PENGOLAHAN CITRA DIGITAL UNTUK MENYELEKSI KEMASAN KALENG YANG CACAT

Dinamika

Oleh :

Nama : Galih Aris Harmawan

NIM : 00.41020.0014

Program : S1 (Strata Satu)

Program Studi : Sistem Komputer

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

SURABAYA

2006

PENGOLAHAN CITRA DIGITAL UNTUK MENYELEKSI KEMASAN KALENG YANG CACAT

SKRIPSI

Diajukan sebagai salah satu syarat untuk menyelesaikan

Program Sarjana Komputer

UNIVERSITAS Dinamika

Oleh :

Nama : Galih Aris Harmawan

NIM : 00.41020.0014

Program : S1 (Strata Satu)

Program Studi : Sistem Komputer

SEKOLAH TINGGI

MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER

SURABAYA

2006

PENGOLAHAN CITRA DIGITAL UNTUK MENYELEKSI KEMASAN KALENG YANG CACAT

Disusun Oleh :

Nama : Galih Aris Harmawan

NIM : 00.41020.0014

Surabaya, Mei 2006

Telah diperiksa, diuji dan disetujui :

Pembimbing I	Pembimbing II
Rahmadwati, MT	**Harianto, S.Kom**
NIP/NID	NIP/NID

Mengetahui :

Wakil Ketua Bidang Akademik

Drs. Antok Supriyanto, M.MT
NIP/NID. 07.085.05.00455

SEKOLAH TINGGI MANAJEMEN INFORMATIKA & TEKNIK KOMPUTER SURABAYA

UNIVERSITAS
Dinamika

Kerjakanlah tugasmu tanpa mengenal ragu…!!!
Hanya dengan kesungguhan hati dan niatan yang tulus, puing citamu akan bisa terkumpul menjadi sebuah perwujudan kebanggaan atas kerja kerasmu

Kupersembahkan untuk
Ayah, Bunda serta Adik-Adikku tercinta

UNIVERSITAS Dinamika

ABSTRAKSI

Perkembangan teknologi di bidang otomasi industri yang semakin pesat mendorong pengembangan-pengembangan pengetahuan di bidang sistem kontrol kualitas. Kebutuhan akan kualitas yang baik menjadi salah satu alasan diadakannya riset secara terus-menerus terutama dalam bidang sistem kontrol kualitas kemasan. Sistem yang dibangun dalam Tugas Akhir ini adalah sebuah sistem yang bertujuan untuk menyeleksi kemasan kaleng yang cacat. Melalui metode yang ada pada pengolahan citra, dapat diketahui kondisi objek kemasann yang di ujikan.

Citra yang di olah adalah dari hasil proses pengambilan citra menggunakan kamera. Kemudian ada beberapa langkah yang harus dilalui oleh citra tersebut diantaranya adalah proses *Gray Scale*, proses ini akan mengubah citra menjadi citra dengan format lebih sederhana. Hasil dari *Gray Scale* tersebut akan dilanjutkan dengan proses Perbaikan Efek Ketidak Seragaman Pencahayaan, setelah itu citra tersebut masuk dalam proses Modifikasi Kecemerlangan Citra dan terakhir masuk dalam proses Deteksi Gerakan (*Motion Detection*).

Hasil proses Deteksi Gerakan kemudian dapat dihitung prosentase *error*-nya yang muncul. Dari *error* tersebut dapat digunakan untuk menentukan keadaan fisik kemasan kaleng.

UNIVERSITAS Dinamika

KATA PENGANTAR

Dengan penuh rasa syukur kehadirat Allah SWT, penulis dapat menyelesaikan tugas akhir yang berjudul PENGOLAHAN CITRA DIGITAL UNTUK MENYELEKSI KEMASAN KALENG YANG CACAT sebagai salah satu persyaratan yang harus dipenuhi untuk menyelesaikan program sarjana komputer di program studi strata satu Sistem Komputer Sekolah Tinggi Manajemen Informatika & Teknik Komputer Surabaya (STIKOM).

Penulis juga mengucapkan penghargaan dan terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah mendukung dan atau membantu penulis dalam proses pengerjaan tugas akhir ini, yaitu :

1. Ayah, Ibu dan seluruh keluarga yang tidak henti-hentinya memberi dukungan dan doa selama penulis menempuh studi.
2. Ibu Rahmadwati, MT, sebagai dosen pembimbing I atas segala arahan dan bimbingannya .
3. Bapak Harianto, S.Kom, sebagai dosen pembimbing II atas bimbingan dan dorongannya.
4. Bapak Basuki rahmat, S.Si., MT, yang telah memberikan ide untuk pengajuan judul Tugas Akhir.
5. Rekan-rekan angkatan 2000, yang telah memberikan tantangan sehingga penulis dapat menyelesaikan Tugas Akhir ini dapat penulis selesaikan dengan baik.
6. Serta pihak-pihak lainnya yang tidak mungkin penulis sebutkan satu-persatu.

UNIVERSITAS Dinamika

semoga Allah SWT memberikan pahala yang setimpal kepada semua pihak yang telah memberikan bantuan, bimbingan, ataupun nasehat-nasehat kepada penulis.

Penulis juga menyadari sepenuhnya bahwa karya tugas akhir ini masih jauh dari kesempurnaan, untuk itu dengan segala kerendahan hati penulis membuka akses seluas-luasnya untuk pengembangan tugas akhir ini lebih lanjut agar dapat berguna untuk pengembangan teknologi khususnya teknologi di bidang otomasi kontrol industri.

Surabaya, Mei 2006

Penulis

UNIVERSITAS Dinamika

DAFTAR ISI

UNIVERSITAS Dinamika

UNIVERSITAS Dinamika

DAFTAR TABEL

UNIVERSITAS Dinamika

DAFTAR GAMBAR

UNIVERSITAS Dinamika

UNIVERSITAS Dinamika

DAFTAR ISTILAH DAN SINGKATAN

AC	*Alternating Current*
BMP	*Bitmap*
Brightness	*Kecemerlangan*
CMOS	*Complementary Metal Oxide Semiconductor*
CCD	*Charged Coupled Device*
DC	*Direct Current*
EMF	*Elektromagnetic Field*
ECP	*Extended Capabilities Mode*
EPP	*Enhanced Parallel Port*
DOS	*Disk Operating System*
GUI	*Graphical User Interface*
Hex	*Hexadecimal*
IDE	*Integrated Development Environment*
IC	*Integrated Circuit*
Indeks	*Urutan*
I/O	*Input / Output*
Looping	*Perputaran / Siklus*
LSB	*Least Significant Bit*
MDI	*Multiple Document Interface*
MSB	*Most Significant Bit*
NPN	*Negative Positive Negative*

UNIVERSITAS Dinamika

NTSC	*National Television System Committee*
Pixel	*Picture Element*
PNP	*Positive Negative Positive*
Range	*Batasan*
RAD	*Rapid Application Development*
RAM	*Random Access Memory*
RGB	*Red-Green-Blue*
TTL	*Transistor-transistor Logic*
USB	*Universal Serial Bus*
WMF	*Windows Metafiles*

UNIVERSITAS Dinamika

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Penelitian mengenai kualitas buah telah dilakukan oleh Ahmad Nidhom. Penelitian tersebut membahas tentang pemilahan / penentuan klasifikasi buah manggis berdasarkan citra dari buah tersebut. Terdapat juga penelitian tentang kontrol kualitas yang dilakukan oleh departemen perindustrian dan perdagangan direktorat jendral industri dan dagang kecil menengah yang membahas tentang kontrol kualitas dan aspek ekonominya, dalam hal ini tentang kontrol kualitas kemasan suatu barang.

Perkembangan berbagai bidang industri sekarang ini semakin pesat. Hal ini juga berpengaruh pada manajemen dan piranti-piranti industri yang digunakan. Terutama pada industri yang berfokus pada kualitas kemasan suatu barang. Sebagai contoh, kontrol kualitas kemasan, karena baik atau buruknya kualitas kemasan barang sangat mempengaruhi daya jualnya di pasaran.

Hal utama yang dinilai oleh konsumen dalam pemilihan barang adalah kondisi fisik barang, dalam hal ini wadah / kemasan kaleng.

Berdasarkan uraian diatas, peneliti berkeinginan untuk melakukan penelitian tentang pemilahan / penentuan kualitas kemasan kaleng. Penelitian ini menggunakan metode-metode pengolahan citra *digital*. Metode deteksi gerakan (*motion detection*) dapat mendeteksi kemasan kaleng yang mengalami perubahan bentuk fisik, sehingga resiko kerugian akibat ketidak sempurnaan kemasan barang dapat di atasi dengan baik.

1.2. Perumusan Masalah

Berdasarkan latar belakang masalah dalam penelitian tugas akhir ini dapat dirumuskan beberapa perumusan masalah, diantaranya:

1. Bagaimana merancang komputer agar dapat membaca *input* yang berupa citra dari kamera ?
2. Bagaimana proses mengubah citra berwarna menjadi citra *gray scale.*
3. Bagaimana proses mengurangi efek ketidak seragaman pencahayaan pada citra hasil proses *gray scale.*
4. Bagaimana proses memodifikasi kecemerlangan citra *gray scale.*
5. Bagai mana proses mencari nilai pixel yang berbeda antara dua citra dengan menggunakan algoritma deteksi gerakan (*Motion Detection*).
6. Bagaimana merancang suatu sistem agar dapat melakukan pemilahan terhadap objek yang kurang sempurna berdasarkan input yang diterima ?
7. Bagaimana membuat *software* yang bisa di aplikasikan terhadap proses pemilahan / penentuan kualitas kemasan kaleng ?

1.3. Pembatasan Masalah

Perlu diberikan beberapa pembatasan masalah dengan tujuan agar pembahasan tidak meluas dan menyimpang dari tujuan. Adapun batasan permasalahan dari sistem yang akan dibuat ini adalah:

1. Citra yang diproses adalah citra dengan format BMP.
2. Citra yang di proses adalah citra yang di peroleh dari pengambilan oleh kamera yang kemudian disimpan dengan format seperti tersebut di atas.
3. Citra yang di proses memiliki ukuran file yang sama.

UNIVERSITAS Dinamika

4. *Interface* dari komputer ke alat menggunakan *interface* sederhana yaitu *parallel port.*
5. *Interface* dari komputer ke kamera menggunakan *interface USB Port.*
6. Tidak membahas *interface* untuk kamera
7. Tidak membahas rangkaian pada kamera.
8. Objek penelitian hanya menggunakan kemasan kaleng dengan satu ukuran.
9. Tidak membahas mengenai kemasan kaleng secara detail (warna, ukuran, dll).
10. Hanya mendeteksi sisi samping kaleng, tidak termasuk sisi atas dan bawah.

1.4. Tujuan

Membuat sebuah aplikasi pengolahan citra yang dapat digunakan untuk memilah kemasan kaleng dari suatu barang menjadi baik atau cacat.

1.5. Kontribusi

Penelitian tentang kualitas saat ini telah banyak dilakukan. Beberapa diantaranya dikembangkan melaui metode-metode pengolahan citra digital. Pengklasifikasian buah oleh Ahmad Nidhom, dimana pengklasifikasian buah manggis berdasarkan citra dari buah tersebut. Penelitian tentang kontrol kualitas dan aspek ekonomi oleh DEPERIDAG dirjen Industri dan Dagang Kecil Menengah di jakarta yang membahas tentang kontrol kualitas kemasan barang.

Dalam Tugas Akhir ini akan diteliti kemampuan algoritma deteksi gerakan untuk mendeteksi nilai-nilai *pixel* dari citra yang berubah bentuk fisiknya, dilengkapi dengan proses *gray scale* untuk merubah citra warna menjadi citra dengan skala keabuan, perbaikan efek ketidak seragaman pencahayaan untuk

memperbaiki ketidak seragaman pencahayaan (*illumination*) saat peroses pengambilan citra menggunakan kamera dan kecemerlangan citra untuk dapat memperjelas citra objek pada ruang dengan pencahayaan yang kurang. Sampai dengan saat ini belum ada yang mempublikasikannya, sehingga sangat penting untuk menambah kemajuan di bidang kontrol kualitas kemasan.

1.6. Sistematika Penulisan

Penulisan buku Tugas Akhir ini, secara garis besar dapat di kelompokkan dalam dua bagian, yaitu bagian *software* dan bagian *hardware*. Penulisannya dilakukan secara urut dengan membahas bagian *software*-nya terlebih dahulu, kemudian dilanjutkan dengan bagian *hardware*-nya. Selengkapnya dapat dilihat pada penjelasan berikut:

BAB I : PENDAHULUAN

Pada bab ini, dijelaskan mengenai latar belakang dari topik tugas akhir yang diambil, kemudian dirumuskan menjadi suatu permasalahan yang akan diselesaikan dalam tugas akhir ini, batasan-batasan masalah yang akan diteliti, tujuan dari penelitian tugas akhir ini.

BAB II : LANDASAN TEORI

Pada bab ini, dijelaskan teori-teori digunakan sebagai dasar analisa permasalahan. Diawali dengan penjelasan mengenai algoritma yang digunakan untuk membangun *software* aplikasi. Kemudian penjelasan tentang komponen-komponen yang digunakan untuk membangun *hardware* (bagian elektroniknya).

UNIVERSITAS Dinamika

BAB III : METODE PENELITIAN

Dalam bab ini, dijelaskan tentang perancangan *software* (algoritma-algoritma yang digunakan), perancangan elektronik (komponen-komponen penting yang digunakan), dan mekanik (motor yang digunakan). Penjelasannya akan ditulis secara urut, sesuai dengan urutan diatas. Kemudian dilanjutkan dengan pembuatan *software* (potongan program yang terdapat dalam *software* aplikasi), *hardware* (gambar desain rangkaian yang digunakan dan gambar fisik mekanik yang dibuat).

BAB IV : PENGUJIAN DAN EVALUASI SISTEM

Dalam bab ini, membahas mengenai pengujian terhadap bagian-bagian sistem yang di bangun (bagian *software, hardware*). Pembahasan akan dilanjutkan dengan pengujian terhadap keseluruhan sistem yang dibangun. Kemudian dijelaskan tentang data hasil pengujian sistem secara keseluruhan.

BAB V : PENUTUP

Bagian ini merupakan bagian akhir dari laporan penelitian tugas akhir ini yang menguraikan kesimpulan-kesimpulan yang diperoleh dari proses penelitian serta saran-saran untuk pengembangan penelitian selanjutnya.

UNIVERSITAS Dinamika

BAB II

LANDASAN TEORI

2.1. Kamera

Kamera merupakan alat paling populer dalam aktivitas fotografi. Nama ini didapat dari *camera obscura*, bahasa Latin untuk "ruang gelap", mekanisme awal untuk memproyeksikan tampilan di mana suatu ruangan berfungsi seperti cara kerja kamera fotografis yang modern, kecuali tidak ada cara pada waktu itu untuk mencatat tampilan gambarnya selain secara manual mengikuti jejaknya. Saat ini ada dua jenis kamera yaitu :

- Kamera konfensional yaitu kamera yang mengunakan film sebagai media penangkapan gambar.
- Kamera digital yaitu kamera yang menggunakan sensor sebagai media penangkapannya dan disimpan secara digital pada tempat penyimpanan data.

Secara garis besar cara kerja kedua kamera ini sama, hanya yang membedakannya hanya pada media penangkapan dan penyimpanan, jika konfesional media penangkapan sekaligus juga merupakan media penyimpanan, sedangkan pada digital media penangkapan bukan merupakan media penyimpanan. Kelebihan kamera digital bila dibandingkan dengan kamera konfensional adalah kamera digital dapat langsung dilihat hasilnya saat itu juga, dapat melakukan pemrosesan citra secara langsung, sedang kelemahannya hasil dari kamera digital tergantung dari berapa besar pixel yang mampu didapat, jika

UNIVERSITAS Dinamika

citra dibesarkan akan terlihat pecah, sedang kamera konfensional mampu dicetak pada ukuran besar misal untuk digunakan biro iklan (Wikipedia, 2006).

2.1.1. Webcam

Gambar 2.1 Webcam

UNIVERSITAS Dinamika

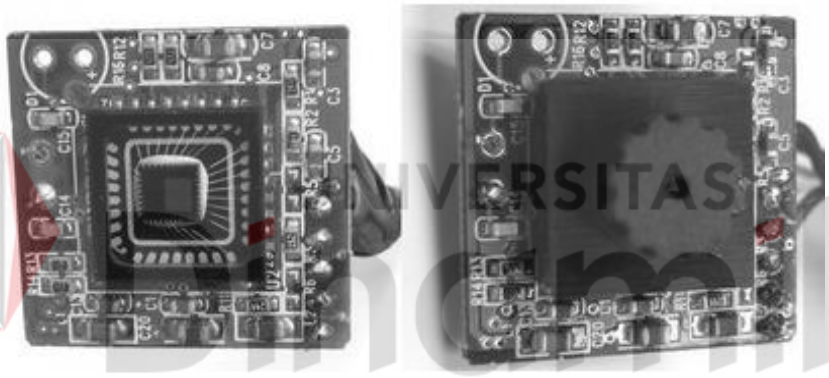
Webcam (web *camera*) pertama di publikasikan pada tahun 1991 di universitas Cambridge. Adalah suatu kamera yang bekerja secara *real-time* seperti pada gambar 2.1, dimana citra yang diperoleh dapat diakses menggunakan komputer. Cara kerjanya hampir sama dengan kamera digital, hanya penyimpanan dan aksesnya yang berbeda, dapat menggunakan komputer atau alat yang dibuat dengan tujuan tersebut (www.wikipedia.com).

Bagian-bagian dari webcam secara umum diantaranya adalah lensa, sensor citra, dan beberapa rangkaian elektronik pendukung.

Lensa Berbentuk silinder dan ditempatkan di depan badan kamera. Lensa akan memfokuskan cahaya sehingga dihasilkan bayangan sesuai ukuran sensor. Lensa dikelompokkan sesuai panjang *focal lenght* (jarak antara kedua lensa). *Focal lenght* mempengaruhi besar komposisi citra yang mampu dihasilkan. Dalam

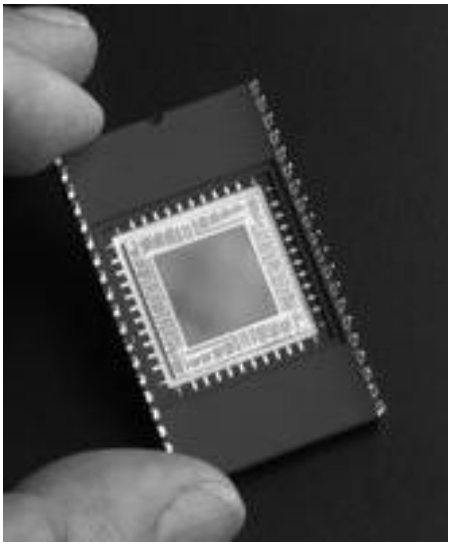
masyarakat umum, lebih dikenal dengan istilah *zoom*. Untuk jenis lensa yang digunakan sangat beragam, mulai dari lensa plastik sampai degan lensa yang terbuat dari kaca.

Sensor citra dapat berupa CMOS atau CCD. CMOS dibuat dengan proses standar silikon seperti *microprocessor, microcontroller*. Arsitektur CMOS dibuat seperti sel memori atau flat panel *display*. Tiap *pixel* berisi foto dioda yang berfungsi untuk merubah cahaya menjadi elektron, bagian *charge to voltage conversion,* reset dan transistor pemilih dan bagian *amplifier* seperti pada gambar dibawah ini.



Gambar 2.2 Sensor CMOS

Teknologi CCD telah dikembangakan untuk aplikasi citra, dan proses fabrikasinya telah dioptimalkan untuk membuat sebuah sensor citra sesuai dengan kemampuan optik dan kualitas citra, seperti pada gambar 2.3.



Gambar 2.3 Sensor CCD

CCD terdiri dari sekumpulan *pixel* atau elemen dari citra. Yang disusun dalam matriks x,y yang terdiri dari baris dan kolom. Tiap pixel terdiri dari *photodiode* mengubah cahaya (*Photon*) menjadi elektron. Banyak elektron yang dilumpulkan sesuai dengan intensitas cahaya, dan kemudian ditransver kedalam kolom-kolom. Selanjutnya muatan tersebut dibaca, per baris data , temasuk signal dari suatu pixel disetiap kolomnya ditransfer dari registrasi muatan vertikal ke regristasi muatan horisontal. Muatan tersebut akan dibaca secara seri. Proses tersebut akan terus berulang hinggga citra dapat dilihat (Wahana Komputer, 2005:21).

CCD teknologi telah dikembangkan untuk meningkatkan kualitas citra, operasi CCD juga membutuhkan aplikasi dari beberapa *signal clock, level clock, dan voltage bias.*

Elektronik pendukung yang lain pada umumnya digunakan untuk membaca citra dari sensor dan mengirimkannya ke komputer.

Tabel 2.1 Perbedaan teknologi CCD dan CMOS

CCD	CMOS
Ukuran pixel lebih kecil	Single power supplay
Tingkat gangguan yang rendah	Single master clock
Nilai gangguan pada gelap rendah	Konsumsi tenaga yang rendah
Fill factor untuk *full-frame*	Pengalamatan X,Y dan subsampling
Lebih sensitif	Ukuran sistem yang lebih kecil
	Sircuit yang terintegrasi

Pada tabel 2.1 diatas memperlihatkan perbedaan antara kedua teknologi tersebut. Meskipun banyak sekali variasi pada arsitekture CCD dan CMOS, tetapi karakteristik dasar dan perbedaan antara kedua teknologi ini serupa.

UNIVERSITAS Dinamika

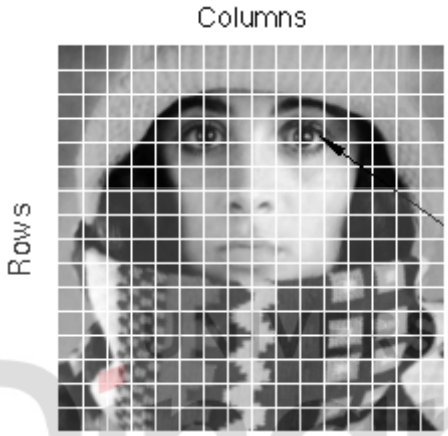
2.2. *Pixel*

Pixel adalah singkatan dari *picture element*. Citra digital dibentuk dari kotak-kotak kecil, seperti halnya suatu tatanan ubin pada dinding kamar mandi atau dapur. Meskipun citra digital nampak sangat halus, sebenarnya terdiri atas berjuta-juta *pixel* (Wahana Komputer, 2005:208).

Masing-masing *pixel* pada citra mempunyai nilai kuantitatif antara 0 dan 255 dan terdiri dari tiga elemen warna untuk citra *true color*.

Salah satu cara untuk dapat mengklasifikasikan citra digital ialah dengan mengukur kualitas *pixel*-nya. Nilai *pixel* sering identik dengan nilai resolusi, yang biasa digunakan untuk menentukan ukuran file citra digital, semakin besar

resolusi yang dimiliki citra semakin banyak pixelnya dan semakin banyak pula kemungkinan warna yang dimiliki. Sebagai contoh suatu citra *true color* 24 bit dengan resolusi sebesar 800 x 600 memiliki arti bahwa citra tersebut memiliki *pixel* sebanyak 480.000 buah *pixel* yang masing-masing *pixel*-nya mempunyai tiga buah nilai elemen warna yaitu merah, hijau, dan biru dengan komposisi warnanya masing-masing. Kemungkinan warna yang terkomposisi sebanyak 2^{24} nilai warna. Gambar berikut menggambarkan *pixel* pada citra digital.



Gambar 2.4 Pixel Pada Citra Digital

ITAS Dinamika

2.3. Representasi Citra Digital

Kemampuan Dasar komputer terletak pada pengolahan sinyal-sinyal *digital*. Sinyal-sinyal *digital* ini memiliki dua kondisi yaitu *High* yang biasa diberi simbol 1 dan *Low* yang biasa diberi simbol 0. Sinyal *digital* ini sering juga disebut sebagai sinyal biner, karena hanya memiliki dua kondisi saja. Dan untuk ukuran penyimpanan satu digit sinyal biner disebut bit (Achmad, 2005:9).

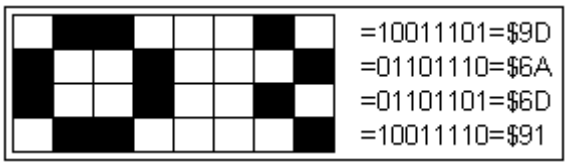
Untuk itu, citra *digital* harus mempunyai format tertentu yang sesuai sehingga dapat merepresentasikan obyek pencitraan dalam bentuk kombinasi data

biner. Suatu matriks A dengan jumlah baris N dan jumlah kolom M merupakan format paling sesuai untuk merepresentasikan citra *digital* tersebut.

$$A = \begin{bmatrix} A(1,1) & A(1,2) & \cdots & A(1,M) \\ A(2,1) & A(2,2) & \cdots & A(2,M) \\ \vdots & \vdots & & \vdots \\ A(N,1) & A(N,2) & \cdots & A(N,M) \end{bmatrix} \quad (2.1)$$

Setiap nilai elemen dari matriks secara visual merupakan representasi nilai warna dari tiap *Pixel* (*Picture Element*) suatu citra *digital* yang *range* nilainya dapat bervariasi sesuai dengan format penyimpanan citra dalam memori. Format citra *digital* yang banyak dipergunakan adalah citra biner (*Monokrom*), skala keabuan (*Gray Scale*), Warna (*True Color*), dan warna berindeks.

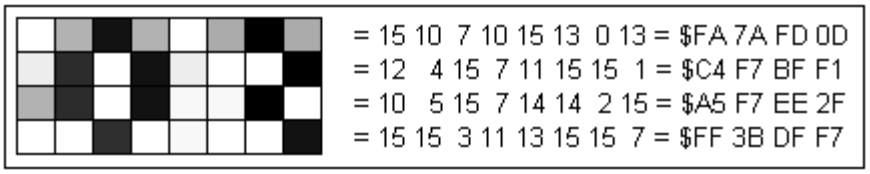
Pada citra biner (*monokrom*), setiap *pixel*-nya bernilai 0 atau 1, masing-masing merepresentasikan warna tertentu. Yang lazim digunakan warna hitam bernilai 0 dan warna putih bernilai 1, setiap *pixel* pada citra hanya membutuhkan 1 bit untuk penyimpanan, sehingga untuk setiap *byte*-nya dapat menampung informasi untuk 8 *pixel*. Seperti pada gambar berikut ini.



Gambar 2.5 Citra biner dan representasinya dalam data *digital*

Citra skala keabuan (*Gray Scale*) memberikan kemungkinan warna yang lebih banyak daripada citra biner, kerana terdapat nilai-nilai lain di antara nilai minimum (biasanya = 0) dan maksimumnya. Banyaknya kemungkinan nilai ini

bergantung pada jumlah bit yang digunakan. Contohnya untuk skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah 2^4 atau 16, dengan *range* nilai dari 0 sampai 2^4-1. Sedangkan untuk skala keabuan 8 bit, jumlah kemungkinan nilainya adalah 2^8 atau sebanyak 256, dengan *range* nilai dari 0 sampai 2^8-1. Seperti gambar dibawah ini.



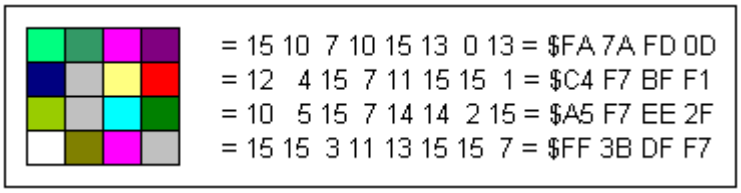
Gambar 2.6 Citra Skala Keabuan dan representasinya dalam data *digital*

Format citra ini disebut skala keabuan karena pada umumnya warna yang dipakai adalah antara warna hitam sebagai warna minimal dan warna putih sebagai warna maksimalnya, sehingga warna diantaranya adalah abu-abu. Namun pada prakteknya warna yang dipakai tidak terbatas pada warna abu-abu.

Pada citra warna (*True Color*), setiap *pixel*-nya mempunyai warna yang merupakan representasi dari kombinasi warna dasar yaitu merah, hijau, dan biru. Format citra ini sering disebut sebagai citra RGB (*Red-Green-Blue*). Setiap warna dasar mempunyai intensitas sendiri dengan nilai maksimum 255 (8 bit), misalnya warna kuning merupakan kombinasi dari warna merah dan hijau, sehingga nilai RGB-nya adalah 255 255 0. Dengan demikian setiap *pixel* pada citra warna membutuhkan 3 *byte* untuk tempat penyimpanannya dalam memori.

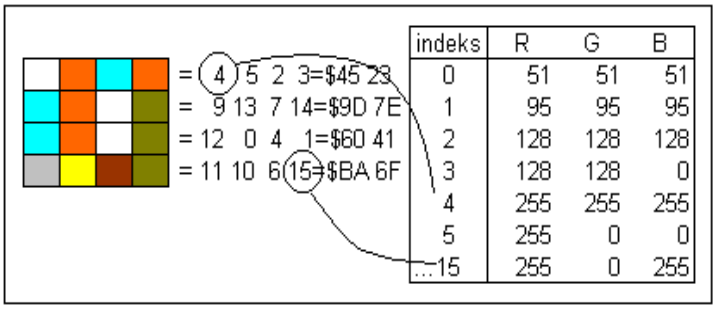
Jumlah kombinasi warna yang mungkin untuk format citra ini adalah 2^{24} atau lebih dari 16 juta warna, sehingga dapat dianggap mencakup semua warna yang ada, ini sebabnya format ini dinamakan *true color*. Seperti gambar berikut ini.

UNIVERSITAS Dinamika



Gambar 2.7 Citra *true color* dan representasinya dalam data *digital*

Citra warna berindeks merupakan pengembangan dari format citra *true color*. Dikarenakan besarnya kapasitas memori untuk menyimpan citra dengan formar *true color*, dan pada kebanyakan kasus jumlah warna yang ada dalam suatu citra terkadang sangat terbatas (jauh dibawah 16 juta kemungkinan warna yang ada), dan karena banyaknya warna dalam sebuah citra tidak mungkin melebihi banyaknya *pixel* dalam citra itu sendiri maka disediakan suatu format citra warna berindeks. Pada format ini, informasi setiap *pixel* merupakan indeks dari suatu tabel yang berisi informasi warna yang tersedia, yang disebut palet warna (pada beberapa buku disebut *color map*). *Color map* merupakan kumpulan warna-warna yang sering digunakan dan nilai RGB-nya disimpan dalam suatu tabel sehingga kita tinggal menggunakan indeks dari tabel tersebut. Seperti gamabar berikut.



Gambar 2.8 Citra warna berindeks

Jumlah bit yang dibutuhkan oleh setiap *pixel* pada citra, bergantung pada jumlah warna yang tersedia dalam palet warna. Sebagai contoh, untuk palet berukuran 16 warna, setiap *pixel* membutuhkan 4 bit, dan untuk palet berukuran 256 warna, setiap pixel membutuhkan 8 bit atau 1 *byte*. Palet warna merupakan bagian dari citra warna berindeks, sehingga pada saat menyimpan citra ini kedalam file, informasi palet warna juga harus disertakan.

Ada beberapa keuntungan yang dapat diambil dari penggunaan palet warna, diantaranya adalah dapat dengan cepat melakukan manipulasi warna tanpa harus mengubah informasi pada setiap *pixel* dalam citra. Keuntungan lainnya adalah besarnya data yang diperlukan untuk menyimpan citra ini lebih kecil bila dibandingkan dengan citra warna *true color* (Achmad, 2005:11).

Setting warna *display* pada Microsoft Windows biasanya memiliki beberapa pilihan sesuai dengan format citra yang telah dijelaskan, yaitu format 16 *color,* format 256 *color,* format *high color* (yang merupakan citra warna berindeks dengan ukuran palet masing-masing 4 bit, 8 bit, dan 16 bit), serta *true color.*

2.4. Algoritma Skala Keabuan (*Gray Scale*)

Mata manusia memiliki 3 jenis sensor kerucut pada retina yang mendeteksi rentang warna yang berbeda pada spektrum cahaya tampak. Sensitivitas mata manusia terhadap warna bisa berbeda-beda. Mata lebih sensitif pada warna hijau, merah, dan biru. Oleh karena itu, konversi informasi suatu citra warna ke dalam skala keabuan dapat dilakukan denganmencari nilai rerata dari

ketiga nilai elemen warna. Persamaan yang digunakan untuk mengkonversi citra berwarna menjadi citra skala keabuan adalah sebagai berikut (Achmad, 2005:66):

$$Gray = (R + G + B) / 3 \quad (2.2)$$

konversi informasi suatu citra warna ke dalam skala keabuan dapat juga dilakukan dengan cara memberi bobot yang berbeda pada setiap elemen warna (Achmad, 2005:71), sehingga persamaan 2.2 dimodifikasi menjadi:

$$Gray = w_R R + w_G G + w_B B \quad (2.3)$$

dengan w_R, w_G, dan w_B masing-masing adalah bobot untuk elemen warna merah, hijau, dan biru. NTSC (*National Television System Committee*) mendefinisikan bobot untuk konversi citra warna ke skala keabuan adalah sebagai berikut:

w_R=0,299 w_G=0,587 w_B=0,114

Untuk citra berwarna nilai dari suatu *pixel* misal adalah X, maka untuk mendapatkan nilai *Red, Green, Blue* dapat digunakan rumus:

$$Blue = \frac{X}{2^{16}}$$
$$Green = \frac{\left(X - Blue * 2^{16}\right)}{2^8} \quad (2.4)$$
$$\mathrm{Re}\, d = X - Blue * 2^{16} - Green * 2^8$$

Selanjutnya, nilai dari variabel *Red, Green, Blue* dapat diproses menggunakan persamaan 2.2 atau 2.3 untuk mendapatkan nilai *gray*-nya.

2.5. Algoritma Perbaikan efek ketidak seragaman pencahayaan

Pencahayaan (*illumination*) yang tidak merata saat dilakukan proses pengambilan citra mengakibatkan citra yang diperoleh tidak sesuai dengan kenyataan. Perbaikan citra akibat ketidak seragaman pencahayaan dapat dilakukan dengan mengurangi citra tersebut dengan citra latarnya, yang diperoleh dengan

melakukan pengambilan citra tanpa adanya obyek / hanya latar saja (Achmad, 2005:133). Sesuai dengan persamaan berikut ini:

$$C(x,y) = A(x,y) - B(x,y) \quad (2.5)$$

C(x,y) adalah citra yang diinginkan, A(x,y) merupakan citra yang akan diproses dalam algoritma ini, dan B(x,y) adalah citra latar yang diperoleh dari kamera tanpa adanya objek benda.

2.6. Algoritma Modifikasi Kecemerlangan Citra (*Brightness*)

Di ruangan yang agak gelap, seringkali perlu menyalakan lampu atau sumber cahaya lainnya agar ruangan menjadi lebih terang dan benda-benda yang ada dalam ruangan dapat terlihat dengan jelas. Dalam pengolahan citra, hal itu sama dengan penambahan warna putih yaitu dengan cara meningkatkan sekala keabuan dari seluruh bagian (setiap titik) dalam citra tersebut untuk meningkatkan kecemerlangannya (*brightness*) (Achmad, 2005:45). Proses diatas dapat dilakukan dengan menerapkan fungsi liner berikut ini untuk memetakan skala keabuan citra.

$$Ko = Ki + C \quad (2.6)$$

dimana C adalah suatu konstanta yang bernilai positif jika kita hendak meningkatkan kecemerlangan citra, dan sebaliknya bernilai negatif jika hendak mengurangi kecemerlangannya.

2.7. Algoritma Deteksi Gerakan (*Motion Detection*)

Operasi yang melibatkan 2 buah citra atau lebih dan menghasilkan sebuah citra keluaran yang merupakan hasil dari operasi matematis sering disebut sebagai operasi berbasis bingkai. Operasi ini dilakukan titik per titik dengan lokasi

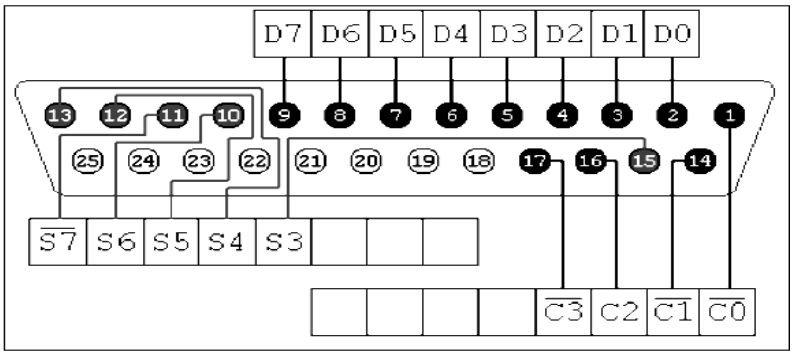
yang bersesuaian pada citra-citra masukan tersebut. Salah satu operasi yang termasuk dalam opersai berbasis bingkai adalah deteksi gerakan.

Deteksi gerakan secara sederhana dapat dilakukan dengan mencari beda antara 2 buah citra yang berurutan pada hasil pengambilan citra menggunakan kamera video *digital.* Operator yang dapat digunakan untuk melakukan deteksi gerakan adalah operator pengurangan, dimana pada bagian yang tidak bergerak / berubah pada citra akan menghasilkan nilai nol, sedangkan bagian yang bergerak / berubah akan memberikan nilai yang tidak nol. Yang perlu di perhatikan bila menggunakan operasi ini, ukuran antara kedua citra masukkan haruslah sama (Achmad, 2005:132).

Persamaan yang dapat diterapkan dalam operasi ini adalah sama dengan persamaan 2.5 yang membedakan adalah bahwa keduanya merupakan citra objek, bukan citra latar.

$$Ko(x,y) = Ki_1(x,y) - Ki_2(x,y) \qquad (2.7)$$

2.8. *Parallel Port* Komputer



Gambar 2.9 *Parallel port* pada komputer

*Parallel po*rt adalah *port* yang sering digunakan dalam mengerjakan *project* yang membutuhkan *intefacing* antara alat dengan komputer (Harries, 2003). *Port* ini dapat memberikan *input* sampai dengan 9 bit atau *output* sampai dengan 12 bit dalam satu satuan waktu. *Port* ini terdiri dari 4 jalur kontrol, 5 jalur status and 8 jalur data, dapat ditemukan di bagian belakang dari komputer yang berbentuk sebuah konektor jenis D-25 *male*.

Parallel Port yang baru telah disesuaikan dengan standar IEEE 1284 yang di release pertama kali pada tahun 1994. Standar ini terdiri dari 5 mode operasi, diantaranya adalah :

1. *Compatibility* Mode.
2. *Nibble* Mode.
3. *Byte* Mode.
4. EPP Mode (*Enhanced Parallel Port*).
5. ECP Mode (*Extended Capabilities* Mode).

UNIVERSITAS Dinamika

Compatibility mode adala mode operasi yang sering digunakan pada *parallel port*. Mode ini dapat melakukan pengiriman data dalam keadaan normal sekitar 50 kbps, tetapi dapat lebih cepat dari itu yaitu sekitar 150 kbps. Dalam proses penerimaan data kita harus melakukan perubahan mode operasi menjadi *Nibble* mode atau *Byte* mode. *Nibble* mode dapat menerima 4 bit data dari *device* ke komputer. Sedangkan *Byte* mode digunakan untuk *parallel port bi-directional* (tidak semua komputer memiliki fasilitas ini).

IBM telah mendefinisikan tiga alamat yang digunakan oleh standar *parallel port*. Ketiga alamat tidak selalu sama anatara satu sistem dengan sistem yang lainnya. Alamat LPT1 yang digunakan pada komputer adalah 378 Hex

sampai 37B Hex. Jadi *base address* yang digunakan adalah 378 Hex. Setiap parallel port seperti yang telah di jelaskan sebelumnya memiliki masing-masing tiga *register*. Tabel 2.2 berikut ini akan menjelaskan tentang *register-register* yang terdapat pada *parallel port.*

Tabel 2.2 *Register-register* pada *parallel port*

Register Name	Address
Data Register	Base + 00h
Status Register	Base + 01h
Control Register	Base + 02h

Dalam tabel 2.3 berikut ini, terlihat bahwa terdapat simbol "n" di depan nama sinyal (seperti *nError*), hal ini menandakan bahwa sinyal tersebut adalah aktif *low*."*Hardware Inverted*" menandakan bahwa sinyal di-*invert* oleh *hardware parallel card.* Sebagai contoh sinyal *Busy*, jika +5V (*logic* 1) dimasukkan pada pin ini dan *status register* kemudian dibaca, maka akan menghasilkan *logic* 0 dalam bit ke-7 pada *status register.*

Base address pada *parallel port* biasanya adalah 378h (LPT1) atau 278h (LPT2). Tabel 2.3 memperlihatkan data *register* yang secara umum digunakan untuk mengeluarkan data melalui *parallel port*. *Register* ini normalnya hanya mengeluarkan data saja. Jika kita mencoba membaca data, maka akan di dapatkan data yang terakhir dikirim ke *parallel port*. Walaupun demikian, jika *parallel port* komputer anda adalah *bi-directional* maka anda dapat menerima data pada alamat ini (Harries, 2003).

Tabel 2.3 Konfigurasi pin *parallel port*

Pin No (D-Type 25)	Pin No (Centronics)	SPP Signal	Direction In/out	Register	Hardware Inverted
1	1	Nstrobe	In/Out	Control	Yes
2	2	Data 0	Out	Data	
3	3	Data 1	Out	Data	
4	4	Data 2	Out	Data	
5	5	Data 3	Out	Data	
6	6	Data 4	Out	Data	
7	7	Data 5	Out	Data	
8	8	Data 6	Out	Data	
9	9	Data 7	Out	Data	
10	10	Nack	In	Status	
11	11	Busy	In	Status	Yes
12	12	Paper-Out / Paper-End	In	Status	
13	13	Select	In	Status	
14	14	nAuto-Linefeed	In/Out	Control	Yes
15	32	nError / nFault	In	Status	
16	31	NInitialize	In/Out	Control	
17	36	nSelect-Printer / nSelect-In	In/Out	Control	Yes
18 – 25	19-30	Ground	Gnd		

Tabel 2.4 *Software* register untuk data *port*

Offset	Name	Read / Write	Bit No.	Properties	Low	High
Base + 0	Data Port	Write (Note-1)	Bit 7 (MSB)	Data 7	0	1
			Bit 6	Data 6	0	1
			Bit 5	Data 5	0	1
			Bit 4	Data 4	0	1
			Bit 3	Data 3	0	1
			Bit 2	Data 2	0	1
			Bit 1	Data 1	0	1
			Bit 0 (LSB)	Data 0	0	1

Tabel 2.5 *Software register* untuk status *port*

Offset	Name	Read / Write	Bit No.	Properties	Low	High
Base + 1	Status Port	Read Only	Bit 7 (MSB)	Busy	Busy	Not busy
			Bit 6	Ack	Nack	Ack
			Bit 5	Paper Out	No paper	Paper
			Bit 4	Select In	Not selected	Selected
			Bit 3	Error	No error	Error
			Bit 2	IRQ (Not)	-	-
			Bit 1	Reserved	-	-
			Bit 0 (LSB)	Reserved	-	-

Status port (*Base address*+1) adalah port yang hanya dapat dibaca saja, lihat tabel 2.4. Data yang dikeluarkan dari *port* ini akan diabaikan. Pada bit ke-7 (*Busy*) adalah aktif *low input*. Sama seperti bit ke-2 (*nIRQ*) juga aktif *low*.Jika bit ke-2 bernilai 1 (*logic* 1) maka menandakan bahwa *interrupt* tidak terjadi.

Tabel 2.6 *Software register* untuk *control port*

Offset	Name	Read / Write	Bit No.	Properties	Low	High
Base + 2	Control Port	Read / Write	Bit 7 (MSB)	Unused	-	-
			Bit 6	Unused	-	-
			Bit 5	Enable Bi-Directional Port	-	-
			Bit 4	Enable IRQ Via Ack Line	Interrupt Disable	Interrupt Enable
			Bit 3	Select Printer	Selected	Not selected
			Bit 2	Initialize Printer (Reset)	False	True
			Bit 1	Auto Linefeed	True	False
			Bit 0 (LSB)	Strobe (Active-Low)	True	false

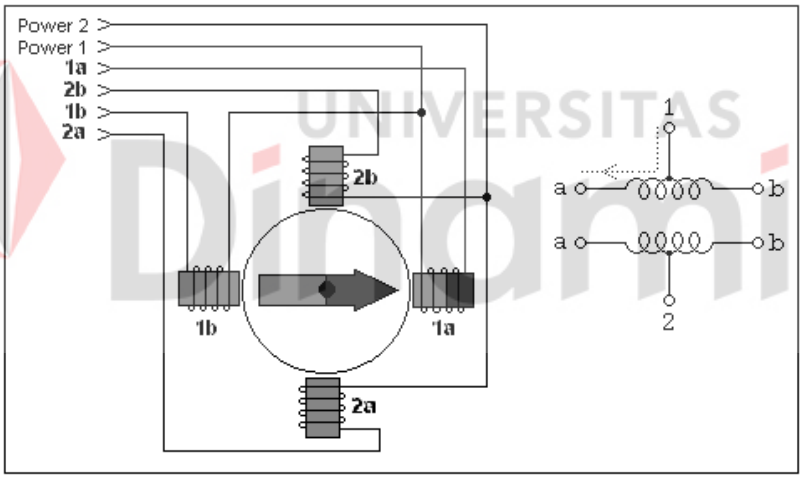
Control *port* (*base address* + 2) seperti yang diperlihatkan pada tabel 2.6 dimaksudkan sebagai *port* yang hanya dapat ditulis saja, walaupun demikian *port* ini juga dapat digunakan untuk menerima data dengan cara mengirimkan *logic high* pada semua pin dalam *register control*.

2.9. Motor *Stepper*

Motor *stepper* adalah perangkat elektronik yang bekerja dengan mengubah pulsa elektronis menjadi gerakan mekanis *diskrit.* Motor *stepper* bergerak berdasarkan urutan pulsa yang diberikan. Karena itu, untuk menggerakkan motor *stepper* diperlukan pengendali motor *stepper* yang membangkitkan pulsa-pulsa periodik. Keunggulan motor *stepper* antara lain adalah:

1. Sudut rotasi motor proporsional dengan pulsa masukan sehingga dapat lebih mudah diatur.
2. Motor dapat langsung memberikan torsi penuh pada saat mulai bergerak.
3. Posisi dan pergerakan repetisinya dapat ditentukan secara presisi.
4. Memiliki respon yang sangat baik terhadap mulai, berhenti, dan berbalik arah (perputaran).
5. Seperti pada motor DC, dapat menghasilkan perputaran yang lambat sehingga beban dapat dikopel langsung ke porosnya.
6. Frekuensi perputaran dapat ditentukan secara bebas dan mudah pada *range* yang luas.

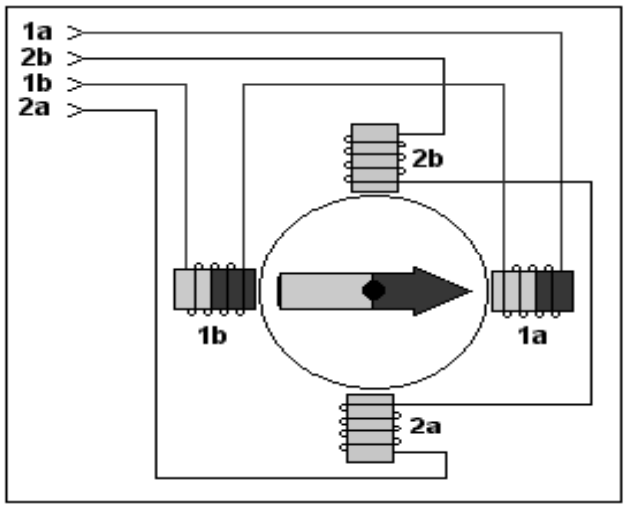
Berdasarkan metode perancangan rangkaian pengendalinya, motor *stepper* dapat dibagi menjadi 2 jenis yaitu motor *stepper unipolar* dan motor *stepper bipolar* (www.doc.ic.ac.uk) (www.stepperworld.com). Rangkaian pengendali motor *stepper unipolar* lebih mudah dirancang karena hanya membutuhkan satu *switch / transistor* setiap lilitannya. Untuk menjalankan dan menghentikan motor ini cukup hanya dengan memberikan pulsa *digital* yang hanya terdiri atas tegangan positif dan nol (*ground*) pada salah satu terminal lilitan motor, sementara terminal lainnya dicatu dengan tegangan positif konstan (VM / Power) pada bagian tengah (*center tap*) dari lilitan, perhatikan gambar 2.10 berikut.



Gambar 2.10 Motor *stepper* dengan lilitan unipolar.

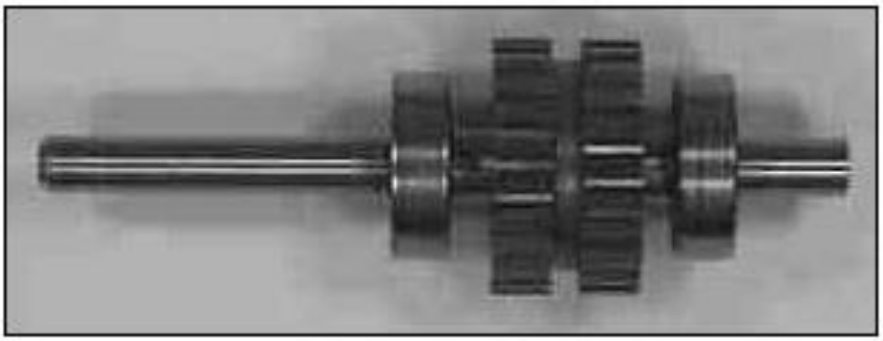
Untuk motor *stepper* dengan lilitan *bipolar*, diperlukan sinyal pulsa yang berubah-ubah dari positif ke negatif dan sebaliknya. Jadi pada setiap terminal lilitan (A dan B) harus dihubungkan dengan sinyal yang mengayun dari positif ke negatif dan sebaliknya, perhatikan gambar 2.11 karena itu dibutuhkan rangkaian pengendali yang lebih kompleks daripada rangkaian pengendali untuk motor

unipolar. Motor *stepper bipolar* memiliki keunggulan dibanding dengan motor *stepper unipolar* dalam hal torsi yang lebih besar untuk ukuran yang sama.

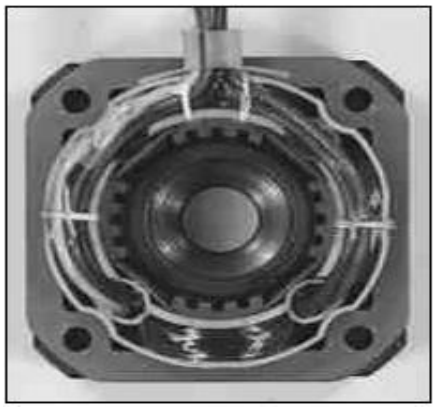


Gambar 2.11 Motor *stepper* dengan lilitan *bipolar*

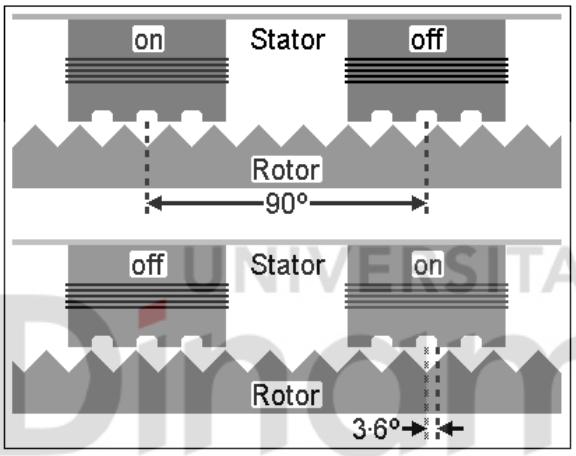
Seperti yang sudah dijelaskan sebelumnya, motor *stepper* digunakan sebagai pengubah energi listrik ke energi gerak. Dalam motor *stepper* terdapat *rotor* dan *stator*. *Rotor* adalah *shaft* yang berputar dengan magnet permanen, sedangkan *stator* adalah *stationary housing* termasuk kutub *coil-wound*.



Gambar 2.12 *Rotor* dalam motor *stepper*



Gambar 2.13 Stator dalam motor *stepper*



Gambar 2.14 Bentuk gigi dalam motor *stepper*

Dengan 25 gigi dan 4 *coils* dalam setiap putaran, motor *stepper* ini menghasilkan 100 *steps* per satu putaran. Spasi antara gigi adalah 360°/25=14.4°. Jadi menghasilkan 360°/100=3.6° per *steps*.

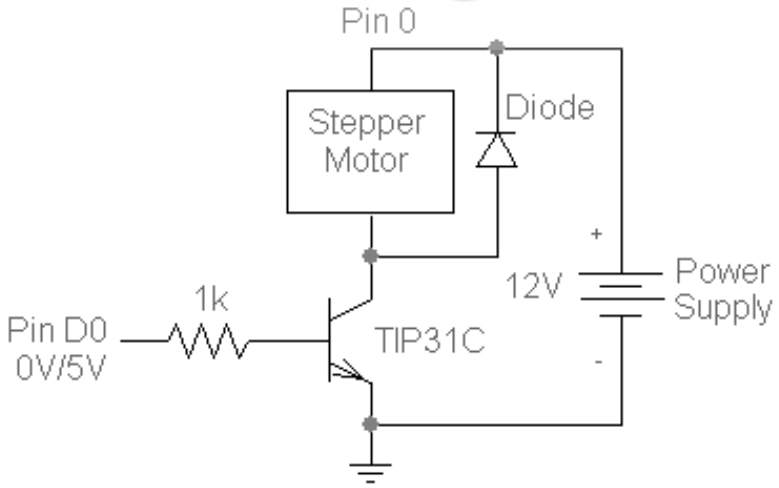
Semakin rapat gigi dalam motor *stepper* maka akan semakin tinggi kepresisian motor *stepper* tersebut.

2.10. *Driver* Motor *Stepper*

Driver motor *stepper* digunakan untuk menguatkan sinyal yang diberikan oleh *interface*. Rangkaian *Driver* ini juga dibutuhkan untuk melindungi komputer atau *microprocessor* dari arus balik (*feedback*) yang ditimbulkan oleh motor *stepper* atau disebut juga EMF (*Electromagnetic Field*).

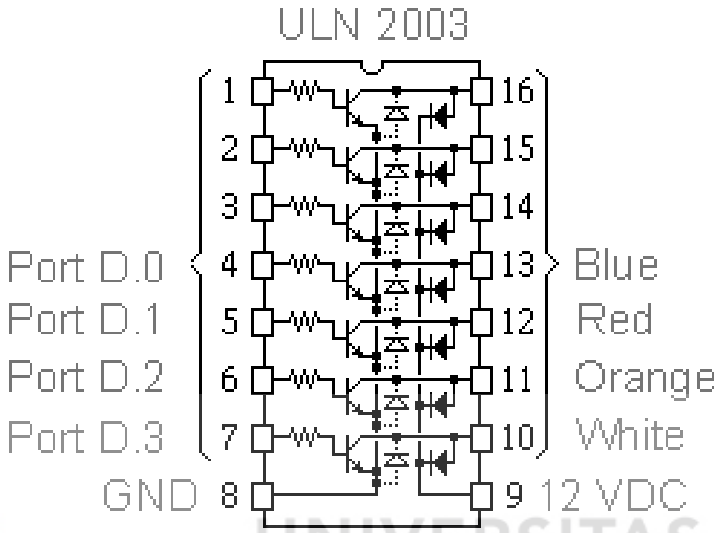
Terdapat beberapa jenis *driver* motor stepper yang bisa digunaka untuk mengendalikan pergeraka dari motor *stepper* yaitu *driver* yang menggunakan transistor, *driver* yang menggunakan IC ULN2003, *driver* yang menggunakan IC 2803, dan lain-lain.

Pada penggunaan transistor sebagai *driver*, dapat digunakan sebanyak empat buah transistor yang dirangkai sedemikian rupa untuk menjalankan motor *stepper*. Untuk bergerak sebanyak satu *step* perlu dikeluarkan signal dari 4 pin port D pada 8515 MCU. Berikut ini gamabar transistor yang digunakan sebagai driver motor stepper (Aprea, 2001).



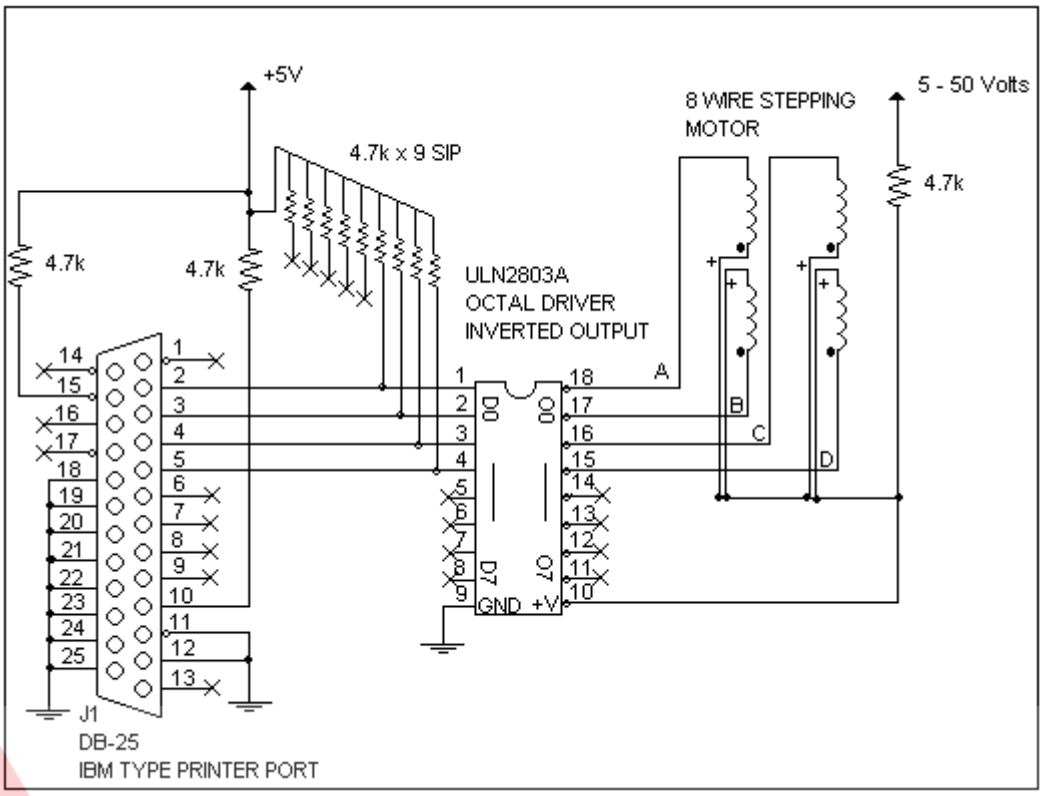
Gambar 2.15 Transistor sebagai *driver* motor *stepper*.

Sedangakn pada penggunaaan IC ULN2003 sebagai driver motor *stepper* prinsipnya sama dengan prinsip penggunaan transistor, ini disebabkan karena pada IC ULN2003 terdapat 7 buah seperti yang ditunjukkan pada gambar berikut (Aprea, 2001).



Gambar 2.16 ULN2003 sebagai *driver* motor *stepper*

Pada penggunaan IC ULN2803 sebagai *driver* motor *stepper*, data yang dikeluarkan oleh *parallel port* di balik kondisinya dari kondisi *high* menjadi *low*. Sesuai dengan karakteristik dari IC ini, maka yang digunakan untuk menggerakkan motor *stepper* adalah data keluaran yang sudah dalam kondisi *low* (Paul, 1994:91). Berikut gambar dari rangkaian *driver* menggunakan IC ULN2803.



Gambar 2.17 ULN2803 sebagai *driver* motor *stepper*

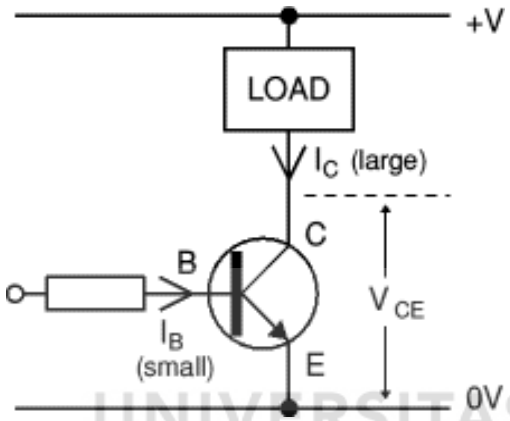
2.11. *Switching* Transistor

Transistor adalah sebuah solid state semikonduktor yang dapat digunakan untuk penguatan, *switching*, stabilisator tegangan, modulasi signal dan lain-lain. Transistor mengalirkan arus dari sumber arus external melalui kedua terminal yang dimilikinya berdasarkan tegangan yang kecil atau arus yang diinginkan menuju ke terminal ketiga.

Ketika transistor digunakan sebagai *switch*, transistor tersebut harus dalah salah satu kondisi yaitu OFF atau dalan kondisi ON. Ketika dalam kondisi ON Vce yang melewati transistor selalu kosong dan itu sering disebut sebagai transistor dalam keadaan saturasi karena tidak dapat melewatkan arus kolektor Ic. Transistor yang digunakan sebagai switch memiliki power yang sangat kecil:

Pada keadaan OFF : *power* = Ic × V_{CE}, tetapi Ic = 0, sehingga powernya adalah nol. Sedangkan pada keadaan ON : *power* = Ic × V_{CE}, tetapi V_{CE} = 0 (selalu), jadi *power*nya sangat kecil.

Ini berarti bahwa transistor tidak akan cepat menjadi panas. Yang penting pada penggunaan transistor sebagai *switch* adalah arus maksimum kolektor Ic(*max*) dan arus minimum gain h_{FE} (Hewes, 2006). Perhatikan gambar berikut ini

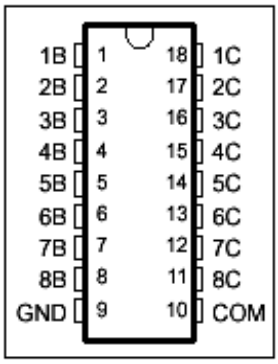


Gambar 2.18 Transistor sebagai *switch*

UNIVERSITAS Dinamika

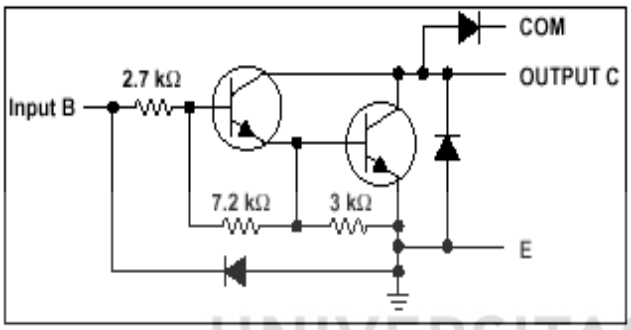
2.12.ULN2803

Setiap *output* dari IC ini dapat mengalirkan arus sebesar 500mA. Dapat digunakan untuk tegangan *output* sampai 50V, sedangkan *input*-nya sesuai dengan berbagai macan tipe IC *logic* (TTL atau CMOS).



Gambar 2.19 Bentuk fisik IC ULN2803

ULN2803 adalah *monolithic* tegangan tinggi, dan arus tinggi yang terbentuk dari susunan transistor Darlington. IC ini terdiri dari 8 pasang transistor NPN darlington yang dapat digunakan untuk *output* tegangan tinggi dengan *common cathode.* Arus *collector* untuk tiap pasang transistor darlington adalah 500mA. Pasangan transistor darlington dapat di-parallel untuk mendapatkan arus yang lebih tinggi.

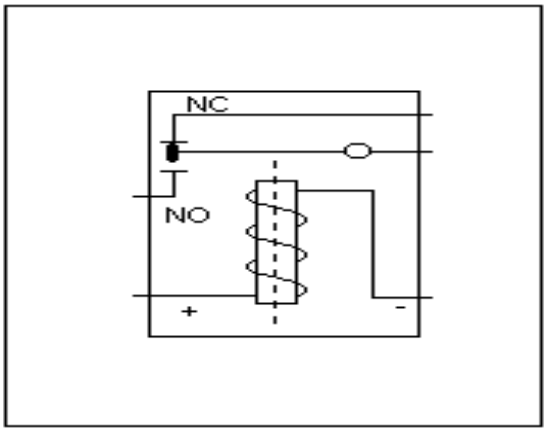


Gambar 2.20 Skematik pasangan transistor Darlington.

UNIVERSITAS Dinamika

2.13. Relay DC

Ada bermacam-macan jenis relay DC yang beredar di pasaran saat ini, tergantung dari besar atau kecilnya nilai tegangan yang akan digunakan. Misal relay DC 6V, 9V, 12V dan lain-lain. Pada prinsipnya relay digunakan untuk mengontrol aliran suatu tegangan dengan cara memutus dan menyambungkan media perantara yang di wakili oleh komponen dalam relay tersebut (Paul, 1994:45).



Gambar 2.21 Relay DC.

2.14. Pemrograman Delphi

Borland Delphi adalah bahasa pemrograman yang digunakan untuk membangun aplikasi *windows* yang berbasis grafis atau dikenal dengan *Graphical User Interface* (GUI). Juga merupakan hasil pengembangan dari bahasa pemrograman yang sebelumnya sudah sangat terkenal dan mudah yaitu Pascal. Bahasa permograman pascal sendiri diciptakan pada tahun 1971 oleh seorang ilmuwan dari Swiss yaitu Nicklous Wirth. Nama Pascal diambil dari nama seorang ahli matematika dan filasafat berkebangsaan Perancis yaitu Blaise Pascal (1623 - 1662).

Pada tahun 1983, sebuah perusahaan perangkat lunak *Borland International Incorporation* merilis Turbo Pascal yang mampu dijalankan pada sistem operasi DOS. Sewaktu Microsoft mulai mengembangkan sistem operasi Windows, yaitu Windows 3.x, Borland International mengembangkan Turbo Pascal yang dapat berjalan pada sistem operasi ini yaitu Turbo Pascal for Windows. Namun tampilan grafis Turbo Pascal sebelumnya kurang begitu menarik, maka Borland International mengembangkannya dengan tampilan yang

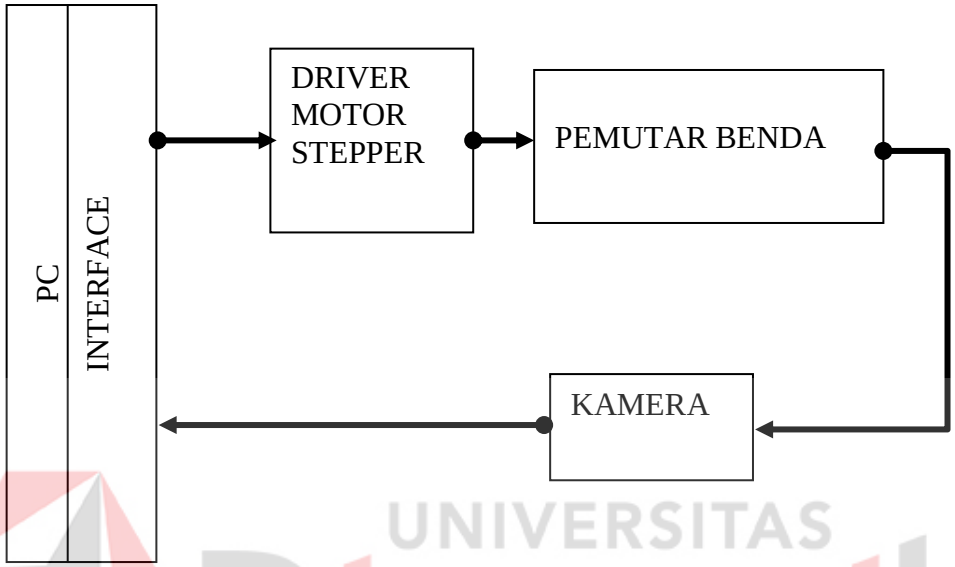
sangat menarik dengan tidak mengurangi kemampuan yang sebelumnya, sehingga pada tahun 1993 muncul sebuah bahasa pemrograman berbasis windows yaitu Borland Delphi 1.0. dengan menggunakan sistem yang disebut RAD (*Rapid Application Development*), sistem ini memanfaatkan bahasa pemrogaman visual yang memudahkan seorang *programmer* untuk mendesain tampilan program yang lebih menarik (*User Friendly Interface*).

UNIVERSITAS Dinamika

BAB III

METODE PENELITIAN

3.1. Perancangan

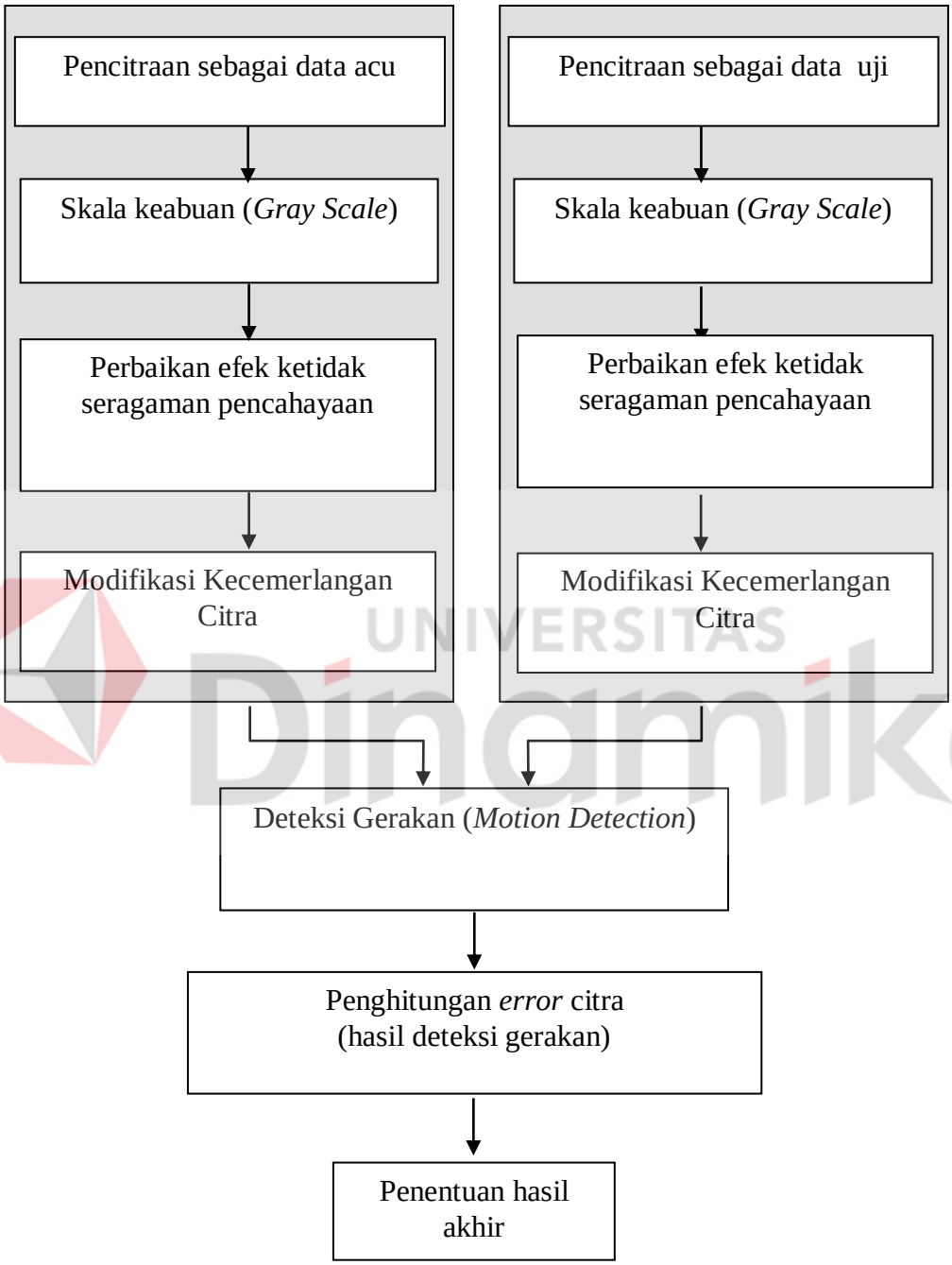


Gambar 3.1 Blok diagram sistem

Perancangan sistem pendeteksi kemasan kaleng yang cacat ini akan di bagi dalam beberapa sub pokok bahasan, masing-masing diantaranya adalah perancangan *software*, perancangan *hardware*. Pada bagian *software* dijelaskan tentang algoritma-algoritma yang digunakan dalam menyelesaiakan permasalahan. Sedangkan pada bagian *hardware* akan dijelaskan tentang *interface*, komponen-komponen yang digunakan untuk membangun *driver* motor *stepper dan* relay. Untuk mekanik akan dijelaskan setelah membahas semua komponen *hardware* diatas.

UNIVERSITAS Dinamika

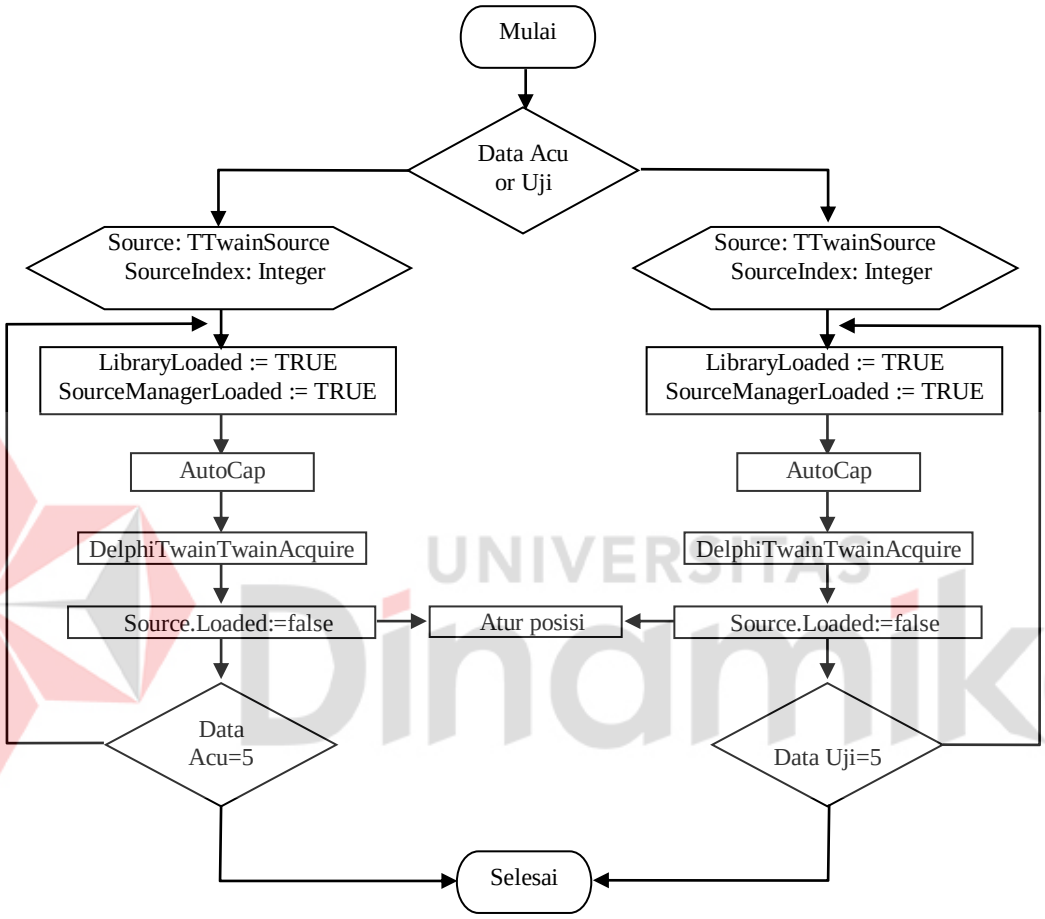
3.1.1. Perancangan *Software*



Gambar 3.2. Blok diagram perancangan *software*

Perancangan *software* untuk sistem pendeteksi kemasan kaleng ini dibangun dengan menggunakan bahasa pemrograman *Borland Delphi 5.*

A. Pengambilan Citra Untuk Data Acu dan Data Uji



Gambar 3.3 Diagram alir pengambilan citra untuk data acu dan data uji

Proses yang ditunjukkan oleh diagram alir pada gambar 3.3 diatas merupakan proses yang pertama kali dilakukan. Ketika program di jalankan maka pengguna harus menentukan salah satu proses, mengambil data acuan atau mengambil data uji. Setelah ditentukan maka proses yang terjadi secara berturut-turut adalah pengaktifan *library* dan *source manager* dari kamera, proses autocap,

penerimaan citra oleh komponen yang ada di delphi, dan penonaktifan *library* dan *source manager* dari kamera.

Hasil dari kedua proses diatas kemudian menjadi masukan untuk proses-proses yang ada berikutnya, yang akan dijelaskan kemudian.

Pada saat aplikasi di eksekusi, keadaan *library* kamera belum diaktifkan oleh karena itu pada saat proses pencitraan terlebih dahulu melakukan pengaktifan dan pemanggilan *library* kamera untuk dijalankan pada aplikasi ini. *Library* yang akan diaktifkan sebanyak lima, karena pengambilan citra untuk data acuan juga sebanyak lima citra. Tetapi pengaktifan untuk *library* berikutnya menunggu *library* sebelumnya di nonaktifkan terlebih dahulu.

```
procedure TFormUtama.LoadCam1;
var
  Source: TTwainSource;
  SourceIndex: Integer;
begin
  FormUtama.DelphiTwain1.LibraryLoaded := TRUE;
  FormUtama.DelphiTwain1.SourceManagerLoaded := TRUE;
  SourceIndex := 0;
  Source :=FormUtama.DelphiTwain1.Source[SourceIndex];
  Source.Loaded := TRUE;
  Source.Enabled := TRUE;
  FormUtama.Autocap;
end;
```

Setelah aktif, pemanggilan dilanjutkan pada proses otomasi pengambilan citra. Program otomasi dapat dilihat pada potongan program berikut:

```
procedure TFormUtama.AutoCap;
begin
  sleep(2000);
  PortOut($378,0);
  PortOut($378,16);
  sleep(1000);
  PortOut($378,0);
end;
```

dalam *listing* potongan program diatas dapat dijelaskan bahwa prosesdur tersebut ketika di eksekusi akan memberikan *delay* selama 2000 *milisecon* atau sama dengan 2 detik, setelah itu prosedur tersebut mengirimkan sinyal berupa data *low* untuk mengantisipasi nilai yang ada pada data bit LPT1, lalu mengirimkan

data 16 desimal untuk memberikan inputan *high* pada data bit ke-6 yang terhubung ke rangkaian relay untuk melakukan pengambilan citra secara automatis. Data bit ke-6 tersebut dibiarkan *high* selama 1 detik agar proses otomatisasi pengambilan citra dapat terjadi. Setelah proses pengambilan citra terjadi prosedur kembali mengirimkan sinyal *low* agar nilai pada data bit kembali *low.*

Terdapat juga *variabel Portout*. *Variabel* tersebut sebenarnya adalah prosedur yang disediakan oleh *file io.dll* yaitu sebuah *file* yang digunakan untuk mengakses *interface* dalam hal ini *parallel port* tanpa harus memikirkan bagaimana mengakses *parallel port* menggunakan bahasa *assembly*.

Setelah proses pengambilan citra pertama, citra yang didapat tersimpan dalam memori ditampung oleh prosesdur berikut ini untuk kemudian di simpan dalam direktori yang disediakan.

```
procedure TFormUtama.DelphiTwain1TwainAcquire(Sender: TObject;
  const Index: Integer; Image: TBitmap; var Cancel: Boolean);
begin
  FormUtama.Timer1.Enabled:=false;
  FormUtama.Image1.Picture.Assign(Image);
  FormUtama.Image1.Picture.SaveToFile('C:\TA\GALIH\GAMBAR\ACUAN\
                                      '+IntToStr(data)+'.bmp');
  UnLoadCAm1;
  FormUtama.Timer2.Enabled:=true;
end;
```

Selesai menyimpan citra, prosedur diatas memanggil prosedur untuk menonaktifkan *library* yang telah dipakai, sekaligus memanggil prosedur *timer* yang berisi *code* program untuk mengkatifkan *library* berikutnya.

```
procedure TFormUtama.UnLoadCam1;
var
  Source: TTwainSource;
  SourceIndex: Integer;
begin
  SourceIndex := 0;
  Source := FormUtama.DelphiTwain1.Source[SourceIndex];
  source.DisableSource;
  source.Modal:=false;
```

```
  source.ShowUI:=false;
  source.Loaded:=false;
  FormUtama.AturPosisi;
  FormUtama.SetFocus;
end;
```

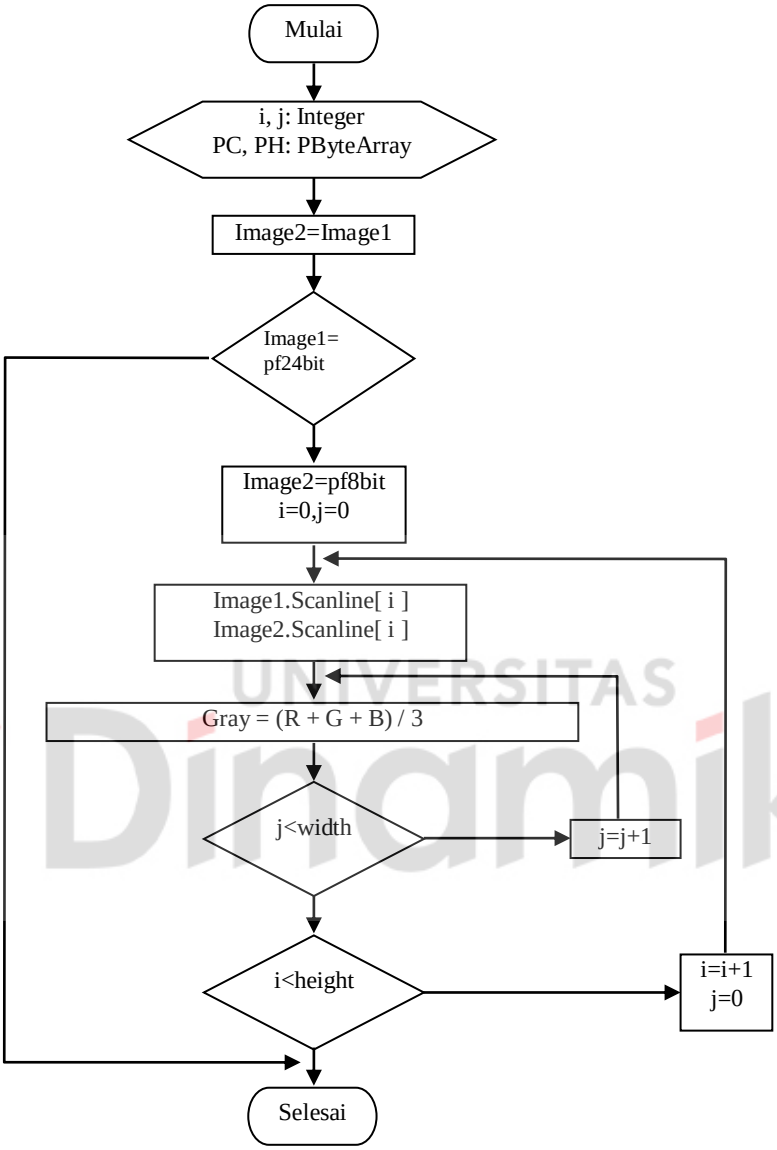
Dalam prosedur penonaktifan *library* yang sudah dipakai, terdapat *code* program untuk mengatur posisi citra yang tersimpan untuk diletakkan kedalan *form* yang sudah dibuat sebelumnya. Berikut ini potongan *listing* programnya.

```
procedure TFormUtama.AturPosisi;
begin
  if Formutama.GroupBox1.Enabled=true then
    begin
      FormUtama.Image1.Picture.LoadFromFile('C:\TA\GALIH\GAMBAR\ACUAN\
                                            '+IntToStr(data)+'.bmp');
      FormUtama.StringGrid.Cells[0,0]:='Urutan';
      FormUtama.StringGrid.Cells[1,0]:='Nama';
      FormUtama.StringGrid.Cells[0,baris]:=IntToStr(data);
      FormUtama.StringGrid.Cells[kolom,baris]:=IntToStr(data)+'.bmp';
      FormUtama.Image1.Height:=FormUtama.Image1.Picture.Height;
      FormUtama.Image1.Width:=FormUtama.Image1.Picture.Width;
      FormUtama.Image2.Height:=FormUtama.Image1.Picture.Height;
      FormUtama.Image2.Width:=FormUtama.Image1.Picture.Width;
      FormUtama.ProsesGray;
      FormUtama.ProsesPerbaikanEfekCahaya;
      FormUtama.Image2.Picture.SaveToFile('C:\TA\GALIH\GAMBAR\ACUAN\
                                          Acuan'+IntToStr(data)+'.bmp');
    end;
  .
  .
end;
```

UNIVERSITAS Dinamika

Begitu seterunya proses pencitraan akan berulang sebanyak citra acuan yang diambil atau sebanyak lima buah citra acuan nantinya. Untuk pengambilan data uji prosesnya sama dengan proses pengambilan data acu.

B. Skala Keabuan (*Gray Scale*)



Gambar 3.4 Diagram alir algoritma *Gray Scale*

Pada proses yang ditunjukkan oleh gambar 3.4 diatas, citra yang diperoleh dari proses sebelumnya satu persatu dirubah informasinya dari citra dengan informasi kedalaman bit sebesar 24 bit citra warna, menjadi citra *gray scale* dengan informasi kedalaman bit sebesar 8 bit. Proses pengkonversian citra

tersebut mengacu pada persamaan 2.2 dimana proses dilakukan berdasarkan nilai baris dan nilai kolom yang sudah di definisikan.

Bagian-bagian proses diatas dilakukan untuk satu buah citra dan akan diulang sebanyak citra yang di dapat oleh kamera.

Setelah citra diletakkan pada *form*, maka proses selanjutnya adalah merubah citra tersebut menjadi citra *gray scale* karena citra yang dihasilkan dari proses pencitraan sebelumnya adalah merupan citra *true color*. Untuk dapat merubah citra *true color* maka dilakukan seperti pada potongan *listing* program berikut

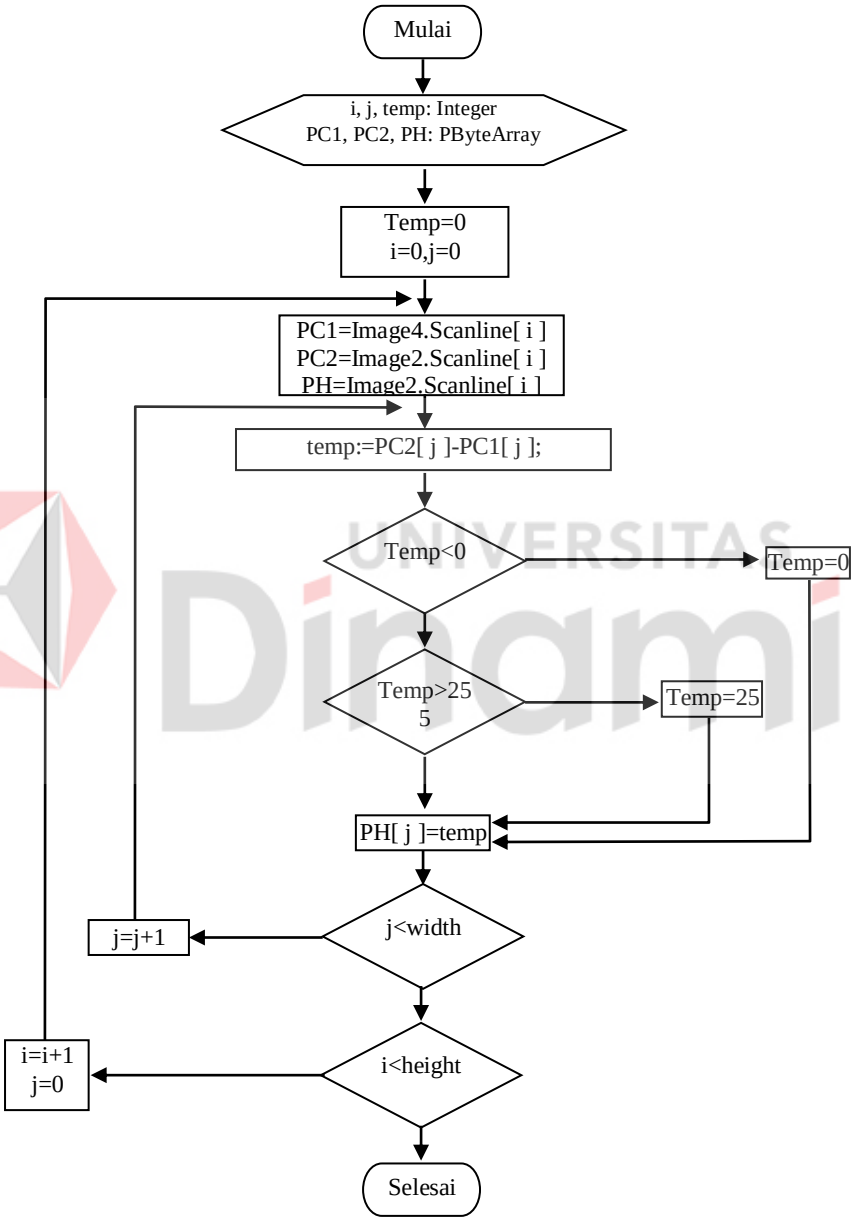
```
procedure TFormUtama.ProsesGray;
var
  i,j:Integer;
  PC,PH:PByteArray;
begin
  FormUtama.Image2.Picture:= FormUtama.Image1.Picture;
  if(FormUtama.Image1.Picture.Bitmap.PixelFormat=pf24bit)then
   begin
     FormUtama.Image2.Picture.Bitmap.PixelFormat:=pf8bit;
     FormUtama.Image2.Picture.Bitmap.
                                  Palette:=CreatePalette(PaletKeabuan.lpal);
     for i:=0 to FormUtama.Image1.Picture.Height-1 do
       begin
         PC:=FormUtama.Image1.Picture.Bitmap.ScanLine[i];
         PH:=FormUtama.Image2.Picture.Bitmap.ScanLine[i];
         for j:=0 to FormUtama.Image1.Picture.Width-1 do
           PH[j]:=Round((PC[3*j]+PC[3*j+1]+PC[3*j+2])/3);
       end;
   end
end;
```

Diatas dapat dilihat, untuk merubah suatu citra *true color* menjadi *gray scale* maka dari citra tersebut harus dicari rerata dari ketiga nilai warna yang terdapat pada tiap *pixel* citra *true color*. Pengambilan *nilai* warna pada *pixel* disesuaikan dengan baris kolom yang sudah di atur sedemikian rupa.

Setelah semua *pixel* selesai dicari nilai reratanya, maka selesai pulalah proses perubahan suatu citra *true color* menjadi citra *gray scale*, karena setelah

nilai rerata ditemukan maka nilai rerata tersebut langsung di simpan pada *pixel* yang bersangkutan menggantikan nilai *pixel* sebelumnya.

C. Perbaikan efek ketidak seragaman pencahayaan



Gambar 3.5 Diagram alir algoritma Perbaikan efek ketidak seragaman pencahayaan

Sebagaimana yang telah di jelaskan pada bab sebelumnya. Bahwa pencahayaan (*illumination*) yang tidak merata saat dilakukan proses pengambilan citra mengakibatkan citra yang didapat tidak sesuai dengan kenyataan. Perbaikan citra akibat ketidak seragaman pencahayaan dapat dilakukan dengan mengurangi citra tersebut dengan citra latarnya, yang diperoleh dengan melakukan pencitraan tanpa adanya obyek, proses ini mengimplementasikan persamaan 2.4. Perhatikan gambar 3.5 yang menunjukkan algoritma perbaikan efek ketidak seragaman pencahayaan.

Pada bagian hasil proses yang melebihi *range* nilai *pixel* dari citra hasil dirubah agar berada diantara *range* yaitu antara 0-255 atau 8 bit.

Bagian-bagian proses diatas juga dilakukan untuk satu buah citra, dan akan diulang sebanyak jumlah citra yang di proses.

Perbaikan efek pencahayaan ini dilakukan ketika citra sudah dalam keadaan *gray scale*, sehingga untuk prosesnya lebih mudah bila dibandingkan dengan perbaikan efek pencahayaan dalam bentuk citra *true color*.

```
procedure TFormUtama.ProsesPerbaikanEfekCahaya;
var
  i,j,temp:Integer;
  PC1,PC2,PH:PByteArray;
begin
  for i:=0 to FormUtama.Image2.Picture.Height-1 do
    begin
      PC1:=FormUtama.Image2.Picture.Bitmap.ScanLine[i];
      PC2:=FormUtama.Image4.Picture.Bitmap.ScanLine[i];
      PH:=FormUtama.Image2.Picture.Bitmap.ScanLine[i];
        for j:=0 to FormUtama.Image2.Picture.Width-1 do
          begin
            temp:=PC2[j]-PC1[j];
            if(temp<0)then
              temp:=0;
            if(temp>255)then
              temp:=255;
            PH[j]:=temp;
          end;
    end;
end;
```

Perhatikan potongan *listing* program diatas, setelah citra selesai dirubah menjadi *gray scale*, maka citra tersebut kembali diproses. Proses kali ini adalah proses dimana citra tersebut akan diambil nilai-nilai warna pada *pixel*-nya untuk kemudian dikurangkan dengan nilai-nilai warna pada *pixel* citra latar yang sebelumnya sudah di siapkan. Jika nilai hasil pengurangan tersebut lebih kecil dari nol maka prosedur diatas merubah agar nilai tersebut menjadi nol. Begitu juga apabila hasil pengurangan tersebut nilainya lebih besar dari 255 maka prosedur merubah nialai tersebut menjadi 255. Ini dimaksudkan agar nantinya nilai yang ada dalam *pixel* citra uji akan memiliki *range* antara 0-255.

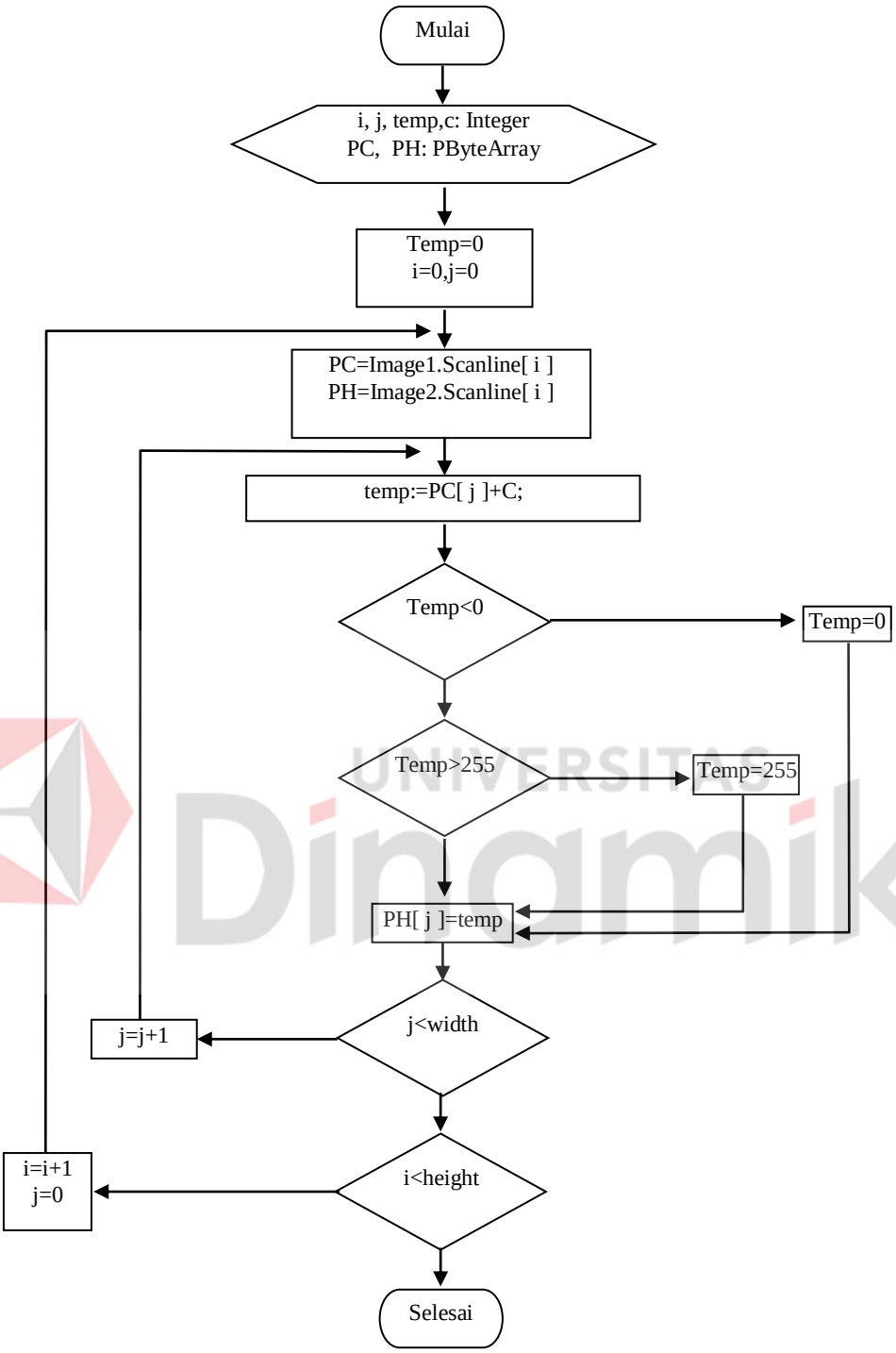
Tujuan dari proses ini adalah agar pencahayaan didalam citra tersebut merata sehingga sesuai dengan aslinya (Achmad, 2005:133). Setelah selesai diperbaiki efek ketidak seragaman pencahayaannya, citra uji tersebut kemudian disimpan sebagai data uji yang akan digunakan nantinya.

UNIVERSITAS Dinamika

D. Modifikasi Kecemerlangan Citra (*Brightness*)

Pengambilan citra yang dilakukan diruang yang memiliki pencahayaan yang kurang menyebabkan citra tidak tampak jelas, dengan memodifikasi kecemerlangan citra maka citra akan tampak lebih jelas dan lebih terang.

Proses yang terjadi pada modifikasi kecemerlangan citra ini adalah, bertambahnya skala keabuan (C) pada seluruh bagian (setiap titik) pada citra sesuai dengan nilai baris kolomnya sehingga objek dalam citra yang diolah dapat terlihat dengan jelas sesuai dengan persamaan 2.5 pada landasan teori.



Gambar 3.6 Diagram alir Modifikasi Kecemerlangan Citra (*Brightness*)

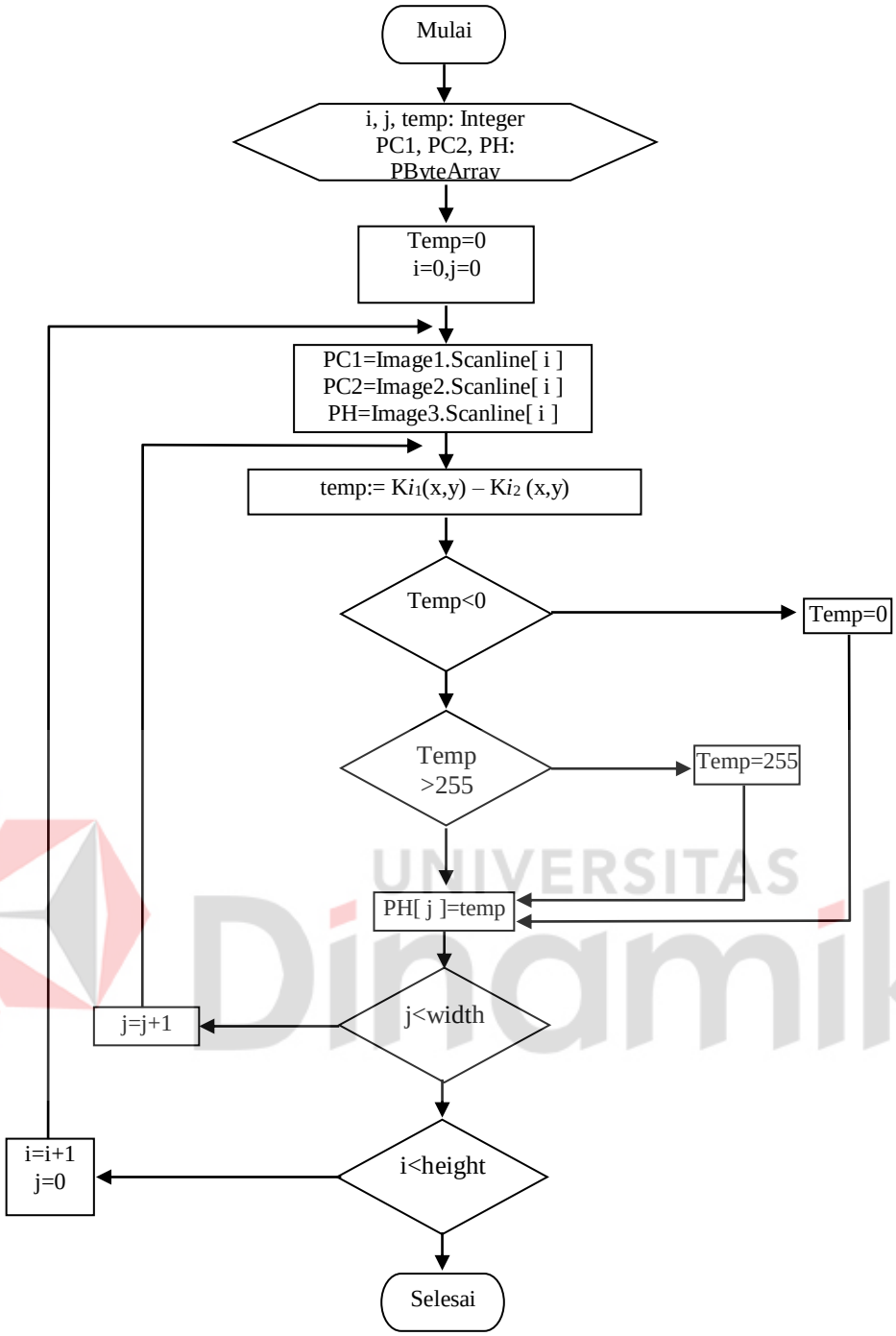
Proses ini bertujuan untuk memperjelas objek yang ada dalam citra yang disebabkan oleh kurangnya kecemerlangan (*brightness*) dari citra tersebut.

```
procedure TFormUtama.Cemerlang;
var
  temp,i,j:integer;
  PC,PH:PByteArray;
begin
  for i:=0 to FormUtama.Image2.Picture.Height-1 do
     begin
       PC:=FormUtama.Image2.Picture.Bitmap.ScanLine[i];
       PH:=FormUtama.Image2.Picture.Bitmap.ScanLine[i];
       for j:=0 to FormUtama.Image2.Picture.Width-1 do
         begin
           temp:=PC[j]+128;
           if(temp>255)then
             temp:=255;
           if(temp<0)then
             temp:=0;
           PH[j]:=temp;
         end;
     end;
 end;
```

E. Deteksi Gerakan (*Motion Detection*)

Pada proses deteksi gerakan ini langkah-langkah yang diambil sama persis dengan langkah-langkah pada proses perbaikan efek ketidak seragaman pencahayaan. Yang membedakan adalah, pada proses deteksi gerakan ini yang diproses merupakan sama-sama citra benda yang sudah dalam bentuk *gray scale* baik itu citra benda acuan maupun citra benda uji yang diambil dari kamera dengan sudut yang sama.

Pada hasil deteksi gerakan ini telah dilakukan sedikit modifikasi pada perbandingan nilai yang tersimpandalam variabel temp, yaitu apabila hasil yang didapat nilainya kurang dari 0 maka nilainya dirubah menjadi nol, sedangkan untuk nilai hasil deteksi gerakan yang melebihi 255 dirubah menjadi 255. Hal ini dilakukan agar nilai yang diperoleh sesuai range yang diinginkan.



Gambar 3.7 Diagram alir Deteksi Gerakan (*Motion Detection*)

Setelah semua data siap (data citra acuan dan citra uji), maka proses selanjutnya adalah proses pendeteksian gerakan. Proses ini dimaksudkan agar apabila objek dalam citra acuan berbeda dengan objek pada citra uji (tetapi dalam

sudut pengambilan yang sama) dapat diketahui dan nantinya data yang dihasilkan oleh proses ini akan digunakan untuk melakukan perhitungan *error*

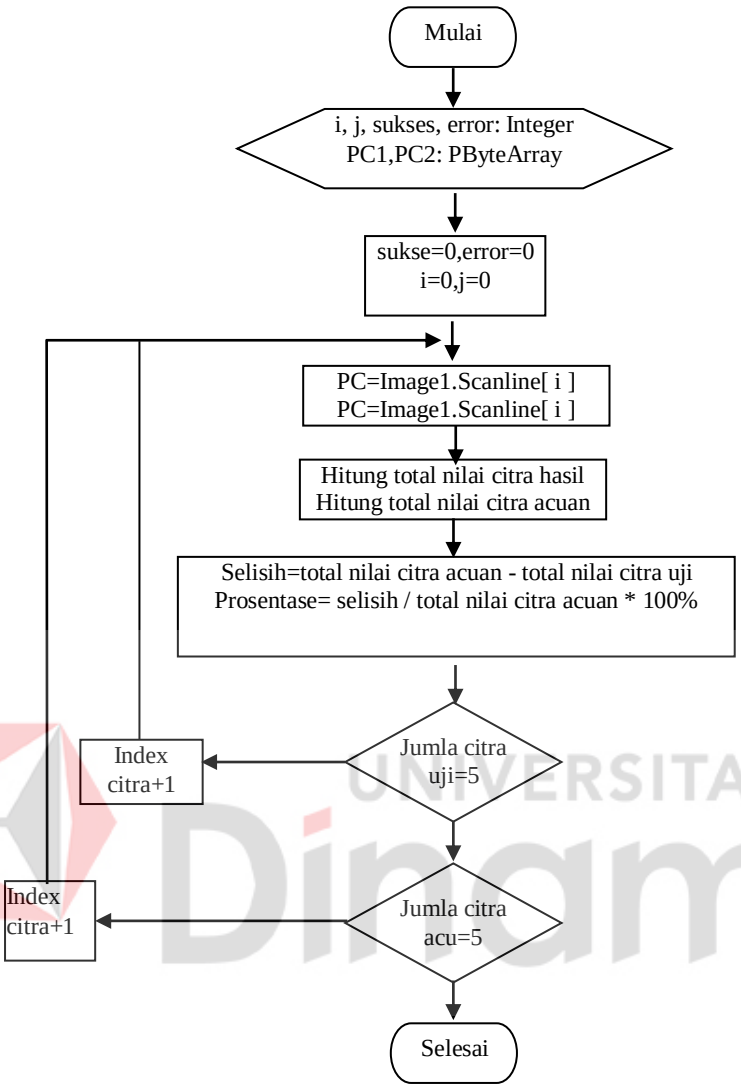
```
procedure TFormUtama.ProsesDeteksiGerak;
var
  i,j,temp:Integer;
  PC1,PC2,PH:PByteArray;

begin
  FormUtama.Image3.Picture:=FormUtama.Image1.Picture;
  FormUtama.Image3.Picture.Bitmap.PixelFormat:=pf8bit;
  for i:=0 to FormUtama.Image1.Picture.Height-1 do

    begin
      PC1:=FormUtama.Image1.Picture.Bitmap.ScanLine[i];
      PC2:=FormUtama.Image2.Picture.Bitmap.ScanLine[i];
      PH:=FormUtama.Image3.Picture.Bitmap.ScanLine[i];
        for j:=0 to FormUtama.Image1.Picture.Width-1 do

           begin
            temp:=PC2[j]-PC1[j];
            if(temp<0)then
              temp:=0;

            if(temp>255)then
              temp:=255;
            PH[j]:=temp;
          end;
    end;
end;
```

Pada prosedur diatas dapat diketahui bahwa yang diproses adalah *pixel-pixel* dari data citra acuan dan *pixel-pixel* dari data citra uji. Dimana nilai *pixel* dari citra acuan dikurangkan dengan nilai *pixel* dari citra uji dengan indeks baris kolom yang sama. Hampir sama dengan proses perbaikan efek ketidak seragaman pencahayaan, yang berbeda adalah citra yang diolah adalah merupakan citra objek. Hasil pengurangan dari proses tersebut apabila nilainya lebih kecil dari nol maka nilai tersebut oleh prosedur dibuat nol. Untuk nilai yang lebih besar dari 255 prosedur juga merubahnya menjadi 255. Jadi perubahan tersebut dilakukan agar dapat berguna dalam perhitungan *error* yang akan dibahas kemudian.

Prosedur ini juga di eksekusi sebanyak citra yang di olah atau sebanyak 5 kali untuk setiap satu kali proses.

UNIVERSITAS Dinamika

F. Penghitungan error citra



Gambar 3.8 Diagram alir Penghitungan error citra

Setelah semua proses pengolahan citra selesai, maka tinggal proses penghitungan *error* yang di hasilkan untuk tiap-tiap citra hasil. Perhatikan gambar 3.8, pada diagram alir yang di sajikan hanya proses penghitungan *error* untuk satu citra hasil yang nilai-nilainya telah disimpan pada variabel *error* dan variabel sukses, dimana nantinya nilai yang tersimpan tersebut akan digunakan untuk menghitung prosentase kesalahan dan prosentase keberhasilan dari satu citra hasil.

Penghitungan selisih *error* suatu citra dilakukan dengan menggunakan jumlah nilai *pixel* citra acuan yang tersimpan di variabel jumlah Acuan dikurangi dengan jumlah nilai *pixel* pada citra hasil. Sedanglan untuk prosentasenya, dihitung dari jumlah nilai *pixel* citra hasil dibagi dengan jumlah nilai *pixel* citra acuan kemudian dikalikan dengan 100 agar nilainya memiliki *range* antara 0-100. Sebaliknya untuk menghitung persentase keberhasilan dalam suatu citra dapat dilakukan dengan cara yang sama atau lebih mudahnya mengurangkan nilai 100% dengan nilai persentase *error* yang di dapat untuk satu buah citra hasil.

Berikut ini potongan *listing* program yang menunjukkan proses penghitungan *error*.

```
procedure TFormUtama.HasilUji;
var
  fileheader:TBitmapFileheader;
  infoheader:TBitmapInfoheader;
  gambar1,gambar2,gambar3,gambar4,gambar5:TFilestream;
  i,j:Integer;
  Error1,Error2,Error3,Error4,Error5:real;
  Sukses1,Sukses2,Sukses3,Sukses4,Sukses5:integer;
  PC:PByteArray;
  c:array[0..255]of integer;
  cMax:Integer;
  PH:PByteArray;


begin
  jumlah1:=0;
  jumlah2:=0;
  jumlah3:=0;
  jumlah4:=0;
  jumlah5:=0;
  jumlah11:=0;
  jumlah12:=0;
  jumlah13:=0;
  jumlah14:=0;
  jumlah15:=0;
.
.
.
              for i:=0 to FormReport.Image1.Picture.Height-1 do
               begin
                 PC:=FormReport.Image1.Picture.Bitmap.ScanLine[i];
                 for j:=0 to FormReport.Image1.Picture.Width-1 do
                   if PC[j]>0 then
                    begin
                      jumlah1:=PC[j];
                      jumlah11:=jumlah11+jumlah1;
                    end;
               end;
.
.
.
            Error1:=jumlah11;
            FormReport.Edit26.Text:=floatToStr(
```

```
                              100-((jumlah11/jumlahAcuan11)*100));
                FormReport.Edit27.Text:=floatToStr(
                                    (jumlah11/jumlahAcuan11)*100);
                FormReport.Edit41.Text:=FloatToStr(Error1);
.
.
.
                   for i:=0 to FormReport.Image2.Picture.Height-1 do
                   begin
                     PC:=FormReport.Image2.Picture.Bitmap.ScanLine[i];
                     for j:=0 to FormReport.Image2.Picture.Width-1 do
                       if PC[j]>0 then
                        begin
                          jumlah2:=PC[j];
                          jumlah12:=jumlah12+jumlah2;
                        end;
                   end;
.
.
.
                FormReport.Edit28.Text:=FloatToStr(
                              100-((jumlah12/jumlahAcuan12)*100));
                FormReport.Edit29.Text:=FloatToStr(
                                    (jumlah12/jumlahAcuan12)*100);
                FormReport.Edit42.Text:=FloatToStr(Error2);
.
.
.
                 for i:=0 to FormReport.Image3.Picture.Height-1 do
                   begin
                     PC:=FormReport.Image3.Picture.Bitmap.ScanLine[i];
                     for j:=0 to FormReport.Image3.Picture.Width-1 do
                       if PC[j]>0 then
                        begin
                          jumlah3:=PC[j];
                          jumlah13:=jumlah13+jumlah3;
                        end;
                   end;
.
.
.
                FormReport.Edit30.Text:=floatToStr(
                              100-((jumlah13/jumlahAcuan13)*100));
                FormReport.Edit31.Text:=floatToStr(
                                    (jumlah13/jumlahAcuan13)*100);
                FormReport.Edit43.Text:=FloatToStr(Error3);
.
.
.
                 for i:=0 to FormReport.Image4.Picture.Height-1 do
                   begin
                     PC:=FormReport.Image4.Picture.Bitmap.ScanLine[i];
                     for j:=0 to FormReport.Image4.Picture.Width-1 do
                       if PC[j]>0 then
                        begin
                          jumlah4:=PC[j];
                          jumlah14:=jumlah14+jumlah4;
                        end;
                   end;
.
.
.
                FormReport.Edit32.Text:=floatToStr(
                              100-((jumlah4/jumlahAcuan14)*100));
                FormReport.Edit33.Text:=floatToStr(
                                   (jumlah14/jumlahAcuan14)*100);
                FormReport.Edit44.Text:=FloatToStr(Error4);
.
.
```

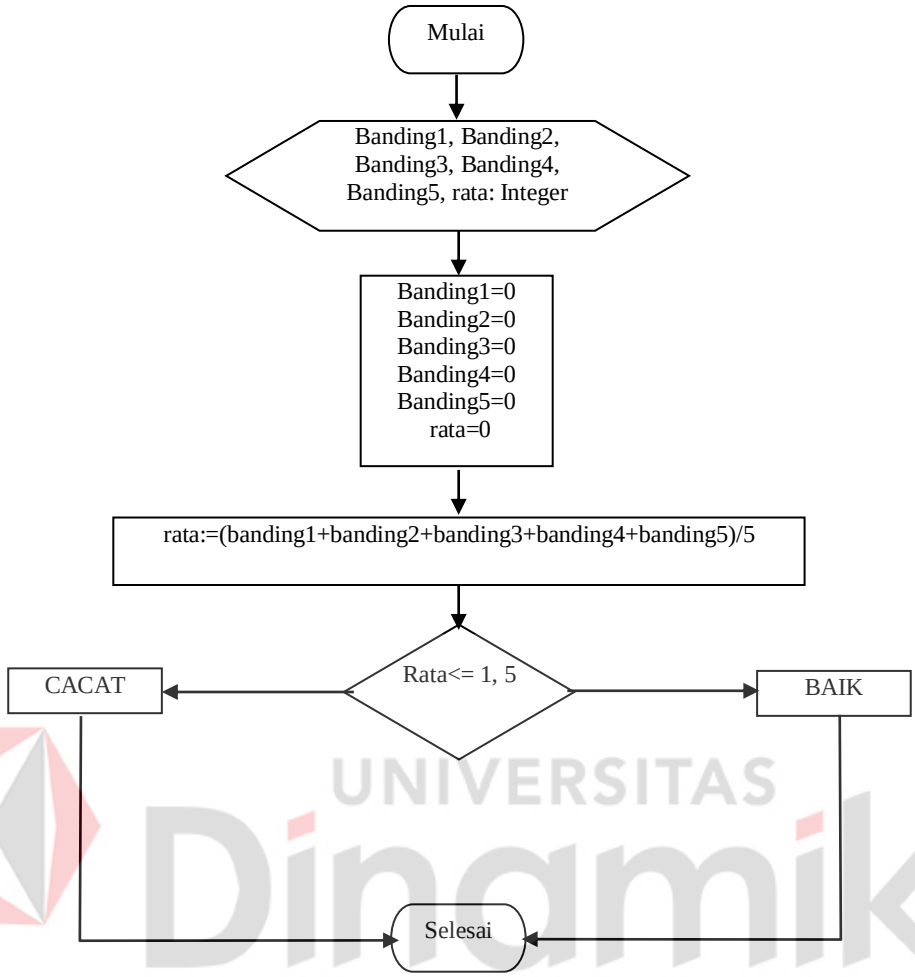
UNIVERSITAS Dinamika

```
.
              for i:=0 to FormReport.Image5.Picture.Height-1 do
                begin
                  PC:=FormReport.Image5.Picture.Bitmap.ScanLine[i];
                  for j:=0 to FormReport.Image5.Picture.Width-1 do
                    if PC[j]>0 then
                     begin
                       jumlah5:=PC[j];
                       jumlah15:=jumlah15+jumlah5;
                     end;
                end;
.
.
.            Error5:=jumlah15;
             FormReport.Edit34.Text:=floatToStr(
                                  100-((jumlah15/jumlahAcuan15)*100));
             FormReport.Edit35.Text:=floatToStr(
                                        (jumlah15/jumlahAcuan15)*100);
             FormReport.Edit45.Text:=FloatToStr(Error5);
.
.
.
end;
```

Pada prosedur diatas penghitungan *error* citra hasil dilakukan langsung untuk kelima citra yang ada, untuk kemudian data *error* yang dihasilkan akan ditampilkan pada *form* dalam bentuk persentase sehingga user dapat mengetahui seberapa besar persentase *error* dan juga seberapa besar prosentase keberhasilan atau kesesuaian citra uji dengan citra acuan.

Bagian-bagian proses diatas juga dilakukan untuk satu buah citra, dan akan diulang sebanyak jumlah citra yang di proses.

G. Penentuan hasil akhir



Gambar 3.9 Diagram alir penentuan hasil akhir

Pada penentuan hasil akhir ini mengacu pada persamaan 2.9, akan ada dua kemungkinan yang mungkin terjadi terhadap kemasan kaleng yang di ujikan. Pertama kemasan kaleng akan di anggap sebagai kemasan kaleng yang BAIK. Dan kedua, kemasan kaleng akan dianggap sebagai kemasan kaleng yang CACAT. Penentuan hasil ini dilakukan berdasarkan hasil dari nilai rata-rata prosentase *error* yang di dapat (tergantung jumlah citra yang dihasilkan).

Setelah *error* citra hasil didapatkan, sekarang tinggal menentukan apakah kemasan kaleng kaleng yang citranya diujikan tersebut di kategorikan BAIK atau dikategorikan sebagai CACAT. Berikut ini listingnya:

```
procedure TFormUtama.SistemReport;
var
  i,j,banding1,banding2,banding3,banding4,banding5,rata:Integer;
begin
.
.
.
.
  banding1:=StrToInt(FormReport.edit27.Text);
  banding2:=StrToInt(FormReport.edit29.Text);
  banding3:=StrToInt(FormReport.edit31.Text);
  banding4:=StrToInt(FormReport.edit33.Text);
  banding5:=StrToInt(FormReport.edit35.Text);
  rata:=(banding1+banding2+banding3+banding4+banding5)/5;
  FormSistemReport.Edit5.Text:=IntToStr(rata);

  if StrToInt(FormSistemReport.Edit5.Text)<=1.5 then
    FormSistemReport.Label3.Caption:='  BAIK'
  else
    FormSistemReport.Label3.Caption:='CACAT';
.
.
.
.
end;
```

Hasil prosentase *error* pada citra hasil yang telah ditampilkan kemudian digunakan kembali untuk mengkategorikan keadaan kemasan kaleng. Menghitung rata-rata dari kelima *error* citra yang ada, kemudian apabila hasilnya kurang dari sama dengan 1.5 maka kemasan kaleng tersebut masuk dalam kategori kemasan kaleng yang BAIK, sedangkan apabila hasil yang diperoleh dari penghitungan rata-rata tersebut melebihi 1.5 maka kemasan kaleng tersebut masuk dalam kategori CACAT. Pemilihan angka 1.5 disebabkan karena, pada saat melakukan penelitian angka rata-rata yang didapat dari hasil uji coba 10 kali kemasan kaleng. Angka tersebut dapat dirubah sesuai dengan keinginan peneliti, berdasarkan hasil penelitian yang dilakuan oleh masing-masing peneliti.

UNIVERSITAS Dinamika

3.1.2. Perancangan *Hardware*

Untuk mengaplikasikan penelitian ini diperlukan perancangan elektronik. Elektronik yang digunakan tersusun atas dua bagian penting yaitu *interface* dan *driver* motor *stepper.*

Interface yang digunakan adalah *parallel port* yang terhubung dengan *driver* motor *stepper* yang digunakan untuk menjalankan pemutar benda dalam hal ini adalah motor *stepper.*

A. Berkomunikasi dengan *Interface*

Setiap akan melakukan komunikasi (proses *input / output*) antara komputer dengan *interface* maka hal pertama yang mutlak harus diketahui adalah alamat untuk melakukan komunikasi tersebut.

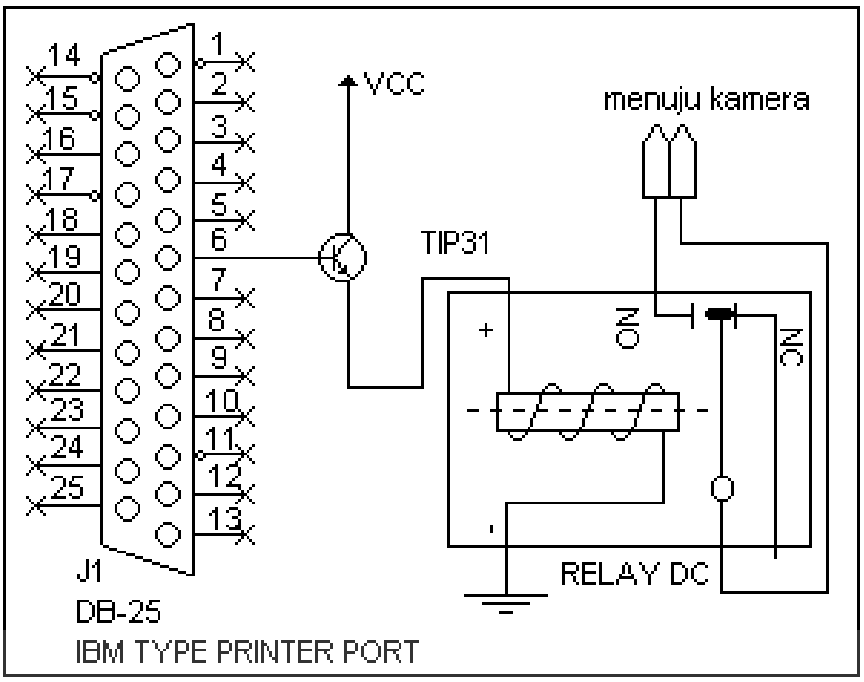
Interface yang digunakan adalah *parallel port,* jadi prosedur yang digunakan untuk mengirim data adalah mengirimkan data secara langsung menuju alamat dari *port* tersebut. Dalam mengerjakan tugas akhir alamat dari *parallel port* yang digunakan adalah 378Hex.

Data yang dikirimkan secara langsung ke alamat ini, tidak perlu mengalamai *inverting* karena data tersebut dikirimkan ke register data pada *parallel port* yang langsung dapat diguanakan dalam implementasi alat. Pada data register yang digunakan adalah D0 - D3 untuk menggerakkan motor *stepper*, D4 digunakan untuk mengendalikan relay. Sedangkan untuk D5 – D7 tidak digunakan. Potongan progran yang digunakan untuk menginisialisasi *port parallel* ini adalah sebagai berikut.

```
procedure PortOut(Port : Word; Data : Byte); stdcall; external 'io.dll';
```

UNIVERSITAS Dinamika

B. Rangkaian Relay



Gambar 3.10 rangkaian relay

Gambar 3.10 adalah skematik dari rangkaian relay yang digunakan untuk proses *automatic capture* pada kamera. Ketika komputer mengirimkan data *high* pada pin ke-6 maka TIP31 akan meneruskan arus ke arah relay untuk menghubungkan tombol kamera sehingga proses *snap* berlangsung.

```
procedure TFormUtama.AutoCap;
begin
  sleep(2000);
  PortOut($378,0);
  PortOut($378,16);
  sleep(1000);
  PortOut($378,0);
end;
```

Pada prosedur diatas menjelaskan bahwa, pada saat prosedur ini diakses pertama kali yang dilakukan adalah memberikan *delay* selama 2 detik agar prosedur tidak melakukan pengiriman data dahulu, kemudian pada alamat data *port* dari *port parallel* dikirimkan nilai desimal 0 agar nilai dari data *port* tidak

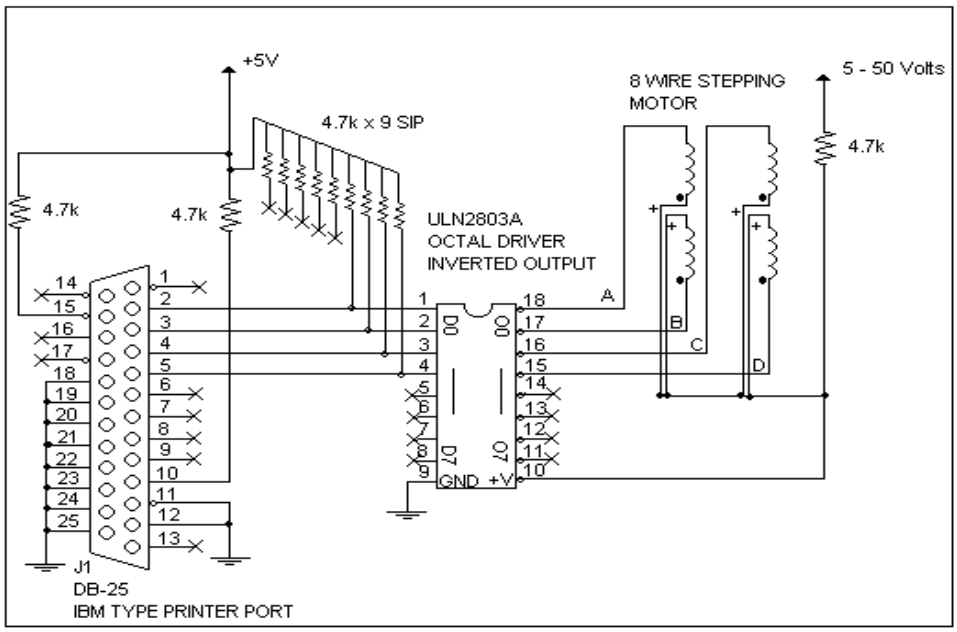
ada yang bernilai *high*, kemudian setelah itu dikirimkan nilai desimal 16 agar data *port* mengeluarkan sinyal *high* pada pin ke 6 atau pada D4. Kemudian setelah itu dikirimkan *delay* selama 1 detik agar proses *snap* terjadi selama 1 detik, setelah itu kembali dikirimkan nilai desimal 0 agar nilai dari data *port* tidak ada yang bernilai *high.*

C. *Driver* Motor *Stepper*

Sebagaimana yang telah dijelaskan sebelumnya, *Driver* motor *stepper* harus dirancang agar dapat digunakan untuk menguatkan sinyal yang diberikan oleh *interface*. Rangkaian *Driver* ini juga dibutuhkan untuk melindungi komputer atau *microprocessor* dari arus balik (*feedback*) yang ditimbulkan oleh motor *stepper*.

Setelah *parallel port* di inisialisasi, maka untuk dapat menggerakkan motor menggunakan data yang dikirimkan harus melalui *driver* motor seperti pada gambar dibawah. Pengaksesan *driver* motor tersebut menggunakan data register dari *parallel port*, dimana melalui program, data untuk menggerakkan motor stepper dikirimkan secara berurutan dengan *delay* yang sudah ditentukan sebelumnya.

UNIVERSITAS Dinamika



Gambar 3.11 Rangkaian *Driver* Motor Stepper

Rangkaian ini harus dapat memberikan arus yang cukup besar sekitar 1A atau lebih, karena motor yang digunakan membutuhkan arus yang cukup besar.

Potongan program yang digunakan untuk mengakses *driver* ini akan disajikan sebagai berikut.

```
procedure TFormUtama.PutarMotor;
var
 i:integer;
begin
   for i:=0 to 9 do
     begin
       PortOut($378,8);sleep(15);
       PortOut($378,4);sleep(15);
       PortOut($378,2);sleep(15);
       PortOut($378,1);sleep(15);
       PortOut($378,0);
     end;
end;
```

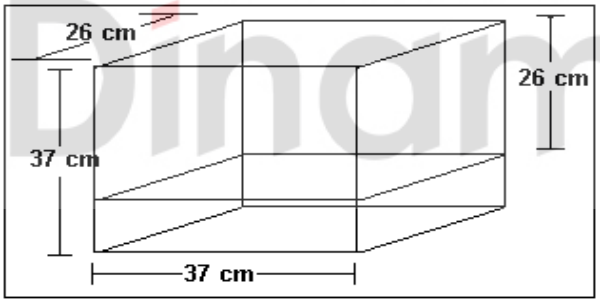
Pada prosedur diatas dikirimkan nilai desimal 8, 4, 2, 1 menuju data *port* untuk dapat menjalankan motor *stepper*, diantara pengiriman nilai tersebut masing-masing diberi *delay* waktu 0.015 detik agar motor dapat berjalan. Pada akhir prosedur dikirimkan nilai desimal 0, nilai ini digunakan agar nilai pada data

port tidak yang bernilai *high.* Begitu seterusnya prosedur ini di *looping* sebanyak 10 kali.

3.1.3. Perancangan Mekanik

Mekanik di desain sedemikian rupa agar dapat sesuai dengan tujuan dan maksud dari penelitian ini. Mekanik yang akan di desain diibaratkan sebagai suatu ruangan yang memiliki pencahayaan yang tidah berubah-ubah intensitasnya, sehingga dalam proses pengambilan citra menggunakan kamera hasil citra yang di dapat memiliki nilai pencahayaan yang tetap.

Berdasarkan tujuan diatas maka dibuatlah sebuah ruangan yang memiliki panjang 37 cm, tinggi 37 cm, dan lebar 26 cm seperti yang terlihat pada gambar 3.8 berikut

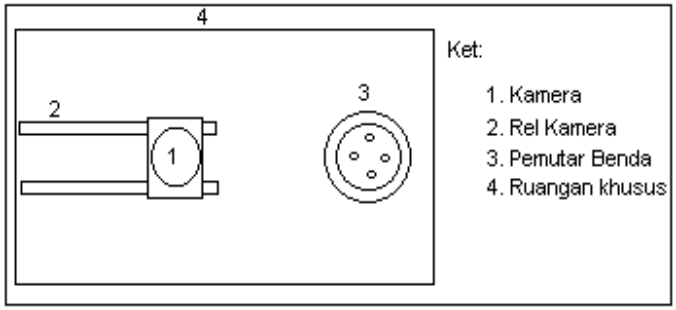


Gambar 3.12 Rancangan Ruangan proses

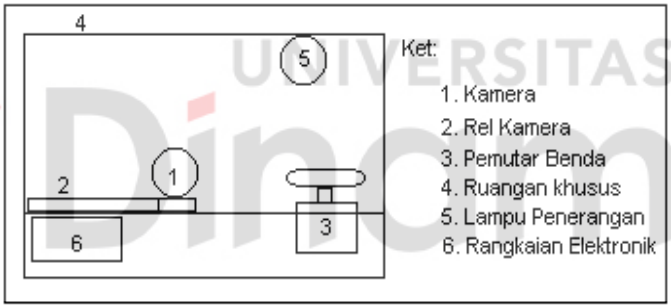
Pada pembuatan fisik mekanik, bahan yang digunakan adalah kayu. Dengan pertimbangan, mudah untuk dipotong, tidak terlalu sulit mendapatkannya, dan yang paling utama harganya lebih murah dari bahan lainnya.

Pergerakan kamera hanya dapat kedepan dan ke belakang secara manual, tergantung dari fokusnya terhadap benda yang ada di atas pemutar. Sedangkan untuk pergerakan dari pemutar benda hanya memiliki satu arah putaran yang

diatur melalui program. Dibagian bawah dari mekanik sengaja disediakan tempat untuk peletakan rangkaian elektronik agar terlihat rapi dan agar rangkaian elektronik dapat terlindungi dengan baik. Berikut ini gambar perancangan mekanik yang akan dibuat.



Gambar 3.13 Bentuk fisik mekanik tampak atas



Gambar 3.14 Bentuk fisik mekanik tampak samping

BAB IV

PENGUJIAN DAN EVALUASI SISTEM

4.1. Prosedur Pengujian

Pada *software* aplikasi terdapat beberapa tahap pengujian yang dapat dilakukan berdasarkan algoritma-algoritma sebelumnya, yaitu:

1. Tahap pengambilan citra sebagai data acuan menggunakan kamera, kemudian citra diolah dengan algoritma *gray scale*, hasil dari proses ini di perbaiki efek ketidak seragaman pencahayaan (*illumination*), dan proses modifikasi kecemerlangan citra (*brightness*).
2. Tahap berikutnya merupakan tahap pengambilan citra sebagai data uji menggunakan kamera, sama halnya dengan tahap sebelumnya, citra diolah dengan algoritma *gray scale*, hasil proses ini juga mengalami perbaikan efek ketidak seragaman pencahayaan dan proses modifikasi kecemerlangan citra (*brightness*) seperti sebelumnya.
3. Selanjutnya merupakan tahap utama dari keseluruhan proses, hasil dari tahap pertama dan tahap kedua kemudian diolah menggunakan algoritma deteksi gerakan (*motion detection*) untuk menghasilkan citra hasil.
4. Pada tahap berikutnya, citra hasil dihitung *error*-nya dengan algoritma penghitungan *error* citra. Pada tahap ini *error* yang dihasilkan dari suatu citra dirubah dalam bentuk prosentase *error*, selain itu juga dilakukan proses perhitungan keterangan pendukung lainnya.
5. Pada tahap akhir, nilai-nilai *error* yang didapat untuk tiap citra hasil dicari rata-ratanya untuk mengetahui hasil akhir proses tersebut.

UNIVERSITAS Dinamika

Pada pengujian *software* ini, untuk mendapatkan hasil dari masing-masing algoritma yang di uji, algoritma yang tidak di uji untuk sementara tidak diaktifkan.

Sedangkan pada pengujian *hardware* dilakukan pengujian terhadap beberapa komponen yang mendukung, diantaranya:

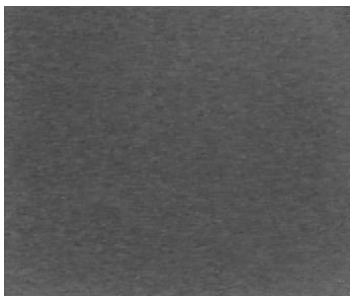
1. Pengujian *interface* melalui operasi *output* dari *parallel port* ke tombol kamera.
2. Selanjutnya dilakukan pengujian *output* dari *parallel port* ke motor *stepper* yang digunakan untuk memutar benda.

4.2. Hasil Pengujian

Citra yang di-*input*-kan adalah citra kemasan kaleng dari hasil proses pengambilan citra menggunakan kamera dengan format *bitmap* (BMP) untuk citra acuan, citra uji, dan citra hasil. Ukuran *file* citra untuk citra acuan dan citra uji adalah sama yaitu 298KB untuk citra *true color*, 101KB citra *gray scale*, 101KB citra perbaikan efek ketidak seragaman pencahayaan, 101KB untuk citra modifikasi kecemerlangan, sedangkan untuk citra hasil deteksi gerakan adalah 101KB.

Berikut ini akan di tampilkan langkah demi langkah gambar dari citra acuan, citra latar, citra uji, serta citra hasil proses deteksi gerakan. Untuk keperluan penulisan buku ini, ukuran gambar sengaja dibuat lebih kecil daripada ukuran sebenarnya.

Pada gambar hasil proses perbaikan citra , citra objek akan nampak lebih gelap daripada sebelumnya. Citra tersebut mengalami proses perbaikan ketidak seragamaan pencahayan yang diproses bersama dengan citra latar seprti dibawah ini.

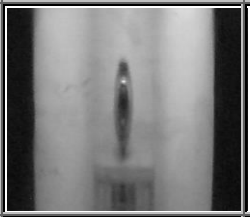
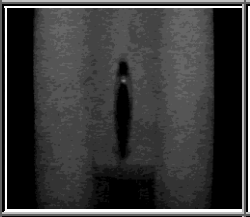
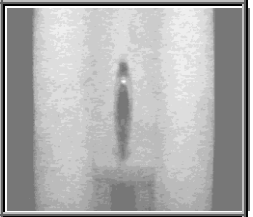
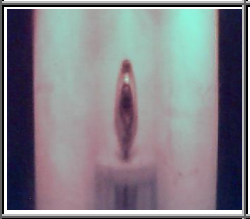
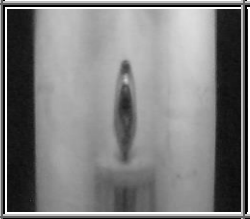
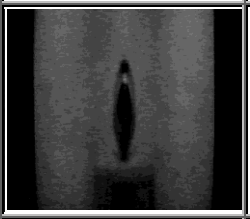
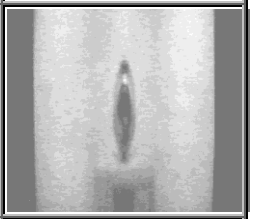
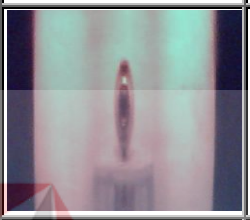
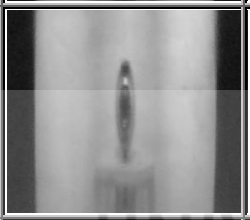
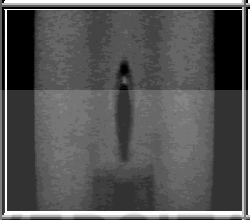
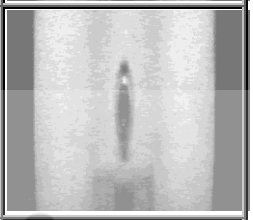
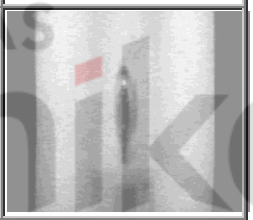
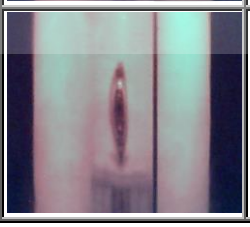
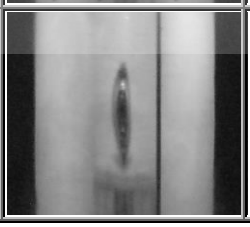
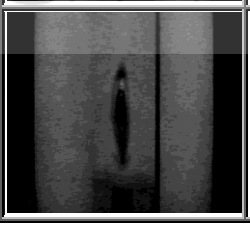
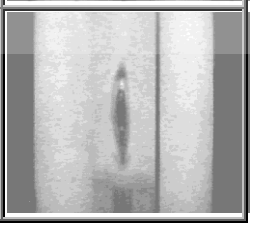


Gambar 4.1 Citra Latar *Gray Scale*

Proses yang terjadi pada citra acu secara berurutan dapat dilihat pada tabel 4.1 dibawah ini. Sedangkan untuk proses yang terjadi pada citra uji secara berurutan dapat dilihat pada tabel 4.2, dan pada tabel 4.3 merupakan tabel gambar proses pendeteksian *error* dari masing masing citra.

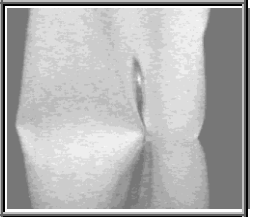
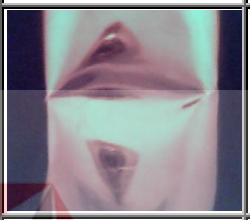
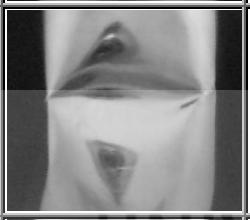
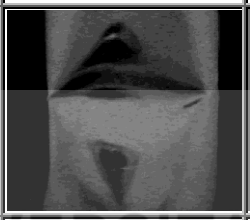
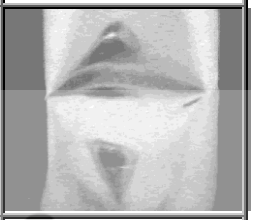
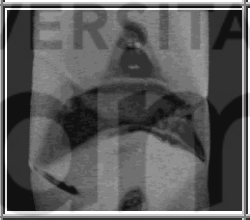
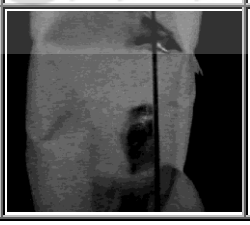
UNIVERSITAS Dinamika

Tabel 4.1 Gambar proses yang terjadi pada citra acuan

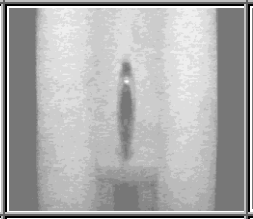
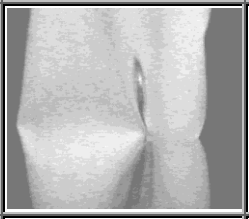
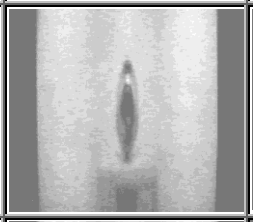
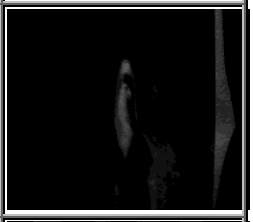
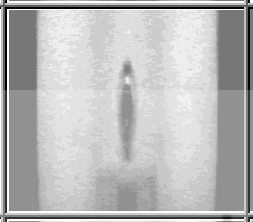
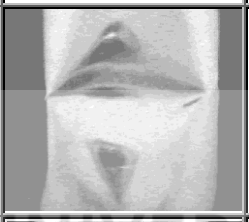
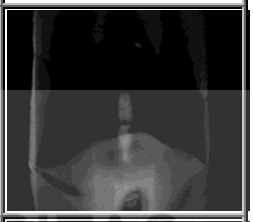
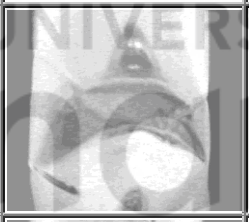
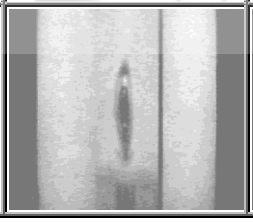
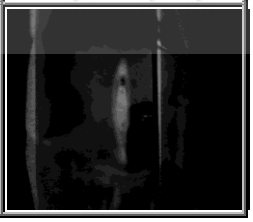
No	Citra True Color	Citra Gray Scale	Perbaikan Citra	Kecemerlangan
1				
2				
3				
4				
5				

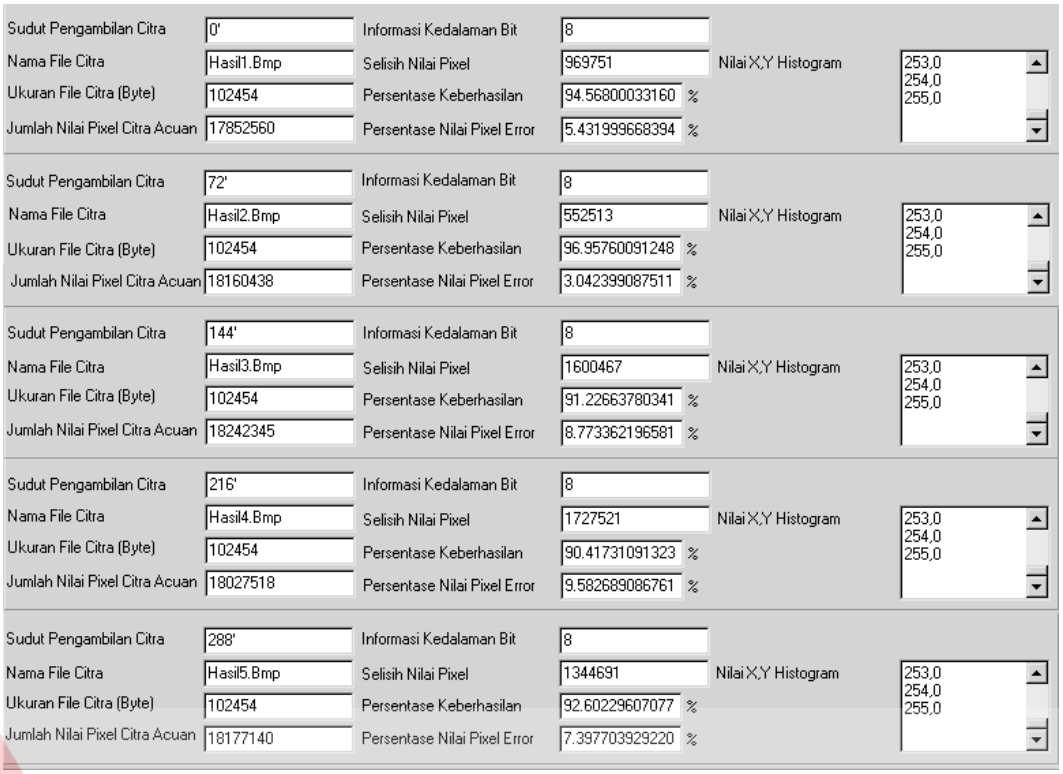
UNIVERSITAS Dinamika

Tabel 4.2 Gambar proses yang terjadi pada citra uji

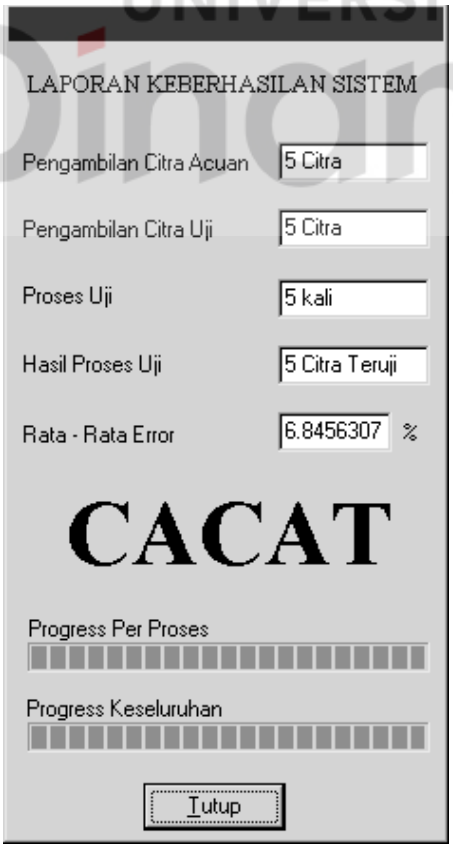
No	Citra True Color	Citra Gray Scale	Perbaikan Citra	Kecemerlangan
1				
2				
3				
4				
5				

Tabel 4.3 Gambar proses pendeteksian error

No	Citra Acu	Citra Uji	Citra Hasil
1			
2			
3			
4			
5			



Gambar 4.2 Penghitungan *error* citra hasil



Gambar 4.3 Penentuan hasil akhir

Pada gambar 4.2 ditampilkan bebrapa keterangan pendukung untuk proses penyeleksian kemasan kaleng, diantaranya adalah : sudut pengambilan citra, nama *file* citra, ukuran *file* citra, jumlah nilai *pixel* citra acuan, informasi kedalaman bit, selisih nilai *pixel*, prosentase keberhasilan, prosentase nilai *pixel error*, dan nilai x,y histogram.

Sudut pengambilan citra adalah keterangan yang menjelaskan tentang sudut yang digunakan oleh kamera dalam mengambil citra, baik citra acuan maupun citra uji. Prinsip dari keterangan ini adalah bahwa untuk melakukan pengambilan gambar sebanyak 5 kali sudut yang di lewati sebesar 360° sehingga untuk pengambilan 1 buah gambar diperlukan selisih sudut sebesar 72°.

Nama *file* dan ukuran *file* citra menerangkan tentang nama dan ukuran dari citra yang disimpan dalam *hardisk*.

Jumlah nilai *pixel* citra acuan adalah total nilai dari tiap-tiap nilai *pixel* yang ada pada citra acuan. Keterangan ini erat kaitannya dengan perhitungan *error* dan prosentase *error* untuk masing-masing citra hasil yang akan dijelaskan kemudian.

Informasi kedalaman bit adalah informasi yang menjelaskan tentang kedalaman bit dari citra hasil deteksi gerakan.

Selisih nilai *pixel* atau jumlah nilai pixel error adalah selisih antara jumlah nilai *pixel* citra acuan dengan jumlah nilai *pixel* citra uji. Atau dengan kata lain nilai selisih dapat diperoleh dengan menghitung total nilai *pixel* citra hasil deteksi gerak. Jumlah selisih nilai *pixel* ini juga akan digunakan untuk menghitung nilai prosentase *error* untuk masing-masing citra hasil.

UNIVERSITAS Dinamika

$$\text{Jumlah Nilai Pixel error} = \text{nilai pixel citra acuan} - \text{nilai pixel citra uji}$$
$$\text{atau} \qquad (4.1)$$
$$\text{Jumlah Nilai Pixel error} = \text{Jumlah nilai pixel citra hasil}$$

Sedangkan prosentase nilai *pixel error*. dengan perhitungan sebagai berikut.

$$\text{Prosentase nilai pixel error} = \frac{\text{Jumlah Nilai Pixel error}}{\text{Jumlah Nilai pixel citra acuan}} x100\% \qquad (4.2)$$

$$\text{Prosentase nilai pixel error} = \frac{969751}{17852560} x100\% = 5{,}431999668394\%$$

Untuk nilai prosentase keberhasilan diperoleh dari mengurangkan nilai 100% dengan nilai prosentase *pixel error*.

Nilai x,y histogram adalah keterangan yang menunjukkan penyebaran nilai warna pada masing-masing citra hasil.

Pada gambar 4.3 ditunjukkan hasil dari proses penentuan hasil akhir dengan perhitungan sebagai berikut.

$$\text{Rata - rata \% error} = \left(\frac{\sum_{0}^{n} \%\text{error}}{n}\right) \qquad (4.3)$$

$$\text{Rata - rata \% error} = \left(\frac{5{,}432\% + 3{,}042\% + 8{,}773\% + 9{,}583\% + 7{,}398\%}{5}\right)$$
$$= 6{,}8456\%$$

Dikarenakan nilai rata-rata prosentase *error* yang diperoleh tersebut lebih besar dari 1.5 maka hasil yang diperoleh adalah CACAT. Nilai batasan ini diperoleh dari rata-rata nilai prosentase *error* sepuluh kali percobaan terhadap kemasan kaleng yang kondisinya masih baik.

Pengukuran waktu eksekusi untuk masing-masing algoritma tidak dilakukan secara *software*, melainkan diukur menggunakan *stopwatch*. Dan dilakukan pada dua spesifikasi yang berbeda yaitu Intel Pentium MMX 200MHz, dengan kapasitas *RAM* 32 MB menggunakan sistem operasi *Windows 98SE* dan Intel Pentium IV 2,4Ghz, dengan kapasitas *RAM* 256 MB menggunakan sistem operasi *Windows Xp*.

Rincian waktu eksekusi algoritma dari *software* yang dibuat hasil pengukurannya menggunakan *stopwatch*, dapat dilihat pada tabel 4.4 berikut ini.

Tabel 4.4 Waktu eksekusi masing-masing algoritma

Algoritma	**Waktu Eksekusi** Pentium MMX (detik)	**Waktu Eksekusi** Pentium IV (detik)
Pengambilan citra Acuan (5 citra), Gray scale, Perbaikan Efek ketidak seragamaan pencahayaan, Modifikasi Kecemerlangan Citra	90 detik	90 detik
Pengambilan citra Uji (5 citra), Gray scale, Perbaikan Efek ketidak seragamaan pencahayaan, Modifikasi Kecemerlangan Citra	91 detik	90 detik
Deteksi Gerakan	10 detik	9 detik
Penghitungan Error	6 detik	1 detik
Penentuan Hasil Akhir	2 detik	0.5 detik

Pengujian untuk *interface* dilakukan bersamaan dengan pengujian motor *stepper*, dengan mengirimkan data ke alamat *parallel port* kemudian dari hasil pengiriman tersebut dapat dilihat hasil pengujian interface dan motor *stepper*. Apabila hasil yang didapatkan sesuai dengan yang di rencanakan maka pengujian dianggap berhasil. Setelah dilakukan beberapa kali percobaan untuk menghasilkan 1 putaran penuh membutuhkan 50 *looping*, sedangkan untuk menghasilkan 1/5 putaran membutuhkan 10 *looping*.

4.3. Analisa

Analisa hasil percobaan pada bagian *software* dilakukan dengan mengolah gambar yang diambil dari proses pengambilan citra menggunakan kamera secara langsung untuk citra sebagai data acuan dan untuk citra sebagai data uji, tetapi tetap dengan prameter pengaturan yang sama dengan proses uji coba.

Percobaan dilakuan satu kali untuk data acuan dan sepuluh kali untuk data uji. Hasil penghitungan waktu (detik) yang diperoleh adalah seperti terlihat pada tabel 4.5 dan 4.6 berikut.

Pada tabel 4.5 dan 4.6 terdapat beberapa singkatan yang memiliki arti sebagai berikut:

- A= Algoritma Pengambilan data uji, *Gray Scale*, dan perbaikan efek ketidak seragaman pencahayaan, dan modifikasi kecemerlangan citra.
- B= Algoritma Deteksi gerakan (*Motion Detection*).
- C= Algoritma Penghitungan *error* citra
- D= Algoritma Penentuan hasil akhir.

UNIVERSITAS Dinamika

Tabel 4.5 Hasil Percobaan *Software* Pentium MMX

Percobaan ke-	A (Detik)	B (Detik)	C (Detik)	D (Detik)	Total (Detik)
1	88	9	5	1	104
2	85	9	5	2	101
3	86	10	5	2	103
4	86	11	5	1	103
5	87	10	5	1	103
6	85	11	5	1	102
7	86	11	5	2	104
8	87	9	5	1	102
9	85	10	5	1	103
10	87	10	5	1	101
Rata-rata	86.2	10	5	1.3	102.6

UNIVERSITAS Dinamika

Tabel 4.6 Hasil Percobaan *Software* Pentium IV

Percobaan ke-	A (Detik)	B (Detik)	C (Detik)	D (Detik)	Total (Detik)
1	80	9	0.9	0.6	90.5
2	82	9	0.5	0.6	92.1
3	82	9	1	0.6	92.6
4	81	9	1	0.5	91.5
5	86	9	0.7	0.7	96.4
6	81	9	0.5	0.6	91.1
7	80	9	0.7	0.6	90.3
8	80	9	0.5	0.6	90.1
9	81	10	0.5	0.9	92.4
10	82	10	1	0.6	93.6
Rata-rata	81.5	9.2	0.63	0.6	92.6

Jadi rata-rata waktu yang dibutuhkan *software* untuk menguji atau mendeteksi satu kemasan kaleng melalui citra digital dengan pengambilan citra sebanyak lima sudut adalah 102,6 detik, atau sekitar 1 menit 42,6 detik. Percobaan yang dilakukan diatas menggunakan komputer Pentium MMX.

Sedangkan rata-rata waktu yang dibutuhkan *software* untuk menguji atau mendeteksi kemasan kaleng menggunakan komputer Pentium IV adalah 92,6 detik, atau sekitar 1 menit 32.6 detik. Perbedaan selisih waktu antara percobaan satu dengan percobaan lainnya pada masing masing spesifikasi adalah akibat dari ada atau tidaknya aktifitas komputer pada saat program dieksekusi.

Selisih yang di dapat dari percobaan dengan menggunakan spesifikasi komputer yang berbeda adalah sebesar 10 detik untuk mendeteksi satu kemasan kaleng.

Pada bagian *interface*, pengujian dilakukan bersamaan dengan pengujian motor *stepper*. Selain dilakukan dengan dengan cara mengirimkan data melalui alamat *interface* kemudian perputaran motor dihitung menggunakan RPM meter, juga dilakukan dengan menggunakan perhitungan untuk mengetahui seberapa besar perbedaan yang terjadi. Seperti yang terlihat pada tabel 4.7 berikut ini.

UNIVERSITAS Dinamika

Tabel 4.7 Hasil uji *interface* dan motor *stepper*

Per Cobaan ke-	Data dikirim (50 step)	Delay (detik)	Putaran /detik (perhitungan)	Putaran /detik (percobaan)	Selisih (Error)
1	1000 0100 0010 0001	0.01	1/2 = 0.500	0.350	0.150
2	1000 0100 0010 0001	0.05	1/10 = 0.100	0.090	0.001
3	1000 0100 0010 0001	0.1	1/20 = 0.050	0.050	0.000
4	1000 0100 0010 0001	0.5	1/100 = 0.010	0.000	0.010
5	1000 0100 0010 0001	1	1/200 = 0.005	0.000	0

Berdasarkan tabel hasil pengujian diatas, untuk dapat mengetahui jumlah putaran tiap detiknya (berdasarkan perhitungan) dapat dicari sebagai berikut

$$\text{delay} = 10 m\text{sec} = 0.01\text{sec}$$
$$1\,\text{looping} = 4 \times \text{delay}$$
$$1\,\text{putaran} = 50\,\text{looping}$$
$$\text{Putaran / Detik} = \frac{1}{50 \times 4 \times 0.01} = \frac{1}{2} = 0.5\,\text{rps}$$

artinya untuk melakukan 1 putaran penuh dibutuhkan waktu selama 2 detik. untuk nilai *delay* yang berbeda dapat digunakan dengan cara yang sama seperti diatas.

Nilai yang diperoleh antara perhitungan rumus dan uji coba secara langsung adalah hampir sama. Untuk selisih antara keduanya dari hasil percobaan yang telah dilakukan menunjukkan nilai selisih terbesar adalah 0.150 detik.

Tabel 4.8 Hasil pengujian sistem secara keseluruhan

No	Sampel	Jumlah Nilai Pixel Error					Prosentase Nilai Error citra					Prosentase rata2 error	Hasil
		I	II	III	IV	V	I	II	III	IV	V		
1	Kaleng uji 1	2809	3741	10934	5886	18320	0.015	0.020	0.059	0.031	0.099	0.045	BAIK
2	Kaleng uji 2	56517	57592	79910	63834	102032	0.302	0.309	0.429	0.342	0.550	0.386	BAIK
3	Kaleng uji 3	372354	221638	275880	249420	412081	1.992	1.189	1.482	1.334	2.220	1.644	CACAT
4	Kaleng uji 4	22175	59381	18709	29961	125505	0.119	0.319	0.101	0.160	0.676	0.275	BAIK
5	Kaleng uji 5	167785	121857	95374	31876	143651	0.898	0.654	0.513	0.171	0.774	0.602	BAIK
6	Kaleng uji 6	29625	35908	51034	37425	106606	0.159	0.193	0.274	0.200	0.574	0.280	BAIK
7	Kaleng uji 7	33385	27634	21218	8116	100248	0.179	0.148	0.114	0.043	0.540	0.205	BAIK
8	Kaleng uji 8	19898	27018	29852	39717	92270	0.106	0.145	0.160	0.213	0.497	0.224	BAIK
9	Kaleng uji 9	16808	54863	13640	9253	98690	0.090	0.294	0.073	0.049	0.532	0.208	BAIK
10	Kaleng uji 10	167785	121857	95374	29961	125505	0.898	0.654	0.513	0.171	0.774	0.601	BAIK

UNIVERSITAS Dinamika

Pada tabel 4.8 diatas disajikan data hasil pengujian sistem yang telah terintegrasi seluruh komponennya, dimaksudkan untuk mengetahui apakah sistem yang telah dibuat dapat bekerja dengan baik atau tidak. Untuk pengujian ini digunakan parameter sesungguhnya yang ada di dalam program yang telah dibuat.

Dari data pada tabel 4.8 diatas jumlah *nilai pixel error* diperoleh dari jumlah seluruh nilai *pixel* dalam citra acuan (angka diperoleh dari perhitungan program) dikurangi jumlah nilai *pixel* pada citra uji (angka diperoleh dari perhitungan program) , atau dengan menghitung jumlah *pixel* citra hasil. Mengacu pada persamaan 4.1 Dapat diperoleh data sebagai berikut:

Jumlah Nilai pixel error = 2089

Sedangkan untuk nilai prosentasenya mengacu pada persamaan 4.2 dengan perhitungan sebagai berikut:

Prosentase Nilai Error = (2809 / 18690666) * 100% = 0.015%

Untuk perhitungan prosentase rata-rata *error* mengacu pada persamaan 4.3 dan diperoleh perhitungan sebagai berikut.

$$\text{Prosentase Rata - rata Error} = \frac{(0.015\% + 0.020\% + 0.059\% + 0.031\% + 0.099\%)}{5}$$

$$= 0.045\%$$

Kemudian, hasil diperoleh dari prosentase rata-rata *error* dibandingkan dengan nilai yang telah ditentukan sebelumnya didalam program.

Berdasarkan hasil pengujian dengan jumlah objek 10 buah dapat di hitung nilai dari keberhsilan sistem ini dalam proses penyeleksian kemasan kaleng yang cacat, yaitu dengan cara sebagai berikut:

$$Keberhasilan\_sistem = \frac{Banyak\_pengujian\_yang\_sesuai}{Banyak\_pengujian} \times 100\% \quad (4.4)$$

$$Keberhasilan\_sistem = \frac{9}{10} \times 100\% = 90\%$$

Jadi, nilai prosentase keberhasilan sistem ini untuk dapat mendeteksi kemasan kaleng yang cacat adalah sebesar 90%. Diperoleh dari jumlah pengujian yang sesuai yaitu banyaknya hasil pengujian yang menyatakan objek sesuai dengan aslinya dibagi dengan banyaknya pengujian kemudian dikalikan dengan 100% untuk mendapatkan nilai prosentasenya.

UNIVERSITAS Dinamika

BAB V

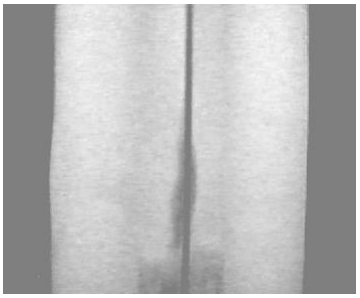
PENUTUP

5.1. Kesimpulan

Dari hasil pengujian dan evaluasi sistem yang telah dilakukan, dapat diambil kesimpulan sebagai berikut :

1. Deteksi gerakan (*Motion Detection*) memiliki kemampuan untuk mendeteksi perubahan bentuk fisik suatu obyek dalam waktu rata-rata 86,2 detik atau sekitar 1 menit 26,2 detik pada spesifikasi pentium MMX 200 MHz dan 81,5 detik atau sekitar 1 menit 21,5 detik pada spesifikasi pentium IV 2,4 MHz. Selisih waktu rata-rata untuk proses deteksi gerakan antara kedua spesifikasi diatas adalah sebesar 4,7 detik.
2. Waktu rata-rata yang dibutuhkan sistem untuk mendeteksi kemasan kaleng adalah sebesar 102,6 detik atau sekitar 1 menit 42,6 detik detik pada spesifikasi pentium MMX 200 MHz dan 92,6 detik atau sekitar 1 menit 32,6 detik pada spesifikasi pentium IV 2,4 MHz. Selisih waktu rata-rata untuk mendeteksi kemasan kaleng antara kedua spesifikasi diatas adalah sebesar 10 detik.
3. Untuk obyek yang perubahan bentuk fisiknya tidak terlalu banyak, pada keluaran yang dihasilkan dari proses pendeteksian kemasan kaleng terkadang dianggap tidak mengalami perubahan. Seperti contoh pada gambar dibawah ini.

UNIVERSITAS Dinamika



Perhatikan sisi kaleng yang sebelah kiri, tampak bahwa kaleng tersebut mengalami sedikit perubahan apabila dibandingkan dengan sisi kanan kaleng sisi kiri nampak tidak lurus. Tetapi pada sistem ini kaleng tersebut dianggap tidak mengalami perubahan.

4. Dari beberapa kali percobaan, *software* yang dibuat untuk mendukung sistem ini dapat mendeteksi kecacatan kemasan kaleng sebesar 90% dari 10 kali percobaan..
5. *Interface* yang berupa *parallel port* merupakan *interface* yang cukup mudah dan cukup baik digunakan untuk percobaan-percobaan lain yang berhubungan dengan pengontrolan.
6. Untuk perubahan ukuran citra yang di proses tergantung dari kamera yang digunakan, untuk *software* yang dibuat tidak ada masalah menyangkut perubahan ukuran citra. Dalam arti bahwa jika ada perubahan ukuran citra maka ukuran antara citra acu dan ukuran citra uji harus selalu sama.
7. Pengambilan gambar yang diproses pada kaleng semakin banyak akan memperkecil rata-rata prosentase error sehingga meningkatkan ketelitian dalam mengambil keputusan cacat tidaknya sebuah kaleng tersebut.

UNIVERSITAS Dinamika

5.2. Saran

Saran-saran yang dapat peneliti berikan untuk pengembangan penelitian sejenis lebih lanjut adalah:

1. Untuk pengembangan selanjutnya, agar pencahayaan menjadi perhatian yang pertama sebelum yang lainnya. Karena dengan terjadinya perbedaan pencahayaan walaupun sedikit akan mempengaruhi hasil yang di peroleh.
2. Agar kaleng yang mengalami perubahan fisik tidak terlalu banyak dapat terdeteksi adalah dengan cara mengatur pencahayaan di ruang tersebut sehingga perubahan tersebut dapat jelas terbaca oleh komputer. Kemudian langkah berikutnya adalah merubah nilai prosentase yang ada pada proses penentuan hasil akhir.
3. Untuk menghasilkan sistem yang lebih presisi, maka diperlukan juga peningkatan kemampuan pada *software* dengan menggabungkan algoritma *artificial inteligent* atau yang lainnya.
4. Untuk memperoleh citra dengan resolusi yang lebih baik dapat menggunakan kamera dengan spesifikasi yang lebih baik.
5. Perputaran motor untuk dapat lebih kuat lagi dapat dikembangkan dengan menggunakan motor jenis yang berbeda. Penggunaan jenis motor yang berbeda dimaksudkan jika ada perubahan ukuran kemasan kaleng yang lebih besar sehingga beban yang dibawa motor lebih berat lagi.
6. Penggunaan *inteface* dapat menggunakan USB (*Universal Serial Bus*) yang memiliki kecepatan yang tinggi sehingga waktu eksekusi dapat lebih cepat.

UNIVERSITAS Dinamika

DAFTAR PUSTAKA

Achmad, B.. & Firdausy, K. 2005. *Teknik Pengolahan Citra Digital Menggunakan Delphi*. Yogyakarta: Ardi Publishing.

Aprea, V. & Grzymkowski, P.2001.*Vertical Plotting Solution*. Cornel University. (http://instruct1.cit.cornell.edu/courses/ee476/FinalProjects/s2001/vp2/index.html, diakses 3 juni 2006).

Bersman, P. 1994. *Controlling The World With Your PC.* United States of America: LLH Technology Publishing.

Bies, L. 2003. *parallelle kabels*. (http://www.lammertbies.nl/ nl_parallel.html, diakses 3 mei 2006).

Chandra, K. I. 2003. *Utility Kamera Digital.* Jakarta: PT. Elex Media Komputindo.

Harries, I. 2003. *IBM-PC Parallel Printer Port Programming Considerations*. (http://www.doc.ic.ac.uk/~ih/doc/par/doc/ParallelPortProgramming.htm).

Hewes, J. 2006, *The Electronics Club*. (http://www.kpsec.freeuk.com, diakse 3 juni 2006).

NoName. 2006. *Web Camera*. (http:// www.wikipedia.com/webcam.htm, diakses 30 mei 2006).

NoName. 2005. *Stepper Motor*. (http:// www.doc.ic.ac.uk/~ih/ doc/stepper/kp4m4/ step.htm, diakses 5 April 2006).

NoName. 2004. *Twain Toolkit For Delphi.* (http:// www.mcm-design.dk/index.php, diakses 5 April 2006).

NoName. 2004. *What is RGB Color.* (http://www.mydesignprimer.com/graphics/0-all_about_graphics_files.shtml, diakses 3 mei 2006).

NoName. 2006. *Digital Image Processing Fundamentals.* (http://show.docjava.com:8086/book/ipij/ch5/html/ch5_t.htm, diakses 3 mei 2006).

Setiawan, C. Y. 2004. *Trik & Tip Delph.* Yogyakarta: Andi Offset .

Wahana Komputer. 2003. *Tip & Trik Pemrograman Delphi 7.0.* Yogyakarta: Andi Offset.

UNIVERSITAS Dinamika

Wahana Komputer. 2005. *Pemanfaatan Kamera Digital & Pengolahan Imagenya*. Yogyakarta: Andi Offset.

Young, I. T. & Gerbrands, J. J. & Vliet, L. J. V. *Fundamentals of Image Processing*. (http:// www.ph.tn.tudelft.nl/Courses/FIP/noframes/fip-.html, diakses 20 Nopember 2005).

UNIVERSITAS Dinamika