논문 14-39C-10-07

The Journal of Korea Information and Communications Society '14-10 Vol.39C No.10
http://dx.doi.org/10.7840/kics.2014.39C.10.948

옵티컬 플로우 분석을 통한 불법 유턴 차량 검지

송 창 호◆, 이 재 성°

Detection of Illegal U-turn Vehicles by Optical Flow Analysis

Chang-ho Song◆, Jaesung Lee°

요 약

오늘날 지능형 영상 검지기 시스템(Intelligent Vehicle Detection System)이 추구하는 방향은 기존 시스템의 교통 소통 정보 습득을 넘어서 교통정체, 사고 등과 같은 부정적인 요인을 줄이는 것이다. 본 논문에서는 도로 교통법규 위반 상황 중에서 가장 치명적인 사고를 유발 할 수 있는 불법 유턴 차량을 검지하는 알고리즘을 제안한다. 영상의 옵티컬 플로우 벡터(Optical Flow Vector)를 구하고 이 벡터가 불법 유턴 경로 상에 나타난다면 불법 유턴 차량에 의해 생긴 벡터일 확률이 높을 것이라는 점에 착안하여 연구를 진행했다. 옵티컬 플로우 벡터를 구하기 전에 연산량 절감을 위하여 코너(corner)와 같은 특징점을 선지정한 후 그 점들에 대해서만 추적하는 피라미드 루카스-카나데(pyramid Lucas-Kanade) 알고리즘을 사용했다. 이 알고리즘은 연산량이 매우 높기 때문에 먼저 컬러 정보와 진보된 확률적 허프 변환(progressive probabilistic hough transform)으로 중앙선을 검출하고 그 주위 영역에만 적용시켰다. 그리고 검출된 벡터들 중 불법 유턴 경로위의 벡터들을 선별하고 이 벡터들이 불법 유턴 차량에 의해 생긴 벡터들인지 확인하기 위해 신뢰도를 검증하여 불법 유턴 차량을 검지하였다. 최종적으로 알고리즘의 성능을 평가하기 위해 알고리즘별 처리시간을 측정하였으며 본 논문에서 제안한 알고리즘이 효율적임을 증명하였다.

Key Words : illegal U-turn vehicle, image processing, hough transform, harris corner, optical flow

ABSTRACT

Today, Intelligent Vehicle Detection System seeks to reduce the negative factors, such as accidents over to get the traffic information of existing system. This paper proposes detection algorithm for the illegal U-turn vehicles which can cause critical accident among violations of road traffic laws. We predicted that if calculated optical flow vectors were shown on the illegal U-turn path, they would be cause of the illegal U-turn vehicles. To reduce the high computational complexity, we use the algorithm of pyramid Lucas-Kanade. This algorithm only track the key-points likely corners. Because of the high computational complexity, we detect center lane first through the color information and progressive probabilistic hough transform and apply to the around of center lane. And then we select vectors on illegal U-turn path and calculate reliability to check whether vectors is cause of the illegal U-turn vehicles or not. Finally, In order to evaluate the algorithm, we calculate process time of the type of algorithm and prove that proposed algorithm is efficiently.

※ 본 연구의 일부는 미래창조과학부 및 정보통신산업진흥원의 IT융합고급인력과정지원사업의 연구 결과로 수행되었으며(NIPA-2014-H0401-14-1007) 나머지 일부는 2012년도 정부(교육과학기술부)의 재원으로 한국연구재단의 지원을 받아 수행된 기초연구사업임(No. 2012R1A1A1038515)

◆ First Author : Korea National University of Transportation Department of Electronic Engineering, schangao@gmail.com, 학생회원

° Corresponding Author : Korea National University of Transportation Department of Electronic Engineering, jaesung.lee@ut.ac.kr, 정회원

논문번호 : KICS2014-07-264, Received July 14, 2014; Revised September 15, 2014; Accepted September 15, 2014

Ⅰ. 서 론

오늘날 지능형 영상 검지기 시스템(Intelligent Vehicle Detection System)이 추구하는 방향은 기존의 교통 소통 정보 습득을 넘어서 교통 소통에 방해가 되는 요인을 자동으로 찾아내고 이를 제거하여 능동적으로 도로 소통 상황을 개선하는 것이다[1]. 그러기 위해서는 교통 법규 위반 행위가 철저히 단속되어져야 한다. 대표적인 예로는 안전거리미확보, 끼어들기, 교차로 꼬리 물기, 불법 유턴, 신호위반 등이 있다. 본 논문에서는 그 중 위반시 가장 치명적인 사고를 유발할 수 있는 불법 유턴 차량을 검지하는 알고리즘을 제안한다.

일반적으로 불법 유턴이라 하면 중앙선을 침범하여 유턴을 하는 경우를 말한다[2]. 따라서 도로 중앙 위에 고정된 카메라를 설치하고 불법 유턴 차량을 촬영 및 영상 처리(Image Processing)를 수행하여 차량의 움직임에 대해 분석하고, 불법 유턴의 움직임을 파악할 수 있는 알고리즘을 고안한다.

본 논문의 구성은 다음과 같다. 2장에서는 제안하는 불법 유턴 차량의 추적 알고리즘을 상세히 설명하고 3장에서는 제안하는 방법을 불법 유턴 차량 영상에 실시간으로 적용하여 검지가 잘 되는 지 실험 및 알고리즘 처리 시간에 대한 성능 분석을 시행한다. 최종적으로 4장에서 본 연구에 대한 결론을 내린다.

Ⅱ. 불법 유턴 차량 추적 알고리즘

불법 유턴은 그림 1과 같이 상행 하행을 구분하는 중앙선을 넘어 U 자 형태로 상행선에서 하행선으로 또는 하행선에서 상행선으로 선회하는 행위를 말한다. 본 논문에서는 영상 내 옵티컬 플로우 벡터(Optical Flow Vector)를 구하고 이 벡터가 불법 유턴 경로 상에 나타난다면 불법 유턴 차량에 의해 생긴 벡터일 확률이 높을 것이라는 점을 착안하여 연구를 진행했다. 그림 1과 같이 차량이 빨간색 경로로 움직인다면 차량의 움직임 벡터는 대다수가 검은색 화살표 방향으로 나타날 것이다.

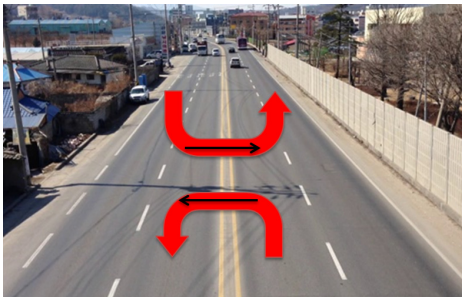


그림 1. 불법 유턴 차량 경로(빨간색)와 수평 방향 벡터(검은색)
Fig. 1. Illegal U-turn vehicle routine (red) and horizontal vector (black)

2.1 알고리즘 개요

먼저 옵티컬 플로우 벡터를 구하고, 이 움직임 벡터가 불법 유턴 경로에 나타나는지 확인해야 한다. 동영상내 옵티컬 플로우 벡터를 구하는 알고리즘은 대표적으로 두 가지로 나뉘는데, 밀집 옵티컬 플로우(dense optical flow)와 희소 옵티컬 플로우(sparse optical flow)가 있다. 밀집 옵티컬 플로우 알고리즘은 큰 움직임 추적에는 어려움이 있으며 특히 연산량이 매우 많아 실시간 추적에 부적합하다. 반면 희소 옵티컬 플로우는 이미지 피라미드(pyramid) 기법을 이용하면 큰 움직임 추적이 가능하고 코너(corner)와 같은 특징점을 지정한 후 그 점들에 대해서만 추적을 하면 되기 때문에 연산량이 상대적으로 적어진다. 본 논문에서는 실시간 영상처리라는 제약 조건에 따라 희소 옵티컬 플로우를 사용하여 벡터를 추출한다.

본 연구에서 제안하는 알고리즘의 흐름도를 그림 2에 나타내었다. 먼저 전처리 과정으로 영상에서 중앙선을 검출하고 이를 중심으로 주변에 관심영역을 설정하게 된다. 이후 관심영역내에 있는 차량의 특징점들을 찾고 그 점들에 대한 움직임 벡터를 계산하여 특징점들이 어디로 이동하는지 확인한다. 만약 불법 유턴경로로 벡터의 이동이 감지되면 그 신뢰성을 검증하여 불법유턴차량인지 아닌지 결정하게 된다.

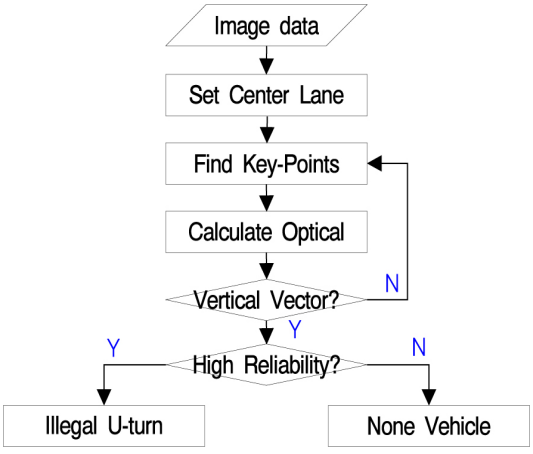


그림 2. 제안하는 알고리즘 흐름도
Fig. 2. Flow chart of the proposed algorithm

2.2 컬러모델 변환을 통한 중앙선 검출

불법 유턴 차량은 그림 1에서 볼 수 있듯이 중앙선을 반드시 침범한다. 이 점에 착안하여 중앙선 주위를 관심 영역(region of interest, ROI)으로 설정하기 위해 중앙선을 검출한다. 특정 객체의 검출을 위해 사용되는 여러 특징 중에서 컬러정보는 잡음이나 부분적인 가려짐에 강인하고 회전이나 크기 변화에도 덜 민감하며 계산속도가 빨라 객체 검출을 위한 매우 중요한 단서로 사용된다.[3]

영상에서 피부색이나 중앙선색 같은 특정 색상을 띄는 물체를 검출하기 위해 여러 컬러 모델이 사용되는데 대표적으로 RGB, HSV, CIE L*a*b* 등이 사용된다.[4] 중앙선은 노란색이기 때문에 R, G, B 성분을 이용한 RGB 컬러 모델은 중앙선 검출이 용이하지 않다. 반면 HSV 컬러모델은 색상(H) 정보로만 색상을 파악 할 수 있고 여기에 옅거나 진한 정도를 채도(S)로, 밝기 정도를 명도(V)로 값을 조정하기 때문에 인간의 색인지에 근접한 모델로 회색 도로 배경 속에서 노란색 중앙선을 검출하기에 적합하다.

일반적으로 RGB 컬러모델로 되어 있는 영상 데이터는 식 1과 같은 비선형 수식을 이용하여 HSV 컬러 모델로 변환하게 된다.

$$H = \cos^{-1}\frac{\frac{1}{2}(R-G)+(R-B)}{\sqrt{(R-G)^2+(R-B)(G-B)}}$$
$$S = 1-\frac{3\min(R,G,B)}{R+G+B} \quad (1)$$
$$V = \frac{R+G+B}{3}$$

그림 4의 왼쪽 그림은 그림 3의 왼쪽 그림을 RGB-to-HSV 변환한 결과이다. 색상(H) 값만으로 중앙선 색상을 어느 정도 검출할 수 있지만 좀 더 정확한 검출을 위해 채도(S), 명도(V) 값들도 고려한다[5]. 그림 4의 오른쪽 그림은 각 픽셀의 색상(H), 채도(S), 명도(V) 값들에 대한 임계치(double - thresholding)에


그림 3. 도로의 중앙선
Fig. 3. Center lane of road

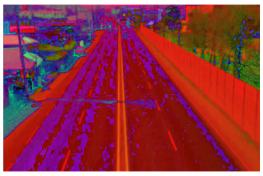
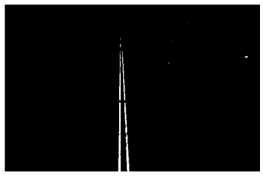



그림 4. RGB-to-HSV 변환 결과와 중앙선 검출
Fig. 4. RGB-to-HSV conversion and detection of center lane

의한 이진화 결과이다(식 2). H, S, V 각각에 대한 임계치 조건을 모두 만족하는 경우에만 해당 픽셀(I(x, y))을 이진이미지의 픽셀 값 '1' 로 변환하고 그 외의 경우는 모두 '0' 으로 변환한다.

$$I(x,y) = \begin{cases} 1 & \text{if}\,(0.06 < H < 0.3) \cap \\ & (0.22 < S < 1.0) \cap \\ & (0.84 < V < 1.0) \\ 0 & otherwise \end{cases} \quad (2)$$

2.3 중앙선 주위 관심영역(ROI)설정

2.2절에서 검출한 중앙선 주위를 관심 영역(region of interest, ROI)으로 설정하고 영상 처리 대상 영역을 축소하여 다음 단계에서 수행될 알고리즘들의 방대한 연산량 부담을 줄인다. 관심 영역의 설정 방법은 다음과 같다.

중앙선(직선)이 검출된 이진 이미지에 대해 일반적으로는 직선 허프 변환(line hough transform)을 이용하여 직선을 찾는다. 그러나 시간대에 따라 중앙선이외의 주변 환경에서도 중앙선과 비슷한 색이 검출될 수 있고 그것을 중앙선으로 잘못 인식하는 현상이 발생할 수 있다. 또한 직선 허프 변환 자체가 삼각함수, 곱셈, 나눗셈 등과 같은 오버헤드가 큰 연산들의 반복적 수행이 필요하고, 변수 공간을 양자화하는 해상도와 검출 정확성 사이의 트레이드-오프(trade-off)를 고려해야하기 때문에 실시간 성능을 보장하기 어렵다는 단점이 있다.[6] 이러한 허프 변환의 단점을 보완한 진보된 확률적 허프 변환(PPHT, progressive probabilistic hough transform)[7]을 사용했다. 진보된 확률적 허프 변환은 기존의 직선의 방정식을 구하는 방법에서 모든 픽셀에 대해 투표(voting)을 하는 것이 아니라 일부 선택된 픽셀만 고려한다. 즉, 특정 위치의 직선의 방정식에 대한 파라미터가 일정 수 이상 투표 되면 더 이상 그 지점의 직선의 방정식에 대해서는 계산이 진행 되지 않아 연산량을 크게 단축시킬 수 있다. 또한 계산과정에서 직선의 끝과 끝을 파악하고 길이를 계산하여 길이에 따른 직선을 선택할 수도 있다.

중앙선은 비교적 긴 편이기 때문에 일정 길이 이상

을 선택해서 오검출률을 낮출 수 있다. 예를 들어 이미지의 세로 사이즈가 600 픽셀이라면 길이가 300 픽셀이상(2분의 1이상)인 직선들만 선택하게 하면 빠르고 좋은 결과를 얻을 수 있다. 그림 5의 왼쪽은 이진화된 중앙선 이미지에 확률적 허프 변환을 적용시킨 결과이다.

그림 5의 왼쪽그림에서 검출된 두 직선들에 대해 평균 중앙선(가운데 하나의 기준선)을 계산한다. 계산된 평균 중앙선을 기준으로 하여 좌우로 일정한 폭을 정하여 그 폭 안쪽 영역을 관심 영역으로 설정한다. 관심 영역의 폭은 촬영환경에 따라 계산(calibration)하여 결정하되 자동으로 설정을 하고자 하는 경우는 검출된 두 중앙선 사이의 최대 간격(그림의 하단 부분)의 3배의 크기 정도로 설정하면 적당하다. 이렇게 결정된 관심영역은 카메라 떨림 등으로 촬영 각도가 바뀔 상황을 대비하여 일정 프레임마다 간헐적으로 다시 세팅한다.

그림 5. 이진화된 중앙선 이미지의 진보된 확률적 허프 변환
Fig. 5. Progressive probabilistic hough transform of binarized Center Lane image

2.4 추적하기 좋은 특징점(Key-Points) 검출

특징점의 효과적인 추적을 위해 차량의 어떤 픽셀점을 특징점으로 사용할 것인지 결정하는 일은 매우 중요하다. 만약 그림 6의 맨 왼쪽과 같이 flat 영역을 추적한다고 하면 다음 프레임에서 동일한 flat 영역을 추적하는 것은 쉽지 않을 것이다. edge 부분도 flat 영역보다는 낫지만 여전히 혼동의 여지가 많다. 반면, corner 점은 위치(rotation), 크기(scale), 밝기 변화(illumination variation), 잡음(noise)에 강인하며 다음 프레임에서도 추적이 용이하다.[8] 대부분의 특징점 추출 알고리즘들은 이러한 코너(corner) 점 검출을 바탕으로 하고 있다.

대표적인 코너 검출 알고리즘으로는 해리스(Harris)가 제안한 알고리즘이 있다.[9] 이 알고리즘은 모라비츠(Moravec)가 제안한 알고리즘[10]을 기반으로 하는데 그 원리는 영상의 각 픽셀의 상하좌우대각의 4방향(그림 6의 화살표)의 미분변화량(절대 값)의 합(sum of squared deviations, SSD)을 구해 그 값이 상대적

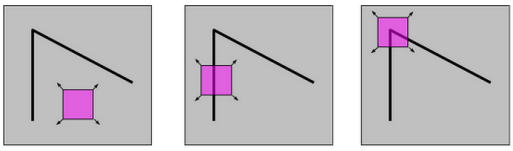


그림 6. flat, edge, corner 영역
Fig. 6. Region of flat, edge, and corner

으로 큰 값이면 코너로 인지하는 것이다.

그림 7은 도로 이미지 전 영역에서, 그림 8은 ROI 영역에서 해리스 코너 알고리즘을 적용하여 코너점을 (초록색) 찾은 영상이다. 건물, 나무, 차량 등 각진 객체는 코너 점들이 검출되었지만 flat 영역인 도로 부분이나 차량의 면 부분에는 코너가 검출되지 않은 것을 확인 할 수 있다.

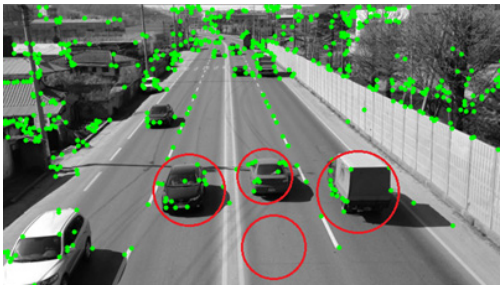


그림 7. 이미지 전 영역에서의 코너 검출
Fig. 7. Corner detection in all area

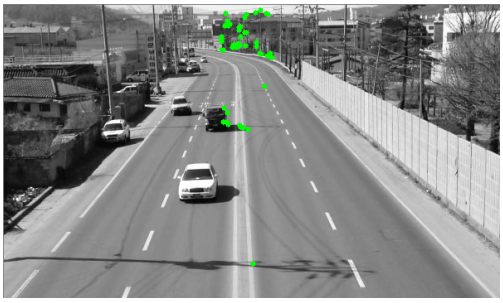


그림 8. 관심 영역(ROI)에서의 코너 검출
Fig. 8. Corner detection in region of interest(ROI)

2.5 움직임 벡터 계산과 추적

본 단계에서는 차량의 코너 점들이 프레임간 어떤 움직임을 갖는지 파악하기 위해 옵티컬 플로우 벡터를 계산하고 추적한다.

동영상에서 움직임이 발생하면 픽셀들의 밝기 변화가 나타나는데 이러한 밝기의 이동을 옵티컬 플로우(optical flow)라고 한다. 이러한 밝기의 움직임 즉, 옵

티컬 플로우를 계산하여 움직임 벡터(motion vector)를 구하고 이를 분석하면 특정 물체를 추적할 수 있다. 옵티컬 플로우를 계산하는 알고리즘은 크게 밀집 옵티컬 플로우(dense optical flow)와 희소 옵티컬 플로우(sparse optical flow) 두 가지로 분류 할 수 있다.

밀집 옵티컬 플로우는 영상 내부의 모든 픽셀에 대해 옵티컬 플로우를 계산하는 알고리즘으로 2.4절에서 언급한 Flat 역영 내의 점에 대한 움직임을 계산할 경우 연산량 낭비가 매우 심하게 된다. 반면 희소 옵티컬 플로우는 영상의 일부 픽셀(특징점)에서만 옵티컬 플로우를 계산하는 알고리즘으로, 특징이 많이 나타나지 않는 영역에서는 계산을 수행하지 않으므로 연산량을 대폭 절감할 수 있게 된다.

본 논문에서는 옵티컬 플로우 알고리즘의 평과 결과[11]중에 랭크(rank)가 높은 알고리즘인 피라미드 루카스-카나데(pyramid Lucas-Kanade, pyramid L-K) 알고리즘을 사용한다.

원래의 루카스 카나데 알고리즘은 설정한 윈도우보다 큰 움직임이 발생하였을 경우에는 움직임을 계산하지 못하는 단점이 있다. 또한 윈도우 크기를 너무 크게 하면 연산량이 제곱수로 증가하고 정확도는 떨어지게 된다. 이러한 문제들을 해결하기 위해 Bouguet는 이미지 피라미드(pyramid)기법[12,13]을 사용하여 작은 윈도우로도 큰 움직임을 계산할 수 있도록 한 것이다.

2.4절에서 검출한 코너 포인트들에 대하여 Bouguet의 방식을 적용하여 옵티컬 플로우를 계산한다. 이 계산을 통해 그 코너 점들의 이동 상황을 감지하고 이때 이상 움직임이 검지되면(즉, 특정 방향과 일정 크기 이상의 벡터 조건을 만족하면) 추적 모드로 돌입한다.

일단 추적 모드로 돌입하면 활동성향이 멈출 때 까지 연속적으로 옵티컬 플로우를 계산해 검출된 코너 점들의 위치를 메모리에 저장시킨다. 그림 9는 그림 7의 이미지 전 영역에서, 그림 10은 그림 8의 ROI 영역에서, 검출된 코너에 대해 옵티컬 플로우를 계산하

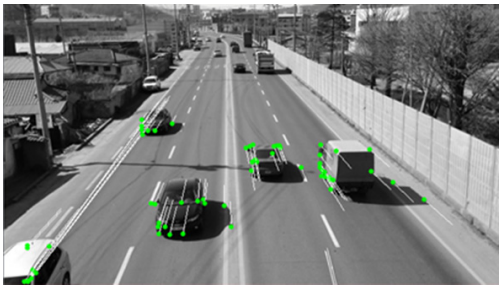


그림 9. 코너에 대한 옵티컬 플로우 계산 및 추적(1)
Fig. 9. Optical flow calculation and traking of corner(1)

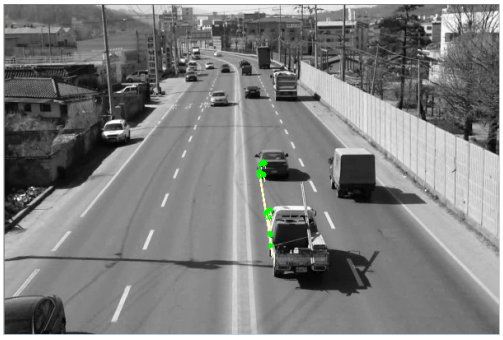


그림 10. 코너에 대한 옵티컬 플로우 계산 및 추적(2)
Fig. 10. Optical flow calculation and traking of corner(2)

고 프레임간 움직임이 (방향에 상관없이) 일정 길이 이상이면 추적한 결과이다. 메모리에 저장된 점들을 하나하나 이으면 코너가 어떻게 움직였는지 정확히 알 수 있다.

참고로 SURF, SIFT 와 같은 알고리즘을 사용하면 옵티컬 플로우를 사용하지 않고도 특징점들을 추적할 수 있지만 이러한 알고리즘들은 로고, 라벨, 상표, 만화 캐릭터 등 자연적인 이미지가 아닌 인공적인 이미지 객체들의 경우에만 추적이 잘된다는 점과 연산량이 매우 많다는 점 때문에 본 논문에서는 고려하지 않았다.

2.6 불법 유턴 경로위의 움직임 벡터 추적

불법 유턴 차량은 2장 서두에서 언급한 바와 같이 중앙선에 대해 수직한 벡터를 남긴다. 이 점에 착안하여 움직임 벡터가 중앙선에 수직한 벡터이면 추적을 시작하고 그렇지 않다면 추적을 하지 않는다.

그림 11은 관심영역(ROI)에서 코너점 P1, P2가 다음 프레임에서 P1', P2'으로 이동했을 때 혹은 그 반대로 이동했을 때 그 움직임 벡터를 나타낸 것이다. 코너점 P1은 상하(y축) 이동은 거의 없고 좌우(x축)으

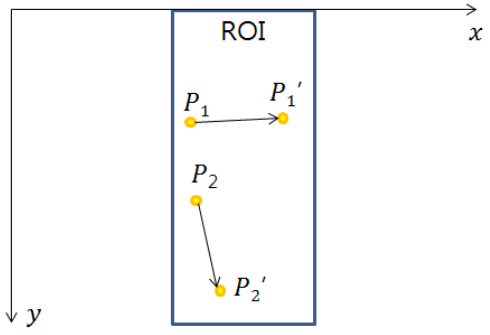


그림 11. ROI 영역에서 코너점들의 이동
Fig. 11. Movement of the corner points in region of interest

로 이동했다. 이는 중앙선을 가로질러 침범한 불법 유턴한 차량의 코너점일 확률이 높다. 반면, 점 P2는 좌우(x축) 움직임이 적고 상하(y축) 움직임이 크다. 이는 불법 유턴 차량과는 관련 없는 평범한 차량일 확률이 높다. 따라서 점 P1에 대해서 옵티컬 플로우를 계산하고 그 움직임이 P1'과 같이 움직였다면 추적을 시도하고, P2' 같이 움직였다면 추적을 하지 않는다. 이에 대한 의사코드를 표 1에 나타내었다.

isMove(Corner)함수는 코너점이 움직임이 있는지 없는지를 판단하는 함수이다. status는 2.5절에서 계산한 옵티컬 플로우 계산 결과가 좋지 않아 추적에 실패했을 경우 false 가 되는 변수이다.

selectVector(Corner)함수는 그림 1에서 수평방향 벡터와 같은 불법 유턴 경로로 지나간 움직임 벡터를 선택하는 함수이다. 코너를 매개변수로 받아 isMove 함수를 이용해 코너가 움직였는지 확인하고, 움직였다는 결과가 나오면 x축으로 최소 픽셀 간격 1보다 큰 크기로 움직이고 y축으로 yMin값 보다 작게 움직였을 때 코너를 메모리에 저장시킨다. yMin값은 코너의 y좌표 이동량의 조건(제한 값)으로 수평 방향 벡터를 선택하기 위해서는 |prevCorner.y - Corner.y| = 0 으로 해야 할 것이다. 하지만 실제 불법 유턴 경로로 지나간 벡터들이 정확히 수평 방향 벡터를 남기지 않고 그림 11의 P1, P1' 과 같이 약간 기울어진 형태의 벡터를 남긴다. 따라서 $\pm\alpha$ 값을 주어 여유를 준다. 본 논문에서는 실험값으로 5픽셀 간격인 '5' 로 설정했다. 이렇게 불법 유턴 경로로 이동하는 코너점을 선택하고 추적한 결과를 그림 12에 나타내었다.

표 1. 움직임 벡터를 선택하기 위한 의사코드
Table 1. Pseudocode for selecting a motion vector

```
isMove(Corner)
 if (status and (|prevCorner.x - Corner.x|
                + |prevCorner.y - Corner.y| > d) )
    then return true;
 else return false;
end isMove()
selectVector(Corner)
 num ← 0;
 if (isMove(Corner))
   then if (|prevCorner.x - Corner.x| > 1
         and |prevCorner.y - Corner.y| < yMin)
           then memory[num+1] ← Corner;
end selectVector()
```

그림 12. 불법 유턴 차량과 불법 유턴 경로 상의 움직임벡터
Fig. 12. Illegal U-turn vehicle and motion vectors on illegal U-turn path

Ⅲ. 불법 유턴 차량 검지 실험 및 성능 분석

본 절에서는 2장에서 검출한 벡터가 불법 유턴 차량에 의해 생긴 것인지 아닌지 판단하여 최종적으로 불법 유턴 차량을 검지한다. 그림 13은 단순히 픽셀 정보 유사성에 의해 검출된 벡터들이다. 그림 8의 도로에 전봇대에 의해 생긴 그림자에 코너 검출이 된 것을 확인 수 있는데, 이 코너에 대해 옵티컬 플로우를 계산이 수행되면 수 프레임이 지나고 차량에 의해 생긴 그림자와 겹치는 순간 코너가 좌우로 움직인 것으로 판단되어 중앙선에 수직하게(불법 유턴 경로로) 움직인 벡터가 검출 되는 경우가 있다. 이러한 벡터들과 그림 12와 같이 진정 불법 유턴 차량에 의해 검출된 벡터들 간의 구분을 위해 신뢰도(reliability)를 측정한다.

표 2는 신뢰도를 구하기 위한 의사코드이다. 신뢰도는 옵티컬 플로우 계산에 의해 저장된 점들의 무게중심(Center)을 구하는 과정이다. 한 점과 저장된 다른 한 점과의 거리(dist)를 계산하여 이 거리가 일정 크기(본 연구에서는 ROI영역의 폭 사용)보다 작다면 같은 집단이라고 보고 두 점 사이의 가운데를 새로운 무게중심점으로 설정하며 본 과정을 반복하면서 신뢰도(Reliability)를 증가 시킨다. 만약 두 점 사이의 거리가 관심영역의 폭보다 크다면 새로운 집단이라고 보고 다른 집단의 무게중심점으로 설정해야 한다. 이

그림 13. 픽셀 정보 유사성에 의해 검출된 벡터들
Fig. 13. Detected vectors by pixel information similarity

표 2. 신뢰도를 계산하기 위한 함수(의사코드)
Table 2. Functions to calculate a reliability (pseudocode)

```
getCenter(Points)
  Reliability, num ← 0;
  Center ← Points[0];
  while(num = numofPoints) do {
    dist ← |Center.x - Points[num].x|
           + |Center.y - Points[num].y|;
    if (dist < ROIwidth) then {
      Center ← (Center + Points[num])/2;
      Reliability ← Reliability + 1;
    }
    num ← num + 1;
  }
  return Reliability;
end getCenter()
```

렇게 모든 점들과 비교해서 일정 신뢰도 수준을 넘으면 불법 유턴 차량에 의해 생긴 벡터 집단이라고 판단한다. 그림 14는 신뢰도 측정 알고리즘을 적용한 결과이다. 신뢰도가 높아 불법 유턴 차량으로 판단되어 무게중심점을 빨간점으로 표시했다.

본 논문에서 제안하는 불법 유턴 차량 검지 알고리즘의 성능을 평가는 Intel(R) Core(TM) i5-3470 CPU @ 3.20GHz 3.20GHz, 메모리 4GB의 하드웨어에서, MS Windows7 운영체제에서 Visual Studio 2010을 이용하여 수행했다.

실험 데이터는 한국교통대 근처 도로에서 촬영한 영상으로, 640 * 480 해상도의 1분 25초(약 2500개의 프레임)길이를 가진 영상이다.

실험은 밀집 옵티컬 플로우 알고리즘인 Farneback 알고리즘과 희소 옵티컬 플로우 알고리즘인 Pyraimd Lucas-Kanade 알고리즘의 처리시간을 비교했다. Farneback 알고리즘은 이미지 전 영역에 대해 처리

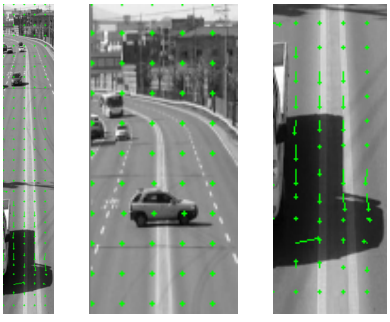
그림 14. 불법 유턴 차량 검지
Fig. 14. Detection of illegal U-turn vehicle

표 3. 지역에 따른 알고리즘의 처리 시간
Table 3. The processing time of the algorithm according to the area

	All area (640 * 480)	**set ROI (78 * 480)**
Farneback	116.43ms	22.73ms
P. L-K	24.12ms	6.43ms

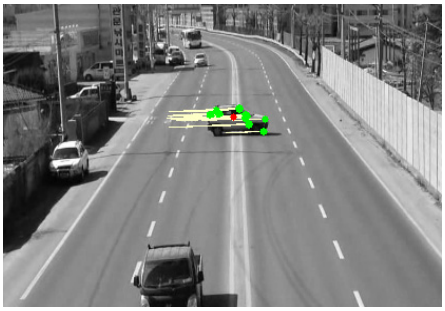
할 경우 120ms에 가까운 처리 시간이 걸려 실시간 처리에 부적합함을 알 수 있었다. 관심 영역을 정하고 처리할 경우 밀집 옵티컬 플로우 알고리즘으로도 실시간 처리는 가능했지만 카메라에서 멀리 떨어진 차량의 움직임은 픽셀 변화가 거의 없어 움직임을 파악하기 어려웠다. 이를 해결하기 위해서는 해상도를 높이거나 피라미드 영상의 단계를 높여야 하는데, 이렇게 하면 연산량이 더욱 증가하여 실시간 처리 요건을 갖추기 어렵게 된다. 또한 Flat 영역의 오차로 불법 유턴 경로로 움직이는 벡터를 선별하기 어려운 문제점도 발견되었다. 그림 15는 관심영역(78 * 480) (a) 에서 불법 유턴 차량부분을 확대한 사진 (b) 인데, 움직임이 매우 작아 벡터의 크기가 매우 작은 것을 확인할 수 있었다. 또한. (c)에서는 그림자 영역의 Flat한 성질 때문에 오차가 많이 난다. 따라서 밀집 옵티컬 플로우를 이용하여 불법 유턴 차량을 검지하는 것은 연산량, 성능 모든 측면에서 문제가 있음을 알 수 있었다.

본 논문에서 사용한 Pyraimd Lucas-Kanade 알고리즘은 이미지 전 지역에 대해 적용시켜도 꽤 좋은 성능(24.12ms)의 결과를 확인할 수 있다. 관심 영역을 설정했을 때는 6.43ms로 상대적으로 매우 빠른 처리 속도를 보였으며, 불법 유턴 경로로 움직이는 벡터를 선별하고 신뢰도계산 및 불법 유턴 차량 검지에 있어서도 오검지율이 없었다. 그림 16은 이미지 전 영역에



(a) (b) (c)

그림 15. Farneback 알고리즘을 적용한 결과
Fig. 15. Result of applying the algorithm Farneback



(a)

(b)

그림 16. Pyraimd Lucas-Kanade 알고리즘을 적용한 결과
Fig. 16. Result of applying the algorithm Pyraimd Lucas-Kanade

서 적용한 결과(a)와 관심 영역을 설정하고 적용한 결과(b)를 나타낸다.

Ⅳ. 결 론

본 논문에서는 도로에서 불법 유턴하는 차량을 자동 검지는 알고리즘을 제안하였다. 불법 유턴 차량은 그 경로위에 수평방향의 움직임 벡터를 남긴다. 이를 인지하기 위해 먼저 코너점들을 검출하고, 이 코너점에 대해 희소 옵티컬 플로우(Sparse Optical Flow)알고리즘을 사용하여 실시간으로 불법 차량을 잡아 낼 수 있도록 하였다. 만일 알고리즘을 이미지 전 영역에 적용시킨다면 연산량이 기하급수적으로 증가하여 실시간 검출이 불가능하기 때문에 연산량 절감을 위하여 전처리과정으로 관심영역을 설정했다. HSV컬러모델을 통해 먼저 중앙선 주변을 관심영역으로 설정하고 그 영역내에서 움직이는 차량의 특징점을 찾고 그 특징점에 대해서만 움직임 벡터를 검출하였다. 검출된 움직임 벡터들 중 불법 유턴 경로위의 벡터들을 저장하고, 이 벡터들이 불법 유턴 차량에 의해 생긴 것인지 판단하기 위한 신뢰도를 계산했으며 신뢰도가 높은 벡터 집단이 불법 유턴 차량에 의해 생긴 벡터 집단임을 확인했다. 최종적으로 알고리즘의 성능을 평가하기 위해 알고리즘별 처리시간을 계산했으며 본 논문에서 제안한 알고리즘이 효율적임을 증명하였다.

References

[1] J. Lee, B. K. Kim, and S. H. Kim, “A study on the application of the interrupted traffic flow incident detection algorithm using fixed detector,” *J. Korean Soc. Civil Eng.,* vol. 4, pp. 33-36, Oct. 2000.

[2] *Act on Special Cases concerning the Settlement of Traffic Accidents*, Korea Law Service Center.

[3] B.-C. Ko, J.-Y. Nam, and J. Y. Kwak, “Object tracking using particle filters in moving camera,” *J. KICS*, vol. 37, no. 5, pp. 375-387, May 2012.

[4] Y.-E. An and J.-A. Park, “Color correlogram using combined RGB and HSV color spaces for image retrieval,” *J. KICS*, vol. 32, no. 5, pp. 513-519, May 2007.

[5] L. N. P. Boggavarapu, et al. “A robust multi color lane marking detection approach for Indian scenario,” *Int. J. Advanced Comput. Sci. & Appl.*, vol. 2, no. 5, pp. 71-75, May 2011.

[6] J.-R. Lee, K. Bae, and B. Moon, “A hardware architecture of hough transform using an improved voting scheme,” *J. KICS*, vol. 9, no. 38, pp. 773-781, Sept. 2013.

[7] J. Matas, C. Galambos, and J. Kittler, “Robust detection of lines using the progressive probabilistic hough transform,” *Computer Vision Image Understanding*, vol. 78, no. 1, pp. 119-137, Apr. 2000.

[8] C. Schmid, R. Mohr, and C. Bauckhage, “Evaluation of interest point detectors,” *Int. J. Computer Vision*, vol. 37, no. 2, pp. 151-172, Jun. 2000.

[9] C. Harris and M. J. Stephens, “A combined corner and edge detector,” *Alvey Vision Conf.*, pp. 147-152, Manchester, United Kingdom, Sept. 1988.

[10] H. Moravec, *Obstacle avoidance and*

navigation in the real world by a seeing robot rover, Stanford Univ. CA Dept. Comput. Sci., No. STAN-CS-80-813, 1980.

[11] Baker, Simon, et al., "A database and evaluation methodology for optical flow," *Int. J. Computer Vision*, vol. 92, no. 1, pp. 1-31, Mar. 2011.

[12] J.-Y. Bouguet, *Pyramidal implementation of the Lucas Kanade feature tracker description of the algorithm*, Retrieved June, 15, 2014, from http://robots.stanford.edu/cs223b04/algo_tracking.pdf.

[13] C. Ha, C. Choi, and J. Jeong, "Contrast enhancement algorithm using singular value decomposition and image pyramid," *J. KICS*, vol. 38, no. 11, pp. 928-937, Nov. 2013.

[14] G. Farneback, "Two-frame motion estimation based on polynomial expansion," *Lecture Notes in Comput. Sci.*, vol. 2749, pp. 363-370, Jun. 2003.

송 창 호 (Chang-ho Song)

2008년 3월~현재 : 한국교통대학교 전자공학과 학사 재학

<관심분야> System-on-Chip, Computer Vision, Machine Learning, Pattern Recognition

이 재 성 (Jaesung Lee)

2001년 2월~2011년 8월 : 한국전자통신연구원 선임연구원

2003년 3월~2008년 8월 : 서울대학교 전기컴퓨터공학부 박사 졸업

2011년 9월~2013년 9월 : 한국교통대학교 전자공학과 조교수

2013년 10월~현재 : 한국교통대학교 전자공학과 부교수

<관심분야> VLSI, System-on-Chip, Image Processing, Computer Vision