IMPLEMENTASI *MARKOV STATIONARY FEATURE – VECTOR QUANTIZATION* UNTUK PENGENALAN EKSPRESI WAJAH

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

Eskpresi Wajah merupakan salah satu bentuk komunikasi, ekspresi wajah memiliki tentunya memiliki bentuk wajah yang berbeda-beda. Dalam proses pengenalan ekspresi wajah mesin belajar tidak langsung mengenali ekspresi begitu saja, namun perlu dilakukan proses ekstraksi fitur terlebih dahulu. Salah satu metode ekstraksi fitur yang dapat digunakan adalah Markov Stationary Feature – Vector Quantization (MSF-VQ). Algoritma MSF-VQ telah diujikan pada kasus pengenalan wajah manusia dan mendapat akurasi sebesar 99.16%. Pada penelitian ini Algoritma MSF-VQ digunakan untuk mengenali ekspresi wajah. Dalam penelitian ini digunakan data yang terdiri dari Data Latih dan Data Uji Total Data yang digunakan untuk proses pengujian sebanyak 1440. Untuk data latih digunakan foto wajah sebanyak 1170 yang didapat dari 15 orang, 6 jenis ekspresi, dan dari satu jenis ekspresi terdapat 13 foto wajah per orang. Pengujian dilakukan menggunakan 270 data uji menggunakan algoritma MSF-VQ menggunakan mesin belajar Multiclass Support Vector Machine dengan kernel linear, berhasil mendapatkan akurasi sebesar 97.41 %.

Kata kunci : Ekspresi Wajah, Markov Stationary Feature, Vector Quantization, Support Vector Machine, Ekstraksi Fitur.

1. PENDAHULUAN

Ekspresi wajah adalah satu atau lebih gerakan atau posisi otot di bawah kulit wajah. Ekspresi wajah adalah bagian yang sangat penting dalam komunikasi. Secara umum ekspresi wajah memiliki 6 jenis ekspresi, yaitu : Bahagia, Sedih, Terkejut, Takut, Marah dan Muak [1].

Pada ekspresi wajah yang sama setiap orang dapat memiliki bentuk wajah yang berbeda, namun ekpsresi wajah memililki pola yang unik. Untuk proses pengenalan ekspresi wajah, mesin pembelajaran harus melalui proses ekstraksi fitur terlebih dahulu agar dapat mengenali pola unik tersebut.

Ekstrasi fitur merupakan proses yang sangat penting, proses ini digunakan untuk mendapatkan nilai-nilai yang nantinya dapat digunakan oleh mesin pembelajaran untuk mengenali ekspresi wajah.

Penelitian-penelitian tentang ekspresi wajah antara lain : Saputra[2] menggunakan metode Local Binary Pattern (LBP) mendapatkan akurasi sebesar 65.1 %, Pengihutan[3] menggunakan Algoritma Discrete Cosine Transform (DCT) dan Principal Component Analysis (PCA) mendapat akurasi sebesar 50%, Saumil[4] menggunakan Linear Discriminant Analysis (LDA) mendapat akurasi sebesar 60%.

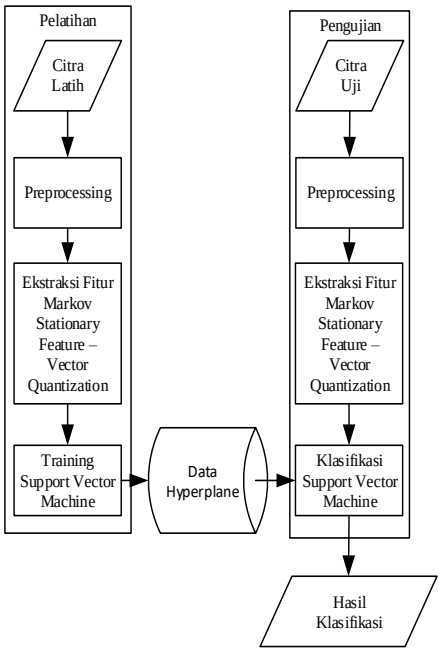
Berdasarkan jurnal Yan dkk.[5] Metode LBP, PCA dan LDA memiliki kekurangan diantarnya : LBP rentan terhadap noise; PCA kurang baik jika citra yang dibandingkan berbeda resolusi atau tingkat kecerahannya berbeda; LDA hasil ekstraksinya kurang baik jika resolusinya rendah. Karena keterbatasan dari metode yang digunakan menyebabkan akurasi dari penelitian-penelitian tersebut masih belum baik.

Markov Stationary Feature - Vector Quantization (MSF-VQ) merupakan metode yang mampu mengatasi kekurangan-kekurangan dari metode yang digunakan pada penelitian-penelitian sebelumnya yang sudah disebutkan, hal ini dibuktikan dengan hasil akurasi yang didapatkan sebesar 99.16% pada kasus pengenalan wajah manusia[5]. Maka dari itu pada penelitian ini MSF-VQ akan dilakukan penelitian guna meningkatkan akurasi dari penelitian sebelumnya.

2. ISI PENELITIAN

2.1. Metode

Sistem pengenalan ekspresi wajah yang akan dibangun terdiri dari dua proses besar yaitu Pelatihan dan Pengujian. Pada tahap pelatihan secara garis besarnya adalah Pre-processing, Ekstraksi Fitur, Pelatihan. Untuk Proses Klasifikasi Melakukan setiap tahap sampai dengan tahap ekstrasi fitur dan melakukan proses klasifikasi dari data yang telah di-*pelatihan* proses dapat dilihat pada Gambar 1. 1.



Gambar 1. 1. Gambaran Umum Sistem

Sistem Secara Umum terbagi menjadi dua tahap besar yaitu : *Pelatihan* dan Klasifikasi. Adapun uraian penjelasannya sebagai berikut :

1. Proses *Pelatihan*

   Dalam proses pelatihan data latih akan melalui proses preprocessing, ekstraksi fitur Markov Stationary Feature – Vector Quantization, Pelatihan Support Vector Machine. Dari tahap diuraikan kembali prosesnya sebagai berikut :

   a. Deteksi Wajah adalah proses pengambilan bagian citra wajah dari citra masukan.
   b. Lowpass filter adalah penyaringan citra yang berintensitas tinggi agar tidak terbawa.

   Setelah proses preprocessing, dilakukan proses ekstrasi fitur menggunakan MSF-VQ. Proses selanjutnya adalah proses Pelatihan Support Vector Machine. Proses ini dilakukan untuk mendapatkan data hyperplane dari data latih yang telah dimasukkan kemudian data hyperplane tersebut dimasukkan ke dalam XML dengan ketentuan dari OpenCV[6].

2. Proses Klasifikasi

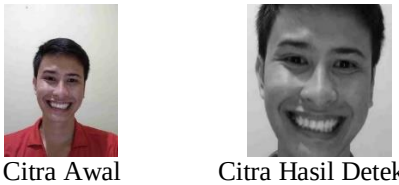
   Proses Klasifikasi adalah proses untuk mengidentifikasi apakah citra yang diproses masuk kelas Bahagia, Sedih, Terkejut, Takut, Marah dan Muak/Jijik. Untuk tahapannya sama dengan proses pelatihan hanya saja proses yang dilakukan hanya sampai ekstrasi fitur dari data hyperplane yang telah disimpan kemudian dibandingkan dengan Metode Klasifikasi Multiclass Support Vector Machine (M-SVM).

2.1.1. Pre-Processing

Proses *Pre-Processing* adalah tahapan untuk mengurangi *noise* dengan tujuan membantu metode ekstraksi fitur agar mendapat akurasi yang lebih baik. Tahapan Preprocessing pada penelitian ini adalah Deteksi Wajah dengan metode *Viola Jones* dan *Lowpass Filter.*

1. Deteksi Wajah (Viola Jones)

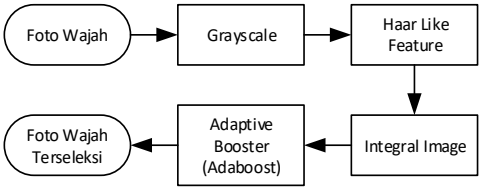
   Deteksi wajah adalah tahapan untuk mengambil citra bagian wajah. Ilustrasinya dapat dilihat pada Gambar 1. 1.



Citra Awal Citra Hasil Deteksi

Gambar 1. 2. Ilustrasi Deteksi Wajah

Pada proses pendeteksian wajah ada beberapa proses yang dilakukan yaitu: Grayscale Haar-Like Feature, Integral image, Adaboost (Adaptive Boosting), dan Cascade Classifie r[7]. Tahap-tahap dari algoritma Viola Jones adalah :



Gambar 1. 3. Alur Pendeteksian Wajah

a. Grayscale

Grayscale adalah proses untuk mengubah citra menjadi warna yang keabuan[8]. Warna memiliki nilai R, G, B nilai-nilai tersebut diubah menjadi sebuah nilai yang disebut dengan nilai grayscale, adapun prosesnya menggunakan persamaan berikut :

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

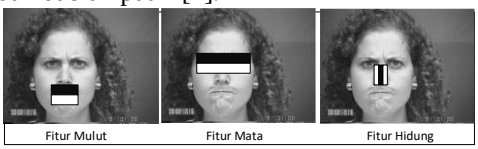
Dari persamaan tersebut dihasilkan citra grayscale sebagai berikut :

Citra Awal Citra Grayscale

Gambar 1. 4. Ilustrasi Grayscale

b. Haar Like Feature

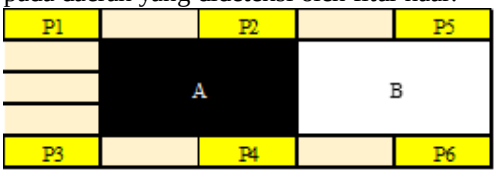
Teknik yang dilakukan yaitu dengan cara mengkotak-kotakkan citra lalu kemudian dan membagi menjadi citra daerah hitam dan daerah putih [7].



Gambar 1. 5. Fitur Haar

c. Integral Image

Integral image digunakan untuk menghitung hasil penjumlahan nilai pixel pada daerah yang dideteksi oleh fitur haar.



Gambar 1. 6. Integral Image

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

d. Adaptive Booster

Adaptive booster adalah tahap untuk meng-*update* bobot dengan melakukan perhitungan menggunakan persamaan sebagai berikut:

m = jumlah gambar negatif, y = 1, gambar negatif adalah citra uji.

l = jumlah gambar positif, y =0, gambar negatif adalah citra latih.

Diketahui :

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

Lalu implementasikan intgral image :

Untuk Citra Positif :

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

Untuk Citra Negatif :

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

Cari nilai error rate :

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

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

Jika $\epsilon_t \vee \epsilon_j < 0$, hentikan iterasi

Kemudian cari bobot baru:

Bobot Positif :

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

Bobot Negatif :

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

Bandingkan dengan persamaan berikut :

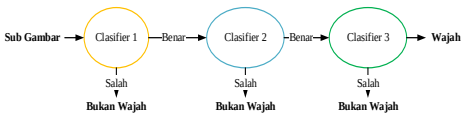
$$H(x) = \begin{cases} 1 \ \sum_{j=1}^{J} \alpha_j h_j \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ \\ 0 \quad bukan\ wajah. \end{cases} \quad (8)$$

Dimana :

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

e. Cascade Clasifier

Setelah diketahui fiturnya kemudian bandingkan citra dengan setiap fitur, dengan ilustrasi sebagai berikut :



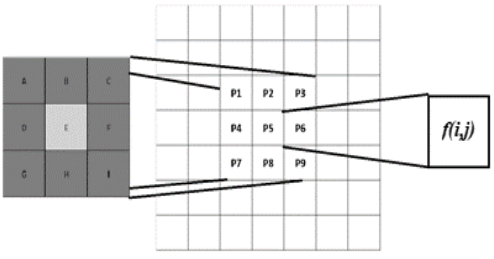
Gambar 1. 7. Cascade Clasifier

2. *Lowpass Filter*

Lowpass Filter atau penapisan lolos rendah adalah suatu proses pada citra untuk meloloskan data pada frekuensi rendah dan akan mengurangi atau menolak data pada frekuensi tinggi. Untuk persamaan yang digunakan adalah sebagai berikut :

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

Untuk proses hitung dari persamaan tersebut diilustrasikan pada Gambar 1. 8.



Gambar 1. 8. Konvolusi Citra

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

Kernel yang digunakan untuk penelitian adalah mean filter dengan kernel sebagai berikut :

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

Dari proses *Lowpass Filter* akan dihasilkan citra sebagai berikut :

Citra Hasil Deteksi Citra Lowpass Filter

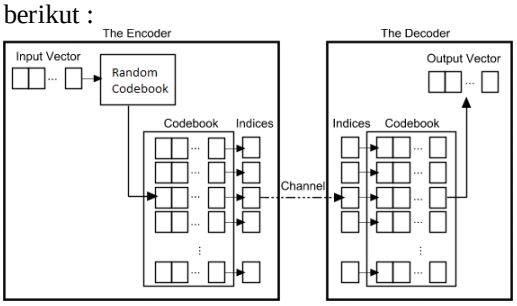
Gambar 1. 9. Citra Hasil *Lowpass Filter*

2.1.2. Ekstraksi Fitur

Ekstraksi fitur adalah proses untuk mendapatkan nilai yang menjadi ciri khas, dan sebagai masukan dari machine learning, dalam penelitian ini ekstraksi fitur ada 2 proses yang bertahap yaitu : *Vector Quantization* dan *Markov Stationary Feature* Adapun tahapannya sebagai berikut :

1. *Vector Quantization*

Vector Quantization (VQ) adalah proses pembentukan citra terkompresi, cara kerja VQ adalah melakukan maping terhadap codebook[9]. Adapun proses dari VQ sebagai berikut :



Gambar 1. 10. Ilustrasi Vector Quantization

Inisialisasi :
k = ukuran dimensi blok.
b = index blok
t = index dari anggota codeword
j = index dari codebook
i = index dari blok
j = index codeword
n = panjang codeword = k x k.
Langkah-langkah VQ :
Langkah 1 : Acak inisial codeboook

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

Diketahui nilai $c_1, c_2, c_3, \dots, c_{Nc}$ merupakan codeword dari codebook,diketahui *codebook* yang diinisialisasi adalah sebagai berikut :

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

Gambar 1. 11. Inisialisasi *Codebook*

Langkah 2 : Bentuk Citra menjadi blok-blok

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

Diketahui nilai $x_1, x_2, x_3, \dots, x_{Nb}$ adalah blok dari citra, untuk blol x_1 diketahui nilainya sebagai berikut :

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

Gambar 1. 12. Blok Citra

Langkah 3 : Cari nilai index, untuk mencari nilai index cari nilai euclidean distance antara inisial *codebook* dengan blok citra untuk mencari jarak.

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

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

$$= 369.83$$

Kemudian lakukan pada setiap codeword, sehingga dihasilkan nilai

$d(x_1, c_2) = 196.636$

$d(x_1, c_3) = 308.412$

$d(x_1, c_4) = 236.548$

Untuk mencari nilai index maka diberikan kenentuan pada persamaan berikut :

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

Persamaan tersebut dimaksudkan untuk mencari nilai euclidean distance terkecil. Diketahui nilai distance terkecil adalah 196.636 maka letak nilai berada pada *codeword* 3, maka index pada blok 1 adalah 3. Selama proses tampung jika index dari blok sama maka tampung ke dalam variabel x_j dan jumlah kemunculannya dihitung dan dan dimasukan ke dalam variabel N_j, Dari kedua variabel yang disimpan berbentuk vektor berdasarkan sesuai dengan ukuran blok.

Langkah 4 : Update codebook

Dari kemunculan index pada setiap blok telah ditampung nilai x_j dan N_j maka lakukan perhitungan dengan persamaan berikut :

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

Nilai tersebut merupakan nilai dari masing-masing nilai tersebut merupakan isi dari codebook baru. Setelah itu cari nilai distorsi dengan melakukan perhitungan *distance* antara *codebook* baru dengan blok :

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

Setelah nilai distorsi dihasilkan, nilai distorsi dibagi dengan jumlah Nb×n.

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

Setelah didapat nilai distorsi lakukan update *codebook* sampai memenuhi dan hentikan proses update *codebook* sampai memenuhi kondisi :

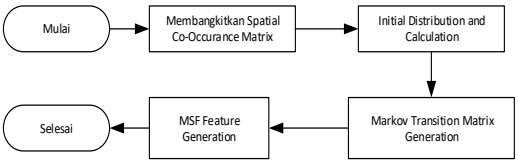
$$\frac{D_{m-1} - D_m}{D_m} \leq \varepsilon = 0.001 \quad (17)$$

Setelah proses update codebook selesai maka lakukan pemetaan terhadap nilai index yang ditampung. Dengan cara mengambil nilai yang ada pada codebook yang telah di-*update*. Sehingga membentuk citra baru yang telah dikompresi sehingga menghasilkan citra sebagai berikut :

Gambar 1. 13. Citra Vector Quantization

2. *Markov Stationary Feature*

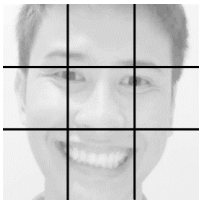
Markov Stationary Feature (MSF) merupakan tahapan untuk membangkitkan fitur dari ekspresi, proses dari MSF dapat dilihat pada Gambar 1. 4.



Gambar 1. 14. Alur Markov Stationary Feature

Adapun Penjelasan dari Alur Markov Stationary Feature adalah sebagai berikut :

a. Bagi gambar menjadi blok-blok sesuai dengan filter dalam implementasi ini dicontohkan menggunakan filter 3x3 dan dapat dilihat pada Gambar 1. 15.



Gambar 1. 15. Blok Filter Citra

b. Membangkitkan *Co-Occurance Matrix*

Sebelum membangkitkan co-occurrence matrix, rentang citra perlu diubah menjadi 0-32. $Warna\ Gray$ adalah nilai warna dari citra, $Graylevel$ merupakan level rentang nilai maksimal, $Graylevel$ yang digunakan bernilai 32. Berikut adalah persamaan yang digunakan :

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

Diketahui citra berukuran 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

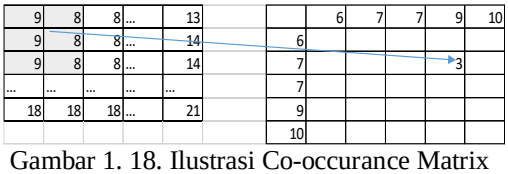
Gambar 1. 16. Citra Blok untuk MSF

Dilakukan perhitungan menggunakan persamaan (18), sehingga menghasilan citra sebagai berikut :

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

Gambar 1. 17. Citra 32

Cari nilai co-occurence matrix dengan mencari nilai ketetanggaan 0^0, 45^0, 90^0, dan 135^0. Untuk Ilustrasi dari co-occurance matrix dapat dilihat pada Gambar 1. 18.



Gambar 1. 18. Ilustrasi Co-occurance Matrix

Dari ilustrasi tersebut maka dihasilkan nilai co-occurance matrix sebagai berikut :

	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

Gambar 1. 19. Co-occurance Matrix

c. Cari Nilai initial distribution
Setelah didapatkan nilai co-occurance matrix maka cari nilai initial distribution sebagai berikut :

	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

Gambar 1. 20. Simulasi initial distribution

Lakukan perhitungan sebagai berikut :

$$sum\pi = 0 + 0 + 318 + 444 + 336 + 582$$

$$+704 + 737 + 671 = 4310$$

$$\pi(0) = ($$

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

Maka didapatkan *initial distribution* :

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

d. Bangkitkan Matrix Transisi
Rata-ratakan nilai matrix co-occurance berdasarkan nilai setiap barisnya sehingga menghasilkan nilai sebagai berikut :

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

Gambar 1. 21. Matrix Transisi

e. Mencari Nilai Stationary Distribution
Untuk mencari nilai *Stationary Distribution* dihitung dengan menggunakan *Markov Chain*, dengan mencari nilai limit dari

$$A = \lim_{n\to\infty} A_n, \quad \text{dimana}$$

$$A_n = \frac{1}{n+1}(I + P + P^2 + \cdots P^n). \quad I$$

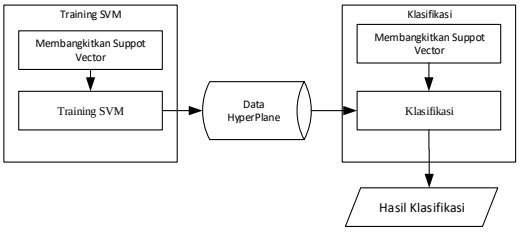
merupakan matrix identitas. Ditentukan nilai n adalah 50, nilai didapatkan dari rata-rata setiap baris $\overrightarrow{a_i}$ dari A_n. Untuk menghitung fitur maka digunakan persamaan sebagai berikut :

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

Hasil dari perhitungan tersebut berupa vektor yang nantinya digabungkan dengan initial distribution, hasil dari gabungan initial distribution digunakan sebagai fitur yang akan dikenali oleh mesin belajar Multiclass Support Vector Machine.

2.1.1 Multiclass Support Vector Machine

Tahap selanjutnya akan dilakukan pembelajaran dan pengujian pada data latih dengan histogram data uji, menggunakan metode M-SVM Pada metode Multiclass Suppport Vector Machine (M-SVM). Metode M-SVM sebenarnya hanya dapat melakukan klasifikasi biner (dua kelas). Pendekatan yang digunakan adalah *one-againist-all* (OAA).[10] Untuk melihat proses dari M-SVM digambarkan pada Gambar 1. 22.



Gambar 1. 22. Tahap Multiclass Support Vector Machine

Terdapat dua buah proses dalam metode M-SVM yaitu : *Pelatihan* dan Klasifikasi. Adapun prosesnya sebagai berikut :

1. Pelatihan
   Proses pelatihan merupakan proses pembentukan hyperlpane yang nantinya akan digunakan sebagai pembanding dari data yang akan diklasifikasi, adapun prosesnya sebagai berikut :
   a. Buat *Support Vector*
      Support Vector adalah nilai yang akan digunakan untuk membentuk hyperplane setiap kelas, untuk membentuk support vector yang perlu dilakukan pertama adalah

melakukan pelabelan dengan ketentuan data fitur dari kelas yang di-pelatihan diberikan label +1 dan kelas yang lainnya diberikan label -1. Simulasinya dapat dilihat pada Tabel 1. 1.

Tabel 1. 1. Input Support Vector

Ekspresi	Fitur	Kelas	Label
Bahagia	…..	1	+1
Sedih	…..	2	-1
Terkejut	…..	3	-1
Takut	…..	4	-1
Marah	…..	5	-1
Muak	…..	6	-1

Untuk mencari nilai kernel tentukan dulu jenis kernel yang akan digunakan, dalam penelitian ini digunakan kernel trick Linear, lakukan perhitungan kernel pada fitur dengan menggunakan persamaan berikut :

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

Lalu lakukan perhitungan untuk mencari nilai y, dengan menggunakan persamaan sebagai berikut :

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

Setelah mendapatkan nilai kernel dari masing-masing kelas, kemudian jumlahkan $K(x_i, x)$ sehingga membentuk nilai X pada support vector machine, lakukan penjumlahan dengan persamaan sebagai berikut :

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

Setelah mendapatkan nilai X cari nilai Y, dengan jumlahkan $K(y_i, y)$ dengan persamaan sebagai berikut :

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

Nilai x_i dan y_i merupakan nilai *support vector*. Notasi dari support vector adalah $\begin{matrix} x_i \\ y_i \end{matrix}$

b. *Kernel Trick*

Setelah nilai x_i dan y_i cari nilai kernel *trick* dengan menggunakan persamaan berikut :

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

c. Nilai a

Setelah didapat nilai *Kernel Trick*, cari nilai a dengan menggunakan persamaan berikut :

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

d. Nilai Bias

Setelah itu tentukan nilai bias dengan menambahkan angka 1 pada *Kernel Trick* dilihat pada persamaan berikut :

$$s_i = \begin{matrix} x_i \\ y_i \\ 1 \end{matrix}$$

e. Nilai Hyperplane

Setelah itu tentukan nilai hyperplane dengan persamaan berikut :

$$W = \sum_i^n a_i s_i \quad (26)$$

2. Klasifikasi

Klasifikasi merupakan tahap untuk melakukan prediksi akan masuk kelas manakah citra yang diuji, adapun persamaannya sebagai berikut :

$$Kelas\ x = \arg\max_{k=1..6} ([w^1]^T.\varphi(x) + b^1, [w^2]^T.\varphi(x) + b^2, [w^3]^T.\varphi(x) + b^3, [w^4]^T.\varphi(x) + b^4), [w^5]^T.\varphi(x) + b^5 \quad (27)$$

2.2 Proses Pengujian

Pengujian program merupakan serangkaian tahapan untuk menguji sistem yang telah dibangun dengan tujuan untuk mengetahui sistem yang dibangun apakah telah sesuai dengan ketentuan.

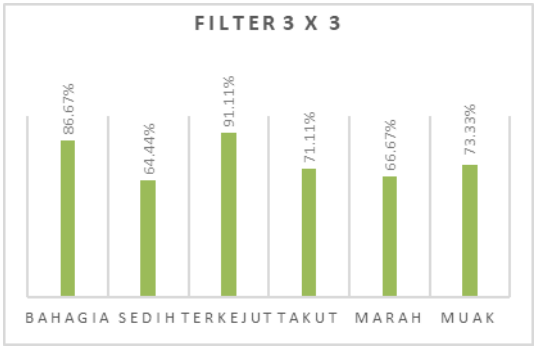
Untuk pengujian dilakukan menggunakan Confussion Matrix untuk menghitung seberapa besar akurasi yang didapat, dalam pengujian digunakan 3 jenis filter yang digunakan yaitu : 3x3, 5x5, dan 7x7. Adapun pengujiannya sebagai berikut :

3. Pengujian Filter 3x3

Pengujian ini dilakukan untuk mengetahui seberapa besar akurasi pada filter 3x3, pengujiannya dapat dilihat pada Tabel 1. 2.

Tabel 1. 2. Confussion Matrix 3x3

Sebenarnya	Prediksi						Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	
Bahagia	39	0	1	3	2	0	86.67%
Sedih	7	29	0	5	0	4	64.44%
Terkejut	1	2	41	1	0	0	91.11%
Takut	6	0	1	32	6	0	71.11%
Marah	1	4	1	5	30	4	66.67%
Muak	2	4	0	2	4	33	73.33%
Rata-Rata Akurasi							75.56%



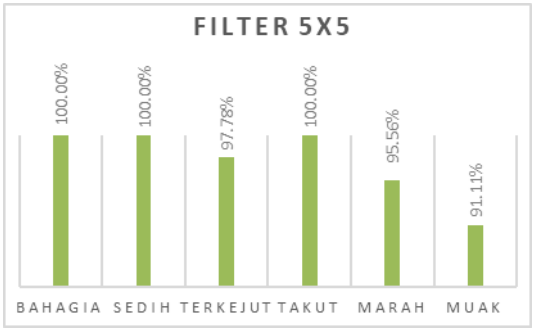
Gambar 1. 23. Grafik Pengujian Filter 3x3

4. Pengujian Filter 5x5

Pengujian ini dilakukan untuk mengetahui seberapa besar akurasi pada filter 5x5, pengujiannya dapat dilihat pada Tabel 1. 3.

Tabel 1. 3. Confussion Matrix 5x5

Sebenarnya	Prediksi						Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	
Bahagia	45	0	0	0	0	0	100.00%
Sedih	0	45	0	0	0	0	100.00%
Terkejut	1	0	44	0	0	0	97.78%
Takut	0	0	0	45	0	0	100.00%
Marah	0	1	0	0	43	1	95.56%
Muak	0	1	0	0	3	41	91.11%
Rata-Rata Akurasi							97.41%



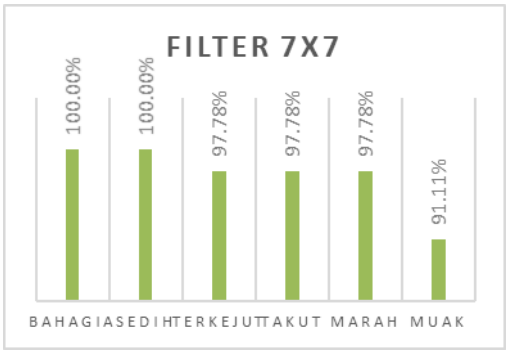
Gambar 1. 24. Grafik Pengujian Filter 5x5

5. Pengujian Filter 7x7

Pengujian ini dilakukan untuk mengetahui seberapa besar akurasi pada filter 7x7, pengujiannya dapat dilihat pada Tabel 1. 4

Tabel 1. 4. Confussion Matrix 7x7

Sebenarnya	Prediksi						Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	
Bahagia	45	0	0	0	0	0	100.00%
Sedih	0	45	0	0	0	0	100.00%
Terkejut	1	0	44	0	0	0	97.78%
Takut	1	0	0	44	0	0	97.78%
Marah	0	0	0	0	44	1	97.78%
Muak	0	1	0	0	3	41	91.11%
Rata-Rata Akurasi							97.41%



Gambar 1. 25. Grafik Pengujian Filter 7x7

Dari pengujian conffusion matrix menggunakan 270 kali pendeteksian, didapatkan akurasi tertinggi pada filter 5x5 dan 7x7 sebesar 97.41%.

3. PENUTUP

Setelah melakukan pengujian terhadap Metode MSF-VQ maka didapatkan hasil tertinggi pada pengujian dengan filter 7x7 sebesar 97.14%, dengan kata lain Metode MSF-VQ mampu mengenali ekspresi wajah, hanya saja sistem masih sistem masih sulit mengenali jika bentuk wajahnya terlalu jauh dan kesulitan mengenali ekspresi jika ada aksesoris tambahan seperti kacamata, kawat gigi, dsb. Sehingga perlu dikembangkan lagi dengan melakukan segmentasi terhadap objek-objek yang menentukan sebuah ekspresi seperti mata, mulut, dan dahi dengan metode *Face sketch synthesis* (FSS), juga perlu dihilangkan lagi *noise* seperti kacamata, kawat gigi, dsb. Dengan metode Learning Residual Images (LRI).

DAFTAR PUSTAKA

[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] The OpenCV Tutorials 2.3. 2011. Intel Corporation.

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

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

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

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