2014년 11월 전자공학회 논문지 제 51권 제 11 호
Journal of The Institute of Electronics and Information Engineers Vol. 51, NO. 11, November 2014 http://dx.doi.org/10.5573/ieie.2014.51.11.117

논문 2014-51-11-15

항공기 데이터블록의 효율적 비중첩 재배치 알고리즘

(Efficient Non-overlapping Aircraft Datablock Relocation Algorithm)

정 재 협*, 원 인 수*, 양 훈 준*, 정 동 석**

(Jae Hyup Jeong©, In Su Won, Hun Jun Yang, and Dong Seok Jeong)

요 약

본 논문에서는 항공기의 데이터블록이 중첩되었을 때 효율적으로 재배치하는 알고리즘을 제안한다. 관제화면에서 항공기 정보를 의미하는 데이터블록이 중첩되면 관제하는데 어려움을 주므로 중첩되지 않게 재배치하는 것이 필요하다. 본 논문에서 제안하는 재배치 알고리즘에서는 데이터블록의 특성을 고려한 최소이동을 이용한다. 최소이동의 이동거리는 중첩 시 생기는 직사각형의 높이 또는 너비를 사용하여 계산되며, 최소이동의 이동방향은 데이터블록의 방향성을 고려하여 계산된다. 또한 항적심벌과 데이터블록의 사이가 멀어진 특정한 상황에서는 기존의 알고리즘을 사용한 재배치가 이루어진다. 제안하는 알고리즘은 데이터블록의 여러 상황을 고려하였으므로 기존의 알고리즘보다 우수한 성능을 보였다.

Abstract

In this paper, we propose an efficient algorithm which can relocate the datablock of an aircraft when it is overlapped. If the datablock which represents the aircraft information in the control display is overlapped, relocation without overlapping is necessary because it is difficult to control the air traffic in this situation. The proposed algorithm relocates the data block with minimum movement by considering the characteristics of datablock. The moving distance of minimum movement is calculated using the height or width of rectangle which is created during overlapping. And the moving direction of minimum movement is calculated by considering the directivity of the datablock. When the distance between the target symbol and datablock is distant enough, the relocation is carried out using the existing algorithm as a special case. The proposed algorithm shows improved performance in comparison with the existing algorithm due to the fact that it considers many different cases of the datablock.

Keywords : rectangle separation, rectangle overlapping, ATC(air traffic control)

Ⅰ. 서 론

항공관제의 목적은 관제사가 항공기를 안전하게 출발지로부터 목적지로 도착하게 하는 것이다. 관제사들은 항공상황을 보여주는 현시시스템으로부터 각자 배정된 섹터그룹에서 항공기를 관제하며, 각자의 관제 범위 내에서 항공기를 안전하게 관제할 책임이 있다[1]. 현재 국내 항공교통량은 국제선의 경우 항공사 운항편 확대, 한국 드라마 인기에 힘입은 외국인 방한수요 확대, 원화 강세로 인한 내국인 해외 관광 등으로 증가하고 있다. 2014년 1/4분기 실적은 전체여객이 1,860만명으로 전년동기대비 국제선 13.8%, 국내선 5.9%로 매년 증가하는 추세이다[2]. 이처럼 매년 항공교통량의 증가로 인해 항공관제를 효율적으로 하는 현시시스템의 필요성이 대두되고 있다. 항공관제시스템의 국산화사업으로 의해 새로운 관제시스템이 개발되었으며[3], 그 중 현시시스템(CWP : Controller Working Position)은 관제사가 직접

* 학생회원, ** 정회원, 인하대학교 전자공학과
(Dept. of Electronic Engineering, Inha University)
© Corresponding Author(E-mail: great_jhyup@hanmail.net)
※ 이 논문은 인하대학교 교내학술연구비의 지원을 받아 수행된 연구임
접수일자: 2014년08월04일, 수정일자: 2014년10월14일
게재확정: 2014년10월28일

화면을 보고 관제하는 시스템으로 보다 쉽고 편리하게 관제를 하는 것을 목적으로 개발되었다[4].

항공교통량이 많아졌을 때, 현시시스템의 효과적인 관제 방법은 성능면과 기능면으로 나눠 볼 수 있다. 먼저, 성능면에서는 시스템과 항적 데이터간의 표준 레이더 감시자료 포맷인 ASTERIX를 효과적으로 파싱하여 현시시스템의 부하를 줄여 시스템을 원활하게 하는 것이다. 이 때 사용된 방법이 패턴매칭으로 파싱에 소모되는 오버헤드를 최소화하여 시스템 처리 지연을 줄이는 방법이다[5]. 다음으로 기능면에서는 관제사가 관제화면에서의 항공기 수와 상관없이 각각의 항공기 정보를 정확하게 보는 편의를 제공하는 것이다. 항공교통량이 많아지면 관제화면에는 수많은 항공기 정보가 현시되므로, 각 정보들이 서로 중첩되는 현상이 발생한다. 이럴 경우 항공기의 정보를 제대로 볼 수 없으므로 일일이 중첩된 항공기 정보를 분리해야하는 작업이 필요하다. 관제사들은 항공기 정보에 포함된 속도, 고도, 경고 및 경보표시 등을 수시로 체크하며 관제하는데 집중해야 하는데, 이러한 추가적인 작업은 관제사들에게 큰 불편함을 제공한다.

본 논문에서는 중첩없는 효율적인 항공기 정보 재배치 알고리즘을 제시하여, 관제사에게 편의를 제공하고자 한다. 항공기의 수가 많아져도 각 항공기 정보를 중첩없이 정확히 보는 것을 목적으로 한다. 논문은 총 5장으로 구성되며, Ⅱ장에서는 현시시스템의 항공기 구성요소의 형태와 중첩에 대한 설명, Ⅲ장에서는 기존의 직사각형 배치 알고리즘에 대한 설명과 제안한 알고리즘에 대한 설명, Ⅳ장에서는 성능평가를 수행하고 실험결과를 도시한다. 마지막으로 Ⅴ장에서는 결론에 대해 기술한다.

Ⅱ. 현시시스템의 항공기 표현

1. 항공기 구성 요소의 형태

현시시스템의 관제화면에서 표현되는 항공기는 그림 1과 같이 항적심벌과 리더라인, 데이터블록으로 구성되어 있다. 항적심벌은 레이더로부터 받은 항공기의 현재 위치를 보여준다. 항공기의 위치는 레이더의 다중 센서 자료 처리를 통해 갱신된다[6]. 데이터블록은 항공기 정보를 나타내는 직사각형형태의 정보 그룹이다. 데이터블록의 내용은 도착 활주로, 항공기 Callsign, 현재 관제

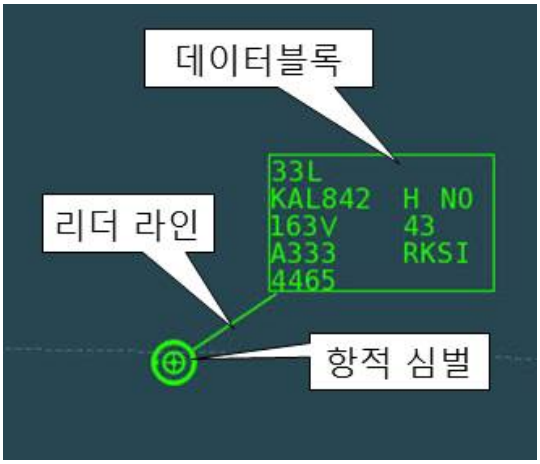


그림 1. 항공기 구성 요소
Fig. 1. Component of aircraft.

섹터, 고도, 속도, 도착공항, 식별 코드 등으로 구성되어 있으며 각 항공기마다 받는 정보의 양이 다르므로 데이터블록의 크기는 변할 수 있다. 마지막으로 리더라인은 항적심벌과 데이터블록을 연결하는 선이다. 데이터블록은 항적심벌과 같이 항공기의 진행 방향과 크기만큼 이동하며, 관제사가 선택하여 원하는 곳으로 수동 배치할 수도 있다. 이에 따라 리더라인의 길이도 변한다.

2. 항공기 구성 요소의 중첩

항공기 구성 요소별로 중첩에 대해 살펴보면, 먼저 항적심벌의 경우 모든 항공기는 각각 정해진 항로와 시간에 따라 비행하므로 항적심벌끼리 중첩되는 현상은 발생할 수 없다. 데이터블록은 항적심벌에 따라 이동하

그림 2. 중첩된 데이터블록의 예
Fig. 2. Example of Datablock overlapping.

므로 다른 데이터블록과의 중첩이 자주 일어난다. 리더라인 역시 자주 일어나나 단순한 선의 중첩이므로 관제에 영향을 주지 않는다.

그림 2는 항공기가 밀접한 곳에서 데이터블록의 중첩이 일어난 예를 보여준다. 여러 항공기들의 데이터블록이 중첩되어 항공기 정보를 제대로 읽을 수 없다. 이 경우 관제사가 일일이 데이터블록을 선택하여 수동으로 재배치해야하는 불편함이 생긴다. 또한, 재배치한 데이터블록이 이후에 다른 데이터블록과 중첩될 수도 있다. 그러므로 데이터블록의 재배치는 현재 그리고 이후에도 중첩없이 자동으로 이루어져야 한다.

마지막으로 데이터블록은 관제화면에서 가장 상위에 위치해 있기 때문에 지도의 기타 그래픽이나 항적심벌 등과 겹친다고 해도 잘 보이므로, 데이터블록과 다른 그래픽간의 중첩은 관제에 지장을 주지 않는다.

Ⅲ. 데이터블록 재배치 알고리즘

데이터블록은 직사각형 모양을 가진 실시간으로 이동하는 벡터이다(약4초마다 이동). 데이터블록의 효율적인 재배치를 위해서는 직사각형이라는 점과 실시간 벡터라는 점을 모두 고려해야 한다. 본 논문에서는 이러한 특징들을 모두 고려하여 데이터블록 재배치에 효율적인 알고리즘을 제안한다.

Ⅲ장에서는 기존의 직사각형을 중첩없이 배치하는 알고리즘인 LFR 알고리즘[7]을 먼저 설명하고, 제안한 알고리즘을 설명한다.

1. LFR 알고리즘

LFR(Largest Free Rectangle) 알고리즘은 정해진 영역 안에서 하나의 직사각형을 배치한 다음 나머지 직사각형들을 차례대로 중첩없이 배치하는 알고리즘이다.

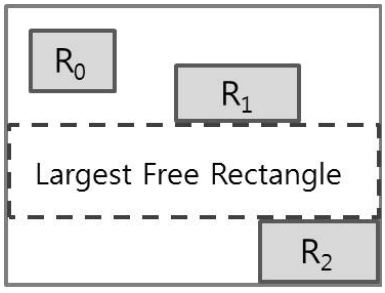


그림 3. LFR 알고리즘의 예
Fig. 3. Example of LFR Algorithm.

```
    LFR_Algorithm(idx)
 1   if idx is 0 then, R0 is located in Space.
 2   else
 3     for i=1 to Num do
 4       FRi is maximum (FRi^r, FRi^b, FRi^l, FRi^t)
 5       LFR is maximum (FRi)
 6     end
 7     Ridx is located in LFR
 8   end
 9   Num = Num+1
10  end LFR_Algorithm
```

그림 4. LFR 알고리즘의 의사코드
Fig. 4. Psuedo code for LFR Algorithm.

LFR 알고리즘에서는 LFR을 직사각형이 배치될 때마다 새롭게 구하는 것이 가장 중요하다. 그림 3에서처럼 배치된 직사각형들을 제외한 영역에서 가장 큰 여백 직사각형 LFR을 구한다. LFR을 구하는 방법은 아래 식 (1)과 같으며, 그림 4는 LFR 알고리즘의 의사코드를 나타낸다.

$$FR_i = \max(FR_i^r, FR_i^b, FR_i^l, FR_i^t)$$
$$LFR = \max(FR_i)_{(i=0,\ldots,n)} \quad (1)$$

먼저, 배치된 하나의 직사각형 R_0를 중심으로 우하좌상의 위치 순서대로 4개의 여백 직사각형(FR_0^r, FR_0^b, FR_0^l, FR_0^t)을 구한다. 그 중 가장 여백이 큰 직사각형이 FR_0이 되고, LFR이 된다. 그 다음 직사각형 R_1을 LFR 안에 배치하고, 새롭게 R_0를 중심으로 한 4개의 여백 직사각형과 R_1을 중심으로 한 4개의 여백 직사각형을 구한다. 각각 가장 여백이 큰 직사각형이 FR_0와 FR_1이 되고, 이 중 가장 여백이 큰 직사각형이 LFR이 되어 이곳에 R_2를 배치한다. 이와 같이 직사각형이 배치될 때마다 새롭게 LFR을 구해서 그 안에 직사각형을 배치하는 것이 LFR 알고리즘이다.

2. 효율적인 데이터블록 재배치 알고리즘

LFR 알고리즘은 데이터블록 배치 시에는 효율적인 알고리즘이라고 할 수 없다. 그 이유는 먼저 데이터블록은 실시간 벡터이기 때문이다. 적절한 공간에 중첩없이 재배치를 한다고 해도 데이터블록은 계속 움직이므로 이후에 다시 중첩이 생겨 재배치를 해야하는 문제가 발생한다. 다음으로 데이터블록은 항적심벌과의 거리가

적당해야 하기 때문이다. 이후에도 중첩이 없을 넓은 공간에 데이터블록은 배치하면 좋겠지만, 이럴 경우 항적심벌과의 거리가 너무 멀어지게 될 수 있다. 이렇게 되면 관제사가 해당 항공기 위치와 정보를 바로 보기가 어려워 관제에 혼란을 준다. 그러므로 이 두 가지 특성을 잘 고려하여 데이터블록 재배치에 특화된 알고리즘을 제안하고자 한다.

가. 중첩 직사각형을 고려한 최소이동

데이터블록이 중첩 시에는 재배치가 되어야한다. 여기서 재배치란 중첩된 데이터블록들이 중첩되지 않게 각각 이동해야 된다는 뜻이다. 이동은 최소한의 거리가 되어야 관제사에게 혼란을 주지 않을 수 있고, 이동한 위치에서는 다른 데이터블록과의 중첩이 일어나지 않아야 한다.

그림 5와 같이 데이터블록 A와 B가 중첩되면 중첩 직사각형이 생긴다. 최소이동을 하기 위해서는 이 중첩 직사각형의 높이(h_o) 또는 너비(w_o)를 고려하여, 데이터블록 A와 B 중 어느 데이터블록이 얼마만큼의 거리로 이동해야 되는지 적절히 판단해야 된다. 중첩된 데이터블록의 이동방향은 2가지로 좌우이동 또는 상하이동이다.

재배치를 위한 이동방향과 이동거리를 정해보면, 먼저 이동방향은 최소이동이 목적이므로 중첩 직사각형의 높이랑 너비 중 더 작은 값을 기준으로 상하이동 또는 좌우이동이 되어야 한다. 즉, 높이가 너비보다 작으면 상하이동이 되고 너비가 높이보다 작으면 좌우이동이 된다. 이어서 데이터블록들의 위치에 따라 좌우 또는 상하가 정해지는데, 그림 5에서는 A가 B보다 좌상으로 위치해 있으므로 좌우이동 시 A가 좌로 B가 우로, 상하이동 시 A가 상으로 B가 하로 이동된다. 다음으로

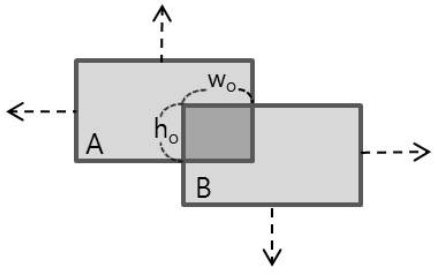


그림 5. 중첩 직사각형
Fig. 5. Overlapping rectangle.

표 1. 중첩에 의한 M_A와 M_B의 값
Table 1. The value of M_A and M_B by overlapping.

	M_A만 중첩	M_B만 중첩	M_A와 M_B 모두 중첩 / 중첩 없음
M_A	$\frac{V}{2}-O_A$	$\frac{V}{2}+O_B$	$\frac{V}{2}$
M_B	$\frac{V}{2}+O_A$	$\frac{V}{2}-O_B$	$\frac{V}{2}$

이동거리(V)는 A와 B의 이동거리의 합이 좌우이동 시에는 w_o, 상하이동 시에는 h_o만큼 되어야 한다.

$$V = \min(w_o, h_o)$$
$$M_A + M_B = V \quad (2)$$

A의 이동이 M_A, B의 이동이 M_B라고 했을 때, 식 (2)처럼 나타낼 수 있다. M_A와 M_B의 초기 값은 V/2이다. M_A와 M_B의 값은 V/2로 각각 이동 시에 생기는 다른 데이터블록과의 중첩여부에 따라 바뀔 수 있다.

표 1에서 O_A와 O_B는 M_A와 M_B로 각각 이동 시, 다른 데이터블록과의 중첩이 발생했을 때 생기는 중첩 직사각형의 높이 또는 너비이다. 예를 들어, A와 B가 V/2 크기로 좌우이동 시 A가 다른 데이터블록 C와 중첩이 발생하면 O_A는 A와 C간의 중첩 직사각형 너비가 된다. 이 경우 M_A는 초기 값 V/2에서 O_A만큼 빼준 값이 되며, M_B는 초기 값 V/2에서 O_A만큼 값이 커진다. 이렇게 되면, A와 C간의 또 다른 중첩이 발생되지 않은 상태에서 A와 B는 재배치되는 것이다. 만약 M_A와 M_B 이동 시 A와 B 모두 다른 데이터블록과 중첩이 발생하게 된다면, 표 1과 같이 중첩이 모두 발생하지 않을 경우와 마찬가지로 초기 값 V/2만큼만 이동한다. 재배치는 두 데이터블록씩 최소이동을 하여 모든 데이터블록이 중첩이 없을 때 까지 차례대로 이루어지기 때문에 이동 시에 새롭게 중첩이 발생하더라도 결국 모든 데이터블록은 중첩없는 재배치가 이루어진다.

나. 방향성을 고려한 재배치

앞에서 설명한대로만 재배치 할 경우, 최소이동으로 모든 데이터블록을 재배치할 수 있지만 이후에 데이터블록이 항적심벌에 따라 이동하게 되면 다시 중첩이 일어날 가능성이 높다. 이것을 최대한 방지하기 위해 데이터블록의 이동방향을 고려한 재배치가 이루어져야 한

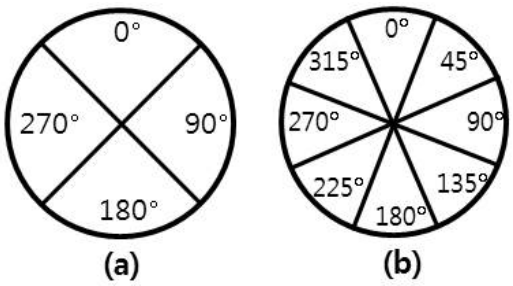


그림 6. 벡터 양자화
(a) 4방향 (b) 8방향
Fig. 6. Vector quantization
(a) 4 Direction (b) 8 Direction

다. 모든 데이터블록은 벡터이므로 방향성을 가진다. 그러므로 중첩된 데이터블록 간의 방향성을 판단하여 경우에 따라 다르게 재배치를 하면, 항공기 이동에 따른 이후의 중첩을 최대한 방지할 수 있다. 데이터블록 벡터의 방향은 0~360도로 다양하다. 이런 다양한 각도를 그대로 쓰면 너무 많은 경우의 수가 발생하기 때문에, 벡터 양자화가 필요하다.

그림 6은 4방향과 8방향의 벡터 양자화이다. 4방향 벡터 양자화는 90도 간격으로 8방향 벡터 양자화는 45도 간격으로 한다. 예를 들어, 데이터블록의 벡터 방향이 120도인 경우 4방향에서는 90도로 8방향에서는 135도로 양자화된다.

다음으로 양자화된 벡터 방향에 따른 중첩된 두 데이터블록의 경우의 수를 보면, 4방향의 경우에는 0, 90, 180도로 3가지 경우가 되고 8방향의 경우에는 0, 45, 90, 135, 180도로 5가지 경우가 된다. 여기서 다시 같은 방향으로 중첩이 되는 0도, 반대 방향으로 중첩이 되는 180도, 그리고 나머지 방향 중첩 각도 이렇게 3가지 경우로 나누어 각각 다르게 재배치를 적용한다.

(1) 같은 방향으로 중첩

같은 방향으로 중첩은 가장 빈번하게 일어나는 중첩이다. 모든 항공기는 정해진 항로에 따라 움직이며 항로간 거리가 가까운 경우를 제외하고 같은 방향으로의 중첩만 일어난다.

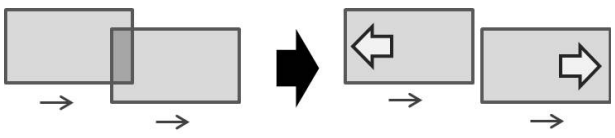


그림 7. 같은 벡터 방향의 재배치의 예
Fig. 7. Example of relocation of same vector direction.

그림 7은 같은 벡터 방향을 가진 데이터블록이 중첩되었을 때 재배치되는 예이다. 그림과 같이 같은 방향으로 중첩되었을 때는 앞서가는 항공기의 데이터블록은 진행방향으로 쫓아가는 항공기의 데이터블록은 진행방향의 반대방향으로 재배치를 한다. 각 데이터블록의 재배치 시 이동거리는 앞에서 언급한 최소이동으로 한다.

(2) 반대 방향으로 중첩

반대 방향으로 중첩은 가장 적게 일어나는 중첩이다. 항로는 상행 또는 하행으로 정해져있으며, 항로 간 거리가 가깝지 않은 이상 반대 방향으로 중첩이 일어나지 않는다. 더욱이 항로 설계상 상행과 하행 항로의 거리는 특정 지역을 제외하고는 가깝지 않다.

그림 8은 반대 벡터 방향을 가진 데이터블록이 중첩되었을 때 재배치되는 예이다. 그림과 같이 반대 방향으로 중첩되었을 때는 각 항공기의 진행방향과 90도 되는 방향으로 재배치를 한다. 이때 각 데이터블록의 위치에 따라 상하 또는 좌우로 최소이동이 이루어진다. (8방향이상 벡터 양자화 시에는 대각선으로 이동 될 수도 있다.)

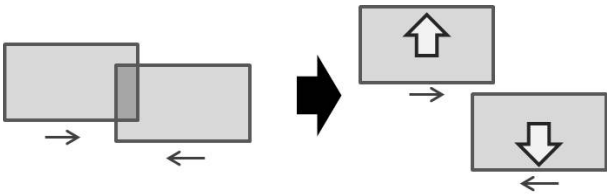


그림 8. 반대 벡터 방향의 재배치의 예
Fig. 8. Example of relocation of opposite vector direction.

(3) 나머지 방향으로 중첩

같은 방향과 반대 방향의 중첩을 제외한 나머지 중첩이다. 항공기의 이동 방향이 자주 변하는 공항 활주로 부근에서 가장 많이 발생하는 중첩이라 할 수 있다.

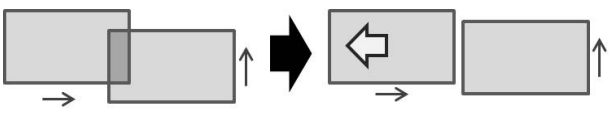


그림 9. 나머지 벡터 방향의 재배치의 예
Fig. 9. Example of relocation of the others vector direction.

그림 9는 나머지 벡터 방향을 가진 데이터블록이 중첩되었을 때 재배치되는 예이다. 그림과 같이 한 데이터블록만 진행방향의 반대방향으로 최소이동 재배치를 한다. 이렇게 되면 재배치한 데이터블록은 다른 데이터블록이 지나갈 때까지 계속 반대방향으로의 재배치를 하게 된다. 이것은 단점이라 할 수 있지만, 최소이동이라는 목표를 놓고 봤을 때 가장 적절한 방법이라 할 수 있다. 여기서 반대 방향으로 재배치되어야 할 데이터블록 선정은 재배치 시 다른 데이터블록과의 중첩이 일어나지 않는 데이터블록으로 한다(재배치 시 중첩이 두 데이터블록 모두 일어나거나 안 일어날 경우에는 랜덤으로 선정한다).

이렇게 방향성을 고려하여 최소이동을 하는 알고리즘을 DM 알고리즘(Direction & Minimum movement Algorithm)으로 명칭 하였으며, 그림 10은 의사코드를 나타낸다. 여기서 D_A과 D_B는 두 개의 중첩된 데이터블록을 의미한다.

```
    DM_Algorithm(DA, DB) // DA&DB is datablock
 1   if No Vector Quantization then
 2     DA is relocated to MA
 3     DB is relocated to MB
 4   else // Use Vector Quantization
 5     if same vector direction then
 6       if DA > DB position then
 7         DA is relocated to same direction
 8         DB is relocated to opposite direction
 9       else visa versa DA&DB
10       end
11     end
12     else if opposite vector direction then
13       if DA > DB position then
14         DA is relocated to - vertical direction
15         DB is relocated to + vertical direction
16       else visa versa DA&DB
17       end
18     end
19     else // other direction
20       if DA is selected then
21         DA is relocated to opposite direction
22       else DB is relocated to opposite direction
23       end
24     end
25   end DM_Algorithm
```

그림 10. DM 알고리즘의 의사코드
Fig. 10. Psuedo code for DM Algorithm.

다. 특정한 상황에서 LFR 알고리즘을 사용한 재배치

앞서 제안한 DM 알고리즘으로 데이터블록 재배치를 하게 되면 특정한 상황에서 문제가 발생 할 수 있다. 특정한 상황이란 그림 2와 같이 항공기가 밀접하게 분포된 지역에서 재배치가 계속 일어나게 되었을 때 항적심벌과 데이터블록 사이가 멀어지게 되는 경우를 말한다. 항로 전체를 놓고 보았을 때 확률은 적지만 발생 할 수 있는 문제이다. 이 경우에는 LFR 알고리즘을 사용하여 항적심벌을 중심으로 적절한 공간을 찾아 새롭게 데이터블록을 배치한다. 기존의 LFR 알고리즘만 사용할 경우는 항적심벌과 데이터블록의 사이가 멀어져 리더라인이 길어지는 문제가 발생하지만, 리더라인이 길어진 특정한 상황에서만 LFR 알고리즘을 사용할 경우 오히려 항적심벌과 데이터블록 사이를 줄이는 재배치를 하게 된다.

식 (3)과 같이 리더라인 길이가 임계값(Threshold)보다 크거나 같을 경우 LFR 알고리즘을 적용하고, 나머지 경우에는 DM 알고리즘을 적용한다. LFR 알고리즘을 적용할 경우 데이터블록이 기존 위치에서 다소 큰 이동을 통한 재배치가 이루어져 한순간 관제에 혼란을 줄 수 있겠지만, 결과적으로 데이터블록이 많이 분포되어 있는 구역에서 해당 데이터블록을 보기 쉽게 빼내는 작업이 되므로 이후 관제에 있어서는 편의를 준다. 이렇게 임계치에 따라 DM 또는 LFR 알고리즘을 적용하는 방법을 DMLFR 알고리즘이라 명칭하였다.

$$\begin{aligned} &\textit{if } \text{Leader\_line} \geq \text{Threshold } \textit{then} \\ &\text{LFR Algorithm relocation} \\ &\textit{else then} \\ &\text{DM Algorithm relocation} \\ &\textit{end} \end{aligned} \tag{3}$$

IV. 실험 및 결과

본 논문은 데이터블록간의 중첩 시에 재배치를 통해 관제에 불편을 주지 않는 것을 목적으로 한다. 재배치 시에는 최소이동으로 항적심벌과 데이터블록 사이를 멀리 떨어지지 않게 하여 관제에 혼란을 주지 않도록 한다. 실험 결과를 평가하기 위해 실험환경을 정의하고, 재배치 시 데이터블록 이전 위치와 현재 위치의 차이 및 리더라인 길이를 측정하여 분석 결과를 기술한다.

1. 실험 환경

실험은 리눅스 환경에서 개발된 현시시스템(CWP)으로 진행하였다. 실험 데이터는 한국방공식별구역(KADIZ)에서 비행한 항공기들을 6시간동안 녹화한 데이터이다. 실험 조건은 그림 11과 같이 관제 섹터를 북부석(NO)로 하였고, 범위 반경 200NM로 두었다. 범위 반경은 상황전시면의 범위 반경이며 커질수록 줌아웃, 작을수록 줌인이 된다. 200NM은 관제사들이 평균적으로 보는 범위 반경이라 할 수 있다.

그림 12에서 최외각 다각형 형태가 KADIZ 구역이며 KADIZ는 다시 섹터로 나뉜다. 실험을 한 북부석의 경우 제일 왼쪽 상단에 위치한 North Sector(NO)이다. 관

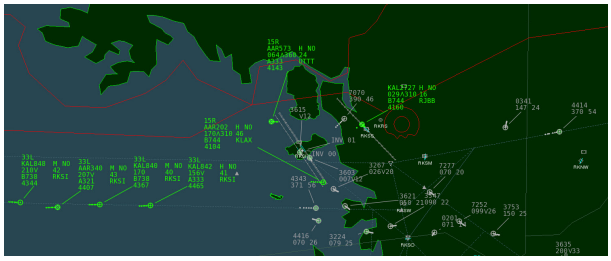


그림 11. NO Sector 상황전시면
Fig. 11. NO sector situation display.

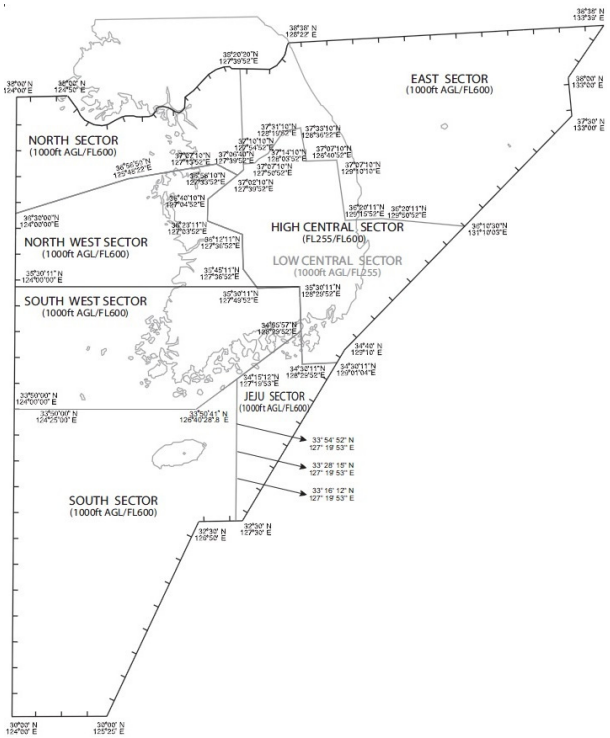


그림 12. KADIZ와 섹터 영역
Fig. 12. KADIZ and Sector region.

제사들은 KADIZ 전체를 관제하는 것이 아니라, 섹터별로 관제하는데, NO섹터는 인천공항과 김포공항이 포함되어 있어 관제섹터 중 가장 혼잡하다. 즉, 항공기가 가장 많은 구역을 중심으로 실험을 진행하였다.

$$D^i_{diff} = D^i_{curr} - D^i_{past} \tag{4}$$

$$L^i_{curr} = D^i_{curr} - S^i_{curr} \tag{5}$$

본 논문에서 실험 평가의 대한 기준은 식 (4)와 식 (5)에 정의하였다. 식 (4)는 해당 데이터블록이 재배치가 일어났을 때, 재배치된 현재 위치(D^i_{curr})와 재배치전 이전 위치(D^i_{past})간의 차이값(D^i_{diff})이다. 식 (5)는 해당 항공기 데이터블록의 현재 위치(D^i_{curr})와 항적심벌의 현재 위치(S^i_{curr})간의 거리인 리더라인의 길이(L^i_{curr})이다. 두 값 모두 작을수록 좋다.

2. 실험 결과

실험은 먼저 DMLFR 알고리즘의 벡터 양자화 정도에 따른 결과와 특정한 상황에서 쓰이는 임계치 값에 따른 결과를 분석하여 최적의 값을 찾는다. 마지막으로 LFR, DM, DMLFR 알고리즘의 결과를 분석한다.

가. 벡터 양자화에 따른 결과

벡터 양자화에 따른 실험은 Ⅲ장의 가.처럼 최소이동만 하는 경우와 4방향, 8방향, 16방향으로 각각 벡터 양자화를 하여 최소이동을 하는 경우로 진행하였다.

표 2는 각 4가지 경우에 대해 중첩이 일어난 모든 항공기의 평균 D_{diff}와 L_{curr}를 구한 결과이다(임계치는 330으로 함). 값의 단위는 상황전시면의 WorldPosition 좌표 단위이며, 그림 1과 같은 초기상태에서 L_{curr} 값은 약 150정도이다.

결과를 살펴보면 최소이동만 하는 경우(No VQ)는 벡터 양자화를 한 경우들보다 값이 모두 크므로, 방향

표 2. 벡터 양자화에 따른 결과
Table 2. Result according to vector quantization.

	D.diff	L.curr
No VQ	29.14078	179.3132
4VQ	6.75708	175.0564
8VQ	6.58995	173.7147
16VQ	9.73043	179.2402

성을 고려하는 것이 더욱 최소이동으로 재배치한다는 것임을 알 수 있다. 각 벡터 양자화를 적용했을 때의 경우를 살펴보면, 8VQ일 때 4VQ와 16VQ보다 값들이 작으므로 8VQ가 최적의 벡터 양자화이다.

나. 임계치에 따른 결과

항적심벌과 데이터블록이 너무 떨어진 특정한 상황에서 LFR 알고리즘 적용을 판단할 때 쓰이는 임계치 실험을 하였다. 임계치에 따라 L_{curr}의 최대값이 설정되므로 적절한 임계치의 범위를 250에서 350으로 선정하였다. 250보다 작으면 LFR 알고리즘이 너무 빈번하게 일어나고, 350보다 크면 리더라인이 너무 길어져 관제화면이 혼란스러워진다.

그림 13은 250부터 350의 임계치 범위에서 10간격으로 모든 항공기의 L_{curr} 평균값을 구한 그래프이다(벡터 양자화는 가장 결과가 좋은 8VQ로함). x축이 임계치 값을 나타내고 y축이 L_{curr}의 평균값을 나타낸다. 결과를 살펴보면, 임계치가 커질수록 L_{curr}가 작아지는 분포를 하고 있으나 330이후에는 오히려 커진 것을 알 수 있다. 즉, 330이 최적의 임계치이다.

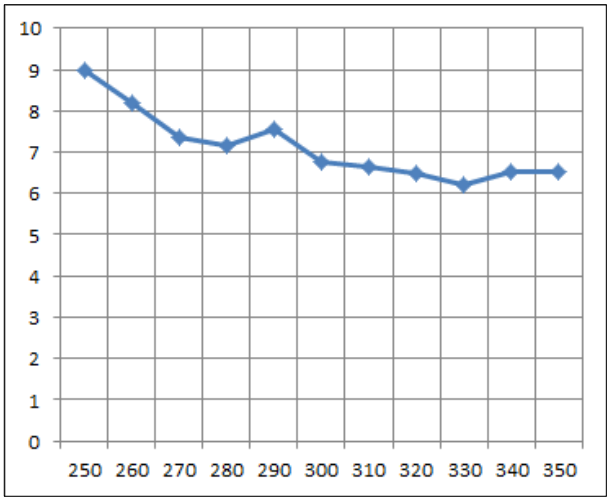


그림 13. 임계치에 따른 Lcurr의 평균 그래프
Fig. 13. Average graph of Lcurr according to threshold.

다. 각 알고리즘 비교 분석

마지막으로 LFR 알고리즘, DM 알고리즘, DMLFR 알고리즘에 대한 결과를 분석한다. 그림 14는 각 알고리즘에 대한 결과 그래프이다. X축은 L_{curr}값이고 Y축은 D_{diff}값이다. 수많은 중첩된 항공기의 데이터블록 중에서

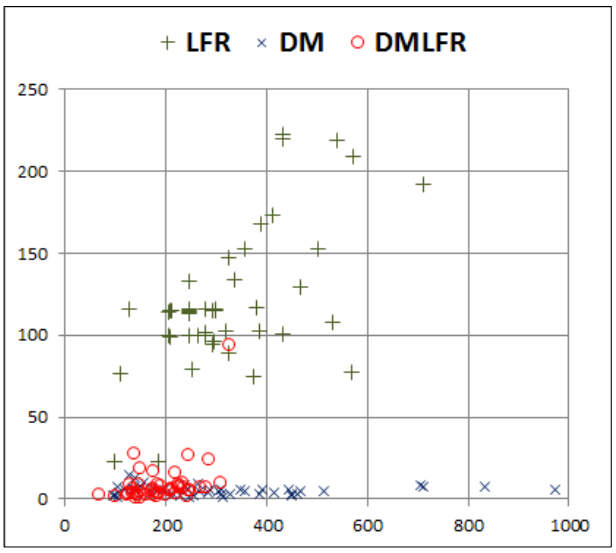


그림 14. 알고리즘에 따른 결과 그래프
Fig. 14. Result graph according to algorithm.

랜덤으로 50개를 뽑아 그래프로 도시하였다. 그래프를 보면 알 수 있듯이, LFR 알고리즘의 경우는 L_{curr}와 D_{diff}값이 상대적으로 큰 분포가 나왔다. DM 알고리즘은 D_{diff}값은 제일 작은 분포지만 L_{curr}값이 큰 분포를 하고 있다. DMLFR 알고리즘의 경우 D_{diff}, L_{curr} 모두 가장 작은 값으로 분포하고 있다.

표 3은 각 알고리즘에 따른 모든 항공기의 D_{diff}와 L_{curr}, 그리고 데이터블록 재배치 처리속도의 평균 결과이다. 평균 결과에서도 DMLFR 알고리즘의 결과가 가장 뛰어나다. D_{diff}에서 DM 알고리즘만 사용한 경우, DMLFR 알고리즘보다 뛰어난 결과를 얻었으나 특정한 상황에서의 예외처리를 하지 않았으므로 L_{curr}이 가장 나쁘다. 제안한 최종 알고리즘 DMLFR과 기존 알고리즘인 LFR을 비교 분석해보면 DMLFR이 D_{diff}에서 약17배 뛰어나며, L_{curr}에서 약2배 뛰어나다. 처리 속도는 ms(밀리초)단위이므로 세 알고리즘 모두 빠르다고 할 수 있으나, 그중에서도 DMLFR이 가장 빠르다.

마지막으로, 예로써 DMLFR과 LFR 알고리즘의 성능을 그림 15와 16으로 나타내었다. DMLFR의 경우는 그

표 3. 알고리즘에 따른 결과
Table 3. Result according to algorithm.

	D.diff	L.curr	처리 속도(ms)
LFR	115.90091	359.7188	0.0961
DM	4.40880	368.1255	0.0103
DMLFR	6.58995	173.7147	0.0088

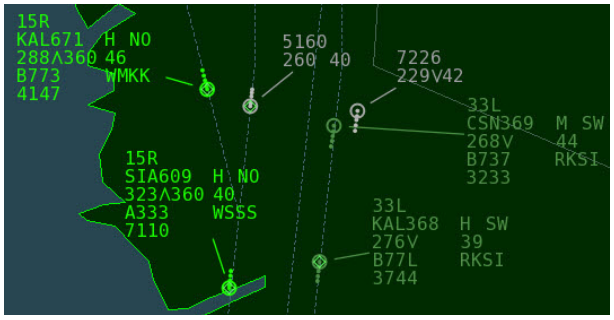


그림 15. DMLFR 알고리즘의 예
Fig. 15. Example of DMLFR algorithm.

그림 16. LFR 알고리즘의 예
Fig. 16. Example of LFR algorithm.

림 15처럼 관제사에게 혼란을 주지 않게 적절히 재배치된 것에 반해, LFR의 경우는 그림 16처럼 데이터블록의 특성을 전혀 고려하지 않고 최적의 빈공간만 찾아 재배치하다보니 혼란스럽게 되었다.

V. 결 론

본 논문에서는 중첩된 항공기의 데이터블록을 재배치하는 알고리즘을 제안하였다. 기존의 직사각형을 배치하는 알고리즘인 LFR 분석하고 데이터블록 특성에 맞게 새롭게 제안한 알고리즘인 DM을 제시하였으며, 최종적으로 각 알고리즘의 단점을 극복하기 위해 리더라인에 임계치를 두어 상황에 따라 DM 또는 LFR 알고리즘을 적용하는 DMLFR 알고리즘을 제안하였다. 알고리즘의 성능을 평가하기 위해 데이터블록의 특성에 따라 재배치 시 이전과 현재 위치의 차이값인 D_{diff}와 리더라인 길이인 L_{curr}를 두었다. 실험 결과는 DMLFR 알고리즘이 DM과 LFR 알고리즘보다 우수한 성능을 보였다.

본 논문에서 제안하는 알고리즘을 사용하여 단순히 벡터만을 가진 직사각형을 재배치할 경우에는 뛰어난 성능을 낸다고 볼 수 없다. 왜냐하면 제안하는 알고리즘은 항공기의 위치를 고려하여 적절히 직사각형(데이터블록)을 재배치하기 때문이다. 즉, 관제의 특성을 고려한 알고리즘이므로 관제에 혼란을 주지 않고 데이터블록을 재배치할 경우에는 매우 효율적인 알고리즘이라 할 수 있다.

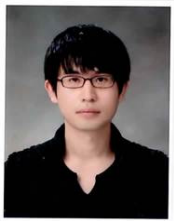
추후 관련 연구가 많이 이루어지면 비교분석을 통해 본 논문에서 제안한 알고리즘이 효율적이라는 것을 더욱 입증해야겠다.

REFERENCES

[1] Y. K Kim, J. W. Han, H. D. Park, “The propose of control sector management for efficient air traffic control,” Electronics and Telecommunications Research Institute, pp. 461-462, November 2011.
[2] MOLIT, “항공운송시장 동향,” No. 22, April 2014.
[3] MOLIT, “전량 수입하던 항공관제시스템, 본격 국산화,” Available at : http://www.mltm.go.kr/USR/NEWS/m_71/dtl.jsp?id=95074076
[4] S. I. Na, J. W. Lee, I. S. Won, S. B. Choi, H. D. Park, D. S. Jeong, “The research of the Control Work Position for developing ATC,” IEIE Conference, pp. 1197-1198, June 2008.
[5] K. H. Kim, H. J. Kim, Y. R. Dong, S. B. Choi, “Design and Implementation of ASTERIX Parsing Module Based on Pattern Matching for Air Traffic Control Display System,” IEIE Journal, Vol 51, No. 3. pp. 89-101, March 2014.
[6] D. G. Jeon, Y. J. Eun, H. K. Kim, C. H. Yum, “Development of Multi-Sensor Data Processing Software for ATC,” KSAS Conference, pp. 608-614, November 2012.
[7] M. Bernard and F. Jacquenet, “Free space modeling for placing rectangles without overlapping,” Journal of Universal Computer Science., Vol. 3, No. 6, pp. 703-720, 1997.

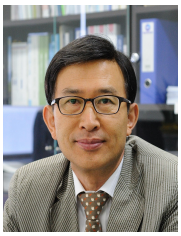
저 자 소 개

정 재 협(학생회원)
2009년 인하대학교 전자공학과 학사 졸업
2011년 인하대학교 전자공학과 석사 졸업
2011년～현재 인하대학교 전자공학과 박사 과정
<주관심분야: 영상신호처리, 패턴인식, 내용 기반 검색, 컴퓨터 비젼, 항공 관제 개발>



양 훈 준(학생회원)
2011년 인하대학교 전자공학과 학사 졸업
2013년 인하대학교 전자공학과 석사 졸업
2013년～현재 인하대학교 전자공학과 박사 과정
<주관심분야: 패턴인식, 비디오압축, 영상처리>

원 인 수(학생회원)
2006년 인하대학교 전자공학과 학사 졸업
2009년 인하대학교 전자공학과 석사 졸업
2009년～현재 인하대학교 전자공학과 박사 과정
<주관심분야: 패턴인식, 컴퓨터 비젼, 내용 기반 검색, 항공 관제 개발>



정 동 석(정회원)
1977년 서울대학교 전기공학과 학사 졸업
1985년 Virginia Tech 전자공학과 공학 석사
1988년 Virginia Tech 전자공학과 공학 박사
1988년～현재 인하대학교 전자공학과 교수
1990년～1994년 전자공학회 논문지 편집위원
1990년～1994년 통신학회 논문지 편집위원
2000년～2004년 정보전자공동연구소 소장
2010년～2012년 인하대학교 IT공대학장
2012년 인하공업전문대학교 총장
<주관심분야 : 영상처리, 컴퓨터 비젼, 패턴인식, 내용기반 멀티미디어검색>