# Mediapipe를 활용한 핸드트래킹 키오스크 시스템

노현수, 김정재, 원종운, 임희호, 이지윤, 정순호
부경대학교 컴퓨터공학과
e-mail : mertiz36@naver.com

# Hand Tracking Kiosk System Using Mediapipe.

Hyun-Soo Noh, Jung-Jae Kim, Jong-Un Won,
Hee-Ho Lim, Ji-Yoon Li, Soon-Ho Jung
Computer Engineering, Pukyong National University

요 약

카메라에 촬영된 이미지를 바탕으로 Mediapipe를 활용해 손을 식별하여 키오스크를 직접 만지지 않고 제어가 가능하게 함으로써, Covid-19 Pandemic 상황에 감염원을 제거하여 사용자의 코로나 감염 확률을 최소화 시킬 수 있다.

## 1. 서론

2021년 8월 7일 기준 Covid-19의 전세계 신규 확진자는 약 63만명, 7일 평균 확진자의 수는 대략 60만명이다. 이것은 [그림 1]과 같이 2020년 3월 기준 전세계 확진자의 수가 대략 1만명 이하였을 때에 비해 엄청나게 늘어난 상황이다. 이러한 Covid-19의 주된 전파경로는 감염자의 비말(飛沫)에 의한 전파이다. 비말은 호흡기 간의 전파뿐만 아니라 사람이 사용하는 물건들에 묻어 전파될 수 있다.

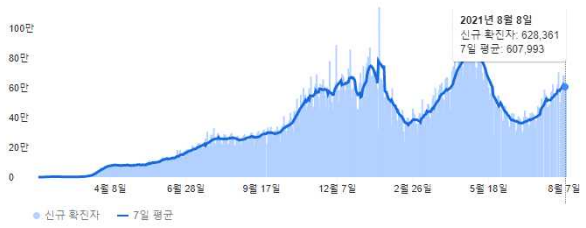


그림 1. 2020년부터 2021년 8월까지의 신규 확진자의 수

이 논문에서는 불특정 다수가 이용하고, 어느 매장에서나 자주 볼 수 있는 키오스크에 주목한다. 영상 처리 및 인식 기술을 통해 손의 제스처를 감지하여 이용자가 키오스크를 직접 만지지 않고 시스템을 제어할 수 있도록 시스템을 개발하여 연구하고 결과를 도출하여 Covid-19 전파를 최소화하는 방법을 고찰한다.

## 2. 관련 연구

이 논문에서는 실시간으로 손의 특징점을 찾아주는 Mediapipe Framework의 Hand Tracking 모듈을 사용한다[1]. 이 모듈은 먼저 손바닥을 검출하기 위해 BlazePalm이라 불리는 Single-Shot Detector Model(이하 SSD)을 사용한다. 이 SSD 모델은 NVIDIA Titan에서 512×512 크기 입력의 경우 76.9%의 높은 mAP(mean Average Precison)와 59 FPS를 달성했다. 즉, 가볍고 빠른 모델을 사용하여 핸드 디텍션을 수행하며, 이는 Mediapipe에서 95.7%의 정확도를 달성함을 보인다[2]. 이렇게 검출한 이미지 영역에서 ML을 통해 손의 특징점을 찾아 21개의 Hand Landmark를 디텍션한다. 이를 활용해 본 논문에서는 손의 제스처를 인식해 동작을 수행하는 키오스크를 설계한다.

## 3. 핸드트래킹 키오스크 시스템

본 시스템은 [그림 2]에서 보는 바와 같이 비접촉 상황에서 손 제스처로 Kiosk를 조작해 주문-처리 프로세스를 수행하는 것이 목적이다.

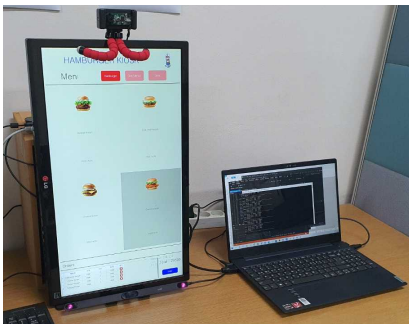

그림 2. 핸드트래킹 키오스크 시스템의 실제 모습

이 시스템의 구현은 PyQt 기반의 Kiosk-UI와 Python 기반의 Mediapipe Framework를 활용하여 핸드트래킹을 통해 손을 탐지하고 제스처를 인식하여 사용자가 키오스크를 만지지 않고 제어할 수 있게 함으로써 이루어진다. 이에 대하여 H/W구성과 S/W구성을 설명하면 아래와 같다.

### 3.1 H/W 구성

이 시스템의 H/W구성은 [그림 3]와 같이 윈도우 os 디바이스와 손 인식을 위한 웹캠 그리고 디스플레이(모니터)로 이루어져 있다. 웹캠의 이미지를 디바이스가 손을 인식하여 각 변수로 받은 후 UI의 컴포넌트들을 제어한다.

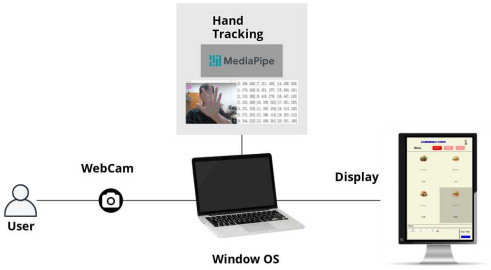


그림 3. H/W 구성도

### 3.2 S/W 구성

본 시스템의 동작을 [그림 4] 시퀀스 다이어그램으로 구현하였다. 키오스크 사용을 위해 키오스크 상단의 카메라에서 실시간 영상을 트래킹 시스템에 전송하여 사용자의 손 동작을 인식하고, 인식된 손 동작에 따라 키오스크 UI의 컴포넌트에 수행 명령을 전달한다. 이러한 동작의 반복으로 키오스크 사용을 구현하였고, 주문을 완료하여 결제 이후 멤버십을 QR코드를 통해 적립을 하게 된다.

이를 [그림 5] 제어 흐름도로 표현하면 아래와 같다.

1) Python 기반 OpenCV 라이브러리를 통해 웹캠의 이미지를 받아, 2) Mediapipe를 통해 손의 정보들을 인식하고 [1] 이를 변수로 받아 3) PyQt 기반의 키오스크 UI를 제어한다. 또한, 4) 전용 QrCode 애플리케이션을 통해 사용자는 멤버십 포인트를 적립할 수 있으며 5) 해당 정보는 Firebase 기반의 데이터베이스에 저장된다.

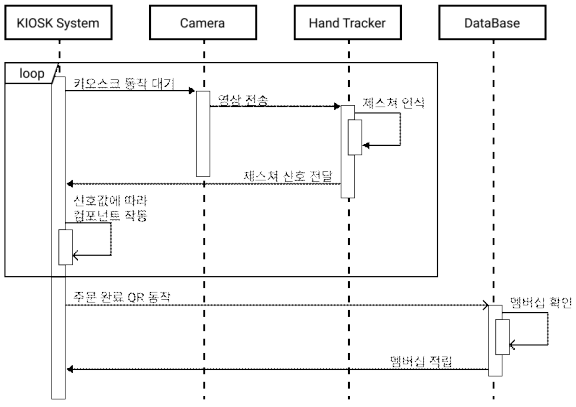


그림 4. 시스템 시퀀스 다이어그램

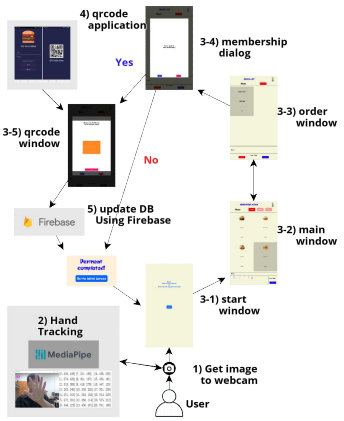


그림 5. 전체적인 시스템의 제어 흐름

#### 3.2.1 Hand Tracking of Google Mediapipe

Google 사의 Mediapipe Framework에서 제공하는 Hand Tracking Module은 GPU가 없는 일반적인 CPU 환경에서도 실시간 detecting이 보장되는 것이 특징이며 크로스 플랫폼을 지원한다. 동작 방식은 다음과 같다. 카메라에서 RGB 채널을 가진 이미지를 받아와 1) 손바닥을 감지하고(bounding box), 2) 손의 landmark 모델을 예측한다. 이처럼 pipelining을 통한 동작 결과 [그림 6]과 같이 손의 landmark를 감지한다.

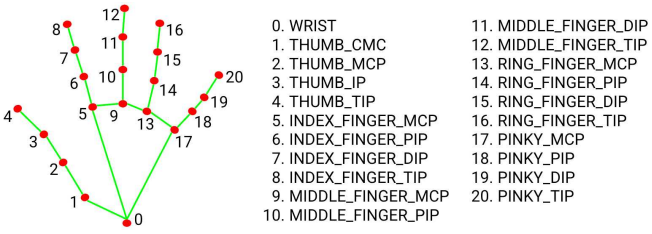


그림 6. Mediapipe's hand landmark

Kiosk에 사용자 손의 정보(손가락을 펼친 개수, 인식된 손의 개수, 클릭 여부, 손가락의 방향 등)를 전달하기 위한 landmark 정의가 필요하다. [그림 6]을 기준으로 landmark의 위칫값, 점들 사이의 각도, 점들 사이의 거리를 정의해 손의 정보를 인식하도록 한다[3]. 예를 들어, 4번 (THUMB_TIP)과 8번(INDEX_FINGER_TIP) 사이의 거리가 기준 거리 미만일 때, Kiosk는 받은 손의 정보를 click이라 인식한다. 아래의 [그림 7]은 손가락을 펼친 동작에 대한 손가락 개수 인식 테스트이고, [그림 8]은 인식된 이미지에 따라 키오스크에 전달되는 데이터의 구조(딕셔너리)이다.

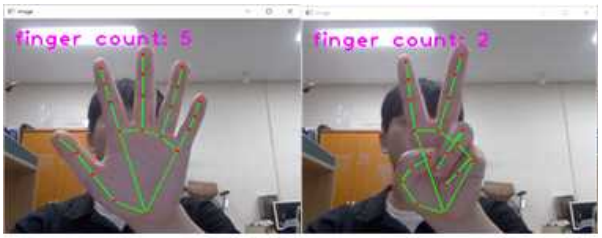


그림 7. Finger Counting using Hand Tracking

```
data = {'is_valid': is_valid,
        'fingers': stretch_fingers,
        'is_action_check_gesture': is_action_check_gesture,
        'direction_of_hand': direction_of_hand,
        'direction_of_hand_using_position': direction_of_hand_using_position,
        'num_of_detected_hands': num_of_detected_hands,
        'num_of_stretch_out_fingers': num_of_stretch_out_fingers
}
```

그림 8. Hand Tracking's Dictionary Data

본 시스템은 Kiosk UI 컴포넌트에 핸드트래킹의 데이터들을 연동하여 동작하며, 핸드트래킹 제스처의 분류는 [표 1]과 같다.

| Window | Gesture | Note |
|---|---|---|
| main_window |  | 오른쪽 카테고리로 |
| |  | 왼쪽 카테고리로 |
| |  | 1, 2, 3, 4번 메뉴 포커스 |
| |  | 포커스 메뉴 선택(okay) |
| |  | 다음 화면 (order_window) |
| order_window |  | 이전 화면 (main_window) |
| other_window |  | 시작 버튼 클릭 Yes 버튼 클릭 |
| |  | No 버튼 클릭 Cancel 클릭 |
| all_window |  | 트래킹 초기화 제스처 |

표 1. 핸드트래킹 가이드

### 3.2.2 Kiosk UI Using PyQt and Firebase DB

본 시스템에서는 C++을 기반으로 한 Qt의 레이아웃에 Python 코드를 연결하여 GUI 프로그램을 만들 수 있게 해주는 Framework인 PyQt를 이용해 UI를 구축한다.

본 시스템의 UI는 [그림 2]와 같이 시작화면, 메인화면, 주문목록 화면, QR 코드 인식화면 등으로 이루어져 있다.

첫 번째, 시작화면(start_window)은 Base Window 초기 상태에 선언되어 있다. Base Window가 실행되거나 초기화될 시에 해당 Window가 실행된다. 구성은 간단한 UI 사용 가이드와 시작 버튼으로 구성되어 있다.

두 번째, [그림 9] 좌측의 메인 화면( main_window)은 카테고리 선택 및 메뉴를 선택할 수 있는 화면이다. PyQt의 경우 메인에서 최초로 실행된 창이 종료될 시 다른 창의 종료 여부와 상관없이 프로그램이 종료되므로 메인화면(Base Window)은 결제가 완료되어도 초기 상태로 돌아갈 뿐 종료되지 않으며 계속해서 실행되어 있다. 메인화면 이후의 창들은 모두 메인화면에서 처리한 주문 Table Widget을 변수로 받아 그 값을 사용해 포인트 적립 및 영수증을 출력한다.

세 번째, [그림9] 우측의 주문목록 화면(order_window)에서는 메인화면의 Table Widget 목록을 크게 확인할 수 있다. 각 메뉴를 클릭 시 테이블에 담긴 메뉴의 개수를 하나씩 차감하는 형식으로 구성되어 있다.


그림 9. main_window(좌), order_window(우)

네 번째, [그림 10]처럼 멤버십을 위한 QR 코드 인식 화면(qrcode_window)에서 전용 애플리케이션으로 생성된 Qr Code 이미지를 읽으면 Firebase를 기반으로 한 데이터베이스에 포인트가 적립된다.

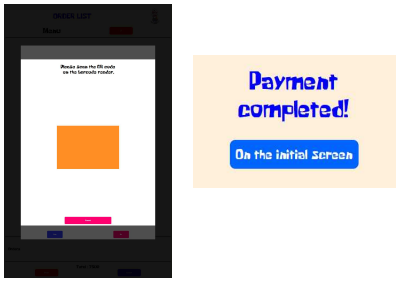


그림 10. qrcode_window(좌), paying_window(우)

마지막으로 결제가 완료되며 영수증을 출력한 후 초기화면으로 돌아간다. 이때 Base Window가 초기화되며

main_window 초기 상태에 선언된 start_window가 같이 실행된다.

## 4. 실험 및 평가

Mediapipe Framwork는 CPU 환경에서도 평균 30 FPS 이상의 준수한 성능과 95% 이상의 Precision :TP/(TP+FP)을 보인다[1][4]. 이러한 Mediapipe Framwork를 기반으로 [그림 11]과 같이 일부 제스처를 실험해본 결과 [표 2]의 정확도를 얻었다.

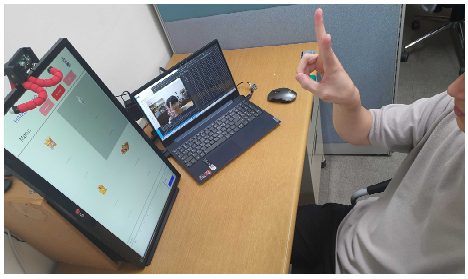

그림 11. 제스처 테스트

* 단위: %(100회 기준)

| 제스처 \ 시도 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| (1) | 62 | 68 | 58 | 64 |
| (2) | 56 | 62 | 60 | 66 |
| (3) | 60 | 64 | 64 | 62 |
| (0) | 68 | 66 | 70 | 66 |

표 2. 제스처 최초 테스트 정확도

시스템의 동작을 확인해본 결과 연속적인 프레임을 분석함에 있어 각 제스처 사이의 중간 동작들을 판별할 때 오류가 발생함을 확인하였다. 이를 해결하기 위해 손동작을 인식하기 위한 시간(1초)을 설정하는 것과 초기화에 해당하는 제스처(다섯 손가락 펼치기)를 설정하는 것으로 처리한다. 이를 통해 최초 테스트 결과보다 높은 [표 3]의 정확도를 얻을 수 있었다.

* 단위: %(100회 기준)

| 제스처 \ 시도 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| (1) | 92 | 90 | 93 | 94 |
| (2) | 96 | 94 | 90 | 91 |
| (3) | 89 | 93 | 91 | 95 |
| (0) | 92 | 95 | 91 | 97 |

표 3. 수정 후의 테스트 정확도

이렇게 수정된 트래킹 시스템을 사용 시 사용자는 약 1초의 인식 시간을 거친 후 시스템을 제어할 수 있다. 실제로 사용 시 사용자는 거의 실시간으로 키오스크를 제어할 수 있다. 여기서 배경과 손을 더욱 정확히 구분하여 처리할 수 있다면 인식 시간을 더욱 단축할 수 있을 것이다.

## 5. 결론

본 논문에서는 Mediapipe를 이용한 핸드트래킹 키오스크 시스템을 구현해보았다. 다섯 손가락을 펼친 제스처인 트래킹 초기화 제스처 추가 및 손동작 인식 시간 설정 등으로 인해 많은 엣지 케이스들을 제거함으로써 높은 손동작 인식률을 보였다. 이에 배경과 손의 정확한 구분, 손의 다양한 각도에 대한 추가 알고리즘 등을 적용한다면 더욱 빠르고 정확한 핸드트래킹이 가능할 것이다.

사용자가 직접 만져서 제어할 수 있는 모든 경우를 핸드트래킹의 제스처로 설정하여 키오스크를 만지지 않고 쉽게 제어할 수 있게 되었으므로 Covid-19의 감염원을 줄여, 확진자 수 감소에 이바지할 수 있을 것으로 예상한다.

## 참고문헌

[1] Fan Zhang, Valentin Bazarevsky, Andrey Vakunov, Andrei Tkachenka, George Sung, Chuo-Ling Chang, Matthias GrundmannValentin, MediaPipe Hands: On-device Real-time Hand Tracking, Department of Computer Science at Cornell University, 2020 Presentation, pp 1-5

[2] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu Alexander C. Berg, SSD: Single Shot MultiBox Detector, University of Michigan, 29 Dec 2016, pp 1-17

[3] Valentin Bazarevsky, Fan Fan Zhang, On-Device, Real-Time Tracking with MediaPipe, Google AI Blog, August 19, 2019

[4] Peter A, Flach, Meelis Kull, Precision-Recall-Gain Curves: PR Analysis Done Right, Department of Computer Science at Bristol University, 2015, pp 1-9

※ 본 논문은 과학기술정보통신부 정보통신창의인재양성사업의 지원을 통해 수행한 ICT멘토링 프로젝트 결과물입니다.