OpenCV를 활용한 마스크 착용 확인 시스템 설계 및 구현

황서진*, 문혜정**
*싱가포르한국국제학교
**세종대학교 외식경영학
thomas0519.hwang@gmail.com, hyejung.moon@sejong.ac.kr

A Design and Implementation of Mask Wearing Face Detection System by OpenCV

Seo-Jin Hwang*, Hye-Jung Moon**
*Singapore Korean International School
**Dept. of Foodservice Mgmt., SeJong University

요 약

코로나바이러스로 인하여 전 세계는 어려움을 겪고 있으며, 바이러스를 확산을 막기 위해서 실외에서는 마스크를 쓰는 것이 일상이 되었다. 하지만, 이를 따르지 않는 사람이 일반 시설에 방문할 때 이를 감지하고 경고를 할 수 있는 시스템이 없어, 마스크 미 착용자로 인한 위험성 방지에 취약점을 가지고 있다. 본 연구에서는 OpenCV 라이브러리를 이용한 마스크 착용 여부를 확인하는 시스템을 설계한다. Haar 특징기반 다단계 분류자를 이용하여 마스크 인식 프로세스를 설계하였으며, 마스크 착용 확인 시스템은 경량 컴퓨터인 라즈베리파이 장치 위에 구현하였다. 또한 확인된 사람의 이미지는 클라우드 시스템에 저장할 수 있도록 구현하였다. 본 연구를 통해, 누구나 손쉽게 해당 마스크 착용 확인 시스템을 중소 매장에 설치하여 사용할 수 있으며, 코로나바이러스 확산 방지에 기여할 수 있다고 예상한다.

1. 서론

2019년 말 발생한 코로나바이러스로 인하여 전 세계는 고통을 겪고 있으며, 이 전염성이 높은 바이러스를 퇴치하기 위한 큰 노력을 하고 있다. 그 노력 중 하나로, 모든 개인은 외부에서 마스크를 쓰는 것이 법으로 권고되었다[1]. 하지만, 일부 개인은 마스크를 착용하지 않은 채 공공시설을 이용하고 있으며, 이를 파악하여 위험을 선 예방하는 것에 어려움을 겪고 있다. 본 연구에서는 공공시설 및 중소 매장에서 손쉽게 설치하여 사용할 수 있는 마스크 확인 시스템을 제안한다. 논문의 2장에서는 마스크 확인 시스템을 설계하기 위한 관련 연구를 진행하며, 3장에서는 마스크 확인 시스템의 설계 및 구현에 관해 기술한다. 마지막 4장에서는 본 논문의 결과 및 의의를 도출한다.

2. 관련 연구

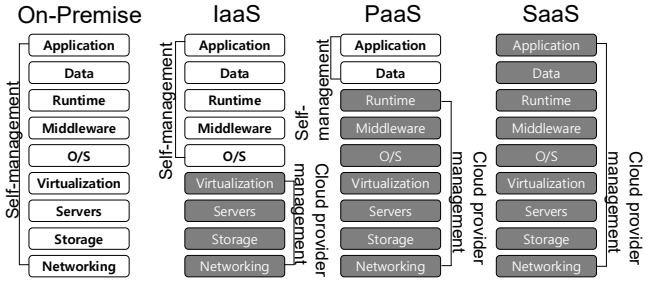
2.1 OpenCV

OpenCV는 실시간 비전 처리를 목적으로 인텔에서 최초로 개발하고 오픈소스로 발전한 프로그래밍 라이브러리이다[2]. 이 라이브러리는 실시간 이미지 처리에 많이 사용되고 있으며, 모바일 애플리케이션에서 사용되는 얼굴 인식, 문서 인식, 이미지 변환 기능 등에 해당 라이브러리가 광범위하게 사용되고 있다. OpenCV 라이브러리에는 머신러닝 기반의 이미지 추적이 가능한 Haar 특징기반 다단계 분류자가 있다. 이 방법은 2001년 제안된 기법으로 사람의 얼굴을 픽셀화한다. 그 후 흑백의 픽셀을 겹쳐 비교하여 해당 패턴을 비교하는 방식을 사용한다. 사람마다 모두 얼굴은 구체적으로는 다르지만, 기본적인 얼굴 생김의 패턴은 모두 비슷하므로 흑백의 명암은 비슷할 것이라는 가정하에 해당 분류자가 제안되어 사용되고 있다. 어떠한 패턴의 집합이 사람 얼굴 또는 특정한 오브젝트인지를 확인하기 특정 분류자는 샘플 이미지 집합을 가지고 Haar Training을 걸쳐 생성할 수 있으며, OpenCV는 사람 얼굴추적 등을 위한 기본 분류자를 이미 배포하고 있다.

2.2 클라우드 컴퓨팅

하드웨어 서버를 포함한 모든 장치를 직접 관리하

는 형태는 많은 인적 물적 관리를 요구한다. 최근 IT 환경은 클라우드 컴퓨팅 환경으로 이동하였으며, 하드웨어 및 소프트웨어를 빌려 쓰는 형태로 발전하였다[4]. 개인 사용자가 Microsoft Office 365와 같은 프로그램을 웹 브라우저상에서 사용하는 것도 소프트웨어를 빌려 쓰는 클라우드의 한 형태이며, Microsoft Azure, Amazon AWS와 같은 클라우드 서비스 제공자들은 기업에서 필요한 컴퓨팅 환경을 클라우드 환경에서 사용할 수 있게 한다. 그림 1과 같이 기존의 모든 자원을 직접 준비해야 하는 On-Premise 방식에서, 컴퓨터를 빌려 쓰는 IaaS 서비스, 데이터베이스와 같이 플랫폼을 빌려 쓰는 PaaS 서비스 및 모두 준비된 애플리케이션을 빌려 쓰는 SaaS 서비스와 같은 다양한 방식의 클라우드 컴퓨팅 방식이 있다[4]. 이렇게 필요한 부분 및 기간 동안만 자원을 빌려 사용하여 큰 비용 및 시간을 절감할 수 있다.



<그림 1> On-Premise 및 클라우드 서비스 비교

IoT 장비들을 이용한 경량의 장치들은 내부에 자체 데이터를 보관할 수 있는 용량이 부족하기 때문에 클라우드 컴퓨팅 환경과 연동하여 결과를 보관하는 것이 보편적이다.

3. 시스템 설계 및 구현

3.1 마스크 착용 구별 인식 프로세스 설계

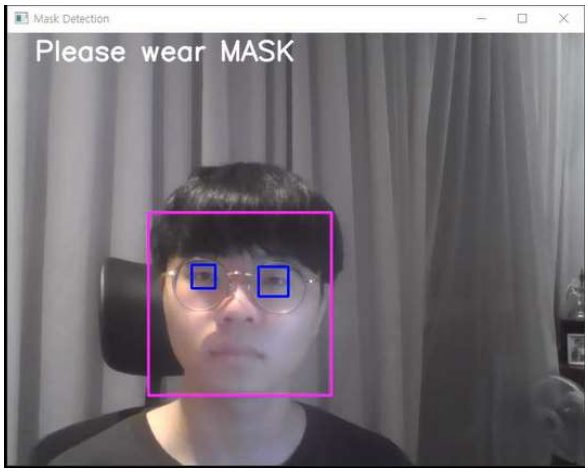
마스크 착용 구분 인식을 위해 OpenCV에서 제공하고 있는 Haar 분류자를 사용하였다. OpenCV에는 전체 얼굴 인식이 가능하게 하는 haarcascade_frontalface_default 분류자 및 전체 얼굴에서 눈만 추적할 수 있는 haarcascade_eye 분류자를 가지고 있다[5]. 이 두 분류자를 이용하여 아래 그림과 같은 마스크 착용 인식 프로세스를 설계한다.

전체 얼굴 인식 frontalface 분류자의 경우에는 얼굴의 일부가 가려져 있는 경우에는 얼굴로 인식하지 못하는 반면에, 눈을 인식하는 eye 분류자의 경우에는 사람 눈만 있는 경우에도 인식을 진행한다. 이에 아래 표 1과 같이 눈만 인식되고 얼굴 인식이 실패한 경우에는 마스크를 착용한 것으로 정의한다. 이를 통해 실험한 결과는 그림 2와 3과 같으며, 코드의 구현은 그림 4와 같이 하였다.

<표 1> 마스크 착용 인식 로직 정의

	얼굴 인식	눈 인식
마스크 미 착용	○	○
마스크 착용	×	○
사람 없음	×	×

<그림 2> 마스크 착용 인식



<그림 3> 마스크 미착용 인식

```
cap = cv2.VideoCapture(0)

face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier('haarcascade_eye.xml')
while 1:
    ret, img = cap.read()
    img = cv2.flip(img, 1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    (thresh, black_and_white) = cv2.threshold(gray, bw_threshold, 255, cv2.THRESH_BINARY)

    # detect face
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)
    eyes = eye_cascade.detectMultiScale(gray, 1.1, 4)
    if (len(faces) == 0 and len(eyes) == 0): #사람이 없음
        cv2.putText(img, "No People", org, font, font_scale, font_color, thickness, cv2.LINE_AA)
    elif (len(faces) == 0 and len(eyes) > 0): #얼굴 인식 실패, 눈 인식 성공
        cv2.putText(img, "Thank you for wearing MASK", org, font, font_scale, font_color, thickness, cv2.LINE_AA)
    elif (len(faces) > 0 and len(eyes) > 0): #얼굴 인식 성공, 눈 인식 성공
        cv2.putText(img, "Please wear MASK", org, font, font_scale, font_color, thickness, cv2.LINE_AA)
```

<그림 4> 마스크 착용 인식 Python 코드

3.2 데이터 저장 프로세스

마스크 착용 여부가 인식된 이후에 해당 결과는 데이터베이스에 저장되어 추후 확인이 가능하도록 한다. OpenCV를 이용한 마스크 인식 프로세스는 실시간으로 계속 이미지의 변화를 추적한다. 그렇기 때문에 일정 시간 또는 연속된 검색 조건이 만족하는 경우에 최종 인식 완료로 판단하고 저장 프로세스로 넘어갈 수 있도록 한다. 본 연구에서는 표 2와 같이 3초의 시간 구간에 확인된 결과를 모두 수집하여 평균을 내도록 하였다. 평균을 낸 결과가 얼굴 인식이 20% 이하 및 눈 인식이 80% 이상이면 마스크를 착용한 것으로 판단한다.

<표 2> 데이터 저장을 위한 최종 판단 기준

연속 3초간 결과		최종 판단
일굴 인식	눈 인식	
20% 이하	80% 이상	마스크 착용
80% 이상	80% 이상	마스크 미착용

특정한 사람이 3초의 연속된 시간이상 계속 카메라 앞에 머물러 있는 경우에, 중복 데이터가 저장될 수 있다. 이를 방지하기 위해서 3초의 시간 구간에 확인된 이후에, 2초 이상의 얼굴 및 눈 미 인식이 있어야 다른 사람이라고 가정하고 프로세스를 계속 진행한다. 이렇게 판단이 된 결과 이미지는 데이터베이스에 발견 시간과 함께 같이 저장되어, 추후 확인이 가능하도록 한다. 데이터베이스에 저장하기 위하여 검출된 이미지 바이너리 파일은 그림 5와 같이 Data URI 스키마 규약[6]을 따라 BASE64 인코딩을 통해 문자열 형태로 변환하여 데이터베이스에 저장할 수 있도록 한다.

```
data:[<mediatype>][;base64],<data>
(예) 
```

<그림 5> Data URI 스키마

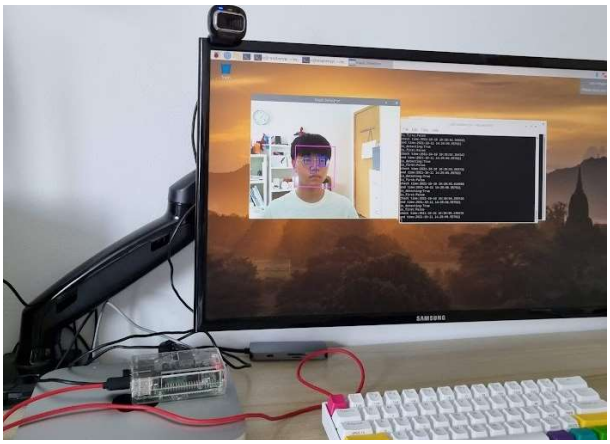
데이터 저장을 위한 데이터베이스 스키마 구조는 그림 6과 같이 저장 날짜, 인식 결과, 이미지를 저장할 수 있는 테이블을 가진다.

```
CREATE TABLE results (
  datetime TIMESTAMP NOT NULL,
  detect_result VARCHAR(20) NOT NULL,
  image TEXT
)
```

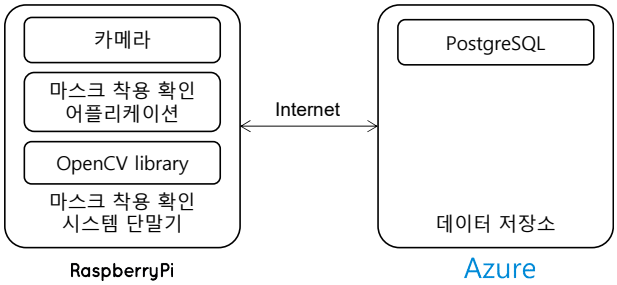
<그림 6> 결과 저장용 테이블 DDL

3.3 마스크 착용 확인 시스템의 구현

전체 시스템은 구성은 그림 6과 같이 각 공공 기관 및 중소 매장에 설치될 수 있는 단말기와 데이터를 저장하는 서버로 구성되어 있다. 본 연구에서는 초소형 컴퓨터 장비로 널리 사용되고 있는 라즈베리파이를 사용하여 단말기를 구성하였다. 그림 7과 같이 라즈베리파이 단말기 위에는 OpenCV 라이브러리가 탑재되어 있고, python 언어를 통해 구현된 마스크 착용 확인 애플리케이션을 구동하였다. 데이터를 저장하는 서버는 그림 8과 같이 Microsoft Azure에서 제공하는 PostgreSQL 서비스를 사용하였다. 이 경우 별도의 서버 자원을 구축해야 하는 번거로움이 없으며, 클라우드 환경 위에 바로 준비된 데이터베이스를 사용하여 저장 공간을 확보할 수 있다.



<그림 7> 라즈베리파이를 이용한 단말기



<그림 8> 시스템 구성도

3.4 마스크 착용 확인 시스템 실험

해당 마스크 착용 확인 시스템을 구동 시킨 후에 다수의 사람을 대상으로 테스트를 수행하였다. 카메라의 정면을 바라보는 정상 각도에서는 마스크 착용 및 미착용 상태를 모두 정상적으로 인식하고, 해당 결과는 그림 9와 같이 클라우드에 위치한 PostgreSQL에 정상적으로 저장된 것을 확인할 수 있었다. 하지만, 라즈베리파이의 한계로 인한 반응속도 저하 및 얼굴 인식 알고리즘의 제한으로 카메라를 정면으로 응시하지 않는 경우에는 인식이 되지 않는 경우도 확인하였다.

```
1  select datetime, detect_result, image from results ;
```

Data Output | Explain | Messages | Notifications

	datetime timestamp without time zone	detect_result character varying (20)	image text
1	2021-10-10 09:36:12.56938	mask	
2	2021-10-10 09:36:17.939785	mask	
3	2021-10-10 09:36:23.254555	mask	
4	2021-10-10 09:36:28.649162	mask	
5	2021-10-10 09:36:33.913533	mask	
6	2021-10-10 09:36:39.183491	mask	
7	2021-10-10 09:36:44.580123	mask	
8	2021-10-10 09:36:49.896107	mask	
9	2021-10-10 09:36:55.19658	mask	
10	2021-10-10 09:37:00.505483	mask	
11	2021-10-10 09:37:05.836539	no mask	
12	2021-10-10 09:37:11.12364	no mask	

<그림 9> 데이터 저장 결과

4. 결론

본 논문에서는 코로나바이러스 확산 예방을 위한 마스크 착용 확인 시스템을 설계 및 구현하였다. OpenCV의 얼굴인식 알고리즘을 확장하여 마스크 착용 및 미착용을 확인할 수 있도록 설계하였으며, 마스크 미착용을 한 사람에게는 경고문을 표시하여 경각심을 줄 수 있도록 하였다. 또한, 제안하는 시스템은 라즈베리파이를 사용하여 경량의 단말기 위에서 작동할 수 있도록 설계 및 구현하였다. 본 연구는 얼굴 인식 알고리즘 개선을 중점으로 하는 연구와 달리, 실제 마스크 착용 여부를 확인하는 현장에서 즉각 사용할 수 있는 IoT 단말기의 형태로 마스크 착용 시스템의 설계 및 구현을 진행한 것에 의의를 둘 수 있다. 이는 각종 중소 매장 및 큰 장비를 설치하기 협소한 각종 장소에 손쉽게 설치 및 활용이 될 수 있다고 생각한다.

현재 마스크 착용 확인 시스템은 마스크를 실제 착용하지 않고, 손으로 가리는 것과 같은 행위에서는 오작동을 할 수 있는 가능성이 있다. 향후 연구에서는 얼굴 하단의 입 부분을 가리고 있는 오브젝트를 확인하여 좀 더 정밀한 확인이 될 수 있도록 시스템을 개선하도록 한다.

참고문헌

[1] 보건복지부, “코로나바이러스 마스크 착용지침”, http://ncov.mohw.go.kr/guidelineList.do?brdId=7&brdGubun=71

[2] OpenCV, https://opencv.org/

[3] 구모세, 김상훈, “안드로이드 기반의 휴대용 스마트폰을 이용한 실시간 얼굴 검출”, 한국정보처리학회 학술대회논문집, 27(2), 1077-1079, 2020.

[4] 유영문, “중앙행정기관의 클라우드 시스템 도입 현황”, 한국기록관리학회지, 19(3), 247-270, 2019.

[5] Mehtab, “S. Face Detection Using OpenCV and Haar Cascades Classifiers”

[6] Mozilla, “Data URIs”, https://developer.mozilla.org/ko/docs/Web/HTTP/Basics_of_HTTP/Data_URIs