碩士學位論文

휴머노이드 로봇을 위한 양팔 경로 생성에 대한 연구

濟州大學校 大學院

메카트로닉스工學科

金 昌 宗

2008 年 2 月

휴머노이드 로봇을 위한
양팔 경로 생성에 대한 연구

指導敎授 崔 劉 賢

金 昌 宗

이 論文을 工學 碩士學位 論文으로 提出함

2008 年 2 月

金昌宗의 工學 碩士學位 論文을 認准함

審査委員長 _____ 印

委 員 _____ 印

委 員 _____ 印

濟州大學校 大學院

2008 年 2 月

A Study on Multi-Arm Path Planning for Humanoid Robot

Chang-Jong Kim

(Supervised by professor Kyung-Hyun Choi)

A thesis submitted in partial fulfillment of the requirement for the Degree of Master of Engineering

2008. 2.

This thesis has been examined and approved.

Department of Mechatronics Engineering

GRADUATE SCHOOL

CHEJU NATIONAL UNIVERSITY

목 차

List of Figures

Summary

The domestic study for humanoid robot is mainly focused on walking and running etc. And the study for Robot Arms is emphasizing in motion that hold goods, but study that robot arms avoids obstacle autonomously is unprepared. And most study for robot arms is focused on the motion of one arm and is not considered about the motion of two arms at the same time.

The ultimate target in this paper is to suggest efficient path planning algorithm about work that is given about 13 DOF upper body with two arms of humanoid robot Maru-1 and to demonstrate it through 3D simulation that the real robot is modeled equally.

For this target, in this paper, kinematics of the robot upper body is analyzed to apply 13 degree of freedoms two arms of robot and body to path generation algorithm and inverse kinematics is analyzed to get each joint angle about the inputted target object of robot two arms. And merits and demerits of each algorithms is reviewed through investigation and study about classical path planning algorithm and the theory of sampling based algorithm to be possible to apply to multi-DOF system is analyzed.

There are RRT (Rapidly-exploring Random Tree) and PRM (Probabilistic Raodmap) in sampling based algorithm. RRT based algorithm is analyzed in priority in this paper. RRT define free configuration space and choose random possible configuration repeatedly and connect these configuration from initial configuration to goal configuration. This method can reduce unnecessary calculation that is used to construct road map in first step unlike PRM and is effective to find the path in configuration space of nonholonomic system and kinodynamic system. In this paper, the acceptance of these algorithm is demonstrated through the result of 3D simulation.

Ⅰ. 서 론

1. 연구 배경

휴머노이드 로봇의 주어진 작업을 성공적으로 수행하기 위해서는 인간의 두뇌작용에 해당하는 지능제어시스템과 이를 바탕으로 한 자율기능, 높은 수준의 인간과 로봇과의 통신을 위한 적절한 인터페이스, 고기능 센서를 이용한 감각기능, 로봇의 활동범위를 넓혀주는 이동기능, 요구되는 작업을 수행하기 위한 조작기능, 임무수행을 위한 처리기능 등이 필요하다. 기존의 로봇 팔에 해당하는 머니퓰레이터의 역할은 산업 현장에 투입되어 단순한 일을 반복 처리하는데 중점을 두어 개발 되었으나, 21세기 들어서 IT, BT 등 새로운 산업의 등장과 함께 사회구조 및 환경 변화에 따라 개인용 로봇이나 휴머노이드 로봇의 수요가 증가하게 되면서 한 차원 높은 로봇 팔은 기존의 단순한 작업 외에 인간과 같은 복잡한 동작을 수행해야 한다. 지금까지의 지능형 휴머노이드 로봇의 연구는 로봇의 걷는 능력을 개선하는 데 중점을 두었으나, 로봇이 손으로 물건을 자유자제로 집거나 옮기는 인간과 비슷한 작업을 수행하기 위해선 상체의 움직임, 특히 휴머노이드 로봇의 양팔과 몸체의 회전을 이용한 움직임을 효율적으로 계획하는 것은 필수 요건이다. 단순한 작업을 수행했던 기존의 로봇과는 달리 체계적이지 않은 환경에서 매우 복잡하고 다양한 작업을 수행해야 함에 따라 작업을 효율적으로 계획 할 수 있는 개선된 모션 계획(Motion Planning)기술이 필요하다. 6자유도이하의 산업용 로봇팔과는 달리 Maru-1 휴머노이드 로봇의 상체는 6자유도의 양팔이 달려 있고 몸체가 베이스에 해당하는 골반을 중심으로 회전을 할 수 있는 13자유도 시스템이다. 주어진 작업을 두 팔을 이용하여 효율적

으로 수행하기 위해서는 이런 다자유도 시스템의 작업계획을 할 수 있는 알고리즘 및 모듈이 필요하다.

2. 연구 목표

국내 휴머노이드 로봇에 대한 연구는 주로 걷기, 뛰기 등 보행을 위주로 한다. 로봇 팔에 대한 연구는 물건을 잡는 움직임에 중점을 두고 있지만, 로봇 팔이 자율적으로 장애물을 피하는 것에 대해서는 연구가 미비한 편이다. 그리고 대부분의 로봇 팔에 대한 연구는 한쪽 팔에 대한 것일 뿐 양팔을 동시에 사용하기 위한 연구가 미비하다.

본 논문에서는 로봇 팔의 장애물 회피를 위해 기존의 여러 가지 알고리즘의 장단점 및 적용 가능성을 조사 및 분석 하고 실제 Maru-1의 양팔이 달린 13자유도 상체에 대해서 주어진 작업에 대한 효율적인 경로 생성 알고리즘을 제안하여 실제 로봇의 기구학적인 설계와 똑같이 모델링된 3D시뮬레이션을 통해서 증명을 하는 것을 본 연구의 궁극적인 목표로 삼는다.

이를 위해서 로봇의 13자유도 양팔 및 몸체를 경로 생성 알고리즘에 적용하기 위한 기구학적인 분석 및 로봇 양팔의 목표 물체에 따른 각 조인트의 각도를 구하기 위한 역기구학 분석을 하고자 한다. 그리고 이제까지 연구된 경로 생성 알고리즘에 대한 조사 및 연구를 통해 각 알고리즘들의 원리 및 장단점들을 되 짚어보고, 요즘 연구되고 있는 로봇 팔과 같은 다자유도 시스템에 적용이 가능한 셈플링(Sampling)기반의 알고리즘에 대한 종류 및 원리들을 분석하여 Maru-1 로봇 상체에 대한 적용여부를 시뮬레이션을 통해 비교 분석할 것이다.

Ⅱ. 경로 생성 알고리즘 연구 동향

경로 생성 알고리즘은 크게는 기존에 연구 되었던 저 차원 시스템 위주의 경로 생성 알고리즘과 근대에 들어 시스템이 고차원이 되면서 고차원 시스템에 적합한 셈플링 기반의 알고리즘으로 나누어진다. 기존의 경로 생성알고리즘은 저 차원 시스템에는 대체로 적용하기 쉽고 빠르지만, 휴머노이드 로봇과 같은 고차원 시스템에는 적용이 어렵다. 그리하여 고차원 시스템에 맞는 셈플링 기반의 알고리즘을 연구할 필요가 있다.

이 장에서는 먼저 기존의 경로 생성 알고리즘의 예를 들어 그에 따른 원리를 간단히 짚어보고 장단점을 분석하였고, 셈플링 기반의 알고리즘에 대한 예를 들어 각각의 장단점 및 본 연구에 쓰이는 로봇 시스템에 대한 적합성을 분석 하였다.

1. 기존의 경로 생성 알고리즘

1.1 셀 분할(Cell Decomposition)

셀 분해법은 공간을 단순한 셀들의 집합으로 분해하여 가까운 셀들 간의 관계로 경로를 찾는 방법이다. 셀 분해법은 셀을 장애물의 형태에 의존적으로 분해하는 방법과 비 의존적으로 분해하는 방법이 있다.[2] 장애물 형태에 의존적으로 셀을 분해하는 방법은 셀의 형태와 개수가 장애물의 형태에 따라 결정된다. 이렇게 결정된 셀은 전체 개수는 적으나 셀 분해 과정이 복잡하고 셀 간의 관계가 처리하기 힘들어 진다. Fig. 1는 장애물 의존적 셀 분해법으로 찾은 경로의 예를 보여준다.

장애물 형태에 비 의존적으로 셀을 분해 하는 방법은 단순한 모양을 하고

있는 작은 크기의 셀로 공간을 분해하여 각각의 셀들이 장애물을 포함하고 있는 지 여부에 따라 종류를 구분 짓고 이 셀들을 종류에 따라 재구성하여 단순화시킨다. 셀의 모양과 위치가 장애물에 독립적이어서 데이터 처리에 용이하다. Fig. 2는 장애물에 비 의존적인 셀 분해법의 예를 보여준다. 또한, 셀 분해법은 활동영역을 셀 단위로 처리하기 때문에 활동영역에 데이터가 줄어들고 알고리즘이 간단하며 반복적으로 구현할 수 있다는 장점을 가지고 있다.

Fig. 1 Cell Decomposition to depend on the obstacle

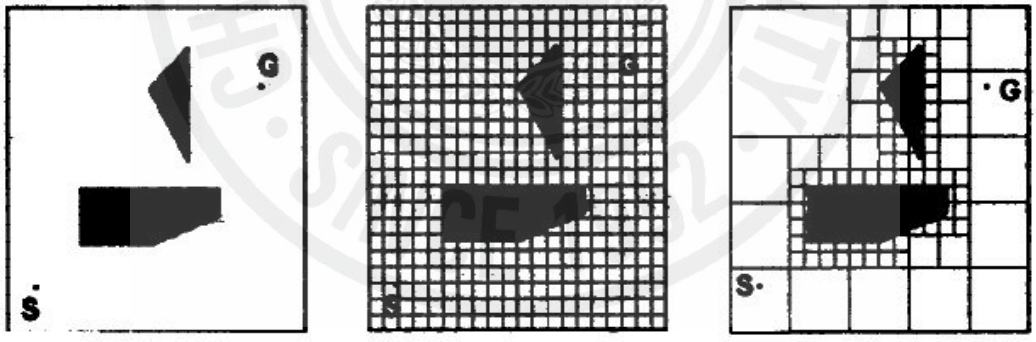


Fig. 2 Cell Decomposition not to depend on the obstacle

1.2 그래프지도(Graph Map)

그래프지도는 Fig.3(1)과 같이 장애물을 볼록한 다각형들로 표시하고 다각형들의 꼭지점이나 모서리를 처리해서 지도를 만드는 방법이다[3]. 다각형을 표시할 때 실제 장애물의 크기보다 크게 해서 로봇이 다각형의 모서리를 따라가

도 장애물에 부딪치지 않도록 한다. 다각형의 꼭짓점을 노드(Node)라 하고, 모서리를 엣지(Edge)라 한다. 그래프지도에는 가시그래프지도 그리고 접선그래프지도 등이 있다. 가시그래프지도는 Fig.3(2)와 같이 각각의 노드에서 보이는 점들을 모두 연결한 그래프지도이다. 가시그래프지도에는 장애물을 피하지 않고 장애물로 돌진하는 경로가 존재한다. 가시그래프지도에서 장애물로 돌진하는 경로를 제외시킨 지도, 즉 두 다각형에 연결된 선분 중 두 다각형 모두의 접선이 되는 선분만을 남겨둔 지도를 접선그래프 지도라고 한다. Fig.3(3)은 접선그래프의 예이다. 가시그래프지도와 접선그래프지도는 로봇을 장애물에 가까이 접근시킨다는 특성을 가진다. 만약 로봇이 경로 추적 과정에서 바퀴의 미끄러짐이나 센서의 부정확성으로 인해 경로에서 조금만 이탈하면 곧바로 장애물에 부딪친다. 이를 해결하기 위해 접선그래프지도에서 노드까리 연결한 선의 중점을 찾고 그 점들의 가능한 연결을 모두 그래프로 그린 것을 마크링크그래프지도라고 한다. Fig.3(4)는 마크링크그래프지도의 예이다.

그래프지도의 일반적인 형태는 접선그래프지도이고, 가시그래프지도는 정밀하게 장애물을 피해가야 할 때 사용되며, 마크링크그래프지도는 로봇이 장애물에서 멀리 떨어져서 주행해야 할 때 사용된다.

그래프지도는 장애물의 모서리를 찾아야 하고, 모서리들로부터 꼭지점을 찾아서 장애물을 다각형으로 표현해야 하며, 그 다각형을 볼록하게 만들어야 하고, 다각형을 다각형의 모서리로 로봇이 통과해도 장애물에 부딪치지 않을 만큼 늘려야 한다. 그리고 가시그래프지도 또는 접선그래프지도, 마크링크그래프지도를 작성해야 하기 때문에 시간이 오래 걸린다. 그러므로 경로 계획에 많은 시간이 걸려 효율적이지 못하다.

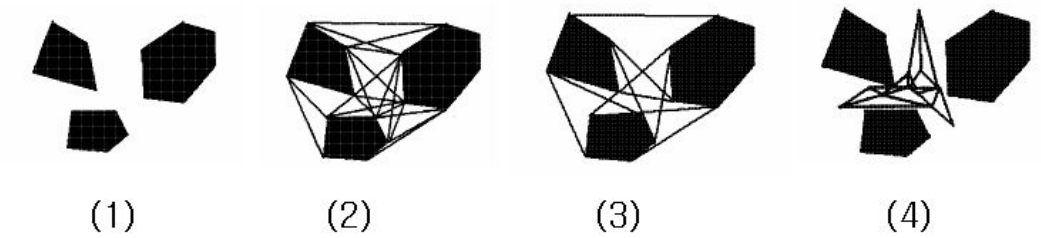


Fig.3 (1)Graph (2)Spine Graph (3)Tangential Graph (4)Mark Link Graph

1.3 Elastic Strips

Elastic Strip 방법은 로봇의 궤적을 탄성물질이라 가정하여 장애물을 안전하게 회피하도록 한 방법이다[4]. 장애물 접근 시 물체 사이에 가상 힘이 작용한다는 것에서는 인공전위계 개념과 유사하나 목표지점까지의 도달 동안 탄성 힘이 작용한다는 데서 인공전위계방식 보다는 보다 안정된 주행 기술을 구사한다. Fig.4처럼 이 방법에는 내부 힘과 외부 힘이 존재 하며 로봇의 주행동안 이 두 힘을 합하여 구해진 벡터의 방향과 힘에 의해 로봇이 목표지점을 찾아가게 된다.

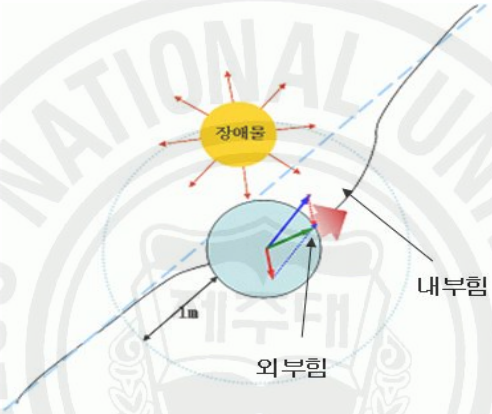
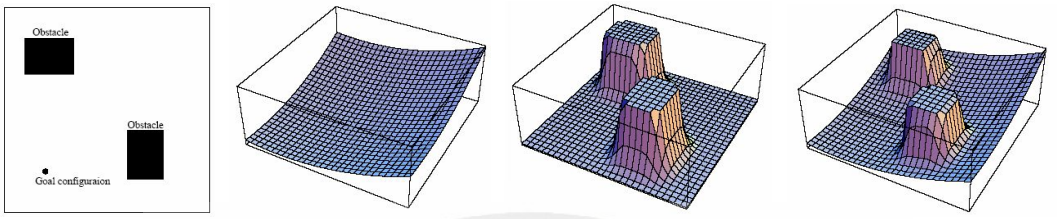


Fig. 4 Computation of the elastic force

이 방법에는 내부 힘과 외부 힘이 존재 하는데, 내부 힘은 로봇이 정확한 목적지까지 이동할 수 있도록 도와주는 힘이라 할 수 있다. 이 힘은 로봇의 궤적상의 형상사이에 스프링이 연결되어 있다 가정한다. 이 궤적으로 이동 중에 궤적을 이탈하게 되면 이 궤적의 이탈을 방지하기 위하여 탄성 힘인 내부 힘 이 작용하여 원래의 궤적을 찾게 도와준다. 이때 궤적을 많이 이탈하게 되면 내부 힘이 커지게 되며 원래 계획되어진 궤적에 가까워지게 되면 서서히 내부 힘이 감소하게 된다. 외부 힘은 자율이동 로봇의 경로 계획이나 작업 공간 내에서 장애물과 접했을 때에 충돌 없이 목적지까지 이동하도록 하는 힘으로서 포텐셜 필드 방식(Potential Fields Method)과 유사하다. 포텐셜 필드 방법은 로봇을 포텐셜 함수의 영향을 받는 물체로 취급하는 것이다. 포텐셜 분

포를 보면 장애물을 제외한 모든 모바일 로봇의 작업 공간 영역에서 도착점으로 기울어져 내려가는 현상을 하게 된다. 따라서, Fig. 5처럼 로봇은 시작점에서 음의 기울기를 따라가면 도착점에 도착하게 된다.



(a)The environment (b)Goal Field (c)Obstacle Field (d)Potential Field

Fig. 5 Potential Field Method

2. 셈플링(Sampling)기반의 알고리즘[5]

이전 장에서 보았던 비충돌 경로 생성알고리즘은 셈플링 기반 알고리즘과 달리 완성적이고 결정론적이다. 그러나 실제로 이 알고리즘들은 모바일 로봇 등의 낮은 차원의 문제에 적용이 가능하나, 다 자유도의 로봇 경로 생성에 적용이 어렵고, 고차원의 구성 공간(Configuration Space)의 확장에 대한 계산이 복잡하며, 포텐셜 필드(Potential field)는 local minima에 빠져들게 되는 등 로봇팔과 상체 등 고 차원의 복잡한 경로 생성 문제에는 적용이 어렵다. 반면 셈플링 기반의 알고리즘은 굉장히 많은 로봇 시스템에 성공적으로 적용이 되며 오늘날 경로 생성 문제를 푸는데 표준적인 방법으로 각광받고 있다. 이 방법은 구성 공간 내에서 분명한 기하학적인 표현과정이 없다. 대신 그 방법은 구성공간으로부터 임의로 로봇의 구성 셈플을 추출하고 로컬 플래너(local planner)에 의해서 추출된 셈플의 연결을 시도한다. 이 방법은 적절한 로컬 플래너에 간단히 접근함으로서 여러 다른 로봇에 적용가능 하다. 그래서 대부분의 고전적인 경로 생성 알고리즘보다 매우 포괄적이다.

샘플링 알고리즘에는 대표적으로 PRM(Probabilistic Roadmap)과 RRT(Rapidly-exploring Random Tree)가 있다. 이 장에서는 두 알고리즘을 비교하여 원리 및 특징들을 살펴 볼 것이다.

2.1 PRM(Probabilistic Roadmap)[5]

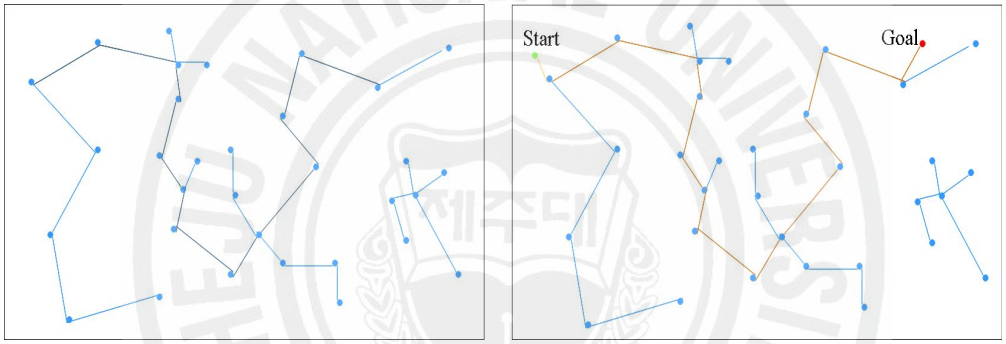
PRM(Probabilistic Roadmap)는 움직이지 않는 장애물 사이를 가상으로 움직이는 로봇의 비충돌 경로를 계산하는 경로 계획 방법론이다. PRM은 충돌체크에 반응이 빠르며 구성공간의 정확한 표현을 계산하는 것을 피하여 경로 계산에 대한 시간을 단축시킨다. 그리고 다자유도 시스템에 적용이 가능하다.

PRM에는 학습 단계(Learning Phase) 와 쿼리 단계(Query Phase) 두 단계로 구성되어 있다. 학습 단계에서는 로컬 플래너(Local Planner[6])에 의해 로봇의 임의의 자유 공간(Random Free Configurations)을 생성하고 간단하게 그것들을 연결함으로서 Probabilistic roadmap를 구축한다. 로컬 플래너에 의해 계산된 노드(configuration)와 노드들을 연결하는 엣지(path segment)를 그래프로 저장한다. Fig. 6에서는 로드맵 구축 과정을 나타내었고, Fig. 7(a)는 노드를 엣지로 연결한 그림이다. 이 그림에서 보면 노드와 엣지가 반복적으로 구해지면서 그래프에 저장되는 과정이 잘 나타나있다.

그 다음 쿼리 단계에서는 그 전 단계에서 구해진 로드맵을 바탕으로 초기 구성 상태(Start configuration)과 목표 구성 상태(Goal configuration)를 입력하고 Distance Metrics를 이용하여 이 두 노드에서 가장 가까운 구성 상태(configuration)을 검색한다. 그다음 이 두 가까운 구성 상태사이의 경로를 계산하여 로봇의 실행 가능한 경로를 완성 시킨다. Fig.7(b)에서는 두 입력과 최종의 경로 생성을 나타내고 있다. Thierry Simeon은 이 알고리즘을 머니퓰레이션 플래닝에 적용하였다[7]. 그는 이전에 언급되었던 구성 상태에 대한 이산적인 집합보다는 움직이는 물체에 대한 접근 및 파지를 모델링하는 것에 대하여 연속적인 집합을 언급할 수 있는 일반적인 머니퓰레이션 플래닝 접근법을 제안하였다.

BUILD ROADMAP
1 $G.init(); \ i \leftarrow 0;$
2 *while* $i < N$
3 if $\alpha(i) \in C_{free}$ *then*
4 $G.add\_vertex(\alpha(i)); \ i \leftarrow i+1;$
5 for *each* $q \in NEIGHBORHOOD(\alpha(i)), G)$
6 if $((!G.same\_component(\alpha(i), q)$ and $CONNECT(\alpha(i))$ *then*
7 $G.add\_edge(\alpha(i), q);$

Fig. 6 The construction algorithm for roadmap



(a)Learning Phase (b)Query Phase

Fig. 7 PRM process

PRM의 특징은 초기 구성 상태와 목표 구성 상태의 연결을 좀 더 세밀하게 하려면 로드맵상의 구성 상태를 더 많이 추출하여 로드맵을 더 빽빽하게 필요가 있으나 계산의 시간이 많이 걸려야한다. 그리고 경로 생성 전에 그래프를 먼저 구축해야 되므로 최종 경로 외의 불필요한 수많은 구성 상태 또는 엣지의 계산이 많아질 수 있으므로 상황에 따라서 단점이 될 수 있다. 그리고 이 알고리즘은 일반적인 nonholonomic planning 문제와 kinodynamic 문제에는 적용하지 못한다.

2.2 Rapidly-exploring Random Tree (RRT)

RRT[8] 알고리즘은 먼저 구성 공간을 구하고 초기 구성 상태(q_{init})와 목표 구성 상태(q_{goal})를 입력하여 q_{init}에서부터 임의의 가능한 q를 선정하고 모서리로 연결을 한다. 그 다음 q_{goal}에 도달 할 때까지 그 방법이 반복되고 최종으로 최적의 경로가 생성된다. 이 방법은 PRM과 달리 첫 단계의 로드맵을 구축하는 불필요한 계산이 줄어든다. 그리고 다 자유도 시스템에 잘 적합할 뿐 아니라 장애물과 nonholonomic 시스템, kinodynamic 시스템을 가지는 고차원 공간을 빨리 찾는데 효과적이다[9].

James J. Kuffner, Jr. 는 RRT와 Connect heuristic을 결합한 RRT-Connect라는 알고리즘을 제안하였다[10]. 이 알고리즘은 RRT와는 달리 q_{init}와 q_{goal}부터 경로가 생성되며 두 경로가 중간 지점에서 만나면 경로를 연결하는 방식이다. 이 방식은 경로를 생성하는 데에 기존 RRT 보다 시간을 많이 줄일 수 있다. RRT는 주로 국외의 휴머노이드 로봇에도 많이 적용되고 있다. Eiichi Yoshida는 Fig. 8과 같은 30자유도 HRP-2 로봇의 충돌회피 및 모션 플래닝에 적용하였고[11], James Kuffner는 Fig.9와 같은 H6 로봇의 메니퓰레이션 작업 계획에 적용하였다[12]. 다음 장에서는 이 RRT알고리즘과 진보된 RRT 알고리즘들을 분석하고, 연구하고자 하는 휴머노이드 로봇의 시뮬레이션에 적용시켜 적용 여부를 밝히고자 한다.

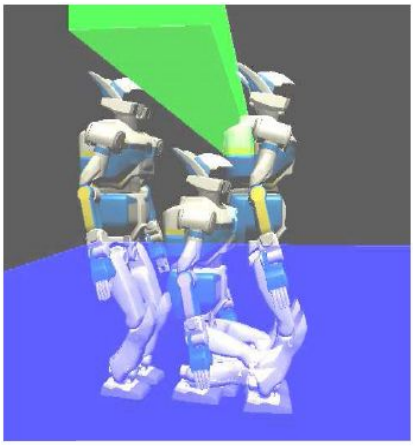


Fig. 8 Avoidance of HFP-2

Fig. 9 Manipulation of H6

Ⅲ. 휴머노이드 로봇 상체의 기구학적 해석

1. 휴머노이드 로봇 Maru-1의 구조

본 연구에서 사용한 휴머노이드 로봇은 Fig.10에서 보는 바와 같이 Maru-1 이다. 이 로봇은 일본의 아시모나 KAIST의 휴보와는 달리 네트워크를 활용하여 로봇 외부의 컴퓨터 시스템을 통해 다양한 인지능력과 인공지능을 지원하는 유비쿼터스 시대의 로봇이라 할 수 있다.

Maru-1의 키는150cm이고 몸무게는 67kg, 최대 보행 속도는 시속 0.9km이다. 자유도는 각 다리 6자유도, 각 팔 6자유도, 각 손 4자유도, 목 2자유도, 허리 1자유도로 총 35자유도로 이루어져 있으며 센서로는 스테레오 카메라, 마이크 , 각 조인트의 힘과 토크를 측정 할 수 있는 힘/토크 센서, 자세 센서들이 포함 되어 있다. 운영체제로는 RT-linux pro가 사용된다. 전원으로는 리튬-폴리머베터리를 사용하고 한번 충전 시 1시간을 사용 할 수 있다.

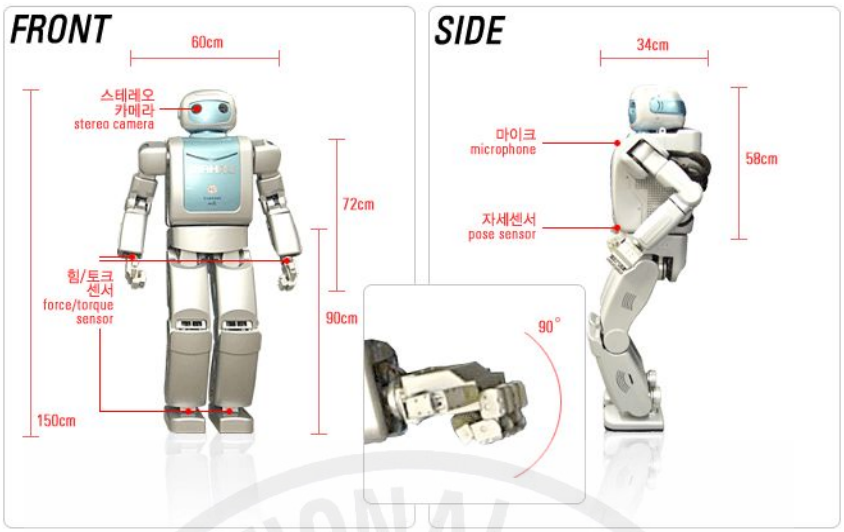


Fig 10. Appearance of Humanoid Robot Maru-1

2. 휴머노이드 로봇의 상체 기구학 해석

2.1 순기구학 해석

본 연구에 사용되는 로봇의 상체는 각각 6자유도(Degree Of Freedom)인 양 팔과 허리를 회전축으로 하는 1자유도의 몸체로 구성된 총 13-DOF 시스템이다. 로봇 상체의 좌표계 설정은 Fig. 11와 같다.

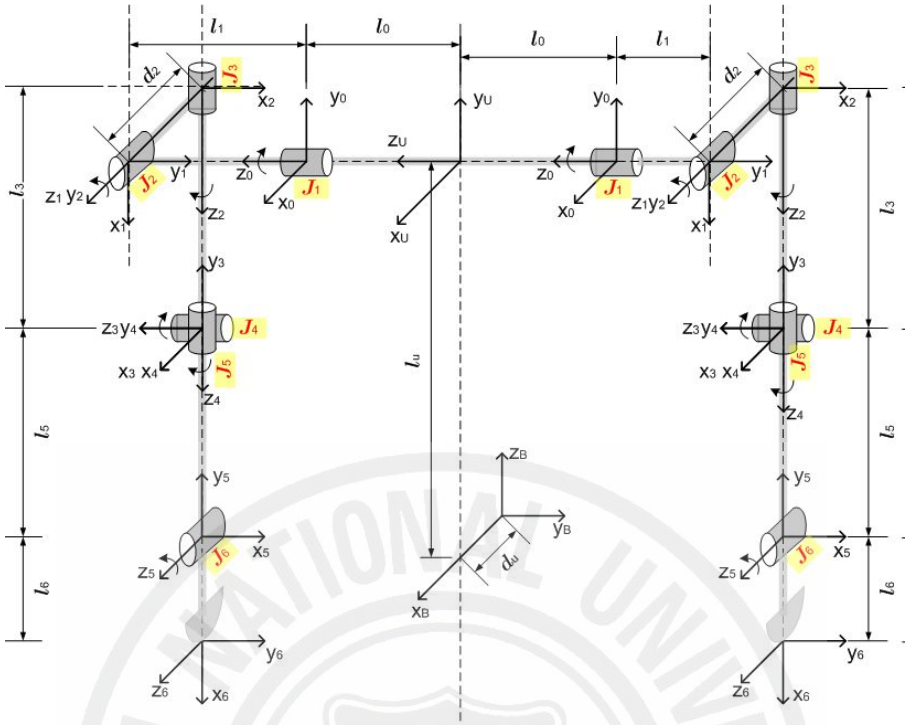


Fig. 11 Setting the coordinates system of each joints

Fig.11에서 보는 바와 같이, 팔은 몸체의 좌표계를 기준으로 어깨 부분의 x, y, z축에 대한 회전 3관절, 팔꿈치 부분의 y, z축 회전 2관절, x 축 회전 1관절로 이루어져 있다. 이 설정된 좌표계를 바탕으로 기구학 해석에 쓰이게 될 D-H(Denavit-Hartenberg) 파라미터를 Table.1 과 같이 나타낼 수 있다. 기구학 분석을 쉽게 하기 위해서는 몸체와 팔 부분을 따로 구분해서 분석해야 한다. 그래서 몸체 회전 각도(θ_b)을 0으로 놓았을 경우 몸체의 원점에서 x축으로 d_u, z축으로 l_u, x축으로 90도 회전을 하면, 오른 팔의 경우 z방향으로 (l_0+l_1)이동, 왼팔의 경우 $-(l_0+l_1)$이동한 꼴이 된다. 그러므로 몸체에서 오른쪽 그리고 왼쪽 어깨까지의 동차 행렬을 구하고, 한쪽 팔에 대한 어깨부터 손끝까지의 동차 행렬을 구해서 기구학 분석을 해야 한다.

Table. 1 Denavit-Hartenberg Parameters

Link	θ_i	d_i	a_i	α_i	θ_{i0}	$l_u = 0.284$
$B-U$	θ_b	l_u	d_u	$\pi/2$	0	$d_u = 0.04$
$U-J_0$	0	$-l_0$	0	0	0	$l_0 = 0.1615$
L_1	θ_1	$-l_1$	0	$-\pi/2$	$-\pi/2$	$l_1 = 0.061$
L_2	θ_2	d_2	0	$\pi/2$	$\pi/2$	$l_3 = 0.224$
L_3	θ_3	l_3	0	$-\pi/2$	$\pi/2$	$d_2 = 0.0055$
L_4	θ_4	0	0	$\pi/2$	0	$l_5 = 0.225$
L_5	θ_5	l_5	0	$-\pi/2$	$-\pi/2$	$l_6 = 0.041$
L_6	θ_6	0	l_6	0	$-\pi/2$	

그리고 각 조인트 좌표계간의 변환행렬을 나타내는 동차행렬(Homogeneous Matrix)을 Table.1의 파라미터를 이용하여 구해 보면, 몸체부터 왼쪽 어깨까지의 동차행렬은 식(1)과 같이 몸체부터 오른쪽 어깨까지의 동차 행렬은 식(2)과 같이 구할 수 있으며, 몸체부터 손끝까지의 각각의 조인트에 대한 동차행렬을 식(3)로 나타낼 수 있다.[1] 이로부터 기저 좌표계와 마지막 좌표계 사이의 변환 관계를 나타내는 동차 행렬 식(4)을 얻게 된다.

$$
\begin{aligned}
{}^{B}T_{LS} &= rot(z, \Theta_{body})\, tran(x, d_u)\, tran(z, l_u)\, rot(x, \pi/2)\, tran(z, -(l_0 + l_1)) \\
&= \begin{pmatrix} \cos\Theta_b & 0 & \sin\Theta_b & -l_0 \sin\Theta_b + d_u \cos\Theta_b \\ \sin\Theta_b & 0 & -\cos\Theta_b & l_0 \cos\Theta_b + d_u \sin\Theta_b \\ 0 & 1 & 0 & l_u \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
\tag{1}
$$

$$
\begin{aligned}
{}^{B}T_{LS} &= rot(z, \Theta_{body})\, tran(x, d_u)\, tran(z, l_u)\, rot(x, \pi/2)\, tran(z, -(l_0 + l_1)) \\
&= \begin{pmatrix} \cos\Theta_b & 0 & \sin\Theta_b & l_0 \sin\Theta_b + d_u \cos\Theta_b \\ \sin\Theta_b & 0 & -\cos\Theta_b & -l_0 \cos\Theta_b + d_u \sin\Theta_b \\ 0 & 1 & 0 & l_u \\ 0 & 0 & 0 & 1 \end{pmatrix}
\end{aligned}
\tag{2}
$$

$$A_1 = \begin{pmatrix} \cos\Theta_1 & 0 & -\sin\Theta_1 & 0 \\ \sin\Theta_1 & 0 & \cos\Theta_1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_2 = \begin{pmatrix} \cos\Theta_2 & 0 & -\sin\Theta_2 & 0 \\ \sin\Theta_2 & 0 & \cos\Theta_2 & 0 \\ 0 & -1 & 0 & d_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_3 = \begin{pmatrix} \cos\Theta_3 & 0 & -\sin\Theta_3 & 0 \\ \sin\Theta_3 & 0 & \cos\Theta_3 & 0 \\ 0 & -1 & 0 & l_3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_4 = \begin{pmatrix} \cos\Theta_4 & 0 & \sin\Theta_4 & 0 \\ \sin\Theta_4 & 0 & -\cos\Theta_4 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_5 = \begin{pmatrix} \cos\Theta_5 & 0 & -\sin\Theta_5 & 0 \\ \sin\Theta_5 & 0 & \cos\Theta_5 & 0 \\ 0 & -1 & 0 & l_5 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$A_6 = \begin{pmatrix} \cos\Theta_6 & \sin\Theta_6 & 0 & l_6\cos\Theta_6 \\ \sin\Theta_6 & \cos\Theta_6 & 0 & l_6\sin\Theta_6 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3)$$

$${}^0_6T = A_1A_2A_3A_4A_5A_6 = \begin{pmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (4)$$

n_x= C6 (C5 (C4 (C1 C2 C3-S1 S3)-C1 S2 S4)+(-C3 S1-C1 C2 S3) S5)+(-C1 C4 S2-(C1 C2 C3-S1 S3) S4) S6

n_y= C6 (C5 (C4 (C2 C3 S1+C1 S3)-S1 S2 S4)+(C1 C3-C2 S1 S3) S5)+(-C4 S1 S2-(C2 C3 S1+C1 S3) S4) S6

n_z= C6 (C5 (-C3 C4 S2-C2 S4)+S2 S3 S5)+(-C2 C4+C3 S2 S4) S6

o_x= C6 (-C1 C4 S2-(C1 C2 C3-S1 S3) S4)-(C5 (C4 (C1 C2 C3-S1 S3)-C1 S2 S4)+(-C3 S1-C1 C2 S3) S5) S6

o_y= C6 (-C4 S1 S2-(C2 C3 S1+C1 S3) S4)-(C5 (C4 (C2 C3 S1+C1 S3)-S1 S2 S4)+(C1 C3-C2 S1 S3) S5) S6

o_z= C6 (-C2 C4+C3 S2 S4)-(C5 (-C3 C4 S2-C2 S4)+S2 S3 S5) S6

a_x= C5 (-C3 S1-C1 C2 S3)-(C4 (C1 C2 C3-S1 S3)-C1 S2 S4) S5

a_y= C5 (C1 C3-C2 S1 S3)-(C4 (C2 C3 S1+C1 S3)-S1 S2 S4) S5

a_z= C5 S2 S3-(-C3 C4 S2-C2 S4) S5

p_x= -d2 S1+C1 l3 S2+l6S6 (-C1 C4 S2-(C1 C2 C3-S1 S3) S4)+l5 (C1 C4 S2+(C1 C2 C3-S1 S3) S4)+l6C6 (C5 (C4 (C1 C2 C3-S1 S3)-C1 S2 S4)+(-C3 S1-C1 C2 S3) S5)

p_y= C1 d2+l3 S1 S2+l6S6 (-C4 S1 S2-(C2 C3 S1+C1 S3) S4)+l5 (C4 S1 S2+(C2 C3 S1+C1 S3) S4)+l6C6 (C5 (C4 (C2 C3 S1+C1 S3)-S1 S2 S4)+(C1 C3-C2 S1 S3) S5)

p_z=C2 l3+l5 (C2 C4-C3 S2 S4)+l6S6 (-C2 C4+C3 S2 S4)+l6C6 (C5 (-C3 C4 S2-C2 S4)+S2 S3 S5)

2.2 역기구학 해석

본 연구에 쓰이는 로봇의 상체는 대수적인 방법으로만은 해석이 불가능 하므로 기하학적인 방법을 사용해야 한다.

로봇 상체의 전체 조인트에 대한 각도를 구하려면 먼저 로봇 허리부분의 몸체 회전 조인트(Θ_{body})를 먼저 구해야 한다. Fig.12에서처럼 Θ_{body}를 구하기 위해서 두 손에 대한 좌표와 방향 성분이 주어졌다고 가정하면, 두 좌표의 x, y 성분만을 사용하여 식(5)와 같이 두 손의 중점중의 x($c.x$), y($c.y$)성분을 구할 수 있다. 로봇은 그 중점을 가르쳐야만 두 목표 물체에 대해서 두 손을 좀 더 자유롭게 사용 할 수 있다. 그리고 이 두 성분과 식(6)을 이용하여 Θ_{body}를 구할 수 있다.

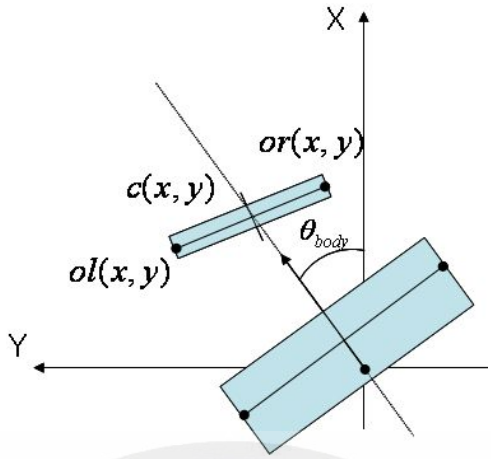


Fig.12 Computation of Θ_{body}

$$c.x = \frac{or.x - ol.x}{2} \quad , \quad c.y = \frac{or.y - ol.y}{2}$$

(5)

$$\Theta_{body} = asin\left(\frac{c.y}{\sqrt{c.x^2 + c.y^2}}\right)$$

(6)

나머지 로봇의 두 팔에 대한 각각의 조인트는 로봇의 목표물체 위치 즉, 두 손의 목표 위치를 Θ_{body}로 반대로 회전 시키고 로봇의 어깨부터 손끝까지의 각각의 각도를 구하면 된다. 즉, 식(7)(8)과 같이 전체 몸체에서 손끝까지의 동차행렬에 몸체에서 어깨까지의 동차행렬에 대한 역행렬을 곱하면 어깨에서 손끝까지의 동차행렬을 구할 수 있다.

$${}^{B}T_{LS}\,{}^{LS}T_{LH} = {}^{B}T_{LH} \quad , \quad {}^{LS}T_{LH} = {}^{B}T_{LS}^{-1}\,{}^{B}T_{LH}$$

(7)

$${}^{B}T_{RS}\,{}^{RS}T_{RH} = {}^{B}T_{RH} \quad , \quad {}^{RS}T_{RH} = {}^{B}T_{RS}^{-1}\,{}^{B}T_{RH}$$

(8)

그다음에 Fig.13을 참고하면, 해석을 위해선 먼저 손목의 위치(P_w)가 구해지고, 그 다음 팔꿈치의 위치(P_e)가 구해져야 한다. 손의 위치(P_h)는 목표점의 위치와 좌표에 의해 구할 수 있다고, 식(9)(10)과 같이 P_w는 P_h에서의 좌표계를 중심으로 l_6만큼 빼면 구할 수 있다.

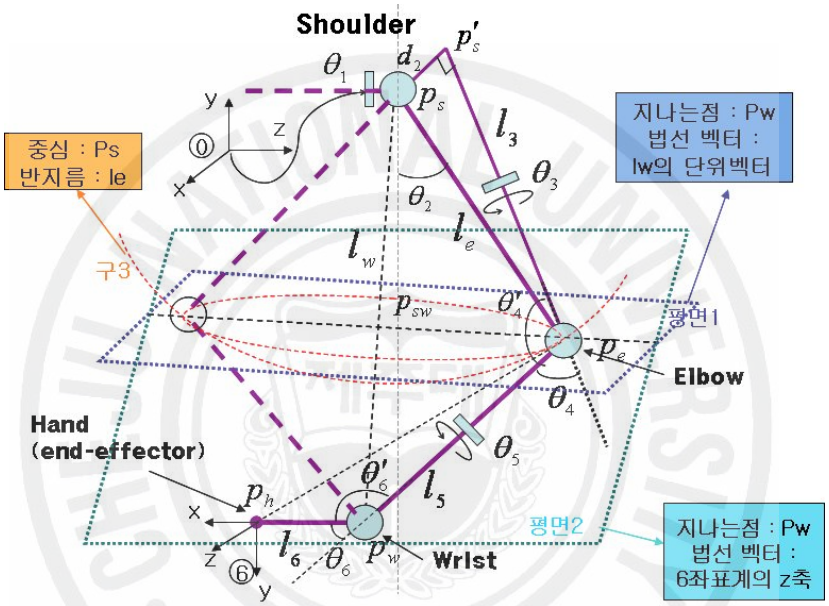


Fig.13 Inverse Kinematics of 6-DOF Robot Arm

$$P_{w-h} = l_6 \cdot [P_w(1,1); P_w(1,2); P_w(1,3)] \quad (9)$$

$$P_w = P_h - P_{w-h} \quad (10)$$

그리고 P_e는 P_w와 P_s를 이용하여 구해진다. Fig.4 에서처럼 평면1은 l_w를 법선벡터로 하고 P_s를 지나며, 평면2는 P_w를 지나고 법선 벡터가 P_w 조인트 좌표계의 z방향이며, 구3은 P_s를 중심으로 하고 반지름이 l_e이

다. 평면1, 평면2, 구3은 각각 식(11),식(12),식(13)으로 나타낼 수 있다. P_e는 평면1과 평면2, 구3의 교점이다. 그러면 P_e는 식(14)와 같이 구할 수 있다. $P_e=(x_e, y_e, z_e)$라고하고 식(14)와 식(15)의 P_e를 같다고 하면 식(16), 식(17)에 의해서 θ_2 와θ_1를 구할 수 있다. 여기서, A행렬은 각 조인트의 좌표계에 대한 동차 행렬로서 Table.1을 이용하여 구할 수 있다.

$$x_w x + y_w y + z_w z - l_w' l_w = 0$$

(11)

$$a_x x + a_y y + a_z z - (a_x x_w + a_y y_w + a_z z_w) = 0$$

(12)

$$\frac{x-x'}{a} = \frac{y-y'}{b} = \frac{z-z'}{c}$$

(13)

(단, 방향계수 $(a, b, c) = (a_z y_w - a_y z_w, a_x z_w - a_z x_w, a_y x_w - a_x y_w)$

$$P_e = \left(\frac{a}{c} z_e + x', \frac{b}{c} z_e + y', \frac{-B \pm \sqrt{B^2 - c^2(a^2+b^2+c^2)(x'^2+y'^2-l_e^2)}}{a^2+b^2+c^2} \right)$$

(14)

(단, $B = cax' + cby'$)

$$P_e = (A_1 A_2 A_3(1,4), A_1 A_2 A_3(2,4), A_1 A_2 A_3(3,4))$$

(15)

$$\Theta_2 = \cos^{-1} \frac{z_e}{l_3}$$

(16)

$$\Theta_1 = -\cos^{-1} \frac{l_3 x_e \sin\Theta_2 + d_2 y_e}{d_2^2 + l_3^2 \sin^2\Theta_2}$$

(17)

θ_4는 식(18)(19)와 같이 P_s', P_e, P_w 세 점을 이루는 삼각형에서 코사인 법칙에 의해 구할 수 있다.

$$P_s' = (-d_2 \sin\Theta_1, d_2 \cos\Theta_1, 0),\ P_e = (x_e, y_e, z_e),\ P_w = (x_w, y_w, z_w) \quad (18)$$

$$\Theta_4 = \frac{\pi}{2} - \cos^{-1}\left(l_3^2 + l_5^2 - \frac{(x_w + d_2 \sin\Theta_1)^2 + (y_w - d_2 \cos\Theta_1)^2 + z_w^2}{2 l_3 l_5} \right) \quad (19)$$

θ_3는 식(20),(21)과 같이 $A_1 A_2 A_3 A_4 A_5$의 (3,4)요소와 z_w가 같다는 것을 이용하여 구할 수 있다.

$$z_w = l_3 \cos\Theta_2 + l_5(\cos\Theta_2 \cos\Theta_4 - \cos\Theta_3 \sin\Theta_2 \sin\Theta_4) \quad (20)$$

$$\Theta_3 = \cos^{-1}\left(\frac{1}{\tan\Theta_2 \tan\Theta_4} - \frac{z_w - l_3 \cos\Theta_2}{l_5 \sin\Theta_2 \sin\Theta_4} \right) \quad (21)$$

θ_6는 식(22)과 같이 P_h, P_e, P_w 세 점을 이루는 삼각형에서 구할 수 있다.

$$\Theta_6 = \frac{\pi}{2} - \cos^{-1}\left(\frac{l_5^2 + l_6^2 - (x_h - x_e)^2 + (y_h - y_e)^2 + (z_h - z_e)^2}{2 l_5 l_6} \right) \quad (22)$$

θ_5는 식(23)(24)과 같이 $A_1 A_2 A_3 A_4 A_5 A_6$와 ${}^S T_H$의 성분을 대입하여 구해질 수 있다. 이 식에서 a, b, c, d 는 각 성분들이 된다.

$$A_1 A_2 A_3 A_4 A_5 A_6 = {}^{s}T_h,\ L(2,3) = R(2,3),\ L(3,3) = R(3,3)$$

(23)

$$\Theta_5 = a\cos\left(\frac{ba_z + d_y}{bc + ad}\right)$$

(24)

(단 $a = \cos\Theta_1\cos\Theta_3 - \cos\Theta_2\sin\Theta_1\sin\Theta_3$

$b = \cos\Theta_4(\cos\Theta_2\cos\Theta_3\sin\Theta_1 + \cos\Theta_1\sin\Theta_3) - \sin\Theta_1\sin\Theta_2\sin\Theta_4$

$c = \sin\Theta_2\sin\Theta_3$

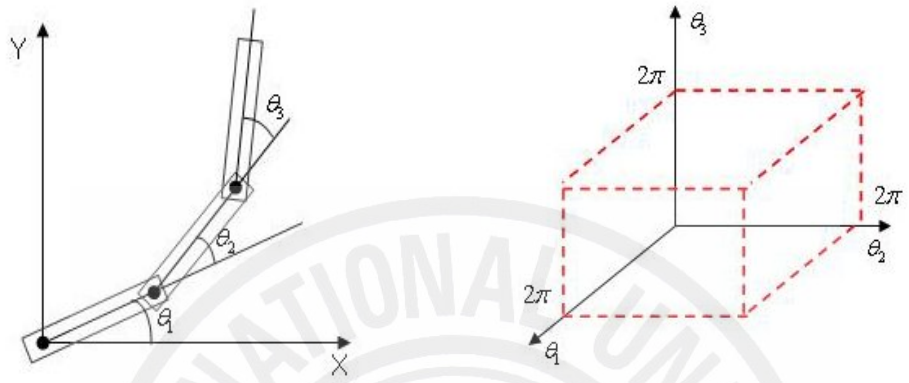
$d = \cos\Theta_3\cos\Theta_4\sin\Theta_2 + \cos\Theta_2\sin\Theta_4$)

Ⅳ. RRT 적용을 위한 선 처리 작업

1. 구성공간(Configuration Space)

구성 공간(C)[13]은 구성 상태(Configuration state(q))의 집합이다. q는 시스템에 따라 틀리지만, 로봇팔의 경우 조인트의 개수 즉, 자유도의 개수로 본 연구에 사용되는 Maru-1의 상체와 같은 경우 13자유도 이므로 q=($\Theta_1, \Theta_2, \Theta_3, \Theta_4, \Theta_5, \Theta_6, \Theta_7, \Theta_8, \Theta_9, \Theta_{10}, \Theta_{11}, \Theta_{12}, \Theta_{13}$)가 된다. 그러므로 C는 13차원이 되며, 영역의 크기는 각 각도의 제한에 따라 달라진다. 예를 들면 2차원에서 3개의 링크로 이루어진 3자유도 링크의 경우 q=($\Theta_1, \Theta_2, \Theta_3$)가 되며 Fig. 14 와 같이 작업공간과 C를 구분할 수 있다. 샘플링 경로 생성에서 C를 사용하는 이유는 작업 공간의 경우 end-effect가 장애물을 피한다고해도 다른 조인트 및 링크가 장애물에 부딪힐 수 있으나, C에서는 로봇팔의 자세를 q라

는 한 점으로 나타내므로 C_{obs}과 C_{free}를 명확하게 하면, 가능한 q사이의 경로를 생성시켜주면 된다. 그러나 13자유도의 C는 기하학적으로는 표현이 불가능하다.



(a) (b)

Fig. 14 Work Space(a) and Configuration Space(b)

2. 장애물 공간(Configuration Obstacle : C_{obs})과 자유공간(Configuration Free : C_{free})

Fig. 15는 2자유도-2링크 시스템에 대한 작업 공간과 C의 관계를 나타낸다. 이처럼 작업공간에서의 장애물은 C에서의 C_{obs}라는 영역으로 표현이 되고 경로생성은 C에서 C_{obs}을 제외한 C_{free}에서 q_{init}와 q_{goal}을 연결하는 것이다. 각각의 장애물은 색깔이 같은 각각의 C_{obs}로 나타내어지고 나머지 흰 영역이 C_{free}이다.

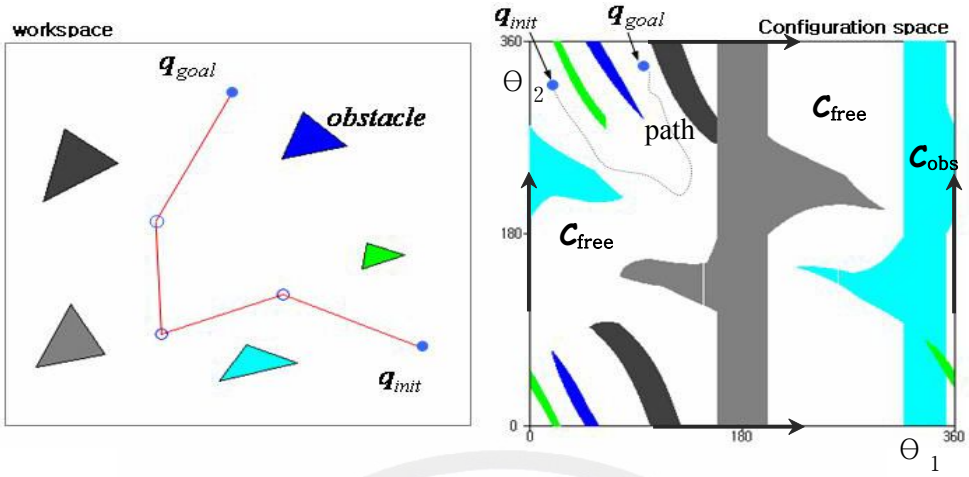


Fig. 15 Configuration Obstacle(C_{obs}) and Configuration Free(C_{free})

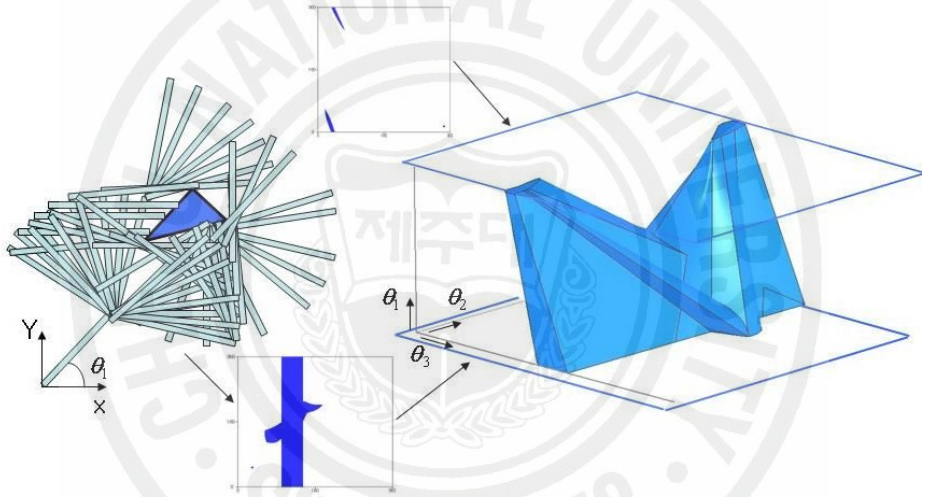


Fig. 16 3D Configuration Obstacle

3DOF 로봇팔의 C_{obs}는 Fig. 16 과 같이 첫 번째 링크를 제한 범위만큼 회전시켜가면서 각 각도에 따라 2차원 C_{obs} 영역을 서로 겹치면 3차원 C_{obs}를 구할 수 있다. 이렇게 구한 C_{obs}를 C에서 제외한 나머지 영역이 C_{free}롤서 이 범위내의 어떤 셈플도 장애물에 부딪힐 수 없다.

3. 표본추출(Sampling) 알고리즘 : Random Sampling

C_{free}는 헤아릴 수 없는 무궁한 영역이다. 그러므로 경로 생성을 위해선 먼저 C에서 표본(sample)들을 추출해야 한다. 표본추출 알고리즘에는 대표적으로 Low-Dispersion Sampling, Low-Discrepancy Sampling[14], Random Sampling[15] 등이 있는데, Low-Dispersion Sampling, Low-Discrepancy Sampling 알고리즘은 결정론적인 방법으로서 정확한 표본을 추출하기 위해 복잡한 수식을 사용하기 때문에 고차원 C에서는 계산량이 굉장히 많아져 시간이 많이 소비될 수 있다. 그래서 기본적인 RRT 알고리즘에서는 복잡한 수식이 필요 없이 간단히 표본을 추출 할 수 있는 Random Sampling 알고리즘이 사용되었다. 그래서 이 방법은 고차원 C를 효과적으로 셈플링 할 수 있고, 구현이 쉬우며 실제 PRM에서도 이 방법을 사용하였다. 셈플을 구하는 방법은 식(25)에서 보는 바와 같이 0에서 1사이의 난수 r을 추출하고 각 조인트 각도의 범위를 곱하여 각 조인트의 임의의 각도 값을 구하고 저장한 다음 이 방법을 구하고자하는 셈플 수만큼 반복한다. 셈플 수와 각 조인트 각도의 범위는 초기에 입력을 할 수 있다.

$$\Theta_i = r \times (\Theta_{i\_max} - \Theta_{i\_min}) \quad (25)$$

4. Nearest Neighbor Function : k-d tree

RRT를 형성하기 위해서는 q_{init}에서부터 반복적으로 근접한 nearest neighbor 노드 q 를 선택해나가면서 트리를 연결해나가야 한다. nearest neighbor 노드를 선택하기 위해서 k-d tree[16]을 사용하였다. 여기서 k는 트리를 가지고 표현 할 수 있는 공간의 차원을 나타낸다. k-d tree는 트리가

분기될 때 분기 되는 방향을 결정하기 위해 이용되는 속성이 트리의 깊이에 따라 다른 이진 탐색 트리이다. 예를 들어 2-d tree의 경우 x좌표에 의해서 루트와 짝수 깊이에서 비교하고 y좌표를 홀수 깊이에서 비교하는 데 이용한다. Fig.17에서 보는 바와 같이 A는 k-d tree의 루트노드이며 A의 x좌표 35보다 작은 x좌표를 갖는 포인터는 A의 왼쪽의 하위트리를 구성하고 A의 x좌표보다 크거나 같은 값은 A의 오른쪽 하위트리를 구성한다.

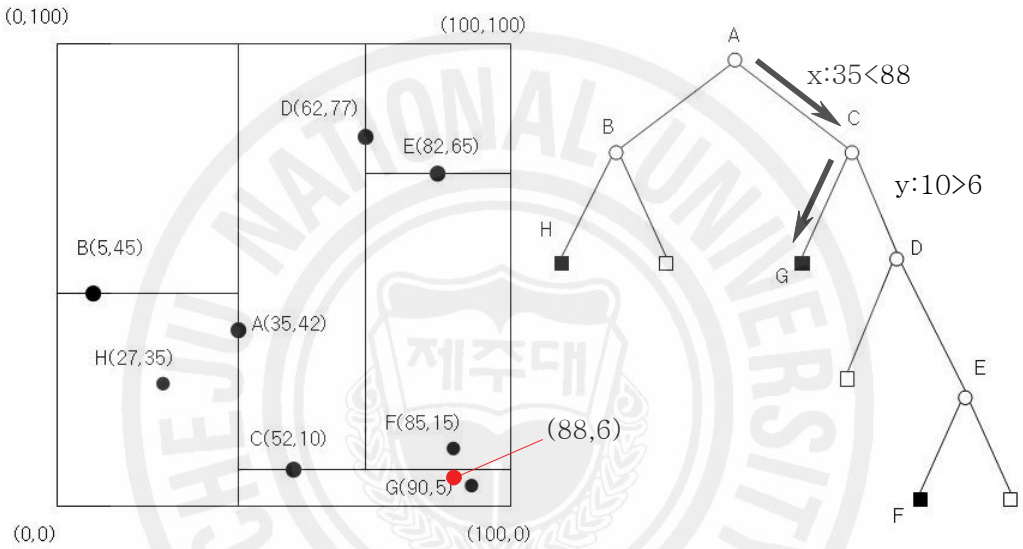


Fig. 17 k-d tree

k-d tree에서의 검색은 검색 영역 내에 존재하는 모든 포인터를 찾아서 반환한다. 이때 검색 영역은 원으로 나타내어지고 검색조건으로는 원의 중심점과 반지름을 파라미터로 받는다. 예를 들어, 한 점 (a,b)와 거리 d에 의해서 만들어진 원내에 존재하는 모든 포인터를 검색할 때, 찾고자 하는 모든 노드들을 (x,y)라 하면 이러한 노드들은 (a,b)로 부터의 거리가 d보다 작거나 같은 거리 안에 존재한다. 즉, $d2 \geq (a-x)2 + (b-y)2$ 의 조건을 만족한다. 그러므로 모든 X 는 $a-d < X < a+d$ 보다 작은 범위에 속하고 Y 또한 $b-d < Y < b+d$ 에 속한다. 이러한 조건을 이용하여 검색 할 때는 우선 다음

과 같은 pruning 을 먼저 실시하여 검색범위를 좁힌 후에 검색 조건에 적합한 노드를 찾는다. 검색하려는 k-d 트리상의 노드를 (e,f)라 하면 다음을 검사한다. KD_COMPARE((a-d,b-d),(e,f)) = 'RIGHT' 이면 (e,f) 노드의 왼쪽 하위 트리를 검색할 필요가 없고, KD_COMPARE((a+d,b+d),(e,f)) = 'LEFT' 이면 (e,f) 노드의 오른쪽 하위 트리를 검색할 필요가 없다. 가령 중심점이 (88,6) 이고 거리가 3인 지역에 포함되는 모든 점을 검색 하려면, A의 x좌표가 33<88 이므로 A의 왼쪽 하위트리를 검색 할 필요가 없다. 또한 C의 y좌표가 10>6 이므로 C의 오른쪽 하위트리를 검색할 필요가 없다.

Ⅴ. RRT 기반의 경로 생성 알고리즘

1. 기본적인 RRT 알고리즘

RRT 알고리즘은 1998년 Iowa university에서 Steven M. Lavalle교수에 의해 개발 되었고, 경로 생성 문제에 굉장히 폭넓게 사용하기 위해 임의의 데이터 구조체(randomized data structure)라는 개념으로 시작되어 지금까지 RRT를 바탕으로 개선된 알고리즘이 많이 개발 되고 있다. 이 알고리즘은 장애물에서 발생하는 algebraic constraints와 nonholonomic, dynamic system에서 발생하는 differential constraints를 가지고 있는 고차원 공간을 빨리 검색하기위한 효과적인 데이터 구조체와 셈플링 스킴을 포함한다. 그리고 구성공간이 고차원인 다자유도 시스템에 잘 적합 하는 장점이 있다.

로봇 팔의 경로 생성은 일반적으로 C_{free}에서 3D공간상에서 기하학적인 로봇팔과 같은 형태의 위치나 방향을 성분으로 하는 q의 가능한 연결을 찾는 것이다. 이러한 단일 경로 생성 작업은 어떤 전처리의 과정을 거치지 않고 초기 구성 상태(q_{init})에서 목표 구성 상태(q_{goal})으로 계속해서 경로를 계산하는 것을 말한다.

RRT의 주 관점은 탐색된 부분에서 탐색되지 않은 부분으로 뻗어 나간다는 것이다. 기본 RRT 알고리즘은 Fig.18 에 나타나 있다. 먼저 RRT를 적용하기 위해선 C가 정의되어야 하고, C_{free}가 결정되어야 한다. 그리고 Random Sampling 방식을 사용하여 가능한 q를 선정해야 한다. 그 다음 q_{init}와 q_{goal}을 입력하면서 RRT 알고리즘이 적용된다. 트리는 q_{init}에서부터 시작하면서 점점 확장해나가면서 q_{goal}에 다다를 때까지 EXTEND 함수를 반복하며 경로를 생성한다.

```
BUILD_RRT(q_init)
1. τ.init(q_init);
2. for k = 1 to K do
3.   q_rand ← RANDOM_CONFIG;
4.   EXTEND(τ, q_rand);
5. return τ;
```

```
EXTEND(τ)
1. q_near ← NEAREST_NEIGHBOR(q,τ);
2. if NEW_CONFIG(q, q_near, q_new) then
3.   τ. add _vertex(q_new);
4.   τ. add _edge(q_near, q_new);
5.   if q_near = q then
6.     return Reached;
7.   else
8.     return Advanced;
9. return Trapped;
```

Fig. 18 RRT(Rapidly-exploring Random Tree) Algorithm

Fig.19 는 EXTEND 함수를 그림으로 나타낸 것이다. EXTEND 함수는 주어진 q를 기준으로 k-d tree를 사용하여 nearest neighbor를 선택한다. NEW_CONFIG는 일정한 증분 ε의 간격으로 q쪽으로 경로가 만들어진다. 여기서 세 가지 경우가 발생 할 수 있다.

Reached는 q에서부터 ε안에 q_{new} 가 포함되어 있다는 경우이고, Advanced 는 $q_{new} \neq q$인 q_{new}가 RRT에 더해졌다는 경우이며 Trapped는 추가하려던 q_{new}가 C_{free}에 포함되지 않기 때문에 제거된 경우 이다. 그러므로 최종으로 Reached가 리턴이 되면 q에 대한 트리가 만들어지고 그것이 경로를 형성 하게 된다.

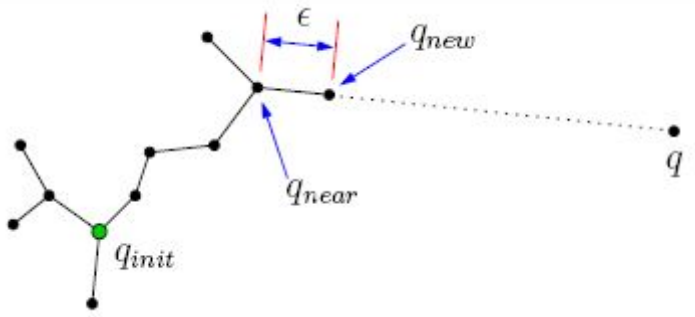


Fig. 19 EXTEND function

확장을 위해서는 Voronoi 영역을 구하고 그 전의 셈플에서 Voronoi 면적이 큰 방향으로 있는 다음 셈플로 확장하게 된다. Voronoi regions를 구하기 위해선 먼저 Voronoi 다이어그램을 구해야 한다. 이 방식은 각각의 셈플이 있으면 그 셈플이 하나만 포함되도록 영역을 분할하는 방식인데, 분할된 영역의 임의의 점이 다른 영역의 기준점에서의 거리보다도, 자신이 속한 영역의 기준까지의 거리가 제일 가깝게 되도록 분할을 한다. Voronoi 다이어그램에서 한 변을 공유하는 영역의 기준점기리 선을 이으면 Delaunay Triangulation이 된다. Delaunay Triangulation은 세 셈플로 이루어진 삼각형의 변을 따라가면서 그 삼각형의 외접원의 중심과 변을 공유하는 삼각형의 외접의 중심을 이으면 Voronoi 다이어그램을 얻을 수 있다. 이 경우 변이 다른 삼각형에 공유되지 않는 경우에는 변의 중심에서 바깥쪽으로 수선을 따라 멀리 있는 점을 잡고, 그 삼각형의 외접원의 중심과 선분을 그으면 convex hull 바깥 영역을 분할한다. Fig. 20 은 이런 방식으로 Voronoi 다이어그램을 표현한 것이다.

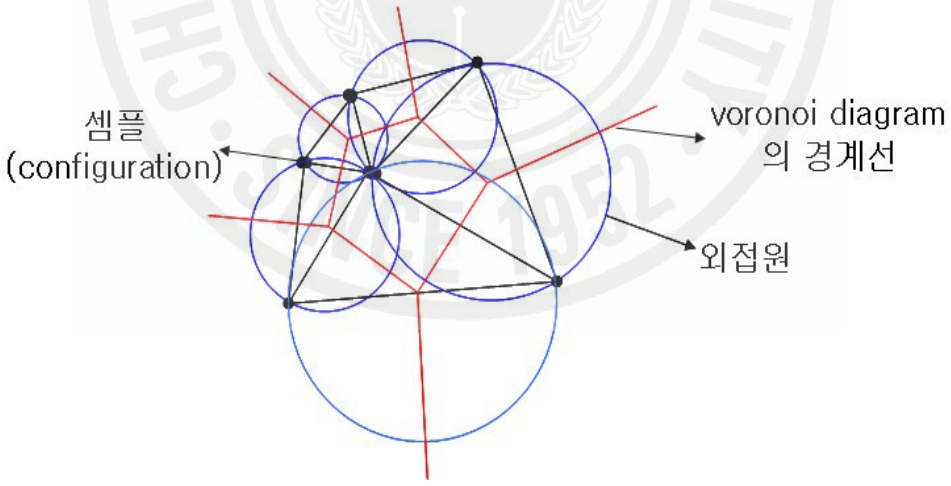


Fig. 20 Voronoi Diagram

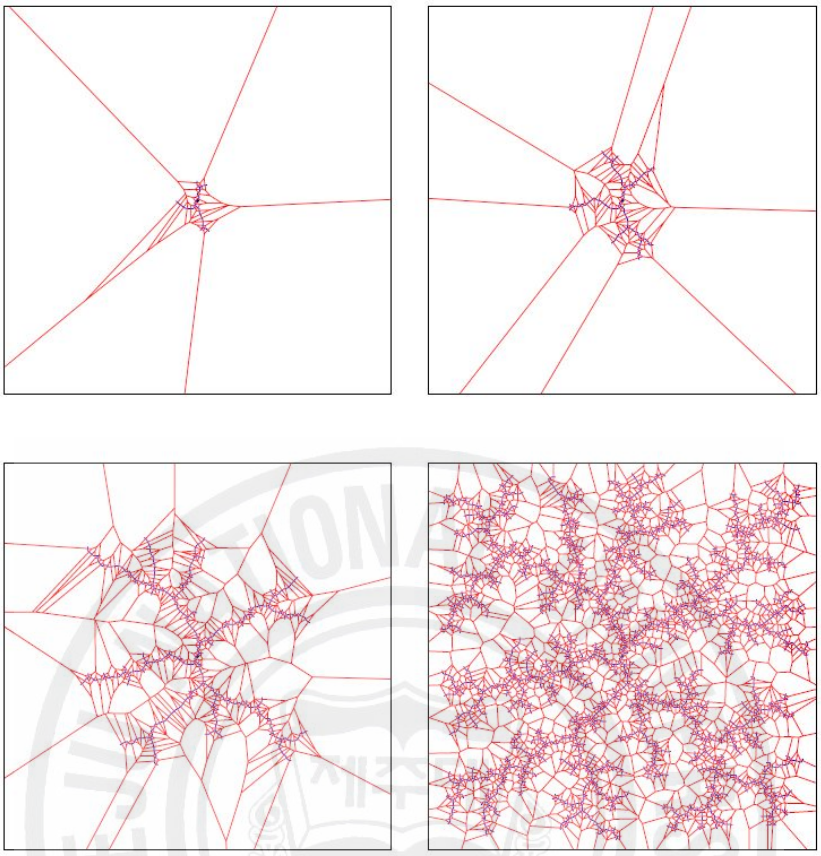


Fig. 21 Voronoi Diagram Implementation

Fig. 21은 RRT에 Voronoi Diagram을 적용한 예이다. 이 그림은 (50,50)에서부터 각 모서리 까지 RRT를 실행한 예제로서 빨간 선은 Voronoi 경계선을 나타내고 중간에서부터 파란 선의 경로가 뻗어 나가는 것을 볼 수 있다.

2. RRT-Connect 알고리즘

Fig. 22 에서 보는 바와 같이 RRT-Connect[10] 알고리즘 기본은 RRT 알고리즘을 바탕으로 하였으나 특이하게 differential constraint가 없는 경로 생성

문제에 적합한 알고리즘이다. RRT-Connect 는 i) 더 긴 거리로 나가려고 시도하는 Connect heuristic 과 ii) RRT가 q_{init} 에서부터만 시작하는 것과는 달리 q_{init}와 q_{goal} 양쪽에서 RRT를 형성시켜 나가는 것 두 가지에 중점을 두고 있다. 이 알고리즘은 q_{init} 과 q_{goal}이 주어지면 이 두 노드로부터 RRT가 만들어져 어느 지점에서 두 트리의 마지막 노드가 만나거나 일정 간격사이에 들어오게 되면 두 트리를 이어 경로를 생성하는 방법이다. 그러므로 이 알고리즘은 트리가 동시에 뻗어나가므로 기본 RRT 보다는 계산 시간을 단축시킬 수 있다. Fig.23 에서는 두 트리가 뻗어 나가는 과정을 나타낸다.

```
CONNECT(T, q)
 1  repeat
 2      S ← EXTEND(T, q);
 3  until not (S = Advanced)
 4  Return S;
```

```
RRT_CONNECT_PLANNER(q_init, q_goal)
 1  T_a.init(q_init); T_b.init(q_goal);
 2  for k = 1 to K do
 3      q_rand ← RANDOM_CONFIG();
 4      if not (EXTEND(T_a, q_rand) = Trapped) then
 5          if (CONNECT(T_b, q_new) = Reached) then
 6              Return PATH(T_a, T_b);
 7      SWAP(T_a, T_b);
 8  Return Failure
```

Fig. 22 The RRT-Connect algorithm

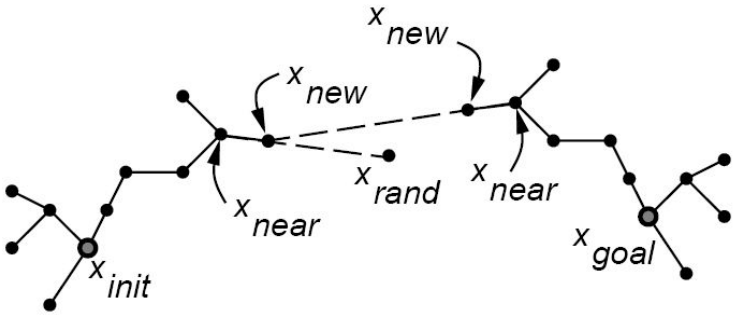


Fig. 23 EXTEND of RRT-Connect

3. Goal-biased RRT 알고리즘

기본 RRT 알고리즘의 random sampling 방법은 선택되는 q_{rand} 가 q_{goal} 을 향하지 않고 전체적으로 퍼져나간다. 그래서 비교적 비효율적이다. 이것을 보완한 알고리즘이 Goal-biased RRT 알고리즘인데 이 방법은 RRT의 random sampling 방법을 수정하여 q_{rand} 선택시 q_{goal}로의 방향성을 부가하여 셈플링이 q_{goal} 쪽으로 향하도록 되므로 RRT 보다는 경로 생성에 불필요한 계산이 없이 좀 더 효율적이 되도록 하였다. Fig.24 는 Goal-biased RRT 알고리즘을 나타낸 것이다.

```
BUILD_RRT(x_init)
1   T.init(x_init);
2   for k = 1 to K do
3       x_rand ← RANDOM_STATE();
4       EXTEND(T, x_rand);
5   Return T

BIASED_RANDOM_STATE()
1. toss ← COIN_TOSS();
2. if toss = heads then
3.    return q_goal ;
4. else
5.    return RANDOM_STATE();
```

Fig. 24 Goal-biased RRT algorithm

4. RC-RRT(Resolution Completeness-RRT) 알고리즘

RC-RRT(Resolution Completeness-RRT)[17] 알고리즘은 이전의 Goal-biased RRT 알고리즘의 기능을 포함하면서 기본적인 RRT 알고리즘의 NEAREST NEIGHBOR 함수를 Fig. 25(a)에서 Fig.25(b)로 수정한 알고리즘이다. 분해 완비성(Resolution Completeness)의 의미는 플래너가 반복적으로 다음 경로에 대한 노드를 찾을 때 제한된 반복 횟수 내에서 해법을 찾는다는 것이다. 이것은 기존의 Randomized 알고리즘이 가지고 있는 충분한 반복 횟수로 같은 확률을 가지고 있는 노드들을 사용하여 한 가지 해에 수렴해 간다는 뜻의 확률적 완비성(Probabilistically Completeness)[18]과는 틀린 성질이다. Fig.25(b) 의 $\sigma(x)$는 CVF(Constraint Violation Frequency)로 k-d tree에 검색이 중복될 가능성이 있는 노드는 배재하고 검색이 안된 노드들을 중심으로 검색하므로서 중복 검색을 줄이기 위해서 쓰여졌다.

$$\sigma(x) = \frac{Number\ of\ box\ leaves\ in\ Explored\ Region}{Number\ of\ all\ the\ leaves}$$

```
NEAREST_NEIGHBOR(x_random, G)
1  d_min ← ∞
2  for all x in G
3      d ← ρ(x, x_random);
4      if d < d_min
5          d_min ← d;
6          x_best ← x;
7  return n_best
```

```
NEAREST_NEIGHBOR(x_random, G)
1   d_min ← ∞;
2   d_min' ← ∞;
3   for all x in N_G
4       if ∃u of x are not EXPANDED
5           d ← ρ(x, x_random);
6           if d < d_min'
7               d_min' ← d;
8               x_best' ← x;
9           r ← random number in [0,1];
10          if r > σ(x)
11              if d < d_min
12                  d_min ← d;
13                  x_best ← x;
14  if d_min ≠ ∞
15      return x_best;
16  else
17      return x_best';
```

(a) previous Nearest Neighbor (b) Nearest Neighbor in RC-RRT

Fig. 25 RC-RRT Algorithm

Ⅵ. 시뮬레이션 및 실험 결과

이 장에서는 앞에서 언급했던 PRM, RRT, Advanced-RRT 알고리즘을 먼저 6DOF 로봇팔에 적용을 시켜보고, 그 다음 13DOF 로봇 상체에 적용을 시켜 경로 생성시 소비되는 시간과 노드(q)의 수를 계산하여 알고리즘의 효율성을 판단하였다. Fig.25에서는 시뮬레이션의 소프트웨어 구조를 나타낸다. 이 시뮬레이션은 리눅스 환경에서 개발되었고, 그래픽은 opengl 라이브러리를 사용하였다. 최종 경로는 각 노드의 요소들을 기구학적으로 계산하여 각 조인트가 움직이는 경로로 나타내어 기하학적으로 보기 쉽게 나타내었다.

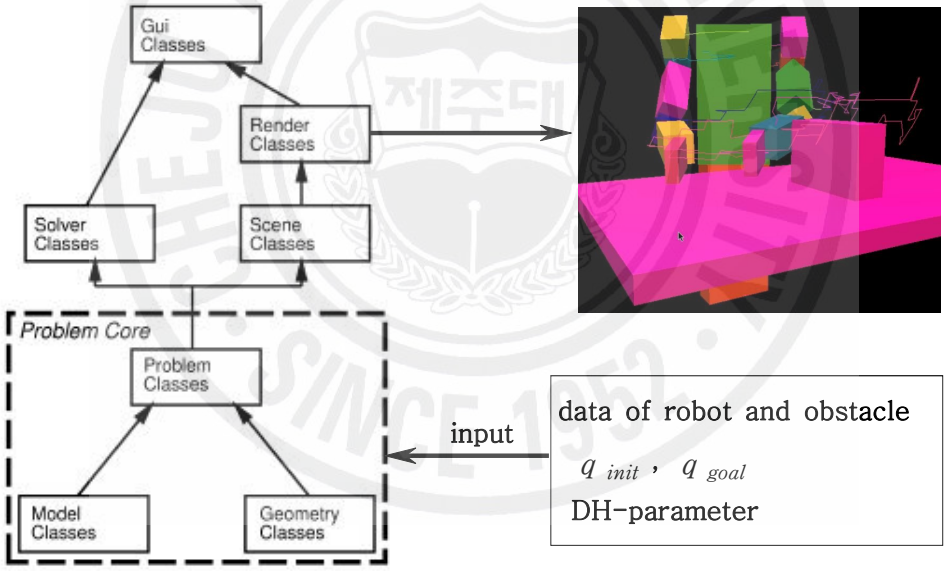
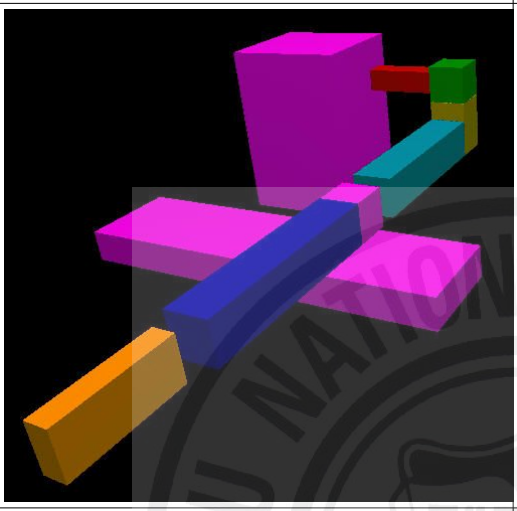
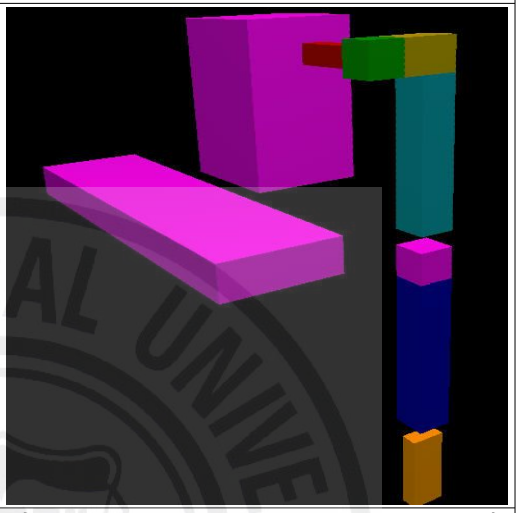


Fig. 25 Architecture of Simulation

다음은 시뮬레이션 결과를 각 알고리즘을 적용하여 나타낸 자료이다. 알고리즘은 PRM, RRT, RRT-Connect, Goal Biased-RRT, RCRRT를 사용하였고, 먼저 6DOF 로봇팔을 적용하였고, 그 다음으로 13DOF 로봇 상체를 사용하였다. 장애물로는 책상을 가정한 직육면체를 하였다.

1. 6DOF 로봇 팔 시뮬레이션

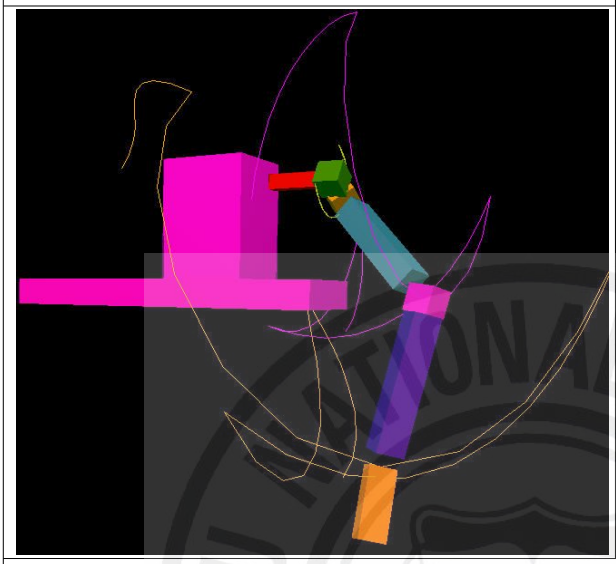
초기 자세	목표 자세
	
상태(q)=(0,1.57,1.57,0,-1.57,-1.57)	상 태 (q)=(-1.57,1.57,1.57,0,-1.57,-1.57)

장애물	책상이라 가정한 몸체 앞의 직육면체
목표동작	팔을 앞으로 뻗은 상태에서 장애물을 피하고 밑으로 내리기
측정치	경로 생성 노드 수, 경로계획시간, 생성된 경로의 사진
적용알고리즘	RRT, PRM, RRT-Connect, Goal-biased RRT, RCRRT

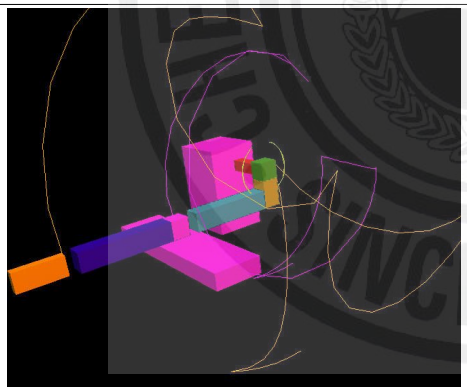
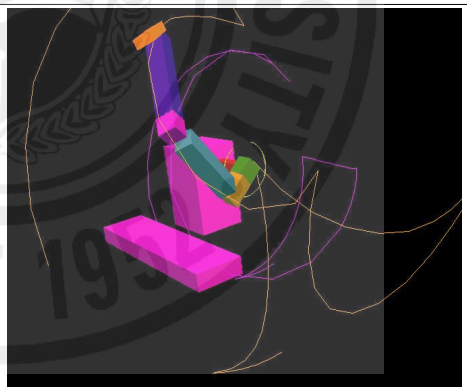
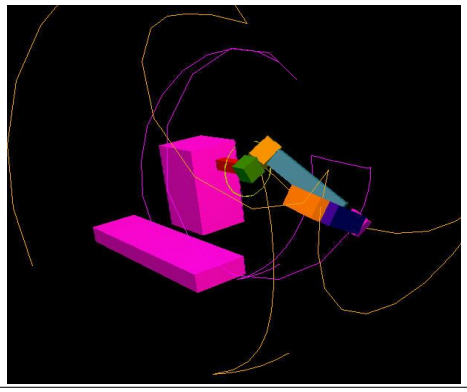
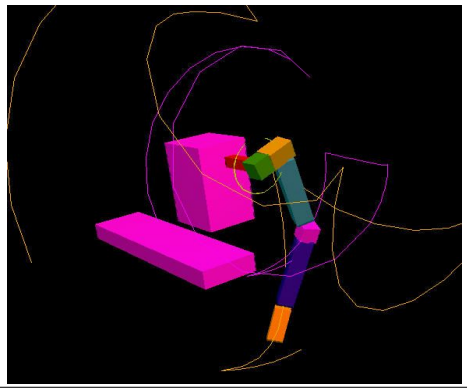
<RRT 알고리즘 적용>

RRT의 적용 결과 : 경로 생성 실패

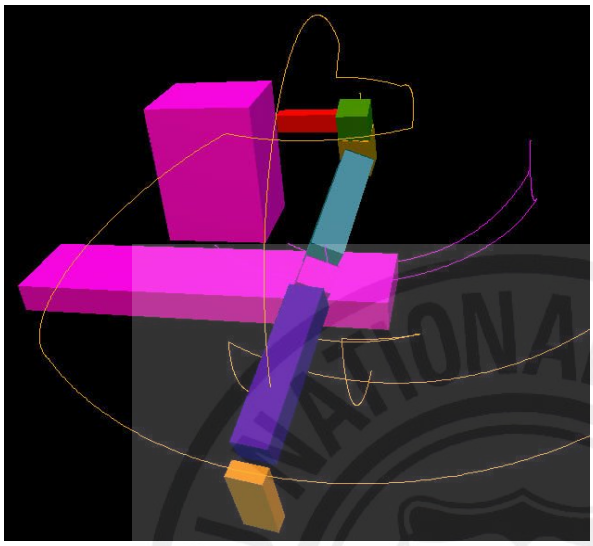
<PRM 알고리즘 적용>

경로 생성 결과	측 정 치	
	로드맵상의 노드수	656
	로드맵상의 엣지수	1388
	로드맵 구축시간	82.98s
	경로 계획 시간	0.08s

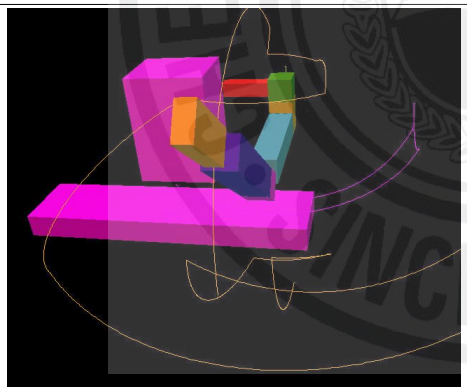
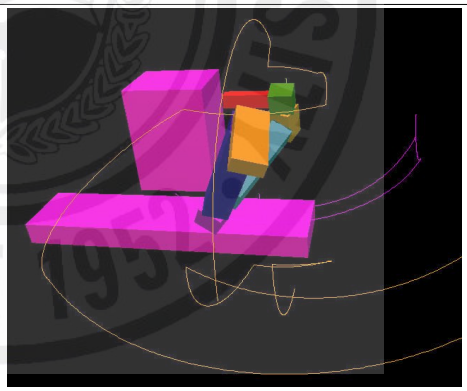
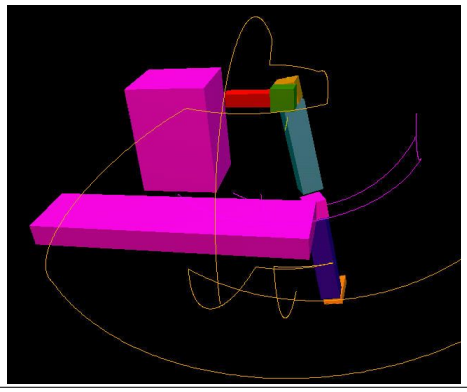
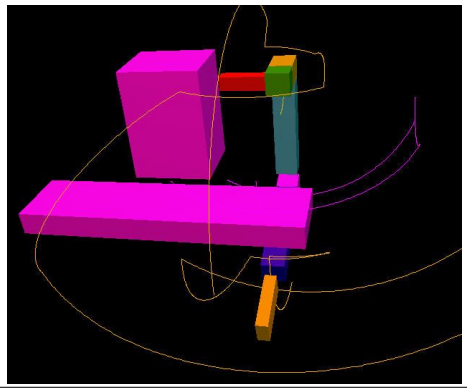
동작 시현

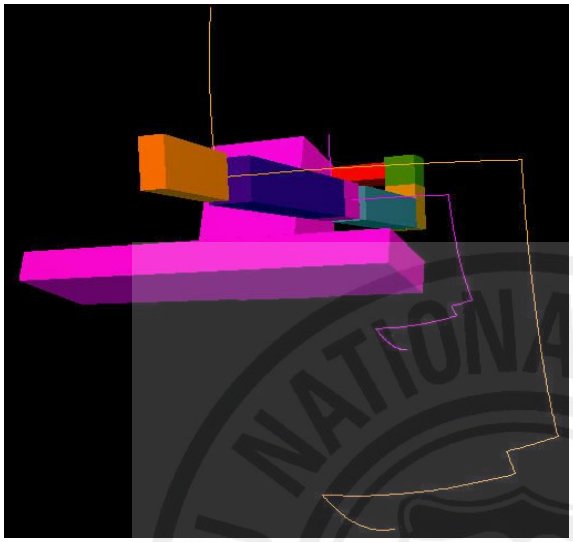
<RRT-Connect 알고리즘 적용>

경로 생성 결과	측 정 치	
	경로 구성 노드수	264
	경로 계획 시간	0.81s

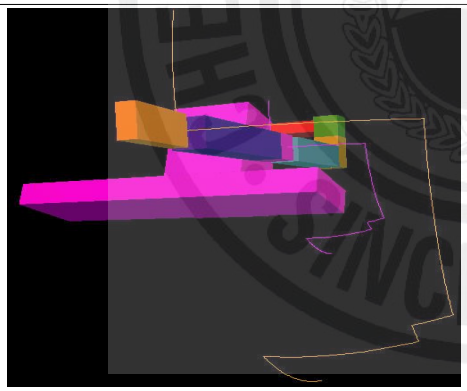
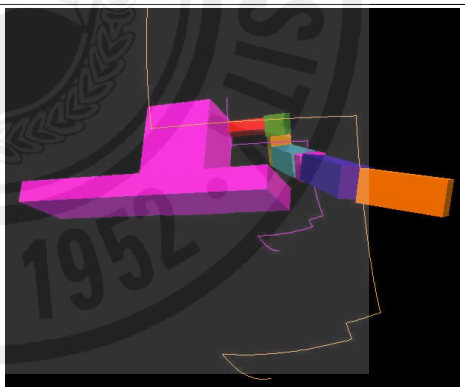
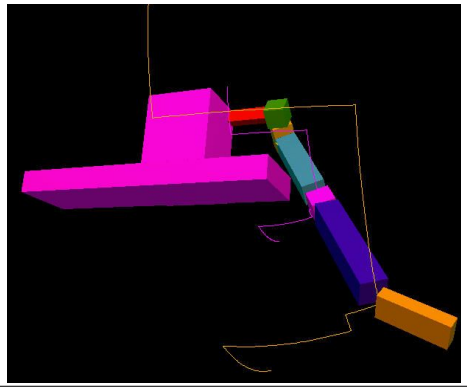
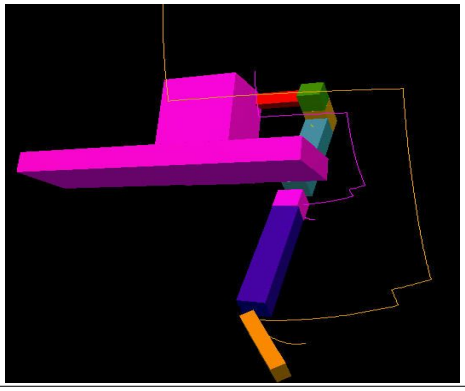
동작 시현

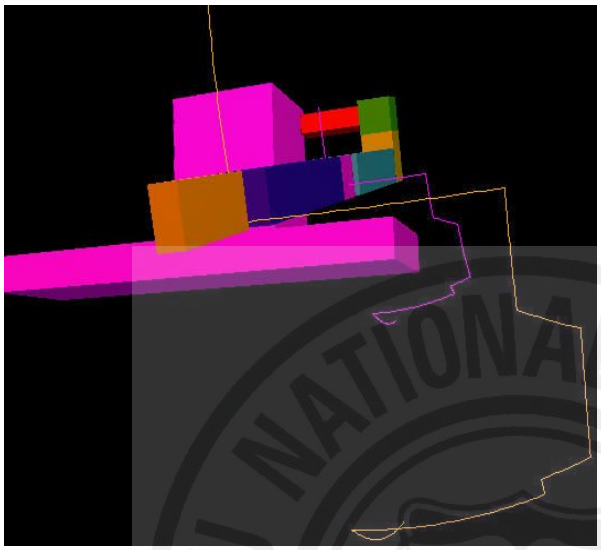
<Goal-biased RRT 알고리즘 적용>

경로 생성 결과	측 정 치	
	경로 구성 노드수	484
	경로 계획 시간	0.469

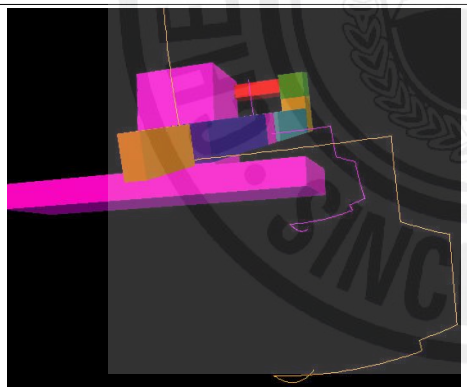
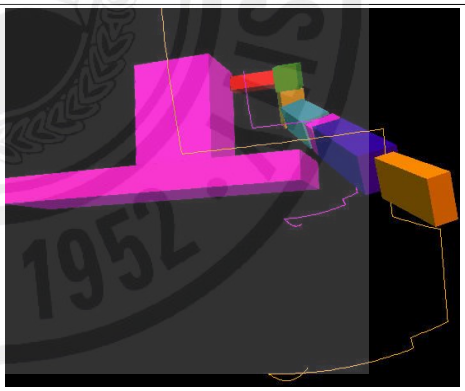
동작 시현

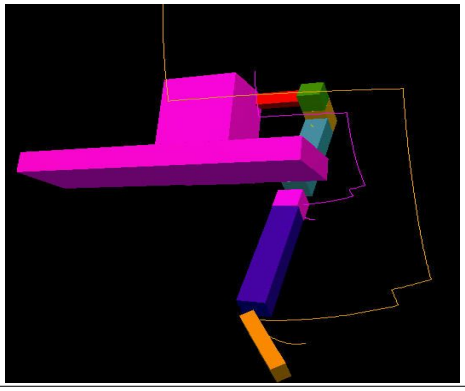





<RC-RRT 알고리즘 적용>

경로 생성 결과	측 정 치	
	경로 구성 노드수	156
	경로 계획 시간	0.395

동작 시현



<시뮬레이션 결과>

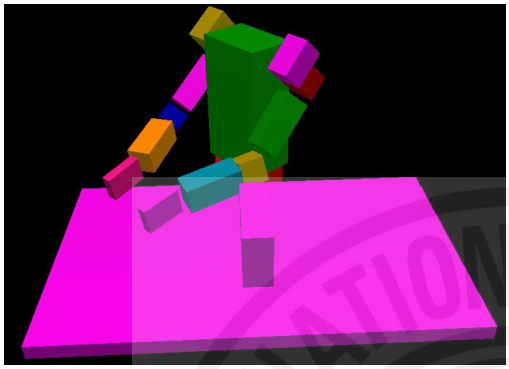
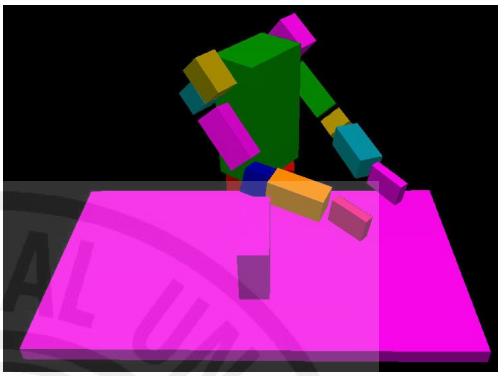
처음에는 비교적 자유도가 적은 6DOF 휴머노이드 로봇의 한 팔에 대하여 여러 가지 알고리즘을 적용시켜보았다. RRT를 2D상에서 모바일로봇에 적용할 때는 경로 생성이 성공 하는 것을 확인 하였으나 본 연구의 목적인 로봇 팔에 대한 적용성을 판단해야 하므로 먼저 6DOF 한쪽 팔에 RRT를 적용한 경우에는 경로 생성에 실패하였다. 이 결과 기본적인 RRT는 자유도가 높은 로봇 팔에는 적용가능성이 굉장히 적다고 판단된다.

한쪽 로봇 팔에 PRM을 적용시켰을 때는 경로 계획 시간은 다른 알고리즘보다 현저히 작다. 그러나 로드맵 구축시간이 굉장히 많이 소비 되므로 장애물의 위치가 바뀔 경우마다 로드맵을 구축해야 하므로 총 경로 생성 시간이 굉장히 오래 걸린다. 그래서 이 알고리즘은 장애물의 위치가 바뀌지 않고 초기 상태와 목표 상태만 바뀔 경우에만 효율적이라고 판단된다. 그리고 각 조인트마다 생성된 경로를 보면 불필요한 동작이 생성되어 로봇 팔 적용에는 비교적 비효율적으로 판단된다.

RRT-Connect를 적용 시켰을 경우 PRM보다는 경로 생성이 비교적 효율적으로 되었다. 그리고 Goal-biased RRT를 적용했을 때 보다는 경로 계획 시간이나 경로를 구성하는 노드 수는 적다. 그러나 생성된 경로를 보면 RRT-Connect를 적용 시켰을 때도 Goal-biased RRT나 RC-RRT와 비교했을 때 보다는 불필요한 동작이 생성된 것을 확인 할 수 있다.

Goal-biased RRT를 적용 시켰을 때는 생성된 경로만을 놓고 보면 RC-RRT를 적용 시켰을 때와 거의 비슷하게 나타난다. 그러나 경로를 구성하는 노드수와 경로 계획 시간을 보면 RC-RRT를 적용 시켰을 때가 굉장히 효율적이라는 것을 알 수 있다. 이 실험결과 6DOF 한쪽 로봇 팔의 경로 생성을 위해선 RC-RRT가 제일 효율적이라고 판단된다.

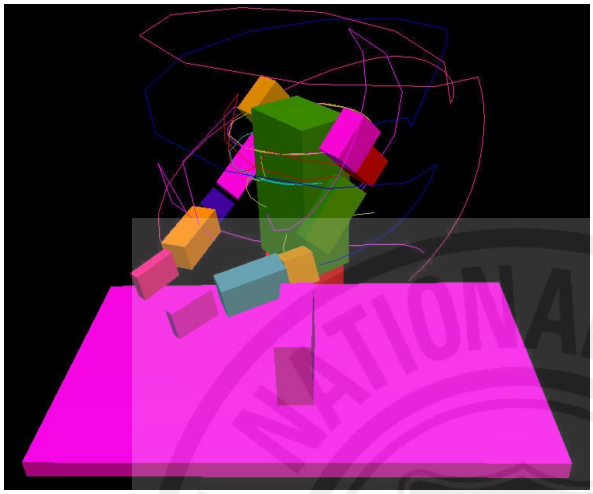
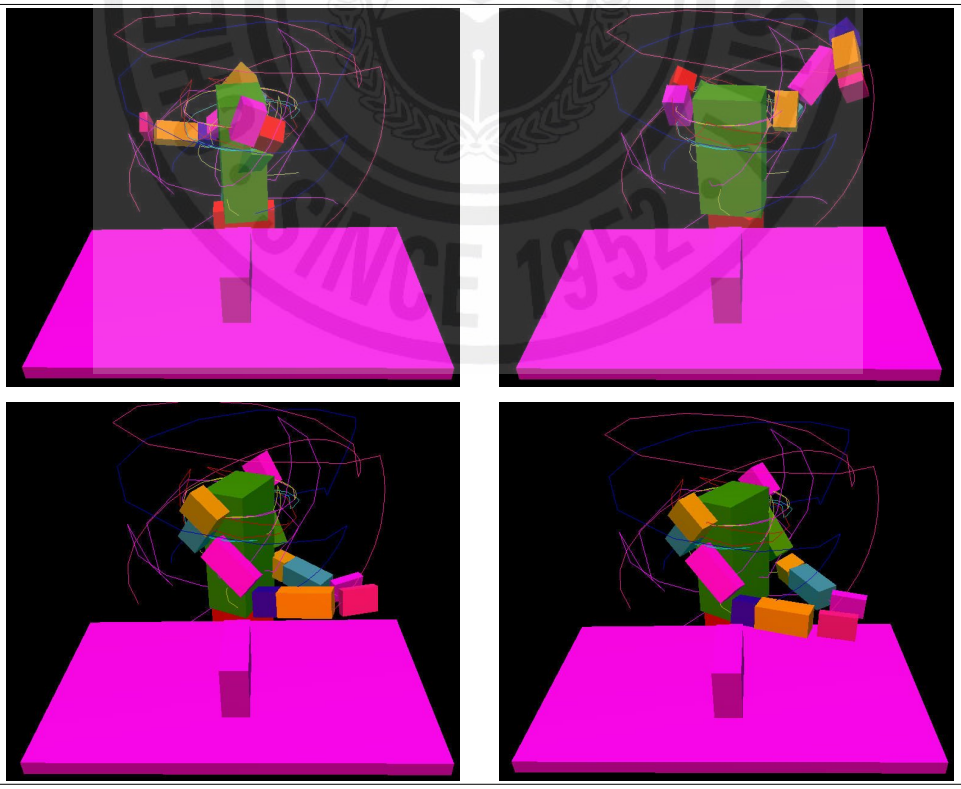
2. 13DOF 로봇 상체 시뮬레이션

<table>
<tr><th>초기 자세</th><th>목표 자세</th></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>상태(q)=(-0.9, -0.7, 1.57, 0.7, 0.7, -0.9, -0.9, -0.8, 1.57, 2.0, 0.7, -1.9, -1.25)</td><td>상태(q)=(0.9, -0.8, 1.57, 1.14, 0.7, -1.24, -1.9, -0.7, 1.57, 2.44, 0.7, -2.24, -1.25)</td></tr>
<tr><td>장애물</td><td colspan="2">책상이라 가정한 몸체 앞의 직육면체와 책상 위 가운데 박스를 가정한 직육면체</td></tr>
<tr><td>목표동작</td><td colspan="2">양 팔을 책상 위 박스의 오른쪽에서 박스를 회피하여 왼쪽으로 움직이는 동작</td></tr>
<tr><td>측정치</td><td colspan="2">경로 생성 노드 수, 경로계획시간, 생성된 경로의 사진</td></tr>
<tr><td>적용알고리즘</td><td colspan="2">RRT, PRM, RRT-Connect, Goal-biased RRT, RC-RRT</td></tr>
</table>

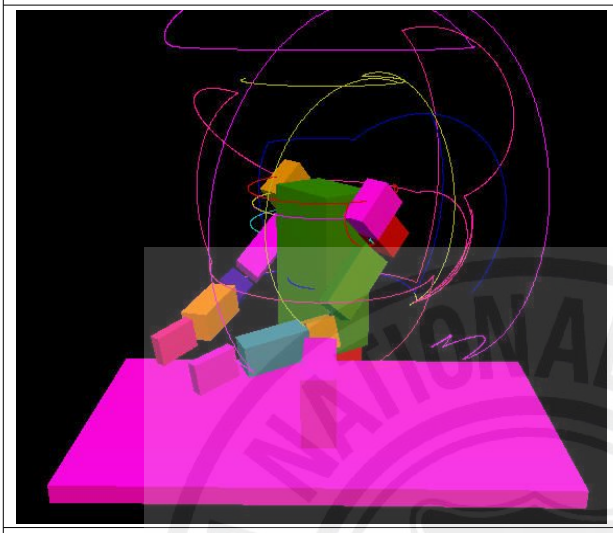
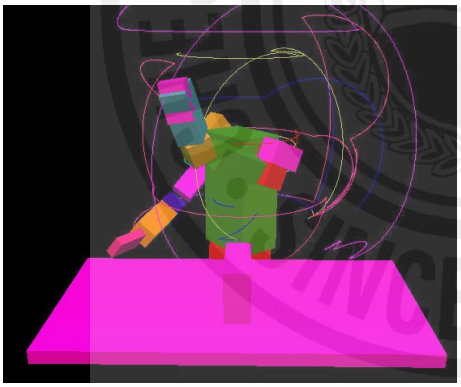
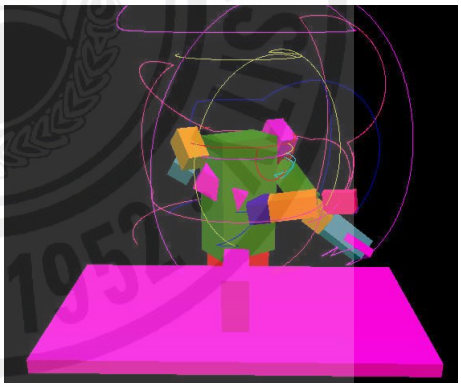
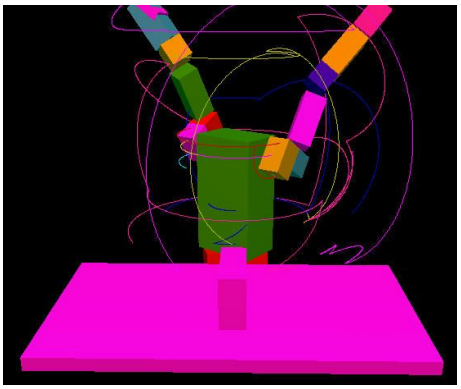
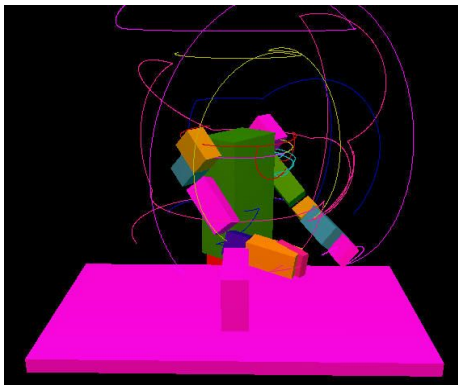
<RRT 알고리즘 적용>

RRT의 적용 결과 : 경로 생성 실패

<PRM 알고리즘 적용>

<table>
<tr><th>경로 생성 결과</th><th colspan="2">측 정 치</th></tr>
<tr><td rowspan="4">  </td><td>로드맵상의
노드수</td><td>781</td></tr>
<tr><td>로드맵상의
엣지수</td><td>1560</td></tr>
<tr><td>로드맵 구축시간</td><td>91.98s</td></tr>
<tr><td>경로 계획 시간</td><td>0.08s</td></tr>
<tr><td colspan="3">동작 시현</td></tr>
<tr><td colspan="3">  </td></tr>
</table>

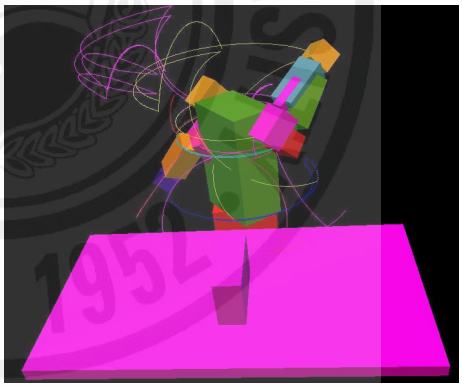
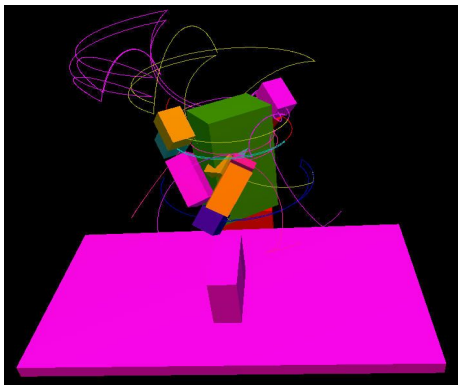
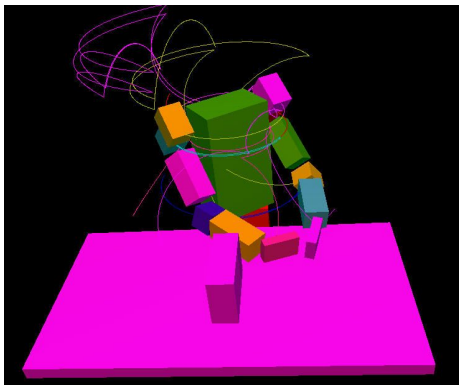
<RRT-Connect 알고리즘 적용>

<table>
<tr><th>경로 생성 결과</th><th colspan="2">측 정 치</th></tr>
<tr><td rowspan="4">  </td><td>경로 구성
노드수</td><td>782</td></tr>
<tr><td>경로 계획 시간</td><td>6.579</td></tr>
<tr><td></td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td colspan="3"></td></tr>
<tr><td colspan="3">동작 시현</td></tr>
<tr><td colspan="3">  
   </td></tr>
</table>

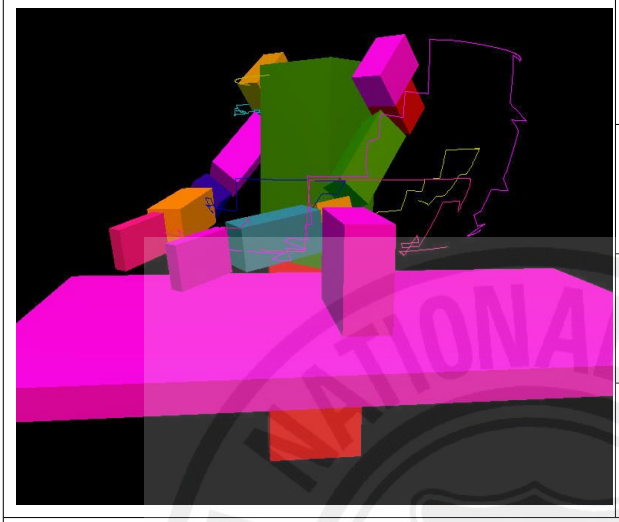
<Goal-biased RRT 알고리즘 적용>

경로 생성 결과	측 정 치	
	경로 구성 노드수	700
	경로 계획 시간	6.05

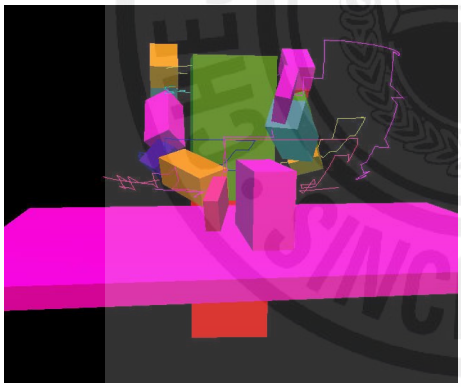
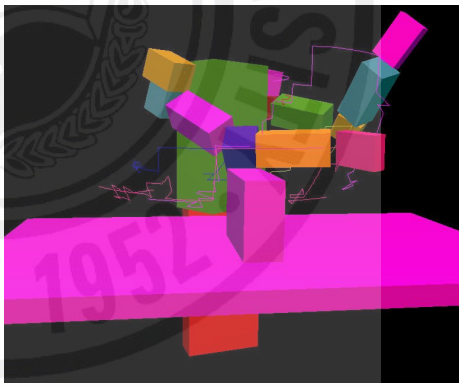
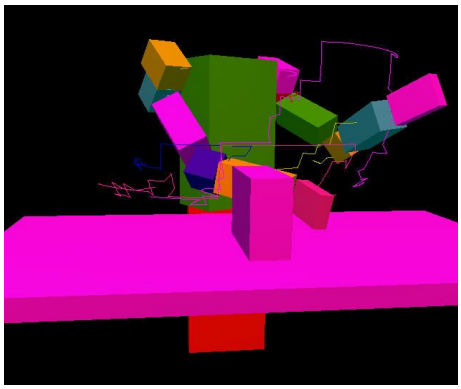
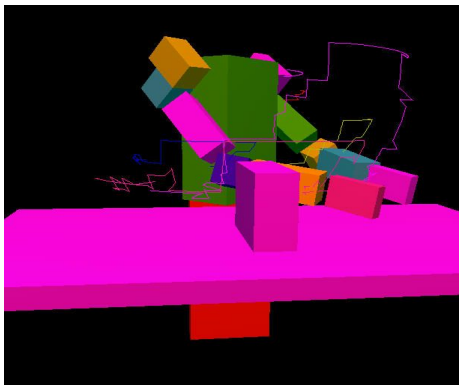
동작 시현

<RC-RRT 알고리즘 적용>

경로 생성 결과	측 정 치	
	경로 구성 노드수	3750
	경로 계획 시간	76.63s

동작 시현

<시뮬레이션 결과>

이번에는 자유도가 높은 13DOF 의 휴머노이드 로봇 상체에 대해서 여러 가지 알고리즘들을 적용 시켜 보았다. 이번에도 RRT를 적용했을 경우에는 경로 생성이 실패 하였다. 이 두 가지의 결과 기본적인 RRT 만으로는 휴머노이드 로봇 팔에 대한 적용이 불가능 하다는 것을 알 수 있다.

PRM을 6DOF 한쪽 팔만 적용 시켰을 때, 13DOF 로봇 상체에 적용 시켰을 때 둘 다 로드맵 구축시간은 비슷하게 나타났다. 그러나 13DOF 에 적용할 때 불필요한 경로가 더 많이 생성되어, 시뮬레이션 상에 불필요한 동작이 더 많이 나타났다. 그리고 로봇 링크 간에 충돌이 있어 이 알고리즘은 13DOF와 같은 고차원 시스템에는 적합하지 않다고 판단된다.

RRT-Connect 와 Goal-biased RRT를 적용 시켰을 때는 경로를 구성하는 노드 수나 경로 시간이 RC-RRT보다는 훨씬 적으나 이때도 불필요한 경로가 많이 생성되고 불필요한 움직임이 많이 생성된다. 그리고 Goal-biased RRT를 6DOF에 적용 시켰을 때 RC-RRT와 비슷한 경로를 생성하였으나 13DOF에 적용 시켰을 때는 RRT-Connect와 비슷한 결과를 가져와서, Goal-biased RRT는 DOF가 높을수록 적용이 어렵다는 것을 알게 되었다.

RC-RRT 의 경우는 경로 생성 시간도 많이 걸리고 경로 구성 노드수도 다른 알고리즘보다 훨씬 많다. 그러나 불필요한 동작도 거의 없는 편이고, 장애물에 대한 충돌이 없어 안정성이 훨씬 뛰어나다.

이와 같이 13DOF 휴머노이드 로봇 상체와 같은 고차원의 시스템에 대한 장애물 회피 경로를 효율적으로 생성하려면 계산량이 많아서 시간이 많이 걸릴 수밖에 없다.

Ⅶ. 결 론

본 논문에서는 기존 경로 생성 알고리즘의 분석을 통하여 양팔을 동작을 동시에 계획할 수 있는 13DOF 휴머노이드 로봇 상체에 대한 경로 생성에 적합한 알고리즘을 제안 하였고, 실제 로봇과 같은 기구학적인 특성과 같이 모델링된 시뮬레이션을 통하여 알고리즘의 효율성 및 안정성을 검증하였다. 이러한 연구를 통하여 다음과 같은 성과를 얻을 수 있었다.

휴머노이드 로봇 상체에 대한 기구학적 해석 : 13DOF 휴머노이드 로봇 상체의 하드웨어 특성에 맞는 기구학을 해석하였으며, 이를 바탕으로 여러 가지 알고리즘을 적용할 수 있었다.

기존 경로 생성 알고리즘 분석 및 연구 : 기존 경로 생성 알고리즘 및 요즘 세계적으로 많이 연구되고 있는 셈플링 기반의 알고리즘에 대하여 분석하고 장단점을 파악하여 고차원 시스템에 대한 적합성을 판단하였다.

RRT 기반의 경로 생성 알고리즘에 대한 시뮬레이션 적용 : PRM, RRT, RRT-Connect, Goal-biased RRT, RC-RRT를 실제 시뮬레이션에 적용시켜 봄으로서 각각의 알고리즘에 대한 효율성 및 안정성을 판단할 수 있었다. 결과적으로 RC-RRT 알고리즘이 생성된 경로로만 봤을 때는 가장 효율적이었으나, 시간이 비교적 많이 소비되는 경향이 있으므로 로봇 하드웨어의 속도 향상 및 더 보완된 알고리즘의 필요성을 느꼈다.

Ⅶ. 참고 문헌

[1] T. Asfour and R. Dillmann , 2003, "Human-like Motion of a Humanoid Robot Arm Based on a Closed-Form Solution of the Inverse Kinematics Problem", Proceedings of the 2003 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems, pp. 1407-1412

[2] 이윤배, 1993, "장애물 환경에서 무인반송차량의 최적경로 제어", 숭실대학교 박사학위논문

[3] Philip John Mckerrow, 1993, "Introduction to Robotics", Addison-Wesley Publishing, pp.438-472

[4] 조수정, 2007, "지능로봇을 위한 실시간 장애물 회피에 관한 연구" 제주대학교 석사학위 논문, pp.27-40

[5] L. E. Kavraki, P. Svestka, J, C. Latombe, and M. Overmars. 1996, "Probabilistic roadmaps for path planning in high dimensional configuration spaces", IEEE Transactions on Robotics and Automation, Vol. 12, No. 4, pp. 566-580

[6] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones and D. Valleju, 1998, "Choosing Good Distance Metrics and Local Planners for Probabilistic Roadmap Methods", IEEE International Conference on Robotics & Automation, Vol. 16, No. 4, pp. 442-447

[7] T. Simeon, 2004, "Manipulation Planning with Probabilistic Roadmaps", The International Journal of Robotics Research, Vol. 23, No. 7-8, pp. 729-746

[8] S. M. Lavalle, 1998, "Rapidly-exploring random trees: A new tool for path planning", Computer Science Dept. Iowa State Univ., TR 98-11

[9] S. M. Lavalle and J. Kuffner. 1999, "Randomized kinodynamic planning", Proc. of the IEEE Robotics and Automation Conference, pp. 473-479

[10] J. Kuffner and S. LaValle. 2000, "RRT-Connect: an efficient approach to single-query path planning," Proc. 2000 IEEE International Conf. on Robotics and Automation, vol.2, pp.995-1001

[11] E. Yoshida, 2005, "Humanoid motion planning using multi-level DOF exploitation based on randomized method" Intelligent Robots And Systems(IROS), pp.25-30

[12] J. Kuffner and K. Nishiwaki, 2005, "Motion Planning for Humanoid Robots" Springer Tracts in Advanced Robotics, 365-374

[13] H. Choset, K. M. Lynch, S. Hutchinson, 2005, "Principles of Robot Motion", Massachusetts Institute of Technology, 39-69,

[14] H. Niederreiter. 1992, Random Number Generation and Quasi-Monte-Carlo Methods. Society for Industrial and Applied Mathematics, Philadelphia,

[15] J. Barraquand, L. Kavraki, J.-C. Latombe, T.-Y. Li, R. Motwani, and P. Raghavan. 1996, "A random sampling scheme for robot path planning." In G. Giralt and G. Hirzinger, editors, Proceedings International Symposium on Robotics Research, pp 249-264.

[16] M. H. Overmars and J. van Leeuwen. 1982, "Dynamic multidimensional data structures based on Quad- and K-D trees", Acta Informatica, vol.17, pp.267-285

[17] Peng Cheng and S. M. LaValle 2002, "Resolution Complete Rapidly-Exploring Random Trees", IEEE International Conference on Robotics and Automation, Vol.1, pp. 267-272

[18] L. Kavraki, J. C. Latombe, R. Motwani, and P. Raghavan. 1995, "Randomized query processing in robot motion planning." In Proc. ACM SIGACT Symp. on Theory of Computing(STOC), pp 353-362

감사의 글

어느덧 4년 반이라는 긴 연구실 생활이 끝났습니다. 2학년 복학하고서 대학원 생활에 이르기까지 연구실은 저의 전부였습니다. 이렇게 마치려고 하니 시원섭섭합니다. 그 동안 연구실에서 우여곡절도 많았는데, 제가 이제까지 버틸 수 있었던 것은 많은 이의 가르침과 도움이 있었기에 가능한 일 이었습니다. 부족하지만 그 분 들에게 글로나마 감사의 마음을 전하고자 합니다.

먼저 긴 연구실생활동안 턱 없이 부족한 저를 받아주시고 끝까지 믿어주시고 아낌없이 가르침과 지원을 해주신 최경현 교수님께 진심으로 감사의 말씀 전하고 싶습니다. 그 가르침을 바탕으로 이제 막 시작하게 될 사회생활에서도 열심히 살아가고자 합니다. 그리고 대학원 2년 동안 프로젝트를 같이 하였던 수학과 유상욱 교수님께도 진심으로 감사를 드립니다. 부족한 저를 그래도 잘 하고 있다고 칭찬해주시고 항상 자신감을 불어넣어주시고 개인적으로 항상 편하게 대해주셨던 그 은혜 잊지않겠습니다. 그리고 학부 때부터 여러 가르침을 주셨던 조경호 교수님, 임종환 교수님, 강철웅 교수님, 김상재 교수님 진심으로 감사를 드립니다. 앞으로도 우리 과를 잘 이끌어나가 주시기를 부탁드리겠습니다.

교수님들 외에도 저에게 많은 도움을 주었던 선배, 동료, 후배들이 있습니다. 이 자리를 빌어서 감사하다는 말 전해 주고 싶습니다. 먼저 연구실에서 저와 같이 가장오래 생활을 했고 여자임에도 불구하고 남학생들보다도 더 고생하고 참아내서 실험실을 이끌어주었던 조수정에게 진심으로 감사하다는 말 전하고 싶습니다. 그리고 여러 가지로 저한테 많은 도움을 주고 있고 연구실 유일한 동료 양형찬, 실험실 맏형 배진이 형, 부족한 저를 잘 따라주었고 실험실의 궂은 일을 맡아서 열심히 했고 항상 웃음을 잃지 않았던 황영봉, 고광표, 박의장, 문지현 이들이 있어서 실험실생활을 재밌게 할 수 있었습니다. 진심으로 감사드립니다. 그리고 같은 연구실이 아닌데도 불구하고 바쁜 와중에도 어려운 일이 닥쳤을 때 많은 도움을 주시고 아낌없는 조언을 해주셨던 양경부

형, 김영근형 진심으로 감사드립니다. 그리고 지금도 연구실을 거의 이끌어가고 있고 앞으로도 제 뒤를 이어서 우리 연구실을 이끌어가게 될 고정범, 양봉수에게도 진심으로 감사드립니다. 이 둘이 있어서 얼마나 든든한지 모르겠습니다. 그리고 바쁜 와중에도 이제까지 우리 연구실 일을 자기 일처럼 애써 도와주었던 김형찬, 김수진 정말 감사드립니다. 그리고 항상 열심히 하는 우리 실험실의 외국인 동료들 Khalid Rahman, Asif Ali Rehmani, Nong Minh Ngoc 감사 합니다.

지금은 비록 멀리 떨어져 자주 만나지는 못하지만 고등학교 때부터 지금까지 잘 해준 것도 없는 저에게 아낌없이 베풀어주고 격려를 해주었던 성일이, 명환이, 정석이, 봉준이, 힘들다 배고프다 엄살 부려도 아까운 내색 하나 없이 밥과 술을 많이 사주었던 정선이, 태어날 때부터 지금까지 힘이 되어준 모든 학교 동창 겸 친구 진현이, 그리고 학부과정 때 항상 함께 했던 동기들 승보, 승철, 시오, 정보 등등 이 자리를 빌어 진심으로 감사의 마음 전하고 싶습니다. 그리고 학부 1학년 부터 지금까지 항상 저한테 친 동생만큼, 그 보다 더 챙겨줄려고 했었던, 한림 갈 때마다 너무나 반겨주시던 양남진형 그리고 바다이야기횟집 강태훈형 너무 존경하고 너무 감사드립니다. 곧 찾아뵙겠습니다.

마지막으로 언제나 항상 이 못난 아들을 믿으시면서 매일 몸이 부서져라 일만 하시는 아버지, 어머니, 일 많이 못 도와드려서 굉장히 죄송스럽고 고맙습니다. 그리고 90을 넘긴 나이에도 아직도 정정하시고 저를 보살펴주신 할머니 몸 건강히 오래오래 사셨으면 합니다. 그리고 어머니처럼 저를 보살펴주신 고모, 부족한 저를 형, 오빠라고 따라준 동생들 창현, 건너, 여울, 동하 이 모든 분들게 부족하지만 이 논문을 바칩니다.

모두들 감사 합니다.

김창종 드림