# PERANCANGAN DAN IMPLEMENTASI PENGUKUR PANJANG DENGAN METODE PENGOLAHAN CITRA

## SKRIPSI

**KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER**

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik


Disusun oleh :

**AINUR ROFIQ**
**NIM. 0410633011-63**

**DEPARTEMEN PENDIDIKAN NASIONAL**
**UNIVERSITAS BRAWIJAYA**
**FAKULTAS TEKNIK**
**JURUSAN ELEKTRO**
**MALANG**
**2009**

# PERANCANGAN DAN IMPLEMENTASI PENGUKUR PANJANG DENGAN METODE PENGOLAHAN CITRA

## SKRIPSI

**KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER**

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik


Disusun oleh :

**AINUR ROFIQ**
**NIM. 0410633011-63**

Telah diperiksa dan disetujui oleh

| **Dosen Pembimbing I** | **Dosen Pembimbing II** |
|---|---|
| **Ir. Muhammad Aswin, MT.** | **R. Arief Setyawan,ST.,MT.** |
| **NIP. 19640626 199002 1 001** | **NIP. 19750819 199903 1 001** |

# PERANCANGAN DAN IMPLEMENTASI PENGUKUR PANJANG DENGAN METODE PENGOLAHAN CITRA

## SKRIPSI

**KONSENTRASI TEKNIK INFORMATIKA DAN KOMPUTER**

Diajukan untuk memenuhi persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh :

**AINUR ROFIQ**
**NIM. 0410633011-63**

Skripsi ini telah diuji dan dinyatakan lulus pada
Tanggal 21 oktober 2009

DOSEN PENGUJI

**Waru Djurianto,ST.,MT.**
**NIP. 19690725 199702 1 001**

**Panca Mudjirahardjo,ST.,MT.**
**NIP. 19700329 200012 1 001**

**Ir. Bambang Siswoyo,MT.**
**NIP. 19621211 198802 1 001**

Mengetahui,
Ketua Jurusan Teknik Elektro

**Rudy Yuwono, ST., M.Sc.**
**NIP. 19710615 199802 1 003**

# KATA PENGANTAR

Puji syukur ke hadirat Allah SWT, atas nikmat, hidayah serta kasih sayang-Nya sehingga penulis dapat menyelesaikan tugas akhir dengan judul " **Perancangan Dan Implementasi Pengukur Panjang Dengan Metode Pengolahan Citra** ". Hanya kepada-Nya kita menyembah dan memohon. Teriring doa keselamatan untuk Rasulullah Muhammad SAW, keluarga, sahabat serta seluruh ummatnya. Tugas Akhir ini disusun untuk memenuhi sebagian persyaratan memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Program Studi Sistem Informatika dan Komputer Fakultas Teknik Universitas Brawijaya Malang. Penulis menyadari bahwa tugas akhir ini masih jauh dari sempurna, oleh karena itu penulis berharap semoga ada pengembangan untuk dapat menyempurnakan tugas akhir ini dan semoga tugas akhir ini bermanfaat bagi siapa saja yang memerlukannya.

Tidak banyak yang bisa penulis sampaikan kecuali ungkapan terima kasih kepada berbagai pihak yang telah dengan tulus ikhlas memberikan bimbingan, arahan, dan dukungan hingga penulisan tugas akhir ini dapat terselesaikan. Pada kesempatan kali ini, dengan segala kesungguhan dan rasa rendah hati, penulis mengucapkan banyak terima kasih kepada:

1. Bapak Rudy Yuwono, ST., M.Sc. dan Muhammad Aziz Muslim, ST., MT., PhD selaku Ketua dan Sekretaris Jurusan Teknik Elektro serta segenap Bapak/Ibu Dosen, Staff Administrasi dan Perpustakaan Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya;
2. Bapak Ir. Muhammad Aswin, MT. selaku Dosen Pembimbing I yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini;
3. Bapak Raden Arief Setyawan, ST., MT. selaku Dosen Pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan tugas akhir ini;

4. Kedua orang tua yang sedikitpun tidak pernah lengah dari doa dan harapan untuk terselesaikannya tugas akhir ini.
5. Seluruh mahasiswa teknik elektro terutama rekan-rekan angkatan 2004, yang membantu sehingga Skripsi ini dapat selesai tepat pada waktunya.
6. Semua teman-teman kontrakan yang turut membantu sehingga Skripsi ini dapat selesai tepat pada waktunya.
7. Serta semua pihak yang tidak dapat penulis sebutkan satu per satu yang terlibat baik secara langsung maupun tidak langsung demi terselesaikannya tugas akhir ini.

Hanya doa yang dapat penulis iringkan kepada bapak, ibu serta kakak dan adik yang selalu memberikan doa, kepercayaan, dorongan dan semangat yang tak pernah berhenti.

Dalam pembuatan Skripsi ini penulis menyadari terdapatnya kekurangan dan keterbatasan di dalamnya. Oleh karena itu dengan segala kerendahan hati, penulis mengharapkan kritik dan saran yang bersifat membangun dari semua pihak demi kesempurnaan Skripsi ini. Semoga Skripsi ini dapat bermanfaat bagi kita semua.

Malang, 18 November 2009

Penulis,

## DAFTAR ISI

## DAFTAR GAMBAR

## DAFTAR TABEL

## ABSTRAK

**AINUR ROFIQ. 2009. Perancangan dan Implementasi Pengukur Panjang dengan Metode Pengolahan Citra. Skripsi Jurusan Teknik Elektro, Fakultas Teknik, Universitas Brawijaya. Dosen Pembimbing : Ir. Muhammad Aswin, MT dan Raden Arif Setyawan, ST., MT.**

Teknologi yang semakin berkembang seiring berkembangnya kebutuhan manusia khususnya dalam bidang penginderaan jauh menjadi input penting bagi pengolahan citra digital. Citra hasil penginderaan jauh dapat dimanfaatkan sebagai input dari program pengukur panjang dengan metode pengolahan citra sehingga menghasilkan informasi-informasi yang dibutuhkan. Oleh karena itu analisis-analisis untuk pengolahan citra digital menjadi kebutuhan yang amat penting. Pada tugas akhir ini, akan dibuat sistem pengukur panjang dengan metode pengolahan citra dengan bantuan dua laser yang nantinya akan dipakai sebagai acuan untuk pengukuran tersebut.

Prosesnya adalah pengambilan citra atau gambar dari benda yang akan diukur dengan sebuah kamera dan dua laser yang telah disusun dengan jarak tertentu, dua laser yang dipakai pada tugas akhir ini berjarak 30 cm. Citra atau gambar tersebut akan dijadikan sebagai input dari sistem pengukur panjang dengan metode pengolahan citra. Sistem akan mendeteksi dimana posisi *pixel* dari titik-titik laser dan tepi-tepi benda sehingga menghasilkan keluaran berupa informasi panjang sebenarnya dari benda yang akan diukur.

Pengujian dilakukan dengan dua cara yaitu *white-box* testing dan *black-box* testing, pengujian *white-box* berfokus pada struktur control program. *Test case* dilakukan untuk memastikan bahwa semua statemen pada program telah dieksekusi paling tidak satu kali selama pengujian dan bahwa semua kondisi logis telah diuji. Pengujian *black-box* testing dilakukan terhadap beberapa kotak yang panjang dan tingginya sama dengan warna yang berbeda-beda, dengan tujuan untuk mecari prosentase dari kesalahan maksimal dari sistem. Untuk mengetahui seberapa kecil dan besar pengukuran yang dapat dilakukan oleh sistem dilakukan pengujian benda dari ukuran terkecil sampai terbesar dengan acuan prosentase kesalahan terbesar yang telah ditemukan. Disimpulkan bahwa aplikasi ini bekerja dengan baik dengan batasan panjang benda yang diukur antara 10 cm sampai 10 m.



**Kata Kunci:** pengolahan citra, posisi *pixel,* titik-titik laser, tepi-tepi benda, *white-box testing , black-box testing*

# BAB I

# PENDAHULUAN

Skripsi ini akan membahas pengembangan sebuah sistem perangkat lunak untuk melakukan proses pengukuran benda dari jarak jauh dengan metode pengolahan citra. Bab ini menjelaskan latar belakang dikembangkannya sistem pengukur panjang dengan metode pengolahan citra, rumusan masalah, batasan masalah, tujuan dan manfaat dikembangkannya sistem ini, serta sistematika pembahasan yang digunakan pada penulisan penelitian ini.

## 1.1 Latar Belakang

Teknologi penginderaan jauh berbasis kamera saat ini merupakan salah satu input penting dalam pengolahan citra digital. Banyak benda-benda berbahaya yang gambarnya tidak bisa diambil dari jarak dekat misalnya binatang buas, benda-benda dengan temperatur tinggi, tegangan tinggi dan lain-lain. Dengan menggunakan kamera kita dapat mengambil gambar-gambar benda dari jarak yang cukup jauh, oleh karena itu analisis-analisis untuk pengolahan citra digital menjadi kebutuhan yang sangat penting. Analisis digital bukan hanya menawarkan waktu pengolahan data yang cepat, namun juga menawarkan analisis yang akurat seiring dengan berkembangnya teknologi dan ilmu pengetahuan.

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. Teknologi dasar untuk menciptakan dan menampilkan warna pada citra digital berdasarkan pada penelitian bahwa sebuah warna merupakan kombinasi dari tiga warna dasar, yaitu merah, hijau, dan biru (*Red, Green, Blue* - RGB).

Segmentasi merupakan salah satu analisis citra digital yang berguna bagi citra penginderaan jauh, karena terkadang segmentasi dapat menjadi sebuah analisis awal pada citra sebelum dilakukan analisis lanjut. Segmentasi memiliki keunggulan dari pada analisis yang berbasis *pixel* tunggal. Segmentasi merupakan

analisis yang dapat bekerja pada sekelompok *pixel* dengan berbasiskan kepada nilai dan ruang spesialnya.

Secara umum segmentasi dapat dibagi menjadi tiga kelompok yaitu segmentasi berdasar klasifikasi (*classification based segmentation*), Segmentasi berdasar tepi (*edge based segmentation*), dan segmentasi berdasar daerah (*region based segmentation*). Segmentasi berdasar klasifikasi adalah proses segmentasi yang dilakukan dengan cara mencari kesamaan dari ukuran tertentu pada nilai *pixel*, Segmentasi berdasar tepi adalah proses segmentasi dengan tujuan untuk mendapatkan batas tepi antar obyek. Sementara itu segmentasi berdasar daerah adalah segmentasi yang dilakukan guna mendapatkan daerah yang diduga sebagai obyek.

Filter-filter deteksi tepi yang selama ini kita kenal seperti roberts, laplacian, sobel, dll digunakan sebagai *single* proses untuk menghasilkan citra tepi. Hal ini disebabkan kernel dari filter-filter tersebut memang dirancang untuk mengembalikan nilai *output* yang peka terhadap tepi. Penelitian ini menawarkan suatu alternatif teknik deteksi tepi lain berdasar kombinasi operasi morfologi. Operasi morfologi yang bertujuan utama untuk memperbaiki struktur dan bentuk obyek jika digunakan dengan kombinasi yang tepat menghasilkan citra tepi tanpa harus menggunakan filter deteksi tepi.

Pengolahan citra (*image processing*) telah diaplikasikan di semua bidang, dari bidang kedokteran sampai bidang militer. Pengolahan citra digunakan untuk memproses citra atau gambar dengan jalan memanipulasinya menjadi data gambar yang diinginkan untuk mendapatkan informasi tertentu.

Obyek yang digunakan dalam dunia *Image Proccessing* sangat luas dan bebas sesuai kebutuhannya. Suatu objek untuk keperluan tertentu pada umumnya diolah dengan menggunakan suatu kamera untuk menangkap *Image* obyek yang akan diolah. Obyek yang diambil dalam proyek akhir ini semua benda yang akan diukur atau yang ingin diketahui panjang dari jarak jauh terutama benda-benda (obyek) yang berbahaya bila diukur dari jarak dekat.

Berdasarkan latar belakang teknologi dan metode di atas akan dirancang sebuah program pengukur panjang benda dari jarak jauh, karena tidak semua benda aman bila diukur dari jarak dekat.

## 1.2 Rumusan Masalah

Masalah yang ditangani pada penelitian ini adalah mendaya-gunakan pengolaan citra sebagai media pengukur jarak jauh sehingga aman digunakan untuk mengukur benda-benda yang berbahaya bila diukur dari jarak dekat. Adapun rumusan masalah pada penelitian ini sebagai berikut:

1. Analisa terhadap kebutuan-kebutuhan yang diperlukan untuk membangun sistem pengukur panjang dengan metode pengolahan citra.
2. Mengambil gambar benda yang akan di ukur dengan bantuan dua laser sebagai acuan untuk pengukuran.
3. Merancang program pengukur panjang dengan metode pengolahan citra digital

## 1.3 Batasan Masalah

Untuk memberi arah yang jelas pada penelitian ini maka diperlukan batasan masalah sebagai berikut:

1. Kamera dapat menangkap gambar dua titik laser yang ada pada benda yang akan di ukur (jarak maksimal laser dengan benda adalah 20 m).
2. Obyek yang diukur harus sejajar taehadap sumbu (x) dengan sensor kamera dan laser(sensor kamera dan sinar laser tegak lurus terhadap objek yang akan di ukur).
3. Pengambilan gambar di fokuskan pada objek yang akan di ukur (tidak melebar) sehingga keseluruhan obyek tertampung di gambar foto.

## 1.4 Tujuan

Tujuan dari Skripsi ini adalah mengaplikasikan pengolahan citra digital untuk teknologi pengukuran benda (objek) pada jarak jauh.

## 1.5 Manfaat

Adapun manfaat penelitian ini adalah Mengembangkan ilmu pengetahuan yang diperoleh dan mengaplikasikan untuk menyelesaikan permasalahan yang timbul dalam keadaan sebenarnya serta memberikan kemudahan bagi masyarakat untuk mengukur objek dari jarak jauh terutama untuk objek yang berbahaya.

## 1.6 Sistematika Penulisan

Pembahasan tugas akhir ini terdiri dari 7 bab yang disusun dengan sistematika penulisan sebagai berikut:

**Bab I Pendahuluan**

Memuat latar belakang, rumusan masalah, tujuan, batasan masalah, manfaat serta sistematika penulisan.

**Bab II Landasan Teori**

Bab ini akan menjelaskan tentang teori-teori yang menunjang penulisan laporan tugas akhir ini, seperti *image processing*, Geometri citra, Definisi citra, Deteksi tepi, laser, RGB dan bahasa pemrograman yang akan dipakai.

**Bab III Metodologi**

Membahas metodologi yang digunakan dalam penulisan yang terdiri dari studi literatur, analisis dan perancangan, implementasi, pengujian, pengambilan kesimpulan, dan penulisan laporan.

**Bab IV Perancangan**

Membahas perancangan program yang sesuai dengan teori yang ada serta membahas pembuatan dan cara kerja dari program yang dibuat.

**Bab V Implementasi**

Membahas lingkungan implementasi, diagram alir, algoritma sistem, proses pengambilan gambar dan tampilan-tampilan program.

**Bab VI Pengujian dan Analisa**

Membahas pengujian dan analisa program yang dibuat.

**Bab VII Penutup**

Memuat kesimpulan yang diperoleh dari pembuatan dan pengujian program, serta saran-saran untuk pengembangan lebih lanjut.

# BAB II
# DASAR TEORI

Bab ini akan membahas dasar teori untuk menunjang penulisan penelitian mengenai pengembangan pengukur panjang dengan metode pengolahan citra dengan menggunakan delphi sebagai program yang akan digunakan untuk membuat sisitem tersebut. Beberapa dasar teori yang dimaksud diantaranya adalah *image processing*, geometri citra, definisi citra, deteksi tepi (*Edge Detection*), Laser, RGB, delphi, analisis dan perancangan dengan DFD, teknik dan strategi pengujian.

## 2.1 *Image Processing*

*Image processing* atau pengolahan citra adalah bidang tersendiri yang sudah cukup berkembang sejak orang mengerti bahwa komputer tidak hanya dapat menangani data teks, tetapi juga data citra. Teknik-teknik pengolahan citra biasanya digunakan untuk melakukan transformasi dari satu citra kepada citra yang lain, sementara tugas perbaikan informasi terletak pada manusia melalui penyusunan algoritmanya.

Bidang ini meliputi penajaman citra, penonjolan fitur tertentu dari suatu citra, kompresi citra dan koreksi citra yang tidak fokus atau kabur. Sebaliknya, sistem visual menggunakan citra sebagai masukan tetapi menghasilkan keluaran jenis lain seperti representasi dari kontur objek di dalam citra, atau menghasilkan gerakan dari suatu peralatan mekanis yang terintegrasi dengan sistem visual.

Penekanan pada sistem visual adalah perbaikan dan pengambilan informasi secara otomatis dengan interaksi manusia yang minimal. Algoritma pengolahan citra sangat berguna pada awal perkembangan sistem visual, biasanya digunakan untuk menajamkan informasi tertentu pada citra, sebelum diolah lebih jauh [NUA-05:4].

*Computer graphics* melalui pemrograman grafik menghasilkan citra dari bentuk geometri yang primitif seperti titik, garis lurus dan garis lengkung, lingkaran dan bentuk-bentuk dasar geometri lainnya. Komputer grafik memainkan

peranan penting dalam visualisasi dan *virtual-reality*, sedangkan sistem visual bekerja sebaliknya, menduga bentuk geometri primitif dan ciri lainnya yang merupakan penyederhanaan dari citra asal yang sifatnya lebih kompleks.

Komputer grafik memadukan unsur-unsur pembentuk citra untuk membentuk atau mensintesa citra sedangkan sistem visual mengerjakan hal kebalikannya yaitu menganalisa citra dan terkadang menguraikannya menjadi bentuk-bentuk yang lebih sederhana, agar dapat dinilai secara kuantitatif .

*Pattern recognation* atau pengenalan pola adalah suatu proses atau rangkaian pekerjaan yang bertujuan mengklasifikasikan data numerik dan simbol. Banyak teknik statistik dan sintaksis yang telah dikembangkan untuk keperluan klasifikasi pola dan teknik-teknik ini dapat memainkan peran yang penting dalam sistem visual untuk pengenalan objek yang biasanya memerlukan banyak teknik. Bentuk-bentuk objek tertentu dalam dunia nyata yang sangat kompleks dapat dibandingkan dengan pola-pola dasar di dalam citra sehingga penggolongan objek yang bersangkutan dapat dilakukan dengan lebih mudah [NUA-05:5].

*Artificial intelligence* atau kecerdasan buatan berhubungan dengan rancangan sistem dengan kecerdasan dan aspek perhitungan kecerdasan. Kecerdasan buatan digunakan untuk menganalisis pemandangan dalam citra dengan perhitungan simbol-simbol yang mewakili isi pemandangan tersebut setelah citra diolah untuk memperoleh ciri khas. Kecerdasan buatan bisa dilihat sebagai tiga kesatuan yang terpadu yaitu persepsi, pengertian dan aksi.

Persepsi menerjemahkan sinyal dari dunia nyata dalam citra menjadi simbol-simbol yang lebih sederhana, pengertian memanipulasi simbol-simbol untuk memudahkan penggalian suatu informasi tertentu, dan aksi menerjemahkan simbol-simbol yang telah dimanipulasi menjadi sinyal yang lain, yang merupakan hasil akhir atau hasil antara, sesuai dengan keperluan banyak teknik dari kecerdasan buatan yang berperan penting dalam segala aspek dalam sistem visual. Sesungguhnyalah bahwa sistem visual sering dianggap sebagai sub-bidang dari kecerdasan buatan [NUA-05:5-6].

Secara ringkas dapat dikatakan bahwa sistem visual menghasilkan pengukuran atau abstraksi dari sifat-sifat geometri pada citra dan melahirkan suatu

interpretasi tertentu. Dengan demikian, dalam memahami sistem visual akan sangat berguna untuk diingat bahwa:

visual = geometri + pengukuran + interpretasi

[NUA-05:6].

## 2.2 Geometri Citra

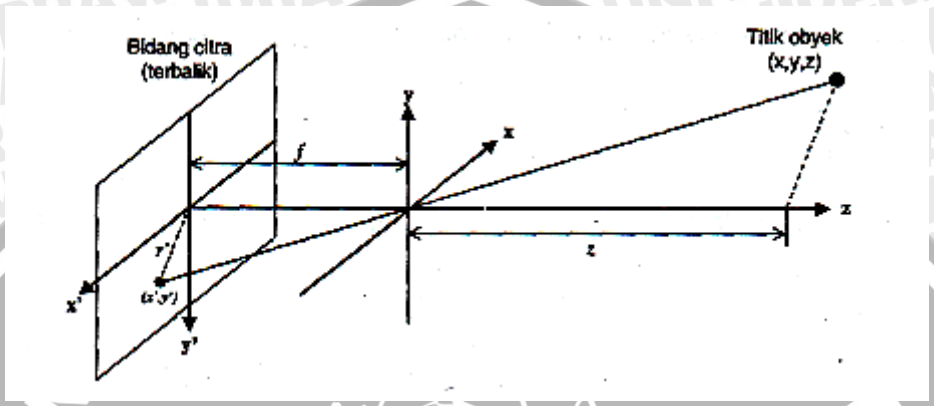
Untuk dapat mengerti dan melakukan operasi pengolahan citra, pertama kali kita harus memahami dengan baik apa dan bagaimana sifat-sifat citra itu sendiri. Ada dua hal penting dan sangat mendasar pada proses pembentukan citra yang harus dipahami dan selanjutnya sangat perlu untuk terus diingat, yaitu:

1. Geometri formasi citra yang menentukan lokasi suatu titik dalam pemandangan yang diproyeksikan pada bidang citra, dan
2. Fisik cahaya, yang menentukan kecerahan suatu titik pada bidang citra sebagai fungsi pencahayaan pemandangan dan sifat-sifat permukaan.

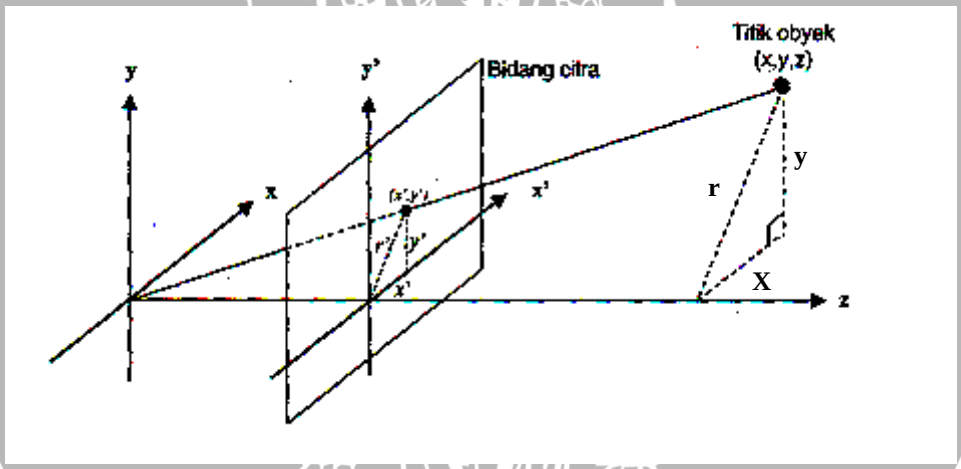
Untuk lebih memahami syarat pembentukan suatu citra di atas, bayangkanlah bahwa anda mempunyai sejumlah batu kerikil yang berwarna-warni, bila anda sebarkan begitu saja sejumlah batu kerikil tadi dalam sebidang tanah tidak akan terbentuk suatu citra yang bermakna, tetapi bila anda menyusunnya sedemikian rupa, sehingga masing-masing kerikil mempunyai posisi tertentu, mungkin akan terbentuk suatu citra yang bermakna. Demikian pula bila susunannya diubah mengikuti pola yang lain, mungkin akan terbentuk citra yang lain pula. Itulah pentingnya formasi atau susunan titik-titik pembentuk citra dengan lokasinya masing-masing [NUA-05:8].

Model dasar untuk proyeksi suatu titik pada pemandangan ke dalam bidang citra diperlihatkan pada Gambar 2.1a. Pada model ini pusat proyeksi sistem pembentukan citra berpotongan dengan titik awal dari koordinat sistem tiga-dimensi, yaitu x, y dan z. Posisi horizontal diwakili oleh x, posisi vertikal oleh y dan jarak dari kamera ke suatu titik objek oleh z. Garis pemandangan dari suatu titik dalam pemandangan adalah sebuah garis yang menyentuh titik tersebut

dan titik pusat proyeksi, sedangkan jarak dari titik ke kamera dinyatakan dengan z, yang sejajar dengan sumbu z. Dalam sistem optik pada kamera yang sesungguhnya, citra hasil pembentukan berada di belakang pusat proyeksi dengan jarak/ Untukmemudahkan, dalam ilustrasi ini diasumsikan bahwa bidang citra berada di depan pusat proyeksi seperti diperlihatkan pada Gambar 2.1 b.




(a)




(b)

**Gambar 2.1** Diagram proyeksi pembentukan citra; (a) citra di belakang pusat proyeksi, (b) citra di depan pusat proyeksi.

Jarak suatu titik *(x,y,z)* dalam pemandangan dari sumbu z dinyatakan dengan $r = \sqrt{(x^2 + y^2)}$, sedangkan jarak titik hasil proyeksi pada citra *(x',y')* dinyatakan dengan $r' = \sqrt{(x'^2 + y'^2)}$ Selanjutnya terdapat hubungan:

$$\frac{f}{z} = \frac{r'}{r} \tag{2.1}$$

$$\frac{x'}{x} = \frac{y'}{y} = \frac{r'}{r} \tag{2.2}$$

Penggabungan persamaan (2.1) dan persamaan (2.2) menghasilkan:

$$\frac{x'}{x} = \frac{f}{z} \text{ dan } \frac{y'}{y} = \frac{f}{z} \tag{2.3}$$

Dengan demikian posisi suatu titik di dalam citra diberikan dengan persamaan berikut:

$$x' = \frac{f}{z} x \tag{2.4}$$

$$y' = \frac{f}{z} y \tag{2.5}$$

Uraian di atas mengasumsikan bahwa pusat proyeksi mempunyai titik pertemuan pada titik awal dari sumbu tiga-dimensi. Operasi pengolahan citra pada sistem visual titik awal bidang citra terletak pada posisi kiri atas. Dengan demikian koordinat pada bidang citra merupakan koordinat absolut.

Hal ini perlu diingat dengan baik karena berbeda dengan koordinat yang dipakai pada ilmu matematika yang sudah dipahami dengan baik, yaitu titik awal suatu grafik berada pada sudut kiri bawah [NUA-05:8-10]

## 2.3 Definisi Citra

Citra (*image*) adalah gambar pada bidang dwimatra (dua dimensi). Ditinjau dari sudut pandang matematis, citra merupakan fungsi menerus (*continue*) dari intensitas cahaya dan pada bidang dwimatra. Sumber cahaya menerangi objek, objek memantulkan kembali sebagian dari berkas cahaya tersebut. Pantulan cahaya ini ditangkap oleh alat-alat optik, mata pada manusia kamera, pemindai (*scanner*), dan sebagainya sehingga bayangan objek yang disebut citra tersebut terekam [RNM-04:2].

Mengerti hubungan formasi geometri citra dan representasi citra di dalam komputer adalah penting untuk memahami bagaimana citra digital disimpan dan diolah. Harus ada jembatan antara notasi matematika untuk mengembangkan algoritma pengolahan citra dan notasi algoritma yang digunakan dalam pembuatan program komputer. Untunglah komputer mempunyai sistem penyimpanan memori dua dimensi yang disebut larik *(array)* atau matriks memori [NUA-05:14].

*Pixel* merupakan singkatan dari *picture elment*, kadang-kadang disebut juga pel. Pixel juga bisa diartikan satuan titik dalam satu *grid* berbentuk persegi atau juga beribu titik yang secara individual "dilukis" menjadi suatu bentuk *image* yang dihasilkan pada layar komputer atau pada *printer*.

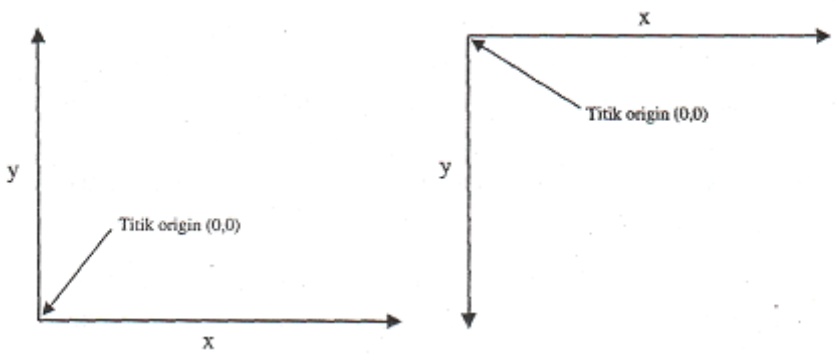
Seperti hanya *bit*, yakni unit informasi terkecil yang bisa diproses oleh komputer, sebuah *pixel* adalah elemen terkecil dari perangkat keras pencetak atau display, seperti *monitor* dan bagi perangkat lunak yang bisa memanipulasi untuk bisa menghasilkan image berupa huruf-huruf angka atau grafik[CJS-07:16].

Sebuah piksel adalah sampel dari pemandangan yang mengandung intensitas citra yang dinyatakan dalam bilangan bulat. Sebuah citra adalah kumpulan piksel-piksel yang disusun dalam larik dua-dimensi. Indeks baris dan kolom (x,y) dari sebuah piksel dinyatakan dalam bilangan bulat. Piksel (0,0) terletak pada sudut kiri atas pada citra, indeks *x* bergerak ke kanan dan indeks *y* bergerak ke bawah. Konvensi ini dipakai merujuk pada cara penulisan larik yang digunakan dalam pemrograman komputer.

Hal yang berlawanan untuk arah vertikal berlaku pada kenyataan dan juga pada sistem grafik dalam matematika yang sudah lebih dulu dikenal. Gambar 2.2. memperlihatkan perbedaan kedua sistem ini. Hubungan antara koordinat citra dan indeks larik untuk citra yang bersangkutan dapat dilihat pada gambar 2.3. pada contoh ini citra ditunjukkan dalam bentuk diagram berupa kumpulan segi-empat dengan sisi-sisi yang sama. Ini disebut *square tesselation*, sebuah teknik yang umum digunakan dalam menggambarkan bagaimana citra digital terbentuk.

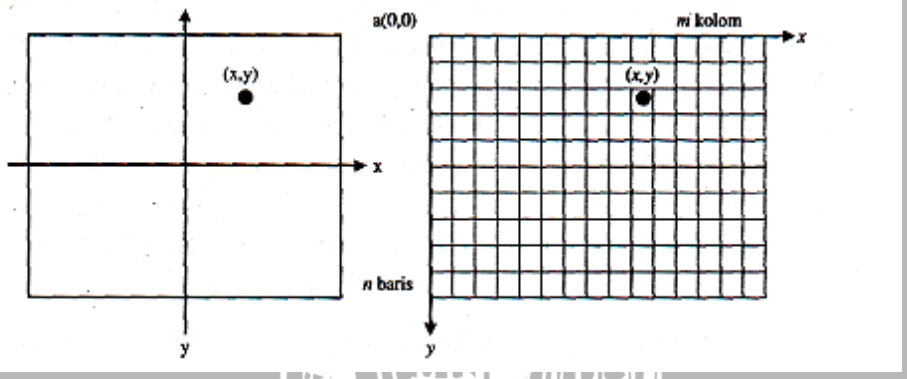
Selain bujur sangkar, bentuk lain seperti segitiga, segienam atau yang lainnya dapat saja digunakan untuk merangkai citra, akan tetapi bentuk segiempat dan bujur sangkar adalah yang paling sederhana dan mudah dipahami. Hal ini

tidak menimbulkan masalah karena *tesselation* hanyalah teknik visualisasi dari citra yang tersimpan dalam memori komputer sebab kenyataannya di dalam komputer memori yang berisi data intensitas citra tidak terletak berdampingan seperti pada teknik *tesselation.*




(a) (b)

**Gambar 2.2** Perbedaan letak titik origin pada koordinat grafik dan pada citra; (a) koordinat pada grafik matematika, (b) koordinat pada citra .



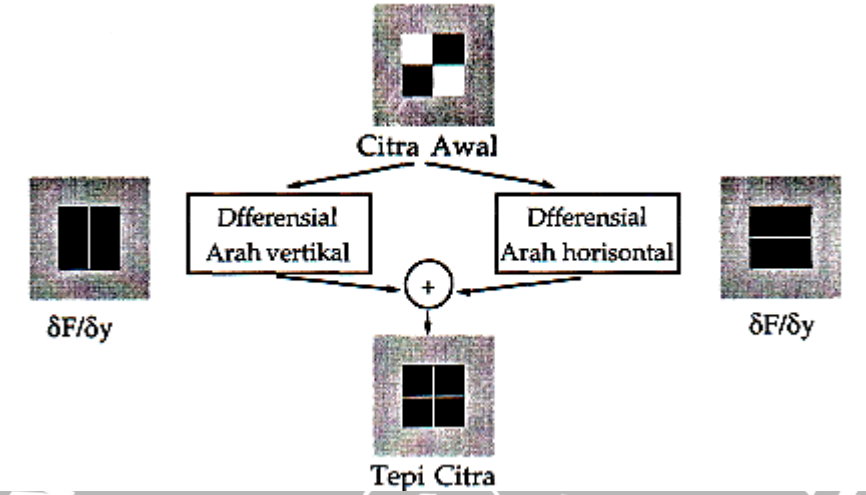

**Gambar 2.3** Hubungan antara koordinat pada citra dan indeks larik pada komputer untuk menyimpan data citra.

Hal yang lebih penting adalah bagaimana kita mengakses set piksel sebagai unit terkecil dari citra yang disimpan dalam memori komputer sehingga operasi atau manipulasi pada citra dapat dilakukan seperti kita memanipulasi kotak-kotak pada citra yang terbentuk dari hasil *tessellation* [NUA-05:14-16].

## 2.4 Deteksi Tepi

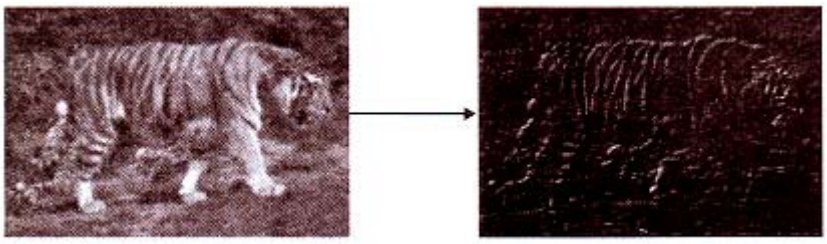
Deteksi tepi *(Edkge Detection)* pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari objek-objek gambar. Suatu titik (x,y) dikatakan

sebagai tepi *(edge)* dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangga. Gambar 2.4 menggambarkan bagaimana tepi suatu gambar diperoleh.




**Gambar 2.4** Proses deteksi tepi

Hasil deteksi dari beberapa citra menggunakan model differensial di atas.



**Gambar 2.5** Hasil beberapa deteksi tepi .

Pada gambar 2.5 terlihat bahwa hasil deteksi tepi berupa tepi-tepi dari suatu gambar. Bila diperhatikan bahwa tepi suatu gambar terletak pada titik-titik yang memiliki perbedaan tinggi. Berdasarkan prinsip-prinsip filter pada citra maka tepi suatu gambar dapat di-peroleh menggunakan High Pass Filter (HPF), yang mempunyai karakteristik:

$$\sum_x \sum_y H(x,y) = 0$$

**Contoh:**

Diketahui fungsi citra f(x,y) sebagai berikut.

| | | | | |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |

Dengan menggunakan filter:

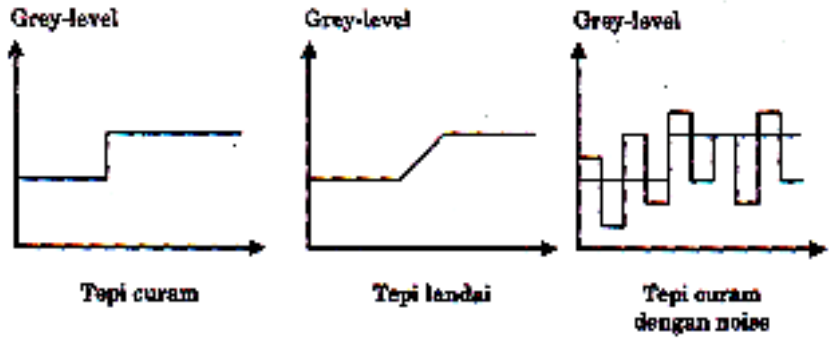
| | | | | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |

Bila digambarkan maka proses filter di atas mernpunyai masukan dan keluaran sebagai berikut.

**Gambar 2.6** Input dan output edge detection.

[FPB-05:150-151].

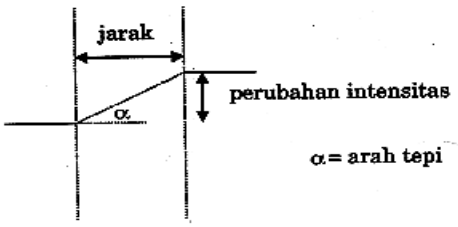
Pada buku lain di katakana bahwa deteksi tepi adalah perubahan intensitas grey-level secara mendadak, dalam jarak yang singkat. Ada tiga macam tepi (edge) yang sering muncul di dalam citra digital: tepi curam, tepi landai, dan tepi yang mengandung noise.




**Gambar 2.7** jenis-jenis tepi.

Untuk mendeteksi keberadaan tepi-tepi pada citra digunakan berbagai teknik berikut:

1. Operator gradien diferensial
2. Operator turunan kedua *(Laplace)*
3. Operator kompas




**Gambar 2.8** Model tepi satu dimensi

**Operator gradien diferensial** berbagai operator yang termasuk dalam kategori operator gradien diferensial adalah :

1. Operator selisih pusat:

   Dengan template :

$$D_x(x,y) = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \text{ dan } D_x(x,y) = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$$

2. Operator Sobel:

   Tinjau berbagai pixel di sekitar pixel *{x,y)*

$$\begin{matrix} a_0 & a_1 & a_2 \\ a_7 & (x,y) & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

Operator sobel merupakan magnitude dari gradian

$$M = \sqrt{s_x^2 + s_y^2}$$

Turunan parsial dihitung dengan

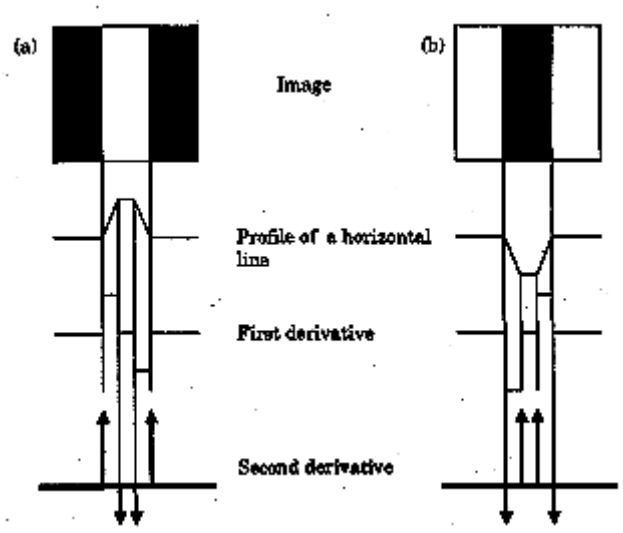
$$s_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6)$$

$$s_y = (a_0 + ca_1 + a_{22}) - (a_6 + ca_5 + a_4)$$ dengan konstanta $c = 2$ dalam bentuk mask $s_x$ dan $s_y$ dinyatakan sebagai:

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } s_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

3. Operator Prewitt

$$s_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \text{ dan } s_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$




**Gambar 2.9.** Elemen-elemen edge detection menggunakan operator derivative, (a) pencahayaan objek pada background gelap. (b) penggelapan objek pada pencahayaan background

4. Operator Robert:

Gradien operator Robert dalam arah- *x* dan arah- y dihitung menggunakan formulasi berikut:

$$R_{+}(x,y) = f(x+1,y+1) - f(x,y)$$

$$R_{-}(x,y) = f(x,y+1) - f(x+1,y)$$

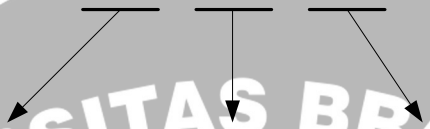
[ZTF-08:53-56].

Pada sistem pengukur panjang dengan metode pengolahan citra digital akan digunakan deteksi tepi dengan metode perulangan, yaitu membedakan warna (RGB) objek dengan backgroudnya dengan cara melakukan pengecekan dari koordinat dari *pixel* dari objek sampai dengan koordinat *pixel* terbesar, jika selisih untuk nilai R,G,B objek dengan backgroundnya lebih kecil dari 10 byte maka akan dianggap warna objek dan jika selisih untuk nilai R,G,B objek dengan backgroundnya lebih besar dari 10 byte maka akan dianggap warna backgroundnya. Contoh algoritma prosedur untuk deteksi tepi kanan objek:

```
RGB :=  image1.Canvas.Pixels[x,y];
k := GetRValue(RGB);
l := GetGValue(RGB);
m := GetBValue(RGB);
FOR i :=  x TO gambar.Width-2  DO
FOR j :=  1 TO gambar.Height-2 DO
BEGIN
  RGB :=  image1.Canvas.Pixels[i,j];
  R := GetRValue(RGB);
  G := GetGValue(RGB);
  B := GetBValue(RGB);
  d:=k-r;
  e:=l-g;
  f:=m-b;
  IF d<0 THEN
  BEGIN
         d:=d*(-1);
  END;
  IF e<0 THEN
  BEGIN
         e:=e*(-1);
  END;
  IF f<0 THEN
  BEGIN
         f:=f*(-1);
  END;
  u:=10;
  IF ((d<u)AND(f<u)AND(e<u))THEN
  BEGIN
         edit4.text:= intTOstr(i);
  END;
END;
```

## 2.5 Pengolah Warna RGB

Dasar pengolahan citra adalah pengolahan warna RGB pada posisi tertentu. Pengolahan citra dipresentasikan dengan nilai heksadesimal dari 0x00000000 sampai 0x00ffffffff. Warna hitam adalah 0x00000000 dan warna putih adalah 0x00ffffff. Gambar 2.10 menunjukkan definisi nilai warna dan variable 0x00 menyatakan anka dibelakangnya adalah heksadesimal.



**Gambar 2.10** Nilai Warna RGB dalam Hexadesimal

Setiap warna mempunyai range nilai 00 (angka desimalnya adalah 0) dan ff (angka desimalnya 255) atau mempunyai nilai derajat keabuan 256 =$2^8$. Dengan demikaian, range warna yang digunakan adalah ($2^8$) ($2^8$) ($2^8$) = $2^{24}$ (atau dikenal dengan istilah true colour pada windowas. Nilai warna yang digunakan merupakan gabungan warna cahaya merah,hijau,dan biru [NBR-05:11-12].

0x00 XX

## 2.6 Laser

Laser (Light Amplification by Stimulated Emission of Radiation). Upaya yang dilakukan untuk meningkatkan intensitas pancaran cahaya pada spektrum tertentu sehingga mampu mencapai jarak yang jauh dan terarah dengan tepat dengan suatu perangkat [LSR-09]. Pada sistem pengukur panjang dengan metode pangolahan citra warna (RGB) dari titik tengah laser yang tertangkap oleh kamera adalah `(R=255)AND(G<250)AND(B<250).`

Nialai R

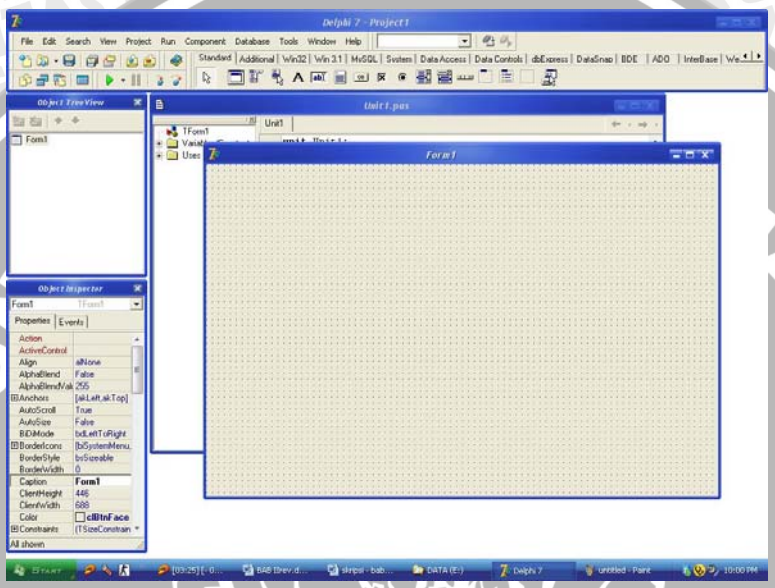
## 2.7 Delphi

Delphi adalah sebuah bahasa pemrograman dan lingkungan pengembangan perangkat lunak. Produk ini dikembangkan oleh Borland (sebelumnya dikenal sebagai Inprise). Bahasa Delphi, yang sebelumnya dikenal sebagai *object pascal* (pascal dengan ekstensi pemrograman berorientasi objek

(PBO/OOP)) pada mulanya ditujukan hanya untuk Microsoft Windows, namun saat ini telah mampu digunakan untuk mengembangkan aplikasi untuk Linux dan Microsoft [IDE-07].

### 2.7.1. Area Kerja Borland Delphi 7

Adapun area kerja Borland Delphi 7 adalah seperti gambar 2.11 di bawah ini:



**Gambar 2.11.** Area Kerja Delphi.

Area kerja tersebut terdiri dari:

1. Menu Utama

   Menu utama terletak pada bagian paling atas dari area kerja. Menu uama antara lain terdiri dari submenu : File, Edit, Search, View, Project, Run dan seterusnya sampai submenu yang paling akhir yaitu: Help.


**Gambar 2.12.** Menu Utama.

2. *Toolbar*

   Pada menu toolbar terdiri dari ikon-ikon shortcut dari New Project, Save dan Run.

**Gambar 2.13.** *Toolbar*.

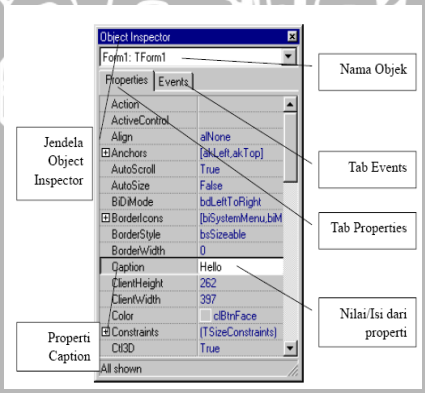
3. *Form Designer*

   *Form designer* merupakan area dimana akan dibuat project, di sini dapat diatur penempatan button-button, label, dan layout dari tampilan visual dari program yang akan dibuat.


**Gambar 2.14.** *Form designer*.

4. *Object Inspector*

   *Object Inspector* berisi properti-properi dari form dan komponen/objek yang digunakan. Pada bagian ini terdapat peengaturan properti-properti dari form dan komponen/objeck tersebut, antara lain: caption,ukuran, warna dan font pada form.




**Gambar 2.15.***Object inspector*

5. *Component Palette*

   Semua komponen-komponen yang ada pada delphi atau yang terinstall pada delphi akan ditampilkan pada bagian component palette.


**Gambar 2.16.** *Component palette*

### 2.7.2. Meyimpan *Form*

Pada Delphi ada 3 buah file utama (***.dp**r, ***.pas** dan ***.dfm**). Masing-masing file tersebut mempunyai fungsi antara lain:

1. ***.dpr** adalah file proyek yang dibuat berisi program kecil untuk :
   1.1. Mendefinisikan Unit yang ada dalam file proyek
   1.2. Menginisialisasi data
   1.3. Membangun form
   1.4. Menjalankan aplikasi

   contoh isi **dpr :**

```
uses
Forms,
Unit1 in 'Unit1.pas' {Form1};
begin
Application.Initialize;
Application.CreateForm(Tform1, Form1);
Application.Run;
end.
```

2. ***.pas** adalah unit-unit (*pascal code file*), bisa terdiri satu atau banyak file
3. ***.dfm** adalah file definisi Form (*special pseudo code file*), bisa terdiri satu atau banyak file. Contohnya:

```
object Form1: Tform1
Left = 200
Top = 108
Width = 696
Height = 480
Caption = 'Form1'
Font.Charset = DEFAULT_CHARSET
Font.Color = clWindowText
Font.Height = -11
Font.Name = 'MS Sans Serif'
Font.Style = []
PixelsPerInch = 96
TextHeight = 13
object Button1: Tbutton
Left = 176
Top = 116
Width = 75
Height = 25
Caption = 'Button1'
TabOrder = 0
end
end
```

Setiap Form (.dfm) harus memiliki sebuah Unit (.pas), tetapi dapat memiliki Unit tanpa sebuah Form (hanya kode saja).

# BAB III
# METODE PENELITIAN

Bab ini menjelaskan langkah-langkah yang akan dilakukan dalam perancangan, implementasi dan pengujian dari aplikasi perangkat lunak yang akan dibuat. Kesimpulan dan saran disertakan sebagai catatan atas aplikasi dan kemungkinan arah pengembangan aplikasi selanjutnya.

## 3.1 Studi Literatur

Melakukan studi literatur yaitu dengan pengumpulan dan pendalaman bahan pustaka yang dibutuhkan dalam penyelesaian tugas akhir. Bahan pustaka tersebut membahas tentang:

1. *Image processing*
2. Geometri Citra
3. Definisi Citra
4. Deteksi tepi *(Edge Detection)*
5. Pengolahan Warna RGB
6. *Laser*
7. Delphi
8. Analisa dan Perancangan

## 3.2 Analisis dan Perancangan

Pada analisis dan perancangan terdiri dari dua tahap yaitu analisis kebutuhan perangkat lunak yang mencakup spesifikasi kebutuhan *user* dan sistem, serta tahap kedua adalah perancangan perangkat lunak pengukur panjang dengan metode pengolahan citra digital dengan menggunakan delphi.

Untuk merancang software citra digital sebagai media pengukur gambar ini, perlu ditentukan spesifikasi perangkat keras dan perangkat lunak yang akan digunakan. Spesifikasinya adalah sebagai berikut:

1. Perangkat Keras:
   1.1. 1 unit PC dengan spesifikasi tertentu, yang akan digunakan sebagai pembuat perangkat lunak citra digital.
   1.2. Kamera (kamera digital, kamera HP atau kamera lainya)
   1.3. Scanner (di buat memasukkan data gambar kedalam pc untuk kamera manual)
2. Perangkat Lunak:
   2.1. Dhelpi 7

### 3.3 Perancangan (*Design*)

Perancangan aplikasi dilakukan setelah semua kebutuhan sistem didapatkan melalui tahap analisis kebutuhan, perancangan dengan DFD, teknik dan strategi pengujian. Perancangan sistem ini akan didahului dengan pendefinisian pelaku atau user yang akan menggunakan progam pengukur panjang dengan metode pengolahan citra digital dan juga alat bantu yang digunakan.

### 3.4 Implementasi

Pada proses implementasi akan dilakukan pembuatan lingkungan implementasi untuk melakukan spesifikasi sistem. Kegiatan yang dilakukan adalah menerjemahkan dari bentuk perancangan yang ada dalam bentuk program ke dalam bahasa pemrograman pascal dengan menngunakan borland delphi 7.

### 3.5 Pengujian dan Analisis

Pengujian dilakukan untuk menemukan kesalahan aplikasi yang dibuat, sehingga sebelum aplikasi yang dibuat digunakan oleh pengguna aplikasi ini diharapkan dapat sedikit mungkin atau bahkan tidak ada kesalahan pada saat digunakan nantinya. Pengujian yang dilakukan pada benda yang telah diketahui ukuranya, kemudian di ambil gambarnya dan diuji dengan menggunakan software citra digital yang telah dibuat.

### 3.6 Pengambilan Kesimpulan

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

# BAB IV
# PERANCANGAN

Bab ini membahas perancangan dan pembuatan perangkat lunak untuk pengukur panjang dengan metode pengolahan citra. Pada bab ini juga akan dibahas mengenai proses deteksi titik-titik laser dan batas-batas tepi yang ada pada gambar yang akan diukur sehingga akan diperoleh informasi panjang gambar sebenarnya.

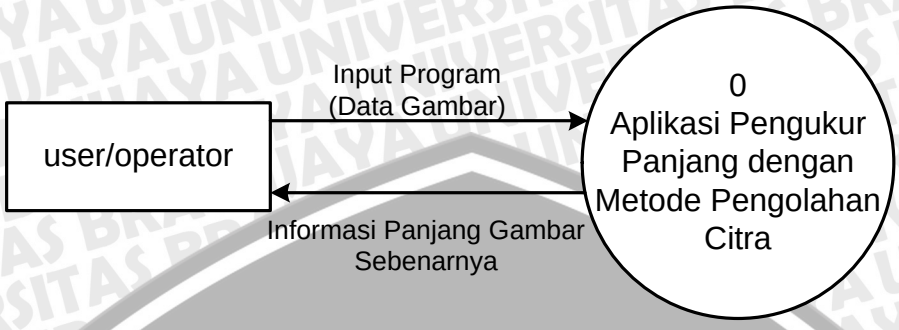
## 4.1 Perancangan Sistem

Perancangan sistem merupakan tahap awal dari perancangan perangkat lunak. Perancangan ini dilakukan untuk mengetahui aplikasi sistem yang akan dibuat secara umum. Perancangan sistem ini akan didahului dengan pendefinisian pelaku atau user yang akan menggunakan progam pengukur panjang dengan metode pengolahan citra digital dan juga alat bantu yang digunakan, yaitu:

1. User : Pelaku yang akan mengambil citra dan juga menjalankan program dengan tujuan untuk mengetahui panjang dari citra yang di ambil.
2. Laser : Alat yang akan dipancarkan cahayanya oleh user pada citra yang akan diukur sebagai acuan untuk mengetahui panjang objek.
3. Kamera : Alat yang akan digunakan user untuk mengambil gambar dari citra atau objek yang akan diukur.
4. Program : Perangkat lunak yang digunakan untuk mengetahui panjang citra/ atau objek yang akan diukur.

### 4.1.1 Diagram Konteks

Diagram konteks adalah sebuah diagram sederhana yang menggambarkan hubungan dengan entitas luar, masukan dan keluaran dari sistem. Diagram konteks aplikasi pengukur panjang dengan metode pengolahan citra ditunjukkan pada gambar 4.1.

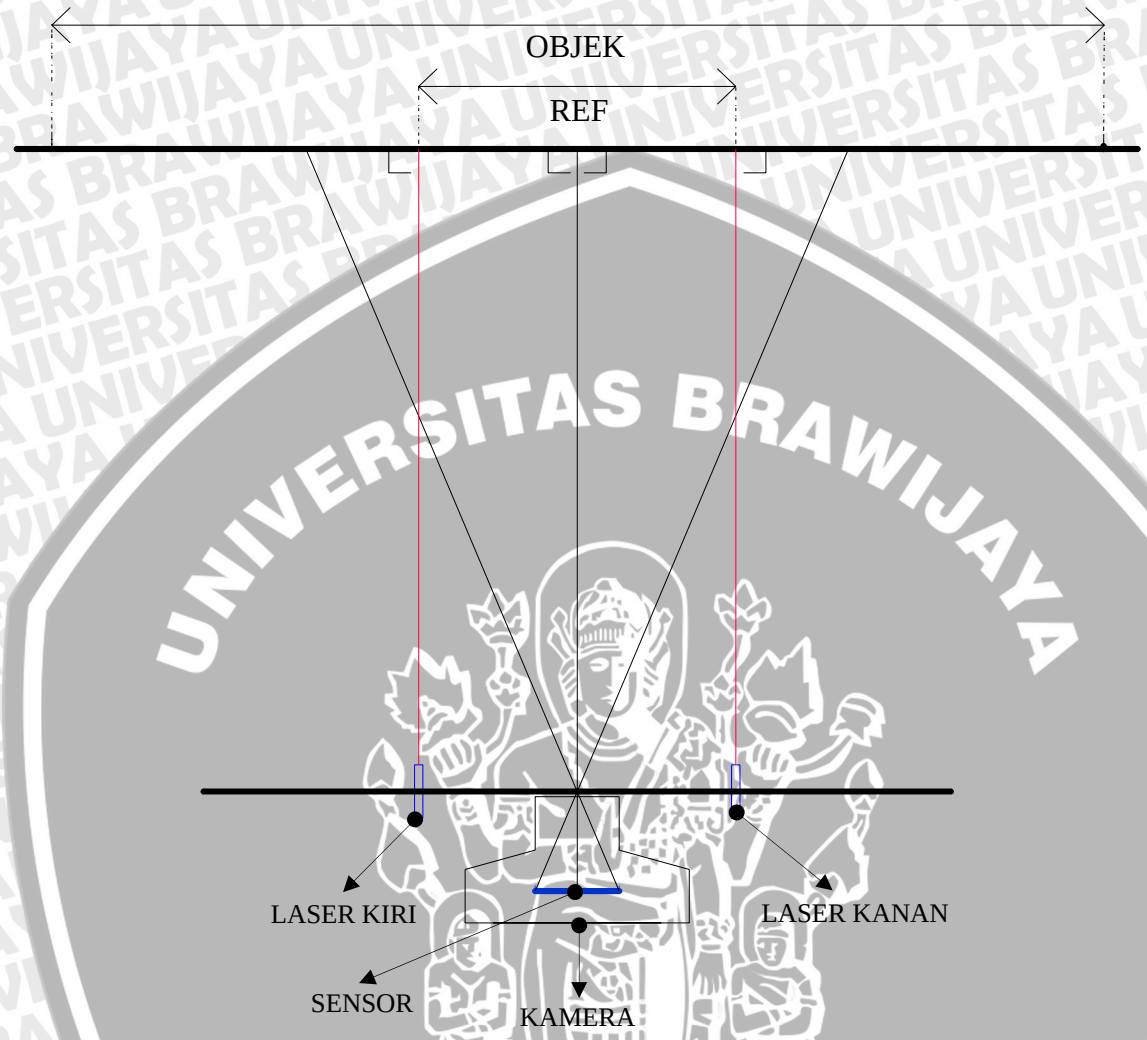



**Gambar 4.1** Diagram Konteks.

## 4.2 Cara Kerja Sistem

Cara kerja dari aplikasi pengukur panjang dengan metode pengolahan citra adalah sebagai berikut:

1. User mengambil gambar dari objek yang akan diukur dengan kamera dan dua laser yang telah dirancang sebagai acuan untuk pengukuran panjang gambar seperti pada gambar 4.2.
2. Gambar yang telah diambil oleh kamera dan laser tersebut digunakan sebagai input dari program.
3. Keluaran dari program adalah informasi-informasi mulai dari posisi-posisi titik laser, tepi objek,jarak antara titik-titik laser, jarak antara tepi gambar (panjang objek dalam gambar) sehingga akan ditemukan informasi yang diinginkan yakni panjang benda atau objek yang diukur.



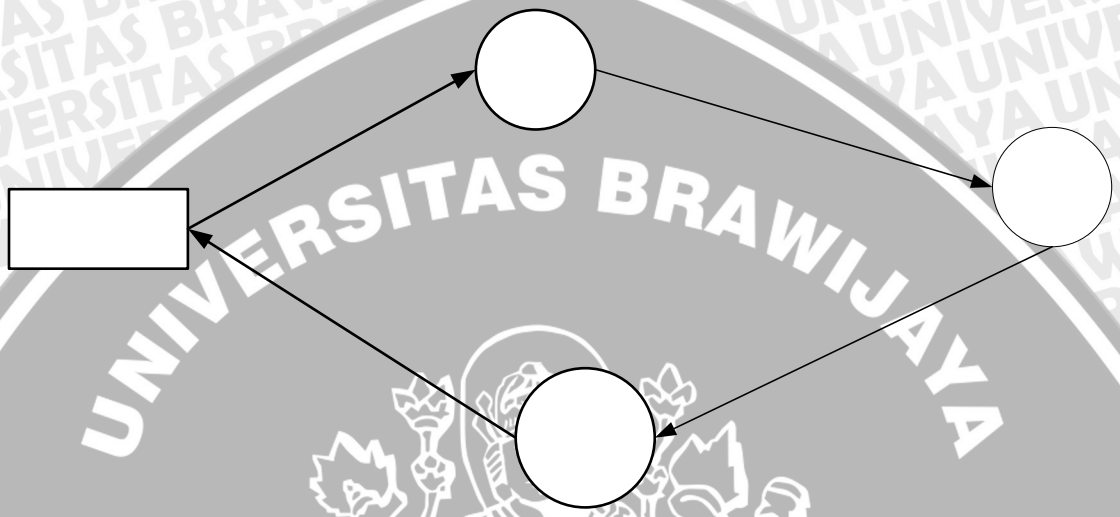

**Gambar 4.2** Proses pengambilan citra dengan bantuan dua laser.

Proses-proses dari aplikasi pengukur panjang dengan metode pengolahan citra akan di gambarkan dengan Data Flow Diagram (DFD).

## 4.3 Perancangan Proses

Perancangan proses menjelaskan masukan dan keluaran dari setiap proses yang terjadi pada aplikasi pengukur panjang dengan metode pengolahan citra. Perancangan proses yang dilakukan meliputi Data Flow Diagram (DFD) dan Diagram alir proses.

### 4.3.1 Data Flow Diagram level 1 Proses Aplikasi Pengukur Panjang dengan Metode Pengolahan Citra




**Gambar 4.3** Data Flow Diagram Level 1 Aplikasi Pengukur Panjang dengan Metode Pengolahan Citra

Pada Data Flow Diagram level 1 aplikasi pengukur panjang dengan metode pengolahan citra ini terdiri dari tiga proses yaitu:

1. Proses load image yaitu proses dimana progam meminta kepada user untuk memasukkan input program berupa gambar yang telah diambil oleh user dengan bantuan kamera dan juga laser sebagai media bantu untuk acuan pengukuran.
2. Proses scan image yaitu proses dimana program akan melakukan penyekenan dari tiap-tiap pixel sehingga di temukan data informasi (jarak antara dua titik laser dan panjang objek dalam dalam gambar). Proses ini di bagi menjadi tiga yaitu Scan Auto, Semi Auto dan Manual yang akan di jelaskan pada DFD level 2 scan image.
3. Proses perhitungan Panjang objek sebenarnya yaitu proses perhitungan yang dilakukan program untuk mengetahui panjang objek sebenarnya yang meliputi

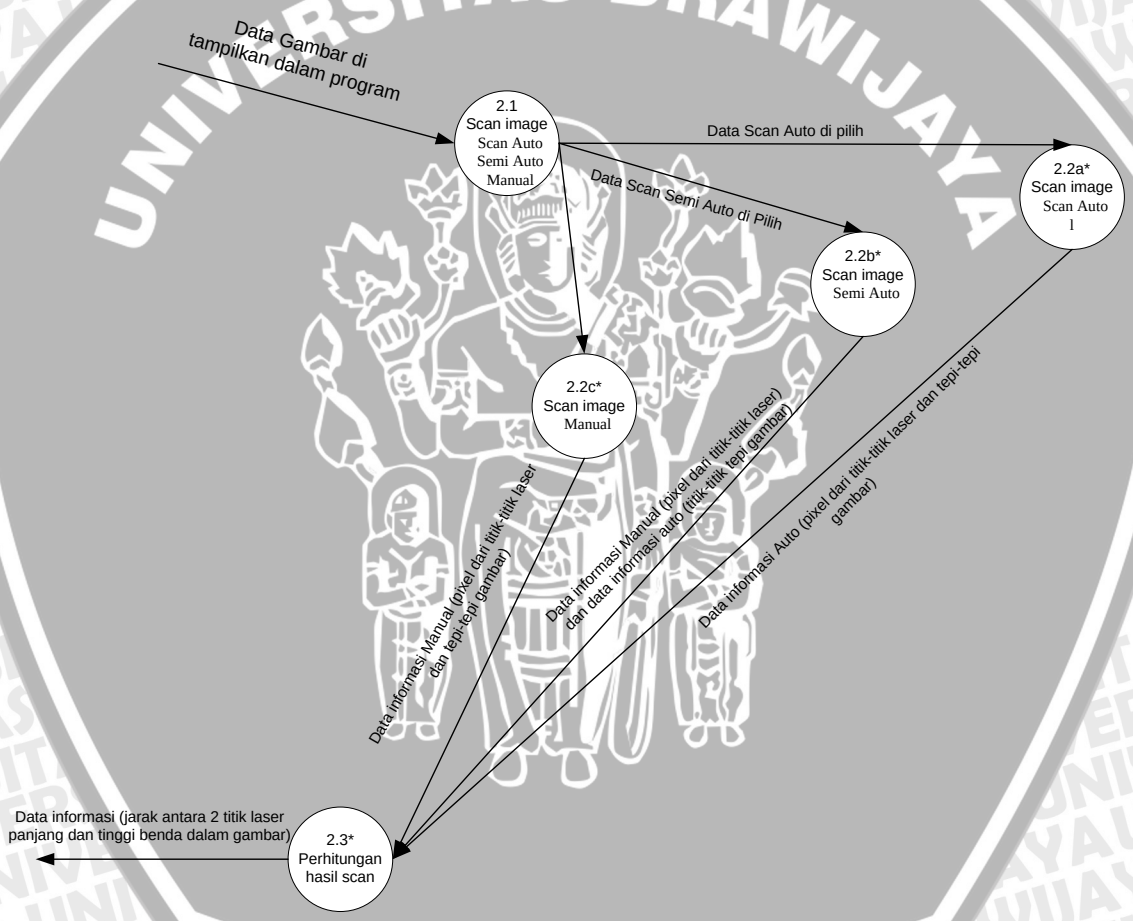
$$Z = Y * A$$

Z= Panjang objek sebenarnya (cm).

Y= Perbandingan antara panjang gambar dan jarak antara titik-titik laser.

A= jarak antara titik-titik laser sebenarnya (cm).

### 4.3.2 Data Flow Diagram level 2 Proses Aplikasi Pengukur Panjang dengan Metode Pengolahan Citra




**Gambar 4.4** Data Flow Diagram Level 2 Aplikasi Pengukur Panjang dengan Metode Pengolahan Citra

Pada DFD Level 2 Proses Scan Image Akan di bagi menjadi tiga bagian yaitu

Scan Auto, Semi Auto dan Manual kemudian akan dilanjutkan dengan Proses Perhitungan Hasil Scan yang akan dijelaskan sebagai berikut:

1. Proses Scan Auto yaitu proses dimana program akan mendeteksi secara otomatis dimana koordinat pixel dari titik-titik laser dan batas tepi gambar dari beckgroundnya sehingga akan ditemukan nilai dari $X_1$dan $X_2$.
2. Proses Semi Auto yaitu proses dimana program akan menyediakan drawing tools untuk user agar user bisa menentukan dimana koordinat pixel dari tititk-titik laser sehingga akan di temukan $X_1$, setelah $X_1$ di temukan maka program akan mendeteksi tepi-tepi gambar secara automatis sehingga $X_2$ ditemukan.
3. Proses Manual yaitu proses dimana program akan menyediakan menu drawing tools untuk user agar user bisa menentukan dimana koordinat pixel dari tititk-titik laser dan panjang yang mana yang akan diukur oleh user sehingga akan di temukan $X_1$ dan $X_2$.
4. Proses Perhitungan Hasil Scan yaitu proses dimana program melakukan perhitungan hasil scan sehingga di temukan $Y = \frac{X_2}{X_1}$

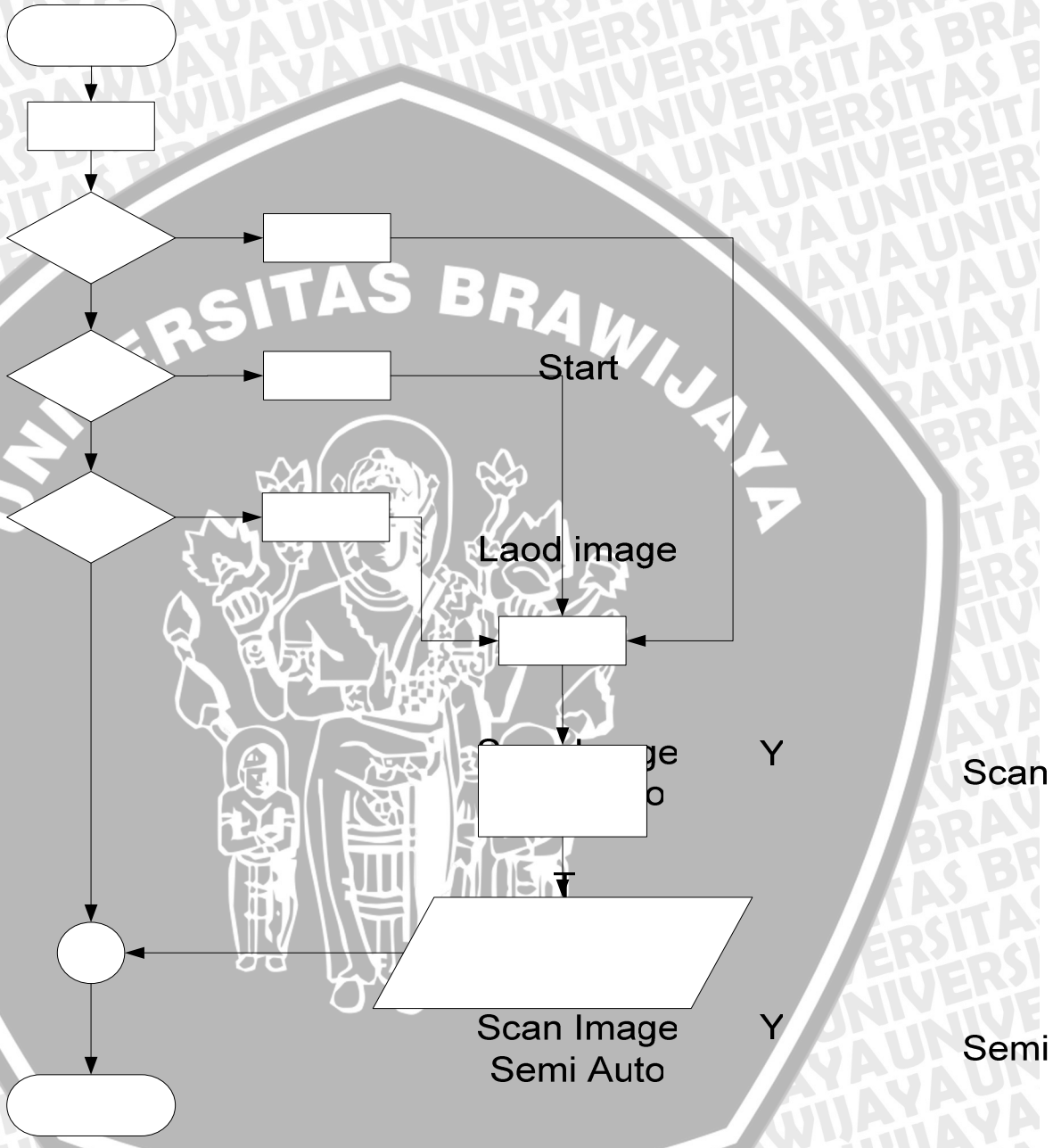
Keterangan:

Y = perbandingan antara panjang gambar dan jarak antara titik-titik laser.

$X_1$ = jarak antara dua titik alser dalam gambar (pixel).

$X_2$ = Panjang objek dalam gambar (pixel).

### 4.3.3 Diagram Alir Aplikasi Pengukur Panjang dengan Metode Pengolahan Citra




**Gambar 4.5** Diagram Alir Aplikasi Pengukur Panjang dengan Metode Pengolahan Citra

Proses dari diagram alir dapat dijelaskan sebagai berikut:

1. Program menyediakan form untuk memasukkan citra (objek yang akan diukur).
2. User memasukkan citra kedalam progam sebagai input program untuk

diketahui panjang dari citra tersebut.

3. Progam menyediakan menu untuk melakukan scan secara auto, semi auto dan manual.
4. User akan memilih dari ketiga menu yang telah disediakan oleh program.
5. Jika User memilih menu Scan Auto maka proses Scan Auto akan dijalankan.
6. Jika User memilih menu Semi Auto maka proses Semi Auto akan dijalankan.
7. Jika User memilih menu Manual maka proses Manual akan dijalankan.
8. Program akan menghitung data dari hasi scan, baik secara Auto, Semi Auto ataupun Manual yang akan menghasilkan perbandingan panjang antara jarak titik-titik laser dan panjang objek dalam gambar (Y).
9. Program Akan menghitung panjang sebenarnya (Z) yaitu mengalikan Y dengan A (jarak antara titik-titik laser sebenernya) atau ( $Z = Y * A$ ).
10. Setelah Nilai Z ditemukan maka akan ditampilkan oleh program dengan satuan centi meter (cm).

Dari tiga menu yang telah disediakan program, baik Scan Auto, Semi Auto ataupun Manual semuanya mengeluarkan output yang sama yaitu $X_1$ dan $X_2$, Penulis menyediakan tiga menu tersebut untuk mengatisipasi adanya gangguan gambar atau kerusakan gambar hasil pengambilan gambar oleh kamera, sehingga ketika user melakukan penyekenen secara automatis dan user tidak menemukan $X_1$, maka user bisa mnggunakan menu Semi auto yaitu dengan menetukan dimana titik-titik laser sehingga $X_1$ dapat di temukan kemudian program akan menemukan $X_2$ secara automatis. Dan jika $X_2$ tidak di temukan juga maka User bias menggunakan menu Manual dengan cara menentukan titik-titik laser dan juga tepi-tepi objek sehingga didapatkan $X_1$ dan $X_2$, jika $X_1$ dan $X_2$ sudah ditemukan maka program akan melakukan perhitungan sebagai berikut:

$$Y = \frac{X_2}{X_1}$$

$$Z = Y * A$$

## 4.4 Resolusi Sisitem

Resolusi adalah unit terkecil dari jumlah pengukuran terrendah yang dapat dideteksi. Pada sistem pengukur panjang dengan metode pengolahan citra unit terkecil yang dapat didetekesi adalah *pixel* dan satuan panjang terkecil yang digunakan adalah centimeter.

Contoh:

1. Untuk gambar berresolusi 3072 x 2304 diambil dari jarak terdekat (1 meter) dan diketahui panjang 30 cm dalam gambar terdiri dari 806 pixel, yang artinya panjang untuk setiap pixelnya adalah $30cm/806 = 0{,}037cm$.
2. Untuk gambar berresolusi 3072 x 2304 diambil dari jarak terjauh (20 meter) dan diketahui panjang 30 cm dalam gambar terdiri dari 60 pixel, yang artinya panjang untuk setiap pixelnya adalah $30cm/60 = 0{,}5cm$.

Sehingga dapat disimpulkan bahwa resolusi untuk pengukur panjang dengan metode pengolahan citra untuk jarak pengambilan gambar terdekat (1 meter) adalah 0,037cm dan untuk jarak terjauh (20 meter) adalah 0,5 cm.

# BAB V
# IMPLEMENTASI

Bab ini membahas mengenai implementasi perangkat lunak perangkat lunak untuk pengukur panjang dengan metode pengolahan citra. Berdasarkan hasil yang telah didapatkan dari proses perancangan perangkat lunak yang telah dibuat. Implementasi merupakan proses transformasi hasil perancangan perangkat lunak ke dalam kode *(coding)* sesuai dengan sintaks dari bahasa pemrograman yang digunakan. Pembahasan terdiri dari penjelasan tentang lingkungan implementasi, pengambilan gambar, algoritma implementasi dan beberapa kendala dalam implementasi.

## 5.1. Lingkungan Implementasi

Sistem dibuat dengan menggunakan aplikasi pemrograman Borland Delphi. Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

1. Komputer Administrasi

   Spesifikasi Hardware :

   1. CPU : AMD Athlon™ 64 3200+
   2. Memory : 1 GB

   Spesifikasi Software :

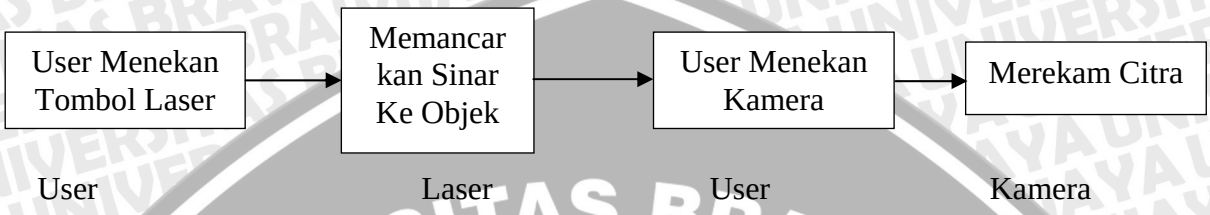
   1. Sistem operasi : Microsoft Windows XP Media Center Edition SP2
   2. Bahasa pemrograman : Borland Delphi 7

## 5.2. Algoritma Sistem

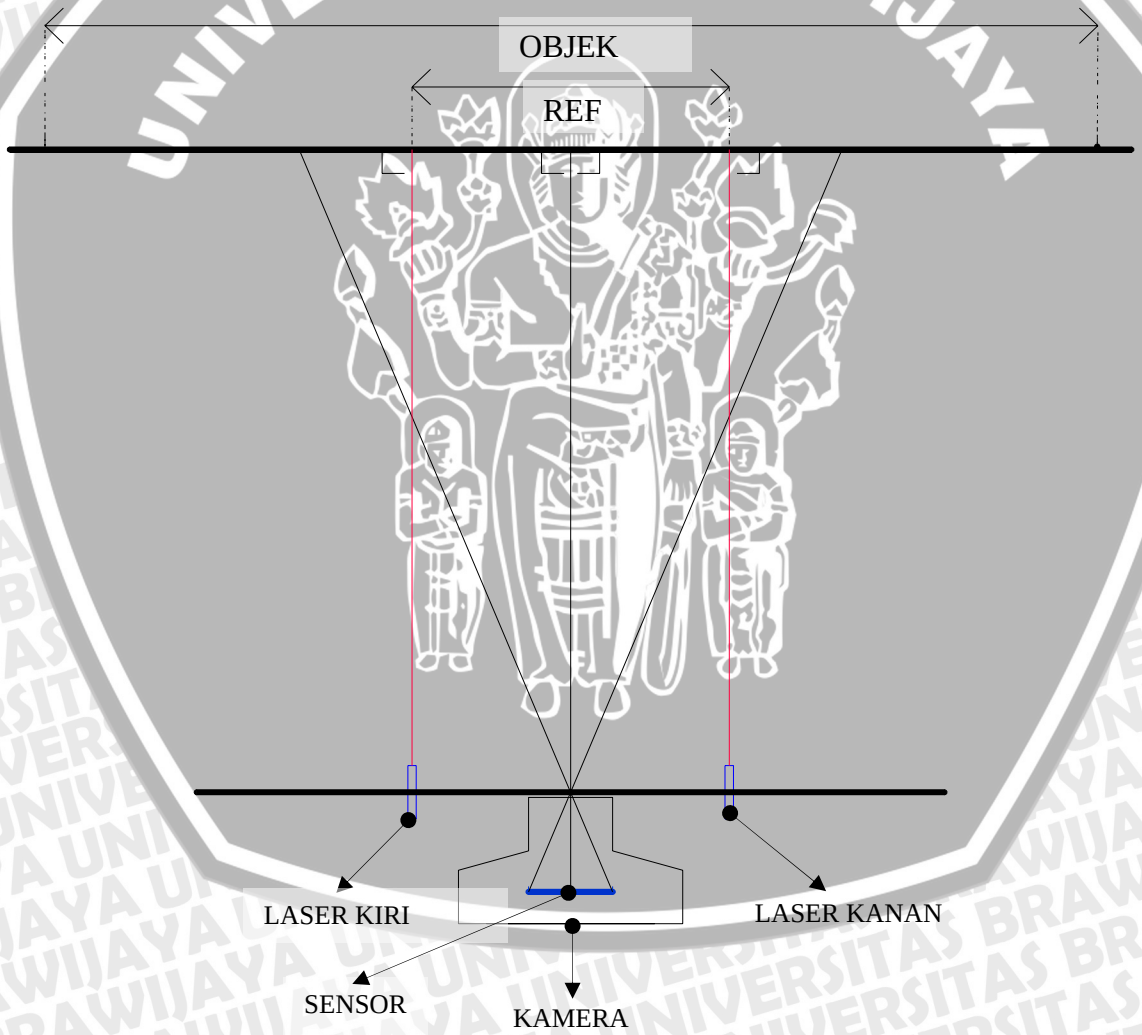
Penyajian yang digunakan berupa urutan baris algoritma seperti kode pemrograman dan tidak memiliki sintak yang baku atau biasa disebut *pseudocode*. Semua proses yang dijelaskan dalam bentuk diagram alir maupun *pseudocede* pada tahap Implementasi ini berdasarkan Perancangan Sistem pada tahap Perancangan.

### 5.2.1. Proses Pengambilan Gambar (Citra)

Proses pengambilan gambar (*capture*) menggunakan kamera pada jarak tertentu, dengan bantuan dua laser yang telah disusun sejajar dengan jarak 30 cm atau dengan jarak tertentu. Urutan pengambilan gambar digambarkan seperti gambar 5.1.



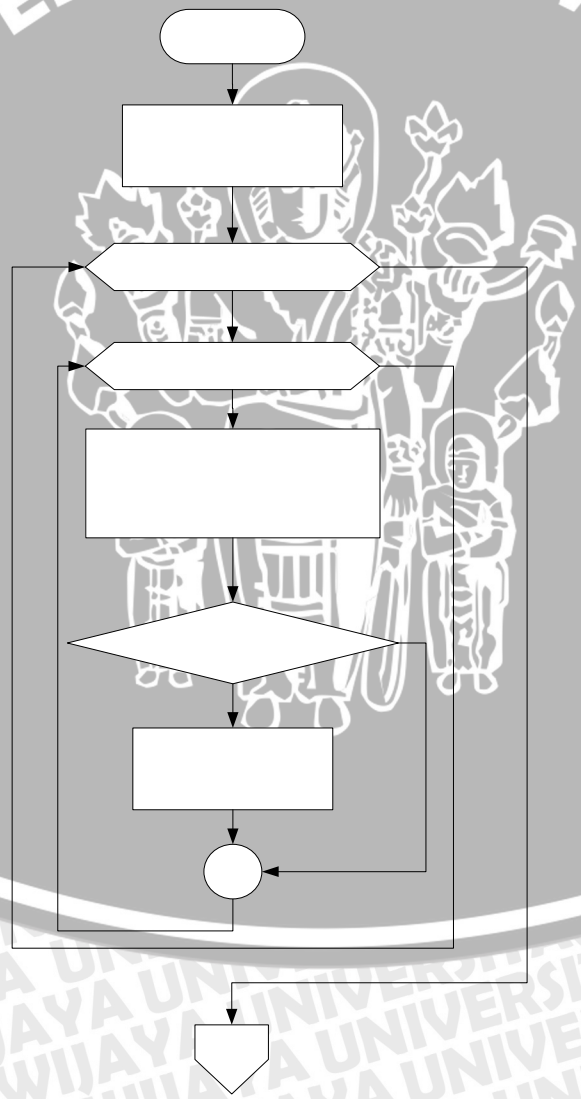

**Gambar 5.1** Urutan dari proses pengambilan citra.




**Gambar 5.2** Proses pengambilan citra dengan bantuan dua laser.

**Gambar 5.3** Hasil pengambilan gambar atau objek yang akan di ukur

### 5.2.2. Diagram Alir dan Procedure ketika dilakukan scan auto pada program



**Gambar 5.4** Diagram Alir Prosedur scan auto on click (deteksi titik laser sebelah kanan).

**Sumber:** [Implementasi].

```
1   procedure TProgram_utama.scanautoClick(Sender: TObject);
2       VAR
3       i,j,u,d,e,f:  INTEGER;
4       r,b,g,k,l,m :byte;
5       a:string;
6       BEGIN
7           gambar := TBitmap.Create;
8
9           TRY
10              gambar.PixelFormat := pf24bit;
11              gambar.Width  := Image1.Width;
12              gambar.Height := Image1.Height;
13
14              //deteksi titik laser sebelah kanan
15              FOR i :=  1 TO gambar.Width  DO
16              FOR j := 1 TO gambar.Height DO
17              BEGIN
18                  RGB :=  image1.Canvas.Pixels[i, j];
19                  R := GetRValue(RGB);
20                  G := GetGValue(RGB);
21                  B := GetBValue(RGB);
22
23                  IF (R>195)AND(G<5)AND(B<100)THEN
24                  BEGIN
25                      edit1.text:= intTOstr(i);
26                  END;
27              END;
28
```

**Gambar 5.5** Algoritma Prosedur scan auto on click (deteksi titik laser sebelah kanan).

**Sumber:** [Implementasi].

Penjelasan Algoritma Prosedur scan auto on click (deteksi titik laser sebelah kanan) pada baris 14 sampai dengan 27 pada gambar 5.5 sebagai berikut:

1. akan dilakukan pengecekan warna mulai dari koordinat pixel

   (1,1) ,(1,2) ,(1,3) ⟶ (1,$n_b$)  $n_b$ = tinggi gambar dalam pixel

   (2,1) ,(2,2) ,(2,3) ⟶ (2,$n_b$)

   ↓ ↓

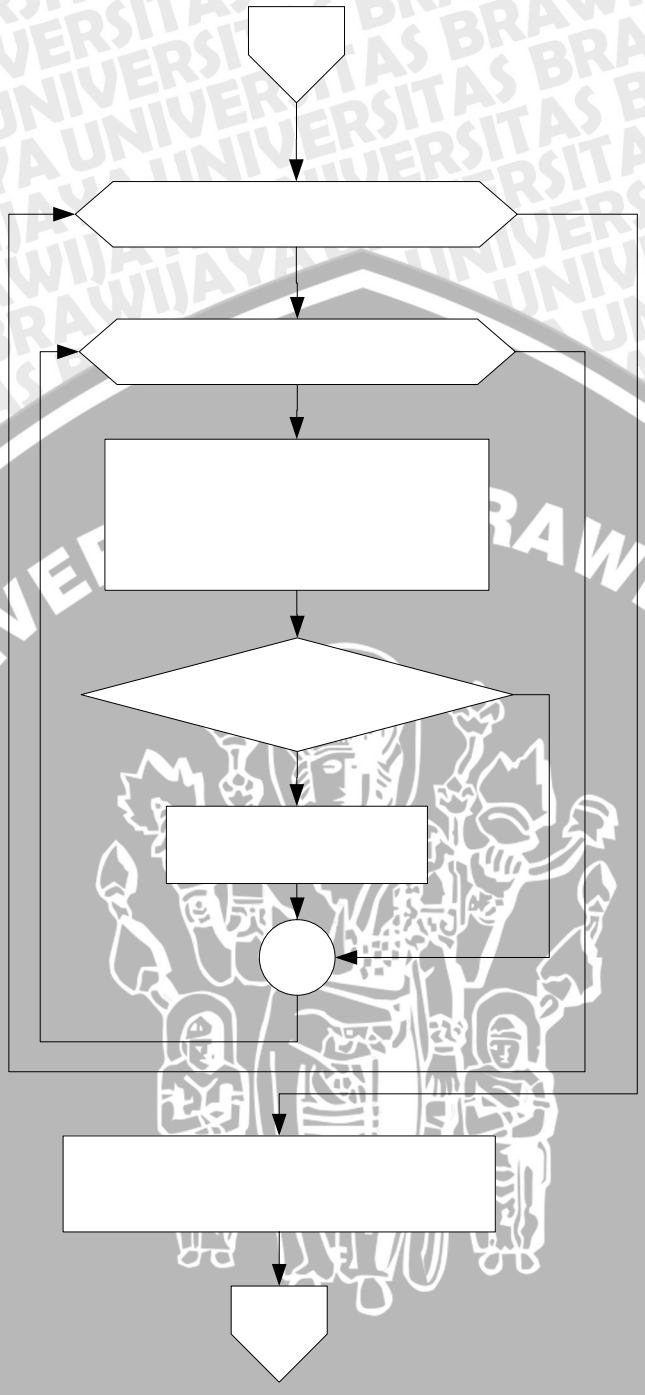
   ($n_a$,1),($n_a$,2),($n_a$,3) sd ($n_a$,nb) $n_a$ = lebar gambar dalam pixel

   Berapakah nilai byte untuk R dari warna dalam (RGB)

   Berapakah nilai byte untuk G dari warna dalam (RGB)

   Berapakah nilai byte untuk B dari warna dalam (RGB)

2. Jika `(R>195 dan G<5 dan B<100)` benar maka akan `edit1.text` akan diisi dengan koordinat pixel (i), berdasarkan pengecekan koordinat pixel(di cek dari koordinat paling kecil ke kordinat paling besar) sehingga bila kondisi `(R>195 dan G<5 dan B<100)` terjadi beberapa kali dengan koordinat pixel yang berbeda-beda maka koordinat yang akn ditulis pada `edit1.text` adalah koordinat pixel (i) paling besar.



**Gambar 5.6** Diagram Alir Prosedur scan auto on click (deteksi titik laser sebelah kiri).

**Sumber:** [Implementasi].

```
29          //deteksi titik laser sebeleh kiri
30          FOR i :=  gambar.Width DOWNTO 1 DO
31          FOR j := gambar.Height DOWNTO 1 DO
32          BEGIN
33              RGB :=  image1.Canvas.Pixels[i, j];
34              R := GetRValue(RGB);
35              G := GetGValue(RGB);
36              B := GetBValue(RGB);
37
38              IF (R>195)AND(G<5)AND(B<100)THEN
39              //IF(R=255)AND(G=226)AND(B=226)THEN
40              BEGIN
41                  edit13.text:= intTOstr(j);
42                  edit2.text:= intTOstr(i);
43              END;
44          END;
45
46          //jarak antara dua titik laser
47          edit3.Text:=IntToStr(strTOint( edit1.text)- strTOint( edit2.text));
48
```

**Gambar 5.7** Algoritma Prosedur scan auto on click (deteksi titik laser sebelah kiri).

**Sumber:** [Implementasi].

Penjelasan Algoritma Prosedur scan auto on click (deteksi titik laser sebelah kiri) pada baris 30 sampai dengan 44 pada gambar 5.7 sebagai berikut:

1. Akan dilakukan pengecekan warna mulai dari koordinai pixel

   $(n_a,n_b)$ , $(n_a,n_b\text{-}1)$ , $(n_a,n_b\text{-}2)$ $\longrightarrow$ $(n_a,1)$

   $(n_a\text{-}1,n_b)$ , $(n_a\text{-}1,n_b\text{-}1)$ , $(n_a\text{-}1,n_b\text{-}2)$ $\longrightarrow$ $(n_a\text{-}1,1)$

   $\downarrow$ $\downarrow$

   $(1,n_b)$ , $(1,n_b\text{-}1)$ , $(1,n_b\text{-}2)$ $(1,1)$

   $n_b$ = tinggi gambar dalam pixel

   $n_a$ = lebar gambar dalam pixel

   Berapakah nilai byte untuk R dari warna dalam (RGB)

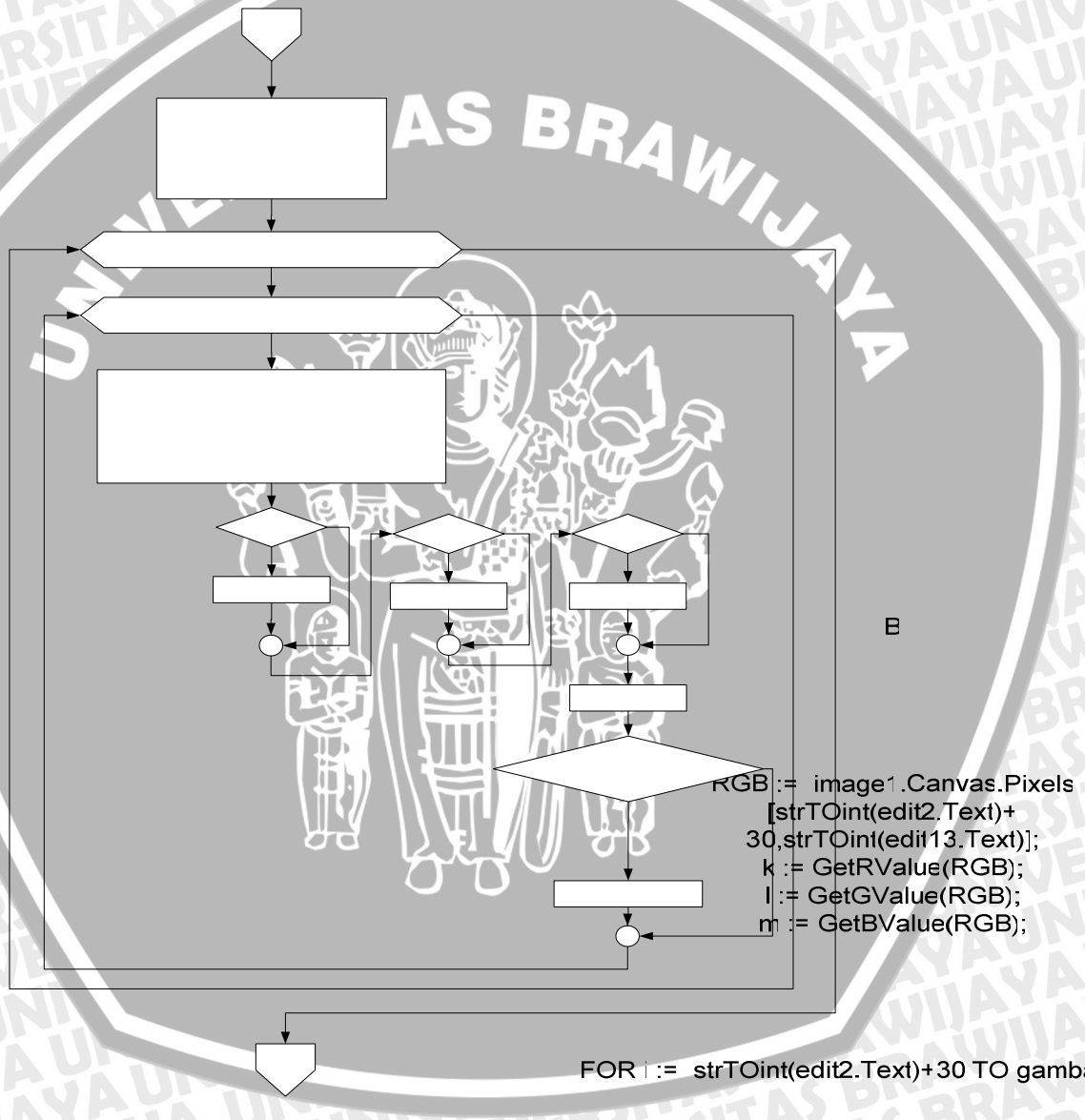
   Berapakah nilai byte untuk G dari warna dalam (RGB)

   Berapakah nilai byte untuk B dari warna dalam (RGB)

2. Jika `(R>195 dan G<5 dan B<100)` benar maka akan `edit2.text` akan diisi dengan koordinat pixel (i) dan `edit13.text` diisi dengan koordinat pixel (j), berdasarkan pengecekan koordinat pixel(di cek dari koordinat paling bear ke kordinat paling kecil) sehingga bila kondisi `(R>195 dan G<5 dan B<100)` terjadi beberapa kali dengan koordinat pixel yang berbeda-beda maka koordinat yang akan ditulis pada `edit2.text` adalah koordinat pixel (i) paling kecil, sedang koordinat pixel (j) yang diisikan

pada `edit13.text` akan di simpan sebagai acuan untuk mencari tepi dari benda yang akan diukur.

3. Pada baris ke 47 `edit3.text` ($X_1$) akan diisi jarak antara dua titik-titik laser dalam gambar (pixel) yaitu dengan cara mengurangkan koordinat dimana titik laser paling kanan ditemukan dengan titik laser paling kiri ditemukan (`edit3.text = edit1.text - edit2.text`) atau ($X_1$= `edit1.text - edit2.text`).




**Gambar 5.8** Diagram Alir Prosedur scan auto on click (deteksi tepi gambar sebelah kanan).

**Sumber:** [Implementasi].

```
49          //mencari tepi gambar sebelah kanan
50          RGB :=  image1.Canvas.Pixels[strTOint(edit2.Text)+ 30,strTOint(edit13.Text)];
51          k := GetRValue(RGB);
52          l := GetGValue(RGB);
53          m := GetBValue(RGB);
54          FOR i :=  strTOint(edit2.Text)+ 30 TO gambar.Width-2  DO
55          FOR j :=  1 TO gambar.Height DO
56          BEGIN
57              RGB :=  image1.Canvas.Pixels[i,j];
58              R := GetRValue(RGB);
59              G := GetGValue(RGB);
60              B := GetBValue(RGB);
61              d:=k-r;
62              e:=l-g;
63              f:=m-b;
64              IF d<0 THEN
65              BEGIN
66                  d:=d*(-1);
67              END;
68              IF e<0 THEN
69              BEGIN
70                  e:=e*(-1);
71              END;
72              IF f<0 THEN
73              BEGIN
74                  f:=f*(-1);
75              END;
76              u:=10;
77              IF ((d<u)AND(f<u)AND(e<u))THEN
78              BEGIN
79                  edit4.text:= intTOstr(i);
80              END;
81          END;
```

**Gambar 5.9** Algoritma Prosedur scan auto on click (deteksi tepi gambar sebelah kanan).

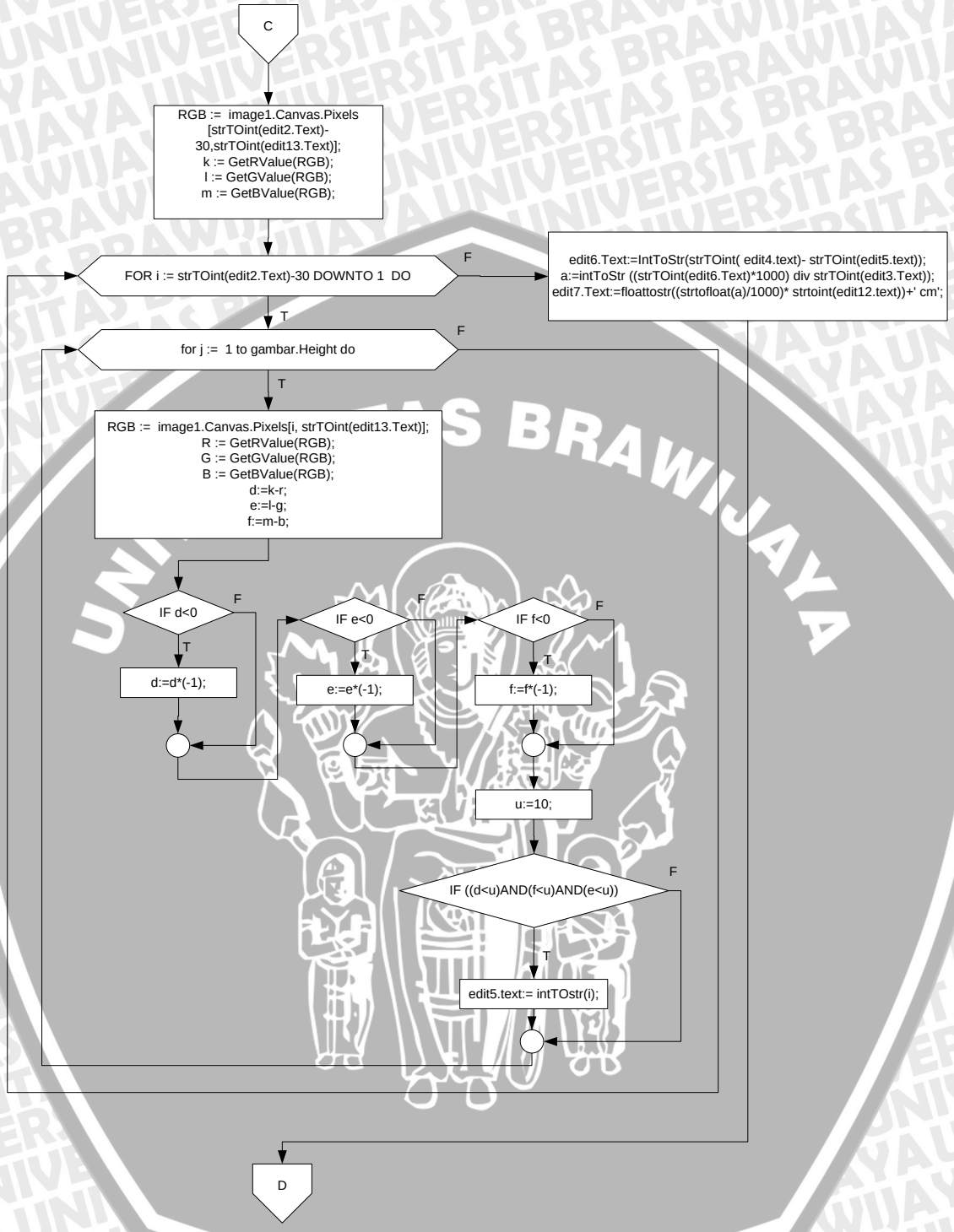
**Sumber:** [Implementasi].

Penjelasan Algoritma Prosedur scan auto on click click (deteksi tepi gambar sebelah kanan) pada Gambar 5.9

1. Pada baris 50 sampai dengan baris 53 akan dilakukan pengecekan warna pada benda atau obyek yang akan diukur yakni dengan cara menggeser 30 pixel kekanan dari titik laser yang paling kiri atau `edit2.text+30`.
2. Setelah ditemukan maka pada baris ke 54 sampai dengan baris 60 akan dilakukan pengecekan warna dengan menggunakan perulangan seperti pada perulangan-perulangan sebelumnya yakni menggeser titik koordinat kebawah dan kekanan, tetapi disini tidak dimulai dari koordinat (1,1) melainkan dari koordinat `(edit2.text+30,1)`. `Edit2.text` adalah koordinat (i) dari titik laser kiri ditemukan sehingga `edit2.text+30`

adalah koordinat (i+30)dari titik laser kiri.

3. pada baris 60 sampai dengan baris 63 akan dilakukan pengurangan kode warna benda atau obyek yang akan diukur dengan kode warna yang di temukan pada tiap-tiap koordinat dari perulangan untuk mencari selisih warna dari warna benda atau objek dengan warna dari tiap-tiap kode warna dari koordinat hasil perulangan.
4. selanjutnya pada baris 64 sampai dengan 75 akan dilakukan pengecekan dari tiap tiap hasil pengurangan apakah hasil dari tiap-tiap pengulangan tersebut < 0, jika hasil dari pengurangan tersebut < 0 maka akan dilakukan dengan (-1) hal ini dilakukan agar hasil-hasil pengurangan tidak negatif.
5. pada baris 76 sampai dengan baris 81 akakn dilakukan pengecekan pada tiap-tiap kode warna yang telah dijadikan positif apakah `((d<u)AND(f<u)AND(e<u))` dengan nilai `u=10`, yang artinya apakah selisih kode warna dari warna benda atau objek dengan kode warna dari tiap-tiap koordinat hasil perulangan kurang dari 10 jika benar maka `edit4.text` akan diisi dengan koordinat pixel (i).
6. kondisi untuk `((d<u)AND(f<u)AND(e<u))` akan terjadi berkali-kali dan `edit4.text` juga akan diisi dengan koordinat pixel (i) berkali-kali juga, sehingga kondisi `((d<u)AND(f<u)AND(e<u))` tidak benar atau batas dari benda paling kanan dengan beckground ditemukan.




**Gambar 5.10** Diagram Alir Prosedur scan auto on click (deteksi tepi gambar sebelah kiri).

**Sumber:** [Implementasi].

```
83              //mencari tepi gambar sebelah kiri
84              RGB :=  imagel.Canvas.Pixels[strTOint(edit2.Text)-30, strTOint(editl3.Text)];
85              k := GetRValue(RGB);
86              l := GetGValue(RGB);
87              m := GetBValue(RGB);
88              FOR i := strTOint(edit2.Text)-30 DOWNTO 1  DO
89              FOR j :=  1 TO gambar.Height DO
90              BEGIN
91                  RGB :=  imagel.Canvas.Pixels[i, strTOint(editl3.Text)];
92                  R := GetRValue(RGB);
93                  G := GetGValue(RGB);
94                  B := GetBValue(RGB);
95                  d:=k-r;
96                  e:=l-g;
97                  f:=m-b;
98                  IF d<0 THEN
99                  BEGIN
100                     d:=d*(-1);
101                 END;
102                 IF e<0 THEN
103                 BEGIN
104                 e:=e*(-1);
105                 END;
106                 IF f<0 THEN
107                 BEGIN
108                     f:=f*(-1);
109                 END;
110                 u:=5;
111                 IF ((d<u)AND(e<u))OR((d<u)AND(f<u))OR((e<u)AND(f<u))THEN
112                 BEGIN
113                     edit5.text:= intTOstr(i);
114                 END;
115             END;
116             //panjang benda dalam gambar
117             edit6.Text:=IntToStr(strTOint( edit4.text)- strTOint(edit5.text));
118             //Panjang Benda Sebenernya
119             a:=intToStr ((strTOint(edit6.Text)*1000) div strTOint(edit3.Text));
120             edit7.Text:=floattostr((strtofloat(a)/1000)* strtoint(editl2.text))+' cm';
```

**Gambar 5.11** Algoritma Prosedur scan auto on click (deteksi tepi gambar sebelah kiri).

**Sumber:** [Implementasi].

Penjelasan Algoritma Prosedur scan auto on click click (deteksi tepi gambar sebelah kanan) pada Gambar 5.11

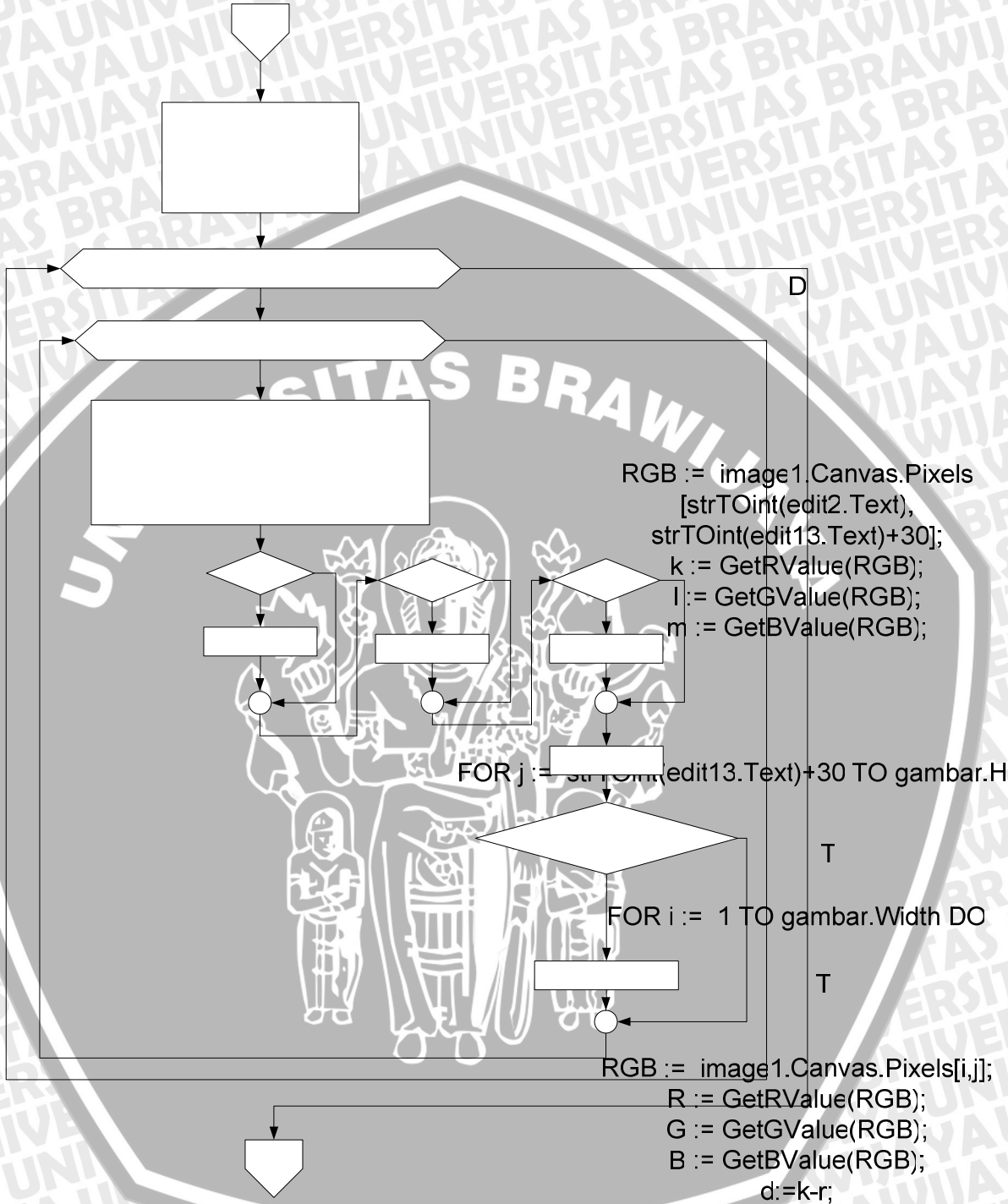
1. Pada baris 84 sampai dengan baris 87 akan dilakukan pengecekan warna pada benda atau obyek yang akan di ukur yakni dengan cara menggeser 30 pixel kekiri dari titik laser yang paling kiri atau `edit2.text-30`.
2. Setelah ditemukan maka pada baris ke 88 sampai dengan baris 94 akan dilakukan pengecekan warna dengan menggunakan perulangan seperti pada perulangan-perulangan sebelumnya yakni menggeser titik koordinat kebawah dan kekiri, tetapi disini tidak dimulai dari koordinat (1,1) melainkan dari koordinat `(edit2.text-30,1)`. `Edit2.text` adalah koordinat (i) dari titik laser kiri ditemukan sehingga `edit2.text-30` adalah koordinat (i-30)dari titik laser kiri.

3. pada baris 95 sampai dengan baris 97 akan dilakukan pengurangan kode warna benda atau obyek yang akan diukur dengan kode warna yang di temukan pada tiap-tiap koordinat dari perulangan untuk mencari selisih warna dari warna benda atau objek dengan warna dari tiap-tiap kode warna dari koordinat hasil perulangan.
4. selanjutnya pada baris 98 sampai dengan 109 akan dilakukan pengecekan dari tiap tiap hasil pengurangan apakah hasil dari tiap-tiap pengulangan tersebut < 0 jika hasil dari pengurangan tersebut < 0 maka akan dilakukan dengan (-1) hal ini dilakukan agar hasil-hasil pengurangan tidak negatif.
5. pada baris 110 sampai dengan baris 115 akakn dilakukan pengecekan pada tiap-tiap kode warna yang telah dijadikan positif apakah `((d<u)AND(f<u)AND(e<u))` dengan nilai `u=10`, yang artinya apakah selisih kode warna dari warna benda atau objek dengan kode warna dari tiap-tiap koordinat hasil perulangan kurang dari 10 jika benar maka `edit5.text` akan diisi dengan koordinat pixel (i).
6. kondisi untuk `((d<u)AND(f<u)AND(e<u))` akan terjadi berkali-kali dan `edit5.text` juga akan diisi dengan koordinat pixel (i) berkali-kali juga, sehingga kondisi `((d<u)AND(f<u)AND(e<u))` tidak benar atau batas dari benda paling kiri dengan beckround ditemukan.
7. pada baris 117 dilakukan perhitungan panjang benda dalam gambar ($X_2$)

```
edit6.Text:=IntToStr(strTOint(edit4.text)-
strTOint(edit5.text));
```

   $X_2$ = yaitu koordinat (i) dari (batas tepi kanan benda – batas tepi kiri benda).
8. pada baris 118 dan 119 dilakukan perhitungan panjang sebenarnya dari benda yang diukur($Y_1 = \frac{X_2}{X_1}$),($Z_1 = Y * A$), agar hasil dari $Z_1$ mempunyai dua angka di belakang koma maka ($Y_1 = \frac{X_2 * 1000}{X_1}$)dalam integer, kemudian($Z_1 = \frac{Y}{1000} * A$) dalam float. perkalian dan pembagian 1000 dikarenakan A merupakan bilangan puluhan sehingga akan menghasilkan 2 angka dibelakang koma (jika angka dibelakang koma tidak sama dengan

nol) karena angka nol dibelakang koma akan di abaikan.




**Gambar 5.12** Diagram Alir Prosedur scan auto on click (deteksi tepi gambar sebelah bawah).

**Sumber:** [Implementasi].

```
239             //mencari tepi gambar sebelah bawah
240             RGB :=  image1.Canvas.Pixels[strTOint(edit2.Text),strTOint(edit13.Text)+30];
241             k := GetRValue(RGB);
242             l := GetGValue(RGB);
243             m := GetBValue(RGB);
244             FOR j :=  strTOint(edit13.Text)+30 TO gambar.Height-2  DO
245             FOR i :=  1 TO gambar.Width DO
246             BEGIN
247                 RGB :=  image1.Canvas.Pixels[i,j];
248                 R := GetRValue(RGB);
249                 G := GetGValue(RGB);
250                 B := GetBValue(RGB);
251                 d:=k-r;
252                 e:=l-g;
253                 f:=m-b;
254                 IF d<0 THEN
255                 BEGIN
256                     d:=d*(-1);
257                 END;
258                 IF e<0 THEN
259                 BEGIN
260                     e:=e*(-1);
261                 END;
262                 IF f<0 THEN
263                 BEGIN
264                     f:=f*(-1);
265                 END;
266                 u:=10;
267                 IF ((d<u)AND(f<u)AND(e<u))THEN
268                 BEGIN
269                     edit8.text:= intTOstr(j);
270                 END;
271             END;
```

**Gambar 5.13** Algoritma Prosedur scan auto on click (deteksi tepi gambar sebelah bawah).

**Sumber:** [Implementasi].

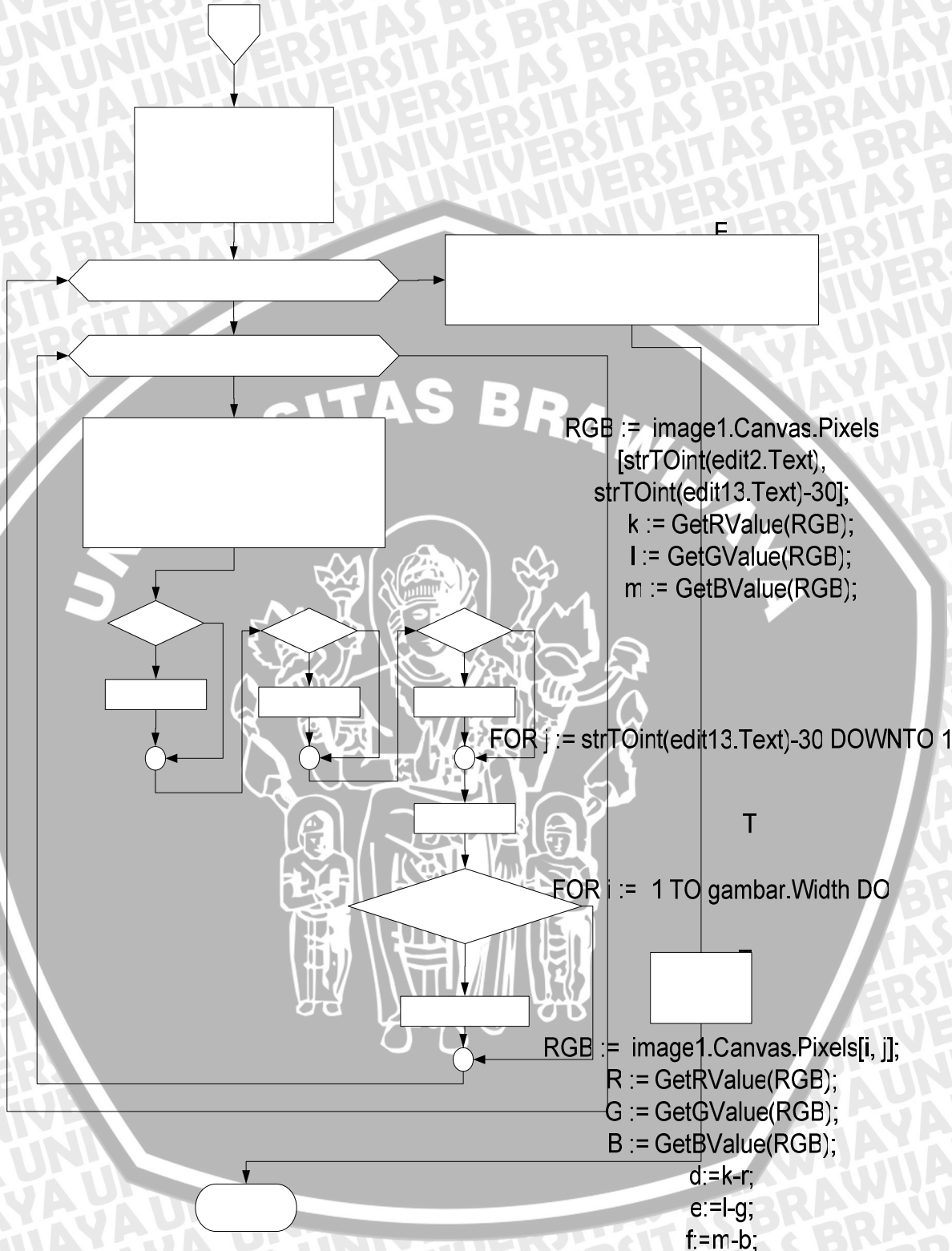
Penjelasan Algoritma Prosedur scan auto on click click (deteksi tepi gambar sebelah bawah) pada Gambar 5.13

1. Pada barir 240 sampai dengan 243 akan dilakukan pengecekan warna pada benda atau obyek yang akan di ukur yakni dengan cara menggeser 30 pixel kebawah dari titik laser yang paling kiri atau `edit13.text+30`.
2. Setelah ditemukan maka pada baris ke 244 sampai dengan baris 250 akan dilakukan pengecekan warna dengan menggunakan perulangan seperti pada perulangan-perulangan sebelumnya yakni menggeser titik koordinat kekanan dan kebawah, tetapi disini tidak dimulai dari koordinat (1,1) melainkan dari koordinat `(1,edit13.text+30)`. `Edit13.text` adalah koordinat (j) dari titik laser kiri ditemukan sehingga `edit13.text+30` adalah koordinat (j+30) dari titik laser kiri.
3. pada baris 251 sampai dengan baris 153 akan dilakukan pengurangan kode warna benda atau obyek yang akan diukur dengan kode warna yang di

temukan pada tiap-tiap koordinat dari perulangan untuk mencari selisih warna dari warna benda atau objek dengan warna dari tiap-tiap kode warna dari koordinat hasil perulangan.

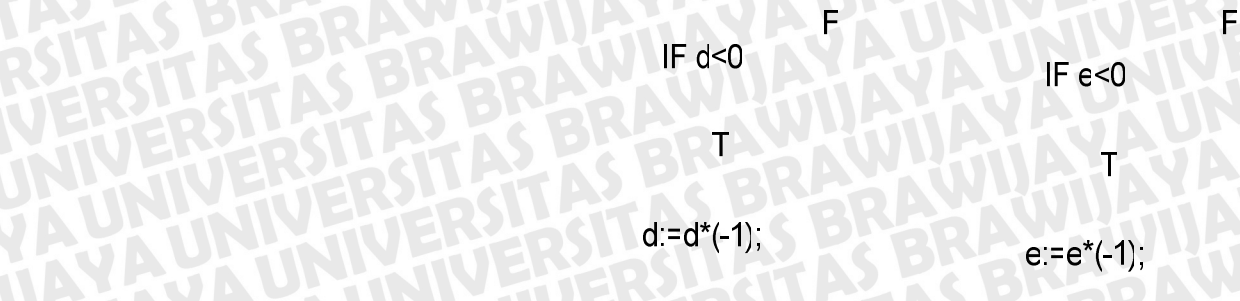
4. selanjutnya pada baris 254 sampai dengan 265 akan dilakukan pengecekan dari tiap tiap hasil pengurangan apakah hasil dari tiap-tiap pengulangan tersebut < 0 jika hasil dari pengurangan tersebut < 0 maka akan dilakukan dengan (-1) hal ini dilakukan agar hasil-hasil pengurangan tidak negatif.

5. pada baris 266 sampai dengan baris 171 Microsoft Office Word 2003.lnk akan dilakukan pengecekan pada tiap-tiap kode warna yang telah dijadikan positif apakah ((d<u)AND(f<u)AND(e<u)) dengan nilai u=10, yang artinya apakah selisih kode warna dari warna benda atau objek dengan kode warna dari tiap-tiap koordinat hasil perulangan kurang dari 10 jika benar maka `edit8.text` akan diisi dengan koordinat pixel (j).
6. kondisi untuk `((d<u)AND(f<u)AND(e<u))` akan terjadi berkali-kali dan `edit8.text` juga akan diisi dengan koordinat pixel (j) berkali-kali juga, sehingga kondisi `((d<u)AND(f<u)AND(e<u))` tidak benar atau batas dari benda paling kanan dengan beckground ditemukan.




**Gambar 5.14** Diagram Alir Prosedur scan auto on click (deteksi tepi gambar sebelah atas).

**Sumber:** [Implementasi].



```
273             //mencari tepi gambar sebelah atas
274             RGB :=  image1.Canvas.Pixels[strTOint(edit2.Text),strTOint(edit13.Text)-30];
275             k := GetRValue(RGB);
276             l := GetGValue(RGB);
277             m := GetBValue(RGB);
278             FOR j := strTOint(edit13.Text)-30 DOWNTO 1  DO
279             FOR i :=  1 TO gambar.Width DO
280             BEGIN
281                 RGB :=  image1.Canvas.Pixels[strTOint(edit2.Text), j];
282                 R := GetRValue(RGB);
283                 G := GetGValue(RGB);
284                 B := GetBValue(RGB);
285                 d:=k-r;
286                 e:=l-g;
287                 f:=m-b;
288                 IF d<0 THEN
289                 BEGIN
290                     d:=d*(-1);
291                 END;
292                 IF e<0 THEN
293                 BEGIN
294                     e:=e*(-1);
295                 END;
296                 IF f<0 THEN
297                 BEGIN
298                     f:=f*(-1);
299                 END;
300                 u:=10;
301                 IF ((d<u)AND(f<u)AND(e<u))THEN
302                 BEGIN
303                     edit9.text:= intTOstr(j);
304                 END;
305             END;
306             //panjang benda dalam gambar
307             edit10.Text:=IntToStr(strTOint( edit8.text)- strTOint( edit9.text));
308             //Tinggi Benda sebenarnya
309             edit11.Text:=floatToStr(strTOfloat(edit10.Text)/ strTOfloat(edit3.Text)* strTOfloat(edit12.text))+' cm';
310             FINALLY
311             gambar.Free
312         END
313     END;
```

**Gambar 5.15** Algoritma Prosedur scan auto on click (deteksi tepi gambar sebelah atas).

**Sumber:** [Implementasi].

Penjelasan Algoritma Prosedur scan auto on click click (deteksi tepi gambar sebelah kanan) pada Gambar 5.15

1. Pada barir 274 sampai dengan baris 277 akan dilakukan pengecekan warna pada benda atau obyek yang akan di ukur yakni dengan cara menggeser 30 pixel keatas dari titik laser yang paling atas atau `edit13.text-30`.
2. Setelah ditemukan maka pada baris ke 278 sampai dengan baris 284 akan dilakukan pengecekan warna dengan menggunakan perulangan seperti pada perulangan-perulangan sebelumnya yakni menggeser titik koordinat ke kiri dan ke atas, tetapi disini tidak dimulai dari koordinat (1,1) melainkan dari koordinat `(1,edit2.text-30)`. `Edit13.text` adalah koordinat (j) dari titik laser kiri ditemukan sehingga `edit13.text-30` adalah koordinat (j-30) dari titik laser kiri.
3. Pada baris 285 sampai dengan baris 287 akan dilakukan pengurangan kode

warna benda atau obyek yang akan diukur dengan kode warna yang di temukan pada tiap-tiap koordinat dari perulangan untuk mencari selisih warna dari warna benda atau objek dengan warna dari tiap-tiap kode warna dari koordinat hasil perulangan.

4. Selanjutnya pada baris 288 sampai dengan 299 akan dilakukan pengecekan dari tiap tiap hasil pengurangan apakah hasil dari tiap-tiap pengulangan tersebut < 0 jika hasil dari pengurangan tersebut < 0 maka akan dilakukan dengan (-1) hal ini dilakukan agar hasil-hasil pengurangan tidak negatif.
5. Pada baris 300 sampai dengan baris 305 akakn dilakukan pengecekan pada tiap-tiap kode warna yang telah dijadikan positif apakah `((d<u)AND(f<u)AND(e<u))` dengan nilai `u=10`, yang artinya apakah selisih kode warna dari warna benda atau objek dengan kode warna dari tiap-tiap koordinat hasil perulangan kurang dari 10 jika benar maka `edit9.text` akan diisi dengan koordinat pixel (j).
6. Kondisi untuk `((d<u)AND(f<u)AND(e<u))` akan terjadi berkali-kali dan `edit9.text` juga akan diisi dengan koordinat pixel (j) berkali-kali juga, sehingga kondisi `((d<u)AND(f<u)AND(e<u))` tidak benar atau batas dari benda paling atas dengan beckground ditemukan.
7. Pada baris 307 dilakukan perhitungan panjang benda dalam gambar ($X_3$)

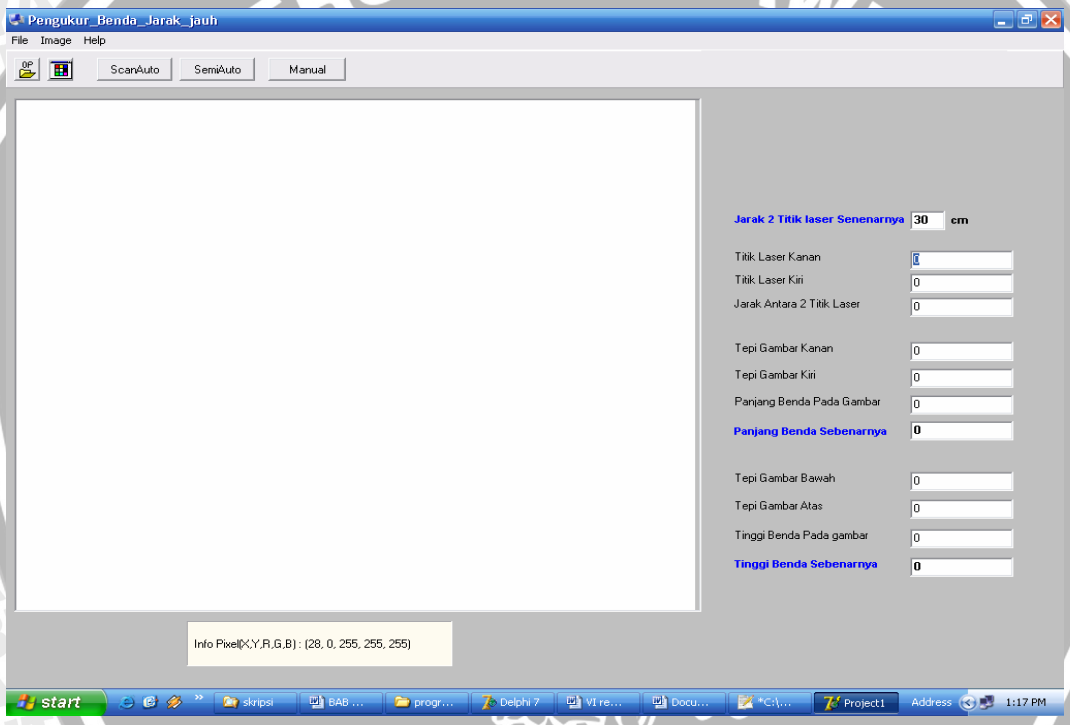
   ```
   edit10.Text:=IntToStr(strTOint(edit8.text)-
   strTOint(edit9.text));
   ```

   $X_3$ = yaitu koordinat (j) dari (batas tepi bawah benda – batas tepi atas benda).
8. Pada baris 309 dan 313 dilakukan perhitungan panjang sebenarnya dari benda yang diukur($Y_2 = \frac{X_3}{X_1}$),($Z_2 = Y * A$), agar hasil dari $Z_2$ mempunyai dua angka di belakang koma maka ($Y_2 = \frac{X_3 * 1000}{X_1}$)dalam integer, kemudian($Z_2 = \frac{Y}{1000} * A$) dalam float. perkalian adan pembagian 1000 dikarenakan A merupakan bilangan puluhan sehingga akan menghasilkan 2 angka dibelakang koma (jika angka dibelakang koma tidak sama dengan nol) karena angka nol dibelakang koma akan di abaikan.

## 5.3. Implementasi Antarmuka Aplikasi Pengukur Panjang dengan Metode Pengolahan Citra

Aplikasi pengukur panjang dengan metode pengolahan citra mempunyai menu menu sebagai berikut

1. Menu untuk mengambil gambar (Load Image) sebagai input dari program, yang ada pada File ⟶ LoadImage seperti pada gambar 5.16.
2. Gambar yang di ambil adalah gambar yang akan diukur artinya gambar tersebut diambil dengan menggunakan kamera dan laser sebagai acuan untuk pengukuran.

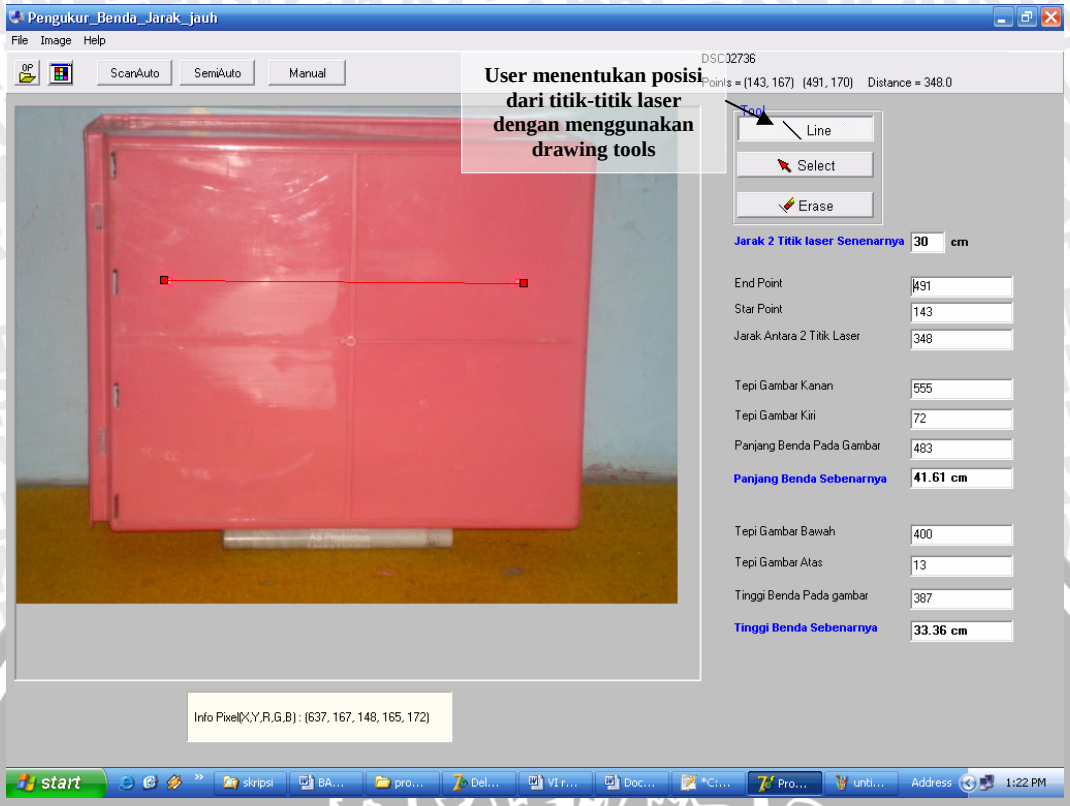



**Gambar 5.16** Menu LoadImage

3. Menu ScanAuto untuk melakukan perhitungan panjang dan tinggi benda secara auto, yang berada pada Image ⟶ ScanAuto seperti pada gambar 5.17.

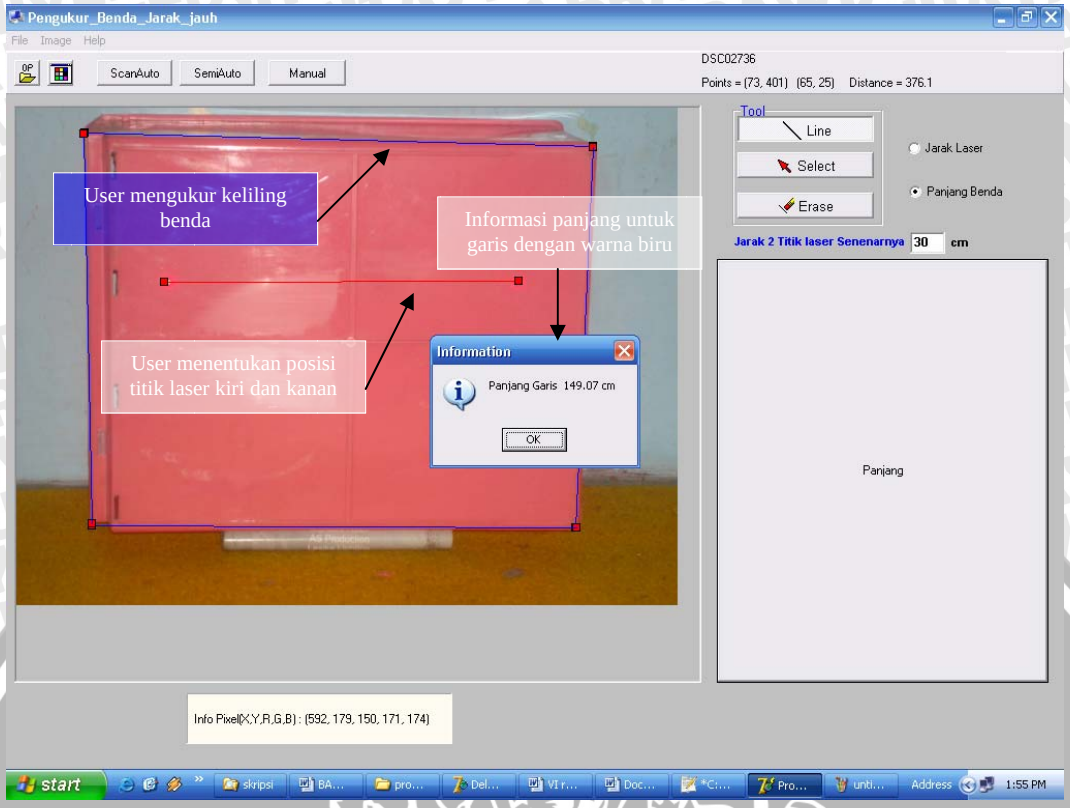

**Gambar 5.17** Menu ScanAuto.

4. Jika proses ScanAuto tidak berhasil dilakukan (warna untuk titik-titik laser tidak terdeteksi) program menyediakan menu `SemiAuto` Dan `Manual` yang ada pada `RadioGroup` apabila user memilih `SemiAuto` maka User akan menentukan dimana posisi titik laser kiri dan kanan dengan menggunakan menu Drawing tools pada form yang telah disediakan, setelah titik titik tersebut ditntukan maka program akan melakukan deteksi tepi dan kemudian menghitung berapa panjang dan tinggi benda tersebut seperti pada gambar 5.18.




**Gambar 5.18** Menu SemiAuto dipilih.

5. Jika proses SemiAuto tidak berhasil (perhitungan kurang akurat), biasanya terjadi karena adanya gangguan warna yang sama dengan warna benda atau user ingin mengukur benda yang rumit seperti panjang ular, keliling benda dll, user bisa memilih menu Manual pada yang ada pada program, yang berarti user akan menentukan dimana titik-titik laser dan dan selanjutnya user menentukan mulai dari mana panjang benda yang akan diukur seperti pada gambar 5.19.




**Gambar 5.19** Menu Manual dipilih.

# BAB VI
# PENGUJIAN DAN ANALISIS

Bab ini membahas proses pengujian dan analisis terhadap aplikasi pengukur panjang dengan metode pengolahan citra yang telah dibangun. Pengujian yang dilakukan meliputi pengujian *White-Box Testing* dan B*lack-Box Testing* dan analisis statistik. Proses analisis dilakukan untuk mengetahui kinerja sistem apakah telah memenuhi kebutuhan yang ada. Analisis yang dibuat mencakup semua hasil pengujian yang telah dilakukan.

## 6.1 *White-Box Testing*

Pengujian *white-box* berfokus pada struktur kontrol program. *Test case* dilakukan untuk memastikan bahwa semua statemen pada program telah dieksekusi paling tidak satu kali selama pengujian dan bahwa semua kondisi logis telah diuji. Pengujian *basic path*, tehnik pengujian *white-box*, menggunakan grafik (matriks grafiks) untuk melakukan serangkaian pengujian yang independent secara linear yang akan memastikan cakupan [ARS-09]. Kesalahan logika dan asumsi yang tidak benar kebanyakan dilakukan ketika coding untuk "kasus tertentu". Dibutuhkan kepastian bahwa eksekusi jalur ini telah dites.Asumsi bahwa adanya kemungkinan terhadap eksekusi jalur yang tidak benar. Dengan white box testing dapat ditemukan kesalahan ini. Kesalahan penulisan yang acak. Seperti berada pada jalur logika yang membingungkan pada jalur normal[PSB-09].

### 6.1.1 Prosedur TProgram_utama.ScanAutoClick

Pengujian yang dilakukan pada Prosedur TProgram_utama.ScanAutoClick adalah untuk memeriksa apakah semua jalur independen yang ada pada algoritma Prosedur TProgram_utama.ScanAutoClick terlewati semua. Pada pengujian prosedur ini dibagai menjadi beberapa *flow graph* yaitu deteksi tepi kanan,kiri,bawah dan atas benda.

```
procedure TProgram_utama.ScanAutoClick(Sender: TObject);
  VAR
  i,j,u,d,e,f:  INTEGER;
  a:string;
   r,b,g,k,l,m :byte;
  BEGIN
    gambar := TBitmap.Create;
    TRY
      gambar.PixelFormat := pf24bit;
      gambar.Width  := Image1.Width;
      gambar.Height := Image1.Height;

RadioButton1.Visible:=false;
RadioButton2.Visible:=false;
RadioButton3.Visible:=false;
RadioButton4.Visible:=false;
RadioButton5.Visible:=false;
RadioButton6.Visible:=false;
RadioButton1.Checked:=false;
RadioButton2.Checked:=false;
RadioButton3.Checked:=false;
RadioButton4.Checked:=false;
RadioButton5.Checked:=false;
RadioButton6.Checked:=false;
edit1.Text:='0';
edit2.Text:='0';
edit3.Text:='0';
edit4.Text:='0';
edit5.Text:='0';
edit6.Text:='0';
edit7.Text:='0';
edit8.Text:='0';
edit9.Text:='0';
edit10.Text:='0';
edit11.Text:='0';
edit13.Text:='0';
```

1

```
//deteksi titik laser sebelah kanan
      FOR i :=  1 TO gambar.Width  DO
      FOR j := 1 TO gambar.Height DO
      BEGIN
        RGB :=  image1.Canvas.Pixels[i, j];
        R := GetRValue(RGB);
        G := GetGValue(RGB);
        B := GetBValue(RGB);

        IF (R=255)AND(G<250)AND(B<250)THEN
        BEGIN
          edit1.text:= intTOstr(i);
        END;
       END;

      //deteksi titik laser sebeleh kiri
      FOR i :=  gambar.Width DOWNTO 1 DO
      FOR j := gambar.Height DOWNTO 1 DO
      BEGIN
        RGB :=  image1.Canvas.Pixels[i, j];
        R := GetRValue(RGB);
        G := GetGValue(RGB);
        B := GetBValue(RGB);

      IF (R=255)AND(G<250)AND(B<250)THEN
         BEGIN
          edit13.text:= intTOstr(j);
          edit2.text:= intTOstr(i);
         END;
        END;

      //jarak antara dua titik laser
      edit3.Text:=IntToStr(strTOint( edit1.text)- strTOint(
edit2.text));
```

2 3 4 5 6 7 8 9 10

```
    if (strTOint(edit1.text)>0) AND
(strtoint(edit2.text)>0)AND(strtoint(edit3.text)>0)THEN
    BEGIN
    //mencari tepi gambar sebelah kanan
    RGB :=  image1.Canvas.Pixels[strTOint(edit2.Text)+
30,strTOint(edit13.Text)];
    k := GetRValue(RGB);
    l := GetGValue(RGB);
    m := GetBValue(RGB);
  FOR i :=  strTOint(edit2.Text)+ 30 TO gambar.Width-2  DO
    for j :=  1 to gambar.Height do
    BEGIN
      RGB :=  image1.Canvas.Pixels[i,j];
      R := GetRValue(RGB);
      G := GetGValue(RGB);
      B := GetBValue(RGB);

      d:=k-r;
      e:=l-g;
      f:=m-b;

      IF d<0          THEN
      BEGIN
        d:=d*(-          1);

      END;

      IF e<0 THEN

      BEGIN
                       e:=e*(-1);

      END;
```

11

A1

A2

A3

A4

A5

A6

A7

A8

A9

A10

```
IF f<0 THEN

    BEGIN
      f:=f*(-          1);

    END;

    u:=10;
    IF ((d<u)AND(f<u)AND(e<u))THEN

    BEGIN
      edit4.text:= intTOstr(i);

    END
  END;

    //mencari tepi gambar sebelah kiri
    RGB :=  image1.Canvas.Pixels[strTOint(edit2.Text)-30,
strTOint(edit13.Text)];
    k := GetRValue(RGB);
    l := GetGValue(RGB);
    m := GetBValue(RGB);

    FOR i := strTOint(edit2.Text)-30 DOWNTO 1  DO
    for j :=  1 to gambar.Height do
    BEGIN
      RGB :=  image1.Canvas.Pixels[i, j];
      R := GetRValue(RGB);
      G := GetGValue(RGB);
      B := GetBValue(RGB);
      d:=k-r;
      e:=l-g;
      f:=m-b;

    IF d<0 THEN
```

A11 A12 A13 A14 A15 A16 B1 B2 B3 B4

```
BEGIN
  d:=d*(-1);
END;
IF e<0 THEN
BEGIN
  e:=e*(-1);
END;
IF f<0 THEN
BEGIN
  f:=f*(-1);
END;
u:=10;
 IF ((d<u)AND(f<u)AND(e<u))THEN
BEGIN
  edit5.text:= intTOstr(i);
END;
END;

//panjang benda dalam gambar
edit6.Text:=IntToStr(strTOint( edit4.text)-
strTOint(edit5.text));
//Panjang Benda Sebenernya
a:=intToStr ((strTOint(edit6.Text)*1000) div
strTOint(edit3.Text));
edit7.Text:=floattostr((strtofloat(a)/1000)*
strtoint(edit12.text))+' cm';
```

B5 B6 B7 B8 B9 B10 B11 B12 B13 B14 B15 B16 B17 B18

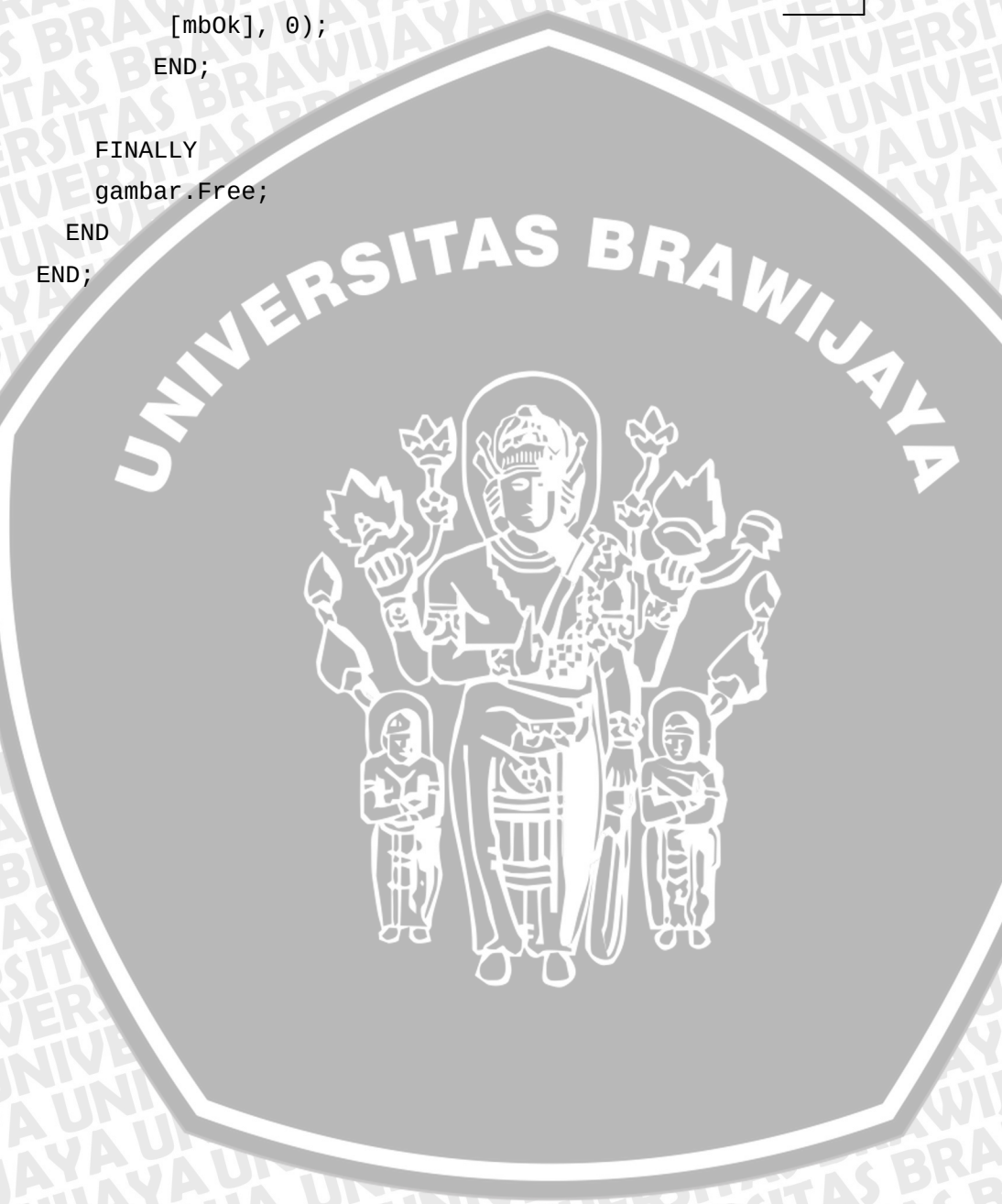
```
//mencari tepi gambar sebelah bawah
RGB :=image1.Canvas.Pixels[strTOint(edit2.Text),
strTOint(edit13.Text)+30];
k := GetRValue(RGB);
l := GetGValue(RGB);
m := GetBValue(RGB);
FOR j :=  strTOint(edit13.Text)+30 TO gambar.Height-2  DO
FOR i :=  1 TO gambar.Width DO
    BEGIN
        RGB :=  image1.Canvas.Pixels[i,j];
        R := GetRValue(RGB);
        G := GetGValue(RGB);
        B := GetBValue(RGB);
        d:=k-r;
        e:=l-g;
        f:=m-b;
        IF d<0 THEN

        BEGIN
              d:=d*(-1);

        END;

        IF e<0 THEN

        BEGIN
              e:=e*(-1);

        END;

        IF f<0 THEN

        BEGIN
              f:=f*(-1);

        END;
```

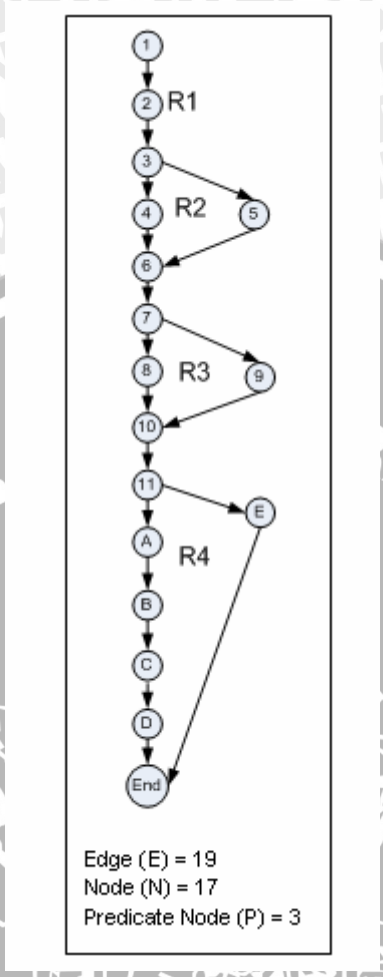
C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 C11 C12

```
        u:=10;                                          C13

        IF ((d<u)AND(f<u)AND(e<u))THEN                  C14
        BEGIN
              edit8.text:= intTOstr(j);                 C15
        END;                                            C16
END;

//mencari tepi gambar sebelah atas
RGB :=
image1.Canvas.Pixels[strTOint(edit2.Text),strTOint(edit13.Text)-30];   D1
k := GetRValue(RGB);
l := GetGValue(RGB);
m := GetBValue(RGB);



FOR j := strTOint(edit13.Text)-30 DOWNTO 1  DO
FOR i :=  1 TO gambar.Width DO
BEGIN
  RGB :=  image1.Canvas.Pixels[i, j];                   D2
  R := GetRValue(RGB);
  G := GetGValue(RGB);
  B := GetBValue(RGB);

  d:=k-r;                                               D3
  e:=l-g;
  f:=m-b;

  IF d<0 THEN                                           D4

  BEGIN
    d:=d*(-1);                                          D5

  END;                                                  D6
```

```
IF e<0 THEN                                    D7

BEGIN
  e:=e*(-1);                                   D8

END;                                           D9

IF f<0 THEN                                    D10

BEGIN
  f:=f*(-1);                                   D11

END;                                           D12

u:=10;                                         D13

  IF ((d<u)AND(f<u)AND(e<u))THEN                D14
  BEGIN
    edit9.text:= intTOstr(j);                  D15
  END;
END;                                           D16

  //Tinggi benda dalam gambar
  edit10.Text:=IntToStr(strTOint( edit8.text)- strTOint(
edit9.text));                                  D17
  //Tinggi benda sebenarnya
  a:=intToStr ((strTOint(edit10.Text)*1000) div
strTOint(edit3.Text));                         D17
  edit11.Text:=floattostr((strtofloat(a)/1000)*
strtoint(edit12.text))+' cm';
  END
```

```
            ELSE
            BEGIN
             MessageDlg('titik-titik laser tidak ditemukan.',
mtInformation,
             [mbOk], 0);
            END;

      FINALLY
      gambar.Free;
    END
  END;
```






**Gambar 6.1.** Pemodelan algoritma Prosedur ScanAutoClick ke dalam *flow graph.*

**Sumber** : [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap Prosedur ScanAutoClick menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* atau V(G) melalui beberapa persamaan diantaranya :

V(G) = Jumlah Region

V(G) = E – N + 2

V(G) = P + 1

dimana :
- E merupakan sisi (garis penghubung antar *node (Edge)*
- N merupakan jumlah simpul *(Node)*
- P merupakan simpul yang memiliki cabang (*Predicate Node)*

Berdasarkan persamaan di atas diperoleh perhitungan :

V(G) = Jumlah Region = (R1,R2,R3,R4) = 4

V(G) = E – N + 2 = 19 – 17 + 2 = 4

V(G) = P + 1 = 3 + 1 = 4

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 4, ditentukan empat buah basis set dari jalur independen yaitu:

Jalur 1 : 1-2-3-4-6-7-8-10-11-A-B-C-D

Jalur 2 : 1-2-3-5-6-7-8-10-11-E

Jalur 3 : 1-2-3-4-6-7-9-10-11-E

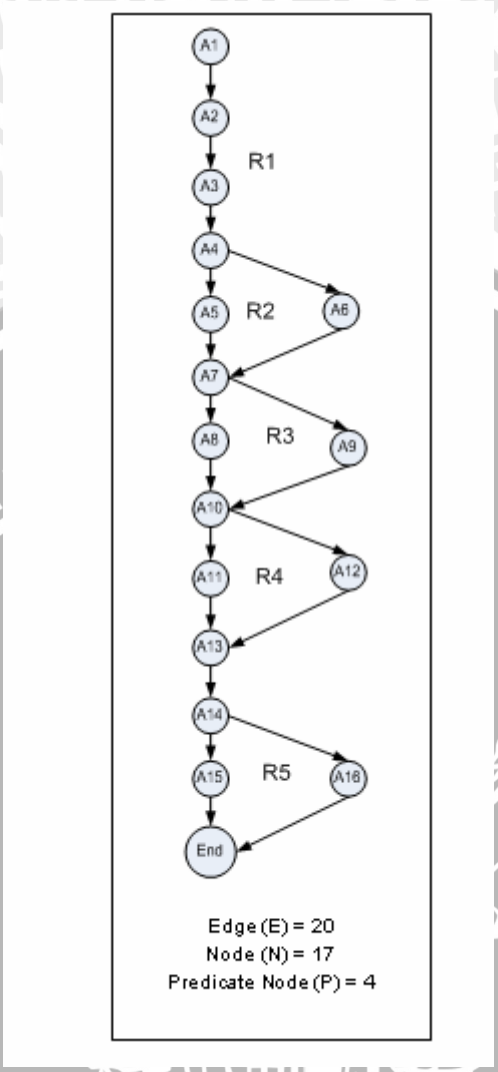
Jalur 4 : 1-2-3-4-6-7-8-10-11-E

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah dijelaskan pada Tabel 6.1.

**Tabel 6.1.** *Test case* untuk pengujian Prosedur ScanAutoClick

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Jarak antara 2 titik laser ditemukan dan tidak = ”0” | Muncul informasi (posisi titik titik pixel dari titik laser kiri, kanan, tepibenda kiri, kanan, atas, bawah,jarak antara 2 titik laser, panjang dan tinggi benda dalam gambar maupun sebenarnya) | Muncul informasi (posisi titik titik pixel dari titik laser kiri, kanan, tepibenda kiri, kanan, atas, bawah,jarak antara 2 titik laser, panjang dan tinggi benda dalam gambar maupun sebenarnya) |
| 2 | Program tidak menemukan warna titik laser kanan | Muncul pesan “titik-titik laser tidak ditemukan.” | Muncul pesan “titik-titik laser tidak ditemukan.” |
| 3 | Program tidak menemukan warna titik laser kiri | Muncul pesan “titik-titik laser tidak ditemukan.” | Muncul pesan “titik-titik laser tidak ditemukan.” |
| 4 | Jarak antara dua titik laser = ”0” | Muncul pesan “titik-titik laser tidak ditemukan.” | Muncul pesan “titik-titik laser tidak ditemukan.” |

**Sumber** : [Pengujian]




**Gambar 6.2.** Pemodelan algoritma Prosedur ScanAutoClick deteksi tepi kanan ke dalam *flow graph*.

**Sumber** : [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap prosedur ScanAutoClick deteksi tepi kanan menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* atau V(G) melalui beberapa persamaan diantaranya :

V(G) = Jumlah Region = (R1,R2,R3,R4,R5) = 5

V(G) = E – N + 2 = 20 – 17 + 2 = 5

V(G) = P + 1 = 4+ 1 = 5

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 5, ditentukan empat buah basis set dari jalur independen yaitu:

Jalur 1 : A1-A2-A3-A4-A5-A7-A8-A10-A11-A13-A14-A15-And

Jalur 2 : A1-A2-A3-A4-A6-A7-A8-A10-A11-A13-A14-A15-And

Jalur 3 : A1-A2-A3-A4-A5-A7-A9-A10-A11-A13-A14-A15-And

Jalur 4 : A1-A2-A3-A4-A5-A7-A8-A10-A12-A13-A14-A15-And

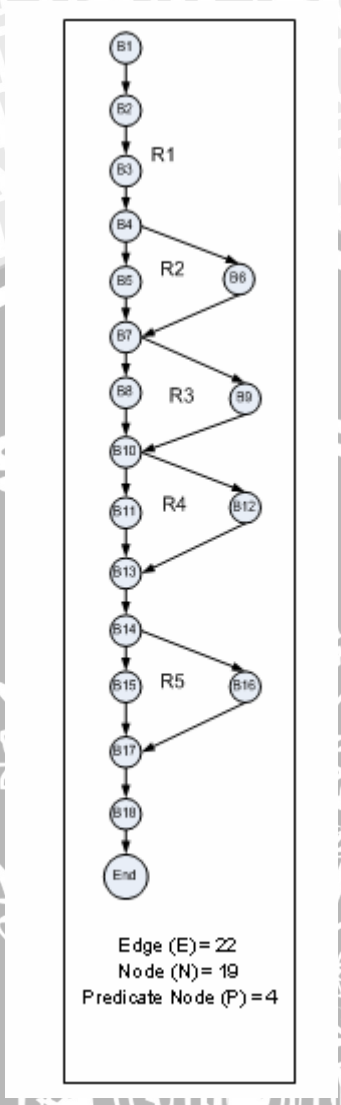
Jalur 5 : A1-A2-A3-A4-A5-A7-A8-A10-A11-A13-A14-A16-And

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah dijelaskan pada tabel 6.2.

**Tabel 6.2.** *Test case* untuk pengujian Prosedur ScanAutoClick deteksi tepi kanan

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Mencari titik pixel dari tepi kanan benda | Edit4.text= posisi titik pixel tepi kanan benda | Edit4.text= posisi titik pixel tepi kanan benda |
| 2 | Jika nilai d positif maka d tidak dikalikan dengan (-1) | d positif (d selalu positif) | d positif (d selalu positif) |
| 3 | Jika nilai d positif maka e tidak dikalikan dengan (-1) | e positif (e selalu positif) | e positif (e selalu positif) |
| 4 | Jika nilai d positif maka f tidak dikalikan dengan (-1) | f positif (d selalu positif) | f positif (d selalu positif) |
| 5 | Jika kondisi (d<u)AND(f<u)AND(e<u)) tidak terpenuhi | Posisi pixel dari tepi kanan tidak ditemukan sehingga edit4.text=”0” (nilai awal edit4.text=”0”) | edit4.text=”0” |

**Sumber** : [Pengujian]




**Gambar 63.** Pemodelan algoritma Prosedur ScanAutoClick deteksi tepi kiri ke dalam *flow graph*.

**Sumber** : [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap Prosedur ScanAutoClick deteksi tepi kiri menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* atau V(G) melalui beberapa persamaan diantaranya :

V(G) = Jumlah Region = (R1,R2,R3,R4,R5) = 5

V(G) = E – N + 2 = 22 – 19 + 2 = 5

V(G) = P + 1 = 4+ 1 = 5

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 5, ditentukan empat buah basis set dari jalur independen yaitu:

Jalur 1 : B1-B2-B3-B4-B5-B7-B8-B10-B11-B13-B14-B15-B17-B18-And

Jalur 2 : B1-B2-B3-B4-B6-B7-B8-B10-B11-B13-B14-B15-B17-B18-And

Jalur 3 : B1-B2-B3-B4-B5-B7-B9-B10-B11-B13-B14-B15-B17-B18-And

Jalur 4 : B1-B2-B3-B4-B5-B7-B8-B10-B12-B13-B14-B15-B17-B18-And

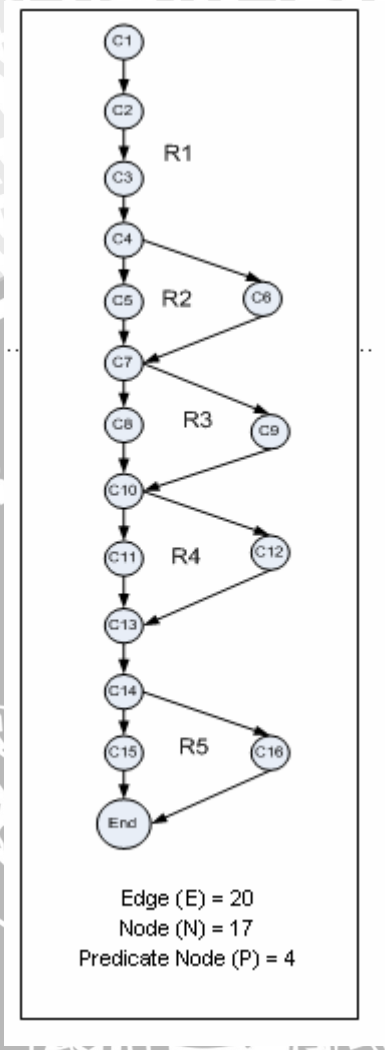
Jalur 5 : B1-B2-B3-B4-B5-B7-B8-B10-B11-B13-B14-B16-B17-B18-And

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah dijelaskan pada tabel 6.3.

**Tabel 6.3.** *Test case* untuk Pengujian Prosedur ScanAutoClick Deteksi Tepi Kiri

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Mencari titik pixel dari tepi kiri benda, perhitungan panjang benda dalam gambar dan panjang benda sebenarnya | Edit5.text = posisi titik pixel tepi kiri benda<br>Edit6.text = panjang benda dalam gambar (pixel)<br>Edit7.text = panjang benda sebenarnya (cm) | Edit5.text = posisi titik pixel tepi kiri benda<br>Edit6.text = panjang benda dalam gambar (pixel)<br>Edit7.text = panjang benda sebenarnya (cm) |
| 2 | Jika nilai d positif maka d tidak dikalikan dengan (-1) | d positif (d selalu positif) | d positif (d selalu positif) |
| 3 | Jika nilai d positif maka e tidak dikalikan dengan (-1) | e positif (e selalu positif) | e positif (e selalu positif) |
| 4 | Jika nilai d positif maka f tidak dikalikan dengan (-1) | f positif (d selalu positif) | f positif (d selalu positif) |
| 5 | Jika kondisi (d<u)AND(f<u)AND(e<u)) tidak terpenuhi | Posisi pixel dari tepi kiri tidak ditemukan sehingga edit5.text=”0” (nilai awal edit5.text=”0”) | Edit5.text=”0” |

**Sumber** : [Pengujian]




**Gambar 6.4.** Pemodelan Algoritma Prosedur ScanAutoClick Deteksi Tepi Bawah ke dalam *flow graph*.

**Sumber** : [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap Prosedur ScanAutoClick deteksi tepi bawah menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* atau V(G) melalui beberapa persamaan diantaranya :

V(G) = Jumlah Region = (R1,R2,R3,R4,R5) = 5

V(G) = E – N + 2 = 20 – 17 + 2 = 5

V(G) = P + 1 = 4+ 1 = 5

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 5, ditentukan empat buah basis set dari jalur independen yaitu:

Jalur 1 : C1-C2-C3-C4-C5-C7-C8-C10-C11-C13-C14-C15-And

Jalur 2 : C1-C2-C3-C4-C6-C7-C8-C10-C11-C13-C14-C15-And

Jalur 3 : C1-C2-C3-C4-C5-C7-C9-C10-C11-C13-C14-C15-And

Jalur 4 : C1-C2-C3-C4-C5-C7-C8-C10-C12-C13-C14-C15-And

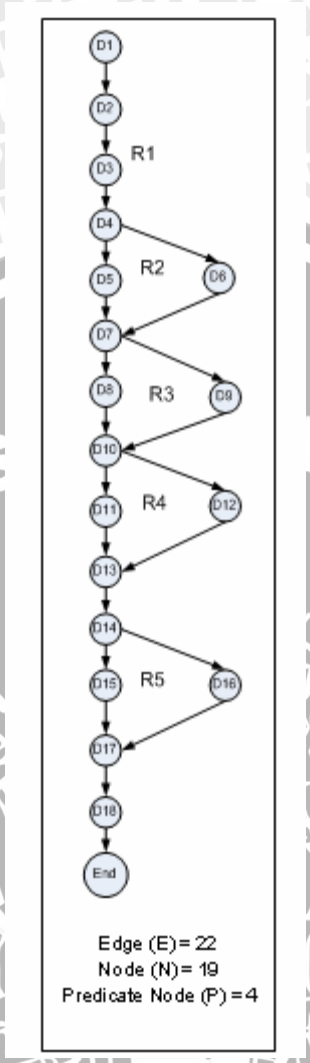
Jalur 5 : C1-C2-C3-C4-C5-C7-C8-C10-C11-C13-C14-C16-And

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah dijelaskan pada Tabel 6.4.

**Tabel 6.4.** *Test case* untuk Pengujian Prosedur Scanautoclick Deteksi Tepi Bawah

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Mencari titik pixel dari tepi bawah benda | Edit8.text= posisi titik pixel tepi bawah benda | Edit8.text= posisi titik pixel tepi bawah benda |
| 2 | Jika nilai d positif maka d tidak dikalikan dengan (-1) | d positif (d selalu positif) | d positif (d selalu positif) |
| 3 | Jika nilai d positif maka e tidak dikalikan dengan (-1) | e positif (e selalu positif) | e positif (e selalu positif) |
| 4 | Jika nilai d positif maka f tidak dikalikan dengan (-1) | f positif (d selalu positif) | f positif (d selalu positif) |
| 5 | Jika kondisi (d<u)AND(f<u)AND(e<u)) tidak terpenuhi | Posisi pixel dari tepi bawah tidak ditemukan sehingga edit8.text=”0” (nilai awal edit8.text=”0”) | Edit8.text=”0” |

**Sumber** : [Pengujian]




**Gambar 6.5.** Pemodelan algoritma Prosedur ScanAutoClick deteksi tepi atas ke dalam *flow graph.*

**Sumber** : [Pengujian]

Pemodelan ke dalam *flow graph* yang telah dilakukan terhadap prosedur ScanAutoClick deteksi tepi atas menghasilkan jumlah kompleksitas siklomatis *(cyclomatic complexity)* atau V(G) melalui beberapa persamaan diantaranya :

V(G) = Jumlah Region = (R1,R2,R3,R4,R5) = 5

V(G) = E – N + 2 = 22 – 19 + 2 = 5

V(G) = P + 1 = 4+ 1 = 5

Dari nilai *cyclomatic complexity* yang telah dihasilkan dari perhitungan yaitu 5, ditentukan empat buah basis set dari jalur independen yaitu:

Jalur 1 : D1-D2-D3-D4-D5-D7-D8-D10-D11-D13-D14-D15-D17-D18-And

Jalur 2 : D1-D2-D3-D4-D6-D7-D8-D10-D11-D13-D14-D15-D17-D18-And

Jalur 3 : D1-D2-D3-D4-D5-D7-D9-D10-D11-D13-D14-D15-D17-D18-And

Jalur 4 : D1-D2-D3-D4-D5-D7-D8-D10-D12-D13-D14-D15-D17-D18-And

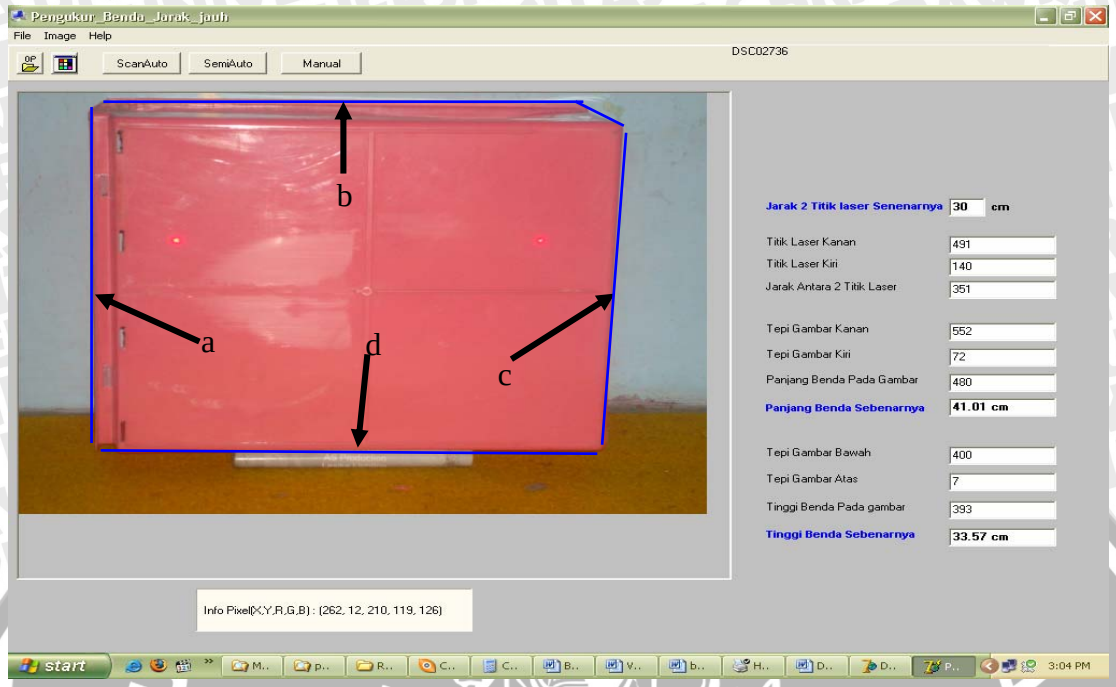
Jalur 5 : D1-D2-D3-D4-D5-D7-D8-D10-D11-D13-D14-D16-D17-D18-And

Penentuan kasus uji *(test case)* untuk masing-masing jalur dan hasil eksekusi untuk masing-masing kasus uji adalah dijelaskan pada tabel 6.5.

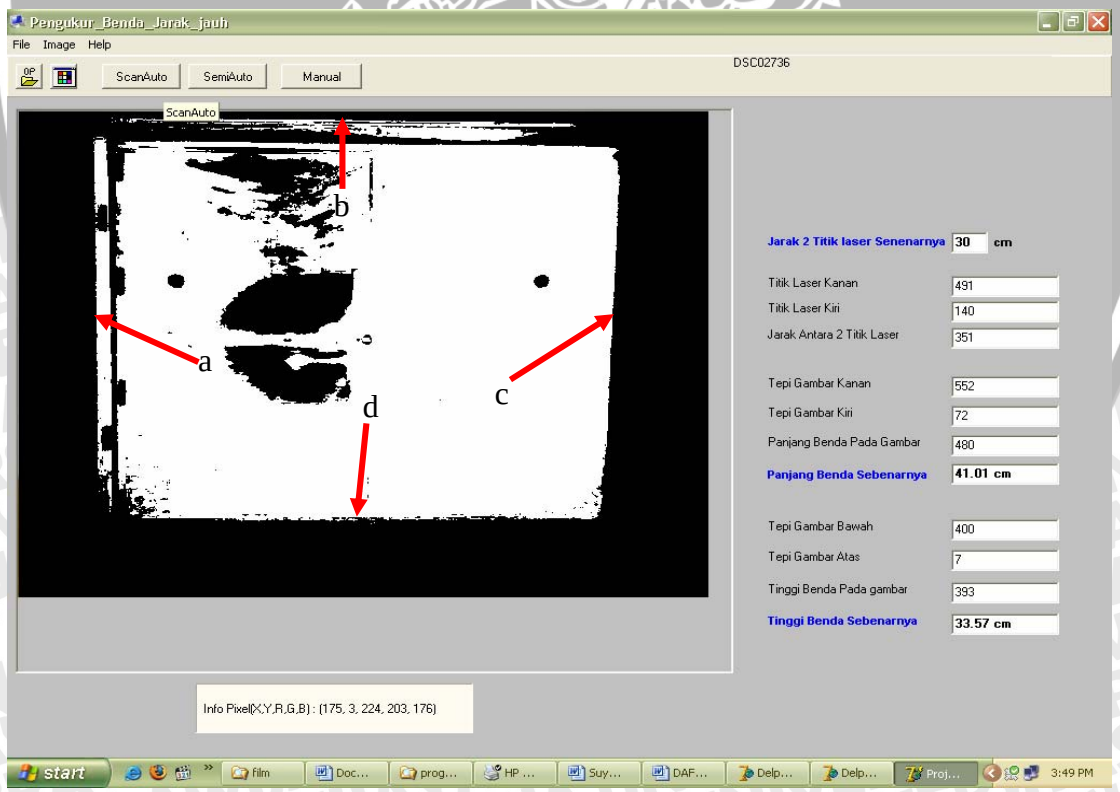
**Tabel 6.5.** *Test case* untuk pengujian Prosedur ScanAutoClick deteksi tepi atas

| Jalur | Kasus Uji | Hasil yang diharapkan | Hasil yang didapatkan |
|---|---|---|---|
| 1 | Mencari titik pixel dari tepi atas benda, perhitungan panjang benda dalam ambar dan panjang benda sebenarnya | Edit9.text = posisi titik pixel tepi atas benda<br>Edit10.text = panjang benda dalam gambar (pixel)<br>Edit11.text = panjang benda sebenarnya (cm) | Edit9.text = posisi titik pixel tepi atas benda<br>Edit10.text = panjang benda dalam gambar (pixel)<br>Edit11.text = panjang benda sebenarnya (cm) |
| 2 | Jika nilai d positif maka d tidak dikalikan dengan (-1) | d positif (d selalu positif) | d positif (d selalu positif) |
| 3 | Jika nilai d positif maka e tidak dikalikan dengan (-1) | e positif (e selalu positif) | e positif (e selalu positif) |
| 4 | Jika nilai d positif maka f tidak dikalikan dengan (-1) | f positif (d selalu positif) | f positif (d selalu positif) |
| 5 | Jika kondisi (d<u)AND(f<u)AND(e<u)) tidak terpenuhi | Posisi pixel dari tepi atas tidak ditemukan sehingga edit9.text=”0” (nilai awal edit9.text=”0”) | Edit9.text=”0” |

**Sumber** : [Pengujian]

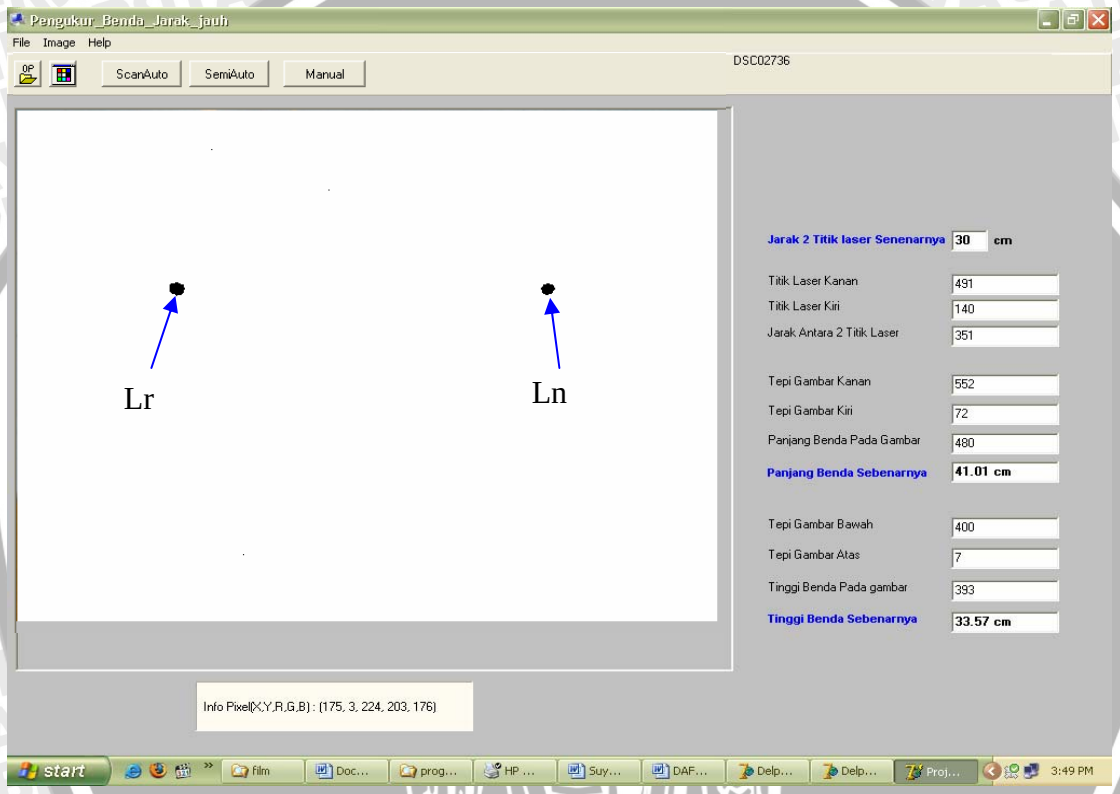



**Gambar 6.6.** Gambar Hasil ScanAutoClick




**Gambar 6.7.** Gambar Hasil ScanAutoClick (Deteksi Tepi)

Dari gambar 6.5 dan gambar 6.7 diketahuai untuk tepi-tepi dari objek yang akan di ukur, untuk tepi kiri (a) berada pada koordinat pixel x = 72 , untuk tepi atas (b) berada pada koordinat pixel y = 7, untuk tepi kanan (c) berada pada koordinat pixel x = 552, untuk tepi bawah (d) berada pada koordinat pixel y = 400, sehingga panjang objek dalam gambar adalah 552 – 72 = 480 dan tinggi objek dalam gambar adalah 400 – 7 = 393.




**Gambar 6.8.** Gambar Hasil ScanAutoClick (Deteksi titik laser)

Dari gambar 6.8 diketahui untuk titik laser sebelah kiri (Lr) berada pada koordinat pixel x = 140, untuk titik laser sebelah kiri (Ln) berada pada koordinat pixel x = 491, sehingga jarak dua titik laser dalam gambar adalah 491 – 140 = 351. sehingga dapat dihitung untuk panjang objek sebenarnya adalah 41,01 cm dan tinggi objek sebenarnya adalah 33,57 cm.

## 6.2 *Black-Box Testing*

Pada pengujian B*lack-Box Testing* kita tidak perlu tahu apa yang sesungguhnya terjadi pada sistem atau perangkat lunak. Yang kita uji adalah masukan dan keluaranya. Artinya, dengan berbagai masukan yang kita berikan, apakah sistem atau perangkat lunak memberikan keluaran seperti yang kita harapkan? Dalam pengujian ini kita akan menggunakan use case diagram serta sekenario yang kita kembangkan saat analisis sebagai panduan. Apakah keluaran sesuai dengan harapan serta kebutuhan pengguna[AGN-05:435-436].

Sistem atau perangkat lunak pengukur panjang dengan metode pengolahan citra ini dibuat dengan citra atau gambar sebagai input dari sistem tersebut. Citra atau gambar yang akan digunakan sebagai input program adalah citra atau gambar dari benda yang akan diukur panjang dan tingginya. Pengambilan citra dilakukan oleh user dengan bantuan kamera dan dua laser yang telah disusun sejajar dengan jarak tertentu, sehingga kamera dapat menangkap seluruh gambar benda yang akan diukur dan gambar dari dua titik laser tersebut. Gambar titik-titik laser yang tertangkap oleh kamera akan digunakan program sebagai acuan untuk pengukuran panjang dan tinggi benda tersebut.

## 6.3 Analisis statistik

Untuk melakukan pengujian Analisa statistik sistem atau perangkat lunak pengukur panjang dengan metode pengolahan citra akan di lakukan percobaan sistem dengan masukan beberapa gambar yang telah diketahui ukuran dari panjang dan tinnginya. Kemudian output dari program akan dibandingkan dengan ukuran dari panjang dan tinggi benda sebenarnya sehingga dapat dihitung nilai rata-rata (*arithmetic mean*), Penyimpangan terhadap nilai rata-rata, penyimpangan rata-rata (*average deviation*) dan deviasi standar kesalahan dari sistem tersebut.

Analisis statistik terhadap data pengukuran adalah pekerjaan yang biasa sebab memungkinkan penentuan ketidakpastian hasil pengujian akhir secara analitis. Hasil dari suatu pengukuran dengan metode tertentu dapat diramalkan berdasarkan data

contoh tanpa memiliki informasi yang lengkap mengenai semua faktor-faktor gangguan [JTK-09].

1. Nilai rata-rata (*arithmetic mean*)

   Nilai rata-rata diberikan oleh persamaan berikut :

$$\bar{x} = \frac{x_1 + x_2 + x_3 + x_4 + ..... + x_n}{n} = \frac{\sum x}{n}$$

   Dimana $\bar{x}$ = nilai rata-rata

   $x_1, x_2, x_3$ = pembacaan yang dilakukan

   $n$ = jumlah pembacaan

2. Penyimpangan terhadap nilai rata-rata

   Penyimpangan (deviasi) adalah selisih antara suatu pembacaan terhadap nilai rata-rata dalam sekelompok pembacaan. Jika penyimpangan pembacaan pertama $x_1$ adalah $d_1$, penyimpangan pembacaan kedua $x_2$ adalah $d_2$, dan seterusnya, maka penyimpangan-penyimpangan terhadap nilai rata-rata adalah :

$$d_1 = x_1 - \bar{x} \qquad d_2 = x_2 - \bar{x} \qquad d_n = x_n - \bar{x}$$

3. Penyimpangan rata-rata (*average deviation*)

   Deviasi rata-rata adalah suatu indikasi ketepatan instrumen-instrumen yang digunakan untuk pengukuran. Menurut definisi, deviasi rata-rata adalah penjumlahan nilai-nilai mutlak dari penyimpangan-pemyimpangan dibagi dengan jumlah pembacaan

Deviasi rata-rata dapat dinyatakan sebagai :

$$D = \frac{|d_1| + |d_2| + |d_3| + ..... + |d_n|}{n} = \frac{\sum |d|}{n}$$

4. Deviasi standar

   Deviasi standar merupakan cara yang sangat ampuh untuk menganalisa kesalahan-kesalahan acak secara statistik. Deviasi standar dari jumlah data terbatas didefinisikan sebgai akar dari penjumlahan semua penyimpangan

(deviasi) setelah dikuadratkan dibagi dengan banyaknya pembacaan. Secara sistematis dituliskan :

$$\sigma = \sqrt{\frac{d_1^2 + d_2^2 + d_3^2 + ..... + d_4^2}{n}} = \sqrt{\frac{\sum d_t^2}{n}}$$

Tentunya dalam praktek, jumlah pengamatan yang mungkin adalah terbatas. Deviasi standar untuk sejumlah data *terbatas* adalah :

$$\sigma = \sqrt{\frac{d_1^2 + d_2^2 + d_3^2 + ..... + d_4^2}{n-1}} = \sqrt{\frac{\sum d_t^2}{n-1}}$$

Pada tabel 6.6 dan 6.7 akan dijelaskan output-output dari sistem pengukur panjang dengan metode pengolahan citra [JTK-09].

**Tabel 6.6** Data Keluaran Program Dengan Input Berbagai Macam Warna Benda.

| No | Input program (gambar box) | Laser kanan (pixel) | Laser kiri (pixel) | Jarak antara laser (pixel) | Tepi kanan (pixel) | Tepi kiri (pixel) | Panjang benda dalam gambar (pixel) | Panjang benda sebenarnya (cm) | Tepi bawah (pixel) | Tepi atas (pixel) | Tinggi benda dalam gambar (pixel) | Tinggi benda sebenarnya (cm) | keterangan |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Warna biru | 494 | 138 | 356 | 574 | 83 | 491 | 41,37 | 431 | 51 | 380 | 32,01 | Scan auto |
| 2 | Warna hijau | 549 | 216 | 333 | 596 | 136 | 460 | 41,43 | 409 | 44 | 365 | 32,88 | Scan auto |
| 3 | Warna merah | 469 | 192 | 277 | 514 | 133 | 381 | 41,25 | 392 | 94 | 298 | 32,25 | Semi auto |
| 4 | Warna kuning | 435 | 204 | 231 | 469 | 151 | 318 | 41,28 | 368 | 121 | 247 | 32,07 | Scan auto |
| 5 | Warna merah mudah | 491 | 140 | 351 | 552 | 72 | 480 | 41,01 | 400 | 14 | 386 | 32,97 | Scan auto |

**Sumber** : [Pengujian]

Nilai rata-rata

$$\bar{x} = \frac{41{,}37 + 41{,}43 + 41{,}25 + 41{,}28 + 41{,}01}{5}$$

$$\bar{x} = 41{,}268 \ \ cm$$

$\therefore$ nilai rata-rata panjang box adalah 44,064 $cm$

$$\bar{y} = \frac{32{,}01 + 32{,}88 + 32{,}25 + 32{,}07 + 32{,}97}{5}$$

$$\bar{y} = 32{,}436 \ \ cm$$

$\therefore$ nilai rata2 tinggi box adalah 34,65 $cm$

Penyimpangan terhadap nilai panjang rata-rata ( $\bar{x}$ )

$d_1 = 41{,}37 - 41{,}268$ $\quad$ $d_1 = 102 \ \ cm$

$d_2 = 41{,}43 - 41{,}268$ $\quad$ $d_2 = 0{,}162 \ \ cm$

$d_3 = 41{,}25 - 41{,}268$ $\quad$ $d_3 = -0{,}018 \ \ cm$

$d_4 = 41{,}28 - 41{,}268$ $\quad$ $d_4 = 0{,}012 \ \ cm$

$d_5 = 41{,}01 - 41{,}268$ $\quad$ $d_5 = -0{,}258 \ \ cm$

Penyimpangan terhadap nilai tinggi rata-rata ( $\bar{y}$ )

$d_1 = 32{,}01 - 32{,}436$ $\quad d_1 = -0{,}426 \ \ cm$

$d_2 = 32{,}88 - 32{,}436$ $\quad d_2 = 0{,}444 \ \ cm$

$d_3 = 32{,}25 - 32{,}436$ $\quad d_3 = -0{,}186 \ \ cm$

$d_4 = 32{,}07 - 32{,}436$ $\quad d_4 = -0{,}366 \ \ cm$

$d_5 = 32{,}97 - 32{,}436$ $\quad d_5 = 0{,}534 \ \ cm$

Deviasi rata-rata untuk panjag

$$D = \frac{0{,}102 + 0{,}162 + 0{,}018 + 0{,}012 + 0{,}258}{5}$$

$D = 0{,}1104 \ \ cm$

Deviasi rata-rata untuk tinggi

$$D = \frac{0{,}426 + 0{,}444 + 0{,}186 + 0{,}366 + 0{,}534}{5}$$

$D = 0{,}3912 \ \ cm$

Defiasi standart untuk pengukuran panjang

$$\sigma = \sqrt{\frac{0{,}102^2 + 0{,}162^2 + 0{,}018^2 + 0{,}012^2 + 0{,}258^2}{5}}$$

$\sigma = 0{,}144$

Defiasi standart untuk pengukuran tinggi

$$\sigma = \sqrt{\frac{0{,}426^2 + 0{,}444^2 + 0{,}186^2 + 0{,}366^2 + 0{,}534^2}{5}}$$

$\sigma = 0{,}408$

Deviasi standar untuk sejumlah data *terbatas* (panjang)

$$\sigma = \sqrt{\frac{0{,}102^2 + 0{,}162^2 + 0{,}018^2 + 0{,}012^2 + 0{,}258^2}{4}}$$

$\sigma = 0{,}16$

Deviasi standar untuk sejumlah data *terbatas* (tinggi)

$$\sigma = \sqrt{\frac{0{,}426^2 + 0{,}444^2 + 0{,}186^2 + 0{,}366^2 + 0{,}534^2}{4}}$$

$\sigma = 0{,}46$

**Tabel 6.7** Data keluaran program dengan input benda dari ukuran terkecil sampai terbesar

| No | Input program Benda dari ukuran terkecil sampai terbesar | Laser kanan (pixel) | Laser kiri (pixel) | Jarak antara laser (pixel) | Tepi kanan (pixel) | Tepi kiri (pixel) | Panjang benda dalam gambar (pixel) | Panjang benda sebenarnya (cm) | Keterangan Menggunakan scan | Deviasi (cm) dan persentasi error | Jarak Antara objek dengan sensor |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Kertas putih (5,1 cm) | 467 | 263 | 204 | 280 | 244 | 36 | 5,28 | Manual (error) | 0,18(3,5%) | 1 meter |
| 2 | Kertas putih (7,5 cm) | 473 | 253 | 220 | 221 | 224 | 57 | 7,77 | Manual (error) | 0,27(3,6%) | 1 meter |
| 3 | Kertas putih (10,5 cm) | 582 | 215 | 367 | 278 | 151 | 127 | 10,38 | Auto (bisa) | 0,12 (1,1%) | 1 meter |
| 4 | Kertas minyak (19,5cm) | 481 | 260 | 221 | 334 | 192 | 142 | 19,26 | Auto (bisa) | 0,24(1,2%) | 1,5 meter |
| 5 | Tembok putih (300 cm) | 1715 | 1456 | 259 | 2810 | 216 | 2594 | 300,45 | Auto (bisa) | 0,45(0,15%) | 3 meter |
| 6 | Tembok biru (755 cm) | 1174 | 1068 | 106 | 2754 | 116 | 2638 | 746,58 | Auto (bisa) | 9(1,2%) | 7 meter |
| 7 | Tembok biru (1026 cm) | 1725 | 1643 | 82 | 3047 | 12 | 3035 | 1110,36 | Semi Auto (bisa) | 15,64(1,4%) | 10 meter |
| 8 | Tembok biru (1470 cm) | 1627 | 1571 | 56 | 2932 | 313 | 2619 | 1403,01 | Manual (error) | 66,99(4,6%) | 15 meter |

**Sumber** : [Pengujian]

Dari perhitungan data yang di peroleh pada tabel 6.6

Deviasi terbesar standar untuk sejumlah data *terbatas*

$\sigma = 0{,}46$

Sehingga dapat dihitung maximal untuk persentasi error adalah $\sigma/y = 0{,}46/32{,}436 \times 100\%$

$= 1{,}4\%$

Untuk perhitungan pada No. 1,2 dan 8 pada tabel 6.7 dianggap error karena

nilai persentasi error >1,4%

Sehingga dapat disimpulkan perhitungan akan lebih akurat untuk batas

terkecil > 10 cm dan

terbesar < 10 m

**Tabel 6.8** Data hasil pengukuran dengan input objek yang diambil dari jarak 1 meter sampai dengan 5 meter dari sensor

| No | Jarak objek dengan sensor | Panjang gambar sebenarnya | Hasil pengukuran | Deviasi | Persentasi error |
|---|---|---|---|---|---|
| 1 | 1 meter | 72.5 cm | 71.97 cm | 0.53 cm | 0.70% |
| 2 | 2 meter | 72.5 cm | 71.94 cm | 0.56 cm | 0.77% |
| 3 | 3 meter | 72.5 cm | 73.11 cm | 0.61 cm | 0.84% |
| 4 | 4 meter | 72.5 cm | 73.14 cm | 0.64 cm | 0.88% |
| 5 | 5 meter | 72.5 cm | 73.32 cm | 0.82 cm | 1.13% |

**Sumber** : [Pengujian]

Dari data hasil pengukuran dengan input objek yang diambil dari jarak 1 meter sampai dengan 5 meter dari sensor dapat disimpulkan bahwa Semakin jauh jarak pengambilan gambar maka semakin besar kesalahan pengukuran tersebut.

# BAB VII
# PENUTUP

Berdasarkan hasil-hasil yang dicapai selama perancangan, pembuatan dan pengujian dari penelitian ini, maka dapat diambil beberapa kesimpulan dan saran.

## 7.1. Kesimpulan

Setelah melakukan pengujian dan analisa pada sistem pengukur panjang dengan metode pengolahan citra maka kesimpulan yang didapat antara lain :

1. Pengambilan gambar diperlukan kecermatan karena dipengaruhi oleh *noise-noise* yang ada disekitar obyek sehingga diupayakan sistem dapat mengenali secara otomatis dimana titik-titik laser yang ada pada obyek, dan juga tepi-tepi gambar obyek.
2. Untuk obyek yang titik-titik laser dan tepi-tepi obyeknya tidak terdeteksi oleh sistem, User dapat memilih menu kedua dan ketiga yaitu Semi auto dan manual yang telah disediakan sistem.
3. Dengan melakukan pengujian beberapa kotak yang panjang dan tingginya sama didapatkan persentasi maksimal kesalahan sistem yaitu 1,4% dari panjang benda.
4. Dengan melakukan pengujian beberapa kotak yang warnanya berbeda-beda disimpulkan bahwa untuk benda dengan warna merah tidak dapat dilakukan pengukuran secara otomatis karena warna titik laser akan melebar.
5. Dengan melakukan pengujian benda dari ukuran terkecil sampai terbesar dan persentasi maksimal kesalahan 1,4% dari panjang benda maka didapatkan batas untuk penjang benda terkecil dan terbesar yang dapat diukur oleh sistem yaitu 10 cm untuk batas terkecil dan 10 m untuk batas terbesar.
6. Semakin jauh jarak pengambilan gambar maka semakin besar kesalahan pengukuran tersebut.

## 7.2. Saran

hasil penelitian ini masih terdapat kekurangan yang masih bisa diperbaiki pada penelitian selanjutnya di tahun yang akan datang. Adapun beberapa kekurangan yang perlu diperbaiki pada penelitian yang akan datang adalah:

1. Diharapkan sisitem bisa mengenali titik-titik dari laser dan tepi-tepi obyek tanpa terpengaruh oleh warna dari obyek dan *noise-noise* yang ada pada sekitar obyek.
2. Diharapkan sistem bisa mengukur untuk benda yang ukurannya lebih kecil dari 10 cm dan benda yang ukuranya lebih besar dari 10 m dengan maksimal persentasi kesalahan lebuh kecil dari1,4%.
3. Diharapkan sistem bisa mengukur panjang benda dengan lebih akurat meskipun pengambilan gambar tidak sejajar dengen bidang kamera.
4. Diharapkan sistem bisa menjadi aplikasi tambahan pada sebuah kamera digital sehingga untuk melakukan pengukuran panjang benda bisa langsung dilakukan pada kamera tersebut.

## DAFTAR PUSTAKA


[ZTF-08] Fadlisya, Taufiq, Zulfikar & Fauzan . *Pengolahan Citra Menggunakan Delphi*, Graha Ilmu.Yogyakarta, 2008.

[NUA-05] Ahmad Usman, *Pengolahan Citra Digital & Teknik Pemrogramanya*. Graha Ilmu, Yogyakarta, 2005.

[FPB-05] Achmad B. Jozua F. Palandi, Fatchurrochman, *Pengolahan Citra Digital Menggunakan Visual Basic*. Graha Ilmu,Yogyakarta, 2005

[NBR-05] Sigit R,Basuki A, Ramadijanti N, Pramadihanto D, *Step by Step Pengolahan Citra Digital.* Andi, Yogyakarta 2005.

[FAH-07] Alfattah H.. *Analisi dan Perancangan Sistem Informasi untuk Keunggulan Bersaing Perusahaan dan Organisasi Modern.* Andi, Yogyakarta 2007.

[AGN-05] Nugroho A. *Analisi dan Perancangan Sistem Informasi dengan Metode Berorientasi Objek.* Informatika, Bandung 2005.

[IDE-07] Indriyawan E. *Delphi 2007 for Win32*, Surabaya, 2007 http://ekoindri.wordpress.com/sejarah/ , tanggal akses 28 mei 2009.

[LSR-09] Anonymous. 2009. *laser*. Akses dari: http://wikipedia.org/wiki/Laser, tanggal akses 28 maret 2009

[CJS-07] Simarmata J. Candra T. *Grafika Komputer*, Andi, Yogyakarta 2007 .

[LBA-06] Al Bahra bin Ladjamuddin B. *Rekayasa Perangkat lunak*, Graha Ilmu, Yogyakarta 2006.

[OJP-07] Pujianto.Scom. *Teknik pemrograman delphi 8.0*, PT Elex Media Komputindo, Jakarta 2007.

[RNM-04] Munir R. *Pengolahan Citra Digital dengan pendekatan alogaritmik*, Informatika, Bandung 2004.

[PSB-09] Busono P., Ir. M.Kom . *Testing dan Implementasi*, Jakarta 2009, http://pksm.mercubuana.ac.id/modul/18019-6-697941132502.doc, tanggal akses 11 agustus 2009.

[ARS-09] Syarif A.,ST.,MT.,*Rekayasa Perangkat Lunak* http://pksm.mercubuana.ac.id/modul/18011-9-536782823857.doc , Tanggal akses 11 agustus 2009.

[JTK-09] Kustija J., M., Sc., *Sistem Instrumentasi Elektronika*, http://pksm.mercubuana.ac.id/modul/14051-1-614357836792.doc, tanggal akses 11 agustus 2009.