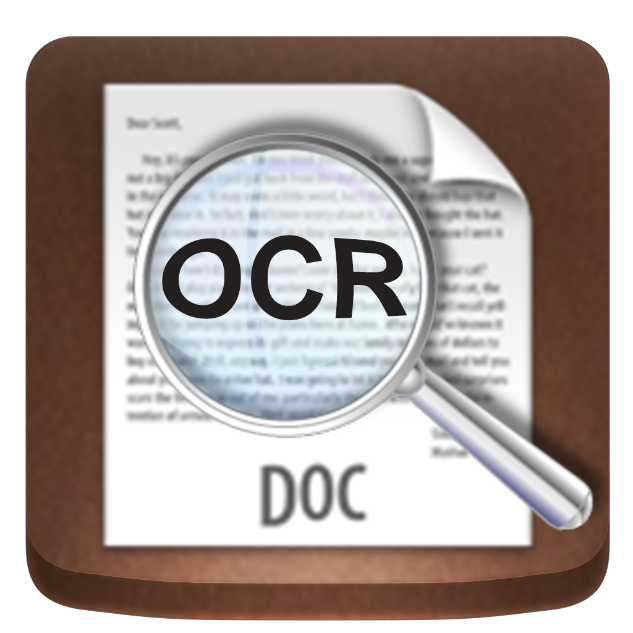
**Bab II**

**Tinjauan Pustaka**

## II.1 Optical Character Recognition (OCR)

OCR (Optical Character Recognition) berfungsi untuk memindai citra digital menjadi teks yang ada didalam citra tersebut. OCR juga bisa menjadi pendukung atau aplikasi tambahan untuk scanner. Dengan adanya OCR, citra yang bertulisan tangan, tulisan mesin ketik atau teks komputer, dapat dimanipulasi. Teks yang discan dengan OCR dapat dicari kata per kata atau per kalimat, dan setiap teks dapat dimanipulasi, diganti, atau diberikan barcode.



Gambar II.1 *Optical Character Recognition* (OCR)

(<https://learn.eprolabs.com/wp-content/uploads/2017/01/OCR-Feature.png>)

Pada Arabic Optical Character Recognition dilakukan dengan 5 tahap yakni sebagai berikut.

1. Tahap *pre-processing* untuk meningkatkan kualitas citra masukan dan karakter yang akan dikenali.
2. Tahap Segmentasi untuk memotong huruf yang terdapat pada rangkaian kata atau kalimat menjadi huruf – huruf yang berdiri sendiri.
3. Tahap *Thinning* berguna untuk mengambil tulang dari huruf yang telah disegmentasi, penulangan beguna untuk mengambil jalur dari gambar.

4. Tahap Ekstraksi Ciri untuk mengambil ciri pembeda dari citra untuk dimasukkan ke proses klafisikasi.

5. Tahap Klasifikasi untuk memproses ciri yang mengidentifikasikan citra karakter dan mengklasifikasikannya kedalam masing – masing kelas.

**2.2 Bahasa Arab**

Bahasa Arab (**اللغة العربية**  *al-lughah al-‘Arabīyyah*, atau secara ringkas **عربي** *‘Arabī*) adalah salah satu bahasa [Semit Tengah](https://id.wikipedia.org/w/index.php?title=Semit_Tengah&action=edit&redlink=1), yang termasuk dalam rumpun [bahasa Semit](https://id.wikipedia.org/wiki/Bahasa_Semit) dan berkerabat dengan [bahasa Ibrani](https://id.wikipedia.org/wiki/Bahasa_Ibrani) dan [bahasa-bahasa Neo Arami](https://id.wikipedia.org/wiki/Bahasa_Aram#Bahasa_dan_dialek_Aram). Bahasa Arab memiliki lebih banyak penutur daripada bahasa-bahasa lainnya dalam rumpun bahasa Semit. Ia dituturkan oleh lebih dari 280 juta orang. sebagai [bahasa pertama](https://id.wikipedia.org/wiki/Bahasa_ibu), yang mana sebagian besar tinggal di [Timur Tengah](https://id.wikipedia.org/wiki/Timur_Tengah) dan [Afrika Utara](https://id.wikipedia.org/wiki/Afrika_Utara). Bahasa ini adalah bahasa resmi dari 25 negara, dan merupakan bahasa peribadatan dalam agama [Islam](https://id.wikipedia.org/wiki/Islam) karena merupakan bahasa yang dipakai oleh [Al-Qur'an](https://id.wikipedia.org/wiki/Al-Qur%27an). Berdasarkan penyebaran geografisnya, bahasa Arab percakapan memiliki banyak variasi (dialek), beberapa dialeknya bahkan tidak dapat saling mengerti satu sama lain. Bahasa Arab modern telah diklasifikasikan sebagai satu makrobahasa dengan 27 sub-bahasa dalam [ISO 639-3](https://id.wikipedia.org/wiki/ISO_639-3). [Bahasa Arab Baku](https://id.wikipedia.org/w/index.php?title=Bahasa_Arab_Baku&action=edit&redlink=1) (kadang-kadang disebut *Bahasa Arab Sastra*) diajarkan secara luas di sekolah dan universitas, serta digunakan di tempat kerja, pemerintahan, dan media massa.

Bahasa Arab Baku berasal dari [Bahasa Arab Klasik](https://id.wikipedia.org/wiki/Bahasa_Arab_Klasik), satu-satunya anggota rumpun [bahasa Arab Utara Kuno](https://id.wikipedia.org/w/index.php?title=Bahasa_Arab_Utara_Kuno&action=edit&redlink=1) yang saat ini masih digunakan, sebagaimana terlihat dalam [inskripsi](https://id.wikipedia.org/wiki/Prasasti) peninggalan Arab pra-Islam yang berasal dari abad ke-4. Bahasa Arab Klasik juga telah menjadi bahasa kesusasteraan dan bahasa peribadatan [Islam](https://id.wikipedia.org/wiki/Islam) sejak lebih kurang [abad ke-6](https://id.wikipedia.org/wiki/Abad_ke-6). [Abjad Arab](https://id.wikipedia.org/wiki/Abjad_Arab) ditulis dari kanan ke kiri.

Bahasa Arab telah memberi banyak kosakata kepada bahasa lain dari dunia Islam, sama seperti peranan [Latin](https://id.wikipedia.org/wiki/Bahasa_Latin) kepada kebanyakan bahasa Eropa. Semasa [Abad Pertengahan](https://id.wikipedia.org/wiki/Abad_Pertengahan) bahasa Arab juga merupakan alat utama budaya, terutamanya dalam sains, matematik adan filsafah, yang menyebabkan banyak bahasa Eropa turut meminjam banyak [kosakata](https://id.wikipedia.org/wiki/Kosakata) dari bahasa Arab. [15]

Karakteristik Bahasa Arab :

1. Huruf arab ditulis dari kanan ke kiri
2. Tidak memiliki huruf kapital
3. Terdiri dari 28 Huruf dasar.
4. Huruf berubah bentuk tergantung pada posisi mereka dalam kata, sehingga memiliki 4 bentuk yakni terisolasi, posisi di awal katam di tengah kata dan di akhir kata.
5. Karakter terhubung ketika diketik atau dicetak
6. Huruf arab banyak kemiripan, dan pada umumnya terdiri dari lekukan dan putaran

**Tabel II.1. Karakter Huruf Hijaiyah (Handayani, Iskandar, & Andrian, 2014)**

| **Isolated** | **Begin** | **Middle** | **End** |  | **Isolated** | **Begin** | **Middle** | **End** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |

**II.3 Praolah Citra**

Pengolahan awal atau praolah (*preprocessing*) citra diperlukan sebelum  
beberapa proses. Tujuan dari pengolahan awal ialah agar pengolahan lebih mudah dilakukan. Pengolahan awal umumnya terdiri dari penghilangan derau (*noise*), penghalusan atau penajaman gambar, dan pengubahan gambar menjadi citra abu-abu atau citra biner.

**II.3.1 Deteksi dan Koreksi Kemiringan**

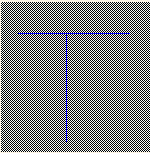
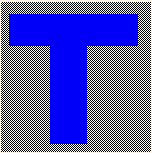
Citra dokumen masukan terkadang dalam posisi miring. Kemiringan gambar dapat menyulitkan proses selanjutnya, terutama pada segmentasi baris.

## II.3.2 Penipisan

Penipisan adalah proses pengambilan tulang dari suatu pola, proses ini mengikis piksel sebanyak mungkin tanpa mempengaruhi bentuk umum. Setelah piksel di kikis, pola tersebut tetap harus dikenali. Kerangka yang diperoleh harus memiliki sifat sebagai berikut :

1. Setipis mungkin
2. Terhubung
3. Berpusat ditengah

Bila sifat ini terpenuhi, algoritma harus berhenti, berikut adalah pola dan hasil penipisannya.



Gambar II.2 Pola huruf dan hasil penipisan

Penipisan berguna apabila kita tidak tertarik pada ukuran dari pola melainkan pada posisi relatif goresan dari pola. Ada beberapa algoritma yang dirancang untuk tujuan ini. Dalam penelitian ini penipisan dilakukan dengan algorima Hilditch.

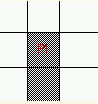
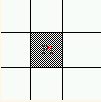
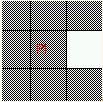
Algoritma Hilditch terdiri dari melakukan beberapa lintasan pada pola dan pada setiap celah, algoritma memeriksa semua piksel dan memutuskan untuk mengubah piksel dari hitam ke putih jika memenuhi empat kondisi berikut:

1. 2 < = B(p1) < = 6
2. A(p1)=1
3. p2.p4.p8=0 or A(p2)!= 1
4. p2.p4.p6=0 or A(p4)!= 1   
   Stop when nothing changes (no more pixels can be removed)

Berikut adalah penjelasan beberapa kondisi yang haru sdipenuhi dalam algoritma Hilditch.

1. Kondisi 1 : *2 < = B(p1) < = 6*

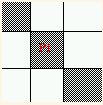
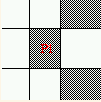
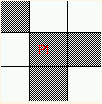
Kondisi ini menggabungkan dua sub kondisi, pertama bahwa jumlah tetangga tidak nol dari p1 lebih besar dari atau sama dengan 2 dan yang kedua yang kurang dari atau sama dengan 6.

Gambar II.3 B*(p1)=1,* B*(p1)=0 dan* B*(p1)=7*

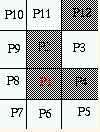
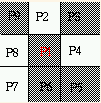
Seperti gambar membuat jelas, jika B (p1) = 1, maka p1 adalah kerangka tip-point dan tidak boleh dihapus. Jika B (p1) = 0, maka p1 adalah titik terisolasi dan juga harus disimpan (jika ini adalah suara, bukan pekerjaan dari proses skeletonisasi untuk menghapus piksel (ini adalah tugas dari algoritma penghilangan derau) Jika B (p1) = 7, p1 tidak ada lagi pada batas pola dan karenanya tidak boleh menjadi kandidat untuk dihapus.

1. Kondisi 2 : *A(p1)=1*

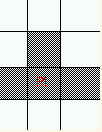
  

Gambar II.4 A(p1)=2, A(p1)=2, dan A(p1)=3

1. Kondisi 3 : p2.p4.p8=0 or A(p2)!= 1

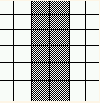
 

Gambar II.5 A(p2) tidak 1 dan p2.p4.p8=0



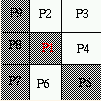
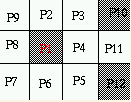
Gambar II.6 p2.p4.p8 tidak 0 dan A(p2)=1

Kondisi ini memastikan bahwa garis vertikal dengan lebar 2 piksel tidak terkikis sepenuhnya oleh algoritma.

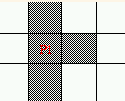


Gambar II.7 Garis vertikal lebar 2 piksel tidak terkikis sepenuhnya oleh algoritma.

1. Kondisi 4 : p2.p4.p6=0 or A(p4)!= 1

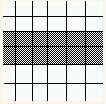


Gambar II.8 A(p4)!= 1 dan p2.p4.p6=0



Gambar II.9 p2.p4.p6!=0 and A(p4)=1

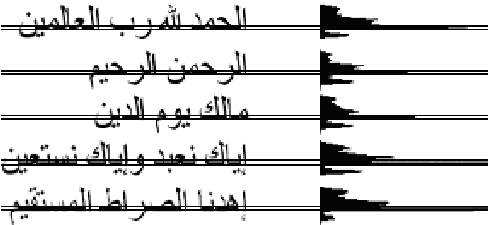
Kondisi ini memastikan bahwa garis horizontal 2 piksel lebar tidak terkikis sepenuhnya oleh algoritma.



Gambar II.10 Garis horizontal lebar 2 piksel tidak terkikis sepenuhnya oleh algoritma.

**II.3.3 Penghitungan Posisi Lini Basis Baris**

Lini basis (*baseline*) adalah garis khayal yang menghubungkan setiap huruf pada  
sebuah kata atau kalimat. Karena sifat tulisan Arab yang bersambung, lini basis berfungsi sebagai garis tempat huruf saling menempel. Posisi lini basis juga berfungsi untuk membantu tahap proses pengenalan huruf selanjutnya. Lini basis dapat dimanfaatkan dalam segmentasi atau ekstraksi fitur. Posisi lini basis dapat dihitung dengan proyeksi horizontal, yakni dengan menghitung jumlah piksel pada citra secara mendatar. Jumlah ini membentuk grafik kepadatan piksel setiap baris, seperti tampak pada Gambar II.11. Puncak dari grafik ini adalah lini basis atau *baseline* dari kata yang bersangkutan



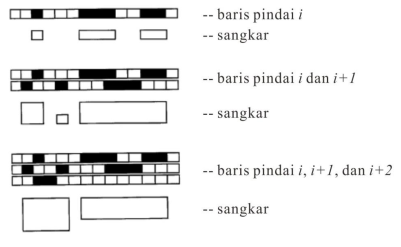
Gambar II.11 Menghitung lini basis dengan proyeksi horizontal

**II.4 Segmentasi**

Segmentasi atau proses pemotongan adalah proses yang penting dalam sistem pengenalan huruf khususnya huruf Arab, dikarenakan kalimat bahasa Arab yang ditulis secara bersambung. Segmentasi memiliki tujuan untuk memisahkan baris, kata dan huruf dalam citra. Setelah terpisah, masing-masing huruf dapat diidentifikasi lebih lanjut.

Jenis segmentasi tergolong dalam dua tipe, yaknik sebagai berikut.

1. Segmentasi eksternal



II.12 Proses membangun pixel terhubung (Amin, 2000)

Segmentasi eksternal adalah memotong dokumen menjadi bagian utama seperti paragraf, baris, kalimat, atau kata. Segmentasi eksternal juga berhubungan dengan analisis dokumen, misalnya untuk memisahkan citra teks dan citra bukan teks, memisahkan upakata dengan proyeksi vertikal akan dilakukan penelusuran dari kiri ke kanan apakah terdapat spasi atau tidak dikalimat tersebut, jika terdapat spasi maka didapatkan upakata. Adapun metode lainnya untuk memisahkan upakata adalah metode yang dikembangkan Amin (2000), yakni dengan cara memisahkan upakata dengan mencari komponen piksel terhubung. Hal ini dilakukan dengan iterasi mencari piksel bersambung mendatar pada setiap baris, seperti tampak pada Gambar II.12.

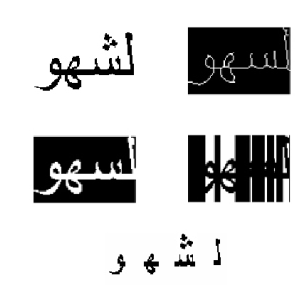
Dilakukan iterasi pada setiap baris. Jika menempel dengan piksel bersambung di  
baris sebelumnya, piksel di masukkan dalam suatu sangkar. Dengan demikian,  
pada akhirnya satu upakata akan terpisahkan selama piksel hitamnya tidak saling  
menempel dengan upakata lain (Amin, 2000)

1. Segmentasi internal

Segmentasi internal adalah berusaha memotong kata menjadi huruf per huruf khususnya pada tulisan bersambung seperti tulisan Arab. Dalam pengenalan huruf latin proses segmentasi internal dapat dilakukan dengan mudah karena huruf – huruf terpisah satu sama lain (kecuali tulisan tangan). Berbeda dengan tulisan Arab yang memiliki sifat bersambung, baik ditulis dengan tangan maupun tulisan cetak. Oleh karena itu masalah segmentasi dalam tulisah arab perlu diperhatikan.

Zidouri (2010) telah memberikan metode baru untuk segmentasi huruf tulisan Arab cetak. Langkah awal dari metode Zidouri ialah melakukan penipisan terhadap citra upakata. Algoritma Zidouri memperkirakan kemungkinan titik potong antar huruf dengan mencari garis horizontal pada citra tulang. Setiap baris pada citra dipindai dari kiri ke kanan untuk mencari deretan piksel horizontal atau pita horizontal dengan lebar lebih dari *Ls*. Nilai *Ls* adalah lebar terpendek dari huruf terkecil.

Setelah deretan piksel horizontal ditemukan, dilakukan pengecekan piksel pada  
bagian atas dan bawah pita piksel tersebut. Jika tidak ditemukan piksel, batas  
deretan piksel tersebut dianggap sebagai kandidat pita pandu segmentasi.  
Kandidat-kandidat pita pandu ini digambar pada citra sesuai lebar dan lokasi  
ditemukannya.



II.13 Contoh segmentasi upakata menjadi huruf (Zidouri, 2010)

Dari kandidat pita pandu vertikal, dilakukan ekstraksi beberapa fitur untuk  
memilih pita yang akan digunakan untuk memotong upakata menjadi huruf. Fitur  
yang diekstrak adalah lebar pita, jarak antar pita, urutan pita, posisi terhadap lini  
basis, dan titik tengah pita. Terdapat beberapa aturan untuk memilih pita  
pemotongan yang akan digunakan dari pita-pita yang ada. Misalnya, apakah  
rentetan piksel yang menyebabkan pita ditemukan berada di atas atau di bawah  
lini basis, apakah jarak pita dengan pita di kanannya kurang dari lebar maksimum  
huruf, dll. Tahapan pemotongan dapat diamati pada Gambar II.13 dan metode  
lebih rinci dapat ditelusuri pada (Zidouri, 2010)

**II.5 Klasifikasi dan Penelitian Terkait**

Penelitian tentang pengenalan huruf Arab lebih sedikit dibandingkan penelitian tentang huruf latin. Beberapa penelitian dalam bidang ini di antaranya adalah oleh Rana, dkk. (2015), Mehmood (2007), dan Albadr (2013).

Rana, dkk. (2015) mengembangkan sistem pengenalan karakter huruf Arab dari tulisan tangan. Ekstraksi fitur yang digunakan adalah jumlah komponen yang terhubung, jumlah lubang, jumlah titik, posisi dari titik dan kerapatan atas dan bawah huruf. Metode klasifikasi yang digunakan adalah Artificial Neural Network (ANN). ANN dilatih menggunakan algoritma Hopfield. Hasil penelitian menunjukkan sistem dapat mengenali 8 tulisan tangan huruf Arab **(ا ب ش س خ ض ع و)** dengan rata – rata pengenalan 77.25% Hal ini dapat dicapai karena sistem hanya mengenali karakter huruf Arab tunggal tanpa melalui proses segmentasi huruf.

Mehmood (2007) mengembangkan sistem pengenalan huruf Arab. Metode klasifikasi yang digunakan adalah Support Vector Machine (SVM), masukan dari SVM adalah ekstraksi fitur *Momment invariant*, jumlah dan posisi titik, dan jumlah lubang dari setiap huruf. Pada klasifikasi digunakan *Multiclass* SVM dengan 58 kelas (bentuk dari karakter). Metode ini menghasilkan tingkat mengenalan 98,34%. Sama dengan penelitian Rana, dkk(2015) penelitian ini dapat mencapai tingkat akurasi yang tinggi karena tidak melalui tahap segmentasi huruf.

Albadr (2013) mengembangkan sistem pengenalan huruf arab dalam rangkaian kata. Adapun tahap yang digunakan adalah pemrosesan awal, segmentasi, penulangan, ekstraksi fitur dan klasifikasi. Pada tahap segmentasi huruf dilakukan dengan algoritma Zidouri. Pada penelitian ini di ekstrak 12 fitur. Tiap fitur dibedakan menjadi 3 bagian, yakni bagian utama, garis keliling tulang dan objek sekunder (titik, hamzah dan zig – zag) pada setiap huruf. Penelitian ini mencapai tingkat akurasi pengenalan 48,3%.

**II.6 Machine Learning**

Menurut Tom M. Mitchel (1997), defenisi fomal tentang machine learning sebagai berikut :

“Sebuah program komputer dikatakan belajar dari pengalaman E yang bergabung pada target T dan ukuran kinerja program P jika kinerja pada target T, menggunakan ukuran P, ditingkatkan oleh pengalaman E.”

Dengan kata lain, *machine learning* meningkatkan mesin untuk memberikan hasil prediksi pada data baru setelah mendapat pengalaman dari sekumpulan data (data training). Berdasarkan input yang diberikan pada data training, *machine learning* dikelompokkan menjadi dua jenis, yaitu :

**II.6.1 *Supervised Learning***

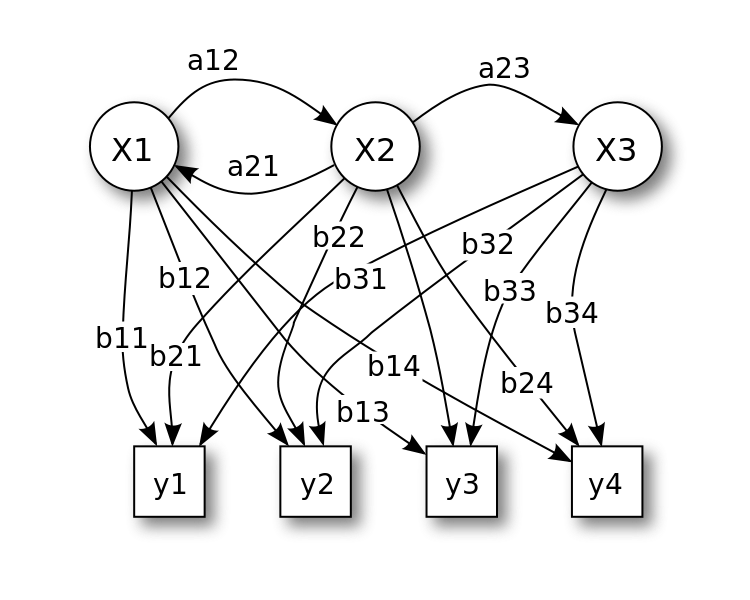
Pada *supervised learning*, *data training* disertai target pada setiap datanya, {xi,ti}, i = 1…N, dengan **x** adalah vector input dan t adalah target. Tujuan dari *supervised learning* adalah membangun model yang dapat memberikan hasil/output secara benar untuk suatu data input. *Supervised learning* digunakan untuk *classification, regression, ordinal regression, ranking,* dll.

**II.6.2 *Unsupervised Learning***

Pada *unsupervised Learning,* data training tidak disertai target pada setiap datanya, {xi}, i = 1…N, dengan **x** adalah vector input. Tujuan dari *unsupervised learning* adalah membangun model yang dapat menemukan variable tersembunyi dari *data training*. *Unsupervised learning* digunakan untuk *clustering, concept extraction, 0-97recommendation, density estimation, dimensionality reduction,* dll.

## II.7 Hidden Markov Model (HMM)

*Hidden Markov Model* (HMM) adalah perluasan dari rantai Markov dimana *state*nya tidak dapat diamati secara langsung (tersembunyi). Perubahan *hidden state* hanya dapat diobservasi melalui pengamatan variable lain.



Gambar II.14 *Hidden Markov Model* (HMM) [16]

Terdapat tiga masalah utaa yang dapat diselesaikan oleh Hidden Markov Model.

**II.7.1 Evaluasi**

Operasi evaluasi dalam HMM adalah perhitungan probabilitas dari urutan nilai observasi yang diberikan oleh *Hidden Markov Models.* Algoritma yang banyak digunakan untuk penyelesaian masalah observasi adalah algoritma maju (*Forward Algorithm*) yang keadaannya mengalir kedepan. Algoritma mundur (*Backward Algorithm*) yang keadaannya mengalir kebelakang dari observasi terakhir pada saat *T,* dan algoritma maju – mundur (*forward-backward algorithm*) yang merupakan gabungan dari algoritma *forward – backward.* .[10]

1. Algoritma maju

Jika variable forward , pada saat *t* dan state *i*, maka persamannya

(II.1)

Dengan *O* = indeks matrik terobservasi

Penyelesain dengan n keadaan dan observasi sampai T secara iterasi

* Inisialisasi :

, (II.2)

dimana = matrik state awal dan = matriks pertama yang terobservasi.

* Induksi :

… (II.3)

dimana :

= jumlah state

= matrik transisi.

* Terminasi :

(II.4)

1. Algoritma mundur

Keadaan mengalir ke belakang dari observasi terakhir saat *t.* Persamaan probabilitas mundur sebagai berikut:

(II.5)

dan dianalogikan dengan prosedur *forward* dengan langkah:

* Inisialisasi

(II.6)

* Induksi

, (II.7)

, (II.8)

**II.7.2 Pengkodean (*Decoding*)**

Permasalahan *decoding* ini yaitu menemukan barisan *state* terbaik (optimal) yang berasosiasi dengan barisan observasi dari sebuah model yang juga telah diketahui. Barisan *state* yang mempunyai probabilitas tertinggi dalam menghasilkan barisan observasi yang telah diketahui sebelumnya. Untuk menentukan keadaan tersembunyi dari suatu barisan observasi perlu digunakan suatu metode yang mempertimbangkan probabilitas transisi *state* pada proses pencarian barisan *state* yang paling optimal. Metode yang digunakan untuk penyelesaian masalah ini antara lain algoritma Viterbi. Langkah – langkah pengkodean sebagai berikut.

* Inisialisasi :

, (II.9)

* Rekursi :

, (II.10)

* Terminasi :

(II.11)

Dimana :

*P* = Probabilitas

**II.7.3 Learning**

Operasi learning dalam HMM adalah melatih parameter HMM jika diberikan data set barisan – barisan tertentu agar dapat menemukan himpunan transisi state yang paling mungkin beserta probabilitas output. Untuk menyelesaikan permasalahan terakhir pada HMM ini, biasanya digunakan algoritma *Baum-Welch* yang merupakan kasus khusus dari algoritma EM (Ekspektasi Maksimum). Algoritma EM sendiri merupakan algoritma yang digunakan untuk mempelajari model – model probabilistik dalam suatu kasus yang melibatkan keadaan – keadaan tersembunyi.

1. Parameter A

Parameter A ditunjukan dengan kumpulan status transisi yang membentuk matrik tertentu yaitu

(II.12)

bentuk

(II.13)

Merupakan peluang ketika state *j* pada waktu *t+1* jika pada waktu *t*  berada di state *i*

= Probabilitas transisi dari state i ke state j

*P*  = probabilitas

*n* = banyaknya hidden state dalam model

1. Parameter B

Parameter B ditunjukkan bahwa

(II.14)

yang merupakan

(II.15)

dan (II.16)

= Probabilitas distribusi matriks observasi

*m* = banyaknya simbol observasi yang berbeda pada tiap state

Pada *continuos density* HMM sering dikarakterisasi oleh fungsi kerapatan (*density function*) atau campuran fungsi kepadatan tertentu di setiap *state* . Dengan asumsi penggunaan *Gaussian* *Mixture*, kepadatan emisi *state* 𝑗 didefinisikan sebagai:

(II.17)

Dimana :

K = *number of mixture*

= *mixing coefficient*

untuk *Gaussian* pada *state* j dengan batasan stokastik dengan persamaan berikut.

(II.18)

Dimana :

= *Gaussian density function* dengan mean dan matrik kovarian untuk *campuran*.

1. Matriks awal di state *i*

Diperlukan inisialisasi matriks awal status yang ditunjukan oleh.

dengan :

(II.19)

Sehingga HMM dapat dilambangkan dengan .

## II.8 Support Vector Machine

Support Vector Machine (SVM) merupakan salah satu metode *machine learning* untuk *pattern recognition*, yang dalam penelitian ini adalah *image recognition*. Algoritma SVM pertama kali dikembangkan oleh Vladimir Vapnik (Bishop,2006). Konsep dasar SVM ialah mengklasifikasikan data menjadi dua kelas yang berbeda dengan cara membuat membuat *hyperplane* atau suatu bidang yang merupakan fungsi klasifikator antara dua kelas dengan menggunakan konsep memaksimumkan margin.

Bentuk umum klasifikator SVM ialah sebagai berikut :

y = *sign*(wTx + b) (II.20) Keterangan :

y merupakan nilai target pada setiap vector baris

x Rn merupakan vector yang dimensinya bergantung dari *n* banyaknya vitur

w Rn vector yang menjadi parameter bobot

b bias atau eror berupa skalar

Hyperplane yang dihasilkan SVM dapat menklasifikasikan data menjadi dua kelas, yaitu kelas positf dan kelas negative yang dimodelkan sebagai berikut :

wTxi + b ≥1, untuk yi bernilai 1 (II.21)

wTxi + b ≥1, untuk yi bernilai -1 (II.22)

dapat dirumuskan sebagai berikut

yi (wTxi + b) ≥1 (II.23)

Misal dibuat dua bidang yang sejajar dengan hyperplane, untuk kelas B+ dapat dirumuskan sebagai berikut,

B+ : wTxi + b = 1 (II.24)

Untuk kelas B- dapat dirumuskan sebagai berikut,

B- : wTxi + b = -1 (II.25)

Margin adalah jarak antara bidang B+ dan B-

X1

Hyperplane

yi = 1

B+ : wTxi + b = 1

B- : wTxi + b = -1

yi = -1

||W||

X2

Gambar II.15 *Hyperplane*

**II.8.1 Fungsi Kernel**

Pada umumnya masalah dalam domain dunia nyata jarang bersifat linear separable. Kebanyakan bersifat non linear. Oleh karena itu untuk menyelesaikan masalah non linear, SVM dapat dimodifikasikan dengan memasukkan fungsi kernel. Dalam non linear SVM, data dipetakan oleh fungsi Ф () ke ruang vector yang berdimensi lebih tinggi. Pada ruang vector yang baru ini, hyperplane yang memisahkan dua kelas tersebut dapat di konstruksikan.

Hal ini sejalan dengan teori Cover yang menyatakan “Jika suatu transformasi bersifat non linear dan dimensi dari feature space cukup tinggi, maka data pada input space dapat dipetakan ke feature space yang baru, dimana pattern-pattern tersebut pada probabilitas tinggi dapat dipisahkan secara linear” (Cover, 1965)

Hyperplane

Gambar II.16 Kernel pada SVM

Pada gambar diatas dapat dilihat bahwa pada kelas kuning dan pada kelas merah yang berada pada input space berdimensi dua tidak dapat dipisahkan secara linear. Selanjutnya ditunjukkan bahwa fungsi Ф memetakan tiap data pada input space tersebut ke ruang vector baru yang berdimensi lebih tinggi (dimensi 3), dimana kedua kelas dapat dipisahkan secara linear oleh sebuah hyperplane. Pemetaan ini dilakukan dengan cara menjaga topologi data, dalam artian dua data yang berjarak dekat pada input space, sebaliknya dua data yang berjarak jauh pada feature space.

**II.8.2 Multi class SVM**

*Hyperplane* yang dihasilkan SVM hanya bisa mengklasifikasikan dua kelas. Sedangkan pada kenyataannya akan banyak ditemukan kasus yang lebih dari dua kelas. Oleh karena itu dapat digunakan *Multiclass* SVM untuk mengklasifikasikan permasalahan yang memiliki lebih dari dua kelas. Terdapat beberapa pendekatan yang dapat menyelesaikan masalah *Multiclass* dengan menggunakan SVM, diantaranya dalah *One Vs All* (OVA) dan *One Vs One* (OVO).

**II.8.2.1 *One Vs All* (OVA)**

Memisahkan permasalahan yang ditemui dari *N* kelas. Sehingga akan dibuat *N decision boundary. Decision boundary* yang dihasilkan merupakan hasil dari pencarian *hyperplane* dari kelas ke *i* dengan kelas sisa yang lainnya. Berikut dilakukan simulasi dengan menggunakan tiga kelas.

A

A

B

B

A

A

B

B

C

C

C

C

Gambar II.17 Contoh pendekatan OVA

Pada gambar diatas terdapat tiga kelas yaitu kelas A, kelas B, dan kelas C. karena menggunakan metode OVA akan dibuat tiga *decision boundary* dari tiga kelas tersebut. Setiap kelas dicari *decision boundary* dengan kelas sisanya.

A

A

B

B

D1

B

A

A

B

C

C

C

C

Gambar II.18 *Decision boundary D1*pada pendekatan OVA

Pada gambar 2.5 dibuat *decision boundary* dari kelas A dengan kelas sisanya (kelas B dan C). Kelas A dilabelkan sebagai positif, sedangkan kelas sisanya dilabelkan menjadi negative, sehingga D1 merupakan decision boundary yang dihasilkan dari kelas A dan kelas sisanya (kelas B dan kelas C).

A

A

A

A

B

B

B

D2

B

C

C

C

C

Gambar II.19 *Decision boundary D2* pada pendekatan OVA

Selanjutnya, sesuai dengan Gambar 2.6, dibuat *decision boundary* dari kelas B dengan kelas sisanya (kelas A dan kelas C). Kelas B dilabelkan sebagai positif, sedangkan kelas sisanya dilabelkan menjadi negatif, sehingga D2 merupakan *decision boundary* yang dihasilkan dari kelas B dan kelas sisanya (kelas A dan kelas C).

B

B

A

A

A

A

D3

B

B

c

c

c

c

Gambar II.20 *Decision boundary D3* pada pendekatan OVA

Lalu yang terakhir adalah mencari *decision boundary* yang terakhir. Sesuai dengan gambar 2.7, dibuat *decision boundary* dari kelas C dengan kelas sisanya (kelas A dan kelas B). kelas C dilabelkan sebagai positif, sedangkan kelas sisanya dilabelkan menjadi negative, sehingga D3 merupakan *decision boundary* yang dihasilkan dari kelas C dan kelas sisanya (kelas A dan kelas B).

A

A

A

A

B

B

B

D2

D1

B

D3

C

C

C

C

Gambar II.21 *Decision boundary D3* pada pendekatan OVA

**II.6.2.2 *One Vs One* (OVO)**

Misalkan permasalahan yang ditemui *N* kelas. Sehingga akan dibuat *N(N-1)/2 Decision Boundary, decision boundary* yang dihasilkan merupakan hasil pencarian *hyperplane* dari setiap kelas yang lainnya. Berikut dilakukan simulasi dengan menggunakan tiga kelas.

C

C

C

B

B

B

A

A

B

A

A

C

Gambar II.22 contoh pendekatan OVO

Pada Gambar 2.8 terdapat tiga kelas, yaitu kelas A, elas B dan kelas C. Karena menggunakan metode OVO dibuat tiga *decision boundary* dari tiga kelas tersebut. Setiap kelas dicari  *decision boundary* dengan masing-masing kelas lainnya.

D1

B

B

A

A

B

A

A

B

Gambar II.22 *Decision Boundary D1* pada pendekatan OVO

Sesuai dengan Gambar 2.9, dibuat *decision boundary* dari kelas A dengan kelas B. kelas A dilabelkan sebagai positif, sedangkan kelas B dilabelkan menjadi negative, sehingga *D1*merupakan *decision boundary* yang dihasilkan kelas A dan elas B.

B

B

B

B

D2

C

C

C

C

Gambar II.12 *Decision Boundary D2* pada pendekatan OVO

Lalu selanjutnya, sesuai Gambar 2.10, dibuat *decision boundary* dari kelas B dengan kelas C. Kelas B dilabelkan sebagai positif, sedangkan kelas C dilabelkan menjadi negative, sehingga *D2* merupakan *decision boundary* yang dihasilkan dari kelas B dan kelas C.

A

A

C

C

C

A

A

D3

Gambar II.23 *Decision Boundary D3* pada pendekatan OVO

Lalu yang terakhir sesuai dengan Gambar 2.11, dibuat *decision boundary* dari kelas A dengan kelas C. Kleas A dilabelkan sebagai positif, sedangkan kelas C dilabelkan menjadi negative *D3*merupakan *decision boundary* yang dihasilkan dari kelas A dan kelas C.

A

D1

B

B

A

A

B

B

A

D3

C

C

D3

C

C

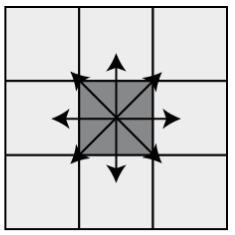
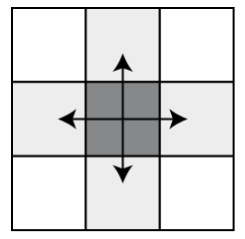
Gambar II.24 Hasil pendekatan OVO

Sehingga metode OVO menghasilkan area seperti Gambar 2.12, yang dibentuk oleh *decision boundary D1, D2 dan D3.*

**II.7 Chain Code**

Pada pengenalan objek, *chain code* adalah suatu atribut unik yang dapat diekstraksi dari suatu objek di dalam citra dengan cara menelusuri *pixel* batas objek tersebut berdasarkan arah-arah yang telah ditetapkan. *Chain code* pertama kali diusulkan oleh Freeman pada tahun 1961. Keluaran dari pengambilan *chain code* adalah angka-angka yang menunjukkan arah, yang merepresentasikan batas objek. Untuk mempermudah penentuan batas objek, pencarian *chain code* hanya dapat dilakukan pada citra biner. Pada kasus ini, objek dapat didefinisikan sebagai kumpulan *pixel* hitam yang terhubung membentuk satu kelompok.

Ada dua skema ketetanggaan yang dapat digunakan dalam pencarian *chain code*: *4-  
connectivity* dan *8-connectivity*. Skema *4-connectivity* hanya memperbolehkan pemeriksaan ketetanggaan pada empat arah utama (atas, kanan, bawah, dan kiri). *8-connectivity* memperbolehkan pemeriksaan ketetanggaan pada 8 arah (atas, atas-kanan, kanan, kanan-bawah, bawah, kiri-bawah, kiri, dan kiri-atas). Gambar 2.2 mengilustrasikan perbedaan kedua skema ini.



Gambar II.15 Chain Code dengan 4 – Connectivity dan 8 – Connectivity

Sistem ini menggunakan skema *8-connectivity* karena *chain code* yang dihasilkan lebih fleksibel dibandingkan dengan menggunakan skema *4-connectivity*. Untuk memangkuskan perubahan koordinat titik observasi, arah *chain code* dapat digunakan  
untuk mengubah koordinat x dan y dari titik observasi. Pada aplikasi ini, pemeriksaan pada setiap iterasi dimulai dari *pixel* di atas *pixel* yang sedang diperiksa dikunjungi dan dilakukan searah jarum jam (*clockwise*). Setiap arah diberi nilai antara 1 hingga 8, dengan 1 menyatakan arah awal kiri atas (atas, ↖), 2 menyatakan arah selanjutnya kearah atas dan seterusnya hingga 8 menyatakan arah terakhir yakni kiri.

*Chain code* memiliki empat kekurangan utama sebagai fitur rekognisi:

1. *Chain code* tidak bersifat *scale-invariant*. Objek yang sama dengan ukuran yang berbeda akan memiliki *chain code* yang berbeda, sehingga *chain code* huruf A dengan ukuran 36 tidak bisa digunakan untuk mengenali huruf A dengan ukuran 12.
2. *Chain code* tidak fleksibel. Objek dengan tipe yang berbeda akan menghasilkan *chain code* yang berbeda, contohnya huruf A dengan font Arial akan menghasilkan chain code yang berbeda dengan huruf A pada font Times New Roman, padahal keduanya adalah huruf A. Hal tersebut akan mengurangi keakuratan pengenalan, atau malah membuat pengenalan jadi tidak bisa dilakukan.

3. *Chain code* tidak dapat mendeteksi bagian dalam dari objek. Jika suatu objek memiliki satu atau lebih “lubang” yang membedakan objek tersebut dengan objek lain yang mirip, *chain code* tidak dapat digunakan. Hal ini terjadi karena *chain code* hanya memeriksa batas luar objek. Sebagai contoh, angka 0 *7-segment* dan angka 8 *7-segment* memiliki *chain code* yang sama (dengan asumsi ukurannya juga sama), namun seharusnya diklasifikasikan sebagai dua objek yang berbeda.

4. Tidak dapat merepresentasikan percabangan. Untuk merepresentasikan percabangan pada suatu objek (terutama pada objek yang sudah menjadi kerangka), *chain code* harus dibagi menjadi beberapa bagian, sesuai dengan jumlah cabang yang terbentuk.