자동 미장 로봇을 위한 스위핑 경로 계획

A planning of Sweeping Path for a Smearing Robot

현 웅 근, 박 상 규
(Woong-Keun Hyun and Sang-Kyoo Park)

Abstract : This paper presents a sweeping path planning algorithm for an autonomous smearing robot. An automatic planner generates a sweeping path pattern by proposed five basic procedures. This algorithm recognizes obstacle on the architectural CAD draft and generates subgoals as tracking points which executes the area filling task based on heuristic approach. A sweeping path is planned by sequentially connecting the tracking points in such a way that (1) the connected line segments should not be crossed, (2) the total tracking points should be as short as possible, and (3) the tracking line should not pass through the obstacle. Feasibility of the developed techniques has been demonstrated on a real architectural CAD draft.

Keywords : smearing robot, automatic planner, sweeping path

I. 서론

최근 로봇의 적용은 공장내에서의 활용에서 벗어나 일반적인 작업으로의 활용으로 확장되고 있다. 특히 원자력 설비 등의 정기적 점검, 건설작업을 위한 미장 가공, 청소 작업, 페인팅 등의 작업의 로봇화가 강력하게 요구되지고 연구되어 지고 있다[4][9][11][14]. 이러한 작업을 위한 경로계획은 작업 영역인 2차원 면 영역의 빈공간을 훑트면서 돌아다니는 동작 계획을 요구하며 이를 스위핑 작업(sweeping task) 경로계획 이라 부른다. 스위핑 작업에 대한 동작 계획 방법은 크게 온 라인(On-Line) 방법과 오프라인(Off-Line) 방법으로 대별 된다. 온 라인 방법은 로봇이 작업 영역주변을 이동하면서 작업 환경을 센서로 인식하여 그것을 직접 동작에 반영하는 방법이다[1]-[3][12][13]. 그러나 이러한 온 라인 작업 방법은 작업시 전체 지도를 이용하지 않기 때문에 로봇이 주어진 시간내에 스위핑 작업의 완료가 보장되지 못하며, 작업 능력이 환경의 형상에 따라 좌우된다[4][5]. 또한 장애물 환경에 따라서는 같은 영역을 맴돌게 되는 데드록(dead-lock) 현상에 빠지게 되는 문제 점도 있다. 이러한 문제로 인해 실제 미장 작업을 하는 기존의 대부분의 로봇은 일정한 동작 패턴을 반복하는 기능만을 갖추고 있으며 기능인 작업자에 의해서 조종되는 반자동 형태이다. 오프 라인 방법은 작업할 환경에 대한 정보를 알고 있다는 가정하에서 스위핑 작업 경로를 생성하는 방법이다. 스위핑 작업 경로의 자동 생성에 관한 연구로 D.Kurabashi 등이 제안한 방법이 있는 바, 이는 청소 로봇 및 미장 로봇을 위한 스위핑 작업 경로 생성시에 장애물과의 충돌회피 경로 계획을 위해 보로노이 다이어그램을 적용한 방법이다[4]. 이 방법의 문제점은 보로노이 다이어그램이 무충돌 경로를 모양공간(configuration space)상에서 생성하는게 아니라 직교좌표계에서 장애물과 작업 영역(workspace) 사이의 중앙선 또는 중심점의 연결로 생성하므로 생성된 경로가 장애물에 너무 인접해서 생성되어 실제 로봇의 주행시 경로제어를 정교하게 해야 하는 문제가 발생할 수 있고, 경우에 따라서는 장애물사이에 생성된 무 충돌 경로를 실제 로봇이 갈수 없는 경우도 발생하게 된다.

본 연구에서는 자율주행 미장 로봇을 위하여 오프라인으로 스위핑 작업 경로계획 방법을 제안한다. 본 연구에서 제안한 방법은 다음과 같은 기능을 갖고 있다. 1) 건설 CAD S/W 와의 인터페이스 및 도면상에서의 장애물 인식, 2) 오프 라인 장애물 지도 구성, 3) 스위핑 작업을 위한 중간 목표점 생성 알고리즘, 4) 중간 목표점의 연속적 추적 알고리즘, 그리고 5) 장애물 회피 알고리즘.

본 논문의 구성은 다음과 같다. 우선 2장에서는 전체 알고리즘의 구성을 설명하며, 3장에서는 스위핑 작업을 위한 중간 목표점 생성 알고리즘을 설명하고, 4장에서는 장애물을 회피하면서 스위핑 경로(sweeping path) 를 생성하는 알고리즘을 설명한다.

II. 환경의 모델링 및 CAD 상에서의 스위핑 작업을 위한 경로 계획 방법

1. 환경의 모델링

스위핑 작업은 2차원 평면상에서 이루어지므로 로봇과 대상 작업물은 2차원 평면상에서 모델링 한다. 로봇의 모델링은 다음과 같다.

1) 이동 로봇은 임의의 다각형으로 모델링 된다.

2) 이동 로봇은 전방위로 이동 가능하고(omnidirectional motion) 자체회전이 가능하다.

접수일자 : 2000. 4. 11., 수정완료 : 2000. 8. 19.
현웅근 : 호남대학교 전자공학과
박상규 : 여수대학교 자동차공학부 기계공학 전공
※ 본 논문은 과학기술부-한구과학재단 지정 여수대학교 설비자동화 및 정보시스템 연구개발센터 연구비 지원에 의한 것임

3) 작업 공간내의 대상물은 다각형 형태로 모델링 된다.

작업 환경내의 장애물은 다각형 형태와 곡선 형태로 나누어 진다. 다각형 형태는 각 꼭지점에 대한 정보를 갖고 있으며, 곡선 형태의 장애물은 일정한 길이의 분해능을 갖는 직선 선분으로 분할되어 그 점들의 집합으로 곡선을 표시한다. 그리고 화면상에 표시할때는 이러한 점들을 커브 피팅(curve fitting)하여 곡선 장애물을 표시한다. 장애물에 대한 데이터 구조는 다음과 같다.

```
Struct object {
        double *x_position;
        double * y_position;
        double radius_x, radius_y;
        int edge_No;
        char status;
      } *object_posture;
```

여기서 status는 장애물이 다각형, 원, 타원 또는 곡선 형태를 나타내는 상태 플랙이며, edge_No는 object가 다각형 형태인 경우 꼭지점의 갯수를 나타낸다. 원이나 타원형인 경우 radius_x, radius_y의 변수가 사용되며, 임의의 곡선인 경우 충분한 분해능(resolution)을 갖는 일정한 선분(segment)으로 등분하여 그 형태 및 위치 데이터를 나타낸다

작업의 대상이 되는 영역은 2차원 유크리디언 공간으로 $W = R^2$ 로 표시한다. O_j 를 W 내에서의 j번째 배치된 정지물체, 즉 고정된 장애물이라 하면 모양공간(configuration space) 에서의 장애물 영역은

$$CO_j = \{q \in C \mid A(q) \cap O_j \neq \phi\} \quad (1)$$

으로 주어진다. 여기서 q 는 이동 로봇의 위치 x, y 및 자세 θ 를 나타내는 집합이며 $A(q)$ 는 x, y, 그리고 q로 결정되는 이동 로봇의 위치 및 자세이다. W 내에 고정 장애물의 갯수를 p라 할때 장애물 영역의 총 공간은 $\bigcup_{i=1}^{p} CO_i$ 이므로 모양공간내의 장애물이 없는 자유공간 C_{free} 는 (2)와 같이 표현된다.

$$C_{free} = C \setminus \bigcup_{i=1}^{p} CO_i \quad (2)$$

여기서 기호 "\"는 집합의 감산을 의미한다. C_{free} 에 포함되는 모든 q에 대하여 $A(q)$ 로 구성되는 영역을 W_{sweep} 이라 기술하고, 이를 스위핑 작업 가능 영역이라 하자. 그러면 W_{sweep} 은 영역 W 내에서 이동 로봇이 물리적으로 진입 가능한 영역으로 표시되며, 스위핑 작업은 W_{sweep} 공간 내에서 수행된다. 그 조건하에서, 경로 τ 가 (3)을 만족할 때 스위핑 작업이 만족된다. 여기서 $d(\tau, A(q))$ 는 경로 τ 와 로봇 $A(q)$ 사이의 거리를 나타낸다.

$$\forall A(q) \in W_{sweep}, d(\tau, A(q)) \leq r \quad (3)$$

여기서 r은 로봇이 스위핑 작업시 스캐닝 폭이다.

2. 경로 계획방법의 구성

본 논문에서는 이동 로봇 1대가 W_{sweep} 을 스위핑 작업을 하는 연속 경로를 생성한다. 스위핑 작업 경로의 생성은 그림 1과 같은 순서로 수행한다.

스위핑 작업을 위한 이동 로봇의 경로 계획 방법은 그림 1과 같이 구축하였다. 전체 모듈은 크게 1) AUTO CAD 와의 소프트웨어적인 링크와 장애물 해석 모듈, 2) 스위핑 경로(sweeping path)를 위한 중간 목표점 생성, 3) 충돌회피 알고리즘, 그리고 4) 최종 경로 생성 등 4개의 모듈로 구성되어 있다. 우선 AUTO CAD 인터페이스 모듈에서는 CAD 언어인 Auto LISP 과 DCL(Dialog Control Library)방식을 활용하여 대화형 메뉴를 제작했으며, 로봇의 형태, 작업영역, 작업결과의 표시 및 경로 데이터를 로봇으로 전송하는 작업이 수행된다. 이 모듈에서는 CAD 의 엔티티(Entity)를 이용하여 검색된 도면정보를 통하여 스위핑 작업 로봇이 활용할 실 데이터를 취득하고 선택된 영역에서 장애물(선, 원, 다각형, 원호 등)에 대한 정보 데이터를 검색한다.

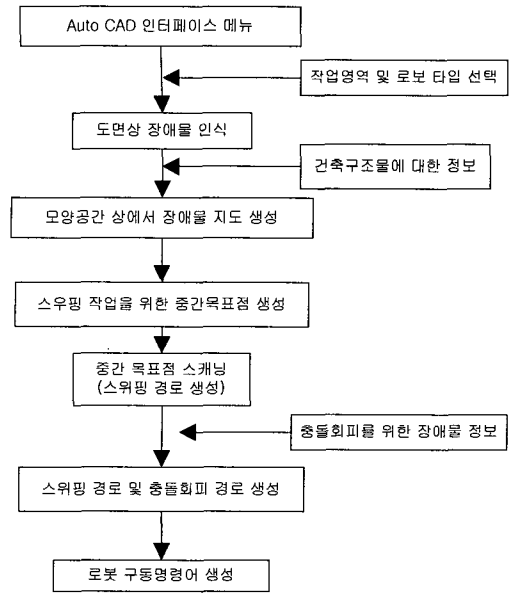


그림 1. 제안된 프로그램의 구성.

Fig. 1. A structure of the proposed method.

장애물 지도형성(Obstacle Map Building) 모듈에서는 전 모듈에서 획득한 장애물 및 작업환경 데이터를 이용하여 설계된 2차원 형태의 로봇에 대하여 x, y, 그리고 θ 인 3차원의 자유 모양공간(Configuration space) C_{free} 를 형성한다. 그리고 다음 모듈인 중간목표점 생성 모듈(Subgoal Generation module)에서는 W_{sweep} 에서 장애물과 로봇의 충돌형태에 따라 스위핑 작업 경로와 중간 목표점을 생성한다. 그리고 스위핑 작업 경로를 생성하는 과정에서 생성된 충돌 경로에 대하여 장애물 충돌 회

피경로가 생성되며, 이 모듈에서 C_{free}에서 장애물을 회피하면서 목표점까지 갈 수 있는 최단 거리의 충돌회피 경로가 생성된다.

III. 중간목표점 생성 및 스위핑 작업 경로 생성 알고리즘

1. 중간목표점 생성 알고리즘

중간 목표점 생성 모듈은 2차원 평면상의 작업 공간에 대하여 스케닝(scanning) 폭 r로 작업영역을 스캔하며, 스캔하는 동안 로봇과 장애물간의 접촉점 그리고 로봇과 작업 영역간의 접촉점을 저장한다. 스캔 폭 r이 작을수록 sweeping 폭이 작아지며 중첩되는 영역이 많아지지만 더욱 촘촘하게 청소나 미장 작업을 하게 된다. 로봇이 선택된 작업 영역을 스캔하는 동안 로봇과 장애물, 그리고 로봇과 작업영역의 벽과의 접촉 위치와 충돌 상태를 저장한다. 이러한 위치는 스위핑 작업 경로를 위한 중간 목표점으로 활용된다. 저장을 위한 중간 목표점의 데이터 구조는 다음과 같다

```
Struct  Subgoal{
            int  position[DIMENSION];
            char status;
            char tracked_flag;
            int obstacleNo;
            } *SubgoalPositions;
```

여기서 position[DIMENSION]은 로봇과 장애물 또는 작업영역 벽과의 접촉위치이며, status는 접촉 형태로써 접촉형태에 따라 벽과 로봇의 좌측이 접촉했을 경우 LEFT_WALL, 로봇의 우측과 벽이 접촉했을 경우 RIGHT_WALL, 로봇의 좌측과 장애물이 접촉했을 경우 LEFT_OBSTACLE, 그리고 우측과 접촉했을 경우 RIGHT_OBSTACLE로 각각 표현한다.

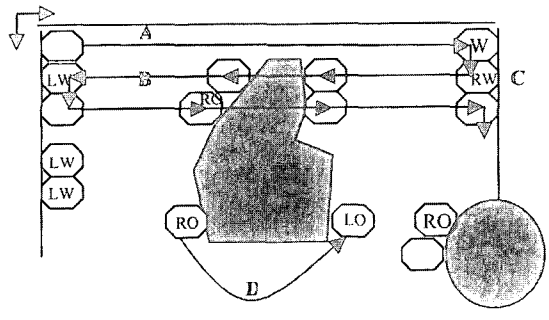


그림 2. 스위핑 경로를 위한 중간목표점 생성방법 (여기서 LW, RW, LO, 와 RO 는 각각 외쪽벽, 오른쪽벽, 왼쪽 장애물, 그리고 오른쪽 장애물과의 접촉 플랙을 의미한다).

Fig. 2. Strategy of the subgoal point generation algorithm for sweeping path. (Where LW, RW, LO, and RO mean Left_Wall, Right_Wall, Left_Obstacle, and Right Obstacle, respectively).

그리고 tracked_ flag은 스위핑 작업 경로의 생성을 위하여 중간목표점을 선택하여 리스트 S에 삽입했을 경우 TRACKED로, 선택이 않된경우 UNTRACKED로 표시한다. 그리고 obstacleNo는 중간목표점을 생성할 때 로봇이 접촉한 장애물의 번호를 표현한다. 중간 목표점 생성을 위해서 로봇은 작업 역영을 좌측 상단에서 우측 하단까지 스캔하며, 로봇의 자세는 0^{o} 를 유지한다. 중간 목표점 생성을 위한 알고리즘은 다음과 같다.

SP를 편의상 중간목표점(subgoal point)이라 하고, i 번째 중간 목표점 데이터 리스트 엔트리를 S_i라 하자. 그리고 리스트 S를 SP의 집합인 S={ $S_1, S_2, \ldots, S_{i\_max}$ } 라 하자. 여기서 i_max는 S_i에 저장된 SP의 최대수이다. 로봇의 초기위치를 (x,y,θ)=(0,0,0)라하며 Δx, Δy를 각각 X축과 Y축 스캔 폭이라 하면,

Step 1 : x, y 그리고 i를 각각 0로 한다. 그리고 로봇을 이동시켜 작업영역 내의 좌측상단에 접촉되게 하며, 그 위치를 초기위치로 정하고 그 x, y 위치를 S_i에 기록하며, 상태를 LEFT_WALL로 표기하며, tracked_flag을 UNTRACKED라 표기한다. $i \leftarrow i+1$.

Step 2 : 로봇을 y축의 위치를 고정시킨 상태에서 x 축을 따라 이동 시킨다. 만일 로봇이 작업영역 내의 장애물과 만나면 그 위치를 S_i에 기록하고 접촉상태는 RIGHT_OBSTACLE 이라 기록하며 i를 증가시킨다. 그리고 로봇을 계속 x축선상에서 이동 시켜 장애물벽과 접촉하면 접촉상태를 RIGHT_WALL 이라 기록한다. tracked_flag은 UNTRACKED 로 기록한다. $i \leftarrow i+1$.

Step 3 : 만일 로봇이 작업영역의 Y축 하단인 맨 밑바닥과 만나면 Step 5 로 이동한다. 그렇지 않으면 로봇을 Y축을 따라 Δy만큼 이동한다. 그리고 Y축을 고정시킨 상태에서 로봇을 X축을 따라 왼쪽으로 이동시킨다. 만일 로봇이 작업 영역내의 장애물과 접촉하면 그 위치를 S_i에 기록하고 접촉 상태를 LEFT_OBSTACLE이라 기록하며, 그때의 장애물 넘버를 기록한다. 그리고 로봇을 계속 오른쪽 작업영역 벽까지 이동시키서 로봇이 벽과 접촉할 때는 접촉위치를 기록하고 접촉 상태를 LEFT_WALL로 기록한다. 이 때 Tracked_flag은 UNTRACKED라 기록한다. $i \leftarrow i+1$.

Step 4 : 로봇을 Y축을 따라 Δy만큼 이동하고 Step 1으로 이동한다.

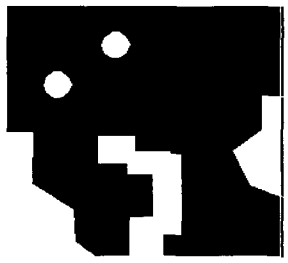


그림 3. 5개의 장애물을 갖은 작업공간 예.

Fig. 3. Exemplary workspace with 5 obstacles.

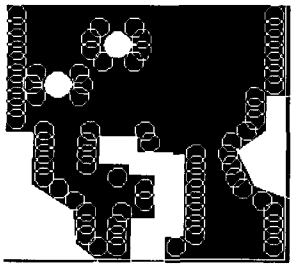


그림 4. 제안된 알고리즘에 의해 생성된 중간 목표점.
Fig. 4. Generated Subgoal points by proposed algorithm.

Step 5 : i를 스위핑 작업 경로를 위한 중간 목표점의 최대수인 i-max로 기록한다.

그림 3은 5개의 장애물이 포함된 작업영역 예이며, 그림 4는 제안된 중간 목표점 알고리즘을 적용시켜 SP를 찾은 결과를 보여주고 있다

2. 스위핑 경로 생성 알고리즘

중간 목표점 스캔 알고리즘은 그림 4에서와 같이 스위핑 작업 경로를 위한 중간 목표점을 생성한다. 여기서는 전술한 중간 목표점 생성 알고리즘에서 생성된 중간 목표점을 1) 모든 중간목표점은 2번이상 연결되지않게 하며, 2) 중간 목표점을 연결하여 형성된 스위핑 작업 경로는 가능하면 짧게 한다는 조건하에 전체 경로를 생성한다. 이 알고리즘은 현재의 중간 목표점의 접촉 상태에 따라 연결될 다음 중간 목표점을 찾는다. 스위핑 작업 경로를 찾는 알고리즘은 중간 목표점의 상태 LEFT_WALL이나 LEFT_OBSTACLE인 경우(CASE I)와, RIGHT_WALL 이나 RIGHT_OBSTACLE인 경우 (CASE II) 등 두 경우에 따라 스위핑 작업 경로τ를 생성하는 방법이 결정된다. 중간 목표점을 생성하기 위하여 로봇을 작업영역의 왼쪽 상단에서 오른쪽 하단까지 스캔하기 때문에 리스트 S_i 의 초기 엔트리에 저장된 로봇의 위치 데이터는 작업영역의 좌측 상단을 기르키며, 그 때의 접촉 상태는 LEFT_WALL이된다. 스위핑 작업경로를 생성하기위한 알고리즘은 다음과 같다.

y_max를 작업공간 W_{sweep} 의 Y축 최대 영역, x_max를 X축최대 영역이라하고, τ 를 스위핑 작업 경로를 위한 리스트라하면,

CASE 1 : 현재의 중간 목표점의 접촉 상태가 LEFT_WALL 이거나 LEFT_OBSTACLE 인 경우

Step 1 : W_{sweep} 공간상에서 리스트 S에서 같은 y값이 같은 SP중에서 접촉상태가 RIGHT_WALL 이거나 RIGHT_OBSTACLE 인 중간 목표점을 찾는다.

이 경우는 그림 2에서 (A)의 경우이다. 즉 현재 로봇의 접촉 상태가 LEFT_WALL 이므로 리스트 S에서 SP를 탐색하되 같은 Y축 선상에서 접촉 상태가 RIGHT_WALL인 SP를 탐색하여 스위핑 경로 선분(segment)을 형성한다. (B)의 경우처럼 또한 만일 현재 SP의 접촉 상태가 LEFT_WALL이면 리스트 S에서 같은 y위치 값을 갖는 SP중에서 접촉 상태가 RIGHT_WALL 이거나 RIGHT_OBSRACLE인 SP를 찾는다.

Step 2 : Step 1에서 찾은 SP 중에서 최단거리의 SP를 찾아서 리스트 τ 에 삽입한다. 만일 찾은 SP가 TRACKED 라고 표기됐고 현재 SP에서 연결될 다음 SP까지 로봇을 이동했을 경우 장애물과 충돌이 일어나면 Step 6으로 가고 그렇지 않으면 Step 3으로 간다.

그림 2에서 (B)의 경우가 현재 SP의 접촉상태가 LEFT_WALL이고 다음 접촉상태의 SP가 RIGHT_OBSTACLE이거나 RIGHT_WALL인 2개의 SP를 갖은 경우이다. 이 경우 후보 SP중에서 거리가 짧고 생성된 경로 선분이 장애물과 충돌이 일어나지 않는 SP를 연결될 SP로 선정하여 리스트 τ 에 삽입한다.

Step 3 : $y \leftarrow y + \Delta y$, 만일 $y < 0$ 이면, Step 5로 간다.

Step 4 : 같은 Y축 선상에서 접촉 상태가 RIGHT_WALL이거나 RIGHT_OBSTACLE인 SP를 찾아 경로 선분(pasth segment)이 무 충돌인 경우 그 SP를 리스트 τ 에 삽입한다.

그림 2에서 (C)의 경우이다. 여기서 SP의 현재 접촉 상태는 RIGHT_WALL이며 y값을 스캐닝 폭 Δy만큼 증가시킨 후 같은 접촉상태를 찾는다.

Step 5 : S 데이터 리스트 중에서 UNTRACKED 표시된 SP를 찾되 현재 SP 와 가장 거리가 가까운 SP를 다음 연결될 SP로 결정한다. 만일 현재 SP부터 다음 연결될 SP까지의 경로가 장애물과 충돌을 한다면 충돌회피 알고리즘을 부른다.

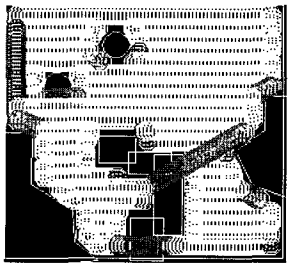


그림 5. 스위핑 경로 생성 알고리즘에 의해 생성된 스위핑 경로 (충돌회피 알고리즘 적용 전).
Fig. 5. A sweeping path by proposed sweeping path generation algorithm. (before implyingthe collision avoidance algorithm).

그림 2의 (D)의 경로이다. 이 경우 충돌회피 알고리즘을 부른다. 그림 5는 그림 3의 예제에서 충돌회피 알고리즘을 적용시키지 않고 스위핑 작업 경로를 찾은 경우를 보여주고 있다. 그림 5에서는 5개의 충돌 경로 선분이 보여지고 있다.

Step 7 : 만일 연결된 SP의 숫자가 i_max와 같다면 프로그램을 끝낸다. 그렇지 않으면 그 SP를 path 리스트 τ 에 삽입하고 해당 tracked_flag을 TRACKED로 표시한다. 링크된 SP의 상태에 따라 CASE I 이나 CASE II의 Step 1 으로 간다.

CASE 2 : 현재 연결된 SP의 접촉상태가 RIGHT_WALL 이거나 RIGHT_OBSTACLE일 경우 CASE II의 경우는 CASE I의 Step 1에서 Step 4까지 에서 RIGHT_WALL과 RIGHT_OBSTACLE을 각각 LEFT_WALL과 LEFT_OBSTACLE로 대체시키며 나머지 알고리즘은 CASE 1과 같다.

3. 충돌회피 경로계획 알고리즘

충돌회피 경로계획 알고리즘은 [9][10]에서 제안한 2단 경로 계획 방법을 사용하였다. 이 방법은 계층적 국부 탐색 알고리즘으로써 로봇의 위치인 x, y 와 로봇의 자세인 θ로 표시되는 3차원의 모양공간을 일정한 크기의 격자로 분할한 격자화된 모양공간(grid-based configuration space)에서 경로를 계획한다. 격자화된 셀(quantized cell) b는 장애물에 포함된경우 장애물 셀, 포함되지 않은 경우 자유 셀이라 정의 한다. 경로 계획 방법을 간략하게 표현하면 1) 기본 크기의 격자를 몇 개 합친 셀(격자화된 3차원 모양공간에서 최소 셀의 3×3×3 또는 2×2×2 크기의 셀)을 하나의 큰 셀 B로 보아 이 큰 셀들을 연결하여 로봇의 시작 자세가 포함된 큰셀과 목표자세가 포함된 큰셀 들을 사이의 연결통로를 만들되 큰 셀의 선택은 선택하고자 하는 큰 셀과 목표 셀과의 거리, 시작 자세의 셀과의 거리 그리고 큰 셀내의 장애물 셀의 갯수등으로 표현되는 평가함수를 최소로 하는 큰 셀들을 선택하여 연결한다. 2) 큰 셀의 연결중에 뒤엉킴이 현상이 있으면 제안된 재추적 방법으로 뒤엉킴이 없는 연결 상태를 만든다. 3) 이러한 큰 셀의 연결로 이루어진 연결 통로 내에서 기본 단위의 셀들의 연결로 이루게 되는 최종 경로를 만든다. 만일 최종 경로가 만들어지지 않는 경우 큰 셀들을 가상 장애물 셀로 기억하여 1)과정에서부터 큰 셀의 연결에서 제외 시킨다. 사용한 충돌회피를 위한 알고리즘의 전체 흐름도는 그림 7과 같다. 뒤엉킴 셀이란 큰셀로 통로를 구성함에 있어서 큰셀 연결 가지가 그림 6과 같이 여러 갈래로 나뉘어 져있는 상태를 말한다. 즉 셀을 연결하는 리스트에서 헤드(head)와 테일(tail)에 하나씩 만의 셀이 연결된 것이 아니라 두개 이상 연결된 상태를 말한다. 이러한 데이터 구조는 뒤엉킴이 시작된 셀을 중간 목표점으로 하여 역추적(backtracking)하면 뒤엉킴이 없는 셀들의 집합으로 재구성된다[8].

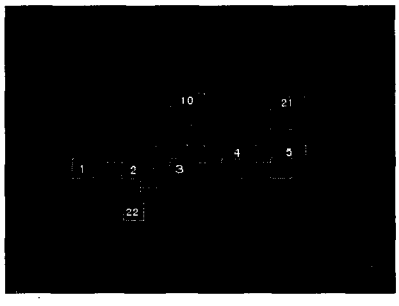


그림 6. 뒤엉킴 셀의 연결(2번, 3번 그리고 4번의 셀이 뒤엉킴 셀임).

Fig. 6. Connection of the Lumped cells (There are 3 lumped cells(number 2, 3, and 4)).

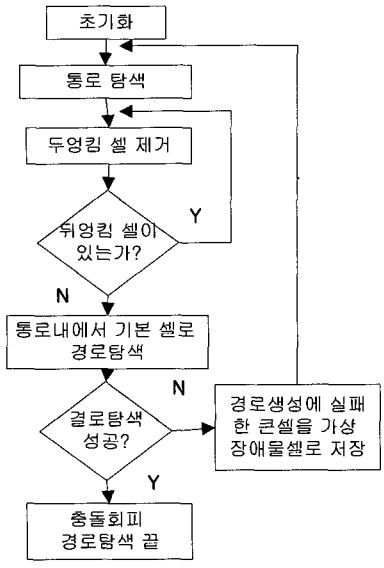


그림 7. 충돌회피 알고리즘의 간략화된 플로우차트.

Fig 7. Symplified flowchart of algorithm for collision free path

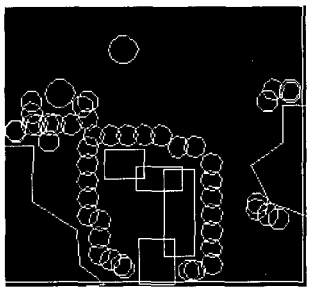


그림 8. 충돌 회피 경로.

fig. 8. A collision avoidance path.

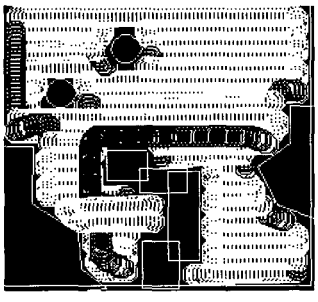


그림 9. 충돌회피 알고리즘이 적용된 최종 경로.

Fig. 9. A final resultant path with collision avoidance algorithm.

그림 3에 대한 SP의 생성 결과는 그림 5에 보인 바와 같다. 이 경우 5군데에서 경로상 충돌이 보이고 있다. 이에 대해서는 CASE I과 CASE II의 Step 5에서 충돌회피 경로 알고리즘을 수행하며 그 결과는 그림 8과 같고 충돌회피를 고려한 최종 스위핑 작업 경로의

결과는 그림 9와 같다. 그림 9에서 백색으로 표현된 이동 경로는 미장(또는 청소)작업 경로 이며, 청색(흑백으로 프린트된 경우 짙은 회색)으로 표현된 경로는 로봇이 미장작업(또는 청소작업)을 수행한 영역의 중복영역을 이동하는 경로를 표현한다.

IV. 계산시간의 평가

오프 라인 경로 계획 방법을 사용한 D.Kurabayashi 등의 방법[4]에서 보는 바와 같이 스위핑 작업 경로 생성을 위해 1) 보로노이(voronoi) 선도를 이용하여 경로 그래프를 작성하고, 2) 이 그래프 상에서 중국 우편배달부 문제(Chinese postman's problem) 알고리즘[4]을 이용하여 무 충돌 경로를 생성하는 것이 일반적이다. 이러한 방법에 대한 계산시간은 보로노이 선도를 작성하는 계산시간이 $O(N_v \log N_v)$ 이며[4][7][8], 중국 우편배달부 문제 알고리즘은 $O(N_{node}^3)$ 이다[4][6]. 여기서 N_v 는 작업 영역내의 장애물의 꼭지점의 갯수이며, N_{node} 는 스위핑 작업을 위해 생성된 그래프의 절점(node)의 수이다. 본 본문에서의 방법은 3장에서 기술한 바와 같이 보로노이 선도나 중국 우편배달부 문제와 같은 알고리즘을 이용하는 것이 아니라, 작업 영역내에서 1) 작업 영역을 좌-상단에서 우-하단까지 일정 간격으로 스캐닝하는 스위핑 경로 중간 목표점 생성, 2) 중간 목표점 SP들의 연결에 의한 스위핑 작업 경로 생성, 그리고 3) SP의 연결로 생성된 스위핑 작업 경로 선분(path segment)이 장애물과 충돌이 있을 경우 충돌 경로 회피 생성 등으로 구성 되어있다. 따라서 계산 시간의 분석은 1) 중간 목표점 생성을 위한 시간, 2) 중간 목표점의 연결에 의한 경로 생성 시간 및 충돌회피 경로 계획 시간등 2부분으로 나누어 분석된다.

1) 중간 목표점 생성을 위한 계산 시간 분석

스위핑 경로를 생성하기 위한 중간 목표점은 2차원 평면 작업 공간 W_{sweep} 의 좌측 상단에서 우측 하단 까지 일정 간격인 스캔폭 d로 스캔하며, 스캔하는 동안 작업영역 혹은 장애물과 이동로봇의 접촉위치에서 스위핑 경로를 위한 중간 목표점 SP(subgoal point)를 생성한다. 따라서 생성되는 SP의 갯수는

$$N_{SP} = \frac{W_l}{d} + \sum_{j=1}^{N_o} l_j / d \qquad (4)$$

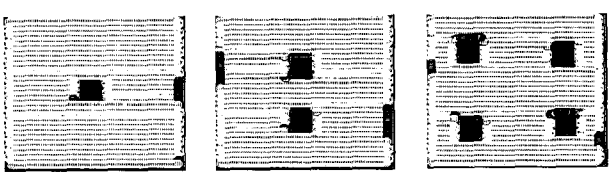
와 같다. 여기서 w_l 은 작업 영역의 둘레이고 l_j 는 j 번째 장애물 둘레, N_o 는 장애물의 갯수, d는 스캐닝 폭이다. 따라서 이 알고리즘의 계산시간은 $O(N_{SP})$ 이다. 즉, 장애물과 작업영역의 둘레의 길이에 비례한다.

2) 중간 목표점 연결에 의한 스위핑 경로 생성 알고리즘 및 충돌회피 알고리즘 계산 시간 분석 경로 생성 알고리즘은 3.2절에 표현된 바와 같이 W_{sweep} 에서 생성된 SP중에서 같은 X축 선상에 있는 SP 들을 대상으로 다음 연결될 SP를 탐색하여, 연결될 SP가 없을 경우 현재 접촉된 장애물 또는 작업 영역의 접촉 선상의 위 스캔선 상과 아래 스캔선 상에 있는 SP를 탐색한다. 따라서 최악의 경우 탐색할 장애물 개수는 탐색할 SP가 같은 X축 선상에 있는 경우이며, 이 경우는 같은 X축선상에 모든 장애물이 걸려있는 경우 이므로 $N_{SP} \cdot (2+N_O)$ 이다. 따라서 계산 시간은 $O(N_{SP} \cdot N_O)$ 가 된다. 하지만 이 경우는 SP의 연결로 스위핑 작업 경로를 만들 경우 경로가 장애물을 회피하는 계산시간을 고려하지 않은 경우이다. 장애물을 회피하는 경로를 만들 경우 충돌회피 알고리즘은 3.3절과 [10]에서 제안한 알고리즘을 사용한다. 이 알고리즘의 계산시간은 통로 탐색의 계산시간은 W_{sweep} 에 대한 모양 공간 내에서 큰 셀B를 시작점에서 목표점까지 탐색하여 트리 구조의 그래프를 만들므로 $O(N_B \log N_B)$ 이다. 여기서 $N_B = \frac{N_{SP}}{M}$ 이며 M은 큰 셀 B내의 기본 셀의 개수이다. 그리고 뒤엉킴 제거를 위한 재탐색은 탐색된 큰셀로 이루어진 통로 내에서 뒤엉킴이 시작되는 셀을 중간 목표점으로 하여 역탐색(backtracking) 하므로 최악의 경우 $O(N_B \log N_B)$ 이 된다. 그리고 최종 경로는 재탐색 되어 뒤엉킴 큰 셀이 없는 통로 내에서 경로가 탐색 되므로 탐색 되는 셀의 개수는 통로내의 큰 셀의 개수를 N_T 라고 할 경우 $M \cdot N_T$ 가 곱해지게 되고 $M \cdot N_T$ 는 상수이므로 충돌 회피를 위한 계산 시간은 $O(N_B \log N_B)$ 에 비례하게 된다. 따라서 전체적으로 스위핑 경로 생성과 충돌회피 알고리즘은 $O(N_{SP} \log N_{SP})$ 에 비례한다.

V. 모의 실험 결과

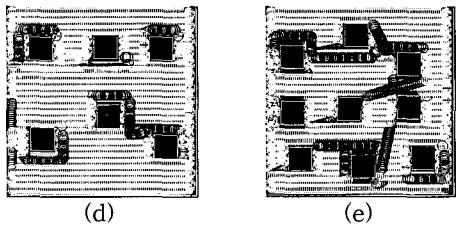
본 장에서는 제안된 방법을 모의 실험에 적용한다. 작업영역 W_{sweep} 의 크기는 최대 21m × 21m이며, 로봇은 크기 한 변이 41.42cm인 정팔각형 형태로 모델링하였다. 충돌회피 경로 탐색을 위한 모양공간은 작업영역을 10cm 단위로 나누어 격자화 시켰으며 모양공간 내의 기본 셀의 갯수는 210×210 = 44,100개로 이루어져 있다. 스캔 폭은 75cm로 하였다. 사용 컴퓨터는 IBM compatible PC이며 Pentium급 586 MMX 200Mhz, RAM 64Mbyte이다. 실험은 다음과 같이 3가지로 나누어 실행되었다.

실험 1 : 작업 공간내에서 240cm×240cm의 정사각형 형태의 장애물을 1개, 2개, 4개, 6개 그리고 9개를



(a) (b) (c)

(a) 장애물이 있은 경우
(b) 2개 장애물이 있는 경우
(c) 4개의 장애물이 있는 경우
(d) 1 obstacle included case
(e) 2 obstacles included case
(f) 4 obstacles included case



(d) (e)

(a) 6개 장애물이 있는 경우
(b) 9개 장애물이 있는 경우
(c) 6 obstacles included case
(d) 9 obstacles included case

그림 10. 최종 스위핑 작업 경로.
Fig 10. A final resultant sweeping path.

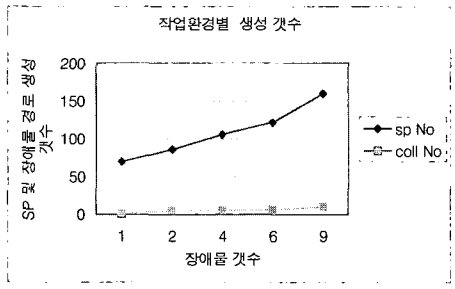


그림 11. SP와 충돌 경로 개수에 관한 표.
Fig. 11. A chart on the number of SP and collision path.

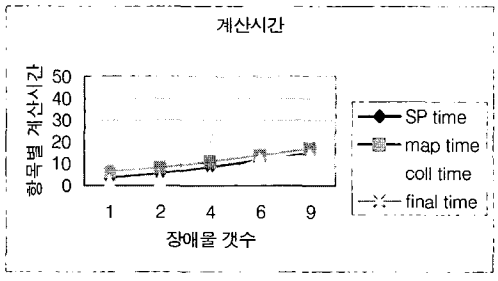


그림 12. 계산 시간에 관한 표.
Fig. 12. A chart on the computational time.

표 1. 작업환경내의 장애물의 갯수에 따른 계산 시간 및 SP생성 개수.
Table 1. The computational time and number of SP with respect to the number of obstacles in work space.

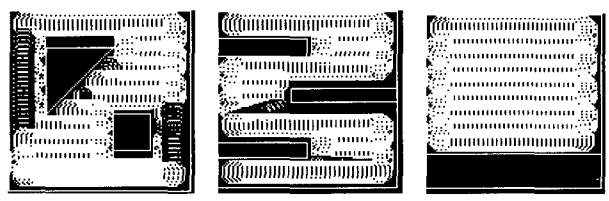
시간단위 : sec

항목\ 장애물갯수	SP 생성 개수	충돌경로 개수	SP생성 시간	모양공간 생성시간	충돌회피경로생성시간	최종경로 생성시간
1	70	1	3.516	6.593	1.593	11.703
2	86	5	5.384	8.131	2.362	15.879
4	106	5	8.571	11.099	4.560	24.231
6	122	6	11.538	13.736	11.923	37.198
9	160	10	14.725	16.978	13.846	45.549

임의로 설치하여 SP생성 시간, 생성 개수, 그리고 모양공간의 생성 시간과 충돌회피 경로생성 계산시간을 분석하였다. 경로의 최종결과는 그림 10(a)에서 그림 10(e) 까지에 나타내었으며, 계산 시간 및 분석표는 표 1과 그림 11, 그림 12에 나타내었다. 표1과 그림 11, 그림 12에 나타난 바와 같이 계산 시간과 SP의 생성 개수는 장애물이 일정한 규격을 갖을 경우 $O(N_B \log N_B)$에 비례함을 알 수 있다.

표 1과 그림 11, 그림 12에서 보는 바와 같이 충돌 경로의 수는 장애물의 수에 비례하지는 않는다. 그 이유는 충돌 경로는 장애물의 위치에 따라 결정되기 때문이다. 하지만 작업공간 내에 장애물의 갯수가 많을 경우 충돌 경로가 많이 발생함을 보이고 있다. 그림 11과 그림12는 표1에 대한 SP생성 개수 및 충돌 경로 개수, 그리고 계산시간을 도표로 나타낸 것이다.

실험 2 : D.Kurabayashi 방법[4]을 적용한 작업환경에 본 논문에서 제안한 방법을 적용하여 스위핑 경로의 생성 과 최종 경로 길이를 비교하였다. 작업 영역의 크기는 4m×4m이고 작업 로봇의 크기는 반경 30cm이다. 그림 13(a), 그림 13(b), 그리고 그림 13(c)는 [4]에서 적용한 동일한 예제에 대하여 본 논문에서 제안한 방법을 적용하여 스위핑 경로를 생성 시킨 것이다. 그리고 표 2는 동일한 예제에 대해 생성된 경로 길이의 비교이다. [4] 에서의 경로 계획 방법은 소용돌이 모양 같이 회전하면서 작업영역의 중심으로 향해 움직이는 컨투어 페러렐(contour parallel)형태이며 본 논문에서 제안한 방법은 그림 13(a)부터 그림 13(c)까지에서 알 수 있듯이 지그재그(zigzag) 모양으로 움직이는 디렉션 페러렐(direction parallel) 형태이다. [4]에서 제안한 방법은 장애물이 없는 작업 영역인 그림 13(c)에서 본 본문의 방법인 디렉션 페러렐 형태로 생성된 경로보다 약 1.7%가량 짧았으며, 그림 13(a)의 경우에서와 같이 장애물이 작업 영역 중앙에 분포되어 있는 경우에도 약 9.9%가량 작업 영역이 짧았다. 하지만 그림 13(b)와 같은 작업 영역의 경로 생성 결과에서 볼 수 있듯이 장애물이 작업영역의 벽면에 부축된 형태로 분포된 경우에는 본 논문 방법과 같은 디렉션 페러렐 형태의 경로가 [4]에서의 방법보다 약 11%가량 경로 길이가 짧았다. 따라서 장애물의 배치에 따라서



(a) (b) (c)

그림 13. [4]에서 제시한 작업 예제에 대한 본 방법의 경로 생성 결과.
Fig. 13. Sweeping paths generated by proposed method for map examples in [4].

표 2. 본 방법과 [4]에서 제시한 방법의 스위핑 경로 비교.

Table 2. The comparison between the proposed method and the method in[4].

예제 방법	그림 13(a)	그림 3(b)	그림 13(c)
D.Kurabayashi 방법	3169 cm	3231 cm	2583 cm
본 본문 방법	3484.09 cm	2900 cm	2627.17 cm
비 교	[4]의 방법이 9.9%짧음	본 방법이 11% 짧음	[4]의 방법이 1.7% 짧음

생성된 경로의 길이가 각각 다르므로 경로 생성 방법에 대한 우열의 차이는 나지않는다고 사료된다.

실험 3 : 실험 3에서는 제안된 경로계획 알고리즘을 실제 건설도면에 적용하였다. 건설도면은 AUTO CAD R 14 로 제작된것 이며 (모 대학의 복지관 건설도면 CAD 파일) CAD SW와의 인터페이스 메뉴는 AUTO LISP으로 제작되었다. 이렇게 함으로써 건설 도면의 작성은AUTO CAD SW를 이용하며, AUTO LISP로 제작된 인터페이스에서는 로봇의 형태, 작업 영역의 크기등을 결정하게 되며 작업 영역내에 선택된 물체를 선별한 후 C언어로 프로그램된 SP 생성 알고리즘과 충돌회피 알고리즘을 불러들인다. 그림 14와 그림 15, 그림 16에서 결과를 보이며 실제 도면에 적용 시켰을 경우에도 제안된 알고리즘이 성공리에 스위핑 작업 경로를 생성함을 보이고 있다.

그림 14. 건설 CAD 도면 예.

Fig. 14. Exemplary architectural CAD draft.

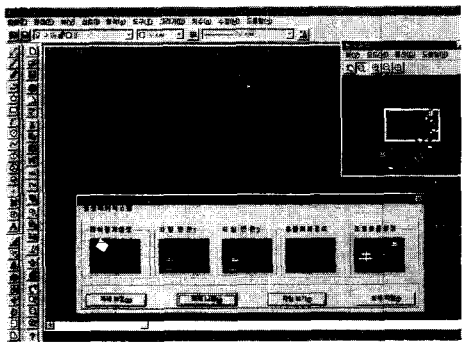


그림 15. 제작된 AUTO CAD 인터페이스 메뉴.

Fig. 15. AUTO CAD interface menu.

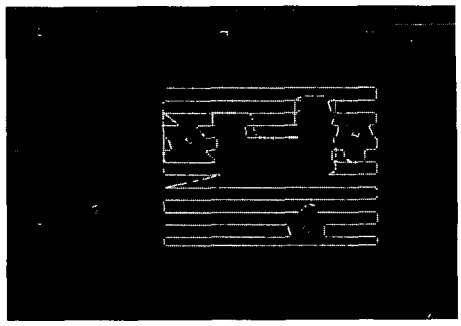


그림 16. 건설 CAD 도면 상의 미장 로봇을 위한 최종 스위핑 작업 경로.

Fig. 16. A resultant sweeping path for smearing robot on architectural CAD draft.

VI. 결론

미장작업 및 청소작업을 위한 스위핑 작업 경로를 오프라인으로 탐색하는 계획방법이 제안되었다. 제안된 방법은 1) 건설 CAD S/W 와의 인터페이스 및 도면상에서의 장애물 인식, 2) 오프라인 장애물 지도 구성, 3) 스위핑 작업 을 중간 목표점 생성 알고리즘, 4) 중간 목표점의 연속적 추적 알고리즘, 그리고 5) 장애물 회피 알고리즘.등으로 구성되어있다. 제안된 방법의 성능을 분석하기 위하여 작업 환경 내에서 장애물 수를 증가시킨 후 계산 시간 분석을 한 예와 실제 건설 도면인 AUTO CAD 도면 파일을 대상으로 실험하였고, 그 경우 계산시간은 $O(N_B \log N_B)$에 비례함을 실험을 통해 확인 하였으며, 실제 도면상에서도 우수하게 동작됨을 보였다. 하지만 실제 미장 작업이나 청소작업을 할 경우 작업을 할 실제 환경이 건설도면 등과 다른 경우가 발생하게 되는데 이 경우 실제 환경을 고려하여 작업경로를 수정할 수 있는 방법에 대한 연구가 본 연구의 향후 과제이다.

참고문헌

[1] Shermen. Y. T. Lang and Bing-Yang Chee, "Coordinate behaviors for mobile robot floor robot cleaning," *proc. of IEEE/RSJ. Intl. Conf. on Intelligent Robots and Systems*, vol. 2, pp. 1236-1241, 1998.

[2] C. Hofner and G. Schmidt, "Path planning and guidance techniques for an autonomous mobile cleaning Robot," *proc. of the IEEE/RSJ Int.conf. on Intelligent robot and Systems*, pp. 610-617, 1994.

[3] K. S. Chong and L. Kleeman, "Sonar based map building for mobile root," *proc. of IEEE robotics and automation*, 1997.

[4] D. kurabayashi and J. Ota, " Motion planning of multiple robots for cooperative sweeping," *Journal of robotics society of Japan*, vol. 16, no. 2, pp. 181-188, 1998.

[5] D. kurabayashi, T. Ari, and K. Iwase, "Real-time adaptation for sweeping by autonomous mobile robots," *Journal of robotics society of Japan,* vol. 17, no. 5, pp. 677-684, 1999.

[6] J. C. Latombe, "Robot motion planning," Kluwer Academic publishers, 1993.

[7] J. Edmonds and E. L. Johson, "Matching eluer tours and the chinese postman," *Mathematical programming*, no. 5, pp. 88-124, 1973.

[8] M. kwan, "Graphic programming using Odd and even points," *Chinese Math*, 1, pp. 273-277, 1962.

[9] W. K. Hyun, D. S. Shin, and I. H. Suh, "A path planning algorithm for autonomous smearing robot," *JSME pioneer Intl. Symposium on MOVIC,* vol. 1, pp. 63-67, 1999.

[10] W. K. Hyun and I. H. Suh, "A heuristic collision-free path planning algorithm for robots," *IEEE IROS*, vol. 2, pp. 837-845, 1995.

[11] 노영식, 김호중, "회전 트로웰의 원판형 가정을 통한 콘크리트 미장 로봇의 전방향 운동 모델링," 제어·자동화·시스템 공학회지, 제5권, 제4호, pp. 454-462, 1999.

[12] 차영성, 권대갑, "METRO-레이저를 장착한 자율이동 로봇," 제어·자동화·시스템 공학회지, 제2권, 제3호, pp. 200-208, 1996

[13] 정학영, 박솔잎, 이장규," 초음파 센서를 이용한 이동로보트의 네트워크 환경 모델 구성," 제어·자동화·시스템 공학회지, 제5권, 제4호, pp. 593-599, 1999.

[14] 주진화, 이장명, "신경망을 이용한 이동로봇의 실시간 고속정밀 제어," 제어·자동화·시스템 공학회지, 제5권, 제1호, pp. 95-104, 1999.

현 웅 근

1993년 한양대학교 전자공학과 박사수료,공학박사. 1993년 ~ 현재 호남대학교 전자공학과 재직중, 조교수, 1997 ~ 1998년 일본 오사카 전기통신대학 객원 연구원, 1996 ~ 1999 한국기계연구소 자동화 구룹 위촉연구원, 관심분야는 인공지능제어, 로보틱스., 근거리 원격 통신 제어.

박 상 규

여수대학교 자동차공학부/기계공학전공, 교수. 1989년 인하대 기계공학과 박사수료,공학박사. 여수 RRC 센터소장, 관심분야 유체공학, 시스템 자동화 분야.