Moving Object Detection Using Histogram of Uniformly Oriented Gradient
Wei-Jong Yang, Yu-Siang Su, Pau-Choo Chung, Jar-Ferr Yang

Abstract—Moving object detection (MOD) is an important issue in advanced driver assistance systems (ADAS). There are two important moving objects, pedestrians and scooters in ADAS. In real-world systems, there exist two important challenges for MOD, including the computational complexity and the detection accuracy. The histogram of oriented gradient (HOG) features can easily detect the edge of object without invariance to changes in illumination and shadowing. However, to reduce the execution time for real-time systems, the image size should be down sampled which would lead the outlier influence to increase. For this reason, we propose the histogram of uniformly-oriented gradient (HUG) features to get better accurate description of the contour of human body. In the testing phase, the support vector machine (SVM) with linear kernel function is involved. Experimental results show the correctness and effectiveness of the support vector machine.

II. HOG FEATURE AND SVM CLASSIFIER

A. HOG

The HOG is a useful feature, as proposed in [7] for pedestrian detection, which essentially compute the gray gradient statistics of local image regions. The HOG based algorithms could be extended for face recognition [13], [14]. Assume the pixel located at \((x, y)\) has a gray value \(L(x, y)\), the gradients along \(x\) and \(y\) directions are respectively given as:

\[
g_x(x, y) = L(x + 1, y) - L(x - 1, y)
\]

and

\[
g_y(x, y) = L(x, y + 1) - L(x, y - 1)
\]

The gradient magnitude \(g(x, y)\), gradient direction \(\theta(x, y)\) are then respectively computed as:
\[ g(x, y) = (g_x^2(x, y) + g_y^2(x, y))^{1/2} \]  
\[ \theta(x, y) = \tan^{-1}(g_y(x, y) / g_x(x, y)) \]  

As stated in [7], the computation of gradient only refers to the neighborhood of the center pixel \( p \) in an 8×8 window. Each detection window could be divided into cells of 8×8 pixels and each group of 2×2 cells is integrated into a block in a sliding fashion. Each cell could consist of a 9-bin HOG and each block contains a concatenated vector of all its cells. Each block is thus represented by a feature vector that is normalized to an L2 unit length. Each 64×128 detection window is represented by 7×15 blocks, giving a total of 3780 features per detection window. These features are then used to train a linear SVM classifier.

**B. SVM**

The SVM [8]-[10] can achieve optimal classification of linearly separable data. For a linear SVM with the training samples, \( \{ (x_i, y_i) | 1 \leq i \leq N \} \), where \( x_i \) is the \( i \)-th instance sample, \( y_i \) is the corresponding category labels, its decision surface equation can be expressed as:

\[ \omega \cdot x + b = 0 \]  

where \( x \) is the input vector, \( \omega \) is the dynamically variable weight vector, and \( b \) is the offset. In order to find an optimal classification result, the classifier tried to find an optimal hyper plane according to (5), which can not only separate two classes correctly but also maximize the between-class distance. Accordingly, support vectors refer to the training sample points located in the classification boundaries, which are the key elements of the training sample set. Based on these theories and concepts [15]-[18], we classify the input samples by using:

\[ f(x) = \text{sgn} \left( \sum_{i=1}^{k} \alpha_i^* y_i (x_i \cdot x) + b^* \right) \]  

where \( \alpha_i^* \) is the weight coefficient corresponding to the support vector \( x_i \).

### III. THE PROPOSED SYSTEM

**A. Overview of the Proposed System**

The presented MOD system, as shown in Fig. 1, involves three major parts: 1) Selecting of regions of interest, 2) Calculating HUG, and 3) Performing the SVM for classification. In order to reduce computation time, we first scale down the resolution of the input video from 1280×720 to 320×180 and then we choose regions of interest (ROI) which are at \( x \) direction from 40 pixels to 280 pixels and at \( y \) direction from 15 pixels to 130 pixels to search the moving objects. Also, we set the red region which restrict the detection window should be within the region and the green lines which simulate the virtual traffic lanes. Then, we define the dangerous zone which is the overlap of red region and green line, as shown in Fig. 2. If the detection window within the ROI is activated, we would treat them as moving objects. The design of ROI can remove some misdetection objects in order to reduce the false alarm rate.

**B. HUG**

To compute the robust gradient magnitude and gradient direction, we modify the computations of direction-\( x \) and \( y \) gradients by adopting a one-dimensional center gradient operator weighted by \([-0.5, -0.5, 0, 0.5, 0.5]\) to achieve precise horizontal gradient and vertical gradient are defined as:

\[ \hat{g}_x(x, y) = \frac{1}{2} [L(x + 2, y) + L(x + 1, y)] - \frac{1}{2} [L(x - 2, y) + L(x - 1, y)] \]  
\[ \hat{g}_y(x, y) = \frac{1}{2} [L(x, y + 2) + L(x, y + 1)] - \frac{1}{2} [L(x, y - 2) + L(x, y - 1)] \]  

Thus, the modified gradient magnitude \( \hat{g}(x, y) \), gradient direction \( \hat{\theta}(x, y) \) are then respectively computed as:

\[ \hat{g}(x, y) = (\hat{g}_x^2(x, y) + \hat{g}_y^2(x, y))^{1/2} \]  

and

**Fig. 1 Flow diagram of the proposed MOD system**

**Fig. 2 ROI and dangerous zone**

\[
\hat{g}(x,y) = \tan^{-1}\left(\frac{\hat{g}_x(x,y)}{\hat{g}_y(x,y)}\right)
\]  

(10)

Afterwards, for each cell consisting of 8×8 pixels the gradient magnitudes are weighted according to their gradient direction and then accumulated in their bin direction, resulting in a gradient histogram. We choose a directional range from 0°~180° for HUG features of moving objects, with eight segments of 22.5° and correspondingly 8 bin directions. Then, four neighboring cells build up as a block. The blocks are scanned through all the ROI with sliding windows in overlapped fashions. Finally, to reduce the influence of illumination and background variation, normalization is essential. We use L2-norm for normalization:

\[
v = \frac{v}{\|v\|^2_2 + \epsilon^2} \rightarrow v
\]

(11)

where \(v\) stands for the characterization vector and \(\epsilon\) is a small constant.

The sizes of input sub-images are 40×80, 32×64 and 24×48 separately. To compute HUG features, we adopt 50 (5×10) cells, 32 (4×8) cells and 18 (3×6) cells, respectively. After that, four consecutive blocks construct a block. A 32-element vector is used to describe the local gradient distribution. Finally, with 36 (4×9) blocks, 21 (3×7) blocks and 10 (2×5) blocks, 1152-element vector, 672-element vector and 320-element vector can be respectively built to describe the whole gradient information of the image. Concatenating the histograms from all the sub-regions gives the final HUG feature vector, as illustrated in Fig. 3.

IV. EXPERIMENTAL RESULTS

A. Database

Our pedestrian training database contains three kinds of different scale of resolutions which are 40×80, 32×64 and 24×48. Our database contains 1044 standing pedestrian positive samples with various aspects and poses. Some of these sub-images are from the public downloadable MIT pedestrian dataset [8], while the sub-images of the scooter data set are captured by our camera systems. The negative examples are cropped from videos. For the scooter training data, we collected 235 positive samples and 136 negative samples. Some of the positive training samples are shown in Fig. 4.

![Fig. 4 Some selected positive samples from training database](image)

B. Training and Evaluation

The proposed system is developed in Visual c++ and OpenCV function library. For the hardware systems, we use the computer with Intel I-7 CPU, NVIDIA GeForce GTX 460, 8G RAM and fisheye camera.

We use the database as described in Subsection IV-B, as the training set. In our experiments, we detect the pedestrian and scooter only in ROI and evaluated the recognition rates and false alarm rates. We also compared the classification performance between HOG with SVM and HUG with SVM. For fair comparisons, their features are all extracted on blocks with same block sizes and with same cell sizes. Thus, their dimensions are 1152, 672 and 320. We tested on two different scenes as urban road and company road, as shown in Fig. 5.

The recognition rates, false alarm rates and speed are shown in Table I. We can observe that the proposed method performs better than the HOG+SVM in both of pedestrian detection (PD) and scooter detection (SD) in urban road and company road. Besides, the false alarm rate in the proposed method also achieves the best expected result. Furthermore, it can achieve real-time and can be easily implemented by the hardware system as well.

**TABLE I**

<table>
<thead>
<tr>
<th>Detected Objects</th>
<th>Methods</th>
<th>Testing video</th>
<th>Recognition rate</th>
<th>False alarm rate</th>
<th>Frame rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pedestrian</td>
<td>HOG</td>
<td>Urban</td>
<td>60.00%</td>
<td>4.35%</td>
<td>90fps</td>
</tr>
<tr>
<td></td>
<td>HUG</td>
<td>Road</td>
<td>80.00%</td>
<td>0.76%</td>
<td>90fps</td>
</tr>
<tr>
<td>Scooter</td>
<td>HOG</td>
<td>Company</td>
<td>100.00%</td>
<td>0.07%</td>
<td>90fps</td>
</tr>
<tr>
<td></td>
<td>HUG</td>
<td>Road</td>
<td>100.00%</td>
<td>0.00%</td>
<td>90fps</td>
</tr>
<tr>
<td>Pedestrian</td>
<td>HOG</td>
<td>Urban</td>
<td>80.65%</td>
<td>0.69%</td>
<td>90fps</td>
</tr>
<tr>
<td></td>
<td>HUG</td>
<td>Road</td>
<td>80.65%</td>
<td>0.54%</td>
<td>90fps</td>
</tr>
</tbody>
</table>
V. CONCLUSIONS

In this paper, a MOD system, which could perform pedestrian detection and scooter detection, is presented. We presented the new features extraction method, HUG for down scaled video to reduce the computation. The classification is based on a linear SVM classifier. In the experimental results, we compared with the original HOG with SVM classifier. The proposed system was tested in two situations including city road and urban road. The experimental results show that the proposed system with the HUG feature achieves a favorable detection rate and false alarm rate. It also showed that the system can achieve real-time detection which is able to be implemented on hardware easily.

REFERENCES


