중앙대학교 예술공학대학
College of Art & Technology
Chung-Ang University

Java OpenCV 기반 Mask Detection과 Mask Video Filter 기능 구현

Based on Java OpenCV with Processing 3

중 앙 대 학 교　예 술 공 학 대 학
컴퓨터예술학부 20191184 김범수

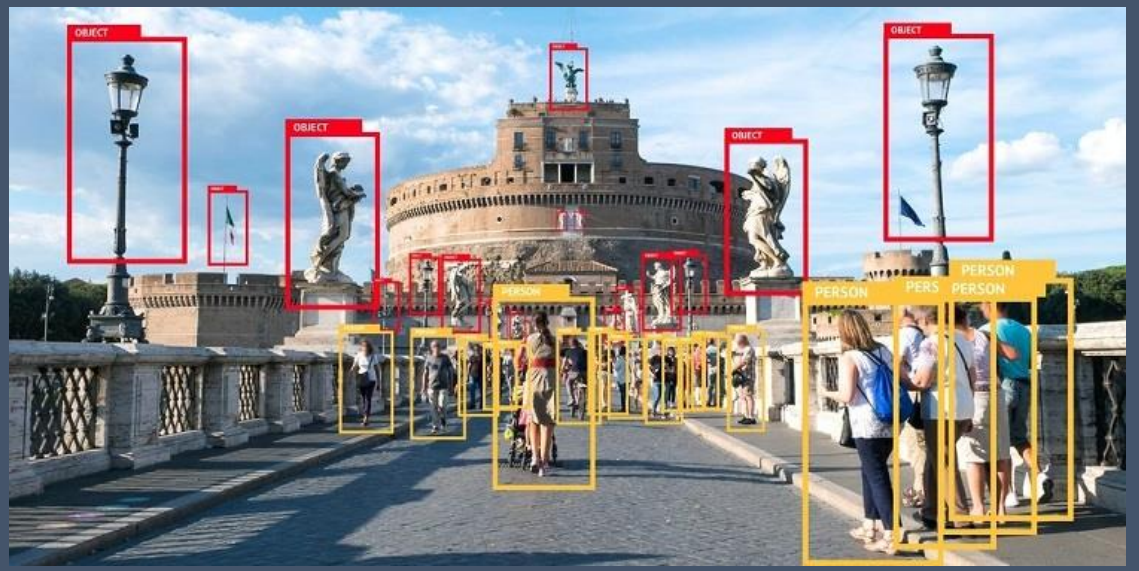
목차

1. Object Detection : 객체 탐지 및 인식

- 컴퓨터 비전과 영상처리 분야의 기술
- 객체 탐지(Object Detection), 객체 인식(Object Recognition)은 엄연히 다른 의미
  - 객체 탐지 : 오브젝트가 존재하는 지 판단
  - 객체 인식 : 오브젝트가 어떤 것인지 판단 (즉, Classification을 포함)
  - But! 혼용되어 사용, 혹은 포괄적인 범주로 모두 Object Detection 이라고도 한다.

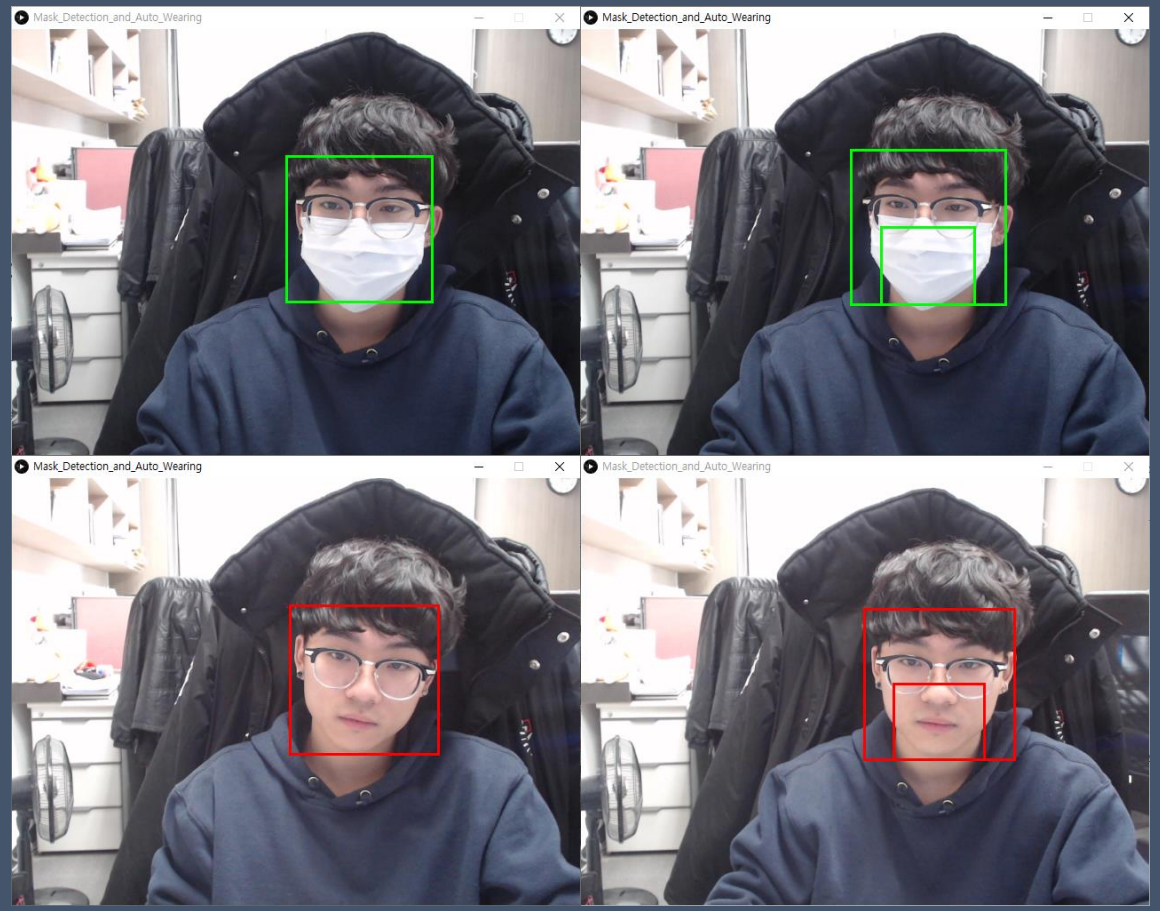
- 객체 탐지 및 인식을 사용하기 위해 OpenCV 라이브러리를 사용 (OpenCV for Processing)

- OpenCV 란?
  - 실시간 컴퓨터 비전을 목표로 개발된 라이브러리
  - 2D 기반 프로세싱에 중점
  - ↔ 3D 기반 프로세싱 라이브러리 : PCL, Open3D



-Object Detection 예시-

2. 마스크 착용 감지 기능



<마스크 착용>

- 마스크 착용 시 바운딩 박스가 초록색으로 출력되며, 이는 안전하다는 의미를 시각적으로 표현한다.
- openCV에서 넘겨주는 파라미터(타입)을 받아 마스크 착용 검사 함수를 정의하였다.

<마스크 미착용>

- 마스크 미착용 시 바운딩 박스가 빨간색으로 출력되며, 위험하다는 의미를 시각적으로 표현한다.
- 화면 하단에 버튼이 생기며, 토글 버튼을 클릭 시 마스크 비디오 필터 기능이 작동되어 마스크를 강제로 착용하게 한다.

3. 마스크 필터 기능

version0.1

version0.2

<마스크 필터 기능>

- 인식된 얼굴에서 하단 부분에 마스크를 실시간으로 입혀준다.

<미스크 필터 파라미터>

- openCV에서 넘겨준 파라미터를 받아와 특정 부분에 2차원으로 이미지를 입히는 방식으로 구현하였다.

4. 핵심 구현 코드 : 프로세싱

Mask_Detection_and_Video_Filter.pde

Mask_Detection_and_Auto_Wearing | cv_mode | isMaskDetection

```
19  opencv.loadCascade(OpenCV.CASCADE_FRONTALFACE);
20  cam.start();
21 }
22
23 void draw() {
24   opencv.loadImage(cam);
25   image(cam, 0, 0);
26
27   cv_mode();
28 }
```

매 프레임마다 cv_mode 함수를 실행

isMaskDetection.pde

Mask_Detection_and_Auto_Wearing | cv_mode | isMaskDetection

```
4  */
5
6  Boolean isMaskDetection(Rectangle[] r) {
7    int num, ans;
8    color c;
9    Boolean is = false;
10
11   for (int i=0; i<r.length; i++) {
12     ans=0; num=0;
13     for (int x=0; x<r[i].width*0.6; x++) {
14       for (int y=0; y<r[i].height*0.5; y++) {
15         c = cam.get(int(r[i].x+r[i].width*0.2+x), int(r[i].y+r[i].height*0.5+y));
16         ans += red(c) + green(c) + blue(c);
17         num+=3;
18       }
19     } is = (ans/num>200) ? true : false;
20   } return (is==true) ? true : false;
21 }
```

1번 기능의 경우 마스크를 착용했는지 에 따라서 값을 다르게 리턴

마스크 착용 여부는 흰색 픽셀 값에 따라서 결정

cv_mode.pde

Mask_Detection_and_Auto_Wearing | cv_mode | isMaskDetection

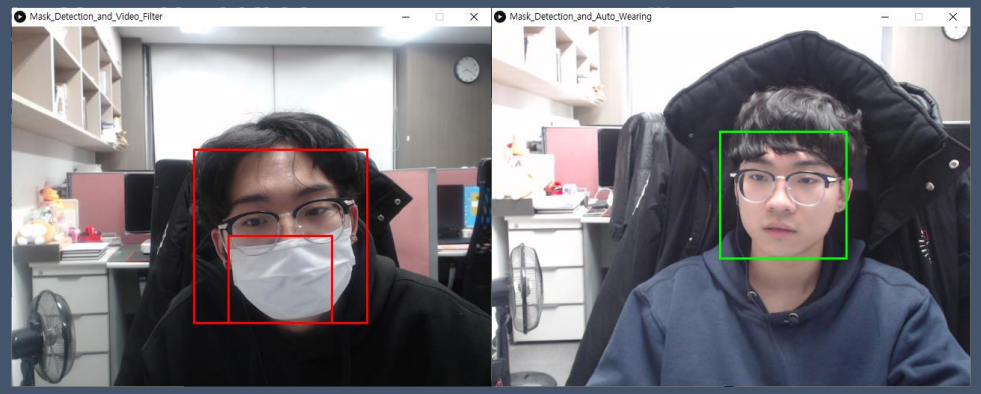
```
19   case '2':
20     for (int i=0; i<faces.length; i++) {
21       //image(img, faces[i].x+0.1*faces[i].width+5, faces[i].y+0.5*faces[i].height-25,
22       //  faces[i].width*0.8, faces[i].height*0.5);
23       image(img, faces[i].x+0.1*faces[i].width, faces[i].y+0.5*faces[i].height,
24         faces[i].width*0.8, faces[i].height*0.5);
25     }
26     break;
27
28   // Mode 1 : Mask Detection(Default)
29   case '1':
30   default :
31     noFill();
32     strokeWeight(3);
33
34     if (!isMaskDetection(faces)) {
35       stroke(255, 0, 0); //Not Safety
36     } else {
37       stroke(0, 255, 0); //Safety
38     }
39
40     for (int i=0; i<faces.length; i++) {
41       rect(faces[i].x, faces[i].y, faces[i].width, faces[i].height);
42       rect(faces[i].x+0.2*faces[i].width, faces[i].y+0.5*faces[i].height,
43         0.6*faces[i].width, 0.5*faces[i].height);
44     } break;
45   }
46 }
```

입력받은 키에 따라 각자 다른 기능을 수행

5. 오작동 원인과 극복 방안

- 원인 1 : 특정 조명 위치에서 반사가 심하게 일어나 피부가 일시적으로 급격히 밝아지는 현상
- 원인 2 : 특정 조명 위치에서 그림자가 짙게 일어나 피부가 일시적으로 급격히 어두워지는 현상

- 극복 : 오작동을 일으키는 현상은 짧은 시간 내에 일어나므로, 특정 기간 프레임에서 대부분을 차지하는 값을 기준으로 작동하도록 설계
  - 예) framRate를 초당 36으로 설정하고, 2초마다 Detection을 하도록 설계한다. 총 72프레임 중에 54개 이상의 프레임이 리턴하는 값으로 2초마다 최종 리턴한다. 즉, 마스크를 감지하여 다른 색을 출력하는 바운딩 박스는 72프레임 중에 지배적인 값(True or False)을 기준으로 2초마다 색을 정하여 갱신된다. 따라서 일시적으로 일어나는 오류 값들은 지배적이지 않으므로 필터링된다.



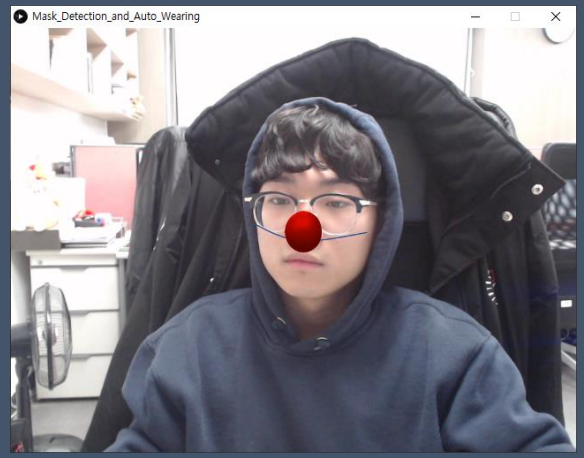
-오작동 예-

6. 기대효과 및 향후 활용 방향

1. 웹캠을 사용하여 점포 입구에 간단하게 설치하면, 마스크 착용 여부를 자동으로 확인할 수 있다

2. 비대면 수업으로 사용률이 급상승한 줌(zoom) 플랫폼의 필터 기능과 동일하게 사용할 수 있다.

-줌 플랫폼의 비디오 필터 기능 예시-

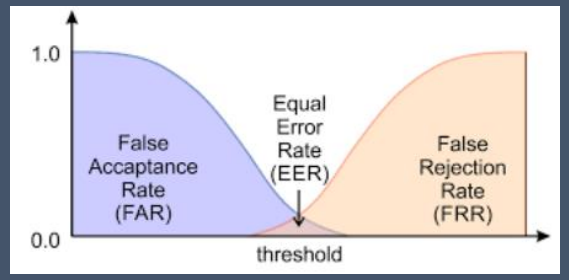


-예시 필터 1 : 루돌프 사슴 코 착용-

-예시 필터 2 : 떠그 라이프 안경 착용-

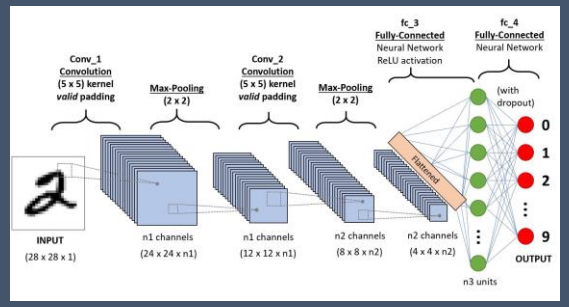
7. 개선 방향1

- 개선점 : 해당 기능은 사람 얼굴을 잘 감지하지만, 다른 사물에 대해서도 너무 쉽게 사람이라고 인식한다.
- 개선 방안 1 : EER 그래프 활용



바이오 인증 성능 지표인 EER 그래프는 인가된 사용자를 인가받지 못하는 오거부율(FRR)과 인가되지 않은 사용자가 인가되는 오인식율(FAR)의 비율이 같을 때 효과적이라고 한다.
-> 사람을 인식하지 못하는 비율과 사물을 사람이라고 인식하는 비율을 연구하여 효과적인 threshold를 찾아, 특정 경우에만 Detection을 이루어지게 하여 오류를 개선한다.

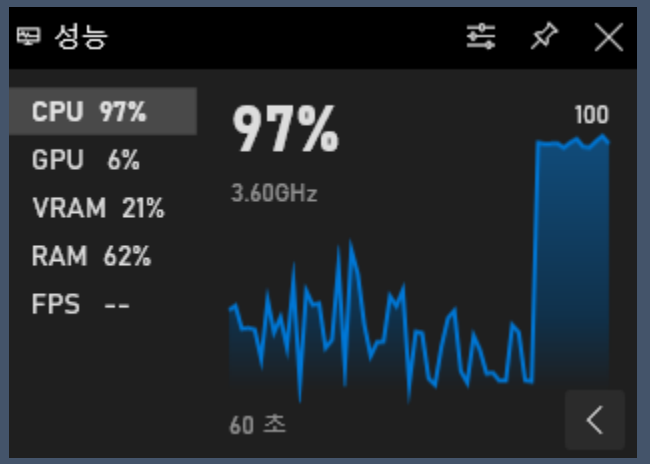
- 개선 방안 2 : 심층 신경망 활용(CNN, SSD, Yolo)



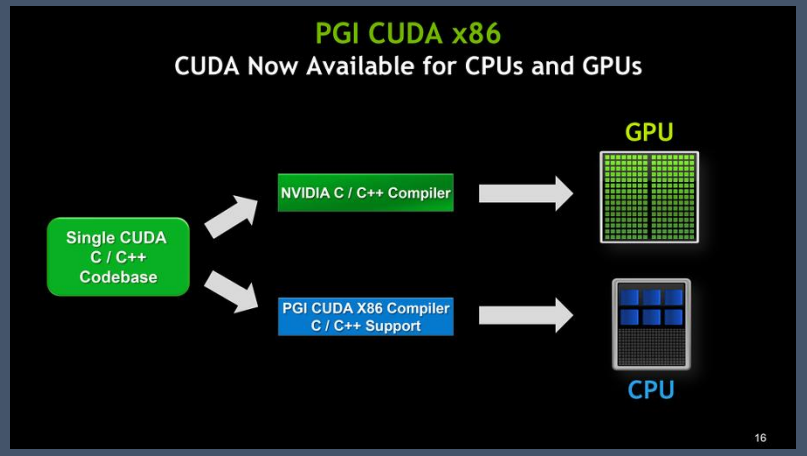
심층 신경망(DNN)은 인공지능 분야 중 하나인 딥러닝 기술의 아키텍처이다. 딥러닝 분야 중 이미지 기반의 모델인 CNN, SSD, Yolo 모델은 사람과 유사한 뉴런 구조를 통해서 이미지를 학습하기 때문에 높은 정확도를 보인다. 이를 사용하면 조금 더 정확하게 사람을 인식할 뿐만 아니라 직접 학습하여 다른 객체에 대해서도 검출할 수 있다.

7. 개선 방향2

- 개선점 : CPU 연산 처리 부담으로 인한 성능 다운 이슈.
- 개선 방안 : NVIDIA CUDA를 이용한 병렬처리 – 그래픽카드를 사용하자



-어플을 실행하자 CPU 사용률이 급격하게 뛰는 모습-



그래픽카드(GPU)는 특정 연산에 대해서 CPU 보다 더 높은 성능을 가지고 있다. 반복적인 병렬 계산에 필요한 기능을 제공하주는 소프트웨어인 CUDA는 CPU에서 명령한 연산 처리를 병렬로 수행하여 결과물을 메모리에 저장해 두었다가 메인 메모리로 업로드하고, 이를 CPU가 나머지 작업을 수행하는 방식의 처리 과정을 진행한다. 따라서 CPU에 부담되는 연산들은 NVIDIA CUDA를 이용하여 연산하는 방식으로 진행하면 CPU 연산 부담을 줄일 수 있다.

> 실제 딥러닝 분야에서는 CUDA와 cuDNN(딥러닝 연산 소프트웨어) 을 통하여 연산을 수행한다. 딥러닝 프레임워크 tensorflow, pytorch 모두 해당 소프트웨어를 기반으로 작동한다.

8. 참고 자료 및 버전

- 참고 서적
  - 프로세싱 교과서, 케이시 리아스, 벤 프라이, 유엑스리뷰(UX REVIEW), (2018)

- 소스 코드
  - 깃허브 리포 : https://github.com/gh-BumsooKim/3D-Entertainment-Design

- 사용 버전
  - Processing 3.5.4
  - OpenCV for Processing 0.5.4
  - Java OpenCV 2.4.5.0

- 어플리케이션
  - 해당 파일과 같은 폴더에 exe 파일 별도로 추출 (더블 클릭하여 바로 재생, 웹캠 연결 필요)

9. 매뉴얼

사용법 – 키보드 인터랙션
환경세팅 – 웹캠 연결 (최소 사양 640 X 480)

- Key==‘1’ – 마스크 착용 검사
- Key==‘2’ – 마스크 필터 기능
- Key==‘3’ – 루돌프 사슴 코
- Key==‘4’ – 떠그 라이프 안경