전력전자학술대회 논문집 2000. 7.19-22

촉각 제어 시스템을 위한 제어용 인터럽트 타이머의 구현

박진석*, 김대현*, 김영동**
*조선대학교 제어계측공학과 **조선대학교 전기·제어계측공학부

Implementation of a Controllable Interrupt Timer for Haptic Control System.

Jin-Suk Park*, Dae-Hyun Kim*, Young-Dong Kim**
*Dept.of Control and Instrumentation, Graduate school of Chosun University
**Dept. of Control and Instrumentation, Chosun University

ABSTRACT

In this paper, we propose a controllable interrupt timer for haptic control system. haptic control system, which was divided into two processes as virtual environment(VE) manager and haptic controller The VE manager displays the 3D graphic scene at low update rates 25Hz and haptic controller controls the haptic display at high update rates 1000Hz. To archive the accurate update rate, we have imple- mented a timer so called "AccTimer" based on Windows® multimedia functions. The proposed "Acc Timer" for haptic control system has been imple- mented in a personal computer with a 6-DOF haptic interface. Experimental results show that our system is robust with respect to tolerances in the control rates, and also, through the accurate control rate the operator can always feel a stable force.

1. 서론

최근 컴퓨터와 인터넷의 발전은 가상현실을 비롯하여 컴퓨터 시스템과 사용자간의 상호관계에 혁명을 가져왔다. 대부분 가상현실 시스템은 GUI와 같은 3차원 그래픽과 3차원 음향으로 구성되어지지만, 현실감 있는 가상 환경의 개발을 위해서는 촉각 표현이 필수적이다. 현재 대부분 응용분야에서 그래픽 시뮬레이션과 촉각 표현을 각기 다른 컴퓨터에서 구현하고 있다[1,2]. 이와 같은 방법은 연속적인 동작의 표현을 위해 초당 25프레임의 그래픽 갱신과 현실과 같은 촉각을 표현하기 위한 1000Hz 이상의 제어를 실행하기 때문이다[3,4]. 따라서, 3차원 그래픽 시뮬레이션과 촉각 제어부를 분리한 촉각 제어 시스템을 구현할 때 촉각 루프는 실시간 시스템에서 실행시키며, 계산량이 많은 시뮬레이션 루프는 고가의 워크스테이션에서 실행시키고 있다[5,6]. 윈도우 플랫폼에서 촉각 제어 시스템을 구현하면 3차원 그래픽 데이터 처리에 시스템의 부하가 많이 걸려 촉각 루프의 갱신을 위한 인터럽트 타이머의 요청이 무시되며, 3차원 그래픽 화면의 갱신 주기도 매우 낮아진다. 이는 Windows의 타이머가 하드웨어 타이머 인터럽트에 기반을 두고는 있지만 타이머 호출이 응용 프로그램에서 매우 낮은 우선권을 갖고 있으며 비동기 적으로 작동되기 때문이다[7]. 그러므로, 촉각 제어부가 1000Hz의 갱신 주기를 가지더라도 윈도우 플랫폼에선 정확한 샘플링 주기를 보장해 줄수가 없다.

본 논문은 GUI 환경인 Windows 플랫폼에서 일반적인 타이머 함수 대신에 멀티미디어 제어함수에 기반을 둔 *"AccTimer"* 라는 제어용 타이머를 제안하며 이를 촉각 제어 시스템에 활용하였다. 구현한 촉각 제어 시스템은 이러한 타이머를 이용하여 한 대의 PC에서 VE 매니저와 촉각 제어부로 나누어 동작킬수 있었다.

2. 촉각 제어 시스템

촉각 제어 시스템은 그림 1과 같이 VE 매니저와 촉각 제어부로 분리된다.

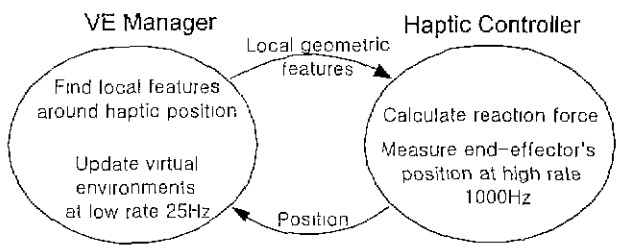


그림 1. VE 매니저와 촉각 제어부로 분리한 시스템

2.1 VE 매니저

VE 매니저는 DirectX 모델링 환경하에서 동작하며 그래픽 시뮬레이션을 담당한다 매 사이클마다 촉각 장치의 말단부 위치가 갱신되면, VE 매니저는 시뮬레이션 화면을 갱신하며, 모든 물체의 충돌을 체크한다. 이런 과정후 VE 매니저는 물체의 번호나 꼭지점 정보, 구속평면등의 표면정보를 촉각

제어부에 보낸다. 다음은 그림 2에 보여진 준비된 평면과의 충돌 처리진행을 보여준다.

Step 1 : 촉각 장치의 말단부의 위치를 중심점으로 일정반경의 구안에 존재하는 물체를 찾는다.

Step 2 : Step 1에 의해 찾아낸 각 물체의 표면에서 촉각 장치의 말단부와의 최근접점을 찾는다

Step 3 : Step 1에 의해 찾아낸 각 물체에서 Step 2에 의해 찾아낸 최근접점을 통과하는 평면을 찾는다

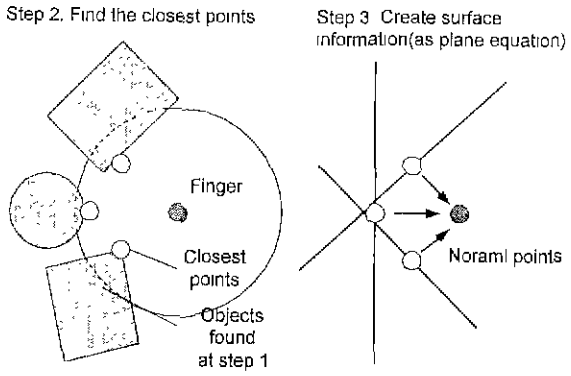


그림 2. 준비된 평면의 처리 구조

2.2 촉각 제어부

6 자유도 촉각 장치는 200mm×200mm×200mm에 이르는 안정된 작업 공간과 높은 출력 토크를 낼 수 있도록 설계되었다. 350MHz 펜티엄II PC는 촉각 장치에 제시할 힘을 계산하며, 인터럽트 타이머는 1000Hz로 동작시켰다. 사용된 엑추에이터는 브러시리스 DC 모터이며, 케이블 전달방식을 이용한 토크 증폭을 하였다.

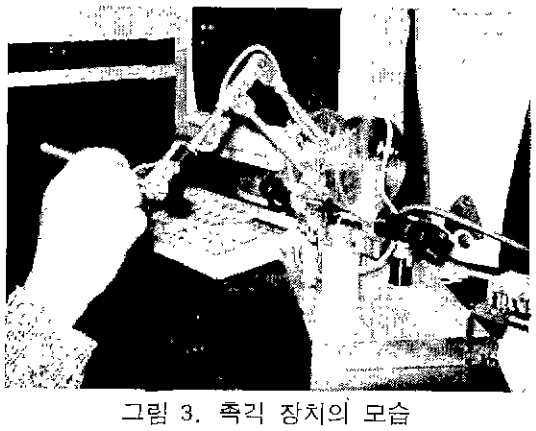


그림 3. 촉각 장치의 모습

3. 제어용 인터럽트 타이머

3.1 타이머의 문제점

가상 환경에서 물체의 연속적인 움직임을 위한 그래픽 시뮬레이션의 갱신 주기는 25Hz 정도이다. 촉각 제어부도 같은 주기로 갱신된다면 물체의 움직임은 거칠고 고르지 못한 느낌을 준다. 현실과 같은 촉각을 표현하기 위해선 촉각 제어부가 물체의 움직임과 힘의 상호작용을 고려하면서도 1000 Hz 이상의 빠른 갱신 주기를 가지고 실행되어야 한다. 이러한 갱신 주기는 바로 인터럽트 타이머에 의해서 처리되고 있다.

윈도우 운영 체제는 인터럽트 타이머에 대해 세가지 방법을 제공한다.

첫 번째 방법은 *"SetTimer"*를 사용하는 것이다. *"Set Timer"*는 타이머 ID와 타이머 호출 간격을 가지고 있으며 *WM_TIMER* 메시지를 *"OnTimer"* 함수에 연결하거나 응용 프로그램 차원에서 정의된 콜백 함수를 호출한다. 이 방법들에 의한 타이머 호출은 응용 프로그램에서 매우 낮은 우선권을 갖게 된다. 따라서, 타이머는 메시지 큐에 다른 메시지들이 없을 때에만 처리된다. 만일 촉각 제어부가 매 1ms마다 타이머 호출을 설정했더라도 VE 매니저가 그래픽 시뮬레이션 처리로 인해 바쁘다면 1초에 천번에 이르는 타이머 호출을 받지 못한다.

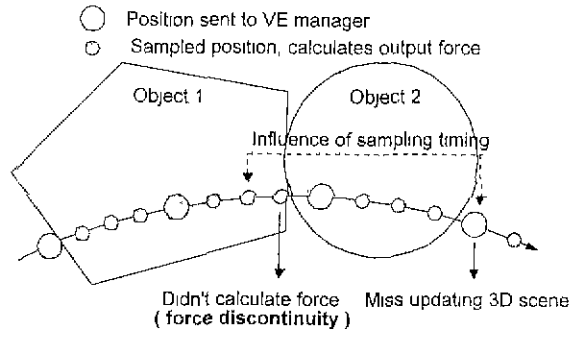


그림 4. 잘못된 샘플링 타임의 영향

다른 방법은 멀티 스레드를 사용하는 것이다. 멀티 스레드는 *WM_TIMER*에 바탕을 두고 있으며, 운영체제가 동시에 복수의 프로그램을 돌릴수 있는 능력을 말한다. 멀티 스레드 방식의 단점중 하나가 *"race condition"*인데, 이것은 한 스레드가 다른 스레드의 데이터를 참조하면서 동작하다 데이터를 훼손하여 다른 스레드가 멈춰버리는 현상이다. 이런 문제점은 *"critical sections"*으로 해결할 수 있으나 *"deadlock"*이라 불리우는 또다른 문제점을 가진다.

3.2. 타이머 디자인

샘플링 주기가 중요한 응용 프로그램에서 윈도우의 일반 타이머 메시지나 멀티 스레드는 사용시 그 주기를 보장할 수 없다. 촉각 제어부도 샘플링 주기가 중요하므로 윈도우 일반 타이머 메시지로는 불충분하다. 한 대의 PC에서 VE 매니저와 촉각 제어부를 정확히 제어하기 위해 몇가지 함수의 사용만으로 높은 해상력의 타이머를 구현하는 멀티미디어 API를 사용하며 그 알고리즘은 아래와 같다.

Algorithmically-defined timer

If timer is running system
  Set thread ID and resolution
  Create thread with thread callback function
  Map callback function to VE manager
ELSE thread is running in system
    Release thread and callback pointer
    Set timer ID, resolution and interval
    Store system elapse time
    Determine optimal resolution factor
    Begin timer period
    Set time to event with callback function

타이머는 MFC 기반으로 만들며, 실제로 구현해서 사용한 예제 코드는 아래와 같다.

```
CAccTimer accTimer;

accTimer.Set(TimerCallbackFunc, this);
accTimer.Resolution(HAPTIC_PERIOD);
accTimer.Interval(HAPTIC_PERIOD);
accTimer.Thread();
VEManager.period = VE_MAN_PERIOD;
sDevice = new CHapticController;
sDevice->DeviceInitialize();
VEManager.AddHapticDev(sDevice);

void CShinGIRUView::TimerCallbackFunc(void* arg)
{
  cShinGIRUView *view = (CShinGIRUView*_)arg;
  view->VEManager.HapticStep(HAPTIC_PERIOD);
}
```

3.3 VE 매니저와 촉각 제어부의 상호 통신

각각의 사이클마다 말단부 위치가 VE 매니저에 보내지면 VE 매니저는 촉각 제어부에 보내질 평면을 준비한다. 말단부 위치와 평면과의 충돌이 검출되면 촉각 제어부는 힘을 계산하며 다시 VE 매니저는 힘 정보와 물체의 속도를 가지고 VE 매니저는 그래픽 화면을 갱신한다. 처리 알고리즘은 아래와 같다.

Notions:

Δt	: the period of haptic controller updates
$t = k \cdot \Delta t$	: the period of VE Manager updates
v_i	: velocity of the object
ω_i, m	: angular velocity and mass of the object$_i$
I_i	: the inertia matrix of the object$_i$
$P_i(t), N_i(t)$	: the position and normal of the plane
F_i	: the force from an operatior to object
$f_i(t)$	: the force from the plane to the operator

Producer of the VE Manager

Receive $F_i, C(nT)$

Update each object's velocity

$$v_i(nT) = v_i((n-1)T) + (1/m) \cdot F_i$$

$$\omega_i(nT) = \omega_i((n-1)T) + I^{-1}(C(nT) - P_i((n-1)T) \times F_i$$

Update the position of all objects.

Find closet point of the objeti : $P_i(nT)$

Calculate normal for each object :

$$N_i(nT) = C(nT) - P_i(nT) / |C(nT) - P_i(nT)|$$

Send $P_i(nT)$ and $N_i(nT)$ to haptic controller

Producer of the haptic controller

Receive $P_i(nT)$ and $N_i(nT)$

For $t = nT$ to $t = (n+1)T - \Delta t$

Get the position of the end-effecter : $C(t)$

Interpolate planes :

$$P_i(t) = (t - nT)/T \cdot P_i(nT) + (1 - (t - nT)/T) \cdot P_i((n-1)T)$$

$$N_i(t) = (t - nT)/T \cdot N_i(nT) + (1 - (t - nT)/T) \cdot N_i((n-1)T)$$

Calculate force :

If $((C(t) - P_i(t)), N_i(t))N(t) > 0$ then

$f_i(t) = ((C(t) - P_i(t)), N_i(t))N(t)$

Else $f_i(t) = 0$

If $t = nT$ then $F_i(t) = f_i(t)$

$t = t + \Delta t$

Send F_i to the haptic device.

4. 실험 및 고찰

4.1. 실험 시스템

그림 5는 실험에 사용된 촉각 제어 시스템[8]의 구조이다.

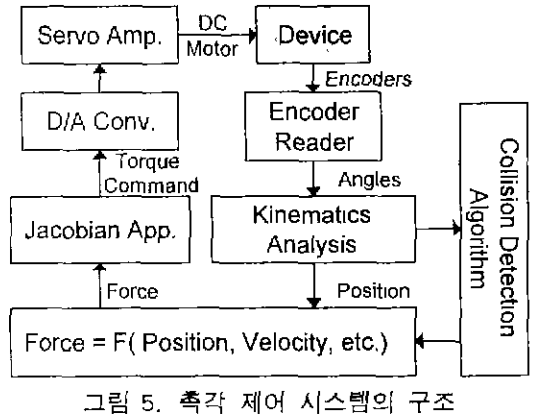


그림 5. 촉각 제어 시스템의 구조

엔코더에 의해 측정된 관절의 위치는 24비트 카운터 칩을 통하여 얻어진다. 전류 증폭을 위한 서보 값들은 12비트 D/A 컨버터를 통해 정확하게 값으로 제어되고 있다. 타이머의 정확성을 검증하고자 그림 6과 같은 가상 피즐 실험에서 세 가지 방식을 이용하여 제어 타이밍 시그날을 측정하였다.

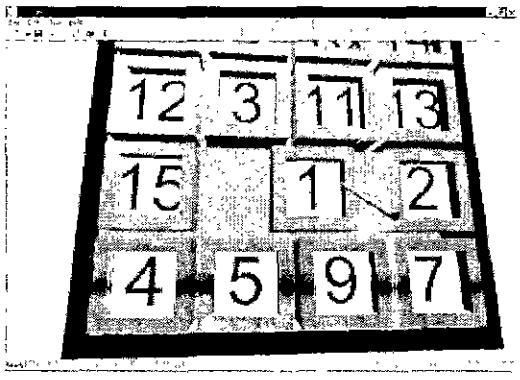
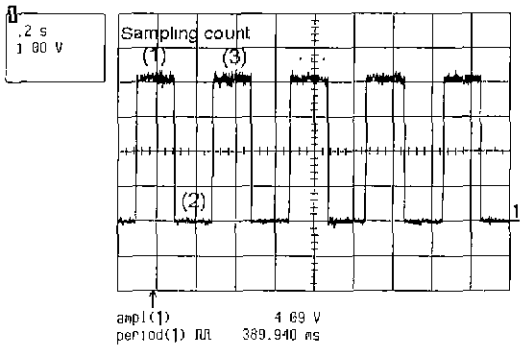


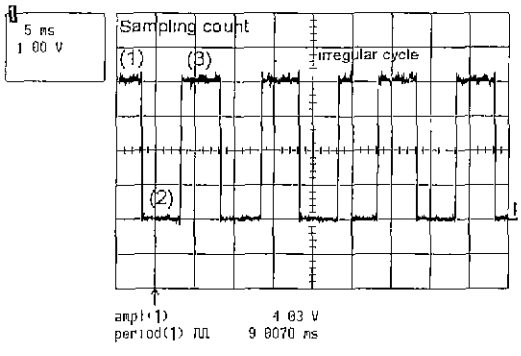
그림 6. 가상 퍼즐 실험

4.2 실험 결과

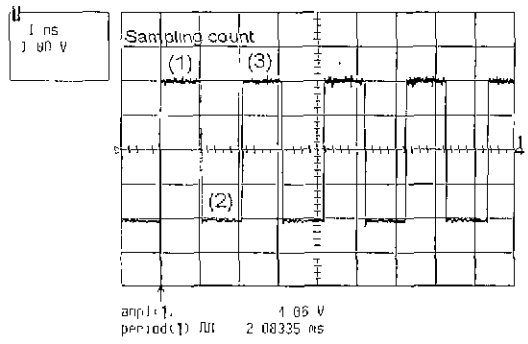
그림 7(a)는 일반적인 타이머 메시지를 사용한 촉각 제어부의 갱신 주기이다. 측정된 주파수는 5.2Hz 였다. 이 경우 말단부 위치가 불규칙하게 나타나며 시스템이 통제가 힘든 발진이 일어나며 불안정성이 증가하였다. 그림 7(b)은 멀티 스레드 방법을 사용한 촉각 제어부의 갱신 주기이다. 그림 7(a)와 비교시 샘플링 주파수가 222Hz로 증가하였으나 가끔씩 불규칙한 주파수를 볼수가 있었다. 힘의 장애는 보통 타이머 메시지를 이용한 방법보다는 높아졌으나 안정된 힘의 제시는 힘들었다. 그림 7(c)는 제안한 *"AccTimer"* 타이머을 사용한 결과이다. 측정된 샘플링 주파수는 960Hz 로서 원했던 1Khz는 아니지만 안정된 힘제시를 하는데 충분하였다.



(a) 일반 타이머 메시지를 사용한 경우



(b) Multithread를 사용한 경우



(c) "AccTimer"를 사용한 경우

그림 7. 실험 결과

5. 결론

본 논문에서는 정확한 제어 타이밍을 위하여 윈도우 멀티미디어 함수를 이용한 제어용 타이머 "AccTimer"를 제안하였다. 제안한 "AccTimer"를 가지고 VE 매니저와 촉각 제어부로 분리된 촉각 제어 시스템에서 가상 퍼즐 실험을 수행하였다. 비록 측정된 주파수는 960Hz로서 정확한 1KHz는 아니었지만 실험을 수행하는 데는 충분하였다. 타이머 메시지를 이용한 경우는 5.2Hz 였으며, 멀티 스레드를 이용한 경우는 222Hz 였으며 이 둘과 비교시 제안된 타이머는 제어-타이밍 문제가 분명히 개선되었음을 보여주며 촉각제어에 효과적임을 증명하였다. 또한 제안된 타이머는 타이머 인터럽트가 필요한 응용 프로그램에서 효과적으로 사용될 수 있을거라 기대된다.

참 고 문 헌

[1] Diego. Ruspini, Krasimir Kolarov and Oussama Khatib, "Haptic Interaction in Virtual Environments",Int.Conf.onIEEE IntelligentRobots Systems, 1997.

[2] S.Vedula and D.Baraff, "Force Feedback in Interactive Dynamic Simulation", Proc. of First PHANToM User's Group workshop, 1996.

[3] Lonnie Love and Wayne Book, "Contact Stability Analysis of Virtual Wall", Proc. of Dynamic Systems and Control Division, ASME pp.689-694, 1995.

[4] Beeling Chang and J.Enward Colgate, "Real-Time Impulse-Based Simulation for Haptic Display", Proc of ASME International Mechnical Engineering Congress and Exhibition, 1997

[5] Mark. W. R., Randolph, S. C., Finch, M.,Van Verth, J.M , and Talor III, R.M., "Adding Force Feedback To Graphics Systems : Issue and Solutions", Proc. of SIGGRAPH 96, pp.447-452, 1996.

[6] J E.Colgate, et. Al. "Implementation of Stiff Virtual Walls in Forcre-Reflecting Interfaces", Proc.of IEEE VR Annual Int. Sysmposium, pp.203-210, 1995

[7] Charles Petzold, "Programming Windows", Fifth Edition, Microsoft Express, 1999.

[8] D.H.Kim, J.S.Park, and Y.D.Kim, "A Personal computer Based Force-Reflecting system : Design and Application", Proc. of the 3rd International Workshop on Advanced Mechatronics, Chunchon Korea, pp.131-135, December, 1999.