개선된 버드아이뷰 변환을 활용한 효율적인 차선 검출 방법

정현석, 임석호, 윤현주
금오공과대학교 컴퓨터공학과
e-mail:peebbv6364@kumoh.ac.kr

Efficient Lane Detection Method using Improved Bird's Eye View Transform

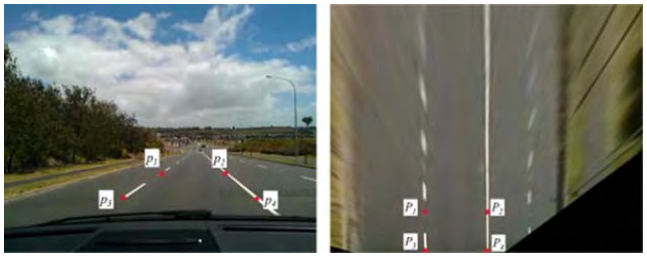
Hyeon-Seok Jeong, Seok-Ho Im, Hyeon-Ju Yoon
Dept. of Computer Engineering, Kumoh National Institute of Technology

요 약

차선 검출은 자율주행 자동차의 가장 기본 기능 중의 하나이다. 전방 카메라를 통하여 얻은 입력 영상을 변환하여 주행 방향을 정할 수 있도록 차로를 검출하는 방법은 여러 가지가 있는데, 본 논문에서는 버드아이뷰 영상을 활용하는 방법을 채택하고 여러 가지 성능이 제한적인 임베디드 시스템에서 이를 보다 효율적으로 수행할 수 있도록 EPM(Expected Perspective Mapping) 방법과 변환 영상을 이용해 차로를 검출하는 슬라이딩 윈도우 알고리즘의 개선 방안을 제안한다. 제안된 방법은 기존의 차선 검출 방법에 비해 약 30% 이상 적은 연산량으로 수행할 수 있으면서 기존 방법과 동일한 결과를 생성하여 실시간성이 중요한 상황에서 정확한 차선 검출을 할 수 있음을 보여 준다.

1. 서론

차선 검출은 자율주행 자동차의 가장 기본 기능 중의 하나로, 허프 라인 트랜스폼[2], RANSAC (RAndom SAmple Consensus)[2], 버드아이뷰(Bird's Eye View) 변환[2]과 같은 방법들이 활용된다. 이 중 버드아이뷰 변환은 차선 검출 전에 영상을 하늘에서 아래로 내려다보는 형태로 변환하는 것이다. 이 변환을 이용하면 차선 검출 시, 기존 영상에서 검출하는 것에 비해 차선의 기울기 값이 양쪽 차선 모두 고르게 나와서 처리가 쉬워진다.



(그림 1) 버드아이뷰 변환

전방 카메라로 촬영된 영상은 (그림 1)의 왼쪽과 같은 형태이며, 버드아이뷰 변환을 적용하면 오른쪽 그림처럼 된다. 변환하는 방법으로는 IPM(Inverse Perspective Mapping)[2], WPM(Warp Perspective Mapping)[2]이 있는데 이는 (그림 1)의 두 영상의 P_1, P_2, P_3, P_4 네 점에 대한 변환 호모그래피(homography) 행렬을 구해서 다른 모든 픽셀에 각각 적용해 전체 영상을 변환하는 것이다.

본 논문에서는 하드웨어 성능이 제한적인 상황에서 보다 빠른 속도로 버드아이뷰 변환을 수행하기 위해 EPM(Expected Perspective Mapping) 방식을 제안한다. EPM은 차로가 직선인 점에 착안해 픽셀마다 호모그래피 행렬 연산을 수행하지 않고 직선의 방정식을 활용해 차선에 대한 기댓값을 계산하여 버드아이뷰 영상으로 변환시키는 방법이다. 다른 방식과 비교하여 시간 복잡도는 $O(n)$으로 동일하지만 320×240 해상도의 영상을 기준으로 할 때 계산량이 약 80% 이상 줄어들기 때문에 실시간 판단이 중요한 자율주행 자동차에 보다 적합한 방식이며, 생성된 결과는 다른 방법과 동일하여 주행 정확성이 유지된다.

한편, 변환된 영상에서 직선 및 곡선 차로를 검출하는 방법으로는 허프 라인 변환, RANSAC(RAndom SAmple Consensus), 슬라이딩 윈도우[1]와 같은 다양한 방법이 제안되었다. 이 중에서 슬라이딩 윈도우 알고리즘이 타 검출 방법에 비해 선을 검출하는 속도가 빠르기 때문에 우리는 이 방식을 채택하였다. 그리고 조금 더 연산 속도를 증가시킬 수 있도록 계산량이 평균 30% 이상 줄어드는 개선된 슬라이딩 윈도우 알고리즘을 제안하였다.

본 논문에서 제시하는 EPM과 개선된 슬라이딩 윈도우 알고리즘 모두 기존 방법들에 비해 연산양이 줄어들어서 실시간성이 보장되어야 하는 자율주행자동차 응용에 보다 적합한 성능을 보인다.

2. 개발 하드웨어 사양

(그림 2) 개발 자율주행 모형자동차(현대오트론) 사진

항목	규격 및 내용
크기, 무게	330×190×160mm, 약 2.3kg
구동모터	DC서보 모터 엔코더(12V-12W)
서브모터	6V, 9kg (바퀴 조향 및 카메라 틸트)
카메라	CMOS VGA급(640x480), CVBS 출력
보드	NVIDIA Tegra3(연산), Arudino(제어)

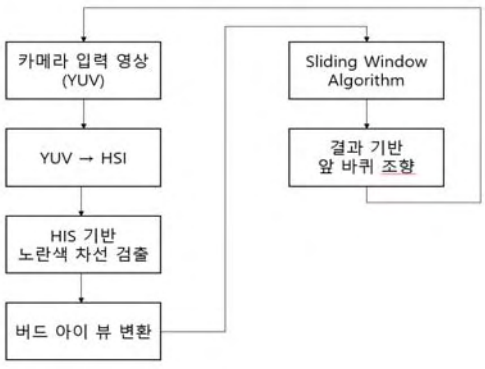
(표 1) 개발 자율주행 모형자동차 하드웨어 사양 요약

본 연구의 기반이 되는 자율주행 모형자동차는 2017 임베디드 소프트웨어 경진대회 본선진출 팀에게 주어지는 기기로 상세 기기 사양은 공식 홈페이지[3]를 통해 확인할 수 있다((그림 2) 및 (표 1) 참조)

3. 차선검출을 위한 과정

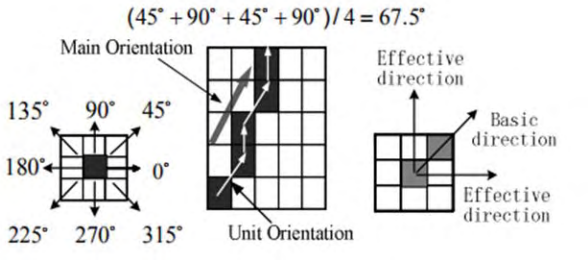
실시간으로 차선 검출을 위해 자동차에 부착된 카메라에서 10Hz 속도로 영상을 받는다. 입력 영상을 색채에 민감한 HSI 좌표계로 변환 후 Hue값이 20~55사이에 있는 값을 추출하여 밝기, 채도에 민감하지 않은 노란색 차선을 검출한 이진영상을 추출한다. 추출된 영상을 변환하여 버드아이뷰 영상으로 만든 후 슬라이딩 윈도우 알고리즘을 이용하여 차선 정보를 얻은 후에 결과를 기반으로 앞바퀴를 조향한다. 아래 (그림 3)은 차선 검출을 위한 전체적인 순서도다.

버드아이뷰 변환은 앞에서 소개한 바와 같이 기준 네 점을 이용한 호모그래피 행렬의 생성과 각 픽셀에 대해 행렬을 적용하는 연산으로 이루어진다.



(그림 3) 차선 검출을 위한 순서도

변환된 영상에서 차선을 검출하기 위해서는 영상을 작은 영역, 즉 슬라이딩 윈도우로 분할하여 (그림 4)와 같이 아래부터 위쪽으로 올라가면서 영상에서 차선이 검출되는 지점을 찾는다. 양쪽 차선검출을 위해 왼쪽, 오른쪽 각각 방향에서 픽셀이 검출되는 지점을 찾은 후 검출되는 지점을 차로로 인식한다. 이후 아래쪽에서 올라오면 0° ~ 180° 각도에 있는 픽셀을 확인하고 이 가중치로 평균을 내어 차선의 각도를 검출한다.



(그림 4) 슬라이딩 윈도우를 활용한 차선 검출[2]

4. 개선된 차선 검출 과정

본 연구에서는 계산량을 줄여 임베디드 시스템의 실시간 처리를 빠른 속도로 수행하기 위해 버드아이뷰 변환 과정과 슬라이딩 윈도우 알고리즘을 개선하였다.

4.1 EPM을 활용한 버드아이뷰 변환

EPM(Expected Perspective Mapping) 방법은 모든 픽셀에 대해 변환 행렬을 적용하는 것이 아니라 차로가 직선 형태임에 착안하여 직선의 방정식을 활용하고 일부 픽셀에 대해서만 변환을 수행한다. 카메라 영상으로부터 현재 차선이 직진 차로일 때 나타나는 네 점을 예상해서 양쪽 차선의 함수를 구한다. 이 함수를 (식 1)의 (1)과 (2)로 나타낼 수 있다.

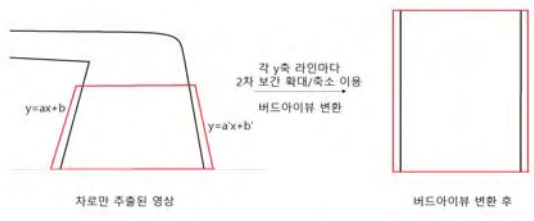
$$(1)\ y = ax + b \ \Rightarrow\ \frac{y-b}{a} = x$$

$$(2)\ y = a'x + b' \ \Rightarrow\ \frac{y-b'}{a'} = x'$$

(식 1) 예상 양쪽 차선의 식과 변형

(식 1)에서 x, y는 예상되는 픽셀에 대한 좌표를 의미하고 a, b는 네 점을 통해 구해진 함수의 기울기와 y절편 값이다. (1)과 (2)는 각각 양쪽 차선에 대한 기대함수이다. 기대함수를 y값을 기준으로 변형하면 각 차선마다 시작과 끝 픽셀의 x값을 알 수 있다. 이를 2차 보간 확대/축소 방법을 이용해서 원하는 해당 라인을 버드아이뷰의 가로축 길이만큼 만들면 버드아이뷰 영상을 구할 수 있다.

(그림 5)는 (식 1)의 EPM 방식의 버드아이뷰 변환 방법을 그림으로 나타낸 것이다. 결과적으로 WPM 방식을 적용했을 때와 동일한 결과영상이 추출되므로 차선을 찾는 알고리즘을 적용하는데 차이가 없다.

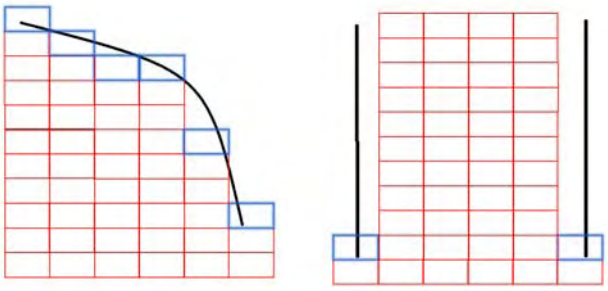


(그림 5) EPM을 활용한 버드 아이뷰 변환

4.2. 개선된 슬라이딩 윈도우 알고리즘

슬라이딩 윈도우 알고리즘을 개선하기 위해 차선의 모양에 따라 적응적으로 슬라이딩 윈도우를 생성하는 기법을 제안한다. 입력 영상을 가로로 분할하여 세로축을 따라 윈도우를 올리면서 차선이 검출되는 지점을 찾을 때 영상 내 모든 영역을 찾는 것이 아니라, 윈도우 중 픽셀들이 최초로 모이는 지점을 찾으면 그 곳을 차선이 존재하는 곳으로 인식하여 기록하고 진행을 멈춘다.

(그림 6)은 개선된 슬라이딩 윈도우 알고리즘이 실행되는 그림이다. 이 그림은 320×240 영상에서 윈도우 크기를 40×20으로 설정했다. 그러면 가로축 기준 8개, 세로축 기준 12개로 분할이 되는데 세로는 별도의 분할 없이 아래쪽 윈도우에서 올라오면서 검출이 되는 경우 멈춘다. 8개의 가로축 기준으로 분할된 모든 윈도우를 위쪽으로 슬라이딩 하면서 현재 차선상태를 기록하면 된다.

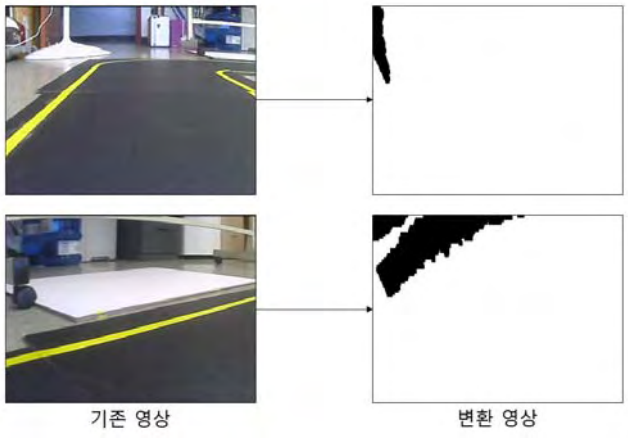


(그림 6) 개선된 슬라이딩 윈도우 알고리즘

(그림 6)을 보면 좌측 사진의 좌회전 차로일 경우 6개의 윈도우에서 차선이 검출되고 검출되는 윈도우의 y축 값이 왼쪽으로 올라가는 방향으로 검출된다. 만약, 우회전 차로일 경우 오른쪽으로 검출되는 윈도우 y축 값이 증가할 것이다. 반면 우측 사진의 직진 차로일 경우 오직 2개의 윈도우에서 차선이 검출되는 것을 알 수 있다. 이를 통해 검출되는 윈도우의 수와 검출되는 윈도우의 y축 증가방향을 확인하면 진행 방향을 정할 수 있다.

5. 결과 및 분석

(그림 7)은 실제 하드웨어 카메라를 통해 입력하고 변환을 통해 곡선 차로 및 직선 차로를 추출한 영상이다. (그림 7)의 위 영상은 좌측으로 치우친 직선차로일 때의 영상으로 이 경우 약간만 우측으로 방향을 조향해서 양쪽 차선이 추출되면 다시 바퀴를 직진 방향으로 조향한다. 아래 영상은 우회전 차로일 때 나오는 영상으로 5개 이상의 윈도우에서 오른쪽으로 증가하는 방향으로 검출이 되므로 우회전이라고 판단하고 우회전 방향으로 바퀴를 조향한다.



(그림 7) 실제 모형자동차에서 적용한 결과 영상

우리가 제안한 방법은 기존 알고리즘들에 비해 계산량이 적어서 실시간 자율주행에 보다 적합하다. (표 2)는 EPM 방법과 기존에 제시된 IPM, WPM 방법의 시간 복잡도이다. O(n)은 사다리꼴에 해당하는 모든 픽셀들을 한 번 반복문으로 접근할 때 시간 복잡도이고 n은 픽셀 수를 의미한다.

알고리즘	시간 복잡도
IPM (Inverse Perspective Mapping)	$9\times O(n)$ + @
WPM (Warp Perspective Mapping)	$9\times O(n)$
EPM (Expected Perspective Mapping)	$O(n)$

(표 2) 버드아이뷰 변환 알고리즘 비교

IPM, WPM은 3×3 호모그래피 행렬을 모든 픽셀에 적용해야 하므로 9×O(n) 시간 복잡도를 나타낸다. 더해서 IPM은 현재 카메라의 위치, 각도, 카메라 중앙방향 위치의 실제 거리 등을 고려하여 구현하므로 시간이 WPM보다 조금 더 걸린다. EPM은 양쪽 끝점을 알고 있으므로 행렬 적용은 한 번만 수행하고 보간법으로 해당 라인의 확대/축소를 수행한다. 보간법은 현재 기존 영상의 y축을 기준으로 첫 점과 끝 점 픽셀좌표와 현재 픽셀의 x좌표를 통해 변환 영상에 들어갈 픽셀을 매핑할 수 있다. 시간 복잡도는 $O(n)$이며 모든 픽셀에 3×3 행렬의 원소를 곱하는 기존 변환 알고리즘에 비해 약 80%의 연산량이 줄어든다.

슬라이딩 윈도우 알고리즘의 경우, 기존 방식은 모든 픽셀에 한 번씩 접근하여 연산하므로 $O(n)$의 시간 복잡도를 가진다. 그러나 개선된 슬라이딩 윈도우 알고리즘은 원하는 픽셀만큼을 가지는 윈도우를 찾는 시점에서 세로축 진

행이 종료되므로 최악의 경우 즉, 영상에서 차선이 검출되지 않는 경우에만 전체 픽셀을 접근하여 기존 방식과 계산량이 같다. 모형 자동차가 주행할 경로는 대부분 차로가 존재하므로 개선된 방식은 (그림 6)을 기준으로 전체 윈도우 수에서 검출되지 않는 윈도우 수를 빼보면 직선과 곡선 차로에서 평균 약 30% 이상 계산량이 줄어든다.

모든 경우에 대해 우리가 사용하는 방법이 완벽한 결과를 나타내는 것은 아니다. 주행 중 치우침으로 인해 한쪽 차선만 검출되는 경우나 너무 급격한 각도로 회전하는 경우 등은 별도의 예외 처리를 통해 주행이 이루어진다.

또한 개선된 슬라이딩 윈도우 알고리즘은 기존 방식에 비해 모든 픽셀에 대한 정보를 기록하지 않아 연산 시간이 줄어들 수 있다는 장점이 있는 반면, 카메라 노이즈에 비교적 약하다는 문제점이 있다. 아래쪽 윈도우에서 찾아 올라오면서 차선이 검출되었다고 기록한 윈도우가 실제로는 노이즈라면 오류가 전파되어 추후 문제가 커질 수 있기 때문이다.

이러한 노이즈 문제에 대해서는 Morphology 기법의 Closing 기법[4]처럼 노이즈를 미리 제거하거나 차로를 표현하는 색상 범위를 더 엄격하게 설정하는 방법을 적용할 수 있다.

6. 결론

본 논문에서는 제한된 계산 성능을 가진 임베디드 시스템 환경에서 영상으로부터 효율적으로 차선을 검출하는 방법을 제안하여 자율주행 모형자동차에 적용하였다. 제안된 EPM과 개선된 슬라이딩 윈도우 알고리즘은 기존 방식들보다 현저히 적은 계산량으로도 동일한 차선 검출 성능을 나타낸다. 이 방법은 현재 2017 임베디드 소프트웨어 경진대회 자율주행 모형자동차 부문 본선진출 팀 '금호우'에서 실제로 적용하고 있다.

주어진 하드웨어에 제약이 있어 보다 다양한 시도를 할 수 없었지만 최근의 추세에서처럼 GPU가 포함된 처리기를 사용할 수 있다면 제안한 EPM과 슬라이딩 윈도우 알고리즘의 부분 연산을 각 GPU 코어에 분배하여 훨씬 높은 성능 향상을 이룰 수 있을 것이다.

참고문헌

[1] Mehdi Sqalli, "Advanced Lane detection", https://medium.com/@MSqalli/advanced-lane-detection-6a769de0d581

[2] QING LIN, *Robust Lane Detection Method Using Bird's-eye View Transform,* 숭실대학교 학위(석사) 논문, 2011년

[3] 임베디드 소프트웨어 경진대회, http://eswcontest.com

[4] 임동훈, *OpenCV를 이용한 영상처리,* 자유아카데미, 2008년

[5] Duy Phuong Nguyen, *Implementation of High Performance Realtime Lane Detection Algorithm,* 숭실대학교 학위(석사) 논문, 2016년