가상차선 생성 및 PID 제어 기반의 자율주행 자동차 차선 추종 시스템 구현

Virtual Lane Generation and PID Control for Autonomous Vehicle System

요 약

자율주행 자동차의 차선 인식 기법은 영상 입력을 바탕으로 차선 인식을 수행하는데, 급커브 구간에서 차선 일부가 사라지는 등 차선 영상을 확보할 수 없는 상황에서는 차선 인식이 어려워진다. 본 논문에서는 이러한 영상 기반 차선 인식 기법의 성능 한계를 보완하기 위한 가상차선 생성 기법을 제안한다. 제안하는 기법은 일부 차선 영상을 확보할 수 없는 경우 이미지 구간 분할과 2차 곡선 다항식을 활용하여 가상의 차선을 구성하고 도로의 중앙값을 검출한다. 또한, 자율주행 자동차의 현재 위치와 차선 중앙 위치를 효과적으로 일치시키기 위해 PID 알고리즘을 적용한다.

1. 서 론

오늘날 테슬라, 구글, 엔비디아, 아우디 등 수 많은 회사에서 자율주행 자동차 개발에 뛰어들고 있다. 자율주행 자동차를 개발하기 위해 자동차와 각종 센서, 다양한 트랙, 안전 제어 장치 등의 환경 조성비와 실험에서 발생하는 유지 및 보수비 등의 천문학적인 예산이 필요하다. 하지만 위와 같은 대기업들 이외에는 자율주행 자동차 개발을 위해 막대한 예산을 투자하는 것은 현실적으로 쉽지 않다. 따라서 실제 자율주행 자동차 사업을 하기 힘든 중소기업과 개인 연구자들은 이러한 비용을 최소화하기 위해 시뮬레이션이나 모형 자동차를 통한 자율주행 제어 분야에 도전하고 있다.

본 논문에서는 영상 처리 기법[1]을 기반으로 도로의 차선을 인식하고 이에 따라 PID 알고리즘[2]을 통해 모형 자동차를 제어한다. 영상 처리 분야에서 차선을 인식하기 위해 쓰이는 대표적인 기법들로는 hough transform[3], canny edge[4], ransac[5] 등의 알고리즘이 있다. 하지만 단순히 위의 알고리즘만 사용하면 모형 자동차를 위한 트랙의 급커브 구간의 차선을 인식할 수 없다. 이러한 구간을 처리하기 위해 이미지 구간 분할 방식과 2차 곡선 다항식(2 degree polynomial curve fitting)을 위 기법들과 함께 적용한다.

2. 자율주행 자동차 시스템 구조

그림 1과 같이 실제 자동차의 위치와 카메라에 보이는 화면상에서의 자동차 위치와는 일정한 gap이 존재한다. 따라서 이 gap을 줄이기 위해 2개의 쓰레드를 생성한다.

그림 2는 자율주행 자동차 전체 시스템 구조도이다. 첫 번째 쓰레드는 웹캠을 통해 받은 영상을 grayscale, gaussian filter[6], canny edge, top view[7], lane detection(window sliding, polyfit) 등의 작업을 한다. 이를 통해 찾은 왼쪽 lane 또는 오른쪽 lane을 기반으로 도로 폭만큼 x축으로 이동하여 virtual lane을 생성한 후 현재 location을 구하고 이를 queue에 enqueue한다. 두 번째 쓰레드는 그림 1의 gap만큼 일정 시간 sleep을 하여 real sight와 camera sight를 일치시킨 후 queue로부터 dequeue하여 반환된 location을 토대로 pid process와 pid 최적화를 위한 twiddle process를 통하여 pid 값을 구하고 이를 Namedpipe를 통해 ROS(Robot Operating System)에 전달한다. ROS에서는 넘겨받은 pid를 기준으로 speed와 steering을 제어한다.

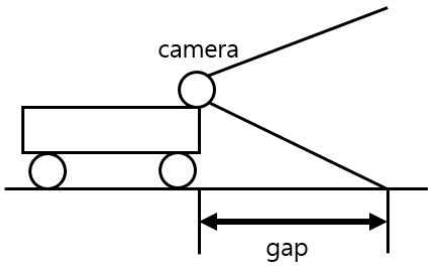


그림 1. real sight, camera sight 비교그림

3. 실 험

3.1. 영상처리

RGB의 모든 정보가 포함된 영상에서는 데이터 처리를 위한 시간이 오래 걸리기 때문에 edge 검출 이전 이미지를

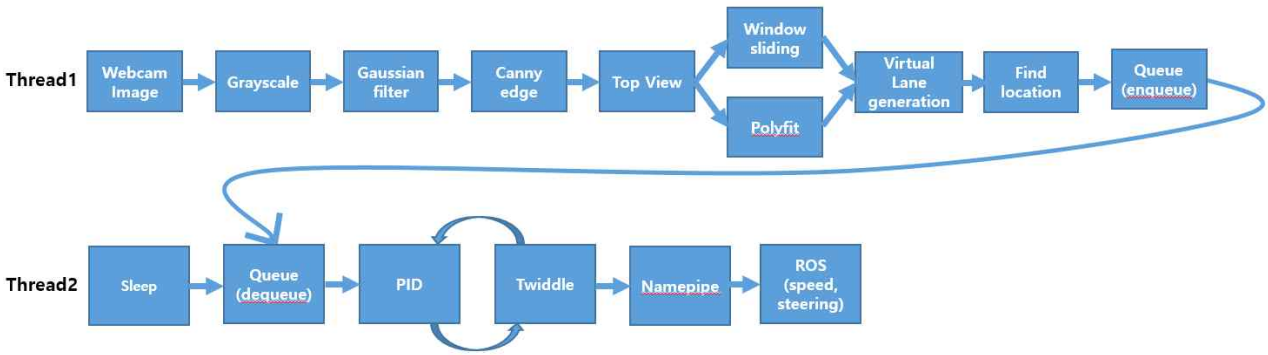
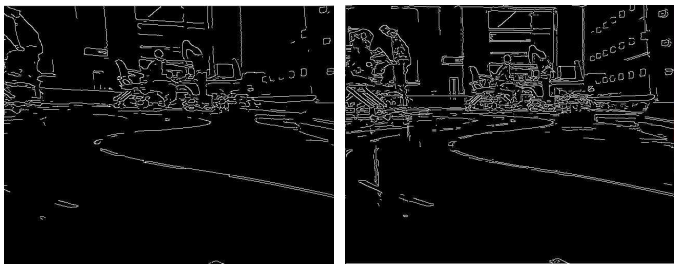


그림 2. 자율주행 자동차 시스템 구조도

grayscale로 변경한다. grayscale이 적용된 영상에서 잡음을 처리하기 위해 gaussian filter를 이용하여 blur 처리를 한다. blur 처리된 영상을 canny edge detection을 사용하여 화면의 edge들을 검출한다.



(a) (b)

그림 3. (a) 가우시안 필터 사용한 canny edge
(b) 가우시안 필터 미사용한 canny edge

그림 3.(b)의 좌측 중앙을 보면 빛에 반사된 잡음이 생기는 것을 확인할 수 있지만 가우시안 필터를 사용한 그림 3.(a)에서는 잡음이 없어진 것을 확인할 수 있다.

전방의 영상에는 도로의 차선 이외에도 수많은 영상이 잡힌다. 불필요한 영상 제거와 정확한 방향성 파악을 위해 영상을 수직에서 보는 것처럼 변경한다. 이를 위해 warper(top view 또는 bird eye view)를 사용한다.

그림 4.(a)는 warper를 사용하기 이전의 영상이다. warper를 사용하고 난 뒤 그림 4.(a)의 사각형 안의 화면이 그림 4.(b)와 같이 바뀐 것을 확인할 수 있다. 사각형 이외의 부분을 제거함과 동시에 사각형 내부 부분의 영상을 강제로 왜곡함으로써 마치 영상을 위에서 본 것처럼 만든다. 이 기법을 이용하여 거리와 좌표를 확인할 수 있다.

warper가 적용된 후의 영상에서 차선을 인식하기 위해 차선 인식 시작점을 이미지 구간 분할 기법으로 검출한다. canny edge detection이 적용된 이진 영상에서 0이 아닌 점을 검출하여 그림 5 아래 부분의 3개 영역과 같이 이미지 구간 분할 영역 내에 일정 수(threshold) 이상의 점이 잡히면 선이 들어온 것으로 인식한다. 미세한 잡음을 선으로 인식할 수도

(a) (b)

그림 4. (a) warper 사용 이전 (b) warper 사용 이후

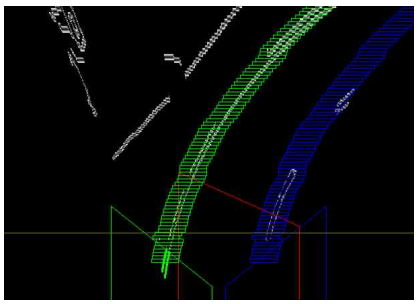


그림 5. 좌측 선을 기준으로 인식

있으므로 첫 번째 이미지 구간 분할 영역 안에서 잡힌 모든 점의 x, y의 각 평균값을 시작점으로 인식한다.

그림 4.(b)를 보면 모형 자동차 트랙을 기준으로 왼쪽이 실선 차선, 오른쪽이 점선 차선으로 되어 있으므로 직진이나 우회전을 할 때는 그림 4.(b)나 그림 5처럼 실선 차선을 항상 볼 수 있다. 그림 5를 보면 실선 차선의 시작점(왼쪽 아래 사각형)을 기준으로 일정 폭 만큼 x, y 구간을 만들어 직사각형(rectangle window)을 생성한다. 이 구간 안에 있는 x의 평균값을 다음 직사각형(rectangle window + 1)의 시작 x좌표로 설정하고 y축은 일정 폭 만큼 올려 sliding window 기법으로 선을 끝까지 인식할 때까지 반복한다. 점선 차선은 실선 차선을 기준으로 도로의 폭만큼 x축으로 이동하여 가상의 오른쪽 선을 그어 인식한다.

그림 6과 같이 좌회전을 하게 되는 경우 오른쪽 점선 차선만 보이게 되는데, 이 경우에는 점선이기 때문에 시작점이 이

미지 구간 분할 영역(우측 아래 사각형) 내에 들어오지 않을 수도 있다. 따라서 이때에는 새로운 이미지 구간 분할 영역(중앙 아래 사각형)을 적용하여 해당 구간 내의 2차 곡선 다항식(2 degree polynomial curve fitting)을 이용하여 선을 인식하고 도로의 폭만큼 −x축으로 이동하여 가상의 왼쪽 선을 그어 인식한다. 또 이 경우에는 선을 인식하는 시작점(y좌표)을 알 수 없으므로 가장 최근의 오른쪽 시작점을 파라미터로 넘겨받아 해당 지점을 시작점으로 인식한다.

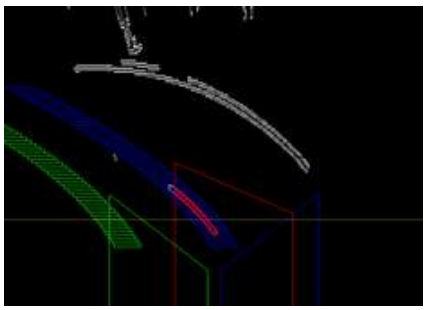


그림 6. 우측 선을 기준으로 인식

3.2. 모형자동차 제어

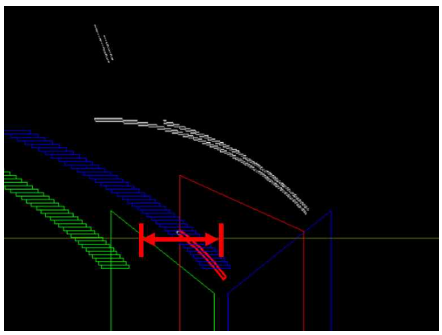


그림 7. 오류 값 검출

그림 7과 같이 자동차의 현재 위치와 도로의 중앙 위치의 차이를 오류 값으로 검출하여 차가 항상 도로의 중앙에 위치할 수 있도록 PID 알고리즘을 통해 제어한다.

모형자동차가 트랙을 벗어나지 않는다고 가정한 후 영상 처리를 진행하였기 때문에 트랙에서 나가면 다시 돌아올 수 없다. 따라서 일차적으로 오픈소스 자율주행 시뮬레이터(TORCS)를 이용하여 PID를 어느 정도 제어한다. 이차적으로 트랙 가운데로 움직이는 영상을 모형 자동차를 이용하여 수동으로 직접 촬영한 후 모형 자동차를 이용하여 바퀴가 제대로 움직이는지 확인하고 제어한다. 이후 실제 트랙에서 twiddle 알고리즘[8]을 이용하여 PID를 최적화한다.

화면에서 처리하는 영상은 자동차가 그림 1의 gap만큼 이동한 n초 후의 시간이 된다. 이를 제어하기 위하여 2개의 쓰레드를 생성한다. 하나는 영상처리와 오류 값을 입력하기 위한 buffer queue의 enqueue를 위한 쓰레드이며 다른 하나는 n초 후 오류 값을 출력을 위한 buffer queue의 dequeue 및 자동차 제어 프로세서로 데이터 전송을 위한 IPC(Namedpipe) 처리를 하는 쓰레드이다. 두 번째 쓰레드에서 n초 동안 대기를 한 후 데이터를 전송하게 되므로 자동차 제어 프로세스에서는 실시간인 것처럼 인지하게 된다.

4. 결론 및 향후 연구방향

급커브 구간에서 차선이 하나가 손실되는 경우 하나만 인식하고 도로의 폭을 고려하여 가상의 선을 만들어 줌으로써 도로의 중앙값 및 손실된 차선을 예측할 수 있다. 하지만 이 알고리즘은 왼쪽이 실선 차선이라는 가정이 들어간 상황이다. 따라서 양쪽 차선이 실선 차선인지 점선 차선인지 먼저 확인할 수 있는 알고리즘을 생성하여야 더욱 정밀한 자율주행을 할 수 있을 것이다. 그뿐만 아니라 장애물이나 보다 많은 차선의 손실 등을 고려할 때 Lidar, GPS, 기계학습 등의 기술을 통하여 더욱 정밀한 제어가 필요하다.

참 고 문 헌

[1] 오일석, (2014). “컴퓨터 비전 Computer Vision”, 한빛아카데미

[2] H. Wu, W. Su and Z. Liu, "PID controllers: Design and tuning methods," 2014 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, 2014, pp. 808−813.

[3] Ballard, D.H., “Generalizing the Hough transform to detect arbitrary patterns,” Pattern Recognition, 13, 2 (1981), pp. 111−122.

[4] P. Bao, L. Zhang, and X. Wu, “Canny edge detection enhancement by scale multiplication,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 27, no. 9, pp. 1485-1490, Sep. 2005.

[5] G. Ramu and S. B. G. T. Babu, "Image forgery detection for high resolution images using SIFT and RANSAC algorithm," 2017 2nd International Conference on Communication and Electronics Systems (ICCES), Coimbatore, 2017, pp. 850−854.

[6] E. Elboher and M. Werman, "Efficient and accurate Gaussian image filtering using running sums," 2012 12th International Conference on Intelligent Systems Design and Applications (ISDA), Kochi, 2012, pp. 897−902.

[7] C.−C. Lin and M.−S. Wang, “A vision based top−view transformation model for a vehicle parking assistant,” Sensors, vol. 12, no. 4, pp. 4431-4446, 2012

[8] CHASE, P J. Algorithm 382, Combinations of M out of N objects. Comm. ACM 15, 6 (June 1970), 368 See also Remark on Algorithm 382, p. 376.