# OTOMATISASI PENGHITUNGAN JUMLAH KENDARAAN MENGGUNAKAN METODE PENDETEKSIAN WARNA

**SKRIPSI**

Sebagai salah satu syarat untuk memperoleh
gelar Sarjana Komputer dalam bidang Ilmu Komputer


Oleh :

**KARDILA NUR'AINUN**
**0410960030-96**




**PROGRAM STUDI ILMU KOMPUTER**
**JURUSAN MATEMATIKA**
**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**
**UNIVERSITAS BRAWIJAYA**
**MALANG**

**2008**

UNIVERSITAS BRAWIJAYA

## LEMBAR PENGESAHAN SKRIPSI

**OTOMATISASI PENGHITUNGAN JUMLAH KENDARAAN MENGGUNAKAN METODE PENDETEKSIAN WARNA**

oleh:
**KARDILA NUR'AINUN**
**0410960030-96**

Setelah dipertahankan di depan Majelis Penguji
Pada tanggal 04 Maret 2008
dan dinyatakan memenuhi syarat untuk memperoleh
gelar Sarjana Komputer dalam bidang Ilmu Komputer

| Pembimbing I, | Pembimbing II, |
|---|---|
| Drs. Muh Arif Rahman, M.Kom | Edy Santoso, S.Si., M.Kom |
| NIP. 131 971 481 | NIP. 132 304 307 |

Mengetahui,
Ketua Jurusan Matematika
Fakultas MIPA Universitas Brawijaya

Dr. Agus Suryanto, MSc.
NIP. 131 126 049

UNIVERSITAS BRAWIJAYA

## LEMBAR PERNYATAAN

**Saya yang bertanda tangan di bawah ini :**

| | | |
|---|---|---|
| **Nama** | : | Kardila Nur'ainun |
| **NIM** | : | 0410960030-96 |
| **Jurusan** | : | Matematika |
| **Program Studi** | : | Ilmu Komputer |
| **Penulis Tugas Akhir berjudul** | : | Otomatisasi Penghitungan Jumlah Kendaraan Menggunakan Metode Pendeteksian Warna |

**Dengan ini menyatakan bahwa :**

1. **Isi dari tugas akhir yang saya buat adalah benar-benar karya sendiri dan tidak menjiplak karya orang lain, selain nama-nama yang termaktub di isi dan tertulis di daftar pustaka dalam Tugas Akhir ini.**
2. **Apabila dikemudian hari ternyata Tugas Akhir yang saya tulis terbukti hasil jiplakan, maka saya akan bersedia menanggung segala resiko yang akan saya terima.**

**Demikian pernyataan ini dibuat dengan segala kesadaran.**

**Malang, 04 Maret 2008**
**Yang menyatakan,**

**(Kardila Nur'ainun)**
**NIM. 0410960030-96**

**OTOMATISASI PENGHITUNGAN JUMLAH KENDARAAN MENGGUNAKAN METODE PENDETEKSIAN WARNA**

## ABSTRAK

Otomatisasi penghitungan jumlah kendaraan adalah sebuah teknik yang secara otomatis menyajikan data jumlah kendaraan suatu ruas jalan dalam satuan waktu tertentu. Pendeteksian warna merupakan bagian dari pengolahan citra *digital* yang memanfaatkan elemen citra (*pixel*) sebagai parameter yang diamati.

Dalam penelitian ini elemen citra yang akan diproses berasal dari video arus lalu lintas yang diambil secara *real time*, dan jumlah kendaraan yang terbentuk merupakan hasil pencatatan perubahan warna suatu elemen citra yang diamati (selanjutnya disebut titik sensor) selama video dimainkan. Warna awal titik sensor adalah warna jalan (tanpa kendaraan) yang akan diamati. Setiap titik sensor mencatat perubahan warna sebagai sebuah kendaraan yang sedang melintas.

Untuk mengevaluasi metode yang digunakan dan kualitas hasil penghitungan jumlah kendaraan yang dihasilkan, dilakukan perbandingan jumlah kendaraan hasil sistem dengan jumlah kendaran hasil penghitungan manual dengan parameter resolusi dan toleransi warna (*threshold*) titik sensor.

Hasil ujicoba dan evaluasi menunjukkan bahwa dengan menggunakan 10 variasi titik sensor (2x2 pixel, 4x4 pixel, 6x6 pixel, 8x8 pixel, 10x10 pixel, 12x12 pixel, 14x14 pixel, 16x16 pixel, 18x18 pixel, dan 20x20 pixel) dan 5 variasi nilai *threshold* (10, 20, 30, 40 dan 50), resolusi titik sensor yang baik digunakan adalah lebih besar dari 12 (12x12 pixel), *threshold* yang baik digunakan dalam proses perhitungan sistem adalah 20 serta warna yang baik dijadikan acuan dalam proses perhitungan adalah warna merah.

**AUTOMATIC NUMBER OF VEHICLES CALCULATION USING COLOR DETECTION**

## ABSTRACT

Automatic number of vehicles calculation is a technique which automatically provides number of vehicles on parts of the road at certain times. Color detection is a part of digital image processing that uses picture elements (pixel) as parameter which is analysed.

In this research pixel will be processed provide by video of traffic taked over really time, and number of vehicles created are output of color alteration recording of pixel which is analysed (called points censor). Each points censor records color alteration as a vehicle crossing over.

To evaluating this methode and quality of result of system, is comparing number of vehicles as a result of system and result of manually calculation with different parameter resolution and threshold of them.

The result show that, with 10 variation of points censor (2x2 pixel, 4x4 pixel, 6x6 pixel, 8x8 pixel, 10x10 pixel, 12x12 pixel, 14x14 pixel, 16x16 pixel, 18x18 pixel, and 20x20 pixel) and 5 variation of threshold (10, 20, 30, 40 and 50), good resolution used are more than 12 (12x12 pixel), threshold 20 and standart color are red.

# KATA PENGANTAR

*Alhamdulillahi Robbil alamin*, puji syukur penulis panjatkan kepada Allah SWT atas segala limpahan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan tugas akhir ini sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dalam bidang Ilmu Komputer

Tugas akhir ini bertujuan untuk menerapkan pengolahan citra *digital* dalam penghitungan jumlah kendaraan dengan menggunakan perangkat lunak.

Pada penyusunan tugas akhir ini, penulis ingin mengucapkan terima kasih kepada :

1. Bapak Drs. M. Arif Rahman, M.Kom, selaku pembimbing utama penulisan tugas akhir.
2. Bapak Edy Santoso, S.Si., M.Kom, selaku pembimbing pendamping dalam penulisan tugas akhir.
3. Bapak Wayan Firdaus Mahmudy, S.Si., MT, selaku Ketua Program Studi Ilmu Komputer, Jurusan Matematika, FMIPA Universitas Brawijaya.
4. Bapak Dr. Agus Suryanto, Msc, selaku Ketua Jurusan Matematika, FMIPA Universitas Brawijaya.
5. Ibu Dian Eka Ratnawati, S.Kom., M.Kom, selaku penasehat akademik.
6. Segenap bapak dan ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Program Studi Ilmu Komputer Jurusan Matematika FMIPA Universitas Brawijaya.
7. Segenap staf dan karyawan di Jurusan Matematika FMIPA Universitas Brawijaya yang telah banyak membantu penulis dalam pelaksanaan penyusunan tugas akhir ini.
8. Orang tua penulis atas dukungan materi dan doa restunya kepada Penulis.
9. Rekan-rekan di Program Studi Ilmu Komputer FMIPA Universitas Brawijaya yang telah banyak memberikan bantuannya demi kelancaran pelaksanaan penyusunan tugas akhir ini.
10. Dan semua pihak yang telah membantu dalam penyusunan tugas akhir ini yang tidak dapat penulis sebutkan satu per satu.

Penulis sadari bahwa masih banyak kekurangan dalam laporan ini, oleh karena itu penulis sangat menghargai saran dan kritik yang sifatnya membangun demi perbaikan penulisan dan mutu isi tugas akhir ini untuk kelanjutan penelitian serupa di masa mendatang.

Semoga laporan tugas akhir ini dapat bermanfaat. Amin.

Malang, Maret 2008

Penulis

## DAFTAR ISI

## DAFTAR GAMBAR

## DAFTAR TABEL


UNIVERSITAS BRAWIJAYA

## DAFTAR LAMPIRAN

# BAB I
# PENDAHULUAN

## 1.1. Latar Belakang

Jumlah kendaraan merupakan salah satu parameter yang diperlukan dalam menentukan kebijakan-kebijakan dalam bidang transportasi darat seperti pelebaran badan jalan, memperbaiki prasarana lalu lintas seperti *traffic light* (Warpani, Suwardjoko, 1990). Selama ini informasi jumlah kendaran diperoleh dengan cara survey langsung ke jalan dan menghitung satu persatu kendaraan yang melintas pada suatu titik yang diamati. Dengan metode penghitungan seperti ini membutuhkan sumber daya manusia lebih banyak dan rentang waktu yang cukup lama.

Metode yang bisa diterapkan dalam penghitungan jumlah kendaraan secara otomatis ini adalah pendeteksian warna yang merupakan bagian dari pengolahan citra (*image processing*).

Suatu citra *digital* dapat dianggap sebagai suatu *array* dari bilangan yang dipresentasikan oleh sejumlah bit–bit, dengan indeks baris dan kolomnya menyatakan koordinat sebuah titik pada citra tersebut dan nilai masing – masing elemennya menyatakan intensitas cahaya pada titik tersebut. Titik pada sebuah citra *digital* sering disebut sebagai elemen citra (*picture element*), *pixel* atau *pel* (Murniasih, Euis, 2006).

Pendeteksian warna dalam proses penghitungan jumlah kendaraan ini berdasarkan pada interval warna (*threshold*) dari beberapa kandidat *pixel* yang akan diamati. Beberapa kandidat *pixel* ini selanjutnya disebut titik sensor kendaraan. Warna awal dari titik sensor kendaraan adalah interval warna jalan normal, artinya di jalan tersebut tidak ada kendaraan atau semacamnya. Jika nilai titik sensor kendaraan berada diluar interval warna awal, maka dideteksi adanya sebuah kendaraan yang melintas pada satuan waktu tertentu, dan masing-masing titik sensor kendaraan memasukkan informasi tersebut ke dalam daftar jumlah kendaraan.

Pengolahan citra menggunakan pendeteksian warna sebelumnya pernah dilakukan. Yaitu simulasi pendeteksian kemacetan dalam suatu ruas jalan tertentu. Tahap awal yaitu menyimpan gambar jalan yang sedang kosong untuk disimpan ke *database*. Selanjutnya dilakukan pembacaan gambar jalan yang sama namun, dalam kondisi ada kendaraan. Proses pembacaan *pixel* dilakukan dalam posisi-posisi tertentu. kalau warna *pixel*-nya masih dalam interval warna jalan normal, artinya di titik itu tidak ada kendaraan atau semacamnya, tetapi kalau warna *pixel*-nya berbeda dari interval warna normal, berarti di jalan tersebut ada kendaraan. Kemudian informasi tersebut disimpan dalam

daftar kandidat. Apabila objek kendaraan yang sama tetap berada dalam posisi yang relatif sama dalam rentang waktu tertentu dan tanpa diselingi oleh deteksi *pixel* jalan normal berarti kendaraan dalam kondisi diam. Apabila semua objek diam ditempat dalam waktu tertentu, menandakan kemacetan sedang terjadi (Jimmi, 2007).

Agar penyajian informasi jumlah kendaraan suatu ruas jalan lebih efisien maka perlu dibangun suatu sistem yang secara otomatis bisa menghitung jumlah kendaraan.

Berdasarkan latar belakang yang telah dipaparkan, maka judul yang diambil dalam Tugas Akhir ini adalah ***"Otomatisasi Penghitungan Jumlah Kendaraan Menggunakan Metode Pendeteksian Warna".***

Dengan adanya sistem penghitungan jumlah kendaraan secara otomatis ini diharapkan bisa membantu pihak-pihak yang bergerak di bidang transportasi darat khususnya dan berdaya guna bagi pihak-pihak lain yang membutuhkan informasi jumlah kendaraan sebagai alat bantu.

## 1.2. Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan sebelumnya, maka rumusan masalah yang diambil adalah:

1. Bagaimana merancang sebuah perangkat lunak yang dapat menghitung jumlah kendaraan secara otomatis ?
2. Bagaimana hasil evaluasi dari jumlah kendaraan yang dihasilkan oleh perangkat lunak ini dengan jumlah kendaraan hasil penghitungan manual, dari segi ketepatan informasi yang dihasilkan ?
3. Apakah jumlah kendaraan yang dihasilkan oleh perangkat lunak ini lebih baik jika dibandingkan dengan hasil perhitungan manual ?

## 1.3. Tujuan Penelitian

Tujuan dari tugas akhir ini adalah :

1. Membangun perangkat lunak yang dapat menghitung jumlah kendaraan secara otomatis.
2. Mengevaluasi jumlah kendaraan yang dihasilkan oleh perangkat lunak ini.
3. Membandingkan jumlah kendaraan hasil perangkat lunak dengan jumlah kendaraan hasil penghitungan manual.

## 1.4. Batasan Masalah

Untuk menghindari meluasnya permasalahan, penulis membatasi masalah pada hal berikut:

- Sistem yang akan dibangun hanya untuk menghitung jumlah Kendaraan roda empat di suatu ruas jalan tertentu.
- Inputan sistem berupa video arus lalu lintas MPEG 4 AVI dengan durasi kurang lebih 1 menit.
- *Frame size* video yang digunakan 320 x 240 pixel.
- Rekaman video yang digunakan adalah rekaman pada siang hari.
- Rekaman video yang digunakan dalam kondisi kendaraan berjalan.
- Ruas jalan yang diamati adalah dua lajur satu arah.

## 1.5. Manfaat Penelitian

Manfaat yang dapat diambil dari tugas akhir ini adalah menyediakan perangkat lunak yang mampu menghitung jumlah kendaraan secara otomatis dari sejumlah rekaman arus lalu lintas di suatu ruas jalan.

## 1.6. Sistematika Penulisan

Untuk memberikan gambaran tentang tugas akhir, berikut disajikan garis besar pembahasan dari keseluruhan isi laporan tugas akhir untuk setiap bab.

BAB I : PENDAHULUAN

Bab ini berisi latar belakang penulisan, permasalahan yang ada, batasan masalah, tujuan dan manfaat serta sistematika penulisan tugas akhir.

BAB II : TINJAUAN PUSTAKA

Bab ini berisi penjelasan singkat tentang teori yang melandasi proses perancangan sistem penghitungan jumlah kendaraan.

BAB III : METODOLOGI DAN PERANCANGAN

Bab ini berisi metode-metode yang digunakan dalam menyelesaikan masalah penghitungan jumlah kendaraan.

BAB IV : HASIL DAN PEMBAHASAN

Bab ini berisi tentang penjelasan implementasi sistem dan hasil pengujian yang dilakukan.

BAB V : PENUTUP

Bab ini berisi kesimpulan yang diperoleh dari hasil pengujian dan saran-saran untuk pengembangan lebih lanjut.

# BAB II
# TINJAUAN PUSTAKA

## 2.1 Jumlah Kendaraan

Berdasarkan Manual Kapasitas Jalan Indonesia (MKJI) 1997 fungsi utama dari suatu jalan adalah memberikan pelayanan transportasi sehingga pemakai jalan dapat berkendaraan dengan aman dan nyaman. Jumlah kendaraan merupakan faktor penting dalam perencanaan lalu lintas. Jumlah kendaraan menurut MKJI 1997 adalah banyaknya kendaraan yang melewati suatu titik pengamatan. Kendaraan yang dimaksud diantaranya kendaraan yang tergolong kendaraan ringan (*Light Vehicle*/LV) seperti : opelet, mikro bis, angkot, makro bis, pick-up dan truk kecil ; kendaraan berat (*Heavy Vehicle*/HV) seperti : bis, truk dua as, truk tiga as dan truk kombinasi sesuai dengan sistem klasifikasi Bina Marga; sepeda motor (*Motor Cycle*/MC) seperti : motor, kendaraan roda tiga sesuai dengan klasifikasi Bina Marga; dan kendaraan tak bermotor seperti : sepeda, becak, kereta kuda dan kereta dorong sesuai sistem klasifikasi Bina Marga (Efendi, Indra Rachman, 2007).

## 2.2 Citra *Digital*

Citra adalah suatu representasi, kemiripan, atau imitasi dari suatu obyek atau benda (Achmad, Balza , Kartika Firdausy, 2005).

Citra dapat dikelompokkan menjadi citra tampak (foto keluarga, lukisan, apa yang tampak di layar monitor dan televisi, serta citra optis) dan citra tidak tampak (citra yang direpresentasikan menjadi fungsi matematis).

Citra *digital* adalah citra yang telah ditransformasikan ke dalam bentuk numeris dari nilai tingkat keabuan pada titik-titik elemen citra. Kegiatan untuk mengubah informasi citra *non digital* menjadi *digital* disebut sebagai pencitraan (*imaging*) (Achmad, Balza, Kartika Firdausy, 2005).

Citra *digital* dalam bidang pengolahan citra dapat dianggap sebagai suatu matriks (*f(x,y)*) dari bilangan yang dipresentasikan oleh sejumlah bit–bit, dengan indeks baris dan kolomnya (N) menyatakan koordinat *(x,y)* sebuah titik pada citra tersebut dan nilai masing – masing elemennya menyatakan intensitas cahaya pada titik tersebut. Titik pada sebuah citra *digital* sering disebut sebagai elemen citra (*picture element*), *pixel* atau pel.

Bentuk matriks citra *digital* seperti yang terlihat pada gambar 2.1.

$$f(x,y) = \begin{pmatrix} f(0.0) & f(0.1) & \ldots. & f(0,N-1) \\ f(1,0) & \ldots.. & \ldots.. & f(1,N-1) \\ \ldots\ldots & \ldots.. & \ldots.. & \ldots.. \\ \ldots\ldots & \ldots.. & \ldots.. & \ldots.. \\ f(N-1,0) & \ldots. & \ldots.. & f(N-1,N-1) \end{pmatrix}$$

Gambar 2.1 Bentuk Matriks Citra *Digital*

## 2.3 Komponen Citra *Digital*

Setiap citra *digital* memiliki beberapa karakteristik, antara lain ukuran citra, resolusi, dan format nilainya. Umumnya citra *digital* berbentuk persegi panjang yang memiliki lebar dan tinggi tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya titik atau *pixel*, sehingga ukuran citra selalu bernilai bulat.

Ukuran citra dapat dinyatakan dengan resolusi yang merupakan ukuran banyaknya titik untuk setiap satuan panjang dalam satuan dpi (*dot per inch*). Makin besar resolusi makin banyak titik yang terkandung dalam citra dengan ukuran fisik yang sama, semakin baik kualitas citra.

Format citra *digital* dapat dinyatakan secara bervariasi. Citra yang merepresentasikan informasi 2 keadaan tidak sama dengan citra yang merepresentasikan informasi lebih kompleks sehingga memerlukan lebih banyak keadaan yang mewakilinya. Pada citra *digital* semua informasi tadi disimpan dalam bentuk angka, sedangkan penampilan angka tersebut biasanya dikaitkan dengan warna.

## 2.4 Representasi Citra *Digital*

Komputer dapat mengolah isyarat-isyarat elektronik *digital* yang merupakan kumpulan sinyal biner (bernilai dua: 0 dan 1). Untuk itu, citra *digital* harus mempunyai format tertentu yang sesuai sehingga dapat merepresentasikan obyek pencitraan dalam bentuk kombinasi data biner. Format data citra berhubungan erat dengan warna. Format citra *digital* yang banyak dipakai adalah citra biner, skala keabuan, warna, dan warna berindeks (Achmad, Balza , Kartika Firdausy, 2005).

### 2.4.1 Citra biner (*monocrom*)

Pada citra biner, setiap titik bernilai 0 atau 1, masing-masing merepresentasikan warna tertentu. Contoh yang paling lazim: warna hitam

bernilai 0 dan warna putih bernilai 1. Setiap titik pada citra biner hanya membutuhkan 1 bit, sehingga setiap *byte* dapat menampung informasi 8 titik.

### 2.4.2 Citra skala keabuan (*gray scale*)

Citra skala keabuan memberi kemungkinan warna lebih banyak dari pada citra biner, karena ada nilai-nilai lain di antara nilai minimum (biasanya = 0) dan nilai maksimumnya. Banyaknya kemungkinan nilai dan nilai maksimumnya bergantung pada jumlah bit yang digunakan. Contoh untuk skala keabuan 4 bit, maka jumlah kemungkinan nilainya adalah $2^4 = 16$, dan nilai maksimumnya adalah $2^4 - 1 = 15$; Sedangkan untuk skala keabuan 8 bit, maka jumlah kemungkinan nilainya adalah $2^8 = 256$, dan nilai maksimumnya adalah $2^8 - 1 = 255$.

### 2.4.3 Citra warna (*true color*)

Pada citra warna, setiap titik mempunyai warna yang spesifik yang merupakan kombinasi dari 3 warna dasar, yaitu merah, hijau, dan biru. Format 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 warna merah dan hijau sehingga nilai RGB-nya 255 255 0. Dengan demikian setiap titik pada citra warna membutuhkan data 3 byte.

Jumlah kombinasi warna yang mungkin untuk format citra ini adalah $2^{24}$ atau lebih dari 16 juta warna, dengan demikian bisa dianggap mencakup semua warna yang ada, inilah sebabnya format ini dinamakan *true color*.

### 2.4.4 Citra warna berindeks

Jumlah memori yang dibutuhkan untuk format citra warna *true color* adalah 3 kali jumlah titik yang ada dalam citra yang ditinjau. Di lain pihak, pada kebanyakan kasus, jumlah warna yang ada dalam suatu citra terkadang sangat terbatas (jauh dibawah 16 juta kemungkinan warna yang ada), karena banyaknya warna dalam sebuah citra tidak mungkin melebihi banyaknya titik dalam citra itu sendiri. Untuk kasus tersebut, disediakan format warna berindeks. Pada format ini, informasi setiap titik merupakan indeks dari suatu tabel yang berisi informasi warna yang tersedia, yang disebut palet warna (*color map*).

Jumlah *bit* yang dibutuhkan oleh setiap titik pada citra bergantung pada jumlah warna yang tersedia dalam palet warna. Sebagai contoh, untuk

palet berukuran 16 warna, setiap titik membutuhkan 4 bit, dan untuk palet berukuran 256 warna, setiap titik membutuhkan 8 bit.

## 2.5 Operasi Pengolahan Citra *Digital*

Pengolahan citra pada dasarnya dilakukan dengan cara memodifikasi setiap titik dalam citra tersebut sesuai keperluan. Secara garis besar, modifikasi tersebut dikelompokkan menjadi :

a. Operasi titik

Operasi titik adalah operasi terhadap citra di mana setiap titik diolah secara tak gayut (hanya nilai titik itu sendiri yang dimodifikasi) dengan titik-titik lain. Beberapa operasi pengolahan citra yang termasuk dalam kelompok operasi titik adalah, operasi modifikasi kecemerlangan (*brightness modification*), peningkatan kontras (*kontras enhancement*), negasi (*negation*), dan operasi pengambangan (*thresholding*).

b. Operasi global

Pada operasi global, proses yang dilakukan bergantung pada karakteristik global dari citra yang akan dimodifikasi. Karakteristik global tersebut biasanya berupa sifat statistik dari citra itu sendiri, yang direpresentasikan dengan histogram tingkat keabuan. Sebagian pakar mengelompokkan operasi pengambangan (*thresholding*) sebagai operasi global, karena pemilihan batas ambang dilakukan dengan menginspeksi histogram tingkat keabuan yang diturunkan secara global (mempertimbagkan keseluruhan titik pada citra tersebut).

c. Operasi temporal (berbasis bingkai)

Operasi berbasis bingkai (*frame*) adalah operasi yang melibatkan 2 buah citra atau lebih dan menghasilkan sebuah citra keluaran yang merupakan hasil operasi matematis. Operasi ini dilakukan titik per titik dengan lokasi yang bersesuaian pada citra-citra masukan tersebut. Operasi matematis yang biasa digunakan dalam operasi berbasis bingkai adalah operasi aritmatik (penjumlahan, pengurangan, perkalian dan pembagian) dan operasi logika (*and*, *or* dan *xor*). Beberapa operasi pengolahan citra yang termasuk operasi berbasis bingkai adalah, operasi pengurangan derau (*noise*), penggabungan citra (*image blending*), deteksi gerakan, pendeteksian tepi serta digital *angiography*.

d. Operasi geometri

Pada operasi geometri, proses yang dilakukan terutama dengan memodifikasi koordinat titik dalam suatu citra dengan kemungkinan mengubah nilai skala keabuan dari titik tersebut dengan pendekatan tertentu. Operasi geometri berhubungan erat dengan perubahan geometri citra, yaitu baik ukuran ataupun orientasinya. Operasi geometri diantaranya meliputi pencerminan (*flipping*), rotasi (*rotation*), penskalaan (*scaling/zooming*), dan pembengkokan (*warping*).

e. Operasi banyak titik tetangga

Pada operasi banyak titik tetangga (*neighborhood operation*), proses yang dilakukan dengan memodifikasi nilai keabuan sebuah titik berdasarkan nilai-nilai keabuan dari titik-titik yang ada disekitarnya (bertetangga) yang masing-masing mempunyai bobot tersendiri. Bobot-bobot tersebut nilainya bergantung pada operasi yang akan dilakukan, sedangkan banyaknya titik tetangga dapat bervariasi, misalkan 2x2, 3x3, 3x4, 7x7, dan sebagainya.

Operasi bertetangga pada dasarnya adalah konvolusi antara citra dengan sebuah *filter* (*mask/ kernel*). Nilai-nilai *filter* tersebut merupakan bobot-bobot kontribusi titik tetangga terhadap operasi bertetangga yang dilakukan.

## 2.6 Operasi Pengambilan Nilai Keabuan

Dalam video *digital* setiap *frame* terdiri dari citra *digital* dengan format *true color*. Dimana, tiap *pixel* warna tersusun dari 3 komponen warna dasar (warna RGB). Nilai tiap *pixel* bisa dinyatakan dalam bilangan desimal maupun hexadesimal. Untuk memperoleh nilai dari masing-masing komponen warna (dalam desimal) bisa dinyatakan sebagai berikut :

$$Blue = (\sum_{x}^{K} \sum_{y}^{K} Warna) \ div \ 256^2 \quad \text{.....(2.1)}$$

$$Green = ((\sum_{x}^{K} \sum_{y}^{K} Warna) \bmod 256^2)) \ div \ 256 \quad \text{.....(2.2)}$$

$$\text{Re}\, d = (\sum_{x}^{K} \sum_{y}^{K} Warna) \bmod 256 \quad \text{.....(2.3)}$$

Dengan :
x dan y = koordinat titik sensor
K = resolusi titik sensor
Warna = warna titik sensor

Dalam penghitungan jumlah kendaraan secara otomatis ini, operasi titik yang digunakan untuk masing-masing komponen warna adalah sebagai berikut :

$$WarnaT = \left| (\sum_{x}^{K} \sum_{y}^{K} W_{rt} - W_0) \right| \quad \text{.....(2.4)}$$

Dengan :
x dan y = koordiant titik sensor
$W_{rt}$ (W *real time*) = perubahan tiap komponen warna saat video dimainkan
$W_0$ (W awal) = komponen warna jalan tanpa kendaraan

## 2.7 Video *Digital*

Video *digital* merupakan gabungan citra *digital* yang dibaca berurutan dalam suatu waktu dengan kecepatan tertentu. citra *digital* yang digabung tersebut dinamakan *frame* dan kecepatan pembacaan gambar disebut dengan *frame rate*, dengan satuan fps (*frame per second*). Karena dimainkan dalam kecepatan yang tinggi maka tercipta ilusi gerak yang halus, semakin besar nilai *frame rate* maka akan semakin halus pergerakan yang ditampilkan (Gora, Winastwan, 2006).

Adapun elemen-elemen dasar video *digital* adalah sebagai berikut:

1. *Frame rate*

*Frame rate* adalah jumlah gambar yang terlihat setiap detiknya. Diperlukan *frame rate* minimal sebesar 10 fps untuk menghasilkan gambar pergerakan yang halus (Gora, Winastwan, 2006). *Film-film* yang kita lihat di gedung bioskop adalah *film* yang diproyeksikan dengan *frame rate* sebesar 24 fps, sedangkan video yang kita lihat di televisi kira-kira memiliki *frame rate* sebesar 30 fps (tepatnya 29.97 fps) untuk negara yang memakai format standar NTSC (*National Television Standards Comitte*) yaitu Amerika Serikat, Jepang, Kanada, Meksiko dan Korea. Untuk negara Indonesia, Inggris, Australia, Eropa dan China format video standar yang digunakan adalah format PAL (*Phase Alternate Line*) dengan *frame rate* sebesar 25 fps. Sedangkan negara Perancis, Timur Tengah dan Afrika menggunakan format video standar SECAM

(*Sequential Couleur Avec Memoire*) dengan *frame rate* sebesar 25 fps. Format standar *frame rate* negara-negara di dunia dapat dilihat pada tabel 2.1

Tabel 2.1 Format Standar Frame Rate Negara-Negara di Dunia

| Format Standar | Negara | Frame Rate |
|---|---|---|
| NTSC | Amerika Serikat, Jepang, Kanada, Meksiko dan Korea | 29,97 |
| PAL | Indonesia, Inggris, Australia, Eropa dan China | 25 |
| SECAM | Perancis, Timur Tengah dan Afrika | 25 |

2. *Frame size*

*Frame size* adalah lebar dan tinggi *frame video*, yang menggunakan satuan *pixel*, misal *video* dengan *frame size* 640x480 pixel. Dalam dunia video *digital*, *frame size* disebut juga resolusi. Semakin tinggi resolusi gambar maka semakin besar pula informasi yang dimuat, berarti akan semakin besar pula kebutuhan memori untuk membaca informasi tersebut (Gora, Winastwan, 2006).

3. *Pixel aspect ratio*

*Pixel aspect ratio* menyatakan rasio perbandingan lebar dengan tinggi dari sebuah *pixel* dalam sebuah citra. *Frame aspect ratio* menggambarkan perbandingan lebar dengan tinggi pada dimensi *frame* dari sebuah citra. Sebagai contoh, D1 NTSC memiliki *pixel aspect ratio* 0.9 (0.9 lebar dari 1 unit tinggi) dan memiliki pula *frame aspect ratio* 4:3 (4 unit lebar dari 3 unit tinggi).

4. *Bith dept*

*Bith dept* menyatakan banyaknya *bit* (satuan terkecil penyimpanan informasi dalam dunia komputer) yang tersimpan untuk mendeskripsikan warna suatu *pixel*. Sebuah citra yang memiliki 8 bit per *pixel* dapat menampilkan 256 warna, sedangkan gambar dengan 24 bit dapat menampilkan warna sebanyak 16 juta warna. Komputer (PC) menggunakan 24 bit RGB (*Red Green Blue*) sedang sinyal video menggunakan standar 16 bit YUV sehingga memiliki jangkauan warna yang terbatas.

5. *Bit rate*

*Bit rate* disebut juga dengan nama *date rate*. *Bit rate* menentukan jumlah data yang ditampilkan saat *video* dimainkan. *Date rate* ini dinyatakan dalam satuan bps (*bit per second*). *Date rate* berkaitan erat dengan pemakaian dan pemilihan *codec* (metode kompresi video). Misalnya MPEG-2 yang digunakan dalam format DVD dapat menggunakan *bit rate* maksimum 9800 kbps atau 9,8 Mbps (Gora, Winastwan, 2006).

## 2.8 Kompresi Video

Ketika kita memakai sebuah digital *camcorder* dan *video capture card* lalu hasil video tersebut kita *capture* atau *digitized* maka berarti kita telah melakukan proses kompresi *video*. Proses kompresi sangat penting mengingat penanganan data video *digital* membutuhkan tempat penyimpanan data (*harddisk*) yang sangat besar. Jika sebuah *frame* membutuhkan 1 Mb, dan apabila menggunakan standar format NTSC (30 fps), maka dibutuhkan ruang penyimpanan sebesar 30 Mb untuk tiap detiknya dan sebesar 1,5 Gb untuk tiap menitnya, apabila video *digital* tidak mengalami kompresi.

Kompresi digunakan untuk mereduksi atau mengurangi besarnya data video. Untuk mengatur kompresi digunakan *codec* (*compressor-decompressor*). *Codec* adalah program yang digunakan untuk menganalisa video dan membuang data yang tidak diperlukan. Misalnya apabila di dalam video terdapat obyek yang muncul terus menerus maka informasi yang sama dapat diulang untuk memperkecil ukuran *file*.

a. Kompresi DV25

DV25 merupakan format standar kompresi video *digital* yang dipakai pada banyak *camcorder* pada kelas *consumer* (pengguna rumahan dan *hobbies*) maupun *prosumer* (*professional* dan *broadcasting*). DV25 menggunakan rasio kompresi 5:1, dengan *bit rate* data video sebesar 25 Mbps. Untuk durasi 1 jam, video dengan kompresi DV25 membutuhkan media penyimpanan sebesar 13 Gb.

b. Kompresi MPEG-1

MPEG (*Motion Pictures Expert Group*), sebuah organisasi para professional dalam bidang *film* dan video yang menentukan peraturan standar industri. Angka 1 menyatakan versi dari standar ini (versi pertama). Kompresi ini memiliki ukuran *frame size* 352x240 pixel.

Kompresi ini masih dipakai sebagai acuan standar untuk VCD, CD-ROM dan *web* video.

c. Kompresi MPEG-2

MPEG-2 (versi 2) merupakan format standar kompresi *video digital* yang ideal untuk DVD yang memiliki *data rate* sebesar 9,8 Mbps. Kompresi MPEG-2 ditujukan untuk distribusi *video* bukan untuk konsumsi *editing* video. Jadi, dalam proses *editing* video kita menggunakan kompresi DV25, untuk kemudian dikompresi menggunakan MPEG-2 untuk menghasilkan media DVD.

d. Komresi MPEG-4

MPEG-4 (versi 4) merupakan format standar kompresi video *digital* yang memungkinkan integrasi produksi, distribusi, maupun konten untuk 3 bidang, yakni *digital televition*, interactive *graphics application*, dan *interactive multimedia*, termasuk konten untuk *World Wide Web* (WWW). Jika pada format MPEG-2 tidak dimungkinkan membuat konten interaktif untuk ditanamkan dalam DVD atau *broadcast* interaktif, maka hal sebaliknya terjadi pada MPEG-4. Faktor inilah yang menarik perhatian studio-studio *film* besar. Hal lain yang membedakan MPEG-4 adalah efisiensinya. Dalam MPEG-4, meskipun video dan audio di-*coding* dengan teknik kompresi yang canggih, namun gambar, teks, dan obyek sintetik memiliki coding tersendiri, dan bukan memampatkannya dalam *pixel* atau *waveform* (Audio Video, 2004).

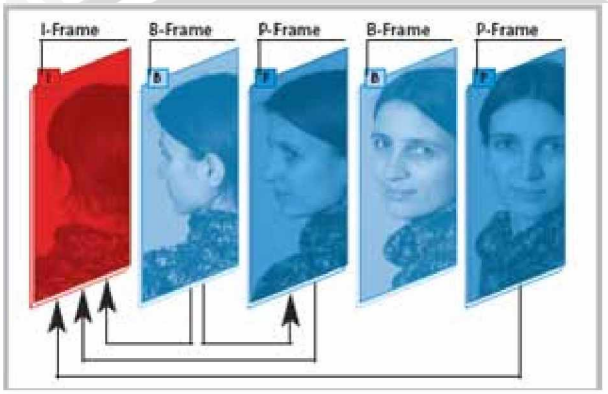
## 2.9 Pembacaan Video *Digital*

Video terdiri atas gambar-gambar tunggal (*frame*), pada standar NTCS jumlahnya 30 gambar/detik. Langkah pertama dalam proses pembacaan *file* dalam video adalah mengkonversi setiap gambar, artinya, mengubah bitmap menjadi JPEG.

Dalam hal ini *encoder* membagi sebuah *frame* ke dalam blok-blok berukuran 8 x 8 pixel, masing-masing dengan nilai kecerahan dan warna yang sesuai. Dalam setiap blok ini nilai kecerahan dan warna tersebut diubah oleh *Discrete Cosinus Transformation* (DCT) menjadi frekuensi-frekuensi sesuai alurnya.

Setelah *encoder* membagi *frame* ke dalam blok-blok, selanjutnya *encoder* menyimpan (merangkum) 12 – 15 *frames* ke dalam sebuah *group of Pictures* (GOP). Hanya *frame* pertama-*Intra-Frame* (*I-frame*) yang berisi semua informasi gambar seperti dalam JPEG (Sun, Hui Fang, dkk,

2005). Selebihnya merupakan *Predicted-Framed* (*P-frame*) dan *Bidirectional-Frame* (*B-frame*) yang menunjukkan perbedaan dibandingkan *frame* sebelumnya dan berikutnya. *P-frame* dan *B-frame* lebih hemat tempat (gambar 2.2).




Gambar 2.2 GOP dengan *I-frame* yang Berisi Informasi Gambar

Perubahan dihitung *encoder* melalui *Motion Estimation*, dengan membagi sebuah *frame* ke dalam *macroblock* berukuran 16x16 pixel sampai 4x4 pixel, selanjutnya masing-masing blok diproses. Mengingat ukuran *macroblock* ini relatif jauh lebih kecil dibanding ukuran satu *frame*, maka kemungkinan terdapat *macroblock* yang sama antara *frame* yang satu dan *frame* yang lain semakin besar.

Penggunaan *macroblock* akan membuat penyimpanan informasi *pixel* menjadi lebih kecil, karena tidak diperlukannya penyimpanan seluruh *pixel* dalam setiap *frame*, karena bila ditemukan *macroblock* yang sama (pada *frame* sebelumnya) maka informasi *pixel* dalam *macroblock* tersebut bisa digunakan kembali (mengingat isinya sama).

Proses pencarian *macroblock* ini dinamakan *motion search*. Ketika *encoder* memproses sebuah *frame* pertama, langkah selanjutnya adalah, mencari *macroblock* yang sama pada *frame* selanjutnya, bila ditemukan *encoder* akan mencatat posisi *macroblock* yang baru dan menyimpannya dalam bentuk vektor pergerakan (perpindahan *macroblock*), perpindahan *macroblock* ini terlihat sebagai pergerakan gambar, sehingga dinamakan *motion search* (gambar 2.3).

Jadi, pada *frame* awal *encoder* menyimpan seluruh informasi *pixel* dari seluruh *macroblock*, tetapi untuk *frame* selanjutnya *encoder* tidak

menyimpan seluruh informasi *pixel* dari *frame* kedua, melainkan *encoder* hanya menyimpan kumpulan vektor (*motion vector* / MV). Kumpulan *vector* ini yang digunakan *decoder* untuk menggambar *frame* kedua, yaitu dengan menggunakan *macroblock* pada *frame* pertama sesuai informasi dari *vector* tersebut.


Gambar 2.3 Kumpulan *Motion* Vektor

Gambar 2.3 memperlihatkan bagaimana *encoder* membagi *frame* dalam *grid* yang elemennya disebut *macroblock* (titik pada gambar adalah pusat *macroblock*), kemudian dari pusat *macroblock* ini terdapat banyak tanda panah (*vector*) pada area muka dan panah umumnya lebih pendek pada *background*, hal ini menggambarkan objek muka yang bergerak dan daerah *background* relatif diam (*macroblock* yang sama tetap berada ditempatnya semula), kemudian perhatikan bagian tangan dimana tidak terdapat *vector*, hal ini dikarenakan *encoder* tidak dapat menemukan *macroblock* yang sama untuk daerah tangan sehingga tidak terdeteksi adanya pergerakan dari *frame* sebelumnya, akibatnya *encoder* harus memasukkan informasi *pixel* bukannya *vector*.


UNIVERSITAS BRAWIJAYA

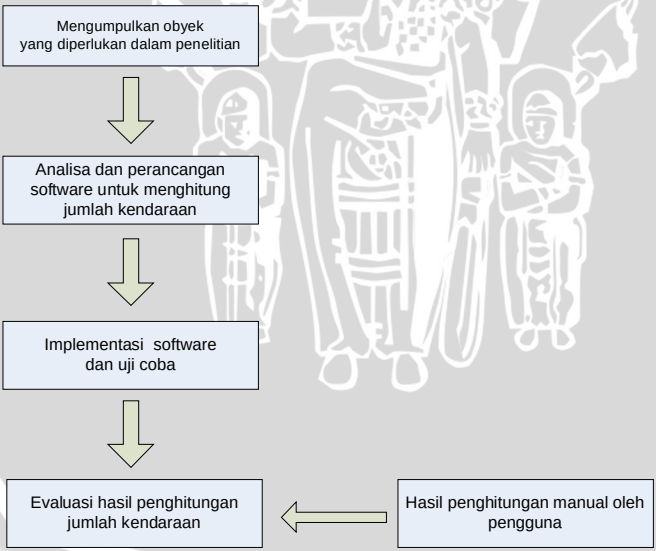
# BAB III
# METODOLOGI DAN PERANCANGAN

Pada bab metodologi dan perancangan ini akan dibahas metode, rancangan yang digunakan dan langkah-langkah yang dilakukan dalam penelitian otomatisasi penghitungan jumlah kendaraan menggunakan metode pendeteksian warna.

Penelitian dilakukan dengan tahapan – tahapan berikut ini :

1. Mengumpulkan obyek yang diperlukan dalam penelitian.
2. Menganalisis dan melakukan perancangan sistem untuk menghitung jumlah kendaraan.
3. Membuat perangkat lunak berdasarkan analisis dan perancangan yang telah dilakukan.
4. Melakukan uji coba perangkat lunak menggunakan rekaman video arus lalu lintas suatu ruas jalan tertentu.
5. Mengevaluasi hasil analisa yang dilakukan oleh sisitem, berdasarkan penghitungan jumlah kendaraan manual (menghitung satu persatu kendaraan yang melintas).

Tahapan-tahapan dalam penelitian ini dapat digambarkan seperti pada gambat 3.1 berikut :




Gambar 3.1 Diagram Alir Penelitian

## 4.1 Analisa Sistem

Pada subbab ini akan dibahas berbagai hasil analisa terhadap sistem dan elemen-elemen yang terkait, seperti pengguna dan semua yang diperlukan dalam proses penghitungan jumlah kendaraan.

### 1 5.2 Analisa data

Pada penelitian ini obyek yang diambil sebagai obyek penelitian diperoleh dari rekaman video arus lalu lintas yang diambil pada bulan Januari 2008, dengan format MPEG 4 dengan *frame size* 320 x 240 pixel dan *frame rate* 30 fps. Sebelum video di-*input*-kan ke dalam sistem terlebih dahulu dikonversi ke format MPEG 4 AVI.

### 2 5.2 Deskripsi umum sistem

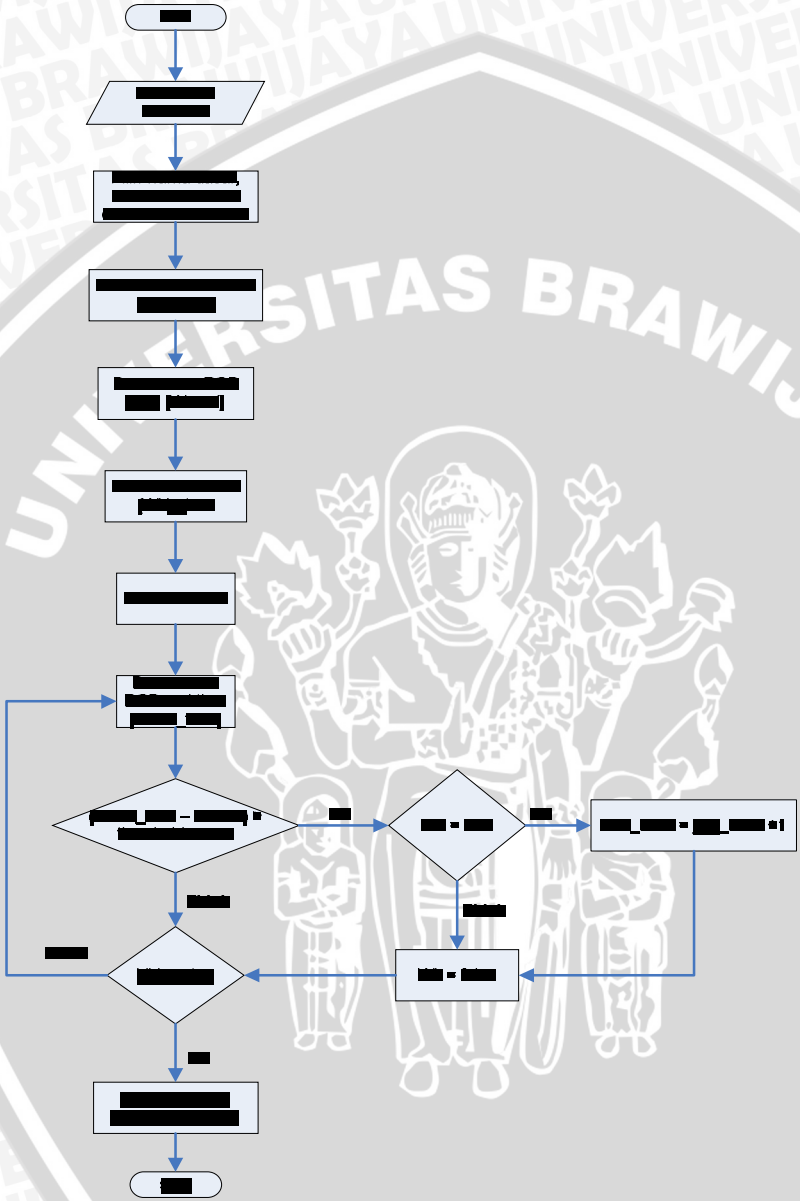
Penghitung jumlah kendaraan secara otomatis yang akan dibuat merupakan sistem yang membaca sebuah video arus lalu lintas, kemudian pengguna meletakkan titik sensor pada posisi jalan yang pasti dilewati kendaraan dan secara otomatis menghasilkan (*generate*) jumlah kendaraan. Jumlah kendaraan yang dihasilkan akan bersifat indikatif (dijadikan sebuah referensi, yang membantu pengguna untuk mengetahui jumlah kendaraan dari video tanpa menghitung satu persatu kendaraan yang melintas). Metode yang digunakan pada sistem ini adalah pendeteksian warna dengan operasi titik dan operasi global yang ada dalam pengolahan citra *digital*.

Jumlah kendaraan dibuat berdasarkan pada perbedaan interval (*thresholding*) warna jalan tanpa kendaraan pada tiga elemen dasar warna RGB dan warna *gray scale* . Ketika sistem akan menghitung jumlah kendaraan dari sebuah video yang dimainkan dalam durasi tertentu, proses yang dilakukan adalah:

- Pengguna meng-*input*-kan video arus lalu lintas ke dalam sistem.
- Pengguna memilih komponen warna yang digunakan dalam pengamatan, resolusi titik sensor, serta *threshold* warna yanng akan digunakan.
- Kemudian pengguna meletakkan titik sensor pada video sebanyak jumlah sensor yang digunakan (4 buah titik) pada posisi jalan tanpa kendaraan sesuai dengan arah kendaraan bergerak.
- Kemudian oleh sistem, warna (RGB) tiap *pixel* titik sensor kendaraan dibaca (persamaan 2.1, 2.2 dan 2.3). Warna awal titik sensor adalah warna jalan tanpa kendaraan yang ditangkap sistem pertama kali.

- Selama video dimainkan sistem akan mencatat selisih tiap komponen warna antara warna titik sensor *real time* (perubahan warna yang ditangkap selama video dimainkan) dengan warna awal titik sensor. Jika nilai selisih masing-masing komponen warna (merah, hijau, biru atau *gray scale*) lebih dari nilai *threshold* yang digunakan, maka sistem akan mendeteksi ada sebuah unit kendaraan sedang melintas.
- Kemudian masing-masing titik sensor akan mengakumulasi jumlah kendaraan. ($Sensor_i = \sum_{frame=1}^{Nframe} total\_kendaraan$).

Dari uraian di atas, dapat digambarkan tahap-tahap penghitungan jumlah kendaraan menggunakan pendeteksian warna seperti *flowchart* pada gambar 3.2 berikut.



Gambar 3.2 *Flowchart* Tahap-Tahap Penghitungan Jumlah Kendaraan Menggunakan Pendeteksian Warna

### 3 5.2 Batasan sistem

Batasan dari sistem yang akan dikembangkan adalah :

1. Dalam peletakan titik sensor kendaraan, pengguna harus bisa mempertimbangkan posisi dimana kendaraan akan melintas.
2. Diasumsikan jumlah titik sensor yang ideal digunakan sebanyak 4 buah, dengan masing-masing lajur mempunyai 2 titik sensor.
3. Diasumsikan susunan titik sensor yang ideal adalah sejajar.
4. Kecepatan titik sensor dalam membaca perubahan warna adalah 100 milisecond (setiap 3 frame).

## 4.1 Perancangan Proses

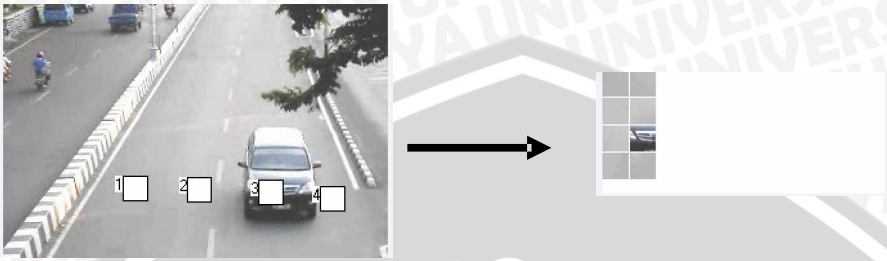
Dalam proses penghitungan jumlah kendaraan ini, video arus lalu lintas yang akan digunakan diperoleh dari rekaman video arus lalu lintas yang diambil pada bulan Januari 2008. Proses-proses yang akan dilakukan oleh sistem secara terperinci akan dijelaskan disini.

Proses penghitungan jumlah kendaraan dalam melewati proses penentuan titik sensor kendaraan hingga menghasilkan informasi jumlah kendaraan terdiri dari 3 tahap yaitu proses *capture* video sebesar resolusi titik sensor, pembacaan warna titik sensor dan proses penghitungan jumlah kendaraan.

### 3.2.1 *Capture* Video

Video adalah kumpulan citra yang dibaca berurutan dalam suatu waktu dengan kecepatan tertentu. Banyaknya citra yang dimainkan dalam sebuah video dinyatakan dalam ukuran *frame*. Untuk dapat meng-*capture* video sebesar resolusi titik sensor, maka *frame* pertama dari video harus ditampilkan sebelum video dimainkan.

Kemudian setelah pengguna meletakkan titik sensor pada *frame* pertama, hasil *capture* ditampilakn pada tempat sementara, agar hasilnya mudah untuk dianalisa. Representasi proses *capture* video dapat dilihat pada gambar 3.3.




Gambar 3.3 Hasil *Capture* Video Sebesar Resolusi Titik Sensor

### 1 5.2 Pembacaan warna

Video adalah kumpulan citra yang dibaca berurutan dalam suatu waktu dengan kecepatan tertentu. Citra dibangun dari sekumpulan *pixel* yang memiliki nilai tertentu. Kumpulan citra dalam video arus lalu lintas ini merupakan citra berwarna (*true color*), dimana setiap *pixel*-nya merupakan kombinasi dari 3 warna dasar, yaitu : merah (*Red*), hijau (*Green*), dan biru (*Blue*) atau sering disebut citra RGB.

Proses pembacaan warna ini dilakukan dengan pencuplikan citra warna sebesar resolusi titik sensor yang digunakan pada posisi (koordinat) tertentu. Nilai warna titik sensor akan diuraikan menjadi 4 bagian bagian yaitu berdasarkan warna merah, berdasarkan warna hijau, berdasarkan warna biru dan berdasarkan warna rata-rata ketiganya (*gray scale*). Representasi citra pada pembacaan warna titik sensor kendaraan ditunjukkan pada gambar 3.4.


Gambar 3.4 Representasi Citra pada Pembacaan Warna Titik Sensor Kendaraan

Misalkan pengguna menggunakan 4 buah titik sensor dengan resolusi 10x10 pixel, maka untuk masing-masing titik sensor terdiri dari 100 pixel. Jika nilai warna titik sensor diuraikan menjadi 4 bagian tiap *pixel*-nya maka prosesnya diperoleh dari persamaan (2.1, 2.2 dan 2.3) adalah sebagai berikut (tabel 3.1 dan 3.2) :

Tabel 3.1 Warna Awal Titik Sensor

| Titik Sensor | Warna Awal ($W_0$) | Red | Green | Blue | Grayscale |
|---|---|---|---|---|---|
| $S_1$ | 11579568 | 176 | 176 | 176 | 176 |
| $S_2$ | 12435393 | 193 | 191 | 189 | 191 |
| $S_3$ | 12961221 | 197 | 197 | 197 | 197 |
| $S_4$ | 12500670 | 190 | 190 | 190 | 190 |

Tabel 3.2 Warna *Real Time* Titik Sensor

| Titik Sensor | Warna Real time ($W_{rt}$) | Red | Green | Blue | Grayscale |
|---|---|---|---|---|---|
| $S_1$ | 11316396 | 172 | 172 | 172 | 172 |
| $S_2$ | 12303807 | 191 | 189 | 187 | 189 |
| $S_3$ | 12556133 | 101 | 151 | 191 | 147 |
| $S_4$ | 11974326 | 8 | 8 | 8 | 8 |

### 3.2.3 Penghitungan jumlah kendaraan

Kemampuan tiap titik sensor kendaraan dalam mendeteksi perubahan warna harus mempunyai waktu (*timing*) yang sama. Tujuan dari pendefinisian waktu ini adalah untuk memperkecil kemungkinan titik sensor kendaraan kehilangan informasi perubahan nilai warna selama *video* dimainkan. Waktu ini diperoleh dari banyaknya *frame* yang dimainkan dalam video. Jika video mempunyai *frame rate* 30 fps, maka untuk membaca warna tiap 3 frame dibutuhkan waktu 100 milisecond.

Sama seperti pada proses pembacaan warna, proses penghitungan jumlah kendaraan juga didasarkan pada 4 bagian yaitu berdasarkan warna merah, berdasarkan warna hijau, berdasarkan warna biru dan berdasarkan warna rata-rata ketiganya (*gray scale*). Untuk masing-masing bagian dilakukan operasi pengurangan antara warna *real time* dengan warna awal ($W_{real\_time}$ - $W_{awal}$). Ketepatan jumlah kendaraan yang dihasilkan oleh sistem akan dibandingkan dengan jumlah kendaraan hasil perhitungan manual, dengan toleransi warna (*threshold*) yang bervariasi.

Dalam proses penghitungan jumlah kendaraan ini diasumsikan jumlah titik sensor yang ideal digunakan adalah 4 buah, dimana masing-masing lajur terdiri dari 2 buah titik sensor. Jumlah kendaraan akhir (JKA) diperoleh dengan menjumlahkan list kendaraan maksimum yang tercatat tiap lajur (sensor 1 dan sensor 2 pada lajur kanan, sensor 3 dan sensor 4 pada lajur kiri).

Dari persamaan 2.4 masing-masing komponen warna dapat dinyatakan sebagai berikut :

$$\mathrm{Re}\,dT = \left| \sum_{x}^{K} \sum_{y}^{K} (\mathrm{Re}\,d_{rt} - \mathrm{Re}\,d_{0}) \right| \quad \text{.....(3.1)}$$

$$GreenT = \left| \sum_{x}^{K} \sum_{y}^{K} (Green_{rt} - Green_{0}) \right| \quad \text{.....(3.2)}$$

$$BlueT = \left| \sum_{x}^{K} \sum_{y}^{K} (Blue_{rt} - Blue_{0}) \right| \quad \text{.....(3.3)}$$

$$GrayScale = \left| \frac{\sum_{x}^{K} \sum_{y}^{K} (\mathrm{Re}\,dT + GreenT + BlueT)}{3} \right| \quad \text{.....(3.4)}$$

Misal dari tabel 3.1 dan 3.2 proses pada sensor 1 ($S_1$) adalah sebagai berikut :

$Red_0$ = 172; $Green_0$ = 172; $Blue_0$ = 172
Red_rt = 176; Green_rt = 176; Blue_rt = 176

RedT = |Red_rt - $Red_0$|
= |172 – 176|
= 4
GreenT = |Green_rt - $Green_0$|
= |172 – 176|
= 4

BlueT = |Blue_rt - $Blue_0$|
= |172 – 176|
= 4

Grayscale = | (RedT + GreenT + BlueT) / 3 |
= | (4 + 4 + 4)/3|
= 4

Hasil dari perhitungan untuk masing-masing titik sensor seperti yang dilihat pada tabel 3.3.

Tabel 3.3 Nilai dari 4 Komponen Warna

| Titik Sensor | Warna Awal ($W_0$) | Warna Real time ($W_{rt}$) | RedT | GreenT | BlueT | Grayscale |
|---|---|---|---|---|---|---|
| $S_1$ | 11579568 | 11316396 | 4 | 4 | 4 | 4 |
| $S_2$ | 12435393 | 12303807 | 2 | 2 | 2 | 2 |
| $S_3$ | 12961221 | 12556133 | 96 | 46 | 6 | 49 |
| $S_4$ | 12500670 | 11974326 | 8 | 8 | 8 | 8 |

Misalkan nilai *threshold* yang digunakan =10. Maka hasilnya seperti yang terlihat pada tabel 3.4.

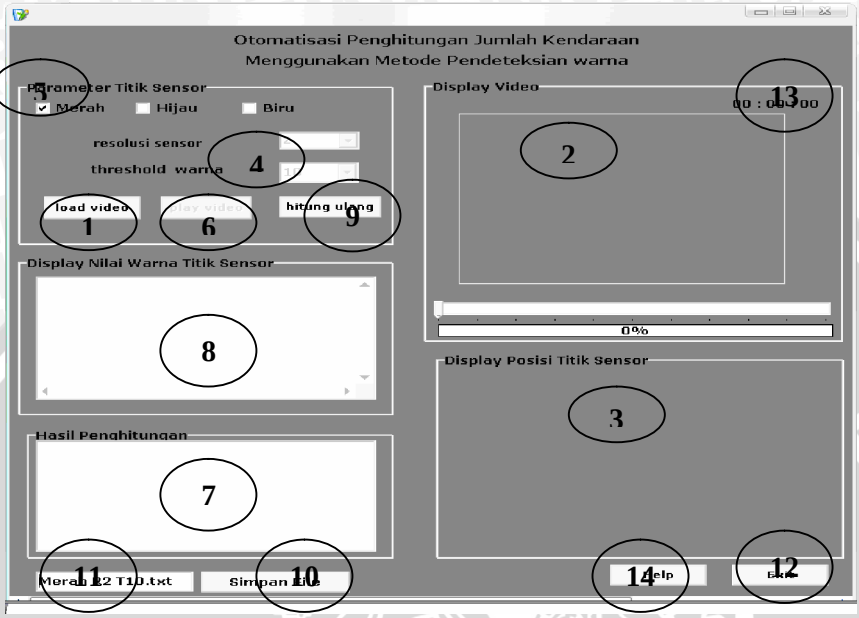
Tabel 3.4 Status Kendaraan Berdasarkan 4 Komponen Warna

| Titik Sensor | RedT | GreenT | BlueT | Grayscale | Status |
|---|---|---|---|---|---|
| $S_1$ | 4 | 4 | 4 | 4 | Tidak ada kendaraan |
| $S_2$ | 2 | 2 | 2 | 2 | Tidak ada kendaraan |
| $S_3$ | 96 | 46 | 6 | 49 | Ada kendaraan |
| $S_4$ | 8 | 8 | 8 | 8 | Tidak ada kendaraan |

### 3.3 Perancangan Antar Muka

Berdasarkan hasil analisa secara keseluruhan terdapat beberapa bagian yang dibutuhkan dalam antar muka sistem peringkas dokumen ini, yaitu :

1. Bagian untuk meng-*input*-kan *video* yang akan akan dianalisis oleh sistem dengan menggunakan komponen *button* yang diberi label *load* video.
2. Bagian untuk menerima *video* masukkan dari pengguna. Karena video mempunyai *frame size* 320 x 240 pixel, maka digunakan komponen panel yang ukurannya dapat disesuaikan dengan *frame size* video.
3. Bagian untuk menampilkan posisi titik sensor kendaraan masukkan dari pengguna. Karena posisi titik sensor ditentukan oleh pengguna, sehingga digunakan komponen *image* untuk menampilkan visualisasi posisi titik sensor.
4. Bagian untuk memberi inputan resolusi titik sensor, serta *threshold* warna .
5. Bagian untuk memilih kondisi elemen warna yang digunakan dalam menghitung jumlah kendaraan. Karena elemen warna bisa dikombinasikan, maka digunakan komponen *checkbox*.
6. Bagian untuk memainkankan video, sekaligus menangkap perubahan warna titik sensor. Bagian ini menggunakan komponen *button* yang diberi label *play* video.
7. Bagian untuk menampilkan jumlah kendaraan yang tercatat pada masing-masing titik sensor, serta jumlah kendaraan akhir yang diperoleh sisitem. Karena informasi jumlah kendaraan yang disajikan terdiri dari banyak baris, maka digunakan komponen *memo*.
8. Bagian untuk menampilkan nilai warna titik sensor selama video dimainkan. Karena perubahan nilai warna yang ditangkap terdiri dari banyak baris, maka digunakan komponen *memo*.
9. Bagian untuk menghapus data nilai warna selama video dimainkan serta hasil jumlah kendaraan . Agar pengguna tidak menghapus secara langsung dari *memo* disediakan tombol ini.
10. Bagian untuk menyimpan jumlah kendaraan yang tercatat oleh sistem.
11. Bagian untuk memberi nama *file* dimana jumlah kendaraan dicatat dalam format .txt.
12. Bagian untuk mengetahui cara menjalankan sistem.
13. Bagian untuk menampilkan lamanya video dimainkan.
14. Bagian untuk keluar dari sistem ini. Dengan menekan tombol ini pengguna dapat keluar dari aplikasi.




Gambar 3.5 Rancangan Antar Muka Penghitung Jumlah Kendaraan

## 3.4 Perancangan Uji Coba

Pada subbab ini akan dilakukan perancangan uji coba dari sistem penghitungan jumlah kendaraan secara otomatis, baik pengujian terhadap sistem apakah telah sesuai dengan analisa dan perancangan, maupun evaluasi jumlah kendaraan yang dihasilkan. Jumlah kendaraan hasil sistem akan dievaluasi, meliputi evaluasi berdasarkan jumlah kendaraan hasil perhitungan manual terhadap 4 komponen warna yaitu berdasarkan warna merah, berdasarkan warna hijau, berdasarkan warna biru dan berdasarkan warna rata-rata ketiganya (*gray scale*).

Untuk mengetahui tingkat keberhasilan sistem dalam menyajikan jumlah kendaraan secara otomatis, maka dilakukan perbandingan dengan jumlah kendaraan yang dihasilkan dari perhitungan manual dengan rumus

$$akurasi = (1 - (\frac{JumlahSistem - JumlahManual}{JumlahManual})) \ x \ 100\% \ldots\ldots\ldots(3.5)$$

### 3.4.1 Bahan pengujian

Bahan yang akan digunakan pada proses pengujian ini, diperoleh dari rekaman video arus lalu lintas yang diambil pada bulan Januari 2008, berdurasi lebih dari 1 menit.

### 3.4.2 Tujuan pengujian

Beberapa hal yang menjadi tujuan dari pelaksanaan pengujian terhadap sistem ini, yaitu :

1. Memeriksa kesesuaian hasil implementasi dengan hasil analisis dan perancangan.
2. Memeriksa perangkat lunak apakah telah berjalan baik (tidak terjadi error).
3. Mengevaluasi jumlah kendaraan hasil sistem berdasarkan 4 komponen warna (merah, hijau, biru dan *grayscale*).
4. Membandingkan jumlah kendaraan hasil sistem dengan jumlah kendaraan hasil penghitungan manual.

### 3.4.3 Skenario dan kriteria pengujian

Pengujian yang dilaksanakan pada Tugas Akhir ini dibagi menjadi 3 bagian, yaitu pengujian implementasi prosedur, evaluasi hasil penghitungan jumlah kendaraan tiap komponen warna berdasarkan resolusi titik sensor dan *threshold* yang digunakan, dan evaluasi tingkat akurasi jumlah kendaraan tiap komponen warna terhadap jumlah kendaraan hasil perhitungan manual.

#### 3.4.3.1 Implementasi prosedur

Sesuai dengan tujuan pengujian pertama dan kedua maka pengujian bagian pertama ini befungsi untuk memeriksa fungsionalitas perangkat lunak dan kesesuaian dengan hasil analisis dan perancangan. Skenario dan kriteria pengujian yang digunakan dapat dilihat pada Lampiran 11.

#### 3.4.3.2 Evaluasi hasil penghitungan jumlah kendaraan

Untuk mengetahui kualitas jumlah kendaraan hasil perhitungan sistem, jumlah kendaraan hasil sistem akan dibandingkan dengan jumlah kendaraan hasil perhitungan manual oleh pengguna. Jumlah kendaraan ini akan diamati tiap komponen warna serta tiap resolusi titik sensor berdasarkan *threshold* warna yang digunakan. Resolusi titik sensor yang digunakan sebanyak 10 variasi yaitu 2x2 pixel, 4x4 pixel, 6x6 pixel, 8x8 pixel, 10x10 pixel, 12x12 pixel, 14x14 pixel, 16x16 pixel, 18x18 pixel,

dan 20x20 pixel dan *threshold* yang digunakan sebanyak 5 variasi yaitu 10, 20, 30, 40 dan 50. Jumlah kendaraan yang akan disajikan merupakan jumlah kendaraan rata-rata tiap resolusi titik sensor yang digunakan dari 5 kali uji coba.

Hasil evaluasi penghitungan jumlah kendaraan akan disajikan dalam tabel berikut :

Tabel 3.5 Rancangan Tabel Hasil Evaluasi Jumlah Kendaraan Rata-Rata Berdasarkan Resolusi Titik Sensor

| Warna | Resolusi Titik Sensor | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| Red | | | | | | | | | | |
| Green | | | | | | | | | | |
| Blue | | | | | | | | | | |
| Grayscale | | | | | | | | | | |

**3.4.3.3 Evaluasi tingkat akurasi**

Hasil penghitungan jumlah kendaraan sistem akan dibandingkan dengan jumlah kendaraan hasil penghitungan manual oleh pengguna. Kemudian hasilnya akan dianalisis tingkat akurasi (ketepatan sistem) dengan persamaan 3.5. Evaluasi tingkat akurasi akan dianalisis berdasarkan *threshold* dan tiap komponen warna dengan posisi titik sensor tetap. Tingkat akurasi yang akan disajikan merupakan hasil dari tingkat akurasi tiap nilai *threshold* dari 5 kali uji coba.

Hasil evaluasi tingkat akurasi akan disajikan dalam tabel berikut :

Tabel 3.6 Rancangan Tabel Hasil Evaluasi Tingkat Akurasi Tiap *Threshold* Berdasarkan Resolusi dan Komponen Warna

| Warna | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | Rata-Rata |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Red | | | | | | | | | | | |
| Green | | | | | | | | | | | |
| Blue | | | | | | | | | | | |
| Gray Scale | | | | | | | | | | | |

Hasil evaluasi tingkat akurasi rata-rata berdasarkan nilai *threshold* akan disajikan dalam tabel 3.7 berikut ini :

Tabel 3.7 Rancangan Tabel Hasil Evaluasi
Tingkat Akurasi Rata-Rata Berdasarkan *Threshold*

| Threshold | Akurasi |
|---|---|
| 10 | |
| 20 | |
| 30 | |
| 40 | |
| 50 | |

Sedangkan hasil evaluasi tingkat akurasi berdasarkan komponen warna akan disajikan dalam tabel 3.8 berikut ini :

Tabel 3.8 Rancangan Tabel Hasil Evaluasi
Tingkat Akurasi Rata-Rata Berdasarkan Komponen Warna

| Komponen Warna | Akurasi |
|---|---|
| Red | |
| Green | |
| Blue | |
| Grayscale | |

### 3.5 Contoh Penghitungan Jumlah Kendaraan

Berikut ini dicontohkan bagaimana cara kerja sistem dalam menghitung jumlah kendaraan. Misalkan jumlah kendaraan hasil perhitungan manual adalah 13 unit, resolusi titik sensor yang digunakan 10 x 10 pixel (seperti yang terlihat pada gambar 3.6) dan nilai *threshold* yang digunakan adalah 10.


Gambar 3.6 Contoh Posisi dan Resolusi Titik Sensor

Maka proses penghitungan jumlah kendaraan adalah sebagai berikut :

1. Hasil pembacaan nilai warna awal ($W_0$) titik sensor dapat dilihat pada tabel 3.9

Tabel 3.9 Hasil Pembacaan Nilai Warna Awal Titik Sensor

| Titik Sensor | Warna Awal ($W_0$) | Red | Green | Blue | Grayscale |
|---|---|---|---|---|---|
| $S_1$ | 11579568 | 176 | 176 | 176 | 176 |
| $S_2$ | 12435393 | 193 | 191 | 189 | 191 |
| $S_3$ | 12961221 | 197 | 197 | 197 | 197 |
| $S_3$ | 12500670 | 190 | 190 | 190 | 190 |

2. Hasil pembacaan nilai warna *real time* ($W_{rt}$) titik sensor dapat dilihat pada tabel 3.10

Tabel 3.10 Hasil Pembacaan Nilai Warna *Real Time* Titik Sensor

| Titik Sensor | Warna Real time ($W_{rt}$) | Red | Green | Blue | Grayscale |
|---|---|---|---|---|---|
| $S_1$ | 11316396 | 172 | 172 | 172 | 172 |
| $S_2$ | 12303807 | 191 | 189 | 187 | 189 |
| $S_3$ | 12556133 | 101 | 151 | 191 | 147 |
| $S_3$ | 11974326 | 8 | 8 | 8 | 8 |

3. Hasil penghitungan jumlah kendaraan berdasarkan selisih tiap komponen warna dapat dilihat pada tabel 3.11

Tabel 3.11 Hasil Penghitungan Jumlah Kendaraan Berdasarkan Selisih Tiap Komponen Warna

| Titik Sensor | Warna Awal ($W_0$) | Warna *Real Time* ($W_{rt}$) | RedT | GreenT | BlueT | Grayscale |
|---|---|---|---|---|---|---|
| $S_1$ | 11579568 | 11316396 | 4 | 4 | 4 | 4 |
| $S_2$ | 12435393 | 12303807 | 2 | 2 | 2 | 2 |
| $S_3$ | 12961221 | 12556133 | 96 | 46 | 6 | 49 |
| $S_4$ | 12500670 | 11974326 | 8 | 8 | 8 | 8 |

4. Hasil pendeteksian status kendaraan berdasarkan nilai *threshold* dapat dilihat pada tabel 3.12.

Tabel 3.12 Hasil Pendeteksian Status Kendaraan Berdasarkan Nilai *Threshold*

| Titik Sensor | RedT | GreenT | BlueT | Grayscale | Threshold | Status |
|---|---|---|---|---|---|---|
| $S_1$ | 4 | 4 | 4 | 4 | <10 | Tidak ada |
| $S_2$ | 2 | 2 | 2 | 2 | <10 | Tidak ada |
| $S_3$ | 96 | 46 | 6 | 49 | >10 | Ada |
| $S_4$ | 8 | 8 | 8 | 8 | <10 | Tidak ada |

# BAB IV
# IMPLEMENTASI DAN PEMBAHASAN

Pada bab ini akan dijelaskan seluruh proses yang sudah dirancang pada bab sebelumnya, tampilan antarmuka dan bagian-bagian *source code* yang dibuat, serta analisa terhadap data yang dihasilkan sistem.

## 4.1 Lingkungan Implementasi

Implementasi merupakan proses transformasi representasi rancangan ke dalam bahasa pemrograman yang dapat dimengerti oleh komputer. Pada bab ini, lingkungan implementasi yang akan dijelaskan meliputi lingkungan implementasi perangkat keras dan perangkat lunak.

### 4.1.1 Lingkungan perangkat keras

Perangkat keras yang digunakan dalam pengembangan sistem penghitungan jumlah kendaraan ini adalah sebuah PC*(Personal Computer)* dengan spesifikasi sebagai berikut :

1. *Processor* Intel® Celeron® 2.66 GHz
2. 256 MB RAM
3. 40 GB HDD
4. Monitor 15”
5. Keyboard
6. Mouse

Perangkat keras ini akan difungsikan sebagai tempat perangkat lunak untuk penelitian dijalankan.

### 4.1.2 Lingkungan perangkat lunak

Perangkat lunak yang digunakan dalam pengembangan sistem penghitungan jumlah kendaraan ini adalah :

1. Sistem operasi *Microsoft Windows XP Professional Edition Service Pack 2* sebagai tempat aplikasi dijalankan.
2. *Borland Delphi 5.0* sebagai *software development* dalam mengembangkan aplikasi untuk menghasilkan data nilai warna titik sensor kendaraan pada saat video dimainkan.

## 4.2 Implementasi Program

Berdasarkan analisa dan perancangan proses yang terdapat pada bab 3, maka pada subbab ini akan dijelaskan implementasi proses-proses tersebut.

### 4.1.2 Implementasi *Capture* Video

Tahap awal dari sistem penghitungan jumlah kendaraan ini adalah melakukan *capture* video sebesar resolusi titik sensor yang digunakan yang telah ditentukan oleh pengguna.

Setelah video di-*input*-kan ke sistem, pengguna meletakkan titik sensor pada permukaan video, kemudian oleh sistem, posisi yang telah ditentukan dicapture warnanya. *Sourcecode* dari proses *capture* video adalah sebagai berikut :

```
  ...
jmlsensor:=4;
    threshold:=strtoint(CBthreshold.Text);
//mengcapture video pada posisi display video
   dcdesk:=GETWINDOWdc(PShowVideo.Handle);
   // Mencapture  4 buah titik sensor
   for i:=0 to jmlsensor-1 do
   begin
      //mencatat koordinat titik sensor yang dipilih pengguna
      kx:=strtoint(MSumbuX.Lines.Strings[i]);
      ky:=strtoint(MSumbuY.Lines.Strings[i]);

//meletakkan hasil capture sebesar resolusi yang digunakan
BITBLT(bmp.CANVAS.HANDLE,(k+1),(k+1)*(i),k,k,DCDESK,kx,ky,SRCCOPY);
   end;
  IMAGE1.Picture.Bitmap:=bmp;
 ...
```

### 4.2.2 Implementasi pembacaan warna

Tahap awal dari sistem penghitungan jumlah kendaraan ini adalah melakukan pembacaan warna titik sensor kendaraan yang telah ditentukan oleh pengguna.

Sebelum sistem melakukan pembacaan warna, pengguna memasukkan resolusi serta *threshold* titik sensor yang digunakan. Berdasarkan masukan ini, sistem akan membaca warna berdasarkan koordinat titik sensor yang telah ditentukan oleh pengguna dan besarnya resolusi titik sensor. *Sourcecode* dari proses pembacaan warna titik sensor kendaraan adalah sebagai berikut :

```
...
for i:=0 to jmlsensor-1 do
   begin
        HIjauT:=0;
        biruT:=0;
        merahT:=0;
        warnaT:=0;

        //mencatat nilai RGB tiap pixel
      FOR RESX:=0 TO K-1 DO
      FOR RESY:=0 TO K-1 DO
      Begin

        //mencatat RGB warna awal
       warna:=IMAGE1.CANVAS.Pixels[(k+1)*0+resx,(k+1)*(i)+resy]  ;
        //mencatat RGB warna real time
      warna1:=IMAGE1.CANVAS.Pixels[(k+1)*1+resx,(k+1)*(i)+resy]  ;

        //mencatat nilai awal masing-masing komponen warna (RGB)
        biru:= warna div $10000;
        HIjau:= (warna mod $10000) div $100;
        merah:= warna mod $100 ;

//mencatat nilai real time masing-masing komponen warna (RGB)
        biru1:= warna1 div $10000;
        HIjau1:= (warna1 mod $10000) div $100;
        merah1:= warna1 mod $100 ;

//penjumlahan nilai selisih masing-masing komponen warna
        HIjauT:=HIjauT+ABS(HIJAU1-HIJAU);
        biruT:=biruT+ABS(biru1-biru);
        merahT:=merahT+ABS(merah1-merah);
        warnat:=warnat+abs(warna1-warna);
      end;

//menampilkan     nilai     masing-masing     komponen     warna
memo1.Lines.Add(inttostr(mediaplayer.position)+';'+inttostr(MERAHT
DIV (K*K))
      +';'+  inttostr(HIJAUT  DIV  (K*K))+';'+  inttostr(BIRUT  DIV
(K*K)));
...
```

### 4.2.3 Implementasi penghitungan jumlah kendaraan

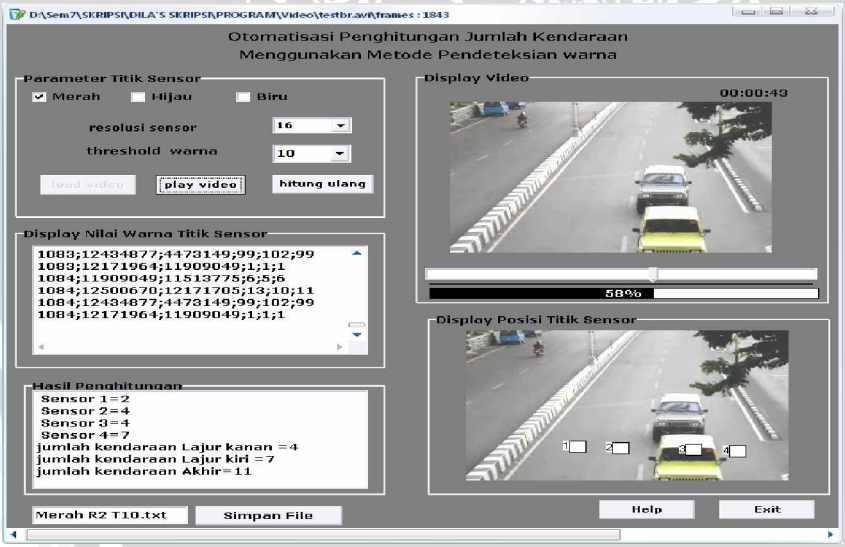
Setelah itu, pengguna menekan tombol *play video,* maka proses pembacaan warna *real time* dan proses penghitungan jumlah kendaraan dimulai. Bila warna yang tercatat berada diluar interval warna awal titik sensor ($W_{realtime} > threshold$) maka diasumsikan kendaraan sedang melintas, dan nilai jumlah kendaraan diakumulasi menjadi 1. Setelah *video* selesai dimainkan, maka jumlah kendaraan akhir (JKA) akan ditampilkan. *Sourcecode* dari proses penghitungan jumlah kendaraan adalah sebagai berikut :

```
...
//menghitung jumlah kendaraan berdasarkan threshold
     if (nilaitotal DIV (K*K))>threshold then
     begin

//status peletakan titik sensor
       if bsensor[i]=false then
//jumlah kendaraan ditingkatkan 1
            ijumlahmobil[i]:=ijumlahmobil[i]+1;
       bsensor[i]:=true;
     end
     else
     begin
       bsensor[i]:=false;
     end;

   //jumlah sensor ideal 4 buah
   for i:=0 to jmlsensor-1  do
   begin
//menampilkan jumlah kendaraan tiap titik sensor
      mlap.Lines.Strings[i]:=' Sensor
'+inttostr(i+1)+'='+inttostr(ijumlahmobil[i]);
   end;
//sensor 1 dan 2 pada lajur kanan
   if ijumlahmobil[0]>ijumlahmobil[1]then
   JalurKanan:= ijumlahmobil[0]
   else
   JalurKanan:= ijumlahmobil[1];

//sensor 3 dan 4 pada lajur kiri
  if ijumlahmobil[2]>ijumlahmobil[3]then
   JalurKiri:= ijumlahmobil[2]
   else
   JalurKiri:= ijumlahmobil[3];
   mlap.Lines.Strings[4]:= 'jumlah kendaraan Lajur kanan ='
+inttoStr(JalurKanan);
   mlap.Lines.Strings[5]:= 'jumlah kendaraan Lajur kiri ='
+inttoStr(JalurKiri);
   mlap.Lines.Strings[6]:= 'jumlah kendaraan Akhir='
+inttoStr(JalurKiri+JalurKanan);
RELEASEdc(PShowVideo.Handle,DCDESK);
```

```
//mengcapture video pada posisi display video
   dcdesk:=GETWINDOWdc(PShowVideo.Handle);
//letakkan hasil capture sebesar resolusi video
BITBLT(bmp1.CANVAS.HANDLE,0,0,
PShowVideo.Width,PShowVideo.height,DCDESK,0,0,SRCCOPY);
   IMAGE2.Picture.Bitmap:=bmp1;
   RELEASEdc(PShowVideo.Handle,DCDESK);
   k:=StrtoInt(CBResolusiSensor.Text);
     //membaca koordinat titik sensor yang dipilih
   for i:=0 to jmlsensor-1 do
   begin
      //mencatat koordinat titik sensor yang dipilih pengguna
      kx:=strtoint(MSumbuX.Lines.Strings[i]);
      ky:=strtoint(MSumbuY.Lines.Strings[i]);
       IMAGE2.Picture.Bitmap.Canvas.Rectangle(kx,ky,kx+k,ky+k);
       IMAGE2.Canvas.TextOut(kx-6,ky,inttostr(i+1));
    end;
end;
end;
...
```

### 4.3 Implementasi Antar Muka

Berdasarkan rancangan antarmuka pada subbab 3.3 maka dihasilkan antarmuka seperti dtunjukkan pada gambar 4.1.




Gambar 4.1 Antar Muka Penghitung Jumlah Kendaraan Otomatis

### 4.4 Implementasi Uji Coba

Pada subbab ini akan dilakukan pembahasan mengenai pengujian yang telah dilakukan pada sistem dan hasil evaluasi dari ringkasan hasil sistem.

#### 4.4.1 Skenario evaluasi

Pada pengujian sistem penghitungan jumlah kendaraan ini, jumlah kendaraan yang diamati berdasarkan 4 komponen warna (*red, green, blue dan grayscale*), dimana *grayscale* merupakan hasil kombinasi dari warna *red*, *green* dan *blue*. Selain itu pembacaan warna *real time* dilakukan tiap 3 frame, karena berdasarkan analisa pada saat membangun sistem, perubahan warna tiap frame dan tiap 3 frame perbedaannya tidak terlalu signifikan.

Dalam pengujian ini variasi resolusi yang digunakan sebanyak 10 variasi (2x2 pixel, 4x4 pixel, 6x6 pixel, 8x8 pixel, 10x10 pixel, 12x12 pixel, 14x14 pixel, 16x16 pixel, 18x18 pixel, dan 20x20 pixel) dan nilai *threshold* yang digunakan dengan 5 variasi (10, 20, 30, 40 dan 50). Dari video yang berdurasi kurang lebih 1 menit, pengamatan dilakukan tiap resolusi. Jadi untuk masing-masing komponen warna jumlah kendaraan diamati sebanyak 10 kali.

Untuk mempelajari tingkat keberhasilan sistem terhadap jumlah kendaraan yang dihasilkan, dibandingkan dengan jumlah kendaraan hasil perhitungan manual, maka dilakukan 5 kali uji coba untuk masing-masing komponen warna dengan nilai *threshold* 10, 20, 30, 40 dan 50.

Ketepatan sistem penghitungan akan dievaluasi menggunakan persamaan (3.5).

#### 4.4.2 Hasil evaluasi

Berukut ini adalah tabel hasil jumlah kendaraan rata-rata dari 5 kali uji coba (dapat dilihat pada lampiran 12) dengan 5 variasi nilai *threshold* dan 10 variasi resolusi titik sensor dengan posisi titik sensor seperti yang terlihat pada gambar 4.2.

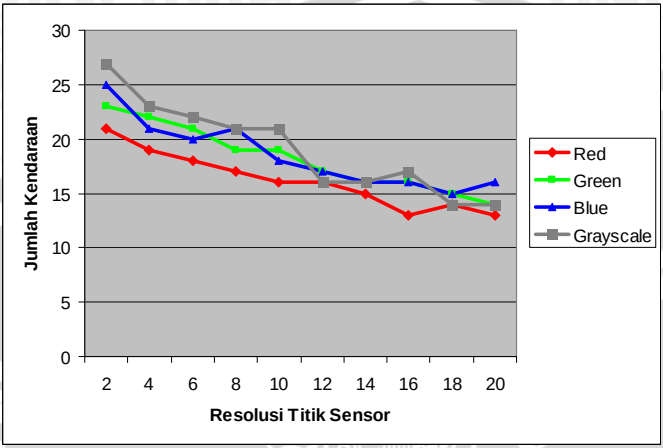

Gambar 4.2. Posisi titik Sensor dengan Resolusi 10x10 Pixel

Tabel 4.1 Evaluasi Jumlah Kendaraan Rata-Rata Berdasarkan Resolusi Titik Sensor

| Warna | Resolusi Titik Sensor | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| Red | 21 | 19 | 18 | 17 | 16 | 16 | 15 | 13 | 14 | 13 |
| Green | 23 | 22 | 21 | 19 | 19 | 17 | 16 | 16 | 15 | 14 |
| Blue | 25 | 21 | 20 | 21 | 18 | 17 | 16 | 16 | 15 | 16 |
| Grayscale | 27 | 23 | 22 | 21 | 21 | 16 | 16 | 17 | 14 | 14 |

Pada tabel 4.1 dapat dilihat bahwa dengan 10 variasi resolusi titik sensor yang digunakan, resolusi lebih besar dari 12, jumlah kendaraan yang dihasilkan mendekati jumlah kendaraan hasil perhitungan manual (13 unit).

Tabel 4.1 dapat disajikan dalam bentuk grafik sebagai berikut:




Gambar 4.3 Grafik Jumlah Kendaraan Rata-Rata
Berdasarkan Resolusi Titik Sensor

Dari gambar 4.3 dapat dilihat bahwa resolusi yang menghasilkan jumlah kendaraan mendekati hasil perhitungan manual adalah pada ukuran lebih besar dari 12 (12 x 12 pixel).

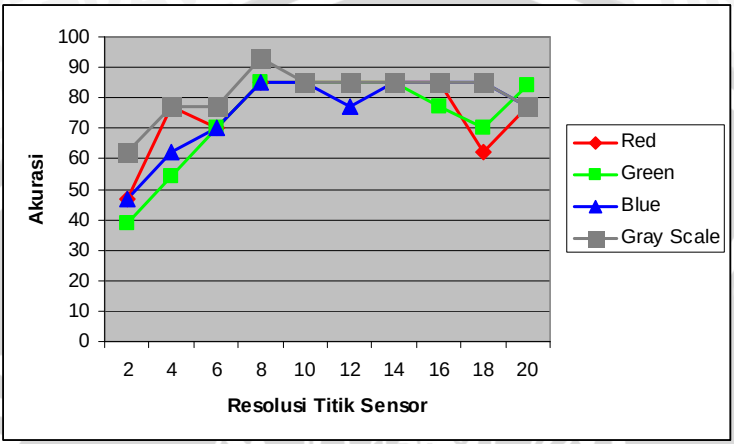
### 4.4.3 Akurasi

nilai akurasi masing-masing *threshold* terhadap jumlah kendaraan hasil perhitungan manual sesuai dengan persamaan 3.5 akan disajikan pada tabel berikut ini :

Tabel 4.2 Evaluasi Tingkat Akurasi
Jumlah Kendaraan dengan *Threshold* = 10

| Warna | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | Rata-Rata |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Red | 47 | 77 | 70 | 85 | 85 | 85 | 85 | 85 | 62 | 77 | 75.8 |
| Green | 39 | 54 | 70 | 85 | 85 | 85 | 85 | 77 | 70 | 84 | 73.4 |
| Blue | 47 | 62 | 70 | 85 | 85 | 77 | 85 | 85 | 85 | 77 | 84.,3 |
| Grayscale | 62 | 77 | 77 | 93 | 85 | 85 | 85 | 85 | 85 | 77 | 81.1 |

Tabel 4.2 dapat disajikan dalam bentuk grafik sebagai berikut:



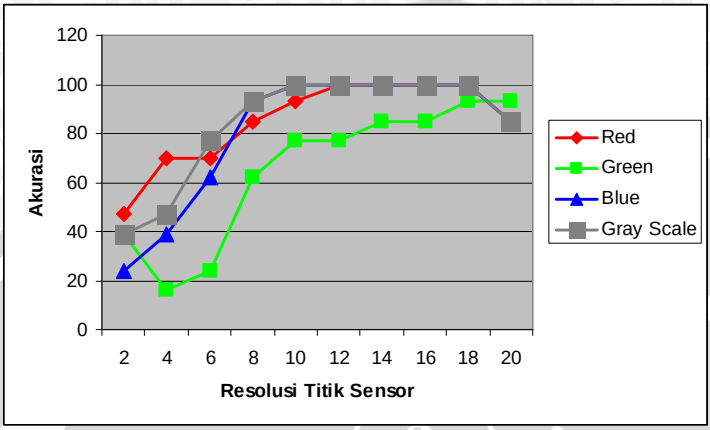

Gambar 4.4 Grafik Tingkat Akurasi
dengan *Threshold* = 10

Pada gambar 4.4 dapat dilihat bahwa pada *threshold* 10, tingkat akurasi masih variatif pada 10 variasi resolusi titik sensor yang digunakan.

Tabel 4.3 Evaluasi Tingkat Akurasi
Jumlah Kendaraan dengan *Threshold* = 20

| Warna | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | Rata-Rata |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Red | 47 | 70 | 70 | 85 | 93 | 100 | 100 | 100 | 100 | 85 | 85 |
| Green | 39 | 16 | 24 | 62 | 77 | 77 | 85 | 85 | 93 | 93 | 65.1 |
| Blue | 24 | 39 | 62 | 93 | 100 | 100 | 100 | 100 | 100 | 85 | 80 |
| Grayscale | 39 | 47 | 77 | 93 | 100 | 100 | 100 | 100 | 100 | 85 | 84 |

Tabel 4.3 dapat disajikan dalam bentuk grafik sebagai berikut:



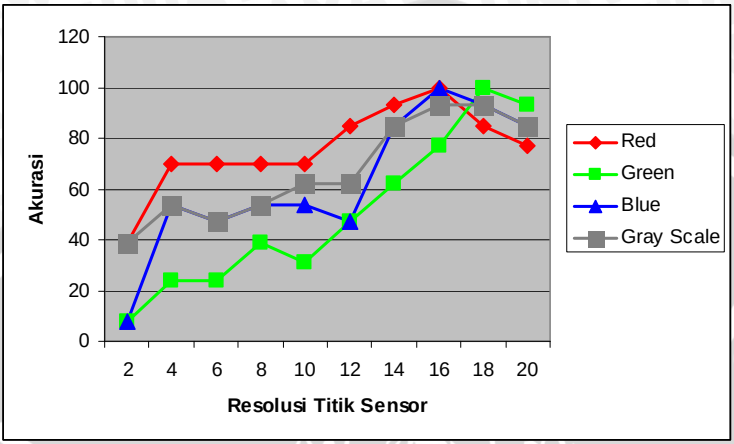

Gambar 4.5 Grafik Tingkat Akurasi
dengan *Threshold* = 20

Pada gambar 4.5 dapat dilihat bahwa pada *threshold* 20, tingkat akurasi hampir konstan pada resolusi di atas 12.

Tabel 4.4 Evaluasi Tingkat Akurasi
Jumlah Kendaraan dengan *Threshold* = 30

| Warna | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | Rata-Rata |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Red | 39 | 70 | 70 | 70 | 70 | 85 | 93 | 100 | 85 | 77 | 75.9 |
| Green | 8 | 24 | 24 | 39 | 31 | 47 | 62 | 77 | 100 | 93 | 50.5 |
| Blue | 8 | 54 | 47 | 54 | 54 | 47 | 85 | 100 | 93 | 85 | 62.7 |
| Grayscale | 39 | 54 | 47 | 54 | 62 | 62 | 85 | 93 | 93 | 85 | 67.4 |

Tabel 4.4 dapat disajikan dalam bentuk grafik sebagai berikut:



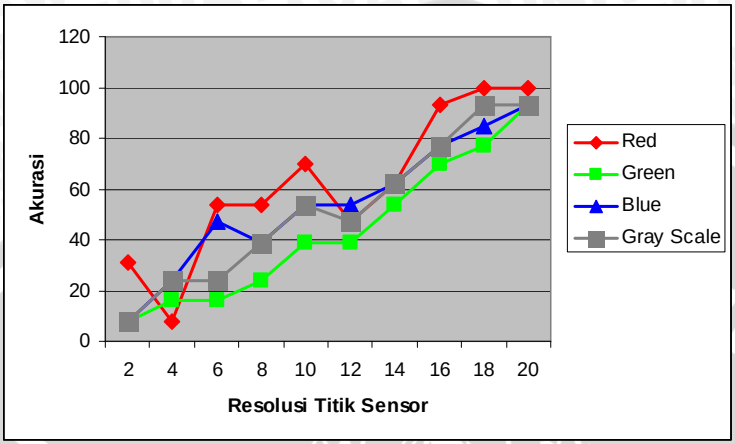

Gambar 4.6 Grafik Tingkat Akurasi
dengan *Threshold* = 30

Pada gambar 4.6 dapat dilihat bahwa pada *threshold* 30, tingkat akurasi mulai konstant pada resolusi 16, tetapi terjadi penurunan tingkat akurasi pada resolusi 18 dan 20.

Tabel 4.5 Evaluasi Tingkat Akurasi
Jumlah Kendaraan dengan *Threshold* = 40

| Warna | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | Rata-Rata |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Red | 31 | 8 | 54 | 54 | 70 | 47 | 62 | 93 | 100 | 100 | 61.9 |
| Green | 8 | 16 | 16 | 24 | 39 | 39 | 54 | 70 | 77 | 93 | 43.6 |
| Blue | 8 | 24 | 47 | 39 | 54 | 54 | 62 | 77 | 85 | 93 | 54.3 |
| Grayscale | 8 | 24 | 24 | 39 | 54 | 47 | 62 | 77 | 93 | 93 | 52.1 |

Tabel 4.5 dapat disajikan dalam bentuk grafik sebagai berikut:



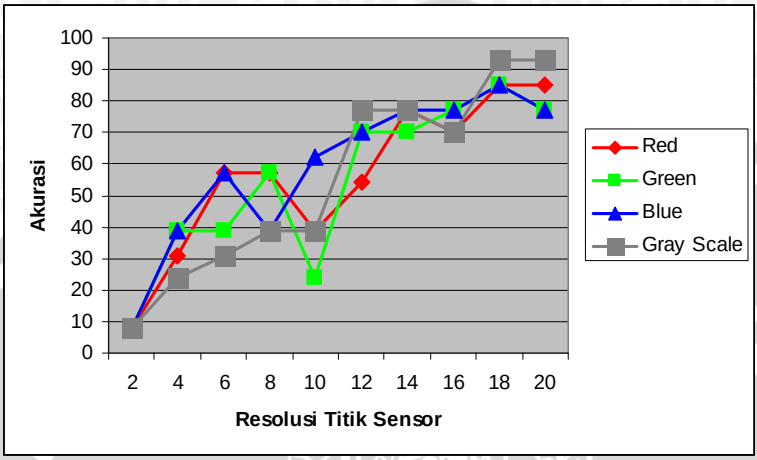

Gambar 4.7 Grafik Tingkat Akurasi
dengan *Threshold* = 40

Pada gambar 4.7 dapat dilihat bahwa pada *threshold* 40, tingkat akurasi hampir sama pada resolusi 20.

Tabel 4.6 Evaluasi Tingkat Akurasi
Jumlah Kendaraan dengan *Threshold* = 50

| Warna | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 | Rata-Rata |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Red | 8 | 31 | 57 | 57 | 39 | 54 | 77 | 70 | 85 | 85 | 54.3 |
| Green | 8 | 39 | 39 | 57 | 24 | 70 | 70 | 77 | 85 | 77 | 53.6 |
| Blue | 8 | 39 | 57 | 39 | 62 | 70 | 77 | 77 | 85 | 77 | 58.1 |
| Gray Scale | 8 | 24 | 31 | 39 | 39 | 77 | 77 | 70 | 93 | 93 | 55.1 |

Tabel 4.6 dapat disajikan dalam bentuk grafik sebagai berikut:




Gambar 4.8 Grafik Tingkat Akurasi
dengan *Threshold* = 50

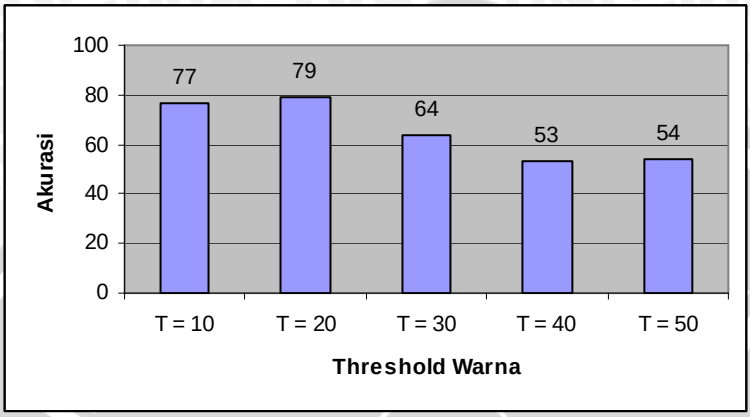
Pada gambar 4.8 dapat dilihat bahwa pada *threshold* 10, tingkat akurasi kembali variatif pada 10 variasi resolusi titik sensor yang digunakan.

Berdasarkan tabel 4.2, tabel 4.3, tabel 4.4, tabel 4.5 dan tabel 4.6, maka tingkat akurasi rata-rata berdasarkan nilai *threshold* dapat disajikan sebagai berikut :

Tabel 4.7 Evaluasi Tingkat Akurasi Rata-Rata
Jumlah Kendaraan Berdasarkan Nilai *Threshold*

| Threshold | Akurasi |
|---|---|
| 10 | 77 |
| 20 | 79 |
| 30 | 64 |
| 40 | 53 |
| 50 | 54 |

Tabel 4.7 dapat disajikan dalam bentuk grafik sebagai berikut:




Gambar 4.9 Grafik Tingkat Akurasi Rata-Rata
Berdasarkan Nilai *Threshold*

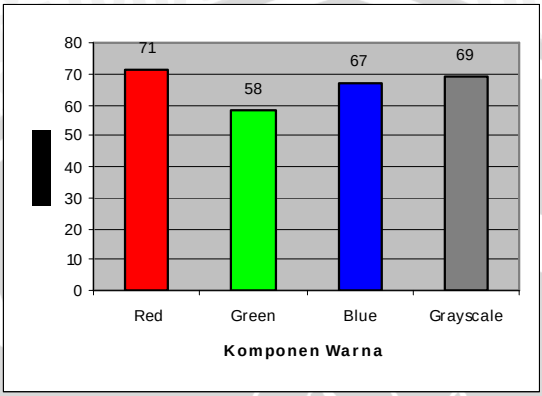
Pada gambar 4.4 dapat dilihat bahwa nilai akurasi (tingkat ketepatan) terbesar pada nilai threshold 20.

Berdasarkan tabel 4.2, tabel 4.3, tabel 4.4, tabel 4.5 dan tabel 4.6, maka tingkat akurasi rata-rata berdasarkan komponen warna dapat disajikan sebagai berikut :

Tabel 4.8 Evaluasi Tingkat Akurasi Rata-Rata
Jumlah Kendaraan Berdasarkan Komponen Warna

| Komponen Warna | Akurasi |
|---|---|
| Red | 71 |
| Green | 58 |
| Blue | 67 |
| Grayscale | 69 |

Tabel 4.8 dapat disajikan dalam bentuk grafik sebagai berikut:




Gambar 4.10 Grafik Evaluasi Tingkat Akurasi Rata-Rata Jumlah Kendaraan Berdasarkan Komponen Warna

Pada gambar 4.10 dapat dilihat bahwa nilai akurasi (tingkat ketepatan) terbesar ada pada komponen warna merah (*red*).

### 4.4.4 Analisa hasil

Dari hasil evaluasi nilai akurasi, didapatkan nilai akurasi yang berbeda untuk nilai *threshod* yang berbeda. Pada *threshold* 10 dan 50 nilai akurasi masih bersifat variatif pada tiap resolusi titik sensor, sehingga belum bisa ditentukan resolusi yang tepat digunakan dalam pengujian sistem. Pada *threshold* 20 mulai terlihat nilai akurasi hampir konstan pada resolusi di atas 12. Pada *threshold* 30 nilai akurasi hampir konstant pada resolusi 16, tetapi terjadi penurunan pada resolusi 18 dan 20, dan pada *threshold* 40 nilai akurasi hampir sama pada resolusi 20.

Dari hasil evaluasi tingkat akurasi di atas, maka resolusi titik sensor yang baik digunakan adalah di atas 12, karena untuk tiap *threshold*, nilai akurasi nya mendekati konstan.

Dari 5 kali uji coba di atas, pada *threshold* 10 dan 50 warna acuan yang memiliki nilai akurasi terbesar adalah warna biru (*blue*), sedangkan pada pada *threshold* 20, 30 dan 40 warna acuan yang memiliki nilai akurasi terbesar adalah warna merah (*red*). Berdasarkan hasil tersebut, maka komponen warna yang baik digunakan adalah warna merah.

Sehingga dapat disimpulkan dengan posisi titik sensor seperti pada gambar 4.2 resolusi yang baik digunakan adalah resolusi dengan ukuran besar (resolusi > 12x12 pixel) dengan nilai *thershold* 20, dan komponen warna yang dijadikan acuan adalah warna merah.

# BAB V
# KESIMPULAN DAN SARAN

## 5.1 Kesimpulan

Berdasarkan penjelasan, uraian, dan analisa pada bab-bab sebelumnya dapat penulis simpulkan bahwa:

1. Dalam penghitungan jumlah kendaraan secara otomatis ini, ukuran resolusi yang baik digunakan adalah lebih besar dari 12 (12x12 pixel).
2. *Threshold* yang baik digunakan dalam proses perhitungan adalah 20.
3. Warna yang paling baik digunakan sebagai acuan dalam proses penghitungan jumlah kendaraan adalah warna merah (*red*).

## 5.2 Saran

Saran yang dapat penulis berikan setelah pengerjaan penelitian ini adalah sebagai berikut.

1. Dari hasil penelitian ini dapat dikembangkan proses penghitungan jumlah kendaraan dengan parameter jumlah titik sensor dan waktu sampel meng-*capture* perubahan warna yang bervariasi.
2. Dari hasil penelitian ini juga dapat digunakan pada jumlah lajur lebih banyak.
3. Sistem pengambilan keputusan terhadap kebijakan-kebijakan yang berhubungan dengan transportasi darat.

## DAFTAR PUSTAKA


Achmad, Balza , Kartika Firdausy. 2005. *Teknik Pengolahan Citra Digital Menggunakan Delphi*. Penerbit Ardi. Yogyakarta.

Audio Video. 2004. *Mengenal MPEG-4 dan DivX*. http://www.audiovideo.co.id Tanggal akses 8 januari 2008.

Gora, Winastwan. 2006. *Dasar Digital Video*. http://mti.ugm.ac.id/~gora/BAB-I-LangkahPraktisWMM2.pdf. Tanggal akses 13 Juli 2007.

Jahne, Bernd. 1997. *Digital Image processing, Concepts, Algorithms, and Scientific Applications*. Univercity of California. Heidelberg.

Jimmi. 2007. *Program deteksi Kemacetan*. http://detik.com Tanggal akses 13 Desember 2007.

Murniasih, Euis. 2006. *Pengenalan Warna Cepat dengan Menggunakan Sensor Visual (Webcam)*. http://digilib.unikom.ac.id/go.phpwww. Tanggal akses 13 Desember 2007.

Robi'in, Bambang. 2004. *Pemrograman Grafis, Multimedia Menggunakan Delphi*. Penerbit Andi. Yogyakarta.

Sun, Hui Fang, dkk. 2005. Digital *Video Transcoding for Transmission and Storage*. CRC Press. USA.

Warpani, Suwardjoko. 1990. *Merencanakan Sistem Perangkutan*. Penerbit ITB. Bandung.

# LAMPIRAN

Lampiran 1. Posisi Titik Sensor dengan Resolusi 2x2 pixel


Lampiran 2. Posisi Titik Sensor dengan Resolusi 4x4 pixel

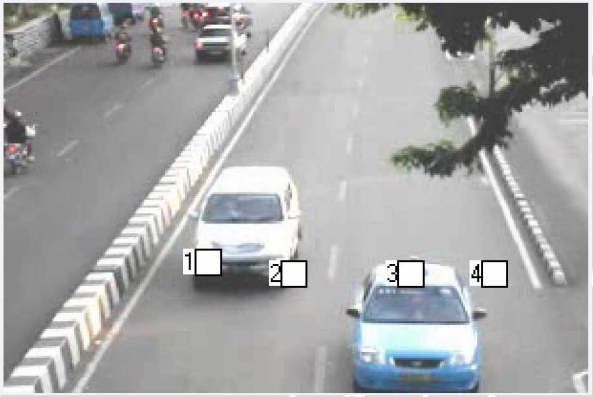
Lampiran 3. Posisi Titik Sensor dengan Resolusi 6x6 pixel


Lampiran 4. Posisi Titik Sensor dengan Resolusi 8x8 pixel

Lampiran 5. Posisi Titik Sensor dengan Resolusi 10x10 pixel


Lampiran 6. Posisi Titik Sensor dengan Resolusi 12x12 pixel

Lampiran 7. Posisi Titik Sensor dengan Resolusi 14x14 pixel




Lampiran 8. Posisi Titik Sensor dengan Resolusi 16x16 pixel

Lampiran 9. Posisi Titik Sensor dengan Resolusi 18x18 pixel


Lampiran 10. Posisi Titik Sensor dengan Resolusi 20x20 pixel

Lampiran 11. Pengujian Fungsionalitas Sistem

| No | Hal yang diujikan | Prosedur yang terlibat | Skenario pengujian | Kriteria pengujian | Hasil |
|---|---|---|---|---|---|
| 1. | Capture video sebesar resolusi titik sensor | BacaSensor | Jalankan aplikasi, klik tepat pada permukaan display video | Pada bagian penyimpanan sementara (komponen image) akan muncul hasil capture | Sukses |
| 2. | Menampilkan posisi titik sensor | BacaSensor | Jalankan aplikasi, klik tepat pada permukaan display video | Pada bagian display posisi titik sensor (berbentuk persegi) akan terlihat gambar titik sensor sesuai dengan resolusi yang digunakan | Sukses |
| 3. | Pembacaan Warna | BacaSensor | Jalankan aplikasi, klik tepat pada permukaan display video, tekan tombol play video | Pada bagian penyimpanan sementara (komponen image) akan terlihat perubahan warna selama video dimainkan, dan pada bagian display nilai warna titik sensor akan ditampilkan nilai dari | Sukses |

| | | | | | |
|---|---|---|---|---|---|
| | | | | perubahan warna tiap frame | |
| 4. | Penghitungan Jumlah Kendaraan | BacaSensor | Jalankan aplikasi, klik tepat pada permukaan display video, tekan tombol play video | Pada bagian hasil penghitungan akan ditampilkan jumlah kendaraan tiap titik sensor dan jumlah kendaraan akhir | Sukses |

Lampiran 12.Tabel Hasil Pengujian Jumlah Kendaraan Berdasarkan Nilai *Threshold*

*Threshold* = 10

| Warna | Resolusi Titik Sensor | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| Red | 20 | 16 | 17 | 15 | 15 | 15 | 15 | 15 | 18 | 16 |
| Green | 21 | 19 | 17 | 15 | 15 | 15 | 15 | 16 | 17 | 16 |
| Blue | 20 | 18 | 17 | 15 | 15 | 16 | 15 | 15 | 15 | 16 |
| Grayscale | 18 | 16 | 16 | 14 | 15 | 15 | 15 | 15 | 15 | 16 |

*Threshold* = 20

| Warna | Resolusi Titik Sensor | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| Red | 20 | 17 | 17 | 15 | 14 | 13 | 13 | 13 | 13 | 15 |
| Green | 21 | 24 | 23 | 18 | 16 | 16 | 15 | 15 | 14 | 14 |
| Blue | 23 | 21 | 18 | 14 | 13 | 13 | 13 | 13 | 13 | 15 |
| Grayscale | 21 | 20 | 16 | 14 | 13 | 13 | 13 | 13 | 13 | 15 |

*Threshold* = 30

| Warna | Resolusi Titik Sensor | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| Red | 21 | 17 | 17 | 17 | 17 | 15 | 14 | 13 | 11 | 10 |
| Green | 26 | 23 | 23 | 21 | 22 | 20 | 18 | 16 | 13 | 12 |
| Blue | 25 | 19 | 20 | 19 | 19 | 20 | 15 | 13 | 12 | 11 |
| Grayscale | 21 | 19 | 20 | 19 | 18 | 18 | 15 | 12 | 12 | 11 |

*Threshold* = 40

| Warna | Resolusi Titik Sensor | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| Red | 22 | 25 | 19 | 19 | 17 | 20 | 18 | 14 | 13 | 13 |
| Green | 26 | 24 | 24 | 23 | 21 | 21 | 19 | 17 | 16 | 14 |
| Blue | 25 | 23 | 20 | 21 | 19 | 19 | 18 | 16 | 15 | 12 |
| Gray Scale | 25 | 23 | 23 | 21 | 19 | 20 | 18 | 16 | 14 | 12 |

*Threshold* = 50

| Warna | Resolusi Titik Sensor | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |
| Red | 25 | 22 | 20 | 20 | 21 | 19 | 16 | 17 | 15 | 15 |
| Green | 25 | 21 | 21 | 20 | 23 | 17 | 17 | 16 | 15 | 16 |
| Blue | 25 | 21 | 20 | 21 | 18 | 17 | 16 | 16 | 15 | 16 |
| Grayscale | 27 | 23 | 22 | 21 | 21 | 16 | 16 | 17 | 14 | 14 |


UNIVERSITAS BRAWIJAYA