Int. J. Advance Soft Compu. Appl, Vol. 11, No. 3, November 2019
ISSN 2074-8523

Algorithm Border Tracing vs Scanline in Blob Detection for Robot Soccer Vision System

Awang Hendrianto Pratomo[1], Anggit Ferdita Nugraha[1], Joko Siswantoro[2], Mohammad Faidzul Nasruddin[3]

[1]Department of Informatics Engineering, Faculty of Industrial Engineering, Universitas Pembangunan Nasional "Veteran" Yogyakarta, Jl. Babarsari No.2 Tambakbayan 55281 Yogyakarta Telp. (0274) 485323
e-mail: awang@upnyk.ac.id, anggitferdita@gmail.com
[2]Department of Informatics Engineering, Faculty of Engineering, Universitas Surabaya, Jl. Kali Rungkut, Surabaya, 60293, Indonesia
e-mail: joko_siswantoro@staff.ubaya.ac.id
[3]Center for Artificial Intelligence Technology Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia 43600, UKM, Bangi, Selangor, Malaysia
e-mail: mfn@ukm.edu.my

Abstract

In a robotic soccer competition, the locations of the ball and the robot are recognized through the vision system in a strong dynamic environment. Therefore, the robot and the computer need to identify the object in real-time and accurately through the vision system. In this study, algorithm border tracing and algorithm scan line were applied in vision system for the robotic soccer team of the Universitas Pembangunan Nasional UPN "Veteran" Yogyakarta. Two algorithms were introduced to detect the blob in order to identify the robot, the ball and the other object in the field. The Experimental results showed that border tracing algorithm has 100% accuracy in all shape (triangle, square and rectangle) of blob. Whether scan line algorithm has 25% accuracy for triangle, 0% for rectangle and 100% for square. Therefore it has been shown that border tracing algorithm is superior to scan line.

Keywords: *bob detection, scanline algorithm, border tracing algorithm, vision system, robot soccer*

1 Introduction

Vision system is an important element that will determine the course of the game[1] in robot soccer. Vision device such as camera is a sensor used to capture all objects existed around the environment [2]. The captured objects are then processed to produce information used by a robot to run instructions based on its position[3]. To produce the information, each captured object is identified as a robot, a ball, or another object based on its blob shape and its color. This process is known as blob detection. In a robot soccer game, the vision system has a very important role in detecting robots, balls and all objects in the field rapidly, precisely, and accurately so that the robot can move according to a given instruction[4].

To ensure the robot soccer game runs well, the robot soccer vision system must be supported by a reliable and low computational vision algorithm. In addition, the algorithm should easily detect all objects in the field. Currently, the algorithm commonly used to recognize objects in a robot soccer field is scanline algorithm [5]. The algorithm employs an effective run-length encoding region to reduce the size of image by sampling along vertically spaced-apart scanned lines, and depends on the manual definition of elaborated models of each object[6]. By taking the sample line, the algorithm is very fast. However, the modeling of each object in the field is quite time-consuming and the algorithm is not easily extendable to new objects[7].

The Border Tracing algorithm is also known as Contour Tracing algorithm or Boundary Following algorithm. It is one of many algorithms for edge detection[8]. The border tracing algorithm works by tracing the boundaries on an image to produce a more efficient process for classifying certain patterns on the image[9].

The objective of this study is to compare the Scanline and Border Tracing algorithms in vision system for the robotic soccer team of the Universitas Pembangunan Nasional UPN “Veteran” Yogyakarta. The time processing and the accuracy of blob detection will determine for both algorithm.

2 System overview

2.1 Robot soccer

Robot soccer is one of the smallest football robot games involving motor control, radio communication, computer vision, image processing, motion planning, machine learning, and multi agent coordination [10-12]. Robot soccer is limited by size and moves automatically in a 220x180cm field. According to FIRA, the matches are divided into three categories, which are small field three-on-three (3 vs. 3), medium filed five versus five (5 vs. 5), and large field eleven versus eleven (11 vs. 11) (FIRA, 2014). The rules that used in this game are similar to human soccer games. This study employed robot soccer with the following specification

size 7.5 cm x 7.5 cm x 7 cm, Motor driver Mosfed, MotorDC Faullhabber 2212SR, and a Wireless Communication, as shown in Fig.1.

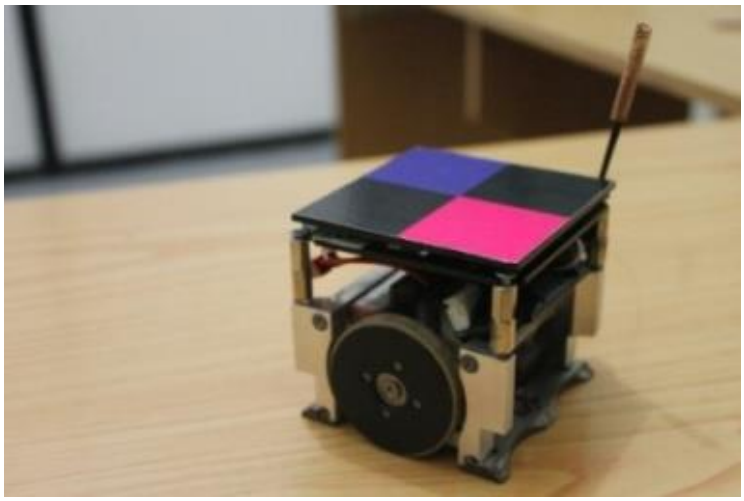

Fig. 1. Robot Mirosot Informatics Department UPN "Veteran" Yogyakarta.

2.2 Vision System

A Basler Scout scA640-74fc camera, as shown in Fig. 2, was used as a vision system to support the main computer in making decisions [1]. It met the requirement as a vision system in a soccer robot match, with are a digital camera with HD quality with minimum frame rate 50 fps [4].

The camera as a sensor was mounted on the top of the field such that it could detect robots, ball and other objects based on their shapes and colors by using object-based pattern recognition algorithm. The results of such process were location and orientation information of each object within the field. The information was used to determine the instructions for the robot in motion[5,13,14].

Fig. 2. Basler camera for vision system.

2.3 Object detection

In the vision system, object detection was used to know the kind of objects that are located around the robot in the field [1]. In identifying a robot, the vision system would detect the color patch of object placed at the top of the robot as the cover [4]. Color patches were made with specific color and consisted of team colors identity, and robot colors identity[15,17]. Color patches were also designed to help a human operator easily identify robots. The design of color patch made the robot soccer game more realistic as a robot player and can be easily used to identify the robot through its color that work like the number of the backs of human soccer players[16,17].

Object detection process used edge detection in four or eight directions of connectivity started from a certain point according to the boundary condition of the image. Connectivity was obtained based on the direction of the predefined notation, so that the search will always move along the edge and will stop when it is back to the starting point. The coordinates of detected boundary were then averaged to obtain the midpoint coordinate of the blob.

3 Methodology

3.1 Blob detection

In general the process of blob detection is illustrated in Fig. 3. In Fig. 3, the blob detection process is depicted in various functions symbolized by process notation and decision. Process notation is a notation that contains the main function for detection, whereas the notation of decision is a function of limitations to eliminate unnecessary processes so that the detection process becomes faster. The detail of every function is explained in the following sub sections.

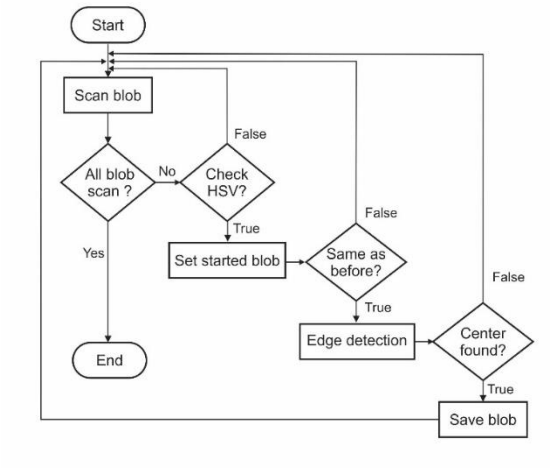

Fig. 3. Flowchart for blob detection

3.1.1 Scan Blob Function

The Scan Blob Function was used to scan every node or pixel in the frame. The processes in the Scan Blob Function are explained in the Fig. 4.

```
Input  : ColorId
Output : coordinate x, coordinate y

Step 1   : check each pixel in the frame from left to right starting
           from the top row to the bottom row.
Step 2   : Convert any pixel value into hue (H), saturation (S), and
           value (V) values. Then compare with the range of HSV values
           determined through the checkHSV function.
Step 3   : If the pixel value is still within the specified range of
           HSV values then check whether the point is close to the
           midpoint ever found before. If the distance is close, then
           skip the process. If it is far then continue the process to
           the SetStartedBlob function
Finished
```

Fig. 4. Process Scan Blob function.

3.1.2 CekHSV Function

CekHCS Function is the boundary function. It used in determining whether the point is part of the object or not. The process performed on this checkHSV function is to compare the value of HSV from the coordinate point of scanning blob results with the HSV value of the color identity that has been defined in the global variable. The processes in the CekHSV Function are explained in the Figure 5.

```
Input  : ColorId, Hue, Saturation, value
Output : true and false

Step 1   :Take the value of hue, saturation and values of the pixels
           scanned on the BLOB scan function
Step 2   : compare the value of hue, saturation, and value of pixels
           with hue, saturation and values of the color identity.
Step 3   : If the value of hue, saturation, and values are in the range
           of hue, saturation and value values of the color identity,
           then return the true value to the scanblob function.
           Otherwise, return the false value.
Finished
```

Fig. 5. Process CekHSV function.

3.1.3 SetStartedBlob Function

The SetstartedBlob function is an advanced function of the BLOB scan if the hue value, saturation, and values are in the range of hue, saturation and color identity values. This function was used to determine the starting point that will be the

reference in the performing of the edge detection process as well as being the end point to stop the edge detection process. The processes in the SetStartedBlob Function are explained in the Fig. 6.

```
Input  :  ColorId, coordinate x,
coordinate y
Output :  coordinate x_start,
coordinatey_start
Step 1    : Take the x coordinates, and y coordinates of the Scanblob
          function corresponding to the HSV value of the color identity.
Step 2    : compare the values of hue, saturation, and pixels to the
          left of the x coordinate point.
Step 3    : If the left point still matches the value of hue, saturation
          then subtract the coordinate value x.
Step 4    : repeat processes 2 and 3 until the value of hue, saturation,
          and value to the left of x coordinates no longer matches the
          value of hue, saturation and color identity values.
Step 5    : if the value of hue, saturation, and value of the left
          coordinate x, then the value x_mulai equal to the coordinate
          value x.
Step 6    : compare the value x_start with the previous value of
          x_start. If the value is different, then proceed to the edge
          detection function.
```

Fig. 6. Process SetStartedBlob function.

3.1.4 EdgeDetection Function

The EdgeDetection Function is a function that used to detect the edges of each blob and then calculate the average of which the result is the midpoint of the blob. The EdgeDetection function used the help function in charge of moving the search for the edge of the object by way of testing based on the eight-way notation of the wind as listed in Table 1. The processes in the EdgeDetection Function are explained in the Fig 7.

Table 1. Notation eight winds direction for edge detection algorithm.

Notation	Direction	Coordinate
1	Left	x-1, y
2	Upper left	x-1, y-1
3	Up	x, y-1
4	Upper right	x+1, y-1
5	Right	x+1, y
6	Bottom right	x+1, y+1
7	Bottoh	x, y+1
8	Bottom left	x-1, y+1

```
Input  : ColorId, coordinate x_start,
coordinatey_start, notation direction
Output : coordinate x_middle,
coordinatey_middle
Step 1   : compare HSV value x_start and y_start with HSV value in
         direction notation. Direction starts from notation 1.
Step 2   : if the value of HSV is appropriate, then point the
         coordinates according to the notation information. If not
         appropriate, then navigate to the next notation until you find
         the appropriate HSV value. if the notation is up to 8 then it
         will return to notation 1.
Step 3   : calculate the sum of the x and y coordinates passed by
         direction notation, then calculate the mean to be the
         coordinates x_tengah and the coordinates y_tengah.
Step 4   : repeated the process until it returns to the starting point
         of coordinatesx_start and coordinates y_start
Step 5   : if the midpoint has been found and back to the starting
         point, then proceed to the SaveCenterBlob function
Finished
```

Fig. 7. Process EdgeDetection fuction

3.1.5 SaveCenterBlob Function

Once the midpoint of the blob was found, next step was to save that point. The process of storing the midpoint was represented in the SaveCenterBlob Function. The processes in the SaveCenterBlob Function are explained in the Fig. 8.

```
Input  : ColorId, coordinate x_middle,
coordinate y_middle, notation direction
Output : coordinate x_middle _blob,
coordinate y_middle _blob
Step 1: check whether the coordinates x_middle, and y_ middle have
         been saved previously. If it has been saved then the process
         is complete. If it has not been saved already then save as
         value x_middle_blob, value y_middle _blob, and colorId
Finished
```

Fig. 8. Process SaveCenterBlob function

After the design process was done, the next step was the design of the test, where the initial test was done by blackbox method. The method was based on programming functionality that is more concerned with the suitability of output with the logic in making the function. After functionality testing was done, the next step was to develop the test into cases such as by placing the color patch in various positions. The patch that is in trial was a mode with square shapes, rectangles, and triangles. The testing process was done by detecting the shape of the color patches at various positions to determine the accuracy and processing speed of object detection as shown in Fig 9.

Fig. 9. The color patches position testing in the field.

3.2 Scanline Algorithm

The Scanline algorithm is one of many hidden surface removal algorithms that used to solve large memory usage problems with a single scan line to process all surface objects [7]. The scanline algorithm worked by sweeping the screen from top to bottom and a horizontal scan line of the y field was tested for all surfaces of the object. The intersection between the scan line and the surface was in the form of a line. The algorithm performs a scan with the y-axis direction so that it intersected all field surfaces with the direction of x and y and discarded the hidden lines[18]. Instead of scanning a surface once in a single process, it would be related to scanning multiple surfaces in a single process. As each scan line was processed, all polygon surfaces were cut by the scan line to determine which is visible. At each position along the scan line, a depth calculation was made for each surface to determine which is the nearest of the field of view. When the visible surface was determined, the intensity price was inserted into the buffer. The steps for Scanline Algorithm are depicted in Fig.10.

The advantages of the Scanline algorithm was less memory usage compared to other algorithms. This is because the process of sweeping the screen that uses a bit of function tends to be simpler than other algorithms so that the processing rate becomes faster, while the weakness of the scanline algorithm lies in the lack of detail on the processing result resulting in a non-optimal value[18].

In a soccer robot match, the scanline algorithm was used to find the midpoint of the blob on the detection process that used in the vision system by Merlin Inc. The process of determining the center point of the blob was done by scanning the entire row until it found the starting point of the specified condition. After the start point wass found, next step was scanning the line under the condition (scanning right).Fig. 11 shows the flowcart for Scanline algorithm.

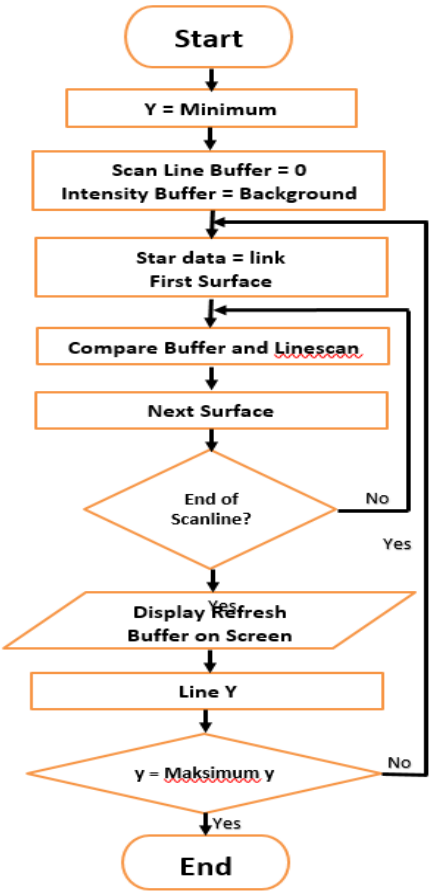

Fig. 10. Flowchart for Scanline algorithm.

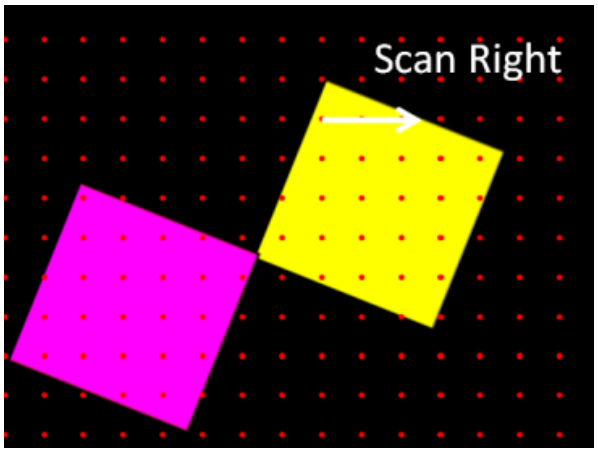

Fig .11. Scanning right process

After the right point was found, the next step was to find the left point by scanning the screen from right to left on the line (scan left), as shown in Fig. 12.

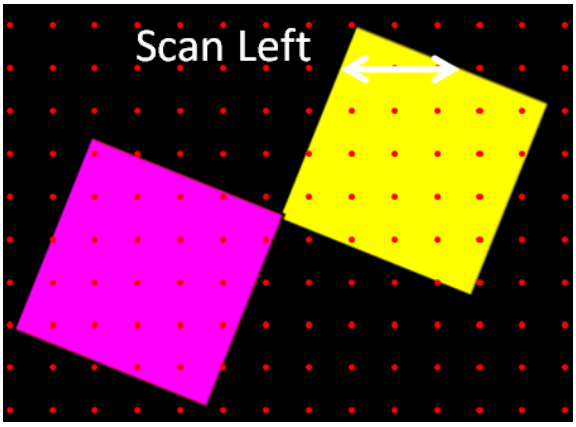

Fig 12. Scanning left process

From both points found, the next step was calculated its midpoint to be a reference for scanning down and scanning upwards to determine its midpoint. From the midpoint, scanning right and scanning left were again performed and the new middle point was calculated the middle point blob, as shown in Fig. 13[18,20].

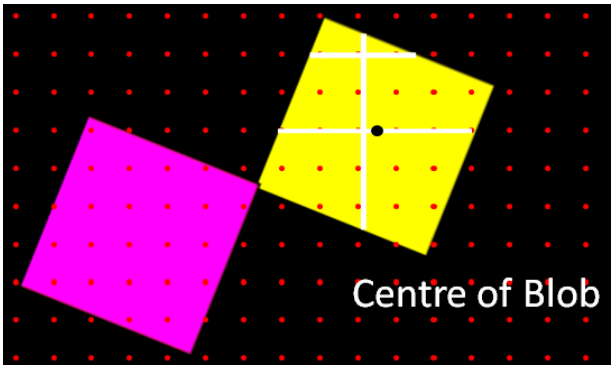

Fig 13. Blob center result

3.3 Border Tracing Algorithm

The Border Tracing algorithm is one of many algorithms for edge detection[9]. The border tracing algorithm is also known as Contour tracing algorithm or boundary following algorithm[21]. The border tracing algorithm worked by tracing the boundaries of images to produce a more efficient process of classifying certain patterns in a digital image [8]. This border tracing algorithm required a starting point that serves as a start and stop sign in tracing the edge. The image edge search process was connected using a connectivity known as direction notation. The notation was classified into four connectivity directions and eight connectivity

directions according to the conditions specified, as shown in Fig. 14. So that the search would always move through the edge based on the connectivity that has been determined and will stop when it is back to the starting point.

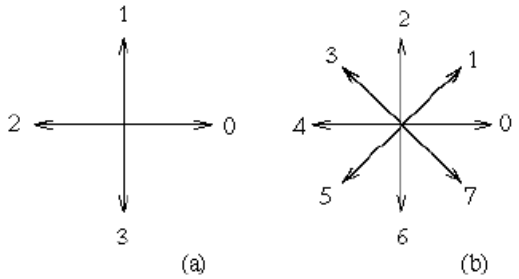

Fig 14. (a) Direction notation uses 4 Connectivity (b) Direction notation uses 8 Connectivity for object edge detection with Border Tracing algorithm

4 Results and Discussions

The test was performed according to the position of the color patch shape and the planned test position in the research methodology chapter. Color patches used in testing were the shape of square, rectangle and triangle. The various position of color path in testing for the shape of square, rectangle and triangle are shown in Fig. 15, Fig. 16, and Fig. 17, respectively. The test is aimed to compare processing time and detection accuracy between Scanline algorithm and border tracing algorithm.

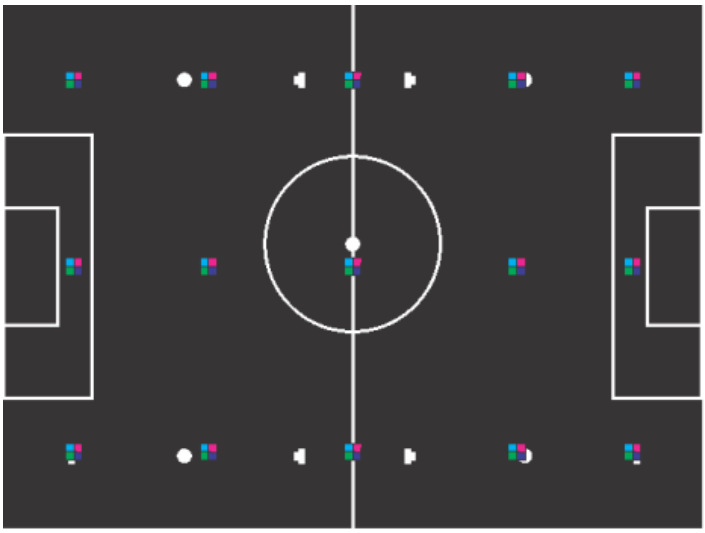

Fig 15. Square color patches testing at various positions.

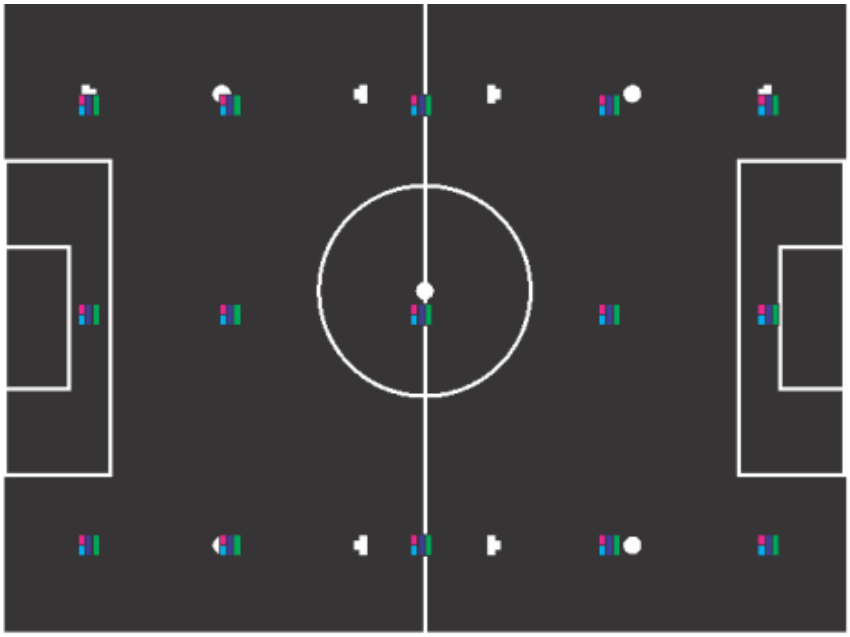

Fig 16. Rectangle color patches testing in various positions.

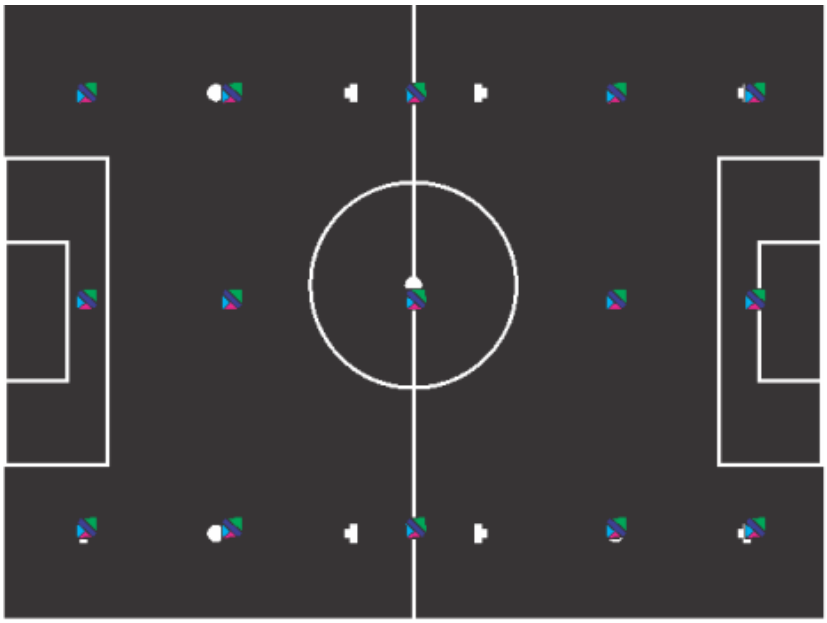

Fig 17. Triangle color patches testing at various positions.

4.1 Testing for processing time comparison

From various positions that have been designed with various shape of color patches, the process was done by loading one of the color patch in each position and all the positions using both vision systems for the next recorded simulation results based

on the speed of the process in detecting blob. Processing time in millisecond (ms) was recorded and the results are tabulated in Tabel 2.

Table 2. The processing time for detection process.

No	Position	Scanline (ms)			Border Tracing (ms)		
		S	R	T	S	R	T
1	A	10	9	8	9	9	11
2	B	10	8	8	10	10	11
3	C	10	9	12	11	10	11
4	D	10	8	9	9	11	11
5	E	10	7	8	9	9	9
6	F	10	8	11	9	11	9
7	G	10	7	9	11	11	10
8	H	10	8	9	9	12	10
9	I	10	8	12	9	11	9
10	J	10	9	9	10	11	10
11	K	10	8	8	10	9	10
12	L	10	7	8	9	12	12
13	M	10	8	8	10	10	11
14	N	10	9	8	9	11	10
15	O	10	8	8	9	9	9
16	Average	10	7	8	10	10	9

Note : S : Square; R: Rectangle; T: Triangle.

As can be seen in Table 2, the average processing time for detection process using scanline algorithm on the shape of square, rectangle, and triangle were was 10 ms, 8 ms, and 8.9 ms, respectively. On the other hand, the average processing time for detection process using border tracing algorithm were 9.5 ms, 10.3 ms, and 10.1 ms for the shape of square, rectangle, and triangle, respectively. Therefore, it can be concluded that border tracing algorithm is slightly slower than scanline algorithm in performing detection process.

4.2 Testing for detection accuracy comparison

The next test was to compare the detection accuracy for various blob models in various positions. The test was done as in the previous test that was by loading one of the color patches in each position as well as on all its positions using both algorithms. The number of detected blob was used to measure the accuracy level of the blob detection process and the results are tabulated in Table 3.

Table 3. The accuracy level for detection process

No	Position	The number of detected blob					
		Scanline			Border Tracing		
		S	R	T	S	R	T
1	A	4	0	1	4	4	4
2	B	4	0	1	4	4	4

3	C	4	0	1	4	4	4
4	D	4	0	1	4	4	4
5	E	4	0	1	4	4	4
6	F	4	0	1	4	4	4
7	G	4	0	1	4	4	4
8	H	4	0	1	4	4	4
9	I	4	0	1	4	4	4
10	J	4	0	1	4	4	4
11	K	4	0	1	4	4	4
12	L	4	0	1	4	4	4
13	M	4	0	1	4	4	4
14	N	4	0	1	4	4	4
15	O	4	0	1	4	4	4
16	Total	60 (100%)	0 (0%)	15 (25%)	60 (100%)	60 (100%)	60 (100%)

Note: S : Square; R: Rectangle; T: Triangle.

As can be seen in Table 3, the total detected blob for detection process using Scanline algorithm on the shape of square, rectangle, and triangle were was 60 (100%), 0 (0%), and 15 (25%), respectively. On the other hand, the total detected blob for detection process using border tracing algorithm was 60 (100%) for all shapes. Therefore, it can be concluded that border tracing algorithm is more accurate than Scanline algorithm in performing detection process.

5 Conclusion

From the results of experiment, the development of vision systems, especially in the detection process, was done by developing various shapes of blob on the color patch such as square, rectangular, and triangular shape and then detected with border tracing algorithm instead of Scanline algorithm. From the changes of the algorithm, various shapes of blob on the color patch could be detected properly.

The process of blob detection using border tracing algorithm had a higher accuracy rate than Scanline algorithm. The experimental results show that blob detection process with border tracing algorithm had 100% accuracy level on all blob shapes, while Scanline algorithm had smaller accuracy in detection blob with rectangle and triangle shapes.

The fact that of blob detection process using border tracing algorithm is slightly slower than scanline algorithm does not reduce the conclusion that the development of the vision system is better than ever. This is because the fast processing time on Scanline algorithm does not give good results when compared with the results obtained with border tracing algorithm.

ACKNOWLEDGEMENTS

The authors would like to Department of Informatics Engineering Universitas Pembangunan Nasional "Veteran" Yogyakarta for providing facilities. Fundamental Research Grant schema from Ministry of Research and Higher Education No. 04/UN62.21/PT/IV/2019.

References

[1] Pratomo, A. H., Zakaria, M. S., Nasrudin, M, F., Prabuwono, A. S., Liong, C-Y., Azmi, I. 2015. *Robust Camera Calibration for the MiroSot and the Androsot Vision Systems using Artificial Neural Networks.* Springer-Verlag Berlin Heidelberg, 571-585.

[2] Chen, Jianhui., Carr, Petter., Little, James J., , Where should cameras look at soccer games: Improving smoothness using the overlapped hidden Markov model, 2016, Computer Vision and Image Understanding, 159, 2017, 59-73, ISSN 1077-3142,https://doi.org/10.1016/j.cviu.2016.10.017.

[3] Weiss, N,. 2009. *Adaptive Supervision of Moving Objects for Mobile Robotics Applications.* Robotics and Autonomous System, Dortmund, Germany 57 : 982-995.

[4] Guodong, C., Xia, Z., Sun, R., Wang, Z., Ren, Z., Sun, L., 2011, "*A learning algorithm for model based object detection*" Ubiquitous Robots and Ambient Intelligence (URAI) 8th International Conference : 101-106.

[5] Boukezzoula, Reda., Coquin, Didier., Nguyen, Thanh-Long., Perrin, Stéphane., Multi-sensor information fusion: Combination of fuzzy systems and evidence theory approaches in color recognition for the NAO humanoid robot, 2018, Robotics and Autonomous Systems [100], pp. 302-316, ISSN 0921-8890,

[6] Bruce, J., Balch, T., and M. Veloso, *Fast and inexpensive color image segmentation for interactive robots*, Proceedings. 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2000) (Cat. No.00CH37113), Takamatsu, Japan, 2000, vol.3.pp. 2061-2066 doi: 10.1109/IROS.2000.895274

[7] Hermann, S., 2014 "*Evaluation of Scan-Line Optimization for 3D Medical Image Registration*" Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference : 3073-3080.

[8] Shi, J., Guo, C., "*The Extraction of Circle Contour Based on Improved Boundary Tracing Algorithm*" Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2012 4th International Conferencevol.2 : 104-107

[9] Lee, D.H., Kim, D.E., Hwang, K.H., Chung,C.W. danKuc, TaeY. 2006.*A Novel Color patch System for the Large League MIROSOT*. SICE-ICASE International Joint Conference,Busan, Korea.

[10] Dodds, R., Iocchi, L., Guerrero, P., Ruiz-del-Solar, J. 2011 *Benchmarks for Robotic Soccer Vision*. RoboCup 2011: Robot Soccer World Cup XV,

Volume 7416, DOI 10.1007/978-3-642-32060-6_36, Series ISSN 0302-9743. Springer Berlin Heidelberg, pp. 427-439.

[11] Vieira, F.C., Alsina, P.J. dan Medeiros, A.A.D. 2011. *Micro-Robot Soccer Team - Mechanical And Hardware Implementation*. DCA – CT – Universidade Federal do Rio Grande do Norte, Campus Universitário – Lagoa Nova – 59072-970 – Natal.

[12] Xia, Renbo., Hu, Maobang., Zhao, Jibin., Chen, Songlin.,Chen, Yueling., Fu, ShengPeng., Global calibration of non-overlapping cameras: State of the art, 2018,Optik, 158, 951-961, ISSN 0030-4026, https://doi.org/10.1016/j.ijleo.2017.12.159.

[13] Han, X., Li, J., Li, Y. & Xu, X. 2004.*An Approach of Color Object Searching for Vision System of Soccer Robot*. IEEE International Conference on Robotics and Biomimetics, Shenyang, China : 535-539.

[14] Jabson, N. G., Leong, K. G. N., Licarte, S. W., Oblepias, G. M. S., Palomado, E. M. J. & Dadios, E. P. 2008. *The Autonomous Golf Playing Micro Robot: With Global Vision and Fuzzy Logic Controller*. International Journal on Smart Sensing and Intelligent System, Manila, Philippines 1(4) : 824-841.

[15] Muramalla, C., S., Muthukumaran, B., 2015. "Color Vision Robot". International Journal of Science, Engineering and Technology Research (USETR) Vol. 4. ISSN: 2278-7798 : 852:854.

[16] Choi, I.S. & Ha, J.E. 2010. Easy Calibration Method for Omni-stereo Camera System. Proceedings of International Conference on Control, Automation and Systems: 2315-2317.

[17] Jiang, H.A., Peng, Q., Lee, H.A.C., Teoh, E.L., & Sung, H.L. 2004. Color Vision and Robot/Ball Identification for a Large Field Soccer Robot System. Proceedings of International the 2nd International Conference on Autonomous Robot and Agent: 323-327.

[18] Pratomo, Awang Hendrianto., Zakaria, Mohamad Shanudin., & Prabuwono, Anton Satria., Teknik Deteksi Warna pada Robot Sepak Bola MiroSot Menggunakan Algorima Pemindaian Garis., Proceding of National Conference SENATKOM 2015, Padang, Indonesia, ISSN : 2460-4690

[19] Salma, R., Hidayatno, A. Isnanto, R. 2011, *Aplikasi Penghitung Jumlah Wajah Dalam Sebuah Citra Digital Berdasarkan Segmentasi Warna Kulit*, Jurnal Undip.

[20] Pratomo, A. H., Zakaria, M. S., Prabuwono, A. S., &Liong, C.-Y. 2013. *Camera Calibration: Transformation Real-World Coordinates into Camera Coordinate Using Neural Network*. Springer-Verlag Berlin Heidelberg, 345-360.

[21] Kang, L., Zhong, S., Wang, F., 2011 "*A new contour tracing method in a binary image*," Multimedia Technology (ICMT), 2011 International Conference, pp: 6183-6186