Journal of Korea Multimedia Society Vol. 19, No. 10, October 2016(pp. 1759-1766)
http://dx.doi.org/10.9717/kmms.2016.19.10.1759

임베디드 환경에서 실시간 가상 터치 인식 시스템의 구현

권순각[†], 이동석[††]

Implementation of Real-time Virtual Touch Recognition System in Embedded System

Soon-Kak Kwon[†], Dong-Seok Lee[††]

ABSTRACT

We can implement the virtual touch recognition system by mounting the virtual touch algorithm into an embedded device connected to a depth camera. Since the computing performance is limited in embedded system, the real-time processing of recognizing the virtual touch is difficult when the resolution of the depth image is large. In order to resolve the problem, this paper improves the algorithms of binarization and labeling that occupy a lot of time in all processing of virtual touch recognition. It processes the binarization and labeling in only necessary regions rather than all of the picture. By appling the proposed algorithm, the system can recognize the virtual touch in real-time as about 31ms per a frame in the depth image that has 640×480 resolution.

Key words: Embedded System, Depth Information, Real-time Virtual Touch

1. 서 론

최근 아이폰의 성공으로 인해 다양한 분야의 기기에 터치 인터페이스를 접목해 나가고 있는 추세이다. 하지만 현재 터치 인식 기술은 스크린 크기에 비례하여 가격이 증가하는 단점이 존재한다. 이러한 단점을 극복하기 위해 깊이 영상을 촬영하여 터치 여부를 확인하는 가상 터치 방법을 적용할 수 있다[1]. 가상 터치 방법에서 기존 터치 방법과 제일 차별화 되는 요소는 이동성이다. 기존에 제일 많이 쓰이는 정전식 터치 인식 방법[2]은 스크린과 터치 인식 장치가 결합되어 있어 중대형 스크린 환경에서는 이동성이 저하되는 문제가 있다. 하지만 가상 터치 방법은 스크린의 크기에 상관없이 가상 터치 인식 기기의 이동성이 높으면 어디에서든지 손쉽게 설치를 할 수 있다. 이를 위해 임베디드 환경에서 가상 터치 인식 시스템을 구현한다.

이 때 가상 터치 인식에서 제일 중요한 요소 중 하나는 실시간으로 가상 터치를 처리하는 것이다. 여기서 실시간의 정의는 입력 값에 대해 정해진 시간 내에 정확한 반응을 하는 정보처리 행위 또는 시스템을 뜻한다[3-4]. 가상 터치 시스템에서 실시간성을 만족시키기 위해서는 깊이 영상을 촬영함에 있어서 각 프레임 촬영 간격 시간 내에 터치에 대해 반응을

※ Corresponding Author : Soon-Kak Kwon, Address: (47340) Eomgang-ro 176, Busanjin-gu, Busan, Korea, TEL : +82-51-890-1727, FAX : +82-51-890-2629, E-mail : skkwon@deu.ac.kr
Receipt date : Sep. 19, 2016, Revision date : Oct. 10, 2016
Approval date : Oct. 11, 2016

[†] Dept. of Computer Software Engineering, Dongeui University
[††] Dept. of Computer Software Engineering, Dongeui University (E-mail : 14177@deu.ac.kr)
※ This work (Grants No. C0396237) was supported by Business for Cooperative R&D between Industry, Academy, and Research Institute funded Korea Small and Medium Business Administration in 2016.

해야 한다. 임베디드 환경에서는 프로세서의 성능에 제약이 있기 때문에 실시간성이 더 중요한 요소가 된다.

실시간 시스템을 구현하는 방법은 크게 시스템의 작업들을 분배하는 스케쥴링 정책을 실시간 시스템에 알맞게 변경하는 방법과 시스템의 반응 속도를 개선시키는 방법으로 나눌 수 있다. 실시간 시스템의 스케쥴링은 시스템 내의 여러 작업들에 대해 프로세서 점유 등의 시스템 자원을 입력에 대해 정해진 시간 내에 반응을 얻기 위해 작업들을 시스템에 정해진 방식에 따라 우선순위를 두어 분배를 하는 것이다[4-5]. 실시간 스케줄링은 응답시간이 치명적인 요소가 되는 임베디드 시스템에 주로 쓰이고 있다. 실시간 스케쥴링 방법을 적용하는 방법은 여러 작업을 병렬적으로 수행하거나 입력을 받기 위해 대기 중인 상태에서 다양한 입력에 대해 각각의 반응을 해야 하는 시스템에 대해서는 효율적으로 실시간성을 만족시킬 수 있지만, 단일 작업 내의 알고리즘이 복잡할 경우에는 실시간 스케쥴링 방법을 적용하여도 실시간성을 만족시킬 수 없는 한계점이 있다.

실시간 시스템을 구현하기 위해 시스템의 반응 속도를 개선시키는 방법은 다수의 프로세서들 또는 프로세스 코어들 및 GPU를 이용한 병렬처리 방법과 시스템 내의 알고리즘을 개선하여 반응 속도를 증가시키는 방법으로 나눌 수 있다. 병렬 처리는 다중 프로세서 환경이나 멀티 코어 프로세서에 대해 시스템 또는 모듈 내 분할 가능한 작업들에 대해 각각의 프로세서 또는 코어에 할당시켜 처리하는 것을 뜻한다. 효율적인 병렬 처리를 적용한 시스템은 시스템 자원을 효율적으로 이용하기 때문에 처리 속도가 빨라진다. 하지만 병렬 처리를 적용하기 위해서는 임계 구역에 대한 상호 배제를 만족시키기 위해 신중하게 접근해야 한다. 만약 임계구역에 대해 주의를 기울이지 않는다면 예상하지 못한 결과가 나오는 문제가 발생한다. 알고리즘을 개선하는 방법은 범용적인 알고리즘을 적용한 부분에 대해 시스템의 특성을 이용하여 알고리즘 내 필요 없는 요소를 제거하거나 최적화하여 알고리즘의 복잡도를 낮추어 알고리즘의 수행 시간을 줄이는 방법이다. 시스템에 적용된 알고리즘을 최적화함으로써 수행 시간을 효율적으로 줄일 수 있다.

영상 처리 분야에서 실시간 시스템에 대한 연구는 다음과 같다. 먼저 작업을 분할하여 스케줄링을 통해 실시간 시스템을 구현하는 연구가 이루어졌다[6-7]. 또한 임베디드 기술의 발달로 인해 임베디드 기기에 GPU가 장착되는 경우도 있는데, 이를 이용하여 임베디드 기기에 장착된 GPU를 이용하여 병렬 처리를 통해 실시간 영상처리를 하거나[8-9], H.264 형식의 영상을 병렬 처리를 통한 실시간 스트리밍 방법[10], UHD 영상의 HEVC 방식 부호화를 다중의 CPU 및 GPU를 통해 병렬 처리 하는 방법[11], GPU를 이용한 병렬처리를 통해 로봇에서 촬영한 영상을 통해 사람의 상대위치를 추정하는 연구[12] 등 GPU를 이용한 병렬 영상처리 방법이 최근에 연구되었다. 이처럼 병렬 처리를 통해서는 영상을 실시간으로 처리하는 분야의 연구가 대다수였다. 그밖에 실시간 스케줄링과 병렬 처리를 같이 적용하여 실시간 시스템을 구현하는 연구[13]도 이루어졌다. 이처럼 실시간 영상 처리시스템에 대한 연구가 많이 이루어져 왔지만 임베디드 환경에서의 실시간 깊이 영상 처리에 대한 연구는 부족하다.

본 논문에서는 가상 터치 인식 알고리즘의 개선을 통해 임베디드 환경에서의 실시간 가상 터치 인식 기기를 구현한다. 먼저 제2장에서 임베디드 환경에서의 가상 터치 인식 시스템을 구현하고, 가상 터치 인식 시스템 내에서 처리 시간에 많은 영향을 끼치는 단계들을 찾는다. 그 후 제3장에서 그 단계들에 대해 알고리즘을 분석하고 가상 터치에서 나타나는 깊이 영상의 특성에 맞게 알고리즘을 개선한다. 마지막으로 제4장에서 실험을 통해 개선된 알고리즘을 통해 실시간 가상 터치 인식이 가능하다는 것을 보이고, 제5장에서 연구에 대한 결론을 내린다.

2. 임베디드 환경에서 가상 터치 인식 시스템의 구현

가상 터치는 깊이 카메라를 통해 깊이 정보를 획득 한 뒤, 깊이 정보를 통해 배경 영상과 촬영 영상을 획득하여 이를 통해 객체의 터치를 인식하는 방법이다[14]. 이러한 가상 터치 인식 흐름도는 Fig. 1과 같다.

이러한 가상 터치 인식 시스템은 기존 PC환경에서도 작동을 할 수 있지만, 가상 터치를 범용적으로 사용하기 위해서는 이동성이 좋은 임베디드 기기 환

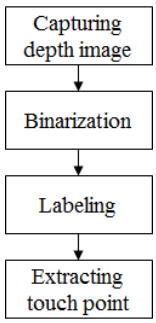

Fig. 1. Flowchart of the existing method of touch detection.

경에서 구현하는 것이 바람직하다. 이 때 가상 터치 인식 시스템을 Raspberry Pi 등의 깊이 카메라를 장착할 수 있는 임베디드 기기에 탑재를 하여 가상 터치 인식 시스템을 구현할 수 있다[10]. 임베디드 기기는 대부분 Linux를 탑재하여 동작하기 때문에 해당 임베디드 환경에서 사용할 수 있는 이식도구를 이용하면 기존 PC환경에서 구동하는 시스템을 임베디드 환경으로 이식할 수 있다. 이를 통해 임베디드 환경에서의 가상 터치 인식 시스템을 구현할 수 있다.

또한 임베디드 기기에서 제어 대상인 클라이언트 기기로 터치 메세지를 전달하여 메세지에 맞는 터치 이벤트를 수행해야 한다. 임베디드 기기에서 클라이언트 기기로 메세지를 전달하는 방법은 여러 가지가 있지만, 본 논문에서는 UART 통신 방법을 이용하여 임베디드 기기와 클라이언트 기기간 통신을 수행하였다. Fig. 2은 이러한 방법을 이용하여 임베디드 환경에서 구현된 가상 터치 인식 시스템의 구성도이다.

가상 터치 인식 시스템에서 제일 중요한 것은 깊이 영상을 촬영하여 실시간으로 가상 터치를 처리할 수 있는지의 여부이다. 실시간으로 가상 터치를 인식하기 위해서는 깊이 카메라가 깊이 영상을 촬영할 때, 각 프레임 간 촬영 간격내로 촬영된 영상을 처리

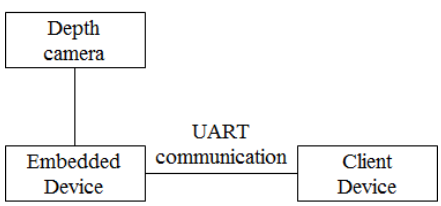

Fig. 2. Diagram of the virtual touch system in embedded system.

하여 가상 터치를 인식하여야 실시간으로 가상 터치를 인식할 수 있다. 만약 처리시간이 프레임 간 촬영 간격 이상일 경우에는 그 사이에 읽혀진 프레임의 깊이 영상이 처리가 되지 않고 다음 프레임의 촬영 영상을 읽게 되어 실시간 가상 터치를 처리할 수 없다. 이는 터치 인식 속도가 느려지는 원인이 되어 사용자가 원활하게 터치 인터페이스를 제어하지 못하는 요인이 된다. 따라서 실시간으로 가상 터치를 처리하는 것은 가상 터치 인식 시스템에서 제일 중요한 요소이다.

PC 환경에서는 장착된 프로세서의 초당 연산횟수가 충분히 크기 때문에 원활하게 실시간 가상 터치를 지원할 수 있다. 하지만 임베디드 환경에서는 PC환경에 비해 장착된 프로세서의 성능이 떨어지기 때문에 임베디드 기기 환경에서 실시간 가상 터치를 지원하기 위해서는 PC 환경에서보다 더욱 가상 터치 처리 시간에 주의를 기울여야 한다.

Fig. 3은 PC와 임베디드 기기에서의 깊이 영상 해상도 별 처리 속도를 나타낸 그래프이다. 이 때 처리 속도는 깊이 영상의 해상도에 비례한다. 기존 PC 환경에서는 해상도가 높아져도 처리 시간이 빠르기 때문에 프레임 간 간격 내에 충분히 가상 터치 인식 알고리즘을 처리할 수 있다. 하지만 임베디드 환경에서는 해상도가 증가함에 따라 가상 터치 처리 시간이 프레임 간 간격보다 크게 되어 실시간 가상 터치 인식이 불가능하다.

가상 터치 인식을 하는 과정은 Fig. 1에서와 같이 깊이 영상을 획득하는 단계, 이진화 단계, 라벨링 단계, 터치점 추출 단계로 나눌 수 있다. 하지만 깊이

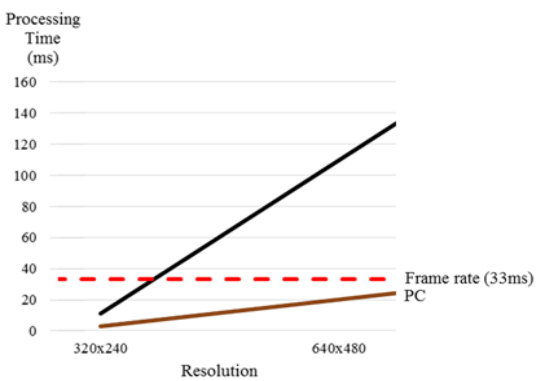

Fig. 3. Relationship between the resolution of the depth image and the processing time of recognizing the virtual touch.

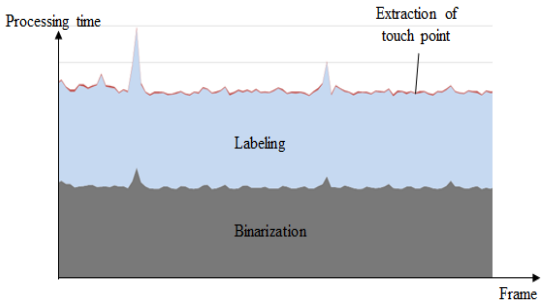

Fig. 4. Processing time of each step of the recognizing the virtual touch.

영상을 획득하는 부분은 단순히 깊이 카메라의 버퍼에 담겨진 최신의 깊이 영상을 읽어오는 단계이기 때문에 처리 시간에 영향을 끼치지 않는다고 봐도 무방하다. Fig. 4는 이진화 단계, 라벨링 단계, 터치점 추출 단계에서 각각의 처리 시간을 측정한 그래프이다. 여기서 가상 터치 인식을 처리하는 과정 중에 라벨링과 이진화 단계가 제일 큰 비중을 차지하고, 터치 점 인식 단계는 앞 두 단계에 비해 비중이 낮다는 것을 알 수 있다. 따라서 임베디드 기기에서 실시간으로 가상 터치를 인식하는 시스템을 구현하기 위해서는 이진화와 라벨링 알고리즘을 가상 터치 인식에서 나타나는 깊이 영상의 특성에 맞게 알고리즘을 개선하는 것이 필수적이다.

3. 임베디드 환경에서 실시간 가상 터치를 위한 알고리즘 개선 방법

가상 터치 처리 시간은 깊이 영상의 해상도와 비례하는 관계에 있다. 따라서 임베디드 환경에서 실시간 가상 터치를 구현하기 위해서는 적절한 깊이 영상 해상도에서 구동하는 것이 중요하다. 이 때 가상 터치 시스템에서의 알고리즘을 개선하여 처리 시간을 줄임으로써 더 높은 해상도에서도 실시간 가상 터치를 지원할 수 있다. 이는 사용자에게 원활한 터치 인터페이스를 제공해 줄 뿐만 아니라, 더 높은 깊이 영상 해상도를 지원함으로써 가상 터치 정확도를 더 높일 수 있다. 가상 터치 시스템에서 Fig. 4와 같이 이진화, 라벨링 단계들이 처리 시간에서 거의 대부분을 차지한다. 본 논문에서는 이러한 이진화, 라벨링 단계를 가상 터치 시스템에 맞게 개선하는 방법을 제안한다.

기존의 이진화 단계에서는 전 영역을 이진화하는 과정을 거쳤다. 하지만 드래그를 하기 위해서 터치를 계속 유지하고 있을 때는 기존 터치점에서 다음 터치점까지의 이동 거리가 제한된다. 이를 이용하여 이전 터치점을 중심으로 다음 터치를 하기 위해 최대로 갈 수 있는 거리의 범위 내의 영상만을 이진화 한다. 이 때 영상 내에서 터치점 간 최대 허용 픽셀 거리가 수평 방향으로 T_x, 수직방향으로 T_y이고, 현재 t 프레임에서 촬영된 영상을 I_t, 이진화 후 영상을 B_t, 이전 터치 지점을 P_{t-1}라 할 때, 제안하는 알고리즘은 Fig. 5와 같이 나타낼 수 있다. 제안된 알고리즘을 적용한 결과 Fig. 6처럼 실제 터치가 가능한 지점만 이진화

```
function Binarization(I_t : Image) return B_t : Image
is
begin
    if P_{t-1} is exist
        SearchArea = (PX_{t-1} - T_x ... PX_{t-1} + T_x, PY_{t-1}
        - T_y ... PY_{t-1} + T_y)
    else
        SearchArea = All Area of I_t
    end if
    for each pixel (x, y) in SearchArea
        if I_t(x, y) > threshold
            B_t(x, y) = 1
        else
            B_t(x, y) = 0
        end if
    end for
    return B_t
end function
```

Fig. 5. Proposed binarization algorithm for the recognizing the virtual touch.

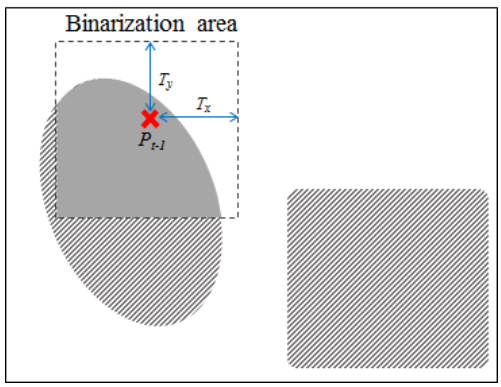

Fig. 6. Result of appling proposed binarization algorithm.

를 하게 된다.

또한 기존의 라벨링 단계에서는 전체 영역에 대하여 라벨링을 수행하였다. 하지만 가상 터치를 수행할 때, 터치를 수행하기 위해 영상을 촬영하는 과정에서 영상의 중앙부에 터치 대상 스크린이 있기 때문에 객체도 영상의 중앙에 가까운 부분에서 촬영될 확률이 높다. 따라서 라벨링을 수행하기 위해 화소를 탐색할 때, 객체가 시작하는 점을 알 수 있다면 라벨링을 수행하기 위해 검색해야 할 화소의 수를 반 이상 줄일 수 있다. 이를 위해 라벨링을 수행하기 전 단계인 이진화 단계에서 첫 객체가 출연하는 위치 정보 bs_t를 라벨링 단계에 전달하여 효율적인 라벨링을 수행하도록 한다. 이를 위해 이진화 알고리즘을 Fig. 7처럼 변경한다.

또한 라벨링에서 객체를 검사하는 중에 객체 크기가 S보다 크면, 잡음이 아니라고 판단하고 라벨링 수행을 중단 후에 터치점 검출 단계로 넘어간다. 이러한 라벨링 알고리즘은 Fig. 8과 같이 나타날 수 있다. 제안된 알고리즘을 적용한 결과 Fig. 9처럼 터치 확인에 필요한 부분에 대해서만 라벨링이 이루어진다.

4. 모의실험 결과

본 논문에서 제안한 임베디드 시스템을 위한 가상

```
function Binarization(I_t : Image) return B_t : Image,
bs_t : Point is
begin
    flag = false
    ...
    for each pixel (x, y) in SearchArea
        if I_t(x, y) > threshold
            B_t(x, y) = 1
            if flag is not true
                flag = true
                bs_t = (x, y)
            end if
        else
            B_t(x, y) = 0
        end if
    end for
    return B_t, bs_t
end function
```

Fig. 7. Changing binarization algorithm for proposed labeling algorithm.

```
function Labeling(B_t : Image, bs_t : Point) return
L_t : Image is
begin
    for each pixel (x, y) in B_t from bs_t
        if B_t(x, y) is connected to B_t(bs_t)
            size = size + 1
            L_t(x, y) = 1
            if size >= S
                flag = true
                return L_t
            end if
    end for
    return L_t
end function
```

Fig. 8. Proposed labeling algorithm for the recognizing the virtual touch.

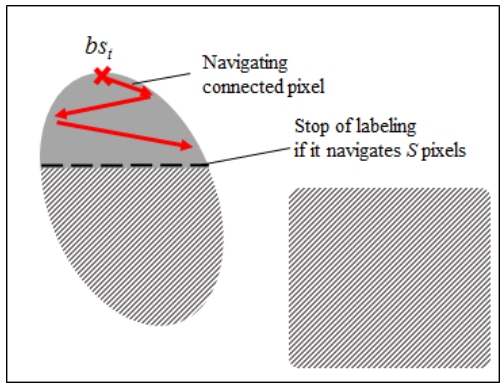

Fig. 9. Result of appling proposed labeling algorithm.

터치 알고리즘의 실시간 처리 가능 여부를 확인하기 위해 알고리즘 내의 라벨링 과정과 이진화 과정, 가상 터치 인식 과정으로 나누어 수행시간을 측정하였다. 이 때 임베디드 시스템에 사용할 기기는 Raspberry pi 3을 사용하였다. Raspberry pi 3은 1.2Ghz의 쿼드 프로세서와 1Gb의 RAM을 장착한 임베디드 기기이다. 또한 깊이 카메라로는 ASUS사의 Xtion Pro Live를 사용하였다. 이 깊이 카메라를 통해 초당 30프레임으로 영상을 촬영하였다. 이 때 총 100프레임을 측정하여 속도를 측정하였다. 이 때 각 프레임 간 간격은 약 34ms이므로, 실시간 가상 터치를 구현하기 위해서는 처리 시간이 33ms 이하여야 한다. 본 실험 장치는 Fig. 10과 같이 구성되었다.

Table 1에서는 기존 이진화 방법과 라벨링 방법을

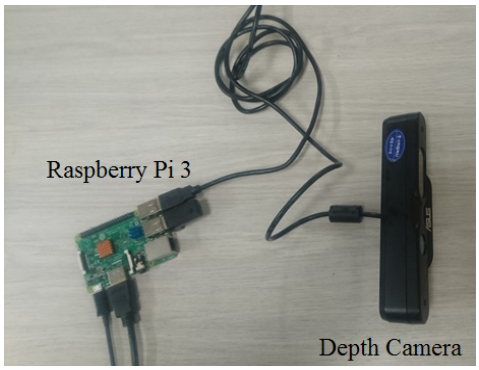

Fig. 10. Devices in the simulation.

Table 1. Processing time of virtual touch for each resolution

Device	Processing time per frame (ms)	
	320 × 240	680 × 480
PC	3	11
Raspberry Pi 3	28	78

이용하였을 때의 처리시간을 PC와 비교하여 측정한 것이다. 이 때 PC에서는 해상도에 상관없이 매우 빠르게 가상 터치 알고리즘이 처리되지만, 임베디드 기기에서는 해상도가 640 × 480인 경우에 처리 속도가 78ms로, 실시간으로 가상 터치를 인식할 수 없음을 알 수 있다.

Table 2는 주어진 임베디드 환경에서 640x480의 깊이 영상을 획득하여 가상 터치를 수행할 때, 기존 이진화, 라벨링 방법을 적용하였을 때의 각 단계별 처리 속도를 나타낸 것이다. 이 때 가상 터치 인식 내에서 이진화 단계와 라벨링에서 대부분의 시간이 소요됨을 알 수 있다.

가상 터치 인식 내에서 깊이 영상을 이진화 하는 방법을 기존의 방법인, 전체 영상을 이진화 하는 방법과 제안된 방법인, 이전 터치 지점을 중심으로 일부 영역만을 이진화하는 방법을 적용하였을 때의 처리 속도를 Table 3과 같이 비교하였다. 터치를 하지

Table 2. Processing time of each step within virtual touch

Step	Processing time per frame (ms)
Binariization	25
Labelling	26
Touch recognition	0.3

Table 3. Comparition of processing time according to binarization method

Binarization method	Processing time per frame (ms)	
	No touch	Within touch
Conventional	25	25
Proposed	25	0.8

않는 상태일 경우에는 이전 터치 지점이 없기 때문에 기존의 방법과 제안된 방법의 처리 시간이 차이가 나지 않는다. 하지만 터치 중인 상태에서는 기존 방법에서 26ms이던 이진화 과정 시간이, 제안된 방법에서는 0.8ms로 확연히 줄어듬을 확인할 수 있다.

이진화된 영상을 라벨링하는 방법을 기존 방법인, 전체 영상을 라벨링 하는 방법과, 제안된 방법인, 객체가 처음 나타났을 때를 시작점으로 하여 라벨링을 시작하는 방법을 적용하였을 때의 처리 속도를 Table 4와 같이 비교하였다. 기존 라벨링 방법을 적용하였을 때는 라벨링 과정 시간이 26ms로 나타났으나, 제안된 라벨링 방법을 적용하였을 때는 6ms로 줄어들었다. 이는 터치를 수행할 객체가 영상 외곽에 나타날 확률보다 영상 중앙에 나타날 확률이 높기 때문에, 라벨링 개선 효과가 높게 나타난 것으로 분석된다.

가상 터치 인식 시스템에서 제안된 방법들을 적용하여 총 처리 시간을 Table 5와 같이 비교하였다. 이 때, 제안된 이진화, 라벨링 방법을 적용하였을 때, 총 처리시간이 31ms로써, 실시간으로 가상 터치를 수행

Table 4. Comparition of processing time according to labeling method

labeling method	Processing time per frame (ms)
Conventional	26
Proposed	6

Table 5. Comparition of processing time according to binarization and labeling methods

Method	Processing time per frame (ms)
Conventional	78
Proposed binarization only	56
Proposed labeling only	53
Proposed binarization & labeling	31

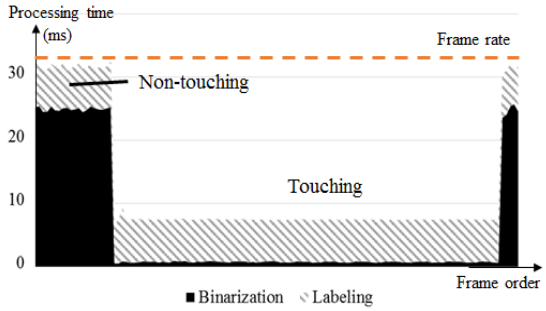

Fig. 11. Processing time of the virtual touch recognition applied the proposed method.

할 수 있다. 이 때 Fig. 11과 같이 터치 중일 때 가상 터치 인식 속도가 비약적으로 빨라지는 것을 확인할 수 있다.

5. 결 론

본 논문에서는 임베디드 기기를 이용하여 가상 터치 인식 시스템을 임베디드 환경에서 구현하였다. 이를 통해 깊이 인식 기기를 소형화하여 범용적으로 가상 터치 인식을 수행할 수 있도록 하였다. 임베디드 환경에서 해상도가 높아짐에 따라 실시간 가상 터치 인식 처리를 할 수 없는 문제를 발견하였다. 이러한 문제를 해결하기 위해, 가상 터치 인식 처리 과정에서 이진화와 라벨링 과정에서 대부분의 시간이 소요된다는 것을 확인하여, 가상 터치에 알맞게 이진화와 라벨링 과정을 개선하였다. 그 결과 기존 640×480 깊이 영상에서 프레임 당 처리시간이 기존 78ms 걸리던 것에서 제안된 방법을 적용한 결과 31ms로 측정되어 처리속도가 60.3%만큼 단축된 것을 확인할 수 있었다. 또한 제안된 방법을 적용함으로써 실시간 가상 터치 인식 시스템을 구현할 수 있었다.

실시간 가상 터치 인식 기기를 임베디드 환경에서 구현함에 따라 기존 값비싼 물리적 터치스크린을 충분히 대체할 수 있다는 것을 보였다. 이에 따라 대형 스크린 환경에서도 터치 인터페이스를 값싸게 도입할 수 있게 되어 터치 응용 분야에 널리 이용될 것으로 기대된다.

REFERENCE

[1] D.S. Lee and S.K. Kwon, "Touch Pen Using Depth Information," *Journal of Korea Multimedia Society*, Vol. 18, No. 11, pp. 1313-1318, 2015.

[2] V. Soni, M. Patel, and R.S. Narde, "An Interactive Infrared Sensor Based Multi-Touch Panel," *International Journal of Scientific and Research Publications*, Vol. 3, No. 3, pp. 610-623, 2013.

[3] J.A. Stankovic, "Misconceptions About Real-Time Computing: A Serious Problem for Next Generation Systems," *IEEE Computer*, Vol. 21, No. 10, pp. 10-19, 1988.

[4] K.G. Shin and P. Ramanathan, "Real-Time Computing: A New Discipline of Computer Science and Engineering," *Proceeding of the IEEE*, Vol. 82, No. 1, pp. 6-24, 1994.

[5] L. Sha, T. Abdelzaher, K.E. Årzén, A. Cervin, T. Baker, A. Burns, et al., "Real Time Scheduling Theory: A Historical Perspective," *Real-time Systems*, Vol. 28, No. 2-3, pp. 101-155, 2004.

[6] W.Y. Lee, "Power-efficient Scheduling of Periodic Real-time Tasks on Lightly Loaded Multicore Processors," *Journal of The Korea Society of Computer and Information*, Vol. 17, No. 8, pp. 11-19, 2012.

[7] J.Y. Kim, K.K. Kwon, S.I. Lee, and K.S. Ahn, "Implementation of IDE for OSEK/VDX OS Extended Real-time Processing and Protection Functions," *Journal of Korean Institute of Information Technology*, Vol. 11, No. 2, pp. 119-126, 2013.

[8] N. Singhal, J.W. Yoo, H.Y. Choi, and I.K. Park, "Implementation and Optimization of Image Processing Algorithms on Embedded GPU," *IEICE Transactions on Information and Systems*, Vol. 95, No. 5, pp. 1475-1484, 2012.

[9] J.H. Lee, S.H. Kang, M.H. Lee, S.Z. Li, H.I. Kim, and I.K. Park, "Real-Time Parallel Image Processing Library Using Mobile GPU," *Journal of KIISE: Computing Practices and Letters*, Vol. 20, No. 2, pp. 96-100, 2014.

[10] C.M. Huang, C.W. Lin, and W.P. Tsai, "A Multi-Core Based Parallel Streaming Mechanism for Concurrent Video-on-Demand Applications," *IEEE Communications Letters*, Vol. 13, No. 4, pp. 286-288, 2009.

[11] S.W. Hong and Y.L. Lee. "CPU Parallel Processing and GPU-accelerated Processing of UHD Video Sequence Using HEVC," *Journal of Broadcast Engineering*, Vol. 18, No. 6, pp. 816-822, 2013.

[12] J.U. Lee, J.Y. Sun, and M.C. Won, "Real-Time Algorithm for Relative Position Estimation Between Person and Robot Using a Monocular Camera," *Transactions of the Korean Society of Mechanical Engineers-A*, Vol. 37, No. 12, pp. 1445-1452, 2013.

[13] M.C. Lee, C.H. Jang, and M.H. Sunwoo, "A Task Scheduling Strategy in a Multi-core Processor for Visual Object Tracking Systems," *Transaction of the Korean Society of Automotive Engineers*, Vol. 24, No. 2, pp. 127-136, 2016.

[14] D.S. Lee and S.K. Kwon, "Video Event Control System by Recognition of Depth Touch," *Journal of the Korea Industrial Information Systems Research*, Vol. 21, No. 1, pp. 35-42, 2016.

권 순 각

1990년 2월 경북대학교 전자공학과 졸업
1992년 2월 KAIST 전기및전자공학과 석사
1998년 2월 KAIST 전기및전자공학과 박사
1997년 3월~1998년 8월 한국전자통신연구원 연구원
1998년 9월~2001년 2월 기술신용보증기금 기술평가센터 팀장
2003년 9월~2004년 8월 Univ. of Texas at Arlington 방문 교수
2010년 9월~2011년 8월 Massey University 방문 교수
2001년 3월~현재 동의대학교 컴퓨터소프트웨어공학과 교수
관심분야 : 멀티미디어신호처리, 영상통신

이 동 석

2015년 2월 동의대학교 컴퓨터소프트웨어공학과 졸업
2015년 3월~현재 동의대학교 컴퓨터소프트웨어공학과 석사과정
관심분야 : 멀티미디어신호처리, 영상인식