PENGENALAN CITRA ALPHABET BERDASARKAN PARAMETER MOMEN INVARIAN DENGAN METODE *CASE-BASED REASONING*

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:

HANIFA VIDYA RIZANTI

NIM. 0910680078

KEMENTRIAN PENDIDIKAN DAN KEBUDAYAAN

PROGRAM STUDI TEKNIK INFORMATIKA

PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2013

LEMBAR PERSETUJUAN

PENGENALAN CITRA ALPHABET BERDASARKAN PARAMETER MOMEN INVARIAN DENGAN METODE *CASE-BASED REASONING*

SKRIPSI

Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:

HANIFA VIDYA RIZANTI

NIM. 0910680078

skripsi ini telah disetujui oleh dosen pembimbing pada tanggal 4 Juni 2013

Dosen Pembimbing:

Pembimbing I	**Pembimbing II**
Suprapto, ST, MT.	**Rekyan Regasari MP,ST,MT.**
NIP. 197107271996031001	**NIP. 770414 06 12 0253**

LEMBAR PENGESAHAN

PENGENALAN CITRA ALPHABET BERDASARKAN PARAMETER MOMEN INVARIAN DENGAN METODE *CASE-BASED REASONING*

SKRIPSI
Untuk memenuhi sebagian persyaratan mencapai gelar Sarjana Komputer

Disusun Oleh:
HANIFA VIDYA RIZANTI
NIM. 0910680078

Skripsi ini telah diuji dan dinyatakan lulus pada tanggal 14 Juni 2013

Dosen Penguji I	**Dosen Penguji II**
Dian Eka Ratnawati, S.Si., M.Kom	**Imam Cholissodin, S.Si., M.Kom.**
NIP. 19730619 200212 2 001	**NIK. 850719 16 1 1 0422**

Dosen Penguji III

Ahmad Afif Supianto, S.Si., M.Kom.
NIK. 820623 16 1 1 0425

Mengetahui,
Ketua Program Studi Informatika

Drs. Marji, MT.
NIP. 19670801 199203 1 001

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Hanifa Vidya Rizanti

NIM : 0910680078

Program Studi : Informatika

Fakultas : Program Teknologi Informasi dan Ilmu Komputer

Penulis Skripsi dengan Judul : **PENGENALAN CITRA ALPHABET BERDASARKAN PARAMETER MOMEN INVARIAN DENGAN METODE *CASE-BASED REASONING***

Dengan ini menyatakan bahwa:

Isi dari skripsi yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di dalam daftar pustaka dalam skripsi ini.

Apabila di kemudian hari ternyata skripsi yang saya tulis terbukti hasil jiplakan, maka saya bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan segala kesadaran dan penuh tanggung jawab dan digunakan sebagaimana mestinya.

Malang, 14 Juni 2013
Yang menyatakan,

Hanifa Vidya Rizanti
0910680078

KATA PENGANTAR

Alhamdulillahi rabbil 'alamin. Puji syukur penulis panjatkan kehadirat Allah SWT, karena atas segala rahmat dan limpahan hidayah-Nya, penulis dapat menyelesaikan skripsi yang berjudul "**PENGENALAN CITRA ALPHABET BERDASARKAN PARAMETER MOMEN INVARIAN DENGAN METODE *CASE-BASED REASONING***".

Dalam pelaksanaan dan penulisan skripsi ini penulis mendapatkan banyak bantuan dari berbagai pihak baik secara moril maupun materiil. Dalam kesempatan ini penulis ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

1. Ibunda Noorsanti, Ayahanda Syaiful Rizal, dan seluruh keluarga atas segenap dukungan dan kasih sayang yang telah diberikan.
2. Suprapto ST, MT, dan Rekyan Regasari MP, ST, MT selaku dosen pembimbing selama pelaksanaan skripsi.
3. Ir. Sutrisno, M.T, Ir. Heru Nurwasito, M.Kom, Himawat Aryadita, S.T, M.Sc, dan Eddy Santoso, S.Kom selaku Ketua, Wakil Ketua 1, Wakil Ketua 2 dan Wakil Ketua 3 Program Teknologi Informasi dan Ilmu Komputer Universitas Brawijaya.
4. Drs. Marji, M.T dan Issa Arwani, S.Kom, M.Sc selaku Ketua dan Sekretaris Program Studi Teknik Informatika Universitas Brawijaya.
5. Seluruh Dosen Teknik Informatika Universitas Brawijaya atas kesediaan membagi ilmunya kepada penulis.
6. Seluruh Civitas Akademika Teknik Informatika Universitas Brawijaya yang telah banyak memberi bantuan dan dukungan selama penulis menempuh studi di Teknik Informatika Universitas Brawijaya dan selama penyelesaian skripsi ini.
7. Sahabat-sahabat yang senantiasa mendukung, Milani Winangga, Winda Ayu Irianto, Fauziah Mayasari, Ari Agustina, Adestiana Rahmawati, Arianty Anggraeni, Dian Arisandi, Novelia Kharisma, dan Ervin Yohannes.

8. Seluruh pihak yang telah membantu kelancaran penulisan skripsi yang tidak dapat penulis sebutkan satu persatu.

Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak kekurangan baik format penulisan maupun isinya. Oleh karena itu, saran dan kritik membangun dari para pembaca senantiasa kami harapkan guna perbaikan bagi skripsi selanjutnya. Semoga skripsi ini dapat memberikan manfaat bagi semua pihak, Aamiin.

Malang, Juni 2013

Penulis

ABSTRAK

Hanifa Vidya Rizanti. 2009. PENGENALAN CITRA ALPHABET BERDASARKAN PARAMETER MOMEN INVARIAN DENGAN METODE *CASE-BASED REASONING*. Skripsi Program Studi Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya. Pembimbing: Suprapto, ST, MT. dan Rekyan Regasari MP,ST,MT.

Pengolahan citra digital dewasa ini banyak diaplikasikan di berbagai bidang, seperti kesehatan, pertanian, pemerintahan, dan pendidikan. *Optical Character Recognition* (OCR) merupakan salah satu objek penelitian citra digital yang terus dikembangkan oleh para peneliti. Proses pengenalan karakter diawali dengan *preprocessing* untuk perbaikan citra, dilanjutkan dengan segmentasi untuk mengelompokkan objek, kemudian ekstraksi fitur serta pengenalan setiap objek.

Preprocessing yang digunakan pada skripsi ini adalah binerisasi citra (gambar hitam-putih), dan operasi *closing* (dilasi lalu erosi). Selanjutnya, *Region Growing* digunakan sebagai metode segmentasi. Momen Invarian digunakan sebagai metode ekstraksi fitur, dan *training* data dengan *Case-Based Reasoning* (CBR) yang menggunakan *K-Nearest Neighbour* (KNN) sebagai metode klasifikasinya. Momen Invarian menghasilkan tujuh parameter berupa bilangan riil dari setiap objek citra digital. Kasus-kasus yang telah ada dikumpulkan, dipelajari oleh sistem (CBR), kemudian dikenali dengan klasifikasi K-NN. Metode Momen Invarian mempunyai kelebihan yakni invarian (parameternya tidak terpengaruh) terhadap skala, rotasi, dan translasi.

Hasil yang telah diperoleh melalui implementasi dan pengujian sistem adalah performa sistem cenderung meningkat (membaik) dengan semakin banyaknya data *training* yang digunakan. Akurasi terbaik yang diperoleh sistem adalah 89,74% dengan data training sebanyak 45 set huruf dan banyak tetangga proses K-NN (K) 26.

Kata kunci: citra digital, Momen Invarian, *Case-Based Reasoning*, *K-Nearest Neighbour*

ABSTRACT

Digital image processing nowadays being alot implemented in many subject, such as health, agriculture, goverment, and education. Optical Character Recognition (OCR) is one of the reseach objects in digital image processing which continuous to be developed by the researcher. Character recognition proccess begins with preprocessing for image reconstruction, followed by segmentation for grouping the object, and then features extraction and recognition for every object.

Preprocessing used in this research are binerization (black and white iamge) and closing operation (dilation then erosion). Next, Region Growing is used as segmentation method. Moment Invariant is used as feature extraction method, and the data training process use Case Based Reasoning (CBR) that implement K-Nearest Neighbour (KNN) as the classification method. Moment Invariant generates seven parameters in real number for every object in digital image. Cases that have been collected are learned by the system (CBR), and then recognized using K-NN Classification method. Moment invariant has advantages which is invariant (its parameters are not affected) by scaling, rotating, and translating.

The results obtained through implementation and testing are: system performance tended to increase (got better) as much as the number of training data which is used. The best accuration obtained from the test is 89.74% using 45 sets of each character in training data and 26 neighbours in K-NN method.

keywords: digital image, moment invariant, case-based reasoning, k-nearest neighbour

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Di dunia pendidikan saat ini banyak digunakan media-media pembelajaran interaktif dengan menggunakan komputer. Contohnya media pembelajaran anak untuk pengenalan huruf, kata, warna, dan bentuk. Beranjak dari hal tersebut, penulis memiliki ide untuk membuat sebuah media yang dapat membantu siswa dalam belajar menuliskan huruf. Namun sebelum sampai pada pembuatan aplikasi tersebut, dalam penelitian ini, penulis terlebih dahulu berkeinginan untuk meneliti bagaimana mengenali sebuah karakter yang diperoleh dari tulisan tangan siswa.

Dengan demikian skripsi ini nantinya akan melibatkan proses pengolahan citra digital berupa tulisan tangan sebuah huruf. Untuk melakukan hal tersebut, citra huruf diolah, mulai dari *preprocessing* citra hingga deteksi dan identifikasi, semuanya dilakukan secara bertahap dan diolah untuk mendapatkan hasil yang diinginkan. Sejalan dengan perkembangan teknologi, penelitian-penelitian dilakukan untuk dapat mendeteksi objek-objek dalam citra digital. Salah satu objek pengenalan citra digital adalah pengenalan karakter tulisan tangan (*handwriting recognition*) seperti pada penelitian Jong Oh [OHJ-01], Jaehwa Park [PAR-99], dan Kam-Fai Chan [CHA-98].

Sama seperti pengenalan objek-objek lain, pengenalan tulisan tangan diawali dengan *preprocessing* berupa perbaikan kontras dan *binarization image*. Setelah itu dilakukan proses deteksi karakter-karakter yang akan dikenali. Sebelum menuju proses pengenalan, jika perlu, akan dilakukan normalisasi terhadap karakter-karakter tersebut.

Pengenalan alphabet adalah dasar dari pengenalan karakter tulisan tangan. Banyak penelitian telah dilakukan terhadap pengenalan alphabet dan akan terus berkembang. Proses pengenalan alphabet sendiri mempunyai banyak metode. Metode-metode yang banyak digunakan misalnya jaringan saraf tiruan (*neural network*) dan *Hidden Markov Model (HMM)*. [SUK-01] Namun, pada sistem pengenalan kali ini, proses deteksi ciri suatu karakter akan diambil dari tujuh

parameter Momen Invarian. *Training* data dan pengenalan karakter tersebut akan dilakukan dengan metode *Case-Based Reasoning (CBR)*.

Jan Flusser, seorang peneliti *pattern recognition* asal Praha mengungkapkan berdasarkan banyak penulisannya mengenai Momen Invarian, bahwa Momen Invarian tidak terpengaruh (invarian) terhadap skala, translasi, dan rotasi. [FLU-05] Sisi menarik dari Momen Invarian ini menarik untuk diterapkan pada pengenalan alphabet.

Untuk menyempurnakan pengenalan citra alphabet ini,*case-based reasoning* (CBR) akan digunakan sebagai metode pembelajaran. CBR banyak diaplikasikan pada sistem pengambilan keputusan, dan mempunyai ciri khas mempelajari dari kasus-kasus sebelumnya. Setiap kali sistem melakukan kesalahan, pengguna memberi tahu sistem pembenaran dari kesalahan yang dilakukannya.

Berdasarkan keunggulan dari masing-masing Momen Invarian dan CBR, kedua metode inilah yang akan dipadukan pada penulisan skripsi ini. Dari penulisan ini, diharapkan alphabet dapat dikenali dengan tidak dipengaruhi skala, translasi, dan rotasi. Sistem diharapkan dapat belajar dari kesalahan dan kebenaran yang dilakukan pada kasus sebelumnya, untuk menyempurnakan penggunaan Momen Invarian. Hasil akhir penulisan skripsi ini diharapkan penggunaan Momen Invarian dan CBR akan memiliki performa yang bagus pada objek pengenalan citra alphabet.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka skripsi ini mempunyai rumusan masalah sebagai berikut:

1. Bagaimana merancang serta mengimplementasikan pengenalan citra alphabet berdasarkan parameter Momen Invariannya dengan metode *case-based reasoning* menjadi sebuah aplikasi?
2. Bagaimana pengaruh nilai k (banyak tetangga pada proses K-NN) terhadap performa aplikasi pengenalan citra alphabet berdasarkan parameter Momen Invariannya dengan metode *case-based reasoning*?

3. Bagaimana pengaruh banyaknya data *training* terhadap performa aplikasi pengenalan citra alphabet berdasarkan parameter Momen Invarian dengan metode *case-based reasoning*?
4. Bagaimana performa pengujian aplikasi pengenalan citra alphabet berdasarkan parameter Momen Invarian dengan metode *case-based reasoning* berdasarkan tingkat akurasinya?

1.3 Batasan Masalah

Berdasarkan latar belakang dan rumusan masalah yang telah dikemukakan, penelitian ini mempunyai batasan-batasan masalah sebagai berikut:

1. Citra yang diproses adalah hasil penggambaran *user* pada *picture box* yang telah disediakan oleh sistem.
2. Karakter yang dapat dikenali adalah alphabet kapital, yaitu A-Z, dengan jumlah 26 karakter.
3. Huruf yang digambar sebagai masukan tidak boleh saling tumpang tindih (*overlapping*).
4. Implementasi pengenalan citra alphabet ini juga mengeluarkan suara (*sound*) yang sesuai dengan karakter hasil pengenalan berdasarkan input *user*.
5. Implementasi pengenalan citra alphabet ini menggunakan bahasa pemrograman Visual C#.NET.
6. *Database Management System* (DBMS) yang digunakan adalah MySQL.

1.4 Tujuan

Tujuan yang ingin dicapai dalam pembuatan skripsi ini adalah mengetahui performa penggunaan parameter Momen Invarian dan *case-based reasoning* untuk mengenali citra alphabet yang diwujudkan dengan sebuah aplikasi.

1.5 Manfaat

Pengajuan skripsi ini diharapkan mempunyai manfaat yang baik dan berguna bagi pembaca dan penulis. Adapun manfaat yang diharapkan adalah sebagai berikut:

1. Bagi penulis
   a. Sebagai media untuk pengimplementasian ilmu pengetahuan teknologi khususnya di bidang *Optical Character Recognition (OCR)*.
   b. Mendapatkan pengetahuan dan wawasan pengolahan citra digital.
2. Bagi pembaca
   a. Mendapatkan wawasan akan pengimplementasian dari *Optical Character Recognition (OCR).*
   b. Mendapatkan wawasan akan penggunaan parameter Momen Invarian
   c. Mendapatkan wawasan akan bagaimana cara kerja *case-based reasoning* dalam *training* dan mengenali citra alphabet.

1.6 Sistematika Penyusunan Laporan

Sistematika penulisan penelitian ditunjukkan untuk memberikan gambaran dan uraian dari penyusunan skripsi secara garis besar yang meliputi beberapa bab, sebagai berikut.

BAB I : Pendahuluan

Menguraikan mengenai latar belakang, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika pembahasan.

BAB II : Kajian Pustaka dan Dasar Teori

Menguraikan tentang dasar teori dan referensi yang mendasari pembuatan sistem Pengenalan Citra Alphabet Berdasarkan Parameter Momen Invarian Dengan Metode *Case-Based Reasoning*.

BAB III : Metode Penelitian dan Perancangan

- Menguraikan tentang metode dan langkah kerja yang dilakukan dalam penulisan skripsi yang terdiri dari studi

literatur, metode pengambilan data, analisis kebutuhan, perancangan sistem, implementasi, pengujian dan analisis serta pengambilan kesimpulan.

- Menjabarkan perencanaan aplikasi yang dibuat, meliputi deskripsi aplikasi, spesifikasi kebutuhan, dan perancangan aplikasi sistem penentu kemiripan topik penelitian.

BAB IV : Implementasi

Membahas implementasi dari Pengenalan Citra Alphabet Berdasarkan Parameter Momen Invarian Dengan Metode *Case-Based Reasoning* yang sesuai dengan perancangan sistem yang telah dibuat.

BAB V : Pengujian dan analisis

Memuat hasil pengujian dan analisis terhadap Pengenalan Citra Alphabet Berdasarkan Parameter Momen Invarian Dengan Metode *Case-Based Reasoning* yang telah direalisasikan.

BAB VI : Penutup

Pada bab ini berisi kesimpulan yang diambil berdasarkan analisa sistem setelah pengujian, kelebihan atau kekurangan, serta saran–saran untuk penyempurnaan aplikasi yang dibuat.

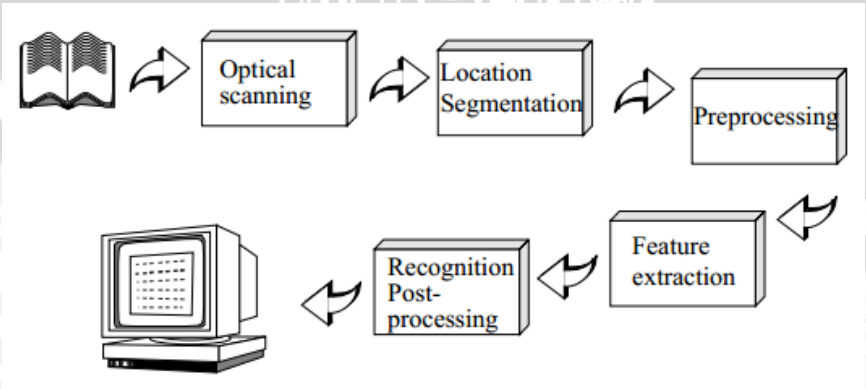
BAB II
KAJIAN PUSTAKA DAN DASAR TEORI

2.1 *Optical Character Recognition*

Secara umum, pengolahan citra digital menunjuk pada pemrosesan gambar 2 dimensi menggunakan komputer.Dalam konteks yang lebih luas, pengolahan citra digital mengacu pada pemrosesan setiap data 2 dimensi. Citra digital merupakan sebuah larik (*array*) yang berisi nilai-nilai real maupun komplek yang direpresentasikan dengan deretan bit terentu. [PUT-10]

Salah satu aplikasi yang paling penting dalam dunia pengolahan citra adalah pengenalan objek (*object recognition*). Aplikasi yang paling banyak dijumpai adalah OCR (*Optical Character Recognition*). [PUT-10] OCR digunakan untuk mengidentifikasi citra huruf yang kemudian diubah menjadi *file* berbentuk teks.

Sistem OCR terdiri atas beberapa komponen (Gambar 2.1). Tahap pertama proses yaitu mengubah dokumen menjadi citra digital dengan *scanner*. Ketika region (pada citra) yang mengandung teks ditemukan, setiap simbol/karakter diekstrak melalui proses segmentasi. Simbol/karakter yang telah diekstrak dapat dilakukan *preprocessing*, penghapusan *noise*, untuk memudahkan ekstraksi ciri di tahap selanjutnya. [EIK-93] Hasil ekstraksi ciri, selanjutnya, akan dibawa pada proses klasifikasi yang dilakukan untuk mengenali karakter/simbol berdasarkan metode serta data *training* yang ada.



Gambar 2.1 Komponen Sistem OCR

sumber: [EIK-93]

2.2 *Preprocessing*

Preprocessing adalah proses pengolahan citra dalam meningkatkan kualitas citra atau memodifikasi citra perlu dilakukan untuk meningkatkan kualitas penampakan atau untuk menonjolkan beberapa aspek informasi yang terkandung di dalam citra sehingga dapat meningkatkan kemungkinan dalam keberhasilan pada tahap pengolahan citra berikutnya.

2.2.1 Binerisasi

Binerisasi merupakan sebuah metode untuk mengubah citra keabuan menjadi citra biner sehingga objek yang diinginkan dalam gambar dapat terpisah dari latar belakangnya. Pengertian serupa juga diungkapkan oleh Davies (1990, p79) yang mengatakan bahwa binerisasi adalah sebuah metode untuk mengubah citra keabuan menjadi citra biner dengan objek terlihat hitam sedangkan latar belakangnya berwarna putih. Menurut Schalkoff (1989, pp167-168) untuk sebuah citra basil pindaian harus diubah terlebih dahulu tingkat keabuannya sebelum dimanipulasi agar objek menjadi lebih jelas.

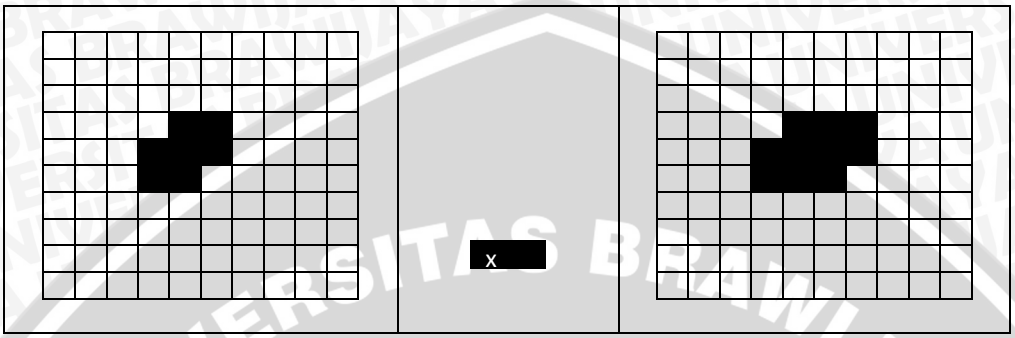
Tujuan dari binerisasi adalah untuk memisahkan piksel yang mempunyai nilai keabuan (*gray value*) lebih tinggi dengan yang lebih rendah. Piksel yang nilai keabuannya lebih tinggi akan diberi nilai biner J sedangkan piksel dengan nilai keabuan Iebih rendah akan diberi nilai biner 0. Pemberian nilai biner dapat pula dibalik, disebut binerisasi terbalik (*inverse threholding*), untuk Iebih memperjelas objek yang akan diteliti. Berdasarkan penentuan nilai ambangnya. binerisasi dibedakan menjadi 2 macam:

a. *fixed threshold*, nilai ambang dipilih secara independen.
b. *histogram derived thresholding*, nilai ambang ditentukan secara otomatis berdasarkan histogram.

2.2.2 Dilasi

Dilasi merupakan proses penggabungan titik-titik latar (0) menjadi bagian dari objek (1), berdasarkan *structuring element* (S) yang digunakan. dengan A adalah citra biner yang dikenakan operasi (dan mengalami perubahan bentuk pada obyeknya) dan B adalah citra yang melakukan operasi (dan tidak mengalami

perubahan bentuk). Citra B sering disebut elemen penstruktur (*structuring element*) dan komposisinya akan menentukan hasil dari operasi dilasi yang dilakukan.



(a) (b) (c)

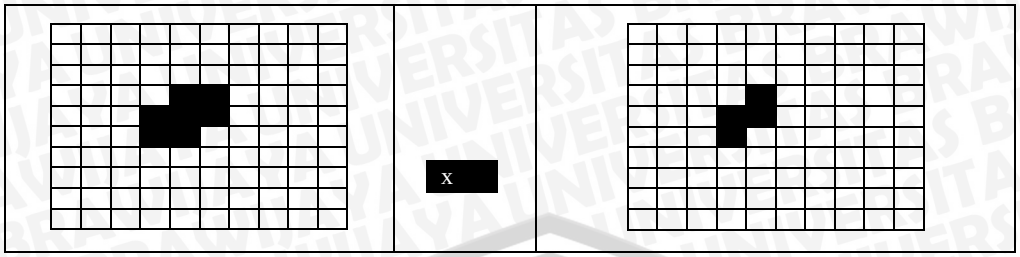
Gambar 2.2 Operasi dilasi A oleh B

(a) citra asli A (b) elemen penstruktur B (c) hasil operasi dilasi A oleh B

2.2.3 Erosi

Erosi merupakan operasi kebalikan dari dilasi, digunakan untuk menghapus atau mengurangi piksel-piksel obyek, atau untuk memperkecil ukuran obyek. Pada kasus citra biner, erosi akan menghapus piksel-piksel pada lapisan terluar obyek.

Suatu citra sebagai hasil operasi erosi mengandung piksel-piksel dimana elemen penstruktur yang ditranslasikan bersesuaian dengan piksel-piksel pada citra A. Jadi jelaslah bahwa hasil operasi erosi merupakan bagian dari citra aslinya, dimana piksel-piksel obyek yang tidak sesuai dengan pola yang digunakan maka pada citra hasil akan dihapus. Akan tetapi, situasi ini sangat tergantung pada elemen penstruktur dan penempatan titik origin. Hal yang perlu diingat adalah bahwa erosi bukan merupakan operasi negatif *(inverse)* dari dilasi.



(a) (b) (c)

Gambar 2.3 Operasi erosi A oleh B

(a) citra asli A (b) elemen penstruktur (c) hasil operasi erosi A oleh B

2.3 Thinning

Thinning merupakan metode yang digunakan untuk *skeletonizing* yang salah satu penggunaanya adalah dalam aplikasi *pattern recognition*. Terdapat cukup banyak algoritma untuk *image thinning* dengan tingkat kompleksitas, efisiensi dan akurasi yang berbeda-beda. Citra yang digunakan adalah citra biner, jika citra itu merupakan suatu citra *grayscale*, biasanya dilakukan *thresholding* terlebih dahulu sedemikian rupa sehingga citra tersebut menjadi citra biner. Citra biner adalah citra yang hanya memiliki 2 kemungkinan nilai pada setiap piksel-pikselnya, yaitu 0 atau 1. Nilai 0 adalah *background points*, biasanya bukan merupakan bagian dari citra sesungguhnya. Sedangkan nilai 1 adalah *region points*, yaitu bagian dari citra sebenarnya (bukan latar belakang). Citra hasil dari algoritma *thinning* biasanya disebut dengan *skeleton*.

Umumnya suatu algoritma *thinning* yang dilakukan terhadap citra biner seharusnya memenuhi kriteria-kriteria sebagai berikut:

- *Skeleton* dari citra kira-kira berada di bagian tengah dari citra awal sebelum dilakukan *thinning*.
- Citra hasil dari algoritma *thinning* harus tetap menjaga struktur keterhubungan yang sama dengan citra awal.
- Suatu *skeleton* seharusnya memiliki bentuk yang hampir mirip dengan citra awal.
- Suatu *skeleton* seharusnya mengandung jumlah piksel yang seminimal mungkin namun tetap memenuhi kriteria-kriteria sebelumnya.

2.3.1 Algoritma Zhang-Suen

Algoritma ini adalah salah satu algoritma *thinning* yang cukup populer dan telah digunakan sebagai suatu basis perbandingan untuk *thinning*. Algoritma ini cepat dan mudah diimplementasikan.

Setiap iterasi dari metode ini terdiri dari dua sub-iterasi yang berurutan yang dilakukan terhadap *contour points* dari wilayah citra. *Contour point* adalah setiap piksel dengan nilai 1 dan memiliki setidaknya satu *8-neighbor* yang memiliki nilai 0.

Dengan informasi ini, langkah pertama adalah menandai *contour point* p untuk dihapus jika semua kondisi ini dipenuhi:

(a) $2 \leq N(p_1) \leq 6$;

(b) $S(p_1) = 1$;

(c) $p_2 \cdot p_4 \cdot p_6 = 0$;

(d) $p_4 \cdot p_6 \cdot p_8 = 0$;

dimana $N(p_1)$ adalah jumlah tetangga dari p_1 yang tidak 0; yaitu,

$$N(p_1) = p_2 + p_3 + ... + p_8 + p_9$$

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

dan $S(p_1)$ adalah jumlah dari transisi 0-1 pada urutan $p_2, p_3, ..., p_8, p_9$.

Dan pada langkah kedua, kondisi (a) dan (b) sama dengan langkah pertama, sedangkan kondisi (c) dan (d) diubah menjadi:

(c') $p_2 \cdot p_4 \cdot p_8 = 0$;

(d') $p_2 \cdot p_6 \cdot p_8 = 0$;

Langkah pertama dilakukan terhadap semua *border pixel* di citra. Jika salah satu dari keempat kondisi di atas tidak dipenuhi atau dilanggar maka nilai piksel

yang bersangkutan tidak diubah. Sebaliknya jika semua kondisi tersebut dipenuhi maka piksel tersebut ditandai untuk penghapusan.

Piksel yang telah ditandai tidak akan dihapus sebelum semua *border points* selesai diproses. Hal ini berguna untuk mencegah perubahan struktur data. Setelah langkah 1 selesai dilakukan untuk semua *border points* maka dilakukan penghapusan untuk titik yang telah ditandai (diubah menjadi 0). Setelah itu dilakukan langkah 2 pada data hasil dari langkah 1 dengan cara yang sama dengan langkah 1 sehingga, dalam satu kali iterasi urutan algoritmanya terdiri dari:

(1) Menjalankan langkah 1 untuk menandai *border points* yang akan dihapus,
(2) hapus titik-titik yang ditandai dengan menggantinya menjadi angka 0,
(3) menjalankan langkah 2 pada sisa *border points* yang pada langkah 1 belum dihapus lalu yang sesuai dengan semua kondisi yang seharusnya dipenuhi pada langkah 2 kemudian ditandai untuk dihapus,
(4) hapus titik-titik yang ditandai dengan menggantinya menjadi angka 0.

Prosedur ini dilakukan secara iteratif sampai tidak ada lagi titik yang dapat dihapus, pada saat algoritma ini selesai maka akan dihasilkan *skeleton* dari citra awal.

2.4 Segmentasi *Region Growing*

Region Growing merupakan sebuah prosedur dimana sekumpulan piksel atau subregion hingga region yang lebih besar berdasarkan kriteria yang sudah didefinisikan Pendekatan dasar misalnya sekumpulan titik awal dan dari sini region tumbuh dengan menambahkan pada masing-masing *seed* ke piksel tetangga yang memiliki properti yang sama dengan *seed* tersebut.

Secara umum, langkah-langkah segmentasi *Region Growing* adalah sebagai berikut:

1. Tentukan beberapa *pixelseed*. Seed dapat ditentukan manual atau secara random.
2. Untuk setiap *pixelseed*, lihat 4 atau 8 tetangganya. Jika kriterianya sama maka tetangga tersebut bisa dianggap berada dalam 1 region/daerah dengan *pixelseed*.

3. Teruskan proses dengan mengecek tetangga dari tetangga yang sudah dicek.

2.5 Momen dan Momen Invarian

Momen Invarian dikenalkan oleh Hu pada tahun 1962 yang merupakan fungsi nonlinear yang invarian terhadap rotasi, translasi dan skala dan dideferensialkan dalam moment geometri foto. [GON-02]

Momen dapat menggambarkan suatu objek dalam hal area, posisi, orientasi dan parameter terdefinisi lainnya. Persamaan dasar dari momen suatu objek didefinisikan sebagai berikut.

Momen dapat menggambarkan suatu objek dalam hal area, posisi, orientasi dan parameter terdefinisi lainnya.Persamaan dasar dari momen suatu objek didefinisikan sebagai berikut.

$$m_{ij} = \sum_x \sum_y x^i y^j a_{xy} \tag{2.1}$$

sumber: [FLU-09]

dengan order dari momen adalah *(i + j). x* dan *y* menyatakan koordinat titik, sedangkan a_{xy} menyatakan intensitas titik. Momen tingkat ke-0 dan ke-1 (*zero and first-order moments*) didefinisikan sebagai berikut.

$$m_{00} = \sum_x \sum_y a_{xy} \tag{2.2}$$

$$m_{10} = \sum_x \sum_y x.a_{xy} \tag{2.3}$$

$$m_{01} = \sum_x \sum_y y.a_{xy} \tag{2.4}$$

sumber: [FLU-09]

Pada citra biner yang mana a_{xy} akan bernilai 0 atau 1, momen tingkat ke-0 (m_{00}) adalah sama dengan area dari objek. Pusat dari area atau massa (*centroid)* adalah parameter yang baik untuk menyatakan lokasi dari objek. Pusat area dari objek didefinisikan sebagai berikut.

$$x' = \frac{m_{10}}{m_{00}} \quad dan \quad y' = \frac{m_{01}}{m_{00}} \tag{2.5}$$

sumber: [FLU-09]

dengan *(x',y')* merupakan pusat koordinat dari objek.

Momen pusat (*central moment*) μ adalah momen yang bersesuaian dengan pusat area, didefinisikan sebagai berikut.

$$\mu_{ij} = \sum_x \sum_y (x - x')^i (y - y')^j a_{xy} \tag{2.6}$$

sumber: [FLU-09]

dan momen pusat yang ternormalisasi dinyatakan dengan persamaan berikut.

$$\eta_{ij} = \frac{\mu_{ij}}{(\mu_{00})^{\lambda}} \tag{2.7}$$

sumber: [FLU-09]

dengan $\lambda = \frac{(i+j)}{2} + 1,$ dengan *(i+j)* ≥ 2 (momen tingkat ke-1 adalah selalu invarian).

Dari momen ternormalisasi di atas, sekumpulan momen-Momen Invarian (*invariant moments*) dapat didefinisikan.Momen-momen ini sangat berguna dalam membuat vektor ciri untuk pengenalan objek. Berikut ini adalah persamaan dari momen-Momen Invarian:

$$\phi_1 = \eta_{20} + \eta_{02} \tag{2.8}$$

$$\phi_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2 \tag{2.9}$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{2.10}$$

$$\phi_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \tag{2.11}$$

$$\phi_5 = (\eta_{30} - 3\eta_{12})\ (\eta_{30} + \eta_{12})\ \{(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\} + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\{3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\} \tag{2.12}$$

$$\phi_6 = (\eta_{20} - \eta_{02})\{(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\} + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \tag{2.13}$$

$$\phi_7 = (3\eta_{21} - \eta_{30})(\eta_{30} + \eta_{12})\{(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\} + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\{3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\} \quad (2.14)$$

sumber: [FLU-09]

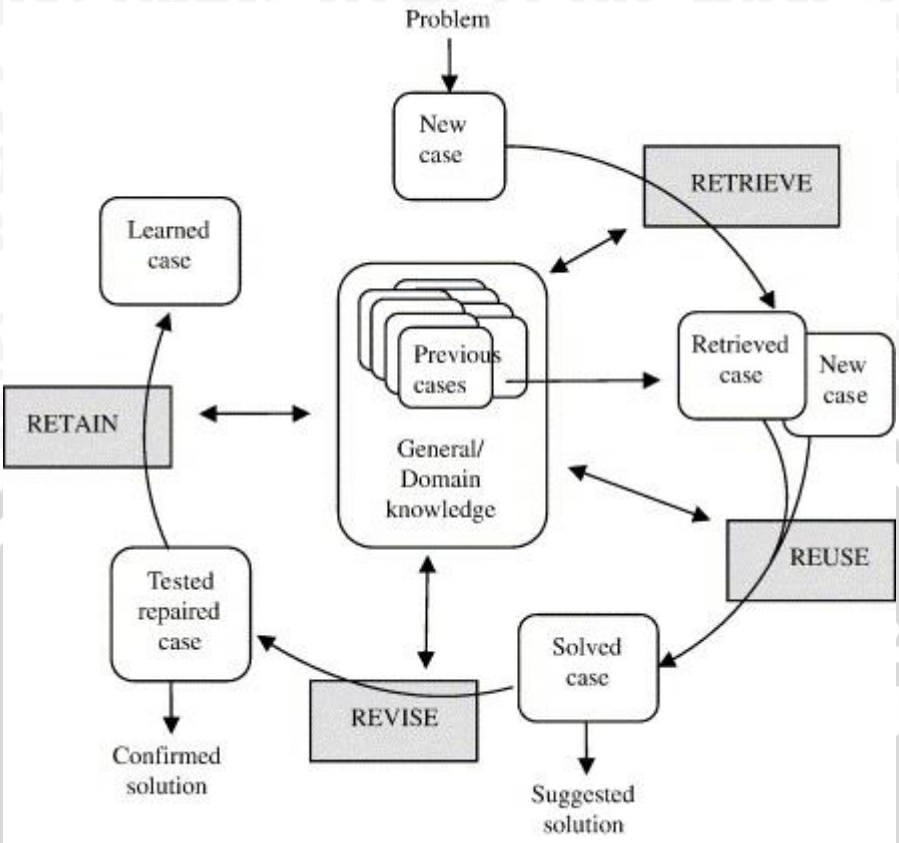
Tujuh Momen Invarian adalah invarian terhadap transformasi citra termasuk di dalamnya skala, translasi dan rotasi. Bagaimanapun juga tujuh Momen Invarian ini tidak invarian terhadap perubahan kontras.

2.6 *Case-Based Reasoning* (CBR)

CBR adalah sebuah metode pendekatan dari Sistem Penunjang Keputusan, yang terdiri dari mengacu kembali, menggunakan kembali, meninjau ulang, dan mendalami kasus yang telah lalu. Metode ini memproses permasalahan yang diajukan dengan menggunakan solusi pada kasus sebelumnya yang memiliki persamaan. Proses tersebut akan menghasilkan solusi yang telah dikembangkan dan disesuaikan untuk mengatasi permasalahan.

Terdapat empat tahap penyelesaian masalah pada sistem CBR, yaitu:

1. *Retrieve*: penelusuran kasus-kasus yang dialami sebelumnya, dipilih kasus yang dianggap mirip dengan kasus yang saat ini dialami.
2. *Reuse*: menyalin atau memodifikasi solusi dari kasus mirip yang terjadi.
3. *Revise*: mengadaptasi solusi yang diberikan untuk menyelesaikan masalah.
4. *Retain*: solusi baru akan dimasukkan sebagai pengetahuan, setelah kebenarannya terkonfirmasi dan tervalidasi. [SHI-04]



Gambar 2.4 CBR *life-cycle*

sumber: [SHI-04]

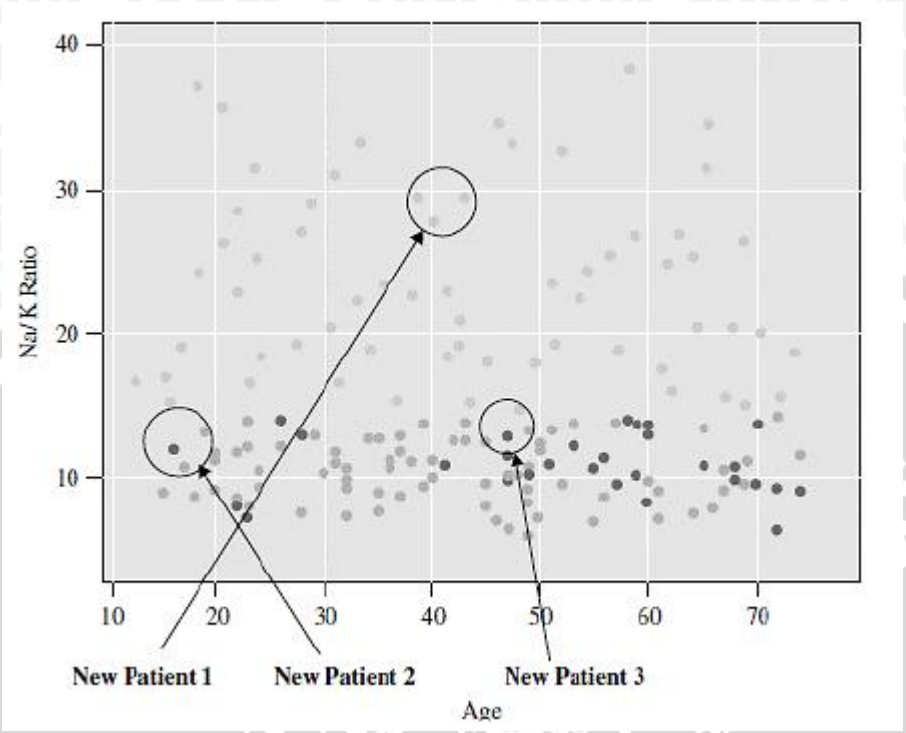
2.7 *K-Nearest Neighbour*

Metode K-Nearest Neighbor (k-NN atau KNN) adalah sebuah metode untuk melakukan klasifikasi terhadap objek berdasarkan data yang jaraknya paling dekat dengan objek tersebut. Metode ini merupakan metode yang paling umum digunakan untuk estimasi dan prediksi. [LAR-05]

Keunggulan dari metode KNN ini adalah relatif tidak terpengaruh dari *error* dari data dan juga dapat digunakan dengan kumpulan data dengan jumlah besar. Namun kekurangan metode ini adalah proses pelaksanaannya lambat.

Algoritma KNN dapat dilihat pada contoh berikut. Kita ingin mengklasifikasi tipe obat untuk resep seorang pasien berdasarkan karakteristik pasien seperti umur dan rasio sodium/potassium (Na/K). Gambar 2.3 merepresentasikan plot dari umur pasien dan rasio Na/K. Resep yang mengandung obat tertentu dibedakan berdasarkan warna lingkarannya. Titik abu-abu muda

melambangkan obat Y. Titik abu-abu normal melambangkan obat A dan X. Titik abu-abu gelap melambangkan obat B dan C.

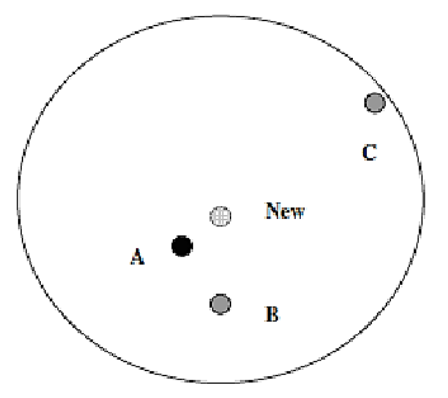


Gambar 2.5 Contoh pasien berdasarkan umur dan rasio Na/K

sumber: [LAR-05]

Misalkan saat ini kita memiliki data pasien baru, tanpa klasifikasi obat, dan ingin mengklasifikasi resep untuk pasien tersebut berdasarkan pasien lain yang memiliki atribut yang sama. Identifikasi pasien 1 berumur 40 dan rasio Na/K 29. Berdasarkan Gambar 2.3 pasien 1 berada di posisi dalam lingkaran New Patient 1. Dari lingkaran tersebut kita mengklasifikasi resep untuk pasien 1 adalah obat Y karena titik di sekitar pasien 1 semuanya diberikan resep yang sama.

Berikutnya pasien 2 berumur 17 tahun dengan rasio Na/K 12,5. Gambar 2.4 memperlihatkan pembesaran dari lingkaran New Patient 2 di Gambar 2.3 beserta dengan letak deskripsi pasien 2. Misalkan kita menentukan bahwa k = 1 untuk algoritma KNN, jadi pasien 2 akan diklasifikasikan berdasarkan satu titik yang berada paling dekat dengan titik pasien 2 yaitu titik A dengan resep obat B dan C.

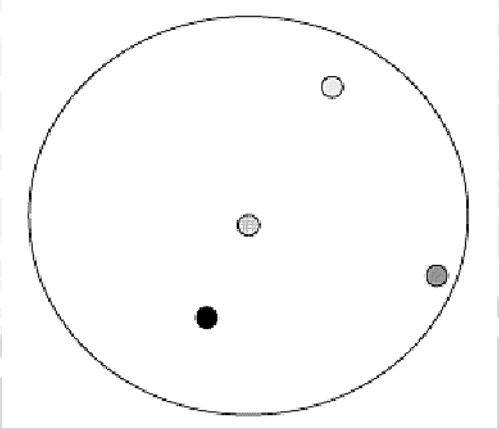


Gambar 2.6 Pembesaran gambar New Patient 2 pada Gambar2.3

sumber: [LAR-05]

Namun, apabila menentukan k = 2 maka pasien 2 akan diklasifikasikan berdasarkan 2 titik yang berada paling dekat, tetapi tidak terdapat hasil yang ditemukan karena titik A dan titik B merupakan 2 titik yang berbeda. Namun, jika k = 3 maka pasien 2 akan diklasifikasi berdasarkan 3 titik yang paling dekat. Berdasarkan Gambar 2.4 ada 1 titik abu-abu gelap dan 2 titik abu-abu normal. Maka pasien 2 akan diklasifikasi sama dengan titik abu-abu normal karena jumlahnya lebih banyak jadi resep obat A dan X.

Terakhir, pasien 3 berumur 47 tahun dan rasio Na/K 13,5. Gambar 2.5 memperlihatkan pembesaran dari lingkaran New Patient 3 di Gambar 2.3 beserta dengan letak deskripsi pasien 3. Untuk nilai k = 1 maka pasien 3 akan diklasifikasi berdasarkan titik abu-abu gelap maka resep obat B dan C. Untuk k = 2 tidak ditemukan hasil karena 2 titik berbeda, demikian juga dengan k = 3 karena 3 titik tersebut berbeda satu sama lain.



Gambar 2.7 Pembesaran gambar New Patient 3 pada Gambar 2.3

sumber: [LAR-05]

Berdasarkan contoh kasus di atas KNN mengklasifikasi data baru berdasarkan jarak titik yang paling dekat dengan posisi titik yang baru. Untuk mengukur jarak kedekatan antara data yang baru dengan data-data yang lama digunakan rumus *distance*. Rumus *distance* yang paling umum dipakai

$$d_{Euclidean}(x,y) = \sqrt{\sum_i (x_i - y_i)^2} \quad (2.15)$$

di mana x = x_1, x_2, …, x_m dan y = y_1, y_2, …, y_m. x dan y adalah nilai data yang ingin dihitung kedekatan jaraknya dan m adalah jumlah atribut data yang ada. Seperti contoh, misal A adalah x_1 = 20 tahun dan rasio Na/K x_2 = 12, sedangkan B adalah y_1 = 30 tahun dan rasio Na/K y_2 = 8. Maka Euclidean *distance* antara 2 data tersebut adalah

$$d_{Euclidean}(x,y) = \sqrt{\sum_i (x_i - y_i)^2} = \sqrt{(20-30)^2 + (12-8)^2} \quad (2.16)$$

$$= \sqrt{100 + 16} = 10{,}77 \quad (2.17)$$

Untuk mengidentifikasi suatu data baru berdasarkan beberapa macam data seperti contoh di atas, maka diperlukan combination function untuk menentukan keputusan mengklasifikasi data baru tersebut. *Combination function* tersebut dibagi menjadi 2 yaitu: [LAR-05]

- *Simple Unweighted Voting*

  1. Sebelum melakukan algoritma KNN, tentukan dahulu nilai k yang akan dipakai
  2. Kemudian gunakan Euclidean *distance*. Data yang terdekat yang dipilih
  3. Setelah mendapatkan data yang akan digunakan sebagai acuan sejumlah nilai k, jarak kedekatan data baru terhadap data lama tidak berguna lagi. Kemudian dilakukan penghitungan *voting* dari data yang sudah didapatkan sebelumnya

  Contoh penggunaan *Simple Unweighted Voting* ini seperti yang terlihat dari contoh kasus di atas pada New Patient 2 dengan menggunakan nilai k = 3.

- *Weighted Voting*

  Pada *weighted voting*, data yang jarak kedekatannya dengan data baru lebih jauh tidak memiliki nilai *voting* yang sama dengan data yang jarak kedekatannya lebih dekat. Data yang jarak kedekatannya lebih dekat memiliki nilai *voting* yang lebih besar. Rumus untuk mendapatkan *Weighted Voting* yaitu:

$$Votes(y) = \frac{1}{d(x,y)^2} \tag{2.18}$$

Tabel 2.1 Contoh pasien berdasarkan umur dan rasio Na/K

Record	Age	Na/K	Age_{MMN}	Na/K_{MMN}
New	17	12,5	0,05	0,25
A (*dark gray*)	16,8	12,4	0,0467	0,2471
B (*medium gray*)	17,2	10,5	0,0533	0,1912
C (*medium gray*)	19,5	13,5	0,0917	0,2794

sumber: [LAR-05]

Dari contoh New Patient 2 dengan k = 3 dari Gambar 2.4. Diasumsikan nilai dari contoh di atas seperti terlihat pada Tabel 2.1

beserta dengan hasil Min-Max *normalizations*. Kemudian ditemukan nilai *distance*-nya sebagai berikut:

$$d(new, A) = \sqrt{(0{,}05 - 0{,}0467)^2 + (0{,}25 - 0{,}2471)^2 = 0{,}004393}$$
$$d(new, B) = \sqrt{(0{,}05 - 0{,}0533)^2 + (0{,}25 - 0{,}1912)^2 = 0{,}58893}$$
$$d(new, C) = \sqrt{(0{,}05 - 0{,}0917)^2 + (0{,}25 - 0{,}2794)^2 = 0{,}051022}$$

Data A memberi *voting* untuk data baru sebagai abu-abu gelap (obat B dan C) jadi perhitungan *weighted voting*-nya

$$votes\ (darkgray) = \frac{1}{d(new, A)^2} = \frac{1}{0{,}004393^2} \cong 51{,}818$$

Untuk data B dan C member *voting* untuk data baru sebagai abu-abu normal (obat A dan X) jadi perhitungan *weighted* voting-nya:

$$votes(mediumgray) = \frac{1}{d(new, B)^2} + \frac{1}{d(new, C)^2}$$
$$= \frac{1}{0{,}058893^2} + \frac{1}{0{,}051022^2} \cong 672$$

Jadi berdasarkan nilai *weighted voting* maka data baru termasuk sebagai abu-abu gelap (obat B dan C) karena nilainya *voting*-nya lebih besar. hal ini berlawanan dengan hasil klasifikasi dengan menggunakan *simple unweighted voting*. Namun dengan adanya penambahan perhitungan *weighted voting* maka proses komputer akan berjalan lambat karena perlu dilakukan perhitungan ulang terhadap jarak kedekatan dan *weighted voting*-nya setiap kali ingin mengklasifikasi data yang baru.

2.5 Kajian Pustaka

Banyaknya manfaat pengenalan karakter dalam kehidupan sehari-hari, menarik perhatian para peneliti untuk terus mengembangkan sistem OCR dengan

berbagai metode ekstraksi ciri maupun klasifikasi. Sejalan dengan perkembangan berbagai metode pada ranah pengolahan citra, penelitian-penelitian terkait OCR juga tidak berhenti begitu saja, akan selalu berkembang untuk memperoleh sistem pengenalan yang lebih efektif dan efisien. Berikut adalah penelitian-penelitian terkait OCR:

Tabel 2.2 Penelitian terkait OCR

Sumber:	**Tahun**	**Judul**	**Metode**	**Hasil**
[NAG-66]	1966	*Self-Corrective Character Recognition System*	*Self-Corrective Algorithm*	*error* menurun hingga 0,7%
[STE-67]	1967	*Two Operations in Character Recognition: Some Evidence from Reaction-Time Measurements*	*Reaction-Time Function*	waktu reaksi lebih lambat menggunakan *Test-Stimulus*
[STA-76]	1976	*Approaches to Chinese Character Recognition*	berbagai metode ekstraksi ciri karakter mandarin	penggunaan metode yang berbeda untuk *stroke*, coretan, bulatan.
[YHA-81]	1981	*An On-Line Chinese Character Recognition System*	metode pada elektronik tablet (tidak disebutkan)	*recognition rate*: 91,1%
[WHI-83]	1983	*Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction*	*Dynamic Treshold*	penerapan penghapusan *background* kasus *bank-check*

[FUJ-92]	1992	*Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis*	*Pattern-Oriented Segmentation Algorithm*	hasil segmentasi 99,96%
[TRI-95]	1995	*Feature Extraction Methods for Character Recognition-A Survey*	ektraksi fitur, *Neural Network*	penggunaan berbagai metode untuk berbagai jenis karakter
[CHA-98]	1999	*Recognizing on-line handwritten alphanumeric characters through flexible structural matching*	ektraksi struktur, *structural matching*	*recognition rate*: 92,74%
[KIM-00]	2000	*Feature Selection Using Genetic Algorithms for Handwritten Character Recognition*	*Genetic Algorithm*	*recognition rate*: 96,89%
[THI-06]	2006	*Character Segmentation-by-Recognition Using Log-Gabor Filters*	*Log-Gabor Filters*	*recognition rate*: 80,5%
[CHO-11]	2011	*Improving The Character Recognition Efficiency of Feed Forward Bp Neural Network*	*neural network*	*correctly recognized*: 92,30%
[YOU-12]	2012	*Hidden Markov Model with Parameter-Optimized K-Means Clustering for Handwriting Recognition*	HMM, K-Means	akurasi 78% - 83,5%

Sebagai salah satu metode OCR, Momen Invarian Hu juga mengalami perkembangan melalui beberapa penelitian pada berbagai objek, yang dikombinasikan dengan metode lainnya:

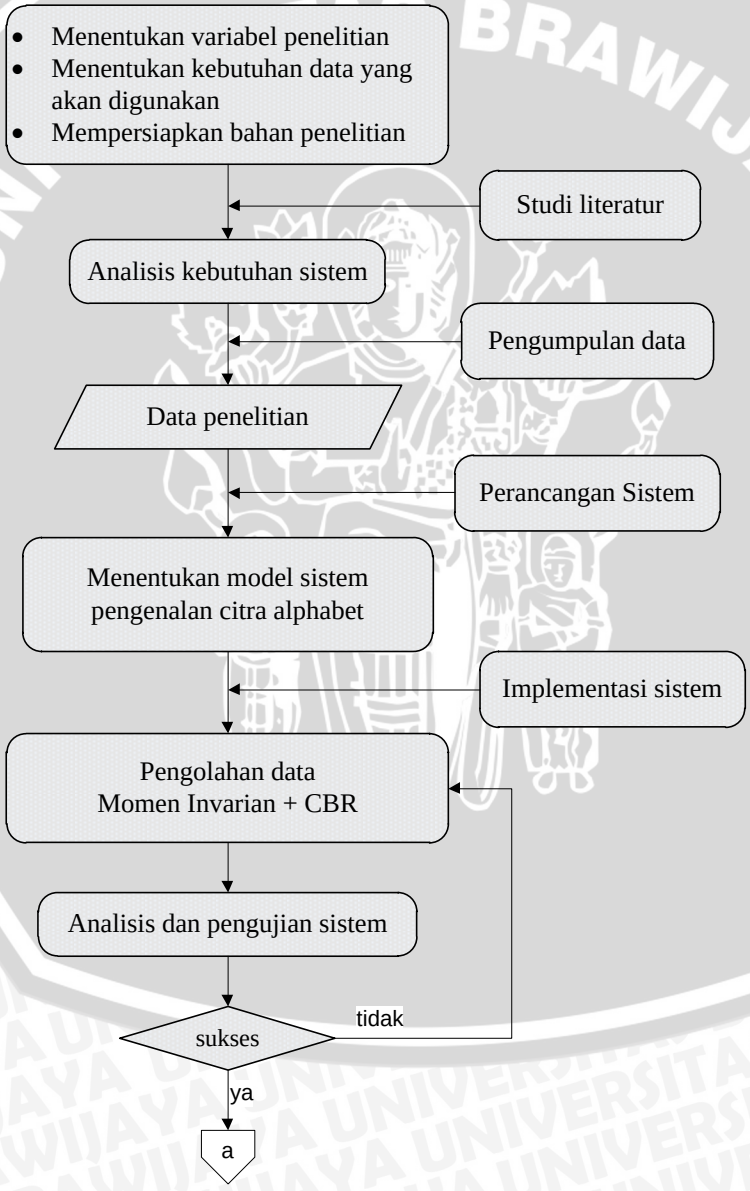
Tabel 2.3 Penelitian terkait Momen Invarian

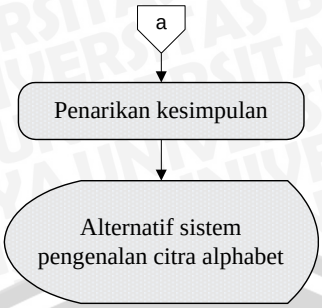
Sumber:	**Tahun**	**Judul**	**Objek**	**Hasil**
[WON-93]	1993	*Generation of Moment Invariants and Their Uses for Character Recognition*	*Character recognition*	*recognition rate* 98,8%
[DEH-97]	1997	*Farsi Handwritten Character Recognition With Moment Invariants*	*Farsi characters*	*error rate* 3,06%, faktor diskriminan 96,92%
[MER-05]	2005	*Real object recognition using moment invariants*	*shape recognition*	*performance* 98,14%
[SAN-11]	2011	*Recognition of Isolated Handwritten Kannada Characters Using Invariant Moments and Chain Code*	*Kannada characters*	akurasi 88,46%
[KEF-12]	2012	*Comparative Study of the Use of Geometrical Moments for Arabic Handwriting Recognition*	*Arabic characters*	*recognition accuracy* 89%

BAB III
METODE PENELITIAN DAN PERANCANGAN

3.1. Metode Penelitian

Pada sub-bab ini akan dibahas metode-metode yang akan digunakan dalam pembuatan sistem pengenalan citra alphabet berdasarkan parameter Momen Invarian dengan metode CBR. Adapun desain penelitiannya digambarkan pada diagram alir sebagai berikut:





Gambar 3.1 Desain Penelitian Sistem Pengenalan Citra Alphabet

3.1.1 Studi Literatur

Studi literatur digunakan untuk mengumpulkan dan mempelajari berbagai referensi yang bersumber dari buku, artikel *online*, jurnal ilmiah, laporan penelitian yang telah ada. Referensi utama yang digunakan untuk mendukung penulisan skripsi ini meliputi:

1. Pemahaman tentang *Optical Character Recognition*
2. Tujuh parameter Momen Invarian
3. Metode *Case-Based Reasoning*
4. Klasifikasi *weighted K-Nearest Neighbour*
5. Bahasa pemrograman *desktop* Visual C#.NET
6. *Database* MySQL

3.1.2 Metode Pengumpulan Data

Data yang dibutuhkan untuk penelitian ini adalah data berupa karakter alphabet tulisan tangan.

Data penilaian memerlukan validasi dari manusia secara langsung, sehingga memerlukan data primer. Data primer adalah data yang didapatkan langsung dari responden penelitian. Metode pengumpulan data primer yang bersifat kuantitatif menggunakan instrumen kuesioner. Kuesioner adalah sejumlah pertanyaan tertulis yang digunakan untuk memperoleh informasi dari responden tentang hal-hal yang diketahui.

3.1.3 Analisa Kebutuhan Sistem

Analisis pengetahuan bertujuan untuk menganalisis dan mendapatkan kebutuhan-kebutuhan yang diperlukan dalam perancangan aplikasipengenalan citra alphabet berdasarkan parameter Momen Invarian dengan metode *Case-Based Reasoning*.

Metode analisis yang digunakan adalah *procedural analysis* dengan menggunakan bahasa pemodelan prosedural. Pemrograman berbasis prosedur merupakan teknik pemrograman yang dikembangkan berdasarkan algoritma untuk memecahkan suatu masalah. Algoritma merupakan cara-cara yang ditempuh dalam memanipulasi data sehingga masalah yang dihadapi bisa dipecahkan.

Kebutuhan yang digunakan dalam pembuatan sistem penentuan kemiripan skripsi ini diantaranya

1. Kebutuhan *hardware*, meliputi:
   - Laptop atau PC
2. Kebutuhan *software*, meliputi:
   - Microsoft Windows 7 Home Premium sebagai sistem operasi
   - MySQL sebagai server *Database Management System*
   - Microsoft Visual Studio 2008 sebagai aplikasi untuk pembuatan sistem menggunakan bahasa pemrograman C#.NET.
3. Data yang dibutuhkan yaitu karakter alphabet tulisan tangan responden.

3.1.4 Implementasi

Pada tahap ini akan dilakukan penerapan desain sistem. Sistem ini dibangun dengan bahasa pemrograman Visual C#.NET yang dikoneksikan dengan *database* MySQL untuk menyimpan tujuh parameter Momen Invarian yang diperoleh dari coretan user. Implementasi sistem pengenalan citra alphabet ini meliputi:

1. Penerapan komponen-komponen *Optical Character Recognition (OCR).*
2. Penerapan Momen Invarian yang menghasilkan tujuh parameter dari setiap citra yang digunakan sebagai ciri untuk mengenali citra.

3. Penerapan metode *weighted K-Nearest Neighbour* untuk mengklasifikasikan karakter alphabet.
4. Penerapan *case-based reasoning* untuk proses *training*.
5. Pembuatan *user interface* aplikasi pengenalan citra alphabet untuk *end-user*.

3.1.5 Pengujian dan Analisa Sistem

Pengujian sistem dilakukan untuk menguji kelayakan aplikasi pengenalan citra alphabet yang telah dibuat dan menunjukkan bahwa aplikasi bekerja dengan baik sesuai spesifikasi sistem yang telah ditentukan.

1. Akurasi *recogniton rate* berdasarkan penggunaan nilai k (banyak tetangga) yang berbeda-beda.
2. Akurasi *recognition rate* berdasarkan banyak set *data training* yang digunakan. (*n* set x 26 alphabet).

Analisis terhadap sistem dilakukan untuk menarik kesimpulan dari hasil dari pembuatan aplikasi dan pengujian sistem pengenalan citra alphabet berdasarkan parameter Momen Invarian dengan metode *Case-Based Reasoning*. Analisa dan evaluasi sistem pengenalan citra alphabet ini bertujuan untuk menguji performa penggunaan parameter Momen Invarian dan *Case-Based Reasoning* sebagai kombinasi pada pengenalan citra alphabet.

3.1.6 Penarikan Kesimpulan

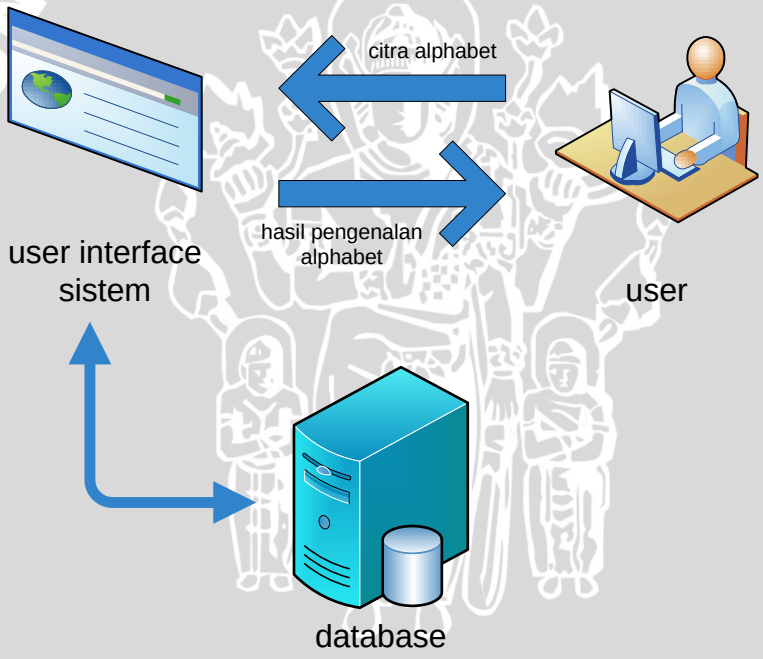
Penarikan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi dan pengujian sistem aplikasi telah selesai dilakukan. Kesimpulan diambil dari hasil pengujian dan analisis terhadap sistem yang dibangun.Tahap terakhir dari penulisan adalah saran. Saran bertujuan untuk memperbaiki kesalahan-kesalahan yang terjadi dan menyempurnakan penulisan serta untuk memberikan pertimbangan atas pengembangan aplikasi selanjutnya.

3.2. Perancangan Sistem

Perancangan sistem bertujuan untuk tahap awal pengimplementasian desain sistem dan perancangan *interface* yang digunakan. Perancangan sistem dibangun berdasarkan hasil studi literatur dan analisis kebutuhan sistem.Tahap perancangan sistem meliputi arsitektur berbasis desktop pada aplikasipengenalan citra alphabet berdasarkan parameter Momen Invarian dengan metode *Case-Based Reasoning*.

3.2.1 Arsitektur Sistem Pengenalan Citra Alphabet

Aplikasi pengenalan citra alphabet berbasis *desktop* dapat melayani penyimpanan, pemrosesan, dan penggunaan informasi. Gambar 3.2 menunjukkan arsitektur sistem pengenalan citra alphabet berbasis *desktop*.



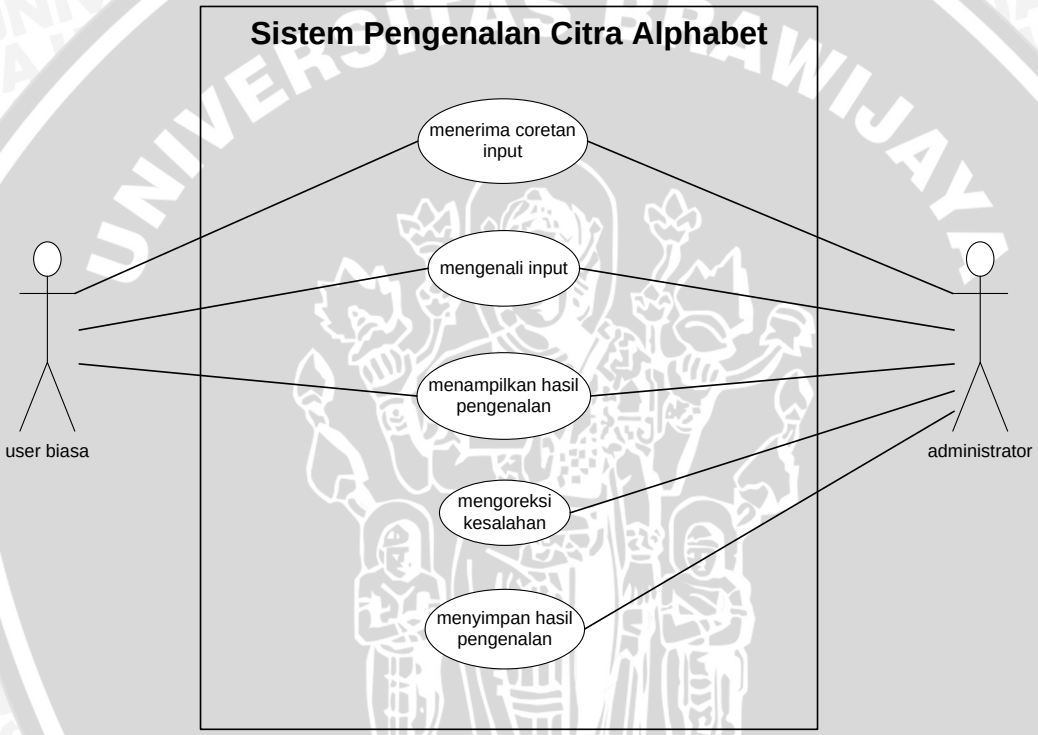
Gambar 3.2 Arsitektur sistem pengenalan citra alphabet

3.2.2 Analisis Kebutuhan

Aplikasi yang akan dibuat pada penelitian ini adalah aplikasi yang digunakan untuk mengenali tulisan tangan. Sistem yang akan dibuat mempunyai fungsi-fungsi sebagai berikut:

1. Menerima *input* dari *user* berupa coretan pada kanvas.
2. Sistem dapat mengenali lebih dari satu inputan *user*.
3. Administrator dapat mengoreksi kesalahan pengenalan sistem.
4. Administrator dapat menyimpan hasil pengenalan yang benar
5. Sistem dapat menampilkan hasil pengenalan.

Kebutuhan sistem ini diterjemahkan ke dalam *use case* diagram pada Gambar 3.3.



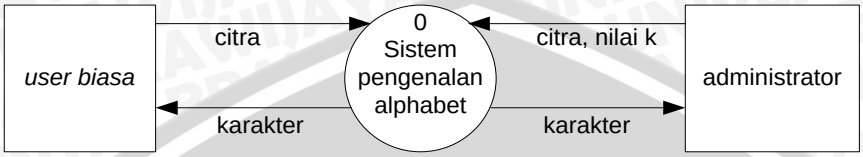
Gambar 3.3 *Use case* sistem pengenalan citra alphabet

3.2.3 *Data Flow Diagram*

Data flow diagram menggambarkan aliran arus data dari suatu proses, entitas, dan penyimpanan (*storage*). Berikut ini adalah *data flow diagram* sistem pengenalan citra alphabet.

3.2.3.1. *Data Flow Diagram* Level 0

Data flow diagram level 0 sistem pengenalan citra alphabet ditunjukkan pada gambar 3.4.

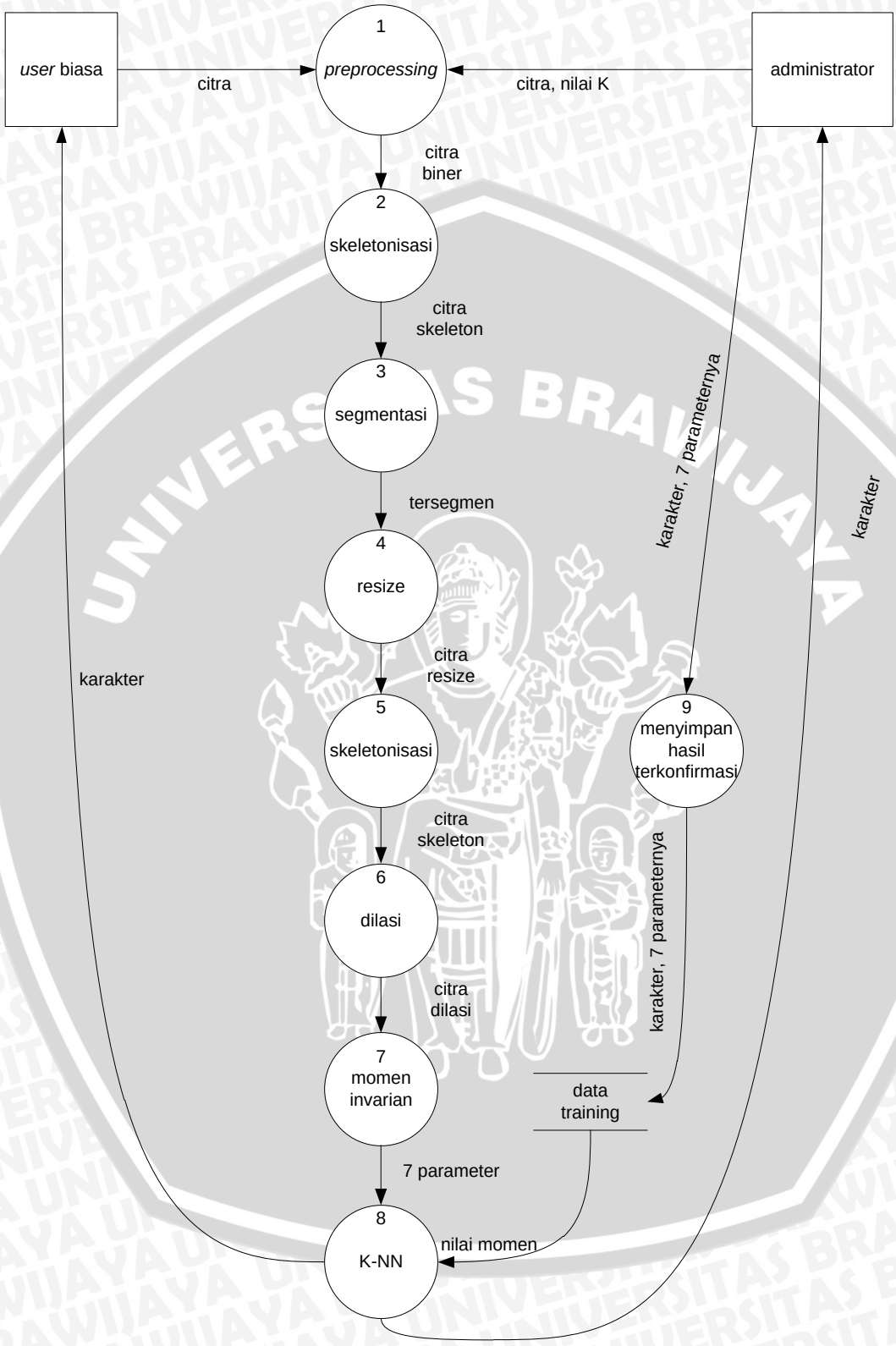


Gambar 3.4 *Data flow diagram* level 0 sistem pengenalan citra alphabet

Sistem pengenalan citra alphabet ini diawali oleh administrator yang memasukkan data *training* berupa citra. Selanjutnya, administrator mendapatkan karakter hasil *training*. *User* biasa (*end user*) dapat memasukkan citra hasil coretan pada kanvas dan mendapatkan hasil pengenalan. *User* biasa tidak memasukkan nilai k (banyak tetangga pada proses K-NN) karena nilai k telah ditetapkan oleh administrator.

3.2.3.2. *Data Flow Diagram* Level 1

Data flow diagram level 1 sistem pengenalan citra alphabet ditunjukkan pada Gambar 3.5.



Gambar 3.5 *Data flow diagram* level 1 sistem pengenalan citra alphabet

Administrator memasukkan data *training* mentah berupa citra-citra latih ke dalam proses pelatihan, yang menghasilkan nilai-nilai momen dan disimpan ke *database*.

User biasa memasukkan data uji berupa citra yang diambil dari coretan pada kanvas yang disediakan. Pada proses *preprocessing*, citra tersebut diubah menjadi citra biner, yang selanjutnya akan disegmentasi dan menghasilkan segmen-segmen citra. Setiap segmen citra akan diekstraksi ciri dengan metode Momen Invarian untuk memperoleh tujuh parameter Momen Invarian, yang akan dibandingkan dengan tujuh parameter Momen Invarian dari *database* data *training* pada proses K-NN. Baik administrator maupun *user* biasa dapat melakukan proses pengenalan ini dan memperoleh hasil berupa karakter alphabet.

3.2.4 Representasi Data dan Rancangan Basis Data

Sistem pengenalan citra alphabet ini membutuhkan data training berupa tujuh nilai parameter Momen Invarian dari citra-citra latih yang digunakan. Ketujuh parameter tersebut disimpan di dalam *database* "skripsi". *Database* "skripsi" memiliki satu tabel, yaitu tabel "momen". Berikut ini adalah struktur tabel "momen":

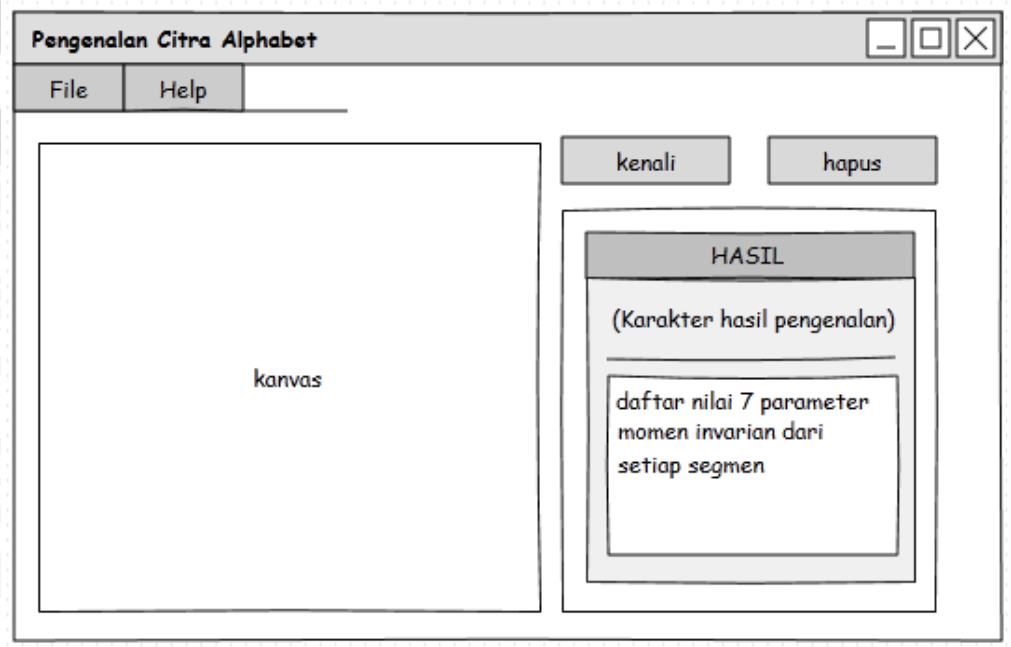
Tabel 3.1 Atribut dan tipe data *database* sistem

Column	**Type**	**Index**
id	int	PRIMARY
huruf	char(1)	-
p1	float	-
p2	float	-
p3	float	-
p4	float	-
p5	float	-
p6	float	-
p7	float	-

3.2.5 Rancangan *User Interface*

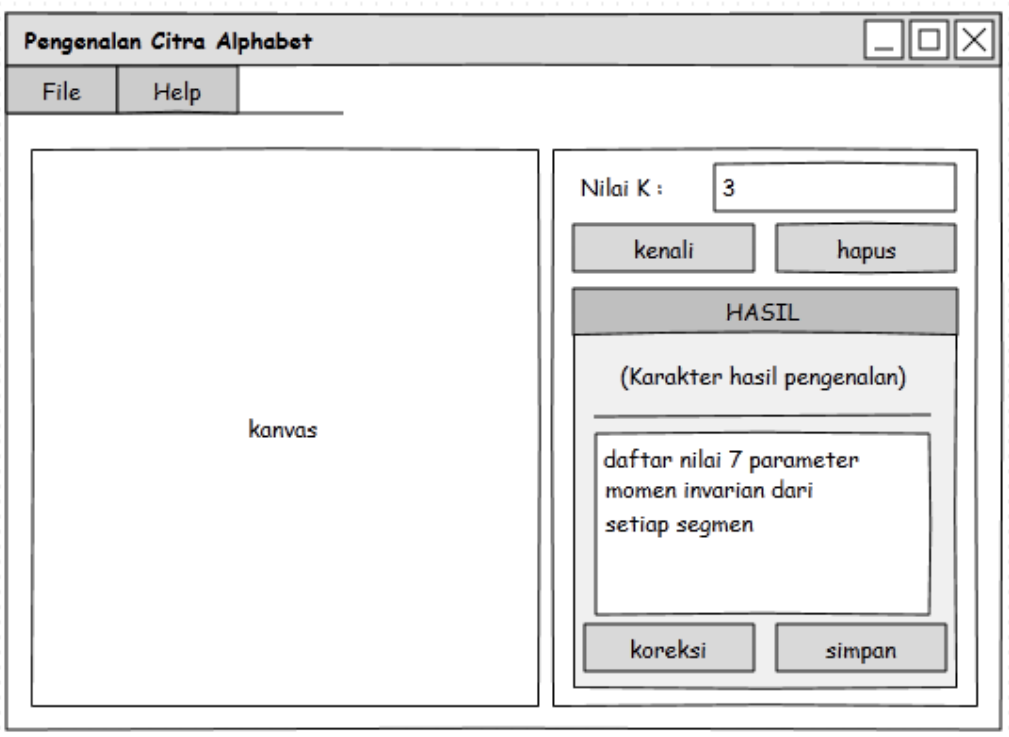
Tujuan perancangan desain *user interface* ini bertujuan agar *user* dapat nyaman dalam menggunakan aplikasi yang akan dirancang. Perancangan desain *user interface* aplikasi pengenalan citra alphabet berbasis *desktop* ini memiliki beberapa hal:

1. *User interface* administrator dan *user* biasa:
   - *Menubar* yang terdapat beberapa fasilitas seperti *help*, *about*, dan *quit*.
   - Kanvas putih sebagai media *user* menggambar alphabet. Jika *pointer* diarahkan pada kanvas, user harus menekan tombol kiri *mouse* untuk dapat menggambar pada kanvas.
   - Label untuk menampilkan abjad hasil pengenalan sistem.
   - RichTextBox untuk menampilkan daftar nilai tujuh parameter Momen Invarian dari setiap segmen.
   - Tombol "kenali" yang ditekan *user* setelah menggambar alphabet pada kanvas untuk mengenali citra.
   - Tombol "hapus" untuk membersihkan kanvas seperti awal.



Gambar 3.6 Rancangan GUI *user* biasa sistem pengenalan citra alphabet

2. *User interface* administrator:
   - *TextBox* untuk memasukkan nilai k (banyak tetangga).
   - Tombol "koreksi" yang harus ditekan *user* ketika sistem menampilkan hasil pengenalan yang salah. *User* dapat memberi tahu sistem alphabet yang sebenarnya.
   - Tombol "simpan" untuk menyimpan hasil pengenalan yang sudah benar ke dalam *database*.



Gambar 3.7 Rancangan GUI administrator sistem pengenalan citra alphabet

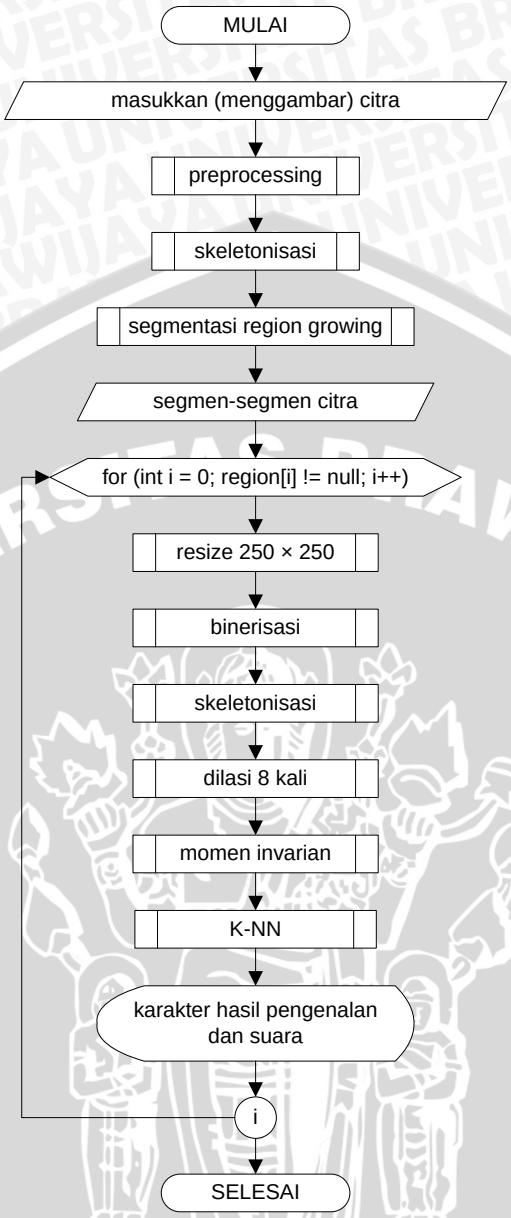
3.3. Alur Proses

Sistem pengenalan citra alphabet ini akan menerapkan gabungan dari beberapa metode yang telah dipelajari, sehingga sistem yang akan diimplementasikan mempunyai rancangan sebagai berikut:

1. User memasukkan citra, lalu menekan tombol 'kenali'.
2. Citra *input user* dimasukkan ke *preprocessing* yang terdiri dari:

- Binerisasi
- Operasi *closing*: Dilasi dan Erosi

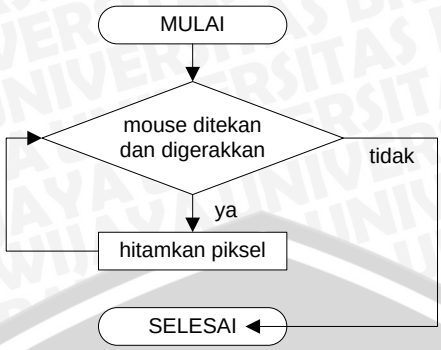
3. Citra hasil *preprocessing* dimasukkan ke proses skeletonisasi untuk diambil morfologinya.
4. Citra hasil skeletonisasi dimasukkan ke proses segmentasi *Region Growing*, menghasilkan segmen-segmen.
5. Untuk setiap segmen, selanjutnya dilakukan:
   a. *Resize* ukuran segmen menjadi 250 × 250 piksel.
   b. Binerisasi segmen yang telah di-*resize*.
   c. Skeletonisasi.
   d. Dilasi 8 kali.
   e. Ekstraksi ciri, mencari tujuh nilai parameter Momen Invarian.
   f. Klasifikasi K-NN untuk mengenali alphabet, berdasarkan tujuh nilai parameter Momen Invarian.
   g. Menampilkan alphabet hasil pengenalan dan memutar suara.



Gambar 3.8 *General flowchart* sistem pengenalan citra alphabet

3.3.1 *Input* Citra

Citra *input* diperoleh dari coretan *user* pada kanvas yang disediakan di *user interface*. Proses pengambilan *input* ini seperti menggambar pada Ms. Paint. Jika *mouse* ditekan dan digerakkan, piksel pada kanvas akan diwarnai hitam, dengan tebal *brush* 8 pt.



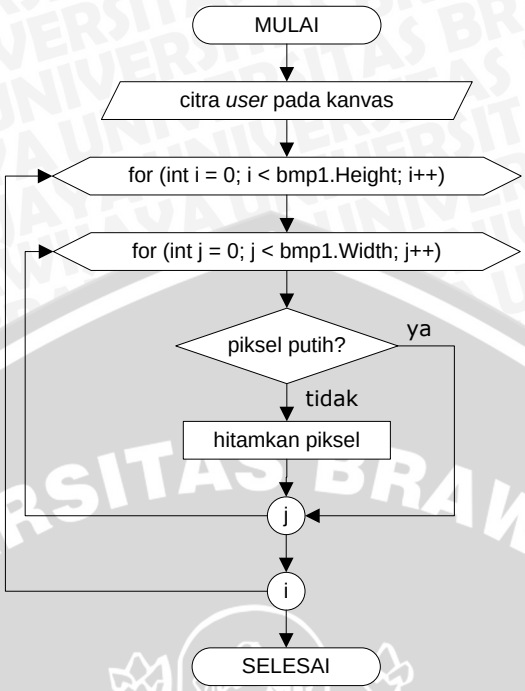
Gambar 3.9 *Flowchart input* citra sistem pengenalan citra alphabet

3.3.2 *Preprocessing*

Preprocessing pasa sistem ini terdiri dari operasi binerisasi dan *closing*. Operasi binerisasi bertujuan untuk mengubah citra menjadi dua warna saja, yaitu hitam dan putih. Operasi *closing* bertujuan untuk menyempurnakan citra *input user* yang kurang sempurna, misalnya terputus-putus. Citra *input* yang digunakan memiliki tingkat kecerahan 24bit, terdiri dari 8bit *red*, 8bit *green*, dan 8bit *blue*.

3.3.2.1 Binerisasi

Operasi binerisasi mengubah citra menjadi dua warna saja, yaitu hitam dan putih. Jika warna suatu piksel tidak putih, maka piksel tersebut diwarnai hitam. Jika piksel tersebut putih, tidak dilakukan perubahan warna (tetap putih).



Gambar 3.10 *Flowchart* binerisasi citra sistem pengenalan citra alphabet

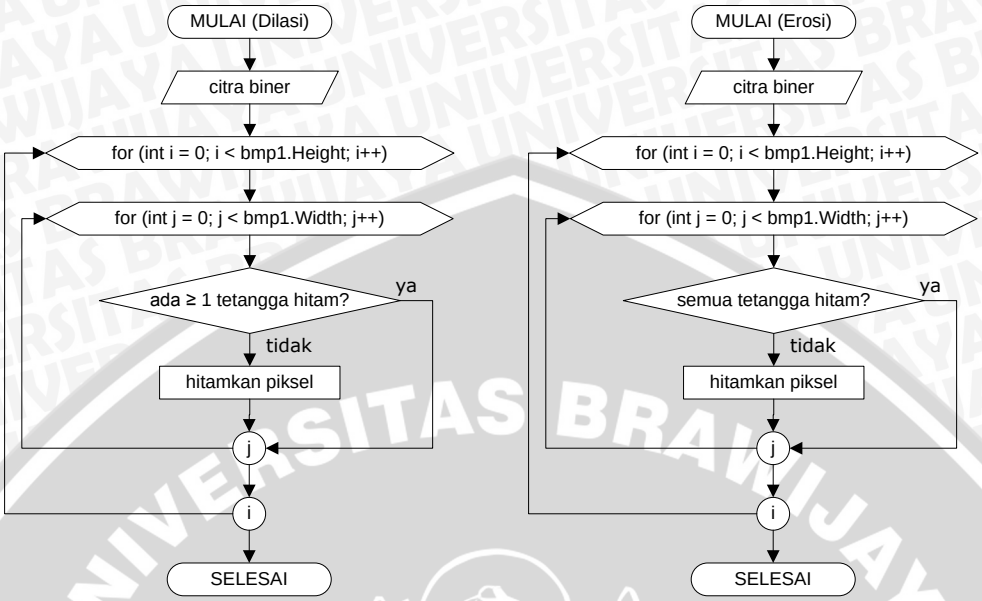
3.3.2.2 *Closing* (Dilasi dan Erosi)

Operasi *closing* terdiri dari dua operasi utama, yaitu dilasi dan erosi. Dilasi adalah operasi yang bertujuan untuk mempertegas kedudukan objek terhadap *background*, sedangkan erosi untuk memperkecil objek. Singkatnya, operasi *closing* menyempurnakan citra *input user* dengan memperbesar (dilasi) lalu diperkecil kembali (erosi).

Pada sistem pengenalan citra alphabet ini, *background* berwarna putih, dan objek berwarna hitam, dengan proses dilasi-erosi menggunakan kedekatan 8 tetangga. Pada proses dilasi, jika suatu piksel memiliki sekurang-kurangnya satu tetangga yang berwarna hitam, maka piksel tersebut diberi warna hitam. Pada proses erosi, jika suatu piksel memiliki delapan tetangga yang semuanya berwarna hitam, maka piksel tersebut diwarnai hitam.

(x-1, y-1)	(x, y-1)	(x+1, y-1)
(x-1, y)	(x, y)	(x+1, y)
(x-1, y+1)	(x, y+1)	(x+1, y+1)

Gambar 3.11 Delapan ketetanggaan proses dilasi dan erosi



Gambar 3.12 *Flowchart* operasi *closing* sistem pengenalan citra alphabet

3.3.3 *Thinning* (*Zhang Suen*)

Operasi *thinning* bertujuan untuk mendapatkan morfologi dari citra alphabet yang diinputkan *user*. Metode *thinning* yang digunakan adalah Zhang Suen. *Input* untuk operasi *thinning* ini adalah citra yang telah dibinerkan, dan *output*-nya berupa citra morfologi berukuran 1 piksel.

Jika pada Bab II subbab *Thinning* menjelaskan bahwa 0 (hitam) adalah *background* dan 1 (putih) adalah objek, maka sebaliknya, sistem pengenalan citra alphabet ini menggunakan 0 sebagai objek dan 1 sebagai *background*. Maka, keempat syarat untuk *thinning* pun berubah sesuai keadaan sistem tersebut.

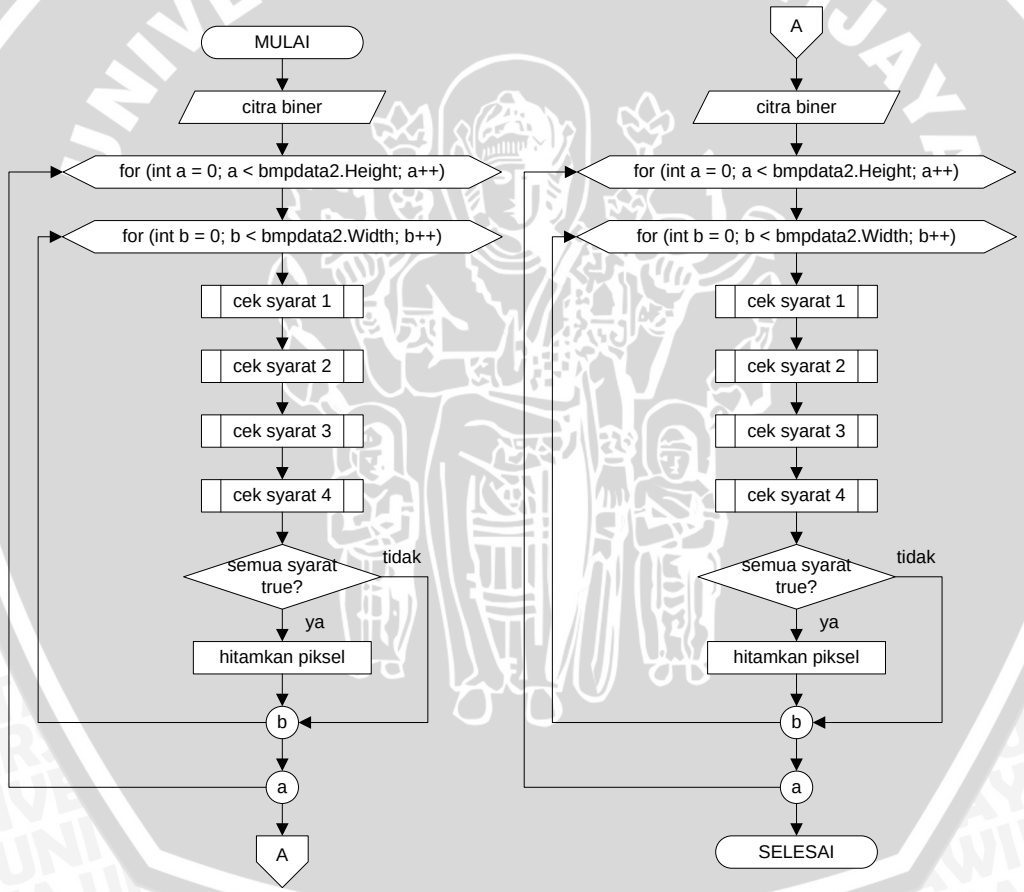
Syarat pertama adalah jumlah kedelapan tetangga yang berwarna sama dengan objek adalah 2 sampai 6. Pada sistem ini, warna objek adalah 0 (hitam), sehingga jika piksel hitam yang diproses mempunyai 2 sampai 6 tetangga yang juga hitam, maka syarat pertama bernilai *true*, selain itu *false*.

Syarat kedua adalah jumlah perubahan (transisi) warna tetangga-tetangga. Pada Bab II subbab *Thinning*, transisi yang digunakan dari 0 (hitam) ke 1 (putih). Sebaliknya, pada sistem ini transisi yang digunakan dari 1-0. Jika hanya ada satu transisi, syarat kedua bernilai *true*, selain itu *false*.

Sedangkan syarat ketiga dan syarat keempat tidak berubah dari teori pada Bab II subbab *Thinning*. Jika ketiga piksel semuanya bernilai 0 atau semuanya bernilai 1, maka nilai syarat ketiga adalah *false*, selain itu *true*.

Prosses *thinning* ini dilakukan sebanyak 2 kali iterasi. Perbedaan iterasi pertama dan kedua adalah pada syarat ketiga dan keempat, yang menghendaki 3 piksel tetangga yang berbeda sebagai syarat. Sedangkan syarat pertama dan kedua sama untuk kedua iterasi.

Kedua iterasi ini dilakukan terus-menerus, berulang-ulang hingga tidak ada satupun piksel objek (hitam, 0) yang berubah warna menjadi *background* (putih, 1).



Gambar 3.13 *Flowchart* operasi *thinning* sistem pengenalan citra alphabet

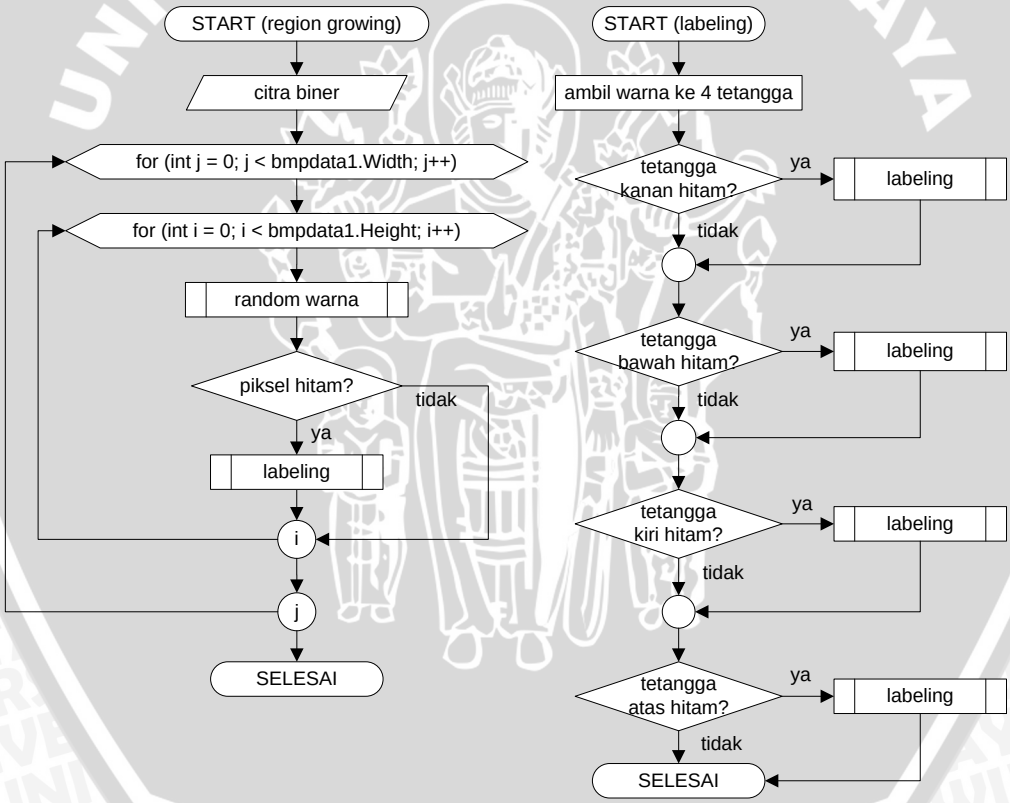
3.3.4 Segmentasi (*Region Growing*)

Sistem pengenalan citra alphabet ini menggunakan segmentasi *Region Growing*. Segmentasi bertujuan untuk memperoleh segmen-segmen yang

dihasilkan dari coretan *user* pada kanvas. Segmen-segmen itulah yang nantinya akan dikenali satu-persatu sebagai alphabet.

Segmentasi *Region Growing* pada sistem ini memiliki masukan berupa citra biner hasil *preprocessing*. Sistem akan menandai piksel hitam (objek) yang pertama kali ditemukan sebagai *seed*. Label yang digunakan untuk masing-masing segmen adalah warna yang diambil random dan berbeda-beda. Sistem akan melakukan pengecekan terhadap empat tetangganya secara berurutan (kanan → bawah → kiri → atas). Jika tetangga tersebut berwarna hitam, akan ditandai sebagai *seed* baru.

Hasil proses segmentasi ini berupa segmen-segmen yang disimpan dalam bitmap-bitmap.

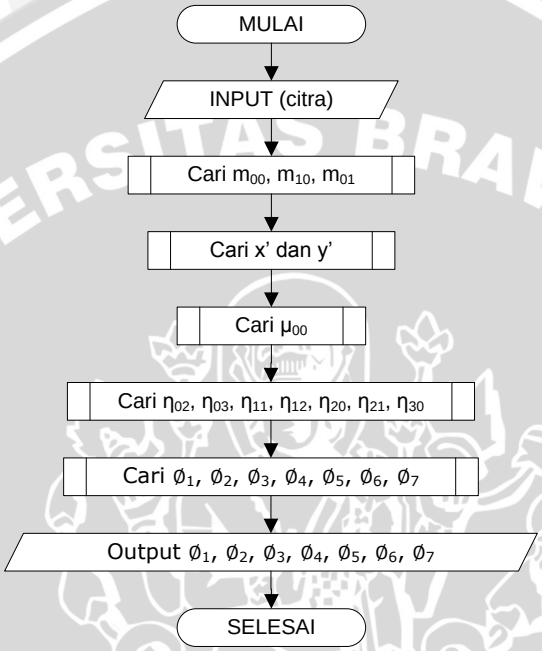


Gambar 3.14 *Flowchart* segmentasi *Region Growing* sistem

3.3.5 Ekstraksi Ciri (Momen Invarian)

Proses ekstraksi ciri bertujuan untuk mengambil ciri citra yang di-*input user* pada kanvas. Metode ekstraksi ciri bermacam-macam, dan yang digunakan pada sistem pengenalan alphabet ini adalah Momen Invarian.

Metode Momen Invarian mengekstraksi citra menjadi tujuh bilangan (parameter) yang mewakili rotasi, skala, dan transformasi dari citra tersebut.

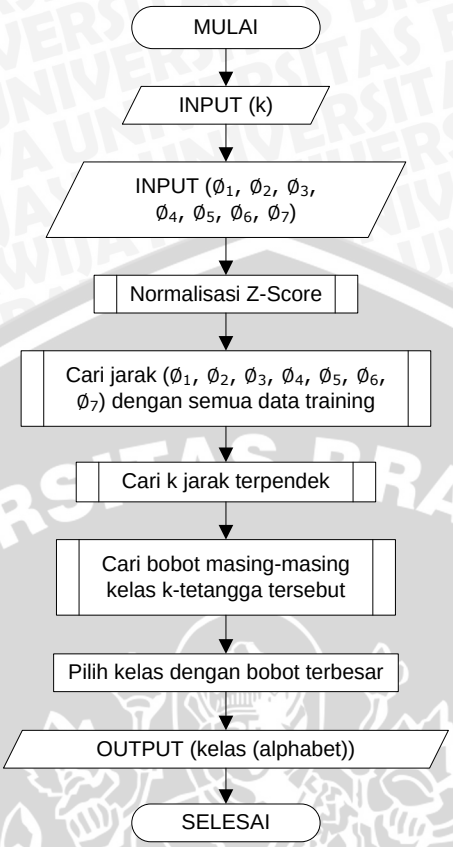


Gambar 3.15 *Flowchart* Momen Invarian sistem pengenalan citra alphabet

3.3.6 Klasifikasi K-Nearest Neighbour

Setelah memperoleh nilai tujuh parameter Momen Invarian, langkah selanjutnya sistem akan mengklasifikasikan nilai tersebut menggunakan metode K-NN (*K-Nearest Neighbour*). Proses K-NN dilakukan dengan mencari jarak antara nilai Momen Invarian segmen citra dengan nilai Momen Invarian data *training*. Normalisasi Z-score dilakukan untuk memperkecil range data. Setelah semua jarak diperoleh, sistem akan mendapatkan k jarak terdekat.

Untuk setiap kelas dari k tetangga tersebut, sistem akan mencari bobotnya. Hasil klasifikasi adalah kelas yang memiliki bobot paling tinggi.



Gambar 3.16 *Flowchart* klasifikasi K-NN sistem pengenalan citra alphabet

3.3.7 Menampilkan Hasil dan Memutar Suara

Hasil pengenalan yang ditampilkan oleh sistem berupa karakter alphabet yang diperoleh dari proses segmentasi, ekstraksi ciri, dan klasifikasi.

Setelah menampilkan karakter alphabetnya, sistem akan mengeluarkan suara ejaan dari karakter-karakter tersebut.

3.4. Contoh Implementasi Kasus

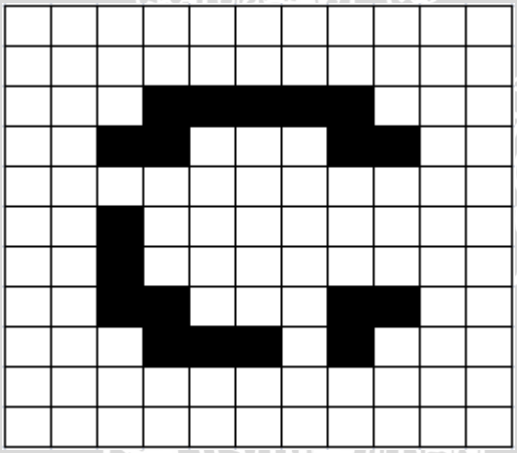
Pada subbab ini akan dibahas contoh sederhana penerapan kasus pada sistem pengenalan citra alphabet sesuai dengan rancangan pada subbab sebelumnya. Berikut adalah contoh kondisi sistem saat akan dijalankan:

1. Data *training* yang tersimpan di *database*:

Tabel 3.2 Contoh data *training* pada *database*

Huruf	p1	p2	p3	p4	p5	p6	p7
A_1	0,25899	0,08903	0,00521	0,00945	$7{,}44\times10^{-9}$	0,00883	$1{,}12\times10^{-10}$
A_2	0,25933	0,08954	0,00582	0,00989	$7{,}89\times10^{-9}$	0,00926	$1{,}11\times10^{-11}$
B_1	0,69965	0,01782	0,01099	0,70052	$2{,}56\times10^{-7}$	0,00099	$2{,}53\times10^{-8}$
B_2	0,69987	0,01734	0,01018	0,70064	$2{,}78\times10^{-7}$	0,00134	$1{,}33\times10^{-8}$
C_1	0,50329	0,25367	0,00223	0,00115	$1{,}22\times10^{-5}$	-0,00025	$1{,}13\times10^{-5}$
C_2	0,50354	0,25378	0,00273	0,00174	$1{,}99\times10^{-5}$	-0,00032	$1{,}55\times10^{-5}$

2. Citra (coretan) *user* pada kanvas:



Gambar 3.17 Contoh citra *input* dari *user*

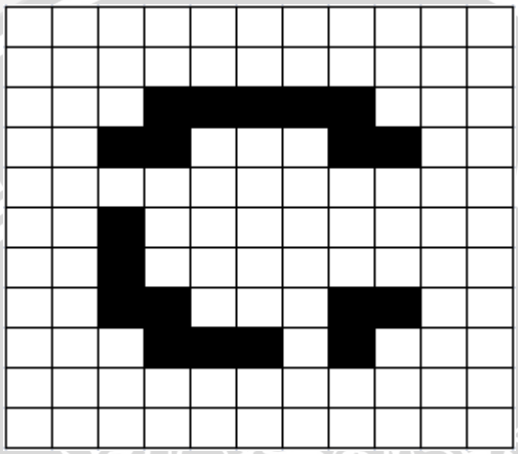
Dengan ketentuan, hitam adalah objek, putih adalah *background*.

3. Nilai k untuk proses K-NN = 3.

Mengacu pada rancangan kerja Gambar 3.8, dengan kondisi sistem tersebut, maka sistem akan melakukan:

1. Binerisasi

Proses binerisasi, seperti rancangan pada Gambar 3.10, mengubah citra *input user* menjadi dua warna saja, yaitu hitam dan putih. Jika piksel yang bersangkutan bukan putih, akan dihitamkan. Berdasar pada kondisi tersebut, maka hasil citra *input user* setelah melakukan proses binerisasi adalah:

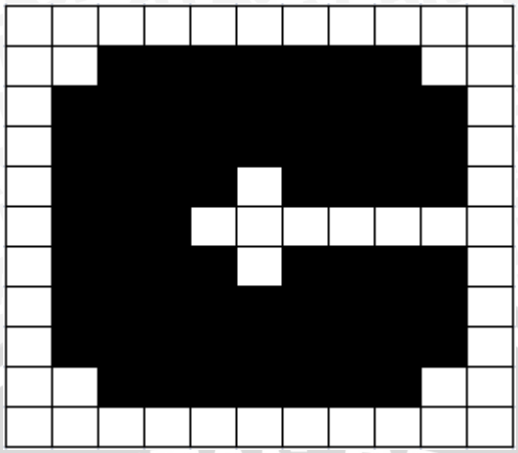


Gambar 3.18 Contoh citra hasil binerisasi dari *input user*

2. *Closing*

Operasi *closing* terdiri dari dua operasi utama yaitu dilasi dan erosi. Setelah citra *input* dari *user* menjadi citra biner (hitam-putih), proses selanjutnya yang dilakukan sistem adalah dilasi dengan 8 ketetanggaan. Pada proses dilasi, jika suatu piksel memiliki sekurang-kurangnya satu tetangga yang berwarna hitam, maka piksel tersebut diberi warna hitam.

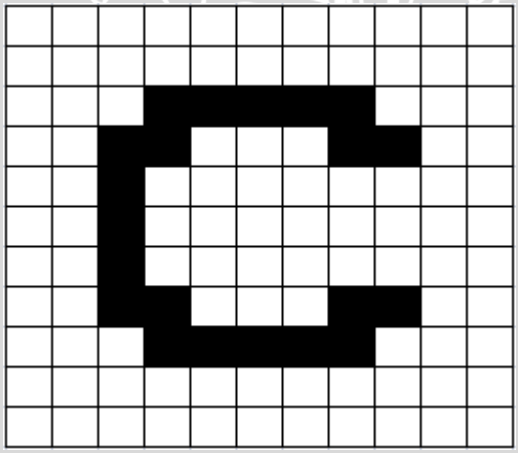
Berdasarkan syarat kondisi tersebut, hasil citra biner setelah dilakukan proses dilasi adalah:



Gambar 3.19 Contoh citra hasil dilasi

Seperti proses dilasi, proses erosi yang dilakukan oleh sistem berdasarkan 8 ketetanggaan. Pada proses erosi, jika suatu piksel memiliki delapan tetangga yang semuanya berwarna hitam, maka piksel tersebut diwarnai hitam. Jika tidak semua tetangganya hitam, piksel tersebut diwarnai putih.

Berdasarkan syarat kondisi tersebut, hasil dari Gambar 3.19 setelah dilakukan proses erosi adalah:



Gambar 3.20 Contoh citra hasil erosi

3. *Thinning* (Skeletonisasi)

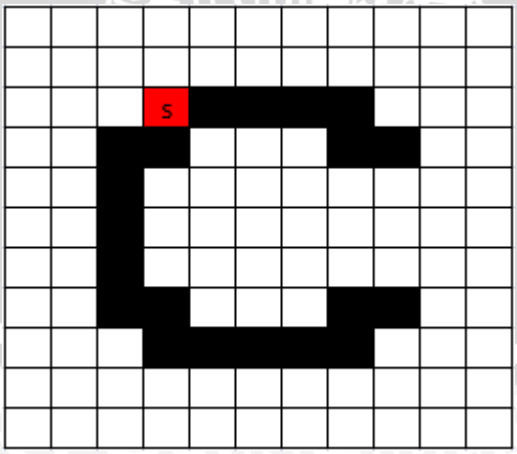
Hasil citra biner dari proses erosi selanjutnya akan ditipiskan (*thinning*) menggunakan algoritma Zhang-Suen. Proses *thinning* pada sistem ini bertujuan untuk memperoleh morfologi dari coretan yang diinputkan *user*.

Proses *thinning* dilakukan berulang-ulang hingga objek menipis hanya setebal 1 piksel. Karena citra hasil erosi pada Gambar 3.20 sudah setebal 1 piksel, maka perulangan proses *thinning* akan dilakukan hanya sekali, dan hasil dari proses *thinning* tersebut tetap seperti Gambar 3.20.

4. *Region Growing*

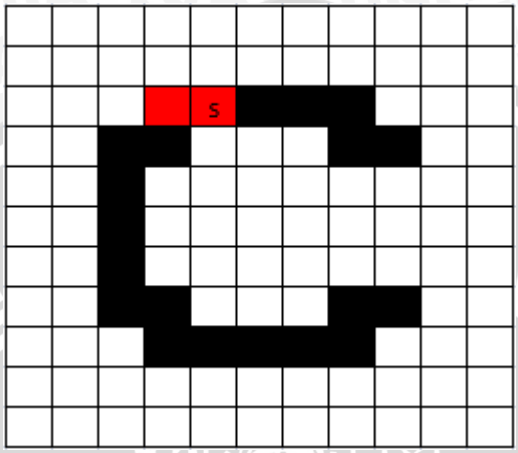
Hasil proses *thinning* selanjutnya dijadikan *input* oleh proses segmentasi. Sistem akan menandai piksel hitam (objek) yang pertama kali ditemukan sebagai *seed*. Label yang digunakan untuk masing-masing segmen adalah warna yang diambil acak dan berbeda-beda. Sistem akan melakukan pengecekan terhadap empat tetangganya secara berurutan (kanan → bawah → kiri → atas). Jika tetangga tersebut berwarna hitam, akan ditandai sebagai *seed* baru, dan diwarnai sama dengan warna random milik *seed* lama.

Sebagai contoh, sistem merandom dan mendapatkan warna merah. Piksel pertama akan ditandai sebagai *seed* dan diwarnai merah.



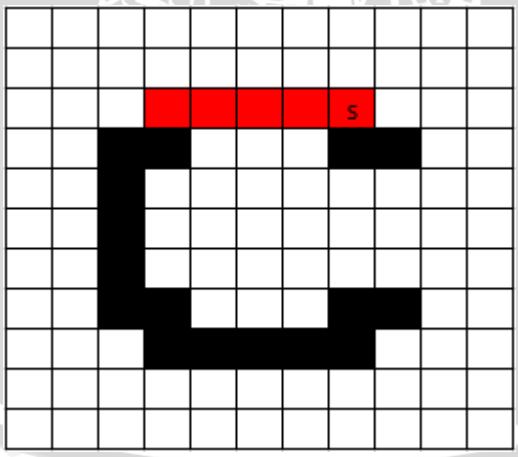
Gambar 3.21 Piksel pertama sebagai *seed* dan merah.

Selanjutnya sistem mengecek piksel di kanan *seed*. Jika piksel tersebut belum ditandai (tidak berwarna merah) dan bukan *background* (tidak berwarna putih), piksel tersebut di tandai (warnai merah) dan dijadikan *seed* baru.



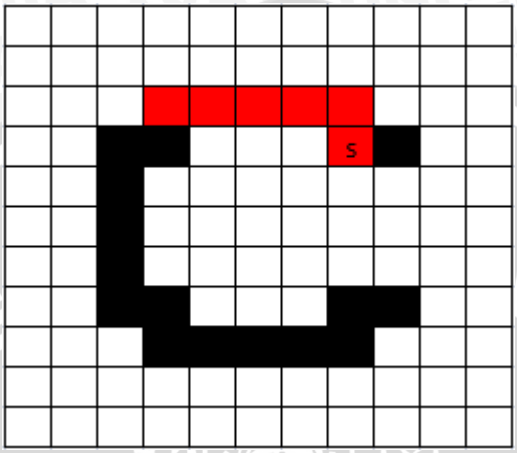
Gambar 3.22 Piksel kanan di warnai merah dan menjadi *seed* baru.

Setelah itu sistem kembali mengecek tetangga kanan dari *seed* baru, lalu dijadikan *seed* baru lagi, begitu seterusnya hingga piksel di sebelah kanan *seed* sekarang berwarna merah (sudah ditandai) atau putih (*background*).



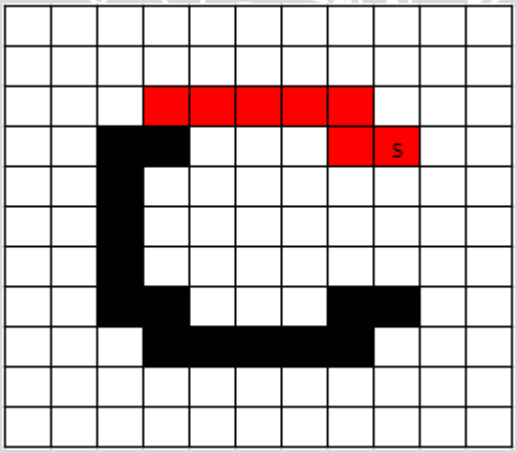
Gambar 3.23 Piksel kanan *seed* yang sekarang bukan putih atau merah.

Jika kondisi citra telah sampai pada keadaan seperti Gambar 3.23, selanjutnya sistem mengecek piksel di bawah *seed* yang sekarang. Karena piksel di bawah memenuhi syarat, piksel tersebut di warnai merah dan ditandai sebagai *seed* baru.



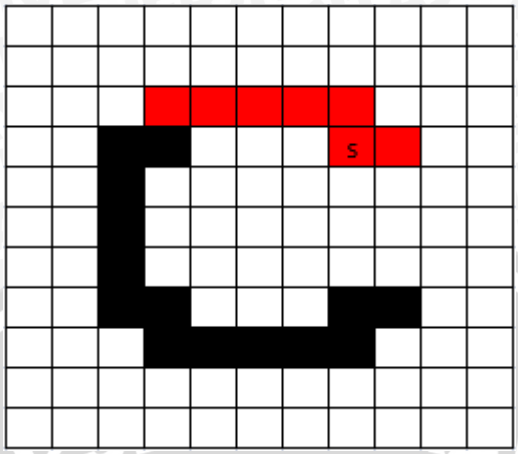
Gambar 3.24 Piksel bawah ditandai sebagai *seed* baru dan diwarnai merah.

Setelah itu kembali dilakukan pengecekan ke kanan tetangga *seed*, lalu diwarnai merah sebagai *seed* baru.



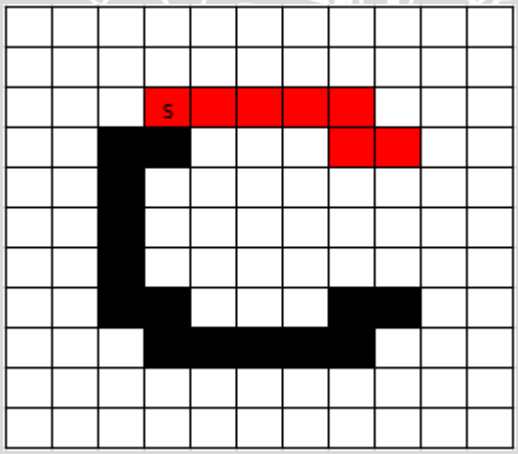
Gambar 3.25 Piksel kanan ditandai sebagai *seed* baru dan diwarnai merah.

Sistem kembali melakukan pengecekan ke tetangga kanan → bawah → kiri → atas. Karena tidak ada satupun tetangga yang memenuhi syarat untuk ditandai, maka *seed* mundur ke *seed* sebelumnya.



Gambar 3.26 *Seed* kembali ke *seed* sebelumnya.

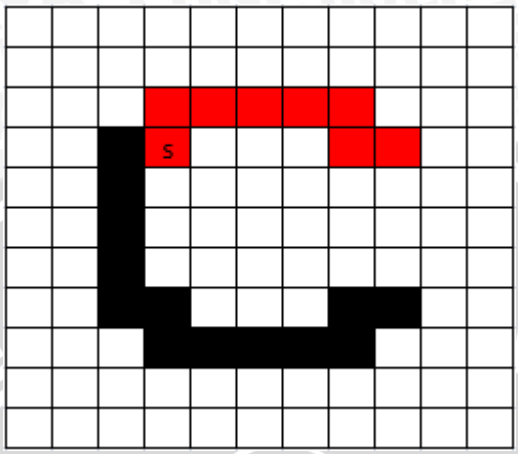
Karena pada *seed* sebelumnya telah dilakukan pengecekan ke kanan, maka pengecekan yang tersisa adalah bawah →kiri → atas. Tidak ada tetangga yang memenuhi syarat, *seed* kembali ke *seed* sebelumnya lagi yaitu *seed* di atas. Begitu seterusnya, selama tidak ada satupun tetangga yang memenuhi syarat untuk ditandai, *seed* akan terus mundur.



Gambar 3.27 *Seed* kembali ke *seed* awal.

Setelah akhirnya ada *seed* yang memiliki tetangga yang memenuhi syarat, maka tetangga tersebut akan ditandai sebagai *seed* baru. Pada Gambar 3.27, *seed*

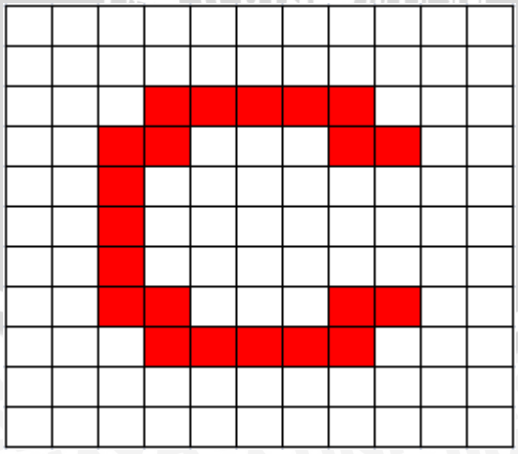
yang sekarang memiliki tetangga bawah yang memenuhi syarat. Tetangga bawah tersebut ditandai sebagai *seed* baru.



Gambar 3.28 Piksel bawah diwarnai merah dan menjadi *seed* baru.

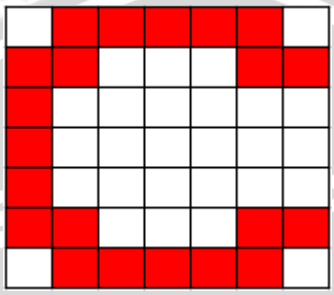
Begitu seterusnya, sistem akan mengecek semua tetangga dari semua piksel objek hingga semuanya berwarna merah. Proses *Region Growing* akan berhenti jika semua piksel objek sudah menjadi *seed*, dan sudah dilakukan pengecekan terhadap tetangga kanan → bawah → kiri → atas.

Hasil akhir proses *Region Growing* pada sistem ditunjukkan pada Gambar 3.29.



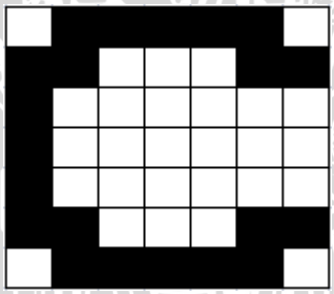
Gambar 3.29 Hasil segmentasi *Region Growing*.

Setelah itu sistem akan memotong (*crop*) citra hasil segmentasi per *region*. Pada hasil segmentasi Gambar 3.29, citra hasil hanya memiliki satu *region* sehingga hasil pemotongan citra hanya satu. Hasil pemotongan tersebut ditunjukkan pada Gambar 3.30.



Gambar 3.30 Hasil *crop*.

Setiap objek pada *region* akan dikembalikan warnanya menjadi hitam seperti semula.



Gambar 3.31 Hasil akhir *crop* per *region*.

5. Momen Invarian

Setelah sistem mendapatkan hasil segmentasi berupa *region-region*, proses selanjutnya adalah mencari tujuh parameter Momen Invarian dari Gambar 3.31. Jalannya ekstraksi ciri metode Momen Invarian ini dapat dilihat pada Gambar 3.14.

Setelah melalui tahap *preprocessing*, hasilnya adalah Gambar 3.31 berukuran $(7 - 1) \times (7 - 1)$ piksel. Artinya nilai $x = 6$ dan $y = 6$. Variabel a_{xy} menunjukkan

warna pada piksel (x, y). Pada sistem ini, a_{xy} bernilai 1 jika piksel (x, y) berwarna hitam (objek) atau bernilai 0 jika putih (*background*).

Berikut adalah langkah-langkah ekstraksi ciri tersebut setelah Gambar 3.31 di-*input*-kan.

- Cari m_{00}, m_{10}, dan m_{01}.

  Pencarian m_{00}, m_{10}, dan m_{01} mengacu pada persamaan (2.2), (2.3), dan (2.4), maka:

$$m_{00} = \sum_{x=6}\sum_{y=6} a_{xy}$$
$$m_{00} = (a_{00} + a_{01} + \cdots + a_{06}) + (a_{10} + a_{11} + \cdots + a_{16}) + \cdots + (a_{60} + a_{61} + \cdots + a_{66}) \quad (3.1)$$
$$m_{00} = (0 + 1 + \cdots + 0) + (1 + 1 + \cdots + 1) + \cdots + (0 + 1 + \cdots + 0)$$
$$m_{00} = 21$$

$$m_{10} = \sum_{x=6}\sum_{y=6} a_{xy}$$
$$m_{10} = (0(a_{00}) + 0(a_{01}) + \cdots + 0(a_{06})) + (1(a_{10}) + 1(a_{11}) + \cdots + 1(a_{16})) + \cdots$$
$$+ (6(a_{60}) + 6(a_{61}) + \cdots + 6(a_{66})) \quad (3.2)$$
$$m_{10} = (0(0) + 0(1) + \cdots + 0(0)) + (1(1) + 1(1) + \cdots + 1(1)) + \cdots$$
$$+ (6(0) + 6(1) + \cdots + 6(0))$$
$$m_{10} = 63$$

$$m_{01} = \sum_{x=6}\sum_{y=6} y(a_{xy})$$
$$m_{01} = (0(a_{00}) + 1(a_{01}) + \cdots + 6(a_{06})) + (0(a_{10}) + 1(a_{11}) + \cdots + 6(a_{16})) + \cdots$$
$$+ (0(a_{60}) + 1(a_{61}) + \cdots + 6(a_{66})) \quad (3.3)$$
$$m_{01} = (0(0) + 1(1) + \cdots + 6(0)) + (0(1) + 1(1) + \cdots + 6(1)) + \cdots$$
$$+ (0(0) + 1(1) + \cdots + 6(0))$$
$$m_{01} = 54$$

- Cari x’ dan y’

  Koordinat (x’, y’) menyatakan titik pusat dari objek. Untuk mencari koordinat tersebut, digunakan persamaan (2.5), sehingga:

$$x' = \frac{m_{10}}{m_{00}} = \frac{63}{21} = 3 \quad (3.4)$$

dan

$$y' = \frac{m_{01}}{m_{00}} = \frac{54}{21} = 2{,}57143 \quad (3.5)$$

Maka koordinat pusat objek adalah (3, 2.57143).

- Cari μ_{00}

  Momen pusat μ adalah momen yang bersesuaian dengan pusat area, didefinisikan pada persamaan (2.6).

$$\mu_{00} = \sum_{x=6}\sum_{y=6}(x-x')^0(y-y')^0 a_{xy}$$
$$c_{00} = \big(1(1)(0) + 1(1)(1) + \cdots + 1(1)(0)\big) + \big(1(1)(1) + 1(1)(1) + \cdots + 1(1)(1)\big) + (1(1)(0) + 1(1)(1) + \cdots + 1(1)(0)) \quad (3.6)$$
$$\mu_{00} = 21$$

- Cari η_{02}, η_{03}, η_{11}, η_{12}, η_{20}, η_{21}, η_{30}

  Variabel η menyatakan momen pusat yang ternormalisasi. Perhitungan nilan η mengacu dari persamaan (2.7), maka:

$$\lambda = \frac{(0+2)}{2} + 1 = 2$$
$$\eta_{02} = \frac{\mu_{02}}{(\mu_{00})^2}$$
$$\eta_{02} = \frac{\sum_{x=6}\sum_{y=6}(x-x')^0(y-y')^2 a_{xy}}{21^2}$$
$$\eta_{02} = \frac{\begin{array}{c}\big(1(0-2{,}57143)^2(0) + 1(1-2{,}57143)^2(1) + \cdots + 1(6-2{,}57143)^2(0)\big) + \\ \big(1(0-2{,}57143)^2(1) + 1(1-2{,}57143)^2(1) + \cdots + 1(6-2{,}57143)^2(1)\big) + \\ \ldots + \big(1(0-2{,}57143)^2(0) + 1(1-2{,}57143)^2(1) + \cdots + 1(6-2{,}57143)^2(0)\big)\end{array}}{441} \quad (3.7)$$
$$\eta_{02} = 0{,}21574$$

$$\lambda = \frac{(0+3)}{2} + 1 = 2{,}5$$

$$\eta_{03} = \frac{\mu_{03}}{(\mu_{00})^{2{,}5}}$$

$$\eta_{03} = \frac{\sum_{x=6}\sum_{y=6}(x-x')^0(y-y')^3 a_{xy}}{21^{2{,}5}}$$

$$\eta_{03} = \frac{\begin{array}{l}\left(1(0-2{,}57143)^3(0) + 1(1-2{,}57143)^3(1) + \cdots + 1(6-2{,}57143)^3(0)\right) + \\ \left(1(0-2{,}57143)^3(1) + 1(1-2{,}57143)^3(1) + \cdots + 1(6-2{,}57143)^3(1)\right) + \\ \ldots + \left(1(0-2{,}57143)^3(0) + 1(1-2{,}57143)^3(1) + \cdots + 1(6-2{,}57143)^3(0)\right)\end{array}}{2020{,}91588} \quad (3.8)$$

$$\eta_{03} = 0{,}02127$$

$$\lambda = \frac{(1+1)}{2} + 1 = 2$$

$$\eta_{11} = \frac{\mu_{11}}{(\mu_{00})^2}$$

$$\eta_{11} = \frac{\sum_{x=6}\sum_{y=6}(x-x')^1(y-y')^1 a_{xy}}{21^2}$$

$$\eta_{11} = \frac{\begin{array}{l}\left((0-3)^1(0-2{,}57143)^1(0) + \cdots + (0-3)^1(6-2{,}57143)^1(0)\right) + \\ \left((1-3)^1(0-2{,}57143)^1(1) + \cdots + (1-3)^1(6-2{,}57143)^1(1)\right) + \\ \ldots + \left((6-3)^1(0-2{,}57143)^1(0) + \cdots + (6-3)^1(6-2{,}57143)^1(0)\right)\end{array}}{441} \quad (3.9)$$

$$\eta_{11} = 0$$

$$\lambda = \frac{(1+2)}{2} + 1 = 2{,}5$$

$$\eta_{12} = \frac{\mu_{12}}{(\mu_{00})^{2{,}5}}$$

$$\eta_{12} = \frac{\sum_{x=6}\sum_{y=6}(x-x')^1(y-y')^2 a_{xy}}{21^{2{,}5}}$$

$$\eta_{12} = \frac{\begin{array}{l}\left((0-3)^1(0-2{,}57143)^2(0) + \cdots + (0-3)^1(6-2{,}57143)^2(0)\right) + \\ \left((1-3)^1(0-2{,}57143)^2(1) + \cdots + (1-3)^1(6-2{,}57143)^2(1)\right) + \\ \ldots + \left((6-3)^1(0-2{,}57143)^2(0) + \cdots + (6-3)^1(6-2{,}57143)^2(0)\right)\end{array}}{2020{,}91588} \quad (3.10)$$

$$\eta_{12} = 1{,}75797 \times 10^{-18}$$

$$\lambda = \frac{(2+0)}{2} + 1 = 2$$

$$\eta_{20} = \frac{\mu_{20}}{(\mu_{00})^2}$$

$$\eta_{20} = \frac{\sum_{x=6}\sum_{y=6}(x-x')^2(y-y')^0 a_{xy}}{21^2}$$

$$\eta_{20} = \frac{\begin{array}{c}\left((0-3)^2(0-2{,}57143)^0(0)+\cdots+(0-3)^2(6-2{,}57143)^0(0)\right)+\\ \left((1-3)^2(0-2{,}57143)^0(1)+\cdots+(1-3)^2(6-2{,}57143)^0(1)\right)+\\ \ldots+\left((6-3)^2(0-2{,}57143)^0(0)+\cdots+(6-3)^2(6-2{,}57143)^0(0)\right)\end{array}}{441} \quad (3.11)$$

$$\eta_{20} = 0{,}28818$$

$$\lambda = \frac{(2+1)}{2} + 1 = 2{,}5$$

$$\eta_{21} = \frac{\mu_{21}}{(\mu_{00})^{2{,}5}}$$

$$\eta_{21} = \frac{\sum_{x=6}\sum_{y=6}(x-x')^2(y-y')^1 a_{xy}}{21^{2{,}5}}$$

$$\eta_{21} = \frac{\begin{array}{c}\left((0-3)^2(0-2{,}57143)^1(0)+\cdots+(0-3)^2(6-2{,}57143)^1(0)\right)+\\ \left((1-3)^2(0-2{,}57143)^1(1)+\cdots+(1-3)^2(6-2{,}57143)^1(1)\right)+\\ \ldots+\left((6-3)^2(0-2{,}57143)^1(0)+\cdots+(6-3)^2(6-2{,}57143)^1(0)\right)\end{array}}{2020{,}91588} \quad (3.12)$$

$$\eta_{21} = 0{,}02333$$

$$\lambda = \frac{(3+0)}{2} + 1 = 2{,}5$$

$$\eta_{30} = \frac{\mu_{30}}{(\mu_{00})^{2{,}5}}$$

$$\eta_{30} = \frac{\sum_{x=6}\sum_{y=6}(x-x')^3(y-y')^0 a_{xy}}{21^{2{,}5}}$$

$$\eta_{30} = \frac{\begin{array}{c}\left((0-3)^3(0-2{,}57143)^0(0)+\cdots+(0-3)^3(6-2{,}57143)^0(0)\right)+\\ \left((1-3)^3(0-2{,}57143)^0(1)+\cdots+(1-3)^3(6-2{,}57143)^0(1)\right)+\\ \ldots+\left((6-3)^3(0-2{,}57143)^0(0)+\cdots+(6-3)^3(6-2{,}57143)^0(0)\right)\end{array}}{2020{,}91588} \quad (3.13)$$

$$\eta_{30} = 0$$

- Cari $\emptyset_1$, $\emptyset_2$, $\emptyset_3$, $\emptyset_4$, $\emptyset_5$, $\emptyset_6$, $\emptyset_7$

  Dari momen ternormalisasi, sekumpulan momen-Momen Invarian (*invariant moments*) dapat didefinisikan. Momen-momen ini dinyatakan dalam $\emptyset$ dan dapat dilihat pada persamaan (2.8) sampai (2.14). Ketujuh momen inilah yang

disebut sebagai Momen Invarian, yang merupakan hasil ekstraksi fitur citra *input user*.

$$\begin{aligned}\emptyset_1 &= \eta_{20} + \eta_{02}\\ \emptyset_1 &= 0{,}28818 + 0{,}21574\\ \emptyset_1 &= 0{,}50392\end{aligned} \tag{3.14}$$

$$\begin{aligned}\emptyset_2 &= (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2\\ \emptyset_2 &= (0{,}28818 + 0{,}21574)^2 + 4(0)^2\\ \emptyset_2 &= 0{,}253935366\end{aligned} \tag{3.15}$$

$$\begin{aligned}\emptyset_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2\\ \emptyset_3 &= (0 - 3(1{,}76^{-18}))^2 + (3(0{,}02333) - 0{,}02127)^2\\ \emptyset_3 &= 0{,}002373638\end{aligned} \tag{3.16}$$

$$\begin{aligned}\emptyset_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2\\ \emptyset_4 &= (0 + 1{,}76^{-18})^2 + (0{,}02333 + 0{,}02127)^2\\ \emptyset_4 &= 0{,}00198916\end{aligned} \tag{3.17}$$

$$\begin{aligned}\emptyset_5 &= (\eta_{30} - 3\eta_{12})\,(\eta_{30} + \eta_{12})\{(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\} + (3\eta_{21} - \eta_{03})(\eta_{21}\\ &\quad + \eta_{03})\{3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\}\\ \emptyset_5 &= \big(0 - 3(1{,}76^{-18})\big)(0 + 1{,}76^{-18})\{(0 + 1{,}76^{-18})^2 - 3(0{,}02333 + 0{,}02127)^2\}\\ &\quad + (3(0{,}02333) - 0{,}02127)\,(0{,}02333\\ &\quad + 0{,}02127)\{3(0 + 1{,}76^{-18})^2 - (0{,}02333 + 0{,}02127)^2\}\\ \emptyset_5 &= 1{,}29304 \times 10^{-5}\end{aligned} \tag{3.18}$$

$$\begin{aligned}\emptyset_6 &= (\eta_{20} - \eta_{02})\left\{(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right\} + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})\\ \emptyset_6 &= (0{,}28818 + 0{,}21574)\{(0 + 1{,}76^{-18})^2 - (0{,}02333 + 0{,}02127)^2\}\\ &\quad + 4(0)(0 + 1{,}76^{-18})(0{,}02333 + 0{,}02127)\\ \emptyset_6 &= -0{,}000144095\end{aligned} \tag{3.19}$$

$$\begin{aligned}
\emptyset_7 &= (3\eta_{30} - \eta_{12})\,(\eta_{30} + \eta_{12})\{(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\} + (3\eta_{21} - \eta_{03})(\eta_{21} \\
&\quad + \eta_{03})\{3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\} \\
\emptyset_7 &= (3(0) - 1{,}76^{-18})(0 + 1{,}76^{-18})\{(0 + 1{,}76^{-18})^2 - 3(0{,}02333 + 0{,}02127)^2\} \\
&\quad + (3(0{,}02333) - 0{,}02127)\,(0{,}02333 \\
&\quad + 0{,}02127)\{3(0 + 1{,}76^{-18})^2 - (0{,}02333 + 0{,}02127)^2\} \\
\emptyset_7 &= 1{,}29304 \times 10^{-5}
\end{aligned} \tag{3.18}$$

6. *K-Nearest Neighbour*

Setelah sistem memperoleh tujuh parameter Momen Invarian melalui proses ekstraksi ciri, selanjutnya sistem tidak memproses citra. Sistem akan melakukan klasifikasi (pengenalan) alphabet menggunakan tujuh parameter Momen Invarian yang telah diperoleh.

Jalannya proses klasifikasi ini mengacu pada *flowchart* Gambar 3.15. Berdasarkan *flowchart* tersebut, proses klasifikasi *K-Nearest Neighbour* akan berjalan sebagai berikut.

- Normalisasi Z-Score. Normalisasi dilakukan untuk memperkecil *range* antarparameter data *training* dan data uji.

Data *training* dan data uji (x) sebelum normalisasi:

Tabel 3.3 Sebelum normalisasi, mencari rata-rata dan standar deviasi.

Huruf	p1	p2	p3	p4	p5	p6	p7
A_1	0,25899	0,08903	0,00521	$9{,}45\times10^{-03}$	$7{,}44\times10^{-09}$	0,00883	$1{,}12\times10^{-10}$
A_2	0,25933	0,08954	0,00582	$9{,}89\times10^{-03}$	$7{,}89\times10^{-09}$	0,00926	$1{,}11\times10^{-11}$
B_1	0,69965	0,01782	0,01099	$7{,}01\times10^{-01}$	$2{,}56\times10^{-07}$	0,00099	$2{,}53\times10^{-08}$
B_2	0,69987	0,01734	0,01018	$7{,}01\times10^{-01}$	$2{,}78\times10^{-07}$	0,00134	$1{,}33\times10^{-08}$
C_1	0,50329	0,25367	0,00223	$1{,}15\times10^{-03}$	$1{,}22\times10^{-05}$	-0,00025	$1{,}13\times10^{-05}$
C_2	0,50354	0,25378	0,00273	$1{,}74\times10^{-03}$	$1{,}99\times10^{-05}$	-0,00032	$1{,}55\times10^{-05}$
data uji (x)	0,50392	0,25394	0,00237	0,00199	$1{,}29\times10^{-05}$	-0,00014	$1{,}29\times10^{-05}$
rata-rata	0,48979	0,13930	0,00563	0,20362	$6{,}51\times10^{-06}$	0,00282	$5{,}68\times10^{-06}$
standar deviasi	0,18034	0,11103	0,00366	0,33950	$8{,}32\times10^{-06}$	0,00431	$7{,}18\times10^{-06}$

Data *training* dan data uji (x) setelah normalisasi:

Tabel 3.4 Data *training* dan data uji setelah dinormalisasi

Huruf	p1	p2	p3	p4	p5	p6	p7
A_1	-1,27988	-0,45279	-0,11954	-0,57194	-0,78166	1,397167	-0,7914
A_2	-1,278	-0,4482	0,047268	-0,57064	-0,78161	1,49706	-0,79142
B_1	1,16367	-1,09416	1,461011	1,463591	-0,75179	-0,42413	-0,7879
B_2	1,16489	-1,09849	1,239515	1,463944	-0,74915	-0,34282	-0,78957
C_1	0,074813	1,030075	-0,93442	-0,59639	0,683672	-0,71219	0,782697
C_2	0,076199	1,031066	-0,7977	-0,59465	1,609078	-0,72845	1,367767
data uji (X)	0,078306	1,032507	-0,89614	-0,59391	0,771457	-0,68664	1,009821

- Cari jarak ($\emptyset_1$, $\emptyset_2$, $\emptyset_3$, $\emptyset_4$, $\emptyset_5$, $\emptyset_6$, $\emptyset_7$) dengan semua data *training*

  Secara umum ada dua cara untuk menghitung jarak antardata, yaitu Euclidean *distance* dan Manhattan *distance*. Sistem pengenalan alphabet ini menggunakan perhitungan jarak Euclidean, yang ditunjukkan oleh persamaan (2.15).

Tabel 3.5 Perhitungan jarak Euclidean

huruf	A_1	A_2	B_1	B_2	C_1	C_2
$(p1 - p1_x)^2$	1,844674	1,839556	1,178014	1,180663	1,22E-05	4,44E-06
$(p2 - p2_x)$	2,20612	2,192496	4,522731	4,541138	$5{,}91\times10^{-6}$	$2{,}08\times10^{-6}$
$(p3 - 3)^2$	0,60311	0,890016	5,556152	4,561016	0,001466	0,009691
$(p4 - p4_x)^2$	0,000483	0,000541	4,233323	4,234778	$6{,}12\times10^{-6}$	$5{,}42\times10^{-7}$
$(p5 - p5_x)^2$	2,412181	2,412013	2,320282	2,312234	0,007706	0,701608
$(p6 - p6_x)^2$	4,342237	4,768528	0,068911	0,118209	0,000653	0,001749
$(p7 - p7_x)^2$	3,244412	3,244462	3,231784	3,237797	0,051585	0,128126
$\sqrt{\sum(p - p_x}$	3,827951846	3,91760283	4,5946921	4,492864926	0,247859344	0,917159022

- Setelah mendapatkan jarak data uji dengan semua data training, sistem akan memilih k-jarak yang terdekat. Nilai k pada contoh implementasi telah ditetapkan 3, maka sistem akan mengambil 3 jarak terdekat, yaitu jarak C_1, C_2, dan A_1.

- Perhitungan bobot. Perhitungan bobot (*vote*) untuk tiap kelas dijelaskan pada persamaan (2.18), sehingga:

$$vote(C) = \frac{1}{(jarak(C_1))^2} + \frac{1}{(jarak(C_2))^2}$$
$$vote(C) = \frac{1}{0{,}24786^2} + \frac{1}{0{,}91716^2} \quad (3.18)$$
$$vote(C) = 17{,}46637$$

$$vote(A) = \frac{1}{(jarak(A_1))^2}$$
$$vote(A) = \frac{1}{3{,}82795^2} \quad (3.18)$$
$$vote(A) = 0{,}06824$$

- Sekarang sistem telah memperoleh bobot dari kelas k-jarak terdekat. Dengan membandingkan hasil *vote* tersebut, sistem memilih huruf C sebagai hasil pengenalan data uji karena *vote*(C) > *vote*(A).

BAB IV
IMPLEMENTASI

Dalam bab ini akan dibahas implementasi sistem yang sudah dijelaskan dalam bab III. Urutan proses yang terjadi dalam sistem, secara khusus dapat dilihat pada DFD dalam gambar 3.5. Implementasi dari masing-masing langkahnya dijelaskan sebagai berikut:

4.1. *Preprocessing*

Proses *preprocessing* adalah proses yang dilakukan pada citra digital untuk mendapatkan bentuk gambar yang diinginkan. Langkah langkah dalam *preprocessing* adalah: binerisasi, *closing*, dan skeletonisasi.

4.1.1. Binerisasi

Proses binerisasi digunakan untuk mendapatkan citra dengan dua buah warna saja yaitu hitam (nilai 0) dan putih (nilai 255). Implementasinya dapat dilihat pada kode program 4.1.

Kode 4.1. Binerisasi

```
1   //input: bmp1 -> citra masukan
2   //output: bmp1 -> citra yang sudah di binerkan
3   // T: nilai threshold
4
5   unsafe private Bitmap Binerisasi(Bitmap bmp1)
6   {
7           BitmapData bmpdata1 = bmp1.LockBits(newRectangle(0, 0, bmp1.Width,
8                                   bmp1.Height), ImageLockMode.ReadWrite,
9                                   PixelFormat.Format24bppRgb);
10          byte* pixel1 = (byte*)bmpdata1.Scan0;
11          int T = 200;
12
13          for (int i = 0; i < bmp1.Height; i++)
14        {
15                  for (int j = 0; j < bmp1.Width; j++)
16                  {
17                          if (pixel1[0] > T && pixel1[1] > T &&
18                                                          pixel1[2] > T)
19                          {
20                                  pixel1[0] = pixel1[1] = pixel1[2] = 255;
21                          }
22                          else
23                                  pixel1[0] = pixel1[1] = pixel1[2] = 0;
24                          pixel1 += 3;
25                  }
26                  pixel1 += bmpdata1.Stride - bmpdata1.Width * 3;
27          }
28          bmp1.UnlockBits(bmpdata1);
29          return bmp1;
30  }
```

Nilai T (*threshold*) pada baris 12, yaitu T = 200 digunakan sebagai batasan perubahan nilai. Untuk tiap *pixel* dalam citra (bmp1), jika seluruh nilai R, G, dan B dalam sebuah *pixel* memiliki nilai di atas nilai T (baris 18), maka nilai R, G, dan B pada *pixel* tersebut diubah menjadi 255 (putih) seperti terlihat pada baris 21. Selain itu, semua nilai R, G, dan B diubah menjadi 0 (hitam), terlihat pada baris 23.

4.1.2. *Closing*

Proses *closing* dimaksudkan untuk menutupi/menyambung bagian-bagian huruf yang terputus. *Closing* terdiri dari 2 tahap, yaitu dilasi, kemudian erosi. Dilasi digunakan untuk memperluas area objek dalam citra. Sedangkan erosi digunakan untuk mengurangi area objek dalam citra.

Implementasi dilasi dapat dilihat pada kode program 4.2.

Kode 4.2. Dilasi

```
1   //input:bmp1 -> citra biner
2   //      count -> banyaknya proses dilasi dilakukan
3   //output: bmp1 -> citra yang sudah di dilasi
4   //bmp2: citra sementara untuk membantu proses dilasi
5
6   unsafe private Bitmap Dilasi(int count, Bitmap bmp1)
7   {
8           BitmapData bmpdata1, bmpdata2;
9           Bitmap bmp2;
10          byte r1, r2, r3, r4, r5, r6, r7, r8;
11          byte* pixel1, pixel2, temp;
12
13          for (int k = 0; k < count; k++)
14          {
15                  ...
16                  ...
17
18                  for (int i = 0; i < bmpdata1.Height; i++)
19                  {
20                          for (int j = 0; j < bmpdata1.Width; j++)
21                          {
22                                  if (i == 0 || i == bmpdata1.Height - 1 || j == 0
23                                          || j == bmpdata1.Width - 1)
24                  pixel2[0] = pixel2[1] = pixel2[2] = 255;
25                                  else
26          {
27                  r1 = r2 = r3 = r4 = r5 = r6 = r7 = r8 = 1;
28                                          #region cek tetangga
29                  try
30                  {
31          //kiri-atas
32                                          temp = pixel1 - bmpdata1.Stride - 3;
33                                          r1 = temp[0];
34                                          //atas
35                                          temp += 3;
36                                          r2 = temp[0];
37                                          //kanan-atas
38                                          temp += 3;
39                                          r3 = temp[0];
40                                          //kiri
41                                          temp = pixel1 - 3;
```

```
42                                      r4 = temp[0];
43                                      //kanan
44                                      temp = pixel1 + 3;
45                                      r5 = temp[0];
46                                      //kiri-bawah
47                                      temp = pixel1 + bmpdata1.Stride - 3;
48                                      r6 = temp[0];
49                                      //bawah
50                                      temp += 3;
51                                      r7 = temp[0];
52                                      //kanan-bawah
53                                      temp += 3;
54                                      r8 = temp[0];
55                                  }
56                                  catch{ }
57
58                                  if (r1 == 0 || r2 == 0 || r3 == 0 || r4 == 0 ||
59                                  r5 == 0 ||r6 == 0 || r7 == 0 || r8 == 0)
60                                      pixel2[0] = pixel2[1] = pixel2[2] = 0;
61                                  else pixel2[0] = pixel2[1] = pixel2[2] = 255;
62                                  #endregion
63                                  }
64                                  pixel1 += 3;
65                                  pixel2 += 3;
66                              }
67                              pixel1 += bmpdata1.Stride - bmpdata1.Width * 3;
68                              pixel2 += bmpdata2.Stride - bmpdata2.Width * 3;
69                          }
70                          bmp2.UnlockBits(bmpdata2);
71                          bmp1.UnlockBits(bmpdata1);
72                          bmp1 = newBitmap(bmp2);
73                  }
74                  return bmp1;
75          }
```

Pada *preprocessing*, dilasi dilakukan sebanyak 2 kali (*count* = 2). Proses dilasi menggunakan *structuring element* diimplementasikan pada kode 4.2 baris 27 sampai 61. *Structuring element* yang digunakan berukuran 3 x 3 seperti berikut:

1	1	1
1	1	1
1	1	1

Setelah dilasi, dilakukan proses erosi juga sebanyak 2 kali. Implementasi fungsi untuk erosi, dapat dilihat pada kode 4.3.

Kode 4.3. Erosi

```
1   //input: bmp1 -> citra hasil dilasi
2   //      count -> banyaknya proses erosi dilakukan
3   //output: bmp1 -> citra yang sudah di Erosi
4   //bmp2: citra sementara untuk membantu proses erosi
5
6   unsafe private Bitmap Erosi(int count, Bitmap bmp1)
7   {
8           BitmapData bmpdata1, bmpdata2;
```

```
9           Bitmap bmp2;
10          byte r1, r2, r3, r4, r5, r6, r7, r8;
11          byte* pixel1, pixel2, temp;
12
13          for (int k = 0; k < count; k++)
14          {
15                  ...
16                  for (int i = 0; i < bmpdata1.Height; i++)
17                  {
18                          for (int j = 0; j < bmpdata1.Width; j++)
19                          {
20                                  if (i==0 || i==bmpdata1.Height - 1 || j == 0 ||
21                                          j == bmpdata1.Width - 1)
22                                          pixel2[0] = pixel2[1] = pixel2[2] = 255;
23                                  else
24                                  {
25                                          r1 = r2 = r3 = r4 = r5 = r6 = r7 = r8 = 1;
26                                          #region cek tetangga
27                                          try
28                                          {
29                                          //kiri-atas
30                                          temp = pixel1 - bmpdata1.Stride - 3;
31                                          r1 = temp[0];
32                                          //atas
33                                          temp += 3;
34                                          r2 = temp[0];
35                                          //kanan-atas
36                                          temp += 3;
37                                          r3 = temp[0];
38                                          //kiri
39                                          temp = pixel1 - 3;
40                                          r4 = temp[0];
41                                          //kanan
42                                          temp = pixel1 + 3;
43                                          r5 = temp[0];
44                                          //kiri-bawah
45                                          temp = pixel1 + bmpdata1.Stride - 3;
46                                          r6 = temp[0];
47                                          //bawah
48                                          temp += 3;
49                                          r7 = temp[0];
50                                          //kanan-bawah
51                                          temp += 3;
52                                          r8 = temp[0];
53                                          }
54                                          catch
55                                          { }
56                                          if (r2==0 && r4 == 0 && r5 == 0 && r7 == 0)
57                                                  pixel2[0]=pixel2[1]=pixel2[2] = 0;
58                                          else pixel2[0]=pixel2[1] = pixel2[2] = 255;
59                                          #endregion
60                                  }
61                                  pixel1 += 3;
62                                  pixel2 += 3;
63                          }
64                          pixel1 += bmpdata1.Stride - bmpdata1.Width * 3;
65                          pixel2 += bmpdata2.Stride - bmpdata2.Width * 3;
66                  }
67                  bmp2.UnlockBits(bmpdata2);
68                  bmp1.UnlockBits(bmpdata1);
69                  bmp1 = newBitmap(bmp2);
70          }
71          return bmp1;
72  }
```

Dengan menggunakan *structuring element* yang sama dengan dilasi, prosesnya dapat dilihat pada baris 22 sampai dengan 56. Setelah didilasi 2 kali,

objek akan bertambah besar. Untuk mengembalikan ukuran objek seperti semula, erosi yang dilakukan juga sebanyak 2 kali.

4.2. Skeletonisasi

Proses skeletonisasi adalah proses untuk mengambil ciri geometri dari sebuah objek. Yang diambil adalah bentuk struktur tulang dari objek.

Kode 4.4. Skeletonisasi

```
unsafe private Bitmap Skeleton(Bitmap bmp)
{
        Bitmap bmpx = newBitmap(bmp);
        BitmapData bmpdata1, bmpdata2;
        byte* pixel1, pixel2;
        bool s1, s2, s3, s4;
        int count;
        do
        {
                #region iterasi
                count = 0;
                bmp = newBitmap(bmpx);
                bmpdata1 = bmp.LockBits(...
                bmpdata2 = bmpx.LockBits(...
                pixel1 = (byte*)bmpdata1.Scan0;
                pixel2 = (byte*)bmpdata2.Scan0;
                for (int a = 0; a < bmpdata2.Height; a++)
                {
                        for (int b = 0; b < bmpdata2.Width; b++)
                        {
                        try
                        {
                                if (pixel1[0] == 0)
                                {
                                        s1 = CekJumlah(bmpdata1, pixel1);
                                        s2 = CekPerubahan(bmpdata1, pixel1);
                                        s3 = CekKali(pixel1 - bmpdata1.Stride,
                                        pixel1 + 3, pixel1 + bmpdata1.Stride);
                                        s4 = CekKali(pixel1 + 3, pixel1 +
                                        bmpdata1.Stride, pixel1 - 3);

                                        if (s1==true&&s2==true&&s3==true&&s4==true)
                                        {
                                        pixel2[0] = pixel2[1] = pixel2[2] = 255;
                                        count++;
                                        }
                                }
                        }
                        catch { }
                        pixel1 += 3;
                        pixel2 += 3;
                }
                pixel1 += bmpdata1.Stride - bmpdata1.Width * 3;
                pixel2 += bmpdata2.Stride - bmpdata2.Width * 3;
        }
        bmpx.UnlockBits(bmpdata2);
        bmp.UnlockBits(bmpdata1);

        bmp = newBitmap(bmpx);
        bmpdata1 = bmp.LockBits(...
        bmpdata2 = bmpx.LockBits(...
        pixel1 = (byte*)bmpdata1.Scan0;
        pixel2 = (byte*)bmpdata2.Scan0;
        for (int a = 0; a < bmpdata2.Height; a++)
        {
```

```
56                  for (int b = 0; b < bmpdata2.Width; b++)
57                  {
58                      try
59                      {
60                          if (pixel1[0] == 0)
61                          {
62                              s1 = CekJumlah(bmpdata1, pixel1);
63                              s2 = CekPerubahan(bmpdata1, pixel1);
64                              s3 = CekKali(pixel1 - bmpdata1.Stride,
65                                  pixel1 + 3, pixel1 - 3);
66                              s4 = CekKali(pixel1 - bmpdata1.Stride,
67                                  pixel1+bmpdata1.Stride,pixel1 - 3);
68
69                              if (s1 == true&& s2 == true&& s3 == true
70                                  && s4 == true)
71                              {
72                              pixel2[0] = pixel2[1] = pixel2[2] = 255;
73                              count++;
74                              }
75                          }
76                      }
77                      catch { }
78                      pixel1 += 3;
79                      pixel2 += 3;
80                  }
81                  pixel1 += bmpdata1.Stride - bmpdata1.Width * 3;
82                  pixel2 += bmpdata2.Stride - bmpdata2.Width * 3;
83              }
84              bmpx.UnlockBits(bmpdata2);
85              bmp.UnlockBits(bmpdata1);
86              #endregion
87              } while (count != 0);
88              pbClosing.Image = (Image)bmpx;
89              return bmpx;
90      }
```

Proses skeletonisasi meliputi 2 langkah utama. Langkah pertama yaitu menghapus *pixel* pada tepian objek jika memenuhi 4 syarat:

a. Syarat pertama (S1), yaitu memeriksa jumlah nilai 0 (hitam) di tetangga. Bernilai benar jika jumlahnya antara 2 sampai 6. Ditunjukkan oleh fungsi

   `s1 = CekJumlah(bmpdata1, pixel1);` (baris ke-25)

b. Syarat kedua (S2), yaitu memeriksa *pixel* tetangga searah jarum jam, dimulai dari *pixel* di atasnya. Dan menghitung jumlah peralihan nilai *pixel* tetangganya dari nilai 255 (putih) ke nilai 0 (hitam). Bernilai benar ketika jumlahnya = 1.

   `s2 = CekPerubahan(bmpdata1, pixel1);` (baris ke-26)

c. Syarat ketiga (S3), yaitu memeriksa *pixel-pixel* tetangganya, dan bernilai benar jika p2, p4, dan p6 tidak bernilai 0 semuanya atau tidak benilai 255 semuanya.

`s3 = CekKali(pixel1-bmpdata1.Stride, pixel1+3, pixel1+bmpdata1.Stride);` (baris ke-27)

d. Syarat keempat (S4), yaitu memeriksa *pixel-pixel* tetangga yaitu p4, p6, dan p8 seperti pada syarat S3.

`s4 = CekKali(pixel1+3, pixel1+bmpdata1.Stride, pixel1-3);` (baris ke-29)

P9	P2	P3
P8	P1	P4
P7	P6	P5

Langkah pertama dalam algoritma ini diimplementasikan pada kode program 4.4 pada baris 17 sampai 47.

Langkah kedua yaitu menghapus *pixel* tepian objek yang memenuhi seperti syarat S1 dan S2 pada langkah pertama, ditambah dengan syarat ketiga (S3) dengan memeriksa *pixel* p2, p6, dan p8, dan syarat keempat (S4) dengan memeriksa *pixel* p2, p4, dan p8. Dan implementasinya ditujukkan pada kode program 4.4 pada baris 54 sampai dengan 85.

Kedua langkah tersebut diulang-ulang sehingga tidak ada lagi *pixel* yang dihapus. Dengan demikian keluaran dari kode program 4.4 adalah citra yang sudah terskeletonisasi.

4.3. Segmentasi (*Region Growing*)

Proses segmentasi ini digunakan untuk menandai/melabeli setiap objek yang berada dalam satu region. Hal ini bermanfaat ketika seorang pengguna sistem memasukkan lebih dari satu huruf ke dalam aplikasi. Dengan adanya pelabelan ini, setiap huruf akan terlabeli sebagai objek yang berbeda-beda, baru kemudian masing-masing objek tersebut dikenali.

Implementasi *Region Growing* pada kode program dapat dilihat pada kode program 4.5.

Kode 4.5. *Region Growing*

```
1   unsafe private Bitmap RegionGrowing(Bitmap bmp1)
2   {
3           BitmapData bmpdata1 = bmp1.LockBits(...);
4           byte* pixel = (byte*)bmpdata1.Scan0;
5           byte* temp;
6           for (int j = 0; j < bmpdata1.Width; j++)
7           {
8                   temp = pixel;
9                   for (int i = 0; i < bmpdata1.Height; i++)
10                  {
```

```
11                          int var;
12                          topOfStack = (int)&var;
13                          byte warna1 = Randomizer();
14                          byte warna2 = Randomizer();
15                          byte warna3 = Randomizer();
16
17                          if ((pixel[0] == 0) && (pixel[1] == 0) && (pixel[2] == 0))
18                                  Labeling(bmpdata1, j, i, pixel, pixel[2],
19                                  pixel[1], pixel[0], warna1, warna2, warna3);
20
21                          pixel = pixel + bmpdata1.Stride;
22                  }
23                  pixel = temp + 3;
24          }
25          bmp1.UnlockBits(bmpdata1);
26          bmp1 = GabungRegion(bmp1);
27          return bmp1;
28  }
```

Citra yang dijadikan masukan dalam fungsi ini adalah citra hasil skeletonisasi. Perulangan pada baris 6 dan 9 menunjukkan perulangan untuk menelusuri tiap *pixel* dalam citra. Baris 13, 14, dan 15 digunakan untuk menghasilkan warna acak untuk menandai sebuah region. Inti dari kode program ini adalah baris 17 sampai 19. Jika nilai *pixel* yang dikunjungi bernilai 0 (hitam) makan tandai dengan memanggil prosedur *labeling*. Prosedur *labeling* sendiri dapat dilihat pada kode program 4.6.

Kode 4.6. Labeling

```
1   unsafe private void Labeling(BitmapData bmpdata1, int j, int i, byte* pixel,
2   byte WR, byte WG, byte WB, byte warna1, byte warna2, byte warna3)
3   {
4           byte* temp;
5           int remaining;
6           remaining  = stackSize - (topOfStack - (int)&remaining);
7           if (remaining < spaceRequired){ }
8           else
9           {
10                  if (((i < bmpdata1.Height - 1) && (i > 0)) && ((j <
11                                                  bmpdata1.Width - 1) && (j > 0)))
12                  {
13                          //0 biru, 1 hijau, 2 merah
14                          pixel[0] = warna1;
15                          pixel[1] = warna2;
16                          pixel[2] = warna3;
17
18                          byte xyR, x1R, x_1R, y1R, y_1R;
19                          byte xyG, x1G, x_1G, y1G, y_1G;
20                          byte xyB, x1B, x_1B, y1B, y_1B;
21                          byte s1R, s2R, s3R, s4R;
22                          byte s1G, s2G, s3G, s4G;
23                          byte s1B, s2B, s3B, s4B;
24
25                          //tengah
26                          xyB = pixel[0];xyG = pixel[1];xyR = pixel[2];
27                          //kanan
28                          temp = pixel + 3;
29                          x1B = temp[0];x1G = temp[1];x1R = temp[2];
```

```
30                          //kiri
31                          temp = pixel - 3;
32                          x_1B = temp[0];x_1G = temp[1];x_1R = temp[2];
33                          //bawah
34                          temp = pixel + bmpdata1.Stride;
35                          y1B = temp[0];y1G = temp[1];y1R = temp[2];
36                          //kanan-bawah
37                          temp = temp + 3;
38                          s1B = temp[0];s1G = temp[1];s1R = temp[2];
39                          //kiri-bawah
40                          temp = temp - 6;
41                          s2B = temp[0];s2G = temp[1];s2R = temp[2];
42                          //atas
43                          temp = pixel - bmpdata1.Stride;
44                          y_1B = temp[0];y_1G = temp[1];y_1R = temp[2];
45                          //kiri-atas
46                          temp = temp - 3;
47                          s3B = temp[0];s3G = temp[1];s3R = temp[2];
48                          //kanan-atas
49                          temp = temp + 6;
50                          s4B = temp[0];s4G = temp[1];s4R = temp[2];
51
52                          //kanan
53                          if (x1B == 0 && x1G == 0 && x1R == 0)
54                             Labeling(bmpdata1, j + 1, i, pixel + 3, WR, WG, WB,
55                                 warna1, warna2, warna3);
56                          //kanan-bawah
57                          if (s1B == 0 && s1G == 0 && s1R == 0)
58                             Labeling(bmpdata1, j + 1, i + 1, pixel+bmpdata1.Stride
59                                 + 3, WR, WG, WB, warna1, warna2, warna3);
60                          //bawah
61                          if (y1B == 0 && y1G == 0 && y1R == 0)
62                             Labeling(bmpdata1, j, i + 1, pixel + bmpdata1.Stride,
63                                 WR, WG, WB, warna1, warna2, warna3);
64                          //kiri-bawah
65                          if (s2B == 0 && s2G == 0 && s2R == 0)
66                             Labeling(bmpdata1, j-1, i + 1, pixel + bmpdata1.Stride
67                                 - 3, WR, WG, WB, warna1, warna2, warna3);
68                          //kiri
69                          if (x_1B == 0 && x_1G == 0 && x_1R == 0)
70                             Labeling(bmpdata1, j - 1, i, pixel - 3, WR, WG, WB,
71                                 warna1, warna2, warna3);
72                          //kiri-atas
73                          if (s3B == 0 && s3G == 0 && s3R == 0)
74                             Labeling(bmpdata1, j-1, i - 1, pixel - bmpdata1.Stride
75                                 - 3, WR, WG, WB, warna1, warna2, warna3);
76                          //atas
77                          if (y_1B == 0 && y_1G == 0 && y_1R == 0)
78                             Labeling(bmpdata1, j, i - 1, pixel - bmpdata1.Stride,
79                                 WR, WG, WB, warna1, warna2, warna3);
80                          //kanan-atas
81                          if (s4B == 0 && s4G == 0 && s4R == 0)
82                             Labeling(bmpdata1, j+1, i - 1, pixel - bmpdata1.Stride
83                                 + 3, WR, WG, WB, warna1, warna2, warna3);
84                      }
85                  }
86          }
```

Baris 35 sampai dengan 50 digunakan untuk mengambil nilai *pixel* di 8 tetangga *pixel* yang di tandai. Pada baris selanjutnya (baris 53 sampai dengan 84) prosedur *labeling* ini dipanggil lagi secara rekursif, untuk menandai tetangga yang juga berwarna hitam sebagai satu region. Demikian seterusnya hingga tidak ada

lagi tetangga *pixel* yang berwarna hitam. Artinya semua *pixel* hitam yang berada dalam satu region sudah ditandai.

4.4. Momen Invarian

Momen Invarian adalah fitur yang digunakan dalam penelitian ini untuk mengidentifikasi objek huruf. Momen Invarian akan dihitung pada tiap region yang dihasilkan melalui proses segmentasi. Tiap segmen kemudian diambil, dan di-*resize* ke dalam ukuran 250 x 250 *pixel*. Kemudian berturut-turut dilakukan *preprocessing* lagi (binerisasi, skeletonisasi, dan kemudian didilasi sebanyak 4 kali). *Preprocessing* dilakukan lagi sebab hasil *resize* menyebabkan muncul lagi *pixel* selain hitam dan putih.

Setelah terbentuk beberapa objek dengan ukuran 250 x 250 tersebut. Barulah kemudian dilakukan perhitungan Momen Invarian. Kode programnya dapat dilihat pada kode 4.7.

Kode 4.7. Moment Invarian

```
1   //Langkah 1: mencari m00, m01, m10
2   m00 = MomentM(bmpdata, 0, 0);
3   m01 = MomentM(bmpdata, 0, 1);
4   m10 = MomentM(bmpdata, 1, 0);
5
6   //Langkah 2: mencari xMean dan yMean
7   xMean = m10 / m00;
8   yMean = m01 / m00;
9
10  //Langkah 3: mencari M00
11  M00 = CentralMomentM(bmpdata, 0, 0);
12
13  //Langkah 4: mencari n02, n03, n11, n12, n20, n21, n30
14  n02 = NormalizedCentralMoment(bmpdata, 0, 2);
15  n03 = NormalizedCentralMoment(bmpdata, 0, 3);
16  n11 = NormalizedCentralMoment(bmpdata, 1, 1);
17  n12 = NormalizedCentralMoment(bmpdata, 1, 2);
18  n20 = NormalizedCentralMoment(bmpdata, 2, 0);
19  n21 = NormalizedCentralMoment(bmpdata, 2, 1);
20  n30 = NormalizedCentralMoment(bmpdata, 3, 0);
21
22  //Langkah 5: mencari o[0], o[1], o[2], o[3], o[4], o[5], o[6]
23  O[0] = o[0] = n20 + n02;
24  O[1] = o[1] = Math.Pow(n20 - n02, 2) + 4 * Math.Pow(n11, 2);
25  O[2] = o[2] = Math.Pow(n30 - 3 * n12, 2) + Math.Pow(3 * n21 - n03, 2);
26  O[3] = o[3] = Math.Pow(n30 + n12, 2) + Math.Pow(n21 + n03, 2);
27  O[4] = o[4] = (n30 - 3 * n12) * (n30 + n12) * (Math.Pow(n30 + n12, 2) - 3 *
28                Math.Pow(n21 + n03, 2)) + (3 * n21 - n03) * (n21 + n03) * (3 *
29                Math.Pow(n30 + n12, 2) - Math.Pow(n21 + n03, 2));
30  O[5] = o[5] = (n20 - n02) * (Math.Pow(n30 + n12, 2) - Math.Pow(n21 + n03, 2))
31                + 4 * n11 * (n30 + n12) * (n21 + n03);
32  O[6] = o[6] = (3 * n21 - n03) * (n30 + n12) * (Math.Pow(n30 + n12, 2) - 3 *
33                Math.Pow(n21 + n03, 2)) + (3 * n21 - n03) * (n21 + n03) * (3 *
34                Math.Pow(n30 + n12, 2) - Math.Pow(n21 + n03, 2));
```

Baris 2 sampai 4 adalah implementasi dari perhitungan *zero and first order moment* seperti pada persamaan 2.2, 2.3 dan 2.4. Baris ke-11 merupakan implementasi perhitungan momen pusat dan baris ke 14 sampai 20 adalah momen pusat yang ternormalisasi, yang dihitung sampai order ke-3.

Perhitungan 7 buah nilai momen yang akan digunakan sebagai fitur dalam penelitian ini, diimplementasikan pada baris 23 sampai 34. Tujuh buah nilai momen dinyatakan dengan variabel O[0..6].

4.5. *K-Nearest Neighbour*

Untuk proses klasifikasi hurufnya, digunakan metode K-Nearest Neighbour (K-NN). K-NN akan mencari k-data terdekat dari keseluruhan data *training*. Nilai k diambil dari textbox `tbK.Text` (pada baris ke-11). Keseluruhan proses K-NN bisa dilihat pada kode program 4.8.

Kode 4.8. K-Nearest Neighbour

```
1   private char KNN()
2   {
3           Neighbour[] distance = new Neighbour[dataGridView1.RowCount - 2];
4           distance = CariJarak();
5           distance = CariTerdekat(distance);
6
7           if (distance.Length == 1)
8                   huruf = distance[0].karakter;
9           else
10          {
11                  Neighbour[] vote = new Neighbour[Convert.ToInt32(tbK.Text)];
12                  int idx = 0;
13
14                  (...)
15
16                  for (int i = 0; i < distance.Length; i++)
17                          if (CheckIfExist(i, distance[i].karakter, vote) != true)
18                          {
19                                  vote[idx].karakter = distance[i].karakter;
20                                  idx++;
21                          }
22
23                  for (int i = 0; i < distance.Length; i++)
24                  {
25                          idx = GetIndex(distance[i].karakter, vote);
26                          vote[idx].jarak += (1 / Math.Pow(distance[i].jarak, 2));
27                  }
28
29                  huruf = vote[0].karakter;
30                  double jarak = vote[0].jarak;
31                  dataGridView3.Rows.Clear();
32
33                  for (int i = 0; i < vote.Length; i++)
34                  {
35                          (...)
36
37                          try
38                          {
```

```
39                                  if (jarak < vote[i].jarak)
40                                  {
41                                          jarak = vote[i].jarak;
42                                          huruf = vote[i].karakter;
43                                  }
44                          }
45                          catch { }
46                  }
47          }
48          return huruf;
49  }
```

Fungsi `private char KNN` memberikan output berupa satu karakter hasil klasifikasi menggunakan metode KNN. Fungsi KNN diawali dengan inisialisasi *array* `distance` bertipe `Neighbour` sepanjang jumlah seluruh data *training*. `Neighbour` adalah sebuah *struct* yang mendefinisikan setiap data, terdiri dari `karakter` (tipe `char`) dan `jarak` (tipe `float`).

Selanjutnya, *array* `distance` diisi dengan jarak antara data uji dengan seluruh data training dengan memanggil fungsi `CariJarak()`. Fungsi `CariJarak()` akan menghitung jarak antara data uji dengan setiap data training dengan metode *Euclidean Distance*. Fungsi `CariJarak()` memberikan kembalian (*return*) berupa *array* bertipe `Neighbour`.

Setelah memperoleh jarak antara data uji dengan semua data *training*, proses selanjutnya adalah mencari k-data terdekat menggunakan fungsi `CariTerdekat()`. Fungsi `CariTerdekat()` akan mengurutkan seluruh jarak secara *ascending* (kecil ke besar) dengan algoritma *Buble Sort*. Selanjutnya, akan diambil sebanyak k-data pertama dari *array* dan dikembalikan (*return*) kepada *array* `distance`.

Jika panjang *array* `distance` hanya 1 (nilai k = 1) maka fungsi KNN memberikan *return* berupa nilai variabel huruf dari `distance` indeks ke-0 (`distance[0]`).

Sebaliknya, jika panjang *array* `distance` lebih dari satu, akan diinisialisasi *array* `vote` bertipe `Neighbour` sepanjang nilai k. Array `vote` untuk menyimpan hasil selama proses klasifikasi.

Perulangan pertama (baris 16-21) bertujuan untuk mengisi nilai variabel `karakter` pada *array* `vote` secara unik. Misal pada nilai k = 5, variabel `karakter` pada *array* `distance` adalah [‘A’, ‘A’, ‘B’, ‘B’, ‘C’] maka setelah perulangan akan diperoleh variabel `karakter` pada *array* `vote` berisi [‘A’, ‘B’, ‘C’, *null*, *null*].

Perulangan kedua (baris 23-27) bertujuan untuk mengisi nilai variabel `jarak` pada *array* `vote`. Mula-mula, dengan fungsi `GetIndex()`, akan dicari indeks pada *array* `vote` yang memiliki karakter yang bersesuaian dengan karakter pada variabel `distance`. Nilai indeks tersebut disimpan ke variabel `idx`.

Misal, `karakter` pada *array* `distance` [‘A’, ‘A’, ‘B’, ‘B’, ‘C’] dan pada *array* `vote` [‘A’, ‘B’, ‘C’, *null*, *null*]. Ketika perulangan mencapai indeks ke-2 pada *array* `distance` (karakter ‘B’) maka variabel `idx` akan bernilai 1, sebab karakter ‘B’ berada pada indeks ke-1 pada *array* `vote`.

Setelah memperoleh nilai `idx`, akan dilakukan perhitungan *voting* (sesuai persamaan 2.18) yang nilainya akan diakumulasikan pada variabel `jarak` di *array* `vote` indeks ke-`idx`.

Perulangan terakhir (baris 33-46) mencari nilai variabel `jarak` terbesar pada *array* `vote` beserta variabel `karakter`-nya. Misal nilai variabel `karakter` dan `jarak` pada *array* `vote` berturut-turut adalah [‘A’ dan 10; ‘B’ dan 20; ‘C’dan 15; *null* dan *null*; *null* dan *null*]. Perulangan akan mengambil pasangan ‘B’ dan 20 sebagai karakter serta *voting* terbesar.

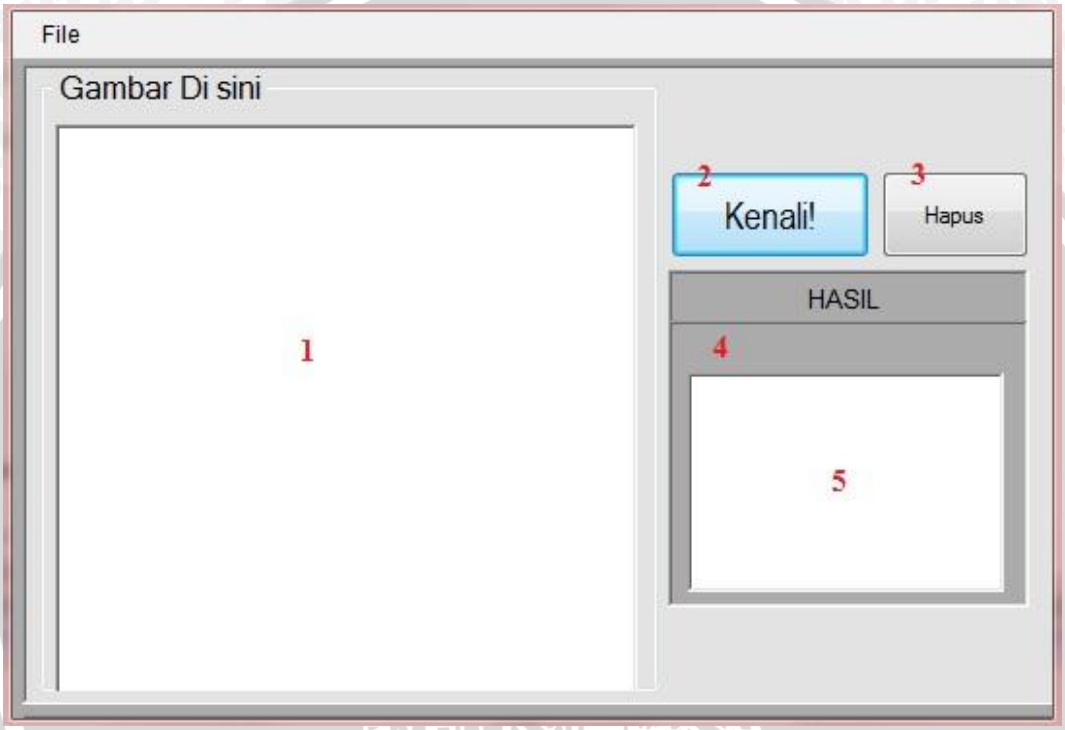
Fungsi `KNN` akan memberikan kembalian (*return*) karakter ‘B’.

4.6. Antarmuka

Desain antarmuka dilakukan sesuai dengan desain pada bab III. Namun selanjutnya dalam penelitian ini diperlukan detail-detail lain yang harus ditampilkan untuk mempermudah penulis. Ada 2 buah antarmuka dalam aplikasi ini, yaitu antarmuka untuk pengguna biasa dan untuk administrator.

4.6.1. Antarmuka pengguna biasa

Antarmuka untuk pengguna biasa ini dapat dilihat pada Gambar 4.1. Di dalam antarmuka yang disediakan untuk pengguna biasa, pengguna dapat menggambar banyak huruf dalam tempat yang disediakan (nomor 1). Kemudian sistem akan menampilkan hasilnya pada tempat yang ditunjukkan pada nomor 4.



Gambar 4.1 Implementasi antarmuka pengguna biasa

Beberapa bagian lain dalam antarmuka ini antara lain: Tombol Kenali (nomor 2) digunakan untuk mengenali tulisan tangan yang dimasukkan pengguna. Tombol Hapus (nomor 3) digunakan untuk membersihkan tempat gambar. Textbox nomor 5 menampilkan nilai 7 parameter Momen Invarian dari masing-masing objek yang ditemukan sistem dari kanvas nomor 1.

4.6.2. Antarmuka administrator

Antarmuka administrator dapat dilihat pada Gambar 4.2. Antarmuka ini digunakan untuk memasukkan data *training* dalam basis data. Dalam antarmuka ini, administrator hanya mampu memasukkan satu buah huruf untuk dikenali.

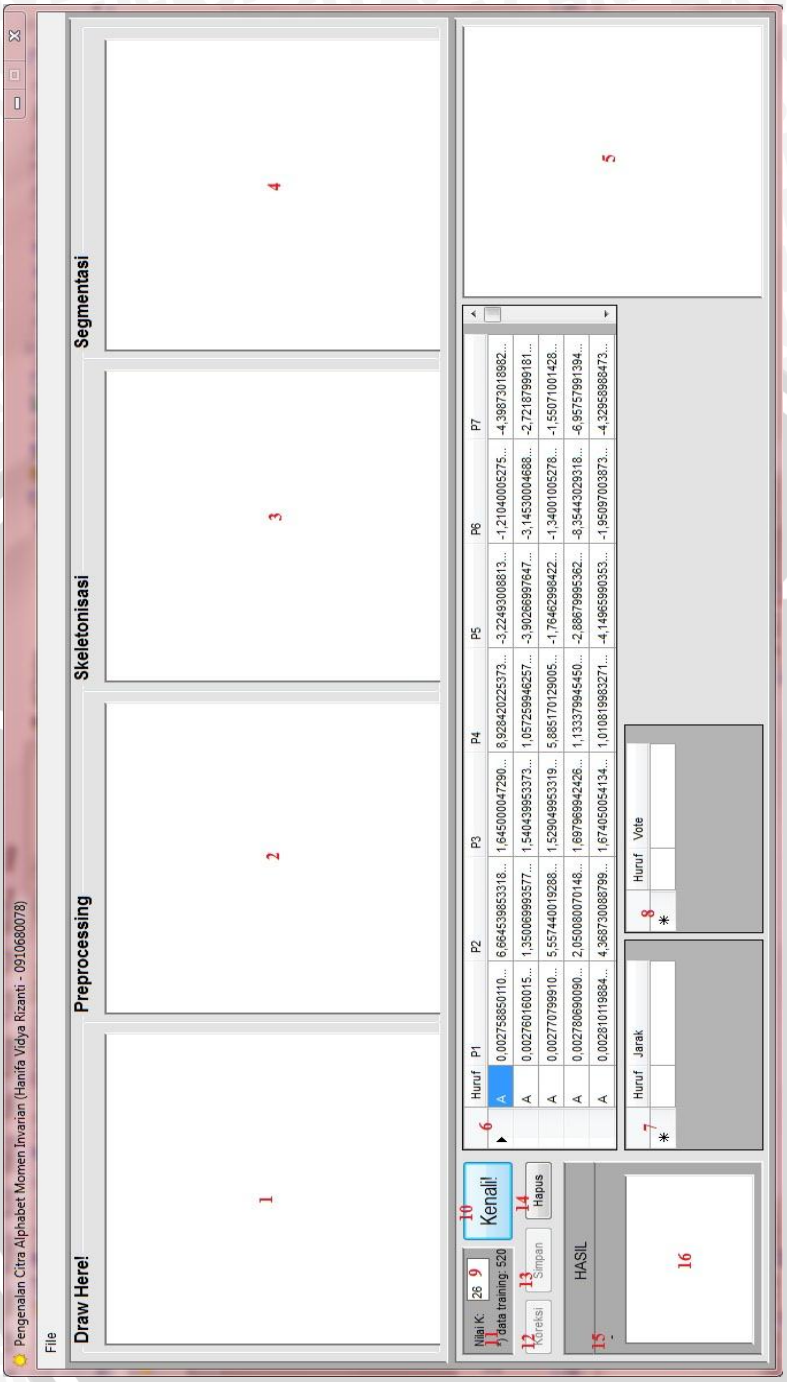
Terdapat 4 buah tombol dalam antarmuka ini. Yang pertama adalah tombol Kenali (nomor 10), berikutnya adalah tombol Hapus (nomor 14). Kedua tombol ini memiliki fungsi yang sama dengan tombol Kenali dan tombol Hapus pada antarmuka untuk pengguna biasa. Tombol berikutnya adalah tombol koreksi (nomor 12). Tombol Koreksi digunakan untuk memperbaiki hasil pengenalan, ketika sistem salah mengenali. Dan terakhir adalah tombol Simpan (nomor 13), yang digunakan untuk memasukkan hasil pengenalan berupa 7 nilai momen ke dalam basis data.

Label nomor 15 menampilkan karakter/huruf hasil pengenalan, dan RichTextBox 16 menampilkan tujuh nilai parameter Momen Invariannya.

Bagian lain dalam antarmuka ini adalah kanvas (nomor 1) sebagai media untuk menampilkan hasil coretan/input data *training*. Kanvas nomor 2, 3, dan 4 beturut-turut menampilkan hasil proses *preprocessing*, skeletonisasi, dan segmentasi. Kanvas nomor 5 menampilkan hasil *crop* skeleton yang sudah di dilasi sebanyak 4 kali.

Tabel nomor 6 menampilkan seluruh data *training* yang diambil langsung dari *database*. Tabel nomor 7 menampilkan k-data terdekat terhadap data uji beserta jaraknya. Sedangkan tabel nomor 8 menampilkan hasil perhitungan *vote* dari setiap huruf dari tabel nomor 7.

Textbox nomor 9 adalah media bagi administrator untuk mengubah banyak tetangga (nilai k) untuk proses K-NN. Label nomor 11 menampilkan banyaknya data *training* yang digunakan.

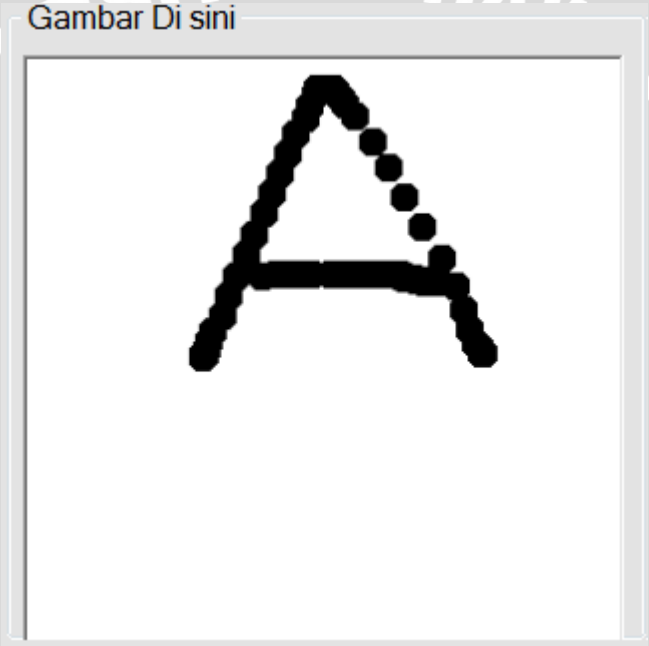


Gambar 4.2 Antarmuka Administrator

BAB V
PENGUJIAN DAN ANALISIS

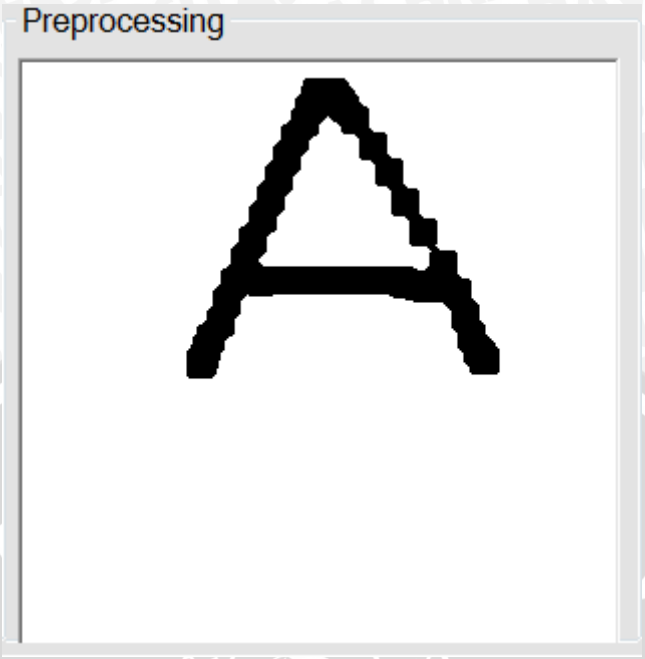
Pengujian sistem mengunakan aplikasi untuk administrator, agar data *training* dapat disimpan, ditambah, dan nilai K dapat diubah selama percobaan. Berikut adalah contoh hasil percobaan salah satu kasus yang diuji menggunakan aplikasi untuk administrator:

1. Gambar/citra *input*:



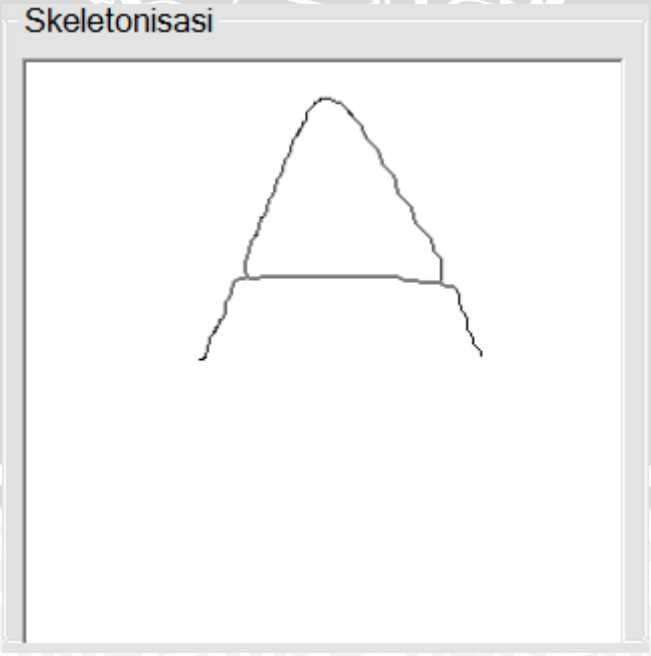
Gambar 5.1 Contoh citra *input*

2. Setelah itu, sistem melakukan proses *prerocessing* pada citra, yaitu binerisasi, dilasi, dan erosi menjadi:



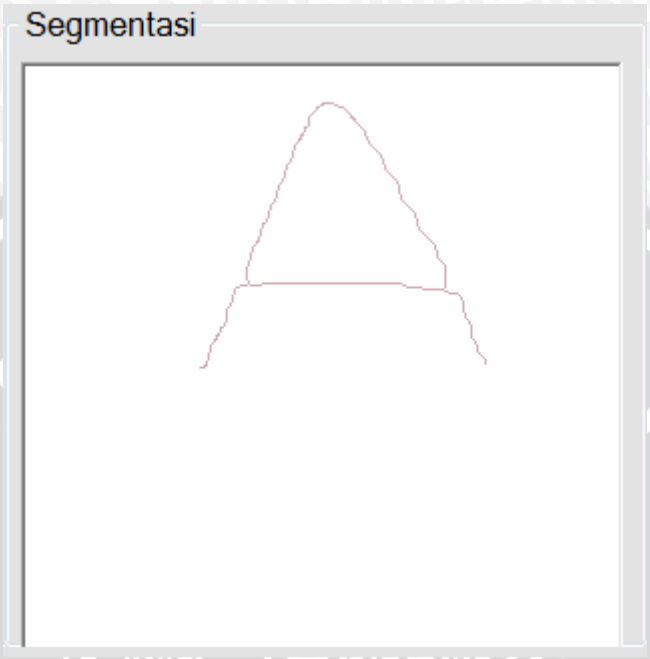
Gambar 5.2 Hasil preprocessing citra *input*

3. Proses selanjutnya yang dilakukan sistem adalah men-*skeleton* citra hasil *preprocessing*, lalu menampilkan hasilnya:



Gambar 5.3 Hasil skeletonisasi citra

4. Dari *skeleton* yang diperoleh, dilakukan proses segmentasi *Region Growing* dengan memberi warna setiap objek berbeda yang ditemui sistem dari citra. Hasilnya:



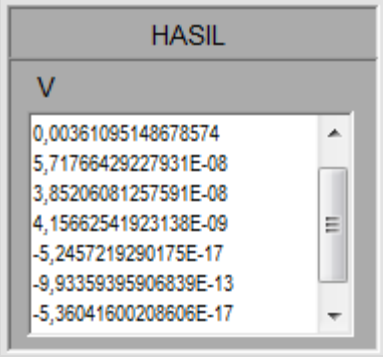
Gambar 5.4 Hasil segmentasi *Region Growing* citra

Pada proses selanjutnya, perhitungan parameter Momen Invarian serta proses pengenalan tidak menampilkan gambar/citra apapun. Proses yang berjalan pada sistem tidak tampak pada *user interface*.

Proses perhitungan parameter Momen Invarian akan menampilkan tujuh buah bilangan pada *RichTextBox* yang disediakan oleh aplikasi. Sebagai contoh, berikut adalah citra input yang diberikan kepada sistem:

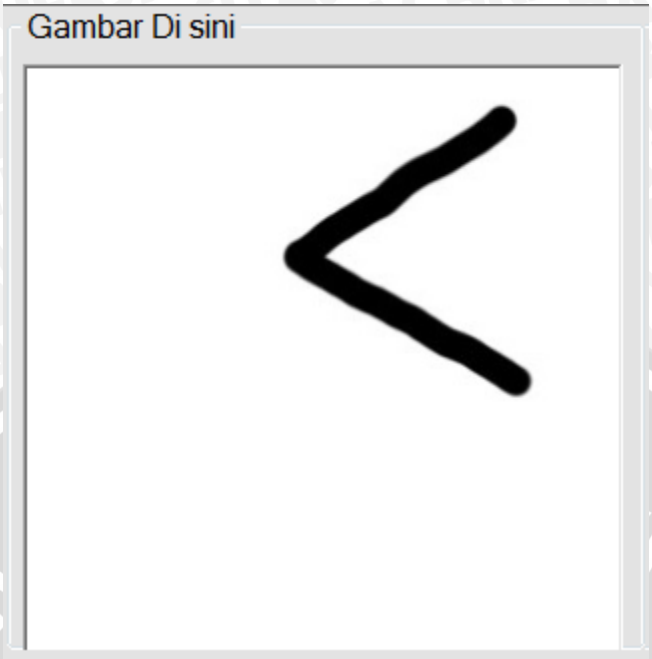
Gambar 5.5 Citra alphabet ‘V’ tegak

Citra alphabet ‘V’ pada Gambar 5.5 selanjutnya akan menghasilkan tujuh momen parameter, yaitu:



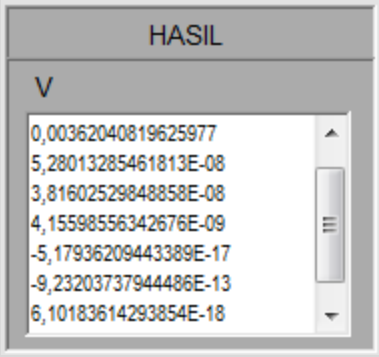
Gambar 5.6 Momen Invarian citra ‘V’ tegak

Berdasarkan sifat metode Momen Invarian yang invarian (tidak berubah) terhadap rotasi, maka dilakukan pengujian dengan menginputkan citra ‘V’ yang telah dirotasi 90^{o} searah jarum jam.



Gambar 5.7 Citra alphabet ‘V’ dirotasi

Setelah citra ‘V’ pada Gambar 5.7 di-*input*-kan, sistem memberikan hasil tujuh parameter Momen Invariannya:



Gambar 5.8 Momen Invarian citra ‘V’ dirotasi

Melalui hasil perhitungan parameter Momen Invarian yang diperoleh kedua citra ‘V’, terlihat bahwa ketujuh Momen Invarian yang dihasilkan oleh keduanya memiliki nilai yang berbeda, namun perbedaannya sangat tipis. Perbedaan sangat tipis ini terjadi karena sifat Momen Invarian itu sendiri yang invarian terhadap rotasi.

Sifat invarian ini ditunjukkan oleh ketujuh parameter milik ‘V’ dirotasi yang mirip parameter ‘V’ tegak, sehingga kedua citra sama-sama dikenali sebagai ‘V’ tegak.

Skripsi ini bertujuan untuk menguji metode Momen Invarian dan CBR dengan K-NN untuk pengenalan tulisan tangan. Beberapa percobaan dilakukan dengan dua buah parameter pengujian, yaitu jumlah kasus yang digunakan sebagai bahan pengetahuan, dan besar nilai K yang digunakan.

Beberapa jumlah kasus yang digunakan untuk pengetahuan adalah 5, 10, 15, ... , dan 60 kasus untuk masing-masing huruf. Masing-masing kondisi dengan jumlah pengetahuan yang berbeda-beda tersebut dicoba dengan menggunakan nilai K yang berbeda-beda pula dalam implementasi K-NNnya. Nilai K yang digunakan antara lain 3, 5, 10, 15, dan 26. Percobaan dengan berbagai kondisi nilai K dan jumlah pengetahuan digunakan untuk mencari kombinasi kedua nilai tersebut agar menghasilkan tingkat akurasi yang baik.

Percobaan dilakukan untuk semua huruf. Masing-masing huruf diujikan sebanyak 3 kali. Hasil percobaan untuk nilai akurasi ini dapat dilihat pada Tabel 5.1 dan 5.2. Dan hasil detail dari setiap percobaan untuk tiap hurufnya dapat dilihat pada Lampiran 1.

Tabel 5.1 Hasil akurasi dengan perhitungan jarak Euclidean

	Jumlah Pengetahuan (per huruf)					
K	5	10	15	20	25	30
3	71,79%	76,92%	78,21%	80,77%	82,05%	82,05%
5	78,21%	76,92%	83,33%	84,62%	82,05%	82,05%
10	74,36%	79,49%	83,33%	87,18%	82,05%	85,90%
15	78,21%	78,21%	82,05%	85,90%	83,33%	87,18%
20	79,49%	79,49%	85,90%	82,05%	85,90%	85,90%
26	76,92%	78,21%	87,18%	85,90%	82,05%	87,18%
K	35	40	45	50	55	60
3	83,33%	83,33%	87,18%	83,33%	84,62%	83,33%
5	83,33%	82,05%	85,90%	83,33%	84,62%	87,18%
10	84,62%	85,90%	87,18%	83,33%	88,46%	88,46%
15	84,62%	85,90%	87,18%	84,62%	88,46%	87,18%
20	87,18%	84,62%	87,18%	83,33%	85,90%	85,90%
26	84,62%	84,62%	89,74%	88,46%	88,46%	87,18%

Tabel 5.2 Hasil akurasi dengan perhitungan jarak Manhattan

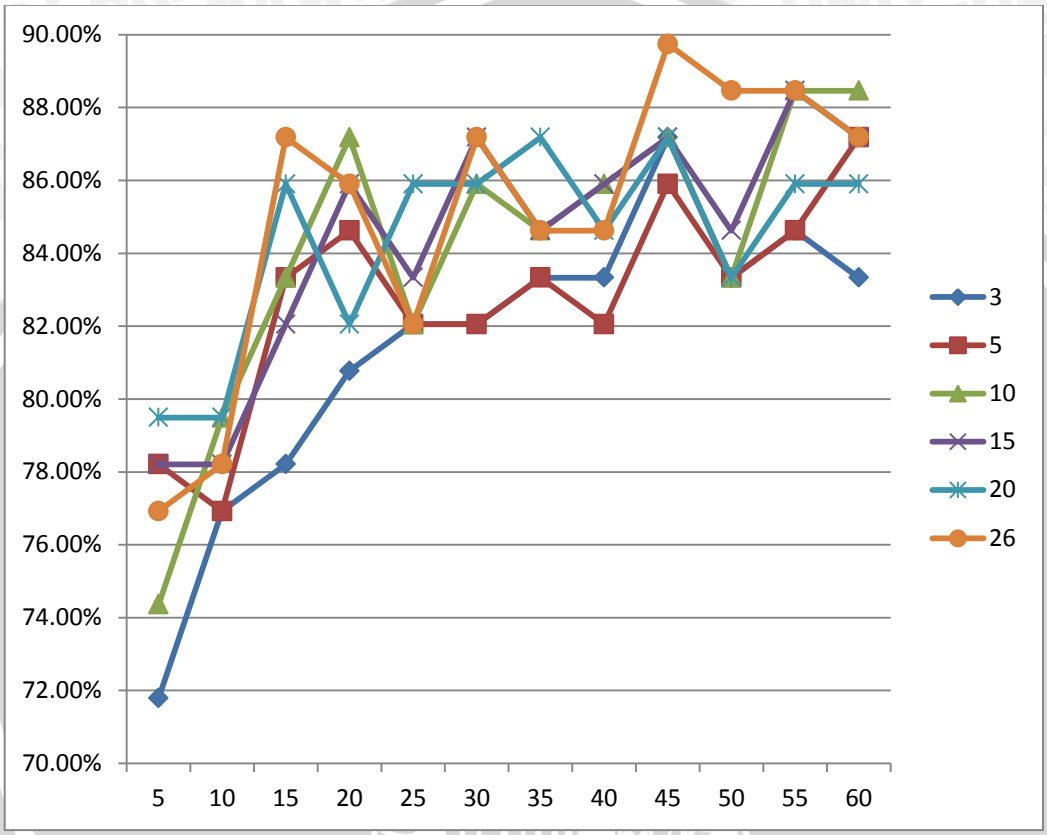
	Jumlah Pengetahuan (per huruf)					
K	5	10	15	20	25	30
3	71,79%	78,21%	82,05%	83,33%	82,05%	83,33%
5	75,64%	79,49%	80,77%	83,33%	83,33%	84,62%
10	76,92%	84,62%	84,62%	87,18%	85,90%	87,18%
15	75,64%	80,77%	85,90%	88,46%	89,74%	89,74%
20	76,92%	82,05%	88,46%	89,74%	87,18%	88,46%
26	78,21%	80,77%	87,18%	91,03%	88,46%	88,46%
K	35	40	45	50	55	60
3	83,33%	85,90%	87,18%	83,33%	84,62%	84,62%
5	82,05%	85,90%	89,74%	85,90%	85,90%	88,46%
10	87,18%	85,90%	88,46%	85,90%	89,74%	88,46%
15	89,74%	84,62%	89,74%	88,46%	89,74%	89,74%
20	89,74%	85,90%	89,74%	91,03%	92,31%	93,59%
26	91,03%	87,18%	89,74%	91,03%	91,03%	92,31%

Tabel 5.1 dan Tabel 5.2 menunjukkan hasil uji coba sistem menggunakan dua perhitungan jarak, yaitu Euclidean dan Manhattan. Dapat disimpulkan bahwa uji coba sistem dengan perhitungan jarak Manhattan pada proses KNN memberikan akurasi yang lebih baik daripada jarak Euclidean, yaitu sebesar 93,59%.

Akurasi sistem menggunakan jarak Manhattan lebih baik karena perhitungan jarak Manhattan lebih memperhatikan vektor jarak antartitik daripada perhitungan jarak Euclidean. Pada perhitungan jarak Euclidean, jarak yang dihasilkan adalah jarak yang diperoleh dengan rumus *Phytagoras*. Rumus *Phytagoras* ini akan menghasilkan satu buah jarak, sehingga vektor jarak antartitik yang dicari jaraknya tidak menjadi pertimbangan lagi.

Sedangkan perhitungan jarak Manhattan benar-benar memperhatikan vektor antartitik. Pada sistem ini, terdapat tujuh dimensi (parameter Momen Invarian) sehingga ada tujuh pertimbangan bagi sistem untuk menetapkan jarak Manhattan. Tujuh pertimbangan jarak Manhattan ini lebih teliti daripada pertimbangan jarak Euclidean yang hanya satu, sehingga akurasi sistem dengan jarak Manhattan lebih baik.

Dari Tabel 5.1 didapatkan 72 hasil percobaan dengan tingkat akurasi yang berbeda-beda. Rata-rata tingkat akurasinya cenderung meningkat ketika semakin banyak jumlah kasus yang diimplementasikan sebagai pengetahuan. Untuk lebih jelasnya, hasil percobaan dapat dilihat pada grafik Gambar 5.9.



Gambar 5.9 Grafik hasil percobaan

Grafik Gambar 5.9 menunjukkan bahwa meski akurasi mempunyai kecenderungan naik, ada kalanya akurasi turun misalnya pada K = 10, akurasi terlihat naik dan turun walau tidak banyak. Naik dan turunnya performa ini disebabkan oleh data *training* baru yang dimasukkan memiliki bentuk berbeda (agak miring, bengkok, dan sebagainya) dengan data *training* lama yang sudah ada di *database*. Masuknya data *training* baru yang berbeda ini mempengaruhi *range* Momen Invarian untuk setiap kelas (huruf).

Begitu pula dengan naiknya performa sistem, menunjukkan bahwa data *training* baru yang dimasukkan memiliki bentuk yang baik sehingga akurasi meningkat.

Nilai akurasi terbesar terletak pada percobaan dengan nilai K = 26 dan jumlah kasus untuk pengetahuan sebanyak 45 untuk masing-masing huruf. Akurasi terbaik ini diperoleh karena besarnya nilai K mencakup semua kelas huruf dan pengetahuan (*training*) huruf yang dimasukkan semakin baik sehingga performa sistem maksimal.

Dari beberapa hasil percobaan didapatkan beberapa hasil yang salah. Huruf yang paling sering salah dikenali adalah huruf N dan Z. Kedua huruf ini sering tertukar. Hal ini desebabkan karena keduanya memiliki bentuk yang sama. Jika N dirotasi 90 derajat searah jarum jam, maka bentuknya akan sama dengan Z. Analisis kesalahan ini didukung dengan pengujian terhadap huruf ‘V’ tegak dan ‘V’ dirotasi yang telah dilakukan, memberikan hasil bahwa huruf tegak dan huruf dirotasi memiliki tujuh parameter Momen Invarian yang nyaris sama sehingga dikenali sebagai huruf yang sama.

Yang sering salah dikenali juga adalah huruf S. Huruf S juga memiliki bentuk yang hampir sama dengan Z hanya saja S adalah hasil *mirroring* dari Z. Dalam Momen Invarian, objek yang di-*mirror* akan memiliki 7 nilai momen yang sama, kecuali nilai ke 7 yang bernilai negatif. Dalam proses perhitungan jarak, nilai negatif pada 7 nilai momen diabaikan, sehingga terkadang, nilai S juga dikenali sebagai nilai Z atau sebaliknya.

Kasus seperti itu masih mungkin dapat diatasi dengan memerhatikan orientasi huruf. Hal ini dapat dilakukan pada penelitian lebih lanjut.

Beberapa kesalahahan lainnya dapat disebabkan berbagai hal, misalnya huruf yang diujikan adalah huruf yang benar-benar memiliki bentuk baru. Selain itu kasus untuk bentuk huruf serupa belum pernah dimasukkan ke dalam pengetahuan.

BAB VI
PENUTUP

6.1. Kesimpulan

Berdasarkan perancangan, implementasi dan hasil pengujian dari sistem pengenalan citra alphabet dengan parameter tujuh Momen Invarian, maka didapatkan kesimpulan sebagai berikut:

1. Sistem pengenalan citra alphabet ini menggunakan tujuh parameter Momen Invarian sebagai ekstraksi fitur, *K-Nearest Neighbour* sebagai metode klasifikasi, serta *Case-Based Reasoning* sebagai metode pembelajaran. Oleh karena itu, aplikasi yang telah dibuat menerapkan ketiga metode tersebut di dalamnya. Terdapat dua aplikasi utama, untuk administrator dan *user* biasa. Aplikasi administrator digunakan untuk *training* data, dapat menyimpan data *training* dan mengubah nilai K pada proses K-NN.
2. Sistem pengenalan citra alphabet ini menggunakan K-NN sebagai metode klasifikasi, dan metode ini membutuhkan nilai K (banyak tetangga) untuk proses klasifikasinya. Melalui pengujian, diperoleh hasil bahwa sistem bekerja dengan cukup baik dengan nilai K kurang dari lima, dan perfomanya semakin naik dengan nilai K lebih dari 5. Sementara dengan nilai K lebih dari 20, performa sistem tidak mengalami perubahan yang signifikan; tetap dapat bekerja dengan sangat baik. Untuk pengukuran jarak dengan Euclidian distance, akurasi terbaik (89,74%) diperoleh ketika nilai K adalah 26. Dan untuk pengukuran jarak dengan Manhattan distance, akurasi terbaik (93,59%) diperoleh ketika nilai K adalah 20.

3. Sistem pengenalan citra alphabet ini diuji dengan jumlah data set huruf bervariasi. Satu data set terdiri dari 26 huruf. Melalui pengujian, diperoleh hasil bahwa performa sistem meningkat ketika diuji coba dengan 5, 10, 15, dan 20 data set. Sementara saat diuji dengan lebih dari 20 data set, performa sistem tidak mengalami perubahan signifikan; tetap dapat bekerja dengan baik. Untuk pengukuran jarak dengan Euclidian distance, akurasi terbaik (89,74%) diperoleh ketika sistem diuji dengan 45 data set. Dan untuk pengukuran jarak dengan Manhattan distance, akurasi terbaik (93,59%) diperoleh ketika sistem diuji dengan 60 data set.
4. Melalui pengujian nilai K dan jumlah data set (data *training*) yang digunakan, dapat disimpulkan bahwa peningkatan performa sistem pengenalan citra alphabet besar dipengaruhi oleh banyaknya data set yang digunakan. Ketika data set yang digunakan lebih dari 20 set, berapapun nilai K yang digunakan tidak banyak berpengaruh terhadap performa sistem. Performa sistem terbaik diperoleh saat pengujian K = 26 untuk data set = 45.

6.2. Saran

Skripsi sistem pengenalan citra alphabet ini tidak lepas dari kesalahan. Untuk itu, beberapa saran yang dapat diberikan untuk pengembangan penelitian selanjutnya antara lain:

1. Sistem dapat ditambahkan data *training* dengan bentuk yang lebih bervariasi untuk meningkatkan performa sistem.
2. Pengembangan/penambahan metode ekstraksi ciri untuk melengkapi Momen Invarian dalam menghasilkan tujuh parameter, sehingga tidak invarian terhadap rotasi (contoh pada kasus huruf N, S, dan Z).
3. Objek pengenalan sistem pengenalan citra alphabet dapat dikembangkan untuk mengenali huruf kecil maupun angka. Pengembangan ini dapat dilakukan dengan menambahkan data *training* berupa huruf kecil dan/atau angka, serta menambahkan memodifikasi metode sistem agar dapat membedakan huruf kapital, huruf kecil, dan angka.

DAFTAR PUSTAKA

[CHA-98] Chan, Kam-Fai and Dit-Yan Yeung. Recognizing On-line Handwritten Alphanumeric Characters Through Flexible Structural Matching. 1998. on the journal of the pattern recognition society. Hongkong.

[CHO-11] Choudary, Amit and Rahul Rishi. Improving the Character Recognition Efficiency of Feed Forward Bp Neural Network. 2011. International Journal of Computer Science & Information Technology (IJCSIT), Vol 3, No 1, Feb 2011

[DEH-97] Dehghan, Mehdi and Karim Faez. Farsi Handwritten Character Recognition With Moment Invariants. 1997. Iran.

[EIK-93] Eikvil, Line. Optical Character Recognition. 1993. Oslo.

[FLU-99] Flusser, Jan. On the independence of rotation moment invariants. 1999. Institute of Information Theory and Automation, Academy of Sciences of the Czech Republic, 182 08 Prague 8, Czech Republic

[FLU-05] Flusser, Jan. Moment Invariants in Image Analysis. 2005. World Academy of Science, Engineering and Technology 11.

[FLU-09] Flusser, Jan and Tomas Suk. 2009. Moment and Moment Invarian in Pattern Recognition. Prague, Czech Republic: WILEY.

[FUJ-92] Fujisawa, Hiromichi and Yasuaki Nakano et al. Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis. 1992. PROCEEDINGS OF THE IEEE VOL 80 NO 7, JULY 1992

[GON-02] Gonzales, R.and R. E. Wood. 2002. *Digital Citra Processing*, Second Edition. Prentice Hall, Inc., New Jersey.

[HEJ-00] Hejlsberg, Anders. 2000. C#Language Reference.doc, version 0.17b

[KEF-12] Kef, Maâmar and Leila Chergui et al. Comparative Study of the Use of Geometrical Moments for Arabic Handwriting Recognition. 2012. Algeria.

[KIM-00] Kim, Gyeonghwan and Sekwang Kim. Feature Selection Using Genetic Algorithms for Handwritten Character Recognition. 2000. Korea.

[KUS-00] Kusniyati ST., Harni. Pengolahan Citra. Pusat Pengembangan Bahan Ajar Universitas Mercubuana

[LAR-05] Larose, T.Daniel. Discovering Knowledge In Data: An Introduction to Data Mining. 2005. New Jersey: Wiley-Interscience.

[MER-05] Mercimek, Muharrem and Kayhan Gulez et al. Real object recognition using moment invariants. 2005. Turkey.

[NAG-66] Nagy, G. and G. L. Shelton, Jr. Self-Corrective Character Recognition System. 1966. New York.

[OHJ-01] Oh, Jong. An On-Line Handwriting Recognizer with Fisher Matching, Hypotheses Propagation Network and Context Constraint Models. 2001. New York University.

[PAR-99] Park, Jaehwa. Hierarchical Character Recognition and its Use in Handwritten Ord/Phrase Recognition. 1999. University of New York.

[PUT-10] Putra, Darma. Pengolahan Citra Digital. 2010. Yogyakarta: CV Andi Offset.

[SAN-11] Sangame, S. K and R.J. Ramteke et al. Recognition of Isolated Handwritten Kannada Characters Using Invariant Moments and Chain Code. 2011. India.

[SHI-04] Shiu, Simon C. K. Ney Jersey: Wiley-Interscience, 2004.

[SRI-12] Computer Science Universitas Gajah Mada http://cs.ugm.ac.id/computerscience/2012/05/09/sri-mulyana-m-kom/ (diakses pada 20 September 2012 20:03)

[STA-76] Stallings, William. Approaches to Chinese Character Recognition. 1976. Virginia.

[STE-67] Sternberg, Saul. Two Operations in Character Recognition: Some Evidence from Reaction-Time Measurements. 1967. Bell Telephone Laboratories.

[SUK-01] Sukamto, Rosa Ariani.*Preprocessing* Pengenalan Tulisan Tangan dengan Ciri-ciri Geometrik (*Offline Handwriting Recognition*)

[THE-06] Theodoris, Sergios. 2006. Pattern Recognition: Third Edition. San Diego: Academic Press

[THI-06] Thillou, C'eline Mancas and Bernard Gosselin. Character Segmentation-by-Recognition Using Log-Gabor Filters. 2006. Belgium.

[TRI-95] Trier, Oivind Due and Anil K, Jains et al. Feature Extraction Methods for Character Recognition--A Survey. 1995. Oslo, Norway.

[WAT-97] Watson, Ian. 1997. Applying Case-Based Reasoning: Techniques for Enterprise Systems. San Fransisco, California: Morgan Kauffman Publishers, Inc.

[WHI-83] White, J. M and G. D Rohrer. Image Thresholding for Optical Character Recognition and Other Applications Requiring Character Image Extraction. 1983. IBM J. Res. Develop.

[WIK-12] Wikipedia Ensiklopedi http://en.wikipedia.org/wiki/Image_moment (diakses pada 17 September 2012 15.34)

[WON-93] Wong, Wai-Hong and Wan-Chi Siu et al. Generation of Moment Invariants and Their Uses for Character Recognition. 1993. Hong Kong.

[YHA-81] Yhap, E. F and E. C Grenias. An On-Line Chinese Character Recognition System. 1981. IBM J. Res. Develop.

[YOU-12] Youzhi, Zhang and Wang Jin. Hidden Markov Model with Parameter-Optimized K-Means Clustering for Handwriting Recognition. 2012. China.

LAMPIRAN

Lampiran 1 Hasil Percobaan

DATA = 5; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	F	T	T	2
F	T	F	T	2
G	F	F	T	1
H	T	T	T	3
I	T	T	T	3
J	F	T	T	2
K	F	T	F	1
L	T	T	T	3
M	F	T	T	2
N	F	F	T	1
O	T	F	T	2
P	F	T	T	2
Q	T	T	T	3
R	T	T	T	3
S	T	T	F	2
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	F	F	T	1
Y	T	T	T	3
Z	F	T	F	1
Total Benar				60
Prosentase Benar				76,92%

DATA = 10; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3

D	T	F	T	2
E	F	T	T	2
F	T	F	T	2
G	T	F	T	2
H	T	T	F	2
I	T	T	T	3
J	F	T	T	2
K	T	T	F	2
L	T	T	T	3
M	F	T	T	2
N	F	F	F	0
O	F	F	T	1
P	T	T	T	3
Q	T	T	T	3
R	T	T	T	3
S	T	T	F	2
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	F	F	T	1
Y	T	T	T	3
Z	T	T	F	2
Total Benar				61
Prosentase Benar				78,21%

DATA = 15; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	F	T	2
G	T	T	T	3
H	T	T	T	3
I	T	T	T	3
J	F	T	T	2
K	T	T	F	2
L	T	T	T	3

M	F	T	T	2
N	F	F	T	1
O	F	T	T	2
P	T	T	T	3
Q	T	T	T	3
R	T	T	T	3
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	T	T	F	2
Total Benar				68
Prosentase Benar				87,18%

DATA = 20; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	F	T	2
H	T	T	F	2
I	T	T	T	3
J	F	T	T	2
K	T	T	F	2
L	T	T	T	3
M	F	T	T	2
N	F	F	T	1
O	F	T	T	2
P	T	T	T	3
Q	T	T	T	3
R	T	T	T	3
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3

V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	T	T	F	2
Total Benar				67
Prosentase Benar				85,90%

DATA = 25; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	T	T	3
I	T	T	T	3
J	F	T	T	2
K	T	F	F	1
L	T	T	T	3
M	F	T	T	2
N	F	F	T	1
O	F	F	F	0
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	F	2
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	T	T	F	2
Total Benar				64
Prosentase Benar				82,05%

DATA = 30; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	T	T	3
I	T	T	T	3
J	F	T	T	2
K	T	F	T	2
L	T	T	T	3
M	F	T	T	2
N	F	F	T	1
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				68
Prosentase Benar				87,18%

DATA = 35; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	F	T	2
H	T	T	T	3

I	T	T	T	3
J	T	T	T	3
K	T	F	F	1
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	F	T	F	1
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				66
Prosentase Benar				84,62%

DATA = 40; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	F	T	2
H	T	F	F	1
I	T	T	T	3
J	T	T	T	3
K	T	F	F	1
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3

R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				66
Prosentase Benar				84,62%

DATA = 45; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	F	T	2
I	T	T	T	3
J	T	T	T	3
K	T	T	T	3
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2

Y	T	T	T	3
Z	T	F	F	1
Total Benar				70
Prosentase Benar				89,74%

DATA = 50; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	F	F	1
I	T	T	T	3
J	T	T	T	3
K	T	T	F	2
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	T	T	F	2
Total Benar				69
Prosentase Benar				88,46%

DATA = 55; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3

D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	F	T	T	2
I	T	T	T	3
J	T	T	T	3
K	T	T	F	2
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				69
Prosentase Benar				88,46%

DATA = 60; K = 26	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	F	T	T	2
I	T	T	T	3
J	T	T	T	3
K	T	T	F	2
L	T	T	T	3

M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	F	2
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	F	1
Y	T	T	T	3
Z	T	T	T	3
Total Benar				69
Prosentase Benar				88,46%

DATA = 45; K = 3	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	T	T	3
I	T	T	T	3
J	T	T	T	3
K	T	F	F	1
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	F	2
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3

V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				68
Prosentase Benar				87,18%

DATA = 45; K = 5	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	T	T	3
I	T	T	T	3
J	T	T	T	3
K	T	F	F	1
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	F	2
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				68
Prosentase Benar				87,18%

DATA = 45; K = 10	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	F	T	2
I	T	T	T	3
J	T	T	T	3
K	T	F	F	1
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				68
Prosentase Benar				87,18%

DATA = 45; K = 15	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	F	T	2

I	T	T	T	3
J	T	T	T	3
K	T	F	F	1
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3
R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	T	F	1
Total Benar				68
Prosentase Benar				87,18%

DATA = 45; K = 20	Percobaan I	Percobaan II	Percobaan III	Jumlah Benar
A	T	T	T	3
B	T	T	T	3
C	T	T	T	3
D	T	F	T	2
E	T	T	T	3
F	T	T	T	3
G	T	T	T	3
H	T	F	T	2
I	T	T	T	3
J	T	T	T	3
K	T	F	T	2
L	T	T	T	3
M	F	T	T	2
N	F	T	T	2
O	T	T	T	3
P	T	T	T	3
Q	T	T	T	3

R	T	F	T	2
S	T	T	T	3
T	T	T	T	3
U	T	T	T	3
V	T	T	T	3
W	T	T	T	3
X	T	F	T	2
Y	T	T	T	3
Z	F	F	F	0
Total Benar				68
Prosentase Benar				87,18%