두 팔 로봇의 조립 작업을 위한 지능적 조립 계획 시스템

김형욱†, 황병훈†, 박진서†*, 서일홍†, 이병주††

† 한양대학교 전자공학과.
†† 한양대학교 제어계측공학과.

요 약

본 논문에서는 조립부품 데이타 베이스(ODB)생성기와 완성품 데이타 베이스(TDB)생성기, Task-Level 명령어 번역기로 구성된 Task-Level 조립 계획 시스템을 제안한다. ODB 생성기는 길이, 폭, 높이, 잡는 방향 등을 포함하여, 조립 부품들의 3차원 형상정보를 만드는 역할을 한다. TDB 생성기는 ODB 의 부품을 사용하여 장애물을 피하고, 최적의 경로를 찾는 역할을 한다. 여기에서 사용자는 그래픽 환경에서 완성품의 조립 순서를 바탕으로 한 Task-Level 명령어를 지정하게 된다.

또한, TCI 는 로봇이 TDB 에 의해 계산된 최적 경로를 따르면서 두 로봇간의 충돌방지를 고려한 로봇 제어 명령들을 생성한다. 이렇게 생성된 기본적인 명령어들은 조립 작업을 수행할 두 팔 로봇 제어 시스템으로 전달된다. 제안한 시스템의 효용성을 보이기 위해, 13 개의 레고블럭을 이용하여 조립작업을 수행하였다.

1. 서론

산업 현장에서 생산공정의 자동화 기술은 일반적으로 로봇을 중심으로 개발되어 왔다. 특히, 최근들어 인건비의 상승과 조립 공정의 복잡성으로 인해 조립 자동화의 필요성이 강하게 대두되고 있다. 지금까지, 대부분의 로봇 언어는 로봇 작업을 중심으로 개발되었지만 이런 언어는 전문 시스템 개발자가 아니면 사용하기에 어려운 점이 많았다. 그러므로, 조립작업을 쉽게 프로그래밍하고 이를 구현할수 있는 Task-Level 언어의 개발은 일선 작업자들에게 로봇을 이용한 조립 작업의 편의성을 증대시킬 수 있을 것이다.[1]

Task-Level 언어는 로봇의 기구학에 대한 전반적인 지식이 없어도, 사용자가 조립될 부품의 시작과 종료 상태만 지정해 주면 되는 새로운 형태의 언어이다. AUTOPASS 를 시초로 하여 Task-Level 제어 언어에 대한 많은 연구들이 진행되어 왔지만, 일반적으로, 로봇 언어의 서로 다른 형식과 비호환성 때문에 범용의 Task-Level 제어 언어의 개발에는 아직도 많은 어려움이 남아 있다[3]. 본 논문에서는 조립 작업에 필요한 여러 Task-Level 명령어들은 정의하고, 그래픽 화면에서 조립 작업을 수행하면 Task-Level 명령어들을 SCARA 형 두 팔 로봇의 제어에 필요한 로봇 명령어들로 전환해주는 Task-Level 조립 계획 시스템을 제안한다.

본 논문은 다음과 같이 구성된다. : 2 절에서는 Task-Level 명령어를 정의하고, 조립 계획 시스템의 전반적인 내용을 주로 설명할 것이다. 3 절에서는 ODB, TDB, TCI 에 관련된 알고리즘을 설명하고 4 절에서는 두 팔 로봇의 충돌방지에 대해 설명한다. 실험 결과와 추후 과제는 5 절과 6 절에서 설명한다.

2. Task-Level 조립 계획 시스템의 설계

2.1 Task-Level 제어 언어의 정의

Task-Level 제어 언어는 부품들간의 결합 관계를 고려해야 한다. Ko 와 Lee[9]는 부품들간의 관계를 Against, Fit, Tight-fits, Contact 등의 네 가지 형태로 분리 하였고, Hommem de Mello 와 Arthur C. Sanderson[4], J.P Thoms, P.N Nissanke[7]도 부품들의 형상 정보와 부품간의 결합 정보로 구성되는 relation model graph 를 소개하였다.

본 논문에서는 기존의 연결 관계 표현 방법과 물체의 기본적인 이동 방법을 토대로 위치 정보를 이용한 *PICK*, *PLACE* 와 위치와 힘 정보를 이용한 *ATTACH*, *INSERT* 를 정의한다.[5] 이런 명령어들은 실제로 적용되는 로봇에 맞게 다음과 같이 재정의될 수 있는데 예를 들어, SCARA 로봇의 경우에는 다음과 같이 정의할 수 있다.

PICK robot# <part>

: robot#를 이용하여 <part>를 집어 end-effector 를 최고 높이까지 들어 올린다. SCARA 로봇의 경우, end-effector 를 최고 높이로 들어 올리면 물체를 피할 수 있다고 가정한다.

PLACE robot# <part> TO <location>

: z 축의 위치를 유지하면서 <location>까지 이동해서 수직으로 end-effector 를 내린다.

ATTACH robot# <part1> TO <part2>

: <part2>방향으로의 접촉힘을 일정한 범위로 유지하면서 목표점에 도달하면 정지하고 그 위치에 <part1>을 놓는다.

INSERT robot# <part1> INTO <part2>

: 삽입방향으로의 접촉힘을 일정한 범위로 유지하면서 목표지점에 도달하면 정지하고 <part1>을 놓는다.

위에서 정의한 PLACE, ATTACH, INSERT 는 조립 전에 로봇이 물체를 PICK 한 것을 전제로 한다. 기본적으로 로봇은 계산된 경로를 따라 물체에 PICK 된 부품을 조립하기 위해 이동하고 다른 부품을 PICK 하기 위해 같은 경로를 따라 돌아 온다. 일반적으로, 조립 작업은 PICK 과 다른 작업 명령어들중 하나와의 조합으로 구성된다.

2.2 전체 시스템의 구성

제안하고자 하는 Task-Level 조립 계획 시스템의 구조는 크게 두 부분으로 나누어 진다. 첫째로, Task-Level 명령어 번역기는 주어진 Task-Level 프로그램을 해석하여, 로봇 명령어들로 바꾸고 이를 로봇 제어 시스템으로 전달한다.

두번째 부분은 완성품 데이타베이스 생성기인데, 그래픽 환경에서 조립 순서를 사용자가 선택하면 특정 완성품에 대한 Task-Level 명령어는 자동으로 생성되고 이들 정보는 TDB 파일에 저장된다. 일반적으로 특정완성품에 대해서 조립 순서는 여러 개가 존재할 수 있다.

조립 계획 시스템의 구조는 그림 1 과 같고 각 부분의 역할은 다음과 같다.

- TDB(Target Data Base) : 완성품에 대하여 가능한 조립 순서별 Task-Level 제어 명령 및 이를 수행할 로봇 제어 명령들이 저장된다.
- TCI(Task-Level Command Interpreter) : Task-Level 명령어를 로봇 명령어로 바꾸어 준다.
- EDB(Element Data Base) : 각 부품들을 구성하는 기하학적인

모양들의 형상정보가 저장되어 있다.

- ODB(Object Data Base) : 조립 부품을 이루는 기본 부품들에 대한 3 차원 형상정보와 기본 부품들간의 연결 관계를 포함한다.
- Obstacle modeling : ODB 를 사용하여 작업 공간에서 표현된 장애물을 configuration space 로 전환해준다.
- Path planning : 시작점부터 종료점까지의 경로를 생성한다.

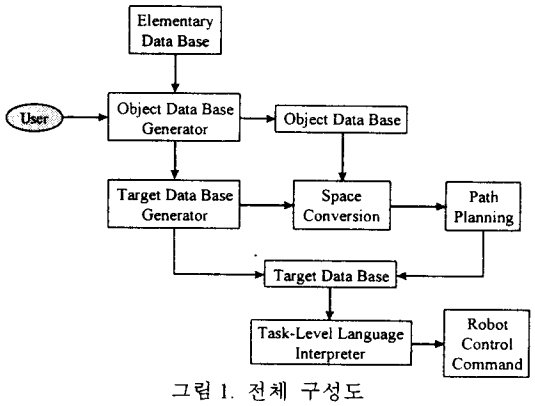

그림 1. 전체 구성도

3. 완성품 데이타 베이스의 구성

3.1 조립 부품의 모델링 및 데이타 베이스의 구성

기계적인 조립은 상호 결합된 부품들을 하나의 고정된 형태로 구성하는 것인데, 각 부품은 강체이기 때문에 그 형태는 변하지 않는다. 조립 계획을 위해서는 길이, 폭, 높이 또는 잡는 위치와 같은 부품들의 모델링을 필요로 한다.

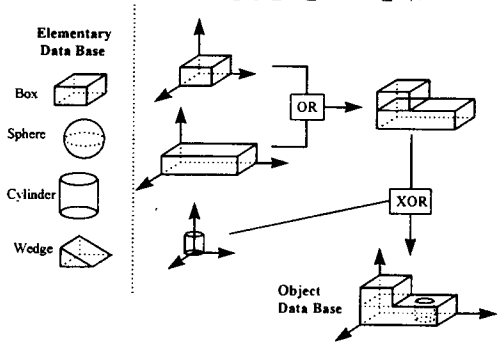

그림 2. 조립물체의 3 차원 형상정보 표현

본 논문에서는 그림 2 와 같이 부품들간의 기하학적 정보를 나타내기 위해 기본 부품들의 연결 정보와 파라미터를 EDB 에 저장 한다. 기본 부품들간의 연결에 대한 정보인 잡는위치, 로봇의 자세등은 ODB 에 저장되는데, ODB 의 구조는 다음과 같다.

```
struct data_base {
    char name[ ];              /* 물체의 이름 */
    char cur_loc[ ];           /* 물체의 현재위치 */
    double grasp_point[3];     /* 물체를 잡기 위한 위치 */
    double grasp_angle;        /* 4 축의 각도 */
      struct item {
          int current_object;  /* 기본부품의 종류 */
          double length;       /* 물체의 길이 */
          double width;        /* 물체의 폭 */
          double height;       /* 물체의 높이 */
          double radius;       /* 물체의 반지름 */
          double dx, dy, dz;   /* 좌표계 원점의 offset */
          int connect;         /* 연결방법 */
      } feature[ ];            /* 물체의 형상정보 */
} object[ ];
```

여기서 "cur_loc[]"는 부품들을 공급하는 feeder 의 현재 위치를 나타내고, "feature[]"는 기본 부품들간의 정보와 연결 관계(OR, XOR)를 나타낸다. "grasp_point"와 "grasp_angle" 은 end-effector 의 중심 위치와 로봇의 4 축의 각을 나타낸다.

3.2 완성품 데이타 베이스의 구축

조립 부품 데이타 베이스가 만들어지면 각 조립 부품의 형상 정보를 이용하여 완성품 데이타 베이스(TDB)를 생성한다. TDB 는 조립 순서, 조립 경로, 부품들간의 연결점, Task-Level 명령어 등을 포함하는 데이타 베이스이다. TDB 생성기를 사용하여 사용자는 그래픽 환경에서 조립 순서와 부품들간의 결합 위치를 지정하고, 조립을 위해 필요한 경로를 생성한다. 이런 정보는 다음의 데이타구조에 의해 TDB 에 저장된다.

```
struct target_data {
    char target_name[ ];       /* 완성품의 이름 */
    int connect_num;           /* 연결횟수 */
    char *sequence[ ];         /* 조립순서 */
      struct con {
          char task[ ];        /* Task 명령의 이름 */
          char grasped[ ];     /* 이동할 부품의 이름 */
          int command_num;     /* 로봇제어명령의 갯수 */
          char *command[ ];    /* 로봇제어명령 */
      } connect[ ];            /* 완성품 연결정보 */
} target[ ];
```

4 경로계획

4.1 장애물 공간의 생성

일반적으로, 완성품은 많은 조립 부품들을 조립하여 이루어 진다. 이때, 조립의 각 단계에서 이전에 조립된 부품들은 이동중인 로봇에게는 장애물이 되기 때문에 조립 작업을 성공적으로 수행하려면 이동중인 로봇과 조립된 부품간의 충돌이 방지되어야 한다. 그러므로, 로봇의 경로계획은 ODB 에 있는 3 차원 형상정보를 이용하여 구성된 configuration space 에서 형성되어야 한다. Configuration space 에서 안전한 경로는 어떤 장애물과의 충돌도 없는 연속적인 경로를 의미한다.[6]

SCARA 로봇의 z 축의 방향은 일정하게 유지되고, 1 축과 2 축의 관절각인 θ_1, θ_2에 의해 위치(x, y)가 결정됨을 이용하여 다음의 알고리즘과 같이 모든 관절각에 대해서 검색하지 않고도 장애물 공간을 구할 수 있다.

< 장애물 공간을 구하는 알고리즘 >

1. 두 조립부품이 결합될 때의 자세에 의해서 4 축각도 θ_4를 결정하여 고정한다.
2. 각 조립부품의 형상정보를 이용하여 각 조립부품을 포함하는 최소육면체를 구한다.
3. 2 에서 얻어진 최소육면체의 크기에 따라서 3 축각도 θ_3의 범위를 정한다.
4. X-Y 평면에 이 다면체를 사영한다.
5. 1 축과 2 축관절각을 미소씩 변화시켜 이동중인 물체의 사영된 다각형이 이전에 조립된 물체에 걸리는지를 조사한다. 사영된 도형에 걸리면, 3 에서 계산된 경계를 조사하여 충돌의 위치를 찾아낸다. 충돌점은 장애물 공간에 저장된다.

4.2 최적 경로 계획

로봇이 부품을 PICK 했을 때의 관절각을 초기 위치로 하고, 조립이 이루어졌을 때의 관절각을 목표 지점으로 한다. 두 위치를 연결하는 경로는 조립할 때에 다른 부품들과 충돌하지 않아야 하고, 조립은 가능한 빨리 이루어져 한다.

본 논문에서는 적은 계산량으로 최소 조립 시간과 최소 거리를 얻는 경로 계획 알고리즘을 제안한다.

1. 초기위치에서 목표지점을 잇는 직선이 장애물에 걸리는지를 조사한다.
2. 직선이 장애물에 걸리면 기본방향을 따라 장애물에 걸리지 않는 방향을 찾는다.
3. 충돌이 없는 방향중에서 목표지점까지의 거리가 최소인 쪽을 선택한다.
4. 현재위치에서 정해진 방향을 연결하는 직선상의 지점 중에 목표지점까지의 거리가 가장 짧은 지점을 다음 경유점으로 선택한다.
5. 다음 경유점으로 이동한다.
6. 목표점에 도달할 때까지 1 부터 5 단계까지의 과정을 반복한다.

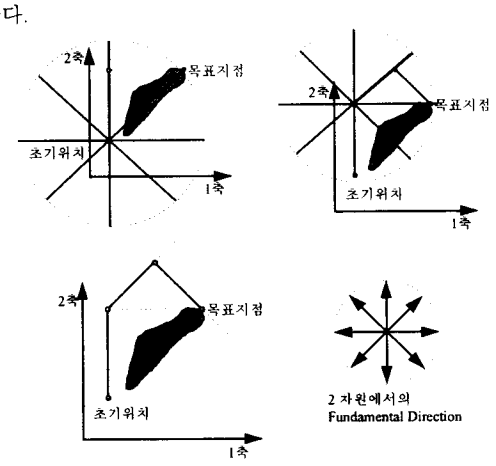

그림 3. 가감속 횟수의 최적화

위의 알고리즘에 의해서 두 개의 경유점을 지닌 경로를 그림 3 처럼 구할 수 있다. 그러나, 이 경로는 최소 경로가 아니기 때문에 거리를 최적화시키기 위해 목표점부터 초기점까지 반대로 경로를 조사한다. 우선, 목표점을 현재 위치로 하고, 이전 경로점을 a, a 이전의 경유점을 b, 목표점과 a 점을 연결하는 직선을 따라 움직이는 점을 c 라고 한다. 그러면, b 와 c 를 연결하는 직선이 장애물과 충돌하지 않을 때까지 미소 단위만큼 c 점으로 이동한다.

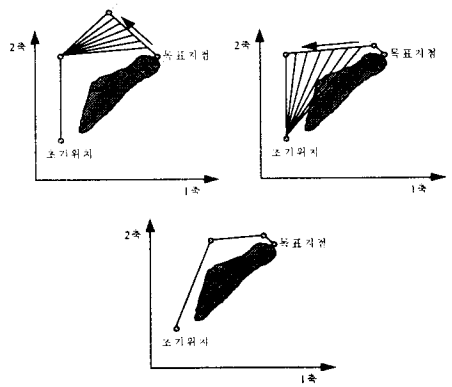

그림 4. 거리의 최적화

4.3 충돌 방지

여러 로봇이 동일한 작업공간에서 작업을 할 경우에는 로봇간에 서로 충돌하지 않도록 하는 것이 가장 중요하다. 일반적으로는 세마포어(또는 플래그)를 사용하여 하나의 프로세스가 끝난 후에 다른 프로세스를 실행함으로서 충돌을 방지하는 방법을 많이 사용하지만, 본 논문에서는 로봇 제어 시스템에서 제공하는 명령들을 이용하여 로봇간의 충돌을 방지하였다.

삼성 FARA SRC421 로봇 제어기는 여러 개의 레지스터를 제공하는데, 두 로봇 팔은 이들 레지스터를 통하여 상호간의 정보를 공유할 수 있다. 조립 작업이 시작되었을 때, 각 로봇 프로그램은 세개의 레지스터를 임의의 값으로 지정을 하고, 각 Task-Level 명령어가 수행될 때마다 레지스터를 검사한다.

Robot1	***Robot2***
WHILE(@R3 != 99)	WHILE(@R3 != 99)
DELAY 0.1	DELAY 0.1
ENDWHILE	ENDWHILE
READY	READY
@R1 = 0	@R2 = 0
(other robot commands)	*(other robot commands)*
WHILE(@R2 == 10)	WHILE(@R1 == 10)
DELAY 0.1	DELAY 0.1
ENDWHILE	ENDWHILE
@R1 = 10	@R2 = 10
(other robot commands)	*(other robot commands)*

위의 로봇 프로그램에서 "@"는 공유레지스터를 의미한다. Robot2 가 초기에 공유공간에서 부품을 조립한다면, 레지스터 R1 을 0 으로 지정하고 레지스터 R2 를 10 으로 지정한다. 이 때에 Robot1 은 Robot2 가 작업을 마칠 때까지 기다리게 된다. R2 가 0 이 되면 Robot2 는 멈추고, Robot1 이 작업을 시작한다.

5. 실험 및 결과

본 논문에서 제안한 자동 조립 계획 시스템은 IBM PC 에서 Lynx-OS 에서 기반으로 Motif 를 사용하여 구현하였다. 먼저 사용자가 화면에서 부품들을 조립하면, 조립 계획 시스템은 Task-Level 명령어를 생성하고 이를 로봇 제어 명령어로 바꾸어 준다. 이렇게 생성된 명령어는 조립을 위해 로봇 제어기로 전달된다. 제안된 계획 시스템의 효용성을 보이기 위해, 13 개의 레고 블럭을 이용하여 실제 조립작업을 수행하였다.

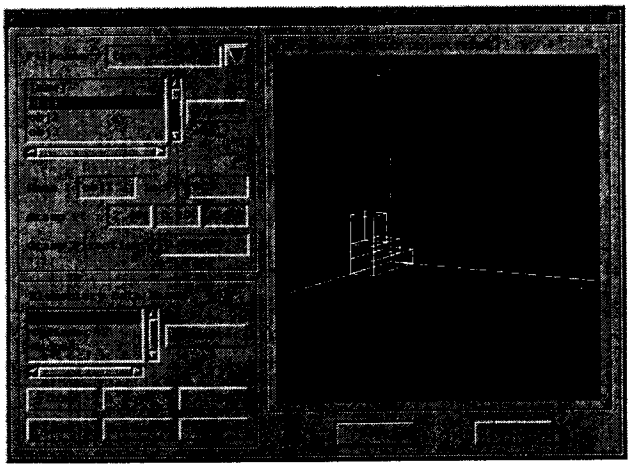

그림 5. Object Data Base Generator

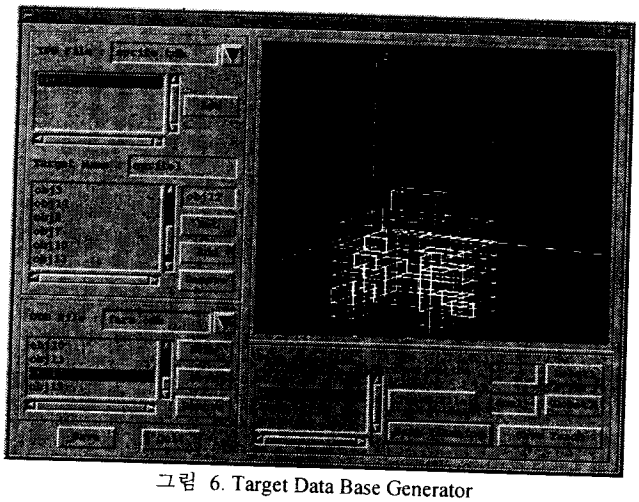

그림 6. Target Data Base Generator

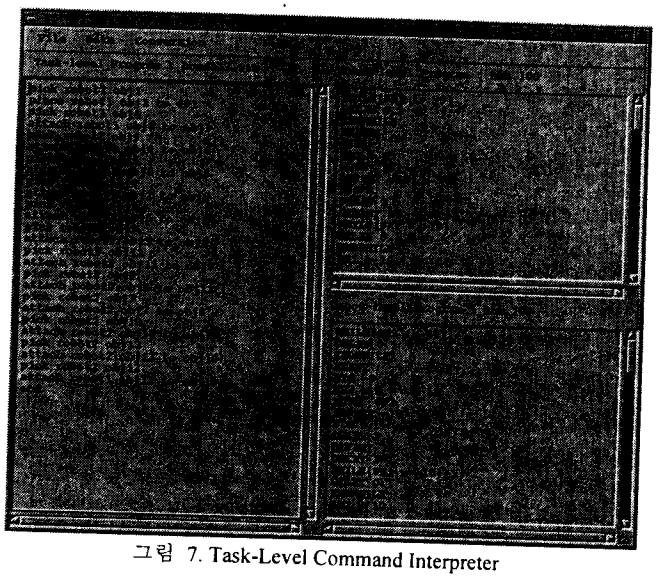

그림 7. Task-Level Command Interpreter

위의 세개의 그림은 조립 작업의 3 단계를 보여준다. ODB 생성기를 이용하여 조립부품 데이타 베이스를 생성하고(그림 5), 화면에서 조립을 하고, TDB 생성기에서 경로계획을 하고 (그림 6) TCI 에서 Task-Level 프로그램을 로봇 프로그램으로 변환시켜준다(그림 7).

6. 결론 및 추후 과제

본 논문에서는 조립 계획 시스템을 제안하였다. Task-Level 언어는 로봇을 이용하여 조립 작업을 수행할 때 작업 자체를 중심으로 기술함으로써 로봇 명령어를 모르더라도 편리하게 작업을 수행할 수 있도록 하기위하여 개발되어 왔다. 제안된 시스템은 ODB 생성기, TDB 생성기, TCI 로 구성되는데 조립 계획이 실시간으로 이루어 질 수 있고, 시스템의 효용성을 보이기 위하여 13 개의 레고 블럭으로 구성되는 집모양을 조립하였다.

앞으로 이루어져야 할 연구로는 복잡한 부품의 모델링 방법과 가능한 조립순서들 중에서 최적 순서 자동생성, 비전이나 힘센서와 같은 외부 센서를 이용하는 Task-Level 언어 개발 등이 있다.

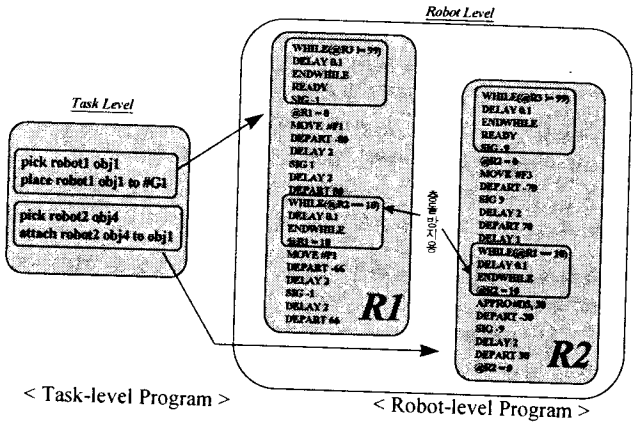

그림 8. 변환된 로봇 제어 명령어

Reference

[1] D. Halperin & R. H. Wilson, "Assembly Partitioning along Simple Path:the Cost of Multiple Translations," *Proc. of the IEEE International Conference on Robotics & Automation*, pp.1585-1592, May, 1995

[2] L. I. Lieberman and M.A.Wesley, "AUTOPASS : An Automatic Programming System for Computer Controlled Mechanical Assembly," *IBM J.Res.Develop*, Vol. 21, No. 4, pp. 321-333, 1977.

[3] E. C-Maniere, B. Espiau and E. Rutten, "A Task-Level Robot Programming Language and its Reactive Execution," *Proc. of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pp. 2751-2756, July, 1992.

[4] L. S. Homem de Mello and A. C. Sanderson, "Representations of Mechanical Assembly Sequences," *IEEE Trans. On Robotics and Automation*, Vol. 7. No. 2. pp.211-227, April, 1991

[5] E. Cervera, A. P. del Pobil, E. Marta and M. A. Serna, "A Sensor-Based Approach for Motion in Contact in Task Planning," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 468-473, May, 1995.

[6] J. Latombe, *Robot Motion Planning*, Boston, MA: Kluwer, 1991.

[7] J.P Thoms and P.N Nissanke, "A Graph-based formalism for Modelling Assembly Tasks," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 1296-1301, May, 1995.

[8] A. Stentz, "Optimal and Efficient Path Planning for Partially-Known Environments," *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 3310-3317, May, 1994.

[9] N.Y.Ko, and B.H.Lee, and M.S.Ko, "An Approach to Robot Motion Planning for Time-Varying Obstacle Avoidance Using View-Time Concept." *ROBOTICA*, Vol.11, pp315-327, July 1993.