Proceedings of 2015 IEEE
International Conference on Mechatronics and Automation
August 2 - 5, Beijing, China

# Real-time Shape and Pedestrian Detection with FPGA

Han Xiao, Haitao Song, Wenhao He and Kui Yuan
*Institute of Automation, Chinese Academy of Sciences*
*Beijing, China*
{han.xiao, haitao.song, wenhao.he, kui.yuan}@ia.ac.cn

***Abstract* – Detecting objects according to their shapes from infrared images is needed in many applications. In this paper, two methods of shape detection are designed and implemented on an FPGA (Field Programmable Gate Array) to achieve real-time performance with embedded system. The first method is for rigid object detection and is named Shape Constraint (SC), which uses 56 binary templates at each scale to represent an object with different viewpoints and rotation angles. The templates are arranged in a tree structure with common regions extracted as nodes on different levels. The computation is completely pipelined in the FPGA. Three toys are used for experiment and can be detected simultaneously from cluttered scenes, thus demonstrating the effectiveness of this method. The second method is for nonrigid objects and is based on Naïve Bayes. Implemented in a pipeline, it enables the FPGA to perform pedestrian detection in real time from infrared images. In contrast to conventional methods in which a huge number of negative samples are collected for training, an even distribution is assumed in our method and no negative samples are needed, thus greatly shortening the training time. The hardware computation architecture of these two methods can be easily applied to video frames of arbitrary resolution, generating detection results for each frame at the same rate as image capturing.**

***Index Terms - FPGA; Shape Matching; Pedestrian Detection; Naïve Bayes; Infrared Image***

## I. INTRODUCTION

Infrared cameras are widely used in many areas. Since objects like pedestrians and vehicles have higher temperatures than the background, they are brighter in infrared images. Therefore, if thresholded properly, infrared images can be turned into binary images in which objects with higher temperatures are distinct and presented as certain shapes. On this basis, it is possible to detect objects from infrared images according to their shapes, which is a desirable functionality in applications like intelligent visual surveillance and mobile robot navigation.

If implemented on a CPU (Central Processing Unit) with pure software, shape detection requires a huge amount of computation, impeding its real-time performance. Fortunately, as each pixel in a binary image is represented with only 1 bit in digital logic, it is hardware efficient to process binary images with FPGAs (Field Programming Gate Array). Therefore, it is very promising to realize shape detection with an FPGA, as it not only accelerates the computation, resulting in real-time performance, but also requires very low power, enabling embedded applications.

A number of researchers have worked with FPGA on rigid object detection. In [1], shapes can be detected from an 8×8 binary matrix with FPGA by comparing the pixels, but no rotation is allowed. In [2], morphological operations are used to detect certain shapes with FPGA and again, rotation invariance is lacking. [3] focuses on the detection of the vertices of polygonal shapes, while [4] is a more general method which can detect arbitrary shapes with FPGA. However, too many software based ideas and algorithms are inherited in [4], which limits the performance. For example, connected component analysis is performed in the first stage of processing in [4] and therefore only isolated objects can be detected. In other words, if multiple adjacent objects are connected with one another in the binary image, none of them can be detected by the method in [4].

In order to achieve robustness in rigid object detection against viewpoint change, rotation as well as adjacency, a method named Shape Constraint (SC) is proposed in this paper and implemented on a low cost FPGA. Section II gives a detailed description of this method and displays the experimental results.

As to nonrigid objects, shape detection methods with more flexibility should be considered. The most common nonrigid objects in infrared images must be pedestrians which are upright most of the time, requiring less rotation invariance in the detector. [5] focuses on pedestrian detection with FPGA and HOG (Histogram of Oriented Gradients) features. Although the method of [5] is based on visible images, it is possible to migrate it to infrared images. [6] detects pedestrians with FPGA using an adaptive shape model which is operated on contour edges. The method of [6] needs floating point computations and therefore requires expensive high-end FPGAs, and still the acceleration is very limited—only 4 to 15 times compared with a software implementation.

Naïve Bayes is a good method for nonrigid object detection. In [7], Naïve Bayes is used to classify a binary feature vector extracted from the silhouette of a moving object detected by applying background differencing. Obviously, if the background is changing all the time, such as in mobile robot applications, the method of [7] will not work. Nonetheless, Naïve Bayes indeed can be used to detect objects directly from a changing background based on the appearances of the objects. [8] is a good example in which Naïve Bayes is implemented on an FPGA to detect faces from edge images. The problem with [8] is that a huge number of negative samples are employed for training so as to let the

Work supported by National Natural Science Foundation of China (No. 61273360 & No. 61203328) and the National High Technology Research and Development Program of China (No. 2013AA040903).

detector know what is not a face. This could lead to a very long training time, yet still cannnot avoid false positives because the entire image space in [8] consists of $256^{(20\times20)}$ elements, which is too huge a space to sample.

In this paper, we propose a Naïve Bayes based method for the detection of pedestrians from infrared images with FPGA. This method is conceptually similar to the face detection method of [8], but is fundamentally different in that no negative samples are needed for training. Instead, an even distribution is assumed for negative samples, because we have no reason to prefer any pixel in the detection window when it does not contain a pedestrian. In this way, the training time is significantly shortened. Experiments have demonstrated the effectiveness of this method. Details are given in Section III.

## II. Rigid Object Detection

A method named Shape Constraint (SC) is proposed here for FPGA based rigid object detection. Three toys are used for experiment, as shown in Fig. 1. In order to fully demonstrate the power of this method, the experiments are conducted on visible images which are generally more cluttered than infrared images. The operation details of SC are described as follows.

Firstly, as the gray pixel stream is input into the FPGA, the original gray image is turned into a binary image by local thresholds which are obtained by averaging the gray values of the pixels in a 32×32 sliding window. Such local thresholds are more adaptive than a global one, and can give a more stable binary image under illumination changes. When implemented on the FPGA, the 2D averaging operation is separated into two 1D averaging operations along the columns and rows respectively.

Secondly, as the binary pixel stream is generated by the aforementioned operation, a sliding window for shape detection is opened on the binary image. All the objects are detected from within this window, and therefore it is called "the top window". The size of the top window is customized according to the largest scale on which the objects are to be detected. In our experiment on the 3 objects (Fig. 1), the size of the top window is 21×16.

Finally, specific detectors for each object in each posture at each scale run in parallel in the top window and generate the detection results for each pixel position as the window scans the whole image. These detectors are obtained during offline training in which the template images of the 3 objects in 56 postures at 2 scales are horizontally flipped to generate their mirror images. As a result, there are 3×56×2×2=672 detectors in all.

However, the 672 detectors are not implemented independently. In fact, they are divided into 12 groups with 56 detectors in each group arranged in a tree structure and implemented together in the pipeline. Take one of the toys for example. First, pictures are taken from 8 viewpoints to allow for out-of-plane rotation (Fig. 2). Then the 8 pictures are turned into binary images and rotated by 7 different angles (in-plane rotation) resulting in 56 binary template images which are called "postures" (Fig. 3). Notation for the 56 postures is given in Fig. 4. The next step is to combine the 56 postures into a tree structure. As shown in the left part of Fig. 5, the first row of the postures (a5~h5) are divided into 4 adjacent pairs. Initially, the first pair a5 and b5 represent the first two template images in Fig. 3. Then after combination, a new template image ab5 which represent the common part of a5 and b5 is created, and the meanings of a5 and b5 are updated to only represent the remaining parts. In this way, a hierarchical structure is built, in which ah5 represents the common part of all the 8 template images in the first row of Fig. 3. After horizontal operations have been completed on all the rows, the building process continues in the vertical direction, as shown in the right part of Fig. 5.

Such a tree structure is named a "Common Tree". Obviously, nodes on the higher levels are shared among more detectors than nodes on the lower levels of the tree. Consequently, we have found from experiments that it is not necessary to fully implement the Common Tree in the FPGA. Instead, only the root node and a few higher levels are enough.

Fig. 1. Three toys for the experiment of rigid object detection

Fig. 2. One of the toys from 8 viewpoints

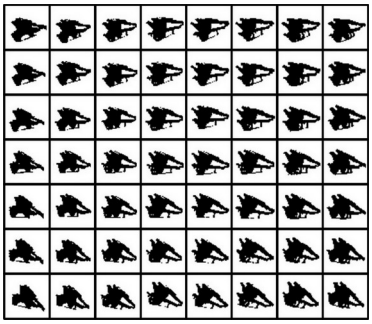

Fig. 3. Binary template images of one of the toys in 56 postures (7 rotation angles by 8 viewpoints)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| a5 | b5 | c5 | d5 | e5 | f5 | g5 | h5 |
| a3 | b3 | c3 | d3 | e3 | f3 | g3 | h3 |
| a1 | b1 | c1 | d1 | e1 | f1 | g1 | h1 |
| a | b | c | d | e | f | g | h |
| a2 | b2 | c2 | d2 | e2 | f2 | g2 | h2 |
| a4 | b4 | c4 | d4 | e4 | f4 | g4 | h4 |
| a6 | b6 | c6 | d6 | e6 | f6 | g6 | h6 |

Fig. 4. Notation for the 56 postures

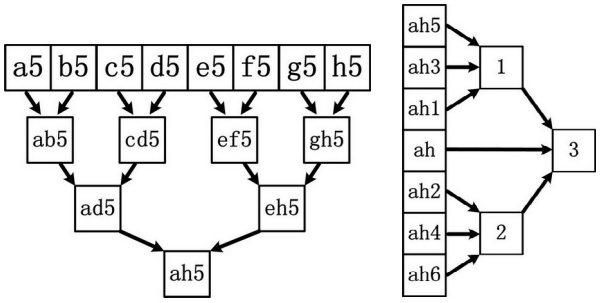


Fig. 5. Tree-structured combination of the 56 postures

In our experiments, the sizes of the detectors for each object at each scale are 20×13, 16×11, 21×13, 17×11, 17×10 and 18×9. In order to get the shapes as clearly as from infrared images, the toys are placed in front of white paper. The left picture in Fig. 6 demonstrates that the three toys can be detected simultaneously from a cluttered scene, while the right picture in Fig. 6 shows that no false positive occurs when the toys are not in the scene. The numbers in the bottom of the picture denote the heights of the toys in the image (or the heights of the detectors), while the number in the upper right corner of the picture represents the total number of objects (toys) detected from the image. Fig. 7 demonstrates that the objects can be detected from images under rotation and scale changes. Please note that the three toys are marked with different rectangles to differentiate them and denote the directions of their postures (toward left or toward right). Since the detectors for the third toy are designed on two scales (17×10 and 18×9), this toy can only be detected when its height is 8 to 11 pixels in the image. That is why it cannot be detected at a smaller size (7 pixels in height) in the right picture of Fig. 7.

## III. PEDESTRIAN DETECTION

Pedestrians in infrared images do have their shapes (Fig. 8), but these shapes are very flexible and hence cannot be detected

Fig. 6. Detection of 3 toys

Fig. 7. Detection of 3 toys with rotation and scale changes

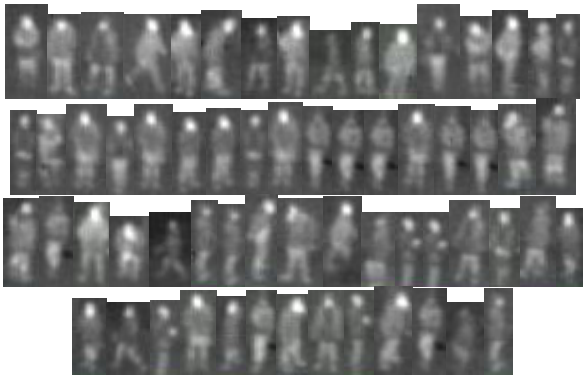

Fig. 8. Pedestrians in infrared images

with methods intended for rigid objects. Therefore, we use the Naïve Bayes method to detect pedestrians from infrared images with FPGA.

The general operational framework is similar with the SC method described in Section II. First, binary pixel stream is obtained with local thresholds. Then, a top sliding window is opened on the binary image. After that, detectors are applied in the top window to get detection results. The main difference is in the detectors where Naïve Bayes is used instead of Common Tree.

As shown in Fig. 9, three detectors are used to detect pedestrians of different sizes. The respective intended widths of pedestrians are listed in Table I. In order to gain more robustness against posture variations, only the upper body is used for training. Sixty binary images of the upper bodies of pedestrians are collected (Fig. 10) and horizontally flipped to get 120 binary images in all for training. The probability distribution obtained from these 120 images is shown in Fig. 11 in which brighter areas have higher probabilities of containing white pixels.

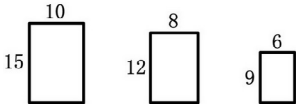


Fig. 9. Three pedestrian detectors

Fig. 10. Binary images of the upper bodies of pedestrians

Fig. 11. The probability distribution

Table I. Sizes of Detectors and the Intended Widths of Pedestrians

| Size of Detector (pixels) | Intended Width of Pedestrian (pixels) |
|---|---|
| 10×15 | 8~9 |
| 8×12 | 6~7 |
| 6×9 | 4~5 |

To facilitate the fixed-point number calculations in the FPGA, we turn the probabilities into fractional numbers with a unified denominator 128 and only use the numerators to construct the probability matrices. On this basis, the corresponding probability distribution matrices (PDMs) of Fig. 11 for the three detectors are given in Fig. 12, Fig. 13 and Fig. 14. Each element in these matrices represents the probability of a white pixel appearing in the corresponding position. By subtracting each element from 128, the three matrices can be turned into complementary matrices in which each element represents the probability of a black pixel appearing in that position.

Judging whether a pedestrian exists in the detector window is a two-class classification problem. For the positive class (pedestrians), the PDMs of white pixels are given in Fig. 12, Fig. 13 and Fig. 14. For the negative class (background), we assume that the probability of a pixel being white or black is equal, resulting in three PDMs in which every element is 64.

Naïve Bayes assumes that each pixel is independent. Although this assumption is not the truth, the performance of a Naïve Bayes classifier is surprisingly good because it is actually using a super ellipsoid to enwrap the positive class. On this basis, the product of many elements in the PDMs has to be computed to get the "likelihood". However, this will require lots of hardware resources in the FPGA. Therefore, we take the logarithm of each element in a PDM with base 2 and multiply it by 32 to get the logarithmic probability matrix (LPM) with fixed-point numbers and turn the multiplications into additions. For example, the corresponding LPM of Fig. 12 is given in Fig. 15.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 3 | 4 | 5 | 5 | 4 | 3 | 1 | 1 |
| 1 | 2 | 15 | 49 | 94 | 94 | 49 | 15 | 2 | 1 |
| 1 | 7 | 37 | 87 | 122 | 122 | 87 | 37 | 7 | 1 |
| 2 | 9 | 52 | 105 | 125 | 125 | 105 | 52 | 9 | 2 |
| 4 | 17 | 71 | 114 | 124 | 124 | 114 | 71 | 17 | 4 |
| 3 | 30 | 89 | 122 | 122 | 122 | 122 | 89 | 30 | 3 |
| 6 | 35 | 92 | 118 | 119 | 119 | 118 | 92 | 35 | 6 |
| 6 | 38 | 103 | 111 | 113 | 113 | 111 | 103 | 38 | 6 |
| 5 | 45 | 98 | 113 | 111 | 111 | 113 | 98 | 45 | 5 |
| 4 | 44 | 90 | 113 | 105 | 105 | 113 | 90 | 44 | 4 |
| 5 | 37 | 83 | 112 | 100 | 100 | 112 | 83 | 37 | 5 |
| 3 | 31 | 71 | 109 | 101 | 101 | 109 | 71 | 31 | 3 |
| 4 | 17 | 64 | 102 | 106 | 106 | 102 | 64 | 17 | 4 |
| 1 | 12 | 54 | 108 | 113 | 113 | 108 | 54 | 12 | 1 |
| 1 | 10 | 60 | 113 | 116 | 116 | 113 | 60 | 10 | 1 |

Fig. 12. Probability distribution matrix (10×15)

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 6 | 11 | 11 | 6 | 1 | 1 |
| 1 | 10 | 47 | 100 | 100 | 47 | 10 | 1 |
| 2 | 19 | 78 | 118 | 118 | 78 | 19 | 2 |
| 5 | 27 | 94 | 124 | 124 | 94 | 27 | 5 |
| 9 | 49 | 109 | 123 | 123 | 109 | 49 | 9 |
| 9 | 64 | 111 | 122 | 122 | 111 | 64 | 9 |
| 12 | 69 | 116 | 106 | 106 | 116 | 69 | 12 |
| 10 | 66 | 114 | 107 | 107 | 114 | 66 | 10 |
| 6 | 55 | 111 | 100 | 100 | 111 | 55 | 6 |
| 3 | 38 | 97 | 102 | 102 | 97 | 38 | 3 |
| 3 | 29 | 92 | 110 | 110 | 92 | 29 | 3 |
| 1 | 25 | 99 | 116 | 116 | 99 | 25 | 1 |

Fig. 13. Probability distribution matrix (8×12)

| | | | | | |
|---|---|---|---|---|---|
| 3 | 19 | 54 | 54 | 19 | 3 |
| 8 | 53 | 105 | 105 | 53 | 8 |
| 16 | 71 | 115 | 115 | 71 | 16 |
| 23 | 84 | 117 | 117 | 84 | 23 |
| 28 | 90 | 109 | 109 | 90 | 28 |
| 29 | 86 | 105 | 105 | 86 | 29 |
| 19 | 75 | 101 | 101 | 75 | 19 |
| 9 | 61 | 106 | 106 | 61 | 9 |
| 6 | 59 | 113 | 113 | 59 | 6 |

Fig. 14. Probability distribution matrix (6×9)

Since Fig. 15 is the LPM for white pixels in the 10×15 detector, we record it as LPMW(10×15). Correspondingly, the LPM for black pixels in the 10×15 detector is recorded as LPMB(10×15). Also, there are LPMW(8×12), LPMB(8×12), LPMW(6×9) and LPMB(6×9), which can be computed from Fig. 13 and Fig. 14.

For each white pixel in the detector window, the element in the corresponding position of LPMW is chosen; for each black pixel in the detector window, the element in the corresponding position of LPMB is chosen. By adding these elements, the likelihood of the content being a pedestrian is computed, which we record as $L_{pedestrian}$. Meanwhile, the likelihood of the content being background $L_{background}$ is a constant value for each detector window, because an even distribution of the pixel values is assumed for the negative class. The criteria for judging can then be written as

$$IsPedestrian = \begin{cases} 1, & L_{pedestrian} \geq L_{background} + K \\ 0, & L_{pedestrian} < L_{background} + K \end{cases} \quad (1)$$

where $K$ is an adjustable parameter representing the ratio of the prior probability of negative class to the prior probability of positive class.

After offline training with the 60 binary images in Fig. 10, the maximum, average and minimum of $L_{pedestrian}$ as well as $L_{background}$ for each of the 3 detectors are given in Table II. By subtracting $L_{background}$ from the minimum of $L_{pedestrian}$, we get the reference values of $K$ for each detector, namely 398, 266 and 59. However, since the 6×9 detector is too small, many false positives occur when $K$ takes on the value 59. Therefore, we raise this parameter to 350 according to experiments.

To further reduce false positives, gray value contrast is used as an additional judging criterion. Since the temperature of a pedestrian and the temperature of the background both have their respective range, the difference between the average gray value of a pedestrian and the average gray value of the ambient background in an infrared image is also in a certain range. By adding this criterion, false positives are reduced to a reasonable level.

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 51 | 64 | 74 | 74 | 64 | 51 | 0 | 0 |
| 0 | 32 | 125 | 180 | 210 | 210 | 180 | 125 | 32 | 0 |
| 0 | 90 | 167 | 206 | 222 | 222 | 206 | 167 | 90 | 0 |
| 32 | 101 | 182 | 215 | 223 | 223 | 215 | 182 | 101 | 32 |
| 64 | 131 | 197 | 219 | 223 | 223 | 219 | 197 | 131 | 64 |
| 51 | 157 | 207 | 222 | 222 | 222 | 222 | 207 | 157 | 51 |
| 83 | 164 | 209 | 220 | 221 | 221 | 220 | 209 | 164 | 83 |
| 83 | 168 | 214 | 217 | 218 | 218 | 217 | 214 | 168 | 83 |
| 74 | 176 | 212 | 218 | 217 | 217 | 218 | 212 | 176 | 74 |
| 64 | 175 | 208 | 218 | 215 | 215 | 218 | 208 | 175 | 64 |
| 74 | 167 | 204 | 218 | 213 | 213 | 218 | 204 | 167 | 74 |
| 51 | 159 | 197 | 217 | 213 | 213 | 217 | 197 | 159 | 51 |
| 64 | 131 | 192 | 214 | 215 | 215 | 214 | 192 | 131 | 64 |
| 0 | 115 | 184 | 216 | 218 | 218 | 216 | 184 | 115 | 0 |
| 0 | 106 | 189 | 218 | 219 | 219 | 218 | 189 | 106 | 0 |

Fig. 15. Logarithmic probability matrix for white pixels (10×15)

Table II. The Parameters for Each Detector

| Size of Detector (pixels) | 10×15 | 8×12 | 6×9 |
|---|---|---|---|
| **max**($L_{pedestrian}$) | 32224 | 20509 | 11240 |
| **average**($L_{pedestrian}$) | 31074 | 19846 | 10914 |
| **min**($L_{pedestrian}$) | 29198 | 18698 | 10427 |
| $L_{background}$ | **28800** | **18432** | **10368** |
| **min**($L_{pedestrian}$) − $L_{background}$ | 398 | 266 | 59 |
| Ratio of the Prior Probability ($K$) | **398** | **266** | **350** |

Outdoor experimental results are shown in Fig. 16 in which detected pedestrians are marked with rectangles. When two or more detected pedestrians are near to one another, they are marked with a common rectangle. The numbers in the upper right corners of the images represent how many rectangles there are in the image. Although the detectors are intended for pedestrians whose widths range from 4 to 9 pixels, they can actually detect pedestrians as wide as 12 pixels according to our experiments.

## IV. Physical System and FPGA Resource Usage

Fig. 17 shows the embedded image processing card developed by our research team. The FPGA on the card is Altera's Cyclone III EP3C40F484, and the DSP is TI's TMS320DM642. In addition, there are two 512K×16bits SRAMs (Static Random Access Memory), two 4M×32bits SDRAMs (Synchronous Dynamic Random Access Memory) and a 4M×8bits Flash Memory on the board. The working frequency of the DSP is 600 MHz, and the FPGA works with two clocks. The total power consumption of the image card is about 5 W, while the power consumption of the FPGA is less than 0.3 W.

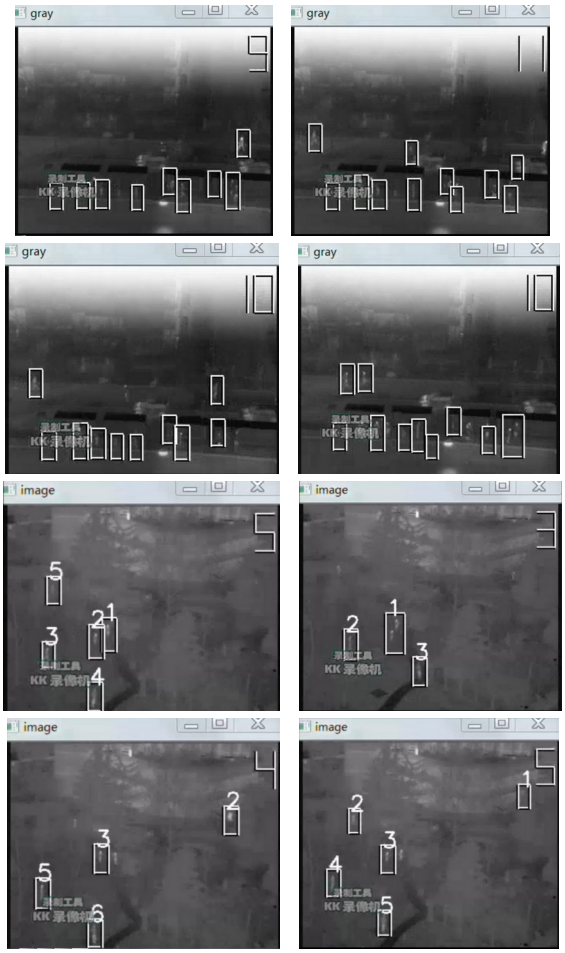

Fig. 16. Pedestrian detection from infrared images

Our current system runs with BT.656 PAL videos provided by analog cameras, and the image resolution is 360×288. Since most of the computation is performed on a hardware pipeline in the FPGA, the processing speed equals the image capturing speed which is 25 frames per second.

The FPGA resource usage for SC (Section II) is listed in Table III, while the FPGA resource usage for pedestrian detection is listed in Table IV.

## V. Conclusions

In this paper, we have presented two FPGA based methods for shape detection. The first method is named Shape Constraint and is intended for rigid object detection. The second method is based on Naïve Bayes and is intended for pedestrian detection from infrared images. Our experiments are done with analog cameras whose speed is 25 fps, and the image resolution is 360×288. However, as most of the computation is performed in the FPGA on a pipeline driven by a global clock, this design can also be applied to images with other resolutions and generate detection results for each frame in real time.

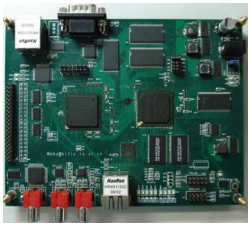
Fig. 17. Embedded image processing card (15.7×12.2 cm$^2$)

Table III. FPGA Resource Usage for Shape Constraint

| | Used Amount | Total Amount | Percentage |
|---|---|---|---|
| Logic Elements | 22,204 | 39,600 | 56% |
| RAM（bits） | 104,204 | 1,161,216 | 9% |
| Multiplier (9-bit) | 0 | 252 | 0% |

Table IV. FPGA Resource Usage for Pedestrian Detection

| | Used Amount | Total Amount | Percentage |
|---|---|---|---|
| Logic Elements | 4,997 | 39,600 | 13% |
| RAM（bits） | 103,756 | 1,161,216 | 9% |
| Multiplier (9-bit) | 0 | 252 | 0% |

Although the experimental results are impressive, false positives still exist in pedestrian detection, especially with the 8×12 detector and the 6×9 detector. Future improvements can take advantage of the fact that the head of a pedestrian is much brighter than other parts of the body as well as the background.

## REFERENCES


[1] M. H. Husin, F. Osman, M. F. M. Sabri, W. A. W. Z. Abidin, A. Othman, and A. S. W. Marzuki, "Development of Shape Pattern Recognition for FPGA-Based Object Tracking System," in *2010 International Conference on Computer Applications and Industrial Electronics (ICCAIE 2010)*, Kuala Lumpur, Malaysia, 2010, pp. 80-84.

[2] E. C. Pedrino, O. M. Jr., E. R. R. Kato, and V. O. Roda, "Intelligent FPGA Based System for Shape Recognition," in *2011 VII Southern Conference on Programmable Logic (SPL)*, Cordoba, Argentina, 2011, pp. 197-202.

[3] J. Martínez-Carballido, J. Guevara-Escobedo, and J. M. Ramírez-Cortés, "FPGA Design and Implementation for Vertex Extraction of Polygonal Shapes " in *21st International Conference on Electrical Communications and Computers (CONIELECOMP)*, San Andres Cholula, Puebla, Mexico, 2011, pp. 217-221.

[4] D. Kim, S. Jin, D. D. Nguyen, and J. W. Jeon, "Real-time Binary Shape Matching System Based on FPGA," in *2008 IEEE International Conference on Robotics and Biomimetics (ROBIO 2008)*, Bangkok, Thailand, 2009, pp. 1194-1199.

[5] J. Duan and W. MacDowell, "Pedestrian Detection Image Processing with FPGA," Bachelor of Science A Major Qualifying Project Report, Department of Electrical and Computer Engineering, WORCESTER POLYTECHNIC INSTITUTE, Worcester, Massachusetts, 2014.

[6] J. Xu, Y. Dou, J. Li, X. Zhou, and Q. Dou, "FPGA Accelerating Algorithms of Active Shape Model in People Tracking Applications " in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, Lubeck, 2007, pp. 432-435.

[7] H. Meng, K. Appiah, A. Hunter, and P. Dickinson, "FPGA Implementation of Naive Bayes Classifier for Visual Object Recognition," in *2011 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Colorado Springs, 2011.

[8] D. Nguyen, D. Halupka, P. Aarabi, and A. Sheikholeslami, "Real-Time Face Detection and Lip Feature Extraction Using Field-Programmable Gate Arrays," *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics,* vol. 36, pp. 902-912, August 2006.