IMPLEMENTATION MARKOV STATIONARY FEATURE – VECTOR QUANTIZATION FOR FACIAL EXPRESSION RECOGNITION

Fascal Sapty Jarockohir[1], Irfan Maliki, S.T., M.T.[2]

[1,2] Program Studi Teknik Informatika
Fakultas Teknik dan Ilmu Komputer Universitas Komputer Indonesia
Jl. Dipatiukur No. 112-114 Bandung
E-mail : fsapty@email.unikom.ac.id[1], irfan.maliki@email.unikom.ac.id[2]

ABSTRAK

Facial Expression is one form of communication, facial expressions have different facial shapes. In the process of recognition of facial expressions, machine learning does not directly recognize expressions, but it is necessary to do the feature extraction process first. One feature extraction method that can be used is Markov Stationary Feature - Vector Quantization (MSF-VQ). The MSF-VQ algorithm has been tested in cases of human face recognition and has an accuracy of 99.16%. In this study, the MSF-VQ algorithm is used to recognize facial expressions. In this study used data consisting of Training Data and Test Data Total data used for the testing process were 1440. For the training data, 1170 face photos were obtained from 15 people, 6 types of expressions, and from one type of expression, there were 13 faces. per person. The test was performed using 270 test data using the MSF-VQ algorithm using a Multiclass Support Vector Machine learning machine with a linear kernel, successfully obtaining an accuracy of 97.41%.

Kata kunci : Facial Expression, Markov Stationary Feature, Vector Quantization, Support Vector Machine, Feature Extraction.

1. PREFACE

Facial expressions are one or more movements or positions of muscles under the facial skin. Facial expression is a very important part of communication. In general, facial expressions have 6 types of expressions, namely: Happines, Sadness, Surprise, Fear, Anger and Disgust [1].

In the same facial expression everyone can have a different face shape, but facial expression has a unique pattern. For the facial expression recognition process, the learning machine must go through the feature extraction process in order to recognize the unique pattern.

Feature extraction is a very important process, this process is used to obtain values that can later be used by learning machines to recognize facial expressions.

Researches on facial expressions include: Saputra [2] using the Local Binary Pattern (LBP) method, obtaining an accuracy of 65.1%, Pengutanutan [3] using the Discrete Cosine Transform (DCT) Algorithm and Principal Component Analysis (PCA) having an accuracy of 50 %, Saumil [4] using Linear Discriminant Analysis (LDA) gets an accuracy of 60%.

Based on the journal Yan etc. [5] The LBP, PCA and LDA methods have their shortcomings: LBP is vulnerable to noise; PCA is not good if the image compared is different in resolution or the brightness level is different; LDA extraction results are not good if the resolution is low. Because of the limitations of the methods used, the accuracy of these studies is still not good.

Markov Stationary Feature - Vector Quantization (MSF-VQ) is a method that is able to overcome the shortcomings of the methods used in previous studies that have been mentioned, this is evidenced by the results obtained accuracy of 99.16% in cases of human face recognition [5] Therefore in this study MSF-VQ will be conducted research to improve accuracy from previous research.

2. CONTENT OF RESEARCH

2.1. Method

The facial expression recognition system that will be built consists of two major processes namely Training and Testing. In the training phase, in general is Pre-processing, Feature Extraction, Training. For the Classification Process Doing every step up to the feature extraction stage and doing the classification process from the data that has been trained in the process can be seen in Figure 1. 1.

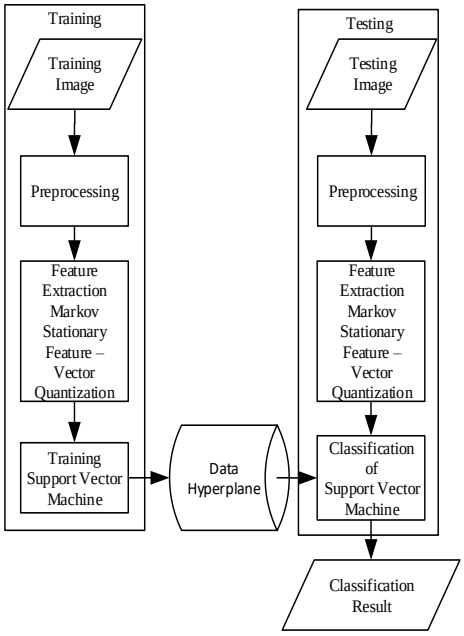


Figure 1. 1. System Process

The General System is divided into two major stages: Training and Classification. The description is as follows :

1. Training Process

   In the training process the training data will go through a preprocessing process, Markov Stationary Feature feature extraction - Vector Quantization, Training Support Vector Machine. From the process, the process is described as follows:

   a. Face Detection is the process of taking part of the face image from the input image.
   b. Low Pass filter is a filtering process of remove image have high intenstiy level.

   After the preprocessing process, feature extraction process is performed using MSF-VQ. The next process is the Training Support Vector Machine process. This process is carried out to obtain hyperplane data from training data that has been entered then the hyperplane data is entered into XML with the provisions of OpenCV.

2. Proses Klasifikasi

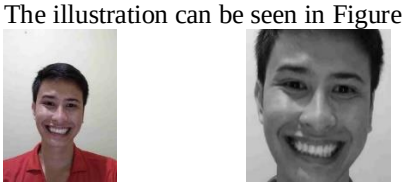
   The Classification Process is a process to identify whether the image processed is in the class Happiness, Sadness, Surprised, Fear, Anger and Disgust. For the same stage as the training process, the process is carried out only until the extraction of features from the hyperplane data that has been saved and then compared to the Multiclass Support Vector Machine Classification Method (M-SVM).

2.1.1. Pre Processing

The Pre Processing process is a step to reduce noise with the aim of helping feature extraction methods to get better accuracy. Preprocessing stages in this study are Face Detection with Haar Cascade method and Lowpass Filter.

1. Face Detection (Viola Jones)

   Face detection is a step to take the image of the face. The illustration can be seen in Figure 1. 2.



Initial Image — Result of Image Detection

Figure 1. 2. Ilustration of Facial Detection

In the face detection process there are several processes carried out, namely: Grayscale Haar-Like Feature, Integral image, Adaboost (Adaptive Boosting), and Cascade Classifier [7]. The steps of the Viola Jones algorithm are:

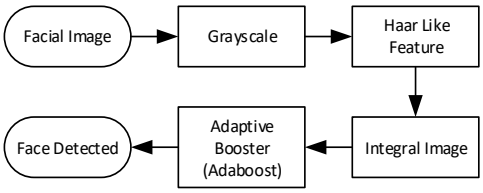


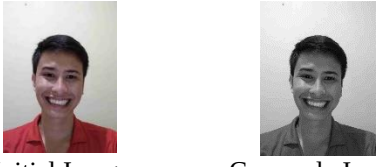
Figure 1. 3. Flow Face Detection Process

a. Grayscale

   Grayscale is a process to change the image into a gray color, calculated by formula [5]. Color has a value of R, G, B, these values are converted into a value called the grayscale value, while the process uses the following equation :

$$Gr = R \times 0.29 + G \times 0.59 + B \times 0.11 \quad (1)$$

   From the equation, the grayscale image is produced as follows:



Initial Image — Grayscale Image

Figure 1. 4. Grayscale Illustration

b. Haar Like Feature

   The technique used is by boxing the image and then dividing into the image of the black area and the white area [7].

Figure 1. 5. Fitur Haar

c. Integral Image

Integral image is used to calculate the sum of pixel values in regions detected by the Haar feature.

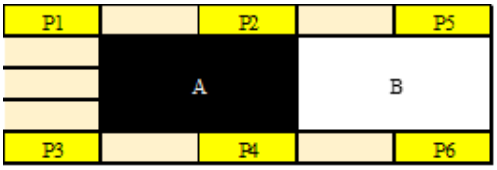


Figure 1. 6. Integral Image

$$h(x) = |((P4 + P1) - (P3 + P2)) - ((P6 + P2) - (P7 + P5))| \quad (2)$$

d. Adaptive Booster

Adaptive booster is the stage to update the weights by implementing the following formula:

m = number of negative images, y = 1, negative images are testing images.

l = number of positive images, y = 0, negative images are training images.

Is Known:

First Weight = $w_{j_{y_i}} = \frac{1}{2m}, w_{t_{y_i}} = \frac{1}{2l}$ (3)

Then implement integral image:

For positive image :

$$h_t(x) = |((P4 + P1) - (P3 + P2)) - ((P6 + P2) - (P7 + P5))|$$

For Negative image :

$$h_j(x) = |((P4 + P1) - (P3 + P2)) - ((P6 + P2) - (P7 + P5))|$$

Find error rate :

Positif image: $\epsilon_t = (\sum_t^T w_{t,i})|h_t(x) - y_i|$ (4)

Negative image : $\epsilon_j = (\sum_j^J w_{j,i})|h_j(x) - y_i|$ (5)

If $\epsilon_t \vee \epsilon_j < 0$, stop iteration

Find new weight:

Positive Weight:

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t} \quad (6)$$

Negative Weight:

$$\beta_j = \frac{\epsilon_j}{1-\epsilon_j} \quad (7)$$

Compare this with the following equation :

$$H(x) = \begin{cases} 1 \ \sum_{j=1}^{J} \alpha_j h_j \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 \qquad not\ face. \end{cases} \quad (8)$$

When :

$$\alpha_j = \log \frac{1}{\beta_j}, \alpha_t = \log \frac{1}{\beta_t} \quad (9)$$

e. Cascade Clasifier

After the features are known, then compare the image with each feature, with the following illustration::

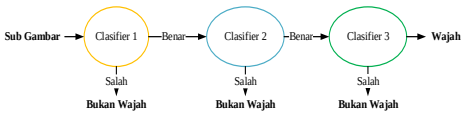


Figure 1. 7. Cascade Clasifier

2. Lowpass Filter

Lowpass Filter or low pass filtering is a process in the image to pass images at low frequencies and will reduce or reject data at high frequencies. For the equations used are as follows: :

$$h(x) = f(x) * g(x) \quad (10)$$

For the calculation process of the equation is illustrated in Figure 1. 8.

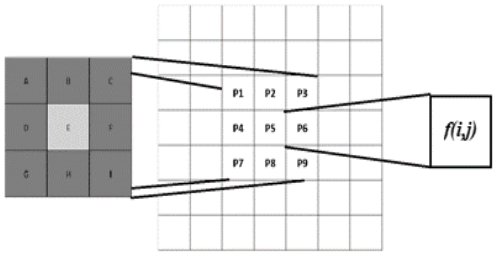


Figure 1. 8. Image Convolution

$$f(i,j) = (A \times P1) + (B \times P2) + (C \times P3) + (D \times P4) + (E \times P5) + (F \times P6) + (G \times P7) + (H \times P8) + (I \times P9) \quad (10)$$

The kernel used for research is the mean filter with the kernel as follows:

$$g = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \quad (11)$$

From the LowPass Filter process the image will be generated as follows:

Image Detection Image Lowpass Filter

Figure 1. 9. Lowpass Filter Result

2.1.2. Ekstraksi Fitur

Feature extraction is a process to get values that are characteristic, and as input from machine learning, in this study feature extraction has 2 phased processes, namely: Vector Quantization and Stationary Feature Markov The stages are as follows :

1. Vector Quantization

Vector Quantization (VQ) is the process of forming compressed images, the way VQ works is to map the codebook [8]. The process of VQ is as follows:

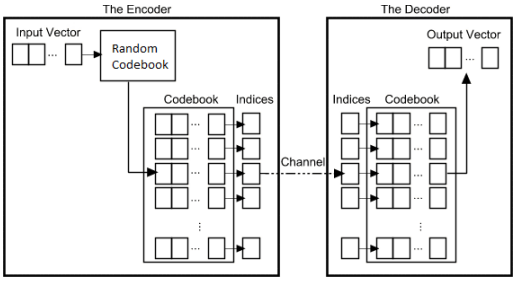


Figure 1. 10. Illustration of Vector Quantization

Initialization :

k = size of block dimension.

b = index block

t = index of codeword members

j = index of codebook, index of codebook

i = index of block

n = codeword length = k x k.

Steps of VQ :

Step 1 : Random codebook initials

$C = \{c_1, c_2, c_3, \dots, c_{Nc}\}$

Value is known $c_1, c_2, c_3, \dots, c_{Nc}$ is a codeword from the codebook, it is known that the codebook that is initialized is as followst :

Codeword	Nilai Codeword			
c1	223	223	223	216
c2	138	135	135	130
c3	191	193	192	186
c4	160	162	163	130

Figure 1. 11. Codebook Initialization

Step 2 : Divide image into blocks

$X = \{x_1, x_2, x_3, \dots, x_{Nb}\}$

Value is known $x_1, x_2, x_3, \dots, x_{Nb}$

is block from image, for block x_1 the value is as follows :

Posisi Pixel	pixel(x,1)	pixel(x,2)
pixel(1,x)	35	27
pixel(2,x)	47	37

Figure 1. 12. Block image

Step 3 : Find index

To find the index find the euclidean distance value between the initials codebook and the image block to find the distance..

$$d(x_b, c_j) = \sqrt{\sum_{t=0}^{n-1}(x_t - c_t)^2} \quad (12)$$

$$d(x_1, c_1) = \sqrt{\begin{matrix}(35-223)^2 + (27-223)^2 \\ +(47-223)^2 + (37-216)^2\end{matrix}}$$

$$= 369.83$$

Kemudian lakukan pada setiap codeword, sehingga dihasilkan nilai :

$d(x_1, c_2) = 196.636$

$d(x_1, c_3) = 308.412$

$d(x_1, c_4) = 236.548$

To find the index value, the following equation is given:

$$indeks \in X: d(x_b, c_{j-1}) < d(x_i, c_j) \quad (13)$$

The equation is intended to find the smallest euclidean distance value. Given the smallest distance value is 196.636 then the value is located in codeword 3, then the index in block 1 is 3. During the process if the index of the block is the same then it contains into the variable x_j and the number of occurrences is calculated and entered into the variable N_j, from the second The variables stored are shaped vectors based on the size of the block..

Step 4 : Update codebook

From the appearance of the index on each block x_j has been accommodated and N_j then do the calculation with the following equation:

$$c_t = \frac{x_t}{N_j} \quad (14)$$

This value is the value of each of these values is the content of the new codebook. After that, find the distortion value by calculating the distance between the new codebook and the block:

$$d(X, C) = \sum_{t=0}^{n-1}(x_t - c_t)^2 \quad (15)$$

After the distortion value is generated, the distortion value is divided by the number of Nb × n.

$$D_m = \frac{d(X,C)}{Nb \times n} \quad (16)$$

After obtaining the distortion value, update the codebook until it meets and stops the codebook update process until it meets the conditions :

$$\frac{D_{m-1} - D_m}{D_m} \leq \varepsilon = 0.001 \quad (17)$$

After the codebook update process is complete, do a mapping of the index values that are accommodated. By taking the values in the updated codebook. So that it forms a new image that has been compressed to produce the following image:

Figure 1. 13. Vector Quantization Image

2. *Markov Stationary Feature*

The Markov Stationary Feature (MSF) is part of the feature, the process from MSF can be seen on Figure 1. 14.

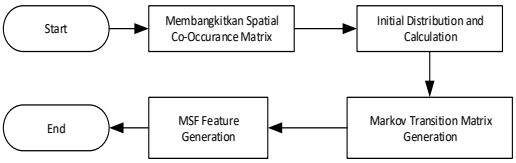


Figure 1. 14. Flow Markot Stationary Feature

The explanation of the Stationary Feature Markov Flow is as follows:

a. Divide images into blocks according to the filter in this implementation, exemplified by using 3x3 filters and can be seen on Figure 1. 15.

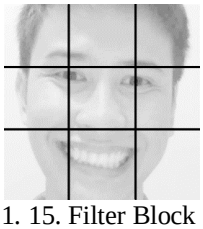


Figure 1. 15. Filter Block 3x3

b. Generate *Co-Occurance Matrix*

Before generating co-occurrence matrix, the image range needs to be changed to 0-32. $Gray\ Color$ is the color value of the image, Graylevel is the maximum value range level, the $Graylevel$ is using 32. Here are the equations used :

$$Warna32 = \frac{Gray\ Color \times Graylevel}{255} \quad (18)$$

Is Known image size is 12x12 :

72	63	63	67	72	63	63	67	72	63	63	67
74	64	63	68	74	64	63	68	74	64	63	68
74	65	63	68	74	65	63	68	74	65	63	68
77	69	68	72	77	69	68	72	77	69	68	72
105	100	99	99	105	100	99	99	105	100	99	99
110	104	104	104	110	104	104	104	110	104	104	104
112	106	107	107	112	106	107	107	112	106	107	107
116	112	111	111	116	112	111	111	116	112	111	111
142	144	145	146	142	144	145	146	142	144	145	146
146	148	150	150	146	148	150	150	146	148	150	150
147	149	150	150	147	149	150	150	147	149	150	150
144	147	147	148	144	147	147	148	144	147	147	148

Figure 1. 16. Image Block for MSF

The calculation is done using equation (18), so as to produce the following image:

9	8	8	8	9	8	8	8	9	8	8	8
9	8	8	9	9	8	8	9	9	8	8	9
9	8	8	9	9	8	8	9	9	8	8	9
10	9	9	9	10	9	9	9	10	9	9	9
13	13	12	12	13	13	12	12	13	13	12	12
14	13	13	13	14	13	13	13	14	13	13	13
14	13	13	13	14	13	13	13	14	13	13	13
15	14	14	14	15	14	14	14	15	14	14	14
18	18	18	18	18	18	18	18	18	18	18	18
18	19	19	19	18	19	19	19	18	19	19	19
18	19	19	19	18	19	19	19	18	19	19	19
18	18	18	19	18	18	18	19	18	18	18	19

Figure 1. 17. Images 32

Find co-occurence matrix using neighbour value of 0^0, 45^0, 90^0, dan 135^0. Illustration can seen in .

9	8	8	...	13
9	8	8	...	14
9	8	8	...	14
...	...	...	...	...
18	18	18	...	21

	6	7	7	9	10
6					
7				3	
7					
9					
10					

Figure 1. 18. Ilustrasi Co-occurance Matrix

Result of Ilustration in is :

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	96	0	0	0	0	0	0	0
2	0	96	318	84	12	2	4	0	0	0
3	0	0	66	444	36	14	12	9	3	4
4	0	0	0	16	336	24	28	0	1	11
5	0	0	0	8	12	582	50	45	7	0
6	0	0	0	8	10	22	704	115	17	46
7	0	0	0	4	6	14	110	518	182	11
8	0	0	0	4	4	12	8	122	737	92
9	0	0	0	0	0	0	0	16	113	671

Figure 1. 19. Co-occurance Matrix

c. Find Initial Distribution

After getting the co-occurrence matrix value, find the initial distribution value as follows :

	0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	0	0	0	0
1	0	0	96	0	0	0	0	0	0	0
2	0	96	318	84	12	2	4	0	0	0
3	0	0	66	444	36	14	12	9	3	4
4	0	0	0	16	336	24	28	0	1	11
5	0	0	0	8	12	582	50	45	7	0
6	0	0	0	8	10	22	704	115	17	46
7	0	0	0	4	6	14	110	518	182	11
8	0	0	0	4	4	12	8	122	737	92
9	0	0	0	0	0	0	0	16	113	671

Figure 1. 20. Sumulation of initial distribution

Perform the following calculations:

$sum\pi = 0 + 0 + 318 + 444 + 336 + 582 + 704 + 737 + 671 = 4310$

$\pi(0) = ($

$0/4130, 0/4130, 318/4130, 444/4130, 336/4130, 582/4130, 704/4130, 518/4130, 737/4130, 671/4130$

)

Then obtained initial distribution::

$\pi(0)$ =(0,0,0.073781903, 0.103016241, 0.077958237,0.135034803,0.163341067,0.120185615,0.17099768,0.155684455)

d. Generate Transition Matrix

Averaging the co-occurrence matrix value based on the value of each row so as to produce the following values: :

0	0	0	0	0	0	0	0	0	0
0	0	0.98969	0	0	0	0	0	0	0
0	0.18533	0.6139	0.16216	0.02317	0.00386	0.00772	0	0	0
0	0	0.11168	0.75127	0.06091	0.02369	0.0203	0.01523	0.00508	0.00677
0	0	0	0.0381	0.8	0.05714	0.06667	0	0.00238	0.02619
0	0	0	0.01128	0.01693	0.82087	0.07052	0.06347	0.00987	0
0	0	0	0.00862	0.01078	0.02371	0.75862	0.12392	0.01832	0.04957
0	0	0	0.00469	0.00704	0.01643	0.12911	0.60798	0.21362	0.01291
0	0	0	0.00405	0.00405	0.01216	0.00811	0.12361	0.74671	0.09321
0	0	0	0	0	0	0	0.01978	0.13968	0.82942

Figure 1. 21. Matrix Transisi

e. Mencari Nilai Stationary Distribution

To find the Stationary Distribution value is calculated using the Markov Chain, by finding the limit value from $A = \lim_{n\to\infty} A_n$, when $A_n = \frac{1}{n+1}(I + P + P^2 + \cdots P^n)$. I an identity matrix. Determined the value of n is 50, the value is obtained from the average of each line $\vec{a_i}$ of A_n. For calculate features, the following equation is used::

$\pi \approx \frac{1}{K}\sum_{i=1}^{K}\vec{\alpha}_{ij}$, when $A_n = [\vec{a_1}, \ldots, \vec{a_n}]^T$. (20)

The result of the calculation is in the form of a vector which will be combined with the initial distribution, the result of the combined initial distribution is used as a feature that will be recognized by the learning machine Multiclass Support Vector Machine.

2.1.1 Multiclass Support Vector Machine

The next step will be learning and testing on training data with histogram test data, using the M-SVM method in the Multiclass Suppport Vector Machine (M-SVM) method. The M-SVM method can only do binary classifications (two classes). The approach used is one-againist-all (OAA). To see the process of M-SVM is illustrated in Figure 1. 21.

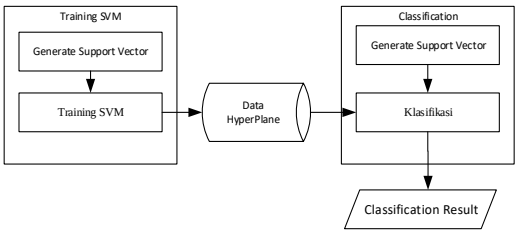


Figure 1. 22. Multiclass Support Vector Machine Process

There are two processes in the M-SVM method, namely: Training and Classification. The process is as follows:

1. Training

The training process is the process of forming a hyperplane which will later be used as a comparison of the data to be classified, as for the process as follows:

a. Generate *Support Vector*

Support Vector is a value that will be used to form a hyperplane for each class, to form a support vector that needs to be done first is labeling provided that the feature data from the class being trained is labeled +1 and the other class is labeled -1. The simulation can be seen in Table 1. 1.

Table 1. 1. Input Support Vector

Expression	Feature	Class	Label
Bahagia	…..	1	+1
Sedih	…..	2	-1
Terkejut	…..	3	-1
Takut	…..	4	-1
Marah	…..	5	-1
Muak	…..	6	-1

To find the kernel value, first determine the type of kernel that will be used. In this research, the Linear kernel trick is used. Calculate the kernel on the feature using the following equation:

$$K(x_i, x) = x_i^T x \quad (20)$$

Then do the calculation to find the y value, using the following equation :

$$K(y_i, y) = y_1^T y \qquad (21)$$

Setelah mendapatkan nilai kernel dari After getting the kernel value from each class,

then add $K(x_i, x)$ so that it forms the X value in support vector machine, do the sum with the following equation:

$$x_i = \sum_{j=1}^{n} x_i^T x_j \quad (22)$$

After getting the X value, find the Y value, by adding $K(y_i, y)$ with the following equation:

$$y_i = \sum_{j=1}^{n} y_i^T y_j \quad (23)$$

The value of x_i and y_i are the support vector values. The notation of support vector is $\begin{matrix} x_i \\ y_i \end{matrix}$.

b. *Kernel Trick*

After x_i and y_i founded, find the kernel trick using equation (24):

$$\varphi\begin{pmatrix} x \\ y \end{pmatrix} = \begin{cases} \sqrt{x_n^2 + y_n^2} > 2, maka \begin{bmatrix} \sqrt{x_n^2 + y_n^2} - x & |x - y| \\ \sqrt{x_n^2 + y_n^2} - x & |x - y| \end{bmatrix} \\ \sqrt{x_n^2 + y_n^2} < 2, maka \begin{pmatrix} x \\ y \end{pmatrix} \end{cases} \quad (24)$$

c. Nilai a

After Kernel Trick founded, then find **a** value using this equation:

$$\sum_{i=1,j=1}^{n} a_i T_i^T T_j \quad (25)$$

d. Nilai Bias

After that, determine the bias value by adding the number 1 in the Kernel Trick seen in the following equation :

$$s_i = \begin{matrix} x_i \\ y_i \\ 1 \end{matrix} \quad (26)$$

e. Nilai Hyperplane

After that, specify the hyperplane value with the following equation:

$$W = \sum_i^n a_i s_i \quad (27)$$

2. Classification

Classification is the stage for predicting which class will be entered into the image, as for the equation as follows:

$$Kelas\ x = \arg\max_{k=1..6} ([w^1]^T.\varphi(x) + b^1, [w^2]^T.\varphi(x) + b^2, [w^3]^T.\varphi(x) + b^3, [w^4]^T.\varphi(x) + b^4), [w^5]^T.\varphi(x) + b^5 \quad (28)$$

2.2 Testing Process

Program testing is a series of stages to test a system that has been built with the aim of knowing whether the system built is in accordance with the provisions.

For testing is done using Confussion Matrix to calculate how much accuracy is obtained, in testing used 3 types of filters used, namely: 3x3, 5x5, and 7x7. The testing is as follows:

3. 3x3 Filter Testing

This test is carried out to determine the accuracy of the 3x3 filter, the test can be seen in Table 1. 2.

Table 1. 2. Confussion Matrix 3x3

	Prediction						Accuration
	Happiness	Sadness	Surprised	Fear	Anger	Disgust	
Happiness	39	0	1	3	2	0	86.67%
Sadness	7	29	0	5	0	4	64.44%
Surprised	1	2	41	1	0	0	91.11%
Fear	6	0	1	32	6	0	71.11%
Anger	1	4	1	5	30	4	66.67%
Disgust	2	4	0	2	4	33	73.33%
Average Accuration							75.56%

Figure 1. 23. Filter Testing 3x3 Chart

4. Filter Testing 5x5

This test is carried out to determine the accuracy of the 5x5 filter, the test can be seen on:

Table 1. 3. Confussion Matrix 5x5

	Prediction						Accuration
	Happiness	Sadness	Surprised	Fear	Anger	Disgust	
Happiness	45	0	0	0	0	0	100.00%
Sadness	0	45	0	0	0	0	100.00%
Surprised	1	0	44	0	0	0	97.78%
Fear	0	0	0	45	0	0	100.00%
Anger	0	1	0	0	43	1	95.56%
Disgust	0	1	0	0	3	41	91.11%
Average Accuration							97.41%

Figure 1. 24. Filter Testing 5x5 Chart

5. Filter Testing 7x7

This test is done to find out how much accuracy the 7x7 filter has, the test can be seen on

Table 1. 4. Confussion Matrix 7x7

	Prediction						Accuration
	Happiness	Sadness	Surprised	Fear	Anger	Disgust	
Happiness	45	0	0	0	0	0	100.00%
Sadness	0	45	0	0	0	0	100.00%
Surprised	1	0	44	0	0	0	97.78%
Fear	1	0	0	44	0	0	97.78%
Anger	0	0	0	0	44	1	97.78%
Disgust	0	1	0	0	3	41	91.11%
Average Accuration							97.41%

Figure 1. 25. Filter Testing 7x7 Chart

From the confusion matrix test using 270 times of detection, the highest accuracy was obtained on 5x5 and 7x7 filters by 97.41%.

3. Conclusion

After testing the MSF-VQ method, the highest results were obtained with a 7x7 filter with 97.14%, in other words the MSF-VQ method was able to recognize facial expressions, only the system was still difficult to recognize if the face was too far away and had difficulty recognizing the expression if there are additional accessories such as glasses, braces, etc. So it needs to be developed again by segmenting objects that determine an expression such as the eyes, mouth, and forehead with the Face sketch synthesis (FSS) method, also need to be eliminated noise such as glasses, braces, etc. With the Learning Residual Images (LRI) method.

REFERENCES

[1] Upadhyay, A, A. Kumar. 2016. Facial Expression Recognition : A Review. International Research Journal of Engineering and Technology. pp. 1616–1620, 2016.

[2] B. W. Adi, Saputra, Tjokorda. 2015. Pengenalan Ekspresi Wajah Menggunakan Local Binary Pattern (LBP) Facial Expression Recognition Using Local Binary Pattern (LBP), Telkom University.1.

[3] R. P. Situmeang. 2017. Implementasi Algoritma Hidden Markov Model Untuk Pengenalan Isyarat Wajah, Universitas Komputer Indonesia.1.

[4] S. Srivastava, 2012. Real Time Facial Expression Recognition Using a Novel Method". International Conference on Advanced Computing and Communication Technologies.4.

[5] Y. Yan, F. Lee, X. Wu, and Q. Chen. 2018. Face Recognition Algorithm using Extended Vector Quantization Histogram Features, PLoS One. 1, pp. 1–24.

[6] P. Hidayatullah. 2017. Pengolahan Citra Digital.

[7] K. Randy. 2013. Aplikasi Pendeteksi Mata Mengantuk Berbasis Citra Digital Menggunakan Metode Haar Classifier Secara Realtime. Skripsi Sarjana diterbitkan, Universitas Komputer Indonesia.

[8] Y. Linde, B. Andres, R.M. Gray. 1980. An Algorithm for Vector Quantizer Design, IEEE Transaction Communication.1.

[9] E. Prasetyo. 2014. Data Mining Mengolah Data Menjadi Informasi Menggunakan Matlab. Yogyakarta: CV. Andi Offset.