BAB 2
DASAR TEORI

Transmisi dari citra adalah hal penting dalam komunikasi citra interaktif pada beberapa aplikasi seperti pengamatan jarak jauh (*remote surveillance*), pembelanjaan elektronik (*electronic shopping*), *telebrowsing* dan akses pada *database* yang besar. Untuk mengurangi ukuran data yang ditransmisikan digunakan teknik kompresi citra, salah satunya yaitu *Run Length Encoding* (*RLE)*. [6]

Identifikasi tanda tangan digunakan pada banyak aplikasi misalnya cek, validasi kartu kredit, sistem keamanan (*security system*), sertifikat, surat kontrak, dan lain-lain. Identifikasi tanda tangan tidak dapat dianggap permasalahan pengenalan bentuk (*pattern recogniton*) semata, hal ini karena contoh dari tanda tangan dari beberapa orang sama tetapi tidak identik. Variasi yang banyak dapat diamati dalam tanda tangan tergantung negara, umur, waktu, kebiasaan, keadaan mental dan psikologi, fisik dan kondisi praktis. [6]

Proses dari identifikasi tanda tangan terdiri dari proses pembelajaran dan pengujian (*learning stage dan testing stage*). Tujuannya adalah membuat *file* referensi, yang pada akhirnya untuk menghitung kesamaan diantara pengujian berdasarkan referensi tanda tangan untuk mengecek kecocokkan tanda tangan yang dimaksud.

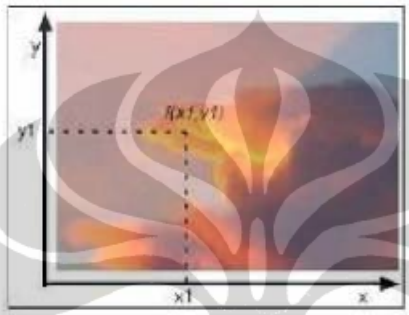
Saat ini, pentingnya identifikasi *biometric* mengalami peningkatan seiring dengan adanya perdagangan elektronik (*electronic commerce*). Identifikasi tanda tangan dikembangkan secara luas sebagai salah satu metoda identifikasi *biometric*. Salah satu metoda identifikasi untuk tanda tangan digunakan Hidden Markov Model (HMM) [7].

2.1 Pengolahan Citra

Citra digital adalah gambar dua dimensi yang dihasilkan dari perubahan gambar analog dua dimensi yang kontinu menjadi gambar diskrit. [3] Pengolahan citra dapat didefinisikan sebagai proses memperbaiki kualitas citra dengan menggunakan berbagai teknik pengolahan citra yang akan mentransformasikan citra menjadi citra lain agar mudah diinterpretasikan oleh manusia. Proses ini mempunyai ciri data masukan dan informasi keluaran yang

berbentuk citra. Pengolahan citra merupakan suatu proses awal atau *pre-processing*.

Nilai-nilai elemen penyusun matriks disebut dengan piksel sedangkan posisi elemen dalam baris dan kolom menyatakan koordinat titik-titik (x,y) dalam citra. Fungsi nilai f(x,y) adalah intensitas citra pada koordinat x dan y. Hal ini dapat dilihat pada Gambar 2.1. Tiap piksel mempunyai nilai digital yang dapat merepresentasikan citra sebenarnya.



Gambar 2.1 Citra digital [3]

2.1.1 *Leveling, Cropping dan Reshaping*

Leveling merupakan pemberian level pada nilai *graylevel* suatu citra dari suatu array tertentu ke suatu *range* nilai diskrit tertentu.

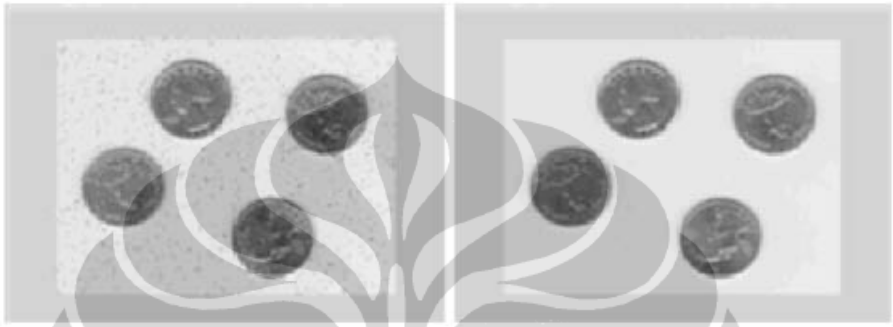
Cropping merupakan proses pemotongan citra pada elemen-elemen tertentu dari citra. Proses ini bertujuan untuk mengambil elemen-elemen yang diinginkan dari citra digital. Hasil pemotongan tersebut akan menjadi *sample-sample* citra yang selanjutnya akan dilatih dan diuji pada sistem pengenalan tanda tangan.

Reshapping merupakan proses perubahan ukuran matriks suatu citra menjadi ukuran matriks tertentu namun dengan jumlah *array* yang sama, agar ukuran matrik yang dimasukkan sama . Misalkan citra dengan matriks M x N diubah menjadi citra dengan ukuran N x M.

2.1.2 *Filter Median*

Filter median sangat bermanfaat untuk menghilangkan *outliers*, yaitu nilai-nilai yang ekstrim. *Filter median* menggunakan *sliding neighborhoods* untuk memproses suatu citra suatu operasi dimana filter ini akan menentukan nilai masing-

masing *pixel* keluaran yang bersebelahan dengan menentukan m x n disekitar *pixel* masukkan yang bersangkutan, *filter median* mengatur nilai-nilai *pixel* yang bersebelahan dan memilih nilai tengah atau median sebagai hasil. Pada Gambar 2.2 merupakan contoh menambahkan *salt and pepper* pada citra koin lalu menghilangkannya dengan menggunakan *filter median.*



Gambar 2.2 Penghilangan noise dengan *filter median* [3]

2.2 Kompresi Citra dengan Run Length Encoding (RLE) [4]

Komputer grafis digunakan dalam banyak bidang kehidupan sehari-hari untuk mengubah banyak tipe informasi komplek ke citra. Kita sering mendapati *file* citra yang sangat besar oleh karena itu pengkodean dan kompresi pada citra diperlukan.

Adapun tujuan dari kompresi yaitu :

- Mengurangi volume dari data yang akan ditransmisikan (*text, fax, image*).
- Mengurangi *bandwidth* yang diperlukan untuk transmisi dan untuk mengurangi penyimpanan yang diperlukan (*speech, audio, video*).

Kompresi *RLE* adalah jika data *d* terjadi sebanyak n kali pada aliran data input, maka kejadian n akan diganti dengan *nd*. Data yang terjadi berurutan *n* kali disebut *run length* dari *n*, dan pada kompresi disebut *run-length encoding* atau *RLE*. Kejadian yang berulang dari karakter yang sama disebut *'run'*. Jumlah dari pengulangan disebut panjang dari '*run*'.

RLE dapat juga digunakan untuk mengkompresi *grayscale image*[data compression complete reference]. Tiap *run* dari *pixel* dengan intensitas yang sama (*gray level*) dikodekan sebagai pasangan panjang *run* dan nilai *pixel* (*run length, pixel value*). *Run length* biasanya menempati satu byte, sampai *run* mencapai 255 *pixel*.

Nilai *pixel* menempati beberapa bit, tergantung pada jumlah *gray levels* (biasanya diantara 4 dan 8 bit).

Contoh kompresi *RLE* pada 8-bit *bitmap grayscale*

Data asli : 12 12 12 34 55 55 55 55 11 11 11 11 11 34 34 34

Data encoded : (3,12)(1,34)(4,55)(5,11)(3,34)

.

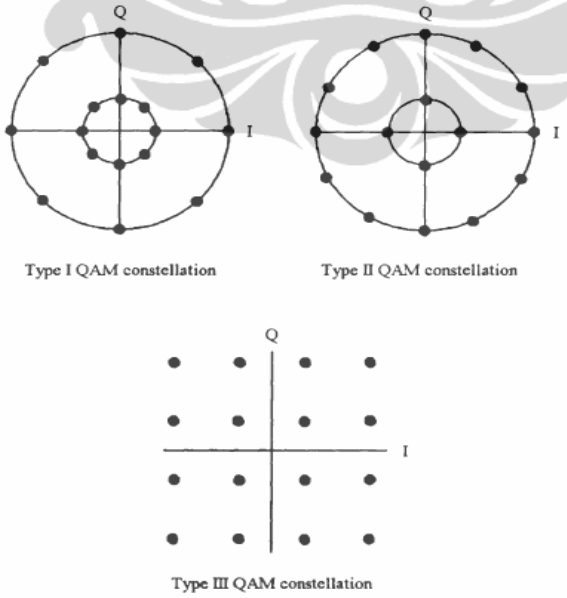
2.3 Modulasi *Quadrature Amplitude (QAM)* [5]

Quadrature amplitude modulation (*QAM*) adalah jenis modulasi yang memiliki selubung yang konstan (*constant envelope*) yang dapat mencapai efisiensi bandwidth yang lebih tinggi daripada *MPSK* dengan rata-rata daya sinyal yang sama. Sinyal baseband modulasi *QAM* adalah :

$$s_i(t) = A_i \cos(2\pi f_c t + \theta_i), \quad i = 1, 2, \ldots M \qquad \text{.....(2.1)}$$

dimana A_i adalah amplitudo dan θ_i adalah fase dari sinyal *baseband* yang ke i.

Representasi geometrik yang disebut konstelasi adalah cara yang jelas untuk menjelaskan sinyal QAM. Sinyal QAM ditunjukkan dengan titik (atau vektor, atau phasor) dengan koordinat (S_{i1} , S_{i2}). Kedua sumbunya biasanya dilabelkan dengan sumbu 'I' dan sumbu 'Q'. Gambar 2.3 menunjukkan contoh dari 3 tipe konstelasi *QAM*.



Gambar 2.3 Contoh tipe I, II dan III konstelasi QAM [5]

2.4 Kanal Fading Rayleigh

Model sistem sinyal yang diterima pada kanal *flat fading Rayleigh*, yaitu, [10]

$$y = hx + n \qquad \text{.....(2.2)}$$

dimana,

y adalah simbol yang diterima,

h dimodelkan sebagai variabel komplek Gaussian dengan *mean* '0' dan *variance* σ^2,

x adalah simbol yang ditransmisikan,

n adalah Additive White Gaussian Noise (AWGN)

Kanal AWGN adalah model kanal yang umum digunakan [Fuqin Xiong]. Pada model kanal ini ditambahkan derau putih *Gaussian* (*white Gaussian noise*) pada sinyal yang melewatinya. Hal ini menunjukkan bahwa respon frekuensi amplitudo kanal adalah *flat* dan respon frekuensi fasenya linier untuk semua frekuensi jadi sinyal yang termodulasi melewati kanal ini tanpa amplitudo yang hilang dan kerusakan fase dari komponen frekuensi. Tidak terdapat fading disini. Sinyal yang diterima adalah

$$r(t) = s(t) + n(t) \qquad \text{.....2.3}$$

dimana n(t) adalah *additive white Gaussian noise*.

2.5 *Fast Fourier Transform (FFT)*

DFT merupakan perluasan dari transformasi fourier yang berlaku untuk sinyal-sinyal diskrit dengan panjang yang berhingga. *Discrete Fourier Transform* (*DFT*) untuk *finite-length sequence* $x[n]$ yang terdefinisi untuk rentang $0 \le n \le N-1$ dinyatakan sebagai, [2]

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi kn/N} \qquad \dots \qquad (2.4)$$

Invers Discrete Fourier Transform (*IDFT*) diberikan oleh

$$x[n] = \frac{1}{N} \sum_{n=0}^{N-1} X[k] e^{j2\pi kn/N} \qquad \dots \qquad (2.5)$$

Penghitungan *DFT* dapat dinyatakan dalam bentuk matriks sebagai

$$\mathbf{X} = \mathbf{D}_N \mathbf{x} \qquad \ldots \qquad (2.6)$$

dengan **X** menyatakan vektor hasil *DFT* yaitu

$$\mathbf{X} = \begin{bmatrix} X[0] & X[1] & \ldots & X[N-1] \end{bmatrix}^T \qquad \ldots \qquad (2.7)$$

dan **x** adalah vektor input yaitu

$$\mathbf{x} = \begin{bmatrix} x[0] & x[1] & \ldots & x[N-1] \end{bmatrix}^T \qquad \ldots \qquad (2.8)$$

Jika didefinisikan

$$W_N = e^{-j2\pi/N} \qquad \ldots \qquad (2.9)$$

maka matriks DFT berukuran $N \times N$ maka

$$\mathbf{D}_N = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & W_N^1 & W_N^2 & \ldots & W_N^{N-1} \\ 1 & W_N^2 & W_N^4 & \ldots & W_N^{2(N-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_N^{N-1} & W_N^{2(N-1)} & \ldots & W_N^{(N-1)\times(N-1)} \end{bmatrix} \qquad \ldots \qquad (2.10)$$

Begitu juga IDFT dapat dinyatakan dengan bentuk matriks sebagai

$$\begin{bmatrix} x[0] \\ x[1] \\ \vdots \\ x[N-1] \end{bmatrix} = \mathbf{D}_N^{-1} \begin{bmatrix} X[0] \\ X[1] \\ \vdots \\ X[N-1] \end{bmatrix} \qquad \ldots \qquad (2.11)$$

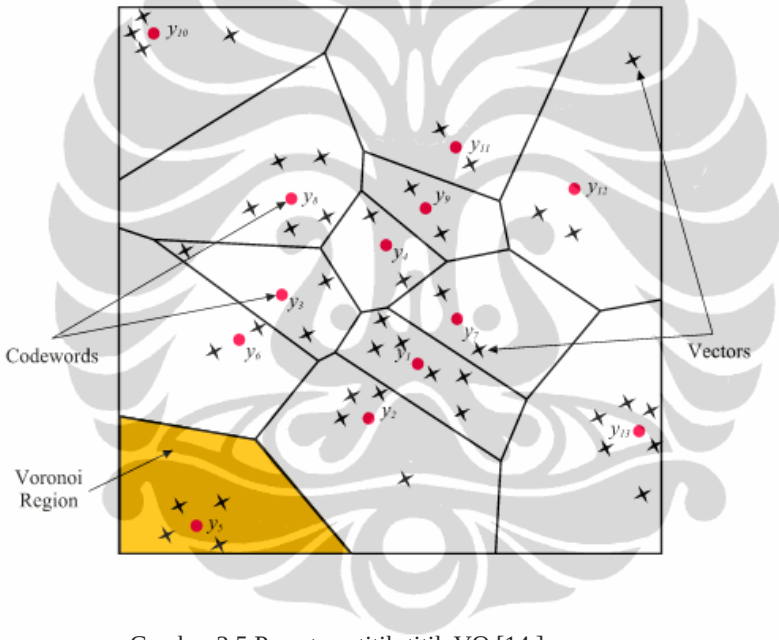
Berdasarkan persamaan (2.4) dan (2.5), perhitungan DFT dan IDFT memerlukan N^2 perkalian kompleks dan $N(N-1)$ penjumlahan kompleks. Namun, ada suatu metode yang dikembangkan untuk mengurangi kompleksitas algoritma menjadi hanya $N(\log_2 N)$. Teknik tersebut disebut sebagai algoritma *Fast Fourier Transform* (FFT) dan *invers Fast Fourier Transform* (IFFT).

Prinsip dasar FFT adalah menguraikan penghitungan *N*-titik DFT menjadi penghitungan DFT dengan ukuran yang lebih kecil dan memanfaatkan periodisitas dan simetri dari bilangan kompleks W_N^{kn}.

2.6 *Vector Quantization*

Vector quantization (VQ) adalah proses dari pemetaan vektor dari ruang vektor yang besar menjadi sebuah wilayah yang terbatas. Masing-masing wilayah ini disebut *cluster* dan dapat direpresentasikan dengan *centroid* yang disebut *codeword*. Koleksi dari semua *codeword* disebut *codebook* yang berhubungan untuk gelombang yang telah diketahui.

Pada (VQ) terjadi proses pemetaan vektor data yang merupakan titik-titik hasil dari proses FFT ke dalam sebuah wilayah yang terbatas dalam grafik dua dimensi (X-Y) dimana sumbu X merupakan komponen real dari masing-masing titik dan sumbu Y merupakan komponen imajiner dari masing-masing titik. Pemetaan titik-titik tersebut ditunjukkan oleh Gambar 2.5.



Gambar 2.5 Pemetaan titik-titik VQ [14]

Tujuan dari proses vektor kuantisasi adalah untuk menyederhanakan panjang data masukan agar proses selanjutnya menjadi lebih mudah. Tiap komponen dari spketrum frekuensi yang merupakan hasil FFT memiliki beberapa titik yang masing-masing memiliki komponen real dan imajiner. Kumpulan dari titik-titik yang memiliki jarak berdekatan membentuk suatu *cluster* dan setiap *cluster* yang terbentuk dapat direpresentasikan dengan *centroid* yang disebut *codeword*. Koleksi dari semua *codeword* disebut *codebook* . VQ *codebook* untuk masing-masing citra yang telah diketahui dibuat dengan mengumpulkan vektor *training*-nya menjadi sebuah cluster.

Jarak antara satu vektor ke codeword terdekat disebut *VQ Distortion*. Semakin kecil *VQ Distortion*nya, maka *cluster* yang terbentuk menjadi lebih akurat.

Pada tahap pengenalan, sebuah input dari citra lainnya yang tidak dikenal akan dilakukan proses *vector-quantized* dengan menggunakan semua trained codebook dan selanjutnya dihitung total VQ *distortion*-nya. Total distortion yang paling kecil antara codeword dari salah satu citra dalam database dan VQ *codebook* dari input citra diambil sebagai hasil identifikasi.

Untuk memperbaiki VQ dalam proses pembuatan *codebook*, digunakan *General Lylod Algorithm* (GLA) atau biasa disebut *LBG Algorithm*. Prosedur implementasinya adalah sebagai berikut[16]:

(1) Mendesain suatu vektor *codebook* yang merupakan *centroid* dari keseluruhan vektor *training*.

(2) Mengubah ukuran *codebook* menjadi dua kalinya dengan membagi masing- masing *current codebook*, C_n, menurut aturan:

$$C_n^+ = C_n(1+\varepsilon) \quad \text{.....(2.12)}$$

$$C_n^- = C_n(1-\varepsilon) \quad \text{.....(2.13)}$$

di mana *n* bervariasi dari 1 sampai sebesar ukuran *codebook* dan ε adalah parameter *splitting* (ε = 0,01).

(3) Melakukan *Nearest Neighbour Search* dengan mengelompokkan vektor- vektor *training* yang berkumpul pada blok-blok tertentu. Selanjutnya menentukan *codeword* dalam *current codebook* terdekat dan memberikan tanda vektor berupa sel yang diasosiasikan dengan *codeword* terdekat.

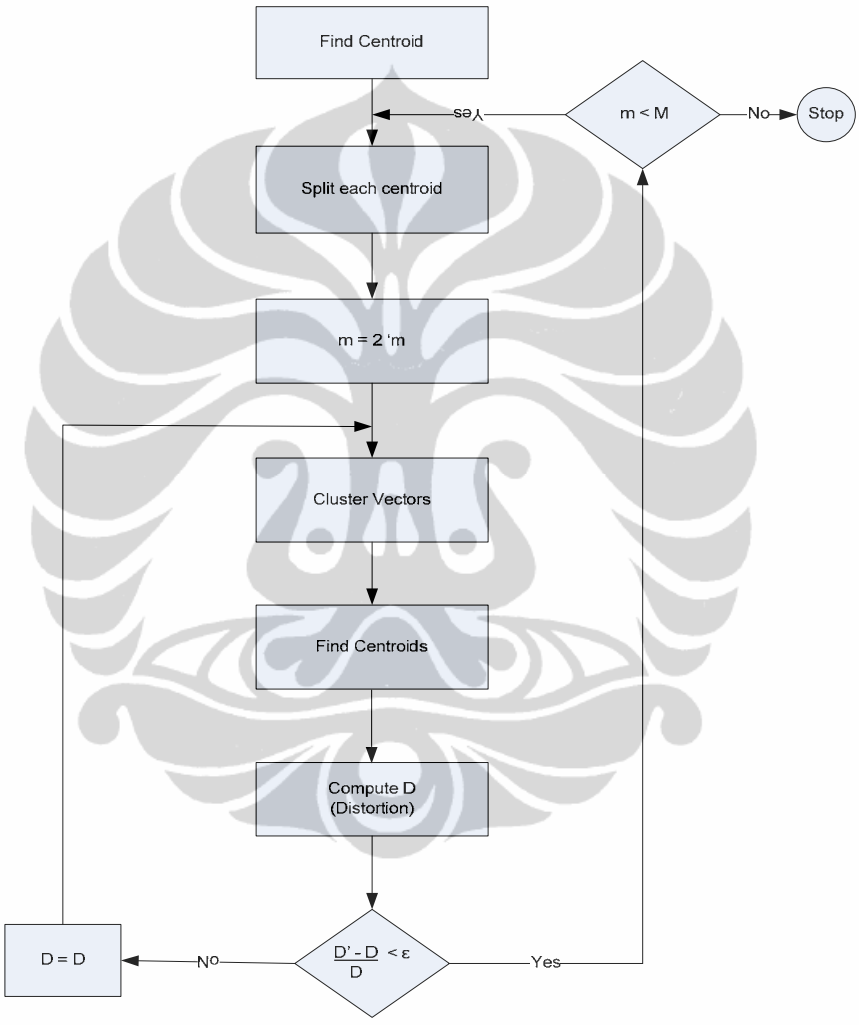
(4) Melakukan *centroid update* dengan menentukan *centroid* baru pada masing- masing sel dengan menggunakan vektor *training* pada sel tersebut.

(5) Melakukan iterasi pertama dengan mengulang langkah 3 dan 4 sampai jarak rata-rata di bawah *present treshold*.

(6) Melakukan iterasi kedua dengan mengulang langkah 2, 3, dan 4 sampai *codebook* berukuran *M*.

Gambar 2.5 menunjukkan diagram alir dari algoritma LBG. *Cluster vector* menerapkan prosedur *nearest neighbour search* yang menandai masing-masing

training vector ke sebuah cluster yang diasosiasikan dengan *codeword* terdekat. “find centroid” merupakan prosedur meng-update centroid untuk menentukan codeword yang baru. “Compute D (distortion)” berarti menjumlah jarak semua training vector dalam *nearest neighbour search* terhadap *centroid* untuk menentukan besarnya *distortion*.



Gambar 2.6 Diagram alir algoritma LBG [16]

2.7 *Hidden Markov Model (HMM)*

Hidden Markov Model (HMM) adalah sebuah model statistik dari sebuah sistem yang diasumsikan sebuah *Markov Process* dengan parameter yang tak diketahui, dan tantangannya adalah menentukan parameter-parameter tersembunyi

(*hidden*) dari parameter-parameter yang dapat diamati. Parameter-parameter yang ditentukan kemudian dapat digunakan untuk analisis yang lebih jauh, misalnya untuk aplikasi *pattern recognition*. Sebuah HMM dapat dianggap sebagai sebuah *Bayesian Network* dinamis yang paling sederhana. Adapun perhitungan untuk menghitung probabilitas kondisional (*conditional probabilities*) dirumuskan dengan aturan *Bayes*.

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)} \qquad \ldots\ldots(2.14)$$

Pada model *Markov* umum, state-nya langsung dapat diamati, oleh karena itu probabilitas transisi *state* menjadi satu-satunya parameter. Di dalam *Model Markov* yang tersembunyi (*hidden*), *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-statenya.

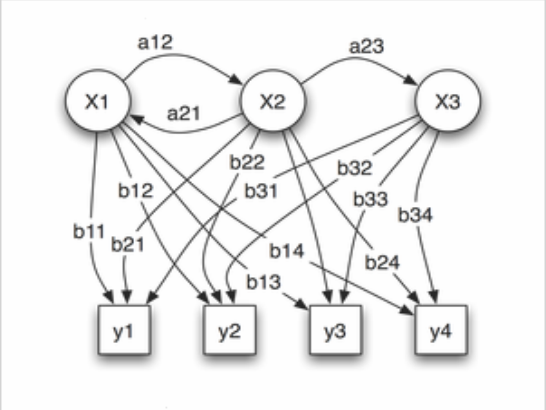
Dari Gambar 2.7 dapat dilihat parameter probabilitas dari *Hidden Markov Model* yaitu :

x = state

y = observasi yang mungkin

a = probabilitas transisi state

b = probabilitas keluaran

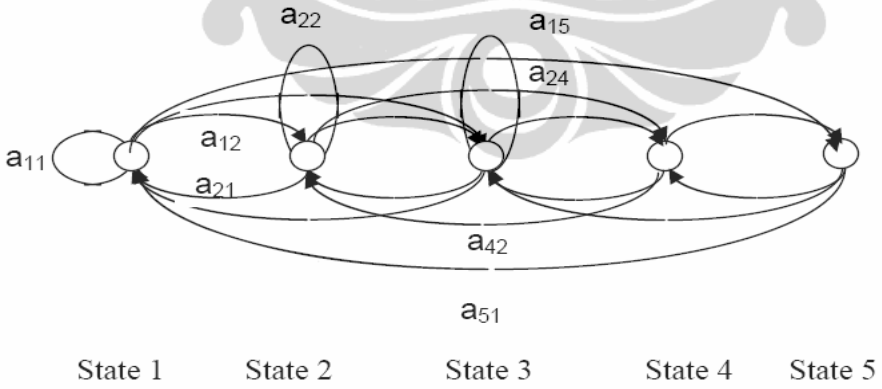


Gambar 2.7 Contoh parameter probabilitas dari Hidden Markov Model [15]

Pemilihan model topologi HMM adalah hal yang prinsip untuk mendapatkan hasil yang memuaskan pada fase pembelajaran dan verifikasi (learning and verification phase). Untuk model diskrit terdapat dua faktor yang dominan [12]. Pertama jumlah dari state yang digunakan dan kedua adalah jumlah transisi diantara state. Pada karakter tanda tangan digunakan model diskrit *left-to-right*, karena hal ini menggunakan karakteristik dinamik dari tulisan tangan Latin (*Latin handwriting*), dimana gerakan tangan selalu dari kiri ke kanan.

Gambar 2.8 Model *left–to-right* pada proses pembelajaran HMM[12]

Salah satu cara untuk mengklasifikasikan HMM adalah dengan melihat bentuk matriks transisinya (A) dari rantai Markov. Bentuk yang umum adalah bentuk *ergodic* atau bentuk yang setiap *state* saling terhubung (*fully connected HMM*). Seperti terlihat pada Gambar 2.9 untuk N = 5 *state*, model ini mempunyai nilai a_{ij} antara 0 dan 1.



Gambar 2.9 State diagram dari HMM atau HMM *chain* dengan 5 state [17]

2.8 Parameter-parameter *Hidden Markov Model* [11]

Hidden Markov Model dapat dituliskan sebagai λ = (A, B, π). Dengan diketahuinya N, M, A, B, dan π, Hidden Markov Model dapat menghasilkan urutan observasi $O = O_1O_2...O_T$ dimana masing-masing observasi O_t adalah simbol dari V, dan T adalah jumlah urutan observasi.

Perhitungan yang efisien dari P(O|λ), yaitu probabilitas urutan observasi apabila diberikan urutan observasi $O = O_1O_2O_3...O_T$ dan sebuah model λ = (A, B, π). Misalkan diberikan urutan state

$$Q = q_1q_2...q_T \qquad .....(2.15)$$

Dimana q_1 adalah inisial state. Dengan demikian probabilitas urutan observasi O untuk urutan state pada persamaan 2.15 adalah

$$P(O|Q,\lambda) = \prod_{t=1}^{T} P(O_t|q_t,\lambda) \qquad .....(2.16)$$

Sehingga didapatkan

$$P(O|Q,\lambda) = b_{q1}(O_1)b_{q2}(O_2)...b_{qT}(O_T) \qquad .....(2.17)$$

Probabilitas dari urutan state Q dapat dituliskan

$$P(Q|\lambda) = \pi_{q1}a_{q1q2}a_{q2q3}...a_{qT-1qT} \qquad .....(2.18)$$

Probabilitas gabungan dari O dan Q yaitu probabilitas dari O dan Q yang terjadi secara bersamaan. Probabilitas gabungan ini dapat dituliskan

$$P(O,Q|\lambda) = P(O|Q,\lambda)P(Q|\lambda) \qquad .....(2.19)$$

Probabilitas observasi O yang diberikan, diperoleh dengan menjumlahkan seluruh probabilitas gabungan terhadap semua kemungkinan urutan state q, yaitu

$$P(O|\lambda) = \sum_{allQ} P(O|Q,\lambda)P(Q|\lambda) \qquad .....(2.20)$$

Atau dapat juga ditulis

$$P(O|\lambda) = \sum_{q1q2...qT} \pi_{q1} b_{q1}(O_1) a_{q1q2} b_{q2}(O_2) ... a_{qT-1qT} b_{qT}(O_T) \quad .....(2.21)$$

Untuk menghitung 2.24 dengan menggunakan prosedur *forward*. Variabel *forward* $\alpha_1(i)$ didefinisikan sebagai probabilitas sebagian urutan observasi $O_1O_2...O_t$ (hingga waktu t) dan state S_i pada waktu t, dari model λ yang diberikan.

$$\alpha_t(i) = P(O_1 O_2 ... O_t, q_t = S_i | \lambda) \quad .....(2.22)$$

Untuk menyelesaikan $\alpha_1(i)$ adalah sebagai berikut :

1. Inisialisasi

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq N \quad .....(2.23)$$

2. Induksi

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right] b_j(O_{t+1}), \quad 1 \leq j \leq N \quad .....(2.24)$$

3. Terminasi

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i) \quad .....(2.25)$$

Adapun parameter-parameter Hidden Markov Model dapat dituliskan sebagai $\lambda = (A, B, \pi)$ adalah sebagai berikut :

1. Parameter A pada HMM dinyatakan dalam sebuah matriks dengan ukuran M x M dengan M adalah jumlah state yang ada. Matriks transisi pada Gambar 2.8 terdiri dari 5 state sehingga setiap state memiliki 5 hubungan transisi, maka parameter A dapat dituliskan dalam bentuk matriks seperti pada persamaan 2.26.

$$A = aij = \begin{pmatrix} a11 & a12 & a13 & a14 & a15 \\ a21 & a22 & a23 & a24 & a25 \\ a31 & a32 & a33 & a34 & a35 \\ a41 & a42 & a43 & a44 & a45 \\ a51 & a52 & a53 & a54 & a55 \end{pmatrix} \quad \ldots\ldots (2.26)$$

2. Parameter B : disebut sebagai probabilitas *state*, merupakan probabilitas kemunculan suatu state dalam deretan seluruh *state* yang ada.

   Parameter B dalam HMM dituliskan dalam bentuk matriks kolom dengan ukuran M x 1 dimana M merupakan jumlah seluruh state yang ada. Sebagai contoh, jika terdapat 5 buah *state* dalam suatu kondisi, maka matriks B yang terbentuk ditunjukkan oleh persamaan 2.27.

$$B = \begin{pmatrix} b1 \\ b2 \\ b3 \\ b4 \\ b5 \end{pmatrix} \quad \ldots\ldots (2.27)$$

3. Parameter π : disebut sebagai probabilitas awal, merupakan probabilitas kemunculan suatu *state* di awal. Sama halnya dengan parameter B, parameter π juga dituliskan dalam bentuk matriks kolom dengan ukuran M x 1 dimana M adalah jumlah *state*nya. Jadi jika terdapat 5 *state*, maka parameter π yang dihasilkan akan ditunjukkan seperti pada persamaan 2.28.

$$\Pi = \begin{pmatrix} c1 \\ c2 \\ c3 \\ c4 \\ c5 \end{pmatrix} \quad \ldots\ldots (2.28)$$

Dari ketiga parameter utama maka HMM dapat dituliskan dalam bentuk $\lambda = (A, B, \pi)$. Dari kesemua parameter yang ada maka bisa diperoleh suatu

probabilitas observasi(O). Fungsi untuk probabilitas O ditunjukkan oleh persamaan 2.29

$$P(O) = \sum_{i=1}^{N} P(A_{ij}) * P(B_i) \qquad \text{. . . . .(2.29)}$$

Berikut adalah contoh perhitungan untuk mencari probabilitas observasi:

Citra 1 → (w1, w1, w2, w1, w2) → P(O) citra1 = $c_1 * a_{11} * b_1 * a_{12} * b_1 * a_{21} * b_2 * a_{12} * b_1$

Citra 2 → (w2, w1, w1, w3, w2) → P(O) citra2 = $c_2 * a_{21} * b_2 * a_{11} * b_1 * a_{13} * b_1 * a_{32} * b_3$