JANT 한국항행학회논문지
Journal of Advanced Navigation Technology

항공 우주 해상

J. Adv. Navig. Technol. 23(1): 1-7, Feb. 2019

항공기 지상 이동 Fast-Time 시뮬레이터 개발

Development of Fast-Time Simulator for Aircraft Surface Operation

김 태 영 · 박 배 선 · 이 현 웅 · 이 학 태*
인하대학교 항공우주공학과

Tae Young Kim · Bae-Seon Park · Hywonwoong Lee · Hak-Tae Lee
Department of Aerospace Engineering, Inha University, Incheon, 100, Korea

[요 약]

본 연구에서는 항공기 지상 이동 fast-time 시뮬레이터를 개발하였다. 시뮬레이터는 FCFS (first-come first-served) 스케줄러로부터 생성된 스케줄을 사용하여 항공기를 지상 이동시키는데, 항공기의 움직임을 모사하기 위해 1차원 등속 이동 운동 모델을 적용하였다. 공항 내 항공기 충돌 위험이 발생하는 상황을 분석하여 총 6개의 상황으로 분류하였으며 충돌 감지 및 회피 알고리즘을 구현하여 분리 거리를 유지하고 교착 상태를 방지하도록 하였다. 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 위험 상황이 없어지는 대신 추가적인 지연이 발생함을 확인하였다. 또한 회피 알고리즘에서 3가지 통행 우선순위 부여 방식을 구현하여 각 방식에 따른 지연 대수와 평균 지연 시간을 비교하였다. 남은 거리 또는 남은 시간에 따라 우선 순위를 부여하는 방식이 각 상황별 최소 지연을 선택하는 방식에 비하여 전체 추가 지연이 작아짐을 확인하였다.

[Abstract]

This study presents the development of a fast-time airport surface simulator. The simulator uses the output from a first-come first-served (FCFS) scheduler and has adopted one-dimensional dynamic model to simulate the movement of the aircraft on the surface. Higher collision risks situations in the airport surface traffic are analyzed to classify those situations into six cases. A conflict detection and resolution algorithm is implemented to maintain separation distance and to prevent deadlock. The simulator was tested with a scenario at the Incheon International Airport that contains 72 aircraft. Without the conflict detection and resolution, various conflict situations are identified. When the conflict detection and resolution algorithm is managing the traffic, it is confirmed that the conflicts are removed at the price of additional delays. In the conflict resolution algorithm, three prioritization strategies are implemented, and delayed aircraft count and average additional delays are compared. Prioritization based on remaining time or distance showed smaller total additional delay compared to choosing minimum delay priority for each situation.

Key word : Fast-time simulator, Surface operation, Separation algorithm, Conflict detection and resolution.

https://doi.org/10.12673/jant.2019.23.1.1

Received 28 December 2018; **Revised** 28 January 2019
Accepted (Publication) 27 February 2019 (28 February 2019)

***Corresponding Author; Hak-Tae Lee**

Tel: +82-32-860-8656
E-mail: tae0719@gmail.com

Ⅰ. 서 론

매년 항공 교통량이 크게 증가함에 따라 공항 내에서 항공기의 움직임이 복잡해지고, 출도착 지연이 빈번하게 발생하고 있다. 이러한 문제를 해결하기 위한 방안으로 출도착 통합관리 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 시뮬레이터가 필요하다. 일반적으로 스케줄링 알고리즘은 상대적으로 단순화된 지상의 모델을 이용해서 스케줄링 결과를 도출하게 되고, 이 결과를 지상 이동 시뮬레이션을 통해 검증하게 된다. 또한 지상 이동 시뮬레이터를 통해 항공기의 지상 궤적을 예측하고 이를 다시 스케줄링에 반영할 수도 있다. 미국 NASA에서는 SARDA (spot and runway departure advisor) 라는 출발 관리 시스템을 개발하여[1] 미국 내의 몇몇 주요 공항에서 테스트를 진행 중에 있으며, 이를 위해 필요한 fast-time 시뮬레이터인 SOSS (surface operations simulator and scheduler)[4] 를 개발하여 사용하고 있다.

본 논문에서는 기존의 FCFS (first-come first-served) 스케줄러와 연동이 가능한 fast-time 시뮬레이터에 대해 설명한다. Fast-time 시뮬레이션은 실시간에 대비되는 개념으로 일반적으로 실시간보다 빨리 진행되며, 따라서 중간에 사람의 개입이 없다. NASA에서 개발한 SOSS와 유사하지만 국내에서 처음으로 항공기 분리 유지 기능이 적용된 시뮬레이터이다. 시뮬레이터는 주어진 스케줄을 최대한 준수하도록 지상에서 항공기를 이동 시키지만, 자체적으로 분리 유지 알고리즘을 가지고 있어서 필요한 경우 지연을 발생시킬 수 있다. 분리 유지 알고리즘은 같은 유도로 내에서의 전후 분리 간격 유지시키고 교차로에서의 통행 우선순위를 부여하는 기능을 포함하고 있다.

지상이동 시뮬레이터는 인천공항의 실제 운용기록을 바탕으로 개발된 72대의 항공기를 포함하는 시나리오를 이용하여 모든 기능이 작동하는지를 시험하였고, 4가지의 교차로 통과 우선순위 부여 방식에 대한 결과를 비교하였다.

II장에서는 전반적인 시뮬레이터의 구성에 대하여 설명하고, III장에서는 분리 유지 알고리즘을 설명한다. IV 장에서 테스트 결과를 종합하고, V 장에서 결론 및 향후 계획을 제시하였다.

Ⅱ. 시뮬레이터 구성

2-1 항공기 이동 모델

$$x_{ac} = (x_2 - x_1) * \frac{t_{current} - t_1}{t_2 - t_1} + x_1 \quad (1)$$

$$y_{ac} = (y_2 - y_1) * \frac{t_{current} - t_1}{t_2 - t_1} + y_1 \quad (2)$$

x_{ac}, y_{ac} : 항공기의 x축, y축 좌표
x_1, y_1, x_2, y_2 : 링크의 두 노드의 x축, y축 좌표
t_1, t_2 : 링크 진입 시간과 탈출 시간
$t_{current}$: 시뮬레이션 상의 현재 시간

본 시뮬레이터에는 1차원 질점 항공기 모델[5]이 적용되어있다. 항공기는 2차원의 공항 지상에서 이동하지만 경로가 지정되어 있어 링크 상에서만 이동한다고 가정할 수 있으므로 1차원 상에서 이동하는 것과 같다. 또한 링크 내에서 등속도로 이동하며, 속도 변화는 노드에서 계단 함수 형태로 이루어진다고 가정하였다.

항공기의 위치를 업데이트하기 위해 내분점 공식을 사용하였으며 이는 항공기의 무게중심의 위치이다. 현재 시간과 경로상의 링크 진출입 시간으로부터 항공기가 이동 중인 링크의 두 노드와 좌표를 알 수 있으며, 이는 그림 1에서 'Node 1'과 'Node 2'로 표시되어 있다. 두 노드의 좌표와 각 노드의 도착 시간, 현재 시간 정보를 식 (1)과 (2)에 적용하여 현재 시간에서의 위치를 알 수 있다.

2-2 입력 데이터

시뮬레이터 구동을 위해 노드-링크 모델과, 스케줄 결과가 필요하다. 노드 데이터는 노드의 종류와 ID, 위경도 좌표로 이루어져 있으며, 링크 데이터는 링크의 타입과 ID, 링크를 구성하는 두 노드의 ID로 이루어져 있다.

스케줄 결과는 항공기의 편 명, wake turbulence category, 시작과 도착 노드 명, 경유한 링크 명, 각 링크의 진출입 시간 등으로 구성되어 있다. 시뮬레이션을 위한 초기화 과정에서는, 항공기를 생성한 후 경유한 노드와 각 노드 도착 시간을 저장한다.

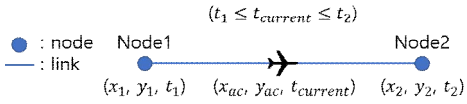


그림 1. 내분점 공식을 이용한 항공기의 위치 업데이트
Fig. 1. Aircraft position update using linear interpolation.

* Nodes in RKSI			
* 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

(a) Node model file format

* Links in RKSI			
* 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

(b) Link model file format

그림 2. 노드-링크 모델 예시
Fig. 2. Example of node-link model.

* Flight ID	Address	Type	Wake Category	State	Entry Time	Exit Time	Previous Link	Current Link	Next Link	Speed-Up	Slow-Down
KAL854	HL8001	A333	H	ARR	1980	1990	XXXX	33R	Rwy7	0	0
KAL854	HL8001	A333	H	ARR	1990	1997	Rwy7	Rwy6	Rwy5	0	0
KAL854	HL8001	A333	H	ARR	1997	2004	Rwy6	Rwy5	Rwy4	0	0
KAL854	HL8001	A333	H	ARR	2004	2049	Rwy5	Rwy4	Tx15	0	0
KAL854	HL8001	A333	H	ARR	2049	2097	Rwy4	Tx15	Tx23	0	0

그림 3. 스케줄 결과의 예시
Fig. 3. Example of surface schedule.

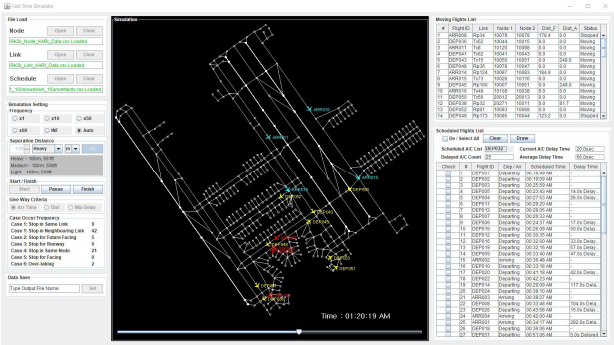


그림 4. 실행중인 시뮬레이터 화면
Fig. 4. Screen of simulator in operation.

2-3 User Interface 화면 구성

본 프로그램은 5개의 모듈로 구성되어 있는 설정 모듈과 이동 중인 항공기의 정보 확인을 위한 2가지 모듈로 구성된 시뮬레이션 모듈, 결과를 확인하기 위한 1개의 모듈, 총 8개의 모듈로 구성되어 있다.

설정 모듈은 파일 입력과 시뮬레이션의 시작/종료를 담당하면서, 배속, 분리 간격, 항공기 통행 우선권 등을 설정할 수 있도록 구성되어 있다. 분리 간격은 항공기의 wake turbulence category에 따라 다르게 설정할 수 있다. 시뮬레이션 모듈에서는 항공기의 위치를 가시화 하고, 항공기 목록에서 진행상황을 보여준다. 충돌 위험이 있는 항공기들은 색이 변하도록 되어있고 각각의 위험 상황에 대한 통계를 계산한다.

Ⅲ. 항공기 분리 유지 알고리즘

지상 이동 시 항공기간의 충돌 방지를 위한 분리 간격은 항공기 무게 중심을 기준으로 계산한다. 항공기의 충돌 위험이 발생하는 경우는 기본적으로 전후 간격 미준수와 교차로에 다수의 항공기가 접근하는 경우가 있다. 항공기의 충돌 위험 상황을 4가지로 분류하고, 3가지 정지 및 재출발 알고리즘을 적용하였다.

3-1 항공기간 거리 계산

항공기 충돌 방지를 위해 두 항공기간의 거리 계산이 필요하다. 서로 이웃하지 않은 링크에서 이동 중인 항공기간에는 충돌 위험이 없다고 가정하였으며, 이웃한 링크 상의 항공기간의 거리만 계산하였다. 또한 두 항공기간의 거리는 실제거리가 아닌, 링크를 1차원 상에 펼쳐놓았을 때의 거리로 정의하였으며, 그림 5에 설명되어 있다.

3-2 Case 1: 동일 선상

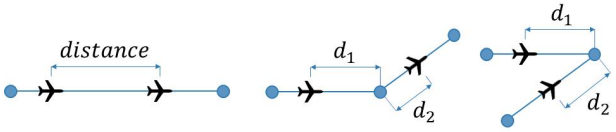
두 항공기의 진행 방향이 일치하여 전후 구분이 명확한 경우이며, 두 대의 항공기가 동일 링크에서 진행 중인 경우와, 이웃한 두 링크에 각각 한 대의 항공기가 진행 중인 경우로 나누어 볼 수 있다.

1) 동일 링크

그림 6은 두 항공기가 동일 링크 내에서 이동하며 진행 방향이 일치하는 경우에서 분리 거리 유지를 위해 항공기가 정지하는 것을 보여준다. 두 항공기의 거리가 분리 간격보다 가까운 경우, 후행 항공기를 정지시키고, 거리가 멀어져 분리 간격이 준수될 때, 후행 항공기를 재출발 시킨다.

2) 이웃 링크

그림 7은 두 항공기가 진행 중인 링크가 서로 다르지만 한 항공기의 출발 노드와, 다른 항공기의 목표 노드가 일치하여 두 항공기가 동일 선상에 있다고 볼 수 있는 경우에 분리 거리 유지를 위해 항공기가 정지하는 것을 보여준다. 두 항공기의 거리가 분리 간격보다 가까운 경우, 후행 항공기를 정지시키고, 거리가 멀어져 분리 간격이 준수될 때, 후행 항공기를 재출발 시킨다.



$$distance = d_1 + d_2$$

그림 5. 항공기간 거리 계산 방식
Fig. 5. Calculation of distance between two aircraft.

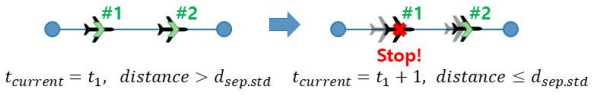


그림 6. 동일 링크 내에서 충돌 방지
Fig. 6. Collision avoidance in same link.

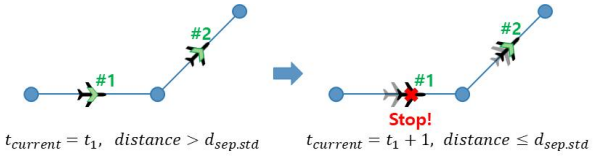


그림 7. 이웃한 링크에서 충돌 방지
Fig. 7. Collision avoidance in neighbouring link.

3-3 Case 2: 사용 중인 다음 링크

그림 8은 두 항공기가 동일 노드로 향하며, 2번 항공기의 진행 예정 링크에서 1번 항공기가 마주보는 방향으로 접근하는 상황을 보여준다. 2번 항공기가 교차로에 먼저 진입하게 될 경우, 1번 항공기와 마주보게 되어 두 항공기 모두 이동하지 못한다. 이러한 상황을 사전에 방지하기 위해 1번 항공기를 선행 항공기로 지정하고, 교차로를 빠져나간 후 분리 간격이 유지될 때 2번 항공기를 재출발시킨다. 이 때, 2번 항공기를 정지하는 기준은 이전 상황과 다르다. 두 항공기간의 거리에 무관하게 교차로로부터 2번 항공기까지의 거리가 분리거리기준의 절반보다 가까워졌을 때 정지하도록 하였다.

3-4 Case 3: 활주로 양보

그림 9는 두 항공기가 동일 노드로 향하는 상황을 볼 수 있으며, 이때 1번 항공기는 활주로에서 이착륙중인 항공기이다. 2번 항공기의 경로 계획 등 모든 상황에 무관하게 1번 항공기가 우선권을 부여받아 선행 항공기로 지정되어야 하며, 교차로를 먼저 통과하여야 한다. 2번 항공기가 정지하게 되는 기준은 Case 2와 동일하다.

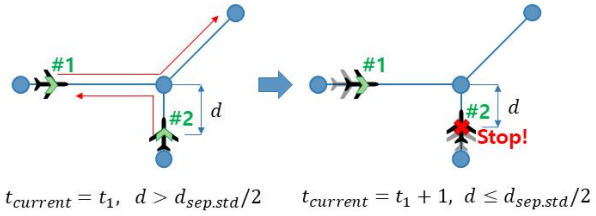


그림 8. 다음 링크를 고려하여 교차로 통과 우선순위 선정
Fig. 8. Prioritization based on downstream link.

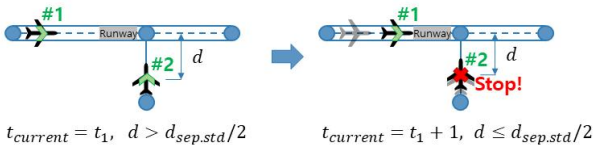


그림 9. 활주로에서 이동 중인 항공기가 교차로 우선 통과
Fig. 9. Aircraft operating on runways have higher priority.

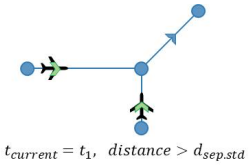


그림 10. 동일 노드로 접근중인 항공기
Fig. 10. Two aircraft approaching to same node.

3-5 Case 4: 교차로 접근

그림 10은 두 항공기가 서로 다른 링크에서 진행 중이며 동일 노드로 향하는 상황을 볼 수 있으며, 교차로에 다수의 항공기가 접근하는 일반적인 상황이다. 선행 항공기를 구별하는 기준이 모호하며 사용자가 기준을 선택할 수 있도록 3가지 기준을 제시하였다.

1) 남은 거리 기준

그림 11은 두 항공기의 거리가 분리 간격보다 가까워졌을 때, 노드까지의 거리를 기준으로 선행 항공기를 지정하는 것을 보여준다. 두 항공기 간의 분리 거리가 기준을 위반하는 순간에 2번 항공기가 1번 항공기보다 노드로부터 더 가까이 있어 교차로를 먼저 통과하게 되며, 1번 항공기는 정지한다.

2) 남은 시간 기준

그림 12는 두 항공기의 거리가 분리 간격보다 가까워졌을 때, 노드에 예상 도착 시간을 기준으로 선행 항공기를 지정하는 것을 보여준다. 1번 항공기의 노드 도착 예정 시간이 2번 항공기의 도착 예정 시간보다 이른 경우 2번 항공기가 교차로에 먼저 진입하게 되며, 1번 항공기는 정지한다.

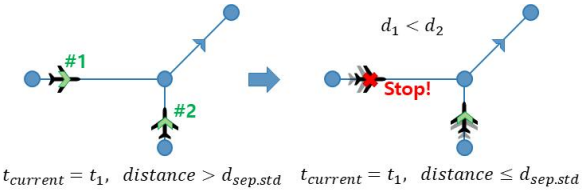


그림 11. 노드로부터 가까이 있는 항공기 우선 통과
Fig. 11. Closest aircraft from node with higher priority.

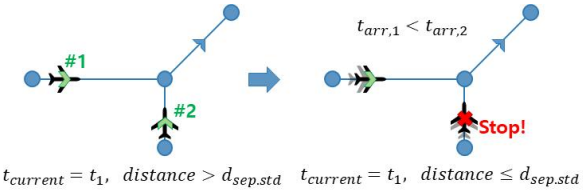


그림 12. 노드 도착 예정 시간이 이른 항공기 우선 통과
Fig. 12. Earliest time of arrival aircraft with higher priority.

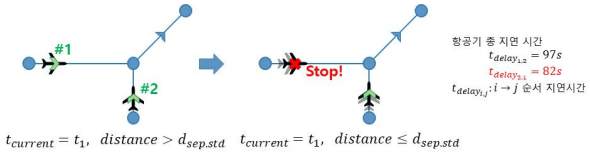


그림 13. 최소 지연 시간 조합의 순서대로 항공기 통과
Fig. 13. Aircraft pass junction with order of combination of shortest delay time.

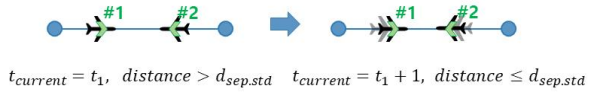


그림 14. 같은 링크에서 두 항공기가 마주보는 상황
Fig. 14. Two aircraft confront in same link

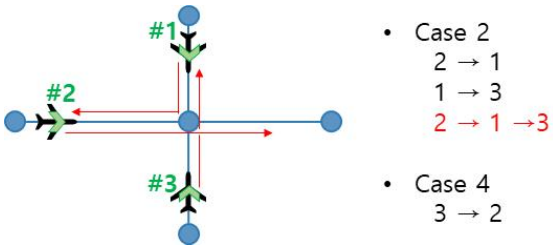


그림 15. 교차점에서 모든 항공기 정지 상황
Fig. 15. All aircraft stop at the junction.

3) 최소 지연 시간 기준

그림 13은 교차로에 진입한 항공기에 우선권을 부여할 수 있는 모든 경우의 수에 대해 총 지연 시간을 계산한 후, 지연 시간이 가장 짧은 경우의 순서대로 교차로를 통과시키도록 하는 것을 보여준다. 그림 13에서 1번 항공기가 먼저 통과할 경우 총 지연시간이 97초로 계산되며, 2번 항공기가 먼저 통과할 경우 총 지연시간이 82초로 계산될 때, 총 지연시간이 더 짧은 두 번째 경우의 순서에 따라 2번 항공기를 선행 항공기로 지정하여 먼저 통과하도록 한다.

3-6 Case 5: 마주보며 접근

항공기가 마주보는 방향으로 동일 링크에 진입하는 상황이 존재해서는 안 된다. 스케줄링 알고리즘에서 이를 고려하여 스케줄을 계산하기 때문에 현재 fast-time 시뮬레이터에는 이 경우를 고려한 정지, 재출발 하는 기능은 구현되지 않은 상태이다. 이 상황이 발생할 시, 서로를 인지하지 못한 체 그대로 진행하게 되며 사용자에게 알림을 준다.

3-7 예외 상황: 교착 상태

그림 15에서는 3대의 항공기가 교차로에 접근중인 상황에서 각 항공기의 경로 계획을 볼 수 있으며, 앞서 설명한 Case 2와 Case 4가 복합적으로 발생하는 것을 확인할 수 있다. Case 2는 항공기가 이동 중인 링크에 다른 항공기가 반대 방향으로 진입하는 것을 사전에 방지하기 위한 것으로 이에 따라 2번, 1번, 3번 항공기 순으로 교차로를 통과하도록 한다. 하지만 Case 4에 의해 3번 항공기가 2번 항공기보다 높은 우선순위를 부여받은 경우, 모든 항공기가 차선 순위로 밀려 결국 정지하여 교착 상태가 발생한다. Case 1, 2, 3은 항공기 충돌을 방지하기 위한 알고리즘으로 위반 시 항공기 충돌을 초래할 수 있으며, Case 4는 항공기의 이동을 효율적으로 관리하기 위한 알고리즘이다. 따라서 이러한 상황에서는 Case 1, 2, 3을 위반하지 않는 조건하에 우선순위가 가장 높은 항공기가 통과하도록 하였으며, 최종적으로 교차로 통과 순서는 2번, 1번, 3번 항공기 순서로 진행하게 된다.

Ⅳ. 결 과

시뮬레이션을 수행한 테스트 케이스는 인천국제공항에서 약 1시간 10분 동안 총 72대의 항공기가 이착륙하는 스케줄이며, 분리 간격은 180 m로 설정하였다.

항공기 정지 기능을 비 활성화하여 동일한 스케줄로 시뮬레이션 한 결과 후행 항공기의 분리 간격이 유지되지 못하는 횟수가 59회 발생하였으며, 교차로로 다수의 항공기가 접근하는 경우도 14회 발생하였다. 항공기가 마주 오는 상황이 2회 발생하였으며, 모두 게이트에서 빠져나오는 항공기와 진입하는 항공기가 마주 오는 상황이었다. 실제 운용에 있어서 계류장은 움직임이 보다 자유롭기 때문에 실제 상황에서는 우회가 가능할 것으로 예상되나, 추후에는 시뮬레이터에서도 이 부분을 처리할 수 있는 기능을 추가할 예정이다. 충돌 위험이 발생하는 5가지 경우에 대한 발생 횟수는 아래의 표 1에 정리되어 있다.

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 지연이 더 크게 됨을 볼 수 있는데, 이는 각 노드에서의 지연을 최소화 하는 것이 전체 시스템의 지연을 최소화 하지 못할 수도 있음을 보여준다.

표 1. 항공기 정지 기능 미적용 시 분리 거리 유지 실패 횟수
Table 1. Frequency of failure to maintain separation distance with aircraft pause function disabled.

Cases of link separation failure	frequency
Case 1: Aircraft on the same link	18
Case 1: Aircraft on the neighbouring link	41
Case 2: Future link occupied by another aircraft	5
Case 3: Runway priority	1
Case 4: Aircraft meet at the junction	14
Case 5: Aircraft confront on the same link	2

표 2. 정지 알고리즘 별 지연 항공기 대수 및 평균 지연 시간
Table 2. Delayed aircraft count and average delay time of each pause algorithm

Pause algorithm	Delayed aircraft count	Average delay time (sec)
Remaining distance	28	242
Remaining time	25	224
Minimum delay	24	269

Ⅴ. 결 론

본 연구에서는 공항에서의 항공기 지상 이동을 모사할 수 있는 fast-time 시뮬레이터의 개발에 대하여 기술하고, 사용된 분리 유지 알고리즘을 설명하였다. 개발된 시뮬레이터는 인천국제공항에서의 72대의 항공기의 출도착 스케줄을 이용하여 테스트 되었으며 우선순위 설정 방식에 따른 다른 결과가 나타남을 확인하였다.

현재 개발된 항공기 모델은 1차원 질점 모델로써, 계단 함수 형태의 정지와 가속이 가능하며 링크 위에 놓여서 정해진 경로를 따라 이동할 수 있다. 하지만 실제에는 항공기가 링크의 중심축을 기준으로 좌우로 움직이며 2차원 기동을 한다. 또한 항공기의 가속 성능에 의해 스케줄링 된 진출입 시간을 준수하지 못할 수 있다. 추후 2차원 점 질량 항공기 모델을 개발, 적용하여 보다 현실적인 시뮬레이션을 통한 검증을 수행할 계획이다.

Acknowledgments

본 연구는 국토교통부의 '다목적 FCFS Multi-Point 스케줄링 연구 (19ATRP-C088159-06)'에 의하여 이루어진 연구로써, 관계부처에 감사드립니다.

References

[1] Y. Jung, W. Malik, L. Tobias, G. Gupta, T. Hoang, and M. Hayashi. "Performance evaluation of SARDA: An individual aircraft-based advisory concept for surface management," *Air Traffic Control Quarterly*, Vol. 22, No. 3, 2015, pp.195-221, April, 2015.

[2] Y. J. Eun, D. K. Jeon, H. B. Lee, Y. C. Jung, Z. Zhu, M.S. Jeong, H. K. Kim, E. M. Oh, and S. K. Hong. "Optimization of airport surface traffic: A case-study of Incheon International Airport," in *17th AIAA Aviation Technology, Integration, and Operations Conference, AIAA AVIATION Forum*, Denver: Colorado, p. 4258, June 5-9, 2017.

[3] 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, June 2018.

[4] Z. Wood, S. Rathinam, M. Kistler, and Y. Jung, "A simulator for modeling aircraft surface operations at airports," in AIAA Modeling and Simulation Technologies Conference, Chicago: IL, August 10-13, 2009.

[5] D. Wu, Y. J. Zhao, and B. Capozzi, "Fundamental surface trajectory models for air traffic automation," in *10th AIAA Aviation Technology, Integration, and Operation (ATIO) Conference* , Fort Worth: TX, September 13-15, 2010.

김 태 영 (Tae Young Kim)
2017년 2월 : 인하대학교 항공우주공학과 (공학사)
2017년 3월 ~ 현재 : 인하대학교 항공우주공학과 석사과정
※관심분야 : 항공교통, M&S

박 배 선 (Bae-Seon Park)
2014년 2월 : 인하대학교 항공우주공학과 (공학사)
2014년 3월 ~ 현재 : 인하대학교 항공우주공학과 통합과정
※관심분야 : 항공교통, M&S

이 현 웅 (Hyeonwoong Lee)
2016년 2월 : 인하대학교 항공우주공학과 (공학사)
2018년 8월 : 인하대학교 항공우주공학과 (공학석사)
2018년 9월 ~ 현재 : 인하대학교 항공우주공학과 박사과정
※관심분야 : 항공교통, M&S

이 학 태 (Hak-Tae Lee)
2000년 12월 : 미국 스탠포드대학교 항공우주공학과 (공학석사)
2006년 1월 : 미국 스탠포드대학교 항공우주공학과 (공학박사)
2013년 9월 ~ 현재 : 인하대학교 항공우주공학과 부교수
※관심분야 : 항공교통, 공탄성, 전산유체