논문 2022-17-06

# 경량화된 임베디드 시스템에서 역 원근 변환 및 머신 러닝 기반 차선 검출

## (Lane Detection Based on Inverse Perspective Transformation and Machine Learning in Lightweight Embedded System)

홍 성 훈, 박 대 진*
(Sunghoon Hong, Daejin Park)

Abstract : This paper proposes a novel lane detection algorithm based on inverse perspective transformation and machine learning in lightweight embedded system. The inverse perspective transformation method is presented for obtaining a bird's-eye view of the scene from a perspective image to remove perspective effects. This method requires only the internal and external parameters of the camera without a homography matrix with 8 degrees of freedom (DoF) that maps the points in one image to the corresponding points in the other image. To improve the accuracy and speed of lane detection in complex road environments, machine learning algorithm that has passed the first classifier is used. Before using machine learning, we apply a meaningful first classifier to the lane detection to improve the detection speed. The first classifier is applied in the bird's-eye view image to determine lane regions. A lane region passed the first classifier is detected more accurately through machine learning. The system has been tested through the driving video of the vehicle in embedded system. The experimental results show that the proposed method works well in various road environments and meet the real-time requirements. As a result, its lane detection speed is about 3.85 times faster than edge-based lane detection, and its detection accuracy is better than edge-based lane detection.

Keywords : Advanced driver assistance systems, Inverse perspective transformation, Lane detection, Machine learning

## Ⅰ. 서론 및 관련 연구

첨단 운전자 보조 시스템 (ADAS)은 운전자가 차량을 주행하는데 도움을 주는 시스템으로 운전 피로를 감소시키고, 안전한 운전을 도와준다. 그 중에 차선 이탈 경고 시스템 (Lane departure warning system, LDWS)은 자동차가 주행 중인 차로를 벗어났을 때 운전자에게 경고를 주는 시스템으로, 최근에는 차선 이탈 복귀 시스템 (Lane keeping system, LKS) 및 차선 유지 보조 시스템 (Lane keeping assistance system, LKAS) 기능으로 확대되고 있다 [1]. 이러한 시스템들은 차선 검출 알고리즘의 높은 정확도가 요구된다. 다양하고 복잡한 도로 환경에서 실시간으로 차선을 검출하고 유지하는 것은 항상 큰 도전 과제였으며 차선 검출 분야의 연구자들에게 많은 관심을 받고 있다.

기존의 차선 검출 방법은 에지 (Edge) 정보를 기반으로 차선 검출하는 연구가 활발했다. Low et al. [2]는 캐니 연산자 (Canny operators)를 사용하여 차선의 가장자리를 검출하고 허프 변환 (Hough transform)을 사용하여 최상의 선들을 찾는다. Bounini et al. [3]은 캐니 에지 검출기, 최소자승법 (Least square method), 예측을 위한 칼만 필터 (Kalman filter)를 결합하여 허프 변환을 기반으로 도로 경계 및 차선 검출 알고리즘을 제안했다. Ding et al. [4]는 소실점 위치 정보를 이용하여 도로 영역을 검출하고, 해당 영역에서 허프 변환을 이용하여 선분을 검출한다. 그러나 허프 변환을 기반으로 한 차선 검출 방식은 영상에 선이 많을 경우 차선의 오 검출이 상대적으로 높다는 단점이 있다 [5, 6].

본 논문에서는 도로에서 차선의 위치를 정확하게 파악하는 것을 목표로 하여 머신 러닝을 기반으로 한 차선 검출 알고리즘을 소개한다. 하르 (Haar) 특징을 기반으로 한 캐스케이드 분류기 (Cascade classifier)는 머신 러닝의 한 종류이며 사물 검출에 있어 효과적이다 [7, 8]. 차선 검출의 정확도를 높이기 위해 원본 영상을 조감도 (Bird's-eye view) 이미지로 변환하는 과정이 필요하다. 조감도 이미지는 실제 3차원 환경에서의 물체를 위에서 바라보는 것처럼 2차원 이미지에 투영하기 때문에, 지면으로부터 설치된 카메라의 높이와 각도가 일정하다면 차선에 대한 폭도 일정하다. 이러한 특성을 반영하여 관심 영역 (ROI)은 차선 검출에 적합한 고정된 크기에서만 고려하는 것이 가능하며, 그 외의 크기는 고려하지 않아도 되기에 효율적이다. 머신 러닝을 적용하기에 앞서 조감도 이미지에서 차선을 대략적으로 빠르게

*Corresponding Author (boltanut@knu.ac.kr)
Received: Oct. 9, 2021, Revised: Nov. 18, 2021, Accepted: Nov. 29, 2021.
S.H. Hong: Carnavicom (Senior Engineer)
S.H. Hong: Kyungpook National University (Ph.D. Student)
D.J. Park: Kyungpook National University (Assoc. Prof.)
※ 이 연구는 2021년도 산업통상자원부 및 산업기술평가관리원(KEIT) 연구비 지원에 의한 연구임 (20014592).


ISSN : 1975-5066
http://dx.doi.org/10.14372/IEMEK.2022.17.1.41

검출하기 위해 임의로 만든 1차 분류기를 적용한다. 머신 러닝에서 관심 영역에 대한 크기는 고려하지 않아도 되기 때문에 기존의 캐스케이드 분류기 보다 검출 속도가 빠르며, 에지 기반으로 차선 검출하는 방법보다 검출 속도 및 정확도가 향상되었다.

본 논문의 구성은 다음과 같다. 2장에서는 에지와 제안하는 머신 러닝 기반 차선 검출의 구조를 소개하고, 3장에서는 실험을 통해 제안한 알고리즘의 성능이 향상되었음을 입증하며, 마지막으로 4장에서는 결론을 맺는다.

## Ⅱ. 머신 러닝 기반 차선 검출

LDWS의 최종 목표는 주행 중인 차로의 차선을 인식하여 현재 차량이 의도하지 않게 차선을 벗어나는 경우 운전자에게 차선이탈 경보를 알려주는 것이다. LDWS의 성능은 차선 검출의 정확도에 영향을 받으며 차선을 검출하기 위한 시스템은 그림 1과 같다.

입력 영상으로부터 조감도 이미지로 변환하고 차선 검출을 위해 그레이스케일로 변환한다. 변환된 영상으로부터 차선을 탐색하기 위해 슬라이딩 윈도우 접근법을 사용하는데, 고정된 20×20 크기의 관심 영역을 사용한다. 탐색 시 먼저 1차 분류기를 적용하여 차선에 대한 확률을 계산한다. 계산된 확률이 특정 임계값 이하이면 다음 관심 영역을 탐색하고, 초과이면 캐스케이드 분류기를 적용하여 검사한다.

캐스케이드의 모든 단계를 통과하면 해당 관심 영역을 후보 영역으로 등록하고, 그렇지 않으면 다음 관심 영역을 탐색한다. 모든 관심 영역에 대한 탐색이 끝나면 사전에 검출된 후보 영역들의 개수를 확인하여, 2개보다 많으면 동일한 차선인지 아닌지에 대한 판별을 위해 그룹화 작업을 한다. 그룹화된 후보 영역들을 통해 왼쪽, 오른쪽 차선에 대한 후보 영역들을 추출하고, 가장 적합한 직선을 찾는 직선 적합 (Line fitting) 작업을 통해 원본 이미지로 차선을 시각화한다.

### 1. 조감도 이미지 변환

일반적으로 카메라 센서로 투영되는 이미지는 원근감 효과를 가지기 때문에 실제로는 차선이 평행하지만 그림 2와 같이 이미지에서는 차선이 평행하지 않은 선으로 나타난다. 이러한 원근감 효과 때문에 운전자가 거리를 제대로 느낄 수 없고 영상 처리나 분석도 어려워진다. 역 원근 변환 (Inverse perspective transformation)은 원근 효과를 제거하고 실제 상황과 일치하는 평면도 이미지를 생성하여 2차원 이미지에 3차원 물체를 표시하는데 사용한다.

조감도는 원본 이미지에서 역 원근 변환을 사용하여 생성한 이미지로 위에서 바라보는 효과를 가진다. 조감도 이미지는 실제 평행한 차선들을 표현할 수 있기에 차선 검출에 용이하다. 역 원근 변환을 위해서는 그림 3과 같이 핀홀 (Pinhole) 카메라 모델을 기반으로 계산한다.

핀홀 카메라 모델은 3차원 공간에서의 임의의 점이 이상적인 핀홀 카메라의 2차원 이미지 평면으로 투영될 때 수학

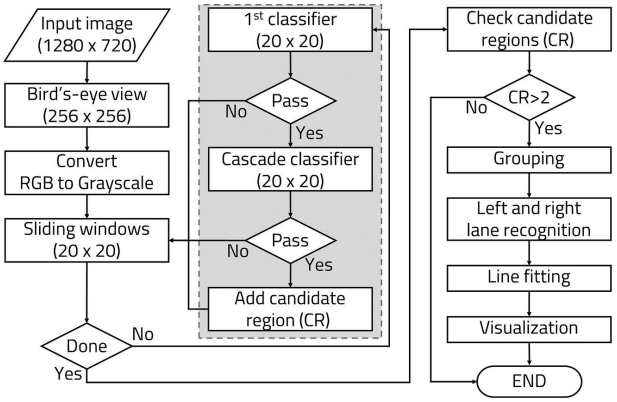



그림 1. 차선 검출 흐름도
Fig. 1. Lane detection flowchart

그림 2. 카메라 입력 이미지
Fig. 2. Camera input image

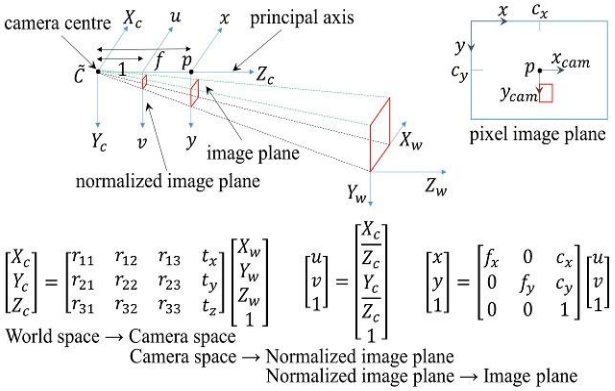



그림 3. 핀홀 카메라 모델
Fig. 3. Pinhole camera model

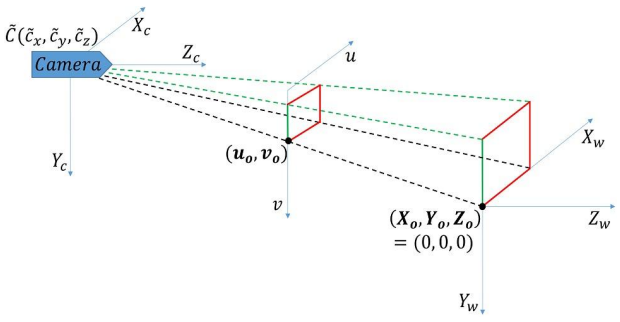



그림 4. 월드 공간에서 원점에 대한 원근 투영
Fig. 4. Perspective projection to the origin in world space

적인 관계를 표현한 것이다. 3차원 공간에서의 원점이 2차원 이미지로 투영되는 과정은 그림 4와 같다.

카메라는 오직 $\theta$ (Theta)에 대한 회전 성분만 있다고 가정하면, 월드 공간 (World space)에서의 원점이 정규화된 이미지 평면 (Normalized image plane)으로 투영되기 위한 좌표 계산은 식 1과 같다.

$$\begin{bmatrix} u_o \\ v_o \\ 1 \end{bmatrix} = s \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & \cos\theta & -\sin\theta & t_y \\ 0 & \sin\theta & \cos\theta & t_z \end{bmatrix} \begin{bmatrix} X_o \\ Y_o \\ Z_o \\ 1 \end{bmatrix}. \quad (1)$$

$\theta$는 카메라의 틸트 (Tilt) 각도이고 $s$는 월드 공간에서 정규화된 이미지 평면으로 투영하기 위한 스케일 팩터 (Scale factor)이다. 회전 행렬 ($R$)과 이동 벡터 ($T$)는 식 2와 같이 월드 공간에서의 한 점을 카메라 공간으로 이동하기 위한 변환 (Transformation)행렬이다.

$$P_c = R(P_w - \tilde{C}) = RP_w - R\tilde{C} = RP_w + T. \quad (2)$$

$\tilde{C}$는 월드 공간에서 카메라의 중심 좌표이다. 이동 벡터는 식 3, 4와 같이 계산된다.

$$T = -R\tilde{C}, \quad (3)$$

$$\begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} = - \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} \tilde{c_x} \\ \tilde{c_y} \\ \tilde{c_z} \end{bmatrix}. \quad (4)$$

정규화된 이미지 평면에서의 한 점은 식 5와 같이 정의된다.

$$\begin{bmatrix} u_o \\ v_o \\ 1 \end{bmatrix} = s \begin{bmatrix} X_o - \tilde{c_x} \\ Y_o\cos\theta - Z_o\sin\theta - (\tilde{c_y}\cos\theta - \tilde{c_z}\sin\theta) \\ Y_o\sin\theta - (\tilde{c_y}\sin\theta + \tilde{c_z}\cos\theta) \end{bmatrix}. \quad (5)$$

$Y_o$는 월드 공간에서 $Y_w$축에 대한 원점일 때 $v_o$는 식 6과 같이 정의 된다.

$$v_o = \frac{-\tilde{c_y}\cos\theta + \tilde{c_z}\sin\theta}{-\tilde{c_y}\sin\theta - \tilde{c_z}\cos\theta}. \quad (6)$$

$\tilde{c_y}$는 차량에 설치된 카메라의 높이, $\tilde{c_z}$는 카메라와 3차원 물체 사이의 $Z_w$축에 대한 거리이며 식 7과 같이 표현할 수 있다.

$$\tilde{c_z} = \frac{f_y\cos\theta - (y_o - c_y)\sin\theta}{(y_o - c_y)\cos\theta + f_y\sin\theta}\tilde{c_y}. \quad (7)$$

$f_y$는 $y$축에 대한 카메라 초점 거리 (Focal length)이고, $c_y$는 $y$축에 대한 카메라의 주점 (Principal point)이다. $y_o$는 월드 공간에서의 원점이 이미지 평면으로 투영될 때 $y$축에 대한 좌표이다.

최종적으로 카메라와 역 원근 변환으로 계산된 3차원 좌표 사이의 거리를 나타내는 $\tilde{c_z}$는 입력 이미지에서 $y$축에 해당되는 픽셀 좌표에 의해 계산된다.

$X_o$는 월드 공간에서 $X_w$축에 대한 원점일 때 $u_o$는 식 8과 같이 정의 된다.

$$u_o = \frac{\tilde{c_x}}{\tilde{c_y}\sin\theta + \tilde{c_z}\cos\theta}. \quad (8)$$

$\tilde{c_x}$는 카메라와 3차원 물체 사이의 $X_w$축에 대한 거리이며 식 9와 같이 표현할 수 있다.

$$\tilde{c_x} = \frac{x_o - c_x}{f_x}(\tilde{c_y}sin\theta + \tilde{c_z}cos\theta). \quad (9)$$

$f_x$는 $x$축에 대한 카메라 초점 거리이고, $c_x$는 $x$축에 대한 카메라의 주점이다. $x_o$는 월드 공간에서의 원점이 이미지 평면으로 투영될 때 $x$축에 대한 좌표이다.

차선은 차량과 같은 지면상에 있기 때문에 카메라의 광학축이 지면과 이루는 각도 ($\theta$) 및 카메라가 설치된 지면으로부터의 높이 ($\tilde{c_y}$)를 알고 있다면 3차원 상의 각 점들로부터 카메라가 얼마나 떨어져 있는지를 나타내는 $\tilde{c_x}$ 및 $\tilde{c_z}$를 계산 할 수 있다.

일반적으로 조감도 이미지를 생성하기 위해 원본 이미지에서 역 원근 변환을 하는 경우, 조감도 이미지의 좌표 사이에 픽셀 값이 없는 경우가 발생한다. 256×256 크기를 가지는 조감도 이미지의 모든 좌표에 대해서 픽셀 값을 넣어주기 위해 그림 5와 같이 조감도 이미지에서 원근 변환하여 1280×720 크기를 가지는 원본 이미지로부터 계산된 좌표의 픽셀 값을 가져온다.

$Z_w$축은 3차원 물체의 전방에 대한 거리 값을 나타내며 $X_w$축은 우측에 대한 거리 값을 표현한다. 흰색 점은 두 개의 평행한 차선이 이미지에서 만나는 소실점 (Vanishing point)이며 카메라가 기울어진 각도는 식 10과 같이 계산된다.

$$\tan(\theta) = \frac{c_y - v_y}{f_y}. \quad (10)$$

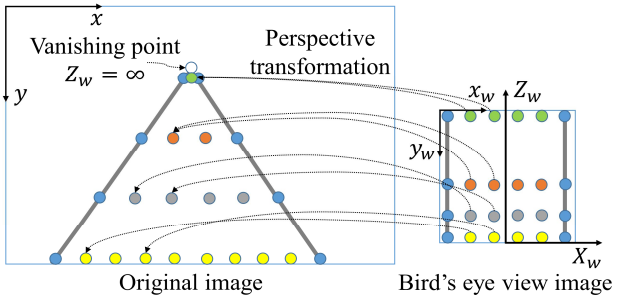



그림 5. 조감도 이미지 변환
Fig. 5. Bird's-eye view image transformation

$v_y$는 원본 이미지의 $y$축에 대한 소실점 좌표를 나타내며 전방에 대한 거리를 소실점으로 표현하면 식 11과 같다.

$$\tilde{c_z} = \frac{f_y^2 - (y_o - c_y)(c_y - v_y)}{f_y(y_o - v_y)} \tilde{c_y}. \quad (11)$$

$y_o$가 $y$축에 대한 소실점 좌표와 같아지면 분모가 0으로 수렴되어 전방에 대한 거리가 무한대로 계산된다. 거리가 늘어나면 $x$축에 대한 분해능이 줄어들기에 조감도 이미지의 녹색 점과 주황색 점은 각각 원본 이미지의 동일한 녹색 점 및 주황색 점에 대한 좌표로 계산된다. $\tilde{c}_z$는 $y_o - v_y$에 반비례하기 때문에 원본 이미지에서는 $y$축에 대해서 일정한 간격의 점들이 있더라도 조감도 이미지에서는 $Z_w$축에 대한 거리의 증가율이 점점 커진다.

조감도 이미지에서 원근 변환하기 전에 식 12, 13과 같이 3차원에 대한 $X_w$, $Z_w$축으로 변환한다.

$$X_w = (x_w - \frac{img_{width}}{2}) \times ResolutionX, \quad (12)$$

$$Z_w = (img_{height} + Z_{offset} - y_w) \times ResolutionZ. \quad (13)$$

$x_w$ 및 $y_w$는 조감도 이미지에 대한 좌표이며 $img_{width}$, $img_{height}$는 조감도 이미지의 가로, 세로 크기이다. $ResolutionX$, $ResolutionZ$는 한 픽셀 당 거리를 나타내는 역 원근 변환에 대한 해상도이며 $Z_{offset}$는 $Z_w$축의 역 원근 변환에 대한 오프셋이다. 식 14, 15는 조감도 이미지 좌표에서 원근 변환하여 계산된 원본 이미지의 좌표 $(x, y)$이며 해당되는 좌표에 대한 픽셀 값을 가져온다.

$$x = \frac{X_w}{\tilde{c_y} sin(\theta) + Z_w \cos(\theta)} f_x + c_x, \quad (14)$$

$$y = \frac{\tilde{c_y} cos\theta - Z_w \sin\theta}{\tilde{c_y} sin\theta + Z_w \cos\theta} f_y + c_y. \quad (15)$$

호모그래피 (Homography) 기법을 통한 원근 변환과 달리 카메라의 내부 파라미터, 설치된 높이 및 각도 정보를 사용하여 원근 변환하기 때문에 파라미터에 대한 자유도 (Degrees of freedom)가 8 DoF에서 6 DoF로 줄어드는 장점이 있다.

### 2. 에지 기반 차선 검출

기존에 차선을 검출하기 위한 몇 가지 방법 중 하나는 이미지에서 에지 정보를 사용하여 차선을 검출한다. 에지 기반 차선 검출 알고리즘은 별도의 전처리 작업이 필요하다. 먼저 차선과 비슷한 색상을 추출하기 위해 RGB에서 HSV로 색상 변환을 한다. HSV는 색을 표현하는 하나의 방법으로 색상 (Hue), 채도 (Saturation), 명도 (Value)의 좌표를 써서 특정한 색을 지정한다. HSV 이미지에서 차선 색상과 비슷한 흰색, 노란색 계열에 대해서만 추출하여 흰색과 검은색 두 가지 색으로만 이루어진 바이너리 이미지 (Binary image)를 생성한다.

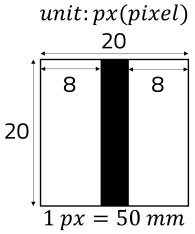



그림 6. 하르 특징 기반 1차 분류기
Fig. 6. First classifier based on haar-like feature

생성된 바이너리 이미지는 캐니 에지 (Canny edge) 기법을 사용하여 이미지에서 에지 정보들을 추출한다. 추출된 에지들은 허프 변환 (Hough transform)을 사용하여 선분들을 검출하고 선분들 사이의 간격을 계산하여 실제 차선의 폭과 비슷한 값이면 차선 후보군으로 추가한다.

캐니 에지 검출과 허프 변환의 경우 사전에 설정된 파라미터에 영향을 많이 받으며 차선 검출에 대한 정확도가 낮기에 머신 러닝 기법을 사용하여 정확도를 향상시킨다.

### 3. 1차 하르 특징 기반 차선 검출

차선을 대략적으로 빠르게 검출하기 위해 그림 6과 같이 임의로 만든 하르 특징을 사용하여 조감도 이미지에서 차선을 검출한다. 에지 기반 차선 검출과는 달리 별도의 전처리 작업이 필요하지 않고 오직 명암 정보만을 사용하기에 검출 속도가 빠른 장점이 있다.

한 픽셀 당 50mm의 해상도 (Resolution)를 가지며 슬라이딩 윈도우 접근법 (Sliding window approach)을 사용하여 $20 \times 20$ 크기를 가지는 1차 분류기에서 계산된 차선일 확률은 식 16, 17과 같다.

$$p = \frac{1}{n} \sum_{y=1}^{20} \left\{ \sum_{x=9}^{12} I_{(x,y)} - \frac{1}{4} \left( \sum_{x=1}^{8} I_{(x,y)} + \sum_{x=13}^{20} I_{(x,y)} \right) \right\}, \quad (16)$$

$$n = \frac{1}{4 \times 20 \times 255}. \quad (17)$$

$I_{(x,y)}$는 조감도 이미지의 관심 영역에서 각 좌표에 대한 픽셀 값이며 $n$은 검은색 사각형 영역의 크기 대한 평균 픽셀 값을 확률로 계산하기 위한 값이다. $p$는 해당 영역이 차선일 확률을 나타내며 0에서 1사이의 값을 가진다. 매번 이중 적분을 하는 것은 비효율적이기 때문에 하나의 조감도 이미지에 대해 특정 영역의 픽셀 값의 합을 효율적으로 계산하기 위한 적분 이미지 (Integral image)를 사용한다.

$p$가 특정 임계값 (Threshold) 보다 크면 차선과 비슷한 밝기의 변화가 있다고 판단하여 머신 러닝 기법을 통해 정밀하게 차선을 검사한다. 특정 임계값 보다 작은 경우 현재 영역을 버리고 다음 영역을 검사하게 된다. 조감도 이미지의 장점은 실제 3차원 물체에 대한 크기가 반영되어, 슬라이딩 윈도우 접근법에서 스케일 (Scale)은 고려하지 않아도 되기 때문에, 계산 복잡도가 줄고 검출 속도가 빠르다.

1차 분류기는 흰색 영역들의 픽셀 평균과 검은색 영역의

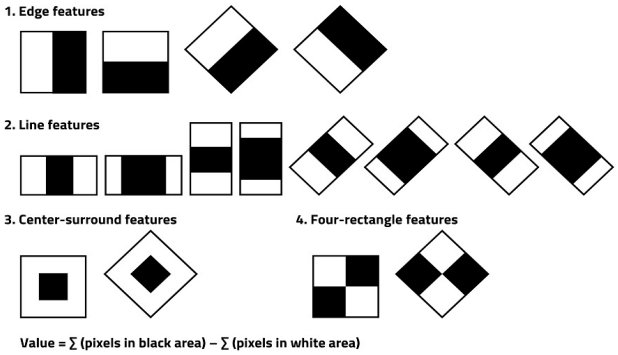



그림 7. 하르 특징
Fig. 7. Haar-like feature

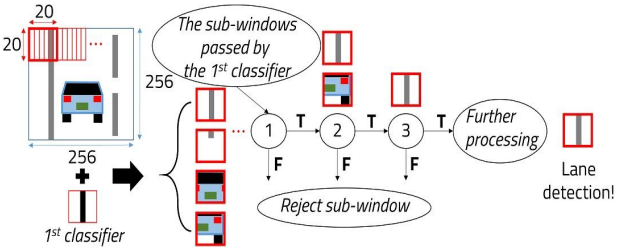



그림 8. 캐스케이드 분류기
Fig. 8. Cascade classifier

픽셀 평균의 차를 통해 계산하기 때문에 특정 임계값을 넘어가더라도 실제로 차선이 아닌 경우가 발생한다. 이렇게 차선의 오 검출을 방지하기 위해 머신 러닝 기법을 적용하여 보다 강인하게 차선을 검출한다.

#### 4. 머신 러닝 기반 차선 검출

차선 검출 알고리즘은 하르 특징을 기반으로 한 캐스케이드 분류기를 사용한다. 하르 특징을 기반으로 한 캐스케이드 분류기는 머신 러닝 기반이며 딥 러닝 기법 보다 더 빠르고 가벼운 장점이 있다.

그림 7과 같이 하르 특징들은 지정된 흰색 사각 영역들의 모든 픽셀 합과 검은색 사각 영역들의 모든 픽셀 합의 차이를 통해서 단일 값으로 계산된다. 이러한 하르 특징 하나가 하나의 약 분류기 (Week classifier)에 해당되는데 약 분류기 하나만 가지고 차선을 검출하기에는 정보가 많이 부족하다. 그렇기에 약 분류기를 여러 개 사용하여 보다 강인한 강 분류기 (Strong classifier)를 만들어야 하는데, 차선 검출에 있어 의미 있는 약 분류기들만을 추출하기 위해 Adaboost 알고리즘을 사용해서 학습한다 [9, 10].

최종적으로 캐스케이드 함수는 앞서 1차 분류기를 통과한 20×20 크기의 차선에 해당하는 수많은 포지티브 이미지들 (Positive images)과 1차 분류기를 통과한 차선이 아닌 수많은 네거티브 이미지들 (Negative images)을 통해 학습되어 적합한 강 분류기들을 그림 8과 같이 단계별로 구성한다.

만약에 캐스케이드를 사용하지 않고 학습을 통해 얻은 하르 특징인 약 분류기들을 모두 다 사용한다면, 흰색 배경과 같은 20×20의 관심 영역에서 불필요한 약 분류기를 모두 다 비교하기 때문에 비효율 적이다. 캐스케이드는 학습되어 얻어진 약 분류기들을 적절하게 그룹화 하여 여러 개의 강 분류기들을 생성해 보다 효율적으로 단계별로 검사를 한다.

이미지에서 검출 하고자 하는 관심 영역은 각 단계에서 학습된 강 분류기를 통해 비교하는데, 계산된 값이 특정 임계값 이상이면 다음 단계로 넘어간다. 계사된 값이 특정 임계값 이하이면 현재 관심 영역은 버리고 다음 관심 영역에 대해 비교를 하고, 모든 단계를 다 통과하면 최종적으로 차선으로 검출이 된다.

1차 분류기를 통과한 영역에 대해서 캐스케이드 분류기를 적용해 차선인지를 단계별로 검사한다. 낮은 단계에서는 적은 수의 약 분류기들로 생성된 강 분류기에 의해 빠르게 판단하며 상위 단계로 갈수록 약 분류기들이 많아 좀 더 정밀하게 검사한다. 캐스케이드 또한 조감도 이미지를 기반으로 검사하기 때문에 스케일은 고려하지 않고 오직 20×20 크기의 영역에 대해서만 계산하여 검출 속도가 빠르다.

## Ⅲ. 실험 및 결과

본 논문에서는 I.MX6Q [11] 보드의 리눅스 환경에서 사전에 저장된 다양한 환경에서의 고가도로 주행 영상을 통해 알고리즘 성능 비교 테스트를 진행 하였으며, 보드는 그림 9와 같이 구성되어있다. I.MX6Q는 NXP 회사에서 만든 프로세서로 4개의 32-bit ARM Cortex-A9 프로세서가 탑재되어 있으며 코어 당 최대 동작 속도는 1.2 GHz이다.

그림 10과 같이 에지를 기반으로 차선을 검출하는 방식은 조감도 이미지로부터 HSV변환, 바이너리 이미지 생성, 캐니 에지 검출, 허프 변환을 통한 선분 검출, 차선 후보 추출의 단계가 진행되며 한 프레임당 처리시간이 평균 33.71 msec가 소요되었다.

하지만 제안하는 방식은 조감도 이미지로부터 그레이스케일 변환, 1차 분류기 및 캐스케이드 적용, 차선 후보 추출의 단계가 진행되며 한 프레임당 처리시간이 평균 8.75 msec가 소요되어 차선 검출 속도가 약 3.85배 향상되었음을 확인할 수 있다.

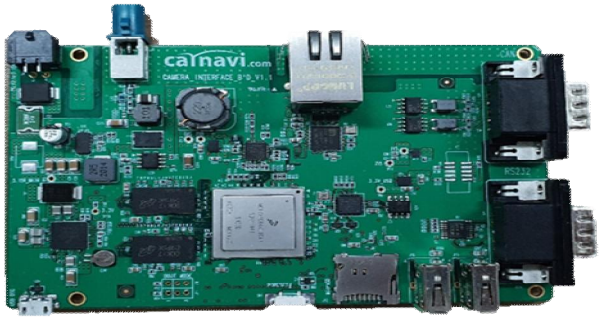



그림 9. I.MX6Q 보드
Fig. 9. I.MX6Q board

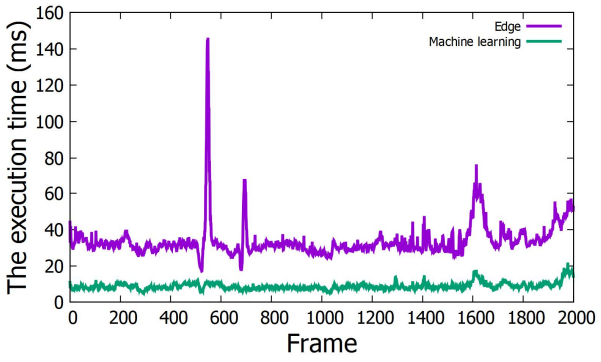



그림 10. 프레임 수에 따른 실행 시간
Fig. 10. The execution time according to the number of frames

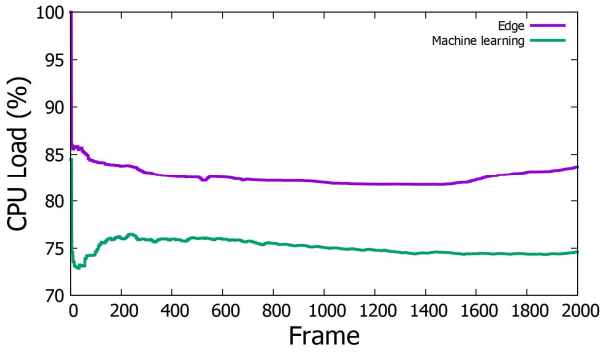



그림 11. 프레임 수에 따른 CPU 부하
Fig. 11. CPU Load according to the number of frames

그림 11과 같이 프레임 수에 따른 CPU 부하 또한 에지 기반 차선 검출은 약 82.65 %이며, 머신 러닝 기반 차선 검출은 약 75.06 %로 측정되어 약 1.1배의 부하가 감소했다.

이러한 결과들은 제안하는 알고리즘이 캐니 에지 및 허프 변환을 위한 작업보다 효율적이며 속도가 빠르다는 것을 보여준다. 캐니 에지 변환은 사전에 정의한 파라미터에 영향을 많이 받으며 영상의 노이즈와 밀접한 관련이 있다. 노이즈 제거를 위해 성능 향상을 위한 전처리 작업을 추가하는 경우, 계산 복잡도가 높아져 속도가 저하되는 경우가 있다. 허프 변환 또한 사전에 정의한 파라미터에 영향을 받는데, 이러한 최적화된 파라미터 작업은 환경마다 달라 가변적이며 사람이 직접 작업을 해야 하는 단점이 있다. 가끔씩 처리시간이 순간적으로 높아지는 경우가 있는데, 이미지에서 에지 성분들을 많이 포함하는 경우이다. 하지만 머신 러닝 기반 차선 검출은 별도의 파라미터 설정 없이 다양한 환경에서 처리시간이 비교적 일정한 것을 보여준다.

그림 12는 차선 검출 순서도에서 각 단계별로 처리되는 결과이다. 원본 이미지로부터 조감도 이미지를 생성하고 (a), 그레이스케일로 변환하여 에지 및 머신 러닝 기반 차선 검출을 수행한다. 에지 기반 차선 검출의 경우 차선에 대한 색상을 검출하기 위해 HSV 변환 후 바이너리 이미지를 생성한다 (b). 바이너리 이미지에서 캐니 에지 검출을 통해 에지 성분을 검출하고 허프 변환을 통해 선분들을 추출한다 (c). 생성된 선분들에 대해 간격을 계산하고 실제 차선의 간격과 비슷한 값들을 차선 후보군에 등록한다 (d).

머신 러닝 기반 차선 검출의 경우 슬라이딩 윈도우에 의해 선택된 영역에 대해서 1차 분류기를 적용하여 차선을 검출한다 (e). 1차 분류기를 통과한 영역들은 바로 머신 러닝 기반 캐스케이드 분류기를 적용하여 통과한 영역들은 차선 후보군에 등록한다 (f). 캐스케이드 분류기를 통과한 영역들

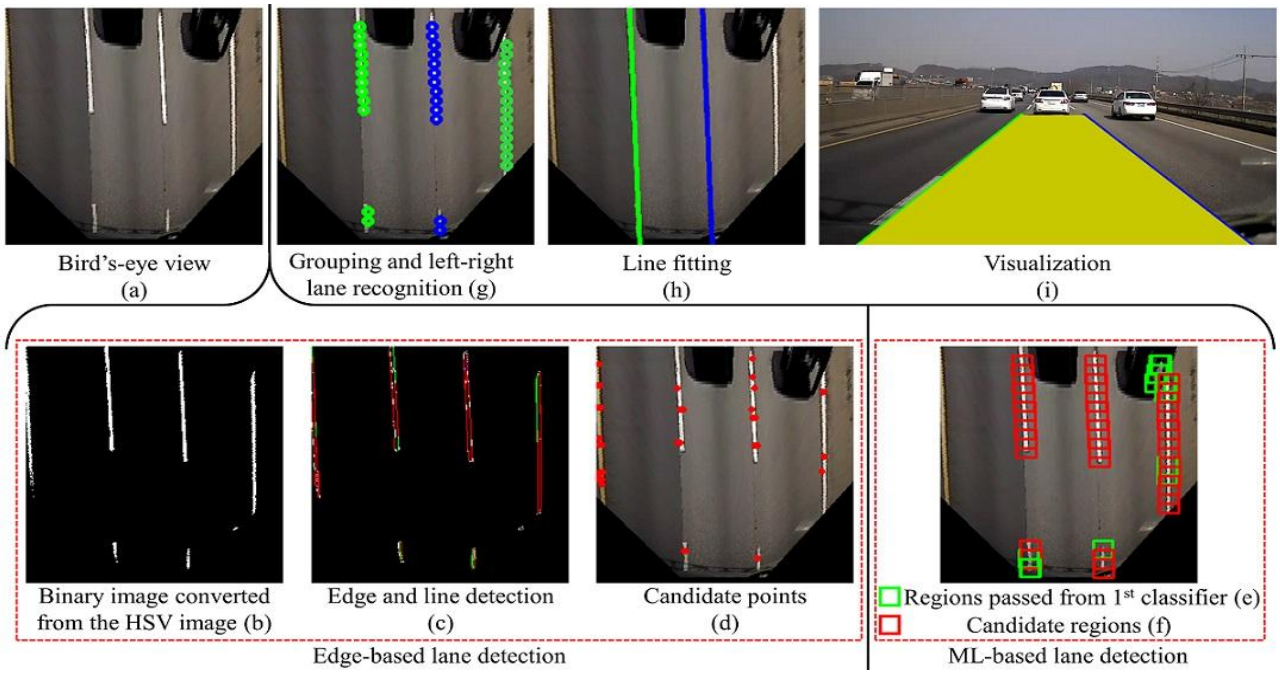



그림 12. 차선 검출 순서도의 각 단계 결과
Fig. 12. Results of each step in the lane detection flowchart

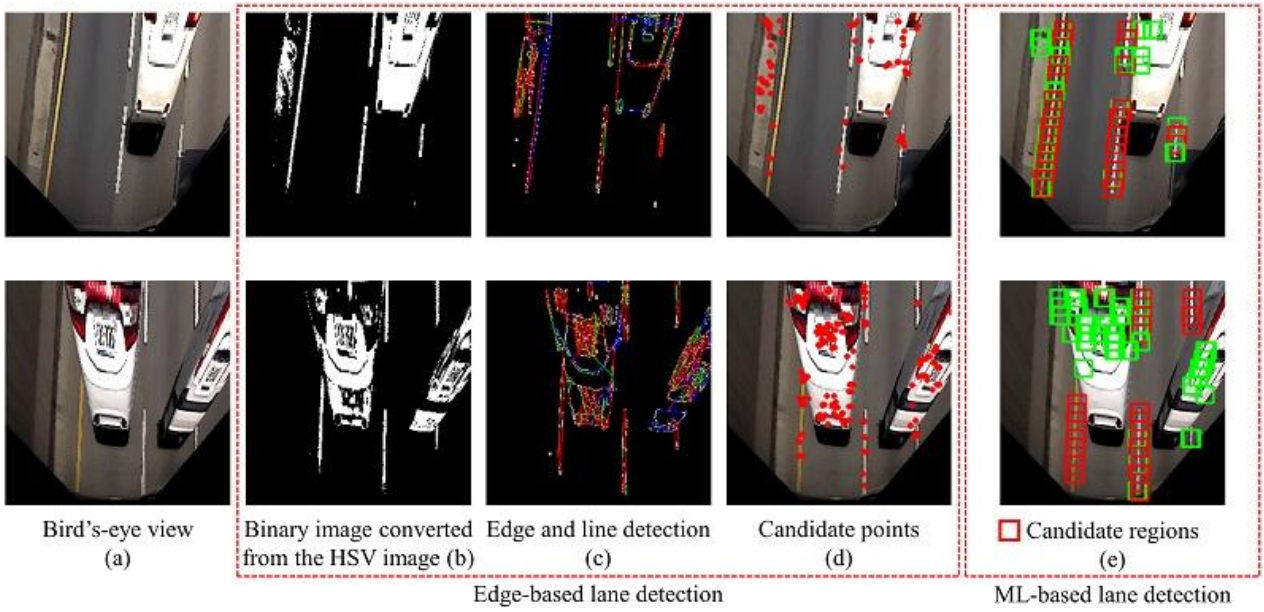



그림 13. 흰색 차량의 두 가지 특수한 경우에 대한 에지 및 ML 기반 차선 검출 비교
Fig. 13. Comparison of edge and ML-based lane detection for two special cases of white vehicles

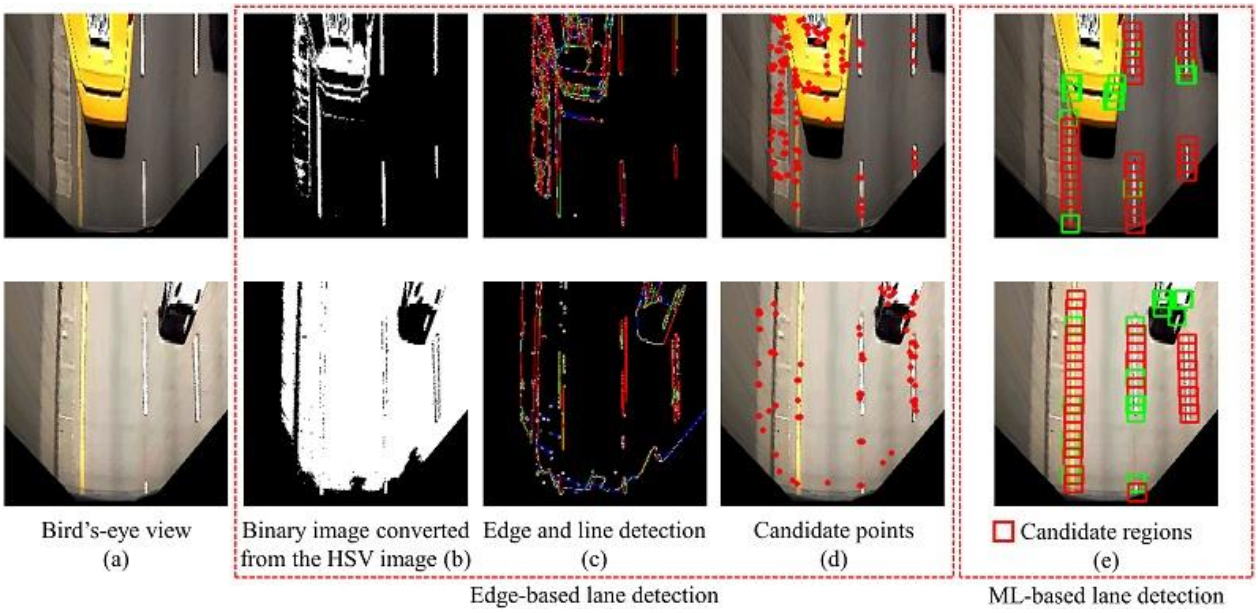



그림 14. 노란색 차량과 밝은 환경의 두 가지 특수한 경우에 대한 에지 및 ML 기반 차선 검출 비교
Fig. 14. Comparison of edge and ML-based lane detection for two special cases of yellow vehicle and bright environment

은 빨간색 사각형으로 표시되며, 녹색 사각형을 포함한다. 이렇게 등록된 차선 후보군들은 동일한 차선인지 아닌지에 대한 판별을 위해 그룹화 작업을 하고 왼쪽 차선 (녹색 점)과 오른쪽 차선 (파란색 점)으로 구분한다 (g). 구분된 왼쪽, 오른쪽 차선에 대한 유한한 점들로부터 노이즈 제거 및 여러 위치로부터 값들을 유추하기 위해 가장 적합한 직선을 찾는 직선 적합 작업을 수행한다 (h). 마지막으로 직선 적합한 차선들을 원본 이미지로 투영하여 시각화를 통해 차선을 표시한다 (i).

그림 13은 전방에 흰색 차량이 있는 경우에 대한 에지 및 머신 러닝 기반 차선 검출을 비교한 결과이다. 에지 기반 차선 검출에서는 조감도 이미지에서 (a) HSV 변환 후 흰색과 노란색 성분에 대해서만 픽셀 값을 255로 계산하고 나머지는 0으로 만드는 바이너리 이미지를 생성한다 (b). 흰색 차량 또한 같은 색상 성분을 가지기에 바이너리 이미지에 노이즈가 형성된다. 노이즈를 포함한 바이너리 이미지에서 에지 검출 및 허프 변환을 하게 되면, 다음과 같이 무수히 많은 선분들이 검출된다 (c). 검출된 선분들 또한 노이즈가

포함되어있기 때문에 차선 후보군들이 흰색 차량에 대해서도 검출되는 것을 확인 할 수 있다 (d). 하지만 머신 러닝 기반 차선 검출은 흰색 차량의 일부 영역에서는 1차 분류기가 통과되었지만 (녹색 사각형), 캐스케이드 분류기에서는 통과되지 않아 (빨간색 사각형) 차선 후보군에서 제외된 것을 확인 할 수 있다 (e).

그림 14는 전방에 노란색 차량이 있거나 또는 주위의 환경이 밝아지는 경우에 대한 에지 및 머신 러닝 기반 차선 검출을 비교한 결과이다. 조감도 이미지에서 (a) 생성된 바이너리 이미지 또한 노이즈가 포함되어, 에지 기반 차선 검출에서는 차선 후보군의 오 검출이 많은 것을 확인 할 수 있다 (b)-(d). 이와 반대로 머신 러닝 기반 차선 검출은 색상과 밝기 변화에 상관없이 차선을 정확하게 검출하는 것을 확인 할 수 있다 (e).

## Ⅳ. 결 론

본 논문은 에지 기반으로 차선 검출하는 것 보다 정확도 및 검출 속도가 향상된 머신 러닝 기반 차선 검출을 소개하였다. 차선을 정확하게 검출하기 위해서는 원근 효과가 제거된 이미지가 필요하다. 조감도 이미지는 역 원근 변환을 사용하여 원근 효과를 제거하는데, 호모그래피 변환과는 달리 오직 설치된 카메라의 높이와 기울어진 각도만 알고 있다면 조감도 이미지를 생성할 수 있다.

에지 정보는 환경 변화에 취약하며 차선을 검출하기 위해서는 정보가 부족하기 때문에 오 검출이 많이 발생하는 것을 확인 할 수 있었다. 이러한 오 검출을 방지하기 위해서는 별도로 보완해주는 작업이 필요한데 에지 정보만으로는 쉽지가 않다. 이와 반대로 머신 러닝 기법은 다양한 차선 정보들과 차선이 아닌 정보들을 사용하여 학습하기 때문에 정확한 차선 검출이 가능하다. 에지 기반 차선 검출은 HSV 변환, 그레이스케일 변환, 바이너리 이미지 생성, 캐니 에지 검출, 허프 변환을 통한 선분 검출과 같은 별도의 전처리 작업이 필요하다.

임베디드 환경에서 실시간으로 동작 성능을 보장하기 위해서는 계산 복잡도를 낮춰 경량화 하는 작업이 중요하다. 기존의 캐스케이드 분류기에 비해 계산 복잡도가 낮은 1차 분류기를 그레이스케일로 변환된 조감도 이미지에 적용하여 차선에 대한 확률을 계산한다. 기존의 캐스케이드 분류기는 학습된 하르 특징들을 기반으로 단계별로 계산하기 때문에 계산 복잡도가 높고 가변적이다. 하지만 1차 분류기는 한 개의 하르 특징을 기반으로 차선에 대한 확률을 계산하기 때문에 계산 복잡도가 낮아 전처리 작업에 적합하며 속도 향상이 가능하다. 차선을 보다 정밀하게 검출하기 위해 1차 분류기를 통과한 영역에 대해서 캐스케이드 분류기를 사용한다. 1차 분류기 및 캐스케이드 분류기에서는 크기가 고정된 20×20 크기의 관심 영역에 대해서만 계산하기 때문에 모든 크기를 고려하는 슬라이딩 윈도우 접근법 보다 검출 속도가 빠른 장점이 있다.

차선 검출의 정확도를 향상시키기 위해서는 다양한 환경에서의 수많은 이미지에 대한 학습이 필요한데, 머신 러닝의 경우 이미지들을 직접 분류하여 수동으로 제공해야 한다는 번거로움이 있다. 또한 학습한 내용을 기반으로 판단이나 예측하기 때문에 새로운 환경에서는 오 검출이 발생할 수 있다.

이와 반대로 딥 러닝은 분류에 사용할 데이터를 스스로 학습할 수 있으며 뇌의 뉴런과 유사한 정보 입출력 계층을 사용하기에 이미지 인식 능력이 뛰어난 장점이 있다. 그러나 많은 양의 연산이 필요하기 때문에 신경망의 연산 속도를 가속화하는 기법이 필요하다.

향후에는 딥 러닝 기반으로 차선 검출을 하면서 다양한 환경에 대한 정확도를 향상시키고, 실시간으로 동작할 수 있도록 최적화하는 연구를 진행 할 예정이다.

## References

[1] A. Kamble, S. Potadar, "Lane Departure Warning System for Advanced Drivers Assistance," 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), pp. 1775-1778, 2018.

[2] C. Y. Low, H. Zamzuri, S. A. Mazlan, "Simple Robust Road Lane Detection Algorithm," 2014 5th International Conference on Intelligent and Advanced Systems (ICIAS), pp. 1-4, 2014.

[3] F. Bounini, D. Gingras, V. Lapointe, H. Pollart, "Autonomous Vehicle and Real Time Road Lanes Detection and Tracking," 2015 IEEE Vehicle Power and Propulsion Conference (VPPC), pp. 1-6, 2015.

[4] D. Ding, C. Lee, K. Lee, "An Adaptive Road ROI Determination Algorithm for Lane Detection," 2013 IEEE International Conference of IEEE Region 10 (TENCON 2013), pp. 1-4, 2013.

[5] J. Shin, E. Lee, K. Kwon, S. Lee, "Lane Detection Algorithm Based on Top-view Image Using Random Sample Consensus Algorithm and Curve Road Model," 2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 1-2, 2014.

[6] H. K. Kim, Y. H. Ju, J. H. Lee, Y. W. Park, H. Y. Jeong. "Lane Detection for Adaptive Control of Autonomous Vehicle," IEMEK J. Embed. Sys. Appl., Vol. 4, No. 4, pp. 180-189, 2009 (in Korean).

[7] P. Viola, M. Jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Vol. 1, pp. I-I, 2001.

[8] C. Zhang, G. Liu, X. Zhu, H. Cai, "Face Detection Algorithm Based on Improved AdaBoost and New Haar Features," 2019 12th International Congress on Image and Signal Processing, BioMedical Engineering and

Informatics (CISP-BMEI), pp. 1-5, 2019.
[9] Z. Hao, Q. Feng, L. Kaidong, "An Optimized Face Detection Based on Adaboost Algorithm," 2018 International Conference on Information Systems and Computer Aided Education (ICISCAE), pp. 375-378, 2018.
[10] K. Chang, C. Fan, "Cost-Efficient Adaboost-based Face Detection with FPGA Hardware Accelerator," 2019 IEEE International Conference on Consumer Electronics - Taiwan (ICCE-TW), pp. 1-2, 2019.
[11] NXP i.MX 6 Series of Industrial Applications Processors [Internet]. Available: https://www.nxp.com/docs/en/data-sheet/IMX6DQIEC.pdf.

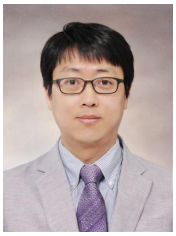
**Sunghoon Hong (홍 성 훈)**

2014 Electronic Engineering from Cheongju University (B.S.)
2016 Intelligent Robot Engineering from Hanyang University (M.S.)
2021~Electronic and Electrical Engineering from Kyungpook National University (Ph.D. Student)

Career:
2016 Research Engineer, Carvi
2018 Assistant Engineer, Yujin Robot
2021 Senior Engineer, Carnavicom

Field of Interests: Human-like Artificial Intelligence Autonomous Driving Systems
Email: hopsison@gmail.com

**Daejin Park (박 대 진)**



2001 School of Electronics Engineering from Kyungpook National University (B.S.)
2003 School of Electronics Engineering from KAIST (M.S.)
2014 School of Electronics Engineering from KAIST (Ph.D.)
2016~School of Electronics Engineering from Kyungpook National University (Associate Professor)

Career:
2003~2014 Research Engineer, SK Hynix/ Samsung

Field of Interests: Low-power SoC Design, Robust Embedded Systems
Email: boltanut@knu.ac.kr