TUGAS AKHIR – TM141585

PENDETEKSIAN ARAH JALAN PADA GPS GOOGLEMAPS SEBAGAI NAVIGASI MOBIL TANPA PENGEMUDI

HENDIJANTO DIAN PRADIKTA
NRP: 2111 100 115

Dosen Pembimbing
Arif Wahjudi,ST,MT,PhD.

Jurusan Teknik Mesin
Fakultas Teknlogi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

(*Halaman ini sengaja dikosongkan***)**

FINAL PROJECT – TM141585

DIRECTION OF DETECTION ON ROAD GPS NAVIGATION IN GOOGLE MAPS GPS FOR AUTONOMOUS CAR

HENDIJANTO DIAN PRADIKTA
NRP: 2111 100 115

Advisor
Arif Wahjudi,ST,MT,PhD.

Jurusan Teknik Mesin
Fakultas Teknlogi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2016

(*Halaman ini sengaja dikosongkan***)**

LEMBAR PENGESAHAN

PENDETEKSIAN ARAH JALAN PADA GPS GOOGLEMAPS SEBAGAI NAVIGASI MOBIL TANPA PENGEMUDI

TUGAS AKHIR

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Teknik
pada
Program Studi S-1 Jurusan Teknik Mesin
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Oleh:
HENDIJANTO DIAN P.
Nrp. 2111 100 115

Disetujui oleh Tim Penguji Tugas Akhir:

1. Arif Wahjudi, S.T, M.T, Ph.D. .....(Pembimbing)
   NIP. 197303222001121001
2. Prof. Dr. Ing. Ir. I Made Londen Batan, M.Eng
   .....(Penguji I)
   NIP. 195811061986011001
3. Ir. Bambang Pramujati, M.Sc., Eng, Ph.D. .....(Penguji II)
   NIP. 196912031994031001
4. Dinny Harnany, S.T, M.Sc. .....(Penguji III)
   NIP. 2100201405001

SURABAYA
Juli 2016

(Halaman ini sengaja dikosongkan)

PENDETEKSIAN ARAH JALAN PADA GPS GOOGLEMAPS SEBAGAI NAVIGASI MOBIL TANPA PENGEMUDI

Nama : **Hendijanto Dian Pradikta**
NRP : **2111100115**
Jurusan : **Teknik Mesin FTI-ITS**
Dosen Pembimbing : **Arif Wahyudi ST,MT,Ph.D.**

ABSTRAK

Pada era globalisasi saat ini perkembangan teknologi banyak mempermudah kehidupan kita. Hasil penelitian terutama dibidang otomotif yang terus menerus menghasilkan teknologi yang mempermudah kehidupan manusia sepeti teknologi kemudi yang adaktif, peringatan akan tabrakan, hingga sistem yang membantu kendaraan untuk parkir sendiri. Selain itu perkembangan *smartphone* berbasis *android* mengalami perkembangan yang pesat terutama dalam visualisasi tampilan pada *handphone* yang *simple* sehingga memudahkan pengguna. Dengan permasalahan yang ada diperlukan suatu program yang cukup sederhana yang membantu dalam proses navigasi dengan menggunakan image processing. *Image processing* adalah proses mengubah citra masukan dan mengolahnya agar kualitasnya menjadi lebih baik. *Image prossesing* tersebut dapat membantu dalam memproses dan menangkap gambar dari GPS pada *hanphone* sehingga dapat digunakan untuk mendeteksi jalan berdasarkan dari warna jalan yang ada.

Dalam tugas akhir ini pendeteksian serta pencarian sudut belok pada mobil tanpa *driver* berbasis *image processing* dimulai dengan membuat program pendeteksi navigasi sebagai pengarah mobil. Kemudian mendeteksi warna jalan yang berwarna merah,kuning, dan biru. Setelah itu merancang dan membuat

program titik dan garis acuan serta melihat hasil dari program melalui titik dan garis yang terbentuk apakah posisinya sudah sesuai dengan posisi *centroid* segitiga dan titik pada jalan.

Hasil yang diperoleh dari tugas akhir ini adalah dapat mendeteksi jalan dengan tiga warna dapat dilakukan secara bersamaan dengan fungsi bitwisedengan range nilai HSV dari setiap warna yaitu(46-255,180-255,109-255)untuk jalan warna biru, (14-107,158-255,101-255)untuk jalan warna kuning, dan (0-0,255-255,71-255)untuk jalan warna merah. Hasil nilai validasi sudut jalan dengan membandingkan data sudut hasil image processing dengan sudut kompas dengan metode pair sample t-test diterima karena didapatkan nilai t secara teori yaitu sebesar 0,499 lebih kecil dibandingkan dengan nilai t pada table yang sebesar 2,045.

Kata Kunci-Autonomous car, Image Processing, Smartphone

DETECTION OF DIRECTION ON ROAD GPS NAVIGATION IN GOOGLE MAPS GPS FOR AUTONOMOUS CAR

Name : Hendijanto Dian Pradikta
NRP : 2111100115
Department : Teknik Mesin FTI-ITS
Advisor : Arif Wahyudi ST,MT,Ph.D.

ABSTRACT

In modern era, the development of technology has facilitated our life. The result of experiment in automotive has produced many technologies that improve our life such as (adaptive cruise control (ACC), collision warning alarm, and automatic parking system). In addition, the development android-based smartphone experiences a dramatic growth in terms of the display of simple mobile phone so that it eases its users. With the existing problem, a simple program that is able to help navigation process by using image processing is required. Image Processing is a process which changes an input image and turns it into a better image. This process is beneficial to process and capture an image from GPS in mobile phone so that it can be implemented to detect road based on its colour.

In this final project, navigation detector program was the first step to be carried out in order to detect and find out (sudut belok) from a car without using driver that implemented image processing. After that, red, yellow, and blue-colored road was recognized. Then, the program of line and dot reference was designed and established. The result was examined through the

formed dots and lines to verify whether the position was match with the position of the centroid of the triangle and the dots in the road.

Based on this experiment, road detection using three colors can be done simulatenously with bitwise function with the range of HSV value from (46-255,180-255,109-255) for blue road, (14-107,158-255,101-255) for yellow, and (0-0,255-255,71-255) for red road. The result of validation value of road's angle by comparing the angle from image processing with compass which employs pair sample t-test method is acceptable as the theoritical value of t is 0,499 times smaller compared to the value of t in the table which is around 2,045.

Key Word-Autonomous car, Image Processing, Smartphone

KATA PENGANTAR

Puji syukur penulis panjatkan atas kehadirat Tuhan Yang Maha Esa, atas berkat, rahmat dan hidayah-Nya maka penulis dapat menjalankan dan menyelesaikan penulisan laporan tugas akhir. Laporan tugas akhir ini merupakan persyaratan untuk memperoleh gelar sarjana teknik pada Jurusan Teknik Mesin Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember Surabaya

Dalam penyusunan laporan tugas akhirini, penulis mendapatkan banyak dukungan dan bantuan, oleh sebab itu penulis mengucapkan terima kasih kepada:

1. Allah SWT yang telah memberikan kesehatan, kemudahan, kelancaran dan perlindungan kepada penulis selama dalam penyelesaian penulisan laporan ini;
2. Orang tua terutama ibu dan keluarga penulis yang senantiasa mendukung aktivitas penulis terutama dalam menjalankan dan pengerjaan laporan tugas akhir;
3. Bapak Arif Wahjudi S.T, M.T, Ph.D. atas bimbingan, ketulusan, kesabaran, dan totalitas yang diberikan kepada penulis selama proses pengerjaan tugas akhir ini;
4. Bapak Ir. Bambang Pramujati S.T, M.Eng.Sc.,Ph.D, selaku ketua jurusan Teknik Mesin ITS dan Bapak Wahyu Wijanarko S.T, M.T., selaku dosen wali, dan seluruh dosen Teknik Mesin yang telah memberikan ilmu dan mendidik penulis selama ini;
5. Bapak Ir. Bambang Pramujati S.T, M.Eng.Sc.,Ph.D, Ibu Dinny Harnany S.T, M.Sc., dan Bapak Prof. Dr.Ing., I Made Londen Batan M. Eng, selaku dosen penguji yang telah membuka wawasan penulis dan memberikan banyak informasi penting untuk kemajuan tugas akhir.
6. Febriana Puspa Ayu Pasha, Valya Ika Dhanie, Dwi Anis Kusuma Wardhani, Mutafawiqin Rizqoni A,Heri, Fahmi H., Chandra dan seluruh teman-teman yang ada di lab

perancangan dan pengembangan produk yang telah membantu penulis dalam memberikan informasi.

Penulis menyadari bahwa tugas akhir ini masih banyak kekurangan. Oleh karena itu, dengan segala kerendahan hati saran dan kritik terhadap penulis sangatlah diperlukan demi menyempurnakan tugas akhir ini. Dan akhirnya semoga tugas akhir ini dapat berguna bagi semua pihak yang membutuhkan.

Surabaya, 15 Juli 2016

Hendijanto Dian P.
2111 100 115

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1.Latar Belakang

Pada era globalisasi saat ini kita dapat menikmati perkembangan teknologi yang dapat mempermudah kehidupan kita. Perkembangan teknologi mobil saat ini merupakan hasil penelitian yang kontinu yang awalnya dimulai dari kendaraan manual yang sepenuhnya dikemudikan oleh manusia yang pada akhirnya menjadi *AutoVehicle* yang dimana sebagian atau secara keseluruhan dari kendaraan tersebut dijalankan dengan program yang ada pada mobil itu sendiri. Dengan teknologi yang berkembang secara terus menerus saat ini kendaraan dapat menentukan sendiri keputusan sebagaimana manusia ketika berkendara. Beberapa teknologi yang sudah berkembang antara lain sistem peringatan akan tabrakan, control kemudi yang adaktif, sistem pengikut garis atau marka dan teknologi yang memungkinkan kendaraan untuk parkir sendiri [1]

Dewasa ini banyak sekali jenis-jenis mobil pintar yang berlalu-lalang di jalanan. Namun saat ini mobil pintar yang ada masih menggunakan sistem yang rumit serta mahal dan hanya dapad diakses melalui GPS yang perlu dibeli dan kemudian dilakukan penyetingan serta perlu dilakukan langganan berkala untuk terus mengakses jalan. Namun tidak dipungkiri perkembangan *smartphone* yang berbasis *android* yang memiliki akses GPS yang dapat diakses secara gratis dan mudah semakin berkembang. Tampilan visualisasi pada *handphone* yang *simple* atau sederhana dapat memudahkan pengguna dimana kami sebagai pengguna dengan mudah memproses dan mengolahnya menjadi suatu citra.

Pengolahan citra (*image processing*) adalah proses mengolah citra yang mengubah citra masukan menjadi citra dalam bentuk lain agar keluaran memiliki kualitas yang lebih baik dibandingkan kualitas citra masukan. Pengolahan citra memiliki berbagai macam manfaat diantaranya adalah untuk meningkatkan

kualitas citra, menghilangkan cacat pada citra, mengidentifikasi objek dan untuk menggabungkan dengan bagian citra yang lain. Dengan memanfaatkan teknologi tersebut, maka diharapkan adanya suatu aplikasi baru yang dapat menangkap suatu objek yang ada di depan kamera. Selain itu mampu mengidentifikasi dan mendeteksi jenis objek, serta melakukan *tracking* objek secara *real-time*.

Pendeteksian dan pembacaan rute atau jalur pada *googlemaps* dapat dilakukan dengan menggunakan input baik berupa gambar maupun video. Pengenalan rute *googlemaps* dengan sebuah gambar adalah tahap untuk membaca rute tanpa harus melakukan *tracking*. Sedangkan objek deteksi secara *real-time*,merupakan langkah dimana pembacaan rute *googlemaps* dengan menggunakan *tracking*.

Dalam tugas akhir ini, objek yang akan digunakan merupakan jalan pada *googlemaps* yang akan diidentifikasi dengan satu kamera secara *real-time* berdasarkan pembagian citra warna RGB(*red,green,blue*). Identifikasi dilakukan untuk menentukan arah dan sudut jalan sebagai representasi arah jalan yang akan dilalui mobil. Sedangkan hasil dari pendeteksian tersebut dianalisa apakah mobil bergerak sesuai sudut yang ditunjukan jalan.

1.2. Perumusan Masalah

Rumusan masalah yang dapat ditentukan berdasarkan latar belakang tugas akhir ini adalah sebagai berikut:

1. Bagaimana mendeteksi jalan berdasarkan warna jalan serta arah jalan yang akan dilewati oleh mobil pada GPS dari *Handphone* secara real-time?

1.3. Batasan Masalah

Batasan masalah yang digunakan sebagai pembatas untuk mempermudah pengerjaan tugas akhir ini antara lain:

1. Warna jalan pada GPS yang akan diambil adalah biru, merah, dan kuning sebagaimana yang ada pada navigasi googlemaps

2. Perlu adanya proses *re-center* untuk mengatur posisi jalan
3. Cahaya ha
4. Pengambilan data dilakukan pada siang hari

1.4. Tujuan Penelitian

Tujuan dari tugas akhir ini adalah sebagai berikut:

1. Mengetahui cara mendeteksi jalan sesuai program yang dibuat serta mengetahui sudut dan arah jalan yang akan dilewati mobil dengan melakukan visualisasi citra secara *Real-Time*

1.5. Manfaat Penelitian

Manfaat yang dapat diperoleh dari tugas akhir ini adalah sebagai berikut:

1. Untuk membuat sistem yang dapat digunakan untuk mengetahui arah belok pada mobil
2. Dasar dari aplikasi dapat digunakan sebagai acuan mobil ketika mobil bergerak sesuai dengan arah dari rute navigasi pada *googlemaps*

(*Halaman ini sengaja dikosongkan***)**

BAB II
DASAR TEORI

2.1 Pengenalan *Smart Car*

Ketika teknologi modern pada mobil tumbuh secara cepat, maka semakin banyak tantangan yang muncul. Mulai dari kemampuan proses yang sangat canggih dan bermacam-macam aksesoris yang menunjang kinerja mobil itu sendiri. Akan tetapi batas kemampuan memproses pada computer yang ada pada mobil pintar secara secara terus – menerus akan menyebabkan *bottlenecks* pada control mobil. Isi dari aplikasi pada mobil pintar juga mempengaruhi kinerja dari mobil itu sendiri sehingga diperlukan aplikasi yang cukup simpel namun mampu menjalankan program sesuai yang diharapkan. Selain itu aplikasi software yang digunakan juga harus mampu digunakan pada banyak *smart car* ,dimana pun mobil berada dan kapanpun sang pengendara mengendarai kendaraan tersebut.[2]

Perkembangan teknologi mobil saat ini adalah hasil penelitian secara kontinu antara mesin konvensional, kendaraan manual yang dikemudikan sepenuhnya oleh manusia dan *AutoVehicles* yang dimana sebagian atau secara keseluruhan dari kendaraan tersebut dijalankan dengan program yang ada pada mobil itu sendiri dan memungkinkan tidak dibutuhkan pengemudi sama sekali. Dengan teknologi yang berkembang secara terus menerus saat ini kendaraan dapat menentukan sendiri keputusan sebagaimana manusia ketika berkendara. Beberapa teknologi yang sudah berkembang antara lain sistem peringatan akan tabrakan, kontrol kemudi yang adaktif, sistem pengikut garis atau marka dan teknologi yang memungkinkan kendaraan untuk parkir sendiri.[1]

Pada perkembangannya mobil dibagi menjadi beberapa tingkat. Tingkatan-tingkatan tersebut disesuaikan terhadap kecanggihan, keamanan, dan kenyamanan. Pembagian tersebut dilakukan bertujuan untuk meningkatkan kinerja mobil, kenyamanan pengguna, tingkat keamanan pada sistem yang ada

didalam mobil, harga jual dari mobil dan prestasi yang ditunjukan oleh perusahaan pembuat mobil tersebut.. Sebagaimana pembagian tingkatan – tingkatan yang telah dibagi oleh National Highway Traffic Safety Administration sebagai berikut:[9]

- Level 0 (no automaton): sang pengemudi sebagai pengontrol secara penuh mengatur fungsi kendaraan meliputi pengereman, fungsi kemudi, pedal gas, transmisi pada saat yang bersamaan, dan diperlukan kemampuan respon yang sangat bagus dari si pengendara untuk mengawasi keadaan sekitar dan demi keamanan kendaraan sendiri. Pada masa ini penemuan teknologi untuk kendaraan otomotif sangat sedikit dan masih sulit karena teknologi untuk penunjang perkembangannya masih belum ada. Untuk melakukan produksi mobil pun juga masih sulit dan harga untuk mobil saat itu masih mahal jika dibandingkan dengan harga mobil saat ini. Masyarakat yang memiliki mobil pada saat itu masih jarang dan dapat dikatakan orang yang kaya. Tenaga yang dihasilkan juga masih kecil jika dibandingkan mobil saat ini sehingga, kecepatan mobil masih lamban.[9]

Gambar 2.1 Gambar Mobil No Automaton[3]

Gambar 2.1 menunjukan bahwa mobil pada generasi pertama yang tidak memiliki sistem pengontrol otomatis sehingga pengemudi adalah satu-satunya pengatur dari input kendaraan. Gambar diatas merupakan gambar mobil ford

tipe T yang banyak digunakan di Amerika dengan rentang produksi yaitu pada tahun 1908 hingga 1927. Dengan jumlah produksi total sekitar empat belas juta unit. Mobil tersebut menggunakan transmisi berbentuk planetary gear dengan dua tingkat percepatan.[3]

- Level 1 (otomasi pada fungsi sepesifik) otomasi pada level ini melibatkan satu atau beberapa fungsi control secara spesifik , jika beberapa fungsi dibuat menjadi otomatis ,mereka akan bergerak sendiri terhadap yang lain sehingga sang pengemudi mampu mengontrol kendaraan lebih baik , meskipun membutuhkan respon yang baik untuk keamanan penggunaan , namun pengemudi dapat memilih kemampuan kontrol utama seperti ACC(Adaptive Cruice Control) meskipun masih terbatas. Kendaraan secara otomatis dapat mengasumsikan batas kemampuan pada kontrol utama.[9]

Gambar 2.2. Mobil Dengan Fungsi Otomatis[12]

Gambar 2.2 memperlihatkan stering dari salah satu pabrikan mobil yang menggunakan system control spesifik yang mengatur dari salah satu kinerja mobil. Pada gambar ditunjukan dua bua shift pedal yang digunakan sebagai pengatur percepatan secara otomatis sebagai pengganti fungsi kopling untuk pergantian percepatan pada mobil. Selain itu fungsi pedal tersebut juga digunakan sebagai pengatur fungsi kecepatan statis pada *cuirse control*.

- Level 2(kombinasi fungsi otomatis): pada level ini paling sedikitnya melibatkan pengotomatisan dua fungsi kontrol

utama sehingga pengemudi tidak perlu mengontrol fungsi tersebut. Kendaraan pada level ini mampu berbagi wewenang ketika sang pengemudi mengaktifkan fungsi otomatis pada kontrol utama saat situasi ketika mengemudi terbatas. Sang pengemudi masih bertanggung jawab untuk memonitor keadaan jalan dan keamanan operasi, dan diharapkan mampu mengkontrol kendaraan sepanjang waktu maupun saat sesaat. System dapat melepas control tanpa pemberitahuan sehingga pengemudi harus selalu siaga untuk mengkontrol kendaraan agar aman.[9]

Gambar 2.3 Mobil Dengan Teknologi Fungsi Otomatis Kombinasi[4]

Gambar 2.3 menunjukkan gambar dari Nissan Juke consept yang akan dirilis pada tahun 2018 di Frankfurt dengan konsep *futuristic design* serta mengusung teknologi pengaturan *hybrid* sebagai outputannya. Teknologi pengaturan *hybrid* sendiri merupakan pengatiran keluaran tenaga dari dua daya yang didapat dari motor listrik serta dengan mesin berbahan bakar bensin sebagai penggerak utamanya.[4]

- Level 3(*limited self-driving automation*): Kendaraan pada level semi otomatis ini membolehkan pengemudi untuk memberikan control secara penuh dari keseluruhan fungsi dengan tingkat keamanan kritis ketika ada pada keadaan lalu

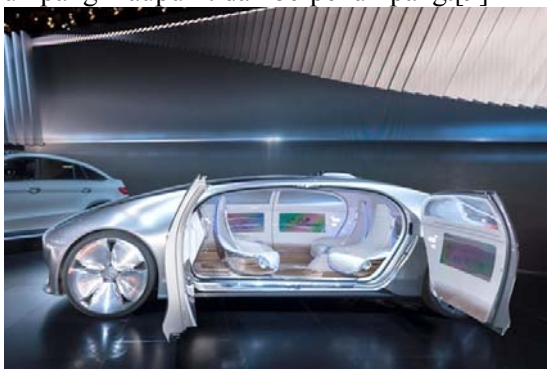
lintas tertentu atau kondisi yang memungkinkan kepada sistem. Apabila keadaan jalan berubah menjadi berbahaya maka sistem mengembalikan fungsi kontrol kepada sang pengemudi. Proses tersebut terjadi ketika keadaan dari sistem yang terbatas dan terlalu berat sehingga tidak mampu mengkontrol dari perubahan kondisi lingkungan yang rumit seperti contohnya saat macet, terdapat suatu kecelakaan lalu lintas, pengemudi lain yang menyalip secara tiba – tiba, pejalan kaki yang menyebrang, parker secara parallel, tanjakan yang curam dan lain sebagainya..[9]

Gambar 2.4 Mobil Tesla s Generasi Pertama Dengan System *Limited Automation[5]*

Gambar 2.4 menunjukan salah satu contoh dari mobil yang menggunakan teknologi sistem *Limited Automation.* pengemudi dapat mengatur daripada fungsi kemudi baik pengaturan stering,pengaturan tenaga maupun navigasi namun hanya terbatas dan hanya bisa dijalankan ketika keadaan jalan tidak begitu ramai serta lalu lintas yang lancar sehingga proses yang dilakukan system tidak mengalami kendala.

- Level 4(*full self-driving automation*): kendaraan didesain agar mampu mengatur seluruh keamanan fungsi kemudi dengan tingkat keamanan kritis dan pengawasan jalur pada keseluruhan perjalanan. Suatu desain antisipasi dari

pengemudi akan meningkatkan input pada navigasi serta input tujuan, namun kontrol yang ada tidak selalu dapat diharapkan disepanjang perjalanan. Baik pada kendaraan berpenumpang maupun tidak berpenumpang.[9]



A

Gambar 2.5. (A) Eksterior dan (B) Interior Dari Mercedes F-015[6]

Gambar (A) dan (B) memperlihatkan desain dari Mercedes F-015 yang menggunakan *full self-driving automation* sebagai pengatur daripada mobil. Dengan desain yang futuristic dan minimalis serta menggunakan bahan pilihan yang memberikan kesan mewah. Dengan menggunakan system *full self-driving automation* pengemudi dan penumpang mampu melakukan aktivitas lain

didalam mobil sebagaimana diperlihatkan pada gambar yang menunjukan pengemudi dan penumpang melakukan rapat.

Teknologi *Autonomos vehicle* dewasa ini semakin menyita perhatian publik terutama setelah google melibatkan ratusan hingga ribuan *autonomous* mobil yang digunakan untuk mengambil pemetaan jalan diseluruh dunia yang mampu menyita perhatian media dan google mampu mendemonstrasikan teknologi ini dengan perkembangan yang luar biasa. Setiap pabrikan mobil saat ini berlomba – lomba untuk melakukan *research* dalam hal ini menciptakan mobil *autonomous*

2.1.1 Google Autonomous Car

Untuk memahami lebih lanjut mengenai teknologi mobil autonomous, saya menggunakan contoh google autonomous car. Google autonomous car merupakan mobil autonomous yang dapat mengendalikan dirinya sendiri untuk memudahkan transportasi manusia. Sebuah kendaraan autonomous dapat mendeteksi lingkungan di sekitarnya dan dapat menavigasi dirinya sendiri dengan bantuan sensor-sensor yang ada padanya. Pada pengembangannya saat ini, manusia/operator dapat mengatur destinasi yang diinginkan tanpa perlu melakukan kegiatan mekanis seperti memindahkan gigi atau menginjak kopling mobil untuk mengoperasikan mobil itu. Autonomous car yang digunakan google adalah Toyota Prius yang dimodifikasi sedemikian rupa untuk menyesuaikan dengan konsep mobil autonomous yang diinginkan.

Pada google autonomous car, terdapat banyak sensor yang masing-masing berguna sebagai sarana navigasi mobil dan detektor keadaan lingkungan berkendara seperti lalu lintas, pejalan kaki, atau pengendara lainnya. Berikut adalah contoh sensor-sensor yang digunakan pada google autonomous car:[10]

1. LIDAR
   Lidar merupakan suatu sensor yang terletak di atas atap mobil yang melakukan gerakan rotasi 360 derajat ketika

mobil berkendara secara otomatis. Cara kerja sensor ini adalah melakukan gerakan memutar untuk mendapatkan visualisasi lingkungan sekitar dalam radius 200 kaki. Dari hasil visualisasi tersebut, computer didalam mobil akan memproses data yang diperoleh untuk mendapatkan peta secara 3D secara real time pada lingkungan disekitar mobil.[10]

2. Position Estimator
   Sensor ini diletakkan pada sekitar ban kendaraan. Cara kerja dari position Estimator adalah mendeteksi arah gerakan roda agar sesuai dengan instruksi yang diberikan.[10]

3. Video Camera
   Video kamera digunakan sebagai mata kendaraan tersebut,dimana fungsi utamanya adalah untuk mendeteksi objek-objek yang bergerak seperti halnya pejalan kaki atau orang yang menyebrang.[10]

4. Radar
   Radar berfungsi untuk menentukan suatu objek yang mendekat atau menjauhi kendaraan. Cara kerja dari radar ini hamper sama dengan sensor ultrasonik

Gambar 2.6 Pembacaan Sensor *Google Smart Car*

Gambar 2.6 memperlihatkan kinerja dari sensor *Google Smart Car*. Pada gambar diatas diperlihatkan garis yang berwarna hijau tua yang searah dengan mobil yang merupakan petunjuk dari rute yang akan dilalui maupun yang telah dilalui oleh mobil. Gambar berwarna hijau muda yang terlihat menyerupai gambar manusia serta keadaan sekitar mobil merupakan hasil pembacaan dari sensor LIDAR yang memberitahukan keadaan di sekitar mobil. Sedangkan linkaran berwarna oranye sendiri merupakan garis kayal yang digambarkan sebagai pancaran dari sensor lidar itu sendiri. Dan tulisan serta tanda pada bagian kanan atas merupakan petunjuk dari keadaan jalan yang sedang dilalui.[10]

2.2. OpenCV

OpenCV adalah sebuah *libraryopensource*untuk visi komputer. *Library* ini ditulis dengan bahasa C dan C++, serta dapat dijalankan dengan Linux, Windows, dan Mac OS X. OpenCV dirancang untuk efisiensi komputasional dan dengan fokus yang kuat pada aplikasi *real-time*.[13]

Salah satu tujuan OpenCV adalah untuk menyediakan infrastruktur visi komputer yang mudah digunakan yang membantu orang-orang dalam membangun aplikasi-aplikasi visi yang *sophisticated*dengan cepat. *Library* pada OpenCV berisi lebih dari 500 fungsi yang menjangkau berbagai area dalam permasalahan visi, meliputi inspeksi produk pabrik, pencitraan medis, keamanan, antarmuka pengguna, kalibrasi kamera, visi stereo, dan robotika. Karena visi komputer dan pembelajaran mesin seringkali berkaitan, OpenCV juga memiliki *MachineLearningLibrary*(MLL). *Sublibrary*ini berfokus pada pengenalan pola statistik dan *clustering*. MLL sangat berguna untuk tugas-tugas visi yang berada dalam misi inti OpenCV, tetapi MLL cukup umum digunakan untuk permasalahan pembelajaran mesin.

Lisensi *opensource*pada OpenCV telah distrukturisasi sehingga pengguna dapat membangun produk komersial menggunakan seluruh bagian pada OpenCV. Tidak ada kewajiban untuk meng-*opensource*produk tersebut atau untuk memberikan peningkatan ke domain publik. Sebagian karena peraturan lisensi liberal ini, maka terdapat komunitas pengguna dalam jumlah yang sangat besar, termasuk di dalamnya orang-orang dari perusahaan besar (seperti IBM,Microsoft, Intel, SONY, Siemens, dan Google) serta pusat-pusat penelitian (seperti Stanford, MIT, CMU, Cambridge, dan INRIA).[13]

Sejak peluncuran pertamanya pada Januari 1999, OpenCV telah digunakan pada banyak aplikasi, produk, dan usaha-usaha penelitian. Aplikasi-aplikasi ini meliputi penggabungan citra pada peta web dan satelit, *imagescanalignment*, pengurangan *noise*pada citra medis, sistem keamanan dan pendeteksian gangguan, sistem pengawasan otomatis dan keamanan, sistem inspeksi pabrik, kalibrasi kamera, aplikasi militer, serta kendaraan udara tak berawak, kendaraan darat,dan kendaraan bawah air.

OpenCV adalah singkatan dari *OpenComputerVision*, yaitu *libraryopensource*yang dikhususkan untuk melakukan pengolahan citra. Tujuannya adalah agar komputer mempunyai kemampuan yang mirip dengan cara pengolahan visual pada manusia. *Library*ini dibuat untuk bahasa C/C++ sebagai optimasi aplikasi *real-time*. OpenCV memiliki API (*ApplicationProgrammingInterface*) untuk pengolahan tingkat tinggi maupun tingkat rendah. Pada OpenCV juga terdapat fungsi-fungsi siap pakai untuk me-*load*, menyimpan, serta mengakuisisi gambar dan video.[13]

*Library*OpenCV (10) memiliki fitur-fitur sebagai berikut:

- Manipulasi data gambar (mengalokasi memori, melepaskan memori, menduplikasi gambar, mengatur serta mengkonversi gambar)
- *Image*/Video I/O (bisa menggunakan kamera yang sudah didukung oleh libraryini)

- Manipulasi matriks dan vektor, serta terdapat juga routinesaljabar linear (*products*, *solvers*, *eigenvalues*, SVD)
- Pengolahan citra dasar (penapisan, pendeteksian tepi, samplingdan interpolasi, konversi warna, operasi morfologi, histogram, piramida citra)
- Analisis struktural
- Kalibrasi kamera
- Pendeteksian gerakan
- Pengenalan objek
- GUI dasar (menampilkan gambar dan video, mengontrol *mouse* atau *keyboard*, *scrollbar*)
- *Imagelabelling*(garis, kerucut, poligon, penggambaran teks)

*Libraries*OpenCV menyediakan banyak algoritma visi komputer dasar, dengan keuntungan bahwa fungsi-fungsi tersebut telah diuji dengan baik dan digunakan oleh para peneliti di seluruh dunia. *Libraries*OpenCV juga menyediakan sebuah modul untuk pendeteksian objek yang menggunakan algoritma Viola Jones. [10]

2.2. Image processing

Gambar adalah salah satu bentuk atau cara manusia dalam mengirimkan dan berbagi informasi. Karena didalam sebuah gambar memiliki seribu kata. Gambar dapat memberikan kita informasi baik berupa jarak , ukuran dan hubungan dari setiap benda yang terdapat pada gambar tersebut. Manusia mampu menangkap informasi lebih baik dengan melihat gambar. Saat ini manusia banyak melakukan trobosan dalam memproses suatu data berupa gambar melalui image processing atau pengolahan citra. Pengolahan citra sendiri adalah setiap pengolahan sinyal dimana yang dijadikan input adalah gambar. Gambar yang diproses adalah suatu foto atau video yang diolah dengan melibatkan presepsi visual. Istilah Image processing secara umum diartikan sebagai proses mengolah gambar dua dimensi dengan menggunakan software pada computer sebagai media. Selain itu Image processing juga memiliki arti lain yaitu pengolahan citra digital

yang mencakup semua data dua dimensi. Dimana citra digital merupakan wujud dari suatu barisan bilangan nyata yang diwakili dengan bit-bit tertentu.[14]

Pada umumnya citra digital memiliki bentuk persegi atau persegi panjang , namun pada kasus tertentu terdapat juga yang berbentuk persegi enam atau heksagonal dengan memiliki panjang dan lebar tertentu. Ukuran ini biasanya dinyatakan dalam banyaknya *pixel* atai titik sehingga ukuran citra selalu bernilai bulat. setiap titik memiliki koordinat sesuai posisi dan dinyatakan dalam bilangan bulat positif yang biasa dimulai dari angka nol (0) atau satu (1) tergantung dari system yang digunakan dalam image processing. Format pengolahan citra yang sering digunakan adalah ; citra warna, gray scale, threshold, HSV(hue-saturation-value), Segmentasi Citra, Canny dan lain sebagainya. [7]

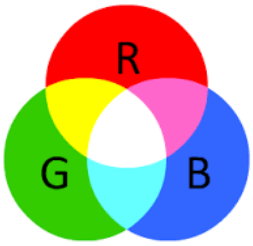
2.3.1. Citra Warna

Citra warna adalah cara untuk menspesifikasikan suatu warna tertentu dengan mendefinisikan suatu system koordinat kartesian, dan bagian tersebut dapat dibentuk menjadi suatu warna yang dapat dibentuk kedalam bentuk tertentu. Suatu warna yang dapat dispesifikasikan kedalam suatu model dengan salah satu titik dalam suatu ruangan yang didefenisikan. Masing – masing warna diharapkan kedalam standart tertentu antara lain yaitu ; RGB dan CMY.[14]

2.3.1.1 Model Warna RGB

Suatu citra dalam pemodelan RGB terdiri dari tiga bidang citra yang saling lepas. Warna utama yang digunakan adalah merah, hijau dan biru. Suatu warna dispesifikasi sebagai campuran dari beberapa komponen warna dasar atau warna utama. Warna dipresentasikan dalam suatu koordinat untuk membentuk warna baru, dan saling berhubungan dengan koordinat yang lain untuk membentuk sinar warna campuran. Dengan menggunakan warna utama merah, hijau, dan biru dapat dicampurkan atau dikombinasi

untuk membentuk warna sekunder warna yang dihasilkan akanmenghasilkan gradasi sesuai dengan persentase tingkat pencampuran warnanya. Dapat diambil contoh warna ungu yang diambil dari pencampuran warna merah dan biru namun, bila semakin banyak ditambahkan warna merah akan berubah menjadi warna magenta dan bila semakin banyak warna biru yang dicampurkan warna yang dihasilkan akan semakin mendekati warna biru. Untuk warna putih pada RGB diperoleh dari nilai maksimal dari ketiga dasar yang membentuk RGB. Apabila kita menginginkan warna sekunder lain seperti ; kuning yang berasal dari kombinasi (merah+hijau), cyan dari kombinasi (biru+hijau), magenta dari kombinasi (merah+biru) dan putih yang berasal dari kombinasi (merah+hijau+biru) maka kita perlu membandingkan tingkat persentase dari tiap warna dasar yang dicampurkan dan kemudian ditotal sehingga menghasilkan warna yang kita inginkan.. Model warna RGB biasa digunakan untuk televisi, monitor komputer dan video kamera. Dalam program ini menggunakan format warna RGB(Red Green Blue) sebagai citra inputan. Pada kode pemrograman yang dibuat menggunakan model warna dasar (RGB)Red Green and Blue sebagai warna dasar pembentuk gambar yang akan digunakan sebagai gambar input pada kode pemrograman tersebut.[14]



Gambar 2.7 Komponen Warna Dasar RGB[15]

Gambar 2.7 menunjukan pembagian dari warna dasar dari RGB yang menghasilkan beberapa warna skunder seperti kuning, magenta dan cyan. Selain itu dari kombinasi dari ketiga warna tersebut kita dapat menghasilkan warna putih.

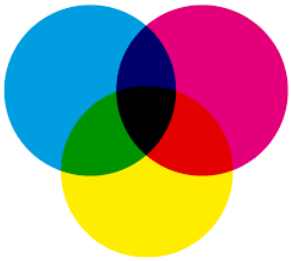
2.3.1.2 Model Warna CMYK

Permodelan warna pada CMYK (*Cyan Magenta Yellow Key*) didasarkan pada kualitas penyerapan cahaya dari tinta yang akan dicetak pada kertas. Warna dari pemodelan ini tidak begitu banyak digunakan karena sebagian besar alat menggunakan model pewarnaan RGB. Namun pada laporan ini dijabarkan sedikit tentang pengertian dari warna CMYK. Pada computer / laptop sendiri terdapat dua model warna yang dipakai yaitu CMYK dan RGB. Tingkatan warna dari CMYK ditentikan dari tingkat persentase dari tiap warna yang mendukung warna tersebut . Dapat diambil sampel warna putih yang menyebabkan tinta seolah tembus pandang, hal ini disebabkan oleh sebagian dari spectrum warnanya yang diserap sedangkan sebagian lainnya dipantulkan kembaki ke mata. Pada permodelan CMYK, masing – masing pixel diberi nilai untuk tiap pewarnaannya. Warna yang paling mendekati putih diberikan persentasi yang rendah, sedangkan warna yang mendekati hitam mempunyai persentasi yang tinggi. Pada image yang menggunakan CMYK sebagai dasar Pewarnaannya, warna putih akan muncul apabila semua komponen bernilai nol(0) persen. Sedangkan warna hitam muncul apabila semua komponen bernilai seratus (100) persen. Setiap warna pada permodelan CMYK memiliki gradasi warna tersendiri.[14]

$C = 1 - R$…….....…….(2.1)

$M = 1 - G$……...…(2.2)

$Y = 1 - B$……...(2.3)



Gambar 2.8 Komponen Warna CMYK[14]

Gambar 2.8 memperlihatkan daripada pembagian warna yang menciptakan CMYK yang terdiri dari kuning, magenta, cyan serta jitam sebagai warna dasar yang menghasilkan warna skunder seperti hijau, merah dan biru tua.

2.3.2 Grayscale Mode

Grayscale adalah rentan warna yang ada dalam rentang gradasi warna hitam dan putih. Mode yang sering digunakan saat ini adalah menggunakan pembagian 256 jenis warna abu – abu. Setiap pixel pada suatu foto atau video yang diproses ke dalam mode grayscale akan mempunyai warna kecerahan (*brightness*) dengan rentang nol(0) untuk hitam sampai dua ratus lima puluh lima (255) untuk gambar yang berwarna putih. Nilai *grayscale* juga dapat diterapkan ke dalam bentuk persentase dengan nilai 100% untuk hitam dan nol (0) persen untuk putih. Cara mengkonversikan warna RGB ke dalam mode grayscale adalah dengan mencari rata-rata nilai dari ketiga warna dasar tersebut dengan menggunakan rumus sebagai berikut:

$Gray = (R+G+B)/3$ ……(2.4)

Nilai konversi warna RGB ke dalam mode *grayscale* atau keabuan dapat juga dilakukan dengan cara member bobot pada setiap elemen warna ,sehingga persamaan diatas diubah menjadi :

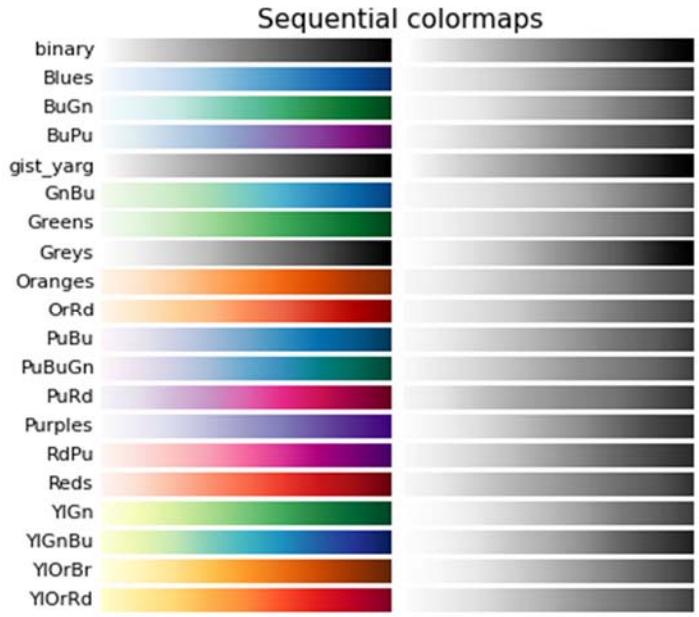
$Gray = w_R R + w_G G + w_B B$ ……(2.5)

Dengan nilai w pada setiap warna dasar (*red,green,blue*) yaitu sebesar :

$w_R = 0{,}299$ ……(2.6)

$w_G = 0{,}587$……....(2.7)

$w_B = 0{,}114$ ……....(2.8)



Gambar 2.9 Hasil Pembacaan Tingkat Keabuan Dari beberapa Kombinasi Warna

Gambar 2.9 memperlihatkan beberapa hasil pembacaan tingkat keabuan dari beberapa perpaduan warna yang didapat dari pemodelan warna baik RGB maupun CMYK. Dapat dilihat bahwa warna yang cenderung memiliki warna gelap maka ketika pembacaan pada *grayscale* warna yang mencolok adalah warna hitam. Sedangkan pada warna yang cerah atau cenderung pudar

maka warna yang terlihay pada *grayscale* mendekati warna putih. [14]

2.3.3 Threshold

Threshold adalah membatasi dan didalam kamus terjemah Bahasa inggris ke dalam Bahasa Indonesia threshold sendiri memiliki arti ambang. *Threshold* adalah mode yang membaca atau memetakan pixel yang memenuhi syarat ambang batas pemetaan menjadi satu nilai pixel yang kita kehendaki dimana rumus persamaannya sebagai berikut :

$f(x,y) = 0, \quad f1(x,y) < T$ ……..…..(2.9)

$f(x,y) = Tmax, \quad f1(x,y) \geq T1$ …….……...…(2.10)

Dimana :

F(x,y) = citra hasil threshold

T = nilai pemetaan pixel yang digunakan sebagai acuan

Gambar 2.10 Hasil Dari Tresholding Pada Grayscale Gambar Wajah Wanita[8]

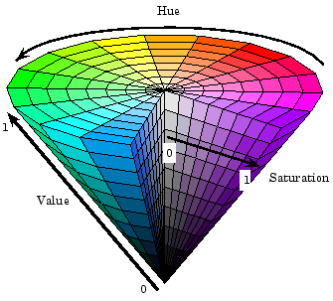
Gambar 2.10 di atas memperlihatkan hasil dari *tressholding* pada gambar wajah wanita yang sebelumnya diubah menjadi grayscale terlebih dahulu. Pada hasil tressholding terlihat bahwa warna yang memiliki nilai pixel yang kurang dari nilai T yang telah ditetapkan akan berubah menjadi hitam. Sedangkan warna yang memiliki nilai pixel yang lebih dari nilai T akan berubah menjadi putih. Fungsi dari threshold ini sendiri adalah membatasi warna sesuai nilai pixel (T) yang telah ditetapkan agar gambar dapat

dibedakan baik gambar wajah wanita serta backgroundnya sendiri.hasil tersebut bergantung dengan nilai yang digunakan sebagai batas.[14]

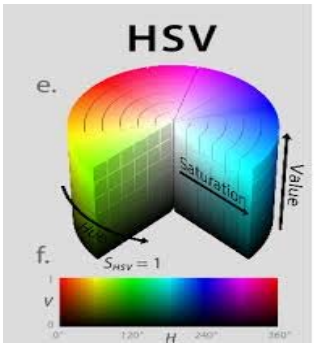
2.3.4 HSV

HSV adalah salah satu mode warna yang mampu mendefinisikan warna dasar (RGB) menjadi kedalam terminology *Hue* ,*Saturation* and *Value*, dimana :

- Hue adalah warna yang direfleksikan oleh sebuah objek. Nilai dari hue berdasarkan dari lokasi melingkar yang diekspresikan dengan derajat sudut 0^0 sampai 360^0. Dalam penggunaannya, hue mengidentifikasi nama sesuai dengan warna yang muncul seperti orange(jingga),pink dan kuning.[15]
- Saturation, sering dikenal dengan chroma, yaitu ukuran atau kemurnian sebuah warna, Saturation merepresentasikan ukuran (kuantitas) dari proporsi keabuan pada hue, ukurannya dalam bentuk persentase dari 0% (*white*) sampai dengan 100% (*fully saturated*).[15]
- Value memiliki arti kecerahan dari warna yang didapatkan dari nilai hue dengan warna saturation. Nilainya berkisar antara 0 sampai 100%. Apabila nilainya 0 maka warnanya akan menjadi hitam dan apabila nilainya dinaikkan maka kecerahan akan menjadi naik dan akan muncul variasi-variasi baru dari warna tersebut.[15]
- Penampakan dari penjelasan Hue Saturation Value diatas digambarkan sebagaimana gambar pemetaan warna pada gambar 2.11 sebagai berikut:



A



B

Gambar 2.11 (A) dan (B) Adalah Pemetaan Warna HSV[15]

$$V = \frac{1}{3}(R+G+B) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.11)$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.12)$$

$$H = \begin{cases} 0, & f_1(x,y) < T \\ 255, & f_1(x,y) \geq T \end{cases} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.13)$$

$$\Theta = \cos^{-1}\left\{\frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}}\right\} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (2.14)$$

Dimana :

V = nilai dari terminology *Value*
S = nilai dari terminology *Saturated*
H = nilai dari terminology *Hue*
Θ = nilai gradien dari *hue,saturation dan value*

2.3.5. *Canny Edge Detection*

Edge Detection (deteksi tepi) pada suatu citra adalah operasi yang dijalankan untuk mendeteksi garis tepi (*edges*) yang membatasi dua wilayah citra homogen yang memiliki tingkat kecerahan yang berbeda. Secara umum, *edge detection* dalam citra dinyatakan sebagai titik yang nilai warnanya berbeda cukup besar dengan titik yang ada di sebelahnya.[14] Tujuan dari *edge detection* ini adalah:

- Untuk menandai bagian yang menjadi detail citra.
- Untuk memperbaiki detail dari citra yang kabur, yang terjadi karena *error* atau adanya efek dari proses akuisisi citra.
- Mengubah citra 2D menjadi bentuk kurva. Suatu titik (x,y) dikatakan sebagai tepi (*edge*) dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dalam pikselnya.

2.3.5 *Bitwise Operator*

Operator bitwise adalah fungsi yang digunakan untuk memanipulasi data. Data yang dapat diolah dengan menggunakan operator bitwise adalah data yang sudah diubah ke dalam bentuk biner. Bentuk dari operator bitwise beraneka macam.[16] Adapun merupakan contoh operator bitwise seperti berikut :

Tabel 2.1 Contoh Beberapa Fungsi Bitwise

Bentuk Operasi	Keterangan
AND	Bitwise AND
OR	Bitwise OR
XOR	Bitwise OR eksclusive
NOT	Bitwise NOT
SHL	Bitwise Shift Left
SHR	Bitwise Shift Right

2.3.6.1Operator Bitwise (Shift Right)

Operator bitwise shift right digunakan untuk menggeser nilai elemen bit ke kanan. Biasa digunakan dalam menambah nilai dari bit secara otomatis.[16]

2.3.6.2 Operator Bitwise (Shift Left)

Operator bitwise shift left digunakan untuk menggeser nilai elemen bit ke kiri. Biasa digunakan untuk mengurangi nilai bit secara otomatis[16]

2.3.6.3 Operator Bitwise (or)

Digunakan untuk mencari nilai elemen bit dari dua operasi. operasi tersebut akan bernilai benar jika salah satu atau kedua nilai bit pada gambar benar. Biasa digunakan untuk menggabungkan gambar dengan nilai bit yang berbeda.[16]

2.3.6.4 Operator Bitwise (and)

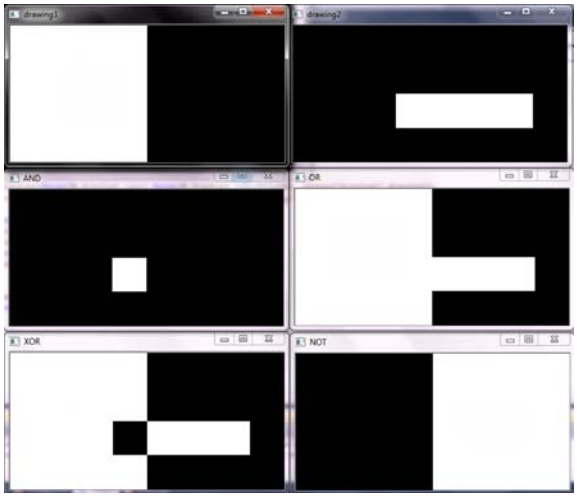
Digunakan untuk mencari nilai elemen bit dari dua operasi. Operasi tersebut akan bernilai benar jika kedua nilai bit tersebut nilainya sama.[16]

2.3.6.5 Operator Bitwise (XOr)

Operasi Bitwise (XOR) Digunakan untuk membandingkan elemen bit dari suatu operasi. Operator Bitwise akan bernilai benar jika kedua nilai bit nya sama benar.[16]

2.3.6.6 Operator Bitwise (Not)

Operasi bitwise (NOT) digunakan untuk membalik nilai bit dari suatu operator.[16]

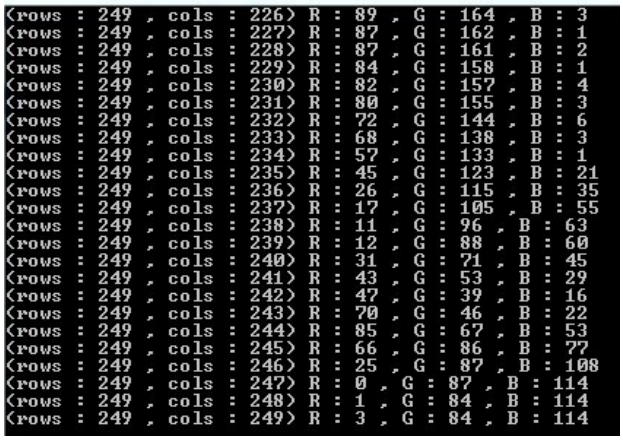


Gambar 2.12 Hasil Perbandingan Dari Beberapa Hasil Scaning Operasi bitwise Pada Dua Windows Drawing (Drawing 1 dan Drawing 2)[16]

Gambar 2.12 menunjukan perbandingan hasil dari beberapa scanning operasi bitwise pada dua jendela (windows1 dan windows2).

2.3.7 Pixel Scanning Value

Nilai dari pixel tiap warna yang ada pada suatu foto dapat diketahui juga dengan menggunakan scanning pixel. Proses ini dapat mengetahui komposisi warna yang didapat dari nilai pixel yang terbaca. Proses scanning dapat dimulai dari berbagai maram arah. Proses scanning dilakukan dengan proses yang runtut dan tidak saling menimpang. [14]



Gambar 2.13 Hasil Dari Pixel Scanning[14]

Gambar 2.13 diatas menunjukan bahwa nilai dari tiap pixel didapatkan melalui *scanning* yang dilakukan secara runtut atau urut mengikuti barisnya maupun kolom dari gambar. Berikut beberapa contoh langkah dari proses *scanning* yang dilakukan berdasarkan arah:

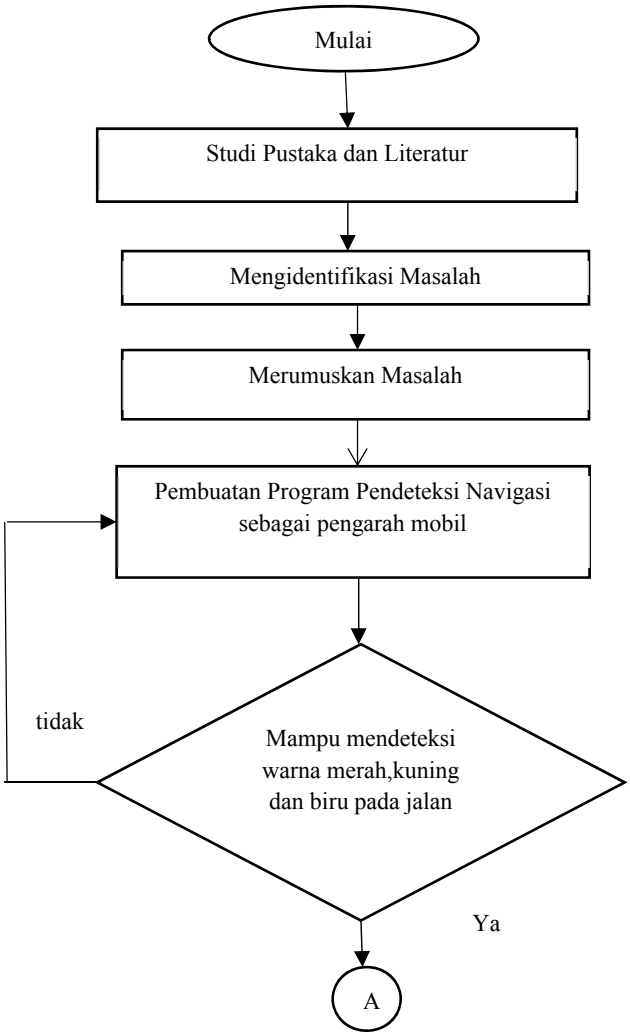
- Dari kiri ke kanan dimulai dari sebelah atas
- Dari kiri ke kanan dimulai dari sebelah bawah
- Dari kanan ke kiri dimulai dari sebelah atas
- Dari kanan ke kiri dimulai dari sebelah bawah
- Dari atas ke bawah dimulai dari sebelah kiri
- Dari atas ke bawah dimulai dari sebelah kanan
- Dari bawah ke atas dimulai dari sebelah kiri
- Dari bawah ke atas dimulai dari sebelah kanan

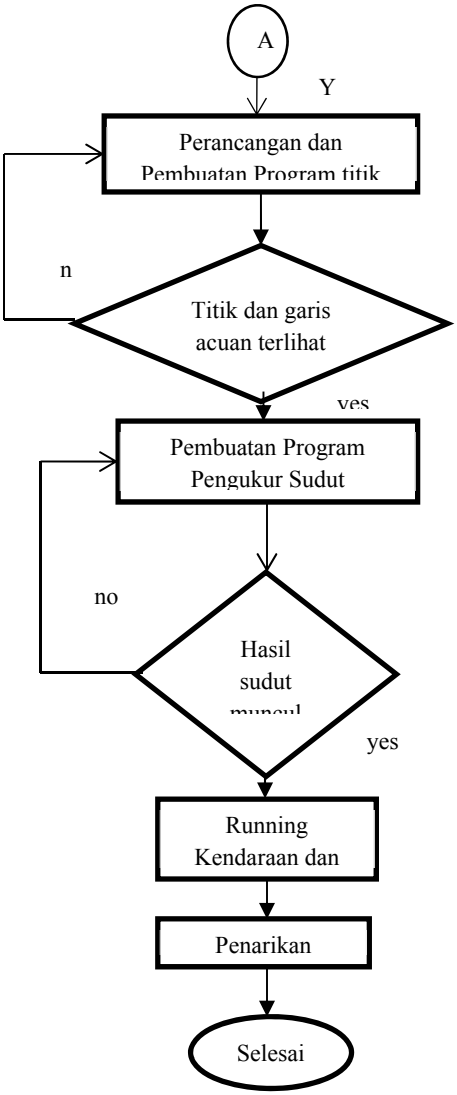
(*Halaman ini sengaja dikosongkan***)**

BAB III
METODE PENELITIAN

Pada Bab III ini akan menjelaskan langkah apa saja yang akan dilakukan sebagai acuan dari penelitian untuk menciptakan aplikasi yang dapat mendeteksi navigasi dari tiga warna jalan antara lain yaitu; jalan berwarna merah,jalan berwarna kuning, dan jalan berwarna biru serta dapat menunjukan nilai sudut dari navigasi pada *googlemaps* sehingga dapat digunakan sebagai guidance pada *vehicle* yang akan dibuat. Tahap penelitian yang akan dilakukan ditampilkan sebagai diagram alir metodelogi penelitian seperti flowchart metode penelitian dibawah ini. Metode penelitian menjelaskan awal dari perancangan program dari mempelajari studi pustaka , mengidentifikasi masalah, merumuskan masalah, lalu membuat program navigasi sebagai pengarah mobil setelah itu membuktikan apakah navigasi berhasil terbaca dan apabila tidak maka kembali lagi ke perancangan program sedangkan bila berhasil maka dilanjutkan ke langkah selanjutnya yang akan dijelaskan lagi lebih rinci pada gambar. Pada diagram alir menjelaskan pendeteksian tiga warna warna dari jalan yang akan proses serta gambar . sedangkan pada diagram alir 3.3 yang menjelaskan bagaimana cara mendeteksi sudut sebagai berikut.

3.1 Flowchart Metode Penelitian





Gambar 3.1 Flowchart Metode Penelitian

Gambar 3.1 menunjukan flowchart dari metode penelitian yang dilakukan untuk membuat kode pemrograman pendeteksian arah jalan pada GPS googlemaps sebagai navigasi mobil tanpa *driver*. Langkah pertama yang dilakukan adalah melakukan pencarian studi pustaka yang bertujuan untuk memperoleh informasi tentang penelitian tentang autonomous car. Lalu mengidentifikasi masalah yang didapatkan dari proses studi pustaka. Kemudian dari pengidentifikasian masalah didapatkan rumusan masalah yang akan dibahas sebagai perumusan masalah dari tugas akhir ini. Kemudian dilakukan pembuatan program pendeteksi navigasi sebagai pengarah mobil. Kemudian melihat hasil yang ditunjukan dari program pendeteksian navigasi mobil dengan melihat warna yang dapat dideteksi oleh program. Warna yang dideteksi adalah warna merah, kuning, dan biru yang ditampilkan pada jalan gps googlemaps. Apabila tidak mampu mendeteksi warna yang sudah ditentukan tersebut maka perlu dilakukan proses pembuatan program pendeteksi navigasi ulang. Apabila program mampu mendeteksi warna jalan yang sudah ditentukan tersebut maka program tersebut dianggap benar dan dilanjutkan dengan proses pembuatan perancangan dan pembuatan program titik dan garis acuan. Pembuatan program titik dan garis acuan dapat dikatakan benar jika posisi titik dan garis acuan sesuai dengan posisi yang telah ditentukan. Penentuan posisi titik tersebut berdasarkan letak dari centroid pointer dan titik hasil scanning piksel pada jalan googlemaps. kemudian dua titik tersebut digabungkan menjadi sebuah garis. Lalu membuat garis vertical yang tegak lurus dengan titik centroid segitiga. Apabila pada program muncul dua garis yang sesuai dengan posisi titik yang dijadikan acuan tersebut maka program dianggap benar. Apabila pada program tidak terdapat garis atau posisi garis yang tidak tepat maka program salah dan perlu dilakukan proses pembuatan program ulang. Setelah berhasil melakukan pembuatan program titik dan acuan dilanjutkan dengan pembuatan program pengukuran sudut. Program pembuatan sudut dikatakan benar apabila hasil yang didapat muncul pada window dan memiliki

range penyimpangan pembacaan sebesar kurang lebih 2^o dari yang ditampilkan pada window. Setelah seluruh proses dilakukan kemudian dilanjutkan dengan proses running kendaraan dan melakukan validasi sudut. Setelah melakukan seluruh langkah-langkah program maka proses dari penelitian selesai.

3.2 Tahap Awal

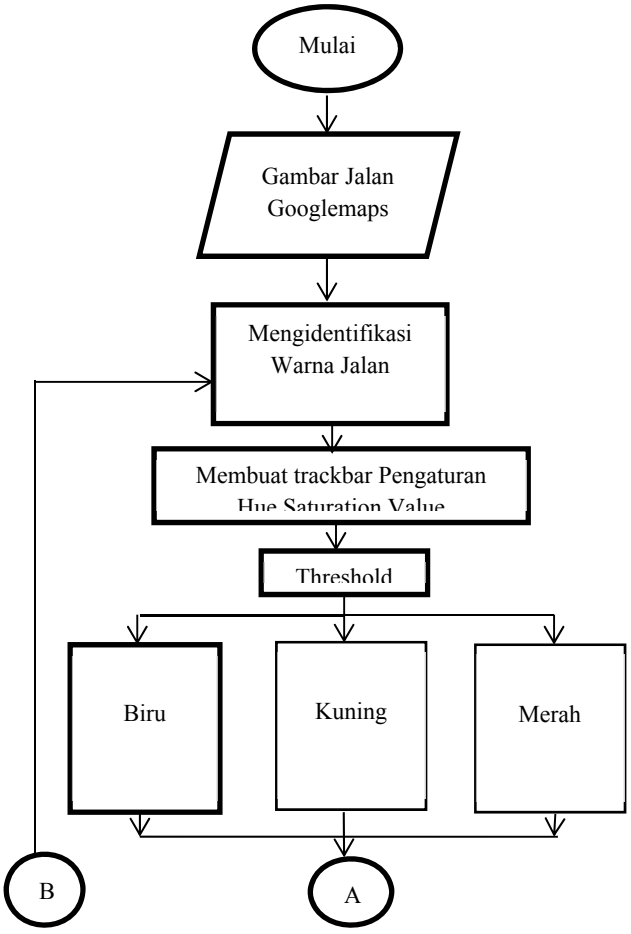
Pada tahap awal atau persiapan yang perlu dilakukan antara lain mempelajari studi pustaka dari beberapa penelitian yang sudah ada,lalu kemudian mencoba mengidentifikasi masalah yang akan berhubungan dengan tugas akhir ini dan melakukan perumusan masalah. Studi pustaka adalah sebagai acuan untuk penulis dalam mencari tahu permasalahan yang akan diteliti dalam pembuatan program pendeteksian rute pada *googlemaps* beserta nilai sudut dari jalan. Perumusan masalah mencakup perancangan program untuk deteksi jalan dan pencarian sudut jalan. Pembuatan program untuk navigasi

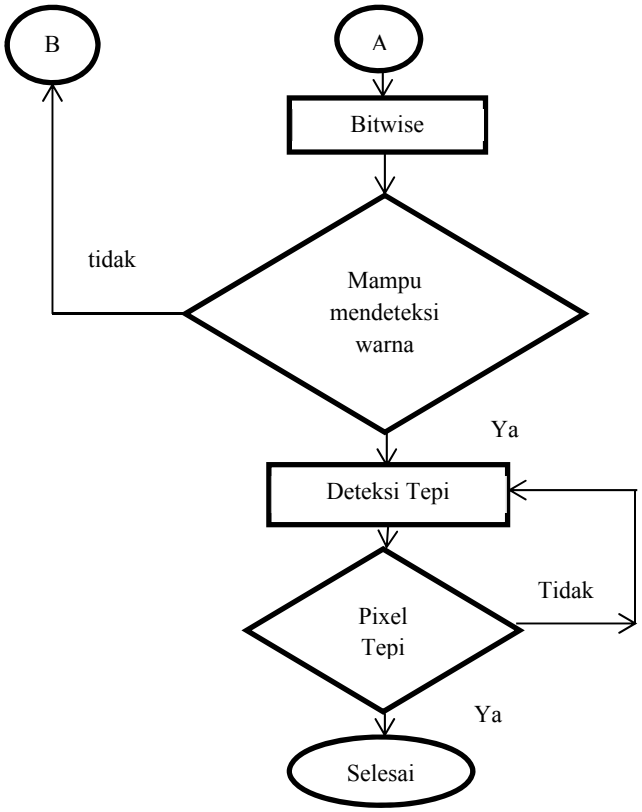
3.3 Tahap Perancangan Program

Tahap perancangan program merupakan salah satu tahap yang penting. Dimana apabila salah pada tahap ini maka program tidak dapat mendeteksi jalan dan sudutnya. Langkah awal dari perancangan program pendeteksi jalan yaitu mendeteksi dan mengenalkan rute dari googlemaps secara *real-time* yang ditangkap oleh kamera. Namun untuk membuat program yang mampu menangkap dan mengenali jalan dari warnanya sekaligus mencari sudut jalan tersebut dibutuhkan dua proses antara lain, membuat program pendeteksi rute dan membuat program untuk mencari sudut jalan.

3.3.1 Flowchart Program Deteksi Rute

Berikut merupakan flowchart perancangan program pendeteksian jalan pada googlemaps





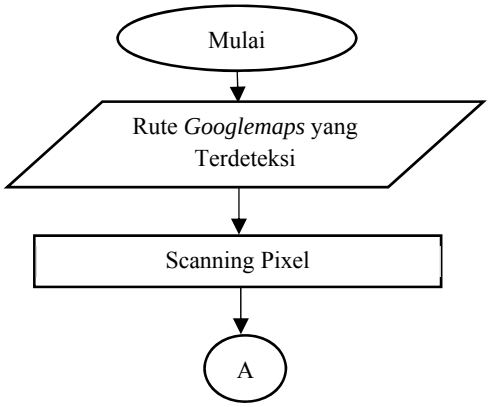
Gambar 3.2 *Flowchart* Perancangan program pendeteksi jalan googlemaps

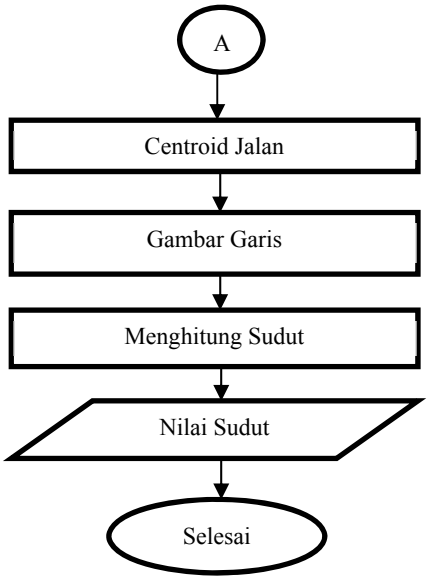
Gambar 3.2 menunjukan urutan urutan dari perancangan program pendeteksi jalan googlemaps. Langkah awal yang dilakukan adalah mendeteksi gambar jalan yang ada pada citra googlemaps kemudian mengubah format warnya dari jalan yang semula RGB (Red Green Blue) ke dalam format HSV (Hue Saturation Value). Setelah semua warna terdeteksi langkah selanjutnya adalah menggabungkan semua hasil threshold ke dalam satu frame dengan menggunakan fungsi bitwise. Lalu

melakukan pengujian program bitwise dengan melihat apakah ketiga warna jalan yaitu merah, kuning, dan biru dapat terbaca semua hasilnya pada satu frame sesuai keadaan warna yang muncul pada jalan yang ditunjukan dengan gambar threshold jalan yang terus terhubung. Apabila terdapat bagian threshold jalan yang terputus maka program dikatakan gagal sehingga perlu dilakukan proses pengidentifikasi warna jalan kembali. Sedangkan apabila hasil yang ditunjukan threshold jalan saling terhubung tanpa putus maka dilanjutkan ke proses selanjutnya yaitu membuat program proses deteksi tepi yang berguna untuk memperhalus bagian tepi jalan. Apabila hasil yang didapat banyak terdapat nois atau bercak maka perlu dilakukan pengaturan program ulang. Apabila gambar deteksi tepi terdapat sedikit nois maka langkah flowcart selesai.

3.3.2 *Flowchart* Perancangan Program Pencari Sudut Rute Pada *Googlemaps*

Berikut merupakan Flowchart dari perancangan program pendeteksi rute pada *googlemaps* lalu setelah itu dilakukan perancangan pengukuran sudut rute pada *googlemaps:*





Gambar 3.3 Flowchart Perancangan program Pencari Sudut Rute Pada *Googlemaps*

Gambar 3.3 menjelaskan diagram alir perancangan program yang digunakan untuk mencari sudut jalan pada googlemaps. Langkah awal yang dilakukan adalah scanning pixel untuk mencari posisi tengah pada jalan. Lalu dilanjutkan dengan mencari centroid dengan menggunakan fungsi moments. Setelah itu menghubungkan titik tersebut menjadi sebuah garis. Setelah menggambar garis dilakukan perhitungan sudut berdasarkan tangensian yang diperoleh dari pertemuan dua garis yang telah dibuat. Setelah nilai sudut keluar maka proses pembuatan program selesai.

3.3.3. Cara Kerja

Cara kerja program ini adalah dengan mengambil gambar jalan pada *googlemaps* kemudian ditangkap menggunakan kamera yang terhubung dengan laptop. Input yang berupa video yang diambil secara *real-time* ,kemudian dinalisa dengan menggunakan program C++ dari openCV dan Visual Studio. Setelah berhasil mendeteksi jalan , maka hasil analisa tersebut digunakan untuk mencari nilai sudut jalan.

3.3.4 Proses Pengerjaan

Proses pengerjaan rancangan program pada tugas akhir ini memiliki beberapa tahapan antara lain ; pengenalan gambar, pendeteksian jalan pada *googlemaps*, dan pencarian nilai sudut rute pada *googlemaps*

1. **Pengenalan Gambar**
   Pengenalan gambar yaitu ketika kita mengambil gambar lalu kemudian diatur dan dianalisa. Tanpa pengenalan gambar yang tepat maka segmentasi dan proses pengerjaan selanjutnya akan sulit dan tidak efisien. Alasan utama untuk pengenalan gambar adalah untuk memperbaiki kualitas gambar sehingga gambar yang diproses terpusat pada wilayah yang diinginkan, dalam kasus ini adalah jalan pada *googlemaps*.

2. **Pendeteksian Jalan**
   pendeteksian jalan dilakukan agar jalan dari objek yang dijadikan inputan dalam hal ini jalan pada *GPS googlemaps* yang terletak didalam mobil dapat diketahui. Terdapat beberapa tahapan dalam pendeteksian jalan. Yang pertama dilakukan adalah segmentasi warna untuk membedakan warna jalan yang akan dilalui mobil dengan jalan yang tidak dilalui mobil.terdapat tiga warna utama yang kemungkinan akan dilalui mobil. Kemudian mengubah model warna dari red,green,blue menjadi HSV llalu dilanjutkan melakukan pencarian nilai HSV(Hue Saturation dan Value) dari dari

setiap warna tersebut. Setelah nilai HSV setiap warna diketahui kemudian melakukan proses threshold. Setelah berhasil mentreshold ketiga warna tersebut kemudian melakukan proses penggabungan nilai dari setiap warna dengan menggunakan proses bitwise. Apabila ada warna yang tidak terbaca maka dilakukan pengidetifikasian warna lagi. Apabila berhasil mengidentifikasi seluruh warna jalan pada GPS yang akan dilalui mobil dilakukan proses pendeteksian tepi sehingga gambar jalan dapat terdeteksi lebih halus. Apabila telah selesai maka proses pendeteksian jalan telah selesai dan dilanjutkan ke proses selanjutnya yaitu proses pendeteksian sudut. Hasil akhir dari proses ini adalah gambar jalan yang telah terdeteksi dengan bentuk warna biner.

3. Mencari Nilai Sudut Belok dan Arah Jalan

Dalam pencarian nilai sudut dibutuhkan beberapa metode untuk mendapatkan nilainya. Dibutuhkan setidaknya dua garis dalam mencari sudut. Untuk membuat garis pertama titik awal yang dicari adalah titik tengah jalan dengan menggunakan *Pixel scanning* dimana titik tengah jalan tersebut disamakan dengan titik *center of gravity* pada pointer. Titik yang kedua merupakan titik pada ujung atas dari jalan yang didapat dengan menggunakan *scanning pixel*. Setelah didapatkan dua titik maka dapat digambarkan sebuah garis mengikuti titik tersebut. Sedangkan garis kedua didapatkan dari titik *center of gravity* untuk titik pertama dan titik pada koordinat (cg,y) *pixel*. Dari kedua garis tersebut kita dapat membaca nilai sudut yang dapat digunakan sebagai nilai input pada mobil.

3.4. Tahap Analisa

Tahap analisa adalah tahap yang membahas dan menarik kesimpulan dari software yang dirancang. Pada pembahasan dilakukan analisa rancangan *software* yang dijalankan atau di

running secara *real time.* Analisa s*oftware* yang dilakukan meliputi pendeteksian warna jalan dan analisa nilai sudut jalan yang didapat dari proses *image processing* dari gambar GPS secara *real-time*.Dari pendeteksian warna dan sudut tersebut kemudian ditarik kesimpulan apakah terdapat kekurangan ataukah sudah sesuai dengan yang diharapkan. Kesimpulan yang telah didapatkan adalah tujuan akhir dari penelitian ini

BAB IV
RANCANG BANGUN PROGRAM

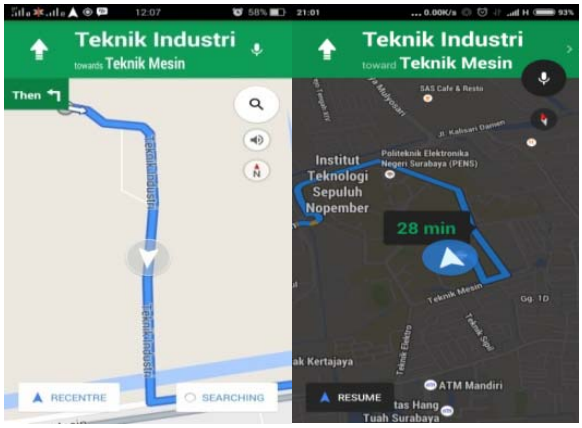
4.1 Perancangan System

Perancangan program pendeteksi jalan serta mencari sudut pada GPS Googlemaps menggunakan teknologi s*oftware* berbasis *computer vision* .Dimana teknologi berbasis computer tersebut membutuhkan beberapa input compiler atau penerjemah bahasa program serta library atau kamus yang dapat diproses. Software yang digunakan untuk merancang program ini antara lain, Microsoft visual studio 2012 dan opencv 2.4.10 version.

Untuk mendapatkan hasil pendeteksian warna jalan serta nilai sudut yang sesuai dengan harapan, maka memerlukan pengujian menggunakan data input. Data input yang diuji merupakan data jalan pada GPS googlemaps secara real-time dan memiliki beberapa spesifikasi sebagai berikut:

A. Warna jalan googlemaps berwarna biru,kuning,merah dan diambil pada kondisi *day mode*
B. Letak jalan googlemaps pada layar handphone tidak menyimpang dari posisi pointer googlemaps dan tidak berpindah tempat

Pada system ini perlu dilakukan beberapa perubahan setting googlemaps terlebih dahulu sebelum dilakukan pengambilan data dengan image processing. Setting googlemaps yang dilakukan pertama adalah mengatur kondisi jalan ke kondisi day mode.



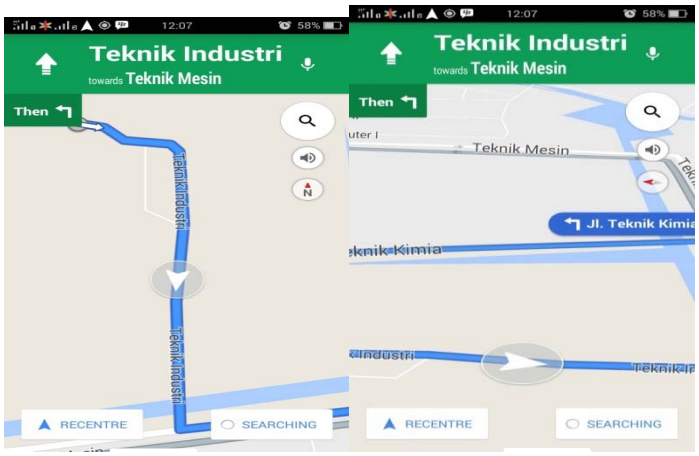
A B

Gambar 4.1 Perbedaan warna Google Maps Pada(A) Siang Hari dan (B) Malam Hari

Gambar 4.1 menunjukan perbedaan warna pada dua mode navigasi google maps. Pada kondisi day mode, jalan yang tidak dituju/jalan disekitar rute navigasi berwarna putih dan terdapat lingkaran putih disekitar pointer dan warna pointer memiliki warna yang hampir sama dengan jalan yaitu berwarna biru. Sedangkan pada kondisi night mode, lingkaran di sekitar area pointer yang memiliki warna yang sama dengan jalan sedangkan warna pointer berubah menjadi putih dan jalan yang tidak dituju/ jalan disekitar rute navigasi berubah menjadi abu-abu kehitaman. Dalam pengambilan data image processing mode yang digunakan adalah googlemaps dengan kondisi day mode, karena pada day mode antara jalan dan pointer dipisahkan oleh lingkaran putih sehingga jalan yang berwarna biru dapat diidentifikasi .sedangkan apabila night mode yang digunakan maka lingkaran disekitar pointer berwarna biru yang dapat menyebabkan kesalahan dalam proses scanning jalan karena tidak dapat dipisahkan antara jalan dengan lingkaran.

Kemudian dilakukan pengaturan untuk arah kompas. Arah kompas diatur agar tidak selalu menghadap ke utara. Tujuan dari

pengaturan tersebut adalah untuk memudahkan dalam proses pencarian sudut karena arah jalan selalu sesuai dengan arah kompas.

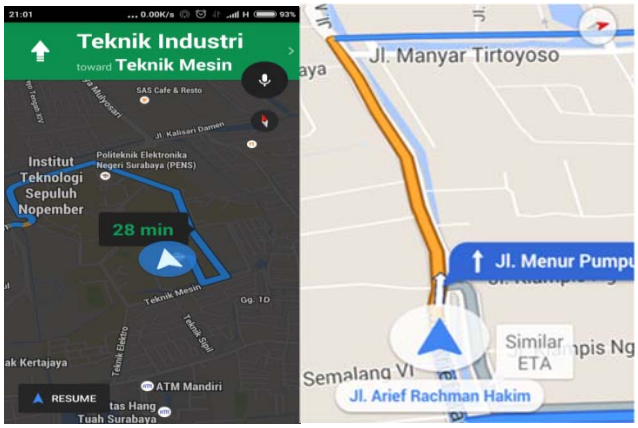


A B

Gambar 4.2 (A)Arah Kompas Googlemaps Selalu Menghadap Utara, (B) Arah Kompas Berubah-Ubah

Gambar 4.2 menunjukan perbedaan antara dua setting kompas googlemaps yang berbeda. Pada gambar (A) arah kompas diatur selalu menghadap utara (B) arah kompas yang berubah-ubah. Perbedaannya adalah, pada pengaturan ini jalan akan selalu berusaha menghadap ke atas. Dan sudut kompas pada googlemaps sesuai dengan arah jalan.

Pengaturan berikutnya adalah *re-center* yang diubah ke mode on agar posisi titik pusat atau titik acuan (dalam hal ini adalah titik pusat pointer)yang terhubung dengan titik tengah jalan pada navigasi googlemaps tidak berpindah tempat dan sesuai dengan tempat mobil berada. Apabila tidak diatur ke *re-center* sudut jalan yang terbaca tidak sesuai dengan sudut yang diharapkan dan sudut jalan tidak bisa divalidasi karena letak titik yang dijadikan acuan tidak sesuai dengan posisi mobil.



A B

Gambar 4.3 (a) Letak Jalan Ketika Tanpa Re-Center, (b) Letak Jalan Dengan Re-Center

Gambar 4.3 menunjukkan perbedaan posisi jalan dalam tampilan layar smartphone saat tidak menggunakan *re-center* dengan menggunakan *re-center*. Letak posisi jalan dan pointer dalam gambar (a) tidak dalam keadaan posisi re-center, sehingga untuk melihat letak jalan dan pointer pada saat navigasi, pengguna harus menggeser layar untuk melihat posisi jalan. Sedangkan pada gambar (b) dalam keadaan posisi re-center, sehingga letak posisi jalan selalu mengikuti pointer sebagai titik acuan yang selalu di tengah atau tidak berubah. Dari pengaturan google maps yang telah dilakukan tersebut dapat memudahkan dalam mengidentifikasi jalan pada google maps dan memudahkan dalam pencarian nilai sudut yang dibentuk oleh jalan google maps.

Computer yang digunakan untuk mengolah data dari image processing memiliki spesifikasi sebagai berikut:

A. Toshiba NB205-N311 dengan processor Intel® Atom™ @1,66 GHz
B. RAM 2,00 GB

C. System Type 32-bit Operating System
D. Operating System Windows 7 Ultimate

4.2 Konstruksi Program

Konstruksi program merupakan salah satu sub bab yang membahas langkah-langkah dari metode yang dipilih ketila melakukan perancangaan program pendeteksi jalan dan menghitung sudut belok. Langkah-langkah yang dilakukan antara lain membuat program deteksi jalan googlemaps kemudian, membuat dua garis yaitu garis vertical dan garis yang terhubung dengan centroid pointer dan garis jalan yang terhubung dengan centroid pointer sebagai representasi arah. Pembuatan konstruksi program tersebut dilakukan dengan menggunakan Microsoft visual studio 2012 yang sudah terhubung dengan opencv yang digunakan sebagai library dari bahasa pemrograman C++ yang diperlukan.

4.2.1 Deteksi Jalan Google Maps

4.2.1.1 Pre-Processor Directive

Untuk mengatur kode pemrograman agar sesuai yang kita harapkan maka perlu dilakukan *pre-processor directive*. *Pre-processor directive* yang dimaksud adalah mengatur fungsi pada kode pemrograman menuju *library* yang sudah disiapkan. Berikut adalah potongan program yang digunakan sebagai *pre-processor directive*:

```
(Global Scope)
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\imgproc\imgproc.hpp>
#include <iostream>
#include <windows.h>
#include <math.h>

using namespace std;
using namespace cv;
```

Gambar 4.4 Pre-processing Directive

Gambar 4.1 diatas adalah bagian dari *pre-processor directive*. *Pre-processing directive* diperlukan untuk mengatur kompilasi program agar mengarah kepada *library* yang diinginkan. Program ini menggunakan #include sebagai pre-processor program yang digunakan untuk memproses *head filecompiler* yang ada pada program. Beberapa compiler dari library C++ yang digunakan pada program ini antara lain; iostream.h, windows.h, dan math.h. sedangkan compiler yang berasal dari opencv antara lain; <opencv2\highgui\highgui.hpp> dan <opencv2\imgproc\imgproc.hpp>. Head file compiler memiliki beberapa fungsi baik sebagai perintah input, pengaturan dari kumpulan class,object atau fungsi dari program tertentu serta pengaturan perintah output. Sedangkan untuk memanggil program yang telah di-*include* program yang digunakan adalah menggunakan *using namespace* dimana (std) sebagai tanda yang digunakan untuk memanggil bahasa C++ sedangkan (cv) sebagai tanda untuk memanggil fungsi yang ada pada opencv.

4.2.1.2 Open Webcam

Untuk menjalankan fungsi dari kode pemrograman maka diperlukan aktifasi. Aktifasi tersebut dilakukan pada *main function*. Dibawah ini merupakan potongan kode pemrograman pada fungsi utama. Didalan fungsi utama juga terdapat berbagai macam fungsi yang digunakan untuk membuat kode pemrograman untuk mencari nilai sudut pada jalan yang akan digunakan sebagai input pada mobil.

```
int main ()
{
    VideoCapture cap (1);

        waitKey (1);
}
```

Gambar 4.5 Potongan Dari Fungsi Utama Dimana Salah Satu Fungsinya Digunakan Untuk Menangkap Video

Gambar 4.2 menjelaskan bagian dari fungsi utama dari suatu program. Semua fungsi yang telah dipanggil seperti yang ada pada pre-processing directive digabungkan dan diolah di dalam program inti atau pada fungsi utama. Fungsi adalah modul yang berisi kode-kode untuk menyelesaikan masalah-masalah tertentu. Tanda { (kurung buka kurawal) menandakan dari awal program dimulai sedangkan }(kurung tutup kurawal)menandakan akhir program. Pada awal program yaitu VideoCapture cap , merupakan sebuah fungsi dari opencv yang digunakan untuk meng-aktifkan kamera. Nilai satu(1) menunjukan bahwa kamera yang aktif adalah kamera eksternal, apabila nilai (0) maka yang akan aktif adalah kamera bawaan pada laptop. Kamera yang digunakan pada pembuatan kode pemrograman ini menggunakan kamera eksternal. Kamera tersebut diletakkan pada mobil dan berfungsi sebagai penangkap gambar GPS pada *handphone*

Gambar 4.6 Hasil Dari Fungsi VideoCapture pada Proses Open Webcam (Dokumentasi Sendiri)

4.2.1.3 Pembuatan Trackbar

Untuk mengatur proses pencarian nilai HSV yang optimal diperlukan suatu alat. Alat yang dimaksud adalah beberapa *trackbar* yang fungsinya mengatur beberapa variabel yang ada pada pengaturan nilai HSV (*Hue, Saturation, Value*). Selain itu pada trackbar tersebut juga berfungsi sebagai pengatur erode, dilate, dan blur. Dibawah ini merupakan kode pemrograman yang digunakan untuk membuat *trackbar* :

```
//creat trackbar
namedWindow("control_jalan",CV_WINDOW_NORMAL);

cvCreateTrackbar ("HueL","control_jalan",&HL,255);
cvCreateTrackbar ("HueM","control_jalan",&HH,255);

cvCreateTrackbar ("SaturationL","control_jalan",&SL,255);
cvCreateTrackbar ("SaturationH","control_jalan",&SH,255);

cvCreateTrackbar ("ValueL","control_jalan",&VL,255);
cvCreateTrackbar ("ValueH","control_jalan",&VH,255);

cvCreateTrackbar ("Erode","control_jalan",&Ero,10);
cvCreateTrackbar ("Dilate","control_jalan",&Dil,10);
cvCreateTrackbar ("Blur","control_jalan",&Blur,10);
```

A

```
//creat trackbar
namedWindow("control_jalank",CV_WINDOW_NORMAL);

cvCreateTrackbar ("HueLk","control_jalank",&HLk,255);
cvCreateTrackbar ("HueMk","control_jalank",&HHk,255);

cvCreateTrackbar ("SaturationLk","control_jalank",&SLk,255);
cvCreateTrackbar ("SaturationHk","control_jalank",&SHk,255);

cvCreateTrackbar ("ValueLk","control_jalank",&VLk,255);
cvCreateTrackbar ("ValueHk","control_jalank",&VHk,255);

cvCreateTrackbar ("Erodek","control_jalank",&Erok,10);
cvCreateTrackbar ("Dilatek","control_jalank",&Dilk,10);
cvCreateTrackbar ("Blurk","control_jalank",&Blurk,10);
```

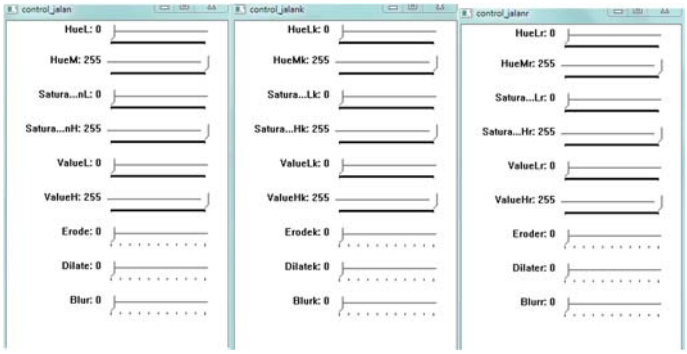
B

```
//creat trackbar
namedWindow("control_jalanr",CV_WINDOW_NORMAL);

cvCreateTrackbar ("HueLr","control_jalanr",&HLr,255);
cvCreateTrackbar ("HueHr","control_jalanr",&HHr,255);

cvCreateTrackbar ("SaturationLr","control_jalanr",&SLr,255);
cvCreateTrackbar ("SaturationHr","control_jalanr",&SHr,255);

cvCreateTrackbar ("ValueLr","control_jalanr",&VLr,255);
cvCreateTrackbar ("ValueHr","control_jalanr",&VHr,255);

cvCreateTrackbar ("Eroder","control_jalanr",&Eror,10);
cvCreateTrackbar ("Dilater","control_jalanr",&Dilr,10);
cvCreateTrackbar ("Blurr","control_jalanr",&Blurr,10);
```

C

Gambar 4.7 Kode pemrograman Untuk Mengatur Nilai HSV, Nilai Erode, Nilai Dilate dan Nilai Blur Pada Warna (A)Biru, (B)Kuning, dan (C) Merah

Gambar 4.7 menjelaskan beberapa kode pemrograman untuk mengatur nilai HSV (*Hue Saturation and Value*) serta mengatur nilai *erode, dilate* dan *blur* dari setiap warna jalan yang akan diproses . Pada kode pemrograman ini warna jalan yang di proses adalah jalan berwarna merah,kuning dan biru. Fungsi dari trackbar sendiri adalah mengatur nilai bit sesuai dengan nilai range yang ditentukan terlebih dahulu. Pada pengaturan trackbar yang digunakan pada kode pemrogramaan ini ditetapkan nilai nol (0) untuk nilai pengaturan terkecil dan (255) untuk nilai pengaturan maksimal.



A B C

Gambar 4.8 (A)Trakbar Kontrol HSV Jalan Biru,(B) Trackbar Kontrol HSV Jalan Kuning,(C) Trackbar Kontrol HSV Jalan Merah

Gambar 4.8 menunjukan hasil dari kode pemrograman pembuatan trackbar yang digunakan untuk mengontrol nilai dari hue, saturation, dan value dari setiap warna jalan (biru, kuning dan merah).

4.2.1.5 HSV dan Threshold

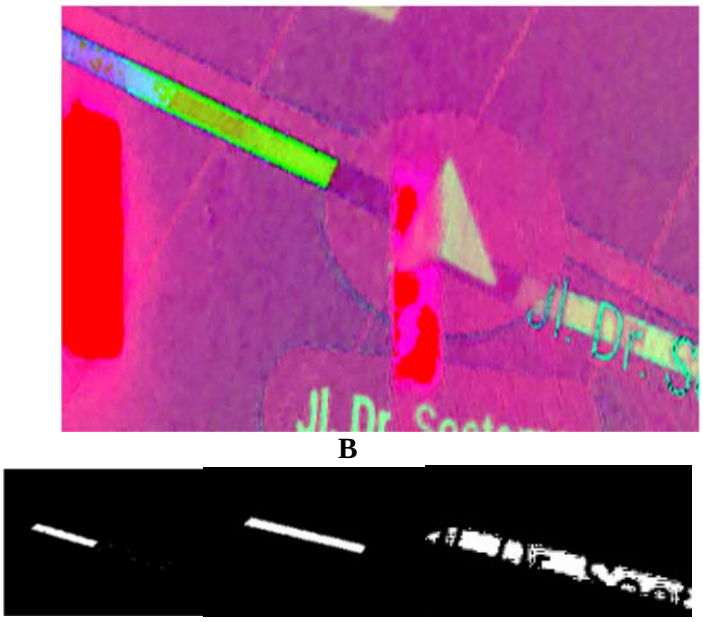
Untuk mengubah format warna agar bisa diatur maka, format warna yang ada saat ini yaitu RGB (*Red, Green and Blue*) diubah menjadi format HSV (*Hue, Saturation, and Value*). Agar format warna dapat diubah maka perlu mengaktifkan fungsi dari code pemrograman BGR2HSV yang ada pada *library* opencv.

```
cvtColor (frame,hsv,COLOR_BGR2HSV_FULL);
cvtColor (framek,hsvk,COLOR_BGR2HSV_FULL);
cvtColor (framer,hsvr,COLOR_BGR2HSV_FULL);

inRange(hsv,Scalar(HL,SL,VL),Scalar (HH,SH,VH),thd);
erode(thd,thd,getStructuringElement(MORPH_CROSS,Size(Ero+1,Ero+1),Point(Ero,Ero)));

inRange(hsvk,Scalar(HLk,SLk,VLk),Scalar (HHk,SHk,VHk),thdk);
erode(thdk,thdk,getStructuringElement(MORPH_CROSS,Size(Erok+1,Erok+1),Point(Erok,Erok)));


inRange(hsvr,Scalar(HLr,SLr,VLr),Scalar (HHr,SHr,VHr),thdr);
erode(thdr,thdr,getStructuringElement(MORPH_CROSS,Size(Eror+1,Eror+1),Point(Eror,Eror)));
```

Gambar 4.9 kode Pemrograman Untuk Mengubah Format Warna dan Treshold

Gambar 4.9 menjelaskan kode pemrograman Untuk Mengubah Format Warna RGB ke HSV yang kemudian dilakukan proses threshold. Untuk mengubah format warna kita fungsi pengubah format warna pada opencv dengan menulis (BGR2HSV). Proses selanjutnya adalah mengatur nilai dari HSV(Hue Saturation Value) dari setiap warna. Nilai tersebut masuk dalam code inRange. Untuk proses eroding, dilate dan blur hanya dibutuhkan ketika gambar yang diproses kurang atau tidak begitu jelas. Hasil keluaran dari code inRange adalah gambar biner yaitu putih dan hitam. Satu nama fungsi(Class) hanya berlaku untuk satu warna sehingga dibutuhkan tiga fungsi untuk mengaktifkan tiga warna yang berbeda secara bersamaan.

A



B

C D E

Gambar 4.10 Menunjukan (A)Gambar Jalan, (B)Hasil HSV, dan Hasil Proses Threshold Dari Warna (C) Merah (D) Kuning (E) Biru

Gambar 4.10 menunjukan hasil proses threshold dari gambar (A) yang diubah menjadi format HSV seperti gambar (B), lalu dilakukan threshold seperti gambar(C),(D) dan (E). Gambar (C) menunjukan hasil threshold dari jalan berwarna merah pada gambar (A). Gambar (D) menunjukan hasil threshold dari jalan berwarna kuning pada gambar (A). Gambar (E) menunjukan hasil threshold dari jalan berwarna biru pada gambar (A).

4.2.1.6 Bitwise_or Function

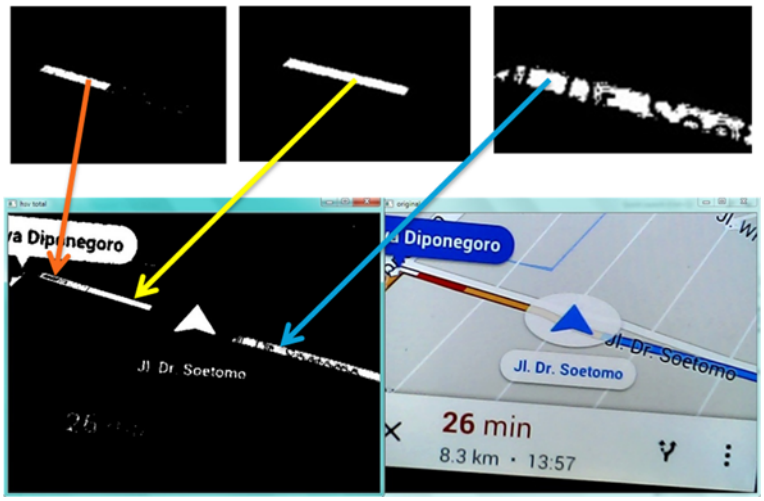
Setelah mengubah format warna ke HSV agar dapat di *threshold* , proses selanjutnya adalah menggabungkan seluruh

hasil nilai HSV kedalam satu window. Berikut adalan fungsi kode pemgrograman yang digunakan untuk menggabungkan nilai *threshold*:

```
    // bitwise Function
    Mat res;
    Mat total;


bitwise_or(thd,thdk,res);
bitwise_or(res,thdr,total);

    //imshow ("or",res);
    imshow ("hsv total",total);
```

Gambar 4.11 kode Pemrograman Fungsi Bitwise

Gambar 4.11 merupakan kode pemrograman fungsi bitwise yang digunakan untuk menggabungkan ketiga warna (merah,kuning,dan biru) tersebut menjadi satu frame. Pada kode pemrograman kali ini menggunakan fungsi Bitwise_or sebagai fungsi penggabung dari ketiga warna tersebut. Tujuan dari penggabungan ini adalah agar jalan pada googlemaps yang digunakan sebagai input dapat terbaca secara terus menerus tanpa harus mengatur ulang nilai *hue, saturation and value* ketika jalan mengalami perubahan warna dari biru ke kuning, kuning ke merah dan sebaliknya atau dapat membaca jalan baik itu ketika jalan berwarna biru (jalan lancar) , jalan kuning (jalan padat merayap), jalan merah (Macet total). Hasil dari fungsi bitwise tersebut selanjutnya akan diproses lagi dengan menggunakan fungsi yang lain sehingga hasil akhir dari pembuatan kode pemrograman ini dapat tercapai.



Gambar 4.12 Hasil Dari Fungsi Bitwise

Gambar 4.12 menunjukan hasil dari penggabungan tiga warna jalan yang berbeda dengan menggunakan fungsi bitwise. Pada gambar dibagian atas merupakan bagian jalan yang terthreshod dengan warna yang berbeda. Sedangkan pada gambar kiri bawah merupakan hasil dari fungsi bitwise yang dapat dibandingkan dengan keadaan gambar jalan sebenarnya pada gambar kanan.

4.2.1.4 Crop Frame

Ketika kamera webcam diaktifkan untuk menangkap gambar, maka otomatis akan muncul windows keluaran *default* yang muncul pada layar laptop dengan resolusi 640x480 piksel. Ukuran tersebut kurang sesuai jika digunakan untuk memproses gambar jalan. Maka dari itu diperlukan sebuah windows baru dengan ukuran tertentu dan diambil dari windows sebelumnya yang mempunyai ukuran lebih kecil dari 640x480 piksel. Dalam penulisan kode pemrograman *crop image* didapat ukuran windows yang sesuai untuk jalan google maps. Nilai 180 dan 110 merupakan

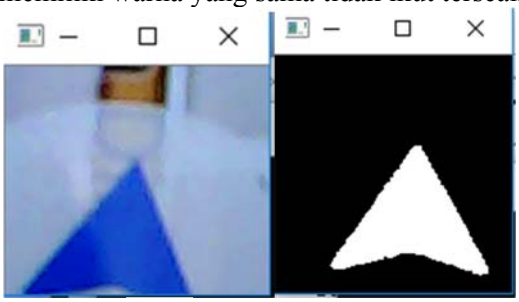
letak koordinat x,y yang mulai diambil dari ujung kanan atas pada windows 640x480 piksel. Sedangkan nilai 140,110 yaitu menjelaskan jarak sepanjang 140 pixel kearah sumbu x dan 110 pixel ke arah sumbu y. Hasil dari kode pemrograman yang dibuat yaitu berupa windows berukuran lebih kecil dengan resolusi sebesar 140x110 yang akan digunakan untuk menangkap gambar jalan pada *gps googlemaps.* Hasil gambar dari *crop image* sudah berupa gambar yang digabungkan dengan proses pengaturan HSV *(Hue, Saturation, and Value)*

```
//Crop windows

Rect myROI(325,50,150,130);
Mat frame2 = total(myROI);
Mat frame1 = total(myROI);
Mat frame3(frame1.clone());

Rect ROIrect(10,25,125,100);

Mat fillROI = frame1(ROIrect);

fillROI = Scalar(0);

Mat inverseFill(frame3.clone());

Mat inverseMask(inverseFill.size(), CV_8UC1,Scalar(1));

Mat inverseMaskROI = inverseMask(ROIrect);

inverseMaskROI = Scalar(0);

inverseFill.setTo(Scalar(0), inverseMask);
```

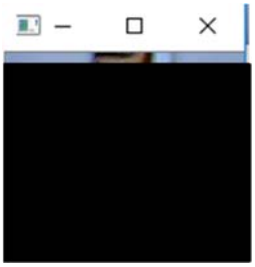
Gambar 4.13 Kode Pemrograman *Crop Image*

Gambar 4.13 menunjukan bagian dari kode pemrograman yang mengatur ukuran dari frame yang dibutuhkan mencari sudut jalan. Setelah ukuran windows sesuai dengan yang dibutuhkan proses selanjutnya yaitu menutup pointer dengan menggunakan InversMask sehingga ketika melakukan scanning pixel. Agar

pointer yang memiliki warna yang sama dengan jalan tidak ikut ter-*scanning*. Ukuran dari mask atau topeng yang menutupi pointer disesuaikan dengan ukuran pointer pada window hasil *crop image*. Sedangkan untuk mencari titik centroid pada segitiga pointer dapat menggunakan InverseFill untuk menutupi bagian jalan agar bagian jalan yang memiliki warna yang sama tidak ikut terscan.



A B



C

Gambar 4.14 (A) Hasil Crop Image, (B) Hasil InverseFill Untuk Mencari Titik centroid, (C) Hasil InverseMask Digunakan Untuk Mencari Titik Jalan

Gambar 4.14 menunjukan hasil gambar setelah mengalami proses crop image. Gambar A menunjukan hasil crop image sebelum tresholding. Gambar B menunjukan hasil dari inversefill untuk memisahkan gambar pointer dengan gambar jalan yang selanjutnya digunakan untuk proses pencarian titik

centroid.Gambar C menunjukan hasil dari inverse mask yang akan digunakan untuk mencari titik jalan.

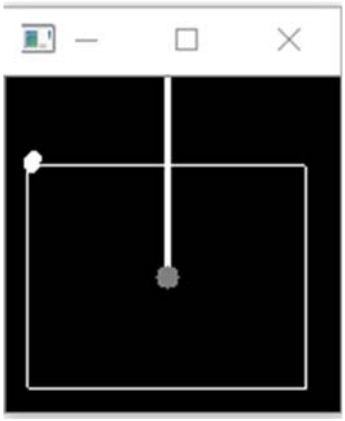
4.2.2 Centroid

Untuk mencari nilai dari centroid pada program ini dapat dicari dengan menggunakan fungsi moments yang ada pada opencv. Objek yang akan dilakukan proses pencarian centroid ini terlebih dahulu dirubah dalam bentuk objek biner.

```
//centroid
float sumx = 0;
float sumy = 0;
float num_pixel = 0;
for(int x=0; x<frame2.cols; x++)
{
    for(int y=0; y<frame2.rows; y++)
    {
        int val =frame2.at<uchar>(y,x);
        if( val >= 253)
        {
            sumx += x;
            sumy += y;
            num_pixel++;
        }
    }
}
```

Gambar 4.15 kode Pemrograman Pencarian Centroid

Gambar 4.15 menunjukan kode perograman pencarian centroid. Untuk mencari nilai dari centroid pada program ini dapat dicari dengan menggunakan fungsi moments yang ada pada opencv. Objek yang akan dilakukan proses pencarian centroid ini terlebih dahulu dirubah dalam bentuk objek biner. Karena pada prosesnya, untuk menentukan titik centroid, fungsi ini membaca satu-persatu piksel yang berwarna putih pada sebuah frame. Setelah didapat total keseluruhan lokasi piksel berwarna putih yang berada pada gambar, maka dengan otomatis, lokasi titik centroid dari objek tersebut dapat diketahui.



Gambar 4.16 Titik Berwarna Abu-Abu Menunjukan Titik Centroid Dari Pointer

Gambar 4.16 menunjukan hasil dari fungsi momen yang berfungsi sebagai penentu posisi pointer yang akan digunakan sebagai acuan pada titik jalan. Pada gambar diatas ditunjukan dengan titik berwarna abu-abu.

4.2.3 Scanning Pixel

Untuk mencari tau posisi dari jalan pada gambar dapat menggunakan proses scanning pixel. Namun pada opencv sendiri tidak ada fungsi yang dapat digunakan untuk melakukan scanning pixel sehingga perlu membuat suatu fungsi sendiri yang dapat digunakan untuk proses scanning.

```
//Scanning Pixel
for(int j=frame2.rows-1;j>=0;j--)
{
    for(int i=frame2.cols-1; i>=0;i--)
    {
        int scan1 = frame2.at<uchar>(j,i);
        if(scan1>253)
        {
            baris1=j;
            kolom1=i;
        }
    }
}
circle(frame2,Point(kolom1,baris1),2,Scalar(255,255,255),2,8 );

for(int j=frame2.rows-1;j>=0;j--)
{
    for(int i=0;i<frame2.cols;i++)
    {
        int scan2 = frame2.at<uchar>(j,i);
        if(scan2>253)
        {
            baris2=j;
            kolom2=i;
        }
    }
}
```

Gambar 4.17 Kode Pemrograman Scanning Pixel

Gambar 4.17 menunjukan kode pemrograman dari proses scanning pixel.. Pada code diatas scanning dimulai dari kiri ke kanan dan dimulai dari atas ke bawah. Dengan melakukan proses scanning, titik piksel yang digunakan dapat digunakan untuk mengetahui posisi jalan yang dilihat dari posisi x dan y nya.

Gambar 4.18 Gambar Lingkaran Hijau Menunjukan Titik Hasil *Scanning Pixel*

Gambar 4.18 menujukan titik dari hasil scanning pixel pada gambar jalan yang tertangkap kamera. Hasil titik pada scanning

pixel kemudian dirata-rata sehingga menghasilkan titik tengah pada jalan.

4.2.4 Perhitungan Sudut Jalan Google Maps

Untuk mendapatkan sudut jalan diperlukan setidaknya terdapat dua garis. Dari pertemuan dua garis pada titik centroid sebagai acuan sudut jalan dapat diketahui. Pada pembuatan sebuah garis dibutuhkan minimal 2 titik agar garis tersebut bisa tergambar pada windows.

```
//Perhitungan Sudut
float result1;

float x1 = (kolom3-sumx/num_pixel);
float y1 = (baris3-sumy/num_pixel);

result1 = atan(x1/y1) * 180/3.14; //kiri positif, kanan negatif

float result2;

float x2 = ((kolom1+kolom2)/2-sumx/num_pixel);
float y2 = ((baris1+baris2)/2-sumy/num_pixel);

result2 = atan(x2/y2)*180/3.14; //kiri positif, kanan negatif
```

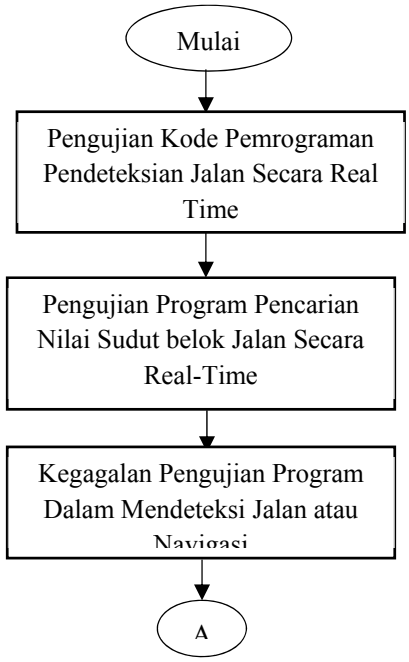
Gambar 4.19 kode pemrograman Perhitungan Sudut Jalan

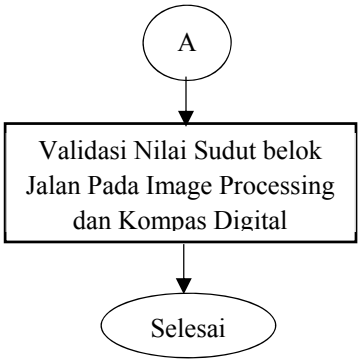
Gambar 4.19 menunjukan kode pemrograman untuk menghitung sudut jalan. Titik yang akan digunakan sebagai acuan adalah yang pertama titik centroid dan yang kedua adalah titik hasil scanning pixel pada jalan googlemaps. Garis alfa menggunakan centroid dan dihubungkan dengan titik jalan google maps. Sedangkan garis kedua adalah garis vertical yang dibentuk dari titik centroid dan titik centroid yang dikurangi 150 piksel searah sumbu y. Dengan dua garis yang didapat, maka dapat dicari nilai dari sudut jalan, dan hasil dari perhitungan sudut sudut yang didapat kemudian ditampilkan pada suatu windows baru. Sudut yang dihasilkan masih belum diketahui tingkat kebenarannya sehingga perlu dilakukan proses validasi. Pembahasan validasi akan dibahas pada bab 5.

BAB V
HASIL PENGUJIAN PROGRAM DAN PEMBAHASAN

5.1 Proses Pengujian Program

Untuk mencari sudut belok jalan pada citra GPS googlemaps diperlukan Perancangan program.perancangan program dimulai dengan dengan membuat program untuk mendeteksi jalan pada citra GPS googlemaps. Tujuan dari pembuatan program tersebut adalah agar kamera dapat mengenali jalan yang akan di tracking. Setelah jalan berhasil terdeteksi, langkah selanjutnya adalah membuat program yang digunakan untuk mencari nilai sudut yang dihasilkan jalan. Program yang dibuat perlu diuji tingkat akurasinya.



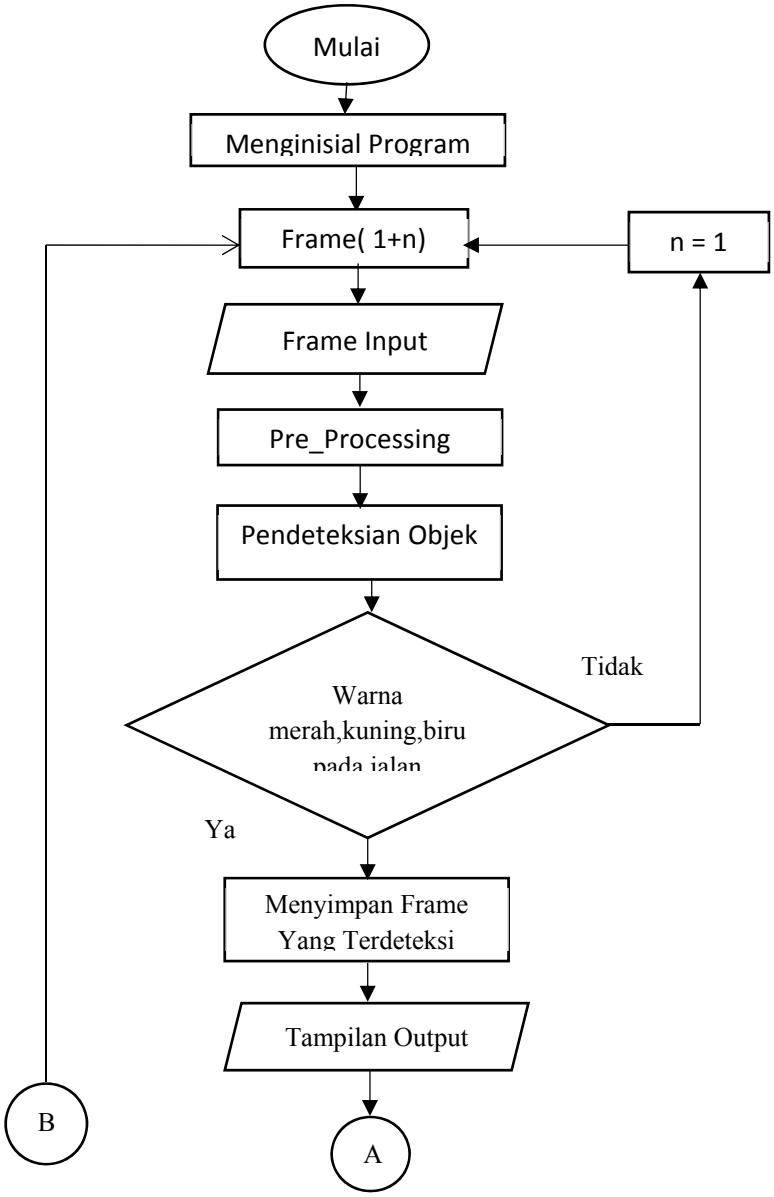


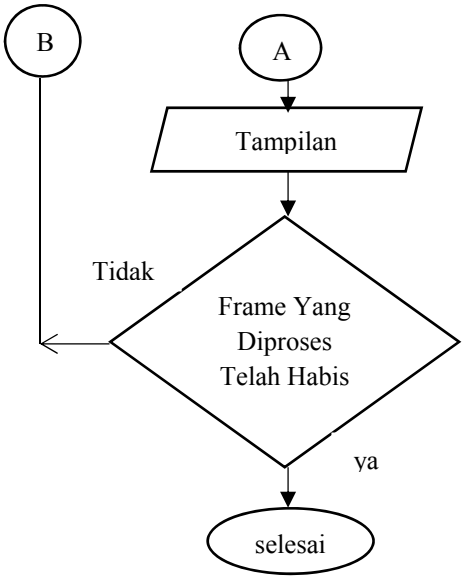
Gambar 5.1 Diagram Alir Proses Validasi Program

Gambar 5.1 menunjukan diagram alir langkah – langkah dalam melakukan validasi beserta membahas penyebab beberapa kegagalan dalam proses pendeteksian jalan. Proses pertama adalah pengujian kode pemrograman pendeteksi jalan secara real time. Berikutnya dilanjutkan proses pengujian program pencari sudut. Setelah itu pembahasan error pada pendeteksian gambar. Dan kemudian membahas validasi nilai sudut belok jalan pada image processing dan kompas digital

5.2 Pengujian Kode Pemrograman Pendeteksian Warna Jalan Secara Real Time

Proses pengujian kode pemrograman pendeteksian jalan dilakukan secara *real time*. Dimana pengambilan data langsung dilakukan dengan melihat pergerakan mobil secara *real time.* Benar atau tidaknya program dilihat dari mampu atau tidaknya mobil untuk berbelok. Sehingga perlu dilakukan pengujian program pengambilan gambar seperti yang ditunjukan pada diagram alir dibawah ini:


Mulai
Menginisial Program
Frame(1+n)
n = 1
Frame Input
Pre_Processing
Pendeteksian Objek
Warna
merah,kuning,biru
pada jalan
Tidak
Ya
Menyimpan Frame
Yang Terdeteksi
Tampilan Output
B
A



Gambar 5.2 Diagram Alir Proses Pendeteksian jalan Secara Real-Time

Gambar 5.2 menunjukan langkah-langkah dari pendeteksian jalan dari pengambilan gambar hingga output gambar yang berupa jalan yang telah di threshold. Langkah pertama dari pendeteksian jalan adalah menginisial program dengan cara menggabungkan program Microsoft studio dengan opencv versi 2.4.10. Untuk bahasa program yang digunakan adalah bahasa program C++ dan menggunakan library pada opencv. Kemudian langkah selanjutnya adalah mengaktifkan kamera. Gambar/frame yang ditangkap dari kamera selanjutnya diolah dengan cara mendeteksi warna di setiap jalan dengan mengubah format warna RGB menjadi HSV dan kemudian dilakukan proses threshold seperti yang dijelaskan pada program pengatur variable HSVyaitu dengan mengubah posisi dari trackbar hingga memperoleh hasil gambar atau frame hasil threshold yang bersih dan hanya terdapat sedikit nois .

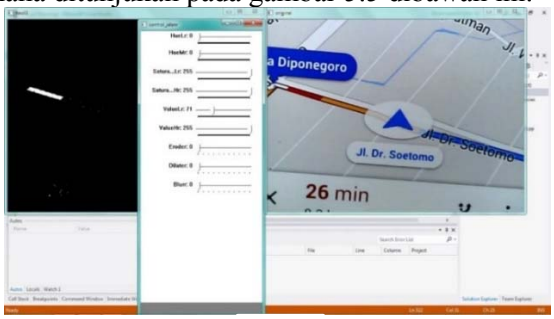
Sekanjutnya dilakukan proses preprocessing dimana pada proses ini dilakukan penghubungan antara library pada opencv dengan visual studio sebagai compiler. Kemudian dilakukan proses pendeteksian objek. Objek yang dideteksi adalah warna dari jalan yaitu merah, kuning dan biru. Apabila terdapat warna yang tidak dapat terdeteksi maka proses kembali melakukan penangkapan frame baru yang lebih jelas dan apabila sudah dapat mendeteksi keseluruhan warna jalan maka hasil frame yang terdeteksi tersebut disimpan. Bentuk dari gambar yang disimpan yaitu tampilan output yang berupa frame seperti yang ditunjukan gambar 5.3 dibawah ini. Apabila tidak ada input frame yang perlu diproses maka langkah proses selesai. Sedangkan apabila masih ada kembali ke pendeteksian gambar frame lagi.

A B C

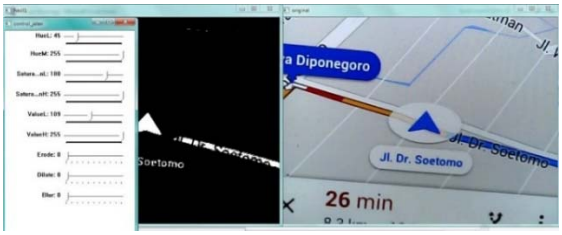
Gambar 5.3 Hasil Dari Pengaturan HSV Untuk Warna Merah, (B) Warna Kuning , Warna Biru

Gambar 5,3 menunjukan contoh hasil pengaturan sehingga didapatkan nilai optimal untuk hue, saturation dan value dari warna merah , kuning, dan biru. Nilai HSV dari masing-masing warna antara lain sebesar; 46 untuk hue low warna biru 255 untuk hue high warna biru,180 Untuk saturation low warna biru, 255 Untuk saturation high warna biru., 109 untuk value high warna biru,255 untuk value low warna biru sedangkan untuk warna kuning didapatkan nilai sebesar, 14 untuk hue low warna kuning 107 untuk hue high warna kuning ,158 Untuk saturation high warna kuning, 255 untuk saturation low warna kuning, 101 untuk value high warna kuning,255untuk value low warna kuning sedangkan pada warna merah didapatkan nilai sebesar 0 untukhue low warna merah 0 untuk hue high warna merah,255 Untuk saturation low warna

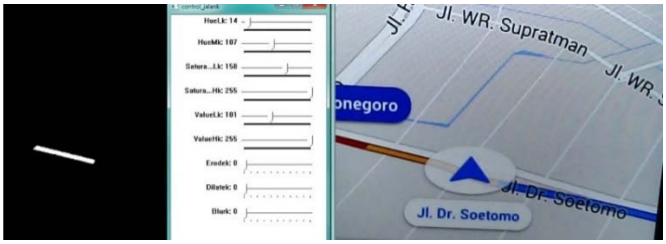
merah ,255 Untuk saturation high warna merah ,71 untuk value high warna merah dan ,255untuk value low warna merah. Sebagaimana ditunjukan pada gambar 5.5 dibawah ini.



A



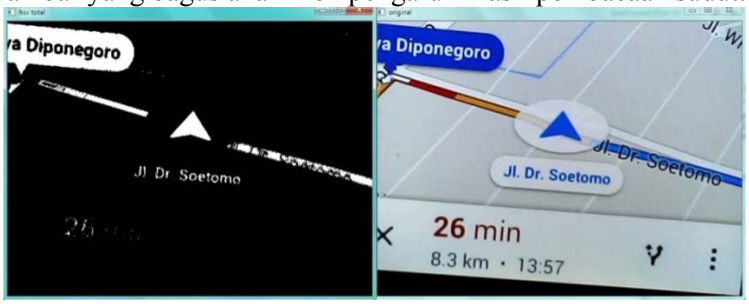
B



C

Gambar 5.4 Hasil Proses Pencarian Nilai Optimum Jalan Dari Beberapa Nilai HSV (a) Pada Warna Merah (b) Pada Warna Biru (c) Pada Warna Kuning

Gambar 5.4 Menunjukan nilai HSV optimum dari masing masing warna yang diperoleh dengan cara*try and error*. pengaturan tersebut berfungsi untuk memperjelas hasil dari proses threshold. Gambar yang bagus akan mempengaruhi hasil pembacaan sudut.

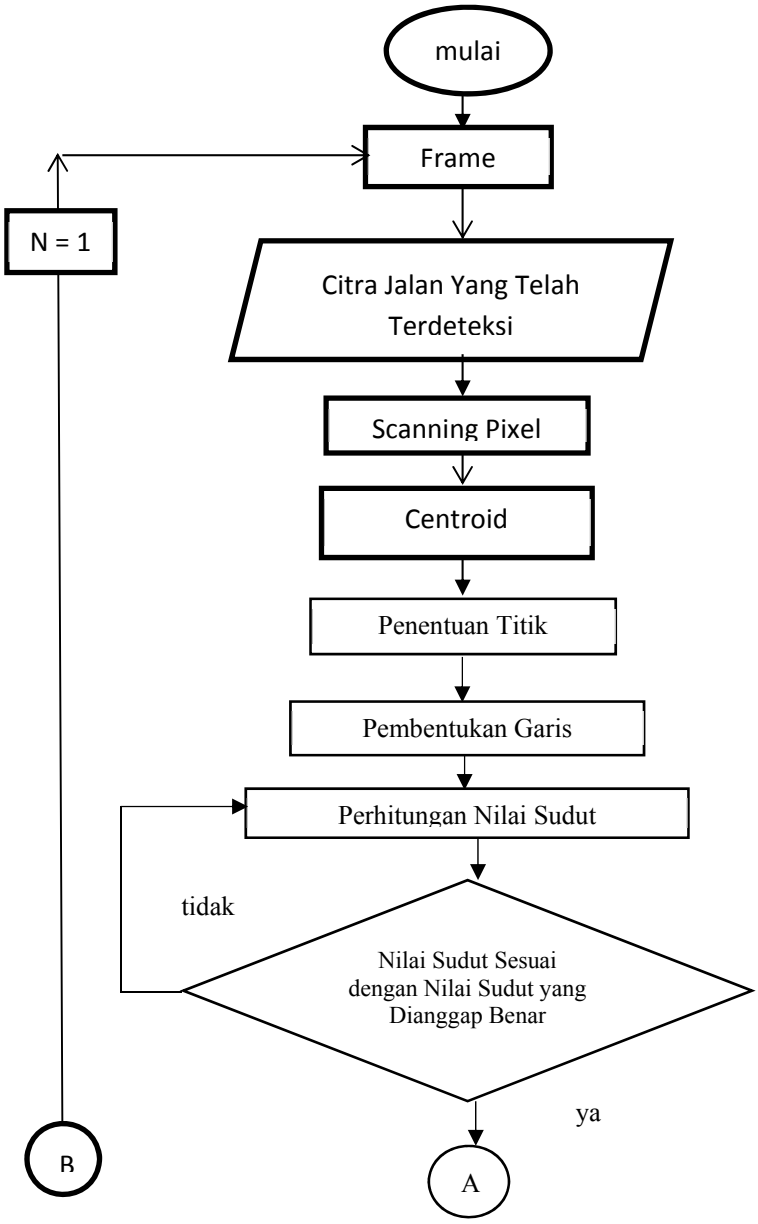


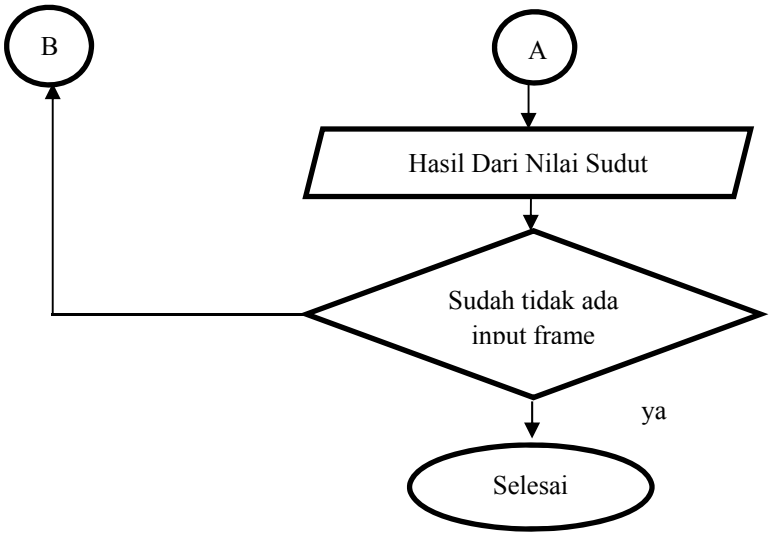
Gambar 5.5 Hasil Penggabungan Warna Jalan Dengan Menggunakan Fungsi Bitwise

Gambar 5.5 menunjukan hasil penggabungan warna jalan dengan menggunakan fungsi bitwise. Gambar hasil proses ditampilkan pada window baru. Jelas tidaknya gambar hasil penggabungan dipengaruhi oleh bagus tidaknya proses threshold.

5.3 Pengujian Program Pencarian Nilai Sudut Belok Jalan Secara Real-Time

Untuk menentukan arah yang akan dituju mobil maka perlu adanya program yang digunakan untuk mencari sudut belok jalan. Dibawah ini adalah diagram alir untuk mencari sudut yang di representasikan oleh jalan:


mulai
Frame
N = 1
Citra Jalan Yang Telah Terdeteksi
Scanning Pixel
Centroid
Penentuan Titik
Pembentukan Garis
Perhitungan Nilai Sudut
tidak
Nilai Sudut Sesuai dengan Nilai Sudut yang Dianggap Benar
ya
B
A



Gambar 5.6 Diagram Alir Langkah – Langkah Pencarian Nilai Sudut Dalam Program

Gambar 5.6 adalah urutan dari langkah – langkah yang dilakukan dalam pembuatan program pencarian sudut belok jalan mulai dari pendeteksian citra hingga didapatkan nilai sudut. Pertama yaitu menangkap gambar GPS dengan menggunakan kamera. Input berupa gambar atau Citra dari jalan yang telah terdeteksi kemudian diolah dengan mengubah format warnanya sebagaimana telah dijelaskan pada gambar diagram alir 5.2. Setelah format warna diubah kemudian melakukan proses threshold . Setelah berhasil men – threshold hal selanjutnya adalah melakukan scanning pixel untuk mencari posisi berwarna putih pada gambar. Setelah melakukan scanning pixel posisi jalan dapat diketahui. Kemudian memberi tanda pada kedua sisi jalan yang telah terscan tersebut. Setelah itu membagi dua nilai dari jarak tiap sisi jalan yang telah ter-scan untuk mendapatkan nilai tengah jalan. Setelah posisi titik tengah dapat diketahui selanjutnya adalah menghubungkan titik tengah jalan dengan centroid sehingga

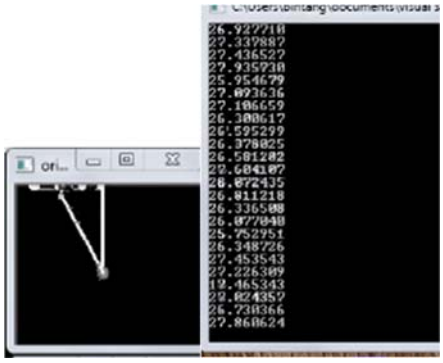
membentuk sebuah garis. Kemudian membuat garis vertical yang terhubung dengan itik centroid.Dalam mencari nilai sudut perlu penggambaran 2 buah garis, sehingga dapat diketahui nilai sudut yang akan dicari. Garis dapat terbentuk apabila terdapat minimal 2 titik. Hasil dari tangensial sudut yang terbentuk dari dua garis tersebut kemudian ditampilkan didalam *window.* Nilai dari tangensial tersebut memiliki toleransi sudut sebesar $\pm 2^{o}$ terhadap sudut yang ditunjukan dengan pengukuran menggunakan alat bantu busur. Hasil dari gambar disimpan. Apabila tidak terdapat gambar input lain maka proses diagram alir selesai. Sedangkan apabila masih terdapat gambar input maka kembali ke proses scanning pixel.

Untuk mencari nilai dari centroid pada program ini dapat dicari dengan menggunakan fungsi moments yang ada pada opencv. Objek yang akan diproses harus berada di dalam windos yang sudah melalui proses crop image. Karena gambar yang berada diluar daerah windows tidak ikut terdeteksi karena tidak ikut dihitung nilai dari pixelnya. Objek yang akan dilakukan proses pencarian centroid ini terlebih dahulu dirubah dalam bentuk objek biner. Proses mencari titik centroid berhubungan dengan proses scanning pixel karena pada prosesnya, untuk menentukan titik centroid diperlukan pembacaan pixel, fungsi ini membaca satu-persatu pixel yang berwarna putih pada sebuah windows. Setelah didapat total keseluruhan lokasi pixel berwarna putih yang berada pada gambar, maka dengan otomatis, lokasi titik centroid dari objek tersebut dapat diketahui.

Gambar 5.7 Titik Pada jalan Yang Berada Diluar Lingkaran(Ditunjukan Pada Lingkaran Hijau) Yang dihubungkan Dengan Centroid Pointer (Ditunjukan pada Lingkaran Merah)

Gambar 5.7 Menunjukan letak titik tengah yang diambil dari nilai dari dua titik hasil scanning pada ujung window (pada gambar ditunjukan dengan lingkaran hijau)yang terhubung dengan titik pusat jalan yang mengikuti titik pusat pointer (Ditunjukan pada Lingkaran Merah) sebagai acuan arah belokan di depan mobil.

Dalam membuat sebuah garis untuk mencari sudut belok jalan , opencv telah menyediakan fungsi line untuk membuat garis. Dalam penulisan program pada opencv dibutuhkan dua titik yang memiliki lokasi x1,y1 dan x2,y2 agar menjadi sebuah garis. Pada gambar 5.8 dibawah ini terdapat dua garis, yaitu garis yang menghubungkan titik tengah/centroid dari pointer dengan titik tengah jalan dan garis kedua adalah garis vertical berdasarkan titik tengah / centroid. Berdasarkan dua garis tersebut didapatkan nilai sudut belok jalan pada program dan ditampilkan dalam sebuah windows.



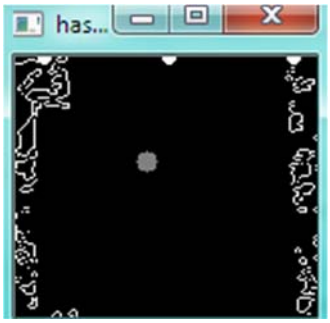
Gambar 5.8 Nilai Sudut Yang Didapat Antara Tangensial Garis Tengah Jalan Dengan Garis *Vertical*

Gambar 5.8 Menunjukan nilai Sudut yang didapat antara garis tengah jalan dengan garis vertical. Ketika garis jalan segaris dengan garis vertical maka nilai sudut belok jalan bernilai nol (0). Ketika garis tengah jalan disebelah kiri garis vertical maka akan bernilai positif sedangkan ketika garis jalan disebelah kanan garis vertical maka nilai sudut belok jalan akan bernilai negative. Waktu yang diperlukan untuk mengakses gambar dari webcam hingga mendapatkan nilai sudut belok pada gambar jalan adalah 10 frame untuk setiap satu detik.

5.4 Kegagalan Pengujian Program Dalam Mendeteksi Jalan atau Navigasi

Untuk mencari tahu tingkat pembacaan citra jalan dengan menggunakan metode image processing maka perlu dilakukan pengujian program. Pengujian program yang dilakukan adalah dengan melakukan uji coba pengambilan data secara outdoor dan secara real time, dimana rute jalan yang dilalui berupa jalan lurus, belok kanan dan belok kiri.

Kegagalan pembacaan jalan dapat disebabkan oleh garis tepi jalan yang tidak terhubung karena setting HSV yang tidak optimal . Selain itu posisi kamera dan handphone yang berubah pada saat uji coba, dimana perubahan posisi tersebut disebabkan oleh getaran pada mobil.



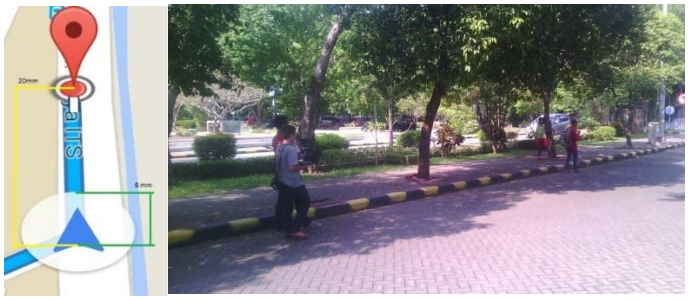
Gambar 5.9 Contoh Kegagalan Pada Proses Pendeteksian Warna

Gambar 5.9 menunjukan kegagalan yang terjadi ketika melakukan proses pendeteksian warna. Gambar yang dihasilkan kurang begitu jelas akibat pengaturan nilai HSV *(Hue, Satutation, and Value)* pada gambar. Selain itu kegagalan juga dapat disebabkan oleh adanya noise dengan bentuk tertentu yang ada pada gambar.

5.5 Perbandingan Nilai Sudut Belok Jalan Pada Image Processing dan Kompas Digital

Sebelum mengambil data untuk perhitungan sudut belok jalan, perlu membandingkan pengukuran jarak pada gps dengan jarak jalan yang sebenarnya. Hal ini dilakukan karena titik sudut yang diambil berada didepan mobil. Selain itu pada GPS googlemaps sering terjadi floating. Pada gps didapatkan bahwa jarak pusat segitiga dengan titik luar lingkaran yang diambil sebegai intepretasi titik sudut belok jalan adalah sepanjang 0.6 cm yang dihitung dengan menggunakan penggaris pada layar hp. Pengukuran yang dilakukan tidak menggunakan zoom in atau pun

zoom out atau dengan kata lain menggunakan ukuran yang sudah diberikan oleh GPS untuk mengurangi tingkat kesalahan pembacaan ketika mulai diaktifkannya mode navigasi. Pada gambar A dibawah terdapat gambar poin merah yang jika diukur dengan menggunakan penggaris memiliki panjang dua centimeter dan jarak perkiraan yang ditunjukan pada gps terhadap jarak sebenarnya adalah 40 meter sehingga jika dilakukan perhitungan maka didapatkan skala 1cm pada GPS Googlemap handphone setara dengan 20meter lalu sekala tersebut dikalikan dengan jarak pointer terhadap titik luar lingkaran putih sehingga jarak yang didapat adalah sebesar 12meter. Data yang didapat kemudian dibandingkan dengan memindahkan titik merah sepanjang 12 meter berdasarkan posisi awal yang sudah ditetapkan pada jalan yang lurus. Setelah itu dilakukan pengambilan data jarak sebenarnya dengan cara berjalan menuju poin berwarna merah hingga poin tersebut hilang. Dari sepuluh kali penggaambilan data jalan didapatkan data sebagaimana yang ditampilkan pada table 5.1 dibawah ini.



A B

Gambar 5.10 (A) Perbandingan Antara Panjang Jarak Centroid Dengan Titik Terluar Lingkaran dan Jarak centroid Dengan Titik Akhir dan (B) Pengambilan Data Panjang Jalan Sebenarnya

Gambar 5.10 menunjukan perbandingan jarak antara titik *centroid* dengan titik luar lungkaran dan jarak *centroid* dengan

tujuan akhir jalan. Panjang centroid menuju titik diluar lingkaran adalah 6mm sedangkan panjang titik acuan dengan tujuan akhir adalah 20mm. Pengukuran diambil pada jalan lurus sehingga dapat meminimalisir floating yang terjadi. Didapatkan skala gambar 1 : 2000 dimana skala tersebut mengacu pada jarak perkiraan yang muncul pada GPS googlemaps.

Dari sepuluh kali pengambilan data jarak didapatkan hasil seperti yang ditunjukan pada table dibawah ini:

Table 5.1: Perbandingan Data Jarak Jalan

No	GPS	Jarak Sebenarnya
1	12	11
2	12	13
3	12	13
4	12	14
5	12	14
6	12	12
7	12	12
8	12	11
9	12	9
10	12	11
Rata-rata	12	12

Tabel 5.1 menunjukan data hasil perbandingan antara jarak jalan pada GPS dan jarak sebenarnya. Didapatkan hasil yang sedikit berbeda dari kedua proses pengambilan data tersebut dikarenakan pembacaan gps yang masih kurang responsif untuk saat ini sehingga jarak yang diambil dapat lebih panjang dan bisa juga lebih pendek dari perhitungan panjang sebenarnya. Pada jarak yang sebenarnya didapatkan rata-rata dari nilai jarak sebesar 12 meter sehingga data pada GPS yang bernilai 12 meter dapat digunakan sebagai acuan jarak pada jalan.

Sudut belok jalan yang diperoleh dari program image processing merupakan inputan yang akan digunakan pada mobil

.Namun sudut yang dihasilkan masih belum diketahui tingkat kebenarannya. Oleh sebab itu perlu dibandingkan dengan kompas dimana posisi kompas berada dua belas meter didepan posisi mobil saat ini. Posisi tersebut merupakan posisi dimana titik pusat jalan yang terbaca oleh program. Proses pembandingan tersebut dilakukan secara real time.

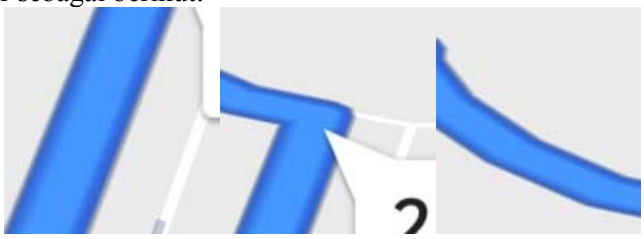
A B

Gambar 5.11 Perbandingan Nilai Sudut Pada (A) Kompas dan (B) Image Processing

Gambar 5.11 menunjukan hasil perbandingan nilai sudut kompas dengan sudut yang didapatkan dari centroid pointer segitiga (ditunjukan dengan lingkaran merah)yang terhubung dengan bagian luar lingkaran putih(ditunjukan pada lingkaran hijau) pada keadaan real.Dari nilai sudut yang dibandingkan tersebut kemudian diolah dengan metode statistic yaitu menggunakan metode paired sample T test.

5.7.1 Hasil Validasi Sudut Image Processing dengan Sudut Kompas

Pengujian sudut belok diambil data sudut pada jalan di daerah perumahan galaxy. Kontur jalan yang diambil memiliki kontur sebagai berikut:



A B C

Gambar 5.12 Rute Pengujian Sudut Belok Jalan (A) Lurus (B) Belok Kiri, (C) Belok Kanan

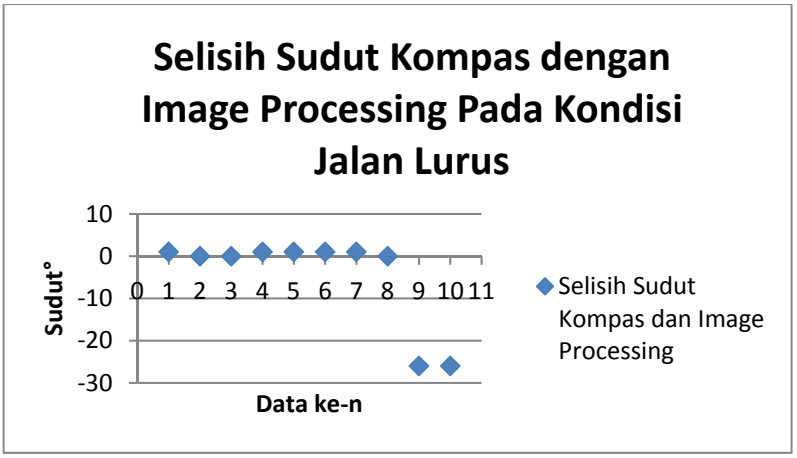
Gambar 5.12 menunjukan kontur jalan yang digunakan untuk pengujian sudut belok jalan. Titik mulai pengambilan data adalah jalan lurus setelah putar balik. Lalu bermelok kekiri dan dilanjutkan melewati bundaran kearah kanan. Pengambilan data dilakukan dengan berjalan kaki untuk meminimalisir getaran.

Table 5.2 Perbandingan antara data image processing dan data dari kompas .

No	Kompas	Image Processing	Di	Di^2	
1	1	0	1	1	Jalan Lurus
2	0	0	0	0	
3	0	0	0	0	
4	1	0	1	1	
5	1	0	1	1	
6	1	0	1	1	
7	1	0	1	1	
8	0	0	0	0	
9	0	26	-26	676	
10	0	26	-26	676	

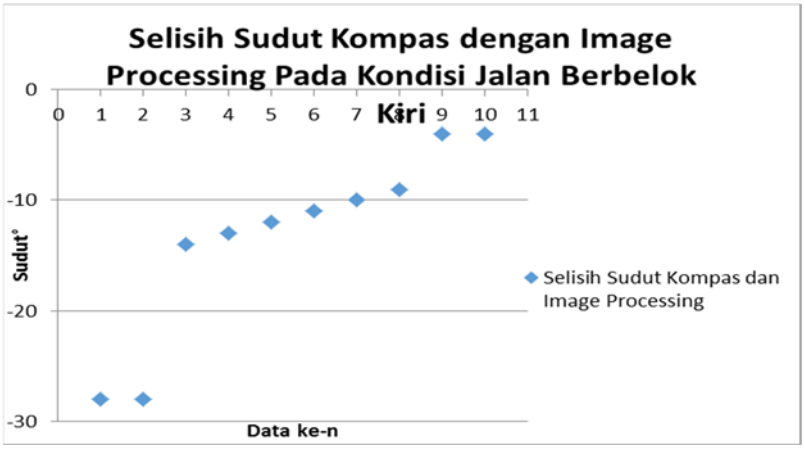
11	-2	26	-28	784	Belok Kiri
12	-2	26	-28	784	
13	12	26	-14	196	
14	13	26	-13	169	
15	14	26	-12	144	Belok Kiri
16	15	26	-11	121	
17	16	26	-10	100	
18	17	26	-9	81	
19	22	26	-4	16	
20	23	27	-4	16	
21	34	32	2	4	Belok Kanan
22	35	27	8	64	
23	36	34	2	4	
24	48	34	14	196	
25	50	32	18	324	
26	68	31	37	1369	
27	70	48	22	484	
28	0	-11	11	121	
29	3	-13	16	256	
30	3	-6	9	81	

Tabel 5.2 menunjukan perbandingan antara hasil pembacaan kompas dengan sudut yang diperoleh dengan image processing. Table nomor satu hingga sepuluh (ditunjukan dengan tabel warna hijau)diambil pada jalan yang lurus. Sedangkan data selanjutnya merupakan data yang diambil sesaat sebelum mobil berbelok ke kiri kemudian sedikit ke kanan sesuai rute. Pada table berwarna merah menunjukan data yang didapat dengan mengkompensasi sudut belok jalan dengan cara membandingkan sudut belok jalan sebelum berputar dan sudut jalan yang akan dilewati mobil dengan sudut yang ada pada kompas. Sehingga nilai dari sudut belok jalan setelah berputar sama dengan nilai sudut yang ditunjukan kompas.



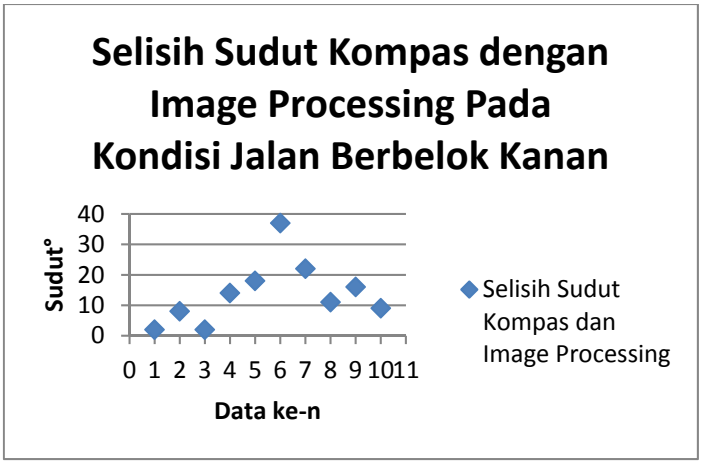
Gambar 5.13 Grafik Selisih Sudut Kompas Dengan Image Processing Pada Kondisi Jalan Lurus

Gambar 5.13 menunjukan grafik selisih sudut kompas dengan image processing pada kondisi jalan yang lurus. Pada hasil yang didapat terdapat sedikit perbedaan sudut yang disebabkan floating sudut yaitu sebesar satu derajat. Namun pada data ke Sembilan dan sepuluh menunjukan perbedaan yang cukup besar yaitu dua puluh enam derajat. Hal ini disebabkan oleh perbedaan jarak pada gps yang dijadikan acuan terhadap jarak yang terbaca pada image processing. Sudut dua puluh enam derajaat tersebut adalah hasil dari pembacaan rute belok yang ada di depan mobil.



Gambar 3.14 Grafik Sudut Kompas Dengan Image Processing Pada Kondisi Jalan Berbelok kekiri

Gambar 3.14 menunjukan grafik yang didapatkan pada jalan dengan kontur berbelok ke kiri. Pada hasil data yang didapat didapat hasil data negative yang cukup besar diawal daerah pengambilan data yang disebabkan oleh floating jarak yang terjadi pada daerah titik yang dijadikan acuan sehingga sudut pada image processing terbaca terlebih dahulu dibandingkan dengan sudut yang didapat dari kompas yang masih menunjukan nilai nol pada keadaan jalan sebenarnya. Kemudian terjadi penurunan data yang mengindikasi bahwa hasil data yang didapatkan mulai menunjukan hasil yang sama baik sudut dari image processing maupun pada kompas.



Gambar 5.15 Grafik Selisih Sudut Kompas Dengan Image Processing Pada Kondisi Jalan Berbelok Kekanan

Gambar 5.15 menunjukan selisih yang didapat dari sudut kompas yang dibandingkan dengan sudut image processing. Pada data menunjukan peningkatan sudut yang semakin tinggi namun kembali menurun pada data ke tujuh yang disebabkan oleh perubahab arah orientasi jalan. Nilai puncak menunjukan dimana posisi belok kritis pada jalan. Namun ketika sudah melewati titik kritis jalan data kembali mengecil. Lalu dari semua data tersebut dibandingkan dengan pengujian paired sample t-test yang bertujuan untuk membandingkan apakah hasil yang didapatkan dari image grocessing dapat dijadikan acuan. Dengan menggunakan hipotesa dari pair sample t-test sebagai berikut:

$$H_0 : \delta_0 = \delta$$
$$H_1 : \delta_0 \neq \delta$$

Dimana :

δ adalah selisih nilai sudut *image processing* dengan kompas dan δ_0 adalah 0.

Dan untuk mendapatkan nilai dari hipotesa dapat menggunakan rumus:

$d_i = X_{i1} - X_{i2}$……..……...……..(5.1)

$\bar{d} = \frac{\sum d_i}{n}$…….……...……(5.2)

$s_d = \left(\frac{\sum d_{i^2}}{n-1} - \frac{n}{n-1}\bar{d}^2\right)^{1/2}$ …….……..…….(5.3)

$t = \frac{(\delta-\delta_0)\sqrt{n}}{sd}$……..……(5.4)

Dimana:

d_i = Perbedaan sample1 dengan sample2

$\bar{d}$ = Rata – rata pebedaan atau rata – rata dari d_i

s_d = standart deviasi dari rata – rata perbedaan (d_i)

t = nilai t dari perhitungan pair sample

Dari perbandingan data sudut tersebut didapatkan nilai $\bar{d}$ sebesar 1,6, kemudian didapat nilai S_d sebesar 15,10 dengan nilai t sebesar 0,495. Dengan membandingkan nilai t yang didapat dari perhitungan dengan nilai t yang didapat dari table yaitu sebesar 2,045 dengan jumlah n sebanyak 30 maka hasil perbandingan tersebut menunjukan nilai yang lebih kecil. Apabila nilai dari t perhitungan lebih kecil maka hipotesa H_0 diterima dan menolak H_1. Sedangkan apabila nilai dari t perhitungan lebih besar dari nilai t yang didapatkan dari table maka hipotesa H_0 ditolak dan menerimaH_1 Karena nilai t perhitungan menunjukan nilai yang lebih kecil daripada nilai table maka dapat disimpulkan untuk menerima hipotesa H_0 dan menolak H_1. Dimana H_0 menerima nilai sudut yang didapat dari image processing sama dengan nilai sudut dari kompas.

BAB VIKESIMPULAN DAN SARAN
KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan yang dapat ditarik dari hasil penelitian tugas akhir ini adalah sebagai berikut:

- Program telah mampu mendeteksi tiga warna yang biasa ditunjukan *googlemaps* antara lain biru, kuning, dan merah. Sudut jalan secara statistik tidak berbeda secara signifikan dengan sudut yang ditunjukan oleh alat yang dianggap benar yaitu kompas.

6.2 Saran

Berdasarkan hasil pengujian, dan pembahasan yang sudah dilakukanpada tigas akhir ini dapat di ajukan saran untuk pengembangan penelitian selanjutnya vantara lain:

- Karena titik yang digunakan untuk mengambil data sudut jalan saat ini memiliki jarak yang terlalu jauh dari posisi titik yang dijadikan sebagai acuan posisi mobil maka diperlukan modifikasi program sehingga dapat mendeteksi warna jalan yang berada didalam lingkaran pointer sehingga jarak dari pengambilan data dengan jarak pointer lebih dekat. Dengan semakin dekat jarak titik jalan dengan centroid pointer maka sudut yang terbaca oleh mobil diharapkan akan lebih sesuai dengan keadaan jalan pada posisi mobil tersebut lalu perlu ditambahkan sensor lain yang dapat membantu mobil untuk mengetahui segala medan.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

Yan, J. 2012. ***A cloud-based design of smart car information services.*** Research Online,*University of Wollongong,* Australia.<URL:http://Uow.edu.au/pubs/research_online>.

Anderson, J.M. 2016 ***Autonomous Vehicle Technology.***<URL:http://www.rand.org/pubs/research_report/RR443-2.html>.

Domm, R. W. 2009. ***Michigan Yesterday and Today.***.Minneapolis. Voyageur Press. <URL:http//en.wikipedia.org/wiki/Ford_Model_T>.

Mclloy, J. 2016. ***Classier,Smarter New Nissan Juke Coming In 2017.*** <URL:http://www.autoexpress.co.uk/nissanjuke/92856/*Classier,Smarter New Nissan Juke Coming In 2017*>.

Read, R. 2013. ***Tesla Model S: So Save, It Broke NHTSA's Normal Testing Equipment.*** **<**URL:http://www**.**TheCarConnection. USA >.

Benz,M. 2016 .***The Mercedes-Benz F 105 Luxury In Motion.*** <URL: http://www.mercedes-benz.com/en/Mercedes-benz/innovation/research-f-105-luxury-in-motion>.

Rinaldi, M. 2004. **Pengantar Pengolahan Citra.** Teknik Elektro dan Informatika Institut Teknologi Bandung.Bandung.

Agustinus, N. 1997. **Pengolahan Gambar Secara Digital.** Elex Media Komputindo, Surabaya.

Pravin, V. 1993. ***IEEE TRANSACTIONS ON AUTOMATIC CONTROL.*** VOL. 38**,** NO. 2, Februari .

Erico, G. 2010. ***How Google's Self-Driving Car Works.***<URL:http://Spectrum.IEEE.org/automaton/robotic/artificial-intelegence/how-google-self-driving-car-work>.

Mahisaa,J.Y. Jan. 2013. **Aplikasi Pengolahan Citra.**<URL:http://Mahisaajy.blogspot.com/pdf/aplikasi_pengolahan_citra>.

BMW, 2014. ***Shift Pedal.***USA <URL:http//www.shopbmwusa.com/accecories/interior/shifters/lever_steering_wheel>.

Gregori, Eric. 2011. ***Introduction to Computer Vision Using Opencv.***<URL:http://www.embedded-vision.com/platinum-members/bdti/embedded-vision-training/documents/pages/introduction-computer-vision-using-opencv>.

Opencv. 2016. ***The Opencv Tutorial***. <URL:http//doc.opencv.org/2.4/open_tutorial/pdf>.

Robo,B .2005. ***Converting From RGB to HSV***. <URL:http://Webchace.googleusercontent.com/search/ece.illinois.edu/spring05/hsv_writeup>.

Regarnaburju. 2009. ***Bitwise_AND_Bitwise_OR_Bitwise_NOT_Bitwise_XOR***.

LAMPIRAN

TABEL B.5 t distribution

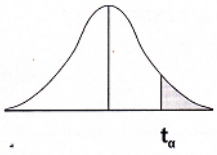


Table t Distribution

d.f.	$t_{.100}$	$t_{.050}$*	$t_{.025}$**	$t_{.010}$	$t_{.005}$	d.f.
1	3.078	6.314	12.706	31.821	63.657	1
2	1.886	2.920	4.303	6.965	9.925	2
3	1.638	2.353	3.182	4.541	5.841	3
4	1.533	2.132	2.776	3.747	4.604	4
5	1.476	2.015	2.571	3.365	4.032	5
6	1.440	1.943	2.447	3.143	3.707	6
7	1.415	1.895	2.365	2.998	3.499	7
8	1.397	1.860	2.306	2.896	3.355	8
9	1.383	1.833	2.262	2.821	3.250	9
10	1.372	1.812	2.228	2.764	3.169	10
11	1.363	1.796	2.201	2.718	3.106	11
12	1.356	1.782	2.179	2.681	3.055	12
13	1.350	1.771	2.160	2.650	3.012	13
14	1.345	1.761	2.145	2.624	2.977	14
15	1.341	1.753	2.131	2.602	2.947	15
16	1.337	1.746	2.120	2.583	2.921	16
17	1.333	1.740	2.110	2.567	2.898	17
18	1.330	1.734	2.101	2.552	2.878	18
19	1.328	1.729	2.093	2.539	2.861	19
20	1.325	1.725	2.086	2.528	2.845	20
21	1.323	1.721	2.080	2.518	2.831	21
22	1.321	1.717	2.074	2.508	2.819	22
23	1.319	1.714	2.069	2.500	2.807	23
24	1.318	1.711	2.064	2.492	2.797	24
25	1.316	1.708	2.060	2.485	2.787	25
26	1.315	1.706	2.056	2.479	2.779	26
27	1.314	1.703	2.052	2.473	2.771	27
28	1.313	1.701	2.048	2.467	2.763	28
29	1.311	1.699	2.045	2.462	2.756	29
inf.	1.282	1.645	1.960	2.326	2.576	inf.

There is only a 5% probability that a sample with 10 degrees of freedom will have a t value greater than 1.812.
*** one tail 5% α risk ** two tail 5% α risk**

Code

```
#include <opencv2\highgui\highgui.hpp>
#include <opencv2\imgproc\imgproc.hpp>
#include <iostream>
#include <windows.h>
#include <math.h>

using namespace std;
using namespace cv;

int baris1=0,kolom1=0;
int baris2=0,kolom2=0;
int baris3=0,kolom3=0;

Mat framek;
Mat hsvk;
```

```
Mat thdk;



int HLk=0;

int HHk=255;

int SLk=0;

int SHk=255;

int VLk=0;

int VHk=255;

int Erok=0;

int Dilk=0;

int Blurk=0;

Mat framer;

Mat hsvr;

Mat thdr;
```

```
int HLr=0;
int HHr=255;

int SLr=0;
int SHr=255;

int VLr=0;
int VHr=255;

int Eror=0;
int Dilr=0;
int Blurr=0;

Mat frame;
Mat hsv;
Mat thd;
```

```
int HL=0;
int HH=255;

int SL=0;
int SH=255;

int VL=0;
int VH=255;

int Ero=0;
int Dil=0;
int Blur=0;

int HL_jalan=0;
int HH_jalan=255;

int SL_jalan=0;
int SH_jalan=255;
```

```
int VL_jalan=0;
int VH_jalan=255;

int Ero_jalan=0;
int Dil_jalan=0;
int Blur_jalan=0;

Mat frame2;
Mat frame3;
Mat thr3;
Mat hsv3;
Mat thr2;
Mat hsv2;
float sudut_akhir;
int main ()
{
        VideoCapture cap (1);
```

```
        namedWindow("control_jalan",CV_WINDOW_NORMAL)
;


        cvCreateTrackbar ("HueL","control_jalan",&HL,255);

        cvCreateTrackbar ("HueM","control_jalan",&HH,255);


        cvCreateTrackbar
("SaturationL","control_jalan",&SL,255);

        cvCreateTrackbar
("SaturationH","control_jalan",&SH,255);


        cvCreateTrackbar ("ValueL","control_jalan",&VL,255);

        cvCreateTrackbar ("ValueH","control_jalan",&VH,255);


        cvCreateTrackbar ("Erode","control_jalan",&Ero,10);

        cvCreateTrackbar ("Dilate","control_jalan",&Dil,10);

        cvCreateTrackbar ("Blur","control_jalan",&Blur,10);


        VideoCapture capk (1);
```

```
        namedWindow("control_jalank",CV_WINDOW_NORMAL
);



        cvCreateTrackbar ("HueLk","control_jalank",&HLk,255);

        cvCreateTrackbar ("HueMk","control_jalank",&HHk,255);



        cvCreateTrackbar
("SaturationLk","control_jalank",&SLk,255);

        cvCreateTrackbar
("SaturationHk","control_jalank",&SHk,255);



        cvCreateTrackbar ("ValueLk","control_jalank",&VLk,255);

        cvCreateTrackbar
("ValueHk","control_jalank",&VHk,255);



        cvCreateTrackbar ("Erodek","control_jalank",&Erok,10);

        cvCreateTrackbar ("Dilatek","control_jalank",&Dilk,10);

        cvCreateTrackbar ("Blurk","control_jalank",&Blurk,10);



        VideoCapture capr (1);
```

```
        namedWindow("control_jalanr",CV_WINDOW_NORMAL
);


        cvCreateTrackbar ("HueLr","control_jalanr",&HLr,255);

        cvCreateTrackbar ("HueMr","control_jalanr",&HHr,255);


        cvCreateTrackbar
("SaturationLr","control_jalanr",&SLr,255);

        cvCreateTrackbar
("SaturationHr","control_jalanr",&SHr,255);


        cvCreateTrackbar ("ValueLr","control_jalanr",&VLr,255);

        cvCreateTrackbar
("ValueHr","control_jalanr",&VHr,255);


        cvCreateTrackbar ("Eroder","control_jalanr",&Eror,10);

        cvCreateTrackbar ("Dilater","control_jalanr",&Dilr,10);

        cvCreateTrackbar ("Blurr","control_jalanr",&Blurr,10);


        for (;;)

                {
```

```
                cap>> frame;

                cap>> framek;

                cap>> framer;


                cvtColor
(frame,hsv,COLOR_BGR2HSV_FULL);

                cvtColor
(framek,hsvk,COLOR_BGR2HSV_FULL);

                cvtColor
(framer,hsvr,COLOR_BGR2HSV_FULL);


                inRange(hsv,Scalar(HL,SL,VL),Scalar
(HH,SH,VH),thd);

        erode(thd,thd,getStructuringElement(MORPH_CROSS,Siz
e(Ero+1,Ero+1),Point(Ero,Ero)));


                inRange(hsvk,Scalar(HLk,SLk,VLk),Scalar
(HHk,SHk,VHk),thdk);

        erode(thdk,thdk,getStructuringElement(MORPH_CROSS,
Size(Erok+1,Erok+1),Point(Erok,Erok)));
```

```
                    inRange(hsvr,Scalar(HLr,SLr,VLr),Scalar
(HHr,SHr,VHr),thdr);

          erode(thdr,thdr,getStructuringElement(MORPH_CROSS,S
ize(Eror+1,Eror+1),Point(Eror,Eror)));

                    imshow ("hasil1",thd);

                    imshow ("hasil2",thdk);

                    imshow ("hasil3",thdr);

                    Mat res;

                    Mat total;

          bitwise_or(thd,thdk,res);

          bitwise_or(res,thdr,total);

                    //imshow ("or",res);

                    imshow ("hsv total",total);
```

```
//Crop windows

Rect myROI(325,50,150,130);
Mat frame2 = total(myROI);
Mat frame1 = total(myROI);
Mat frame3(frame1.clone());

Rect ROIrect(10,25,125,100);

Mat fillROI = frame1(ROIrect);

fillROI = Scalar(0);

Mat inverseFill(frame3.clone());

Mat inverseMask(inverseFill.size(),
CV_8UC1,Scalar(1));

Mat inverseMaskROI = inverseMask(ROIrect);
```

```
                inverseMaskROI = Scalar(0);


                inverseFill.setTo(Scalar(0), inverseMask);



                Canny(frame2,frame2,100,200,3);
                //Scanning Pixel
                for(int j=frame2.rows-1;j>=0;j--)
                {
                        for(int i=frame2.cols-1; i>=0;i--)
                        {
                                int scan1 =
frame2.at<uchar>(j,i);
                                if(scan1>253)
                                {
                                        baris1=j;
                                        kolom1=i;
                                }
                        }
                }
```

```
        circle(frame2,Point(kolom1,baris1),2,Scalar(255,255,255)
,2,8 );


                for(int j=frame2.rows-1;j>=0;j--)
                {
                        for(int i=0;i<frame2.cols;i++)
                        {
                                int scan2 =
frame2.at<uchar>(j,i);
                                if(scan2>253)
                                {
                                        baris2=j;
                                        kolom2=i;
                                }
                        }
                }


        circle(frame2,Point(kolom2,baris2),2,Scalar(255,255,255)
,2,8 );
```

```
        circle(frame2,Point((kolom1+kolom2)/2,(baris1+baris2)/
2),2,Scalar(255,255,255),2,8 );



                Canny(thr3,thr3,100,200,3);
                //centroid
                float sumx = 0;
                float sumy = 0;
                float num_pixel = 0;
                for(int x=0; x<frame2.cols; x++)
                {
                        for(int y=0; y<frame2.rows; y++)
                        {
                                int val =frame2.at<uchar>(y,x);
        if( val >= 253)
                                {
                                        sumx += x;
                                        sumy += y;
                                        num_pixel++;
```

```
                    }
                }
            }

            Point p(sumx/num_pixel, sumy/num_pixel);

            Moments m = moments(frame2, false);
            Point p1(m.m10/m.m00, m.m01/m.m00);



            //scanning pixel
            for(int j=frame2.rows-1;j>=0;j--)
            {
                for(int i=frame2.cols-1; i>=0;i--)
                {
                    int scan3 =
frame2.at<uchar>(j,i);
                    if(scan3>=253)
                    {
```

```
                                        baris3=j;
                                        kolom3=i;
                                    }
                                }
                            }

                            line(frame2, Point(sumx/num_pixel,
sumy/num_pixel) ,Point(sumx/num_pixel, sumy/num_pixel-200),
Scalar( 255,255,255 ),  2, 8 );
                            circle(frame2, p, 5, Scalar(128,0,0),-1);
                            //garis jalan
                            line(frame2, Point(sumx/num_pixel,
sumy/num_pixel),Point((kolom1+kolom2)/2,(baris1+baris2)/2),Sc
alar( 255,255,255 ),2,8);

                            //Perhitungan Sudut
                            float result1;

                            float x1 = (kolom3-sumx/num_pixel);
                            float y1 = (baris3-sumy/num_pixel);
```

```
                result1 = atan(x1/y1) * 180/3.14; //kiri positif,
kanan negatif


                float result2;


                float x2 = ((kolom1+kolom2)/2-
sumx/num_pixel);
                float y2 = ((baris1+baris2)/2-sumy/num_pixel);


                result2 = atan(x2/y2)*180/3.14; //kiri positif,
kanan negatif


                //kondisi 1
                if (result1>=0 && result2>=0 && result1>result2)
                {
                        sudut_akhir = result1-result2;
                }
                else if (result1>=0 && result2>=0 &&
result1<result2)
                {
                        sudut_akhir = (result2-result1)*-1;
```

```
                    }

                    //kondisi 2

                    else if (result1<=0 && result2<=0 &&
result1<result2)

                    {

                              sudut_akhir = result1-result2;

                    }

                    else if (result1<=0 && result2<=0 &&
result1>result2)

                    {

                              sudut_akhir = (result2-result1)*-1;

                    }

                    //kondisi 3

                    else if (result1>=0 && result2<=0)

                    {

                              sudut_akhir = result1-result2;

                    }

                    //kondisi 4

                    else if (result1<=0 && result2>=0)

                    {
```

```
                sudut_akhir = result1-result2;
            }

            printf("%f\n",result2);

            //imshow("original4",frame3);
            //imshow("original5",inverseFill);
            //imshow("original6",inverseMask);
            imshow ("hasil total",frame2);

                waitKey (1);
            }
            return 1;
}
```

BIODATA PENULIS

Penulis dilahirkan di Surabaya, 29 Oktober 1992, merupakan anak pertama dari dua bersaudara. Penulis telah menempuh pendidikan formal yaitu di TK Alfaroq, SD Bulak, SMPN 9 Surabaya, dan SMAN 9 Surabaya. Setelah lulus dari SMA pada tahun 2011 penulis mengikuti SNMPTN dan berhasil diterima di jurusan teknik mesin FTI-ITS dan terdaftar dengan NRP 2111100115.

Selama menempuh pendidikan di ITS penulis mengambil konsentrasi bidang studi Manufaktur dan menjadi anggota serta Kordinator Lab Perancangan dan Pengembangan Produk. Selama kuliah penulis berusaha menjadi Aktivis baik di dalam kelas maupun di luar kelas. Di luar kelas penulis aktif berorganisasi ditingkat kampus mulai dari Badan Eksekutif Mahasiswa sampai menjadi Sarjana LKMM dengan menyelesaikan LKMM TD. E-mail yang bisa dihubungi yaitu hendi.jansen@yahoo.com