한국공작기계학회 '99년도 추계학술대회 논문집 pp.16~21

가상제조환경에서 공간상의 충돌검색을 위한 이산사건/연속상태 혼합시뮬레이션

황문호, 천상욱, 유민호
{hmh, csu, ymh}@cubictek.com

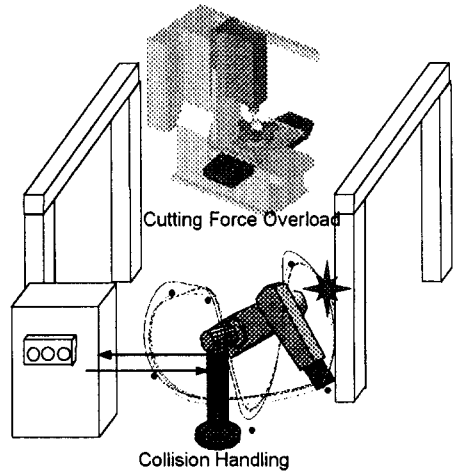
큐빅테크 기술연구소
서울특별시 강서구 등촌동 684-1 에이스테크노타워 1201호

요약

본 연구에서는 컴퓨터 상에 제어기와 구동부 등으로 구성된 가상제조시스템(Virtual Manufacturing System)을 구축하고 시뮬레이션하는 방법론을 다룬다. 특히 설비들간의 이동시 발생할 수 있는 3차원 공간상의 충돌을 검출하는 이산사건 및 연속상태 혼합 시뮬레이션 방법론을 소개한다. 시뮬레이션 모델은 DEVS(discrete event system specification) 형식론(formalsim)에 기초한 형상기구학 정보를 갖는 이른바 GKDEVS(Geometrical Kinematic DEVS)을 이용하였고, 시뮬레이션 방법론은 DEVS의 추상화 시뮬레이션(abstract simulation)방법을 확장하였다.

1. 서론

제조시스템의 제반활동을 가상환경인 컴퓨터 상에 구현하여 시뮬레이션 및 실 운용에 활용하자는 가상제조시스템(Virtual Manufacturing system)의 개념이 소개되고 있으며[Onosato 93][Choi 99] 애니메이션 분야에서부터 그 연구가 시작되었다[Jones 93][Kim 95]. 그러나 최근들어 구성 설비가 단위 공작기기 수준에서부터 유연생산시스템(Flexible Manufacturing System)과 같이 복수개의 설비들이 유기적으로 작동하는 시스템 수준으로 복잡성이 증가함에 따라서 이산사건 시스템과 연속상태 시스템의 통합 시뮬레이션 방법론이 요구되고있다[Klingstam 99]. 특히 고가의 설비들로 구성된 제조시스템의 경우, <그림 1>에서 보여주는 것과 같은 과부하(overload)나 공간상 충돌(spatial collision) 등은 설비 고장과 같은 치명적인 손실을 가져옴에 따라 시뮬레이션 단계에서 이들 오류의 요소들을 제거 해야한다.

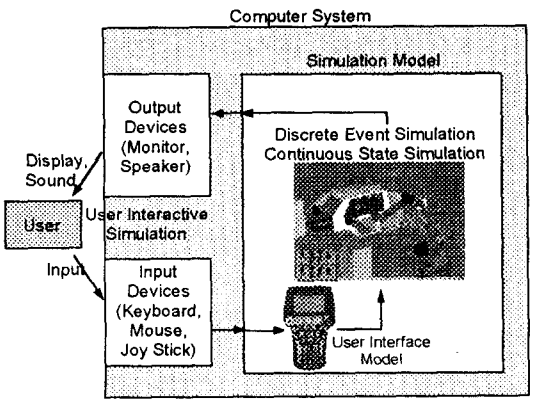


<그림 1> 치명적인 고장을 일으키는 상황들

이에 본 연구에서는 DEVS 형식론[Zeigler 84]과 이의 추상화 시뮬레이션 방법을 확장하여 3차원 설비의 모델링 및 시뮬레이션을 수행하고자한다.

2. 연구 범위

본 연구에서 구축하고자 하는 시스템의 개념을 <그림 2>는 보여주고 있다. 사용자는 시뮬레이션 상황을 애니메이션을 통하여 파악하고, 설비들을 조작할 수 있으며, 내부에서는 모델들 간에 사건을 주고받는 이산사건 시뮬레이션과 충돌 등과 같이 미리 계획할 수 없는 사건(unschedulable event)의 검출 등을 목적으로 한 연속상태 시뮬레이션을 함께 수행되는 환경이다.



<그림 2> 가상제조시스템 환경의 범위

<그림 2>와 같은 환경구축을 위하여 다음 항목의 연구가 필요하다.

1) 가상환경에 맞는 이산사건 및 연속상태 시스템의 모델 개발

2) 시뮬레이션 방법론

- 이산사건 시뮬레이션
- 연속상태 시뮬레이션
- 사용자 참여 시뮬레이션

3. Geometrical and Kinematic DEVS: GKDEVS

형상과 기구학 정보를 갖고 내부에 회귀적(recursive)인 구조를 갖는 확장된 모델은 다음과 같이 정의된다.

$GKDEVS$

$= \langle X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta, M, Z, SELECT \rangle$

where

X: 입력 이벤트 집합; Y: 출력 이벤트 집합; S: 상태변수 집합, $S = \langle GK, S^r \rangle$, $GK = \langle G, type_{link}, F, {}^{U}F, A, v, [v_{\min}, v_{\max}], \rangle$

G: 볼록 다각형 집합; $F = \langle R, P \rangle$; 상위 GK.F의 상대적 프레임, R(3x3 matrix): 상대 회전정보, P ($\in R^3$): 상대 위치 정보; ${}^{U}F = \langle {}^{U}R, {}^{U}P \rangle$: 절대 좌표계상의 프레임, ${}^{U}R$(3x3 matrix): 절대 회전정보, ${}^{U}P$($\in R^3$): 절대 위치 정보; $type_{link} \in \{fixed, prismatic, revolute\}$ 조인트(joint) type; $A \in R^3$: 링크의 축벡터; $vin[v_{\min}, v_{\max}]$ $v, v_{\min}, v_{\max} \in R$: 축벡터상의 값 및 이동범위; S^r: 링크정보를 제외한 자신의 상태변수 집합, w.r.t $GK' \cap S^r = \{\}$; M: 연결된 링크 GKDEVS의 집합; $\delta_{int}: S \rightarrow S$: 내부 상태천이 함수 ; $\delta_{ext}: Q \times (X \cup \{\varnothing, c\}) \rightarrow S$: 외부 상태천이 함수, 단, $Q = \{(s, e) | s \in S, 0 \le e \le ta(s)\}$, 총체적 상태, $\varnothing$: 공(공) 입력 이벤트(non-event); c: 공간상의 충돌이벤트; $\lambda: S \rightarrow Y$, 외부 출력함수; $ta: S \rightarrow R_0^{\infty}$, 시간전진함수; $Z: Y \rightarrow X$: 출력전달함수; $SELECT$: $2^{M \cup \{self\}} - \{\} \rightarrow M \cup \{self\}$, 타이 해결함수;

여기서 프레임(F)를 4x4 동차변환행렬(homogeneous transformation matrix)로 표현하면 다음과 같다.

$$T = \begin{cases} \left[\begin{array}{ccc|c} & R & & P \\ 0 & 0 & 0 & 1 \end{array}\right] & \text{if type= fixed} \\ \left[\begin{array}{ccc|c} & R & & P+vA \\ 0 & 0 & 0 & 1 \end{array}\right] & \text{if type= prismatic} \\ \left[\begin{array}{ccc|c} & RR_A^v & & P \\ 0 & 0 & 0 & 1 \end{array}\right] & \text{if type= revolute} \end{cases}$$

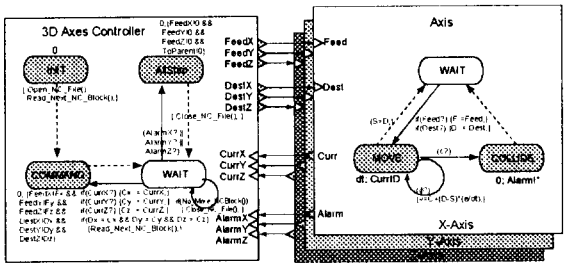
where

$$R_A^v = \begin{bmatrix} a_x a_x kv + cv & a_x a_y kv - a_z sv & a_x a_z kv + a_y sv \\ a_x a_y kv + a_z sv & a_y a_y kv + cv & a_y a_z kv - a_x sv \\ a_x a_z kv - a_y sv & a_y a_z kv + a_z sv & a_z a_z kv - cv \end{bmatrix}$$

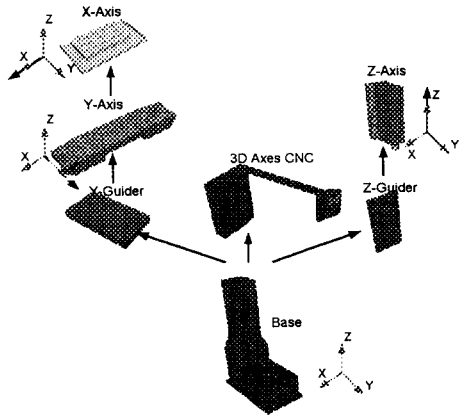
where $cv = \cos v$, $sv = \sin v$, $kv = 1 - \cos v$ $A = [a_x, a_y, a_z]$. 따라서, 어떤 프레임 관점에서 본 형상정보를 ${}^{b}G$라고 한다면 ${}^{b}G = T \cdot G = \bigcup_{g \in G} T \cdot g$로 계산가능하다. 행렬식의 관계와 활용에 대한 자세한 내용은 [Craig 89]를 참조 바란다.

GKDEVS를 이용하여 3축 CNC(computer numerical control) 밀링(milling) 시스템의 모델링 예에 대하여 <그림 3>은 동적인 행태를 갖는 요소와 이들간의 연결(coupling)관계를 보여주고 있으며 <그림 4>에서는 사용된 GKDEVS와 이들간의 계층관계를 보여주고 있다. <그림 3>에서 알수 있듯이, 3축 제어기(3D Axes CNC)는 NC 파일을 읽어 가면서 X,Y,Z 축이 이동해야할 목적지와 이때의 속도를 보낸다. 이때 각 축으로부터 목적지에 도달했음을 알리는 신호(Curr)을 받게되며 만약, 축 이동 중에 3차원 공간상에서 충

돌과 같은 이상상황이 발생하게되면 (c 가 전달됨), 축 모델은 COLLIDE 상태로 빠져, Alarm을 이벤트를 제어기에 전달하게되고, 제어기는 각 축의 이동을 정지시키게 된다.

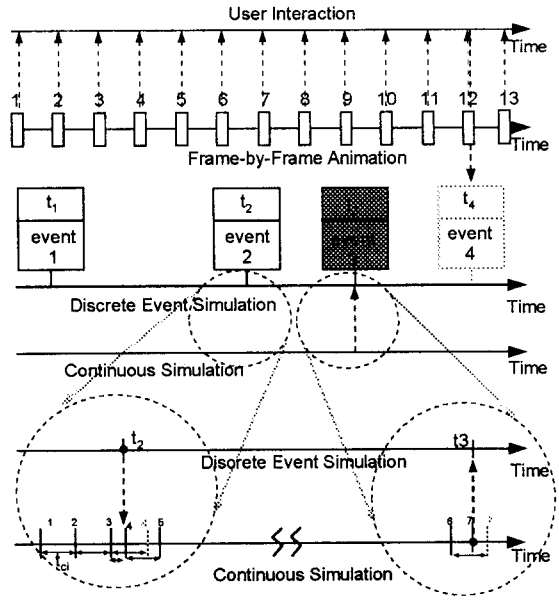


<그림 3> CNC Milling 모델의 상태변이도



<그림 4> CNC Milling 모델의 계층관계

본 연구에서는 기존의 DEVS 모델의 추상화 시뮬레이션 기법[Zeigler 84][Praehofer 91]과 유사한 방법론을 적용하였다. 본 시뮬레이션에서 고려하고 있는 메시지의 종류는 ① (*,t): 내부변이 이벤트 메시지; ② (x,t): 외부변이 이벤트 메시지; ③(done,t): 재 스케줄 메시지; ④ (∅,t): 공 이벤트 메시지(특정시각의 연속상태갱신); ⑤ (c,t): 연속상태에서 충돌이벤트 발생 메시지로 구성된다.

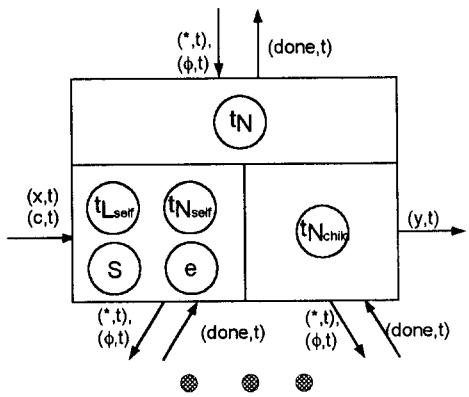


<그림 5> 시뮬레이션 종류와 관계

4. 추상화 시뮬레이션

4.1 추상화 시뮬레이션 개요

<그림 5>는 시간의 흐름과 시뮬레이션의 요소들을 설명하고 있다. 내부에 가장 핵심이 되는 이산사건 시뮬레이션 모델에서는 next schedule기법[Zeigler 84]에 의해서 사건을 계획해 나간다. 이와 함께 진행되는 충돌검색을 위한 연속상태 시뮬레이션은 특정시간까지 연속상태로 진행시키는 단계(<그림 5>의 좌측 하단 원의 확대부분)와 연속상태 시스템에서 발견된 사건(여기서는 충돌이 해당됨, <그림 5>의 우측하단 원의 확대부분)의 이산사건 발생등록의 기능을 수행한다. 이들의 동적인 시뮬레이션 상태는 애니메이션 시각마다 프레임으로 사용자에게 가시화 되고, 사용자는 참여하는 이산사건모델에 전달된다.



<그림 6> GKSimulator의 구조와 messages

4.2 GKSimulator

본 연구에서는 GKDEVS에 대한 시뮬레이션 프로세서(processor)인 GKSimulator를 사용하였는데 이것은 데이터와 콘트롤의 분리 개발차원에서 바람직하다

[Kim 97]. GKSimulator는 <그림 6>에서와 같이 자신의 마지막 이벤트 발생시각(t_{Lself}), 모델의 상태변수집합(S), 상태지속시간(e), 그리고 자신의 다음 이벤트 계획시각(t_{Nself})를 관리하고 있다. 또한 자식들의 다음 이벤트 계획시각(t_{Nchild})을 기억하고 있다.

```
Procedure GKSimulator::when_receive_(*,t)
 1 if t = t_N then
 2     if t_Nself < t_Nchild or t_Nself = t_Nchild and
          self is selected by SELECT then
 3         y := λ(s);
 4         send to (y,t) to influencee simulators;
 5         s := δ_int(s);
 6         t_Lself := t;
 7         t_Nself := t_Lself + ta(s)
 8     else
 9         find the immiment simulators;
10         select i' and send the input (*, t)to it;
11         t_Nchild := minimum of component t_N's;
12         resort children by their t_N ;
13     end if
14     t_N := MIN(t_Nself, t_Nchild);
15 else
16     ERROR;
17 end if
Procedure GKSimulator::when_receive_(x,t)
 1 if t_L <= t <= t_N then
 2     e := t-t_L;
 3     s := δ_ext(s,e,x);
 4     t_Lself := t;
 5     t_Nself := t_L + ta(s);
 6     t_N := MIN(t_Nself, t_Nchild);
 7     send (done, t_N) to the parent Simulator;
 8 else
 9     ERROR;
10 end if
Procedure GKSimulator::when_receive_(done,t)
 1 if t_L <= t <= t_N then
 2     resort children by their t_N ;
 3     t_Nchild := minimum of component t_N's;
 4     t_N := MIN(t_Nself, t_Nchild);
 5     send (done, t_N) to the parent Simulator;
 6 else
 7     ERROR;
 8 end if
Procedure GKSimulator::when_receive_(ϕ,t)
 1 if t_L <= t <= t_N then
 2     e := t-t_L;
 3     s := δ_ext(s,e,ϕ);
 4     for-each m in child simulators
 5         send (ϕ,t) to m;
 6     end_of_for-each
 7 else
 8     ERROR;
 9 end if
Procedure GKDEVS::δ_ext(s,e,ϕ)
 1 uT_p := get pareant uT;
 2 uT   := uT_p s.T;
 3 uG   := G uT;
Procedure GKSimulator::when_receive_(c,t)
 1 if t_L <= t <= t_N then
 2     e := t - t_L;
 3     s := δ_ext(s, e, c);
 4     t_Nself := t;
 5     t_N := MIN(t_Nself, t_Nchild);
 6     send (done, t_N) to the parent Simulator;
 7 else
 8     ERROR;
 9 end if
```

<그림 7> GKSimulator의 메시지 처리과정

<그림 7>은 GKSimulator의 5가지의 메시지, 즉 (*,t), (x,t), (done,t), (∅,t), (c,t) 처리과정을 보여주고 있다. <그림 7>의 처리과정에 대한 좀더 자세한 설명은 [Hwang 99]를 참조 바란다.

4.2 RootCoordinator

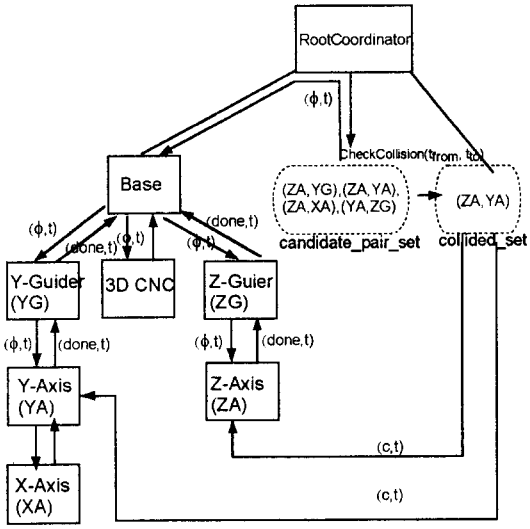
시뮬레이터의 계층구조상에서 가장상위에 존재하는 RootCoordinator의 순환루프를 <그림 8>은 보여주고 있다. 시뮬레이션 종료시각(t_f)에 도달하거나 처리해야 할 사용자의 입력을 담고있는 $buffer_x$가 비이있는 경우 시뮬래이션은 종료된다(<그림 8>의 4번째줄). 시뮬레이션이 수행되는 동안 (1) 사용자의 입력 명령을 처리하거나(<그림 8>의 5~7줄) (2) 연속상태 시뮬레이션을 수행하거나(8~13줄), (3) 스케줄 된 이산사건 이벤트(*,t)를 수행시키거나(14~17줄) (4) 애니메이션을 수행한다(18~20줄).

```
Procedure RootCoordinator:main(TimeType t_f,
                               TimeType t_ci,
                               TimeType t_vi)
 1 TimeType t_ne = its child simulator's t_N;
 2 TimeType t_cc := 0; /* 연속상태 시뮬레이션 시각 */
 3 TimeType t_nv := t_vi;
 4 while (buffer_x is not empty ||
          MIN(t_ne, t_nv, t_cc+t_ci) < t_f) begin
 5     if buffer_user_event is not empty then
 6         x_user = remove the first event from buffer_x;
 7         send (x_user, t_now) to its child simulator;
 8     else if t_cc < MIN(t_ne, t_nv) then
 9         TimeType t_nc := MIN(t_cc+t_ci,MIN(t_ne, t_nv));
10         t_cc := CheckCollision(t_cc, t_nc);
11         if t_cc < t_nc then /* no collisions */
12             t_ne := its child simulator's t_N;
13         end if
14     else if t_ne <= t_nv then
15         send (*, t_ne) to its child simulator;
16         send (ϕ, t_ne) to its child simulator;
17         t_ne = its child simulator's t_N;
18     else
19         update_screen(); /* draw windows */
20         t_nv := t_nv + t_vi;
21     end if
22 end_of_while
23 send (ϕ, t_f) to its child simulator;
```

<그림 8> RootCoordinator의 메인 루프

4.3 충돌 검사 과정

본 연구에서는 충돌 검사를 위해 두 집합의 개념을 도입하였는데, 하나는 검사를 받을 대상의 집합(candidate_pair_set)이고 또 다른 하나는 검사에서 충돌 판정을 받은 것들의 집합(collided_set)이다. <그림 9>는 충돌 검사와 충돌 발생시의 이산사건의 재 스케줄링 개념을 보여주고 있다.



<그림 9> 충돌검색과 사건의 재 스케줄 과정 예

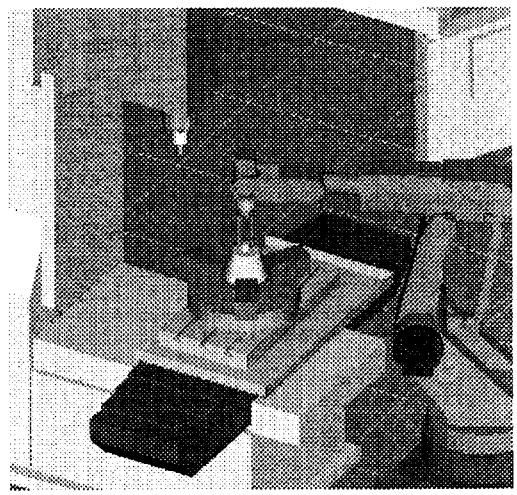
먼저 사용자에 의해서 정의된 충돌 가능한 집합인 candidate_pair_set이 형성 돼 있는 상황에서 앞서 설명한 것과 같이 연속상태의 시간 진행방식에 따라 CheckCollision()이란 함수가 호출된다. 앞서 <그림 4>에서 소개한 3축 머시닝 센터에서 축의 이동시 충돌이 발생할 수 있는 조합으로 (Z-Axis(ZA), Y-Guider(YG)), (Z-Axis(ZA), Y-Axis(YA)), (Z-Axis(ZA), X-Axis(XA)), (Y-Axis(YA), Z-Guider(ZG))가 있다고 하면, <그림 9>에서와 같이 candidate_pair_set에 원소가 된다. <그림 9>에서는 축들간의 이동 중에 Z-Axis와 Y-Axis가 충돌된 상황에서 (c,t) 메시지가 Z-Axis와 Y-Axis의 GKSimulator에게 전달되어지고 이것으로 재 스케줄링을 알리는 (done, t)가 상위로 전달되어지는 과정을 보이고 있다.

검색조건이 끝난 시점에 collided_set에 있는 모델들의 시뮬레이터들에게 이산사건이 일어났음을 알리는 (c,t)를 전송한다. (c,t)를 받으면 시뮬레이터의 계층구조를 따라서 (c,t)의 시점을 고려한 재 스케줄 작업이 일어난다. candidate_pair_set의 두 개의 GKDEVS 모델을 꺼내어 충돌을 검사하는 과정에서 polygon으로 구성된 형상의 충돌을 검사하는 연구[Hubbard 96][Hudson 97]들이 활용 가능하다.

5. 유연 제조시스템으로의 적용

GKDEVS모델과 GKSimulator를 이용하여 유연 제조시스템(Flexible Manufacturing System)의 시뮬레이션 연구에 적용하였다. 대상 제조시스템은 한 대의 CNC 선반(Turning Machine)과 한 대의 CNC 밀링(Milling Machine)을 절삭설비로 가지고 있으며 하나의 입력 테이블과 출력 테이블을 갖는다. 중간의 물류의 이송을 담당하는 설비로 6관절 로봇이 사용되는데 이 로봇은 무인대차설비 위에 탑재되어 있다.

4장에서 설명한 시뮬레이션 방법론에 따라서, 선반 밀링 그리고 로봇의 물류 이송시의 충돌들의 발생유무를 <그림 10>에서와 같이 단위작업별로 수행할 수 있다. 이때에는 충돌 등의 발생과 이의 제거에 관심을 가지고 시뮬레이션을 수행하게 되므로 연속상태의 샘플링 간격을 조밀하게 하는 것이 유리하다. 각 단위 작업의 설계가 완료되며, 전체 시스템 관점에서의 유기적인 운영에 초점을 맞추어 <그림 11>과 같은 시스템 레벨의 시뮬레이션이 진행된다. 이때에는 단위작업에서 수행된 충돌 등의 검사는 불필요하므로, 충돌 검사 등의 옵션을 선택하지 않는 것이 속도 측면에서는 바람직하겠다.



<그림 10> 로봇의 물류 공급 시뮬레이션

6. 결론 및 추후연구과제

본 연구에서 DEVS를 기초로 하여 가상제조시스템을 모델링할 수 있는 GKDEVS를 제안하였다. 제안된 GKDEVS는 이산사건시스템의 특징과 공간상의 운동에 따른 계층적이고 연속적인 영향관계를 모델링할 수 있다. 뿐만 아니라, GKDEVS에 대한 추상화 시뮬레이션 방법론을, 이산사건 및 연속상태 시뮬레이션과 사용

자 참여 시뮬레이션의 관점에서 제안되었고 이를 응용하여 유연제조시스템의 시뮬레이션을 단위설비들로부터 시스템 수준으로 확장하면서 실시하였다. 단, 시뮬레이션, 애니메이션, 연속상태 시뮬레이션의 수행성 평가와 개선에 관한 추후연구가 필요하다.

참고 문헌

[Choi 99] 최병규, "VMS 기술과 산업공학의 역할," *IE 매거진*, V6, N1, 1999, p18-21

[Craig 89] John J. Craig, *Introduction to Robotics, Mechanics and Control*, 2nd Edition, Addison Wesley, 1989

[Hwang 99] 황문호, 최병규, 천상욱, "가상제조환경을 위한 형상기구학 모델링 및 시뮬레이션으로의 DEVS 확장," *한국시뮬레이션학회 '99 추계학술대회 논문집, 1999.10.9 한양대학교*, p24-29

[Hubbard 96] Philip M. Hubbard, "Approximation Polyhedra with Spheres for Time-Critical Collision Detection," *ACM Transactions on Graphics*, V15, N3, July 1996

[Hudson 97] T.C. Hudson, M.C. Lin, J. Cohen, S. Gottschalk, D. Manocha, "V-Collide: Accelerated Collision Detection for VRML," *In Proc. of VRML Conference*, p119-125, 1997

[Jones 93] K.C. Jones and M.W. Cygnus, "Virtual Reality for Manufacturing Simulation," Proceedings of the 1993 Winter Simulation Conference, 1993, p882-887

[Kim 95] B.H. Kim, A Study on the Development of an Animation for AMS Graphic Simulation, MS Thesis, IE Dept., KAIST, 1995 (in Korean)

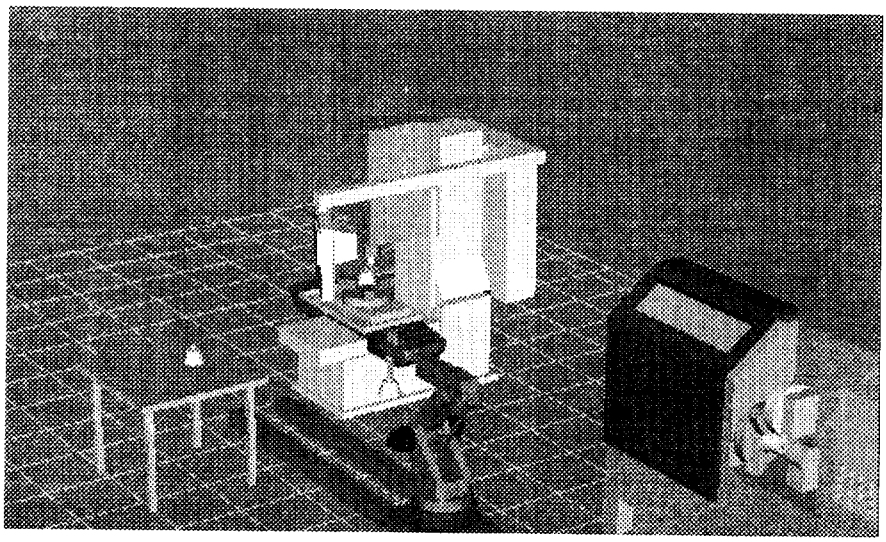
[Kim 97] T.G. Kim, Lecture Note: EE617, EE Dept, KAIST, 1997 (http://sim.kaist.ac.kr)

[Klingstam 99] p. Klingstam and P. Gullander, "Overview of Simulation Tools for Computer-Aided Production Engineering," Computer in Industry, V38, 1999, p173-186

[Onosato 93] M. Onosato and K. Iwata, "Development of a Virtual Manufacturing System by Integrating Product Models and Factory Models," Annals of the CIRP, V42, 1993, p475-478

[Praehofer 91] H. Praehofer, System Theoretic Foundation for Combined Discrete-Continuous System Simulation, Ph.D. Thesis , Department of Systems Theory, University of Linz, Austria, 1991

[Zeigler 84] Bernard P. Zeigler, *Multifacetted Modeling and Discrete Event Simulation*, Academic Press, Orlando, FL, 1984,



<그림 11> FMS 전체 시뮬레이션