APLIKASI PENGENALAN RAMBU BERBENTUK BELAH KETUPAT

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana
Jurusan Teknik Informatika

Oleh:
Nama: Andhika Pratama
No. Mahasiswa: 07523318

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA
YOGYAKARTA
2011

ii

LEMBAR PENGESAHAN PEMBIMBING

APLIKASI PENGENALAN RAMBU BERBENTUK BELAH KETUPAT

TUGAS AKHIR

Oleh:
Nama: Andhika Pratama
No. Mahasiswa: 07523318

Yogyakarta, 6 Oktober 2011

Pembimbing

Izzati Muhimmah, ST., M.Sc.

iii

LEMBAR PENGESAHAN PENGUJI

APLIKASI PENGENALAN RAMBU BERBENTUK BELAH KETUPAT
TUGAS AKHIR

Oleh:

Nama : Andhika Pratama
No Mahasiswa : 07523318

Telah Dipertahankan di Depan Sidang Penguji Sebagai Salah Satu Syarat
Untuk Memperoleh Gelar Sarjana Teknik Informatika
Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 24 Oktober 2011

Tim Penguji

Izzati Muhimmah, ST., M.Sc., Ph.D.
Ketua

Zainudin Zukhri, S.T., MIT.
Anggota I

Syarif Hidayat, S.Kom., MIT.
Anggota II

Mengetahui,
Ketua Jurusan Teknik Informatika
Universitas Islam Indonesia

Yudi Prayudi, S.Si., M.Kom.

HALAMAN PERSEMBAHAN

Kupersembahan karya sederhana ini untuk:

Ayahanda IB. DT. Tunaro, S.Pd
Terimakasih Ayah buat doa, dukungan, kasih sayang dan perhatian yang telah diberikan selama ini.

Ibunda Azwarni
Terimakasih Bunda buat doa, dukungan, kasih sayang dan perhatian yang telah diberikan selama ini.

Patri Sandy, Treesa Mulyana, Salwa Naqia Syafani
Terimakasih Adik-adikku yang telah mendukung dan memberikan kecerian buat uda selama ini.

Arvia Nevi Firstiani
Terimakasih *My Beloved* buat doa, dukungan, kasih sayang, perhatian, dan pesan-pesan yang membangun, yang diberikan selama ini.

HALAMAN MOTTO

"Pahlawan bukanlah orang yang berani menetakkan pedangnya ke pundak lawan, tetapi pahlawan sebenarnya ialah orang yang sanggup menguasai diriinya dikala ia marah"

(Nabi Muhammad SAW)

"Sesungguhnya setelah kesulitan tersimpan sebuah kemudahan"
QS. Al-Insyirah : 6

KATA PENGANTAR

Assalamu'alaikum Wr.Wb

Alhamdulillah, segala puji bagi Allah SWT atas segala rahmat, hidayah dan inayah-Nya, sehingga penulisan laporan tugas akhir yang berjudul **Aplikasi Pengenalan Rambu Berbentuk Belah Ketupat** dapat penulis selesaikan dengan baik.

Laporan tugas akhir ini disusun sebagai salah satu syarat guna memperoleh gelar Sarjana Teknik Informatika pada Universitas Islam Indonesia. Dan juga sebagai sarana untuk mempraktekkan secara langsung ilmu dan teori yang telah diperoleh selama menjalani masa studi di Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.

Penyusunan laporan tugas akhir ini tidak lepas dari bimbingan, dukungan dan bantuan baik materiil maupun spirituil dari berbagai pihak. Oleh karena itu dalam kesempatan ini dengan segala kerendahan hati, penulis ingin menyampaikan ucapan terima kasih yang sebesar-besarnya kepada :

a. Allah SWT, Tuhan bagi seluruh alam yang melimpahkan rahmat dan karuniannya sehingga penulis selalu diberi kesehatan dan kemudahan selama pembuatan tugas akhir ini.
b. Ayahanda IB.DT.Tunaro,S.Pd dan Ibunda Azwarni yang telah memberikan seluruh do'a dan restu, serta dorongan baik spiritual maupun materiil sehingga penulis dapat menyelesaikan studi dengan baik.
c. Adik-adikku tercinta yang selalu memberikan semangat dan memberikan inspirasi untuk terus maju.
d. Arvia Nevi Firstiani, yang slalu berikan perhatian, kasih sayang, doa, dan dorongan, serta selalu mengingatkan hal-hal yang baik demi kelancaran penyelasaian tugas akhir ini.
e. Bapak Ir. Gumbolo Hadi Susanto, M.Sc., selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia.

f. Bapak Yudi Prayudi, S.Si., M.Kom, selaku Ketua Jurusan Teknik Informatika.
g. Ibu Izzati Muhimmah, ST., M.Sc., Ph.D., selaku dosen pembimbing yang telah memberikan pengarahan, bimbingan, serta masukan selama pelaksanaan tugas akhir dan penulisan laporan.
h. Dosen-dosen Jurusan Teknik Informatika. Terima kasih atas semua ilmu pengetahuan dan motivasi serta bantuannya.
i. Seluruh teman-teman Laboratorium Informatika Terpadu Universitas Islam Indonesia, khususnya Laboratorium Komputasi Sistem Cerdas.
j. Sahabat-sahabatku yang jauh disana dan selalu mendoakanku, terima kasih atas semuanya. Semoga kebaikan kalian selama ini dapat dibalas oleh Allah SWT. Amin.
k. Serta semua pihak yang memberikan dukungan, yang tidak dapat saya sebutkan satu per satu.

Terima kasih kepada semua pihak yang telah membantu terselesaikannya penulisan laporan tugas akhir ini Semoga Allah SWT melimpahkan rahmat dan hidayahnya dan membalas semua kebaikan kalian.

Penulis menyadari bahwa dalam penyusunan laporan tugas akhir ini masih banyak terdapat kekeliruan dan kekurangan. Untuk itu penulis menyampaikan permohonan maaf sebelumnya serta sangat diharapkan kritik dan saran yang sifatnya membangun untuk penyempurnaan di masa mendatang.

Akhir kata semoga laporan ini dapat bermanfaat bagi penulis dan semua pembaca.

Wassalamu'alaikum Wr.Wb.

Yogyakarta, 10 Oktober 2011

Andhika Pratama

SARI

Rambu lalu lintas adalah salah satu alat perlengkapan jalan dalam bentuk tertentu yang memuat lambang, huruf, angka, kalimat atau perpaduan di antaranya. Berdasarkan jenis pesan yang disampaikan, rambu lalu lintas dapat dikelompokkan menjadi beberapa jenis seperti rambu peringatan, larangan, dan perintah. Jenis rambu tersebut dibedakan berdasarkan bentuknya, seperti rambu peringatan yang memiliki bentuk belah ketupat. Aplikasi ini perlu untuk dapat membedakan rambu berbentuk belah ketupat dengan bentuk lainya. Oleh karena itu pada penelitian ini akan dibangun sebuah aplikasi yang dapat mengenali rambu dengan bentuk belah ketupat. Metode yang digunakan adalah *canny edge detection* untuk proses deteksi tepi, dan *diagonal neighbors* untuk proses filter. Metode-metode yang digunakan ini diharapkan dapat mengenali rambu berbentuk belah ketupat dan untuk mengetahui apakah dapat mengenali dengan baik maka sistem akan melalui beberapa pengujian.

Hasil dari penelitian skripsi ini adalah berhasil membuat aplikasi pengenalan rambu berbentuk belah ketupat. Sistem yang dibangun dapat membantu mengenali rambu dan mengetahui adanya rambu yang berbentuk belah ketupat. Dari hasil pengujian sistem diperoleh hasil bahwa proses pengenalan rambu berbentuk belah ketupat dengan metode *diagonal neighbors* telah tepat, sehingga sistem tidak dapat mengenali rambu dengan bentuk lain.

Kata kunci : *Canny edge detection, Diagonal neighbors, Rambu, Belah ketupat*

TAKARIR

Accesibility	:	Tingkat akses.
Button	:	Tombol yang digunakan untuk pilihan
Canny Edge Detection	:	Salah satu metode deteksi tepi.
Citra	:	Gambar.
Error	:	Kesalahan.
FileChooser	:	Form untuk pengimputan file.
Filter	:	Saringan.
Flow Chart	:	Diagram alur
GPS	:	Alat untuk mengetahui posisi dan navigasi.
Hardware	:	Perangkat keras.
Image Processing	:	Ilmu tentang proses pengolahan citra atau gambar.
Input	:	Masukkan data untuk diproses.
Interface	:	Tampilan dari sebuah sistem.
Morfologi	:	Proses pendeteksian pola dengan penggabungan dua buah *filter*.
Netbeans	:	Software develop *java*.
Noise	:	Derau.
Output	:	Hasil keluaran dari sistem.
Processor	:	Otak dari komputer
Recognition	:	Pengenalan.
Sensitifitas	:	Tingkat kesensitifan.
Software	:	Perangkat lunak.
Stabilitas	:	Tingkat kestabilan.
Tool	:	Perangkat.
Value	:	Nilai.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pengolahan citra digital merupakan proses yang bertujuan untuk mengolah suatu objek citra dengan cara mengekstraksi informasi penting yang terdapat di dalamnya. Dari informasi tersebut dapat dilakukan proses analisis dan klasifikasi secara cepat memanfaatkan algoritma perhitungan komputer. Dari pengolahan citra diharapkan terbentuk suatu sistem yang dapat memproses citra masukan hingga citra tersebut dapat dikenali cirinya. Pengenalan ciri inilah yang sering diaplikasikan dalam kehidupan sehari-hari dan dapat diimplementasikan dalam berbagai bidang salah satunya lalu lintas.

Salah satu penerapan pengolahan citra di bidang lalu lintas yaitu pengenalan rambu. Rambu lalu lintas dapat dibedakan berdasarkan pesan yang disampaikan seperti rambu peringatan, larangan, dan perintah. Berdasarkan jenis pesan tersebut, rambu lalu lintas memiliki bentuk yang spesifik misalnya rambu larangan yang memiliki bentuk belah ketupat. Sedangkan untuk pesan lainnya memiliki bentuk seperti bulat, segi tiga, dan lain-lain. Untuk dapat membedakan antara rambu yang memiliki bentuk belah ketupat dengan bentuk lainnya, maka pada penelitian ini akan dibangun sebuah *tool* yang dapat mengenali rambu dengan pola berbentuk belah ketupat. Sistem ini disebut *Rhombus Sign Recognition System* yang dibangun dengan menerapkan metode dalam ilmu pengolahan citra seperti proses deteksi tepi dan proses filter untuk proses pengenalan bentuk objek belah ketupat.

1.2 Rumusan Masalah

Untuk memperjelas permasalahan yang akan diteliti, maka masalah tersebut dirumuskan sebagai berikut :

1. Bagai mana cara menentukan pola rambu yang berbentuk belah ketupat?
2. Bagaimana cara mengenali letak bagian dari rambu pada sebuah *image* menggunakan metode *canny edge detection*?

1.3 Batasan Masalah

Batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Untuk sampel pada penelitian ini berupa citra yang terdapat objek rambu didalamnya.
2. Hanya mengenali rambu dengan pola belah ketupat.
3. Sudut pandang pengambilan sampel hanya dari depan.
4. Untuk kondisi pengambilan sampel hanya satu kondisi yaitu di siang hari yang cerah.
5. Data berupa image dengan format JPG.
6. Aplikasi ini merupakan aplikasi dekstop.

1.4 Tujuan Penelitian

Tujuan yang diharapkan dari penelitian ini adalah:

1. Membangun aplikasi untuk mengetahui rambu yang memiliki pola berbentuk belah ketupat dari sebuah citra dengan menerapkan proses pengolahan citra.
2. Memanfaatkan dan mempelajari metode-metode pengolahan citra dalam pembuatan tools pengenalan objek, menggunakan pemrograman utama yaitu *Java 2 Platform Standard Edition* (*J2SE*).

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat bermanfaat:

1. Bagi peneliti sebagai dorongan untuk meningkatkan kemampuan dan lebih memahami metode metode yang ada dalam proses pengolahan citra.
2. Bagi mahasiswa sebagai panduan penelitian selanjutnya misalnya untuk proses pengenalan pola yang lebih rumit.
3. Bagi Masyarakat umum sebagai alat untuk mengetahui dan mengenali rambu sebagai panduan untuk berkendaraan.

1.6 Metodologi Penelitian

Dalam suatu penelitian, diperlukan metodologi agar data yang diperlukan dalam penelitian sesuai dengan data yang ada di lapangan.

1.6.1 Lokasi Penelitian

Penelitian ini dilakukan di beberapa tempat tertentu yang menjadi tempat pengambilan sampel, khususnya di daerah kota Yogyakarta.

1.6.2 Alat yang Digunakan

Dalam penelitian ini alat yang digunakan yaitu kamera digital yang berfungsi untuk pengambilan data sampel dan komputer untuk proses pengolahan data dan komputerisasi. Kamera yang digunakan pada penelitian ini yaitu kamera DSLR Canon 1000D dengan lensa Canon EF-S18-55mm f/3.5-5.6 IS.

1.6.3 Sampel

Sampel pada penelitian ini sebanyak 8 sampel dimana 5 sampel yang digunakan untuk menguji dan 3 merupakan sampel untuk kasus.

1.6.4 Pengumpulan Data

Data yang di kumpulkan merupakan citra dengan format JPG dengan jarak pengambilan data citra sejauh 5 meter dari objek rambu dengan arah dari depan.

1.6.5 Pengolahan Data

Data diolah menggunakan *tool* yang dibuat menggunakan bahasa pemprograman *java.* Langkah awal yang dilakukan yaitu membaca atau melakukan deteksi tepi. Setelah melakukan deteksi tepi akan dilakukan proses filter untuk mengurangi piksel yang tidak dibutuhkan. Dan proses *morfologi* yang bertujuan untuk membuat garis pendeteksi dari objek pada citra menggunakan fungsi *graphics2D*.

1.6.6 Analisis Data

Analisis data dilakukan berdasarkan pada metode yang dilakukan dalam proses pendeteksian sampel. Dalam proses pendeteksian sampel akan dianalisis dari pola yang berbentuk belah ketupat. Metode yang digunakan yaitu *Canny edge detection.* dan dalam proses penentuan polanya akan menggunakan *morfologi* dan *filtering* menggunakan konsep *diagonal neighbors*. Dari metode ini diharapkan dapat mengenali rambu berbentuk belah ketupat dengan baik dan akurat.

1.7 Sistematika Penulisan

Dalam penyusunan laporan Tugas Akhir, disusun per bab dan berurutan untuk mempermudah pembahasannya. Secara garis besar, sistematika dari penulisan terdiri atas lima bab, yaitu:

BAB I PENDAHULUAN

Bab ini berisi pembahasan masalah umum dari Sistem Pengenalan Rambu Berbentuk Belah Ketupat yang meliputi latar belakang masalah, rumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, metodologi penelitian dan sistematika penulisan.

BAB II LANDASAN TEORI

Bagian ini memuat landasan teori yang berfungsi sebagai sumber atau alat dalam memahami permasalahan yang berkaitan dengan pendeteksian rambu lalu lintas.

BAB III METODOLOGI

Bagian ini memuat uraian tentang metodologi analisis, analisis kebutuhan dari sistem, perancangan perangkat lunak, dan perancangan digram alir (*flowchart*) dari aplikasi pengenalan rambu berbentuk belah ketupat sesuai dengan proses dan metode yang digunakan dalam pengenalan pola rambu.

BAB IV HASIL DAN PEMBAHASAN

Pada bab ini membahas mengenai implementasi sistem yang meliputi penjelasan dari aplikasi pengenalan rambu berbentuk belah ketupat, perangkat keras, perangkat lunak, proses pendeteksian pola rambu, dan implementasi antar muka.

BAB V PENGUJIAN

Untuk proses pengujian dibahas pada bab ini yang mana meliputi pengujian terhadap sampel data, pengujian beban komputasi, pengujian stability, penanganan kesalahan pada sistem, permasalahan yang belum terselesaikan, serta kelebihan dan kekurangan dari sistem.

BAB VI KESIMPULAN DAN SARAN

Berisi kesimpulan-kesimpulan yang merupakan rangkuman dari hasil analisis kinerja pada bagian sebelumnya dan saran yang perlu diperhatikan berdasarkan keterbatasan yang ditemukan dan asumsi-asumsi yang dibuat selama pembuatan *tool* untuk pengenalan rambu dengan pola belah ketupat.

BAB II
LANDASAN TEORI

2.1 *Image Processing*

Image processing adalah suatu metode yang digunakan untuk memproses atau memanipulasi gambar dalam bentuk 2 dimensi (Gonzalez, 2002). *Image processing* dapat juga dikatakan semua operasi yang bertujuan untuk memperbaiki, menganalisa, atau mengubah suatu gambar. Konsep dasar pemrosesan suatu objek pada gambar menggunakan *image processing* diambil dari kemampuan indera penglihatan manusia yang selanjutnya dihubungkan dengan kemampuan otak manusia.

Dalam sejarahnya, *image processing* telah diaplikasikan dalam berbagai bentuk, dengan tingkat kesuksesan cukup besar. Seperti berbagai cabang ilmu lainnya, *image processing* juga berhubungan dengan cabang-cabang ilmu lainnya, diantaranya adalah optik, elektronik, matematika, fotografi, dan teknologi komputer.

Pada umumnya, objektifitas dari *image processing* adalah mentransformasikan atau menganalisis suatu gambar sehingga informasi baru tentang gambar dibuat lebih jelas.

Dilihat dari proses, *image processing* terbagi menjadi 3 tipe yaitu:

1. *Low-level process*: proses-proses yang berhubungan dengan operasi primitif seperti *image pre-processing* untuk mengurangi *noise*, menambah kontras dan menajamkan gambar. Pada *low-level process*, *input* dan *output*-nya berupa gambar.
2. *Mid-level process*: proses-proses yang berhubungan dengan tugas-tugas seperti segmentasi gambar (membagi gambar menjadi objek-objek), pengenalan (*recognition*) suatu objek individu. Pada *mid-level process*, *input* pada umumnya berupa gambar tetapi *output*-nya berupa atribut yang dihasilkan dari proses yang dilakukan gambar tersebut seperti garis, garis *contour*, dan objek-objek individu.

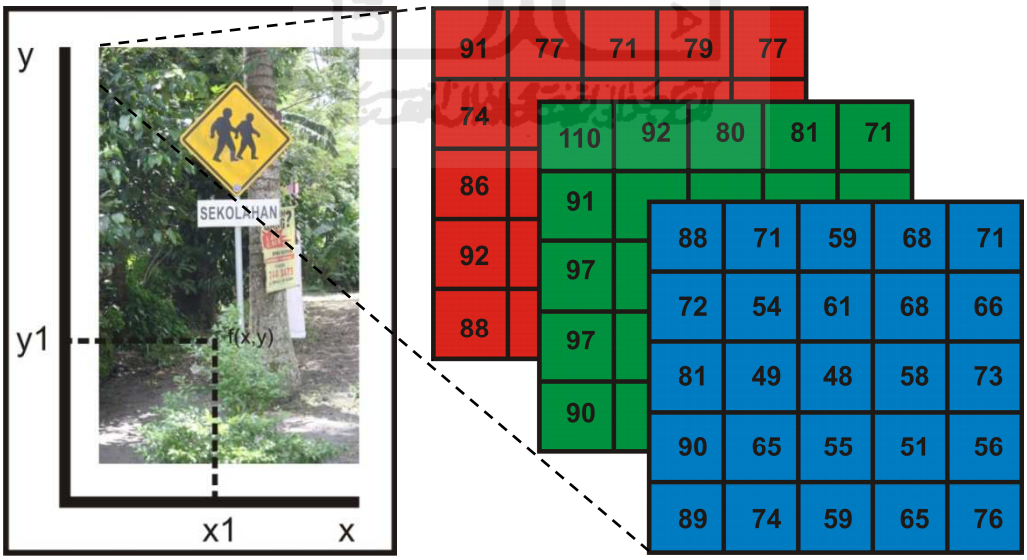
3. *High-level process*: proses-proses yang berhubungan dengan analisis citra yang menghasilkan informasi suatu data citra. Beberapa hal yang dapat diidentifikasi dari sebuah citra seperti format file citra, ukuran file citra, jumlah pixel, dimensi citra, resolusi citra, dan lain-lain.

2.2 Citra Digital

Citra digital dapat didefinisikan sebagai fungsi dua variabel, *f(x,y)*, dimana *x* dan *y* adalah koordinat spasial, dan nilai *f(x,y)* adalah intensitas citra pada koordinat tersebut, hal tersebut diilustrasikan pada Gambar 2.1. Citra Digital tersusun dalam bentuk *grid*. Setiap kotak (*tile*) yang terbentuk disebut piksel (*picture element*) dan memiliki koordinat(x,y)

1. Sumbu x (horisontal): kolom
2. Sumbu y (vertikal): baris

Setiap piksel memiliki nilai (*value* atau *number*) yang menunjukkan intensitas keabuan pada piksel tersebut.



A B

Gambar 2.1 Citra Digital

Operasi-operasi yang dilakukan di dalam pengolahan citra banyak ragamnya. Namun, secara umum operasi pengolahan citra dapat diklasifikasikan dalam beberapa jenis sebagai berikut:

1. Perbaikan kualitas citra (*image enhancement*).

   Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat di dalam citra lebih ditonjolkan.Contoh-contoh operasi perbaikan citra:

   a.perbaikan kontras gelap/terang

   b. perbaikan tepian objek (*edge enhancement*)

   c. penajaman (*sharpening*)

   d. pemberian warna semu (*pseudocoloring*)

   e. penapisan derau (*noise filtering*)

2. Segmentasi citra (*image segmentation*).

Jenis operasi ini bertujuan untuk memecah suatu citra ke dalam beberapa segmen dengan suatu kriteria tertentu, seperti memisahkan obyek dari latar belakangnya. Jenis operasi ini berkaitan erat dengan pengenalan pola.

3. Ekstraksi citra (*image analysis*)

   Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan deskripsinya. Teknik pengorakan citra mengekstraksi ciri- ciri tertentu yang membantu dalam identifikasi objek. Proses segmentasi terkadang diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya.

   Contoh-contoh operasi pengorakan citra:

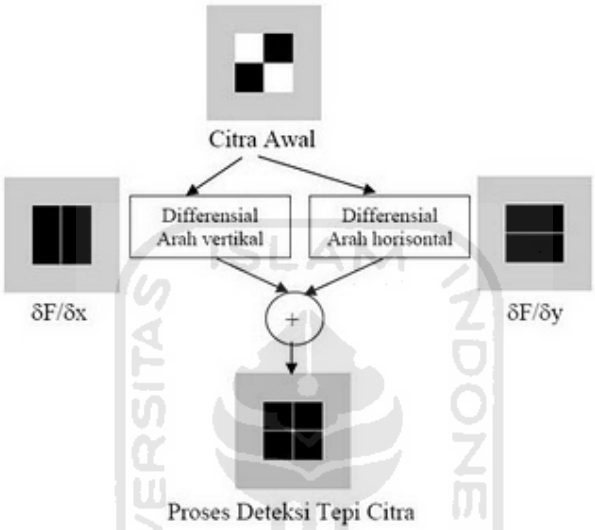
a. Pendeteksian tepi objek (*edge detection*)

   Deteksi tepi (*Edge Detection*) pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, tujuannya adalah (Joe, 2009):

   a. Untuk menandai bagian yang menjadi detail citra.

b. Untuk memperbaiki detail dari citra yang kabur, yang terjadi karena *error* atau adanya efek dari proses akuisisi citra.

Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Pada Gambar 2.2 digambarkan bagaimana tepi suatu gambar diperoleh.



Gambar 2.2 Proses Deteksi Tepi

Canny Edge Detection adalah salah satu algoritma deteksi tepi modern yang memiliki beberapa kriteria pendeteksi tepian paling optimum (Nixon dan Aguado,2002):

a. Mendeteksi dengan baik (kriteria deteksi)

Kemampuan untuk meletakkan dan menandai semua tepi yang ada sesuai dengan pemilihan parameter-parameter konvolusi yang dilakukan. Sekaligus juga memberikan fleksibilitas yang sangat tinggi dalam hal menentukan tingkat deteksi ketebalan tepi sesuai yang diinginkan.

b. Melokalisasi dengan baik (kriteria lokalisasi)

Dengan Canny dimungkinkan dihasilkan jarak yang minimum antara tepi yang dideteksi dengan tepi yang asli. Sedangkan menggunakan operator lain

menghasilkan jarak yang tidak teratur dengan tepi asli dari sebuah objek pada citra. Hal ini disebabkan karena proses pengecekan terhadap piksel tidak baik.

c. Respon yang jelas (kriteria respon)

Hanya ada satu respon untuk tiap tepi. Sehingga mudah dideteksi dan tidak menimbulkan kerancuan pada pengolahan citra selanjutnya. Untuk mengakomodasi kriteria-kriteria tersebut di atas, (Canny,1986) menambahkan pula prosedur-prosedur perbaikan sebelum dan sesudah pendeteksian tepi (*pre dan post processing*) agar hasil deteksi tepi yang diperoleh menjadi lebih baik.

Pre dan *post processing* yang dilakukan pada deteksi tepi metode *Canny* menyangkut (Nixon dan Aguado, 2002):

1. *Smoothing (preprocessing);*
2. *Non maximum suppresion (post-processing);*
3. *Hysteresis thresholding (post-processing).*

Proses *smoothing* dilakukan untuk menghilangkan derau dan menurunkan pengaruh tekstur pada citra sehingga diperoleh hasil deteksi yang lebih baik. Pada metode *Canny,* digunakan *filter gaussian* dalam bentuk matriks *template* yang merupakan bobot (weight) dalam perhitungan nilai rata-rata suatu kelompok piksel pada citra input yang diantaranya berukuran 3x3.

Tabel 2.1 Matriks template dari *filter Gaussian*

0.37	0.61	0.37
0.61	1	0.61
0.37	0.61	0.37

Nilai matriks *template* pada Tabel 2.1 di atas diperoleh dari persamaan (2.1) yang merupakan fungsi sebaran normal *gauss g* pada koordinat x, y dimana besarnya nilai elemen matriks template ditentukan oleh nilai σ_2 [Nixon dan aguado, 2002].

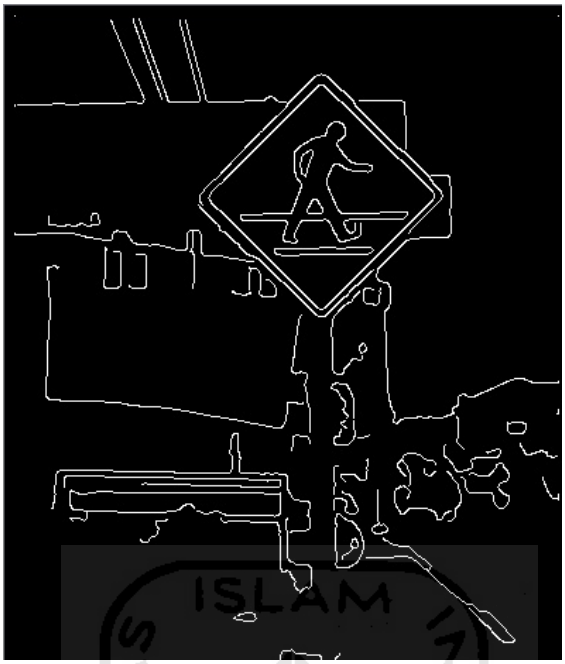
$$g(x,y) = e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \quad \text{(P2.1)}$$

Proses Non Maximum Suppression yang mirip dengan proses *thinning* (perampingan) dilakukan untuk menentukan piksel tepi dengan posisi paling mendekati lokasi terjadinya perubahan nilai piksel diantara banyaknya piksel tepi yang terdeteksi. Dimana pada umumnya, perubahan nilai piksel berada pada pusat kumpulan piksel tepi [Nixon dan Aguado, 2002].

Penentuan pusat kumpulan piksel tepi diantaranya dengan penghitungan jarak *Euclides* antara setiap piksel tepi p(x,y) ke piksel bukan tepi q(s,t) yang memiliki bentuk persamaan (2.2)

$$D = ([x-s]^2 + [y-t]^2)^2 \quad \text{(P2.2)}$$

Berbeda dengan metode thinning, pada proses *Non Maximum suppresion*, pengubahan menjadi citra biner tersebut menggunakan dua nilai *threshold T1* dan *T2* dimana *T1 > T2* yang sering disebut juga *hysteresis thresholding* [Nixon dan Aguado,2002]. Setiap piksel tepi dengan nilai lebih besar dari T1 dipertahankan sebagai piksel tepi. Proses inilah yang membedakan metode canny dengan metode-metode lainnya. Seperti *Proses Non Maximum Suppression* yang hanya dimiliki oleh metode canny yang berfungsi untuk mempertipis garis setiap tepi yang terdeteksi. Sedangkan hasil dari metode lain akan menghasilkan garis yang tebal yang membuat proses deteksi tepi tidak maksimal. Hasil dari rangkaian proses deteksi tepi dengan metode *Canny* pada suatu *image* adalah citra biner yang terdiri dari piksel-piksel tepi tunggal seperti pada Gambar 2.3.



Gambar 2.3 Hasil Deteksi Tepi dengan T1 = 18 dan T2 = 0.010

b. Ekstraksi batas (*boundary*)

Dalam proses pengenalan objek dibutuhkan ekstraksi batas yang sangat halus. untuk memaksimalkan ekstraksi batas maka diperlukan proses pengikisan atau penipisan dari tepi garis. Hal ini sangat sesuai dengan metode deteksi tepi yang digunakan karena pada deteksi objek dengan metode canny terdapat proses penipisan tepi atau disebut juga dengan *Non Maximum suppretion*. Hal ini dipengaruhi oleh *threshold* dimana nilai *threshold* ini akan membantu mengidentifikasi lokasi piksel dengan intensitas tinggi yang juga disebut tepi. Hasil dari proses ini adalah batas gambar yang diekstraksi dua atau lebih piksel tebal gambar.

2.3 Citra Biner

Citra biner adalah gambar dimana nilai dari setiap pikselnya hanya memiliki dua buah nilai intensitas. Yaitu 0 dan 1, dimana 0 menyatakan warna latar belakang (*background*) dan 1 menyatakan warna garis atau tinta (*foreground*). Karena pada citra biner hanya memiliki dua buah nilai intensitas

maka warna pada citra biner juga hanya terdiri dari dua buah warna yang biasanya direpresentasikan dengan warna hitam dan putih. Jadi nilai 0 menyatakan warna hitam dan 1 menyatakan warna putih. Contohnya dapat dilihat pada Gambar 2.4.

Gambar 2.4 Citra Biner

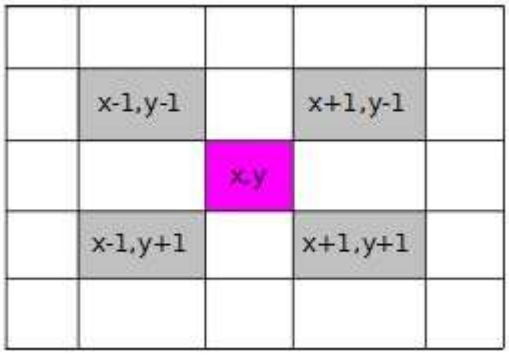
2.4 *Filtering*

Filtering adalah suatu proses dimana diambil sebagian sinyal dari frekwensi tertentu dan membuang sinyal frekwensi yang tidak dibutuhkan. Tujuannya untuk perbaikan kualitas citra, seperti menghaluskan dan menghilangkan *noise* yang ada pada citra, baik secara linear maupun secara non-linear. Pada penelitian ini proses filtering menerapkan konsep diagonal neighbors dimana dilakukan pengecekan terhadap piksel-piksel tetangganya secara diagonal.

a. *Diagonal Neighbors*

Pada jenis ini dinotasikan dengan $N_D(p)$. *Diagonal neighbors* juga memiliki 4 piksel tetangga pada kasus pengenalan rambu belah ketupat. Koordinatnya dapat ditulis dengan:

$$\mathbf{(x-1,y-1),(x+1,y-1),(x-1,y+1),(x+1,y+1).} \quad \text{(P2.3)}$$



Gambar 2.5 Diagonal neighbors

b. Konvolusi

Konvolusi (convolution) adalah sebuah proses dimana citra dimanipulasi dengan menggunakan eksternal mask / subwindows untuk menghasilkan citra yang baru. Konvolusi sangat banyak dipergunakan dalam pengolahan citra untuk memperhalus (smoothing), menajamkan (crispening), mendeteksi tepi (edge detection), dan lain-lain. Konvolusi dapat didefinisikan sebagai perkalian dari dua buah fungsi yaitu f(x) dan g(x).

$$h(x) = f(x) * g(x) \quad \text{(P2.4)}$$

Pada operasi konvolusi pada persamaan 2.4, g(x) disebut kernel konvolusi atau filter. Kernel g(x) merupakan suatu jendela yang dioperasikan secara bergeser pada sinyal masukan f(x), yang dalam hal ini, jumlah perkalian kedua fungsi setiap titik merupakan hasil konvolusi yang dinyatakan dengan keluaran h(x).

2.5 Operasi *Morfologi*

Operasi morfologi merupakan proses pendeteksian pola berbentuk belah ketupat dengan menggunakan penggabungan dua buah *filter* yang melakukan pengecekan terhadap piksel tetangganya secara diagonal. *Filter* tersebut yaitu:

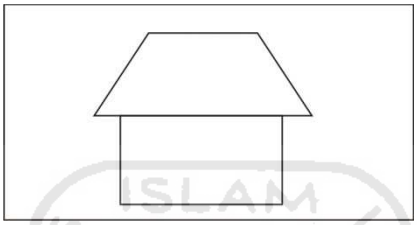
F1 : [1 0 0; 0 1 0; 0 0 1]

F2 : [0 0 1; 0 1 0; 1 0 0]

Dari dua buah *filter* tersebut dilakukan proses penggabungan dengan formula.

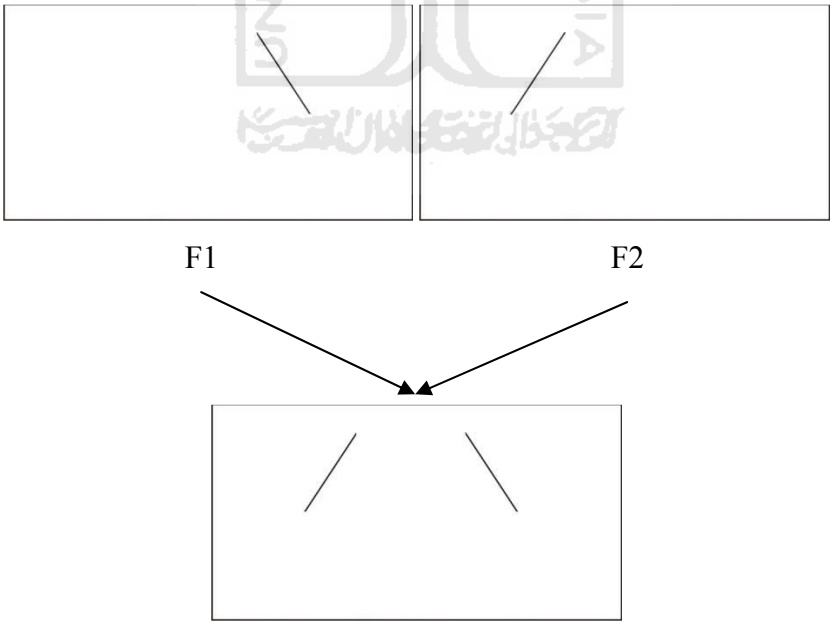
$$\mathbf{H = F1 \cup F2} \qquad \text{(P2.5)}$$

Dari rumus diatas dapat kita representasikan pada sebuah objek gambar dan hasil dari penggabungan kedua buah filter tersebut.



Gambar 2.6 Citra objek

Apabila gambar diatas diterapkan operasi morfologi maka akan menghasilkan gambar seperti dibawah ini.



Gambar 2.7 Proses penggabungan F1 dan F2

Dari Gambar 2.7 dapat dilihat tahapan dari proses morfologi yang menerapkan filtering menggunakan diagonal neighbors. Gambar F1 merupakan hasil dari filter pertama yaitu F1 = [1 0 0; 0 1 0; 0 0 1] dan gambar F2 merupakan hasil dari filer kedua yaitu F2 = [0 0 1; 0 1 0; 1 0 0]. Setelah digabungkan maka akan terbentuk seperti pada gambar F1 ∪ F2. Pada algoritma pemprograman untuk melakukan penggabungan digunakan operator || (atau).

BAB III
METODOLOGI

3.1 Metodologi Analisis

Metode Analisis merupakan sebuah metode untuk menjabarkan aplikasi berdasarkan komponen-komponen dan berbagai fungsi yang bertujuan untuk mengidentifikasi dan mengevaluasi permasalahan yang terdapat pada sistem. Tahap analisis ini merupakan tahapan yang paling penting dalam program yang dirancang, karena jika terjadi kesalahan dalam tahap ini akan menyebabkan terjadinya kesalahan pada tahap selanjutnya. Karena itu dibutuhkan suatu metode sebagai pedoman dalam mengembangkan sistem yang dibangun.

3.2 Analisis Kebutuhan

3.2.1 Analisis Kebutuhan Input

Input adalah suatu data masukan yang akan diolah sehingga menghasilkan sebuah informasi yang dapat diambil atau disimpulkan. Kebutuhan input aplikasi ini yaitu:

1. Tipe data dari citra yang akan diproses harus bertipe JPG
2. Ukuran dari data citra maksimal berukuran 1024 X 768 piksel.

3.2.2 Analisis Kebutuhan Fungsi dan Kinerja

Fungsi dan kinerja yang dibutuhkan pada aplikasi ini adalah :

1. Proses menginputkan gambar kedalam sistem untuk di proses.
2. Proses *edge detection*. Digunakan metode *Canny edge detection.*
3. Proses *filtering* terhadap matrik *citra* yang dihasilkan dari proses sebelumnya yaitu *edge detection* menggunakan metode *Canny edge detection.*
4. Proses penukaran warna sesuai dengan indeks dari matrik hasil *filtering* yang berguna untuk pendeteksi objek rambu pada *citra*.

3.2.3 Analisis Kebutuhan *Output*

Output dari aplikasi ini adalah berupa *citra* yang sudah terdeteksi letak dari objek rambu dengan tanda garis berwarna merah pada setiap tepi objek rambu. Garis merah tersebut berguna sebagai penanda bahwa objek tersebut merupakan objek yang telah dikenal sebagai rambu yang berbentuk belah ketupat.

3.2.4 Analisis Kebutuhan Antarmuka

Antar muka yang dibutuhkan dalam sistem ini adalah:

1. Antarmuka Halaman Utama.
2. Antarmuka *FileChooser.*
3. Antarmuka Hasil.
4. Antarmuka Halaman Bantuan.
5. Antarmuka Halaman Keterangan.

3.3 Perancangan Perangkat Lunak

3.3.1 Metode Perancangan

Dalam perancangan sistem ini, proses pembangunan aplikasi ini digambarkan dengan diagram Alir (*Flow Chart*), menunjukan alur yang dilakukan dalam pembuatan program hingga akhirnya menjadi aplikasi ini. Proses jalannya aplikasi ini digambarkan dalam *flowchart* secara umum dan secara detail untuk memudahkan *user* memahami penggunaan aplikasi ini.

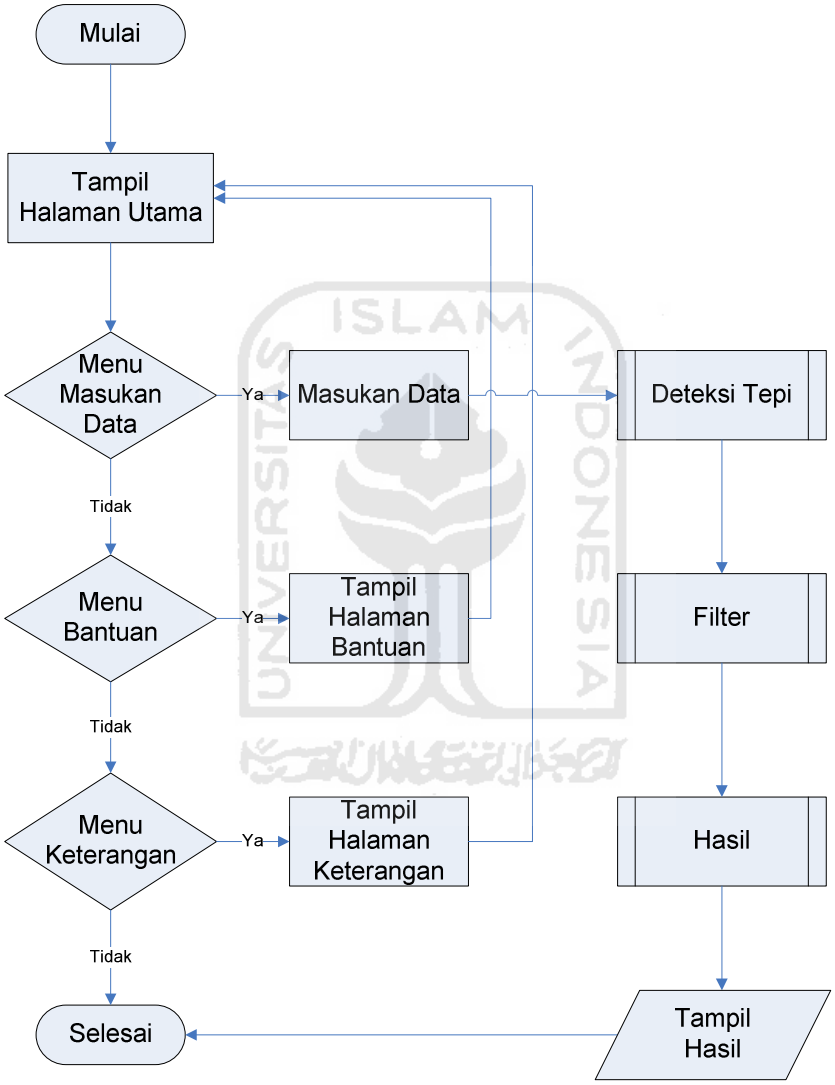
3.3.2 Hasil Perancangan

Hasil pada tahap perancangan sangat berkaitan erat dengan hasil tahap analisis. Karena pada tahap analisis telah ditemukan metode, fungsi-fungsi yang digunakan, perangkat lunak yang digunakan, serta antarmuka yang diharapkan.

3.4 Perancangan Diagram Alir (Flow Chart)

3.4.1 Rancangan Umum

Diagram alir digunakan untuk menggambarkan langkah-langkah kerja dari sistem yang akan dibuat. Untuk gambaran umum dari proses pada aplikasi ini dapat dilihat dari *flow chart* berikut ini.

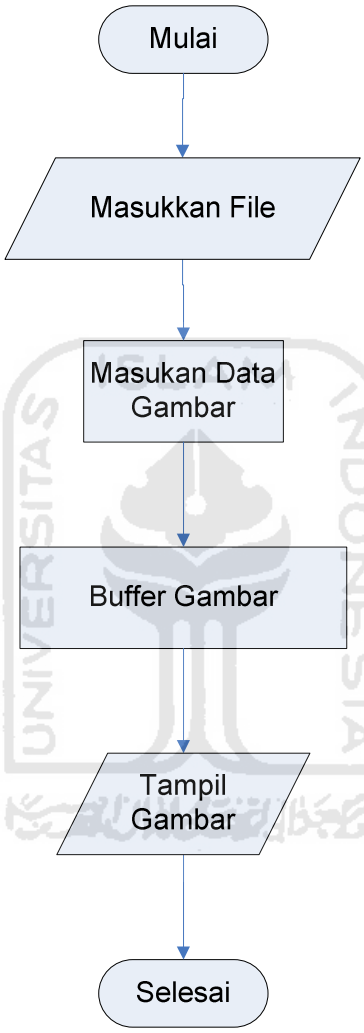


Gambar 3.1 Proses Umum Sistem

Flowchart pada Gambar 3.1 menjelaskan tentang proses secara umum dalam pembuatan aplikasi pengenalan rambu berbentuk diagonal, maksudnya adalah proses yang menunjukkan alur kerja sistem mulai dari awal hingga akhir sesuai dengan metode yang diterapkan.

3.4.2 Rancangan Masukan Gambar

Rancangan masukkan gambar (data yang akan diolah) dapat digambarkan dengan *flowchart* seperti pada Gambar 3.2.

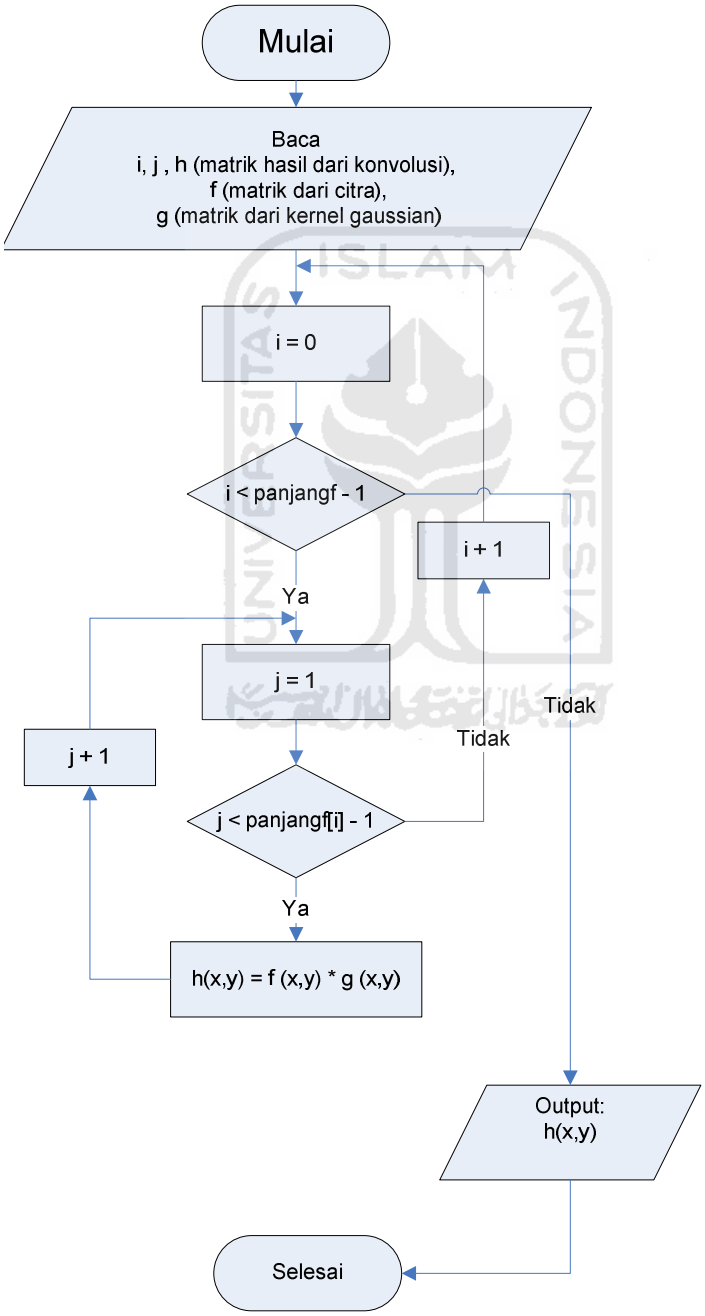


Gambar 3.2 Proses *input* data

Flowchart pada Gambar 3.2 menggambarkan proses *input* data dimana pada proses ini sistem menggunakan *file chooser*. Data yang dipilih dari *file chooser* akan di buffer, hal ini bertujuan untuk mendapatkan informasi data dari citra berupa indeks dan nilai dari setiap pixel yang berguna pada saat proses pengolahan.

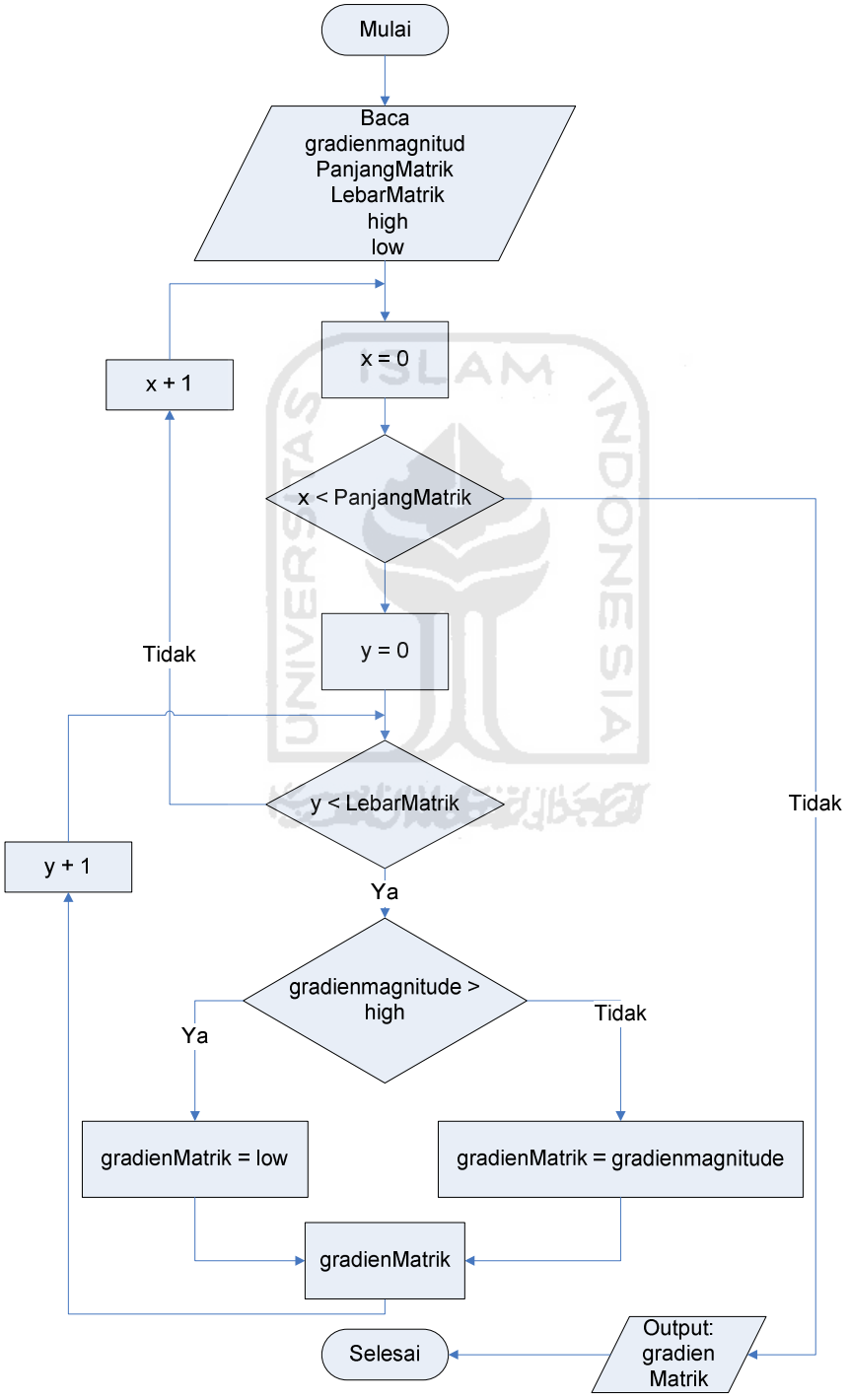
3.4.3 Rancangan Deteksi Tepi

Proses deteksi tepi merupakan suatu proses yang harus dilewati dalam pengenalan objek rambu ini. Hasil deteksi tepi akan menentukan hasil dari pengenalan objek. Jika hasil deteksi tepi detil maka hasil pengenalan juga akan detil. Gambar 3.3 menunjukkan diagram *flow* untuk deteksi tepi dengan metode Canny.

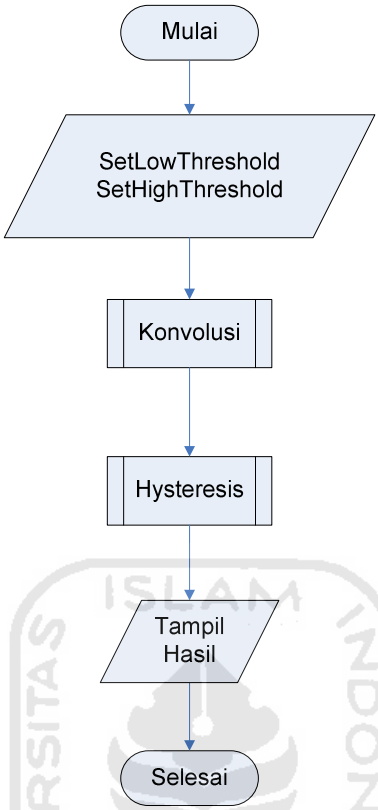


Gambar 3.3 Proses Konvolusi

Proses konvolusi merupakan proses perkalian antara dua buah matrik yaitu matrik dari sebuah citra dengan matrik kernel gaussian. Hal ini bertujuan untuk menghilangkan derau atau *noise.* Pada *flowchart* Gambar 3.3 menghasilkan output h(x,y) yang merupakan matrik hasil perkalian dari dua buah matrik.



Gambar 3.4 Proses Hysteresis



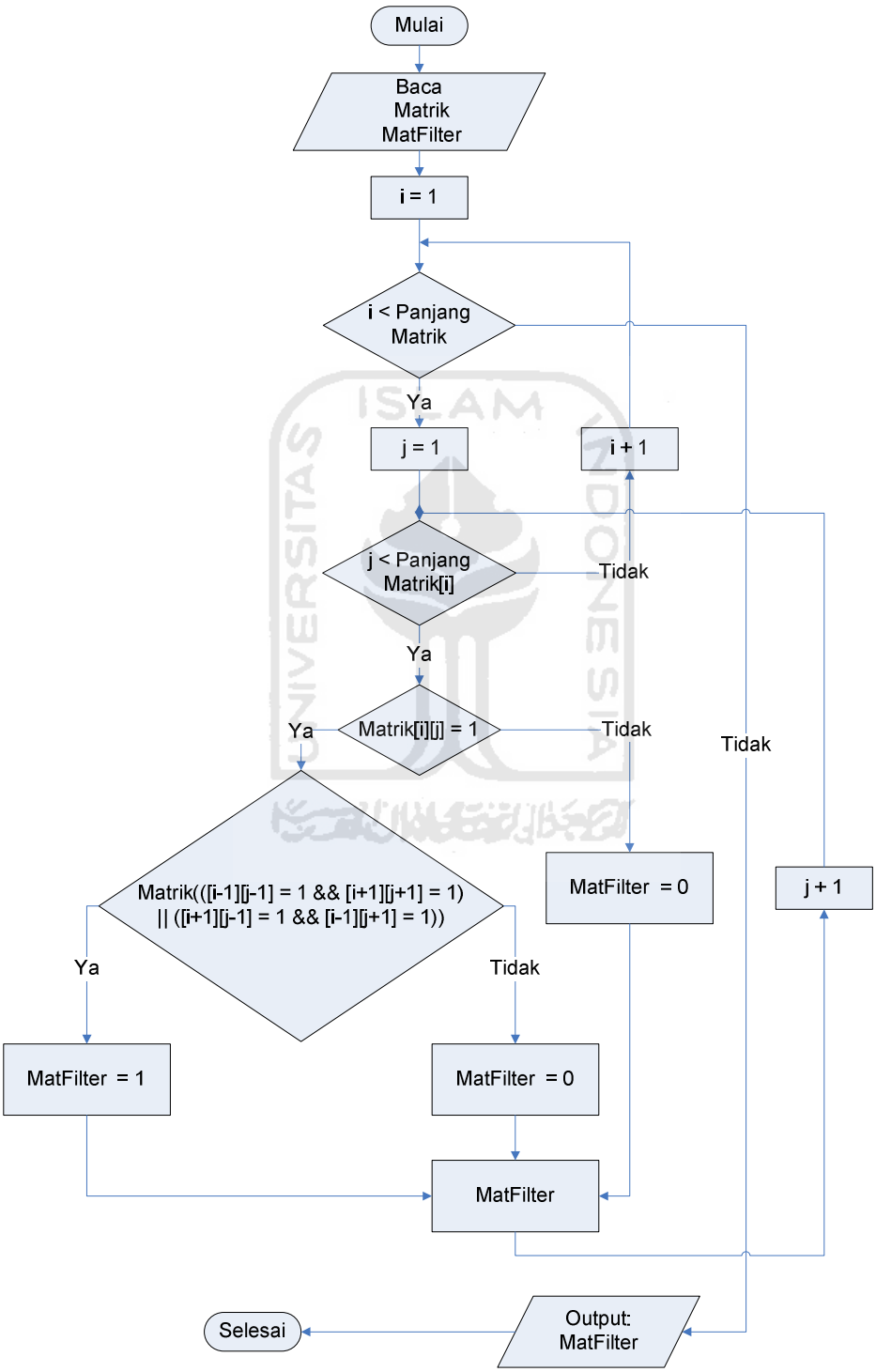
Gambar 3.5 Proses deteksi tepi

Gambar 3.5 menunjukkan diagram alir proses deteksi tepi. Pada proses deteksi tepi pertama kali dilakukan pengaturan nilai *threshold* yang berguna pada proses *non max suppretion* atau proses penipisan tepi. Nilai *threshold* yang diset ada dua yaitu *highthreshold* yang menjadi batas atas dan *lowthreshold* yang menjadi batas bawah dari nilai *threshold* dari setiap piksel. Setelah itu dilakukan proses konvolusi untuk menghilangkan noise dengan menggunakan kernel gausian. Hystersis yaitu proses mengklasifikasikan dengan dua buah nilai yang telah kita set dari awal yaitu *high threshold* dan *low threshold,* proses ini dapat dilihat pada Gambar 3.4. Setelah itu akan tampil hasilnya.

3.4.4 Rancangan Filter

Filter yaitu saringan dimana kita melakukan proses menangkap piksel-piksel yang dibutuhkan untuk mewakili objek yang akan dikenali. Pada proses *filtering* ini sistem menerapkan konsep diagonal *neighbors*. Jadi sistem akan

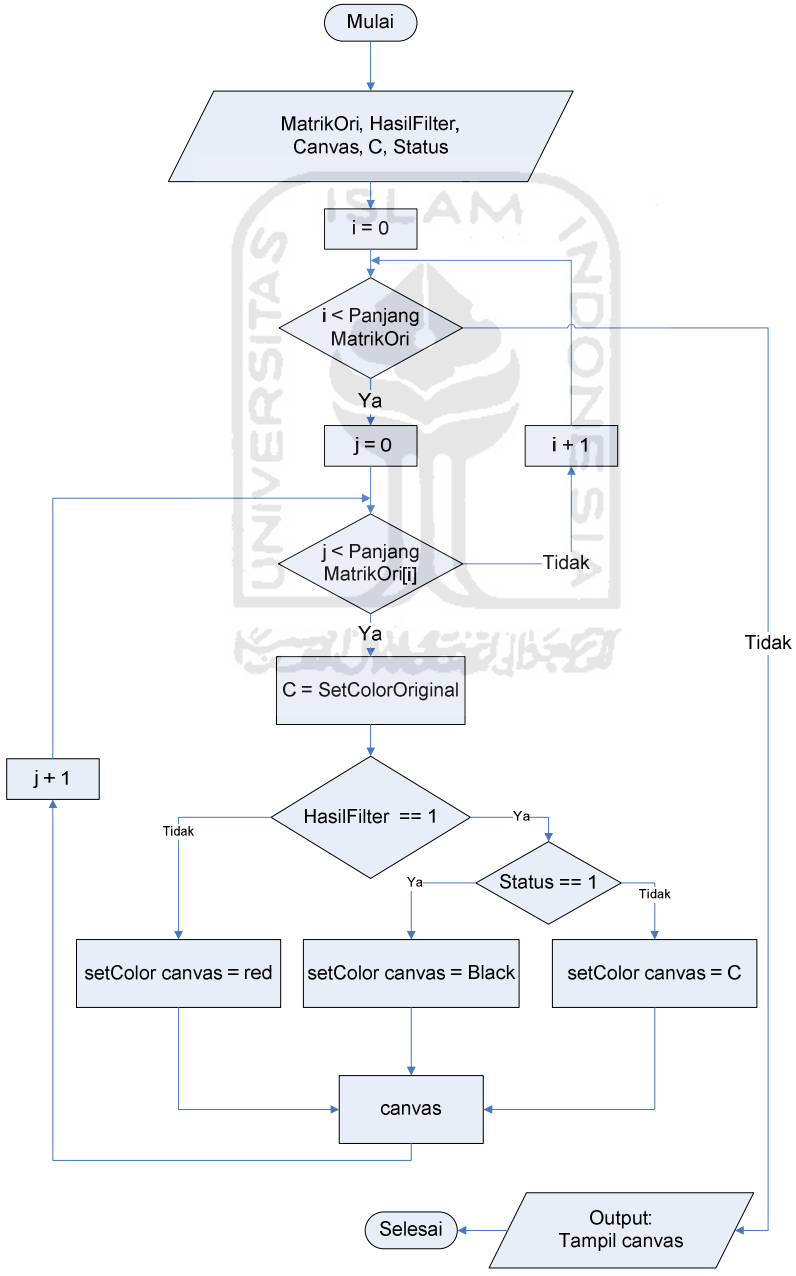
melakukan pengecekan terhadap piksel-piksel tetanggal secara diagonal. Dapat dilihat pada flowchart Gambar 3.6 berikut ini.



Gambar 3.6 Proses *Filtering*

3.4.5 Rancangan Hasil

Hasil yaitu *output* yang diharapkan dari sistem ini. *Output* dari sistem ini berupa image. Jika pada data input dikenali adanya objek rambu dengan pola belah ketupat maka keluaranya menghasilkan gambar sesuai dengan masukkan tetapi akan terdapat garis berwarna merah yang akan menandakan objek rambu yang dikenali dengan pola belah ketupat. Proses ini dapat dilihat pada Gambar 3.7.



Gambar 3.7 Proses *Output* (Hasil)

3.5 Perancangan Antarmuka

Antarmuka (*interface*) merupakan bagian yang juga harus kita perhatikan dengan baik, karena *interface* sangat menentukan ketertarikan dan kemudahan pengguna dalam menjalankan sistem. Antarmuka dirancang harus semudah dan semenarik mungkin agar pengguna dapat dengan mudah menggunakan sistem.

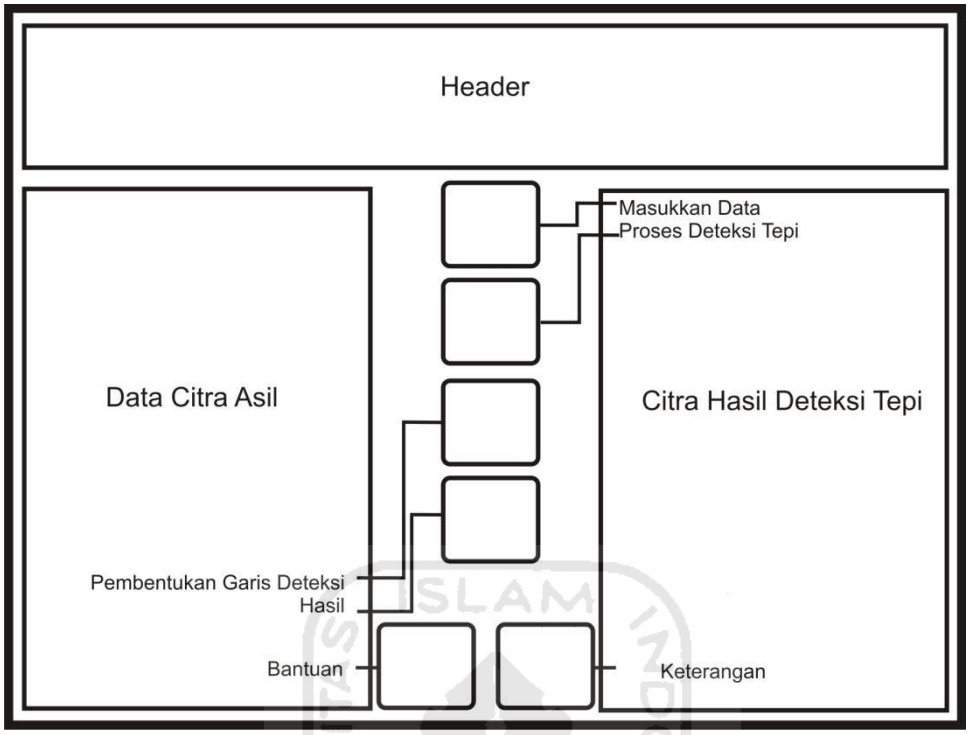
Antarmuka yang sulit dipahami akan membingungkan penggunanya dan menyebabkan sistem tidak dapat digunakan dengan sempurna. Kemudahan pengguna dalam menggunakan sistem dapat dikatakan sebagai keberhasilan perancangan antar muka.

3.5.1 Antarmuka Halaman Utama

Antarmuka halaman utama merupakan tampilan yang pertama kali muncul ketika aplikasi dijalankan. Pada halaman utama ini terdapat beberapa *button* yang berfungsi untuk melakukan proses pengolahan data*, buttonnya* yaitu:

1. Input data
2. Proses deteksi tepi
3. Pembentukan Garis Deteksi
4. Hasil
5. Bantuan
6. Keterangan

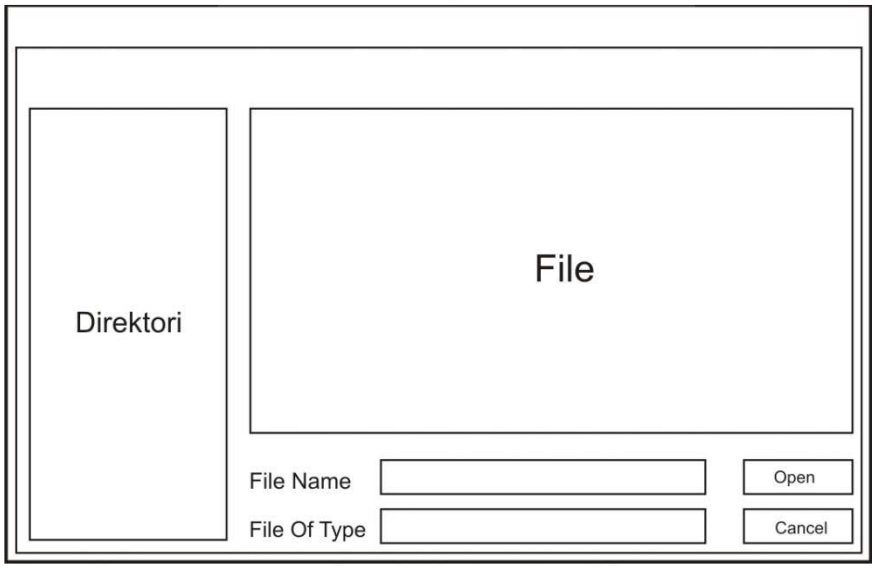
Button input data yaitu *button* yang berfungsi untuk memasukkan data kedalam sistem untuk diolah. Untuk *button* proses deteksi tepi berguna untuk melakukan proses deteksi tepi dari data yang telah dimasukkan. Pada proses ini akan menghasilkan *citra biner* yang hanya memiliki 2 intensitas warna yaitu putih dan hitam. Sedangkan *button* hasil berfungsi untuk proses *filtering* yang bertujuan untuk melakukan proses deteksi terhadap objek rambu yang bentuknya diagonal. Untuk button bantuan berfungsi untuk membuka sebuah *panel* yang berisikan cara penggunaan aplikasi dan persyaratan dari data *inputan* yang akan dilakukan proses pendeteksian. Dan *button* about merupakan *button* yang berfungsi untuk membuka sebuah *panel* yang berisikan keterangan dan data dari pembuat aplikasi ini.



Gambar 3.8 Rancangan Antarmuka Halaman Utama

3.5.2 Antarmuka *FileChooser*

FileChooser merupakan form yang digunakan untuk pengimputan data kedalam sistem untuk di proses. *FileChooser* akan muncuk ketika *button* masukan data di klik. Berikut antarmuka dari *FileChooser*.



Gambar 3.9 Rancangan Antarmuka *FileChooser*

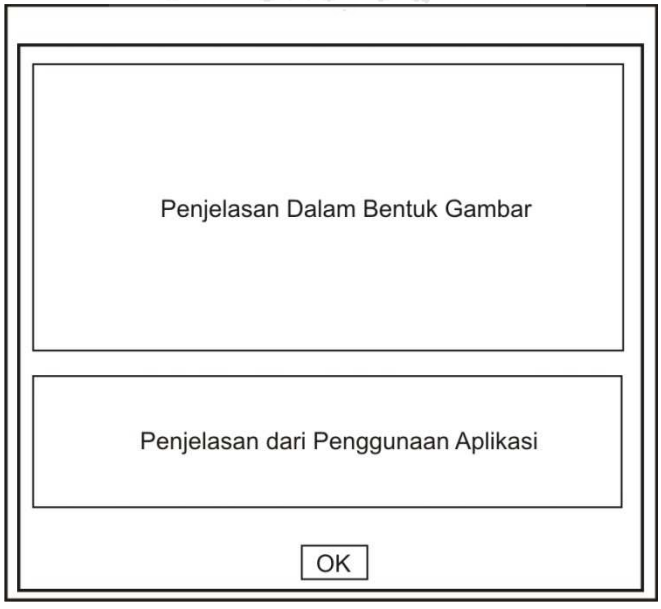
3.5.3 Antarmuka Hasil

Hasil merupakan *form* untuk menampilkan hasil dari proses pengenalan rambu-rambu dengan pola belah ketupat. Berikut antarmuka dari hasil.

Gambar 3.10 Rancangan Antarmuka Hasil

3.5.4 Antarmuka Halaman Bantuan

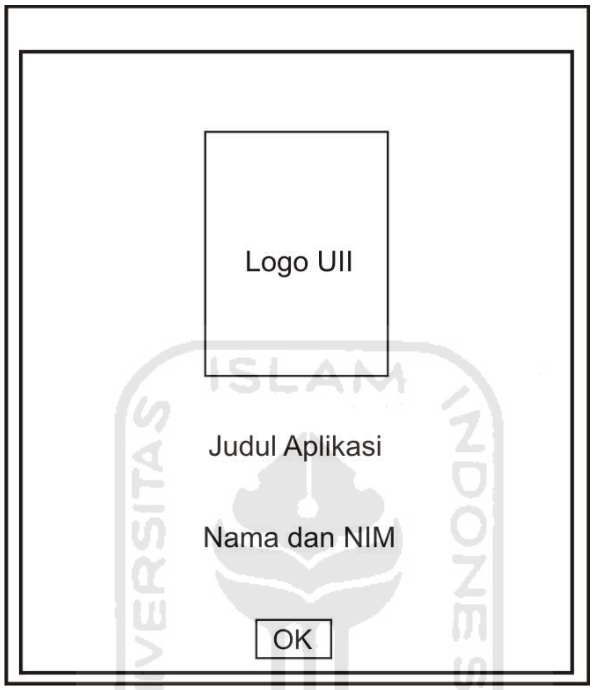
Bantuan merupakan halaman yang berisi tentang penjelasan dari cara penggunaa aplikasi pengenalan rambu belah ketupat. Mulai dari pengimputan data, proses pengenalan sampai hasil. Berikut antarmuka dari halaman bantuan.



Gambar 3.11 Rancangan Antarmuka Bantuan

3.5.5 Antarmuka Halaman Keterangan

Halaman about berisi tentang data dari pembuat dan judul dari aplikasi. Halaman ini akan tampil apabila *user* memilik menu about. Berikut antarmuka dari halaman keterangan.



Gambar 3.12 Rancangan Antarmuka Keterangan

BAB IV
HASIL DAN PEMBAHASAN

Hasil dan pembahasan memaparkan tentang implementasi perangkat lunak yang meliputi aplikasi pengenalan rambu berbentuk belah ketupat, perangkat keras, perangkat lunak, proses pendeteksian, implementasi antarmuka, dan pengujian program.

4.1 Aplikasi Pengenalan Rambu Berbentuk Belah Ketupat

Aplikasi ini merupakan aplikasi dekstop. Karena pembuatan aplikasi ini menggunakan java maka aplikasi ini dapat berjalan di beberapa *platform* seperti windows dan linux. Tetapi untuk menjalankan aplikasi ini diperlukan sebuah JRE(*Java Runtime Error*). JRE merupakan sebuah paket yang dibutuhkan untuk menjalankan sebuah aplikasi java. Aplikasi ini dibangun menggunakan *software develop* yang merupakan produk dari SUN Microsystem yaitu Netbeans 6.9. Pada aplikasi ini terdapat beberapa tahap proses yang dilakukan untuk mengenali pola dari rambu yang berbentuk belah ketupat.

4.2 Perangkat Keras untuk Implementasi

Perangkat keras digunakan sebagai alat pengolahan data yang berasal dari *input* berupa *citra*. Dalam proses pengolahan data *citra* sangat membutuhkan proses komputasi yang besar. Hal ini disebabkan proses pengolahan *citra* sangat kompleks. Dalam hal ini secara tidak lansung proses ini membutuhkan spesifikasi komputer yang tinggi, terutama *processornya*. Adapun spesifikasi komponen perangkat keras yang diperlukan untuk perancangan dan pembuatan aplikasi ini adalah sebagai berikut:

1. Piranti berupa Keyboard dan mouse.
2. Piranti *output* berupa monitor dengan resolusi minimal 1024x768.
3. *Processor* minimal *dual core* dan memiliki kecepatan diatas 2 ghz.
4. Memori RAM minimal 1 GB.
5. Harddisk yang memiliki ruang kosong minimal 1 GB.

Selain perangkat komputer ada perangkat keras lainnya yang juga sangat dibutuhkan yaitu sebuah kamera digital yang memiliki resolusi minimal 640x480 piksel.

4.3 Perangkat Lunak untuk Implementasi

Selain perangkat keras, perangkat lunak juga sangat diperlukan dalam proses pembuatan aplikasi dan dalam menjalankan aplikasi. Perangkat lunak yang dibutuhkan adalah sebagai berikut:

1. *Sistem Operasi*, sistem operasi yang dibutuhkan untuk proses perancangan dan pembuatan aplikasi adalah antara lain Windows XP, Vista, atau Seven.
2. *NetBeans 6.9*, NetBeans 6.9 merupakan aplikasi yang digunakan untuk merancang tampilan dan membuat fungsi-fungsi yang digunakan untuk membuat sebuah aplikasi. NetBeans adalah *Integrated Development Environment* (IDE) berbasiskan Java dari Sun Microsystems yang berjalan di atas swing. Swing merupakan sebuah teknologi Java untuk pengembangan aplikasi desktop yang dapat bejalan di berbagai macam platforms seperti Windows, Linux, Mac OS X and Solaris. Suatu IDE adalah lingkup pemrograman yang diintegrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan pembangun *Graphic User Interface* (GUI), suatu text atau kode editor, suatu compiler atau interpreter dan suatu debugger. Netbeans merupakan *software* development yang *Open Source*, dengan kata lain *software* ini di bawah pengembangan bersama (gratis).
3. *JDK,* JDK (Java Development Kit) adalah *software development kit* yang digunaka dalam bahasa pemrograman Java. Untuk dapat mengembangkan atau merancang dan membuat aplikasi JAVA, pada komputer haruslah terinstal JDK. JDK berjalan diatas sebuah *virtual machine* yang dinamakan JVM (Java Virtual Machine). JVM adalah sebuah machine imajiner (maya) yang bekerja menyerupai aplikasi pada sebuah mensin nyata, menyediakan spesifikasi *hardware* dan *platform* di mana kompilasi java terjadi. JVM bebas dari platform manapun, karena proses kompilasi

diselesaikan oleh JVM itu sendiri (tidak tergantung dengan *platform* lainnya.

4. *JRE,* JRE adalah paket lingkungan yang dibutuhkan jika ingin menjalankan aplikasi JAVA. JRE menterjemahkan *Byte code* yang merupakan hasil kompilasi. JRE bersifat spesifik *platform*. Contoh: masing-masing sistem operasi memiliki JRE tersendiri.

4.4 Proses Pendeteksian

Dalam melakukan proses pengenalan terhadap objek, sistem ini melalui beberapa tahapan proses sesuai dengan fungsi dan metode yang digunakan. Mulai dari *input* data untuk diolah sampai menghasilkan *output*.

4.4.1 Input Data

Dalam proses input data sistem ini menggunakan file *chooser*. Berikut *code* yang berfungsi untuk memasukkan data dan menampilkannya di sistem.

```
public void inputGambar() {
        jButton3.setEnabled(false);
        jFileChooser1.setSelectedFile(null);
        jFileChooser1.showOpenDialog(this);
        gambar = jFileChooser1.getSelectedFile().toString();
        ImageIcon icon = new ImageIcon(gambar);
        jLabel2.setIcon(icon);
        JOptionPane.showMessageDialog(null, "Input Data
berhasil");
    }
```

Data yang diinputkan melalui *file chooser* diinisialisasikan ke dalam sebuah varibel dengan cara:

```
gambar = jFileChooser1.getSelectedFile().toString();
```

dan untuk menampilkan dalam aplikasi yaitu dengan menginstansiasikan variabel gambar melalui parameter dari ImageIcon.

```
ImageIcon icon = new ImageIcon(gambar);
jLabel2.setIcon(icon);
```

4.4.2 Deteksi Tepi Canny

Pada proses deteksi tepi dengan menggunakan metode canny ditentukan oleh dua buah nilai T (*Threshold*). Yaitu *Low Threshold* dan *High Threshold*. Dua

nilai ini merupakan rentang nilai *threshold* yang akan di hilangkan. Dan dua nilai ini akan menentukan hasil dari deteksi tepi.

```
canny.setLowThreshold(0.005f);
canny.setHighThreshold(18f);
```

Jadi untuk nilai ambang dua buah nilai *threshold* yang disetkan yaitu 0.005f untuk nilai *low threshold* dan 18f untuk *high threshold*. Dengan rentang nilai ambang diatas akan mendapatkan hasil yang maksimal untuk hasil deteksi tepi apabila data memiliki tingkat kontras yang bagus. Hasil dikatakan maksimal yaitu ketika tidak terlalu banyak *noise* yang didapat tetapi menghasilkan tepi yang halus. Berikut potongan *code* proses dari fungsi deteksi tepi dengan menggunakan metode canny.

```
public void process() {
        width = sourceImage.getWidth();
        height = sourceImage.getHeight();
        picsize = width * height;
        initArrays();
        readLuminance();
        if (contrastNormalized) {
            normalizeContrast();
        }
        computeGradients(gaussianKernelRadius,
gaussianKernelWidth);
        int low = Math.round(lowThreshold * MAGNITUDE_SCALE);
        int high = Math.round(highThreshold *
MAGNITUDE_SCALE);
        performHysteresis(low, high);
        thresholdEdges();
        writeEdges(data);
    }
```

Hasil dari proses deteksi tepi Canny adalah sebuah citra biner. Citra Biner disebut juga dengan *bi-level* yaitu citra digital yang hanya mempunyai dua kemungkinan nilai pada pixel-pixelnya (hitam atau putih).

4.4.3 Mengambil Nilai Matrik

Fungsi ini berguna untuk mengambil nilai dan mengetahui letak piksel berdasarkan baris dan kolom matrik dari sebuah citra yang akan diolah. Pada aplikasi ini disebut juga dengan *getMatrik.* Pengambilan indeks dan nilai dari

piksel berguna untuk proses filter dan proses pembentukan garis deteksi. Berikut potongan *code* pengambilan nilai matrik.

```
public void getPixOriginal(BufferedImage image, int w, int h)
{
        int barisCounter = 0;
        int kolomCounter = 0;

        matrikRGBTemp = new int[h][w];

        //Get all pixels
        for (int i = 0; i < w * h; i++) {
            if (kolomCounter == w) {
                kolomCounter = 0;
                barisCounter++;
            }
            matrikRGBTemp[barisCounter][kolomCounter] =
image.getRGB(kolomCounter, barisCounter);
            kolomCounter++;
        }
    }
```

Setelah melakukan pengambilan nilai piksel dilakukan perubahan nilai piksel menjadi 0 dan 1. Nilai 0 mendeklarasikan nilai dari warna hitam dan nilai 1 untuk mendeklarasikan warna putih. Dalam proses perubahan nilai ini dilakukan pengecekan, berikut potongan *code* untuk perubahan nilai piksel menjadi 0 dan 1.

```
for (int i = 0; i < matrik.length; i++) {
            for (int j = 0; j < matrik[i].length; j++) {
                if (matrik[i][j] > 1) {
                    matrik[i][j] = 0;
                } else{
                    matrik[i][j] = 1;
                }
            }
        }
        System.out.println("\n\n");
```

4.4.4 Filter

Filter bertujuan untuk menyaring suatu data dimana pada aplikasi ini berupa matrik agar mendapatkan hasil yang sesuai dengan kriteria dari filter itu sendiri. Dalam proses ini filter digunakan untuk mendapatkan ciri atau kriteria

dari rambu-rambu yang berbentuk belah ketupat. Fungsi filter pada aplikasi ini menerapkan metode diagonal neighboars (P2.3), hal ini dikarenakan proses pengecekan dilakukan terhadap piksel tetangga secara diagonal. Berikut *code* untuk proses filter.

```
matFilter = new int[matrik.length][matrik[1].length];
        for (int i = 1; i < matrik.length - 1; i++) {
            for (int j = 1; j < matrik[i].length - 1; j++) {
                if (matrik[i][j] == 1) {

                    for (int k = 1; k <= 3; k++) {
if ((matrik[i - k][j - k] == 1 &&     matrik[i + k][j + k] ==
1) || (matrik[i + k][j - k] == 1 && matrik[i - k][j + k] == 1))
{
                            matFilter[i][j] = 1;

                        } else {
                            matFilter[i][j] = 0;

                        }
                    }

                } else {
                    matFilter[i][j] = 0;
                }
            }
        }
```

Filterisasi pada aplikasi ini dilakukan dengan pengecekan terhadap piksel tetangga dari matrik yang berasal dari sebuah gambar hasil dari deteksi tepi.

```
matFilter = new int[matrik.length][matrik[1].length];
```

matFilter merupakan matrik baru yang berguna untuk menampung nilai matrik setelah dilakukan proses filter. Rangkaian proses ini dinamakan fungsi *palette component* dengan *code* sebagai berikut.

```
if ((matrik[i - k][j - k] == 1 && matrik[i + k][j + k] == 1)||
(matrik[i + k][j - k] == 1 && matrik[i - k][j + k] == 1)){
                            matFilter[i][j] = 1;
                        } else {
                            matFilter[i][j] = 0;
                        }
                    }
```

4.4.5 Pembentukan Garis Deteksi

Pada proses pembentuk garis deteksi dilakukan proses perubahan warna pada piksel yang masuk dalam kategori filter sehingga membentuk garis. Warna dari piksel tersebut akan dirubah menajadi warna merah. Garis berwarna merah ini berguna sebagai penanda bagian dari pola yang terdeteksi yaitu pola berbentuk belah ketupat pada sebuah citra. Setelah terbentuk pola dengan garis merah maka piksel selain yang berwarna merah tersebut akan dirubah kembali dengan warna asli sesuai dengan gambar awal. Berikut *code* dari fungsi paint.

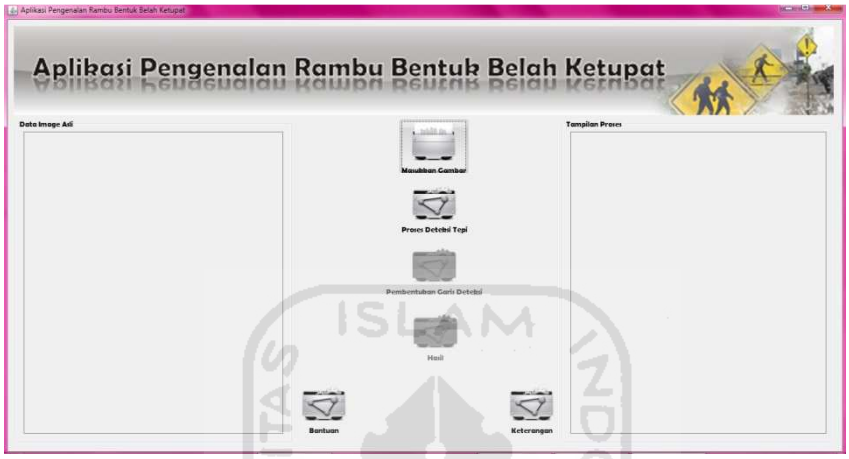
```
public void paintComponent(Graphics g) {
        Graphics2D g2d = (Graphics2D) g;
        Color c;

        int counter = 0;

        for (int i = 0; i < this.a.length; i++) {
            for (int j = 0; j < this.a[i].length; j++) {
                c = new Color(this.a[i][j]);

                if (this.filterMatrixCuys[i][j] == 1) {
                    g2d.setColor(Color.RED);
                } else {
                    if (this.status == 1)
                    {
                        g2d.setColor(Color.BLACK);
                    } else
                    {
                        g2d.setColor(c);
                    }
                }
                int x = j;
                int y = i;

                g2d.drawLine(x, y, x, y);
                counter++;
            }
        }
    }
```

4.5 Implementasi Antarmuka

Implementasi antarmuka adalah hasil tampilan pembuatan sistem sesuai dengan rancangan antarmuka sebelumnya. Setelah pembuatan maka dapat dipresentasikan hasil dari perancangan yang telah dibuat.

4.5.1 Halaman Utama

Halaman ini merupakan halaman yang menjadi induk dari semua proses. Pada halaman ini terdapat dua panel yang menampilkan data awal yang akan diproses yaitu pada panel sebelah kiri dan data setelah proses deteksi tepi pada panel sebelah kanan. Gambar 4.1 menunjukkan tampilan dari halaman utama.

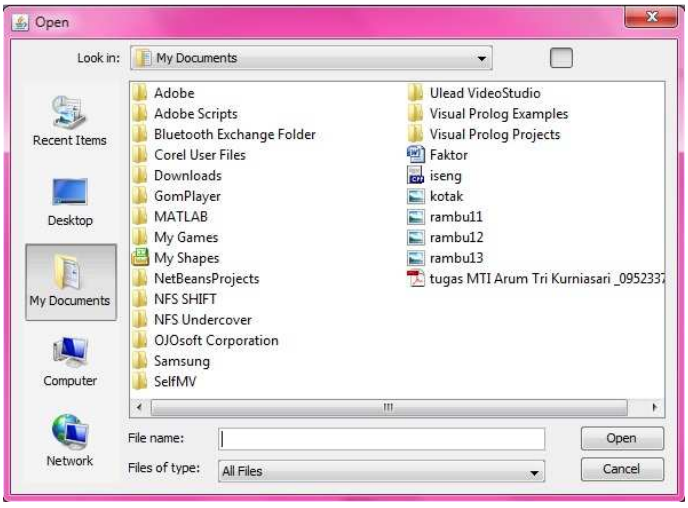


Gambar 4.1 Tampilan Halaman Utama

Terdapat lima *button* pada halaman ini yaitu untuk proses *input* data, proses deteksi tepi, hasil, bantuan, dan keterangan.

4.5.2 Halaman *FileChooser*

Halaman ini akan tampil apabila *user* memilih menu masukkan gambar. Menu ini berguna untuk memilih dan memasukkan gambar untuk diolah kedalam sistem. Gambar 4.2 menunjukkan tampilan dari *FileChooser*.



Gambar 4.2 Tampilan FileChooser

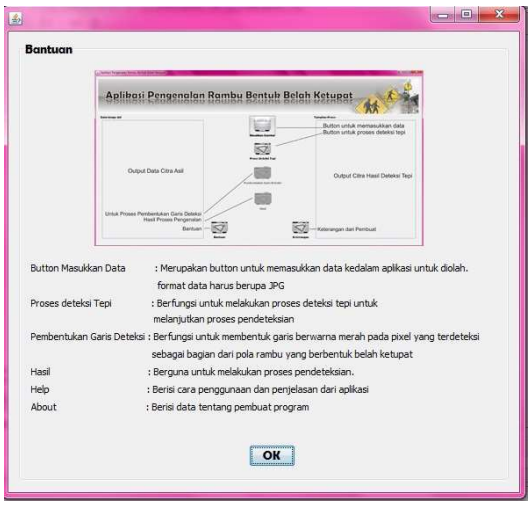
4.5.3 Halaman Hasil

Halaman ini hanya merupakan panel untuk menampilkan hasil dari proses *filter* dan hasil pengenalan pola rambu belah ketupat. Gambar 4.3 menunjukkan tampilan dari halaman hasil.

Gambar 4.3 Tampilan Hasil

4.5.4 Halaman Bantuan

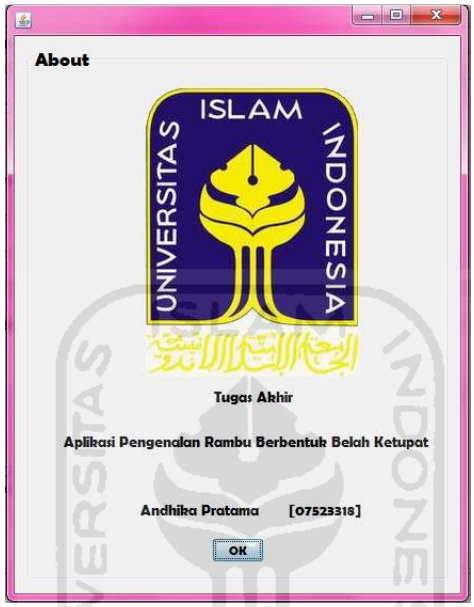
Dari namanya tentu sudah jelas bahwa halaman ini merupakan petunjuk atau disebut juga dengan *user manual* dari aplikasi ini. Pada halaman ini hanya terdapat satu button untuk kembali ke halaman utama. Untuk tampilan halaman bantuan dapat dilihat pada Gambar 4.4.



Gambar 4.4 Tampilan Halaman Bantuan

4.5.5 Halaman Keterangan

Halaman ini merupakan salah satu bagian yang penting dari sebuah sistem dimana halaman ini menjelaskan hak cipta yang membuat aplikasi ini. Sama dengan halaman bantuan, halaman keterangan juga hanya memiliki satu button untuk kembali ke halaman utama. Dapat dilihat pada Gambar 4.5.



Gambar 4.5 Tampilan Halaman Keterangan

BAB V
ANALISIS KINERJA PERANGKAT LUNAK

Analisis kinerja perangkat lunak yangakan dibahas meliputi pengujian aplikasi terhadap sampel data, pengujian beban komputasi, pengujian *stability,* penanganan kesalalahan pada sistem, permasalahan yang belum terselesaikan, serta kelebihan dan kekurangan sistem

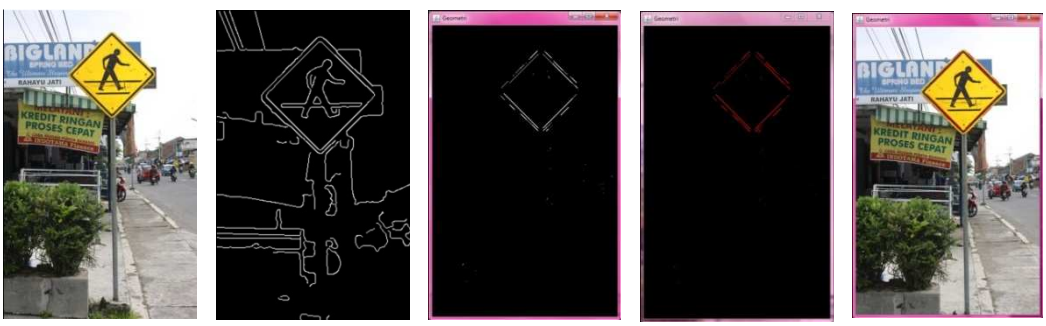
5.1 Pengujian Aplikasi Terhadap Sampel Data

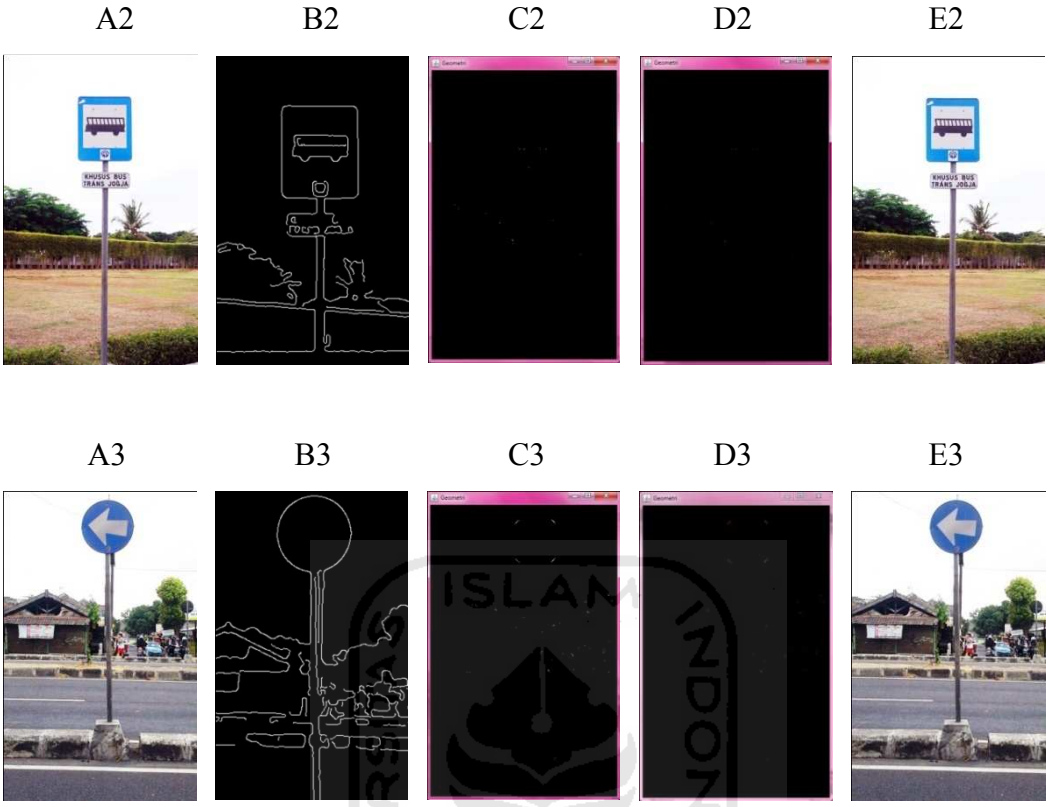
Setelah selesainya pembuatan aplikasi pengenalan rambu berbentuk belah ketupat maka proses selanjutnya yaitu pengujian terhadap beberapa sampel data yang berbeda.

A. Pengujian dengan Sampel yang Berbeda Bentuk

Pada sub bab ini akan dilakukan pengujian terhadap hasil pengenalan dan akan dilakukan perbandingan dari tiga sampel data yang mana bentuk dari objek rambunya berbeda . Pada perbandingan ini akan di tampilkan sesuai dengan tahapan dari masing masing proses dintaranya yaitu, A adalah gambar data asli, B adalah gambar hasil dari proses deteksi tepi, C adalah gambar dari hasil *filter* dan *morfologi*, D adalah gambar hasil pengenalan pola, dan D adalah hasil akhir. Ilustrasi perbandingan ini dapat dilihat pada Gambar 5.1.

A1 B1 C1 D1 E1





Gambar 5.1 Hasil Perbandingan Berdasarkan Bentuk

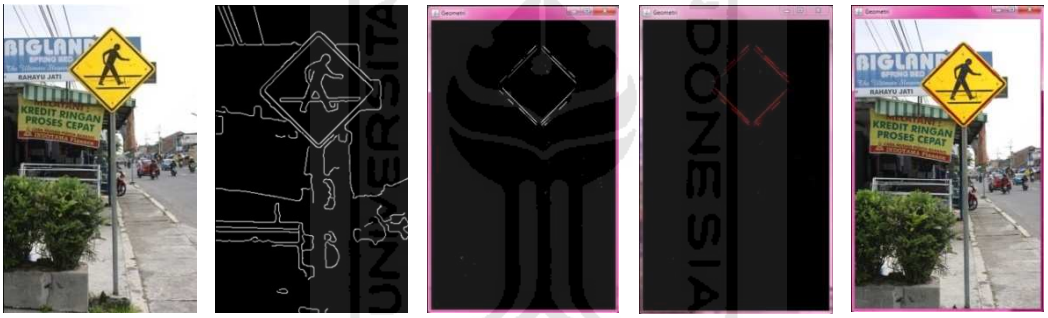
Dari beberapa sampel dalam Gambar 5.1, dapat dilihat hasil dari setiap proses dan dapat dibandingkan mulai dari hasil deteksi tepi, *morfologi* dan hasil akhir pengenalan pola. Dapat dilihat pada Gambar 5.1 A1 yang merupakan hasil dari deteksi tepi dari sampel data pertama menghasilkan banyak tepi dari objek yang terdeteksi, hal ini disebabkan karena banyaknya objek yang terdapat pada sampel data pertama dan kontras dari image sangat tinggi sehingga menghasilkan tepi yang jelas. Pada sampel data pertama terlihat objek rambu berbentuk belah ketupat, sampel kedua berbentuk bulat dan sampel ketiga berbentuk kotak. Dari ketiga sampel dapat dilihat dan bandingkan hasil dari *filtering* dan *morfologi*nya pada Gambar 5.1 C1, C2, dan C3. Terlihat dengan jelas perbedaaan dari ketiga sampel bahwa pada sampel data pertama yaitu Gambar 5.1 C1 terdeteksi adanya sebuah objek yang bentuknya belah ketupat sehingga pada hasil *filtering* dan *morfologi* direpresentasikan kedalam garis berwarna merah pada Gambar 5.1 D1, sedangkan pada sampel data kedua dan ketiga tidak ada objek yang terdeteksi. Hal

ini disebabkan pada sampel data dua dan tiga tidak ada objek yang masuk dalam kriteria filter yaitu berbentuk belah ketupat. Dengan kata lain, filter yang dipilih sudah tepat untuk mengenali bentuk belah ketupat.

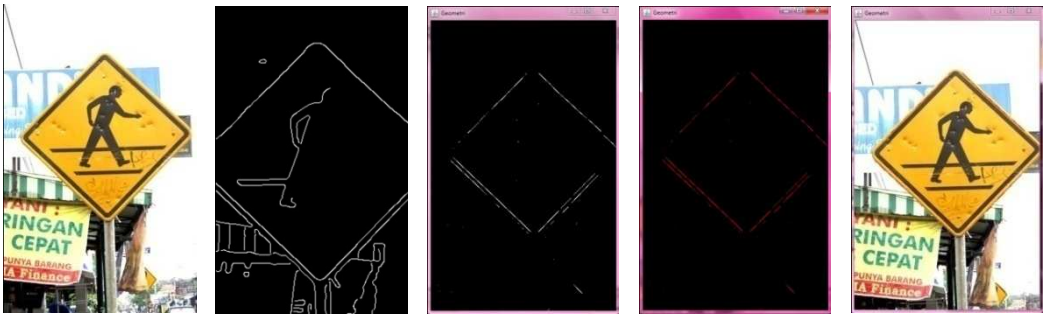
B. Pengujian dengan Sampel yang Posisi Berbeda

Selain bentuk atau pola dari objek, posisi juga merupakan salah satu parameter dalam pengujian sistem ini. Aplikasi ini akan diujikan terhadap sampel data yang memiliki objek rambu dengan pola belah ketupat, tetapi dengan 3 variasi posisi dari objek rambu tersebut yaitu atas, tengah, dan bawah, lihat Gambar 5.2.

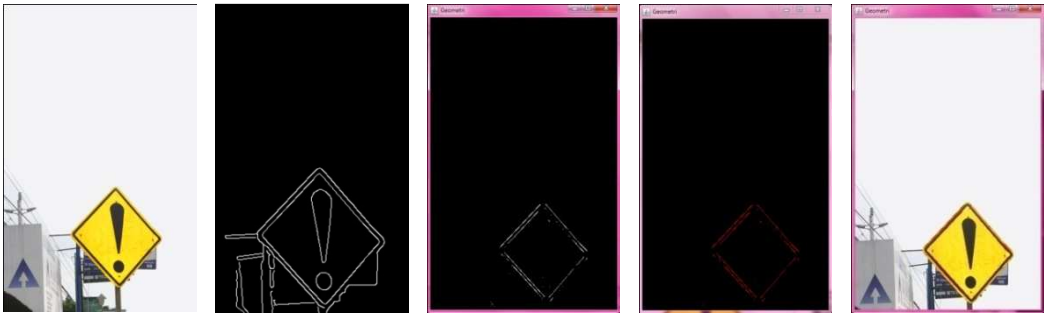
A1 B1 C1 D1 E1



A2 B2 C2 D2 E2



A3 B3 C3 D3 E3



Gambar 5.2 Hasil Perbandingan Berdasarkan Posisi

Dari hasil perbandingan diatas dapat dilihat bahwa posisi tidak mempengaruhi proses pengenalan. Walaupun posisi rambu ada di atas, tengah, atau bawah, aplikasi tetap dapat mengenali rambu-rambu dengan pola belah ketupat.

5.2 Pengujian Beban Komputasi

Proses uji komputasi dilakukan untuk mengetahui seberapa besar proses komputasi yang diperlukan. Proses ini dilakukan dengan menghitung waktu yang dipakai untuk memproses sebuah data dengan cara manual menggunakan sebuah alat yaitu stopwatch. Perhitungan waktu dimulai pada saat *button* diklik sampai proses selesai dan menampilkan hasil. Pada pengujian ini dilakukan perbandingan dari tiga sampel data. Berikut perbandingan dari hasil pengujian komputasi yang dapat dilihat pada Tabel 5.1.

Tabel 5.1 Hasil Perbandingan Beban Komputasi

No	Data	Deteksi Tepi dan Filtering	Pembentukan Garis Deteksi	Hasil	Total Waktu
1		4,270 s	4,500 s	4,585 s	13,355 s

2		4,600 s	4,620 s	3,850 s	13.07 s
3		2,190 s	1,850 s	1,845 s	5.885 s

Dari Tabel 5.1 diatas dapat dilihat bahwa semakin komplek objek dari suatu gambar, maka akan semakin besar juga waktu yang di perlukan dalam melakukan proses. Waktu rata-rata dari total waktu yang diperlukan dalam semua proses pengenalan yaitu 10.77 s. Sedangkan standar deviasinya yaitu 4.233 s

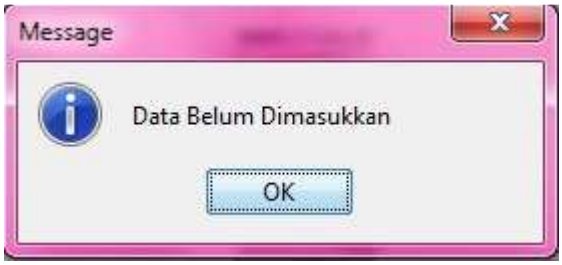
5.3 Pengujian *Stability*

Proses pengujian *stability* dilakukan untuk mengetahui sejauh mana tingkat kinerja dari sistem yang dibuat. Proses pengujian ini dilihat dari kondisi *error* yang terjadi pada saat aplikasi dijalankan. Dari pengujian yang dilakukan pada aplikasi ini terdapat beberapa kondisi *error* yang berhubungan dengan tingkat kestabilan dari kinerja aplikasi. Contohnya yaitu terkadang sensitifitas dari *button* kurang yang mengakibatkan proses penekanan *button* berulang. Hal ini juga bisa disebabkan karena kurang tepatnya letak kursor di atas *button.*

Selain dari sensitifitas dari *button* terkadang juga terjadi *error* pada saat dilakukan pengulangan *input* data dan proses deteksi tepi**.** *error* yang terjadi yaitu sistem tidak dapat mendeteksi data yang telah dimasukkan, sehingga akan keluar *eventhandler* yang mengatakan data belum dimasukkan.

5.1.4 Penanganan Kesalahan Pada Sistem

Penanganan kesalahan ini dilakukan untuk menangkap setiap kesalahan apabila terjadi *error* pada sistem ataupun terjadi kesalahan *user* pada saat memasukkan data. Dalam aplikasi ini dapat dilihat ketika *user* membatalkan untuk memasukkan data, maka akan keluar *eventhadler* seperti di bawah ini.

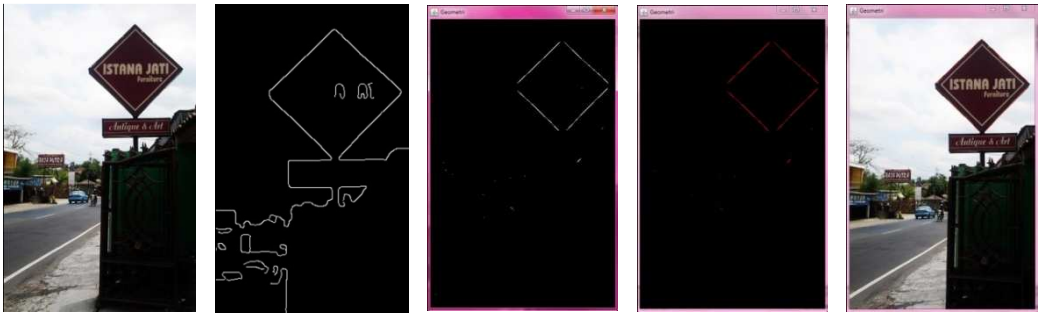


Gambar 5.3 Peringatan Data Belum Masuk

5.5 Permasalahan yang Belum Terselesaikan

Ada beberapa permasalahan yang belum dapat diselesaikan pada aplikasi ini. Gambar 5.4 menunjukan beberapa kasus yang menjadi permasalahan yang belum bisa diselesaikan.

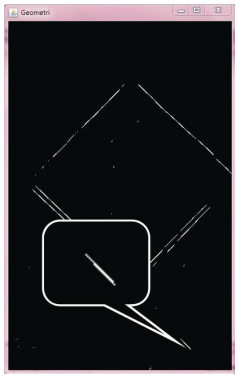
A3 B3 C3 D3 E3



Gambar 5.4 Perbandingan Hasil dari Kasus

Dari perbandingan hasil kasus di atas dapat dilihat bahwa ada beberapa kasus yang belum dapat diselesaikan oleh aplikasi ini. Diantaranya yaitu:

1. Apabila data memiliki kontras yang rendah yang diakibatkan kurangnya cahaya pada saat pengambilan data maka aplikasi tidak dapat melakukan proses pengenalan dengan maksimal seperti yang dapat dilihat pada Gambar 5.4 kasus A1.
2. Sudut pengambilan data juga berpengaruh dalam proses pengenalan ini. Aplikasi ini tidak dapat mengenali rambu apabila sudut pengambilanya terlalu miring. Dapat dilihat pada Gambar 5.4 kasus A2.
3. Aplikasi ini belum dapat membedakan antara rambu dengan benda selain rambu yang sama-sama berbentuk belah ketupat. Sehingga aplikasi ini juga mendeteksi benda selain rambu apabila bentuknya juga belah ketupat. Seperti pada Gambar 5.4 kasus A3
4. Apabila ada garis yang memiliki kemiringan ± 45^{o} juga terdeteksi. Seperti pada Gambar 5.5 dibawah ini.



Gambar 5.5 Kasus Garis

5.6 Kelebihan dan Kekurangan Sistem

Pada bagian ini akan dijelaskan tentang kelebihan dan kekurangan sistem yang dibangun, dilihat dari segi performa dan hasilnya. Berikut adalah ulasannya:

A. Kelebihan

Kelebihan dari Aplikasi Pengenalan Rambu Berbentuk Belah Ketupat ini adalah:

1. Proses deteksi tepi yang digunakan yaitu metode canny yang mana dapat menghasilkan deteksi tepi yang lebih maksimal. Hal ini karena metode canny merupakan sebuah metode deteksi tepi yang baik dibandingkan dengan metode lainnya seperti Prewit, Sobel, dan lainnya.
2. Proses pengenalan rambu belah ketupat dengan filter pilihan yang digunakan telah tepat, sehingga rambu bentuk lain tidak dikenali.

B. Kekurangan

Masih banyak kekurangan pada sistem ini, diantaranya yaitu:

1. Hanya bisa memproses data dengan type data JPG
2. Data yang memiliki kontras rendah atau data yang diambil dengan tingkat kontras yang rendah tidak akan mendapatkan hasil.
3. Tidak dapat mengenali pola apabila sudut pengambilan data terlalu miring.
4. *Stabilitas* pada sistem ini belum baik.

BAB VI
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Berdasarkan hasil penelitian, analisis, perancangan sistem dan pembuatan program sampai dengan tahap penyelesaian, maka dapat ditarik beberapa kesimpulan antara lain sebagai berikut:

1. Aplikasi Pengenalan Rambu Berbentuk Belah Ketupat telah berhasil dibangun dengan menerapkan metode *canny edge detection* untuk proses deteksi tepi dan *diagonal neighbors* untuk proses pengenalan pola belah ketupat.
2. Setelah diujikan terhadap beberapa sampel yang berbeda dapat mengenali rambu bentuk belah ketupat secara akurat.

6.2 Saran

Berdasarkan pada pengujian yang telah dilakukan terhadap *aplikasi* yang dibuat, perlu pengembangan lagi agar menjadi lebih baik, yaitu:

1. Dapat mengolah data dengan tipe data citra selain JPG.
2. Dapat melakukan pengenalan dengan berbagai macam tingkat kontras pada data citra, kegiatan penajaman kontras perlu dilakukan.
3. Lebih dinamis dengan berbagai kondisi objek pada saat pengambilan data citra, penambahan transformasi yang kebal terhadap perputaran bentuk perlu diinvestigasi.
4. Dapat membedakan antara rambu yang sebenarnya dengan papan nama toko atau objek lain yang bentuknya juga belah ketupat, penambahan kriteria pengenalan dengan warna atau bentuk untuk piksel-piksel di dalam belah ketupat perlu dilakukan.

DAFTAR PUSTAKA

Canny, John "*A Computational Approach to Edge Detection*", IEEE Transaction On Pattern Analysis and Machine Intelligence., vol PAMI -8 No 6, November 1986.

Gonzalez, C. Rafael., and Richard E. Woods. 2002. *Digital Image Processing*. 2nd Edition. New Jersey : Prentice Hall, Inc.

Nixon, Mark., and Aguado, S Alberto. 2008. *Feature Extraction & Image Processing*. 2nd Edition". USA: Academic Press.

Burger, Wilhelm., and Mark J. Burge. 2008. *Digital Image Processing An Algorithmic Approach Using Java*. New York: Springer.

Joe. 2009. "*Deteksi Tepi Suatu Citra atau Gambar*". Diakses pada 5 mei 2011 dari http://joyhomework.wordpress.com/2009/12/04/deteksi-tepi-suatu-citra-atau-gambar/