'92 韓國自動制御學術會議論文集 1992. 10. 19~21

다중로보트의 동작결정을 위한 시뮬레이터 구성

* 김 정 찬　　김 진 걸

인하대학교 자동화 공학과

Construction of Simulator for Cooperative Multi-Robot Motions

* Jeong Chan Kim　　Jin Geol Kim

Department of Industrial Automation
Inha University

ABSTRACT

We decribe about the graphic simulation system which supports the determination of efficient multi-robot motions during cooperation. For the construction of the simulation software for multi-robot motions, two problems are presented. First problem is that all the robot motions must be determinded using both the desired object motions and the holonomic constraints with the object. To find the robot motions combined with the various object motion path, the robot motions are derived from the desired object path instead of a master robot path. Therefore robot motions can be easily modifiable with the various object motions. This type of motion determination is different from that of the master-slaves method using the master robot motions. The other is that the developments of robot application softwares need a heavy cost when the participated robots or the tasks given to the robots are changed. Based on object-oriented programming paradigm, we present useful software objects describing robot application programming environment. The object-oriented programming paradigm increases the software reusability, reliability, and extensibility, and also provides the structual concepts to cope with the various demands of robot application programming.

I. 서 론

자동화된 생산공정에서 사용되는 대부분의 로보트 시스템은 한 대의 로보트를 이용한 단독작업을 수행한다. 여러 대의 로보트를 이용하여 동일한 대상물을 조작하는 다중로보트 협력시스템은 단일 로보트만으로는 수행하기 어려운 여러종류의 작업을 가능하게 하며, 기존에 설치되었던 로보트들을 재배치하여 로보트들의 이용도를 보다 높힐 수 있는 장점이 있다. 예를 들어 인간의 두팔로써 수행할 수 있는 작업의 종류를 고려한다면 두 대의 로보트를 이용한 협력시스템의 활용방안은 극히 다양하다. 다중로보트 협력시스템의 동작결정을 위한 프로그래밍 방식에 있어서 기존의 직접교시 방법으로는 효율적인 프로그래밍이 용이하지 않으며, 다양한 경로를 갖는 대상물의 동작에 대한 로보트들의 효율적인 경로산출이 곤란하므로 OFF-LINE 프로그래밍을 지원하는 시뮬레이터의 필요성이 더욱 요구된다. 시뮬레이터를 이용하여 실제의 작업 이전에 다양한 작업에 대하여 수행되는 로보트의 상태 및 서로의 간섭을 임의의 시간에 대하여 알아볼 수 있으므로 효율적인 경로를 산출해 낼 수 있다. 로보트의 시뮬레이션 소프트웨어에 관하여 현재 많은 연구가 수행되어 왔으며[1-5], 특히 Voltz[6]는 ADA 언어를 사용하여 로보트를 추상화된 자료형으로 정의하여 생산셀(Manufacturing Cell)에서의 프로그래밍에 적용시켰고 Cox[7]는 객체지향 언어인 C++언어를 사용하여 로보트 응용 소프트웨어의 신뢰성이 향상됨을 보여주었다. 다중로보트의 협력운동에 관한 이전의 연구로서 Lim[8]은 대상물의 경로를 이용하여 로보트들의 위치를 결정하였고, Luh[9]와 Zheng[10]은 로보트들을 주 로보트와 종속 로보트들로 분류하여 로보트들의 위치 와 속도 및 가속도 문제를 다루었다. 주-종(master-slave)방식은 대상물의 주된 움직임을 수행하는 주 로보트의 움직임을 이용하여 다른 로보트들의 동작을 결정하는 방식으로 다양한 대상물의 경로마다 주 로보트의 동작을 알아내야 하는 어려움이 있다. 그러나 수행되는 대상물 좌표계의 이동을 이용하여 로보트들의 동작을 결정하는 방식은 다양하게 주어지는 대상물의 경로에 대해 전체 로보트들의 움직임을 직관적으로 파악할 수 있으며, 또한 대상물 좌표계의 원점을 한 로보트의 엔드이펙터 좌표계와 일치시키면 다른 로보트들의 동작을 종속시킬 수 있다.

본 논문에서는 협력작업을 수행하는 다중로보트의 동작결정을 지원하는 그래픽 시뮬레이션 시스템을 제시하였다. 다중로보트 협력시스템을 위한 시뮬레이션 소프트웨어의 구성을 위하여 다음 두가지 사항을 고려하였다. 첫째, 다중로보트의 동작은 원하는 대상물의 경로와 함께 쥐고있는 대상물과 일정하게 유지되는 엔드이펙터의 고정관계를 이용하여 구한다. 협력작업에서 일정하게 유지되는 엔드이펙터와 대상물과의 고정관계는 엔드이펙터들은 대상물을 미끄러짐 없이 쥐고있다는 가정이 수반된다. 둘째로, 로보트시스템의 환경변화에 대하여 소프트웨어 개발 부담을 줄이기 위하여 객체지향 프로그래밍 기법을 이용한 로보트 응용 프로그래밍 환경의 기본적인 객체들을 제시하였다.

II. 협력작업의 표현

다수의 다관절 로보트들과 대상물로 구성되는 협력작업은 한 대의 로보트를 이용하는 작업들에 비하여 많은 새로운 문제가 발생하게 되며, 그 중에서도 가장 먼저 해결되어야 할 것은 다양하게 움직이는 대상물에 대하여 로보트들의 움직임을 결정하는 문제이다. 본 장에서는 동일한 대상물을 이동시키는 두 로보트의 동작을 유도한다.

2.1 엔드이펙터의 위치및 자세결정

작업대상이 되는 대상물의 시간에 대해 이미 주어진 경로함수를 이용하여 각 엔드이펙터의 위치와 자세를 결정한다. 두 대의 로보트를 이용하는 협력작업 과정에서 각 로보트의 기저 링크는 바닥에 항상 고정되어 있으며, 모든 엔드이펙터는 변형이 없는 강체의 대상물을 미끄러짐 없이 쥐고있다고 가정한다. 따라서 대상물의 어떤 중심점에 대한 모든 엔드이펙터들의 상대적인 위치와 자세는 전체 협력작업 과정에서 일정하게 유지되므로 협력작

업에 참여하는 각 로보트에 대한 엔드이펙터의 위치와 자세를 결정짓는 중요한 조건식이 된다. 그림 1은 강체의 대상물을 고정적으로 취고있는 로보트들의 구성을 나타낸 것으로, 표시된 좌표계들은 다음과 같이 정의한다.

{ R } : 기준 좌표계(Reference Frame)

{ B_i } : i번째 로보트의 기저 좌표계(Base Frame)

{ T_i } : i번째 로보트의 엔드이펙터 좌표계(End-Effector Frame)

{ P } : 대상물 좌표계(Part Frame)

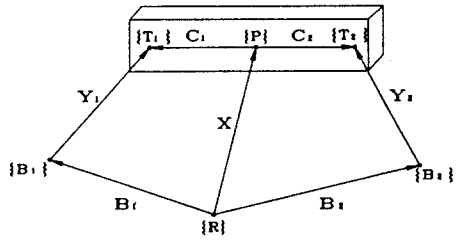

그림 1. 협력 좌표계의 표현

Fig. 1. Representation of Cooperative Coordinate System

그림 1에서 정의된 좌표계들 사이의 변환은 동차변환행렬(Homogeneous Transformation Matrix)이며 이들은 다음과 같이 주어진다.

$$C_i = \left[\begin{array}{c|c} {}^{P}R_i & {}^{P}P_i \\ \hline 0 & 1 \end{array}\right] \in R^{4x4}$$

: 대상물 좌표계에 대한 i번째 엔드이펙터 좌표계의 변환 행렬

$$B_i = \left[\begin{array}{c|c} {}^{R}R_i & {}^{R}P_i \\ \hline 0 & 1 \end{array}\right] \in R^{4x4}$$

: 기준 좌표계에 대한 i번째 기저 좌표계의 변환 행렬

$$X = \left[\begin{array}{c|c} {}^{R}R & {}^{R}P \\ \hline 0 & 1 \end{array}\right] \in R^{4x4}$$

: 기준 좌표계에 대한 대상물의 좌표계의 변환 행렬

$$Y_i = \left[\begin{array}{c|c} {}^{B}R_i & {}^{B}P_i \\ \hline 0 & 1 \end{array}\right] \in R^{4x4}$$

: i번째 로보트의 기저 좌표계에 대한 엔드이펙터 좌표계의 변환 행렬

대상물 좌표계의 움직임을 따라서 협력작업에 참가하는 모든 엔드이펙터 좌표계는 가정에 의하여 상호간에 고정된 위치와 자세를 유지하며 함께 움직인다. 즉, B_i와 C_i는 상수이고 대상물의 경로가 시간의 함수로 주어진다. 결정되어야 할 값들은 각 로보트의 기저 좌표계에 대한 엔드이펙터 좌표계의 시간에 대한 변화로써 다음과 같이 표현된다.

$$Y_i(t) = B_i^{-1} X(t) C_i \quad , \quad i = 1,2 \qquad (1)$$

식(1)은 협력작업 과정에서 이루어지는 엔드이펙터들의 상대적인 위치와 자세에 대한 조건식을 함축하고 있으며 이를 분리하면 다음과 같다.

$${}^{B}P_i - {}^{R}R_i{}^{T}({}^{R}R\ {}^{P}P_i + {}^{R}P - {}^{R}P_i) = 0 \quad , \quad i = 1,2 \qquad (2)$$

$${}^{B}R_i - {}^{R}R_i{}^{T}\,{}^{R}R\ {}^{P}R_i = 0 \quad , \quad i = 1,2 \qquad (3)$$

식(2)는 협력작업을 위하여 각 로보트에 주어진 위치 조건식이며, 식(3)은 자세 조건식을 나타낸다.

2.2 엔드이펙터의 속도와 가속도

각 로보트의 기저 좌표계에 대한 엔드이펙터 좌표계의 속도를 구하기 위하여, 대상물의 주어진 경로함수를 이용하여 구한 순간선속도 v와 각속도 ω로 이루어진 대상물의 속도 $v=[v^T\ \omega^T]^T \in R^{6x1}$를 구한 후, 기준 좌표계에 대한 대상물 좌표계의 원점에서부터 각 로보트의 엔드이펙터 좌표계까지의 위치벡터 $p_i=[p_{xi}\ p_{yi}\ p_{zi}]^T$를 구한다. 여기서 $[\cdot]^T$은 전치(Transpose)행렬을 나타낸다. 따라서 기준 좌표계에 대한 각 로보트의 엔드이펙터 좌표계의 속도는 다음과 같다.

$${}^{R}v_i = h_i{}^{T} v \quad , \quad i = 1,2 \qquad (4)$$

식 (4)에서 h_i는 아래와 같다.

$$h_i = \left[\begin{array}{c|c} I_3 & 0 \\ \hline p_{3i} & I_3 \end{array}\right] \in R^{6x6},$$

$$p_{3i} = \begin{bmatrix} 0 & -p_{zi} & p_{yi} \\ p_{zi} & 0 & -p_{xi} \\ -p_{yi} & p_{xi} & 0 \end{bmatrix} \in R^{3x3}$$

그러므로 각 로보트의 기저 좌표계에 대한 엔드이펙터 좌표계의 속도는 식 (5)와 같이되며

$$v_i = R_i{}^{T} h_i{}^{T} v \quad , \quad i = 1,2 \qquad (5)$$

R_i는 다음과 같다.

$$R_i = \left[\begin{array}{c|c} {}^{R}R_i & 0 \\ \hline 0 & {}^{R}R_i \end{array}\right] \in R^{6x6}$$

또한, 각 로보트의 기저 좌표계에 대한 엔드이펙터 좌표계의 가속도는 아래의 식으로 표현된다.

$$\dot{v}_i = R_i{}^{T} h_i{}^{T} \dot{v} + \dot{R}_i{}^{T} N_i \quad , \quad i = 1,2 \qquad (6)$$

여기서 N_i는 다음과 같이 주어진다.

$$N_i = \begin{bmatrix} \omega \times (\omega \times p_i) \\ 0 \end{bmatrix} \in R^{6x1}$$

Ⅲ. 로보트 객체의 표현

본 장에서는 객체지향 프로그래밍 기법을 이용하여 다양한 로보트 응용 프로그래밍 환경의 기본요소를 이루는 객체들을 제시한다. 로보트 응용 프로그래밍 환경은 로보트의 종류와 부여되는 기능에 따라서 많은 다양성과 복잡성을 갖고 있으므로 요구되는 소프트웨어 환경의 변화에 따라 수정이 용이하지 않고 경제적인 부담이 많이 드는 어려움이 있다. 객체지향 프로그래밍 기법은 소프트웨어 환경내의 각 성분들의 기능을 분류하는 기존의 프로그래밍 방식과는 달리 객체들의 속성과 상호관계의 파악에 기본을 두고 있으며, 정보의 은닉과 추상화 기능 및 객체들간의 상속성(Inheritance)을 특징으로 한다[11]. 객체들의 구성방법으로 프로그래밍 환경에서 독립적으로 관리되는 자료들을 객체로 취급하며, 하부의 객체들은 해당되는 상위 객체안에서 은닉되어 상위 객체가 소속 객체들을 대표하는 계층구조 형식을 취하였다. 정의된 객체는 사용자 정의 자료형(User-Defined Data Type)으로 취급되며, 내장 자료형(Built-in Data Type)과 동일하게 선언된다. 객체지향 프로그래밍 언어인 C++에서 클래스(Class)는 객체를 나타내며 그림 2는 로보트 프로그래밍 환경을 구성하는 기본 객체들의 구성을 보여준다.

3.1 Class ROBOT

ROBOT 객체는 JOINT 객체의 배열및 LINK 객체의 배열을 포함하며, 엔드이펙터 좌표계를 위한 PATH_FRAME 객체를 이용한다. ROBOT 객체는 로보트의 이름을 나타내는 객체의 식별자와 구성되는 관절의 갯수로 구분되고, LINK 객체는 로보트의 몸체를 이루

는 개개의 링크를 나타내며 몇 개의 기하학적인 단위성분들의 조합으로 이루어질 수 있다. FRAME 객체는 동차변환 행렬을 다루는 모든 객체들의 내부에서 사용되며 행렬 연산자 함수가 존재하여 동차변환 행렬의 편리한 조작방법을 제공한다. ROBOT 객체는 ROBOT robot_name; 의 형식으로 이용된다. robot_name으로 ROBOT 자료형에 대한 식별자(Identifier)가 선언되고, 내부의 생성자(Constructor)에 의해 표 1의 값을 입력으로 각 내부 성분들이 초기화 되며, 식별자의 가시성(Visibility)이 사라지면 스스로 소멸자(Destructor)를 호출하여 자료의 초기화 및 종료의 일관성을 제공한다. 표 1은 로보트의 링크 및 관절 값들에 대한 설정 값으로 로보트의 형태에 따라 정의된 입력 화일 형태를 갖는다. 따라서 ROBOT 객체는 식별자와 식별자에 해당하는 각 로보트의 정의화일을 이용하여 스스로 동적인 메모리 할당과 소멸을 수행한다.

표 1. 로보트 객체를 위한 입력항목

1) Kinematic Parameter Description total number of joints and joint types. Denavit-Hartenberg parameters table. joint limit values. 2) Dynamic Parameter Description link masses and inertia tensors. 3) Control Parameter Description motor resolutions and gear reduction ratios.

표 1의 값으로 입력된 Denavit-Hartenberg 변수들은 JOINT 객체와 LINK 객체의 내부로 전달되며 각 관절 좌표계의 변환은 다음과 같이 수행된다.

```
for ( i=0; i<JNT; i++)
D_H_trans[i]=jnts[i].jnt_frame()*links[i].link_frame();
```

객체내의 정보들 중에서 객체의 외부에서 접근이 안되도록 은닉될 필요가 있는 정보들은 해당 객체에 대하여 가시성을 국한하여 내부자료의 안전을 보장하고, 다른 객체내의 정보들는 해당 객체내에서 외부로 허용된 정보들을 이용하여 조작하게 된다. 이런 기능은 복잡한 프로그래밍 환경에서 객체내의 세세한 정보를 다루지 않고 객체들의 구현을 보다 쉽게 한다.

3.2 Class WORKCELL

WORKCELL 객체는 ROBOT 객체의 배열과 협력작업 상태에 있는 각 로보트의 궤적계획을 수행하기 위하여 대상물의 궤적 정보의 저장을 위한 PATH_FRAME 객체의 배열을 포함한다. PATH_FRAME 객체는 경로 좌표계의 벡터표현 성분과 속도 및 가속도 값들을 저장하고, 궤적계획 과정에서 이용되는 객체이다. WORKCELL 객체는 협력작업을 하는 로보트 시스템을 정의하며, 수행되야 할 대상물에 대한 궤적 정보를 이용하여 각 로보트의 운동을 결정하는 역할을 수행한다. 표 2는 WORKCELL 객체의 초기화를 위한 자료이며 협력작업을 하는 로보트들의 기저 좌표계 위치와, 대상물 좌표계에 대한 엔드이펙터의 위치 등을 나타낸다.

표 2. WORKCELL 객체를 위한 입력항목

1) Robots Description total number of robots. robot base frames with respect to station frame. robot constraints with respect to object frame. 2) Object Description mass of object. inertia tensor of object.

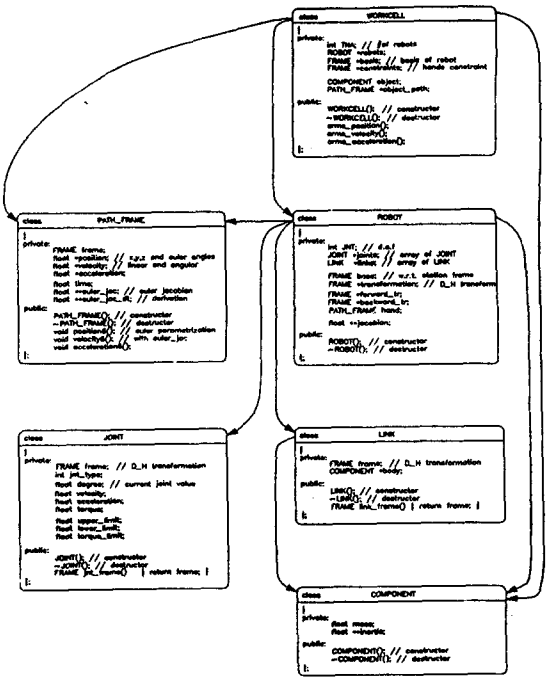

그림 2. 객체의 구성
Fig. 2. Construction of Objects

Ⅳ. 대상물의 궤적계획

3차원 공간상에 주어진 두 경로점 사이를 보간(Interpolation)하는 방법에는 경로점마다 대응되는 관절공간에서 보간을 수행하는 방법과, 경로점의 위치와 자세를 표현하는 R^6공간에서 보간하는 방법이 있는데 후자의 방법은 보간되는 점들이 3차원 작업공간에서의 실제 움직임을 나타내므로 작업공간에서 존재할 수 있는 장애물 등을 회피하는 다양한 경로를 구해낼 수 있다. 수행되는 궤적계획 과정은 PATH_FRAME 객체내의 정보를 이용하여 직교좌표계(Cartesian Coordinate Frame)상에서 직선보간을 한다. 보간대상이 되는 경로점에서의 급격한 속도변화를 피하기 위하여 경로점을 지나는 궤적은 감속의 상태로 경로점을 지난 후 가속을 수행하여 경로점에서의 연속동작을 이루어낸다. 동일한 기준 좌표계에 대하여 표시된 임의의 시간 간격을 갖는 두 개의 직교좌표계를 위치성분과 3개의 독립변수로 변환시킨 회전성분으로 이루어진 6x1 벡터들을 시간 분포함수 S(t)를 이용하여 보간한다. 동차변환 행렬의 9개 회전성분을 3개의 독립변수로 변환시키는 방법으로는 회전을 수행하는 등등한 축(Equivalent Axis)의 각 성분에 회전각의 상수곱을 취하는 방법과, 3개의 축에 대한 연속적인 회전의 곱으로 표현하는 Euler-Angle 방식 등이 있는데 본 논문에서는 Euler-Angle 방식을 이용한다. 보간대상이 되는 두 개의 경로점은 3개의 Euler-Angle 값들과 위치 값들로 구성된 6x1 벡터들인 X_1과 X_2로 변환한 후 다음과 같이 보간한다.

$$X(t) = X_1 (1 - S(t)) + X_2 S(t) \quad , 0 \le S(t) \le 1 \qquad (7)$$

식(7)은 벡터공간내의 두점을 잇는 직선식을 나타낸 것으로 직선상의 점들을 지나는 값들은 아래와 같이 정의되는 함수 S(t)에 의하여 구할 수 있다.

$$S(t) = \begin{cases} \frac{Kt^2}{2T_1} & , 0 \le t \le T_1 \\ Kt - \frac{KT_1}{2} & , T_1 \le t \le T_2 \\ \frac{-K(t-T_2)^2}{2(T_f-T_2)} + Kt - \frac{Kt_1}{2} & , T_2 \le t \le T_f \end{cases} \quad (8)$$

$$K = \frac{2}{T_f + T_2 - T_1}$$

식 (8)에서 K는 보간구간 마다 결정되는 상수로써 S(0)=0, S(T_f)=1을 만족하며 [0,T_1]구간은 선형 가속구간, [T_2,T_f]구간은 선형 감속구간, [T_1,T_2]구간은 등속 운동구간이다.

V. 그래픽 시뮬레이션

다중로보트의 운동계획 과정을 다음과 같은 세 형태로 구분하여 구성하였다.

(1) 초기 상태(Home Position Mode)
협력작업을 수행하기 위한 각 로보트의 계획된 초기위치로 로보트의 현재위치를 이동한다.

(2) 단독 작업 상태(Single Operation Mode)
각 로보트의 이동된 초기위치에서 협력작업이 진행되기 위한 전 상태까지의 이동으로 각 로보트마다 독립적으로 계획되며 다른 로보트들의 동작과의 관계가 적다.

(3) 협력작업 상태(Cooperation Mode)
협력작업 상태에서는 기준 좌표계에 대한 대상물 좌표계의 변환값을 입력으로 대상물에 대한 엔드이펙터들의 고정 관계를 만족하는 동작을 결정하여 협력작업에 참여하는 모든 로보트들의 각각에 대한 기저 좌표계의 값으로 표현된 경로들이 동시에 산출되며, 이는 단독작업 과정에서 이미 주어진 경로와 연결되어 전체 협력작업 과정에 대한 경로가 이루어진다. 협력작업 과정에 참여하는 로보트들의 경로를 대상물의 경로를 이용하지 않고 어떤 한 대의 로보트에 대하여 다른 로보트들의 움직임을 종속 시키는 경우에는, 대상물 좌표계를 협력작업 과정을 이끄는 로보트의 엔드이펙터 좌표계와 동일하게 취급할 수 있다.

협력작업의 예제로 5개의 회전관절을 갖고 있는 Rhino-XR3 로보트 두 대를 이용하여 두 로보트의 마주보는 공통 평면상에서 두가지의 작업에 대한 그래픽 시뮬레이션을 시도하였다.

(A) 예제 1
협력작업의 예제로 5자유도의 두 로보트가 협력하여 30Cm 길이의 막대모양의 대상물을 17Cm 위로 들어올린 후 90도 회전시키는 작업으로 그림 3에 그래픽 시뮬레이션 과정이 나타나 있다. 표 3은 협력작업에 지정된 조건 값들이며 동차변환 행렬의 자세성분과 위치성분을 포함한 3x4 행렬을 나타낸다. 그림 3에서 그림번호 ①은 전체 협력작업의 초기상태이며 그림번호 ④까지의 변환과정은 단독작업 과정이고, 그림번호 ⑧은 협력작업의 수행결과를 나타낸다.

표 3. 예제 1

	로보트 1	로보트 2
기저 링크의 위치	-1.0 0.0 0.0 65.0 0.0 -1.0 0.0 45.5 0.0 0.0 -1.0 2.0	-1.0 0.0 0.0 65.0 0.0 -1.0 0.0 -40.5 0.0 0.0 1.0 2.0
대상물과의 조건식	0.0 1.0 0.0 0.0 0.0 0.0 -1.0 11.0 -1.0 0.0 0.0 0.0	0.0 -1.0 0.0 0.0 -1.0 0.0 0.0 -13.0 0.0 0.0 -1.0 -1.0

(B) 예제 2
예제 1과 다른 협력작업의 예제로 30x20Cm 평판의 대상물과 다른 엔드이펙터의 조건식으로 그래픽 시뮬레이션을 수행하였다. 그림 4에 수행과정이 나타나 있고 그림번호 ④는 표 4의 조건값을 수행하는 엔드이펙터의 상태를 보여준다.

표 4. 예제 2

	로보트 1	로보트 2
기저 링크의 위치	0.0 1.0 0.0 65.0 -1.0 0.0 0.0 45.5 0.0 0.0 1.0 2.0	0.0 -1.0 0.0 65.0 1.0 0.0 0.0 -40.5 0.0 0.0 1.0 2.0
대상물과의 조건식	-1.0 0.0 0.0 0.0 0.0 0.0 -1.0 11.0 0.0 -1.0 0.0 0.0	-1.0 0.0 0.0 0.0 0.0 0.0 1.0 -11.0 0.0 -1.0 0.0 0.0

VI. 결 론

본 논문에서는 두 대의 로보트가 강체의 대상물을 미끄러짐 없이 쥐고있는 상황에서 주어진 대상물의 경로와 작업수행 과정에서 일정하게 유지되는 로보트들과 대상물간의 조건식을 이용하여 각 로보트의 운동을 알아내었으며, 그래픽 시뮬레이션을 통하여 로보트들의 작업 수행 과정과 상호 간섭 여부를 미리 확인할 수 있음을 보였다. 객체 지향적인 프로그래밍 기법을 이용하여 로보트 응용 프로그래밍의 소프트웨어 환경에서 객체들을 정의하고 객체를 계층별로 분류하여 다양한 로보트 프로그래밍 환경에 효과적으로 대응할 수 있는 객체들의 기본적인 구성을 제시하였고 이를 다중로보트 협력시스템의 그래픽 시뮬레이션 프로그램에 이용하였다. 보다 나은 시뮬레이터의 구성을 위하여 CAD Database 객체의 구성과 아울러 다양한 그래픽 처리부분이 요구되며, 협력상태에 있는 로보트의 동작 결정에 있어서는 협력상태 이전 혹은 이후의 개별적인 동작들과의 효과적인 전환 부분이 개발되어야 할 것이다.

참 고 문 헌

[1] B. Nemec, "A Robot Simulation System Based on Kinematic Analysis", Robotica, Vol.3, pp.79-84, 1985.
[2] P. Simkens, "Generating Off-Line Robot Programs with Geometrical Data from A CAD-Database", Proc. Int. Sym. on Industrial Robots, pp.309-322, 1988.
[3] A. Wozniak, and J. Warczynski, "Robot Simulation and Programming System", IFAC Robot Control, pp.437-442, 1988.
[4] E. Trostmann, and B. Palstrom, "CAD-Based Robot Planning and Control", IFAC Robot Control, pp.465-469, 1988.
[5] D. M. A. Lee, "ROBOSIM: A CAD-Based Off-Line Programming and Analysis System for Robotic Manipulators", CAE Journal, pp.141-148, 1990.
[6] R. A. Volz, T. N. Mudge, and D. A. Gal, "Using Ada as A Programming Language for Robot-Based Manufacturing Cells", IEEE Trans. on Syst. Man. Cybern., Vol.14, No.6, pp.863-878, 1984.
[7] I. J. Cox, "C++ Language Support for Guaranteed Initialization, Safe Termination and Error Recovery in Robotics", IEEE Int. Conf. on Robotics and Automation, pp.641-643, 1988.
[8] J. Lim, and D. H. Chyung, "Cooperative Control of Two Manipulators", Proc. IFAC Conf. Control Science and Tech., Vol.2, pp.870-878, 1985.
[9] J. Y. S. Luh, and Y. F. Zheng, "Constrained Relations between Two Coordinated Industrial Robots for Motion Control", Int. J. of Robotics Research, Vol.6, No.3, 1987.
[10] Y. F. Zheng, and J. Y. S. Luh, "Control of Two Coordinated Robots in Motion", Proc. 24th IEEE Conf. Decision and Control, Vol.3, pp.1761-1765, 1985.
[11] G. Booch, Object Oriented Design with Applications, The Benjamin/Cummings Publishing Company, Inc. 1991.

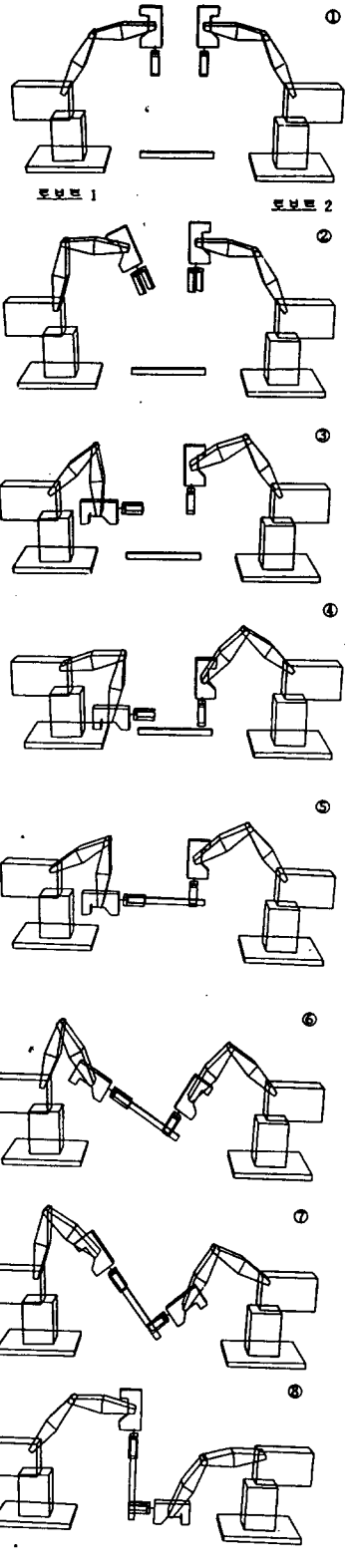

그림 3. 시뮬레이션 1
Fig. 3. Simulation 1

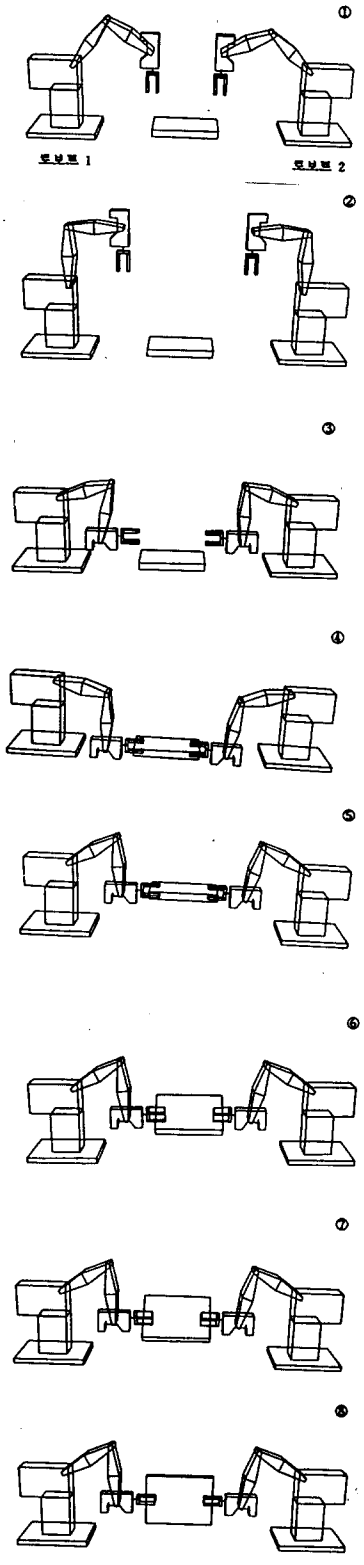

그림 4. 시뮬레이션 2
Fig. 4. Simulation 2