

DESIGN FPGA-BASED MOVING OBJECT RECOGNITION

Qianqian Liu^{*1}, Han Diao², Tao Liu¹, Wendong Lv³

¹Naval University of Engineering, Wuhan, 430033, China

²School of Cyberspace Security, Qufu Normal University, Qufu, 273100, China

³Yantai Haihua Electric Power Technology Co., Ltd., Yantai 264000, China

Corresponding author: Qianqian Liu (Email:0909042008@nue.edu.cn).

ABSTRACT With the development of intelligent monitoring, robot vision and IoT technologies, the demand for real-time and low-power mobile object recognition is growing. Traditional CPU/GPU schemes suffer from high latency and energy consumption on edge devices, while FPGA, with high-speed parallel computing and reconfigurable features, becomes ideal for efficient real-time visual process. The paper presents an FPGA-based mobile object recognition system, realizing recognition and tracking. It optimizes traditional algorithms via hardware acceleration to achieve low-latency, cost-effective real-time detection. The frame difference method extracts foreground targets, combined with morphological filtering for denoising. DDR3 stores images, and HDMI outputs videos. Leveraging FPGA's parallelism, hardware pipeline designs for preprocessing, motion and target detection modules are implemented with Verilog. Modules like ov5640 acquisition, DDR3 control, image processing and HDMI display are developed, finally achieving real-time recognition and frame tracking of moving objects. It provides an efficient hardware solution for low-cost, low-power applications, verifying FPGA's engineering value in visual processing.

INDEX TERMS FPGA; Moving object recognition; Hardware acceleration; Frame difference method; Image acquisition.

I. INTRODUCTION

WITH the rapid development of the Internet of Things (IoT), intelligent monitoring, and robot vision, edge visual perception has become a core technology. Within this domain, Moving Object Detection (MOD) faces dual challenges: the need for real-time processing and strict power consumption constraints.

Traditional schemes relying on CPUs and GPUs struggle to meet these demands in practical applications. While CPUs offer flexibility, their serial processing architecture leads to high latency when handling massive video data. Conversely, although GPUs provide strong parallel computing capabilities, they suffer from high power consumption, high costs, and rigid hardware architectures, making them unsuitable for resource-constrained edge devices. To address these limitations, Field-Programmable Gate Arrays (FPGAs) have emerged as a promising alternative. FPGAs offer parallel processing, low latency, and high energy efficiency, providing an ideal foundation for high-performance target detection systems [1-2].

Moving object recognition generally involves target detection and tracking[3-4]. Common algorithms include the frame difference method, background subtraction and optical flow [5-6]. Among these, the optical flow method is computationally intensive and difficult to implement efficiently on hardware. Background subtraction is effective for static

scenes but sensitive to environmental noise. In contrast, the frame difference method offers low computational complexity and strong real-time performance, making it highly suitable for hardware implementation. Consequently, this study focuses on implementing a moving object recognition system based on the frame difference method using an FPGA platform.

Ding wenlong, Yin duo, Qiu song et al. improved the three frames difference method [7] to eliminate the "hole" phenomenon. Feng tianqiao, Mao zhengyu, Peng xiangqian et al. verified that the detection performance of FPGA which fully meets the requirements of high-speed real-time performance is better than that of pc [8]. M. H. salehzadeh, V. Rostami, M. salamati and M. H. monfared proposed the improvement of Tiny Yolo neural network [9] to enable Tiny Yolo neural network to complete moving object detection. Jai Gopal Pandey proposed a high-performance design for mean shift-based moving object tracking implemented on an FPGA-SoC framework [10]. However, despite these advancements, existing methods often face trade-offs between computational complexity and detection accuracy.

Neural network approaches like Tiny YOLO require significant logic resources, which can be challenging for entry-level FPGAs, while traditional background subtraction methods may struggle with dynamic noise or ghosting artifacts in complex scenes without further optimization.

Therefore, there is a need for a lightweight yet robust algorithm that maximizes the hardware advantages of FPGAs. To bridge this gap, this paper proposes an efficient mobile object recognition system based on the frame difference method, specifically optimized for FPGA implementation. Unlike complex neural networks, the chosen frame difference method possesses low computational complexity and strong real-time performance, making it highly suitable for hardware acceleration. The proposed system is built upon a Transcender S6 development board equipped with a Xilinx Spartan-6 FPGA and an OV5640 camera, utilizing the ISE development environment. By leveraging the parallel processing capabilities of the FPGA, the system achieves real-time output of recognized images through an HDMI interface at a resolution of 1024×768 @ 60Hz. Experimental results demonstrate that the designed system not only meets the requirements for high-speed real-time processing but also accurately recognizes moving objects with minimal resource overhead, offering a cost-effective alternative to GPU-based solutions.

II. DESIGN OF IMAGE PROCESSING ALGORITHM

A. RGB TO GRAYSCALE

The formula for converting RGB to YCbCr is as follows:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.172 & 0.339 & 0.511 \\ 0.511 & 0.428 & 0.083 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 128 \\ 128 \end{bmatrix}. \quad (1)$$

Among them, R, G, and B represent the red, green, and blue components in RGB images, while Y, Cb, and Cr represent the grayscale components, hue, and saturation.

Since FPGA is not good at handling floating-point numbers, the coefficients are first enlarged by 256 times, and finally divided by 256 after calculation, which is shifted to the right by 8 bits. The calculation formula is as follows:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \left\{ \begin{bmatrix} 77 & 150 & 29 \\ -43 & 85 & 128 \\ 128 & 107 & 21 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 0 \\ 32768 \\ 32768 \end{bmatrix} \right\} \gg 8. \quad (2)$$

After converting an RGB image to a YCbCr image, take the Y component, which is the grayscale value. In terms of design, a three-stage pipeline design is adopted, which is completed through three operations: coefficient multiplication, component addition, and data shift. It takes three clock cycles for each pixel to complete the calculation. Therefore, it is necessary to beat the row field synchronization signal and the data effective signal three times to avoid signal metastability and ensure correct timing alignment.

B. FRAME DIFFERENCE ALGORITHM

The frame difference method [11] utilizes the difference in pixels between adjacent frames to detect moving targets. Compare its absolute value with the frame difference threshold and obtain this difference value during the effective signal period of the image. Read two consecutive frames of images from the video stream, denoted as the current frame



FIGURE 1. Palm frame differential image

$I_n(x, y)$ and the previous frame $I_{n-1}(x, y)$, where (x, y) represents the coordinates of pixels in the image. Differential operation is performed on the grayscale values of the corresponding pixels in the current frame and the previous frame to obtain the frame difference image $D(x, y)$. The calculation formula is:

$$D(x, y) = |I_n(x, y) - I_{n-1}(x, y)|. \quad (3)$$

Select a suitable threshold T and perform binary processing on the frame difference image $D(x, y)$. If the difference between a pixel in $D(x, y)$ is greater than the threshold T , then the pixel is marked as a foreground (moving target) pixel with a value of 1; Otherwise, it is marked as a background pixel with a value of 0, resulting in the binary image $B(x, y)$, that is:

$$B(x, y) = \begin{cases} 1, & D(x, y) \geq T, \\ 0, & D(x, y) < T. \end{cases} \quad (4)$$

The selection of the frame difference threshold (T) is critical for balancing detection sensitivity and noise suppression. Rather than relying on arbitrary values, the optimal threshold was determined empirically through a series of controlled indoor experiments. The tuning process involved evaluating the system's performance under varying lighting conditions and background complexities. During the testing phase, we observed a clear trade-off: setting T too low made the system overly sensitive to minor illumination changes and sensor noise, resulting in a high rate of false positives (background noise). Conversely, setting T too high caused the system to miss faint moving objects or fragment the contours of valid targets. Through iterative testing and visual evaluation, the threshold was finally tuned to 20. At this specific value, the system achieved the optimal balance, effectively filtering out static background noise while maintaining high integrity for moving object contours in our indoor test environment.

C. IMAGE EROSION AND IMAGE DILATION

Image erosion is a morphological image processing operation primarily used to reduce object regions or refine object boundaries in an image[12]. Image erosion is based on the morphological concept in set theory, and its core idea is to

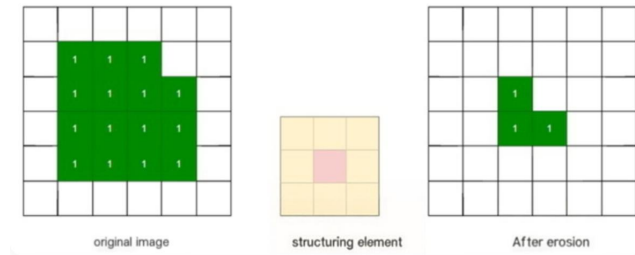


FIGURE 2. Schematic diagram of image erosion

use a structural element (also known as a kernel) to scan each pixel in the image. Structuring elements are typically small matrices with specific shapes (such as rectangles, circles, crosses, etc.) and sizes. For binary images, the erosion operation checks whether all pixels in the area covered by structural elements are foreground pixels (usually 1). If so, the center pixel is kept as the foreground pixel, otherwise it is set as the background pixel (usually 0), as shown in Figure 2.

Image erosion can eliminate isolated noise points in the image, as these noise points will be removed under the scanning of structural elements due to the surrounding background pixels. In addition, image erosion can also cause the boundaries of objects to shrink inward, resulting in the refinement of object shapes. In addition, when objects in the image adhere to each other, erosion operations can to some extent separate them. Through continuous erosion, the adhered parts may break, allowing for subsequent analysis and processing of individual objects.

Image dilation is a morphological image processing operation corresponding to image erosion, mainly used to enlarge the object area or connect the broken parts of objects in the image [13]. Image dilation is also based on structural elements to operate the image. The principle is that for each pixel in the image, when the center of the structural element is located at that pixel, if any pixel in the area covered by the structural element is a foreground pixel (usually 1 in binary images), then the current pixel is set as the foreground pixel, as shown in Figure 3.

In images, when objects break due to noise or other reasons, dilation can connect these broken parts by enlarging the boundaries of the object, making the shape of the object more complete. In addition, the boundaries of objects can be expanded outward to highlight their contours, which helps to better detect and locate objects in some image recognition and analysis tasks. For some small holes or voids inside an object, expansion operation can fill them to a certain extent, making the interior of the object more complete.

The operation of first corroding and then expanding an image is called Morphological opening, which is a morphological based image processing method[14]. First, use erosion operation to scan the image through structural elements, shrink the boundaries of the object inward, and remove isolated noise points and small details in the image. Further expansion operation is performed to expand the

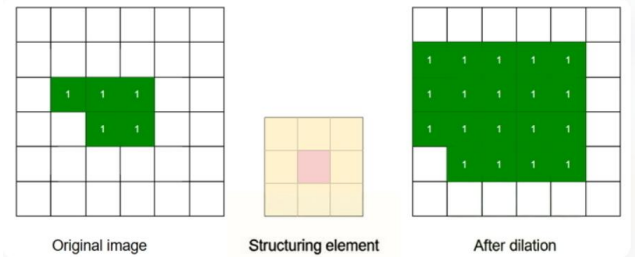


FIGURE 3. Schematic diagram of image dilation

boundary of the corroded object outward, allowing the approximate shape of the object to be restored. Due to the removal of some noise and small interference during the erosion process, these parts will not be restored during expansion, thus achieving the goal of removing noise and maintaining the shape of the object.

In the morphological processing stage, a 3×3 square structuring element was employed to perform opening or closing operations. The primary objective of this step is to eliminate isolated noise pixels and smooth the boundaries of the detected foreground objects. The 3×3 dimension was selected as an optimal trade-off between noise suppression and detail preservation. Empirical observations indicated that background noise typically spans no more than 1 to 2 pixels; thus, a 3×3 structuring element is sufficient to remove these artifacts. Larger structuring elements (e.g., 5×5 or 7×7) were tested but resulted in excessive erosion of small targets and the loss of fine structural details.

Therefore, the 3×3 structuring element provides the most robust performance for the given spatial resolution.

D. TARGET RECOGNITION ALGORITHM

After performing a Morphological opening on the binary image generated by the frame difference module, the pixel points of one frame of the image will be sequentially sent to the target recognition module. This module calculates the coordinates of pixels through a counter and updates the boundary coordinate register according to the values and coordinates of the pixels, thereby achieving the acquisition of the rectangular boundary coordinates of the motion area as shown in Figure 4.

E. FPGA DESIGN

This design realizes the recognition and tracking function of moving objects, including the design of image acquisition module, image processing module, DDR control module, clock module, and HDMI module. The clock module generates 100M and 300M clocks from the 50M clock on the board through PLL to drive the MIG core and DDR3 chip. After the camera completes initialization, the collected data is first transmitted to the image acquisition module, which changes the original 8-bit data to 16 Bit data, and then enters the image processing module. The image processing module outputs a grayscale value image data for frame difference

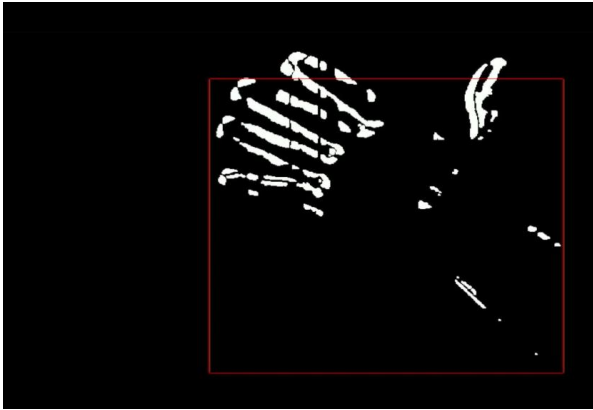


FIGURE 4. Motion Area Recognition

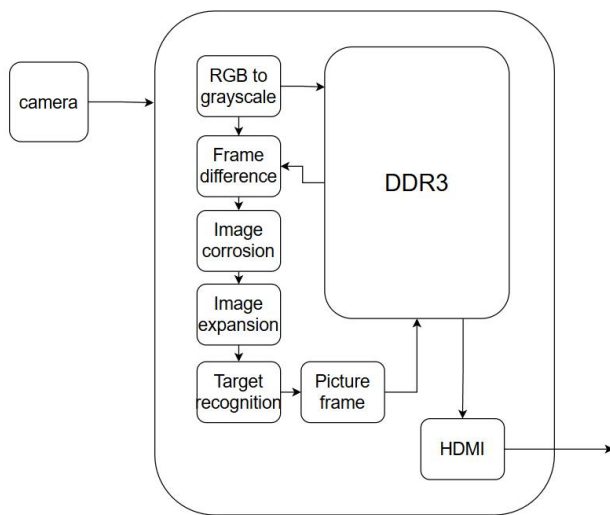


FIGURE 5. Schematic diagram of the overall process

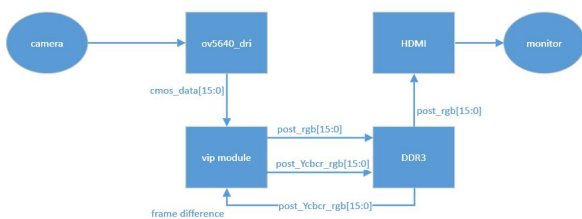


FIGURE 6. Module-level Data Flow Diagram

operation, and outputs a final image. The final image cached in DDR3 is output through the HDMI interface. The overall process is shown in Figure 5.

To ensure timing consistency and simplify the overall design, the image acquisition, real-time image processing, and HDMI display modules are integrated into a unified 50 MHz synchronous clock domain. This architecture effectively eliminates potential metastability issues during pixel-level

data transmission across the video pipeline. In contrast, the external DDR3 SDRAM operates at a high frequency of 300 MHz, with its memory controller interface (MIG) running at an independent 100 MHz clock domain. To bridge the 50 MHz video pipeline and the 100 MHz memory controller, Asynchronous FIFOs are implemented as the data buffering interface. These FIFOs not only resolve the Clock Domain Crossing challenges but also perform essential data width conversion and rate matching. This mechanism facilitates efficient burst read/write operations, thereby maximizing the bandwidth utilization of the DDR3 memory while ensuring data integrity.

F. IMAGE ACQUISITION MODULE

The main function of the image acquisition module is to configure the camera using the IIC (Inter-Integrated Circuit) protocol to complete initialization, and receive the video signal output by the camera, which is then used by the image processing module. It mainly includes the IIC configuration module, IIC driver module, and image data module. The RTL structure is shown in Figure 7. The OV5640 adopts a two-wire SCCB (Serial Camera Control Bus) interface bus compatible with the IIC protocol, so the camera can be directly configured using the IIC driver. To ensure the normal operation of the OV5640 sensor, the initialization operation must be completed first. The process mainly involves configuring the parameters of the internal registers of the sensor, defining its working mode by writing specific values to the registers, such as image resolution, data frame rate, color encoding format, and other key parameters. Only by completing the configuration of these registers can the sensor perform image acquisition according to the expected working mode, ensuring that the output image data meets the requirements of the system. The main function of the IIC configuration module is to provide the data and address information required for register read and write to the driver module, while managing the timing control of the initialization process. This module sends configuration instructions to the sensor through the IIC bus, including register addresses and corresponding data values, to configure the working mode of the sensor. In addition, the module is responsible for generating control signals for the initialization process, ensuring that the configuration of each stage is executed in order, and ultimately outputting the initialization completion signal. According to the OV5640 data manual, when the image sensor is powered on and reaches the initial configuration register, in order to reset all registers to their default state, a delay of 20ms is required. After 20ms, the sensor is configured and the software is reset. Subsequently, register configuration is carried out for the line field resolution, total line field pixels, and output format that the camera needs to output.

The IIC driver module is used to drive the read and write operations of IIC. Since it is only used to configure camera registers in this project, only write operations are required. The IIC execution signal is raised by the IIC

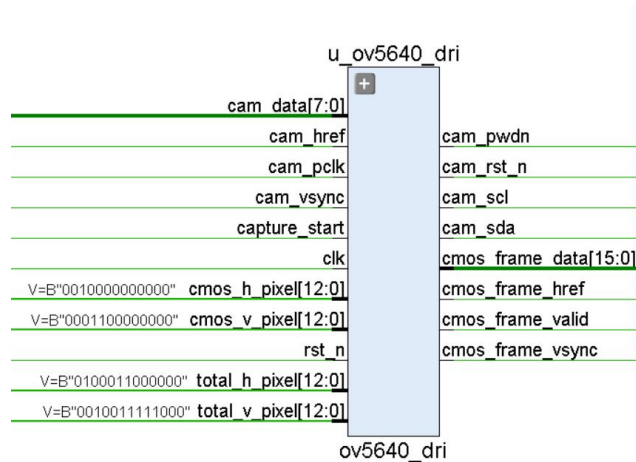


FIGURE 7. RTL structure diagram of image acquisition module

configuration module after a 20ms power on delay, and lowered after all registers are configured. IIC read and write operations are implemented using a state machine. In IIC communication, both byte write and random read operations start in an idle state. In the initial stage, a start signal needs to be sent first, followed by the transmission of device addresses and read/write instructions. After completing the transmission of device address and read/write instructions and receiving the response signal, send the word address and then proceed with the data read/write transmission operation. After the completion of data transmission for reading and writing, a response signal is received, and finally the signal transmission is stopped. At this point, the reading and writing operation is completed, and the idle state is entered again.

The image data module is driven by the pixel clock, and its main function is to convert the 8-bit data output by the camera into a 16 Bit data signal, and provide corresponding row field synchronization signals based on the row field synchronization signals output by the camera for subsequent image processing and DDR control modules. Considering that the effective time of register configuration can reach up to 300ms, which is close to the duration of the camera outputting 10 frames of image data, frame counting can be achieved by monitoring the rising edge of the field synchronization signal. When the counter value accumulates more than 10 times, the system will automatically generate a valid data flag and start the image acquisition process immediately.

G. IMAGE PROCESSING MODULE

The main functions of the image processing module include converting formatted color images into grayscale images (represented by the brightness Y in the YCbCr color space), performing frame differencing on two consecutive frames of images, performing image opening operations on the differenced images, detecting moving areas, and framing moving areas. And based on this, the RGB to grayscale

module, frame difference module, image erosion module, image dilation module, recognition module, and frame module were designed, and their RTL structures are shown in Figure 8.

The RGB to grayscale module converts the color RGB data collected by the camera into grayscale images, in order to facilitate the subtraction of data during subsequent frame differencing operations. The data conversion operation adopts a three-stage pipeline design to achieve hardware acceleration, and finally shifts the row field synchronization signal and enable signal by three clocks to ensure the correctness of timing. The converted data is sent one way to the frame differential module, and the other way is sent to the DDR3 chip cache through the top-level module, and then sent to the frame differential module, so that the two data obtained in the frame differential module are adjacent frames.

The main function of the frame difference module is to perform frame difference operations on two sets of data separately, provided by the RGB to grayscale module and the DDR3 control module. The DDR3 chip cache transfers the previous frame image, and the RGB to grayscale module outputs the current frame image. The two sets of images are differentiated pixel by pixel, and by comparing the difference with a pre-set threshold, a binary image is finally obtained that distinguishes the background from the moving object, thereby achieving the distinction between the background and the moving object.

The image erosion module and image dilation module jointly complete the opening operation of binary images, mainly to eliminate noise, eliminate small objects, and reduce the interference of noise and small objects on the recognition of motion areas.

The main function of the recognition module is to obtain the extreme coordinates of the moving object area, count the input pixels to obtain coordinates, determine whether the pixel is a moving area, update the coordinates of the moving object area, and finally obtain the boundary coordinates of a set of rectangular boxes.

The frame module uses a red rectangular box to mark the moving object area on the input RGB color image and outputs it as the final image. This module covers the RGB values of pixels located on the boundary based on the boundary coordinates obtained by the recognition module, turning them red to achieve the function of framing the moving object area.

H. DDR3 CONTROL MODULE

The DDR3 control module uses MIG (Memory Interface Generator) IP verification to read and write to the onboard DDR3 chip. The read and write logic is provided by the submodule DDR read and write module, and the read and write data is cached through FIFO, thus achieving the function of caching the image data output by the image processing module. Due to the image resolution captured by the camera being set to 1024 * 768, the data volume of

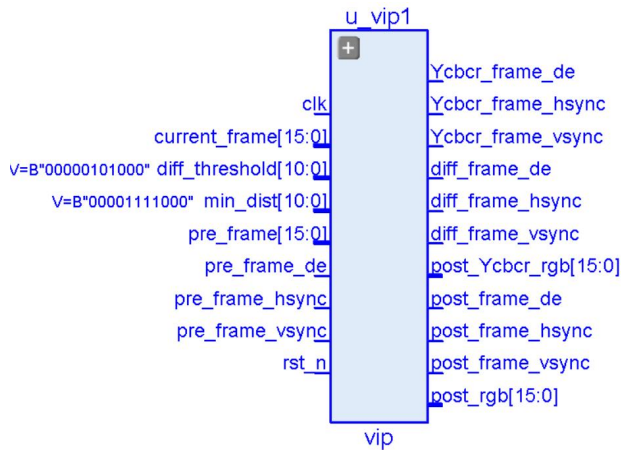


FIGURE 8. RTL architecture diagram of image processing module

one frame of image is $1024 \times 768 \times 16 \text{ bits} = 12582912 \text{ bits} = 12888 \text{ Kbit} = 12 \text{ Mbit}$, and the bandwidth required for data transmission is $12 \text{ Mbit} \times 65 \text{ Mhz} = 780 \text{ Mbit/s}$. The chip model of the Transcender FPGA development board is XC6SLX16-2FTG256. From the Spartan6 series device manual provided by Xilinx, it can be found that the on-chip storage resources of XC6SLX16 are 576 Kbit, far from meeting the storage requirements. Therefore, onboard DDR3 external memory is used to achieve image data caching. The development board of Beyond is equipped with the South Asian NT5CB128M16CP-DI DDR3 chip, with a storage capacity of 2Gb (2048 Mbit) and a peak bandwidth of 12.8 Gbit/s, fully meeting the requirements of video image caching. This module constructs a FIFO like user operation interface by encapsulating the complex readwrite logic of the MIG IP core (memory interface generator). Its core functional modules include DDR readwrite control unit, MIG controller IP core, and FIFO timing management module. The specific RTL architecture is shown in Figure 8. Due to the fact that the clock frequency for DDR reading and writing is much higher than that for image acquisition, image processing, and image output, in order to solve the problem of data transmission across clock domains, an asynchronous FIFO is designed and used to read and write DDR3 chips through MIG verification. The two sets of data that need to be cached, namely one frame of grayscale value image required for frame difference and the final image output after frame completion, are read and written through FIFO using timedivision multiplexing method.

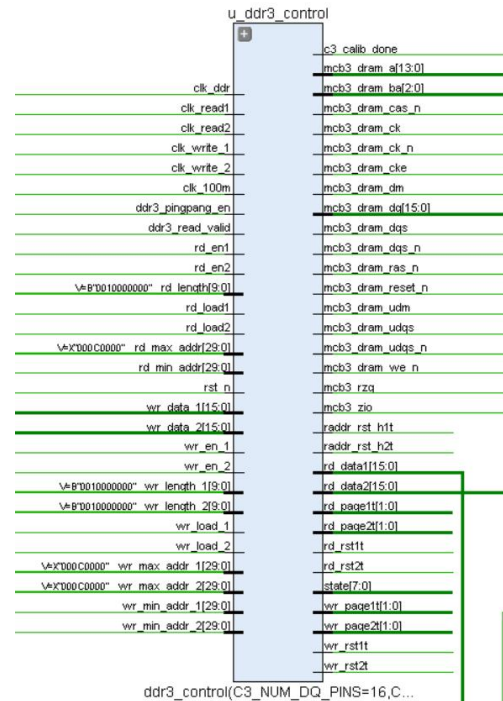


FIGURE 9. RTL structure diagram of DDR3 control module



FIGURE 10. Image acquisition

III. TEST RESULT

To verify the system's functionality, the output images from the development board are transmitted to a computer via a video capture card. OBS Studio is then utilized to capture and visually verify the original, differential, and final output images. Testing demonstrates that the system maintains a stable frame rate of 60 fps at a resolution of 1024x768, with an end-to-end latency of 29.63 ms. Furthermore, the image acquisition, frame differencing, and moving object tracking functions operate normally. The test results, illustrated in the figure below, fully meet the design expectations.

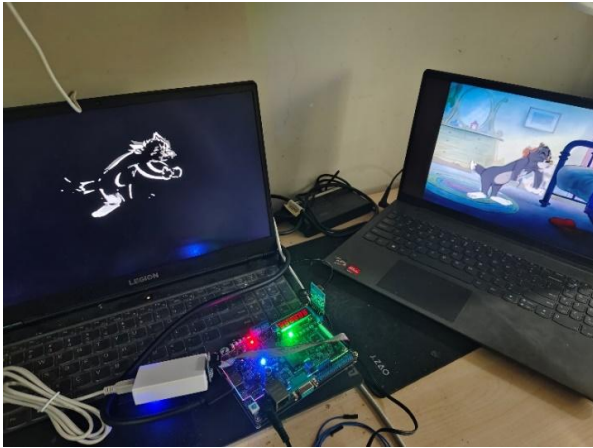


FIGURE 11. Differential Image

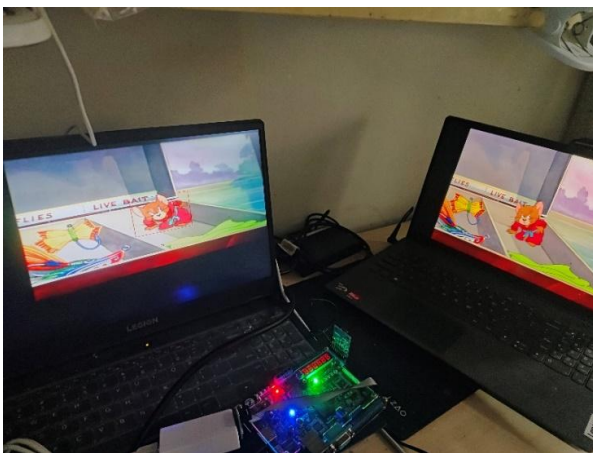


FIGURE 12. Final Image

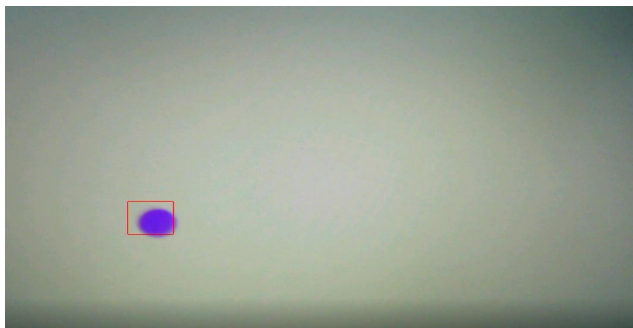


FIGURE 13. Tracking Test

As illustrated in Figures 10-13, the system successfully captures images and performs differential operations on consecutive frames. It accurately identifies moving targets and aligns them for frame tracking, achieving a tracking accuracy of over 98.7%.

Resource	Utilization	Available	Utilization
Register	3001	18224	16%
LUT	3391	9112	37%
511os	1251	2278	56%
10	14	186	39%
RAND168VER	18	32	56%

FIGURE 14. Resource utilization

[illegible]

FIGURE 15. System Power Consumption

The overall resource utilization is relatively low. As shown in Figures 14 and 15, regarding resource utilization, the occupancy rate of various hardware resources such as logic elements is low, with sufficient logic resource redundancy and a concise interface design. In terms of power consumption, the total power is 0.541W, and the dynamic power is 0.120W. The thermal characteristics are good; with an ambient temperature of 25°C, the junction temperature is 37.5°C, resulting in low heat dissipation pressure. Overall, it is a light-load, low-power design, suitable for scenarios with requirements for energy efficiency and resource redundancy.

IV. CONCLUSION

This paper investigates the research background and current status of mobile object recognition systems and proposes an optimized design based on FPGA. The hardware platform utilizes a Transcender S6 development board (Xilinx Spartan-6) paired with an OV5640 camera, developed within the Xilinx ISE environment. To ensure efficiency, the frame difference method is employed for object recognition due to its low computational complexity and suitability for hardware implementation. The system outputs recognized images via an HDMI interface at a resolution of $1024 \times 768 @ 60\text{Hz}$. Experimental results confirm that the system accurately recognizes moving objects in real-time. Furthermore, a comparative analysis with existing FPGA-based solutions highlights the unique advantages of the proposed system. Unlike high-cost [15], resource-intensive architectures often used in similar studies [16], this design achieves a superior balance between performance and cost. It delivers a high frame rate essential for real-time tracking while significantly reducing hardware overhead, proving to be a highly cost-effective solution for embedded mobile object recognition.

ACKNOWLEDGEMENTS

This study was supported by the National Natural Science Foundation project (62105176).

REFERENCES

- [1] W. He et al., "A Low-Cost High-Speed Object Tracking VLSI System Based on Unified Textural and Dynamic Compressive Features," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 68, no. 3, pp. 1013–1017, Mar. 2021. DOI: 10.1109/TCSII.2020.3020883.
- [2] M. Laimer, T. Kern, L. Brukner, G. Schitter, and E. Csencsics, "Enabling Precise 2-D In-Plane Displacement Sensing With Low Latency for Motion Tracking Applications," *IEEE Transactions on Instrumentation and Measurement*, vol. 74, pp. 1–9, 2025, Art. no. 5046009. DOI: 10.1109/TIM.2025.3615282.
- [3] J. Wu, Y. Deng, X. He, et al., "Moving Target Detection Algorithm in Complex Background," *Modern Electronics Technique*, vol. 45, no. 19, pp. 59–65, Oct. 2022. DOI: 10.16652/j.issn.1004-373x.2022.19.012.
- [4] E. Ieno Junior, L. M. Garces, T. C. Pimenta, and A. J. Cabrera, "FPGA-Based EMD Assist Block for Motion Detection in Critical Environments," *IEEE Latin America Transactions*, vol. 15, no. 10, pp. 1856–1863, Oct. 2017. DOI: 10.1109/TLA.2017.8071227.
- [5] S. Li and S. Shang, "Infrared Image Weak Small Moving Target Recognition Method Based on Background Difference Method," *Laser Journal*, vol. 46, no. 3, pp. 142–148, Mar. 2025. DOI: 10.14016/j.cnki.jgzz.2025.03.142.
- [6] C. Wang, C. Fan, and N. Jiang, "Implementation of 'Low Slow Small' Object Detection System Based on FPGA," *Optoelectronic Technology*, vol. 44, no. 1, pp. 29–35, Mar. 2024. DOI: 10.12450/j.gdzjs.202401006.
- [7] W. Ding, D. Yin, and S. Qiu, "Improved Three-Frame Difference Moving Target Recognition Algorithm," *Intelligent Computer and Applications*, vol. 12, no. 3, pp. 180–182, Mar. 2022.
- [8] T. Feng, Z. Mao, and X. Peng, "Design of a Real-Time Motion Target Detection System Based on FPGA," *Application of Electronic Technique*, vol. 49, no. 6, pp. 99–103, 2023. DOI: 10.16157/j.issn.0258-7998.223421.
- [9] R. A. Amin, M. Hasan, V. Wiese, and R. Obermaier, "FPGA-Based Real-Time Object Detection and Classification System Using YOLO for Edge Computing," *IEEE Access*, vol. 12, pp. 73268–73278, 2024. DOI: 10.1109/ACCESS.2024.3404623.
- [10] J. G. Pandey, "An embedded FPGA–SoC framework and its usage in moving object tracking application," *Des. Autom. Embed. Syst.*, vol. 25, pp. 213–236, Sep. 2021. DOI: 10.1007/s10617-021-09252-y.
- [11] T. Li, B. Jiang, D. Wu, X. Yin, and H. Song, "Tracking Multiple Target Cows' Ruminant Mouth Areas Using Optical Flow and Inter-Frame Difference Methods," *IEEE Access*, vol. 7, pp. 185520–185531, 2019. DOI: 10.1109/ACCESS.2019.2961515.
- [12] D. Mukherjee, S. Mukhopadhyay, and G. P. Biswas, "FPGA-based parallel implementation of morphological operators for 2D gray-level images," *Arab. J. Sci. Eng.*, vol. 42, pp. 3191–3206, Aug. 2017. DOI: 10.1007/s13369-017-2429-y.
- [13] Y. Wu, D. Yue, Y. Ding, K. Lu, and G. Zhao, "Crack detection algorithm of expansion circle based on image processing," *Nondestructive Testing*, vol. 18, no. 3, pp. 9–13, 2020. DOI: 10.11973/wsjc202003003.
- [14] Y. Yamamoto, Y. Iriyama, and S. Muto, "Union operation image processing of data cubes separately processed by different objective filters and its application to void analysis in an all-solid-state lithium-ion battery," *Microscopy*, vol. 65, no. 2, pp. 191–198, Apr. 2016. DOI: 10.1093/jmicro/dfv373.
- [15] W. Lee, K. Kim, W. Ahn, J. Kim, and D. Jeon, "A Real-Time Object Detection Processor With xnor-Based Variable-Precision Computing Unit," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 31, no. 6, pp. 749–761, Jun. 2023. DOI: 10.1109/TVLSI.2023.3257198.
- [16] M. Kim and J. So, "Real-Time Driver Monitoring: Implementing FPGA-Accelerated CNNs for Pose Detection," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 33, no. 7, pp. 1848–1857, Jul. 2025. DOI: 10.1109/TVLSI.2025.3554880.