碩 士 學 位 論 文

자동차 운전자의 전방주시 태만 방지 시스템

A study of protection system for driver attention-free

高麗大學校 大學院
컴퓨터 電波通信工學科

韓 薰 熙

2017年 6月

吳 泰 元 敎授指導

碩 士 學 位 論 文

자동차 운전자의 전방주시 태만 방지 시스템

A study of protection system for driver attention-free

이 論文을 工學 碩士學位 論文으로 提出함

2017年 6月

高麗大學校 大學院
컴퓨터 電波通信工學科

韓 薰 熙

韓 薰 熙의 工學 碩士學位 論文

審査를 完了함.

2017年 6月

委員長 오 태 원 (印)

委 員 육 동 석 (印)

委 員 이 원 준 (印)

A Study of Protection System for Driver Attention-free

Hun-Hee Han.

Advised by Professor Tae-Won Oh

Department of Radio Communications Engineering

The Graduate School, Korea University

Abstract

The most common cause of traffic accidents is the negligence in keeping eyes forward, which accounts for 70 percent of all traffic accidents in the nation. The driver's act of negligence in keeping eyes forward includes active change of one's view and passive drowsiness as well, but as to interpret it in mechanically recognizable term, it can be defined as sudden change of facing angle and eyelid closure. These driver's actions that cause traffic accidents are closely related to the vehicle speed, so if one moves at high speed, even a very short negligent act can cause severe accidents.

In this study, we have studied a system that can detect the above-mentioned accident causing action to warn the driver in advance, and developed a "Protection System for Driver Attention-free" App. Over 95% of success was observed when eyelids closure for 1 second repeated on a fixed face. And it recorded over 90% success when tried the same eyelids close on up, down, left and right directed face. However, as the only capability of detecting eyelid-closed time for 1 second would not be satisfying for actual protection, a faster processor and application program should be prepared for practical implementation.

Furthermore, we developed an APP to prevent phone conversations for a car driver which are the main causes of negligence of keeping eyes forward. It is possible to prevent a phone conversation during a drive by turning the phone to silent mode and sending a text message in response to the incoming call. The probability of accurately sending a text message for an incoming call was 96.9% on average, and accurately recognizing the driving speed was about 83%. If it is supported legally and this app becomes obliged, it can be very helpful to prevent forward gaze neglect caused by the phone in operation.

목 차

<표 목차>

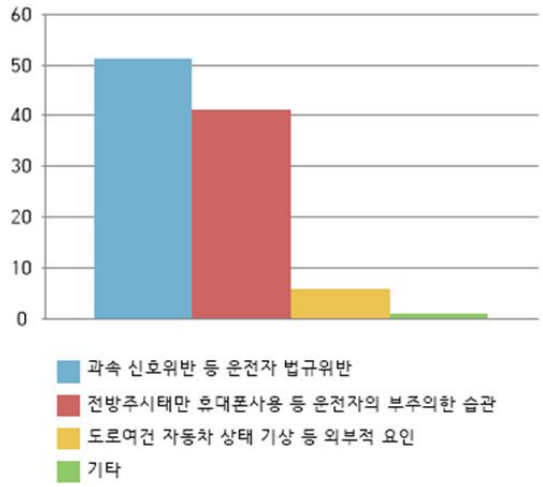
<그림 목차>

제1장 서 론

2007년부터 2015년까지의 도로교통공단 교통사고 분석 시스템에 의하면 안전운전 불이행이 전국 교통사고 사망자 수의 약 70%로 가장 많은 비율을 차지하고 있다. 운전에 집중하지 못해 나타나는 안전불감증이나 전방주시 태만의 결과로 일어나는 사고가 이에 해당한다.

교통안전공단의 2012년 안전운전 불이행 유형 및 원인 분석 연구에 따르면 일반 국민 1,000명을 대상으로 교통안전 운전의식 실태조사를 시행한 결과 자동차 교통사고의 주된 원인으로 과속, 신호위반 등 운전자 법규위반에 이어 전방주시 태만 및 휴대폰 사용 등 운전자의 부주의한 습관이 무려 41.2%로 2위를 차지했다.

그림 1.1 자동차 교통사고의 주된 원인



출처: 교통안전공단 교통사고 안전 불이행 유형 및 원인분석 연구

동일한 1,000명을 대상으로 조사한 결과 전방주시 태만이나 주의력 산만으로 인해 교통사고가 날 뻔한 경우는 22.5%로 나타났다. 전방주시 태만을 야기하는 하나의 종류로 볼 수 있는 졸음운전 경험은 한 달간 운전 경험이 있는 537명을 대상으로 분석한 결과 26.1%로 나타났다. [1] 기존의 연구와 통계를 통해 자동차 운전자들이 운전 중의 전방주시 태만과 졸음운전 행위를 매우 심각하게 인식하고 있다는 것을 알 수 있다.

졸음운전 방지에 관한 연구 논문은 이미 발표된 바가 있지만, 기존의 스마트폰을 이용한 졸음운전 방지시스템의 논문은 운전자의 눈 감지율에만 연구가 집중되어 있다. [2] 실질적으로 졸음이 심하게 올 때는 고개가 떨어지기도 하는데, 이때 기존의 시스템은 오히려 눈 인식이 무력화되어 본연의 기능을 상실하게 된다. 즉, 운전자가 깊은 졸음으로 인해 전방을 전혀 주시하지 못하는 상황에서는 아무런 알림을 주지 못한다.

좌우 풍경을 바라보거나 동승자와의 대화 등, 주의산만으로 인해 전방을 주시하지 않을 때도 같은 이유로 얼굴 및 눈 검출 자체가 불가능하게 되어 알림을 주지 못한다.

위에서 제시한 두 문제점을 해결할 수 있는 전방주시 태만 방지 시스템의 개발이 요구된다. 졸음방지기능을 활용하고 더 나아가 자동차 운전자의 얼굴이 정면으로 인식이 되지 않을 때 알림을 주는 기능을 첨가하면 졸음운전을 포함한 전방주시 태만 행위를 광범위하게 방지하는 데 크게 도움이 될 것이다.

운전 중 걸려온 전화를 받는 행위 또한 자동차 운전자의 전방주시 태만을 일으키는 중요한 요인이다. 자동차 운전 중 걸려온 전화에 운전 중임을 자동으로 응답해 준다면, 안전운전 불이행 행위 중 하나인 운전 중 전화통화 행위를 미연에 방지할 수 있다. 본 논문에 제시한 애플리케이션을 통해 교통사고의 약 70%를 차지하는 전방주시 태만과 졸음운전의 예방이 이루어진다면 교통사고율을 크게 감소시킬 수 있다.

그림 1.2 기존 졸음 방지 시스템의 문제점

1.1. 자동차 교통사고 실태

표 1.1은 도로교통공단 TAAS 교통사고 분석시스템을 이용해 2007년부터 2015년까지의 전국 교통사고 총사망자 수와 안전운전 불이행으로 인한 사망자 수를 토대로 안전운전 불이행으로 인한 사망자 비율을 계산해 본 결과를 나타낸 것이다. [3] 운전에 집중하지 못해 나타나는 안전

불감증이나 전방주시 태만의 결과로 일어나는 사고가 이에 해당한다. 표 1.1을 통해 안전운전 불이행으로 인한 교통사고 사망자의 비율이 매년 70%에 근접하는 것을 확인할 수 있으며 자동차 운전자의 전방주시 태만이 얼마나 심각한 사회적 문제인지 직감할 수 있다.

표 1.1 도로교통공단 TAAS 교통사고 분석 시스템 - 가해자 운전자 법규 위반 유형별 교통사고

기준년도	2007	2008	2009	2010	2011	2012	2013	2014	2015
전국 교통사고 총사망자수	6,166	5,870	5,838	5,505	5,229	5,392	5,092	4,762	4,621
안전운전불이행 사망자수	4,337	4,078	4,098	3,829	3,709	3,872	3,673	3,372	3,165
안전운전불이행 사망자 비율(%)	70.3	69.5	70.2	69.6	70.9	71.8	72.1	70.8	68.5

출처: TAAS 교통사고분석시스템

1.2 관련 법규

도로교통법에 전방주시 태만으로 발생하는 사고를 방지하기 위해 다음과 같은 조항이 있다. 도로교통법 제49조 10항에는 “운전자가 자동차를 운전할 때 휴대용 전화를 사용하지 아니할 것”, 11항에는 “자동차 등의 운전 중에는 방송 등 영상물을 수신하거나 재생하는 장치가 운전자가 운전 중 볼 수 있는 위치에 영상이 표시되지 아니하도록 할 것”이라고 명시되어 있다. [4]

본 논문에서는 “자동차 운전자의 전방주시 태만 방지 시스템”과 “운전 중 통화방지 시스템”을 소개하고 이에 관한 실험결과를 제시할 것이다. 이어서 안드로이드와 OpenCV의 동향에 대해 살펴보고 마무리 짓고자 한다.

제2장 전방주시 태만 방지 시스템

2.1. 자동차 운전자의 사고 유발 행위의 정의

기존에 발표된 졸음운전 방지 시스템은 졸음운전 시 눈 감김 현상을 졸음운전 사고 유발 행위로 정의하여 눈 깜빡임 감지 위주로 개발되었다. 이러한 시스템의 가장 큰 문제점은 졸음으로 인한 고개 떨어짐, 주의 산만으로 인한 전방주시 태만 시에는 전혀 알림을 주지 못하는 것이다.

본 시스템에서는 이러한 점을 보완하기 위해 눈을 감는 행위에 추가로 상, 하, 좌, 우로 고개를 돌려 전방을 주시하지 않는 경우에도 운전자에게 경고를 주도록 하였다. 그리고 전방주시 태만 및 졸음운전 시 고개 떨어짐 현상도 사고를 일으킬 수 있는 행위로 규정하여 알림을 줄 수 있도록 제작하였다.

그림 2.1 전방주시 태만 행위의 정의

눈을 감는 행위

졸음으로 인해 고개를 떨어뜨리는 행위

상, 하, 좌, 우로 고개를 돌리는 행위

2.2. 시스템의 구성

2.2.1. 전체 시스템 작동 개요

본 시스템은 OpenCV Android를 이용해 추가적인 장비 없이 안드로이드 스마트폰만으로 전방주시 태만 및 졸음운전을 방지하기 위해 개발되었다. OpenCV Android를 이용해 JNI에 프로그래밍하여 운전자의 얼굴과 눈을 검출할 수 있도록 하였다. 얼굴과 눈의 검출 여부를 통해 전방주시 태만과 눈 감김 현상을 감지해 안드로이드의 JAVA 영역으로 데이터를 가져와 운전 중 전방주시 태만, 졸음운전 시에 경고를 주도록 구성하였다.

그림 2.2 카메라로 얼굴을 검출해 내는 장면

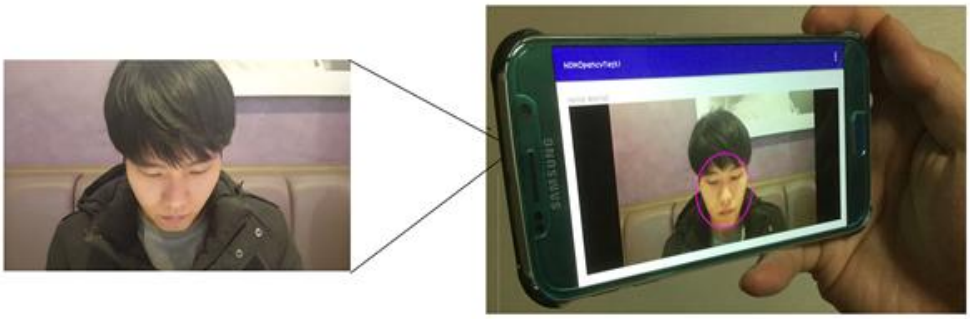


그림 2.2과 같이 스마트폰 카메라로 실시간 전송되는 데이터를 이용하여 그림 2.3에 제시한 장면과 같이 전방을 주시하는 경우 알림을 주지 않지만, 상, 하, 좌, 우로 고개를 돌리거나 아래를 바라보는 경우, 눈을 감은 경우를 전방주시 태만으로 인지하여 알림을 준다.

그림 2.3 운전자의 상태에 따른 얼굴 및 눈 인식에 대한 알림 구분

2.2.2. OpenCV Android 영상처리 구조

Haar Cascade classifier는 특정한 형태를 띠는 물체를 검출해 낼 때 사용하는 가장 대표적인 방법이다. 본 시스템은 얼굴 및 눈 추적에 OpenCV를 사용하기 위해 구성한 JNI단에 haarcascade_frantalface_alt.xml 파일로 검출한 얼굴 영역을 표시한 뒤 haarcascade_eye_tree_eye-glasses.xml 파일로 얼굴 영역 내에서 눈을 검출해내도록 구현하였다. 즉, 눈 검출 확률을 높이기 위해 얼굴을 먼저 인식한 후 다시 그 안에서 눈을 검출해 내도록 구성하였다. JNI단에서 검출한 얼굴과 눈을 Java CameraView로 받아와 얼굴 영역은 분홍색의 원, 눈 영역은 붉은색의 작은 원으로 표시하도록 하였다.

그림 2.4 OpenCV Android 영상처리 구조도

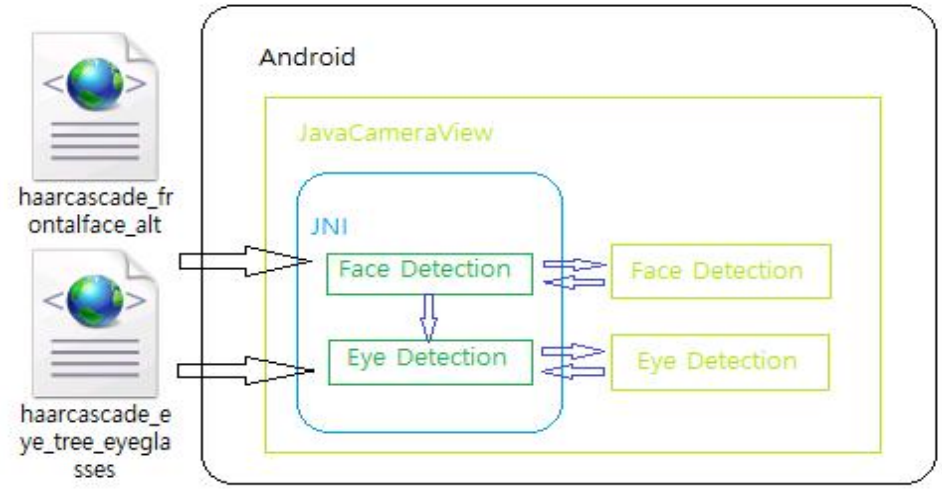


그림 2.5 JNI 단에 입력한 얼굴 및 눈 검출 코드

```
//-- Detect faces
face_cascade.detectMultiScale(frame_gray, faces, 1.1, 2, 4 | CV_HAAR_SCALE_IMAGE, Size(30, 30));

for (size_t i = 0; i < faces.size(); i++) {
  Point center(faces[i].x + faces[i].width * 0.5, faces[i].y + faces[i].height * 0.5);
  ellipse(frame, center, Size(faces[i].width * 0.5, faces[i].height * 0.5), 0, 0, 360,
          Scalar(255, 0, 255), 4, 8, 0);

  Mat faceROI = frame_gray(faces[i]);
  //std::vector<Rect> eyes;

  //-- In each face, detect eyes
  eyes_cascade.detectMultiScale(faceROI, eyes, 1.1, 2, 0 | CV_HAAR_SCALE_IMAGE, Size(30, 30));


  if(faces.size()>0) {
    for (size_t j = 0; j < eyes.size(); j++) {
      Point center(faces[i].x + eyes[j].x + eyes[j].width * 0.5,
                   faces[i].y + eyes[j].y + eyes[j].height * 0.5);
      int radius = cvRound((eyes[j].width + eyes[j].height) * 0.25);
      circle(frame, center, radius, Scalar(255, 0, 0), 4, 8, 0);


    }
  }
```

2.2.3. 전방주시 태만 방지 시스템의 순서도

그림 2.6은 전방주시 태만 방지 시스템의 순서도이다.

애플리케이션을 실행하면 얼굴 영역의 검출 여부를 확인한다. 얼굴 영역이 검출되지 않으면 알림을 주고 검출 시에는 얼굴 영역을 분홍색 원으로 표시한다.

얼굴 영역이 검출되었을 경우 눈의 검출 여부를 확인한다.

눈 검출이 되지 않은 경우 알림을 준다. 눈이 검출된 경우 눈 영역은 붉은색 원으로 표시한다.

즉, 얼굴이 검출되지 않거나 눈이 검출되지 않을 경우에는 졸음으로 인해 눈을 감거나 전방을 주시하지 않은 것으로 간주하여 알림을 주는 시스템이다.

그림 2.7은 눈이 검출된 사이즈로 눈의 감지 여부를 확인하여 안드로이드의 JAVA 영역에 구성해 놓은 이벤트를 불러와 알림을 줄 수 있도록 구현해 놓은 소스코드이다.

그림 2.8과 같이 안드로이드의 사운드 시스템을 이용하여 알림을 줄 수 있으며 스마트폰의 진동, 스마트폰과 연동된 블루투스 장비를 통한 알림 기능 등을 이용해 다양한 방식으로 운전자에게 효과적으로 알림을 줄 수 있다.

그림 2.6 전방주시 태만 방지 시스템의 순서도

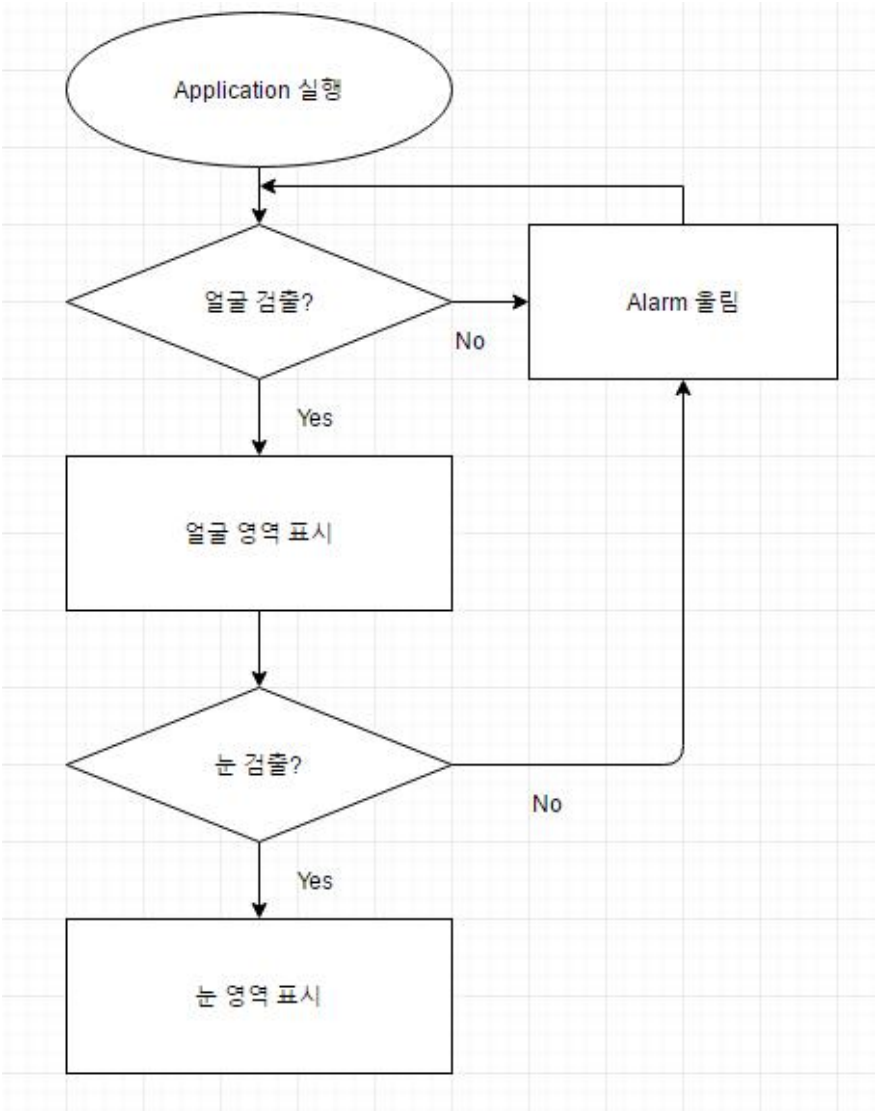


그림 2.7 눈의 검출 여부에 따른 이벤트를 불러오기 위한 소스코드

```
if (eyes.size() == 0) {
  jclass jCallJava = GlobalEnv->FindClass("com/example/hooni/ndkopencvtest1/MainActivity");
  jmethodID BBBBBB = GlobalEnv->GetStaticMethodID(jCallJava, "BBBBBB", "()V");
  GlobalEnv->CallStaticVoidMethod(jCallJava, BBBBBB);
  // Alarm On
}

else if (eyes.size() > 0)

{
  jclass jCallJava = GlobalEnv->FindClass("com/example/hooni/ndkopencvtest1/MainActivity");
  jmethodID AAAAAA = GlobalEnv->GetStaticMethodID(jCallJava, "AAAAAA", "()V");
  GlobalEnv->CallStaticVoidMethod(jCallJava, AAAAAA);
  //Alarm Off

}
```

그림 2.8 알림 구현 소스코드 (RingtoneManager 이용)

```
private void call(){
    //        Toast.makeText(mContext, "소리울림", Toast.LENGTH_SHORT).show();
    Uri alert = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_RINGTONE);

    try {
        if(!mPlayer.isPlaying()){
            mPlayer.setDataSource(mContext, alert);
            mPlayer.setAudioStreamType(AudioManager.STREAM_RING);
            mPlayer.setLooping(true);
            mPlayer.prepare();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
    if(!mPlayer.isPlaying()){

        mPlayer.start();

    }
```

제3장 전방주시 태만 방지 시스템 실험

3.1. 얼굴 각도에 따른 작동 기준

표 3.1은 20명을 표본으로 정면으로부터 움직인 얼굴의 각도에 따른 작동 기준각의 평균을 기록해 놓은 것이다. 좌, 우로는 약 40도를 기준으로 인식 여부가 결정되며 상으로는 45도, 하로는 30도를 기준으로 인식 여부가 결정되는 것을 확인할 수 있다. 그림 3.1에 이를 이해하기 쉽도록 도식화해 놓았다.

그림 3.1 얼굴 각도에 따른 작동 기준

표 3.1 정면으로 부터의 얼굴 각도 변화에 따른 작동 기준 각

	눈			
방향	좌	우	상	하
작동 기준 각도 (degree)	40	40	45	30

3.2. 얼굴 및 눈 감지 후 알림을 주는데 까지 걸리는 응답시간

눈 감김 또는 전방주시 태만을 감지하고 알림을 주는 데까지 걸린 시간을 한 차수 당 10회씩 측정한 후 평균값을 내어 총 10차분을 표 3.2에 작성해 놓았다. 그 결과 최고 반응 속도 '0.31초', 최저 반응 속도 '0.43초'로 평균 '0.35초' 이내에 알림이 작동하는 것을 확인할 수 있었다.

표 3.2 얼굴 및 눈 감지 후 알림을 주는데 까지 걸리는 응답시간

시험 차수	1	2	3	4	5	6	7	8	9	10	평균
응답 시간 평균	0.33 sec	0.43 sec	0.36 sec	0.36 sec	0.43 sec	0.32 sec	0.32 sec	0.31 sec	0.34 sec	0.33 sec	0.35 sec

3.3. 눈감지 인식률 및 안전을 위해 필요한 시스템의 응답시간

눈을 감는 경우, haarcascade_eye_tree_eyeglasses.xml를 이용한 눈 검출이 불가능해진다. 이를 이용하여 졸음운전 시 눈 감김 현상을 검출하게 된다. 눈 감김 현상 감지율은 전방주시 태만 방지에 매우 중요한 요소이다.

표 3.3 정지거리와 전방주시 태만 시 질주하는 거리 비교

	일반 정지거리 (m)	빙판길 정지거리 (m)	1초 전방주시 태만 시 (m)	2초 전방주시 태만 시 (m)
10km/h	2.5	3.3	2.8	5.6
20km/h	6	9	5.6	11.1
30km/h	10	18	8.3	16.7
40km/h	16	29	11.1	22.2
50km/h	22	43	13.9	27.8
60km/h	30	59	16.7	33.3
70km/h	36	78	19.4	38.9
80km/h	47	100	22.2	44.4
90km/h	56	124	25.0	50.0
100km/h	69	151	27.8	55.6

TS 교통안전공단에서 제공하는 자료를 바탕으로 작성한 표 3.3을 통해 시속 20km/h로 달릴 경우 건조한 아스팔트 포장 도로에서의 자동차 정지거리는 6m인 것을 확인할 수 있다. [5, 6] 동일한 속도에서 1초 동안 졸거나 전방주시를 태만하게 할 경우, 앞을 보지 않은 상태로 5.6m를 질주하게 된다. 이는 거의 6m에 이르는 거리로 정지거리를 고려하면 매우 위험한 것이다. 더 빠른 속도에서의 1초 동안의 전방주시 태만 또는 졸음운전은 사고 발생 시 더 심각한 피해를 야기하게 된다. 이러한 이유로 1초를 눈을 감고 있는 기준 시간으로 정하여, 본 시스템의 눈 깜빡임 인식률을 측정해 보았다. 1초 간격으로 200번씩 눈을 감았다 뜨기를 5회 시행해 본 결과 표 3.4의 값을 얻어 평균 95%의 정확성을 확인하였다.

표 3.4 눈 깜빡임 감지 성공률

회수	1회	2회	3회	4회	5회	평균
감지 성공률	90%	88%	100%	100%	98%	95%

3.4. 기존 시스템과 본 시스템의 고개를 돌릴 때의 눈감지 인식률 비교

3.3에 기술한 이유로 1초를 기준으로 상, 하, 좌, 우로 표 3.1에 기재한 작동 기준 각 이상으로 고개를 돌릴 시의 눈 감지 인식률을 측정해 보았다. 기존의 시스템은 상, 하, 좌, 우로 고개를 돌릴 시 눈의 인식이 불가

능해지므로 사방으로 모두 0%의 인식률을 보였다. 하지만 본 시스템에서는 상 90.2%, 하 92.2%, 좌 91.2%, 우 91.2%의 인식률을 보여 사방으로 90% 이상의 인식률을 확인할 수 있었다.

표 3.5 기존 시스템과 본 시스템의 사방으로 고개 돌릴 시의 눈 깜빡임 감지 성공률

	상	하	좌	우
본 시스템: 기준 각 이상으로 고개를 돌릴 시 인식률	95%	97%	96%	96%
본 시스템: 고개를 돌릴 시 눈 감지 인식률	90.2%	92.2%	91.2%	91.2%
기존 시스템: 고개를 돌릴 시 눈 감지 인식률	0%	0%	0%	0%

3.5. Haar feature와 LBP feature를 이용하여 전방주시 태만 방지 시스템을 구현하였을 때의 차이점

본 시스템은 OpenCV에서 제공하는 haarcascade 방식을 이용하여 Face Detection을 시행하였다. LBP 방식으로 이를 대체해도 검출기능에는 변화는 없다. 하지만 두 방식에는 차이점이 존재한다.

1) LBP feature 방식을 사용할 때:

Haar feature 방식보다 안드로이드 스마트폰에서 빠르고 쾌적하게 작동한다. 그러나 얼굴 이외의 다른 물체도 얼굴로 인식하여 오 표시 하는 경우가 많아 정확도 면에서 상대적으로 부족하다.

2) Haar feature 방식을 사용할 때:

LBP feature 방식보다 얼굴을 더욱 정확하게 인지하였지만 비교적 지연 현상이 심해 고해상도에서 작동 시 쾌적하지 못하다. 따라서 해상도 조절기능을 이용해 단말기 성능에 따라 사용자가 이를 조절해야 더욱 정확하고 쾌적한 사용이 가능하다.

본 논문에서는 검출의 정확도에 우위를 두어 Haar feature 방식을 채택하였다. 두 방식의 장점만을 취해 더욱 빠르고 정확한 인식률을 가진 feature가 개발된다면 더욱 쾌적한 환경에서 정확한 서비스가 가능해질 것이다.

제4장 자동차 운전 중 통화 방지 시스템

4.1. 시스템의 구성

4.1.1 전체 시스템 작동 개요

사람의 도보 속도는 평균 5km/h이다. 이를 바탕으로 7km/h 이상의 속도일 경우를 자동차가 주행 중으로 정의했다. 7km/h 이상의 속도로 주행할 경우 스마트폰이 현 상황을 자동차 주행 중으로 인식하여 전화가 수신될 때 송신자의 연락처로 문자메시지를 자동으로 송신하고 벨 소리를 무음으로 처리하게 된다. 이 시스템을 통해 자동차 운전자의 주행 중 전화 통화를 방지하여 운전 중 전화 통화 및 휴대폰 사용으로 인해 발생하는 전방주시 태만 관련 교통사고의 감소에 크게 기여할 수 있다.

그림 4.1 자동차 운전 중 통화 방지 시스템 작동방식

4.1.2 자동차 운전 중 통화 방지 시스템의 순서도

그림 4.2는 자동차 운전 중 통화 방지 시스템의 작동 순서도이다. 애플리케이션을 실행하면 벨 소리와 전화 수신 기능이 사용자가 기본으로 설정해 놓은 상태로 작동한다. 이를 그림 4.2에 Normal Mode라고 명명하였다. 자동차의 속도가 7km/h 이상일 경우 일반적인 인간의 도보 속도를 초과하므로 자동차 주행 중으로 인식하여 스마트폰을 무음 모드로 변경시키고 전화 수신 시 수신된 번호로 현재 운전 중임을 문자메시지로 자동전송하게 된다. 다시 7km/h 이하의 속도가 되면 도보 상황이거나 자동차 서행 상황으로 인식하여 전화가 걸려올 경우 벨 소리로 알림을 받게 된다.

즉, 일정 속도 이상으로 자동차 주행 시 걸려온 전화를 무음 처리하고 수신된 번호로 문자메시지를 자동으로 전송해 현재 운전 중임을 알려주어 자동차 운전자의 전방주시 태만을 방지하는 기능을 한다.

그림 4.2 자동차 운전 중 통화 방지 시스템의 순서도

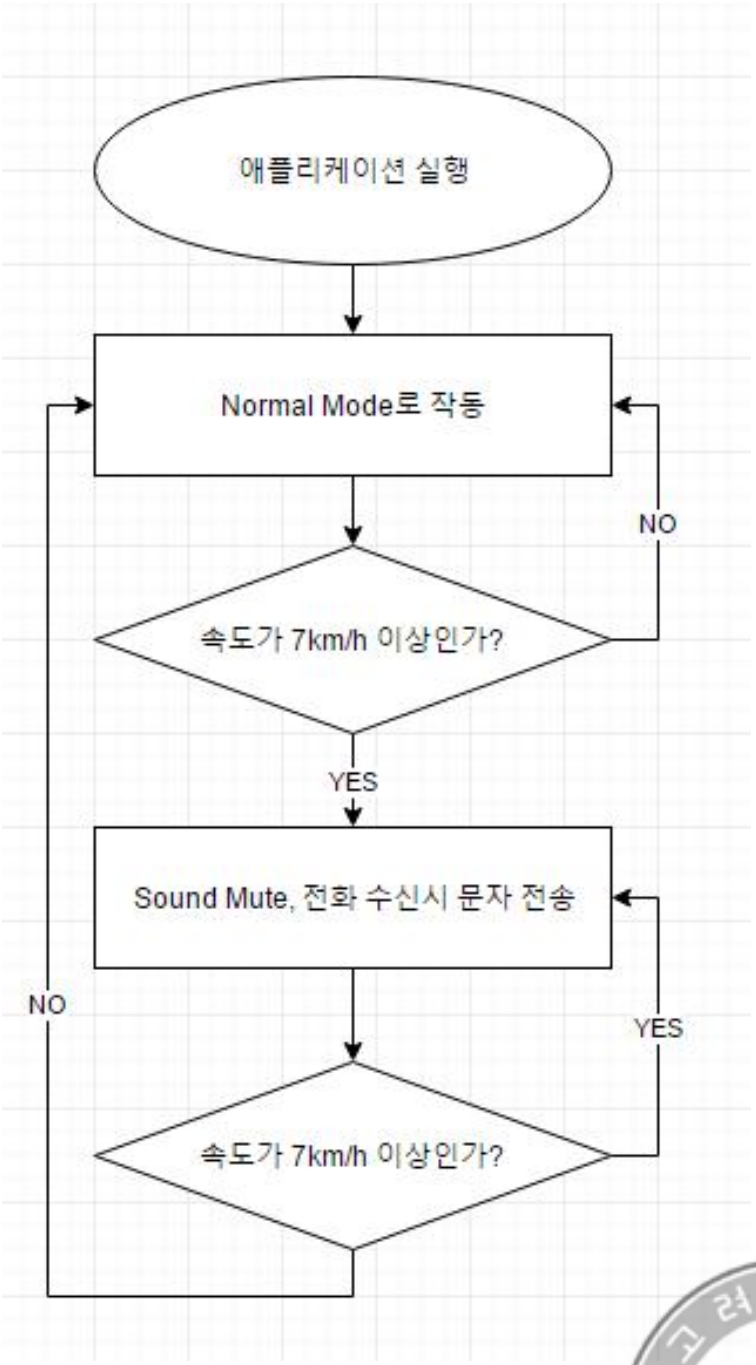


그림 4.3은 스마트폰의 GPS를 이용하여 현재 속도를 받아오기 위해 작성한 소스코드이다.

그림 4.3 스마트폰 GPS 기능을 이용한 속도계 소스코드

```
public void onLocationChanged(Location location) { // 위치가 변경되었을때에 사용 하는 메소드
    if (location != null) { // location에 데이터가 들어있지 않을 경우
        mySpeed = location.getSpeed()*3600/1000; // 속도를 얻어와 myspeed에
        // 저장 원래 단위가 m/s이므로 km/h로 변경하기 위한 수식을 입력함
        if (mySpeed > maxSpeed) {//최고속도가 현재속도보다 작은 경우
            maxSpeed = mySpeed; // 현재속도를 최고속도에 대입
        }
        CurrentSpeed.setText("Current Speed : " + mySpeed+ " km/h "); //현재속도를 textview에 표시함
        MaxSpeed.setText("Max Speed : " + maxSpeed + " km/h"); //최고속도를 textview에 표시함
    }
}
```

그림 4.4는 자동차 운전 중 걸려온 전화의 수신번호를 자동으로 가져와 문자메시지를 전송하는 기능을 하는 소스코드이다.

그림 4.4 자동차 운전 중 수신된 전화번호로 송신하기 위한 소스코드

```
if(arg1.getAction().equals("android.intent.action.PHONE_STATE")){//폰에 전화가 걸려온 경우
    String state = arg1.getStringExtra(TelephonyManager.EXTRA_STATE);
    if (state.equals(TelephonyManager.EXTRA_STATE_RINGING)){// 걸려온 전화의 전화번호를 후킹하는 부분
        String number = arg1.getStringExtra(TelephonyManager.EXTRA_INCOMING_NUMBER);
        Log.e(TAG, "AnswerCallBroadcastReceiver incoming number : " + number);

        if(MapsActivity.insta != null){// map액티비티의 인스타가 널이 아닐경우
            MapsActivity.number = number; //전화번호를 맵액티비티에있는 넘버에 입력함
        }

    }
    else if(state.equals(TelephonyManager.EXTRA_STATE_IDLE)){ // 그렇지 않을경우 아이들한 상태로 유지
        Log.d(TAG, "AnswerCallBroadcastReceiver Inside EXTRA_STATE_IDLE");
    }
}
```

그림 4.5는 자동차 운전 중 수신된 번호로 문자메시지를 자동으로 전송해주는 소스코드이다.

그림 4.5 자동차 운전 중 수신된 전화번호로 SMS를 보내는 소스코드

```
    // 속도가 7km 이상이면 SMS발송
    if (mySpeed > MAX_KMH) {
        Log.d("TAG", "속도 7km이상");
        // SMS발송(전화가 왔을때만)
        Toast.makeText(MapsActivity.this, "7km 이상 달라는중~", Toast.LENGTH_SHORT).show();
        if(number != null){
            aManager.setRingerMode(AudioManager.RINGER_MODE_SILENT);
            sendSMS(number, "자동차가 7km 이상으로 주행 중입니다.");
        }
    } else {
        if(number != null){
           // sendSMS(number, "테스트 문자");
        }

        Log.d("TAG", "속도 7km이하");
    }
}
```

그림 4.6은 속도가 7km/h 이하일 경우 무음 모드로 변경되었던 상태를 다시 벨 소리 모드로 변경해 주는 소스코드이다.

그림 4.6 7km/h 이하의 속도일 때의 벨소리 모드로 변환하는 소스코드

```
@Override
protected void onDestroy() {
    super.onDestroy();
    if(aManager != null)
        aManager.setRingerMode(AudioManager.RINGER_MODE_NORMAL);

}
```

제5장 자동차 운전 중 통화 방지 시스템 실험

5.1. 자동차 운전 중 전화 수신 시 문자 메시지 자동 전송 성공률

자동차 운전 중 전화가 걸려올 때 문자 메시지 자동 전송 성공률을 확인해 보기 위해 차수 별로 100회의 전화 수신을 반복하여 총 10차에 걸쳐 문자메시지 전송 성공률을 측정한 결과를 표 5.1에 기록하였다. 9회차를 제외하고 모두 94% 이상의 문자메시지 전송 성공률을 보였으며 평균 96.9%의 전송 성공률을 보였다.

표 5.1 자동차 운전 중 수신전화에 대한 문자 메시지 전송 성공률

시험 차수	1	2	3	4	5	6	7	8	9	10	평균
전송 성공률 (%)	100	94	98	98	100	98	100	98	83	100	96.9

5.2. 안드로이드폰 GPS를 이용한 속도 측정의 정확도

Android developers에서 제공한 자료에 의하면 getSpeedAccuracy MetersPerSecond()는 반경 68%의 신뢰도를 갖는다고 한다. 위치에 대한 속도를 중심으로 하나의 원을 그리고 속도의 정확도에 관한 원을 그릴 경우 실제 속도가 이 원 안에 들어올 확률이 68%라는 설명이다. [7] 실

사용에서의 속도 측정의 정확도를 살펴보기 위해 1분간 자동차를 일정한 속도로 주행하여 평균을 낸 결과와 자동차의 계기판 속도를 비교해 표 5.2에 작성해 놓았다. 이 표를 통해 평균적으로 약 17%의 오차율을 보여 약 83%의 주행 속도 인식 정확성을 확인할 수 있었다.

표 5.2 자동차 주행 시 자동차 계기판과 앱으로 측정한 속도 비교

계기판 속도 (km/h)	0	6	7	8	10	20	30	40	50
앱 속도 (km/h)	0	5	5.5	6.5	8	14	24	33	46
오차율 (%)	0	16.6	21.4	18.7	20.0	30.0	20.0	17.5	8.0

제6장 안드로이드

6.1. 안드로이드란

안드로이드는 스마트폰 태블릿 PC를 비롯한 여러 휴대용 장치를 위해 개발된 OS와 미들웨어, 핵심 애플리케이션을 포함하는 집합을 말한다. JAVA 언어를 통해 Android Studio나 Eclipse를 사용하여 개발할 수 있으며 Dalvik Virtual Machine과 ART(Android Runtime) 런타임 라이브러리를 제공하여 컴파일된 바이트 코드를 구동할 수 있다. 또한, SDK (Software Development Kit)를 통해 다양한 Tool과 API(Application Programming Interface)를 제공한다. [8]

안드로이드는 스마트폰의 급격한 발전과 세계적인 확산으로 인해 빠르게 성장해 왔다. 시장을 먼저 선점한 애플의 IOS가 강세였으나 안드로이드는 2008년 10월 21일에 오픈 소스로 선언함으로써 고품질 서비스와 수많은 애플리케이션을 갖추게 되어 IOS를 빠른 속도로 따라잡았다.

6.2. 안드로이드의 버전별 특징

앤디 루빈이 2003년 미국 캘리포니아 주의 팔로알토에 설립한 안드로이드는 Alpha 버전부터 가장 최신 버전인 7.0 Nougat까지 꾸준히 발표

하며 지속적인 발전을 하고 있다. 안드로이드에 대한 이해와 발전 방향을 살펴보기 위해 각 버전과 특징을 간략하게 정리해 보았다. [9, 10, 11]

1) 버전 1.0

2008년 9월 23일에 공개한 정식으로 출시된 최초의 안드로이드 버전이다. SDK가 함께 배포되었으며 2008년 10월 22일에 출시한 T-Mobile G1에 탑재되었다. 커널 버전은 2.6.25이다.

2) 버전 1.1 프티 푸르 (Petit Four)

Petit Four는 프랑스어로 한입에 들어가는 과자라는 뜻이다. 각종 먹거리 이름으로 버전의 이름을 짓는 것은 이때부터 시작되었다. 1.0과 동일한 커널 버전을 사용했고 G1에서 나타난 문제점들을 보완한 버전이다.

3) 버전 1.5 컵케이크 (Cupcake)

2009년 4월 30일에 공개되었으며 커널 버전은 2.6.27이다. API Level은 3이며 가상키보드, 라이브 폴더, 음성인식 지원, 풀 스크린 위젯, 홈 스크린 기능이 추가되었다. 이 버전부터 한국어를 지원하였다.

4) 버전 1.6 도넛 (Donut)

2009년 9월 15일에 공개되었으며 커널 버전은 2.6.29이다. API Level은 4이며 문장을 목소리로 변환해 주는 TTS(Text To Speech) 엔진 기능이 추가되었으며 CDMA가 지원되기 시작했다. 통합검색 기능이 추가되었으며 카메라와 캠코더, 갤러리 등의 UI 업데이트가 이루어졌다.

5) 버전 2.0/2.1 에클레어 (Eclair)

2009년 10월 26일에 2.0 버전이 발표되었다. 커널 버전은 그대로 2.6.29이며 API Level은 5이다. 블루투스 2.1을 도입했고 여러 개의 구글 계정을 동시에 등록할 수 있게 되었다. 멀티터치 공식지원 및 인터넷 브라우저에 주소창이 추가되었으며, HTML5를 지원하게 되면서 멀티미디어와 카메라 기능이 향상되었다. 2.0.1은 마이너 업데이트였으며 2.1 버전은 넥서스 원의 출고 버전으로 라이브 웹 페이퍼 관련 클래스와 전화 신호 세기 모니터링, 음성 녹음, 라이브 배경화면 등의 마이너 업데이트가 이루어졌다.

6) 버전 2.2 프로요 (Froyo)

2010년 5월 20일에 공개되었으며 커널 버전은 2.6.32, API Level은 8이다. Virtual Machine에 새로운 실시간 컴파일러를 적용해 앱 실행속도

를 2~5배 빠르게 하였으며 V8 엔진을 이용하여 자바스크립트 구동 속도를 2~3배 증가시켰다. 메모리 회수 기능 개선을 통해 애플리케이션 구동이 부드러워졌다. 외장메모리에 애플리케이션 설치를 지원하게 되었으며 PC 웹 브라우저에서 안드로이드 마켓 탐색이 가능해졌다. 테더링을 지원하게 되었고 플래시 플레이어 10.1버전을 탑재하여 웹에서 플래시 재생이 가능해졌다.

7) 버전 2.3 진저브레드 (Gingerbread)

2010년 12월 6일 넥서스 S와 함께 공개된 버전으로 커널 버전은 2.6.35, API Level은 2.3.2까지는 9, 2011년 2월 9일 공개된 버전부터는 10을 사용하였다. UI 단순화 및 성능 개선이 이루어졌으며 애플리케이션 및 전원 관리능력, OpenSL ES의 소프트웨어 구현을 통해 오디오 출력 기능을 향상했다. VOIP 사용이 가능해졌으며 회전 벡터, 선형가속, 중력, 자이로스코프, 기압계 등의 다양한 센서와 NFC를 지원하게 되었다.

8) 버전 3.0/3.1/3.2 허니콤 (Honeycomb)

2011년 2월 22일에 3.0 버전을 시작으로 2011년 5월 10일 3.1, 같은 해 7월 15일에 3.2 버전이 공개되었다. 커널 버전은 2.6.36이며 API Level은 11~13이다. 순수 태블릿 전용 OS로 출시되었으며 스마트폰용

앱 구동이 잘 안 되는 경우가 많아 문제가 되기도 했다. 가상 버튼 UI를 적용했고 USB 포트를 이용한 게임패드와 조이스틱을 지원하게 되었다. PC 연결 방식이 MTP로 바뀌었으며 Data와 SD card 파티션을 통합하였다.

9) 버전 4.0 아이스크림 샌드위치 (Ice Cream Sandwich)

2011년 10월 19일 갤럭시 넥서스와 함께 공개되었으며 커널 버전은 4.0.1까지는 3.0.1이며 4.0.2부터는 3.0.8을 사용한다. API Level은 4.0.2까지는 14, 4.0.3부터는 15 버전을 사용한다. 허니콤에 적용된 UI가 스마트폰 형태로 적용되었으며 앱 구동 성능이 크게 개선되었다. 카메라 셔터 반응 시간 감소, GPU 사용 최적화, 안드로이드 빔, 얼굴 인식 잠금 해제가 가장 큰 특징이다.

10) 버전 4.1/4.2/4.3 젤리빈 (Jelly Bean)

2012년 6월 28일 태블릿 PC 넥서스 7과 함께 발표되었다. 커널 버전은 3.0.31이며 API Level은 16이다. SoC 내의 CPU와 GPU 처리를 함께 할 수 있도록 변경하면서 반응 속도가 개선되었다. 전력 효율을 향상시켰으며 단어 예측기능을 갖춘 키보드가 적용되었다. 카메라 앱의 순간 포착 기능이 추가되었다. 4.2버전은 2012년 10월 29일에 공개되었으며

넥서스 4와 넥서스 10이 레퍼런스 모델이었다. 커널 버전은 3.4.0이며 API Level은 17이다. 4.2 버전의 신기능은 다중 계정 지원, 포토 스피어 카메라, 키보드에 제스쳐 타이핑, 화면보호기, 락 화면에서의 위젯 관리 등이다. 4.3 버전은 2013년 7월 25일 넥서스 7 2세대와 함께 발표되었으며 커널 버전은 3.4.39이고 API Level은 18이다. 에너지 소비가 적어진 블루투스 스마트, OpenGL ES 3.0, TRIM 등이 추가됐다.

11) 버전 4.4 킷캣 (Kitkat)

2013년 9월 4일 넥서스 5와 함께 발표되었으며 커널 버전은 3.4.0, API Level은 19이다. 가상 머신을 Dalvik에서 ART로 교체한 것이 가장 큰 특징이다. 기본 웹 브라우저를 구글 크롬으로 바꾸었으며 애니메이션 프레임워크를 개선하였다. SMS용 퍼블릭 API를 추가하였고 구글 행아웃을 기본 앱으로 사용하였으며, 센서 전력 소모를 감소시켰다.

12) 버전 5.0/5.1 롤리팝 (Lollipop)

2014년 6월 26일 넥서스 5와 넥서스 7 2세대에 제공되었다. 커널 버전은 3.4.0, API Level은 21, 22이다. 머티리얼 디자인이 적용되었으며 알림 기능 및 보안을 강화하였다. Project Volta를 통해 배터리 효율을 높였다. 상단 알림바에 빠른 설정 기능이 추가되었고 ART가 기본으로 적

용되었다. 64bit를 지원하게 되었으며 오디오 기능도 개선되었다. Tap & Go 기능으로 NFC를 이용하여 데이터를 주고받을 수 있게 되었다.

13) 버전 6.0/6.0.1 마시멜로 (Marshmallow)

2015년 9월 29일에 넥서스 5X, 넥서스 6와 함께 공개되었으며 커널 버전은 3.10.24이고 API Level은 23이다. 미국에서 안드로이드 페이 기능과 앱 링크 기능을 제공하게 되었다. 또한, USB Type-C와 향상된 배터리 관리 Doze를 지원하였다. 볼륨조절이 더욱 세분되었고 외부 저장소를 공식으로 지원하게 되었다. 시스템 차원에서의 지문인식과 VoLTE를 기본으로 지원했다. 애플리케이션별 권한관리를 통해 권한을 요청할 때마다 허가할 수 있도록 변경되었다.

14) 버전 7.0/7.1/7.1.1/7.1.2 누가 (Nougat)

2016년 8월 23일 발표된 버전으로 LG V20에 최초 탑재되었다. 커널 버전은 3.10.73이고 API Level은 24이다. 다양한 이모티콘과 더 많은 키보드 언어 지원, 멀티윈도우 기능이 생겼으며, VR 기능과 Doze 모드의 향상, 심리스 업데이트, 파일에 기반을 둔 암호화 등이 누가의 가장 큰 특징이다.

6.3. 안드로이드 폰의 점유율

표 6.1은 가트너가 제공한 2017년 1분기 전 세계 스마트폰 판매량이다. 이에 따르면 스마트폰 OS 점유율은 안드로이드 84.1%, IOS 14.8%로 안드로이드가 크게 우위를 차지하는 것을 알 수 있다. [12] 이를 통해 안드로이드가 현재 가장 대중적인 모바일 운영체제가 되었음을 확인할 수 있다.

표 6.1 2017년 1분기 전 세계 스마트폰 운영체제별 최종 판매량

(단위 : 천대)

운영체제	2017년 1분기 판매량	2017년 1분기 시장 점유율(%)	2016년 1분기 판매량	2016년 1분기 시장 점유율(%)
안드로이드	327,163.6	86.1	292,746.9	84.1
IOS	51,992.5	13.7	51,629.5	14.8
기타 운영체제	821.2	0.2	3,847.8	1.1
총계	379,997.3	100.0	348,224.2	100.0

출처: 가트너

6.4. 스마트폰의 다양한 센서

본 논문의 시스템은 스마트 폰 카메라, GPS 등 스마트폰에 내장된 센서를 이용하여 개발하였다. 스마트폰에 내장된 다양한 센서를 파악하면 본 논문에서와 같이 다채로운 응용이 가능해진다. 스마트폰에 내장된 센서의 종류와 기능을 알아보기 위해 이를 조사하여 표 6.2에 기재해 놓았다.

표 6.2 스마트폰 센서

센서	기능
지문 인식	아이폰 5S에 처음 사용되기 시작하였다. 인간의 고유한 지문 패턴을 인식할 수 있다. 본인 인증에 주로 활용되고 있다.
홍채 인식	일본 후지쯔가 ARROWS NX F-04G 모델을 통해 최초로 탑재했으며 삼성전자에서도 이 기능을 활용하고 있다. 안경이나 렌즈를 착용해도 홍채를 정확히 인식할 수 있으며 스마트폰 잠금, 본인 인증에 다양하게 활용되고 있다.
심장박동	심장의 박동 속도를 측정해주는 기능으로 갤럭시 S5에 처음 사용되었으며 주로 피트니스 용도로 사용된다.
지자기 센서	지구의 자기장을 탐지해 내는 센서로 나침반 기능을 수행한다.
기압계	공기의 압력을 감지해 내며 이를 통해 고도를 측정할 수 있어 정확한 운동량을 체크할 수 있다.
온/습도 센서	단말 주변의 온도와 습도를 측정하는 데 사용한다. 현재 잘 활용되지 않는 센서이다.

모션 센서	여러 센서가 복합된 것으로 지자기, 가속 센서, 기압계 등을 이용하여 움직임 또는 위치를 찾아낼 때 사용한다. 화면을 보는 도중 화면이 꺼지지 않게 하는 스마트 스테이, 음악 재생 중 스마트폰을 흔들어 조작하는 것 등이 응용 사례이다.
홀 센서 (Hall Sensor)	자기장의 세기를 감지하여 스마트폰의 플립커버 닫힘 유무를 확인하는데 주로 사용된다.
밝기 센서	주변의 조도를 감지해 디스플레이의 밝기를 자동으로 조절할 때 사용한다.
RGB 센서	주변의 빛과 색의 농도를 검출하는 기능을 하며 디스플레이의 색을 보정한다.
근접 센서	어떠한 물체가 근접해 있는지 알려주는 센서로 스마트폰의 앞면에 위치해 있다. 통화 중 귀를 가까이 댈 경우 터치 오작동 방지를 위해 화면이 자동으로 꺼지는 기능, 스마트폰 커버의 닫힘 유무를 판별해 홀센서와 비슷한 기능으로 사용되기도 한다.
자이로 센서	X, Y, Z 좌표를 기준으로 움직이는 방향을 측정할 때 주로 사용한다. 스마트폰을 바닥에서 들어 올릴 때 화면이 실행되도록 하거나 스마트폰의 기울기를 이용해 조정하는 게임 등에도 활용된다.
가속 센서	X, Y, Z 좌표를 기준으로 이 좌표가 움직이는 속도를 측정하는 데 사용한다. 자이로 센서와 함께 사용되며 움직이는 물체나 스마트폰 자체의 이동속도를 측정하는 데 이용한다. 최근에는 헬스케어 기능에 들어가는 만보계에 주로 활용된다.
GPS 센서	GPS 위성을 이용해 현재의 위치와 시간을 측정할 수 있다. 다양한 위치 기반 서비스를 구현할 수 있으며 내비게이션 앱이 대표적인 활용 예이다.
중력 센서	X, Y, Z 좌표를 기준으로 중력 가속도를 측정하여 스마트폰이 가로 방향 또는 세로 방향으로 되어있는지 알려주는 기능을 한다.

출처: 유플러스 블로그 [13]

제7장 OpenCV

7.1. OpenCV

OpenCV (Open Source Computer Vision Library)는 Open source computer vision 및 Machine learning software library이다. 1999년 인텔에서 처음 제작되었다. OpenCV는 BSD 라이선스 제품이며 2,500개 이상의 Open source computer vision 및 Machine learning을 보유하고 있다. 이 알고리즘은 얼굴 감지, 물체 식별, 인간의 행동 추적, 카메라의 움직임 추적 등에 다양하게 사용된다.

Google, Yahoo, Microsoft, Intel, IBM, Sony, Honda, Toyota 등의 대기업에서 사용되며 신생 기업에서도 다용도로 이용되고 있다.

OpenCV는 주로 C++언어로 사용되고 있지만, C, Python, Java, MATLAB으로 사용이 가능하며 Windows, Linux, Android, Mac OS 환경에서 사용할 수 있다. [14]

7.2. OpenCV와 컴퓨터 비전의 응용 분야

1) 문자 인식

문자 인식은 매우 대중적으로 사용되고 있다. 자동차 번호를 인식하거나 카드번호 인식, 사진으로 촬영한 언어 인식 등을 통해 다양하게 실생활에 활용되고 있다.

2) 생체 인식

사진 촬영 시 얼굴 영역에 초점을 맞추거나 얼굴, 홍채, 지문, 손등의 핏줄 모양 등을 인식하여 인체 고유의 정보를 기억한 후 이를 통해 보안 시스템에 접근하는 방향으로 주로 적용되고 있다. 이는 각종 결제나 출입 제어 등에 활용된다.

3) 의료(醫療)

의료 장비들이 촬영한 영상을 컴퓨터로 분석하여 의사의 판단에 중요한 도움을 줄 수 있다. CT, MRI, X-ray, 성형 모의 시술 등에 다양하게 응용되고 있다.

4) 제조 공정 검사

공정 자동화 과정에서 공정 과정을 모니터링 하며 각 과정을 검사하는데 사용된다. 또한, 제품 부착 라벨의 부착 상태나 제조 제품의 불량 유무를 검사하는데 응용되기도 한다.

5) 지능형 자동차

현재 일부 고급 차량에는 차선 감지를 이용한 졸음방지, 자동 주행, 자동 주차, 앞차와의 간격 유지, 사물 감지를 통한 알림 기능 등을 제공한다. 이와 같이 OpenCV를 이용한 안전운행 및 편의성을 위한 개발이 활발히 진행되고 있다.

6) 게임

인간이 사용하는 제스처를 인식하여 제어하는 게임이 이미 시중에 판매되고 있다. 스포츠 분야의 게임에 주로 적용되고 있는데 이를 스포츠 선수의 역량 향상에 활용하기도 한다. [15]

제8장 결론

본 논문에서는 추가적인 장비 없이 안드로이드 스마트폰만을 이용하여 “자동차 운전자의 전방주시 태만 방지 시스템”과 “운전 중 통화방지 시스템”을 개발하고 이에 관한 실험결과를 제시하였으며, 마지막으로 안드로이드와 OpenCV의 동향에 대해 알아보았다.

2~3장에서는 기존의 눈 깜빡임 감지를 이용한 졸음감지 시스템의 고개 떨어짐 및 얼굴이 다른 방향으로 향했을 때 알림을 전혀 주지 못하는 단점을 보완해 전방주시 태만과 졸음운전을 동시에 검출해 낼 수 있는 “전방주시 태만 방지 시스템”을 구현해 보고 실험해 보았다.

스마트폰에서 구현되는 얼굴 전면인식 기술의 특성을 이용해 전방으로부터 상 45도, 하 30도, 좌 40도, 우 40도 이상 다른 곳을 바라볼 시 알림을 주어 운전 중 전방주시 태만을 방지할 수 있음을 본 연구를 통해 확인하였으며 눈감음 감지의 경우 평균 95%의 성공률을 보였다.

기준 각 이상으로 사방으로 고개를 돌릴 시 상 90.2%, 하 92.2%, 좌 91.2%, 우 91.2%의 눈 감김에 대한 인식률을 보여 90% 이상의 성공률을 확인하였다.

4~5장에서는 운전 중 전방주시 태만 및 전화 통화 방지를 위해 주행속도를 인식하여 수신된 전화를 무음으로 처리하고 이에 대해 자동으로 문자메시지를 보내주는 애플리케이션을 구현해 보고 실험하였다.

자동차 주행 중 걸려온 전화에 문자메시지를 정확히 보낼 확률은 평균 96.9%로 나타났으며 자동차 주행 속도의 인지 정확도는 약 83%로 나타났다.

본 논문의 “전방주시 태만 방지 시스템”과 “자동차 운전 중 통화 방지 시스템”이 실용화 된다면, 자동차 사고의 70%를 차지하는 전방주시 태만으로 인한 교통사고가 현저하게 줄어들 것이다.

참고문헌

[1] 조준한. (2012) “교통사고 안전 불이행 유형 및 원인분석 연구” TS 교통안전공단 108p

[2] 김건중, 신봉기. (2014) "OpenCV4 android와 스마트폰을 통한 졸음 운전 감지 시스템." 한국정보과학회 학술발표논문집

[3] TAAS 교통사고분석시스템. http://taas.koroad.or.kr

[4] 도로교통법 제49조 10항, 11항

[5] 고명수. (2012) "안전거리미확보 실태분석 및 교통사고 방지대책 연구." TS 교통안전공단, 6-8p

[6] 도로교통공단 전 CEO 정일영 블로그.

http://blog.naver.com/PostView.nhn?blogId=skylandciy&logNo=20144532815

[7] https://developer.android.com/reference/android/location/Location.html

[8] 장용식, 성낙현. "Step by Step 안드로이드 프로그래밍” 6p, 15p

[9] 안드로이드 공식. https://source.android.com/source/build-numbers

[10] 위키백과. https://ko.wikipedia.org/wiki/안드로이드_버전_역사

[11] 나무위키. https://namu.wiki/w/안드로이드(운영체제)/버전?=안드로이드%20버전

[12] Gartner. http://www.gartner.com

[13] 유플러스 블로그. http://blog.uplus.co.kr/1860

[14] OpenCV. http://opencv.org/about.html

[15] 최형일. “컴퓨터 비전” 2-11p