SITEKIN: Jurnal Sains, Teknologi dan Industri, Vol. 23, No. 1, December 2025, pp.198 - 208
ISSN 2407-0939 print/ISSN 2721-2041 online

Vote Detection on Ballots Using Thresholding and Centroid Detection Techniques

Lailatul Qadriah[1], Nur Hamid[2]
[1.2] Department of Mathematics education, Faculty of Social and Humanities, Universitas Nurul Jadid
Jl. PP Nurul Jadid, Karanganyar, Paiton, Probolinggo, Jawa Timur 67291
Email: lailatulqodriah057@gmail.com[1], nurhamid@unuja.ac.id[2]

ABSTRACT

General elections are an agenda carried out to elect and determine leaders in each region. One of the important stages in the general election process is the vote-counting stage. This study aims to implement several digital image processing methods. Digital image processing plays an important role in the automatic reading of ballot papers to increase the speed of the vote-counting process. In this study, the process of reading ballot images was conducted to produce numerical data based on the coordinates of specific parts of the image. Image processing was performed using GNU Octave software, which is simple yet effective in detecting votes on ballot papers and converting them into numerical data based on centroid coordinates. This method has advantages in terms of implementation simplicity and computational efficiency. The main stages of this study include image conversion to grayscale, thresholding, black pixel detection, segmentation, centroid coordinate detection of punched ballot marks, and conversion into numerical form. In this study, 47 ballot image samples were used. The results of this study show that this method can achieve an accuracy rate of 78.7%.

Keywords: *image processing, ballot, centroid, general elections, GNU Octave*

Introduction

The general election, held once every five years, is a form of the democratic process. One of the important stages in an election is the voting stage, followed by converting the physical results into a record. This stage requires a considerable amount of time due to several processes that must be carried out in accordance with the recapitulation procedures by the election committee. Therefore, converting ballot paper documents into digital image formats is proposed to automatically detect the vote counts for each candidate pair. GNU Octave software is then used to detect vote counts on ballot images that have been converted into digital form, with the aim of improving the efficiency of reading digital-based vote images and accelerating computation time through the application of image processing techniques.

A previous study conducted by Emdud F. Agyemang et al., entitled *"Baseline comparative analysis and review of election forensics: Application to Ghana's 2012 and 2020 presidential elections"* (Heliyon, 2023), applied baseline screening methods such as Benford's second-order test and Hartigan's dip test to detect anomalies in the distribution of vote counts at the aggregate results level in order to identify fraud or irregularities [1].

Based on the research conducted by Joseph Redmon et al., *"You Only Look Once: Unified, Real-Time Object Detection"*, the YOLO method was used to detect objects by mapping pixels directly to the location and class of the object [2]. Meanwhile, in the study by Theron Ji et al., *"An Analysis of Write-in Marks on Optical Scan Ballots"*, image processing methods were used to detect vote counts based on handwritten candidate names in designated areas, as well as filled circles or bubbles. The results showed that nearly half of the voters did not fill in the bubble despite writing the candidate's name, which meant that votes could potentially go undetected by conventional systems [3].

Digital image processing is a field concerned with the manipulation of digital images using computers. This field serves various purposes, including enhancing image quality or extracting information from images [4], [5]. Common techniques in digital image processing include grayscale conversion, thresholding, black pixel search algorithms, segmentation, object labeling, and data extraction and conversion.

A grayscale image is a representation that uses only variations of gray intensity without color [6]. Each pixel in a grayscale image has an intensity ranging from black (0) to white (255) [7]. Grayscale images are widely applied in various applications, one of which is digital image processing.
Thresholding is a technique for converting grayscale images into binary (black-and-white) images by setting a threshold value: pixels with values above the threshold become white, while the rest are turned black. This process produces an image that separates the object from the background, where the object is represented by pixels with a value of 1 (white) and the background by pixels with a value of 0 (black) [7]. Thresholding is also a method for converting grayscale images into binary images [8].

In this process, each pixel can be accessed using row and column indices, allowing iteration to easily search for black pixel values. The black pixel search is used to detect the starting or ending boundary of an object in the image. This search is performed using a linear search algorithm—a method that does not require sorted data, examining each element one by one until the desired value is found. The algorithm stops when a pixel with a value of zero (0) is found and is then automatically stored [9], [10], [11].

Image segmentation is the process of dividing an image into smaller parts (segments), each consisting of pixels with similar characteristics [12], [13]. Image labeling is the process of assigning labels (annotations) to pixels in an image to mark specific different regions [12], [14]. During the labeling stage, centroid detection is performed to find the central point of each group, making it easier to assign labels to the detected centroids.

Feature extraction is the process of retrieving important information from an image, such as shape, texture, size, position, and spatial relationships between objects. This process is useful for further analysis, such as classification or object recognition, and also helps to simplify the data without losing important information. [4], [15], [16].

Image data conversion aims to adjust the data format to suit the requirements of subsequent processing steps [17]. In this study, numeric arrays conversion to strings and strings conversion to numbers [18]. Converting a numerical array to a string is the process of transforming an array containing numbers into a string from to facilitate data manipulation [19]. Converting a string to a number involves transforming data in string from into a numeric representation, so that it can be used in mathematical operations [20].
Accuracy is a measure of the correctness of an observation, measurement, or prediction compared to the actual value. The accuracy value can be calculated using the formula [21]:

$$\text{ACCURACY} = \left(\frac{1}{n}\sum_{i=n}^{n}\delta_i\right) \times 100\% \quad (1)$$

The percentage error value can be calculated using the formula:

$$\text{ERROR RATE} = \left(\frac{1}{n}\sum_{i=n}^{n}(1-\delta_i)\right) \times 100\% \quad (2)$$

With description:
- n: the total number of data points or predictions.
- δi: the correctness indicator of the iii-th prediction, defined as:

$$\delta\text{i} = \begin{cases} 1, \text{if the i} - \text{th prediction is correct} \\ 0, \text{if the i} - \text{th prediction is incorrect} \end{cases}$$

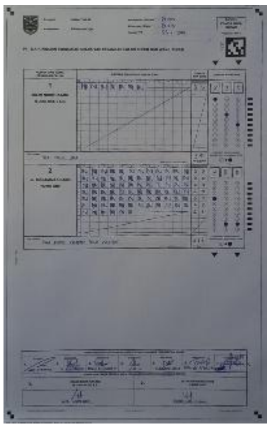
In this article, ballot images are processed to generate numerical data based on the coordinates of specific regions within the image. Thresholding techniques and search algorithms are used to determine the vote coordinates based on the position of the punched marks. In this study, 47 sample ballot images were used to perform automatic ballot reading.

Research Methods

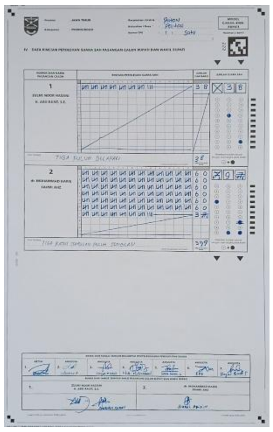
In this study, a digital image processing approach is used to detect valid votes on ballot images. The stages of the research are as follows:

1. Image Pre-processing

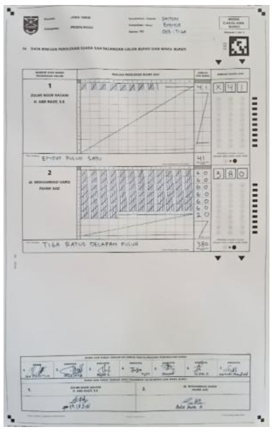
Before the image is processed using the Octave application, it is first converted into a digital format (such as JPG, PNG, and so on). All images are converted to the same format to facilitate the image reading process. Next, the *image* package in GNU Octave is activated to provide image processing functions, after which the image is read using the `imread` function, which loads the pixel data in matrix form for further processing.



Picture 1. Original Image



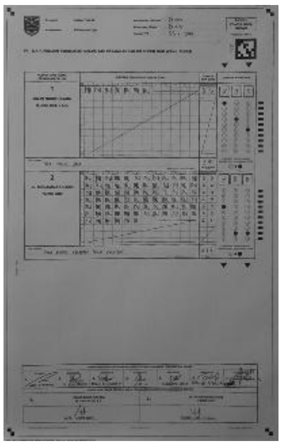
Picture 2. The Original image is problematic



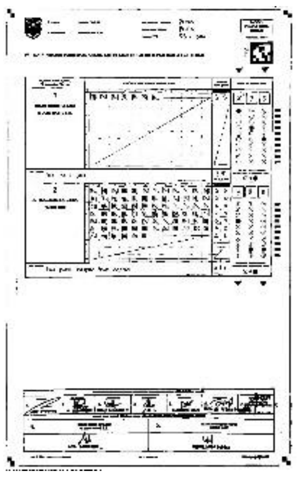
Picture 3. The Original Image ia Problematic

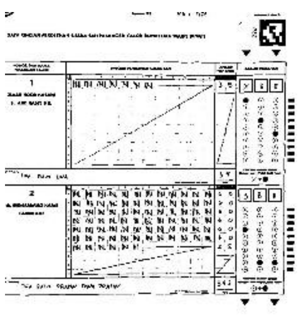
2. *Grayscale Image and Thresholding*

The RGB image is converted to a grayscale image using the `rgb2gray` function, which transforms the three-dimensional color representation into a two-dimensional format that represents only variations in gray intensity. This conversion allows subsequent processes to be performed more quickly. Next, thresholding is applied to convert the grayscale image into a binary (black-and-white) image, in order to separate the objects from the background in the ballot image. The threshold value used in this study is 100, determined after several trials to ensure the consistency of the thresholding results. After this process, an initial image cropping is performed by removing specific portions of rows and columns (such as margins or empty areas) to eliminate irrelevant parts, based on test results from several image samples.



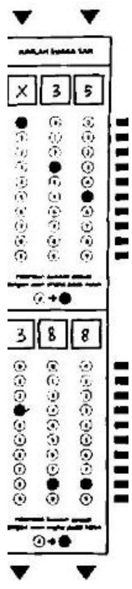
Picture 4. Grayscale Image Results


Picture 5. Thresholding Results


Picture 6. Initial Cropping

3. *Black Pixel Search Algorithm*

The binary image that has undergone the initial cropping process is then subjected to black pixel detection using a linear search algorithm on the two-dimensional binary image (black and white). This process involves examining each element (pixel) one by one, starting from the bottom-right coordinate and moving toward the top-left, until a black pixel with a value of 0 is found. Once the black pixel is detected, a second cropping is performed, retaining only the portion of the image containing the main object for the subsequent image processing stages.


Picture 7. Second Cropping

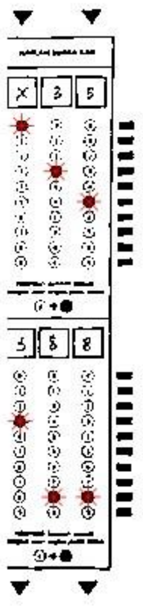
4. *Image Segmentation and Labeling*

Segmentation at this stage aims to locate the coordinates of the numerical markers by detecting black pixel values (0). This is done by extracting all rows at column position *(last – 10)* from the binary image obtained after the second cropping process. The `diff` function is then used to find the difference in coordinates between black pixels (0) along these rows. If a significant gap between rows is detected, it is assumed that there is a separation between groups. This separation is carried out using the `find(diff_idx > 1)` function, which produces the start and end positions of each group. Since the ballot contains two candidate pairs (*paslon*), the data is divided into two groups: group 1–10 representing candidate pair 1, and group 11 to the end representing candidate pair 2.

After segmentation, labeling is performed, in which labels are assigned to the centroids of the detected objects. Labeling begins by determining the centroid matrix indices based on the following criteria:

1. **img**: binary image (0 = black, 1 = white)
2. **min_area**: the minimum pixel count for a group to be considered valid, set to 80 pixels based on tests with several image samples. This avoids detecting small noise while still identifying valid centroids.
3. **start** and **end**: boundary coordinates (x or y) to determine whether a centroid falls within the assumed valid range.

Detected centroids are then labeled using the `bwlabel` function. The `regionprops` function is used to obtain the properties of each connected region. To display the results, the `hold on` function is applied so that new elements can be added without removing the existing ones. An array is created to store the centroid data. Each labeled region is examined in a loop, and a centroid is mapped if its area is greater than or equal to `min_area`. If the centroid lies within the defined range, it is stored and displayed using a red asterisk (`'*'`). Finally, the `hold on` function is closed with `hold off`.



Picture 8. Detection Results

5. Data Extraction and Conversion

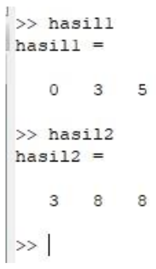
In this stage, object detection is performed on the image obtained from the second cropping process by using the minimum pixel count within the area from the start point to the end point to limit the detection region. The result is the centroid coordinates of each recognized object. The purpose of this step is to ensure that only relevant areas are processed.

The obtained centroids are then sorted based on their x-axis coordinates using the `sortrows` function so that the positions of the centroids can be identified from left to right according to the ballot layout. Once sorted, the centroids are divided into two groups—centroid 1 and centroid 2—corresponding to the number of candidate pairs (*paslon*) in the ballot image.

Next, an empty array is created to store the final results in the form of group numbers. Each centroid is grouped based on its y-coordinate, aligning each centroid with the nearest group, namely `akhir1` or `akhir2`. The centroids are then sorted into a numerical array representing the sequential vote counts.

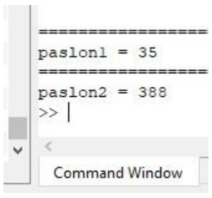
Following this, a data conversion process is carried out. The numerical array is converted into a string using the `sprintf` function. The string value is then converted into a numeric form using the `str2num` function to facilitate the counting process. To display the final result, the `fprintf` function is used.

Below is the resulting numerical array along with the final output from this process:



```
>> hasil1
hasil1 =

   0   3   5

>> hasil2
hasil2 =

   3   8   8

>> |
```

Picture 9. Numerical Arrays



```
=================
paslon1 = 35
=================
paslon2 = 388
>> |
```

Picture 10. Final Result in Numerical Form

Results and Discussion

In this study, experiments were conducted on 47 sample images. The testing was carried out through an iterative looping process on all images, where each image had been processed using the methods described previously. Each image was processed automatically to obtain the vote results from each ballot image.

This study employed a digital image processing approach, which included converting the image to grayscale, applying thresholding to obtain a binary image, cropping relevant parts of the image, and detecting the centroids of black objects that represent numbers.

Below are the ballot reading results from several ballot images:

Table 1. Multiple Image Detection Results of Ballot Papers

Image No	Candidate Pair Results 1	Candidate Pair Results 2	Information
1	28	385	Valid
2	38	3979	Image Problem
3	35	388	Valid
4	15	352	Valid
5	6	415	Valid
6	5	450	Valid
7	17	464	Valid
8	30	399	Valid
9	38	441	Valid
10	-	-	Image Problem
11	22	369	Valid
12	20	362	Valid
13	35	352	Valid
14	132	473	Invalid
15	36	331	Valid
16	110	270	Valid
17	39	362	Valid
18	30	3702	Image Problem
19	5	247	Valid
20	24	408	Valid

21	-	-	Invalid
22	24	367	Valid
23	24	338	Valid
24	32	378	Valid
25	36	383	Valid
26	64	339	Valid
27	35	319	Valid
28	17	320	Valid
29	31	-	Invalid
30	15	433	Valid
31	45	445	Invalid
32	5	333	Valid
33	-	-	Image Problem
34	15	355	Valid
35	30	252	Valid
36	177	294	Valid
37	220	288	Valid
38	339	381	Image Problem
39	263	249	Valid
40	121	364	Valid
41	104	302	Valid
42	84	269	Valid
43	84	418	Valid
44	60	412	Valid
45	33	419	Valid
46	40	4141	Invalid
47	25	409	Valid

The reading results for the ballot images are shown in Table 1. Out of a total of 47 images, most were successfully processed, achieving an overall accuracy of 78.7% and an overall error rate of 21.2%. However, several images failed in the centroid detection process. This was caused by several factors, such as uneven lighting intensity in the image, tilted images, ballots with more than three markings for one candidate pair, and ballots without any markings. These conditions led to certain images being analyzed inaccurately.

Table 2. Confusion Matrix from the Ballot Image Detection Results Table

Actual \ Predicted	Valid	Invalid	Image Problem
Valid	37	0	0
Invalid	0	5	0
Image Problem	0	0	5

This confusion matrix was generated from the ballot image detection results table with three categories: Valid, Invalid, and Image Problem, based on the comparison between the ground-truth labels and the system's classification results. The accuracy rate of 78.7% was obtained from the overall success rate of the system in reading ballot images. However, not all images classified as valid had their numerical vote counts correctly read. The remaining 21.2% represents errors caused by failures during the reading or data processing stages.

The labeling and centroid extraction process in GNU Octave was able to detect the selected objects accurately, provided that the minimum area parameter was appropriately chosen. In this study, a minimum area of 80 pixels was considered suitable to avoid detecting noise or small irrelevant objects.

The row segmentation process based on the position of black pixels provided sufficient results to separate the vote groups for the first and second candidate pairs. The extraction and data conversion process proved useful for displaying the results of ballot readings that had been labeled for valid votes.

However, this study identified several issues that need to be noted:

Problematic images caused centroids to be undetected, resulting in inaccurate outcomes.

The fixed threshold value (100) for binary conversion was sometimes unsuitable for all images, especially when there were differences in lighting conditions.

Dependence on object position, since cropping was based on the last black pixel at the bottom-right corner; if the number position differed, parts could be cropped out or go undetected.

Conclusion

This study demonstrates that digital image processing techniques can be utilized to automatically read and convert data from ballot images into numerical information. The process was carried out in GNU Octave through several stages, namely grayscale conversion, thresholding, segmentation, object labeling, centroid detection, as well as data extraction and conversion. Based on the 47 sample images tested, this approach was able to achieve an accuracy rate of 78.7%.

However, several challenges remain, such as poor image quality, misalignment of object positions, limitations of using a fixed threshold value in thresholding, and the presence of some ballot images without valid vote markings, all of which affect the success of centroid detection.

Future research is expected to improve the accuracy achieved in this study by implementing gradient-based methods for skewed images to obtain better detection results. Improvements to certain parts of the code, such as the application of adaptive thresholding to address variations in lighting, are also recommended. With these enhancements, the method can be further optimized to become more efficient and applicable in real-world implementations.

References

[1] E. F. Agyemang, E. N. N. Nortey, R. Minkah, and K. Asah-Asante, "Baseline comparative analysis and review of election forensics: Application to Ghana's 2012 and 2020 presidential elections," *Heliyon*, vol. 9, no. 8, p. e18276, Aug. 2023, doi: 10.1016/j.heliyon.2023.e18276.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," May 09, 2016, *arXiv*: arXiv:1506.02640. doi: 10.48550/arXiv.1506.02640.

[3] T. Ji, E. Kim, R. Srikantan, A. Tsai, A. Cordero, and D. Wagner, "An Analysis of Write-in Marks on Optical Scan Ballots".

[4] R. C. Gonzalez and R. E. Woods, *Digital Image Processing, Third Edition*, 3rd ed., vol. 14. 2008. Accessed: May 03, 2025. [Online]. Available: http://biomedicaloptics.spiedigitallibrary.org/article.aspx?doi=10.1117/1.3115362

[5] J. Valente, J. António, C. Mora, and S. Jardim, "Developments in Image Processing Using Deep Learning and Reinforcement Learning," *J. Imaging*, vol. 9, no. 10, p. 207, Sept. 2023, doi: 10.3390/jimaging9100207.

[6] V. M. Mohan, R. Kanaka Durga, S. Devathi, and K. Srujan Raju, "Image Processing Representation Using Binary Image; Grayscale, Color Image, and Histogram," in *Proceedings of the Second International Conference on Computer and Communication Technologies*, vol. 381, S. C. Satapathy, K. S. Raju, J. K. Mandal, and V. Bhateja, Eds., in Advances in Intelligent Systems and Computing, vol. 381. , 2016, pp. 353–361. doi: 10.1007/978-81-322-2526-3_37.

[7] R. Dijaya, *Buku Ajar Pengolahan Citra Digital*. Umsida Press, 2023. doi: 10.21070/2023/978-623-464-075-5.

[8] R. C. Gonzalez and R. E. Woods, *Digital image processing*, Fourth edition, Global edition. New York: Pearson, 2017.

[9] V. P.Parmar and C. Kumbharana, "Comparing Linear Search and Binary Search Algorithms to Search an Element from a Linear List Implemented through Static Array, Dynamic Array and Linked List," *Int. J. Comput. Appl.*, vol. 121, no. 3, pp. 13–17, July 2015, doi: 10.5120/21519-4495.

[10] R. Guha, "GRID SEARCHING Novel way of Searching 2D Array," *Int. J. Comput. Appl. Technol. Res.*, vol. 5, no. 1, pp. 26–33, Jan. 2016, doi: 10.7753/IJCATR0501.1005.

[11] D. Amato, G. Lo Bosco, and R. Giancarlo, "Standard versus uniform binary search and their variants in learned static indexing: The case of the searching on sorted data benchmarking software platform," *Softw. Pract. Exp.*, vol. 53, no. 2, pp. 318–346, Feb. 2023, doi: 10.1002/spe.3150.

[12] M. Arzt *et al.*, "LABKIT: Labeling and Segmentation Toolkit for Big Image Data," *Front. Comput. Sci.*, vol. 4, p. 777728, Feb. 2022, doi: 10.3389/fcomp.2022.777728.

[13] M. Ameur, M. Habba, and Y. Jabrane, "A comparative study of nature inspired optimization algorithms on multilevel thresholding image segmentation," *Multimed. Tools Appl.*, vol. 78, no. 24, pp. 34353–34372, Dec. 2019, doi: 10.1007/s11042-019-08133-8.

[14] C. Sager, C. Janiesch, and P. Zschech, "A survey of image labelling for computer vision applications," *J. Bus. Anal.*, vol. 4, no. 2, pp. 91–110, July 2021, doi: 10.1080/2573234X.2021.1908861.

[15] U. Upadhyay and S. Gupta, "A Survey on Image Feature Extraction Techniques," *Int. J. Sci. Res.*, vol. 10, no. 3, 2024.

[16] M. Isik, "Comprehensive empirical evaluation of feature extractors in computer vision," *PeerJ Comput. Sci.*, vol. 10, p. e2415, Nov. 2024, doi: 10.7717/peerj-cs.2415.

[17] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using Mathlab*, vol. 57. Tom Robbins, 1991. Accessed: May 15, 2025. [Online]. Available: https://journals.sagepub.com/doi/10.1177/003754979105700407

[18] S. Lockhart and E. Tilleson, *An Engineer's Introduction to Programming with MATLAB 2017*. SDC Publications, 2017. [Online]. Available: https://www.sdcpublications.com/Textbooks/Engineers-Introduction-Programming-MATLAB-2017/ISBN/978-1-63057-125-2/

[19] R. Singh and S. Gulwani, "Learning Semantic String Transformations from Examples," presented at the Proceedings of the VLDB Endowment, VLDB Endowment, Apr. 2012, pp. 740–751. Accessed: June 16, 2025. [Online]. Available: http://arxiv.org/abs/1204.6079

[20] P. Cerda and G. Varoquaux, "Encoding high-cardinality string categorical variables," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 3, pp. 1164–1176, Mar. 2022, doi: 10.1109/TKDE.2020.2992529.

[21] I. P. Mulya, I. M. Murjana, and Irianto, "Analisis Perbandingan dan Tingkat Akurasi Metode Altman Z-Score, Zmijewski, Springate dan Grover dalam Memprediksi Kebangkrutan Perusahaan," *J. Ilm. Manaj.*, vol. 3, no. 1, p. 11, Oct. 2024.