Real-time Background Removal and Substitution based on FPGA

Huang Ray-Ting, Hsu Yi-Chun, Li Yun-Ru
Department of Electrical Engineering,
National Taiwan University, Taipei, Taiwan
b10505029@ntu.edu.tw, b10502070@ntu.edu.tw, b10901085@ntu.edu.tw

I. Introduction

Object detection and background removal play a crucial role in a variety of applications, including surveillance and augmented reality to advanced human-computer interaction. We perform the implementation of a Real-Time Background Removal and Substitution System based on FPGA, applying efficient algorithms and hardware-based optimizations. By integrating preprocessing techniques such as a Mean Filter, Haar 2D-DWT and Thresholding strategy, the system achieves robust and adaptable background removal even under varying lighting conditions. Furthermore, the real-time performance is achieved by low-latency design and dynamic data handling. The design focuses on optimizing memory usage, processing throughput, and visual output quality, making it a practical solution for embedded systems.

II. Algorithm Overview

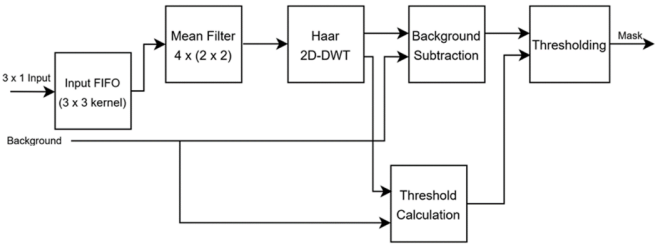

Fig. 1: Overall Architecture

The overall architecture is shown in Fig. 1 . The proposed method can be divided into the following parts:

A. Mean Filter

In the preprocess, a Mean Filter is applied to remove high frequency edge and light intensity variation of different pixel. Each 3 × 3 kernel is divided into four 2 × 2 sub-regions and fed to a Mean Filter and obtain four mean results.

B. Haar 2D-DWT

After Mean Filter, a Haar 2D-DWT (Discrete Wavelet Transform) is then applied to obtain the final result of the kernel. 2D-DWT helps get the real image, it contains one low-pass filter and one high-pass filter, the LL-Band is taken for further process. The DWT result is then compared with the corresponding background pixel and a threshold value to determine whether the pixel should be masked.

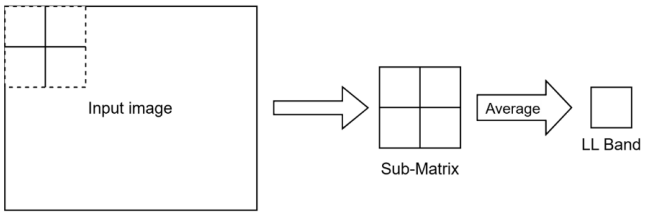

Fig. 2: In the Haar 2D-DWT, a sub-matrix is extracted from the result of the filter, averaged to compute the LL band, and iteratively processed for feature extraction.

C. Threshold calculation

Real-time images are often affected by light and shadow interference. To address this, we modified the subtraction formula by introducing a threshold to help eliminate errors caused by such variations.

$$Sub_i = |Input\ Frame_i - Background\ Image| \quad (1)$$

$$WME_i = \frac{\sum Input_i - Background_i}{2 \cdot Width \cdot Height} \quad (2)$$

$$Threshold_i = WME_{i-1} + \frac{3}{32} \cdot 2^{Bits\ of\ Grayscale} \quad (3)$$

$$Sub_{Mod} = \begin{cases} Sub_i, & \text{if } Sub_i \geq Threshold_i \\ 0, & \text{else} \end{cases} \quad (4)$$

When the subtracted result exceeds the threshold, the pixel is determined not to be part of the background and thus not to be masked. We accumulate the partial sum while calculating the subtracted result of each pixel. The weighted subtracted value of all pixels (WME) is used to compute the threshold. For the i-th frame, since the WME has not yet been determined, the WME from the previous frame is used to determine the threshold.

III. Hardware Implementation

A. System

The overall hardware architecture is shown in Fig. 3. The memory usages are listed in TABLE I, and TABLE II shows the clock frequency of each module.

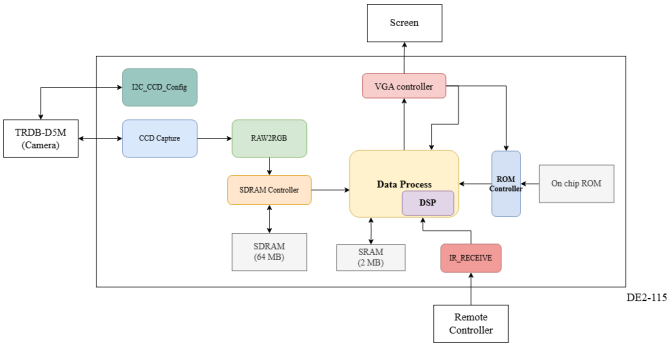

Fig. 3: the block diagram of the system

Memory Type	Data Stored in Memory	Memory Used
SDRAM	Camera Frame	1.83MB
SRAM	Reference Background	468.75KB
BRAM	Image for Replacement, Line Buffer, FIFO	246.58KB

TABLE I: Type and size of RAMs and pictures used

Module Name	Clock Frequency
VGA	40MHz
Data Process	40MHz
DSP	40MHz
SDRAM Controller	100MHz

TABLE II: Clock frequency used in different module

B. FSM

Our system is primarily controlled by counters and a remote controller, as shown in the fig. 4. After a system reset, an initialization phase is conducted via I2C to configure the D5M module. After initialization, the system waits for the user to adjust exposure and positioning settings. When the capture button on the remote controller is pressed, the system captures an image from the camera and stores it in SRAM. Subsequently, it transitions to the processing state, where the frame captured by the camera is stored in SDRAM and read through a FIFO for real-time processing and display on a VGA output. Simultaneously, pre-stored images are retrieved from the on-chip ROM to serve as the new background. If the user wants to adjust the background image, they can press the recapture button on the remote controller to capture a new image.

C. Store background in SRAM

When storing the background, we take into account that each SRAM address stores a 16-bit word. The RGB24 data from the Data Process module is converted to an 8-bit grayscale format using the $Gray_{approx.}$ formula. The grayscale data is then stored row by row into SRAM. This approach ensures that the addressing scheme remains consistent with VGA's data access method when processing the current frame later.

$$Gray_{original} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad (5)$$

$$Gray_{approx.} = (38 \cdot R + 75 \cdot G + 15 \cdot B) >> 7 \quad (6)$$

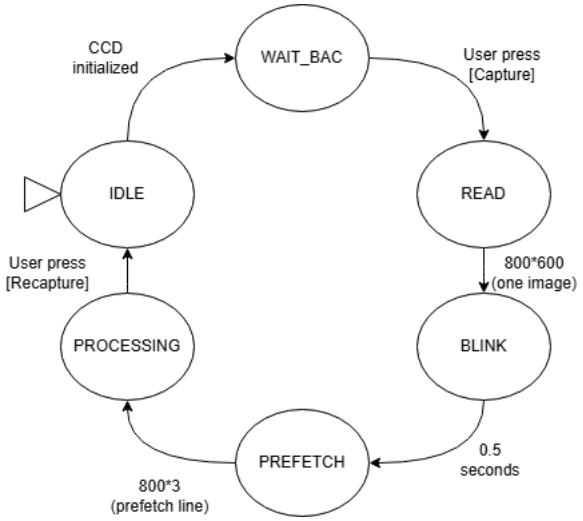

Fig. 4: FSM of the Data process module

D. Fetch current frame from SDRAM

As illustrated in Fig. 5, before entering the processing state, the Data Process module prefetches three rows of data from SDRAM. This prefetching ensures that the DSP does not receive corrupted data that is not sourced from SDRAM. Subsequently, during VGA valid access periods, the Data Process module fetches pixel data for the current frame from SDRAM. On this stage, pixel data is stored in a line buffer with RGB24 format. While reading the data, the line buffer shifts, maintaining three rows of data to meet the 3×3 kernel requirements of the DSP.

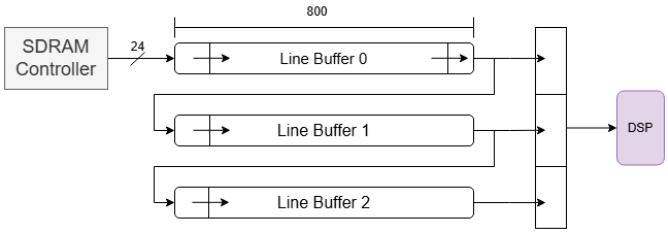

Fig. 5: Line buffers used to store 3 rows of current frame

E. DSP

DSP handles the background removal algorithm. In the DSP module, a 3×3 kernel is applied to an input frame of 800 $\times$ 600 pixels, and a 3×3 buffer is used to process the movement of the kernel across the frame (shown in Fig.6).

During each cycle, the input FIFO of the DSP module receives a 3 $\times$ 1 segment from Data Process, which the DSP module then processes.

F. Real-time combining mask and current frame

After processing by the DSP, mask is returned sequentially, where each mask is represented by a single bit. The Data Process module uses this mask to determine whether the pixel

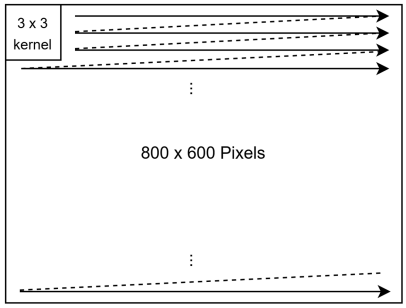

Fig. 6: A kernel is employed to specify the data

retrieved from the line buffer should be displayed. If the mask equals to 1, the original pixel captured by the camera is recognized as an object and displayed; otherwise, the pixel is replaced with either a green screen or the new background pre-stored in ROM.

G. Background substitution

In addition to background removal, our system supports background substitution. The new background is pre-processed and stored in a ROM created using BRAM during the FPGA programming phase. Due to the limited capacity of the DE2-115's BRAM (500 KB) and the VGA display resolution of 800×600, the replaced background image is resized to 400×300 and stored in RGB565 format to optimize memory usage. During run-time, pixel data is retrieved from the ROM based on the VGA display coordinates and is supplied to the Data Process module to render the desired replacement background.

IV. Conclusion

We implement background removal algorithms, with background substitution functionality incorporated. Improve algorithms and apply dynamic data processing techniques to achieve real-time processing while minimizing buffer space usage.

References

[1] Horprasert, T., Harwood, D., & Davis, L. S. (1999, September). A statistical approach for real-time robust background subtraction and shadow detection. In Ieee iccv (Vol. 99, No. 1999, pp. 1-19). Citeseer.

[2] Wang, Y. K., & Chen, H. Y. (2012, July). The design of background subtraction on reconfigurable hardware. In 2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (pp. 182-185). IEEE.

[3] Sarkar, S., Bhairannawar, S. S., KB, R., & Venugopal, K. (2015). FPGA implementation of moving object and face detection using adaptive threshold. International Journal of VLSI design & Communication Systems, 6(5), 15-21.592-598, doi: 10.1109/ASPDAC.2018.8297387.

[4] Sharath, S., and H. G. Rangaraju. "FPGA Based Human Detection Using Background Subtraction." International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT). IEEE, 2018.

[5] Sridevi N., M. Meenakshi."FPGA based Object Detection using Background Subtraction and Variable Threshold Technique",International Journal of Applied Engineering Research Volume 13, Number 16, 2018