Journal of Institute of Control, Robotics and Systems (2025) 31(5):573-579
https://doi.org/10.5302/J.ICROS.2025.25.0002 ISSN:1976-5622 eISSN:2233-4335

모바일 2D 카메라 기반 실시간 손동작 인식을 활용한 온-디바이스 XR 작업공간 시스템 개발

Development of an On-device XR Workspace System Using Real-time Hand Gesture Recognition With a Mobile Two-dimensional Camera

김 민 호†, 김 현 석†, 이 의 진*
(Minho Kim[1,†], Hyeonseok Kim[1,†], and Yeejin Lee[1,*])

[1]Department of Electrical and Information Engineering, Seoul National University of Science and Technology

Abstract: This paper presents a mobile on-device extended reality workspace system. Unlike conventional augmented reality systems, which offer limited interaction capabilities, the proposed system enables users to intuitively interact with virtual objects in an extended reality environment using hand gestures. The system uses the Unity engine for creating the virtual environment, integrates a deep learning model, and builds an Android application. The deep learning model tracks the user's hand from the video feed, estimates hand joint coordinates, and classifies hand gestures based on these coordinates. The recognized gestures are then converted into control signals for manipulating objects in the virtual environment. The model focuses on efficiency, allowing real-time processing in resource-limited, on-device environments and operates reliably on mobile devices with limited hardware. This study addresses the physical constraints of real-world workspaces by facilitating intuitive interactions with virtual environments, showcasing the practical application of artificial intelligence recognition models in practical scenarios.

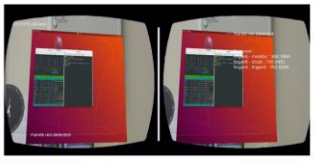
Keywords: eXtended Reality, on-device AI, hand gesture recognition, real-time processing

I. 서론

증강현실(augmented reality, AR)은 현실 세계 위에 디지털 가상 객체를 겹쳐 보여주는 기술로, 사물의 크기, 형태, 위치 등과 같은 물리적 한계를 극복할 수 있는 가능성을 제공한다. 이러한 특성은 원격 로봇 제어와 같은 분야에서 활용될 수 있으며, AR 장비를 착용한 엔지니어가 지구에서 우주에 있는 인공위성 수리 로봇을 원격으로 조작하는 등 공간적 제약을 효과적으로 극복할 수 있도록 돕는다[1].

확장현실(extended reality, XR)은 AR의 개념을 확장한 기술로, 현실과 가상의 경계를 확장하여 가상 객체를 현실 세계에 자연스럽게 융합할 뿐만 아니라, 현실 환경과 가상 환경이 상호작용할 수 있도록 한다. 이러한 XR 기술은 물리적 제약을 가지는 작업공간의 한계를 효과적으로 극복할 수 있는 가능성을 제공한다. 예를 들어, 실세계의 제한된 공간에서는 모니터와 같은 사무용 물품의 크기를 작업 환경에 맞게 맞춰야 하지만, XR 기반 작업공간에서는 가상 환경을 활용하여 공간의 제약 없이 원하는 작업 환경을 자유롭게 구성할 수 있다.

이와 같은 XR 기술에 대한 관심이 높아지면서, 애플은 Vision Pro를 출시하며 XR 서비스를 제품화하여 상용화하였다[2]. 그러나 Vision Pro는 최소 256GB 모델이 약 500만 원에 달하는 높은 가격으로 판매되고 있어, 일반 소비자가 쉽게 접근하기 어려우며, 보급에도 한계가 있다. 이에 본 논문에서는 XR 서비스에 대한 접근성을 높이기 위해 온-디바이스(on-device) 환경에서 실행되는 안드로이드 기반 XR 작업공간 시스템을 구축하고자 한다. 사용자가 보유한 디바이스를 활용하기 위하여, 제안하는 시스템에서는 모바일 디바이스만으로 작동하는



(a) User's perspective

(b) External perspective

그림 1. XR 작업공간 시스템.
Fig. 1. XR workspace system.

† Equal contribution
* Corresponding Author
Manuscript received January 10, 2025; revised January 22, 2025; accepted March 17, 2025
김민호: 서울과학기술대학교 전기정보공학과 석사과정(mhkim98@seoultech.ac.kr, ORCID 0000-0002-1615-4456)
김현석: 서울과학기술대학교 전기정보공학과 석사과정(khslab@seoultech.ac.kr, ORCID 0009-0001-3177-0062)
이의진: 서울과학기술대학교 전기정보공학과 부교수(yeejinlee@seoultech.ac.kr, ORCID 0000-0002-3439-5042)
※ 이 논문은 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원-지역지능화혁신인재양성사업의 지원(IITP-2025-RS-2020-II201741, 50%)과 정부(과학기술정보통신부)의 재원으로 한국연구재단의 지원을 받아 수행된 연구임(RS-2025-00555715, 50%).

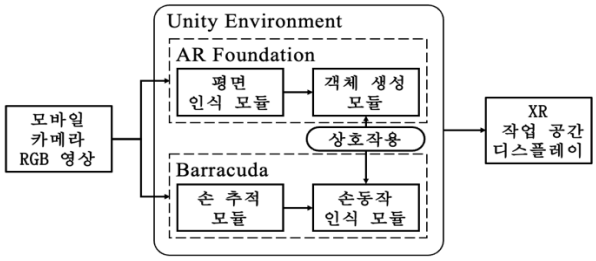


그림 2. XR 작업공간 구성도.
Fig. 2. Configuration diagram of the XR workspace.

표 1. 하드웨어 규격.
Table 1. Hardware specifications.

모델	LG G7 ThinQ(LM-G710N)
프로세서	퀄컴 스냅드래곤 845 SDM845
메모리	4 GB LPDDR4X SDRAM
카메라	후면 1,600만 화소
규격	71.9 x 153.2 x 7.9 mm, 162 g
운영체제	안드로이드 10, VELVET UI

XR 작업공간 시스템을 구축하였으며, 모든 연산을 디바이스 내부 모듈에서 수행하도록 하는 온-디바이스(on-device) 기술을 사용하였다. 온-디바이스(on-device)란 인공지능(AI) 및 데이터 처리를 위한 연산이 외부 서버를 거치지 않고, 디바이스 자체에서 수행되는 기술을 의미한다[4-6].

구체적으로 XR 환경을 구축하기 위하여 제안된 시스템에서는 현실 세계를 모바일 디바이스의 카메라로 촬영하고, XR 화면은 모바일 디스플레이를 통해 사용자에게 제공된다. 그림 1의 (a)는 사용자 시점에서 본 제안된 시스템의 모습을 보여주며, 이를 통해 사용자가 물리적 제약 없이 원하는 작업공간을 자유롭게 구성할 수 있음을 확인할 수 있다. 그림 1의 (b)는 실제 사용자가 시스템을 활용하는 모습을 나타내며, 물리적으로 협소한 공간에서도 효율적인 작업공간을 조성할 수 있음을 보여준다.

또한, 본 논문에서는 Vision Pro와 유사하게 추가 장비 없이 영상 속 사용자의 손을 매개체로 활용하여 가상 환경의 객체를 직접 조작할 수 있는 시스템을 제안한다. 기존에는 가상 환경 내 디지털 객체를 조작하기 위해 컨트롤러[1], 마우스, 또는 근전도 센서[7,8]와 같은 별도의 하드웨어 장비가 주로 사용되어 왔다. 그러나 이러한 방식은 추가 장비 없이는 가상 객체와의 상호작용이 불가능하다는 한계를 가진다. 이를 극복하기 위해 제안된 시스템에서는 모바일 카메라를 활용하여 손동작을 직접 감지하는 방식을 채택하였고, 모바일 카메라는 현실 배경을 촬영하는 기능뿐만 아니라 사용자의 손동작 신호를 실시간으로 감지하는 역할을 수행한다. 본 시스템에서는 RGB 영상을 기반으로 손동작을 인식하기 위해 딥러닝 모델을 적용했으며, 제안된 모델은 사용자의 손을 실시간으로 추적하고 손동작을 인식하여 가상 환경 내 객체를 조작할 수 있도록 설계되었다.

II. 시스템 구성

제안하는 시스템은 Unity 엔진[9]을 기반으로 구축되었으며, AR Foundation [10] 프레임워크, Barracuda [11] 패키지, 그리고 VNC (Virtual Network Computing) [12]을 통합하여 설계되었다. 시스템의 전체 구성도는 그림 2에 제시되어 있으며, 크게 두 가지 주요 모듈로 구성된다. 첫 번째는 모바일 디바이스 내 XR 환경 구축을 위한 모듈이며, 두 번째는 XR 환경에서 디바이스와 상호작용하기 위한 손동작 인식 모듈이다.

본 장에서는 XR 환경 구축 및 가상 객체 생성 방식과 손동작 인식 모델 임베딩 방식을 설명하며, III장에서 손동작 인식 모듈의 구체적인 구현 방식을 다룬다.

1. 가상 환경 구축 및 사용자 인터페이스

본 논문에서 구현한 XR 가상 작업 환경 시스템은 모바일 디바이스에서 원활하게 구동될 수 있도록 안드로이드 애플리케이션 형태로 제작되었다. 특히, 안드로이드 10 이상의 운영 체제를 지원하는 기기에서 실행될 수 있도록 개발되었으며, 실험 및 검증을 위해 LG전자의 G7 ThinQ (LM-G710N) 디바이스를 사용하였다. 해당 디바이스의 주요 규격은 표 1에 요약하였다.

제안하는 시스템은 Unity 엔진을 활용하여 XR 작업공간을 구축한 후, 안드로이드 애플리케이션으로 빌드하여 모바일 디바이스에서 실행되도록 설계되었다. 가상 객체 생성 및 손동작 인식을 포함한 모든 연산은 모바일 디바이스 내에서 실시간으로 처리되며, 이를 통해 외부 서버 의존도를 줄이고 온-디바이스 환경에서의 효율적인 실행을 가능하게 하였다.

2. 실제 환경 인식 및 가상 객체 생성

본 시스템의 사용자 인터페이스(UI)는 사용자가 양안 시점(binocular view)에서 보다 자연스럽게 XR 환경을 경험할 수 있도록 설계되었다. 이를 위해, 그림 3과 같이 화면을 두 개의 시점으로 나누어 각 눈에 맞춘 디스플레이를 제공함으로써 몰입감을 높였다.

또한, 가상 환경 내에 객체의 위치를 고정하려면, 먼저 실제 환경의 평면을 인식하여 AR 환경을 구축해야 한다. 이렇게 인식된 평면은 가상 객체의 기준점 역할을 하며, 사용자의 시점이 변화하더라도 가상 객체가 현실 공간의 특정 위치에 고정될 수 있도록 한다. 제안하는 시스템에서는 AR Foundation을 활용하여 실제 환경의 평면(그림 4의 노란색 영역)을 감지하고, 이를 기반으로 가상 객체(그림 4의 시계)를 위치에 고정한다. 특히, 본 시스템은 단순히 가상 환경에 시계나 달력과 같은 정적인 물체를 생성하는 것을 넘어, 실제 작업 환경에서 활용 가능한 XR 시스템 구축을 목표로 한다. 이를 위해, 가상 데스크톱 모니터를 생성하여

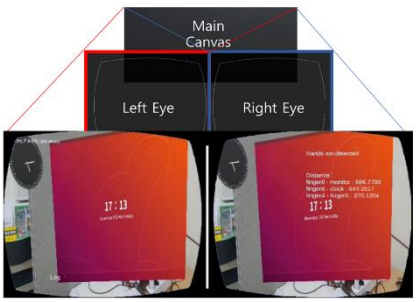


그림 3. XR 작업공간 사용자 인터페이스 구성.
Fig. 3. XR workspace UI configuration.

그림 4. AR Foundation[10]을 사용하여 평면 탐지(노란 영역) 및 가상 객체(시계) 고정.

Fig. 4. Plane detection and virtual object (clock) anchoring using AR Foundation[10].

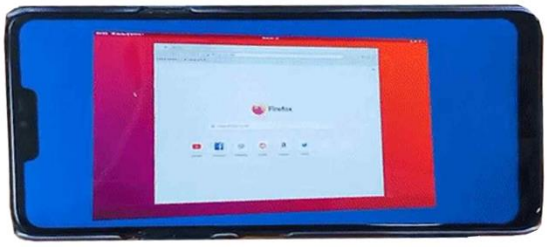


그림 5. VNC[12]를 통한 데스크톱 화면 디스플레이.

Fig. 5. Desktop screen display via VNC [12].

사용자가 원격으로 데스크톱을 제어할 수 있도록 설계하였다. 이와 같은 원격 제어 기능을 구현하기 위해 VNC 기술을 적용하였으며, 그림 5와 같이 가상 환경 내에서 데스크톱 화면을 생성하여 사용자가 XR공간에서 직접 데스크톱을 활용할 수 있도록 하였다. 또한, 모바일 디바이스에 블루투스를 통해 키보드와 마우스를 연결하면 가상 환경 내에서 데스크톱을 조작할 수 있도록 구현하였다.

3. 상호작용을 위한 딥러닝 모델 임베딩

제안하는 시스템에서는 가상 객체와 상호작용을 위한 매개체로 사용자의 손동작을 활용하며, 별도의 추가 장비 없이 영상 내에서 감지된 손동작만으로 가상 객체를 조작할 수 있도록 설계되었다. 이를 위해, 영상에서 손동작을 인식할 수 있도록 딥러닝 알고리즘을 설계하였으며, 해당 모델을 Unity 환경에 임베딩하여 온-디바이스 연산이 가능하도록 구현하였다. 딥러닝 모델을 Unity 환경에 임베딩하기 위해 Barracuda 패키지를 활용하였다. Barracuda는 온-디바이스 환경에서 CPU 및 GPU를 활용한 신경망 연산을 지원하는 패키지로, PyTorch등 기존의 딥러닝 프레임워크에서 ONNX (Open Neural Network Exchange) 형식으로 변환된 모델을 Unity 내에서 직접 실행할 수 있도록 한다.

III. 사용자와 가상 환경 간의 상호작용

제안하는 시스템에서는 사용자의 손을 가상 환경과의 상호작용을 위한 주요 매개체로 활용한다. 따라서, 시스템을 제어하기 위해 손동작을 정의한 후, 영상에서 사용자의 손을 인식하고 손동작을 인식하는 과정이 필요하다. 이를 위해, 영상 내에서 사용자의 손 위치를 추적한 후, 손 관절(joint)의 좌표를 추출한다. 이후, 획득한 손의 관절 좌표를 바탕으로 손동작을 인식하여 가상 환경 내 객체와의 상호작용이 이루어진다.

1. 손동작 정의 및 상호작용 방식

본 논문에서 제안하는 시스템에서는 그림 6에 제시된 5가지 손동작('Expand Size', 'Reduce Size', 'Grab', 'Default', 'Pointing')을 사용한다. 각 동작은 오른손과 왼손 모두 동일한 손가락 순서를 기준으로 관절 번호를 부여하여, 손의 좌우 여부에 관계없이 일관된 방식으로 인식된다. 손동작의 관절 정보는 구글의 Mediapipe hands [13]에서 정의된 관절 포인트 좌표를 활용하였으며, 각 손동작은 그림 7과 같이 21개의 관절 좌표로 구성된다. 'Default' 동작은 가상 객체를 조작하지 않고 대기하는 상태를 나타내며, 해당 손동작을 취한 경우 가상 객체와 상호작용하지 않는다. 반면, 나머지 손동작들은 다음과 같은 방식으로 가상 환경 내 객체와 상호작용을 수행한다.

2. 손동작을 통한 가상 객체 조작

2.1 가상 객체 크기 조정

사용자는 'Expand Size'와 'Reduce Size' 손동작을 통해 가상 환경 내 객체의 크기를 조정할 수 있다. 크기를 조정하고 싶은 객체 위에서 'Expand Size' 또는 'Reduce Size' 손동작을 취하면, 해당 물체의 크기를 점진적으로 확대하거나 축소할 수 있다. 본 시스템은 객체가 독립적으로 동작하기 때문에, 사용자는 가상 공간 내 존재하는 객체의 크기를 개별적으로 조정할 수 있다.

2.2 가상 객체 위치 조정

사용자는 'Grab' 손동작을 통해 가상 환경 내에 존재하는 객체의 위치를 조정할 수 있다. 위치를 조정하려는 객체 위에서 'Grab' 손동작을 취하면, 사용자가 객체를 '잡은' 상태가 되어 이동시킬 수 있는 상태가 된다. 이때 손동작을 유지한 채 손의 위치를 변경하면, 가상 객체도 손의 움직임을 따라 이동하므로 사용자가 객체의 위치를 임의로 변경할 수 있다.

2.3 가상 객체 생성

사용자는 'Pointing' 손동작을 이용하여 가상 환경 내에서 새로운 객체를 생성할 수 있다. 가상 공간에서 'Pointing' 손동작을 취한 상태에서 그림 7의 관절 포인트 4번과 6번 간의 거리를 좁히면 관절 포인트 8번 위치에 새로운 가상 객체가 생성된다.

3. 손 추적 모듈

제안하는 시스템에서는 사용자의 손을 매개체로 가상 환경과 상호작용하기 때문에 영상 내에서 실시간으로 손의 위치를 탐지하고 추적하는 과정이 필요하다. 손 추적 메커니즘은 그림 8에 제시된 단계들을 통해 수행된다.

본 시스템에서는 구글의 Mediapipe hands 딥러닝 모델을 활용하여 손을 추적한다. 이 모델은 RGB 카메라로 입력된 영상에서 손의 위치를 감지한 후, 감지된 위치에서 손 관절 포인트 좌표를 추출하는 것을 목표로 한다. 관절 포인트 좌표는 2.5차원 좌표(x, y, z)로 구성되며, 여기서 x와 y는 2차원 평면상의 위치를, z는 기준점 대비 상대적인 깊이 정보를 의미한다. 따라서, 이 좌표 데이터를 처리하기 위해 포인트 클라우드 데이터 형식으로 구조화한 후 손동작 인식 모듈의 입력으로 사용한다.

4. 손동작 인식 모듈

손동작 인식은 앞서 III장 3절에서 추출한 21개의 손 관절 정보를 이용하여 수행된다.

4.1 손동작 인식 모델

본 논문에서는 III장 1절에서 정의된 손동작을 인식하기 위한 딥러닝 모델을 제안한다. 손동작은 III장 2절에서 획득한 21개의 손 관절 포인트로 구성된 포인트 클라우드 형태로 표현된다. 온-디바이스 AI 환경에서는 메모리 용량과 딥러닝 연산 성능에 제약이 따르기 때문에, RGB 영상을 직접 사용하여 동작을 분류하는 방법보다는 손 관절 좌표 데이터를 활용하는 방식이 계산 복잡도를 낮추고 연산 속도를 개선하는 데 더욱 효과적이다. 이에 따라, 본 시스템의 손동작 인식 과정은 포인트 클라우드 분류 작업과 유사한 방식으로 설계되었으며, 대표적인 포인트 클라우드 분류 모델인 PointNet [14]을 기반으로 최적화된 손동작 인식 모델을 개발하였다.

PointNet은 포인트 클라우드 데이터를 효과적으로 처리하기 위해 설계된 모델로, SMLP (Shared Multi-Layer Perceptron) [14]을 활용하여 개별 포인트의 특성을 추출한 뒤, Max Pooling 연산을 통해 전체 포인트 클라우드의 전역 특성 벡터를 생성한다. 특히, SMLP는 일반 MLP와 달리 손 관절 좌표 데이터에 대해 가중치를 공유하여 학습을 수행함으로써

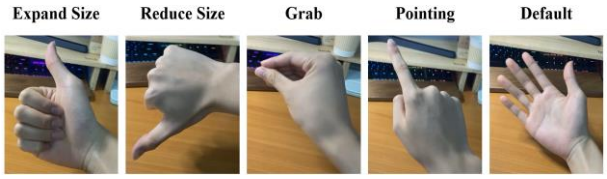


그림 6. 손동작 예시.
Fig. 6. Examples of hand gestures.

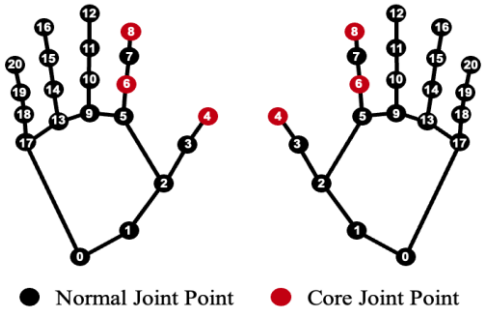


그림 7. Mediapipe hands [13]에서 정의된 손 관절 랜드마크.
Fig. 7. Hand joint landmark defined in Mediapipe Hands [13].

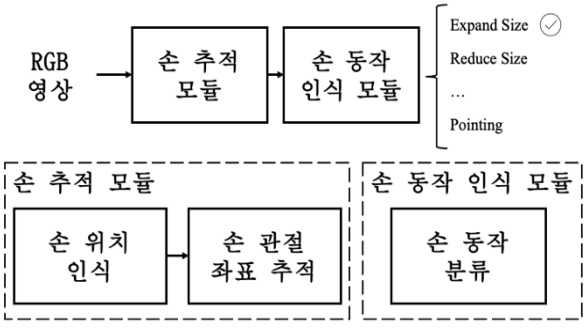


그림 8. 손동작 인식 메커니즘.
Fig. 8. Hand gesture recognition mechanism.

손 관절 데이터의 특성을 효과적으로 추출할 수 있다. 이후, 생성된 특성 벡터는 최종적으로 MLP를 통해 분류 작업을 수행한다. PointNet은 1024개 이상의 포인트로 구성된 포인트 클라우드 데이터를 처리하도록 설계된 모델로, 복잡한 3D 데이터 분석에서 우수한 성능을 보인다. 그러나 본 논문의 손동작 인식 모델은 손 관절 데이터를 기반으로 하며, 단 21개의 포인트만을 입력으로 사용하기 때문에, 실시간성 확보와 연산 효율성을 고려하여 PointNet 구조를 경량화하고, 온-디바이스 환경에 맞게 최적화하였다.

제안하는 모델의 전체 구조는 그림 9에 제시된 바와 같이 구성된다. 설계한 손동작 인식 모델은 21개의 손 관절 포인트 클라우드 $\boldsymbol{p} \in \mathbb{R}^{3\times21}$ 를 입력 데이터로 받는다. 입력된 포인트 $\boldsymbol{p}$는 3층(layer) SMLP를 거쳐, 평탄화(flatten) 층을 통해 포인트 클라우드 특성 벡터 $\boldsymbol{f} \in \mathbb{R}^{105}$ 로 변환된다. 먼저 첫 번째 SMLP 층인 $s_1(\cdot)$ 는 $\boldsymbol{p}$ 를 $\boldsymbol{f}_1 \in \mathbb{R}^{64\times21}$로 변환하고 두 번째 SMLP 층인 $s_2(\cdot)$는 $\boldsymbol{f}_1$를 $\boldsymbol{f}_2 \in \mathbb{R}^{32\times21}$로 변환하며, 마지막 SMLP 층인 $s_3(\cdot)$는 $\boldsymbol{f}_2$를 $\boldsymbol{f}_3 \in \mathbb{R}^{5\times21}$로 변환한다. 각 SMLP 층은 1차원 합성곱(convolution) 층, 1차원 배치 정규화(batch normalization), ReLU(Rectified Linear Unit) 활성화 함수로 구성된다. $\boldsymbol{p}$로부터 특성 벡터 $\boldsymbol{f}$를 추출하는 과정은 식 (1)과 같이 표현된다.

$$\boldsymbol{f} = \text{flatten}(s_3(s_2(s_1(\boldsymbol{p})))) \tag{1}$$

식 (1)에서 구한 특성벡터 $\boldsymbol{f}$ 는 2층의 MLP 층을 거쳐 softmax 층을 통해 손동작 예측 확률 $\hat{\mathbf{p}}$로 변환된다. 첫 번째 MLP 층인 $m_1(\cdot)$ 은 $\boldsymbol{f}$ 를 $\boldsymbol{f}_4 \in \mathbb{R}^{32}$ 로 변환한 뒤 두 번째 MLP 층인 $m_2(\cdot)$ 는 $\boldsymbol{f}_4$ 를 $\boldsymbol{f}_5 \in \mathbb{R}^5$ 로 변환한다. 첫 번째 MLP 층은 선형(linear) 층과 ReLU 활성화 함수로 구성되고 두 번째 MLP 층은 선형 층과 softmax 층으로 구성된다. 예측 확률 $\hat{\mathbf{p}} \in \mathbb{R}^5$를 구하는 과정은 다음 식 (2)와 같다.

$$\hat{\mathbf{p}} = \text{softmax}(m_2(m_1(\boldsymbol{f}))) \tag{2}$$

마지막으로 예측 확률 $\hat{\mathbf{p}}$의 요소들 중 가장 큰 값을 가지는 클래스가 손동작이 입력 데이터의 예측 클래스로 정해진다

손동작 인식 모델을 학습하기 위해서는 크로스 엔트로피 (cross-entropy) 손실함수가 사용된다. 손동작 데이터의 목푯값 (target)은 원-핫 인코딩(one-hot encoding) 방식으로 나타내며, 이를 $\mathbf{t}$라 하고, 배치 사이즈를 N이라 할 때 손실함수 $\mathcal{L}$은 식 (3)과 같이 정의된다.

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{5}\mathbf{t}_{\mathbf{j}}\log\hat{\mathbf{p}}_j \tag{3}$$

4.2 학습 데이터셋

III장 4.1절에서 제안하는 모델을 학습하기 위해 본 연구에서는 딥러닝 모델 학습을 위한 데이터 셋을 직접 수집하였다. 데이터셋을 수집하기 위해 정의된 5가지 손동작을 양손 각 각 촬영한 후, III장 3절의 손 추적 알고리즘을 적용하여 손관 절 좌표를 추출한다. 손동작 데이터 수는 'Expand Size' 5,964개, 'Reduce Size' 5,930개, 'Grab' 5,764개, 'Default' 6,420개, 'Pointing' 5,722개로 총 29,800개의 손동작 데이터를 수집하였다. 수집된 데이터는 두 그룹으로 나누어, 하나는 모델 학습을 위한 그룹으로 사용하고 다른

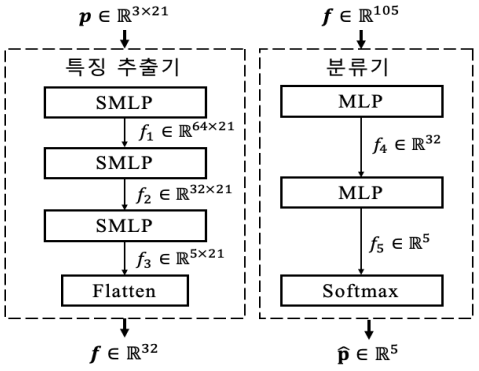


그림 9. 손동작 인식 모델 구조.

Fig. 9. Hand gesture recognition model architecture.

표 2. 손동작 인식 데이터셋 구성.

Table 2. Composition of the hand gesture recognition dataset.

	Train	Validation	Test	Total
# of Sample	18,941	4,736	6,123	29,800

하나는 일반화 성능 평가를 위한 그룹으로 사용한다. 모델 학습을 위한 그룹에서 수집한 데이터는 80%를 학습(train) 데이터로, 나머지 20%를 검증(validation) 데이터로 사용하였다. 일반화 성능 평가를 위한 그룹에서 수집한 데이터는 모두 테스트(test) 데이터로 사용하였다. 데이터셋 구성은 표 2에 정리하였다.

4.3 실험

모든 실험은 NVIDIA RTX 3090 24GB GPU를 사용하여 수행하였다. 딥러닝 프레임워크로는 PyTorch 1.11.0을 사용하였으며, CUDA 11.3 및 cuDNN 8.9 환경에서 실행되었다. 모델 학습을 위해 배치 사이즈 N은 128로 설정하였고, 학습률(learning rate)은 1e-3로 지정하였다. 옵티마이저(optimizer)로는 Adam을 사용하였으며, 손실 함수로는 식 (3)의 크로스 엔트로피 함수를 사용하여 총 50 epoch 동안 학습을 진행하였다. 손동작 인식은 밝은 조명 환경에서 맨손을 사용하고, 한 손을 사용하는 조건하에 수행하였다.

모델 학습이 완료된 후, 최종 모델의 구조 및 가중치는 ONNX 포맷으로 변환하였다. 손동작 인식 모델은 총 6,324개의 파라미터를 가지며, 이는 기존 3,471,054개의 파라미터를 가지는 PointNet 대비 약 0.2% 수준으로, 훨씬 경량화 된 구조를 갖는다. 일반적으로 모델 파라미터는 float32 자료형을 사용하며, 이에 따라 본 모델의 파라미터를 저장하는 데 필요한 용량은 약 24.7 KB에 불과하다. 이러한 경량화는 온-디바이스 환경에서의 실시간 처리와 메모리 효율성을 크게 향상시킨다.

표 3. 손동작 인식 정확도 (%) 및 파라미터 비교.

Table 3. Comparison of hand gesture recognition accuracy (%) and parameters.

Model	Validation	Test	# of parameters
PointNet[12]	99.4±0.4	74.4±5.0	3,471,054
MLP	99.5±0.2	91.1±1.1	6,378
Ours	**99.7±0.2**	**94.9±0.9**	**6,324**

표 4. 왼손, 오른손 손동작 인식 정확도 (%) 비교.

Table 4. Comparison of hand gesture recognition accuracy (%) for left and right hands.

Model	Validation		Test	
	Left	Right	Left	Right
PointNet[14]	99.2±0.6	99.6±0.2	70.0±6.6	78.7±3.7
MLP	99.6±0.2	99.5±0.3	96.0±1.9	86.3±1.9
Ours	99.8±0.3	99.6±0.1	94.5±0.8	95.2±1.5

표 5. 테스트셋에서 손동작별 인식 정확도 (%) 비교.

Table 5. Comparison of hand gesture recognition accuracy (%) by gesture on Test set.

Model	Expand Size	Reduce Size	Grab	Pointing	Default
PointNet [14]	76.0±10.9	66.0±9.6	76.0±4.2	89.8±7.1	64.0±8.9
MLP	95.6±2.1	86.1±4.9	82.4±6.2	99.1±1.2	93.8±4.0
Ours	90.3±1.0	99.4±1.0	92.4±4.2	99.5±0.4	90.3±1.0

제안된 손동작 인식 모델의 성능을 검증하기 위해 본 연구에서는 제안된 모델, PointNet, 그리고 제안된 모델에서 SMLP를 일반 MLP로 대체하여 파라미터 수를 유사하게 조정한 모델을 비교 대상으로 설정하였다. 실험은 서로 다른 10개의 가중치 초기값을 설정하여 반복 수행하였으며, 표 3은 이러한 비교 실험의 결과를 요약하여 평균 ± 표준오차로 제시한다.

실험결과, 제안된 모델은 PointNet 대비 더 우수한 일반화 성능을 보였다. PointNet의 경우, 검증(validation) 데이터셋에서 약 99.4%의 정확도를 달성하였으나, 테스트(test) 데이터셋에서는 74.4%로 성능이 급격히 감소하였다. 이는 PointNet이 학습 데이터에 포함된 사용자에 대해 과적합(overfitting) 되었음을 시사한다. 반면, MLP 모델은 검증 정확도 99.5%, 테스트 정확도 91.1%를 기록하며, PointNet 대비 과적합이 완화되었음을 확인할 수 있었다. 본 논문에서 제안하는 모델은 검증 정확도 99.7%, 테스트 정확도 94.9%를 달성하였으며, 이는 유사한 파라미터 수를 가진 MLP 모델보다도 높은 일반화 성능을 갖는다는 것을 의미한다.

또한, 손의 좌우 구분에 따른 성능 차이를 분석한 결과, 이러한 경향이 더욱 뚜렷하게 나타났다. 표 4는 모델별 왼손과 오른손의 손동작 인식 정확도를 평균 ± 표준오차 형식으로 비교하여 제시한다. 모든 모델이 검증 데이터셋에서는 좌우 손에 대한 성능 차이가 크지 않았으나, 테스트 데이터셋에서는 PointNet과 MLP 모델이 특정 손에 편향되는 경향을 보였다. 특히, PointNet은 왼손과 오른손의 성능 차이가 약 9.5%, MLP 모델은 10.1%로 나타났다. 반면, 제안된 모델은 테스트 데이터셋에서 왼손 99.5%, 오른손 95.2%의 정확도를 기록하였으며, 손 별 성능 차이가 1.5%로 상대적으로 매우 낮았다. 이는 제안된 모델이 손의 좌우 특성을 균형 있게 학습하여, 보다 안정적인 손동작 인식 성능을 제공함을 의미한다.

모델별 손동작 인식 정확도는 각 손동작에 대한 예측 성공 여부를 기반으로 측정되며, 이는 재현율(recall)로 해석될 수

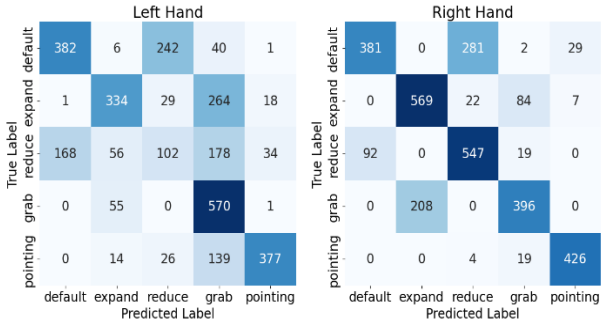


그림 10. 테스트셋에서 PointNet [12] 모델의 왼손과 오른손 손동작에 대한 혼동 행렬 시각화.

Fig. 10. Visualization of the confusion matrix for left and right hand gestures using the PointNet [12] model on the test set.

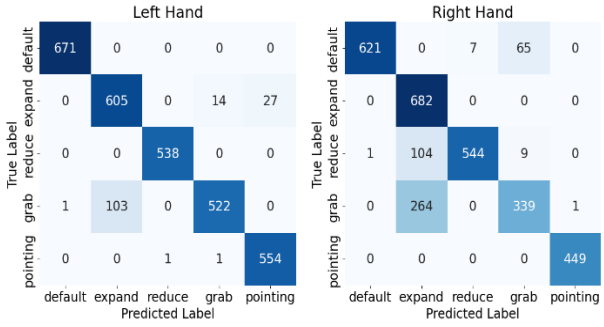


그림 11. 테스트셋에서 MLP 모델의 왼손과 오른손 손동작에 대한 혼동 행렬 시각화.

Fig. 11. Visualization of the confusion matrix for left and right hand gestures using the MLP model on the test set.

있다. 결과는 표 5에 평균 ± 표준오차 형식으로 제시된다. 실험 결과, PointNet 모델은 'Default' 및 'Reduce Size' 손동작에 대해 약 60.0%의 낮은 정확도를 보였다. 특히, 그림 10의 혼동행렬을 통해 PointNet 모델이 'Default'와 'Reduce Size' 손동작 간, 그리고 'Expand Size'와 'Grab' 손동작 간에서 자주 오분류되는 경향을 확인할 수 있다.

MLP 모델은 표 3과 표 4에서 확인할 수 있듯이, PointNet 모델에 비해 전반적인 정확도가 개선되었으나, 'Reduce Size'와 'Grab' 손동작에 대해서는 평균보다 낮은 80.0%대의 정확도를 보였다. 그림 11의 혼동 행렬을 보면, 왼손의 경우 일부 'Grab' 손동작이 'Expand Size'로 예측되었으며, 오른손의 경우 'Grab' 손동작의 절반이 'Expand Size'로 예측되고, 일부 'Reduce Size' 손동작 또한 'Expand Size'로 잘못 예측되었음을 확인할 수 있다. 이로 인해 'Expand Size' 손동작에 대한 예측이 다소 낮은 정밀도(precision)를 나타내는 것으로 해석된다.

마지막으로, 제안된 모델은 MLP 모델에 비해 손동작별로 더 균일한 정확도를 나타냈다. 그림 12의 혼동 행렬에 따르면, 왼손의 경우 'Expand Size' 손동작 일부가 'Grab' 손동작으로 예측되긴 했으나, 전반적으로 모든 손동작에 대해 90.0%가 넘는 정확도를 보였다. 이는 제안된 모델이 뛰어난 일반화 성능을 보임을 시사하며, MLP 모델과의 구조적 차이를 통해 SMLP 연산이 모델의 일반화에 기여하고 있음을 확인할 수 있다. 또한, SMLP 연산이 사용된 PointNet 모델과의 파라미터

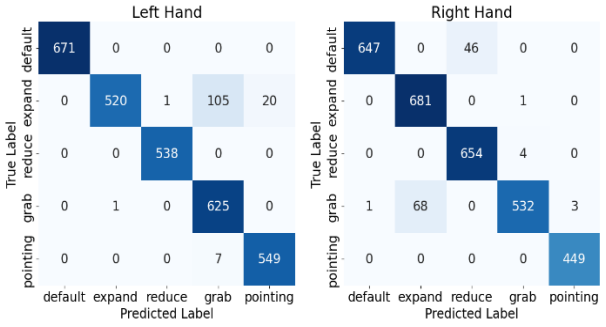


그림 12. 테스트셋에서 제안하는 모델의 왼손과 오른손 손동작에 대한 혼동 행렬 시각화.

Fig. 12. Visualization of the confusion matrix for left and right hand gestures using the proposed model on the test set.

표 6. 앱 실행 시 FPS, Latency 및 CPU 사용률 비교.

Table 6. Comparison of FPS, Latency, and CPU usage during application execution.

Model	FPS	Latency (ms)	CPU Usage (%)
PointNet[14]	10.1	99.0	78.4
MLP	12.1	82.6	75.1
Ours	**12.4**	**82.4**	**67.1**

수 차이를 통해 적절한 파라미터 수가 모델의 일반화 성능 향상에 중요한 역할을 한다는 점을 시사한다.

표 6에는 각 모델을 사용하여 애플리케이션을 실행했을 때의 FPS (Frame per Second), 지연 시간(latency), 그리고 CPU 사용률을 요약하였다. 해당 실험에서 애플리케이션 성능측정에는 Unity Profiler를 사용하였으며, CPU 사용률은 다음의 식을 이용하여 계산하였다:

$$\text{CPU 사용률 (\%)} = \left(\frac{\text{프레임당 CPU 실행 시간 (ms)}}{\text{렌더링 Latency (ms)}}\right) \times 100 \quad (4)$$

실험 결과, 제안된 모델이 기존 모델들과 비교하여 낮은 지연시간과 CPU 사용률을 보이면서도 상대적으로 높은 FPS를 유지하는 것을 확인하였다. PointNet 모델의 경우, FPS가 10.1로 가장 낮았으며, 지연시간은 99.0ms로 가장 높았다. 또한, CPU 사용률이 78.4%로 상당히 높은 편이었다. MLP 모델은 PointNet 보다 향상된 성능을 보였으며, FPS는 12.1로 증가하고, 지연시간은 82.6ms로 감소하였으며, CPU 사용률은 75.1%로 다소 감소하였다. 반면, 제안된 모델은 가장 높은 FPS(12.4)를 기록하였으며, 지연시간은 82.4ms로 가장 낮고, CPU 사용률도 67.1%로 가장 낮게 측정되었다. 이러한 결과는 제안된 모델이 비교 모델에 비해 연산 효율성이 뛰어나며, CPU 부하를 감소시킴을 보여준다.

4.4 한계점 및 향후 연구 방향

본 논문에서 사용한 손 추적 모듈은 Google의 Mediapipe를 기반으로 하고 있다. Mediapipe는 뛰어난 성능과 편리한 사용성을 제공하지만, 조도가 낮은 환경이나 사용자가 장갑을 착용한 경우 손 인식 정확도가 크게 저하되는 한계가 있다. 이러한 환경적 제약을 극복하기 위해 다양한 조명 조건에서의 손동작 인식을 강화할 수 있는 데이터 증강 기법 및 도메인 적응(domain adaptation) 기법의 적용을 고려할 수 있다.

제안 시스템의 주요 한계점은 제한된 환경에서 동작하는 애플리케이션과 딥러닝 모델의 성능 제약에 있다. 특히, 상용화 수준의 FPS를 달성하는 것은 어려운 과제로, 접근성을 강점으로 하는 본 시스템의 한계로 작용할 수 있다. 따라서 향후 연구에서는 애플리케이션 최적화 및 딥러닝 모델의 양자화를 통한 경량화 등의 방법을 고려하여 성능을 개선할 수 있을 것으로 예상한다. 최근 NPU가 탑재된 모바일 디바이스가 개발되면서 온-디바이스 방식의 인공지능 연산이 더욱 효율적으로 수행될 수 있게 되었다. 본 연구에서 사용한 모바일 디바이스는 CPU 기반으로 동작하기 때문에, NPU를 활용할 경우 손동작 인식 메커니즘 전체의 연산 속도가 향상될 것으로 기대할 수 있다.

또한, 현재 손 좌표 추정은 2D 이미지를 기반으로 수행되며, 2D 이미지 특성 상 깊이 값은 실측이 아닌 추정 값을 사용하고 있다. 향후 깊이(depth) 카메라를 도입할 경우, 보다 정확한 깊이 추정이 가능해져 손동작 인식 성능이 더욱 향상될 것으로 예상된다.

IV. 결론

본 연구는 모바일 디바이스에서 동작하는 온-디바이스 XR 작업공간 시스템을 제안하였다. 기존의 제한적인 상호작용만을 제공하던 AR 시스템과 달리, 본 시스템은 사용자의 손동작을 매개체로 가상 환경 내 객체와 상호작용할 수 있도록 설계되었다. 이를 위해 Unity 엔진을 활용하여 가상 환경을 구축하고, 딥러닝 모델 임베딩과 안드로이드 애플리케이션 빌드를 수행하였다. 제안된 딥러닝 모델은 영상 내에서 사용자의 손을 추적하여 손 관절 좌표를 추정하며, 추정된 좌표를 기반으로 손동작 클래스를 분류하였다. 인식된 손동작은 가상 환경 내 객체를 조작하는 신호로 사용되며, 모델은 온-디바이스 환경에서 실시간 처리가 가능하도록 효율성을 최우선으로 고려하여 설계되었다. 이러한 설계는 제한된 하드웨어 자원을 사용하는 모바일 디바이스에서도 안정적인 성능을 보여주었다. 현실 작업공간의 물리적 한계를 극복하고 가상 환경과의 직관적인 상호작용을 가능하게 함으로써, 본 연구에서 제안한 인공지능 인식 모델이 실질적인 응용 분야에서 활용될 수 있음을 입증하였다.

REFERENCES

[1] F. Garcia-Luna, A. Rodriguez-Ramirez, M. Nandayapa and A. Flores-Abad, “Augmented reality-based robotic system for in-space servicing,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 39, no. 1, pp. 18-31, Jan. 2024.

[2] Vision Pro, https://www.apple.com/kr/shop/buy-vision/apple-vision-pro.

[3] Google Cardboard, Google, https://arvr.google.com/cardboard/.

[4] P. Dai, Y. Zhang, T. Liu, Z. Fan, T. Du, Z. Su, X. Zheng, and Z. Li, “HMD-Poser: On-device real-time human motion tracking from scalable sparse observations,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 874-884, 2024.

[5] W. Gou, Z. Yi, Y. Xiang, S. Li, Z. Liu, D. Kong, and K. Xu, “SYENet: A simple yet effective network for multiple low-level vision tasks with real-time performance on mobile device,” *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 12182-12195, 2023.

[6] J. Y. Yun and P.E. Kim, “Query-based Object-aware Mapping for On-device Visual Language Mapping and Navigation,” *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 30, no. 10, pp. 1068 - 1075, Oct. 2024.

[7] M. Y. Yim, D. I. Han, and S. Y. Lee, “Development of a hand-operated wearable mouse for virtual reality,” *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 29, no. 11, pp. 893-900, Nov. 2023.

[8] Y. U. Jo and D. C. Oh, “Study on the EMG-based grasp gesture classification using deep learning and application to active prosthetics,” *Journal of Institute of Control, Robotics and Systems (in Korean)*, vol. 28, no. 11, pp. 1067-1073, Jul. 2022.

[9] Unity Engine, https://unity.com/.

[10] AR Foundation, https://unity.com/solutions/xr/ar.

[11] Barracuda, https://github.com/Unity-Technologies/barracuda-release.

[12] T. Richardson, Q. Stafford-Fraser, K. R. Wood and A. Hopper, “Virtual network computing,” *IEEE Internet Computing*, vol. 2, no. 1, pp. 33-38, Jan.-Feb. 1998.

[13] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. L. Chang, and M. Grundmann, “MediaPipe hands: On-device real-time hand trackin,”ArXiv, abs/2006.10214.

[14] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 652-660, 2017.

김 민 호

2024년 서울과학기술대학교 전기정보공학과 학사. 2024.03~현재 서울과학기술대학교 전기정보공학과 석사과정 재학 중. 관심분야는 컴퓨터 비전, 멀티 모달 학습, 제로-샷 학습.

김 현 석

2024년 서울과학기술대학교 전기정보공학과 학사. 2024년 서울과학기술대학교 ICT 인공지능 전공 학사(복수전공). 2024. 03~현재 서울과학기술대학교 전기정보공학과 석사과정 재학 중. 관심분야는 3차원 컴퓨터 비전, 포인트 클라우드 분할, 제로-샷 학습.

이 의 진

2017년 University of California, San Diego, Electrical and Computer Engineering 박사. 2018년 University of California, Los Angeles, Radiology, 박사 후 연구원. 2018년~현재 서울과학기술대학교 전기정보공학과 전임교원. 관심 분야는 머신러닝, 로봇비전, 컴퓨터비전.