PENGENALAN ALPHABET BAHASA ISYARAT TANGAN MENGGUNAKAN SMART K-*NEAREST NEIGHBOR* (SMART KNN) UNTUK TUNA RUNGU DAN TUNA WICARA

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Gilbert Dany Naviri
NIM: 115060900111037

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN
PROGRAM STUDI INFORMATIKA
PROGRAM TEKNOLOGI INFORMASI DAN ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2015

PENGESAHAN

PENGENALAN ALPHABET BAHASA ISYARAT TANGAN MENGGUNAKAN SMART K-*NEAREST NEIGHBOR* (SMART KNN) UNTUK TUNA RUNGU DAN TUNA WICARA

SKRIPSI

KEMINATAN TEKNIK KOMPUTER

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Gilbert Dany Naviri
NIM: 115060900111037

Skripsi ini telah diuji dan dinyatakan lulus pada
3 Desember 2015
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Fitri Utaminingrum, Dr. Eng., S.T., M. T	Hurriyatul Fitriyah, S.T., M.Sc.
NIP: 19820710 200812 2 001	NIP: 2013048510012001

Mengetahui
Ketua Program Studi Informatika

Marji, Drs., M.T
NIP: 19670801 199203 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 3 Desember 2015

Gilbert Dany Naviri

NIM: 115060900111037

KATA PENGANTAR

Puji syukur kehadirat Allah SWT, berkat rahmat dan karunia –Nya penulis dapat menyelesaikan penyusunan skripsi dengan baik. Shalawat salam smoga senantiasa terlimpah kepada Nabi Muhammad SAW. Amin.

Penulisan skripsi ini diajukan untuk memenuhi salah satu syarat mendapatkan gelar Sarjana pada Program Studi Sistem Komputer, Fakultas Ilmu Komputer, Universitas Brawijaya Malang. Judul skripsi yang penulis ajukan adalah "Pengenalan Alphabet Bahasa Isyarat Menggunakan SMART K-*Nearest Neighbor* (SMART KNN) untuk Tuna Rungu dan Tuna Wicara".

Dalam penyusunan dan penulisan ini tidak terlepas dari bantuan, bimbingan serta dukungan dari berbagai pihak. Oleh karena itu penulis menyampaikan banyak terimakasih kepada yang terhormat :

1. Bapak Prof. Dr. Ir. Mohammad Bisri, M.S. selaku Rektor Universitas Brawijaya Malang yang telah mengesahkan penulisan skripsi ini
2. Bapak Ir. Sutrisno, M.T. selaku Dekan Fakultas Ilmu Komputer yang telah memberikan izin untuk melakukan penelitian
3. Bapak Adharul Muttaqin, S.T., M.T. selaku Kepala Program Studi Sistem Komputer yang telah mendukung dilaksanakannya penelitian skripsi ini
4. Bapak Marji, Drs., M.T. selaku Kepala Program Studi Informatika yang telah menyetujui dilaksanakannya penelitian skripsi ini
5. Bu Fitri Utaminingrum, Dr. Eng., S.T., M.T. selaku dosen pembimbing 1 yang telah mendukung, membimbing dan mendidik penulis, sehingga pengerjaan skripsi ini bisa selesai tepat waktu
6. Bu Hurriyatul Fitriyah, S.T., M.Sc. selaku dosen pembimbing 2 yang telah memberikan waktunya untuk membimbing penulis, sehingga penulisan dan penelitian skripsi ini bisa selesai dengan baik
7. Nenek, Ibu dan seluruh keluarga penulis yang telah memberikan dukungan dan doa, segala fasilitas yang dibutuhkan untuk menyelesaikan penelitian skripsi ini
8. Semua pihak yang telah membantu penulis menyelesaikan skripsi ini

Semoga Allah SWT memberikan balasan yang berlipat ganda kepada semuanya. Demi perbaikan selanjutnya, kritik dan saran sangat penulis butuhkan. Akhir kata semoga hasil penulisan ini bisa bermanfaat bagi semua.

Malang, 3 Desember 2015

Penulis

Gilbert Dany Naviri

ABSTRAK

Penyandang tuna rungu dan tuna wicara menggunakan alphabet bahasa isyarat sebagai salah satu sarana komunikasi berdasarkan pola tangan yang dibentuk. Tetapi masih sedikit orang yang mengerti alphabet bahasa isyarat tangan. Penelitian ini dibuat sebagai penerjemah alphabet bahasa isyarat ke bentuk teks. Sehingga diharapkan mempermudah orang mengerti alphabet bahasa isyarat untuk tuna rungu dan tuna wicara. Banyak metode yang bisa digunakan untuk mencapai hasil pengenalan pola tangan yang terbaik. Salah satu metode yang bisa digunakan adalah metode KNN (*K-Nearest Neighbor*). Kekurangan dari metode KNN adalah membutuhkan waktu pembelajaran yang lebih lama pada jumlah data latih yang banyak. Selain itu jumlah data latih yang banyak mengaburkan data latih yang seharusnya lebih dekat dengan data uji atau menurunkan akurasi klasifikasi. Untuk meningkatkan efektifitas klasifikasi KNN, maka KNN bisa dikombinasikan dengan metode yang lain. Beberapa kombinasi KNN dengan metode lain untuk meningkatkan akurasi klasifikasi diantaranya *Random* KNN, *Tree* KNN dan Fuzzy KNN. Kombinasi metode KNN dengan metode lain terbukti menghasilkan akurasi klasifikasi yang lebih baik. Pembobotan SMART (*Simple Multi Attribute Rating Technique*) menghasilkan keluaran (*output*) yang spesifik sesuai nilai akhirnya. Pembobotan SMART dimulai dengan menentukan kriteria pembobotan, menentukan bobot kriteria dan normalisasinya, menentukan nilai utilitas dan terakhir menghasilkan grup rekomendasi yang beranggotakan kumpulan alphabet tertentu. Penelitian ini menggabungkan pembobotan SMART dan klasifikasi KNN untuk pengenalan bahasa isyarat tangan menjadi bentuk teks. Hasil penelitian ini adalah program simulasi dengan rata-rata akurasi 93%. Penelitian ini menunjukkan hasil segmentasi citra tangan dengan sedikit noise, akurasi klasifikasi tinggi dan berhasil menerjemahkan alphabet bahasa isyarat tangan ke bentuk teks.

Kata Kunci : Alphabet Bahasa Isyarat Tangan, KNN (*K-Nearest Neighbor*), SMART (*Simple Multi Attribute Rating Technique*)

ABSTRACT

The deaf and the mute were used alphabet sign language as one of form communication facility base on hand pattern. But still few people was known that alphabet sign language. This research was created to convert alphabet sign language into text. So hopefully this research could help more people understand about alphabet sign language for the deaf and the mute. When there are many methods can be used to hand pattern recognition, KNN (*K-Nearest Neighbor*) *was choose to apply. The weakness of this method is waste much time processing for big data. Otherwise the big data could obscure data training that should be closest to data testing. It means accuracy will be decreased. The* KNN *combination with another method was recommended to increase the accuracy. Some of* KNN *combination were Tree* KNN*, Random* KNN*,* Fuzzy KNN*, etc. The* KNN *combination with another method was proved has high accuracy.* SMART (*Simple Multi Attribute Rating Technique*) *weighting was given specific output base on the final value.* SMART (*Simple Multi Attribute Rating Technique*) *weighting was started from determine criterias of weithing, the weighting criterias with normalizations, utility criteria and for the last given group recommendation. This research was combining the* SMART *weighting with* KNN *classification to converted alphabet sign language into text. Result of this research was displayed as simulation program that has 93% average accuracy. This research showed segmentation image with bit noise, high accuracy and succeed convert the alphabet sign language for the deaf and the mute into text.*

Keyword: Sign Language, KNN (*K-Nearest Neighbor*), SMART (*Simple Multi Attribute Rating Technique*)

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 1 PENDAHULUAN

Pendahuluan menjelaskan pentingnya penelitian dan tahapan-tahapan Pengenalan Alphabet Bahasa Isyarat Menggunakan SMART K-*Nearest Neighbor* (SMART KNN) untuk Tuna Rungu dan Tuna Wicara.

1.1. Latarbelakang masalah

Penyandang tuna rungu dan tuna wicara menggunakan alphabet bahasa isyarat sebagai salah satu sarana komunikasi berdasarkan pola tangan yang dibentuk. Tetapi masih sedikit orang yang mengerti alphabet bahasa isyarat tangan. Penelitian ini dibuat sebagai penerjemah alphabet bahasa isyarat ke bentuk teks. Sehingga diharapkan mempermudah orang mengerti alphabet bahasa isyarat untuk tuna rungu dan tuna wicara.

Banyak metode yang bisa digunakan untuk mencapai hasil pengenalan pola tangan yang terbaik. Salah satu metode yang bisa digunakan adalah metode KNN (*K-Nearest Neighbor*). Kekurangan dari metode KNN adalah membutuhkan waktu pembelajaran yang lebih lama pada jumlah data latih yang banyak. Selain itu jumlah data latih yang banyak mengaburkan data latih yang seharusnya lebih dekat dengan data uji atau menurunkan akurasi klasifikasi. Untuk meningkatkan efektifitas klasifikasi KNN, maka KNN bisa dikombinasikan dengan metode yang lain. Beberapa kombinasi KNN dengan metode lain untuk meningkatkan akurasi klasifikasi diantaranya *Random* KNN, *Tree* KNN dan Fuzzy KNN, menunjukkan bahwa metode *K-Nearest Neighbor* (*KNN*) menarik untuk dikaji. Dengan rumus perhitungannya yang sangat sederhana, seharusnya berbanding lurus dengan efisiensi waktu eksekusi sistem. Seperti penelitian dari Silva dan Hernandez yang berhasil menunjukkan bahwa hasil kombinasi antara *Self-Organizing Maps Artificial Network* (SOM) dan *K-Nearest Neighbor* (KNN) memberikan hasil yang lebih baik daripada menggunakan klasifikasi yang berdiri sendiri untuk klasifikasi. Hasil akhirnya perbandingan antara klasifikasi SOM, KNN dan SOM-KNN yaitu ratio keberhasilan 89,48; 84,23; dan 91,03 [SIL-11].

Pembobotan SMART (*Simple Multi Attribute Rating Technique*) menghasilkan keluaran (*output*) yang spesifik sesuai nilai akhirnya. Pembobotan SMART dimulai dengan menentukan kriteria pembobotan, menentukan bobot kriteria dan normalisasinya, menentukan nilai utilitas dan terakhir menghasilkan grup rekomendasi yang beranggotakan kumpulan alphabet tertentu. Pada suatu penelitian yang membandingkan metode *Analytical Hierarchy Process* (AHP) dan SMART, dimana setiap indikator yang sebelumnya diolah menggunakan AHP, diujicobakan dengan SMART. Salah satu hasilnya menunjukkan bahwa pembobotan SMART terbukti mampu memberikan hasil rekomendasi yang lebih baik untuk membantu mengambil keputusan saat disatukan dengan algoritma AHP [KAS-13].

Berdasarkan penelitian tersebut menunjukkan SMART yang merupakan algoritma sederhana yang memberikan rekomendasi klasifikasi yang lebih baik saat ditambahkan dengan algoritma lain.

Dari latar belakang tersebut, Penelitian ini menggabungkan pembobotan SMART dan klasifikasi KNN untuk pengenalan bahasa isyarat tangan menjadi bentuk teks. Tahapan awal penelitian ini adalah melakukan proses untuk menangkap citra tangan. Kemudian dengan pengolahan citra melalui metode segmentasi untuk membedakan antara tangan dan latarbelakangnya. Hasil pengolahan citra tersebut kemudian diproses kembali untuk mendapatkan titik tengah dan ordinat tepian tangannya. Terakhir pengenalan alphabet bahasa isyarat dikenali dengan menggunakan metode *Simple Multi Attribute Rating Technique* (SMART) dan klasifikasi K-*Nearest Neighbour* (KNN).

1.2. Rumusan masalah

Berikut uraian permasalahan yang diangkat untuk menyusun penelitian ini.

1. Bagaimana membuat program simulasi untuk membedakan antara tangan dan latarbelakangnya?
2. Bagaimana menerapkan SMART KNN untuk mengklasifikasi alphabet bahasa isyarat dari citra tangan?
3. Bagaimana mengubah hasil klasifikasinya ke bentuk teks?

1.3. Tujuan

Hasil dari penelitian ini diharapkan mampu menyelesaikan beberapa permasalahan yang telah diuraikan pada poin sebelumnya.

1. Menggunakan pembobotan *SMART* dan metode *KNN* untuk klasifikasi pola tangan berdasar alphabet bahasa isyarat ke dalam bentuk teks
2. Membuat program simulasi pembelajaran alphabet bahasa isyarat penderita tuna rungu dan tuna wicara

1.4. Manfaat

Penelitian ini diharapkan dapat membantu orang normal agar lebih mudah mempelajari alphabet bahasa isyarat penderita tuna rungu dan tuna wicara.

1.5. BATASAN MASALAH

Dalam suatu penelitian harus fokus pada pokok permasalahan yang diangkat. Sehingga diperlukan suatu batasan yang jelas untuk menghindari kerancuan pokok bahasan dari suatu penelitian dengan penelitian yang lain.

1. Gambar masukan yang diolah hanya bagian tangan

2. Hasil dari penelitian ini hanya berbentuk program simulasi
3. *Hand Gesture* yang dimaksud disini adalah pola tangan
4. Bahasa isyarat menggunakan bahasa isyarat versi amerika
5. Tangan yang diuji tidak boleh berwarna hitam atau tidak boleh memiliki background berwarna mirip warna kulit
6. Ukuran citra yang digunakan fix lebar 300 cm dan tinggi 400 cm
7. Kapasitas jumlah huruf dalam teks maksimalnya 5 huruf

1.6. Sistematika penulisan

Penulisan suatu penelitian ilmiah harus sistematis. Sehingga mempermudah pembaca untuk memahami alur penelitian tersebut.

BAB I : PENDAHULUAN

Pendahuluan menjelaskan pentingnya penelitian dan tahapan-tahapan Pengenalan Alphabet Bahasa Isyarat Menggunakan SMART K-*Nearest Neighbor* (SMART KNN) untuk Tuna Rungu dan Tuna Wicara

BAB II : LANDASAN KEPUSTAKAAN

Berisikan ulasan yang menjelaskan dasar teori tentang pengenalan Tuna Rungu dan Tuna Wicara, deteksi pola tangan (*hand gesture*), pembobotan *Simple Multi Attrubute Rating Technique* (SMART) dan *K-Nearest Neighbor* (KNN). Juga kajian terhadap penelitian lain yang relevan tentang penelitian ini.

BAB III : METODE PENELITIAN DAN PERANCANGAN

Dalam bab ini menjelaskan tentang metode atau langkah-langkah penelitian dan perancangan yang digunakan dalam pembuatan program simulasi pengenalan alphabet bahasa isyarat menggunakan pembobotan Simple Multi Attribute Rating Technique (SMART) dan klasifikasi K-Nearest Neighbor (KNN) untuk membantu Tuna Rungu dan Tuna Wicara

BAB IV : IMPLEMENTASI

Dalam bab ini menjelaskan penerapan tahapan perancangan yang telah dibuat sebelumnya. Menggunakan aplikasi compiler Matlab® 2013b untuk membuat GUI sederhana dan mengolah gambar hingga menampilkan hasil klasifikasi

BAB V : PENGUJIAN DAN ANALISA

Dalam bab ini membahas metode untuk menguji program yang dibuat, yaitu pengujian fungsional dengan menghitung akurasi klasifikasi SMART KNN. Serta melakukan analisa data berdasarkan hasil pengujian tersebut

BAB VI : PENUTUP

Dalam bab ini berisikan kesimpulan diperoleh dari hasil pengujian dan analisa hingga saran yang perlu ditambahkan untuk penelitian selanjutnya

DAFTAR PUSTAKA

BAB 2 LANDASAN KEPUSTAKAAN

Berisikan ulasan yang menjelaskan dasar teori tentang pengenalan Tuna Rungu dan Tuna Wicara, deteksi pola tangan (*hand gesture*), pembobotan *Simple Multi Attrubute Rating Technique* (SMART) dan *K-Nearest Neighbor* (KNN). Selain itu juga melalui kajian terhadap penelitian lain yang relevan tentang penelitian ini.

2.1. Kajian pustaka

Penelitian tentang penerjemahan alphabet bahasa isyarat untuk tuna rungu dan tuna wicara sudah pernah dilakukan sebelumnya. Beberapa penelitian tersebut bisa menjadi acuan sebagai bahan pengembangan penelitian sebelumnya.

1. Sebelumnya dilakukan oleh Rahman, Ahsan dan Aktaruzzman, yaitu pengenalan pola tangan menggunakan metode Backpropagation Neural Network. Dengan tangan kosong tanpa penutup apapun sebagai objek yang diteliti menghasilkan akurasi 80,26%. Tapi dalam penelitian tersebut diantara 598 citra yang disiapkan, sebanyak 80% digunakan sebagai data latih dan sisanya sebagai data uji [RAH-11].
2. Selanjutnya penelitian menggunakan Centroid, Roundness dan Scan line sebagai fiturnya. Dalam penelitian ini menggunakan sarung tangan untuk membedakannya dari latarbelakangnya dan membuat rule tertentu untuk klasifikasinya. Sebagai hasil akhirnya penelitian ini menghasilkan akurasi sekitar 81% [CHA-13].
3. Penelitian lain menggunakan algoritma Difference of Gaussian dan Scale Invariant Feature Transform (SIFT) untuk mengenali pola tangannya. Sebanyak 26 alphabet diproses sebagai data latih dan menambahkan 10 alphabet pengulangan dengan beda pencahayaan dan orientasi. 80% dari data yang ada digunakan untuk data latih dan sisanya sebagai data uji. Hasilnya akurasi 100% [MNA-13].

Dari ketiga penelitian di atas menunjukkan batasan-batasan kemampuan yang perlu ditingkatkan untuk menghasilkan program simulasi yang lebih baik. Sehingga menghasilkan program simulasi yang menggunaan data uji berupa hasil tangkapan kamera saat itu juga dan bisa mendeteksi tangan tanpa penutup apapun.

2.2. Pengenalan pola tangan

Pengenalan pola merupakan bagian dari tahap pembelajaran sistem untuk meletakkan pola-pola tertentu ke dalam kategori. Menemukan keterhubungan dan membedakan pola dari masing-masing objek yang diteliti. Pola yang dimaksud bisa berupa objek, proses atau kejadian yang bisa diberi identitas. Dalam penelitian ini

menggunakan pendekatan statistikal, yaitu pendekatan pengenalan pola berdasarkan fitur atau atribut tertentu dengan menghitung kedekatan antar tetangga yang diperoleh dari perhitungan kedekatan tiap fitur data uji dengan data latih. Sehingga menghasilkan klasifikasi berdasarkan jarak terdekatnya dengan data latih.

Agar sistem bisa mengenali pola-pola tersebut, maka diperlukan adanya pembelajaran oleh sistem terlebih dulu. Tahapan pembelajaran ini sangat penting. Karena tanpa adanya pembelajaran oleh sistem (berisikan kumpulan data latih), sistem tidak akan bisa menghasilkan klasifikasi. Tahapan pembelajaran (*learning*) pada umumnya membutuhkan waktu yang lebih lama daripada tahap deteksi. Hal ini lebih disebabkan pada banyaknya jumlah data latih yang perlu dimasukkan ke dalam sistem. Semakin banyak data latih dan spesifik fitur yang dipilih akan membuat sistem menjadi semakin pintar. Fitur yang digunakan dalam penelitian ini menggunakan jarak antara titik tengah tangan ke tepian tangan. Pemilihan fitur ini merupakan pengembangan dari pelitian edukasi anak kecil menggunakan pengenalan pola tangan berbasis smartphone yang dilakukan oleh Deongsok Yang, dkk. Dalam penelitian tersebut menggunakan fitur extrema point. Dimana pengukuran jarak antara titik tengah tangan ke titik-titik tertentu dari tepian tangan yang disajikan ke dalam bentuk linear [YAN-14]. Dalam penelitian fitur tersebut dikembangkan dengan tidak hanya mengukur titik tengah ke titik tertentu dari tepian tangan, melainkan ke seluruh titik tepi tangan, baik sisi kiri dan kanan tepian tangan.

Metode pembelajaran mesin yang ada saat ini ada 2, yaitu *Supervised Learning* yang berarti pembelajaran dengan memasukkan sejumlah data latih berdasar fitur yang dipilih hingga kategorinya dan *Unsupervised Learning* yang merupakan metode pembelajaran yang menetapkan kategori dari sejumlah data latih berdasarkan aturan tertentu. Dalam penelitian ini menggunakan metode *Supervised Learning* yang sesuai dengan penelitian ini terkait pengenalan alphabet bahasa isyarat yang jumlahnya sudah pasti 26 huruf dan tidak dimungkinkan adanya klass baru diluar klass yang sudah ditentukan. Karena masing-masing data sudah memiliki kategori, maka dipilihlah metode klasifikasi *K-Nearest Neighbor* (KNN) yang merupakan metode sederhana dalam klasifikasi yang berdasar pada statistik. KNN ini diperkuat dengan pembobotan *Simple Multi Attrubute Rating Technique* (SMART) untuk memperoleh huruf yang sesuai berdasar masukan citra tangan yang sudah dikenali pola tanganya.

2.3. Pengolahan citra digital

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 bit tertentu [PUT-10]. Dengan demikian, proses deteksi warna kulit, segmentasi, *filtering noise* sampai *cropping* merupakan bagian dari pengolahan citra digital.

2.3.1. Citra Warna RGB

Penggambaran suatu citra digital direpresentasikan ke dalam bentuk dimensi (panjang dan lebar) dan kedalaman warna (bit per pixel). Dengan rumus pengukuran citra seperti pada persamaan (2.1) [WIJ-10].

$$S = \text{Panjang} \times \text{Lebar} \quad \ldots (2.1)$$

Ket :
S = Ukuran citra

Untuk tingkat kedalamannya citra warna dibagi menjadi 3, yaitu 8 bit (*Truecolor*) yang berarti tiap pixel berisikan rentang warna 0-255 dengan format seperti pada tabel 2.1.

Tabel 2.1. Format warna 8 bit

Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
R	R	R	G	G	B	B	B

Kemudian citra warna 16 bit berarti tiap pixel citra tersebut diwakili rentang 0 sampai 65.536 warna. Untuk format penyebaran warna R-G-B seperti pada table 2.2.

Tabel 2.2. Format warna 16 bit

Bit-15	Bit-14	Bit-13	Bit-12	Bit-11	Bit-10	Bit-9	Bit-8	Bit-7	Bit-6	Bit-5	Bit-4	Bit-3	Bit-2	Bit-1	Bit-0
R	R	R	R	R	G	G	G	G	G	G	B	B	B	B	B

Dasar penempatan 5 R (*red*), 6 G (*green*) dan 5 B (*blue*) adalah kepekaan manusia dalam menangkap penyebaran warna yang lebih sensitive terhadap warna hijau. Demikian pula untuk citra warna 24 bit yang berarti terdapat 24 bit warna di

tiap pixelnya. Oleh karena itu terdapat 16.777.216 variasi warna yang bisa diterapkan. Variasi ini merupakan yang paling umum untuk merepresentasikan seluruh warna yang dapat dilihat dengan penglihatan manusia. Dengan penyebaran R-G-B semua dibagi rata menjadi 8 bit R, 8 bit G dan 8 bit B.

2.3.2. Citra Keabuan (*Grayscale*)

Menggambarkan citra ke dalam bentuk keabuan dengan kedalaman 8 bit yang direpresentasikan ke dalam bentuk array dua dimensi. Dimana tiap elemen dalam array menunjukkan intensitas (*greylevel*) dari citra pada posisi koordinat yang bersesuaian. Apabila suatu citra direpresentasikan dalam 8 bit maka berarti pada citra terdapat 28 atau 256 *level grayscale*, (biasanya bernilai 0 – 255), dimana 0 menunjukkan level intensitas paling gelap dan 255 intensitas paling terang [MIN-13].

2.3.3. Citra Biner

Merupakan citra digital yang menggambarkan tiap bit pixel dengan nilai 0 atau 1. Dengan demikian citra yang dihasilkan jelas akan berwarna hitam atau putih (monokrom). Citra ini memiliki kedalaman 1 bit per pixel, sehingga ukuran citra bisa menjadi sangat kecil, karena tinggal tergantung dimensinya saja. Jika dimensi citra 300x400, berarti ukuran citra tersebut 300x400x1 = 120000 bit = 120000/8 byte = 15000/1024 Kb = 14,65 Kb.

2.3.4. Deteksi warna kulit

Untuk memperoleh hasil ekstraksi warna kulit yang sempurna, peneliti membuat ambang batasan warna (*Threshold*) berdasarkan HSV (*Hue-Saturation-Value*) dan YCbCr [ALB-01].

a. RGB to HSV

Model HSV (*Hue-Saturation-Value*) menunjukkan ruang warna dalam bentuk tiga komponen utama, yaitu *Hue, Saturation* dan *Value*. *Hue* adalah sudut dari 0 sampai 360 derajat, biasanya 0 adalah merah, 60 adalah kuning, 120 derajat adalah hijau, 180 derajat adalah cyan, 240 derajat adalah biru, 300 derajat adalah magenta. *Saturation* adalah ukuran seberapa besar kemurnian dari warna tersebut. *Value* dapat bernilai 0 hingga 100%. Warna dengan value 100% akan tampak sangat cerah. Begitupula sebaliknya, jika value 0 maka akan tampak sangat gelap [MEN-12]. Untuk rumus persamaan mengubah RGB ke HSV mengikuti rumus persamaan (2.2).

$$H = \tan\left[\frac{3\,(G-B)}{(R-G)+(R-B)}\right]$$

$$S = 1 - \frac{\min(R,G,B)}{V}$$

$$V = \frac{R+G+B}{3} \qquad \text{.... (2.2)}$$

Keterangan :

H	=	Hue	R	=	Red
S	=	Saturation	G	=	Green
V	=	Value	B	=	Blue

Nilai Hue selanjutnya digunakan untuk menggantikan nilai Y dalam menentukan ambang batas warna kulit.

b. RGB to YCbCR

Dalam proses mendapatkan segmentasi tangan yang baik, membedakan secara jelas antara tangan dengan latarbelakangnya, salah satu algoritma yang bisa digunakan deteksi warna kulit menggunakan YCrCb. Dengan rumus yang sangat sederhana dan mudah diimplementasikan, mengubah nilai RGB menjadi sebaran warna Chromium seperti pada rumus persamaan (2.3).

$$Y = 0.299*R + 0.587*G + 0.114*B$$
$$Cr = (R - Y)*0.713 + 128$$
$$Cb = (B - Y)*0.564 + 128 \qquad \text{... (2.3)}$$

Keterangan :

Y	=	Yue	R	=	Red
Cr	=	Chromium Red	G	=	Green
Cb	=	Chromium Blue	B	=	Blue

Hasil dari persamaan (2.2) dan persamaan (2.3) diatas adalah bentuk biner 0 dan 1, dimana pixel yang memiliki sebaran nilai yang masuk ambang batas CrCb akan diberi biner 1 dan diluar ambang batas diberi biner 0. Dengan demikian hasil segmentasi gambar merupakan kumpulan biner 1 yang membentuk pola tangan dan latar belakangnya menjadi gelap karena merupakan susunan biner 0.

2.3.5. Segmentasi

Tahap ini merupakan bagian dari pra-processing pengolahan citra *digital*. Dimana citra dibagi kedalam wilayah-wilayah tertentu berdasarkan *discontinuity* (perubahan intensitas secara tiba-tiba) atau *similarity* (keserupaan sifat tertentu). Dalam penelitian ini menggunakan segmentasi berdasarkan keserupaan, yaitu pengambangan (*thresholding*) untuk mengubah citra dari bentuk 3D ke 2D yang berwarna keabuan. Lebih dari itu, dengan metode pengambangan ini mengubah citra dari keabuan menjadi bentuk biner dengan rumus persamaan (2.4).

$$g(x,y) = \begin{cases} 1 \; if \; f(x,y) \geq T \\ 0 \; if \; f(x,y) < T \end{cases} \quad \ldots (2.4)$$

Keterangan :

g(x,y) = Hasil binerisasi dalam nilai 0 atau 1 tiap ordinat (x,y)
f(x,y) = Nilai keabuan dari ordinat (x,y)
T = Nilai ambang (*Thresholding*)

Dengan demikian data yang diujikan bisa diteliti dengan mudah. Karena citra dalam bentuk 2 dimensi dengn nilai 0 atau 1 di tiap pixelnya.

2.3.6. *Filtering Noise*

Tahap ini ditujukan untuk memperbaiki kualitas gambar yang diteliti. *Noise* dalam penelitian ini adalah titik-titik piksel yang tidak diinginkan yang ikut tertangkap kamera saat pengambilan citra. Untuk menghindari kesalahan penghitungan dalam penelitian, maka *noise* perlu dihilangkan. Dalam penelitian ini perbaikan citra dilakukan dengan algoritma mengisi lubang atau titik kosong yang merupakan biner 0 diantara area biner 1 yang berdekatan dengan biner 1 tetangganya dan algoritma dilasi yang ditujukan untuk menambahkan nilai piksel ke titik tertentu pada area yang lebih luas.

2.3.7. Pemotongan (*Cropping*)

Tahap ini merupakan proses pemotongan area yang tidak diinginkan untuk mendapatkan hasil yang presisi menyesuaikan ukuran objek yang diteliti. Dalam hal ini pemotongan mengikuti bentuk tangan pengguna. Untuk setiap luas area yang dengan biner 1 yang saling terhubung dan terputus saat bertemu biner 0, maka area tersebut dianggap area mandiri yang mendapatkan label. Kemudian cek area lain diluar area terlabeli dan lakukan hal sama (pelabelan) jika menemukan luas area

biner 1 yang saling terhubung hingga semua area mandiri terlabeli. Rumus pemotongan area yang seluas objek yang diteliti berarti melakukan pemotongan berdasarkan ordinat (x,y) pojok kiri atas serta selebar (width) dan tinggi (length) area area mandiri dengan persamaan (2.5).

$$\text{width} = x_o - x$$
$$\text{length} = y_o - y$$
$$\text{Crop} = [x\ y\ \text{width}\ \text{length}] \quad \ldots(2.5)$$

Keterangan :

x : ordinat x kiri terluar area mandiri/ pojok kiri
y : ordinat y atas terluar area mandiri/ pojok atas
x_o : ordinat x kanan terluar lurus dari variabel x dalam satu area mandiri
y_o : ordinat y bawah terluar lurus dari variabel y dalam satu area mandiri
width : lebar area mandiri
length : tinggi area mandiri

2.4. *Simple multi attribute rating technique* (SMART)

Metode ini dibuat oleh Edward pada tahun 1977. Merupakan metode pembobotan yang digunakan untuk membantu mengambil keputusan berdasarkan pada teori bahwa setiap alternatif terdiri dari sejumlah kriteria yang memiliki nilai – nilai dan setiap kriteria memiliki bobot yang menggambarkan seberapa penting ia dibandingkan dengan kriteria lain. Pembobotan ini digunakan untuk menilai setiap alternatif agar diperoleh alternatif terbaik [PRA-15]. Adapun langkah-langkah yang harus dikerjakan sebagai berikut :

a. Menentukan kriteria pembobotan

Langkah pertama, menentukan beberapa kriteria yang digunakan sebagai ketentuan yang harus dipenuhi yang mengarahkan ke kesimpulan. Adapaun pertimbangan yang diperlukan untuk menentukan kriteria bisa diambil berdasarkan analisa masalah yang ada. Kemudian diambil beberapa faktor dianggap berpengaruh pada hasil akhir. Misal dalam alphabet bahasa isyarat amerika, jika diamati secara detil akan terlihat bahwa beberapa alphabet yang memiliki bentuk dasar tangan jari menggenggam, jari mengarah ke atas dan ada jarinya yang condong ke samping bahkan ke bawah. Seperti alphabet A,E,M,N,O,S,T yang memiliki dasar bentuk tangan jari menggenggam. Jadi jari menggenggam bisa dipilih menjadi salah satu kriteria atau bisa dilambangkan dengan variabel K1.

b. Menentukan bobot kriteria (W)

Nilaia bobot kriteria merupakan rentang nilai yang menunjukkan skala prioritas dengan syarat tertentu. Mungkin saja dari yang paling urgensi sampai yang bisa ditunda, atau dalam alphabet bahasa isyarat, bobot kriterianya dihitung berdasar banyaknya alphabet yang memiliki kesamaan bentuk dasar tangan. Dalam kasus tersebut, mungkin saja diberikan nilai 0,23 untuk kriteria yang memiliki bentuk dasar tangan menggenggam. Karena jumlah anggota dari bentuk tangan menggengam ada 7, sedangkan jumlah alphabet dalam bahasa isyarat ada 26. Angka tersebut diperoleh dengan persamaan (2.6).

$$w = \frac{N\ anggota\ kriteria}{\sum alphabet\ bahasa\ isyarat} \quad \ldots(2.6)$$

Keterangan :
w = Bobot Kriteria

c. Menentukan bobot relatif/ normalisasi (R)

Dalam menentukan nilai normalisasi (R) untuk masing-masing kriteria diperoleh dengan pendekatan hasil dari bobot kriteria yang berbanding terbalik dengan jumlah seluruh bobot kriteria seperti pada persamaan (2.7).

$$R = \frac{W}{\sum W} \quad \ldots(2.7)$$

Keterangan :
R = Bobot Relatif/ Normalisasi
w = Bobot Kriteria

d. Menentukan nilai utilitas (U)

Untuk menentukan nilai utilitas (U) skala 0 sampai 100 yang merupakan nilai pencapaian masing-masing alternatif terhadap suatu kriteria, yang harus dinilai berdasarkan syarat tertentu [SIT-15].

e. Menghitung nilai akhir (NA)

Tahap akhir dari pembobotan *simple multi attribute rating technique* (SMART) adalah dengan menentukan nilai akhir (NA) tiap alternatif, kemudian memilih NA terbesar yang menjadi alternatif yang dianggap paling mungkin diambil menjadi kesimpulan menggunakan persamaan (2.8).

$$NA = \sum_{i=1}^{k} R_i \; x \; U_i \qquad \text{...(2.8)}$$

Keterangan :
NA : Nilai Akhir
R_i : Bobot Relatif/ Normalisasi Kriteria ke-i
U_i : Nilai Utilitas Alternatif terhadap Kriteria ke-i
k : Jumlah Kriteria yang ada

2.5. *K-NEAREST NEIGBOR* (KNN)

Merupakan metode yang sangat umum digunakan dalam proses klasifikasi pengenalan pola karena memiliki rumus perhitungan yang sangat sederhana. Metode yang menghitung jarak kedekatan antara data uji dengan data latihnya. Jarak tersebut dihitung menggunakan algoritma Euclidean seperti pada persamaan (2.9).

$$D(a,b) = \sqrt{\sum_{i=1}^{1100} (a_i - b_i)^2} \qquad \text{... (2.9)}$$

Keterangan :
D(a,b) = Jarak Euclidean
a_i = Fitur Data Uji ke-i
b_i = Fitur Data Latih ke-i digunakan

Mengacu pada rumus persamaan 2.9, nilai D(a,b) tiap fitur akan diranking dari nilai terkecil yang berada diurutan paling atas hingga terbesar di paling bawah. Proses klasifikasinya ditentukan berdasarkan jumlah tetangga/ nilai K yang didapatkan dari pelatihan (*training*). Banyaknya nilai K mempengaruhi hasil klasifikasinya. Untuk K bernilai 1, berarti klass yang terpilih adalah klass dengan nilai D(a,b) terkecil [LIN-14]. Jika nilai K lebih dari 1, maka klass yang paling banyak muncul menjadi klass terpilih. Misalkan K=3, jika klass dengan D(a,b) terdekatnya 2 klass A dan 1 klass B, maka klass terpilihnya adalah klass B. Namun untuk kasus dimana ketiga klass terdekatnya berbeda semua, misal 1 klass A, 1 klass B dan 1 klass C, maka yang terpilih adalah D(a,b) terkecil dari ketiga klass tersebut [LIU-14].

2.6. TUNA RUNGU DAN TUNA WICARA

Secara fisik, anak tunarungu tidak berbeda dengan anak yang dapat mendengar pada umumnya, sebab orang akan mengetahui bahwa anak menyandang ketunarunguan pada saat berbicara, mereka berbicara tanpa suara atau dengan suara yang kurang atau tidak jelas artikulasinya, atau bahkan tidak berbicara sama

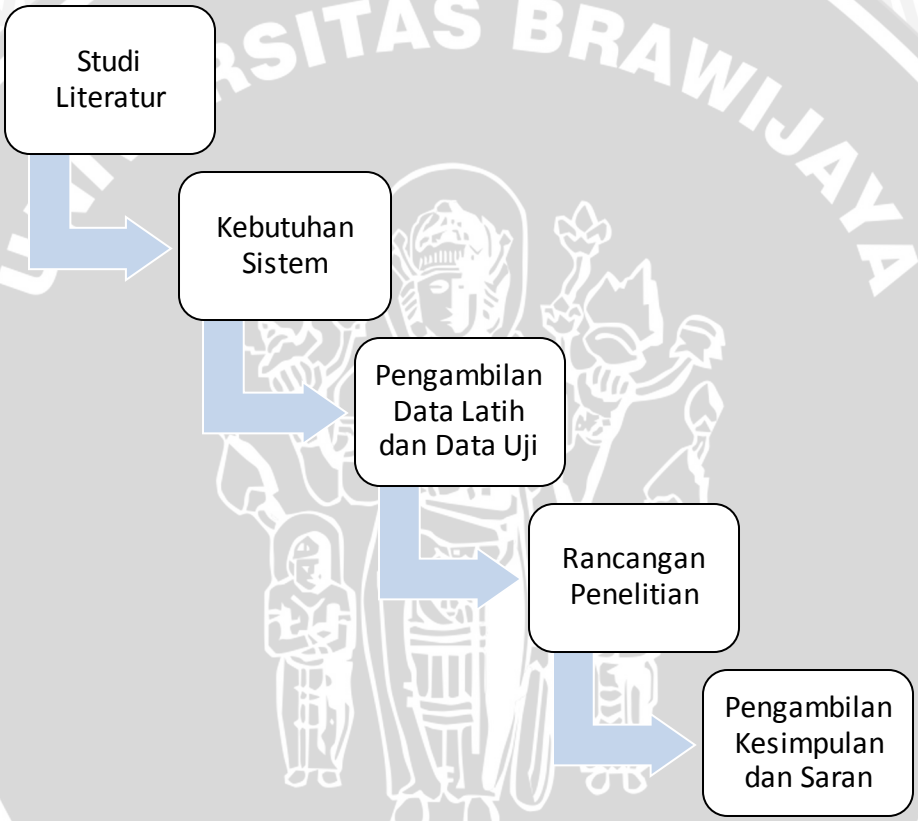
sekali, mereka hanya berisyarat. Anak tunarungu adalah anak yang mengalami gangguan pendengaran dan percakapan dengan derajat pendengaran yang berfariasi antara 27dB –40 dB dikatakan sangat ringan 41 dB – 55 dB dikatakan Ringan, 56 dB – 70 dB dikatakan Sedang, 71 dB – 90 dB dikatakan Berat, dan 91 ke atas dikatakan Tuli.

Dari ketidakmampuan anak tunarungu dalam berbicara, muncul pendapat umum yang berkembang, bahwa anak tunarungu ialah anak yang hanya tidak mampu mendengar sehingga tidak dapat berkomunikasi secara lisan dengan orang dengar. Karena pendapat itulah ketunarunguan dianggap ketunaan yang paling ringan dan kurang mengundang simpati, dibanding dengan ketunaan yang berat dan dapat mengakibatkan keterasingan dalam kehidupan sehari-hari. Batasan ketunarunguan tidak saja terbatas pada yang kehilangan pendengaran sangat berat, melainkan mencakup seluruh tingkat kehilangan pendengaran dari tingkat ringan, sedang, berat sampai sangat berat [SET-15].

Berdasarkan paparan di atas bisa disimpulkan bahwa ada keterkaitan antara tuna rungu dan tuna wicara. Bahwa orang dengan kelainan pendengaran yang parah sejak lahir tidak bisa mendengar apapun, sehingga tidak bisa mendapatkan pembelajaran apapun yang pada akhirnya tidak bisa mengembalikan pembicaraan apapun. Berbeda dengan tuna rungu karena kecelakaan, dimana mereka sebelumnya sudah belajar untuk berkomunikasi dengan berbicara, sehingga mereka hanya tuna rungu saja namun tetap bisa berbicara.

BAB 3 METODE PENELITIAN

Dalam bab ini menjelaskan tentang metode atau langkah-langkah penelitian dan rancangan yang digunakan dalam pembuatan simulasi pengenalan alphabet bahasa isyarat menggunakan pembobotan *Simple Multi Attribute Rating Technique* (SMART) dan klasifikasi *K-Nearest Nehbor* (KNN) untuk membantu Tuna Rungu dan Tuna Wicara.



***Flowchart* 3.1. Langkah-langkah Pengenalan Alphabet Bahasa Isyarat**

3.1. Studi Literatur

Adapun pembahasan dalam studi literatur meliputi pembobotan, pengolahan gambar dan pengklasifikasian. Untuk pembobotannya, sesuai yang dituliskan dalam kajian pustaka, penelitian ini menggunakan Simple Multi Attribute Rating Technique (SMART). Metode pembobotan yang sangat sederhana, dimana untuk menghasilkan rekomendasinya, harus melewati tahapan pengelompokan kriteria, menentukan nilai bobot masing-masing kriteria, menentukan nilai utilitas dan nilai akhir. Untuk

pengolahan gambarnya, menggunakan deteksi warna tangan CrCb dan segmentasi. Sedangkan untuk klasifikasinya, menggunakan metode K-Nearest Nehbor (KNN) dan perhitungan Eucli-Distance untuk menentukan jarak kedekatan data latih dan data uji. Sebagai penelitian implementatif perancangan, penelitian ini menghasilkan gui sederhana yang bisa digunakan untuk menerjemahkan bahasa isyarat menjadi teks.

3.2. Kebutuhan Sistem

Beberapa permasalahan terkait efektifitas kerja program sering terjadi karena lingkungan kerja program tidak mendukung penerapan suatu program yang baru. Sehingga kebutuhan minimal diperhatikan agar program yang dijalankan dengan efektif.

a. Perangkat Keras

Berikut kebutuhan minimal dari aspek perangkat keras yang harus dipenuhi untuk menjalankan program yang dihasilkan penelitian ini.

- Prosesor : Intel Pentium(R) Dual-Core 2,1 Ghz
- RAM : 2048 MB
- Memori : 500 GB
- Kamera : Logitech HD 720p

b. Perangkat Lunak

Adapun spesifikasi perangkat lunak yang digunakan untuk mengimplementasikan program hasil penelitian ini adalah sebagai berikut.

- Sistem Operasi : Windows 7 Home Premium
- Aplikasi *Compiler* : Matlab 2013a
- Bahasa Pemrograman : Matlab Language

3.3. Pengambilan Data Latih dan Uji

Data latih yang digunakan dalam penelitian ini adalah citra tangan dengan kondisi telapak tangan lebih pendek daripada jari – warna kulit cerah dan telapak tangan lebih panjang daripada jari – warna kulit gelap yang diambil secara manual menggunakan program yang dibuat peneliti sendiri. Sehingga citra yang dihasilkan sangat presisi atau fokus hanya pada tangan saja. Sedangkan untuk data ujinya menggunakan citra tangan pengguna langsung yang ditangkap menggunakan program yang sama dengan yang digunakan untuk menangkap citra tangan data latih. Jumlah keseluruhan data latih yang disiapkan dalam penelitian ada 5 data untuk tiap kategori klasifikasi. Ada 26 kategori alphabet yang disiapkan yang merujuk pada jumlah alphabet internasional, maka ada $26 \times 5 \times 2$ data latih yang disiapkan, sehingga jumlahnya ada 260 data latih.

3.4. Rancangan Penelitian

Untuk membangun purwarupa perangkat lunak, diperlukan beberapa pendekatan utama yang perlu diperhatikan seperti pada diagram blok 3.1 berikut.

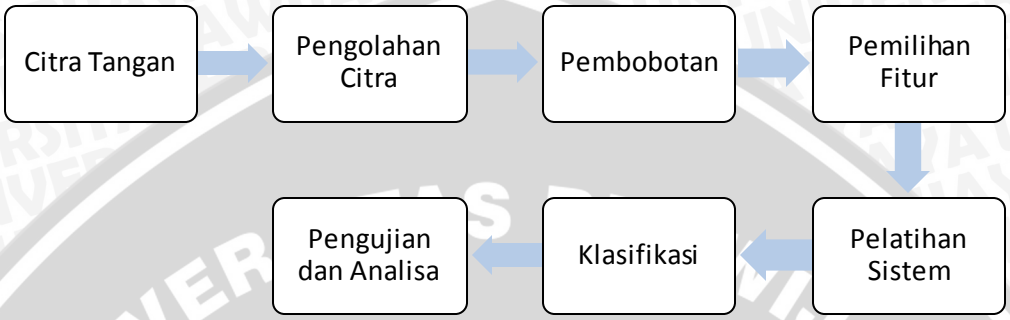


Diagram Blok 3.1. Rancangan Penelitian

Rancangan penelitian ini dimulai dengan memasukkan citra tangan yang akan diujikan dengan menangkap langsung citra tangan dari kamera yang diarahkan ke kotak GUI (*Graphical User Interface*), kemudian diolah menggunakan algoritma deteksi warna kulit dan segmentasi yang menghasilkan citra biner tangan. Selanjutnya dilakukan pembobotan untuk mengarahkan citra tangan yang diuji ke dalam grup tertentu yang berisikan kumpulan alphabet yang digunakan sebagai data latih. Setelah masuk ke salah satu grup, citra biner akan diambil nilai fiturnya berdasarkan jarak titik tengah ke tepian. Pelatihan sistem berisikan pemanggilan kumpulan data latih alphabet berdasarkan grup rekomendasi yang ditunjuk pembobotan. Karena data uji dan data latih sudah siap, maka tahap selanjutnya melakukan klasifikasi data uji menggunakan KNN dan menampilkan alphabet hasil klasifikasi ke GUI. Tahap terakhir yaitu melakukan pengujian untuk menunjukkan akurasi metode SMART KNN untuk mengenali alphabet bahasa isyarat. Kemudian dianalisa untuk menunjukkan faktor yang mempengaruhi keakuratan hasil klasifikasi.

3.5. Pengambilan Kesimpulan dan Saran

Kesimpulan dalam penelitian ini meliputi hasil pembahasan evaluasi dan analisa sistem dalam menyelesaikan rumusan masalah. Ditutup dengan saran yang berisi hal-hal yang mungkin bisa ditambahkan untuk pengembangan topik skripsi selanjutnya.

BAB 4 PERSYARATAN

Dalam bab ini menjelaskan persyaratan minimal yang harus dipenuhi untuk perancangan hingga implementasi. Dengan harapan perancangan dan implementasi program purwarupa ini bisa berjalan dengan baik.

4.1. Analisa Permasalahan

Seperti yang dijelaskan kajian pustaka pada BAB 2, dari beberapa penelitian yang pernah dilakukan memiliki keterbatasan yang perlu ditingkatkan. Diantaranya penggunaan citra yang sudah disiapkan sebelumnya untuk dijadikan data latih sekaligus beberapa sebagai data uji dan penggunaan sarung tangan untuk membedakan tangan dan latar belakangnya. Sehingga diperlukan sebuah pendekatan baru yang bisa digunakan untuk meningkatkan performa pemrosesan data agar lebih optimal, data uji lebih fleksibel dan akurasi lebih dari 70%.

Oleh karena itu dalam rumusan masalah dituliskan bahwa masalah yang diangkat adalah pengenalan pola tangan, klasifikasi menggunakan metode SMART KNN dan perancangan simulasi simulasi penerjemah alphabet bahasa isyarat ke dalam bentuk teks. Selain itu, dalam penelitian ini data uji yang digunakan adalah citra yang ditangkap saat ini juga saat pengguna ingin melihat hasil klasifikasinya. Tangan yang diteliti (citra hasil tangkapan kamera) tidak menggunakan sarung tangan untuk membedakan antara tangan dan latar belakangnya.

4.2. *Stakeholders*

Perangkat lunak purwarupa rupa ini selanjutnya bisa digunakan untuk mempermudah dalam mempelajari alphabet bahasa isyarat dan bisa juga digunakan untuk menerjemahkan bahasa isyarat yang ingin disampaikan penderita tuna rungu dan tuna wicara ke dalam bentuk teks dalam rangka berkomunikasi dengan orang normal. Secara umum perangkat ini bisa digunakan oleh pengguna yang memiliki tangan kanan dengan jari yang lengkap.

4.3. Analisa Kebutuhan Perangkat Lunak

Dalam membangun perangkat lunak, perlu diperhatikan beberapa hal yang menjadi acuan utama perancangan dan implementasi. Seperti kebutuhan fungsional, non fungsional, batasan sistem dan lingkungan operasi sistem.

a. Fungsional

Adapun kebutuhan fungsional dari perangkat lunak yang akan dibangun adalah sebagai berikut :

- Program harus bisa memasukkan citra tangan pengguna yang ditangkap pada saat itu juga untuk dijadikan data uji
- Pengguna harus bisa melihat alphabet hasil klasifikasi saat itu juga
- Pengguna bisa mengulang dari awal (*restart*) program

b. Non-Fungsional

Adapun kebutuhan non-fungsional dari perangkat lunak yang akan dibangun adalah sistem harus menyediakan GUI untuk mempermudah penggunaan.

c. Batasan Sistem

Program yang dibuat ini dalam bentuk purwarupa yang hanya bisa dijalankan menggunakan *compiler* matlab 2013. Sedangkan bahasa yang digunakan pun hanya bahasa pemrograman matlab itu sendiri. Semua citra yang ditangkap kamera maupun citra yang digunakan untuk data latih disimpan dalam direktori memori komputer.

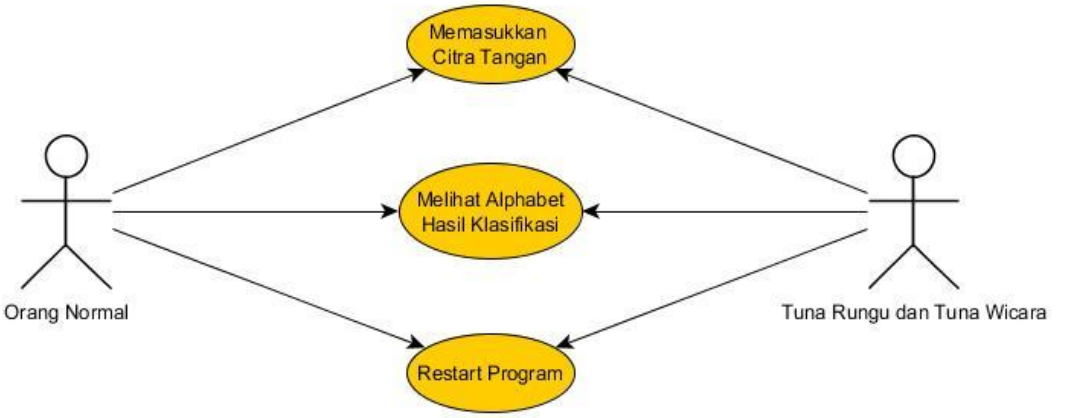
d. Lingkungan Operasi

Perangkat lunak yang sesuai harus dipersiapkan pengguna untuk menjalankan program yang dihasilkan penelitian ini agar program bisa berjalan dengan baik.

- Sistem Operasi : Windows 7
- Compiler : Matlab 2013
- Format Citra : .tiff

4.4. *Use Case Diagram*

Dalam penerapan tiap fungsi dari program ini digambar ke dalam *use case diagram* 4.1 berikut.



Gambar 4.1 Use Case Diagram Pengenalan Alphabet Bahasa Isyarat

Hanya terdapat 3 fungsi yang bisa dijalankan dari program purwarupa ini, yaitu memasukkan citra tangan, melihat alphabet hasil klasifikasi dan menjalankan ulang (*restart*) program.

4.5. *Use Case Spesification*

Berikut beberapa skenario yang menjelaskan spesifikasi masing-masing fungsi pada *use case diagram* :

a. Skenario Memasukkan Citra Tangan

Penjelasan isi dari fungsi memasukkan citra tangan disajikan di dalam Tabel 4.1 berikut.

Tabel 4.1 Skenario Memasukkan Citra Tangan

Ringkasan	Pengguna mengarahkan tangan ke kotak "*Camera Preview*" yang ada pada GUI untuk menghasilkan citra tangan yang digunakan sebagai data uji
Prioritas	Utama
User	Orang Normal, Tuna Rungu dan Tuna Wicara
Skenario Keberhasilan	1. Pengguna mengarahkan tangan ke kotak "*Camera Preview*" yang ada pada GUI 2. Dalam waktu beberapa detik kamera akan otomatis menangkap citra tangan 3. Citra tangan disimpan di direktori penyimpanan

Tabel 4.1 Skenario Memasukkan Citra Tangan (Lanjutan)

Skenario Alternatif	1. Jika tangan yang akan diujikan tidak pas masuk ke kotak yang disiapkan, dalam waktu beberapa detik yang sudah diatur, maka tangan harus segera digeser menjauh atau mendekat, ke kanan atau kiri, ke atas atau ke bawah kamera hingga tangan pas di kotak yang disediakan

b. Skenario Melihat Alphabet Hasil Klasifikasi

Penjelasan isi dari fungsi melihat alphabet hasil klasifikasi disajikan di dalam Tabel 4.2 berikut.

Tabel 4.2 Skenario Melihat Alphabet Hasil Klasifikasi

Ringkasan	Pengguna bisa melihat hasil klasifikasi alphabet yang dihasilkan program pada kotak "Alphabet" dan "Teks" yang ada pada GUI
Prioritas	Utama
User	Orang Normal, Tuna Rungu dan Tuna Wicara
Skenario Keberhasilan	1. Citra RGB tangan diolah program hingga menghasilkan citra biner tangan 2. Pembobotan citra biner untuk menghasilkan grup rekomendasi 3. Klasifikasi citra biner berdasarkan alphabet yang sesuai 4. Menampilkan alphabet hasil klasifikasi pada kotak "Alphabet" dan "Teks" " yang ada pada GUI
Skenario Alternatif	1. Jika citra rgb yang gagal diolah dan tidak menghasilkan alphabet klasifikasi, program akan berhenti dengan sendirinya

c. Skenario Mengulang (*Restart*) Program

Tabel 4.3 Skenario Mengulang (*Restart*) Program

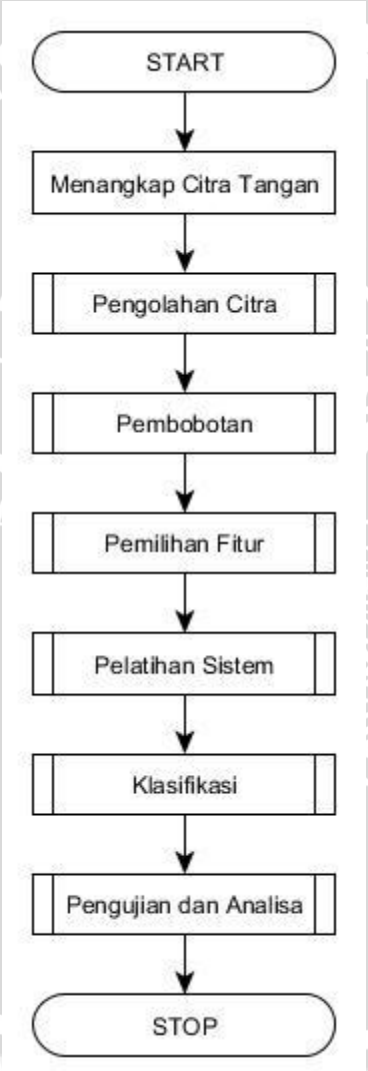
Ringkasan	Pengguna menekan tombol "*Restart*" yang sudah disediakan untuk memutar ulang program
Prioritas	Tambahan
User	Orang Normal, Tuna Rungu dan Tuna Wicara
Skenario Keberhasilan	1. Pengguna menekan tombol "*Restart*" 2. Program akan menghentikan proses pengolahan citra dan menutup GUI 3. Program memanggil ulang program
Skenario Alternatif	1. Jika tombol "*Restart*" gagal mengeksekusi perintah, maka tidak akan terjadi perubahan apapun pada program

BAB 5 PERANCANGAN DAN IMPLEMENTASI

Dalam bab ini menjelaskan penerapan tahapan perancangan yang telah dibuat sebelumnya. Menggunakan aplikasi compiler matlab 2013b untuk membuat GUI sederhana dan mengolah gambar hingga menampilkan hasil klasifikasi.

5.1. Perancangan

Untuk menjelaskan langkah-langkah yang akan dilakukan dalam perancangan ini mengikuti alur yang ada pada diagram alir 5.1 berikut.

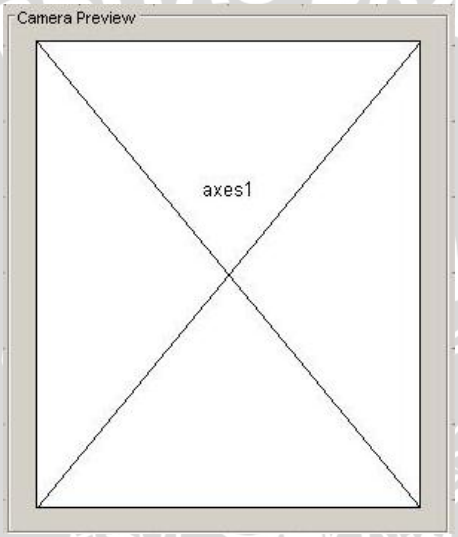


***Flowchart* 5.1 Alur Perancangan Sistem**

Berdasarkan flowchart 5.1, alur perancangan sistem ini dimulai dengan proses menangkap citra, kemudian dilanjutkan dengan subroutine pengolahan citra, pembobotan, pemilihan fitur, pelatihan sistem, klasfikasi, pengujian dan analisa.

5.1.1. Proses Menangkap Citra

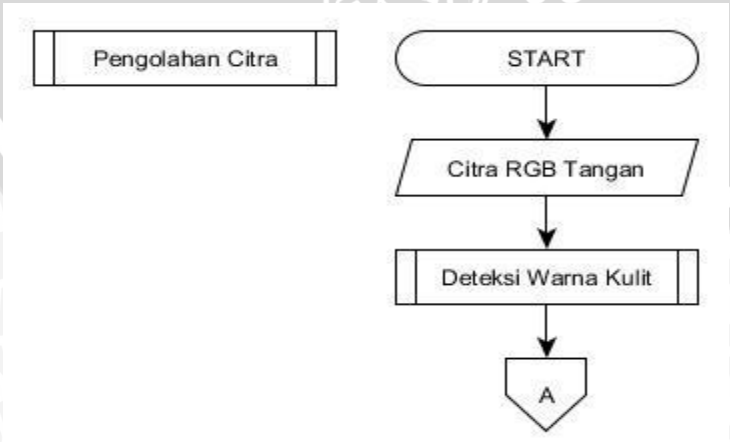
Langkah awal yang harus dilakukan untuk memulai penelitian ini dengan menangkap citra tangan yang akan diteliti. Yaitu dengan mengarahkan tangan ke kotak antar muka pengguna yang sudah disediakan, yang diberi nama kotak *Camera Preview*. Ukuran gambar yang dihasilkan yaitu citra RGB dengan resolusi 300×400.

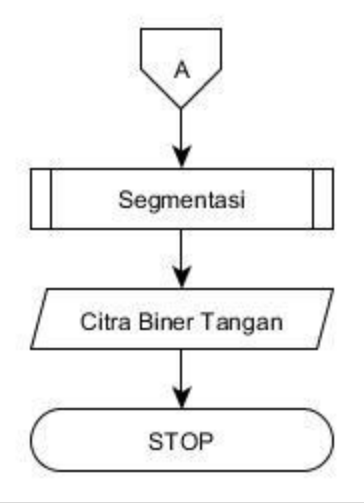


Gambar 5.1 Rancangan GUI "*Camera Preview*"

5.1.2. Subroutine Pengolahan Citra

Tahapan ini menjelaskan langkah-langkah binerisasi citra RGB sesuai pada flowchart 5.2.



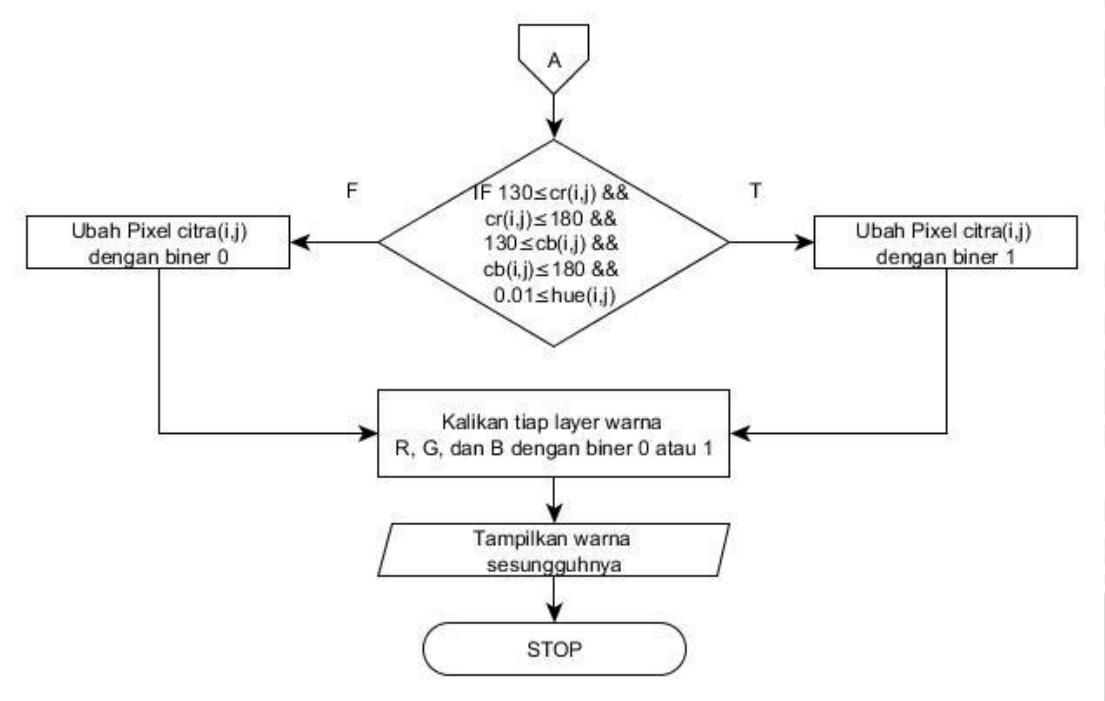


***Flowchart* 5.2 Pengolahan Citra**

Masukan tahapan ini adalah citra RGB yang ditangkap kamera. Citra RGB dihilangkan latarbelakangnya menggunakan *skin detection*. Kemudian hasil akhirnya citra dengan bentuk biner 0 dan 1.

- Deteksi Warna Kulit (*Skin Detection*)

  Tahapan ini menjelaskan langkah-langkah mendeteksi piksel yang sesuai dengan batasan warna kulit. Piksel diluar batasan yang sudah ditentukan akan diganti nilainya dengan 0. Sehingga latarbelakang dari tangan bisa hilang semua.

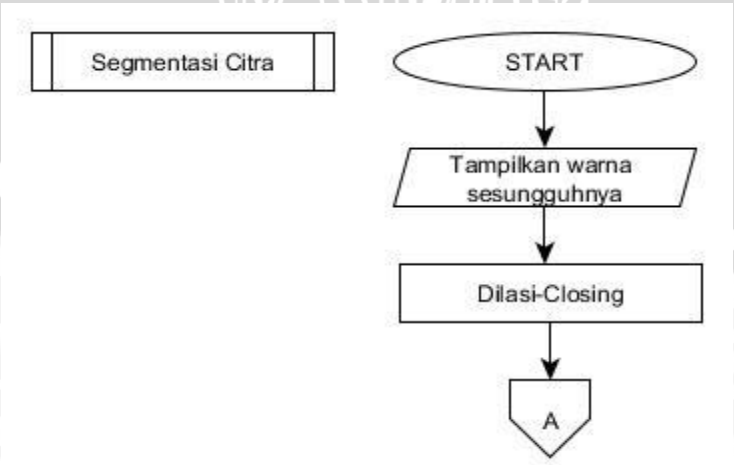


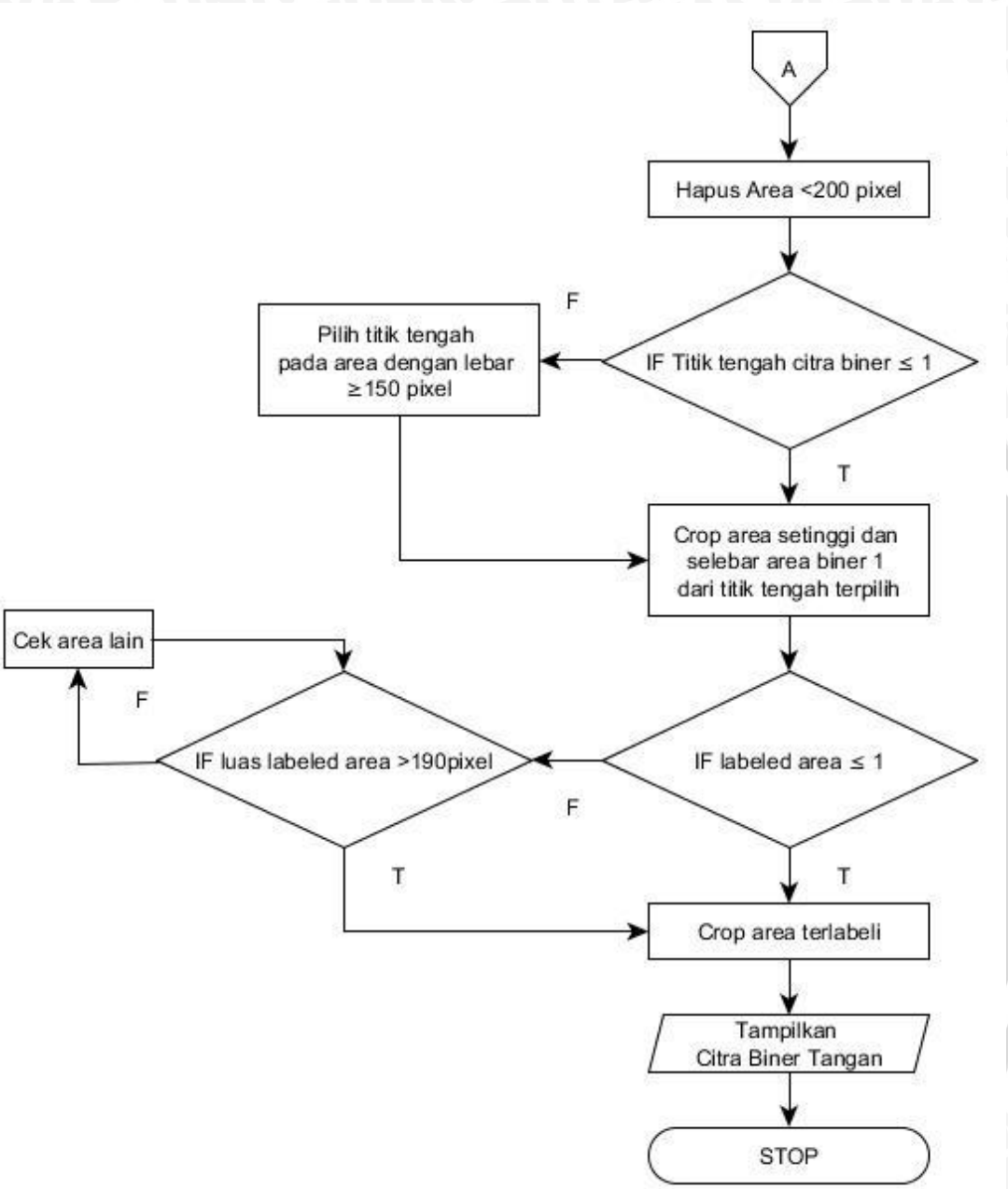
***Flowchart* 5.3 Deteksi Warna Kulit**

Untuk *subroutine* deteksi warna kulit, citra tangan diambil nilai hue, Cr (Chromium Red) dan Cb (Chromium Blue) nya yang nanti dijadikan batasan dalam proses ekstraksi warna tangan. Dengan ketentuan $130 \leq Cr \leq 180$, $130 \leq Cb \leq 190$, dan $0.01 \leq hue$ sebagai batasan ekstraksi warna tangannya.

- Segmentasi Citra

Tahapan ini menjelaskan citra yang sudah dibedakan antara tangan dengan latarbelakangnya diubah menjadi bentuk citra biner 0 dan 1.





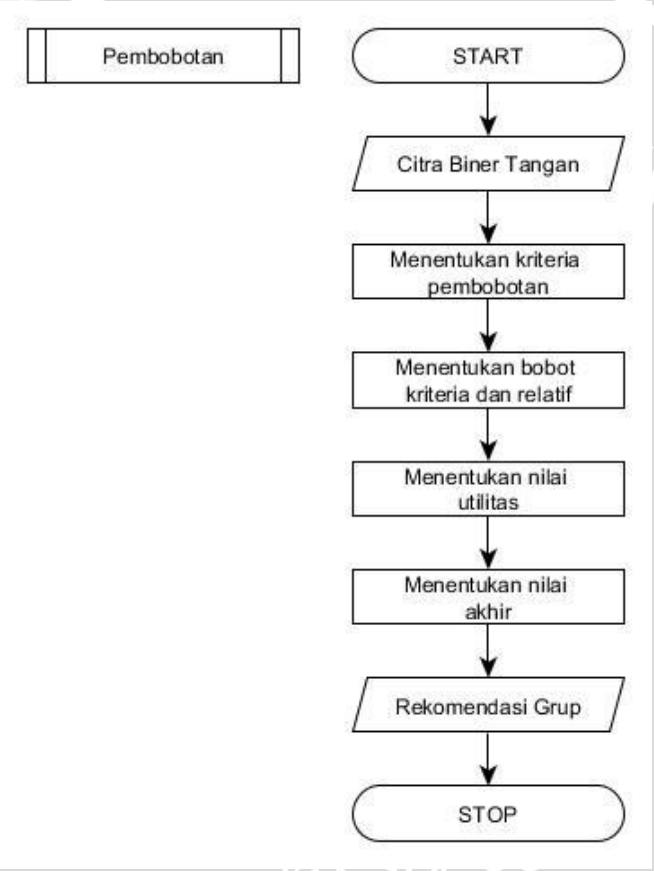
***Flowchart* 5.4 Segmentasi Citra**

Sedangkan untuk segmentasi, noise dihilangkan dengan metode pengisian lubang dan dilasi, menghapus area yang kurang dari 100 pixel, serta pemotongan citra yang menyesuaikan bentuk tangan. Di dalam matlab 2013a, penutupan lubang menggunakan imfill(gambar, ‘holes’) dan dilasi menggunkan bwmorph(gambar, 'dilate'), kemudian untuk area yang kurang dari 100 pixel

dihapus dengan bwareaopen. Sedangkan untuk memotong sesuai bentuk tangan, menggunakan regionprops dan dipilih FilledImage.

5.1.3. Subroutine Pembobotan

Tahapan ini menjelaskan langkah-langkah data uji yang berbentuk citra biner diolah menggunakan pembobotan SMART sehingga menghasilkan rekomendasi pembobotan yang sesuai nilai akhirnya.



***Flowchart* 5.5 Subroutine Pembobotan**

Metode yang digunakan dalam penelitian ini adalah pembobotan *Simple Multi Attribute Rating Technique* (SMART). Seperti yang dijelaskan dalam kajian pustaka, adapun langkah-langkah hingga menghasilkan rekomendasi pembobotan yaitu menentukan kriteria, bobot masing-masing kriteria, nilai utilitas dan nilai akhir.

- Kriteria Pembobotan

Pemiliihan kriteria pembobotan didasarkan pada pola bentuk dasar masing-masing alphabet bahasa isyarat seperti yang ditunjukkan dalam Tabel 5.1.

Tabel 5.1. Kriteria pembobotan

Kriteria	Bentuk Dasar Tangan	Alphabet	Jumlah Anggota Kriteria (N)
K1	Jari menggenggam	A,E,M,N,O,S,T	7
K2	Jari mengarah ke atas	B,C,D,F,I,K,L,R, U,V,W,X,Y	13
K3	Jari condong kesamping	G,H,J,P,Q,Z	6

Anggota kriteria 1 yaitu alphabet dengan bentuk dasar menggenggam (A,E,M,N,O,S,T). Anggota kriteria 2 yaitu alphabet dengan bentuk dasar jari tegap mengarah ke atas (B,C,D,F,I,K,L,R,U,V,W,X,Y). Sedangkan untuk kriteria 3 yaitu alphabet berbentuk dasar jari mengarah condong kesamping (G,H,J,P,Q,Z).

- Bobot Kriteria (W) dan Bobot Relatif (R)

Berikut Tabel bobot kriteria yang dihitung berdasarkan jumlah anggota kriteria dibagi jumlah anggota seluruh kriteria seperti pada persamaan (2.6) beserta bobot realtifnya yang dihitung menggunakan persamaan (2.7).

Tabel 5.2. Bobot kriteria (W) dan Bobot relative (R)

Kriteria (K)	Jumlah Anggota Kriteria (N)	Bobot Kriteria (W)	Bobot Relatif/ Normalisasi (R)
K1	7	$\frac{7}{26}$	0,27
K2	13	$\frac{13}{26}$	0,50
K3	6	$\frac{6}{26}$	0,23

- Nilai Utilitas (U_k)

Penentuan nilai utilitas untuk masing-masing kriteria merupakan sebaran nilai dari 0 sampai 100. Dalam penelitian ini dibuat hanya 2 kondisi, yaitu 0 dan 100. Tujuannya agar citra tangan yang diuji bisa diarahkan ke salah satu grup rekomendasi dengan pasti. Penentuan *range* ordinat tersebut dilakukan melalui beberapa percobaan terhadap citra data latih dengan membandingkan nilai *gx*

yang tersebar di baris 75 pada kolom 1 sampai 400 dan *gy* baris 1 sampai 300 pada kolom 25, hasil terhadap kondisi pertama seperti yang ditampilkan pada tabel 5.3.a, 5.3.b, 5.3.c.

Tabel 5.3. Nilai *gx* di baris 75 pada kolom 1 sampai 300 dan *gy* di baris 1 sampai 400 pada kolom 25

(a) Grup 1

Grup 1	gx	gy
A	212	248
A	242	249
A	258	285
A	244	273
A	241	260
S	280	341
S	263	178
S	280	285
S	286	262
S	287	262

(b) Grup 2

Grup 2	gx	gy
B	151	207
B	183	149
B	157	216
B	182	137
B	180	205
R	77	106
R	89	114
R	60	129
R	84	116
R	74	108

(c) Grup 3

Grup 3	gx	gy
H	209	53
H	220	54
H	197	50
H	249	64
H	216	54
Z	75	34
Z	66	26
Z	59	27
Z	65	35
Z	56	27

Selain itu percobaan untuk mendapatkan nilai *gx* yang tersebar di baris 100 pada kolom 1 sampai 400 dan *gy* baris 1 sampai 300 pada kolom 50, hasil terhadap kondisi pertama seperti yang ditampilkan pada tabel 5.4.a, 5.4.b, 5.4.c.

Tabel 5.4. Nilai *gx* di baris 100 pada kolom 1 sampai 300 dan *gy* di baris 1 sampai 400 pada kolom 50 kondisi pertama

(a) Grup 1

Grup 1	gx	gy
A	248	314
A	259	334
A	264	339
A	258	329
A	262	328
S	283	380
S	263	364
S	283	380
S	292	370
S	293	370

(b) Grup 2

Grup 2	gx	gy
B	204	295
B	242	305
B	203	288
B	230	314
B	240	286
R	98	152
R	102	240
R	84	176
R	100	236
R	95	170

(c) Grup 3

Grup 3	gx	gy
H	196	101
H	208	104
H	172	95
H	266	126
H	208	109
Z	97	42
Z	84	38
Z	70	33
Z	75	34
Z	92	33

Serta nilai *gx* yang tersebar di baris 125 pada kolom 1 sampai 400 dan *gy* baris 1 sampai 300 pada kolom 75, hasil terhadap kondisi pertama seperti yang ditampilkan pada tabel 5.5.a, 5.5.b, 5.5.c.

Tabel 5.5. Nilai *gx* di baris 125 pada kolom 1 sampai 300 dan *gy* di baris 1 sampai 400 pada kolom 75

(a) Grup 1

Grup 1	gx	gy
A	254	334
A	261	340
A	269	345
A	259	341
A	261	340
S	290	384
S	276	389
S	289	384
S	296	370
S	296	370

(b) Grup 2

Grup 2	gx	gy
B	216	311
B	247	316
B	224	313
B	227	325
B	240	309
R	105	190
R	120	254
R	96	177
R	115	244
R	107	249

(c) Grup 3

Grup 3	gx	gy
H	171	119
H	176	115
H	146	110
H	260	137
H	161	119
Z	140	41
Z	140	32
Z	148	41
Z	107	67
Z	137	36

Berdasarkan data yang diambil dari citra sampel data latih di atas, menunjukkan bahwa nilai *gx* yang tersebar di baris 75 pada kolom 1 sampai 400 dan *gy* baris 1 sampai 300 pada kolom 25 yang terbaik. Karena memiliki keteraturan sebaran nilai *gx* dan *gy* yang selanjutnya bisa dibuat menjadi persamaan (5.1) yang digunakan untuk mengarahkan citra data uji sesuai grup yang seharusnya.

$$U_{k3}=\begin{cases}100, & gy<90\\ 0, & other\end{cases}$$
$$U_{k2}=\begin{cases}100, & gy>90\ and\ gx<200\\ 0, & other\end{cases}$$
$$U_{k1}=\begin{cases}100, & gy>90\ and\ gx>200\\ 0, & other\end{cases} \quad \text{...(5.1)}$$

Jika nilai *gy* < 90, maka nilai utilitas kriteria 3 adalah 100 dan kriteria lainnya 0. Penentuan batasan *gy* yang tanpa mempedulikan nilai *gx* ini didasarkan pada pembulatan nilai *gy* maksimal dari Tabel 5.6 yang merupakan hasil perhitungan nilai *gx* dan *gy* dari masing-masing 5 alphabet anggota kriteria 3.

Tabel 5.6 Nilai gx dan gy alphabet anggota kriteria 3

Abjad	*gx*	*gy*
G	115	37
G	75	36
G	120	34
G	121	41
G	84	32
H	209	53
H	220	54
H	197	50
H	249	64
H	216	54
J	164	65
J	220	54
J	197	52
J	249	64
J	216	60
P	281	46
P	292	48
P	182	36
P	173	39
P	167	32
Q	183	56
Q	160	88
Q	209	59
Q	168	59
Q	188	60
Z	75	34
Z	66	26
Z	59	27
Z	65	35
Z	56	27
MIN	52	29
MAX	292	88

Jika nilai *gy* > 90 dan *gx* < 200, maka nilai kriteria 2 adalah 100 dan kriteria lainnya 0.

Tabel 5.7 Nilai *gx* dan *gy* alphabet anggota kriteria 2

Abjad	gx	gy
B	151	207
B	183	149
B	157	216
B	182	137
B	180	205
C	106	118
C	93	99
C	93	96
C	108	117
C	121	121
D	66	128
D	62	137
D	66	149
D	57	227
D	57	188
F	137	205
F	138	175
F	134	200
F	106	180
F	93	219
I	107	159
I	121	207
I	198	267
I	45	308
I	52	296
K	108	132
K	136	61
K	110	82
K	117	93
K	162	63
L	34	164
L	34	213
L	34	116
L	31	129

Tabel 5.7 Nilai *gx* dan *gy* alphabet anggota kriteria 2 (Lanjutan)

Abjad	gx	gy
L	31	154
R	77	106
R	89	114
R	60	129
R	84	116
R	74	108
U	117	95
U	132	104
U	116	99
U	116	118
U	122	143
V	100	119
V	112	156
V	122	158
V	104	92
V	149	152
W	172	136
W	198	119
W	189	137
W	165	179
W	185	150
X	66	114
X	66	102
X	66	102
X	66	102
X	87	134
Y	97	271
Y	97	270
Y	97	270
Y	97	270
Y	97	270
MIN	31	92
MAX	198	271

Sedangkan jika perhitungan nilai *gy* > 90 dan *gx* > 200, maka nilai kriteria 1 adalah 100 dan lainnya 0. Penentuan batasan *gx* dan *gy* ini didasarkan pembulatan nilai minimal dari Tabel 5.8 yang merupakan hasil perhitungan nilai *gx* dan *gy* dari semua alphabet anggota kriteria 1.

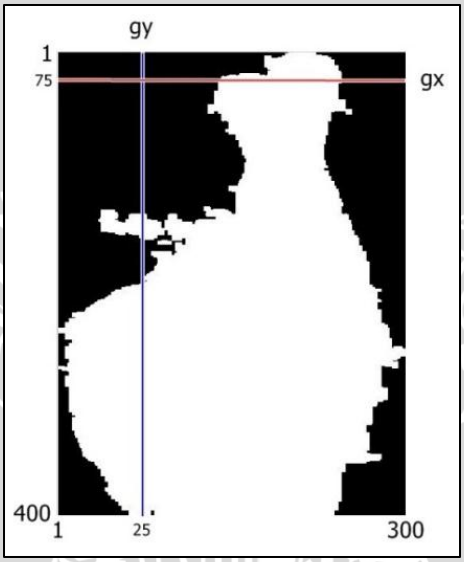
Tabel 5.8 Nilai gx dan gy alphabet anggota kriteria 1

Abjad	gx	gy
A	212	248
A	242	249
A	258	285
A	244	273
A	241	260
E	249	135
E	250	132
E	230	126
E	238	130
E	249	135
S	263	178
S	280	285
S	286	262
S	287	262
S	263	178
M	215	121
M	251	100
M	250	98
M	254	96
M	250	100
N	250	143
N	244	153
N	244	153
N	244	153
N	244	153
T	203	188
T	217	191
T	217	191
T	217	191
T	217	191
O	215	135
O	224	141

Tabel 5.8 Nilai gx dan gy alphabet anggota kriteria 1 (Lanjutan)

Abjad	gx	gy
O	224	141
O	224	141
O	224	141
MIN	203	100
MAX	280	288

Berikut ilustrasi gx dan gy pada alphabet dengan bentuk dasar tangan jari mengarah ke atas seperti pada Gambar 5.2.



Gambar 5.2 *gx* dan *gy* pada citra sampel

- Nilai Akhir (N_a)

Penghitungan nilai akhir didapatkan dengan rumus jumlah keseluruhan dari nilai utilitas dikali bobot kriterianya seperti pada rumus persamaan (5.2).

$$N_a = \sum_{i=1}^{3} R_i \; x \; U_{ki} \quad \text{...(5.2)}$$

Keterangan :

N_a : Nilai Akhir
R_i : Bobot Relatif/ Normalisasi Kriteria ke-i
U_i : Nilai Utilitas Alternatif terhadap Kriteria ke-i

Setelah mendapatkan nilai akhir, maka akan diberikan rekomendasi alternatif yang sesuai seperti yang tertera pada Tabel 5.6.

Tabel 5.9 Alternatif kesimpulan

No.	Alternatif Kesimpulan	Prediksi Nilai Akhir Alternatif (N_a) = $\sum$ (R x U_k)
1	G1	Karena Na G1=27, G2=0 dan G3=0.
2	G2	Karena Na G2=50, G1=0 dan G3=0.
3	G3	Karena Na G3=23, G2=0 dan G1=0.

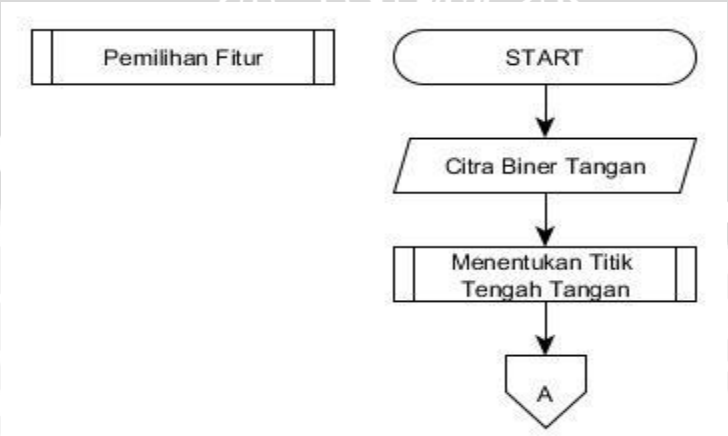
Berdasarkan Tabel 3.6 yang merupakan alternatif kesimpulan yang bisa diambil, maka citra yang diuji akan masuk ke dalam grup yang beranggotakan kumpulan alphabet tertentu sesuai persaaman (5.3).

$$\text{Citra} = \begin{cases} G1 & N_a = 27 \\ G2 & N_a = 50 \\ G3 & N_a = 23 \end{cases} \quad \text{...(5.3)}$$

Masing-masing alternatif merupakan grup kumpulan alphabet. Dimana anggota G1 adalah anggota K1, anggota G2 adalah anggota K2 dan anggota G3 adalah anggota K3. Dengan demikian citra yang diuji setelah melewati proses pembobotan akan diklasifikasi menggunakan data latih sesuai anggota grup terpilih. Diharapkan proses ini selain bisa menghemat waktu proses pengklasifikasia juga bisa meminimalkan salah klasifikasi.

5.1.4. Subroutine Pemilihan Fitur

Tahapan menjelaskan langkah-langkah pengambilan nilai fitur berdasarkan jarak titik tengah tangan dengan tepian tangan.

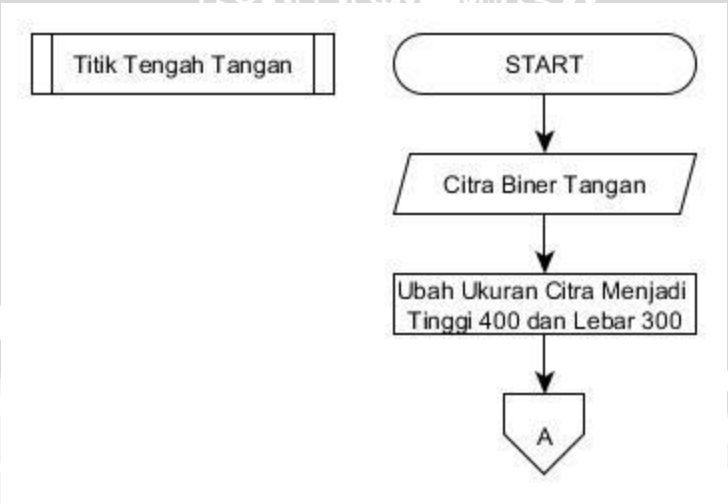


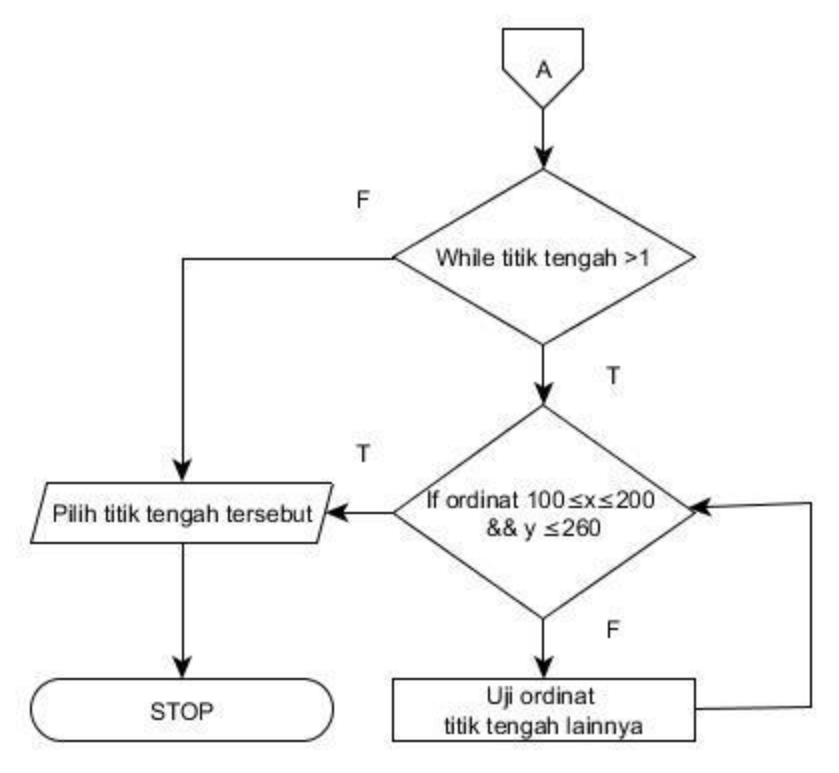
***Flowchart* 5.6 Subroutine Pememilihan Fitur**

Dalam proses ini terdapat 2 *subroutine* yang menjadi proses utama dalam pemilihan fitur. Pertama *subroutine* tentang menentukan tepian tangan dan kedua mengukur jarak titik tengah ke tepian tangan.

- Menentukan Titik Tengah Tangan

Tahapan ini menunjukkan langkah-langkah deteksi titik tengah tangan.





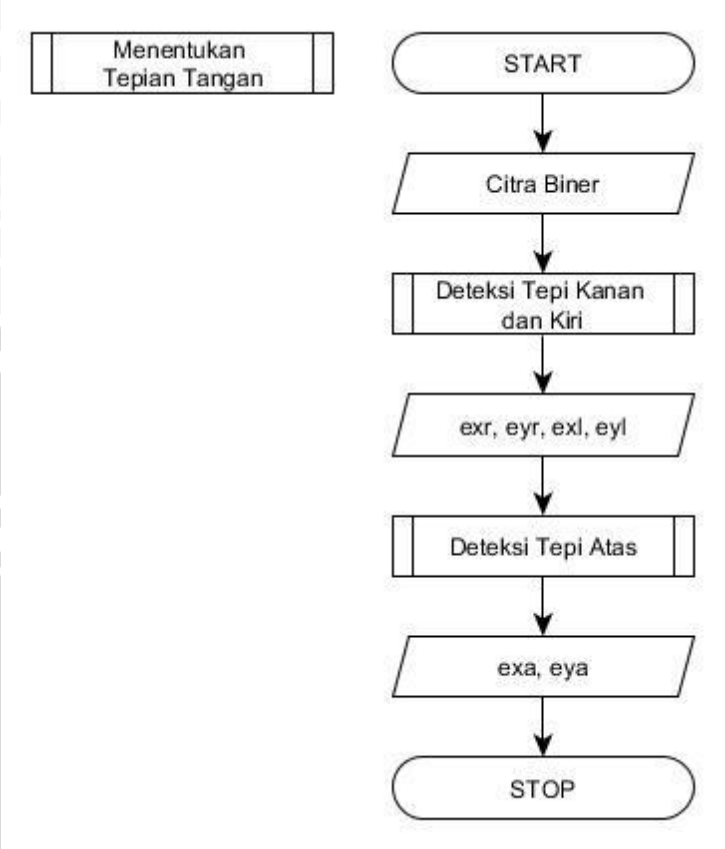
***Flowchart* 5.7 Subroutine Titik Tengah Tangan**

Untuk memudahkan proses pengukuran jarak titik tengah ke tepian tangan nanti, maka ukuran citra biner distandarkan menjadi tinggi 300 dan lebar 400 pixel (300x400 pixel). Kemudian gunakan regionprops(bw, centroid) untuk identifikasi titik tengah citra biner. Jika jumlah titik tengah yang terdeteksi hanya 1, maka langsung pilih titik tengah tersebut untuk digunakan dalam proses pengukuran. Jika sebaliknya, maka lakukan eliminasi menggunakan persamaan (5.4).

$$\text{Titik tengah} = \begin{cases} 100 \le x \le 200 \text{ dan } 140 \le y \le 260 & \text{, Pilih titik tengah} \\ \text{Lainnya} & \text{, Uji ordinat titik tengah lainnya} \end{cases} \quad \text{...(5.4)}$$

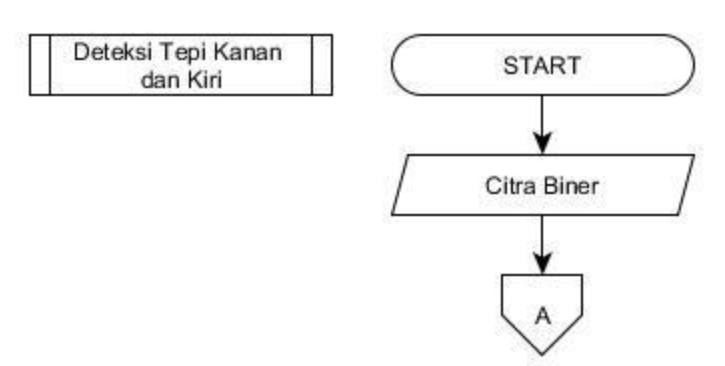
- Menentukan Tepian Tangan

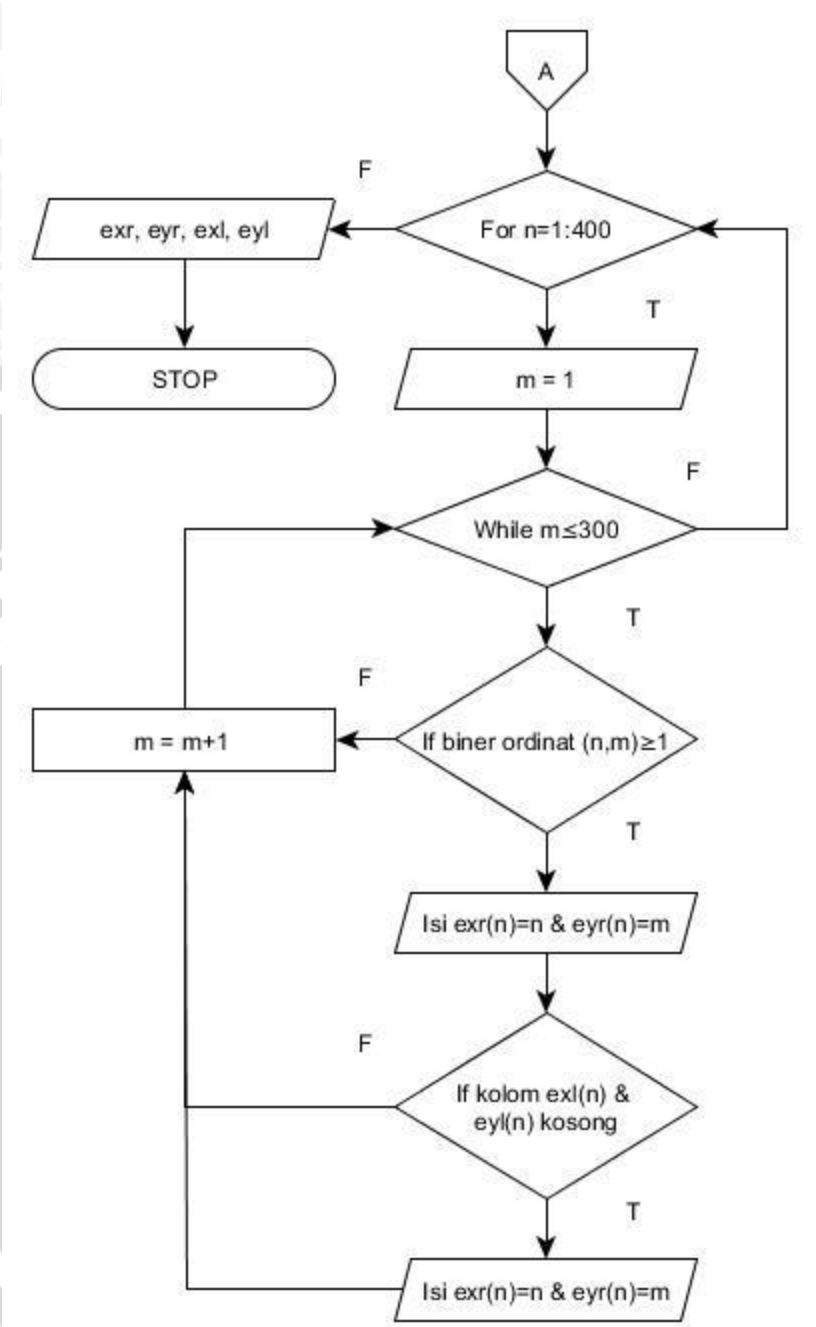
  Tahapan ini menunjukkan langkah-langkah scanning baik dari atas ke bawah maupun dari kiri ke kanan untuk mendapatkan koordinat tepian tangan.



***Flowchart* 5.8 *Subroutine* Menentukan Tepian Tangan**

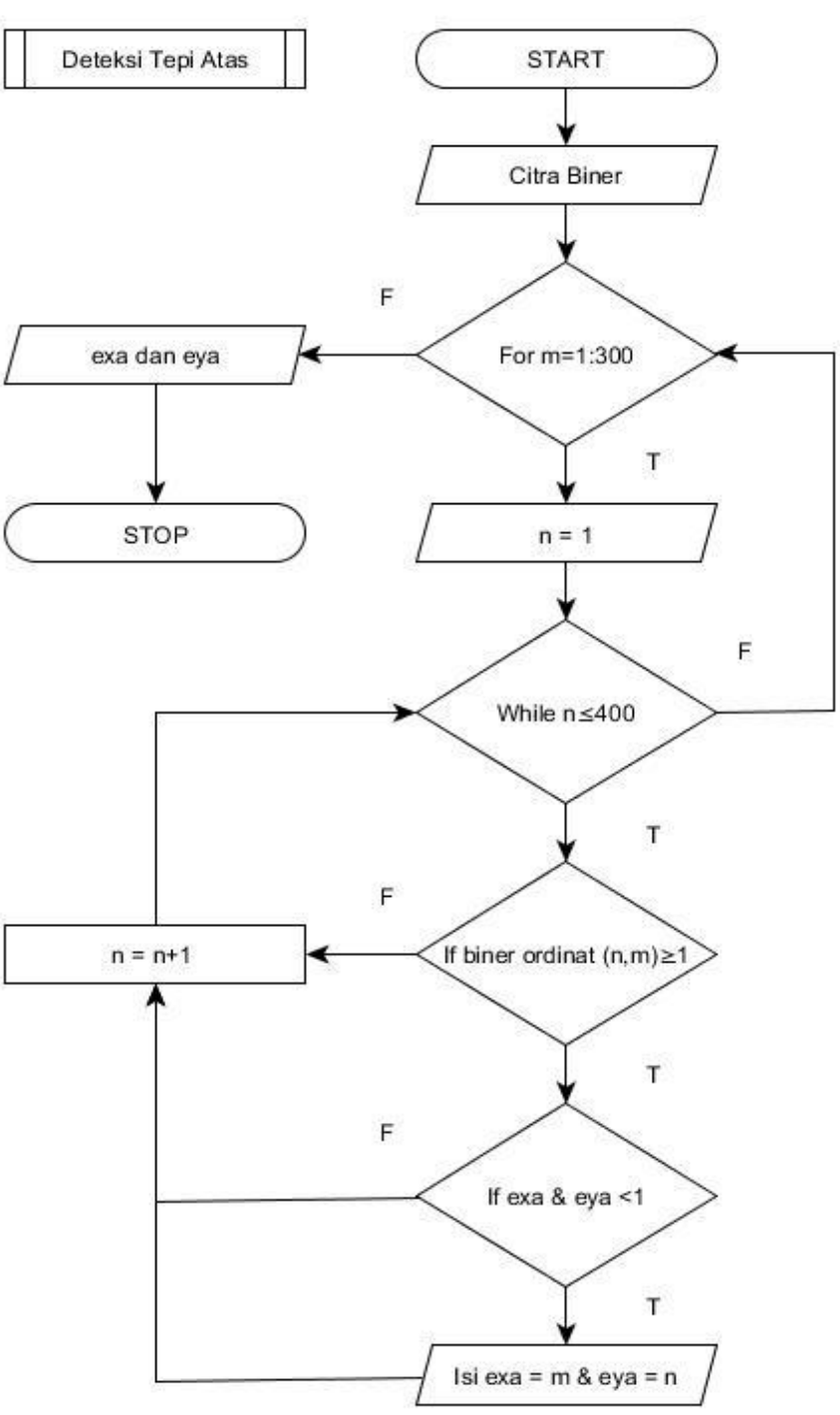
Di dalam flowchart 5.8 ditunjukkan subroutine untuk deteksi tepi kanan dan kiri tangan sebagai bagian untuk mendapatkan fitur klasifikasi. Tahapan penentuan tepi kiri dan kanan tangan ditunjukkan pada *flowchart* 5.9.





***Flowchart* 5.9 *Subroutine* Menentukan Tepi Kanan dan Kiri**

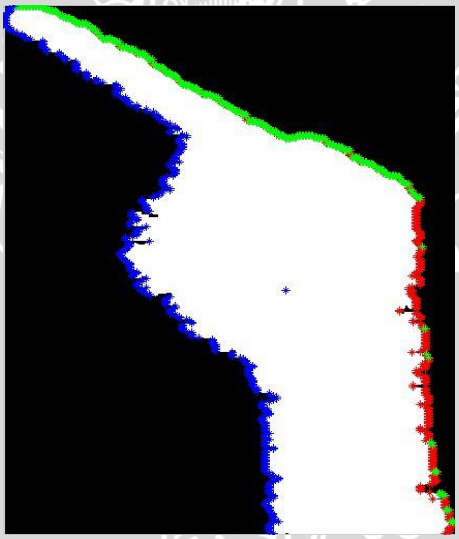
Di dalam flowchart 5.8 ditunjukkan subroutine untuk deteksi tepi atas tangan sebagai bagian untuk mendapatkan fitur klasifikasi. Tahapan penentuan atas tangan ditunjukkan pada *flowchart* 5.9.



***Flowchart* 5.10 *Subroutine* Menentukan Tepi Atas**

Dalam *flowchart* 5.10 menunjukkan *looping* for sebanyak 400 kali yang di dalamnya terdapat *looping* 300 kali. Hal ini dikarenakan menyesuaikan ukuran citra yang diteliti dengan tinggi 400 dan lebar 300. Jadi arah *scanning* binernya dari arah kiri ke kanan. Tujuannya untuk mencatat ordinat tepi kiri dan kanan dari citra biner tangan. Dimana terdapat 4 variabel yang digunakan untuk menyimpan masing-masing ordinat, yaitu variabel exr menyimpan ordinat x sisi kanan, exl menyimpan ordinat x sisi kiri, eyr menyimpan ordinat y sisi kanan dan eyl menyimpan ordinat y sisi kiri.

Selanjutnya dalam *flowchart* 3.11 terdapat *looping* untuk deteksi tepian sisi atas citra biner tangan. Proses *looping* kali ini melakukan *scanning* dari atas ke bawah. Dimulai dengan looping sebanyak 300 kali yang didalamnya terdapat looping sebanyak 400 kali. Saat deteksi biner 1 dari atas ke bawah, simpan ordinat tepian atasnya ke dalam variable exa dan eya. Berikut tampilan hasil deteksi tepian tangan beserta titik tengahnya :

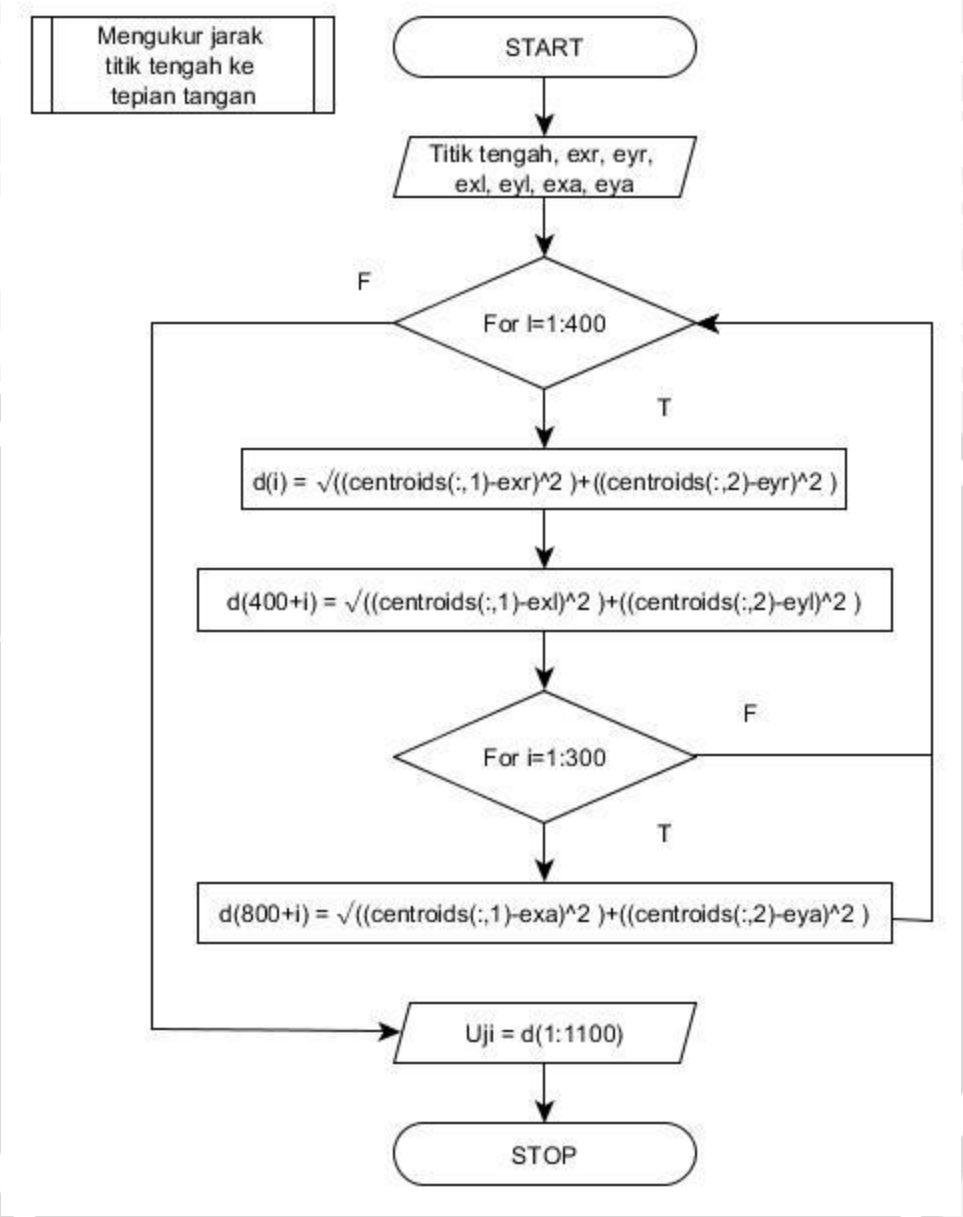


Gambar 5.3 Hasil deteksi tepian tangan dan titik tengah

Pada gambar di atas exr dan eyr ditunjukkan dengan warna merah, exa dan eya ditunjukkan dengan warna hijau, sedangkan exl dan eyl ditunjukkan dengan warna biru.

- Mengukur Jarak Titik Tengah ke Tepian Tangan

Setelah ditemukan titik tengah, tepian kiri, atas dan kanan tangan, selanjutnya mengambil nilai fitur untuk klasifikasi. Pengukuran dilakukan pada titik tengah ke seluruh tepian tangan.



***Flowchart* 5.11 *Subroutine* Mengukur Jarak Titik Tengah ke Tepian Tangan**

Seperti yang dijelaskan dalam deteksi tepian tangan. Bahwa sisi tangan yang di deteksi mulai dari sisi kiri, atas dan kanan. Untuk d ke tepian kiri disimpan ke dalam d(1:400), d ke tepian kanan disimpan ke dalam d(401:800), dan d ke tepian atas disimpan ke dalam d(501:1100). Maka ada 1100 fitur yang harus dihitung jarak titi tengah ke tepiannya yang kemudian semunya disimpan ke dalam variable d yang ditunjukkan pada persamaan (5.5).

d(1:400) $= \sqrt[2]{((centroids(:,1) - exr)^2) + ((centroids(:,2) - eyr)^2)}$

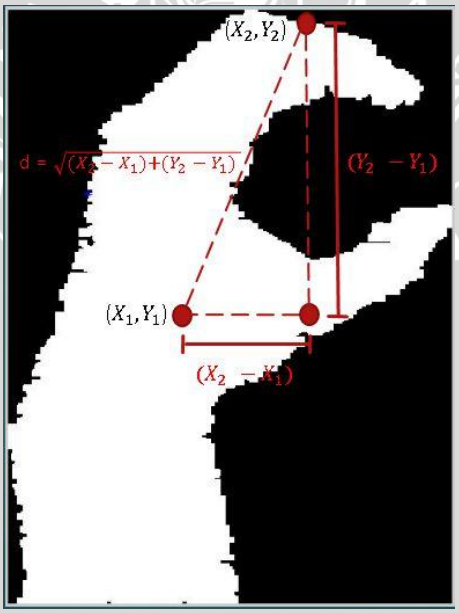
d(401:800) $= \sqrt[2]{((centroids(:,1) - exl)^2) + ((centroids(:,2) - eyl)^2)}$

d(801:1100) $= \sqrt[2]{((centroids(:,1) - exa)^2) + ((centroids(:,2) - eya)^2)}$ ...(5.5)

Keterangan :

d(1:400)	: d kolom 1 sampai 400	exr	: ordinat x sisi kanan
d(401:800)	: d kolom 401 sampai 800	exl	: ordinat x sisi kiri
d(801:1100)	: d kolom 801 sampai 1100	exa	: ordinat x sisi atas
$centroids(:,1)$	: ordinat x titik tengah	eyr	: ordinat y sisi kanan
$centroids(:,2)$	: ordinat y titik tengah	eyl	: ordinat y sisi kiri
		eya	: ordinat y sisi atas

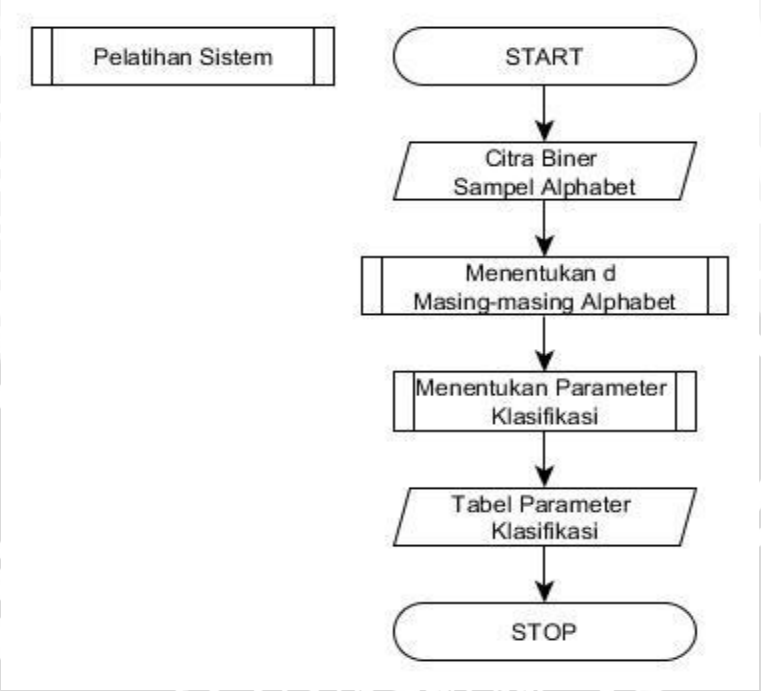
Untuk lebih jelasnya rumus tersebut diperoleh berdasar perhitungan pitagoras. Karena nilai d merupakan panjang sisi miring seperti pada Gambar 5.4.



Gambar 5.4 d Titik tengah ke salah satu titik tepian tangan

5.1.5. Subroutine Pelatihan Sistem

Penerapan metode KNN dimulai dengan pengambilan nilai fitur pada masing-masing citra yang digunakan sebagai data latih. Nilai fitur yang digunakan adalah jarak titik tengah ke tepian kiri, atas dan kanan tangan. Semua nilai tersebut dikumpulkan menjadi satu ke dalam database variable yang sudah disiapkan untuk selanjutnya dihitung ke dekatannya dengan nilai fitur data uji.

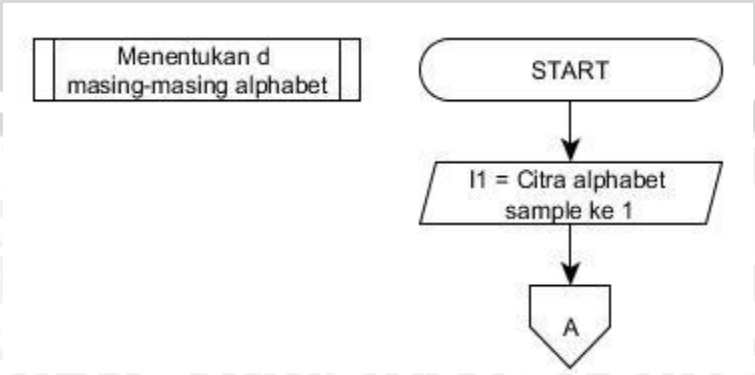


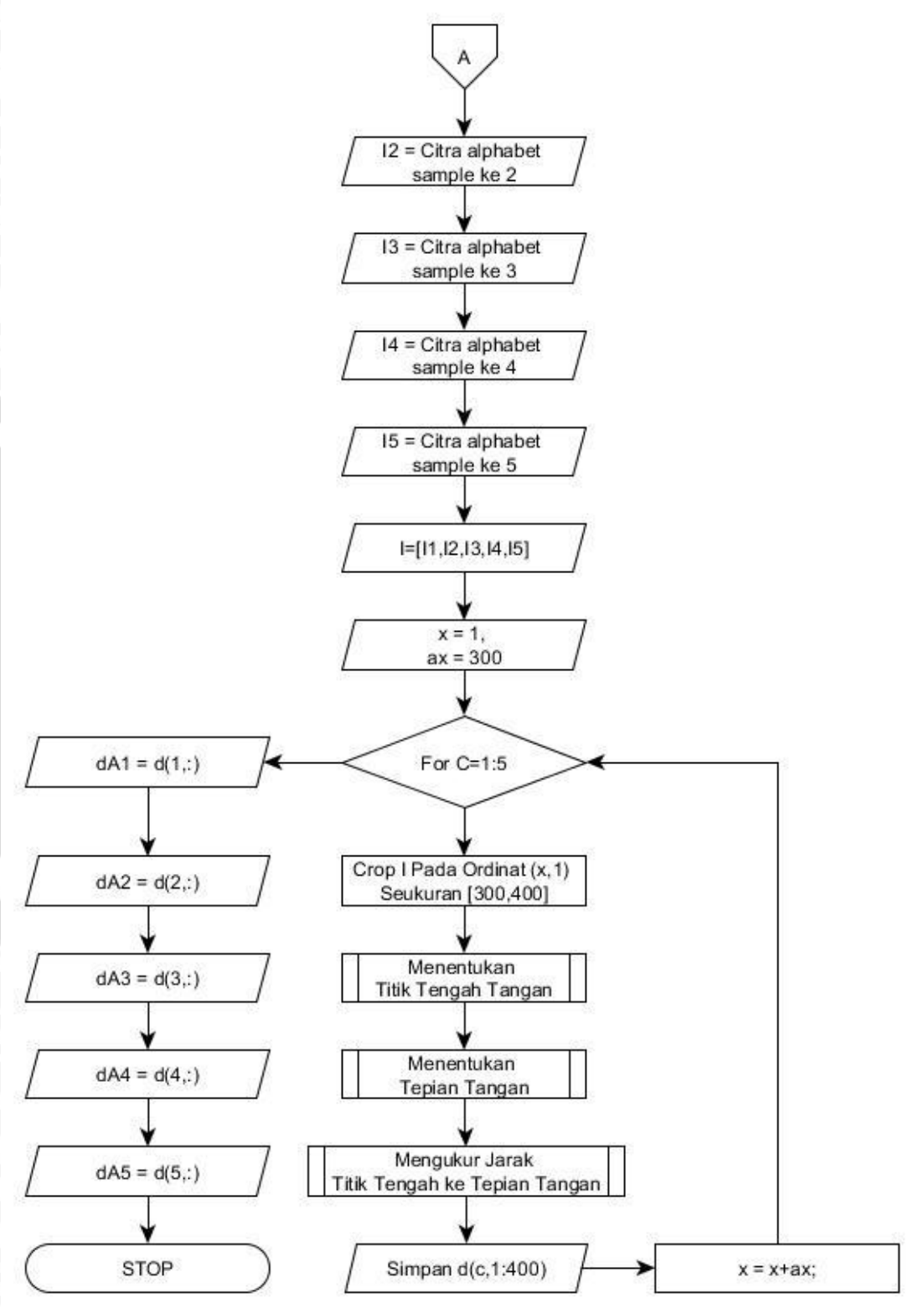
***Flowchart* 5.12 *Subroutine* Pelatihan Sistem**

Agar sistem bisa mengenali alphabet bahasa isyarat, maka sistem harus dilatih dengan memasukkan aturan-aturan yang mengarah pada grup yang sudah ditentukan. Dalam penelitian ini, pelatihan yang diberikan menggunakan metode pelatihan terawasi (*Supervised Learning*). Berarti data latih yang telah dikumpulkan sebelumnya, harus dihitung nilai fiturnya seperti yang dijelaskan dalam *subroutine* pemilihan fitur. Kemudian dilanjutkan dengan menentukan parameter klasifikasinya.

- *Subroutine* Menentukan d Masing-masing Alphabet

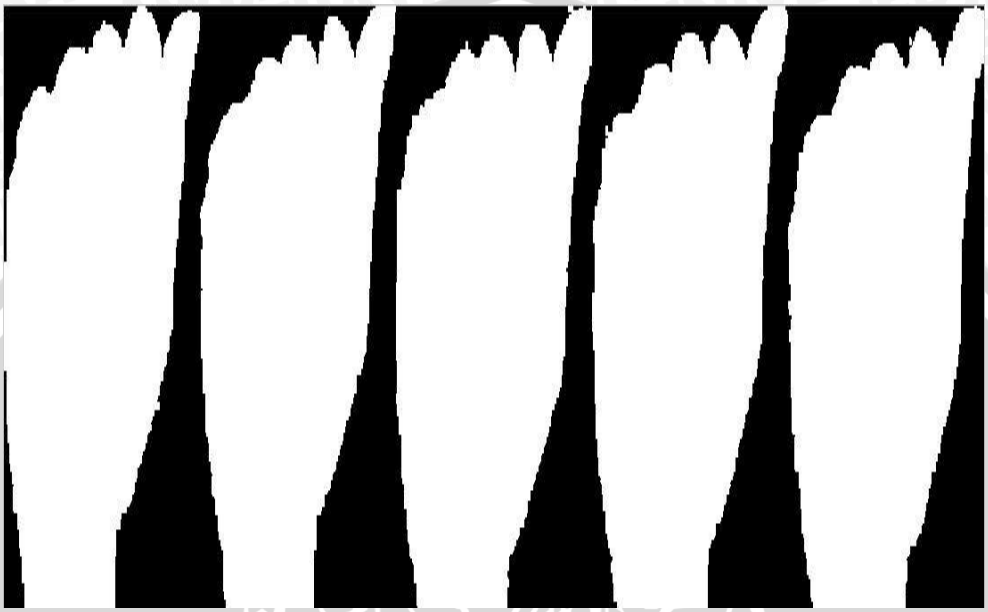
  Tahapan ini menunjukkan pengumpulan 5 citra dari masing-masing alphabet yang digunakan sebagai data latih diambil nilai fiturnya untuk digunakan sebagai data latih.





***Flowchart* 5.13 *Subroutine* d Masing-masing Alphabet**

Dalam *subroutine* ini terdapat kumpulan proses yang panjang. Mulai dari menyatukan 5 citra alphabet sampel yang akan diambil nilai fiturnya yang kemudian disatukan menjadi 1 citra utuh seperti pada Gambar 5.5.



Gambar 5.5 Citra sampel yang dijadikan satu menjadi 1 citra

Setelah proses penyatuan tersebut, citra akan dipotong seukuran lebar 300 dan tinggi 400 untuk mendapatkan 1 bentuk tangan saja yang diteliti. Itulah yang membuat perlunya proses looping 5 kali pada *flowchart* di atas. Karena ada 5 citra biner bentuk tangan yang harus diteliti.

Selanjutnya masuk *subroutine* menentukan titik tengah tangan, menentukan tepian tangan dan menentukan jarak titik tengah ke tepian. Dimana semua memiliki proses yang sama. Hanya saja hasil perhitungan jarak titik tengah ke tepian tangan disimpan ke dalam variabel dA yang mewakili nilai fitur masing-masing alphabet seperti pada persamaan (5.6).

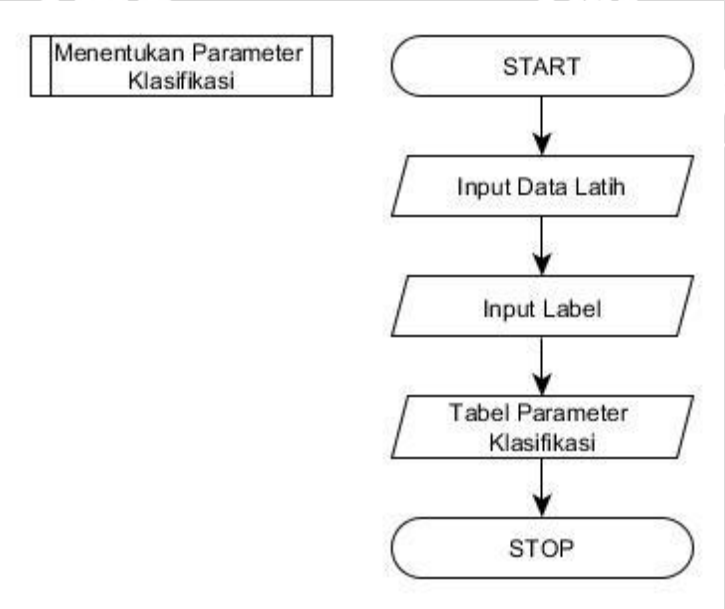
$$\begin{aligned} dA1 &= d(1,:) \\ dA2 &= d(2,:) \\ dA3 &= d(3,:) \\ dA4 &= d(4,:) \\ dA5 &= d(5,:) \end{aligned} \quad \ldots(5.6)$$

Keterangan :

dA1	: d Alphabet ke 1	d(1,:)	: d bari 1 dan seluruh kolom
dA2	: d Alphabet ke 2	d(2,:)	: d bari 2 dan seluruh kolom
dA3	: d Alphabet ke 3	d(3,:)	: d bari 3 dan seluruh kolom
dA4	: d Alphabet ke 4	d(4,:)	: d bari 4 dan seluruh kolom
dA5	: d Alphabet ke 5	d(5,:)	: d bari 5 dan seluruh kolom

- *Subroutine* Menentukan Parameter Klasifikasi

Flowchart 5.14 menunjukkan langkah-langkah menambahkan nilai label yang merepresentasikan kesimpulan dari klasifikasi KNN. Hasilnya berupa tabel yang berisikan data latih beserta kesimpulan klasifikasi pada masing-masing barisnya.



***Flowchart* 5.14 *Subroutine* Parameter Klasifikasi**

Tahap pertama dalam *subroutine* ini adalah menentukan Data Latih yang berisikan hasil dari *subroutine* menentukan d masing-masing sampel alphabet sebanyak alphabet anggota grup pembobotan terpilih (N) seperti pada persamaan (5.7).

$$\text{Data Latih} = 5 \times N \quad \ldots(5.7)$$

Jika masing-masing alphabet memiliki 5 sampel yang digunakan sebagai data latih, dengan demikian ada 130 data latih seperti yang ditunjukkan pada Tabel 5.7.

Tabel 5.10. Tabel jumlah data latih masing-masing kriteria

Grup	Jumlah Alphabet Anggota Grup (N)	Jumlah Sampel	Data Latih
G1	7	5	35
G2	13	5	65
G3	6	5	30
Total			130

Selanjutnya karena dalam klasifikasi di matlab nilai grup tidak bisa diisi langsung dengan alphabet, maka dibutuhkan angka untuk mewakili suatu grup yang disebut Label yang terurut dari 1 sampai jumlah anggota kriteria dari grup pembobotan terpilih (N). Pada Tabel 5.11 yang menunjukkan parameter klasifikasi grup 1.

Tabel 5.11 Label G1

Alphabet Anggota G1	Label
A	1
E	2
M	3
N	4
O	5
S	6
T	7

Tabel 5.12 yang menunjukkan parameter klasifikasi grup 2.

Tabel 5.11. Label G2

Alphabet Anggota G1	Label
B	1
C	2
D	3
F	4
I	5
K	6
L	7
R	8
U	9
V	10
W	11
X	12
Y	13

Tabel 5.13 yang menunjukkan parameter klasifikasi grup 3.

Tabel 5.13 Label G3

Alphabet Anggota G1	Label
G	1
H	2
J	3
P	4
Q	5
Z	6

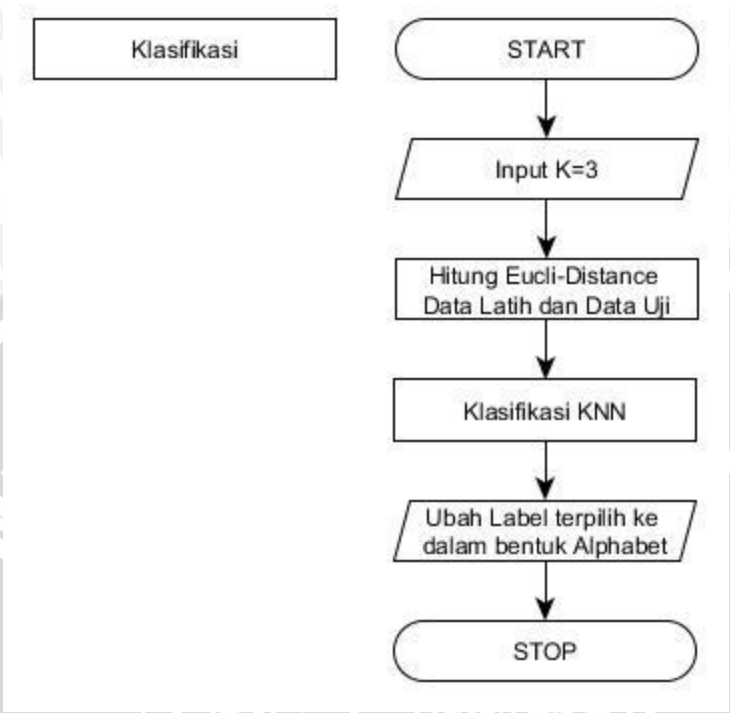
Setelah selesai menentukan nilai Label dan Data Latih, selanjutnya membuat Tabel dengan jumlah baris sebanyak Data Latih dan kolom sebanyak jumlah fitur data uji seperti pada Tabel 5.14.

Tabel 5.14 Contoh parameter klasifikasi

Alphabet	Fitur ke 1	Fitur ke 2	Fitur ke 3	...	Fitur ke 1100	Label
A	151.2382	149.4122	148.5295	...	268.425	1
A	190.9188	187.3953	186.6815	...	267.0131	1
A	190.9188	190.213	189.5099	...	266.4432	1
A	192.333	191.6272	190.9241	...	266.1766	1
A	202.3116	201.6259	189.5521	...	263.2128	1
E	157.8512	157.8005	156.8502	...	265.9323	2
E	166.7693	166.2348	165.7106	...	268.9721	2
E	163.0491	164.3959	163.878	...	264.6394	2
E	168.434	169.7793	169.2602	...	262.6176	2
E	160.1624	159.4804	163.0491	...	266.9476	2
M	166.012	166.6283	166.1114	...	259.9885	3
M	167.3141	166.7693	166.2348	...	259.3839	3
M	162.2498	164.1585	163.233	...	260.931	3
M	157.8924	160.0125	159.38	...	260.931	3
M	158.619	158.3982	157.9652	...	266.1597	3
...	...	...	...	...	...	...

5.1.6. Subroutine Klasifikasi

Flowchart 5.15 menunjukkan langkah-langkah menentukan alphabet hasil klasifikasi KNN.



***Flowchart* 5.15 *Subroutine* Klasifikasi**

Seperti yang sudah dijelaskan bahwa penelitian ini menggunakan metode *K-Nearest Nehbor* (KNN) untuk klasifikasinya. Metode ini dipilih karena memiliki perhitungan yang sangat sederhana. Ditunjang dengan pengukuran jaraknya menggunakan *Eucli-Distance*, sehingga membuat proses perhitungan yang lebih cepat.

Untuk keperluan klasifikasi sistem membutuhkan data uji dan data latih. Data uji diperoleh dari proses menangkap citra tangan hingga pengolahan citra seperti yang ditunjukkan pada *flowchart* 5.15. Sedangkan data latihnya menggunakan hasil dari *subroutine* pelatihan sistem. Dengan demikian dalam proses klasifikasi ini hanya memerlukan memasukkan nilai K=3, kemudian menggunakan *Eucli-Distance* untuk menghitung kedekatan data uji dengan data latih seperti pada persamaan (5.8).

$$D = \sqrt{\sum_{i=1}^{1100} (U_i - L_i)^2} \quad \text{... (5.8)}$$

Keterangan :

D = Eucli-Distance

U_i = Nilai Fitur Data Uji ke-i

L_i = Nilai Fitur Data Latih ke-i

Setelah mendapatkan nilai *Eucli-Distance*, nilai tersebut akan disusun dalam tetangga terdekat yaitu nilai perhitungan *Eucli-Distance* terkecil, hingga terjauh yaitu dengan nilai *Eucli-Distance* terbesar. Nilai K=3 ditentukan dengan melakukan percobaan dengan menghitung *success rate* yang dihasilkan KNN dengan k angka ganjil mulai dari 1 sampai 19 yang ditunjukkan pada Tabel 5.15.

Tabel 5.15 *Success rate* percobaan nilai k pada group 1

k	Success Rate
1	100.00 %
3	100.00 %
5	95.65 %
7	95.65 %
9	95.65 %
11	91.30 %
13	82.61 %
15	78.26 %
17	69.57 %
19	82.61 %

Tabel 5.16 *Success rate* percobaan nilai k pada group 2

k	Success Rate
1	100.00 %
3	100.00 %
5	100.00 %
7	97.67 %
9	95.35 %
11	100.00 %
13	90.70 %
15	83.72 %
17	88.37 %
19	74.42 %

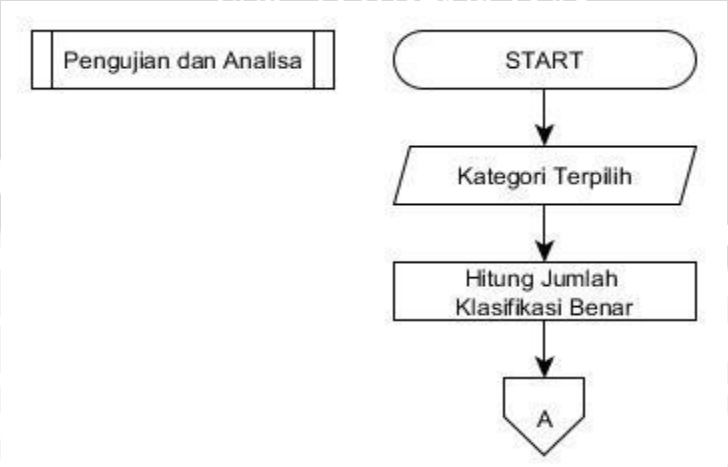
Tabel 5.17 *Success rate* percobaan nilai k pada group 3

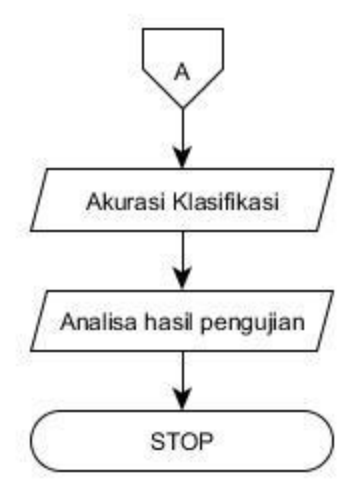
k	Success Rate
1	100.00 %
3	100.00 %
5	100.00 %
7	90.00 %
9	100.00 %
11	85.00 %
13	90.00 %
15	95.00 %
17	95.00 %
19	90.00 %

Percobaan dilakukan dengan mengkondisikan $\frac{1}{3}$ dari keseluruhan data latih digunakan sebagai data uji, sedangkan sisanya sebagai data latih. Karena ada 260 data set yang digunakan, maka ada 86 data uji. Percobaan dilakukan dengan membagi 260 data set ke dalam 3 group rekomendasi, hasilnya ditampilkan dalam Tabel 5.12, 5.13 dan 5.14 yang menunjukkan bahwa pada k=3 memiliki *success rate* tertinggi dari ketiga group. Sehingga k=3 dipilih dalam penelitian ini untuk menghasilkan klasifikasi pengenalan alphabet yang terbaik. Setelah mendapatkan klass hasil klasifikasinya yang masih berupa angka Label, maka ubah angka tersebut menjadi string agar bisa menampilkan alphabet yang seharusnya pada antarmuka pengguna.

5.1.7. Subroutine Pengujian dan Analisa

Tabel 5.16 menunjukkan langkah-langkah validasi hasil klasifikasi SMART KNN dan menganalisa perilaku hasil klasifikasi SMART KNN.





***Flowchart* 5.16 *Subroutine* Pengujian dan Analisa**

Dalam penelitian ini data diuji dalam 2 kondisi, yaitu menggunakan tangan dengan telapak tangan lebih pendek daripada jari tangan dan warna kulit cerah, sedangkan kondisi lainnya telapak tangan lebih panjang daripada jari tangan dan warna kulit gelap. Hasil pengujian dinyatakan dalam 2 nilai, yaitu benar dan salah. Hal ini didasarkan pada rumusan masalah dan tujuan dibuatnya program simulasi ini yang berfokus pada pengolahan citra hingga klasifikasi citra tangan yang ditangkap kamera untuk menghasilkan alphabet yang sesuai. Sehingga pengujian bernilai benar apabila citra tangan yang ditangkap kamera diklasifikasikan ke dalam kategori alphabet yang sama dengan alphabet bahasa isyarat yang seharusnya. Misalkan citra tangan yang diarahkan agar ditangkap kamera adalah alphabet A, maka bernilai benar, jika kategori yang dihasilkan dari proses klasifikasinya adalah kategori alphabet A dan bernilai salah apabila hasil klasifikasinya kategori alphabet lain. Kemudian semua nilai benar pada masing-masing alphabet dihitung jumlahnya dibagi dengan banyak pengujian dikali 100% untuk menghitung akurasi kebenaran pada penelitian ini. Tiap alphabet diuji sebanyak 5 kali dalam 2 kondisi di atas. Sehingga adapun total percobaan pada penelitian ada 260 kali.

Sedangkan proses analisa adalah proses pengamatan terhadap perilaku hasil pengujian yang ada untuk selanjutnya digunakan sebagai pertimbangan dalam penarikan kesimpulan.

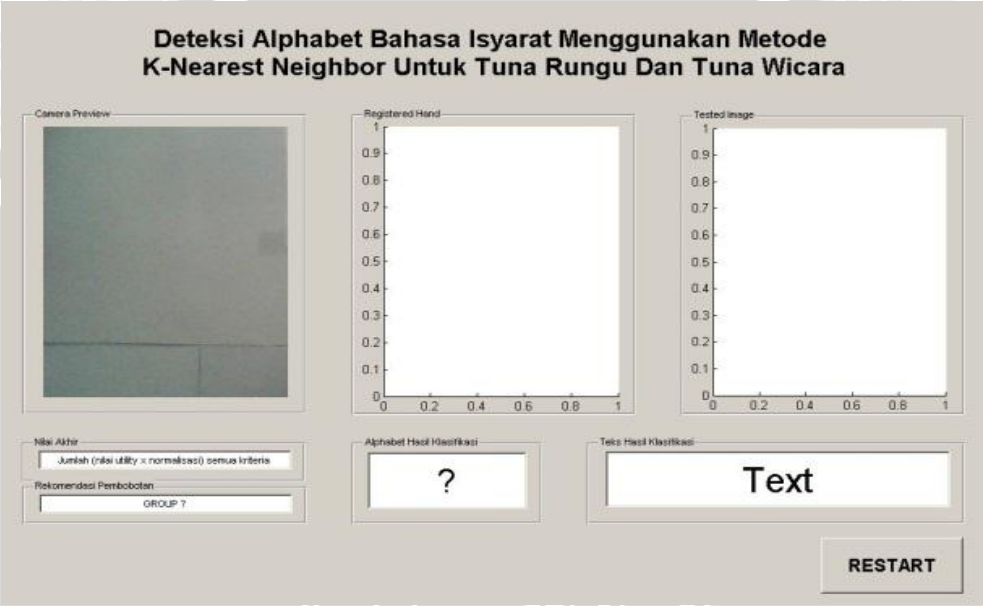
5.2. Implementasi

Pada bagian ini menjelaskan alur implementasi perancangan yang telah dibuat sebelumnya. Mulai dari membuat GUI yang dibutuhkan sebagai interface yang

memudahkan pengguna dalam menjalankan operasi programnya, menangkap gambar dan mengolahnya hingga menampilkan hasil klasifikasi.

a. Graphical User Interface (GUI)

Gambar 5.6 ditunjukkan GUI yang sedang berjalan siap untuk menangkap citra tangan yang diarahkan ke kamera sesuai kotak yang sudah disiapkan.



Gambar 5.6 *Launching* GUI

Tujuan dibuatnya GUI ini untuk memudahkan pengguna untuk melihat hasil pemrosesan data yang diolah dalam sistem ini dan sekaligus menjalankan perintah selanjutnya yang harus dikerjakan sistem. Adapun deskripsi kegunaan kotak dan tombol dalam Gambar 5.6 seperti yang tertulis pada Tabel 5.18.

Tabel 5.18 Deskripsi kegunaan kotak dan Tabel pada GUI

Nama Kotak atau Tombol	Fungsi
Kotak *Camera Preview*	Menampilkan citra yang ditangkap untuk diuji
Kotak *Registered Hand*	Menampilkan citra hasil deteksi warna kulit
Kotak *Tested Image*	Menampilkan citra biner hasil segmentasi beserta titik tengahnya
Kotak Nilai Akhir	Menampilkan angka hasil perhitungan nilai akhir yang merujuk pada rekomendasi
Kotak Rekomendasi Pembobotan	Menampilkan group hasil pembobotan

Tabel 5.18 Deskripsi kegunaan kotak dan Tabel pada GUI (lanjutan)

Nama Kotak atau Tombol	Fungsi
Kotak Alphabet Hasil Klasifikasi	Menampilkan alphabet hasil klasifikasi
Kotak Teks Hasil Klasifikasi	Menampilkan susunan alphabet hasil klasifikasi
Tombol *Restart*	Memberikan perintah untuk mengulang program simulasi ini dari awal

b. Menangkap Citra Tangan

Gambar 5.7 ditunjukkan tangan yang diarahkan ke kamera ditangkap dalam waktu yang sudah ditentukan sebelumnya. Jadi proses menangkap citra dilakukan secara otomatis tanpa perlu menekan tombol tangkap.

Gambar 5.7 Menangkap gambar tangan

Untuk memudahkan fokus dalam menangkap objek yang diteliti, maka tangan harus diarahkan sesuai ukuran kotak yang disediakan GUI.

c. Pengolahan Citra

Adapun tahapan dalam pengolahan citra meliputi deteksi warna kulit untuk membedakan tangan dan latarbelakangnya, serta segmentasi untuk binerasi citra.

- Deteksi Warna Kulit

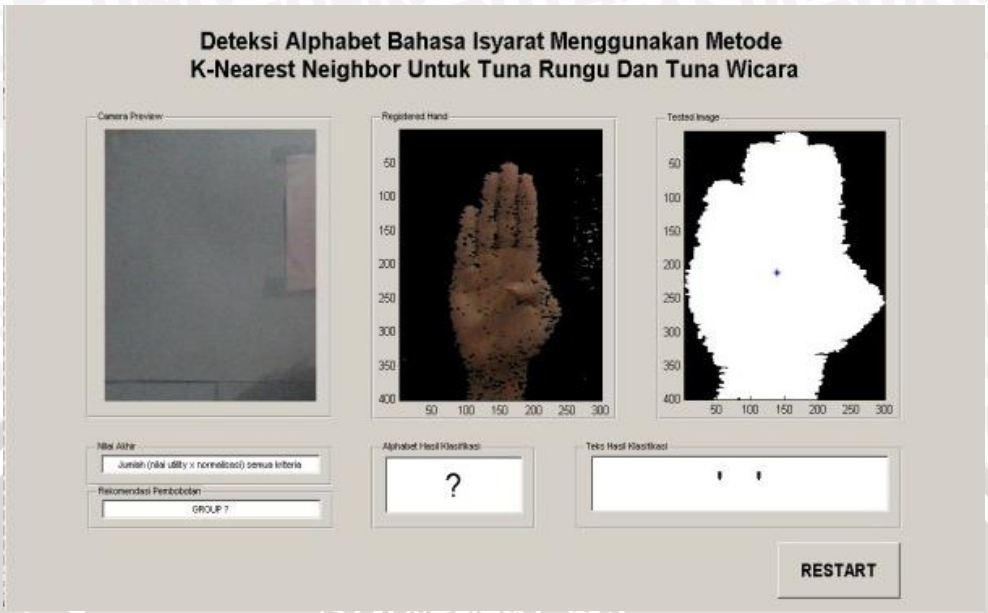
Citra RGB hasil tangkapan kamera pada proses sebelumnya kemudian diolah menggunakan deteksi warna kulit untuk menghilangkan latarbelakang dan menyisakan hanya tangan yang akan diteliti.

Gambar 5.8 Hasil *Skin Detection*

Ini merupakan bagian dari tahap preprocessing untuk memfokuskan penelitian hanya pada tangan saja dan menghapuskan objek lainnya dari tangan.

- Segmentasi

Citra tangan yang sudah dihilangkan latarbelakangnya kemudian dibinerisai dan ditampilkan sesuai kotak yang sudah disiapkan.

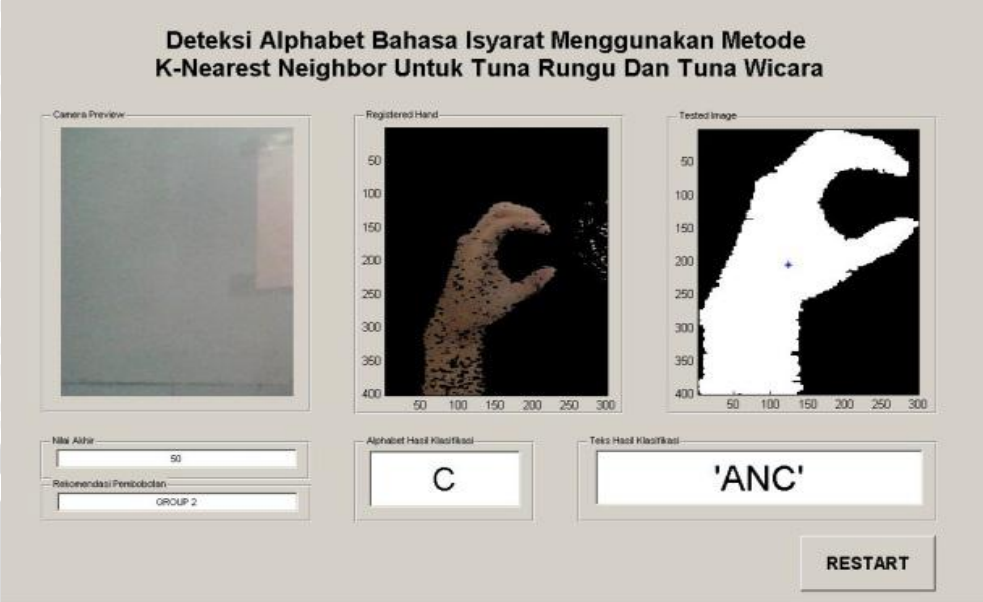


Gambar 5.9 Hasil segmentasi citra dan menampilkan titik tengahnya

Ini merupakan tahapan mengubah hasil registrasi citra menjadi keabuan dengan thresholding, pengisian lubang, dilasi dan menghapus area yang kurang dari 100 pixel yang bertujuan untuk meminimalkan noise dari objek yang diteliti sekaligus pada Gambar 5.9 menampilkan titik tengah yang digunakan untuk menentukan nilai fitur dalam proses klasifikasi nanti.

d. Klasifikasi

Citra biner yang sudah siap dilanjutkan dengan pembobotan SMART hingga menghasilkan grup rekomendasi yang sesuai. Selanjutnya citra biner diambil nilai fiturnya berdasarkan algoritma yang sudah ditentukan sebelumnya. Kemudian nilai fitur data uji diklasikasi dengan data latih yang berisikan alphabet anggota grup yang terpilih menggunakan KNN.



Gambar 5.10 Menampilkan hasil klasifikasi membentuk teks

Ini merupakan tahap akhir dalam proses pengolahan citra. Setelah menentukan nilai fiturnya, maka selanjutnya data-data yang telah dikumpulkan diklasifikasi menggunakan SMART KNN dan menampilkan hasil klasifikasinya ke dalam bentuk teks antar muka pengguna seperti pada Gambar 5.10.

5.3. Source Code Program

Pada bagian ini menjelaskan *source code* yang digunakan untuk menghasilkan program penelitian ini. Mulai dari menangkap citra tangan hingga klasifikasi. Sedangkan untuk pengujian dan analisa dilakukan secara manual.

a. Menangkap Citra Tangan

Gambar 5.11 berisikan kumpulan *source code* yang digunakan untuk menangkap citra pada GUI yang sudah disiapkan. Kemudian menetapkan jumlah alphabet yang akan dihasilkan untuk ditampilkan sebagai teks pada variabel l.

```
global vid
% memfokuskan area yang ditangkap seukuran axes 1
axes(handles.axes1);
vid = videoinput('winvideo',1,'MJPG_1024x576');
hImage = imagesc(zeros(400,300), 'Parent', handles.axes1);
preview(vid, hImage);set(vid,'ReturnedColorSpace','rgb');
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb');
pause(1);

% menentukan panjang alphabet yang ditampilkan sebanyak l
l = 1;
teks = [1:l];

% mulai menangkap citra
for acquired = 1:l
pause(5);
snap = getsnapshot(vid);
```

Gambar 5.11 *Source code* menangkap citra tangan

Pada source code di atas menunjukkan variable global vid digunakan untuk menyimpan fungsi memanggil kamera. Format ukuran citra yang dihasilkan kamera Logitech HD 720p yang terdeteksi di matlab adalah MJPG_1024x576. Sedangkan variable hImage digunakan untuk menampilkan area untuk menangkap citra tangan seukuran tinggi 400 dan lebar 300 pixel seperti pada Gambar 5.11. Setelah dalam penelitian ini melakukan menangkap citra sebanyak variable l. Variabel tersebut yang menentukan jumlah citra yang akan diteliti dalam sekali proses berjalan. Kemudian tiap citra yang ditangkap disimpan dalam variable snap.

b. Pengolahan Citra

Dalam proses pengolahan citra terdapat 2 proses utama yang dijelaskan ke dalam 2 subroutine, yaitu Registrasi Citra dan Segmentasi.

- Deteksi Warna Kulit

Gambar 5.12 berisikan kumpulan *source code* yang digunakan untuk mengubah citra hasil tangkapan kamera ke bentuk double dan inisiasi nilai HSV dan YCrCb. Gambar 5.13 digunanakan untuk penghapusan latarbelakang dari citra tangan hasil tangkapan kamera menggunakan metode YCrCb-HSV.

```
% menampilkan axes hand register
axes(handles.axes2);
% crop the image to focus on cam preview only
gambar = imcrop(snap,[1 5 300 400]);
imwrite(gambar, 'pict.tif');
I = double(gambar);

%%skin detection
[hue,s,v]=rgb2hsv(I);
cb =  0.148 * I(:,:,1) - 0.291 * I(:,:,2) + 0.439 * I(:,:,3) + 128;
cr =  0.439 * I(:,:,1) - 0.368 * I(:,:,2) - 0.071 * I(:,:,3) + 128;
[w h] = size(I(:,:,1));
```

Gambar 5.12 *Source code* inisiasi HSV dan YCbCr

```
for i=1:w
  for j=1:h
      % kamera webcam
      % kondisi cerah di gazebo filkom jam 10.00
      %if  120<=cr(i,j) && cr(i,j)<=150 && 170<=cb(i,j) && cb(i,j)<=190 && 0.01<=hue(i,j) && hue(i,j)<=0.09

      % kamera external Logitech HD 720p
      % kondisi ruang cahaya redup
      if  130<=cr(i,j) && cr(i,j)<=180 && 130<=cb(i,j) && cb(i,j)<=180 && 0.01<=hue(i,j)
      % kondisi cerah di gazebo filkom jam 10.21
      %if  125<=cr(i,j) && cr(i,j)<=140 && 150<=cb(i,j) && cb(i,j)<=190 && hue(i,j)<=0.8
          segment(i,j)=1;
      else
          segment(i,j)=0;
      end
  end
end

im(:,:,1)=I(:,:,1).*segment;
im(:,:,2)=I(:,:,2).*segment;
im(:,:,3)=I(:,:,3).*segment;
imwrite(uint8(im), 'registered_hand.tif');
imagesc(uint8(im));
```

Gambar 5.13 *Source code* deteksi warna kulit

Untuk keperluan registrasi tangan, pertama citra tangan RGB diubah menjadi HSV menggunakan [hue,s,v] = rgb2hsv[I] dan CbCr menggunakan menggunakan variable cb dan cr seperti pada Gambar 5.12. Kemudian proses registrasi tangan menggunakan batasan nilai seperti pada Gambar 5.13.

- Segmentasi Citra

Gambar 5.14 berisikan kumpulan *source code* yang digunakan untuk mengubah citra hasil deteksi warna kulit menjadi bentuk biner. Kemudian *filtering noise* untuk membersihkan citra biner dari noise yang bisa mengurangi efektifitas klasifikasi. Gambar 5.15 berisikan *soure code* untuk cropping citra sesuai ukuran tangan.

```
% ------- Data Uji -------
axes(handles.axes3);
% mengubah citra menjadi bentuk biner
gray = rgb2gray(imread('registered_hand.tif'));
t=graythresh(gray);
bw=im2bw(gray, t);
% filterig noise
bw=imfill(bw,'holes');
bw=bwmorph(bw,'dilate');
bw=imfill(bw,'holes');
% hapus pixel kurang dari 200
bw=bwareaopen(bw, 200);

% cropping fokus pada objek yang diteliti
% memotong pada area yang memiliki tinggi (row) >200 pixel
bb=regionprops(bw,'BoundingBox');
% filter centre boundingbox
i = 1;
while i <= length(bb)
    if bb(i).BoundingBox(:,3)>200
        bb = bb(i);
        i = length(bb);
    end
    i = i+1;
end
img=imcrop(bw,bb(1).BoundingBox);
```

Gambar 5.14. *Source code* binerisasi citra dan cropping pertama

```
% mengambil fokus pada image
% memotong pada area yang memiliki panjang (kolom) >200 pixel
s=regionprops(img,'FilledImage');
% choose centre filled image
i = 1;
while i <= length(s)
    [m n] = size(s(i).FilledImage);
    if n>200
        s = s(i);
        i = length(s);
    end
    i = i+1;
end
img=s.FilledImage;
```

Gambar 5.15 *Source code* cropping kedua

Sedangkan untuk keperluan segmentasi citra mulai dari mengubah citra menjadi warna keabuan, mengubahnya menjadi bentuk biner ditunjukkan pada Gambar 5.13. Untuk keperluan cropping seukuran bentuk tangan diperlukan 2 kali proses cropping yang ditunjukkan dengan img=imcrop(bw,bb(i).Boundingbox) pada Gambar 5.14 dan kedua seluruh proses pada Gambar 5.15.

c. Pembobotan

Gambar 5.16 berisikan kumpulan *source code* yang digunakan untuk menentukan nilai gx dan gy sebagai nilai utilitas pembobotan. Sedangkan pada gambar 5.17 berisikan aturan untuk mengarahkan citra data uji ke grup rekomendasi yang sesuai nilai akhir.

```
img = imread('segmented_hand.tif');

% --- Atribut Pembobotan gx1 berdasar ordinat (1:300,75) ---
c = 1; r = 75; x = 300;
uk = 0; gx = [1:x];
i= 1; l = length(gx);

while i<=l
    gx(i) = img(r,c);
    uk = uk+gx(i);
    i = i+1; c = c+1;
end
gx = uk;

% --- Atribut Pembobotan gy1 berdasar ordinat (25,1:400) ---
c = 25; r = 1; y = 400;
uk = 0; gy = [1:y];
i = 1; l = length(gy);

while i<=l
    gy(i) = img(r,c);
    uk = uk+gy(i);
    i = i+1; r = r+1;
end
gy = uk;

% inisiasi nilai utility kriteria
nk1 = 0; nk2 = nk1; nk3 = nk1;
```

Gambar 5.16 *Source code* inisiasi gx dan gy

```
% inisiasi nilai utility kriteria
nk1 = 0; nk2 = nk1; nk3 = nk1;

if gy<90
     nk3=100;
elseif gx<200
     nk2=100;
else
     nk1=100;
end

% bobot kriteria setelah normalisasi
bk1 = 0.27;
bk2 = 0.50;
bk3 = 0.23;

% nilai akhir
na = (nk1*bk1)+(nk2*bk2)+(nk3*bk3);
na = floor(na);
set(handles.nilai_akhir, 'String', na);

% rekomendasi
if na==27
     text='GROUP 1'; set(handles.rekomendasi, 'String', text);
     group1;
elseif na==50
     text='GROUP 2'; set(handles.rekomendasi, 'String', text);
     group2;
elseif na==23
     text='GROUP 3'; set(handles.rekomendasi, 'String', text);
     group3;
end
```

Gambar 5.17 *Source code* proses pembobotan

Dalam proses pembobotan, untuk menentukan kriteria pembobotan dan menentukan bobot masing-masing kriteria dilakukan secara manual. Sisanya untuk keperluan menentukan nilai utilitas maka diperlukan proses inisiasi batas gx dan gy yang ditunjukkan pada Gambar 5.16. Sedangkan untuk menentukan nilai akhir, mulai dari menentukan nilai utilitas citra yang yang diteliti hingga perhitungan nilai akhir menggunakan source pada Gambar 5.17.

d. Pemilihan Fitur

Dalam proses ini terdapat 2 *subroutine* yang menjadi proses utama dalam pemilihan fitur. Pertama *subroutine* tentang menentukan tepian tangan dan kedua mengukur jarak titik tengah ke tepian tangan.

- Menentukan Titik Tengah tangan

  Gambar 5.18 berisikan kumpulan *source code* yang digunakan untuk memilih titik tengah yang benar-benar tepat di titik tengah tangan. Karena ketika citra biner tidak bersih dari noise, maka akan ada beberapa titik tengah yang terdeteksi fungsi *regionprops centroid* yang bukan menjadi titik tengah yang seharusnya.

```
% resize citra dan tampilkan di gui
% ukuran ini digunakan sebagai patokan ukuran citra yang diuji
% agar berapapun ukuran tangan pengguna yang ditangkap tidak mempengaruhi
% pengukuran jarak center point ke tepian objek yang diteliti
img = imresize(img, [400 300]);
imwrite(img, 'segmented_hand.tif');
% inisiasi center point
% menentukan ceter point baru setelah cropping pertama
centroids = regionprops(img,'centroid');
% convert struct into matrix
centroids = floor(cat(1, centroids.Centroid));
% menghapus center point lain yang tidak dibutuhkan
i = 1;
while i <= length(centroids(1))
    % jika jumlah center poin hanya satu maka keluar dari looping
    if length(centroids(:,1))<=1
        i = length(centroids);
    % jika center poin > 1 maka eliminasi center point yg tdk dibutuhkan\
    elseif 100<=centroids(i,1) && centroids(i,1)<=200 && centroids(i,2)<260
        centroids1(:,1) = centroids(i,1); centroids1(:,2) = centroids(i,2);
        i = length(centroids);
        centroids = centroids1;
    end
    i = i+1;
end
imagesc(img);
% menunjukkan letak center point
hold on
plot(centroids(:,1),centroids(:,2), '*')
hold off
```

Gambar 5.18 *Source code* menentukan titik tengah tangan

Tahap awal untuk menentukan titik tengah tangan adalah dengan mengubah citra yang diteliti keukuran standar tinggi 400 dan lebar 300 menggunakan img = imresize(img, [400, 300]). Kemudian masuk proses eliminasi titik tengah yang tidak diperlukan menggunakan fungsi *while* seperti pada Gambar 5.18.

- Menentukan Tepian Tangan

Gambar 5.19 berisikan kumpulan *source code* yang digunakan untuk menentukan tepian kiri dan kanan tangan menggunakan *scanning* biner 1 dari kiri ke kanan. Kemudian menyimpan nilai koordinat tiap tepian kiri di variabel (exl, eyl) dan (exr, eyr). Sedangkan pada gambar 5.20 merupakan *source code* untuk deteksi tepian atas tangan. Dengan teknik *scanning* biner 1 dari atas ke bawah, nilai koordinat tepian atas tangan disimpan ke dalam variabel (exa, eya).

```
% menentukan tepian tangan
e = 1:400; e(1:400)=0;
exl = e; eyl = e;
exr = e; eyr = e;
for n = 1:400
    m = 1;
    while m<=300
        if img(n,m)>=1
            eyr(n)=n; exr(n)=m;
            if exl(n)<1 && eyl(n)<1
                eyl(n)=n; exl(n)=m;
            end
        end
        m=m+1;
    end
end
```

Gambar 5.19 *Source code* menentukan tepi kiri dan Kanan tangan

```
e = 1:300; e(1:300)=0;
exa = e; eya = e;
for m = 1:300
    n = 1;
    while n<=400
        % cek biner 1 ordinat (n,m)
        if img(n,m)>=1
            % cek isi exa dan eya
            if exa(m)<1 && eya(m)<1
                exa(m)=m; eya(m)=n;
            end
        end
        n=n+1;
    end
end
```

Gambar 5.20 *Source code* menentukan tepi atas tangan

Untuk proses menentukan tepian tangan, yaitu deteksi sisi kanan yang ordinatnya disimpan ke dalam variabel exr dan eyr, sisi kiri yang ordinatnya disimpan ke dalam variabel exl dan eyl seperti pada Gambar 5.19. Sedangkan untuk sisi atasnya yang mengerjakan *scanning* dari atas ke bawah nilai ordinatnya disimpan ke dalam variabel exa dan eya seperti pada Gambar 5.20.

➢ Mengukur Jarak Titik Tengah ke Tepian Tangan

Gambar 5.21 berisikan kumpulan *source code* yang digunakan untuk mengukur dan menyimpan nilai fitur klasifikasi ke dalam variable d. Jumlah fitur yang terdeteksi dalam penelitian ini ada sebanyak 1100 fitur.

```
% -- Sub Mengukur Jarak Titik Tengah ke Tepian Tangan
% inisiasi 1100 ruang untuk mengukur jarak sbg fitur
d = 1:1100; d(1:1100) = 0;
for i=1:400
    d(i) = sqrt(abs((centroids(:,1)-exr(i))^2)+(abs((centroids(:,1)-eyr(i))^2)));
    d(400+i) = sqrt(abs((centroids(:,1)-exl(i))^2)+(abs((centroids(:,1)-eyl(i))^2)))
    if i<=300;
    d(800+i) = sqrt(abs((centroids(:,1)-exa(i))^2)+(abs((centroids(:,1)-eya(i))^2)))
    end
end
% menjadikan matrix hasil perbandingan jarak sebagai data uji
uji = d;
```

Gambar 5.21 *Source code* menentukan jarak titik tengah ke Tepian tangan

Penghitungan fitur menggunakan rumus 5.5 yang jika diaplikasikan ke dalam bentuk source code maka hasilnya seperti pada Gambar 5.21. Dimana hanya ada looping sebanyak 400 kali yang didalamnya ada 3 proses utama, yaitu menghitung d(i) yang digunakan untuk menghitung d titik tengah ke tepi kanan, d(400+i) untuk menghitung d titik tengah ke tepi kiri dan terakhir jika i $\leq$ 300, maka proses d(800+i) untuk menghitung d titik tengah ke tepi atas. Kondisi tersebut dimaksudkan agar, jumlah looping untuk menghitung d titik tengah ke tepi atas hanya berlangsung 300 kali. Dimana sesuai dengan jumlah tepi atas yang hanya terdapat 300 ordinat yang disimpan ke dalam variabel exa dan eya.

e. Pelatihan Sistem

➢ Menentukan d Masing-masing Alphabet

Gambar 5.22-5.25 berisikan urutan pengambilan nilai fitur data latih.

```
I1 = imread('A.tif');
I2 = imread('A (2).tif');
I3 = imread('A (3).tif');
I4 = imread('A (4).tif');
I5 = imread('A (5).tif');
I = [I1,I2,I3,I4,I5];
imshow(I);

% inisiasi variabel
ax = 300;
d = 1:ax; d(1:ax) = 0; d = [d;d;d;d;d];
x = 0;
for c=1:5
    crop = imcrop(I, [x 1 300 400]);
    % cek center point
    centroids = regionprops(crop,'centroid');
    % convert struct into matrix
    centroids = floor(cat(1, centroids.Centroid));
    % menghapus center point lain yang tidak dibutuhkan
    i = 1;
    while i <= length(centroids)
        % jika jumlah center poin hanya satu maka keluar dari looping
        if length(centroids(:,1))<=1
            i = length(centroids);
        % jika center poin > 1 maka eliminasi center point yg tdk dibutuhkan
        elseif 100<=centroids(i,1) && centroids(i,1)<=200 && centroids(i,2)<250
            centroids1(:,1) = centroids(i,1); centroids1(:,2) = centroids(i,2);
            i = length(centroids);
            centroids = centroids1;
        end
        i = i+1;
    end
```

Gambar 5.22 *Source code load* citra data sampel dan menentukan Titik tengahnya

```
% -- Sub Menentukan tepian tangan --
e = 1:400; e(1:400)=0;
exl = e; eyl = e;
exr = e; eyr = e;
for n = 1:400
    m = 1;
    while m<=300
        if crop(n,m)>=1
            eyr(n)=n; exr(n)=m;
            if exl(n)<1 && eyl(n)<1
                eyl(n)=n; exl(n)=m;
            end
        end
        m=m+1;
    end
end

% Tepian Tangan Atas
e = 1:300; e(1:300)=0;
exa = e; eya = e;
for m = 1:300
    n = 1;
    while n<=400
        if crop(n,m)>=1
            if exa(m)<1 && eya(m)<1
                eya(m)=n; exa(m)=m;
            end
        end
        n=n+1;
    end
end
```

Gambar 5.23 *Source code* menentukan tepian tangan

```
    % -- Sub Menentukan jarak titik tengah ke tepian --
    for i=1:400
        d(c,i) = sqrt(abs((centroids(:,1)-exr(i))^2)+(abs((centroids(:,1)-eyr(i))^2)));
        d(c,400+i) = sqrt(abs((centroids(:,1)-exl(i))^2)+(abs((centroids(:,1)-eyl(i))^2)));
        if i<=300;
        d(c,800+i) = sqrt(abs((centroids(:,1)-exa(i))^2)+(abs((centroids(:,1)-eya(i))^2)));
        end
    end

    % eksekusi gambar selanjutnya
    x = x+ax;
end
```

Gambar 5.24 *Source code* menentukan d masing-masing alphabet

```
% menjadikan matrix hasil perbandingan jarak sebagai data uji
dA1 = d(1,:);
dA2 = d(2,:);
dA3 = d(3,:);
dA4 = d(4,:);
dA5 = d(5,:);
```

Gambar 5.25 *Source code* menyimpan d alphabet ke-1 sampai 5

Pada tahap ini sistem akan membuat 1 citra utuh seperti pada Gambar 5.22 yang disusun dari kumpulan citra sampel yang disimpan dalam variable I. Kemudian agar bisa dilakukan pengukuran d masing-masing citra sampel ke tepian tangannya, maka dilakukan proses *cropping* seukuran lebar 300 dan tinggi 400 menggunakan fungsi imcrop dan sekaligus menentukan titik tengah tangannya menggunakan *source code* seperti pada Gambar 5.22.

Selanjutnya lakukan proses menggunakan deteksi tepian seperti pada Gambar 5.23 dan pengukuran d atau jarak ke tepian tangan seperti pada Gambar 5.24 yang hasilnya akan disimpan ke variable dA1, dA2, dA3, dA4 dan dA5 seperti source code pada Gambar 5.25. Untuk perhitungan d alphabet lainnya maka nama variable untuk menyimpan nilai hasil perhitungan jaraknya

menyesuaikan nama alphabet. Missal untuk alphabet B, maka variabel tempat menyimpannya diganti variabel dB1, dB2, dB3, dB4 dan dB5, dst.

- Menentukan Parameter Klasifikasi

Gambar 5.26 berisikan *source code* mengumpulkan nilai data latih dari suatu grup rekomendasi pembobotan untuk kemudian ditambahkan label sebagai representasi kesimpulan klasifikasi.

```
% data latih
Sample_A; Sample_E; Sample_M;
Sample_N; Sample_O; Sample_S; Sample_T;
% parameter klasifikasi
training = [
     dA1;dA2;dA3;dA4;dA5;
     dE1;dE2;dE3;dE4;dE5;
     dM1;dM2;dM3;dM4;dM5;
     dN1;dN2;dN3;dN4;dN5;
     dO1;dO2;dO3;dO4;dO5;
     dS1;dS2;dS3;dS4;dS5;
     dT1;dT2;dT3;dT4;dT5;
];

lable = 1:35;
lable(1:5) = 1; lable(6:10) = 2;
lable(11:15) = 3; lable(16:20) = 4;
lable(21:25) = 5; lable(26:30) = 6;
lable(31:35) = 7;
group = lable';
```

Gambar 5.26 *Source code* membuat parameter klasifikasi

Untuk membuat parameter klasifikasi seperti pada Tabel 5.11, maka langkah pertama yang perlu dilakukan adalah menentukan data latih sesuai ketentuan pada Tabel 5.7, selanjutnya dikumpulkan ke dalam 1 variabel, yaitu training. Kemudian input angka yang digunakan sebagai Label sesuai ketentuan pada Tabel 5.8, 5.9 dan 5.10. Sedangkan untuk implementasinya menggunakan source code pada Gambar 5.26.

f. Klasifikasi

Dalam proses klasifikasi pertama menentukan data uji menggunakan *source code* mulai dari menangkap citra pada Gambar 5.6 hingga 5.27 yang merupakan tahapan pemilihan fitur. Karena dalam menentukan data uji tersebut menggunakan citra tangan yang ditangkap langsung saat itu juga dari tangan

pengguna. Kemudian dilanjutkan dengan menentukan nilai K=3 karena nilai tersebut dianggap sudah memberikan akurasi klasifikasi yang cukup.

- Klasifikasi KNN

Gambar 5.27 berisikan *source code* pengecekan nilai hasil klasifikasi yang disesuaikan dengan nilai label. Jika nilai klasifikasi sesuai dengan suatu label tertentu, maka tampilkan alphabet yang seharusnya.

```
% Nilai k
k = 3;

clc;

% ------- KNN Classifying -------
class = knnclassify(uji, training, group, k, 'euclidean');
if class == 1
     text = 'A';     %disp(text);
     teks(acquired) = text;
     set(handles.alphabet, 'String', text);
elseif class == 2
     text = 'E';     %disp(text);
     teks(acquired) = text;
     set(handles.alphabet, 'String', text);
elseif class == 3
     text = 'M';     %disp(text);
     teks(acquired) = text;
     set(handles.alphabet, 'String', text);
elseif class == 4
     text = 'N';     %disp(text);
     teks(acquired) = text;
     set(handles.alphabet, 'String', text);
elseif class == 5
     text = 'O';     %disp(text);
     teks(acquired) = text;
     set(handles.alphabet, 'String', text);
```

Gambar 5.27 *Source code* proses metode knn dan mengubah angka hasil klasifikasi ke bentuk string

Untuk tahapan klasifikasinya, menggunakan fungsi knnclassify yang di dalamnya menggunakan metode *eucli distance* untuk mengukur kedekatan jarak antara data uji dengan data latihnya. Hasilnya berupa angka yang didapat dari label tadi yang mewakili klass hasil klasifikasi. Untuk menerjemahkan angka klass tersebut maka tiap angka klass diubah menjadi string alphabet menggunakan syntax set(handles.alphabet, 'String', text) seperti pada Gambar 5.27.

- Menampilkan Alphabet Hasil Klasifikasi

Gambar 5.28 berisikan source code untuk mengubah kumpulan nilai hasil klasifikasi sepanjang l yang sudah ditentukan sebelumnya ke dalam bentuk string agar bisa ditampilkan sebagai teks.

```
% menampilkan teks
teks = mat2str(teks);
set(handles.teks, 'String', teks);
```

Gambar 5.28 *Source code* menampilkan teks

Tahap terakhir, untuk menampilkan hasil klasifikasnya ke kolom Teks ke antar muka pengguna seperti pada Gambar 5.6, maka kumpulan string alphabet tadi ditampilkan menggunakan *source code* seperti pada Gambar 5.28.

BAB 6 PENGUJIAN DAN ANALISA

Dalam bab ini membahas tahapan dan kondisi pengujian program yang dibuat. Dalam pengujian ini fokus menghitung akurasi klasifikasi menggunakan SMART KNN dengan kondisi pertama yaitu telapak tangan besar-jari pendek dan warna kulit gelap, sedangkan kondisi kedua yaitu telapak tangan kecil-jari panjang dan warna kulit terang. Terakhir melakukan analisa data berdasarkan hasil pengujian tersebut.

6.1. Akurasi Klasifikasi

Mengacu pada rumusan masalah untuk membedakan tangan dengan latarbelakangnya, maka perlu dilakukan pengujian dengan kondisi warna kulit gelap dan terang. Serta untuk memenuhi penerapan SMART KNN untuk mengenali alphabet bahasa isyarat, dimana salah satu aspek dalam pembobotan masih menggunakan batasan berupa $\sum \text{biner} 1$ pada *gx* dan *gy*, maka perlu kondisi pengujian untuk mengukur beda panjang telapak tangan dan panjang jarinya.

Sehingga dalam penelitian ini pengujian dilakukan dengan 2 kondisi berbeda, yaitu kondisi pertama telapak tangan panjang-jari pendek dan warna kulit gelap seperti pada Gambar 6.1, sedangkan kondisi kedua yaitu telapak tangan pendek-jari panjang dan warna kulit terang seperti pada Gambar 6.2.

Gambar 6.1 Tangan kondisi pertama

Gambar 6.2 Tangan kondisi kedua

Menggunakan kedua kondisi di atas, diharapan bisa menunjukkan bahwa program simulasi ini bisa digunakan oleh siapapun walaupun ukuran dan warna kulit tangan yang berbeda-beda.

a. Kondisi Pertama, Telapak Besar-Jari Pendek dan Kulit Tangan Gelap

Tabel 6.1 menunjukkan hasil pengujian yang dilakukan menggunakan tangan tangan dengan kondisi Telapak Besar-Jari Pendek dan Kulit Tangan Gelap.

Tabel 6.1 Hasil pengujian kondisi pertama

Alphabet	Percobaan	Benar	Salah	Akurasi
A	5	5	0	100%
B	5	5	0	100%
C	5	5	0	100%
D	5	5	0	100%
E	5	5	0	100%
F	5	5	0	100%
G	5	5	0	100%
H	5	5	0	100%
I	5	5	0	100%
J	5	5	0	100%

Tabel 6.1 Hasil pengujian kondisi pertama (Lanjutan)

Alphabet	Percobaan	Benar	Salah	Akurasi
K	5	5	0	100%
L	5	5	0	100%
M	5	3	2	60%
N	5	5	0	100%
O	5	3	2	60%
P	5	5	0	100%
Q	5	5	0	100%
R	5	5	0	100%
S	5	5	0	100%
T	5	3	2	60%
U	5	5	0	100%
V	5	5	0	100%
W	5	5	0	100%
X	5	5	0	100%
Y	5	5	0	100%
Z	5	5	0	100%
Total	130	123	7	95%

b. Kondisi Kedua, Telapak Kecil-Jari Panjang dan Kulit Tangan Terang

Tabel 6.2 menunjukkan hasil pengujian yang dilakukan menggunakan tangan dengan kondisi Telapak Kecil-Jari Panjang dan Kulit Tangan Terang.

Tabel 6.2 Hasil pengujian menggunakan kondisi kedua

Alphabet	Percobaan	Benar	Salah	Akurasi
A	5	5	0	100%
B	5	5	0	100%
C	5	5	0	100%
D	5	4	1	80%
E	5	5	0	100%
F	5	5	0	100%
G	5	5	0	100%

Tabel 6.2 Hasil pengujian menggunakan kondisi kedua (Lanjutan)

Alphabet	Percobaan	Benar	Salah	Akurasi
H	5	5	0	100%
I	5	5	0	100%
J	5	5	0	100%
K	5	3	2	60%
L	5	4	1	80%
M	5	4	1	80%
N	5	5	0	100%
O	5	5	0	100%
P	5	3	2	60%
Q	5	5	0	100%
R	5	5	0	100%
S	5	5	0	100%
T	5	3	2	60%
U	5	5	0	100%
V	5	4	1	80%
W	5	5	0	100%
X	5	5	0	100%
Y	5	5	0	100%
Z	5	5	0	100%
Total	130	120	10	92%

Berdasarkan tabel di atas, selanjutnya menghitung rata-rata akurasi dari tabel 6.1 dan 6.2 seperti yang ditampilkan pada tabel 6.3.

Tabel 6.3 Rata-rata akurasi dari kedua kondisi percobaan

Tabel ke-	Akurasi
1	95%
2	92%
Total	93%

Seperti pada tabel 6.3 menunjukkan rata-rata akurasi dari kedua kondisi percobaan yang mencapai 93%. Persentase angka tersebut menunjukkan keberhasilan program simulasi ini bisa digunakan oleh siapapun. Sedangkan untuk mendapatkan hasil terbaik, maka tangan yang diujikan memenuhi kondisi pertama.

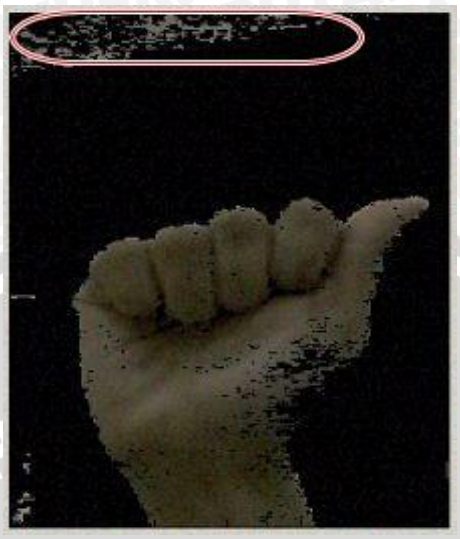
6.2. Analisa Hasil Pengujian

Setelah pengujian selesai dilakukan, berdasarkan pengujian tersebut ada beberapa alphabet yang salah klasifikasi. Dari kesalahan hasil klasifikasi tersebut, secara umum ada 4 jenis penyebab kesalahan yang harus dianalisa untuk selanjutnya dimasukkan ke dalam saran.

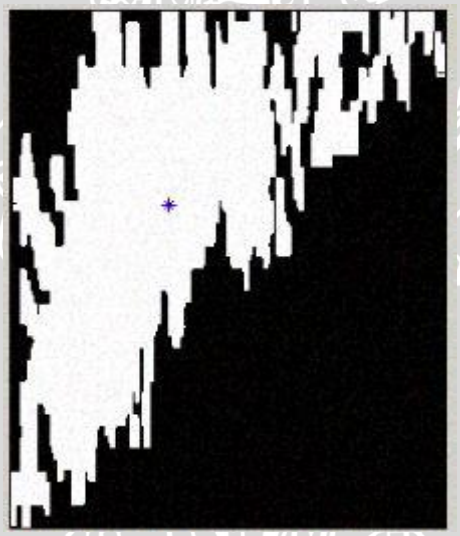
a. Proses Pengolahan Citra Tangan

Di dalam proses ini berisikan metode deteksi warna kulit yang digunakan untuk membedakan antara tangan dengan latarbelakangnya dan segmentasi. Jika *threshold* deteksi warna kulit tidak sesuai akan menghasilkan banyak *noise*, dimana membuat hasil ekstraksi mengalami banyak kesalahan seperti yang ditunjukkan pada Gambar 6.3, 6.4 dan 6.5.

Gambar 6.3 Citra RGB Warna Kulit Terang



Gambar 6.4 Hasil deteksi warna kulit tidak sempurna

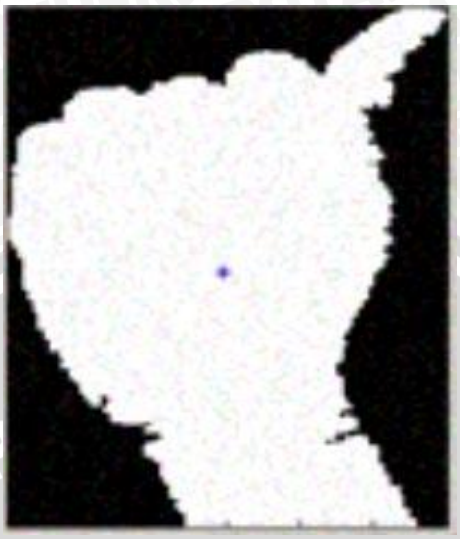


Gambar 6.5 Hasil binerasi salah area *cropping*

Sebaliknya, jika hasil pengolahan citra memiliki hasil yang mendekati sempurna, maka tidak akan terlalu berpengaruh pada citra data ujinya.

6.6 Citra RGB warna kulit gelap

Gambar 6.7 Hasil deteksi warna kulit sempurna



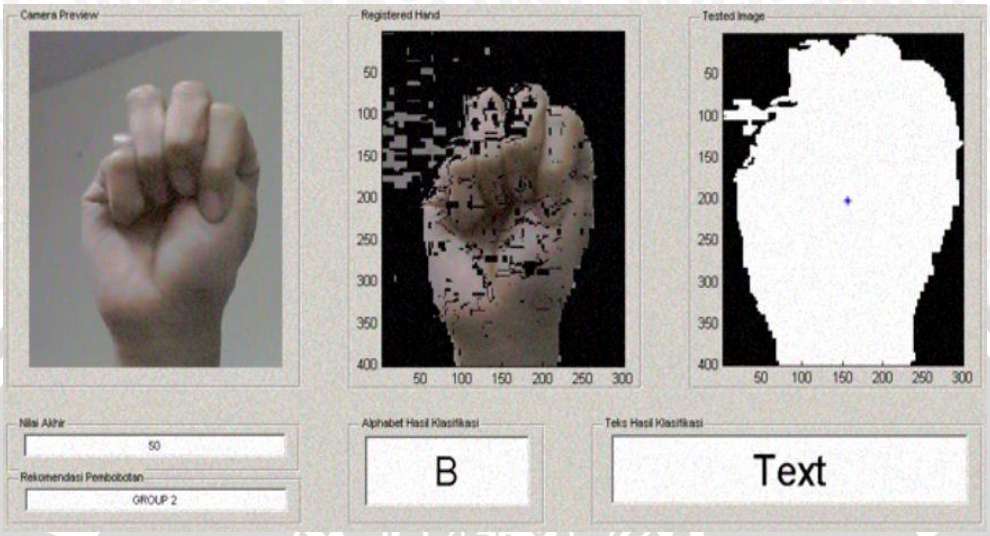
Gambar 6.8 Hasil binerasi deteksi warna kulit sempurna

Berdasarkan data di atas menunjukkan bahwa kulit yang memiliki *Value* rendah di HSV nya, performa segmentasinya tinggi. Juga pengaruh *noise* pada hasil deteksi warna kulit, jika salah area cropping maka menghasilkan klasifikasi yang salah pula. Buktinya akurasi tangan gelap lebih tinggi dari terang. Pada citra 6.3 memiliki warna kulit yang lebih gelap akan menghasilkan hasil ekstraksi warna kulit yang lebih sempurna seperti pada 6.4, tapi salah klasifikasi dikarenakan terdapat *noise* yang membuat *cropping* pada area yang salah yaitu area yang dilingkari, sehingga menghasilkan citra biner seperti pada Gambar 6.5. Sedangkan pada warna kulit yang lebih terang, membuat hasil ekstraksi deteksi warna kulit kurang sempurna seperti pada Gambar 6.7, namun karena berhasil cropping pada area yang benar, maka menghasilkan citra biner 6.8 yang membuat klasifikasinya pun benar.

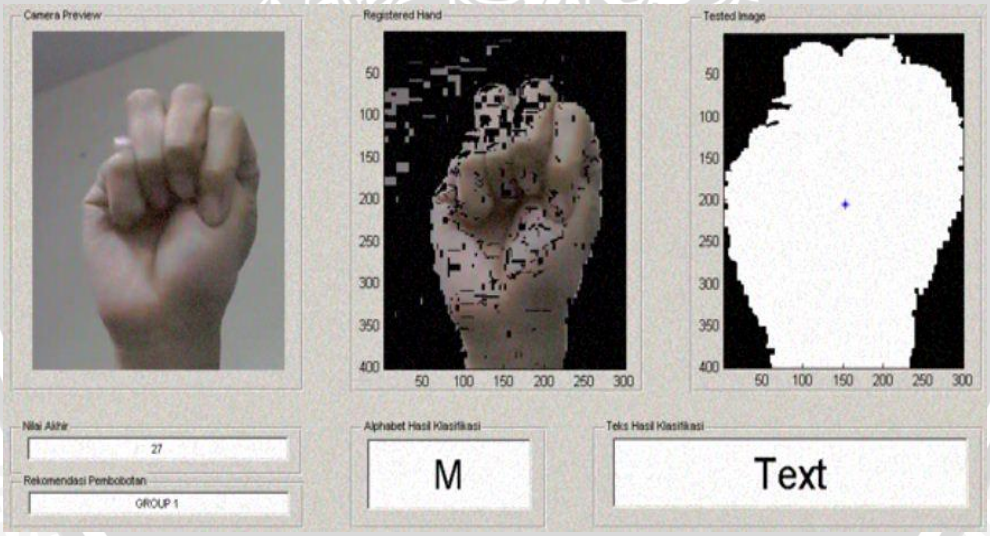
b. Ukuran Tangan

Dalam penelitian ini untuk penentuan nilai utilitas kriterianya menggunakan jumlah biner 1 yang tersebar pada garis ordinat kolom 1 sampai 300 pada baris 75 yang disimpan dalam variabel *gx* dan jumlah biner 1 yang tersebar pada garis ordinat kolom 25 pada baris 1 sampai 400 yang disimpan dalam variabel *gy*. Hasilnya seperti yang tertera di tabel 5.3, 5.4 dan 5.5. Sehingga membuat batasan bahwa citra biner tangan yang diuji harus memiliki lebar dan tinggi yang cukup agar bisa memenuhi syarat untuk dimasukkan ke dalam grup pembobotan yang benar. Karena setiap grup berisikan alphabet-alphabet tertentu yang nantinya

digunakan sebagai data latih. Hal ini yang membuat jika masuk grup pembobotan yang salah, maka hasil klasifikasinya pun salah seperti pada Gambar 6.9.



Gambar 6.9. Rekomendasi grup pembobotan salah



Gambar 6.10 rekomendasi grup pembobotan benar

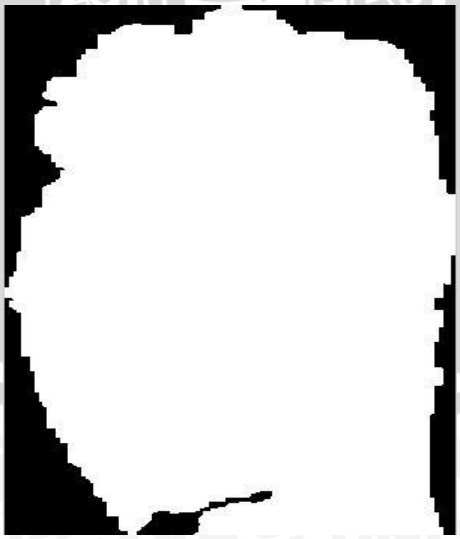
Seperti yang ditunjukkan pada kotak rekomendasi pembobotan pada Gambar 6.9 yang bertuliskan Grup 2, membuat hasil klasifikasinya salah karena menghasilkan alphabet B. Sedangkan yang benar adalah klasifikasinya yang ditunjukkan pada Gambar 6.10. Karena masuk grup 1 sehingga menghasilkan alphabet yang benilai benar, yaitu alphabet M.

Kesalahan dalam pemilihan grup tersebut dikarenakan ukuran jari yang kurang cukup besar pada citra biner di kotak *tested image* Gambar 6.9 yang membuat nilai utilitas kriterianya tidak memenuhi syarat batas jumlah biner 1 *gy* >90 dan jumlah biner 1 *gx* <200. Adapun alphabet yang sering mengalami salah grup pembobotan tersebut adalah alphabet M dan T. Rasio salah pada kedua kondisi tangan berimbang, yaitu masing-masing salah grup 1 kali. Walau demikian kesalahannya sangat kecil.

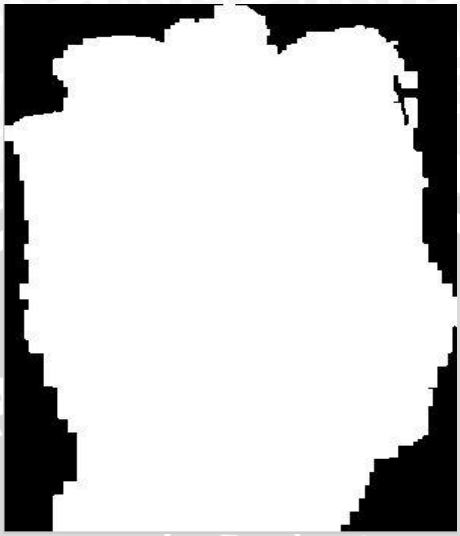
c. Citra Data Latih

Sangat perlu diperhatikan bentuk tangan dalam citra data latih haruslah pada posisi berbeda antara satu alphabet dengan alphabet yang lain. Hal ini dipengaruhi oleh rumus pemilihan fitur yang menitik beratkan pada jarak antara titik tengah tangan ke tepiannya. Dengan fitur tersebut, proses klasifikasi menggunakan metode SMART KNN membuat data uji yang digunakan harus memiliki posisi yang sangat mendekati/ mirip dengan data latih untuk mendapatkan hasil optimalnya.

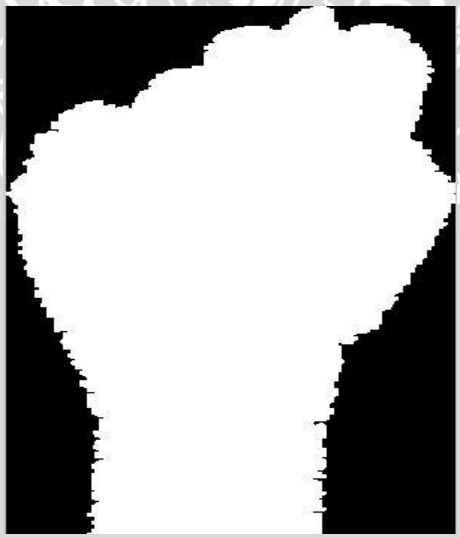
Hal ini dibuktikan dalam klasifikasi beberapa bentuk yang memiliki kemiripan bentuk seperti pada huruf A, E, M, N, O, S dan T. Seperti yang ditunjukkan pada tabel 6.1, saat pengujian alphabet M yang menghasilkan akurasi 60% karena hasil klasifikasinya yang lain menghasilkan alphabet E. Sedangkan pada tabel 6.2, saat pengujian alphabet T yang menghasilkan akurasi 60% karena hasil klasifikasinya yang lain menghasilkan alphabet N.



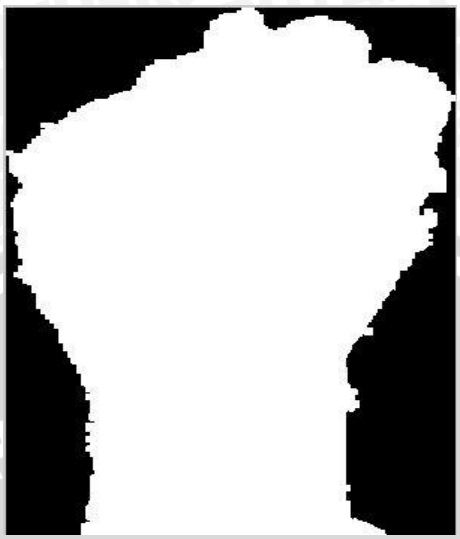
Gambar 6.11 Citra data latih M



Gambar 6.12 Citra data latih E



Gambar 6.13 Data Latih T

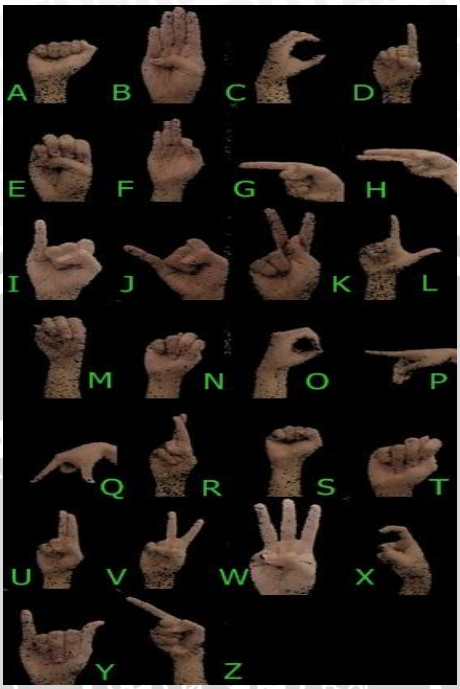


Gambar 6.14 Data Latih N

Pada Gambar 6.11 yang merupakan data latih M menunjukkan kemiripan bentuk dengan data latih E seperti pada Gambar 6.12. Pembeda untuk mengurangi kesalahan dalam proses klasifikasi, saat menguji data M maka harus dipastikan tangan lebih dekat ke kamera sehingga menghasilkan bentuk tangan yang mendekati bentuk data latih M. Sedangkan pada Gambar 6.13 menunjukkan data latih T yang memiliki kemiripan dengan bentuk N seperti pada Gambar 6.14.

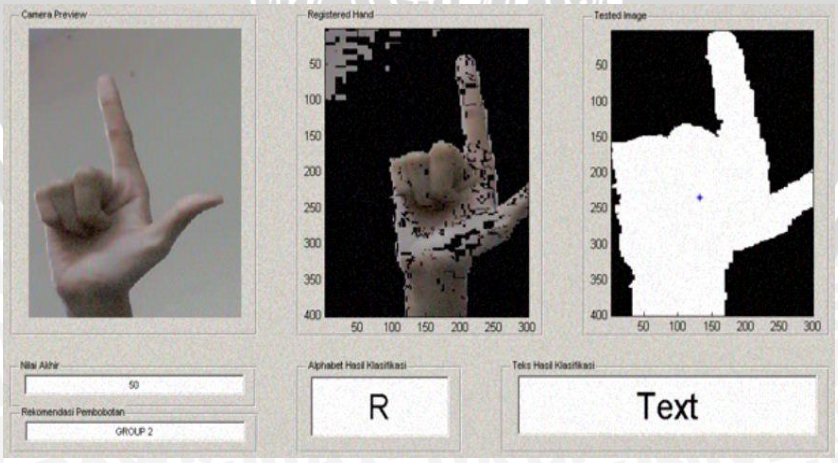
d. Mengikuti Standar Posisi

Berdasarkan tabel rata-rata akurasi pengujian 6.3 yang menghasilkan akurasi 93%, selain karena faktor kecerahan warna kulit, berarti juga dipengaruhi faktor bentuk dan posisi tangan. Sehingga diperlukan standar posisi yang harus diikuti untuk mencapai rata-rata akurasi 93% tersebut seperti yang ditunjukkan pada Gambar 6.15.



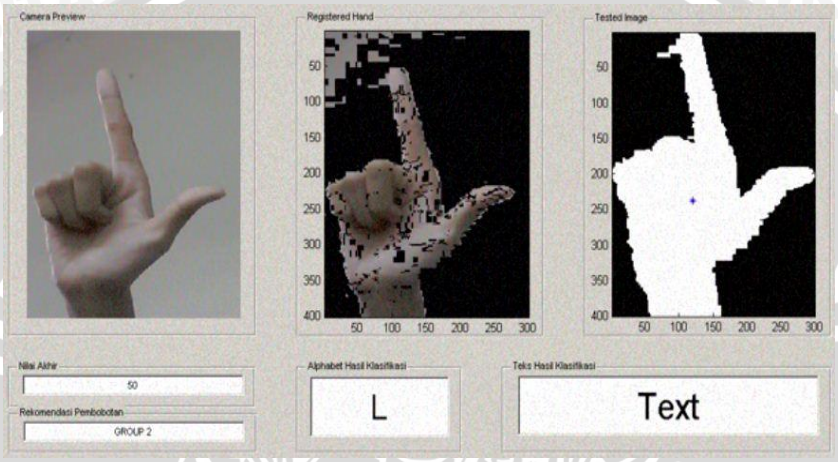
Gambar 6.15 Alphabet bahasa isyarat

Sesuai posisi yang ditunjukkan pada Gambar 6.15, jika Gambar tangan pada standar posisinya mengikutkan lengan, menunjukkan bahwa posisi tangan berada pada jarak yang menjauh dari kamera. Sebaliknya pada Gambar yang menunjukkan hanya area tangan saja, posisi tangan yang ditangkap harus mendekat ke kamera atau menempel pada batas bawah kotak *camera preview* yang telah tersedia di antar muka pengguna. Karea kesalahan memposisikan tangan akan menghasilkan salah klasifikasi seperti pada Gambar 6.16.



Gambar 6.16 Salah posisi tangan

Pada Gambar 6.16 menunjukkan bahwa citra yang tertangkap kamera dan diproses hingga bentuk biner tidak mengikuti standar yang ada. Kesalahan pada citra yang diujikan, akibatnya data uji lebih dekat dengan data latih alphabet R. sehingga menghasilkan klasifikasi alphabet yang seharusnya alphabet L, akhirnya diklasifikasin sebagai alphabet R. Namun akan menghasilkan klasifikasi yang benar jika mengikuti standar posisinya seperti pada Gambar 6.17.



Gambar 6.17 Posisi tangan benar

Jelas pada Gambar 6.17 menunjukkan bahwa jika tangan yang diujikan pada posisi yang benar, maka menghasilkan alphabet klasifikasi benar pula, yaitu alphabet L.

BAB 7 PENUTUP

Berisikan kesimpulan diperoleh dari hasil pengujian dan analisa hingga saran yang perlu ditambahkan untuk penelitian selanjutnya.

7.1. KESIMPULAN

Berdasarkan hasil pengujian dan analisa yang ada, bisa disimpulkan bahwa :

a. Hasil Pengolahan Citra Tangan

Setelah mengikuti tahapan pengolahan citra mulai dari deteksi warna kulit hingga segmentasi, menunjukkan pengaruh kecerahan warna kulit dan pemilihan area *cropping*. Batas ambang deteksi warna kulit menggunakan HSV dan YCrCb pada program simulasi ini, menampilkan hasil terbaik pada tangan yang memiliki warna kulit gelap. Seperti pada Gambar 6.3 dengan warna kulit gelap terbukti menghasilkan citra dengan membedakan tangan dengan latarbelakang yang bersih seperti pada Gambar 6.4. Sedangkan pada Gambar 6.6 yang memiliki warna kulit cerah, akan menghasilkan citra dengan membedakan tangan dengan latarbelakang yang kurang sempurna seperti pada Gambar 6.7. Namun faktor cropping pada area yang salah, akan menghasilkan klasifikasi yang salah pula seperti pada Gambar 6.5. Hal ini dipengaruhi oleh perhitungan jarak antara titik tengah tangan ke tepian data uji yang jauh dari jarak titik tengah ke tepian data latih.

b. Pengaruh Ukuran Tangan

Seperti yang dijelaskan pada analisa pengujian bahwa ukuran tangan yang diuji ikut memberikan pengaruh yang signifikan pada hasil klasifikasi. Ini dikarenakan penentuan nilai utilitas kriteria pembobotan masih menggunakan jumlah biner 1 yang ada pada ordinat gx dan gy. Sehingga jika masuk grup pembobotan yang salah, akan menghasilkan alphabet klasifikasi yang salah pula seperti pada Gambar 6.9. Untuk hasil terbaik, maka tangan yang akan diujikan harus diposisikan sedikit condong atau lebih dekat ke kamera untuk memanipulasi besar telapak tangan dan panjang jari untuk memenuhi syarat penilaian utilitas kriteria pembobotan.

c. Pengaruh Citra Data Latih

Sangat perlu diperhatikan bahwa fitur yang digunakan dalam penelitian ini yaitu menghitung antara titik tengah ke tepian tangan pada citra biner. Sehingga membuat bentuk dan posisi masing-masing tangan yang digunakan sebagai data latih harus berbeda-beda. Hal ini diperlukan untuk memperkecil kesalahan klasifikasi yang dikarenakan kemiripan yang terlalu dekat antar data latih seperti kemiripan bentuk alphabet M dan E pada Gambar 6.11 dan 6.12 atau alphabet T dan N pada Gambar 6.13 dan 6.14.

d. Pengaruh Standar Posisi

Dengan 5 kali tangkapan kamera untuk tiap proses klasifikasi masing-masing alphabet, maka tiap pengujian alphabet harus mengikuti bentuk dan posisi standar posisi seperti pada gambar 6.15 yang sudah ditentukan sampai pengujian saat itu selesai. Standar posisi tersebut juga harus tetap diikuti jika menginginkan hasil klasifikasi yang membentuk teks yang terdiri dari 5 alphabet yang berbeda-beda atau membentuk suatu kata. Standar tersebut dibuat berdasarkan bahwa tiap alphabet diharuskan memiliki bentuk dan posisi yang semirip mungkin dengan citra biner tangan data latih untuk menghindari kesalahan yang terjadi seperti pada gambar 6.16 dan menghasilkan alphabet klasifikasi yang benar seperti gambar 5.17.

Seperti yang dijelaskan dalam analisa, bahwa ada 4 hal yang harus dipenuhi untuk mendapatkan rata-rata akurasi 93%. Diantaranya pengaturan batasan deteksi warna kulit HSV-YCrCb yang tepat untuk proses pengolahan citra tangan, reakayasa posisi tangan agar memiliki jumlah biner 1 yang masuk pada batasan penentuan nilai utilitas kriteria pembobotan, pemilihan citra biner sebagai data latih dan mengikuti bentuk dan posisi sesuai standar yang telah ditentukan dan mempertahankan posisi tangan tersebut hingga proses pengujian selesai.

7.2. SARAN

Adapun saran yang saya ajukan, penelitian ini masih sangat sederhana. Diharapkan kedepan tidak hanya sebatas simulasi, melainkan bisa diterapkan pada perangkat-perangkat yang lain. Mungkin ditambahkan arduino atau diterapkan dalam bentuk aplikasi desktop atau mobile untuk remote yang memberikan perintah untuk menjalankan suatu fungsi. Dalam penentuan nilai utilitas pembobotan SMART masih menggunakan batasan yang sangat sederhana. Sehingga mungkin bisa menerapkan algoritma perhitungan jarak center of mass ke foreground sebagai gantinya. Bahkan mungkin penggunaan metode SMART KNN ini untuk mengklasifikasi dalam penelitian yang lain.

DAFTAR PUSTAKA

[KAS-13] Kasie, Fentahun Moges, "Combining Simple Multiple Attribute Rating Technique and Analytical Hierarchy Process for Designing Multi-Criteria Performance Measurement Framework", Global Journal of Researches in Engineering Industrial Engineering Volume 13 Issue 1 Year 2013

[SIL-11] Silva, L.A. dan Del-Moral-Hernandez, E. 2011. "A SOM combined with KNN for Classification Task". Proceedings of International Joint Conference on Neural Networks, San Jose, California, USA, July 31 – August 5, 2011

[RAH-11] Rahman, A.M., Ahsan-Ul-Ambia, and Aktaruzzaman, M., "Recognition Static Hand Gestures of Alphabet in ASL", Issn 2078-5828 (Print), Issn 2218-5224 (Online), Volume 02, Issue 01, Manuscript Code: 11074 Ijcit 2011

[CHA-12] Chai, J., Liu, J.N.K., and Ngai, E.W.T. 2012. "Application of decision-making techniques in supplier selection: A systematic review of literature", 2012 Elsevier Ltd. All rights reserved

[MNA-13] M., Nachamai, "Alphabet Recognition Of American Sign Language: A Hand Gesture Recognition Approach Using Sift Algorithm", International Journal of Artificial Intelligence & Applications (IJAIA), Vol.4, No.1, January 2013

[YAN-14] Yang, D., Lim, J., and Choi, Y., "Early Childhood Education by Hand Gesture Recognition using a Smartphone based Robot", The 23rd IEEE International Symposium on Robot and Human Interactive Communication August 25-29, 2014. Edinburgh, Scotland, UK

[PUT-10] Putra, Darma. 2010. "Pengolahan Citra Digital", Yogyakarta : Penerbit C.V. Andi Offset

[WIJ-10] Wijanarko, L. 2010. "Cara menghitung ukuran file gambar dan foto digital", diakses pada tanggal 9 agustus 2015 dari http://www.ahlidesain.com/cara-menghitung-ukuran-file-gambar-dan-foto-digital.html

[MIN-13] Minariningtyas, B.A, 2013. "INFORMATIKA : Artikel Teknik Informatika dan Sistem Informasi, Grayscale", diakses pada 9 agustus 2015 dari http://informatika.web.id/grayscale-2.htm

[ALB-01] Albiol, A., Torres, L., and Delp, E.J. 2001. "Optimum Color Spaces For Skin Detection", Image Processing, 2001. Proceedings. 2001 International Conference on (Volume:1)

[MEN-12] Meng, Z.Y., Pan, J., Tseng, K., and Zheng, W.M., "Dominant Points based Hand Finger Counting for Recognition under Skin Color Extraction in Hand Gesture Control System", Sixth International Conference on Genetic and Evolutionary Computing 978-0-7695-4763-3/12 © 2012 IEEE

[PRA-15] Pratiwi, Heny. 2015. "Simple Multi Attribute Rating Technique (SMART)", diakses pada tanggal 1 agustus 2015 dari http://www.henypratiwi.com/2015/04/simple-multy-attribute-rating-technique.html

[SIT-15] Situmenang, Mesdina. 2015. "Perancangan Aplikasi Penilaian Hasil Kinerja Dosen Terbaik Dengan Metode Simple Multi Attribute Rating Technique (Studi Kasus : Akper Yayasan Binalita Sudama Medan", Pelita Informatika Budi Darma, Volume : IX, Nomor: 1, Maret 2015 ISSN : 2301-9425.

[LIN-14] Lin, Yun and Wang, Jie. 2014. "Research on Text Classification Based on SVM-KNN", 978-1-4799-3279-5 /14/$31.00 ©2014 IEEE.

[LIU-14] Liu, Qingfeng and Liu, Chengjun. "A New Locally Linear KNN Method with an Improved Marginal Fisher Analysis for Image Classification", Biometrics (IJCB), IEEE International Joint Conference on Sept. 29 2014-Oct. 2 2014.

[SET-15] Setiarto, Bambang. 2015. "Anak Tuna Rungu", diakses dari website http://tunarungu.com/ tanggal 13 juni 2015.

[DUA-11] Duan, H.X., Zhang, Q. and Ma, W. 2011. "An Approach to Dynamic Hand Gesture Modeling and Real-time Extraction", IEEE 978-1-61284-486-2/111

[REN-11] Ren, Z., Meng, J., and Yuan, J. 2011. "Depth Camera Based Hand Gesture Recognition and its Applications in Human-Computer-Interaction", IEEE 978-1-4577-0031-6/11

[MEL-11] Melo, R.M., Medeiros, D.D., and de Almeida, A.T. 2011. "Selection and Ranking of Improvement Approaches in Construction Companies: SMARTS Method", 978-1-4577-0739-1/11 ©2011 IEEE

[JUA-11] Juan, Li. 2011. "TKNN: an improved KNN algorithm based on tree structure". Seventh International Conference on Computational Intelligence and Security

[CHA-13] Chavda, A., and Thakkar, S. 2013. "A Vision Based Static Hand Gesture Alphabet Recognition", International Journal of Engineering Research & Technology (IJERT) ISSN: 2278-0181 Vol. 2 Issue 4, April – 2013