복잡한 공간에서 그룹화 기반의 실용적 지능형 청소 로봇 알고리즘

Practical Intelligent Cleaning Robot Algorithm Based on Grouping in Complex Layout Space

전 흥 석*, 조 재 욱, 노 삼 혁
(Heung Seok Jeon, Jae Wook Jo, and Sam H. Noh)

Abstract : The random-based cleaning algorithm is a simple algorithm widely used in commercial vacuum cleaning robots. This algorithm has two limitations, that is, cleaning takes a long time and there is no guarantee that the cleaning will cover the whole cleaning area. This has lead to customer dissatisfaction. Thus, in recent years, many intelligent cleaning algorithms that takes into consideration information gathered from the cleaning area environment have been proposed. The plowing-based algorithm, which is the most efficient algorithm known to date when there are no obstacles in the cleaning area, has a deficiency that when obstacle prevail, its performance is not guaranteed. In this paper, we propose the Group-*k* algorithm that is efficient for that situation, that is, when obstacle prevail. The goal is not to complete the cleaning as soon as possible, but to clean the majority of the cleaning area as fast as possible. The motivation behind this is that areas close to obstacles are usually difficult for robots to handle, and hence, many require human assistance anyway. In our approach, obstacles are grouped by the complexity of the obstacles, which we refer to as 'complex rank', and then decide the cleaning route based on this complex rank. Results from our simulation-based experiments show that although the cleaning completion time takes longer than the plowing-based algorithm, the Group-*k* algorithm cleans the majority of the cleaning area faster than the plowing algorithm.

Keywords : cleaning robot, obstacle avoidance, cleaning algorithm, Group-*k*

I. 서론

청소 로봇은 사람을 대신하여 주거 공간이나 사무 공간에 대한 청소를 해주는 차세대 지능형 로봇의 대표적인 예이다. 지금까지 청소 로봇에 대한 많은 연구가 이루어져 최근에는 몇 가지 시제품이 출시되어 실제 가정에서 활용되고 있으며 새로운 신제품들도 시장 출시를 기다리고 있는 상태이다. 그러나 청소로봇이 기존의 사람에 의한 청소 방식을 획기적으로 변화시켜줄 것이라는 소비자의 기대와는 달리, 매우 고가이면서도 기술적으로 부족한 부분들이 많아 대중적으로 널리 활용되지는 못하고 있다.

일례로, 현재 출시되어있는 제품에 대한 청소 알고리즘을 분석해보면 대부분이 랜덤 (random) 방식의 알고리즘에 기반을 두고 몇 가지 형태의 패턴으로 진행하는 청소 알고리즘을 채용하고 있다. 이러한 방식의 청소 알고리즘은 설계 및 구현이 용이하다는 장점이 있지만 청소 시간이 매우 오래 걸리는 단점을 가지고 있다.

랜덤 방식의 청소 알고리즘을 개선하기 위하여 많은 연구가 이루어져왔는데 그 중에서 가장 성능이 좋다고 알려져 있는 것은 격자 모양의 지도에 근거한 바둑판식 청소 알고리즘이다[2-4]. 바둑판식 청소 알고리즘은 청소 대상을 격자 방식으로 왕래 하면서 청소하기 때문에 랜덤 방식의 청소 알고리즘에 비해 획기적인 성능 향상을 가져오게 된다.

그러나 바둑판식 청소 알고리즘도 장애물이 없거나 적은 공간에서는 좋은 성능을 보여주지만 장애물이 많아 복잡한 공간에 대한 청소 시에는 장애물 처리를 위한 지연으로 인해 성능 저하 현상이 발생한다[2-4]. 실제로 청소의 대상이 되는 실내 구조는 매우 비정형적이고 동적인 특징을 가지고 있다. 우리의 주거 구조는 그 크기나 구조가 매우 다양하다. 더욱이 실내 구조에 배치된 식탁, 쇼파, 냉장고, 테이블 등의 여러 장애물들의 배치는 유동적일 수 있어 그 어려움을 더하게 된다.

이로 인해 청소 로봇이 사람을 대신하여 청소를 해주기는 하지만 청소 로봇이 청소하는데 소요되는 시간은 사람이 청소하는 시간에 비해 오랜 시간이 필요하다. 더욱이 장애물이 복잡하거나 벽면, 구석, 낮은 장애물의 아랫부분, 장애물의 윗부분 등에 대해서는 사람이 마무리를 해주어야 만족할 만한 청결 수준에 도달 할 수 있다[11].

또한, 기존에 판매된 청소로봇의 구매자 형태를 분석해본 결과 대부분의 구매자는 상식적으로 예상되는 직장 여성이 아니라 주부였다는 사실을 주목해볼 필요가 있다[11]. 이는 청소로봇의 사용자가 청소 로봇을 구동하고 항상 외출하는 것이 아니라 동일한 공간에 함께 공존하는 경우가 많다는 것이다. 따라서 청소 로봇이 사람의 손을 전적으로 대신하여 모든 것을 청소해주는 것을 기대하기 보다는 사

* 책임저자(Corresponding Author)
논문접수 : 2005. 10. 12., 채택확정 : 2006. 3. 22.
전흥석 : 건국대학교 컴퓨터응용과학부(hsjeon@kku.ac.kr)
조재욱, 노삼혁 : 홍익대학교 컴퓨터공학과
(mrtajo@cs.hongik.ac.kr/samhnoh@hongik.ac.kr)
※ 이 논문은 2004년도 건국대학교 학술진흥연구비 지원에 의한 논문임.

람과 로봇의 적절한 협력 시스템을 갖추는 것이 더 실용적일 수 있다. 예를 들어 청소 로봇이 장애물이 적은 부분을 먼저 청소하고, 사람이 장애물이 복잡한 공간을 이후에 혹은 동시에 마무리하면 청소 로봇이 청소를 완료하는데 소요되는 시간은 복잡한 공간을 고려하지 않았을 경우의 청소 완료시간과 동일하다할지라도 실질 청소 완료시간은 더 단축될 수 있다. 따라서 청소 로봇이 사람과의 협력을 전제하에 효율을 높이기 위해서는 가능한 빠른 시간 내에 전체 대상 공간 중에서 복잡하지 않은 많은 부분에 대해 청소를 완료하는 것이 바람직하다.

이를 위해 본 논문에서는 전체 대상 공간의 장애물들을 복잡성에 의해 그룹으로 형성하고, 복잡하지 않은 그룹부터 청소를 진행함으로써 전체 청소 구간에 대해 가능한 빠른 시간 내에 많은 면적이 청소 되어 인간과 상호 협력을 통해 청소 작업을 완료할 수 있는 새로운 청소 알고리즘을 제안한다.

II. 관련 연구

지금까지 바둑판식 알고리즘의 장애물 처리에 관한 몇 가지 형태의 개선 시도들이 이루어지고 있다[4-9]. 대표적인 것이 전체 공간을 장애물이 적은 n개의 공간으로 분할하여 개별 공간에 바둑판식 알고리즘을 적용한 후 다시 연결하는 것이다[4,5].

이와 유사하게 전체 공간을 그래프 이론에 적용하여 분할된 그룹간의 연결을 그래프의 최단 방문 경로 해법을 이용하는 연구도 이루어지고 있다[6].

이러한 바둑판식 청소 알고리즘을 개선하기 위한 연구들의 공통적인 목표는 장애물에 대한 효율적인 처리를 통해 전체 청소 완료 시간을 단축하는 것이다. 즉, 이들 연구들의 공통적인 관심사는 청소 로봇 단독으로 전체 대상 공간에 대한 청소를 완료하는데 소요되는 시간을 단축하는 것이다.

그러나 본 논문에서는 기존의 관련 연구들과는 달리 사람과 로봇의 협력을 통해 실질적인 청소 완료 시간을 단축하기 위한 새로운 청소 알고리즘인 Group-*k* 알고리즘을 제안한다. Group-*k* 알고리즘에 대한 자세한 설명은 다음절에서 기술한다.

III. Group-*k* 알고리즘

이 절에서는 본 논문에서 제안하는 Group-*k* 알고리즘에 관해 설명한다. Group-*k* 알고리즘은 크게 세 단계에 걸쳐서 동작한다. 첫 번째 단계에서는 기본적인 장애물 그룹화 방법에 의해 여러 장애물들을 하나의 그룹으로 형성하고, 두 번째 단계에서는 생성된 각 그룹의 적절성을 판단하여 이를 조절하는 단계이다. 마지막 단계는 각 그룹들의 복잡성에 의한 우선순위에 따라 청소를 진행하는 것이다. 자세한 내용은 다음의 각 절에서 설명한다.

1. 장애물의 그룹화

첫 번째 문제는 과연 어떤 장애물들을 동일한 그룹으로 설정할 것인가이다. 본 논문에서 장애물들을 그룹화하려는 이유는 장애물이 복잡한 공간을 그룹화 하여 복잡하지 않은 공간부터 먼저 청소를 하고, 복잡한 공간은 나중에 청소를 함으로써 가능한 빠른 시간 내에 많은 면적이 청소될 수 있도록 하는 것이다. 따라서 그룹은 임의로 설정하는 것보다는 그림 1과 같이 인접한 복잡한 장애물들을 하나의 그룹으로 형성하는 것이 바람직할 것이다.

이러한 목표에 부합하기 위하여 본 논문에서는 장애물들의 그룹을 형성하는 방법으로 nearest neighbour 알고리즘을 이용한다. Nearest neighbour 알고리즘은 원래 traveling salesperson problem을 해결하기 위한 알고리즘으로서 본 연구에 적용하기 위해서는 약간의 수정이 필요하다. 구체적으로, nearest neighbour 알고리즘은 전체 노드의 방문 순서를 결정하기 위하여 시작 노드로부터 가장 인접한 노드를 선택하고, 이후의 선택 대상에서 이미 선택 되었던 노드들을 제외하여 선택함으로써 전체 노드를 하나의 링크로 연결하는 과정이다. 그러나 본 논문에서는 그룹을 형성하기 위하여 각 장애물로부터 가장 인접한 노드들을 선택하여 가장 인접한 노드들 간에 링크를 생성한다. 즉, 모든 노드에 대한 최 인접 노드의 선택 대상에 기존의 선택 여부와 관계없이 모든 노드들이 대상이 된다. 모든 노드에 대한 가장 인접한 노드를 선택하게 되면 링크를 통해 연결된 노드들을 하나의 그룹으로 설정한다. 이러한 과정을 알고리즘으로 정리하면 그림 2와 같다.

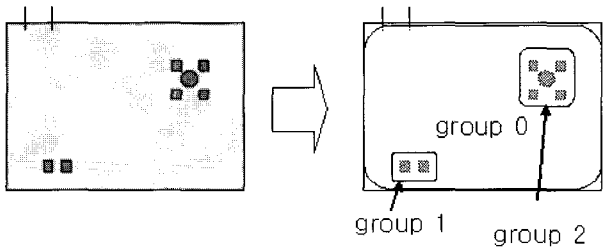


그림 1. 인접 장애물의 그룹화.
Fig. 1. Grouping of neighbour obstacles.

Algorithm *Basic_Group* :

Step 1: Make two sets of nodes, set A and B, and make a set of groups, set G

Step 2: Put all nodes into set A and B

Step 3: Pick a arbitrary node a in set A

Step 4: Search the node b in set B which is the closest to the node a, and make the link ab between the nodes a and b

Step 5: **If** (a group g_i which has the link containing a or b node exists in set G) **then**
append the link ab to the group g_i
Else
generate new group g_{new}
append the link ab to the group g_{new}

Step 6: Repeat step 3 until all nodes are in set A is empty.

그림 2. *Basic_Group* 알고리즘.
Fig. 2. *Basic_Group* algorithm.

2. 그룹의 복잡성

Basic_Group 알고리즘에 의하여 장애물의 그룹화를 진행하면 많은 인접한 장애물들이 각각의 그룹으로 형성된다. 하지만 *Basic_Group* 알고리즘에 의한 장애물들의 그룹화 결과가 항상 적절한 것은 아니다. 때로는 최 인접 노드이기는 하지만, 두 노드를 포함한 그룹이 전체 면적을 모두 차지하거나, 복잡한 노드들을 집약하여 장애물들을 그룹화 하는데 적절하지 않을 수도 있다. 예를 들어 그림 3과 같은 경우를 고려해보자.

그림 3에서 a, b, 그리고 c는 각각 장애물을 의미한다. 위의 그림과 같은 실내에 *Basic_Group* 알고리즘을 적용하면 장애물 a는 b를, b는 a를, c는 b를 최 인접 장애물로 선택하여 a, b, c 모두 하나의 그룹으로 묶여진다. 그러나 직관적으로 판단하였을 때 a와 b가 하나의 그룹을 형성하고, c는 그룹에서 제외되는 것이 바람직하다. 그러므로 *Basic_Group* 알고리즘을 통한 결과를 최종적인 그룹으로 사용하기 위해서는 그 적절성을 판단하고 조정하는 단계가 필요하다.

그룹의 적절성이란 그룹 내의 장애물들의 구조가 충분히 복잡하여 그 그룹에 대한 청소를 덜 복잡한 공간에 비해 나중에 진행하는 것이 바람직한가에 대한 판단의 문제이다. 이 문제를 해결하기 위하여 본 논문에서는 그룹의 복잡성 ($C(g_i)$: Complex rank of group i) 이라는 개념을 도입한다.

그룹의 복잡성을 정의할 수 있는 방법에는 여러 가지가 존재할 수 있겠지만, 본 논문에서는 실내 디자인 분야나 영상처리 분야에서 화면의 복잡한 정도를 계산하는데 사용되는 히스토그램 방법을 사용한다[10]. 히스토그램은 영상의 명암 값 프로파일을 보여주기 위해 사용되는 도구이다. 히스토그램은 영상의 전체 명암 값의 분포를 담을 수 있는 도구로서 이를 이용하면 영상의 명암도 분포 상태를 알 수 있다. 구체적으로 히스토그램 구성 방법은 0부터 255까지의 명암 값을 인덱스로 하고, 영상을 구성하고 있는 각 화소의 명암 값에 해당하는 개수를 빈도수로 간주하여 1씩 증가시키는 방법을 사용한다. 이때 0은 검은색을 의미하고, 255는 흰색을 나타낸다.

실내 구조도를 히스토그램으로 표현하는 방법은 실내에 존재하는 장애물을 검은색 픽셀, 청소 대상 공간이 되는 바닥을 흰색 픽셀들의 집합으로 가정하여 처리하면 된다. 그러면 전체 실내 구조도는 흰색과 검은색 등 두 색의 분포로 이루어지게 되며 이때 검은색 혹은 흰색에 대한 픽셀 값의 분포도를 측정하여 이미지의 복잡도를 판단할 수 있다. 픽셀에 대한 색상 분포 히스토그램을 분석하면 평균값이나 표준 편차 등의 영상에 관한 다양한 정보를 정량적으로 분석할 수 있는데 이 값들 중에서 본 논문에서 의미가 있는 것은 평균값(mean)이다. 평균값이 크면 클수록 이미지 상에 분포되어 있는 검은색 픽셀 수가 적은 것이며, 값이 적을수록 검은색 값이 많이 분포한다는 의미이다. 따라서 평균값이 작을수록 실내 공간이 복잡한 구조임을 나타내고, 값이 클수록 단순한 구조라고 판단할 수 있다[10].

히스토그램에 의하여 복잡성을 판단할 경우 그림 4와 같이 동일한 장애물이라도 그룹의 전체 면적이 다를 경우에

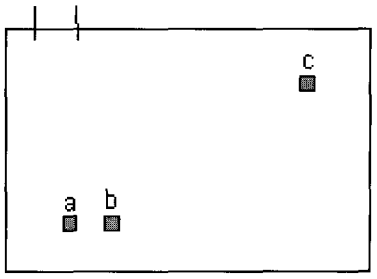
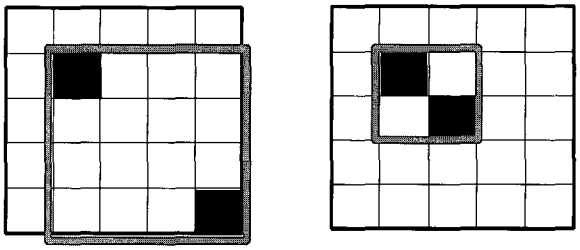
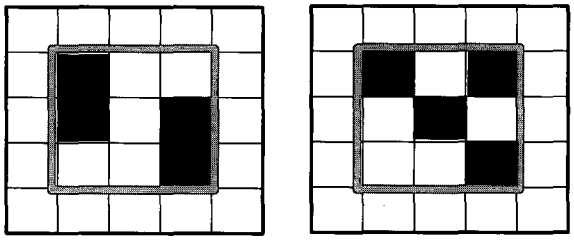


그림 3. *Basic_Group* 알고리즘의 문제점.
Fig. 3. A problem of *Basic_Group* algorithm.



(a) 복잡성이 낮은 그룹 (b) 복잡성이 높은 그룹

그림 4. 히스토그램에 의한 복잡성 판단의 적절한 예.
Fig. 4. An appropriate example of Histogram for complexity.



(a) 단순한 장애물 경우 (b) 복잡한 장애물 경우

그림 5. 히스토그램에 의한 복잡성 판단의 부적절한 예.
Fig. 5. An inappropriate example of histogram for complexity.

는 평균값이 달라지기 때문에 두 그룹의 복잡성이 달라져서 히스토그램의 평균값이 그룹의 적정성을 판단하는 타당성 있는 기준임을 알 수 있다.

그러나 히스토그램을 이용한 복잡성의 분석은 전체 대상 그룹의 면적과 그룹 내의 장애물의 전체 면적은 동일하나, 그 구조가 다를 경우에 문제점이 발생한다. 예를 들어 그림 5와 같은 경우를 살펴보자. 그림 5를 보면 직관적으로 당연히 (a) 보다는 (b)의 구조가 더 복잡함을 알 수 있다. 그러나 히스토그램은 두 가지 경우 모두 동일한 색상 분포 값을 나타내게 된다.

이것은 히스토그램이 장애물의 지리적 위치나 모양은 전혀 고려하지 않기 때문에 발생하는 문제이다. 그러므로 장애물의 복잡성을 측정하기 위해 히스토그램에 의한 색상의 분포만을 고려하면 합리적이지 못한 판단을 할 수가 있다.

따라서 본 논문에서는 히스토그램의 평균값 이외에 장애물의 외양적 구조를 추가로 고려하여 그룹의 장애물 복잡

성 산출 과정을 보완한다. 장애물의 외양적 구조는 장애물이 동일한 면적 분포를 차지하지만 모양이 다를 경우를 구별하기 위한 것이다.

장애물의 외양적 구조를 나타내는 요소는 여러 가지가 가능할 수 있겠지만 본 논문에서는 그룹 내의 장애물들의 전체 꼭짓점의 수에 따라 장애물의 구조적인 복잡성을 판단한다. 즉, 꼭짓점이 많은 다각형일수록 복잡하다고 판단하고, 적을수록 단순한 구조로 판단하는 것이다. 원 모양의 장애물도 세부적으로는 사각형 모양의 단위 셀에 맵핑 되므로 복잡한 다각형의 구조로 인식된다. 장애물의 꼭짓점을 고려할 경우 그림 5에서 왼쪽의 (a) 그림은 전체 8개의 꼭짓점을 가지고, 오른쪽 (b)의 그림은 16개의 꼭짓점을 가지게 되므로 오른쪽 그림이 왼쪽 그림에 비해 더 복잡한 것으로 판단할 수 있는 근거를 제공해준다.

정리하면, 본 논문에서는 그룹의 복잡성을 히스토그램의 평균값과 장애물의 외양적 구조를 나타내는 꼭짓점의 수를 조합하여 판단한다. 문제는 서로 다른 측정 단위인 히스토그램의 평균값과 장애물의 꼭짓점의 수를 어떻게 결합할 것인가이다. 이를 위해 본 논문에서는 각각의 두 요소들을 최소 값이 0이고 최대 값이 1이 되는 정규화된 비율로 변환하여 두 비율의 평균값을 복잡성으로 결정한다. 이때 복잡성 값 0의 의미는 아무런 장애물이 없는 상태이고, 1은 가장 복잡한 구조임을 나타낸다.

예를 들어 특정 그룹 (g_i)에 대한 히스토그램의 평균값 ($H_{mean}(g_i)$)의 경우 0에서 255사이의 값을 산출하게 되는데 이를 0에서 1사이의 비율로 변환한다. 주의해야할 것은 복잡성은 0부터 1사이의 값을 갖게 되는데 이때 1이 가장 복잡한 구조를 나타낸다. 그런데 히스토그램의 경우에 0은 검은색을 나타내어 가장 복잡한 경우를 나타내고, 255는 흰색을 나타내어 가장 단순한 경우를 나타내므로 히스토그램의 평균값이 0일경우를 1로 255일경우를 0으로 변환해야 한다. 히스토그램 값이 127.5일 경우는 변환 값이 0.5가 된다. 따라서 만약 히스토그램의 평균값이 0과 가까워질수록 변환 값은 1과 가까워지게 되어 복잡함을 나타내고, 히스토그램 평균값이 255에 가까워질수록 변환 값은 복잡성이 낮음을 나타내는 0에 가깝게 변환된다.

둘째, 장애물의 꼭짓점의 수는 그룹 내 가능한 꼭짓점 수의 합 ($A_{sum}(g_i)$)에 대한 그룹 (g_i)내 장애물들의 꼭짓점 수의 합 ($A_{obstacle}(g_i)$)의 비율로써 산정한다. 만일 그룹 내 전체 꼭짓점 수의 합과 장애물의 꼭짓점 수의 합이 동일할 경우를 1로 산정하고, 그룹 내의 장애물이 없는 경우는 0이 된다. 그룹 내 장애물의 꼭짓점 수가 전체 꼭짓점 수의 반이 될 때의 비율을 0.5로 평가한다.

따라서 특정 그룹 (g_i)의 복잡성 $C(g_i)$는 위에서 언급한 두 요소들의 평균값으로 결정되며 이러한 내용을 수식으로 정리하면 다음과 같다.

$$C(gi\ ) = \frac{(1 - \frac{Hmean(gi\ )}{255}) + (\frac{Aobstacle(gi)}{Asum(gi)})}{2} \qquad (1)$$

물론 이러한 요소들이 특정 공간의 복잡성을 결정하는 절대적인 해결책은 아닐 수 있지만 가능성 있는 하나의 방법이다. 또한, 만일 더 적절한 복잡성의 산출 방법이 제안될 경우 해당 방법을 적용할 수 있도록 복잡성의 계산이 모듈화 되어 설계될 수 있으므로 복잡성의 결정 방법은 본 논문에서 제안하는 Group-k 알고리즘의 기본적 생각 및 장점과는 독립적인 문제이다.

3. 그룹의 해제 및 통합

다음 단계는 그룹의 복잡성 C 를 이용해서 전체 그룹들을 조절하는 것이다. 그룹 조절 작업은 크게 두 단계로 이루어진다. 첫 번째 단계는 그룹의 복잡성이 특정한 기준 k 이하일 경우 해당 그룹 내의 장애물에 대한 그룹화가 적절하지 못하다고 판단하여 해당 그룹을 축소 혹은 해제하는 것이다. 이를 위한 구체적인 방법은 그림 6과 같다.

먼저 실내 공간의 전체 그룹에 대해 순차적으로 각 그룹별 복잡성을 계산한다. 만일 특정 그룹의 복잡성이 k 이하일 경우에는 해당 그룹 내에서 가장 긴 링크를 해제하고 남은 장애물들로 그룹을 재형성하여 그룹의 복잡성을 다시 계산한다. 장애물들의 그룹을 형성하는 과정에서 각 장애물들은 전체 장애물들 중에서 가장 인접한 노드를 선택하도록 하였기 때문에 가장 긴 링크를 해제하면 하나의 노드, 즉 하나의 장애물과 나머지 장애물들의 그룹으로 분리된다. 이러한 과정을 그룹의 복잡성이 k 이상이 될 때까지 모든 그룹에 대해 반복한다. 만약 한 그룹 내의 모든 링크가 해제되면 해당 그룹 자체가 해제된다. *Release_Group* 과정을 거치고 나면 남은 모든 그룹들의 복잡성은 적어도 k 이상이 된다.

Algorithm *Release_Group* :
Step 1: Compute the complexity $C(g_i)$ of group g_i
Step 2: If ($C(g_i) < k$) **then**
Release the longest link in the group g_i
Step 3: Repeat Step 1 until $C(g_i) > k$
Step 4: Repeat Step 1 for all the groups in G

그림 6. *Release_Group* 알고리즘.
Fig. 6. *Release_Group* algorithm.

Algorithm *Merge_Group* :
Step 1: Select the Nearest Neighbour group of the group g_i in G
Step 2: Make a virtual group v of g_i and its nearest neighbour group
Step 3: Compute the complexity $C(v)$ of group v
Step 4: **If** ($C(v) \geq k$) **then**
Merge the group g_i and its nearest neighbour group
Else
Release the group v
Step 5: Repeat step 1 for all the groups in G

그림 7. *Merge_Group* 알고리즘.
Fig. 7. *Merge_Group* algorithm.

Algorithm Group-k :

Step 1: Build a map for the space to be cleaned and Initialize the position of the cleaning robot
Step 2: Call the procedure *Basic_Group*
Step 3: Call the procedure *Release_Group*
Step 4: Call the procedure *Merge_Group*
Step 5: Sort and make a list L for all the groups by the complexity $C(g_i)$ in increasing order
Step 6: Clean the area which has not been included in any group.
Step 7: Pick the group g_i from the head of the list L
Step 8: Move the cleaning robot to the group gi by the shortest path search algorithm
Step 9: Clean the group gi by the plowing_method algorithm
Step10: Repeat step 7 until all groups are in list L is empty.

그림 8. Group-k 알고리즘.
Fig. 8. Group-k algorithm.

다음 단계는 그룹의 범위를 확장 조절하는 것이다. 이것은 생성된 소규모 그룹의 수가 너무 많을 경우 전체 면적에서 그룹에 속하지 않은 부분의 구조가 복잡해지는 것을 방지하기 위한 것이다. 이 과정도 첫 번째 단계의 *Basic_Group* 알고리즘과 유사하게 nearest neighbour 알고리즘을 응용한다. 먼저 각 그룹별 가장 인접한 그룹을 선택하여 두 그룹을 가상으로 병합한 후 병합한 가상 그룹의 복잡성을 계산한다. 만일 가상 그룹의 복잡성이 k 이상이 되면 두 그룹을 실제로 합병하게 된다. 이 과정은 그림 7에서 보여준다.

4. 청소 알고리즘

실내 공간의 장애물들에 대한 그룹 조절 작업이 모두 이루어지면 이제 생성된 그룹을 기반으로 청소를 해야 할 차례이다. 청소는 모든 그룹들의 복잡성을 계산하여 이를 바탕으로 복잡성이 낮은 순서대로 정렬하는 작업부터 진행된다. 청소는 먼저 그룹에 속하지 않은 전체 구역을 먼저 청소하고, 그 다음부터는 복잡성이 낮은 그룹 순서대로 청소를 실시한다. 만약 복잡성이 동일하다면 이전 청소한 그룹과 가장 가까운 거리에 있는 그룹부터 청소를 하는 것이 바람직할 것이다. 물론 그룹 내에는 바둑판식 알고리즘을 적용하여 청소한다. 이러한 과정을 정리한 것이 그림 8의 알고리즘이다.

IV. 성능 평가

이 절에서는 본 논문에서 제안한 새로운 Group-k 청소 알고리즘의 성능을 평가한다. 성능 평가는 시뮬레이션 기반으로 기존의 바둑판식 청소 알고리즘과 본 논문에서 제안한 Group-k 청소 알고리즘의 성능을 비교 분석한다. 먼저 실험에 사용된 시뮬레이터에 대해 살펴보고 다음으로 실험 내용 및 결과를 분석한다.

1. 시뮬레이터의 작성 및 환경 설정

시뮬레이션은 레드헷 리눅스 8.0을 탑재한 펜티엄 VI 시스템에서 이루어졌다. 시뮬레이터는 C 언어를 이용하여 작성하였다. 시뮬레이터에는 기존의 대표적인 청소 알고리즘인 바둑판식 알고리즘과 본 논문에서 제안하는 Group-k 알고리즘을 각각 구현하였다.

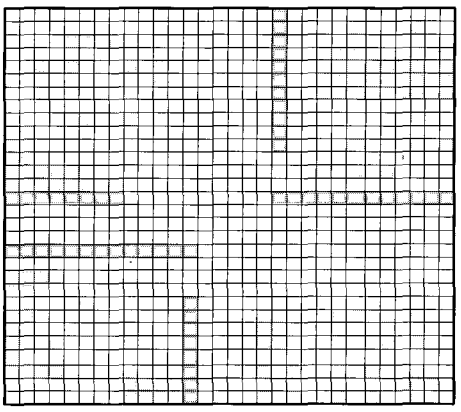


그림 9. 청소 알고리즘 분석을 위한 대상 실내 구조 맵.
Fig. 9. A target map of indoor layouts for analysis of cleaning algorithms.

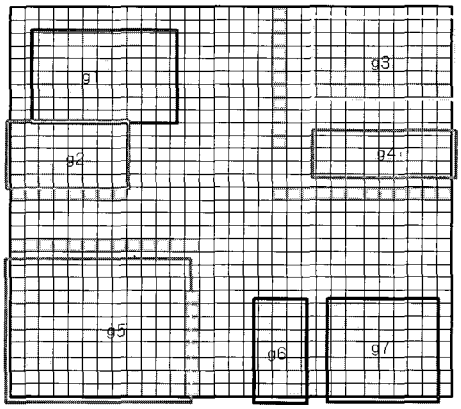


그림 10. *Basic_Group* 수행 결과(C(g1)=0.46, C(g2)=0.34, C(g3)=0.44, C(g4)=0.74, C(g5)=0.24, C(g6)=0.56, C(g7)=0.57).
Fig. 10. Result of *Basic_Group* (C(g1)=0.46, C(g2)=0.34, C(g3)=0.44, C(g4)=0.74, C(g5)=0.24, C(g6)=0.56, C(g7)=0.57).

실험을 위해 그림 9와 같은 실내 구조를 위한 맵을 사용한다. 그림 9에서 진한 회색 부분은 벽을 나타내고, 여린 회색 부분은 장애물을 나타낸다. 그 외 부분은 청소 대상 구역을 나타낸다. 로봇의 초기 위치는 가장 왼쪽 윗부분인 좌표(1,1)로 설정한다. 맵에서 한 좌표는 실제의 공간에서 약 10cm를 나타내도록 설계한 것이다. 알고리즘의 성능 판단을 위해 로봇의 이동 속도는 기존 청소 로봇의 평균 이동 속도를 참고하여 0.17m/s라고 설정하였으며, 방향 전환을 위한 회전 지연 시간은 90o 회전 시 1초, 180o 회전 시 2초로 설정하였다.

2. 실험 및 결과 분석

가장 먼저 Group-k 알고리즘을 이용하여 장애물이 복잡한 실내 공간에 대한 그룹화 실험을 진행하였다. *Basic_Group* 알고리즘을 수행하면 그림 10과 같이 모두 7개의 그룹이 생성된다.

생성된 각 그룹에 대한 복잡성 값은 0.24부터 0.74에 이르기까지 각 그룹의 장애물 구조에 따라 다양하게 나타난다. 다음 단계인 *Release_Group* 알고리즘을 적용하기 위해 그룹 형성을 위한 임계 값 k를 0.5로 설정하였다. 이것은 복잡성 값의 변위인 0부터 1까지의 중간 값이기 때문이다. 그림 12는 k에 0.5를 적용하여 *Release_Group* 알고리즘을 수행한 결과이다. 그룹 g1에서는 가장 먼 여러 장애물 들 중에서 임의의 하나를 제외하고 그룹을 재형성하게 된다. 그룹 g5는 재형성된 그룹의 복잡성이 k보다 커질 때까지 반복하여 적용한 결과이다. 이제 조절된 각 그룹의 복잡성은 적어도 0.5의 값을 가지게 된다.

다음은 그룹화의 마지막 단계인 *Merge_Group* 알고리즘을 수행할 단계이다. 각 그룹과 가장 인접한 그룹들과의 병합을 통해 k 값인 0.5보다 큰 복잡성을 나타낼 수 있는 것은 그림 11에서 그룹 g6와 g7의 병합뿐이다. 그룹 g6와 g7을 병합하면 복잡성 C는 0.51이 된다. 최종적으로 이들을 통합하면 최종적으로 그림 12와 같은 그룹으로 결정된다. 그림 12에 최종적으로 결정된 그룹을 복잡성 값에 대한 오름차순으로 g1부터 g5까지 제시하였다. 이제 앞 절에서 제안한 Group-k 알고리즘에 의하여, 그룹에 속하지 않은 부분을 가장 먼저 청소하고, 다음은 복잡성 값의 순서에 의해서 g1부터 g5까지 차례대로 청소가 진행된다.

그림 13은 장애물이 복잡한 실내 구조에 대한 바둑판식 알고리즘과 본 논문에서 제안한 Group-k 알고리즘의 실험 결과를 보여준다. 그래프는 시간의 흐름에 따른 청소가 완료된 면적의 비율을 나타낸다. 실험 결과에 의하면 Group-k 알고리즘을 적용하였을 때의 청소 완료 시간이 바둑판식 알고리즘을 적용하였을 때 보다 약 19.66% 단축되어 성능이 향상됨을 보여준다. 이것은 바둑판식 알고리즘이 앞 절에서 문제를 제기한 것처럼 장애물 처리를 위해 장애물이 존재하는 부근의 노드들에 대한 방문 횟수가 많아지게 되기 때문이다. 동일한 장애물이라도 Group-k 알고리즘은 지역적으로 분포하는 장애물들을 그룹화 하여 이동의 반경을 제한하지만 바둑판식의 경우는 그 제한이 없이 이동 경로상의 장애물들을 폭 넓게 처리하기 때문에 전체 지체 시간이 더 많이 발생하게 된다. 이것은 바둑판식 알고리즘의 청소 완료 율 곡선이 장애물 처리를 위해 여러 번 계단식 분포를 나타내는 것을 보면 알 수 있다.

또한 Group-k 알고리즘은 전체 청소 완료시간 뿐만 아니라 바둑판식 알고리즘보다 단기간에 많은 부분을 청소할 수 있음을 보여준다. 예를 들어 그림 13을 보면 바둑판식 알고리즘을 이용하여 전체 공간에 대해 약 40-50% 정도의 청소를 진행할 시점에 Group-k 알고리즘은 이미 80-90% 정도의 완료 율을 알 수 있다. 따라서 Group-k 알고리즘이 적용된 청소로봇과 함께 사람이 직관적으로 알 수 있는 복잡한 장애물 그룹에 해당하는 지역을 함께 청소해준다면 실질적인 청소 시간은 실험 결과 보다 더 감소될 수 있다.

마지막으로, 본 논문에서 제안한 Group-k 알고리즘은 복잡한 공간을 효율적으로 처리하기 위해 설계된 것이지만 복잡하지 않은 공간에서는 바둑판식 알고리즘과 동일하게 작동한다. 장애물이 없을 경우 바둑판식 알고리즘과 Group-k 알고리즘은 동일하게 전체 노드를 거의 1회씩 방문 하게 되어, 청소 완료 율을 나타내는 곡선 분포는 시간의 흐름에 따라 단조 증가하는 직선 분포를 보이게 된다. 이에 관한 실험 결과는 지면 관계상 생략하였다. 따라서 Group-k 알고

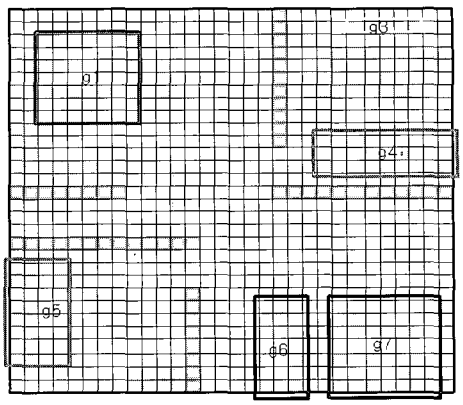


그림 11. *Release_Group* 수행 결과(k=0.5, C(g1)=0.52, C(g3)=0.54, C(g4)=0.74, C(g5)=0.53, C(g6)=0.56, C(g7)=0.57.

Fig. 11. Result of *Release_Group* (k=0.5, C(g1)=0.52, C(g3)=0.54, C(g4)=0.74, C(g5)=0.53, C(g6)=0.56, C(g7)=0.57.

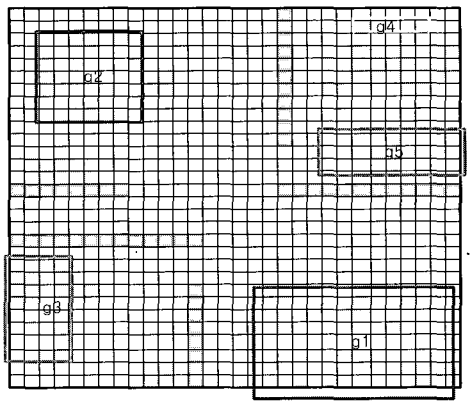


그림 12. *Merge_Group* 수행 후 최종 그룹화 결과(C(g1)=0.51, C(g2)=0.52, C(g3)=0.53, C(g4)=0.54, C(g5)=0.74).

Fig. 12. Final grouping result of *Merge_Group* (C(g1)=0.51, C(g2)=0.52, C(g3)=0.53, C(g4)=0.54, C(g5)=0.74).

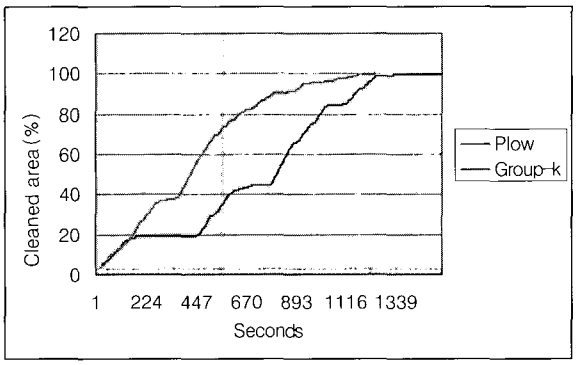


그림 13. 청소 완료율.

Fig. 13. Cleaning completion rate.

리즘은 단순한 공간에서는 바둑판식 알고리즘과 동일하게 작동하여 대등한 성능을 나타내고, 복잡한 공간에서는 전체 청소 대상에서 복잡한 부분을 그룹화하고 복잡함의 정도에 따라 청소의 순서를 달리하게 되므로 공간의 복잡도와 관계없이 다양한 공간에서 적용될 수 있는 실용적인 알고리즘이다.

V. 결론 및 향후 연구

본 논문에서는 청소 로봇의 대표적인 바둑판식 청소 알고리즘의 문제점을 해결하는 새로운 Group-*k* 알고리즘을 제안하였다. 바둑판식 알고리즘은 장애물이 적은 공간에서는 효율적이나 장애물이 많아 복잡한 공간에서는 문제점을 가지고 있다. 이를 해결하기위하여 Group-*k* 알고리즘은 전체 공간을 복잡성에 따라 그룹으로 형성하고 복잡성이 낮은 순서에 따라 청소를 진행한다. Group-*k* 알고리즘은 시뮬레이션에 기반을 둔 성능 평가 결과 복잡한 공간에 대해 바둑판식 알고리즘보다 빠른 시간 내에 많은 구역에 대한 청소를 완료하는 등 좋은 성능을 나타내었다.

Group-*k* 알고리즘에서는 그룹을 병합하는 과정에서 두 그룹을 가상으로 병합한 그룹의 복잡성이 *k* 이상이 되면 하나의 큰 그룹으로 병합하게 된다. 그러나 큰 그룹 내에서의 효율적인 청소를 위해서는 단일 그룹으로의 병합이 아닌 그룹의 레벨을 구분하여 멀티 레벨의 그룹을 형성하고 이에 따라 그룹 내에서도 소그룹들의 복잡성 등의 다양한 기준에 의해 조금 더 섬세하게 처리하는 전략이 필요할 것이다.

마지막으로, Group-*k* 알고리즘은 청소 대상이 되는 실내 공간에 대한 맵을 미리 알고 있다는 가정을 하고 있기 때문에 실제 구현을 위해서는 주변 공간에 대한 정보를 수집하여 맵을 구축하는 연구가 함께 이루어져야 할 것이다.

참고문헌

[1] G. S. Sukhatme and M. J. Mataric, "Embedding robots into the internet," *Communications of the ACM*, vol. 43, no. 5, May. 2000.

[2] G. Schmidt and C. Hofner, "An advanced planning and navigation approach for autonomous cleaning robot operations," *In Proceedings of the IEEE Int. Symposium on Intelligent Vehicles*, pp. 364-369, 1995.

[3] I. ULirich, F. Mondada, and J.-D. Nicoud, "Autonomous vaccum cleaner," *Robotics and Autonomous Systems*, vol. 19, pp. 233-245, 1997.

[4] R. Neumann de Carvalho, H. A. Vidal, P. Vieria, and M. I. Riberio, "Complete coverage path planning guidance for cleaning robots," *In Proceedings of the IEEE Int. Symposium on Industrial Electronics*, vol. 2, 1997.

[5] S. C. Wong and B. A. MacDonald, "A topological coverage algorithm for mobile robots," *In Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, Las Vegas, Nevada, Oct. 2003.

[6] 윤상훈, 박세훈, 최병준, 이연정, "Path planning for cleaning robots: a graph model approach," *In Proceedings of the International Conference on Control, Automation and Systems*, Cheju National Univ. Jeju, Korea, pp. 2199-2202, October 17-21, 2001.

[7] S. Koenig and Y. Liu, "Terrain coverage with ant robots: A simulation study," *In Proceedings of the ACM AGENTS'01*, Montreal, Quebec, Canaga. May 28-June 1, 2001.

[8] A. Howard, S. Siddiqi, and G. S. Sukhatme, "An experimental study of localization using wireless ethernet," *In Proceeding of the 4th International conference on Field and Service Robotics*, July 14-16, 2003.

[9] M. A. Batalin and G. S. Sukhatme, "Coverage, exploration and deployment by a mobile robot and communication network," *In Proceedings of the 2nd International Workshop on Information Processing in Sensor Networks,* Palo Alto Research Center(PARC), Palo Alto, April 22-23, 2003.

[10] H.-J. Lee, and J.-S. Lee, "Study of digital analysis efficiency through a complexity analysis," *Journal of Korea Institute of Interior Design*, vol 31, Apr. 2002.

[11] 오연택, "청소로봇의 소비자 조사," In Proceedings of 2nd Annual Workshop of Korea Robotics Society, 2005.

전 흥 석

1996년 홍익대학교 컴퓨터공학과(공학사). 1998년 홍익대학교 전자계산학과(이학석사). 2001년 홍익대학교 전자계산학과 (이학박사). 2000년~2001년. (주) 이칼로스 기술담당부사장. 2001년~2002년 (주)이씨앤아이티 기술연구소 소장. 2002년~현재 건국대학교 컴퓨터응용과학부 조교수. 관심분야는 운영체제, 지능형 로봇, 임베디드 시스템.

노 삼 혁

1986년 서울대학교 컴퓨터공학과(공학사). 1993년 메릴린드대학교 컴퓨터과학과(박사). 1993년 George Washington University 객원교수. 1994년~현재 홍익대학교 정보컴퓨터공학부 교수. 관심분야는 운영체제, 임베디드 시스템 관련 시스템 소프트웨어.

조 재 욱

1999년 홍익대학교 컴퓨터공학과(공학사). 1999년~2002년 팅크웨어(주) 연구원. 2005년 홍익대학교 컴퓨터공학과(공학석사). 2005년~현재 (주) JCEntertainment 주임 연구원. 관심분야는 Distributed network system, Robot AI algorithm.