PENGARUH SEGMENTASI MENGGUNAKAN VIOLA AND JONES TERHADAP FITUR MARKOV STATIONARY FEATURE - VECTOR QUANTIZATION PADA KASUS PENGENALAN EKSPRESI WAJAH

Encep Suryana[1], Irfan Maliki[2]

[1,2] Program Studi Teknik Informatika
Fakultas Teknik dan Ilmu Komputer - Universitas Komputer Indonesia
Jl. Dipatiukur No. 112-114 Bandung
E-mail : encep.suryanajr@gmail.com[1], irfanmaliki007@gmail.com[2]

ABSTRAK

Ekspresi wajah berperan penting dalam berkomunikasi dengan orang lain sebagai ungkapan perasaan atau emosi. Pada pendeteksian ekspresi wajah dapat dilakukan proses segmentasi yang bertujuan untuk memisahkan objek yang terdapat pada citra wajah diantaranya: kedua mata, hidung dan mulut. Salah satu metode segmentasi yang dapat digunakan adalah *Viola and Jones*. Pada kasus pengenalan ekspresi wajah tanpa segmentasi menggunakan metode *Markov Stationary Feature – Vector Quantization* (MSF-VQ) dan menggunakan metode pengujian *Confusion Matrix* mendapatkan akurasi 97.01%. Pada penelitian ini data yang digunakan terdiri dari Data Latih dan Data Uji, total data sebanyak 2205 data dari 42 orang, diantaranya: 21 orang berekspresi bahagia, sedih, terkejut, takut, marah, muak dan 21 orang lainnya berekspresi netral. Data diambil sebanyak 15 citra per orang. Penelitian ini yang berjudul "pengaruh segmentasi *Viola and Jones* terhadap fitur MSF-VQ pada kasus pengenalan ekspresi wajah" dibantu dengan mesin pembelajaran *Multiclass Support Vector Machine* dengan *kernel linear* berpengaruh pada kecepatan dalam pelatihan dan pengujian data menjadi lebih cepat dan menghasilkan akurasi yang lebih baik dalam proses pengenalan ekspresi wajah mendapatkan akurasi sebesar sebesar 100% pada pengenalan ekspresi wajah dan adanya peningkatan sebesar 2.99%.

Kata kunci : Ekspresi Wajah, Segmentasi, *Viola and Jones, Markov Stationary Feature, Vector Quantization, Support Vector Machine,* Pengujian, *Confusion Matrix,* Ekstraksi Fitur.

1. PENDAHULUAN

Ekspresi wajah berperan penting dalam berkomunikasi dengan orang lain sebagai ungkapan perasaan atau emosi [1]. Pada pendeteksian ekspresi wajah dapat dilakukan proses segmentasi yang bertujuan untuk memisahkan objek yang terdapat pada citra wajah diantaranya: kedua mata, hidung dan mulut, dan segmentasi banyak digunakan untuk keperluan di bidang *human computer interaction* dan *data driven animation*, terutama pada segmentasi citra wajah, digunakan untuk mempartisi citra digital wajah menjadi beberapa segmen atau piksel yang bertujuan untuk menyederhanakan atau merubah representasi sebuah citra wajah menjadi lebih mudah untuk dianalisis [2].

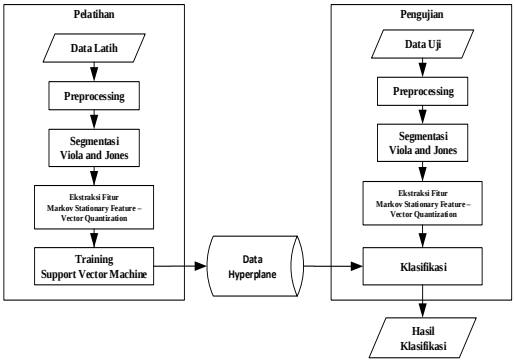
Penelitian-penelitian yang telah dilakukan tentang segmentasi diantaranya: Yusron melakukan penelitian deteksi wajah berdasarkan segmentasi model warna menggunakan *Template Matching* pada objek bergerak mendapatkan akurasi sebesar 65% [3], Douglas melakukan penelitian tentang segmentasi wajah menggunakan metode *skin-color map* pada aplikasi *videophone* mendapatkan akurasi sebesar 82% [4].

Berdasarkan penelitian yang telah dilakukan oleh Chandra menggunakan metode *Viola and Jones* mendapatkan akurasi sebesar 90,99% untuk mendeteksi objek wajah [5]. Penelitian oleh Dharani tentang *Human Segmentation Using Haar-Classifier* mendapatkan akurasi sekitar 93% [2] dan penelitian tugas akhir oleh Fascal implementasi *Markov Stationary Feature – Vector Quantization* Untuk Pengenalan Ekspresi Wajah mendapatkan akurasi sebesar 97.14% [6]. Maka dari itu penelitian tentang Pengaruh Segmentasi *Viola and Jones* terhadap Fitur *Markov Stationary Feature – Vector Quantitazion* pada kasus pengenalan ekspresi wajah, guna menyederhanakan area penelitian dari objek wajah dengan menggunakan metode segmentasi *viola and jones*.

2. ISI PENELITIAN

2.1. Analisis Metode

Analisis metode yang digunakan memiliki dua proses diantaranya proses pelatihan dan proses pengujian. Proses pelatihan digunakan untuk melatih data latih dan proses pengujian digunakan untuk pengujian terhadap data latih pada data uji, berikut adalah alur sistem pada penelitian ini:



Gambar 1. Alur sistem

Dari analisis metode berikut adalah penjelasan dari alur sistem:

2.1.1 Proses Pelatihan

Proses pelatihan dilakukan untuk proses pelatihan data masukan berupa citra latih dan kemudian melalui proses *preprocessing*, segmentasi *viola and jones*, ekstraksi fitur *markov stationary feature – vector quantization* dan *Training Support Vector Machine*. Untuk tahapan dari proses pelatihan sebagai berikut:

2.1.1.1. *Preprocessing*

Preprocessing adalah tahap untuk merubah ukuran citra menjadi lebih kecil kemudian merubah citra berwarna (RGB) menjadi citra *grayscale* dan *image smoothing* sebelum proses segmentasi dan ekstraksi fitur. Adapun tahapannya sebagai berikut:

1. *Resize Image*

Resize Image digunakan untuk mengubah ukuran citra menjadi lebih kecil. Proses ini digunakan agar proses perhitungan menjadi lebih cepat dan tidak banyak menghabiskan memori penyimpanan sementara, berikut proses perhitungannya:

$$ResizeW = w \times SkalaResize \quad (1)$$
$$ResizeH = h \times SkalaResize \quad (2)$$

Keterangan:
ResizeW = ukuran lebar citra baru (*width*)
ResizeH = ukuran panjang citra baru (*height*)
w = ukuran lebar citra asli (*width*)
h = ukuran panjang citra asli (*height*)

2. *Grayscale*

Proses *grayscale* atau konversi citra berwarna (RBG) ke citra keabuan bertujuan untuk merubah citra RGB menjadi satu nilai saja. Sebagai contoh sebagai berikut:

Gambar 2. *Grayscale*

Proses merubah citra RGB menjadi citra *grayscale*, menggunakan perhitungan berikut:

$$\textbf{\textit{Grayscale}} = \mathbf{R \times 0.29 + G \times 0.59 + B \times 0.11} \quad (3)$$

Keterangan:
R : nilai dari piksel warna merah (*Red*)
G : nilai dari piksel warna hijau (*Green*)
B : nilai dari piksel warna biru (*Blue*)

3. *Image Smoothing*

Proses *image smoothing* digunakan untuk merubah citra menjadi lebih halus dengan menggunakan metode konvolusi sebagai perhitungannya, dimana metode ini merupakan proses untuk memperoleh suatu piksel didasarkan pada nilai piksel itu sendiri dan piksel tetangganya, dengan melibatkan suatu matriks yang disebut karnel yang merepresentasikan pembobotan. Proses perhitungan dari penghalusan citra menggunakan filter rata-rata.

$$\boldsymbol{h(x,y) = \frac{1}{mn}\sum_{x=0}^{m-1} \quad \sum_{y=0}^{n-1}} \quad (4)$$

Keterangan:
h(x,y) = filter *h* (filter rata-rata)
n = jumlah baris pada filter *h* (filter rata-rata)
m = jumlah kolom pada filter *h* (filter rata-rata)
x = koordinat letak citra pada titik *x*
y = koordinat letak citra pada titik *y*

Citra *grayscale* didefinisikan sebagai *f(x,y), kemudian g*unakan karnel 3×3 untuk proses penghalusan citra sebagai *h(x,y)* dan merupakan nilai konvolusi dari *f(x,y)* pada citra *grayscale*, sebagai berikut:

$$h(x,y) = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Berikut tahapan untuk merubah citra menjadi lebih halus terhadap nilai *f(x,y)* sebagai berikut:

i. Pilih *f(x,y)* ukuran 3×3, dimulai dari pojok kiri atas hingga kanan bawah,
ii. Kemudian digantikan nilai piksel pada posisi tengah dengan nilai piksel setelah proses perhitungan konvolusi,
iii. Selanjutnya geser piksel 3×3 satu piksel kekanan dan lakukan sampai nilai piksel terakhir, dan
iv. Untuk nilai piksel yang berada pada bagian sisi kanan, kiri, atas dan bawah atau piksel luar nilai pikselnya tetap atau nilai piksel sebelumnya dari nilai piksel *grayscale*.

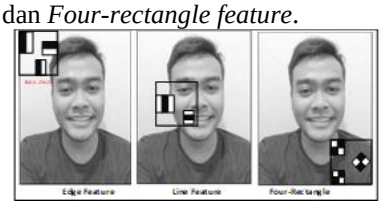
2.1.1.2. Segmentasi *Viola and Jones*

Proses segmentasi digunakan untuk memisahkan objek dengan latar belakang yang tidak dibutuhkan untuk mempersempit area penelitian pada wajah difokuskan pada objek mulut, objek mata dan hidung. Berikut tahapannya:

1. *Haar-Like Feature*

Haar-Like Feature merupakan teknik untuk pendeteksian objek pada citra dengan cara

mengkotak-kotakkan setiap piksel yang disebut dengan basis pada citra mulai dari ujung kiri atas sampai kanan bawah, proses ini dilakukan untuk pencarian apakah ada fitur wajah pada citra. Dalam algoritma *Viola and Jones*, ada beberapa jenis fitur yang bisa digunakan seperti *Edge-feature, Line feature,* dan *Four-rectangle feature*.



Gambar 3. Ilustrasi *haar-like feature*

2. *Integral image*

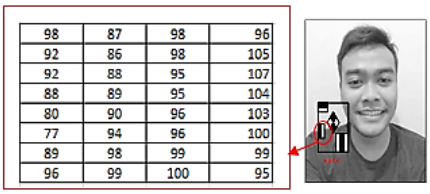
Integral Image digunakan untuk proses pendeteksian objek, untuk proses perhitungan menggunakan *integral image* menggunakan waktu lebih cepat dan hasil yang akurat, perhitungan didapat dari nilai-nilai piksel pada fitur *haar*, kemudian akan dihitung nilai *integral image* dengan rumus berikut:

$$s(x,y) = i(x,y) + s(x,y) + s(x,y-1) + s(x-1,y) - s(x-1,y-1) \quad (5)$$

Keterangan:

s(x,y) = merupakan nilai hasil penjumlahan dari tiap-tiap piksel.
i(x,y) = merupakan nilai intensitas diperoleh dari nilai piksel dari citra masukan.
s(x-1,y) = merupakan nilai piksel pada sumbu *x*.
s(x,y-1) = merupakan nilai piksel pada sumbu *y*.
s*(x-1,y-1)* = merupakan nilai piksel diagonal.

Berikut proses perhitungan *integral image*:



98	87	98	96
92	86	98	105
92	88	95	107
88	89	95	104
80	90	96	103
77	94	96	100
89	98	99	99
96	99	100	95

Gambar 4. Nilai piksel pada sebuah fitur.

i. Perhitungan pada piksel(36,57):

i(x,y) =98	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0
0	0	0	0

Gambar 5. Nilai piksel(36,57)

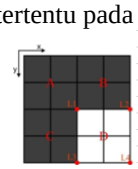
Nilai intensitas piksel(36,57), dan*i(x,y)* = 98
i(x,y) = 98
s(x-1,y) = 0 (di luar batas matriks)
s(x,y-1) = 0 (di luar batas matriks)
s(x-1,y-1) = 0 (di luar batas matriks)
s(x,y) = 98 + 0 + 0 - 0 = 98.

Perhitungan dilakukan hingga piksel terakhir, mendapatkan nilai citra *integral* sebagai berikut:

98	185	185	194
190	167	195	201
184	174	195	209
180	173	191	211
168	171	192	208
157	181	196	203
166	204	199	202
185	204	200	195

Gambar 6. Hasil *image integral*

Kemudian lakukan perhitungan untuk mencari wilayah tertentu pada citra sebagai berikut:



98	185	185	194
190	167	195	201
184	174	195	209
180	173	191	211
168	171	192	208
157	181	196	203
166	204	199	202
185	204	200	195

Gambar 7. Hitung piksel pada daerah tertentu

Proses perhitungannya dengan menjumlahkan piksel pada daerah D dengan rumus sebagai berikut:

$$\mathbf{D = L1 + L4 - (L2 + L3)} \quad (6)$$

3. *Adaptive booster*

Adaptive booster merupakan teknik digunakan untuk mengkombinasikan banyak *classifier* lemah dan membentuk suatu gabungan *classifier* yang lebih baik. Dengan menggunakan algoritma *adaboost* sebuah citra kemudian dideteksi kembali, perhitungannya sebagai berikut:

Bobot awal: $\boldsymbol{w_{j_{y_i}} = \frac{1}{2m}, w_{t_{y_i}} = \frac{1}{2l}}$ (7)

keterangan:

w = *week classifier*
m = jumlah citra positif
l = jumlah citra negatif
t = indeks iterasi dari citra positif
j = indeks iterasi dari citra negatif
$h_t(x)$ = nilai fitur citra positif
$h_j(x)$ = nilai fitur citra negatif

Untuk mendapatkan nilai *error rate* pada setiap *weak classifier*, maka pada setiap fitur dilakukan proses sebagai berikut:

i. Hitung nilai bobot awal:

Bobot awal = $w_{1,0} = \frac{1}{2\times4} = 0.125, w_{1,1} = \frac{1}{2\times1} = 0.5$

Tentukan citra positif, untuk proses peteksian objek atau bukan:

a. Citra positif ke-1

173	150	150	149	146	146	146	146	153	151	150	145	144	142	142	142	145	144
151	150	150	149	146	146	146	146	151	150	149	145	144	143	142	142	145	143
151	150	150	149	146	146	146	146	150	150	149	145	144	143	143	143	145	143
151	150	150	149	146	146	146	146	150	149	149	146	145	144	144	143	145	143
151	150	150	149	146	146	146	146	146	146	146	146	145	145	144	144	145	143
151	150	150	149	146	146	146	146	145	145	146	146	146	146	145	145	145	143
151	150	150	149	146	146	146	146	144	144	145	146	146	146	146	145	145	143
151	150	150	149	146	146	146	146	143	144	145	146	149	149	146	146	145	143
151	151	150	150	150	149	133	146	132	146	146	149	149	146	146	146	145	143
151	151	150	150	149	149	146	146	145	146	146	146	146	146	146	145	145	143
151	150	150	150	149	146	146	146	145	146	146	146	146	146	146	145	145	143
155	150	150	150	149	146	146	145	147	145	146	146	146	146	145	145	145	146

Gambar 8. Citra positif ke-1

Perhitungan:

a) $h_t(x)$ = |((147+173)-(155+153))-((146+153)-(147+144))| = |12-(-8)|=4
b) Nilai *error rate* yang didapatkan:

$$\epsilon_t = (0.125)|4 - 1| = 0.375$$

Lakukan hingga seluruh citra positif dan menghasilkan *error rate* sebagai berikut: 0.375, 29.13, 31.875 dan 18.7. kemudian pilih *error rate* terkecil yaitu 0.375, dimana: $\beta_t = \frac{0.375}{1-0.375} = 0.6$

ii. Bobot iterasi 1 adalah: $w_{2,1}$ = 0.375×0.6 = 0.225

ϵ_t = (0.375+0.225)|2-1| = 0.6

$$\beta_t = \frac{\epsilon_t}{1-\epsilon_t} = \frac{0.6}{1-0.6} = 1.5$$

Lakukan hingga iterasi ke n < 0 maka iterasi berhenti, untuk mendapatkan hasil dari klasifikasi pada citra positif menggunakan rumus berikut:

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

Dimana:

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

Kondisi:

Jika posisi $H(x)$ = hasilnya 1 maka citra tersebut objek

Jika posisi $H(x)$ = hasilnya 0 maka citra tersebut bukan objek

Keterangan:

$H(x)$ = *Strong Classifier* atau klasifikasi yang menyatakan objek atau bukan.

α_j = Tingkat pembelajaran citra positif.

α_t = Tingkat pembelajaran citra negatif.

β_j = Nilai bobot setelah *error rate* pada citra negatif.

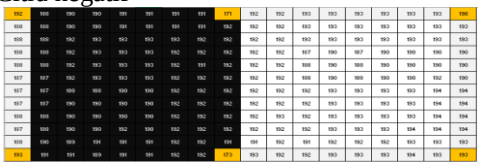
β_t = Nilai bobot setelah *error rate* pada citra positif.

h_j = *weak* atau *basic classifiers* (awal dari klasifikasi) citra negatif.

h_t = *weak* atau *basic classifiers* (awal dari klasifikasi) citra positif.

Menentukan citra negatif, merupakan objek atau bukan, sebagai berikut:

a. Citra negatif



Gambar 9. Citra negatif

Perhitungannya sebagai berikut:

1. $h_j(x)$ = |((173+192)-(193+171))-((193+171)-(173+198))| = |1-(-7)|=8
2. Nilai *error rate* yang didapatkan:
   ϵ_j = (0.08)|8-1| = 0.58
3. Nilai bobot setelah *error rate*
   $\beta j = \frac{\epsilon_j}{1-\epsilon_j}$ Maka $\beta j = \frac{0.58}{1-0.58}$ = 1.38
   ϵ_t = (1 + (1.38))| 1.38| = 3.28

Setelah iterasi 1, kemudian *update* bobotnya sebagai berikut:

$w_{3,1}$ = 0.08×(-1,462) = -0.115

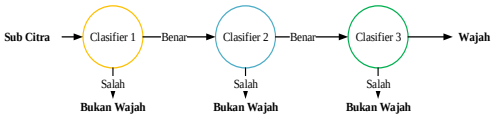
Untuk mencari nilai pada $H(x) = \sum_j^J a_j h_j(x) \geq \frac{1}{2} \sum_t^T a_t$ digunakan untuk menentukan apakah citra negatif merupakan objek yang dicari atau bukan, berikut adalah hasilnya:

$H(x) = 32.97984177 \geq -0.069939543$

Karena $32.97984177 \geq -0.069939543$ merupakan bernilai benar maka citra negatif merupakan objek.

4. *Casecade Classifier*

Cascade classifier merupakan metode untuk mengkombinasikan *classifier* yang kompleks dalam sebuah struktur yang bertingkat dan dapat meningkatkan kecepatan pendeteksian sebuah objek pada citra yang memfokuskan pada daerah citra yang berpeluang saja, dan berikut adalah proses dari *cascade classifier*:



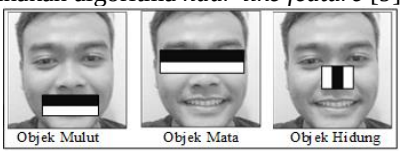
Gambar 10. *Cascade Classifier*

Dari proses *haar clasified* kemudian dihasilkan gambar yang telah terseleksi bagian wajah, seperti gambar berikut ini:

Gambar 11. Citra Terdeteksi

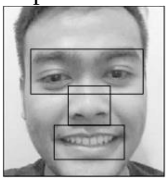
5. *Bounding Box haar-like feature*

Proses ini merupakan proses untuk mencari objek pada citra seperti objek kedua mata, objek hidung dan objek mulut dengan memberikan tanda pada area yang akan digunakan berupa garis horizontal dan vertikal yang membentuk *box* pada objek tersebut. Berikut hasil dari pencarian menggunakan algoritma *haar-like feature* [9]:



Gambar 12. Hasil citra pendeteksian

Setelah itu lakukan proses pemotongan citra yang telah memiliki *bounding box* untuk memisahkan objek yang tidak diperlukan pada citra wajah. Berikut contoh *bounding box* pada citra:



Gambar 13. *Bounding box* citra

Berikut contoh citra yang telah dipotong, dengan memisahkan objek yang tidak digunakan:

Gambar 14. Citra yang telah dipotong.

6. *Thresholding*

Dengan menggunakan *thresholding* untuk merubah citra menjadi citra biner [7], pada penelitian ini nilai derajat keabuan menggunakan nilai 195, nilai tersebut diambil dari pengujian citra sebelumnya menggunakan bantuan *tools imageJ* [12].

Pada dasarnya proses *thresholding* adalah merubah kuantisasi pada citra, sehingga untuk melakukan *thresholding* dengan derajat keabuan menggunakan rumus sebagai berikut:

$$\boldsymbol{x = \frac{w}{b}} \quad (10)$$

Keterangan:

w = nilai derajat keabuan sebelum *thresholding.*

b = jumlah derajat keabuan yang diinginkan.

x = nilai derajat keabuan setelah *thresholding*.

Berikut adalah hasil perbandingan citra terpotong dan citra setelah tahap *thrusholding*, sebagai berikut:

Gambar 15. Citra *thresholding*

2.1.1.3. Ekstraksi fitur *Markov Stationary Feature – Vector Quantization*

Ekstraksi fitur *markov stationary feature – vector quantization* (MSF-VQ) adalah untuk kompresi citra dan membangkitkan setiap fitur pada citra untuk mengenali ekspresi. Berikut adalah tahapannya:

1. Metode *Vector Quantization*

Vector quantization (VQ) digunakan untuk kompresi citra agar citra kualitasnya menjadi lebih baik dan salah satu metode *kompresi lossy* yang paling kuat dengan kompleksitas perhitungan rendah [8] [10]. Berikut tahapan dari metode VQ:

a. Pembuatan *codebook* secara acak, dengan itisial CB_0 sebagai *codebook.*

b. Tambahkan *i = 0.*

c. Lakukan tahap berikut untuk setiap vektor latih:

   i. Hitung jarak *Euclidean* antara vektor latihdan *codebook* dalam CB_i. Dan jarak *Euclidean* didefinisikan sebagai berikut:

$$d(X, C) = \sqrt{\sum_{t=1}^{k} (x_t - c_t)^2} \quad (11)$$

   ii. Cari *codeword* terdekat pada CB_i

d. Pisahkan *codebook* kedalam kolom *N*.

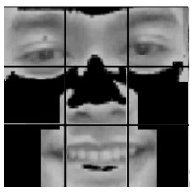
e. Hitung rata-rata dari setiap kolom untuk mendapatkan *codebook* baru CB_i.

f. Hitung distorsi rata-rata pada CB_{i+1}, Jika diubah dengan jumlah terkecil pada iterasi terakhir, perhitungan *codebook* terus diulang sampai prosdur terhenti. Jika tidak *i = i + 1* dan lanjutkan kepada tahap ke-3.

2. *Markov Stationary Feature*

Markov Stationary Feature (MSF) digunakan untuk membangkitkan fitur pada setiap ekspresi dari citra yang telah diproses sampai tahap *Vector Quantization* [11], adapun tahapannya sebagai berikut:

Sebelum proses dari MSF, ditambahkan blok-blok pada setiap citra dengan memberikan filter seperti filter 3×3, 5×5 dan 7×7 yang digunakan untuk pembobotan pada ekstraksi ciri pada citra, dan berikut adalah contoh menggunakan filter 3×3 pada citra latih:



Gambar 16. Filter 3×3 pada citra

a. Membangkitkan *Co-Occurrance Matrix*

Pembangkitan *co-occurance matrix* digunakan untuk ekstraksi fitur dengan menormalisasikan citra latih dengan mengubah nilai intensitas piksel. Dengan cara merubah citra latih menjadi citra 5 bit rentang nilai antara 0-32 bit, menggunakan rumus berikut ini:

$$\boldsymbol{Citra\ Uji = Nilai\ Grayscale} * \frac{\boldsymbol{Nilai\ Graylevel}}{\boldsymbol{255}} \quad (12)$$

Kemudian lakukan perhitungan seluruh nilai citra latih berdasarkan nilai sudut yang digunakan adalah 0^0, 45^0, 90^0, dan 135^0

b. Pencarian *Initial Distribution*

Pencarian *intial distribution* dengan cara menyeleksi matriks *co-occurance* secara diagonal, dari kiri atas sampai kanan bawah, sehingga mendapatkan nilai *inisialisai matriks*, kemudian lakukan penjumlahan nilai secara diagonal dan lakukan merata-ratakan nilai tersebut dengan membagikannya dengan jumlah nilai diagonal tersebut.

c. Pembangkitan *Markov Transition Matrix*

Untuk membangkitkan *markov transition matrix* dengan menggunakan perhitungan dari matriks *co-occurance* terhadap *markov transition matrix*, menentukan nilai p_{ij} dengan menggunakan rumus berikut:

$$p_{ij} = c_{ij} / \pi \approx \frac{1}{K} \sum_{i=1}^{K} c_{ij} \quad (13)$$

Keterangan:

p_{ij} = sebagai *markov transition matrix*.

c_{ij} = sebagai matriks dari *Initial Distribution*

K = nilai dari *Initial Distribution.*

d. Mencari *Stationary Distribution*

Untuk mencari nilai *stationary distribution* menggunakan perhitungan *Markov Chain*, dengan melakukan pencarian nilai limit dari $A = \lim_{n\to\infty} A_n$, dimana $A_n = \frac{1}{n+1}(I + P + P^2 + \cdots P^n)$ (14)

Keterangan:

I = merupakan matriks identitas.

n = nilai rata-rata setiap baris $\overrightarrow{a_i}$ dari A_n.

Lakukan perhitunggan pada nilai fitur menggunakan rumus berikut:

$\pi \approx \frac{1}{K}\sum_{i=1}^{K} \vec{\alpha}_{ij}$, dimana $A_n = [\overrightarrow{a_1}, \dots, \overrightarrow{a_n}]^T$ (15)

e. MSF *Feature Generation*

MSF *Feature Generation* merupakan tahapan untuk menormalisasi matriks transisi digunakan untuk mencari Fitur MSF dilakukan untuk mencari nilai $\overrightarrow{MSF} = [\pi(0), \pi]^T$ (16)

2.1.1.4. Training *Multiclass Support Vector Machine*

Multiclass Suppport Vector Machine (M-SVM) digunakan untuk melakukan klasifikasi biner, metode ini dipilih karena termasuk metode yang populer dan dapat membagi ruang vektor menjadi 2 yaitu kelas positif dan kelas negatif. Hal tersebut tidak menutup kemungkinan untuk menggunakan untuk keperluan membagi menjadi lebih dari 2 kelas [14]. Pendekatan yang digunakan menggunakan metode *One-Against-All* (OAA), dan berikut proses M-SVM:

Terdapat 7 kelas yang akan dibuat pada data pelatihan, diantaranya: Bahagia dengan label 1, Sedih dengan label 2, Terkejut dengan label 3, Takut dengan label 4, Marah dengan label 5, Muak dengan label 6 dan Netral dengan label 7. Berikut proses M-SVM:

a. M-SVM dengan label 1 akan diberikan tanda positif (+1) dan label 2, label 3, label 4, dan label 5 diberikan label negatif (-1). Jika hasilnya kelas positif (+1), maka hasil merupakan label 1. Jika hasilnya negatif (-1) maka lanjut ke perulangan b.
b. M-SVM dengan label 2 akan diberikan label positif (+1) dan label 1,label 3, label 4, dan label 5 diberikan label negatif (-1). Jika hasilnya kelas positif (+1), maka hasil merupakan label 1. Jika hasilnya negatif (-1) maka lanjut ke perulangan kelas selanjutnya.

Pada proses pelatihan, digunakan nilai-nilai fitur yang telah didapatkan dari proses ekstrasi fitur sebagai nilai *support vector*.

Tabel 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
Netral	.....	7	-1

Untuk mencari nilai kernel menggunakan *kernel trick Linear*, perhitungan pada kernel dengan menggunakan rumus berikut:

$\mathrm{K}(\mathrm{x_i}, \mathrm{x}) = \mathrm{x_i^T x}$ (17)

$K(\mathrm{y_i}, \mathrm{y}) = y_1^T\, y$ (18)

Untuk mendapatkan kelas dari nilai kernel, jumlahkan $\mathrm{K}(\mathrm{x_i}, \mathrm{x})$ untuk nilai x pada *support vector*, menggunakan rumus berikut:

$x_i = \sum_{j=1}^{n} x_i^T x_j$ (19)

$y_i = \sum_{j=1}^{n} y_i^T y_j$ (20)

Untuk nilai x_i dan y_i yang merupakan nilai *support vector*, selanjutnya menentukan:

a. *Kernel Trick*

Pada nilai x_i dan y_i telah ditemukan, selanjutnya cari nilai kernel *trick* menggunakan rumus berikut:

$$\varphi\binom{x}{y} = \begin{cases} \sqrt{x_n^2 + y_n^2} > 2, maka \begin{bmatrix} \sqrt{x_n^2 + y_n^2} - x & |x - y| \\ \sqrt{x_n^2 + y_n^2} - x & |x - y| \end{bmatrix} \\ \sqrt{x_n^2 + y_n^2} < 2, maka \binom{x}{y} \end{cases} \quad (21)$$

b. Nilai a

Untuk menentukan nilai a dengan menggunakan rumus berikut:

$\sum_{i=1,j=1}^{n} a_i T_i^T T_j$ (22)

c. Nilai Bias

Untuk menentukan nilai bias dengan menambahkan angka 1 pada *Kernel Trick* sebagai berikut:

$$s_i = \begin{matrix} x_i \\ y_i \\ 1 \end{matrix} \quad (23)$$

d. Nilai *Hyperplane*

Kemudian menentukan nilai *hyperplane* menggunakan rumus berikut:

$W = \sum_i^n a_i s_i$ (24)

2.1.2. Proses Pengujian

Proses pengujian untuk menguji setiap citra yang telah dilatih, tahapannya: *preprocessing*, segmentasi *viola and jones*, ekstraksi fitur *markov stationary feature – vector quantization* dan klasifikasi. Dalam proses pengujian, proses *preprocessing*, segmentasi *viola and jones*, ekstraksi fitur *markov stationary feature – vector quantization*, dengan proses yang sama para tahap pelatihan dan berikut adalah tahap klasifikasi pada proses pengujian:

1. Klasifikasi

Proses klasifikasi merupakan proses pengujian terhadap citra latih dengan menggunakan nilai *support vector*, kemudian nilai dari vektor tersebut didistirbusikan menggunakan rumus berikut:

$$kelas\ x = \arg\max_{k=1..7}([w^1]^T.\varphi(x) + b^1, [w^2]^T.\varphi(x) + b^2, [w^3]^T.\varphi(x) + b^3, [w^4]^T.\varphi(x) + b^4, [w^5]^T.\varphi(x) + b^6, [w^6]^T.\varphi(x) + b^6), [w^7]^T.\varphi(x) + b^7 \quad (25)$$

2.2. Pengujain Performansi

Pada engujian performansi menggunakan metode *Confusion Matrix* [13] yang digunakan untuk

mengukur kinerja pada saat klasifikasi. Pengujian dilakukan pada filter 3×3, 5×5 dan 7×7 dalam dua kali proses pengujian diantaranya:

a. Data latih 100% dan data uji 100%, dan
b. Data latih sebanyak 80% dan data uji 20%.

Keterangan dari data yang diujikan sebagai berikut: pengujian pertama dengan menggunakan data latih 100% dan data uji 100%, semua data yang diujikan sudah dilatih sebelumnya sebanyak 2205 data, kemudian pengujian kedua dengan menggunakan data latih 80% sebanyak 1764 data dan data uji sebanyak 20% yang sudah dilatih sebelumnya sebanyak 441 data.

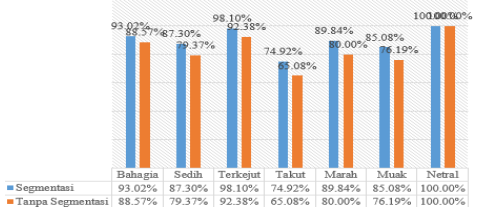
2.2.1. Hasil pengujian

1. Pengujian Pertama

a. Filter 3×3

Tabel 2. *Confusion Matrix* filter 3×3

Sebenarnya	Prediksi							Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	Netral	
Bahagia	293	4	1	14	3	0	0	93.02%
Sedih	14	275	0	11	4	11	0	87.30%
Terkejut	0	3	309	2	0	1	0	98.10%
Takut	23	22	3	236	28	3	0	74.92%
Marah	7	6	2	7	283	10	0	89.84%
Muak	10	11	2	3	21	268	0	85.08%
Netral	0	0	0	0	0	0	315	100.00%
Rata-Rata Akurasi								89.75%

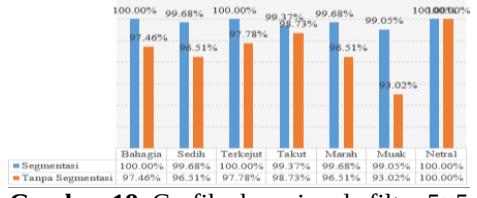


Gambar 17. Grafik akurasi pada filter 3×3

b. Filter 5×5

Tabel 3. *Confusion Matrix* filter 5×5

Sebenarnya	Prediksi							Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	Netral	
Bahagia	293	4	1	14	3	0	0	93.02%
Sedih	14	275	0	11	4	11	0	87.30%
Terkejut	0	3	309	2	0	1	0	98.10%
Takut	23	22	3	236	28	3	0	74.92%
Marah	7	6	2	7	283	10	0	89.84%
Muak	10	11	2	3	21	268	0	85.08%
Netral	0	0	0	0	0	0	315	100.00%
Rata-Rata Akurasi								89.75%

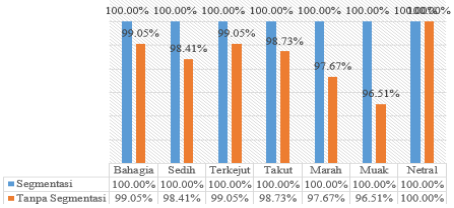


Gambar 18. Grafik akurasi pada filter 5×5

c. Filter 7×7

Tabel 4. *Confusion Matrix* filter 7×7

Sebenarnya	Prediksi							Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	Netral	
Bahagia	315	0	0	0	0	0	0	100.00%
Sedih	0	315	0	0	0	0	0	100.00%
Terkejut	0	0	315	0	0	0	0	100.00%
Takut	0	0	0	315	0	0	0	100.00%
Marah	0	0	0	0	315	0	0	100.00%
Muak	0	0	0	0	0	315	0	100.00%
Netral	0	0	0	0	0	0	315	100.00%
Rata-Rata Akurasi								100.00%



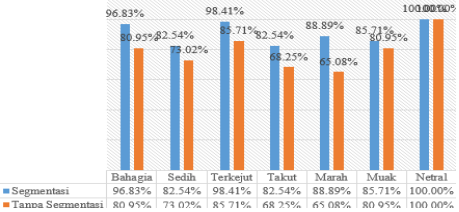
Gambar 19. Grafik akurasi pada filter 7×7

2. Pengujian Kedua

a. Filter 3×3

Tabel 5. *Confusion Matrix* filter 3×3

Sebenarnya	Prediksi							Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	Netral	
Bahagia	61	0	0	2	0	0	0	96.83%
Sedih	2	52	2	3	2	2	0	82.54%
Terkejut	0	1	62	0	0	0	0	98.41%
Takut	0	3	0	52	7	1	0	82.54%
Marah	0	2	1	0	56	4	0	88.89%
Muak	0	4	1	0	4	54	0	85.71%
Netral	0	0	0	0	0	0	63	100.00%
Rata-Rata Akurasi								90.70%

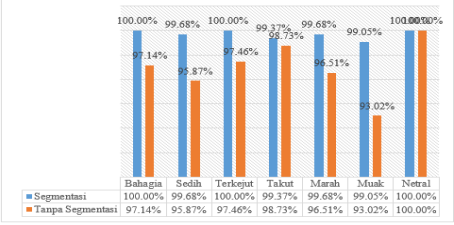


Gambar 20. Grafik akurasi pada filter 3×3

b. Filter 5×5

Tabel 6. *Confusion Matrix* filter 5×5

Sebenarnya	Prediksi							Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	Netral	
Bahagia	63	0	0	0	0	0	0	100.00%
Sedih	0	61	0	1	0	1	0	96.83%
Terkejut	0	0	63	0	0	0	0	100.00%
Takut	0	1	0	62	0	0	0	98.41%
Marah	0	0	0	0	62	1	0	98.41%
Muak	0	0	0	0	0	63	0	100.00%
Netral	0	0	0	0	0	0	63	100.00%
Rata-Rata Akurasi								99.09%

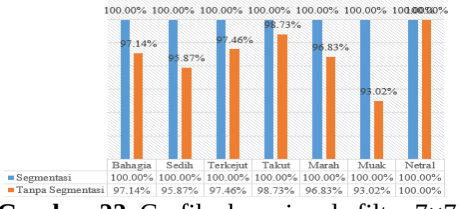


Gambar 21. Grafik akurasi pada filter 5×5

c. Filter 7×7

Tabel 7. *Confusion Matrix* filter 7×7

Sebenarnya	Prediksi							Akurasi
	Bahagia	Sedih	Terkejut	Takut	Marah	Muak	Netral	
Bahagia	63	0	0	0	0	0	0	100.00%
Sedih	0	63	0	0	0	0	0	100.00%
Terkejut	0	0	63	0	0	0	0	100.00%
Takut	0	0	0	63	0	0	0	100.00%
Marah	0	0	0	0	63	0	0	100.00%
Muak	0	0	0	0	0	63	0	100.00%
Netral	0	0	0	0	0	0	63	100.00%
Rata-Rata Akurasi								100.00%



Gambar 22. Grafik akurasi pada filter 7×7

Hasil dari pengujian akurasi dengan menggunkan metode *confusion matrix* pada 3 filter yang berbeda didapatkan data akurasi sebagai berikut:

Tabel 8. Data Akurasi

Metode	Pengujian	Data Training	Data Testing	Akurasi Filter 3×3	Akurasi Filter 5×5	Akurasi Filter 7×7
MSF-VQ+SVM	Pertama	2205	2205	83.08%	96.96%	97.01%
	Kedua	1764	441	79.14%	95.69%	96.60%
Segmentasi+MSF-VQ+SVM	Pertama	2205	2205	89.75%	99.68%	100.00%
	Kedua	1764	441	90.70%	99.09%	100.00%

Dari hasil pengujian yang telah dilakukan dengan menggunakan proses segmentasi, mendapatkan akurasi terbesar pada filter 7×7 dengan nilai akurasi yaitu 100% pada pengujian pertama dan kedua sedangkan tanpa proses segmentasi mendapatkan akurasi terbesar pada filter 7×7 dengan nilai akurasi yaitu 97.01% pada pengujian pertama dan adanya peningkatan dalam proses pendeteksian ekspresi wajah sebesar 2.99% dengan filter yang sama.

2.2.1.1. Hasil Kecepatan

Sedangkan dari perbandingan kecepatan dalam proses pelatihan data, berikut data yang didapatkan berdasarkan waktu pelatihan menggunakan segmentasi dan tanpa segmentasi:

Tabel 9. Data kecepatan

Metode	Pengujian	Data Training	Data Testing	Filter	Waktu Pelatihan	Waktu Pengujian
MSF-VQ+SVM	Pertama	2205	2205	3×3	56 menit	39 menit
				5×5	1 jam 2 menit	45 menit
				7×7	1 jam 15 menit	50 menit
	Kedua	1764	441	3×3	56 menit	15 menit
				5×5	1 jam 2 menit	18 menit
				7×7	1 jam 15 menit	22 menit
Segmentasi+MSF-VQ+SVM	Pertama	2205	2205	3×3	35 menit	29 menit
				5×5	43 menit	31 menit
				7×7	49 menit	34 menit
	Kedua	1764	441	3×3	35 menit	6 menit
				5×5	43 menit	7 menit
				7×7	49 menit	8 menit

3. PENUTUP

Setelah melakukan pengujian terhadap pengaruh segmentasi menggunakan *Viola and Jones* terhadap fitur MSF-VQ pada kasus pengenalan ekspresi wajah bahwa memiliki pengaruh pada kecepatan dalam pelatihan dan pengujian data lebih cepat dan menghasilkan akurasi yang lebih baik dalam proses pengenalan ekspresi wajah yang mendapatkan akurasi sebesar 100%. Dan untuk mengembangkan penelitian ini maka diberikan saran dengan menggunakan metode *Markov Stationary Feature – Vector Quantization*, manfaatkan metode *Markov Stationary Feature – Vector Quantization* pada video untuk mengenali ekspresi wajah secara langsung pada kasus untuk mengenali perasaan seseorang atau *lie detector*.

DAFTAR PUSTAKA

[1] Updadhyay, A., & Kumar, A. (2016). Facial Expression Recognition : A Review. *International Research journal of Engineering and Technology*, 1616-1620.

[2] Dharani S, G. S. (2014). Human Segmentation Using Haar-Classifier . *Journal of Engineering Research and Applications*, 89-93.

[3] Yusron Rijal, R. D. (2008). *Deteksi Wajah Berbasis Segmentasi Model Warna Menggunakan Template Matching Pada Objek Bergerak.* Surabaya: Skripsi S1 / Jurusan Sistem Komputer, Sekolah Tinggi Manajemen Informatika & Teknik Komputer Surabaya.

[4] Douglas Chai, K. N. (1999). Face Segmentation Using Skin-Color Map in Videophone Applications. *IEEE Transactions On Circuits And Systems For Video Technology*, 4-9.

[5] Chandra Kirana, C. K. (2016). Face Identification For Presence Applications Using Violajones and Eigenface Algorithm. *Jurnal SISFOKOM*, 1-5.

[6] Fascal Sapty Jarockohir, I. Maliki. (2018). *Implementasi Markov Stationary Feature – Vector Quantization Untuk Pengenalan Ekspresi Wajah.* Bandung: Skripsi, Fakultas Teknik dan Ilmu Komputer Universitas Komputer Indonesia.

[7] Emilio N.M. Cirillo, M. C. (2015). *Threshold effects in zero range processes.* Netherlands : Centre for Analysis, Scientific computing and Applications Department of Mathematics and Computer Science Eindhoven University of Technology.

[8] Yoseph Linde, Y. L. (1980). An Algorithm for Vector Quantizer Design. *IEEE Transactions On Communications*, 1-28.

[9] Pavani, S.-K., Delgado, D., & F.Frangi, A. (2010). Haar-like features with optimally weighted rectangles for rapid object detection. *Networking Research Center on Bioengineering, Biomaterials and Nanomedicine (CIBER-BBN), Barcelona, Spain*, 160-172.

[10] Athawale, A. K. (2007-2010). *Chapter 6 Information Hiding Using Vector Quantization.* Department of Information & Technology.

[11] Jianguo Li, W. W. (2008). *One Step Beyond Histograms: Image Representation using Markov Stationary Features.* Beijing: Intel China Research Center, Haidian, Beijing, 100190.

[12] Rasband, W. (2018, 12 1). *ImageJ*. Retrieved from ImageJ : Image Processing and Analysis in Java: https://imagej.nih.gov/ij/docs/intro.html

[13] Marina Sokolovaa, G. L. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing and Management*, 427–437.

[14] Nelly Indriani Widiastuti, E. R. (2017). Peringkasan dan Support Vector Machine pada Klasifikasi Dokumen. Jurnal Infotel, 416-421.