템플릿을 기반으로 한 보행자 교차 상황에서의 특정 보행자 검출 방법

조경민* · 차의영*

*부산대학교

Method for detecting specific pedestrian based template in pedestrian crossing

Kyeong-min Jo* · Eui-young Cha*

*Pusan National University

E-mail : jkm0723@pusan.ac.kr

요 약

본 논문에서는 보행자 검출 시, 교차 상황에서 발생하는 문제 해결을 위한 방법을 제안한다. 영상에서 특정 보행자를 검출하는 동안 다른 보행자와 교차하는 경우, 기존에 검출하던 보행자가 아닌 다른 보행자를 잘못 검출하는 문제가 발생한다. 문제 해결을 위해 제안하는 방법은 다음과 같다. 먼저, 검출할 특정 보행자를 bounding box로 선택하고 해당영역을 템플릿으로 추출한다. HOG를 이용하여 영상에서 보행자들을 검출하고, 후보영역으로 지정한다. 후보영역으로 지정된 보행자들을 앞서 템플릿으로 추출한 특정보행자와 비교하여 검출할 보행자를 최종 선택한다. 비교에는 템플릿 매칭, 히스토그램 비교와 LBP를 이용한다.

ABSTRACT

In this paper, we propose a method for detecting pedestrian, problem-solving situations that occur in a cross. When a pedestrian crossing and other, there occurs a problem of detecting the other pedestrians for detecting a specific pedestrian in the image. The proposed method for solving the problem is as follows. First, select a specific pedestrian detected by bounding box, and extracts the area as a template. Detecting a pedestrian from the image using the HOG, and designated as a candidate region. The final choice of the pedestrian detected by comparison with a candidate pedestrian with the specific pedestrian extracted for template. In comparison, using the Template matching, Histogram comparison and LBP.

키워드

보행자 검출, HOG, LBP, Template matching, Histogram comparison

Ⅰ. 서 론

객체 검출 및 추적에 관한 연구는 대부분 객체의 교차상황에서의 문제점을 가지고 있다. 객체를 검출하는 과정에서 비슷한 특징을 가진 객체와 교차가 발생하면, 기존에 검출하던 객체가 아닌 교차되는 다른 객체를 검출하게 된다.[1][2]

본 논문은 보행자 검출에서 발생하는 같은 문제를 해결하기 위한 방법을 제안한다. 보행자 검출을 위해 HOG를 이용하는데, 검출을 위한 다른 알고리즘을 이용할 경우 다양한 객체의 검출에 적용 시킬 수 있을 것이다.

II. 제안하는 알고리즘

그림 1은 제안하는 알고리즘 순서도이다. 먼저, 입력된 영상의 첫 프레임에서 검출하고자 하는 특정 보행자를 bounding box로 선택하고, 선택된 영역을 템플릿으로 추출한다. 입력된 본 영상에서는 HOG를 이용하여 영상 안의 모든 보행자들을 검출하는데, 여기서 검출되는 보행자들은 최종 검출될 보행자의 후보가 된다. HSV 변환 이미지를 비교할 때 히스토그램을 이용하는데 배경의 변화에 영향을 많이 받는다. LBP역시 마찬가지이므로 배경의 영향을 줄이기 위해, 비교에 앞서 템플릿으로 추출된 특정보행자의 영역과 보행자가 검출된 영역을 그림 2.(a)와 같이 상체와 하체 부분으로 나누고 ROI를 설정한다. 그런 다음, 후보 보행자가 오검출 되는 것을 막기 위해 템플릿 ROI와 후보 보행자 ROI를 템플릿 매칭 한다. 템플릿 매칭에 성공하면 ROI로 설정된 각각의 영역에서 HSV 변환 이미지와 LBP를 구하고 비교하여 최종 보행자를 검출한다. 그 후, 템플릿은 최종 검출된 보행자의 영역으로 업데이트 되고 다음 프레임의 후보 보행자들과 비교되어진다.

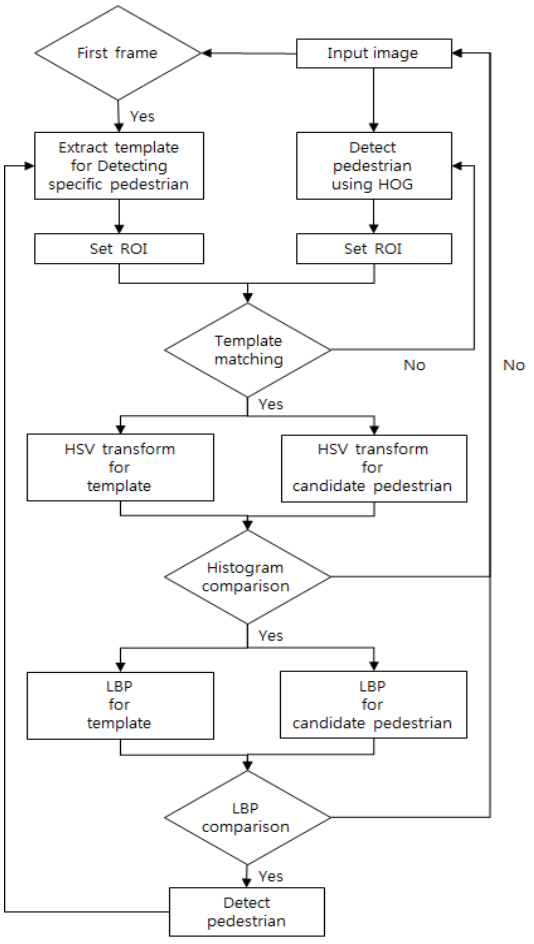

그림 1. 제안하는 알고리즘 순서도

그림 2. (a)보행자 ROI, (b)하체 ROI를 블록으로 나누어 구한 LBP

II-1 HOG

HOG[3]는 Dalal과 Trigg가 보행자 검출을 목적으로 제안한 알고리즘으로, 국소 영역의 근접 화소들 간의 밝기 변화인 그래디언트의 크기와 방향성을 구해 히스토그램으로 구성하고 이를 특징 벡터로 나타내어 형태 정보로 활용한다. 생성된 특징 벡터는 SVM을 이용하여 슬라이딩윈도우 방식으로 탐색에 활용된다. 식(1)의 과정으로 입력 영상 I(x,y)에 대한 x축과 y축의 기울기 f_x, f_y를 구하고, 이를 이용해 그래디언트의 크기M과 방향 θ을 구할 수 있다.

$$f_x = I(x+1,y) - I(x-1,y) \quad (1)$$
$$f_y = I(x,y+1) - I(x,y-1)$$

$$M(x,y) = \sqrt{f_x^2 + f_y^2}$$

$$\theta(x,y) = \tan^{-1}\frac{f_y}{f_x}$$

본 논문에서는 Open CV 라이브러리가 제공하는 HOG descriptor를 이용하여 보행자를 검출하였다.

II-2 템플릿 매칭

템플릿 매칭은 원본이미지에서 템플릿 이미지를 슬라이딩하여 가장 유사도가 높은 위치를 찾는다. 식 (2)로 구한 correlation을 이용하는데 R은 결과 이미지, T는 템플릿 이미지, I는 원본 이미지이다.

$$R(x,y) = \frac{\sum_{x',y'}(T(x',y'))\,I(x+x',y+y')}{\sqrt{\sum_{x',y'}T(x',y')^2 \sum_{x',y'}I(x+x',y+y')^2}} \quad (2)$$

본 논문에서는 ROI로 설정된 템플릿을 원본 영상에서 템플릿 매칭하여 가장 유사도가 높은 위치를 찾고, HOG로 검출된 후보 보행자의 위치와 일치하는지 확인하기 위해 사용된다.

Ⅱ-3 히스토그램 비교

히스토그램 비교는 식 (3)으로 구한 correlation distance로 유사도를 측정한다. d는 distance, H_1과 H_2는 비교에 이용하는 히스토그램, N은 히스토그램의 bin 개수이다. 본 논문에서는 템플릿과 후보 보행자의 ROI를 히스토그램 비교하는데, 비교에 앞서 HSV변환하고 hue와 saturation의 히스토그램을 비교한다. correlation distance를 threshold값으로 이용하여 템플릿과 후보보행자의 ROI가 같다고 판단한다. 실험결과 0.65에서 최적의 결과를 얻었다.

$$d(H_1,H_2)=\frac{\sum_I(H_1(I)-H_1^{'})(H_2(I)-H_2^{'})}{\sqrt{\sum_I(H_1(I)-H_1^{'})^2(H_2(I)-H_2^{'})^2}} \quad (3)$$

$$H_k^{'}=\frac{1}{N}\sum_J H_k(J)$$

Ⅱ-4 LBP

LBP[4]는 Ojala가 영상의 텍스처를 분류하기 위해 개발한 알고리즘으로, 영상의 모든 픽셀에 대해 픽셀 주변의 3x3영역에 대한 밝기 변화를 Binary값으로 코딩한다. 이웃한 픽셀이 중간픽셀보다 밝으면 1, 어두우면 0으로 코딩하고 Binary로 표현된 값은 Decimal로 나타낼 수 있다. 각 픽셀을 LBP를 이용하여 Decimal로 나타내고 그 값들로 히스토그램을 구해 텍스처 모델로 이용한다.

본 논문에서는 LBP를 템플릿으로 추출한 보행자의 영역과 후보 보행자의 영역을 비교하는 데 사용한다. 앞서 템플릿 매칭과 HSV 변환 이미지의 히스토그램 비교를 통해 비교를 하지만, 이 단계에서 오검출 되는 보행자를 다시 LBP를 통해 검증한다. 기존 LBP와 같이 각 픽셀에 대해 LBP를 구하면 프레임과 프레임사이의 변화를 잘 감지하지 못한다. 그런 이유로 그림 2.(b)와 같이 ROI 영역을 3x3의 블록으로 나누어 각 블록의 평균값을 이용하여 LBP를 구하고 비교한다. 하체 ROI에 대해서만 적용하는데 보행자가 교차하는 상황은 하체 ROI에서 먼저 변화가 감지되기 때문이다.

Ⅲ. 실험 결과 및 분석

Intel Core(TM)2 Duo E8400 3.00GHz 4GB 머신을 이용하였으며, Open CV라이브러리를 이용하여 실험하였다. 실험에 사용한 영상은 BoBoT – Bonn Benchmark on Tracking[1]의 영상을 사용하였다. 영상은 320x240 사이즈, 1017 프레임으로 구성되어 있다. 본 논문에서는 recall과 precision을 구하여 성능을 측정한다. recall과 precision은 식(4)를 이용하여 구한다.

$$Recall=\frac{tp}{tp+fn} \quad (4)$$

$$\mathrm{P}recision=\frac{tp}{tp+fp}$$

식(2)의 tp는 영상에 검출할 특정 보행자가 있을 때 제대로 검출한 경우이다. fn은 영상에 검출할 특정 보행자가 있을 때 검출하지 않는 경우이다. fp는 후보 보행자가 오검출 되는 경우이다. 특정보행자 검출을 위한 템플릿이 매번 업데이트 되는데 후보 보행자가 템플릿이 될 경우, 그 다음 프레임부터는 선택한 특정 보행자가 검출되지 않는다. 이러한 이유로, 히스토그램 비교를 할 때 fp가 검출되지 않도록 threshold값을 정한다. tn은 영상에 특정 보행자가 없을 때 검출을 하지 않은 경우이다. 본 논문에서는 그림 3.(a)와 같이 특정 보행자가 다른 보행자와 교차 되는 상황에서 반 이상이 가려져 HOG로 검출이 되지 않는 경우를 영상에 특정 보행자가 없다고 판단하였다.

표 1은 여러 threshold값에서 최적의 결과를 얻었을 때의 Confusion matrix이다. 히스토그램을 생성할 때 hue와 saturation의 bin의 개수가 각각 50, 100. 그리고 비교에 이용한 correlation distance가 0.65일 때, 그리고 LBP를 이용한 비교에서 하체 ROI의 중앙 픽셀 값이 77일 때의 결과이다.

표 1. Confusion matrix

	True	False
Positive	429	0
Negative	382	206

표 2. 표 1의 결과로 계산된 성능

Recall	0.53
Precision	1.00

표 2는 표 1의 결과로 계산된 성능인 Recall과 Precision이다. 제안하는 알고리즘이 fp의 검출이 이루어지지 않도록 하므로 Precision의 성능이 1.00로 높게 나온 반면, Recall의 경우 0.53의 성능을 나타내었다. fp의 검출을 막기 위해 검출할 특정보행자와 후보 보행자의 비교를 세 단계에 걸쳐 한 결과로 fn이 늘어났기 때문이다.

1) “Bonn Benchmark on Tracking”, http://www.iai.uni-bonn.de/~kleind/tracking/

그림 3. (a)특정 보행자가 가려져 영상에 없다고 판단한 경우, (b)교차상황 발생 시, 처리 과정

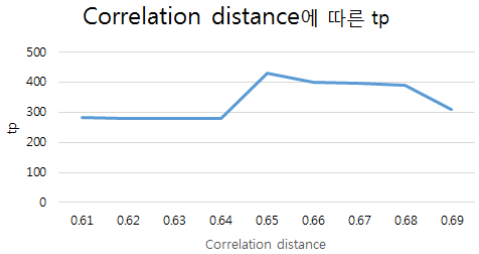

그림 4. Correlation distance에 따른 tp

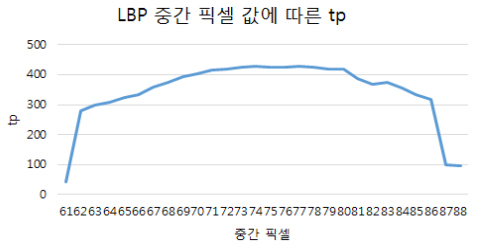

그림 5. LBP의 중간 픽셀 값에 따른 tp

그림 4는 히스토그램 비교의 threshold값으로 이용한 correlation distance에 따른 tp의 그래프이다. threshold값이 작아짐에 따라 tp가 증가하다가 0.64에서 확연히 tp가 줄어든다. fp가 측정되어 후보 보행자가 오검출되고, 업데이트된 템플릿으로 더 이상 검출이 불가능하기 때문이다.

그림 5는 ROI의 LBP를 구할 때 중간 픽셀의 값에 따른 tp의 그래프이다. 중간 픽셀과 이웃 픽셀의 값이 거의 유사할 경우, 이웃 픽셀이 중간 픽셀과 비교 하여 커지고 작아지는 것이 임의성을 띄어 LBP의 비교가 힘들다. 그러므로 중간 픽셀을 근사한 픽셀 값으로 대체하고 LBP를 구하여 최적의 결과를 얻는다. 첫 프레임에서 중간 픽셀의 값은 69이고, 그 근사 값들로 실험하여 중간 픽셀이 77일 때, 429개의 tp를 얻었다.

Ⅳ. 결 론

본 논문에서는 특정 보행자를 검출하는 동안 다른 보행자와 교차하는 경우, 기존에 검출하던 보행자를 검출하지 못하는 문제를 해결하기 위한 방법을 제안하였다. 검출할 특정 보행자를 템플릿으로 설정한 후, 입력 영상에서 HOG를 이용하여 얻은 후보 보행자들과 비교 하여 특정 보행자를 최종적으로 검출하였다. 검출된 특정 보행자는 다시 템플릿으로 설정되어 다음 frame에서 이용된다. 비교에는 템플릿 매칭, 히스토그램비교와 LBP를 이용하였다.

실험 결과, 제안하는 방법이 특정 보행자가 아닌 후보 보행자가 검출되는 fp의 검출이 이루어지지 않도록 하므로 Precision의 성능이 1.00로 높게 나온 반면, Recall의 경우 0.53의 성능을 나타내었다. fp의 검출을 막기 위해 검출할 특정보행자와 후보 보행자의 비교를 세 단계에 걸쳐 한 결과로 fn이 늘어났기 때문이다.

향후 recall의 성능을 높이기 위해 fn을 줄일 수 있는 방법에 대한 연구가 필요하다. 그림 3.(b)는 교차상황이 발생할 때, 제안하는 방법이 처리하는 과정이다. 칼만필터와 같은 예측 기법을 이용해 fn을 줄임과 동시에 다른 보행자와의 교차로 인해 영상에서 특정 보행자가 관찰되지 않는 상황에서도 위치 파악이 가능해 질 것으로 보인다. 템플릿을 비교하기에 앞서 ROI를 설정하지만 여전히 배경에 영향을 받기 때문에 배경의 영향을 없애기 위한 연구 또한 필요하다.

참고문헌

[1] Kalal, Z., Mikolajczyk, K., & Matas, J. Tracking-learning-detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 34.7. pp. 1409-1422. 2012

[2] Ben Shitrit, Horesh, et al. Multi-commodity network flow for tracking multiple people. Pattern Analysis and Machine Intelligence, IEEE Transactions on 36.8. pp. 1614-1627. 2014.

[3] Dalal, N., & Triggs, B. Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. pp. 886-893. 2005.

[4] Ojala, T., Pietikäinen, M. and Harwood, D.. A Comparative Study of Texture Measures with Classification Based on Feature Distributions. Pattern Recognition 29.1 pp. 51-59, 1996.