JCENIM

ISSN 2986-1764
Volume 2, Nomor 2, Juli 2024

Pengembangan Simulasi Lalu Lintas Menggunakan Graf Berbobot Waktu

Eko Pramunanto[1], Bagus Himawan[1],

[1]Dept. Teknik Komputer Institut Teknologi Sepuluh Nopember Surabaya, Indonesia

Email: ekopram@te.its.ac.id, bagus.himawan14@mhs.te.its.ac.id

Abstrak

Setiap tahun jumlah kendaraan di jalan mengalami peningkatan, oleh karena itu jumlah kemacetan lalu lintas juga meningkat. Harga kendaraan yang semakin murah serta skema pembayaran yang lebih terjangkau bagi konsumen membuat masyarakat semakin mudah untuk memiliki mobil. Dengan meningkatnya jumlah kendaraan di jalan, kemacetan lalu lintas menjadi rutinitas yang normal bagi masyarakat terutama di kota besar. Kemacetan lalu lintas berarti waktu tempuh menuju tujuan menjadi lebih lama dan polusi semakin meningkat. Sementara kapasitas jalan tidak mampu mengimbangi pertumbuhan jumlah kendaraan di jalan, maka diperlukan solusi lain. Solusi yang paling mudah adalah dengan mengatur ulang aturan lalu lintas sesuai dengan kondisi lalu lintas. Untuk mengetahui aturan terbaik, alat simulasi dapat sangat membantu. Dengan menggunakan simulasi, aturan lalu lintas baru dapat diuji terlebih dahulu pada simulasi sebelum diterapkan di dunia nyata. Dengan manajemen jalan yang lebih baik, hal ini akan membantu mengurangi kemacetan lalu lintas dan mengurangi polusi karena masyarakat dapat mencapai tujuan mereka dengan lebih cepat.

Keyword: Sistem Lalu Lintas Cerdas, Simulasi Lalu Lintas, Simulasi, Teori Graf, Windows, Unity Engine.

Diterima Redaksi: 05-07-2024 Selesai Revisi: 15-07-2024 Diterbitkan Online: 15-07-2024
DOI: https://doi.org/10.59378/jcenim.v2i2.53

I. PENDAHULUAN

Simulasi umumnya digunakan untuk memvisualisasikan suatu sistem atau proses dengan model statistik atau aturan tertentu [1]. Dalam simulasi lalu lintas, sistem harus mampu memvisualisasikan setiap aspek penting dari lalu lintas, seperti kendaraan, persimpangan, lampu lalu lintas, serta aturan lalu lintas di dalam simulasi [2].

Simulasi lalu lintas yang baik harus mampu memvisualisasikan situasi yang kompleks sehingga dapat digunakan untuk berbagai keperluan, misalnya untuk memvisualisasikan pengalihan jalan, atau untuk memvisualisasikan kondisi lalu lintas setelah aturan tertentu diterapkan pada sistem, dan berbagai keperluan lainnya [3].

Menurut GAIKINDO (Gabungan Industri Kendaraan Bermotor Indonesia), jumlah mobil di jalan pada Agustus 2017 meningkat sebanyak 800.000 unit dibandingkan dengan tahun sebelumnya [3]. Dengan jumlah kendaraan yang sangat besar tersebut sementara kapasitas jalan tidak mampu mengimbangi peningkatan jumlah kendaraan, hal ini dapat menimbulkan permasalahan yang lebih besar, seperti kemacetan lalu lintas [1]. Salah satu cara selain menambah atau membangun jalan baru untuk menyesuaikan jumlah kendaraan adalah dengan membuat aturan lalu lintas tertentu yang mampu mengatur arus lalu lintas dan mengurangi kecelakaan, sehingga dengan bantuan simulasi lalu lintas hal ini dapat dicapai dengan lebih mudah[4] [3].

Untuk membuat simulasi lalu lintas yang benar-benar bermanfaat, simulasi tersebut perlu memenuhi beberapa persyaratan. Simulasi harus mampu memvisualisasikan lebih dari sekadar satu persimpangan dengan sistem lampu lalu lintasnya, harus mampu menangani jaringan jalan dalam skala besar, mampu memvisualisasikan kemacetan lalu lintas sehingga pengguna dapat melihat seberapa besar pengaruh aturan lalu lintas terhadap respons lalu lintas, serta mampu mengubah aturan lalu lintas tertentu, seperti batas kecepatan maksimum atau durasi lampu lalu lintas. Dengan seluruh kemampuan tersebut, simulasi dapat digunakan untuk menciptakan aturan lalu lintas yang lebih baik [2].

Untuk mengetahui apakah modifikasi aturan lalu lintas memengaruhi kondisi lalu lintas, simulasi memerlukan suatu metode pengukuran. Pada Google Maps, jalan dengan lalu lintas padat ditampilkan

dengan warna merah, sedangkan lalu lintas ringan ditampilkan dengan warna hijau. Metode pengukuran seperti ini sangat berguna untuk menentukan kondisi lalu lintas [1]. Untuk mencapai hal tersebut, jaringan jalan harus dibangun menggunakan graf dengan sisi berbobot.

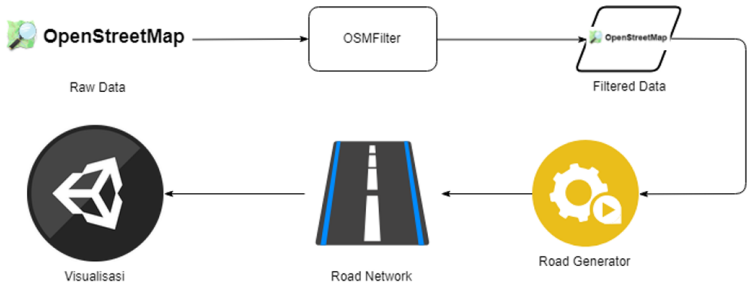
Pada umumnya, graf berbobot menggunakan jarak sebagai bobotnya. Namun, untuk simulasi lalu lintas, bobot yang menggunakan nilai waktu lebih sesuai, karena dalam kondisi nyata, jarak terdekat bisa saja membutuhkan waktu tempuh yang paling lama. Oleh karena itu, rute yang lebih cepat lebih disukai dibandingkan rute terdekat [3].

Jenis simulasi seperti ini sangat berguna untuk keperluan publik maupun penelitian, khususnya pada kota dengan jumlah kendaraan, jalan, dan populasi yang besar [1].

II. JARINGAN JALAN DENGAN GRAF BERBOBOT WAKTU

Tinjauan terhadap penelitian praktis dan teoretis yang relevan disajikan dalam bagian ini [1]. Contoh kueri juga ditampilkan. Selain itu, bagian ini secara singkat membahas penggunaan basis data peta terbuka OpenStreetMap dalam aplikasi semacam ini. Terakhir, pemodelan jaringan jalan menggunakan graf berbobot waktu sebagai penelitian terkait akan dijelaskan dalam bagian ini.

Subbagian ini membahas cara memodelkan jaringan jalan menggunakan graf berbobot waktu. Berbeda dengan graf biasa yang menggunakan jarak, graf ini menggunakan waktu sebagai bobotnya [3]. Permasalahan utama adalah bagaimana cara menghitung bobot tersebut. Menurut penelitian terkait [1], dengan menggunakan kecepatan dan panjang sisi, nilai keterlambatan atau bobot dapat dihitung. Perbedaan utama antara makalah ini dan penelitian terkait [1] adalah bahwa makalah ini menggunakan OpenStreetMap untuk memperoleh data jalan.



Gambar 1: Tahapan untuk menghasilkan jaringan jalan

Setelah jaringan jalan dihasilkan dan setiap simpul jalan saling terhubung menggunakan apa yang biasa disebut sebagai sisi, maka setelah sisi siap digunakan, bobot awal dari sisi dapat dihitung. Dengan menggunakan dua variabel pada sisi tersebut, yaitu panjang sisi dan batas kecepatan, sesuai dengan [1], hal ini sudah cukup untuk menghitung bobot sisi [3].

Bobot awal digunakan untuk menghitung berapa lama waktu yang dibutuhkan untuk melewati suatu sisi. Berdasarkan Persamaan 1, diperlukan dua parameter, yaitu L sebagai panjang sisi dan Vm sebagai batas kecepatan maksimum.

$$T_i = \frac{L}{V_m} \tag{1}$$

Untuk menghitung keterlambatan pada suatu sisi, perlu dihitung waktu rata-rata kendaraan saat bergerak pada sisi tersebut. Hal ini dapat dicapai dengan menggunakan Persamaan 2. L tetap merupakan panjang sisi, namun alih-alih menggunakan batas kecepatan, Va merupakan kecepatan rata-rata seluruh kendaraan pada sisi tersebut [1].

$$T_m = \frac{L}{V_a} \tag{2}$$

Untuk memperoleh nilai keterlambatan dari sisi tersebut, langkah selanjutnya adalah menghitung deviasi antara Ti sebagai bobot optimal atau awal dan Tm sebagai waktu yang dibutuhkan untuk melewati sisi berdasarkan kecepatan rata-rata saat ini, sebagaimana ditunjukkan pada Persamaan 3 [1].

$$T = T_{initial} - T_{movingVehicle} \tag{3}$$

III. PEMODELAN JARINGAN JALAN

A. Ekstraksi Data

Proses ini bertujuan untuk menentukan area di dunia yang akan divisualisasikan di dalam simulasi. Pada makalah ini, data akan diekstraksi dari OpenStreetMap. OpenStreetMap merupakan peta berbasis OpenData yang dibangun oleh komunitas yang berkontribusi dan memelihara data mengenai jalan, jalur, tempat, sungai, dan banyak lagi. Kontributor peta menggunakan berbagai teknologi mulai dari perangkat kelas atas hingga perangkat kelas bawah untuk melakukan pemetaan dan menyimpan data ke dalam OpenStreetMap.

Karena sifatnya yang berbasis OpenData, visualisasi suatu area menjadi jauh lebih mudah karena seluruh detail data yang tersedia dapat diperoleh, dan proses perbaikan data juga menjadi lebih mudah. Data yang telah dikoreksi dapat diunggah kembali ke OpenStreetMap dan selanjutnya akan ditinjau oleh kontributor lain, sehingga apabila data yang dikoreksi ternyata tidak tepat, perubahan tersebut dapat dikembalikan.

Gambar 2: Pemilihan area

Langkah pertama yang dilakukan adalah memilih area yang akan diekstraksi dari OpenStreetMap. Proses ini sangat mudah dilakukan karena OpenStreetMap telah menyediakan fitur tersebut dan dapat digunakan hanya dengan satu kali klik. Gambar 2 menunjukkan area yang akan divisualisasikan pada makalah ini.

B. Penyaringan Data

Data yang diekspor langsung dari situs OpenStreetMap memiliki ukuran yang sangat besar. OpenStreetMap tidak hanya mengekspor data jalan, tetapi juga mengekstraksi data bangunan, jalur, dan sungai dalam proses tersebut. Sementara itu, simulasi hanya membutuhkan data jalan untuk divisualisasikan. Untungnya, terdapat sebuah alat bernama OSMFilter yang dapat digunakan untuk menyaring data OSM yang telah diekstraksi sebelumnya.

```
osmfilter.exe Sidoarjo.osm
--keep="highway=motorway=trunk=primary=secondary=tertiary
=unclassified=motorway_link=trunk_link=primary_link
=secondary_link=traffic_signals and visible=true"
--keep-relations="restriction"
> SidoarjoWays.osm
```

Menjalankan OSMFilter dengan perintah di atas akan menghasilkan data OSM yang telah disaring sehingga ukurannya lebih kecil dan hanya berisi data jalan. Data lain seperti bangunan, sungai, atau

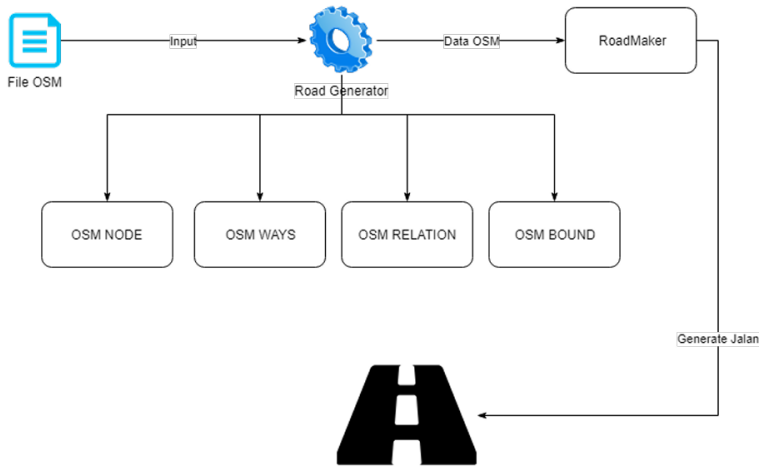
jalur pejalan kaki akan dihapus dalam proses ini.

C. Pembuatan Model Jalan

Pada proses ini, simulasi akan menghasilkan model jalan berdasarkan data dari OpenStreetMap. Untuk mencapai hal tersebut, Road Generator akan menyimpan seluruh data berdasarkan jenisnya. Terdapat empat jenis data yang berbeda, yaitu:

1. **OSMNode**: Node merupakan salah satu elemen inti dalam model data OpenStreetMap. Node terdiri dari satu titik di ruang yang didefinisikan oleh lintang, bujur, dan ID node.
2. **OSMWays**: Way merupakan daftar node yang tersusun secara berurutan dan biasanya memiliki setidaknya satu tag atau termasuk dalam suatu Relation. Sebuah way dapat memiliki antara 2 hingga 2.000 node, meskipun dimungkinkan terdapat way yang bermasalah dengan nol atau satu node. Way dapat bersifat terbuka atau tertutup. Way tertutup adalah way yang node terakhirnya juga merupakan node pertama. Way tertutup dapat diinterpretasikan sebagai polyline tertutup, area, atau keduanya.
3. **OSMRelation**: Relation merupakan salah satu elemen data inti yang terdiri dari satu atau lebih tag serta daftar node, way, dan/atau relation lain yang tersusun secara berurutan sebagai anggota. Relation digunakan untuk mendefinisikan hubungan logis atau geografis antar elemen lainnya. Setiap anggota relation dapat memiliki peran yang menjelaskan fungsi suatu fitur tertentu di dalam relation tersebut.
4. **OSMBound**: Data yang berisi nilai lintang dan bujur maksimum serta minimum. Data ini sangat penting karena berfungsi sebagai batas koordinat lintang dan bujur.

OSMNode perlu dikonversi dari format lintang dan bujur ke format vector3 yang didukung oleh Unity. Dengan menggunakan proyeksi Mercator, proses ini dapat dilakukan dengan mudah.



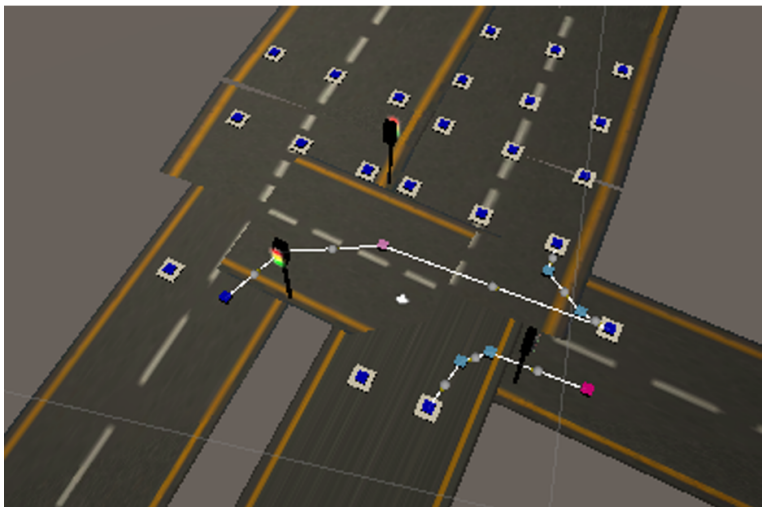
Gambar 3: Visualisasi pembuatan jalan

Road Maker akan menghasilkan model jalan menggunakan data yang tersedia. Gambar 3 menjelaskan bagaimana Road Generator membaca data dan Road Maker menggunakan data tersebut untuk menghasilkan model jalan.

Pada tahap ini, simulasi sudah mampu membaca data dari OpenStreetMap ke dalam dunia virtual dan hampir siap digunakan untuk simulasi lalu lintas.

D. Koreksi Manual

Data yang dihasilkan secara otomatis sering kali mengandung kesalahan. Hal ini merupakan hal yang wajar karena OpenStreetMap dibangun oleh komunitas sehingga terkadang data yang tersedia tidak sempurna atau terdapat informasi yang hilang. Oleh karena itu, untuk menghasilkan simulasi yang lebih realistis, beberapa aturan lalu lintas atau fitur perlu ditambahkan secara manual.



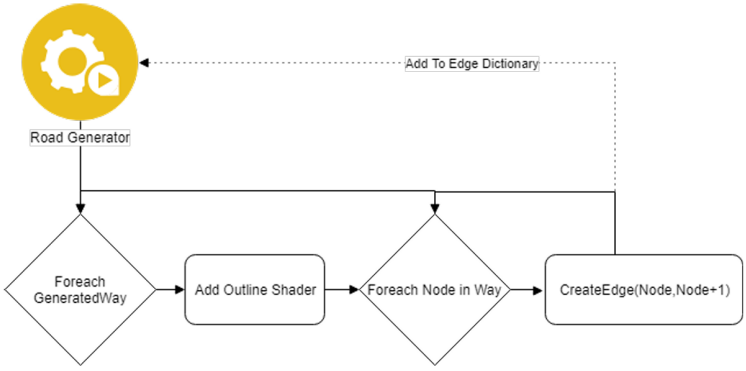
Gambar 4: Koreksi persimpangan

Sebagian besar data yang disediakan oleh OpenStreetMap untuk memvisualisasikan persimpangan secara realistis masih jauh dari ideal. Oleh karena itu, hampir setiap persimpangan perlu dikoreksi secara manual, seperti apakah persimpangan tersebut memiliki lampu lalu lintas, atau ke arah mana kendaraan boleh atau tidak boleh berbelok, dan sebagainya. Gambar 4 menunjukkan sebuah persimpangan setelah dilakukan koreksi secara manual.

IV. IMPLEMENTASI

Untuk membuat simulasi sebagai sebuah alat untuk menyelesaikan suatu permasalahan, simulasi tersebut perlu dilengkapi dengan fitur-fitur yang dapat diubah oleh pengguna. Sebagai contoh, pengguna harus dapat menonaktifkan lampu lalu lintas, menutup jalan, dan sebagainya. Pengguna juga harus diberikan informasi yang cukup mengenai apa yang terjadi pada lalu lintas. Pada bagian ini akan dijelaskan bagaimana jaringan graf jalan akhirnya dibentuk, serta ditampilkan desain antarmuka pengguna yang memungkinkan pengguna untuk mengakses beberapa konfigurasi.

A. Pembentuk Edge



Gambar 5: Pembentuk Edge

Untuk membentuk graf berbobot yang dapat berfungsi, program perlu mengetahui edge mana yang menghubungkan dua node. Untuk mencapai hal tersebut, edge perlu dihasilkan secara otomatis. Dari Gambar 5, seluruh OSM Way yang dihasilkan perlu diproses. Road Generator akan melakukan iterasi pada seluruh way yang dihasilkan dan menambahkan sebuah indikator ke dalamnya. Indikator ini akan digunakan untuk menunjukkan kondisi lalu lintas dari warna hijau hingga merah. Selanjutnya, Road Generator akan memproses setiap node lalu lintas yang saling terhubung. Seluruh node yang telah

diproses akan ditambahkan ke dalam dictionary edge dengan beberapa kunci, yaitu node asal dan node tujuan.

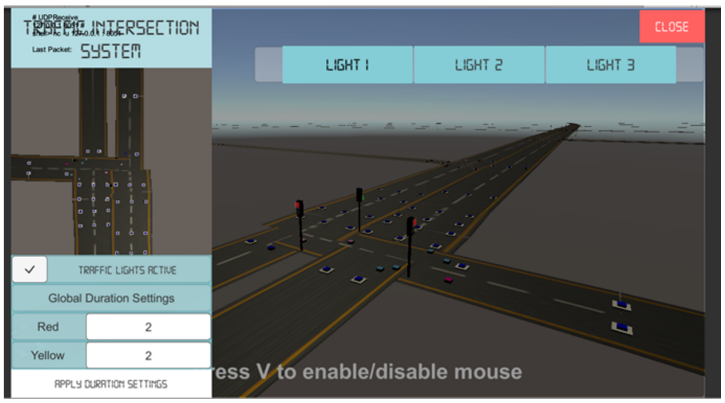
B. Penambahan Kendaraan

Gambar 6: Berbagai Model Kendaraan

Untuk membuat simulasi menjadi lebih realistis, simulasi lalu lintas harus dilengkapi dengan lebih dari satu kendaraan. Pada Gambar 6 ditampilkan beberapa kendaraan yang digunakan sebagai contoh. Setiap kendaraan memiliki fitur yang berbeda sehingga dapat mensimulasikan skenario dunia nyata dan membuat simulasi lalu lintas menjadi lebih baik dibandingkan sebelumnya.

C. Pengaturan Lalu Lintas

Untuk memungkinkan pengguna mengubah berbagai pengaturan pada simulasi lalu lintas, perlu dirancang sebuah antarmuka pengguna sehingga pengguna dapat mengubah aturan lalu lintas secara langsung.

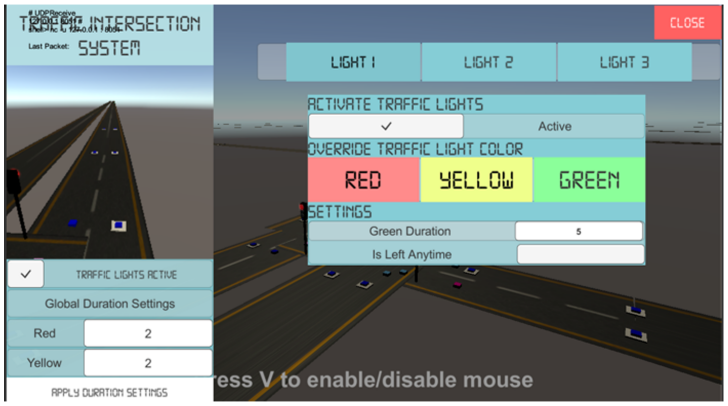


Gambar 7: Pengaturan Persimpangan

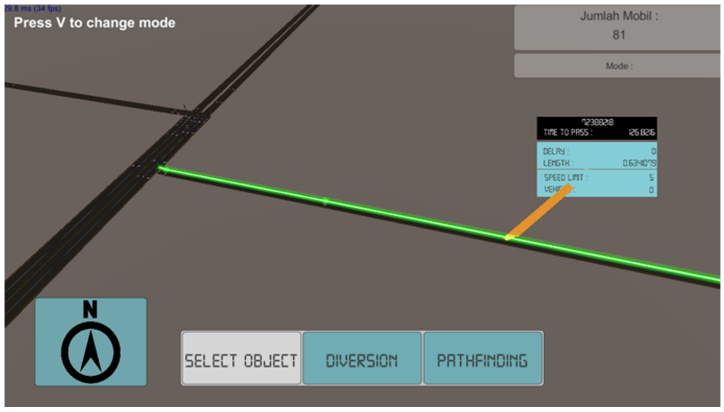
Pada Gambar 7, pengguna dapat mengakses pengaturan utama persimpangan. Melalui menu ini, pengguna dapat mengubah durasi lampu merah dan lampu kuning, serta menonaktifkan lampu lalu lintas.

Ketika pengguna memilih atau mengklik opsi Light pada Gambar 7, pengaturan lampu lalu lintas akan ditampilkan kepada pengguna. Gambar di sisi antarmuka juga akan berubah sesuai dengan bagian persimpangan yang dipilih. Melalui menu ini, pengguna dapat memaksa lampu lalu lintas ke warna tertentu, mengubah durasi lampu hijau, serta memaksa kendaraan untuk selalu dapat belok kiri seperti yang ditunjukkan pada Gambar 8.

Antarmuka pop-up yang ditunjukkan pada Gambar 9 akan muncul ketika pengguna mengarahkan kursor mouse ke suatu ruas jalan tertentu. Antarmuka ini akan memberikan informasi kepada pengguna mengenai jalan tersebut, seperti batas kecepatan, waktu tunda, dan informasi lainnya.

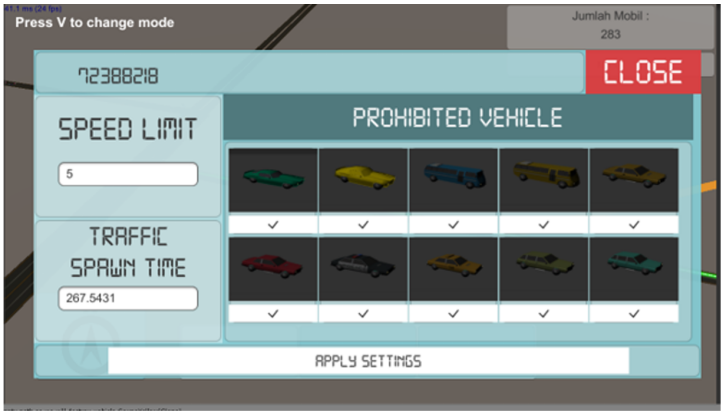


Gambar 8: Pengaturan Lampu Lalu Lintas



Gambar 9: Antarmuka Penyorotan Jalan

Ketika pengguna mengklik sebuah jalan saat Gambar 9 ditampilkan, maka pengaturan jalan seperti yang ditunjukkan pada Gambar 10 akan ditampilkan kepada pengguna. Melalui menu tersebut, pengguna dapat mengatur batas kecepatan, membatasi jenis kendaraan yang diperbolehkan melewati jalan tersebut, serta mengubah waktu kemunculan lalu lintas.



Gambar 10: Pengaturan Jalan

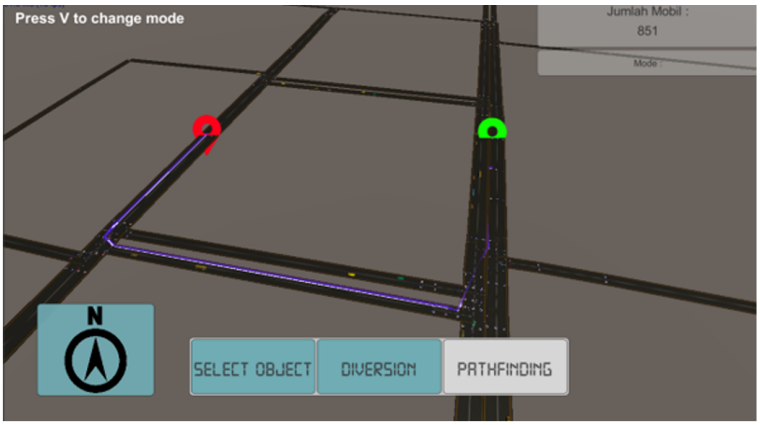
D. Penutupan Jalan

Pengguna dapat menutup sebuah edge, yang berarti tidak ada satu pun kendaraan yang dapat melewati jalan tersebut. Untuk mencapai hal ini, tersedia mode penempatan penutupan jalan, di mana pengguna dapat menempatkan penutup jalan di antara dua node lalu lintas. Model penutupan jalan pada Gambar

Gambar 11: Jalan yang Ditutup

11 sebenarnya hanya digunakan untuk visualisasi, sedangkan fungsi sebenarnya adalah mengubah suatu nilai pada edge yang bersangkutan sehingga jalan tersebut tidak dapat dilewati oleh kendaraan lain.

E. Pencarian Jalur



Gambar 12: Mode Pencarian Jalur

Pencarian jalur merupakan salah satu fitur utama yang dimungkinkan oleh penggunaan graf berbobot. Dengan graf berbobot waktu, pencarian jalur dapat menemukan rute tercepat menuju tujuan, bukan berdasarkan jarak terdekat. Algoritma pencarian jalur yang digunakan merupakan A* yang dimodifikasi dengan metode heuristik khusus [1].

Algoritma pencarian jalur didefinisikan oleh Persamaan 4:

$$F = H + G + T \tag{4}$$

Dengan keterangan:

- F = Metode Heuristik
- H = Jarak antara dua node
- G = Jarak yang telah ditempuh
- T = Waktu Tunda Rata-rata

Untuk menggunakan mode pencarian jalur, pengguna menentukan dua node yang berbeda sebagai posisi awal dan tujuan. Posisi awal akan berfungsi sebagai titik awal, sedangkan tujuan merupakan titik akhir. Pertama, pengguna memilih posisi awal yang ditandai dengan warna hijau. Setelah posisi awal dipilih, pengguna memilih tujuan dengan penanda berwarna merah. Setelah pengguna menempatkan

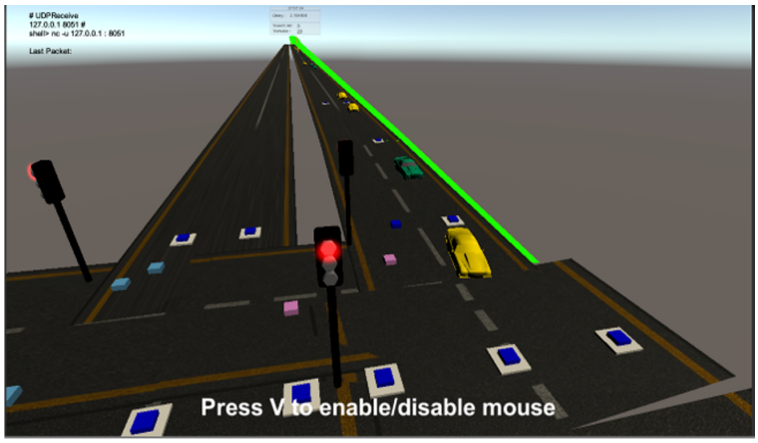
posisi tujuan, sistem pencarian jalur akan secara otomatis mencari jalur tercepat dari posisi awal menuju tujuan seperti yang ditunjukkan pada Gambar 12.

V. PENGUJIAN

Terdapat 3 bagian pengujian, yang pertama adalah pengujian skenario, pada pengujian ini sistem lalu lintas akan diberikan beberapa skenario dan aturan lalu lintas akan diubah untuk melihat pengaruh dari aturan baru tersebut. Bagian kedua adalah pengujian pencarian jalur, pengujian ini dilakukan dalam dua tahap, tahap pertama dilakukan ketika jalan masih kosong tanpa lalu lintas, dan tahap kedua dilakukan pada jalan dengan lalu lintas padat. Kedua pengujian tersebut memiliki posisi awal dan tujuan yang sama.

A. Pengujian Skenario

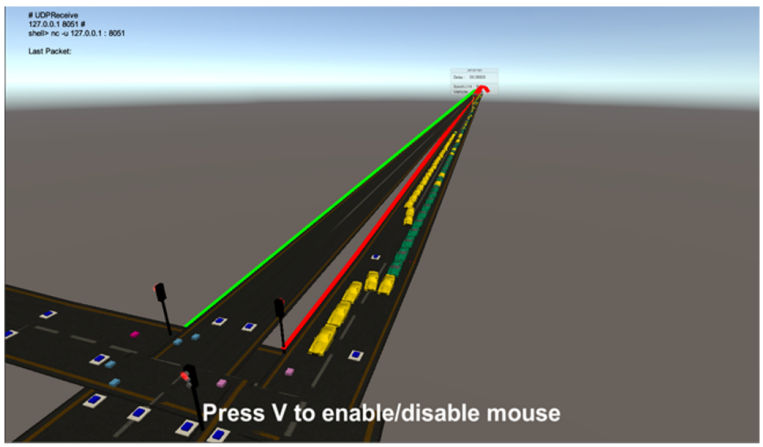
Bagian ini bertujuan untuk mengetahui apakah visualisasi dan antarmuka pengguna telah bekerja sesuai dengan yang diharapkan. Langkah pertama adalah simulasi lalu lintas padat:



Gambar 13: Kondisi Tanpa Lalu Lintas

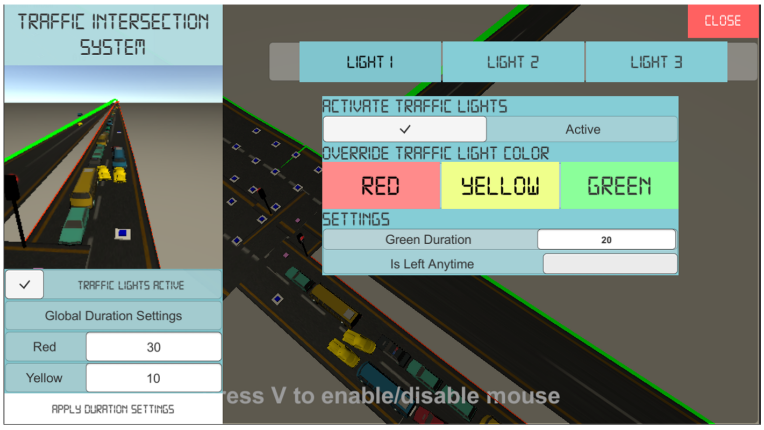
Pada awal simulasi seperti yang terlihat pada Gambar 13, visualisasi lalu lintas berwarna hijau karena kondisi lalu lintas yang rendah.

Beberapa waktu kemudian pada Gambar 14, ketika lalu lintas menjadi padat, warna visualisasi berubah menjadi merah, sama seperti pada Google Maps ketika lalu lintas sedang padat. Hal ini berarti visualisasi lalu lintas bekerja dengan baik. Pengujian selanjutnya adalah mengurai kemacetan dengan mengubah aturan lampu lalu lintas.

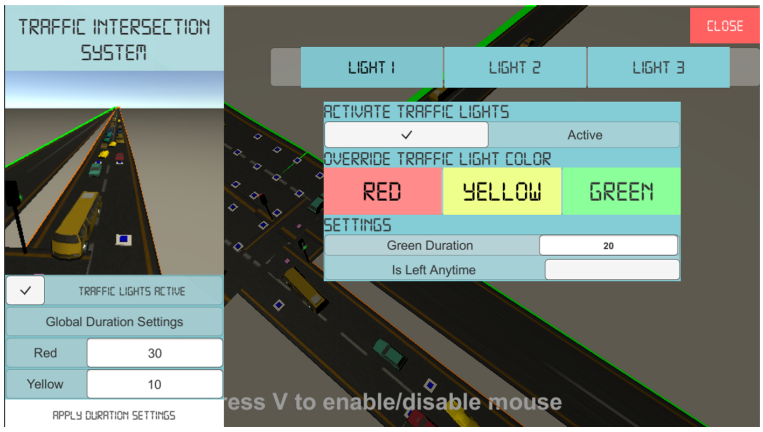


Gambar 14: Kondisi Lalu Lintas Padat

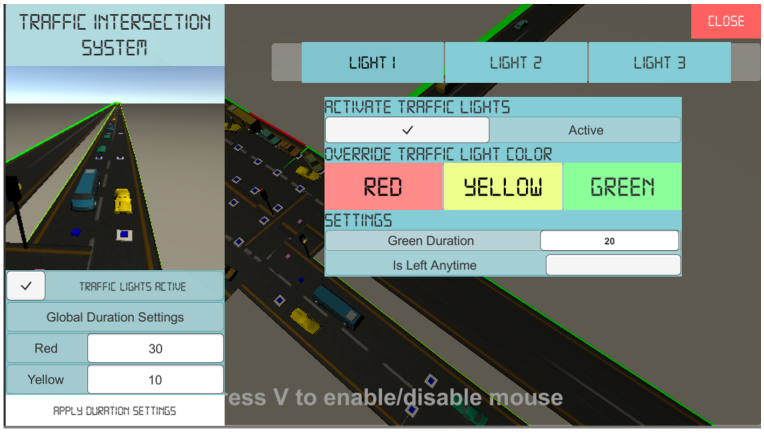
Pada Gambar 15, kondisi lalu lintas sangat buruk dan warna visualisasi berubah menjadi merah, yang berarti terdapat beban lalu lintas yang tinggi pada jalan tersebut. Cara tercepat untuk membuat lalu lintas kembali bergerak adalah dengan melakukan override pada sistem lampu lalu lintas, sehingga lampu lalu lintas pada jalan yang bersangkutan diubah menjadi hijau.



Gambar 15: Sebelum Override Lampu Lalu Lintas



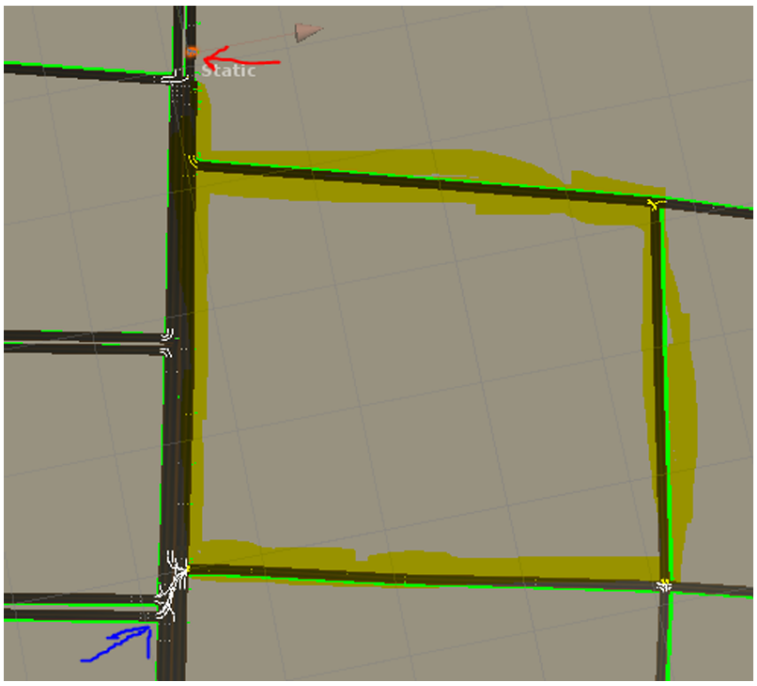
Gambar 16: Sesaat Setelah Override Lampu Lalu Lintas



Gambar 17: Setelah Override Lampu Lalu Lintas

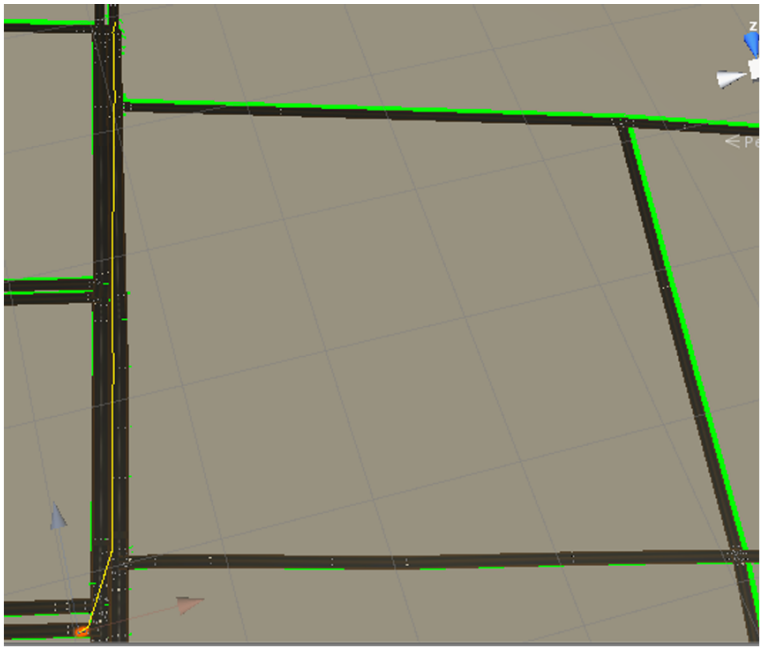
Setelah lampu lalu lintas di-override, lalu lintas mulai bergerak seperti yang terlihat pada Gambar 16. Ketika lalu lintas mulai bergerak kembali, warna visualisasi mulai berubah dari merah ke kuning, dan pada Gambar 17 lalu lintas kembali mengalir dan warna visualisasi berubah menjadi hijau.

B. Pengujian Pencarian Jalur



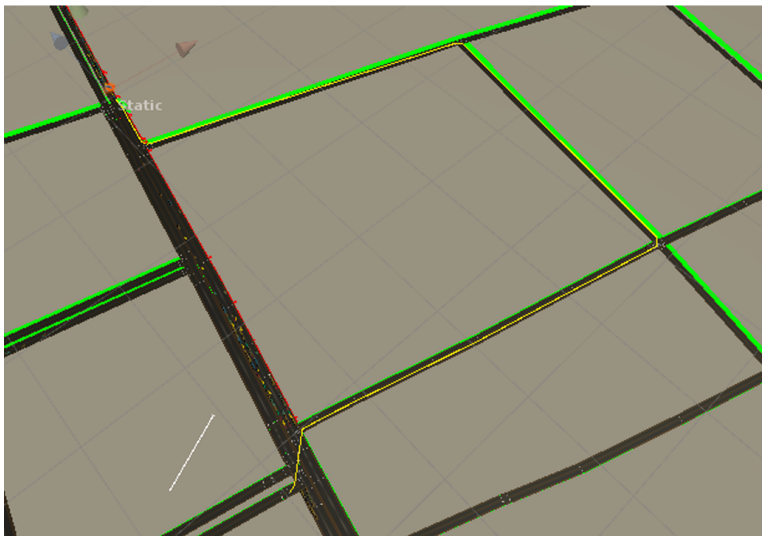
Gambar 18: Posisi Awal dan Tujuan

Pada Gambar 18, panah merah menandai posisi awal, sedangkan panah biru menandai posisi akhir. Pencarian jalur akan mencari jalur tercepat dalam dua kondisi, yaitu pertama ketika lalu lintas sangat ringan, dan kedua ketika lalu lintas sangat padat.



Gambar 19: Pencarian Jalur pada Lalu Lintas Ringan

Pada Gambar 19, garis kecil berwarna kuning merupakan hasil pencarian jalur pada kondisi lalu lintas sangat ringan. Jalur tersebut langsung menuju dari posisi merah ke posisi biru. Namun pada percobaan kedua seperti yang terlihat pada Gambar 20, jalur yang dihasilkan berbeda. Peningkatan lalu lintas pada jalur sebelumnya menyebabkan algoritma pencarian jalur menghasilkan rute yang berbeda.



Gambar 20: Pencarian Jalur pada Lalu Lintas Padat

C. Pengujian Performa

Bagian pertama dari pengujian ini adalah pengujian beban GPU. Proses rendering pada simulasi lalu lintas dengan jumlah kendaraan yang banyak membutuhkan sumber daya yang sangat besar, terutama pada GPU. Untuk menguji seberapa besar pengaruh GPU terhadap simulasi, simulasi dijalankan pada mesin yang sama tetapi dengan GPU yang berbeda.

Tabel 1: Spesifikasi PC

Komponen	**Perangkat A**	**Perangkat B**
OS	Windows 10 64 bit	
CPU	Intel Core i7 3630QM 2.4GHz - 3.2GHz (Boost)	
GPU	Nvidia Geforce GT650m	Intel HD 4000
RAM	8GB	

Tabel 2: Hasil Pengujian Performa GPU

Jumlah Kendaraan	**Geforce GT 650m (FPS)**	**Intel HD 4000 (FPS)**
0	40	12
50	30	11
100	26	11
150	23	11
200	22	11
250	20	11
300	20	10
350	17	10
400	16	10
450	15	10
500	14	10

Tabel 2 menunjukkan hasil pengujian performa GPU, di mana baik GPU maupun CPU mempengaruhi performa simulasi.

Bagian kedua dari pengujian ini bertujuan untuk mengetahui bagaimana perbedaan prosesor mempengaruhi performa simulasi. Sama seperti pengujian GPU, pengujian ini dijalankan pada mesin yang sama tetapi dengan spesifikasi CPU yang berbeda.

Tabel 3: Spesifikasi CPU

Komponen	Perangkat A	Perangkat B
OS	Windows 10 64 bit	
CPU	Intel Core i7 3630QM 2.4GHz - 3.2GHz (Boost)	1.19GHz
GPU	Nvidia Geforce GT650m	
RAM	8GB	

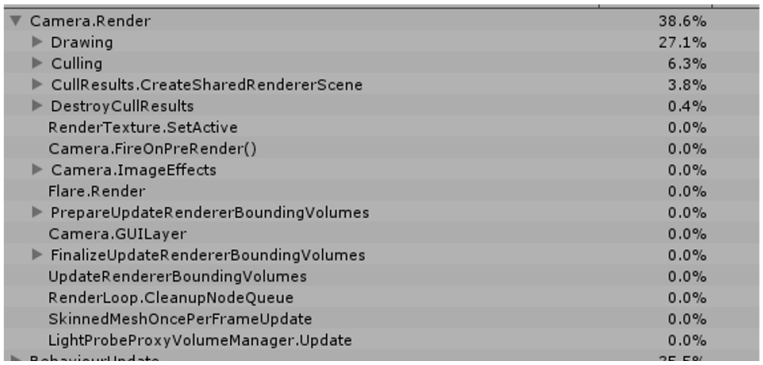
Tabel 4: Hasil Pengujian Performa CPU

Jumlah Kendaraan	2.4 - 3.2 GHz (FPS)	1.19 GHz (FPS)
0	40	22
50	30	18
100	26	17
150	23	9
200	22	7
250	20	6
300	20	5
350	17	5
400	16	4
450	15	3
500	14	3

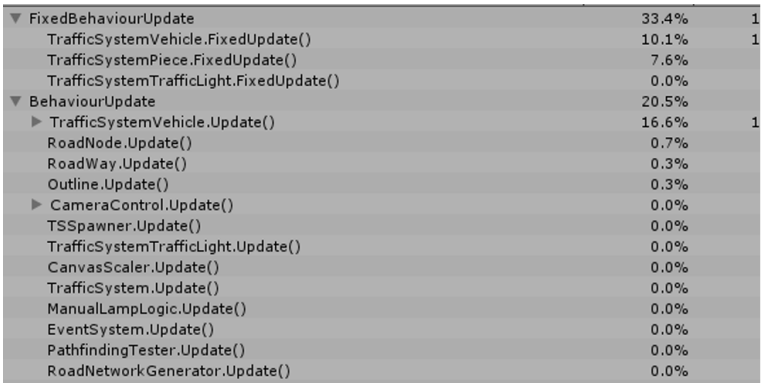
Tabel 4 menyajikan hasil pengujian performa CPU yang menunjukkan variasi performa yang signifikan berdasarkan kecepatan CPU.

Berdasarkan kedua pengujian GPU dan CPU tersebut, dapat disimpulkan bahwa baik GPU maupun CPU mempengaruhi performa simulasi. Namun, berdasarkan pengujian ini masih belum cukup untuk menentukan proses apa yang menyebabkan simulasi berjalan di bawah 30 FPS.

Untuk mengetahui hal tersebut, dilakukan pengujian tambahan dengan memanfaatkan fitur Unity Profiler.



Gambar 21: Data Profiler - Menit Awal



Gambar 22: Data Profiler - Waktu Selanjutnya

Dari dua gambar di atas, yaitu Gambar 21 dan Gambar 22, pada awal simulasi ketika jumlah kendaraan yang disimulasikan masih sedikit, proses yang paling berat adalah proses rendering. Namun, 30 menit kemudian proses yang paling berat adalah proses perilaku. Hal ini menjelaskan mengapa penggunaan GPU yang baik sekalipun masih belum mampu menghasilkan FPS di atas 30.

VI. KESIMPULAN

Dengan penggunaan graf berbobot waktu, pemanfaatan simulasi lalu lintas menjadi jauh lebih baik, tidak hanya memvisualisasikan kondisi lalu lintas tetapi juga memberikan informasi terkait. Aturan lalu lintas yang dapat dikustomisasi memungkinkan pengguna untuk memvisualisasikan berbagai skenario aturan lalu lintas. Namun, semakin banyak kendaraan lalu lintas yang bergerak, semakin besar pula sumber daya CPU yang dibutuhkan. Hal ini menyebabkan penurunan FPS hingga berada pada kondisi yang tidak dapat digunakan. Permasalahan ini seharusnya dapat diatasi dengan bantuan pemrograman multithread pada kecerdasan buatan kendaraan lalu lintas.

DAFTAR PUSTAKA

[1] Z. Zeng and Y. Zhang, "The impact of traffic crashes on urban network traffic flow," *Sustainability*, vol. 11, no. 23, 2019.

[2] L. Talamini, A. La Torre, and E. De Angelis, "On the impact of the rules on autonomous drive learning," *Applied Sciences*, vol. 10, no. 20, 2020.

[3] P. Coppola, F. P. de Andrea, and A. Bonifazi, "Fuzzy-based variable speed limits system under connected vehicle environment: A simulation-based case study in the city of naples," *IEEE Open Journal of Intelligent Transportation Systems*, 2023.

[4] H. F. Halauoi, "Smart traffic online system (stos): Presenting road networks with time-weighted graphs," in *Conference on Information Society*. London, UK: IEEE, 2010.

[5] M. S. E. Hamam, "Macro-micro simulation of traffic flow," Thesis, Faculty of Applied Sciences - University of Artois, 2006.

[6] GAIKINDO, "Domestic auto market and production (2017)," https://www.gaikindo.org.id/domestic-auto-market-production-2017/, 2017, accessed: 2026-01-16.

[7] N. G. Indonesia, "Penyebab kemacetan di jalan raya," http://nationalgeographic.co.id/berita/2017/11/ternyata-ini-penyebab-kemacetan-tanpa-alasan-yang-sering-terjadi-di-jalan-raya/, November 2017, accessed: 2026-01-16.

[8] S. Hegde and D. Hegde, "Development of safety envelopes and subsea traffic rules for autonomous remotely operated vehicles," *Journal of Loss Prevention in the Process Industries*, vol. 63, pp. 150–156, 2019.

[9] J. Bonnell and A. L. Wilson, "Velocity obstacles and emergent rules-of-the-air for autonomous drone traffic management," *Applied Sciences*, vol. 11, no. 2, 2021.