계층적 모델링에 의한 두 팔 로봇의 상호충돌방지 실시간 경로제어

Hierarchical Model-based Real-Time Collision-Free Trajectory Control For a Dual Arm Robot System

이 지 홍, 원 경 태
(Ji-Hong Lee and Kyoung-Tae Won)

Abstract : A real-time collision-free trajectory control method for dual arm robot system is proposed. The proposed method is composed of two stages; one is to calculate the minimum distance between two robot arms and the other is to control the trajectories of the robots to ensure collision-free motions. The calculation of minimum distance between two robots is, also, composed of two steps. To reduce the calculation time, we, first, apply a simple modeling technique to the robots arms and determine the interested part of the robot arms. Next, we apply more precise modeling techniques for the part to calculate the minimum distance. Simulation results show that the whole algorithm runs within 0.05 second using Pentium 100MHz PC.

Keywords : collision-free motion, dual arm robots, 3-D distance calculation

I. 서론

사람의 신체 구조를 닮은 두 팔 로봇의 응용 분야가 자주 거론되고 있는 요즘, 컴퓨터의 성능 향상으로 하나의 제어기가 두 대 이상의 로봇 팔을 제어, 관리하는 상용 시스템의 등장이 멀지 않았다고 판단된다. 이러한 제어기는 두 팔을 하나의 제어기가 관장하므로 각 팔이 독립적인 제어기를 갖는 경우에 비해 통신 등의 부담이 적어지고 협조 작업 등의 경우에 일관성을 유지할 수 있어 실질적으로 매우 효율적이라고 할 수 있다. 그런데 이렇게 두 개의 팔이 하나의 시스템으로 제어되는 경우에는 원격조작 시스템처럼 각 팔을 조작자의 각 팔이 관장한다든지 경로 계획이 복잡하여 한 팔씩 계획 한 후 동시에 동작시킨다든지 하는 등의 상황이 발생할 수 있다. 물론 두 팔이 공동의 물체를 들고 있다든지 하는 경우는 본 논문에서 제안하는 방법이 이용될 수 없는 경우이나, 각 팔의 경로 추종이 중요하지 않은 자유공간 이동이나 안전을 위한 보조적인 조치가 필요한 상황에서는 로봇 제어기는 두 팔의 상호 충돌에 끊임없이 대비해야 한다. 이러한 충돌 판단은 실시간으로 이루어져야 한다는 조건을 수반하게 되는데 이러한 실시간 충돌 방지가 본 논문의 주된 관심사가 되겠다.

실시간으로 두 팔 로봇의 상호 충돌을 감지하고 방지하려면 크게 두 가지의 내용이 수행되어야 한다. 첫째는 두 팔이 가까운 정도, 즉 거리를 구해야 하고 그 다음 적절한 경로제어에 의해 충돌을 피하는 방법이 도입되어야 한다.

두 팔 사이의 거리를 실시간으로 구한다는 관점에서 보면 가능한 하나의 방법은 각 로봇의 관절 각도 값이 주어지면 가장 가까운 링크, 위치 및 그 링크 사이의 거리가 구해지는 수식[8]이나 테이블을 만드는 것이나 이 방법은 로봇 기구학식의 복잡성과 많은 관절 변수의 함수를 다루어야 하는 이유로 현실적이지 못하다. 또 하나 가능한 다른 방법은 각 링크의 기하학적인 구조[6]으로 부터 3차원 거리 구하는 문제[1][5][7]로 다루는 방법인데 본 연구에서는 이 방법을 채택하기로 한다. 이러한 과정을 실시간으로 수행하기 위해서는 세밀한 계산을 일일이 수행해서는 곤란하기 때문에 본 연구에서는 먼저 두 로봇 팔을 간단한 기하학적인 구조로 모델링하고 이 모델로부터 간단한 계산과정에 의해 문제의 링크를 구하고(1차 거리계산) 그 문제의 링크들에 대해서만 구체적인 거리 계산을 하는(2차 거리계산) 단계적 방법을 택하였다. 본 연구의 핵심은 [1][2]의 기본 착상이 실제로 이용될 수 있도록 현실화 하였다는 것이다. 이때 중요하게 도입된 아이디어는 간단-복잡으로 이어지는 계층적 모델링과 각 모델링에 있어서의 거리계산 방법이라 볼 수 있고 결과로 개인용 컴퓨터에서 50msec정도의 계산시간 안에 수행이 가능하였다는 것이다. 보통 로봇의 경우 작업공간(Cartesian space)에서의 샘플링 시간이 100msec 정도임을 고려하면 실용적인 결과라고 할 수 있다. 부분적으로는 [9]의 접근 방법과 맥을 같이 하나 실시간 계산이 가능하도록 다단계 거리계산 방법과 임피던스 제어기법을 동력학식이 고려된 실제 상황 모의 실험에 적용하여 효용성을 보인 것이 본 연구의 기여라 할 수 있다. 다시 말해서 첫째 단계로 본 연구에서 제안된 방법으로 문제가 되는 로봇 팔의 부위를 구해내고 일단 두 개의 다면체로 모델링된 두 링크가 구해지면 둘째 단계로 [1]의 방법으로 두 다면체 사이의 거리와 최단 거리가 되는 두 다면체의 각 지점을 구해내고 셋째 단계로 이 두 지점사이의 가상 반발력을 임피던스 개념으로 구해 충돌방지가 안정하게 이루어지도록 하였다. 본 연구의 고유 아이디어는 첫째와 셋째 단계에 있다고 할 수 있다.

두 물체 사이의 거리를 구하는 방법은 컴퓨터 그래픽스 등의 분야에서 많이 연구되고 있으나 본 연구에서 가상한 상황에서는 실시간에 모든 계산이 완료되어야 한다는 제약조건이 가해지게 된다. 로봇 몸체를 다면체로 실제의 형상에 가깝게 묘사하면 계산은 정확하겠지만 작업공간에서 일반적으로 채택되는 100msec 정도의 샘플링 시간을 만족시키기가 매우 어렵다. 본 연구는 실제로 이용될 수 있는 두 팔 로봇 사이의 거리를 구하는 방법을 제시함을 목적으로 한다. 따라서 1차 계산에 이용되는 간단화된 모델로는 계산량과 오차를 고려해 일반적으로 간단화된 모델링에 많이 사용되는 원기둥 대신 타원기둥을 택한다. 타원기둥은 기둥의 중심을 지나는 주축과 타원기둥 단면의 장, 단축만으로 두 타원기둥사이의 거리가 구해질 수 있으며 보통의 로봇처럼

접수일자 : 1996. 5. 20., 수정완료 : 97. 10. 30.
이지홍, 원경태 : 충남대학교 메카트로닉스공학과
※본 연구는 한국과학재단의 특정연구 지원하에 이루어졌습니다.

직사각형의 모양을 갖는 구조에 대해 보통의 원기둥으로 모델링하는 경우에 비해 작은 오차로 모델링이 가능하다. 2차 거리 계산에서는 볼록 다면체간의 거리를 구하는 방법을[1] 이용하였는데 이 방법은 각 다면체의 꼭지점들을 잇는 벡터들의 상호 관계에 의해 불필요한 계산을 없애는 방법으로 전체 꼭지점 사이의 모든 경우를 고려한 경우에 비해 계산량이 적다.

일단 두 팔 사이의 최단 거리가 구해지면 충돌도 피하고 또한 안정적인 동작을 위해 그 거리뿐만 아니라 가까워지는 속도 등이 고려에 포함된 가상 임피던스를[2] 계산하여 두 팔 사이의 거리가 가까워지는 것을 막는 가상 반발력을 생성하도록 하였다. 로봇 몸체의 동력학을 고려한 모의 실험에서 각각 단계별 계산 시간을 구했으며 전체적으로는 펜티엄 100MHz 개인용 컴퓨터로 0.05초 이내에 모든 계산이 수행됨을 확인하였다. 사용한 로봇 모델은 PUMA560이다[3].

이어 2장에서는 2단계로 모델링하는 방법을, 3장에서는 모델로부터 거리 계산하는 과정을 설명한다. 그리고 4장에서는 충돌회피 경로계획을, 5장에서는 PUMA560 로봇에 적용한 예와 계산 시간을, 그리고 6장에서는 결론을 맺도록 하겠다.

II. 계층적 모델링

로봇 팔의 외부 모양은 대체적으로 평면과 곡면으로 덮인 모양이 일반적이다. 이러한 두 팔 로봇사이의 거리를 구하는 방법으로 본 연구에서는 계층적 모델링 방법을 사용하기로 한다. 이 방법의 골자는 일단 간단한 기하학적인 구조로 로봇의 각 부위를 간단화시켜 문제가 되는, 즉 상대편 팔과 가장 가까운 부위(링크)를 구해낸 뒤 그 부위에 대해 정밀한 모델링을 사용하여 실제 거리를 구해내는 단계적인 접근 방법이다. 여기서 해당 부위를 실수없이 찾아내기 위해서는 계산이 간단하면서도 오차가 비교적 작은 모델링 방법이 필요한데 본 연구에서는 이 부분에 타원기둥 모델링 방법을 사용하기로 하며, 정밀한 모델링으로는 다면체 모델링을 사용하기로 한다.

1. 타원기둥 모델링

본 연구에서는 그림 1에 나타낸 연구대상 PUMA560 로봇 팔의 모양을 모양과 복잡성의 정도에 따라 4부류로 분류하였다. 첫째는 (타)원기둥 부분, 둘째는 (타)원기둥의 일부분, 셋째는 사각기둥 모양, 넷째는 모양이 복잡한 손(gripper) 부분이다. 이렇게 분류된 각각에 대해 다음과 같은 방법에 의해 간단한 타원기둥이나 구로 모델링한 후, 주축(3차원 상에서의 선분)을 정의하여 1차 거리계산에서 간단한 계산에 의해 가장 가까운 부위를 구하게 된다. 다음에 각 부류의 주축을 정하는 방법을 설명한다.

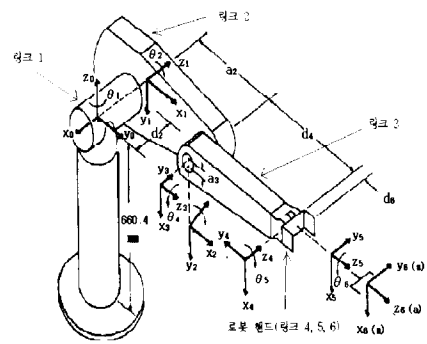


그림 1. PUMA560 로봇의 외관.
Fig. 1. A configuration of PUMA560 robot manipulator.

원래의 모양이 원기둥인 부분은 별도의 모델링이 필요치 않으며 주축은 이 원기둥의 중심 축이 된다(그림 2).

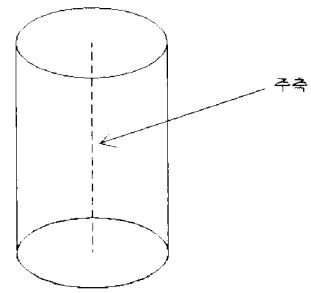


그림 2. 원래 원기둥인 로봇 몸체 부분.
Fig. 2. A cylinder-type part of robot arm.

타원기둥의 일부의 예는 그림 3에 나타내었는데 곡면 부위가 꼭 타원의 방정식을 만족시키지 않는 경우는 그러한 곡면을 포함하도록 타원 모델링을 하면 된다. 이 경우 주축은 가상의 부분까지 포함한 타원 기둥의 중심 축이 된다.

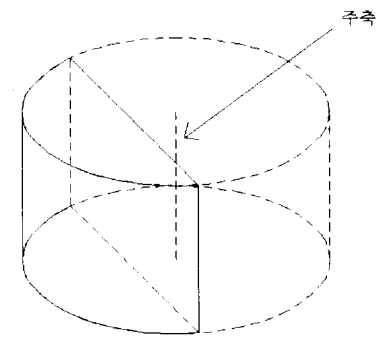


그림 3. 타원기둥의 일부인 로봇 몸체 부분.
Fig. 3. A part of robot arm modeled by part of cylinder.

사각 기둥모양의 예는 그림 4에 나타내었는데 양쪽단면의 크기가 달라도 상관없게 된다. 이 경우 주축은 양쪽 타원의 중심을 잇는 선분이 된다. 타원 기둥을 모델링하는데 있어서 실제 다면체와의 오차를 최소화하기 위해 다음의 계산을 수행한다. 그림 5에서처럼 타원 기둥의 단면은 다면체의 단면을 모두 포함하여야 하며 이를 만족시키는 타원 중 가장 면적이 작아야 한다. 이러한 문제는 다음과 같은 제약조건하에서

$$l \geq a \tag{1-a}$$

$$s \geq b, \tag{1-b}$$

다음으로 표현되는 타원 방정식에서,

$$\frac{a^2}{l^2} + \frac{b^2}{s^2} = 1 \tag{2}$$

면적 $A = l\,s\,\pi$ 를 가장 작게 하는 계수들을 구하면 된다. 여기서 a, b는 주어진 직사각형의 두 변의 길이이며 l, s 는 타원의 장축 또는 단축의 길이이다. 이렇게 구해진 해는

$$l : s = a : b \tag{3}$$

을 만족시키며 구해진 타원의 방정식은 다음으로 주어진다.

$$\frac{x^2}{(bl/a)^2} + \frac{y^2}{l^2} = 1 \tag{4}$$

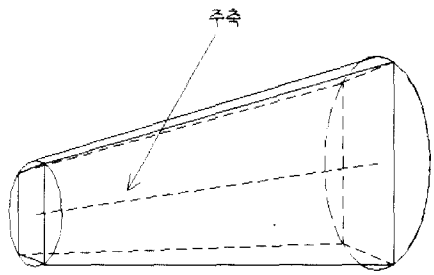


그림 4. 사각기둥의 타원 기둥 모델링.
Fig. 4. Modeling of a square pillar bar by a cylinder.

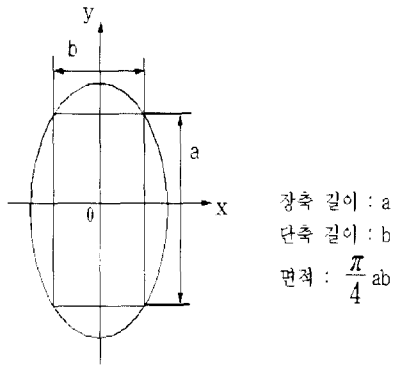


그림 5. 사각형을 둘러싸는 최소면적 타원.
Fig. 5. Minimum-area ellipsoid containing a rectangular.

가장 복잡한 부분이 손목부터 손끝까지의 부분인데 이 부분은 실제로 작업을 위하여 작업 대상물이나 나머지 로봇의 손과 접촉이 잦은 부위이므로 충돌회피 목적으로는 그다지 정밀한 거리계산이 필요치 않다고 가정하여 이 손 전체를 포함하는 최소 부피의 구로 모델링하기로 한다. 이 경우 주축은 원의 중심이 된다(그림 6).

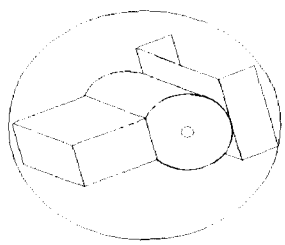


그림 6. 손목 부위의 모델링.
Fig. 6. Modeling for gripper.

이렇게 각 부위가 모델링되고 주축이 정의되면 1차 거리 계산에서는 이 주축끼리의 거리, 즉 두 선분 사이의 최단 거리 벡터 및 거리를 구하고 여기서 각 도형의 주축으로부터 타원기둥 표면과 최단 거리 벡터의 교차점까지의 거리를 뺀 도형간 거리를 구하여 이 중에서 가장 가까운 부위를 구하여 2차 거리 계산으로 넘어가게 된다. 두 선분의 종점의 좌표가 각각 주어져 있을 때 [9]에서의 방법으로 두 선분 사이의 거리는 닫힌 꼴로 구해진다.

표 1은 이제까지 설명한 방법으로 PUMA560로봇을 타원기둥의 모델로 만들었을 때 각 링크의 모델링 오차를 나타낸 것이다. 여기서 링크 2가 각각 두 개의 수치를 가지는 것은 이 링크가 단면적이 다른 두 개의 직육면체의 접합으로 이루어져있기 때문이다. 로봇의 각 링크 중에서 최종 모델링이 언급되지 않은 링크 2, 3을 각 그림 7, 8에 나타내었다. 두 주축, 즉 선분 사이의 거리를 구하는 1차 거리 계산 방법을 다음 장에서 설명한다.

(a) (b)

그림 7. 링크 2의 (a) 실제 모양, (b) 타원기둥 모델링.
Fig. 7. (a) Real shape, (b) modeling of link 2.

(a) (b)

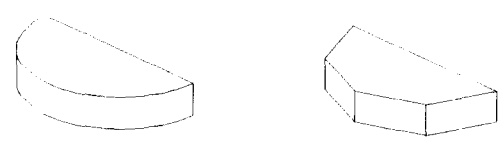
그림 8. 링크 3의 (a) 실제 모양, (b) 타원기둥 모델링.
Fig. 8. (a) Real shape, (b) modeling of link 3.

표 1. PUMA의 각 링크의 다면체와 타원기둥간의 최대오차.
Table 1. Maximum differences between Polytope and Ellipsoid of each link of PUMA robot.

링크 / 수치	링크 2 (mm)		링크 3 (mm)	링크 4 (mm)	링크 5, 6 (求반지름) (mm)
장 축	226.28	113.14	98.995	98.995	127.41
단 축	84.835	84.853	84.853	84.853	127.41
최대 오차	66.28	33.14	18.995	18.995	54.72

2. 다면체 모델링

타원 기둥 모델링으로 주축간의 거리를 계산한 후에는 선택된 부위간의 정밀한 거리 계산이 필요하게 되는데 이를 위하여 로봇 팔의 각 부위를 보다 정확하게 다면체로써 모델링한다. 로봇 팔의 여러 부위 중에서 손 부분은 구로 모델링하며 원래가 원기둥인 부분은 그대로 이용한다. 타원의 일부로 모델링되었던 부분은 그림 9와 같이 근사화 된 다면체로 모델링한다. 또한 다면체로 분리되는 부분은 그 모양 그대로를 이용한다. 이렇게 보다 정밀한 도형 사이의 거리를 계산하는 2차 계산 방법은 다음 장에서 설명한다.



(a) (b)

그림 9. 곡면 부분에 대한 근사적 모델링 : (a) 실제 모양, (b) 근사화 된 모델.
Fig. 9. An approximated modeling for smooth surface: (a) real shapes, (b) approximated model.

III. 2단계 거리계산

다면체간에 거리 구하는 방법은 다면체의 개수에 계산 시간이 주로 좌우된다. 그러므로 두 팔 로봇의 경우와 같이 여러 개의 다면체로 이루어져 있는 형상의 경우는 계산 시간을 줄이는 일이 이 방법 자체로는 불가능하다. 그러므로 여기서는 다면체간의 거리를 구하기 전에 우선 두 로봇의 링크들 중 충돌위험이 가장 높은 것을 간단한 방법으로 선별해 내고 이렇게 구해진 두 링크에 대해 보다 정밀한 계산을 하는 2단계 거리계산으로 계산시간을 절약하는 방법을 제안하겠다.

두 로봇간의 가장 가까운 거리를 갖는 링크들을 구하는 방법은 크게 두 가지의 방향이 가능하나 서론에서 언급한 바와 같이 여기서는 두 팔의 각 관절 각도의 값이 주어지면 두 로봇 사이에 가장 가까운 거리를 주는 링크가 결정되는 수식이나 테이블을 구하는 방법대신 각 로봇의 기하학적 구조를 간단히 모델링한 후, 그 간단화된 형상에 의하여 충돌위험이 없는 링크들 사이의 거리를 계산하지 않는 방법을 택하기로 하겠다. 전자의 경우는 수식을 해석적인 꼴로 구하는 일이 어렵고 테이블을 구한다 하더라도 그 메모리 크기가 상당히 커서 현재로는 바람직한 방법이라고 할 수 없다. 이에 의해 각 로봇은 타원기둥 등으로 모델링되어 두 주축간의 거리를 구하는 1차 거리계산을 거쳐 선택된 두 도형에 대해 보다 정밀한 다면체 모델링으로 2차 거리계산에 의해 최종 거리를 구한다.

1. 두 타원기둥 사이의 거리 계산

두 타원기둥사이의 거리를 계산하기 위해서는 우선 각 타원의 주축을 이루는 두 선분사이의 거리를 계산하여야 한다. 3차원 공간상에서의 두 선분사이 거리를 계산하는 방법으로는 여러 가지가 있을 수 있다. 두 타원기둥의 주축 사이의 거리를 [10]에서의 방법으로 계산할 수도 있지만 본 논문에서 사용되는 방법을 통일하기 위해 선분을 다면체의 일종으로 놓고 다음 절에서 설명할 다면체간의 거리 계산 알고리즘을 그대로 이용하여 두 선분사이의 거리를 구하기로 한다. 즉 두 다면체간의 거리 계산에 사용되는 객체는 꼭지점이 3개 이상으로 이루어진 다면체이고, 이 절에서 두 선분사이의 거리 계산에 사용되는 객체는 꼭지점이 단지 2개로 이루어진 다면체라 설정하고 다면체간의 거리 계산 알고리즘을 쓰면 된다. 이 두 타원간의 그 최단 거리 벡터를 구해낸 다음 이 벡터와 타원 표면까지의 거리만큼을 빼서 두 타원 표면까지의 거리를 구하게 된다.

타원 표면까지의 거리는 두 타원 중 하나의 타원의 주축을 베이스 좌표 계의 x축과 일치하도록 하는 rotation 행렬을 구하여 타원기둥의 단면과 두 주축 사이의 최단거리 벡터를 yz평면에 투영시켜 거리 벡터에서 각 타원기둥의 반지름을 빼서 두 타원기둥 사이의 표면 거리를 구한다. 그림 10에서 한 점은 원점으로 평행 이동시키고 나머지 한 점의 같은 이동 결과 좌표를 (x_p, y_p, z_p)라하면 그림에 표시된 각도들은 다음 식으로 구해진다.

$$\theta = \cos^{-1}(x_p/\sqrt{(x_p^2+y_p^2)}) \tag{5}$$

$$\phi = \cos^{-1}(\sqrt{(x_p^2+y_p^2)}/(x_p^2+y_p^2+z_p^2)) \tag{6}$$

이 각도들로 평행 이동된 주축 벡터를 $Rot(z,\theta)\,Rot(x,\phi)$ 시키면 주축이 x 축과 방향이 일치하고 최단 거리 벡터와 타원 기둥 표면까지의 거리를 구할 수 있게 된다. 여기서 $Rot(z,\theta)$ 및 $Rot(x,\phi)$는 각각 z축, x축을 중심으로 θ, ϕ 만큼씩 회전이동 시키는 변환이다.

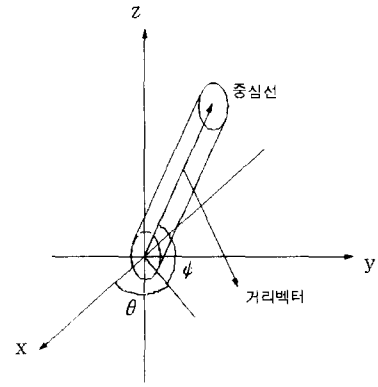


그림 10. 타원 기둥의 주축 사이의 거리계산.
Fig. 10. Calculation of the distance between principle axes of two cylinders.

최단 거리를 갖는 타원 기둥 짝을 구하는데 있어서 모델링 오차를 고려할 필요가 있는데 이는 주축간의 거리를 구한 후 그 거리에 모델링할 때 포함된 오차의 최대 값들을 포함하여 정밀 계산 후보를 복수의 짝으로 구해 냄으로써 해결한다.

2. 다면체간의 거리 계산

두 다면체 사이에서 가장 거리가 가까운 지점과 그 지점과의 거리를 구해 내기 위한 방법으로 본 논문에서는 [1]의 방법을 이용한다. 이 방법의 핵심은 m차원 공간에서 주어진 두 개의 다면체가 있을 때 두 다면체간의 거리계산 문제를 그 다면체들을 이루는 각 꼭지점의 좌표를 서로 빼서 여러 값의 좌표 집합을 만들어서 구해진 좌표 값들의 집합에서 원점에 가장 가까운 좌표 값을 찾는 문제로 간단화시키는 것이다. 이러한 해를 찾아가는 과정에서 각 좌표간의 상호 관계로부터 해가 될 수 없는 부분들을 사전에 걸러 냄으로써 빠른 시간에 원점에서 가장 가까운 점, 즉 두 다면체간에 가장 가까운 지점과 그 거리를 구해 내게 된다. [1]에서는 두 다면체사이의 최단거리 문제를 크게 3단계로 나누어 풀고 있는데 그것은 다음과 같다.

단계 1 : 두 다면체의 꼭지점들의 좌표 값들을 서로 뺀다.

단계 2 : 단계 1에서 생성된 좌표 집합에서 일정한 한도내의 개수를 가진 부분 집합을 선택한다.

단계 3 : 단계 2에서 선택된 부분 집합에서 원점과 가장 가까운 점, 선 또는 면을 찾아 거리를 계산한다.

즉, [1]에서는 단계 1에서 만든 집합의 원소들 중 원점과 가장 가까운 거리를 이루게 되는 부분 집합을 단계 2와 단계 3을 반복하여 찾음으로써 두 다면체사이의 최단 거리를 구하게 된다.

각 단계에 대해서 설명하기에 앞서 우선 [1]에서 주로 사용하고 있는 식이나 정의들에 대해 간단히 설명하기로 한다. 우선 m차원 공간상에서 다면체를 이루는 꼭지점들의 집합을 $X=\{x_1, x_2, \cdots, x_l\}$라고 하면, 그 다면체의 표면을 이루는 모든 점들의 집합을 $co(X)$라 하면 이들은 다음과 같은 식으로 나타낼 수 있다.

$$co(X)=\Big\{ \sum_{i=1}^{l} \lambda^i x_i : x_i \in X,\ \lambda^i \geq 0,$$

$$\lambda^1 + \cdots + \lambda^l = 1 \Big\} \tag{7}$$

그리고 집합 X로 이루어진 다면체에서 원점과 가장 가까운 점 $v(co(X))$도 (7)와 같은 형태로 나타낼 수 있는데 즉,

$$\upsilon(co\ (X)) = \sum_{i=1}^{l} \lambda_{\upsilon}^{i} x_i,\ \ x_i \in X,$$

$$\lambda_{\upsilon}^{1} + \cdots + \lambda_{\upsilon}^{1} = 1 \tag{8}$$

와 같이 나타낼 수 있다.

다음으로 거리계산 주 알고리즘을 기술하기 위해 필요한 몇 가지 함수 및 기호를 정의하기로 한다. 먼저 X의 support 함수, $h_X : R^m \rightarrow R$를 (9)와 같이 정의한다.

$$h_X(\eta) = \max\{ x \cdot \eta : x \in X \} \tag{9}$$

그리고 이 식의 해를 $s_X(\eta)$라고 표시하면 이 해는 다음의 식을 만족하게 된다.

$$h_X(\eta) = s_X(\eta) \cdot \eta,\ \ s_X(\eta) \in X \tag{10}$$

이 수식은 뒤에 설명할 최단거리 주 알고리즘에서 부분집합을 선택하는 기준이 되는 중요한 식이다.

이제부터 앞의 각 단계에 대해 간단히 설명하기로 한다. 우선 단계 1은 각각 꼭지점의 개수가 M_1, M_2개인 두 다면체의 꼭지점의 상대적 위치를 구하는 것인데, 그것은 다음과 같이 (11)과 같이 쓸 수 있고, 이때 집합 Z는 $M_1 \times M_2$ 개의 원소를 갖게 된다.

$$X = co\ (Z) \tag{11}$$

여기서, $Z = \{ z_{1i} - z_{2j} : i = 1, \cdots, M_1,\ j = 1, \cdots, M_2 \}$

단계 2는 최단거리 주 알고리즘으로서 단계 1의 (11)에서 생성된 두 다면체의 꼭지점간의 상대적 위치를 나타내는 집합 Z에서 최단거리를 이루는 부분집합을 선택하는 단계이다. 부분 집합 원소의 개수는 m 차원공간에서 최대 m+1개까지 가능하다. 예를 들면 3차원 공간에서는 부분 집합 원소는 최대 4개까지 있을 수 있게 된다.

$K \subset R^m$인 집합 K가 compact한 볼록 다면체라고 하고 (12)에 의해 $g_K : R^m \rightarrow R$ 라고 정의한다.

$$g_K(x) = |x|^2 + h_K(-x) \tag{12}$$

이상 정의된 기호 및 함수로 다음의 최단거리 계산 주 알고리즘을 정리한다.

최단거리 계산 주 알고리즘 : 다면체 집합 $K \subset R^m$와 초기 꼭지점들 $y_1, \cdots, y_\upsilon \in K, 1 \le \upsilon \le m+1$ 이 주어진 상태에서 다음과 같은 단계를 수행한다.

주 1 : 집합 $V_0 = \{ y_1, \cdots, y_\upsilon \}$ 를 설정하고, $k = 0$으로 놓는다.

주 2 : $\upsilon_k = \upsilon(co\ (V_k))$ 를 설정한다.

주 3 : 만약에 $g_K(\upsilon_k) = 0$이면, $\upsilon(K) = \upsilon_k$로 놓고, 주 알고리즘을 마친다.

주 4 : $V_{k+1} = \hat{V}_k \cup \{ s_K(-\nu_k) \}$로 설정한다. 여기서 $\hat{V}_k \subset V_k$는 원소의 개수가 m보다 같거나 작고, $\nu_k \in co\ (\hat{V}_k)$를 만족한다. 그리고 나서, k를 증가시키고, 단계 주 2로 돌아간다.

즉, 최단거리 계산 주 알고리즘 단계 주 1에서는 우선 단계 1을 통해 얻어진 두 다면체의 꼭지점들의 상대적 위치를 나타내는 점들의 집합의 공집합이 아닌 부분집합을 하나 생성하고 단계 주 2에서는 그 부분집합에서 원점과 가장 가까운 위치를 구한다. 그리고 단계 주 3에서는 단계 주 2에서 구해진 부분집합의 최단거리가 전체집합의 최단거리가 되는가를 점검하는데 그때 바로 (12)를 사용한다. 그리고 만약 단계 주 3에서 실패하면, 단계 주 4에서는 앞 단계에서 사용해 오던 부분집합의 원소들 중 몇 개를 전체집합의 다른 원소로 대체하는데 그때 (9)와 (10)을 사용한다.

단계 3은 최단거리 부 알고리즘으로서 주 알고리즘의 단계 주 2에 해당하고 앞서 언급한 바와 같이 주어진 꼭지점들의 상대위치를 나타내는 좌표집합의 부분집합 중에서 원점과 가장 가까운 거리에 있는 점, 점들을 연결하고 있는 선, 또는 면들 안에 있는 점을 찾아 주는 알고리즘이다. 먼저 다음과 같은 몇 가지 양을 정의를 기술한다.

$$\nu(co\ Y) = \sum_{i \in I_S} \lambda^i y_i \tag{13}$$

$$\sum_{i \in I_S} \lambda^i = 1,\ \lambda^i > 0,\ i \in I_S \subset \{ 1, \cdots, \nu \} \tag{14}$$

$$Y_S = \{ y_i : i \in I_S \} \tag{15}$$

$$\Delta_i(\{ y_i \}) = 1\ ,\ \ i \in I \tag{16}$$

$$\Delta_j(Y_s \cup \{y_j\}) = \sum_{i \in I_s} \Delta_i(Y_s)(y_i \cdot y_k - y_i \cdot y_j),$$

$$k \in I_s\ ,\ \ j \in I_s' \tag{17}$$

$$\Delta(Y_s) = \sum_{i \in I_s} \Delta_i(Y_s) \tag{18}$$

여기서 I_s'은 I에 대한 부분집합 I_s의 보집합이고, Y_s, $s = 1, \cdots, \sigma$, 를 Y의 정렬된 부분집합들이 된다. 그리고 I_s의 원소인 k를 임의로 선택해도 $\Delta_i(Y_s \cup \{ Y_j \})$의 값에 아무런 영향도 끼치지 않음을 밝혀 둔다. 또, (13)의 λ^i는 다음과 같이 주어진다.

$$\lambda^i = \Delta_i(Y_s)/\Delta(Y_s) \tag{19}$$

그러면, 이제부터 최단거리 부 알고리즘에 대해 설명하겠다.

최단거리 계산 부 알고리즘 : 집합 $Y = \{ y_1, \cdots, y_\nu \} \subset R^m$이 주어지고, Y의 정렬된 모든 부분집합을 Y_s, $s = 1, \cdots, \sigma$, 라 한 다음, 다음과 같은 단계를 수행한다.

부 1 : $s = 1$로 설정한다.

부 2 : 만약 $\Delta(Y_s) > 0$, $\Delta_j(Y_s) > 0$, $j \in I_s$, 그리고 $\Delta_j(Y_s \cup \{ y_j \}) \le 0$, $j \in I_s'$의 조건들이 성립하면, (13)과 (19)에 의해 $\nu(co\ (Y))$를 구하고, 부 알고리즘을 마친다.

부 3 : $s < \sigma$이면, s를 증가시키고, 단계 1로 돌아간다.

부 4 : 부 알고리즘을 마치고, 프로그램 실패를 표시한다.

최단거리 부 알고리즘에서 단계 부 4가 발생하는 경우는 주로 계산상의 오차이거나 두 다면체가 충돌하여 거리가 0보다 작을 때 일어난다. [1]에서는 이를 위해 음거리 개념에 대해 설명하는데, 이 논문에서는 두 로봇이 충돌을 하여 서로 파고들지 않는다고 가정하고 있으므로 이러한 경우는 고려 대상에서 제외된다.

IV. 충돌방지 경로 제어

두 팔 로봇의 상호 충돌을 피하기 위하여 본 논문에서 제안한 방법을 정리하면 다음과 같다.

단계 1 : 먼저, 타원기둥 모델로부터 두 팔 사이에 가장 거리가 가까운 링크를 구한다.

단계 2 : 그리고 나서 그 두 링크사이를 실재 모습과 거의 같도록 모델링한 다면체 모델로 가장 가까운 지점을 구한다. 이 지점을 각각 p_1, p_2라 한다.

단계 3 : 회전 관절이라 가정하고 구해진 지점 p_1, p_2까지의 자코비안 매트릭스 $J_1(q_1)$, $J_2(q_2)$를 다음 식에 의하여 구한다.

$$J_{r,i}(q) = [z_{r,i-1} \times p_{r,i-1}],\ \ r = 1, 2 \tag{20}$$

단 여기서 $z_{r,i-1}$는 로봇 r의 $i-1$ 좌표 계의 z축 방향

단위 벡터이고 $p_{r,i-1}$ 는 다음으로 정의된다.

$$p_{r,i-1}=p_r-O_{r,i-1} \tag{21}$$

$O_{r,i-1}$은 로봇 r의 $i-1$ 좌표 계의 원점이고 r이 붙은 벡터는 모두 로봇 r의 기준 좌표계로 표시된다.

단계 4 : 충돌 방지를 위해 다음의 가상의 작업공간에서의 반발력을 구한다.

$$\Delta f_1=[k_1f(z)+k_2f(\dot{z})+k_3f(\ddot{z})]n_{21} \tag{22-a}$$

$$\Delta f_2=[k_1f(z)+k_2f(\dot{z})+k_3f(\ddot{z})]n_{12} \tag{22-b}$$

단, n_{12}는 p_1에서 p_2까지의 벡터를 로봇 1의 좌표 계로 표시한 것이고 n_{21}는 p_2에서 p_1까지의 벡터를 로봇 2의 좌표 계로 표시한 것이다.

변수 z는,

$$z=\frac{1}{d}-\frac{1}{d_0} \tag{23}$$

인데, d 는 p_1, p_2 사이의 거리이고 d_0 는 필요 없는 계산을 피하기 위해 도입된 상수이다. 또한 함수 f 는 다음으로 정의된다.

$$f(z)=\begin{cases}0 & z<0\\ z & z\geq 0\end{cases} \tag{24}$$

단계 5 : 다음 식에 의해 충돌방지를 위한 보정량을 구한다.

$$\Delta\tau_1=J_1^T(q_1)\Delta f_1 \tag{25-a}$$

$$\Delta\tau_2=J_2^T(q_2)\Delta f_2 \tag{25-b}$$

이상으로 구해진 보정량을 조작자의 지령에 추가함으로써 조작자의 명령과 두 팔 사이의 충돌방지를 동시에 고려하는 계획을 수행한다. 결국 로봇의 최종 위치는 다음과 같다.

$$\tau_i = \tau_{cmd,i}+\Delta\tau_i \ , i=1,2 \tag{26}$$

여기서 $\tau_{cmd,}$는 기존 경로계획에 의한 관절 토크이다. 그런데 보정량 $\Delta\tau$를 구하는 과정에는 로봇 모델이 고려되지 않았다. 그러므로 완벽한 충돌회피나 일정한 안전거리 등의 보장이 어려운 실정이고 여러 가상 반발력 방법에서와 마찬가지로 본 연구에서도 이 보정량에 사용되는 여러 계수들은 적용되는 상황에 따라 경험적, 직관적 방법에 의해 결정하기로 한다. 또 모든 온라인 충돌회피 방법이 그렇듯이 충돌회피 동작이 시작되면 원래의 작업 경로로부터 이탈할 수밖에 없음을 주의할 필요가 있고 이러한 가상 반발력 방법은 경로 추종이 중요치 않은 목적지에 도달하기만 하면 되는 자유 운동 등에 유용하게 사용될 수 있다고 판단된다. 그 대신 임피던스 개념을 도입함으로써 속도, 가속도 항에 관계하는 반발력 항이 추가되어 스프링 효과만의 경우보다는 안정적인 동작을 얻을 수 있다고 판단된다.

V. 시뮬레이션 예

본 연구에서 사용한 로봇의 좌표계는 그림 1에 나타나 있고 그림 11은 그 로봇 두 대를 y축으로 60 cm 간격으로 서로 떨어져 놓은 것을 보여주고 있다. 그리고 그림 11에서 한 로봇에 대한 기구학 및 동력학적 계수들은 [10]의 Tarn의 양을 택하였다.

다음으로 충돌방지를 위해 도입된 가상의 전위공간을 도입하여 충돌을 피하도록 하는 방법의 적용례를 보이도록 하겠다. 이 모의 실험에서의 주어진 경로 계획은 두 로봇을 그림 11에서와 같은 초기자세에서 그 중 한 대는 가만히 정지시키고 다른 한 대만 1번 관절을 10초 동안 0°~150°만큼 회전시켜 두 로봇이 서로 부딪히는 경로계획을 그림 12와 같이 하였다. 일단 충돌회피 알고리즘을 적용하지 않았을 때의 거리의 변화를 그림 13-(a)에 나타내었고, 그림 13-(b)는 임피던스화된 가상전위 공간의 효과를 알아보기 위해 $(k_1,\ k_2,\ k_3)=(6,\ 0,\ 0)$으로 하여 거리 정보만을 사용한 경우를 나타낸 것이다. 여기서 충돌회피 알고리즘을 적용하지 않았을 때에는 움직이는 로봇의 링크 2와 움직이지 않는 로봇의 링크 3이 서로 충돌하게 된다. 그리고 그림 13-(b)를 보면 충돌 방지를 위한 가상의 스프링의 역할로 가까워지려는 경로 명령에 보정량을 추가하여 충돌을 방지하는 효과를 관찰할 수 있으나 스프링의 경우와 같이 진동이 유발되는 문제가 발생한다.

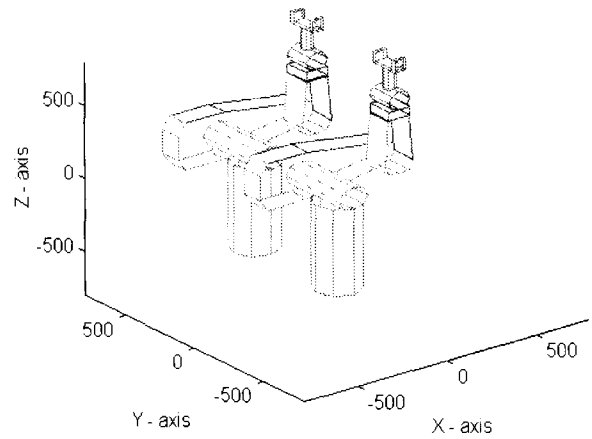


그림 11. 두 대의 PUMA560 로봇.
Fig. 11. Two PUMA560 robots.

다음으로 $(k_1,\ k_2,\ k_3)=(6,\ 100,\ -15)$으로 한 경우는 그림 12에서 볼 수 있듯이 이러한 문제가 해결되었음을 알 수 있다. 이 계수들은 PID 제어기에서의 계수들을 정하는 경우와 마찬가지로 구하기 위한 해석적인 방법을 아직 마련되어 있지 않으나 k_1은 스프링의 역할 k_2는 댐퍼의 역할, k_3는 관성항의 역할을 한다고 볼 수 있고 이러한 역할을 생각하여 몇 번에 걸친 시행착오로 구하였다.

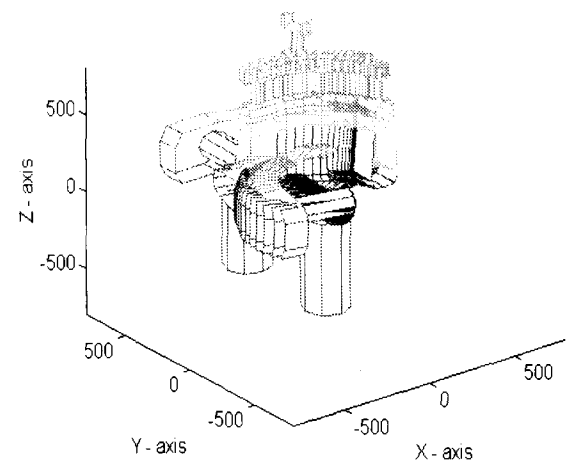
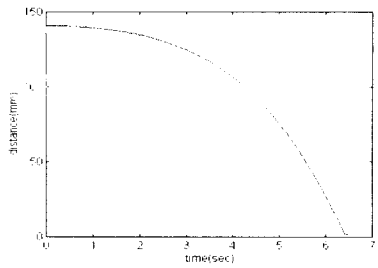
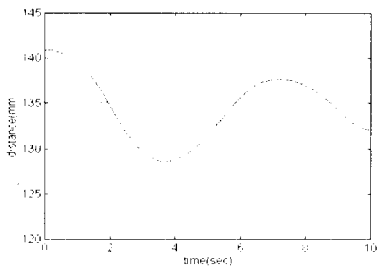


그림 12. 두 팔의 경로 계획에 대한 로봇의 움직임.
Fig. 12. Resultant motion of the robots without collision avoidance algorithm.



(a)



(b)

그림 13. (a) 두 팔의 경로 계획에 충돌방지 기법을 포함시키지 않았을 때의 결과, (b) 두 팔의 경로 계획에 충돌방지 기법을 포함시켰을 때의 결과. $(k_1,\ k_2,\ k_3)=(6,0,0)$ 의 경우.

Fig. 13. (a) Distance profile without collision avoidance algorithm, (b) Distance profile with collision avoidance algorithm for the case of $(k_1,\ k_2,\ k_3)=(6,0,0)$.

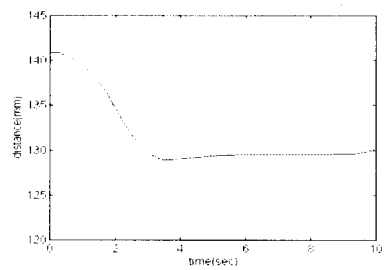


그림 14. 두 팔의 경로 계획에 $(k_1,\ k_2,\ k_3)$ = (6,100, -15)로 충돌방지 기법을 포함시켰을 때의 결과.

Fig. 14. Simulation result with proposed collision-free trajectory planning algorithm $(k_1,\ k_2,\ k_3)$ = (6,100, -15).

또한 그림 14의 상황을 로봇 자세로 그림 15에 나타내었다.

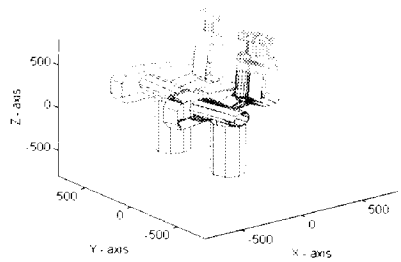
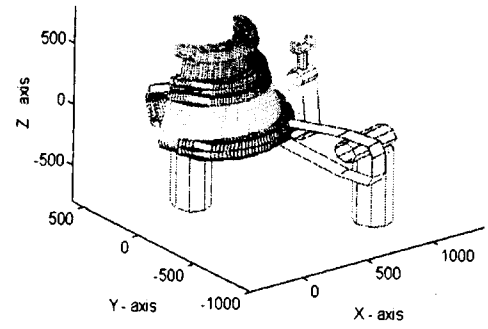


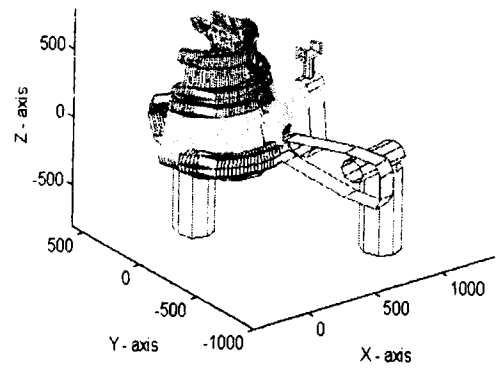
그림 15. 그림 14 상황의 로봇의 움직임.

Fig. 15. Resultant motion of the robots for Fig. 14.

이상의 모의 실험은 충돌 감지에 주안점이 있으나 이 방법이 기존의 경로를 일부 수정하여 충돌이 없도록 경로계획을 변경하는 기능이 있음을 보이는 그림 16의 예를 보인다. 그림 16에서는 왼쪽 로봇의 관절 2가 위로 11.5° 올라가 있고 로봇을 위에서 보았을 때 시계방향으로 1번 축이 40°에서 160°까지 움직이게 하는 계획된 경로를 본 알고리즘이 포함되지 않은 경우와 포함된 경우를 각각 보여주고 있다.



(a)



(b)

그림 16. (a) 두 팔의 경로 계획에 충돌방지 기법을 포함시키지 않았을 때의 경로 결과, (b) 두 팔의 경로 계획에 충돌방지 기법을 포함시켰을 때의 경로 결과.

Fig. 16. (a) Resultant trajectory without collision avoidance algorithm, (b) Resultant trajectory with collision avoidance algorithm.

본 연구에서 중요하게 고려한 것은 실시간에 계산이 가능한가 하는 것이므로 계산에 걸린 시간에 대해 정리하면 다음과 같다. 사용한 컴퓨터는 Pentium 100MHz로 그 결과를 표 2에 나타내었다. 이 표에서 알 수 있듯이 두 로봇을 각각 다면체로 모델링하여 모든 링크에 거리 구하는 알고리즘을 똑같이 적용시키면 거리 계산 시간은 약 0.09초로 실시간 충돌방지 경로계획에 이용되기에는 약간의 무리가 있다. 그리고 좀더 정확한 모델링을 위하여 다면체의 꼭지점을 더 늘리게 되면 시간적 문제는 매우 심각하게 된다. 그 다음으로 타원 모델링 방법을 적용한 경우의 총 계산 시간은 약 0.05초로, 이 결과에 의하면 일단 문제의 링크 두 개가 구해진 후 이 두 개의 다면체간의 거리계산 시간보다는 문제의 링크를 찾는데 적지 않은 시간이 소요됨을 알 수가 있어 간단화 된 모델링 방법에 대한 추가적인 연구로 더욱 시간을 줄일 수 있는 가능성이 있음을 보여준다.

표 2. 각 단계별 계산시간.
Table 2. Calculation time for each step.

단위 ms

	가까운 링크 찾기	다면체의 거리 계산	자코비안 생성(역)	임피던스 제어	합 계
다면체 + 타원 모델	42.1	4.81	0.279	0.081	47.27
다면체 모델	없음.	92.4	0.281	0.088	92.769

VI. 결론

두 팔 로봇시스템에서 두 팔이 서로 충돌하지 않도록 충돌을 방지하는 실시간 수행 방법을 제안하였다. 제안한 방법은 계산시간을 최소화하기 위해 2단계 모델링에 의해 두 팔 사이의 최단 거리를 구하도록 하였는데 첫째로는 로봇 팔의 각 부위를 타원기둥이나 구로 모델링하여 타원의 중심을 지나는 두 선분 사이의 거리를 구하여 문제가 되는 링크들을 찾아낸 뒤, 그 해당 링크들에 대해 보다 정밀한 모델링에 의해 실제 거리를 구해 내는 방법을 제안하였다. 일단 거리가 구해지면 그 거리의 값과 속도 등을 이용하여 가상의 임피던스 힘을 구해 두 팔이 충돌하지 않도록 하였다. PUMA560 로봇을 예로 모의 실험에 의해 제안된 알고리즘이 0.05초안에 수행 될 수 있음을 보였다.

일단 이러한 계산이 로봇의 작업공간 샘플링 시간 안에 계산이 가능함을 보였으나 로봇 제어기는 이러한 충돌 방지 이외에도 다른 여러 일을 수행해야 하므로 단일 CPU를 사용하는 제어기의 경우는 이 계산 시간을 더욱 줄이는 일이 필요하다. 계산 시간을 살펴보면 타원기둥으로부터 문제의 링크를 구해 내는데 상당한 시간이 소요되고 있음을 알 수 있다. 그러므로 이 단계에서 시간을 줄이는 적절한 조치가 추후에 계속 연구될 필요가 있다고 판단된다.

참고문헌

[1] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing and distance between complex objects in three-dimensional space," *IEEE Journal of Robotics and Automation*, vol. 4, no. 2, pp. 193-203, April 1988.

[2] S. A. Schneider and R. H. Cannon, Jr., "Object impedance control for cooperative manipulation: theory and experimental results," *IEEE Transactions on Robotics and Automation*, vol 8, no. 3, pp. 383-394, June, 1992.

[3] K. S. Fu, R. C. Gonzalez and C. S. G. *Lee, Robotics Control, Sensing, Vision, and Intelligence*, New York : McGraw-Hill, pp. 12-52, 1987.

[4] Gilbert Strang, *Linear Algebra and its Application: Third Edition*, Harcourt Brace Jovanovich International Edition, pp. 65-221, 1988.

[5] K. Sridharan and H. E. Stephanou, "On computing a distance measure for path planning," *Proceedings of IEEE Int. Conference on Robotics and Automation*, vol. 1, pp. 554-559, 1993.

[6] J. G. Hocking and G. S. Young, *Topology*, Addison-Wesley, 1961.

[7] K. Sridharan and H. E. Stephanou, "Algorithms for rapid computation of some distance functions between objects for path planning," *Proceedings of IEEE Int. Conference on Robotics and Automation*, pp. 967-972, 1994.

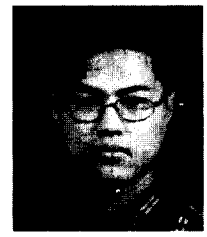
[8] N. K. Sancheti and S. S. Keerthi, "Computation of certain measure of proximity between convex polytopes : A complexity view point," *Proc. IEEE Int. Conference on Robotics and Automation*, pp. 2508-2513, 1992.

[9] 장철, 두 대의 일반적인 로봇 머니퓰레이터를 위한 거리함수를 이용한 충돌회피 방법, 박사학위 논문, 한국과학기술원, 1990.

[10] P. I. Corke, B. A. Helouvry, "A search for consensus among model parameters reported for the PUMA 560 robot," *Proceedings of IEEE Int. Conf. on Robotics and Automation*, pp. 1608-1613, 1994.

이 지 홍

1983년 서울대 전자공학과 졸업. 1985년 한국과학기술원 전기 및 전자공학과 석사. 1991년 한국과학기술원 전기 및 전자공학과 박사. 1994년 ~ 현재 충남대학교 메카트로닉스공학과 부교수. 관심분야는 로보틱스, 인공지능.



원 경 태

1997년 충남대 메카트로닉스공학과 졸업. 1997년 ~ 현재 충남대 석사 과정. 관심분야는 로보틱스, 인공지능, 멀티미디어, 스포츠과학.