APLIKASI AKSES *MOUSE* MENGGUNAKAN *LASER POINTER*

SKRIPSI

KONSENTRASI REKAYASA KOMPUTER

Diajukan untuk memenuhi persyaratan

memperoleh gelar Sarjana Teknik

Disusun Oleh :

GALLANT HENDRAYANA PUTRA

NIM. 0710633038 - 63

KEMENTERIAN PENDIDIKAN DAN KEBUDAYAAN

UNIVERSITAS BRAWIJAYA

FAKULTAS TEKNIK

MALANG

2012

LEMBAR PERSETUJUAN

APLIKASI AKSES *MOUSE* MENGGUNAKAN *LASER POINTER*

SKRIPSI

KONSENTRASI REKAYASA KOMPUTER

Diajukan untuk memenuhi persyaratan
memperoleh gelar sarjana teknik

Disusun Oleh
GALLANT HENDRAYANA PUTRA
NIM. 0710633038 - 63

Telah diperiksa dan disetujui oleh
Dosen Pembimbing

Dosen Pembimbing I	**Dosen Pembimbing II**
Adharul Muttaqin, ST., MT.	**Ir. Muhammad Aswin,MT.**
NIP.19760121 200501 1 001	**NIP. 19640626 199002 1 001**

LEMBAR PENGESAHAN

APLIKASI AKSES *MOUSE* MENGGUNAKAN *LASER POINTER*

SKRIPSI

JURUSAN TEKNIK ELEKTRO

Diajukan untuk memenuhi persyaratan

Memperoleh gelar SarjanaTeknik

Disusun Oleh :

GALLANT HENDRAYANA PUTRA

NIM. 0710633038 - 63

Skripsi ini telah diuji dan dinyatakan lulus pada

tanggal 20 Juli 2012

DOSEN PENGUJI

Ali Mustofa, ST., MT.

NIP. 19710601 200003 1 001

A. Abdul Razak, ST.,MT.,M.Eng

R.Arief Setyawan, ST., MT.

NIP. 19750819 199903 1 001

Mengetahui

Ketua Jurusan Teknik Elektro

Sholeh Hadi Pramono, Dr., Ir., MS.

NIP. 19580728 198701 1 001

PENGANTAR

Dengan mengucapkan Alhamdulillahi Rabbil'alamin, puji syukur kepada Allah SWT yang telah melimpahkan rahmat, ridho, taufik dan hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Aplikasi Akses Mouse Menggunakan *Laser* Pointer" sebagai salah satu syarat yang diajukan untuk memperoleh gelar Sarjana Teknik di Jurusan Teknik Elektro Fakultas Teknik Universitas Brawijaya Malang.

Penulis menyadari bahwa penulisan skripsi ini tidak akan terselesaikan tanpa adanya bantuan dari berbagai pihak karena itu ucapan terima kasih penulis sampaikan kepada semua pihak yang telah dengan rela meluangkan waktunya untuk membantu penulis dalam proses penulisan skripsi dan juga dengan ikhlas memberikan bimbingan, arahan, masukan, dan semangat sehingga penulis dapat menyelesaikan skripsi ini. Oleh karena itu pada kesempatan ini, penulis mengucapkan terima kasih kepada:

1. Bapak Dr. Ir. Sholeh Hadi Pramono., MS. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
2. Bapak M. Azis Muslim, S.T., M.T., Ph.D selaku Sekretaris Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
3. Bapak Moch. Rif'an. ST., MT. selaku Ketua Program Studi Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya.
4. Bapak Waru Djuriatno ST., MT. selaku Ketua Kelompok Dosen Keahlian Rekayasa Komputer Jurusan Teknik Elektro Universitas Brawijaya.
5. Bapak Adharul Muttaqin ST., MT. dan Bapak Ir. Muhammad Aswin.,MT. selaku dosen pembimbing I dan dosen pembimbing II yang telah banyak memberikan bimbingan, masukan dan arahan dalam penyusunan skripsi ini.
6. Ibu Nurussaadah Ir., MT. selaku dosen penasehat akademik yang telah memberikan nasehat dan arahan dalam proses akademik penulis.
7. Ibunda Kasriatin, Ayahanda Alm. Supriyono, Kakak Anna Febriyanti Kakak Dewi Wijayanti, Kakak Muawiyah, Kakak Arie Setiadi dan seluruh keluarga besar, terimakasih atas segala dukungan dan do'a serta dorongan semangat hingga terselesaikannya skripsi ini.
8. Risa Pradita Camelia beserta keluarga yang selalu memberi dukungan dan semangat kepada penulis selama proses pengerjaan skripsi ini,

9. Rekan dalam pengerjaan skripsi Fanni Kharisma, Adam Hendra Brata, Mas Luky Mardhana, Galih Chandra M, Ferdiansyah, Kharisma Darmawan yang telah membantu memberi solusi pada permasalahan yang muncul dan memberi masukan dalam proses pengerjaan skripsi ini.
10. Sahabat Nine Brother yang selalu memberikan dukungan dan motivasi kepada penulis.
11. Seluruh teman-teman angkatan 2007 (CORE '07) yang selalu bersama dalam keceriaan.
12. Mas Galih Chandra , Mas Diaz, Agung Pramudita, dan teman-teman KIASS.
13. Semua pihak yang telah membantu penyelesaian skripsi ini baik secara langsung dan tidak langsung yang tidak dapat penulis sebutkan satu per satu.

Penulis berharap semoga skripsi ini bermanfaat bagi pembaca dan dapat memberikan sumbangan pikiran bagi pihak-pihak lain khususnya mahasiswa jurusan teknik elektro. Segala kritik dan saran yang penulis terima demi kesempurnaan skripsi ini karena penulis menyadari bahwa penulisan skripsi ini masih jauh dari sempurna dan masih banyak memiliki kekurangan.

Malang, Juli 2012

Penyusun

DAFTAR ISI

DAFTAR ISI ..... **iii**
DAFTAR GAMBAR ..... **vii**
DAFTAR TABEL ..... **ix**
ABSTRAK ..... **x**
BAB I
PENDAHULUAN **1**
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah 2
1.4 Tujuan 2
1.5 Sistematika Penulisan 2
BAB II
TINJAUAN PUSTAKA **4**
2.1 Citra Digital 4
2.1.1 Ukuran Piksel dan Resolusi Citra 5
2.1.2 Level Kuantisasi 6
2.2 Jenis Jenis Citra 6
2.2.1 Citra Biner 6
2.2.2 Citra Skala Keabuan (*grayscale*) 7
2.2.3 Citra Berwarna 7
2.2.4 Konversi Citra Berwarna (RGB) ke Citra *Grayscale* 8
2.2.5 Konversi Citra *Grayscale* menjadi Citra Biner 9
2.3 Model Warna RGB 10
2.4 *Bitmap* 11
2.5 Pengolahan Tranformasi Geometri 13
2.5.2 *Resize* 14
2.5.3 Interpolasi 14

2.5.3.1 Interpolasi Nearest Neigbour 14
2.5.3.2 Interpolasi Bilinier 15
2.5.3.3 Interpolasi Bicubik 15
2.6 *Direct Show* 15
2.7 Blooming 16
2.8 *Laser* 18
2.8.1 Range Warna Merah *Laser* 19
2.8.2 Range Warna Hijau *Laser* 20
2.9 *Mouse* 20

BAB III

METODOLOGI 22
3.1 Studi Literatur 22
3.2 Perangkat Yang Digunakan 22
3.3 Perancangan dan Implementasi Sistem 23
3.4 Pengujian dan Analisis 24
3.5 Pengambilan Kesimpulan dan Saran 24

BAB IV

PERANCANGAN DAN IMPLEMENTASI 25
4.1 Perancangan Secara Umum 25
4.1.1 Blok Sistem Diagram 25
4.1.2 Cara Kerja Aplikasi 26
4.2 Perancangan Aplikasi 27
4.2.1 Inisialisasi *Webcam* 28
4.2.2 Pengolahan Citra Dan Mouse Event 28
4.2.2.1 Operasi *Bitmap* 30
4.2.2.3 Cropping dan *Resize* 31
4.2.2.4 Perancangan Area Pembangkit *Mouse Event* 31
4.2.2.5 Pengolahan Citra Bagian *Color Filter* 33
4.2.2.6 Pembangkit *Event* klik dan *scroll* 35
4.2.2.7 Pembangkit *Event* Kontrol Kursor 39
4.3 Perancangan Program Keseluruhan 43
4.4 Implementasi Sistem 44

4.5 Lingkungan Implementasi 44
4.6 Implementasi Interface (Antarmuka) 45
4.6.1 Implementasi Antarmuka *Open Local Capture Device* 45
4.6.2 Implementasi Antarmuka untuk Melakukan *Tune Color* 46
4.6.3 Implementasi Tampilan Saat Program Berjalan 46
BAB V
PENGUJIAN 48
5.1 Pengujian Kamera 48
5.2 Pengujian FPS *Video* 49
5.3 Pengujian Pendeteksian Warna *Laser* Terhadap Kondisi Penerangan 50
5.4 Pengujian Pengaruh Skala *Resize* Terhadap Kecepatan Proses Citra 51
5.5 Pengujian Pengaruh Skala Resize Dan Kamera terhadap Waktu Yang Dibutuhkan Untuk Mengaktifkan Event Klik Dan Hasil Tracking *Laser* 52
5.6 Analisis Faktor Kegagalan 53
5.7 Faktor kegagalan pendeteksian sinar *laser* 53
5.8 Kesimpulan Hasil Pengujian 54
BAB VI
PENUTUP 55
6.1 Kesimpulan 55
6.2 Saran 55
DAFTAR PUSTAKA 56

DAFTAR GAMBAR

Gambar 2.1 Representasi citra digital 4
Gambar 2.2 *Image size and resolution* 5
Gambar 2.3 Citra Biner 6
Gambar 2.4 Skala keabuan kedalaman 8 bit 7
Gambar 2.5 Citra berwarna dengan kanal setiap warna dasar penyusunnya 8
Gambar 2.6 Pemisahan Citra berwarna pada tiap kanal RGB dan luminance. 9
Gambar 2.7 Konversi citra berwarna menjadi citra *grayscale.* 9
Gambar 2.8 Konversi citra *grayscale* menjadi citra biner 10
Gambar 2.9 Ruang warna RGB 11
Gambar 2.10 Format file *bitmap* umum 12
Gambar 2.11 Sebuah Citra Di-*croping* Sebesar W x H 13
Gambar 2.12 Ilustrasi Interpolasi 14
Gambar 2.13 Komponen Di Dalam *Direct Show* 16
Gambar 2.14 *Dynamic Range of Sensor.* 17
Gambar 2.15 *Pixel Size and Dynamic Range* 17
Gambar 2.16 Efek *Blooming* Pada Citra 18
Gambar 2.17 Spektrum Elektromagnetik 19
Gambar 2.18 Range Warna Untuk Mendeteksi *Laser* Merah 19
Gambar 2.19 Range Warna Untuk Mendeteksi *Laser* Hijau 20
Gambar 3.1 Rancangan sistem pendeteksi *laser pointer* secara keseluruhan 23
Gambar 4.1 Diagram Blok Sistem Secara Umum 26
Gambar 4.2 Detail Desain Aplikasi 28
Gambar 4.3 Flowchart Bagian Pengolahan Citra dan Mouse Event 30
Gambar 4.4 Area Pembangkit Mouse Event 31
Gambar 4.5 Flowchart Pembagian Area Pembangkit Event 32
Gambar 4.6 Flowchart Pembangkit *Mouse Event* Klik dan *Scroll* 35
Gambar 4.7 Flowchart Pembangkit *Event* Kontrol Kursor 39
Gambar 4.8 Flowchart Program Keseluruhan 43
Gambar 4.9 Tampilan Program Utama Pendeteksi *Laser Pointer* 45
Gambar 4.10 Tampilan Form *Open Local Capture Device* 46
Gambar 4.11 Tampilan *Tune Color* Pada Program Utama 46

Gambar 4.12 Tampilan Program Utama Saat Aplikasi Berjalan 47
Gambar 5.1 Pengujian Terhadap Kondisi Penerangan Ruangan. 50
Gambar 5.2 o*verexposed* pada *webcam* 54

DAFTAR TABEL

Tabel 5.1 Hasil pengujian koneksi *webcam* 48

Tabel 5.2 Pengujian Pengaruh Skala *Resize* Terhadap FPS 49

Tabel 5.3 Pengujian Respon Program terhadap Intesitas Penerangan Ruangan 50

Tabel 5.4 Pengujian Pengaruh Skala *Resize* Terhadap Kecepatan Proses 1 *Frame*. 51

Tabel 5.5 Pengujian Pengaruh Skala *resize* terhadap Waktu Yang Dibutuhkan Untuk Mengaktifkan *Event* Klik dan Hasil *Tracking Laser* 52

ABSTRAK

GALLANT HENDRAYANA PUTRA, Jurusan Teknik Elektro, Fakultas Teknik Universitas Brawijaya, Juni, 2012. *Aplikasi Akses Mouse Menggunakan Laser pointer*. Dosen Pembimbing: Adharul Muttaqin,ST.,MT. dan Ir.Muhammad Aswin,MT.

Pengunaan *laser pointer* pada saat presentasi umumnya digunakan sebagai *pointing* saja. Dengan adanya teknik pengolahan citra dengan menggunakan komputer, *laser pointer* dapat difungsikan sebagai *mouse* yang mempunyai *mouse event* seperti *click dan Scroll*.

Skripsi ini membahas bagaimana cara mendeteksi sinar *laser pointer* sehingga dapat mengakses *mouse event*. Dimulai dari inisialisasi *device* kamera menggunakan *Direct Show*, yaitu komponen yang berperan untuk mengakses inti driver *Windows Device Modul* (WDM) kamera *webcam*. Selanjutnya, kamera *webcam* ini akan digunakan sebagai mata atau sensor untuk mendeteksi obyek berupa berkas sinar *laser*. Warna *laser* dijadikan sebagai acuan objek untuk dapat dikenali aplikasi komputer dengan menggunakan teknik pengolahan citra. Fungsi-fungsi pengolahan citra yang diperlukan diambil dari *open source computer vision library* seperti *AForge*. *Library* ini berfungsi sebagai pengolah citra untuk mendeteksi berkas *laser* dan mendapatkan kordinat (x.y) pada citra, kemudian kordinat tersebut digunakan untuk mengaktifkan *mouse event*. Pengaktifan *mouse event* memanfaatkan *library windows API* dengan cara menginisialisasi *user32.dll* kedalam program aplikasi.

Aplikasi ini mampu bekerja dengan menggunakan dua jenis *laser* yaitu *red laser pointer* dan *green laser pointer*. Parameter intensitas cahaya, kualitas kamera dan brightness pada LCD proyektor sangat menentukan keberhasilan aplikasi agar tidak terjadi *overexposed* pada citra yang diamati.

BAB I
PENDAHULUAN

1.1 Latar Belakang

Sejak munculnya komputer personal atau *desktop* beberapa dekade lalu, *mouse* dan *keyboard* merupakan perlengkapan standar operasional input yang wajib ada. Keyboard merupakan *device* berupa papan yang terdiri atas tombol-tombol untuk mengetik kalimat dan simbol-simbol khusus pada komputer, sedangkan *mouse* digunakan sebagai *device pointing* untuk menggerakkan pointer. Disamping itu banyak perangkat pointing bermunculan seperti *touchpad, trackball, IR wii remote, eye mouse (gaze tracker)* dan lain lain.

Pengolahan citra merupakan salah satu jenis teknologi untuk menyelesaikan masalah pemrosesan citra. Dalam perkembangannya, teknologi ini mendukung komputer untuk dapat melihat selayaknya manusia dengan menggunakan kamera. Tentu saja hal itu membawa perkembangan besar terhadap interaksi aktif manusia dengan komputer. Dalam kasus penggunaan *laser pointer,* berkas sinar *laser pointer* hanya digunakan sebagai penanda saja saat melakukan presentasi. Di pasaran banyak beredar *laser mouse* yang dilengkapi *wireless* beserta tombol pengatur kursor *key up, down, left* dan *right.* Namun dalam hal kecepatan perpindahan kursor dirasa kurang leluasa seperti menggunakan perangkat *mouse.*

Dalam skripsi ini penulis mengadopsi dari skripsi yang berjudul "*Aplikasi Pendeteksi Laser pointer Untuk Menggerakkan Kursor Mouse pada Komputer*"(Luky Mardhana, ST. 2011). Pada skripsi tersebut aplikasi dapat menggerakan kursor *mouse* dengan menggunakan *laser pointer* tetapi belum mampu mengaktifkan *mouse event* seperti klik dan *scroll*. Agar aplikasi dapat digunakan layaknya mouse yang dapat mengaktifkan *event* klik dan *scroll,* penulis memanfaatkan *library windows API* pada program aplikasi, sehingga diharapkan mempermudah seseorang dalam melakukan presentasi tanpa harus repot mengaktifkan klik dan *scroll* dengan menggunakan *mouse* komputer. Hal ini tentunya berguna bagi seseorang yang melakukan suatu presentasi tanpa menggunakan operator komputer.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan masalah ditekankan pada:

1. Bagaimana cara membangkitkan *event* klik dan *scroll.*
2. Bagaimana cara memodifikasi algoritma dan program pada aplikasi sebelumnya, sehingga dapat mengintegrasikan fungsi-fungsi pembangkit *event* untuk membangkitkan *event* klik dan *scroll*.

1.3 Batasan Masalah

Beberapa hal yang menjadi batasan masalah dalam pembuatan program ini antara lain:

1. Data yang diambil dari objek yang ditangkap adalah berupa citra digital yang di-*capture* dengan alat bantu *webcam*.
2. Sistem bekerja mendeteksi berkas sinar *laser* dan mengendalikan *kursor mouse*.
3. *Laser pointer* yang digunakan adalah *laser diode*.
4. Bahasa pemrograman yang dipakai adalah C# Microsoft Visual Studio 2010.NET

1.4 Tujuan

Tujuan penyusunan skripsi ini adalah untuk mengembangkan skripsi "*Aplikasi Pendeteksi Laser pointer untuk Menggerakkan Kursor pada Komputer*"(Luky Mardhana, ST. 2011) dengan menambahkan *event* klik dan *scroll*, sehingga *laser pointer* dapat berfungsi layaknya mouse komputer.

1.5 Sistematika Penulisan

Sistematika penulisan laporan skripsi ini adalah sebagai berikut :

BAB I Pendahuluan

Menjelaskan latar belakang, rumusan masalah, ruang lingkup, tujuan dan sistematika penulisan skripsi.

BAB II Dasar Teori

Menjelaskan kajian pustaka dan dasar teori yang digunakan.

BAB III Metodologi

Menjelaskan metode yang digunakan dalam pengerjaan skripsi.

BAB IV Perancangan dan Implementasi

Menjelaskan langkah langkah Perancangan Aplikasi Pendeteksi *Laser pointer* untuk mengakses *Mouse event* beserta hasil implementasinya.

BAB V Pengujian

Menjelaskan langkah-langkah pengujian dari sistem yang telah dibuat dan membahas hasil pengujiannya.

BAB VI Kesimpulan dan Saran

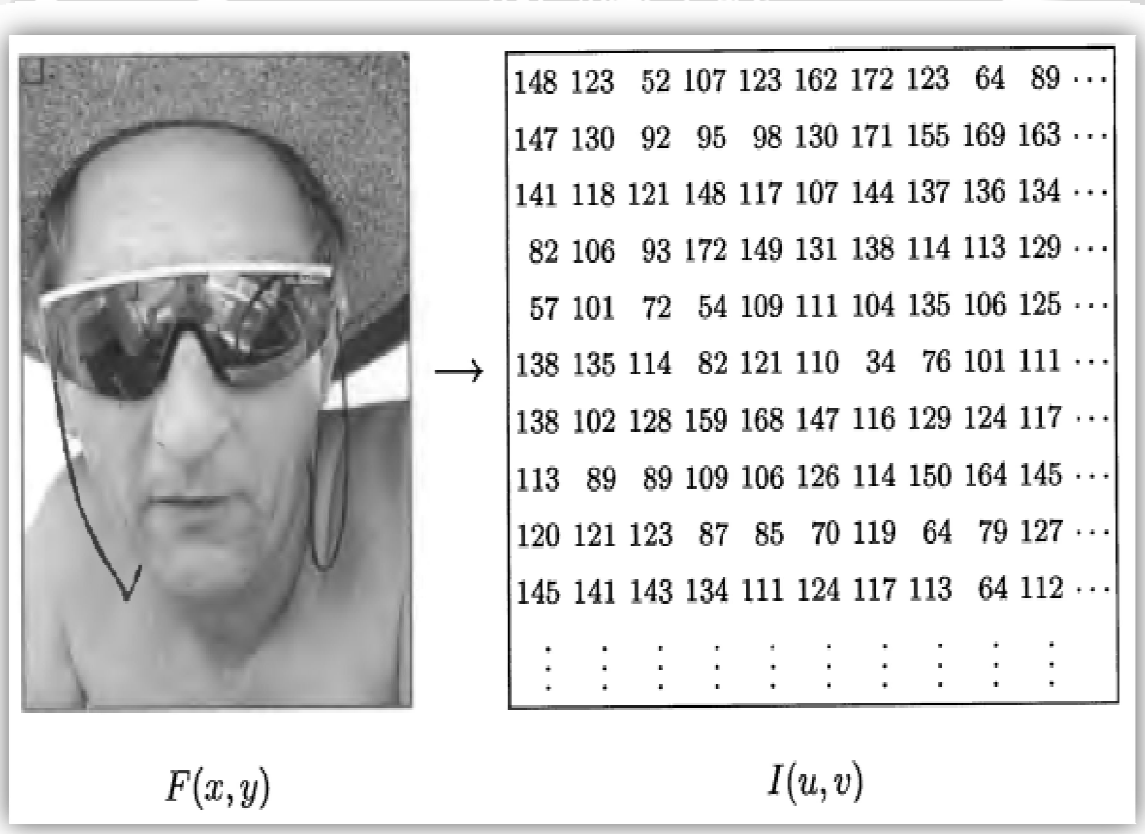
Berisi Kesimpulan dan Saran

BAB II

TINJAUAN PUSTAKA

2.1 Citra Digital

Citra adalah suatu representasi (gambaran), kemiripan, atau imitasi dari suatu objek.Citra digital merupakan diskritisasi dari nilai suatu citra, baik dalam hal koordinat spasial maupun intensitas cahaya (T Suyono dkk, 2009). Sebuah citra diubah ke dalam bentuk digital agar dapat disimpan dalam memori komputer atau media lain. Proses mengubah citra menjadi digital dapat dilakukan dengan perangkat elektronik misalnya *scanner*, kamera digital, *handycam*, *webcam*, dan lain lain. Seperti pada Gambar 2.1 citra digital didefinisikan sebagai fungsi dua variable F(x,y), dimana *x* dan *y* adalah koordinat spasial dan nilai I(u,v)adalah intensitas citra pada koordinat tersebut.



Gambar 2.1 Representasi citra digital

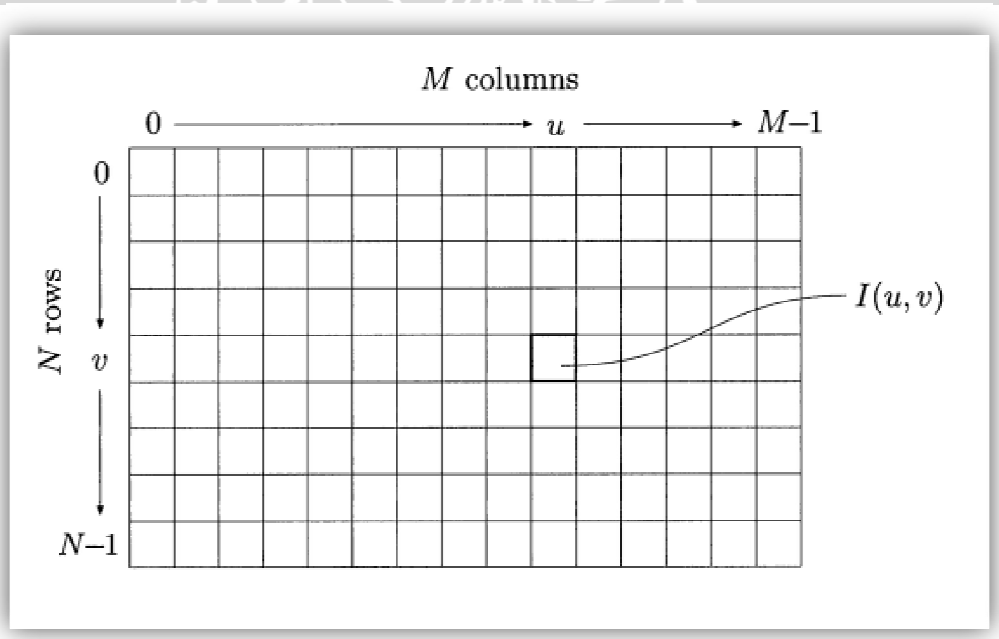
Sumber: Wilhem Burger., Mark James Burger. Digital Images Processing. 2008

Format citra digital didefinisikan sebagai array 2 dimensi dari intensitas cahaya yang diwakili oleh piksel. Setiap piksel mempunyai elemen lokasi dan nilai intensitas sehingga mengandung informasi spasial dan temporal yang bisa mengambarkan fenomena fisis yang direkam. Ada tiga atribut penting dari citra digital, yaitu

1. Ukuran Citra
2. Resolusi citra
3. Level Kuantisasi

2.1.1 Ukuran Piksel dan Resolusi Citra

Ukuran citra yaitu dimensi panjang dan lebar dari matriks citra sekaligus menunjukkan banyaknya piksel. Piksel merupakan satuan komponen terkecil yang menentukan ukuran dari suatu gambar (Gonzales & Woods, 2002). Resolusi citra menyatakan ukuran bit dari suatu citra dalam satuan piksel *image*. Untuk memudahkan, dapat di asumsikan dengan gambar segiempat dengan panjang M (dalam kolom) dan tinggi N (dalam baris) dari matrix I, seperti pada Gambar 2.2.



Gambar 2.2 *Image size and resolution*

Sumber: Wilhem Burger, Mark James Burger. Digital Images Processing. 2008

Banyaknya piksel persatuan panjang dikenal dengan satuan *dot per inch* (dpi). Semakin banyak titik per incinya, kualitas citra semakin baik dan detail, namun semakin banyak pula memori (bit) yang diperlukan untuk menyimpan citra tersebut.

2.1.2 Level Kuantisasi

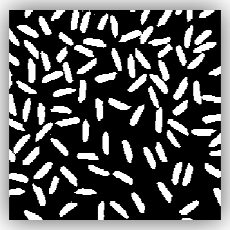
Level kuantisasi adalah untuk merepresentasikan nilai intensitas cahaya yang sudah terkuantisasi. Semakin besar skala nilai kuantisasi, semakin baik kualitas citra karena semakin banyak *luminance* yang dapat diwakili menggambarkan keadaan intensitas yang sebenarnya. *Luminance* yaitu tingkat intensitas cahaya yang dipancarkan oleh warna tertentu.

2.2 Jenis Jenis Citra

Informasi penting dari suatu citra adalah informasi mengenai intensitas cahaya yang tertangkap, ini erat hubungannya dengan warna. Suatu citra digital dapat dikelompokkan berdasarkan format intensitas penyusunnya yaitu citra biner (*monokrom*), citra skala keabuan (*grayscale*), dan citra berwarna.

2.2.1 Citra Biner

Citra biner atau *monokrom* adalah tipe gambar yang setiap pikselnya hanya berisi oleh dua kondisi nilai, yaitu hitam atau putih seperti yang ditunjukkan pada Gambar 2.3. Nilai ini biasanya di-*encode* dengan 1 bit (0/1) per pikselnya. Citra biner biasanya digunakan untuk merepresentasikan garis gambar, pengkodean untuk transmisi fax serta printer *black* and *white.*

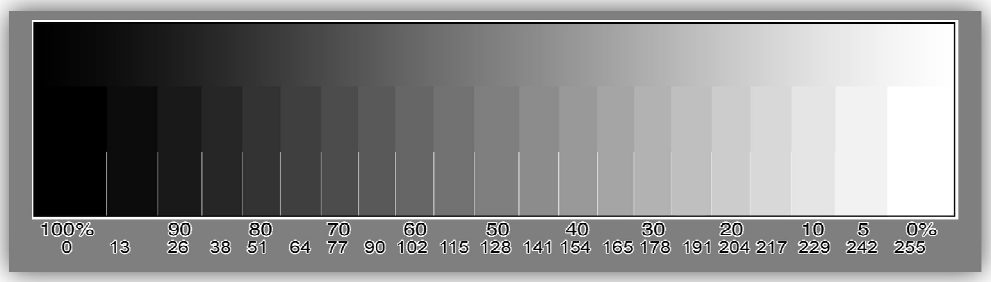


Gambar 2.3 Citra Biner

Sumber : http://blogs.mathworks.com

2.2.2 Citra Skala Keabuan (*grayscale*)

Citra skala keabuan adalah tipe gambar yang setiap pikselnya memiliki warna diantara hitam dan putih. Nilai antara ini adalah nilai abu abu, banyaknya kemungkinan warna abu-abu tergantung dari besarnya kedalaman bit dari citra tersebut. Misalnya suatu citra memiliki kedalaman sebesar 8 bit, maka citra tersebut mempunyai skala warna sebanyak 2^8 atau 256 skala mulai dari 0 sampai 255. Angka nol merepresentasikan warna hitam dan angka 255 merepresentasikan warna putih, sedangkan nilai diantara 0 sampai 255 merepresentasikan warna abu abu (Gonzales & Woods, 2002), ditunjukkan pada Gambar 2.4.



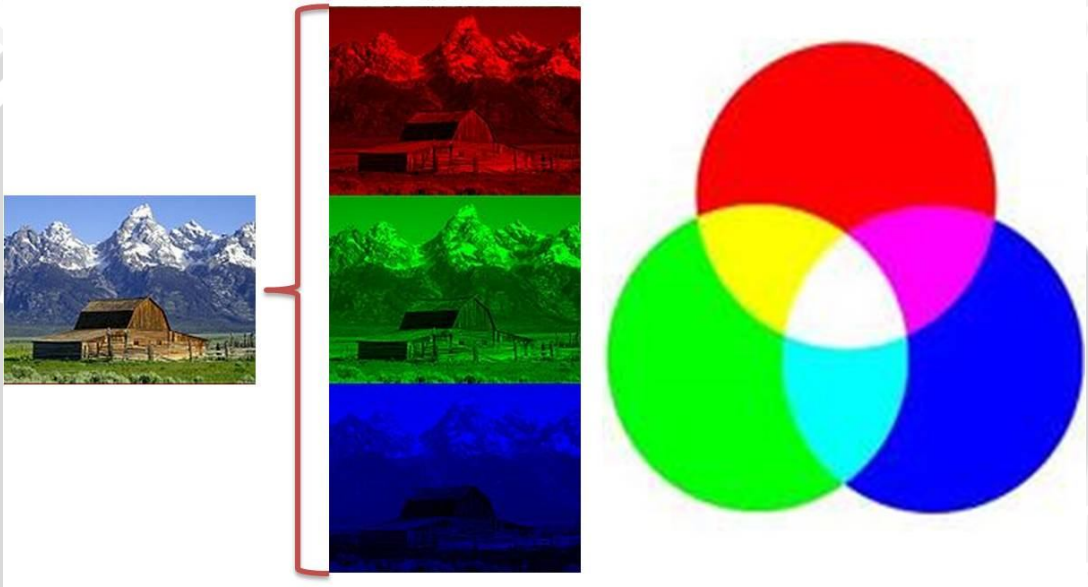
Gambar 2.4 Skala Keabuan Kedalaman 8 Bit

Sumber : http://www.inksupply.com/html/images/21stepwide8bit.jpg

2.2.3 Citra Berwarna

Citra berwarna merupakan gabungan dari beberapa lapis citra kanal warna. Teknologi dasar untuk menciptakan warna pada citra digital merupakan kombinasi dari kanal tiga warna dasar yaitu merah, hijau, dan biru (*RGB- Red, Green, Blue*) ditunjukkan pada Gambar 2.5. Kombinasi kanal warna ini didapat dari penangkapan intensitas cahaya pada frekuensi yang berbeda, yaitu frekuensi 560 nm (merah), 530 nm (hijau) dan 430 nm (biru). Masing masing kanal mempunyai nilai yang mengambarkan intensitas setiap warna dasarnya.Pringsip dasar pengolahan citra adalah pengolahan warna RGB pada posisi tertentu. Dalam pengolahan citra warna direpresentasikan dengan nilai hexadecimal dari 0x00000000 sampai 0x00ffffff. Definisi nilai warna tersebut berisikan variable 0x00 yang menyatakan angka dibelakangnya adalah hexadecimal. Terlihat bahwa setiap warna mempunyai range nilai 00 (angka decimal bernilai 0) sampai ff (angka desinal bernilai 255) atau mempunyai derajat keabu-abuan 256 = 2^8,

dengan demikian range warna RGB adalah (2^8) $(2^8)$$(2^8)$ = 2^{24} atau dikenal dalam *windows* dengan istilah *true colour*. Nilai warna yang digunakan merupakan gabungan warna cahaya merah, hijau, dan biru. Jadi, untuk menyajikan warna warna tertentu dapat dengan mudah dilakukan dengan cara mencampur ketiga warna dasar RGB. Contoh format lain adalah CMY (*cian*, *magenta*,dan *yellow*), format CMY biasa digunakan pada industri percetakan sedangkan RGB format umum yang digunakan kamera, monitor.



Gambar 2.5 Citra Berwarna Dengan Kanal Setiap Warna Dasar Penyusunnya

2.2.4 Konversi Citra Berwarna (RGB) ke Citra *Grayscale*

Citra warna terdiri dari beberapa lapisan kanal warna RGB dan setiap kanal warna mewakili suatu intensitas suatu warna cahaya. Pada setiap kanal RGB dapat diubah formatnya secara ekuivalen kedalam citra skala keabuan, ditunjukkan pada Gambar 2.6. Konversi citra kanal warna dengan skala keabuan dilakukan dengan mencocokkan *luminance* (pancaran) dari citra berwarna dengan warna abu-abu. Pada citra RGB 8 bit, masing masing kanal RGB mempunyai nilai intensitas 0 sapai 255. Setiap intensitas mempunyai ekuivalen dengan warna abu-abu pada citra skala keabuan, seperti pada Persamaan 2-1.

$$Y = luminance\ (RGB) = grayscale = (wr \times red) + (wg \times green) + (wb \times blue) \quad (2\text{-}1)$$

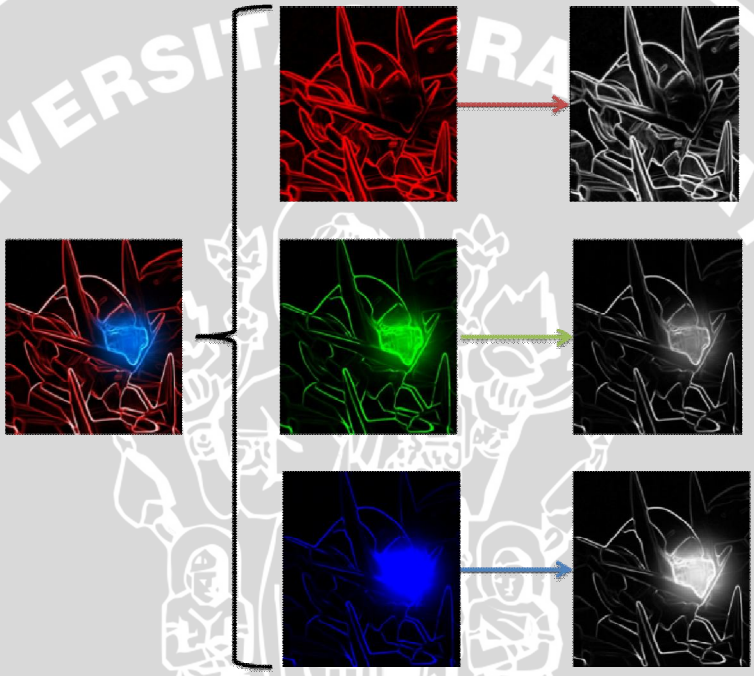
pUntuk mendapatkan citra *grayscale* dari citra warna dengan bobot

wr = 0,299

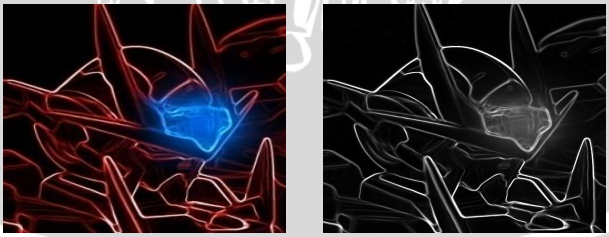
wb= 0,587

wg= 0,114

Setelah ketiga kanal warna mempunyai nilai skala keabuan dari hitungan matematis Persamaan 2-1, hasil konversi citra berwarna ke citra *grayscale* seperti yang ditunjukkan pada gambar 2.7.



Gambar 2.6 Pemisahan Citra Berwarna Pada Tiap Kanal RGB dan Luminance.



Gambar 2.7 Konversi Citra Berwarna Menjadi Citra *Grayscale*.

2.2.5 Konversi Citra *Grayscale* menjadi Citra Biner

Konversi citra *grayscale* menjadi citra biner dikenal dengan istilah *thresholding*, yaitu mengubah suatu citra menjadi citra hitam-putih atau biner. Proses pengubahannya dengan cara operasi nilai piksel yang memenuhi sarat

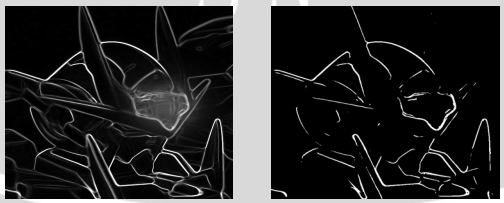
ambang batas dan nilai yang dikehendaki (*threshold*). *Thresholding* merupakan proses pengelompokan warna dengan menentukan nilai ambang atau nilai batas (a_{th}) yang diterapkan untuk seluruh warna pada suatu daerah citra. Persamaan fungsi *threshold* merubah semua piksel menjadi dua keadaan dengan nilai a_0 atau a_1 seperti persamaan dibawah ini:

dengan $0 < a_{th} \leq a_{max}$, citra biner yang hanya memiliki dua nilai yaitu 1 atau 0 yang jika di implementasikan kedalam 8-bit nilai yang digunakan adalah 0 dan 225.

Algoritma ini umum digunakan untuk mempertegas objek citra maupun membuang noise dan gangguan yang tidak perlu. Secara umum algoritma *threshold* sederhana dengan membandingkan nilai setiap piksel pada citra dengan nilai *threshold* yang telah ditentukan. Setelah dibandingkan ditetapkan syarat berikut:

1. Jika nilai piksel yang dibandingkan dengan nilai *threshold* lebih besar, maka ganti nilai piksel dengan nilai 255 (putih).
2. Jika nilai piksel yang dibandingkan dengan nilai *threshold* lebih kecil, maka ganti nilai piksel dengan nilai 0 (hitam).

Sehingga didapat suatu citra baru yang hanya berwarna hitam atau putih saja, seperti yang diilustrasikan pada Gambar 2.8. Pemilihan nilai *threshold* juga disesuaikan dengan kebutuhan, pemilihan tersebut didasarkan pada sebaran terang-gelap pada citra originalnya.

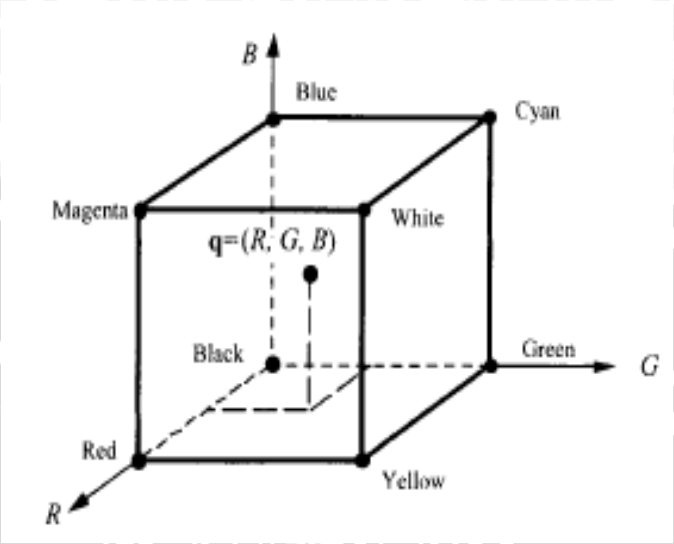


Gambar 2.8 Konversi citra *grayscale* menjadi citra biner

2.3 Model Warna RGB

Model warna adalah sebuah sistem koordinat yang bisa memetakan semua warna dalam sistem tersebut dengan sebuah titik. Model warna RGB berorientasi *hardware*, RGB sering digunakan didalam sebagian besar aplikasi komputer,

karena ruang warna ini tidak diperlukan transformasi terutama untuk warna monitor dan warna pada kamera digital. Dalam model ini tiap warna ditunjukan dengan kombinasi ketiga warna primer (*red, green, blue*) ketiga warna tersebut membentuk sistem koordinat 3 dimensi.



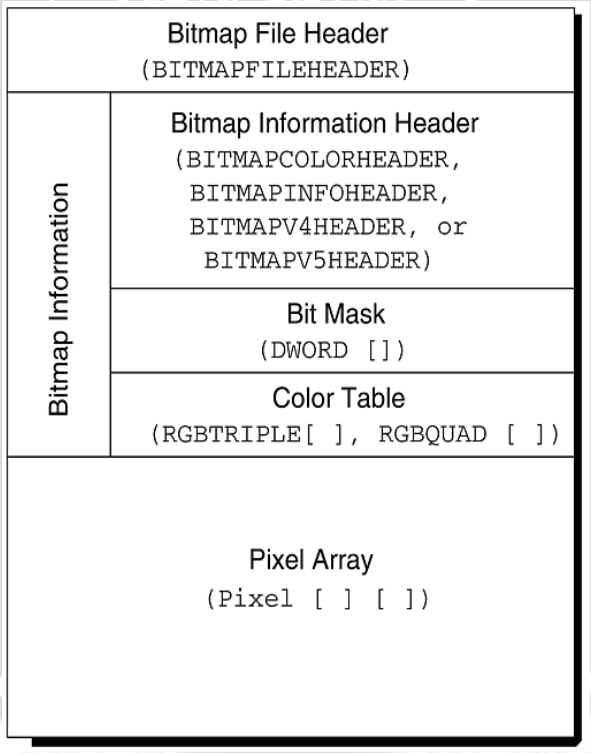
Gambar 2.9 Ruang warna RGB

Sumber: Andreas Koschan, Digital Color images Processing. 2007

2.4 *Bitmap*

Bitmap atau raster merupakan gambar digital yang direpresentasikan dalam bentuk matriks atau dipetakan dengan menggunakan bilangan biner ataupun bilangan lain. Citra ini memiliki kelebihan untuk memanipulasi warna dari sebuah gambar serta menyimpan data kode citra secara digital dan lengkap (cara penyimpanan per piksel). Masing masing piksel memiliki informasi warna, jumlah kemungkinan warna pada suatu piksel tergantung pada satuan bit yang dimiliki gambar *bitmap* tersebut. Pada citra format *bitmap* yang tidak terkompresi, piksel citra disimpan dengan kedalaman warna 1, 4, 8, 24, atau 32 bit per piksel. Sebagai contoh gambar dengan format 8 bit, maka piksel-piksel yang menyusunnya dapat menampilkan kemungkinan warna sebanyak 2 pangkat 8 atau 256 warna. Gambar *bitmap* dengan resolusi (jumlah piksel setiap satuan ukur) besar, akan terlihat lebih halus dibandingkan dengan yang memiliki resolusi rendah. Resolusi gambar *bitmap* dinyatakan dalam satuan *dot per inch* (dpi) atau *piksel per inch* (ppi).

Struktur penyimpanan file *bitmap* pada umumnya terdiri dari 4 blok data yaitu: *BMP header*, *Bit Information* (*DIB header*), *Color Palette*, dan *Bitmap Data*. *BMP header* berisi informasi umum dari citra *bitmap*. Blok ini berada pada bagian awal file citra dan digunakan untuk mengidentifikasi citra. Beberapa aplikasi pengolah citra akan membaca blok ini untuk memastikan bahwa citra tersebut berformat *bitmap* dan tidak dalam kondisi rusak. Bit information berisi informasi detail dari citra *bitmap*, yang akan digunakan untuk menampilkan citra pada layar. *Color palette* berisi informasi warna yang digunakan untuk indeks warna *bitmap*, dan *bitmap* data berisi data citra yang sebenarnya, piksel per piksel. Model ruang warna yang digunakan pada citra *bitmap* adalah RGB (red, green, dan blue). Sebuah ruang warna RGB dapat diartikan sebagai semua kemungkinan warna yang dapat dibuat dari tiga warna dasar red, green, dan blue. RGB sering digunakan di dalam sebagian besar aplikasi komputer karena dengan ruang warna ini tidak diperlukan transformasi untuk menampilkan informasi di layar monitor.



Gambar 2.10 Format File *Bitmap* Secara Umum

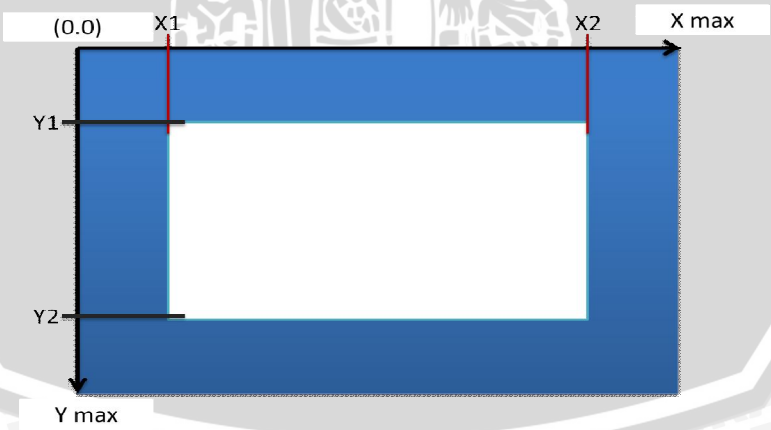
Sumber : Feng Yuan. Windows Graphics Programming Win 32. 2000

2.5 Pengolahan Tranformasi Geometri

Citra terdiri atas piksel yang didalamnya terdapat dua komponen yaitu koordinat piksel (x,y) dan intensitas warna, pada pembahasan di atas hanya melakukan modifikasi warna piksel saja, sedangkan koordinat piksel tetap tidak mengalami perubahan. Untuk memenuhi kebutuhan pengolahan citra seringkali diperlukan operasi geometri baik ukuran maupun orientasinya seperti pencerminan (*flipping*), rotasi, pemotongan (cropping) penskalaan dengan interpolasi, transpose citra, efek gelombang, efek warp, efek swirl, dan efek glass. Dalam pembuatan aplikasi ini digunakan teknik tranformasi geometri *cropping*, *resize* dan interpolasi.

2.5.1 Pemotongan citra (*cropping*)

Dalam dunia fotografi dan percetakan cropping adalah teknik pengolahan citra untuk memotong area pada gambar gambar atau citra. Cropping merupakan salah satu dasar teknik manipulasi gambar yang digunakan untuk menghilangkan bagian yang tidak di inginkan dari gambar, teknik ini digunakan untuk memotong suatu bagian dari citra sehingga diperoleh citra yang berukuran lebih kecil. Ilustrasi cropping ditunjukkan pada gambar 2.16.



Gambar 2.11 Sebuah Citra Di-cropping Sebesar W x H

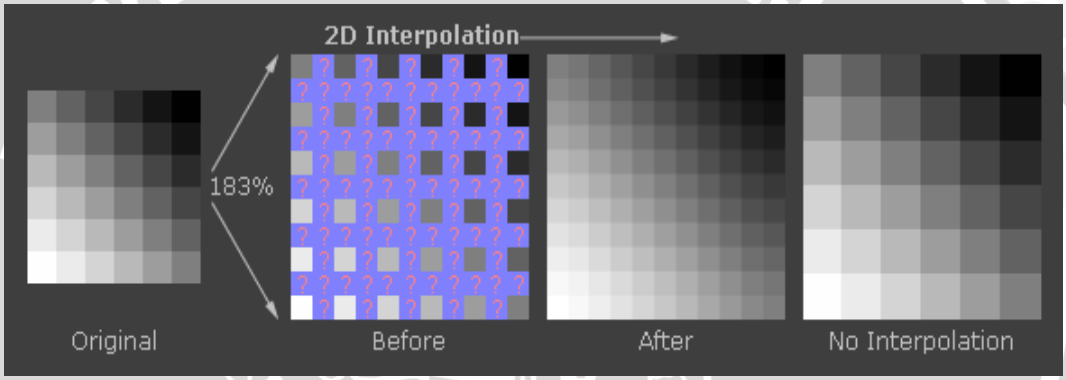
Sumber : Sutoyo, T., Mulyanto, Edy., Suhartono, Vincent., Nurhayati, Oky Dwi.Wijanarto . Teori Pengolahan Citra Digital. 2009.

2.5.2 *Resize*

Operasi ini dimaksudkan untuk memperbesar atau memperkecil citra. Hal ini dapat dilakukan dengan mengitroduksikan parameter skala horisontal dan vertikal.

2.5.3 Interpolasi

Operasi geometri mengakibatkan perubahan letak piksel dari suatu citra, bisa jadi letak piksel baru tersebut bukanlah bilangan bulat sehingga akibatnya citra mengalami aliasing. Untuk menghasilkan citra yang mendekati citra aslinya perlu dilakukan interpolasi. Interpolasi terdapat 3 macam yaitu interpolasi nearest neigbour, bilinear dan bicubik



Gambar 2.12 Ilustrasi Interpolasi

2.5.3.1 Interpolasi Nearest Neigbour

Algoritma interpolasi nearest neigbour merupakan metode yang sederhana untuk membuat piksel-piksel menjadi besar. Piksel baru dibuat dengan cara mengambil informasi dari 1 piksel tetangga terdekatnya. Apabila memperbesar gambar sebesar 200% maka satu piksel akan diperbesar menjadi 2 x 2, dimana informasi 4 piksel baru diambil dari informasi piksel lamanya dengan warna yang sama

2.5.3.2 Interpolasi Bilinier

Algoritma interpolasi bilinier membuat piksel baru dengan cara mengambil rata rata 4 piksel (2 x 2) tetangga yang terdekat dari piksel lamanya. Interpolasi bilinier mengambil nilai rata-rata dari empat titik yang mengelilingi titik yang dimaksudkan. Keempat titik ini memberikan sumbangan terhadap nilai

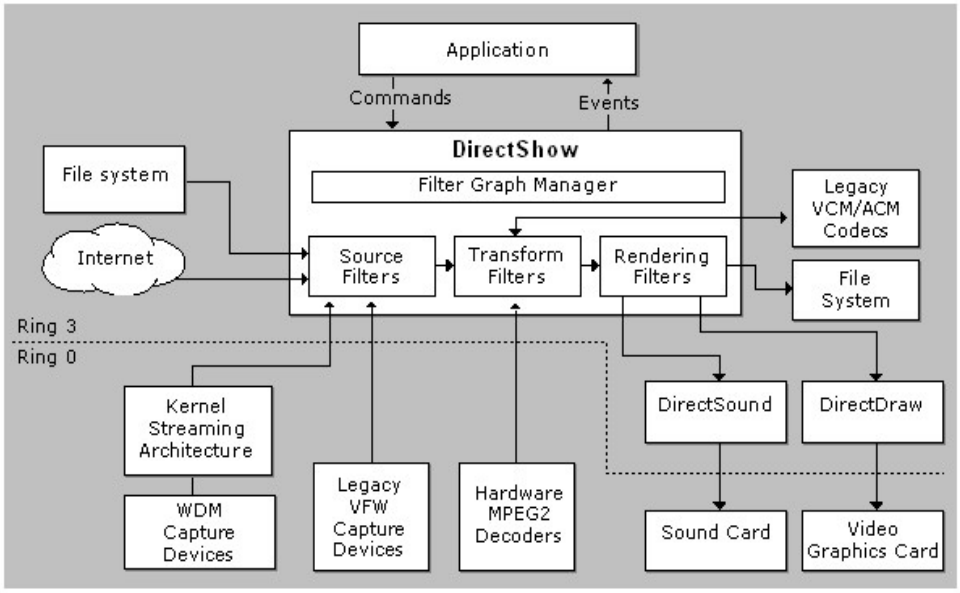
titik tersebut dengan bobot masing masing titik yang linier dengan jaraknya terhadap koordinat yang dimaksudkan. Makin dekat titik tetangga tersebut, makin besar bobotnya, dan sebaliknya makin jauh akan semakin kecil bobotnya.

2.5.3.3 Interpolasi Bicubik

Algoritma Interpolasi yang lebih rumit dan menghasilkan gambar yang lebih halus dibanding algoritma interpolasi bilinier. Piksel baru yang akan dibuat dengan cara mengambil informasi dari 16 piksel (4 x 4) tetangga yang terdekat dari citra aslinya.

2.6 *Direct Show*

Direct Show adalah framework multimedia dan API yang diciptakan oleh Microsoft untuk pengembang software berbasis COM (Component Object Model) seperti ditunjukkan pada Gambar 2.13, memiliki kemampuan untuk mengeksplorasi fitur-fitur untuk mengambil data dari berbagai devais, mengolah, dan memainkan media gambar dan suara. Komponen penyusun *Direct Show* adalah perangkat lunak berupa komponen pengaliran data yang disebut filter. Filter adalah berupa obyek COM yang dapat mewakili devais perangkat keras, perangkat lunak enkoder atau dekoder, sebuah pembuat audio atau video, ataupun yang memiliki kemampuan memproses audio video. *Direct Show* menyediakan filter yang mendukung pengaksesan dan setting perangkat keras yang berbasis w*indows driver model* (WDM), juga mendukung filter *legacy* seperti *Video for Windows* (VFW) dan codec untuk interface *audio compression manager* (ACM) dan *video compressinon manager* (VCM).



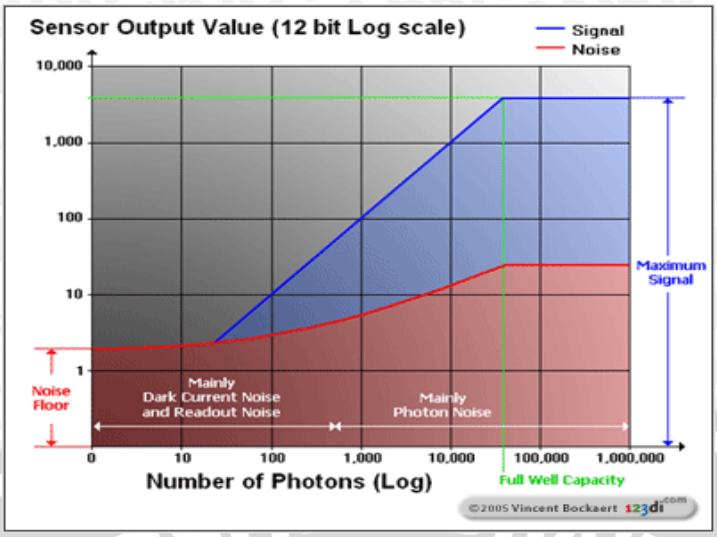
Gambar 2.13 Komponen Didalam *Direct Show*

Sumber: MSDN Microsoft *Direct Show*

2.7 *Blooming*

Apabila kita keluar dari ruangan gelap lalu menuju ruangan yang terang, maka akan terjadi kejutan pada mata kita sebelum bisa menyesuaikan diri. Sebaliknya ketika kita berada pada ruangan yang terang dan memasuki ruangan gelap, mata kita perlu menyesuaikan diri sebelum bisa melihat disekitarnya. Begitu juga sensor sebuah kamera digital bisa mengalami kesulitan menangkap perbedaan terang dan gelap yang terlalu besar pada suatu scene. Untuk bisa menangkap perbedaan itu, sensor kamera harus memiliki kemampuan dynamic range yang cukup besar.

Secara teori, dynamic range sebuah sensor didefinisikan sebagai proses 'pembacaan' subject dimana possible signal terbesar yang bisa ditangkap sebuah sensor dibagi dengan possible signal terkecil. Possible signal terbesar diartikan sebagai nilai kemungkinan terbesar dari kapasitas tertinggi yang mungkin ditangkap oleh pixel pada sensor kamera. Sedang signal terendah adalah ketika sensor sama sekali tidak terexpose oleh cahaya apapun, disebut juga dengan istilah *noise floor*.



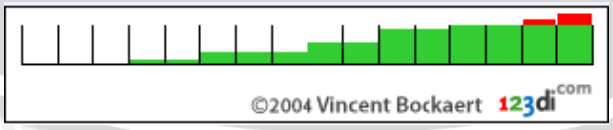
Gambar 2.14 *Dynamic Range of Sensor*

Sumber : http://cyberwayang.multiply.com

Bahasa mudahnya, sebuah kamera dengan kemampuan dynamic range yang tinggi mampu menangkap detail shadow dan detail highlight secara bersamaan.

Sensor kamera digital memiliki jutaan pixel yang menangkap photon saat sensor sedang mengexpose subject, dimana area yang lebih terang pada subject, artinya lebih banyak photon yang ditangkap. Setelah proses exposure selesai, nilai atau value tertentu akan diberikan kepada masing-masing pixel. Secara konsep, pixel yang kosong (tidak berisi photon) akan diberikan nilai 0 dan pixel yang penuh akan diberikan nilai 255, dengan kata lain nilai 0 mewakili warna hitam (pure black) dan nilai 255 mewakili nilai putih terang (brighter).

Gambar konsep dibawah memperlihatkan sebuah sensor yang terdiri dari 16 pixel (sebuah sensor pada kenyataannya terdiri dari jutaan pixel) yang menerangkan proses kumpulan pixel-pixel yang menangkap photon, dimana pixel yang menangkap subject terang akan dipenuhi photon dengan sangat cepat.



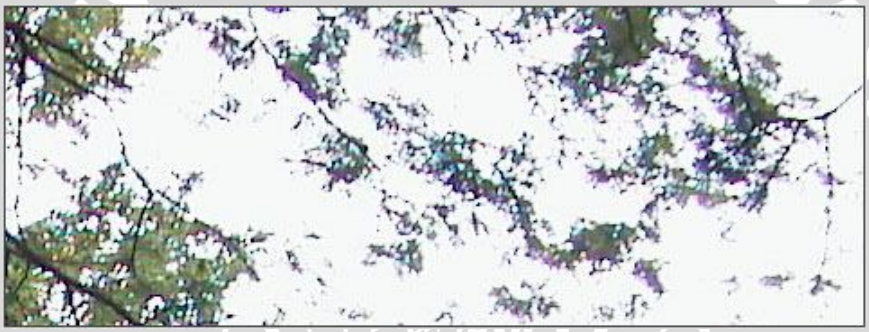
Gambar 2.15 ***Pixel Size and Dynamic Range***

Sumber : http://cyberwayang.multiply.com

Ketika pixel-pixel dipenuhi oleh photon, bisa saja terjadi overflow atau kelebihan beban yang mengakibatkan apa yang disebut dengan efek blooming. Ketika pixel-pixel itu kelebihan beban (yang diindikasikan dengan warna merah pada gambar

diatas), pixel-pixel itu akan berisi nilai 255, padahal seharusnya pixel-pixel itu diisi oleh nilai yang mungkin berbeda karena menangkap level warna yang juga berbeda. Dengan kata lain, kita akan kehilangan detail warna pada subject, dalam kasus ini kita kehilangan detail pada daerah *highlight* (*clipped highlight*).

Untuk menghindari kehilangan detail pada daerah highlight, kita bisa saja mengurangi waktu *exposure*. Sehingga pixel tidak akan kelebihan beban saat mengambil daerah *highlight.* Akan tetapi apabila waktu *expsore* terlalu sebentar justru akan mengakibatkan pixel tidak memiliki waktu untuk mengambil daerah gelap (*shadow*) sehingga nilai *pixel* yang mengambil daerah gelap menjadi 0. Istilah ini disebut dengan *clipped shadow* (kehilangan detail pada daerah shadow/ gelap).



Gambar 2.16 Efek *Blooming* Pada Citra

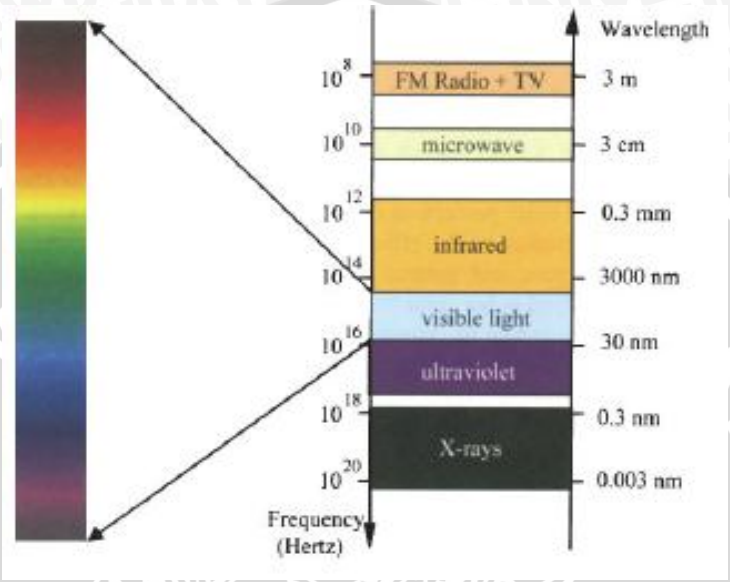
Sumber : http://cyberwayang.multiply.com

Pada contoh gambar di atas, terlihat detail batang ranting dan daun menjadi kabur, karena informasi asli yang disimpan oleh pixel menjadi *overflow* akibat menampung kelebihan *photon* dari *pixel* tetangganya yang menangkap cahaya terang, sehingga bagian yang seharusnya tidak terang menjadi *overexposed* juga. Efek *blooming* ini dapat semakin memperjelas munculnya efek-efek jelek lain, misalnya efek *purple fringing.*

2.8 *Laser*

Light Amplification by Stimulated Emission of radiation (LASER) adalah mekanisme suatu alat yang memancarkan radiasi elektromagnetik yang berupa cahaya tampak dan cahaya tak tampak. Berkas *laser* umumnya sangat koheren yaitu cahaya yang dipancarkan tidak meyebar dan dalam rentang frekuensi yang sempit (cahaya monokromatik) serta fokus dan sangat terang. Dalam proses

pendeteksian warna *laser* tidak dapat dilakukan secara warna mutlak misal jika warna *laser* merah maka komponen RGBnya 255 untuk kanal merah 0 untuk kanal hijau dan 0 untuk kanal biru, hal ini disebabkan oleh faktor cahaya, jarak dan kualitas sensor kamera, untuk itu proses pendeteksian dengan memakai range warna pada setiap kanal.



Gambar 2.17 Spektrum Elektromagnetik

Sumber: Andreas Koschan, Digital Color images Processing. 2007

2.8.1 Range Warna Merah *Laser*

Jangkauan warna *laser* merah dilihat dengan cara meng*capture* berkas *laser* kemudian kanal-kanal warnanya diuraikan dengan menggunakan *color picker* yang ada pada aplikasi photoshop, hasilnya didapat range untuk kanal merah > 240 untuk kanal hijau < 170 dan untuk kanal biru < 170.

Gambar 2.18 Range Warna Untuk Mendeteksi *Laser* Merah

Sumber : Mardhana, Luky. 2011. *Aplikasi Pendeteksi Laser pointer Untuk Menggerakan Kursor Mouse Pada Komputer.*

2.8.2 Range Warna Hijau *Laser*

Jangkauan warna *laser* hijau dilihat dengan cara meng*capture* berkas *laser* kemudian kanal-kanal warnanya diuraikan dengan menggunakan color picker yang ada pada aplikasi photoshop, hasilnya didapat range untuk kanal merah < 200 untuk kanal hijau > 240 dan untuk kanal biru < 200.

Gambar 2.19 Range Warna Untuk Mendeteksi *Laser* Hijau.

Sumber : Mardhana, Luky. 2011. *Aplikasi Pendeteksi Laser pointer Untuk Menggerakan Kursor Mouse Pada Komputer.*

2.9 Mouse

Mouse adalah perangkat penunjuk yang bekerja dengan mendeteksi dua dimensi gerak relatif terhadap permukaan pendukungnya. Implementasi fungsi *mouse* pada program menggunakan *library Windows API* yang salah satu fungsinya adalah untuk mengatur fungsi-fungsi kontrol dasar yang terdapat di *windows.*

Berbagai cara operasi *mouse* yang menyebabkan terjadinya *event* tertentu pada GUI:

1. *Clik*: menekan dan melepaskan tombol.
2. *Left Single Click* : mengklik tombol utama.
3. *Left Double Click* : mengklik tombol dua kali dalam jumlah berturut-turut.
5. *Right Click* : mengklik tombol sekunder.
6. *Middle Click* : mengklik tombol tengah.

Adapaun cara agar aplikasi dapat mengakses *mouse* dengan memanfaatkan *user32.dll* yang merupakan library *windows API.* Contoh perintah untuk mengakses mouse menggunakan library tersebut antara lain:

```
[DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention = CallingConvention.StdCall)]
    public stsatic extern void mouse_event(int dwFlags, int dx, int dy, int cButtons, int dwExtraInfo);

    private const int MOUSEEVENTF_RIGHTDOWN = 0x0008;
    private const int MOUSEEVENTF_RIGHTUP = 0x0010;
```

```
private const int MOUSEEVENTF_LEFTDOWN = 0x0002;
private const int MOUSEEVENTF_LEFTUP = 0x0004;
```

Sumber : www.code-project.com

Nama field MOUSEEVENTF_RIGHTDOWN, MOUSEEVENTF_RIGHTUP, MOUSEEVENTF_LEFTDOWN, MOUSEEVENTF_LEFTUP hanyalah sebuah penamaan field dan bisa diubah, sedangkan nilai 0x0008, 0x0010, 0x0002, 0x0004 merupakan kode ASCII yang apabila nilainya diubah maka akan menghasilkan *event mouse* yang berbeda. Kode ini dipakai agar dapat mengakses mouse event sesuai perintah nama fieldnya.

BAB III

METODOLOGI

Metode penelitian yang digunakan dalam penyusunan skripsi perancangan aplikasi mampu mendeteksi berkas sinar *laser* untuk menggerakkan kursor *mouse,* mengaktifkan *event* klik dan *scroll*. Metode penelitian yang digunakan dalam penyusunan skripsi ini adalah:

3.1 Studi Literatur

Studi literatur menjelaskan dasar teori yang digunakan untuk menunjang penulisan skripsi. Teori-teori pendukung tersebut meliputi:

1. *Image processing* dan *computer vision* untuk mengetahui metode apa saja yang akan digunakan untuk membuat aplikasi.
2. Metode pendeteksian warna pada citra, untuk mengetahui bagaimana suatu apilikasi dapat mendeteksi warna tertentu.
3. pemrograman dengan menggunakan Microsoft Visual Studio.NET. Untuk mengimplementasikan algoritma yang telah dirancang kedalam suatu bahasa pemrograman.

3.2 Perangkat Yang Digunakan

Menentukan perangkat yang digunakan untuk menunjang pembuatan aplikasi:

a) Perangkat Keras:
   1) 1 unit PC
      INTEL CORE 2 DUO, *mainboard* Jetway 945GCM2S-A2-6H, DDR II 2048MB, VGA GEFORCE 9500 1GB, harddisk 512 GB
   2) 1 buah *webcam*
      M-TECH : resolusi video: 640 x 480 piksel atau 320 X 240 piksel, *frame rate*: up to 30fps, mic: external *built in*, antarmuka: *USB*

b) Perangkat Lunak:
   1) OS Windows XP service pack 2
   2) Microsoft Visual Studio 2010.NET

3.3 Perancangan dan Implementasi Sistem

Secara garis besar desain yang akan dibuat untuk mengembangkan aplikasi pendeteksi berkas sinar *laser pointer* dengan menggunakan teknik seleksi piksel pada kanal RGB sebagai color filter dengan menggunakan *webcam*, perancangan aplikasi ini terdiri dari dua bagian, yaitu:

1. Memahami program pengolahan citra pada aplikasi sebelumnya.
2. Mengintegrasikan fungsi Windows API pada program agar aplikasi mampu membangkitkan *event* klik.

Dari kedua bagian perancangan tersebut, akan digabungkan dan dijalankan. Sehingga saat program mendeteksi adanya berkas sinar *laser pointer*, maka program akan mengolah dan mencari titik kordinat x,y piksel berkas sinar *laser* pada citra, setelah itu program mentranslasikan titik x,y citra ke dalam koordinat x,y layar komputer guna menggerakkan kursor kemudian memisahkan area khusus pada layar untuk membangkitkan *event* klik.

Berikut ini desain rancangan sistem dari Aplikasi pendeteksi *laser pointer* untuk menggantikan fungsi-fungsi mouse:

Gambar 3.1 Rancangan Sistem Pendeteksi *Laser Pointer* Secara Keseluruhan

Sumber : Perancangan

Cara kerja dari aplikasi pendeteksi sinar *laser pointer* agar dapat mengakses *mouse event* :

1. *Webcam* di gunakan oleh *user* untuk mengambil *video* pada layar proyektor tempat *laser* bergerak dengan bantuan perangkat lunak yang terdapat pada komputer. *webcam* sebaiknya dipasang tegak lurus dengan layar agar mendapatkan posisi citra yang tepat untuk memudahkan pengkalibrasian. *Webcam* digunakan sebagai penangkap citra berkas sinar *laser* dengan bantuan perangkat lunak yang terdapat dalam komputer. kemudian *webcam* akan melakukan *capturing* dan merekam dalam bentuk file *video* yang kemudian dicuplik frame-framenya berupa data bitmap untuk diolah.
2. Frame-frame *bitmap* yang dicuplik *webcam* digunakan sebagai data masukan dari program yang akan diolah
3. Keluaran dari program adalah berupa koordinat x,y dari kedua jenis *laser* hasil pengolahan citra yang digunakan untuk mengerakkan kursor *mouse* komputer dan membangkitkan *mouse event.*

Proses selanjutnya adalah implementasi, yaitu proses transformasi hasil perancangan perangkat lunak yang telah dibuat ke dalam kode *(coding)* sesuai dengan sintaks dari bahasa pemrograman yang digunakan.

3.4 Pengujian dan Analisis

Pengujian dan analisis dilakukan untuk menemukan kesalahan aplikasi yang dibuat. Sehingga sebelum aplikasi yang dibuat digunakan oleh pengguna, aplikasi ini diharapkan hanya terdapat sedikit kesalahan atau bahkan tidak ada pada saat digunakan nantinya.

3.5 Pengambilan Kesimpulan Dan Saran

Kesimpulan diperoleh dari data-data hasil setelah semua analisis dan pengujian aplikasi dilakukan, sehingga diperoleh saran yang berguna untuk perbaikan aplikasi pada penelitian selanjutnya.

BAB IV

PERANCANGAN DAN IMPLEMENTASI

Perancangan dan implementasi aplikasi pendeteksi sinar *laser* dikerjakan dengan beberapa tahap, antara lain metode inisialisasi *webcam*, pengolahan citra, dan kontrol *mouse*, sedangkan bagian yang telah dilakukan pada aplikasi sebelumnya adalah :

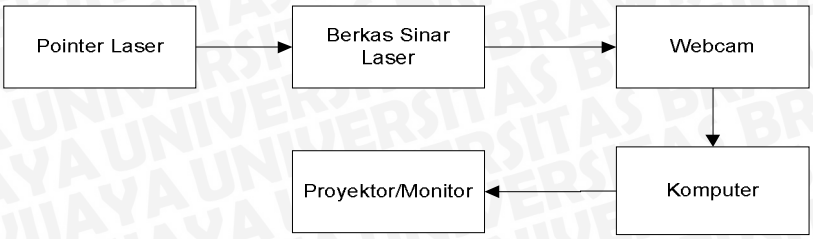
1. Perancangan Inisialisasi *webcam*, sehingga aplikasi dapat mendeteksi *webcam* yang terpasang pada komputer agar dapat digunakan.
2. Perancangan program pengolahan citra yang terdiri dari:
   a. Pengolahan citra bagian operasi *bitmap*. Sehingga aplikasi dapat mengolah suatu citra yang akan diproses
   b. Pengolahan citra bagian *cropping* dan *resize*. Sehingga aplikasi dapat merubah ukuran citra untuk melakukan penyesuaian dengan layar LCD proyektor.
   c. Pengolahan citra bagian *color filter*. Sehingga aplikasi dapat menemukan warna berkas *laser* yang diinginkan.

4.1 Perancangan Secara Umum

Perancangan aplikasi secara umum merupakan tahap awal sebagai acuan dalam perancangan aplikasi yang akan dibuat. Perancangan ini didahului dengan pendefinisian kegiatan pelaku atau *user* dalam menggunakan program pendeteksi sinar *laser pointer* untuk menangani *mouse* komputer dengan menggunakan teknik pengolahan citra serta perangkat yang digunakan meliputi blok sistem diagram dan cara kerja aplikasi.

4.1.1 Blok Sistem Diagram

Garis besar desain yang akan dibuat untuk aplikasi pendeteksi *laser pointer* melalui *webcam* terdiri dari beberapa langkah. Berikut ini blok diagram pendeteksi *laser pointer*, ditunjukkan dalam Gambar 4.1.



Gambar 4.1 Diagram Blok Sistem Secara Umum

Fungsi masing-masing bagian dalam diagram blok ini adalah sebagai berikut:

1. *Laser Pointer* di pakai sebagai sumber cahaya dalam aplikasi ini. Jenis *laser* yang di gunakan adalah *laser* diode dengan spesifikasi green *laser pointer* class III dengan panjang gelombang 532 nm ± 10
2. Berkas sinar *laser* yang akan diamati oleh sistem.
3. Dalam sistem ini piranti masukan yang dipakai adalah *webcam* yang digunakan untuk meng-*capture* citra.
4. Komputer berfungsi untuk melakukan mekanisme pengambilan citra dengan media *webcam*. Kemudian diolah (*image processing*) untuk menentukan bagian citra yang diamati, dan menentukan koordinat titik *laser*.
5. Pancaran dari proyektor berfungsi sebagai acuan untuk menentukan area kerja aplikasi.

4.1.2 Cara Kerja Aplikasi

Cara kerja aplikasi pendeteksi sinar *laser pointer* dengan menggunakan teknik pengolahan citra, dimulai dengan proses pengambilan citra atau peng-*capturan* menggunakan *webcam*. Yaitu dengan cara meletakkan *webcam* di depan layar atau dinding tempat berkas sinar *laser* ditembakkan. *Webcam* digunakan sebagai sensor untuk menangkap berkas sinar *laser*. Dalam pengambilan citra pada *webcam*, hal-hal yang harus diperhatikan dalam memasang perangkat yaitu posisi *webcam* sebaiknya sejajar dan tegak lurus terhadap layar screen atau dinding agar diperoleh citra yang tepat kotak persegi empat dan tidak trapesium untuk menyederhanakan kalibrasi. Pendeteksian warna memanfaatkan *open source computer vision library* dari *AForge* untuk menyeleksi piksel pada kanal RGB dan menemukan posisi kordinat sinar *laser*

pada citra. Informasi posisi koordinat yang ditemukan nantinya akan digunakan untuk menggerakan kursor dan mengaktifkan *event*.

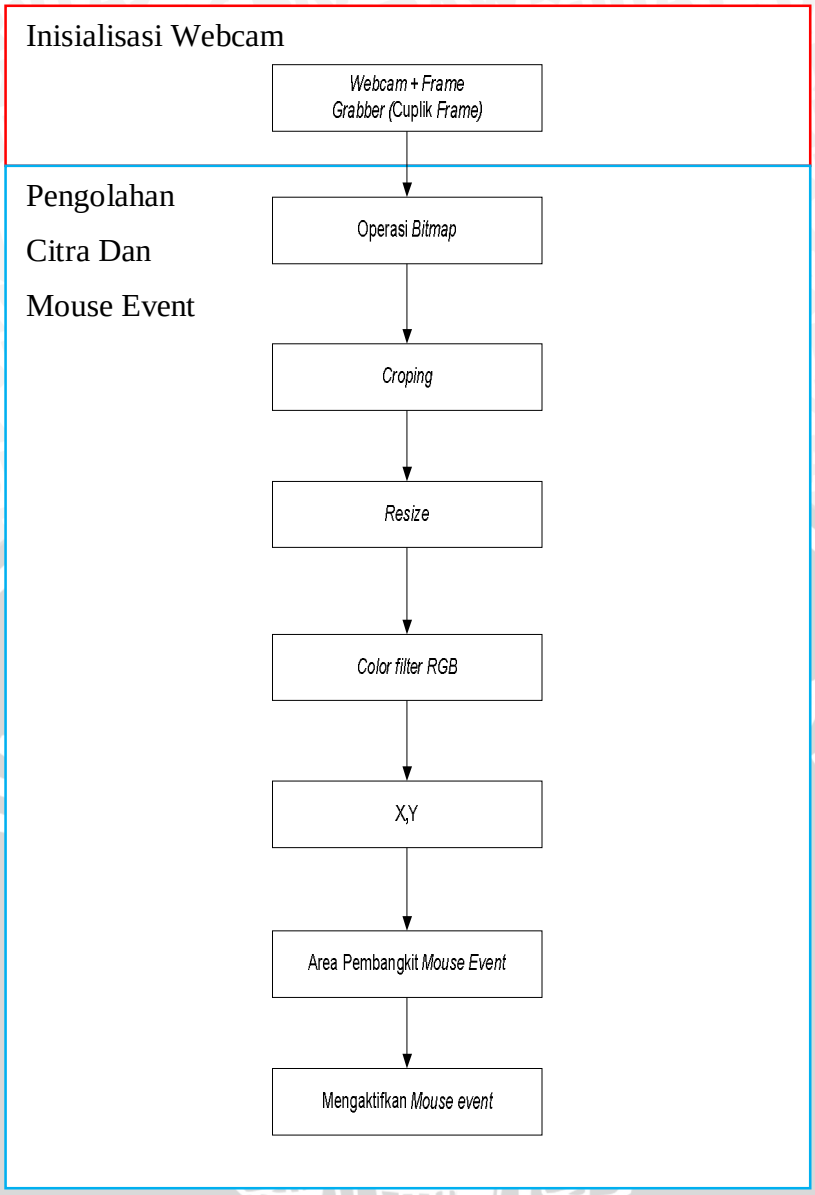
Proses inisialisialisasi *webcam* menggunakan komponen *Direct Show* framework yang dibuat oleh Andrew Kirilov, komponen ini juga berfungsi untuk merubah citra video menjadi frame-frame citra *bitmap* yang nantinya diproses dengan teknik pengolahan citra. Teknik pengolahan citra yang akan dilakukan berupa *cropping*, *resize*, dan *color filter*.

4.2 Perancangan Aplikasi

Perancangan aplikasi dibangun dengan menggunakan bahasa pemrograman Microsoft Visual Studio C# 2010. Sistem ini dirancang dengan spesifikasi sebagai berikut :

1. Mengakses perangkat keras Windows Driver Model (WDM) *webcam*.
2. Melakukan *cropping* untuk menentukan daerah kerja sistem.
3. Melakukan *resize* untuk menentukan skala citra .
4. Melakukan *color filter* pada kanal *red*, *green*, dan *blue*.
5. Melakukan pembagian area pembangkit *mouse event*
6. Menentukan *event* yang diambil berdasarkan koordinat warna *laser* yang terdeteksi pada citra yang diamati .

Proses pembuatan program dibagi menjadi 2 bagian yaitu inisialisasi *webcam*, dan pengolahan citra untuk membangkitkan *mouse event*. Detail desain aplikasi secara umum ditunjukkan pada Gambar 4.2



Gambar 4.2 Detail Desain Aplikasi

Sumber : Perancangan

4.2.1 Inisialisasi *Webcam*

Perancangan inisialisasi *webcam* dilakukan dengan memanfaatkan beberapa fitur komponen *Direct Show* yang dibuat oleh Andrew Kirillov (developer Aforge framework) sebagai antar muka perangkat keras. *Direct Show* adalah pustaka (library) *Component Object Model* (COM) untuk dapat mengakses perangkat keras berbasis *Windows Driver Model* (WDM) seperti *webcam*.

4.2.2 Pengolahan Citra Dan *Mouse Event*

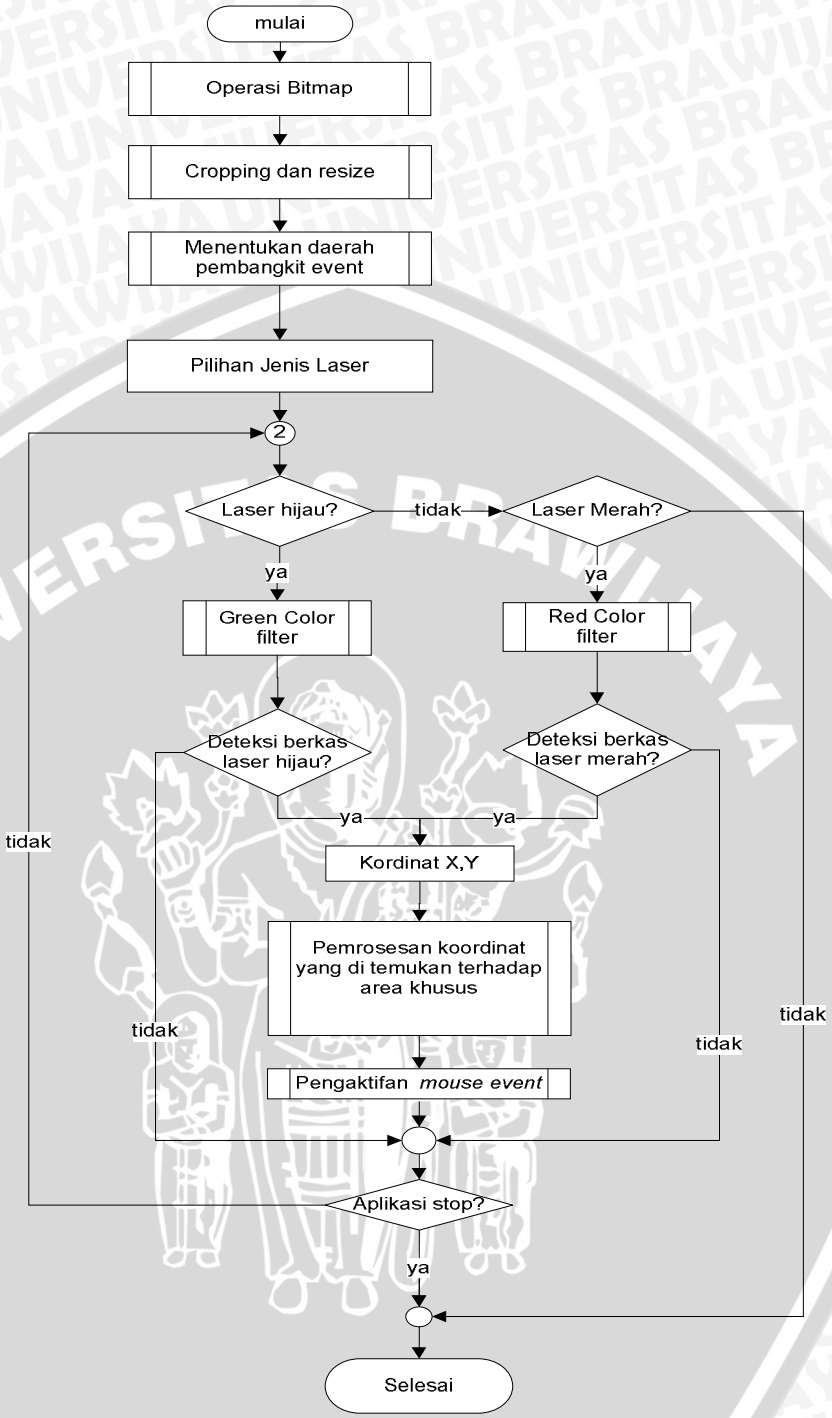
Pengolahan citra terdiri atas beberapa bagian yaitu operasi *bitmap*, cropping, *resize*, filter warna (seleksi piksel pada tiap kanal RGB) dan set kursor

(x,y). Operasi *bitmap* digunakan untuk mengakses secara langsung piksel beserta data informasi yang terletak di memori.

Pemotongan citra atau cropping digunakan sebagai kalibrasi sederhana guna membatasi area kerja aplikasi dalam proses pengolahan citra. *Resize* digunakan untuk mempercepat iterasi komputasi dalam mengakses semua piksel di dalam citra, dengan ilustrasi untuk jumlah piksel pada resolusi citra 360 X 240 adalah 76800 piksel, bila kita me-*resize* 1:2 maka didapatkan resolusi citra 160 X120 jumlah piksel yang diamati menjadi 19200 piksel, jadi nilainya empat kali lebih kecil dari 76800.

Setelah melewati proses *cropping* dan *resize*, kemudian adalah mengakses piksel-piksel pada citra *bitmap*, proses seleksi sinar *laser* menggunakan teknik seleksi piksel pada setiap kanal (RGB) kemudian dihitung nilai *grayscale* pada piksel yang diamati untuk memastikan bahwa citra yang di deteksi adalah berkas sinar *laser*. Untuk warna sinar *laser* hijau nilai kanal merah mendekati 0, nilai kanal hijau mendekati 255 dan nilai kanal biru mendekati 0. Sedangkan warna sinar *laser* merah untuk kanal merah mendekati 255, kanal hijau mendekati 0 dan nilai kanal biru mendekati 0.

Output dari proses *color filtering* berupa koordinat (x,y). Kordinat (x,y) yang ditemukan kemudian dicek posisinya apakah koordinat berada pada area pembangkit *event* atau tidak. Pengecekan posisi koordinat (x,y) terhadap area pembangkit *event* bertujuan untuk menentukan *mouse event* mana yang akan aktif. Panjang dan lebar wilayah area pembangkit *mouse event* adalah sepersepuluh dari panjang dan lebar citra *bitmap* yang digunakan dan diletakan pada keempat sudut citra *bitmap*. Untuk membangkitkan *mouse event* digunakan *library* Windows API dengan cara mengintergrasikan fungsinya kedalam program yang dibuat. Flowchart bagian pengolahan citra dan *mouse event* dapat dilihat pada gambar 4.3.



Gambar 4.3 Flowchart Bagian Pengolahan Citra Dan *Mouse Event*

Sumber: Perancangan

4.2.2.1 Operasi *Bitmap*

Class *bitmap* pada GDI+ menyediakan fitur method lockBits dan unlockBit yang memungkinkan untuk memanipulasi sebagian dari array *bitmap*

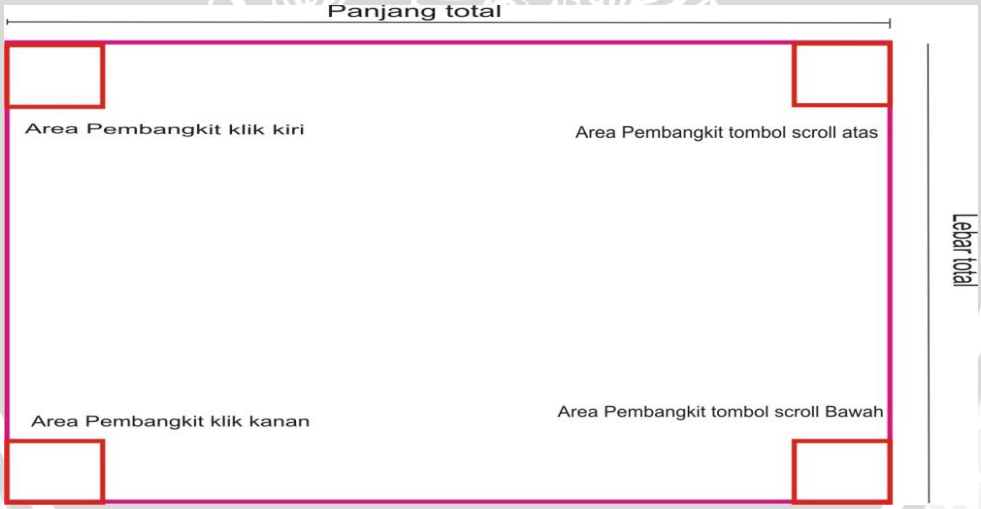
piksel secara langsung pada memori, akses langsung ini dapat berupa seleksi piksel atau mengganti bit-bit pada *bitmap* dengan data yang sudah dimodifikasi.

4.2.2.3 Cropping dan *Resize*

Proses ini dibuat untuk melakukan potongan pada citra yang di seleksi oleh *user*, sehingga menghasilkan citra baru yang akan diproses pada program. Citra tersebut di seleksi oleh *user* untuk menentukan area dari pancaran proyektor, sehingga *user* lebih mudah untuk menyesuaikan citra yang akan ditangkap terhadap layar LCD proyektor.

4.2.2.4 Perancangan Area Pembangkit *Mouse Event*

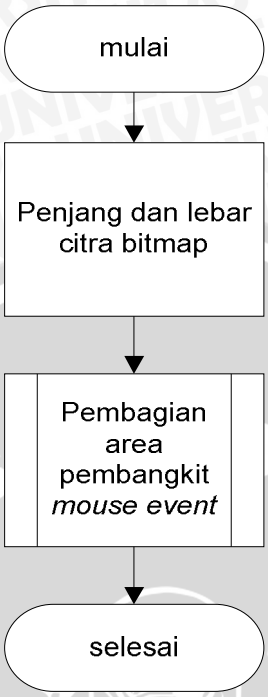
Dalam melakukan penentuan area pembangkit *mouse event*, hal pertama yang dilakukan adalah mengetahui panjang dan lebar citra yang kita peroleh dari Proses *croping* dan *resize*, setelah itu menentukan batasan batasan area khusus sebagai pembangkit *event*. Area khusus yang dibuat dapat dilihat pada gambar 4.4.



Gambar 4.4 Area Pembangkit Mouse Event

Sumber : Perancangan

Panjang dan lebar area untuk membangkitkan *mouse event* adalah $\frac{1}{10}$ dari panjang dan lebar citra bitmap yang digunakan, kemudian area tersebut diposisikan pada keempat pojok sisi citra. Flowchart area pembangkit *mouse event* dapat dilihat pada gambar 4.5.



Gambar 4.5 Flowchart Pembagian Area Pembangkit Event

Sumber : Perancangan

Penjelasan flowchart:

1. Menemukan panjang dan lebar citra *bitmap*.
2. Melakukan pembagian area pembangkit *event*, dimana panjang dan lebarnya adalah $\frac{1}{10}$ dari panjang dan lebar citra bitmap yang digunakan, kemudian area tersebut diposisikan pada keempat pojok sisi citra.
3. Selesai

Implementasi pembagian area pembangkit *event* menggunakan bahasa pemrograman C# di jelaskan sebagai berikut :

1. Membuat objek citra agar dapat dibuat gambar area pembangkit *event.*

```
Graphics areaKlik = Graphics.FromImage(tmpImage1);
Pen pen = new Pen(Color.Red, 2);
```

2. Menentukan luas tiap-tiap area pembangkit *event.*

```
xscrollats=panjangCitra*9/10;
xscrollbwh=panjangCitra*9/10;
```

```
ykanan= lebarCitra*9/10;
xjarak=panjangCitra*1/10;
yjarak=lebarCitra*1/10;
```

3. Menggambar area pembangkit *event* sesuai dengan luas yang telah di tentukan.

```
areaKlik.DrawRectangle(pen, 0, 0, xjarak, yjarak); //kotak klik atas
areaKlik.DrawRectangle(pen, 0, ykanan, xjarak, yjarak); //kotak klik bawah
areaKlik.DrawRectangle(pen, xscrollats, 0, xjarak, yjarak); //kotak scroll atas
areaKlik.DrawRectangle(pen, xscrollbwh, ykanan, xjarak, yjarak); //kotak scroll bawah
areaKlik.Dispose();
```

Sumber : Perancangan

4.2.2.5 Pengolahan Citra Bagian *Color Filter*

Aplikasi sebelumnya telah berhasil menemukan warna sinar *laser* yang dicari pada citra dengan cara melakukan iterasi satu persatu terhadap pixel pada citra yang ditangkap oleh *wabcam*. Jika warna sinar *laser* yang dicari pada citra berhasil ditemukan, maka program akan menyimpan posisi koordinat ditemukanya warna sinar *laser* yang kemudian digunakan sebagai acuan untuk menentukan *mouse event* yang akan diaktifkan.

1. Mendapatkan panjang dan lebar citra.

```
panjangCitra = tmpImage1.Width;
lebarCitra = tmpImage1.Height;
tmpImage0.Dispose();
```

2. Langkah untuk mengakses piksel.

```
warna.OperasiBitmap operasiBitmap = new warna.OperasiBitmap(tmpImage1);
bool pixelLaserDitemukan = false;
operasiBitmap.LockBitmap();
int xPosisi = 0;
int yPosisi = 0;
```

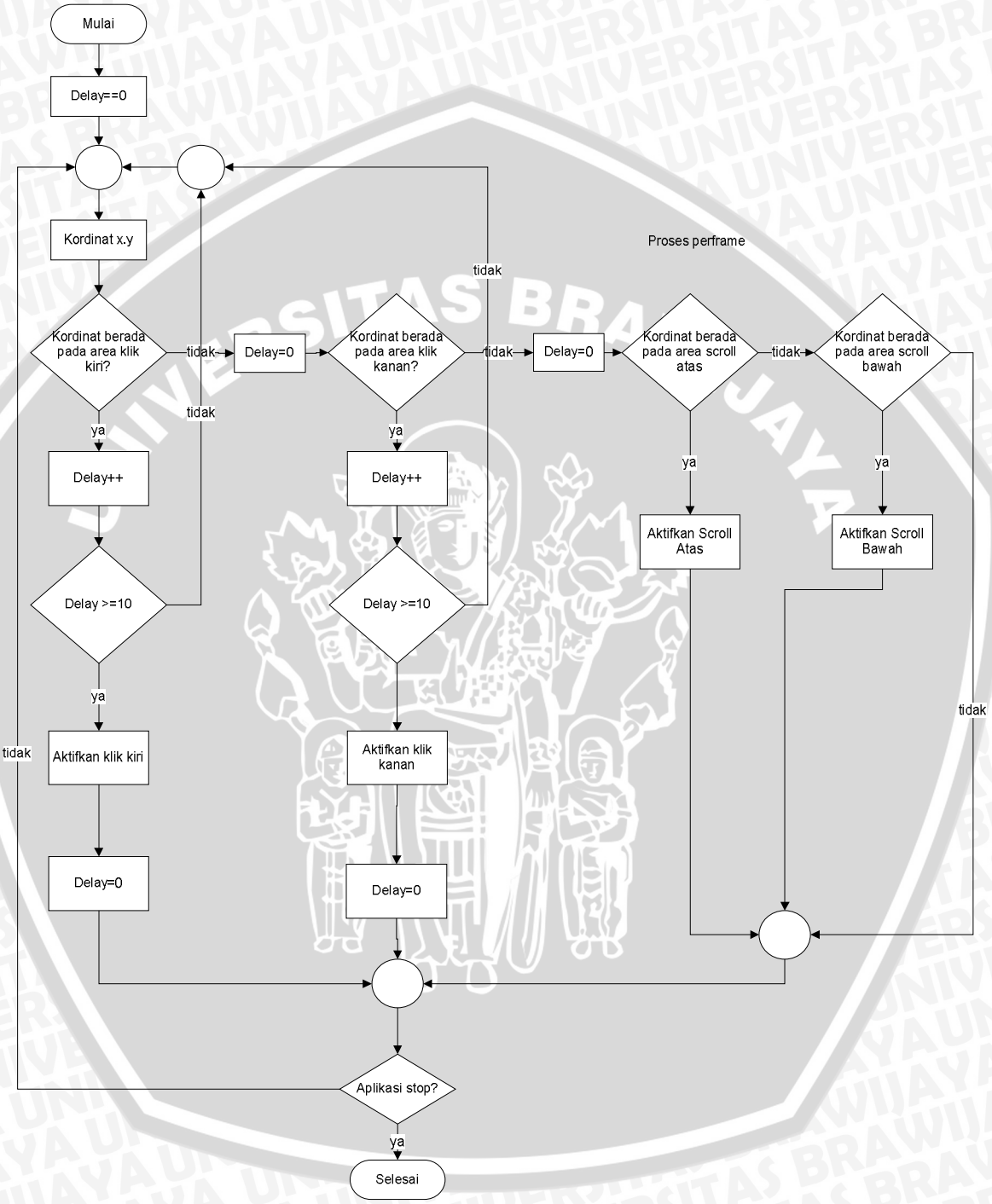
3. Gambar *true color* pada dasarnya terdiri dari 3 unsur yaitu merah, hijau, dan biru. Setiap piksel yang membentuk gambar terdiri dari campuran ketiga warna ini dengan intensitas yang berbeda satu dengan lainya. Untuk *green laser* agar warna hijau menyala saja yang difilter maka komparasi komponen R< 200 AND G>240 AND B<200. Untuk Warna Laser merah agar warna merah menyala saja yang difilter maka komparasi warna komponennya R>220

AND G<170 AND B<170. Sedangkan untuk *tune color* mode hanya memvariabelkan nilai-nilai pada RGB.

```
if((_formUtama.laserHijau == true)
 ||(_formUtama.customLaserHijau== true))
{
for (int y = 0; y < lebarCitra; y += 1)
{
for (int x = 0; x < panjangCitra; x += 1)
{
byte merah, hijau, biru;
merah = operasiBitmap.GetPiksel(x, y).red;
hijau = operasiBitmap.GetPiksel(x, y).green;
biru  = operasiBitmap.GetPiksel(x, y).blue;
float tesPiksel = (299 * merah + 587 * hijau + 114 * biru) /
    1000;
if (tesPiksel> _formUtama.threshold)
{
if ((merah <= 200) && (hijau >= 240) && (biru <= 200)) //laser hijau
        {
         xPosisi = x;
         yPosisi = y;
         pixelLaserDitemukan = true;
        }}
} }// x,y  loop
```

Sumber : *"Aplikasi Pendeteksi Laser pointer untuk Menggerakkan Kursorpada Komputer"* (Luky Mardhana, ST. 2011)

4.2.2.6 Pembangkit *Event* klik dan *scroll*



Gambar 4.6 Flowchart Pembangkit *Mouse Event* Klik dan *Scroll*

Sumber : Perancangan

Berikut Penjelasan dari flowchart pembangkit *event* klik dan *scroll*

1. Menetapkan nilai awal *delay*=0 dimana nilai *delay* digunakan sebagai syarat kondisi untuk mengaktifkan *event* klik.
2. Dari proses *colour filtering*, akan diperoleh koordinat x,y yang akan diproses pada tahap selanjutnya.
3. Program melakukan pengecekan, apakah koordinat x,y berada didalam area pembangkit event, mulai dari pengecekan area klik kiri, area klik kanan, area *scroll* atas dan area *scroll* bawah
4. Jika koordinat x,y yang ditemukan berada pada area pembangkit *event* klik maka terjadi penambahan nilai satu terhadap jumlah variabel *delay*, variable *delay* ditambahkan jika koordinat yang di temukan berada pada area pembangkit *event klik*. Dan apabila *laser* hilang atau tidak di temukan pada frame selanjutnya, maka nilai variabel delay akan dikembalikan menjadi 0. Hal ini di lakukan agar ketika *user* ingin mengaktifkan *event klik*, maka *user* harus mengarahkan *laser pointer* ke area yang dituju selama proses 10 frame secara tidak terputus.
5. Kemudian dilakukan pengecekan variable *delay*, apakah melebihi 10, jika ia maka program akan membangkitkan *event* klik sesuai dengan area pembangkit *event*.
6. Setelah *event* klik di aktifkan , maka program akan mereset nilai variabel *delay* menjadi 0,.
7. Khusus pada area pembangkit *scroll* tidak digunakan variable *delay*, karena aplikasi dinilai berjalan tidak efektif jika menggunakan *delay* karena membutuhkan waktu yang terlalu lama untuk mengaktifkan *event scroll*.
8. Selesai

Implementasi pembangkit *event* klik dan *scroll* menggunakan bahasa pemrograman C# di jelaskan sebagai berikut :

1. Menggunakan *library* Windows API pada program aplikasi dengan cara mengimport *user*32.dll yang berfungsi mengatur kontrol dasar (mouse dan keyboard) yang terdapat pada windows:

```
[DllImport("user32.dll", CharSet = CharSet.Auto, CallingConvention =
CallingConvention.StdCall)]
```

```
public static extern void mouse_event(int dwFlags, int dx, int dy, int
cButtons, int dwExtraInfo);
```

2. Menginisialisasi *field* – *field* pembangkit event.

```
private const int MOUSEEVENTF_RIGHTDOWN = 0x0008;
private const int MOUSEEVENTF_RIGHTUP = 0x0010;
private const int MOUSEEVENTF_LEFTDOWN = 0x0002;
private const int MOUSEEVENTF_LEFTUP = 0x0004;
```

3. Selanjutnya membuat method /fungsi untuk melakukan pengecekan posisi koordinat terhadap area pembangkit event.

a. Pembangkitan Area klik kiri :

```
if (xPosisi <= klikKiriX && yPosisi <= klikKiriY)
{
                    delleft++;
                    if (delleft == 15)
                    {
                   mouse_event(MOUSEEVENTF_LEFTDOWN, 0, 0, 0,  0);
                   mouse_event(MOUSEEVENTF_LEFTUP, 0, 0, 0, 0);
                    delleft = 0;
                    }
                }
                else
                {
                    delleft = 0;
                }
```

b. Pembangkit Area Klik Kanan :

```
klikkananx= panjangCitra*1/10;
klikkanany=lebarCitra*9/10;
if (xPosisi <=klikkananx  && yPosisi >=klikkanany )
                    {
                        delr++;

                        if (delr == 15)
                        {

                   mouse_event(MOUSEEVENTF_RIGHTDOWN, 0, 0, 0,0);
                   mouse_event(MOUSEEVENTF_RIGHTUP, 0, 0, 0, 0);
                             delr = 0;

                        }
                    }

                    else
                    {
                        delr = 0;
                    }
```

c. Pembangkit Area *Scroll* Atas :

```
scrollAtsX=panjangCitra * 9 / 10;
scrollAtsY =lebarCitra * 1 / 10 ;
                if (xPosisi >= scrollAtsX && yPosisi <= scrollAtsY)
```

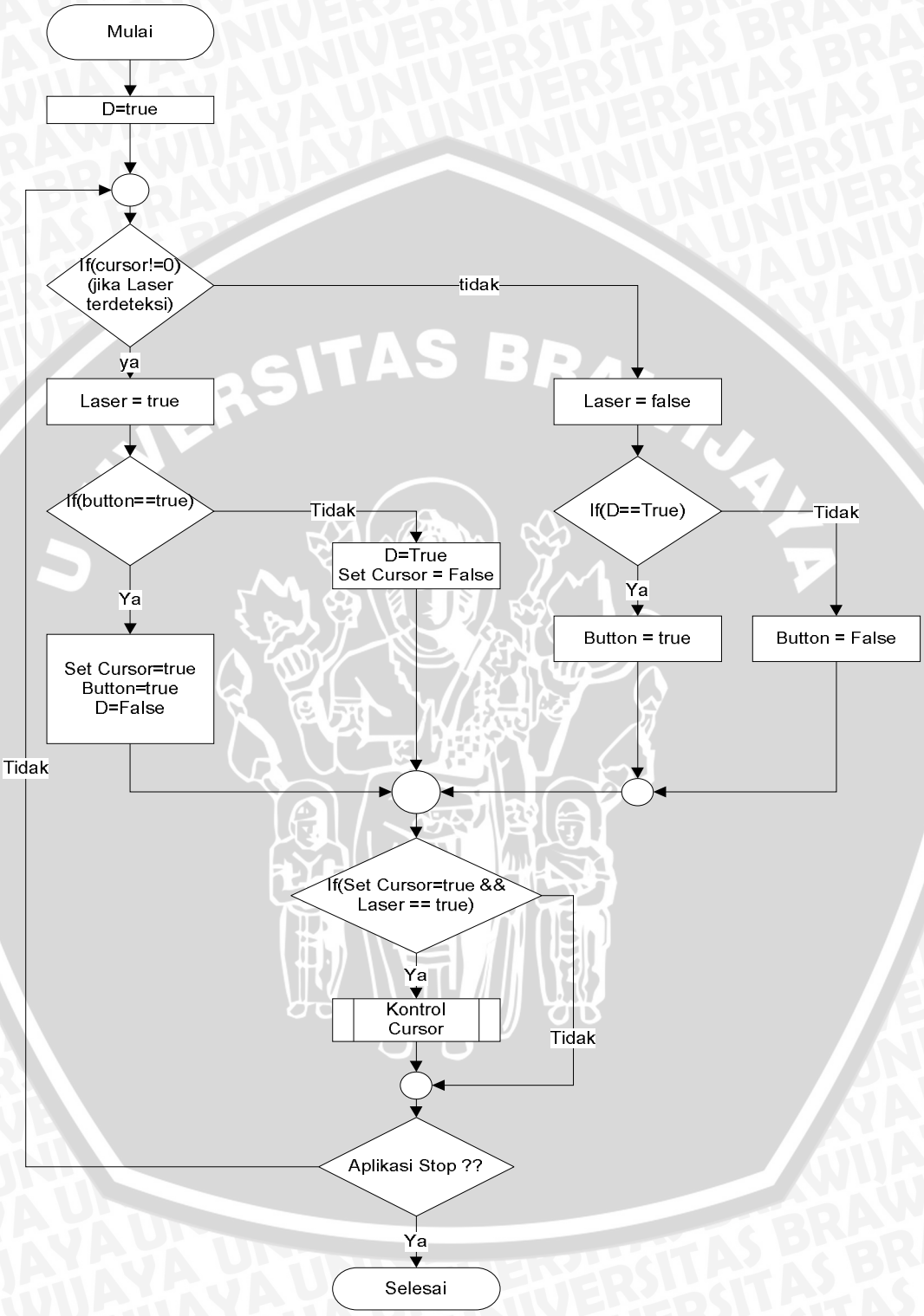
```
        {
            mouse_event(MOUSEEVENTF_WHEEL, 0, 0, 20, 0);
        }
```

d. Pembangkit Area *Scroll* Bawah :

e.

```
scrollBwhY=lebarCitra * 9/10 ;
if(xPosisi>=scrollAtsX && yPosisi >= scrollBwhY) {
mouse_event(MOUSEEVENTF_WHEEL, 0, 0, -20, 0); }
```

4.2.2.7 Pembangkit *Event* Kontrol Kursor



Gambar 4.7 Flowchart Pembangkit *Event* Kontrol Kursor

Sumber : Perancangan

Perancangan Algoritma tersebut dibuat untuk mengaktifkan proses kontrol kursor dan mematikan proses kontrol kursor. Hal ini bertujuan untuk mengatasi masalah yang ditemukan pada saat *user* akan mengaktifkan *event* klik agar kursor tidak selalu mengikuti berkas laser pointer ketika akan mengaktifkan *event* klik atau ketika *user* meleset dalam membidik area pembangkit *event*. Oleh karena itu dibuatlah perancangan seperti algoritma diatas agar kursor tidak mengikuti berkas *laser* yang tertangkap pada area pembangkin *event* klik maupun *scroll*, sehingga pergerakan kursor menjadi lebih fleksibel. Penjelasan algoritma tersebut dapat dijelaskan sebagai berikut:

1. Memberikan nilai awal pemicu kontrol kursor (D) dengan nilai True.
2. Melakukan pengecekan kondisi apakah *laser* terdeteksi atau tidak, jika ya maka nilai field *laser* = *true* dimana field ini akan digunakan sebagai salah satu syarat kondisi untuk membangkitkan kontrol kursor, jika *laser* tidak terdeteksi maka nilai *laser* = *false*.
3. Selanjutnya jika *laser* terdeteksi, maka sistem akan melakukan pengecekan nilai *field button*, jika *button* = true, maka nilai filed *set cursor* = *true*, *button* = *true*, dan nilai *field* pemicu D = *false*. Dan apabila *laser* tidak terdeteksi, maka nilai *laser* = false, kemudian program melakukan pengecekan kondisi nilai D, jika D=*true*, maka *buttton* = *true*, dan jika D= *false*, maka *button* = *false*.
4. Selanjutnya untuk megaktifkan kontrol kursor dilakukan pengecekan kondisi field *laser* dan *set cursor* , apakah keduanya bernilai true atau tidak, jika ia maka program akan menjalankan proses kontrol kursor, dan jika tidak memenuhi kondisi tersebut, maka program akan menjalankan kembali ke pengecekan pada point 2, sampai program dihentikan.

Implementasi pembangkit *event* kontrol kursor menggunakan bahasa pemrograman C# dijelaskan sebagai berikut :

1 . Pengecekan terhadap *laser* yang di temukan.

Nilai D=true, sudah diberikan sebagai nilai default pada saat program pertama di jalankan

```
if (xKoordinat != 0 && yKoordinat != 0)
        {
            if (saklar == true)
```

```
            {
                setkursor = true;//DISINI PEMBENTUK SAKLAR ON OFF
                saklar = true;
                cstop = 0;
            }
            else
            {
               D = true;
                setkursor = false;
            }
```

2. Jika *laser* yang dicari tidak di temukan

```
            }
            else
            {
                if (D == true)
                {
                    saklar = true;
                }
                else
                {
                    saklar = false;
                }

            }
```

3. Pengecekan terhadap yang berfungsi untuk membangkitkan kontrol kursor

```
            if (_formUtama.kontrolKursor == true && pixelLaserDitemukan ==
true
                    {
                if (setkursor == true)
                {
                    kontrolKursor(xPosisi, yPosisi);
                }
            }
```

4. Proses Kontrol Kursor (x,y)

Setelah didapatkan koordinat (x,y) hasil seleksi piksel pada citra, kemudian koordinat tersebut digunakan untuk menggerakkan kursor *mouse* dengan cara mentranslasi koordinat hasil seleksi citra kedalam koordinat layar utama komputer. Proses translasi dengan cara menghitung perbandingan lebar resolusi citra dengan lebar resolusi monitor serta panjang resolusi citra dengan panjang resolusi monitor. Semakin kecil nilai perbandingan skala citra, gerakan mouse semakin halus dan sebaliknya semakin besar nilai perbandingan maka perpindahan gerak kursor semakin kasar, dengan ilustrasi bila skala resize citra 1:2 dan input citra 640X480 maka citra hasil resize menjadi 320x240 piksel, bila resolusi monitor yang dipakai 800X600 maka setiap gerakan 10 pixel berkas *laser*

pada citra sama dengan 25 piksel pada monitor. Implementasi kontrol kursor menggunakan bahasa pemrograman C# dijelaskan sebagai berikut :

a) Mendapatkan panjang dan lebar monitor.

```
Int panjangLayar = Screen.PrimaryScreen.WorkingArea.Width;
int lebarLayar = Screen.PrimaryScreen.WorkingArea.Height+34;
```

Lebar layar area kerja windows tidak termasuk lebar menu taskbar, agar mendapatkan ukuran layar penuh maka tinggi yang didapatkan ditambahkan 34.

b) Menghitung perbandingan koordinat (x,y) berkas *laser* terhadap koordinat layar

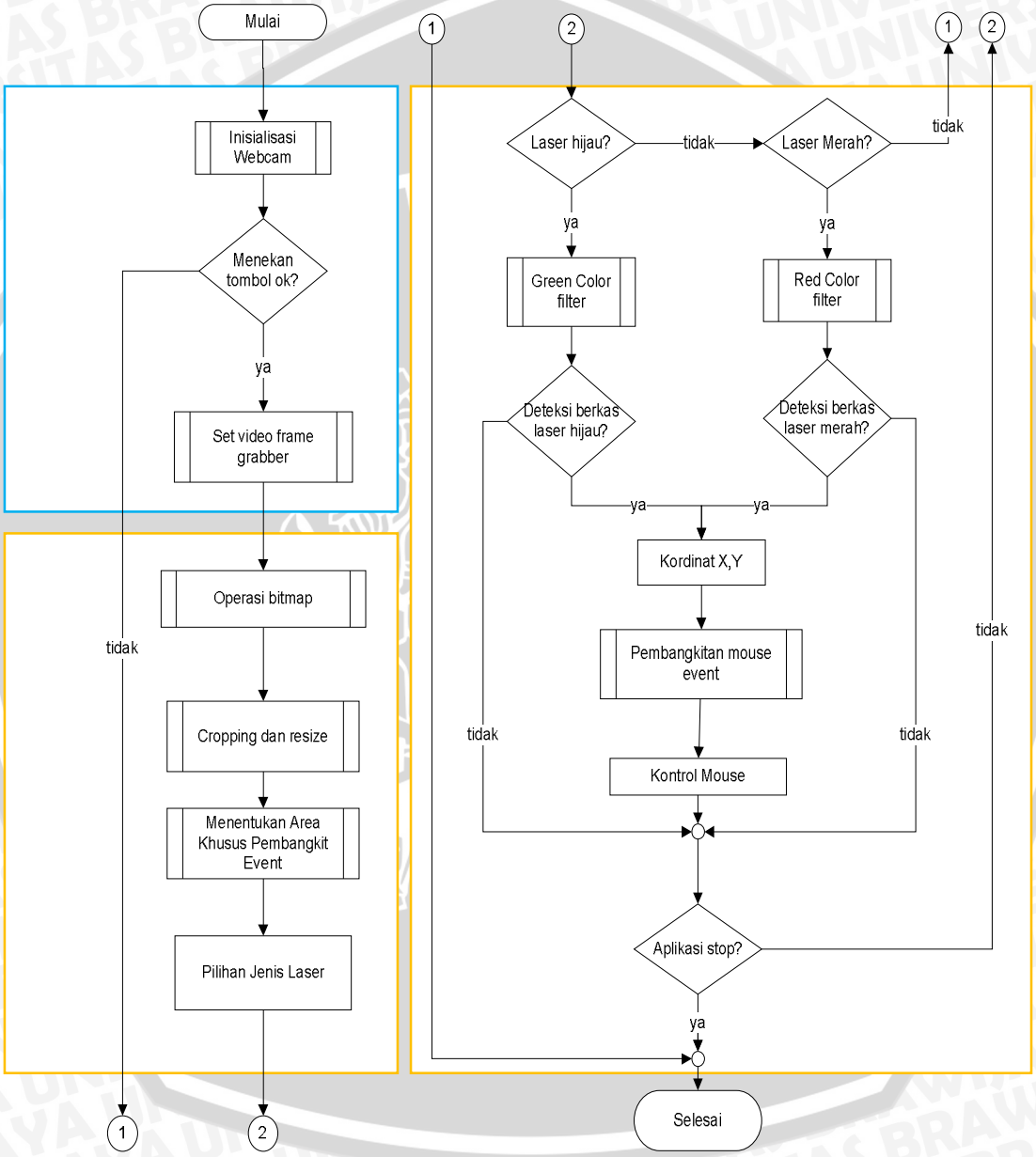
```
kursorX = ((float)panjangLayar / panjangCitra) * x;
kursorY = ((float)lebarLayar / lebarCitra) * y;
```

c) Set posisi kursor

```
Kursor.Position = new Point((int)kursorX, (int)kursorY)
```

4.3 Perancangan Program Keseluruhan

Setelah pembuatan perancangan inisialisasi *webcam*, pengolahan citra dan *event* pada perancangan umum, maka langkah selanjutnya adalah membuat kerangka bagian yang akan membentuk menjadi program yang utuh. Flowchart perancangan program secara keseluruhan ditunjukkan pada Gambar 4.8.



Gambar 4.8 Flowchart Program Keseluruhan

Sumber:Perancangan

4.4 Implementasi Sistem

Setelah tahap perancangan maka untuk selnjutnya adalah tahap implementasi. tahap implementasi merupakan proses transformasi dari proses perancangan yang telah dibuat sebelumnya dimana hasil perancangan ditransformasikan ke dalam bentuk bahasa pemrograman (*coding*) yang sesuai dengan sintaks yang ada pada bahasa pemrograman C# NET.

4.5 Lingkungan Implementasi

Aplikasi dibuat dengan menggunakan Microsoft Visual Studio C# dengan Aforge.NET framework. Sistem diimplementasikan dengan menggunakan spesifikasi sebagai berikut:

Perangkat keras :

A. Komputer

Spesifikasi :

- Processor : INTEL CORE 2 DUO processor 2.2 GHz
- Memory : 2024 MB RAM
- OS : Windows XP Service Pack II
- VGA : GEFORCE 9500 GT 1GB

B. *Webcam*

Spesifikasi:

- Model : M-TECH
- Resolusi : 1,3 megapiksel
- Frame/sec : 30 fps
- Video *Capture* : AVI - 640 x 480, AVI - 320 x 240

C. *Laser pointer*

Spesifikasi:

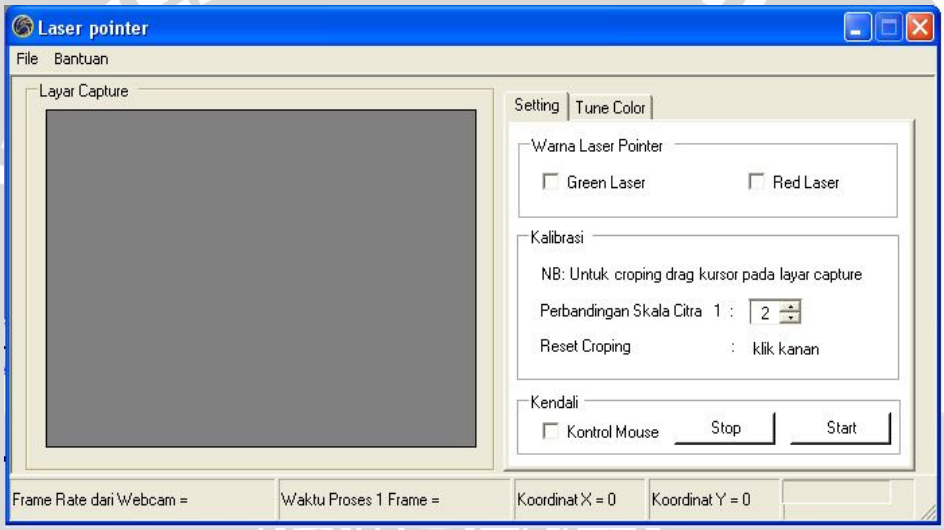
- Model : Green *Laser pointer*
- Output : Class III *Laser* Product max output<200mW
- Wavelenght : 532nm $\pm$ 10

Perangkat Lunak :

- Sistem operasi : Microsoft Windows XP Service Pack II
- Bahasa pemrograman: Microsoft Visual Studio C#.NET 2010
- Framework : Aforge .NET Framework

4.6 Implementasi Interface (Antarmuka)

Pada tampilan utama ini terdapat pilihan dua macam *laser* yaitu *green laser* dan *red laser*. Pilihan itu harus dipilih *user* sesuai dengan *laser pointer* yang akan dipakai. Perbandingan skala pada menu perbandingan skala antara lain 1:1 , 1:2, 1:3 dan 1:4. Untuk melakukan *cropping user* harus menentukan titik *cropping* dengan cara men-*drag* kursor *mouse* pada layar *capture* program secara manual dengan menggunakan *mouse*. Untuk mereset hasil cropping gunakan klik kanan pada *mouse*. Cek menu kontrol *mouse* untuk mengakses *mouse* dengan menggunakan *laser*. Aplikasi akses mouse komputer menggunakan *laser pointer* mempunyai tampilan utama seperti pada Gambar 4.9.

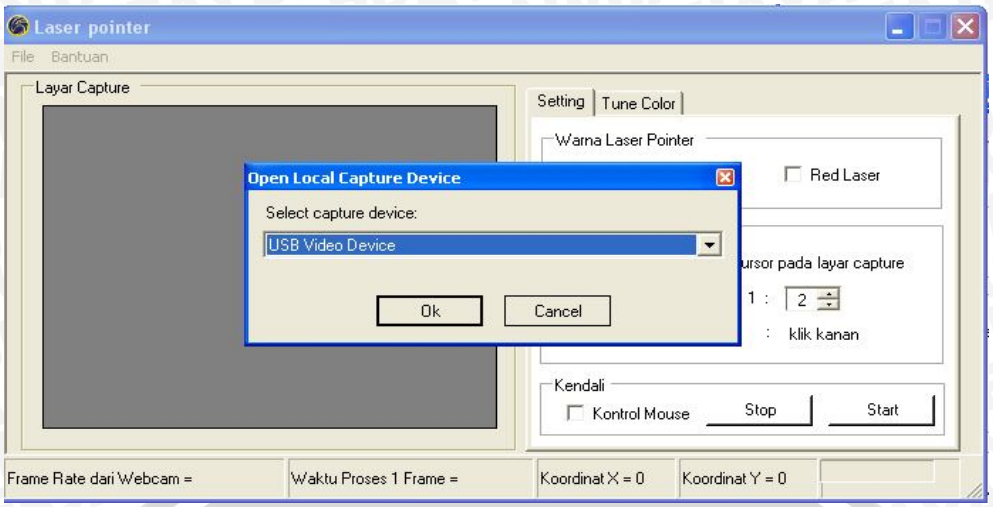


Gambar 4.9 Tampilan Program Utama Pendeteksi *Laser Pointer*

Sumber : Implementasi

4.6.1 Implementasi Antarmuka *Open Local Capture Device*

Langkah Pertama yang dilakukan sebelum mendeteksi warna adalah mengaktifkan *webcam* pada program ini. Proses pengaktifan dengan cara mengklik pada panel menu strip yaitu *file* →Inisialisasi kamera yang akan memunculkan dialog box, pilih *webcam* yang digunakan kemudian klik *button* OK,seperti pada gambar 4.10.



Gambar 4.10 Tampilan Form *Open Local Capture Device*
Sumber: Implementasi

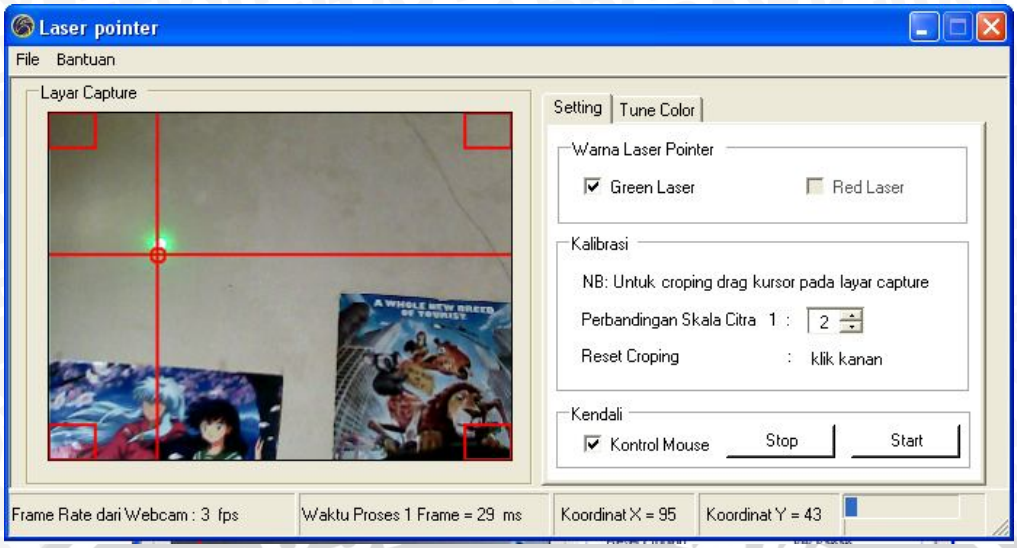
4.6.2 Implementasi Antarmuka untuk Melakukan *Tune Color*

Penggunaan *tune color* bila hasil pendeteksian warna *laser pointer* secara *default* kurang presisi akibat pengaruh intensitas cahaya pada daerah *tracking*. *User* dapat menentukan sendiri parameter nilai pada tiap-tiap *channe*l RGB dan *brightness threshold* secara manual. Seperti ditunjukkan pada Gambar 4.16.

Gambar 4.11 Tampilan *Tune Color* Pada Program Utama
Sumber: Implementasi

4.6.3 Implementasi Tampilan Saat Program Berjalan

Pada saat program berjalanmaka akan muncul gambar kotak merah kecil pada tiap sudut citra. Kotak tersebut adalah area pembangkit *event* yang akan membangkitkan *event* ketika *laser* berada pada area tersebut.



Gambar 4.12 Tampilan Program Utama Saat Aplikasi berjalan

Sumber: Implementasi

BAB V

PENGUJIAN

Untuk mengetahui apakah sistem bekerja dengan baik dan sesuai dengan perancangan, maka diperlukan serangkain pengujian. Pengujian yang dilakukan adalah sebagai berikut:

1. Pengujian *webcam.*
2. Pengujian FPS *(frame per-second).*
3. Pengujian program pendeteksi sinar *laser* terhadap kondisi penerangan.
4. Pengujian *skala resize* terhadap kecepatan proses citra.
5. Pengujian *skala resize* terhadap hasil *tracking* dan waktu yang dibutuhkan untuk mengaktifkan *event.*
6. Analisis faktor kegagalan.
7. Kesimpulan hasil pengujian.

5.1 Pengujian Kamera

Pengujian ini bertujuan untuk mengetahui apakah aplikasi dapat mendeteksi berkas laser dengan menggunakan kamera yang bebeda. Kamera yang digunakan pada pengujian sebelumnya yaitu :

1.Acer Crystal Eye
2.Lexcron Vimicro

Sedangkan untuk membandingkan hasil proses citra pada pengujian ini digunakan tiga jenis Kamera yang berbeda, untuk mengetahui apakah aplikasi bisa mendeteksi berkas sinar laser atau tidak pada tiap-tiap kamera yang digunakan. Kamera yang digunakan yaitu:

1.M-tech (kamera kualitas menengah)

2.G-Star (kamera kualitas rendah)

3. HandyCam Sony 9 MB (kamera kualitas tinggi)

Tabel 5.1 Pengujian Pengaruh Kamera Terhadap Pendeteksian Berkas Sinar *Laser*

Kamera	Status
M-Tech	Terdeteksi
G-STar	Terdeteksi
Sony Handycam	Terdeteksi

Sumber: Pengujian

Pada pengujian tersebut digunakan *software manycam* yang berfungsi sebagai *webcam virtual*. Hal ini dilakukan karena keterbatasan dalam memperoleh *webcam* dengan kualitas tinggi. Sehingga citra tidak diproses secara langsung oleh kamera, melainkan disimpan terlebih dahulu dalam bentuk video.

5.2 Pengujian FPS *Video*

Pengujian dilakukan untuk mengetahui frame per detik (fps) yang dihasilkan oleh masing-masing *video* yang ditangkap menggunakan tiga jenis kamera yang berbeda bila perbandingan skala yang digunakan bebeda-beda. Perubahan perbandingan skala *resize* yang dipakai adalah 1:1, 1:2, 1:3 dan 1:4. Data hasil pengujian FPS *webcam* ini diperlihatkan pada Tabel 5.2

Tabel 5.2 Pengujian Pengaruh Skala *Resize* Terhadap FPS

No	*Webcam*	Skala *Resize*	FPS
1	M-Tech (1.3MB)	1:1	7
		1:2	9
		1:3	10
		1:4	10
2	G-Star (VGA)	1:1	7
		1:2	10
		1:3	10
		1:4	10
3	Handycam Sony(9MB)	1:1	7
		1:2	9
		1:3	10
		1:4	10

Sumber : Pengujian Aplikasi

Hasil tabel pengujian diperoleh FPS yang berbeda-beda pada setiap jenis kamera, yaitu semakin tinggi nilai perbandingan skala maka semakin banyak frame yang dihasilkan perdetiknya.

5.3 Pengujian Pendeteksian Warna *Laser* Terhadap Kondisi Penerangan

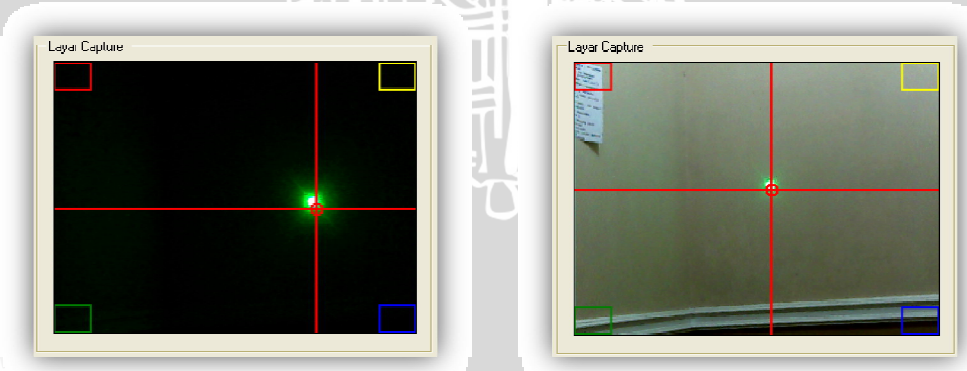
Pada pengujian ini intensitas cahaya ruangan dikondisikan menjadi dua kondisi, yaitu kondisi tanpa penerangan dimana kondisi pencahayaan ruang yang dipantau gelap, dan kondisi dengan penerangan yaitu kondisi pencahayaan ruang dengan penerangan satu lampu Philips putih 20 watt. Sedangkan jarak *webcam* terhadap dinding atau layar pantau dibuat tetap.

Tabel 5.3 Pengujian Respon Program terhadap Intesitas Penerangan Ruangan

Warna *Laser*	Intensitas Penerangan	Respon program
Hijau	Tanpa penerangan	Terdeteksi
	Dengan penerangan	Terdeteksi
Merah	Tanpa penerangan	Terdeteksi
	Dengan penerangan	Terdeteksi

Sumber: Pengujian

tabel pengujian di atas menunjukan bahwa program dapat mendeteksi berkas sinar *laser* dengan kondisi tanpa penerangan maupun dengan penerangan.



Gambar 5.1 Pengujian Terhadap Kondisi Penerangan Ruangan

Sumber : Pengujian

5.4 Pengujian Pengaruh Skala *Resize* Terhadap Kecepatan Proses Citra

Untuk mengetahui pengaruh skala *resize* terhadap waktu pemrosesan satu *frame*, maka dilakukan pengujian dengan mengubah nilai skala *resize* pada *aplikasi*. Pengujian dilakukan dengan menggunakan tiga jenis kamera yang memiliki kualitas berbeda dengan intensitas cahaya tetap. Kamera digunakan untuk merekan video berkas sinar *laser* dan *capture* dari jarak 2 meter. Hasil pengujian ditunjukkan pada tabel 5.4

Table 5.4 Pengujian Pengaruh Skala *Resize* Terhadap Kecepatan Proses 1 *Frame*

Jenis *webcam*	Skala *Resize*	Kecepatan proses 1 frame dalam milidetik (ms)
M-Tech 1 MB (640 X 480)	1:1	30
	1:2	7
	1:3	3
	1:4	2
G-STAR VGA (320 X 240)	1:1	29
	1:2	8
	1:3	3
	1:4	2
Handycam Sony (9 MB)	1:1	30
	1:2	7
	1:3	3
	1:4	2

Sumber : Pengujian

Pada tabel diatas, dapat dilihat bahwa semakin besar nilai perbandingan skala, maka semakin cepat proses tiap satu framenya. Untuk *video* dengan skala 1:1 prosesnya memakan waktu rata rata 30 ms dan semakin besar nilai perbandingan skala, maka waktu yang dibutuhkan untuk memproses satu frame akan semakin cepat. Sehingga dapat disimpulkan bahwa semakin besar nilai perbandingan skala, maka kecepatan proses tiap frame juga akan semakin cepat.

Hal ini karena piksel citra yang diakses oleh aplikasi semakin berkurang dari piksel aslinya.

5.5 Pengujian Pengaruh Skala Resize Dan Kamera terhadap Waktu Yang Dibutuhkan Untuk Mengaktifkan Event Klik Dan Hasil Tracking *Laser*

Pengujian ini dilakukan untuk mengetahui pengaruh kamera terhadap hasil traking sinar *laser* dan pengaruh skala resize terhadap waktu yang di butuhkan untuk membangkitkan *mouse event*. Pengujian dilakukan dengan cara mengubah nilai skala *resize* pada aplikasi dengan menggunakan tiga buah kamera yang memiliki kualitas berbeda . Hasil Pengujian dapat dilihat pada tabel 5.5

Tabel 5.5 Pengujian Pengaruh Skala *resize* terhadap Waktu Yang Dibutuhkan Untuk Mengaktifkan *Event* Klik dan Hasil *Tracking Laser*

Jenis Kamera	Skala *Resize*	Waktu yang dibutuhkan Untuk mengaktifkan *event* klik. dalam milidetik (ms)	Hasil *Tracking* Laser
M-Tech 1 MB (640 X 480)	1:1	1320	+++
	1:2	1630	+++
	1:3	1780	++
	1:4	350	+
G-STAR VGA (320 X 240)	1:1	1670	+++
	1:2	2520	+
	1:3	2680	+
	1:4	2580	+
Handycam Sony (9 MB)	1:1	700	++++
	1:2	660	++++
	1:3	530	+++
	1:4	2150	++

Catatan :

++++ : Warna yang dideteksi *(tracking)* sangat bagus dan stabil .

+++ : Warna yang dideteksi *(tracking)* bagus dan stabil, hanya terjadi sedikit kegagalan pendeteksian warna laser.

++ : Warna yang dideteksi *(tracking)* bagus dan stabil, terjadi cukup banyak kegagalan pendeteksian warna laser.

+ : Warna yang dideteksi *(tracking)* tidak bagus, terdapat banyak kegagalan pendeteksian warna sinar *laser*.

Sumber : Pengujian

Dari tabel tersebut dapat diketahui bahwa waktu yang dibutuhkan untuk membangkitkan *event* klik berbeda-beda pada tiap skala *resize* yang digunakan. Hasil *tracking* sinar laser lebih stabil ketika kamera yang digunakan memiliki spesifikasi yang bagus (Handycam Sony 9 MB). Terjadinya *tracking* warna yang tidak stabil dikarenakaan kualitas kamera yang digunakan dan pencahayaan ruangan yang kurang baik sehingga dapat mengakibatkan kegagalan pendeteksian berkas laser pada citra yang diproses pada daerah tertentu.

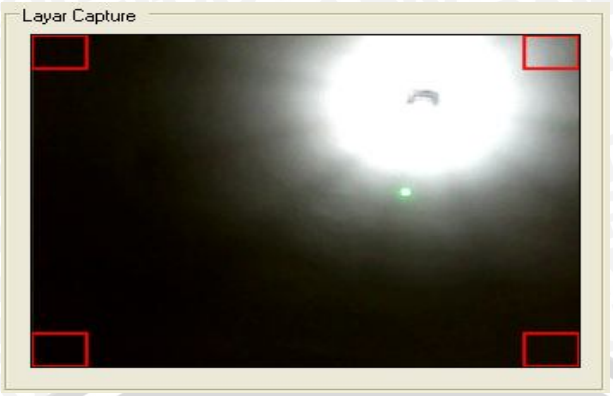
5.6 Analisis Faktor Kegagalan

Pada kenyataannya, aplikasi pendeteksian ini tidak selalu berjalan optimal. Hal ini dipengaruhi beberapa faktor antara lain spesifikasi hardware komputer yang digunakan, sehingga mempengaruhi kecepatan proses aplikasi dan hasil citra yang ditangkap, kemudian faktor luar seperti kondisi pencahayaan ruangan sekitar yang kurang baik sehingga dapat menyebabkan hasil tracking yang kurang baik.

5.7 Faktor kegagalan pendeteksian sinar *laser*

Aplikasi yang berjalan tidak selalu berjalan optimal. Hal ini disebabkan beberapa faktor:

1. Kondisi penerangan ruangan yang kurang baik (intensitas cahaya terlalu terlalu tinggi dan cahaya yang tidak konstan).
2. Kualitas sensor *webcam* yang kurang bagus, bila ada cahaya kuat baik dari lampu penerangan atau *level brightness* terlalu tinggi pada LCD proyektor, maka citra yang ditangkap tidak sempurna sehingga terjadi kegagalan pendeteksian warna *laser* karena sensor *webcam* terlihat banyak bagian berwarna putih. seperti ditunjukkan pada gambar 5.2



Gambar 5.2 O*verexposed* Pada *Webcam*

Sumber : Pengujian

5.8 Kesimpulan Hasil Pengujian

Berdasarkan hasil pengujian yang telah dilakukan, Dapat disimpulkan bahwa secara fungsional sistem sudah dapat menghasilkan *output* yang diharapkan.

BAB VI

PENUTUP

6.1 Kesimpulan

Berdasarkan hasil perancangan, implementasi, pengujian dan analisis sistem maka dapat diambil kesimpulan sebagai berikut:

1. Aplikasi dapat mengaktifkan *mouse event* dengan cara mengintegrasikan fungsi pembangktit *mouse event* kedalam program aplikasi sebelumnya.
2. Aplikasi dapat mengaktifkan *mouse event* berupa *event* klik dan *scroll* dengan memanfaatkan *user*32.dll yang merupakan *library* dari *Windows API*.

6.2 Saran

Hal yang dapat dikembangkan dalam skripsi ini adalah membuat aplikasi pembangkit *mouse event* dengan menggunakan deteksi pola tertentu menggunakan *laser pointer*.

DAFTAR PUSTAKA

Mardhana, Luky. 2011. *Aplikasi Pendeteksi Laser pointer Untuk Menggerakan Kursor Mouse Pada Komputer*. Skripsi Teknik Elektro. Universitas Brawijaya.

Andreas, Koschan. 2007. *Digital Color Images Processing*.United State: Wiley.

Bar, Avi http://www.codeproject.com/Articles/32556/Auto-Clicker-C (diakses tanggal 20 February 2012).

Anonim. *BMP File Format*. http://en.wikipedia.org/wiki/BMP_file_format (diakses tanggal 10 Agustus 2010).

Burger, Wilhelm., Burge, Burge, Mark James. 2008. *Digital Image Processing*. New York: Spinger.

Hartato, Budi. 2008. *Memahami Visual C# Secara Mudah.*Yogyakarta: Andi.

Kirilov, Andrew. *Image Processing and Computer Vision. http://aforgenet.com/forum/viewforum.php?f=4&sid=70eaae4c47de1d0ecbf8736a6bf8c935* (Di akses tanggal 23 Februari 2012).

Noble, Joshua. 2009. *Programing Interactivity*. O' Reil.United State of America.

Serban, Iulian., Brezoi, Dragos., rdlu, Tiberiu., dan Ward, Adam. 2006. *GDI+ Custom Control with C#*. UK: Packt.

Sutoyo, T., Mulyanto, Edy., Suhartono, Vincent., Nurhayati, Oky Dwi., Wijanarto. 2009. *Teori Pengolahan Citra Digital*. Andi: Yogyakarta.

Wijaya, Marvin Ch., Prijono.Agus.2007. *Pengolahan Citra digital menggunakan Matlab*. Bandung: Informatika.

Yuan, Feng. 2000. *Windows Graphics Programming Win32 GDI and DirectDraw*. Prentice Hall.