Contents

1. 과제 개요

1) 제안 멀티미디어 기술 소개 (기술 배경, 필요성 등)

이번 과제의 주제를 이미지 인식 기술에 대한 내용으로 정했다.

우리는 요즘 실생활에서 스마트폰만 하더라도 얼굴인식 기능으로 가깝고 쉽게 이미지 인식 기술을 접할 수 있다. 하지만 이미지 처리의 복잡성과 다양성은 음성보다 크기 때문에 아직 세계적으로 이미지 인식 기술은 음성인식에 비해 많이 부족한 상황이다.

하지만 이미지 인식 기술은 다양한 곳에서 아주 유용하게 사용되고 있고 좀 더 발전한다면 더 다양한 곳에서 사용할 수 있다. 예를 들어 전자제품 공장에서 사용하게 된다면 부품 불량의 원인인 기포와 먼지를 구별할 수 있게 되고 보안 검색대에서는 보다 정확하게 물건들을 확인할 수 있으며 무인스토어에서는 구매자가 고른 제품들을 계산대에 올리면 제품들의 종류를 자동으로 인식하게 계산할 수 있게 되는 등 우리의 삶의 질을 향상시켜 주는데 아주 많은 도움이 될 수 있는 기술이라고 생각한다.

따라서 나는 아직은 부족한 이미지 기술의 발전과 이 기술이 발전된다면 우리에게 미칠 잠재성에 대해 매우 흥미롭게 느꼈기 때문에 이 주제를 선택하게 되었다.

또 아무래도 이미지 분석 및 인식 기술에서 가장 대표적인 기술 중 하나인 얼굴인식 기술에 대해 좀 더 자세하게 살펴볼 예정이다.

2) 기존 기술의 현황, 문제점 및 개선 방안, 기대효과

영국 BBC에 따르면 10월 24일(현지시간) 미국에서 출시된 구글 ‘pixel4’의 경우 사용자가 눈을 감고 있는 데도 얼굴이 인식돼 잠금이 풀렸다. 전작에 있던 지문인식 센서를 없애고 얼굴인식 기능을 추가했다가 사달이 난 것이다.
같은 달 31일 JTBC는 국내에서 애플 ‘아이폰X’ 사용자 김모씨의 초등학생 아들이 아버지의 페이스ID를 해제해 게임 아이템 1000만원어치를 결제했다고 보도했다. 아이폰X부터 적용된 페이스ID는 사용자 얼굴을 암호화한 보안 시스템이지만 사용자를 닮은 아들 얼굴에 허점을 노출했다. 애플은 자사 홈페이지에서 “얼굴인식이 잘못될 확률은 100만분의 1(0.0001%)”이라면서도 “사용자와 닮은 얼굴엔 통계적으로 달라질 수 있어 우려되면 비밀번호 설정을 권장한다”고 대응했다.

[출처] : https://news.joins.com/article/23651357

위의 기사의 내용과 같이 아직 완벽하지 않은 안면인식 기술로 인한 다양한 문제들이 있다. 또한 현재 다양한 전자기기에서 사용되고 있는 안면인식 기술은 딥러닝을 활용한 기술이기 때문에 딥러닝에 필요한 시간까지는 정확성이 떨어진다는 단점 또한 존재한다.

AI 이미지 분석은 방대한 양의 이미지를 학습해 이미지를 정확하게 인식하는 기술이다. 핵심은 이 과정에서 '노이즈'를 얼마나 판별해 실제 이미지를 구분하느냐는 것이다. 예를 들어 푸들이 앞 다리를 내밀고 찍은 사진 모양만 인식해 비슷하게 생긴 치킨을 푸들로 인식하거나 눈, 코, 입 까만색과 위치만을 인식해 건포도가 비슷한 위치에 박힌 머핀을 치와와로 인식하는 오류를 범하기 쉽다.

[출처] https://m.etnews.com/20200512000207

위의 기사처럼 이미지 분석 시 방대한 양의 데이터가 필요할뿐더러 그 데이터들의 quality 또한 분석의 정확성에 큰 영향을 미치며 이 말은 즉 quality가 낮은 이미지는 정확하지 분석하지 못한다는 말이 된다.

2. 과제 연구 내용

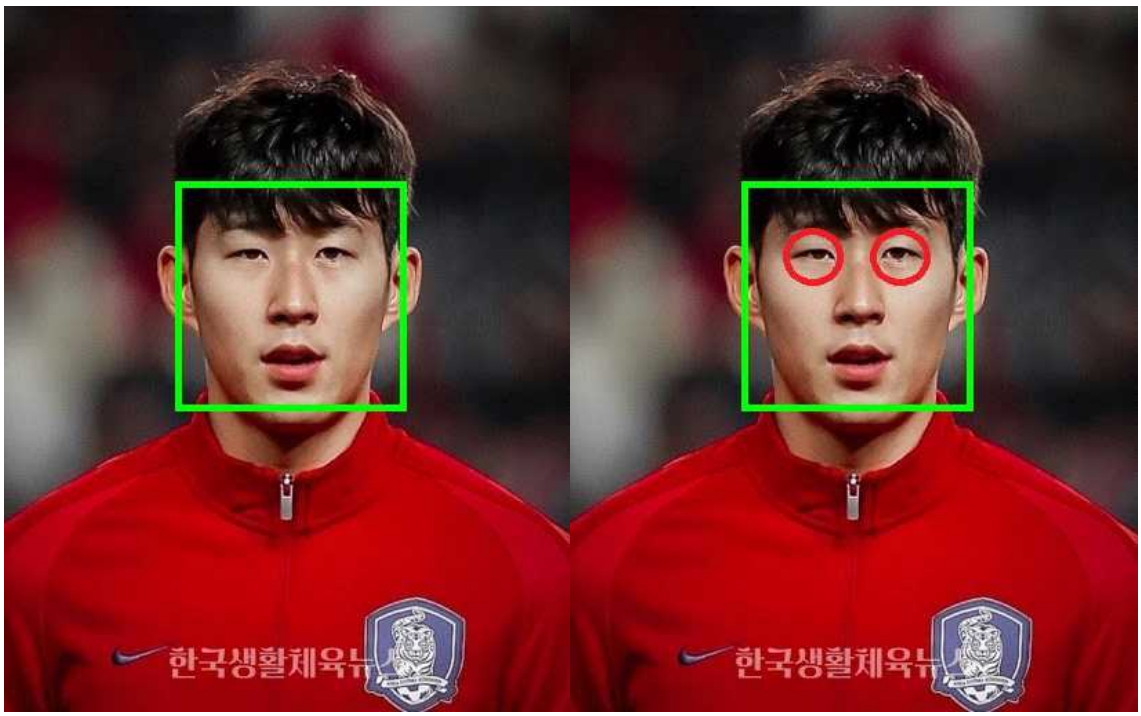
1) 연구 멀티미디어 기술 소개

C++ Code

```
#include <iostream>
#include <cv.h>
#include <highgui.h>

using namespace std;
using namespace cv;

String casc = "C:/user/haarcascade_frontalface_alt.xml";
String eye_casc = "C:/user/haarcascade_eye.xml";
String img_name = "son.jpg";

CascadeClassifier face;
CascadeClassifier eye;

int main()
{
	Mat img = imread(img_name);

	if (img.data == NULL) {
		cout << img_name << "image open failed" << endl;
		return -1;
	}

	if (!face.load(face_cascade) || !eye.load(eye_cascade)) {
		cout << "Cascade file open failed" << endl;
		return -1;
	}

	#pragma region
	Mat gray;
	cvtColor(img, gray, CV_RGB2GRAY);

	vector<Rect> face_pos;
	face.detectMultiScale(gray, face_pos, 1.1, 3, 0 | CV_HAAR_SCALE_IMAGE,
				Size(15, 15));

	for (int i = 0;i < (int)face_pos.size();i++) {
		rectangle(img, face_pos[i], Scalar(0, 255, 0), 2);
	}
	#pragma endregion
```

```
        #pragma region
        for (int i = 0;i < (int)face_pos.size();i++) {
                vector<Rect> eye_pos;

                Mat roi = gray(face_pos[i]);
                eye.detectMultiScale(roi, eye_pos, 1.1, 3, 0 |
                                        CV_HAAR_SCALE_IMAGE, Size(10, 10));

                for (int j = 0;j < (int)eye_pos.size();j++) {
                        Point center(face_pos[i].x + eye_pos[j].x +
                                (eye_pos[j].width / 2),
                                face_pos[i].y + eye_pos[j].y + (eye_pos[j].height / 2));

                        int radius = cvRound((eye_pos[j].width +
                                        eye_pos[j].height) * 0.2);
                        circle(img, center, radius, Scalar(0, 0, 255), 2);
                }
        }
        #pragma endregion

        namedWindow("Detection");
        imshow("Detection", img);
        waitKey();
        return 0;
}
```

[출처] : https://vision0814.tistory.com/113?category=569547

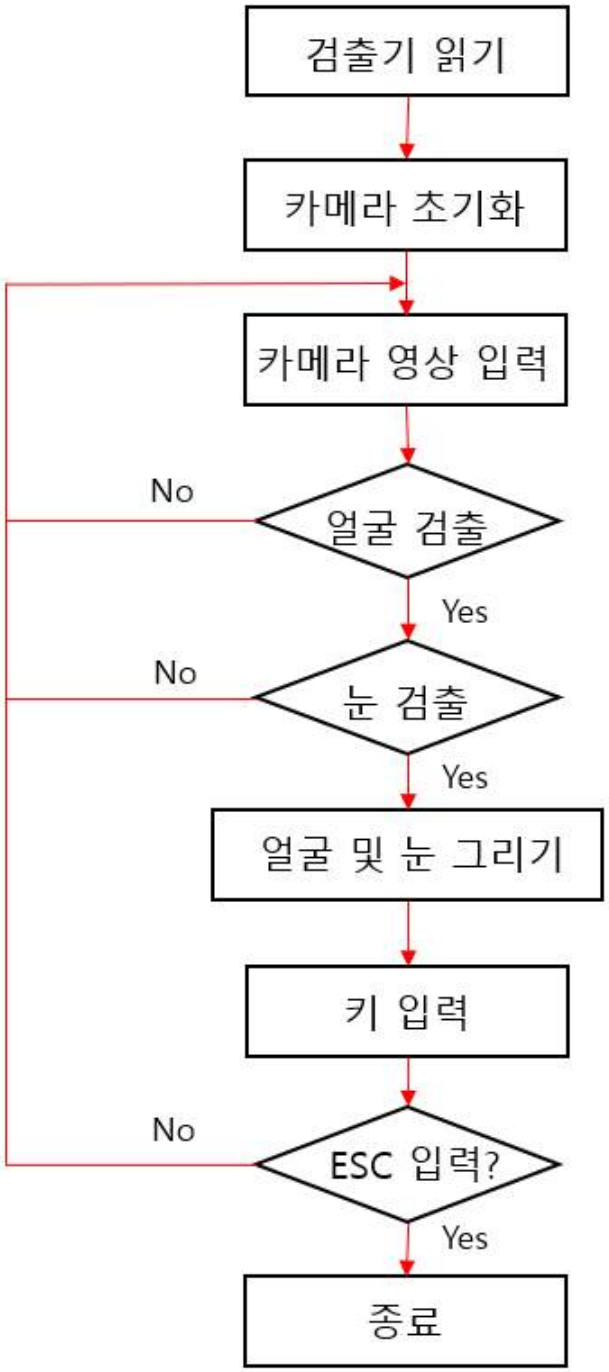


<얼굴>

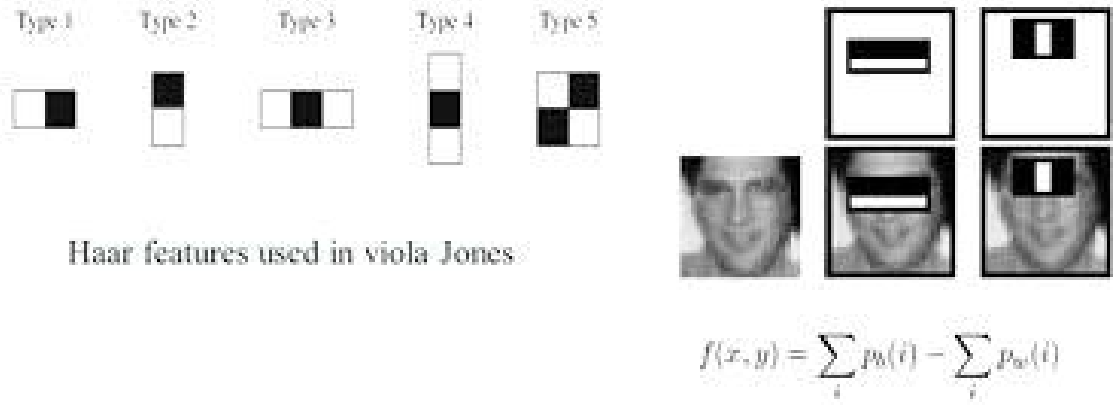
<얼굴&눈>

```
<opencv_storage>
  - <haarcascade_frontalface_alt type_id="opencv-haar-classifier">
      <size>20 20</size>
    - <stages>
      - <_>
            <!-- stage 0 -->
        - <trees>
          - <_>
                <!-- tree 0 -->
            - <_>
                  <!-- root node -->
              - <feature>
                - <rects>
                    <_>3 7 14 4 -1.</_>
                    <_>3 9 14 2 2.</_>
                  </rects>
                  <tilted>0</tilted>
                </feature>
                <threshold>4.0141958743333817e-003</threshold>
                <left_val>0.0337941907346249</left_val>
                <right_val>0.8378106951713562</right_val>
            </_>
          </_>
          - <_>
                <!-- tree 1 -->
            - <_>
                  <!-- root node -->
              - <feature>
                - <rects>
                    <_>1 2 18 4 -1.</_>
                    <_>7 2 6 4 3.</_>
                  </rects>
                  <tilted>0</tilted>
                </feature>
                <threshold>0.0151513395830989</threshold>
                <left_val>0.1514132022857666</left_val>
                <right_val>0.7488812208175659</right_val>
            </_>
          </_>
          - <_>
```

위에 있는 그림은 haarcasade_frontalface.xml 파일의 내용 중 일부이다. <size>20 20은 얼굴 크기고 그 안의 영역을 사각형이 돌아다니면서 눈과 코의 명암의 평균 차를 측정하는 것이다. rects값으로 나타난 값들은 각각 3(x값), 7(y값), 14(폭(width)), 4(높이(height)), -1(이득(gain))를 나타낸다. 이득으로 표시된 마지막 값은 부호 또한 나타낸다. -1이면 pixel 값의 평균을 빼주라는 뜻이고, 2는 2를 곱한 다음 더하란 뜻이 된다. <threshold>는 임계값을 나타낸다.



결과에서 얼굴과 눈이 검출 됨을 알 수 있다. cascade파일과 이미지를 불러오고 검출 함수로 출력하는 구조이다.
함수는 파일을 불러온 뒤 동작한다는 사실을 파악할 수 있다. 파일에 따라 얼굴만 검출되기도, 얼굴과 눈이 검출되기도 했다.
불러온 파일들을 보면 haarcascade란 단어가 있는데 이것에 대해 알아보겠다.



[출저] : http://csjournals.com/IJCSC/PDF7-1/11.%20Sonia.pdf

<그림 1>

그림을 보면 사람의 얼굴에는 특별한 패턴이 있다는 것을 알 수 있다. 코는 명암이 밝고 두 눈은 명암이 어둡다. 이러한 명암차이를 이용해 패턴을 구하는 것이다. 이러한 방식이 Haar like feature 이다. 얼굴 위에 흑백의 사각형을 겹치고 밝은 영역에 속한 pixel값들의 평균에서 어두운 영역에 속한 pixel값들의 평균의 차이를 구한다. 그 차이가 threshold를 넘으면 얼굴의 haar like feature가 존재한다고 한다. 얼굴은 다양하지만 생김새의 패턴은 비슷하기 때문에 임의의 얼굴이라도 코와 눈 사이의 명암의 차이가 비슷한 점을 이용한다.

Haar Cascade 알고리즘의 원리

Haar Cascade는 머신러닝 기반의 오브젝트 검출 알고리즘으로 이미지에서 오브젝트를 검출하기 위해 사용된다.
직사각형 영역으로 구성되는 특징을 사용하기기 때문에 pixel을 직접 사용할 때에 비해 동작 속도가 빠르다.
찾고 싶은 객체(얼굴)가 포함된 이미지와 객체가 없는 이미지를 활용하여 Haar Cascade Classifier(haar 특징 분류기)를 학습시킨다.
그리고 난 뒤 분류기를 사용하여 객체를 검출한다.

알고리즘은 4단계로 구성된다.
1) Haar Feature Selection
2) Creating Integral Images
3) Adaboost Training
4) Cascading Classifiers

1) Haar Feature Selection
첫 단계는 이미지에서 Haar Features를 계산하는 것이다.
가능한 모든 크기의 Kernel을 가지고 이미지 전체를 스캔하여 haar 특징을 계산한다.
예로 (24 x 24) 크기의 window를 사용할 때는 160000개가 넘는 haar 특징을 구하게 된다.
haar 특징은 이미지를 스캔하고 위치를 이동하면서 인접한 직사각형들의 영역 안에 있는 pixel 간의 합의 차이를 이용하는 것이다.
사각 영역 내부의 pixel들을 빠르게 더하기 위해서 적분 이미지를 활용 및 사용하여 더해준다.
haar 특징에는 다음 세 가지가 있다.

① haar 특징의 값은 두 사각형 영역 안에 있는 pixel들을 합을 구하여 검은색 영역의 합(하얀색 영역의 합)한 결과로 나온 값을 활용하여 구한 다. 두 사각형의 크기와 모양은 동일하다.
② 세 개의 사각형으로 구성된 haar 특징의 값은 중앙에 위치한 검은색 사각 영역 내부 pixel값들의 합(바깥에 위치한 두 개의 하얀색 사각 영역 내부 pixel값들의 합)한 결과로 나온 값을 활용하여 구한다.
③ 네 개의 사각형으로 구성된 haar 특징의 값은 대각선에 위치한 영역들 간의 차이를 활용하여 구한다.

2) Integral Images(적분 이미지)
haar 특징을 계산하기 위해선 검은색 사각형, 하얀색 사각형 아래에 있는 pixel들의 합 또한 구해야 한다.
pixel의 합을 구하는 과정의 속도를 빠르게 만들기 위하여 적분 이미지를 활용한다.
적분 이미지를 활용하면 큰 이미지라도 지정한 영역에 대한 pixel들의 합을 구하는 과정을 빠르게 해결할 수 있다.
적분 이미지는 다음처럼 생성한다.
기존 이미지의 높이와 너비에 대해 각각 1씩 더해주고 크기가 더 큰 이미지를 만들고 왼쪽과 위쪽은 0으로 채워준다.
그리고 기존 이미지에서 영역을 지정하여 영역 내부의 pixel값들의 합을 구하고 적분 이미지에서 기존 이미지에서 지정한 영역 오른쪽 하단에 있는 pixel에 대응하는 위치에 대한 합을 입력해준다.
기존 이미지에서 영역을 지정하여 영역 내부의 값을 구해줄 때는 적분 이미지에 대응되는 영역 네 곳의 pixel값들을 사용하여 영역에 대한 합을 계산할 수 있다.
적분 이미지에 대응되는 영역의 오른쪽 하단에 있는 pixel의 값에서 위쪽 pixel값과 하단 pixel값을 빼고 대각선 방향의 pixel값을 더해준다.

3) Adaboost Training
먼저 이미지의 앞에서 선택한 haar 특징을 활용하여 특징을 계산한다.
160000개 이상의 특징이 검출되는데 이 중에서 얼굴 검출을 위해 도움이 될 만한 의미 있는 특징을 골라야 한다.
이 때 처리를 위한 성능 향상을 위해 Adaboost를 사용한다.
앞에서 구한 특징들 중 대부분의 특징들은 의미가 없다.
얼굴에서 가로 방향의 검은색과 하얀색 사각영역이 있는 특징의 경우엔 눈 부분이 코와 빰 보다 더 어둡다는 특성을 사용한다.
세로 방향의 하얀색 사각 영역이 있고 좌우로 검은색 사각 영역이 존재하는 특징의 경우에는 양쪽에 있는 눈 부분이 중앙에 있는 코보다 더 어둡다는 특성을 사용한다.
Adaboost는 다음 과정을 통해 이루어진다.
최적의 특징을 찾기 위해서 모든 학습 이미지의 특징을 적용하여 찾는다.
각각의 특징에 있어서 얼굴이 포함된 이미지와 얼굴이 포함되지 않은 이미지를 분류하기 위해 최적의 임계값(threshold)을 찾는다.
(하얀색 영역과 검은색 영역의 차가 일정 임계값 이상일 때 얼굴에 대한 특징이라 보는 것이다. 얼굴을 제외한 영역에서는 두 영역에 대한 차가 거의 없다고 보고 만든 알고리즘이기 때문이다.)
이 때 분류가 잘못될 가능성이 존재하기 때문에 error rate가 낮은 특징을 선택해야 한다.
즉 얼굴이 포함된 이미지인지 얼굴이 포함되지 않은 이미지인지를 보다 정확하게 분류 가능한 특징을 선택해야 한다는 것이다.

4) Cascade Classifier
다음으로 이미지를 (24 x 24) 크기의 window를 사용하여 16000개의 haar 특징을 적용하여 얼굴을 검출한다.
하지만 이렇게 모두 계산한다면 양이 너무 많아서 비효율적이다.
이미지에서 얼굴이 없는 영역들이 많은 영역을 차지한다. 그래서 현재 window가 가르키는 영역이 얼굴 영역인지를 단계별로 확인하는 방법을 사용한다.
얼굴 영역인지 아닌지 하위 단계에서는 짧은 시간 안에 판단하고 상위 단계로 갈수록 더 시간이 오래 걸리는 연산을 수행한다.
이러한 방식을 Cascade Classifier라고 한다.
window가 이미지 위를 이동할 때 각각 16000개의 특징을 모두 적용할 수 없기에 여러 단계의 그룹으로 분류하여 사용한다.
처음 단계의 특징에서 얼굴 영역이 아니라고 판정이 되면 다음 위치로 window를 이동하며 얼굴 영역이라고 판정이 되면 현재 window가 위치한 곳에서 다음 단계의 특징을 적용한다.

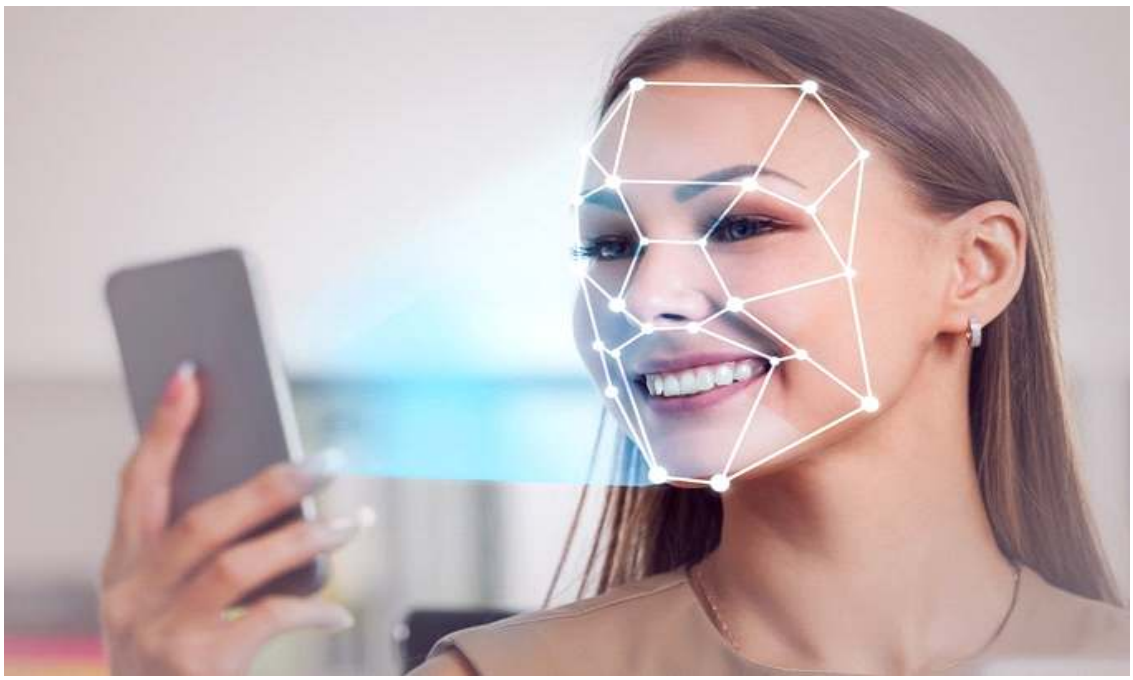
[출처] : https://webnautes.tistory.com/1352

2) 응용 사례

이미지 분석 기술의 응용 사례는 위에서 언급했듯이 대표적으로 무인 편의점에서 물건을 스캔하는 것, 보안 검색대에서 사람의 눈으로 물건을 하나하나 식별함으로써 떨어지는 정확성을 보완하여 물건을 분류하는 것, 또 지금 전 세계 사람들이 흔하게 사용하고 있는 스마트폰의 얼굴(안면)인식 등을 응용 사례라고 할 수 있을 것이다. 스마트폰에서 지문 인식과 얼굴 인식이 등장함으로 인해 복잡했던 은행 업무 또는 다른 여러 가지 강한 보안이 필요한 업무들이 간단하게 생체 인증 기반 보안 방식으로 해결되면서 우리의 일생은 한층 편해졌다. 물론 이러한 기술들이 보안적인 측면에서 약한 것도 사실이고 보통 사람들의 인식 또한 공인인증서 등의 기술보다 보안이 떨어진다고 생각하는 것도 사실이지만 어느 기술이나 100%의 보안은 없다고 생각한다. 따라서 오히려 좀 더 편리한 보안이 더 좋을 것이라고 생각한다.

- 생체 인증 -

안전하게 만든 비밀번호는 안전한 형식의 인증이 되지만 실제로 이러한 종류의 비밀번호는 만들기 번거롭고 기억하기도 어렵다. 그래서 흔히들 사람들은 “1234”와 같이 보안이 낮은 비밀번호를 설정하기도 한다. 또 자신들의 비밀번호를 보안이 낮은 곳(수첩, 휴대폰의 메모장 등)에 보관하는데 이러한 행위는 비밀번호를 설정하는 목적 자체를 의미없게 만드는 행동이다. 하지만 생체 인증 보안은 우리에게 고유한 인식 형식을 제공하므로 이러한 생체 비밀번호는 번거롭지도 보안이 낮은 곳에 저장할 일도 없게 해준다. 생체 인증은 우리가 어려운 비밀번호를 만들거나 보안이 낮게 보관할 일이 없게 하고 대신 신체의 고유한 요소를 스캔한다. 따라서 우리에게 편리함을 제공해주고 삶의 질을 향상시켜주는 이러한 이미지 분석 기술의 발전은 아주 중요하다고 생각한다.



- 물건 스캔 -

요즘 무인 편의점들이 늘고 있다. 무인 편의점이 등장한 초반에는 소비자들이 바코드기에 물건을 하나하나 찍어야 했지만 요즘의 무인 편의점에서는 계산대 위에 물건을 올려두기만 해도 카메라가 물건을 인식하고 편의점 물품에 대한 데이터베이스에 기초하여 가격을 책정한다. 이러한 방식은 속도를 중요시 생각하는 인간에게는 매우 중요한 기술이 되었다고 생각한다.

3. 본인의 제안 사항

1) 해당 기술에 대한 본인의 독창적인 생각, 제안, 활용 방안 등

이미 이미지 분석 및 인식 기술에 대한 여러 가지 기술들이 나와 있는 상태에서 새로운 활용 방안을 생각해 낸다는 것이 쉽지 않았다. 그래도 나는 현재 우리에게 가장 큰 문제인 코로나가 생각났고 건강이라는 단어가 머리에 떠올랐다. 또 건강이라는 단어 자체도 너무 광범위하다고 생각했고 나는 우리에게 어떠한 아이디어가 우리의 일상에 조금 더 가깝게 접근할 수 있을까 생각했고 그 결과 지금부터 내가 소개할 활용 방안을 선택하게 되었다.

내가 생각한 활용 방안은 소변의 RGB로 우리의 건강상태를 체크하는 것 이였다. 변기 안쪽을 보는 방향으로 변기 커버에 RGB를 분류하는 카메라를 설치한다. 이 카메라는 소변 색깔을 인식하여 data로 저장한다. 소변에 대한 전체적인 데이터베이스를 토대로 변기 사용자의 소변 색깔에 대한 건강상태를 확인한다. 또 카메라는 사용자의 소변 색깔이 분류된 RGB table에서 이상이 있다고 판단되어지는 색깔일 때 사용자에게 신호를 보내 경고를 해준다.

지금까지 활용 방안에 대한 간단한 설명이었고 아래의 코드는 이 활용 방안에 대한 코드를 구현해본 것이다.

```
import cv2 as cv
from sklearn.cluster import KMeans

#CV 처리
def set_img():
    imgBgd = cv.imread("pee.jpg", cv.IMREAD_COLOR)
    imgRgb = cv.cvtColor(imgBgd, cv.COLOR_BGR2RGB)
    x, y, channel = imgBgd.shape
    x = int(x / 2)  # img_box width
    y = int(y / 2)  # img_box height
    dis = imgRgb.copy()
    dis = dis[(x + 25):(x - 25), (y + 25):(y - 25)]

    # K-mean 알고리즘을 사용하여 색깔들의 평균값 결정
    newRgb = dis.reshape((dis.shape[0] * dis.shape[1], 3))
    clt = KMeans(n_clusters=1)
    clt.fit(newRgb)
    p_col = clt.cluster_centers_.astype("uint8").flatten().tolist()
    return p_col
```

```
#RGB에 따라 소변유형 분류
def pee_col_type(color):
    r, g, b = color
    p_type = 0
    if r >= 200 and r <= 255:
        if g >= 200 and g <= 225 and b >= 200 and b <= 225:
            p_type = 1
        elif g >= 226 and g <= 240 and b >= 160 and b <= 199:
            p_type = 2
        elif g >= 180 and g <= 199 and b >= 100 and b <= 159:
            p_type = 3
        elif g >= 180 and g <= 220 and b >= 70 and b <= 100:
            p_type = 4
        elif g >= 180 and g <= 220 and b >= 40 and b < 70:
            p_type = 5
        elif g >= 120 and g <= 160 and b >= 100 and b <= 140:
            p_type = 7
        elif g >= 150 and g <= 190 and b >= 100 and b <= 130:
            p_type = 8
    elif r >= 151 and r <= 199:
        if g >= 140 and g <= 180 and b >= 100 and b <= 140:
            p_type = 6
        elif g >= 170 and g <= 200 and b >= 120 and b <= 150:
            p_type = 9
    elif r >= 100 and r <= 150:
        if g >= 90 and g <= 140 and b >= 80 and b <= 130:
            p_type = 1
        elif g > 113 and g <= 140 and b >= 0 and b <= 40:
            p_type = 2
        elif g >= 60 and g <= 113 and b >= 0 and b <= 40:
            p_type = 3
        elif g >= 90 and g <= 130 and b >= 40 and b <= 90:
            p_type = 4
        elif g >= 90 and g <= 130 and b >= 20 and b < 40:
            p_type = 5
    return p_type
```