제스처 인식을 통한 프리젠테이션 인터페이스

김진욱*, 김세훈*, 홍광진*, 정기철*
*숭실대학교 미디어학과
e-mail:jinukskim@gmail.com

Presentation Interface based on Gesture Recognition

Jin-uk Kim*, Se-hoon Kim*, Kwang-jin Hong*, Kee-chul Jung*
*School of Media, Soongsil University

요 약

본 논문에서는 키넥트를 사용하여 제스처를 인식해 프리젠테이션이 가능한 인터페이스를 제작하였다. 키넥트 카메라는 Microsoft Kinect SDK1.7 라이브러리를 사용해서 신체의 좌표값을 받아 손의 위치와 손의 제스처를 인식하는데 사용했으며, 파워포인트를 제어하기 위해 후킹 기능을 사용한다. 기존 키넥트에서 사용하던 제스처인 sweep 을 기본으로, grip 과 push 제스처로 프리젠테이션에 필요한 기능을 추가했다. 제스처 인식의 결과를 후킹을 통해 파워포인트로 전달해서, 슬라이드의 이동 뿐 아니라 메모 기능과 지우기 기능이 추가된 인터페이스를 제공한다. 인터페이스는 발표자의 발표능력 향상과 더불어, 제스처 인식 인터페이스를 타 콘텐츠에 적용이 가능하므로 추후 콘텐츠의 제작 및 상용화가 가능하다.

1. 서론

1980년대 후반부터 보편화된 GUI 인터페이스는 마우스 등을 이용해 작업하는 제한된 환경구축이 필요했다. 하지만 현대사회에서의 다양한 입출력장치의 발전은 사용자가 자연스럽게 컴퓨터와 상호작용하는 인터페이스 개발을 촉진시켰다.

1979년 MIT Media Lab 의 Chris Schmandt 가 음성과 제스처를 이용한 그래픽 인터페이스 개발연구를 시작으로, 제스처 인식은 장비의 조작미숙 및 불편함을 해소하는 대체수단으로 각광받고 있다[1].

기본적인 타이핑이나 PC 의 조작과는 달리, 체감형 인터렉티브 게임이나 인터렉티브 콘텐츠에서는 사용자와 시스템과의 자연스러운 상호작용이 필수적이다. 관람객이 직접 움직여 체험해 보는 체감형 인터렉티브 게임이나 전시장, 박물관 등에 적용되는 정보전달 인터렉티브 콘텐츠 등에는 기존의 장치 인터페이스가 아닌 제스처 인식으로 사용자의 경험과 참여를 극대화 시킬 수 있어 관련 연구가 필요하다.

컴퓨터와 인간의 상호작용의 발전은 인간의 삶에도 많은 도움을 줄 수 있는데 그중 하나가 프리젠테이션에서 사용하는 경우가 된다. 프리젠테이션(Presentation)을 진행하는데 있어, 발표자의 자연스러운 행동은 청중의 집중도를 높이는데 중요한 역할을 한다. 하지만 프리젠테이션을 제어하기 위해 기기를 통한 제어가 필요하며, 이 행위는 집중도를 떨어뜨리는 행위로 연결될 수 있다.

이에 본 논문에서는 키넥트(Kinect)센서의 스켈레톤(Skeleton) 좌표인식을 이용한 손의 위치추적으로 파워포인트 조작이 가능한 제스처 인식 인터페이스를 제안한다. 사용자가 파워포인트(PowerPoint)의 슬라이드쇼(Slide Show)를 통한 프리젠테이션을 하는 도중에 자연스럽게 슬라이드 이동이 가능하며, 기존 프리젠테이션을 제어하는 장치에서는 사용할 수 없는 메모 및 지우기 기능을 사용할 수 있도록 함으로써, 더 나은 인터페이스 환경을 구축하여 발표자의 프리젠테이션 능력 향상에 이바지한다.

2. 관련연구

1993년 MIT Media Lab 이 장갑을 사용하여 3차원 그래픽 인터페이스를 제공하는 연구는 장치를 통한 제스처 인식 연구의 시초라 할 수 있다[2]. 이후 위(Wii), 키넥트 등 다양한 장치가 개발되었고, 이를 이용한 제스처 연구는 활발히 진행 중에 있다.

특히 2010년에 Microsoft 사의 키넥트가 등장한 이후에 컴퓨터비전 관련분야뿐만 아니라, 음성인식을 통한 인터렉션에 관한 많은 연구가 진행되고 있다. 이중에서도 신체 움직임을 이용한 연구나 인터렉티브 콘텐츠는 실제 많이 상용화되고 있다.

Smorkalov 등은 스켈레톤을 이용해 virtual image 를 생성하여 활용하는 콘텐츠를 개발하였다[3]. 실제 사용자의 움직임을 스켈레톤 이미지로 받아들인 뒤, 이를 3D 가상 모델 강사로 만들어 스트리밍 강의를 하는 콘텐츠로서 가상교육에 초점이 맞춰져 있는 이 연구는 스켈레톤 좌표값을 받아 3D 모델링을 하는 것이 주된 기능이며, 제스처를 통한 제어는 하지 않는다.

스켈레톤에서 손의 좌표값과 미리 정해놓은 좌표값의 일

치여부를 판별하여 마우스조작을 대신하는 연구도 진행되고 있다. Osunkoya 등은 좌클릭, 우클릭, 더블클릭 등의 마우스조작을 손의 좌표값과 다른 신체부위의 좌표값을 이미 정해진 좌표값과 비교하여 제스처로 인식하고, 이를 통해 파워포인트를 제어하는 인터페이스를 제안하였다[4]. 좌표값 비교의 특징 상 제스처가 명확해야 하므로 큰 움직임이 필요해서 사용자가 불편함을 느낄 수 있고, 제스처로 인식하고 싶지 않은 움직임에 의해 인식되는 오류가 발생하기 때문에 단순한 파워포인트의 슬라이드 이동 기능만을 지니고 있어 보완이 필요하다.

GIRBACIA 등은 키넥트와 위를 함께 사용하여 프리젠테이션을 하는 연구를 진행하였다[5]. 키넥트의 음성인식 기능으로 프리젠테이션을 동작시키고, 위모트의 가속도센서와 버튼을 이용하여 조작한다. 키넥트와 위모트 두가지의 장치를 사용하는 이 방법은 제스처 인식이 아닌, 장치조작에 의한 제어에 가깝다.

Ren 등이 제안하는 키넥트 기반의 hand gesture 를 인식하는 연구는 일반적인 RGB 카메라에서 손을 검출하고, 손의 윤곽선을 찾아 time-series curve 를 만들어 이를 바탕으로 feature vector 를 구성하는 방법이 있다[6]. 위의 방법은 90%이상의 정확도를 갖지만 손에 검은띠를 착용해서 손의 위치를 정확히 읽어야 하고, 이미 라이브러리화된 모든 특징값(feature)과 비교해야 하는 작업이 필요하다.

본 논문에서 구현하는 인터페이스는 키넥트를 사용하여 사용자가 불편함 없이 취하는 제스처를 인식하되 기존의 좌표값 비교 위주의 인식에서 벗어나, 주먹을 쥐거나 손을 앞으로 뻗는 행위 등의 제스처 인식을 보강한다. 또한 인터페이스로서 사용자가 사용하기 쉬우면서 기본 슬라이드 기능 이외에 메모 기능과 지우기 기능을 추가하여 효과적인 프리젠테이션 환경을 제공하는데 초점을 둔다.

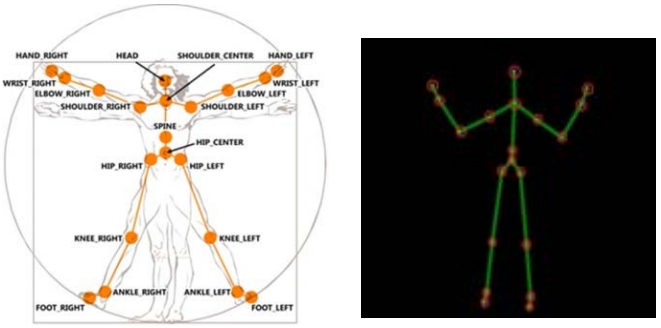
3. 시스템구성 및 기능

인터페이스 제작에 사용된 키넥트(Kinect for xbox 360)는 RGB 카메라, IR 센서, 마이크로폰, Tilt Motor 로 구성되어 있으며, IR 센서를 통해 일반 카메라와 다르게 사물의 깊이를 측정할 수 있다. 키넥트의 사물 인식 거리는 0.8m 부터 4m 까지의 깊이를 인식 할 수 있으며 Tilt Motor 를 이용한 기기의 상하 움직임과 마이크로폰을 통한 음성인식도 가능하다.

Microsoft Kinect SDK v1.7 [7] 라이브러리로 음성인식과 스켈레톤 정보를 받아오며, 스켈레톤 정보의 x, y, z 좌표값으로 제스처를 인식한다.

기존 Microsoft 의 파워포인트 슬라이드쇼 제어는 키보드와 마우스 등의 입력장치를 이용한다. 본 논문에서는 키넥트를 사용해서 사용자 신체부위의 좌표값으로 gesture 와 position 을 인식해서 제어한다. 특히, 기본적인 슬라이드의 이동뿐만 아니라 메모 기능과 지우기 기능도 제어가 가능하다.

그림1와 같이, 키넥트는 사람을 20개 영역으로 구분하여 인식한다. 본 논문에서 우리는 20개 영역 중 왼손, 오른손 2개 영역만을 사용한다.



(그림 1) 키넥트가 인식하는 사람신체 부위

키넥트 라이브러리를 통해 검출한 좌표값이 키넥트의 일반 화면공간(Screen)에서는 픽셀로 측정되며 공간좌표가 좌측상단이 원점인 반면에, 스켈레톤공간(Kinect)의 기본 단위는 m(미터)이고 화면 정중앙이 원점이다. 각각 공간의 단위와 좌표가 다르므로 통일성이 필요하며, 이를 위해 스켈레톤 공간의 단위를 pixel로 변환시키고 화면 좌측상단을 원점으로 만들어 좌표값을 추출한다[8]. 이를 토대로 일반화면공간의 너비와 높이를 S_w, S_h 라 하고 스켈레톤 공간의 너비와 높이를 K_w, K_h라 명시할 때, 얻고자하는 좌표 x, y는 다음 식1로 추출한다.

식1) $$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} S_w/K_w & 0 & -S_w/2 \\ 0 & S_h/K_h & -S_h/2 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} K_x \\ K_y \\ 1 \end{bmatrix}$$

pixel단위의 좌표값 추출은 단순히 오른손의 x, y좌표값으로 마우스를 움직일 수 있게 하며, 손의 제스처를 인식하기에도 무리가 없다.

화면전환 기능은 슬라이드의 앞, 뒤 이동이며, 편집 기능은 중요 부분에 메모하는 기능(메모 기능)과 메모를 지우는 기능(지우기 기능) 등으로 구분된다. 필요한 기능과 그에 해당하는 단축키를 정리하면 다음 표1과 같다.

<표 1> 슬라이드쇼 기능 및 단축키

기능	단축키
다음 슬라이드 이동	방향키→
이전 슬라이드 이동	방향키←
마우스포인터를 펜툴로 변환	Alt + P
펜툴을 마우스포인터로 변환	Alt + A
마우스포인터를 지우기툴로 변환	Alt + E

명시된 각 기능들은 사용자의 제스처가 인식되는 동시에 내부적으로 해당 단축키가 실행되어 수행한다.

슬라이드쇼 상황에서는 슬라이드의 앞, 뒤로 이동과 펜

툴, 지우기툴 등에서 마우스의 드래그 또는 클릭이벤트를 통한 효과가 필요하다. 이에 따른 제스처는 크게 sweep, grip, press 를 혼합하여 각 기능이 실행되도록 설계된다. 각 기능을 제어하기 위한 gesture 는 다음 표2와 같다.

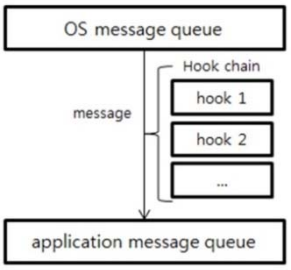
<표 2> 기능제어를 위한 제스처

기능	제스처
Mouse Click	grip gesture
Mouse Click Release	release gesture
Next Slide	grip & sweep right
Back Slide	grip & sweep left
Mouse Cursor Change	Press gesture

다음 슬라이드 이동은 오른손이 grip 된 상태에서 오른쪽으로 이동 후 grip 한 손을 펴는 행위가 이뤄지는 순간이 종료기점으로 슬라이드이동이 이뤄진다. 오른쪽으로의 이동은 이전프레임과의 x좌표값 비교를 통해 일정수준 이상의 이동이 생겼을 경우로 한다. 이전 슬라이드 이동 역시 왼손을 이용해서 방향의 반대로만 적용한다. push 와 grip 은 손의 z좌표를 이용해서 이를 제스처로 인식한다.

제스처 인식과 더불어 필요작업은 사용자의 제스처와 키넥트를 통해 프리젠테이션을 제어할 수 있도록 하는 것이다. 파워포인트 슬라이드쇼는 윈도우에서 가장 상위에 창이 존재할 때 제어가 가능하다. 이를 위해 본 인터페이스는 후킹을 사용하여 키넥트에서 받아온 좌표값으로 마우스를 제어한다.

후킹(hooking)이란 소프트웨어 공학 용어로, 운영 체제나 응용 소프트웨어 등의 각종 컴퓨터 프로그램에서 소프트웨어 구성 요소 간에 발생하는 함수 호출, 메시지, 이벤트 등을 중간에서 바꾸거나 가로채는 명령, 방법, 기술이나 행위를 말한다. 이때 이러한 간섭된 함수 호출, 이벤트 또는 메시지를 처리하는 코드를 후크라고 한다[9](그림2).



(그림 2) 후킹

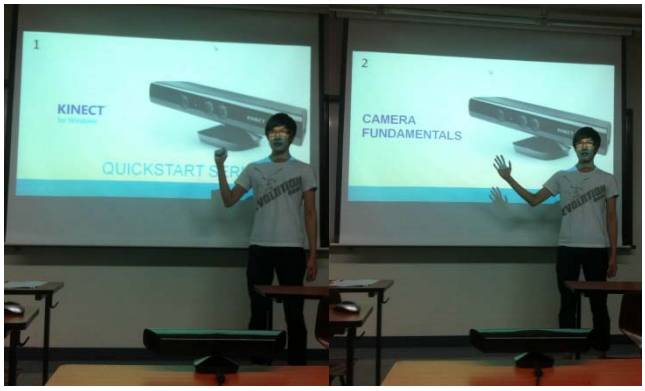
후킹을 이용해서 키넥트 제스처가 인식되는 것이 일반적인 윈도우의 마우스 핸들러와 같이 사용되도록 해서 파워포인트 상에서 작동된다.

4. 실험결과 및 분석

실험은 일반 강의실을 가정한 실험공간에서 진행한다. 공간에서 키넥트와 사용자간의 거리는 프로젝터간의 거리와 같은 약 2m 로 하고, 인식범위는 프로젝터의 영상범위로 제한한다.

실험은 다음슬라이드이동, 이전슬라이드이동, 기능전환, 메모 기능, 지우기 기능의 동작여부를 확인한다.

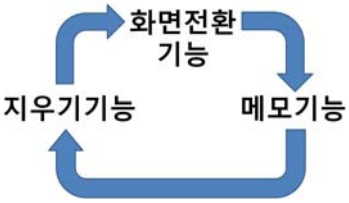
사용자가 실제 프리젠테이션을 하고 있는 상황으로 가정할 때, 슬라이드를 다음 장으로 이동해야하는 경우가 발생한다. 이 경우 아래의 그림3과 같이 손을 좌측에서 우측으로 움직인다. 이때 제스처의 시작점에서 손을 움켜쥐고, 제스처를 마무리하는 지점에서 손을 펴준다.



(그림 3) 다음슬라이드로 이동

프리젠테이션 도중 이전슬라이드로 이동해야하는 경우, 위와 유사하게 손을 우측에서 좌측으로 움직인다(그림4).

(그림 4) 이전슬라이드로 이동

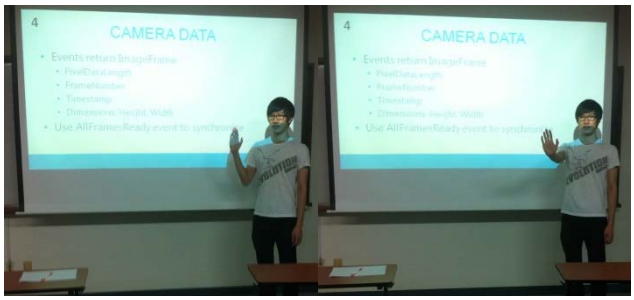


(그림 5) 기능전환 루트

인터페이스에서는 화면전환 기능 이외에 메모 기능, 지우기 기능 등의 편집 기능을 제공하는데, 각각 기능을 사용하기 위해서는 파워포인트의 커서자체를 바꿔야 한다. 커

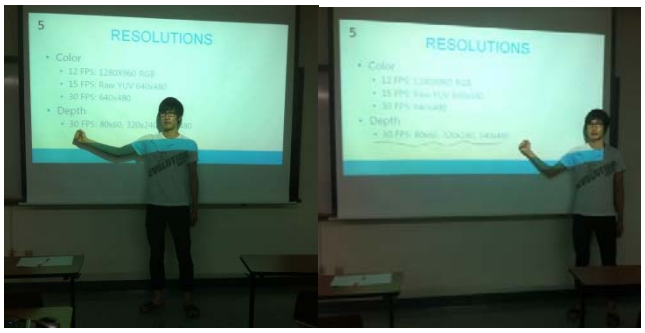
서는 순차적으로 전환되며, 이를 위한 제스처로 push 가 사용되었다(그림5).

각 기능의 전환은 아래의 그림6과 같이 손을 정면으로 뻗는 행위로 가능하다.

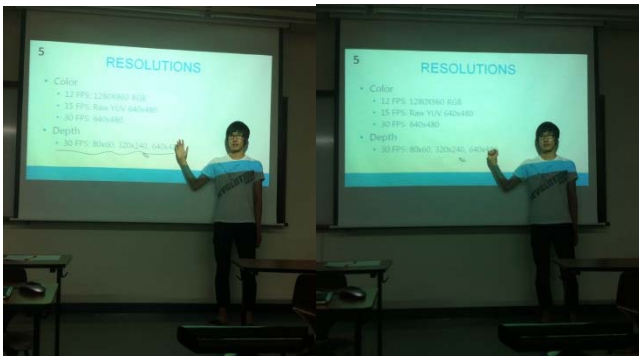


(그림 6) 기능전환

메모 기능에서는 손의 좌표에 따라 슬라이드에 메모가 기록된다. 손의 움직임에 따라 항상 메모가 그려지게 된다면 정확한 메모를 기록할 수 없으므로, 그려야 하는 부분에서 손을 grip 하는 동작을 취함으로써 그려야 하는 부분을 구분하는 마우스클릭 역할을 한다. 즉, 선을 그려야 하는 부분에서 grip 상태로 손을 움직이면 마우스 드래그 상태가 되며 메모가 그려지게 되고, 그리는 행위가 종료되면 주먹을 다시 release 하게 되면 마우스드래그가 종료되고 메모기록이 종료된다(그림7).



(그림 7) 메모 기능



(그림 8) 지우기 기능

그렸던 메모를 지워야 하는 경우가 발생하면 지우기 기능을 이용한다. 지우기 기능은 위에서 언급한 기능전환을 통해 전환 후 사용한다. 지우기 기능 역시, 메모 기능과 같은 방법으로 동작하는데, 기능전환 후 grip 으로 실행한다(그림8).

5. 결론 및 향후 연구

본 연구는 파워포인트를 통해 프리젠테이션을 할 경우에 제스처 기반의 제어가 가능하도록 인터페이스를 제작하였다. 키넥트에서 사람을 스켈레톤으로 인식받아 프리젠테이션의 슬라이드이동을 하는 기존의 연구들과는 달리, 메모 및 지우기 기능을 추가하였고 제스처 기반을 바탕으로 기능 제어를 통해 발표자의 발표를 돕는 역할을 한다. 또한 기존의 손을 sweep 함으로써 제어되는 슬라이드의 이동을, grip 과 sweep 를 동시에 하는 것으로 대체함으로써 조작미숙에 의한 슬라이드의 이동을 방지할 수 있었다.

프리젠테이션을 제어하기 위해 만든 제스처 인식 모듈은 후에 다른 콘텐츠에서도 사용할 수 있도록 재사용성이 가능하며, 이는 추후 다른 연구에 큰 도움이 될 것으로 예상된다.

참고문헌

[1] Bolt, Richard A. “Put-that-there”: Voice and gesture at the graphics interface. Vol. 14. No. 3. ACM, 1980.

[2] Sturman, David J., and David Zeltzer. "A survey of glove-based input." Computer Graphics and Applications, IEEE 14.1 (1994): 30-39.

[3] Smorkalov, Andrey, Mikhail Fominykh, and Ekaterina Prasolova-Førland. "Virtualizing Real-life Lectures with vAcademia and Kinect."

[4] Osunkoya, Toyin, and Johng - ChernChern. "Gesture-Based Human-Computer-Interaction Using Kinect for Windows Mouse Control and PowerPoint Presentation."

[5] GIRBACIA, Florin, and Silviu BUTNARIU. "DEVELOPMENT OF A NATURAL USER INTERFACE FOR INTUITIVE PRESENTATIONS IN EDUCATIONAL PROCESS."Conference proceedings of" eLearningand Software for Education". No. 02. 2012.

[6] Ren, Zhou, et al. "Robust hand gesture recognition with kinect sensor." Proceedings of the 19th ACM international conference on Multimedia.ACM, 2011.

[7] http://www.microsoft.com/en-us/kinectforwindows/develop/developer-downloads.aspx

[8] Webb, Jarrett, and James Ashley. Beginning Kinect Programming with the Microsoft Kinect SDK. Apress, 2012.

[9] http://ko.wikipedia.org/wiki/%ED%9B%84%ED%82%B9