WWW.IJRPR.COM

**International Journal of Research Publication and Reviews**

Journal homepage: www.ijrpr.com ISSN 2582-7421

# A Survey on Blob Detection Methods in Images

***Priya Panchal [a], Kirti Bhatia [b], Rohini Sharma [c]***

[a] *Student, Sat Kabir Institute of Technology and Management, Bahadurgarh, India*
[b] *Assistant professor, Sat Kabir Institute of Technology and Management, Bahadurgarh, India*
[b] *Assistant professor, GPGCW, Rohtak,, India*

**A B S T R A C T**

Image enhancing, image segmentation, pattern recognition, and object detection are examples of common embedded vision (EV) algorithms. The extraction and measurement of the things that are present in the scene are topics covered by object detection. Background segmentation and blob analysis are the two operations that can be further broken down into this assignment. Blob analysis focuses on the identification of particular foreground regions that indicate object shapes, whereas background segmentation involves the separation of background and foreground information. Blob detection is the first phase of blob analysis. Blob identification looks for round or bright items against a dark backdrop or vice versa in order to determine objects of interest or blobs in the graphic. Blob detection is a method to find connections between visual objects, or neighbouring pixel areas with the same colour value. High data transfers (image data saved on storage media), implementation of algorithms (serial vs. parallel), and hardware resources (CPU vs. GPU) all cause bottlenecks in ordinary EV operations. To create an effective EV system, it may be necessary to create specialised hardware that is intended to process raw visual data.

Keywords: Blob Detection, Feature Extraction, Classification

## 1. Introduction

Over the past few years, usage of automatic image recognition technologies has been continuously increasing (Malamas, 2003). In modern electronics, including laptop computers, mobile phones, and personal digital assistants (PDAs), as well as smart cars and automated surveillance systems, embedded cameras are frequently used. Applications for image processing today need to allow performing intricate calculations on huge quantities of visual data. Personal computers might supply this technology, but their high power requirements, limited size, and lack of mobility frequently prevent their usage in many vision-related applications. Specialised image processing platforms provide the tools required to carry out real-time image processing activities with an emphasis on efficiency using a mix of integrated computer power and video sensors. A branch of computer vision known as embedded vision (EV) focuses onsystems where sensors are frequently integrated into the hardware they operate. On-board processing necessitates a constant trade-off between computationally demanding algorithms and resource usage during system design. EV applications are numerous, and their implementation extends from human support tools like robots and other autonomous devices (Hegde, 2013) to amusement interaction platforms like video games (Camplani, 2013) and virtual reality systems (Todorov, 2015). Embedded sensor-based systems are frequently utilised in the automotive sector to tackle difficult tasks including obstacle detection (Torres-Torriti , 2013), lane departure avoidance (Besbes, 2015), and pedestrian recognition (Azami, 2017).

## 2. Related Work

Algorithms that only require one pass are a distilled version of sequential approaches. To identify and label blobs in a single image scan is the major goal of the algorithms in this subcategory (Tapia-Espinoza, 2013) suggests implementing border object detection to do this. This method relies on predicting the outlines of blobs and filling the interior of each blob with a label that depends on nearby pixel areas.

These concepts have been used to create blob identification algorithms that are widely documented. Platforms for broad computing are often targeted by the implementation. An efficient connected component labelling software is presented in the work in (Paralic , 2012). The OpenCV (Open Source Computer Vision) framework is utilised in the development of the application. The algorithm searches the input image for related sub-regions using pixel neighbourhoods, and then combines multiple sub-regions into a single region. The method employs a partial neighbourhood mask that creates a few isolated tiny holes.

For software-based video surveillance, the authors of (Nguyen , 2009) present a real-time blob detection technique. Their approach is based on doing a corrective phase before component detection. While labelling is being done, the correction uses morphological filtering, which fills in image gaps. It eliminates small, disconnected pixel patches and is known as neighbour foreground pixel propagation (NPP). The technique appears to attain great processing speeds, although the algorithm's maximum throughput is not specified.

Vision systems are more common and necessary in today's embedded devices, such as PDAs, sensor networks, and mobile phones. However, the demands on size, power consumption, and the necessity to operate in real-time define its performance and require that the existing algorithms be revised in order to adapt them to the new platforms. Since CCL algorithms are vital components to carry out intermediate processing tasks and play a crucial part in the creation of high level image processing systems, this adaption process is extremely significant for CCL algorithms.

Due to its importance, significant effort has been made to create and refine this type of algorithm, whose goal is to assign a distinct label to each pair of pixels in the image that have the same features. As a result, several study paths have been taken that can be categorised using a wide range of criteria (such as the regularity of memory accesses, the manner of image representation, etc.). In addition, the advancement of integrated circuit fabrication technology has made feasible to use many parallelism techniques that were previously ignored because of their extremely high prerequisites.

## 3. Various Types of blob Detection Methods

### *3.1 Stack Algorithm*

A new label is applied to each unlabeled data pixel that is discovered while scanning the image. The identical labels are applied to and pushed onto all of the neighbours of the current data pixel. Put a value at the top of the stack, give the pixel at the top-referred location the same label, and push its neighbours into the stack. Pixels are given the same label and all nearby pixels are pushed into the stack even when it is not empty. When the stack is empty, each pixel in the same blob receives a unique label. The following data pixel is looked for and given a new name when the entire blob has been labelled. Until all of the blobs in the image are labelled, this process is repeated (Jankowski ).The fundamental drawback of a stack-based algorithm is that the stack might get quite huge.

### *3.2 Two Pass Scanning Algorithm*

The image is twice scanned in a two-pass method, from left to right. A mask containing all previously labelled neighbour pixels is applied during the first pass. The current pixel is given the minimum label from the mask, and the label equivalence data is kept in the equivalence table. Each label is changed to the root label found in the similarity table during the second pass. For each new label in the two-pass process, an entry in the equivalence table is created. Thus, it is necessary to have at least NL * NL of memory space, where NL is the number of provisional labels assigned during the first pass. Additionally, four neighbours for each object pixel must be compared during the first run in order to determine the minimum labels, and the equivalence table must be changed for each of them. This causes the processing time for huge images to increase significantly.

The two-pass algorithm developed in (Bailey , 2011) serves as the foundation for the FPGA design. The algorithm uses a binary picture as its input. Background pixels are removed from the first picture scan, and foreground pixels are categorised according to their neighbourhood labels. Possible connectivity issues are solved with a second pass. A Spartan-3A DSP FPGA board, operating at 27 MHz and processing 640 x 480 (VGA) images at 60 frames per second, is used for the hardware implementation.

In (Mauch , 2014), the authors use a single-pass component-labeling approach in yet another implementation. Their technology consists of a bespoke Shack-Hartmann wavefront sensor (SHWVFS) paired to a charge-coupled device (CCD) array, which collects 224 244 frames. A Xilinx Spartan-6 150LX FPGA running at 70 MHz is used to implement the detecting algorithm. The SHWVFS gadget enables a 905 FPS high-speed throughput.

### *3.3 Union Find Algorithm*

The two pass scanning algorithm has been improved by the union find algorithm. In order to reduce the amount of comparisons, a decision tree is utilised. In the second optimisation strategy, union find is used to resolve equivalent labels. A one-dimensional array is used to implement the union find data structure. Time is greatly reduced by using an array.

### *3.4 Contour Tracing Algorithm*

In a way, this algorithm is also a two-pass one. In the contour tracing approach, a scan line is moved row by row until it reaches the border of an unlabeled object, at which point it traces all of the boundary pixels of the blob and gives them the same label. This approach is predicated on the idea that a blob's border defines it. Until every boundary pixel is labelled, this process is repeated (Chang, 2004) . To find the interior contour, the same procedure is done. All pixels between two contours are labelled with the same label during the second pass.

### *3.5 Run Based Two Scan Labeling Algorith*

Data from the first scan is gathered. A new provisional label is given to the current run if it is not tied to the run that came before the scan in the preceding row. It is given the same temporary label as the leftmostconnected run if it is connected to runs in the row above the scan row. The provisional label sets for all connected runs are then merged, with the smallest of them being used as the representative label. Each provisional label is changed to its representative label during the second scan. However, with this technique, merging operations are necessary when the current run and the preceding run are related. Additionally, we must select the minimal label from the label list.

### *3.6 In Visual Surveillance*

Blob detection is a crucial step in several computer programmes, including intelligent visual surveillance. Previous blob detection techniques, though, are still computationally demanding, making it prohibitively difficult to implement real-time multichannel intelligent visual surveillance in a workstation or even one-channel real-time visual surveillance in an embedded system. Blob detection in visual surveillance involves a number of processing steps, including the extraction of the foreground mask, the correction of the foreground mask, and the segmentation of the blob using connected component labelling.

The correction of the foreground mask required for an accurate detection is often carried out via morphological procedures like opening and shutting. Morphological operations are computationally prohibitive, and because they need processing that is very distinct from that of connected component labelling, it is challenging to do them in parallel with that routine.

The use of visual navigation systems is used in many different contexts, including traffic enforcement and the direction of autonomous vehicles. One of the crucial processes is object detection, which shows the location of the barrier in the visual situation and identifies it. Blob detection has been used to identify objects and gather the necessary data about them. The blob detection algorithm requires more hardware resources, such as more logic gates, in order to be implemented on an FPGA. Applications for visual navigation systems include traffic surveillance, autonomous vehicle guidance, search and rescue operations, outdoor and indoor building inspection, etc. Visual navigation consists of four phases. Creating a scenario is the first step. Object detection and classification as static or moving constitute the second stage. The third stage involves finding a path free of obstacles. The beginning of navigation action is the last phase.

Blob feature adoption for achieving object detection has been facilitated by the ease of getting blobs from a picture. A big collection of pixels that can be used to detect things is known as a binary large object (blob). Applications like visual navigation require object identification algorithms to be processed in real time. Because of its parallel processing and pipelining capabilities, Field Programmable Gate Array (FPGA) is preferred for image processing applications. The blob detection technique must be implemented on an FPGA,which demands more hardware in the form of memory blocks, logic gates, etc.

### *3.7 GPU-Based*

In the literature on image processing, GPU-based solutions are frequently investigated. However, they are rarely employed as embedded computing systems because of their excessive power consumption. The authors create a comprehensive face detection system that is CUDA-implemented on an NVIDIA GTX 470 graphics card. The system performs a variety of image processing operations, concentrating on blob identification utilising a parallel integral image computation. Using a multiscan kernel method, this operation is designed to efficiently compute integral pictures in parallel. A parallel Haar evaluation filter that scans the input image for facial features comes next in the processing chain. The system analyses 1920 x 1080p HD video at 35 frames per second. In 2.3 ms, the parallel integral picture is computed.

Blob detection applications for embedded platforms have grown in popularity recently. A complementary metal-oxide semiconductor (CMOS) sensor that is housed in parallel with a digital signal processor (DSP) is one example given in (Srinivasa , 2007), where the authors propose a face recognition technique. The system is designed to speed up image analysis. The system recognises the faces of five people in 4.2 ms on 640 480 (VGA) images at 30 frames per second. In (Wang , 2003), the authors propose a parallel approach to reduce the computing burden for embedded blob identification. Four concurrent processing elements (PE) are assigned to four partitions of the input image. The labels of an input pixel and its four nearby neighbours are compared by each PE. Based on this knowledge, the corresponding label is generated, necessitating a minimum of four image scans. The final labelled image is produced during a merging stage.

### *3.8 Pseudo partitioning*

The strategy described in (Ercan, 1999) suggests an algorithm built on a technique known as pseudo partitioning. Without a merging phase, this method enables labelling. Nine DSPs are used in an array to run the algorithm. Based on neighbourhood search, each DSP performs component labelling for a segment of the image. This method produces good performance and speed since it does not require DSP intercommunication. Authors introduced the light speed labelling (LSL) algorithm. This blob detection technique is intended for RISC processor architecture implementation. It focuses on optimising the CPU pipeline by lowering memory accesses and conditional assessments. It is built on the idea of "line-relative labelling," which is a segment-based adjacency detection technique used to make equivalence between image rows easier to understand. For the algorithm to generate corrected input, three image scans are necessary. The authors of (Yuhai, 2011) suggest using a one-scan technique to consume the least amount of memory. The processing of picture sub-regions in search of related components is based on pseudo-partitioning. When a connected component is found, the binary image-based approach propagates labels. Label information may be stored in just 75.6 kbits of on-chip RAM because to small sub-region pictures. Frame sizes of 512 x 512, 1280 x 720, 1024 x 1024, 1920 x 1280, and 1280 x 1280 can all be processed by the architecture. The blob detection system is implemented on a Stratix II FPGA running at 97.4 MHz; it processes 1280 1280 frames at a frame rate of 49 FPS.

### *3.9 Digital Signal processing*

Authors (Bramberger, 2006) proposed a personalised computer platform method for monitoring and tracking vehicles. The authors claim that the hardware architecture was selected to support performance, versatility, and quick prototype development. An inbuilt monochrome CMOS image sensor allows for picture acquisition. This gadget transmits VGA-resolution pictures at 30 frames per second. The actual image processing techniques are carried out by the DSP array at a speed of 80 billion instructions per second. Real-time image processing can also be accomplished by creating unique embedded architectures. The ability of FPGA-based solutions to create custom hardware and software has increased their use in computer vision. The development of a complete system on a chip (SOC) using FPGA technology allows for the use of bespoke processors to analyse data that has already been acquired.

### *3.10 Intersecting pixels*

A method based on the recognition of crossing pixels on each image row and column is suggested by the authors in (Kiran , 2013). A series of filters (such as grayscale conversion, median filtering, and an ultimate threshold action) are used to pre-process the image first. Pre-processing is done to improve detection outcomes, and after that, a series of logical tests are run to identify each intersecting pixel. Their solution computes object area and centroid and is made for visual navigation systems. The technique is implemented on a 100 MHz Xilinx Virtex V FPGA board. Their system performs 1024 1024 picture processing at 61.72 FPS (16.2 ms) and 100 100 image processing at 4545.45 FPS (0.22 ms). It uses four blocks of RAM and detects a total of five things.

By dividing the input frame into vertical slices, the work of [20] takes advantage of parallel processing that is built into hardware architecture. The partitioning concept is comparable to the research shown in (Bailey , 2011). A pixel processing component simultaneously examines each image slice for related components. All label data from various distinct pixel processing components is combined in one place. The technique is implemented on a 1024 by 1024 input frame, 136.4 MHz-clocked Xilinx Virtex 6 XC6VLX240T FPGA. An image with a resolution of 1024 x 1024 can be viewed at 1.1 GPixels per second (1049 FPS) because to the architecture's high throughput optimisation.

The same authors present an improved structure in (Klaiber , 2015). A control structure that can identify the last pixel of an image is one of the new features. Others include memory resource optimisation, recycling of old labels, and streamlining labelling procedures by lowering the number of label lookups required for each pixel. Blobs are picked up by the hardware architecture in images ranging in resolution from 640 480 to 7680 4330. With the aid of various FPGA families, the implementation is assessed.

### *3.11 Virtual Reality (VR)Applications*

A blob detection method for virtual reality (VR) applications is suggested by the authors in(Bochem, 2011). Blobs are found by the system, and their centre coordinates are computed instantly. An adjacency test employing a four- and eight-pixel neighbourhood is used to find blobs. Once a blob has demonstrated adjacency, a computer calculates its coordinates using bounding boxes. It operates by looking for each blob's minimum and maximum XY coordinates. Then, a FIFO structure is written using the properties of the blob. The system is implemented on a DE2 Altera FPGA board that operates at 125 MHz and can process 640 x 480 frames at a top speed of 50 frames per second without using a monitor output. 13,311 logic elements and 239,316 memory bits are used as resources.

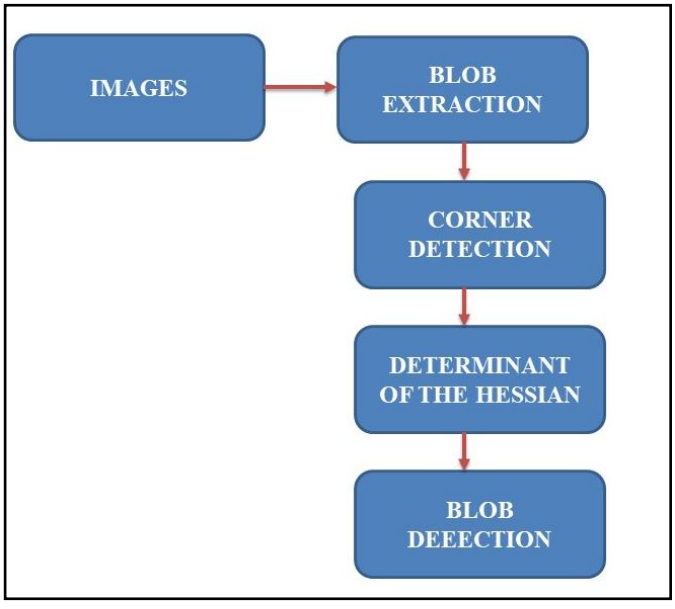


Fig. 1: Process of Blob Detection

## 4. Conclusion

For computer vision applications, blob detection is a typical task. It is frequently used as an algorithm that uses picture storage on general-purpose computing platforms. System memory and computing capacity are constrained for embedded system implementation, hence other methods must be developed. In this research, a blob detection method was designed with a system-on-chip application in mind and recommended for implementation on embedded hardware. Fast processing and low memory utilisation have been emphasised.

The trade-off between memory throughput, processing speed, and detection accuracy must be balanced when creating an object detection algorithm for embedded hardware. With the construction of data structures used to express blob information across the input image , these problems can be partially resolved. If contiguity has been established, a label can be spread from a parent node through the rest of the tree, reducing the need for additional processing. It is feasible to express blob data using linked-list. A linked-list is a type of node structure that has a pointer to a node that comes before or after it [56]. This list offers design freedom while supporting dynamic data because it can swiftly change its root node without effecting its child nodes.

**References**

Malamas E.N., Petrakis E.G.M., Zervakis M., Petit L., Legat J. A survey on industrial vision systems, applications and tools, image and vision computing. Image Vis. Comput. 2003;21:171–188.

Hegde G., Ye C., Robinson C., Stroupe A., Tunstel E. Computer-Vision-Based Wheel Sinkage Estimation for Robot Navigation on Lunar Terrain. IEEE/ASME Trans. Mechatron. 2013;18:1346–1356

Camplani M., Mantecon T., Salgado L. Depth-Color Fusion Strategy for 3-D Scene Modeling With Kinect. IEEE Trans. Cybern. 2013;43:1560–1571.

Kumar V., Todorov E. MuJoCo HAPTIX: A virtual reality system for hand manipulation; Proceedings of the 2015 IEEE-RAS 15th International Conference on

Tapia-Espinoza R., Torres-Torriti M. Robust Lane Sensing and Departure Warning under Shadows and Occlusions. Sensors. 2013;13:3270–3298..

Besbes B., Rogozan A., Rus A.M., Bensrhair A., Broggi A. Pedestrian Detection in Far-Infrared Daytime Images Using a Hierarchical Codebook of SURF. Sensors. 2015;15:8570–8594.

Azami N., Idrissi D.E., Amrane S., Harmouchi M. Computer blob detection and tracking for highly repeatable optical fiber sensor; Proceedings of the 2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14); Rabat, Morocco. 7–8 May 2014; pp. 1–5.

Tapia-Espinoza R., Torres-Torriti M. Robust Lane Sensing and Departure Warning under Shadows and Occlusions. Sensors. 2013;13:3270–3298.

Paralic M. Fast connected component labeling in binary images; Proceedings of the 2012 35th International Conference on Telecommunications and Signal Processing (TSP); Prague, Czech Republic. 3–4 July 2012; pp. 706–709.

Nguyen T.B., Chung S.T. An Improved Real-Time Blob Detection for Visual Surveillance; Proceedings of the CISP ’09. 2nd International Congress on Image and Signal Processing; Tianjin, China. 17–19 October 2009; pp. 1–5.

M. Jankowski and J.P.kuska “Connected Components Labeling –algorithms in Mathematica, Java, C++ and C#”.

Ming, Y. S., . Li, H. D, He., X. M. (2016) .Contour completion without region segmentation. IEEE Transactions on Image Processing, vol. 25, no. 9, pp. 3597–3611.

F. Chang, C. J. Chen, and C. J. Lu, “A linear-time componentlabeling algorithm using contour tracing technique,” Computer Vision and Image Understanding, vol. 93, pp. 206–220, 2004..

bor Wang K., lin Chia T., Chen Z., Lou D. Parallel execution of a connected component labeling operation on a linear array architecture. J. Inf. Sci. Eng. 2003;19:353–370.

Srinivasa Kumar D., Krishna I.V.M., Tiruveedhula1 V.R. Real-time Face Recognition Using SIMD and VLIW Architecture. J. Comput. Inf. Technol. (CIT) 2007;15:143–149..

Ercan M., Fung Y.F. Connected component labeling on a one dimensional DSP array; Proceedings of the IEEE Region 10 Conference TENCON 99; Cheju Island, Korea. 15–17 September 1999; pp. 1299–1302.

Bramberger M., Doblander A., Maier A., Rinner B., Schwabach H. Distributed embedded smart cameras for surveillance applications. Computer. 2006;39:68–75..

Bailey D.G. Design for Embedded Image Processing on FPGAs. 1st ed. Wiley-IEEE Press; Hoboken, NJ, USA: 2011..

Mauch S., Reger J. Real-Time Spot Detection and Ordering for a ShackHartmann Wavefront Sensor with a Low-Cost FPGA. IEEE Trans. Instrum. Meas. 2014;63:2379–2386.

Kiran D., Rasheed A.I., Ramasangu H. FPGA implementation of blob detection algorithm for object detection in visual navigation; Proceedings of the 2013 International conference on Circuits, Controls and Communications (CCUBE); Channasandra Bengaluru, India. 27–28 December 2013; pp. 1–5.

Klaiber M.J., Bailey D.G., Ahmed S., Baroud Y., Simon S. A high-throughput FPGA architecture for parallel connected components analysis based on label reuse; Proceedings of the 2013 International Conference on Field-Programmable Technology (FPT 2013); Kyoto, Japan. 9–11 December 2013; pp. 302–305.

Klaiber M.J., Bailey D.G., Baroud Y.O., Simon S. A Resource-Efficient Hardware Architecture for Connected Components Analysis. IEEE Trans. Circuits Syst. Video Technol. 2015:1–16.

Yuhai L., Mei K., Dong P. An Effiient and Low Memory Requirement Algorithm for Extracting Image Component Information. Int. J. Adv. Intell. 2011;3:255–267..

Bochem A., Kent K.B., Herpers R. FPGA based real-time object detection approach with validation of precision and performance; Proceedings of the 2011 22nd IEEE International Symposium on Rapid System Prototyping (RSP 2011); Karlsruhe, Germany. 24–27 May 2011; pp. 9–15