Journal of the Korea Academia-Industrial cooperation Society
Vol. 12, No. 6 pp. 2793-2800, 2011

DOI : 10.5762/KAIS.2011.12.6.2793

스마트폰용 차선이탈경보 애플리케이션 개발

노광현[1*]
[1]한성대학교 산업경영공학과

Development of a Lane Departure Warning Application on a Smartphone

Kwang-Hyun Ro[1*]
[1]**Department of Industrial & Management Engineering, Hansung University**

요 약 본 연구에서는 범용 이동정보통신기기인 스마트폰을 플랫폼으로 하는 차선이탈경보 애플리케이션을 개발하고 최적화하였다. 최근 안전주행지원 솔루션 중 하나인 차선이탈경보시스템이 상용화되고 있지만, 고성능의 전용 플랫폼을 필요로 하기 때문에 시장에 쉽게 진입하지 못하고 있다. 본 연구에서는 스마트폰인 iPhone 3GS를 플랫폼으로 하는 차선이탈경보 애플리케이션으로 개발하고 처리 속도를 최적화하였다. 효율적인 영상처리를 위해서 OpenCV를 사용하였고, 차선인식을 위해서는 차선의 기하학적 특징을 고려한 Hough Transform 기반의 휴리스틱 알고리즘이 고안되었다. 차선이탈경보 애플리케이션은 매킨토시 컴퓨터에서 Xcode 3.2.4 개발툴을 사용하여 개발되었고, 스마트폰에 다운로드하여 실제 도로에서 실험하였다. 실험결과 1.52fps의 처리 속도를 보였고, 최적화 작업을 통해 처리 속도를 3.84fps까지 향상시켰다. 향후 차선인식 알고리즘 보완, 추가 최적화 작업 및 고성능 스마트폰 플랫폼 채택 등을 통해 스마트폰용 차선이탈경보 애플리케이션을 상용화할 계획이다.

Abstract The purpose of this research is to develop and optimize a lane departure warning application based on a smartphone which can be applicable as a new platform for various mobile information applications. Recently, a lane detection warning system which is a representative application among safe driving assistant solutions is being commercialized. Due to the necessity of powerful embedded hardware platform and its price, its market is still not growing. In this research, it is proposed to develop and optimize a lane departure warning application on iPhone 3GS. OpenCV is used for efficient image processing, and for lane detection a heuristic algorithm based on Hough Transform is proposed. The application was developed under Macintosh PC platform with Xcode 3.2.4 development tools, downloaded to the iPhone and has been tested on the real paved road. The experimental result has shown that the detection ratio of the straight lane was over 90% and the processing speed was 1.52fps. For the enhancement of the speed, a few optimization methods were introduced and the fastest speed was 3.84fps. Through the improvement of lane detection algorithm, additional optimization works and the adoption of a new powerful platform, it will be successfully commercialized on smartphone application market.

Key Words : Safe Driving, Lane Departure Warning System, Smartphone, OpenCV, Hough Transform

1. 서론

지능형 자동차에 대한 연구는 2000년대 초반부터 국내외 유명 자동차메이커 및 연구기관에서 수행되어 왔으며, 최근 차선이탈경보시스템(LDWS, Lane Departure Warning System)을 비롯한 장애물경보시스템, 졸음운전경보시스템 등의 지능형 기능이 구현된 상용 제품들이 고급자동차에 before-market용으로 탑재되어 고가로 판매

본 연구는 한성대학교 교내연구비 지원과제임.
*교신저자 : 노광현(khrho@hansung.ac.kr)
접수일 11월 04월 04일 수정일 (1차 11년 05월 29일, 2차 11년 05월 31일) 게재확정일 11년 06월 09일

되고 있다. After-Market용으로는 차량용 블랙박스가 네비게이션 시장 수준으로 성장하고 있다. 이렇게 사고 발생시 원인을 분석할 수 있는 장치뿐만 아니라 운전자의 부주의로 발생할 수 있는 교통사고를 미연에 방지할 수 있는 지능형 장치에 대한 시장의 요구가 높아지고 있는 실정이다.

본 연구에서는 최근 음성통화 기능을 벗어나 다양한 분야에 적용되고 있는 스마트폰을 활용한 차선이탈경보 애플리케이션을 개발하였다. 차선이탈경보시스템은 몇 년전부터 before&after market용으로 개발되어 판매되고 있지만 고성능의 전용 임베디드 플랫폼 사용으로 인한 성능 대비 고가의 가격, 설치의 어려움 등의 이유로 시장이 활성화되지 못하고 있다. 본 연구에서는 이러한 문제점을 해결할 수 있는 방안으로 차선이탈경보장치의 플랫폼을 스마트폰으로 선정하였고, 차선인식 및 이탈경보를 애플리케이션으로 개발하여 일반 사용자들이 쉽고 편리하며 저렴하게 차선이탈경보기능을 사용할 수 있도록 하고자 하였다.

본 연구에서는 스마트폰 활성화의 중심에 있는 애플사의 아이폰 3GS를 플랫폼으로 선택하였다. 차선인식 알고리즘에 대해서는 이미 많은 연구가 진행되었다. 본 연구의 초점은 기존의 차선인식용 전용 장치에 비해 컴퓨팅 성능이 뒤지는 범용 이동정보통신기기인 스마트폰에서 차선인식 및 이탈경보를 할 수 있는 수준의 적절한 알고리즘을 선정하여 적용하는 것이다. 동시에 스마트폰용 다양한 어플리케이션이 개발되고 있지만 안전 주행을 위한 애플리케이션은 찾아보기 어렵기 때문에 이러한 시장에 진출할 수 있는 기반 기술을 마련하기 위한 것이다. 이와 유사한 연구와 관련하여 국내에서는 보고된 바 없고, 외국에서는 관련 연구와 함께 최근 시제품이 출시되고 있는 상황이다. 하지만 국내 도로 환경에 맞는 애플리케이션을 제공되지 못하고 있다.

제안하는 아이폰 기반의 차선이탈경보 애플리케이션은 아이폰의 외장 카메라에서 입력되는 전방 도로 영상에서 연속적으로 차선을 추출하고 차선이탈이 예상되는 경우 영상 표시와 경고음을 통해 운전자에게 경고 상황을 알려준다. 영상처리를 위해서는 인텔에서 제공하는 영상처리 라이브러리인 OpenCV를 사용하였고, 차선인식은 Hough Transform을 기반으로 차선의 기하학적 특징을 고려한 휴리스틱 알고리즘을 제안하였다.

본 논문은 2장에서는 연구에 사용된 플랫폼인 아이폰과 OpenCV에 대해 살펴보고, 3장에서는 제안된 차선인식알고리즘을 설명한다. 4장에서는 아이폰에 구현된 차선이탈경보 애플리케이션의 개발환경, 결과물 및 실험결과를 설명하고, 5장에서는 애플리케이션의 속도 향상을 위한 방법과 실험결과를 설명하고, 6장은 결론이다.

2. 모바일 LDWS 플랫폼

2.1 하드웨어 플랫폼: iPhone 3GS

PC 수준의 성능과 함께 멀티터치 스크린, GPS, WiFi, G-센서, CMOS 카메라 등을 제공하는 스마트폰의 등장으로 영상처리 응용분야 및 애플리케이션이 크게 확대되고 있다. 아이폰은 애플사의 데스크탑 PC인 iMAC보다는 사양이 월등히 뒤지지만, 아이폰 3G 모델이 출시된 이후로 아이폰 3GS가 출시되면서 사양이 높아졌다. 현재는 한층 향상된 사양의 아이폰4가 출시되어 애플리케이션의 한계점을 높여가고 있다. 아이폰의 사양 변화는 표 1과 같다[1].

[표 1] 아이폰 모델간 사양 비교
[Table 1] Comparison between iPhone Models

제조사	iPhone 3G	iPhone 3GS	iPhone 4G
CPU	412MHz	600MHz	1GHz
Memory	128MB	256MB	512MB
Screen Size	3.5inch	3.5inch	3.5inch
Resolution	320×480	320×480	640×960
Multi-tasking	x	○ (iOS4.0)	○

아이폰이 처음 출시되었을 당시에는 필요에 따라 하드웨어를 확장할 수 없고, 사용자들이 직접 개발한 애플리케이션을 추가로 설치할 수 없었다. 하지만 사용자들의 요청에 따라 아이폰 SDK가 공개되었고, 폐쇄적인 조건하에서 일반 사용자들도 애플리케이션을 쉽게 개발할 수 있게 되었다.

아이폰 애플리케이션 개발에는 Mac OS X 10.3버전 이상의 환경이 필요하다. 애플리케이션을 개발할 때는 개발툴(IDE: Integrated Deve-lopment Environment)인 Xcode내의 아이폰 시뮬레이터에서 테스트할 수 있지만, 카메라나 가속 센서 같이 실제 장치를 반드시 이용해야 하는 애플리케이션을 테스트할 경우에 아이폰 개발자 프로그램에 유료로 가입해야 하고, 이를 통해 개발한 애플리케이션을 실제 아이폰에서 구동할 수 있다.

초기 아이폰 모델은 멀티태스킹이 불가능하여 제약사항이 많았으나 iOS4.0버전이 출시된 이후부터 멀티태스킹이 가능해져 애플리케이션을 실행하면서 동시에 다른 작업이 가능해졌다. 하지만 가비지 컬렉션(Garbage Collection)을 제공하지 않아 메모리 점유율이 높은 애플

리케이션은 운영체제가 강제 종료시키는 등의 제약사항이 여전히 존재하고 있다. 정확한 기준은 공개되어있지 않지만, 22MB 정도가 적정한 애플리케이션의 가용 메모리이다. 이 크기를 초과하여도 애플리케이션을 실행되지만 아이폰의 전체적인 실행속도가 낮아지기 때문에 애플리케이션 개발 시 메모리 점유율에 대해 많은 고려를 해야 한다. 본 연구에 개발한 애플리케이션의 메모리 점유율은 약 18MB로 측정되었다.

아이폰 카메라에서 생성되는 이미지 프레임을 실시간으로 입력받는 함수는 그 동안 공개되지 않아 UIGetScreenImage라는 Private API를 무단 접근하여 사용하였지만, iOS 4.0부터는 AVCaptureDevice라는 이미지 실시간 처리 함수가 제공되어 실시간 영상처리나 증강현실 애플리케이션에 폭넓게 적용되고 있다.

LDWA(Lane Departure Warning App)이 탑재된 아이폰을 차량에 설치한 모습은 그림 1과 같다. 자동차 전면 유리창에 차량용 거치대를 설치하여 아이폰을 설치하고 USB 차량용 시거잭을 사용하여 지속적으로 전원을 공급할 수 있도록 하였다. 전방도로 영상을 촬영하는 아이폰 카메라가 거치대에 가려지지 않도록 설치하였다.

[그림 1] 차량에 아이폰을 설치한 모습
[Fig. 1] Installation of a iPhone on a Car

2.2 영상처리 라이브러리: OpenCV

OpenCV(Open Source Computer Vision Library)는 인텔에서 제공하는 영상처리 라이브러리이다. 컬러 공간 처리, 화소값 기반 처리, 에지 추출, 영상 분석, 카메라 교정, 기계 학습 알고리즘 등 사용하기 쉬운 라이브러리를 실시간 연산이 가능하도록 연산의 효율성을 최대한 고려하여 C와 C++로 작성된 함수로 제공한다.

OpenCV는 크게 다섯 개의 구성 요소로 나뉜다. CV는 기본적인 영상 처리와 고수준의 컴퓨터비전 알고리즘을 포함하며, MLL은 통계 분류와 군집화 도구 등의 기계 학습 라이브러리를 포함한다. HighGUI는 입출력에 관련된 함수들과 영상과 비디오를 불러오고 저장하는 기능을 포함하며, CXCORE는 기본적인 자료 구조와 알고리즘을 포함하고 있고, CVAux는 실험적인 알고리즘을 포함하고 있다.

OpenCV는 Windows, Linux, Mac OS X에서 사용이 가능하고, 아이폰에서도 하드웨어의 발전으로 OpenCV 라이브러리를 사용하여 컴퓨터비전 프로그램을 구현하기에 용이하다. 또한 공개된 라이브러리로서 상업적 목적으로 사용하기도 용이하다. 본 애플리케이션은 아이폰 카메라에서 입력 받은 영상을 제어하고, 차선인식을 위한 영상처리를 위해 HighGUI와 CV의 함수들을 사용하여 제작되었다. OpenCV의 이미지 구조체는 IplImage이고, 아이폰 SDK의 이미지 구조체는 UIimage인데, 아이폰 SDK는 아이폰 카메라로 UIImage를 입력받은 뒤에 IplImage로 변환한 뒤에 이미지 처리 후 UIImage로 반환할 수 있게 해준다.

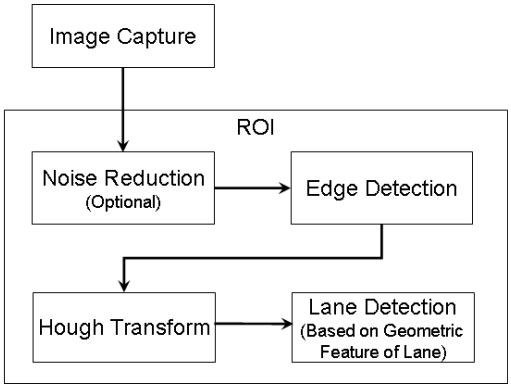
3. 차선인식알고리즘

본 장에서는 아이폰 3GS에 탑재하여 구동될 차선인식 알고리즘에 대해 설명한다. 차선인식에 대한 연구는 세계적으로 ITS가 화두가 되기 시작한 2000년대 초반부터 활발히 시작되었다. 현재까지 다양한 알고리즘이 개발되었고, 최근 일부 고급 차량에 상용화된 제품이 탑재되고 있다.

현재까지 연구된 차선인식방법은 크게 특징 기반 방법과 모델 기반 방법으로 분류할 수 있다[3-11]. 특징 기반 방법은 차선 에지 등과 같이 차선 특징을 사용하여 차선을 인식하는 방법[3-5]으로 차선 표식의 선명도에 따라 차선 인식률에 차이가 발생할 수 있다. 즉, 차선 표식이 희미하거나 영상 잡음 등이 포함되어 있는 경우 차선 인식이 어려울 수 있다. 모델 기반 방법은 차선을 기하학적 변수로 표현되는 곡선으로 표현한다[6-11]. 이 방법은 특징 기반 방법에 비해 차선 표식의 선명도나 영상 잡음에 따른 인식률에 덜 민감하다. 하지만 일반적으로 복잡한 모델링 처리 과정이 요구되고, 서로 다른 유형의 도로 영상에 대해 공통적으로 적용하기 쉽지 않다.

아이폰과 같은 컴퓨팅 성능이 제한적인 스마트폰 기반 플랫폼에서 실행되는 차선인식 애플리케이션을 개발하기 위해서는 알고리즘이 단순하고 효율적이어야 한다. 본 연구에서 모델 기반 방법 중 계산량을 최소화하기 위해 차선을 선형 모델(linear model)로 제한하였고, 이 모델의 파라메터 결정을 위해 Hough Transform을 적용하였다. 계산량을 최소화하기 위해서 차선인식 절차는 입력된 전

방 도로 영상에서 차선이 존재하는 영역(ROI, Region of Interest)에 대해서만 수행하며 그림 2와 같다.



[그림 2] 차선 인식 절차도
[Fig. 2] Flow chart of lane detection

3.1 ROI 설정

일반적인 영상처리 알고리즘에서는 입력 영상에 포함된 노이즈를 제거 하는 등의 목적으로 전처리 과정을 수행한다. 아이폰용 차선인식 알고리즘에서도 전처리 과정을 수행한다. 하지만 전처리 과정은 영상 전체 혹은 일부 영역에 대해 특정 연산처리를 수행해야 하므로 많은 계산량을 필요로 한다. 일반 PC 개발 환경에서도 이 과정에 많은 시간이 할당되는데 아이폰의 하드웨어 플랫폼의 특성을 고려한다면 전처리 영역을 최소화하는 것이 바람직하다. 따라서 본 연구에서는 전방 도로 영상 중 차선이 존재하는 영역을 ROI로 설정하고 모든 차선인식 절차에서 이 영역의 데이터만을 처리하도록 하였다. 아이폰 카메라의 입력 영상 해상도는 320×480이고, 그림 1과 같이 전방 도로를 향하도록 아이폰을 설치하면 그림 3과 같은 영상을 입력받게 된다. 이 영상에서 ROI 설정은 사용자가 수동으로 할 수 있으나, 본 연구에서는 기본값으로 영상을 가로로 3등분하는 경우 하단의 2/3 영역을 ROI로 설정하였다. 즉, 도로 차선을 확장하면 만나게 되는 지점의 y좌표가 세로 1/3지점이 되도록 하는 것이다. 본 애플리케이션을 상용하는 단계에서는 아이폰 설치시 이에 대한 가이드라인을 사용자에게 제공할 것이다. ROI 설정시 주의할 점은 도로면이 균일하지 않아 차량의 진동과 거치대의 흔들림으로 카메라의 tilt 각도가 지표면에 대해 일정하지 않으므로 ROI 영역을 여유 있게 설정하여야 한다.

3.2 전처리 과정(Pre-Processing)

차선인식에서는 조명이나 그림자 등의 다양한 환경에 의해서 카메라로 받은 영상에 잡음이 생길 수 있다. 잡음이 생기면 차선인식 결과의 신뢰성 저하가 발생되기 때문에 영상 보정 작업을 하는 전처리 과정이 필요하다. 잡음 제거에는 Median Filter를 적용하였고, 에지 추출을 위해서는 Canny Edge Operator를 사용하였다. 잡음 제거 과정의 경우 실제 실험시 차선인식에 큰 영향을 주지 않아 애플리케이션의 처리 속도 향상을 위해 실제 구현에서는 선택적으로 포함시키도록 하였다.

3.3 Hough Transform

Hough Transform은 영상에서 직선, 원, 또는 다른 간단한 모양을 찾는 방법이며, OpenCV에서는 Standard Hough Transform과 Progressive Probabilistic Hough Transform을 제공한다. 본 연구에 사용된 알고리즘에 쓰인 Hough Transform방식은 Standard Hough Transform방식이다. 이 방법은 일부 차선인식 알고리즘에 사용되고 있는데, 연산속도가 오래 걸리는 단점이 있다.

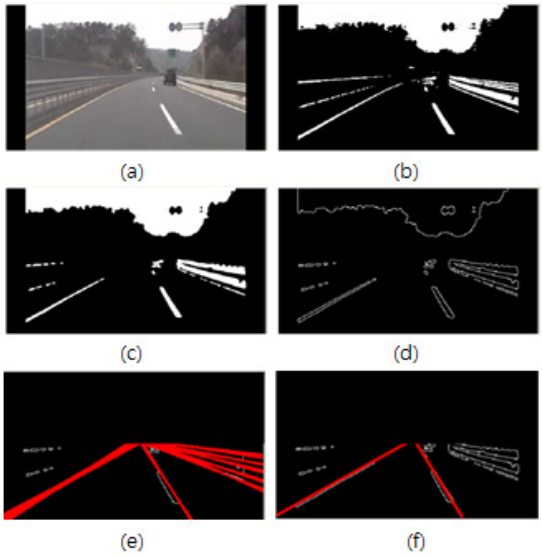
본 연구에서는 ROI에서 추출된 에지에 대해서만 Hough Transform을 적용하였고, 전방 도로 차선이 직선이라는 가정하에 차선 후보를 추출하였다. 또한 영상내 차선 후보 직선이 되기 위해서는 해당 직선에 포함되는 에지가 임계값 이상 포함되어야 차선 후보로 선정하였다. 즉 소수의 에지로 인해 추출된 직선은 차선일 가능성이 낮으므로 차선 후보에서 제외시킨 것이다.

3.4 기하학적 특징 기반의 차선인식

Hough Transform을 이용해 추출한 직선들 중에서 차선의 기하학적 특징을 이용하여 차선을 추출하였다. 차선의 대표적인 기하학적 특징은 다음과 같다.

- 차선은 공통된 소실점(vanishing point)에서 만난다. 따라서 여러 차선 후보 직선들이 만나는 점 중 가장 많은 직선들이 교차하는 영역을 소실점으로 설정하고 이 영역을 지나는 직선을 차선 후보로 결정한다.
- 아이폰 카메라의 경우 줌 기능이 없으므로 입력 영상에서의 차선폭은 일정한 가로 픽셀 수 범위로 표현된다. 따라서 후보 직선의 한 쌍에 대한 차선 탐지 결정시 이 조건을 활용한다.
- 정상적인 도로 주행시 차선을 직선으로 가정하는 경우 양쪽 차선의 기울기는 정해진 범위내에 존재한다. 따라서 이 범위를 벗어나는 차선 후보는 제외한다.
- 한번 차선이 인식된 이후에는 다음 영상에서의 차선의 위치가 급격하게 변하기 어렵다. 따라서 이전 차선의 위치와 현재 차선 후보의 위치의 위치 변동량을 계산하여 정확한 차선 탐지한다.

이외에도 추가적인 조건을 활용하여 차선을 추출하였고, 차선인식 단계별 실험 결과는 그림 3과 같다. 이 그림에서는 이해를 돕고자 이진영상을 추출 결과를 포함시켰고, ROI도 Hough Transform 단계부터 적용하였다.



[그림 3] 차선 인식 절차 예 (a) 입력 영상 (b) 이진 연상 (c) 미디안 필터 적용 (d) 캐니 에지 탐지 영상 (e) ROI/Hough 변환 (f) 차선 탐지
[Fig. 3] An example of lane detection with intermediate results. (a) Input Image (b) Binary Image (c) Median Filter (d) Canny Edge Detection (e) ROI/Hough Transform (f) Lane Detection

4. 실험 결과

4.1 실험 환경

본 연구는 OpenCV 라이브러리를 이용하여 Windows XP 환경에서 Visual Studio 6.0을 이용하여 알고리즘을 구현한 뒤, Mac OS X Snow Leopard 환경에서도 Xcode 3.2.4 개발툴을 이용하여 구현하였다. 이후 아이폰 환경에 맞게 마이그레이션하여 LDWS 애플리케이션을 개발하였으며, 각 개발 환경 사양은 표 2와 같다.

[표 2] 이종 플랫폼간 사양 비교
[Table 2] Comparison between multiple platforms

제조사	Windows	Mac OS X	iPhone 3GS
CPU	3.4GHz	3.06GHz	600MHz
Memory	1GB	4GB	256MB

Windows 환경과 Mac OS X 환경에서 구축시 아이폰 환경에 맞게 320×480의 해상도를 고려하였으며, CCD 카메라를 설치하여 실제 주행을 하며 알고리즘을 테스트하였다. 그림 4는 아이폰 시뮬레이터에서의 LDWS 실행결과이다.

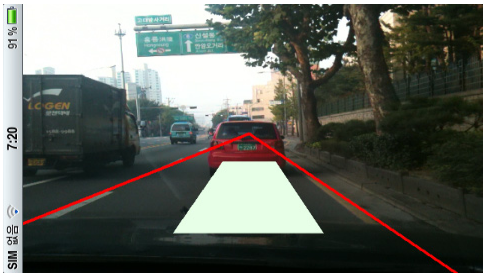
[그림 4] 아이폰 시뮬레이터상의 LDWS
[Fig. 4] LDWS on a iPhone Simulator

위의 과정을 통해 개발된 애플리케이션을 아이폰 3GS에 탑재한 결과와 설명서는 그림 5와 같다. LDWSTest와 LDWSMovie라는 2개의 애플리케이션을 개발하였다.

LDWSTest를 실행하면 그림 6의 화면이 나타나고, 아이콘을 클릭하면 그림 7과 같이 실행된다.

[그림 5] (좌) LDWS 아이콘 (우) LDWS 설명서
[Fig. 5] (Left) LDWS icon (right) Description of LDWS

[그림 6] LDWS 초기 화면
[Fig. 6] Initial screen of LDWS App

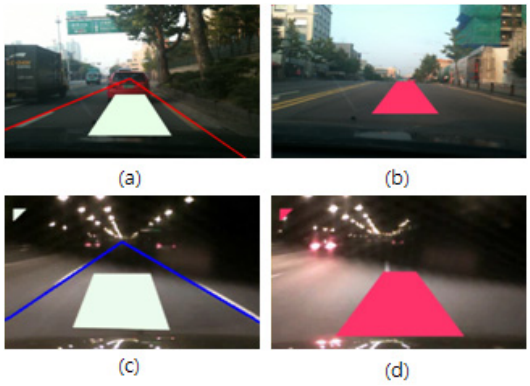


[그림 7] LDWS 실행 화면
[Fig. 7] Execution screen of LDWS App

4.2 실험 결과

차선인식 알고리즘 및 애플리케이션의 성능을 측정하기 위해 도로주행은 직선 주행이 긴 도로를 고려하여 내부순환도로와 자유로 방면을 택했다. 시간대는 주간과 야간으로 구분하여 실험하였다.

처리 속도 측면에서는 Window XP 기반의 노트북을 활용하는 경우 20fps 이상의 처리 속도를 얻을 수 있었지만, 아이폰에서 실행하는 경우에는 1.52fps 이하의 처리 속도를 보였다. 차선 인식률 측면에서는 연속된 직선 도로에서 90% 이상의 차선 인식 결과를 보였으나 곡선 도로에서는 인식률이 크게 떨어졌다. 현재의 차선인식 알고리즘은 직선 차선을 추출하도록 고안되었으므로 곡률이 큰 도로에서는 차선인식 오류가 발생하였다.



[그림 8] 아이폰에 탑재된 LDWS 애플리케이션 실행 결과 (a)주간 정상 주행시 (b) 주간 차선이탈시 (c) 야간 정상 주행시 (d) 야간 차선이탈시
[Fig. 8] Experimental results of LDWS Application on iPhone (a) normal driving at day (b) lane departure at day (c) normal driving at night (d) lane departure at night

그림 8은 주야간 차선인식 결과를 나타내고 있다. 그림(a) 화면내의 사다리꼴 사각형은 차선이 정상적으로 인식되었음을 나타내고, 차선이탈이 예상되는 경우에는 그림(b)와 같이 사다리꼴 사각형이 빨간색으로 변경되며, 이와 함께 사운드로도 경고하고 있다.

PC 환경과 아이폰 환경에서의 처리속도 차이는 존재하지만, 스마트폰 기반에서의 차선이탈경보 애플리케이션의 적용 가능성을 충분히 확인할 수 있었다. 다만 아이폰에서의 차선 인식 처리 속도가 평균 1.52fps로 실시간 처리가 필요한 애플리케이션에 부족하였다.

5. 처리 속도 향상 실험

앞 장에서의 실험 결과 처리 속도로는 실시간 영상처리가 필요한 애플리케이션으로 상용화하기 쉽지 않기 때문에 처리 속도 향상이 필요하였고, 이를 위해 다음의 방법을 적용하였다.

5.1 double형을 float형으로 변환

아이폰 3GS의 CPU는 부동 소수점 연산을 효율적으로 처리하는 장치인 부동소수점 장치 FPU(Floating Point Unit)를 자체 지원하기 때문에 정수인 int형과 실수형 float과의 성능 차이는 크게 나지 않는다. 하지만 float형과 double의 두 실수형은 FPU에서 자동으로 라운딩하여 크기를 변경해주기는 하지만 double형은 float보다 메모리를 2배 사용하기 때문에 속도에 영향을 줄 수 있다. 따라서double형으로 선언된 변수들을 float으로 변환하여 메모리 점유율을 줄이고, 속도 향상도 시도하였다.

5.2 cvSetImageROI 사용

차선의 추출에 사용된 Hough Transform 함수는 연산 속도가 오래 걸리는 단점이 있기 때문에 관심영역만 연산할 수 있는 OpenCV에서 지원하는 ROI 관련 함수를 사용하였다. 아이폰 3GS를 차량용 거치대에 설치하게 되면, 영상의 상단 지점에는 도로 영상이 아닌 영상이 잡히기 때문에 ROI 함수의 사용하여 영상의 상단 지점은 연산에서 제외하였다.

5.3 CGColorSpaceCreatDeviceGray() 사용

아이폰 3GS의 카메라에서 실시간으로 받아오는 영상은 UIImage 구조체에 저장되게 된다[2]. 하지만 OpenCV는 IplImage 이미지 구조체를 사용하기 때문에 OpenCV 함수를 사용하기 위해서는 아이폰SDK에 정의된 UIImage 구조체를 IplImage 구조체로 변형시켜야 한다. 변형되는 과정에서 영상의 색상공간이 전달되게 되는데 영상의 RGB값을 전달하는 CGColorSpaceCreateDeviceRGB() 함

수를 CGColorSpaceCreateDeviceGray() 함수로 변경하면, 속도향상을 가져올 수 있다. 아이폰 환경에서 OpenCV 함수를 사용하는 것은 외부 라이브러리에 접근하는 것이기 때문에 부담이 되는 작업이다. 또한 차선을 추출하는 전처리 과정에서 이미지를 바이너리 영상으로 변환해야 하는데, 이를 OpenCV 함수를 사용하여 변환하는 것보다 최초의 영상을 받아올 때, 아이폰 SDK의 메소드를 호출하여 바이너리 영상으로 받아오게 되면 부담을 크게 줄일 수 있게 되어 속도가 향상되게 된다.

5.4 이미지 크기 조절

아이폰 3GS에 탑재는 카메라는 320×480 해상도를 지원하는데 160×240로 축소시킨 영상을 대상으로 처리하면 속도를 향상시킬 수 있다. 이미지의 해상도를 줄이게 되면 차선추출의 오차율에 영향을 줄 수 있으나 도로영상은 전후 프레임의 영상변화가 크지 않은 특성 때문에 해상도를 줄여도 차선인식 오차율은 크게 변하지 않았다.

5.5 처리 속도 향상 결과

5.1에서 5.4까지 제시한 방법을 통해 최적화하기 전과 후의 처리 속도의 개선 정도는 표 3,4와 같다. 표 3은 320×480 해상도 입력 영상에 대해 double형을 float형으로의 변환과 CGColorSpace CreateDeviceGray() 함수로 변경, 관심영역 연산 수행의 수행 결과이다. 3가지 방법을 적용한 결과 한 프레임당 0.22초가 단축되어 기존의 1.52fps에서 2.27fps로 초당 프레임 속도가 향상되었다. 표 4는 입력 영상의 해상도를 160×240로 축소한 후 표 3과 같이 3가지 방법을 적용한 결과이다. 한 프레임당 0.26초의 속도가 단축되어 기존의 1.52fps에서 3.84fps로 초당 프레임 속도가 향상되었다. 처리 속도 측정은 실시간 이미지 정보가 담긴 UIImage구조체가 생성되는 시점부터 Hough Transform 연산을 이용하여 차선을 추출하고, cvLine 함수를 이용하여 차선을 화면에 나타내는데 걸리는 시간을 기준으로 하였으며 시간값은 NSLog를 이용한 디버깅을 통하여 계산하였다.

[표 3] 처리 속도 향상 결과 (해상도: 320 × 480)
[Table 3] Performance result of processing speed (image resolution: 320 × 480)

최적화 방법	Time
double형 → float 형	-0.06s
전체영역 연산 → 관심영역 연산	-0.14s
CGColorSpaceCreateDeviceRGB() → CGColorSpaceCreateDeviceGray()	-0.02s
한 프레임 처리 속도	0.44s
속도 향상 결과	0.22s 개선

[표 4] 처리 속도 향상 결과 (해상도: 160 × 240)
[Table 4] Performance result of processing speed (image resolution: 160 × 240)

최적화 방법	Time
double형 → float 형	-0.04s
전체영역 연산 → 관심영역 연산	-0.01s
CGColorSpaceCreateDeviceRGB() → CGColorSpaceCreateDeviceGray()	-0.13s
한 프레임 처리 속도	0.26s
속도 향상 결과	0.40s 개선

위 결과와 같이 축소된 해상도에서의 차선인식 오차율에 문제가 없다면 처리 속도 향상을 위해 160×240 해상도를 적용할 계획이다. 이외에도 OpenCV 헤더파일의 수정, 함수 호출 변환 등의 과정을 거친 뒤의 결과를 초당 프레임 수로 환산하면 기존의 처리 속도인 1.52fps (0.66s)에서 최소 2.27fps(0.44s)에서 최대 3.84fps(0.26s)로 향상된 것을 확인되었다.

5. 결론

본 연구는 아이폰 기반의 차선이탈경보 애플리케이션 개발 및 속도 향상을 연구하였고, 제안된 방법을 통해 차선인식 결과 및 속도 개선을 확인하였다. 실험결과 애플리케이션의 처리속도는 최적화 작업을 거쳐 최대 3.84fps의 속도를 보였고, 직선 차선에서는 90%이상의 인식률을 나타내었다.

위의 실험 결과는 상용화를 위해서는 만족스러운 결과는 아니지만 스마트폰의 하드웨어 사양이 빠른 속도로 향상되고 있으므로 실시간 처리에 대한 점은 크게 문제되지 않을 것으로 예상된다. 향후에는 직선 차선 인식률 향상 및 곡선 차선인식을 위한 알고리즘을 보안할 것이며, 처리 속도 향상에 대해서도 추가적인 연구를 수행할 계획이다. 또한 지속적인 연구를 통해 스마트폰이 지능형 자동차를 위한 안전주행지원 솔루션의 플랫폼이 될 수 있도록 할 것이다.

References

[1] http://www.apple.com/iphone/specs.html
[2] iOS Reference Library, http://developer.apple.com/library/ios/navagation/index.html
[3] A. Broggi and S. Berte, “Vision-based Road Detection

in Automotive Systems: a Real-time Expectation-driven Approach", Journal of Artifi cial Intelligence Research, vol.3, pp. 325-348, 1995.

[4] M. Bertozzi and A. Broggi, "GOLD: A Parallel Real-time Stereo Vision System for Generic Obstacle and Lane Detection", IEEE Transactions of Image Processing, pp. 62-81, 1998.

[5] S.G. Jeong, C.S. Kim, K.S. Yoon, J.N. Lee, J.I. Bae, and M.H. Lee, "Real-time Lane Detection for Autonomous Navigation", IEEE Proc. Intelligent Transportation Systems (ITSC01), pp. 508-513, 2001.

[6] Yue Wang, Eam Khwang Teoh and Dinggang Shen, "Lane Detection and Tracking Using B-snake,"Image and Vision Computing, vol. 22, pp. 269-280, 2004.

[7] Jung, C. R. and C. R. Kelber, "A Lane Departure Warning System Using Lateral Offset with Uncalibrated Camera," Proc. IEEE Conf. on Intelligent Transportation Systems (ITSC05), pp. 102-107, 2005.

[8] Zu Kim, "Realtime lane tracking of curved local road", Proc. IEEE Conf. on Intelligent Trans portation System, pp. 1149-1155, 2006.

[9] Qing Lin, Youngjoon Han and Hernsoo Hahn, " Real-time lane detection based on extended edge-linking algorithm", proc. International Conference on Computer Research and Development, 2010.

[10] Feixiang Ren, Jinsheng Huang, Mutsuhiro Terauchi, Ruyi Jiang, Reninhard Klette "Lane Detection on the iPhone", Multimedia Imaging Report 43, 2009.

[11] Soo-Yeong Yi, Ji-Hyoung Ryu and Chang-Goo Lee, "Development of Embedded Lane Detection Image Processing Algorithm for Car Black Box", Vol.11, No.8, pp.2942-2950, Journal of The Korea Academia-Industrial cooperation Society, 2010.

노 광 현(Kwang-Hyun Ro) **[정회원]**

- 1995년 2월 : 고려대학교 산업공학과 (공학사)
- 1997년 2월 : 고려대학교 산업공학과 (공학석사)
- 2001년 8월 : 고려대학교 산업공학과 (공학박사)
- 2001년 10월 ~ 2002년 10월 : Ecole des Mines de Paris, Robotic Center (Post-Doc)
- 2003년 ~ 2006년 : 한국전자통신연구원 연구원
- 2006년 ~ 2007년 : 한국항공우주연구원 선임연구원
- 2007년 9월 ~ 현재 : 한성대학교 산업경영공학과 조교수

<관심분야>
차세대 이동통신, RFID/USN, ITS