DOI: 10.3745/KIPSTA.2011.18A.2.075

장애물을 제외한 가장 큰 공간을 찾는 기법

황 정 환[†] · 전 흥 석[††]

요 약

최근 청소로봇을 위한 여러 가지 알고리즘들이 개발되면서 다양한 청소로봇들이 개발되고 있다. 그 중 청소로봇이 청소하는데 있어 청소시간의 제약이 있을 경우 효율적으로 청소할 수 있는 알고리즘인 DmaxCoverage 알고리즘이 있는데 이 알고리즘을 구현하는데 장애물이 존재하지 않는 비어있는 공간을 찾기 위해 Rectangle Tiling기법을 사용하고 있다. Map을 그리드 형태의 수많은 사각형으로 나타낼 경우 사각형을 찾는 기법 중 수학적 방법으로 Rectangle Tiling이 최적의 값을 찾아줄 수 있다. Rectangle Tiling 기법은 그리드 형식의 map에서 생성될 수 있는 사각형들의 모든 경우의 수를 찾는 것이다. 이때 그리드선의 간격이 촘촘하고 map의 크기가 클 경우 많은 사각형이 생성됨으로 많은 시간을 소모해야 하는 문제점이 있다. 본 논문에서는 Rectangle Tiling에 근접한 정확성과 보다 개선된 속도를 제공하는 Four Direction Rectangle Scanning(FDRS)기법을 제안한다. FDRS기법은 존재하는 모든 사각형을 찾는 것이 아니라 물체가 존재하는 셀의 상하좌우만을 검색하여 빈 공간을 찾는 기법이다. 이 두 알고리즘을 비교하여 FDRS의 효율이 뛰어남을 실험을 통해 보여준다.

키워드 : 청소로봇, 알고리즘, 맵, 사각형, 렉탱글타일링

A new scheme for finding the biggest rectangle that doesn't have any obstacle

Jung Hwan Hwang[†] · Heung Seok Jeon[††]

ABSTRACT

Recently, many cleaning robots have been made with various algorithms for efficient cleaning. One of them is a DmaxCoverage algorithm which efficiently clean for the situation when the robot has a time limit. This algorithm uses Rectangle Tiling method for finding the biggest rectangle that doesn't have any obstacle. When the robot uses grid map, Rectangle Tiling method can find the optimal value. Rectangle Tiling method is to find all of the rectangles in the grid map. But when the grid map is big, it has a problem that spends a lot of times because of the large numbers of rectangles. In this paper, we propose Four Direction Rectangle Scanning(FDRS) method that has similar accuracy but faster than Rectangle Tiling method. FDRS method is not to find all of the rectangle, but to search the obstacle's all directions. We will show the FDRS method's performance by comparing of FDRS and Rectangle Tiling methods.

Keywords : Cleaning Robot, Algorithm, Map, Rectangle, Rectangle Tiling

1. 서 론

첨단 과학 시대인 오늘날 사람들의 생활을 편리하게 해주는 여러 가지 발명품들이 많이 발명되고 있다. 최근 들어 주목 받는 발명품 중 하나가 청소로봇이다. 여러 회사에서 각기 특색을 가진 알고리즘들을 이용하여 다양한 가격대의 청소로봇들을 판매하고 있다. 청소 로봇을 위한 여러 가지 알고리즘들 중에 시간적 제약이 존재할 경우 효율적인 청소를 위해 제안된 DmaxCoverage라는 알고리즘이 있는데 이 알고리즘이 제안하는 효율적인 청소방법은 map에서 장애물이 존재하지 않는 빈 공간을 찾아 청소하고 청소가 끝나면 다음으로 큰 빈 공간을 찾아 청소하는 것이다[1]. 이 때문에 DmaxCoverage에서는 빈 공간을 찾기 위한 방법이 필요한데, 이를 위하여 Rectangle Tiling[2]을 사용한다. 그러나 Rectangle Tiling은 map을 그리드 형식의 사각형으로 나타낼 때 map에 존재하는 모든 형태의 사각형들의 경우의 수

※ 이 논문은 2010년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No.2010-0005855).
† 준 회 원 : 건국대학교 컴퓨터·응용과학부 학부생
†† 종신회원 : 건국대학교 컴퓨터소프트웨어전공 부교수(교신저자)
논문접수 : 2010년 9월 28일
수 정 일 : 1차 2011년 1월 14일
심사완료 : 2011년 1월 19일

를 전부 구하기 때문에 시간이 오래 걸리는 단점이 있고 Rectangle Tiling만으로는 장애물을 제외한 가장 큰 사각형을 찾을 수 없기 때문에 Rectangle Tiling을 통해 구한 사각형들 중 적합한 사각형을 구하는 방법이 추가적으로 필요하게 된다.

따라서 본 논문에서는 DmaxCoverage 알고리즘에서 사용하는 Rectangle Tiling 기법보다 효율이 뛰어난 FDRS 기법을 제안한다. 이 기법은 기존 Rectangle Tiling의 단점인 모든 사각형 객체를 구하지 않고 사각형의 가로 세로 길이만을 이용해 보다 적은 수의 사각형으로 장애물을 제외한 가장 큰 사각형을 찾음으로써 기존 DmaxCoverage 알고리즘의 속도를 향상시킬 수 있다.

본 논문의 나머지 구성은 다음과 같다. 2절에서는 Rectangle Tiling 기법에 대해 살펴본다. 3절에서는 본 논문에서 제안하는 FDRS 기법을 설명한다. 4절에서는 시뮬레이션을 통한 성능 평가 결과를 기술한다. 5절에서는 본 논문의 결론을 기술한다.

2. 관련연구

2.1 DmaxCoverage

2.1.1 DmaxCoverage의 개념

로봇의 이동경로를 결정하여 모든 지역을 방문하게 해주는 Coverage 알고리즘 중에 하나인 DmaxCoverage 알고리즘은 map을 미리 알고 있다고 가정하는 다른 알고리즘과는 달리 map을 로봇이 모르고 있다고 가정하고 SLAM기법을 이용하여 map을 획득하고 획득한 map을 기반으로 완전한 경로를 구하여 장애물이 존재하지 않는 가장 큰 지역을 찾아서 제한된 시간 내에 최대한의 영역을 청소 할 수 있는 알고리즘이다 [1].

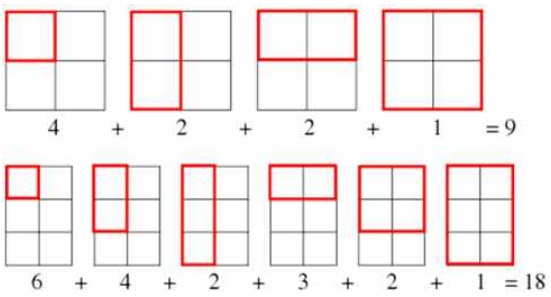
2.1.2 DmaxCoverage의 원리

로봇이 map을 획득하기 위해서 센서를 통해 얻어진 정보들을 바탕으로 MBR(Minimum Bounding Rectangle)을 만든다. 만들어진 MBR에 Rectangle Tiling을 사용하여 MBR에 존재하는 모든 사각형들 중 장애물이 존재하지 않는 모든 사각형을 구한다. 그 후 SetCovering을 통해 구해진 사각형들을 사각형의 크기순으로 정렬한다. getR1함수를 통해 정렬된 사각형들의 방문순서를 결정하고 순서대로 청소를 한다. 청소 중 센서를 통해 새로운 정보가 들어오면 위의 과정을 반복해 map을 업데이트 한다.

2.2 Rectangle Tiling

2.2.1 Rectangle Tiling의 원리

우선 장애물을 제외한 가장 큰 공간을 찾기 위해 DmaxCoverage 알고리즘에서 사용하는 Rectangle Tiling 기법의 원리는 map을 그리드 형태로 나타낼 경우 구할 수 있는 모든 사각형을 구하는 방법이다[2].



(그림 1) Rectangle Tiling의 원리(그림 출처: http://mathworld.wolfram.com/RectangleTiling.html)

Rectangle Tiling은 (그림 1) 과 같이 2x2 행렬의 map에서 존재하는 모든 사각형을 구할 때 1x1인 사각형이 4개, 1x2인 사각형이 2개, 2x1인 사각형이 2개, 2x2인 사각형이 1개 등 총 9가지의 사각형을 구할 수 있다[3].

위 방법을 프로그램으로 구현할 경우 사각형 하나를 나타내기 위해 2개의 좌표를 저장할 공간을 만들어야 하고 그 좌표를 증가 시켜야 하므로 4중의 반복문을 필요로 하게 된다. 우선 사각형의 왼쪽 위의 점을 X1, Y1(0,0)으로 표현하고 오른쪽 아래 점을 X2, Y2(1,1)로 표현한다. 여기서 주의할 점은 X2, Y2의 초기 값은 항상 현재 X1, Y1값보다 1이 큰 값이어야 한다는 것이다. 그 후 반복문을 이용하여₲X2를 1씩 증가시켜 나가면서 사각형을 저장한다. X2가 map의 끝에 닿으면 반복문을 이용하여₲Y2를 1씩 증가시키고 X2를 초기 값으로 바꾸고 다시 위의 반복문을 반복한다. Y2도 map의 끝에 닿게 되면 이번에는 X1의 값을 반복문을 이용해 1씩 증가시키고 X2, Y2의 값을 초기 값으로 변경시킨 후 위의 두 반복문을 반복 수행한다. X1이 map의 끝에 닿게 되면 Y1을 for문을 이용해서 1씩 증가시키고 위의 세 반복문을 반복 수행한다. 이것이 Rectangle Tiling의 동작 원리이다.

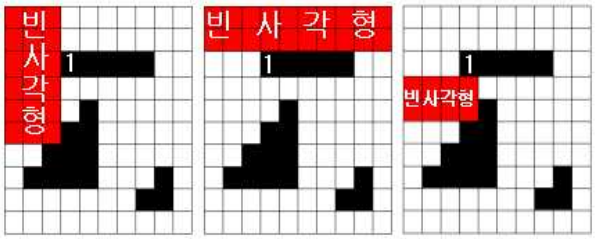
2.2.2 Rectangle Tiling의 단점

장애물을 제외한 가장 큰 공간을 찾기 위해 Rectangle Tiling을 사용할 경우 Rectangle Tiling만으로는 그 공간을 찾을 수 가 없다. Rectangle Tiling은 map에 존재하는 모든 사각형을 찾는 기법이지 가장 큰 빈공간을 찾아주는 기법이 아니기 때문이다. 그러므로 중첩된 4개의 반복문외에 Rectangle Tiling이 찾은 사각형들의 내부에 장애물의 유무를 판단 할 수 있는 코드와 장애물이 없는 Rectangle Tiling이 찾은 사각형들의 집합 중 가장 큰 사각형을 골라내는 코드가 추가적으로 필요하기 때문에 총 6개의 중첩된 반복문이 필요하게 된다. 그러므로 그리드 형태로 나타낸 map의 사각형이 촘촘하고 map의 크기가 클수록 기하급수적으로 증가하는 수행시간이 문제가 된다.

3. FDRS

3.1 FDRS의 원리

FDRS는 Four Direction Rectangle Scanning의 약자이다. 이 알고리즘의 목표는 map상에 존재하는 모든 사각형을 구해야 하는 Rectangle Tiling의 단점을 개선하는 것이다.



〈a〉 좌측 검색 〈b〉 하단 검색 〈c〉 상단 검색

(그림 2) FDRS의 원리.

FDRS는 (그림 2)와 같이 물체가 존재하는 셀의 상, 하, 좌, 우를 검색한다. 위의 (그림 2)에서는 1번 셀을 기준으로 상, 하, 좌측의 빈 사각형을 찾을 수 있고 1번 셀의 우측에는 이미 검은색의 물체가 있는 셀이 존재하기 때문에 스캔하지 않는다.

위 방법을 프로그램으로 구현하는 방법은 map의 장애물 유무를 판단하기 위한 임의의 변수 A1, B1을 (0, 0)으로 초기화 시킨 후 2개의 반복문으로 A1, B1을 map의 X, Y크기까지 증가시키며 장애물의 유무를 판단한다. 장애물을 만나게 되면 그 장애물의 상, 하, 좌, 우를 검색하게 된다.

우선 첫 번째로 만나게 되는 셀이 (그림 2)의 1번 셀이다. 1번 셀의 좌측부터 검사를 하게 되면 현재 장애물이 있는 1번 셀의 일직선상 왼쪽에 장애물이 있는지 없는지를 판단하고 물체가 없으면 map의 왼쪽 변의 X좌표와 현재 장애물 사이의 셀 수를 사각형의 높이로 잡게 된다. 반대로 장애물이 있을 경우 좌측의 장애물과 현재 장애물 사이의 셀 수를 사각형의 높이로 설정한다. (그림 2)에서는 1번 셀의 일직선상 왼쪽에 물체가 없기 때문에 map의 왼쪽 변의 X좌표인 0과 1번 셀의 X좌표인 3을 이용하여 높이의 시작 셀인 0을 X1, 끝 셀인 3을 X2로 설정할 수 있다. X좌표를 이용해 가로를 높이로 설정하였으므로 Y좌표를 이용해 세로의 길이를 구해야 한다. 반복문 두 개를 이용해 A2는 위에서 구한 사각형 높이의 시작비트 0부터 끝 비트 3까지 증가시키고 B2는 0부터 map의 Y크기까지 증가시킨다. 이렇게 검색을 해서 현재 장애물의 Y값에 가장 가까운 위쪽 장애물의 B2값을 Y1, 현재 장애물의 가장 가까운 아래쪽 장애물의 B2값을 Y2로 설정하고 X1, X2, Y1, Y2값을 이용해서 사각형의 크기를 계산한다. (그림 2) 에서는 X좌표 0~3사이의 1번 비트 위쪽으로 물체가 없으므로 Y1은 0, 아래쪽은 가장 가까운 장애물의 셀의 좌표가 (2,6),(3,7)이므로 Y2는 6으로 잡는다. 이렇게 되면 (X1,Y1),(X2,Y2)는 (0,0),(3,6)이 되고 이 빈 사각형의 넓이는 좌표를 이용하여 18로 구할 수 있게 된다.

위와 같은 방식으로 상, 하, 좌, 우를 검색하고 검색이 끝나면 다음 장애물의 셀을 찾아 상, 하, 좌, 우를 검색하는 것을 반복한다. 위의 설명과 같이 FDRS기법은 빈 공간을 한 번에 찾아낼 수 있지만 Rectangle Tiling기법은 3개의 빈 사각형 하나를 찾아내기 위하여 그 사각형 안에 존재 하는 다른 작은 사각형들을 모두 찾아야 한다. 이렇게 본 논문이 제안하는 FDRS 알고리즘은 4개의 중첩된 반복문만으로 장애물을 제외한 가장 큰 공간을 찾을 수 있다.

4. 실험 및 성능 비교

4.1 실험 목표

동일한 map에서 map에 존재하는 사각형이 많아 질수록 지형에 대한 세밀한 정보를 담을 수 있으므로 map에 존재하는 사각형의 개수는 많으면 많을수록 좋다. 하지만 사각형의 개수가 많아지면 많아질수록 연산해야 하는 사각형이 많아지므로 그만큼의 시간이 더 소요된다. 본 논문에서 제안한 FDRS는 장애물을 기준으로 빈공간을 탐색하고 Rectangle Tiling은 존재하는 모든 셀을 탐색하여 빈공간을 찾아낸다. 따라서 이 두 알고리즘에 영향을 미치는 요인은 장애물의 개수와 map에 존재하는 사각형의 개수이다. 우선 실험4.2에서는 map에 존재하는 사각형의 개수에 따른 두 알고리즘의 비교를 위해 장애물의 개수는 고정시키고 map의 크기에 따른 알고리즘의 수행속도를 비교하는 실험을 준비하였다. 4.3에서는 실험 4.2가 장애물의 개수는 고정시키지만 두 map이 완전히 동일한 환경은 아니라는 것에 대한 실험의 취약점을 보완하기 위해 준비하였다. 실험4.4는 장애물의 개수에 따른 두알고리즘의 성능을 비교하는 실험을 준비하였다. 이 실험들을 통해 두 알고리즘의 수행결과와 수행시간을 비교하여 FDRS의 효율이 뛰어남을 증명하는 것이 실험의 목표이다.

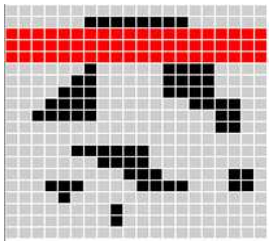
4.2 일정한 수의 장애물에서 map의 크기에 따른 소요시간

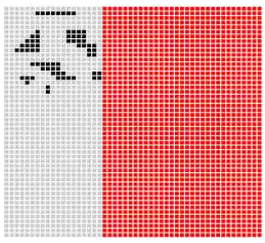
실험목표에서 언급했듯이 알고리즘의 성능에 영향을 미치는 요인은 장애물의 개수와 map에 존재하는 사각형의 개수이다. 첫 번째 실험은 map에 존재하는 사각형이 많아질 경우 Rectangle Tiling과 FDRS의 수행시간을 비교하는 실험이다. 하지만 장애물의 개수 또한 알고리즘의 성능에 영향을 미치므로 본 실험에서는 장애물의 개수를 70개로 고정시키고 map의 크기를 20x20, 50x50, 100x100으로 변화시켜 두 알고리즘의 정확한 성능 차를 비교해 보고자 한다.

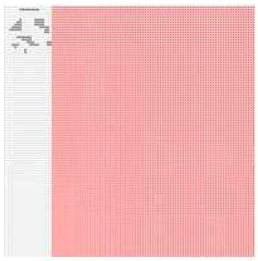
두 알고리즘이 모두 동일한 사각형을 가장 큰 사각형으로 찾게 되어 두 알고리즘의 정확도는 동일하기 때문에 (그림 3)과 같이 하나의 그림으로 나타내었다. 반면에 수행시간은 Rectangle Tiling이 FDRS에 비해 급격히 늘어나는데 반해 FDRS는 미세하게 증가 하는 것을 알 수 있다. 두 알고리즘의 수행 시간을 그래프로 나타내면 (그림 4)와 같다.

4.3 실제 환경의 map에서 map크기(배율)에 따른 소요시간

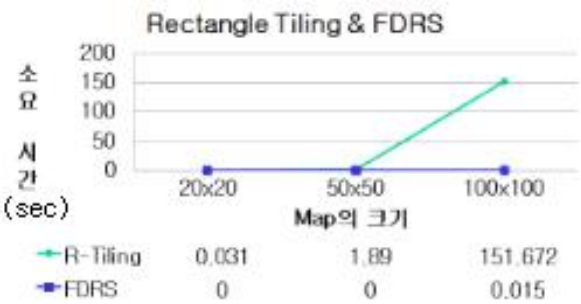
두 번째 실험은 실제 생활환경과 유사한 map을 만들고 각기 다른 사각형의 개수로 같은 모양의 map을 표현하여 수행시간을 비교하는 실험이다. 실험 4.2에서는 알고리즘의 수행시간에 영향을 미치는 원인인 map에 존재하는 사각형


〈a〉 20x20


〈b〉 50x50

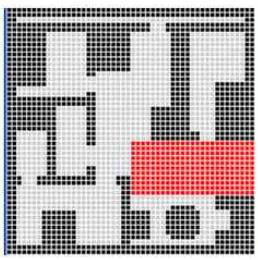

〈c〉 100x100

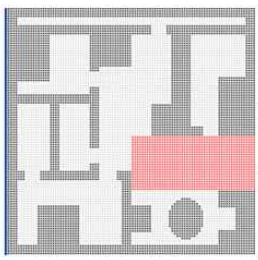
(그림 3) 일정한 장애물에서의map의 크기 변화에 따른 FDRS와 Rectangle Tiling의 동일한 수행결과



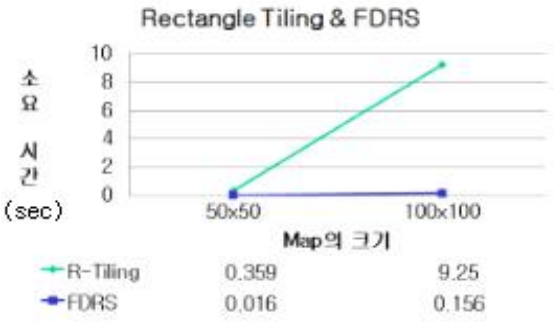
(그림 4) 일정한 수의 장애물에서 map의 크기에 따른 알고리즘 비교

의 개수에 따른 수행시간만을 비교하기 위하여 수행시간에 영향을 미치는 또 다른 원인인 장애물의 개수를 고정시켰다. 하지만 map의 장애물을 고정시킴으로써 세 가지 map의 장애물의 모양은 동일하지만 map자체가 동일하다고 할 수 없게 되었다. 따라서 본 실험을 통해 실험4.2에서의 부족한 부분을 보완하고 실제 생활환경에서도 알고리즘이 정확하게 작동한다는 것을 보여주어 알고리즘에 대한 신뢰성을 높이고자 한다.


〈a〉 50x50


〈b〉 100x100

(그림 5) 실제 환경에서의 map크기 변화에 따른 FDRS와 Rectangle Tiling의 동일한 수행결과

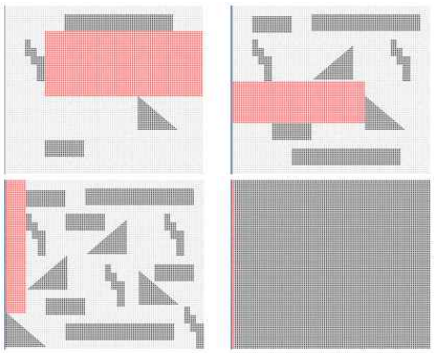


(그림 6) 실제 환경에서의 map크기에 따른 알고리즘 비교

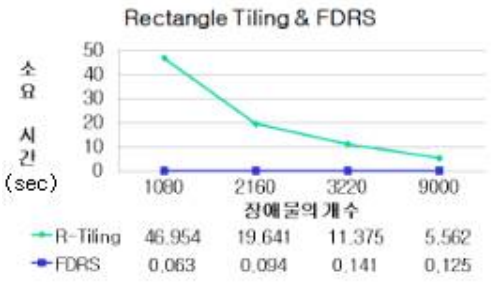
(그림 5)의 실험결과 역시 두 알고리즘 모두 같은 사각형을 가장 큰 사각형으로 찾았으므로 두 알고리즘의 정확도는 같다. 또한 수행시간 역시 Rectangle Tiling이 FDRS에 비해 급격히 늘어나는데 반해 FDRS는 미세하게 증가 나는 결과를 보여준다. 두 알고리즘의 수행 시간을 그래프로 나타내면 (그림 6)과 같다.

4.4 일정한 크기의 map에서 장애물의 수의 변화에 따른 소요시간

세 번째 실험은 map의 크기를 100x100으로 고정시키고 장애물의 수를 변화시켰을 경우에 따른 수행시간의 비교이다. 첫 번째 실험과 두 번째 실험에서 map의 크기에 따른 속도차이는 증명을 했고 작은 크기의 map에서는 성능차이가 거의 없으므로 성능차이가 많이 나는 100x100의 map에서 실험을 진행하였다.



(그림 7) 일정한 크기의 map에서 장애물 수의 변화에 따른 Rectangle Tiling과 FDRS의 동일한 수행결과

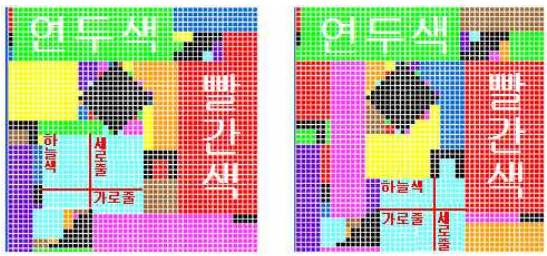


(그림 8) 일정한 크기의 map에서 장애물 수에 따른 알고리즘 비교

두 알고리즘이 모두 동일한 사각형을 가장 큰 사각형으로 찾게 되어 두 알고리즘의 정확도는 동일하기 때문에 (그림 7)과 같이 하나의 그림으로 나타내었다. 그러나 수행시간은 조금 차이가 나는데 FDRS는 수행시간이 거의 비슷한 반면 Rectangle Tiling은 장애물이 늘어날수록 수행시간이 줄어든다. 이는 프로그램을 작성함에 있어서 실험시간을 줄이고자 Rectangle Tiling 알고리즘의 수행이 끝난 후 사각형 내의 장애물의 유무를 판단할 때 장애물을 만나면 더 이상 검사하지 않고 탈출하는 부분을 넣었기에 장애물이 늘어날수록 검사시간이 줄어드는 현상이 생기게 된다. 그러나 그럼에도 불구하고 FDRS의 성능이 우수하다는 것을 증명하기 위해 100x100의 map에 발생할 가능성이 희박한 9000개의 장애물과 1000개의 빈 공간이 있는 환경을 적용하였음에도 여전히 Rectangle Tiling의 수행시간이 오래 걸리는 것을 볼 수 있다. 두 알고리즘의 수행 결과를 그래프로 나타내면 (그림 8)과 같다.

4.5 FDRS의 오차

특정 map에서 FDRS는 Rectangle Tiling 과 같은 실험결과를 보여주지 못하는 경우도 존재한다. 네 번째 실험은 이러한 FDRS와 Rectangle Tiling 의 실험결과에 따른 오차를 알아보기 위한 실험이다. 우선 오차에 대해 설명하기에 앞서 편의상 두 알고리즘이 찾은 장애물이 존재하지 않는 가장 큰 공간을 R1이라고 부르고 반복하여 실행 하였을 경우 R1다음으로 큰 공간을 R2라 지칭하겠다.



〈a〉 Rectangle Tiling 〈b〉 FDRS

(그림 9) 50x50 map에서의 FDRS의 오차

(그림 9)는 크기가 50x50인 임의의 map에 대한 Rectangle Tiling과 FDRS의 수행결과이다. 그림을 보면 두 실행결과가 다름을 알 수 있는데 Rectangle Tiling은 map내에 존재하는 모든 경우의 사각형 수를 전부 따져 비교하기 때문에 Rectangle Tiling의 결과가 정확한 결과이다. FDRS의 수행결과에 오차가 생긴 이유는 우선 알고리즘을 구현할 때 한 map에서 알고리즘을 중첩 실행시켰을 경우 R1을 빨간색, R2를 연두색, R3를 하늘색으로 나타나게끔 하였다. 이때 알고리즘을 세 번째 수행하여 하늘색 사각형을 찾을 때 전에 찾은 R1과 R2는 장애물로 취급하게 된다. FDRS의 특성상 알고리즘을 수행하게 되면 높이(세로줄)를 먼저 찾게 된다. 그런데 (그림 9) <a>의 하늘색 사각형은 장애물이나 빨간색, 연두색 사각형처럼 높이를 탐색할 수 있는 선이 존재하지 않는다. 따라서 FDRS가 (그림 9) <b>와 같은 사각형을 찾게 되는 것이다. (그림 9) <b>의 R3의 세로길이는 장애물과 map의 아래 변 사이의 길이이고 가로길이는 왼쪽 장애물과 R1사이의 길이이다.

비록 FDRS가 항상 최적의 값을 찾는 것은 아니지만 청소 로봇의 청소결과에 영향을 끼치지 않을 차선의 값을 찾을 수 있다는 것을 증명하기 위해 다음과 같은 실험을 진행하였다. 위의 실험에서 두 알고리즘의 수행결과 얻게 된 R1~R9까지의 사각 형의 크기를 조사한 후 두 결과를 (그림 10)의 그래프를 통해 나타내었다.

(그림 10) 오차 발생 map에서 두 알고리즘의 R1~R9까지의 크기(단위는 사각형의 크기)

(그림 10)에서 보는 것과 같이 두 알고리즘의 수행결과가 크게 차이 나지 않고 항상 전에 찾은 사각형이 후에 찾은 사각형보다 크거나 같으므로 비슷한 기울기의 그래프를 나타내기 때문에 FDRS가 Rectangle Tiling보다 큰 값을 찾아내는 경우도 생긴다.

5. 결 론

본 논문에서는 DmaxCoverage 알고리즘에서 장애물을 제외한 가장 큰 공간을 찾기 위해 사용된 Rectangle Tiling의 문제점인 수행시간을 줄이는 FDRS기법을 제안하였다. 이 기법은 모든 사각형을 구하지 않고 물체의 주변을 탐색하며 가로 세로의 길이로 사각형을 구성하여 수행시간을 줄이면서도 Rectangle Tiling의 결과에 근접한 결과를 보였다.

청소 로봇에 사용되는 CPU가 일반 컴퓨터에 비해 낮은 성능을 가지고 있기 때문에 실제 구현 과정에서 Rectangle Tiling과 FDRS의 속도차이는 본 논문의 실험보다 더욱 크게 차이가 날 것으로 예상된다. 따라서 본 논문에서 제안하는 FDRS기법은 실제 청소로봇에 적용함에 있어서 Rectangle Tiling보다 더욱 효과적으로 적용될 것으로 기대된다.

본 논문의 향후 연구 과제는 더욱 빠른 속도를 제공하면서 Rectangle Tiling과 동일한 정확도를 보일 수 있도록 보완하는 것이다.

참 고 문 헌

[1] Heung Seok Jeon, Jung Hwan Park, and Ryumduck Oh, “An Efficient Robot Coverage Algorithm Integrated with SLAM for Unknown Environments”, In Proceedings of International Conference on Artificial Intelligence. pp.512-257. July 12-15, 2010. USA

[2] Ian stewart, "Squaring the Square", Scientific American, pp. 74-76, July, 1997.

[3] http://mathworld.wolfram.com/RectangleTiling.html

황 정 환

e-mail : sayhi59@naver.com

2004년 건국대 컴퓨터·응용과학부 입학

2011년～현 재 건국대학교 컴퓨터·응용과학부 4학년

관심분야 : 지능형 로봇, 임베디드시스템

전 흥 석

e-mail : hsjeon@kku.ac.kr

1996년 홍익대학교 컴퓨터공학과(학사)

1998년 홍익대학교 전자계산학과(석사)

2001년 홍익대학교 전자계산학과(박사)

2002년～2006년 건국대 컴퓨터소프트웨어전공 조교수

2007년～현 재 건국대 컴퓨터소프트웨어전공 부교수

관심분야 : 지능형 로봇, 임베디드시스템, 운영체제