OLP 기반 이벤트/사이클 혼합 방식 산업용 로봇 시뮬레이션 엔진 구현

한정욱° 류기열 이정태 범진환 김종철 김재욱
아주대학교 전문대학원
{scol°, kryu, jungtae, jhborm, ehkjc, allah}@ajou.ac.kr

Implementation of OLP based Industrial Robot Simulation Engine using mixed Event-driven/cycle-based Simulation

Jungwook Han°, Kiyeol Ryu, Jungtae Lee, Jinhwan Borm, Jongchul Kim, Jaewook Kim
The Graduate School of Information and Communication, Ajou University

요 약

산업현장에서 로봇의 사용이 크게 늘게 됨으로서 로봇의 배치와 움직임의 조정을 효율적으로 하는 것이 중요해 졌다. 이를 위해 가상의 공간에서 시뮬레이션 하는 오프라인 프로그래밍이 사용되고 있다. 본 논문에서는 오프라인 프로그래밍 기반의 시뮬레이션 엔진을 설계, 구현하였다. 복수의 로봇 시뮬레이션이 정적/동적인 상황에서 이루어 질 수 있도록 이벤트/사이클 혼합 방식을 사용 하였다.

1. 서 론

최근 산업 현장에서는 생산성 향상, 가격 경쟁력 확보, 제품 개발 주기 단축 등이 중요해 지고 이것을 이루기 위해 국내 산업 현장에서 로봇의 사용이 크게 늘고 있다. 산업용 로봇은 고가의 설비이므로 배치 시간, 잘못된 설계에 의한 공정의 정체나 유휴에 따른 손실이 크기 때문에 단시간에 효율적으로 움직이고 움직임의 충돌이 없도록 배치하고 움직임을 조정하는 것이 필요하다. 또한 다품종 소량 체제에서는 제품의 종류마다 생산방식이나 흐름이 다르기 때문에 설비를 적합한 장소에 경제적으로 배치시키는 것이 중요하다.

로봇의 움직임을 결정하고 배치할 시에 사용자가 로봇의 움직임을 직접 확인하면서 위치를 수정, 프로그램을 변경하는 온라인 작업은 많은 시간이 소요되고 숙달된 기술자를 요구하여 제품 생산 시간이 길어지고 많은 비용이 들게 된다. 이러한 문제를 해결하기 위해 가상의 로봇을 동작시키는 프로그래밍 방식인 오프라인 프로그래밍(OLP : Off-Line Programming)이 사용되고 있다.

본 논문에서 구현된 시뮬레이션엔진은 OLP기반으로 가상의 로봇에 프로그램을 넣어 실제 로봇이 작동하는 것과 같이 작동시키고 이를 컨트롤 하는 방법을 사용한다. 실제 로봇과 달리 한 번에 하나의 가상 로봇만을 움직일 수밖에 없기 때문에 이벤트 방식을 이용하여 복수의 가상 로봇의 순차적 실행이 동시 실행과 같은 결과를 낼 수 있게 하였다. 또한 이벤트 방식만으로는 로봇이 움직이는 도중 발생하는 충돌 검사 등의 동적 상황의 시뮬레이션이 불가능 하므로 사이클 방식을 혼합함으로써 동적인 상황의 복수의 가상 로봇 시뮬레이션을 수행 할 수 있게 하였다.

2. OLP 기반 로봇 시뮬레이션

2.1 로봇 시뮬레이션 환경

OLP는 로봇과 로봇이 작업하는 환경을 컴퓨터상에서 가상현실 시스템으로 구현하여 로봇 본체와 물리적으로 분리된 환경에서 동작이나 작업을 지시하는 프로그래밍 방법이다. OLP기반의 로봇 시뮬레이션은 로봇 프로그래밍 언어를 가상 환경 내에서 실행 시키고, 이에 따라 가상의 로봇을 동작시켜 문제점을 찾아내고 동작의 효율을 검증할 수 있게 한다.

구현된 시뮬레이션 환경(VWORKS)은 SE, VM, VWORKS로 구성되어 있다. 시뮬레이션을 조정하는 것이 SE(시뮬레이션 엔진), 가상 로봇이 VM(Virtual Machine), 로봇을 화면에 나타내고 이를 조정하는 GUI가 VWORKS이다. VM은 로봇 프로그램을 실행 시켜 가상 로봇의 움직이고, SE는 복수의 로봇(VM)이 작동 할 때 올바른 시뮬레이션 결과가 도출 될 수 있도록 VM의 실행을 조정하고 스케줄링한다.

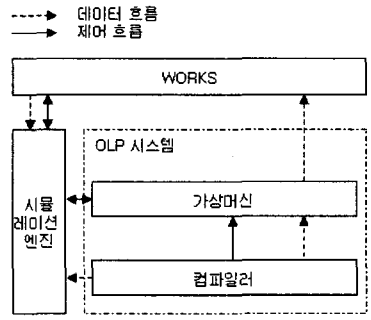

그림 1 전체 시스템 구성도

2.2 로봇 시뮬레이션 언어의 특성

설계한 로봇 시뮬레이션 언어인 WSL은 실제 로봇의 작동을 위한 언어와 시뮬레이션을 위한 언어의 특징을 갖추고 있다. 로봇의 작동을 위한 구문으로 parallel, remote, signal, motion command, if, while이 있고, 시뮬레이션을 위한 구문인 wait until, when, global 이 있다.

표 1 로봇 시뮬레이션 언어 구문

구문	의미
remote	다른 로봇을 작동시킨다. parallel 안에 존재한다
parallel	remote를 이용하여 다른 로봇과 동시에 작동하도록 한다. 각 로봇의 종료를 동기화 시킬 수 있다.
wait_until	조건이 만족될 때까지 프로그램의 수행을 멈춘다.
when	특정 조건이 만족될 때 프로시저를 실행시킨다. 조건으로 cycle시간 경과, 특정 시간 경과, 관련 전역 변수 값의 변경이 있다.
motion command	가상 로봇을 움직인다.
if, while	분기, 반복구문이다
signal	다른 로봇에 signal을 보낸다.
global	전역 변수이다

3. 로봇 시뮬레이션 엔진 설계

3.1 이벤트/사이클 혼합 방식의 시뮬레이션

실제 로봇이 동시에 움직이는 것과 다르게 OLP기반의 시뮬레이션에서는 한 순간에 실행 시킬 수 있는 로봇(VM)이 하나이다. 따라서 복수의 로봇을 동시에 시뮬레이션 하기 위해서는 각 VM이 실행할 프로그램을 실행 구간 나누고 모든 VM이 가지는 실행 구간을 병합하여 순차적으로 실행하는 방법을 사용한다. 실행 구간은 하나의 VM이 다른 VM에 영향을 주거나 받지 않고 로봇의 움직임에 변화가 없는 독립 실행 구간과 그렇지 않은 연관 실행 구간으로 구분된다. 하나의 VM에서 각 구간의 실행을 일시에 하는 것에는 시뮬레이션 결과에 영향을 끼치지 않는다. 또한 서로 다른 VM의 독립 실행 구간의 순차적인 실행, 같은 시점에 있는 서로 다른 VM의 실행 구간(독립 실행 구간, 연관 실행 구간)의 순차적 실행 역시 실행 순서와 관계없이 올바른 시뮬레이션 결과를 얻을 수 있다. 따라서 각 VM의 실행 시점을 병합했을 때 연관 실행 구간과 연관 실행 구간 전후의 독립 실행 구간 모임이 시간적 순서에서 틀리지 않다면 순차적 시뮬레이션에서 동시 시뮬레이션의 결과를 얻을 수 있게 된다. 이 때 연관 실행 구간의 시작과 끝(연관 실행 구간이 하나의 시점이었을 때는 연관 실행 구간 자체)을 이벤트로 본다(그림 2). 따라서 표 1의 시뮬레이션 언어 구문에서 발생될 수 있는 이벤트는 표 2와 같다. SE의 관점에서 이벤트는 VM이 자신 이외의 VM에게 특정 상황을 알려야 할 때, SE에게 자신의 상황을 알릴 때, VM이 처리하지 못하는 명령을 SE가 처리하도록 위임 할 때 발생하게 된다.

SE에서 복수의 VM을 동시에 시뮬레이션 하기 위한 스케줄링은 위에서 언급한 이벤트를 관리하는 방법으로 자연스럽게 이루어지고 SE에서 VM에 이벤트를 요청하면 VM에서 이벤트 발생 전 시점까지 실행 후 이벤트 생성, SE가 이벤트를 받아 처리 후 VM이 이벤트를 처리 하는 순서로 로봇에 대한 시뮬레이션이 이루어진다.

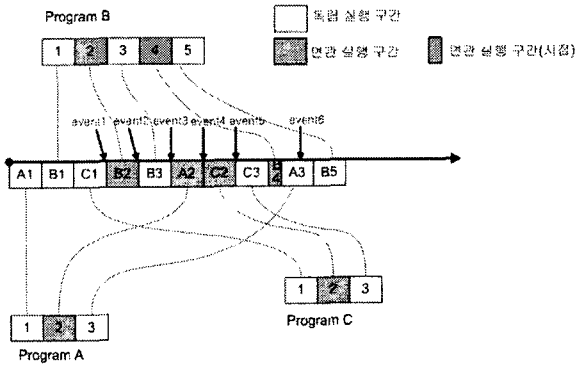

그림 2 실행 구간 구분 및 이벤트 발생 시점

표 2 이벤트 종류

이름	의미
motion_start	모션 명령의 실행
motion_end	모션 명령의 종료
set	전역 변수가 변경됨
wait_until	조건을 만족시킬 때까지 실행을 멈춤
enable_when	WHEN 구문이 활성화됨
disable_when	WHEN 구문이 비활성화 됨
when_end	WHEN 구문의 실행이 종료됨
remote_start	다른 로봇에 특정 프로그램을 실행시켜야 함
cycle_time	cycle이 종료되었음
end	프로그램의 실행이 종료되었음

이벤트 방식의 시뮬레이션에서는 로봇을 움직이는 것이 하나의 연관 실행 구간이 되어 실행된다. 이는 동시에 움직이는 로봇이 순차적으로 움직이는 형태로 시뮬레이션 되는 것을 의미한다. 동시에 움직이는 모든 로봇의 움직임 종료 후 정적 상황은 정확히 시뮬레이션 될 수 있지만 움직이는 도중에 발생하는 동적 상황에 대한 시뮬레이션은 불가능 하다. 따라서 실행 시간이 긴 실행 구간을 짧은 시간 단위로 나누고, 동시에 움직이는 모든 로봇이 나눠진 구간 단위로 동기화 된다면 동적 상황에서의 시뮬레이션이 가능하게 된다. 이러한 동기화 되는 구간이 하나의 사이클이다. VM이 하나 혹은 복수의 실행 구간의 실행 도중 사이클 종료 시 SE에 알려야 한다. 사이클 종료를 알리는 방법은 이벤트를 발생시키는 것이고 이 이벤트는 표 2의 cycle_time이다. SE는 모든 VM이 같은 사이클에 작동 될 수 있도록 이벤트를 관리하는 것으로 동적/정적인 상황의 시뮬레이션이 가능해진다.

3.2 로봇 시뮬레이션 엔진 설계

SE는 시뮬레이션 관리 모듈, 이벤트 관리모듈, WAIT관리 모듈, WHEN관리 모듈을 가지고 있다. 시뮬레이션 관리 모듈은 시뮬레이션 엔진의 메인으로 나머지 3개의 모듈을 이용하여 시뮬레이션을 진행한다. 이벤트 관리 모

듈은 이벤트 큐를 갖고 있고 이벤트를 저장하고 정렬하여 실행할 이벤트의 순서를 정해줌으로서 VM을 스케줄링 순서를 결정하는 역할을 한다. WAIT관리 모듈은 wait 리스트를 내부에 갖고 있어 현재 대기상태에 있는 VM을 저장하는 역할을 한다. WHEN관리 모듈은 내부에 when 리스트를 갖고 있어 현재 활성화 되어있는 WHEN문의 정보를 저장하는 역할을 한다.

SE가 시뮬레이션을 실행하는 알고리즘은 다음과 같다. 모든 VM을 실행 시켜 VM으로부터 이벤트를 받아 이벤트 큐에 저장한다. 가장 먼저 실행될 이벤트를 이벤트 관리 모듈로 부터 받아 VM이 이를 실행하도록 하고, 실행 결과를 받는다. 결과에 따라서 wait리스트, when리스트를 갱신, 사용하고 이는 다음 스케줄링에 반영된다. 이벤트의 처리 후 해당 VM에서 다음 이벤트를 받아 이벤트 큐에 저장하는 것으로 한 번의 이벤트 실행을 종료한다. 이후 이벤트의 선택, 실행, 다음 이벤트의 저장이 반복된다.

전역 변수의 변경을 알리는 set이벤트가 발생 할 경우 이 전역 변수와 관련된 when문을 갖고 있는 VM, 전역 변수의 값이 wait_until을 해제하는 조건인 VM은 when, wait_until에 대한 조건 검사를 해야 한다. 하지만 set이벤트를 발생 시킨 VM이외의 VM은 이에 대해 알지 못하기 때문에 조건 검사에 대한 이벤트를 발생시키지 못한다. 따라서 SE는 각 VM에 check_when, check_wait의 이벤트를 생성해서 이벤트 큐에 넣어 VM에 조건 검사 수행을 SE가 명령할 수 있도록 한다(그림 3). set이벤트 처리 즉시 VM에 명령을 내리지 않고 이벤트를 발생시키는 이유는 VM에 set이벤트 보다 먼저 실행되어야 하는 이벤트가 있을 경우, 해당 이벤트를 실행 시킨 후 조건 검사를 수행해야 하기 때문이다.

시뮬레이션이 진행되는 동안 시간의 경과가 있는 경우는 로봇의 움직임(motion_end)과 대기 상태(wait_until)의 경우만 존재한다. 이외의 실행 구간에서는 연산만 일어나기 때문에 시간의 경과가 일어나지 않는다. 사이클의 종료는 시간이 경과하는 도중에만 발생하기 때문에 motion_end, wait_until이벤트의 실행 도중에 일어나게 된다.

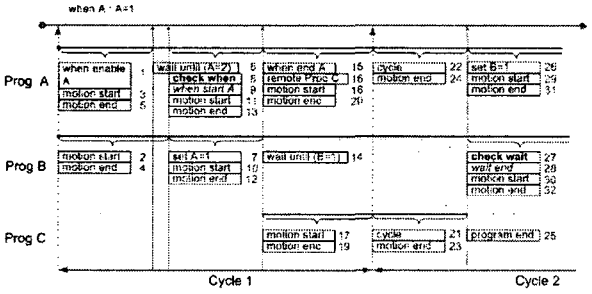

그림 3 이벤트 실행 순서

4. 구현

로봇 시뮬레이션을 위해 설계된 프로그래밍 언어(WSL)을 이용하여 프로그램을 작성, 전용 컴파일러를 이용하여 컴파일해 중간 코드를 생성한다. VWORKS에서 workcell을 로드하여 로봇과 장치를 배치하고, 각 로봇에 중간 코드를 매핑하여 SE가 VM이 해당 중간코드를 실행할 수 있도록 한다. 이후 SE는 시뮬레이션을 실행하고 VWORKS가 사이클마다 VM에 현재 로봇의 상태를 쿼리하여 화면을 갱신해서 움직임을 나타낸다.(그림 4).

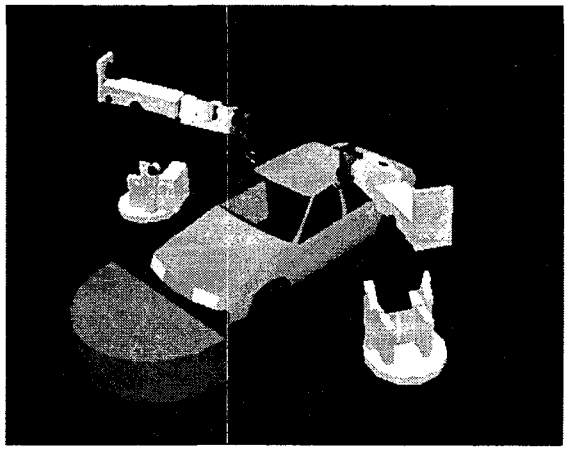

그림 4 시뮬레이션 실행 화면

5. 결론

OLP를 기반으로 한 시뮬레이션 엔진의 구현에 이벤트와 사이클 혼합 방식을 사용하였다. 복수의 로봇을 동시에 시뮬레이션 하는 방식에 이벤트를 사용함으로서 로봇의 순차적인 실행을 동시에 실행하는 것과 같은 시뮬레이션 결과를 갖게 하였다. 또한 짧은 시간의 사이클을 두어 사이클 단위의 실행을 하여 이벤트 방식 하에서 로봇이 움직이는 동적인 상황에서의 시뮬레이션을 수행 할 수 있게 하였다.

6. 참고문헌

[1]박찬진, "이벤트/사이클 혼합 방식을 통한 분산시뮬레이션의 성능향상", 부산대학교 대학원 컴퓨터공학과 석사학위논문, 2006

[2]Eberhard Roos, Arno Behrens, "Off-line programming of industrial robots - Adaptation of simulated user programs to the real environment", Computers in Industry, Vol.33, pp.139-150, 1997

[3]Couretas, J.M. and J.W. Rozenblit, "Generation, Control, and Simulation of Task Level Actions Based on Discrete Event Models", Proceedings of the 5th Annual Conference on AI, Simulation and Planning in High Autonomy Systems, IEEE Computer Society Press, pp.214-220, 1994.

[4]Stefan Anton, Thomas Fries, Thomas Horsch, Friedrich Wilhelm Schoer, Cornelius Willnow, Christian Wolf, "A Framework for Realistic Robot Simulation and Visualisation", www.easy-rob.com/data/ Frame-RRS.pdf