International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.9, No.12, (2016), pp.261-270
http://dx.doi.org/10.14257/ijsip.2016.9.12.25

Research on Embedded Image Edge Detection Algorithm

Gang Li[1] and Ruixiang Huang[2]

[1]*Department of Computer Science, Jining Normal University, Ulanqab, China*
[2]*Department of Mechanical and Electrical Technology, Ulanqab Vocational College, Ulanqab, China*
jnsfxylg@126.com

Abstract

With the development of embedded technology, the embedded image processing algorithms are becoming more and more complex. Edge detection is the basic algorithm of image processing, which is used in the extraction of information from the image. Laplacian operator is useful for the edge detection. Especially, the operator can be determined by the zero crossing point of the two differential positive and negative peaks. In order to meet the fast hardware implementation of embedded image processing, this paper presents the hardware acceleration of Laplacian operator by Zynq-7000. Using HLS method, the complex image algorithm is automatically translated into hardware language, and the function of the algorithm can be quickly completed. The experimental results show that, the embedded hardware accelerated image based on Zynq-7000 can not only realize software programming and hardware programmable combination, but also improve the embedded system flexibility, scalability, and accelerate embedded image processing product design time.

Keywords: *Embedded system; Image Processing; Edge Detection*

1. Introduction

With the continuous development of embedded system technology, embedded image processing technology is becoming more and more complex [1-3]. Consequently, an embedded system is usually designed to perform one specific task, or a small range of specific tasks, often with real-time constraints. An obvious application of an embedded image processing system is a digital camera. There the imaging functions include exposure and focus control, displaying a preview, and managing image compression and decompression. Embedded image processing is also useful for smart vision, where the camera not only captures the image, but also processes it to extract information as required by the application. Examples of where this would be useful are intelligent surveillance systems, industrial inspection or control, robot vision, and so on.

A requirement of many embedded image processing systems is that they need to be of small size, and light weight. Many run off batteries, and are therefore required to operate with low power. Even those that are not battery operated usually have limited power available. However, acquisition speed of embedded image processing system is slow based on ARM in the front-end, and the image processing algorithm is not easy to use FPGA hardware acceleration. Therefore, in order to improve the acquisition speed of t image processing, Zynq chip is used as an embedded image processing algorithm in order to accelerate the FPGA hardware. In the context of Zynq devices, this means moving code from the ARM dual-core Cortex-A9 processor to the FPGA logic for acceleration [4].

The other parts of the paper are organized as follows: In Section 2, we introduce Zynq-7000 all programmable soc system and its advantages in embedded system design. Section 3 discusses Laplacian operator of image edge detection algorithm is

ISSN: 2005-4254 IJSIP

preferred which is most trustworthy and gives an efficient output in the field of image and video processing for the extraction of object edges. The main contribution of this paper is presented in Section 4, where edge detection algorithm is designed and verified using Zynq-7000. Finally, Section 5 summarizes the main conclusions of this work.

2. Introduction of Zynq-7000 All Programmable SoC

The Zynq-7000 family is based on the Xilinx All Programmable SoC architecture [5-6]. These products integrate a feature-rich dual-core ARM Cortex-A9 based processing system (PS) and Xilinx programmable logic (PL) in a single device and the basic structure is shown in Figure 1. The Cortex-A9 CPUs are the heart of the PS and also include on-chip memory, external memory interfaces, and a rich set of peripheral connectivity interfaces. The Zynq-7000 architecture enables implementation of custom logic in the PL and custom software in the PS. It allows for the realization of unique and differentiated system functions.

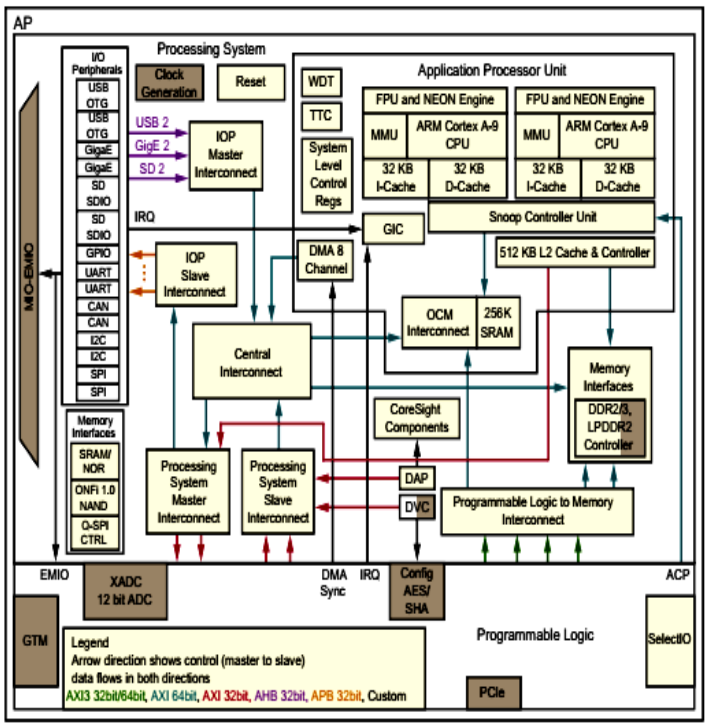


Figure 1. Zynq-7000 AP SoC Processing System Structure Block Diagram

The application processing unit (APU) consists of two ARM Cortex-A9 processor with a snoop control unit (SCU), which is responsible for maintaining the cache coherency between the two processors. Each processor has its own 32 KB level-one (L1) instruction and data caches, memory management unit (MMU), and separate media processing engine (NEON). L1 caches include two parts: instruction-side cache (I-Cache) and data-side cache (D-Cache). I-Cache is responsible for providing an instruction stream to the Cortex-A9 processor. D-Cache is responsible for holding the data used by the Cortex-A9 processor. The MMU in the ARM architecture involves both memory protection and address translation. The MMU works closely with the L1 and L2 memory systems in the process of translating virtual addresses to physical addresses. NEON is co-processor and extends the

Cortex-A9 to provide support for the ARM v7 advanced single instruction multiple data and vector floating-point instruction sets.

The PL is derived from Xilinx 7 series FPGA technology. The PL is used to extend the functionality to meet specific application requirements. The PL includes many different types of resources including configurable logic blocks, port and width configurable block RAM, DSP slices with a 25 x 18 multiplier, 48-bit accumulator and pre-adder, a user configurable analog to digital converter, clock management tiles.

3. Edge Detection

Edge detection is a fundamental image processing operation used in many computer vision solutions [7-8]. The goal of edge detection algorithms is to find the most relevant edges in an image or scene. These edges should then be connected into meaningful lines and boundaries, resulting in a segmented image containing two or more regions. Subsequent stages in a machine vision system will use the segmented results for tasks such as object counting, measuring, feature extraction, and classification.

Edge detection is a hard image processing problem. Most edge detection solutions exhibit limited performance in the presence of images containing real-world scenes, that is, images that have not been carefully controlled in their illumination, size and position of objects, and contrast between objects and background. Consequently, it is common to precede the edge detection stage with preprocessing operations such as noise reduction and illumination correction. Edge detection methods usually rely on calculations of the first or second derivative along the intensity profile. The first derivative has the desirable property of being directly proportional to the difference in intensity across the edge; consequently, the magnitude of the first derivative can be used to detect the presence of an edge at a certain point in the image. The sign of the second derivative can be used to determine whether a pixel lies on the dark or on the bright side of an edge. Moreover, the zero crossing between its positive and negative peaks can be used to locate the center of thick edges.

This is known as a discrete Laplacian. The laplacian has the advantage over first derivative methods in that it is an isotropic filter; this means it is invariant under rotation. That is, if the laplacian is applied to an image, and the image then rotated, the same result would be obtained if the image was rotated first, and the laplacian applied second [9]. This would appear to make this class of filters ideal for edge detection. However, a major problem with all second derivative filters is that they are very sensitive to noise, and the results of edge function are shown schematically in Figure 2.

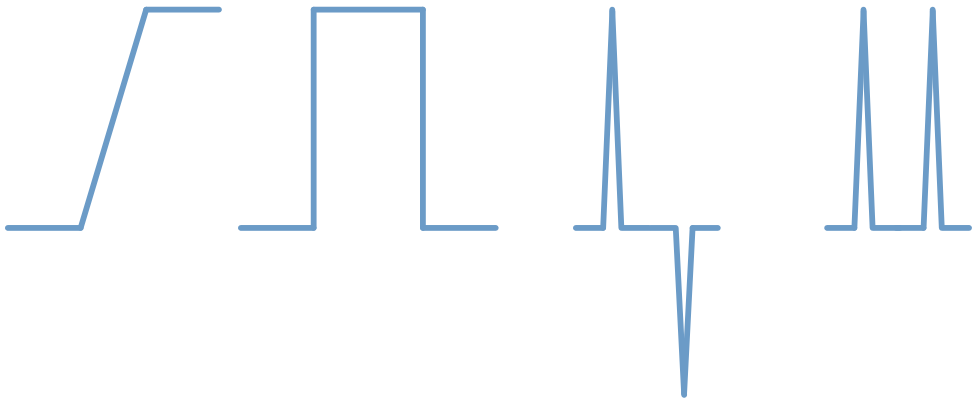


Figure 2. Second Derivatives of an Edge Function

The Laplacian of an image $f(x, y)$ is defined as

$$\nabla^2(x,y) = \frac{\partial^2(x,y)}{\partial x^2} + \frac{\partial^2(x,y)}{\partial y^2} \quad (1)$$

Where the second derivatives are usually approximated, and are given by

$$\frac{\partial^2(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y) \quad (2)$$

and

$$\frac{\partial^2(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y) \quad (3)$$

Results in a convenient expression for the Laplacian expressed as a sum of products:

$$\nabla^2(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \quad (4)$$

The Laplacian operator of image edge detection algorithm [10-12] is showed in Figure 3. Figure 3 (a), gives primitive template, and expand template is showed Figure 3 (b). Equation (4) can be implemented by the convolution mask, which is shown in Figure 3(c). An alternative digital implementation of the Laplacian takes into account all eight neighbors of the reference pixel in the input image and can be implemented by the convolution mask in Figure 3 (d).

0	1	0
1	-4	1
0	1	0

(a) Primitive Template

1	1	1
1	-8	1
1	1	1

(b) Expand Template

0	-1	0
-1	4	-1
0	-1	0

(c) Implementation Template

-1	-1	-1
-1	8	-1
-1	-1	-1

(d) Implementation Template

Figure 3. Laplacian Operator Template

We can see from Figure 3, if a bright spot appears in the dark area of the image, then the bright spot will become brighter using the Laplacian operator. This is because the edge of the image is the occurrence of gray jump area, so the Laplacian sharpening template is useful for the edge detection. General enhancement techniques are difficult to determine the position of the edge lines for steep edges and slowly varying edges. However, the operator can be determined by the zero crossing point of the two differential

positive and negative peaks. Therefore, the operator is especially suitable for the purpose of isolated points and isolated lines in the image. Figure 4, shows how MATLAB functions are used to implement the Laplacian operator.

(a) Original image (b) Laplacian "edge" (c) Sharpened image

Figure 4. Edge Sharpening Using the Laplacian Operator

4. Edge Detection Algorithm Hardware Accelerator via Zynq-7000

The Vivado HLS tool provides a methodology for migrating algorithms from a processor onto the FPGA logic. In the context of Zynq devices, this means moving code from the ARM dual-core Cortex-A9 processor to the FPGA logic for acceleration. The code implemented with the HLS tool in hardware represents the computational bottleneck of the algorithm.

Image input is generated by the image sensor from ON Semiconductor, which is configured for 1080p60 resolution. The raw Bayer sub-sampled image is converted to an RGB image by an image processing pipeline implemented using Laplacian operator that remove defective pixels, de-mosaic, and color-correct the image. An image buffer is implemented in the processing system (PS) DDR3 memory, making images accessible to the ARM processor cores via the AXI Video Direct Memory Access (VDMA). The image frame buffer is not required for the operation of the image processing pipeline, but is included in the design to enable the capture of input video images for analysis. Figure 5, shows a block diagram of the system.

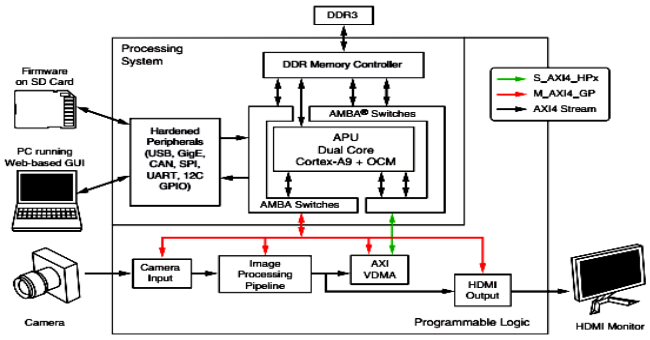


Figure 5. Image Processing System Design Block Diagram

4.1. Laplacian Operator by HLS Description

High-Level Synthesis (HLS) compiler provides the same functionality for C/C++ programs targeted to FPGA. HLS shares key technology with processor compilers for the interpretation, analysis, and optimization of C/C++ programs. The main difference is in the execution target of the application. By targeting an FPGA as the execution fabric, HLS enables a software engineer to optimize code for throughout, power, and latency

without the need to address the performance bottleneck of a single memory space and limited computational resources. This allows the implementation of computationally intensive software algorithms into actual products, not just functionality demonstrators.

Laplacian operator data flow diagram using HLS described as shown in Figure 6. We can see from the diagram, and the direction of the arrow indicates the dependence of the data. In each clock cycle, data parallel operations in the cell. A pixel represented by 24bit is entered into the algorithm module through the AXI4-Stream interface. First, the data is processed by gray scale. Then, these data are stored in the cache, and the data window is composed of the cache and the new data. In the end, the data is converted to 8 bit pixels, and the data is written back to the DDR3. Part of the code of the Laplacian operator by HLS description is shown in Figure 7.

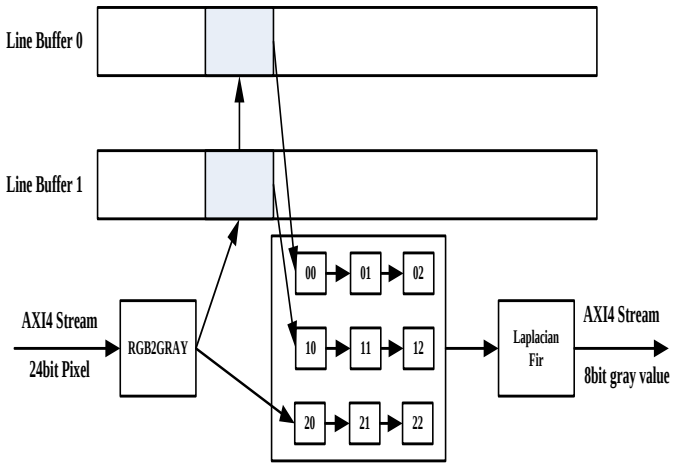


Figure 6. Laplacian Operator Data Description Block Diagram

```
for (int i=0;i<rows;i++)
{   for(int j=0;j<cols;j++)
    {   img_0>>pix_in;
        gray=rgb2gray(pix_in);
        win[0][0]=win[0][1];
        win[0][1]=win[0][2];
        win[0][2]=y_buf[0][i];
        win[1][0]=win[1][1];
        win[1][1]=win[1][2];
        win[1][2]=y_buf[1][i];
        win[2][0]=win[2][1];
        win[2][1]=win[2][2];
        win[2][2]=gray;
        y_buf[0][i]=y_buf[1][i];
        y_buf[1][i]=gray;
    laplasian=(*lap00)*win[0][0]+(*lap01)*win[0][1]+(*lap02)*win[0][2]
              +(*lap10)*win[1][0]+(*lap11)*win[1][1]+(*lap12)*win[1][2]
              +(*lap20)*win[2][0]+(*lap21)*win[2][1]+(*lap22)*win[2][2];
    }
}
```

Figure 7. Part of the Code of Laplacian Operator by HLS Description

4.2. Laplacian Operator Verification by HLS Description

This design is divided into two steps to verify the Laplacian operator. The first step is the C language simulation, using HLS design tools to complete the functional verification of the operator. The second step is the co simulation of C and RTL. In this step, according to the testing code written in C language, HDL testing vector is generated by HLS tools. Based on the generated HDL testing vector, the RTL level simulation is performed to

verify the correctness of the Laplacian operator. Part of the testing code is shown in Figure 8,

```
int main (int argc,char ** argv)
{
  IplImage * src=cvLoadImage("test_image");
  IplImage * dst=cvCreatImage(cvGetSize(src),src->depth,src->nChannels);
  ap_int<8> lap00=-1;ap_int<8> lap01=-1;ap_int<8> lap02=-1;
  ap_int<8> lap10=-1;ap_int<8> lap11=-8;ap_int<8> lap12=-1;
  ap_int<8> lap20=-1;ap_int<8> lap21=-1;ap_int<8> lap22=-1;
  AXI_STREAM src_axi,dst_axi;
  IplImage2AXIvideo (src,src_axi);
  laplasian_filter(&lap00,&lap01,&lap02,&lap00,&lap10,&lap11,&lap12,&lap20,
                   &lap21,&lap22,src_axi,dst_axi);
  AXIvideoIplImage (dst_axi,dst);
  cvSaveImage("test_image",dst);
  cvSaveReleaseImage(&src);
  cvSaveReleaseImage(&dst);
  return 0;
}
```

Figure 8. Part of the Testing Code of Laplacian Operator

The hardware platform is shown in Figure 9. The input and output of the image, respectively, are the HDMI interface and VGA interface. The image is displayed in the HDMI interface, and edge detection image is displayed in the VGA interface. The experimental results are shown in Figure 10.

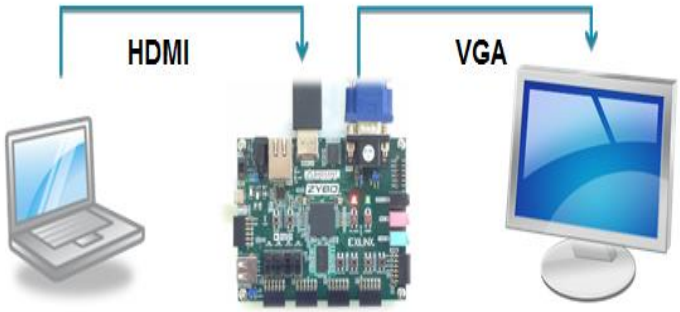
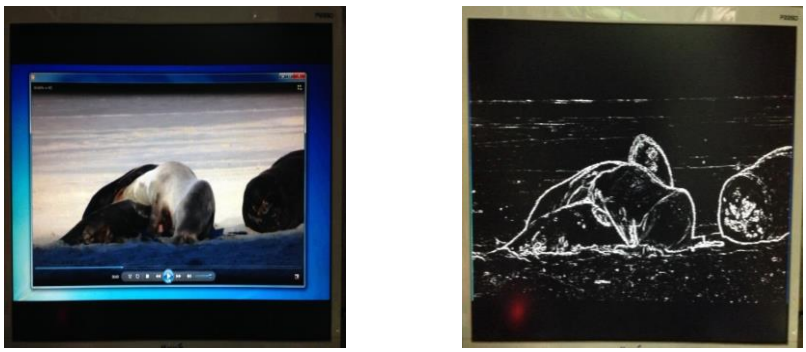


Figure 9. Experimental Hardware Testing Platform



(a) Original Image (b) Laplacian "Edge"

Figure 10. Experimental Hardware Testing Results

Laplacian operator is implemented by Vivado HLS tool, and the hardware consumption of resources is shown in Table 1. We can see from Table 1, Zynq XC7Z020 SoC can be completely achieved Laplacian operator.

Table 1. Hardware Resource Consumption of Laplacian Operator

Name	BRAM_18K	DSP48E	FF	LUT
Expression	-	-	-	-
FIFO	0	-	73	280
Instance	2	12	910	580
Memory	-	-	-	-
Multiplexer	-	-	-	-
Register	-	-	6	-
Total	2	12	1007	884
Available	120	82	35200	17600
Utilization	1	13	2	5

5. Conclusion

Laplacian operator is useful for the edge detection. General enhancement techniques are difficult to determine the position of the edge lines for steep edges and slowly varying edges. However, the operator can be determined by the zero crossing point of the two differential positive and negative peaks. Therefore, Laplacian operator is especially suitable for the purpose of isolated points and isolated lines in the image. However, Laplacian edge detection algorithm is a very complex process, and the hardware implementation is very difficult. In order to meet the fast hardware implementation of image processing algorithms, Zynq-7000 SOC is used as a hardware implementation by Vivado HLS tool. The measured results of the hardware of the operator indicated that, the embedded hardware accelerated image based on Zynq-7000 can not only realize software programming and hardware programmable combination, but also improve the embedded system flexibility, scalability. These advantages accelerate embedded image processing product design time.

Acknowledgments

We would like to thank professor Wang for stimulating discussions with respect to the topic of this paper and laboratory equipment. Moreover, we greatly appreciate the reviewers' comments that lead to an improved presentation of the results.

References

[1] L. Maggiani, C. Bourrasset, M. Petracca, F. Berry, P. Pagano and C. Salvadori, "HOG-Dot: A Parallel Kernel-Based Gradient Extraction for Embedded Image Processing", IEEE Signal Processing Letters., vol. 22, no. 11, **(2015)**, pp. 2132-2136.

[2] R. Lerm, D. Doering, R. H. A. Rech, A. Rettberg and C. E. Pereira, "A model-based design space exploration for embedded image processing in industrial applications", Proceedings of the 12th International Conference on Industrial Informatics, Porto Alegre, Brazil, **(2014)** July 27-30.

[3] L. Thieling, A. Schuer, G. Hartung and G. Buchel, "Embedded image processing system for cloud-based applications", Proceedings of the 21th International Conference on Systems, Signals and Image Processing, Dubrovnik, Croatia, **(2014)** May 12-15.

[4] J. Silva, V. Sklyarov and I. Skliarova, "Comparison of On-chip Communications in Zynq-7000 All Programmable Systems-on-Chip", IEEE Signal Processing Letters., vol. 7, no. 1, **(2015)**, pp. 31-34.

[5] Xilinx, "Zynq-7000 All Programmable SoC Technical Reference Manual", [Online]. Available: http://www.xilinx.com/support/documentation/user_guides/ug585-Zynq-7000-TRM.pdf.

[6] Y. Nie, Z. Ma and L. Jing, "Research on the Design of Multi-Core Embedded System Based on Microblaze", International Journal of Control & Automation., vol. 8, no. 12, **(2015)**, pp. 425-434.

[7] A. Aquino, M. E. Gegúndez-Arias and D. Marín, "Detecting the optic disc boundary in digital fundus images using morphological, edge detection, and feature extraction techniques", IEEE Transactions on Medical Imaging., vol. 29, no. 11, **(2010)**, pp. 1860-1869.
[8] L. Xu and Y. Li, "Multi-Scale Edge Detection of Rice Internal Damage Based on Computer Vision", Proceedings of International Conference on Automation and Logistics, Qingdao, China, **(2008)** September 1-3.
[9] S. G. Javed, A. Majid and N. Kausar, "Combining Robust Statistical and 1D Laplacian Operators Using Genetic Programming to Detect and Remove Impulse Noise from Images", Proceedings of the 13th International Conference on Frontiers of Information Technology, Islamabad, Pakistan, **(2015)** December 1-3.
[10] S. A. Coleman, B. W. Scotney and S. Suganthan, "Edge detecting for range data using Laplacian operators", IEEE Transactions on Image Processing., vol. 19, no. 11, **(2010)**, pp. 2814-2824.
[11] S. C. Tai and S. M. Yang, "A fast method for image noise estimation using Laplacian operator and adaptive edge detection", Proceedings of the 3th International Symposium on International Symposium on International Symposium on Communications, Control, and Signal Processing, St. Julians, Malta, **(2008)** March 12-14.
[12] B. Gardiner, S. A. Coleman and B. W. Scotney, "Multiscale Edge Detection Using a Finite Element Framework for Hexagonal Pixel-Based Images", IEEE Transactions on Image Processing., vol. 25, no. 4, **(2016)**, pp. 1849-1861.

Authors

Gang Li, he is a lecturer of department of Computer Science at Jining Normal University, China. He received the M.S. degree in College of Computer Science at Inner Mongolia University, China. His main research interests are image processing, education informatization, computer network.

Ruixiang Huang, she is a lecturer of department of Mechanical and Electrical Technology at Ulanqab Vocational College, China. She received the B.S. degree in Inner Mongolia Normal University in 2005, China. Her main research interests are image processing, electronic technology, UAV applications.