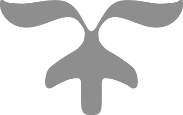


Final Project Report:

A Simple Object Detection and Tracking implemetation

EECE 4377/5377 FPGA Design



April 27, 2018

Lin, Bor-Tyng/Qiu, Hao/Guo, Junlin/ Wang, Xuan

Department of Electrical Engineering

# Abstract

Computer vision algorithms can utilize the parallelism of hardware and gain a significant speedup over the software ones [1]. This work gives a full-hardware implementation of the color-threshold object detection method on DE2-115 board. An interactive pong game based upon the real-time object detection system is deployed. We also explored the hardware implementation of state-of-art feature detection algorithms and finished a Harris-corner feature matching VHDL prototype which is not integrated into the final design.

# Project Overview

As shown in Fig. 1, the design is pipelined and consists of the following main blocks: (1) OV7670 camera interface, (2) Colored-object detection, (3) Ball movement logic, (4) VGA display.



Fig. 1, Architecture of the real-time object detection system and interactive pong game

We configured the design to detect a red-colored and a green-colored object, two markers indicating the centers of colored-objects will be displayed over the real-time video. An interactive pong game could be easily implemented via the ball movement logic. The user is also able to adjust the display mode of real-time video, including the video processed by Sobel Operators, which is the fundamental for more complex feature detection methods.

# Background Information and Motivation

The most crucial part of real-time detection and tracking problem is the speed requirement [2]. To improve the speed, the hardware can be implemented with efficient parallel architecture to support high motional capturing frame from input devices. According to the typical tracking system shown below, the selection of the method of sample extraction and tracking algorithm will highly affect the way to implement hardware under resource limitation.

# Project Objectives

1. Real-time colored-object detection system and interactive Pong game implemented on DE2-115 board.
2. Spatial filter of real-time video using FPGA, Harris Corner detection prototype in VHDL, and Matlab demonstration.

# Strategy and Design Details

1. **Real-time colored-object detection system and interactive Pong game**

We followed the hierarchical design flow in this project. The colored-object detection system and interactive Pong game is composed of: (1) OV7670 camera interface, (2) Colored-object detection, (3) Ball movement logic, (4) VGA display. Each of these blocks contains several sub-entities. This section will introduce how we design these important functional blocks.



Fig. 2, Camera interface block diagram

1. **OV7670 camera interface**

OV7670 is a cheap and easy-to-use VGA camera. The camera is capable to provide a maximum 640x480, 30fps, RGB555/565 video to VGA [3]. We used the open-source VHDL designs [4] to drive the camera and modified the design to fit our own DE2-115 top level entity and VGA\_SYNC module [5]. The OV7670 camera interface consists of the following sub-entities and its block diagram is shown in Fig. 2.

***Open-source OV7670 camera driver***

This design applied I2C communication protocol [4] [6] to control the OV7670 camera, then the capture logic module feeds the RGB444 pixels into the frame buffer. The resolution is configured to 320x240. This RGB444 pixel flow is synchronized with the pixel clock (25.2MHz) of VGA\_SYNC, which indicates we can get the real-time video on VGA display after we process these pixels properly.

***Address Generator and Frame buffer***

When converting the sequential pixel flow into consecutive frames on VGA display, we need a memory structures and the corresponding address generation modules.

As mentioned before, the real-time video is configured in 320x240, RGB444. Then we need an address space of 76800, therefore the width of address should be 17 bits at least. By connecting the “pixel\_colum/row” outputs of VGA\_SYNC to the inputs of address generator, we can easily locate a pixel and assign its address using the following equation:

(1)

When the pixel location goes to the end of active display region, address will be reset to zero. At last the address sequence is fed into the frame buffer and following stages.

Though the frame buffer is included in the open-source design, I separate it from the open-source entities as the frame buffer is an important module in many VGA-relevant FPGA designs. The Frame buffer we implemented contains two dual port RAM (auto generated using Altera Megafuction Wizard). According to the size of address space and color format, each RAM has 76800 12-bit words and synchronized with VGA pixel clock. The write/read operations of these two RAMs are interleaving. When one of the SRAMs is written by OV7670 capture module, another RAM is being read by the next stage, e.g. VGA display.

1. **Colored-object detection**

This design is configured to detect a red object and a green object. A red/green ring is displayed over the real-time video as the marker of the center. The colored-object detection block takes the pixel flow from the output of frame buffer as input. These RGB444 pixels is then passed into color detection and threshold module. The threshold pixel flow will be processed by center calculation module. The coordx/coordy outputs of center calculation modules provide us the coordinates of two detected objects. The schematic of the color detection and ball movement logic stage is shown in Fig.3.

***Color detection and threshold***

Because we plan to detect red and green objects, the threshold for color detection should be defined. The color format of input pixel is RGB444, then there are 16 color levels for each channel. If the upper four bits of the pixel is greater than “0101” (5 in decimal) and the decimal value of the rest two channels is less than 2, then the pixel is red. Similarly, if four bits in the green channel is greater than ”0101” and the red/blue channel is less than 2, the pixel is green.

Once we determine a pixel to be red or green, threshold operation will be carried on input video. The threshold image is applied to simplify the center calculation. Here we define that: if the pixel is red, its threshold RGB value is “111111111111” (strong white), if the pixel is green, its threshold RGB is “000000111111”. If the pixel is neither green nor red, its threshold will simply be all zeros (black).

***Center Calculation***

The center calculation module takes the threshold image, pixel\_row/column as the data inputs, and clock, Vsync as controlling inputs. Two similar center calculation modules are deployed here, one for red, another one for green. We will take the Red one for example here:

The center calculation module gets the average location of the red pixels in both X and Y directions. First, we initialize the temp variable sumx, sumy, countx, county to be 0. If the threshold pixel RGB value is “111111111111”, we will accumulate the “sumx” with current “pixel\_column”, and “countx” with ‘1’. The average (sumx over countx) will be calculated at

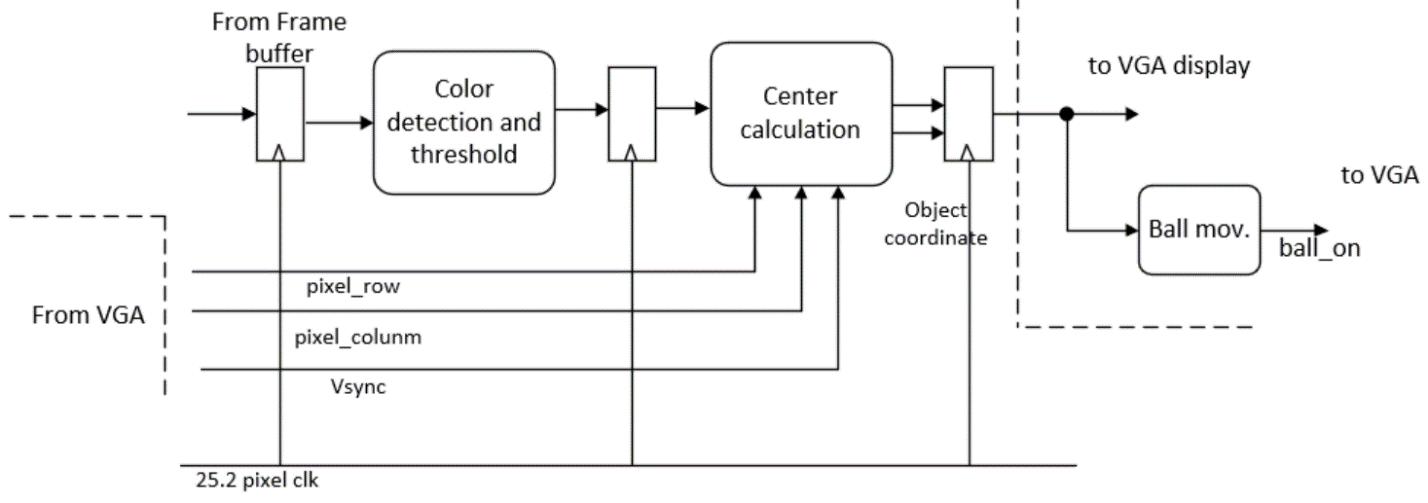


Fig. 3, Color detection and ball movement logic block diagram

the last pixel within the active display region. *i.e.* coordinate (320,240). The similar operation is also carried on Y direction. Finally, the temp variables will be reset to 0 at the beginning of the next frame. i.e. when *Vsync* equals to 0. The average coordinates we get here are defined as the centers of detected objects and fed into next stages, the Ball movement logic and VGA display.

1. **Ball movement logic**

Pong is one of the first computer games that ever created, this simple "tennis like" game features two paddles and a ball [7], we neglected the score board feature in this work. We can simply add a “ball” into VGA display.The red/green object detection modules can tell us the location of object on VGA display. A virtual paddle can be defined around the center of detected objects. A Nintendo Wii-like Pong game can be implemented by defining the movement of the ball, i.e. might be the simplest physics engine for video games.

The ball movement logic is developed based on Lab3, problem 2 [5]. We changed the boundary of screen into 320 and 240 to fit the 320x240 real time video. Apart from the boundary collision, the collision with the object is added to “Ball.vhd”. In this work, the paddles are defined to be 20 pixels wide (X direction) and 60 pixels high (Y direction). The moving direction of the ball will turn opposite if the ball hits the boundary or virtual paddles. Note that the movement logic on X direction for green paddle is opposite to the red paddle, the reason for which is the nature of tennis game.

1. **VGA display**

The VGA\_SYNC module applied here is based on Lab3, problem 5 [5]. This VGA module accepts RGB888 color inputs and display 640x480, 24-bit colored image on monitor. Therefore,

we define the active display region to be 320x240, and concatenate “1111” to each channel of RGB444 pixels. Besides, the pixel flow from the frame buffer is processed by many previous stages, we must design a controller to determine what should be displayed on monitor. An entity named “frame drawer” is deployed here. The block diagram of VGA display is shown in Fig .4.

***Frame Drawer***

The frame drawer can be considered as a primitive GPU, which takes the clock, display modes as controlling inputs. A series of data buses such as image data, object coordinates, VGA coordinates served as the data inputs. In this project, the frame drawer has two operation modes and for display modes. We use SW (5) to turn on the interactive Pong game, otherwise the system operates on object detection mode. SW [17:16] controls the image display mode. The “Opcode” and corresponding display mode is shown in Table.1.

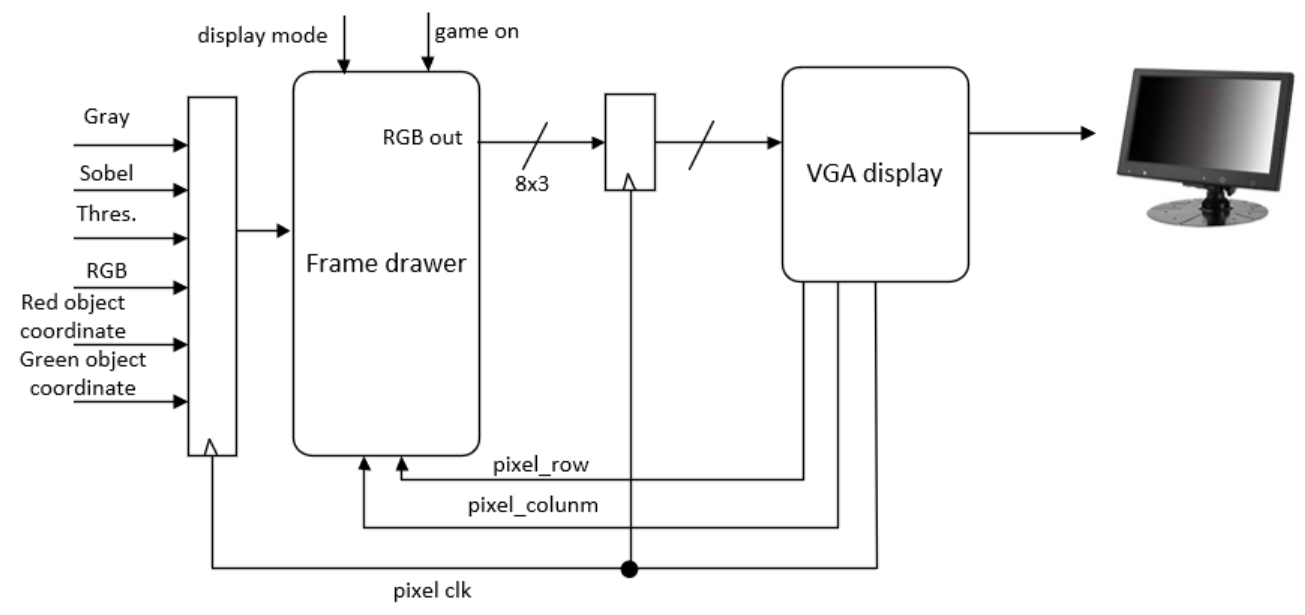


Fig. 4, VGA control and display sync block diagram

Table. 1. Display mode Opcodes definition

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Opcode | 00 | 01 | 10 | 11 |
| Display mode | Regular RGB | Edge detection | Grayscale | Color threshold |

1. **Spatial filter and Harris-corner detection prototype**

In Part (a), a hardware implementation for demo is provided with a simple color intensity detection mode along with a mini-PONG game. However, to detect complex images in real life, lots of various detection and detection methods are developed. To reduce the complexity and to avoid floating point calculation, Harris-Corner (HC) detector is chosen to be our prototype implementation. The reason of choosing HC algorithm is listed below:

(i) Only basic fixed-point additions, subtractions, multiplications and shifting are used in the arithmetic operations. The most difficult operation is dealing with the (x, y) derivatives. This can be achieved by spatial filtering techniques.

(ii) Requires only current frame information to estimate the location of detected pixels in present image. Lots of other detection or tracker algorithms require next frame pixel information to compute the relation factors forming current state equation.

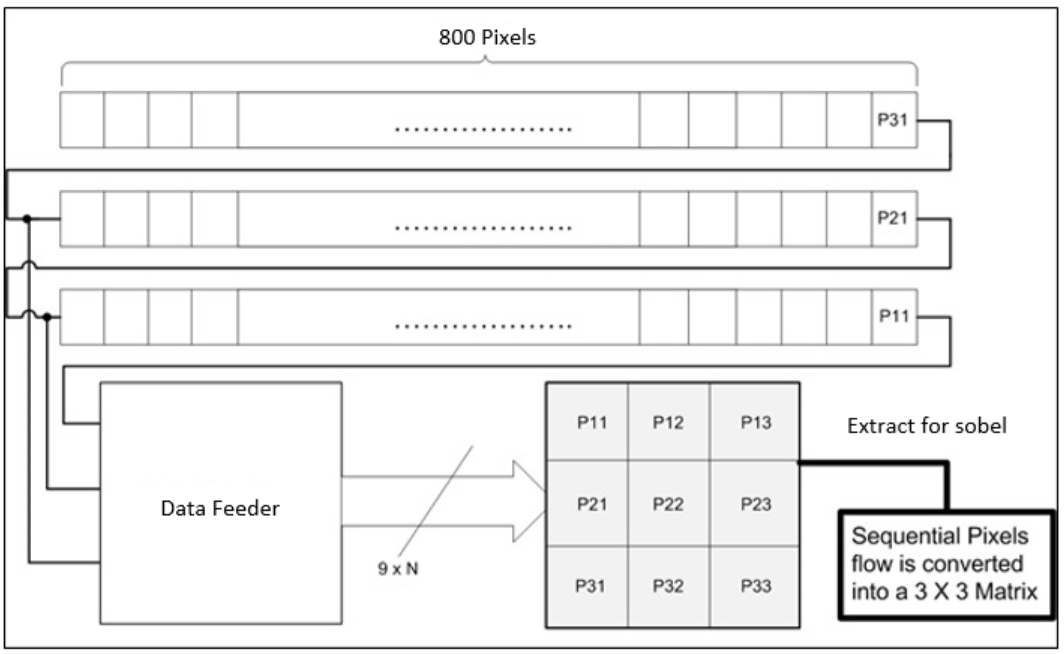


Figure 5*.* Schematic of the line buffer and data feeder

1. **Spatial Filter using FPGA**

***RGB to grayscale***

The state-of-art feature matching tend to be computational intensive and may require large memory resources to store intermediate variables. Hence, we compressed 12-bit RGB color pixel value into 8-bit Grayscale intensity using equation (2) [8]. We applied right shift (divide by ) to intimate the floating-point numbers.

(2)

***Line buffer and data feeder***

Spatial filtering on images is typically accomplished through the convolution between 2-D image matrix and a certain kernel. However, the pixel flow from video input is sequential, we must convert this sequential pixel flow into a parallel one. In this work, we implemented Sobel edge detector on input video. Here we take it as an example, to filter the image with Sobel operators, a

line [8] [9] buffer built with Altera RAM-based Shift Register IP Core [10] is utilized to convert the flow of sequential pixels into 1x3 vector, i.e. consecutive 3 pixels in the same column. Afterwards a data feeder consists of shift registers is applied to further transfer 1x3 vectors into 3x3 matrices. Schematic of the line buffer and data feeder is shown in Fig.5.

As the total number of pixels of a line in VGA\_SYNC is 800, the distance between 3 taps of the shift register is set to be 800. Line buffer and data feeder are synchronized with 25.2MHz pixel clock.

*Spatial filter example: Sobel Operator*

The spatial filtering operation can be easily accomplished after we get 3x3 matrices from the data feeder. The center pixel value is replaced with the new center pixel value after the 2-D convolution. 2-D convolution can be implemented using a series if matrix multiplications.

For example, Sobel Operator is a widely used edge detector and a good approximation of the derivatives for the image. We approximated Sobel Operator using equation (3) and the center pixel in the sliding 3x3 window is replaced with

\*, \* … (3)

If the kernel size is reasonable and the kernel can be approximated into fixed point expression, we can implement most of the spatial filters using the similar method.

1. **Harris Corner detector**

Unlike sobel filter which are generally used only in image edge detection, Harris corner detection provides good feature detection on image corners by small scanning window like those in Fig.6. The corner points contain contour junctions intuitively and generate stable features when target features are captured in different viewpoints. Because of these advantages, the selected patches from template frame source will be good candidates to achieve feature matching in target frame.

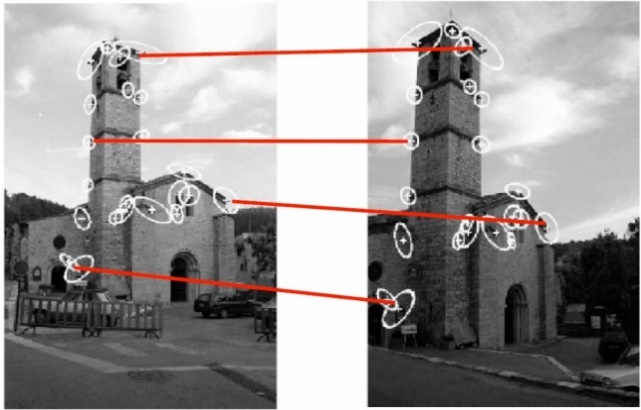
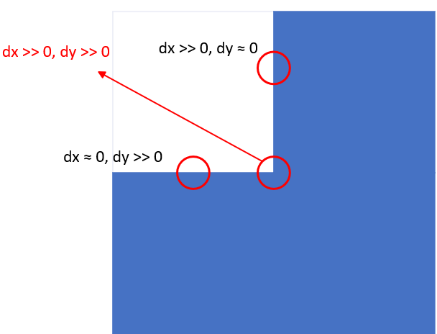


Figure 6. Matching point by corner detection of same object in different frames (or object in motion). [12]

Figure 7. Derivative pairs around the edge and corner of color region

Fig.7 shows that the basic concept to detect a corner is to capture the pixel point (xcenter, ycenter) which has large color intensity difference comparing with neighborhood pixels in a small defined region. The center point (xcenter, ycenter) of the window is then chosen as corner candidates. The mathematical equation to determine whether the examined center pixel is a corner is shown below:

Where is the shifting of x&y direction; =1 defines active window;

stand for the gradients of (x, y), i.e. (dx, dy). To check whether the color intensity difference () caused by shifting is large enough to extract a corner pixel, a response metrics is provided in [ref]:

: an empirically developed constant;

Although the original method uses eigenvalues to obtain value, we will not use exact eigenvalue calculation to acquire due to high complexity in hardware implementation. Moreover, to avoid the floating-point operation of multiplying by , we followed [13] to perform an approximation of the multiplication by (multiple by 5, shift right to LSB by 7 bits for division). After is computed, current examined pixel point (xcenter, ycenter) can be classified

either as an edge, a corner or a flat region according to the threshold value of R. The classification metric is listed below:

: threshold value of classification, positive integer

The HC algorithm [11] mentioned above can be easily performed by MATLAB simulation. For hardware implementation, the HC detector block diagram is shown in Fig. 8. The detector is divided into R-calculation and corner candidate selection. In R-calculation block, a line-buffer parallelly passes a 77 window of pixels’ gray scale value array into the gradient operator. There are 25 points gradient pair () calculated in the gradient operator by sobel filter. Total 50 subtractions are executed during sobel kernel convolution with adjacent 33 image pixels. The , , product terms and the sum of each product over 25 points are computed next in the parallel multiplier and adder unit.

The final R response value is completed by the final multiplication and shift operation in the last small block. For each active pixel examined, a corner tag will be set to 1 if exceed for certain pixel. After all the s in a frame that have been computed, a 1-bit 640480 array is stored with 0 and 1 to record the corner map point of the active frame. If exceeds , the corner tag of that pixel will be set to 1 or it will be 0. According to the coordinates provided by map information, we can draw a small white circle that indicate target corners. Finally, all the current frame is substituted with detected while circle on every sensitive corner that HC detector have captured. This completes the object detection phase by corner candidate selection block.

# Results

As we were making progress throughout the project, we found implementing complex real time feature matching is not practical in the given time window. We picked the simplest feature: color to design the object detection system. The design colored-object detection was carried smoothly, then we decided to add an interactive Pong game. Though the collision logic is not perfect, we successfully developed a basic prototype of “Nintendo Wii-like” video game. We successfully build a real-time colored-object detection system and interactive Pong game; a prototype of Harris-corner feature detector model is also given. The accuracy and the speed of our colored-

object detector is acceptable if the surrounding lighting is good. Though the collision logic of interactive Pong game can be further refined, the game can be considered as a prototype.

Comparing to the objectives we raised in proposal, we utilized the color as the feature of the object to be detected and get our design to work perfectly. The full-hardware object detector is capable to work with non-stable background. OV7670 camera is also integrated into our project smoothly. Overall, we have accomplished image detections for both fix frame and real-time inputs from camera OV7670 which achieves our original goals to Stage 2.

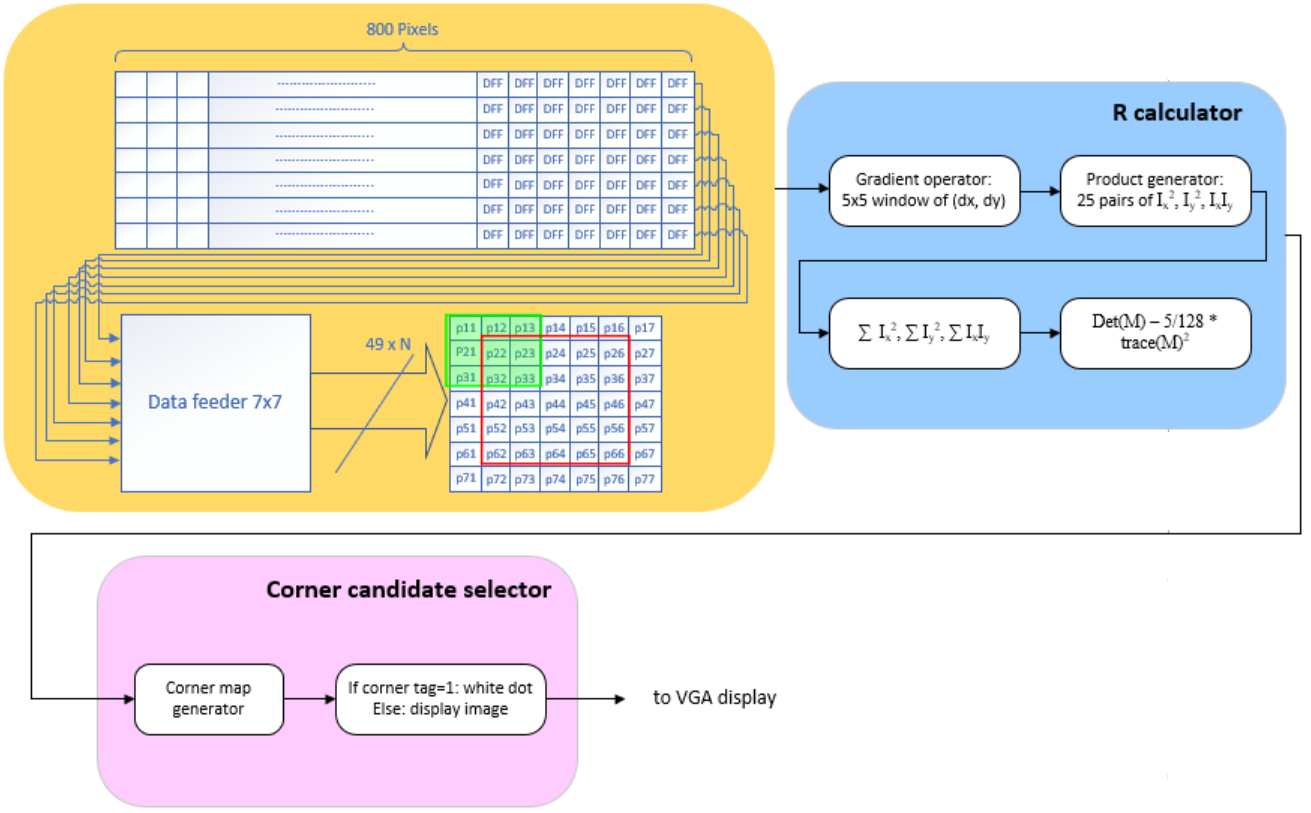


Figure 8. HC detector block diagram

# Discussion & Conclusion

Over various object detection methods, the color intensity detection is simple but efficient to image that contain stable big flat region of mono-color scale features. However, single color detection won’t be useful in feature matching. Unlike color detection, sobel edge detection mainly focuses on edge contour tracing. If the scanned feature in certain frame have many edge contours generated by background noises, target object cannot be detected perfectly. This is probably the downside of sobel. According to HC detection result by MatLab simulation, both detection and feature matching can be improved by tuning the *RT* carefully. However, the hardware complexity to implement one is crucial which is demonstrated in block diagram mentioned in previous sections. Not to mention the effort required to integrate with ov7670 stream input fetching block.

In conclusion, one should choose the most appropriate method that provide acceptable detecting quality and hardware resources based on one’s design specification for hardware implementation. Normally the complexity increases as the algorithm tends to collect more local pixel information. When this is extended to tracking problem, the complexity increases as the number of next coming frame required to collect for current tracking prediction.

# Reference

1. D. B. K. Trieu and T. Maruyama, "An Implementation of the Mean Shift Filter on FPGA," *FPL*, Chania, 2011, pp. 219-224.
2. Al-Hussein A. El-Shafie and S. E. D. Habib, “A Survey on Hardware Implementations  
   Visual Object Trackers,” *Published in* *ArXiv*, 2017.
3. OV7670/7171 CMOS VGA camera datasheet: *https://www.voti.nl/docs/OV7670.pdf*
4. Dr. Cristinel Ababei’s Digital Camera Project: [*http://www.dejazzer.com/eigenpi/digital\_camera/digital\_camera.html*](http://www.dejazzer.com/eigenpi/digital_camera/digital_camera.html)
5. Vanderbilt University EECE-4377 course resources.
6. Jonathan Valdez and J. Becker, Understanding the Bus, *Application Report, Texas Instruments,* 2015
7. PongGame.org: <http://www.ponggame.org/>
8. Cornell University ECE5760 course website: <http://people.ece.cornell.edu/land/courses/ece5760/index.html>
9. Digital Video & Image Processing tutorial slides, Xilinx: <http://engrwww.usask.ca/classes/EE/862/video_processing.pdf>
10. RAM-Based Shift Register (ALTSHIFT\_TAPS) IP Core User Guide, Altera
11. C. Harris, M. Stephens, “A combined corner and edge detector,” *Proc. Alvey Vision Conference*, 1988, pp. 147-151.
12. R. Collins, “Lecture 06: Harris Corner Detector,” *CSE486 Penn State*.
13. T. L. Chao, K. H. Wong, “An efficient FPGA implementation of the Harris Corner feature detector”, *IAPR International Conference on Machine Vision Applications*, pp. 89-93, May 2015.