An OpenCL Parallelized Traffic Sign Recognition

De-kai KANG*, Xing CAI, Xu-sen GUO, Jie-xin ZHENG and Xiao-mei ZHOU
No. 135, Xingang Xi Road, Guangzhou, 510275, P.R. China

*Corresponding author

Keywords: TSDR, AdaBoost, OpenCL, ADAS.

Abstract. Traffic sign detection and recognition (TSDR) plays crucial roles in advanced driving assistant system (ADAS). In this work, we propose an OpenCL parallelized TSDR method to address the time-consuming challenge. The method employs AdaBoost algorithm for traffic sign detection and Fisherface algorithm for the traffic sign recognition. The Haar feature extraction and Adaboost algorithm are accelerated by sliding windows paralleling and stage classifier group scheduling strategies. The results of experimental work reveal that our approach offers about 12x speed-up for 1920x1080 resolution, which effectively compress the computation time.

Introduction

Traffic sign detection and recognition (TSDR) is a critical problem in Advanced Driving Assistant System (ADAS). This is a challenging task due to variable lighting, intensity inconsistencies and so on. Color and shape information are two important factors in TSDR algorithm. SVM [1], Neural network [2] and Adaboost [3,4] are common methods used by researcher.

AdaBoost algorithm [5] is one of the most employed methods for object detection. Although AdaBoost maintains a high degree of accuracy and robustness, however, it is time-consuming and can not satisfy the real-time requirement. To address this challenge, some researchers have accelerated the algorithm using OpenCL at FPGA and GPU, since these devices are more suitable for compute-intensive tasks.

Martínez-Zarzuela et al. [6] proposed an approach for AdaBoost face detection using Haar-like features on GPU and obtained overall 3.3x speed-up. Cheng et al. [7] used the techniques of scale paralleling, stage partitioning and dynamic stage scheduling on AdaBoost algorithm, solved load-unbalanced problems when realized in multicore CPU and GPU platform. Lee et al. [8] accelerated the Viola-Jones face detection algorithm and achieved 6.29x speed-up with the INHA FACE dataset.

In this paper, we proposed an OpenCL parallelized TSDR method, employing AdaBoost algorithm for traffic sign detection and Fisherface algorithm[9] for traffic sign recognition. Sliding windows parallelism and stage classifier group scheduling strategies are used to compress the execution time. The rest of the paper is organized as below: Section 2 describes the method and implementation of our method. In Section 3, Experiments and results are listed. Section 4 presents the conclusions.
Preprocessing

In the preprocessing step, we mainly enhance image details and improve the algorithm robustness. Firstly, the RGB image is transformed to HSI color space, due to the HSI is less affected from illumination changes and weather variation. Then, the intensity channel of HSI image is filtered by the homomorphic filter to enhance the image details in low light intensity. Finally, HSI image is transformed to red, blue and yellow channel images for subsequent detection step.

The preprocessing step can be high-efficiently implemented by parallel computations on GPU devices. Color space transformation can be applied to each pixel independently. Meanwhile, The DCT already has the efficiency algorithm to realize the data paralleling [10].

Traffic Sign Detection

Haar Feature and AdaBoost algorithm are employed for traffic sign candidate windows generation. Haar feature is a common feature in face recognition. In AdaBoost algorithm, every classifier contains three haar features, a certain amount of classifier constitute a stage classifier. The cascade classifier will predict the input image contains the target object if all stage classifiers output true. A scanning window is used over the input image at different locations and scales, then the windows containing the target object are processed in third step.

Haar feature extraction of the sliding windows is extremely time-consuming. A common optimization technique is integral image, which can make it easy by calculating the sum of the intensity value within a particular window using only the value of four vertexes. Haar features are computed using Eq. 1 where \( h_j(x, y) \) is the Haar-feature \( j \) computed over coordinates \( (x, y) \) on the window, relative to the position of window on the original image \( i \). The sum of the pixels over \( (m,n) \) inside every rectangle \( r \) conforming the Haar-feature, is weighted according to a \( w_{jr} \) factor. Selection of \( h_j \) and associated weighted are decided by the boosting procedure during training.

\[
h_j(x, y) = \sum_{R=1}^{\text{RectNum}} w_j R \left[ \sum_{(m,n) \in R} i(m,n) \right].
\]  

(1)

AdaBoost cascade classifier contains three parts, which are organized to tree like. Classifier are comprised a number of FeatureNum Haar feature, as stated in Eq. 2. Stage classifier contains a series of classifiers, as stated in equation Eq. 3. The cascade classifier outputs true only if all stage classifiers output true.

\[
H(x, y) = \begin{cases} 
\text{leftValue}, & \text{if } \sum_{j=1}^{j} h_i(x, y) < \text{threshold} \\
\text{rightValue}, & \text{else}
\end{cases}
\]

(2)

\[
H_{\text{stage}}(x, y) = \begin{cases} 
1, & \text{if } \sum_{k=1}^{\text{ClassifierNum}} H_k(x, y) < \text{stageThreshold} \\
0, & \text{else}
\end{cases}
\]

(3)

The RectNum is default 3 in OpenCV, \( R \) and \( w_{jr} \) denote the Haar features which are extracted in training processing, leftValue, rightValue, threshold and stageThreshold are computed during training. Different Haar features and classifier contribute with a different weight to the final decision of the stage classifier.

In order to reduce the feature extraction and classification time, sliding windows paralleling and stage classifier paralleling are employed in OpenCL paralleling accelerated algorithm.
Sliding Windows Paralleling

In Adaboost, sliding window is for detecting different size and location of targets. In CPU implementation, the cascade classifier needs to determine every sliding window sequentially. However, in GPU implementation, a batch of windows can be processed in OpenCL kernel at the same time. Fig. 2 shows the sliding windows number of different resolution. There are millions of sliding windows for larger resolution. Therefore, the sliding windows paralleling can effectively reduce the processing time.

![Figure 2. Sliding Windows Number of Different Resolution.](image)

Stage Classifier Group Scheduling

In Adaboost, a large amount of the non-target windows will be rejected at the early stage classifiers, only a few windows survived for the last stage classifiers. Fig. 3 shows the survived windows ratio at every stage classifier. It shows that the first 10 stage classifiers will reject about 95% sliding windows. This means that a large number of kernel which process the non-target window are idle, and have to wait for the kernel, which go through the most stage classifiers. Thence, a stage classifier paralleling policy is used to balance the process time of every kernel, reduce the kernel idle time.

Haar feature extraction and Adaboost cascade classifier are implemented in OpenCL kernel. In order to achieve the high data paralleling. Firstly, We compute sliding windows number in every scale in CPU. And every OpenCL kernel will generate their sliding window coordinate respectively in GPU, then determines whether the window contains a traffic sign using the cascade classifiers. Meanwhile, every kernel will classify the sliding window only using a group of stage classifier, in order to reduce the kernel idle time.

Traffic Sign Recognition

Gabor Feature based Fisherface algorithm \([11,12]\) is employed for traffic sign recognition. We first extract the sliding window Gabor feature vector using Gabor filter, then perform the principle component analysis(PCA) to reduce the feature dimension. Finally, performs the linear discrimination analysis(LDA) to extract discriminate features and match with the templates. The discriminate features of traffic signs are saved into XML file.
Gabor filter can be defined as follows:

\[
g(x, y) = \frac{1}{2\pi\delta_x\delta_y} e^{-\frac{(x^2)}{2\delta_x^2} -\frac{(y^2)}{2\delta_y^2}} e^{j2\pi f}
\]

(4)

where \(\tilde{x} = x\cos\theta + y\sin\theta\) and \(\tilde{y} = -x\sin\theta + y\cos\theta\), \(\delta_x\) and \(\delta_y\) are standard deviation of the Gaussian envelop along the x and y axes, \(\theta\) donates the orientation of the Gabor filters, \(f\) is the sinusoidal wave frequency.

Let \(I(x, y)\) be the gray scale image, the convolution of image \(I\) and a Gabor kernel \(\psi_{\mu,v}\) is define as \(O_{\mu,v}(z) = I(z) * \psi_{\mu,v}(z)\), and then we have \(\mathcal{Z}\{O_{\mu,v}(z)\} = \mathcal{Z}\{I(z)\} \cdot \mathcal{Z}\{\psi_{\mu,v}(z)\}\), and \(O_{\mu,v}(z) = \mathcal{Z}^{-1}\{\mathcal{Z}\{I(z)\} \cdot \mathcal{Z}\{\psi_{\mu,v}(z)\}\}\), \(S = \{O_{\mu,v}(z): \mu \in \{0,\ldots,7\}, \nu \in \{0,\ldots,4\}\}\) are the extracted Gabor feature. Then, PCA is employed to project the high dimensional Gabor feature into a lower dimensional space.

FLD [13] is a popular discriminant criterion that measures the between-class scatter normalized by the within-class scatter. Let \(a_1, a_2, \ldots, a_L\) and \(N_1, N_2, \ldots, N_L\) donate the classes and the number of images within each class, respectively. Let \(M_1, M_2, \ldots, M_L\) and \(M\) be the means of the classes and the grand mean. The within- and between-class scatter matrices, \(\delta_a\) and \(\delta_b\) are defined as follows:

\[
\Sigma_a = \sum_{\mu=1}^{L} P(\omega) \mathcal{Z}^{-1}\{\mu\} (y^\mu - M_j)(y^\mu - M_j)' \]

(5)

\[
\Sigma_b = \sum_{\mu=1}^{L} P(\omega) \mathcal{Z}^{-1}\{\mu\} (M_i - M)(M_i - M)' \]

(6)

where \(P(\omega)\) is a priori probability, \(\Sigma_a, \Sigma_b \in R^{m\times m}\), and \(L\) denotes the number of classes.

FLD derives a projection matrix \(\Psi\) that maximizes the ratio \(\Phi\Sigma_a\Phi' / \Phi\Sigma_b\Phi'\). This ratio is maximized when \(\Psi\) consists of the eigenvectors of the matrix \(\Sigma_a^{-1}\Sigma_b\Phi = \Phi\Delta\) where \(\Phi, \Delta \in R^{m\times m}\) are the eigenvectors and eigenvalue matrices of \(\Sigma_a^{-1}\Sigma_b\), respectively.

Due to the candidate windows size is very small, so the recognition step is executed in CPU rather than GPU.
Experiment and Result

To quantitatively measure the performance of the proposed TSDR method, several tests were done to compare the CPU and GPU implementation performance.

Dataset

Our dataset contains two parts, a part of data is from German Traffic Sign Benchmark (GTSRB) dataset [14], and the other is collected by ourselves. GTSRB consists of 43 traffic sign classes with over 50,000 annotation. we selected the part of GTSRB which is similar to Chinese traffic sign. Our dataset totally has 20 traffic signs, containing 9 red signs, 9 blue signs and 2 yellow signs. All traffic sign classes in our dataset are shown in Fig. 4.

![Traffic sign classes in our dataset.](image)

Figure 4. Traffic sign classes in our dataset.

Accuracy and Execution time

The experiments were performed in the server with Intel(R) Core(TM) i5-4900 CPU@3.3GHz and Nvidia Quadro K620. We implemented the OpenCL parallelized cascade classifier refer to the cascade classifier of OpenCV 3.2.0. Thus, the same configuration parameters were used when we compared the performance between the CPU implementation of the well-optimized OpenCV library which is widely used and considered accurate, and our OpenCL implementation. Therefore, The OpenCL parallelized TSDR method has the same recognition accuracy with the OpenCV implementation. We compute the average time after 10 executions.

Table 1 shows the execution time at five different resolutions. The result reveals that the OpenCL parallelized TSDR method achieves 12.82x speed up at 1920x1080 resolution.

![Test result of the TSDR method.](image)

Figure 5. Test result of the TSDR method.

<table>
<thead>
<tr>
<th>resolution</th>
<th>320x240</th>
<th>640x480</th>
<th>720x480</th>
<th>1280x720</th>
<th>1920x1080</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenCV frame rate [FPS]</td>
<td>21.98</td>
<td>4.12</td>
<td>3.61</td>
<td>1.18</td>
<td>0.49</td>
</tr>
<tr>
<td>Speed up</td>
<td>1.19</td>
<td>3.21</td>
<td>3.37</td>
<td>7.97</td>
<td>12.82</td>
</tr>
</tbody>
</table>
Summary
This paper proposed an OpenCL parallelized TSDR method, employing Adaboost classifier for traffic sign detection and Fisherface algorithm for recognition. By sliding windows paralleling and stage classifier group scheduling strategies, the computation time is effectively compressed. Compared to the well-optimized CPU implementation, the OpenCL parallelized method achieves 12.82x speed-up at the 1920x1080 resolution.

Acknowledgement
This work was supported by the National Key Research and Development Program of China under grant No.2017YFB1001703 and the Fundamental Research Funds for the Central Universities under grant No. 17lgjc40.

References
