서버/클라이언트 통신을 통한 영상 기반 위치 인식 및 근육 센서를 이용한 상호작용 모바일 증강현실 시스템

Interactive Mobile Augmented Reality System using Muscle Sensor and Image-based Localization System through Client-Server Communication

이성진, Sungjin Lee*, 백다빈, Davin Baik**, 최상연, Sangyeon Choi**, 황성수, Sung Soo Hwang***

요약 실제로 많은 게임들은 사용자의 신체의 움직임을 유발하기 보다는, 마우스 키보드 등 간단한 컨트롤러 조작을 통해서 게임을 플레이하도록 되어 있다. 이러한 게임은 사용자에게 운동부족을 유발하는 한계가 있다. 따라서 본 연구는 움직임을 유발하는 시스템 개발을 통해 이러한 기존의 게임 시스템이 가지고 있던 문제를 해결하고, 더 실감나는 시스템을 사용자에게 제공할 것이다. 이에 주어진 공간에서 사용자의 위치를 인식하며, 증강된 가상의 게임 캐릭터와 상호 작용하는 모바일 증강현실 시스템을 제안한다. 사용자들에게 마치 현실 공간에 가상 캐릭터가 실제로 있는 듯한 현실감과 재미를 느낄 수 있도록 증강현실 기술을 사용하며, 가상 캐릭터와 상호작용하기 위한 사용자의 동작을 인식하는 암밴드 컨트롤러와 호환이 되는 모바일 게임 시스템을 설계한다.

Abstract A lot of games are supposed to play through controller operations, such as mouse and keyboard rather than user's physical movement. These games have limitation that causes the user lack of movement. Therefore, this study will solve the problems that these traditional game systems have through the development of a motion-producing system, and provide users more realistic system. It recognizes the user's position in a given space and provides a mobile augmented reality system that interacts with virtual game characters. It uses augmented reality technology to make users feel as if virtual characters exist in real space and it designs a mobile game system that uses armband controllers that interact with virtual characters.

핵심어: Augmented Reality, game, Localization Recognition, Interaction, Muscle Sensor

본 논문은 2016년도 정부(교육부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임 (No. 2016R1D1A3B03934808).

*주저자: 한동대학교 전산전자공학부 학부생

**공동저자 : 한동대학교 전산전자공학부 학부생

***교신저자 : 한동대학교 전산전자공학부 교수: e-mail: sshwang@handong.edu

■ 접수일 : 2018년 7월 3일 / 심사일 : 2018년 7월 26일 / 게재확정일 : 2018년 10월 2일

1. 서론

최근 증강현실에 대한 관심이 높아지고 증강현실 기술이 점점 대중화 되고 있다. 증강현실은 실제로 존재하지 않는 가상의 정보 또는 영상을 현재의 영상에 실존하듯이 합성해주는 기술을 말한다. 이러한 기술의 특성으로 인해 교육용 어플리케이션이나, 음식점 정보 어플리케이션 등에 사용되기도 한다. 이러한 증강현실 어플리케이션의 예로는 IKEA의 IKEA Place, Vito 사의 Star Walk, Niantic 사의 Pokemon GO 등을 들 수 있다. IKEA Place의 경우 실제로 존재하지 않는 가구를 공간에 배치해보고 결과를 화면을 통해 확인할 수 있다. 또한 Star Walk는 밤하늘 상에서 별의 이름이나 별자리 등을 표시해 준다. 마지막으로 Pokemon GO는 GPS로 확인한 사용자의 좌표에 따라 가상으로 증강된 몬스터를 획득할 수 있는 게임이다.

이러한 증강현실 기술은 게임 분야에 활발하게 적용 가능하며 새로운 형태의 게임 콘텐츠를 개발하는 데에도 활용이 가능하다. 대한민국에서는 게임에 대한 관심이 꾸준히 증가하고 있다. 다년간 대한민국 게임 사용량 중 1위를 차지하고 있는 League of Legends는 게임 사용자 수가 3500만명 정도로 추정된다. 그리고 일간 접속자 또한 1200만으로 실 사용자 또한 매우 많다. 이는 SNS 의 한 종류인 인스타그램의 일 사용자가 750만인 것을 볼 때에 압도적인 숫자로 볼 수 있다[1]. 이러한 관심에 힘입어 고성능 그래픽 카드와 정밀한 물리 엔진을 활용한 현실감 높은 게임 콘텐츠가 다수 개발되고 있다. 그러나 증강현실 기술을 활용하면 또 다른 형태의 현실감 높은 게임 콘텐츠를 개발할 수 있다. 이는 증강현실을 활용한 게임 콘텐츠는 게임 환경 그 자체가 사용자의 현실이 되기 때문이다. 뿐만 아니라 마우스 및 키보드 등의 인터페이스가 아닌 최근 개발되고 있는 사용자의 신체 움직임을 인지하는 인터페이스를 활용하는 경우 현실성이 배가 될 수 있다.

증강현실 기술을 활용한 게임 어플리케이션은 기존의 게임 콘텐츠가 가지는 사용자의 운동량 감소 문제를 해결할 가능성을 가지고 있다. 기존의 게임 콘텐츠는 중독 및 사용자의 현실 인지 능력 저하 등의 문제를 가지고 있으며, 특히 사용자의 운동 능력 저하를 가지고 올 수 있다는 문제점이 있다. 현존하는 대부분의 게임은 그 특성상 한 자리에 앉아서 할 수밖에 없고, 그 결과 운동량이 부족해지는 결과를 낳을 수 있기 때문이다. 그러나 게임의 진행을 위해 사용자의 이동이 요구되는 증강현실 기반 게임을 개발할 경우 이러한 운동 부족 현상을 해결할 수 있다. 특히 신체 움직임을 인지하는 인터페이스를 추가적으로 활용하는 경우에는 운동 효과까지 기대할 수 있다.

따라서 본 연구에서는 게임 진행을 위해 사용자의 이동이 필수적이며, 사용자의 신체 움직임을 인지하는 증강현실 게임 시스템을 제안한다. 본 연구에서는 사용자의 위치를 GPS가 아닌 영상 인식 기반으로 인식하며, 이를 통해 GPS가 작동하지 않는 실내에서도 사용이 가능할 수 있게 하였다. 사용자의 이동 반경을 넓히기 위해 게임 수행 가능 영역을 넓게 설정하였으며, 따라서 해당 영역에 대한 영상 지도를 서버에 위치하고 사용자를 통해 영상을 실시간으로 전송받음으로써 사용자의 위치를 추정하게 하였다. 또한 증강된 객체와의 상호작용은 사용자의 손 동작 인식을 통해 이루어지게 하였으며 이를 통해 1인칭 슈팅 게임(FPS)를 제작하였다.

제안하는 시스템과 유사한 시스템으로는 실내 환경 증강 현실게임[2]과 PTAM을 이용한 3차원 공간 구성 및 게임[3]이 있다. 실내 환경 증강 현실게임은 인위적인 마커를 이용해 게임 환경을 구성하며 카메라를 이용해 마커를 추적하여 사용자의 위치를 추정한다. 또한 영상을 통한 사용자의 손 검출 및 트래킹을 통해 사용자와 시스템 간의 상호작용을 이룬다. 이 시스템의 한계점은 게임을 플레이하기 위해 게임 환경마다 마커를 부착해야 하고, 사용자의 손동작을 인식하기 위해 영상 내 손 영역을 추출하고 트래킹 해야 하기에 많은 계산 량이 필요하다. PTAM을 이용한 3차원 공간 구성 및 게임 시스템의 경우에는 제안 시스템과 유사하게 마커를 사용하지 않으며, 게임 공간상의 특징점 검출을 통해 실내 3D 지도 제작 및 비교를 통해 사용자의 위치를 인식한다. 하지만 PTAM의 특성상 책상이나 방 등의 소규모 공간에서만 원활하게 작동되며 게임 공간이 커질수록 속도가 저하된다.

본 시스템은 앞선 연구들의 한계점을 다음과 같이 보완한다. 사용자 위치 인식에 있어서 인위적인 마커 없이 ORB-SLAM 시스템을 사용한다. 사용자 모바일 기기의 카메라를 이용하여 게임 지도를 제작하고 사용자의 위치를 추적하며, Bag of words 기법을 사용해서 게임 공간이 커지더라도 속도가 저하되지 않는다[4]. 사용자와의 상호작용에 있어서는 Myo 기기의 센서를 통해 사용자의 손동작을 측정한다. Myo 기기는 사용자가 주어진 공간 내를 이동하며 실제적인 신체 동작을 통한 시스템과의 소통을 가능하게 한다. 또한 영상 내 사용자의 손을 검출하는 시스템에 비해 많은 계산 량이 요구되지 않는다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 시스템의 구성에 대하여 설명한다. 3장에서는 제안하는 시스템의 구현에 대하여 설명하며, 4장에서는 제안하는 시스템의 실험 결과에 대해 설명한다. 마지막으로 5장에서 결론을 맺는다.

2. 구성

그림 1은 본 연구에서 제안하는 시스템의 구성도를 나타낸다. 제안 시스템은 크게 세 가지 부분으로 구성되며, 모바일 기기, 서버 컴퓨터, 마지막으로 Myo 기기이다. 먼저 모바일 기기는 모바일 기기의 카메라를 통해서 한 장의 영상을 얻는다. 그리고 얻은 영상을 네트워크를 통해 서버로 전송한다. 서버는 받은 영상을 계산을 한 후에 그 계산을 통해서 카메라의 위치 정보를 추출한다. 그리고 그 추출된 위치 정보를 다시 모바일 기

기로 전송한다. 모바일 기기는 받은 위치 정보를 통해서 특정한 물체를 화면상에 증강해준다. 이러한 증강된 물체와 사용자는 손동작을 통해 소통을 하는데, 그 때 Myo 기기를 사용한다. Myo 기기는 암밴드 컨트롤러로서 사용자의 근육 전기신호에 따라 서로 다른 동작을 저장하고 인식한다. 이에 따라 수행된 동작에 따른 data와 수행된 동작에 대한 신호 메시지를 모바일 기기로 전송한다. 모바일 기기에서는 사용자의 위치 정보 즉, 현재 영상에 대한 정보와 Myo data를 통해 사용자와 상호작용하는 위치기반 실내 증강 현실 게임을 할 수 있다.

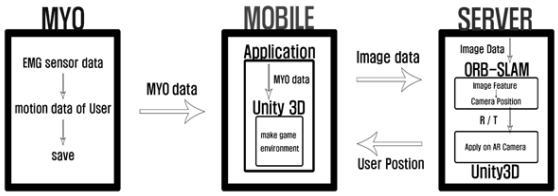

그림 1. 시스템 구성도

2.1 Mobile

모바일에서는 네트워크를 통해 현재 화면의 정보를 영상으로 실시간으로 서버에 전송한다. 서버로부터 받아온 사용자의 위치 정보를 적용시켜 가상 물체를 해당 위치에 증강시키고, Myo 연결 신호에 따라 이 가상물체와 상호작용한다. 모바일과 서버상의 사용자 객체의 상태를 동기화시켜 해당 위치에 따라 증강된 물체를 사용자에게 보여준다. 또한 사용자와 증강 물체 간의 거리에 따른 증강물체 생성시점을 조절하고, 상황에 따른 애니메이션 변화를 적용시킨다.

2.2 Server

서버 시스템은 모바일 기기로부터 시작되며, 현재 사용자가 바라보고 있는 방향의 영상을 입력 받는다. 전송 받은 영상으로 ORB SLAM을 통한 실내 지도를 제작한 다음, 원하는 맵 포인트 좌표에 가상물체를 위치시킨다. 받은 이미지와 실내 지도간의 비교를 통한 계산을 한 후에 그 계산으로 카메라의 위치 정보, 즉 사용자의 위치 정보를 추출한다. 그리고 그 추출된 위치 정보를 다시 모바일 기기로 전송한다.

2.3 Myo

사용자의 손동작에 따라 근육은 다른 모양을 띄는데 Myo 착용으로 이를 감지할 수 있다. Bluetooth를 통해서 모바일 기기와 Myo 기기 간에 연결을 하며, 이를 통해서 정보를 주고 받도록 한다. Myo 데이터는 가공되지 않는 각이나 길이 값만을 반환하기에 Myo 데이터를 보정해주는 작업을 한다. 또한 사용자가 한 번의 행동을 취했을 때 여러 번의 같은 행동 값이 인식되지 않도록 즉, 반복된 Myo 신호가 한 번으로 인식되도록 구현한다. 사용자의 수행된 행동에 따라 서로 다른 동작을 저장하고 인식할 수 있기에, 동작에 따른 응답으로 시스템과 상호작용하여 사용자에게 게임 시스템을 제공한다.

3. 구현

본 시스템은 구성의 분리에 따라 구현 또한 Myo, Mobile, Server 세 파트로 나누어 구현하였다. 먼저 Mobile은 기본적으로 Unity C#을 이용해서 현재 모바일 기기의 카메라로부터 이미지를 받아온다. 받아온 이미지는 JPG 형태로 압축되고, UDP packet 으로 wrapping 되어서 서버로 전송된다. 이미지를 보낸 후 서버로부터 이미지에 따른 카메라 자세 값을 반환 받게 된다. 이 자세 값을 Unity 객체의 정보와 매칭해서 증강현실을 구현한다.

서버는 전송받은 이미지를 ORB-SLAM을 이용해 분석하고, 그에 따른 카메라 자세를 추정한다. 추정된 자세 값은 내부 네트워크를 통해 다른 프로그램인 서버 내의 Unity 프로그램으로 전송된다. Unity는 Unity 네트워크를 통해서 모바일 Unity에게 카메라 자세 값을 실시간으로 공유한다.

마지막으로 Myo는 홀로그래피 라이브러리를 통해서 기기의 측정 데이터를 Bluetooth 통신을 통해 모바일 기기로 전송해준다. 모바일 기기는 Android studio 라이브러리를 사용해서 Myo 측정 데이터를 받고, 분석하여서 적절하게 Unity 의 함수를 호출해주는 형태로 시스템이 동작한다. 그림 2는 본 시스템의 구현 환경을 보여준다.

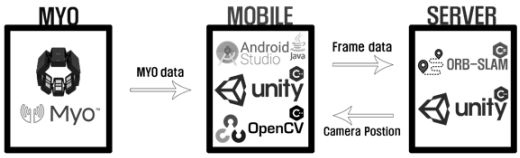

그림 2. 시스템 구현 환경

3.1 Myo 를 이용한 사용자 상호작용 시스템 개발

3.1.1 EMG sensor data를 통한 근육 차트 표현

제안 시스템에서는 팔이 주위를 이동할 때 팔의 가속도에 대한 차트를 제공한다. 이 차트를 제공하기 위해 홀로그래피 라이브러리를 사용하며, 이 라이브러리를 통해 사용되는 근전계 장치의 정보를 기록하고 재생할 수 있게 된다. 홀로그래피 라이브러리는 그래프와 차트를 android 응용 프로그램에 쉽게 통합할 수 있도록 작성된 라이브러리로서, LineGraph view, BarGraph view, PieGraph view 를 포함한다. 보다 정확한 측정을 위해 8

개의 EMG data를 출력하고 각 data 를 line graph 로 나타내었다. 그림 3을 통해 동작에 따라 근육차트가 다르게 제공되는 것을 볼 수 있다.

3.1.2 Myo data에 따른 동작 저장 및 인식

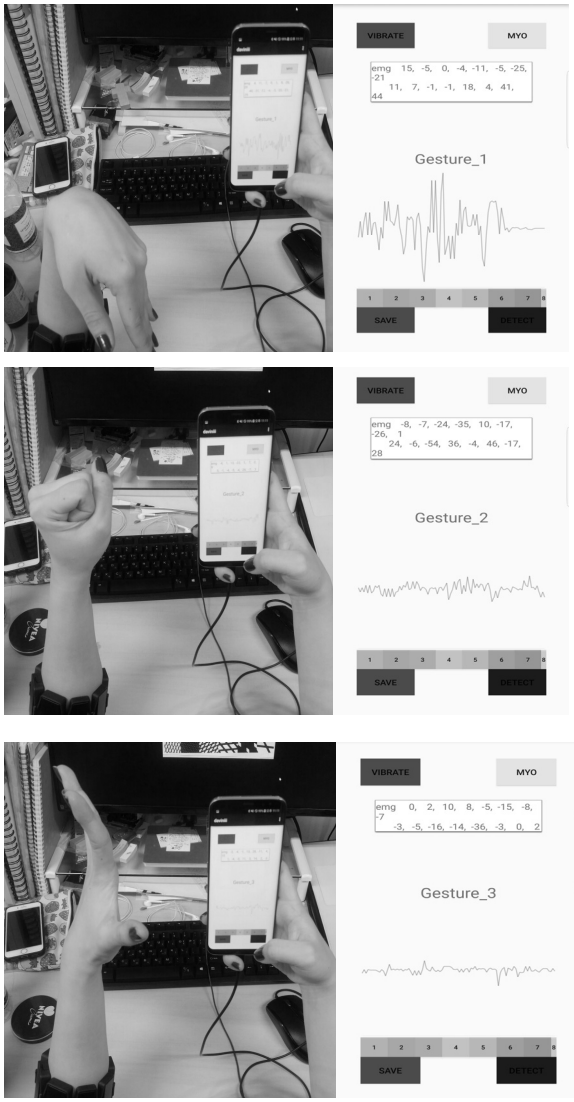

그림3. 손동작에 따른 EMG data와 근육 그래프

제안 시스템에서는 사용자가 특정한 동작을 취했을 때, EMG data가 그에 맞게 변화하며 사용자는 모바일의 공간상에 저장할 수 있다. EMG data에 따른 동작 저장 및 인식은 기존 영상을 통한 손동작 인식보다 사용자가 동작을 정의하는데 있어서 용이하다. 사용자마다 캘리브레이션을 통해 동작을 저장하기에 정확성 또한 높으며, 더 많은 동작을 정의할 수 있다. 최대 저장 가능한 동작은 사용자가 사전에 정의한 만큼 가능하며, 각각의 동작은 data 간의 모호함을 줄이기 위해 확실한 동작을 기준으로 정의해야 한다. 여기서 확실한 동작이란 근육 차트의 8 개의 line graph 상에 모두가 다른 data line 이 그려지는 동작을 뜻한다. 동작을 인식할 때에는 각각의 line에 대한 EMG data 가 들어있는 배열 상에 저장 되어있는 값을 비교함으로써 인식한다. 이렇게 모바일 공간상에 저장된 EMG data 가 사용자가 취한 동작 data의 값들과 일치할 때에 지정된 동작으로 인식하며 모바일 상에 몇 번째 동작인지 나타내어 준다.

3.1.3 Myo 신호에 따른 Unity와 상호작용

Myo 신호가 전송될 경우 사용자에게 실제적인 출력을 주기 위해서는 Unity 와의 상호작용이 필요하다. 하지만 실제로 Myo 입력은 Android studio 상에서 받고, Unity는 Unity Mobile 인터페이스로 한층 더 추상화 되어 있다. 그렇기 때문에 Unity에서 직접 Myo data를 받아서 분석 및 사용을 할 수가 없다. 따라서 Android Studio 상에서 Myo 데이터를 분석한 후에 Unity 내의 대응되는 함수를 호출하는 방법을 통해 Unity 시스템과 상호작용 하도록 구현하였다.

3.2 UDP 및 Unity 를 사용한 영상 전송 프로토콜 개발

3.2.1 출력 중인 영상 변수화

Unity에서 웹캠 또는 내장되어 있는 카메라에 접근하기 위해서 webCamTexture라는 class를 사용한다. 이 class 의 GetPixels() 라는 함수를 통해서 영상 데이터를 가져올 수 있다. 그리고 이때 반환되어서 사용할 수 있는 Unity의 영상 저장 데이터 타입은 Color[] 이다. 영상의 사용을 위해 Color[] 변수를 선언하고 GetPixels() 로부터 받아온 데이터를 선언한 변수에 저장한다.

3.2.2 UDP를 이용한 통신 구현[9]

본 연구에서는 UDP 를 사용하여 Unity의 C# 과 서버의 C++ 에서 각각 UDP를 통해 전송하고 받을 수 있도록 구현하였다. socket 을 만들고 그 socket 을 이용하여 데이터를 전송하도록 만들었다. 전송된 데이터를 서버에서 받게 된다. 기본적인 데이터의 전송과 수신이 원활하게 이루어짐을 확인하였다. 이때 IP의 경우 고정 IP가 없으므로 실험환경인 교내 내부 망에서 주어지는 IP를 사용하도록 설정하였다. 이는 후에 고정 IP가 주어질 경우 외부에서도 접근이 가능하다.

3.2.3 영상을 압축을 통한 UDP 전송

본 연구에서 사용하는 영상의 크기는 640×480으로 해당 크기의 컬러 영상 하나의 크기를 계산하면 921,600byte이다. 반면 UDP 로 전송할 수 있는 65535 바이트이다. 실질적으로 한번에 UDP packet으로 전송하는 것은 불가능하고, 이를 여러 packet으로 잘라서 전송하거나 TCP를 이용해서 byte stream의 형태

로 전송해야 하는 것이다. 하지만 본 논문에서 필요한 전송 속도는 적어도 매 초 15프레임 이상의 영상을 전송해야 하므로 이와 같은 형태로 전송할 경우 속도 저하로 인해서 목표하는 데이터 전송 주기를 맞출 수 없다.

이를 해결하기 위해서 영상 압축 기술을 사용하였다. 우선은 크기를 줄이는 것이 중요하고, 이를 통해 소량의 정보를 잃더라도 대략적인 영상 정보들만 전송이 될 수 있다면 본 연구에 쓰이기에 적합한 영상 압축 기술이라 할 수 있다고 판단하였다. 여러 압축 기술 중에서 JPG 압축 기술을 사용하여 영상을 전송하였다. 실험결과 JPG 압축을 이용하면 기존에 921,600byte 이던 영상의 크기가 34,278byte 정도의 크기로 줄어들게 되고 이를 통해 UDP Packet 하나로 이미지 하나를 전송할 수 있게 된다.

3.2.4 영상 저장

필요할 때 영상을 불러와 처리할 수 있도록 전송받은 영상을 하드디스크 상에 저장한다. 후에 영상을 통해 위치 인식을 할 때, 저장한 영상을 라이브러리를 통해서 불러온 후에 사용할 수 있을 것이다. 그리고 이에 대해서 저장하는 방법 외에 코드 상에서 JPG 압축 파일을 라이브러리 영상 변수 형식으로 변경할 수 있는 방법을 찾을 수 있다.

3.3 ORB-SLAM 을 이용한 사용자 위치인식

3.3.1 영상 변수화

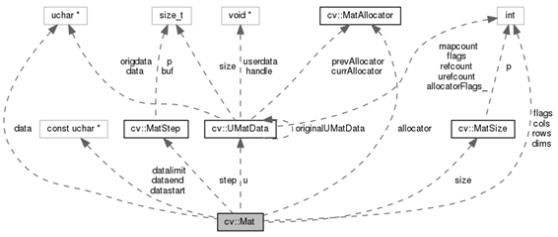

그림 4. Mat Class Reference

그림 4는 Opencv 의 Mat 클래스의 구조이다. Mat 클래스는 OpenCV에서 이미지 처리 시 가장 많이 쓰이는 [n x m] 배열 구조의 데이터 형식으로, 매트릭스의 크기, 저장방법, 매트릭스가 저장되는 주소 정보, 실제 이미지 내 픽셀 값들을 저장하는 데이터 매트릭스를 가리키는 포인터 등을 포함하고 있다. Mat을 통해 한 이미지 내 픽셀좌표와 B, G, R 채널 별 정보를 저장할 수 있으며, 매트릭스 헤더를 복사하고 매트릭스 포인터만 가리키는 방식이기에 이미지 데이터 복사 속도를 개선시킨다. 또한 동적으로 메모리를 할당하기에 자원 낭비를 줄일 수 있다는 장점이 있다. 핸드폰 카메라를 통해 입력 받은 영상을 서버에 JPG 이미지 파일로 압축 및 저장한 후, 이를 서버 라이브러리에서 사용할 수 있도록 Mat 클래스 형식으로 변환한 후 이를 Unity와 SLAM에서 불러온다. 이를 통해 입력 받은 영상과 특징점 기반으로 제작한 실내 지도와의 비교를 통해 회전/이동변환 행렬인 R/T 행렬 값을 추출하여 사용자의 현재 위치를 인식할 수 있다.

3.3.2 ORB SLAM 지도 제작

SLAM은 동시적 위치 추정 및 지도 작성으로 로봇공학 등에서 사용하는 개념으로 임의의 공간에서 이동하면서 주변을 탐색하는 데 쓰이는 기술이다. 다양한 센서를 이용하여 SLAM을 구현할 수 있는데 본 연구에서는 카메라 센서를 이용하는 Visual SLAM의 한 종류로 특징점을 이용하는 ORB SLAM(Oriented FAST and Rotated BRIEF Simultaneous Localization and Mapping)을 사용하여 실내 지도 작성 및 실내 위치 인식을 구현한다.

ORB SLAM 은 처음 시작 위치를 좌표 (0, 0, 0)으로 두고, 이를 기준으로 현재 위치의 좌표를 구한다. 지도를 만들고 위치를 인식하기 위해서는 먼저 하나의 영상에서 ORB 특징점 추출 알고리즘으로 특징점을 추출한다. 그림 5에서 우측 하단의 그림에서 초록색 점은 추출된 특징점을 나타낸다. 연속적인 영상에서 같은 방법으로 특징점을 추출하여 이전 영상과 매칭한다. 매칭한 점을 삼각측량법과 같은 방법으로 3D 좌표 상에 위치시켜 특징점으로 표현된 실내 지도를 작성한다. 이 점들을 현재 프레임에 투사하여 현재 카메라의 위치 및 자세를 찾아내는 방법으로 위치 인식을 한다. 그림 5의 좌측 부분에서 초록색 삼각형은 현재 카메라의 위치 및 자세를 나타내며, 파란색 삼각형들은 연속적으로 추정된 카메라의 위치 및 자세를 나타낸다. 검은색 점들은 추정된 3D 좌표 점들이며, 빨간색 점들은 추정된 3D 좌표점 중 현재 위치에 해당하는 점들을 뜻한다.

그림 5. ORB SLAM

3.3.3 Unity를 이용한 지도 열람

ORB SLAM에서 추출된 특징점들의 3D 좌표 정보를 텍스트 파일로 저장 한 후 이를 Unity 상에 불러온다. 그림 6은 한동대학교 도서관 건물 3층의 SLAM 지도로서, 추출된 특징점의 좌표마다 빨간 박스들을 위치 시켜 가상의 3D 공간 내에 표현하였고, 맨 처음 지도를 생성한 시점을 파란 공으로 표시하였다. 증강 구현 시에는 작성된 지도를 통하여 사용자의 위치와 시점을 파악하고, 증강 컨텐츠를 SLAM 지도 상 원하는 좌표에 위치시킨다.

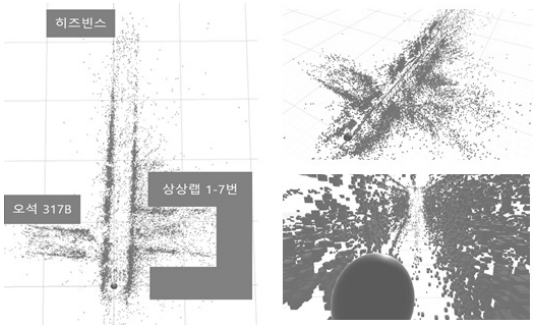

그림 6. 한동대학교 도서관 3층의 SLAM 지도

3.3.4 Unity 네트워크를 이용한 사용자 위치 연동 및 물체 증강

3.3.4.1 전송 받은 R/T값을 통한 가상 공간 내 물체 증강

내부의 UDP 통신을 통해 SLAM의 R/T값을 전송한다. Unity에서 전송된 R/T 값을 받아, 받은 R/T 를 객체에 적용한다. 그림7 같이 가상 공간 내에 물체를 증강하였고, Map point를 추출하여 unity상에 출력하였다.

그림 7. 가상 공간 내 물체 증강

3.3.4.2 네트워크를 통한 사용자 위치 연동 및 물체 증강

유니티 네트워크를 사용하여 서버 유니티와 클라이언트 유니티를 연결한다. 서버 유니티 상에는 Network.InitializeServer() 함수를 통해 서버를 구축하고, 클라이언트 유니티 상에서는 실행 시 서버 유니티와 같은 ip와 port번호에 연결한다. 공통적으로 AR Camera에 Network View Component를 추가하여 OnSerializedNetworkView() 함수를 통해 서버 상의 AR Camera의 위치정보인 R/T 값을 stream.Serialize() 함수를 사용하여 클라이언트에게 송신한다. 그림 8과 같이 실시간으로 네트워크를 사용하여 전송 받은 R/T값으로 사용자의 위치를 연동하였고, unity 상의 가상공간 내에 물체를 심어 두어 사용자의 위치에 따라 물체를 증강시킬 수 있었다.

그림 8. 서버 유니티와 클라이언트 유니티와의 관계

3.3.5 증강된 물체와의 게임 시스템 개발

3.3.5.1 사용자와 증강 물체 간의 거리에 따른 증강물체 생성 시점 조절

사용자가 실내를 탐색할 때, 증강물체와의 거리에 따라 증강물체 생성되는 시점을 조절할 필요성이 있다. 그림 9에서 토끼와 박쥐가 있을 때, 일정 거리에 벗어남에 따라 토끼는 보이고, 박쥐는 보이지 않도록 조절한다. 이를 실내 지도에 적용할 때, 일정 거리에 도착했을 때 증강 물체가 보이도록 하였고, 일정 거리이상 지나갔을 때 사라지도록 하였다. 또한 증강물체에 애니메이션 효과 추가하였다. 공격을 받기 전 물체가 보여질 때에는 물체에 따라 일정한 애니메이션을 추가하여, 공격을 받아 물체가 죽었을 때는 DIE 애니메이션을 만들어 상황에 따른 애니메이션 변화를 적용한다.

그림 9. 사용자와 증강 물체 간의 거리에 따른 증강물체 생성시점 조절

3.3.5.2 게임 효과 및 공격 발사체 조절

사용자와 상호작용을 하는 증강현실 게임이기 때문에, 상호작용에 따른 게임 효과를 추가하도록 한다. 사용자의 손동작은 사전에 Myo를 이용해 3가지로 정의하도록 하였는데, 1번째 동작과 2번째 동작을 할 때에는 공격 발사체가 바뀌도록 하였고, 3번째

동작을 할 때에는 공격을 하도록 설정한다. 공격 발사체는 현재 정의된 동작의 개수에 맞게 총알, 화살, 폭탄으로 사용하였고, 이 공격 발사체들은 화면의 중앙 즉, ARCamera에 bulletpoint를 두어 카메라의 중앙으로부터 발사되도록 한다. 또한 화면에 보여지는 게임물체가 있을 때, 화면의 중앙점을 맞추면 그에 따라 물체의 상태바가 감소하고 게임을 할 수 있도록 구현한다. 이는 사용자가 사전에 정의를 함으로써 변경할 수 있다.

4. 실험 결과

4.1 실내 지도 제작 및 위치 인식

4.1.1 실험 환경

첫 번째 실험으로 제안시스템을 활용하여 영상지도를 사전에 생성하고, 그 후 사용자 단말기의 영상 내 특징점을 활용하여 위치인식을 수행할 수 있는지 확인하였다. 영상 실험을 위한 모바일 기기는 갤럭시 S8을 사용하였다. 모바일 어플리케이션은 Unity를 통해 구현하였고, 통신 기능은 C# 스크립트를 통한 UDP 소켓 라이브러리를 사용하여서 추가하였다. 통신을 위해서는 내부 WIFI망을 사용하였다. 서버는 Intel Core i7-6900k CPU, 32GB RAM 가 장착된 컴퓨터를 사용하였다. 서버 컴퓨터 위에서 C++ 로 구현된 ORB-SLAM 을 실행시켰으며, Unity를 통하여서 사용자와의 위치 통신을 구현하였다. 실험 공간은 한동대학교 도서관에서 진행되었다.

4.1.2 실험 결과

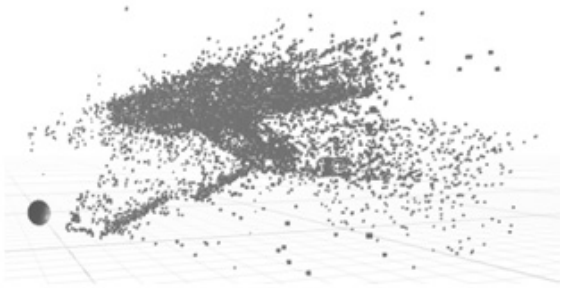

그림 10. ORB-SLAM을 통한 3차원 지도

그림 10은 서버에서 ORB SLAM을 활용하여 생성된 3차원 점들의 모습이다. 3층으로 구성된 지도의 용량은 약 380MB이다. 그림 11은 실제 어플리케이션을 시연하는 사용자의 모습이며, 그림 12는 ORB SLAM으로 계산된 카메라 자세를 모바일 기기로 전송하고, 그 값으로 물체를 증강한 결과 이미지를 나타낸다. 그림 13은 서버 상에서 모바일 기기의 이미지를 통해 현재 사용자의 위치를 추적하는 화면이다. 마지막으로 그림 14는 어플리케이션 실행 시 실시간 진행 환경을 나타낸 결과이며, 사용자가 실내 공간을 돌아다니며 시연을 하는 모습이다. 왼쪽 상단에는 실제 게임 화면이 보이며, 오른쪽 하단에는 생성된 3차원 실내 지도에서 현 사용자의 위치를 나타내는 지도 화면이 보인다.

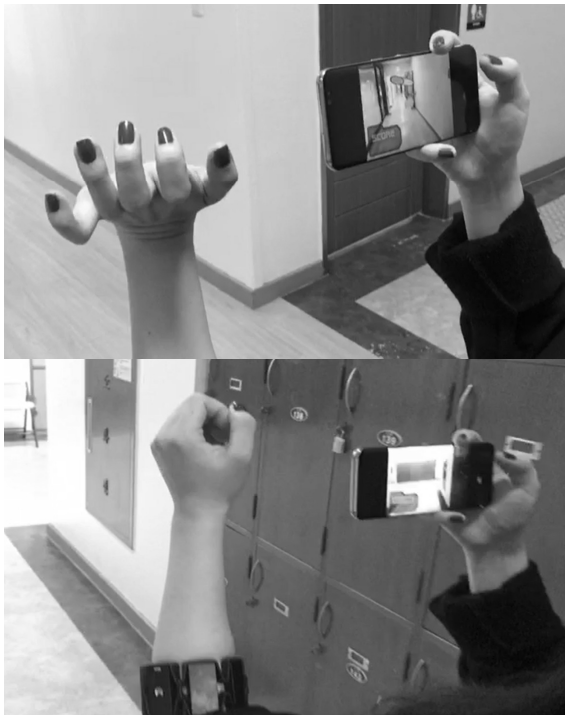

그림 11. 어플리케이션 시연

그림 12. 모바일 상 증강 결과

실험 결과 시스템 작동의 성능 면에서, ORB SLAM은 자세 추정이 안 되는 경우가 종종 발생하였다. 주요 원인은 특징점 매칭문제와 통신 문제이다. 특징점 매칭문제는 ORB 특징점의 특성상 이미지의 블러 현상에서 특징점을 매칭하지 못한다. 또한 조명 변화 또는 물체의 가려짐에 의해 특징 점이 없어질 경우 매칭하지 못한다. 이는 결국, 정확한 카메라 자세를 추정하지 못하는 결과를 나타낸다.

또 다른 문제로 WIFI의 연결이 원활하지 않은 환경에서는,

단말기에서 전송되는 이미지의 UDP 패킷이 전송되지 못한다. 이에 따라 서버의 ORB SLAM에서는 현재 카메라 이미지의 자세를 추정하지 못하게 되며 3차원 점 또한 생성하지 못하게 된다. 그 결과 새로운 이미지의 자세 추정을 위한 3차원 점의 정보가 손실되기 때문에 WIFI가 다시 연결되어도 새로 획득한 이미지의 자세가 정확히 추정되지 못하게 된다. 이를 위하여 ORB SLAM에서는 BOW(Bag Of Words)를 활용한 Re-localization을 수행하여 다시 자세를 추정한다. 이 때 불필요한 계산 시간이 소요되고 카메라 자세 추정을 다시하기 위해 추정을 해왔던 경로로 다시 돌아가야 하는 번거로움이 발생하였다.

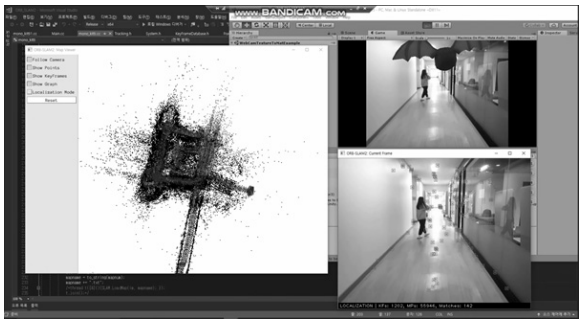

그림 13. 서버 실행 화면

그림 14. 어플리케이션 실행 시 실시간 진행 환경

그림 15는 위치 인식을 위해 소요되는 시간을 히스토그램을 통해 나타낸 것이다. 앞서 언급한 이유로 인해 위치인식에 소요되는 시간은 편차가 있으며, 모바일 기기에서 초기 이미지를 전송한 후부터 서버 상에서 위치를 인식하고 다시 모바일 기기로 카메라 자세 값을 전송하여 적용하는데 까지 평균 0.3초의 시간이 소요되었다.

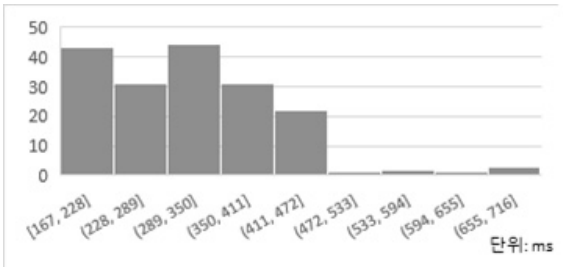

그림 15. 서버 통신으로 인한 Delay 히스토그램

4.2 사용자 손동작을 통한 가상물체와의 상호작용

4.2.1 실험 환경

두 번째 실험으로는 Myo data를 활용하여 증강된 물체와의 상호작용이 원활하게 이루어지는지를 확인하였다. 실험에 사용한 장비는 galaxyS8 와 Myo이다. 증강되는 가상물체는 Unity를 통해 구현하였고, ORB feature를 기반으로 특징점을 매칭하여 증강을 구현하였다. Android Studio를 통해 Myo 어플리케이션을 galaxyS8 기기에 빌드하였다.

어플리케이션에서는 총 3가지 동작을 저장하고 인식할 수 있도록 설정하였다. 사용자가 손동작을 정의하고자 함에 있어서는 사전에 몇 가지 동작을 저장하겠다고 설정할 수 있다. 첫 번째 동작은 주먹을 쥐는 동작, 두 번째 동작은 손바닥을 펴는 동작, 마지막 동작은 손을 안쪽으로 휘는 동작이다. 아래의 그림 16는 왼쪽부터 차례대로 세 가지의 손동작을 나타낸다. 이와 같은 손동작을 설정한 이유는 각 손동작 별로 근육의 EMG센서 값이 분명히 식별 가능한 값들로 나타났기 때문이다. EMG 센서 값이 식별 가능하다면, 사용자가 정의한 여러 가지 동작의 저장 및 인식이 가능함을 확인할 수 있었다.

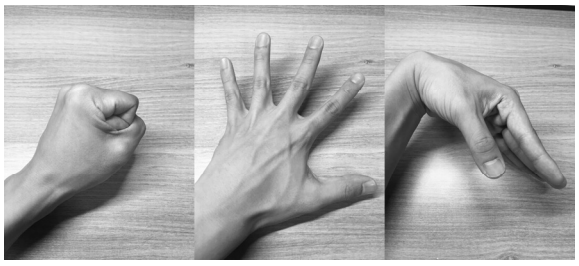

그림 16. 실험 시 사용한 손동작 종류

4.2.2 실험 결과

증강을 제공하기 위해 모바일 디바이스를 통해 실시간으로 입력 받은 영상 속 이미지와 미리 저장된 쿼리 이미지 간의 호모그라피 값을 추출하여 이미지 매칭을 한다. 구해진 호모그라피를 통해서 z축이 0으로 고정되어 있는 임의의 네 점을 투영변환하여 변환된 점을 구한다. 임의의 네 점과 변환된 점을 인자로 사용하여 SolvePnp를 수행하고, 카메라의 자세 값을 추정한다. 이 때 호모그라피를 구하는 과정에서 z축을 0으로 고정해주었고, 호모그라피는 2차원에 대한 투영변환 관계를 나타내는 개념이므로, 증강 물체의 크기는 변하지 않고 x축과 y축 변화에 따른 증강 물체의 위치와 증강 물체가 보이는 면이 달라지게 된다. 이에 따라 증강 물체의 위치가 달라짐을 확인할 수 있으며, 매 프레임마다 물체를 증강하는데 소요되는 시간은 50ms였다.

5. 결론

본 연구의 목적은 위치 인식을 통해 실시간으로 증강현실 환경을 사용자에게 제공하며, 사용자는 증강현실 시스템과 손동작을 이용해서 상호작용하게 하는 것이다. 구현 결과 이미지를 전송하고 전송된 이미지로 인한 카메라의 위치 변화가 적용되기 까지 300ms의 시간이 소요되는 것을 확인하였고, 증강 물체는 실시간으로 증강이 되었다. 상호작용 부분의 경우 게임 실행 이전의 Calibration 파트가 문제없이 실행되어서 사용자에 따라 세 가지의 행동 패턴이 성공적으로 저장이 되었다. 그리고 저장된 행동을 통해 공격 또는 무기 교체가 정상적으로 실행되며, 증강 물체 공격 또한 가능함을 확인하였다.

하지만 실험 결과에서 두 가지 문제점이 발견되었다. 하나는 현재의 이미지를 서버로 보내는 특성으로 인해 발생한다. 이러한 특성으로 인해서 인터넷의 연결 상태에 종속되는 것이다. 두 번째로는 ORB-SLAM이 사용하는 데이터의 특성으로 인해 발생한다. ORB-SLAM은 영상데이터, 특히 ORB-Feature 데이터를 사용해서 식별 가능한 지도를 만든다. 하지만 이로 인해 입력되는 영상 내에 색이나 명도의 차이가 없을 경우, 지도를 만들 수가 없게 되는 것이다. 향후 위 두 가지 문제를 해결할 수 있는 연구가 필요하다.

본 논문을 통해서 얻게 되는 의의는 위치 인식을 통한 다양한 서비스로의 적용 가능성과, 사용자의 직관적인 행동 인식을 통한 서비스 제공이 될 수 있다. 전자의 경우 특별히 장소와 상관없이 모바일 폰을 이용해 위치 인식을 할 수 있다는 점에서 큰 의미가 있다. 기존에 위치를 인식하는 GPS의 경우 실내에서 사용이 불가하고, Wifi를 이용한 위치인식은 Wifi가 설치되어 있는 공간에서만 사용가능 하다는 단점이 존재 한다. 하지만 본 논문에서 사용한 서버를 둔 ORB-SLAM 시스템을 사용한 위치 인식의 경우 지도만 만들어져 있다면, 장소에 관계없이 어디든 현재의 위치를 인식할 수 있다. 그리고 후자는 사용자의 손동작이라는 매우 직관적인 방법을 사용함으로써 사용자의 경험을 고려하는 인터페이스를 제공할 수 있다. 그리고 이는 본 시스템 뿐 아니라 다양 한 시스템의 인터페이스로서 사용될 수 있다.

참고문헌

[1] Talon. E스포츠는 이제 'LOL 스타일'… '145개국, 7000만 명 즐겨'. http://blog.daum.net/_blog/BlogTypeView.do?blogid=0Sa7W&articleno=538&categoryId=20&regdt=20121013084542 2018.06.22.

[2] 김기영, 이민경, 박영민, 이종원, 우운택. ARPushPush: 실내 환경 증강 현실 게임. HCI 2005 학술대회 발표논문집. 한국HCI학회. pp.354-359. 2005.

[3] 김용호, 박종승. PTAM을 이용한 3차원 공간 구성 및 게임 응용. 한국컴퓨터게임학회논문지. 25(4). 한국컴퓨터게임학회. pp. 73-80. 2012.

[4] Mur-Artal, R., Montiel, J. M. M. and Tardos, J. D. ORB-SLAM: A Versatile and Accurate Monocular SLAM system. IEEE Transactions on Robotics. 31(5). IEEE. pp. 1147-1163. 2015.

[5] 김수빈, 배드로, 이상준, 백다빈, 황성수. 증강현실을 사용한 자원 효율적인 정보 제공 인터페이스 연구. HCI 2017 학술대회 발표 논문집. 한국HCI학회. pp. 657-660. 2017.

[6] 김명철, 백다빈, 이성진, 이진규, 최상연, 황성수. ORB SLAM 을 이용한 증강 컨텐츠 서비스 속도 개선. 2017년 한국컴퓨터종합학술대회 논문집. 한국정보과학회. pp. 1948-1950. 2017.

[7] 백다빈, 이성진, 최상연, 이상준, 황성수. 근육 센서를 이용한 상호작용 모바일 증강현실 시스템. HCI 2018 학술대회 발표 논문집. 한국HCI학회. pp. 753-756. 2018.

[8] Silver Moon. UDP socket programming in winsock. http://www.binarytides.com/udp-socket-programming-in-winsock June 22. 2018.