Real Time Detection of Object Blob Localization Application using 1-D Connected Pixel and Windowing Method on FPGA

CheeYuen Lam¹, Phaklen Ehkan¹, Rafikha Aliana A.Raof¹, Suwimol Jungjit²

¹Embedded, Networks and Advanced Computing (ENAC) Research Cluster, School of Computer and Communication Engineering, Universiti Malaysia Perlis, Arau, Perlis, Malaysia.
²Department of Computer and Information Technology, Faculty of Science, Thaksin University, Phatthalung, Thailand.

Corresponding author’s : phaklen@unimap.edu.my

Abstract. Blob detection and localization is a common process used in the machine vision. Current existing blob detection method is using 2-dimensional kernel matrix which is higher in time consumption and also memory space. This study has proposed a dedicated digital architecture consist of two modules to detect binary image blob using only 1-dimensional image pixel. First module is used to detect connected pixel in a row of pixel, and second module is used to perform windowing to justify blob location. This study has been successfully implemented and tested on Altera DE2 FPGA board. The proposed architecture only takes 24 clock cycles to deliver blob location and related features. The tested architecture only utilizes 1597 logic element, or 4.81% of the FPGA total resources.

1. Introduction
Machine vision technology has been widely applied in our daily life nowadays [1]. This technology becomes well received mainly caused by its dynamics inputs which is the digital image captured by the digital camera. Many applications of machine vision system used to detect and locate object within its vision zone for various purpose such as security surveillance, target tracking, product inspection and others. Object detection in machine vision system making use the data produced by object detection process to perform automate mechanical operation without supervision of human operators to increase the productivity of the manufacture process.

Image feature extraction is vital process to achieve object detection operation in machine vision. Feature extraction process are extracting feature of the image such as corners, line, edge, colors, and blob which depend on the application of the vision system. Current blob detector is utilizing 2-D kernel to perform detection. However, this method requires large memory space and calculation steps along the detection process. Besides that, matrix convolution for 2-D operation is time consuming.

This study is aimed to develop a dedicated blob localizer in binary digital image using 1-dimensional pixel data to detect and localize blob in the image in real time.

2. Previous Works
In the context field of binary image blob detection, there are two common methods namely, the connected component labeling (CCL), and Laplacian of Gaussian (LoG) method.
2.1 Connected Component Labeling

Connected Component Labeling (CCL) technique is used to identify blob and differentiate blob component in binary image. There are various of applications use this technique for binary blob detection such as medical image segmentation [2], traffic monitoring [3], vehicle license plate detection [4], and range mapping [5]. While serving variety of applications, CCL is used to perform blob segmentation on the image [6].

Several studies have modified the existing CCL algorithm to improve the processing performance. Those improvements made are to avoid the process of pixel scan through the image in order to achieve better time performance of the algorithm. The algorithm has been modified into line based [7-8] and block based [9] during labeling process. Another alternative method to enhance the CCL performance is implementing this algorithm into digital hardware architecture. Dedicated digital architecture implemented in FPGA is capable to speed up the algorithm processing time [10-12].

2.2 Laplacian of Gaussian (LoG) filter

Besides from CCL algorithm, Laplacian of Gaussian (LoG) is another common blob detection operator used by the researchers. LoG is 2-dimensional operators defined by the Equation (1).

\[
G(x, y; \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]

Equation (1) will produce a 2-dimensional kernel matrix with a peak at the center of the kernel. The peak orientation and width can be modified by manipulating the parameters in the equation. Applications of LoG have been explained in the study of Hui [13]. This technique has been used to detect road vanish point [14], road extraction [15] and vehicle detection [16] in digital image.

As summarize for previous works, both discussed techniques are required 2-D matrix kernel. 2-D operation consumes more processing power of the computer and compromising time performance. Therefore, this study proposed a blob detection and localization technique using only 1-dimension pixel for fast processing time. FPGA-based method is proposed for object detection because it provides better performance [17-19].

3. Method

This study covers software development and hardware implementation. A DE2 TV sample project provided by Altera Corp is used as image acquisition module because it is capable to provide VGA resolution video stream with 60 frames per seconds in real time. The video stream quality is sufficient for machine vision system. The architecture and functional block of the DE2 TV is described in next section.

3.1 Image Preprocessing

DE2_TV sample project is provided in the companion disk of DE2 FPGA board. This sample project is design for the demonstration of converting analog PAL video signal and display the video in VGA LCD screen. This study is using the digital VGA output signal in the VGA controller module.

![Figure 1. FPGA DE2_TV architecture.](image-url)
Based on the architecture in Figure 1, the VGA controller provides control signal include x and y pixel coordinate values of red, green and blue colours. These signals are useful for the design of blob detector. Before blob localization process is taking place, filtered binary images have to be produced. These binary images are required to contain only the object blob in the image with black, ‘0’ pixel value. The method of producing filtered binary image can be varies for application. Figure 2 demonstrates sample of filtered binary image contains object blob.

**Figure 2.** Example of binary image with blob.

### 3.2 Proposed Architecture

The overall development of detector architecture comprises of three (3) major functional modules. The first module is 1-dimension connected pixel detector, second is the clustering element and third, is the control unit. These can be seen in Figure 3.

The 1-D connected pixel detector is responsible to search the connected pixel in the same row and pass the pixel information to the cluster element module. Cluster element module is responsible to perform windowing and localization operation on the connected pixel. After all the pixel of a Region of Interest (ROI) is scanned, the blob location that kept in the cluster element is shifted out in series mode for further process. The control unit is responsible for control the whole detection process and also the shifting process when it is done.

**Figure 3.** Blob localization basic structure.

### 3.3 Proposed 1-dimension Connected Pixel Detector

For the first step of blob localization, connected pixels in same row are required to detect and three values are extracted. The information is x location of the start connected pixel, x location of end connected pixel, and the center point the connected pixel train. This detector is a pixel 1-dimensional kernel (as shown in Figure 4).
This detector is required to scan through row by row in the ROI. The operation of this detector is described in pseudo code as follow:

1. Define length_limit;
2. Set C1 to first pixel of first row;
3. If C0 = ‘1’ AND C1 = ‘0’ AND C2 = ‘0’ then
4.   Start_x_buff = C1 x location
5.   End if;
6. Shift next pixel
7. If Not end of row
8.   If C0 = ‘0’ AND C1 = ‘0’ AND C2 = ‘1’ then
9.     length = C1 x location – start_x_buff;
10.    If length > length_limit then
11.       End_x_buff = C1 x location;
12.      Mid point = End_x_buff – Start_x_buff;
13.     Return start, middle and end value;
14. Shift next pixel;
15. Goto Step 3.
16. End if;
17. End if;
18. Else
19.   If C0 = ‘0’ AND C1 = ‘0’ then
20.     End_x_buff = C1 x location;
21.     Mid point = End_x_buff – Start_x_buff;
22.     Return start, middle and end value;
23. End if;
24. Scan next row; Go to step 1.

As stated in the pseudo code, the 1-dimensional connected pixel detector will detect all black ‘0’ pixels with minimum two consecutive connected pixels. Single pixel is discarded by this detector. To further eliminate interference of noise at the image, length_limit parameter is introduced in the algorithm. This limit value is set as the minimum threshold length of the connecting pixels train. If the pixels train length is fulfilled or longer than limit, it will consider as valid pixel train. Otherwise, the pixel train will be discarded by the detector. This measure is to prevent the salt and pepper noise exists in the image to consider as object blob.

3.4 Proposed Cluster Element Module
After the connected pixels are detected, the start location, end location and mid-point location of the connected pixel string pass the cluster element module. This module is then performing windowing process to calculate the boundary of the object blob. One cluster element is only hold the value for one blob. Multiple cluster element modules are required to connect in cascade manner if there are several blobs to be detect in the ROI of the image (Figure 5).
Figure 5. Parameters in cluster element.

This module is responsible to hold several important parameters for calculate the blob location. These parameters such as previous connected pixel start, middle and end point; previous row number; window size and total number of connected pixel train belong to this blob. At the initial state of this module, all parameters are set to zero, an Empty flag is set. Incoming detected connected pixels occupied an empty clustering module. The middle point and row number are used as reference location in windowing process.

Figure 6. Cascaded cluster element.

A rectangular boundary is set by the middle point and row number of the previous connected pixel train as center of the rectangular. The following incoming pixel train middle point is then evaluated and justified whether it is located within the rectangular boundary. If yes, then all parameters value and row number of the cluster element is updated, and the vote number is incremented (Figure 6). If the incoming pixel train did not fall within the rectangular boundary, the pixel train is then passed to the next cluster element for evaluation. The operation algorithm for cluster element is defined in the pseudo code as follow:

1. Define vertical and horizontal window size.
2. All parameters set zero. Empty flag set ‘1’.
3. Incoming pixel train detection.
4. If Empty flag set ‘1’ then
5. Update parameters registers;
6. Empty flag set ‘0’;
7. Else
8. Compare incoming middle point.
9. If Within range then
10. Update parameters.
11. Else
12. Pass to next cluster element.
13. End if;
14. End if;
Figure 7 displays an example of point windowing operation. The light green, blue and dark green pixels indicated the beginning, middle and end pixels of the detected connected pixels train in a row. Then, windowing operation is using the first middle point as center and project a rectangular boundary. If the next middle point is within boundary, it is belonging to the same blob. Otherwise, a new cluster is created, and belongs to another blob object.

4. Proposed Image Processing Technique
There are two set of image processing algorithms which are based on HSL and RGB spaces proposed in this study. Both algorithms are aimed to filter and produce a binary image that contains only the sausage linking twist blob.

After the algorithm of the connected pixel detector and clustering element is developed, the algorithm is then implemented in digital hardware architecture. Altera Quartus II software is used as development platform, and Altera DE2 Cyclone II FPGA board as hardware implementation platform.

4.1 1-D Connected Pixels Detector Architecture
Based on the algorithm of the connected pixels detector discussed in previous section, it is implemented using a three stages shift register and a state machine. The shift registers are the three pixels kernel. Each of the value stored inside the register are fed to the state machine to perform conditional checking. The finite state machine (FSM) is responsible for controlling the detection sequence and control signal for the output (Figure 8).
The FSM has been implemented with three states which include SEEK_START, CONNECTED_LEN and SEEK_END. SEEK_START state is where the state machine searching for beginning of connected pixels. When beginning of connected pixels is found, the x location of register C1 is store in the START_X_REG register. The state is transition to CONNECTED_LEN for counting the number of connected pixel. The length counting state enables user to decide the minimum length of connected pixels to be considered valid. When the length is valid, its state is transition to SEEK_END to find the ending of the pixel train. When the ending is found, the x location is store in the END_X_REG register. If the length did not fulfill user limit, the state is transition back to SEEK_START again.

Figure 8. RTL of 1-D connected pixel detector.

After the starting and ending location are found, the middle point of the connected pixel train is determined by using Equation (2).

\[
\text{Mid}_x = \text{Start}_x + \left( \frac{\text{End}_x - \text{Start}_x}{2} \right)
\]  

(2)

This calculation operation is implemented using digital arithmetic logic. Therefore, the value of middle is obtained within the period of one clock cycle. When all output data is ready, the state machine triggered a DATA_RDY signal for the clustering element to read the valid value for windowing process.

4.2 Clustering Element Architecture

The architecture of clustering Element is more complex than the 1-D connected pixel detector (Figure 9). It is constructed by two major blocks, first is the input data and shifting block and second is the windowing block. The input data from the preceding component is fed to the registers in shifting block. These data are then used for windowing operation.

The start and end value of x of the connected pixels are averaged. The maximum and minimum of y are determined to obtain the blob height. The window boundary is calculated by rectangular window bound module. The incoming middle point is compared and determined falls within the window. If the point is within boundary, all registers in the windowing component are updated; the vote register is incremented by 1. When the incoming middle point is outside the boundary, no update is done on the registers. The data stored in the shifting registers are shifted to next cluster element for windowing process.
When the ROI is finish scanned, the data in the registers of windowing block is passed to the shifting registers using multiplexers. Then the blob information in each cluster element is shifted out sequentially for further justification. The cluster element with highest vote value represents most possible blob cluster and location. Due to limited shifting registers, the data in the windowing block is separated into two sets.

4.3 Test Architecture
A test architecture which contains of both modules and a RS232 communication module are developed for physical testing. The RS232 module is used to transmit the data in every cluster element to laptop computer for verification. In this testing architecture, there are total of 8 cluster element modules are connected in cascade.

5. Results
The test architecture is implemented in DE2_TV project to detect blob. The detecting ROI region is set at first 30 rows of pixels from the top of VGA screen which has total resolution of 30 x 640 pixels. Once the pushbutton on the DE2 board is pushed, the ROI is started to scan by the detector and the result is sent to the laptop computer by using RS232 serial communication (Figure 10).

Figure 9. RTL of clustering element.

Figure 10. Blob binary image.
The green color horizontal line displayed on the LCD screen is the indicator of the boundary limit of ROI region. Once the ROI region is scanned, the blob information is sent to computer. The data is received and displayed using communication software called Realterm. Figure 11 shows the screenshot of communication terminal of received data.

![Cluster element data verification.](image)

Table 1. Serial terminal data arrangement table.

| Set | Column          |
|-----|----------------|
|     | 1       | 2       | 3       | 4       |
| 1   | Current x | Current y | Vote   | Average start x |
| 2   | Average middle x | Average end x | Max y | Min y |

The digital hardware resources consumed by the 1-D connected pixel detector and cluster element are listed in the Table 2. The overall integrated module with 8 cascaded cluster elements consuming only 4.81% of the Cyclone II FPGA logic elements.

Table 2. Hardware resources utilization.

| Module                                | Used LE | Utilization (%) |
|---------------------------------------|---------|-----------------|
| 1-D connected pixel detector          | 88      | 0.26            |
| Cluster element                        | 189     | 0.57            |
| Cluster element (8 cascaded)           | 1521    | 4.58            |
| 1-D connected pixel detector + Cluster element (8 cascaded) | 1597 | 4.81 |
6. Discussion

6.1 Speed Performance
Based on this architecture, the cluster result can be obtained in 24 clock cycles after scan through ROI. The first 8 cycles are for the last detected point to be windowed in cluster element. Then, another 8 cycles are for the first set of cluster element to be transmitted out, and the last 8 cycles are for the second set data to transmit out. This transmission can be completed within the blank period of VGA signal between rows of pixels. Therefore, the detection process can perform up to the frame rate of the video stream.

6.2 Multiple ROI Blob Detection
Another advantage of this blob localization architecture is performing multiple ROI blob localization within a single image frame time. If the blob localization required to be detecting on full image scale, then the image can be breakdown into several full horizontal length ROI, then this ROI is scanned by the detector. Therefore, multiple ROI for blob localization can be achieved.

6.3 Detection Performance
The blob detector accuracy can be enhanced by adjusting the window size and shape. The window size shape can be adjusted according to the object blob. The smaller window size is suitable for detecting smaller blob, while larger window size is more suitable for detecting larger blob.

6.4 Noise Blob Interference
However, this method is prone to noise blob interference. The noise is caused by the shadow, and object edge. The input image has to be very clean from noise blob interference before sending for blob detection. Noise blob will occupy the cluster element space and might causing the true blob failed to detect. However, the \textit{length\_limit} parameter in the connected pixels detector helps to reduce the noise pixel being detected as blob. This parameter can be adjusted for comply to the object blob length so that noise pixel train is filtered out. This can be seen from Figure 11 where a noise blob is detected. Another measurement to prevent this problem is cascade more cluster element module to increase the blob accommodation capability and select the true blob base on vote value.

7. Conclusion
The proposed 1-dimensional blob detector has been successfully detected and located blob in binary image. Noise pixels and noise blob are filtered by setting the minimum connected pixel length limit. The estimation of start boundary, end boundary and center point of the blob are determined by this detector. These values are useful for post-processing and physical object handling. This architecture has the flexibilities in terms of scale and detection speed. The real-time detection of object blob localization by using 1-dimensional connected pixel and windowing method are efficiently implemented on the FPGA-based hardware with only 1597 (4.81\%) of total logic elements available.

References
[1] Bernard Marr, “7 Amazing Examples of Computer and Machine Vision In Practice”, 2019, https://www.forbes.com/sites/bernardmarr/2019/04/08/7-amazing-examples-of-computer-and-machine-vision-in-practice/#7aec23d21018 (published on 18\textsuperscript{th} April 2019).
[2] A. Bhan, A. Goyal, M. K. Dutta, K. Riha and Y. Omran, “Image-based Pixel Clustering and Connected Component Labeling in Left Ventricle Segmentation of Cardiac MR Images,” 7\textsuperscript{th}. International Congress on Ultra-Modern Telecommunications and Control Systems and Workshops (ICUMT), Brno, 2015, pp. 339-342.
[3] Bevilacqua, A., A. Lanza, et al., “A Single-Scan Algorithm for Connected Components Labelling in a Traffic Monitoring Application Image Analysis”, 13th. Scandinavian Conference, SCIA 2003 Proceedings, Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 677-684.

[4] R. M. Khoshki and S. Ganesan, “Improved Automatic License Plate Recognition (ALPR) system based on single pass Connected Component Labeling (CCL) and Reign Property Function,” IEEE International Conference on Electro/Information Technology (EIT), 2015, pp. 426-431.

[5] A. Ravankar, Y. Kobayashi, A. Ravankar and T. Emaru, "A Connected Component Labeling Algorithm for Sparse Lidar Data Segmentation," 6th. International Conference on Automation, Robotics and Applications (ICARA), Queenstown, 2015, pp. 437-442.

[6] Jau, U. L. and C. S. The, “Real-Time Object-Based Video Segmentation Using Colour Segmentation and Connected Component Labelling”, First International Visual Informatics Conference, IVIC 2009 2009 Proceedings. Berlin, Heidelberg, Springer Berlin Heidelberg, pp. 110-121.

[7] H. L. Zhao, Y. B. Fan, T. X. Zhang and H. S. Sang, "Stripe-based Connected Components Labelling", Electronics Letters, 2010, vol. 46, no. 21., pp. 1434-1436.

[8] K. Yonehara and K. Aizawa, "A Line-Based Connected Component Labeling Algorithm Using GPUs," Third International Symposium on Computing and Networking (CANDAR), Sapporo, 2015, pp. 341-345.

[9] W. Y. Chang and C. C. Chiu, "An Efficient Scan Algorithm for Block-based Connected Component Labeling," 22nd. Mediterranean Conference on Control and Automation, Palermo, 2014, pp. 1008-1013.

[10] D. Kiran, A. I. Rasheed and H. Ramesh, "FPGA Implementation of Blob Detection Algorithm for Object Detection in Visual Navigation," International Conference on Controls, Communications and Informatics (CCUBE), Bengaluru, 2013, pp. 1-5.

[11] J. G. Pandey, A. Karmakar, A. K. Mishra, C. Shekhar and S. Gurunarayanan, "Implementation of an Improved Connected Component Labeling Algorithm using FPGA-based Platform," International Conference on Signal Processing and Communications (SPCOM), Bangalore, 2014, pp. 1-6.

[12] H. Rouabeh, C. Abdelmoula and M. Masmoudi, "A New Efficient Connected Component Labeling Algorithm and Its VHDL Circuit," 28th. International Conference on Microelectronics (ICM), Giza, 2016, pp. 105-108.

[13] H. Kong, H. C. Akakin and S. E. Sarma, "A Generalized Laplacian of Gaussian Filter for Blob Detection and Its Applications," IEEE Transactions on Cybernetics, 2013, vol. 43, no. 6, pp. 1719-1733.

[14] H. Kong, S. E. Sarma and F. Tang, "Generalizing Laplacian of Gaussian Filters for Vanishing-Point Detection," IEEE Transactions on Intelligent Transportation Systems, 2013, vol. 14, no. 1, pp. 408-418.

[15] R. S. Babu, B. Radhakrishnan and L. P. Suresh, "Detection and Extraction of Roads From Satellite Images Based on Laplacian of Gaussian Operator," International Conference on Emerging Technological Trends (ICETT), Kollam, India, 2016, pp. 1-7.

[16] Kim, H., J. Do, et al., “Vehicle Detection Using Running Gaussian Average and Laplacian of Gaussian in the Nighttime”, Future Generation Information Technology Conference, FGIT 2012, Gangneung, Korea, Proceedings. TBerlin, Heidelberg, Springer Berlin Heidelberg, pp. 172-177.

[17] Xianlei Long, Shenhua Hu, Yiming Hu, Qingyi Gu, Idaku Ishii, "An FPGA-Based Ultra-High-Speed Object Detection Algorithm with Multi-Frame Information Fusion", Sensors (Special Issue: Advanced Interface Circuits and Systems for Smart Sensors, 2019, pp. 1-16

[18] Chen B., Shi L., Ke X. A., “Robust Moving Object Detection in Multi-Scenario Big Data for Video Surveillance”, IEEE Trans. Circuits Syst. Video Technol. 2018, 29, pp.982–995

[19] Shaonan Zhang, “Real Time Image Processing on FPGAs”, 2018, PhD. Thesis, University of Liverpool.