COMPILER
Vol. 10, No. 2, November 2021, pp. 51-62
ISSN: 2252-3839 (Print)-2549 2403 (On Line)
Accredited Fourth Grade by Kemenristekdikti, Decree No:200/M/KPT/2020
DOI: 10.28989/compiler.v10i2.978

Detection of fake shallots using website-based haar-like features algorithm

Bambang Agus Setyawan[1,*], Mutaqin Akbar[2]
[1,2] Department of Informatics, Universitas Mercu Buana Yogyakarta, Indonesia

Article Info

Article history:

Received June 14, 2021
Accepted July 5, 2021
Published November 30, 2021

Kata Kunci:

Fake Shallots
Fake Shallots Detection
Haar–Like Feature
Object Recognition

ABSTRAK

Shallots is commonly used as essential cooking spices or complement seasoning. The high market demand for this commodity has triggered some people to counterfeit it. They mix the shallots with defective products of onions to get more benefits. It urges to provide a system that can help people to distinguish whether the shallot is original or fake. This research aims to provides an object recognition system for fake shallots utilizing the Haar-Like Feature algorithm. It used the cascade training data set of 59 positive images and 150 negative images with 50 comparison images. The identification process of the shallots was through the haar-cascade process, integrated image, adaptive boosting, cascade classifier, and local binary pattern histogram. This system was made based on the Django website using the python programming language. The test was conducted 30 times on Brebes shallots mixed with Mumbai's mini onions in a single and mixture test method. The test obtained an average percentage of 69.2% for the object recognition of Mumbai's mini onions.

Corresponding Author:

Bambang Agus Setyawan,
Department of Informatics,
Universitas Mercu Buana Yogyakarta,
Soropadan, Condongcatur, Depok, Sleman, Yogyakarta, Indonesia.
Email: *setyawanbambang27@gmail.com

1. INTRODUCTION

Around 2018, markets in most of the traditional market areas of Java Island received an invasion of fake shallots. This shallot comes from India, which entered Indonesia illegally. This fake shallot is actually onions that is the same size of shallots. The difference between mini onions and shallots is in the slices. Onions do not have child part in the onion, it differs with the common shallots, which have onion tillers. The size of Mumbay's onions that are allowed to be imported in is above 5 cm. From the results of an investigation by the Indonesia's Ministry of Agriculture, it was found that at least 70 percent of onions are in each sack of shallots. This triggers opportunities for fraud by mixing onions that are not fit for circulation with other shallots and then being faked as shallots. This case comes in the price of local shallots dropping drastically [1].

Current technological developments in artificial intelligence (AI) one of which is called image processing or commonly called Computer Vision (CV). Computer Vision is a discipline that studies the process of extract special features of objects contained in an image or recognizing objects in images and translating them into information[2]. With this CV, the consumers or even ordinary people can get a useful information system to provide information whether the shallots to be purchased are genuine or not.

Many studies have been carried out to recognize an object, including research to identify shallots and onions. The method used is the feature extraction of red, green, blue colors on objects with a neural network and a Gabor filter applied to the Matlab application. The results of testing on 100 test images can be recognized correctly [3]. Apart from the neural network, a similar algorithm is a Haar-Like algorithm, and this algorithm can be used to recognize objects. The advantage of this algorithm is that it can perform the recognition process in real-time [4]. So that object detection can be done directly without having to take pictures first. Several studies on the application of the Haar-Like algorithm to detect objects are real-time motor vehicle object recognition [5]. The results obtained in the test have a percentage above 70% for images of motorcycles, cars, and buses. The speed level of the Haar-Like algorithm reading process can also be shown in research calculating the speed limit of motorized vehicles [6]. In addition, implementing the Haar-Like method with other methods is also possible, including research that combines the Haar-Like Method with the Local Binary Pattern Histogram (LBPH) to recognize a person's face [7]. The recognition process can be done quickly because all input images are converted to gray images before processing. Thus, the object extraction process is only one layer, which shortens the process [8].

Based on previous studies and literature it showed that the Haar-Like algorithm has a good and fast object reading rate. In addition, the recognition process can be done in real-time. So, the method is very suitable to solve the problem of the rise of fake onions circulating in the market. The references from both journals and other relevant media. Until now, no research has been found to distinguish mini onions that are very similar to red onions with real-time, website-based cameras. Moreover, the output results immediately appear in the camera frame at that time. Referring back to the previous research reference, the research which is to detect mini onions by combining the Haar-Like algorithm with the Local Binary Pattern histogram method and carry out in real-time is possible to do. And it is expected to have a reliable performance in recognizing fake shallot, which is very similar to shallots.

2. RESEARCH METHOD

The research was conducted with some steps, which were requirements identification, knowledge representation, implementation knowledge, and the last step was testing and evaluation. As shown in the Figure 1.

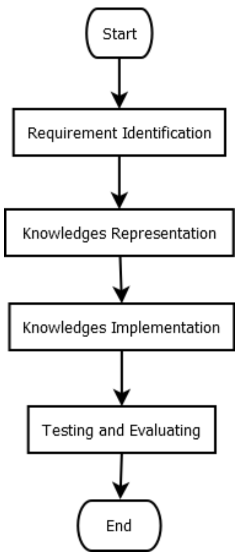

Start
Requirement Identification
Knowledges Representation
Knowledges Implementation
Testing and Evaluating
End

Figure 1 Research Flow

The stage of requirement identification contains the aspects needed in the system before the system can execute the processes. In this case, the requirements identification is built into three aspects: input requirements, process requirements, and output requirements. Input requirements are something needed by the system to process the tasks. Input requirements include positive images, negative images, and cascade datasets. A positive image is an image that contains a fake onion image. A negative image is an image that does not contain an onion image at all.

The onions that will be used as positive images are dark red mini onions bought from the market or store. Then chose the one that was as close as possible to the shallots. The diameter of the selected onion was 2 cm – 4 cm. After that, choose mini onions that are very similar to shallots, both in color and shape, then separate them. Then we took a photo of each mini onion with one onion in at least four different positions. The distance between the camera and the object is 15 cm, and the shooting time is during the day at 10 am – 1 pm. The camera resolution used is 12 Mega Pixels, with bright day conditions and not cloudy. The photos are then put together in a folder.

As for negative images, it comes from photos of all things in the around places. Try not to use photos of objects that contain colors or shapes resembling onions which makes training data errors. The camera resolution for taking negative images is still the same, namely 12 Mega Pixels. Then the photos are put together in a separate folder with the positive images folder.

The positive image used in this stage is 59, and the negative image used is 150 images. This image yet meets the requirements as an input image, and it is necessary to do some editing on it. The purpose of this editing process is to make the process of taking features from the image can be better, and the process is faster without changing the input. The stages in this editing process include the editing process to remove the background, the editing process to change the image pixel size, and the editing process for the cutting process is focusing on the ROI of the object only. It is intended that the feature extraction process can focus more on the object only. There is no interference from the background or other parts of the image.

The process of removing the image background and the editing process of cropping the object ROI is implemented only on positive images, not negative images. However, the editing process to change the pixel value of the image is implemented on the two input images. The pixel value used in this process is 300x400 px. Then all negative and positive images will experience this process. The input requirements are ready at this stage, and the feature data collection process can be implemented.

The initial stage in making this dataset is to convert the color of all existing images into a grayscale image. The purpose of this conversion is to make it easier to capture image features with only one layer, different from a color format that has at least three channels. In addition, the time required to implement the process also becomes shorter. Furthermore, all existing images, both negative and positive, are executed by a haar-like process. The process of making this cascade dataset is as shown in the flowchart (Figure 2).

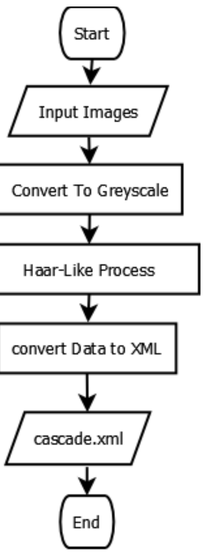


Figure 2 Flowchart cascade.xml

In the stage of haar-like process, it will be processed using four steps, namely haar-like features, integral images, adaptive boosting, and cascade classifier. As shown by the Figure 3 [9]:

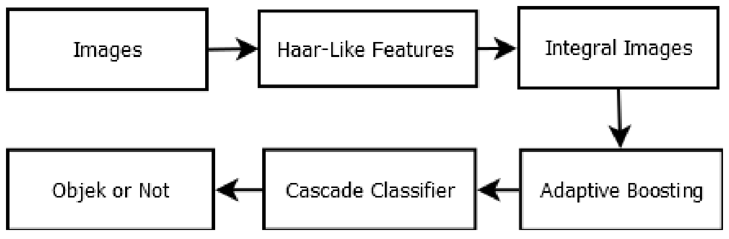


Figure 3 Process of Cascade

The haar-like feature is the process of extracting lines and edges in the image. All existing pictures will be convoluted with haar-like features. The size used in this research is 20x20, so that the convolution process will move from the top left to the right with a box size of 20x20 pixels. This picture below is the haar-like feature used in this process [8]:

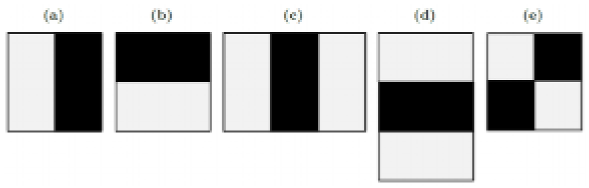


Figure 4 Haar-Like Features

After the haar-like feature process, the next step is the integral image process to the images. Integration means adding small units together. In a small unit image that is added is the pixel value, the integral value for each is the sum of all pixels from the top left corner to the bottom right [8]. The picture below is a flowchart of the integral image process:

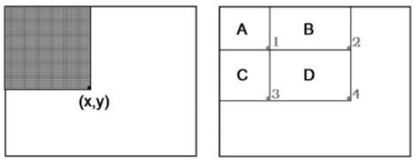


Figure 5 Integral Images Concept

Based on the image above, the left image is an integrated value on the pixel axis (x, y) containing the sum of all pixels in the rectangular area starting from the top left to the location of the axis (x, y). To get the pixel value in the shading area, it is done by dividing the value of the axis (x, y) in the shading area [10].

The next step is the adaptive boosting process (AdaBoost). Boosting assumes the availability of a weak learning algorithm, which is given a labeled training example, resulting in a basic or weak classifier. The purpose of boosting is to improve the performance of a weak learning algorithm while treating it as a "black box" that can be called repeatedly, like a subroutine, but whose interior cannot be observed or manipulated. From this, what can be assumed is that a weak classifier is not completely meaningless in the sense that the error rate is at least slightly better than the random classifier [11].

The Cascade Classifier stage is an image classification process based on the number of features that match a given filter and classified whether it includes objects or not. When the compared image values have a similar pattern, the same pattern area will be cascading. In general, this process is the creation of an area box which will later be processed for the recognition and labeling process [12]. After all the stages are passed, the results of the data obtained will be converted into a dataset with XML format. Then we will immediately test the finished cascade data to see the level of performance in recognizing fake shallot objects. The picture below is the flowchart of testing the dataset cascade.xml that have been create before:

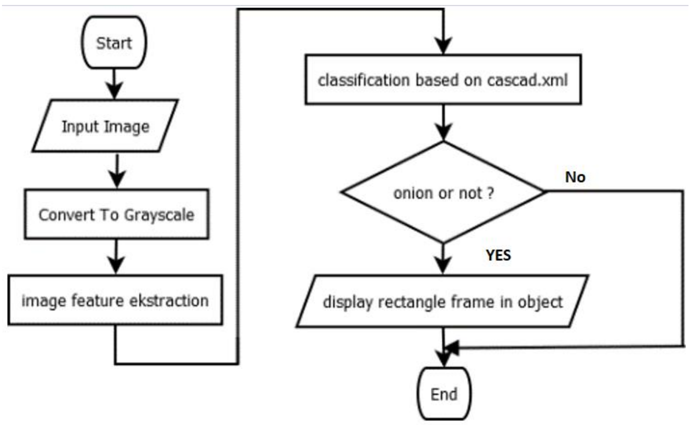

Start
Input Image
Convert To Grayscale
image feature ekstraction
classification based on cascad.xml
onion or not ?
No
YES
display rectangle frame in object
End

Figure 6 Testing Cascade Flowchart

Several parameters were used to test this dataset, including the distance between the camera and the object is about 15-30 cm. The camera used in this test is different. The camera used in this testing phase has a resolution of 5 MP and video Resolution: 1920 x 1080. The position of the object's surface is as clear as possible without being blocked by other objects or covered by more than 1/3 of the part. Tests are carried out with adequate lighting or at least during the day or if at night try to have sufficient lighting.

After the identification stage is complete, the next stage is the stage of identifying process requirements. In this stage, a follow-up process was carried out from the input requirements stage. In this stage, the dataset cascade was created but with a different value. In this case, the image data that is processed is only positive images. The images size is 300x400 px, same as with the positive images used in the requirement process to make file cascade.xml. the compared images in this stage using 50 images with the different position object at all. The extraction process refers to the previous cascade.xml file. The area considered a fake onion by cascade.xml, the histogram value of the image will be calculated. This process is called the local binary pattern histogram (LBPH) process. The process also calculates the image histogram value as a feature compared with the image list features in the haar-like feature. The main character of object recognition with this method is the composition of the micro-texture-pattern, which is a nonparametric operator that describes the local spatial layout of the image [7]. The data obtained will be converted into a file with an XML extension with the name training.xml. as described in the flow of steps below:

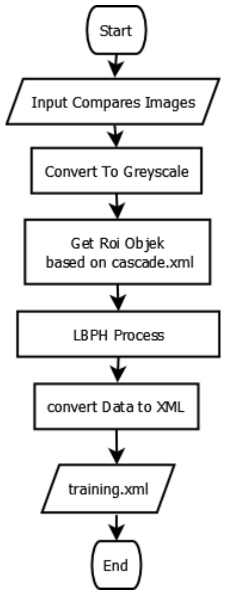


Figure 7 Flowchart training.xml

Identification of output requirements is the need to process various things made in the input requirements and process requirements. The things that have been made in the previous process include the cascade.xml dataset and the training.xml dataset, which will then be processed and extracted to produce output. The steps performed here are as described in the following flowchart:

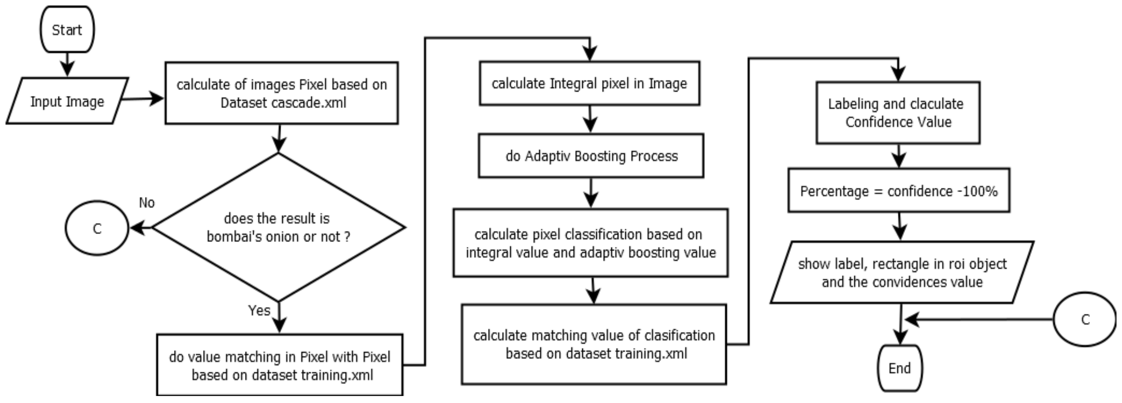

Start
Input Image
calculate of images Pixel based on Dataset cascade.xml
does the result is bombai's onion or not ?
No
C
Yes
do value matching in Pixel with Pixel based on dataset training.xml
calculate Integral pixel in Image
do Adaptiv Boosting Process
calculate pixel classification based on integral value and adaptiv boosting value
calculate matching value of clasification based on dataset training.xml
Labeling and claculate Confidence Value
Percentage = confidence -100%
show label, rectangle in roi object and the convidences value
C
End

Figure 8 Recognition Shallots Recognition

Several parameters were used to recognition system test, including the distance between the camera and the object is about 15-30 cm. The camera used in this test is different. The camera used in this testing phase has a resolution of 5 MP and video Resolution: 1920 x 1080. The position of the object's surface is as clear as possible without being blocked by other objects or covered by more than 1/3 of the part. Tests are carried out with adequate lighting or at least during the day or if at night try to have sufficient lighting.

3. RESULT AND DISCUSSION

This part is the result of implementing the system based on the design that has been made previously. The input data discussed earlier are negative and positive images. The following displays the need for input images that have undergone a repair process as described earlier on the left are positive images. As shown in Figure 9.

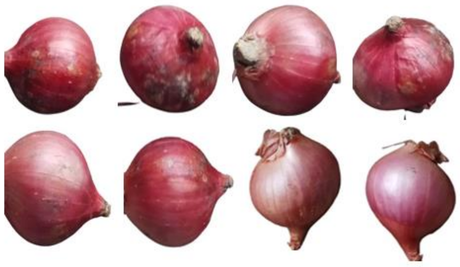
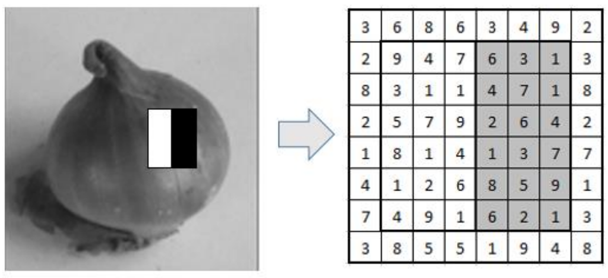


Figure 9 Positive Images

The pictures above are some examples of positive images that have been done in the editing process, where the steps taken are removing the background and cutting the object's ROI. After that, change the image size to 300x400 px. All positive images that are ready will be collected in one folder. It should be noted that the more variations in the position and shape of the fake onion image, the better the reading will be. After the preparation step of the positive image is ready, the next step is to prepare the negative image as training material. Several stages of editing the negative image are also needed, but not as much as the positive image. The following is a display of negative images that have been done a repair process:

Figure 10 Negative Images

The picture above is a sample display of negative images that have been done in the editing process. The editing process that occurs is resizing the image to a size of 300 x 400 px, the same size as the positive image below. After the two input requirements are ready, the next step is to perform the feature extraction process from the two input images. Before the data is processed, all negative and positive images are converted to black and white. After that, the feature-taking process can be implemented on the amber object. The steps take the dark-light feature values with haar-like features from the top left to the bottom right by matching the values according to the feature masking. As in the figure below:



3	6	8	6	3	4	9	2
2	9	4	7	6	3	1	3
8	3	1	1	4	7	1	8
2	5	7	9	2	6	4	2
1	8	1	4	1	3	7	7
4	1	2	6	8	5	9	1
7	4	9	1	6	2	1	3
3	8	5	5	1	9	4	8

Figure 11 Haar Like Feature Process

Then after we get the value of the masking process with the haar-like feature mask is obtained. The next step is to implement an integral process on that value. This stage aims to implement an integral process on the image pixels by increasing the image value at a certain point with the pixel values on the left and above. After the process of adding pixel values is complete. The next step is to calculate the value of the dark and light areas passed by the masking haar-like feature by adding up the average pixel value of the black area minus the number of average pixel values for the white area to obtain a new value. If the value obtained is far from one, it can be interpreted that the haar-like feature masking is not similar to the character of the image composition. However, if the value is close to one, it can be interpreted that the pixel arrangement is following the haar-like feature masking and will be marked as the part that is close to appropriate.

From the pixel position that has been identified on the stage of the integral image, and then it will be processed with the Adaptive Boosting (AdaBoost). This process is aims to group the characteristic values that have been obtained at the integral stage into several classes according to the data features. Then the class division will be carried out, and each class member will be identified regarding the number of members that do not match. Each member who does not fit the class will be considered a weak member, and each weak member will be added to the new weight. Then do the Adaptive Boosting process again. Up to the smallest error value. As explained in the Figure 12.

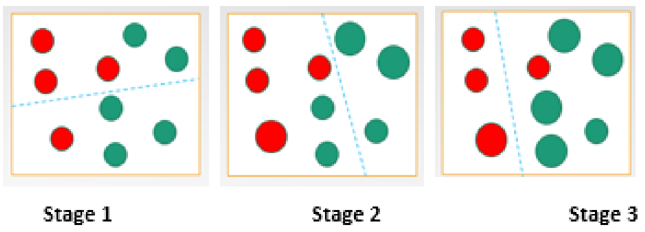
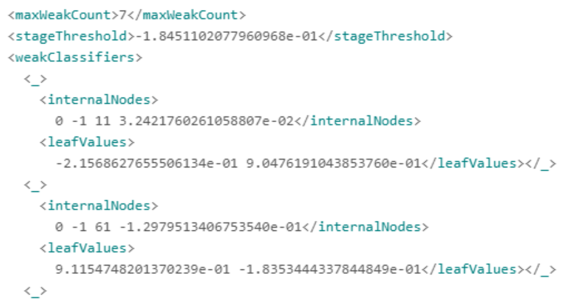


Figure 12 Adaptive Boosting Process

The red color is assumed to be fake shallots, and then the green one is supposed to be another object. The first step is to divide the set into two parts and classify it. Stage 1 was divided as shown in stage 1, and it was found that in the red class 2, classification errors were found, and in the green class 1 misclassification was found. The next stage is stage 2, where the wrong member in each category is added to the new weight, as shown in stage two, and classified into two different classes. And the result is that the red class has two errors, and the green class gets one error so that the wrong member in each category is added to the new weight again. Then at stage 3, the class division is carried out, as shown in the picture stage 3. There is only one error found in the green class, namely one error.

After the smallest error value is obtained, the cascade classifier process can be carried out. This process is classified based on the number of masking haar-like features that match the pixel values in the image. This stage is carried out many times by comparing the haar values in each classification, starting from the smallest match value to the most matching haar value. When each stage is obtained more and more matching values, the category will decide that the object is recognized. If many haar values do not match the pixel character, they can be classified as unrecognized objects.

In that stage, all processes will be carried out to all positive images in the training folder. more input images, make the better the training result. All values that have been obtained in the previous process will then be stored as a dataset. Which will then be converted into a file with an XML extension. At this stage, the requirement to create a cascade dataset has been fulfilled. The process can be carried out by referring to the flowchart that has been shown previously. The following is a snippet of a successful cascade dataset creation. The result of this dataset is named dataset cascade.xml. As explained in the Figure 14.



```
<maxWeakCount>7</maxWeakCount>
<stageThreshold>-1.8451102077960968e-01</stageThreshold>
<weakClassifiers>
  <_>
    <internalNodes>
      0 -1 11 3.2421760261058807e-02</internalNodes>
    <leafValues>
      -2.1568627655506134e-01 9.0476191043853760e-01</leafValues></_>
  <_>
    <internalNodes>
      0 -1 61 -1.2979513406753540e-01</internalNodes>
    <leafValues>
      9.1154748201370239e-01 -1.8353444337844849e-01</leafValues></_>
  <_>
```

Figure 13 Dataset Cascade Value

The image above is a snippet of the negative and positive image extraction results and has been converted into XML format. The existing cascade dataset cannot be implemented into the system to recognize objects before testing. If the test on the fake shallot object has been successfully recognized, the cascade dataset can be implemented. The following are the results of testing the cascade.xml dataset against mini onions or fake onions (Figure 15).

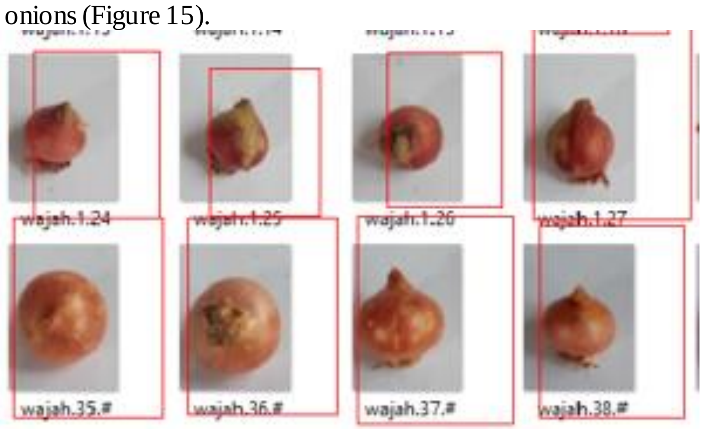
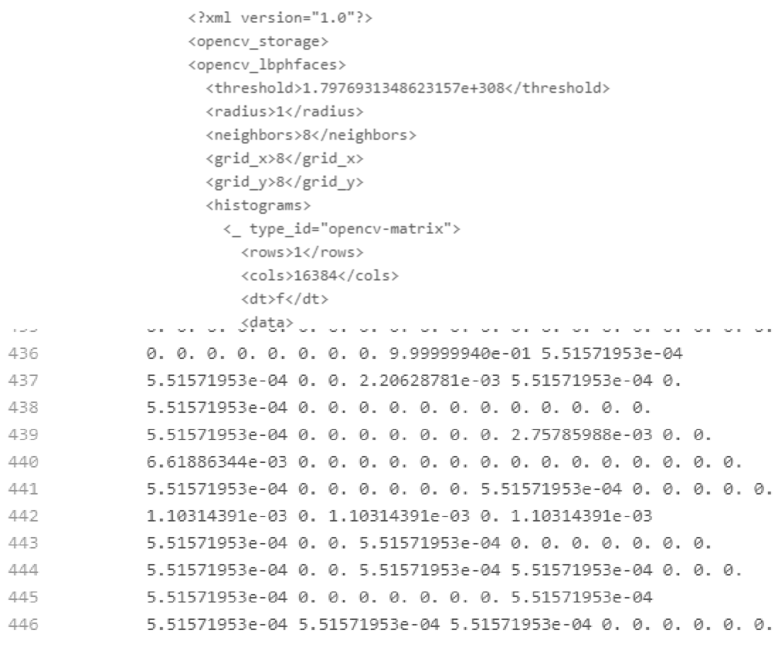


Figure 14 Cascade Result Testing

The display above is the result of testing the cascade.xml dataset against these onions or fake onions. The image used for testing is a photo image, not a video. With the cascade dataset that has been tested and declared to recognize the onion object, the next step is to create a new dataset containing the local binary pattern histogram (LBPH) value with the reference image. The reference image used is 50 images. The histogram values that have been extracted are then converted into an xml file with the name training.xml. The convertion results of dataset training can be seen in the Figure 16.



```
<?xml version="1.0"?>
<opencv_storage>
<opencv_lbphfaces>
  <threshold>1.7976931348623157e+308</threshold>
  <radius>1</radius>
  <neighbors>8</neighbors>
  <grid_x>8</grid_x>
  <grid_y>8</grid_y>
  <histograms>
    <_ type_id="opencv-matrix">
      <rows>1</rows>
      <cols>16384</cols>
      <dt>f</dt>
      <data>
0. 0. 0. 0. 0. 0. 0. 0. 9.99999940e-01 5.51571953e-04
5.51571953e-04 0. 0. 2.20628781e-03 5.51571953e-04 0.
5.51571953e-04 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
5.51571953e-04 0. 0. 0. 0. 0. 0. 2.75785988e-03 0. 0.
6.61886344e-03 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
5.51571953e-04 0. 0. 0. 0. 0. 5.51571953e-04 0. 0. 0. 0. 0.
1.10314391e-03 0. 1.10314391e-03 0. 1.10314391e-03
5.51571953e-04 0. 0. 5.51571953e-04 0. 0. 0. 0. 0. 0. 0.
5.51571953e-04 0. 0. 5.51571953e-04 5.51571953e-04 0. 0. 0.
5.51571953e-04 0. 0. 0. 0. 0. 0. 5.51571953e-04
5.51571953e-04 5.51571953e-04 5.51571953e-04 0. 0. 0. 0. 0. 0.
```

Figure 15 Training Cascade Dataset

After the training file exists, the next step is to implement the output stage. The previously created data cascade, namely cascade.xml and training.xml, will be used to perform the image reading process and calculate the match value based on the histogram value in the reference image with the histogram value input image. The following is the result of displaying the results of reading objects and labeling :

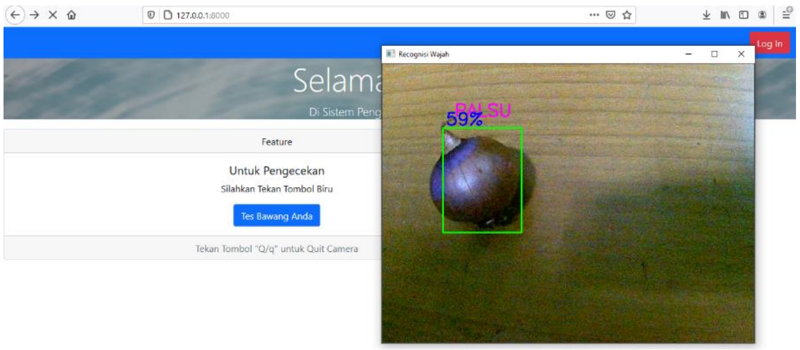


Figure 16 System Testing

From the picture above (Figure 17), it can be seen that the system has worked according to the previously mentioned design. In addition, the system also succeeded in detecting fake onion objects. Then the next step is to test and evaluate the performance of the system. The test was carried out by mixing mini onions with Brebes varieties of shallots. With the sampling method singly or in groups. The test was carried out between 2:00 PM -03:00 PM with a distance of 0-30 cm from the camera to the object. Below is an equation to calculate the percentage of system detection and an equation to calculate the average rate of the system.

$$True\ Percentage = \frac{x}{\sum fake\ shallots}\ X\ 100\ \% \qquad (1)$$

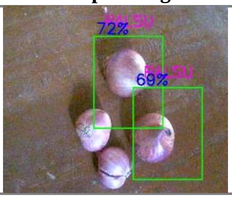
Explanation :

x	= number of fake shallots which are detected by sistem
$\sum Fake\ shallots$	= the number of fake objects should be

$$Average\ Rate\ Procentage = \frac{\sum percentage}{\sum testing\ sample} \qquad (2)$$

The table below is a sample test table accompanied by sample images and recognition result, the percentage value is calculated referring to equation 1 (Table 1).

Table 1 Sample Testing System

No	Sample Image	∑ Fake	∑ Detect	∑ False	% Result
1		2	2	0	$True\ Percentage = \frac{x}{\sum fake\ shallots}\ X\ 100\%$ = 2/2 * 100 % = 100 %

System testing is done by using 30 samples with single objects or groups. Fake onions are fake onions that should be, detect is the number of fake onions detected by the system created, false is the number of fake onions that were not detected or detected incorrectly, and % result is the result percentage calculation as described in equation 1. As has been summarized in the following table (Table 2).

Table 2 Result Testing System

No	Test Sample	∑Fake	∑Detect	∑False	% Result
1	Sample 1	2	2	0	100 %
2	Sample 2	2	2	0	100 %
3	Sample 3	2	2	0	100 %
4	Sample 4	0	0	2	0 %
5	Sample 5	4	3	1	75 %
6	Sample 6	4	3	1	75 %
7	Sample 7	2	7	0	100 %
8	Sample 8	0	1	0	0 %
9	Sample 9	4	4	0	100 %
10	Sample 10	0	2	2	50 %
11	Sample 11	2	1	1	50 %
12	Sample 12	2	2	0	100 %

13	Sample 13	2	0	2	0 %
14	Sample 14	0	0	2	0 %
15	Sample 15	0	0	1	0 %
16	Sample 16	1	1	0	100 %
17	Sample 17	1	1	0	100 %
18	Sample 18	1	1	0	100 %
19	Sample 19	1	1	0	100 %
20	Sample 20	1	1	0	100 %
21	Sample 21	0	0	1	0 %
22	Sample 22	0	0	1	50 %
23	Sample 23	5	4	1	80 %
24	Sample 24	6	6	0	100 %
25	Sample 25	1	1	0	100 %
26	Sample 26	1	0	0	100 %
27	Sample 27	2	2	0	100 %
28	Sample 28	11	9	2	82 %
29	Sample 29	7	2	2	28 %
30	Sample 30	2	2	0	100 %

From the test table above, the average total percentage of object recognition can be found using the equation below:

$$PAverage\ Rate\ Procentage = \frac{\sum percentage}{\sum testing\ sample}$$

$$PAverage\ Rate\ Procentage = (100 + 86 + 100 + 0 + 75 + 75 + 100 + 0 + 100 + 50 + 50 + 100 + 0 + 0 + 0 + 100 + 100 + 100 + 100 + 100 + 0 + 50 + 80 + 100 + 100 + 100 + 100 + 82 + 28 + 100)/30$$

$$PAverage\ Rate\ Procentage = 1976/30 = 69.2\ \%$$

4. CONCLUSION

From the research described above, it can be concluded that the Haar-Like Feature method can be applied as a method to recognize objects. Classification using the Haar-Like method consists of haar-like features, integral images, adaptive boosting, classifiers combined with local binary pattern histogram methods. The results of the feature extraction process from the input data are stored in the form of an XML file. The dataset needed in this method is the object feature dataset and the object histogram pattern dataset. The two datasets are run simultaneously, and after that, they are used to calculate object matches to the training data. The matched data is then used for the labeling process and percent calculation. Testing of objects does not have to be done singly but can also be done in groups as long as the onion's surface is visible and not covered by other objects. From the 30 objects test by testing objects individually or in groups, with a distance between the camera and objects of 15-30 cm, and the testing time between 13:00 - 15:00, the results obtained were 69.2% of objects that were correctly recognized, and as many as six objects or 30.8%. The results of the system are influenced by several factors, including the object's position, the intensity of light, and the distance of the object to the camera. The recognition process with the optimum reading is that the farthest distance of the object is 15 cm and the lighting conditions around the object are bright.

REFFERENCES

[1] H. B. Pratomo, "Pemerintah beri tips kenali bawang merah palsu.," *https://www.merdeka.com*, Jun. 22, 2018. [Online]. Available: https://www.merdeka.com/uang/pemerintah-beri-tips-kenali-bawang-merah-palsu.html

[2] T Sutoyo dkk, *TEORI PENGOLAHAN CITRA DIGITAL*. Yogyakarta: Andi Offset, 2009.

[3] A. Agusriandi, "IdentifikasiBawang Merah dan Bombay dengan PendekatanRadial Basis Function Neural Network (RBFNN)," *JURNAL MEDIA INFORMATIKA BUDIDARMA*, vol. Vol 4 No 4, pp. 1043–1050, Oktober 2020, doi: 10.30865/mib.v4i4.2334.

[4] S. Abidin, "Deteksi Wajah Menggunakan Metode Haar Cascade Classifier Berbasis Webcam Pada Matlab," *ELEKTERIKA*, vol. 15, no. 1, p. 21, May 2018, doi: 10.31963/elekterika.v15i1.2102.

[5] N. D. Mega Anjani, F. Farida, and M. Kurniawan, "ANALISIS FITUR HAAR MENGGUNAKAN ALGORITMA HAAR-LIKE FEATURE PADA CITRA KENDARAAN BERMOTOR," *NERO*, vol. 5, no. 2, p. 124, Dec. 2020, doi: 10.21107/nero.v5i2.187.

[6] M. Zulfikri, E. Yudaningtyas, and R. Rahmadwati, "Sistem Penegakan Speed Bump Berdasarkan Kecepatan Kendaraan yang Diklasifikasikan Haar Cascade Classifier," *Jurnal Teknologi dan Sistem Komputer*, vol. 7, no. 1, pp. 12–18, Jan. 2019, doi: 10.14710/jtsiskom.7.1.2019.12-18.

[7] S. Al-Aidid and D. Pamungkas, "Sistem Pengenalan Wajah dengan Algoritma Haar Cascade dan Local Binary Pattern Histogram," *JRE*, vol. 14, no. 1, pp. 62–67, Apr. 2018, doi: 10.17529/jre.v14i1.9799.

[8] S. Chau, J. Banjarnahor, D. Irfansyah, and S. Kumala, “Analisis Pendeteksian Pola Wajah Menggunakan Metode Haar-Like Feature,” *JITE*, vol. 2, no. 2, p. 69, Jan. 2019, doi: 10.31289/jite.v2i2.2133.

[9] R. E. Putri, T. Matulatan, and N. Hayaty, “Sistem Deteksi Wajah Pada Kamera Realtime dengan menggunakan Metode Viola Jones,” *sustainable*, vol. 8, no. 1, pp. 30–37, May 2019, doi: 10.31629/sustainable.v8i1.526.

[10] D. A. Ayubi, D. A. Prasetya, and I. Mujahidin, “Pendeteksi Wajah Secara Real Time pada 2 Degree of Freedom (DOF) Kepala Robot Menggunakan Deep Integral Image Cascade,” *CYCLOTRON*, vol. 3, no. 1, Feb. 2020, doi: 10.30651/cl.v3i1.4306.

[11] R. E. Schapire and Y. Freund, *Boosting: foundations and algorithms*. Cambridge, MA: MIT Press, 2012.

[12] R. Laganière, *OpenCV 3 computer vision application programming cookbook: recipes to help you build computer vision applications that make the most of the popular C++ library Open CV 3*, Third edition. Birmingham Mumbai: Packt, 2017.