IMPLEMENTASI ALGORITMA HIDDEN MARKOV MODEL UNTUK PENGENALAN ISYARAT WAJAH

Roberto Pangihutan Situmeang

Teknik Informatika – Universitas Komputer Indonesia
Jl. Dipatiukur 112-114 Bandung
E-mail : robertopasitumeang2@gmail.com

ABSTRAK

Wajah manusia menyediakan banyak informasi, banyak hal menarik yang bisa diperhatikan, dan dipelajari secara intensif. Manusia berinteraksi satu sama lain menggunakan berbagai macam isyarat wajah. Isyarat dari wajah bisa menyampaikan ekspresi. Penelitian ini dilakukan dengan implementasi metode *Hidden Markov Model* untuk pengenalan isyarat wajah.

Adapun tahapan proses penelitian ini yaitu setiap citra dilakukan praproses yaitu *resize* ukuran 128x128, *grayscale*, deteksi tepi *canny*. Citra *threshold* dibagi menjadi 4 bagian dengan ukuran 64x64. Ekstraksi fitur *Discrete Cousine Transform* (DCT) pada setiap bagian, lalu matriks setiap bagian DCT diubah menjadi vektor. Ekstraksi fitur *Principle Component Analysis* (PCA) menghasilkan matriks 4x4 kemudian matriks tersebut dibentuk menjadi sekuen. Vektor nilai PCA diproses normalisasi. Hasil dari normalisasi tersebut dilakukan proses klasifikasi *Hidden Markov Model* (HMM).

Hasil dari penelitian ini berupa data klasifikasi *training* sebanyak 120 data citra yang digunakan untuk mendeteksi ekspresi wajah secara waktu nyata untuk *testing*, dan menghasilkan *output* jenis ekspresi wajah yang dikenali. Berdasarkan pengujian *Confussion Matrix* menggunakan 166 kali pendeteksian secara waktu nyata maka didapatkan nilai akurasi rata-rata 50% benar dalam klasifikasi dan 50% salah.

Kata kunci: *Hidden Markov Model, Discrete Cousine Transform, Principle Component Analysis*, klasifikasi Citra, pengolahan citra, isyarat wajah, ekspresi.

1 PENDAHULUAN

Wajah merupakan bagian dari tubuh manusia yang menjadi fokus di dalam interaksi sosial. Wajah juga merupakan salah satu bagian unik dari tubuh manusia yang mempunyai karakteristik yang berbeda. Wajah memiliki peranan penting dengan menunjukkan emosi/ekspresi. Wajah manusia menyediakan banyak informasi, banyak hal menarik yang bisa diperhatikan, dan dipelajari secara intensif.

Beberapa penelitian yang menggali informasi dari wajah manusia adalah pengenalan wajah dan pengenalan isyarat wajah. Ketika manusia berinteraksi satu sama lain, mereka menggunakan berbagai macam isyarat dari wajah untuk menyampaikan informasi. Isyarat wajah yang terbentuk bisa menyampaikan ekspresi wajah tertentu.

Pada penelitian mengenai pengenalan ekspresi wajah berdasarkan bentuk dan tekstur, didapat kesimpulan bahwa metode yang digunakan menghasilkan persentase kebenaran dengan nilai persentase yaitu ekspresi senyum 42.86%, biasa 0%, ekspresi cemberut 0%, kedip 28.58%, ekspresi kerling 42.86%, ekspresi terkejut 28.58%, sehingga penelitian mengenai pengenalan ekspresi wajah berdasarkan bentuk dan tekstur masih mungkin untuk dilakukan selanjutnya menggunakan metode lain, pada penelitian tersebut menggunakan metode SMOM untuk menggambarkan ekspresi dan metode ESTM untuk mencari kesamaan citra [1].

Berdasarkan penelitian tersebut, maka penelitian ini dilakukan dengan implementasi metode *Hidden Markov Model* (HMM) untuk pengenalan isyarat wajah (*Face Gesture*). Dikarenakan pada penelitian mengenai *Hand Gesture Recognition using a real-time tracking method and Hidden Markov Models*, yang menyimpulkan bahwa metode Hidden Markov Model (HMM) baik digunakan dalam mengenali gerakan tangan (Hand Gesture) [2].

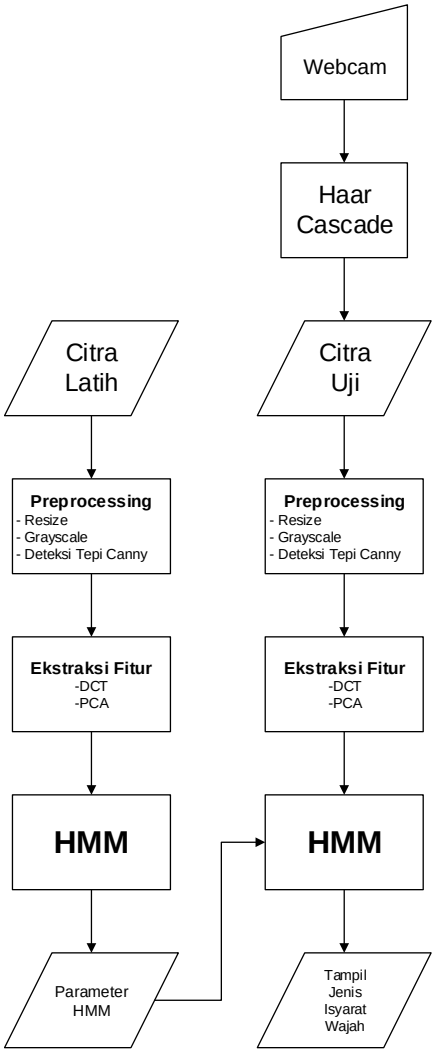
Hidden Markov Model merupakan pemodelan probabilitas suatu sistem dengan mencari parameter-parameter markov yang tidak diketahui untuk memperoleh analisis sistem tersebut. Metode *Hidden Markov Model* (HMM) mampu menangani perubahan statistik dari gambar, dengan memodelkan elemen-elemen menggunakan probabilitas. Salah satu aplikasinya adalah pada *image processing* [3].

Dalam penelitian ini dilakukan pengenalan isyarat wajah (*face gesture*). Secara umum penelitian ini dirancang dalam beberapa tahapan yaitu pendeteksian wajah, ekstraksi wajah, dan pengenalan isyarat wajah. Pendeteksian wajah adalah tahap untuk menentukan dan memisahkan bagian dari suatu citra diam yang terdefinisi sebagai wajah menggunakan metode *Haar cascade*. Ekstraksi wajah adalah tahap untuk pengambilan bagian dari fitur-fitur wajah. Pengenalan isyarat wajah adalah tahapan dalam pengenalan isyarat wajah menggunakan metode *Hidden Markov Model* (HMM).

2 ISI PENELITIAN

2.1 Gambaran Umum Sistem

Tujuan utama dari tugas akhir ini adalah untuk melihat apakah metode *Hidden Markov Model* dapat mengenali isyarat wajah secara waktu nyata ditinjau dari seberapa akurat isyarat wajah dikenali dengan membangun sistem yang dapat melakukan pengenalan gerak isyarat wajah. Proses diagram alur arsitektur sistem untuk data latih dan alur arsitektur pengujian sistem, dapat dilihat pada gambar 1 sebagai berikut:



Gambar 1. Alur Arsitektur Sistem

Pada alur arsitektur sistem untuk citra latih dijelaskan terdapat beberapa tahapan yaitu data latih isyarat wajah dilakukan *preprocessing*, yaitu *Resize* citra wajah dengan ukuran 128x128, lalu citra dilakukan proses *Grayscale*, kemudian proses Deteksi Tepi *Canny*, selanjutnya hasil deteksi tepi canny dilakukan ekstraksi fitur DCT dan PCA selanjutnya diproses pada algoritma HMM, hasil algoritma HMM berupa parameter, yang kemudian parameter HMM disimpan.

Pada alur arsitektur sistem citra uji dijelaskan terdapat beberapa tahapan yaitu deteksi wajah dan menangkap citra yang didefinisikan sebagai wajah menggunakan metode *Haar Cascade* melalui *webcam* secara waktu nyata, kemudian citra tersebut menjadi data uji. Tahap selanjutnya data uji masuk kedalam proses *preprocessing*, selanjutnya dilakukan ekstraksi fitur, kemudian diproses menggunakan algoritma HMM dengan mengambil parameter data training sebagai acuan pengenalan jenis isyarat wajah, lalu sistem menampilkan jenis dari isyarat wajah.

2.2 Data Masukan

Data masukan yang akan diambil pada sistem ini merupakan citra yang diambil saat terjadi gerak pada wajah. Sehingga jelas citra merupakan sebuah gerak isyarat, dan dalam penelitian ini menentukan kategori jenis gerakan yang terdapat pada citra.

Setiap proses pada proses pengolahan citra menggunakan nilai pixel dari citra untuk perhitungannya. Setiap pixel pada citra terdiri dari 3 unsur warna dasar yaitu *Red* (R), *Green* (G) dan *Blue* (B), yang sering disebut RGB, dan setiap unsur memiliki nilai antara 0-255 yang menjadi kode kedalaman warna masing-masing unsur. Sebagai contoh perhitungan pada proses-proses pengolahan citra, digunakan gambar wajah marah dan bahagia.



Gambar 2. Wajah “Biasa”



Gambar 3. Wajah “Marah”



Gambar 4. Wajah “Bahagia”



Gambar 5. Wajah “Sedih”



Gambar 6. Wajah “Muak”



Gambar 7. Wajah “Terkejut”

2.2.1 Preprocessing

Untuk melakukan proses perhitungan matriks citra, contoh menghitung menggunakan gambar 2 yaitu citra “Gambar Wajah Biasa” (dimensi 345 x 345 pixel) yang akan dijadikan matriks terlebih dahulu. Kemudian akan menghasilkan matriks pembentuk citra yang diubah menjadi matriks yang memiliki 3 matriks warna, yaitu *Red, Green* dan *Blue.*

1. Resize

Setelah dilakukan pengambilan matriks dari citra, maka dilakukan *resize*. *Resize* bisa disebut tahap normalisasi dimensi citra yaitu merupakan proses merubah ukuran citra menjadi 128x128 pixel. Tujuannya, untuk menyamakan dimensi citra dari setiap citra yang inputan, sehingga pada proses ekstraksi citra tidak ada perbedaaan dimensi dari matriks data citra wajah tersebut. Metode yang digunakan pada resize adalah metode *bicubic interpolation* [4].

2. Grayscale

Grayscale merupakan citra yang mempunyai nilai dari setiap pixel sampel tunggal. Citra yang ditampilkan yaitu warna abu-abu, variasi warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Kemudian analisis yang dilakukan yaitu membuat matriks *grayscale* menggunakan rumus *Luminosity* [5].

$$Grayscale = 0.2989 \times R + 0.5870 \times G + 0.1141 \times B \quad (1)$$

Contoh perhitungan manual grayscale dapat dilihat pada perhitungan *grayscale*[1,1] berikut:

$$grayscale[1,1] = (0.2989\ x\ R) + (0.5870\ x\ G) + (0.1141\ x\ B)$$

$$grayscale[1,1] = (0.2989\ x\ 253) + (0.5870\ x\ 246) + (0.1141\ x\ 254)$$

$$grayscale[1,1] = (75.6217) + (144.402) + (28.9814)$$

$$grayscale[1,1] = 249.0051$$

Kemudian dibulatkan menjadi 249.

3. Deteksi Tepi Canny

Langkah pertama pada deteksi tepi *canny* adalah perkalian hasil matriks *grayscale* dengan tapis *Gaussian* dengan tujuan menghilangkan derau yang ada pada citra dengan mengimplementasikan konvolusi tapis *Gaussian* dengan persamaan [5]:

Gaussian =

$$\begin{bmatrix} 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \\ 0.0261 & 0.0561 & 0.0721 & 0.0561 & 0.0261 \\ 0.0337 & 0.0721 & 0.0935 & 0.0721 & 0.0337 \\ 0.0261 & 0.0561 & 0.0721 & 0.0561 & 0.0261 \\ 0.0121 & 0.0261 & 0.0337 & 0.0261 & 0.0121 \end{bmatrix}.$$

Grayscale (2)

Cara menghitung *gaussian* dapat dijelaskan pada contoh *Gaussian*[1,1] sebagai berikut [7]:

Gaussian[1,1] =(0.0935x249 + 0.0721x248 + 0.0337x249 + 0.0721x249 + 0.0561x248 + 0.0261x250 + 0.0337x249 + 0.0261x248 + 0.0121x250)

Gaussian[1,1] = 106.0199
Kemudian dibulatkan menjadi 106.

Langkah berikutnya proses deteksi tepi ini dijalankan dengan menggunakan operator *sobel* yang digunakan untuk menghitung perbedaan warna pada pixel yang sedang dihitung dengan pixel sekitarnya secara horizontal (*Gx*) dan vertikal (*Gy*). Berikut rumus *Gx* [5]:

Gx = Sobel *Gx* . Hasil *Gaussian* (3)

Berikut persamaan sobel Gx [7]:

$$Sobel\ Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (4)$$

Cara menghitung *Sobel Gx* tersebut dapat dijelaskan pada contoh *Sobel Gx*[1,1] dimana hasil dari perhitungan tersebut berikut:

Gx[1,1] = (2 x 145) + (1 x 106)
Gx[1,1] = 487

Dan berikut rumus *Gy*:

Gy = Sobel *Gy*. Hasil *Gaussian* (5)

Berikut persamaan sobel Gy:

$$\text{Sobel } Gy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (6)$$

Cara menghitung *Sobel Gy* tersebut dapat dijelaskan pada contoh *Sobel Gy*[1,1] dimana hasil dari perhitungan tersebut berikut:

Gy (1,1) = ((-2) x 144) + ((-1) x 197)

Gy (1,1) = -485

Setelah diketahui semua hasil dari *Gx* dan *Gy*, maka bisa dihitung langkah berikutnya yaitu menentukan *magnitudo gradien* (kekuatan tepi).

Untuk menghitung nilai setiap koordinat $|G|_{(x,y)}$ maka diambil nilai dari masing-masing koordinat $Gx_{(x,y)}$ dan nilai $Gy_{(x,y)}$.

Contoh perhitungan untuk nilai $|G|_{(1,1)}$, maka diambil nilai $Gx_{(1,1)}$ =487, dan nilai $Gy_{(1,1)}$ = (-485). Sehingga contoh perhitungannya dapat dilihat sebagai berikut [7]:

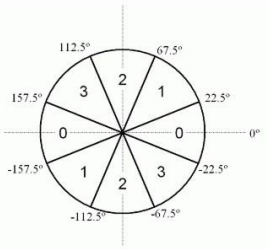
$$|G|_{(1,1)} = \sqrt{(487)^2 + (-485)^2} = \sqrt{237{,}169 + 235{,}225} = \sqrt{472{,}394}$$

$|G|_{(1,1)}$ = 687,3092, kemudian dibulatkan menjadi 687.

Maka langkah selanjutnya adalah menentukan arah tepi dengan menggunakan persamaan berikut:

$$\theta = \arctan\left(\frac{Gy}{Gx}\right) \quad (7)$$

Kemudian untuk menentukan sudut gradien dengan membagi kedalam 4 arah yang dapat dilihat pada gambar 8 berikut [7]:



Gambar 8. Sudut gradien

Pembagian kedalam 4 arah tersebut adalah sebagai berikut:

1. Jika θ>0 dan θ<22.5 atau θ>157.5 dan θ< -157.5 maka sudut gradien 0.
2. Jika θ>22.50 dan θ<67.5 atau θ> -157.5 dan θ< -112.5 maka sudut gradien 45.
3. Jika θ>67.50 dan θ<112.5 atau θ> -112.5 dan θ< -67.5 maka sudut gradien 90.
4. Jika θ>112.50 dan θ<157.5 atau θ> -67.5 dan θ< -22.5 maka sudut gradien 135.

Pada contoh perhitungan berikut menggunakan $Gx_{(1,1)}$ dan $Gy_{(1,1)}$ sebagai berikut:

$$\theta = \tan^{-1}\left(\frac{-485}{487}\right) = \tan^{-1}(-0.99589322) = -44.8821$$

Maka sudut gradien $= 135°$

Kemudian penghapusan *non-maximum suppression* (tekanan non-maksimum) dengan langkah sebagai berikut [7]:

1. Untuk setiap piksel (x,y), tentukan arah gradien terdekat sesuai dengan arah 8 ketetanggan. Dimulai dari nilai magnitudo gradien (2, 2) agar memiliki arah 8 ketetanggaan. Selanjutnya magnitude gradien disebut mg.
2. Untuk sudut gradien = 0, jika mg(x, y) > mg(x, y - 1) dan mg(x, y) > mg(x, y + 1), maka nilai *non-maximum suppression* (x, y) adalah mg(x, y). Jika tidak, maka nilai *non-maximum suppression* (x, y) adalah 0.
3. Untuk sudut gradien = 45, jika mg(x, y) > mg(x + 1, y - 1) dan mg(x, y) > mg(x - 1, y + 1), maka nilai *non-maximum suppression* (x, y) adalah mg(x,y). Jika tidak, maka nilai *non-maximum suppression* (x, y) adalah 0.
4. Untuk sudut gradien = 90, jika mg(x, y) > mg(x - 1, y) dan mg(x, y) > mg(x + 1, y), maka nilai *non-maximum suppression* (x, y) adalah mg(x, y). Jika tidak, maka nilai *non-maximum suppression* (x, y) adalah 0.
5. Untuk sudut gradien = 135, jika mg(x, y) > mg(x - 1, y - 1) dan mg(x, y) > mg(x + 1, y + 1), maka nilai *non-maximum suppression* (x, y) adalah mg(x, y).
6. Jika tidak, maka nilai *non-maximum suppression* (x, y) adalah 0.

Kemudian dapat dilakukan *Thresholding* atau pengambangan, yang merupakan proses pemisahan piksel-piksel berdasarkan derajat keabuan lebih kecil dari nilai *threshold* yang dimilikinya. Pada perhitungan penelitian ini menggunakan metode *otsu* untuk menentukan nilai *threshold*, yang akan menghasilkan sebuah nilai. Kemudian nilai *threshold* ini digunakan untuk mengkonversi data image menjadi data biner dengan tujuan agar proses selanjutnya menjadi lebih mudah.

Prinsip metode *Otsu* dijelaskan sebagai berikut ini. Pertama-tama, probabilitas nilai intensitas *i* dalam histogram dihitung melalui persamaan 8 berikut [4]:

$$p(i) = \frac{n_i}{N}, p(i) \geq 0, \sum_{i=1}^{t} p(i) = 1 \quad (8)$$

Dengan n_i menyatakan jumlah piksel berintensitas i dan N menyatakan, jumlah semua piksel dalam citra. Jika histogram dibagi menjadi 2 kelas (objek dan latar belakang), pembobotan pada kedua kelas dinyatakan pada masing-masing persamaan 9 dan 10 berikut:

$$w_1(t) = \sum_{i=1}^{t} p(i) \quad (9)$$

$$w_2(t) = \sum_{i=t+1}^{L} p(i) = 1 - w_1(t) \quad (10)$$

Dalam hal ini, L menyatakan jumlah aras keabuan. Rerata kedua kelas dihitung melalui persamaan 11 dan persamaan 12 berikut:

$$m_1(t) = \sum_{i=1}^{t} i.\, p(i)/w_1(t) \quad (11)$$

$$m_2(t) = \sum_{i=1}^{t} i.\, p(i)/w_2(t) \quad (12)$$

Varian kedua kelas dinyatakan dengan persamaan 13 dan persamaan 14 berikut:

$$\sigma_1{}^2(t) = \sum_{i=1}^{t} (1 - m_1)^2 . \frac{p(i)}{w_1(t)} \quad (13)$$

$$\sigma_2{}^2(t) = \sum_{i=t+1}^{L} (1 - m_2)^2 . \frac{p(i)}{w_2(t)} \quad (14)$$

Varians total dapat dinyatakan dengan persamaan 15 berikut:

$$\sigma^2(t) = \sigma_W{}^2(t) + \sigma_B{}^2(t) \quad (15)$$

Dalam hal ini, $\sigma_w{}^2$ dinamakan sebagai *Within-Class Variance* (WCV) dan $\sigma_B{}^2$ disebut *Between-Class Variance* (BCV). WCV dapat dinyatakan dengan persamaan 16 berikut:

$$\sigma_W{}^2(t) = W_1(t) . \sigma_1(t)^2 + W_2(t) . \sigma_2(t)^2 \quad (16)$$

Rumus diatas menunjukkan bahwa wcv adalah jumlah varians kelas secara individual yang telah diboboti dengan probabilitas kelas masing-masing. Adapun BCV dinyatakan dengan persamaan 17 berikut:

$$\sigma_B{}^2(t) = W_1 . [m_1(t) - m_T]^2 + W_2 . [m_2(t) - m_T]^2 \quad (17)$$

Dalam hal ini, m_T adalah rerata total ($m_T = \sum_{i=1}^{N} i.\, P(i)$).

Nilai ambang optimum dapat diperoleh dengan 2 cara. Cara pertama dilakukan dengan meminimumkan WCV. Cara kedua dilakukan dengan memaksimumkan BCV. Namun, berdasarkan kedua cara tersebut, cara yang kedua lebih menghemat komputasi.

4. Ekstraksi Fitur

Tahap selanjutnya adalah ekstraksi fitur dari citra. Tujuan ekstraksi fitur ini untuk mendapatkan nilai ciri dari citra tersebut. Pada penelitian ini menggunakan dua ekstraksi fitur, ekstraksi fitur yang pertama adalah *Discrete Cosine Transform* (DCT), kemudian nilai yang dihasilkan pada ekstraksi fitur diproses dengan ekstraksi fitur *Principal Component Analysis* (PCA) [8].

Sebelum proses DCT, dilakukan pembagian gambar menjadi *sub*-gambar dengan ukuran 64x64, maka terbentuk 4 matriks *sub*-gambar, selanjutnya dilakukan proses DCT. Kemudian dapat dilakukan ekstraksi fitur atau ekstraksi ciri dengan transformasi cosinus diskret [9].

Kemudian matriks hasil DCT diubah menjadi vektor dengan menyusun matriks menjadi 1 baris kebawah. Hasil ekstraksi fitur DCT tiap bagian selanjutnya digabung hingga berbentuk sebuah matriks. Kemudian matriks DCT akan dihitung dengan ekstraksi fitur *Principal Component Analysis* (PCA). maka diambil nilai matriks koefisien PCA dari variable COEFF [10], seperti berikut:

Koefisien PCA

$$\begin{bmatrix} 0.2714 & 0.9199 & -0.1679 & -0.2280 \\ 0.1034 & 0.0997 & 0.9712 & -0.1900 \\ 0.9540 & -0.2915 & -0.0695 & 0.0107 \\ 0.0747 & 0.2427 & 0.1539 & 0.9549 \end{bmatrix}$$

Kemudian koeffisien PCA tersebut akan disusun hingga membentuk vektor, seperti sebagai berikut:
Koefisien PCA =

$$[0.2714 \quad 0.1034 \quad 0.9540 \quad \cdots \quad 0.9549]$$

2.2.2 Permodelan, *Sequence* Dan Normalisasi Data Training

Proses pemodelan adalah proses pemberian tanda/simbol pada citra latih ekspresi wajah yang akan dilakukan pelatihan, contoh permodelan dapat dilihat pada table 1 berikut:

Tabel 1. Tabel Pemodelan Data Latih

Ekspresi Citra	Simbol
Biasa	N
Bahagia	B
Marah	H
Muak	M
Sedih	S
Terkejut	T

Sedangkan proses *sequence* data training adalah proses pengubahan matriks PCA dari setiap citra latih ekspresi wajah menjadi berbentuk vektor. Proses *sequence* data latih dilakukan bersamaan dengan pembentukan label data *training*. Pada penelitian ini menggunakan 120 *dataset* dengan 20 *dataset* untuk setiap jenis ekspresi.

Selanjutnya *sequences* data latih nilai PCA dilakukan pengindeksan (normalisasi) untuk semua data latih. Proses ini dilakukan karena pada proses *sequences* HMM hanya menerima nilai bilangan bulat non-negatif. Adapun cara normalisasi yang digunakan adalah sebagai berikut [11]:

1. Panjang interval setiap indeks = (nilai maksimum – nilai minimum) / jumlah indeks.
2. Kemudian setiap nilai *sequences* pada nilai PCA disusun kedalam indeks sesuai dengan indeks yang telah dibuat.
3. Selanjutnya setiap nilai *sequences* yang telah disusun dituliskan kedalam table *sequences*.

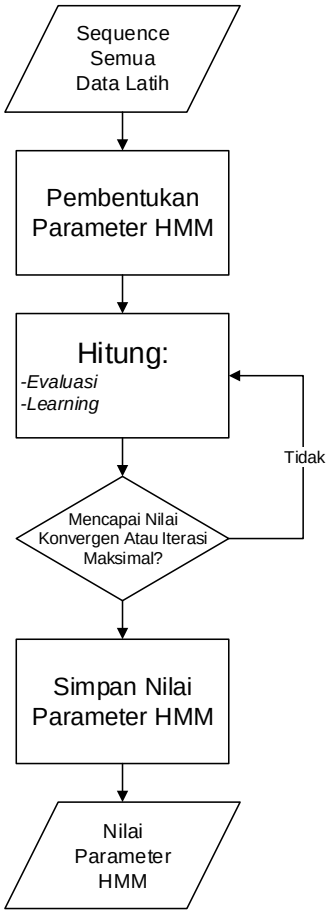
Kemudian nilai hasil normalisasi digunakan sebagai observasi dalam HMM.

2.2.3 Hidden Markov Model

Model Markov Tersembunyi atau lebih dikenal sebagai *Hidden Markov Model* (HMM) adalah sebuah model statistik dari sebuah sistem yang diasumsikan sebuah Proses Markov dengan parameter yang tak diketahui, dan tantangannya adalah menentukan parameter-parameter tersembunyi (*state*) dari parameter-parameter yang dapat diamati (*observer*). Parameter-parameter yang ditentukan kemudian dapat digunakan untuk analisis yang lebih jauh, misalnya untuk aplikasi pengenalan pola (*Pattern Recognition*). Sebuah HMM dapat dianggap sebagai sebuah *Bayesian Network* dinamis yang paling sederhana.

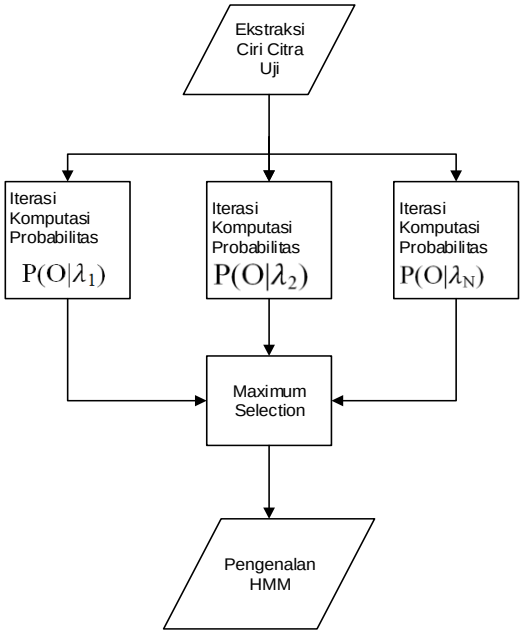
Pada model Markov umum (*Vanilla/Visible Markov Model*), s*tate*-nya langsung dapat diamati, oleh karena itu probabilitas transisi keadaan (*state)* menjadi satu-satunya parameter. Di dalam Model Markov yang tersembunyi, *state*-nya tidak dapat diamati secara langsung, akan tetapi yang dapat diamati adalah variabel-variabel yang terpengaruh oleh *state*. Setiap *state* memiliki distribusi probabilitas atas token-token output yang mungkin muncul. Oleh karena itu rangkaian token yang dihasilkan oleh HMM memberikan sebagian informasi tentang sekuens state-state [12].

Berikut merupakan gambar alur flowchart HMM untuk pelatihan dapat dilihat pada gambar 9.



Gambar 9 Flowchart pelatihan HMM

Berikut merupakan gambar alur flowchart HMM untuk pengujian dapat dilihat pada gambar 10.



Gambar 10 Flowchart pengujian HMM.

2.2.3.1 Pembentukan Parameter HMM

HMM memiliki parameter utama yaitu 3 parameter sudah didapatkan maka HMM dapat dituliskan dalam bentuk $\lambda = (N, M, A, B, \pi)$, parameter tersebut dapat dituliskan dalam bentuk probabilitas yang akan dianalisis [13].
Dimana:
M = jumlah state = 6
N = Panjang Sequence = 16

$A_{MM} = Transisi = \frac{1}{M}, \forall\, m, m = \frac{1}{6} = 0.16667$

$B_{MN} = Emisi = \frac{1}{N}, \forall\, m, m = \frac{1}{16} = 0.0625$

$\pi_0 = 1$

1. Parameter π

Merupakan probabilitas kemunculan suatu state di awal. Parameter π dituliskan sebagai berikut ini:

$\pi_0 = 1$

2. Parameter A

Merupakan probabilitas kedudukan suatu state terhadap semua state yang ada. Probabilitas transisi awal sebelum dilakukan iterasi terhadap data latih setiap ekspresi akan bernilai sama. Adapun perhitungan probabilitas transisi awal dapat dilihat seperti berikut:

$$A(MM) = \begin{bmatrix} 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \\ 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 & 0.1667 \end{bmatrix}$$

3. Parameter B

Merupakan probabilitas kemunculan suatu state dalam deretan seluruh state yang ada. Probabilitas awal dalam HMM dapat dituliskan dalam bentuk matriks, maka dapat ditentukan oleh persamaan sebagai berikut:

$$B(M,N) = \begin{bmatrix} 0.0625 & 0.0625 & 0.0625 & 0.0625 & 0.0625 & \dots & 0.0625 & 0.0625 \\ 0.0625 & 0.0625 & 0.0625 & 0.0625 & 0.0625 & \dots & 0.0625 & 0.0625 \\ 0.0625 & 0.0625 & 0.0625 & 0.0625 & 0.0625 & \dots & 0.0625 & 0.0625 \\ 0.0625 & 0.0625 & 0.0625 & 0.0625 & 0.0625 & \dots & 0.0625 & 0.0625 \\ 0.0625 & 0.0625 & 0.0625 & 0.0625 & 0.0625 & \dots & 0.0625 & 0.0625 \\ 0.0625 & 0.0625 & 0.0625 & 0.0625 & 0.0625 & \dots & 0.0625 & 0.0625 \end{bmatrix}$$

2.2.3.2 Training HMM

Agar HMM dapat dilakukan pelatihan data, ada 2 masalah mendasar dalam HMM yang harus diselesaikan, yakni [14]:

1. Evaluasi

Yakni akan mencari *P(O|λ)* atau peluang dari barisan observasi jika diberikan HMM : *λ*= (A,B,π). Peluang ini dapat ditentukan secara induksi dengan menggunakan algoritma *forward*.Secara induktif *P(O|λ)* dapat dihitung sebagai berikut [15]:

a. Inisialisasi

Untuk state (t) =1, dimana termasuk observasi(1) = 1

$\alpha_i(m) = \pi(m) b_m(O_1)$ (18)

$\alpha_1(1) = \pi(1) b_1(O_1) = (1)(0.0625) = 0.0625$
$\alpha_2(1) = \pi(1) b_1(O_1) = (1)(0.0625) = 0.0625$
$\alpha_3(1) = \pi(1) b_1(O_1) = (1)(0.0625) = 0.0625$
$\alpha_4(1) = \pi(1) b_1(O_1) = (1)(0.0625) = 0.0625$
$\alpha_5(1) = \pi(1) b_1(O_1) = (1)(0.0625) = 0.0625$
$\alpha_6(1) = \pi(1) b_1(O_1) = (1)(0.0625) = 0.0625$

b. Induksi

Pada tahap ini akan dihitung $\alpha_2(j) dan\ \alpha_3(j)$
Probabilitas pada t = 2 sebagai berikut :

$\alpha_{i+1}(j) = \left[\sum_{i=1}^{n} \alpha_1(i)\alpha_{ij}\right] b_j(O_2)$ (19)

$\alpha_2(1) = \left[(\alpha_1(1)\alpha_{12}) + (\alpha_1(2)\alpha_{12}) + (\alpha_1(3)\alpha_{12}) + (\alpha_1(4)\alpha_{12}) + \left(\alpha_1(5)\alpha_{12} + (\alpha_1(6)\alpha_{12})\right)\right] b_1(O_2)$

$\alpha_2(j) = [(0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667)](0.0625)$

$\alpha_2(j) = [(0.01041875) + (0.01041875) + (0.01041875) + (0.01041875) + (0.01041875) + (0.01041875)](0.0625)$

$\alpha_2(j) = [(0.0625125)](0.0625) = 0.003907$

Probabilitas pada t = 3 sebagai berikut :

$\alpha_3(j) = [(0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667)](0.0625)$

$\alpha_3(j) = [(0.01041875) + (0.01041875) + (0.01041875) + (0.01041875) + (0.01041875) + (0.01041875)](0.0625)$

$\alpha_3(j) = [(0.0625125)](0.0625) = 0.003907$

⋮

Hingga probabilitas t = 6 sebagai berikut:

$\alpha_6(j) = [(0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667) + (0.0625)(0.1667)](0.0625)$

$\alpha_6(j) = [(0.01041875) + (0.01041875) + (0.01041875) + (0.01041875) + (0.01041875) + (0.01041875)](0.0625)$

$\alpha_6(j) = [(0.0625125)](0.0625) = 0.003907$

c. Terminasi

Dari tahap terminasi maka algoritma menghasilkan penyelesaian sebagai berikut :

$P(O \mid \lambda) = \sum_{i=1}^{n} a_T(i)$ (20)

$P(O \mid \lambda) = a_1(1) + a_2(1) + a_3(1) + a_4(1) + a_5(1) + a_6(1) + a_1(2) + .. + a_5(6) + a_6(6)$

$P(O \mid \lambda) = 0.140653125$

2. Learning

Untuk menyelesaikan masalah ini digunakan Algoritma Baum-Welch. Dalam algoritma Baum-Welch, didefinisikan tiga variabel, yaitu variabel forward, variabel backward, dan variabel ξ(i, j). Variabel forward dan variabel backward akan digunakan dalam perhitungan variabel ξ(i, j) dan variabel $\gamma_t(i)$.

Variabel pertama, variabel *forward* telah didefinisikan pada persamaan (18), serta tahapan induksi diberikan oleh persamaan (19).

Variabel kedua, yakni variabel *backward* didefinisikan sebagai, $\beta_t(i)$= P($O_{t+1}, O_{t+2,\ldots,} O_T|Q_t = S_i, \lambda$) yaitu, peluang dari barisan observasi parsial $O = O_1,O_2,\ldots, O_T$, diberikan *state* S_i pada waktu t dan model λ. Selanjutnya, $\beta_t(i)$ dapat diselesaikan secara induksi sebagai berikut [14]:

a. Inisialisasi

$\beta_T(i)$=1 (21)

b. Induksi

$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(O_{t+1})\beta_{t+1}(j), \quad t = T-1, T-2, \ldots, 1,\ 1 \le j \le N$ (22)

Variabel ketiga, yakni variabel $\xi(i,j)$ didefinisikan sebagai,

ξ(i,j)= P($Q_t = S_i\ , Q_{t+1} = S_{j,} O|\lambda$) (23)

yaitu, peluang proses pada saat t berada pada state S_i dan pada saat t+1 berada pada state S_j, jika diberikan barisan O dan model λ.

Persamaan (24) dapat ditulis:

$$\xi(i,j)= \frac{P(Q_t = S_i, Q_{t+1} = S_j, O|\lambda)}{P(O|\lambda)} \quad (24)$$

atau dapat diekspresikan dengan variabel forward-backward sebagai:

$$\xi_t(i,j) = P(q_t = S_i, q_{t+1} = S_j \mid O,\lambda) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(i)}{\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(i)} \quad (25)$$

2.2.3.3 Pendeteksian Wajah (Haar Cascade Classifier)

Metode yang digunakan dalam penelitian ini untuk mendeteksi keberadaan wajah yang berasal dari webcam adalah *Haar Cascade Classifier*. Dimana fungsi detektor wajah akan menerima masing masing lokasi *image* guna mengklasifikasikan area wajah atau bukan area wajah.

Pengklasifikasi *Haar Cascade* tersimpan pada file dengan ekstensi XML yang berfungsi untuk memutuskan bagaimana pengklasifikasian masing-masing lokasi keberadaan wajah. Pengklasifikasi yang bisa di *download* ada beberapa macam, diantaranya: pengklasifikasian wajah tambak depan (*frontal face*), klasifikasi profil wajah (*profile face*), pengklasifikasi seluruh tubuh (*full body/pedestrian*), pengklasifikasian untuk tubuh bagian atas (*upper body*) dan pengklasifikasian tubuh bagian bawah (*lower body*).

Pada *OpenCV*, pengklasifikasi *Haar Cascade Classifier* untuk mendeteksi wajah tampak depan ada beberapa pilihan yang bias dimanfaatkan, yaitu [16]:

1. *"haarcascade_frontalface_default.xml"*
2. *"haarcascade_frontalface_alt.xml"*
3. *"haarcascade_frontalface_alt2.xml"*
4. *"haarcascade_frontalface_alt_tree.xml"*

Keempat klasifikasi tersebut akan memberikan hasil deteksi yang berbeda-beda, tergantung program dan kondisi lingkungan sekitarnya.
Pada penelitian ini digunakan "*Haarcascade_frontalface_alt_tree.xml*".

2.2.3.4 Pengenalan Ekspresi Wajah

Setelah proses pembuatan *training* model HMM dilakukan, selanjutnya adalah proses pengenalan ekspresi secara waktu nyata. Pengenalan ekspresi wajah diselesaikan dengan *decoding*.

Akan dicari barisan state yang optimal Q^* $=\{Q_1^*,Q_2^*,\ldots,Q_T^*\}$ jika diberikan barisan observasi O $=\{O_1, O_2,\ldots,O_T\}$ dan model λ = (A B, ,π). Barisan state terbaik yang akan ditentukan yaitu berupa lintasan tunggal yang terhubung dari *t =1,2, ...,T*.

Untuk menyelesaikan masalah *decoding* ini digunakan algoritma Viterbi. Langkah-langkah dalam algoritma Viterbi untuk menentukan barisan state terbaik yaitu [14]:

a. Inisialisasi

$$\delta(i) = \pi_i . b_i(O_1) \quad (26)$$
$$\psi_t(i) = 0$$

b. Rekursi

$$\delta_t(j) = \max_{1\le i\le N}\left[\delta_{t-1}(i).a_{ij}\right].b_j(O_t), \quad 2 \le t \le T,\ 1 \le j \le N \quad (27)$$

$$\psi_t(j) = \arg\max_{1\le i\le N}\left[\delta_{t-1}(i).a_{ij}\right], \quad 2 \le t \le T,\ 1 \le j \le N \quad (28)$$

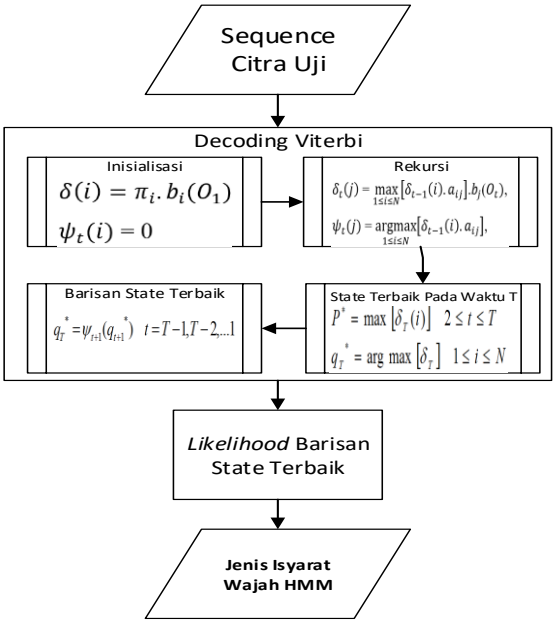
1. *State* terbaik pada waktu T (Q_T)

$$P^* = \max\left[\delta_T(i)\right] \quad 2 \le t \le T \quad (29)$$
$$q_T{}^* = \arg\max\left[\delta_T\right] \quad 1 \le i \le N \quad (30)$$

2. Barisan *state* terbaik pada *t*=*T*-1,*T*-2,….,1

$$q_T{}^* = \psi_{t+1}(q_{t+1}{}^*) \quad t = T-1, T-2, \ldots 1 \quad (31)$$

Langkah awal wajah dideteksi menggunakan *Haar Cascade*, kemudian dilakukan *image preprocessing* dan ekstraksi ciri, selanjutnya hasilnya dilakukan proses HMM Adapun alur pengenalan ekspresi wajah dapat dilihat pada gambar 11 berikut:



Gambar 11. Alur Pengenalan Ekspresi Wajah

2.3 Pengujian Aplikasi

Pengujian aplikasi merupakan tahapan yang memiliki tujuan untuk menemukan kekurangan pada perangkat lunak yang diuji. Pengujian bermaksud untuk mengetahui perangkat lunak yang dibuat sudah memenuhi kriteria yang sesuai dengan tujuan perancangan dan pengujian perangkat lunak ini menggunakan pengujian *black box*. Pengujian *black box* berfokus pada persyaratan fungsional perangkat lunak, dan juga pengujian terhadap performansi yang dilakukan untuk menghasil nilai pengujian terhadap klasifikasi.

2.3.1 Skenario Pengujian Data

Berikut ini adalah scenario pengujian seperti terlihat pada tabel dibawah ini:

Tabel 2. Skenario Pengujian Data

Kelas Uji	Butir Uji	Jenis Pengujian
Gambar	Menguji gambar masukan	*Blackbox*
Pengelolaan data *training*	Mengelola data dari citra latih	*Blackbox*

2.3.2 Skenario Fungsional

Skenario fungsional menjelaskan tentang pengujian terhadap fungsional sistem yang digunakan. Pengujian Performansi.

Pengujian performansi digunakan untuk mengetahui akurasi yang dilakukan menggunakan metode *Confussion Matrix*. *Confussion Matrix* merupakan suatu table yang memvisualisasikan kinerja dari suatu algoritma. Setiap kolom dalam matriks merepresentasikan kelas yang diprediksikan, sedangkan setiap baris merepresentasikan kelas yang sebenarnya [17].

Berikut ini merupakan tabel pengujian performansi metode Hidden Markov Model pada pengenalan isyarat wajah menggunakan *confussion matrix*.

Tabel 3. Pengujian Performansi *Confussion Matrix*

Sebenarnya	Prediksi					
	Biasa	Bahagia	Marah	Muak	Sedih	Terkejut
Biasa	15	4	2	2	3	2
Bahagia	2	13	3	2	2	3
Marah	3	3	14	3	2	2
Muak	2	3	2	12	4	2
Sedih	3	2	3	3	13	2
Terkejut	2	3	4	3	2	11

Dari table tersebut, lalu dihitung akurasi totalnya dan precision untuk setiap kelas. Hasilnya adalah sebagai berikut.

Akurasi = $(15 + 12 + 11 + 9 + 11 + 9)/(15 + 4 + 2 + 2 + 3 + 2 + 2 + 13 + 3 + 2 + 2 + 3 + 3 + 3 + 14 + 3 + 2 + 2 + 2 + 3 + 2 + 12 + 4 + 2 + 3 + 2 + 3 + 3 + 13 + 2 + 2 + 3 + 4 + 3 + 2 + 11) = 0.5$

Wajah biasa = $\frac{15}{15+4+2+2+3+2}$ =0.5357

Bahagia = $\frac{13}{2+13+3+2+2+3}$ =0.52

Marah = $\frac{14}{3+3+14+3+2+2}$ =0.5185

Muak = $\frac{12}{2+3+2+12+4+2}$ =0.48

Sedih = $\frac{13}{3+2+3+3+13+2}$ =0.5

Terkejut = $\frac{11}{2+3+4+3+2+11}$ =0.44

Dari pengujian *conffusion matrix* menggunakan 166 kali pendeteksian, didapatkan akurasi rata-rata 50% benar dalam klasifikasi dan 50% salah. Faktor penyebab kesalahan dalam mengklasifikasikan menggunakan metode *Hidden markov Model* (HMM) karena sangat bergantung pada ciri citra dan banyaknya data latih yang digunakan.

3 KESIMPULAN DAN SARAN

3.1 Kesimpulan

Kesimpulan dari penelitian yang berjudul: "Implementasi Algoritma Hidden Markov Model untuk Pengenalan Isyarat Wajah" adalah sebagai berikut :

1. Penelitian pada pengolahan citra ekspresi wajah dapat digunakan untuk mengenali jenis ekspresi wajah yaitu ekspresi wajah "bahagia", "marah", "muak", "sedih", "terkejut" dan wajah normal.
2. Metode *Hidden Markov Model* dapat diterapkan dalam proses klasifikasi pengenalan jenis ekspresi wajah "Bahagia", "Marah", "Muak", "Sedih", "Terkejut" dan wajah normal yang dilakukan pada pengolahan citra ekspresi wajah dengan hasil akurasi rata-rata 50% benar dalam mengenali.

3.2 Saran

1. Penelitian menggunakan citra ekspresi wajah belum mampu mendeteksi ciri khusus dari ekspresi wajah karena dengan teknik "*preprocessing* deteksi tepi" wajah yang dideteksi merupakan bentuk dari tepian, sehingga dari warna, bentuk dan tekstur diabaikan untuk proses klasifikasi citra.
2. Diharapkan penelitian selanjutnya mampu mengenali ekspresi wajah berdasarkan bentuk, tekstur dan ciri fitur lainnya.

DAFTAR PUSTAKA

[1] S, P Marhutala dan Soelaiman, M.Kom, R. "Pengenalan Ekspresi Wajah Berdasarkan Bentuk dan Tekstur". Teknik Informatika, ITS.

[2] Feng-Sheng Chen, Chih-Ming Fu, dan Chung-Lin Huang. "*Hand Gesture recognition using a real-time tracking method and Hidden Markov Models*". Institute of Engineering, National Tsing Hua University.

[3] Zaenal Abidin, "Pengembangan Sistem Pengenalan Ekspresi Wajah Menggunakan Jaringan Syaraf Tiruan Backpropagation (Studi Kasus Pada Database MUG).", Matematika, Universitas Negeri Semarang, 2011.

[4] Shreyas Fadnavis "Image Interpolation Techniques in Digital Image Processing: An Overview", International Journal of Engineering Research and Applications, 2014.

[5] Abdul Kadir, dan Adhi Susanto, "Teori dan Aplikasi Pengolahan Citra", Andi, 2013.

[6] Mazid kamal, dan Ruki Suko Basuki., M. Kom, "Segmentasi Citra Daun Tembakau Berbasis Deteksi Tepi Menggunakan Algoritma Canny", Teknik Informatika, Universitas Dian Nuswantoro Semarang.

[7] Chrisopher Danil, "Edge Detection dengan Algoritma Canny", STMIK IBBI.

[8] SameerKais Jameel, "Face Recognition System Using PCA and DCT in HMM", University of Raparin, International Journal of Advanced Research in Computer and Communication Engineering, 2015.

[9] Syafruddin Syarif , Nadjamuddin Harun, Muh. Tola, M. Wihardi Tjaronge, Zulfajri B. Hasanuddin, Zulkarnain Jamid, Rifad Zulfikar Asgar., "Sistem Cerdas Deteksi Citra Dengan Metode Discrete Cosine Transform.", Universitas Hasanuddin, 2012.

[10] Ramen Atonov Purba, "Implementasi Algoritma Principal Components Analysis Dalam Sistem Pengenalan Citra Wajah.", LP3M Medan, 2013

[11] Han, J., Kamber, M. and Pei, J., "Data Mining: Concepts and techniques (3nd ed.), Waltham USA : Morgan Kaufmann Publisher, 2012.

[12] Sepritahara, "Sistem Pengenalan Wajah (*Face Recognition*) Menggunakan Metode Hidden Markov Model (HMM)" ,Teknik Elektro, UI.

[13] S.A. Santosh Kumar, D.R. Deepti, and B. Prabhakar, "*Face Recognition using Pseudo-2D Ergodic Hidden Markov Model*",Central Research Laboratory

[14] P Dymarski. 2011."*Hidden Markov Model, Theory and Aplications"*. India: InTech.

[15] Xiaolin Li, Marc Parizeau, and Rejean Plamondon., "Training Hidden Markov Models With Multiple Observations – A combinatorial Method.", IEEE Computer Society, 2000.

[16] Hadi Santoso, dan Agus Harjoko, "Haar Cascade Classifier dan Algoritma Adaboost Untuk Deteksi Banyak Wajah Dalam Ruang Kelas", Sistem Informasi. STIMIK Atma Luhur.

[17] Marini, Yennie., Emiyati., Hawariyah, Siti., dan Hartuti, Maryani., "Perbandingan Metode Klasifikasi Suvervised Maximum Likelihood Dengan Klasifikasi Berbasis Objek Untuk Inventarisasi Lahan Tambak Di Kabupaten Maros", Pusat Pemanfaatan Penginderaan Jauh, 2014.