Adaboost학습알고리듬과 선형Kalman filter를 이용한 보행자 검출시스템 개발

*권태현 **위승우 **정제창

*,***한양대학교 융합전자공학부 **한양대학교 전자컴퓨터통신공학과

*gumil514@naver.com **slike0910@naver.com ***jjeong@hanyang.ac.kr

Pedestrian detection system development based on Adaboost algorithm and Linear Kalman filter

*Kwon, Tae-Hyun **Wee, Seungwoo ***Jeong, Jechang

*,*** Department of Electronic Engineering, Hanyang University

** Deptment of Electronics and Computer Engineering,. Hanyang University

요약

보행자 검출을 위한 기술이 많이 개발되고 있으며 HOG(Histograms of oriented)와 haar-like feature를 이용한 특징값 검출을 통해 보행자를 검출하는 방법들이 대표적이라 할 수 있다. 하지만 이 방법들은 보행자가 사물에 가려졌을 때 보행자를 검출하지 못한다는 단점이 있다. 이에 본 논문에서는 haar-like feature와 adaboost 학습알고리듬을 이용하여 보행자를 검출하고 kalman filter를 이용하여 보행자가 특정 사물에 가려지는 것 과 같은 occlusion 문제를 해결하여 보행자 검출 성능을 높이고자 하였다.

1. 서론

보행자 검출기술은 무인자동차를 비롯해 모니터링, 산업현장, 재난상황 등 여러 분야에 응용이 가능하다. 보행자 검출기술은 다양한 검출 방법들을 통해 이루어 질 수 있다. 특히 Viola와 Jones는 Haar-like feature를 사용하여 객체를 검출하는 방법을 제안했고[1], Dalal과 Triggs는 HOG(Histograms of oriented gradients)을 사용한 보행자 검출방법을 제안했다.[2] 그 밖에도 여러 가지 검출방법들이 존재한다.

하지만 기존의 보행자 검출 방법들은 보행자가 특정 사물에 가려지는 상황에서는 보행자를 검출 할 수 없다는 단점이 있다. 이에 본 논문에서는 Viola와 Jones가 객체검출을 위해 고안한 “Integral image”와 “haar-like feature”를 adaboost학습 알고리듬[3]과 함께 사용해 보행자를 검출하고 선형칼만필터[4]를 적용하여 보행자가 사물에 가려져 보이지 않을 때에도 보행자를 예측하여 검출할 수 있도록 하는 알고리듬을 제안한다.

본 논문의 순서는 다음과 같다. 2장에서 integral image와 haar-like feature, adaboost학습 알고리듬, 선형칼만필터에 대한 이론들을 설명하고 3장에서는 본 논문이 제안하는 보행자검출 알고리듬에 대해 설명한다. 4장에서는 실험을 통해 보행자검출성능을 확인하고 5장에서는 본 논문에 대한 결론과 향후 연구방향을 설정한다.

2. 이론적 배경

2.1 Haar-like feature

Haar-like feature는 Viola와 Jones에 의해 Haar 웨이블렛[5]에 기초하여 만들어진 객체검출시 특징값을 찾아내는 디지털 이미지 특징이다. 그림1은 대표적인 Haar-like feature들이다. Haar-like feature는 검출하려는 객체에 적용되어 백색영역의 픽셀들의 합과 검은색 영역의 픽셀들의 합간의 차를 계산하여 특징값을 얻어낸다. 예를 들어,사람의 눈 부분을 검출한다고 할 때 눈 부분만이 가지고 있는 특징 값을 먼저 찾아야 한다. 이때 얼굴 중에서 눈의 영역이 볼의 영역보다 어둡다는 것이 일반적인 관찰이다. 따라서 눈 부분에 그림 2와 같이 특정 haar-like feature를 적용한 다음 백색 부분의 픽셀 값들의 합에서 흑색부분의 픽셀 값들의 합을 빼면 눈 영역을 특징 짓는 값을 찾게 된다.

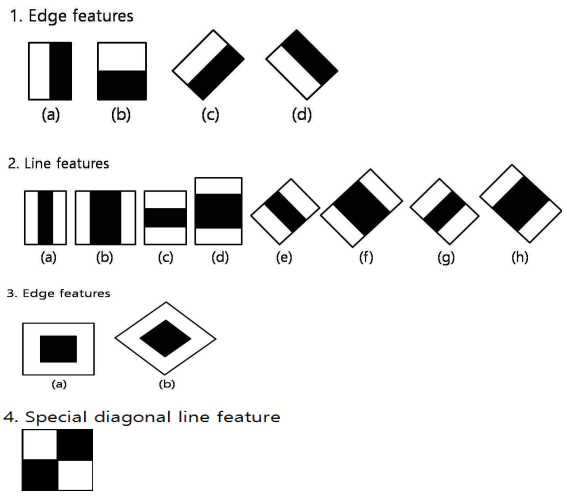

그림 1. 대표적인 Haar-like features

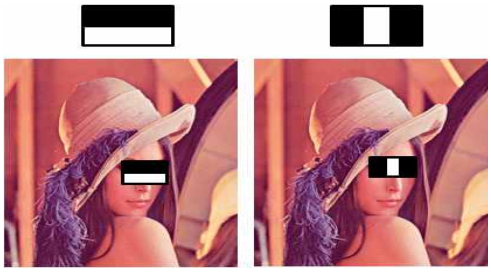

그림 2. Haar-like feature 적용형식

2.2 Integral image

직사각형 특징들인 haar-like featur들은 "Integral image"라고 불리는 적분영상에 의해 빠르게 계산 될 수 있다. 적분영상은 현재픽셀에 이전 픽셀까지의 합이 더해진 영상으로 식(1)로 계산된다.

$$I(x,y) = \sum_{x' \le x} \sum_{y' \le y} i(x',y') \quad (1)$$

$I(x,y)$는 적분영상이고 $i(x,y)$는 원본 영상이다. 예를 들어서 그림 3의 픽셀 A, B, C, D로 둘러싸인 사각형 영역 S는 적분영상을 사용하여 식 (2)와 같이 계산된다.

$$S = I(D) + I(A) - I(B) - I(C) \quad (2)$$

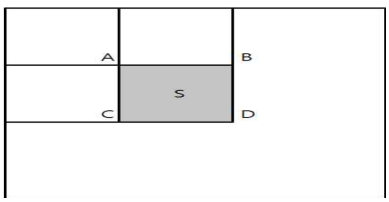

그림 3. 적분영상(Integral image)

2.3 Adaboost 학습 알고리듬

Adaboost학습알고리듬은 Yoav Freund와 Robert E. Schapire가 개발한 기계학습(machine learning)알고리듬이다. Adaboost학습 알고리듬은 약 분류기들을 선택하는 과정과 하나의 강 분류기를 만들어내는 과정으로 나눌 수 있다. 분류가 시작될 때 데이터들은 모두 일정한 초기 가중치를 갖는다. 데이터들은 여러개의 분류기들에 의해서 분류가 되는데 이때 에러율이 최소인 분류기를 약 분류기로 선택한다. 선택된 약 분류기에 의해서 분류된 데이터중 제대로 분류된 데이터에는 낮은 가중치를, 그렇지 못한 데이터에는 상대적으로 높은 가중치를 준다. 이렇게 가중치가 새롭게 업데이트된 데이터는 다음 단계의 분류기로 넘어간다. 그리고 분류기들은 높은 가중치를 갖는 데이터들을 집중적으로 분류한다. 이러한 분류과정은 여러 번 반복되는데 반복할 때마다 약 분류기가 하나씩 만들어지게 된다. 알고리듬의 마지막 단계는 앞에서 만들어진 여러 개의 약 분류기들을 선형 결합하여 하나의 강 분류기를 만들어내는 단계이다. 그림 4는 adaboost알고리듬을 정리한 것이다.

1. 학습데이터 집합은 다음과 같이 주어진다.
$(x_1,y_1),...,(x_m,y_m); x_i \in X, y_i \in -1,+1$
x_i : 이미지
$y_i = -1$: 비보행자, $y_i = 1$: 보행자
$(x,-1)$: 보행자가 없는 이미지
$(x, \quad 1)$: 보행자가 있는 이미지

2. m개의 이미지데이터를 고려할 때 초기 가중치는 다음과 같이 일정하다.
$$D_1(i) = 1/m$$

3. $t = 1,....,T$번 반복하면서 최소에러율을 가지는 T개의 약 분류기 h_t를 만든다.
$$h_t = \arg\min_{h_j \in H} \varepsilon_j = \sum_{i=1}^{m} D_t(i)[y_i \neq h_j(x_i)]$$
$if\ y_i = h_i(x_i) \Rightarrow 0,$
$y_i \neq h_i(x_i) \Rightarrow 1$ 을 반환한다.

4. 훈련과정에서 에러율 계산은 다음과 같다.
$$\varepsilon_t = \frac{\sum_{i=1}^{m} D_t(i)[y_i \neq h_i(x_i)]}{\sum_{i=1}^{m} D_t(i)}$$
if $\varepsilon_t \geq 1/2 \ \Rightarrow stop$
$$\alpha_t = \frac{1}{2} log(\frac{1-\varepsilon_t}{\varepsilon_t}) : \text{약 분류기 가중치}$$

5. 다음 약 분류기로 넘어갈 때 데이터들의 가중치는 다음과 같이 업데이트 된다.
$$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t * y_i * h_t(x_i))}{Z_t}$$

6. 최종적으로 생성되는 강 분류기는 다음과 같다.
$$H_x = sign(\sum_{t=1}^{T} \alpha_t h_t(x))$$

그림 4. Adaboost학습알고리듬

2.4 선형칼만필터(Linear Kalman Filter)

선형칼만필터는 과거의 측정데이터와 새로운 측정데이타를 사용하여 데이터에 포함된 노이즈를 제거시켜 새로운 결과를 추정 하는데 사용하는 알고리듬으로 선형적 움직임을 가지는 대상을 재귀적적용으로 동작시킨다. 즉 과거와 현재 값을 가지고 재귀적 연산을 통하여 최적값을 추적하는 것이다. 선형칼만필터는 주파수 영역이 아닌 시간영역에서 처리된다. 선형칼만필터의 알고리듬은 두 단계로 나눌 수 있다. 첫번째는 현재의 상태변수 추정치 및 공분산 값으로부터 다음 측정시간에서의 상태변수 추정치 및 공분산을 계산하는 단계이다. 두 번째 단계는 새로운 측정치와 첫 번째 단계로부터 얻어진 상태변수 추정치를 혼합하여 상태변수 추정치를 새롭게 계산하는 단계이다. 이 단계에서 칼만필터 게인 K를 최적추정 이론에 따라 계산한다. 선형칼만필터 알고리듬은 그림 5로 요약된다.

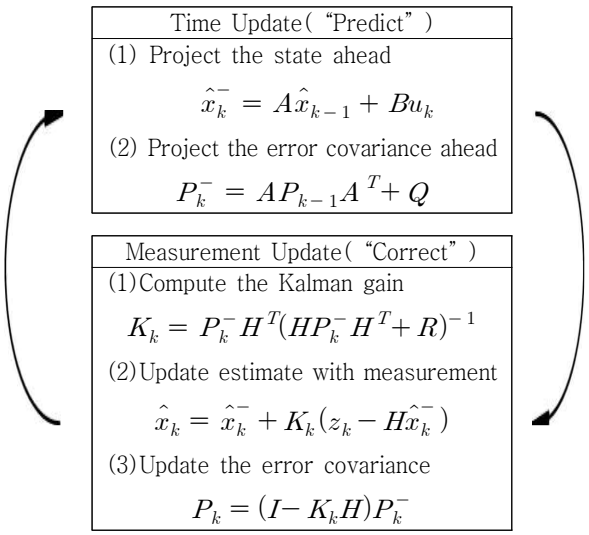

그림 5. 선형칼만필터 알고리듬

그림 5에서 Time Update단계는 이전 데이타를 근거로 예측하는 단계이다. Measurement Update단계는 새로운 측정값으로 교정하는 단계이다. $\hat{x}_{k-1}$는 이전상태변수, $\hat{x}_k$는 현재 상태변수 이다. P_{k-1}는 이전상태 추정오차공분산, P_k는 현재 상태 추정오차공분산이다. K 는 칼만게인(Kalman Gain)이고 "-"는 이전상태를 말한다. Q 는 예측 노이즈 공분산이고 R 은 측정 노이즈 공분산이다. u는 예측 오차 잡음이고 z는 측정 잡음이다. A, H, Q, R은 상수항인 행렬이다.

3 제안하는 알고리듬

기존의 보행자 검출 논문들에서는 눈에 보이는 보행자를 검출하는 것에 초점이 맞춰져있다. 하지만 보행자는 항상 눈에 보이도록 움직이는 것이 아니라 전봇대와 같은 사물에 가려질 수 있다. 따라서 본 논문에서는 Adaboost학습알고리듬을 이용하여 보행자를 분류하는 동시에 선형칼만필터를 적용하여 보행자가 특정 사물에 가려졌을 때 도 보행자를 추정하여 검출이 가능할 수 있도록 한다. 제안하는 알고리듬의 순서는 다음과 같다. 우선 보행자 검출 훈련단계에서 haar-like featur들은 식(3) 에 의해 보행자를 검출한다.

$$\sum_{withe} pixelvalue - \sum_{black} pixelvalue \geq threshold \quad (3)$$
$$\Rightarrow pedestrian$$

그리고 보행자 검출에 사용된 haar-like feature들 중 가장 보행자를 잘 찾는 haar-like feature를 약 분류기로 선택한다. 선택된 약 분류기에 의해서 가중치가 새롭게 업데이트된 데이터들을 대상으로 검출은 다시 진행되고 새로운 약 분류기가 한개 더 만들어 진다. 이 과정은 총 T번 반복되고 그로인해 만들어진 T개의 약 분류기들은 선형 결합되어 하나의 강 분류기, 즉 adaboost분류기를 생성한다. 이 강 분류기를 통해 보행자 검출이 진행될 때 선형칼만필터가 적용되어 occlusion 문제를 해결하도록 하였다.

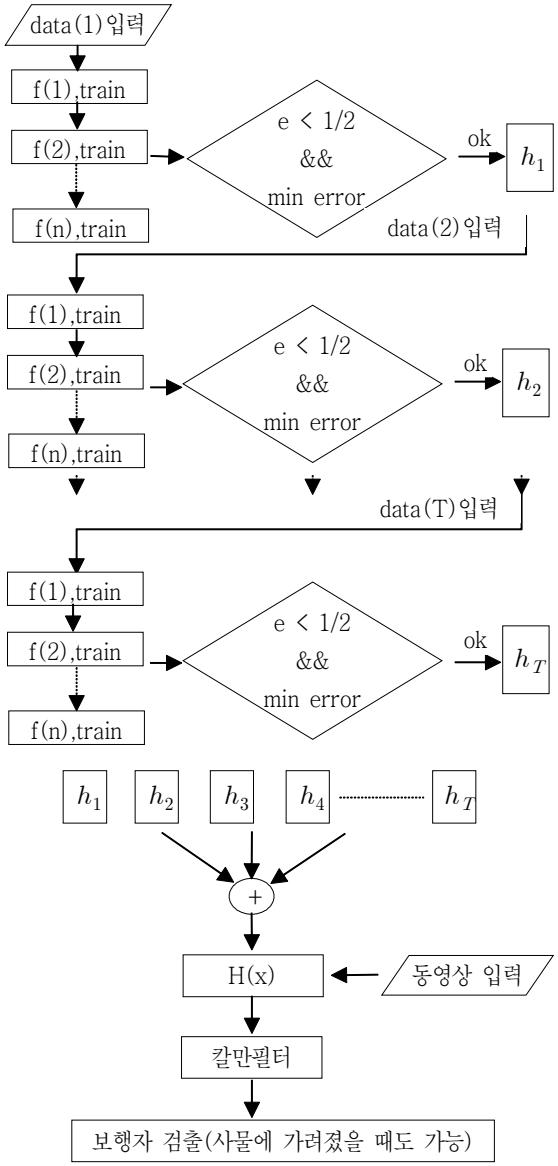

그림 6. 제안하는 알고리듬 흐름도

알고리듬의 흐름도에서 f(1)~f(n)은 haar-like feature들이고 h1~hT는 약 분류기로 선택된 haar-like feature들이다. H(x)는 약 분류기들의 선형결합으로 만들어진 최종 강 분류기이다.

4. 실험 결과

보행자 검출 훈련과 테스트를 위한 데이터베이스는 INRIA Person Databas가 제공하는 64*128사이즈의 288개의 positive sample과 1218개의 negative sample을 훈련데이터로 사용하였다.[6] positive sample은 보행자가 있는 이미지데이터이고 negative sample은 보행자가 없는 배경이미지 데이터이다. 그림 7,8은 실험에 사용된 대표적인 샘플이미지 들이다. 보행자 특징값 검출을 위해 사용된 haar-like feature종류는 그림 1에서 1a, 1b, 2a, 2b, 2c, 2d, 3a, 4에 해당하는 8개의 feature를 사용하였다.[7] 이때 특정 이미지의 보행자에 해당하는 영역을 ROI(Region of interest)를 지정해 주어 haar-like

feature들은 ROI영역을 참조하여 보행자에 해당하는 특징값을 찾아내도록 했다. 보행자 검출훈련은 총 100회를 했고 그로부터 만들어진 100개의 약 분류기로 adaboost분류기를 만들었다[7]. 이렇게 만들어진 adaboost분류기를 적용하여 보행자를 검출하도록 한 후 선형칼만필터를 적용하였다[8]. 그리고 "mitsubishi_768*576"동영상을 현실을 가정한 입력데이터로 사용했으며 adaboost분류기와 Kalman filter는 동영상이 재생되는 동안 매 프레임마다 적용되어 보행자를 찾아내도록 했다. 보행자를 검출하면 보행자에 해당하는 영역을 경계박스를 지정해 주도록 하였다.

그림 7. 실험에 사용된 positive image sample

그림 8. 실험에 사용된 negative image sample

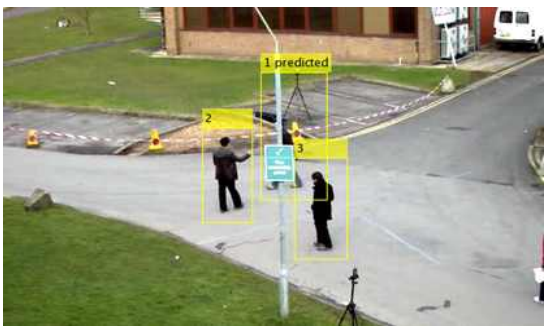

그림 9. 제안한 알고리듬에 의한 보행자 검출

FPPW(false positive per window)		True class	
		p	n
detect	Y	302(TP)	24(FP)
	N	132(FN)	-(TN)

표 1. 제안된 알고리듬의 성능측정결과

표 1은 positive image 434개와 negative image 566개로 총1000개의 이미지를 가지고 제안한 알고리듬의 성능을 측정한 결과이다. 그 결과 FPPW = 24/1000, 즉 10^{-3}에서 2.4%성능을 보여주고 있다. 이는 HOG[1]의 10^{-4}에서 3%성능에는 미치지 못한다. 하지만 보행자가 사물에 가려졌을 때, 즉 Occlusion이 발생했을 때 보행자를 검출할 수 없었던 HOG와는 달리 제안하는 알고리듬은 그림 9와 같이 보행자가 사물에 가려지는 상황에서도 보행자를 검출할 수 있는 장점을 확인할 수 있었다.

5. 결론 및 향후 연구 방향

본 논문에서는 눈에 띄는 보행자 검출에만 집중해온 기존의 방법과는 다르게 눈에띄는 보행자는 물론이고 보행자가 사물에 가려졌을 때에도 보행자를 추정하여 검출할 수 있도록 haar-like feature와 adaboost알고리듬, 선형칼만필터를 이용하여 보행자를 검출하였다. 그 결과 웬만한 보이는 보행자는 거의 검출하였고 전봇대에 가려진 보행자도 검출할 수 있었다. 하지만 최종 강 분류기인 adaboost분류기를 만들어 내기위해서는 수많은 데이터샘플들에 대한 오랜 학습과정을 필요로 했다. 또한 앉아 있는 보행자나 어린아이 같은 경우 검출률이 상당히 저하되었다. 따라서 향후 연구목표로는 데이터학습시간을 줄이는 방향과 성능을 좀 더 높이는 방향으로 연구를 진행해야 할 것이다. 그러한 방법에는 다양한 형태의 positive 샘플들을 추가하여 학습을 좀 더 철저히 진행시키는 방법도 고려할 수 있을 것이다.

감사의 글

"이 논문은 2015년도 정부(미래창조과학부)의 재원으로
한국연구재단의 지원을 받아 수행된
기초연구사업임(NRF-2015R1A2A2A01006004)"

참고문헌

[1] P. Viola and M. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, Vol. 57, pp.137-154, 2002.

[2] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection," in *Proc. of IEEE Conf. Computer Vision and Pattern Recognition*, vol.2005, pp.886-893, 2005.

[3] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," *Mathine Learning-International Workshop then Conference,* Vol. 13, pp.148-156, 1996.

[4] Dan Simon, "Optimal State Estimation : Kalman, H Infinity, and Nonlinear Approaches", pp.72-79, 2006.

[5] R. C. Gonzalez and R. E. Woods "Digital Image Processing", 3rd Edition, pp.565-579. 2011.

[6] INRIA Person dataset, http://pascal.inrialpes.fr/data/human/ (accessed May 10, 2017)

[7] GitHub. (2016. 5. 19). https://github.com/songkey/face-detection (accessed June 6, 2017)

[8] GitHub. (2017. 3. 6). https://github.com/TKJElectronics /KalmanFilter. (accessed June 6, 2017)