GPU Implementation of Efficient Pedestrian Detection Based on MCMC

Jaehoon Yu*, Hiroki Sugano†, Ryusuke Miyamoto†, and Takao Onoye*
* Osaka University
† Nara Institute of Science and Technology
8916–5 Takayama-cho, Ikoma, Nara 630-0192 Japan
E-mail: yu.jaehoon@ist.osaka-u.ac.jp, onoye@ist.osaka-u.ac.jp

Abstract—Sliding window approach used in conventional pedestrian detection samples huge number of sub-windows from an input image to extract features required for classification. In this paper, we propose computationally efficient pedestrian detection based on Markov Chain Monte Carlo and GPU implementation to achieve further speed-up of the proposed method. The results of preliminary experiment using software implementation show the validity of the proposed method itself. Also, the results of GPU implementation using Tesla C1060 show that the proposed method is accelerated 25.7 to 76.1 times faster than the software implementation using Intel Core i7 CPU 975 3.33GHz.

I. INTRODUCTION

Pedestrian detection from visual images is a recent challenging problem in the field of computer vision. Many variations of pedestrians appearance, such as their clothes, poses, and illumination, make it difficult to distinguish pedestrians from other objects. For accurate pedestrian detection, selection and combination of feature descriptors and classification algorithm are important issues. Many schemes related to descriptors have been proposed, such as Haar-like method [21], local self-similarities method [15], contour-based methods [9], [16], and gradient-based methods [1], [10].

Among these works, detection algorithms based-on histograms of oriented gradients (HOG) [1] are now popularly used for pedestrian detection. Many HOG-based detection algorithms are proposed because of its robustness on deformation and illumination change. Recently Co-occurrence histograms of oriented gradients (CoHOG) [22] achieved excellent detection performance as accurate as other HOG-based and non-HOG-based schemes [11], [14], [20]. Most of conventional pedestrian detection methods combine the above feature descriptors and sliding window sampling that extracts huge number of sub-windows for generating feature descriptors in raster scan order from input image as shown in Fig. 4.

On the other hand, real-time processing of pedestrian detection is indispensable for practical applications such as driver assistance and video surveillance. In order to achieve real-time pedestrian detection without degrading detection accuracy on embedded systems a computationally efficient classification scheme using cascade classifier [4], parallel processing using multi-core processors [13], and specialized hardware engines [5], [6], [8] are proposed. However, computational costs to calculate CoHOG features and classify them are too large even for specialized hardware engines because dimension of the feature descriptor is very high (about 35,000 dimensions if the window size is 18 × 36). To solve this problem, a novel method to reduce computational costs is needed.

In this paper, to reduce the computational cost of pedestrian detection, we propose a novel pedestrian detection method using efficient sampling based on Markov Chain Monte Carlo [2], [3], [12], which generates proper samples from the given probability distribution and calculates various expectation values by using convergence of stochastic process. In the proposed scheme, Metropolis-Hastings algorithm, widely used MCMC, is used as sampling method, which is expected to control the number of sampling without degradation of detection accuracy. The target distribution and proposal distribution of Metropolis-Hastings are defined as pedestrian likelihood and random-walk, respectively. In order to examine detection performance, the proposed scheme is implemented on CPU. To reduce the processing time, the proposed method is implemented on GPU based on the profile result of implementation on CPU.

This paper is organized as follows. In Section II describes pedestrian detection based on sliding window approach. Section III details and evaluates the proposed method. Section IV shows our GPU implementation. Finally, Section V concludes this paper.

II. PEDESTRIAN DETECTION BASED ON SLIDING WINDOW APPROACH

This section introduces CoHOG feature descriptor used in this paper and shows a relation between a sampling interval and detection accuracy of pedestrian detection based on sliding window approach.

A. CoHOG-based detection

CoHOG [22] is a powerful feature descriptor, which can express complex shapes of objects by using co-occurrence of gradient orientations with various positional offsets. In
this section, CoHOG feature descriptor and its classification method using SVM is described.

CoHOG is a high-dimensional feature descriptor that uses pairs of gradient orientation. From the pairs, a histogram called co-occurrence matrix is build as shown in Fig. 1. The co-occurrence matrix \( C = (C_{i,j}) \), which represents the length of each bar in Fig. 1, is defined over an \( n \times m \) image of gradient orientations \( I \), as

\[
C_{i,j} = \sum_{p=0}^{n-1} \sum_{q=0}^{m-1} \begin{cases} 
1, & \text{if } I(p, q) = i \text{ and } I(p + x, q + y) = j, \\
0, & \text{otherwise},
\end{cases}
\]

(1)

where \( x \) and \( y \) represent an offset. An image of gradient orientations, \( I \), is generated from an original visual image by computing gradient orientations as

\[
\theta = \arctan \frac{v}{h}
\]

(2)

where \( v \) and \( h \) are vertical and horizontal gradient calculated by appropriate filters. Then \( \theta \) is divided into eight orientations, 45 degrees each, and eight labels are used for representing an orientation for each pixel. Therefore, the size of the co-occurrence matrix \( C \) becomes \( 8 \times 8 \). The offsets, \( (x, y) \), which define pairs of pixels, are shown in Fig. 2. Because of the symmetry, only half of the offsets are used within the maximum distance. The number of valid offsets is 31 including a zero offset. The co-occurrence matrices are computed for all combinations of the offsets and small regions as shown in Fig. 3. By using index \( k \) for small regions, each co-occurrence matrix \( C \) with an offset \( (x, y) \) can be expressed as a 64-dimensional vector \( f_{k,x,y} \). The final CoHOG descriptor is generated by concatenating small vectors \( f_{k,x,y} \) into a long vector \( F \).

Since the CoHOG descriptor is informative enough, accurate pedestrian detection is achieved with a simple learning and classification algorithm, such as linear SVM [7] used by Watanabe et al [22].

B. Relation between sampling interval and detection accuracy

As shown in Fig. 4, sliding window approach extracts feature vectors with a fixed sampling interval. To detect various sizes of pedestrians, it extracts feature vectors from multiple images resized from an input image. The computational cost of detection can be controlled by adjusting the sampling interval. However, if the sampling interval broaden as so to reduce computational cost, the detection accuracy is conspicuously degraded.

To examine the relation between a sampling interval and detection accuracy, we have done preliminary experiments of CoHOG-based detection at various sampling intervals using pedestrian dataset provided by INRIA [1]. Fig. 5 shows the results of them. These results show that Detection Error Trade-off (DET) curve, which represents the detection accuracy, is improved as the sampling interval gets dense, though it is usually expected that the decrease of detection miss rate and the increase of false detection rate occur simultaneously. This is because
true positives tend to appear densely near the pedestrian regions, but false positives tend to appear discretely due to the characteristic of CoHOG-based classification. Therefore, if the sampling interval is dense enough, it is expected to enhance the detection accuracy. However, in real-time applications, the dense sampling is not always good solution because it increases the number of samples and the computational complexity. To solve the above problem, in this paper, we propose a computationally efficient scheme that enables the reduction of the computational costs without degradation of detection accuracy based on Markov Chain Monte Carlo and demonstrate it by using the INIRIA dataset.

III. PEDESTRIAN DETECTION USING MCMC

Shortly, this section introduces Markov Chain Monte Carlo (MCMC), and proposes a novel sampling method for pedestrian detection using Metropolis-Hastings, which is widely used MCMC.

A. Markov Chain Monte Carlo

MCMC can generate proper samples from the given probability distribution and calculate various expectation values by using convergence of stochastic process, called Markov process. Although diverse methods exist in MCMC, such as Metropolis-Hastings algorithm [3], Gibbs sampling [2], Slice sampling [12], etc., in this paper, Metropolis-Hastings algorithm is adopted.

Metropolis-Hastings is iterative algorithm, and can draw samples from any probability distribution \( \pi(x) \), called target distribution. In each iteration, it generates candidate samples from current state by using conditional probability distribution \( q(y|x) \), called proposal distribution, and decides whether to accept each candidate by using Acceptance-Rejection method.

Metropolis-Hastings algorithm with \( n \) iterations can be represented by the following procedure.

- Set initial sample \( x^{(0)} \).
- repeat the following operations at \( t (t = 0, 1, \ldots, n) \),
  1) generate candidate \( y \) from proposal distribution \( q(y|x^{(t)}) \),
  2) and generate random number \( u \) from uniform random generator \( U(0, 1) \),
  \[
  x^{(t+1)} = \begin{cases} 
  y & \text{when } u \leq \alpha(x^{(t)}, y), \\
  x^{(t)} & \text{in other cases}
  \end{cases}
  \]
  (3)
  \[
  \alpha(x, y) = \min(1, \frac{\pi(y)q(x|y)}{\pi(x)q(y|x)}),
  \]
  (4)
  where the value of \( \alpha(x, y) \), called acceptance probability, depends only on the ratio of \( \pi(y) \) to \( \pi(x) \).

B. Pedestrian detection using Metropolis-Hastings

To apply Metropolis-Hastings algorithm for pedestrian detection problem, initial samples, target distribution, and proposal distribution must be defined. This subsection explains how to define these conditions in the proposed method.

In the proposed method, the coordinates of initial samples are given as identical sparse lattice points for same resolution images, as shown in Fig. 6.

Target distribution \( \pi(x) \) is defined by pedestrian likelihood calculated from CoHOG features and Support Vector Machine. In order to use the pedestrian likelihood as target distribution \( \pi(x) \), pedestrian likelihood must satisfy following conditions. First, pedestrian likelihood must be capable of representing the difference of pedestrian and non-pedestrian clearly. Second, pedestrian likelihood must be non-negative because proposal distribution is a probability function.
In our proposal, target distribution is defined as follows.
\[
\pi(\tilde{x}_f) = \exp(d(\tilde{x}_f)),
\]
where \(\tilde{x}_f\) represents a feature vector at coordinate \((x, y)\) and scale \(s\), and \(d\) represents the signed distance between feature vector and decision hyperplane in feature space. This likelihood distribution is the same function that is used in pedestrian tracking based on the particle filter proposed in [17], [19], which is proposed by a part of the authors.

In the proposed method, random-walk chain is applied as the proposal distribution \(q(\tilde{y}|\tilde{x})\). Random-walk chain is usually represented by the following equation.
\[
\tilde{y} = \tilde{x} + \tilde{c},
\]
where \(\tilde{c}\) is independent of \(x\), and the expectation value is 0. In this paper, normal distribution is used as the \(\tilde{c}\). Here, random-walk chain satisfies the following equation.
\[
q(\tilde{y}|\tilde{x}) = q(\tilde{x}|\tilde{y}).
\]
By substituting eq. (8), the eq. (5) is simplified as
\[
\alpha(\tilde{x}, \tilde{y}) = \min\{1, \frac{\pi(\tilde{y})}{\pi(\tilde{x})}\}.
\]
As a result, the computational cost for Metropolis-Hastings algorithm is decreased.

C. Implementation result

The detection accuracy and the computational costs are examined with pedestrian dataset provided by INRIA.

To evaluate detection accuracy, four DET curves are compared in Fig. 7, where the solid line with square points, the dotted line with square points, the solid line with X points, and the dotted line with X points are acquired from 16 pixel step sliding window, 16 times MCMC iteration, 8 pixel step sliding window, and 32 times MCMC iteration. In Fig. 7, the proposed method using 32 times MCMC iteration shows the best accuracy in these four methods.

Examples of detection results are shown in Fig. 8 to Fig. 11. Fig. 8 and Fig. 10 are the results of conventional sampling method using 8 pixel fixed sampling interval. Both of them show detection failures for some pedestrian area. However, in Fig. 9 and Fig. 11, the results of the proposed method using 32 times MCMC iteration, almost all pedestrians are detected correctly.

TABLE I shows the number of samples and Processing time of each method, which indicate computational costs. The values of TABLE I are the average result of 10 times operation using Intel Core i7 CPU 975 3.33GHz, Linux kernel 2.6.32, and 1060 x 605 images.

The experimental results show that the proposed method using 32 times MCMC iteration is more accurate than the conventional method using sliding-window with 8 pixel fixed sampling interval, even though the proposed method used only 55.57% samples and is computed in 58.17% of time compared to the conventional method.

IV. GPU IMPLEMENTATION

To achieve further speed-up with MCMC, the proposed method is implemented on GPU as parallel program. TABLE II shows the profile of the proposed scheme, acquired by Google Performance Tools.

<table>
<thead>
<tr>
<th>Function</th>
<th>Occupancy (%)</th>
<th>w/ Subfunc. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoHOGDetection:calc_hog</td>
<td>55.1</td>
<td>64.0</td>
</tr>
<tr>
<td>CoHOG_MCMC::mcmc_detect</td>
<td>29.8</td>
<td>98.3</td>
</tr>
<tr>
<td>cv:resizeGeneric</td>
<td>3.7</td>
<td>4.3</td>
</tr>
<tr>
<td>CoHOGDetection:calc_angle_bin</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>CoHOG_MCMC::next_coordinate</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

In TABLE II, 1st, 2nd, and 3rd columns show function name, processing time excluding subfunctions, and processing time including subfunctions, respectively. The mcmc_detect, the top function, occupies 98.3% of entire processing time, and calc_hog and several functions with asterisk mark are dominant processes of mcmc_detect. On the other hand, the processing time of next_coordinate that generates sample candidates based on MCMC is nearly 0.0%.

Based on the profile result, we decided to implement functions with asterisk mark and others on GPU and CPU, respectively. For the GPU implementation, CUDA library and Tesla C1060 provided by NVIDIA corp. are used.

GPU consists of multiple Streaming Multiprocessors (SMs), and by operating each SM in parallel, GPU can speed up the proposed method. Fig. 12 shows the process flow of the proposed implementation with GPU and CPU. As mentioned above, the proposed GPU implementation is divided into GPU and CPU parts. Here, the GPU part extracts CoHOG features from input images in parallel, and the CPU part
generates coordinates of initial or next samples. As shown in Fig. 12, GPU and CPU parts are executed one after the other in each iteration of MCMC. In this case, if the processing time is too long in CPU part, which does not operate in parallel, the entire detection process will be slow. However, from the profile result, the processing time of generating coordinates is negligible, so that it is expected that the CPU part hardly slow down the process speed.

Fig. 13 shows the average processing time of GPU and CPU implementations with the proposed method when the size of images is $1060 \times 605$. From the result, the GPU implementation is 25.7 to 76.1 times faster than the implementation on CPU.

Also, Fig. 14 shows the average processing time of the proposed GPU implementation and the conventional GPU implementation using sliding window [18]. Each pair of bars consists of the proposed method and the conventional method with equivalent detection accuracy. From the result, in almost all cases, the proposed GPU implementation is faster than the conventional GPU implementation except the Fixed 8 pix. vs. MCMC 64 pix., 32 iter. case. The result shows that if the number of initial samples is large enough, the
proposed implementation can fully utilize SMs in GPU, and accelerate the detection process successfully, but if the number of initial samples is small, the process operated by CPU affects the process speed, even though CoHOG extraction is still dominant.

V. CONCLUSION

This paper proposed an efficient sampling scheme for pedestrian detection based on Metropolis-Hastings algorithm, widely used MCMC, aiming to maximize the detection accuracy under the constraint of computational cost. To apply Metropolis-Hastings algorithm for pedestrian detection, it is required to define initial samples, proposal distribution, and target distribution. In this paper, the coordinates of initial samples of the proposed method are given as identical sparse lattice points for same resolution images for stable performance, and adopted random-walk chain and pedestrian likelihood as proposal and target distributions, respectively. The implementation results show that the proposed method can detect pedestrians as accurate as a conventional method, which uses fixed sampling interval, even though the proposed method used CoHOG features for rapid pedestrian detection. In Proc. of the 7th International Conference on Computer Vision Systems, pages 53–62, 2009.

REFERENCES