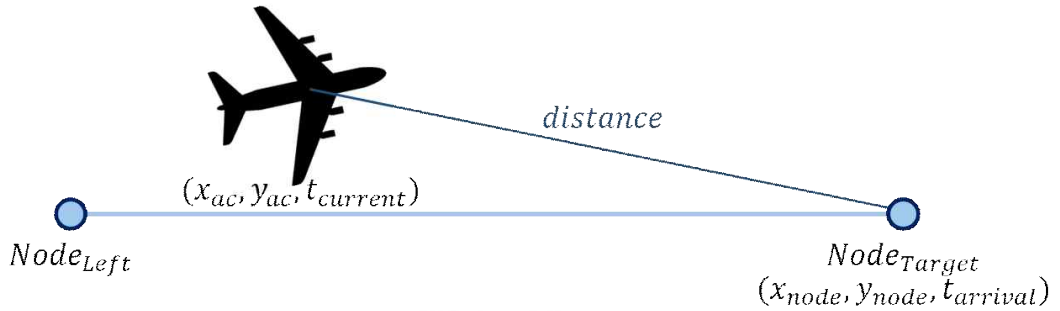


그림 17 속도 명령 계산

$$distance = \sqrt{(x_{node} - x_{ac})^2 + (y_{node} - y_{ac})^2} \quad (4.3)$$

$$v_{desire} = \frac{distance}{t_{arrival} - t_{current}} \quad (4.4)$$

현재 항공기의 위치와 목적 노드의 위치, 현재 시간과 도착 예정 시간을 통해 항공기의 속도를 위의 식 (4.3)와 식 (4.4)을 통해 계산할 수 있다.



그림 18 항공기 속도 제어기 블록선도

위의 그림에서는 속도 제어기의 구성을 확인할 수 있으며, 매순간 계산된 속도 명령 값이 입력된다. PID 제어기를 통해 추력 명령 값 κ 를 계산하게 되고, 리미터를 거친 후 최종 명령 값을 얻게 된다.

항공기 속도를 증가시키기 위해 엔진의 추력을 증가하게 되는데 감속시키기 위해서 엔진의 추력을 진행 방향의 역방향으로 가하는 방법이 있다. 항공기의 역추력 장치는 착륙 항공기가 활주로에서 속도를 줄이기 위한 경우에만 사용되므로 택시 및 유도로에서 속도를 줄이기 위해 브레이크 모델이 필요하다. 브레이크를 통한 감속이 아닌 지면과 바퀴의 마찰력과 공기 저항력을 통해 감속할 경우 힘의 크기가 부족하여 입력한 속도로 빠르게 수렴하기 어렵다. 아래와 같은 조건에서는 항공기의 브레이크를 작동하도록 하였으며, 이때의 항공기의 브레이크 힘은 아래의 식을 통해 계산하게 된다.

$$brake_{status} = \begin{cases} Active & (\frac{v_{ac} - v_{desire}}{v_{ac}} \geq 0.1) \\ Deactive & (\frac{v_{ac} - v_{desire}}{v_{ac}} < 0.1) \end{cases} \quad (4.5)$$

$$F_{brake} = -weight * \min((v_{desire} - v_{ac}) * 0.02, 0.15) \quad (4.6)$$

브레이크의 힘은 현재 속도와 v_{desire} 의 차이를 고려하여 계산하였으며 이때 최대 가속도의 크기는 0.15g를 넘지 못하도록 하였다.

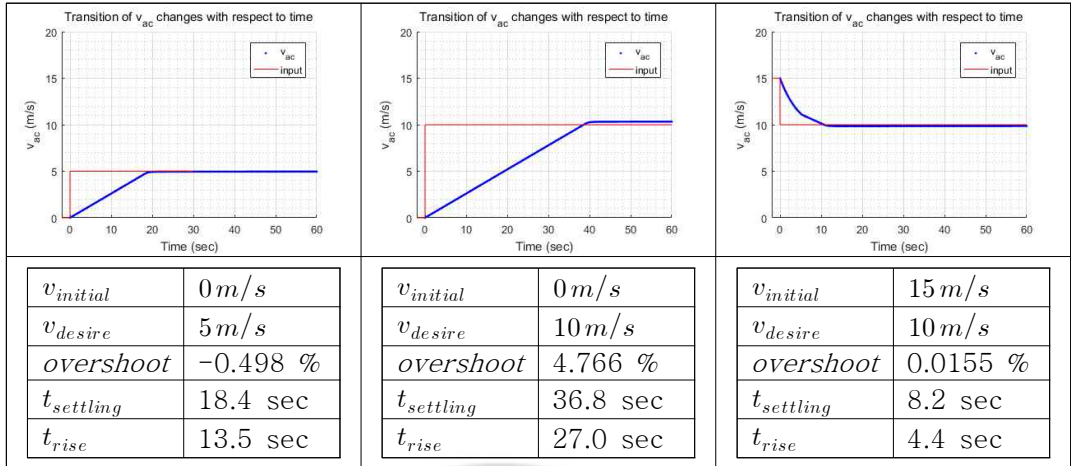
표 8 속도 PID 제어기 이득

Gain	Proportional	Integral	Derivative
값	0.4	0.001	0.1

지상에서 이동하는 항공기의 경우 조종사가 항공기의 속도를 정확하게 조종하기 위해 스로틀을 급하게 조작하지 않는다는 점을 감안하여 오버슈트 0에 가깝도록 구성하였으며 속도 제어기의 이득은 위의 표에서 확인할 수 있다.

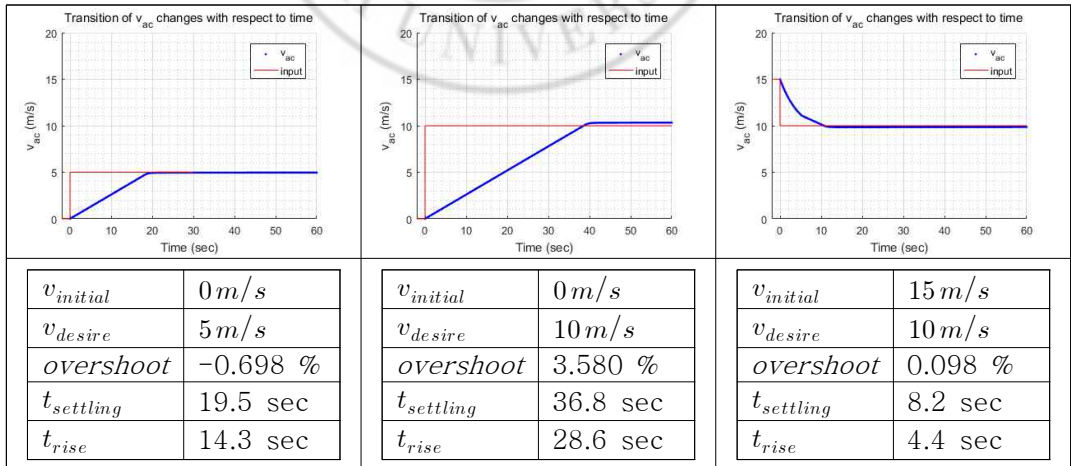
항공기 속도 제어기에도 리미터가 구성되어 있으므로 입력 값과 현재 상태의 차이에 의해 제어기의 성능이 달라질 수 있으므로 입력 값을 변화하며 비교가 필요하다. 항공기 초기속도가 0m/s 일 때, 입력 값을 5m/s와 10m/s으로 설정하여 두 경우에 대해 시뮬레이션을 수행하여 제어기의 성능 검증을 진행하였다. 항공기의 현재 속도가 입력 속도와 차이가 큰 경우 빠르게 수렴하기 위해 추가한 브레이크의 성능 확인이 필요하여 초기속도 15m/s일 때, 입력 값을 10m/s로 설정하여 제어기 성능을 확인하였다.

표 9 방위각 입력 값에 따른 Boeing 738 항공기의 속도 제어 결과 비교



위의 표에서는 Medium급 항공기 Boeing 738의 모델을 적용하여 시뮬레이션을 수행하였을 때의 속도 제어기의 성능을 확인할 수 있다. 모든 입력 값에 대해 오버슈트는 5% 이하인 것을 확인할 수 있으며 제어기 성능 조건을 만족하였다. 지상 이동 시 항공기 추력의 제한으로 인해 초기 속도와 입력 속도의 차이와 상승 시간, 정착 시간이 비례하는 것을 볼 수 있다.

표 10 방위각 입력 값에 따른 Boeing 773 항공기의 속도 제어 결과 비교



위의 표에서는 Heavy급 항공기 Boeing 773의 모델을 적용하여 시뮬레이션을 수행

하였을 때의 속도 제어기의 성능을 확인할 수 있다. Boeing 738 모델의 속도 제어기와 비슷하게 모든 입력 값에 대해 낮은 오버슈트를 가지는 것을 볼 수 있다. Boeing 773 모델에서 또한 추력 제한으로 인해 초기 속도와 입력 값의 차이와 상승 시간, 정착 시간이 비례하는 것을 확인할 수 있다. 두 모델에서 모두 속도 입력 값이 초기 속도보다 낮은 상황에서 오버슈트, 상승 시간, 정착 시간이 모두 짧은 것을 확인하였다. 본 시뮬레이터의 항공기 분리 유지 알고리즘을 위해 항공기 모델의 브레이크의 정지 성능을 강조한 것의 결과이다.

전체 시뮬레이션을 통해 항공기의 속도 제어기가 원활하게 작동하는 것을 확인하기 위해 스케줄의 링크 진출입 시간과 실제 항공기 모델의 링크 진출입 시간을 비교하여 확인이 필요하다.

표 11 DEP001 항공기의 스케줄과 모델의 링크 진출입 시간 비교

링크 명	스케줄		항공기 모델		탈출 시간오차
	진입 시간	탈출 시간	진입시간	탈출시간	
gate70	1129 sec	1167 sec	1129 sec	1167 sec	0 sec
Rp95	1167 sec	1178 sec	1167 sec	1178 sec	0 sec
Rp96	1178 sec	1195 sec	1178 sec	1195 sec	0 sec
Rp101	1195 sec	1226 sec	1195 sec	1226 sec	0 sec
Rp107	1226 sec	1247 sec	1226 sec	1247 sec	0 sec
Rp169	1247 sec	1300 sec	1247 sec	1300 sec	0 sec
Tx74	1300 sec	1367 sec	1300 sec	1367 sec	0 sec
Tx61	1367 sec	1380 sec	1367 sec	1380 sec	0 sec
Tx62	1380 sec	1445 sec	1380 sec	1445 sec	0 sec
Tx63	1445 sec	1458 sec	1445 sec	1458 sec	0 sec
Tx52	1458 sec	1472 sec	1458 sec	1472 sec	0 sec
Tx53	1472 sec	1487 sec	1472 sec	1487 sec	0 sec
Tx54	1487 sec	1501 sec	1487 sec	1501 sec	0 sec

위의 표에서는 시뮬레이터에 입력된 스케줄의 항공기 중 하나인 편명 DEP001 항공기의 스케줄 진출입 시간과 실제 진출입 시간을 비교하였다. 모든 링크의 진출입 시간이 동일하여 속도 제어기가 효과적으로 엔진 추력을 제어한 것을 볼 수 있다.

표 12 ARR001 항공기의 스케줄과 모델의 링크 진출입 시간 비교

링크 명	스케줄		항공기 모델		탈출 시간오차
	진입 시간	탈출 시간	진입시간	탈출시간	
Rwy7	2057 sec	2071 sec	2057 sec	2065 sec	0 sec
Rwy6	2071 sec	2084 sec	2065 sec	2074 sec	6 sec
Rwy5	2084 sec	2093 sec	2074 sec	2081 sec	10 sec
Rwy4	2093 sec	2110 sec	2081 sec	2108 sec	12 sec
Tx1	2110 sec	2176 sec	2108 sec	2176 sec	2 sec
Tx2	2176 sec	2222 sec	2176 sec	2222 sec	0 sec
Tx3	2222 sec	2241 sec	2222 sec	2241 sec	0 sec
Tx4	2241 sec	2261 sec	2241 sec	2261 sec	0 sec
Tx5	2261 sec	2276 sec	2261 sec	2280 sec	0 sec
Tx6	2276 sec	2340 sec	2276 sec	2340 sec	0 sec
Tx7	2340 sec	2357 sec	2340 sec	2357 sec	0 sec
Tx8	2357 sec	2423 sec	2357 sec	2423 sec	0 sec
Tx9	2423 sec	2436 sec	2423 sec	2436 sec	0 sec
Tx10	2436 sec	2502 sec	2436 sec	2502 sec	0 sec
Tx12	2502 sec	2515 sec	2502 sec	2515 sec	0 sec
Tx30	2515 sec	2587 sec	2515 sec	2587 sec	0 sec
Tx13	2587 sec	2603 sec	2587 sec	2603 sec	0 sec
Rp161	2603 sec	2638 sec	2603 sec	2638 sec	0 sec
Rp19	2638 sec	2717 sec	2638 sec	2717 sec	0 sec
Rp34	2717 sec	2774 sec	2717 sec	2774 sec	0 sec
Rp36	2774 sec	2853 sec	2774 sec	2852 sec	1 sec
Rp56	2853 sec	2872 sec	2852 sec	2872 sec	0 sec
gate32	2872 sec	2966 sec	2872 sec	2966 sec	0 sec

위의 표에서는 시뮬레이터에 입력된 스케줄의 항공기 중 하나인 편명 ARR001 항공기의 스케줄 진출입 시간과 실제 진출입 시간을 비교한 것을 볼 수 있다. 활주로에서 감속하는 과정에서 링크의 진출입 시간에 약간의 차이가 있었다. 항공기 모델은 활주로의 착륙하는 순간부터 활주로를 탈출하는 시점까지 상대적으로 일정한 가속도로 감속하도록 설계되어 있다. FCFS 스케줄러의 항공기 모델과 Fast-Time 시뮬레이터의 항공기 모델 모두 활주로 점유 시간은 2초 차이로 계산되었다. FCFS 스케줄러의 경우 상대적으로 큰 가속도로 감속을 하였으며 동일한 활주로 점유시간을 갖기 위해 활주로 탈출 직전의 속도가 상당히 큰 것을 확인하였다. Tx1 진입 시 빠른 속도로 인해 회전 반경이 커져 링크를 이탈하는 문제가 발생하였다. 따라서 본 시뮬레이터의 항공기 모델이

보다 더 현실적인 항공기의 운동을 모사한 것으로 간주할 수 있다.

4.3 분리 거리 유지 제어기 구성

본 시뮬레이터의 핵심 기능인 두 항공기 간의 분리 거리 유지를 위해 필요한 제어기이며 앞서 설명한 두 제어기와 다르게 개루프 시스템으로 구성하였다.

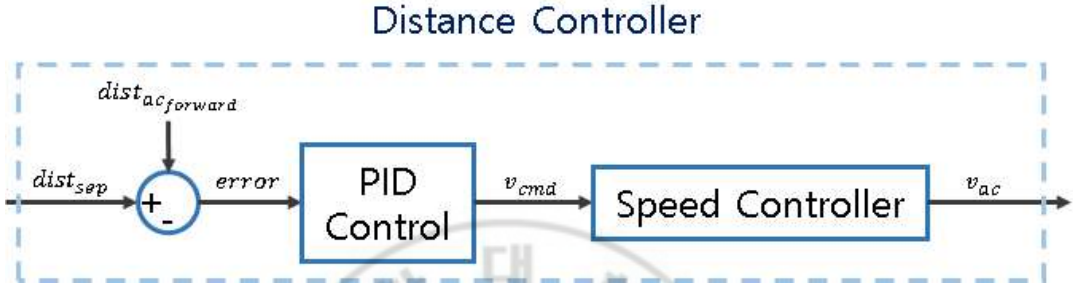


그림 19 분리 거리 유지 제어기 블록선도

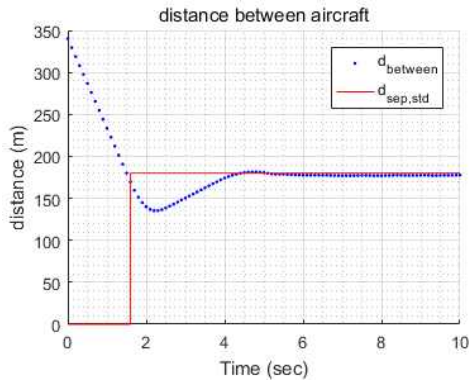
위의 그림에서는 분리 거리 유지 제어기의 구성을 확인할 수 있다. 분리 안전거리 기준을 준수하지 못하는 경우, 앞 항공기와의 거리를 입력하여 PID 제어기를 통해 속도 명령 값을 산출하게 되고, 속도 명령 값을 속도 제어기에 입력하여 속도를 제어하여 분리 거리 기준에 수렴하도록 한다.

표 13 분리 거리 유지 PID 제어기 이득

Gain	Proportional	Integral	Derivative
값	0.15	0.001	0.3

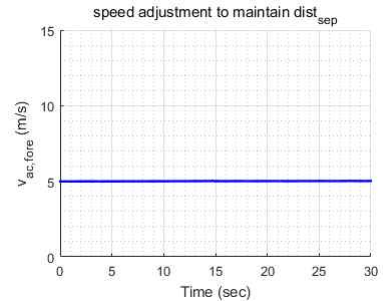
분리 거리 유지 제어기의 이득은 위의 표에서 확인할 수 있다.

두 항공기의 충돌을 방지하기 위한 분리 거리 유지 제어기의 성능을 검증하기 위해 선행 항공기의 속도를 $5m/s$ 로 고정시키고, 후행 항공기가 $15m/s$ 로 접근하는 스케줄을 생성하였다. 두 항공기의 거리가 $180m$ 이내로 가까워졌을 때부터 제어기가 항공기 속도를 제어하도록 구성하였다.

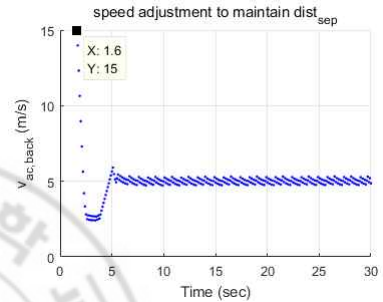


(a) 시간에 따른 분리 거리의 변화

$d_{sep.std}$	180 m
overshoot	24.85 %
$t_{settling}$	2.4 sec



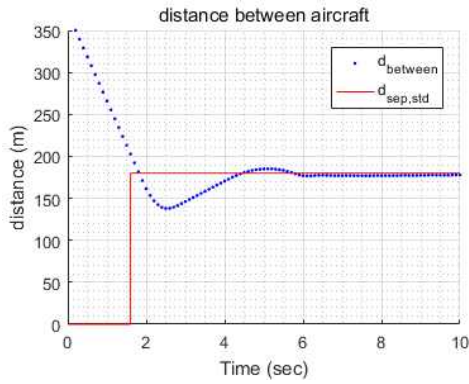
(b) 앞 항공기의 속도 그래프



(c) 뒤 항공기의 속도 그래프

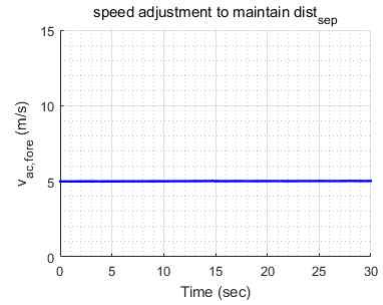
그림 20 분리 거리 유지 제어를 통한 Boeing 738의 속도 제어 및 분리 거리 변화

위의 그림에서는 두 항공기간의 분리거리가 분리 거리 기준보다 가까워진 순간부터 속도 제어를 통해 분리거리가 증가하는 것을 확인할 수 있다. 후행 항공기는 Medium 급 항공기의 모델이 적용되었다. 분리 유지 알고리즘의 특성 상 오버슈트 크게 발생하였으며 두 항공기가 가장 근접하였을 때의 분리거리는 135m이었다. 두 항공기의 분리 거리는 약 2.4초 이후에 분리 거리 기준에 수렴하였다.

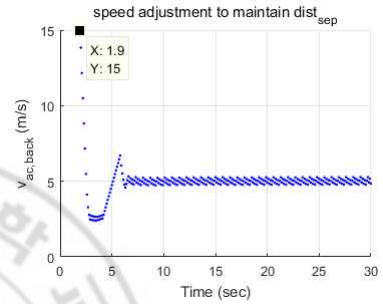


(a) 시간에 따른 분리 거리의 변화

$d_{sep.std}$	180 m
overshoot	23.4 %
$t_{settling}$	2.2 sec



(b) 앞 항공기의 속도 그래프



(c) 뒤 항공기의 속도 그래프

그림 21 분리 거리 유지 제어를 통한 Boeing 773의 속도 제어 및 분리 거리 변화

위의 그림에서는 두 항공기간의 분리거리가 분리 거리 기준보다 가까워진 순간부터 속도 제어를 통해 분리거리가 증가하는 것을 확인할 수 있다. 후행 항공기는 Heavy급 항공기의 모델이 적용되었다. 두 항공기가 가장 근접하였을 때의 분리거리는 138m이었다. 두 항공기의 분리 거리는 약 2.2초 이후에 분리 거리 기준에 수렴하였다.

5. 항공기 분리 유지 알고리즘

공항 내 지상에서 항공기가 이동 중 충돌 위험이 발생하는 경우는 앞, 뒤 간격 미준수와 교차로에 다수의 항공기가 접근하는 경우가 있다. 이러한 경우 한 대 이상의 항공기를 정지함으로써 충돌을 방지하는데, 정지 항공기 선정에 있어 기준이 명확한 경우와, 모호한 경우로 나누어 볼 수 있다. 충돌 위험이 발생할 수 있는 다섯 가지 상황에 대해 정리하였으며, 이 중 네 가지 경우에 대해 정지 및 재출발 알고리즘을 적용하였다.

5.1 항공기 간의 거리 계산

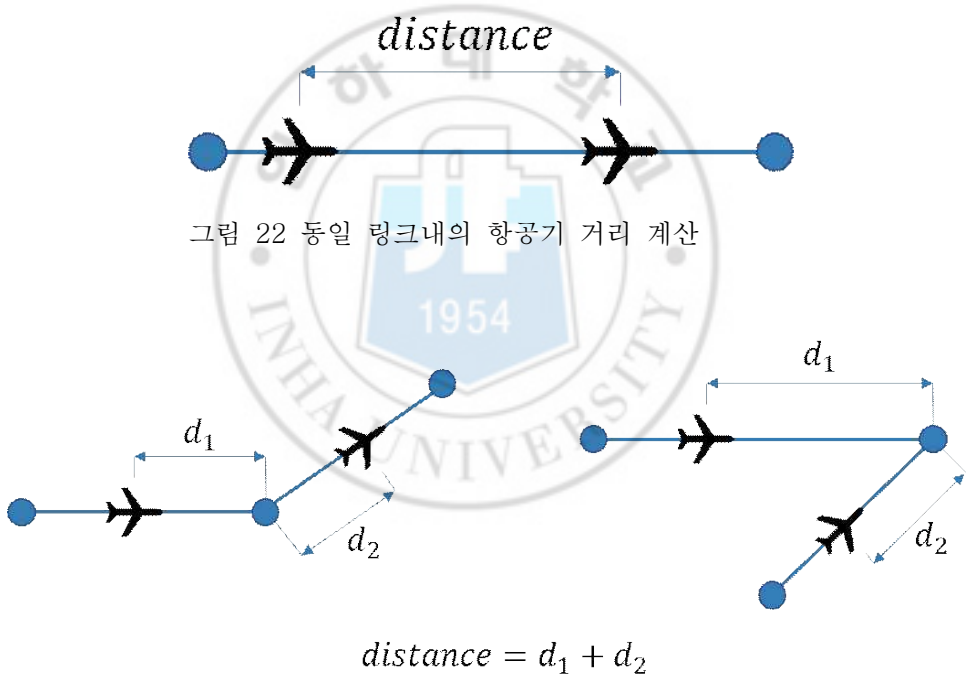


그림 22 동일 링크내의 항공기 거리 계산

그림 23 이웃한 링크의 항공기의 거리 계산

항공기의 충돌을 방지하기 위해 분리 거리를 먼저 계산해야 한다. 서로 연결되어 있지 않은 링크 상에서 이동하는 항공기 간의 충돌 위험은 없다는 가정 하에, 두 항공기의 링크가 연결되어 있거나 동일한 경우에만 거리 계산을 하였다. 실제 두 항공기 간의 거리가 아닌, 링크가 1차원 상에 펼쳐진 상태로 놓여 있을 때의 두 항공기의 거리를 계

산하였다. 항공기가 이동 중인 링크의 두 노드 중 항공기가 향하는 노드를 $Node_{target}$ 로 정의하였으며, 다른 노드를 $Node_{source}$ 로 정의하였다.

5.2 분리 유지 알고리즘

충돌 위험이 발생할 수 있는 5가지 상황 중, 정지 항공기 선정 기준이 명확한 3가지 경우와, 기준이 모호한 2가지 경우가 있다.

5.2.1 항공기 간의 앞, 뒤 구별이 가능

5.2.1.1 Case 1: 동일 선상

5.2.1.1.1 동일 링크에서 진행

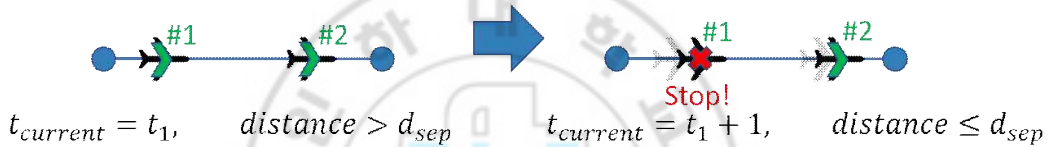


그림 24 항공기 분리 거리 유지 실패 Case 1-1

위의 그림에서는 충돌 방지를 위해 항공기의 속도를 낮추는 상황을 볼 수 있다. 두 항공기는 동일한 링크 내에서 같은 방향으로 정상적으로 진행 중이며, 분리 거리가 위반되는 순간에, 뒤따라오는 1번 항공기의 속도 제어를 통해 두 항공기의 거리가 분리 거리 기준과 동일하도록 유지한다. 이때 도착 노드로부터 두 항공기까지의 거리를 비교하여 선행 항공기와 후행 항공기를 구별하였다. 이후 2번 항공기의 속도가 증가하거나 두 항공기의 진행 중인 링크가 서로 연결되어 있지 않아 분리 유지 알고리즘의 적용이 불필요한 경우, 1번 항공기는 입력된 스케줄의 도착 시간준수를 위해 속도를 증가시켜 이동하게 된다.

5.2.1.1.2 이웃한 두 링크에서 진행

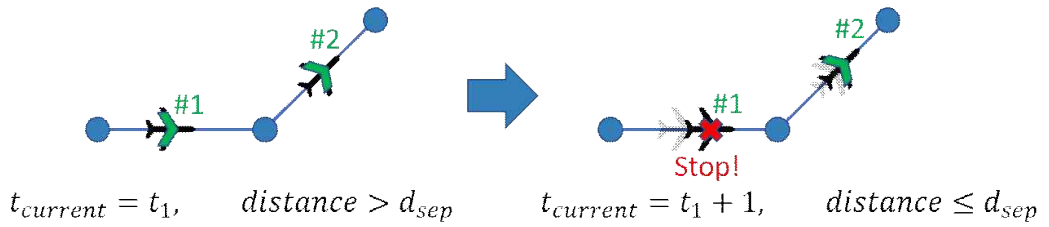


그림 25 항공기 분리 거리 유지 실패 Case 1-2

위의 그림에서는 서로 이웃한 링크에서 정상적으로 이동 중인 두 항공기를 볼 수 있으며, 분리 거리가 위반되는 순간에, 뒤따라오는 1번 항공기의 속도 제어를 통해 두 항공기의 분리 거리가 분리 거리 기준과 동일하도록 유지한다. 이때 두 항공기의 $Node_{target}$ 와 $Node_{source}$ 의 비교를 통해 선행 항공기와 후행 항공기를 구별하였다. 이후 2번 항공기의 속도가 증가하거나, 두 항공기의 진행 중인 링크가 서로 연결되어 있지 않아 분리 유지 알고리즘의 적용이 불필요한 경우, 1번 항공기는 입력된 스케줄의 도착 시간 준수를 위해 속도를 증가시켜 이동하게 된다.

5.2.1.2 Case 2: 다른 항공기가 사용 중인 다음 링크

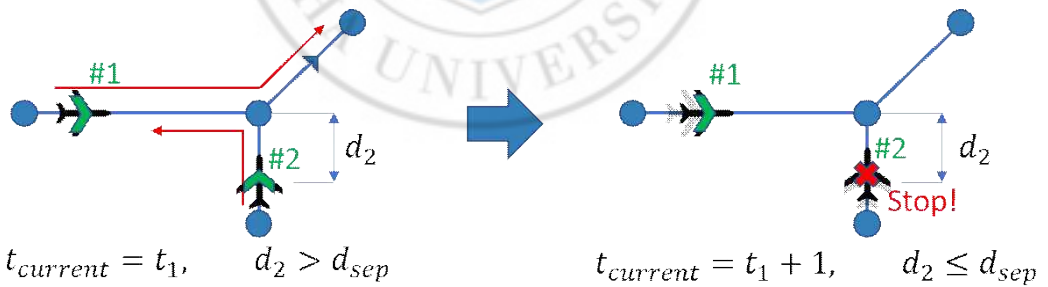


그림 26 항공기 분리 거리 유지 실패 Case 2

위의 그림에서는 두 항공기가 교차로에 접근 중인 상황을 볼 수 있으며, 앞선 예시와 다르게 $Node_{target}$ 와 $Node_{source}$ 정보만을 통해 선행과 후행 항공기의 구별이 불가능한 상황이다. 이런 경우 선행 항공기의 선정에 따라 충돌 위험이 크게 증가할 수 있다. 각 항공기의 경로 계획이 빨간색 화살표의 방향과 같을 때, 2번 항공기가 교차로 통행 우

선권을 부여받아 노드를 통과하는 경우, 다음 상황에서 1번 항공기와 마주보는 상황이 발생한다. 이러한 상황을 사전에 방지하기 위해 항공기의 다음 경로를 고려하여 교차로에 먼저 진입할 항공기를 선정해야 한다. 두 항공기가 동일 노드로 접근하는 경우, 각 항공기의 다음 링크에 상대 항공기의 유무를 확인하여 통과 우선 항공기를 구별하였다. 그림과 같은 상황에서는 1번 항공기가 우선권을 부여 받게 되며, 두 항공기의 충돌을 방지하기 위해 2번 항공기를 정지해야한다. Case 1에서는 충돌 방지를 위해 두 항공기의 분리 거리가 분리 거리 기준보다 가까워진 경우에 분리 거리 유지 제어가 항공기의 운동에 관여하였지만, Case 2에서는 차선 순위 항공기와 $Node_{target}$ 의 거리가 분리 거리 기준보다 가까워지는 시점에 정지하도록 하였다.

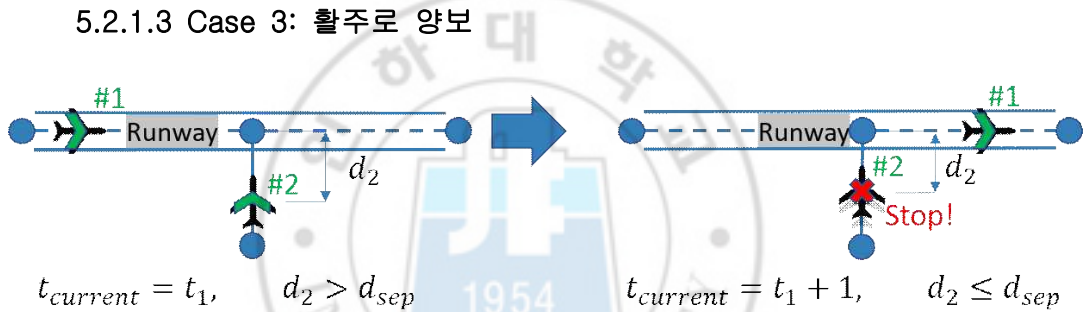


그림 27 항공기 분리 거리 유지 실패 Case 3

위의 그림에서는 두 항공기가 교차로에 접근중인 상황을 볼 수 있으며, Case 2와 동일하게 $Node_{target}$ 와 $Node_{source}$ 정보만을 통해 선행과 후행 항공기의 구별이 불가능한 상황이다. 교차로 통과 후 두 항공기의 경로가 겹치지 않지만 통과 우선 항공기 선정에 따라 충돌 위험이 크게 증가함과 동시에 지연 시간이 크게 발생할 수 있다. 공항의 용량을 위해 이착륙 항공기는 최대한 빠른 시간에 이륙하거나 활주로를 탈출해야한다. 1번 항공기가 활주로에서 이착륙중인 상황에서 2번 항공기가 통과 우선 항공기로 선정될 경우, 1번 항공기의 활주로 점유 시간이 크게 증가해 공항 운용이 비효율적일 수 있다. 이러한 문제를 방지하기 위해 항공기의 현재 이동 중인 링크가 활주로인 경우, 해당 항공기를 통과 우선 항공기로 구별하였으며 2번 항공기를 정지해야한다. Case 2와 동일하게 차선 순위 항공기와 $Node_{target}$ 의 거리가 분리 거리 기준보다 가까워지는 시

점에 정지하도록 하였다.

5.2.2 항공기 간의 앞, 뒤 구별이 불가능

5.2.2.1 Case 4: 서로 다른 링크에서 하나의 노드로 접근

앞서 설명한 3가지 경우와 다르게 항공기의 앞, 뒤 구별이 불가능하고, 통과 우선순위 부여 방식에 따른 충돌 위험이 발생하지 않는 경우이다. 이때 교차로 우선 통과 항공기 선정에 위해 3가지 기준을 제시하였다.

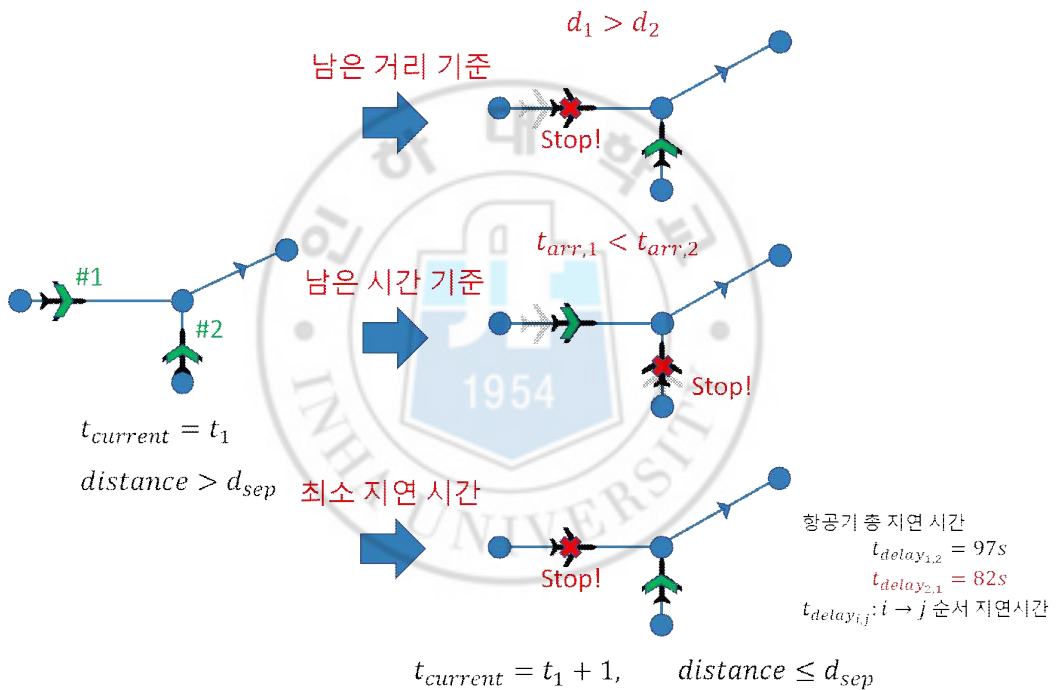


그림 28 항공기 분리 거리 유지 실패 Case 4

위의 그림에서는 교차로로 접근중인 두 항공기 중 3가지 기준에 따라 통과 우선 항공기 선정 결과를 확인할 수 있다.

5.2.2.1.1 노드로부터 가까운 항공기 우선 통과

위의 그림 28-A는 두 항공기의 분리 거리가 분리 기준보다 가까워졌을 경우 노드에

멀리 있는 1번 항공기를 정지하고 2번 항공기를 먼저 통과시키는 그림이다. 노드로부터 두 항공기까지의 거리가 동일한 경우, 노드 도착 예정 시간이 이른 항공기를 우선 통과 항공기로 지정하였다. 2번 항공기가 교차로를 빠져 나간 뒤 두 항공기의 거리가 분리 기준을 만족할 때 재출발하도록 하였다.

5.2.2.1.2 노드 도착 예정 시간이 이른 항공기 우선 통과

위의 그림 28-B는 두 항공기의 분리 거리가 분리 기준보다 가까워졌을 경우 노드 도착 예정 시간이 이른 1번 항공기를 통과 우선 항공기로 지정하고, 2번 항공기를 정지하는 방법이다. 두 항공기의 노드 도착 예정시간이 동일한 경우, 노드로부터 더 가까이 있는 항공기를 우선 통과 항공기로 지정하였다. 1번 항공기가 교차로를 빠져 나간 뒤 두 항공기의 거리가 분리 기준을 만족할 때 재출발하도록 하였다.

5.2.2.1.3 최소 지연 시간 순서로 통과

위의 그림 28-C는 두 항공기가 교차로로 접근중인 경우, 노드를 통과할 수 있는 항공기 순서의 모든 조합에 대해 시뮬레이션 수행 후, 총 지연 시간이 가장 짧은 순서로 항공기를 통과시키는 그림이다. 그림과 같이 1번 항공기와 2번 항공기가 교차로에 접근중이며 이때 항공기가 통과할 수 있는 순서는 2가지가 있다. 1번 항공기를 먼저 통과시킬 경우 지연 시간이 97초이며, 2번 항공기를 먼저 통과시킬 경우 지연시간이 82초로 계산된 경우, 두 번째 순서로 통과할 때 지연시간이 가장 짧아 2번 항공기를 먼저 통과시키도록 하였다.

5.2.2.2 Case 5: 링크 내에서 마주 오는 항공기



그림 29 항공기 분리 거리 유지 실패 Case 5

그림 29는 두 항공기가 동일 링크 내에서 마주보며 접근하는 상황을 보여준다. 이러한 경우는 고려하지 않았으며 서로를 인지하지 못한 채 이동한다. 스케줄 검증 차원에서 사용자에게 정보를 표시하도록 하였다.

5.2.2.3 Case 6: 예외 상황

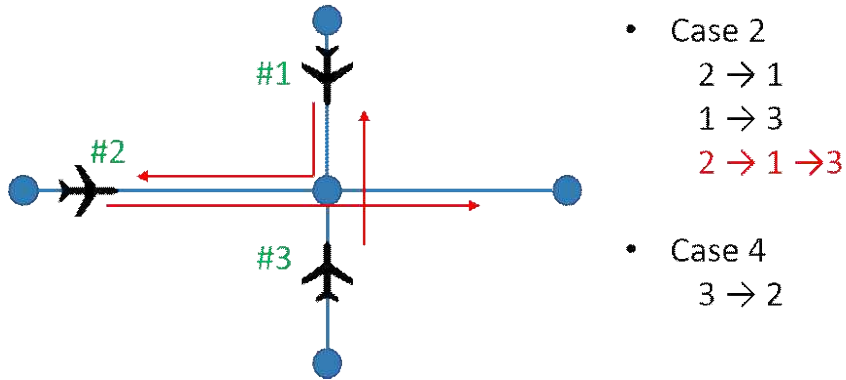


그림 30 복합적 상황에서의 예외 처리

그림 30은 앞서 설명한 Case 2와 Case 4가 복합적으로 일어나 발생할 수 있는 문제점을 보여준다. Case 2는 항공기의 다음 경로를 고려하여 선행 항공기를 선정한다. 이에 따라 교차로를 통과하는 항공기의 순서는 2, 1, 3번 항공기이다. 사용자가 선택한 기준에 따라 Case 4에서 2번 항공기와 3번 항공기를 비교하였을 경우, 3번 항공기가 선행 항공기로 지정될 수 있다. 결과적으로 3대의 항공기 모두 후행 항공기로 지정되어 제자리에 정지하게 된다. Case 1, 2, 3의 경우 위반 시 항공기 충돌을 초래하며, Case 4의 경우 교차로에서 항공기의 효율적인 운용을 위한 기준이다. 따라서 여러 상황이 복합적으로 발생할 경우, Case 1, 2, 3을 위반하지 않는 조건하에 우선순위가 가장 높은 항공기를 먼저 통과하도록 하였다.

6. 시뮬레이션 결과

앞서 설명한 5가지 상황에 대한 항공기 분리 유지 알고리즘을 적용하여 FCFS 스케줄러에 의해 도출된 결과로 시뮬레이션을 수행하였다. 시뮬레이션을 수행한 테스트 시나리오는 인천국제공항에서 약 1시간 10분 동안 총 72대의 항공기가 이착륙하는 스케줄이며, 분리 간격은 180m로 설정하였다.

항공기 정지 기능을 비 활성화하여 동일한 스케줄로 시뮬레이션 한 결과 후행 항공기의 분리 간격이 유지되지 못하는 횟수가 74회 발생하였으며, 교차로로 다수의 항공기가 접근하는 경우도 26회 발생하였다. 5가지 경우에 대한 발생 횟수는 표 1에 정리되어 있다.

표 14 항공기 정지 기능 미 적용 시 분리 거리 유지 실패 횟수

항공기 접근 케이스	횟수
Case 1-1: 동일 선상 (동일 링크)	15
Case 1-2: 동일 선상 (이웃 링크)	22
Case 2: 사용 중인 다음 링크	4
Case 3: 활주로 양보	0
Case 4: 교차로 접근	10
Case 5: 마주보며 접근	0

분리 유지 기능을 사용하였을 경우, 3가지 정지 알고리즘에 따라 지연 시간의 차이가 발생함을 확인하였다. 표 2에 정리된 결과는 오히려 최소 지연 시간 기준을 적용했을 경우, 전체 지연 시간이 가장 짧지 않음을 볼 수 있는데, 이는 각 노드에서의 지연을 최소화하는 것이 전체 시스템의 지연을 최소화 하지 못할 수도 있음을 보여준다.

표 15 정지 알고리즘 별 지연 항공기 대수 및 평균 지연 시간

정지 알고리즘	지연 항공기 대수	평균 지연 시간
남은 거리 기준	17	7 sec
남은 시간 기준	17	7 sec
최소 지연 시간 기준	16	10 sec

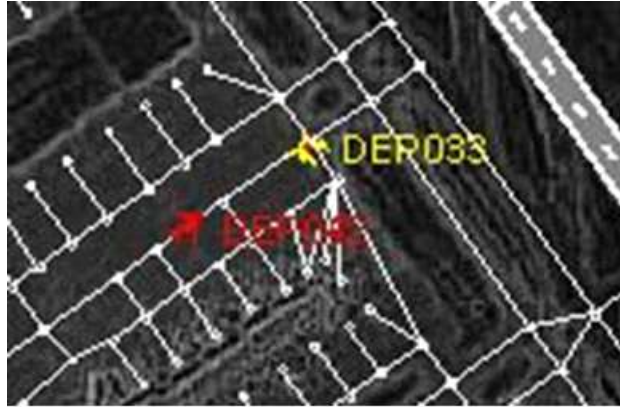
분리 유지 기능을 적용하였을 경우, Case 1-5중 3개의 Case가 발생하는 것을 확인하였다.



#	Flight ID	Velocity	HDG	Link	Node	Dist_F	Dist_A	Status
1	DEP005	5.8m/s	159	Tx61	10041	0.0	0.0	Moving
2	DEP015	4.6m/s	323	Rp17	10113	0.0	0.0	Moving
3	DEP013	6.1m/s	54	Rp20	20240	173.5	0.0	Slow down
4	DEP011	1.3m/s	54	Rp18	10007	200.2	0.0	Moving
5	-	-	-	-	-	-	-	-

그림 31 시나리오에서 발견된 Case 1-2

위의 그림 31은 테스트 시나리오에서 발견된 Case1-2이며 ‘DEP011’ 항공기와 ‘DEP013’ 항공기 간에 발생하였다. 그림의 아래 부분에서는 각 항공기의 상태 정보를 확인할 수 있으며 ‘DEP013’ 항공기와 선행 항공기와의 분리 거리를 의미하는 $dist_F$ 의 값이 분리 거리 기준인 180m보다 가까워져 속도를 낮춘 것을 확인할 수 있다.



#	Flight ID	Velocity	HDG	Link	Node	Dist_F	Dist_A	Status
1	ARR008	4.7m/s	234	Rp19	10079	0.0	0.0	Moving
2	DEP027	7.1m/s	144	Tx62	10043	0.0	0.0	Moving
3	ARR011	7.3m/s	145	Tx6	10062	0.0	0.0	Moving
4	DEP042	0.0m/s	54	Rp85	10088	80.6	0.0	Stopped
5	DEP039	7.0m/s	144	Tx51	10020	0.0	0.0	Moving
6	ARR013	7.1m/s	145	Tx49	10038	0.0	0.0	Moving
7	DEP033	4.4m/s	254	Rp81	10088	0.0	0.0	Moving
8	-	-	-	-	-	-	-	-

그림 32 시나리오에서 발견된 Case 2

위의 그림 32는 테스트 시나리오에서 발견된 Case 2이며 ‘DEP033’ 항공기와 ‘DEP042’ 항공기 간에 발생하였다. 본 시뮬레이션에 사용된 교차로 통과 우선순위 부여 기준은 거리 기준이었으므로 접근중인 노드 ‘10088’에 상대적으로 더 가까이 있는 ‘DEP032’ 항공기가 교차로를 먼저 빠져나가야 한다. 이때 ‘DEP042’ 항공기의 다음 링크는 ‘Rp81’이었으며 ‘DEP033’ 항공기와 마주보는 상황을 방지하기 위해 통과 우선순위 부여 기준을 무시하고 우선순위를 부여받지 못하였다. 그림의 아래 부분에서 교차로로부터 ‘DEP042’ 항공기가 분리 거리 기준보다 가까워 충돌 방지를 위해 정지한 것을 확인할 수 있다.



#	Flight ID	Velocity	HDG	Link	Node	Dist_F	Dist_A	Status
1	ARR001	8.0m/s	144	Tx30	10058	0.0	0.0	Moving
2	DEP014	4.7m/s	234	Rp19	10079	0.0	148.9	Moving
3	ARR002	7.4m/s	144	Tx6	10062	0.0	0.0	Moving
4	ARR003	8.4m/s	349	Tx48	10109	0.0	0.0	Moving
5	DEP009	5.1m/s	144	Rp17	10007	0.0	0.0	Moving
6	DEP010	2.9m/s	285	Rp33	10079	0.0	148.9	Stopping
7	DEP008	2.6m/s	144	gate48	20306	0.0	0.0	Moving
8	-	-	-	-	-	-	-	-

그림 33 시나리오에서 발견된 Case 4

위의 그림 33은 테스트 시나리오에서 발견된 Case 4이며 ‘DEP014’ 항공기와 ‘DEP010’ 항공기 간에 발생하였다. 두 항공기 모두 ‘10079’ 노드로 접근중이며 교차로로부터 상대적으로 더 가까이 있는 ‘DEP014’ 항공기가 통과 우선순위를 부여 받은 것을 확인할 수 있다. 두 항공기의 분리 거리가 분리 거리 기준보다 가까워 ‘DEP010’ 항공기는 충돌을 방지하기 위해 정지중인 것을 확인할 수 있다.

7. 결론

본 논문에서는 FCFS 스케줄러와 연동이 가능한 fast-time 시뮬레이터와 항공기 충돌 방지를 위한 알고리즘을 개발하였다. 비교적 단순한 항공기 모델을 통해 도출된 스케줄러의 결과의 유효성을 검증하기 위해 2차원 질점 항공기 운동 모델을 개발 및 검증하였다. 항공기의 이동을 모사하기 위해 필요한 조종사 모델을 대체하기 위해 개발된 방위각 제어기와 속도 제어기의 구성에 대하여 기술하고, 제어기의 성능을 검증하였다.

항공기의 스케줄이 의도한 것과 다르게 작동할 시 발생할 수 있는 문제들을 분석하였다. 항공기 전후 분리 간격 위반으로 인한 충돌, 교차로에서의 충돌 가능성이 발견되었다. 이를 해결하기 위한 알고리즘을 설명하였으며 항공기의 지연을 최소화하기 위한 한 3가지 알고리즘을 제시하였다.

개발된 시뮬레이터는 인천 국제공항에서의 72대의 항공기의 출도착 스케줄을 이용하여 테스트 되었으며 우선순위 설정 방식에 따른 다른 결과가 나타남을 확인하였다.

8. 참고문헌

- [1] 김태영, 박배선, 이현웅, 이학태, ‘항공기 지상 이동 Fast-Time 시뮬레이터 개발’, 2018 항행학회 학술대회.
- [2] Jung, Y., Malik, W., Tobias, L., Gupta, G., Hoang, T., and Hayashi, M., "Performance Evaluation of SARDA: An Individual Aircraft-based Advisory Concept for Surface Management," Air Traffic Control Quarterly, Vol. 22, Number 3, 2015, p. 195-221, April 2015.
- [3] Yeonju Eun, Daekeun Jeon, Hanbong Lee, Yoon C. Jung, Zhifan Zhu, Myeongsook Jeong, Hyounkyong Kim, Eunmi Oh, and Sungkwon Hong. "Optimization of Airport Surface Traffic: A Case-study of Incheon International Airport", 17th AIAA Aviation Technology, Integration, and Operations Conference, AIAA AVIATION Forum,
- [4] B. S. Park, H. W. Lee, and H. T. Lee, "Extended First-Come First-Served Scheduler for Airport Surface Operation", International Journal of Aeronautical and Space Sciences, Vol 19, Issue 2, pp 509-517
- [5] Zhifan Zhu, "Surface Operations Simulator and Scheduler (SOSS) Presentation", Joint Workshop for KAIA/KARI/IIAC-NASA Collaboration, April 5-7, 2016
- [6] Di Wu, Yiyuan J. Zhao, and Brian Capozzi, "Fundamental Surface Trajectory Models for Air Traffic Automation", 10th AIAA Aviation Technology, Integration, and Operation (ATIO) Conference