PERANCANGAN APLIKASI DETEKSI KEMACETAN BERDASARKAN PENGOLAHAN VIDEO DIGITAL MENGGUNAKAN METODE FRAME DIFFERENCE BERBASIS ANDROID

Traffic Detection Application Design By Video Digital Proccessing Using Frame Difference Method Based On Android

Tikki Capriati Marieski[1], Dr. Ir Bambang Hidayat, DEA.[2], Irma Safitri ST, M.I.T.[3]
[123]Prodi S1 Teknik Telekomunikasi, Fakultas Teknik Elektro, Universitas Telkom

[1]Tikkicapriati@gmail.com [2]bhidayat@telkomuniversity.ac.id
[3]irmasafitri@telkomuniversity.ac.id

Abstrak

Intelligent transport system (ITS) saat ini sedang banyak dikembangkan oleh para *engineer*. Salah satunya adalah untuk aplikasi system monitoring kemacetan di kota metropolitan. Kemacetan di kota-kota besar sudah terbiasa terjadi sehari-hari, untuk itu pada tugas akhir ini dibuat suatu system yang dapat mengetahui kondisi kepadatan lalu lintas disuatu ruas jalan. Salah satu subsistem yang diterapkan pada ITS ini adalah deteksi kepadatan kendaraan di suatu ruas jalan. Dengan memanfaatkan pengolahan sinyal digital kita dapat memproses video yang nantinya dapat menggambarkan kondisi kepadatan jalan tersebut. Pada tugas akhir ini dibuat suatu aplikasi berbasis android yang dapat mengolah video yang direkam menggunakan *handphone* dan selanjutnya video tersebut diproses. Metode yang digunakan dalam penentuan kepadatan lalu lintas ini adalah metode *frame difference*. Dengan membedakan *frame* saat ini dengan *frame* sebelumnya diharapkan system dapat mengidentifikasi perubahan *frame* untuk menentukan kepadatan lalu lintas. Proses merekam gambar dengan menggunakan kamera *handphone* seluler berbasis android yang sudah terinstall aplikasi dan selanjutnya dapat diakses oleh user dengan output sistem berupa kondisi jalan macet atau lancar. Kehandalan sistem diuji dengan melakukan simulasi percobaan. Hasil pengujian menunjukan bahwa sistem mampu melakukan identifikasi tingkat kepadatan lalu lintas suatu ruas jalan secara real time dengan persentase keberhasilan rata-rata sebesar 90% pada kondisi pagi hari, 85%% pada kondisi siang hari, dan 75% pada kondisi malam hari.

Kata kunci : kepadatan lalu lintas, *frame difference* , android

Abstract

Intelligent transport system (ITS) is being developed by many engineers. One of the implementation is an application for traffic management centre in metropolitan city. Traffic jam in big cities such as Jakarta, Bandung, and Surabaya is a usual thing in daily life. Therefore, in this final task, a system to identify traffic density on the roads was made. One of the subsystem which is applied at ITS is traffic density detection on the roads. By utilizing digital signal processing we can process the video that will be describe the condition of traffic roads. This final project is about creating an application that can process a video which is recorded using handphone. Frame difference is used as the method to determine traffic density. By comparing current frame with previous this system is expected to identify the changes of frame to determine traffic density. The process of recording images is using mobile phone cameras based on android that already installed the application. Then this application can be accessed by the users which the output of this system are normal traffic, or heavy traffic. Performance of the system can be tested by applying simulation test. The result shows that the system is able to detect traffic density with accuracy rate is 90% in the morning condition, 85% in the daylight, 75% in the afternoon.

Keywords : ***traffic density, frame difference, android***

1. Pendahuluan

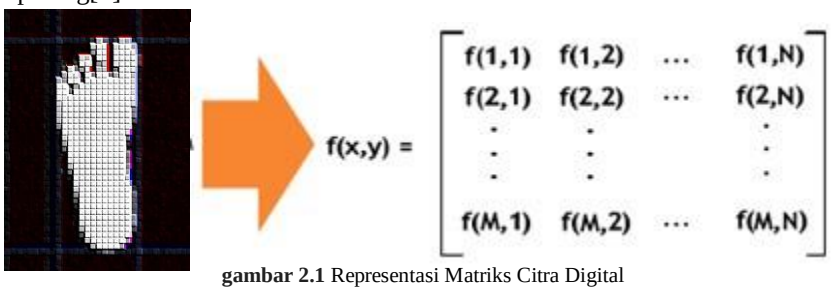
Intelligent transport system merupakan sebuah aplikasi *monitoring* lalu lintas yang sedang banyak dikembangkan untuk mengatur dan memonitoring segala peristiwa yang terjadi di suatu ruas jalan. *Inteligent transport*

system saat ini sudah diterapkan di beberapa negara maju seperti Amerika, dan Jepang. ITS ini pun dibangun dari beberapa subsistem, seperti pengolahan video, database sistem dan lain sebagainya. *Intelligent tansport system* ini diharapkan dapat menjadi solusi bagi kepadatan lalu lintas di ibu kota. Salah satu subsistem yang diterapkan pada ITS ini adalah deteksi kepadatan kendaraan di suatu ruas jalan. Dengan memanfaatkan pengolahan sinyal digital kita dapat memproses video yang nantinya dapat menggambarkan kondisi kepadatan jalan pada saat itu. Pada penelitian sebelumnya, terdapat juga deteksi kepadatan lalu lintas yang dapat diakses oleh user melalui *sms gateway*. Pada penelitian ini, subsistem dirancang menjadi sebuah aplikasi untuk mengetahui kepadatan lalu lintas yang dapat diaskses via aplikasi android. Sehingga pengendara dapat mengetahui kondisi kepadatan jalan. Dengan demikian intelligent transport system ini diharapkan dapat mengurangi kepadatan lalu lintas dan dapat diakses dengan mudah. Dalam pengerjaan tugas akhir ini aplikasi dibangun dari sistem pengolahan video. Pada sistem ini video yang direkam akan dibagi kedalam frame-frame dan hasilnya dapat diketahui untuk menetukan tingkat kepadatan lalu lintas. Frame tersebut akan dihitung perbedaannya dan dibuat threshold tertentu untuk penentuan status pada waktu tersebut di suatu ruas jalan. Keputusan yang akan diperoleh nantinya dibagi kedalam dua kategori, yaitu : macet dan lancar. Dengan adanya sistem ini nantinya diharapkan user yang mengakses dapat mengatahui kondisi kepadatan lalu lintas melalui sistem android..

2. Dasar Teori

2.1 Pengolahan Citra

Citra atau gambar adalah kombinasi antara titik, garis, bidang, dan warna untuk menciptakan suatu imitasi dari suatu obyek biasanya obyek fisik atau manusia. Citra bisa berwujud dua dimensi, seperti lukisan, foto, dan berwujud tiga dimensi, seperti patung[1].



gambar 2.1 Representasi Matriks Citra Digital

Dari Gambar 2.1 mengenai representasi citra digital, terdapat koordinat (*x,y*) yang nilai intensitasnya adalah *f(x,y).* Terdapat beberapa jenis citra digital yang sering digunakan, yaitu citra biner, citra *grayscale* dan citra warna

2.2 Video Digital

Video pada dasarnya merupakan array tiga dimensi. Dua dimensi digunakan untuk menggambarkan ruang pergerakan citra (spatial), dan satu dimensi lainnya menggambarkan waktu. Video digital tersusun atas serangkaian frame yang ditampilkan dalam durasi waktu tertentu sehingga dikenal sebuah satuan fps (frame per second). Tinggi atau rendahnya nilai fps akan mempengaruhi kelajuan frame. Jika laju frame cukup tinggi, maka mata manusia akan melihatnya sebagai rangkaian yang kontinyu. Setiap frame merupakan gambar atau citra digital. Suatu citra digital direpresentasikan dengan sebuah matriks yang masing-masing elemnnya merepresentasikan nilai intensitas atau kedalaman warna. Video digital juga memiliki beberapa karakteristik seperti citra, yaitu terdapat piksel yang menentukan dimensi atau resolusi, selain itu terpadat karakteristik yang lain berupa kuantisasi atau kedalaman bit dan frame rate[2].

2.2.3 Sistem Warna Pada Video Digital

Pada video digital, umumnya representasi warna video dipisahkan menjadi komponen – komponen, baik komponen warna maupun komponen kecerahan. Penyajian semacam ini disebut component video. Berberapa cara pemisahan komponen tersebut adalah RGB, YUV.

Pada RGB data video dipisahkan menjadi kompnen-komponen untuk masing-masing warna, yaitu merah(red), hijau(green), dan biru(blue). Warna tiap piksel ditentukan oleh intensitas masing – masing komponen warna. Pada YUV pemisahan komponen tidak hanya dilakukan dengan pemisahan warna namun juga dilakukan dengan memisahakan menurut komponen kecerahan (luminance) dan komponen warna (crominance).

RGB adalah suatu model warna yang terdiri dari merah, hijau, dan biru, digabungkan dalam membentuk suatu susunan warna yang luas. Setiap warna dasar, misalnya merah, dapat diberi rentang-nilai. Untuk monitor komputer, nilai rentangnya paling kecil = 0 dan paling besar = 255. Pilihan skala 256 ini didasarkan pada cara mengungkap 8

digit bilangan biner yang digunakan oleh mesin komputer. Dengan cara ini, akan diperoleh warna campuran sebanyak 256 x 256 x 256 = 1677726 jenis warna.

Sebuah jenis warna, dapat dibayangkan sebagai sebuah vektor di ruang 3 dimensi yang biasanya dipakai dalam matematika, koordinatnya dinyatakan dalam bentuk tiga bilangan, yaitu komponen-x, komponen-y dan komponen-z. Misalkan sebuah vektor dituliskan sebagai r = (x,y,z). Untuk warna, komponen-komponen tersebut digantikan oleh komponen R(ed), G(reen), B(lue). Jadi, sebuah jenis warna dapat dituliskan sebagai berikut: warna = RGB(xxx, xxx, xxx). Putih = RGB (255,255,255), sedangkan untuk hitam= RGB(0,0,0).

YUV adalah sebuah *color space* dengan Y merupakan komponen *luminance* (*brightness*) dan U dan V adalah komponen *chrominance* (warna). YUV biasanya digunakan pada aplikasi video. Sinyal YUV diciptakan dari sumber RGB (*red*, *green*, *blue*), dengan diberi bobot tertentu, nilai R, G dan B ditambahkan untuk menghasilkan sinyal Y, yang menyatakan terang-gelap secara keseluruhan, atau *luminance* dari titik tersebut. Sinyal U lalu dibentuk dengan mengurangkan sinyal *blue* dari RGB dengan Y, dan V dengan mengurangkan *red*.

Pada monitor nilai *luminance* digunakan untuk merepresentasikan warna RGB, mewakili intensitas sebuah warna RGB yang diterima oleh mata. *Chrominance* merepresentasikan corak warna dan saturasi (*saturation*). Nilai komponen ini juga mengindikasikan banyaknya komponen warna biru dan merah pada warna.

2.2.4 Sistem Warna Greyscale

Grayscalling adalah teknik yang digunakan untuk mengubah citra berwana (RGB) menjadi bentuk grayscale atau tingkat keabuan (dari hitam ke putih)[4]. Dengan pengubahan ini, matriks penyusun citra yang sebelumnya 3 matriks akan berubah menjadi 1 matriks saja. suatu citra digital grayscale atau greyscale adalah suatu citra dimana nilai dari setiap pixel merupakan sample tunggal. Citra yang ditampilkan dari citra jenis ini terdiri atas warna abu-abu, bervariasi pada warna hitam pada bagian yang intensitas terlemah dan warna putih pada intensitas terkuat. Citra grayscale berbeda dengan citra ”hitam-putih”, dimana pada konteks komputer, citra hitam putih hanya terdiri atas 2 warna saja yaitu ”hitam” dan ”putih” saja.

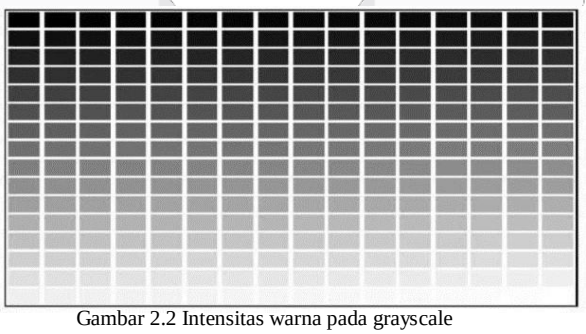
Pada citra grayscale warna bervariasi antara hitam dan putih, tetapi variasi warna diantaranya sangat banyak. Citra grayscale seringkali merupakan perhitungan dari intensitas cahaya pada setiap pixel pada spektrum elektromagnetik single band . Citra grayscale disimpan dalam format 8 bit atau diwakili oleh 1 *byte* untuk setiap sample pixel, yang memungkinkan sebanyak 256 intensitas. Format ini sangat membantu dalam pemrograman karena manupulasi bit yang tidak terlalu banyak. Pada aplikasi lain seperti pada aplikasi medical imaging dan remote sensing biasa juga digunakan format 10,12 maupun 16 bit. cara yang umumnya dilakukan adalah dengan memberikan bobot untuk masing-masing warna dasar red, green, dan blue dimana bila dijumlahkan ketiga bobot tersebut akan bernilai 1.

$$Y = (a \times R)\ (b \times G)\ (c \times B) \quad (2.1)$$

Dimana :

Y = Citra grayscale

a,b,c = koefisien pengali yang berjumlah 1



Gambar 2.2 Intensitas warna pada grayscale

2.3 Sistem Deteksi Gerakan

Sistem deteksi gerakan adalah suatu sistem untuk dapat mendeteksi gerakan objek bergerak dalam video. Input sistem berupa video atau urutan gambar yang ditangkap oleh sebuah kamera sedangkan output sistem berupa pengenalan objek yang bergerak dan klasifikasi atas kemacetan lalu lintas di suatu ruas jalan. Proses deteksi gerakan yang akan dilakukan dalam tugas akhir ini adalah metode perbandingan *frame to frame* (*frame difference method*). Metode ini dilakukan dengan membandingkan frame-frame citra yang ditangkap sesuai dengan urutan waktu[3].

tahap ini dilakukan pengurangan nilai-nilai piksel antara *frame* saat itu dengan *frame* berikutnya. *Frame* yang dikurangi adalah *frame* yang sudah diubah ke dalam format *grayscale*. Hasil pengurangan ini selanjutnya diabsolutkan

atau dimutlakan. Lalu dari hasil pengurangan pada *frame-frame* tersebut dijumlahkan nilai selisih-selisih piksel tersebut untuk ditemukan suatu nilai yang disebut distance. Nilai distance tersebut menjadi patokan ditentukannya nilai *threshold* untuk penentuan tingkat kepadatan lalu lintas.

Dapat di represetasikan dengan persamaan sebagai berikut :

$$SAD = \Sigma^{n}_{i=1} \Sigma^{n}_{J=1} | I - J | \quad \textbf{(2.2)}$$

Dimana : i= frame i.

j= frame j.

n= jumlah *pixel* pada frame.

2.4 ANDROID STUDIO

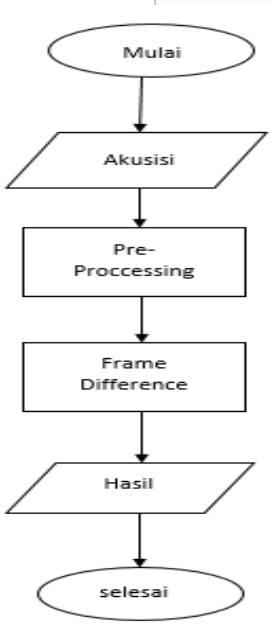
Android Studio adalah sebuah IDE untuk Android Development. Android Studio merupakan pengembangan dari Eclipse IDE [5]. Salah satu alasan menggunakan android studio adalah karena android studio mudah dalam pengoprasiannya dengan *visual layout* yang terlihat nyata. Selain itu, android studio juga berbasis teks dan menampilkan sugesti perintah dengan teks yang terkait yang dapat memudahkan kita untuk menulis *source code* dalam melakukan pemrograman. Maka dari itu, hal tersebut dapat membantu kita dalam memahami setiap baris code yang kita butuhkan dalam membuat aplikasi android.

3. Perancangan dan Implementasi Sistem

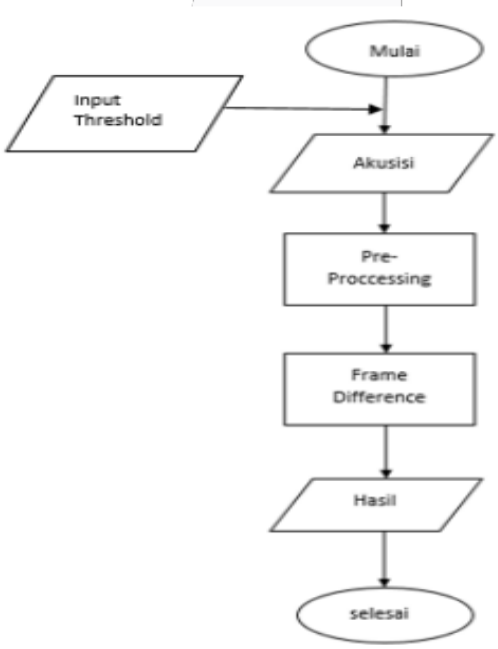
Desain model sistem yang sistematis dan terstruktur dengan baik diperlukan agar menghasilkan sebuah sistem yang optimal. Pada sistem ini akan digunakan piranti berupa mobile application berbasis Android. Piranti ini akan dijadikan sebagai interface antara pengguna (user) dengan aplikasi yang akan dibuat dalam penelitian pendeteksi kemacetan lalu lintas ini. . Sistem disimulasikan dengan cara merekam kondisi suatu jalan terlebih dahulu. Setelah itu hasil rekaman video tersebut diolah untuk melakukan perhitungan kepadatan lalu lintas. Hasil keluaran sistem berupa lancar, atau macet.

3.1 Perancangan Sistem

Berikut adalah diagram secara keseluruhan yang menerangkan cara kerja sistem secara keseluruhan.



Gambar 3.1 blok diagram sistem data latih

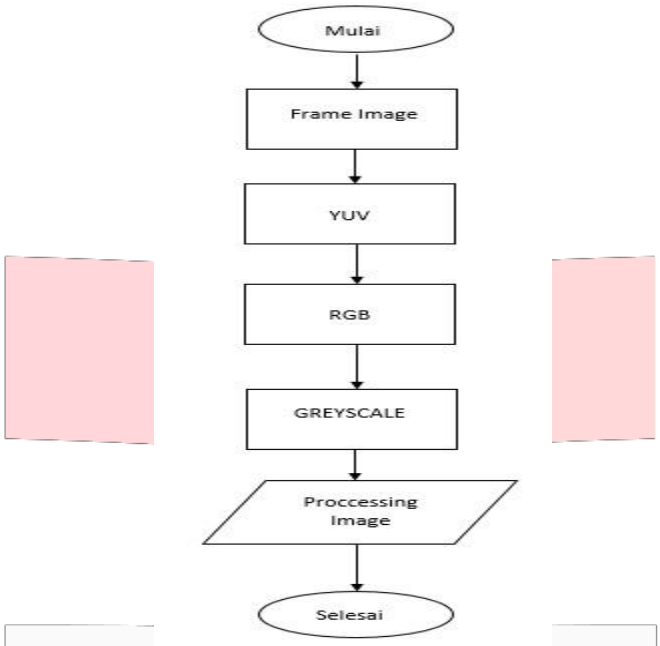


Gambar 3 .2 Blok Diagram Sistem Data Uji

Dari diagram blok diatas, Pada gambar diatas terdapat akusisi video dimana skema ini menggambarkan bagaimana cara pengambilan video yang direkam pada satu sisi ruas jalan. Perekaman video dilakukan secara langsung selama 30 detik. Video yang telah direkam akan diolah untuk dicari nilai distance dari tingkat kepadatannya.

3.2 Pre- Proccessing

Pre-processing merupakan sebuah proses awal pengolahan video untuk mempersiapkan video yang akan diolah ke tahap selanjutnya. Subsistem ini membaca video dengan format YUV beresolusi 320x240 pixel. Video input ini merupakan hasil rekaman yang diekstraksi menjadi frame-frame dan kemudian diubah ke dalam jenis warna grayscale.



Gambar 3 3 Blok Diagram *Pre- Proccessing*

1. Previewing video yang sedang berlangsung secara real time

   Pada tahap ini, kamera akan mengambil citra pada frame – frame preview kamera dan citra yang didapat adalah format YUV. Pada tahap ini merupakan proses yang penting, karena melalui proses inilah data input akan didapatkan untuk dilakukan proses penghitungan. Pada proses ini kamera akan mengambil nilai-nilai *pixel* pada frame-frame yang kamera tangkap. Kecepatan kamera untuk proses *preview* kamera yang digunakan adalah 30 fps.

2. Ubah format video tersebut dari YUV ke RGB

   Dalam tahap ini dilakukan penelusuran seluruh pixel yang berada pada frame citra untuk diketahui nilai dengan format YUV lalu di konversi ke GRB Penelusuran dilakukan mulai dari kiri atas dan berakhir di kanan bawah citra. Proses ini dilakukan karena kamera mendapatkan citra dengan format YUV. RGB merupakan suatu model warna yang terdiri atas tiga warna yaitu R(*Red*), G(*Green*), B(*Blue*). Proses ini dilakukan karena di selanjutnyanya penulis akan melakukan proses *grayscaling* guna mempermudah proses perhitungan. Formulasi konversi dari YUV ke RGB adalah:

   R = (Y + 1634 * V);
   G = (Y - 833 * V - 400 * U);
   B = (Y + 2066 * U); **(3.1)**

3. Ubah format RGB ke Greyscale

   Suatu citra warna RGB di ubah menjadi citra *grayscale* untuk memperoleh informasi intensitas dari gambar, sehingga dapat di sortir mulai dari hitam untuk intensitas yang paling lemah samapai dengan putih yang intensitas yang paling kuat. Dan mendapatkan nilai *grayscale* nya

   Pada tahap ini, Selanjutnya citra di konversi kedalam pixel *grayscale*, agar mempermudah untuk di proses ke tahap selanjutnya. *grayscaling* adalah mengubah citra warna menjadi citra grayscale, hal ini digunakan untuk menyederhanakan model citra dan guna mempermudah proses perhitungan. Untuk memperoleh informasi intensitas dari citra tersebut, citra dapat di sortir secara ekslusif mulai dari hitam untuk intensitas yang paling lemah sampai dengan putih untuk intensitas yang paling kuat. Citra warna terdiri dari 3 layer matrik yaitu R-layer, G-

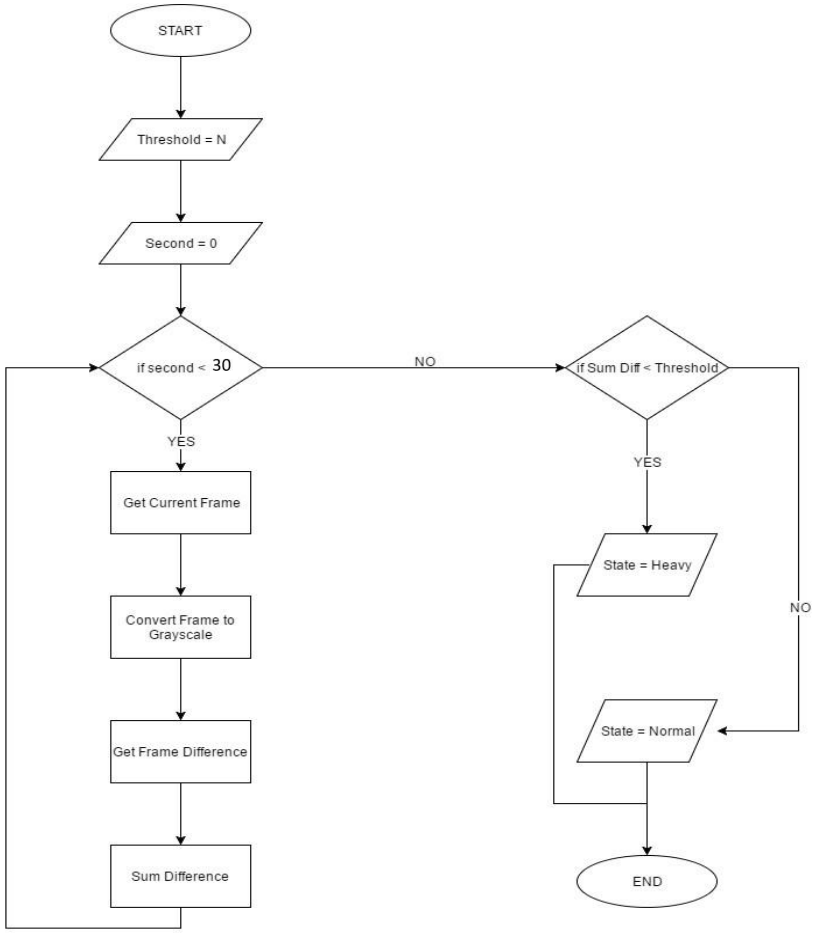
layer, B-layer. Jadi dalam proses ini akan mengubah 3 layer matriks citra berwarna menjadi 1 layer matriks *grayscale* dan hasilnya citra *grayscale*. Dalam citra ini tidak ada lagi warna, yang ada adalah derajat keabuan

Untuk mengubah citra berwarna yang mempunyai nilai matrik masing-masing R, G dan B menjadi citra *grayscale* dengan nilai I. Berikut adalah perhitungan untuk mendapatkan *grayscale*

Grayscale = (R + G + B) / 3 **(3.2)**

4. Kemudian frame ini akan dianalisis untuk menentukan tingkat kepadatan lalu lintas.

3.3 Blok *Frame Difference*



Gambar 3 .4 Blok Diagram *Frame Difference dan Penentuan State*

a. Get Current Frame
Data frame pertama dengan menggunakan camera android

b. Grayscale Convertion
mengambil berupa nilai pixel-pixel dari greyscalling frame pada pre-proccessing

c. *Get Frame Difference*
Pada tahap ini frame-frame yang telah di ekstrak ke grayscale diolah menggunakan metode frame difference, yaitu melakukan pengurangan nilai pixel antara frame saat ini dengan frame selanjutnya atau juga bisa disebut dengan frame odd dan frame even.

d. *Sum Difference*
Setelah melakukan proses previewing, mengubah citra YUV ke citra RGB dan mengubah citra RGB menjadi citra *grayscale*, langkah selanjutnya adalah menghitung nilai *Sum of absolute differences* (SAD) pada setiap frame yang telah dikonversi menjadi *grayscale*. Pada tahap ini, tiap *pixel* akan dibandingkan dengan *pixel* lain. Metode ini bekerja dengan cara mengambil perbedaan mutlak (*Absolute Differences*) antara setiap *pixel* di citra awal dengan citra-citra berikutnya.

langkah-langkah metode SAD adalah sebagai berikut

1. Bandingkan selisih *pixel* 1 di frame 1 dan *pixel* 1 di frame 2, *pixel* 2 pada frame 1 dan *pixel* 2 pada frame 2

dan begitu selanjutnya.

2. *Absolute*-kan nilai hasil selisih *pixel* antar frame.
3. Jumlahkan seluruh selisih *pixel* tersebut sehingga mendapatkan nilai SAD nya.

Proses pendeteksian gerak dengan menggunakan *Sum of absolute differences* (SAD) dimulai dengan menangkap gambar sebagai citra referensi atau citra awal. Karena pada penelitian ini dilakukan pengambilan video secara *real-time*, citra yang menjadi referensi adalah citra yang pertama kali kamera tangkap. Langkah selanjutnya kamera akan menambah gambar sebagai citra baru yang akan dibandingkan dengan citra awal. Jika tidak ada perubahan nilai *pixel* dibandingkan dengan citra awal maka kamera akan menambah citra baru untuk kembali dibandingkan dengan citra sebelumnya dan jika ada perubahan *pixel* maka akan ditandai sebagai pergerakan.

Untuk Setiap frame, lakukan pemrosesan setiap *pixel* yang menggunakan persamaan *Sum of absolute differences* (SAD) berikut :

$$SAD = \Sigma^{n}_{i=1} \Sigma^{n}_{J=1} | I - J | \quad \textbf{(3.3)}$$

Dimana : i= frame i.
j= frame j.
n= jumlah *pixel* pada frame.

Pada proses ini dari hasil pengurangan pada *frame-frame* tersebut diatas dijumlahkan nilai selisih-selisih piksel diantara 2 frame tersebut dan dilakukan secara terus menerus sampai waktu yang ditentukan yaitu 30 detik dan system akan menghasilkan suatu nilai yang disebut *distance*. Kemudian nilai distance ini disimpan dan dicatat hasil perhitungannya secara manual oleh penulis sebagai satu nilai yang disebut *distance* yang digunakan untuk menentukan nilai *threshold* untuk penentuan tingkat kepadatan lalu lintas.

3.4 Peentuan *Threshold*

Tahapan selanjutnya adalah *thresholding* atau penentuan nilai ambang batas untuk menentukan tingkat kepadatan lalu lintas pada suatu daerah pemantauan. Dalam hal ini nilai *threshold* yang ditentukan oleh penulis adalah 190.000.000 . Penentuan nilai *threshold* dilihat dengan cara melihat dan membandingkan hasil persamaan SAD (nilai *distance*) seluruh frame selama 30 detik pada saat *capture* otomatis pada saat kamera di arahkan pada jalan untuk mendeteksi pergerakan secara *real time*. Proses ini dilakukan berulang-ulang sebagai proses data latih tepatnya 20 kali percobaan pada kondisi macet di pagi hari, 20 kali percobaan pada kondisi lancar di pagi hari, 20 kali percobaan pada kondisi macet di siang hari, 20 kali percobaan pada kondisi lancar di siang hari, 20 kali percobaan pada kondisi macet di malam hari, 20 kali percobaan pada kondisi lancar di malam hari dengan total ciri latih 120 nilai distance untuk mendapatkan nilai *threshold* yang cocok. Sehingga diperoleh nilai *threshold* yang cocok. Nilai threshold yang ditentukan oleh penulis adalah 190.000.000 karena nilai 190.000.000 adalah pembulatan nilai tengah dari nilai *distance* yang didapat dari 120 uji latih aplikasi.

Tabel 3 1 Penentuan Threshold

PARAMETER	THRESHOLD
Pagi	279807174,7
Siang	162202827,2
Malam	138891545,1
Jumlah	580901547
THRESHOLD RATA-RATA	**193633849**
PEMBULATAN NILAI THRESHOLD	**190.000.000**

Pada tabel 3.1 diketahui dengan total ciri latih 120 nilai distance untuk mendapatkan nilai *threshold* yang cocok. Penentuan Threshold dilakukan dengan menjumlahkan nilai Threshold pada kondisi pagi, siang dan malam hari dan diambil ambang batas diantara ketiganya sehingga didapatkan nilai Threshold terbaik. Nilai threshold terbaik yang ditentukan oleh penulis adalah 190.000.000 karena nilai 190.000.000 adalah pembulatan nilai tengah dari nilai *distance* yang didapat dari 120 data latih aplikasi.

3.5 *State* (Hasil)

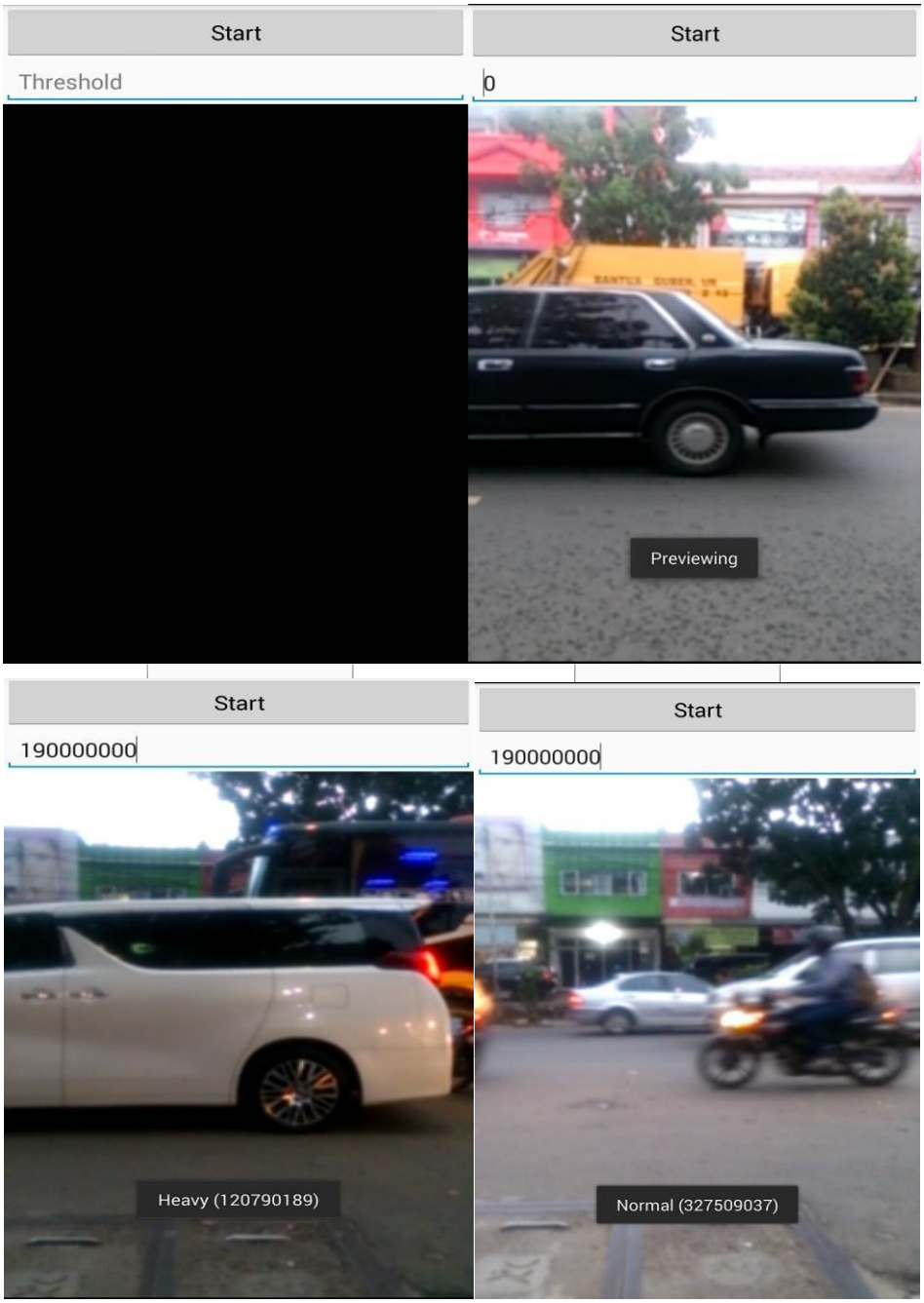
Pada tahap ini, nilai distance data latih yang didapat dari frame differencing yang merupakan hasil ekstraksi ciri SAD digunakan untuk mengklasifikasi kondisi ruas jalan berdasarkan *video processing*, yaitu dengan menginput nilai Threshold pada aplikasi dan membandingkan nilai *threshold* dengan nilai distance data uji. Klasifikasi video ini pada dasarnya dikelompokkan menjadi 2 kelas, yaitu macet dan lancar. Pada tahap latih hasil nilai distance dari ekstraksi ciri disimpan secara manual dengan cara dicapture dan dicatat oleh penulis bukan berbentuk *database* pada system. Hasil nilai distance data uji dibandingkan dengan *threshold* kemudian diklasifikasikan pada aplikasi.

Parameter yang diubah adalah intesintas cahaya data video pada saat proses *recording*. Pada tahap klasifikasi akan dihitung :

- Jika *distance < threshold* maka hasilnya HEAVY TRAFFIC
- Jika *distance > threshold* maka hasilnya NORMAL TRAFFIC

3.4 Model Aplikasi

Langkah terakhir dari aplikasi ini adalah menampilkan output hasil pendeteksian. Sistem aplikasi untuk mendeteksi kemacetan yang sudah dirancang menggunakan *software* Android Studio versi 2.2.2 sebagai *interface* pembuatan program. Berikut pada gambar adalah tampilan dari aplikasi pendeteksi kemacetan menggunakan metode *frame difference*



Gambar 3 .1 Model Aplikasi

3.5 Performansi Sistem

Untuk mengevaluasi performansi sistem yang dibahas, dilakukan pengujian terhadap menggunakan software andoid studio berdasarkan metode *frame difference* untuk mengidentifikasi kemacetan. Pengujian ini dilakukan untuk mengetahui kelebihan dan kekurangan sistem. Performansi sistem diukur berdasarkan parameter sebagai berikut :

1. Akurasi Sistem

   Akurasi merupakan ukuran ketepatan sistem dalam mengenali masukan yang diberikan sehingga menghasilkan keluaran yang benar. Secara matematis dapat dituliskan sebagai berikut :

$$\text{Akurasi} = \frac{\text{Jumlah Data Benar}}{\text{Jumlah Seluruh Data}} \times 100\% \quad (3.1)$$

3.6 Proses Penanaman Software ke Android

Proses selanjutnya yang dilakukan dalam perancangan aplikasi deteksi kemacetan setelah dilakukan proses pengujian performasi kerja aplikasi. Setelah dinyatakan aplikasi deteksi kemacetan layak dan memenuhi syarat dari berbagai hal, akan dilakukan proses pemindahan atau penanaman aplikasi deteksi kemacetan kedalam sebuah piranti dengan platform Android. Tahap-tahapnya adalah :

1. Pembuatan sistem menggunakan bahasa pemprogaman Android Studio.
2. Lakukan Running yang merupakan tools yang terdapat pada Android Studio yang juga sebagai emulator.
3. Setelah dilakukan Running, akan muncul sebuah shourcut aplikasi deteksi kemacetan pada piranti Android
4. Akan muncul pada piranti Android dan dapat langsung digunakan pengguna (user).

4. Analisis dan Hasil Pengujian

Tahap pengujian dilakukan dalam tiga buah skenario untuk mengetahui sejauh mana performansi yang dihasilkan oleh sistem dengan mengubah masing-masing parameter dari metode yang digunakan. Pada skenario pertama dilakukan perekamanan video pada satu sisi jalan dengan melakukan 20 kali percobaan secara real time untuk masing-masing jalan yang macet dan yang tidak macet pada pagi, siang dan malam hari dan mencatat *distance* yang didapat. Selanjutnya pada skenario kedua berdasarkan data yang di catat dari berapa saja *distance* yang didapat untuk masing-masing percobaan jalan yang macet dan tidak macet penulis akan membandingkan hasil distance dari data tersebut dan menentukan nilai *threshold* yang terbaik. Pada skenario ketiga dilakukan pengujian yang sesungguhnya pada sistem dengan melakukan masing-masing 10 kali percobaan pada jalan yang macet dan jalan yang tidak macet dengan *threshold* yang telah ditentukan oleh penulis untuk menentukan performansi dan akurasi sistem. Pengujian sistem dilakukan dengan merubah parameter intensitas cahaya yang digunakan pada proses pengujian. Masing-masing pengujian memiliki hasil yang direpresentasikan dalam bentuk tabel dan kesimpulan dalam bentuk grafik.

4.1 Hasil Pengujian 1

Pada pengujian skenario pertama, penulis akan melakukan 20 kali pengujian pengambilan video pada satu sisi jalan dengan menggunakan metode frame difference yang dilakukan pada Pagi hari dengan Nilai Threshold yang ditentukan oleh penulis adalah 190.000.000 yang telah didapatkan dari perhitungan distance pada data latih. Pada tabel 4.1 merupakan hasil proses dari pengujian tersebut dan mendapatkan nilai akurasi.

TABEL 4 .1 Hasil Pengujian 1

PERCOBAAN	PARAMETER	THRESHOLD	CURRENT CONDITION	DISTANCE	RESULT
1	PAGI	190000000	LANCAR	470569687	TRUE
2	PAGI	190000000	LANCAR	224788906	TRUE
3	PAGI	190000000	LANCAR	290723996	TRUE
4	PAGI	190000000	MACET	120790189	TRUE
5	PAGI	190000000	MACET	112723173	TRUE
6	PAGI	190000000	MACET	139187198	TRUE
7	PAGI	190000000	MACET	139844408	TRUE
8	PAGI	190000000	MACET	168782000	TRUE
9	PAGI	190000000	LANCAR	556135931	TRUE
10	PAGI	190000000	LANCAR	444183045	TRUE
11	PAGI	190000000	LANCAR	222997130	TRUE
12	PAGI	190000000	MACET	243345735	FALSE
13	PAGI	190000000	MACET	113559283	TRUE
14	PAGI	190000000	MACET	227408759	FALSE
15	PAGI	190000000	MACET	183068344	TRUE
16	PAGI	190000000	LANCAR	305772952	TRUE
17	PAGI	190000000	LANCAR	327509037	TRUE
18	PAGI	190000000	LANCAR	242209344	TRUE
19	PAGI	190000000	LANCAR	378755716	TRUE
20	PAGI	190000000	MACET	120790189	TRUE

Dapat dilihat pada tabel 4.1 bahwa tabel tersebut merupakan hasil proses aplikasi dengan perhitungan menggunakan *frame difference* dan menggunakan nilai Threshold sebesar 190000000. Hasil akurasi pada pengujian ini adalah 90%.

4.2 Hasil Pengujian 2

Pada pengujian skenario kedua, penulis akan melakukan 20 kali pengujian pengambilan video pada satu sisi jalan dengan menggunakan metode frame difference yang dilakukan pada siang hari dengan Nilai Threshold yang ditentukan oleh penulis adalah 190.000.000 yang telah didapatkan dari perhitungan distance pada data latih. Pada tabel 4.2 merupakan hasil proses dari pengujian tersebut dan mendapatkan nilai akurasi.

Tabel 4 .2 Hasil pengujian 2

PERCOBAAN	PARAMETER	THRESHOLD	CURRENT CONDITION	DISTANCE	RESULT
1	SIANG	190000000	LANCAR	217700247	TRUE
2	SIANG	190000000	LANCAR	230821964	TRUE
3	SIANG	190000000	LANCAR	155051490	FALSE
4	SIANG	190000000	LANCAR	120912537	FALSE
5	SIANG	190000000	LANCAR	213184747	TRUE
6	SIANG	190000000	MACET	144436654	TRUE
7	SIANG	190000000	MACET	153477628	TRUE
8	SIANG	190000000	LANCAR	239469021	TRUE
9	SIANG	190000000	LANCAR	289377329	TRUE
10	SIANG	190000000	LANCAR	250231595	TRUE
11	SIANG	190000000	LANCAR	196927304	TRUE
12	SIANG	190000000	LANCAR	222578930	TRUE
13	SIANG	190000000	MACET	138260627	TRUE
14	SIANG	190000000	MACET	149414196	TRUE
15	SIANG	190000000	MACET	166110931	TRUE
16	SIANG	190000000	MACET	171181434	TRUE
17	SIANG	190000000	MACET	120623033	TRUE
18	SIANG	190000000	MACET	276203725	FALSE
19	SIANG	190000000	MACET	178625092	TRUE
20	SIANG	190000000	MACET	146807260	TRUE

Dapat dilihat pada tabel 4.2 bahwa tabel tersebut merupakan hasil proses aplikasi dengan perhitungan menggunakan *frame difference* dan menggunakan nilai Threshold sebesar 190000000. Hasil akurasi pada pengujian ini adalah 85%.

4.3 Hasil Pengujian 3

Pada pengujian skenario ketiga, penulis akan melakukan 20 kali pengujian pengambilan video pada satu sisi jalan dengan menggunakan metode frame difference yang dilakukan pada malam hari dengan Nilai Threshold yang ditentukan oleh penulis adalah 190.000.000 yang telah didapatkan dari perhitungan distance pada data latih. Pada tabel 4.3 merupakan hasil proses dari pengujian tersebut dan mendapatkan nilai akurasi.

Tabel 4 .3 hasil pengujian 3

PERCOBAAN	PARAMETER	THRESHOLD	CURRENT CONDITION	DISTANCE	RESULT
1	MALAM	190000000	MACET	189831556	TRUE
2	MALAM	190000000	LANCAR	211170192	TRUE
3	MALAM	190000000	LANCAR	261044225	TRUE
4	MALAM	190000000	LANCAR	214068856	TRUE
5	MALAM	190000000	LANCAR	285494236	TRUE
6	MALAM	190000000	MACET	229358235	FALSE
7	MALAM	190000000	LANCAR	242082452	TRUE
8	MALAM	190000000	LANCAR	264807029	TRUE
9	MALAM	190000000	LANCAR	282036252	TRUE
10	MALAM	190000000	MACET	140809829	TRUE
11	MALAM	190000000	MACET	113348335	TRUE
12	MALAM	190000000	MACET	162242048	TRUE
13	MALAM	190000000	MACET	250824701	FALSE
14	MALAM	190000000	MACET	165197508	TRUE
15	MALAM	190000000	LANCAR	150181725	FALSE
16	MALAM	190000000	LANCAR	229282148	TRUE
17	MALAM	190000000	MACET	100499064	TRUE
18	MALAM	190000000	MACET	111206297	TRUE
19	MALAM	190000000	MACET	328479901	FALSE
20	MALAM	190000000	LANCAR	137138612	FALSE

Dapat dilihat pada tabel 4.3 bahwa tabel tersebut merupakan hasil proses aplikasi dengan perhitungan menggunakan *frame difference* dan menggunakan nilai Threshold sebesar 190000000. Hasil akurasi pada pengujian ini adalah 75%.

4.4 Hasil Pengujian 4

Pada pengujian skenario kelima ini merupakan perbandingan hasil terbaik dari pengujian skenario 1, 2 dan 3. Dimana kita akan mencari akurasi yang paling baik dalam pengujian di aplikasi ini.

TABEL 4 .4 Hasil Pengujian Skenario 4

JENIS PARAMETER	NILAI THRESHOLD	AKURASI
PAGI	190000000	90%
SIANG	190000000	85%
MALAM	190000000	75%

Dari hasil pengujian 4 kita bisa lihat pada tabel 4 yang menunjukan hasil bahwa, untuk akurasi terbaik dalam aplikasi ini yaitu perhitungan deteksi kemacetan paling bagus dilakukan pada pagi hari dengan video berdurasi 30 detik yaitu dengan akurasi 90%. Sedangkan pada siang dan malam hari menghasilkan akurasi 85% dan 75%.

5. Kesimpulan dan Saran

5.1 Kesimpulan

Berdasarkan hasil pengujian yang dilakukan pada sistem deteksi kemacetan menggunakan metode frame difference ini, didapatkan kesimpulan sebagai berikut:

1. Implementasi metode menggunakan *frame difference* mampu mendeteksi perubahan gerakan pada video yang menghasilkan suatu nilai
2. Pengujian sistem terhadap pengaruh intesitas cahaya bertujuan untuk mengetahui bagaimana sistem dapat menentukan kepadatan lalu lintas. Dari hasil pengujian intesitas cahaya pada pagi hari memberikan nilai akurasi tertinggi yaitu sebesar 90%. Pada kondisi siang hari akurasi sistem yang dihasilkan adalah 85% , pada malam hari sebesar 75%. Dari data ini dapat disimpulkan bahwa intesitas cahaya mempengaruhi tingkat akurasi sistem khususnya malam hari.
3. Implementasi system deteksi kemacetan berdasarkan video digital menggunakan metode frame difference dapat diaplikasikan ke dalam android versi 4.4 (kitkat).

5.2 Saran

Sistem deteksi kemacetan berdasarkan video digital menggunakan metode frame difference ini masih dapat dikembangkan, sehingga tingkat akurasi yang diperoleh lebih tinggi. Oleh karena itu, saran untuk pengembangan tugas akhir ini yaitu:

1. Pengujian dilakukan dengan parameter waktu yang lebih bervariasi
2. Pengujian dilakukan dengan nilai Threshold yang lebih bervariasi
3. Pengujian sistem dilakukan pada kondisi yang berbeda seperti pada kondisi hujan atau badai.
4. Mengambil data latih yang lebih banyak, agar nilai distance latih yang dimiliki semakin variatif sehingga pendeteksian nilai threshold pun lebih akurat.
5. Menggunakan metode yang berbeda untuk mendeteksi pergerakan agar dapat dibandingkan metode mana yang lebih baik.
6. Perancangan sistem yang dibuat lebih instant agar pengguna android lebih mudah untuk menggunakannya.
7. Tampilan aplikasi dibuat lebih menarik

Referensi

[1] “Citra” https://id.wikipedia.org/wiki/Citra (diakses pada tanggal 23 November).

[2] “Konsep Pengolahan Video” http://aldotbro.blogspot.sg/2014/08/konsep-pengolahan-video.html (diakses tanggal 30 November 2015).

[3] Muhammad Ihsan Zul, Widyawan, Lukito Edi Nugroho, “Deteksi Gerak dengan Menggunakan Metode Frame Differences pada IP Camera” , Teknik Elektro dan Teknologi Informasi, Universitas Gadjah Mada

[4] “Pengantar Pengolahan Citra” http://vinaaryantii.blogspot.sg/2012/09/pengantar-pengolahan-citra.html (diakses tanggal 28 November 2015)

[5]]“Pengenalan Android Studio” http://www.jadibaru.com/android/pengenalan-android-studio-2 (diakses pada tanggal 30 Maret 2016)