SKRIPSI

APLIKASI TEKNOLOGI *LASER RANGE FINDER* UNTUK PENGUKURAN JARAK SUATU OBJEK MENGGUNAKAN BAHASA PEMPROGRAMAN JAVA

Disusun oleh :

ADELIA AMITYAS
NIM 03.17.016

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

SEPTEMBER 2007

LEMBAR PERSETUJUAN

APLIKASI TEKNOLOGI LASER RANGE FINDER UNTUK PENGUKURAN JARAK SUATU OBJEK MENGGUNAKAN BAHASA PEMPROGRAMAN JAVA

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektronika Strata Satu (S-1)

Disusun Oleh :

ADELIA AMITYAS
NIM : 03.17.016

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Ir. H. Sidik Noertjahjono, MT	**M. Ashar, ST, MT**
NIP. 1028700167	**NIP.1030500408**

Mengetahui
Ketua Jurusan Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP.Y 1039500274

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

2007

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama	: Adelia Amityas
NIM	: 03.17.016
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika
Judul Skripsi	: Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman Java

Dipertahankan di hadapan majelis penguji Skripsi jenjang Strata satu (S-1) pada :

Hari	: Senin
Tanggal	: 3 September 2007
Dengan Nilai	: 80.5 (A)

Ketua Majelis Penguji	Sekretaris Majelis Penguji
(Ir. Mochtar Asroni, MSME)	**(Ir. F. Yudi Limpraptono, MT)**
NIP.Y.1018100036	NIP.Y.1039500274

Penguji I	Penguji II
(Joseph Dedy Irawan, ST, MT)	**(I Komang Somawirata, ST, MT)**
NIP.P.1039800324	NIP.P. 1030100361

KATA PENGANTAR

Dengan memanjatkan puji syukur kepada Tuhan Yang Maha Esa, atas berkat rahmat dan karunia-Nya sehingga dapat menyelesaikan skripsi yang berjudul "Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Java " ini dengan lancar. Skripsi ini merupakan persyaratan kelulusan studi di jurusan Teknik Elektro S-1 konsentrasi Teknik Elektronika ITN Malang dan untuk mencapai gelar sarjana teknik.

Keberhasilan penyelesaian laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penyusun menyampaikan terimakasih kepada:

1. Bapak Prof. DR. Ir. Abraham Lomi, MSME selaku Rektor ITN Malang.
2. Bapak Ir. Mochtar Asroni, MSME selaku Dekan Fakultas Teknologi Industri.
3. Bapak Ir. F. Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S-1.
4. Bapak Ir. H. Sidik Noertjahjono, MT selaku Dosen Pembimbing I.
5. bapak M. Ashar, ST, MT selaku Dosen Pembimbing II.
6. Orang tua serta saudara-saudara kami yang telah memberikan doa restu, dorongan, semangat dan biaya.
7. Semua yang telah membantu dalam penyelesaian penyusunan skripsi ini.

Penyusun telah berusaha semaksimal mungkin dan menyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelesaikan laporan ini, untuk itu penyusun mengharapkan saran dan kritik yang membangun dari pembaca demi kesempuranaan laporan ini.

Harapan penyusun semoga laporan ini memberikan manfaat bagi perkembangan ilmu pengetahuan dan pembaca.

Malang, November 2007

Penulis

ABSTRAKSI

APLIKASI TEKNOLOGI LASER RANGE FINDER UNTUK PENGUKURAN JARAK SUATU OBJEK MENGGUNAKAN BAHASA PEMPROGRAMAN JAVA

Adelia Amityas, 0317016, Jurusan Teknik Elektronika S-1
Dosen Pembimbing I : Ir. H. Sidik Noertjahjono, MT
Dosen Pembimbing II : M. Ashar, ST, MT

Pada makalah ini telah direalisasikan sebuah aplikasi dari teknologi laser rangefinder untuk pengukuran jarak suatu objek menggunakan bahasa pemprograman java, dimana pada aplikasi ini memanfaatkan sinar laser pointer sebagai sumber cahaya dan webcam yang letaknya disesuaikan sehingga dapat menangkap hasil proyeksi berkas sinar laser yang dipantulkan oleh target tersebut kemudian webcam akan mengirimkan data digital yang mengidikasikan koordinat pixel dari objek yang paling terang ke PC dengan interface USB sehingga dengan matematika trigonometri sederhana aplikasi program java pada PC akan langsung dapat menghitung dan memunculkan jarak objek tersebut pada layer monitor dengan jarak minimal sebesar 25 cm dan maksimalnya sebesar103 cm dimana terdapat error sebesar 1.183 %.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

Pada bab ini berisi mengenai materi yang memberikan penggambaran secara umum hal-hal yang berhubungan dengan penulisan tentang skripsi.

1.1. LATAR BELAKANG

Teknologi *Light Amplification by Stimulated Emission of Radiation* (LASER) berdasarkan ide pemikiran Dr. Albert Einstein terus berkembang sejak ditemukan pada tahun 1958. Saat ini penggunaan laser mencakup berbagai bidang, seperti kesehatan, hiburan, teknologi informasi dan komunikasi, robotika, militer, hingga sistem teknologi pengamanan.

Beranjak dari ide memikirkan teknologi LASER ini lah, kami mencoba mengemukakan sesuatu yang baru, yaitu teknologi Laser Radar (LADAR). Tekonologi ini dikenal juga dengan teknologi Area Laser Sensor (ALS) dengan memanfaatkan teknologi *Laser Range Finder* (LRF) untuk mengukur jarak dengan memanfaatkan sinar laser. Dimana sistem yang akan dibuat nanti menggunakan teknik *Digital Image Processing* atau Pengolahan Citra Digital yang dilakukan dengan bantuan komputer untuk menganalisa dan mengkalibrasi pixel pada berkas laser yang jatuh pada objek kemudian dipantulkan sehingga dapat ditangkap oleh webcam dan kemudian diproses oleh *software* untuk memperoleh informasi jarak.

1.2. RUMUSAN MASALAH

Rumusan Masalah yang akan dibahas pada skripsi ini adalah bagaimana teknologi *Laser Range Finder* dapat mendeteksi dan mengukur jarak suatu objek melalui PC.

1.3. TUJUAN

Tujuan dari skripsi ini adalah merancang sistem untuk mengukur jarak suatu objek dengan teknologi *Laser Range Finder* menggunakan bahasa pemprograman java yang diinterfacekan ke PC.

1.4. BATASAN MASALAH

Batasan masalah dalam pembuatan Skripsi ini adalah :

1. Menggunakan Webcam Genius VidioCAM Express 100K Pixels USB Video Resolution 352x288, *Frame rate* 30 fps, *Max Color depth* 24 bits, capture format JPEG dan menggunakan chip CMOS.
2. Tidak membahas tentang pengaruh frekuensi yang ada diarea sinar laser.
3. Menggunakan thresholding (ambang batas) sebesar 192 pada program aplikasi.

1.5. METODOLOGI

Skripsi ini bersifat aplikatif yang ditandai dengan adanya perencanaan dan perancangan sistem. Berikut adalah metodologi yang dilakukan dalam proses perencanaan sistem ini :

1. **Studi literatur**

   Yaitu dengan melakukan studi kepustakaan dari berbagai sumber untuk memperoleh berbagai teori serta gambaran tentang masalah yang akan dibahas untuk mempermudah proses perancangan pada alat ini.

2. **Perencanaan dan pembuatan alat**

   Dalam pembuatan alat ini menggunakan konsep sebagai berikut :

   a) Perancangan sistem secara keseluruhan (pembuatan blok diagram sistem).

   b) Mendeskripsikan fungsi dari masing-masing blok diagram.

   c) Membuat perangkat keras *(hardware)* dan perangkat lunaknya *(software)*.

   d) Implementasi *software* yang telah dirancang ke dalam perangkat keras.

3. **Pengambilan kesimpulan**

   Pengambilan kesimpulan didasarkan atas pengujian dan pengukuran dari alat yang telah dibuat.

1.6 SISTEMATIKA PENULISAN

Penulisan dalam Skripsi ini terdiri dari beberapa bab dan sub bab, antara lain :

1. **BAB I**, sebagai pendahuluan yang terdiri dari latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi dan sistematika penulisan.
2. **BAB II**, sebagai landasan teori yang berisi :

   2.1. Citra Digital

   2.2. Pengolahan Citra Digital

   2.3. Model Citra

   2.4. Segmentasi

   2.5. Grayscale

   2.6. Tresholding

   2.7. JAVA

   2.8. LASER

   2.9. Teori Dasar Teknologi *Laser Range Finder*

   2.10. Paralel Port (LPT1)

   2.11. Komunikasi Port Paralel Menggunakan JAVA
3. **BAB III**, membahas tentang perencanaan dan pembuatan *hardware* dan *software* dari sistem.
4. **BAB IV**, membahas tentang pengujian dan pengukuran alat yang terdiri dari *hardware* dan *software.*
5. **BAB V**, penutup yang terdiri dari kesimpulan dan saran.

BAB II

TEORI DASAR

Pada bab ini akan dibahas mengenai teori-teori yang akan dijadikan materi penunjang dalam pembuatan skripsi ini. Dari teori tersebut akan dijadikan acuan dalam perencanaan dan pembuatan perangkat lunak dan peangkat keras.

2.1. Citra Digital

Citra adalah representasi dari dua dimensi untuk bentuk-bentuk fisik nyata tiga dimensi. Dalam perwujudannya, citra dibagi menjadi dua yaitu citra diam (*still images*) dan citra bergerak (*moving images*). Citra diam adalah citra tunggal yang tidak bergerak. Sedangkan citra bergerak adalah rangkaian citra diam yang ditampilkan secara sekuensial sehingga memberi kesan pada mata kita sebagai gambar yang bergerak. Proses transformasi dari bentuk tiga dimensi menjadi bentuk dua dimensi untuk menghasilkan citra akan dipengaruhi oleh bermacam-macam faktor yang mengakibatkan penampilan suatu benda tidak sama persis dengan bentuk fisik nyatanya. Faktor-faktor tersebut merupakan efek degradasi atau penurunan kualitas yang dapat berupa rentang kontras, distorsi geometric, kekaburan dan noise. Agar citra yang mengalami gangguan mudah diinterpretasi baik oleh manusia maupun mesin, maka citra tersebut perlu dimanipulasi menjadi citra lain yang kualitasnya lebih baik atau disebut pengolahan citra (*image processing*). Karena pengolahan citra dilakukan dalam komputer digital, maka citra yang akan diolah terlebih dahulu ditransformasikan kedalam bentuk besaran-

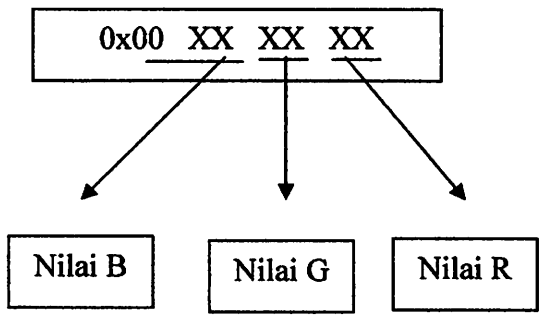
besaran diskrit dari nilai tingkat keabuan pada titik-titik elemen citra. Bentuk dari citra ini disebut citra digital.

2.2. Pengolahan Citra Digital

Image yang diperoleh dari lingkungan masih terdiri dari warna yang sangat komplek sehingga masih diperlukan proses lebih lanjut agar image tersebut dapat untuk memproses sebuah sistem.

Image processing atau sering disebut dengan pengolahan citra digital merupakan suatu metode yang digunakan untuk mengolah atau memproses dari gambar asli sehingga menghasilkan gambar lain yang sesuai dengan kebutuhan. Pengambilan gambar bisa dilakukan oleh kamera video atau alat-alat yang lain yang dapat digunakan untuk mentransfer gambar. Dalam pengolahan citra, dilakukan operasi terhadap citra asli menjadi citra baru berdasarkan citra asli.

Dasar dari pengolahan citra adalah pengolahan warna RGB pada posisi tertentu. Dalam pengolahan citra warna dipresentasikan dengan nilai hexadesimal dari 0x00000000 sampai 0x00ffffff. Warna hitam adalah 0x00000000 dan warna putih adalah 0x00ffffff. Definisi nilai warna di atas seperti gambar 2-1, variabel 0x00 menyatakan angka dibelakangnya adalah hexadecimal.



Gambar 2-1. Nilai warna RGB dalam hexadecimal

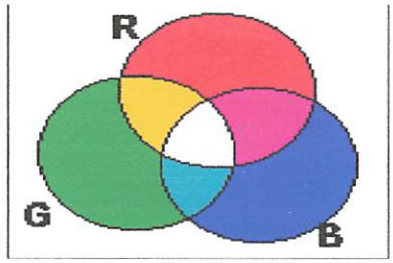
Terlihat bahwa setiap warna mempunyai range nilai 00 (angka desimalnya adalah 0) dan ff (angka desimalnya adalah 255), atau mempunyai nilai derajat keabuan 256 = 2^8. Dengan demikian range warna yang digunakan adalah $(2^8)(2^8)(2^8) = 2^{24}$ (atau yang dikenal dengan istilah *True Colour* pada Windows). Nilai warna yang digunakan di atas merupakan gambungan warna cahaya merah, hijau dan biru seperti yang terlihat pada gambar 2-2. Sehingga untuk menentukan nilai dari suatu warna yang bukan warna dasar digunakan gabungan skala kecerahan dari setiap warnanya.

Dari definisi diatas untuk menyajikan warna tertentu dapat dengan mudah dilakukan, yaitu dengan mencampurkan ketiga warna dasar RGB, Tabel 2-1 berikut memperlihatkan contoh-contoh warna yang bisa digunakan.

Tabel 2-1. Contoh-contoh warna dalam hexadesimal

Nilai	Warna
0x00000000	Hitam
0x000000FF	Merah
0x0000FF00	Hijau
0x00FF0000	Biru
0x0000FFFF	Cyan

Untuk mengetahui kombinasi warna, perlu dibuat suatu program yang dapat menampilkan warna sesuai dengan nilai yang dimasukkan sehingga dapat dicoba berbagai macam kombinasi warna RGB seperti gambar 2-2.



Gambar 2-2. Komposisi warna RGB

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

1. Perbaikan kualitas citra (*image enhancement*)

   Jenis operasi ini bertujuan untuk memperbaiki kualitas citra dengan cara memanipulasi parameter-parameter citra. Dengan operasi ini, ciri-ciri khusus yang terdapat dalam citra lebih ditonjolkan. Contoh-contoh operasi perbaikan citra adalah perbaikan kontras gelap-terang, perbaikan tepian objek (*edge enhancement*), penajaman (*sharpening*), pemberian warna semu (*pseudocoloring*), dan penapisan derau (*noise filtering*)

2. Pemugaran citra (*image restoration*)

   Operasi ini bertujuan untuk menghilangkan atau meminimumkan cacat pada citra. Tujuan pemugaran citra hampir sama dengan operasi perbaikan citra. Bedanya, pada pemugaran citra penyebab degradasi gambar diketahui. Contoh operasi pemugaran citra adalah penghilangan kesamaran (*deblurring*) dan penghilangan derau (*noise*).

3. Pemampatan citra (*image compression*)

---

[1] Rinaldi Munir, PENGOLAHAN CITRA DIGITAL, hal 8-11

Jenis operasi ini dilakukan agar citra dapat direpresentsikan dalam bentuk yang lebih kompak sehingga memerlukan memori yang lebih sedikit. Hal penting yang harus diperhatikan dalam pemampatan adalah citra yang telah dimampatkan harus tetap mempunyai gambar yang bagus.

4. Segmentasi citra (*image segmentation*)

   Jenis operasi ini bertujuan untuk memecah suatu citra kedalam beberapa segmen dengan suatu kriteria tertentu. Jenis operasi ini berkaitan erat dengan pengenalan pola.

5. Scanning citra

   Jenis operasi ini bertujuan menghitung besaran kuantitif dari citra untuk menghasilkan diskripsinya. Teknik scanning citra mengekstraksi ciri-ciri tertentu yang membantu dalam identifikasi objek. Contoh-contoh operasi pengorakan adalah pendeteksian tepi objek (*edge detection*), ekstraksi batas (*boundary*) dan representasi daerah (*region*).

6. Rekontruksi citra (*image reconstruction*)

   Jenis operasi ini bertujuan untuk membentuk ulang objek dari beberapa citra hasil proyeksi. Operasi rekontruksi banyak digunakan dalam bidang medis.

2.3. Model Citra

Citra ada dua macam yaitu citra kontinyu dan citra diskrit. Citra kontinue dihasilkan dari sistem optik yang menerima sinyal analog, misalnya mata manusia dan kamera analog. Citra diskrit dihasilkan melalui proses digitalisasi terhadap citra kontinu. Citra merupakan fungsi kontinue dari intensitas cahaya pada bidang

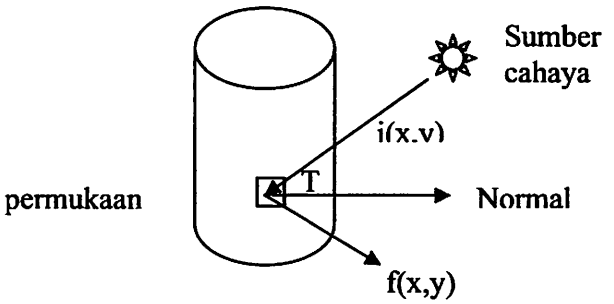
dwimatra. Secara matematis fungsi intensitas cahaya pada bidang dwimatra disimbolkan dengan *f(x, y)* dimana *f* adalah nilai amplitudo pada koordinat pada bidang dwimarta *(x,y)*. [2]Karena cahaya merupakan salah satu bentuk energi yang dalam pernyataan matematis adalah sebagai berikut:

$$0 < f(x, y) < \sim \quad \text{..... (2.1)}$$

Nilai *f(x,y)* sebenarnya adalah hasil kali dari :

i(x,y) = jumlah cahaya yang berasal dari sumbernya *(illumination)*, nilainya antara 0 sampai tidak berhingga, dan

r(x,y) = derajat kemampuan objek memantulkan cahaya *(reflection)* nilainya antara 0 dan 1.



Gambar 2-3. Pembentukkan Citra

Pada gambar 2-3 memperlihatkan proses pembentukan intensitas cahaya. Sumber cahaya menyinari permukaan objek. Jumlah pancaran (iluminasi) cahaya yang diterima objek pada koordinat *(x,y)* adalah *i(x,y)*. Objek memantulkan cahaya yang diterimanya dengan derajat pantulan *r(x,y)*. Hasil kali antara *i(x,y)*

[2] Ibid , hal 15-16

dan *r(x,y)* menyatakan intensitas cahaya pada koordinat *(x,y)* yang ditangkap oleh sensor visual pada sistem optik.

$$\left.\begin{array}{l} f(x,y) = i(x,y)\, r(x,y) \text{ dengan} \\ 0 < i(x,y) < \infty \\ 0 < r(x,y) < 1 \end{array}\right\}$$

Nilai *i(x,y)* ditentukan oleh sumber cahaya, sedangkan *r(x,y)* ditentukan oleh karakteristik objek di dalam gambar. Nilai *r(x,y)* = 0 mengindikasikan penyerapan total, sedangkan *r(x,y)* = 1 menyatakan pemantulan total. Jika permukaan mempunyai derajat pemantulan nol, maka fungsi intensitas cahaya, *f(x,y)*, juga nol. Sebaliknya, jika permukaan mempunyai derajat pemantulan 1, maka fungsi intensitas cahaya sama dengan iluminasi yang diterima oleh permukaan tersebut.

Contoh-contoh nilai *i(x,y)* :

Pada hari cerah, matahari menghasilkan iluminasi *i(x,y)* sekitar 9000 *foot candles.* pada hari mendung (berawan), matahari menghasilkan iluminasi *i(x,y)* sekitar 1000 *foot candles.* Pada malam, sinar bulan menghasilkan iluminasi *i(x,y)* sekitar 0.01 *foot candles.*

Contoh nilai *r(x,y)* :

benda hitam mempunyai *r(x,y)* = 0.01

dinding putih mempunyai *r(x,y)* = 0.8

benda logam dari *stainlessteel* mempunyai *r(x,y)* = 0.65.

Persamaan diatas menandakan bahwa nilai kerefleksian dibatasi oleh nilai 0 (*total absorbtion*) dan nilai 1 (*total reflectance*). Fungsi *i*(*x, y*) yang sudah didiskritkan baik koordinat spasial maupun tingkat kecerahannya. Kata kontinue disini dijelaskan bahwa indeks *x* dan *y* bernilai bilangan bulat. Kita dapat menganggap citra digital sebagai matriks dengan ukuran M x N yang baris dan

kolomnya menunjukkan titik-titiknya, yang diperlihatkan pada persamaan 2.2 berikut :

$$x = f(x,y) = \begin{pmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \dots & \dots & \dots & \dots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{pmatrix} \quad \text{.....(2.2)}$$

Intensitas dari gambar hitam-putih pada titik (*x,y*) disebut derajat keabuan (*greylevel*), yang didalam hal ini derajat keabuannya bergerak dari hitam ke putih, sedangkan citranya disebut citra hitam-putih (*greyscale image*) atau citra monokrom (*monocrhome image*).

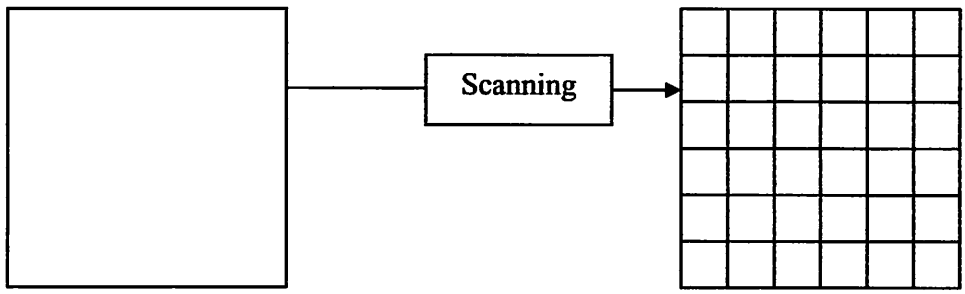
Citra yang dimiliki 16 derajat keabuan (mulai dari 0 mewakili warna hitam sampai 15 yang mewakili warna putih) dipresentasikan oleh 4 bit data. Sedangkan citra dengan 256 derajat keabuan (nilai 0 mewakili warna hitam sampai dengan 256 mewakili warna putih) dipresentasikan oleh 8 bit data.

Citra hitam-putih disebut juga citra satu kanal, karena warnanya hanya ditentukan oleh satu fungsi intensitas saja. Citra berwarna (*colorimages*) dikenal dengan nama citra spektral, karena warna pada citra disusun oleh tiga komponen warna yang disebut komponen RGB, yaitu: merah (*red*), hijau (*green*), dan biru (*blue*). Intensitas suatu titik pada citra berwarna merupakan kombinasi dari tiga intensitas: derajat keabuan merah (*fmerah* (*x, y*)), hijau (*fhijau* (*x,y*)), dan biru (*fbiru* (*x, y*)). Dalam citra berwarna, jumlah warna bisa beragam mulai dari 16, 256, 65536 atau 16 juta warna yang masing-masing dipresentasikan oleh 4, 8, 16 atau 24 bit data untuk setiap pixelnya.

Proses digitalisasi citra ada dua macam :

1. Sampling

Citra kontinue discanning pada grid-grid yang berbentuk bujursangkar (kisi-kisi dalam arah horisontal dan vertikal)

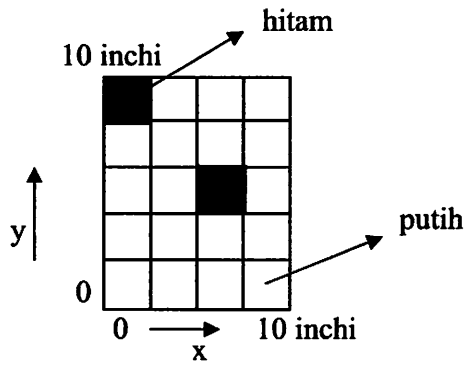


Gambar 2-4. Scanning secara spasial

Terdapat perbedaan antara koordinat gambar (yang discan) dengan koordinat matriks (hasil digitalisasi). Titik asal (0,0) pada gambar dan elemen (0,0) pada matriks tidak sama. Koordinat x dan y pada gambar dimulai dari sudut kiri bawah, sedangkan penomoran *pixel* pada matriks dimulai dari sudut kiri atas.

Element (*i,j*) di dalam matriks menyatakan rata-rata intensitas cahaya pada area citra yang direpresentasikan oleh *pixel*. Sebagai contoh, tinjau citra biner yang hanya mempunyai 2 derajat keabuan, 0 (hitam) dan 1 (putih). Sebuah gambar yang berukuran 10 x 10 inchi dinyatakan dalam matriks yang berukuran 5 x 4, yaitu lima baris dan 4 kolom. Tiap elemen gambar lebarnya 2.5 inchi dan tingginya 2 inchi akan diisi dengan sebuah nilai bergantung pada rata-rata intensitas cahaya pada area tersebut (Gambar 2-5).

Area 2.5 x 2.0 inchi pada sudut kiri atas gambar dinyatakan dengan lokasi (0,0) pada matriks 5 x 4 yang mengandung nilai 1 (yang berarti iluminasi maksimum).



$$\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

(a) (b)

Gambar 2-5. (a) Gambar yang discanning,

(b) matriks yang merepresentasikan gambar

Untuk memudahkan implementasi, jumlah sampling biasanya diasumsikan perpangkatan dari dua,

$N = 2^n$ ..... (2-3)

yang dalam hal ini,

N = jumlah sampling pada suatu baris/kolom

n = bilangan bulat positif

contoh ukuran sampling : 256 x 256 *pixel,* 128 x 256 *pixel*

Pembagian gambar menjadi ukuran tertentu menentukan resolusi yang diperoleh. Semakin tinggi resolusinya, yang berarti semakin kecil ukuran *pixel* (atau semakin banyak jumlah *pixel*-nya), semakin halus gambar yang diperoleh karena informasi yang hilang akibat pengelompokan derajat keabuan pada sampling semakin kecil.

2. Kuantisasi

Langkah selanjutnya setelah proses sampling adalah kuantisasi. Proses kuantisasi membagi skala keabuan (0,*L*) menjadi *G* buah level yang dinyatakan dengan suatu harga bilangan bulat (*integer*), biasanya *G* diambil perpangkatan dari 2,

$G = 2^m$ .....(2-4)

yang dalam hal ini,

G = derajat keabuan

m = bilangan bulat positif

Hitam dinyatakan dengan nilai derajat keabuan terendah, yaitu 0, sedangkan putih dinyatakan dengan nilai derajat keabuan tertinggi, misalnya 15 untuk 16 level. Jumlah bit yang dibutuhkan untuk merepresentasikan nilai keabuan *pixel* disebut kedalaman pixel *(pixel depth)*. Citra sering diasosiasikan dengan kedalaman *pixel*-nya. Jadi, citra dengan kedalaman 8 bit disebut juga 8-bit (atau citra 256 warna).

Pada kebanyakan aplikasi, citra hitam-putih dikuantisasi pada 256 level dan membutuhkan 1 *byte* (8 bit) untuk representasi setiap *pixel*-nya (G = 256 = 2^8).

Citra biner hanya dikuantisasi pada dua level: 0 dan 1. tiap *pixel* pada citra biner cukup direpresentasikan dengan 1 bit, yang mana bit 0 berarti hitam dan bit 1 berarti putih.

Besarnya daerah derajat keabuan yang digunakan menentukan resolusi kecerahan dari gambar yang diperoleh. Sebagai contoh, jika digunakan 3 bit untuk menyimpan harga bilangan bulat, maka jumlah derajat keabuan yang diperoleh hanya 8, jika digunakan 4 bit, maka derajat keabuan yang diperoleh adalah 16

buah. Semakin banyak jumlah derajat keabuan (berarti jumlah bit kuantisasinya makin banyak), semakin bagus gambar yang diperoleh karena kemenerusan derajat keabuan akan semakin tinggi sehingga mendekati citra aslinya.

Penyimpanan citra digital yang disampling menjadi N x M buah *pixel* dan dikuantisasi menjadi $G = 2^m$ level derajat keabuan membutuhkan memori sebanyak

$$b = N \times M \times m \quad \text{.....(2-5)}$$

Sebagai contoh, menyimpan citra yang berukuran dengan 512 x 512 *pixel* dengan 256 derajat keabuan membutuhkan memori sebesar 512 x 512 x 8 bit = 2048.000 bit.

Secara keseluruhan, resolusi gambar ditentukan oleh N dan m. Makin tinggi nilai N (atau M) dan m, maka citra yang dihasilkan semakin bagus kualitasnya (mendekati citra menerus).

Seluruh tahapan proses digitalisasi (sampling dan kuantisasi) dikenal sebagai konversi analog-digital, yang biasanya menyimpan hasil proses didalam media penyimpanan.

[3]Citra mengandung sejumlah elemen-elemen dasar. Elemen-elemen dasar tersebut dimanipulasi dalam pengolahan citra dan dieksploitasi lebih lanjut dalam komputer digital. Elemen-elemen dasar yang penting diantaranya adalah:

1. Kecerahan (*brightness*)

   Kecerahan adalah intensitas cahaya. Kecerahan pada suatu titik (*pixel*) di dalam citra bukanlah intensitas yang riil, tetapi sebenarnya adalah intensitas rata-rata dari suatu area yang melingkupinya.

2. Kontras (*contrast*)

Kontras menyatakan sebaran terang (*lightness*) dan gelap (*darkness*) di dalam suatu citra. Pada citra dengan kontras yang baik, komposisi gelap dan terang tersebar secara merata.

3. Kontur (*contour*)

Kontur adalah keadaan yang ditimbulkan oleh perubahan intensitas pada pixel-pixel yang bertetangga.

4. Warna (*color*)

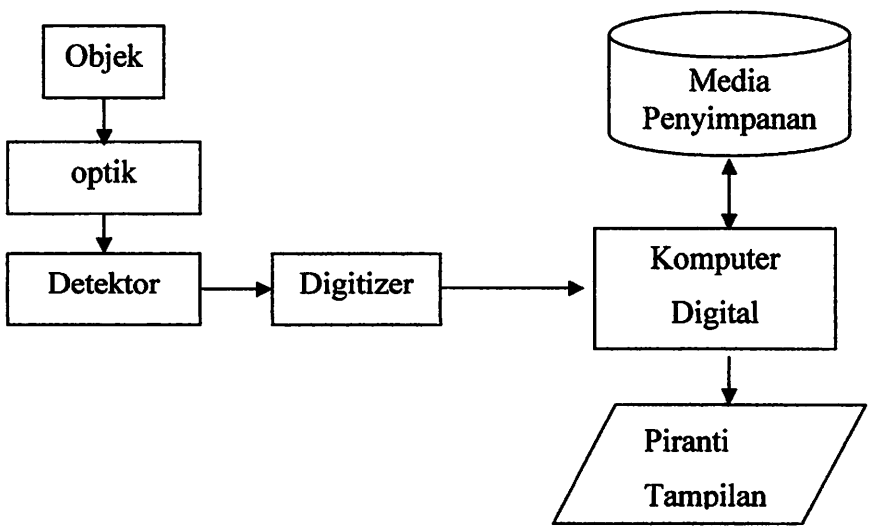
Warna adalah persepsi yang dirasakan oleh sistem visual manusia terhadap panjang gelombang cahaya yang dipantulkan oleh objek.

5. Bentuk (*shape*)

Shape adalah properti intrinsik utama untuk sistem visual manusia.

6. Tekstur (*texture*)

Tekstur dicirikan sebagai distribusi spasial dari derajad keabuan di dalam sekumpulan pixel-pixel bertetangga.



Gambar 2-6. Elemen pemproses citra

Operasi dari sistem pemprosesan citra tersebut dapat dibagi menjadi empat kategori prinsip: digitalisasi, pemprosesan, penayangan, dan penyimpanan.

Citra adalah sumber cahaya yang ditangkap oleh kamera. Optik berfungsi untuk memfokusakan kamera pada objek yang ditangkap, optik pada kamera seperti lensa.

Detektor berfungsi untuk mengetahui ada tidaknya cahaya dari objek yang masuk kekamera.

Digitizer (atau *digital image acquisition system*) merupakan sistem penangkap citra digital yang melakukan penjelajahan citra dan mengkonversinya kerepresentasi numerik sebagai masukan bagi komputer digital. Hasil dari digitizer adalah matriks yang elemen-elemennya menyatakan nilai intensitas cahaya pada suati titik. Contoh *digitizer* adalah kamera digital, webcam.

Digitizer terdiri dari tiga komponen dasar: sensor citra yang bekerja sebagai pengukur intensitas cahaya, perangkat penjelajah yang berfungsi merekam hasil pengukuran intensetas pada seluruh bagian citra, dan pengubah analog-ke-digital yang berfungsi melakukan sampling dan kuantisasi.

Komputer digital yang digunakan pada system pemproses citra dapat bervariasi dari komputer mikro sampai komputer besar yang mampu melakukan bermacam-macam fungsi pada citra digital resolusi tinggi.

Piranti tampilan peraga berfungsi mengkonversi matriks intensitas yang merepresentasikan citra ke tampilan yang dapat diinterpretasi oleh mata manusia. Contoh piranti tampilan adalah monitor peraga.

Media penyimpanan adalah piranti yang mempunyai kapasitas memori besar sehingga gambar dapat disimpan secara permanent agar dapat diproses lagi pada waktu yang lain.

Masing-masing R, G, dan B didiskritkan dalam skala 256, sehingga RGB akan memiliki indeks antara 0 sampai 255. Penjumlahan nilai RGB tidak bulat, sehingga kita harus membulatkannya nilainya, yaitu dengan metode bulat keatas atau kebawah. Misalkan dua warna yaitu hitam (0, 0, 0) kita jumlahkan dengan warna putih (255, 255, 255) akan menghasilkan warna abu-abu antara warna hitam dan putih (127, 127, 127). Hasil akhir dibulatkan kebawah. Contoh lain misalnya ingin menjumlahkan warna merah (255, 0, 0) dengan warna hijau (0, 255, 0) maka akan menghasilkan warna kuning (127, 127, 0).

Selain RGB, warna juga dapat dimodelkan berdasarkan atribut warnanya. Setiap warna memiliki tiga atribut, yaitu:

a. *Intensity / Brightness / Luminance*

Atribut yang menyatakan banyaknya cahaya yang diterima oleh mata tanpa mempedulikan warna. Kisaran nilainya adalah antara gelap (hitem) dan terang (putih).

b. *Hue*

Menyatakan warna sebenarnya, seperti merah, violet, dan kuning. *Hue* digunakan untuk membedakan warna-warna dan menentukan kemerahan (*redness*), kehijauan (*greeness*), dan kebiruan (*blueness*), dan sebagainya. *Hue* berisolasi dengan panjang gelombang cahaya.

c. *Saturation*

Menyatakan tingkat kemurnian warna cahaya, yaitu mengindikasikan seberapa banyak warna putih diberikan pada warna.

2.4. Segmentasi

Segmentasi merupakan proses untuk memisahkan atau membedakan antara objek dengan backgroud citra. Proses segmentasi bertujuan mengelompokkan pixel-pixel objek menjadi wilayah (*region*) yang mempresentasikan objek.

Ada dua pendekatan yang digunakan untuk segmentasi objek:

1. Segmentasi berdasarkan batas wilayah (tepi dari objek)

   Pixel-pixel tepi ditelusuri sehingga rangkaian pixel yang menjadi batas (*boundary*) antara objek dengan latar belakang dapat diketahui secara keseluruhan.

2. Segmentasi (misalnya segmentasi huruf menjadi garis-garis vertikal dan horisontal, segmentasi objek menjadi lingkaran, elips, dan sebagainya)

Warna untuk pixel baik objek maupun background dapat ditentukan berdasarkan kebutuhan. Dapat berupa warna hitam untuk pixel objek dan warna putih untuk pixel backgroung, atapun sebaliknya.

2.5. Grayscale

Proses awal yang banyak dilakukan dalam *image processing* adalah mengubah citra berwarna menjadi citra *gray-scale*, hal ini digunakan untuk menyederhanakan model citra. Pada awalnya citra terdiri dari 3 layer matriks yaitu R-layer, G-layer dan B-layer. Sehingga untuk melakukan proses-proses selanjutnya tetap diperhatikan tiga layer di atas. Bila setiap proses perhitungan dilakukan menggunakan tiga layer, berarti dilakukan tiga perhitungan yang sama. Sehingga konsep itu diubah dengan mengubah 3 layer di atas menjadi 1 layer

matriks grayscale dan hasilnya adalah citra gray-scale. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan.

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing R, G dan B menjadi citra grayscale dengan nilai s, maka konversi dapat dilakukan dengan mengambil rata-rata dari nilai R, G dan B sehingga dapat dituliskan menjadi:

$$s = \frac{R + G + B}{3} \qquad \text{..... (2-5)}$$

2.6. Thresholding

Tujuan dari thresholding adalah untuk mengatur jumlah derajat keabuan yang ada pada citra. Dengan menggunakan thresholding maka derajat keabuan bisa diubah sesuai keinginan, misalkan suatu citra mempunyai perbedaan yang sangat tinggi dan obyek yang diamati dalam citra tersebut adalah berwarna dengan *background* yang berwarna putih kemudian konstanta threshold yang dipilih adalah 192 dalam sekala 0 sampai 255.

2.7. JAVA

Sejarah **java** dimulai pada tahun 1991 ketika perusahaan **Sun Microsystem** memulai **Green Project**, yaitu sebuah proyek penelitian untuk membuat bahasa yang akan digunakan pada chip-chip embeded untuk device **intelligent customer electronic**,dalam penelitiannya Green Project berhasil membuat suatu prototype semanacam PDA yang dapat berkomunikasi antara yang satu dengan yang lainnya, ide berawal saat membuat sistem operasinya yang berbasis **C** dan **C++**, setelah berjalan beberapa lama seorang insinyur di Sun Microsystem ini tidak puas dengan beberapa karakteristik kompiler C++ yang di gunakan karena dinilai banyak bug, berbiaya besar dan sangat bergantung pada platform, karena merasa kurang puas dengan C dan C++ maka di kembangkanlah sebuah bahasa yang memberikan solusi terhadap kelemahan C++, bahasa itu dinamakan **OAK,** dimana kompiler baru ini mirip dengan bahasa C, pada tahun 1994 oak berubah menjadi JAVA, pada era ini java di divisikan sebagai bahasa yang memiliki dukungan di berbagai bidang Technology.

Java adalah bahasa pemprograman berorientasi objek. Java mengunakan kelas untuk membentuk suatu. Sejumlah kelas sudah tersedia yang dapat digunakan dengan mudah dan dikembangkan lebih jauh melalui konsep pewarisan. Java juga mendukung sumberdaya Internet yaitu *World Wide Web.* Java juga mendukung aplikasi klien/server, baik dalam jaringan local (LAN) maupun jaringan berskala luas (WAN).

Program Java tidak tergantung pada platform; Artinya, Java dapat dijalankan pada sembarang computer dan bahkan pada sembarang system operasi yang sering dinyatakan dengan istilah portabilitas. Tingkat portabilitas JAVA

tidak hanya sebatas pada program sumber *(source code)*, melainkan juga pada tingkat kode biner yang disebut *bytecode*.

Bytecode berbeda dengan kode mesin. Kode mesin sangat tergantung pada platform, sedangkan *bytecode* dapat dimengerti oleh semua platform yang telah dilengkapi dengan interpreter Java.

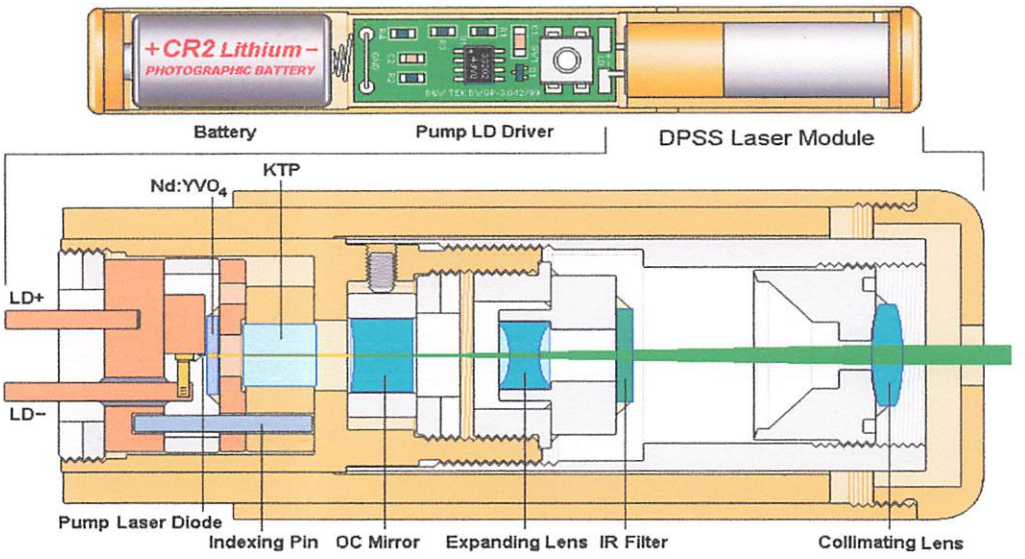
2.8. LASER

MASER (*microwave amplification stimulated emission radiations*) ditemukan pada tahun 1954 pada tahun 1958 ditemukan LASER (*light amplification stimulated emission radiations*). Sumber laser pertama kali dibuat menggunakan kristal batu merah delima.

Batu merah delima sebagian besar terdiri dari aluminium oksida yang beberapa atom alumuniumnya diganti oleh atom-atom kromium. Batu merah delima tersebut dibentuk batangan dengan diameter 0,5 cm sepanjang 4 cm pada salah satu ujungnya dibuat benar-benar rata dan dilapisi dengan perak. Batangan tersebut dipasang dalam sebuah *helically coiled electronic flash tube.*

Ketika suplai diaktifkan, muatan kapasitor bertambah sehingga tegangan naik. Kemudian tombol pembakaran ditekan sehingga terjadi lompatan elektron pada tabung yang menyebabkan lompatan kilatan cahaya dengan intensitas tinggi pada periode yang sangat pendek. Proses tersebut dinamakan pemompaan yang menghasilkan pulsa cahaya. Pulsa cahaya yang tinggi diperoleh oleh atom kromium. Konstruksi sumber laser seperti ditunjukkan pada gambar 5.9. Berkas laser adalah monokromatik dan koheren, dapat dihasilkan dengan diameter 0,5 hingga 3 mm. Jenis laser Argon menghasilkan cahaya dengan panjang gelombang

334 hingga 529 nm dengan pulsa kuat pada 488 nm (biru) dan 514 nm (hijau). Untuk menghasilkan pulsa (garis gelombang) tunggal daya yang diperlukan berkisar antara 50 sampai 100 mW.



Gambar 2-7. Laser Pointer

Laser diklasifikasikan menjadi :

1. Class 1

   Laser jenis ini tidak berbahaya, jika ditatap terus-menerus. Banyak digunakan pada teknologi pencetak, seperti pada printer laser. Laser jenis ini biasanya tidak kasat mata.

2. Class 1M

   Laser jenis ini tidak berbahaya jika harus ditatap terus-menerus dengan mata terlanjang. Akan tetapi menjadi berbahaya, jika ditangkap dengan bantuan optik (untuk memperkuat radiasi laser tersebut). Laser jenis ini biasanya tidak kasat mata. Laser-laser dengan jenis diatas 1M biasanya diberi label khusus.

3. Class 2

   Laser jenis ini kasat mata dengan panjang gelombang 400 – 700 nm. Tidak berbahaya apabila tidak ditatap langsung. Sebaliknya sangat berbahaya apabila ditatap langsung terus-menerus, sama halnya seperti menatap lampu senter secara terus-menerus. Biasanya laser pointer menggunakan laser jenis ini.

4. Class 2M

   Laser jenis ini juga kasat mata dena memiliki panajang gelombang 400 – 700 nm. Tidak berbahaya apabila tidak ditatap langsung. Sebaliknya, berbahaya apabila ditatap langsung terus-menerus, apabila dengan bantuan optik.

5. Class 3R

   Memiliki panjang gelombang 302,5 nm – 1 mm. sangat berbahaya bila ditatap langsung. Penggunaan laser jenis ini dilakukan dengan prosedur keamanan yang ketat.

6. Class 3B

   Laser jenis ini dapat mengakibatkan kerusakan pada mata atau kulit bila ditatap langsung.

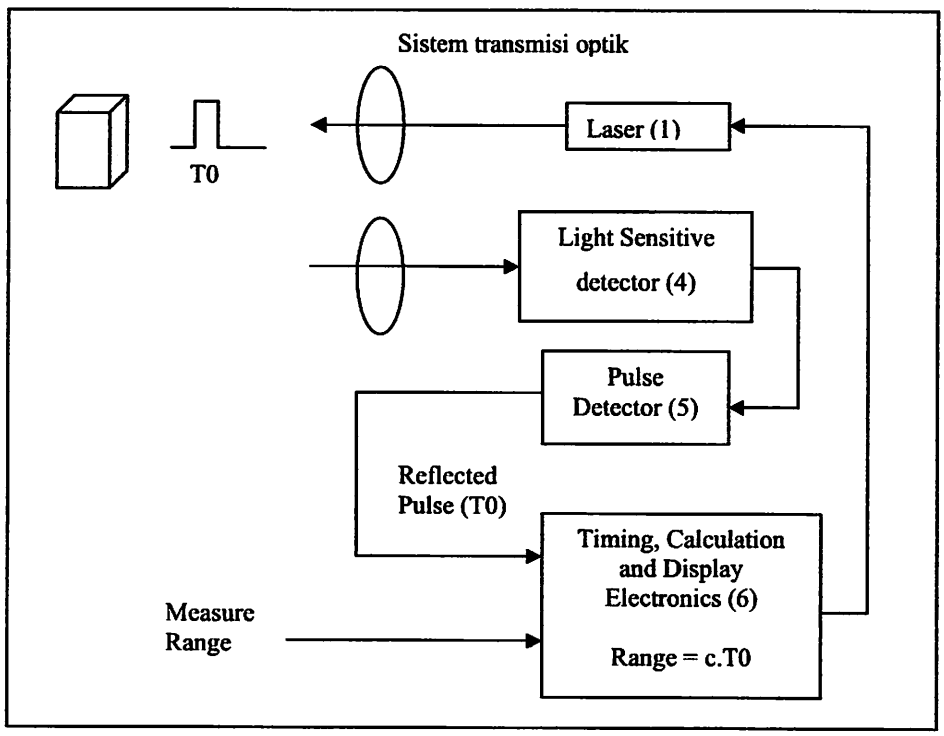
7. Class 4

   Laser jenis ini dapat mengakibatkan luka bakar bagi kulit apalagi mata. Penggunaan laser jnis ini harus menerapkan prosedur standart keamanan yang sangat baik.

2.9. TEORI DASAR TEKNOLOGI *LASER RANGE FINDER*

Teori dasar teknologi *laser range finder*, seperti ditunjukkn pada diagram Gambar 2-8 seperti berikut :

1. Sistem membangkitkan pulsa laser pada waktu **T0** yang ditembakkan pada objek yang akan diukur jaraknya.
2. Pemantulan sinar laser yang mengenai objek tersebut akan ditangkap oleh detector peka cahaya (4) pada waktu T0 dan akan diteruskan ke modul ***Pulse Detector*** **(5)**
3. Pulse Detector (5) berfungsi memilah-milah sumber cahaya untuk mendapatkan pulsa laser yang dikirim tadi.
4. Waktu **T0** dapat diperhitungkan dan jarak dapat diperoleh dengan rumus: c * T0, dimana c adalah cepat rambat cahaya di atmosfir.



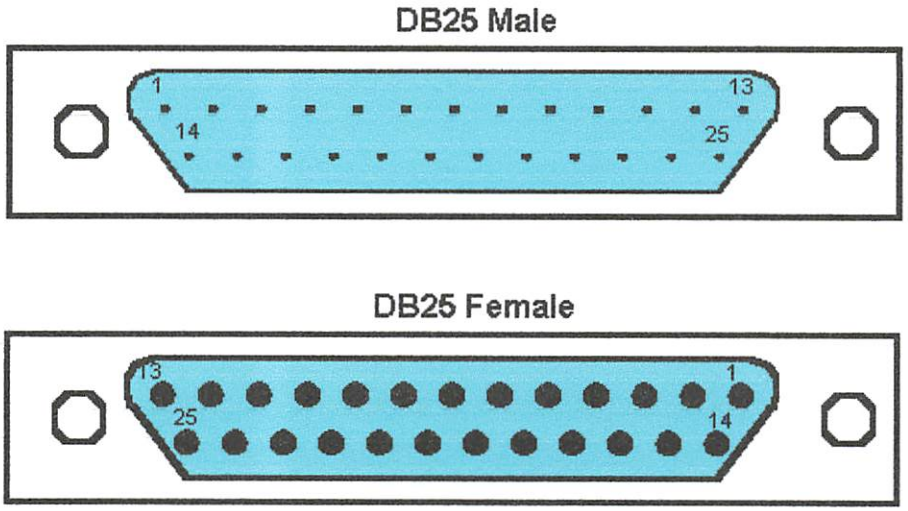
Gambar 2-8. Sistem Laser Range Finder

2.10. Parallel Port (LPT1)

Port parallel banyak digunakan dalam berbagai aplikasi antarmuka. Port parallel dapat dijumpai hampir pada semua komputer, baik itu dalam bentuk *desktop* PC maupun *laptop.* Pada kebanyakan komputer generasi yang terbaru port LPT1-nya sudah terintegrasi dengan mainboardnya. Port parallel ini memiliki tiga jenis alamat fisik, yakni Register Data, Register Status dan Register Kontrol. Pada mode standar port ini memiliki alamat yang khas yakni 378H, 379H dan 37AH. Port parallel ini terdiri dari 4 jalur kontrol (output), 5 jalur status (input) dan 8 jalur data (input/output).

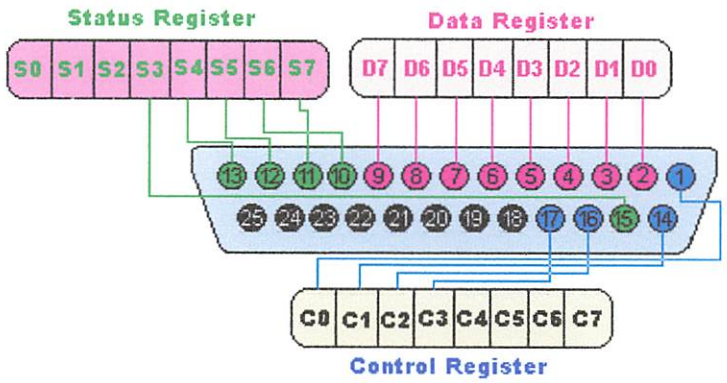
DB25 diperkenalkan oleh IBM PC pertama kali, yang bertujuan untuk menghemat tempat. Karena DB25 lebih praktis, maka untuk koneksi parallel pada komputer sekarang hanya menggunakan DB25, sedang centronic masih digunakan sebagai konektor pada printer (dan/ atau piranti luar lainnya).

Sebagai port komunikasi elektronika, maka parallel port dibuat dalam bentuk konektor jantan dan betina. Di komputer, konektor parallel port yang terpasang adalah DB25 *female* (betina) sehingga kabel penghubung luar adalah DB25 *male* (jantan). Susunan atau bentuk fisik dari DB25 betina dan jantan tersebut tampak seperti gambar 2-9.



Gambar 2-9. DB25 Male And Female

Dari 25 pin konektor DB25 tersebut, seperti yang telah dijelaskan sebelumnya bahwa hanya 17 pin yang digunakan sebagai jalur informasi dan sisanya sebagai ground. Konfigurasi dari masing-masing pin pada DB25 tersebut adalah seperti gambar 2-10 berikut:



Gambar 2-10. Konfigurasi Pin Port Parallel[2]

Port parallel yang baru, distandarisasi dengan standar IEEE.1284 yang dikeluarkan pada tahun 1984. Standar ini mendefinisikan 5 macam mode operasi sebagai berikut:

1. Mode Kompatibilitas
2. Mode Nibel

---

[2] http://www.logix4u.net/

3. Mode Byte
4. Mode EPP (*Enhanced Parallel Port*), mode yang digunakan dalam skripsi ini.
5. Mode ECP (*Extended Capabilities Port*)

Tujuan standarisasi ini untuk membantu merancang penggerak (*driver*) dan piranti yang baru yang kompatibel antara satu dengan lainnya serta kompatibel mundur (*backwards*) dengan SPP (*Standar Parallel Port*).

Dari lima macam mode operasi untuk LPT1 diatas, yang paling banyak dijumpai pada PC keluaran terbaru adalah mode EPP + ECP, sedangkan mode yang lain sudah diwakili dalam mode SPP (*standard parallel port*). Namun dalam mode EPP sendiri juga memuat alamat fisik dari mode SPP, sehingga pada setting bios dari PC umumnya disetting pada mode EPP atau EPP + ECP. EPP sebagai salah satu pilihan mode LPT1 banyak digunakan secara luas untuk peralatan yang sifatnya non-printer, seperti CD-ROM External, TapeDrive, Hard Drive dan lain-lain.

Sedangkan untuk pin-pin dari DB25, karena menggunakan mode EPP maka sangat perlu untuk mengerti definisi dari pin-pin DB25 dalam mode EPP. Pada saat menggunakan mode EPP, fungsi dan nama dari masing-masing jalur di definisikan ulang. Definisi dari pin-pin parallel port dapat dilihat pada tabel 2.2. berikut:

Tabel 2-2. Sinyal EPP dan SPP[4]

Pin	SPP Signal	EPP Signal	In/Out	Function
1	Strobe	Write	Out	Kondisi Low berarti proses Write, High berarti proses Read.
2-9	Data 0-7	Data 0-7	In-Out	Data Bus dua arah.
10	Acknowledge	Interrupt	In	Interrupt Line. Interrupt dijalankan pada saat menuju positif (Positif Rising Edge)
11	Busy	Wait	In	Untuk *handshaking*. Proses EPP dimulai bila kondisi Low dan selesai bila High.
12	Paper Out /End	Spare	In	Kosong. Tidak digunakan pada proses *handshake*
13	Select	Spare	In	Kosong. Tidak digunakan pada proses *handshake*
14	Auto Linefeed	Data Strobe	Out	Low berarti data di transfer
15	Error/ Fault	Spare	In	Kosong. Tidak digunakan pada proses *handshake*
16	Initialize	Reset	Out	Reset. Active low
17	Select Printer	Address Strobe	Out	Saat Low berarti Address di transfer.
18-25	Ground	Ground	Gnd	Ground

Untuk pin *paper out, select* dan *Error* dalam proses EPP *handshake* tidak didefinisikan. Jalur-jalur tersebut masih dapat dimanfaatkan oleh pemakai untuk keperluan yang lain.

[4] http://www.beyondlogic.org/epp/epp.htm

2.11. Komunikasi *Port* Paralel Menggunakan JAVA

Biasanya kita mengakses parallel port menggunakan c++, vcc, delphi dan VB sebagian dari mereka menggunakan inpout32.dll sebagai interpreter ke db25, namun pada java dengan menggunakan jnpout32.dll yang tidak lain adalah modifan dari inpout32.dll.

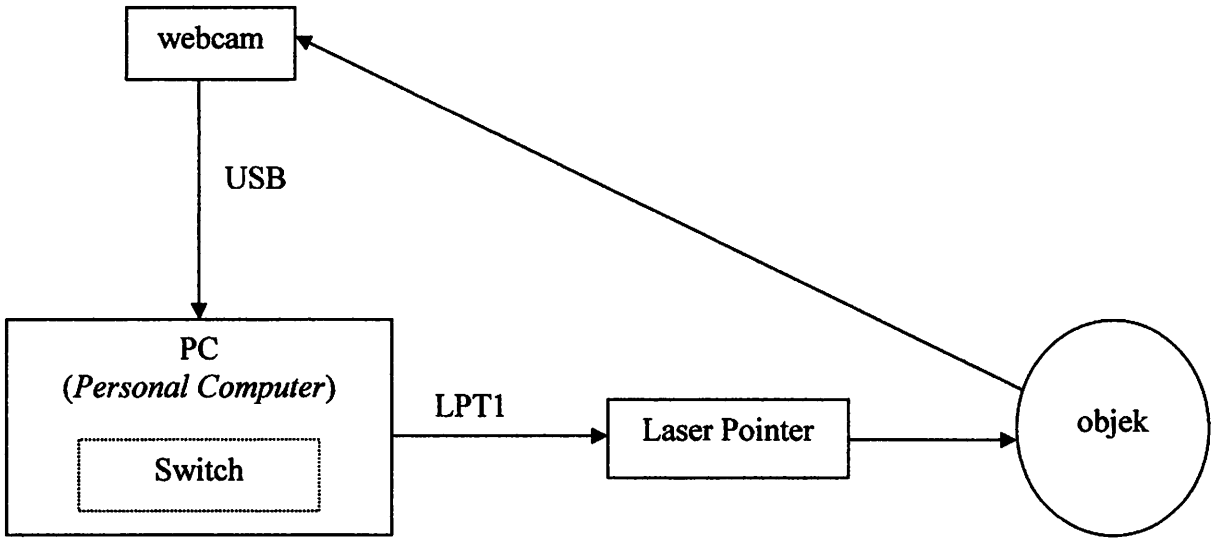
Biasanya banyak orang mengakses serial port atapun parallel port menggunakan java.comm dari java.sun tetapi konfigurasi dan setingan serta coddingnya bisa dibilang cukup menyulitkan, tapi jika menggunakan jnpout32.dll kita cuman mengkopikan saja file .*dll ini ke c:\windows\system32 atau c:\windows\system32\drivers\ lalu kita inisialisasi terlebih dahulu dan kita bisa langsung mengakses parallel port tersebut.

BAB III

PERENCANAAN DAN PEMBUATAN ALAT

Sesuai dengan fungsinya, teknologi LRF (*Laser Range Finder*) banyak digunakan untuk pengukuran jarak. Pada sistem ini, kami menggunakan analisa dan kalibrasi pixel pada gambar, dimana berkas laser jatuh pada objek untuk mendapatkan informasi jarak.

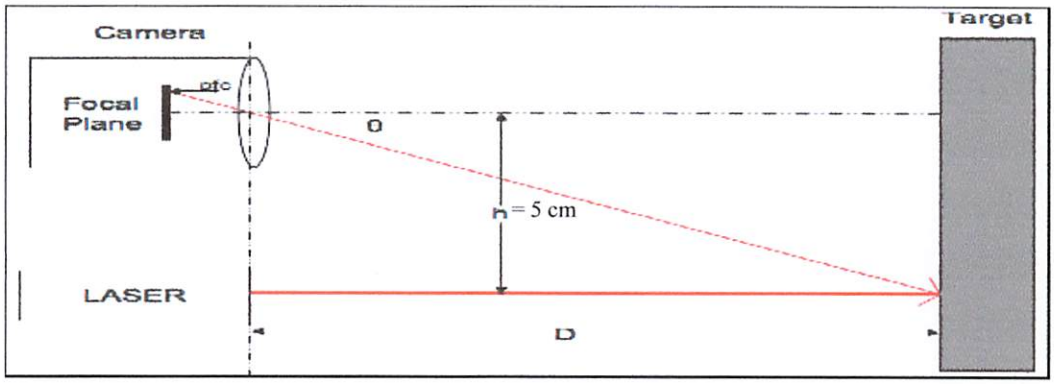
Komponen yang dibutuhkan terdiri atas perangkat hardware dan software yang saling berintegrasi. Secara garis besar perangkar hardware berfungsi sebagai alat pengindra (sensor), sedangkan perangkat software berfungsi mengolah citra yang didapat untuk menghasilkan informasi jarak.



Gambar 3-1. Blok Diagram Sistem

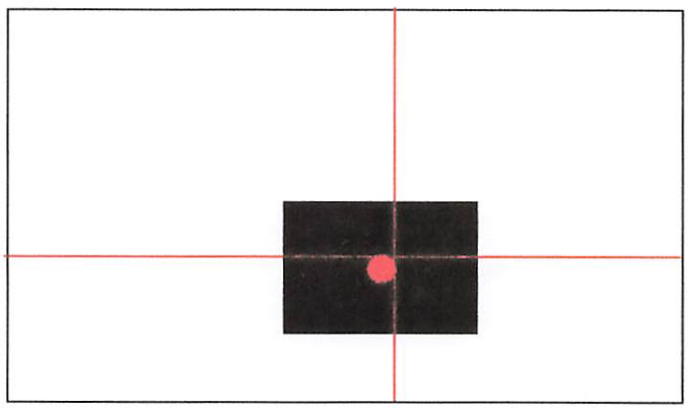
3.1 Cara Kerja

Berkas sinar laser yang diproyeksikan pada suatu target dapat ditangkap oleh kamera. Penempatan posisi sumber sinar diletakkan paralel dengan sumbu kamera dengan jarak 5 cm.



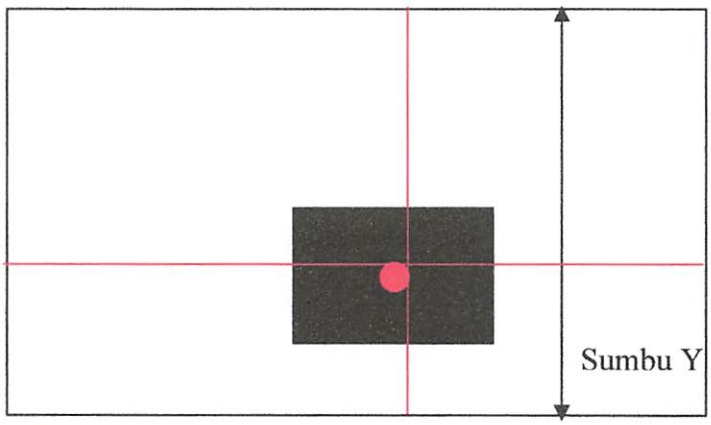
Gambar 3-2. Prinsip Kerja

Berkas sinar laser dalam gambar yang ditangkap oleh webcam akan diolah oleh algoritma program berdasarkan pixel yang paling terang, dengan asumsi proyeksi berkas sinar laser pada target menghasilkan area yang paling terang sehingga posisi berkas sinar pada gambar tersebut dapat diketahui.



Gambar 3-3. Berkas laser yang diproyeksikan pada objek yang menghasilkan area yang paling terang

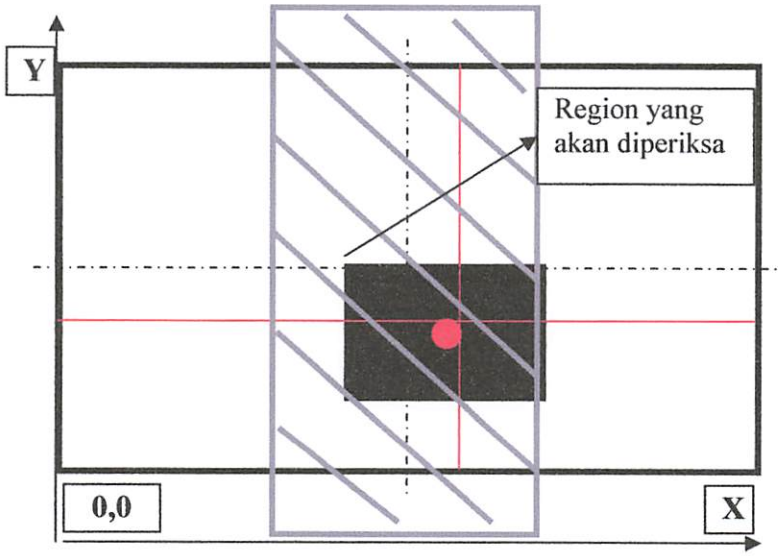
Perhitungan jarak objek berdasarkan pada lokasi jatuhnya berkas sinar laser tersebut pada sumbu **Y** pada gambar. Semakin dekat dengan pusat gambar, maka semakin jauh jarak objek tersebut.



Gambar 3-4. Perhitungan jarak target berdasarkan jatuhnya berkas laser pada sumbu Y

3.2. Koordinat Layar

Gambar 3-4 menunjukkan koordinat layar yang berguna untuk perhitungan lokasi.



Gambar 3-4. Koordinat layar

Koordinat (0,0) terletak pada pojok kiri bawah, sedangkan koordinat maksimum sesuai dengan resolusi maksimum sesuai dengan resolusi maksimum

yang digunakan dalam hal ini (320, 240) terletak dipojok kanan atas. Area/region yang diarsir merupakan area yang akan dimonitor oleh program yakni koordinat awal (80,0) dan koordinat akhir (160, 240).

Jika berkas laser jatuh pada area tersebut, maka program akan memperhitungkan informasi jarak yang diperoleh. Sedangkan jika berkas laser jatuh di luar area tersebut, maka program akan mengabaikannya.

3.3. Perencanaan dan Pembuatan Perangkat Keras

3.3.1. Perangkat Keras Webcam

Pada skripsi ini digunakan *webcam* produk dari *Genius* dengan type *VideoCAM Express*. Kamera ini menggunakan CMOS untuk sensornya. Ini merupakan teknologi paling canggih dari *Genius*. Desain kamera ini tidak memungkinkan untuk berputar 360 derajat, untuk pengambilan gambarnya (*snapshot*) dapat dilakukan dengan menekan bagian atas dari kamera. Adapun spesifikasi dari kamera tersebut adalah sebagai berikut :

- *Image pixel* yang terdapat didalam sensor bayangan CMOS 300,000 pixel
- Resolusinya adalah : 160x120, 176x144, 320x240, 352x288
- *Video preview* dan *recording* mencapai 30 fps
- Terdapat *button-switch* untuk pengambilan gambar (*snapshot*)
- Hasil *record sudah* berformat dalam *MPEG*
- Memungkinkan untuk berotasi 360 derajat
- Pengaturan fokus kamera terdapat pada sisi depan kamera, metode yang digunakan adalah *pan focus*
- Pengaturan keseimbangan warna (*white balance*) secara otomatis

3.3.2. Perangkat Keras Laser Elemen

Fungsi dari laser elemen adalah sebagai sumber cahaya yang digunakan untuk pengukuran jarak. Laser elemen yang digunakan berasal dari laser pointer yang dapat dicari di toko-toko terdekat. Laser pointer ini mengeluarkan berkas sinar laser dengan panjang gelombang 630-680 nm berbentuk titik berwarna merah. Jarak optimal yang dapat ditempuh berkisar 10 meter sehingga cukup baik untuk sistem yang akan kita buat nanti.

Tegangan yang masuk dari port paralel pada saat laser menyala adalah 2.19 Volt sedangkan arusnya sebesar 24.8 mA, sehingga dayanya 2.19 V x 24.8 mA = 54.312 mW.

3.3.3. Dudukan Laser Elemen dan Kamera

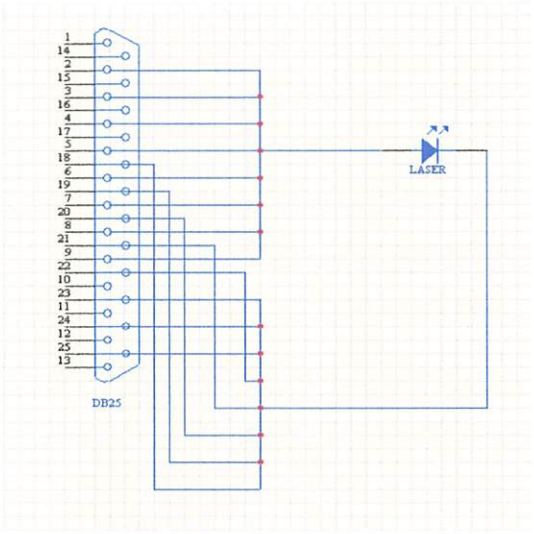
Fungsi dudukan adalah sebagai penyangga dudukan dari laser elemen dan kamera. Dudukan ini bersifat statis. Penulis menggunakan box kecil sebagai dudukan, yang telah diatur agar jarak antara pusat webcam dan laser pointer adalah 5 cm.

3.3.4. PC (Personal Computer)

PC yang digunakan adalah standart PC biasa dengan prosesor Pentium IV, memori minimal RAM 128 Mb, dan harddisk minimal 50 Mb yang berfungsi sebagai tempat pemproses data dan switch laser.

3.3.5. Perangkat Keras DB25

Port paralel atau DB25 digunakan untuk switch laser dari keadaan off ke on atau sebaliknya langsung dari PC, sehingga tidak lagi menggunakan power supply baterai dan penekanan tombol manual. Keluaran delapan pin yang diparalel adalah tegangan 3.34 Volt dan 16.8 mA .



Gambar 3-5. Skematik Rangkaian

3.4. Perencanaan dan Pembuatan Perangkat Lunak

3.4.1. Perangkat Lunak Untuk Menangkap Berkas Sinar Laser (Pada PC)

Untuk membuat aplikasi penghitung jarak antara objek dengan laser maka pada PC (*Personal Computer*) diperlukan *webcam* untuk mengirimkan data *pixel* yang diinginkan oleh program. Program yang akan dibuat menggunakan bahasa pemrograman Java dengan paket J2SE versi 1.4.2 dengan tambahan Java Media Framework (JMF). JMF ini berfungsi sebagai tatap muka program java dalam mengakses sumber peralatan audio dan vidio seperti webcam. Dimana paket java J2SE telah terinstal terlebih dahulu kedalam sistem.

Pada dasarnya program java disini mempunyai 2 tugas yaitu :

2. Membuka port paralel pada PC sekaligus melakukan inisialisasi port.
3. Menganalisa data *pixel* yang diterima untuk menghitung jarak.

Untuk dapat mendaftarkan *webcam* agar dapat dikenali dan digunakan oleh program java, oleh karena itu, hubungkan webcam pada slot USB. Kemudian, JMF *Registry Editor* akan mencoba mendeteksi webcam yang kita pasang tadi. Jika JMF *Registry Editor* berhasil mendeteksi webcam tersebut, maka pada *text-box Capture Devices* akan muncul daftar audio dan video yang berhasil

dideteksi. Kemudian setting beberapa parameter antara lain Encoding : RGB, *Video Size* : 320 x 240, *Frame Rate* : 30 fps, *Bits per Pixel* : 24 bit.

Untuk itu pertama kali yang kita lakukan adalah penginisialisasian *port* paralel yang akan kita gunakan. Diantaranya adalah membuka *port* paralel, menuliskan data ke *port* paralel dan kemudian mengirimkannya, membaca data pada *port paralel* dan menutup *port serial.* Listing program yang digunakan sebagai berikut :

- Membuka *paralel port*

  Pada java untuk membuka paralel *port,* yakni dengan mengkopikan file jnpout32.dll ke C:\WINDOWS\system32 maka bisa langsung mengakses paralel port tersebut. Kemudian ketikan listing program :

  public class ioPort{

  public native void Out32(int PortAddress, int data);

  public native short Inp32(int PortAddress);

  static { System.loadLibrary("jnpout32");}}

- Menutup *port paralel*

  Untuk menutup port palalel yakni dengan mengexit aplikasi yang digunakan untuk membuka port paralel tersebut.

Parameter **pfc**, yaitu pixel dari pusat *focal plane* dapat dihitung dari gambar yang diperoleh. Sedangkan parameter-parameter yang lain, yaitu **rpc** dan **ro** diperoleh dari hasil pengukuran dan kalibrasi.

Untuk melakukan kalibrasi diperlukan beberapa pengukuran, yaitu jarak yang diukur pada objek tertentu (**D**) yang telah diketahui nilainya. Demikian juga dengan nilai **pfc**. Untuk mendapatkan pfc berikut rumusannya :

$$D = \frac{h}{\tan(pfc * rpc + ro)} \qquad \text{.....(3-1)}$$

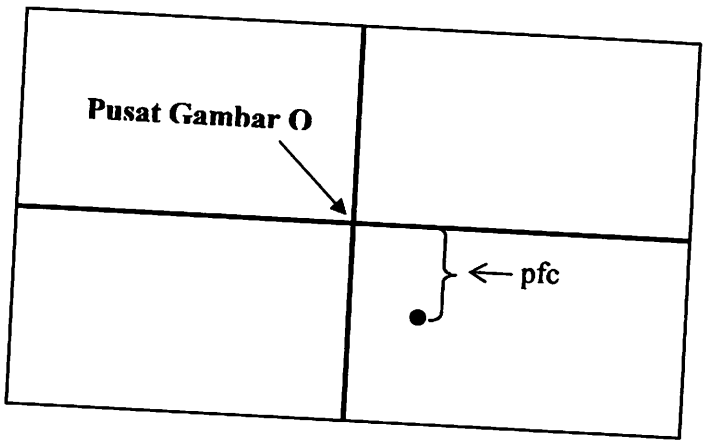
Dengan :

$$pfc = (Y_{max} / 2) - Y' \qquad \text{.....(3-2)}$$

dengan :

Y_{max} = jarak maksimum y

Y' = posisi berkas laser yang jatuh pada sumbu Y



Gambar 3-6. Perhitungan pfc dari pusat gambar oleh program

Tabel 3-1. Perhitungan pfc oleh program dan jarak aktualnya

pfc	D aktual (cm)
119	25
108	30
70	60.5
66	52

72	61
63	78
57	100
56	103

Kita juga dapat menghitung sudut aktual berdasarkan nilai dari **h** dan jarak aktual (**D aktual**) dengan persamaan (4) berikut ini :

$$\theta_{ACTUAL} = \arctan\left(\frac{h}{D_{actual}}\right) \quad \text{..... (3-3)}$$

dengan :

T_{actual} = sudut aktual (sudut sebenarnya)

D_{actual} = jarak aktual (yang telah diketahui) terhadap target

Dengan persamaan (2-2) dan (2-4) di atas dapat diperoleh **rpc** dan **ro.**

$$rpc = \frac{\Theta}{pfc} \quad \text{.....(3-4)}$$

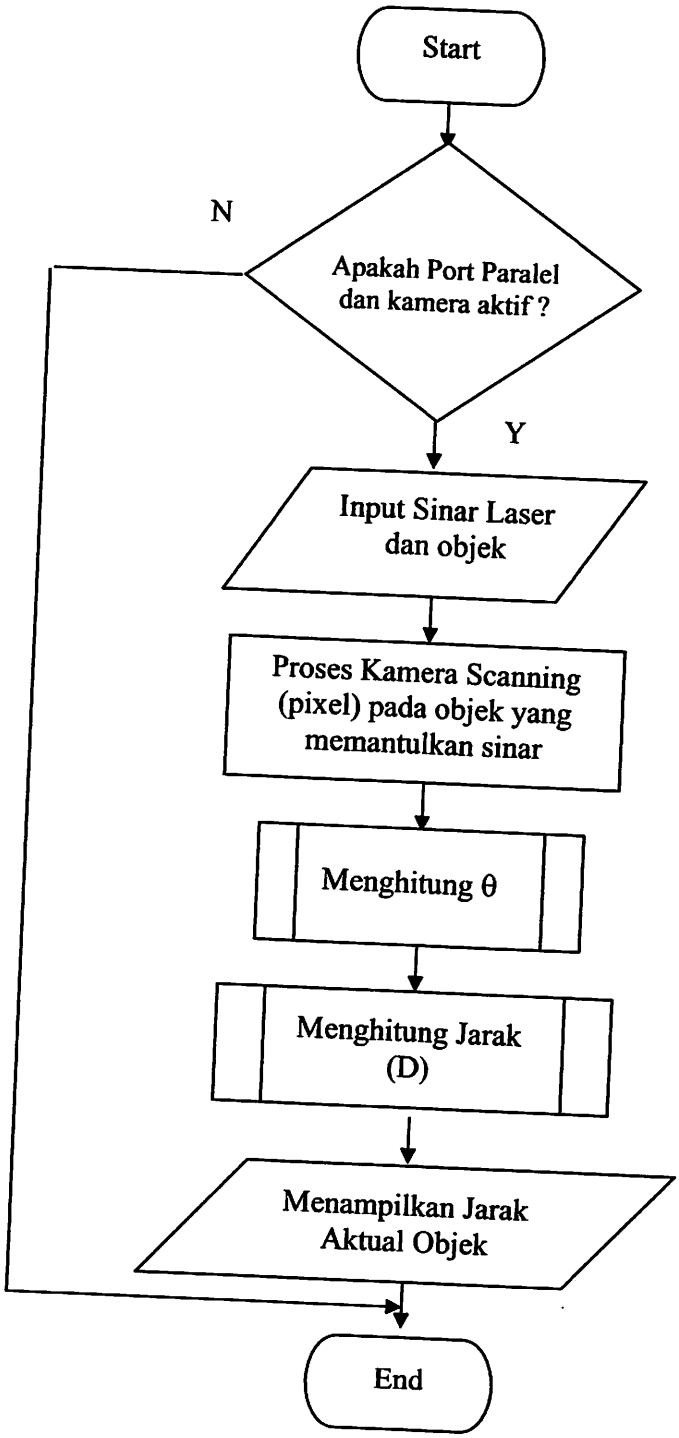
$$ro = \Theta - (pfc * rpc) \quad \text{.....(3-5)}$$

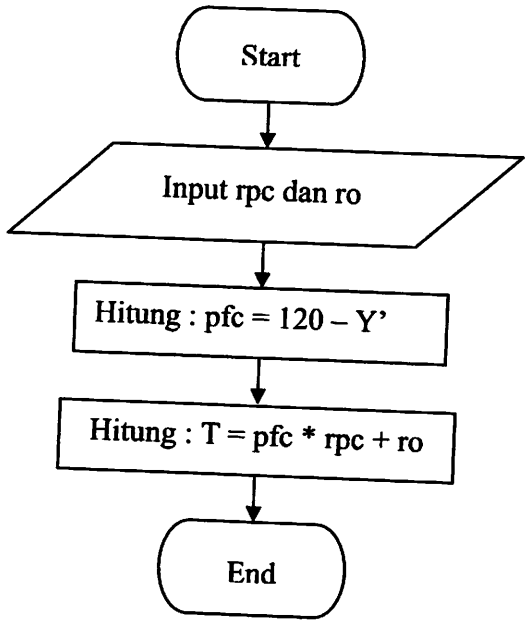
Untuk memudahkan perhitungan, penulis melakukan perhitungan dengan excel. Kita tinggal memasukan nilai parameter **h** yang bernilai konstan 5 cm, **pfc**, dan **D actual** yang diperlukan seperti ditunjukan pada tabel dibawah ini :

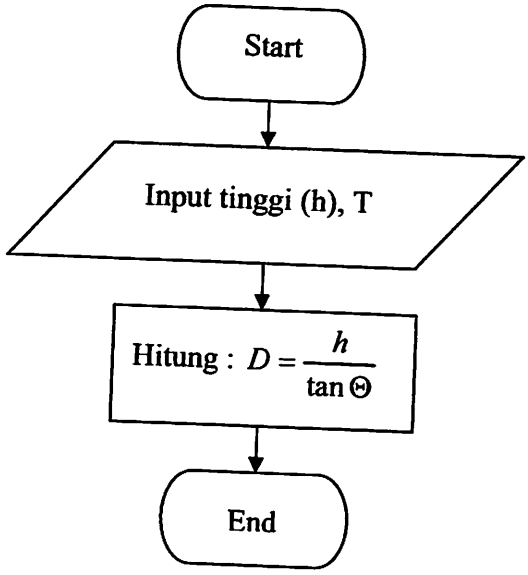
Tabel 3-2. Hasil Perhitungan Kalibrasi

h	pfc	D_actual (cm)	Theta_actual:arctan (h/D_actual) (degree)	pfc-(pfc_total/n_pfc)	Theta_actual - (Theta_actual_total/n_Theta_actual)	rpc	ro
5	119	25	0.19739556	42.625	0.099255061	0.002329	-0.0797
5	108	30	0.165148677	31.625	0.067008179	0.002119	-0.06369
5	70	60.5	0.082457237	-6.375	-0.015683261	0.00246	-0.08975
5	66	52	0.095859147	-10.375	-0.002281351	0.00022	0.081346
5	72	61	0.081784381	-4.375	-0.016356118	0.003739	-0.18739
5	63	78	0.064014978	-13.375	-0.034125521	0.002551	-0.09673
5	57	100	0.049958396	-19.375	-0.048182103	0.002487	-0.09179
5	56	103	0.048505612	-20.375	-0.049634886	0.002436	-0.08791
	611		0.785123988			0.002293	-0.07771

Dari tabel 3.2 diperoleh nilai **rpc** = 0.002293 dan **ro** = -0.07771. Kedua ini nantinya kita masukkan sebagai konstanta GAIN (**rpc**) dan OFFSET (**ro**) dalam program.


Start
N
Apakah Port Paralel dan kamera aktif ?
Y
Input Sinar Laser dan objek
Proses Kamera Scanning (pixel) pada objek yang memantulkan sinar
Menghitung θ
Menghitung Jarak (D)
Menampilkan Jarak Aktual Objek
End





Gambar 3-7 Flowchart program java

Listing program penjelasan flowchart:

1. mengaktifkan laser pointer melalui port paralel:

```
public static boolean set_bit() {
int value, status_port, new_value, address_port, number_bit;
   address_port=0x378;
   number_bit=0;
   value = 255;
   new_value = 0;
   status_port = parport.Inp32(address_port);
   new_value = status_port | value;
   parport.Out32(address_port, new_value);
   return true; }
```

2. Mengaktifkan kamera :

```
      if (s.equals("Connect")) {
              a = new ActionListener() {
                  public void actionPerformed(ActionEvent ae){
                      Vision.p.start();
       p.start();
       // Show the frame
       setVisible(true);
       String videoPropFile =
   System.getProperty("video.properties", video.properties");
```

3. Memuat suatu region yang berbentuk persegi panjang untuk mendapatkan lokasi pixel yang paling terang dan juga ditambahkan listener ke dalamnya. Region tersebut memiliki koordinat awal (80,0) dan koordinat akhir (160,240).

```
        Vision.addRectRegion(1, 80, 0, 160, 240);
        Vision.addColorListener(1, listener, 0);
```

4. Nilai rpc, ro dan h yang didapat dari hasil kalibrasi diberikan kepada konstanta-konstanta pada program. Nilai rpc diberikan pada konstanta GAIN dan ro diberikan

pada konstanta OFFSET, sedangkan nilai h diberikan pada konstanta H_CM. Threshold diberikan konstanta 192

```
public static final float GAIN = 0.002293f;
public static final float OFFSET = -0.07771f;
public static final float H_CM = 5f;
private static final int LIGHT_THRESHOLD    = 192;
```

5. Lakukan looping untuk setiap region dan listener yang dibuat untuk mendapatkan lokasi pixel yang paling terang berdasarkan komposisi nilai RGB (Red-Green-Blue)

```
for(int j=0;j<cl.length;j++) {

int pixCount = 0, totalPixs = 0;
 int xPos = 0, yPos =0;
int aR = 0, aG = 0, aB = 0;

for(int ii=ry; ii<ry+height; ii++) {
for(int jj=rx; jj<rx+width; jj++) {
int pos = ii*lineStrideIn + jj*pixStrideIn;

        int tr = inData[pos+2] & 0xFF;
        int tg = inData[pos+1] & 0xFF;
        int tb = inData[pos] & 0xFF;

if(Math.abs(tr) > Math.abs(maxtr) && Math.abs(tg) >
Math.abs(maxtg) && Math.abs(tb) > Math.abs(maxtb) ) {
                if (tr >= LIGHT_THRESHOLD && tg >=
LIGHT_THRESHOLD && tr >= LIGHT_THRESHOLD) {
                        maxtr =  tr;
                        maxtg =  tg;
                        maxtb =  tb;
                        yPos = ii; xPos = jj;  }} }
```

resolusi gambar yang digunakan adalah 320 x 240. Proses tersebut berulang hingga pixel pada region sudah selesai semua.

6. Variable yPos dan xPos menyimpan posisi berkas laser pada region. Jika nilai xPos dan yPos lebih besar dari 0, maka program akan menghitung informasi jarak, range berdasarkan nilai pfc, ro (OFFSET), dan rpc (GAIN). Informasi jarak tersebut kemudian akan ditampilkan pada layar.

```
if (xPos > 0 && yPos > 0) {
        int pfc = 120 - yPos;
   range = (float) (H_CM / Math.tan(pfc * GAIN + OFFSET));
        Font.println("jarak:" + range + "cm ,pfc:" + pfc +
 "\nx:" + xPos + ",y:" + yPos, Font.FONT_6x11, xPos, yPos,
 (byte) 255,(byte) 0,(byte) 0, outBuffer);
        } else
          range = 0;        }
```

7. Mematikan Laser pointer melalui port parallel :

```
public static boolean clear_bit(){
int value, status_port,
new_value,address_port,number_bit;
value = 0;
new_value = 0;
number_bit=0;
address_port=0x378;
status_port = parport.Inp32(address_port);
new_value = status_port & value;
parport.Out32(address_port, new_value);
return true;}
```

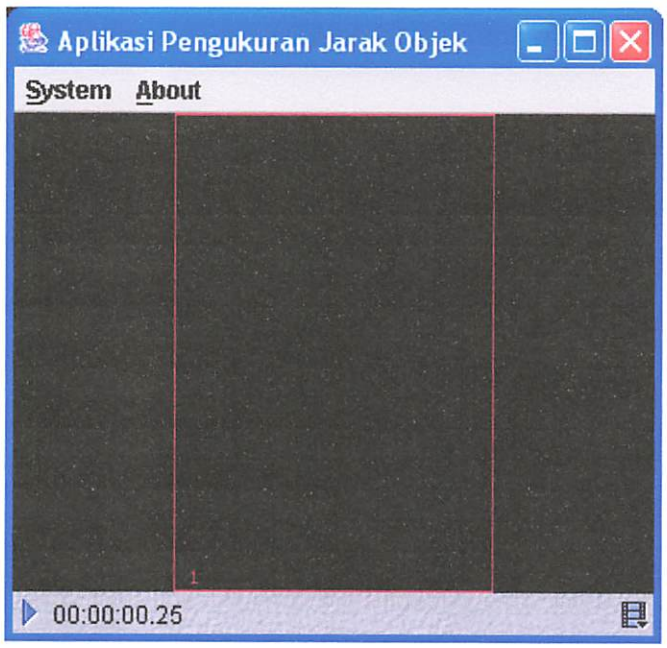
8. Mematikan kamera :

```
Vision.p.stop();
  public static void stopViewer() {
    visionFrame.setVisible(false);
    p.close();
  }
```

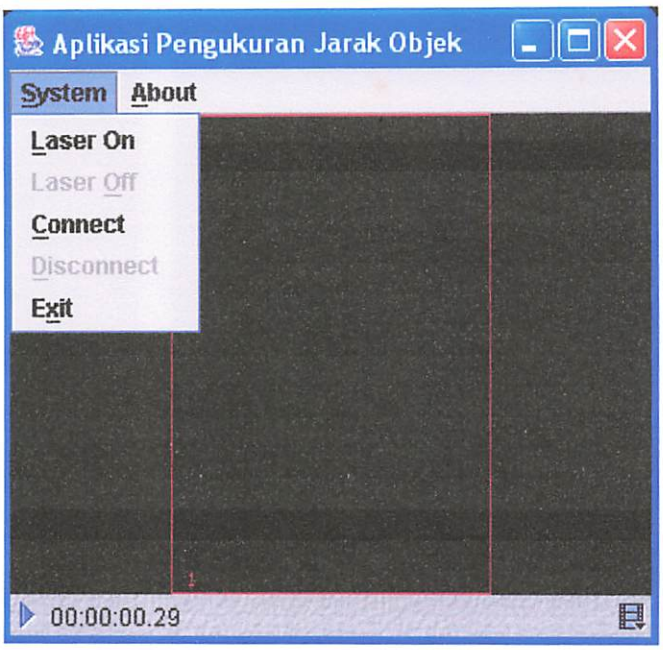
3.4.2. Pengenalan Menu Program Aplikasi Pengukuran Jarak Objek

Untuk menjalankan aplikasi, pastikan kita telah men-set classpath file java. Sebelum menjalankan aplikasi, pastikan pula webcam telah terpasang dengan benar pada komputer. Kemudian kita tinggal meng-klik dua kali file OnvineLPS.bat yang berisi script java OnvineLPS pada folder vision.

Jika webcam balum terpasang dengan benar, maka jika kita tidak bisa menjalankan program. Sedangkan bila berjalan lancar, maka program **OnvineLPS** akan muncul seperti ditunjukkan gambar 3.7



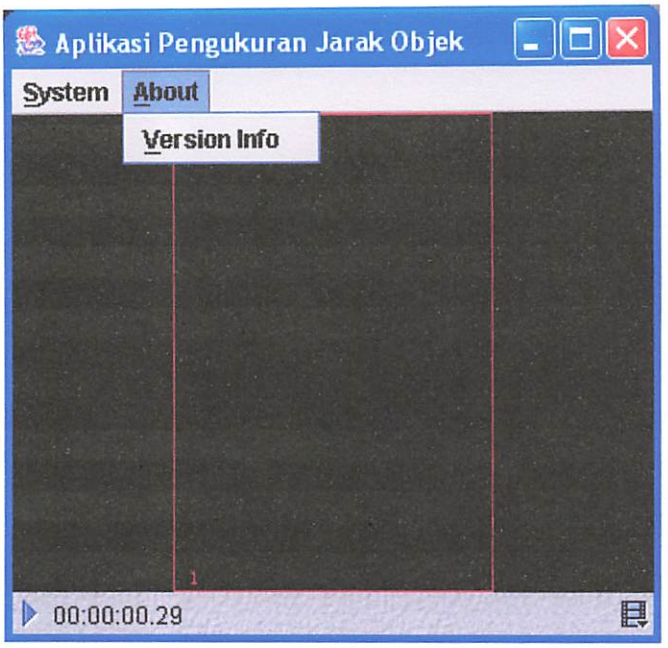
Gambar 3-8. Program Laser Parameter Sensor



Gambar 3-9. Menu Sistem

Program utama aplikasi yang digunakan untuk mengukur jarak dari laser ke objek pada komputer hanya terdiri atas dua menu utama, yaitu **System** dan **About.** Pada menu **System** terdapat submenu :

- *Laser On*, untuk menyalakan sinar laser pointer.
- *Laser Off*, untuk mematikan sinar laser pointer.
- *Connect*, berfungsi untuk melakukan scanner pada webcam berdasarkan pixel yang paling terang.
- *Disconnect*, berfungsi untuk memutuskan scanner pada webcam.
- *Exit*, untuk keluar dari program



Gambar 3-10. Menu About

Sedangkan pada menu **About** hanya terdapat submenu **Version Info** yang berisi informasi program OnvineLPS ini.

3.4.3 Menggunakan Program

Sebelum mekakukan scanning pada webcam, pastikan laser pointer telah dinyalakan dan proyeksi berkas laser cukup nyata untuk ditangkap oleh webcam. Pilih menu **System > Laser On** untuk menyalakan laser pointer, kemudian pilih menu **System > Connect** untuk melakukan scanning pada webcam. Tunggu beberapa saat, dimana program akan mengumpulkan statistik informasi jarak.

Gambar 3-11. Gambar yang ditangkap webcam serta informasi jarak

Jika semuanya berjalan dengan baik, maka program akan menampilkan gambar yang ditangkap oleh webcam, seperti pada gambar 3-10. Pada gambar terdapat satu region yang disertai dengan informasi jarak 44 dan pfc 75.

Untuk menghentikan aplikasi, pilih menu **System > Disconect**. Untuk keluar dari program pilih menu **System > Exit**.

3.5. PERANGKAT KERAS SISTEM PENGUKURAN JARAK

Perangkat keras untuk sistem pengukuran jarak tediri atas : *webcam*, port paralel sebagai *interface* dengan komputer, laser pointer sebagai sumber cahaya. Sistem yang telah dibuat terlihat pada gambar 3-9.

Gambar 3-12. Gambar Perangkat Keras yang sudah dibuat

BAB IV

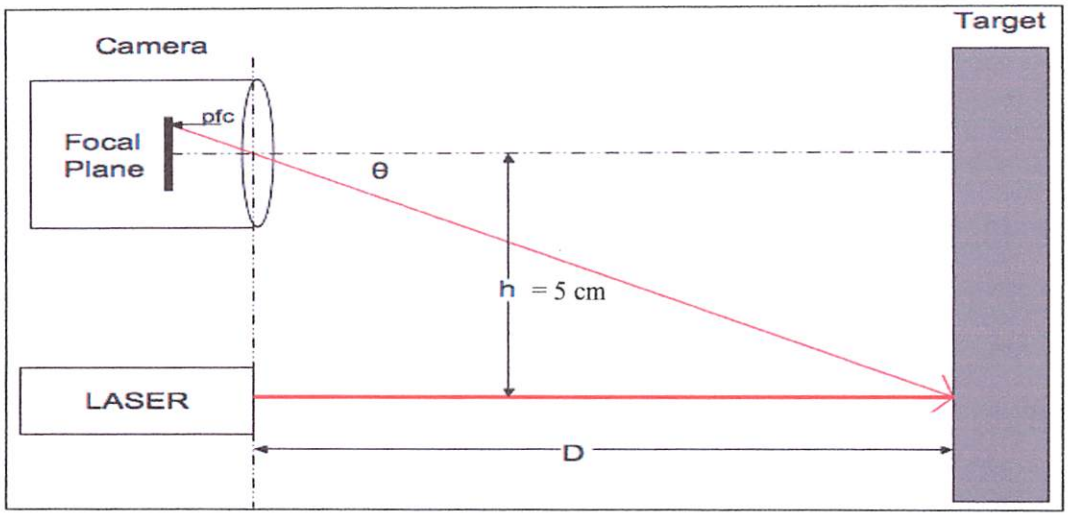
PENGUJIAN DAN ANALISA

Dalam bab ini akan dibahas tentang pengujian berdasarkan perencanaan dari sistem yang dibuat. Pengujian ini dilaksanakan untuk mengetahui kinerja dari suatu sistem yang dibuat. Pengujian ini dilaksanakan untuk mengetahui apakah sudah sesuai dengan perencanaan atau belum.

4.1 Pengujian dan Pengambilan Data

Proses pengujian dan pengambilan data dilakukan di dalam ruangan yang memiliki cahaya cukup terang. Sehingga pengaturan threshold pun dapat dilakukan dengan mudah. Pengambilan gambar harus dilakukan dengan cepat dan tepat untuk mendapatkan hasil yang maksimal.

Data yang akan diambil pada pengujian ini adalah nilai pfc dan jarak objek, sedangkan nilai rpc = 0.002293 dan ro = -0.07771 yang didapat dari kalibrasi. Gambar 4.1 dibawah menunjukkan berkas sinar laser yang diproyeksikan pada suatu target (objek). Kamera diletakkan sejajar dengan laser pointer yang berjarak konstan 5 cm disesuaikan untuk dapat menangkap proyeksi berkas sinar laser pada target sehingga dengan perhitungan matematika sederhana dapat menentukan informasi jarak (D) dari objek tersebut.



Gambar 4.1 Berkas laser yang diproyeksikan pada target

Dengan rumus trigonometri sederhana, maka jarak D dapat diperoleh dengan :

$$pfc = (Y_{max} / 2) - Y' \quad \text{..... (4-1)}$$

Dengan :

Y_{max} = jarak maksimum y = 240

Y' = posisi berkas laser yang jatuh pada sumbu Y

$$D = \frac{h}{\tan(pfc * rpc + ro)} \quad \text{..... (4-2)}$$

Dengan :

h = jarak kontan antara laser pointer dengan webcam

D = jarak antara sinar laser dengan target

pfc = pixel dari pusat *focal plane*

rpc = radian per pixel pitch (gain)

ro = radians offset (kompensasi error)

Berdasarkan rumus tersebut maka dapat dihitung hasil dari percobaan :

- Analisa pada percobaan pertama :

pfc $= (Y_{max} / 2)$-Y'

$= (240 / 2) - 1 = 119$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(119 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.195157)} = \frac{5}{0.19767} = 25.29 \text{ cm}$$

- Analisa pada percobaan kedua :

pfc $= (Y_{max} / 2)$-Y'

$= (240 / 2) - 48 = 72$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(72 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.133246)} = \frac{5}{0.0876} = 57.07 \text{ cm}$$

- Analisa pada percobaan ketiga :

pfc $= (Y_{max} / 2)$-Y'

$= (240 / 2) - 53 = 67$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(67 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.07592)} = \frac{5}{0.076} = 65.7 \text{ cm}$$

- Analisa pada percobaan keempat :

pfc $= (Y_{max} / 2)\text{-}Y'$

$= (240 / 2) - 56 = 64$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(64 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.069)} = \frac{5}{0.0691} = 72.3 \text{ cm}$$

- Analisa pada percobaan kelima :

pfc $= (Y_{max} / 2)\text{-}Y'$

$= (240 / 2) - 58) = 62$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(62 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.064456} = \frac{5}{(0.06454)} = 77.4 \text{ cm}$$

- Analisa pada percobaan keenam :

pfc $= (Y_{max} / 2)\text{-}Y'$

$= (240 / 2) - 60 = 60$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(60 * 0.002293 + (-0.07771)}$$

$$= \frac{5}{\tan(0.0598)} = \frac{5}{(0.05994} = 83.41 \text{ cm}$$

- Analisa pada percobaan ketujuh :

pfc $= (Y_{max} / 2)$-Y'

$= (240 / 2) - 62 = 58$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(58 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.05528)} = \frac{5}{0.05534} = 90.35 \text{ cm}$$

- Analisa pada percobaan kedelapan :

pfc $= (Y_{max} / 2)$-Y'

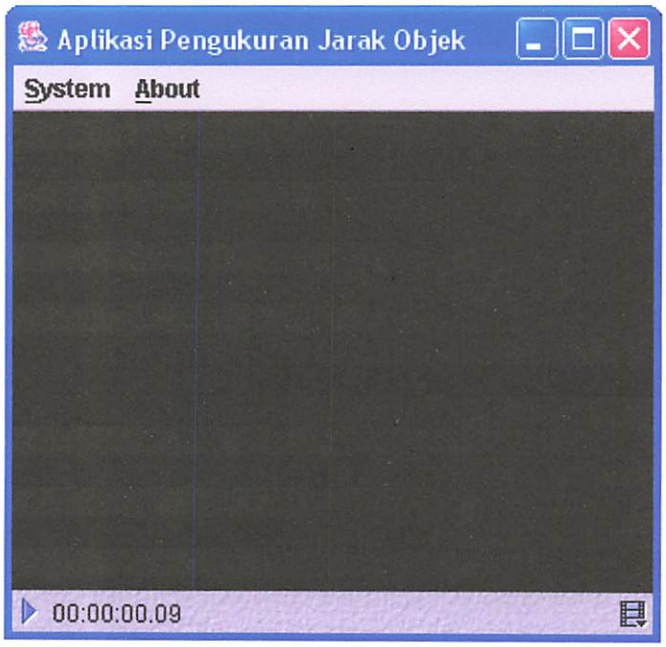
$= (240 / 2) - 65 = 55$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(55 * 0.002293 + (-0.07771))}$$

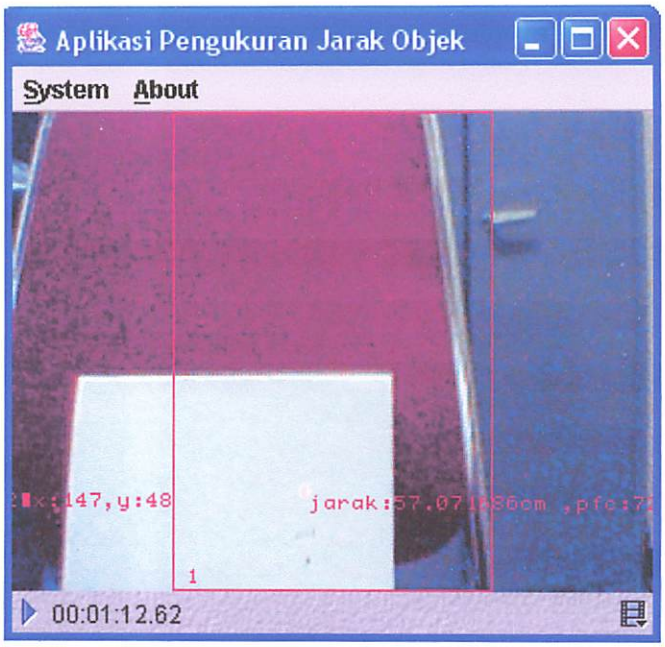
$$= \frac{5}{\tan(0.0484)} = \frac{5}{0.04844} = 103.2 \text{ cm}$$

Berikut tampilan form untuk mempermudah membaca data-data tersebut serta melihat proses pengolahan gambar bagi para pengguna:

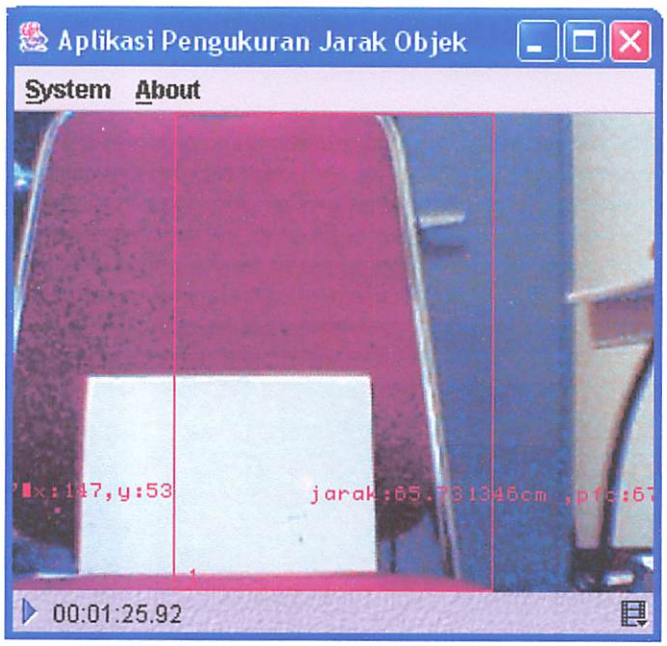


Gambar 4.2 *Tampilan form program*

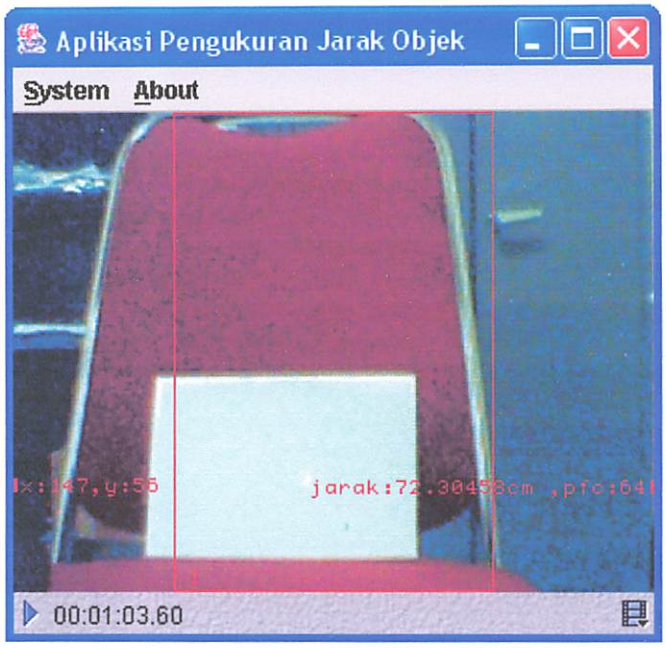
Gambar 4.3 *Tampilan form program pengujian pada jarak actual 25.5 cm*



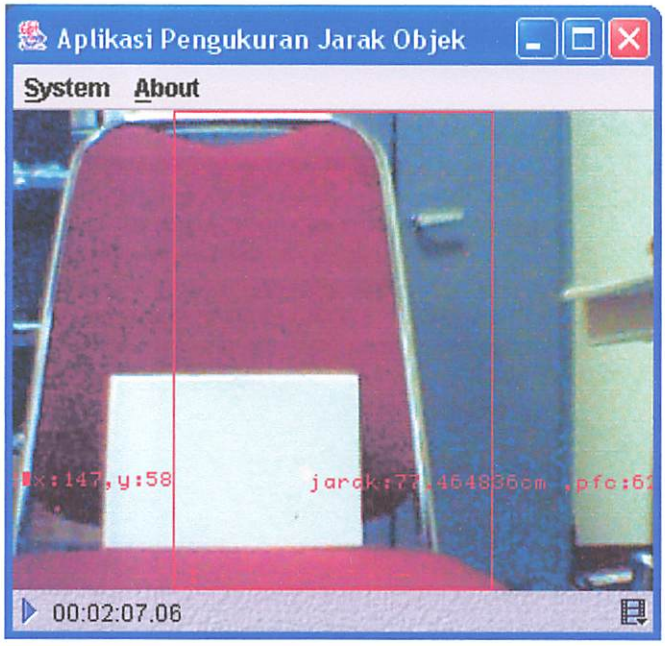
Gambar 4.4 *Tampilan form program pengujian pada jarak actual 59 cm*



Gambar 4.5 *Tampilan form program pengujian pada jarak actual 66 cm*



Gambar 4.6 *Tampilan form program pengujian pada jarak actual 72.5 cm*



Gambar 4.7 *Tampilan form program pengujian pada jarak actual 77.5 cm*

Gambar 4.8 *Tampilan form program pengujian pada jarak actual 84 cm*

Gambar 4.9 *Tampilan form program pengujian pada jarak actual 91.5 cm*

Gambar 4.10 *Tampilan form program pengujian pada jarak actual 104.5 cm*

Tabel 4.1 *Tabel data hasil pengujian software*

pfc	D hasil perhitungan (cm)	D actual (cm)	% error
119	25.294308	25.5	0.806
72	57.071686	59	3.268
67	65.731346	66	1.893
64	72.30458	72.5	0.269
62	77.464836	77.5	0.045
60	83.41447	84	0.697
58	90.34993	91.5	1.256
55	103.21445	104.5	1.23

Gambar 4.10 *Tampilan form program pengujian pada jarak actual 104.5 cm*

Tabel 4.1 *Tabel data hasil pengujian software*

pfc	D hasil perhitungan (cm)	D actual (cm)	% error
119	25.294308	25.5	0.806
72	57.071686	59	3.268
67	65.731346	66	1.893
64	72.30458	72.5	0.269
62	77.464836	77.5	0.045
60	83.41447	84	0.697
58	90.34993	91.5	1.256
55	103.21445	104.5	1.23

Berdasarkan percobaan pada sistem maka dapat diketahui persentase error pada sistem ini :

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|25.294308 - 25.5\right|}{25.5} x100\% \quad = 0.806\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|57.071686 - 59\right|}{59} x100\% \quad = 3.268\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|65.731346 - 67\right|}{67} x100\% \quad = 1.893\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|72.30458 - 72.2\right|}{72.5} x100\% \quad = 0.269\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|77.464836 - 77.5\right|}{77.5} x100\% \quad = 0.045\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|83.41447 - 84|}{84} x100\% \quad = 0.697\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|90.34993 - 91.5|}{91.5} x100\% \quad = 1.256\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|103.21445 - 104.5|}{104.5} x100\% \quad = 1.23\ \%$$

$$\text{Rata-rata \%error } = \frac{\sum error}{\sum pengujian}$$

$$= \frac{0.806 + 3.268 + 1.893 + 0.269 + 0.045 + 0.697 + 1.256 + 1.23}{8}$$

$$= \frac{9.464}{8} = 1.183\ \%$$

Berdasarkan rumus maka dapat dihitung hasil dari percobaan pada papan triplek (coklat) :

- Analisa pada percobaan pertama :

pfc $= (Y_{max} / 2)$-Y'

$= (240 / 2) - 10 = 110$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(110 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1745)} = \frac{5}{0.1763} = 28.358555 \text{ cm}$$

- Analisa pada percobaan kedua :

pfc $= (Y_{max} / 2)$-Y'

$= (240 / 2) - 28 = 92$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(92 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1332)} = \frac{5}{0.1340} = 37.30224 \text{ cm}$$

- Analisa pada percobaan ketiga :

pfc $= (Y_{max} / 2)$-Y'

$= (240 / 2) - 35 = 85$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(85 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1171)} = \frac{5}{0.1177} = 42.468433 \text{ cm}$$

- Analisa pada percobaan keempat :

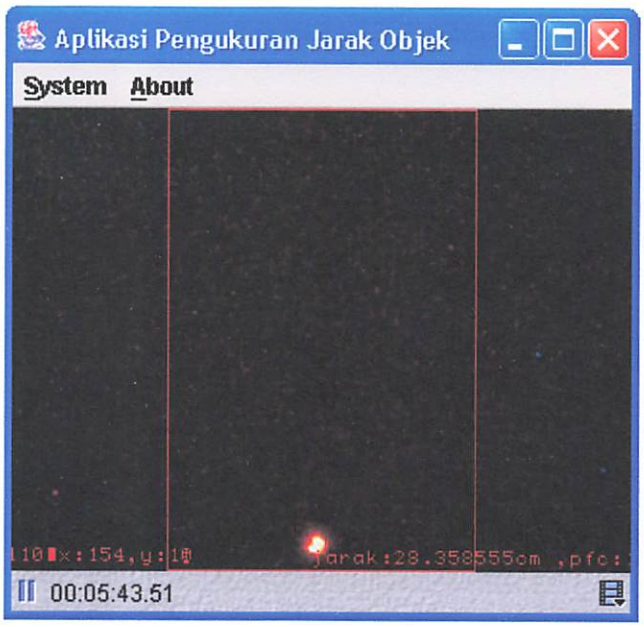
pfc $= (Y_{max} / 2)\text{-}Y'$

$= (240 / 2) - 44 = 76$

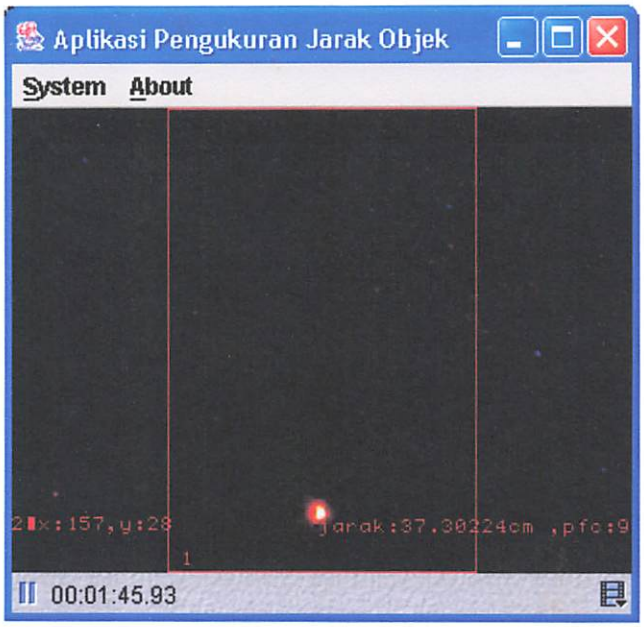
$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(76 * 0.002293 + (-0.07771))}$$

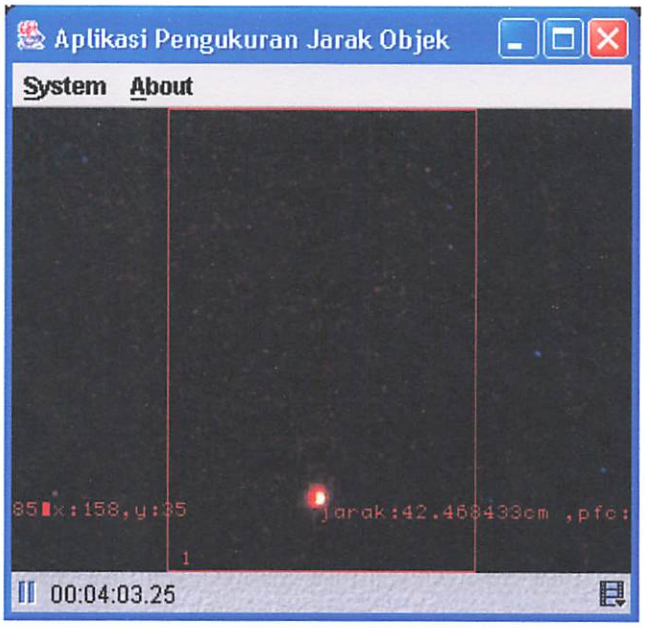
$$= \frac{5}{\tan(0.0965)} = \frac{5}{0.0968} = 51.621315 \text{ cm}$$



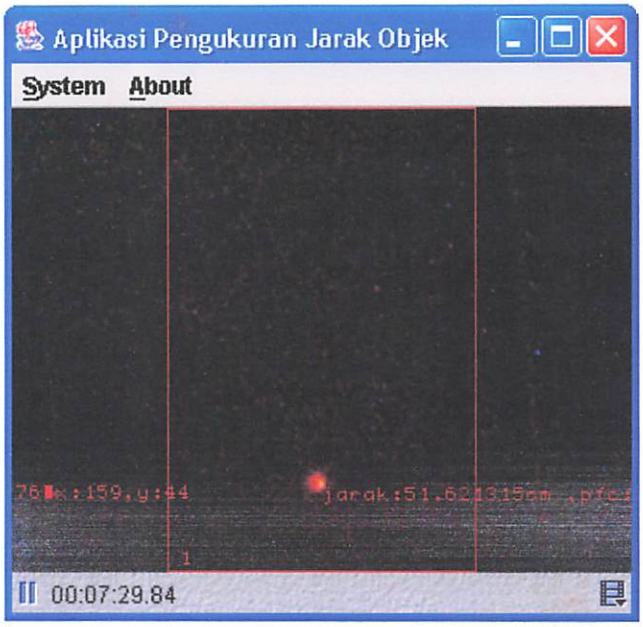
Gambar 4. 2 Form program pengujian triplek pada jarak aktual 25 cm



Gambar 4. 3 Form program pengujian triplek pada jarak aktual 30 cm



Gambar 4. 4 Form program pengujian triplek pada jarak aktual 35 cm



Gambar 4. 5 Form program pengujian triplek pada jarak aktual 40 cm

Tabel 4.2 Tabel data hasil pengujian pada papan trilek (coklat) software

pfc	D hasil perhitungan (cm)	D actual (cm)	% error
110	28.358555	25	13.43
92	37.30224	30	24.34
85	42.468433	35	21.33
76	51.621315	40	29.05

Berdasarkan percobaan pada sistem maka dapat diketahui persentase error pada sistem ini :

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|28.358555 - 25\right|}{25} x100\% \quad = 13.43\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|37.30224 - 30|}{30} x100\% \quad = 24.34\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|42.468433 - 35|}{35} x100\% \quad = 21.33\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|51.621315 - 40|}{40} x100\% \quad = 29.05\ \%$$

$$\text{Rata-rata \%error } = \frac{\sum error}{\sum pengujian}$$

$$= \frac{13.43 + 24.34 + 21.33 + 29.05}{4}$$

$$= \frac{88.15}{4} = 22.03\ \%$$

Berdasarkan rumus maka dapat dihitung hasil dari percobaan pada plastik (hijau) :

- Analisa pada percobaan pertama :

  pfc $= (Y_{max} / 2) - Y'$

  $= (240 / 2) - 1 \quad = \quad 119$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(119 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1951)} = \frac{5}{0.1976} = 25.294308 \text{ cm}$$

- Analisa pada percobaan kedua :

pfc $= (Y_{max} / 2)\text{-}Y'$

$= (240 / 2) - 17 = 103$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(103 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1584)} = \frac{5}{0.1598} = 31.287355 \text{ cm}$$

- Analisa pada percobaan ketiga :

pfc $= (Y_{max} / 2)\text{-}Y'$

$= (240 / 2) - 22 = 98$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(98 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1470)} = \frac{5}{0.1480} = 33.767323 \text{ cm}$$

- Analisa pada percobaan keempat :

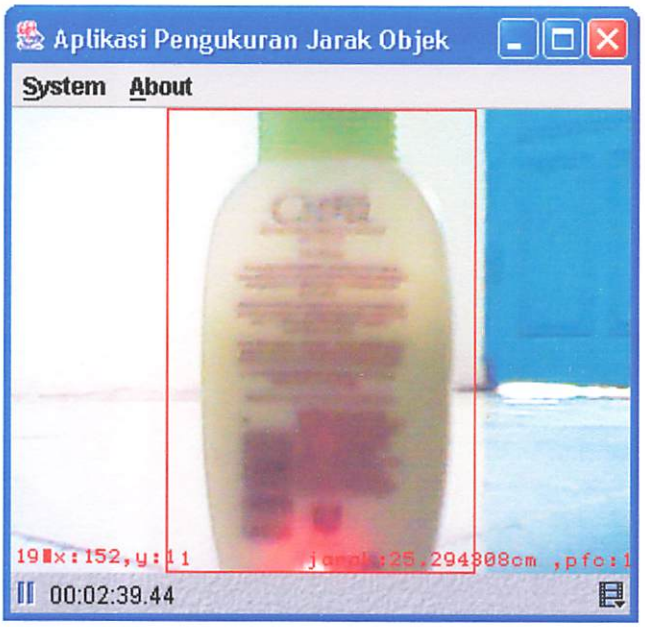
pfc $= (Y_{max} / 2)\text{-}Y'$

$= (240 / 2) - 37 \quad = \quad 83$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

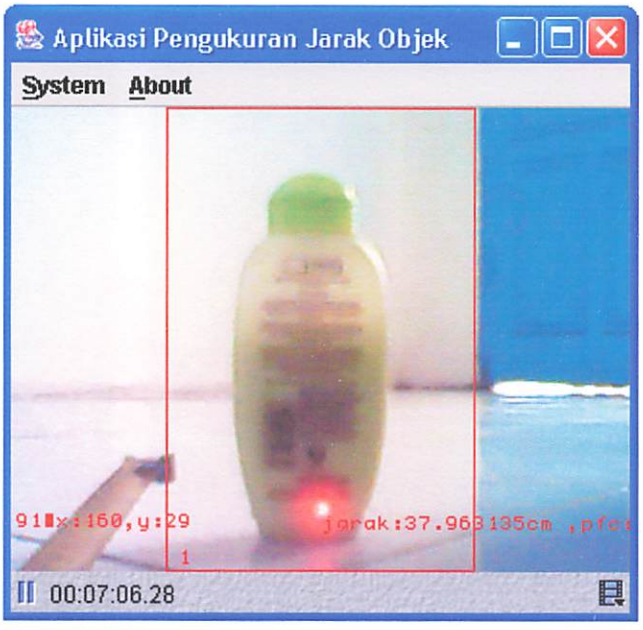
$$= \frac{5}{\tan(83 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1126)} \quad = \frac{5}{0.11308} \quad = 44.213585 \text{ cm}$$

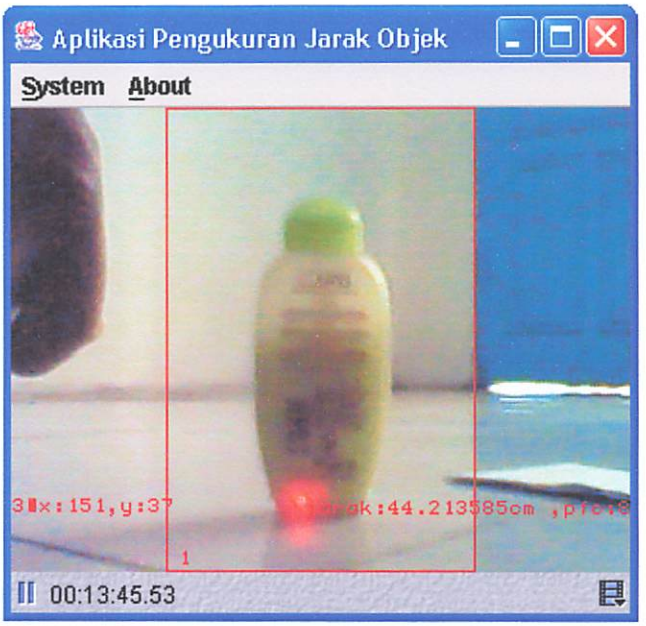


Gambar 4. 6 Form program pengujian plastik pada jarak aktual 25 cm

Gambar 4. 7 Form program pengujian plastik pada jarak aktual 30 cm



Gambar 4. 8 Form program pengujian plastik pada jarak aktual 35 cm



Gambar 4. 9 Form program pengujian plastik pada jarak aktual 40 cm

Tabel 4.3 Tabel data hasil pengujian pada plastik (hijau) software

pfc	D hasil perhitungan (cm)	D actual (cm)	% error
119	25.294308	25	1.17
103	31.287355	30	4.29
91	37.963135	35	8.46
83	44.213585	40	10.53

Berdasarkan percobaan pada sistem maka dapat diketahui persentase error pada sistem ini :

$$\%\ \text{error} = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|25.294308 - 25\right|}{25} x100\% \quad = 1.17\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|31.287355 - 30\right|}{30} x100\% \quad = 4.29\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|37.963135 - 35\right|}{35} x100\% \quad = 8.46\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|44.213585 - 40\right|}{40} x100\% \quad = 10.53\ \%$$

$$\text{Rata-rata \%error } = \frac{\sum error}{\sum pengujian}$$

$$= \frac{1.17 + 4.29 + 8.46 + 10.53}{4}$$

$$= \frac{24.45}{4} = 6.1125\ \%$$

Berdasarkan rumus maka dapat dihitung hasil dari percobaan pada plastik (hijau) :

- Analisa pada percobaan pertama :

  pfc $= (Y_{max} / 2)\text{-}Y'$

  $= (240 / 2) - 1 \quad = \quad 119$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(119 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1951)} = \frac{5}{0.1976} = 25.294308 \text{ cm}$$

- Analisa pada percobaan kedua .

pfc $= (Y_{max} / 2) - Y'$

$= (240 / 2) - 17 = 103$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(103 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1584)} = \frac{5}{0.1598} = 31.287355 \text{ cm}$$

- Analisa pada percobaan ketiga .

pfc $= (Y_{max} / 2) - Y'$

$= (240 / 2) - 22 = 98$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(98 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1470)} = \frac{5}{0.1480} = 33.767323 \text{ cm}$$

- Analisa pada percobaan keempat :

$pfc = (Y_{max} / 2) - Y'$

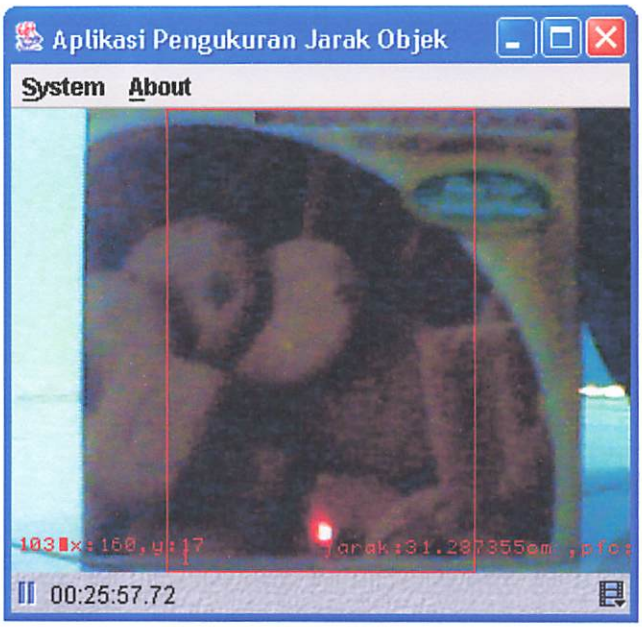
$= (240 / 2) - 37 = 83$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(83 * 0.002293 + (-0.07771))}$$

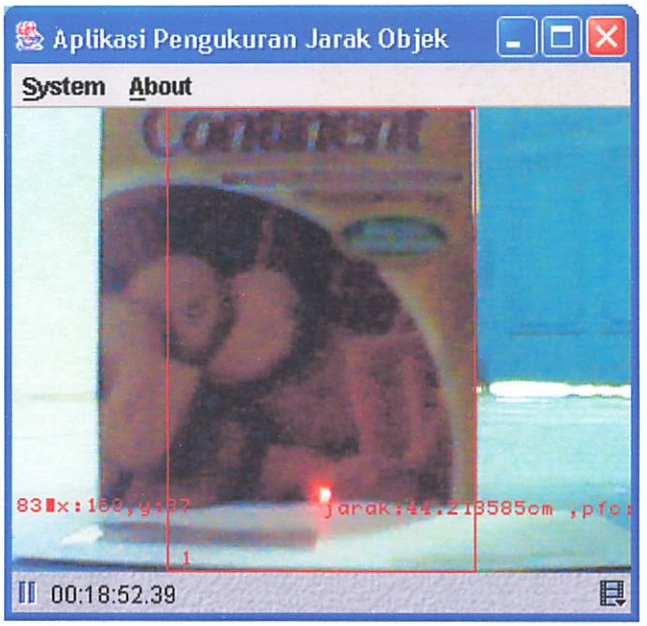
$$= \frac{5}{\tan(0.1126)} = \frac{5}{0.11308} = 44.213585 \text{ cm}$$

Gambar 4. 10 Form program pengujian kaleng pada jarak aktual 25 cm



Gambar 4. 11 Form program pengujian kaleng pada jarak aktual 30 cm

Gambar 4. 12 Form program pengujian kaleng pada jarak aktual 35 cm



Gambar 4. 13 Form program pengujian kaleng pada jarak aktual 40 cm

Tabel 4.4 Tabel data hasil pengujian pada kaleng (coklat) software

pfc	D hasil perhitungan (cm)	D actual (cm)	% error
119	25.294308	25	1.17
103	31.287355	30	4.29
98	33.767323	35	3.52
83	44.213585	40	10.53

Berdasarkan percobaan pada sistem maka dapat diketahui persentase error pada sistem ini :

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|25.294308 - 25\right|}{25} x100\% \quad = 1.17\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|31.287355 - 30|}{30} x100\% \quad = 4.29\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|33.767323 - 35|}{35} x100\% \quad = 3.52\ \%$$

$$\% \text{ error } = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|44.213585 - 40|}{40} x100\% \quad = 10.53\ \%$$

$$\text{Rata-rata \%error } = \frac{\sum error}{\sum pengujian}$$

$$= \frac{1.17 + 4.29 + 3.52 + 10.53}{4}$$

$$= \frac{19.51}{4} = 4.8775\ \%$$

Berdasarkan rumus maka dapat dihitung hasil dari percobaan pada tangan (coklat):

- Analisa pada percobaan pertama :

  pfc $= (Y_{max} / 2)$-Y'

  $= (240 / 2) - 1 = 119$

  $$D = \frac{h}{\tan(pfc * rpc + ro)}$$

  $$= \frac{5}{\tan(119 * 0.002293 + (-0.07771))}$$

  $$= \frac{5}{\tan(0.1951)} = \frac{5}{0.1976} = 25.294308 \text{ cm}$$

- Analisa pada percobaan kedua :

  pfc $= (Y_{max} / 2)$-Y'

  $= (240 / 2) - 17 = 103$

  $$D = \frac{h}{\tan(pfc * rpc + ro)}$$

  $$= \frac{5}{\tan(103 * 0.002293 + (-0.07771))}$$

  $$= \frac{5}{\tan(0.1584)} = \frac{5}{0.1598} = 31.287355 \text{ cm}$$

- Analisa pada percobaan ketiga :

  pfc $= (Y_{max} / 2)$-Y'

  $= (240 / 2) - 26 = 94$

  $$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(94 * 0.002293 + (-0.07771))}$$

$$= \frac{5}{\tan(0.1379)} = \frac{5}{0.1388} = 36.046036 \text{ cm}$$

- Analisa pada percobaan keempat :

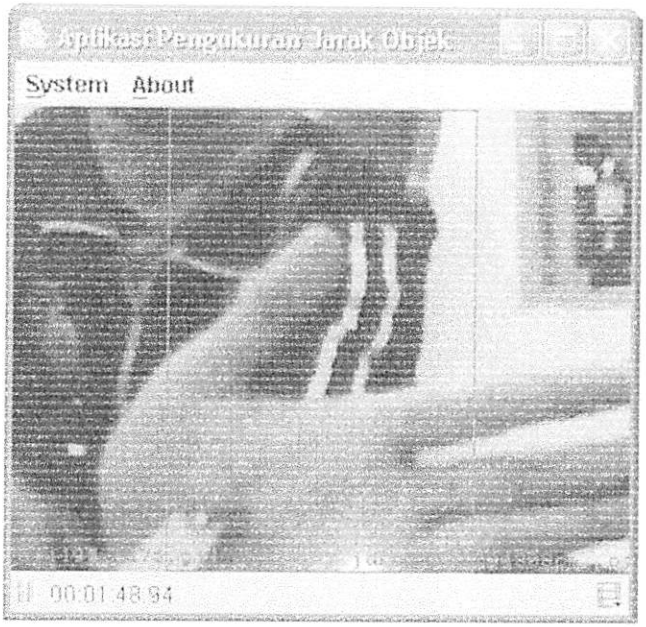
$pfc = (Y_{max} / 2) - Y'$

$= (240 / 2) - 34 = 86$

$$D = \frac{h}{\tan(pfc * rpc + ro)}$$

$$= \frac{5}{\tan(86 * 0.002293 + (-0.07771))}$$

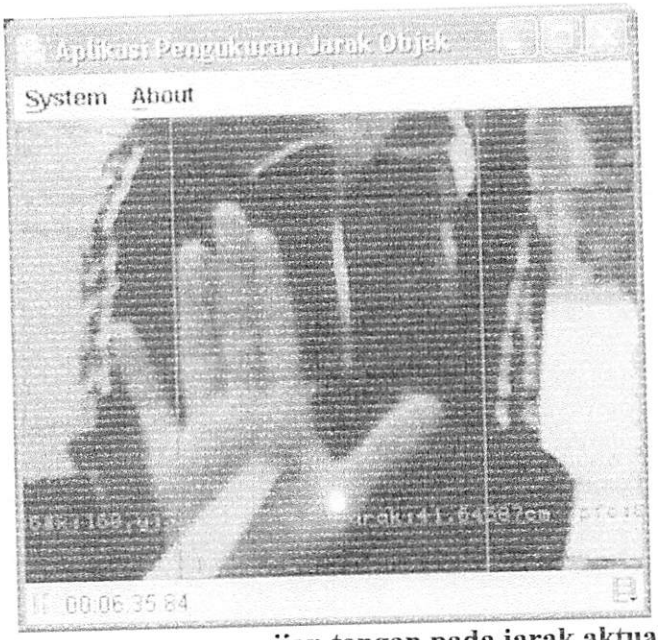
$$= \frac{5}{\tan(0.1194)} = \frac{5}{0.12} = 41.64587\text{cm}$$



Gambar 4. 232 Form program pengujian tangan pada jarak aktual 25 cm

Gambar 4. 24 Form program pengujian tangan pada jarak aktual 30 cm

Gambar 4. 25 Form program pengujian tangan pada jarak aktual 35 cm



Gambar 4. 26Form program pengujian tangan pada jarak aktual 40 cm

Tabel 4.5 Tabel data hasil pengujian pada tangan (coklat) software

pfc	D hasil perhitungan (cm)	D actual (cm)	% error
119	25.294308	25	1.17
102	31.754444	30	5.84
94	36.046036	35	2.94
86	41.64587	40	4.11

Berdasarkan percobaan pada sistem maka dapat diketahui persentase error pada sistem ini :

$$\%\ \text{error} = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{\left|25.294308 - 25\right|}{25} x100\% \quad = 1.17\ \%$$

$$\%\ \text{error} = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|31.754444 - 30|}{30} x100\% = 5.84\ \%$$

$$\%\ \text{error} = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|36.046036 - 35|}{35} x100\% = 2.94\ \%$$

$$\%\ \text{error} = \frac{\left|D_{hasilperhitungan} - D_{aktual}\right|}{D_{aktual}} x100\%$$

$$= \frac{|41.64587 - 40|}{40} x100\% = 4.11\ \%$$

$$\text{Rata-rata \%error} = \frac{\sum error}{\sum pengujian}$$

$$= \frac{1.17 + 5.84 + 2.94 + 4.11}{4}$$

$$= \frac{14.06}{4} = 3.515\ \%$$

Pengujian diatas merupakan pengujian hasil scanning ketika objek yang berupa titik terletak di depan laser dan tertangkap oleh webcam. Setelah webcam melakukan scanning dan menangkap objek dari hasil scanning tadi, maka hasil scanning tersebut akan dikirim ke PC lewat komunikasi usb. Setelah PC mendapat input hasil scanning objek oleh webcam maka program pada PC akan menghitung

input hasil scanning objek oleh webcam maka program pada PC akan menghitung sesuai dengan hasil scanning objek tadi. Tetapi ketika webcam tidak menangkap objek, maka kamera akan terus melakukan scanning sampai menemukan. Batas maksimal dasar peletakan objek sudah dimasukkan dalam setting point pada program di java. Berikut tampilan form hasil scanning ketika objek ditemukan maupun ketika pencarian objek :

Gambar 4.10 *Tampilan form program untuk objek sinar laser terdeteksi*

Ketika proses scanning dilakukan dan objek ditemukan maka pada form program akan langsung muncul nilai jarak, nilai pfc, koordinat sumbu x dan y, hal ini berarti proses scanning untuk mencari objek telah dilakukan dan dari proses scanning tadi telah ditemukan objek.

Gambar 4.11 *Tampilan form program untuk pencarian objek sinar laser yang tak terdeteksi*

Tetapi ketika proses scanning dilakukan dan ternyata sinar laser tidak terdeteksi oleh kamera sehingga kamera akan terus melakukan pencarian sinar laser atau juga akan mendeteksi objek yang memiliki intensitas cahaya sesuai dengan threshold yang sudah ditetapkan oleh program.

4.2. Analisa

Perhitungan jarak suatu objek didasarkan pada lokasi jatuhnya berkas sinar laser tersebut pada sumbu **Y** pada layar yang dijadikan nilai dari variable pfc. Dimana nilai pfc ini mempengaruhi besar sudut theta yang digunakan untuk perhitungan jarak suatu objek.

Berkas sinar laser yang ditangkap oleh webcam akan langsung diproses dengan menggunakan teknik *digital image processing* yakni image yang masih berupa intensitas cahaya (energi) dideteksi oleh detektor kemudian dirubahan menjadi sinyal kontinue elektrik, dimana sinyal kontinue tersebut akan disampling

BAB V

PENUTUP

5.1 Kesimpulan

Dari hasil pengujian skripsi ini dapat ditarik beberapa kesimpulan, antara lain:

1. Berdasarkan hasil pengujian tabel 4.1 bahwa semakin dekat berkas sinar laser dengan pusat gambar, maka semakin jauh jarak target tersebut.
2. Dari hasil pengujian sistem diperoleh tingkat keberhasilan sebesar 98.817% dalam ketepatan mengukur jarak suatu objek.
3. Dari hasil pengujian hanya dapat mengukur objek yang jarak minimal sebesar 25 cm dan maksimal sebesar 103 cm yang berbentuk bidang datar berwarna putih dan bahannya dari kertas dapat memantulkan berkas sinar laser yang ditembakkan.

5.2 Saran

Dari kekurangan-kekurangan yang kami peroleh pada saat pengujian program, kami memberikan beberapa saran, antara lain:

1. Sebelum menjalankan peralatan, dimana peralatan tersebut menggunakan kamera sebagai sensornya, sebaiknya dilakukan pengesetan kamera lebih dahulu, agar gambar yang didapatkan sesuai dengan ketentuan referensi data pada program. Pengesetan tersebut diantaranya adalah pengaturan pencahayaan kamera sesuai dengan tempat pengambilan gambar, peletakan sensor kamera pada mekanik atau plan yang dijalankan.

2. Perlu ditambah suatu program aplikasi agar intensitas cahaya pada waktu pemrosesan dapat stabil atau diberi cahaya dari lampu sehingga intensitasnya stabil.

DAFTAR PUSTAKA

[1] Rinaldi Munir, Pengolahan Citra Digital dengan Pendekatan Algoritmik, INFORMATIKA, Bandung, 2001.

[2] http://www.logix4u.net/

[3] http://www.beyondlogic.org/epp/epp.htm

[4] Laser Range Finder Tutorialhttp://www.pages.drexel.edu/~twd25/webcam_laser_ranger.html.

[5] Danko, Tood. Webcam based DIY Laser Rangefinder. Drexel University, Belgium,2002.

[6] Grundgeiger, Dave. Measuring Distance with a Laser. Stanford University, Stanford, 2004.

[7] Lejos Vision System http://homepage.ntlword.com

[8] http://java.sun.com

[9] http://lejos.sourceforge.net

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

LEMBAR BIMBINGAN SKRIPSI

Nama	: Adelia Amityas
NIM	: 03.17.016
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika
Judul Skripsi	: Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman Java
Mulai Bimbingan Skripsi	: 11 April 2007
Selesai Bimbingan Skripsi	: 11 Oktober 2007
Pembimbing I	: Ir. H. Sidik Noertjahjono, MT
Pembimbing II	: M. Ashar, ST, MT
Dengan Nilai	: 85

Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Ir. H. Sidik Noertjahjono, MT	**M. Ashar, ST, MT**
NIP.1028700163	NIP. 1030500408

Mengetahui
Ketua Jurusan Teknik Elektro S-1

(Ir. F. Yudi Limpraptono, MT)
NIP.Y. 1039500274

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO
KONSENTRASI ELEKTRONIKA

LEMBAR PERSETUJUAN PERBAIKAN SKRIPSI

Dari hasil ujian skripsi jurusan Teknik Elektro jenjang strata satu (S-1), yang diselenggarakan pada:

Hari : Senin
Tanggal : 03 September 2007

Telah dilakukan perbaikan oleh:

Nama : Adelia Amityas
N.I.M : 03.17.016
Jurusan : TEKNIK ELEKTRO S-1
Konsentrasi : ELEKTRONIKA
Judul Skripsi : **Paplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman Java**

Perbaikan meliputi:

No.	Materi Perbaikan	Keterangan
1.	Dilakukan Pengujian Dengan Warna Selain Putih (kertas) 4 analisa	

Penguji

Joseph Dedy Irawan, ST, MT
NIP.132315178

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

FORMULIR PERBAIKAN SKRIPSI

Nama : Adelia Amityas

NIM : 03.17.016

Masa Bimbingan : 11 April 2007 s/d 11 Oktober 2007

Judul : Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman Java

No.	Tanggal	Uraian	Paraf
1	3/09/2007	Dilakukan Pengujian Dengan Warna Selain putih (kertas) 4 analisa	

Disetujui,

Penguji I

(Joseph Dedy Irawan, ST, MT)
NIP.P.1039800324

Penguji II

(I Komang Somawirata, ST, MT)
NIP.P. 1030100361

Mengetahui,

Dosen Pembimbing I

(Ir. H. Sidik Noertjahjono, MT)
NIP.Y. 1028700167

Dosen Pembimbing II

(M. Ashar, ST, MT)
NIP. 1030500408

JURUSAN TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG

FORMULIR PERBAIKAN SKRIPSI

Dalam pelaksanaan Ujian Skripsi, perlu adanya perbaikan skripsi untuk mahasiswa :

Nama	:	ADELIA ANITYAS	
NIM	:	03 17 016	
Fakultas	:	Teknologi Industri	
Jurusan	:	Teknik Elektro S-1	
Konsentrasi	:	1.	Teknik Energi Listrik *)
		2.	Teknik Elektronika *)
		3.	Tenik Komputer dan Informatika *)

Perbaikan meliputi :
*) DILAKUKAN PENGUJIAN DGN WARNA SELAIN PUTIH (KERTAS) & analisa.

INSTITUT TEKNOLOGI NASIONAL
JL. BENDUNGAN SIGURA-GURA 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Adelia Amityas
Nim : 0317016
Masa Bimbingan : 11 April 2007 s/d 11 Oktober 2007
Judul Skripsi : Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman JAVA

NO	Tanggal	Uraian	Paraf Pembimbing
1.	5/4 - 07.	konsultasi judul, cari blm² yg diperlukan.	
2.	15/4 - 07.	Pada dasar² tlg kamera, carikan tlg pixel & CCD theori	
3.	1/5 - 07.	Pada kam. yg akan di pakai, berapa kali frame/det?	
4.	12/5 07.	Lakukan percobaan tlg pengolahan pixel.	
5.	2/6 07.	Bab III, ambil contoh publik tlg. acara perancangan.	
6	16/6 07.	Bab IV, cermati besar pixel yg. mendapat cahaya	
7	22/6 07.	Bab IV, analisa hasil. tulis di Bab V.	
8	29/6 07.	Bab II, masukkan teori tlg. pengolahan citra.	
9	3/7 07.	Bab I, ingat di batasan masalah, garis nulebar.	
10	20/7 07.	Laporan skripsi, siapkan ujian.	

Malang,
Dosen Pembimbing I

Ir. H. Sidik Noerjahjono, MT
NIP. P. 1029900163

Form. S-4a

246

INSTITUT TEKNOLOGI NASIONAL
JL. BENDUNGAN SIGURA-GURA 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Adelia Amitya
Nim : 0317016
Masa Bimbingan : 11 April 2007 s/d 11 Oktober 2007
Judul Skripsi : Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman JAVA

NO	Tanggal	Uraian	Paraf Pembimbing
1.			
2.			
3.			
4.			
5.			
6			
7			
8			
9			
10			

Malang,
Dosen Pembimbing I

Ir. H. Sidik Noerjahjono, MT
NIP. P. 1029900103

Form. S-1a

INSTITUT TEKNOLOGI NASIONAL
JL. BENDUNGAN SIGURA-GURA 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Adelia Amityas
Nim : 0317016
Masa Bimbingan : 11 April 2007 s/d 11 Oktober 2007
Judul Skripsi : Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman JAVA

NO	Tanggal	Uraian	Paraf Pembimbing
1.	21/05/07	Bab I, Rumusan Masalah.	
2.		Bab II revisi	
3.		Bab III revisi	
4.		Bab IV revisi	
5.		Pengertian Laser	
6		Pengertian Jarak	
7		Pengertian Software.	
8		Maju Sidang Seminar	
9		Bab V (revisi)	
10		Acc (compro)	

Malang,
Dosen Pembimbing II

M. Ashar ST, MT
NIP.

Form. S-4a

247

INSTITUT TEKNOLOGI NASIONAL
JL. BENDUNGAN SIGURA-GURA 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : Adelia Amliyas
Nim : 03170165
Masa Bimbingan : 11 April 2007 s/d 11 Oktober 2007
Judul Skripsi : Aplikasi Teknologi Laser Range Finder Untuk Pengukuran Jarak Suatu Objek Menggunakan Bahasa Pemprograman JAVA

NO	Tanggal	Uraian	Paraf Pembimbing
1.			
2.			
3.			
4.			
5.			
6			
7			
8			
9			
10			

Malang,
Dosen Pembimbing II

M. Ashar ST, MT
NIP.

Form. S-4a

LAMPIRAN

OnvineLPS.java

```
//frame & Dimension
import java.awt.*;
//KeyEvent & ActionListener
import java.awt.event.*;
//JOptionPane & JMenuItem, JMenu,JMenuBAr
import javax.swing.*;
//Vector
import java.util.*;

public class OnvineLPS implements ColorListener {
  static private JMenuBar menubar            = null;
  static private JMenuItem menuItem_btnon1   = null;
  static private JMenuItem menuItem_LaserOff  = null;
  static private JMenuItem menuItem_Connect   = null;
  static private JMenuItem menuItem_Disconnect = null;
  static private JMenuItem menuItem_Exit      = null;
  static private JMenuItem menuItem_VersionInfo= null;
  static private JOptionPane optionPane       = null;
  //static { System.loadLibrary("jnpout32");}
  static private float range         = 0;
  static private int CalibrateRange = 0;
  static private boolean GetCheck   = false;
  static private boolean doRun      = true;
  static private Vector vec         = new Vector();
  static private ioPort parport = new ioPort();

  public OnvineLPS() {
          pTimer m_pTimer = new pTimer();
          m_pTimer.start();
  }

  public void colorDetected(int region, int color) {}

  public static void main(String [] args) {

    OnvineLPS listener = new OnvineLPS();

    // Create 1 regions and set them to look for a brightest pixel
    Vision.addRectRegion(1, 80, 0, 160, 240);
    Vision.addColorListener(1, listener, 0);

          // Create the viewer and set its title
    Vision.startViewer("Aplikasi Pengukuran Jarak Objek");
```

```
        // Menu tool bar
    menubar = new JMenuBar();
    menubar.add(createMenu("System"));
        menubar.add(createMenu("About"));

        // Get the frame
    Frame visionFrame = Vision.getFrame();

        Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension mySize = visionFrame.getSize();

        visionFrame.setLocation((screenSize.width - mySize.width) / 2,
(screenSize.height - mySize.height) / 2);

        // Disconnect first
    Vision.p.stop();

        visionFrame.add("North", menubar);
    visionFrame.pack();
    visionFrame.setVisible(true);
}

private static JMenu createMenu(String title) {
    JMenu menu = new JMenu();
    menu.setText(title);

    //Add "System" toolbar
    if (title.equals("System")) {
      menu.setMnemonic(KeyEvent.VK_S);
                    menu.add(createItem("Laser On"));
      menu.add(createItem("Laser Off"));
                    menu.add(createItem("Connect"));
      menu.add(createItem("Disconnect"));
      menu.add(createItem("Exit"));
    }

    //Add "About" toolbar
    if (title.equals("About")) {
      menu.setMnemonic(KeyEvent.VK_A);
      menu.add(createItem("Version Info"));
    }
    return menu;
}

private static JMenuItem createItem(String title) {
    JMenuItem menuItem = new JMenuItem(title);

            //Add "Laser On" sub-toolbar
```

```
    if (title.equals("Laser On")) {
       menuItem.setMnemonic(KeyEvent.VK_L);
       menuItem.getAccessibleContext().setAccessibleDescription("Laser On");
       menuItem_btnon1 = menuItem;
       //menuItem.setIcon(new ImageIcon(filepath + "images" +
System.getProperty("file.separator") + "Laser On1..gif"));
    }

    if (title.equals("Laser Off")) {
       menuItem.setMnemonic(KeyEvent.VK_O);
       menuItem.getAccessibleContext().setAccessibleDescription("Laser Off");
       menuItem_LaserOff = menuItem;
                    menuItem.setEnabled(false);
       //menuItem.setIcon(new ImageIcon(filepath + "images" +
System.getProperty("file.separator") + "Laser On1..gif"));
    }
            //Add "Connect" sub-toolbar
    if (title.equals("Connect")) {
       menuItem.setMnemonic(KeyEvent.VK_C);
       menuItem.getAccessibleContext().setAccessibleDescription("Connect");
       menuItem_Connect = menuItem;
       //menuItem.setIcon(new ImageIcon(filepath + "images" +
System.getProperty("file.separator") + "connect1.gif"));
    }

             //Add "Disconnect" sub-toolbar
    else if (title.equals("Disconnect")) {
       menuItem.setMnemonic(KeyEvent.VK_D);

menuItem.getAccessibleContext().setAccessibleDescription("Disconnect");
       menuItem_Disconnect = menuItem;
       menuItem.setEnabled(false);
       //menuItem.setIcon(new ImageIcon(filepath + "images" +
System.getProperty("file.separator") + "disconnect1.gif"));
    }

            //Add "Exit" sub-toolbar
    else if (title.equals("Exit")) {
       menuItem.setMnemonic(KeyEvent.VK_X);
       menuItem.getAccessibleContext().setAccessibleDescription("Exit");
       menuItem_Exit = menuItem;
       //menuItem.setIcon(new ImageIcon(filepath + "images" +
System.getProperty("file.separator") + "exit.gif"));
    }

            //Add "Exit" sub-toolbar
    else if (title.equals("Version Info")) {
       menuItem.setMnemonic(KeyEvent.VK_V);
```

```
        menuItem.getAccessibleContext().setAccessibleDescription("Version
Info");
        menuItem_VersionInfo = menuItem;
        //menuItem.setIcon(new ImageIcon(filepath + "images" +
System.getProperty("file.separator") + "VersionInfo.gif"));
    }

            menuItem.addActionListener(createActionListener(title));
    return menuItem;
  }
 private static boolean doCalibrate() {
        int f       = 0;
        boolean added = false;
        GetCheck    = false;
        doRun       = true;
        vec.removeAllElements();
        vec.setSize(0);
        while(doRun){
                try {
                        if(f>=100) doRun = false;

                        if(vec.isEmpty())
                                vec.addElement(Float.toString(range));
                        else {
                                for(int g=0; g<vec.size(); g++) {
                                        int  s = Integer.parseInt( (String)
vec.elementAt(g));
                                        added = true;
                                }

                                 if(added) {
                                        vec.addElement(Float.toString(range));
                                        added = false;
                                 }
                        }

                        f++;

                        Thread tclr = Thread.currentThread();
                        tclr.sleep(100);
                }
                catch (InterruptedException ie) { return false; }
        }
        if(vec.size() <= 1) {
                if(Integer.parseInt( (String) vec.elementAt(0))==0) {
                        JOptionPane.showMessageDialog(optionPane, "Sinar Laser
Tidak Terdeteksi", "Error", JOptionPane.INFORMATION_MESSAGE);
                        Vision.p.stop();
                }
```

```
                else
                        GetCheck = true;
        } else
                GetCheck = true;

        return GetCheck;
}
public static boolean set_bit() {
        int value, status_port, new_value, address_port, number_bit;
        address_port=0x378;
        number_bit=0;
        value = 255;
        new_value = 0;
        status_port = parport.Inp32(address_port);
        new_value = status_port | value;
        parport.Out32(address_port, new_value);
        return true;
        }

public static boolean clear_bit()
{
int value, status_port, new_value,address_port,number_bit;
value = 0;
new_value = 0;
number_bit=0;
address_port=0x378;
status_port = parport.Inp32(address_port);
new_value = status_port & value;
parport.Out32(address_port, new_value);
return true;
}

 public static ActionListener createActionListener(String s) {
      ActionListener a = null;

                if (s.equals("Laser On")) {
        a = new ActionListener() {
                                public void actionPerformed(ActionEvent evt)
                        {
                        Object cmd = evt.getSource();
                        if(set_bit())     {

                                                menuItem_btnon1.setEnabled(false);
                                                menuItem_Connect.setEnabled(true);

        menuItem_Disconnect.setEnabled(true);
                                        }
        menuItem_LaserOff.setEnabled(true);
           }
```

```
};
                } else if (s.equals("Connect")) {
a = new ActionListener() {
   public void actionPerformed(ActionEvent ae) {
      Vision.p.start();
                                    if(doCalibrate()) {
                                          menuItem_btnon1.setEnabled(true);

menuItem_LaserOff.setEnabled(true);

menuItem_Connect.setEnabled(false);

menuItem_Disconnect.setEnabled(true);
                                    }
   }
};
        } else if (s.equals("Laser Off")) {
a = new ActionListener() {
   public void actionPerformed(ActionEvent evt) {
                           Object cmd = evt.getSource();
                           if(clear_bit()) {
                  menuItem_btnon1.setEnabled(true);

menuItem_LaserOff.setEnabled(false);
                                          menuItem_Connect.setEnabled(true);

menuItem_Disconnect.setEnabled(true);
                           }
   }
};

                           } else if (s.equals("Disconnect")) {
                  a = new ActionListener() {
   public void actionPerformed(ActionEvent ae) {
      GetCheck = false;
                                    Vision.p.stop();
                                    menuItem_btnon1.setEnabled(true);
                                    menuItem_LaserOff.setEnabled(true);
                                    menuItem_Connect.setEnabled(true);
                                    menuItem_Disconnect.setEnabled(false);
   }
};
        } else if (s.equals("Exit")) {
                  a = new ActionListener() {
   public void actionPerformed(ActionEvent ae) {
                                    Vision.p.close();
                                    System.exit(0);
   }
};
```

```
                } else if (s.equals("Version Info")) {
                        a = new ActionListener() {
            public void actionPerformed(ActionEvent ae) {
                                JOptionPane.showMessageDialog(optionPane,
"Aplikasi Pengukuran Jarak Objek. (C) Adelia Amityas", "Vesion",
JOptionPane.INFORMATION_MESSAGE);
            }
        };
                }
     return a;
}

class  pTimer extends Thread {
                int       d = 0;

        public synchronized void run() {
                        while(true){
                                try  {
                                        Thread t = Thread.currentThread();
                                        t.sleep(10);
                                        range =  Vision.getRange();

                                        if(GetCheck) {
                                                for(int h=0; h < vec.size(); h++) {
                                                        d = Integer.parseInt( (String)
vec.elementAt(h));

                                                }

                                        }
                                catch (InterruptedException ie) { return; }
                        }
        }
 }
}
```

ColorEffect.java

```
import java.awt.*;
import java.awt.image.*;
import javax.media.*;
import javax.media.control.*;
import javax.media.format.*;
import javax.media.protocol.*;
import javax.media.util.*;
```

```
public class ColorEffect extends VisionEffect {
  public static final float GAIN = 0.002293f;
  public static final float OFFSET = -0.07771f;
  public static final float H_CM = 5f;
  private static final int INIT_PIXEL_THRESHOLD = 16;
  private static final int LIGHT_THRESHOLD    = 192;
  public static final int MAX_PIXEL_THRESHOLD = 40;
  public static final int MIN_PIXEL_THRESHOLD = 0;
  public static final int PIXEL_THRESHOLD_INC = 4;
  public static final float INIT_PROPORTION = 0.25f;
  public static final float MAX_PROPORTION = 0.5f;
  public static final float MIN_PROPORTION = 0.05f;
  public static final float PROPORTION_INC = 0.05f;
  public int [] averageRed   = new int[Region.MAX_REGIONS];
  public int [] averageGreen = new int[Region.MAX_REGIONS];
  public int [] averageBlue  = new int[Region.MAX_REGIONS];
  public static int pixelThreshold      = INIT_PIXEL_THRESHOLD;
  public static float requiredProportion = INIT_PROPORTION;

  public float range = 0;

  /*
   * Detect colors and light in regions
   */
  public ColorEffect() {
    super();
  }

  /*
   * Look for colors in regions, and if found, call the associated color listener.
   * Also looks for overall light value and calls associated Light Listener.
   * Copies into to output.
   */
  public int process(Buffer inBuffer, Buffer outBuffer) {
    int outputDataLength = ((VideoFormat)outputFormat).getMaxDataLength();
    validateByteArraySize(outBuffer, outputDataLength);

    outBuffer.setLength(outputDataLength);
    outBuffer.setFormat(outputFormat);
    outBuffer.setFlags(inBuffer.getFlags());

    byte [] inData = (byte[]) inBuffer.getData();
    byte [] outData = (byte[]) outBuffer.getData();

    RGBFormat vfIn = (RGBFormat) inBuffer.getFormat();
    Dimension sizeIn = vfIn.getSize();

    int pixStrideIn = vfIn.getPixelStride();
    int lineStrideIn = vfIn.getLineStride();
```

```
if ( outData.length < sizeIn.width*sizeIn.height*3 ) {
  System.out.println("the buffer is not full");
  return BUFFER_PROCESSED_FAILED;
}

System.arraycopy(inData,0,outData,0,inData.length);

// Find the regions
Region [] regions = Vision.getRegions();

// Look for color listeners

// Examine each non-null region
int bestRegion = -1;
int bestListener = -1;
float bestProportion = -1;

for(int i=0;i<regions.length;i++) {
  if (regions[i] != null) {

    // Find the color listeners for this region
    ColorListener [] cl = regions[i].getColorListeners();

    // Continue if no listeners
    if (cl.length == 0) continue;

    // Get region size
    int rx = regions[i].getX();
    int ry = regions[i].getY();

    int width = regions[i].getWidth();
    int height = regions[i].getHeight();

                int maxtr = 0;
                int maxtg = 0;
                int maxtb = 0;

    // Look for the color associated with each listener
    for(int j=0;j<cl.length;j++) {

      int pixCount = 0, totalPixs = 0;
                  int xPos = 0, yPos =0;
      int aR = 0, aG = 0, aB = 0;

      for(int ii=ry; ii<ry+height; ii++) {
        for(int jj=rx; jj<rx+width; jj++) {

          int pos = ii*lineStrideIn + jj*pixStrideIn;
```

```
        int tr = inData[pos+2] & 0xFF;
        int tg = inData[pos+1] & 0xFF;
        int tb = inData[pos] & 0xFF;

                        if(Math.abs(tr) > Math.abs(maxtr) && Math.abs(tg) >
Math.abs(maxtg) && Math.abs(tb) > Math.abs(maxtb) ) {
                            if (tr >= LIGHT_THRESHOLD && tg >=
LIGHT_THRESHOLD && tr >= LIGHT_THRESHOLD) {
                                maxtr = tr;
                                maxtg = tg;
                                maxtb = tb;
                                yPos = ii; xPos = jj;
                            }
                        }
      }
     }

            if (xPos > 0 && yPos > 0) {
                // Calculate the distance for the laser dot from middle of
frame
                int pfc = 120 - yPos;

                // Calculate range in cm based on calibrated parameters
                range = (float) (H_CM / Math.tan(pfc * GAIN +
OFFSET));

                //System.out.println("xPos:" + xPos + ", yPos:" + yPos + "
,range:" + range);
                Font.println("jarak:" + range + "cm ,pfc:" + pfc + "\nx:" +
xPos + ",y:" + yPos, Font.FONT_6x11, xPos, yPos, (byte) 255,(byte) 0,(byte) 0,
outBuffer);
                } else
                 range = 0;
    }
   }
  }

      return BUFFER_PROCESSED_OK;
 }

 /*
  * Get the name of the Effect
  * @return "Color Effect"
  */
 public String getName() {
  return "Color Effect";
 }
}
```

ioPort.java

```
public class ioPort
{
        // declare native methods of 'jnpout32.dll'

        // output a value to a specified port address
   public native void Out32(int PortAddress, int data);

   // input a value from a specified port address
   public native short Inp32(int PortAddress);

   // load 'jnpout32.dll'
   static { System.loadLibrary("jnpout32");}
}
```

ColorListener.java

```
/**
 * Interface for color listeners
 */
public interface ColorListener {
 /**
  * Triggered when the color is detected in the region
  */
 void colorDetected(int r, int tc);
}
```

FlipControl.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.media.Control;


public class FlipControl implements Control, ActionListener {
 private Component component;
 private JButton button;
 private FlipEffect effect;
 private boolean debug;

 /**
  * Create the Motion Detection Control
  */
 public FlipControl(FlipEffect effect) {
  this.effect = effect;
 }

 /**
  * Return the visual component
```

```
   * @return the component containing the GUI controls
   **/
  public Component getControlComponent () {
    if (component == null) {
      button = new JButton("Toggle Flip");
      button.addActionListener(this);

      button.setToolTipText("Click to toggle horizontal flip on and off");

      Panel componentPanel = new Panel();
      componentPanel.setLayout(new BorderLayout());
      componentPanel.add("Center", button);
      componentPanel.invalidate();
      component = componentPanel;
    }
    return component;
  }

  /**
   * Toggle debug
   * @param e the action event (ignored)
   **/
  public void actionPerformed(ActionEvent e) {
    Object o = e.getSource();
    if (o == button) {
      effect.flip = !effect.flip;
    }
  }
}
```

FlipEffect.java

```
import javax.media.*;
import javax.media.format.*;
import java.awt.*;


public class FlipEffect extends VisionEffect {
  public boolean flip = false;

  /**
   * Create flip effect
   **/
  public FlipEffect() {
    super();
  }


  /**
   * Flip the data in each line
```

```
 **/
 public int process(Buffer inBuffer, Buffer outBuffer) {
  int outputDataLength = ((VideoFormat)outputFormat).getMaxDataLength();
  validateByteArraySize(outBuffer, outputDataLength);

  outBuffer.setLength(outputDataLength);
  outBuffer.setFormat(outputFormat);
  outBuffer.setFlags(inBuffer.getFlags());

  byte [] inData = (byte[]) inBuffer.getData();
  byte [] outData = (byte[]) outBuffer.getData();

  RGBFormat vfIn = (RGBFormat) inBuffer.getFormat();
  Dimension sizeIn = vfIn.getSize();

  int pixStrideIn = vfIn.getPixelStride();
  int lineStrideIn = vfIn.getLineStride();

  if ( outData.length < sizeIn.width*sizeIn.height*3 ) {
   System.out.println("the buffer is not full");
   return BUFFER_PROCESSED_FAILED;
  }

  int lines = inData.length/ lineStrideIn;
  int pixsPerLine = lineStrideIn/pixStrideIn;

  if (!flip) {
   System.arraycopy(inData,0,outData,0,inData.length);
   return BUFFER_PROCESSED_OK;
  }

  // Flip each line horizoontally

  byte [] buf = new byte[lineStrideIn];
  int pos = 0;

  for(int i=0;i<lines;i++) {
   for(int j=0;j<pixsPerLine;j++) {
    for(int k=0;k<3;k++)
     buf[lineStrideIn - (j*pixStrideIn) -3 + k] = inData[pos + (j*pixStrideIn) +
k];
   }
   System.arraycopy(buf,0,outData,pos,lineStrideIn);
   pos += lineStrideIn;
  }
  return BUFFER_PROCESSED_OK;
 }

 // methods for interface PlugIn
```

```
  /**
   * @return "Flip Effect"
   **/
  public String getName() {
    return "Flip Effect";
  }
  private Control[] controls;

  /**
   * Getter for array on one control for adjusing motion threshold and setting
debug.
   * @return an array of one ColorDetectionControl
   */
  public Object[] getControls() {
    if (controls == null) {
      controls = new Control[1];
      controls[0] = new FlipControl(this);
    }
    return (Object[])controls;
  }
}
```

LightListener.java

```
/**
 * Interface for light listeners
 */
public interface LightListener {
  /**
   * Triggered when bright light detected in the region
   */
  void lightDetected(int region);
}
```

MotionDetectionControl.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.media.Control;

public class MotionDetectionControl implements Control, ActionListener,
ChangeListener  {
  private Component component;
  private JButton button;
  private JSlider threshold;
  private JLabel label;
  private MotionDetectionEffect motion;
```

```
/**
 * Create the Motion Detection Control
 */
public MotionDetectionControl(MotionDetectionEffect motion) {
  this.motion = motion;
}

/**
 * Return the visual component
 * @return the component containing the GUI controls
 **/
public Component getControlComponent () {
  if (component == null) {
    label = new JLabel("Set Motion threshold:");
    button = new JButton("Motion Debug");
    button.addActionListener(this);

    button.setToolTipText("Click to turn debugging mode on/off");

    threshold = new JSlider(JSlider.HORIZONTAL,
                            0,
                            motion.THRESHOLD_MAX / 1000,
                            motion.THRESHOLD_INIT / 1000);

    threshold.setMajorTickSpacing(motion.THRESHOLD_INC / 1000);
    threshold.setPaintLabels(true);
    threshold.addChangeListener(this);

    Panel componentPanel = new Panel();
    componentPanel.setLayout(new BorderLayout());
    componentPanel.add("South", button);
    componentPanel.add("Center", threshold);
    componentPanel.add("North", label);
    componentPanel.invalidate();
    component = componentPanel;
  }
  return component;
}

/**
 * Toggle debug
 * @param e the action event (ignored)
 **/
public void actionPerformed(ActionEvent e) {
  Object o = e.getSource();
  if (o == button) {
    if (motion.debug == false) motion.debug = true;
    else motion.debug = false;
  }
```

```
  }

  /**
   * Set the threshold value
   * @param e the Changeevent (ignored)
   **/
  public void stateChanged(ChangeEvent e) {
    Object o = e.getSource();
    if (o == threshold) {
      motion.blob_threshold = threshold.getValue()*1000;
    }
  }
}
```

MotionDetectionEffect.java

```
import javax.media.*;
import javax.media.format.*;
import java.awt.*;
import java.io.*;


public class MotionDetectionEffect extends VisionEffect {

  private int [] blobCount = new int[Region.MAX_REGIONS];

  /**
   * Optimization. Anything above 0 turns it on. By default its
   * disabled.
   */
  public int OPTIMIZATION = 0;

  /**
   * Maximum threshold setting. Setting the threshold above this
   * means to get the motion detection to pass the frame you pretty
   * much have to full the whole frame with lots of motions (ie: drop
   * the camera)
   */
  public int THRESHOLD_MAX = 10000;

  /**
   * By what value you should increment.
   */
  public int THRESHOLD_INC = 1000;

  /**
   * The initial threshold setting.
   */
  public int THRESHOLD_INIT = 1000;
```

```
private byte[] refData;
private byte[] bwData;

private int avg_ref_intensity;
private int avg_img_intensity;

/**
 * The threshold for determing if the pixel at a certain location has
 * changed consideribly.
 */
public int threshold = 10;

/**
 * Our threshold for determinig if the input image has enough motion.
 *
 */
public int blob_threshold = THRESHOLD_INIT;

/**
 * Turn debugging on. Slows down the effect but shows how motion
 * detection effect works.
 */
public boolean debug = false;

/**
 * Initialize the Motion effect plugin.
 */
public MotionDetectionEffect() {
  super();
}

/**
 * Process the image, detecting motion in the regions
 * @param inBuffer the input Buffer
 * @param outBuffer the output Buffer
 * @return BUFFER_PROCESSED_OK or BUFFER_PROCESSED_FAILED
 */
public int process(Buffer inBuffer, Buffer outBuffer) {
  /*
    Optimization ideas:

    - first scale down the image.
    - convert the image to an int[][] array (instead of using byte[][])

    - then do all the calculation on int[][] array instead of
      masking the bits.

    Furthermore, only do the comparison every 5 frames instead of every frame.
  */
```

```
    // Validate and create the necessary buffers

    int outputDataLength = ((VideoFormat)outputFormat).getMaxDataLength();

    validateByteArraySize(outBuffer, outputDataLength);

    outBuffer.setLength(outputDataLength);
    outBuffer.setFormat(outputFormat);
    outBuffer.setFlags(inBuffer.getFlags());

    // Get the data portion of the buffers

    byte [] inData = (byte[]) inBuffer.getData();
    byte [] outData = (byte[]) outBuffer.getData();

    // Get the input format

    RGBFormat vfIn = (RGBFormat) inBuffer.getFormat();
    Dimension sizeIn = vfIn.getSize();

    // Get the stride lengths

    int pixStrideIn = vfIn.getPixelStride();
    int lineStrideIn = vfIn.getLineStride();

    int y, x;

    // Get the input demensions

    int width = sizeIn.width;
    int height = sizeIn.height;

    int r,g,b;
    int ip, op;
    byte result;
    int avg = 0;
    int refDataInt = 0;
    int inDataInt = 0;
    int correction;

    // If a snapshot has been requested, write the JPEG image to the selected file

    if (Vision.takeSnapshot()) {
      try {
        Vision.writeImage(Vision.snapshotFilename, inData, Vision.imageWidth,
Vision.imageHeight);
      } catch (Exception e) {
        System.out.println("Failed to take snapshot");
```

```
    } finally {
      Vision.setSnapshot(false);
    }
  }

  // If we have no reference data, create it, copy input to output
  // and don't attempt to detect motion

  if (refData == null) {
    refData = new byte[outputDataLength];
    bwData = new byte[outputDataLength];

    System.arraycopy (inData,0,refData,0,inData.length);
    System.arraycopy(inData,0,outData,0,inData.length);

    for (ip  = 0; ip < outputDataLength; ip++) {
        avg += (int) (refData[ip] & 0xFF);
    }

    avg_ref_intensity =  avg / outputDataLength;
    return BUFFER_PROCESSED_OK;
  }

  // Check the output buffer

  if ( outData.length < sizeIn.width*sizeIn.height*3 ) {
    System.out.println("the buffer is not full");
    return BUFFER_PROCESSED_FAILED;
  }

  // Calculate the average intensity

  for (ip  = 0; ip < outputDataLength; ip++) {
    avg += (int) (inData[ip] & 0xFF);
  }

  avg_img_intensity = avg / outputDataLength;

  // Calculate the correction factor as the absolute value of
  // the difference between the image and the reference integrity

  correction = (avg_ref_intensity < avg_img_intensity) ?
               avg_img_intensity - avg_ref_intensity :
               avg_ref_intensity - avg_img_intensity;

//     System.out.println(avg_ref_intensity + "; "+avg_img_intensity+" =
"+correction);
//
```

```
avg_ref_intensity = avg_img_intensity;
ip = op = 0;

/**
 * Compare the reference frame with the new frame.
 * We lite up only the pixels which changed, the rest are discarded (on the
 * b/w image - used for determing the motion)
 *
 */

for (int ii=0; ii< outputDataLength/pixStrideIn; ii++) {

  refDataInt = (int) refData[ip] & 0xFF;
  inDataInt = (int) inData[ip++] & 0xFF;
  r =  (refDataInt > inDataInt) ? refDataInt - inDataInt : inDataInt - refDataInt;

  refDataInt = (int) refData[ip] & 0xFF;
  inDataInt = (int) inData[ip++] & 0xFF;
  g =  (refDataInt > inDataInt) ? refDataInt - inDataInt : inDataInt - refDataInt;

  refDataInt = (int) refData[ip] & 0xFF;
  inDataInt = (int) inData[ip++] & 0xFF;
  b =  (refDataInt > inDataInt) ? refDataInt - inDataInt : inDataInt - refDataInt;

  // intensity normalization

  r -= (r < correction) ? r : correction;
  g -= (g < correction) ? g : correction;
  b -= (b < correction) ? b : correction;

  result = (byte)(java.lang.Math.sqrt((double)( (r*r) + (g*g) + (b*b) ) / 3.0));

  // black/white image now.

  if (result > (byte)threshold) {
    bwData[op++] = (byte)255;
    bwData[op++] = (byte)255;
    bwData[op++] = (byte)255;
  } else {
    bwData[op++] = (byte)result;
    bwData[op++] = (byte)result;
    bwData[op++] = (byte)result;
  }
}

// Now eliminate insignificant blobs and count how many
// there are in each region

Region [] regions = Vision.getRegions();
```

```
for(int i=0;i<regions.length;i++) blobCount[i] = 0;

// blob elimination

for (op = lineStrideIn + 3; op < outputDataLength - lineStrideIn-3; op+=3) {
 for (int i=0; i<1; i++) {
  if (((int)bwData[op+2] & 0xFF) < 255) break;
  if (((int)bwData[op+2-lineStrideIn] & 0xFF) < 255) break;
  if (((int)bwData[op+2+lineStrideIn] & 0xFF) < 255) break;
  if (((int)bwData[op+2-3] & 0xFF) < 255) break;
  if (((int)bwData[op+2+3] & 0xFF) < 255) break;
  if (((int)bwData[op+2-lineStrideIn + 3] & 0xFF) < 255) break;
  if (((int)bwData[op+2-lineStrideIn - 3] & 0xFF) < 255) break;
  if (((int)bwData[op+2+lineStrideIn - 3] & 0xFF) < 255) break;
  if (((int)bwData[op+2+lineStrideIn + 3] & 0xFF) < 255) break;
  bwData[op]  = (byte)0;
  bwData[op+1] = (byte)0;
  int yy = (op/lineStrideIn);
  int xx = (op % lineStrideIn)/pixStrideIn;

  for(int j=0;j<Region.MAX_REGIONS;j++) {
   if (regions[j] != null && regions[j].inRegion(xx, yy)) {
    blobCount[j]++;
   }
  }
 }
}

// Call Motion Listeners for regions whose blob count
// exceeds the current threshold

for (int i=0;i<Region.MAX_REGIONS && regions[i] != null;i++) {
 // System.out.println("Region " + i + ", count = " + blobCount[i]);

 if (blobCount[i] > blob_threshold) {

  // System.out.println("Motion detected in region " + i);

  // Call the motion listeners for this region

  MotionListener [] ml = regions[i].getMotionListeners();

  for(int j=0;j<ml.length;j++) ml[j].motionDetected(i+1);

  // If in debug mode split the screen into 4 and show the original
  // picture, the reference picture and the blobs detected

  if (debug) {
```

```
        sample_down(inData,outData, 0, 0,sizeIn.width, sizeIn.height,
        lineStrideIn, pixStrideIn);
        Font.println("original picture", Font.FONT_8x8, 0, 0,
                (byte)255,(byte)255,(byte)255, outBuffer);
           sample_down(refData,outData, 0, sizeIn.height/2,sizeIn.width,
                sizeIn.height, lineStrideIn, pixStrideIn);
           Font.println("reference picture", Font.FONT_8x8, 0,
                sizeIn.height , (byte)255,(byte)255,(byte)255, outBuffer);
        sample_down(bwData,outData, sizeIn.width/2, 0,sizeIn.width,
                sizeIn.height, lineStrideIn, pixStrideIn);
           Font.println("motion detection pic", Font.FONT_8x8,
                sizeIn.width/2, 0 , (byte)255,(byte)255,(byte)255, outBuffer);
         } else {

        // Otherwise copy the input to the output

        System.arraycopy(inData,0,outData,0,inData.length);
      }

      // Whether debug or not make the current picture the new reference picture

      System.arraycopy(inData,0,refData,0,inData.length);

      return BUFFER_PROCESSED_OK;
    }
   }

   // If no motion detected, just copy the input to the output

   System.arraycopy(inData,0,outData,0,inData.length);

   return BUFFER_PROCESSED_OK;
  }

  // methods for interface PlugIn

  /**
   * get the name of the effect
   * @return "Motion Detection Effect"
   **/
  public String getName() {
    return "Motion Detection Effect";
  }

  private Control[] controls;

  /**
   * Getter for array on one control for adjusing motion threshold and setting
debug.
```

```
   * @return an array of one MotionDetectionControl
   */
  public Object[] getControls() {
    if (controls == null) {
      controls = new Control[1];
      controls[0] = new MotionDetectionControl(this);
    }
    return (Object[])controls;
  }

  // Utility methods.

  /**
   * Reduce the data to one quarter size
   **/
  void sample_down(byte[] inData, byte[] outData, int X, int Y, int width,
                   int height, int lineStrideIn, int pixStrideIn) {
    int p1, p2, p3, p4, op,x,y;

    for ( y = 0; y < (height/2); y++) {
      p1 = (y * 2) * lineStrideIn ; // upper left cell
      p2 = p1 + pixStrideIn;        // upper right cell
      p3 = p1 + lineStrideIn;       // lower left cell
      p4 = p3 + pixStrideIn;        // lower right cell
      op = lineStrideIn * y + (lineStrideIn*Y) + (X*pixStrideIn);
      for ( int i =0; i< (width /2 );i++) {
        outData[op++] = (byte)(((int)(inData[p1++] & 0xFF) +
                        ((int)inData[p2++] & 0xFF)+ ((int)inData[p3++] & 0xFF) +
                         ((int)inData[p4++] & 0xFF))/4); // blue cells avg
        outData[op++] = (byte)(((int)(inData[p1++] & 0xFF) +
                         ((int)inData[p2++] & 0xFF)+ ((int)inData[p3++] & 0xFF) +
                          ((int)inData[p4++] & 0xFF))/4); // blue cells avg
        outData[op++] = (byte)(((int)(inData[p1++] & 0xFF) +
                        ((int)inData[p2++] & 0xFF)+ ((int)inData[p3++] & 0xFF) +
                         ((int)inData[p4++] & 0xFF))/4); // blue cells avg
        p1 += 3; p2 += 3; p3+= 3; p4 += 3;
      }
    }
  }
}
```

MotionListener.java

```
/**
 * Interface for motion listeners
 */
public interface MotionListener {
  /**
   * Triggered when motion detected in the region
   */
```

```
 void motionDetected(int r);
}
```

Recorder.java

```
import java.io.*;
import javax.media.*;
import javax.media.format.*;
import javax.media.protocol.*;

/**
 * Video recorder
 */
public class Recorder extends Thread implements ControllerListener {
 // private instance variables

 static private Processor p;
 private Object waitSync = new Object();
 private boolean stateTransitionOK = true;
 private boolean eom = false;
 private boolean failed = false;
 private String filename;
 private int millis;

 /**
  * Create the recorder.
  */
 public Recorder(String filename, int millis) {
  this.filename = filename;
  this.millis = millis;
 }

 public static void stopRecording() {
  p.stop();
 }

 /**
  *
  */
 public void run() {
  Vision.isRecording = true;

  DataSource ds = ((SourceCloneable) Vision.cds).createClone();

  Format formats[] = new Format[2];
  formats[0] = new AudioFormat(AudioFormat.LINEAR);
  formats[1] = new VideoFormat(VideoFormat.CINEPAK);
  FileTypeDescriptor outputType =
   new FileTypeDescriptor(FileTypeDescriptor.QUICKTIME);
```

```
    CaptureDeviceInfo di =
CaptureDeviceManager.getDevice(Vision.soundDevice);

    DataSource [] dss = new DataSource[2];

    dss[0] = ds;

    System.out.println("Creating Audio data source");

    try {
      dss[1] = Manager.createDataSource(di.getLocator());
    } catch (Exception e) {
      System.out.println("Failed to create Audio data source " + e.getMessage());
      System.exit(1);
    }

    System.out.println("Creating Merging data source");

    DataSource mds = null;

    try {
      mds = Manager.createMergingDataSource(dss);
    } catch (Exception e) {
      System.out.println("Failed to merge data sources " + e.getMessage());
      System.exit(-1);
    }

    try {
      p = Manager.createRealizedProcessor(new ProcessorModel(mds, formats,
outputType));
    } catch (Exception e) {
      System.err.println("Failed to create a processor from the given datasource: " +
e);
      System.exit(-1);
    }

    p.addControllerListener(this);

    DataSource source = p.getDataOutput();

    // create a File protocol MediaLocator with the location
    // of the file to which the video is to be written
    MediaLocator dest = new MediaLocator("file://" + filename + ".mov");

    // create a datasink to do the file writing & open the
    // sink to make sure we can write to it.
    DataSink filewriter = null;
    try {
      filewriter = Manager.createDataSink(source, dest);
```

```
    filewriter.open();
  } catch (Exception e) {
    System.out.println("Failed to create file writer");
    System.exit(-1);
  }

  // now start the filewriter
  try {
    filewriter.start();
  } catch (IOException e) {
    System.exit(-1);
  }

  //  and start the processor.

  p.start();

  waitForState(p.Started);

  System.out.println("Recording...");

  // If milliseconds is  zero, wait for stopRecording to be called

  if (millis > 0) {
    playToEndOfMedia(millis);
  } else {
    try {
      synchronized (waitSync) {
        waitSync.wait();
      }
    } catch (InterruptedException ioe) {}
  }

  p.close();

  Vision.isRecording = false;

  System.out.println("Finished");

  filewriter.close();

  // Restart the main processor, which is stopped as a side-effect

  Vision.p.start();
}

/**
 * Controller Listener.
 */
```

```
public void controllerUpdate(ControllerEvent evt) {

  System.out.println(this.getClass().getName()+evt);

  if (evt instanceof StopByRequestEvent) {
    synchronized (waitSync) {
        stateTransitionOK = true;
        waitSync.notifyAll();
    }
  } else if (evt instanceof ResourceUnavailableEvent) {
    synchronized (waitSync) {
        stateTransitionOK = false;
      failed = true;
        waitSync.notifyAll();
    }
  } else if (evt instanceof EndOfMediaEvent) {
    eom = true;
  }
}

/**
 * Block until the processor has transitioned to the given state.
 * Return false if the transition failed.
 */
boolean waitForState(int state) {
  synchronized (waitSync) {
    try {
        while (p.getState() != state && stateTransitionOK)
       waitSync.wait();
    } catch (Exception e) {}
  }
  return stateTransitionOK;
}

public boolean playToEndOfMedia(int timeOutMillis) {
  long startTime = System.currentTimeMillis();
  eom = false;
  failed = false;
  synchronized (this) {
    while (!eom && !failed) {
      try {
        wait(timeOutMillis);
      } catch (InterruptedException ie) {
    }
    if (System.currentTimeMillis() - startTime > timeOutMillis)
      break;
    }
  }
  return eom && !failed;
```

```
  }
}
```

Region.java

```
public class Region {

  public static int MAX_REGIONS = 32;

  private int x, y;
  private int w, h;

  private static final int MAX_MOTION_LISTENERS = 32;
  private static final int MAX_COLOR_LISTENERS = 32;
  private static final int MAX_LIGHT_LISTENERS = 32;

  private MotionListener [] motionListeners= new
MotionListener[MAX_MOTION_LISTENERS];
  private int numMotionListeners = 0;

  private ColorListener [] colorListeners= new
ColorListener[MAX_COLOR_LISTENERS];
  private int numColorListeners = 0;
  private int [] colors = new int[MAX_COLOR_LISTENERS];

  private LightListener [] lightListeners= new
LightListener[MAX_LIGHT_LISTENERS];
  private int numLightListeners = 0;

  /**
   * Create a region
   *
   * @param x the x coordinate of the botton left corner
   * @param y the y coordinate of the botton left corner
   * @param w the width of the region
   * @param h the height of the region
   */
  public Region(int x, int y, int w, int h) {
    this.x = x;
    this.y = y;
    this.w = w;
    this.h = h;
  }

  /**
   * Get the X coordinate of the bottom left corner
   */
  public int getX() {
    return x;
  }
```

```
/**
 * Get the Y coordinate of the bottom left corner
 */
public int getY() {
  return y;
}

/**
 * Get the width of the region
 * @return the width of the region
 */
public int getWidth() {
  return w;
}

/**
 * Get the height of the region
 * @return the height of the region
 */
public int getHeight() {
  return h;
}

/**
 * Test if point is in the region
 * @param tx test x coordinate
 * @param ty test y coordinate
 */
public boolean inRegion(int tx, int ty) {
  return (tx >=x && ty >= y && tx <=x+w && ty <= y+h);
}

/**
 * Add a motion listener
 * @param ml the listener to add
 */
public void addMotionListener(MotionListener ml) {
  motionListeners[numMotionListeners++] = ml;
}

/**
 * Add a color listener
 * @param cl the listener to add
 * @param color the color to look for
 */
public void addColorListener(ColorListener cl, int color) {
  colors[numColorListeners] = color;
  colorListeners[numColorListeners++] = cl;
```

```
  }

  /**
   * Add a light listener
   * @param ll the listener to add
   */
  public void addLightListener(LightListener ll) {
    lightListeners[numLightListeners++] = ll;
  }

  /**
   * Return the array of motion listeners
   * @return the motion listener array
   */
  public MotionListener [] getMotionListeners() {
    MotionListener [] ml = new MotionListener[numMotionListeners];
    for(int i=0;i<numMotionListeners;i++) ml[i] = motionListeners[i];
    return ml;
  }

  /**
   * Return the array of color listeners
   * @return the color listener array
   */
  public ColorListener [] getColorListeners() {
    ColorListener [] cl = new ColorListener[numColorListeners];
    for(int i=0;i<numColorListeners;i++) cl[i] = colorListeners[i];
    return cl;
  }

  /**
   * Return the array of colors corresponding to the color listeners
   * @return the array of colors
   */
  public int [] getColors() {
    return colors;
  }

  /**
   * Return the array of light listeners
   * @return the light listener array
   */
  public LightListener [] getLightListeners() {
    LightListener [] ll = new LightListener[numLightListeners];
    for(int i=0;i<numLightListeners;i++) ll[i] = lightListeners[i];
    return ll;
  }
}
```

RegionControl.java

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.event.*;
import javax.media.Control;

public class RegionControl implements Control, ActionListener {
  private Component component;
  private JButton button;
  private RegionEffect effect;
  private boolean debug;

  /**
   * Create the Region Control
   */
  public RegionControl(RegionEffect effect) {
    this.effect = effect;
  }

  /**
   * Return the visual component
   * @return the component containing the GUI controls
   **/
  public Component getControlComponent () {
    if (component == null) {
      button = new JButton("Toggle Show Regions");
      button.addActionListener(this);

      button.setToolTipText("Click to toggle show regions on and off");

      Panel componentPanel = new Panel();
      componentPanel.setLayout(new BorderLayout());
      componentPanel.add("Center", button);
      componentPanel.invalidate();
      component = componentPanel;
    }
    return component;
  }

  /**
   * Toggle debug
   * @param e the action event (ignored)
   **/
  public void actionPerformed(ActionEvent e) {
    Object o = e.getSource();
    if (o == button) {
      effect.show = !effect.show;
```

```
    }
  }
}
```

RegionEffect.java

```
import javax.media.*;
import javax.media.format.*;
import java.awt.*;

/*
 * Supports overlaying Region information over the video stream
 */
public class RegionEffect extends VisionEffect {
  public boolean show = true;

  public RegionEffect() {
    super();
  }

  public int process(Buffer inBuffer, Buffer outBuffer) {
    int outputDataLength = ((VideoFormat)outputFormat).getMaxDataLength();
    validateByteArraySize(outBuffer, outputDataLength);
    Region [] regions = Vision.getRegions();

    outBuffer.setLength(outputDataLength);
    outBuffer.setFormat(outputFormat);
    outBuffer.setFlags(inBuffer.getFlags());

    byte [] inData = (byte[]) inBuffer.getData();
    byte [] outData = (byte[]) outBuffer.getData();

    RGBFormat vfIn = (RGBFormat) inBuffer.getFormat();
    Dimension sizeIn = vfIn.getSize();

    int pixStrideIn = vfIn.getPixelStride();
    int lineStrideIn = vfIn.getLineStride();

    if ( outData.length < sizeIn.width*sizeIn.height*3 ) {
      System.out.println("the buffer is not full");
      return BUFFER_PROCESSED_FAILED;
    }

    System.arraycopy(inData,0,outData,0,inData.length);

    if (!show) return BUFFER_PROCESSED_OK;

    for(int i=0;i<regions.length;i++) {
      if (regions[i] != null) {
        int rx = regions[i].getX();
```

```
      int ry = regions[i].getY();
      int rw = regions[i].getWidth();
      int rh = regions[i].getHeight();

      int offx = rx * pixStrideIn;
      int offy = ry * lineStrideIn;

      // System.out.println("rw = " + rw + ", rh = " + rh);

      int width = rw * pixStrideIn;
      int height = (rh-1) * lineStrideIn;

      // Draw left side
      for(int j=0;j<rh;j++) {
        outData[offx + offy + (j * lineStrideIn)] = (byte) 0;
        outData[offx + offy + (j * lineStrideIn)+1] = (byte) 0;
        outData[offx + offy + (j * lineStrideIn)+2] = (byte) 255;
      }

      // Draw right side
      for(int j=0;j<rh;j++) {
        outData[offx + width + offy + (j * lineStrideIn)-3] = (byte) 0;
        outData[offx + width + offy + (j * lineStrideIn)-2] = (byte) 0;
        outData[offx + width + offy + (j * lineStrideIn)-1] = (byte) 255;
      }

      // Draw the bottom
      for(int j=0;j<rw;j++) {
        outData[offx + offy + (j * pixStrideIn)] = (byte) 0;
        outData[offx + offy + (j * pixStrideIn) + 1] = (byte) 0;
        outData[offx + offy + (j * pixStrideIn) + 2] = (byte) 255;
      }

      // Draw the top
      for(int j=0;j<rw;j++) {
        outData[offx + offy + height + (j * pixStrideIn)] = (byte) 0;
        outData[offx + offy + height + (j * pixStrideIn) + 1] = (byte) 0;
        outData[offx + offy + height + (j * pixStrideIn) + 2] = (byte) 255;
      }

      // Write region number
      Font.println("" + (i+1), Font.FONT_6x11, rx + 5, ry + 10, (byte) 255,(byte)
0,(byte) 0, outBuffer);
    }
   }

   return BUFFER_PROCESSED_OK;
  }
```

```
  // methods for interface PlugIn
  public String getName() {
    return "Region Effect";
  }

  private Control[] controls;

  /**
   * Getter for array on one control for adjusing motion threshold and setting
debug.
   * @return an array of one MotionDetectionControl
   */
  public Object[] getControls() {
    if (controls == null) {
      controls = new Control[1];
      controls[0] = new RegionControl(this);
    }
    return (Object[])controls;
  }
}
```

StringUtil.java

```
import java.util.*;
import java.util.MissingResourceException;
import java.util.ResourceBundle;
import java.lang.StringBuffer;

public class  StringUtil
{
        private static final String BUNDLE_NAME = "db"; //$NON-NLS-1$
        private static final ResourceBundle RESOURCE_BUNDLE =
ResourceBundle.getBundle(BUNDLE_NAME);

        public static String getString(String key) {
                try {
                        return RESOURCE_BUNDLE.getString(key);
                } catch (MissingResourceException e) {
                        //return '!' + key + '!';
                        return null;
                }
        }


        public String replace(String str, String pattern, String replace) {
      int s = 0;
      int e = 0;
      String xresult ="";
                StringBuffer result = new StringBuffer();
                try{
```

```
while ((e = str.indexOf(pattern, s)) >= 0) {
   result.append(str.substring(s, e));
                    result.append(replace);
   s = e+pattern.length();
            }
result.append(str.substring(s));
xresult = result.toString();
            }catch(Exception ss){
                    ss.printStackTrace();
                    xresult ="";

            }
            return xresult;
   }

public boolean testIntegerValue(String x){
     boolean xxx=false;
     int dz=1;
            for(int g=0;g<x.length();g++){
                    try{

                                Integer.parseInt(x.substring(g,dz));
                 xxx=true;
                    }catch(Exception d){
              xxx=false;
              break;
                    }
                    ++dz;
            }
      return xxx;
}

public String  removeASCIINO(String abc, int ASCII){


  char a,b;
  StringBuffer sb = new StringBuffer();
  for(int g=0; g<abc.length();g++){

    a=abc.charAt(g);
    int x = (int)a;
     if(x!=ASCII){
     b=abc.charAt(g);
     sb.append(b);

     }
  }
return sb.toString();
}
```

```
}
```

Vision.java

```
import java.awt.*;
import java.io.*;
import java.util.*;
import java.awt.event.*;
import javax.media.*;
import javax.media.control.TrackControl;
import javax.media.format.*;
import javax.media.protocol.*;
import javax.media.datasink.*;
import javax.media.control.*;
import javax.sound.sampled.*;
import com.sun.image.codec.jpeg.*;
import java.awt.image.*;

/**
 * Java version of Vision Command.
 */
public class Vision extends Frame implements ControllerListener {

  // package protected fields

  static int imageWidth = -1, imageHeight = -1;
  static float frameRate = 15;
  static String snapshotFilename;
  static Processor p;
  static DataSource cds;
  static String cameraDevice, soundDevice;
  static boolean isRecording = false;
  static boolean captureColor = false;
  static Vision visionFrame;
  static ColorEffect colorEffect = new ColorEffect();
  static FlipEffect flipEffect = new FlipEffect();
  static RegionEffect regionEffect = new RegionEffect();
  static MotionDetectionEffect motionDetectionEffect = new
MotionDetectionEffect();

  // private instance variables

  private Object waitSync = new Object();
  private boolean stateTransitionOK = true;
  private static Properties videoProperties;
  private final static String DEFAULT_VIDEO_DEV_NAME =
    "vfw:Logitech USB Video Camera:0";
  private final static String DEFAULT_SOUND_DEV_NAME =
    "DirectSoundCapture";
  private static Region [] regions = new Region[Region.MAX_REGIONS];
```

```
  private static boolean takeSnapshot = false;

  /**
   * Create the viewer frame with a title.
   * @param title the title for the viewer
   */
  public Vision(String title) {
    super(title);
  }

  /**
   * Get the viewer frame. Allows extra controls to be added.
   * @return the frame
   */
  public static Frame getFrame() {
    return (Frame) visionFrame;
  }

  /**
   * Given a datasource, create a processor and use that processor
   * as a player to playback the media.
   *
   * During the processor's Configured state, the FlipEffect,
MotionDetectionEffect
   * ColorEffect and RegionEffect are inserted into the video track.
   *
   */
  public boolean open(DataSource tds) {

    // Create a cloneable data source so that it cab be cloned for video capture

    cds = Manager.createCloneableDataSource(tds);

    // Create a processor for the capture device

    try {
      p = Manager.createProcessor(cds);
    } catch (Exception e) {
      System.err.println("Failed to create a processor from the given datasource: " +
e);
      return false;
    }

    // Make this Vision instance thr controller

    p.addControllerListener(this);

    // Put the Processor into configured state.
```

```
p.configure();
if (!waitForState(p.Configured)) {
  System.err.println("Failed to configure the processor.");
  return false;
}

// So I can use it as a player.

p.setContentDescriptor(null);

// Obtain the track controls.

TrackControl tc[] = p.getTrackControls();
if (tc == null) {
  System.err.println("Failed to obtain track controls from the processor.");
  return false;
}

// Search for the track control for the video track.

TrackControl videoTrack = null;

for (int i = 0; i < tc.length; i++) {
  if (tc[i].getFormat() instanceof VideoFormat) {
      videoTrack = tc[i];
      break;
  }
}

if (videoTrack == null) {
  System.err.println("The input media does not contain a video track.");
  return false;
}

// Instantiate and set the frame access codec to the data flow path.

try {
  Codec codec[] = {  flipEffect,
                     motionDetectionEffect,
                     regionEffect,
                     colorEffect};
  videoTrack.setCodecChain(codec);
} catch (UnsupportedPlugInException e) {
  System.err.println("The processor does not support effects.");
  return false;
}

// Realize the processor.
```

```
  p.prefetch();
  if (!waitForState(p.Prefetched)) {
    System.err.println("Failed to realize the processor.");
    return false;
  }

  // Layout the components

  setLayout(new BorderLayout());

  // Display the visual & control component if they exist.

  Component cc, vc;

  if ((vc = p.getVisualComponent()) != null) {
    add("Center", vc);
  }

  if ((cc = p.getControlPanelComponent()) != null) {
    add("South", cc);
  }

  // Start the processor.

  p.start();

  // Show the frame

  setVisible(true);

  // Detect the window close event

  addWindowListener(new WindowAdapter() {
    public void windowClosing(WindowEvent we) {
        p.close();
        System.exit(0);
    }
  });

  // If we get here, its worked

  return true;
}

/**
 * Close Video viewer
 */
public static void stopViewer() {
  visionFrame.setVisible(false);
```

```
  p.close();
}

public void addNotify() {
  super.addNotify();
  pack();
}

/**
 * Block until the processor has transitioned to the given state.
 * Return false if the transition failed.
 */
boolean waitForState(int state) {
  synchronized (waitSync) {
    try {
        while (p.getState() != state && stateTransitionOK)
       waitSync.wait();
    } catch (Exception e) {}
  }
  return stateTransitionOK;
}

/**
 * Controller Listener.
 */
public void controllerUpdate(ControllerEvent evt) {

  // System.out.println(this.getClass().getName()+evt);

  if (evt instanceof ConfigureCompleteEvent ||
      evt instanceof RealizeCompleteEvent ||
      evt instanceof PrefetchCompleteEvent) {
    synchronized (waitSync) {
        stateTransitionOK = true;
        waitSync.notifyAll();
    }
  } else if (evt instanceof ResourceUnavailableEvent) {
    synchronized (waitSync) {
        stateTransitionOK = false;
        waitSync.notifyAll();
    }
  } else if (evt instanceof EndOfMediaEvent) {
    p.close();
    System.out.println("End of Media");
    System.exit(0);
  }
}

/**
```

```
   * Start the video viewer frame
   */
  public static void startViewer(String title) {

    /*  We use a properties file to allow the user to define what kind of
     *  camera and resolution they want to use for the vision input
     */

    String videoPropFile =
      System.getProperty("video.properties", "video.properties");

    // Open the video properties file

    try {
      FileInputStream fis = new FileInputStream(new File(videoPropFile));
      videoProperties = new Properties();
      videoProperties.load(fis);
    } catch (IOException ioe) {
      System.err.println("Failed to read property file");
      System.exit(1);
    }

    // Set the camera device

    cameraDevice =
      videoProperties.getProperty("video-device-name",
DEFAULT_VIDEO_DEV_NAME);

    // Set the sound device

    soundDevice =
      videoProperties.getProperty("sound-device-name",
DEFAULT_SOUND_DEV_NAME);

    // If not set by the API, set the image width

    if (imageWidth < 0)
      imageWidth =
        Integer.parseInt(videoProperties.getProperty("resolution-x", "320"));

    // If not set by the API, set the image height

    if (imageHeight < 0)
      imageHeight =
        Integer.parseInt(videoProperties.getProperty("resolution-y", "320"));

    // System.out.println("Searching for [" + cameraDevice + "]");

    /*  Try to get the CaptureDevice that matches the name supplied by the
```

```
     * user
     */
    CaptureDeviceInfo device =
CaptureDeviceManager.getDevice(cameraDevice);

    if (device == null) {
      System.out.println("No device found [ " + cameraDevice + "]");
      System.exit(1);
    }

    // Create a media locator from the device

    MediaLocator ml = device.getLocator();

    // Create a data source from the media locator

    DataSource tds = null;

    try{
      tds = Manager.createDataSource(ml);
    } catch (Exception e) {
      System.err.println("Failed to create a datasource");
      System.exit(1);
    }

    // Set the format on the relevant format control for the device
    // This specifies the image height, width, fram rate etc.
    // These need to be supported by the device.
    // Only 24-bit color is supported

    FormatControl [] formatControls = ((CaptureDevice) tds).getFormatControls();

    // Create the required format. Seem to need the extra 4 bytes

    Format format = new RGBFormat(new Dimension(imageWidth, imageHeight),
                          (imageWidth * imageHeight * 3) + 4,
                          Format.byteArray,
                          frameRate,
                          24,
                          3, 2, 1,
                          3, Format.NOT_SPECIFIED,
                          Format.TRUE,
                          Format.NOT_SPECIFIED);

    if ( formatControls == null || formatControls.length == 0 ) {
      System.out.println("No format controls");
      System.exit(1);
    }
```

```
  // Only one format control is expected

  for ( int i = 0; i < formatControls.length; i++ ) {
   if ( formatControls[i] == null ) continue;

   formatControls[i].setFormat(format);
  }

  // Create the frame
  visionFrame = new Vision(title);

  // Start the video viewer
  if (!visionFrame.open(tds)) System.exit(1);
 }

 /**
  * Play an audio file
  * @param fileName the audio file to play
  */
 public static void playSound(String fileName) {

  // Create an Audio Stream from the file

  try {
    AudioInputStream stream = AudioSystem.getAudioInputStream(new
File(fileName));

    // Get the Audio format
    javax.sound.sampled.AudioFormat format = stream.getFormat();

    // Create a DataLine Info object

    DataLine.Info info = new DataLine.Info(
      Clip.class, stream.getFormat(),
((int)stream.getFrameLength()*format.getFrameSize()));

    // Create the audio clip

    Clip clip = (Clip) AudioSystem.getLine(info);

    // Load the audio clip - does not return until the audio file is completely
loaded

    clip.open(stream);

    // Start playing

    clip.start();
```

```
  } catch (IOException e) {
    System.err.println("Audio file not found");
  } catch (LineUnavailableException e) {
    System.err.println("Could not play the audio file");
  } catch (UnsupportedAudioFileException e) {
    System.err.println("Unsupported Audio File Format");
  }
}

/**
 * Add a rectangular region
 * @param region the region number
 * @param x the x co-ordinate of the region bottom left corner
 * @param y the y co-ordinate of the region bottom left corner
 * @param width the width of the region
 * @param height the height of the region
 */
public static void addRectRegion(int region, int x, int y, int width, int height) {
  regions[region-1] = new Region(x, y, width, height);
}

/**
 * Get the array of regions
 * @return the array of regions
 */
public static Region [] getRegions() {
  return regions;
}

/**
 * Set the frame rate
 * @param rate the required frame rate
 */
public static void setFrameRate(float rate) {
  frameRate = rate;
}

/**
 * Set the size of the video viewer image
 * @param width the required image width
 * @param height the required image height
 */
public static void setImageSize(int width, int height) {
  imageWidth = width;
  imageHeight = height;
}

/**
 * Add a Motion Listener for the region
```

```
 * @param region the region
 * @param ml the Motion Listener
 */
public static void addMotionListener(int region, MotionListener ml) {
  regions[region-1].addMotionListener(ml);
}

/**
 * Add a Color Listener for the region
 * @param region the region
 * @param cl the Color Listener
 * @param color the color to listen for
 */
public static void addColorListener(int region, ColorListener cl, int color) {
  regions[region-1].addColorListener(cl, color);
}

/**
 * Add a Light Listener for the region
 * @param region the region
 * @param ll the Light Listener
 */
public static void addLightListener(int region, LightListener ll) {
  regions[region-1].addLightListener(ll);
}

/**
 * Return the state of the snapshot flag
 * @return true if a snapshot is required, else false
 */
static boolean takeSnapshot() {
  return takeSnapshot;
}

/**
 * Take a snapshot
 * @param filename the JPG file to write the snapshop to
 */
public static void snapshot(String filename) {
  snapshotFilename = filename;
  takeSnapshot = true;
}

/**
 * Set or unset the take snapshot flag
 * @param sanap true if snapshot is required, else false
 */
static void setSnapshot(boolean snap) {
  takeSnapshot = snap;
```

```
}

/**
 * Write to <code>fn</code> file the <code>data</code> using the
 * <code>width, height</code> variables. Data is assumed to be 8bit RGB.
 * A JPEG format file is written.
 *
 * @param fn the filename
 * @param data the data to write
 * @param width the width of the image
 * @param height the height of the image
 * @throws  FileNotFoundException  if the directory/image specified is wrong
 * @throws  IOException  if there are problems reading the file.
 */
public static void writeImage(String fn, byte[] data, int width, int height)
throws FileNotFoundException, IOException {

  // Open the file

  FileOutputStream fOut = new FileOutputStream(fn);

  // Create a JPG encoder for the file

  JPEGImageEncoder jpeg_encode = JPEGCodec.createJPEGEncoder(fOut);

  // Reformat the data to an  array on int

  int ints[] = new int[data.length/3];
  int k = 0;
  for (int i = height-1; i > 0;i--) {
    for (int j=0;j<width;j++) {
      ints[k++] = 255 << 24 |
        (int) (data[i*width*3 + j*3 + 2] & 0xff) << 16 |
        (int) (data[i*width*3 + j*3 + 1] & 0xff) << 8 |
        (int) (data[i*width*3 + j*3] & 0xff);
    }
  }

  // Create a buffered image

  BufferedImage image = new BufferedImage (width,
height,BufferedImage.TYPE_INT_RGB);
  image.setRGB(0,0,width,height,ints,0,width);

  // Encode the image and close the output file

  jpeg_encode.encode(image);
  fOut.close();
}
```

```
/**
 * Start the video recorder
 * @param fileName the file to write the video to
 * @param millis the number of milliseconds to record for.
 * 0 means record until stopRecoder() is called.
 */
public static void startRecorder(String fileName, int millis) {
  // Create the recorder

  // Start the video viewer
}

/**
 * Test is recording is in progress
 * @return true if recoding, else false
 */
public static boolean isRecording() {
  return isRecording;
}

/*
 * Stop recording
 */

/**
 * Get the average red value for the region
 * @param region the region
 * @return the average red value
 */
public static int getAvgRed(int region) {
  return colorEffect.averageRed[region-1];
}

/**
 * Get the average green value for the region
 * @param region the region
 * @return the average green value
 */
public static int getAvgGreen(int region) {
  return colorEffect.averageGreen[region-1];
}

/**
 * Get the average blue value for the region
 * @param region the region
 * @return the average blue value
 */
public static int getAvgBlue(int region) {
```

```
    return colorEffect.averageBlue[region-1];
  }

  /**
   * Get the average RGB value for the region
   * @param region the region
   * @return the average RGB value
   */
  public static int getAvgRGB(int region) {
    return ((new Color(getAvgRed(region), getAvgGreen(region),
getAvgBlue(region))).getRGB()) & 0xffffff;
  }


  /**
   * Flip the image in the image viewer horizontally
   * @param true to flip, else false
   */
  public static void flipHorizontal(boolean flip) {
    flipEffect.flip = flip;
  }

  /**
   * Get range
   * @return the range value
   */
  public static float getRange() {
        return colorEffect.range;
  }
}
```

VisionEffect.java

```
import javax.media.*;
import javax.media.format.*;
import java.awt.*;

/**
 * Abstract Effect that specific Vision Effects inherit from
 * @author Lawrie Griffiths
 */
public abstract class VisionEffect implements Effect {

  protected Format inputFormat;
  protected Format outputFormat;
  protected Format[] inputFormats;
  protected Format[] outputFormats;

  /**
   * Create the Effect. Only 24-bit color is supported.
```

```
 **/
public VisionEffect() {
  inputFormats = new Format[] {
                new RGBFormat(null,
                        Format.NOT_SPECIFIED,
                        Format.byteArray,
                        Format.NOT_SPECIFIED,
                        24,
                        3, 2, 1,
                        3, Format.NOT_SPECIFIED,
                        Format.TRUE,
                        Format.NOT_SPECIFIED)
                };

  outputFormats = new Format[] {
                 new RGBFormat(null,
                        Format.NOT_SPECIFIED,
                        Format.byteArray,
                        Format.NOT_SPECIFIED,
                        24,
                        3, 2, 1,
                        3, Format.NOT_SPECIFIED,
                        Format.TRUE,
                        Format.NOT_SPECIFIED)
                };
}

// Methods for interface codec

/**
 * Get the supported input formats
 * @return the supported input formats
 */
public Format[] getSupportedInputFormats() {
  return inputFormats;
}

/**
 * Get the supported output formats that matches the input format
 * @return the supported output formats
 */
public Format [] getSupportedOutputFormats(Format input) {
  if (input == null) return outputFormats;

  if (matches(input, inputFormats) != null) {
    return new Format[] { outputFormats[0].intersects(input) };
  } else {
    return new Format[0];
  }
```

```
}

/**
 * Set the input format.
 * @param input the required input format
 * @return the input format
 */
public Format setInputFormat(Format input) {
  inputFormat = input;
  return input;
}

/**
 * Set the output format.
 * Ensures size and line stride are in the expected 24-bit 3-byte stride format.
 * @param output the output format
 * @return the output format
 */
public Format setOutputFormat(Format output) {
  if (output == null || matches(output, outputFormats) == null)
    return null;
  RGBFormat incoming = (RGBFormat) output;

  Dimension size = incoming.getSize();
  int maxDataLength = incoming.getMaxDataLength();
  int lineStride = incoming.getLineStride();
  float frameRate = incoming.getFrameRate();
  int endian = incoming.getEndian();

  if (size == null) return null;

  if (maxDataLength < size.width * size.height * 3)
    maxDataLength = size.width * size.height * 3;

  if (lineStride < size.width * 3) lineStride = size.width * 3;

  outputFormat = outputFormats[0].intersects(new RGBFormat(size,
                                      maxDataLength,
                                      null,
                                      frameRate,
                                      Format.NOT_SPECIFIED,
                                      Format.NOT_SPECIFIED,
                                      Format.NOT_SPECIFIED,
                                      Format.NOT_SPECIFIED,
                                      Format.NOT_SPECIFIED,
                                      lineStride,
                                      Format.NOT_SPECIFIED,
                                      Format.NOT_SPECIFIED));
```

```
  return outputFormat;
}

/**
 * Does nothing
 **/
public void open() {
}

/**
 * Does nothing
 **/
public void close() {
}

/**
 * Does nothing
 **/
public void reset() {
}

// methods for interface javax.media.Controls

/**
 * Returns null
 * @return null
 **/
public Object getControl(String controlType) {
  return null;
}

/**
 * Returns null
 * @return null
 **/

public Object[] getControls() {
  return null;
}

// Utility methods.

/**
 * Select the first output format that matches the input format.
 * @return first matching output format ot null if none match
 */
protected Format matches(Format in, Format outs[]) {
  for (int i = 0; i < outs.length; i++) {
    if (in.matches(outs[i]))
```

```
      return outs[i];
    }
    return null;
  }

  /**
   * Validate that the Buffer conforms to the expected format, and create a new
   * byte array if not.
   */
  protected byte[] validateByteArraySize(Buffer buffer,int newSize) {
    Object objectArray=buffer.getData();
    byte[] typedArray;

    if (objectArray instanceof byte[]) {    // is correct type AND not null
      typedArray=(byte[])objectArray;
      if (typedArray.length >= newSize ) { // is sufficient capacity
        return typedArray;
      }

      byte[] tempArray=new byte[newSize];  // re-alloc array
      System.arraycopy(typedArray,0,tempArray,0,typedArray.length);
      typedArray = tempArray;
    } else {
      typedArray = new byte[newSize];
    }

    buffer.setData(typedArray);
    return typedArray;
  }
}
```