VISUALISASI PERGERAKAN KENDARAAN BERMOTOR MENGGUNAKAN METODE DEPTH FIRST SEARCH (DFS)

SKRIPSI

Oleh:
LAILATUL LUTFIYAH
NIM. 11650002

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2015

VISUALISASI PERGERAKAN KENDARAAN BERMOTOR MENGGUNAKAN METODE DEPTH FIRST SEARCH (DFS)

SKRIPSI

Diajukankepada:
FakultasSainsdanTeknologi
Universitas Islam NegeriMaulana Malik Ibrahim Malang
UntukMemenuhi Salah SatuPersyaratanDalam
MemperolehGelarSarjanaKomputer (S.Kom)

Oleh:
LAILATUL LUTFIYAH
NIM. 11650002

JURUSAN TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2015

HALAMAN PERSETUJUAN

VISUALISASI PERGERAKAN KENDARAAN BERMOTOR MENGGUNAKAN METODE DEPTH FIRST SEARCH (DFS)

SKRIPSI

Oleh :

Nama : LailatulLutfiyah
NIM : 11650002
Jurusan : TeknikInformatika
Fakultas : Sains Dan Teknologi

TelahDisetujui, 8 Juni 2015

DosenPembimbing I	DosenPembimbing II
Dr. CahyoCrysdian	**A'laSyauqiM.Kom**
NIP. 19740424 200901 1 008	**NIP. 19771201 200801 1 007**

Mengetahui,

KetuaJurusanTeknikInformatika

Dr. CahyoCrysdian

NIP. 19740424 200901 1 008

HALAMAN PERNYATAAN

ORISINALITAS PENELITIAN

Saya yang bertandatangan di bawahini:

Nama : LailatulLutfiyah

NIM : 11650002

Fakultas/Jurusan : Sains Dan Teknologi / TeknikInformatika

JudulPenelitian :

VisualisasiPergerakanKendaraanbermotorMenggunakanMetode Depth Fisrt Search (DFS)

Menyatakandengansebenarnyabahwaskripsi yang sayatulisinibenar-benarmerupakanhasilkaryasayasendiri, bukanmerupakanpengambilalihan data, tulisanataupikiran orang lain yang sayaakuisebagaihasiltulisanataupikiransayasendiri, kecualidenganmencantumkansumbercuplikanpadadaftarpustaka. Apabila di kemudianhariterbuktiataudapatdibuktikanskripsiinihasiljiplakan, makasayabersediamenerimasanksiatasperbuatantersebut

Malang, 8 Juni 2015

Yang MembuatPernyataan,

LailatulLutfiyah

11650002

MOTTO

“Sedikit demi sedikit lama lamamenjadibukit”

“Sebaikbaikamalanadalah yang istiqomahdalammengamalkannya”

Silahkanmenyambungmaknadariduapepatahdiatas☺

Manusiahidupituterusberproses, dankarena proses itulahkitaakanmerasahidup

Dandari situ kitaakansadarbahwahidupitubutuhperjuangan

So, nikmatilahsetiap proses yang adadanmenangkandirimu.

Allah as always be with us

PERSEMBAHAN

Bismillah, denganmenyebutnama Allah penguasa ilmu semesta
Karyatulis ilmiah ini peneliti persembahkan untuk :

kedua orang tua tercinta
bapak Khamim dan ibu Khabibah
yang senantiasa mencurahkan segenap usaha dan do'a
yang telah mengajari kami ilmu kehidupan yang tak semua
orang mampu memberinya

Kepada kakak penulis, Siti sholikah, Asmaul Khusnah,
M. Amanudin, M. Al-Amin, Ridwan Hakim dan M. Yusuf
Afandi yang telah memberikan dukungan dan gagasan baru.
Semoga Allah senantiasa melancarkan segalaurusan kita,
Amin ...

KATA PENGANTAR

Alhamdulillahirobbil ‘alamainsegalapujibagi Allah yang telahmelimpahkanrahmatsertahidayat-Nyakepadapenulissehinggadapatmenyelesaikanskripsidenganjudul “VisualisasiPergerakanKendaraanBermotorMenggunakanMetode Depth First Search (DFS)” denganbaikdanlancar

Shalawatsertasalamsemogasenantiasatercurahkankepada NabiAgung Muhammad SAW yang telahmembimbingumatnyadarigelapnyakekufuranmenujucahaya Islam yang terangbenderang.

Penulismenyadariketerbatasanpengetahuan yang penulismiliki, karenaitutanpaketerlibatandansumbangsihdariberbagaipihak, sulitbagipenulisuntukmenyelesaikanskripsiini.Makadariitudengansegenapkerendahanhatipa tutlahpenulisucapkanterimakasihkepada:

1. BapakDrCahyoCrysdian,selakudosenpembimbing 1 dansekaligusKetuaJurusanTeknikInformatikayang telahmeluangkanwaktunyauntukmembimbing, mengarahkandanmemberimasukan yang sangatberpengaruhselamapengerjaanskripsiini.
2. BapakAlaSyauqi, M.Kom, selakudosenpembimbing 2 yang selalumemberimasukan, sertapengarahandalampnyusunanlaporanskripsiini.

3. Bapak, Ibukdansegenapkeluargatercinta yang selalumemotivasidanmendoakanselamapengerjaanskripsiini
4. Bapak Prof Dr. H. MudjiaRahardjo, M.Si, SelakuRektor UIN Maulana Malik Ibrahim Malang
5. IbukRirienKusumawati, M.Kom, selakudosenwali yang jugaselalumemberipengarahanterkaitakademikselamamasa study.
6. SegenapjajarandosenTeknikInformatikayang telahmemberikanbimbingankeilmuankepadapenulisselamamasa study
7. Teman- temanseperjuanganTeknikInformatika 2011, Alvin, Masiti, Izza, Wildan, Ulfa, Anshor, Hudan, Bagas dkk yang telah banyak membantu saya selama masa study, dan yang telahmemberimotivasidandorongansampaipengerjaanskripsiiniselesai

Sebagaipenutup, penulismenyadaridalamskripsiinimasihbanyakkekurangandanjauhdarikata sempurna.Semogaapa yang menjadikekuranganbisadisempurnakanolehpenelitiselanjutnya. Apa yang menjadiharapanpenulis, semogakaryainibermanfaatbagikitasemua. Jazakumullahahsalanjaza' Amin.

Malang, 8Juni 2015
Penulis,

LailatulLutfiyah

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Lutfiyah, Lailatul. 2015. **VisualisasiPergerakanKendaraanBermotorMenggunakanMetode Depth First Search (DFS)**. SkripsiJurusanTeknikInformatikafakultasSainsdanTeknologiUniversitas Islam NegeriMaulana Malik Ibrahim Malang.Pembimbing : (I) Dr. CahyoCrysdian (II) AlaSyauqi, M.Kom

Kata Kunci : Visualisasi, pergerakankendaraan, *Depth First Search* (DFS)

Teknologivisualisasikinimulai popular dikalanganmasyarakatumum.Denganvisualisasi orang akanlebihmudahmenyerapinformasi yang disampaikandaripadainformasi yang masihdalambentuksusunan data.Salah satunyaadalahvisualisasipergerakankendaraandimanavisualisasiinimenampilkanperge rakankendaraanpadaruasjalantertentu yang bertujuanuntukmemperlihatkankondisilalulintas yang sedangterjadi.Visualisasiinidapatdigunakansebagailangkahawalmenciptakantransport asi yang efektifdanefisiendimanajalurpergerakankendaraandapatmengantisipasiterjadinyamasa lahtransportasi.

Untukmembangunvisualisasipergerakankendaraaniniadalahdenganmenggunak anmetode Depth First Search (DFS) dimanametodeiniadalahsalahsatumetodepencarianbuta yang digunakanuntukmencariruteperjalanankendaraanberdasarkaninputanasaldantujuanken daraan.Adapunalurpembuatanaplikasiiniadalahdenganmenggunakan input data citrasatelit SRTM, *latitude longitude* danukuranfisikbangunan yang berasaldari Google Earth. Adapun parameter ujicobapadapenelitianiniadalahmengukurkecocokanjalur yang diperolehdanketepatanpergerakankendaraanberdasarkanasalsampaitujuan.Hasilakuras ijalurdiperolehadalah 100% sedangkanketepatanpergerakankendaraanberdasarkanjalurdiperolehdanmencarititikter dekatberdasarkantujuandiperolehakurasi 100%.

ABSTRACT

Lutfiyah, Lailatul. 2015. **Motor Vehicle Movement Visualization Method Using Depth First Search (DFS)**. Thesis Department of Computer Science Faculty of Science and Technology of the State Islamic University of Maulana Malik Ibrahim Malang. Advisors: (I) Dr. CahyoCrysdian (II) A'laSyauqi, M.Kom

Keywords: Visualization, the movement of vehicles, *Depth First Search* (DFS)

Visualization technology is now gaining popularity among the general public. With the visualization will be easier to understand information presented rather than information that is still in the form of the data set. One of them is the visualization of the movement of the vehicle in which visualization showing the movement of vehicles on certain roads which aims to show the traffic conditions is happening. This visualization can be used as an initial step creates an effective and efficient transport where track the movement of vehicles can anticipate the problem of transportation.
To build this vehicle movement visualization method is to use First Depth Search (DFS) where this method is one of the methods blindsearch used to search for vehicle travel route based on the input of origin and destination of the vehicle. The groove making this application is to use satellite imagery SRTM, longitude and latitude physical size buildings from Google Earth. The parameters tested in this study were obtained by measuring the suitability and appropriateness track the movement of vehicles based on origin to destination. The accuracy of the results obtained track is 100% while the accuracy of the movement of the vehicle based on acquired lane and look for the nearest point on the purpose acquired 100% accuracy.

ملخص البحث

لطفية,ليلة.٢٠١٥.المركباتذاتالمحركاتحركةالتصورالطريقةعنطريقالعمقأولادف سالبحث.شعبة الهندسة المعلوماتيّة.كلّيّة العامّة وتكنولوجيا.جامعة الإسلاميّة الحكوميّة مولانا مالك ابراهيم مالانج.

. مرشدالاول : جاهيو كريسديان الدكتور مرشد الثاني : اعلي شوقي الماجيستير

الكلمات الاساسية : دف سالتصور،وحركةالمركبات

تكنولوجياالتصورتكتسبالآنشعبيةبينالجمهورالعام.
معالتصورسيكونمنالأسهلعلىاستيعابالمعلوماتالمقدمةبدلامنالمعلوماتالتيلاتزالفيشكلمجموعةالبيانات.
واحدمنهمهوالتصورللحركةالسيارةالتيالتصورتظهرحركةالمركباتعلىبعضالطرقالتيتهدفلإظهارظروفحركةالمرورية
حدث. هذاالتصوريمكنأنتستخدمكخطوةأوليةيخلقالنقلالفعالوالكفءحيثتتبعحركةالمركباتيمكن.

أننتوقعلمشكلةالنقلحيثهذهالطريقةهي
دف سإحدىمعقالبحثلبناءهذهالسيارةطريقةحركةالتصورهواستخداماستخدامهاللبحثعنطريقالسفر السيارةعلىأساسالم
دخلاتالمنشأوالمقصد ليطلبالطرقأعمىبادخالبياناتالأقمار الصناعية، خطوط دف س منالسيارة.
الأخذودجعلهذاالتطبيقهواستخدامالصورالطولوالعرضالمبانيحجمالماديةمنجوجلالأرض.وقدتمالحصولعلىالمعلماتا
ختبار هافيهذهدقةالدراسةمنخلالقياسمدىملاءمةأثر الملاءمةحركةالمركباتعلىأساسوجهةالمنشأالنتائجالتيتمالتوصلالا
مسارهو 100٪ فيحينأندقةحركةالسيارةعلىأساسحارةالمكتسبةوالبحثعنأقربنقطةعنقصدحصلتعلى 100٪ منالدقة

BAB I

PENDAHULUAN

1.1 Latar Belakang

Ketua Bidang Advokasi Masyarakat Transportasi Indonesia (MTI) jawa timur menyatakan saat ini setiap ruas jalan di kota malang rata-rata dilintasi lebih dari 1.500 unit kendaraan, baik roda dua maupun roda empat dengan ruas jalan rata-rata antara 3, 3-5 meter hal ini membuat kapasistas jalan menjadi *over load* (Antara, 2013).

Sedangkan Menurut UU Republik Indonesia Nomor 14 Tahun 1992 Tentang Lalu Lintas dan Angkutan Jalan pasal 3 berbunyi Transportasi jalan diselenggarakan dengan tujuan untuk mewujudkan lalu lintas dan angkutan jalan dengan selamat, aman, cepat, lancar, tertib dan teratur, nyaman dan efisien, mampu memadukan moda transportasi lainnya, menjangkau seluruh pelosok wilayah daratan, untuk menunjang pemerataan, pertumbuhan dan stabilitas sebagai pendorong, penggerak dan penunjang pembangunan nasional dengan biaya yang terjangkau oleh daya beli masyarakat.

Keterangan yang dinyatakan oleh Ketua Bidang Advokasi MTI menjelaskan bahwa semakin banyaknya jumlah kendaraan yang masuk di kota malang. Hal ini terjadi karena pertumbuhan penduduk yang cukup tinggi dan berdampak pada kebutuhan mobilitas yang semakin tinggi pula. *Over load* memiliki arti dimana jumlah kendaraan melebihi kapasitas maksimal yang dimiliki oleh jalan dan

akibatnya adalah pergerakan kendaraan semakin melambat bahkan terhenti. Hal ini menandakan kemacetan sedang terjadi di ruas jalan tersebut.

Selain itu, adapun tujuan diselenggaraknanya transportasi salah satunya adalah mewujudkan lalu lintas yang aman, cepat, dan lancar. Tujuan tersebut belum sepenuhnya terrealisasikan, masih banyak terjadi kemacetan terjadi di sana sini terutama di kota kota besar yang memiliki sarana transportasi umum yang kurang memadahi dan lokasi tujuan pergerakan yang saling berdekatan misalnya pusat pendidikan yang bersebelahan dengan pusat perbelanjaan. Sedangkan menurut Peraturan derah Kota Malang tahun 2012 tentang Pengelolahan Dampak Lingkungan Hidup Kebisingan pasal 90 ayat 1c yang menyatakan setiap bangunan memiliki zona masing masing.

Malang, merupakan kota yang beberapa tahun belakangan ini menjadi kawasan yang sering terkena macet terutama di wilayah kota yang terdapat beberapa kampus. Dikutip dari media informasi ANTARA (2011) menyatakan bahwa di tahun 2015, malang akan mengalami macet total jika pemerintah tidak segera membenahi arus lalu-lintas baik secara fisik maupun rekayasa jalur kendaraan. Dilanjutkan (2012) menyatakan bahwa Malang merupakan salah satu tempat yang menjadi tujuan wisatawan berkunjung karena keindahan kotanya. Selain itu juga menjadi tujuan mahasiswa untuk menimba ilmu di kota ini. Hanya saja, pesatnya perkembangan kota dan pertumbuhan jumlah penduduk yang setiap tahun memadati kota malang belum mejadi perhatian serius pemkot setempat, sehingga tidak bisa di elakkan lagi kemacetan arus lalu lintas terjadi dan bahkan dari tahun ke tahun semakin parah.

Pemerintah sudah mencoba mencari solusi kemacetan yang terjadi salah satunya adalah dengan memberlakukan jalur satu arah yang khususnya di wilayah sekitar kampus, namun solusi itu justru membuat kemacetan yang terjadi semakin parah, Kemacetan yang semakin parah terjadi di kawasan Sukarno Hatta, jalan Mayjen panjaitan, jalan Bandung hingga Veteran dan jalan Bogor. Padahal sebelumnya didaerah itu khususnya bandung – veteran tidak pernah terjadi kemacetan parah. Berikut gambaran kemacetan yang terjadi di Jl. Soekarno Hatta dan sekitar Jl. Veteran :

(a) (b)

Gambar 1.1 (a) Macet pagi hari di Jl. Soekarno Hatta (Rabu, 24 februari 2015 pukul 06:57) (b) Macet sore hari di sekitar Jl. Veteran (selasa, 17 Maret 2015 pukul 17:01)

Transportasi saat ini menjadi sebuah permasalahan yang sangat kompleks. Karena kebutuhan manusia untuk mobilitas semakin tinggi. Saat ini sudah banyak jenis alat transportasi dengan teknologi canggih bermunculan, diantaranya adalah alat transportasi laut, darat dan udara. Pada zaman Nabi Muhammad s.a.w yaitu pada abad keenam Masehi, alat transportasi yang terkenal adalah unta, kuda,

keledai dan kapal layar atau perahu. Semuanya telah disebutkan dalam Al-Qur'an surat Az Zukhruf 43: 12

وَٱلَّذِى خَلَقَ ٱلْأَزْوَٰجَ كُلَّهَا وَجَعَلَ لَكُم مِّنَ ٱلْفُلْكِ وَٱلْأَنْعَٰمِ مَا تَرْكَبُونَ

Artinya : "*dan Dia (Tuhan) yang telah menciptakan segalanya berpasang - pasangan dan menjadikan untukmu kapal dan binatang ternak yang kamu tunggangi*" disusul dengan ayat Al-Qur'an surat An Nahl 16: 8, yang berbunyi :

وَٱلْخَيْلَ وَٱلْبِغَالَ وَٱلْحَمِيرَ لِتَرْكَبُوهَا وَزِينَةً ۚ وَيَخْلُقُ مَا لَا تَعْلَمُونَ

Artinya : *"dan (dia telah menciptakan) kuda, bagal (kuda yang lebih kecil) dan keledai, agar kamu kendarai dan menjadi perhiasan. dan Allah menciptakan apa yang kamu tidak mengetahuinya*". Maha benar Allah dengan segala firmannya.

Allah telah menciptakan unta, kuda, keledai dan kapal untuk memudahkan kehidupan manusia. Dengan berbagai alat transportasi konvensional ini , jarak yang jauh menjadi dekat dan keletihan berjalan kaki dapat diatasi. Hal-hal yang memberi kemudahan ini disisi lain menjadi bukti kepedulian Allah pada hamba-Nya. Alat-alat transportasi ini juga melahirkan seni dan estetika yang enak dipandang sehingga menjadi perhiasan hidup.

Menurut sebagian ahli tafsir kontemporer, ayat pada surat An-Nahl yang berbunyi "…dan Allah menciptakan apa yang tidak kalian ketahui." Memberi

isyarat akan adanya alat transportasi selain hewan dimasa depan yang belum diketahui pada zaman Nabi Muhammad s.a.w.

Pada masa sekarang telah terjadi revolusi alat transportasi yang amat sangat dahsyat. Kehidupan modern yang menuntut kecepatan dan ketepatan lebih mengandalkan alat transportasi mobil, kereta api, kapal api dan pesawat terbang yang tidak pernah tergambarkan di zaman Nabi dan sekarang merupakan kenyataan sehari – hari (rifyal, 2014).

Dengan adanya alat-alat transportasi ini akan lebih memberikan kemudahan dan kenyamanan pada manusia. Namun melihat realita saat ini, semakin banyak alat transportasi yang di gunakan malah menimbulkan dampak lain yang keluar dari tujuan diciptakannya alat transportasi, yaitu terjadinya kemacetan. Macet akan membuat perjalanan menjadi tidak lancar dan tidak nyaman. Semula dengan adanya alat transportasi membuat waktu tempuh menjadi lebih cepat, namun karena macet waktu tempuh menjadi lama.

Oleh karena itu diperlukan adanya penelitian yang dapat memperlihatkan kondisi lalu lintas dalam rangka memonitoring kemacetan yang terjadi khusunya di kota Malang dalam bentuk viusalisasi pergerakan kendaraan bermotor menggunakan metode Depth Fisrt Search (DFS).

Setelah dikaji dalam Al-Qur'an ternyata Allah SWT telah menyinggung tentang visualisasi pergerakan dalam surat Al-Isro' ayat 1 sebagai berikut :

سُبۡحَٰنَ ٱلَّذِيٓ أَسۡرَىٰ بِعَبۡدِهِۦ لَيۡلٗا مِّنَ ٱلۡمَسۡجِدِ ٱلۡحَرَامِ
إِلَى ٱلۡمَسۡجِدِ ٱلۡأَقۡصَا ٱلَّذِي بَٰرَكۡنَا حَوۡلَهُۥ لِنُرِيَهُۥ
مِنۡ ءَايَٰتِنَآۚ إِنَّهُۥ هُوَ ٱلسَّمِيعُ ٱلۡبَصِيرُ ١

Artinya : "*Maha suci Allah, yang telah memperjalankan hamba-Nya pada suatu malam dari masjidil haram ke masjidil aqsha yang telah kami berkahi sekelilngnya agar kami perlihatkan kepadanya sebagian dari tanda-tanda kebesaran (kami)*. Sesungguhnya Dia Maha mendengar lagi maha mengetahui".

Dari makana yang terkandung dalam surat Al-Isro' ayat 1 dapat kita ketahuui bahwasannya Allah telah menjelaskan tentang memperjalankan hamba-Nya (Nabi Muhammad) dari suatu tempat ke tempat yang lain (asal → tujuan) dengan alat transportasi yang digunakan nabi pada masa itu adalah hewan buroq. Dengan adanya cerita tentang perjalanan isro mi'raj sedikit banyak memberikan gambaran kepada kita semua bahwa setiap perjalanan pasti memiliki asal dan tujuan.

Diharapkan dengan adanya penelitian ini dapat membantu memberikan solusi permasalahan lalu lintas salah satunya adalah masalah kemacetan dengan dilihat dari sudut pandang visualisasi yang berdasarkan pada lokasi sebenarnya dan dengan analisa data dalam bentuk visualisasi sebagai langkah awal mengatasi permasalahan transportasi.

1.2 Identifikasi Masalah

Berdasarkan latar belakang yang telah dipaparkan, maka rumusan yang akan di jawab dalam penelitian ini adalah

1. Bagaimana membangun visualisasi kondisi lalu lintas di kota Malang menggunakan metode Depth First Search (DFS) ?
2. Seberapa baik kinerja metode DFS pada visualisasi pergerakan lalu lintas kendaraan bermotor di Kota Malang ?

1.3 Batasan Masalah

Mengingat luasanya wilayah Indonesia dan banyaknya jalan, maka pembahasan visualisasi pergerakan kendaraan ini dibatasi pada :

1. Penelitian ini memvisualisasikan kondisi lalu-lintas di area sekitar Jalan Veteran Kota Malang
2. Lokasi asal dan tujuan kendaraan telah ditentukan sebelumnya.
3. Pergerakan kendaraan melewati jalur yang sudah diperoleh menggunkan metode Depth First Search (DFS)
4. Untuk menentukan tujuan kendaraan harus lokasi yang melwati cabang berdasarkan loksi asal kendaraan

1.4 Tujuan Penelitian

Tujuan dilakukannya penelitian ini adalah :

1. Membangun visualisasi kondisi lalu lintas di kota Malang menggunakan metode Depth First Search (DFS)
2. Membuktikan bahwa metode Depth First Search (DFS) cocok digunakan untuk menunjang visualisasi pergerakan kendaraan bermotor di Kota Malang

1.5 Manfaat Penelitian

Manfaat yang peneliti harapkan dari penelitian ini adalah :

1. Penelitian ini diharapkan mampu memberi kontribusi terhadap pemerintah khususnya yang menangani tentang transportasi dan kemacetan
2. Memberikan informasi tentang visualisasi pergerakan kendaraan kondisi lalu lintas pada jalan veteran dan sekitarnya.
3. Memudahkan pihak yang membutuhkan informasi tentang visualisasi pergerakan kendaraan untuk memonitoring terjadinya kemacetan.
4. Bisa digunakan sebagai bahan masukan untuk peneliti selanjutnya.

1.6 Sistematika Penulisan

Penulisan skripsi ini tersusun dalam lima bab dengan sistematika penulisan sebagai berikut :

- BAB I Pendahuluan

  Bab ini berisi pembahasan tentang latar belakang masalah, identifikasi masalah, batasan masalah, tujuan penelitian, manfaat penelitian dan sistematika penulisan.

- BAB II Tinjauan Pustaka

  Bab ini berisi pembahasan tentang visualisasi pergerakan kendaraan, transportasi, kemacetan kota Malang, metode Depth First Search (DFS)

- BAB III Perancangan dan Implementasi

  Bab ini menjelaskan tentang perancangan dan implementasi visualisasi pergerakan kendaraan bermotor menggunakan metode Depth First Search (DFS) yang meliputi kebutuhan system, perancangan dan implementasi algoritma, *user interface* dan database.

- BAB IV Hasil dan Pembahasan

  Bab ini berisi pembahasan tentang pengujian aplikasi meliputi pengujian lokasi titik terdekat jalan pada bangunan, pengujian jalur yang dilewati, pengujian pergerakan kendaraan pada jalur yang telah ditemukan dan integrasi visualisasi pergerakan kendaraan bermotor dengan islam.

- BAB V Penutup

  Bab ini berisi tentang kesimpulan dan saran yang diharapkan dapat bermanfaat untuk penelitian selanjutnya.

BAB II

LANDASAN TEORI

2.1 Transportasi

Transportasi diartikan sebagai usaha pemindahan atau pergerakan dari satu lokasi ke lokasi lainnya dengan menggunakan alat tertentu. Dengan demikian transportasi memiliki dimensi seperti lokasi (asal dan tujuan), alat (teknologi) dan keperluan tertentu. System transportasi selalu berhubungan dengan ketiga dimensi tersebut, Jika salah satu dari ketiga dimensi tersebut tidak ada maka bukanlah termasuk transportasi. Transportasi juga dapat diartikan sebagai usaha memindahkan, menggerakkan, mengangkut, atau mengalihkan suatu objek dari satu tempat ke tempat lain, dimana di tempat lain objek tersebut lebih bermanfaat atau dapat berguna untuk tujuan-tujuan tertentu (Miro, 2005).

Salah satu langkah untuk menciptakan transportasi yang efektif dan efisien adalah perancangan pembangunan transportasi pada dasarnya menyelidiki kapasitas sarana dan prasarana transportasi yang diupayakan berimbang dengan kebutuhan masyarakat akan jasa transportasi. Penyediaan fasilitas yang berimbang dengan yang dibutuhkan itu tidak mudah dilakukan. Banyak factor yang harus diperhatikan (misalnya jumlah, penyebaran dan tingkat pertumbuhan penduduk pola perdagangan dan perjalanan penduduk, aspek regulasi dan lain sebagainya). Bila kapasitas fasilitas transportasi yang disediakan berlebihan akan mengakibatkan pemborosan sumber daya yang berarti penyelenggaraan transportasi tidak lancer dan efisien. Sebaliknya jika kapasitas fasilitas

transportasi dalam keadaan kekurangan akan menimbulkan kepadatang yang sangat tinggi dan kemacetan, yang berarti penyelenggaraan transportasi tidak efektif dan tidak lancer (Adisasmita SA, 2012). Maka dari itu diperlukan adanya visualisasi pergerakan kendaraan sebagai langkah awal membangun tranportasi yang efektif dan efisien

System transportasi merupakan suatu satuan dari elemen-elemen yang saling mendukung dalam pengadaan transportasi. Elemen-elemen transportasi tersebt adalah (Khisty *et. al*, 2003)

- Sarana penghubung (*link*) : jalan raya atau jalur yang menghubungkan dua titik atau lebih. Pipa, jalur darat, jalur laut, dan jalur penerbangan juga dapat dikategorikan sebagai sarana penghubung.
- Kendaraan : alat yang memindahkan manusia dan barang dari suatu titik ke titik lainnya di sepanjang sarana penghubung. Contohnya mobil, bus, kapal dan pesawat terbang.
- Terminal : titik-titik dimana perjalanan orang dan barang dimulai atau berakhir. Contohnya garasi mobil, lapangan parkir dan bandara udara.
- Manajemen dan tenaga kerja : orang-orang yang membuat, mengoperasikan, mangatur dan memelihara sarana penghubung, kendaraan dan terminal.

Keempat elemen tersebut berinteraksi dengan manusia, sebagai pengguna mapun non-pengguna sistem, dan berinteraksi pula dengan lingkungan.

2.1.1 Kendaraan Bermotor

Kendaraan bermotor adalah kendaraan yang digerakkan oleh peralatan teknik untuk pergerakannnya, dan digunakan untuk transportasi darat. Umumnya kendaraan bermotor menggunakan mesin pembakaran dalam (perkakas atau alat untuk menggerakan atau membuat sesuatu yang dijalankan dengan roda, digerakkan oleh tenaga manusia atau penggerak, menggunakan bahan bakar munyak atau tenaga alam). Kendaraan bermotor memiliki roda, dan biasanya berjalan di atas jalan (wikipwdia : kendaraan bermotor).

Berdasarkan UU No. 14 tahun 1992, yang dimaksud dengan peralatan teknik dapat berupa motor atau peralatan lainnya yang berfungsi untuk mengubah suatu sumber daya tertentu menjadi tenaga gerak kendaraan bermotor yang bersangkutan. Pengertian kata kendaraan bermotor dalam ketentuan ini adalah terpasang pada tempat sesuai fungsinya. Termasuk dalam pengertian kendaraan bermotor adalah kereta gandengan atau kereta tempelan yang dirangkaikan dengan kendaraan bermotor sebagai penariknya.

Menurut kantor Kepolisian Republik Indonesia, dapat dilihat pada table 2.1 data perkembangan jumlah kendaraan bermotor jenis tahun 1987 – 2013

Tabel 2.1 : perkembangan jumlah kendaraan bermotor

Tahun	Mobil Penumpang	Bus	Truk	Sepeda Motor	Jumlah
1987	1 170 103	303 378	953 694	5 554 305	7 981 480
1988	1 073 106	385 731	892 651	5 419 531	7 771 019
1989	1 182 253	434 903	952 391	5 722 291	8 291 838
1990	1 313 210	468 550	1 024 296	6 082 966	8 889 022
1991	1 590 750	504 720	1 087 940	6 494 871	9 582 138
1992	1 494 607	539 943	1 126 262	6 941 000	10 197 955
1993	1 700 454	568 490	1 160 539	7 355 114	10 784 597
1994	1 890 340	651 608	1 251 986	8 134 903	11 928 837
1995	2 107 299	688 525	1 336 177	9 076 831	13 208 832
1996	2 409 088	595 419	1 434 783	10 090 805	14 530 095
1997	2 639 523	611 402	1 548 397	11 735 797	16 535 119
1998	2 769 375	626 680	1 586 721	12 628 991	17 611 767
1999	2 897 803	644 667	1 628 531	13 053 148	18 224 149
2000	3 038 913	666 280	1 707 134	13 563 017	18 975 344
2001	3 189 319	680 550	1 777 293	15 275 073	20 922 235
2002	3 403 433	714 222	1 865 398	17 002 130	22 985 183
2003	3 792 510	798 079	2 315 781	19 976 376	26 613 987
2004	4 231 901	933 251	2 047 022	23 061 021	30 541 954
2005	5 076 230	1 110 255	2 875 116	32 528 758	37 623 432
2006	6 035 291	1 350 047	3 398 956	28 531 831	43 313 052
2007	6 877 229	1 736 087	4 234 236	41 955 128	54 802 680
2008	7 489 852	2 059 187	4 452 343	47 683 681	61 685 063
2009	7 910 407	2 160 973	4 452 343	61 078 188	67 336 644
2010	8 891 041	2 254 406	4 687 789	52 767 093	76 907 127
2011	9 548 866	2 250 109	4 958 738	68 839 341	85 601 351
2012	10 432 259	2 273 821	5 286 06	76 381 183	94 373 324
2013	11 484 514	2 286 309	5 615 494	84 732 652	104 118 969

Sumber : Badan Pusat Statistika Indonesia (2015)

2.1.2 Pergerakan

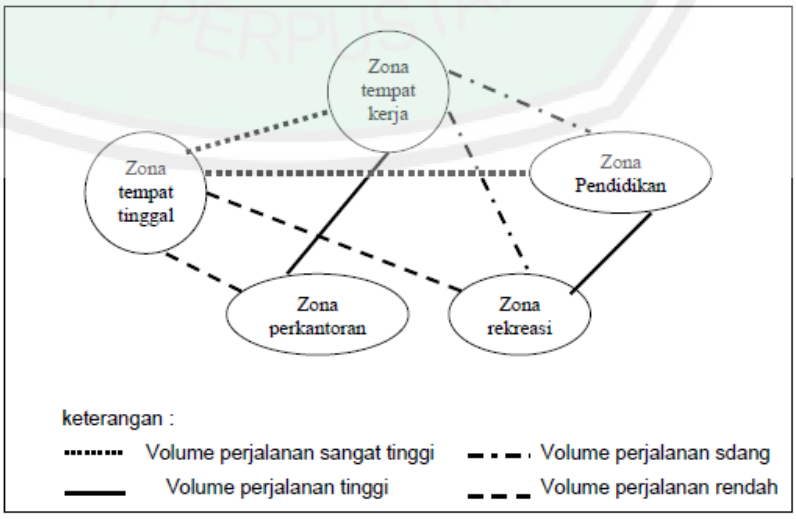
Alasan yang menyebabkan manusia dan barang bergerak dari satu tempat ke tempat lain nya dapat dijelaskan oleh tiga kondisi berikut (Khisty *et al*, 2005:09) :

1. *Komplementaritas*, daya tarik relative antara dua atau lebih tempat tujuan

2. Keinginan untuk mengatasi kendala jarak, di istilahkan sebagai *transferabilitas.* Diukur dari waktu dan uang yang dibutuhkan serta teknologi terbaik apa yang tersedia untuk mencapainya
3. Persaingan beberapa lokasi untuk memenuhi permintaan dan penawaran

Pola pergerakan dalam sistem transpotasi sering dijelaskan dalam bentuk arus pergerakan (kendaraan, penumpang, barang). Arus pergerakan tersebut mempunyai arah dan jumlah mengambarkan besarnya pergerakan penumpang. Arus ini bergerak dari zona asal ke zona tujuan didalam suatu daerah tertentu dan selama periode waktu tertentu. Dari pola perjalanan tersebut dapat ditentukan zona zona yang mengalami pergerakan tinggi, sedang dan rendah.

Pola sebaran pergerakan dapat digambarkan dengan garis keinginan (Desire Line). Garis keinginan adalah baris lurus yang menghubungkan asal dan tujuan sebuah pergerakan. Pola persebaran penduduk dapat dinyatakan dengan garis keinginan ditunjukan pada gambar 2.1 berikut (Tamin, 1993:130) :



Gambar 2.1 pola pergerakan antar zona yang berbeda dalam ruang kota.

Pada dasarnya, kemacetan terjadi akibat dari jumlah arus lalu lintas pada suatu ruas jalan tertentu yang melebihi kapasitas maksimum yang dimiliki oleh jalan tersebut. Peningkatan arus dalam suatu ruas jalan tertentu berarti mengakibatkan peningkatan kerapatan antar kendaraan yang dapat juga berarti terjadinya kepadatan arus lalulintas akan mengakibatkan antrian hingga terjadinya kemacetan lalu lintas.

Di dalam suatu perlalu-lintasan dikenal istilah lalu lintas harian (LHR) atau AADT (*Averafe Annual Daily Traffic*) yaitu jumlah kendaraan yang lewat secara rata-rata dalam sehari (24 jam) pada suatu ruas jalan tertentu, besarnya LHR akan mennetukan dimensi penampung jalan yang akan dibangun. Volume lalu lintas ini bervariasi besarnya, tidak tetap, tergantung waktu, variasi dalam sehari, seminggu maupun sebulan dan setahun. Di dalam suatu hari biasanya terdapat jam sibuk, yaitu pagi dan sore hari. Tetapi ada juga jalan-jalan yang mempunyai variasi volume lalu lintas agak merata.

2.2 Kemacetan di Kota Malang

Bertambahnya kendaraan bermotor di kota Malang berpotensi memunculkan masalah kemacetan yang luar biasa dalam beberapa tahun kedepan. Saat ini beberapa ruas di kota Malang kerap terjadi kemacetan dan antisipasi pelebaran tampaknya sulit dilakukan.

Sebagai salah satu kota pendidikan dan kota wisata di jawa timur, kota malang pada tahun 2015 mendatang, diprediksikan akan terjadi kemacetan total. prediksi tersebut dilihat dari kinerja pemkot yang sampai kini belum ada upaya

perancangan pembuatan jaringan jalan untuk tahun 2010-2030. Disisi lain, pertumbuhan kendaraan bermotor di kota Malang cukup tinggi, yaitu sekitar 10 persen. Meningkatnya jumlah kendaraan tersebut, hingga minggu ini sebanyak 11 ribu, yang terdiri dari 900 kendaraan roda dua dan 200 kendaraan roda empat. Bertumbuhnya jumlah kendaraan ini tidak diimbangi oleh pertambahan jalan yang memadai sehingga wajar jika terjadi kemacetan di beberapa ruas jalan di kota Malang (Media Center, 2013). Berikut dijelaskan pada table 2.2 volume kendaraan di beberapa titik di Kota Malang :

Tabel 2.2 Volume Kendaraan di Kota Malang

no	Lokasi	jenis kendaraan	jumlah kendaraan	Volume kendaraan
1	Jl.Soekarno Hatta	motor	54,6 %	4802 smp/jam
		mobil pribadi	44,2 %	3887 smp/jam
		truk	0,1 %	10 smp/jam
		bus	0,03 %	2 smp/jam
		MPU	0,8 %	71 smp/jam
		sepeda	0,1 %	12 smp/jam
		becak	0.03 %	3 smp/jam
2	Jl.MT.Haryono	motor	62,3 %	3811 smp/jam
		mobil pribadi	35,4 %	2165 smp/jam
		truk	0,1 %	8 smp/jam
		bus	0,03 %	2 smp/jam
		MPU	1,9 %	116 smp/jam
		sepeda	0,07 %	5 smp/jam
		becak	0,1 %	7 smp/jam
3	Jl. Simpang 3 borobudur	motor	55,4 %	3025 smp/jam
		mobil pribadi	40,9 %,	2335 smp/jam
		truk	0,1 %	7 smp/jam
		bus	0,02 %,	1 smp/jam
		MPU	3,3 %,	190 smp/jam
		sepeda	0,08 %,	5 smp/jam,
		becak	0,09 %.	5 smp/jam

4	Jl.LA.Sucipto	motor	47,7 %,	1584 smp/jam
		mobil pribadi	46,8 %,	1557 smp/jam
		truk	0,3 %,	10 smp/jam
		bus	0,06 %,	2 smp/jam
		MPU	4,8 %,	161 smp/jam
		sepeda	0,1 %,	5 smp/jam
		becak	0,2 %.	6 smp/jam
5	Jl.Gatot Subroto	motor	53,2 %,	2342 smp/jam
		mobil pribadi	42,1 %,	1853 smp/jam
		truk	0,7 %,	32 smp/jam
		bus	0,2 %	11 smp/jam
		MPU	3,2 %	142 smp/jam
		sepeda	0,1 %	6 smp/jam
		Becak	0.2 %	smp/jam

2.3 Algoritma Pencarian

Masalah pencarian berhubungan dengan bagaimana menemukan suatu nilai yang telah diberikan, yang di sebut kunci pencarian (search key), dalam suatu himpunan (set) yang telah diberikan. Ada banyak algoritma pencarian untuk dipilih. Mulai dari pencarian skuensial yang lugas sampai pencarian yang efisisen namun terbatas sebagai pencarian biner dan algoritma didasarkan pada adanya himpunan yang mendasarinya dalam bentuk yang berbeda yang lebih kondusif untuk ditemukan.

Dalam sebuah pencarian lintasan terdapat sebuah grafik yang secara informasi dianggap sebagai kumpulan titik dalam bidang yang disebut “vertex” atau “node”, beberapa dari mereka terhubung dengan segmen garis yang disebut “edges” atau “arcs”. Secara formal, suatu grafik G = (*V*,*E*) didefinisikan sebagai suatu pasangan dari dua himpunan, himpunan *V* dari item yang disebut verteks-

verteks dan suatu himpunan *E* dari pasangan item yang disebut garis-garis. Jika pasangan veteks (*u*,*v*) sama dengan pasangan (*v*,*u*) maka disebut garis (*u*,*v*) tidak berarah. Grafik G disebut tidak berarah apabila setiap garisnya tidak memiliki arah. Dan apabila sepsang verteks (*u*,*v*) tidak sama dengan pasangan (*v*,*u*) maka disebut garis (*u*,*v*) berarah dari verteks *u*, disebut ekor (*tail*), ke verteks *v* disebut kepala (*head*). Grafik G disebut berarah apabila setiap garis diberi arah.

2.3.1 Depth First Search (DFS)

Didalam proses mendapatkan solusi permasalahan *problem solving* yang diberikan, AI mempunyai beberapa kategori metode untuk menyelesaikannya. Salah satu dari kategori tersebut adalah Metode Searching yang mencakup beberapa searching algorithm yang umum digunakan dalam *problem solving*. Hal yang sangat menarik dari algoritma ini adalah bagaimana *searching algorithm* yang diterapkan berusaha mencari solusi, yang diistilahakan dengan *goal state (GS)*, paling optimal dan lengkap dengan parameter kompleksitas waktu dan ruang yang dihadapinya dari kondisi awal atau *initial state (IS)* yang diberikan

Algoritma DFS pertama kali diperkenalkan oleh Tarjan dan Hopcrof 20 tahun yang lalu. Mereka menunjukan bagaimana DFS dapat digunakan untuk membangun sejumlah algoritma graf yang efisien (Teneng dkk, 2010:58).

Algoritma DFS merupakan algoritma dengan pencarian buta (*blind search*). Blind search merupakan pencarian asal ketemu. Jika solusi sudah ditemukan, maka pencarian dihentikan. Jika dibuat dalam skemanya, pencarian buta hanya mengenal tiga bagian, [masalah]-[pencarian]-[solusi]. Atau dengan

kata lain *blind search* bisa dikatakan sebagai pencarian biasa karena tidak perlu informasi tambahan.

Depth first Search (DFS) melakukan pencarian secara preorder. Mengunjungi anak suatu simpul sebelum tetangganya. Algoritma DFS menggunakan metode pendekatan yang diimplementasikan dengan menggunakan tumpukan (*Stack*). Stack dalam struktur data berarti sturtur data yang organisasi atau strukturnya bersifat tumpukan atau menyerupai tumpukan.

Algoritma DFS memiliki kelebihan diantaranya adalah cepat mencapai kedalaman ruang pencarian. Jika diketahui bahwa lintasan permasalahan akan panjang maka DFS tidak akan meneroboskan waktu untuk melakukan sejumlah besar kedalaman 'dangkal' dalam permasalahan graf/pohon. DFS juga lebih efisien untuk ruang pencarian dengan banyak cabang karena tidak pernah meng evaluasi semua simpul pada suatu level tertentu pada daftar open. Selain itu, DFS memerlukan memori yang relatif sedikit.

DFS hanya menyimpan sekitar *bd* simpul, dimana *b* adalah factor percabangan dan *d* adalah kedalaman solusi. Jika b = 10 dan d = 3, maka jumlah simpul yang disimpan di memori adalah 1 + 10 + 10 + 10 = 31. Hal ini berbeda jauh dengan BFS yang harus menyimpan semua simpul yang telah dibangkitkan. Pada kasus tersebut BFS harus menyimpan 1 + 10 + 100 + 1000 = 1.111 simpul (Suyanto, 2011:17).

Selain kelebihan, DFS juga memiliki kelemahan, diantaranya adalah memungkinkan tidak ditemukannya tujuan yang diharapkan dan hanya akan mendapatkan satu solusi pada setiap pencarian.

Algoritma Depth First Search (DFS) atau dengan kata lain algoritma pencarian mendalam adalah teknik penelusuran graf yang mirip dengan algoritma BFS. Namun algoritma DFS melakukan penelurusan dengan mengunjungi secara rekrusif mendalam salah satu dari sejumlah simpul yang dibangkitkan sebelum akhirnya pindah ke anak simpul yang lain dari simpul akhir (Septiandi, 2011).

Menurut Adi Nugroho, pendekatan DFS biasanya diimplementasikan menggunakan tumpukan (Stack), sementara BFS diimplementasikan menggunakan antrian (Queue) (2008:450). *Stack* adalah daftar dimana penyisipan dan penghapusan dilakukan hanya di akhir dan titik akhir disebut sebagai puncak karena stack biasanya divisualisasikan secara vertical yang strukturnya beroperasi dengan model "masuk-terakhir-keluar-pertama" (*last-in-first-out*--LIFO).

Menurut Robert Laforce (1998) dalam Adi (2008) menyatakan algoritma *Depth First Search (DFS)* memiliki beberapa aturan, yaitu:

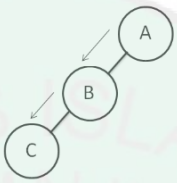
1. Lakukan kunjungan pada simpul-simpul pendamping/tetangga (*adjacent vertex*) yang belum dikunjungi, tandai dan masukkanlah (*push*) ke-*stack.* Aturan ini dapat diterapkan pada simpul-simpul sebelumnya. Pada langkah terakhir ini kita perlu melakukan sesuatu karena tidak ada simpul pendamping yang belum dikunjungi. Dalam hal ini kita bisa masuk ke aturan 2

2. Jika saat kita melakukan aturan di atas kita mengalami kesulitan, maka kita keluarkan *(poped off)* simpul dari*stack*. Mengikuti aturan ini, jika kita mengeluarkan suatu simpul dari suatu *stack* , kita akan sampai simpul di bawahnya. Jika simpul di bawahnya bukan simpul pendamping yang belum dikunjungi, kita keluarkan lagi. Demikian selanjutnya hingga kita tidak bisa melakukannya lagi dan kita harus masuk keaturan ketiga di bawah ini
3. Jika kita bisa mengikuti aturan 1 maupun 2, berarti kita telah menyelesaikan algoritma *Depth First Search.*

Berikut analisa ruang dan waktu yang digunakan dalam menggunakan metode Depth First Search (DFS)

1. Analisa Ruang
   - Setelah berjalan 1 langkah, stack akan berisi b node
   - Setelah berjalan 2 langkah, stack akan berisi (b-1)+b node
   - Setelah berjalan 3 langkah, stack akan berisi (b-1)+(b-1)+b node
   - Setelah berjalan d langkah, stack akan berisi (b-1)*d+1 node, mencapai maksimum
2. Analisa Waktu
   - Pada kasus terbaik, Depth First Search akan mencapai tujuan pada kedalaman d pertama sehingga dibutuhkan pencarian sebanyak d+1 node.

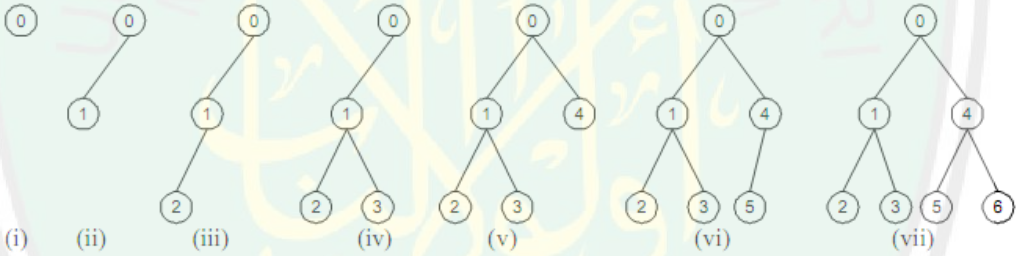
- Pada kasus terburuk, Depth First Search akan mencapai tujuan pada kedalaman d pada node terakhir, sehingga dibutuhkan pencarian sebanyak : $1+b+b^2+b^3+ \ldots+b^d = (b^{d-1}-1)/(b-1)$.



Gambar 2.2 : *Depth First Search*

(sumber : kusumadewi, 2003:26)

Berikut pembentukan pohon ruang status pada DFS



Gambar 2.3 : pembentukan ruang status pada DFS

(sumber : NUM-RN-MLK/IF2211/2013)

DFS menghasilkan dua urutan verteks-verteks , dimana vertex yang dicapai untuk pertama kalinya (didorong ke stack) dan urutan urutan vertex menjadi titik akhir (keluar dari stack). Dasar yang terpenting dari DFS termasuk memeriksa konektifitas dan memeriksa keberulangan grafik. Oleh Karen itu DFS berhenti setelah mengunjungi semua vertex yang terhubung oleh jalur ke vertex awal.

2.4 Penelitian Terkait

Pipin Mega Ayuning Tyas (2010) membahas tentang bagaimana mengimplementasikan algoritma backtracking dengan menggunakan metode Depth First Search (DFS) pada penyelesaian traveling salesman problem suatu diagraph. Penelusuran dimulai dengan membangkitkan simpul awal kemudian dibangkitkan simpul berikutnya yang merupakan solusi. Solusi ini ditentukan oleh fungsi pembatas yang telah ditentukan sebelumnya. Apabila bukan solusi maka simpul tersebut tidak diperhitungkan atau akan dimatikan dan akan dilakukan backtracking ke simpul sebelumnya sampai semua simpul telah dibangkitkan dan ditemukan solusinya.

Sheila Eka Putri (2011) membahas tentang pencarian lintasan terpanjang dengan menggunakan metode Depth Fisrt Search (DFS) dengan menggunakan struktur data Stack saat mencapai suatu simpul atau vertex yang terhubung dalam suatu graph. Dijelaskan disana bahwa kemampuan DFS dengan menggunakan Stack mampu menemukan simpul-simpul yang belum dikunjungi, hal ini memudahkan pencarian optimum dalam suatu persoalan salah satunya persoalan lintasan terpanjang (*longest path problem*). Selain itu juga membahas tentang implementasi suatu program sederhana dan menganalisis algoritma DFS dengan java yang bertujuan untuk memperoleh solusi optimum dari persoalan lintasan terpanjang pada graph. Hasil yang diperoleh adalah solusi optimum dengan nilai maksimum pada persoalan lintasan terpanjang.

Esa Fauzi (2013) membahas tentang visualisasi perjalanan kereta api dengan menggunakan pendekatan semaphore, deadlock solution dan algoritma dijkstra. Penelitian Esa menghasilkan perangkat lunak visualisasi menggunakan objek 2D. Hasil penelitiannya menunjukan bahwa perangkat lunak ini dapat mengatasi masalah tabrakan kereta api dan deadlock. Perangkat lunak visualisasi perjalan kereta api oleh Esa dibangun mengguanakan pendekatan semaphore, deadlock solution dan algoritma dijkstra. Metode pengembangan perangkat lunak yang digunakan adalah metode *iterative incremental*. Metode ini dimulai dari tahap perencanaan awal (*planning*) sampai dengan *deployment* yang didalamnya terdapat perputaran interaksi antara *planning*, analisis, *design*, implementasi serta *testing*.

2.5 Euclidean Distance

Menurut Tarecha (2013) *Euclidean distance* adalah sebuah metode untuk mengukur panjang antara dua titik, *euclidean distance* antara **p** dan **q** adalah panjang ruas garis yang menghubungkan antar keduanya (pq). Perhitungan *euclidean distance* ada dua macam, yang pertama yaitu perhitungan *euclidean distance* satu dimensi, yaitu menghitung jarak tipa poros *x* dan *y* satu persatu. Rumus *euclidean distance* satu dimensi ditunjukan pada rumus 2.1 berikut :

$$dx = |x1 - x2|$$

$$dy = |y1 - y2|$$

Dimana :

dx = *distance* / jarak *x1* ke *x2*

dy = *distance* / jarak *y1* ke *y2*

x1 = titik koordinat *x1* pada poros x

x2 = titik koordinat *x2* pada poros x

y1 = titik koordinat *y1* pada poros y

y2 = titik koordinat *y2* pada poros y

Sedangkan untuk mengitung rumus *euclidean distance* dua dimensi diamana menghitung kedua titik pada poros *x* dan *y* secara bersamaan maka menggunakan rumus *euclidean distance* dua dimensi yang ditunjukan pada rumus 2.2 berikut :

$$d(p,q) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}$$

Dimana :

d(p,q) = *distance* / jarak

x1 = titik koordinat p pada poros x

x2 = titik koordinat q pada poros x

y1 = titik koordinat p pada poros y

y2 = titik koordinat q pada poros y

BAB III

METODOLOGI PENELITIAN

3.1. Objek Penelitian

Pada penelitian kali ini peneliti membahas tentang Visulisasi Pergerakan Kendaraan Bermotor Menggunakan Metode Depth First Search (DFS). DFS merupakan salah satu metode searching. DFS digunakan untuk melakukan pencarian rute jalan pada pergerkan mobil dari lokasi asal menuju lokasi tujuan. Dengan banyaknya jumlah mobil yang akan digerakkan secara bersamaan dengan asal dan tujuan masing masing berbeda, diharapkan metode DFS bisa menjadi solusi pencarian yang efektif. Pergerakan kendaraan pada penelitian ini tidak memperlihatkan situasi jalan dalam artian tidak memperlihatkan keadaan seperti lampu merah karena hanya memvisualisasikan pergerakan kendaraan atau dalam istilah lain adalah pencarian rute terpendek.

Adapun objek penelitian ini di khususkan pada Jl. Veteran kota Malang. Berikut gambar objek penelitian yang di peroleh dari *GoogleEarth* :

Gambar 3.1 Lokasi Objek Penelitian Jl. Veteran
(Sumber : *GoogleEarth*, 2014)

3.2. Sumber Data

Sumber data yang digunakan pada penelitian ini adalah :

1. Data elevasi permukaan bumi SRTM

Data elevasi permukaan tanah dengan menggunakan satuan .dpl digunakan untuk menampilkan permukaan tanah.

2. Data koordinat jalan dan bangunan, yang diperoleh dari GoogleEarth

a. Data Jalan

Data koordinat jalan yang diperoleh dari GoogleEarth adalah titik longitude dan latitude bangunan dan jalan. Berikut struktur data jalan ditunjukan pada tabel 3.1

Tabel 3.1 Tabel Jalan

No	Nama	Tipe
1	Latitude_awal	double
2	Longitude_awal	double
3	Latitude_akhir	double
4	Longitude_akhir	double
5	Lebar	int
6	Index	int

Pada table 3.1 selain meiliki *field* longitude dan latitude juga memiliki *index.* Index digunakan untuk memudahkan membaca data koordinat *lat lang*. selain Tipe data pada latitude_awal dan longitude_awal atau latitude_akhir dan longitude_akhir berupa double atau bukan nilai yang unik, setiap nilai hampir memiliki nilai yang sama. Jadi untuk memudahkan membaca data koordinat pada

jalan peneliti menggunakan nilai *index*. Tabel Jalan ini nantinya digunakan untuk menggambar atau menampilkan jalan dan lokasi mobil melakukan pergerakan.

b. Data Bangunan

Sama halnya dengan data jalan, data bangunan juga membutuhkan koordinat berupa longitude dan latitude yang peneliti peroleh dari GoogleEarth untuk membuat bangunan terletak sama pada lokasi sebenarnya. Berikut struktur data bangunan ditunjukan pada tabel 3.2

Tabel 3.2 Tabel Bangunan

No	Nama	Tipe
1	Latitude	Double
2	Longitude	Double
3	Panjang	Double
4	Lebar	Double
5	Tinggi	Double
6	Sudut	Double
7	Nama_bangunan	Char
8	Index	Int

Pada tabel 3.2 terdapat *field* latitude dan longitude digunakan untuk meletakkan bangunan sesuai dengan lokasi sebenarnya, dan *field* panjang, lebar dan tinggi digunkan untuk ukuran bangunan secara fisik, adapun satuan yang digunakan adalah meter. Sudut digunakan untuk posisi hadap bangunan berdasarkan jalan, Nama_bangunan digunakan untuk penamaan pada data untuk mempermudah peneliti mengingat bangunan tersebut dan Index digunakan sebagai nilai yang akan digunakan sebagai asal dan tujuan kendaraan, karena itu mengapa index bertipe integer. Selain untuk menampilkan visualisasi bangunan,

data bangunan ini nantinya diguanakan sebagai lokasi asal dan juga sekaligus loksai tujuan pergerakan kendaraan.

3. Data mobil

Data mobil berisi data ukuran fisik mobil dan asal tujuan mobil. Berikut struktur data mobil ditunjukan pada tabel 3.3

Tabel 3.3 Tabel Mobil

No	Nama	Tipe
1	Panjang	double
2	Lebar	Double
3	Tinggi	Double
4	Sudut	Double
5	Index	Int
6	Asal	Int
7	Tujuan	Int

Pada tabel 3.3 ditunujukan bahwa setiap mobil yang akan digambar dengan ukuran panjang, lebar tinggi dan sudut juga telah ditentukan asal dan tujuan. Adapun nilai asal dan tujuan mobil mengacu pada *filed* index pada tabel bangunan.

4. Data Pergerakan (arah jalur)

Data pergerakan digunakan untuk menentukan jalur node menuju node yang lain, atau dengan kata lain data pergerakan adalah data yang digunakan untuk membangun tree jalan. Jadi data yang diinputkan telah menunjukkan hubungan antara data pertama dan data kedua pada data selanjutnya. Berikut struktur pada data pergerakan ditunjukan pada tabel 3.4 :

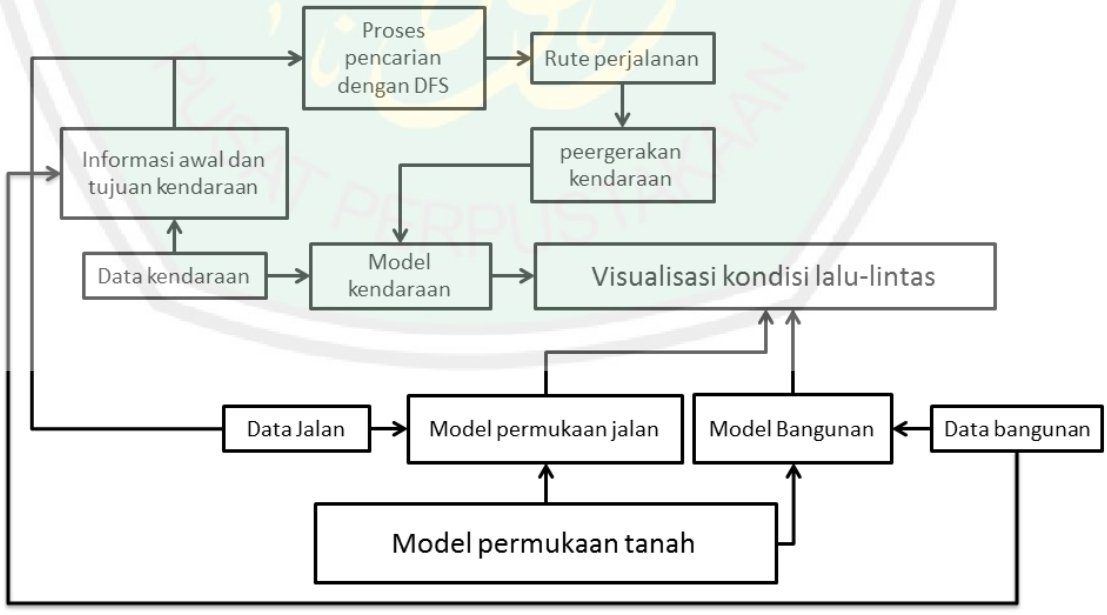
Tabel 3.4 Tabel Arah Gerak

No	Nama	Tipe
1	Nilai_satu	int
2	Nilai_dua	int
3	Nilai_tiga	int

Data yang terdapat pada tabel 3.4 diperoleh dari *filed* index pada tabel jalan dimana nilai_satu dan nilai_dua adalah menunjukkan node yang saling berhubungan dan nilai_tiga adalah satatus.

3.3 Desain Sistem dan Implementasi Algoritma

Pada bagian ini menjelaskan tentang desain system atau rancangan aplikasi yang akan peneliti buat dalam penelitian ini. Adapun diagram perancangan aplikasi yang akan peneliti lakukan ditunjukan pada Gambar 3.2



Gambar 3.2 Flowchart sistem secara garis besar

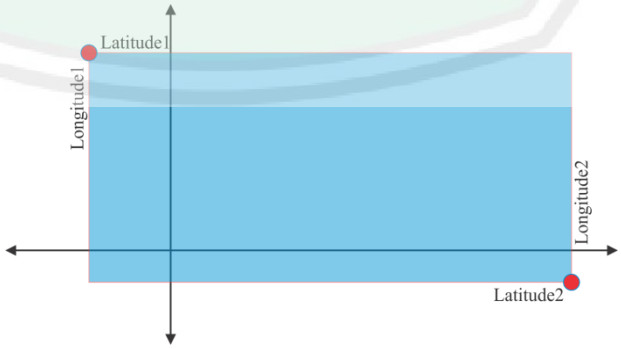
Untuk mendapat visualisasi pergerakan kendaraan maka dibutuhkan semua proses seperti pada Gambar 3.2. berikut penjelasan masing masing subrutin :

3.3.1 Pemodelan Permukaan Tanah

Pada bagaian pemodelan permukaan tanah ini adalah menggunakan data elevasi permukaan bumi SRTM (*Shuttle Radar Topography Mission*) 90 meter dpl (Diatas Permukaan Laut). Dan dibutuhkan input data citra daerah yang akan divisualisasikan. Inputan berupa lokasi ujung atas dan ujung bawah pada lokasi yang akan divisualisasikan, hal ini digunakan untuk menampilkan lokasi yang di *select* saja berdasarkan inputan batasa wilayah. Berikut data batas_wilayah seperti pada tabel 3.5 dan ilustrasi penggunaan batas_wilayah ditunjukan pada Gambar 3.3

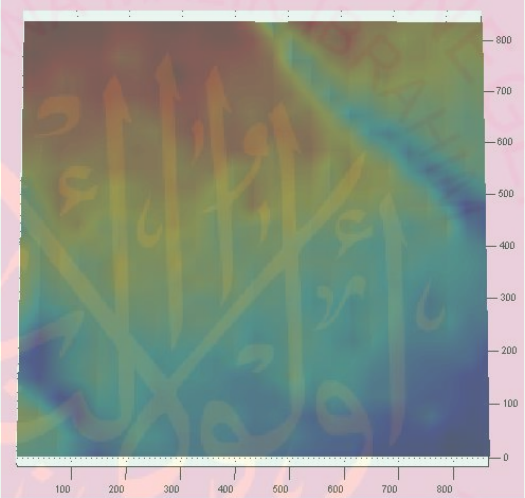
Tabel 3.5 Batas_Wilayah

NO	Longitude	Latitude	Keterangan
1	-7.946280	112.603196	Batas awal
2	-7.966041	112.623230	Batas akhir



Gambar 3.3 Titik merah ilustrasi batas_wilayah

Pada Gambar 3.3 dimana titik merah merupakan titik koordinat yang terdapat pada tabel 3.5 *latitude1-longitude1* satu sebagai batas awal dan *latitude2-longitude2* sebagai batas akhir. Dan warna biru yang ditunjukan pada Gambar 3.3 adalah lokasi permukaan tanah yang akan divisualkan. Berikut lokasi visualisasi permukaan tanah berdasarkan input koordinat seperti pada tabel 3.5 ditunjukan pada Gambar 3.4



Gambar 3.4 Visualisasi permukaan tanah

3.3.2 Pemodelan Jalan

Model permukaan jalan haruslah sesuai dengan kondisi yang sebenarnya. Inputan yang digunakan dalam pemodelan permukaan jalan ini adalah menggunakan data koordinat *latitude-longitude* seperti pada tabel 3.1. Penggambaran jalan dilakukan dengan menyambung koordinat *lat1-lang1* dan koordinat *lat2-lang2* kemudian menyambung *lat2-lang2* dan koordinat *lat3-lang3* sampai seterusnya dan berakhir pada jumlah titik yang terakhir. Berikut ilustrasi pemodelan jalan ditunjukan pada Gambar 3.5

Gambar 3.5 Ilustrasi menggambar jalan

Seperti terlihat pada Gambar 3.5 titik hitam adalah node yang akan dihubungkan dengan node yang lainnya, sedangkan persegi warna abu-abu adalah hasil yang diperoleh dari menyambungkan kedua node, dan warna abu-abu yang akan memberikan kondisi seperti jalan diatas permukaan tanah.

Tahap pertama yang dilakukan untuk menggambar jalan adalah membaca data *road*. Setelah itu mencari nilai *lat-lon* pada data SRTM yang sesuai dengan data `input_lat1 dan input_lon1`. Berikut *source code* mendapatkan nilai pada data SRTM ditunjukan pada Gambar 3.6

```
JALAN = xlsread('road.xls');
for baris ← 1:1:size(JALAN,1)
    input_lat1 = JALAN(baris,1);    input_lat2 = JALAN(baris,3);
    input_lon1 = JALAN(baris,2);    input_lon2 = JALAN(baris,4);
    lebar      = JALAN(baris,5);
error_lama = 100;
index_lat = 1;
for i ← 1:1:size(datalat,1)
    error_baru = abs(datalat(i,1)-input_lat1);
    if error_baru < error_lama
       error_lama = error_baru;
       index_lat = i;
    end;
end;
error_lama = 100;
index_lon = 1;
for j ← 1:1:size(datalon,1)
    error_baru = abs(datalon(j,1)-input_lon1);
    if error_baru < error_lama
       error_lama = error_baru;
       index_lon = j;
    end;
end;
index_lon1 = index_lon;
index_lat1 = index_lat;
tinggi1    = datagrid(index_lat1,index_lon1);
end
```

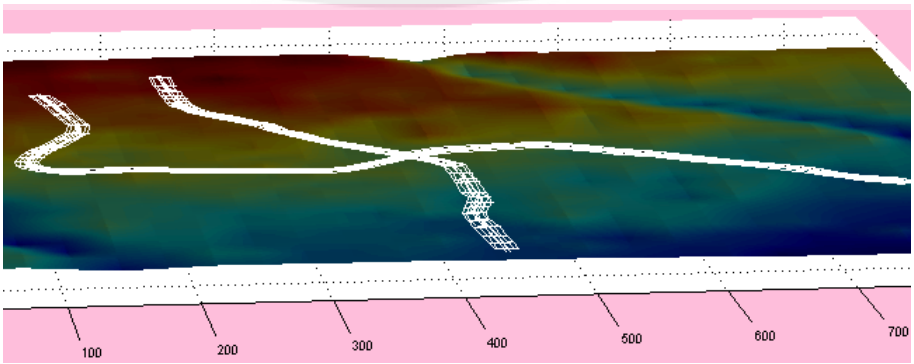
Gambar 3.6 *Source code* mencari index *latitude-longitude* pada data SRTM

Untuk proses mencari index *lat-lon* pada data SRTM berdasarkan data `input_lat2` dan data `input_lon2` sama dengan proses yang ditunjukan pada Gambar 3.6 hanya saja perlu mengubah inputan berdasarkan inisialisasi yang digunakan sebelumnya. Setelah mendapatkan nilai index *lat1-lon1* dan *lat2-lon2* proses selanjutnya adalah menggambar jalan. Berikut *source code* menggambar jalan ditunjukan pada Gambar 3.7

```
a1 = index_lon1;          a2 = index_lon2;
b1 = index_lat1;          b2 = index_lat2;
c1 = tinggi1+1;           c2 = tinggi2+1;
d  = lebar;               resolusi = resolusi;
tinggi1 = c1+2;           tinggi2 = c2+2;
lebar   = d/resolusi;
x = [a1 a1 a1; a2 a2 a2];
y = [b1-lebar/2 b1 b1+lebar/2; b2-lebar/2 b2 b2+lebar/2];
z = [tinggi1 tinggi1 tinggi1; tinggi2 tinggi2 tinggi2];
surface(x,y,z)

x = [a1-lebar/2 a1 a1+lebar/2; a2-lebar/2 a2 a2+lebar/2];
y = [b1 b1 b1; b2 b2 b2];
z = [tinggi1 tinggi1 tinggi1; tinggi2 tinggi2 tinggi2];
surface(x,y,z)
```

Gambar 3.7 *Source code* menggambar jalan

Pada proses menggambar jalan seperti Gambar 3.7 menggunakan fungsi *surface* yang tersedia pada *software* Matlab berdasarkan inputan yang telah diperoleh sebelumnya. Adapun hasil yang diperoleh dituntukan pada Gambar 3.8



Gambar 3.8 Visualisasi Jalan diatas permukaan tanah

3.3.3 Pemodelan Bangunan

Pada dasarnya proses pemodelan bangunan hampir sama dengan pemodelan jalan, hanya saja berbeda pada data yang digunakan dan pemodelan bangunan memerlukan ukuran fisik seperti panjang, lebar dan tinggi bangunan yang akan divisualkan.

Proses selanjutnya adalah mencari index *lat-lon* pada data SRTM berdasarkan `input_lat` dan `input_lon` pada data *building*. Berikut *source code* ditunjukan pada Gambar 3.9

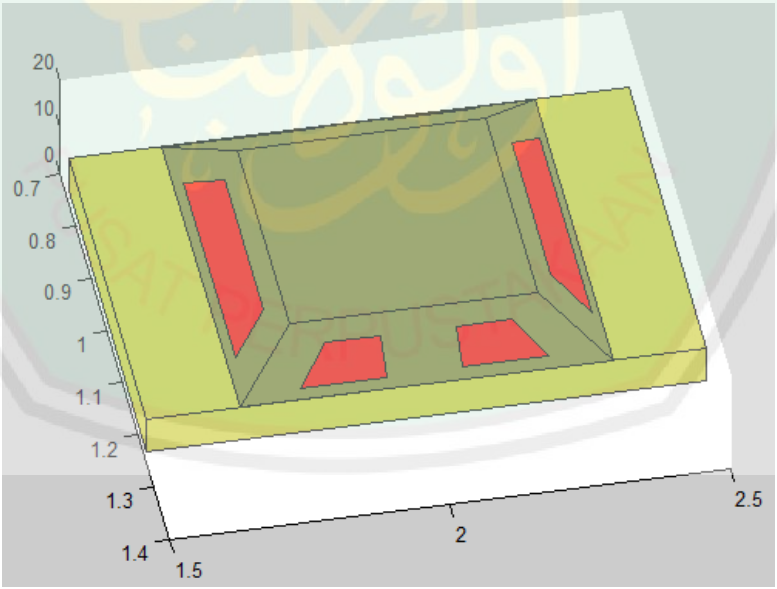
```
RUMAH = xlsread('building.xls');
for baris ← 1:1:size(RUMAH,1)
    input_lat = RUMAH(baris,1);
    input_lon = RUMAH(baris,2);
    panjang = RUMAH(baris,3);
    lebar = RUMAH(baris,4);
    tinggi = RUMAH(baris,5);
    sudut = RUMAH(baris,6);
error_lama = 100;
index_lat = 1;
for i ← 1:1:size(datalat,1)
    error_baru = abs(datalat(i,1)-input_lat);
     if error_baru < error_lama
        error_lama = error_baru;
        index_lat = i;
end; end;
error_lama = 100;
index_lon = 1;
for j ← 1:1:size(datalon,1)
    error_baru = abs(datalon(j,1)-input_lon);
     if error_baru < error_lama
        error_lama = error_baru;
        index_lon = j;
     end;
end;

rumah(index_lon,index_lat,datagrid(index_lat,index_lon),resolus
i,panjang,lebar,tinggi,sudut);
```

Gambar 3.9 *Source code* mencari index *latitude-longitude* pada data SRTM

Setelah memperoleh index *lat-lon,* untuk menggambar rumah terdapat pada fungsi rumah yang ditunjukan pada Gambar 3.9 paling bawah. Dimana parameter didalamnya telah diperoleh dari proses sebelumnya.

3.3.4 Pemodelan Kendaraan

Pada bagian pemodelan kendaraan ini peneliti tidak terlalu menitik berakkan bentuk mobil yang akan digunakan, karena peneliti hanya menggunakan objek berbentuk sederhana untuk visualisasi kendaraan. Karenan nantinya kendaraan akan digerakkan pada permukaan jalan, akan sangat lambat jika bentuk kendaraan terlalu detail karena akan menyimpan space yang banyak. Berikut bentuk visualisasi mobil yang peneliti gunakan ditunjukan pada Gambar 3.10



Gambar 3.10 Visualisasi Mobil

Gambar 3.10 diperoleh dari fungsi mobil yang telah peneliti definisikan sebelumnya. Namun pada visualisasi pergerakan nantinya peneliti akan

menggunakan objek berbentuk kubus. Berikut *source code* visualisasi kendaraan dalam bentuk kubus yang akan peneliti gunakan ditunjukan pada Gambar 3.11

```
function mobil(a,b,c,d,panjang,lebar,tinggi,sudut)
resolusi = d;
ld = lebar/resolusi; pd = panjang/resolusi; td = tinggi;
tinggi_atap = 5;
hold on;
  x1 = [(-ld/2*cosd(sudut))-(-pd/2*sind(sudut)) (ld/2*cosd(sudut))-(-
       pd/2*sind(sudut)) (ld/2*cosd(sudut))-(pd/2*sind(sudut)) (-
       ld/2*cosd(sudut))-(pd/2*sind(sudut))];
  y1 = [(-ld/2*sind(sudut))+(-pd/2*cosd(sudut)) (ld/2*sind(sudut))+(-
       pd/2*cosd(sudut)) (ld/2*sind(sudut))+(pd/2*cosd(sudut)) (-
       ld/2*sind(sudut))+(pd/2*cosd(sudut))];
  z1 = [td td td td];
  fill3(x1+a,y1+b,z1+c,[1 0.18 0],'EdgeColor',[1 0.18 0]);
  x1 = [(-ld/2*cosd(sudut))-(-pd/2*sind(sudut)) (ld/2*cosd(sudut))-(-
       pd/2*sind(sudut)) (ld/2*cosd(sudut))-(-pd/2*sind(sudut)) (-
       ld/2*cosd(sudut))-(-pd/2*sind(sudut))];
  y1 = [(-ld/2*sind(sudut))+(-pd/2*cosd(sudut)) (ld/2*sind(sudut))+(-
       pd/2*cosd(sudut)) (ld/2*sind(sudut))+(-pd/2*cosd(sudut)) (-
       ld/2*sind(sudut))+(-pd/2*cosd(sudut))];
  z1 = [0 0 td td];
  fill3(x1+a,y1+b,z1+c,[1 0.18 0],'EdgeColor',[1 0.18 0]);
  x1 = [(-ld/2*cosd(sudut))-(pd/2*sind(sudut)) (ld/2*cosd(sudut))-
       (pd/2*sind(sudut)) (ld/2*cosd(sudut))-(pd/2*sind(sudut)) (-
       ld/2*cosd(sudut))-(pd/2*sind(sudut))];
  y1 = [(-ld/2*sind(sudut))+(pd/2*cosd(sudut))
       (ld/2*sind(sudut))+(pd/2*cosd(sudut))
       (ld/2*sind(sudut))+(pd/2*cosd(sudut)) (-
       ld/2*sind(sudut))+(pd/2*cosd(sudut))];
  z1 = [0 0 td td];
  fill3(x1+a,y1+b,z1+c,[1 0.18 0],'EdgeColor',[1 0.18 0]);
  x1 = [(-ld/2*cosd(sudut))-(-pd/2*sind(sudut)) (-ld/2*cosd(sudut))-
       (pd/2*sind(sudut)) (-ld/2*cosd(sudut))-(pd/2*sind(sudut)) (-
       ld/2*cosd(sudut))-(-pd/2*sind(sudut))];
  y1 = [(-ld/2*sind(sudut))+(-pd/2*cosd(sudut)) (-
       ld/2*sind(sudut))+(pd/2*cosd(sudut)) (-
       ld/2*sind(sudut))+(pd/2*cosd(sudut)) (-ld/2*sind(sudut))+(-
       pd/2*cosd(sudut))];
  z1 = [0 0 td td];
  fill3(x1+a,y1+b,z1+c,[1 0.18 0],'EdgeColor',[1 0.18 0]);
  x1 = [(ld/2*cosd(sudut))-(-pd/2*sind(sudut)) (ld/2*cosd(sudut))-
       (pd/2*sind(sudut)) (ld/2*cosd(sudut))-(pd/2*sind(sudut))
       (ld/2*cosd(sudut))-(-pd/2*sind(sudut))];
  y1 = [(ld/2*sind(sudut))+(-pd/2*cosd(sudut))
       (ld/2*sind(sudut))+(pd/2*cosd(sudut))
       (ld/2*sind(sudut))+(pd/2*cosd(sudut)) (ld/2*sind(sudut))+(-
       pd/2*cosd(sudut))];
  z1 = [0 0 td td];
  fill3(x1+a,y1+b,z1+c,[1 0.18 0],'EdgeColor',[1 0.18 0]);
```

Gambar 3.11 *Function* mobil

3.3.5 Pencarian Menggunakan Depth First Search

Pada proses pencarian rute perjalanan kendaraan pertama yang diperlukan adalah data informasi lokasi awal dan lokasi tujuan seperti terdapat pada tabel 3.3. Selain data kendaraan, dibutuhkan juga data jalan sebagai lokasi kendaraan melakukan pergerakan.

Proses pencarian jalur pergerakan kendaraan menggunakan metode Depth First Search (DFS). Metode DFS ini merupakan algoritma pencarian simpul dalam garf secara transversal yang dimulai dari simpul akar yang mengecek simpul anaknya yang pertama, setelah itu, algoritma mengecek simpul anak dari simpul anak yang pertama tersebut, hingga mencapai simpul daun atau simpul tujuan. Berikut gambar ilustrasi jalan yang dibangun dari beberapa node ditunjukan pada Gambar 3.12

Gambar 3.12 Ilustrasi jalan yang terbentuk dari susunan node

Dapat diamati pada Gambar 3.12 terdapat ilustrasi penggambaran jalan yang terdiri dari kumpulan node node yang saling terhubung. Juga terdapat arah panah warna biru yang menunjukan jalur atau arah pergerakan kendaraan. Dibangian tengah terdapat lingkaran warna merah dan kuning. Lingkaran tersebut merupakan area dimana terjadi percabangan atau pada istilah yang sering kita gunakan dalam dunia transportasi adalah simpangan. pada setiap jalan di area simpangan memiliki node warna merah digunakan sebagai asal cabang dan node warna kuning digunakan sebagai tujuan cabang. Setiap warna merah yang berada pada lingkaran simpangan memiliki tujuan cabangnya masing masing. Dan pada ujung setiap jalan terdapat warna node hijau yang berarti adalah bahwa jalan yang pada kondisi sebenarnya masih panjang, namun peneliti batasi sampai pada node warna hijau.

Untuk membuat kondisi jalan seperti pada Gambar 3.12 maka dibutuhkan data *arah_gerak* untuk mengatur pergerakan kendaraan secara dinamis. Adapun data yang terdapat pada data *arah_gerak* berasal dari data jalan seperti terlihat pada tabel 3.1 pada bagian *index*. Berikut sebagian Data *arah_gerak* yang diperoleh berdasarkan pada Gambar 3.12 ditunjukan pada tabel 3.6

Tabel 3.6 Data *arah_gerak*

No	Nilai_satu	Nilai_dua	Nilai_tiga
1	1	2	0
2	2	3	0
3	3	4	0
4	4	5	0
5	5	6	0
6	6	7	0
7	7	8	0
8	8	9	0
9	9	10	0
10	10	11	0
11	11	12	0
12	12	13	0
13	13	14	0
14	14	15	0
15	15	16	0
16	16	17	0
17	17	18	0
18	18	19	0
19	19	20	0
20	20	21	0

Karena pada proses pencarian jalur pergerakan kendaraan melibatkan data bangunan, data jalan, data kendaraan dan data *arah_gerak*, maka proses pertama yang dilakukan adalam membaca masing masing data. Berikut *sourcecode* membaca semua data ditunjukan pada Gambar 3.13 :

```
JALAN = xlsread('road.xls');
BANGUNAN = xlsread('building.xls');
MOBIL = xlsread('Cars.xls');
PATH = xlsread('arah_gerak.xlsx');
for h ← 1 : size(MOBIL,1)
    INDEX_mobil_ke = MOBIL(h,5)
    nilai_ASAL_mobil = MOBIL(h,6);
    nilai_TUJUAN_mobil = MOBIL(h,7);
    LOKASI_AT = BANGUNAN(h,8);
    lat1_asal = BANGUNAN(nilai_ASAL_mobil,1);
    lon1_asal = BANGUNAN(nilai_ASAL_mobil,2);
    lat2_tujuan = BANGUNAN(nilai_TUJUAN_mobil,1);
    lon2_tujuan = BANGUNAN(nilai_TUJUAN_mobil,2);
end
```

Gambar 3.13 *Sourcecode* membaca data *all*

Berdasarkan data MOBIL seperti pada tabel 3.4 bahwa asal dan tujuan kendaraan berasal dari bangunan seperti dijelaskan pada Gambar 3.13. maka proses selanjutnya adalah mencari titik atau node terdekat pada data JALAN berdasarkan nilai BANGUNAN. Peneliti menggunakan nilai asal dan tujuan terletak pada bangunan karena menurut *Khisty and Lall, 2003* menyatakan bahwa salahsatu elemen utama transportasi adalah terminal, merupakan titik-titik dimana perjalanan dimulai atau berakhir. Contoh : garasi mobil, lapangan parkir, gudang bongkar muat dan bandara uadara. Sedangkan jalan merupakan sarana penghubung atau *link*. Berikut *sourcecode* mencari nilai terdekat dengan menggunakan metode *euclidean distance* ditunjukan pada Gambar 3.14

```
error_A = 1000;
for X ← 1:size(JALAN,1)
    lat_jalan = JALAN(X,1);
    lon_jalan = JALAN(X,2);
     error_A1 = sqrt(((lat1_asal-lat_jalan)^2)+((lon1_asal-
     lon_jalan)^2));
     if error_A1 <error_A
        error_A = error_A1;
        ASAL_lat = lat_jalan;
        ASAL_lon = lon_jalan;
     End
error_B = 1000;
for Y ← 1:size(JALAN,1)
   lat_jalan = JALAN(Y,1);
   lon_jalan = JALAN(Y,2);
   error_B1 = sqrt(((lat2_tujuan-lat_jalan)^2)+((lon2_tujuan-
   lon_jalan)^2));
   if error_B1 < error_B
      error_B = error_B1;
      TUJUAN_lat = lat_jalan;
      TUJUAN_lon = lon_jalan;
   End
end
end
    ASAL_lat; ASAL_lon;
    ASAL_jalan = [ASAL_lat, ASAL_lon];
    TUJUAN_jalan = [TUJUAN_lat, TUJUAN_lon];
```

Gambar 3.14 *Sourcecode* mencari nilai terdekat menggunakan *euclidean distance*

Dari proses yang ditunjukan pada Gambar 3.14 akan diperoleh nilai jalan yang terdekat dari bangunan berupa *latitude-longitude*. Untuk memudahkan proses pencarian jalur, data *latitude-longitude* pada data jalan dialihkan pada index pada data jalan. Berikut pseudocode merubah atau menconvert data yang akan dibaca pada proses selanjutnya dari nilai *latitude-longitude* ke nilai unik yang dimiliki oleh index pada data jalan ditunjukan pada Gambar 3.15

```
for g = 1:size(JALAN,1)
    if ASAL_lat == JALAN(g,1) && ASAL_lon == JALAN(g,2)
       index_ASAL_jalan = JALAN(g,6);
       index_ASAL_jalan;
    end
    if TUJUAN_lat == JALAN(g,1) && TUJUAN_lon == JALAN(g,2)
       index_TUJUAN_jalan = JALAN(g,6);
       index_TUJUAN_jalan;
    end
end
    index_ASAL_jalan;
    index_TUJUAN_jalan;

    mulai = index_ASAL_jalan
    tujuan = index_TUJUAN_jalan
```

Gambar 3.15 *Sourcecode* convert *latitude-longitude* ke nilai index

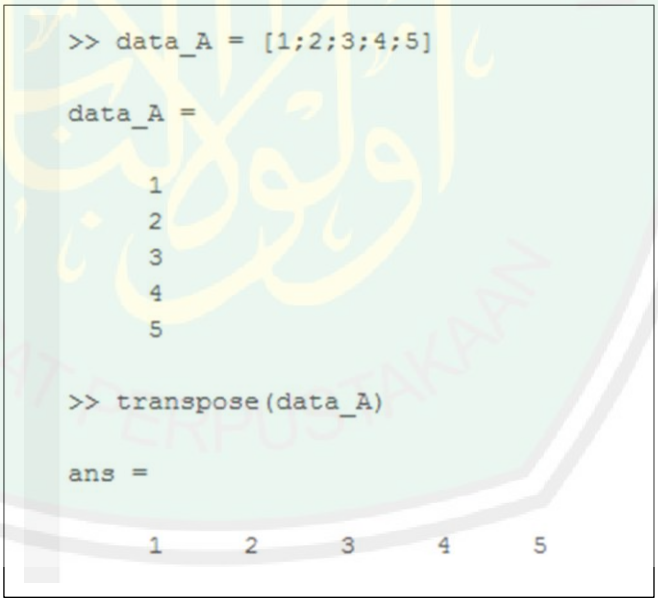
Setelah diperoleh nilai asal / `mulai` dan `tujuan` dalam bentuk yang unik pada index, maka proses selajutnya adalah pencarian menggunakan Metode Depth First Search (DFS)

```
for f ← 1:size(PATH,1)
    if PATH(f,1) == mulai
       index_mulai = f;
       break
    end
end
index_mulai;
for e ← 1:index_mulai-1
    data_dipindah(e,1) = PATH(e,1);
    data_dipindah(e,2) = PATH(e,2);
    data_dipindah(e,3) = PATH(e,3);
end
j = 1;
for d ← index_mulai:size(PATH,1)
    path_baru(j,1) = PATH(d,1);
    path_baru(j,2) = PATH(d,2);
    path_baru(j,3) = PATH(d,3);
    j = j + 1;
end
j = size(path_baru,1)+1;
for c ← 1:size(data_dipindah,1)
    path_baru(j,1) = data_dipindah(c,1);
    path_baru(j,2) = data_dipindah(c,2);
    path_baru(j,3) = data_dipindah(c,3);
    j = j + 1;
end
jumlah_cabang = 0;
j = 1;
for k ← 1:size(path_baru,1)
    if k < size(path_baru,1)
       if path_baru(k,1) == path_baru(k+1,1)
          jumlah_cabang = jumlah_cabang + 1;
          alamat_cabang(j) = k;
          j = j + 1;
       end
    end
end
jumlah_cabang;
path_baru;
baris1 = path_baru(:,1);
baris2 = path_baru(:,2);
baris3 = path_baru(:,3);
baris3A = transpose(baris3);
baris1A = transpose(baris1);
baris2A = transpose(baris2);
sizenya = size(baris1A);
DG1 = sparse(baris1A,baris2A,true,165,165);
order = graphtraverse(DG1,mulai);
ordernya = size(order);
```

Gambar 3.16 *Sourcecode swaping* data berdasarkan nilai mulai

Proses yang ditunjukan pada Gambar 3.16 menunjukan proses pemindahan data pada nilai yang berada diatas nilai asal kendaraan diletakkan pada bagian ujung akhir data dan hasil disimpan pada variable *path_baru.* Misalkan dari data

1,2,3,4,5,6,7,8,9,10,11,12 lokasi asal mobil terdapat pada angka 7, maka otomatis angka diatas nilai asal akan dipindah dibawah data terakhir dan data menjadi 7,8,9,10,11,12,1,2,3,4,5,6 dan data terakhirlah yang akan disimpan pada variable *path_baru*. Karena pada *path_baru* menyimpan seluruh data dari tabel *arah_gerak* maka data perlu dipecah sendiri sendiri untuk mempermudah akses data. Setelah memperoleh nilai *path_baru* dengan masing masing variabelnya maka dilakukan proses *transpose* dengan menggunakan fungsi yang ada pada matlab. Data perlu ditranspose karena peneliti menggunakan fungsi *sparse*. Berikut ilustrasi dari fungsi *transpose* yang akan digunakan pada *sparse* ditunjukan pada Gambar 3.17 berikut :

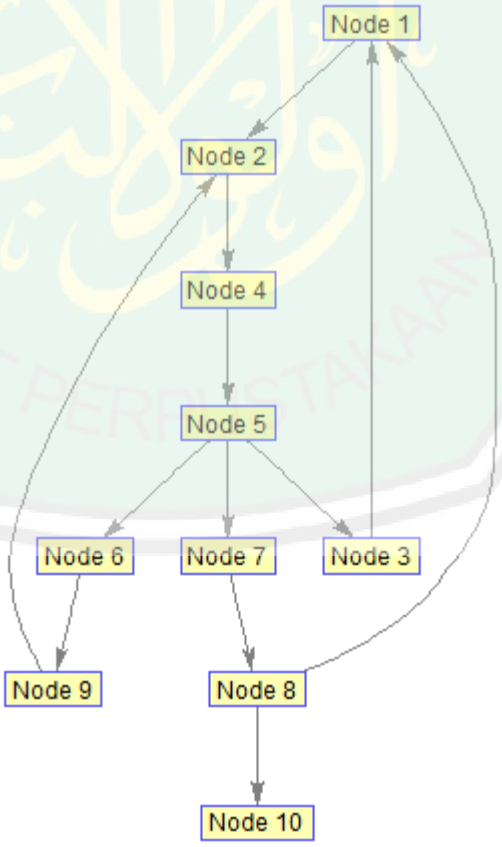


Gambar 3.17 Contoh fungsi transpose menggunakan Matlab

Hasil pertama yang diperoleh dari proses pada Gambar 3.17 disimpan pada variable *order* dengan menggunakan fungsi `graphtraverse` dimana data pertama dimulai dari nilai awal atau lokasi awal kendaraan. adapaun DG1 dengan

menggunakan fungsi `sparse` digunakan untuk membaca node_satu dan node_dua dengan jumlah maksimal dari node_satu dan node_dua, dan true digunakan untuk mengeluarkan nilai yang diperoleh fungi `sparse` pada DG1. Misalkan nilai node_satu adalah [1 2 3 4 5 5 5 6 7 8 8 9] dan nilai node_dua adalah [2 4 1 5 3 6 7 9 8 1 10 2] dimana nilai pada nilai_satu yang lebih dari 1 akan dianggap sebagai cabang dan tujuannya berdasarkan urutan nilai pada node_dua dan nilai maksimal dari kedua data node_satu dan node_dua adalah 10.

Adapun hasil yang diperoleh DG1 jika dikelauarkan dalam bentuk tree dengan menggunakan fungsi `biograph` ditunjukan pada Gambar 3.18

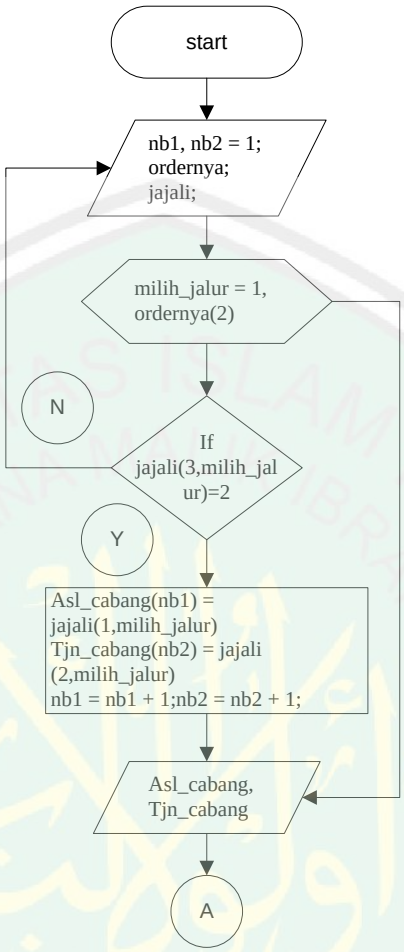


Gambar 3.18 Contoh tree yang diperoleh

Setelah memperoleh nilai DG1 selanjutnya adalah proses penelusuran node yang dimulai dari asal kendaraan atau nilai mulai dengan menggunakan fungsi `graphtraverse` seperti yang ditunjukan pada Gambar 3.17 bagian bawah. Masukkan nilai awal kendaraan misalkan lokasi asal pada nilai 4, maka order akan menghasilkan nilai *Order* : 4 5 3 1 2 6 9 7 8 10.

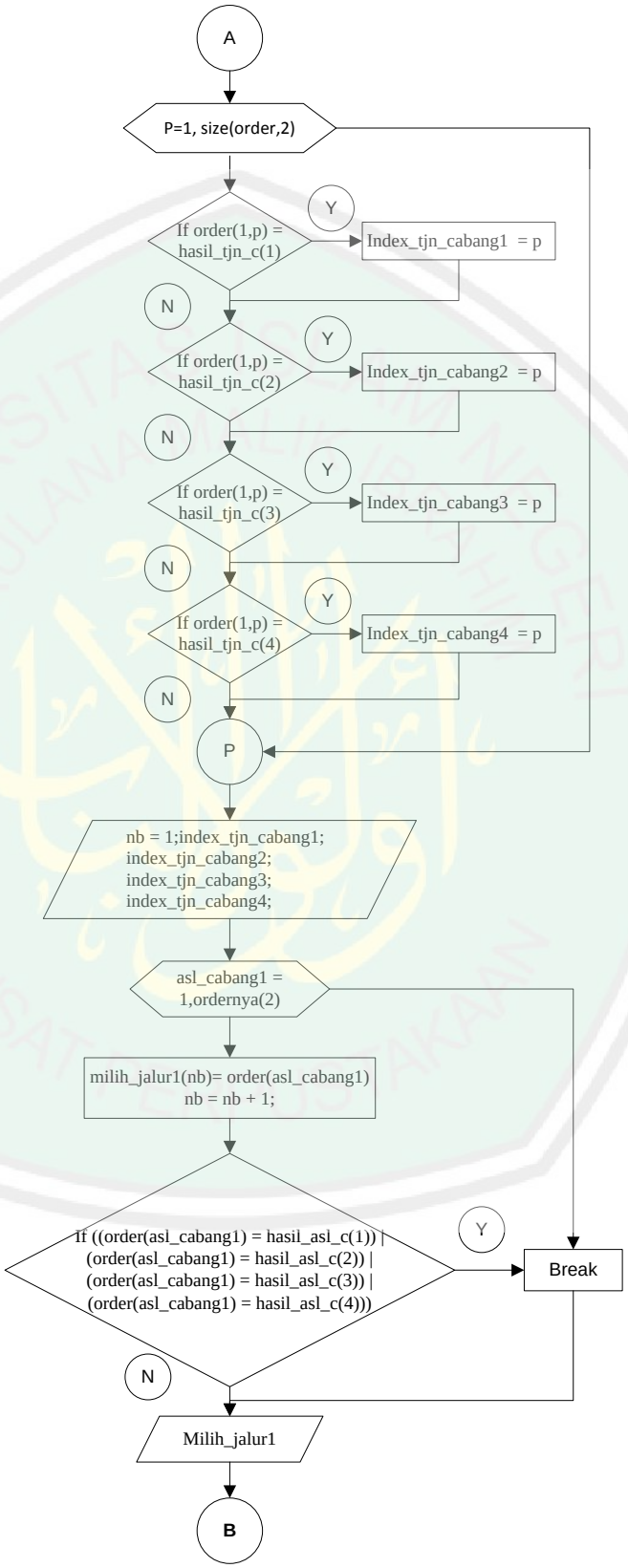
Jika diperhatikan dari output yang diperoleh, order akan menghasilkan data yang dimualai dari nilai 4 dan data selanjutnya order akan membaca data berdasarkan tree yang telah diperoleh. Jika menemukan cabang, fungsi *graphtraverse* akan memilih jalur berdasarkan urutan index atau dalam hal ini tergantung data mana yang diprioritaskan. *Graphtraverse* tidak akan mengunjungi dua kali node yang telah dilewati jika dilihat dari output yang diperoleh order.

Namun yang diperoleh order adalah semua data node, bukan jalur yang dipilih karena memang belum ada inputan tujuan kendaraan. Proses selanjutnya adalah memilih cabang hingga menemukan tujuan cabang dan sampai rute diperoleh ditunjukan oleh *flowchart* pada Gambar 3.19



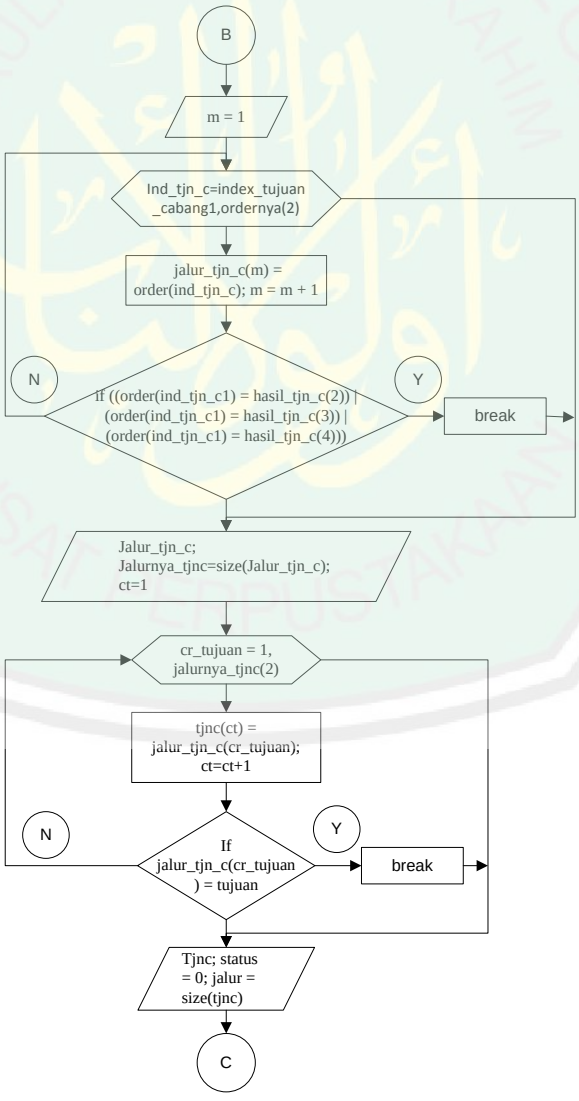
Gambar 3.19 F*lowchart* mendeteksi cabang (asl_cabang dan tjn_cabang)

Dari proses seperti Gambar 3.19 telah diperoleh cabang dimana setiap cabang memiliki dua status jika tidak asal cabang maka akan menjadi tujuan cabang, status ini telah ditentukan sebelumnya pada data *arah_gerak* dimana pada *nilai_satu* yang memiliki nilai lebih dari satu akan menjadi *asl_cabang* dan pada *nilai_dua* yang memiliki index sama dengan *node_satu* akan menjadia *tjn_cabang*. Proses pada Gambar 3.19 akan diteruskan pada proses berikut ditunjukan *flowchart* pada Gambar 3.20

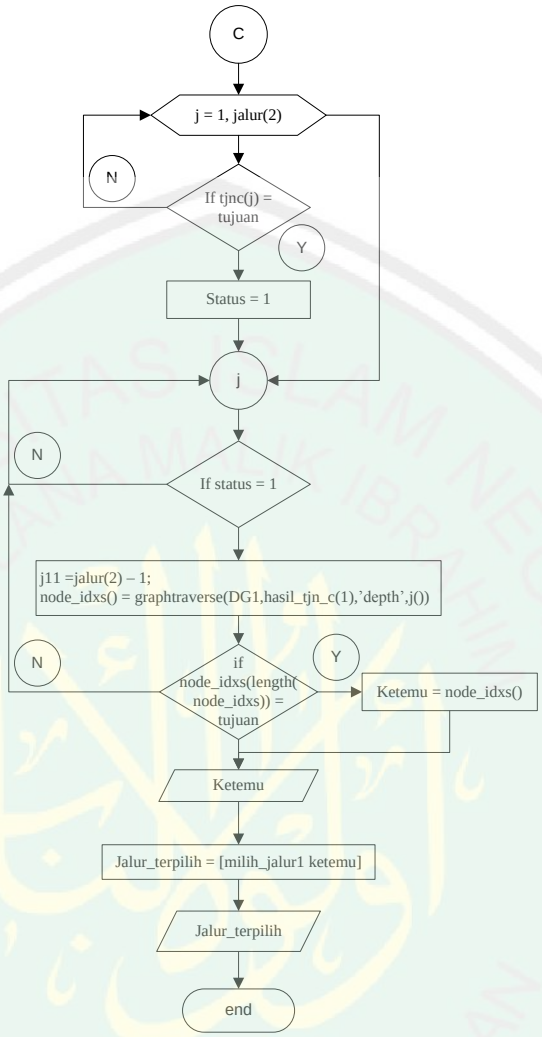


Gambar 3.20 *Flowchart* menemukan cabang pertama

Pada Gambar 3.20 telah diperoleh setengah jalur sampai pada cabang pertama yang temukan yaitu berisi nilai asal kendaraan sampai cabang pertama yang disimpan pada variable *milih_jalur1* hal ini tentu berdasarkan data *order* yang telah diperoleh, karena setiap kendaraan akan memiliki susunan order yang berbeda beda. Setelah menemukan cabang pada status *asl_cabang* maka system akan memilih beberapa cabang yang berstatus *tjn_cabang* yang tersedia. Adapun proses dari pemeilihan tjn_cabang ditunjukan *flowchart* pada Gambar 3.21



(a)



(b)

Gambar 3.21 (a) dan (b) memperoleh jalur asal sampai tujuan

Perhatikan pada Gambar 3.21 (b) bagian akhir telah diperoleh jalur dari asal sampai pada tujuan yang disimpan pada variable *jalur_terpilih.* Pada variable *jalur_terpilih* berisi beberapa node yang berisi dari asal - asl_cbg - tjn_cbg – tujuan. Pada dasarnya DFS mampu mengunjungi semua node yang terhubung, namun system hanya mengeluarkan node yang terhubung sampai pada tujuan yang telah ditentukan. Pada Gambar 3.21 bagian (a) dimana jumlah perulangannya sebanyak jumlah cabang yang diperoleh dari menemukan cabang.

Berikut *sourcecode* proses DFS pada pencarian rute perjalanan ditunjukan pada Gambar 3.22

```
for h = 1
    INDEX_mobil_ke = MOBIL(h,5)
    nilai_ASAL_mobil = MOBIL(h,6);
    nilai_TUJUAN_mobil = MOBIL(h,7);
    LOKASI_AT = BANGUNAN(h,8);
    lat1_asal = BANGUNAN(nilai_ASAL_mobil,1);
    lon1_asal = BANGUNAN(nilai_ASAL_mobil,2);
    lat2_tujuan = BANGUNAN(nilai_TUJUAN_mobil,1);
    lon2_tujuan = BANGUNAN(nilai_TUJUAN_mobil,2);
    error_A = 1000;
    for X = 1:size(JALAN,1)
        lat_jalan = JALAN(X,1);
        lon_jalan = JALAN(X,2);
        error_A1 = sqrt(((lat1_asal-lat_jalan)^2)+((lon1_asal-
        lon_jalan)^2));
        if error_A1 <error_A
            error_A = error_A1;
            ASAL_lat = lat_jalan;
            ASAL_lon = lon_jalan;
        end
        error_B = 1000;
        for Y = 1:size(JALAN,1)
            lat_jalan = JALAN(Y,1);
            lon_jalan = JALAN(Y,2);
        error_B1 = sqrt(((lat2_tujuan-
        lat_jalan)^2)+((lon2_tujuan-lon_jalan)^2));
            if error_B1 < error_B
                error_B = error_B1;
                TUJUAN_lat = lat_jalan;
                TUJUAN_lon = lon_jalan;
            end
        end
    end
ASAL_lat; ASAL_lon;
ASAL_jalan = [ASAL_lat, ASAL_lon];
TUJUAN_jalan = [TUJUAN_lat, TUJUAN_lon];
    for g = 1:size(JALAN,1)
        if ASAL_lat == JALAN(g,1) && ASAL_lon == JALAN(g,2)
            index_ASAL_jalan = JALAN(g,6);
            index_ASAL_jalan;
        end
        if TUJUAN_lat == JALAN(g,1) && TUJUAN_lon == JALAN(g,2)
            index_TUJUAN_jalan = JALAN(g,6);
            index_TUJUAN_jalan;
        end
    end
index_ASAL_jalan;
index_TUJUAN_jalan;
cari_rute_dari_ke = [index_ASAL_jalan, index_TUJUAN_jalan];
sdt = [index_ASAL_jalan:index_TUJUAN_jalan];
```

```
mulai = index_ASAL_jalan
    tujuan = index_TUJUAN_jalan
    for f = 1:size(PATH,1)
        if PATH(f,1) == mulai
            index_mulai = f;
            break
        end
    end
    index_mulai;
    for e = 1:index_mulai-1
        data_dipindah(e,1) = PATH(e,1);
        data_dipindah(e,2) = PATH(e,2);
        data_dipindah(e,3) = PATH(e,3);
    end
    j = 1;
    for d = index_mulai:size(PATH,1)
        path_baru(j,1) = PATH(d,1);
        path_baru(j,2) = PATH(d,2);
        path_baru(j,3) = PATH(d,3);
        j = j + 1;
    end
    j = size(path_baru,1)+1;
    for c = 1:size(data_dipindah,1)
        path_baru(j,1) = data_dipindah(c,1);
        path_baru(j,2) = data_dipindah(c,2);
        path_baru(j,3) = data_dipindah(c,3);
        j = j + 1;
    end
    path_baru;
    jumlah_cabang = 0;
    j = 1;
    for k = 1:size(path_baru,1)
        if k < size(path_baru,1)
            if path_baru(k,1) == path_baru(k+1,1)
                jumlah_cabang = jumlah_cabang + 1;
                alamat_cabang(j) = k;
                j = j + 1;
            end
        end
    end
    jumlah_cabang;
    path_baru;
    baris1 = path_baru(:,1);
    baris2 = path_baru(:,2);
    baris3 = path_baru(:,3);
    baris3A = transpose(baris3);
    baris1A = transpose(baris1);
    baris2A = transpose(baris2);
    sizenya = size(baris1A);
    sizenya1 = sizenya(2)-7;
    jajali = [baris1A;baris2A;baris3A];
    jajal = size(jajali);
    DG1 = sparse(baris1A,baris2A,true,165,165);
    order = graphtraverse(DG1,mulai)
    ordernya = size(order)
```

```
nb1 = 1;
    nb2 = 1;
    nb3 = 1;
    for milih_jalur = 1:ordernya(2)
        if jajali(3,milih_jalur) == 2
            asl_cabang(nb1) = jajali(1,milih_jalur);
            tjn_cabang(nb2) = jajali(2,milih_jalur);
            nb1 = nb1 + 1;
            nb2 = nb2 + 1;
        end
    end
    asl_cabang
    [mix_asl_c,as] = unique(asl_cabang, 'first');
    hasil_asl_c = asl_cabang(sort(as))
    tjn_cabang
    [mix_tjn_c,ts] = unique(tjn_cabang, 'first');
    hasil_tjn_c = tjn_cabang(sort(ts))
    for p = 1:size(order,2)
        if order(1,p) == hasil_tjn_c(1)
            indexnya_tjn_cabang1 = p;
        end
        if order(1,p) == hasil_tjn_c(2)
            indexnya_tjn_cabang2 = p;
        end
        if order(1,p) == hasil_tjn_c(3)
            indexnya_tjn_cabang3 = p;
        end
        if order(1,p) == hasil_tjn_c(4)
            indexnya_tjn_cabang4 = p;
        end
    end
    indexnya_tjn_cabang1; indexnya_tjn_cabang2;
    indexnya_tjn_cabang3; indexnya_tjn_cabang4;
    index_tjn_c = [indexnya_tjn_cabang1 indexnya_tjn_cabang2
    indexnya_tjn_cabang3 indexnya_tjn_cabang4];
    nb = 1;
    for asl_cabang1 = 1:ordernya(2)
        milih_jalur1(nb) = order(asl_cabang1);
        nb = nb + 1;
        if ((order(asl_cabang1) == hasil_asl_c(1)) |
            (order(asl_cabang1) == hasil_asl_c(2)) |
            (order(asl_cabang1) == hasil_asl_c(3)) |
            (order(asl_cabang1) == hasil_asl_c(4)))
            break
        end
    end
    milih_jalur1;
    m1 = 1;
    for ind_tjn_c1 = index_tjn_c(1):ordernya(2)
        jalur_tjn_c1(m1) = order(ind_tjn_c1);
        m1 = m1 + 1;
        if ((order(ind_tjn_c1) == hasil_tjn_c(2)) |
            (order(ind_tjn_c1) == hasil_tjn_c(3)) |
            (order(ind_tjn_c1) == hasil_tjn_c(4)))
            break
        end
    end
```

```
jalur_tjn_c1;
    jalurnya_tjnc1 = size(jalur_tjn_c1);
    ct1 = 1;
    for cr_tujuan1 = 1:jalurnya_tjnc1(2)
        tjnc1(ct1) = jalur_tjn_c1(cr_tujuan1);
        ct1 = ct1 + 1;
        if jalur_tjn_c1(cr_tujuan1) == tujuan
            break
        end
    end
    tjnc1;
    status = 0;
    jalur1 = size(tjnc1);
    for j1 = 1:jalur1(2)
        if tjnc1(j1) == tujuan
            status = 1;
        end
    end
    if status == 1
        j11 = jalur1(2)-1;
        node_idxs1 =
        graphtraverse(DG1,hasil_tjn_c(1),'depth',j11);
        if node_idxs1(length(node_idxs1)) == tujuan
           ketemu = node_idxs1;
        end
    end
    m2 = 1;
    for ind_tjn_c2 = index_tjn_c(2):ordernya(2)
        jalur_tjn_c2(m2) = order(ind_tjn_c2);
        m2 = m2 + 1;
        if ((order(ind_tjn_c2) == hasil_tjn_c(1)) |
           (order(ind_tjn_c2) == hasil_tjn_c(3)) |
           (order(ind_tjn_c2) == hasil_tjn_c(4)))
            break
        end
    end
    jalur_tjn_c2;
    jalurnya_tjnc2 = size(jalur_tjn_c2);
    ct2 = 1;
    for cr_tujuan2 = 1:jalurnya_tjnc2(2)
        tjnc2(ct2) = jalur_tjn_c2(cr_tujuan2);
        ct2 = ct2 + 1;
        if jalur_tjn_c2(cr_tujuan2) == tujuan
            break
        end
    end
    tjnc2;
    status = 0;
    jalur2 = size(tjnc2);
    for j2 = 1:jalur2(2)
        if tjnc2(j2) == tujuan
            status = 1;
        end
    end
```

```
if status == 1
     j22 = jalur2(2)-1;
     node_idxs1 =
     graphtraverse(DG1,hasil_tjn_c(2),'depth',j22);
     if node_idxs1(length(node_idxs1)) == tujuan
         ketemu = node_idxs1;
     end
 end
 m3 = 1;
 for ind_tjn_c3 = index_tjn_c(3):ordernya(2)
     jalur_tjn_c3(m3) = order(ind_tjn_c3);
     m3 = m3 + 1;
     if ((order(ind_tjn_c3) == hasil_tjn_c(1)) |
         (order(ind_tjn_c3) == hasil_tjn_c(2)) |
         (order(ind_tjn_c3) == hasil_tjn_c(4)))
         break
     end
 end
 jalur_tjn_c3;
 jalurnya_tjnc3 = size(jalur_tjn_c3);
 ct3 = 1;
 for cr_tujuan3 = 1:jalurnya_tjnc3(2)
     tjnc3(ct3) = jalur_tjn_c3(cr_tujuan3);
     ct3 = ct3 + 1;
     if jalur_tjn_c3(cr_tujuan3) == tujuan
         break
     end
 end
 tjnc3;
 status = 0;
 jalur3 = size(tjnc3);
 for j3 = 1:jalur3(2)
     if tjnc3(j3) == tujuan
         status = 1;
     end
 end
 if status == 1
     j33 = jalur3(2)-1;
     node_idxs1 =
     graphtraverse(DG1,hasil_tjn_c(3),'depth',j33);
     if node_idxs1(length(node_idxs1)) == tujuan
         ketemu = node_idxs1;
     end
 end
 m4 = 1;
 for ind_tjn_c4 = index_tjn_c(4):ordernya(2)
     jalur_tjn_c4(m4) = order(ind_tjn_c4);
     m4 = m4 + 1;
     if ((order(ind_tjn_c4) == hasil_tjn_c(1)) |
         (order(ind_tjn_c4) == hasil_tjn_c(2)) |
         (order(ind_tjn_c4) == hasil_tjn_c(3)))
         break
     end
 end
```

```
    jalur_tjn_c4;
    jalurnya_tjnc4 = size(jalur_tjn_c4);
    ct4 = 1;
    for cr_tujuan4 = 1:jalurnya_tjnc4(2)
        tjnc4(ct4) = jalur_tjn_c4(cr_tujuan4);
        ct4 = ct4 + 1;
        if jalur_tjn_c4(cr_tujuan4) == tujuan
            break
        end
    end
    tjnc4;
    status = 0;
    jalur4 = size(tjnc4);
    for j4 = 1:jalur4(2)
        if tjnc4(j4) == tujuan
            status = 1;
        end
    end
    if status == 1
        j44 = jalur4(2)-1;
                node_idxs1 =
         graphtraverse(DG1,hasil_tjn_c(4),'depth',j44);
        if node_idxs1(length(node_idxs1)) == tujuan
            ketemu = node_idxs1;
        end
    end
    jalur_terpilih1 = [milih_jalur1 ketemu]
end
save jalur1A jalur_terpilih1
```

Gambar 3.22 *Sourcecode* proses DFS

3.3.6 Visualisasi Pergerakan Kendaraan

Setelah mendapatkan rute perjalanan masing masing kendaraan, mudah untuk melakukan visualisasi pergerakan kendaraan berdasarkan rute yang telah diperoleh. Proses pertama yang dilakukan adalah menyimpan jalur dalam sebuah variable contohnya pada mobil satu `jalur1 = jalur_terpilih1` kemudian merubah nilai pada *jalur1* kembali ke nilai *lat-lang* karena nantinya proses pergerakan kendaraan akan dilakukan diatas permukaan tanah yang menggunakan data koordinat *longitude-latitude.* Adapun proses untuk mengkonvert nilai *jalur1* ke *lat-lang* ditunjukan pada Gambar 3.23

```
sizenya1 = size(jalur1);
for konv_to_ltln1 = 1 : sizenya1(2)
    for id_jalannya1 = 1 : size(JALAN,1)
        if jalur1(konv_to_ltln1) == JALAN(id_jalannya1,6)
            konv_lat1 = JALAN(id_jalannya1,1);
            konv_lon1 = JALAN(id_jalannya1,2);
            lat_lon1(konv_to_ltln1,1) = [konv_lat1];
            lat_lon1(konv_to_ltln1,2) = [konv_lon1];
            size_lat_lon1 = size(lat_lon1);
        end
    end
end
```

Gambar 3.23 Convert nilai ke *lat-lang*

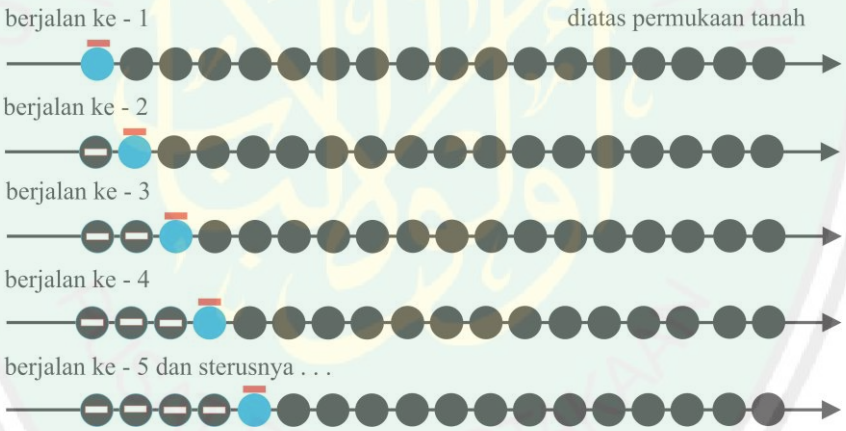
Karena jumlah mobil yang akan digerakkan lebih dari satu, untuk memudahkan mengetahui proses antar mobil satu dan yang lain maka menggunakan index kendaraan setiap kali proses pada masing masing kendaraan. Gambar 3.23 menunjukkan proses convert pada mobil ke-satu. Dengan proses tersebut, maka nilai pada *jalur1* berisi kumpulan koordinat *lat-lang.*

Kemudian untuk menggerakkan kendaraan pertama berdasarkan *jalur1* yang telah diperoleh adalah mencari nilai index pada data SRTM permukaan tanah. Adapun proses mencari index pada permukaan tanah ditunjukan pada Gambar 3.24

```
size_mobil1 = size(jalur1);
A = [size_mobil1(2)]
Perputaran_max = max(A)
for berjalan = 1 : perputaran_max
    if berjalan <= size_mobil1(2)
        lat1 = lat_lon1(berjalan,1);
        lon1 = lat_lon1(berjalan,2);
        error_lama1 = 100;
        index_lat1 = 1;
        for i1=1:1:size(datalat,1)
            error_baru1 = abs(datalat(i1,1)-lat1);
            if error_baru1 < error_lama1
                error_lama1 = error_baru1;
                index_lat1 = i1;
            end;
        end;
        error_lama1 = 100;
        index_lon1 = 1;
        for j1=1:1:size(datalon,1)
            error_baru1 = abs(datalon(j1,1)-lon1);
            if error_baru1 < error_lama1
                error_lama1 = error_baru1;
                index_lon1 = j1;
            end;
        end;
    end
    hold off
    mesh(datagrid);
    view(0,45);
    set(GUI.axes2,'Box','off');
    camlight;
    lighting phong;
    set(GUI.axes2,'projection','perspective');
    set(GUI.axes2,'Units','pixels');
    set(GUI.axes2,'DataAspectRatio',[1 1 resolusi]);
    hold on
    mobil(index_lon1,index_lat1,datagrid(index_lat1,index_lon1),
      resolusi,panjang,lebar,tinggi,sudut)
    jalan
    bangunan
    drawnow;
end
```

Gambar 3.24 Visualisasi pergerakan kendaraan

Jika diamati pada Gambar 3.24 pertamakali proses yang dilakukan adalah mengetahui ukuran atau jumlah jalur pada masing masing kendaraan, dalam contoh adalah mobil pertama, selanjutnya proses mencari nilai index *lat-lang* pada permukaan tanah dan hasil yang diperoleh disimpan pada masing masing variable *index_lat1* dan *index_lon1*. Kemudian memanggil fungsi mobil seperti yang ditunjukan pada Gambar 3.24 dan menginputkan parameter berdasarkan hasil yang telah diperoleh. Setelah itu peneliti mneggunakan fungsi dasar pada matlab yaitu *drawnow* dimana fungsi tersebut bersifat *draw-replace-next.* Adapun ilustrasi fungsi *drawnow* ditunjukan pada Gambar 3.25



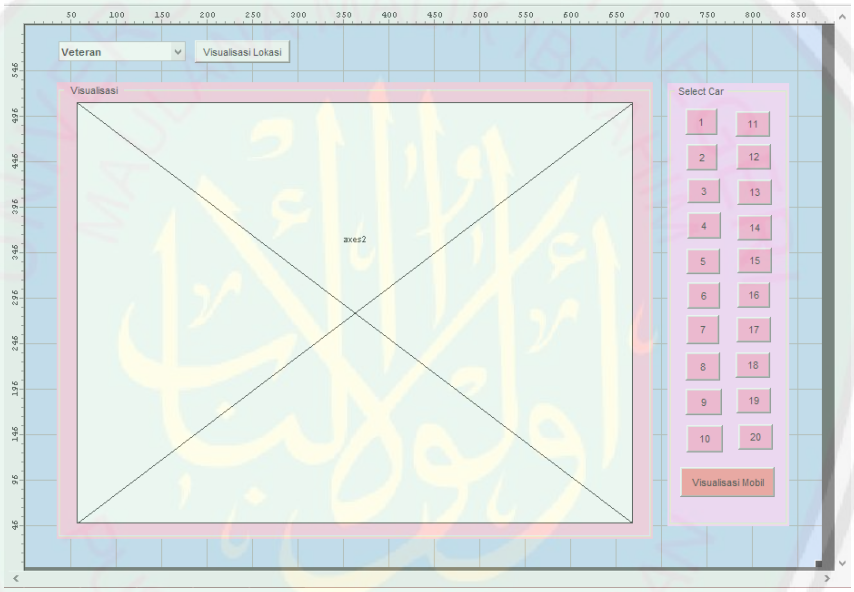
Gambar 3.25 Ilustrasi menggunakan fungsi *drawnow*

Perhatikan pada Gambar 3.25 dimana node warna hitam adalah jalur pergerakan kendaraan yang telah diperoleh namun berlum dilewati pada proses looping dan node warna biru adalah node yang telah dilewati kendaraan. Perhatikan juga persegi warna merah merupakan mobil yang selalu berpindah dari node 1 ke node selanjutnya sampai terakhir jalur yang diperoleh atau tujuan maka mobil akan berhenti secara otomatis.

3.4 Perancangan Interface

Untuk mempermudah pengguna, maka diperlukan adanya tampilan antarmuka (*interface*). Berikut ini tampilan rancangan antarmuka untuk proses visualisasi pergerakan kendaraan bermotor.

a. Interface utama



Gambar 3.26 Antarmuka Visualisasi

Pada halaman interface utama aplikasi visualisasi pergerakan kendaraan terdapat beberapa tombol antara lain sebagai berikut :

1. *Pop-up menu*

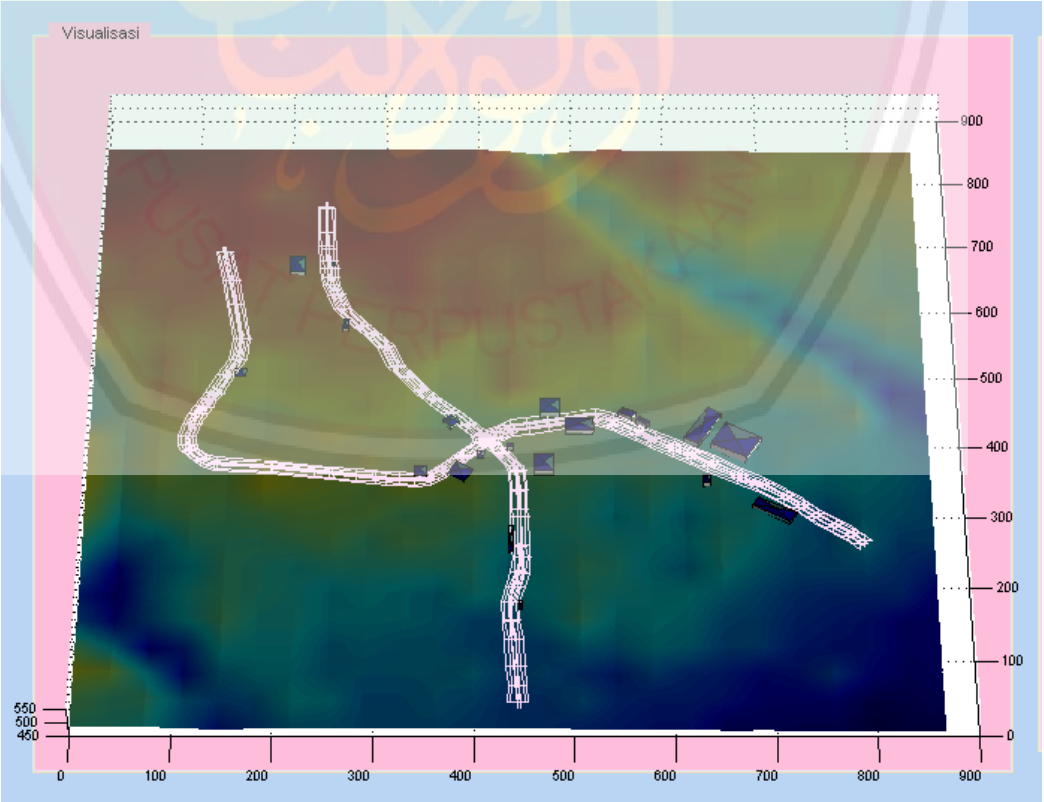
   Pop-up menu berisi pilihan lokasi yang akan divisualisasikan, namun pada penelitian kali ini peneliti hanya focus pada satu lokasi yaitu area sekitar jalan Veteran. Berikut output yang dihasilkan dari pop-up menu ditunjukan pada Gambar 3.27

```
Command Window

tempat_yg_dipilih =

Veteran

tempat_yg_dipilih =

D:\PROGRAM\Data\Veteran\
```

Gambar 3.27 Output diperoleh pop-up menu

2. *Push Button* (Visualisasi lokasi)

   Push Button Visualisasi Lokasi untuk menampilkan lokasi permukaan tanah, bangunan dan jalan. Output yang diperoleh dari *push button* visualisasi lokasi ditunjukan pada Gambar 3.28



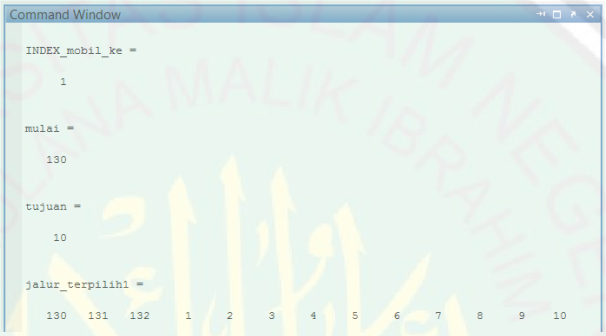
Gambar 3.28 Visualisasi permukaan tanah

3. Axes2 (Visualisasi)

   Axes2 berfungsi untuk menampilkan hasil dari visualisasi

4. *Push Button* (1)

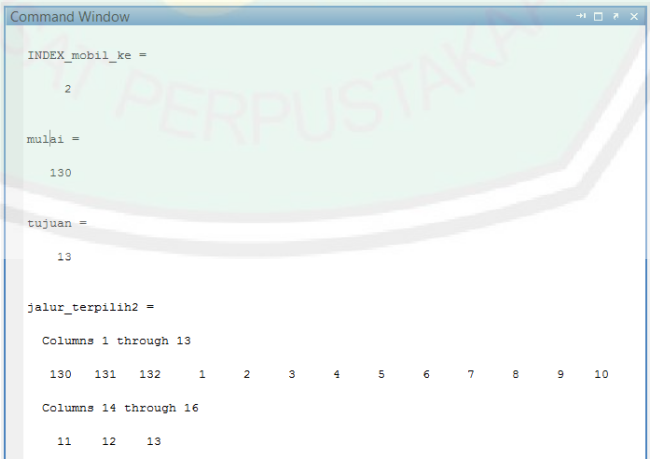
   Push Button 1 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan pertama. Output yang diperoleh *push button* 1 ditunjukan pada Gambar 3.29



Gambar 3.29 Output *push button* 1

5. *Push Button* (2)

   Push Button 2 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan kedua. Output yang diperoleh *push button* 2 ditunjukan pada Gambar 3.30



Gambar 3.30 Output *push button* 2

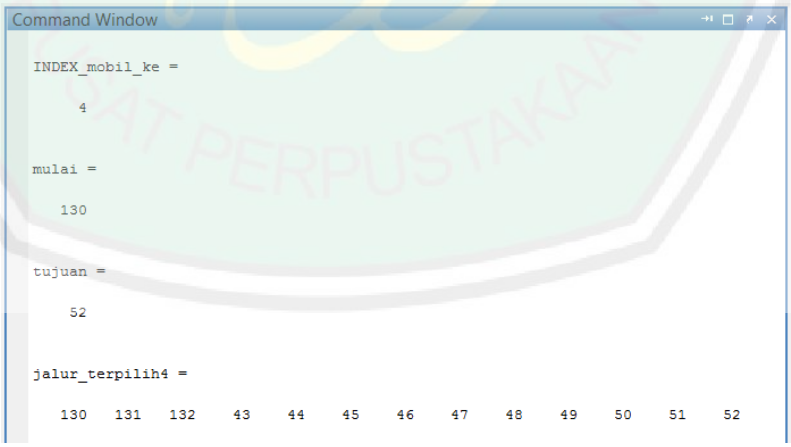
6. *Push Button* (3)

Push Button 3 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ketiga. Output yang diperoleh *push button* 3 ditunjukan pada Gambar 3.31

Gambar 3.31 Output *push button* 3

7. *Push Button* (4)

Push Button 4 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan keempat. Output yang diperoleh *push button* 4 ditunjukan pada Gambar 3.32



Gambar 3.32 Output *push button* 4

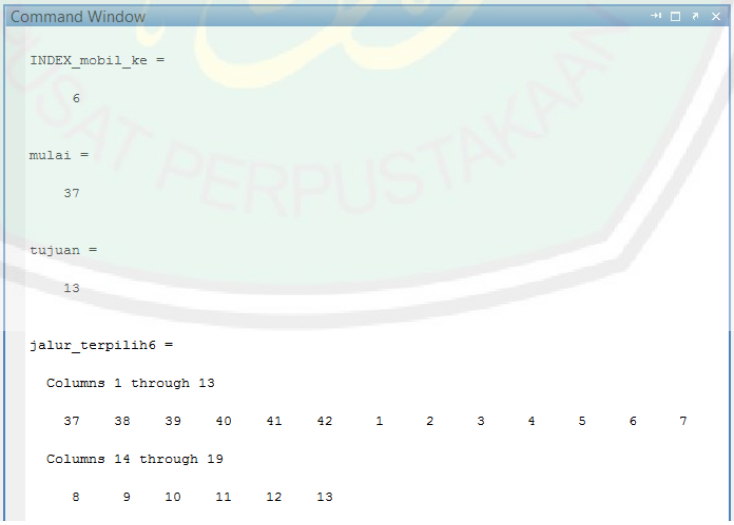
8. *Push Button* (5)

Push Button 5 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan kelima. Output yang diperoleh *push button* 5 ditunjukan pada Gambar 3.33

Gambar 3.33 Output *push button* 5

9. *Push Button* (6)

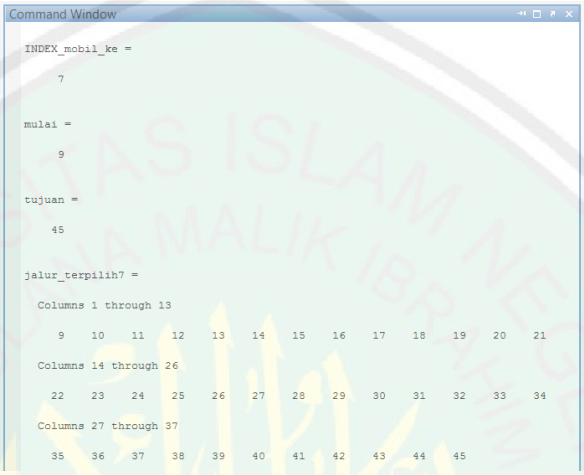
Push Button 6 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke enam. Output yang diperoleh *push button* 6 ditunjukan pada Gambar 3.34



Gambar 3.34 Output *push button* 6

10. *Push Button* (7)

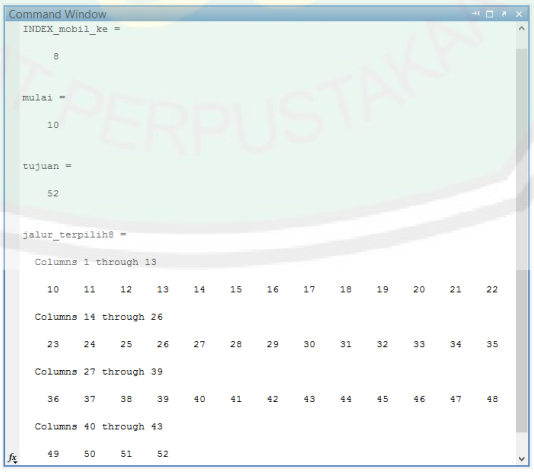
Push Button 7 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ketuju. Output yang diperoleh *push button* 7 ditunjukan pada Gambar 3.35



Gambar 3.35 Output *push button* 7

11. *Push Button* (8)

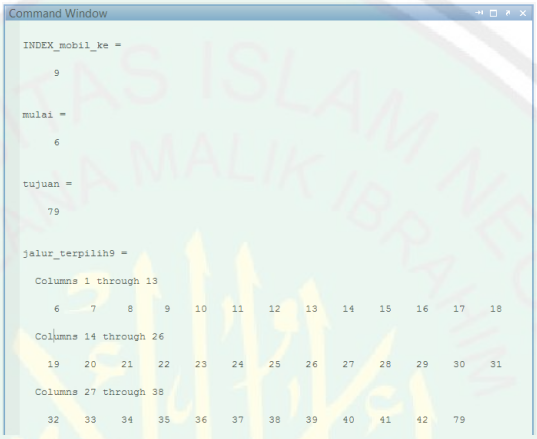
Push Button 8 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan kedelapan. Output yang diperoleh *push button* 8 ditunjukan pada Gambar 3.36



Gambar 3.36 Output *push button* 8

12. *Push Button* (9)

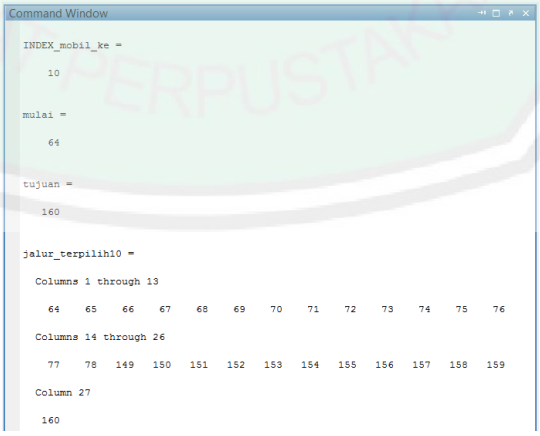
Push Button 9 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan kesembilan. Output yang diperoleh *push button* 9 ditunjukan pada Gambar 3.37



Gambar 3.37 Output *push button* 9

13. *Push Button* (10)

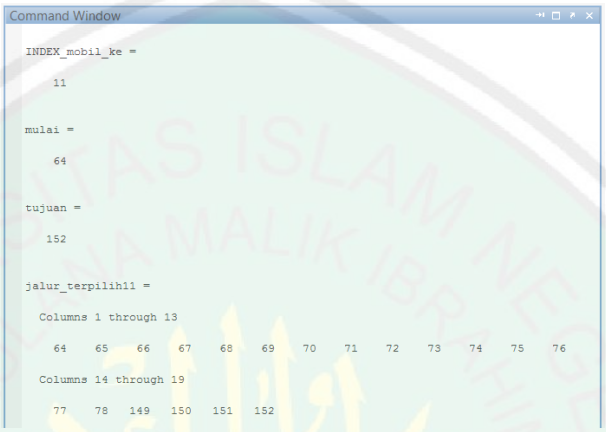
Push Button 10 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 10. Output yang diperoleh *push button* 10 ditunjukan pada Gambar 3.38



Gambar 3.38 Output *push button* 10

14. *Push Button* (11)

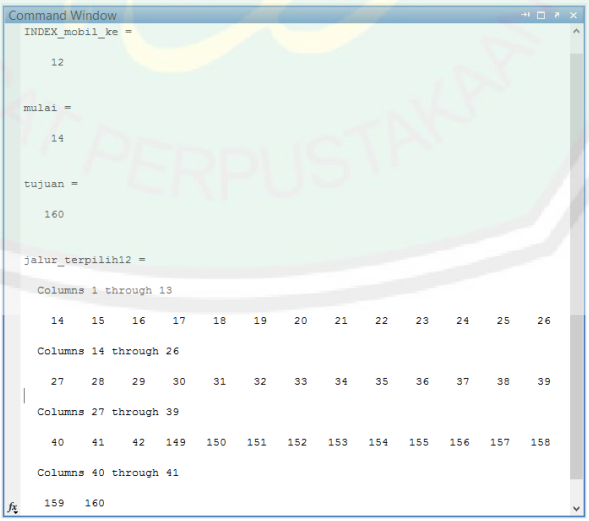
Push Button 11 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 11. Output yang diperoleh *push button* 11 ditunjukan pada Gambar 3.39



Gambar 3.39 Output *push button* 11

15. *Push Button* (12)

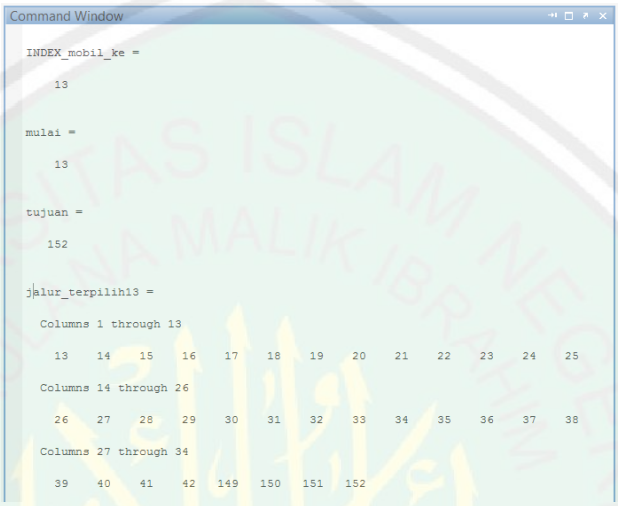
Push Button 12 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 12. Output yang diperoleh *push button* 12 ditunjukan pada Gambar 3.40



Gambar 3.40 Output *push button* 12

16. *Push Button* (13)

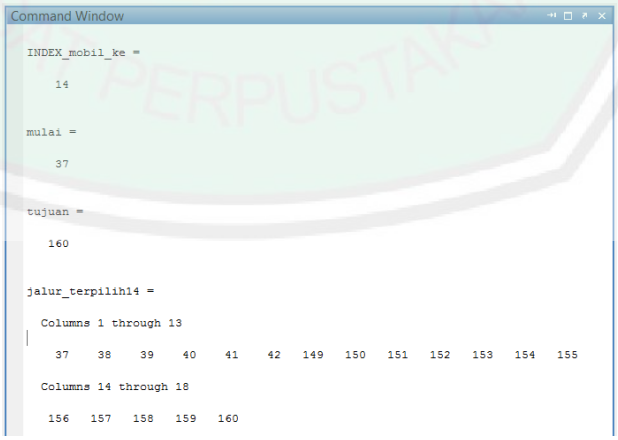
Push Button 13 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 13. Output yang diperoleh *push button* 13 ditunjukan pada Gambar 3.41



Gambar 3.41 Output *push button* 13

17. *Push Button* (14)

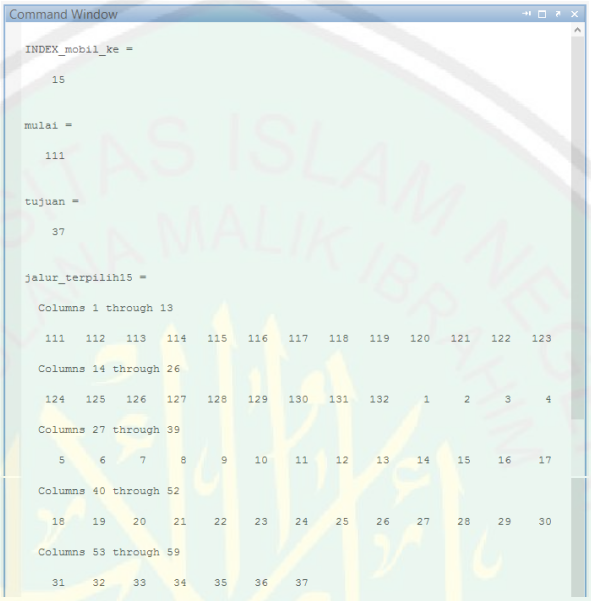
Push Button 14 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 14. Output yang diperoleh *push button* 14 ditunjukan pada Gambar 3.42



Gambar 3.42 Output *push button* 14

18. *Push Button* (15)

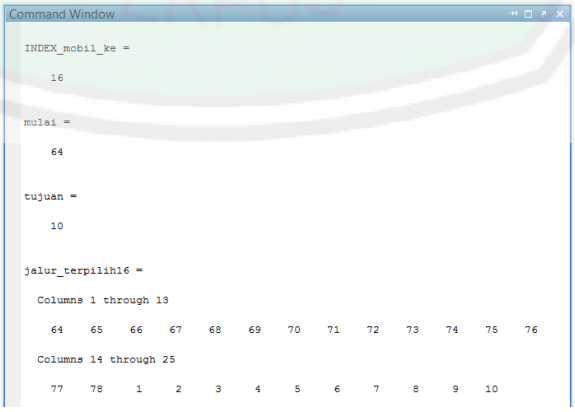
Push Button 15 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 15. Output yang diperoleh *push button* 15 ditunjukan pada Gambar 3.43



Gambar 3.43 Output *push button* 15

19. *Push Button* (16)

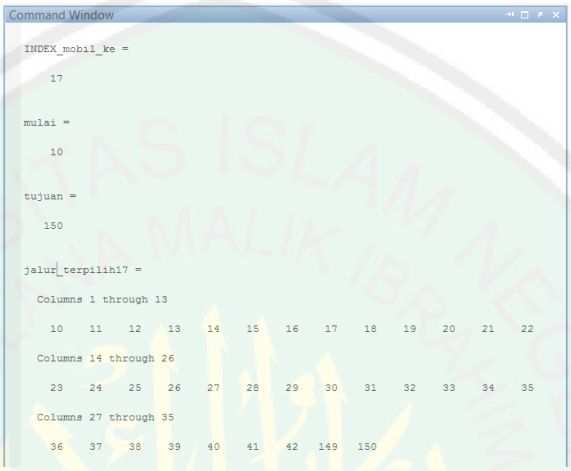
Push Button 16 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 16. Output yang diperoleh *push button* 16 ditunjukan pada Gambar 3.44



Gambar 3.44 Output *push button* 16

20. *Push Button* (17)

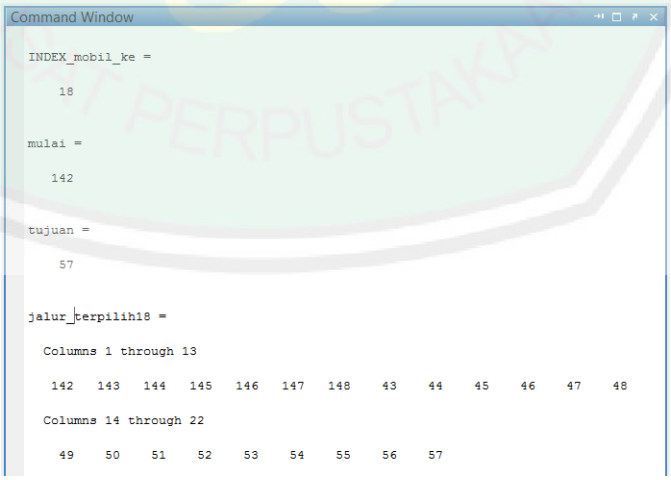
Push Button 17 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 17. Output yang diperoleh *push button* 17 ditunjukan pada Gambar 3.45



Gambar 3.45 Output *push button* 17

21. *Push Button* (18)

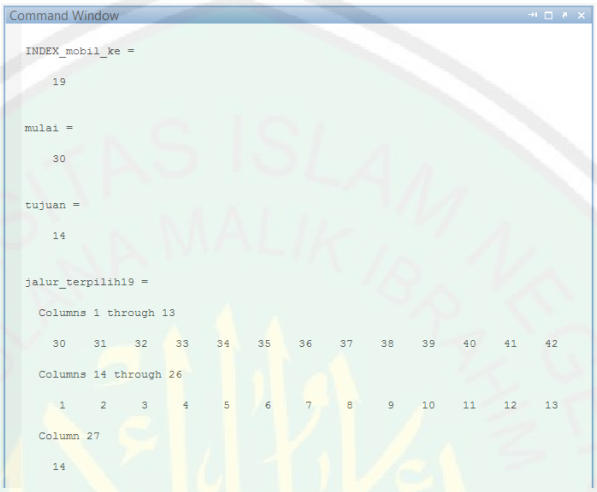
Push Button 18 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 18. Output yang diperoleh *push button* 18 ditunjukan pada Gambar 3.46



Gambar 3.46 Output *push button* 18

22. *Push Button* (19)

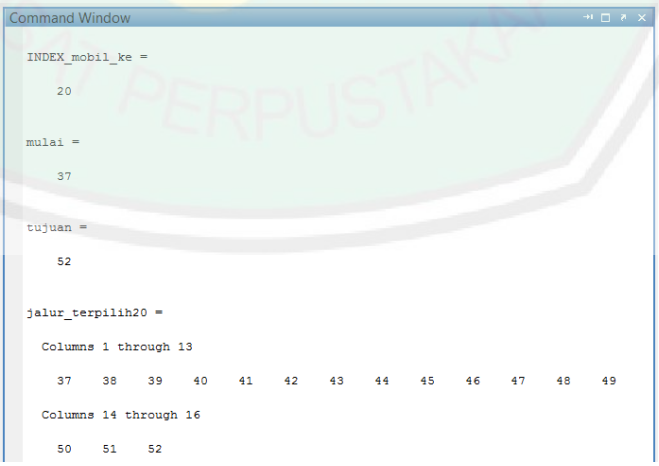
Push Button 19 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 19. Output yang diperoleh *push button* 19 ditunjukan pada Gambar 3.47



Gambar 3.47 Output *push button* 19

23. *Push Button* (20)

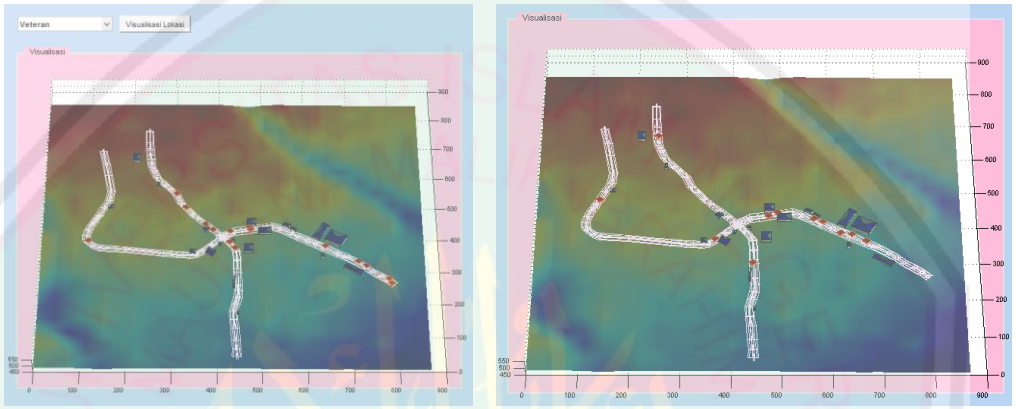
Push Button 20 berfungsi untuk mengeluarkan rute perjalanan pada kendaraan ke 20. Output yang diperoleh *push button* 20 ditunjukan pada Gambar 3.48



Gambar 3.48 Output *push button* 20

24. *Push Button* (Visualisasi Mobil)

Push Button Visualisasi Mobil berfungsi untuk memanggil ke 20 rute perjalanan kendaraan kemudian melakukan visualisasi pergerakan. Output yang diperoleh *push button* visualisasi mobil ditunjukan pada gambar 3.49



(a) (b)

Gambar 3.49 (a) pergerakan pertama (b) pergerakan kedua

Perhatikan warna merah kotak pada Gambar 3.48 adalah simulasi pergerakan kendaraan berdasarkan rute perjalanan yang diperoleh dari push button 1 s/d 20. Jumlah mobil yang divisualisasikan adalah sebanyak 20 kendaraan. Jadi dika akan menambah jumlah kendaraan lagi, maka diperlukan push button 21 dan seterusnya dan proses di masing masing push button yang baru ditambahkan. Karena pada dasarnya setiap kendaraan yang digerakkan memiliki proses pencarian menggunakan metode Depth First Search (DFS) masing masing.

BAB IV

UJI COBA DAN PEMBAHASAN

Pada bab ini peneliti akan membahas tentang pengujian dan pembahasan terhadap implementasi yang telah dilakukan pada aplikasi visualisasi pergerakan kendaraan.

4.1 Langkah – langkah Uji Coba

Langkah – langkah uji coba menjelaskan bagaimana aplikasi ini diuji. Aplikasi ini adalah berbasis desktop. Sebelum menjelaskan tentang pengujian dan pembahasan, berikut platform yang digunakan dalam membangun aplikasi visualisasi

- Hardware

Prosesor : Pentium(R) Dual-Core 2.00GHz

RAM : 2.00GB

OS : Windows 8

- Software

Software : Matlab R2010a

Pendukung : GoogleEarth, Microsoft Excel 2010

Terdapat beberapa parameter yang akan diuji cobakan pada penelitian kali ini, antara lain adalah sebagai berikut :

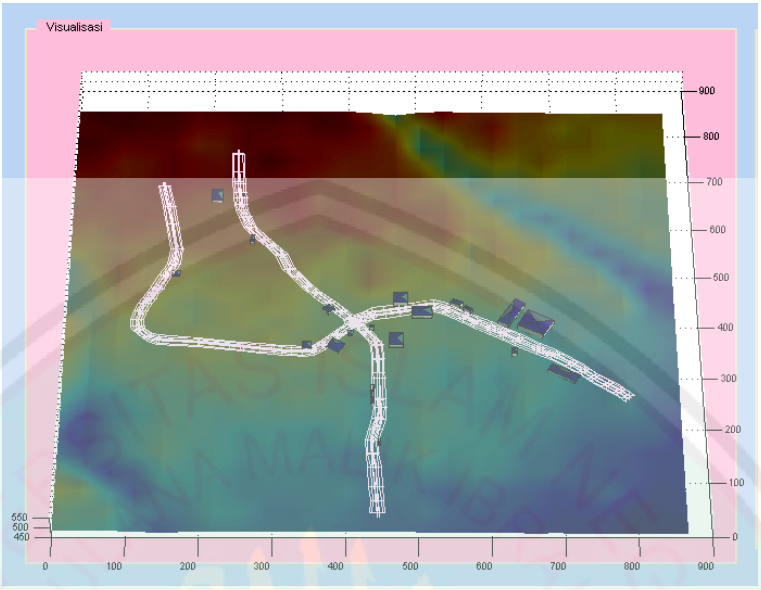
1. Menampilkan permukaan tanah, dimana lokasi tersebut adalah area yang akan dilakukan pergerakan kendaraan dengan menampilkan bangunan sebagai terminal atau tempat mobil dan jalan sebagai sarana untuk pergerakan kendaraan. Untuk menampilkan permukaan tanah adalah dengan menggunakan data SRTM, untuk luas wilayah dibatasi pada data excel yang berisi batas bawah dan batas atas. Untuk menampilkan jalan dan bangunan berdasarkan input data *latitude longitude* yang diperoleh dari marker GoogleEarth yang telah disimpan dalam database berbentuk excel dengan nama file *road* dan *building*.
2. *Euclidian distance*, mencari titik terdekat. Sebelum melakukan pergerakan atau mendapatkan jalur pergerakan, kedua yang dilakukan adalah mencari titik terdekat jalan dari bangunan asal mobil. Pertama, mobil memilki asal dan tujuan yang masing masing terletak pada data *building*, karena mobil akan bergerak berdasarkan data *road*, maka dibutuhkan proses pencarian titik terdekat antara *building* dan *road*. Titik yang diperoleh adalah titik pada *road* yang berada disebelah titik *building* yang diinputkan.
3. Pencarian menggunakan Depth First Search (DFS), setelah mendapatkan titik asal dan tujuan pada data *road*, kemudian aplikasi akan melakukan pencarian jalur dari titik asal menuju titik tujuan. Aplikasi akan menampilkan rute perjalanan jika tujuan telah ditemukan. Adapun parameter yang digunakan adalah titik awal dan titik tujuan yang berupa nilai index pada setiap *lat lang.*
4. Pergerakan kendaraan, setelah menemukan rute perjalanan maka aplikasi akan memvisualisasikannya dalam bentuk pergerakan kendaraan berdasarkan jalur

yang telah diperoleh diatas permukaan tanah dengan menampilkan bangunan dan jalan sebagai sarana penghubung. Pada bagian uji coba pergerakan ini, peneliti membuat uji coba pergerakan dengan dua mode pergerakan. Yang pertama adalah pergerakan dengan 20 kendaraan beberapa mobil memiliki asal yang sama dan tujuan yang berbeda dan beberapa memiliki tujuan yang sama namun asal berbeda atau dalam kata lainnya 20 kendaraan dengan pola pergerakan acak, yang kedua adalah pergerakan 10 kendaraan dengan membuat asal semua kendaraan sama namun tujuan berbeda. dimana pergerakannyapun berbeda dengan pola pergerakan pertama yaitu pola pergerakan antrian.

4.2 Hasil Uji Coba

Setelah langkah langkah pengujian dilakukan maka diperoleh data hasil pegujian. Data tersebut disajikan dalam bentuk table dan screenshot gambar. Hasil pengujian antara lain adalah sebagai berikut :

Hasil pertama diperoleh dari proses pertama yaitu menampilkan permukaan tanah, bangunan dan jalan. Pada aplikasi yang peneliti buat proses pertama dilakukan oleh button ‘Visualisasi Lokasi’. Output yang diperoleh adalah visualisasi permukaan tanah dan bangunan yang ditunjukan pada Gambar 4.1



Gambar 4.1 Visualisasi permukaan tanah, bangunan dan jalan

Untuk lebar atau ukuran permukaan tanah yang akan ditampilkan adalah berdasarkan inputan koordinat latitude longitude pada database excel yang bernama 'Batas_Wilayah', adapun batas wilayah untuk lokasi permukaan tanah seperti ditunjukan pada table 4.1

Tabel 4.1 Batas_Wilayah

No	Latitude	Longitude	Keterangan
1	-7.946280	112.603196	Batas Awal
2	-7.966041	112.623230	Batas Akhir

Hasil uji coba yang kedua adalah nilai longitude dan latitude terdekat jalan dari bangunan tempat asal atau tujuan mobil adalah seperti ditunjukan pada table 4.2

Tabel 4.2 Hasil pengujian Euclidian Distance

No	Nama bangunan	Lat bangunan	Lon bangunan	Lat jalan	Lon Jalan	Index jalan
1	UIN	-7.950897	112.608171	-7.950915	112.608891	57
2	ITN	-7.957835	112.612190	-7.957794	112.611818	81
3	Matos	-7.956883	112.618776	-7.957399	112.618216	14
4	UB	-7.955735	112.614347	-7.956087	112.614262	6
5	UM1	-7.956342	112.615050	-7.957563	112.618081	30
6	POM	-7.957037	112.613376	-7.957152	112.613189	150
7	SMK	-7.956342	112.615050	-7.956156	112.614751	37
8	Indomaret	-7.957211	112.612676	-7.956862	112.612738	79
9	BRI uin	-7.950753	112.609050	-7.950676	112.608937	64
10	Puskesmas	-7.954489	112.606864	-7.954589	112.606565	111
11	Ruko	-7.959957	112.613413	-7.960170	112.613574	142
12	Bengkel	-7.962072	112.613633	-7.961953	112.613432	160
13	UMM	-7.957614	112.614180	-7.957732	112.613558	152
14	Kantor ub	-7.955923	112.616179	-7.956029	112.615849	9
15	Pesantren	-7.952932	112.609405	-7.952917	112.609728	52
16	Toko Kue	-7.957868	112.611251	-7.957974	112.611448	130
17	Percetakan	-7.956187	112.611968	-7.955848	112.611935	45
18	UM2	-7.959062	112.619631	-7.958616	112.619835	27
19	MX	-7.956370	112.617979	-7.957141	112.617781	13
20	Masjid ub	-7.956204	112.616544	-7.956393	112.616449	10

Berdasarkan hasil yang diperoleh pada table 4.2 menunjukan nilai kedekatan antar bangunan dengan titik terdekat jalan sudah sesuai jika dibandingkan dengan sketsa manual peneliti. Dan setiap titik yang diperoleh akan digunakan sebagai asal dan tujuan proses pencarian rute perjalanan. Adapun selisih yang diperoleh dari nilai terdekat yang didapat pada tabel 4.2 ditunjukan pada tabel 4.3

Tabel 4.3 selisih *Euclidean distance*

Bangunan ke	Nama bangunan	Selisih latitude	Selisih longitude
1	UIN	0.000018	0.000072
2	ITN	0.000041	0.000372
3	Matos	0.000518	0.000056
4	UB	0.000352	0.000085
5	UM1	0.001221	0.003031
6	POM	0.000115	0.000187
7	SMK	0.000186	0.000299
8	Indomaret	0.000346	0.000062
9	BRI uin	0.000077	0.000113
10	Puskesmas	0.000001	0.000299
11	Ruko	0.000213	0.000161
12	Bengkel	0.000119	0.000201
13	UMM	0.000118	0.000622
14	Kantor ub	0.000106	0.000033
15	Pesantren	0.000015	0.000323
16	Toko Citra	0.000106	0.000197
17	Percetakan	0.000339	0.000033
18	UM2	0.000446	0.000771
19	MX	0.000771	0.000198
20	Masjid ub	0.000189	0.000095

Jika diamati pada selisih yang diperoleh pada tabel 4.2 selisih terkecil yang diperoleh adalah 0.000001 dan selisih terbesar yang diperoleh adalah 0.003031 hal ini dikarenakan pencarian nilai terdekat dibatasi hanya pada data yang telah tersedia. Oleh karena itu terdapat selisih yang bervariatif.

Selanjutnya pada data '*road'* terdapat *index jalan* yang bisa dilihat pada table 4.2 pada baris ke 6 merupakan parameter yang akan digunakan dalam proses pencarian

jalur. Dimana nilai nilai yang telah diperoleh tersebut digunakan sebagai patokan peneliti untuk melihat keseuaian rute yang diperoleh nantinya

Adapun hasil uji coba yang diperoleh dari implementasi metode DFS adalah ditunjukan pada table 4.3

Tabel 4.4 Jalur yang diperoleh Metode DFS

No	Asal	Tujuan	Rute diproleh	Jumlah Node	Kesesuaian (Y/N)
1	Toko kue	Masjid	130,131,132,1,2,3,4,5,6,7,8,9,10	13	Y
2	@MX	ITN	13,14,15,16,17,18,19,20,21,22,2324,25,26,27,28,29,30,31,32,33,3435,36,37,38,39,40,41,42,79,80,81	33	Y
3	Masjid	UIN	10,11,12,13,14,15,16,17,18,19,2,21,22,23,24,25,26,27,28,29,30,3,32,33,34,35,36,37,38,39,40,41,4243,44,45,46,47,48,49,50,51,52,5354,55,56,57	48	Y
4	Matos	ITN	14,15,16,17,18,19,20,21,22,23,2425,26,27,28,29,30,31,32,33,34,3536,37,38,39,40,41,42,79,80,81	32	Y
5	UB	Bengkel	6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,149,150,151,152,153,154,155,156,157,158,159,160	49	Y
6	Puskesmas	@MX	111,112,113,114,115,116,117,118,119,120121,122,123,124,125,126,127,128,129,130131,132,1,2,3,4,5,6,7,8,9,10,11,12,13	35	Y
7	Ruko	Masjid	142,143,144,145,146,147,148,1,23,4,5,6,7,8,9,10	17	Y
8	UM	ITN	30,31,32,33,34,35,36,37,38,39,4041,42,7980,81	18	Y

9	UB	Indomaret	6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,41,42,79	38	Y
10	BRI uin	Bengkel	64,65,66,67,68,69,70,71,72,73,74,75,76,7778,149,150,151,152,153,154,155,156,157 158,159,160	27	Y
11	BRI uin	UMM	64,65,66,67,68,69,70,71,72,73,7475,76,77,78,149,150,151,152	19	Y
12	Matos	Bengkel	14,15,16,17,18,19,20,21,22,23,24,25,26,2728,29,30,31,32,33,34,35,36,37,38,39,40,4142,149,150,151,152,153,154,155,156,157,158,159,160	41	Y
13	@MX	UMM	13,14,15,16,17,18,19,20,21,22,23,24,25,2627,28,29,30,31,32,33,34,35,36,37,38,39,4041,42,149,150,151,152	34	Y
14	SMK	Bengkel	37,38,39,40,41,42,149,150,151,152,153,154,155,156,157,158,159,160	18	Y
15	Puskesmas	SMK	111,112,113,114,115,116,117,118,119,120121,122,123,124,125,126,127,128,129,130131,132,1,2,3,4,5,6,7,8,9,10,11,12,13,14,1516,17,18,19,20,21,22,23,24,25,2627,28,29,30,31,32,33,34,35,36,37	59	Y
16	BRI uin	Masji	64,65,66,67,68,69,70,71,72,73,7475,76,77,78,1,2,3,4,5,6,7,8,9,10	25	Y
17	BRI uin	POM	64,65,66,67,68,69,70,71,72,73,74,75,76,7778,149,150,151,152	19	Y
18	Ruko	UIN	142,143,144,145,146,147,148,43,44,45,46,47,48,49,50,51,52,53,5455,56,57	22	Y
19	UM	Matos	30,31,32,33,34,35,36,37,38,39,4041,42,1,23,4,5,6,7,8,9,10,11,12,13,14	26	Y
20	SMK	Percetakan	37,38,39,40,41,42,1,2,3,4,5,6,7,8,9,10,11,12,13	19	Y

Berdasarkan tabel 4.3 didapatkan prosesntase nilai akurasi sebesar 100% dan nilai error 0 yang diperoleh dari rumus 4.1

Rumus 4.1

$$\frac{data\ akurasi\ sesuai}{data\ keseluruhan} = \frac{20}{20} x100\% = 100\%$$

Untuk memperoleh jalur seperti pada table 4.3 diatas adalah harus menggunakan susunan data seperti tree yang benar. Dimana setiap node pada data memiliki nilai yang berhubungan dan berkelanjutan, dan setiap nilai memiliki kondisi yang berbeda. Contohnya adalah seperti table 4.4 berikut :

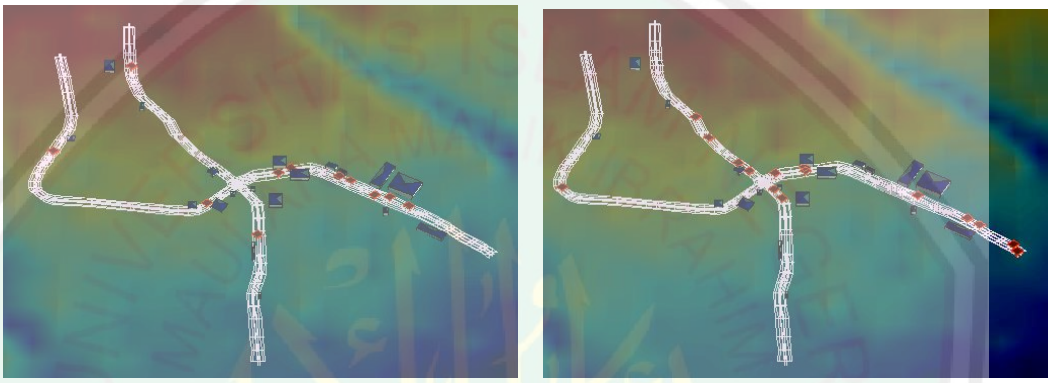
Table 4.5 Nilai kondisi

No	Nilai	Kondisi
1	0	Adalah kondisi berjalan atau jalan lurus
2	1	Adalah kondisi berhenti atau terdapat jalur buntu
3	2	Adalah kondisi dimana terdapat cabang atau pilihan

Kondisi seperti pada tabel 4.4 hanya diimplementasikan pada program ketika pencarian rute perjalanan. Dimana jika kondisi menemui satu maka program harus mencari jalur lain karena jalan buntu. Hal ini sama halnya dengan kondisi nol dan dua. Kondisi nol akan diabaikan karena program akan membaca terus data tersebut. Dan jika bertemu kondisi dua, maka akan muncul pilihan cabang sesuai banyak kondisi dua yang ditemui pada data.

Hasil uji coba selanjutnya adalah melakukan pergerakan kendaraan. Pada uji coba pergerakan kendaraan ini peneliti membagi pola pergerakan menjadi 2 pola

pergerakan. Yang kepertama yaitu pergerakan 20 kendaraan dengan masing masing asal dan tujuan acak, dengan kondisi beberapa kendaraan memiliki asal atau tujuan yang sama, hal ini lebih untuk memperlihatkan kondisi keramaian lalu lintas. Berikut hasil coba pola pergerakan pertama ditunjukan pada gambar 4.2



(a) (b)

Gambar 4.2 (a) dan (b) hasil uji coba pergerakan kedua

Dari ke 20 pergerakan kendaraan seperti yang di tunjukan pada Gambar 4.2 diperoleh tabel ketepatan pergerakan kendaraan berdasarkan asal dan tujuannya ditunjukan pada tabel 4.6

Tabel 4.6 ketepatan pergerakan berdasarkan asal dan tujuan

Mobil ke	Pencarian jalur (Detik)	Pergerakan (Menit)	Kesesuaian Asal (Y/T)	Kesesuaian Tujuan (Y/T)	Ketepatan (Y/T)
1	01.494	01:06	Y	T	T
2	01.520	02:58	Y	T	T
3	01.669	03:52	Y	Y	Y
4	01.169	02:19	Y	Y	Y
5	01.925	04:24	Y	Y	Y
6	02.285	02:57	Y	Y	Y
7	01.717	01:19	Y	Y	Y
8	01.352	01:30	Y	Y	Y
9	01.849	03:26	Y	Y	Y
10	02.061	02:16	Y	Y	Y
11	01.690	01:34	Y	Y	Y
12	01.813	03:22	Y	Y	Y
13	01.975	02:49	Y	Y	Y
14	01.993	01:30	Y	Y	Y
15	02.162	04:54	Y	Y	Y
16	01.490	01:58	Y	Y	Y
17	01.877	01:34	Y	Y	Y
18	01.523	01:45	Y	Y	Y
19	01.527	02:08	Y	Y	Y
20	01.461	01:34	Y	Y	Y

hasil uji coba yang diperoleh dari pergerakan kendaraan seperti pada tabel 4.6 memperoleh nilai akurasi sebesar 90% yang diperoleh dari rumus 4.2

Rumus 4.2

$$\frac{data\ akurasi\ sesuai}{data\ keseluruhan} = \frac{18}{20} x100\% = 90\%$$

Berdasarkan tabel 4.6 telah diperoleh prosesntase ketepatan pergerakan dengan nilai 90% dan nilai error 10%

Selanjutnya untuk estimasi waktu yang digunakan untuk menemukan jalur menggunakan DFS diperoleh waktu 1.726 detik yang diperoleh dari rumus 4.3

Rumus 4.3

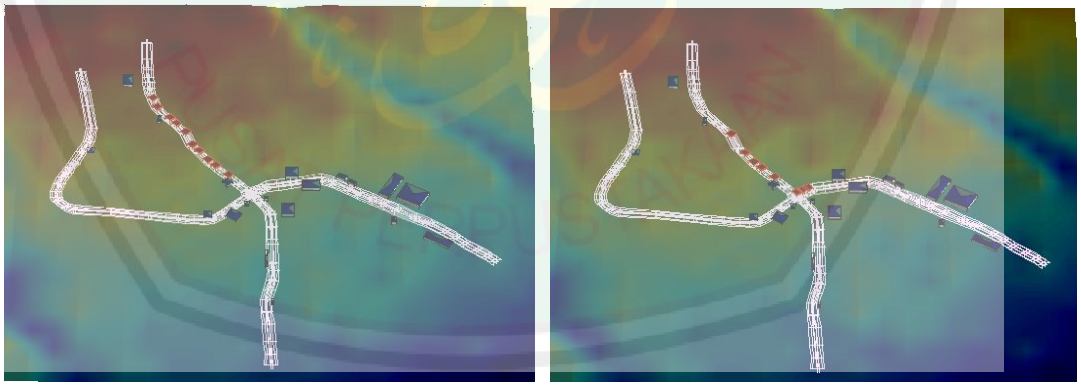
$$\frac{\sum waktu\ pencarian}{data\ keseluruhan} = \frac{34.552}{20} = 1.726\ detik$$

Untuk estimasi waktu pergerakan kendaraan berdasarkan jalur yang telah diperoleh menggunakan metode DFS diperoleh waktu 130.6 menit yang diperoleh dari rumus 4.4

Rumus 4.4

$$\frac{\sum waktu\ pergerakan}{data\ keseluruhan} = \frac{2612}{20} = 130.6\ menit$$

Pola pergerakan yang kedua adalah pergerakan 10 kendaraan dengan pola antrian dimana 10 kendaraan akan keluar satu persatu dengan asal kendaraan sama dan tujuan masing masing berbeda. berikut hasil uji coba pergerakan kedua seperti ditunjukan pada Gambar 4.3



(a) (b)

Gambar 4.3 (a) dan (b) hasil uji coba pergerakan kedua

Adapun jalur yang diperoleh seperti pada Gambar 4.3 ditunjukan pada tabel 4.7 dimana kendaraan keluar pada asal yang sama dan berjalan sampai membentuk kondisi atrian pada jalan.

Tabel 4.7 Jalur pergerakan kedua

Mobil Ke	Aasal	Tujuan	Rute Diperoleh	Jumlah Node	Waktu (Menit)
1	BRI UIN	Matos	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 1,2,3,4,5,6,7,8,9,10 ,11,12,13,14	29	02:24
2	BRI UIN	MX	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 1,2,3,4,5,6,7,8,9,10 ,11,12,13	28	02:22
3	BRI UIN	UB	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 1,2,3,4,5,6	21	01:49
4	BRI UIN	POM	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 149,150	17	01:25
5	BRI UIN	Masjid	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 1,2,3,4,5,6,7,8,9,10	25	02:08
6	BRI UIN	UM1	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 1,2,3,4,5,6,7,8,9,10 ,11,12,13,14, 15,16 ,17,18 19,20,21,22 ,23,24, 25,26,27,28,29,30	45	03:52
7	BRI UIN	Kantor	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 1,2,3,4,5,6,7,8,9	24	02:02
8	BRI UIN	UM2	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 1,2,3,4,5,6,7,8,9,10 ,11,12,13,14, 15,16 ,17,18 19,20,21,22 ,23,24, 25,26,27	41	03:40
9	BRI UIN	Indomart	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 79	16	01:27
10	BRI UIN	UMM	64,65,66,67,68,69,70,71,72,73,74,75,76,77,78, 149,150,151,152	19	01:44

Dari waktu perjalanan yang diperoleh seperti pada tabel 4.3 bahwa setiap waktu kendaraan yang diperoleh telah ditambah sebanyak 5 detik waktu antrian dimulai pada mobil ke tiga karena antrian baru dimulai dari mobil ke dua dan untuk waktu pergerakan selanjutnya tergantung pada jumlah rute/node yang akan dilewati masing masing kendaaraan.

Dari kedua uji coba yang ditunjukan pada Gambar 4.2 dan 4.3 telah menunjukan 90% dan 100% kesesuaian antara rute perjalanan yang diperoleh dan viaslisasi pergerakannya diatas permukaan tanah

Hasil uji coba selanjutnya yaitu untuk menentukan nilai tujuan terdekat berdasarkan jumlah node yang paling kecil, dimana uji coba ini menggunakan lokasi bangunan Universitas Brawijaya dimana terdapat dua pintu masuk yang peneliti simulasikan dalam dua bangunan. Pada uji coba ini peneliti gunakan 2 buah mobil yang akan berjalan pada tujuan yang saling berdekatan dimana peneliti akan menghitung akurasi dari jalur dan pergerakan kendaraan tersebut ditunjukan pada tabel 4.8

Tabel 4.8 uji coba berdasarkan tujuan terdekat

Mobil ke	Asal	Tujuan	jalur	Jumlah node	Waktu pergerakan	Kesesuaian Y/T
1	BRI UIN	UB (gerbang 1)	64,65,66,67,68,69,70,71,72,73,74, 75,76,77,78,1,2,3,4,5,6	21	1:40 (menit)	Y
2	BRI UIN	UB (gerbang 2)	64,65,66,67,68,69,70,71,72,73,74, 75,76,77,78,1,2,3,4,5,6,7,8	23	2:26 (menit)	Y

Dari uji coba pencarian jalur dan pergerakan berdasarkan pada tabel 4.8 dapat diperoleh akurasi kesesuaian jalur yang diperoleh dan kesesuaian pergerakan 100% dan diperoleh estimasi waktu pergerakan ≤ 2:15 menit dari jumlah seluruh waktu dibagi data keseluruhan. brikut rumus akurasi pergerakan kendaraan

Rumus 4.5

$$\frac{data\ akurasi\ sesuai}{data\ keseluruhan} = \frac{2}{2} x100\% = 100\%$$

Jika diamati dari tabel 4.8 dimana selisih jumlah node adalah 2 dan selisih waktu 46 detik. Dapat diketahui bahwa tujuan UB (gerbang 1) merupakan jalur terpendek daripada tujuan UB (gerbang 2).

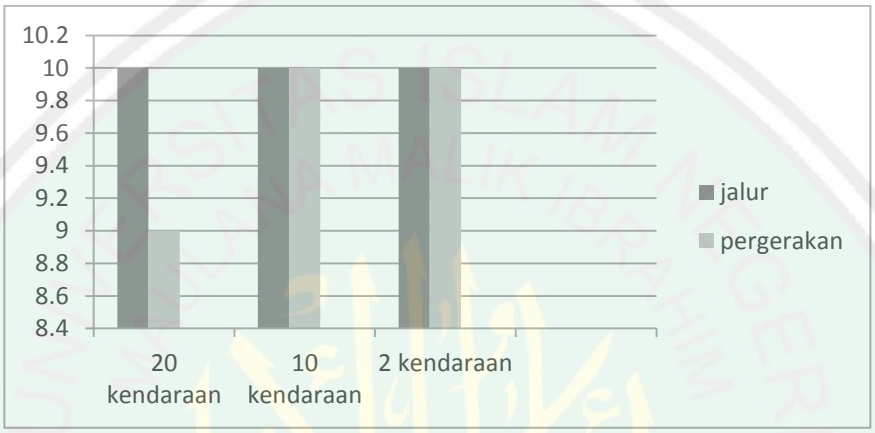
4.3 Pembahasan

Setelah didapatkan hasil uji coba kemudian dilakukan analisa data atau pembahasan terhadap hasil uji coba sebagai bahan untuk pengembangan aplikasi selanjutnya dan agar didapatkan pengetahuan atau ilmu pengetahuan baru berdasarkan hasil penelitian. Berikut pembahasan hasil uji coba

1. Pembahasan hasil uji coba jalur atau rute perjalana yang diperoleh menggunakan Metode Depth Fisrt Search (DFS) terbukti 100% mampu menunjang visualisasi pergerakan kendaraan. Adapun 100% diperoleh dari rute perjalanan 20 kendaraan yang masing masing sesuai dengan syarat yang telah ditentukan sebelumnya. Dari hasil coba yang telah diperoleh bahwa dengan menggunakan metode Depth First Search (DFS) mendapatkan rute perjalanan terpendek yaitu 13 node dan juga mendapatkan rute perjalanan terpanjang yaitu 59 node. hal ini tentu saja berdasarkan asal tujuan yang telah ditentukan masing masing kendaraan. Sedangkan estimasi waktu yang yang digunakan untuk

memperoleh masing - masing jalur pada tiap kendaraan adalah rata – rata 1.726 detik diperoleh dari jumlah waktu diperoleh dibagi dengan jumlah data.

2. Pembahasan hasil uji coba pergerakan kendaraan berdasarkan hasil pengamatan peneliti ditunjukan pada diagram berikut



Pada uji coba pergerakan peneliti menggunakan dua model pergerakan yang berbeda dan satu model pergerakan dimana tujuannya adalah menentukan titik terdekat lokasi tujuan, pada model pergerakan pertama peneliti mengacak asal dan tujuna kendaraan dimana kendaraan akan keluar bersamaan dengan asal dan tujuan berbeda dan kemuadian pergerakannyapun menyesuaikan rute yang telah diperoleh adapun pergerakan ini dapat di istilahkan sebagai pergerakan kendaraan pada keadaan sebenarnya yaitu *unpredictable* (tidak dapat diprediksi). Jadi setiap kendaraan bebas bergerak tanpa mengantri dengan kendaran lain. Pada pola pergerakan pertama, perpindahan kenadaraan dari node satu ke node selanjutnya terkesan lama, hal ini dikarenakan terdapat proses *looping* berulang kali pada masing masing kendaraan dan pada pola pergerakan

pertama diperoleh estimasi waktu 130.6 menit. Selanjutnya pola pergerakan kedua dimana semua kendaraan memiliki asal yang sama dan tujuan yang berdekatan. Kendaraan akan keluar satu per satu sesuai dengan urutan indexnya. Dengan demikian mobil akan berderet membentuk antrian pergerakan dimana durasi antrian antar mobil satu dengan yang lainnya berjarak 5 detik. Pada pola pergerakan kedua ini cocok untuk memvisualisasikan kondisi kemacetan karena mobil belakang akan bergerak ketika mobil pada posisi depan maju pada node selanjutnya. Dan pada pergerakan mencari jalur terpendek ialah berdasarkan node paling sedikit dan waktu yang relative lebih cepat jika dilihat pada tabel 4.8 adalah tujuan pada UB (gerbang 1) dengan selisih 2 node dan durasi waktu 46 detik.

Pada latar berlakang telah dijelaskan bahwa salah satu tujuan penelitian ini adalah untuk memvisualisasikan kondisi lalu lintas yang diharapkan mampu bermanfaat bagi kalangan yang membutuhkan salah satunya pemerintah yang menangani bagian transportasi atau pergerakan kendaraan sebagai langkah awal perancangan transportasi yang efektif dan efisien.

Berdasarkan pembahsan hasil uji coba bahwa Metode DFS mampu menunjang visualisasi pergerakan kendaraan bermotor diharapkan mampu menjadi salah satu alat untuk menganalisi data dalam menentukan atau merancang transportasi yang baik. Baik dalam artian pas pada porsinya, dimana tidak ada salah satu pihak yang dirugikan dalam hasil keputusan jalur pergerakan yang akan disosialisasikan pada

masyarakat. Dalam hal ini pemerintah tentu harus adil dalam menentukan kebijakan arah pergerakan kendaraan berdasarkan hasil analisa data yang telah diperoleh.

Pada dasarnya aplikasi ini dibangun bertujuan untuk mempermudah urusan manusia. Mempermudah manusia dalam hal ini dapat dikatakan sebagai tolong menolong dalam menyelesaikan suatu urusan. Seperti yang telah dijelaskan dalam Al-Qur'an bahwasannya kita dianjurkan untuk saling tolong menolong dalam Qs Al-Maidah 5:2 yang berbunyi :

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ لَا تُحِلُّواْ شَعَٰٓئِرَ ٱللَّهِ وَلَا ٱلشَّهۡرَ ٱلۡحَرَامَ وَلَا ٱلۡهَدۡيَ
وَلَا ٱلۡقَلَٰٓئِدَ وَلَآ ءَآمِّينَ ٱلۡبَيۡتَ ٱلۡحَرَامَ يَبۡتَغُونَ فَضۡلٗا مِّن رَّبِّهِمۡ وَرِضۡوَٰنٗاۚ
وَإِذَا حَلَلۡتُمۡ فَٱصۡطَادُواْۚ وَلَا يَجۡرِمَنَّكُمۡ شَنَـَٔانُ قَوۡمٍ أَن صَدُّوكُمۡ
عَنِ ٱلۡمَسۡجِدِ ٱلۡحَرَامِ أَن تَعۡتَدُواْۘ وَتَعَاوَنُواْ عَلَى ٱلۡبِرِّ وَٱلتَّقۡوَىٰۖ وَلَا
تَعَاوَنُواْ عَلَى ٱلۡإِثۡمِ وَٱلۡعُدۡوَٰنِۚ وَٱتَّقُواْ ٱللَّهَۖ إِنَّ ٱللَّهَ شَدِيدُ ٱلۡعِقَابِ ٢

Artinya "*Hai orang-orang yang beriman, janganlah kamu melanggar* ***syi'ar-syi'ar Allah, dan jangan melanggar kehormatan bulan-bulan haram, jangan (mengganggu) binatang-binatang had-ya, dan binatang-binatang qalaa-id, dan jangan (pula) mengganggu orang-orang yang mengunjungi Baitullah sedang mereka mencari kurnia dan keredhaan dari Tuhannya dan apabila kamu telah menyelesaikan ibadah haji, maka bolehlah berburu. Dan janganlah sekali-kali kebencian(mu)***

kepada sesuatu kaum karena mereka menghalang-halangi kamu dari Masjidilharam, mendorongmu berbuat aniaya (kepada mereka). Dan tolong-menolonglah kamu dalam (mengerjakan) kebajikan dan takwa, dan jangan tolong-menolong dalam berbuat dosa dan pelanggaran. Dan bertakwalah kamu kepada Allah, sesungguhnya Allah amat berat siksa-Nya" (Qs Al-Maidah 5:2).

Pada akhir ayat tersebut yang berarti "… *tolong menolonglah kamu dalam kebajikan dan taqwa …",* yang menurut tafsir ibnu katsir memiliki makna diaman Allah ta'ala memerintahkan hamba-hambanya untu senantiasa tolong menolong dalam berbuat kebaikan, itulah yang disebut dengan al-birru (kebajikan), serta meninggalkan segala bentuk kemungkaran, dan itulah yang disebut dengan at-taqwa. Dan Allah melarang mereka tolong menolong dalam hal kebathilan, berbuat dosa dan mengerjakan hal hal yang haram. Diharap penelitian ini dapat bermanfaat sehingga peneliti dapat melaksana perintah Allah dalam hal tolong menolong.

Selain itu juga terdapat hadist yang menjelaskan tentang tolong menolong yaitu memudahkan urusan orang lain Dari Abu Hurairah ra, Nasbi Bersabda "*Barang siapa yang melepaskan satu kesusahan seorang mukmin, pasti Allah akan melepaskan darinya satu kesusahan pada hari kiamat. Barang siapa yang menjadikan mudah urusan orang lain, pasti Allah akan memudahkannya di dunia dan di akhirat. Barang siapa yang menutupi aib seorang muslim, pasti Allah akan menutupi aibnya di dunia dan di akhirat. Allah senantiasa menolong hamba Nya*

selama hamba Nya itu suka menolong saudaranya". (HR. Muslim, lihat juga Kumpulan Hadits Arba'in An Nawawi hadits ke 36).

Apabila kita mengetahui bahwa sebenarnya kita mampu berbuat sesuatu untuk menolong kesulitan orang lain, maka segeralah lakukan, segeralah beri pertolongan. Terlebih lagi bila orang itu telah memintanya kepada kita. Karena pertolongan yang kita berikan, akan sangat berarti bagi orang yang sedang kesulitan

Selain itu, dengan adanya aplikasi ini akan menghasilkan sebuah keputusan dimana keputusan tersebut berhubungan dengan kemaslahata orang banyak yaitu dalam hal tepat waktu. Karena bagaimapun kebijakan menentukan arah pergerakan sangat bepengaruh dengan durasi / waktu dalam pergerakannya. Karna kita harus menggunakan waktu sebaik mungkin, jangan sampai kita membuang waktu dengan sia sia misalkan sepert isitilah "waktu habis ditengah jalan" yang bisa dimaknai waktu terbuang sia sia. Dalam hal ini Al-Qur'an juga sudah menjelaskan tentang anjuran melaksanakan perintah sholat 5 waktu dengan tepat waktu misalnya, terdapat pada Qs An-Nisa' ayat 103 yang berbunyi :

فَإِذَا قَضَيْتُمُ ٱلصَّلَوٰةَ فَٱذْكُرُوا۟ ٱللَّهَ قِيَـٰمًا وَقُعُودًا وَعَلَىٰ جُنُوبِكُمْ ۚ فَإِذَا
ٱطْمَأْنَنتُمْ فَأَقِيمُوا۟ ٱلصَّلَوٰةَ ۚ إِنَّ ٱلصَّلَوٰةَ كَانَتْ عَلَى ٱلْمُؤْمِنِينَ كِتَـٰبًا مَّوْقُوتًا

Artinya "*Maka apabila kamu telah menyelesaikan shalat(mu), ingatlah Allah di waktu berdiri, di waktu duduk dan di waktu berbaring. Kemudian apabila kamu*

telah merasa aman, maka dirikanlah shalat itu (sebagaimana biasa). Sesungguhnya shalat itu adalah fardhu yang ditentukan waktunya atas orang-orang yang beriman." (Qs An-Nisa' 4:103).

Pada akhir ayat tersebut yang berarti " … *Sesungguhnya shalat itu adalah fardhu yang ditentukan waktunya atas orang-orang yang beriman",* dimana menurut tafsir ibnu katsir bahwa, Dia berkata pula "sesungguhnya sholat memiliki waktu seperti waktu haji". Sedangkan menurut tafsir jalalin memiliki makna bahwasannya waktu sholat sudah ditetapkan, maka ketika sudah tiba waktu menjalakanya jangan diundur undur atau ditangguhkan. Wallahua'lam bisshowab

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Dari hasil implementasi uji coba yang telah peneliti lakukan diperoleh kesimpulan sebagai berikut :

1. metode Depth First Search (DFS) dapat diterapkan dalam aplikasi visualisasi pergerakan kendaraan. Hal ini dapat diamati dari hasil output yang diperoleh yaitu berupa jalur atau rute perjalanan masing masing kendaraan, mulai dari asal dan tujuan yang masing masing berbeda, pemilihan cabang berdasarkan tujuan kendaraan, waktu yang ditempuh untuk pergerakan kendaraan. Semua sudah sesuai dengan rancangan program yang telah dibuat. Sehingga dapat disimpulkan bahwa metode Depth First Search (DFS) mampu diterapkan pada visualisasi pergerakan kendaraan.
2. Akurasi yang diperoleh dari implementasi metode Depth Fisrt Search berdasarkan parameter yang telah peneliti tetapkan sebelumnya dan untuk ketepatan pergerakan kendaraan dengan titik tujuan terdekat adalah 100%.

Pada implementasinya, meskipun DFS mampu untuk diterapkan pada aplikasi visualisasi pergerakan kendaraan masih memiliki kekurangan yaitu pergerakan kendaraan terkesan lambat, karena banyaknya jumlah kendaraan yang digerakkan secara bersamaan, hal ini disinyalir karena setiap mobil memiliki proses pencarian

rute menggunakan DFS dan selain itu banyaknya proses perulangan pada saat menggerakkan kendaraan. Meskipun demikian, setiap mobil dapat dirubah asal tujuannya berdasarkan aturan yang telah ditetapkan dan otomatis DFS akan menemukan rute perjalanan berdasarkan asal tujuan baru yang diberikan.

5.2 Saran

Dalam penelitian ini masih terdapat banyak sekali kekurangan kekurangan, untuk pengembangan lebih lanjut terdapat beberapa saran dari peneliti untuk penelitian selanjutnya. Adapun saran-saran tersebut adalah sebagai berkut :

1. Terdapat banyak proses perulangan dalam melakukan pergerakan, mungkin penelitian selanjutnya dapat menambahkan metode untuk meminimalis proses *looping* sehingga pergerakan kendaraan lebih cepat.
2. Penelitian selanjutnya dapat menampilkan kondisi jalan dengan sebenarnya seperti lampu merah, atau ilustrasi sedang ada galian lubang sehingga pada ruas jalan tertentu tidak dapat dilewati kendaraan, dan lain sebagainya.
3. Pada penelitian selanjutnya setiap node bisa diberikan bobot untuk memudahkan proses menghitung akurasi

DAFTAR PUSTAKA

Adisasmita S.A. 2012, *Perancangan Infrastruktur Transportasi Wilayah,* Penerbit :Graha Ilmu. Yogyakarta

Anany, 2010, *Pengantar Desain dan Analisis Algoritma*. Selemba Infotek. Jakarta

Harnaningrum, 2010, *Strukrur Data Menggunakan java*, Graha Ilmu, Yogyakarta

Kusumadewi, 2003, *Artificial Intelegence (teknik dan aplikasinya),* Penerbit *:* Graha Ilmu Yogyakarta

Khisty *et.al*, 2005, *Dasar-dasar rekayasa lalulintas, jilid 1,* penerbit : Erlangga, Ciracas Jakarta

Miro, Fidel. 2005. *Perencanaan Transportasi Untuk Mahasiswa, Perencana, dan Praktisi. Jakarta*: Penerbit Erlangga

Miro, Fidel. 1997. *Sistem Transportasi Kota. Bandung*: Tarsito

Munir, 2005, *Matematika Diskrit*, Informataika, Bandung

Munir, 2011, *Penerapan BFS dan DFS Pada Pencarian Solusi*. ITB. Bandung

Putra, 2010, *Pengolahan Citra Digital*, Penerbit : Andi Yogyakarta

Putri S.E, 2011, *Implementasi dan Analisa Algoritma Depth First Search (DFS) dalam pencarian Lintasan terpanjang*, Sumatera Utara

Tarecha *et. al*, 2013, *Visualisasi 3D Rupa Bumi Berbasis Data GDEM ASTER 30 meter*, Surabaya : Seminar Nasional Matematika dan Aplikasinya

Tyas, 2010, *Implementasikan Algoritma Backtracking dengan menggunakan metode DFS (Depth First Search) pada penyelesaian traveling salesman problem*.

Suyanto, *Artificial Intelegence edisi revisi,* Penerbit : Informatika Bandung, 2011

Undang undang Republik Indonesia Nomor 14 tahun 1992 tentang perbankan, lembaran Negara No.182

Badan Pusat Statistika Indonesi (BPSI) 2014

Data kemacetan di kota malang – Pemkab Malang

Kantor Kepolisisan Republik Indonesia – Jawa Timur

Transpotasi 2014 (www.wikipedia.com)

E-book Tafsir Al-Qur'an Ibnu Katsir 30 juz (www.daarus-sunnah.com)

Kendaraan bermotor 2014 (www.wikipwdia.com)

Lalu lintas Kota malang . Radar malang (www.radarmalang.co.id)

www.news.bisnis.com/read/20141012/78/264206/kota-malang-volume-kendaraan-sudah-lampaui-kapasitas-jalan -

www.mathworks.com/matlabcentral/fileexchange/24134-gaimc---graph-algorithms-in-matlab-code/content//gaimc/test/test_dfs.m

https://jalandakwahbersama.wordpress.com/2009/07/17/mudahkanlah-urusan-orang-lain/