# Integrative Embedded Car Detection System with DPM

Wei Zhang, Lian-fa Bai, Yi Zhang, and Jing Han(✉)

School of Electronic Engineering and Optoelectronic Technology,
Nanjing University of Science and Technology, Nanjing 210094, China
hajlyx@foxmail.com

**Abstract.** In this paper a embedded system based on CPU and FPGA integration is presented for car detection with the improved Deformable Part Model (Deformable Part Model, DPM). Original images are computed and layered into multi-resolution HOG feature pyramid on CPU, and then transmitted to FPGA for fast convolution operations, and finally return to CPU for statistical matching and display. Due to the architecture of the DPM algorithm, combined with the hardware characteristics of the embedded system, the overall algorithm frameworks are simplified and optimized. According to the mathematical derivation and statistical rules, the feature dimensions and the pyramid levels of the model descend without sacrificing the accuracy, which effectively reduce the amount of calculation and data transmission. The advantages of parallel processing and pipeline design of FPGA are made full used to achieve the acceleration of convolution computation, which significantly reduce the running time of the program. Several experiments have been done for visible images in the unmanned aerial vehicle's view and the driver assistance scene, and infrared images captured in an overlooking perspective are also tested and analyzed. The result shows that the system has good real-time and accuracy performance in different situations.

**Keywords:** DPM · Convolution acceleration · Vehicle detection
Fast feature Pyramid

## 1 Introduction

Vehicle detection is mainly used for traffic safety and UAV surveillance, which has aroused widespread concern of many scholars [1]. Most researches used various sensors to design intelligent system. Although the current GPS and radar sensors have been developed and had a good performance in obstacle avoidance, but the amount of information they provided is far less than visual sensors. With the development of the visual sensors, the cost and size of them are greatly reduced, which make them become more and more convenient in kinds of applications, such as vehicle driving assist system and unmanned airborne systems. At the same time, the chip technology has made considerable progress, which enables some complex algorithms to be implemented on embedded platform. A multi-feature fusion method was proposed to identify the UAV aerial image in [1]. Although this method had good accuracy, the real-time performance was to poor to carry on the unmanned airborne systems.

© Springer International Publishing AG 2017

For the static detection of vehicles, most of the current algorithms were based on the shape feature [2], such as HOG-LBP, Haar-like-Adaboost, DPM-SVM and so on. In order to solve the problem that the accuracy and the real-time is not strong enough in the moving vehicle detection, [3] presented a new detector based on the improved Adaboost algorithm and the inter-frame difference method. [4] combined HOG and SVM for vehicle detection in the outdoor environment, which improved the success rate of classification by adjusting the SVM parameters. Based on the characteristics of HOG, [5] proposed a two stages car detection method using DPM with composite feature sets, which can identify many kinds of vehicles in different viewing angles.

After a comprehensive comparison of the approaches proposed in the literature, aiming at the issues of poor real-time performance, narrow detection range and low accuracy in complex environment, we regard the deformable part model (DPM) as our research direction.

As for the embedded implementation of DPM algorithm, many kinds of vehicle detection system architectures exist in the literature. Most of these methods are entirely based on FPGA, which means that all the steps of the algorithm were implemented on FPGA. These approaches focus on the details of hardware design and optimization, without considering the hardware characteristics and the algorithm structure, which lead to its poor performance in flexibility and adaptability. In order to solve this problem, we decide to use CPU and FPGA embedded system platform to realize vehicle detection algorithm.

## 2 Algorithm and Implementation of Detection System

### 2.1 DPM Algorithm Overview

The Fig. 1 shows that the overall algorithm flow of DPM. DPM is a very successful target detection algorithm, which has become an important part of many classifiers, segmentation, human posture and behavior classification. DPM can be regarded as the expansion of the HOG algorithm. Firstly, the histogram of gradient direction is calculated, and then the gradient models are obtained by SVM. These trained models can be used to detect the target directly. However, the single model matching is falling far short of need of multi-view vehicle detection, so the multiple model matching is necessary. Considering the spatial correspondence among multiple models, we add the position offset between the part model and the root model as cost, that is to say, subtract the offset cost then get the composite score. It is essentially a spatial prior knowledge of the part model and the root model. In this way, we can get the final result of the target detection according to the statistic of response of each model and target.

Through analysis on the flowchart of DPM algorithm, we notice that the convolution response between each model and image in which a large mount of computation should be carried out is basically a simple iterative calculation. On the contrary, the extraction of image features and the calculation of feature pyramid need little computing resources, which are the most complex parts of algorithm. According to the characteristics of structure of the DPM, we decided to design an embedded platform composed of CPU and FPGA to implement the vehicle detection in different scenarios.

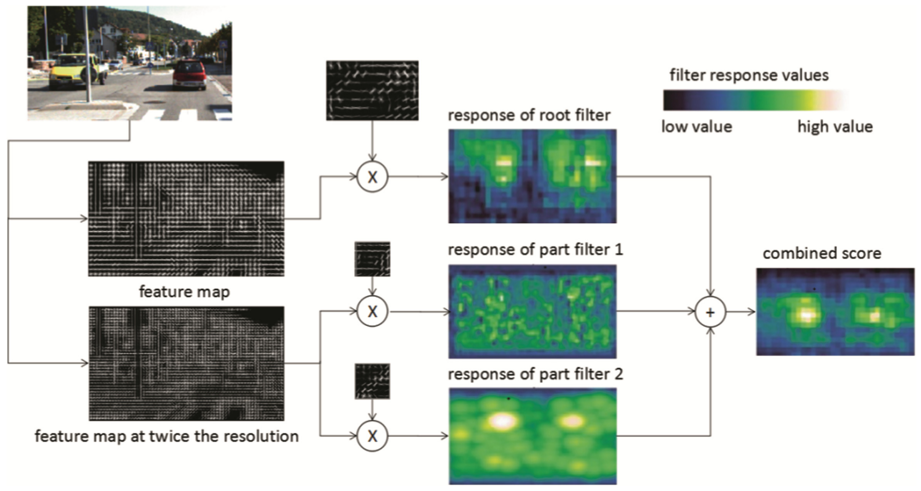


**Fig. 1.** The overall algorithm flow of DPM

## 2.2 System Architecture

In order to implement the DPM algorithm in Advanced Driver Assistant Systems (ADAS) and unmanned aerial system, the consumption and volume of the hardware platform must meet the requirements. Low-power FPGA chip has strong parallel processing ability, which is suitable for the calculation of convolution. It is possible to improve the uptime of the whole algorithm by utilizing pipeline architecture of FPGA. Tegra X1 combines the NVIDIA Maxwell GPU architecture with an ARM CPU cluster to deliver the performance and power efficiency required by computer graphics and artificial intelligence. Designed for power and space constrained applications, Tegra X1 has many kinds of peripherals and interfaces. So we built a embedded image processing system based on NVIDIA Tegra X1 and XILINX Spatan6 XC6SLX100T.

The overall overview of the whole system is shown in Fig. 2. Tegra X1 is responsible for acquiring the video stream through the USB port, image preprocessing and scaling, generating feature pyramid of image, reducing feature dimensions and pyramid levels. Feature images selected transmit to FPGA convolution via ethernet for calculation, and then the images sorted return to Tegra X1 for statistical matching and stream out the results to the HDMI. Due to the complexity of DPM algorithm and the whole system requirements, we need to consider some detail designs in specific implementation of the system. The difficulties in the realization of the system and the specific solutions are described below (Fig. 3).

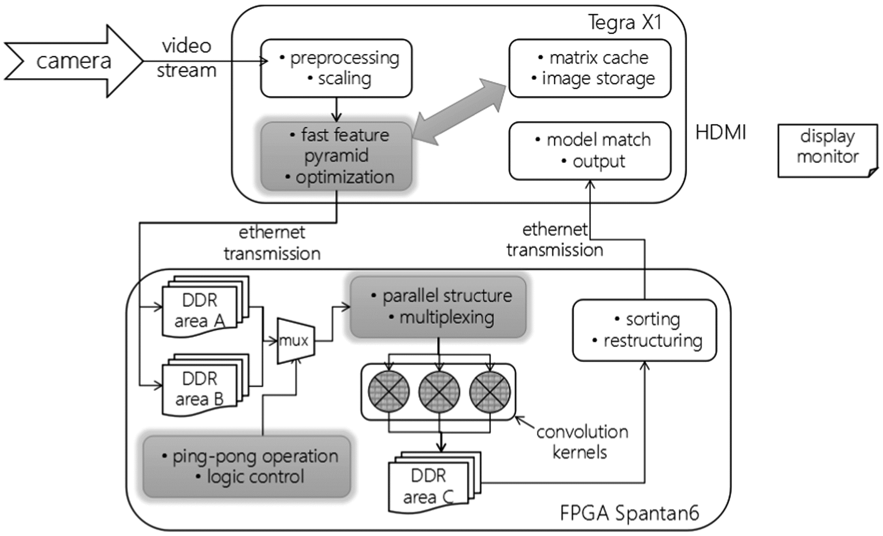

camera
video stream
Tegra X1
• preprocessing • scaling
• fast feature pyramid • optimization
• matrix cache • image storage
• model match • output
HDMI
display monitor
ethernet transmission
ethernet transmission
DDR area A
DDR area B
mux
• parallel structure • multiplexing
• sorting • restructuring
convolution kernels
• ping-pong operation • logic control
DDR area C
FPGA Spantan6

**Fig. 2.** The whole architecture of image processing system

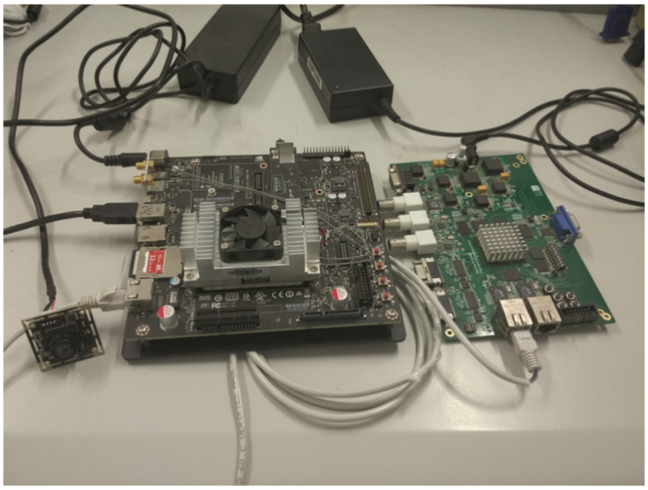


**Fig. 3.** The detection system hardware

## 2.3 Feature Pyramid and Optimization

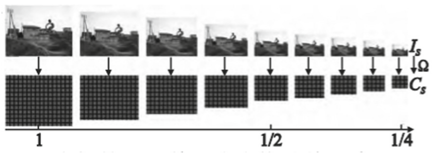
After the original construction of feature pyramid, the different scale images acquired exceed dozens of layers. If these layers are directly calculated by the HOG feature and carried out the subsequent convolution operations, the amount of computation required will be too large, resulting in severe time consuming in the whole system, which means that real-time will be seriously affected. In order to deal with this problem, we should reduce the levels of pyramid as far as possible without affecting detection accuracy. [8] presented a method based on statistical characteristics of fast feature construction of pyramid. The key sight of it was that one may compute finely sampled feature pyramids at a fraction of the cost, without sacrificing performance: for a broad family of features they found that features computed at octave-spaced scale intervals were sufficient to approximate features on a finely-sampled pyramid. This method greatly reduced the levels of pyramid which need to be calculated entirely. The core idea can be described by the following formula:

$$C_s = \Omega\left(I_s\right) \tag{1}$$

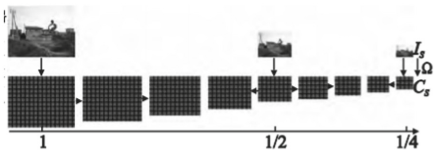
As is said above, $I$ represents the original image and $C$ represents the feature image. Let $I_s$ denote $I$ captured at scale $s$ and $R(C, s)$ denote $I$ resampled by $s$. As long as $C = \Omega(I)$ is computed, we can predict the image $C_s = \Omega\left(I_s\right)$ at a new scale $s$ using only $C$. Instead of computing $C_s = \Omega(R(C, s))$, we propose a new approximation below:

$$C_s \approx R(C, s)s^{-\lambda_\Omega} \tag{2}$$

Figure 4 shows a visual demonstration of Eq. (2). After the original image is scaled to 1/2 resolution, we calculated the HOG features of these two images. By interpolation and scaling these two feature maps we get the other layers of pyramid.



(a) The original construction of pyramid



(b) The proposed construction of pyramid

**Fig. 4.** The construction of fast feature pyramid

## 2.4 Convolution of Feature Matrix

Convolution calculation is implemented on the FPGA. The main method to implement FPGA design efficiently focus on the buffer scheme and the improvement of the convolution kernel module. Following is the specific implementation process:

**Convolution Accelerator.** According to the algorithm needs, we design two sizes of convolution kernel (6 * 6 and 15 * 5), the number of 6 * 6 kernel is 16 and the other is 2. The parallel computation of convolution is realized by shift registers multiplexing and module cascade. The design of the convolution architecture is shown in Fig. 5. The whole convolution accelerator works as follows: buffer the incoming pixel sequence in shift registers, start calculating convolution when first pixel arrive in the first kernel. The outputs of convolution kernel enter the next shift register and control logic module simultaneously. On the one hand the pixel sequence continues to be transmitted to shift register and calculated in convolution kernel, on the other hand it is also transmitted to control logic module for sorting and restructuring. The circulation has been in progress until the first pixel complete calculation in last convolution kernel, at the same time the pixel sequence of the second row and the third row are also done computation in corresponding kernel. These results enter control logic module, sorted by specific rules and output to next module.

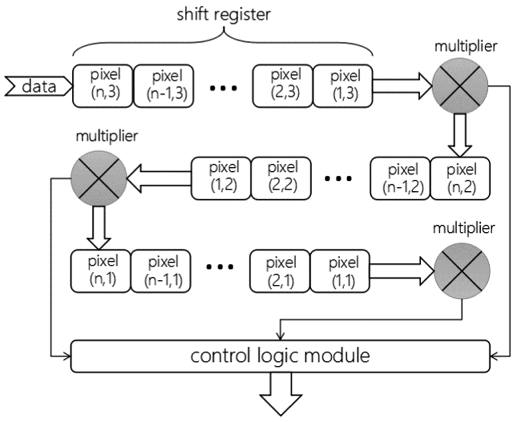
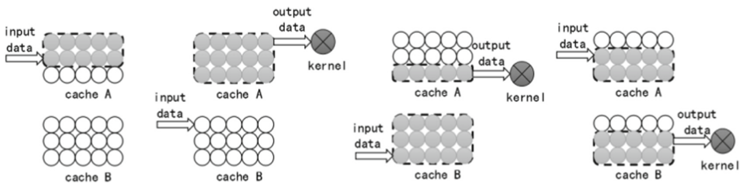


**Fig. 5.** The framework of convolution accelerator

**Buffer Scheme.** FIFO and DDR3 are used to cache the data in order to ensure timing requirement. The main function of FIFO is to maintain the continuity and integrity of data when the data is transmitted. DDR3 is mainly for the realization of the ping-pong operation shown in Fig. 6, with the aim of hiding the data transfer time to the computation time. Figure 6(a) denotes the initial state. The incoming data is stored in A area of DDR3 until this area is full, and then the address control module sends a signal that changes the storage path to the B area. Meanwhile, read the data stored in A area to convolution kernel for computation, this is working state 1. Once B area is full, address control module holds incoming data waiting for the convolution kernel to completed calculation, at this time the system is in working state 2. When convolution kernel finishes all calculations, address control module switches read address and write address, this is working



(a) initial state (b) working state 1 (c) working state 2 (d) working state 3

**Fig. 6.** The ping-pong operation

state 3. And then cycle three working states above. It appears the data is continuously carrying on transmission and processing from the data input and output.

**Logical Architecture and Data Reuse.** Prior to discussion of data reuse, it is necessary to consider the computation to communication ratio that features the DRAM traffic needed by a kernel in a specific system implementation, used to describe the computation operations per memory access. The data reuse optimization can reduce the total number of memory access, improving computation to communication ratio. Under the premise of limited bandwidth and limited on-chip resources, we design corresponding data multiplexing architectures for different sizes of convolution kernels respectively. In order to save the shift register resources, we designed a shift register multiplexing module, effectively reducing the number of shift register.

### 2.5 Communication

The communication between FPGA and Tegra X1 is based on ethernet. Due to the large amount of data to be transmitted and the high stability requirements, there will be serious packet loss and poor data transmission stability if we simply use the UDP/IP protocol to transmit. Although TCP/IP protocol can solve the problem of packet loss, the data transmission speed can not reach the extent of the algorithm needs. After a careful study of our algorithm, we write a protocol that preserves both the stability of TCP and the speed of UDP. We carried out the pseudo TCP coding of the transmitted data over the UDP protocol, which made a speed-stability trade off.

## 3 Experiments and Evaluation

In order to evaluate the adaptability and accuracy of the proposed image processing system in various situations, we have done a lot of experiments and tests. We test UAV aerial images in different heights, aiming to assess the effectiveness of the system for the detection of targets at different scales. For the driver's view, we use the KITTI data sets for test, which basically meet the requirements of the auxiliary driving scene. We also test our system with infrared images.

### 3.1 Assessment Architecture

Precision and recall are two quality indicators used to evaluate the performance of vehicle detection, defined respectively as:

$$precision = \frac{true\ positives}{true\ positives + false\ positives} \tag{3}$$

$$recall = \frac{true\ positives}{true\ positives + false\ negatives} \tag{4}$$

Precision indicates the percentage of detected cars out of the total number of detected objects, which can measure the accuracy of the algorithm. Recall represents the ability of the algorithm to detect all the targets in the image. Therefore, these two indicators constitute a complete architecture.

## 3.2 Results

A total of the 408 frames UAV aerial images with a resolution of 640 * 480 were collected from in 60 m and 100 m height. We selected different sizes of positive and negative samples from these images, training in the semi-supervised way with Matlab. Testing the trained models many times, we set the parameter of NARC (number of aspect ratio cluster to use) to 2, so that the trained models can achieve an optimal balance between speed and accuracy.

Table 1 shows a comparison of the proposed system with other systems. The vehicle detection system we presented is better than [6, 7] in precision and recall index, we can find from Fig. 7 that the background is very complex and numerous cars are shaded. In such a complicated environment our system still have good precision and recall, reflecting the advantage of our method strongly. [9] is slightly better than our system, the following discussion may be the main reason:

- The algorithm proposed by [9] was implemented on a Intel i5 processor at 3.4 GHz while ours was carried on embedded system. There is a marked difference in the aspect of hardware.
- The images tested in [9] have few vehicles with very clear background, on the contrary, images detected in this paper have a number of vehicles and some of them were partial occluded, which increase the difficulty of detection to a certain extent.

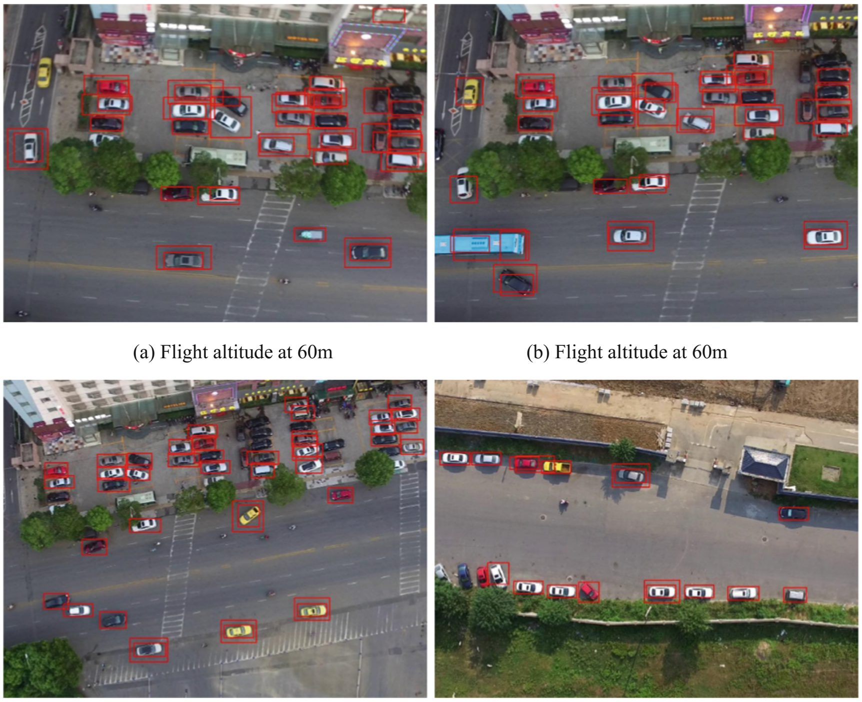
**Table 1.** Comparison of UAV images test

| | Proposed | [6] | [7] | [9] |
|---|---|---|---|---|
| Precision | 84.3% | 75.0% | 83.7% | 91.3% |
| Recall | 81.7% | 65.0% | 51.3% | 90.0% |

As for the problem of insufficient hardware, we can make full use of the Tegra X1 GPU to accelerate the construction of pyramid, improving the performance of detection. For the vehicle detection in complex environment, an effective solution is to strengthen the depth of the training sets, the specific approach is to increase the parameter of NARC or increase the number of training samples.

The test images in KITTI have a total of 379 frames with a resolution of 1392 * 512. We use Matlab to intercepted a large number of positive and negative samples for training. The parameter of NARC is set to 4.

The overall difficulty of detection in driver's view is higher than that of the UAV aerial view. Due to the various sizes of the vehicle targets in this scenario and the differences among vehicles will be more obvious, so we need more samples to be trained, which greatly increase the complexity of the training sets. Furthermore, environmental


(a) Flight altitude at 60m (b) Flight altitude at 60m

(c) Flight altitude at 100m (d) Flight altitude at 100m

**Fig. 7.** Results of unmanned aerial vehicle test

conditions such as Light difference, shadows of obstructions and perplexing objects also decrease the accuracy of detection.

Figure 8(b) shows the Precision-Recall curve obtained after a large number of experimental tests. As can be seen from the graph, the system proposed in this paper is slightly

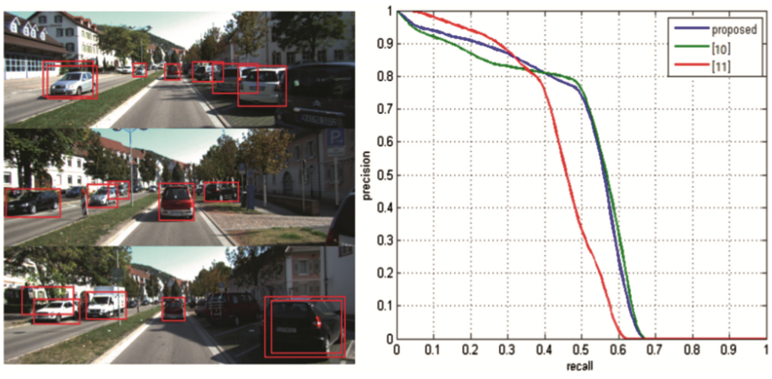


**Fig. 8.** Results of KITTI test

higher than that of [10] in precision when recall is between 0–0.4, and it is almost the same as the [10] while the recall is above 0.4. In view of the overall situation, the performance of our system is slightly better than [10]. Precision of proposed system is slightly inferior to [11] when recall is low but it is significantly higher than that of [11] at higher recall rate. On the whole the system we presented has strong adaptability in a variety of situations.

In order to evaluate the performance of detection with infrared images, we retrain our models to fit the infrared objection. As the infrared images are monochrome images with a limited gray level, the information extracted from them is much less than visible images, leading to a slight decrease in accuracy. As we can see from Fig. 9 that our algorithm still has a good recognition in this situation.

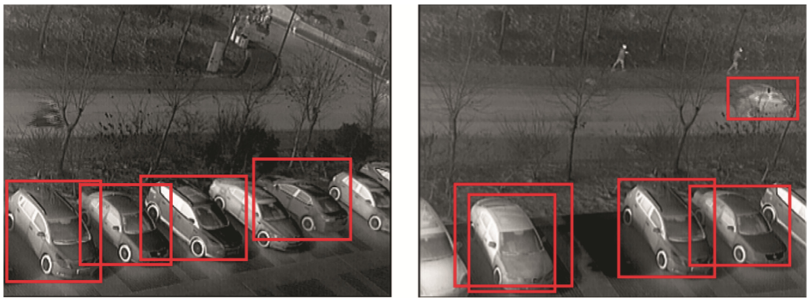


**Fig. 9.** Detection with infrared images

The convolution acceleration module of the system is implemented on FPGA with ISE Design Suite 14.7, and the layout and routing are provided by the ISE toolkit. ISE will also generate resource occupancy report, as shown in Table 2. As can be seen that the use of FPGA hardware resources for the design of the convolution accelerator is quite adequate.

**Table 2.** FPGA resource occupancy report

| XC6SLX100T | Used | Available | Percentage |
|---|---|---|---|
| Flip-flops | 86262 | 126576 | 68.2% |
| LUTs | 57352 | 63288 | 90.6% |
| Block RAM | 148 | 268 | 55.2% |
| DSP slices | 164 | 180 | 91.1% |

In order to evaluate the overall performance of the system, we have done extensive test in different hardware systems, choosing some important indicators to assess. With the bandwidth of 20 MByte/s, according to the total clock cycle of FPGA and the total time spent on Tegra X1 for program we can calculate a theoretical processing time. As shown in Table 3, compared to the other two kinds of hardware our system has a significant advantage in power consumption, and the loss of frame rate and resolution is acceptable, the overall performance is good enough to meet the needs of most application scenarios.

**Table 3.** Comparison of performance and consumption

| Hardware/Performance | Processing time (ms) | Resolution (pixels) |
|---|---|---|
| Intel i5 2.4 GHz with original DPM | 650 | 640 * 480 |
| Intel i5 2.4 GHz with optimized DPM | 185 | 640 * 480 |
| Proposed (UAV) | 65 | 640 * 480 |
| Proposed (KITTI) | 110 | 1392 * 512 |

## 4 Conclusion

In this article we discuss the various methods of vehicle detection, describing the advantages of DPM algorithm in specific applications. With detailed analysis on algorithm we decompose it into several modules to be implemented in the embedded platform. In this paper, a vehicle detection system composed of Tegra X1 and FPGA Spatan 6 is presented, which makes full use of the powerful floating-point computing ability of CPU and the fast parallel processing ability of FPGA. Compared to the traditional CPU and GPU platform, the embedded system proposed has low power consumption, good performance of detection and fabulous mobility and adaptability that other system can not achieve. For the further optimization and promotion scheme of the system, a method that worth trying is adding GPU accelerator to the whole calculation of the system to further improve the calculation speed and detection performance.

**Acknowledgements.** This work was supported by the Natural Science Foundations of China (61231014, 61373061 and 61501235) and Fundamental Research Funds for the Central Universities (30915011335).

## References

1. Wang, J.R.: Research on Multi-feature Fusion of Aerial Photography Image Recognition taken by UAV. Chengdu University of Technology, Chengdu (2015)
2. Yang, X.F., Yang, Y.: A method of efficient vehicle detection based on HOG-LBP. Comput. Eng. **40**(9), 210–214 (2014)
3. Liu, Y., Wang, H.H., Xiang, Y.L., Lu, P.L.: An approach of real-time vehicle detection based on improved adaboost algorithm and frame differencing rule. J. Huazhong Univ. Sci. Technol. **41**(S1), 379–382 (2013)
4. Guzmán, S., Gómez, A., Diez, G., Fernández, D.S.: Car detection methodology in outdoor environment based on histogram of oriented gradient (HOG) and support vector machine (SVM). In: Networked and Electronic Media (LACNEM), Medellin, Colombia, pp. 1–4. IET (2015)
5. Xu, H., Huang, Q., Jay Kuo, C.C.: Car detection using deformable part models with composite features. In: Image Processing (ICIP), Phoenix, AZ, USA, pp. 3812–3816. IEEE (2016)

6. Rosenbaum, D., Charmette, B., Kurz, F., Suri, S., Thomas, U., Reinartz, P.: Automatic traffic monitoring from an airborne wide angle camera system. ISPRS Arch. **37**(B3b), 557–562 (2008)
7. Maria, G., Baccaglini, E., Brevi, D., Gavelli, M., Scopigno, R.: A drone-based image processing system for car detection in a smart transport infrastructure. In: Electrotechnical Conference (MELECON), Lemesos, Cyprus, pp. 1–5. IEEE (2016)
8. Dollár, P., Appel, R., Belongie, S., Perona, P.: Fast feature Pyramids for object detection. IEEE Trans. Pattern Anal. Mach. Intell. **36**(8), 1532–1545 (2014)
9. Qu, Y.Y., Jiang, L., Guo, X.P.: Moving vehicle detection with convolutional networks in UAV videos. In: Control, Automation and Robotics (ICCAR), Hong Kong, China, pp. 225–229. IEEE (2016)
10. Yebes, J., Bergasa, L., Arroyo, R., Lázaro, A.: Supervised learning and evaluation of KITTI's cars detector with DPM. In: Intelligent Vehicles Symposium Proceedings, Dearborn, MI, USA, pp. 768–773. IEEE (2014)
11. Felzenszwalb, P., Girshick, R., McAllester, D.: Cascade object detection with deformable part models. In: Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, pp. 2241–2248. IEEE (2010)
12. Kadota, R., Sugano, H., Hiromoto, M., Ochi, H., Miyamoto, R., Nakamura, Y.: Hardware architecture for hog feature extraction. In: Intelligent Information Hiding and Multimedia Signal Processing, Kyoto, Japan, pp. 1330–1333. IEEE (2009)
13. Ma, X., Najjar, W., Chowdhury, A.R.: High-throughput fixed-point object detection on FPGAs. In: Field Programmable Custom Computing Machines, Boston, MA, USA, p. 107. IEEE (2014)
14. Negi, K., Dohi, K., Shibata, Y., Oguri, K.: Deep pipelined one-chip FPGA implementation of a real-time image-based human detection algorithm. In: Field Programmable Technology (FPT), New Delhi, India, pp. 1–8. IEEE (2011)