Institut
Teknologi
Sepuluh Nopember

TUGAS AKHIR – EC 184801

PENGUKURAN KECEPATAN KENDARAAN BERBASIS *DEEP LEARNING*

Selvy Andy Wijaya
NRP 07211540000011

Dosen Pembimbing
Arief Kurniawan, S.T., M.T.
Dr. Eko Mulyanto Yuniarno, S.T., M.T.

DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

TUGAS AKHIR – EC 184801

PENGUKURAN KECEPATAN KENDARAAN BERBASIS *DEEP LEARNING*

Selvy Andy Wijaya
NRP 07211540000011

Dosen Pembimbing
Arief Kurniawan, S.T., M.T.
Dr. Eko Mulyanto Yuniarno, S.T., M.T.

DEPARTEMEN TEKNIK KOMPUTER
Fakultas Teknologi Elektro
Institut Teknologi Sepuluh Nopember
Surabaya 2019

FINAL PROJECT – EC 184801

VEHICLE SPEED MEASUREMENT BASED ON DEEP LEARNING

Selvy Andy Wijaya
NRP 07211540000011

Advisor
Arief Kurniawan, S.T., M.T.
Dr. Eko Mulyanto Yuniarno, S.T., M.T.

DEPARTMENT OF COMPUTER ENGINEERING
Faculty of Electrical Technology
Sepuluh Nopember Institute of Technology
Surabaya 2019

PERNYATAAN KEASLIAN
TUGAS AKHIR

Dengan ini saya menyatakan bahwa isi sebagian maupun keseluruhan Tugas Akhir saya dengan judul "**Pengukuran Kecepatan Kendaraan Berbasis *Deep Learning***" adalah benar-benar hasil karya intelektual sendiri, diselesaikan tanpa menggunakan bahan-bahan yang tidak diijinkan dan bukan merupakan karya orang lain yang saya akui sebagai karya sendiri.

Semua referensi yang dikutip maupun durujuk telah ditulis secara lengkap pada daftar pustaka.

Apabila ternyata pernyataan ini tidak benar, saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Surabaya, Juli 2019

Selvy Andy Wijaya
NRP. 07211540000011

LEMBAR PENGESAHAN

Pengukuran Kecepatan Kendaraan Berbasis *Deep Learning*

Tugas Akhir ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik di Institut Teknologi Sepuluh Nopember Surabaya

Oleh: Selvy Andy Wijaya (NRP: 07211540000011)

Tanggal Ujian: 19 Juni 2019

Periode Wisuda: September 2019

Disetujui oleh:

(Pembimbing I)

Arief Kurniawan, S.T., M.T.
NIP: 197409072002121001

(Pembimbing II)

Dr. Eko Mulyanto Yuniarno, S.T., M.T.
NIP: 196806011995121009

(Penguji I)

Dr. Supeno Mardi Susiki N., S.T., M.T.
NIP: 197003131995121001

(Penguji II)

Dr. Diah Puspito Wulandari, S.T., M.Sc.
NIP: 198012192005012001

(Penguji III)

Susi Juniastuti, S.T., M.Eng.
NIP: 196506181999032001

Mengetahui
Kepala Departemen Teknik Komputer

Dr. I Ketut Eddy Purnama, ST., MT.
NIP. 196907301995121001

ABSTRAK

Nama Mahasiswa	:	Selvy Andy Wijaya
Judul Tugas Akhir	:	Pengukuran Kecepatan Kendaraan Berbasis *Deep Learning*
Pembimbing	:	1. Arief Kurniawan, ST., MT. 2. Dr. Eko Mulyanto Yuniarno, ST., MT.

Berdasarkan data dari BPS (Badan Pusat Statistik), jumlah kendaraan di Kota Surabaya sebanyak 2.126.168 unit pada tahun 2015 dan pada tahun sebelumnya sebanyak 2.011.512 unit, yang mana mengalami kenaikan sebanyak 114.656 unit atau sekitar 5.7%. Jumlah kendaraan tersebut terus bertambah dari tahun ke tahun dan dapat berpotensi menimbulkan kemacetan. Maka dari itu, diperlukan sebuah sistem yang bisa digunakan untuk memantau kondisi kepadatan arus lalu lintas. Sistem ini menggunakan CCTV untuk merekam video arus lalu lintas di Jalan Basuki Rahmat, Kota Surabaya. Hasil tangkapan kamera digunakan untuk mendeteksi jenis kendaran berbasis *deep learning* dengan metode YOLO *(You Only Look Once)*, yang kemudian dihitung kecepatan kendaraan tersebut yang melewati jarak RoI *(Region of Interest)* yang sudah ditentukan dengan melakukan *tracking* pada objek menggunakan metode *Kalman Filter* dan SORT *(Sort Online and Real-Time Tracking)*. Dari hasil pengujian didapatkan akurasi tinggi pada percobaan pertama untuk kecepatan spidomoter 20 km/h sebesar 95.05% dan pada percobaan ketiga untuk kecepatan spidomoter 40 km/h sebesar 99.28% di jarak RoI 15 m. Untuk kecepatan 60 km/h akurasi tinggi didapatkan pada percobaan kedua sebesar 94.73% di jarak RoI 20 m. Sedangkan untuk kecepatan 30 km/h hanya memiliki akurasi tinggi sebesar 77.3% pada percobaan kedua di jarak RoI 25 m.

Kata Kunci: Kendaraan, Kecepatan, *Deep Learning*, YOLO, *tracking*, CCTV.

Halaman ini sengaja dikosongkan

ABSTRACT

Name : Selvy Andy Wijaya
Title : *Vehicle Speed Measurement Based On Deep Learning*
Advisor : 1. Arief Kurniawan, ST., MT.
2. Dr. Eko Mulyanto Yuniarno, ST., MT.

Based on data from BPS (Badan Pusat Statistik) Kota Surabaya, the number of vehicles is 2.126.168 units in 2015 and in the previous year is 2.011.512 units, which increased by 114.656 units or around 5.7%. The number of vehicles continues to increase year by year and it can potentially cause congestion. Therefore, a system is needed that can be used to monitor the density of traffic flows. This system uses CCTV to record video of traffic flow on Jalan Basuki Rahmat, Kota Surabaya. Sytem will be detect the type of vehicle by using deep learning of YOLO (You Only Look Once), which is then calculated the speed of the vehicle that passes the distance of the RoI (Region of Interest) that has been determined by tracking objects using the Kalman Filter and SORT (Sort Online and Real-Time Tracking) methods. From the test results obtained high accuracy in the first experiment for 20 km/h at 95.05% and on the third trial for the speed of the 40 km/h speedometer is 99.28% at a distance of 15 m RoI. For speeds of 60 km/h high accuracy is measured the second experiment was 94.73% at a distance of 20 m RoI. Whereas the speed of 30 km/h only has high accuracy amounted to 77.3% in the second trial at a distance of 25 m RoI.

Keywords: Vehicle, Speed, Deep Learning, YOLO, tracking, CCTV.

Halaman ini sengaja dikosongkan

KATA PENGANTAR

Puji dan syukur kehadirat Allah SWT atas segala limpahan berkah dan hidayah-Nya, penulis dapat menyelesaikan penelitian ini dengan judul **"Pengukuran Kecepatan Kendaraan Berbasis *Deep Learning*"** untuk mendeteksi jenis kendaraan menggunakan metode YOLO dan menghitung kecepatan kendaraan tersebut yang melewati jarak RoI yang sudah ditentukan.

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Departemen Teknik Komputer, FTE-ITS serta digunakan sebagai persyaratan menyelesaikan pendidikan S1. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Keluarga, Ibu, Bapak dan Saudara tercinta yang telah memberikan dorongan spiritual dan material dalam penyelesaian buku penelitian ini.
2. Bapak Dr. I Ketut Eddy Purnama, S.T., M.T. selaku Kepala Departemen Teknik Komputer, Fakultas Teknologi Elektro, Institut Teknologi Sepuluh Nopember.
3. Secara khusus penulis mengucapkan terima kasih yang sebesar-besarnya kepada Bapak Arief Kurniawan, ST., MT. dan Bapak Dr. Eko Mulyanto Yuniarno, ST., MT. atas bimbingan selama pengerjaan tugas akhir ini.
4. Bapak-ibu dosen pengajar Departemen Teknik Komputer atas pengajaran, bimbingan, serta perhatian yang diberikan kepada penulis selama ini.
5. Dan teman-teman Keputih 3c Baday, teman-teman 2015 serta teman-teman Laboratorium Telematika *B201-crew* yang tidak dapat saya sebutkan satu-persatu.

Kesempurnaan hanya milik Allah SWT, untuk itu penulis memohon segenap kritik dan saran yang membangun. Semoga penelitian ini dapat memberikan manfaat bagi kita semua. Amin.

Surabaya, 13 Juni 2019

Penulis

Halaman ini sengaja dikosongkan

DAFTAR ISI

Halaman ini sengaja dikosongkan

DAFTAR GAMBAR

Halaman ini sengaja dikosongkan

DAFTAR TABEL

NOMENKLATUR

V	: Kecepatan
D	: Jarak Tempuh
T	: Waktu Tempuh
S	: Ukuran *Feature Map*
B	: Jumlah *Bounding Box*
C	: Jumlah *Class*
c_x	: Jarak *box* x dari pojok kiri atas
c_y	: Jarak *box* y dari pojok kiri atas
b_x	: Nilai x *output* model setelah diproses
b_y	: Nilai y *output* model setelah diproses
p_w	: Ukuran *anchor* w
p_h	: Ukuran *anchor* h
b_w	: Nilai w *output* model setelah diproses
b_h	: Nilai h *output* model setelah diproses
$\hat{x_i}$	: Nilai x label
$\hat{y_i}$	: Nilai y label
$\hat{C_i}$	: Nilai *Confidence* label
$p_i\hat{(c)}$	: Nilai probabilitas *Class* label
$\bar{G}$	: Nilai galat
$p(X_t\|U_t, X_t1)$	: Probabilitas pada *state* berikutnya
$p(Z_t\|X_t)$	: Probabilitas pengukuran
$bel(X_0)$	: Nilai kepercayaan awal

Halaman ini sengaja dikosongkan

BAB 1
PENDAHULUAN

Penelitian ini di latar belakangi oleh berbagai kondisi yang menjadi acuan. Selain itu juga terdapat beberapa permasalahan yang akan dijawab sebagai luaran dari penelitian.

1.1 Latar belakang

Berdasarkan data dari BPS (Badan Pusat Statistik) jumlah kendaraan di Kota Surabaya sebanyak 2.126.168 unit pada tahun 2015 dan pada tahun sebelumnya sebanyak 2.011.512 unit, yang mana mengalami kenaikan sebanyak 114.656 unit atau sekitar 5.7% [14]. Jumlah kendaraan tersebut terus bertambah dari tahun ke tahun dan dapat berpotensi mengakibatkan kemacetan sehingga seiring dengan perkembangan teknologi sekarang, setiap ruas jalan telah dipasang CCTV *(Closed circuit television)* untuk mengetahui kondisi arus lalu lintas pada suatu jalan.

Penggunaan CCTV tersebut hanya mampu merekam kondisi arus jalan tanpa bisa melakukan estimasi kecepatan kendaraan beserta klasifikasi jenis kendaraan. Oleh karena itu, dalam metode ini dapat dimanfaatkan dengan menambahkan teknologi berbasis *deep learning* di dalamnya. Teknologi *deep learning* merupakan salah satu bidang riset yang berkembang pesat dalam beberapa tahun belakangan ini. Perhitungan kecepatan kendaraan dengan kemunculan *deep learning* untuk klasifikasi dan deteksi objek ini dimanfaatkan pada beragam aplikasi seperti pengawasan, manajemen lalu lintas, dan tugas penyelamatan untuk HPV *(High Priority Vehicle)* dalam penentuan rute tercepat berdasarkan penggunaan data kecepatan kendaraan.

Pada sistem ini, untuk dapat menghitung kecepatan kendaraan yang melewati suatu jalan maka sistem harus mampu melakukan *tracking* pada kendaraan yang melintas di dalam RoI *(Region of Interest)* dari video. *Deep Learning* membantu mendeteksi kendaraan dan memberikan label sesuai dengan klasifikasi kendaraan kemudian dengan *library OpenCV*, sistem akan menggambar kotak pada kendaraan yang terdeteksi oleh CCTV.

Bedasarkan penelitian sebelumnya oleh Aldiansyah Ramadlan [15] yang mengimplementasikan *image processing* untuk menghitung kecepatan kendaraan menggunakan CCTV dengan menggunakan algoritma *matching based*. Dalam pengujiannya didapatkan tingkat akurasi terbaik sebesar 97.01% pada kondisi tidak ada bayangan dengan jumlah *frame rate* 30 fps dan dengan akurasi sebesar 83.71% pada kondisi ada bayangan. Sistem ini juga dapat menentukan jenis kendaraan mobil atau motor yang terdeteksi melanggar kecepatan dengan akurasi sebesar 89.62%.

Dari penelitian sebelumnya, data kecepatan kendaraan yang didapatkan digunakan untuk memantau pelanggaraan lalu lintas yang melebihi batas kecepatan yang dianjurkan yang mana memiliki keterbatasan, yaitu hanya bisa melakukan klasifikasi dua jenis kendaraan saja (mobil dan motor), tidak bisa digunakan untuk melakukan pengukuran kecepatan kendaraan pada kondisi padat dan pada waktu malam hari. Dalam tugas akhir ini dikembangkan suatu sistem berbasis *deep learning* untuk melakukan pengukuran kecepatan kendaraan beserta klasifikasi jenis kendaraan yang terekam mengggunakan CCTV.

1.2 Permasalahan

Berdasarkan latar belakang tersebut jumlah kendaraan yang meningkat setiap tahunnya dapat berpotensi terjadi kemacetan yang mana diperlukan sebuah sistem pemantauan kondisi arus lalu lintas dengan memanfaatkan data kecepatan kendaraan dikarenakan pemantauan kondisi arus lalu lintas secara konvensional hanya mengandalkan CCTV saja sulit dilakukan, terutama dalam pengukuran kecepatan kendaraan beserta klasifikasi jenis kendaraan pada kondisi arus lalu lintas yang padat.

1.3 Tujuan

Tujuan dari penelitian ini adalah mengembangkan suatu sistem berbasis *deep learning* untuk mendapatkan data kecepatan kendaraan beserta jenis kendaraan yang terekam mengggunakan CCTV yang harapannya bisa digunakan untuk memantau kondisi arus lalu lintas yang padat.

1.4 Batasan masalah

Batasan masalah pada Tugas Akhir ini adalah :

1. Pengujian sistem dilakukan di Jalan Basuki Rahmat.
2. Pengujian sistem dilakukan pada kondisi jalan dengan arus satu arah.
3. Waktu pengujian sistem dilakukan pada waktu pagi, siang, sore dan malam.
4. Pengujian sistem dilakukan pada kondisi cuaca normal (tidak hujan).
5. Kendaraan yang dideteksi hanya 4 jenis, yaitu *car, motorbike, bus* dan *truck*.
6. Pengujian hanya mendeteksi kecepatan secara sistem (estimasi).

1.5 Sistematika Penulisan

Laporan penelitian Tugas akhir ini tersusun dalam sistematika dan terstruktur sehingga mudah dipahami dan dipelajari oleh pembaca maupun seseorang yang ingin melanjutkan penelitian ini. Alur sistematika penulisan laporan penelitian ini yaitu :

1. BAB I Pendahuluan
   Bab ini berisi uraian tentang latar belakang permasalahan, penegasan dan alasan pemilihan judul, sistematika laporan, tujuan dan metodologi penelitian.
2. BAB II Dasar Teori
   Pada bab ini berisi tentang uraian secara sistematis teori-teori yang berhubungan dengan permasalahan yang dibahas pada pengerjaan tugas akhir ini. Teori-teori ini digunakan sebagai dasar dalam pengerjaan, yaitu informasi terkait pertumbuhan jumlah kendaraan, teori tentang *CCTV*, teori mengenai *deep learning*, dan teori-teori penunjang lainya.
3. BAB III Perancangan Sistem dan Impementasi
   Bab ini berisi tentang penjelasan-penjelasan terkait sistem yang akan dibuat. Guna mendukung itu digunakanlah blok diagram atau *work flow* agar sistem yang akan dibuat dapat terlihat dan mudah dibaca untuk implentasi pada pelaksanaan tugas akhir.

4. BAB IV Pengujian dan Analisa
   Bab ini menjelaskan tentang pengujian yang dilakukan terhadap sistem hasil dari tugas akhir ini dan menganalisa sistem tersebut. Spesifikasi perangkat keras dan perangkat lunak yang digunakan juga disebutkan dalam bab ini. Sehingga ketika akan dikembangkan lebih jauh, spesifikasi perlengkapannya bisa dipenuhi tanpa harus melakukan uji coba perangkat lunak maupun perangkat keras lagi.
5. BAB V Penutup
   Bab ini merupakan penutup yang berisi kesimpulan yang diambil dari penelitian dan pengujian yang telah dilakukan. Saran dan kritik yang membangun untuk pengembangkan lebih lanjut juga dituliskan pada bab ini.

1.6 Relevansi

Penelitian mengenai *deep learning* merupakan bidang riset yang sangat dibutuhkan dan dipakai dalam pemenuhan kebutuhan komputasi yang semakin beragam saat ini. Dalam tugas akhir ini, dilakukan penelitian mengenai pengembangan sistem pengukuran kecepatan kendaraan berbasis *deep learning*.

BAB 2
TINJAUAN PUSTAKA

Demi mendukung penelitian ini, dibutuhkan beberapa teori penunjang sebagai bahan acuan dan referensi. Dengan demikian penelitian ini menjadi lebih terarah.

2.1 Pengukuran Kecepatan

Perhitungan kecepatan pada objek yang bergerak dapat dihitung dengan mencari total jarak yang ditempuh dari awal objek bergerak sampai akhir objek bergerak dibagi dengan waktu yang ditempuh dalam melakukan perpindahan menggunakan persamaan berikut.

$$V = \frac{D}{T} \tag{2.1}$$

Pada persaaman 2.1 tersebut, dimana:

V merupakan *velocity* atau kecepatan yang dicapai dalam melakukan perpindahan dengan Satuan Internasional (SI) adalah meter/sekon (m/s).

D merupakan *distance* atau jarak yang ditempuh dengan Satuan Internasional (SI) adalah meter (m)

T merupakan *time* atau waktu yang ditempuh dalam melakukan perpindahan dengan Satuan Internasional (SI) adalah sekon (s).

Proses perhitungan untuk mendapatkan kecepatan laju kendaraan dari hasil tangkapan video dengan objek yang bergerak, dapat dihitung dengan menggunakan persamaan berikut [16]:

$$Kecepatan = \frac{PanjangGaris \times fps \times 3600}{JumlahFrame \times 1000} km/h \tag{2.2}$$

Pada persamaan 2.2 tersebut, dimana:

1. Pajang garis merupakan nilai jarak didapatkan berdasarkan jarak RoI *(Region of Interest)* yang telah ditentukan.
2. *fps* merupakan *frame per second* yaitu kemampuan kamera dalam mengambil citra tiap detiknya.
3. 3600 merupakan konversi dari sekon ke jam.
4. Jumlah *frame* merupakan banyak *frame* yang ditempuh oleh objek dalam batas garis yang sudah ditentukan.
5. 1000 merupakan konversi dari kilometer ke meter.

2.2 CCTV

CCTV atau (*Close Circuit Television*) merupakan kamera pemantau yang digunakan untuk pengawasan dalam bentuk video. Seiring dengan perkembangan teknologi modern, CCTV kamera menggunakan *Internet Protokol* atau IP untuk mengirimkan data gambar dan sinyal kendali atas *Fast Ethernet link*. Dengan demikian, CCTV dapat disebut sebagai *IP Camera* atau disebut sebagai kamera jaringan karena setiap kamera memiliki IP sendiri sehingga kita bisa memilih kamera mana yang mau dilihat [17].

Gambar 2.1: Kamera Vivotek IB8354-C[[1]]

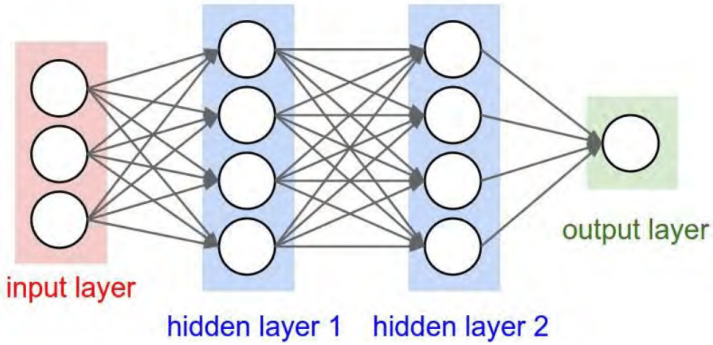
2.3 *Deep Learning*

Secara historis, konsep *deep learning* berasal dari penelitian *artificial neural network*. Pada kasus *Multi layer perceptron* atau MLP sebuah jaringan tanpa *hidden layer* dapat memetakan persamaan linear apapun, sedangkan jaringan dengan satu atau dua *hidden layer* dapat memetakan sebagian besar persamaan pada data sederhana. Namun pada data yang lebih kompleks, MLP memiliki keterbatasan. Pada permasalahan jumlah *hidden layer* dibawah tiga *layer*, terdapat pendekatan untuk menentukan jumlah neuron pada masing-masing layeruntuk mendekati hasil optimal. Penggunaan *layer* diatas dua pada umumnya tidak direkomendasikan dikarenakan akan menyebabkan *overfitting* serta kekuatan *backpropagation* berkurang secara signifikan [18]. *Backpropagation* (BP) dipopulerkan pada 1980-an, telah menjadi algoritma terkenal untuk mempelajari parameter jaringan ini. Sayangnya, *back-propagation* sendiri tidak bekerja dengan baik dalam praktiknya saat itu untuk mempelajari jaringan dengan jumlah lebih dari *hidden layer*. Fungsi jaringan yang dalam adalah sumber utama kesulitan dalam *learning*. *Back-propagation* didasarkan pada informasi awal menggunakan beberapa titik acak. Tingkat keparahan meningkat secara signifikan seiring kedalaman jaringan meningkat.

Dengan berkembangnya *deep learning*, ditemukan bahwa untuk mengatasi kekurangan MLP dalam menangani data kompleks, diperlukan fungsi untuk mentransformasi data *input* menjadi bentuk yang lebih mudah dimengerti oleh MLP. Hal tersebut memicu berkembangnya *deep learning* dimana dalam satu model diberi beberapa *layer* untuk melakukan transformasi data sebelum data diolah menggunakan metode klasifikasi. Hal tersebut memicu berkembangnya model *neural network* dengan jumlah *layer* diatas tiga. Namun dikarenakan fungsi *layer* awal sebagai metode ekstraksi fitur, maka jumlah *layer* dalam sebuah *deep neural network* atau DNN tidak memiliki aturan universal dan berlaku berbeda-beda tergantung dataset yang digunakan.

Metode *deep learning* adalah metode representasi pembelajaran yang merupakan seperangkat metode yang memungkinkan mesin dengan data mentah dapat secara otomatis menemukan representasi diperlukan untuk deteksi atau klasifikasi. Metode representasi

memiliki berbagai tingkat representasi, diperoleh dengan menyusun modul sederhana namun non-linear yang masing-masing mengubah representasi pada satu level (dimulai dengan input mentah) menjadi representasi yang lebih tinggi, level yang sedikit lebih abstrak. Dengan komposisi yang cukup seperti transformasi, maka fungsi yang sangat kompleks bisa dipelajari. Untuk tugas-tugas klasifikasi, *layer* representasi yang lebih tinggi aspek amplifikasi input yang penting untuk diskriminasi dan variasi penindasan yang tidak relevan. Sebuah gambar, misalnya, muncul dibentuk *array* nilai piksel, dan fitur yang dipelajari. Pertama, biasanya *layer* representasi mewakili ada atau tidak adanya tepi orientasi dan lokasi tertentu dalam gambar. Kedua, biasanya mendeteksi motif dengan melihat pengaturan tertentu *edge*, variasi kecil pada posisi *edge*. Ketiga, *layer* dapat merakit motif-motif menjadi kombinasi yang lebih besar yang sesuai ke bagian-bagian dari objek yang akrab, dan lapisan selanjutnya akan mendeteksi objek sebagai kombinasi dari bagian-bagian ini [2].

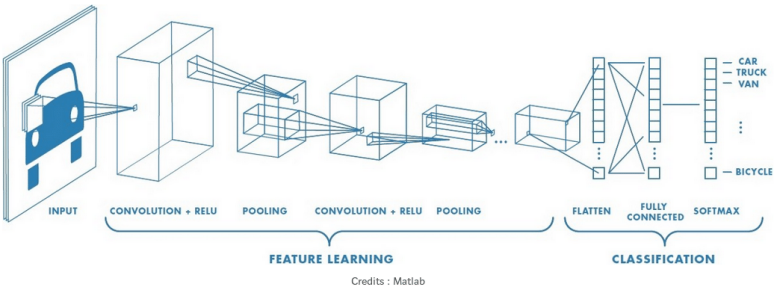


Gambar 2.2: Arsitektur MLP Sederhana[[2]]

2.4 *CNN*

Convolutional Neural Network atau disingkat CNN merupakan pengembangan dari *Multilayer Perceptron* atau MLP yang didesain untuk mengolah data dua dimensi. CNN termasuk dalam jenis *Deep Learning Neural Network* karena kedalaman jaringan yang

tinggi dan banyak diaplikasikan pada data citra. CNN memiliki dua metode, yakni klasifikasi menggunakan *feedforward* dan tahap pembelajaran menggunakan *backpropagation* [18]. Sebuah *Convolutional Neural Network* sederhana memiliki urutan lapisan yang digunakan untuk membangun dimulai dari *Convolutional Layer*, *Pooling Layer*, dan *Fully-Connected Layer* [4].



Gambar 2.3: Lapisan yang terdapat pada CNN[[3]]

2.4.1 *Convolutional Layer*

Convolutional Layer melakukan operasi konvolusi pada *output* dari *layer* sebelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berati mengaplikasikan sebuah fungsi pada *output* fungsi lain secara berulang. Untuk menentukan ukuran volume *output* hasil konvolusi dapat menggunakan persamaan berikut ini :

$$Output = \frac{W - N + 2P}{S} + 1 \tag{2.3}$$

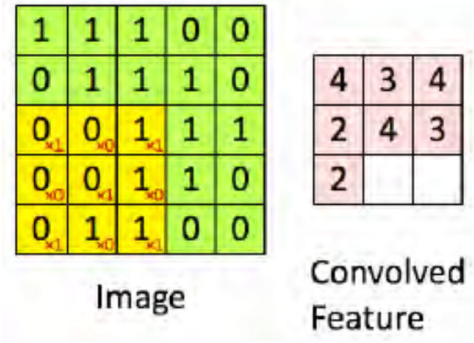
.Pada persaaman 2.3 tersebut, dimana:

N merupakan panjang atau tinggi *input*

W merupakan panjang atau tinggi *filter*

P atau *zero padding* merupakan parameter yang menentukan jumlah *pixels* (berisi nilai 0) yang akan ditambahkan di setiap sisi dari *input*.

S atau *stride* merupakan parameter yang menentukan berapa jumlah pergeseran *filter*.



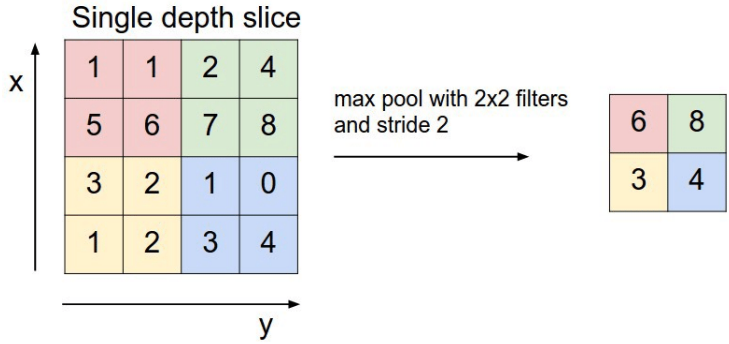
Gambar 2.4: Ilustrasi Operasi Konvolusi[[4]]

Dalam pengolahan citra, konvolusi berati mengaplikasikan sebuah kernel (kotak kuning) pada citra disemua *offset* yang memungkinkan seperti yang ditunjukkan pada Gambar 2.4. Kotak hijau secara keseluruhan adalah citra yang akan dikonvolusi. Kernel bergerak dari sudut kiri atas ke kanan bawah. Sehingga hasil konvolusi dari citra tersebut dapat dilihat pada gambar disebelah kanannya. Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstraksi fitur dari citra *input*. Konvolusi akan menghasilkan transformasi linear dari data *input* sesuai informasi spasial pada data. Bobot pada *layer* tersebut menspesifikasikan kernel konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan *input* pada CNN.

2.4.2 *Pooling Layer*

Pooling layer biasanya berada setelah *convolutional layer*. Pada prinsipnya *pooling layer* terdiri dari sebuah *filter* dengan ukuran dan *stride* tertentu yang akan bergeser pada seluruh area *feature map*. *Pooling* yang biasa digunakan adalah *Max Pooling* dan *Average Pooling*. Sebagai contoh jika menggunakan *Max Pooling* 2x2 dengan *stride* 2, maka pada setiap pergeseran *filter*, nilai maksimum pada area 2x2 *pixel* tersebut yang akan dipilih, sedangkan

Average Pooling akan memilih nilai rata-ratanya. Tujuan dari penggunaan *pooling layer* adalah mengurangi dimensi dari *feature map (down sampling)*, sehingga mempercepat komputasi karena parameter yang harus di-*update* semakin sedikit dan mengatasi *overfitting*. Untuk menentukan dimensi *output* dari *Pooling layer* juga menggunakan persamaan 2.3 sama dengan persamaan *convolutional layer*.



Gambar 2.5: Operasi *Max Pooling*[[4]]

2.4.3 *Fully-Connected Layer*

Layer tersebut adalah *layer* yang biasanya digunakan dalam penerapan MLP dan bertujuan untuk melakukan transformasi pada dimensi data agar data dapat diklasifikasikan secara linear. Setiap *neuron* pada *convolution layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah *fully connected layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversibel, *fully connected layer* hanya dapat diimplementasikan di akhir jaringan.

2.5 YOLOv3

YOLO *(You Only Look Once)* merupakan metode *object detection* berdasarkan CNN *(Convolutional Neural Network)* [19]. Dimensi gambar *input* masing-masing sebesar 32, 16 dan 8. Pengembangan YOLO telah melewati YOLOv1, YOLOv2, dan YOLOv3.

Seri R-CNN membutuhkan proposal wilayah untuk mendeteksi RoI pada gambar. Dibandingkan dengan R-CNN, YOLO tidak mengadopsi prosedur proposal daerah tetapi langsung menggunakan regresi untuk mendeteksi target regresi untuk menyelesaikan masalah deteksi target.

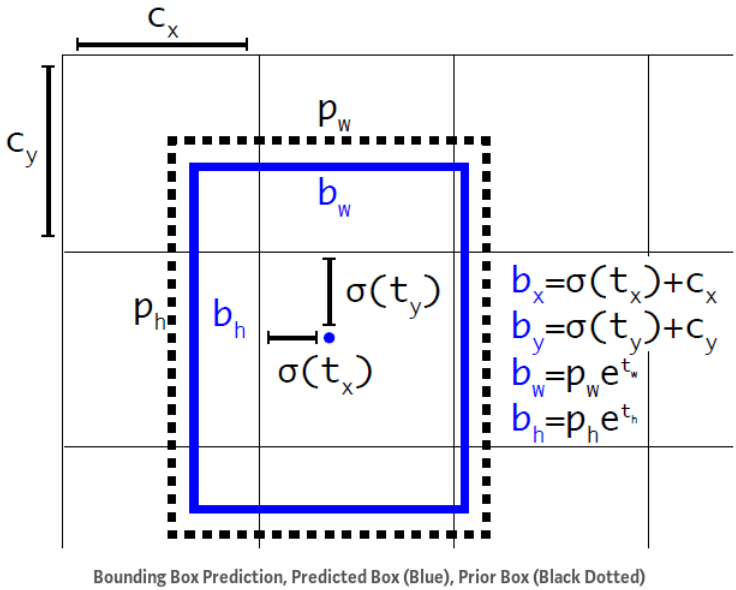
Meskipun YOLO sebagian besar meningkatkan kecepatan deteksi target menggunakan metode regresi, YOLO masih menyajikan margin kesalahan yang cukup besar. Untuk mengatasi cacat tersebut, YOLOv3 membawa kotak jangkar dan algoritma pengelompokan *K-means* dalam jaringannya untuk menghasilkan batas-batas yang sesuai sebelumnya. YOLOv3 merenovasi desain struktur jaringan berdasarkan YOLOv2 dan menggunakan konvolusi pada lapisan *output*. Untuk lebih meningkatkan hasil deteksi, YOLOv3 juga mengadopsi normalisasi *batch*, *classifier* resolusi tinggi, dan sebagainya. Prediksi multiskala juga diadopsi untuk mendeteksi target akhir dalam model YOLOv3. Oleh karena itu, versi terbaru YOLOv3 tidak hanya memiliki presisi dan kecepatan tinggi yang diinginkan tetapi juga memiliki kinerja yang luar biasa pada deteksi target kecil [20].

2.5.1 *Bounding Box Prediction*

Setiap grid dari YOLOv3 memprediksi 5 nilai yaitu: x , y , w , h dan *confidence* ditambah dengan C (jumlah *class* probabilitas). Nilai *(x,y)* koordinat mempresentasikan titik pusat objek berdasarkan *grid cell*. Nilai *(w,h)* merupakan panjang dan lebar yang relatif pada ukuran gambar. Terakhir nilai *confidence* yang memprediksi IOU *(Intersection Over Union)* antara *bounding box* prediksi dan *bounding box* yang asli untuk mendeteksi ada atau tidaknya objek pada *grid cell*, nilai *confidence* kemudian di kalikan dengan probabilitas objek seperti pada persamaan 2.4. Jika tidak ada objek maka nilai *confidence* akan mendekati 0 [21].

$$Pr(Object) * IOU_{truth}^{pred} \tag{2.4}$$

YOLOv3 mendeteksi 3 *bounding box* dalam 3 skala dengan menggunakan COCO dataset yang berjumlah 80 *class* memiliki *output* seperti pada persamaan 2.5. Dengan 3 skala dan tiap skala ada 3 *anchor*, jadi jumlahnya 9 *anchor*. Pengelompokkan diatur dengan



Gambar 2.6: *Bounding Box Prediction*[[5]]

algoritma *k-means*. Pada COCO dataset terdapat 9 ukuran yang digunakan adalah: (10,13), (16,30), (33,23), (30,61), (62,45), (59,119), (116,90), (156,198), (373,326) [22]. Nilai *class* probabilitas dan *confidence* digunakan untuk menentukan klasifikasi objek. nilai *(cx,cy)* berasal dari jarak *grid* dari pojok kiri atas gambar. Sedangkan nilai *(pw, ph)* ukuran *bounding box* yang sudah ditentukan sebelumnya. Ukuran *bounding box* ditentukan dengan mengelompokkan ukuran label kotak menjadi beberapa kelompok.

$$N \times N \times N[B * (4 + 1 + C)] \tag{2.5}$$

$13 \times 13 \times 13 \times 3 \times 85$

$26 \times 26 \times 26 \times 3 \times 85$

$52 \times 52 \times 52 \times 3 \times 85$

Namun, ada kemungkinan satu objek yang sama dideteksi oleh beberapa *bounding box*. Untuk itu kotak dengan prediksi yang sama perlu disingkirkan dengan menggunakan NMS *(Non-Max Suppression)*. Cara kerja NMS adalah pertama memilih objek dengan *confidence score* paling tinggi, lalu menyingkirkan semua kotak lain yang memiliki IOU dibawah standar dari *threshold* yaitu, 0.5.

2.5.2 *Class Prediction*

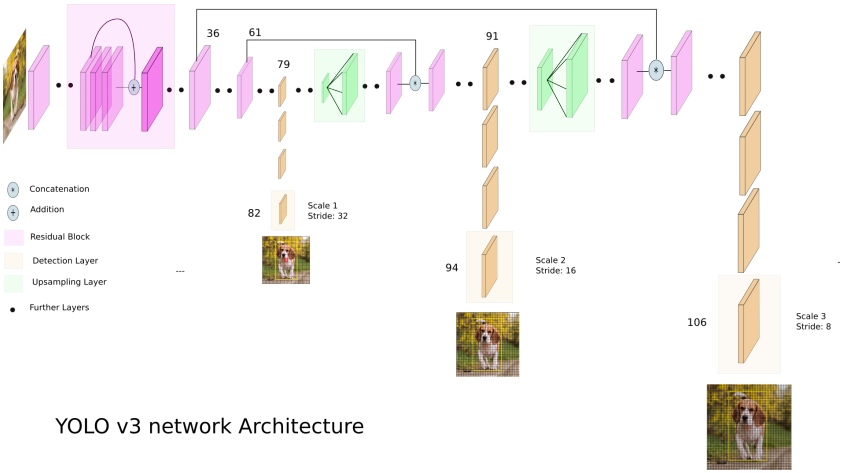
YOLOv3 sudah tidak menggunakan *softmax* dan digantikan dengan menggunakan pengklasifikasi logistik independen untuk setiap kelas. Hal ini dilakukan untuk membuat klasifikasi pada multi-label. Sebagai contoh, di mana ada sebuah mobil ditampilkan dalam gambar dan model melakukan *training* pada mobil dan bus, maka jika menggunakan *softmax* di sini akan menyebabkan probabilitas kelas dibagi antara 2 kelas ini dengan probabilitas 0.4 dan 0.45. Tetapi, pengklasifikasi independen memecahkan masalah ini dan memberikan probabilitas ya atau tidak untuk setiap kelas. Probabilitas bahwa ada seorang mobil dalam gambar akan memberikan 0.8 dan berapa probabilitas bahwa ada bus dalam gambar akan memberikan 0.9 dan kita bisa beri label objek sebagai mobil dan bus.

2.5.3 *Network Architecture*

YOLOv2 menggunakan *Darknet-19* sebagai ekstraktor fitur *backbone*-nya, YOLOv3 menggunakan *network* baru, yaitu *Darknet-53*. *Darknet-53* memiliki 53 lapisan konvolusional, ini lebih dalam dari YOLOv2 dan juga memiliki residu atau koneksi pintasan. Ini lebih kuat dari *Darknet-19* dan lebih efisien daripada *ResNet-101* atau *ResNet-152*.

Dalam gambar 2.7 berikut, gambar *input* berukuran 416 x 416. YOLOv3 membuat prediksi pada tiga skala, yang secara tepat diberikan dengan menurunkan dimensi gambar *input* masing-masing sebesar 32, 16 dan 8.

Deteksi pertama dilakukan oleh lapisan ke-82. Untuk 81 lapisan pertama, gambar turun sampel oleh *network* sehingga lapisan 81 memiliki langkah 32. Jika *input* memiliki gambar 416 x 416, *feature map* yang dihasilkan akan berukuran 13 x 13. Satu deteksi dibuat di sini menggunakan kernel deteksi 1 x 1, memberi *feature*



Gambar 2.7: YOLOv3 *Network Architecture*[[6]]

map deteksi 13 x 13 x 255.

Kemudian, *feature map* dari lapisan 79 dikenakan beberapa lapisan konvolusional sebelum diambil sampel oleh *2x* ke dimensi 26 x 26. *Feature map* ini kemudian digabungkan dengan peta fitur dari lapisan 61. Kemudian peta fitur gabungan lagi dikenakan beberapa lapisan konvolusional 1 x 1 untuk menggabungkan fitur dari lapisan sebelumnya (61). Kemudian, deteksi kedua dilakukan oleh lapisan ke-94, menghasilkan *feature map* deteksi 26 x 26 x 255.

Prosedur serupa diikuti lagi, di mana peta fitur dari lapisan 91 menjadi sasaran beberapa lapisan konvolusional sebelum digabungkan dengan peta fitur dari lapisan 36. Seperti sebelumnya, beberapa lapisan konvolusional 1 x 1 ikut memadukan informasi dari sebelumnya layer (36). Hal ini membuat final dari 3 pada lapisan ke-106, menghasilkan peta fitur ukuran 52 x 52 x 255.

Dari *feature map*, lapisan 13 x 13 bertanggung jawab untuk mendeteksi *large object*, sedangkan lapisan 52 x 52 untuk mendeteksi *small object*, dan lapisan 26 x 26 mendeteksi *medium object*. Tiap grid *cell* output akan mendeteksi 3 *anchor box* yang berbeda. Layer pertama *(13X13X3)=507, (26X26X3)=2028, (52X52X3)=8112*. Ji-

ka di jumlah maka totalnya ada 10,647 kotak yang harus dideteksi.

2.6 *Kalman Filter*

Kalman filter adalah metode yang berisi kumpulan persamaan matematika yang menyediakan perhitungan komputasi yang efisien (rekursif) untuk memperkirakan suatu keadaan dari sebuah proses, dengan cara meminimalkan nilai *error* kuadrat rata-rata antara citra asli dengan citra hasil penyisipan yang biasa disebut dengan MSE *(Mean Squared Error). Filter* ini dapat memperkirakan suatu keadaan di masa sebelumnya, sekarang, dan yang akan datang, bahkan *filter* ini dapat memperkirakan secara tepat sebuah sistem pemodelan yang belum diketahui [23].

Di dalam pembelajaran visi komputer, *kalman filter* digunakan untuk *tracking* suatu objek yang bergerak dengan memperkirakan keadaan suatu sistem linier dengan asumsi keadaan tersebut telah terdistribusi oleh *Gaussian. Kalman filter* melakukan *tracking* suatu objek dengan cara memprediksi posisi objek dari informasi sebelumnya dan memeriksa keberadaan objek pada posisi yang diprediksi.

2.6.1 *Linear Gaussian System*

Salah satu implementasi dari *filter Gaussian-bayes* yang masih umum digunakan hingga saat ini ialah *Kalman Filter. Kalman Filter* merepresentasikan nilai kepercayaan dengan representasi momen; ketika waktu t nilai kepercayaan direpresentasikan dalam dua *random variable* yang tidak lain ialah parameter *Gaussian* μ_t dan σ_t sedemikian sehingga *state posterior* memiliki distribusi *Gaussian* jika memenuhi tiga syarat berdasarkan asusmsi *Markov* yaitu:

(a) Probabilitas pada *state* berikutnya $p(X_t|U_t, X_{t-1})$ harus merupakan merupakan fungsi linear terhadap parameter argumennya dengan penambahan derau *Gaussian.* Hal ini diekspresikan pada persamaan berikut:

$$x_t = A_t X_{t-1} + B_t u_t + \varepsilon_t \tag{2.6}$$

Dimana X_t dan X_{t-1} ialah *vector state*, dan u_t ialah vektor kendali pada waktu t. Sehingga diperoleh distribusi probabi-

litas sebagai berikut, diturunkan dari prinsip *bayes* dan distribusi *Gaussian*.

$$p(X_t|U_t, X_{t-1}) \\ = \frac{1}{\sqrt{2\pi R_t}} exp[-\frac{1}{2}(z_t - C_t X_{t-1} - B_t u_t)^T (Q_t)^{-1}(z_t - c_t X_{t-1}) \\ - (B_t u_t) \tag{2.7}$$

(b) Probabilitas pengukuran $p(Z_t|X_t)$ harus linear terhadap parameter argumennya, dengan penambahan faktor derau dari *Gaussian*.

$$z_t = C_t x_t + \delta_t$$

Sehingga,

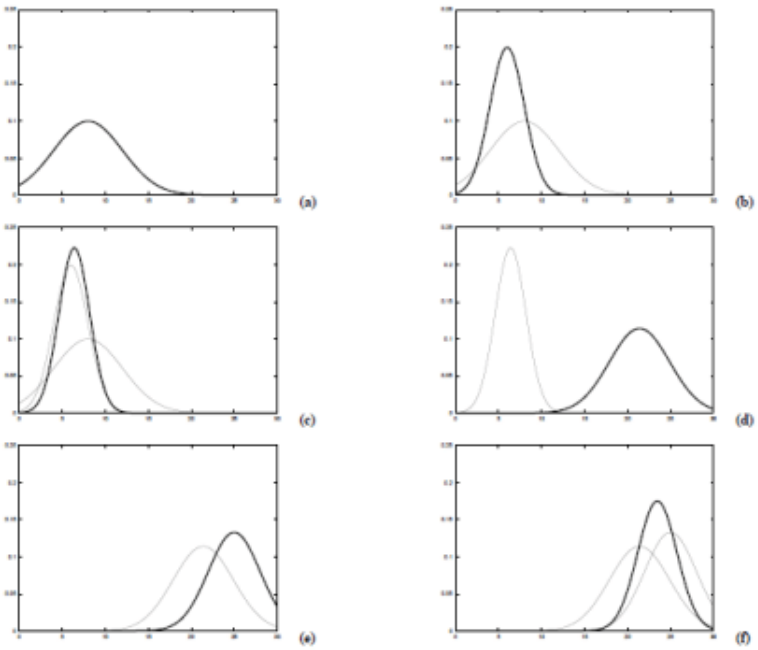
$$p(Z_t|X_t) = \\ = \frac{1}{\sqrt{2\pi Q_t}} exp[-\frac{1}{2}(x_t - A_t X_{t-1} - B_t u_t)^T (R_t)^{-1}(x_t - A_t X_{t-1}) \\ - (B_t u_t) \tag{2.8}$$

(c) Nilai kepercayaan awal $bel(X_0)$harus terdistribusi normal.

$$bel(X_0) = p(X_0) \\ = \frac{1}{\sigma_0 \sqrt{2\pi}} exp[-\frac{1}{2}(x_0 - \mu_0)^T (\mu_0)^{-1}(x_0 - \mu_0) \tag{2.9}$$

Ilustrasi dari kondisi diatas dapat digambarkan sebagai berikut:

Pada gambar 2.8, (a) nilai kepercayaan awal, (b) pengukuran (garis tebal) dengan ketidakpastian terasosiasi (c) nilai kepercayaan setelah pembobotan pengukuran ke dalam nilai kepercayaan *Kalman Filter* (d) nilai kepercayaan setelah pergerakan (e) pengukuran baru dengan ketidakpastian terasosiasi (f) nilai kepercayaan yang dihasilkan [7].



Gambar 2.8: 1ilustrasi *Kalman Filter*[[7]]

2.6.2 *Algortima Kalman Filter*

Kalman filter memperkirakan sebuah proses dalam bentuk kendali umpan balik *(feedback control)*. Persamaan *kalman filter* terbagi dalam dua kelompok, persamaan pembaruan waktu dan persamaan pembaruan pengukuran. Persamaan pembaruan waktu bertugas untuk memproyeksikan keadaan berikutnya (dalam waktu) dan kovarian *error*-nya diperkirakan untuk menghasilkan perkiraan sebelumnya untuk langkah selanjutnya. Persamaan pembaruan pengukuran bertugas sebagai umpan balik. Persamaan ini digunakan untuk memasukkan sebuah pengukuran baru menjadi perkiraan sebelumnya untuk menghasilkan sebuah perkiraan selanjutnya yang lebih baik. Persamaan pembaruan waktu dapat juga disebut dengan

persamaan prediksi, sedangkan persamaan pembaruan pengukuran disebut sebagai persamaan koreksi [24].

$$Xpred_k = A \cdot X_{k-1} + B \cdot U_k + W_{k-1} \tag{2.10}$$

$$Ppred_k = A \cdot P_{k-1} \cdot A + Q \tag{2.11}$$

$Xpred_k$ adalah vektor dari proses suatu keadaan yang terprediksi dalam waktu k. X adalah sebuah vektor 4 dimensi (x, y, dx, dy) dimana x dan y adalah koordinat dari pusat objek, sedangkan dx dan dy mewakili kecepatan. X_{k-1} adalah vektor dari proses suatu kondisi dalam waktu $k-1$. A adalah sebuah proses matriks transisi 4 x 4 dari:

$$A = \begin{matrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{matrix} \tag{2.12}$$

U_k adalah sebuah vektor kendali dan B terhubung secara opsional dengan vektor kendali U_k dalam kondisi ruang. W_{k-1} adalah suatu proses derau *(noise)*. $Ppred_k$ adalah kovariansi *error* yang terprediksi pada waktu k. P_{k-1} adalah sebuah matriks yang mewakili kovarian *error* dalam kondisi prediksi pada waktu $k-1$ dan Q adalah proses kovarian derau. Setelah memprediksi kondisi $Xpred_k$ dan kovarian *error*-nya pada waktu k menggunakan langkah pembaruan waktu. *Kalman filter* selanjutnya akan menggunakan pengukuran untuk mengoreksi prediksi *Kalman* pada saat langkah pembaruan pengukuran.

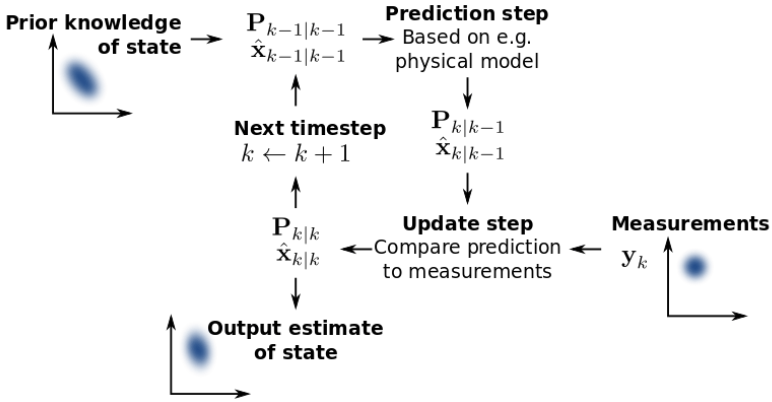
$$K_k = Ppred_k \cdot X_{k-1} + H^T \cdot (H \cdot Ppred_k + H^T + R)^{-1} \tag{2.13}$$

$$X_k = Xpred_k + K_k \cdot (Z_k - H \cdot Xpred_k) \tag{2.14}$$

$$P_k = (1 - K_k \cdot H) \cdot Ppred_k \tag{2.15}$$

Pada persamaan 2.13, K_k adalah penguat *Kalman*. H adalah sebuah matriks yang merubah kondisi ruang menjadi ruang pengukuran dan R adalah kovarian pengukuran derau. Sulit untuk menentukan R_k sebagai pengatur pengukuran, banyak implementasi

Kalman yang secara statistik menganalisa data untuk menentukan nilai R yang tetap pada semua pembaruan waktu yang akan datang. Pada persamaan 2.14, X_k adalah sebuah proses kondisi sebenarnya. Dengan menggunakan K_k dan pengukuran Z_k kondisi proses X_k dapat dilakukan pembaruan. Z_k paling memunginkan adanya kordinat x dan y objek target dalam *frame*. Langkah terakhir dari *Kalman filter* adalah memperbarui kovarian *error* pada $Ppred_k$ yang menjadi P_k sebagaimana dalam persamaan 2.15. Setelah tiap pembaruan waktu dan pengukuran berpasangan, proses diulangi dengan perkiraan lanjutan sebelumnya untuk memproyeksikan dan memprediksi perkiraan sebelumnya yang baru [25].



Gambar 2.9: Konsep dasar *Kalman Filter*[[8]]

2.7 SORT

SORT *(Simple Online and Realtime Tracking)* adalah pendekatan pragmatis untuk MOT *(Multiple Object Tracking)* dengan fokus pada algoritma sederhana dan efektif [26]. Dalam pengaturan masalah MOT, setiap *frame* memiliki lebih dari satu objek jalur. Metode SORT ini untuk mengatasi masalah berikut:

1. Deteksi: Mendeteksi objek dalam *frame*
2. Asosiasi: Setelah deteksi didapatkan, selanjutnya pencocokan dilakukan untuk deteksi yang serupa dengan *frame* sebelum-

nya kemudian *frame* yang cocok akan diikuti melalui urutan untuk mendapatkan *tracking object*

Berikut merupakan algoritma SORT:

1. SORT menggunakan *Faster-RCNN* untuk melakukan deteksi awal per *frame*.

2. Sebelum asosiasi data terdiri dari estimasi model. Ini menggunakan status setiap *tracking* sebagai vektor delapan kuantitas: kotak center *(x, y)*, skala kotak *(s)*, rasio aspek kotak *(a)*, dan turunannya dengan waktu kecepatan. *Kalman Fiter* digunakan untuk memodelkan status ini sebagai dinamis sistem. Jika tidak ada *tracking* deteksi objek untuk *threshold frame* berturut-turut, maka itu dianggap keluar dari *frame* atau hilang. Untuk sebuah kotak yang baru terdeteksi, *tracking* baru dimulai.

3. Pada langkah terakhir, diberikan kondisi prediksi dari *Kalman Filter*, seperti asosiasi dibuat untuk deteksi baru dengan *tracking* objek lama di *frame* sebelumnya. Komputer akan menggunakan *algoritma Hungarian* pada pencocokan grafik bipartit. *Algoritma Hungarian* adalah salah satu dari beberapa teknik-teknik pemecahan yang tersedia untuk masalah-masalah penugasan dengan cara menemukan pemasangan sempurna [27].

2.7.1 Deteksi

Dengan memanfaatkan kemajuan deteksi berbasis CNN, digunakan metode FrRCNN *(Faster Region CNN)* yang lebih cepat. FrRCNN adalah *end-to-end framework* yang terdiri dari dua tahap. Tahap pertama mengekstrak fitur dan mengusulkan daerah untuk tahap kedua yang kemudian menentukan objek di wilayah yang diusulkan. Keuntungan dari *framework* ini adalah bahwa parameter dibagi antara keduanya tahapan menciptakan kerangka kerja yang efisien untuk deteksi. Selain itu, arsitektur jaringan itu sendiri dapat ditukar dengan apa saja desain yang memungkinkan eksperimen cepat berbagai arsitektur untuk meningkatkan kinerja deteksi [28].

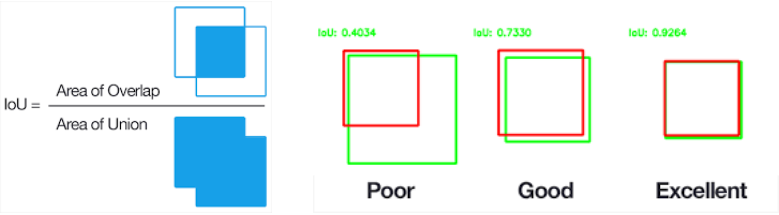
2.7.2 Estimasi Model

Model objek digambarkan menjadi dua, yaitu representasi dan model gerak yang digunakan untuk menyebarkan identitas target ke dalam *frame* selanjutnya. Perpindahan antar *frame* dari setiap objek dengan model kecepatan konstan linier yang tidak tergantung pada objek lain dan gerakan kamera. Keadaan setiap target dimodelkan sebagai:

$$x = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^T \tag{2.16}$$

di mana u dan v mewakili lokasi piksel horizontal dan vertikal dari pusat target, sedangkan skala s dan r mewakili skala (area) dan rasio aspek masing-masing *bounding box* target. Perhatikan bahwa rasio aspek dipertimbangkan menjadi konstan. Ketika deteksi tersebut dikaitkan dengan target, maka *bounding box* yang terdeteksi digunakan untuk memperbarui status target di mana komponen kecepatan diselesaikan secara optimal melalui *Kalman Filter* [29]. Jika tidak ada deteksi yang dikaitkan dengan target, statusnya hanya diprediksi tanpa koreksi menggunakan model kecepatan linier.

2.7.3 Asosiasi Data



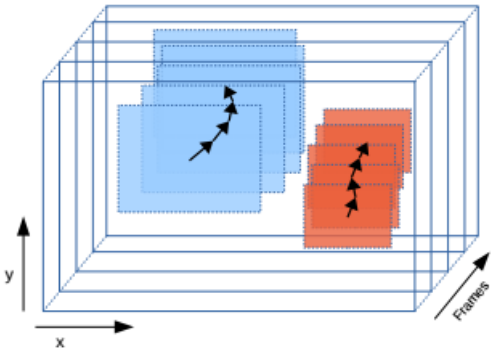
(a) **(b)**

Gambar 2.10: *Intersection Over Union*[[9]]

Dalam menetapkan deteksi ke target yang ada, masing-masing target geometri dari *bounding box* diperkirakan dengan memprediksi lokasi yang baru dalam *frame* saat ini. Matriks biaya penugasan kemudian dihitung sebagai jarak IOU *(Intersection Over Union)* antara setiap deteksi dan semua kotak yang terikat kemudian diprediksi dari target yang ada. Tugas diselesaikan secara optimal meng-

gunakan *algoritma Hungarian*. Selain itu, minimal IOU dikenakan untuk menolak penugasan dimana deteksi target *overlap* kurang dari IOU_{min}.

Jarak IOU dari *bounding box* secara implisit menangani penyumbatan jangka pendek yang disebabkan oleh target yang lewat. Khususnya, ketika target ditutupi oleh objek, hanya occluder yang terdeteksi, karena jarak IOU tepat mendukung deteksi dengan skala serupa. Hal ini memungkinkan target okluder untuk dikoreksi dengan deteksi sementara target yang dicakup tidak terpengaruh karena tidak ada tugas terbuat.



Gambar 2.11: IOU *Tracker*[[10]]

2.7.4 Pembuatan dan Penghapusan ID

Identitas atau ID unik perlu dibuat, maka ketika objek masuk dan meninggalkan gambar. *Tracking* menganggap deteksi apa pun dengan *overlaps* kurang dari IOU_{min} untuk menandakan keberadaan objek yang tidak terlacak. *Tracking* diinisialisasi menggunakan geometri *bounding box* dengan kecepatan diatur ke 0. Karena kecepatannya tidak teramati pada titik ini, komponen kecepatan diinisialisasi dengan nilai besar, mencerminkan ketidakpastian ini. Selain itu, *Tracking* baru kemudian mengalami percobaan di mana target harus dikaitkan dengan deteksi dan mengumpulkan cukup bukti untuk mencegah *tracking* palsu. *Tracking* dihentikan jika tidak terdeteksi untuk T_{Lost} *frame*. Ini mencegah pertumbuhan

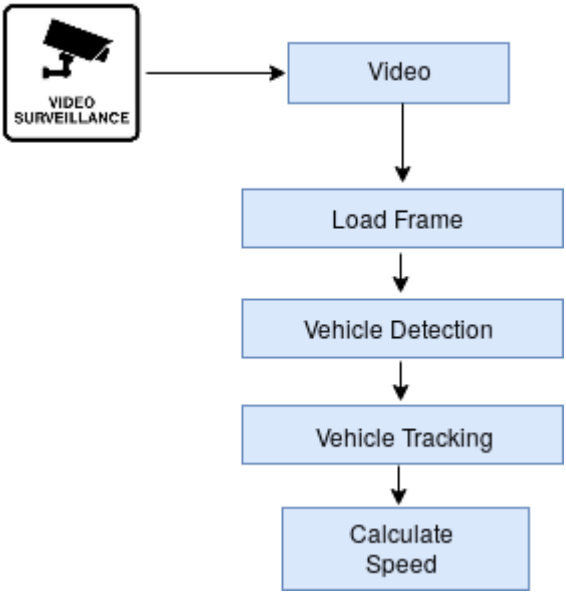
jumlah yang tidak terbatas pelacak dan kesalahan pelokalan yang disebabkan oleh prediksi berakhir dalam jangka waktu lama tanpa koreksi dari detektor. Dalam semua Eksperimen T_{Lost} diatur ke 1 karena dua alasan. Pertama, model kecepatan konstan adalah prediktor yang buruk dari dinamika yang sebenarnya. Kedua, memperhatikan *frame-to-frame* melacak tempat identifikasi ulang objek berada di luar cakupan ini. Selain itu, penghapusan dini dari target yang hilang membantu kekurangan. Jika suatu objek muncul kembali, *tracking* akan secara implisit melanjutkan dengan ID baru [28].

BAB 3
DESAIN DAN IMPLEMENTASI SISTEM

Penelitian ini dilaksanakan sesuai dengan desain sistem berikut dengan implementasinya. Desain sistem merupakan konsep dari pembuatan dan perancangan infrastruktur dan kemudian diwujudkan dalam bentuk blok-blok alur yang harus dikerjakan.

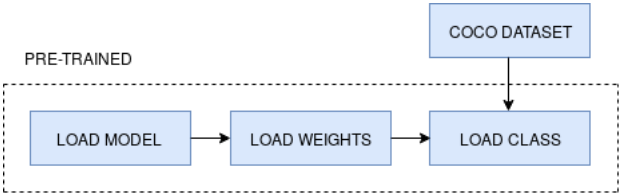
3.1 Desain Sistem

Tugas akhir ini merupakan salah satu bentuk dari pemanfaatan *deep learning* untuk mengukur kecepatan kendaraan yang bertujuan untuk mendapatkan data kecepatan guna memantau kondisi arus lalu lintas pada suatu jalan. Proses kerja dari sistem ditunjukkan pada 3.1.



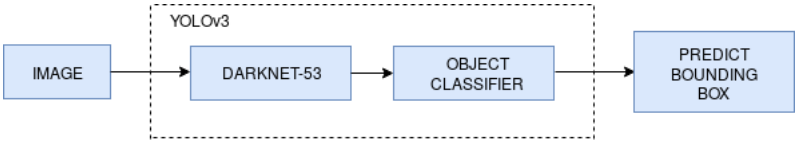
Gambar 3.1: Diagram alir sistem pengukuran kecepatan kendaraan

Berdasarkan diagram alir pada 3.1, CCTV (*Close Circuit Television*) memiliki IP *(Internet Protocol)* yang dapat mengirimkan data gambar secara langsung kemudian tangkapan gambar tersebut direkam menjadi video sehingga *input* video dari CCTV tersebut dapat diekstrak menjadi *frame*. *Frame* yang telah diekstrak akan dibaca satu per satu oleh program, jika terdapat objek-objek yang sudah di*training* oleh *pre-trained* maka objek kemudian akan terdeteksi dengan *bounding box* menggunakan model YOLOv3. *Kalman filter* dan SORT *(Simple Online and Realtime Tracking)* digunakan untuk melakukan *tracking* pada objek yang sudah terdeteksi yang mana hasil *tracking* tersebut digunakan untuk pengukuran kecepatan kendaraan.



Gambar 3.2: Diagram alir proses *pre-trained*

Proses *pre-trained* dapat dilihat pada gambar 3.2, dimana melewati 3 tahapan yaitu *load model* dari *darknet-53*, *load weights* dari YOLOv3 dan yang terakhir *load class* dari COCO dataset yang sudah dilakukan *training*. Terdapat 80 objek yang terdeteksi oleh YOLOv3 menggunakan COCO dataset yang terdiri dari 80 *classification*. Dalam sistem ini, hanya 4 objek yang akan terdeteksi dengan 4 *class* yang termasuk kedalam jenis kendaraan *(vehicle detection)*, yaitu *car*, *motorbike*, *bus* dan *truck*. Selain dari 4 *class* tersebut maka sistem tidak akan mendeteksi adanya objek lain.



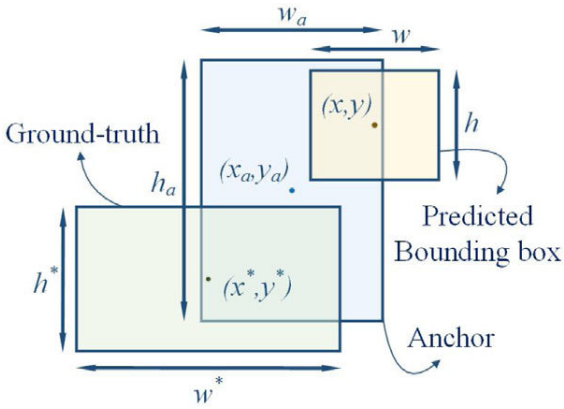
Gambar 3.3: Diagram alir metodologi *object detection*

Pada gambar 3.3 menunjukan gambar diagram alir metodologi *object detection*. *Input* data dari video akan dibaca per-*frame* kemudian *image* pada *frame* yang merupakan objek akan diolah menggunakan YOLOv3 yang didalamnya terdapat model arsitektur *darknet-53*. *Darknet-53* memiliki 53 *layer* baru yang digunakan untuk menggantikan *darknet-19* sebagai fitur ekstrator. *Darknet-53* terutama terdiri dari *filter* 3×3 dan 1×1 dengan melewati koneksi seperti jaringan residual di *ResNet*. *Darknet-53* memiliki BFLOP lebih sedikit (operasi miliar *floating point*) daripada *ResNet-152*, tetapi mencapai akurasi klasifikasi yang sama lebih cepat 2x.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Gambar 3.4: *Darknet-53*[[5]]

Object classifier dibagi menjadi dua tipe, YOLO dan *tiny* YOLO. Tiap tipe memiliki keunggulan sendiri-sendiri dari segi akurasi, kecepatan dan ukuran model. Tipe YOLO memiliki 3 *output* untuk mendeteksi skala yang berbeda. Jika menggunakan ukuran *input* standar sebesar 416 *output* pertama untuk mendeteksi ukuran besar dengan *feature map* 13 X 13 memiliki parameter akhir 1024, kedua untuk mendeteksi ukuran sedang dengan *feature map* 26 X 26 memiliki parameter 512 dan ketiga untuk mendeteksi ukuran besar dengan *feature map* 52 X 52 memiliki parameter 256. Layer yang *output* yang dimiliki YOLO lebih banyak dari *tiny* YOLO. Tipe *tiny* YOLO memiliki 2 *output* untuk mendeteksi skala yang berbeda. Jika menggunakan ukuran *input* standar sebesar 416 *output* pertama untuk mendeteksi ukuran besar dengan *feature map* 13 X 13 memiliki parameter akhir 512, kedua untuk mendeteksi ukuran sedang dengan *feature map* 26 X 26 memiliki parameter 256.



Gambar 3.5: *Predicted Bounding Box*[[11]]

Output dari prediksi *bounding box* berisi informasi mengenai nilai x , y , w , h dan *confidence*. Nilai (x, y) koordinat mempresentasikan titik pusat objek berdasarkan *grid cell*. Nilai (w, h) merupakan panjang dan lebar yang relatif pada ukuran gambar. Ukuran *bounding box* ditentukan dengan mengelompokkan ukuran label kotak menjadi beberapa kelompok. Dikarenakan ada 3 skala dan tiap skala ada 3 *anchor*, jadi jumlahnya 9 *anchor*.

Gambar 3.6: Diagram alir metodologi *tracking*

MOT atau *multiple object tracking* adalah sebuah proses melacak pergerakan satu atau lebih objek di dalam suatu *frame* video. Objek yang dilacak bermacam-macam menurut konteks sistem ini. Di antaranya kendaraan seperti *car, motorbike, bus* dan *truck*. *Multiple object tracking* memiliki beberapa tahapan dalam prosesnya. Untuk setiap metode *multiple object tracking* yang berbeda, tahapan yang dilakukan pun bisa berbeda. Walaupun begitu, secara umum, tahapan dari proses yang dilakukan adalah sebagai berikut:

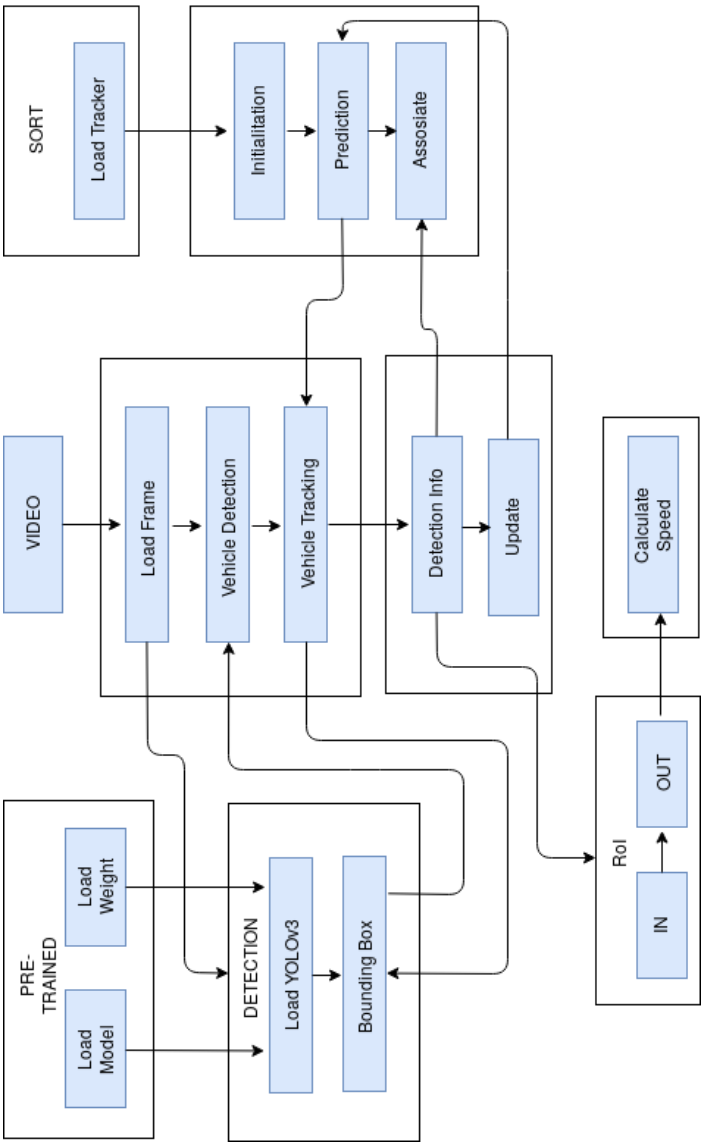
(a) Penugasan ID berbasis *centroid*
Dalam bentuknya yang paling sederhana, ID dapat ditetapkan dengan melihat *centroid* dari *bounding box*. Selanjutnya, menghitung centroid untuk setiap *bounding box* pada *frame* ke 1. Pada *frame* ke 2, *centroid* baru berdasarkan jarak dari *centroid* sebelumnya dapat ditetapkan ID dengan melihat jarak relatif. Asumsi dasarnya adalah *frame to frame centroid* hanya akan bergerak sedikit demi sedikit. Pendekatan sederhana ini bekerja dengan baik selama *centroid* berjarak satu sama lain. Pendekatan ini gagal ketika ada objek yang bersamaan sehingga memungkinkan terjadinya pertukaran ID.

(b) *Kalman Filter*
Kalman Filter adalah peningkatan dari *tracking* berbasis *centroid* sederhana. *Kalman Filter* memungkinkan untuk memodelkan *tracking* berdasarkan posisi dan kecepatan suatu objek dan memprediksi di mana itu mungkin. Ini memodelkan posisi dan kecepatan masa depan menggunakan *gaussians*. Saat menerima *frame* selanjutnya, maka dapat menggunakan probabilitas untuk menetapkan pengukuran ke prediksi dan memperbarui sendiri. Dan karena menggunakan posisi dan kecepatan gerak, maka memiliki hasil yang lebih baik daripada pelacakan berbasis *centroid*.

(c) SORT atau *(Simple Online and Realtime Tracking)*
SORT adalah pendekatan pragmatis untuk pelacakan banyak objek dengan fokus pada algoritma sederhana dan efektif yang merupakan pengembangan dari *Kalman Filter* dengan Algoritma *Hungarian*, karena ekstensi ini, objek dapat dilacak melalui periode oklusi yang lebih lama, mengurangi jumlah sakelar identitas secara efektif. SORT memiliki banyak kompleksitas komputasi ke dalam tahap pra-pelatihan *offline* di mana mempelajari metrik asosiasi yang mendalam pada dataset identifikasi ulang dalam skala besar. Selama *tracking online*, SORT membangun asosiasi pengukuran untuk melacak menggunakan pertanyaan tetangga terdekat pada *frame*. Evaluasi eksperimental menunjukkan bahwa ekstensi SORT dapat mengurangi jumlah pertukaran identitas.

Setelah penjelasan panjang diatas, berikut merupakan diagram sistem metodologi secara keseluruhan (gambar 3.7) terlihat bahwa terdapat tiga proses yang berjalan dalam satu sistem yang menjelaskan dengan detail alur sistem yang berjalan dari *input* video yang awalnya sudah melalui proses *pre-trained* kemudian proses deteksi sampai dengan proses *tracking* yang menghasilkan *output* berupa kecepatan kendaraan.

Pada tahapan proses *detection info*, berisikan informasi seperti *label class* dari objek yang sudah terdeteksi, dan nomor id. Program akan melakukan *tracking* dan melakukan *update* sesuai dengan objek yang terdeteksi pada *frame* hingga objek sudah keluar dari *frame*. *Update* yang dilakukan dalam hal ini adalah posisi objek yang secara terus menerus bergerak dari *frame* satu ke *frame* berikutnya, sehingga setelah objek keluar dari *frame* nomor id yang diberikan saat awal akan bertambah. Pada tahap RoI, nilai *frame in* dan *frame out* dari objek yang terdeteksi akan tercatat pada sistem untuk selanjutnya dimasukan kedalam rumus pengukuran kecepatan, sehingga diperoleh estimasti kecepatan pada sistem.



Gambar 3.7: Diagram alir metodologi sistem secara keseluruhan

3.2 Alur kerja

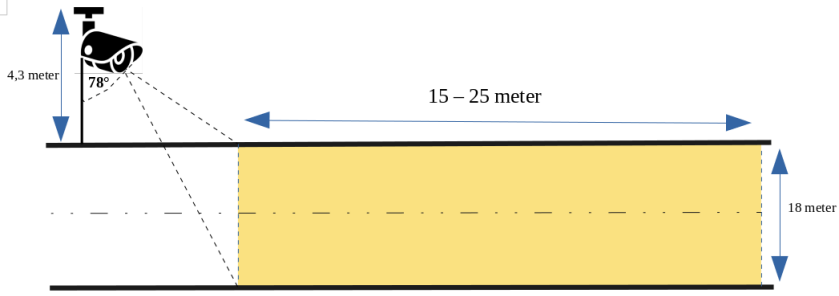
Pengerjaan tugas akhir ini dibagi menjadi beberapa tahapan berdasarkan metodologi penelitian, yaitu :

1. Akusisi Data.
2. *Vehicle Detection.*
3. *Vehicle Tracking.*
4. *Vehicle Speed.*

3.3 Akusisi Data

Pada tahap akuisisi data, terdapat data video yang di proses dalam penelitian ini yaitu data video yang di tangkap dari posisi samping objek.

3.3.1 Persiapan dan Peletakan Kamera



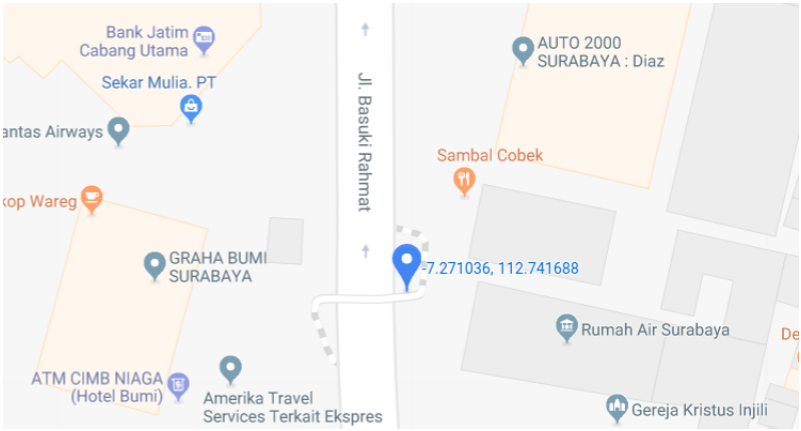
Gambar 3.8: Skema Peletakan CCTV

Akusisi data video pada penelitian ini menggunakan CCTV yang dipasang di pinggir jalan atau di samping objek. Berdasarkan pada gambar 3.8, CCTV dipasang pada ketinggan 4.3 meter, dimana 2.5 meter merupakan tinggi dari anak tangga JPO (Jembatan Penyebrangan Orang) dan 1.8 merupakan tinggi dari tripod kamera serta membentuk sudut kemiringan sebesar 78°. Cakupan luasan jalan dengan lebar 18 meter dapat menangkap RoI *(Region Of Interest)* dengan panjang 15-25 meter.

Gambar 3.9: Gambar Peletakan CCTV

Gambar 3.10: Pengukuran sudut kemiringan kamera

3.3.2 Lokasi Akuisisi Data



Gambar 3.11: Lokasi Akuisisi Data

Lokasi akuisisi data berada di Jalan Basuki Rahmat, Kota Surabaya. Berada di sepanjang jalan depan gedung Graha Bumi dan Rumah Air Surabaya dapat dilihat pada gambar 3.11 dengan *latitude* di koordinat -7.271036 dan *longtitude* di koordinat 112.741688 pada *Google Maps*. Dengan pertimbangan berikut yang dijelaskan dibawah ini, menurut penulis merupakan lokasi yang strategis untuk akuisisi data.

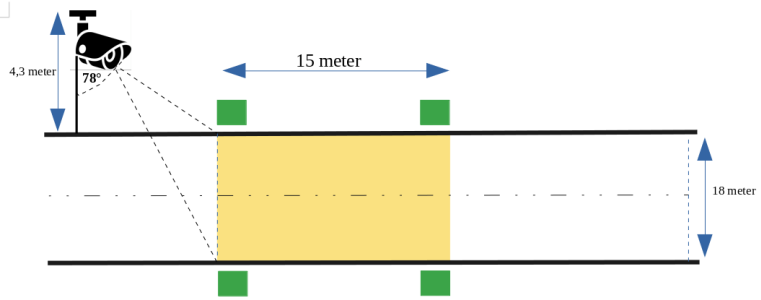
Jalan Basuki Rahmat dapat dilihat pada gambar 3.12 merupakan jalan satu arah yang berada di arteri atau jantung pusat Kota Surabaya yang merupakan jalan akses menuju gedung - gedung perkantoran, gedung - gedung pariwisata seperti hotel, pusat perbelanjaan modern (mall), dan *restaurant* makan cepat saji (*fast food*) serta swalayan mini sehingga ada banyak kendaraan yang melewati jalan ini setiap harinya. Berbagai jenis kendaraan juga diperbolehkan untuk melintas di dijalan ini, seperti mobil, motor, truk dan bus menyebabkan tingkat kepadatannya arus lalu lintas tergolong fluktuatif dengan berbagai kecepatan kendaraan yang melaju menyesuaikan dengan kondisi arus lalu lintas sehingga sangat cocok untuk melakukan akuisisi data.

Pada lokasi ini terdapat JPO (Jembatan Penyebrangan Orang) yang memudahkan pengambilan gambar karena memanfaatkan ketinggiannya agar hasil tangkapan video dapat menjangkau untuk merekam daerah RoI dengan jarak yang sudah ditentukan, sepanjang 15-25 meter. Dengan lebar jalan sepanjang 18 meter dan memiliki tiga ruas jalur jalan sehingga mendukung untuk pengambilan data video kendaraan yang melewati jalan ini.

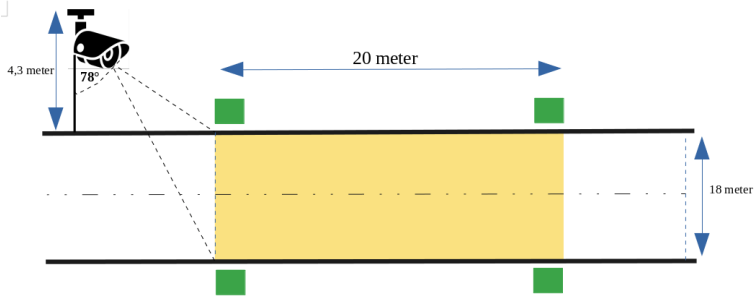
Pengambilan data video dilakukan pada waktu pagi, siang, sore, dan malam hari dengan kondisi cuaca cerah dalam hal ini yang dimaksdkan adalah tidak sedang turun hujan. Pengambilan data video pada pagi hari dilakukan pada pukul 10.15 WIB, pada siang hari dilakukan pada pukul 13.49 WIB, pada sore hari dilakukan pada pukul 15.38 WIB dan pada malam hari dilakukan pada pukul 18.08 WIB. Pada waktu malam hari, kondisi Jalan Basuki Rahmat memiliki lampu penerangan yang sangat mendukung untuk pengambilan data.

Gambar 3.12: Jalan Basuki Rahmat

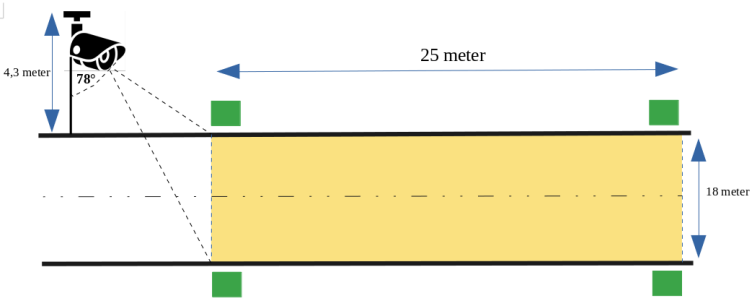
3.3.3 Pemasangan Penanda RoI



Gambar 3.13: Ilustrasi Penanda RoI berjarak 15 meter



Gambar 3.14: Ilustrasi Penanda RoI berjarak 20 meter



Gambar 3.15: Ilustrasi Penanda RoI berjarak 25 meter

RoI *(Region of Interest)* digunakan sebagai acuan titik awal dan titik akhir dalam proses perhitungan kecepatan kendaraan. RoI diletakkan sepasang di pinggir jalan dengan jarak yang bervariasi 15 meter seperti pada gambar 3.13, 20 meter seperti pada gambar 3.14, dan 25 meter seperti pada gambar 3.15. Ketiga jarak RoI yang bervariasi tersebut digunakan sebagai variabel pembanding dalam pengujian sistem ini.

Pada gambar 3.16, penanda RoI dibuat dari kertas asturo berjumlah empat buah bewarna hijau yang diletakkan di pinggir jalan seperti pada gambar 3.17. Dua disisi jalan kiri dan dua disisi jalan kanan. masing-masing sesuai dengan jarak yang ditentukan.

Pemasangan penanda RoI dimaksudkan untuk mendapatkan waktu tempuh dari kendaraan yang melewati jalan tersebut. Wak-

Gambar 3.16: Bentuk Penanda ROI

Gambar 3.17: Pemasangan penanda tampak dekat

tu tempuh dalam objek yang bergerak di dalam sebuah tangkapan video didapatkan dari jumlah *frame* yang melewatinya. Jumlah *frame* dapat dihitung dengan melakukan selisih pada *frame* yang masuk di titik awal dan *frame* yang keluar di titik akhir.

Pada gambar 3.18, 3.19 dan 3.15, penanda RoI yang telah terpasang di pinggir jalan (dapat dilihat dari objek yang dilingkari) dan diletakkan sesuai dengan jarak yang ditentukan. Jarak ini juga digunakan dalam proses perhitungan kecepatan kendaraan. Adapun

untuk menghitung kecepatan kendaraan dapat dilihat pada persamaan 2.1.

Gambar 3.18: Pemasangan Penanda RoI berjarak 15 meter

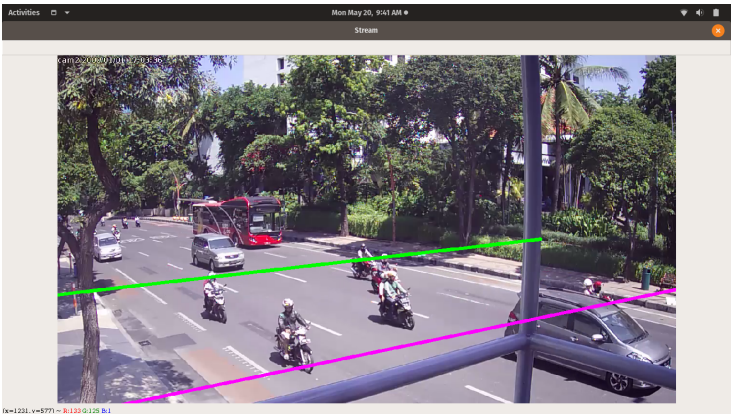
Gambar 3.19: Pemasangan Penanda RoI berjarak 20 meter

Gambar 3.20: Pemasangan Penanda RoI berjarak 25 meter

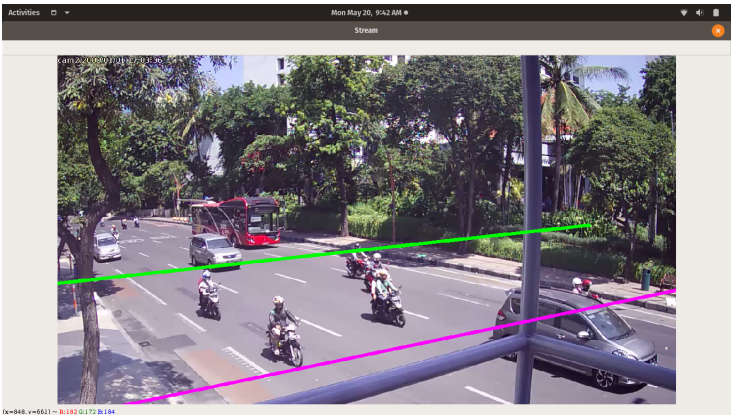
Penanda RoI dapat digunakan sebagai alat bantu untuk membuat *line* yang mana akan *intersect* dengan objek yang terdeteksi sehingga dapat mendapatkan nilai acuan titik awal ataupun akhir. *Intersect* ibarat dua segmen garis AB, dan CD. Ini berpotongan jika dan hanya jika titik A dan B dipisahkan oleh segmen CD dan titik C dan D dipisahkan oleh segmen AB. Jika titik A dan B dipisahkan oleh segmen CD maka ACD dan BCD harus memiliki orientasi berlawanan yang berarti baik ACD atau BCD berlawanan arah jarum jam tetapi tidak keduanya. Oleh karena itu, pada program, terdapat dua *line* yang akan tergambar dengan bantuan dari librari OpenCV sehingga memudahkan visualisasi gambar jika objek sedang terpantau melewati RoI.

Pada gambar 3.21, RoI dibuat dengan koordinat [(0, 495), (1000, 380)] dan [(0, 750), (1300, 480)] untuk jarak 15 meter. Pada gambar 3.22, RoI dibuat dengan koordinat [(0, 470), (1100, 350)] dan [(0, 750), (1300, 480)] untuk jarak 20 meter. Pada gambar 3.23, RoI dibuat dengan koordinat [(0, 445), (850, 350)] dan [(0, 750), (1300, 480)] untuk jarak 25 meter.

Dari koordinat tersebut akan ditarik garis dan terbentuk sebuah *line*. *Line* yang berwarna hijau merupakan acuan untuk titik

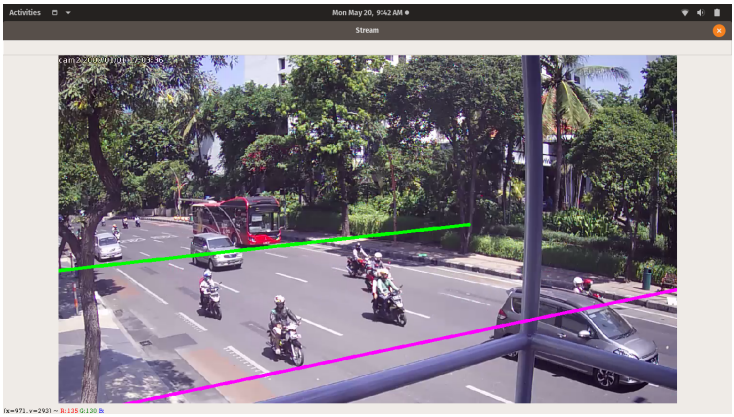


Gambar 3.21: RoI berjarak 15 meter pada program



Gambar 3.22: RoI berjarak 20 meter pada program

awal yang berubah-ubah sesuai dengan jarak yang ditentukan yaitu 15 meter atau 20 meter atau 25 meter, sedangkan *line* yang berwarna ungu merupakan acuan titik akhir yang memiliki koordinat tetap.



Gambar 3.23: RoI berjarak 25 meter pada program

3.3.4 Spesifikasi CCTV

Berdasarkan gambar 2.1, CCTV (*Close Circuit Television*) yang digunakan adalah Vivotek IB8354-C, Dalam penelitian ini, konfigurasi jumlah *frame rate* yang digunakan yaitu 30 fps *(frame per second)* dengan resolusi HD (1280 x 720 piksel).

Tabel 3.1: Spesifikasi Kamera CCTV Vivotek IB8354-C

SPESIFIKASI VIVOTEK IB8354-C	
System Information	
CPU	Multimedia SoC (System-on-Chip)
Flash	128 MB
RAM	256 MB
Feature	
Image sensor	1/3âĂİ Progressive CMOSâĂİ
Maximum Resolution	1280x1024 (1.3MP)
Lens type	Fixed-focal
Focal length	f = 4 mm
Shutter time	1/5 sec. to 1/32,000 sec
General connectors	RJ-45 for PoE connection
Video compression	H.264 and MJPEG
Maximum frame rate	30 fps @ 1280x1024
System Requirements	
Operating system	Microsoft Windows 7/vista/XP/2000, MAC OS
Browser	Mozilla Firefox 7âĹij10 and Internet Explorer 7/8/9/10

Gambar 3.24: Kamera Vivotek IB8354-C[[1]]

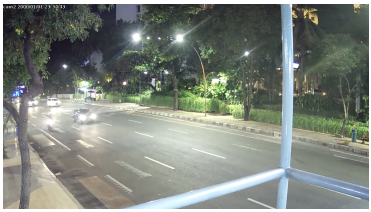
3.3.5 Pengambilan Data Video

Pengambilan video dilakukan pada waktu pagi, siang, sore dan malam hari dengan beberapa parameter sesuai kebutuhan untuk pengujian. Pada gambar 3.25, pengambilan video dilakukan pada pagi hari pukul 10.15 WIB. Pada gambar 3.26 dilakukan pada waktu siang hari pukul 13.49 WIB. Pada gambar 3.26 dilakukan pada waktu sore hari pukul 15.38 WIB. Pada gambar 3.28 dilakukan pada waktu siang hari pukul 18.08 WIB. Pengambilan video membutuhkan penanda yang sudah terpasang sebagai RoI. Setelah itu dilakukannya perekaman video dilakukan selama satu menit untuk proses pengujian. Ukuran resolusi yang diajukan adalah 1280x720 piksel dengan kecepatan merekam standar sebesar 30 fps *(frame per second)*.

Gambar 3.25: Pengambilan Video pada waktu pagi hari

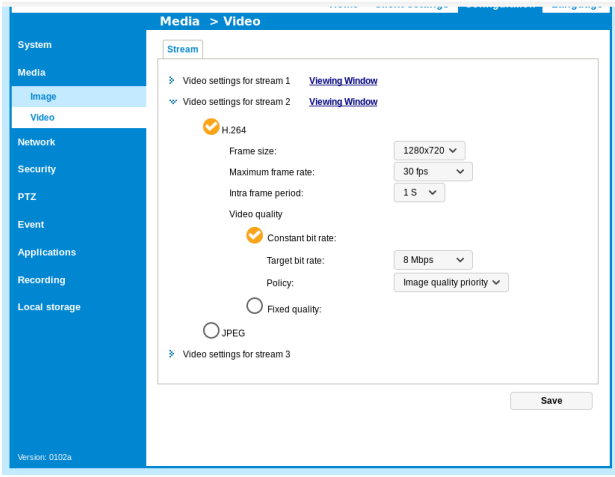
Gambar 3.26: Pengambilan Video pada waktu siang hari

Gambar 3.27: Pengambilan Video pada waktu sore hari



Gambar 3.28: Pengambilan Video pada waktu malam hari

Perekaman video juga dilakukan dengan beberapa parameter lain untuk dilakukan uji coba, seperti perbedaan fps *(frame per second)* pada kamera CCTV dengan jumlah 20 fps, 25 fps dan 30 fps. Pada gambar 3.29, pengaturan fps pada kamera CCTV dapat dilakukan pada web lokal vivotek yang diakses menggunakan IP lokal yang sudah terkonfigurasi. Selain itu, ada parameter lain seperti perbedaan jarak RoI sepanjang 15 meter, 20 meter dan 25 meter dalam melakukan perekaman data. Kedua parameter ini, yaitu perbedaan jumlah *frame rate* dan perbedaan jarak RoI diambil pada waktu pagi hari dengan durasi perekaman video selama satu menit.



Gambar 3.29: Pengaturan FPS pada kamera CCTV

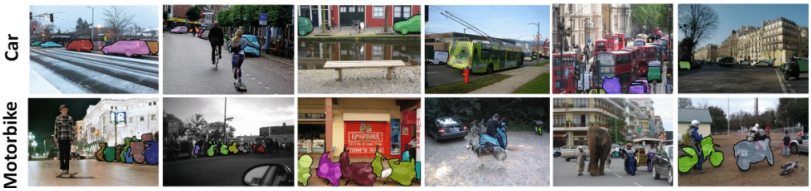
3.4 *Vehicle Detection*

Vehicle Detection merupakan tahap *detection* untuk menemukan objek di dalam *frame* video. Objek yang dicari tergantung pada jenis objek tersebut. Apabila objek yang dicari sudah diketahui, maka objek tersebut terdeteksi yang mana hanya kendaraan saja yang akan dideteksi. Proses pengenalan objek menggunakan metode *convolutional neural network* dan proses deteksi menggunakan YOLOv3 *(You Only Look Once)*. Ada beberapa tahapan untuk mendeteksi objek menggunakan YOLOv3, yaitu *prepocessing* dan *bounding box*.

3.4.1 *Preprocessing*

Pada tahap *preprocessing* merupakan tahapan awal sebelum mengolah data *input* memasuki proses tahapan utama. Di sini, dataset dari objek yang akan dideteksi terlebih dahulu dilakukan *pre-trained* agar objek dapat dikenali oleh sistem. Hal ini dilakukan untuk meningkatkan akurasi sistem *multiple object tracking* atau MOT.

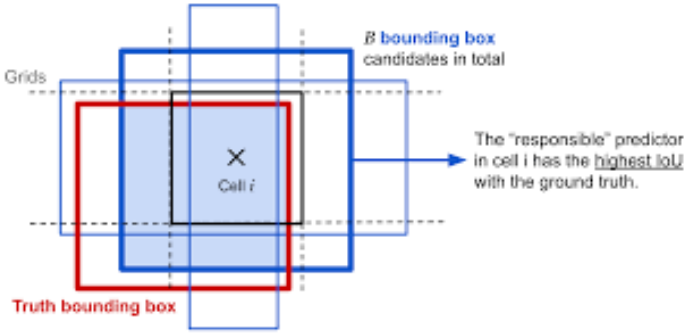
(a) Dataset
COCO *(Common Objects in Context)* dataset dibuat oleh Microsoft berisi 91 kategori objek umum dengan 82 di antaranya memiliki lebih dari 5.000 label. COCO memiliki lebih banyak contoh per kategori. Ini dapat membantu dalam *training* model objek terperinci yang mampu melokalisasi 2D dengan tepat. COCO menyediakan 80 *class* yang siap digunakan dan hanya 4 *class* diantaranya yang digunakan yaitu *car, motorbike, bus* dan *truck*.



Gambar 3.30: Sampel dataset MS COCO[[12]]

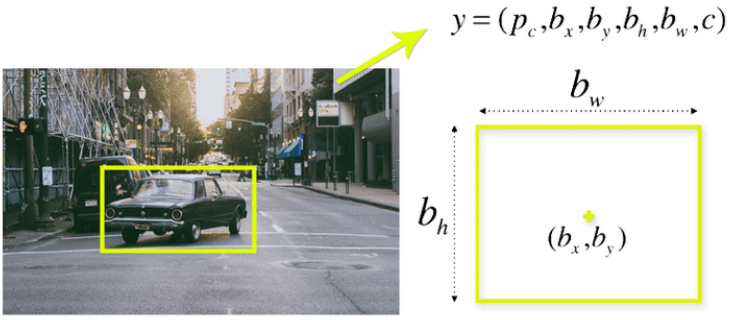
(b) *Pre-trained*
Pre-trained merupakan pembelajaran pengenalan objek. Dataset yang sudah terkumpul (COCO) akan dilakukan *training* dengan *load model* dari *darknet-53* dan *load weight* YOLOv3. *Darknet-53* merupakan model arsitektur yang memiliki *convolution layer* sebanyak 53 lapisan. *Darknet-53* merupakan *framework* yang menerapkan metode CNN (convolutional neural network) untuk *Feature Extractor* yang digunakan oleh YOLOv3. *Load weights* berisi bobot dari masing-masing objek yang sudah dilakukan *training* oleh YOLOv3. Didalamnya merupakan nilai *array* yang berisi lokalisasi objek sesuai dengan *label class* masing-masing.

3.4.2 *Bounding Box*



Gambar 3.31: Ilustrasi Prediksi *Bounding Box*[[5]]

Ukuran gambar *input* standar dari YOLOv3 adalah 416x416. YOLOv3 akan membuat prediksi pada tiga skala, yang secara tepat di- berikan dengan menurunkan dimensi gambar *input* masing-masing sebesar 32, 16 dan 8. Lapisan 13 x 13 untuk mendeteksi *large object*, sedangkan lapisan 52 x 52 mendeteksi *small object*, dan lapisan 26 x 26 mendeteksi *medium object*. Tiap grid cell output akan mendeteksi 3 anchor box yang berbeda. Layer pertama (13X13X3)=507, (26X26X3)=2028,(52X52X3)=8112. Jika di jumlah maka totalnya ada 10,647 kotak yang harus dideteksi.

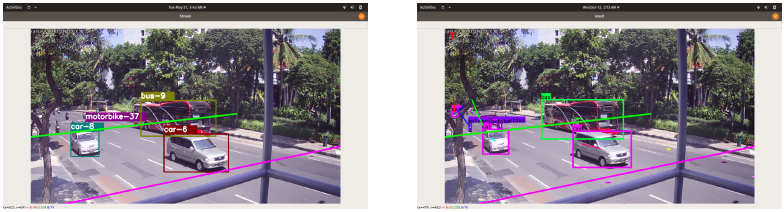


Gambar 3.32: *Bounding Box*[[13]]

Gambar 3.33: Ilustrasi Proses *Non-Max Suppression*[[13]]

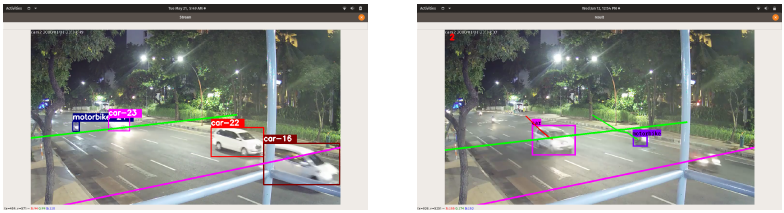
Setiap grid dari YOLOv3 memprediksi 5 nilai yaitu: x , y , w , h dan *confidence* ditambah dengan C (jumlah *class* probabilitas). Nilai x,y koordinat mempresentasikan titik pusat objek berdasarkan *grid cell*. Nilai w,h merupakan panjang dan lebar yang relatif pada ukuran gambar. Terakhir nilai *confidence* yang memprediksi IOU *(Intersection Over Union)* antara *bounding box* prediksi dan *bounding box* yang asli untuk mendeteksi ada atau tidaknya objek pada *grid cell* nilai *confidence*. Karena selisih IOU sedikit kemungkinan semua kotak tersebut mendeteksi objek sama sehingga perlu dihapus. Pada langkah berikutnya dilakukan penghapusan kotak dengan probabilitas objek rendah dan mengikat area kotak dengan

probalitas tertinggi proses ini yang disebut NMS *(non-max suppression)*.



(a) **(b)**

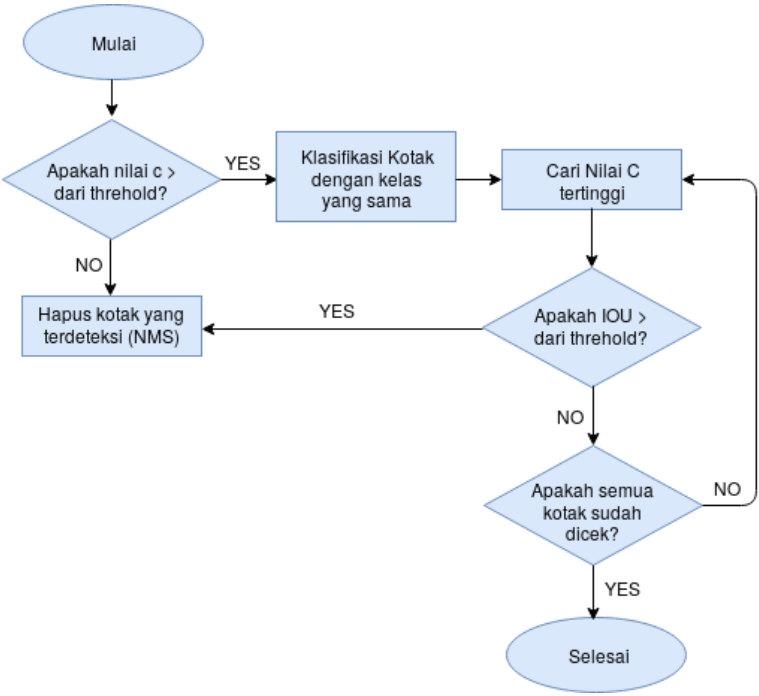
Gambar 3.34: Hasil *bounding box* Kendaraan pada pagi hari



(a) **(b)**

Gambar 3.35: Hasil *bounding box* Kendaraan pada malam hari

Pada gambar 3.31 dan 3.32, objek kendaraan yang sudah terdeteksi, seperti *car, motorbike, bus* dan *truck* oleh YOLOv3 akan di*bounding box* dengan bantuan dari *library OpenCV* untuk menggambar kotak. *Flowchart* dari *bounding box* pada program dapat dilihat pada gambar 3.33. Secara garis besar, alur dari *flowchart* program untuk deteksi *bounding box* merupakan cara kerja dari proses NMS *(non-max suppression)* yang mana membandingkan *treshold* dan nilai dari IOU *(Intersection Over Union)*.



Gambar 3.36: Diagram alir prediksi *Bounding Box*

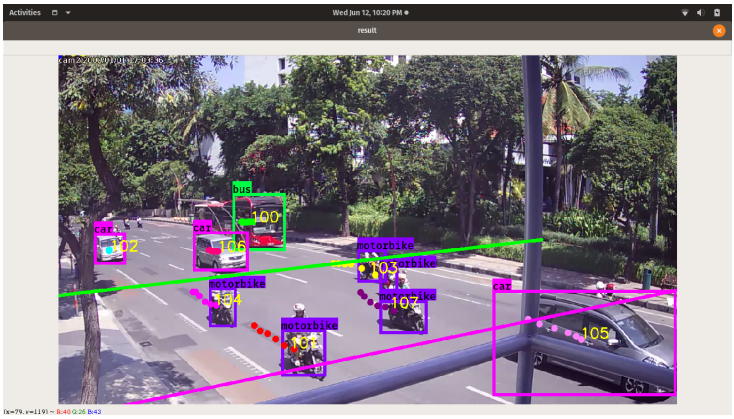
3.5 *Vehicle Tracking*

Pada tahap *vehicle tracking*, setiap objek dalam kumpulan objek terdeteksi dilacak keberadaannya secara terus menerus pada *frame* video selanjutnya. Pada tahap ini juga dilakukan manajemen objek, yaitu objek yang sudah dilacak diperiksa, apakah masih ada di dalam video, sudah menghilang, atau terhalang oleh objek lain. Penambahan ID juga dilakukan dalam tahapan ini untuk memudahkan *tracking* objek. Terdapat beberapa metode yang termasuk ke dalam tahap ini. Di antaranya adalah *Kalman Filter* dan SORT *(Simple Online and Real Time Tracking)*.

3.5.1 *Kalman Filter*

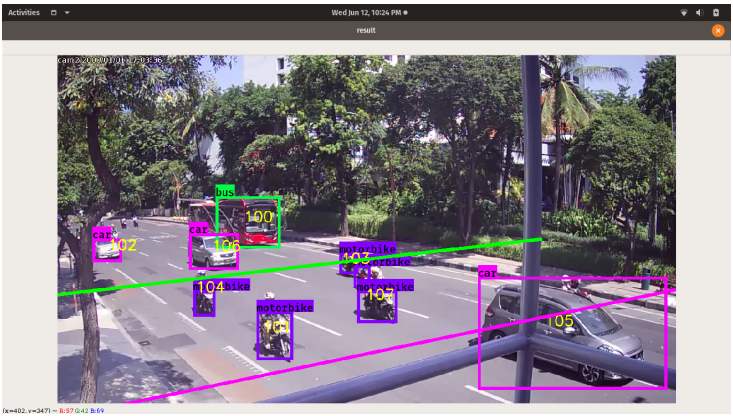
Kendaraan yang sudah dideteksi oleh YOLOv3 akan di-*bounding box*. Dari *bounding box* didapatkan titik tengah atau *centroid* (berwarna merah merupakan titik *centroid* sehingga dapat menetapkan ID dengan cara menghitung *centroid* untuk setiap *bounding box* pada *frame* ke 1. Pada *frame 2*, *centroid* baru berdasarkan jarak dari *centroid* sebelumnya di *frame* ke 1, dapat ditetapkan ID dengan melihat jarak relatif. Asumsi dasarnya adalah *frame to frame centroid* hanya akan bergerak sedikit. Pendekatan sederhana ini menggunakan algoritma dari *Kalman Filter* yang bekerja dengan baik selama *centroid* berjarak satu sama lain jika objeknya sama (satu *class*).

Untuk memudahkan visualisasi *vehicle tracking*, dengan bantuan dari librari OpenCV dibuatlah sebuah garis *line* berwarna merah yang ditarik lurus dari titik centroid *bounding box*. Garis *line* ini akan mengikuti objek yang bergerak dari masuk *frame* hingga keluar *frame*. Selama titik centroid ada di dalam *bounding box*, maka sistem akan terus melakukan *tracking* pada objek.



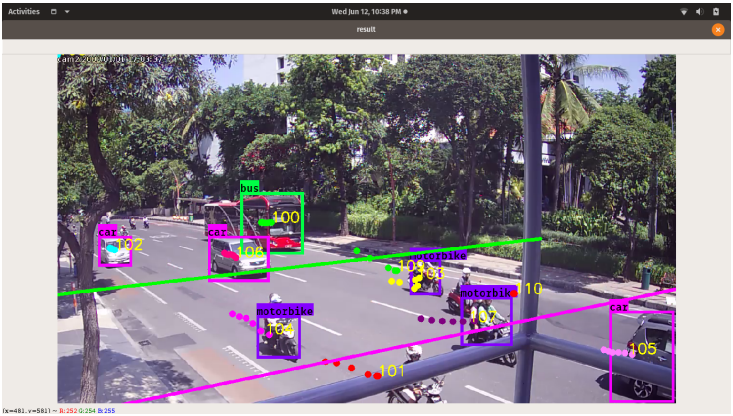
Gambar 3.37: *Tracking* objek menggunakan *Kalman Filter*

Penambahan ID pada *Kalman Filter* diambil dari titik *centroid*. Akan tetapi nilai dari titik *centroid Kalman Filter* selalu berubah-ubah antara disetiap *frame* sehingga pada waktu di *tracking* sebelum dan sesudah nilai kordinat dari setiap objek tidak sa-



Gambar 3.38: Pemberian ID menggunakan *Kalman Filter*

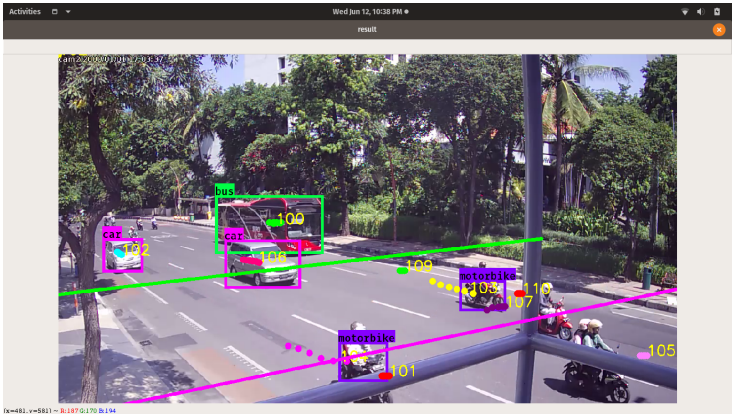
ma dan menyebabkan IDnya berganti dan tidak mengikuti *centroid* dari objeknya.



Gambar 3.39: Objek bergerak melewati RoI

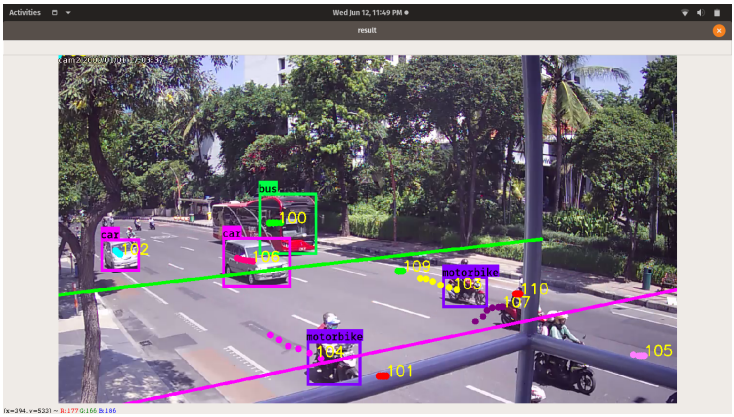
Pada gambar 3.37, objek *motorbike* dengan ID-101 terlacak sedang melintasi jalan dan bergerak menuju garis RoI yang berwarna ungu. Pada gambar 3.39, objek ID-101 sudah keluar dari garis RoI dan pada gambar 3.40, objek keluar dari *frame* tetapi titik *centroid*

dari objek tidak ikut keluar dari *frame*.

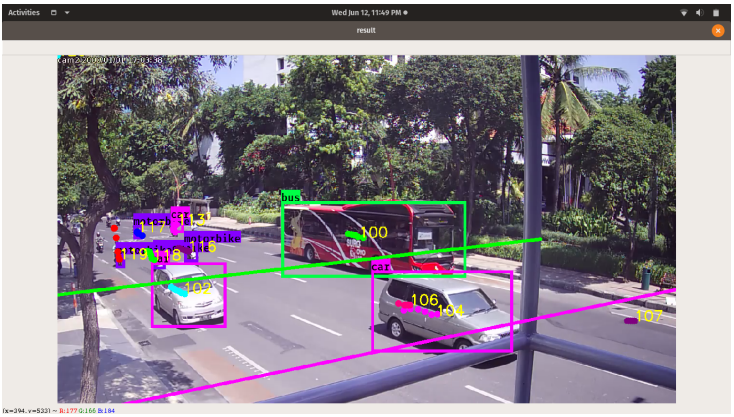


Gambar 3.40: Objek keluar dari *frame*

Pada gambar 3.41, objek ID-106 dengan label *class car* terdeteksi dan terlacak, kemudian pada gambar 3.42 di *frame* selanjutnya dengan objek yang sama ID-106 terlacak berganti ID menjadi ID-104, selanjutnya pada gambar 3.43, objek terlacak berganti ID menjadi ID-107.



Gambar 3.41: Objek ID-106



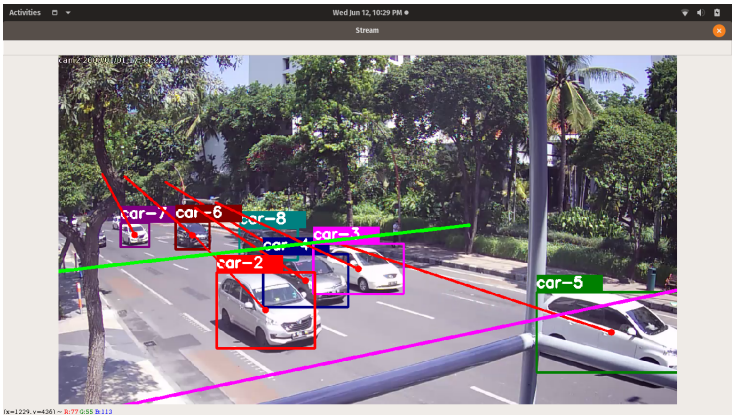
Gambar 3.42: Objek ID-104

Gambar 3.43: Objek ID-107

3.5.2 SORT

SORT atau *Simple Online and Real Time Tracking* merupakan pengembangan dari metode *Kalman Filter* yang mana perpindahan antar *frame* dari setiap objek dengan model kecepatan konstan linier yang tidak tergantung pada objek lain dan gerakan kamera. Keadaan setiap target dimodelkan sebagai $x = [u, v, s, r, u, v,]^T$, di

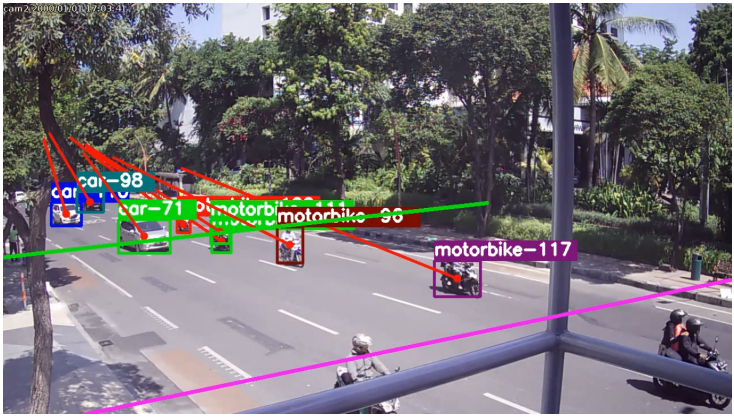
mana u dan v mewakili lokasi piksel horizontal dan vertikal dari pusat target, sedangkan skala s dan r mewakili skala (area) dan rasio aspek masing-masing *bounding box* target. Ketika deteksi dikaitkan dengan target, maka *bounding box* yang terdeteksi digunakan untuk memperbarui status target di mana komponen kecepatan diselesaikan secara optimal melalui *Kalman Filter*.



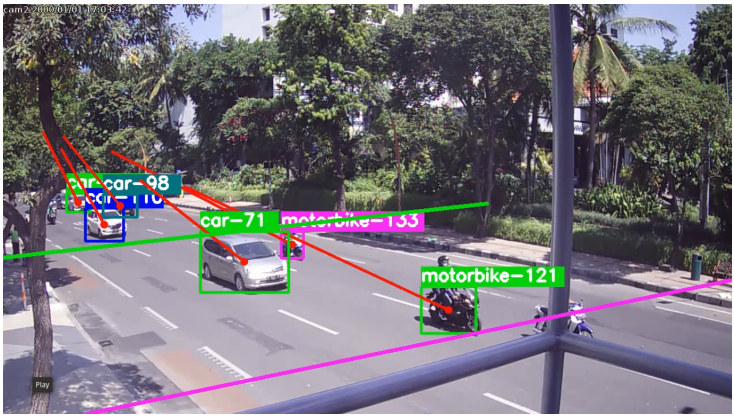
Gambar 3.44: *Tracking* objek menggunakan SORT

Identitas (ID) unik dibuat ketika objek masuk dan meninggalkan gambar. *Tracking* menganggap deteksi apa pun dengan *overlaps* kurang dari IOU_{min} untuk menandakan keberadaan objek yang tidak terlacak. *Tracking* diinisialisasi menggunakan geometri *bounding box* dengan kecepatan diatur ke 0. Karena kecepatannya tidak teramati pada titik ini, komponen kecepatan diinisialisasi dengan nilai besar, mencerminkan ketidakpastian. Selain itu, *Tracking* baru kemudian mengalami percobaan di mana target harus dikaitkan dengan deteksi dan mengumpulkan cukup bukti untuk mencegah *tracking* palsu. *Tracking* dihentikan jika tidak terdeteksi untuk T_{Lost} *frame*. Ini mencegah pertumbuhan jumlah yang tidak terbatas pelacak dan kesalahan pelokalan yang disebabkan oleh prediksi berakhir dalam jangka waktu lama tanpa koreksi dari detektor. Dalam semua Eksperimen T_{Lost} diatur ke 1 karena dua alasan. Pertama, model kecepatan konstan adalah prediktor yang buruk dari dinamika yang sebenarnya. Kedua, memperhatikan *frame-to-frame* melacak tem-

pat identifikasi ulang objek berada di luar cakupan ini. Selain itu, penghapusan dini dari target yang hilang membantu kekurangan. Jika suatu objek muncul kembali, *tracking* akan secara implisit melanjutkan dengan ID baru.



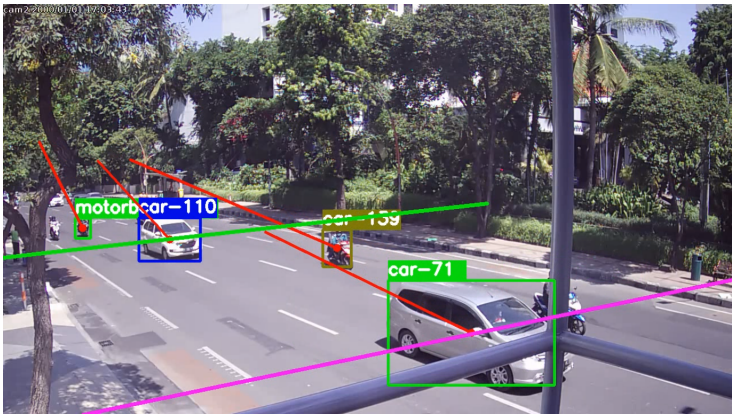
Gambar 3.45: Objek menyentuh *line* hijau



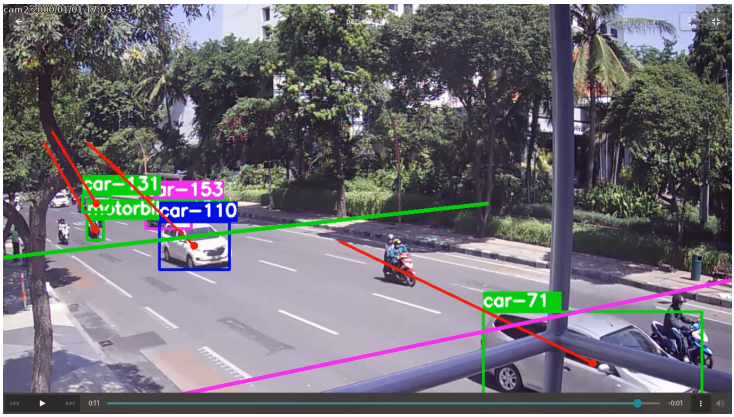
Gambar 3.46: Objek menyentuh masuk ke dalam RoI

Pada gambar 3.45, objek dengan label *class CAR* - ID 71 terlihat menyentuh garis *line* hijau dan masuk ke dalam RoI yang dibuat

(gambar 3.46). Pada gambar 3.47 objek terlihat menyentuk garis *line* ungu dan keluar dari RoI yang telah dibuat (gambar 3.48.



Gambar 3.47: Objek menyentuh *line* ungu



Gambar 3.48: Objek menyentuh ke luar RoI

3.6 *Vehicle Speed*

Pada tahap *vehicle speed*, sistem akan melakukan perhitungan kecepatan kendaraan yang melintas menggunakan persamaan 2.2. Ada 3 parameter *input* yang diperlukan untuk dimasukan kedalam persamaan 2.2. Parameter pertama adalah jarak RoI yang sudah ditentukan yaitu 15 meter, 20 meter atau 25 meter. Parameter kedua adalah selisih nilai *frame-in* dan *frame-out* dari garis yang telah dibuat. Dan parameter ketiga adalah fps (*frame rate per second*) dari kamera CCTV yang digunakan untuk merekam video. Program pada sistem akan melakukan perhitungan secara otomatis setelah semua parameter diatas dimasukan ke dalam persamaan 2.2.

$$Kecepatan = \frac{PanjangGaris \times fps \times 3600}{JumlahFrame \times 1000} km/h$$

3.6.1 Menggunakan *Kalman Filter*

Gambar 3.49: Objek *Motorbike-119* masuk kedalam RoI

Pada gambar 3.49 dan 3.50, *frame-in* diwakilkan dengan garis *line* yang berwarna hijau, sedangkan *frame-out* diwakilkan dengan garis *line* berwarna ungu. Hasil *vehicle spped* dengan menggunakan *Kalman Filter* dapat dilihat pada gambar 3.51, dijelaskan sebagai berikut:

16 merupakan *counting* atau jumlah kendaraan yang melintas.

car merupakan label dari *classification* objek.

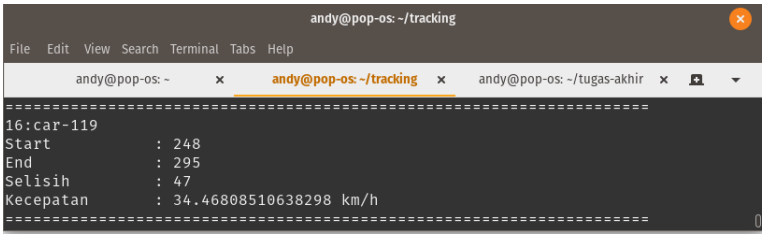
119 merupakan ID dari objek.

Start merupakan nilai dari *frame-in* yang menyentuh garis RoI berwarna hijau.

End merupakan nilai dari *frame-out* yang menyentuh garis RoI berwarna ungu.

Selisih merupakan jumlah keseluruhan *frame* yang dilewati pada jarak RoI yang didapatkan dari selesih nilai *frame-in* dan *frame-out*.

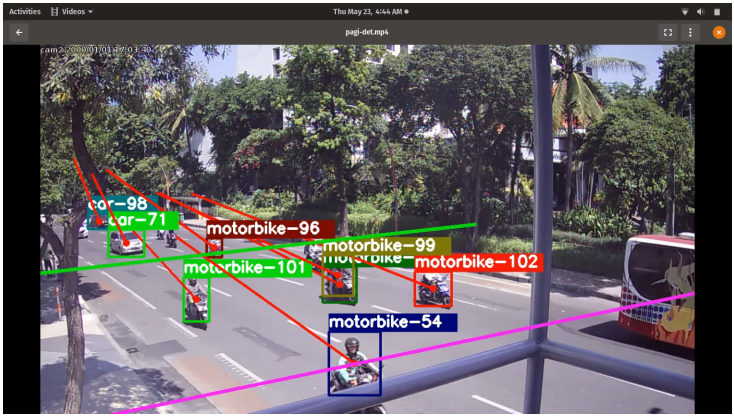
Gambar 3.50: Objek *Motorbike-119* keluar melewati RoI



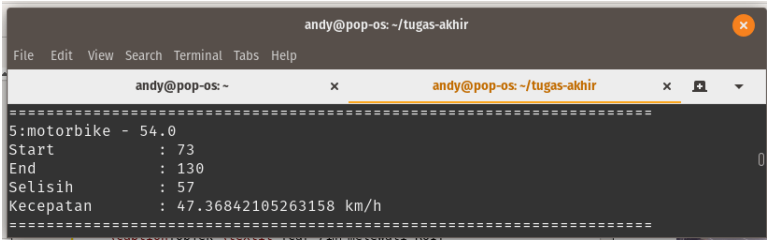
Gambar 3.51: Proses sistem melakukan perhitungan kecepatan kendaraan

3.6.2 Menggunakan SORT

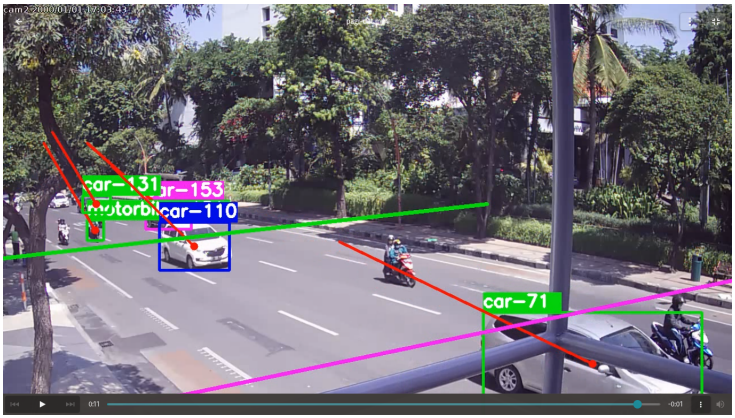
Jika dibandingkan dengan penggunaan *Kalman Filter*, hasil sistem menggunakan SORT dapat menghasilkan hasil estimasi *speed* beserta *class* dari objek dan memiliki ID yang cenderung konstan meskipun ID baru sering muncul ketika *bounding box* dari objek menghilang dan kemudian muncul kembali saat *tracking* sedang berlangsung. Sedangkan, pada *Kalman Filter* menghasilkan estimasi *speed* berserta ID objek dan tidak dapat melakukan *classification* objek dengan tepat meskipun label *classification* pada saat dilakukan *tracking* berjalan baik akan tetapi *output* dari program tidak dapat menentukan *class* dari objek.



Gambar 3.52: Objek *Motorbike-54* melewati RoI

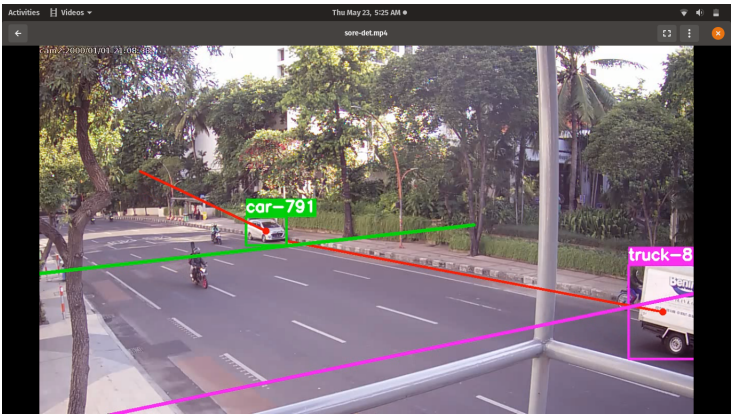


Gambar 3.53: Proses sistem melakukan perhitungan kecepatan kendaraan

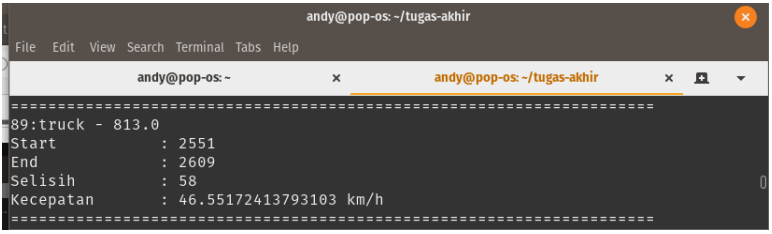


Gambar 3.54: Objek *Car-71* melewati RoI

Gambar 3.55: Proses sistem melakukan perhitungan kecepatan kendaraan



Gambar 3.56: Objek *Truck-813* melewati RoI



Gambar 3.57: Proses sistem melakukan perhitungan kecepatan kendaraan

Pada gambar 3.53, 3.55, dan 3.57 terlihat bahwa hasil dari program menunjukan kesesuaian label *classification* dari objek yang sudah di-*tracking* pada gambar 3.52, 3.54, dan 3.56 melewati garis RoI yang sudah ditentukan.

BAB 4
PENGUJIAN DAN ANALISA

Pada bab ini dipaparkan hasil pengujian serta analisa dari desain sistem dan implementasi. Data yang digunakan dalam pengujian program diperoleh dengan merekam video arus lalu lintas di Jalan Basuki Rahmat yang diambil menggunakan kamera CCTV dari pinggir jalan atau disamping objek. Data video yang sudah direkam, kemudian diolah oleh program dan dilakukan pengujian dengan beberapa parameter yang berbeda. Pengujian yang dilakukan di bagi menjadi beberapa bagian yaitu sebagai berikut :

1. Pengujian dengan perbedaan jumlah *frame rate*.
2. Pengujian dengan perbedaan jarak RoI.
3. Pengujian dengan perbedaan waktu.
4. Pengujian dengan perbedaan prosesor.
5. Pengujian program secara *real-time*.
6. Pengujian perbandingan *Kalman Filter* dan SORT.
7. Pengujian akurasi kecepatan kendaraan.

4.1 Pengujian dengan Perbedaan Jumlah *Frame Rate*

Pengujian dengan perbedaan jumlah *frame rate* bertujuan untuk mengetahui pengaruh jumlah *frame rate* pada hasil deteksi jenis kendaraan dan perhitungan kecepatan kendaraan. Jumlah *frame rate* diatur pada kamera CCTV sebelum melakukan pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel. Ada 3 parameter jumlah *frame rate* yang digunakan, yaitu 20 fps, 25 fps, dan 30 fps.

Pada tabel 4.1, merupakan perbandingan antara data perhitungan jumlah kendaraan secara manual menggunakan penglihatan indera berupa kedua mata dengan perhitungan jumlah kendaraan secara sistem menggunakan program python. Ada 4 jenis kendaraan yang dihitung, C = *car*, M = *motorbike*, B = *Bus*,T = *Truck* dan J= Jumlah.

Tabel 4.1: Tabel Pengujian *Vehicle Detection* dengan Perbedaan Jumlah *Frame Rate*

FPS	Manual					Sistem				
	C	M	B	T	J	C	M	B	T	J
20	55	86	0	2	143	43	38	2	3	86
25	81	71	0	0	152	60	32	0	0	92
30	69	95	2	0	166	48	46	0	5	99

Tabel 4.2: Tabel Pengujian Galat *Vehicle Detection* dengan Perbedaan Jumlah *Frame Rate*

FPS	**Galat**				
	C	M	B	T	J
20	21.82%	55.81%	100%	33.33%	39.86%
25	25.93%	54.93%	0%	0%	39.47%
30	30.43%	51.58%	100%	100%	40.36%

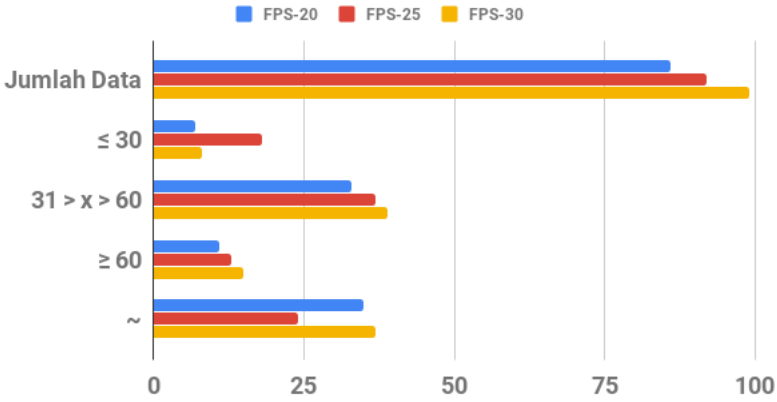
Pada tabel 4.2 masing-masing jenis kendaraan dilakukan perhitungan galat. Untuk jumlah galat secara keseluruhan pada deteksi jenis kendaraan *(vehicle detection)* sebesar 39.86% pada kecepatan kamera 20 fps, 39.47% pada kecepatan kamera 25 fps dan 40.36% pada kecepatan 30 fps. Semakin banyak jumlah *frame rate* berbanding lurus dengan jumlah jenis kendaraan, maka galat untuk mendeteksi jenis kendaraan juga akan semakin tinggi.

Setelah perhitungan jumlah kendaraan secara manual dan sistem dibandingkan, maka selanjutnya adalah menghitung nilai galat. Nilai galat diperoleh dari selesih jumlah manual dan jumlah sistem kemudian dibandingkan dengan jumlah yang sebenarnya dikali 100%. Perhitungan nilai galat untuk setiap pengujian dapat ketahui dengan menggunakan persamaan 4.1.

$$\bar{G} = \frac{JumlahManual - JumlahSistem}{JumlahManual} * 100\% \tag{4.1}$$

Tabel 4.3: Tabel Pengujian Pengukuran Kecepatan dengan Perbedaan Jumlah *Frame Rate*

FPS	Σ VD	Kecepatan (km/h)			
		≤30	31> x <60	≥60	∼
20	86	7	33	11	35
25	92	18	37	13	24
30	99	8	39	15	37



Gambar 4.1: Grafik pengujian pengukuran kecepatan kendaraan bedasarkan FPS

Pada tabel 4.3, Σ VD = jumlah kendaraan yang dideteksi oleh sistem, selanjutnya akan secara otomatis dideteksi estimasi kecepatan dari kendaraan tersebut. Ada 4 kategori estimasi pengukuran kecepatan yaitu :

1. Pada fps-20 : kecepatan ≤30 dideteksi sebanyak 7 unit, kecepatan diantara 31> x <60 sebanyak 33 unit, kecepatan ≥60 sebanyak 11 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 35 unit.

2. Pada fps-25 : kecepatan ≤30 dideteksi sebanyak 18 unit, kecepatan diantara 31> x <60 sebanyak 37 unit, kecepatan ≥60 sebanyak 13 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 24 unit.

3. Pada fps-30 : kecepatan ≤30 dideteksi sebanyak 8 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 15 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 37 unit.

4.2 Pengujian dengan Perbedaan Jarak RoI

Pengujian dengan perbedaan jarak RoI bertujuan untuk mengetahui pengaruh jarak RoI pada hasil deteksi jenis kendaraan dan perhitungan kecepatan kendaraan. RoI dipasang dan diletakkan sepasang sesuai dengan jarak yang telah ditentukan di pinggir jalan sebelum melakukan pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel kecepatan kamera sebesar 30fps. Ada 3 parameter jarak RoI yang digunakan, yaitu 15m, 20m, dan 30m.

Pada tabel 4.4, merupakan perbandingan antara data perhitungan jumlah kendaraan secara manual menggunakan penglihatan indera berupa kedua mata dengan perhitungan jumlah kendaraan secara sistem menggunakan program python. Ada 4 jenis kendaraan yang dihitung, C = *car*, M = *motorbike*, B = *Bus*,T = *Truck* dan J= Jumlah.

Tabel 4.4: Tabel Pengujian *Vehicle Detection* dengan Perbedaan Jarak RoI

RoI	Manual					Sistem				
	C	M	B	T	J	C	M	B	T	J
15	69	95	2	0	166	48	46	0	5	99
20	35	37	0	2	74	26	21	0	2	49
25	70	52	0	1	123	53	36	0	3	92

Pada tabel 4.8 masing-masing jenis kendaraan dilakukan perhitungan galat. Untuk jumlah galat secara keseluruhan pada deteksi jenis kendaraan *(vehicle detection)* sebesar 40.36% pada jarak

Tabel 4.5: Tabel Pengujian Galat *Vehicle Detection* dengan Perbedaan Jarak RoI

RoI	**Galat**				
	C	M	B	T	J
15	30.43%	51.58%	100%	100%	40.36%
20	25.71%	43.24%	0%	0%	33.78%
25	24.29%	30.77%	0%	66.67%	25.20%

RoI 15 meter, 33.78% pada jarak RoI 20 meter dan 25.20% pada jarak RoI 30 meter. Semakin besar jarak RoI, maka galat untuk mendeteksi jenis kendaraan juga akan semakin rendah.

Tabel 4.6: Tabel Pengujian Pengukuran Kecepatan dengan Perbedaan Jarak RoI

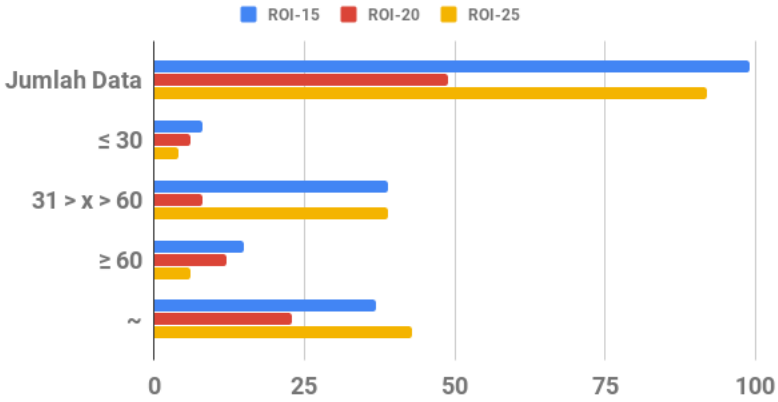
RoI	Σ VD	Kecepatan (km/h)			
		≤30	31> x <60	≥60	∼
15	99	8	39	15	37
20	49	6	8	12	23
25	92	4	39	6	43

Pada tabel 4.6, Σ VD = jumlah kendaraan yang dideteksi oleh sistem, selanjutnya akan secara otomatis dideteksi estimasi kecepatan dari kendaraan tersebut. Ada 4 kategori estimasi pengukuran kecepatan yaitu :

1. Pada RoI-15 : kecepatan ≤30 dideteksi sebanyak 8 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 15 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 37 unit.

2. Pada RoI-20 : kecepatan ≤30 dideteksi sebanyak 6 unit, kecepatan diantara 31> x <60 sebanyak 8 unit, kecepatan ≥60 sebanyak 12 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 23 unit.

3. Pada RoI-30 : kecepatan ≤30 dideteksi sebanyak 4 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 6 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 43 unit.



Gambar 4.2: Grafik pengujian pengukuran kecepatan kendaraan bedasarkan RoI

4.3 Pengujian dengan Perbedaan Waktu

Pengujian dengan perbedaan waktu bertujuan untuk mengetahui pengaruh perbedaan waktu pada hasil deteksi jenis kendaraan dan perhitungan kecepatan kendaraan. Pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel dengan kecepatan kamera sebesar 30fps. Ada 4 parameter waktu yang digunakan, yaitu pagi, siang, sore dan malam.

Pada tabel 4.7, merupakan perbandingan antara data perhitungan jumlah kendaraan secara manual menggunakan penglihatan indera berupa kedua mata dengan perhitungan jumlah kendaraan

secara sistem menggunakan program python. Ada 4 jenis kendaraan yang dihitung, C = *car*, M = *motorbike*, B = *Bus*,T = *Truck* dan J= Jumlah.

Tabel 4.7: Tabel Pengujian *Vehicle Detection* dengan Perbedaan waktu

Waktu	Manual					Sistem				
	C	M	B	T	J	C	M	B	T	J
Pagi	69	95	2	0	166	48	46	0	5	99
Siang	80	96	0	0	176	60	58	1	7	126
Sore	57	68	2	1	128	45	44	0	3	92
Malam	27	67	0	0	94	14	0	0	1	15

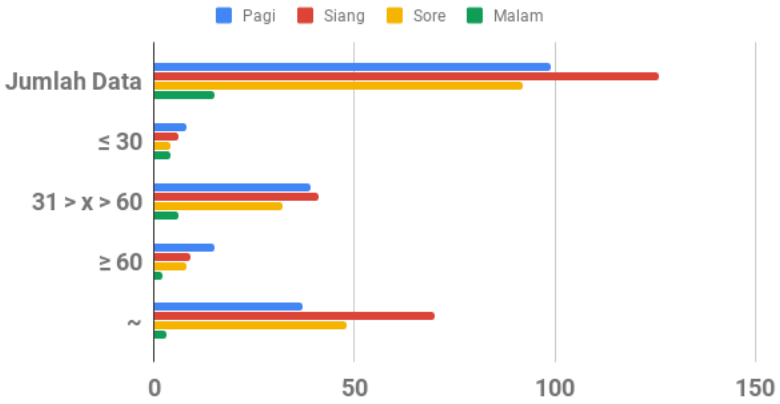
Tabel 4.8: Tabel Pengujian Galat *Vehicle Detection* dengan Perbedaan Waktu

Waktu	**Galat**				
	C	M	B	T	J
Pagi	30.43%	51.58%	100%	100%	40.36%
Siang	25.00%	39.58%	100%	100%	28.41%
Sore	21.05%	30.77%	100%	66.67%	28.13%
Malam	48.15%	100%	0%	100%	84.04%

Pada tabel 4.8 masing-masing jenis kendaraan dilakukan perhitungan galat. Untuk jumlah galat secara keseluruhan pada deteksi jenis kendaraan *(vehicle detection)* sebesar 40.36% pada pagi hari, 28.41% pada siang hari, 28.13% pada sore hari dan 84.04% pada malam hari.

Tabel 4.9: Tabel Pengujian Pengukuran Kecepatan dengan Perbedaan Waktu

Waktu	Σ VD	Kecepatan (km/h)			
		≤30	31>x <60	≥60	∼
Pagi	99	8	39	15	37
Siang	126	6	41	9	70
Sore	92	4	32	8	48
Malam	15	4	6	2	3



Gambar 4.3: Grafik pengujian pengukuran kecepatan kendaraan bedasarkan waktu

Pada tabel 4.6, Σ VD = jumlah kendaraan yang dideteksi oleh sistem, selanjutnya akan secara otomatis dideteksi estimasi kecepatan dari kendaraan tersebut. Ada 4 kategori estimasi pengukuran kecepatan yaitu :

1. Pada pagi hari : kecepatan ≤30 dideteksi sebanyak 8 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 15 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 37 unit.

2. Pada siang hari : kecepatan ≤30 dideteksi sebanyak 6 unit, kecepatan diantara 31> x <60 sebanyak 41 unit, kecepatan ≥60 sebanyak 9 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 70 unit.

3. Pada malam hari : kecepatan ≤30 dideteksi sebanyak 4 unit, kecepatan diantara 31> x <60 sebanyak 32 unit, kecepatan ≥60 sebanyak 8 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 48 unit.

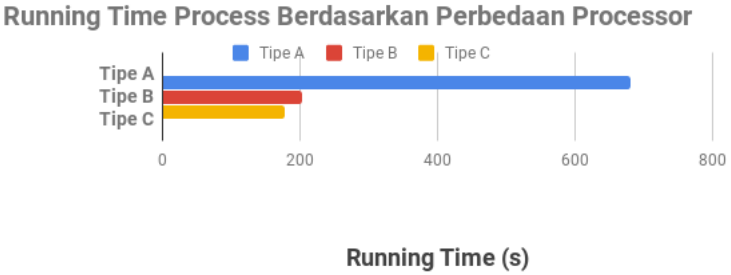
4. Pada malam hari : kecepatan ≤30 dideteksi sebanyak 4 unit, kecepatan diantara 31> x <60 sebanyak 6 unit, kecepatan ≥60 sebanyak 2 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 3 unit.

4.4 Pengujian dengan Perbedaan Processor

Pengujian dengan perbedaan processor bertujuan untuk mengetahui pengaruh perbedaan processor pada hasil kecepatan komputasi sistem. Pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel dengan kecepatan kamera sebesar 30fps selama satu menit. Ada 3 jenis processor yang digunakan, yaitu ASUS A456U (Tipe A): i5 generasi ke-6 NVIDIA GeForce G930Mx, ASUS ROG FX553VD (Tipe B): i7 generasi ke-7 NVIDIA GeForce GTX 1050, Lenovo legion y530 (Tipe C): i7 generasi ke-8 NVIDIA GeForce GTX 1050 Ti.

Tabel 4.10: Tabel Pengujian komputasi dengan perbedaan processor

Tipe Komputer	**Frame**	**Waktu S/Frame**	**Running Time (s)**
Tipe A	1765	0.3853368687562159	680
Tipe B		0.1147370134467106	203
Tipe C		0.09996012101430056	176



Gambar 4.4: Grafik pengujian kecepatan komputasi sitem

Dari hasil kecepatan komputasi sistem menjalankan 1765 *frame* didapatkan hasil bahwa ASUS A456U : i5 generasi ke-6 NVIDIA GeForce G930Mx berkecepatan paling lambat dengan 680 sekon (sekitar 10 menit) atau sebesar 64,2%, kemudian disusul oleh ASUS ROG FX553VD : i7 generasi ke-7 NVIDIA GeForce GTX 1050 berkecepatan 203 sekon (sekitar 4 menit) atau sebesar 19,2% dan Lenovo legion y530 : i7 generasi ke-8 NVIDIA GeForce GTX 1050 Ti berkecepatan paling kencang dengan 176 sekon (sekitar 3 menit) atau sebesar 16,4%.

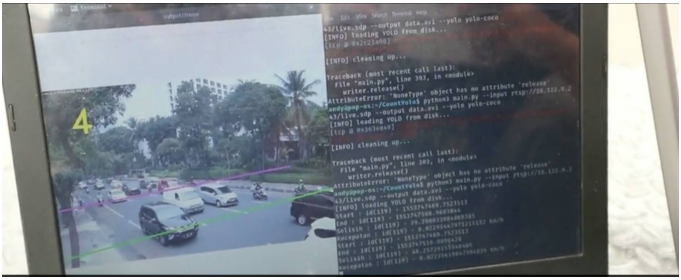
4.5 Pengujian program secara *real-time*

Pengujian program secara *real-time* bertujuan untuk mengetahui bagaimana sistem jika dijalankan secara *(real-time)*. Pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel dengan kecepatan kamera sebesar 30fps pada waktu pagi hari pukul 9:55 - 13:00 (gambar 4.5) dan pada waktu malam hari pukul 20:00 - 23:00 (gambar 4.8) menggunakan 3 unit jenis laptop yang berbeda ASUS A456U : i5 generasi ke-6 NVIDIA GeForce G930Mx, ASUS ROG FX553VD : i7 generasi ke-7 NVIDIA GeForce GTX 1050, Lenovo legion y530 : i7 generasi ke-8 NVIDIA GeForce GTX 1050 Ti.

Setelah dilakukan uji coba, program dapat berjalan secara *real-time* untuk melakukan deteksi jenis kendaraan akan tetapi jika untuk melakukan proses perhitungan kecepatan kendaraan, program mengalami hambatan. Objek yang lewat tidak tertangkap oleh ka-

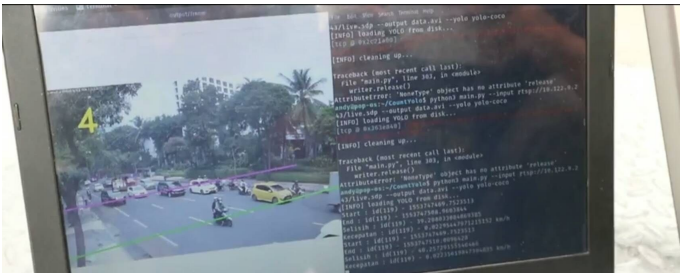
mera CCTV, bila objek tertangkap oleh kamera CCTV maka program akan mengalami *stuck* karena objek yang baru masuk tertangkap *frame* dengan cepat menghilang (*ter-skip* keluar dari tangkapan kamera) sehingga program tidak dapat berjalan seperti apa yang diharapkan.

Gambar 4.5: Pengujian *real-time* pada waktu pagi hari

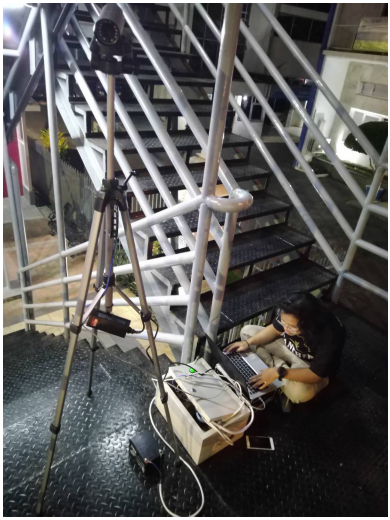


Gambar 4.6: Hasil pengujian *real-time* pada waktu pagi hari

Pada gambar 4.6, objek tertangkapan kamera CCTV, namun pada *frame* berikutnya objek dengan cepat menghilang dan berganti dengan objek baru (gambar 4.7). Pada gambar 4.9, terlihat objek terdeteksi oleh *bounding box*. Pada gambar 4.10, objek dengan cepat melaju pada *frame* berikutnya yang menyebabkan program tidak dapat memproses untuk melakukan perhitungan kecepatan.

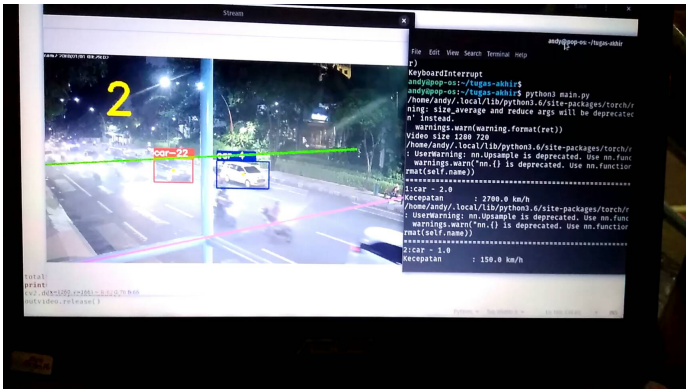


Gambar 4.7: Hasil pengujian *real-time* pada waktu pagi hari

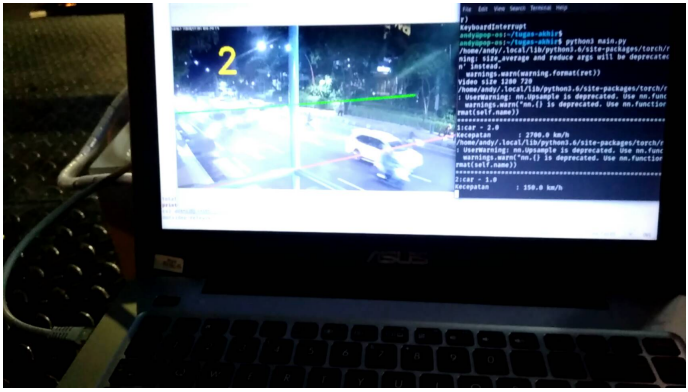


Gambar 4.8: Pengujian *real-time* pada waktu malam hari

Analisa yang pertama adalah karena masalah spesifikasi laptop ASUS A456U : i5 generasi ke-6 NVIDIA GeForce G930Mx yang kurang memumpuni untuk melakukan komputasi sehingga saya menggunakan laptop lain ASUS ROG FX553VD : i7 generasi ke-7 NVIDIA GeForce GTX 1050 dan Lenovo legion y530 : i7 generasi ke-8 NVIDIA GeForce GTX 1050 Ti. Hasil uji coba yang dilakukan tetap sama, program tidak dapat berjalan secara *real-time* karena setiap *frame* membutuhkan waktu untuk mengolah data.



Gambar 4.9: Hasil pengujian *real-time* pada waktu malam hari



Gambar 4.10: Hasil pengujian *real-time* pada waktu malam hari

4.6 Pengujian perbandingan *Kalman Filter* dan SORT

Pengujian perbandingan *Kalman Filter* dan SORT bertujuan untuk membandingkan hasil pengukuran kecepatan kendaraan dari *tracking* menggunakan metode *Kalman Filter* dengan hasil pengukuran kecepatan kendaraan dari *tracking* menggunakan metode SORT.

4.6.1 Pengujian Pengukuran Kecepatan dengan Perbedaan Jumlah *Frame Rate*

Jumlah *frame rate* diatur pada kamera CCTV sebelum melakukan pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel. Ada 3 parameter jumlah *frame rate* yang digunakan, yaitu 20 fps, 25 fps, dan 30 fps.

Tabel 4.11: Tabel Pengujian Pengukuran Kecepatan dengan Perbedaan Jumlah *Frame Rate*

FPS	**Kalman Filter**					**SORT**				
	Σ VD	≤30	31>x<60	≥60	∼	Σ VD	≤30	30>x>60	≥60	∼
20	66	35	14	11	6	86	7	33	11	35
25	70	33	20	5	24	92	18	37	13	24
30	86	47	16	16	7	99	8	39	15	37

Dari tabel 4.11, Σ VD = jumlah kendaraan yang dideteksi oleh sistem, selanjutnya akan secara otomatis dideteksi estimasi kecepatan dari kendaraan tersebut. Ada 5 kategori estimasi pengukuran kecepatan yaitu:

1. Pada FPS-20:
   - Kalman Filter
     Σ VD dideteksi sebanyak 66 unit, kecepatan ≤30 dideteksi sebanyak 35 unit, kecepatan diantara 31> x <60 sebanyak 14 unit, kecepatan ≥60 sebanyak 11 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 6 unit.
   - SORT
     Σ VD dideteksi sebanyak 86 unit, kecepatan ≤30 dideteksi sebanyak 7 unit, kecepatan diantara 31> x <60 sebanyak 33 unit, kecepatan ≥60 sebanyak 11 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 35 unit.
2. Pada FPS-25:
   - Kalman Filter
     Σ VD dideteksi sebanyak 70 unit, kecepatan ≤30 dideteksi sebanyak 33 unit, kecepatan diantara 31> x <60 sebanyak 20 unit, kecepatan ≥60 sebanyak 5 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 24 unit.

- SORT

  Σ VD dideteksi sebanyak 92 unit, kecepatan ≤30 dideteksi sebanyak 18 unit, kecepatan diantara 31> x <60 sebanyak 37 unit, kecepatan ≥60 sebanyak 13 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 24 unit.

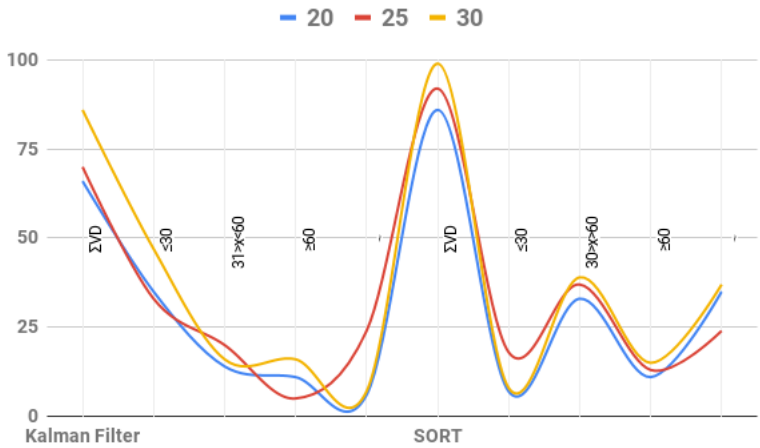
3. Pada FPS-30:

   - Kalman Filter

     Σ VD dideteksi sebanyak 86 unit, kecepatan ≤30 dideteksi sebanyak 47 unit, kecepatan diantara 31> x <60 sebanyak 16 unit, kecepatan ≥60 sebanyak 16 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 7 unit.

   - SORT

     Σ VD dideteksi sebanyak 99 unit, kecepatan ≤30 dideteksi sebanyak 8 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 15 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 37 unit.



Gambar 4.11: Grafik pengujian pengukuran kecepatan kendaraan berdasarkan jumlah *frame rate*

4.6.2 Pengujian Pengukuran Kecepatan dengan Perbedaan Jarak RoI

Pengujian dengan perbedaan jarak RoI bertujuan untuk mengetahui pengaruh jarak RoI pada hasil deteksi jenis kendaraan dan perhitungan kecepatan kendaraan. RoI dipasang dan diletakkan sepasang sesuai dengan jarak yang telah ditentukan di pinggir jalan sebelum melakukan pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel kecepatan kamera sebesar 30fps. Ada 3 parameter jarak RoI yang digunakan, yaitu 15m, 20m, dan 30m.

Tabel 4.12: Tabel Pengujian Pengukuran Kecepatan dengan Perbedaan Jarak RoI

RoI	Kalman Filter					SORT				
	Σ VD	≤30	31>x <60	≥60	∼	Σ VD	≤30	30>x >60	≥60	∼
15	70	47	19	2	18	99	8	39	15	37
20	52	25	14	4	9	49	6	8	12	23
25	86	47	16	16	7	92	1	39	6	43

Dari tabel 4.11, Σ VD = jumlah kendaraan yang dideteksi oleh sistem, selanjutnya akan secara otomatis dideteksi estimasi kecepatan dari kendaraan tersebut. Ada 5 kategori estimasi pengukuran kecepatan yaitu:

1. Pada RoI-15:

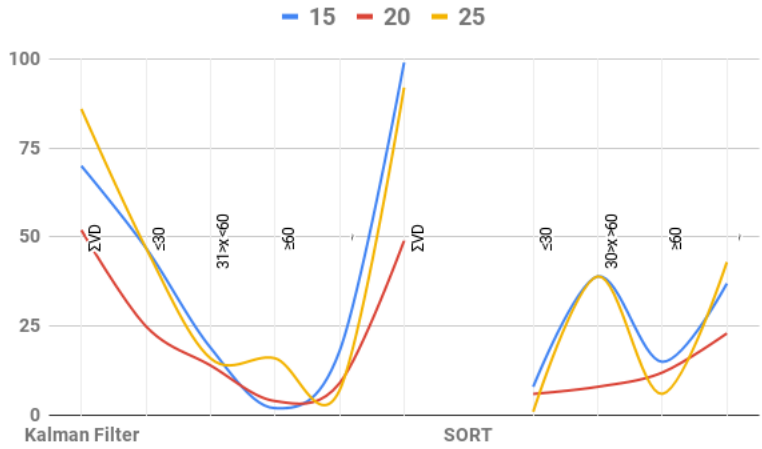
   - Kalman Filter
     Σ VD dideteksi sebanyak 70 unit, kecepatan ≤30 dideteksi sebanyak 47 unit, kecepatan diantara 31> x <60 sebanyak 19 unit, kecepatan ≥60 sebanyak 2 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 18 unit.
   - SORT
     Σ VD dideteksi sebanyak 99 unit, kecepatan ≤30 dideteksi sebanyak 8 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 15 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 37 unit.

2. Pada RoI-20:

   - Kalman Filter
     Σ VD dideteksi sebanyak 52 unit, kecepatan ≤30 dideteksi sebanyak 25 unit, kecepatan diantara 31> x <60 sebanyak 14 unit, kecepatan ≥60 sebanyak 4 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 9 unit.
   - SORT
     Σ VD dideteksi sebanyak 49 unit, kecepatan ≤30 dideteksi sebanyak 6 unit, kecepatan diantara 31> x <60 sebanyak 8 unit, kecepatan ≥60 sebanyak 12 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 23 unit.

3. Pada RoI-25:

   - Kalman Filter
     Σ VD dideteksi sebanyak 86 unit, kecepatan ≤30 dideteksi sebanyak 47 unit, kecepatan diantara 31> x <60 sebanyak 16 unit, kecepatan ≥60 sebanyak 16 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 7 unit.
   - SORT
     Σ VD dideteksi sebanyak 92 unit, kecepatan ≤30 dideteksi sebanyak 1 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 6 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 43 unit.



Gambar 4.12: Grafik pengujian pengukuran kecepatan kendaraan bedasarkan jarak RoI

4.6.3 Pengujian Pengukuran Kecepatan dengan Perbedaan Waktu

Pengujian dengan perbedaan waktu bertujuan untuk mengetahui pengaruh perbedaan waktu pada hasil deteksi jenis kendaraan dan perhitungan kecepatan kendaraan. Pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel dengan kecepatan kamera sebesar 30fps. Ada 4 parameter waktu yang digunakan, yaitu pagi, siang, sore dan malam.

Tabel 4.13: Tabel Pengujian Pengukuran Kecepatan dengan Perbedaan Waktu

Waktu	**Kalman Filter**					**SORT**				
	Σ VD	≤30	31>x <60	≥60	~	Σ VD	≤30	30>x >60	≥60	~
Pagi	86	47	16	16	7	99	8	39	15	37
Siang	94	46	18	8	22	126	6	41	9	70
Sore	79	37	12	8	22	92	4	32	8	48
Malam	43	19	15	2	7	15	4	6	2	3

Dari tabel 4.11, Σ VD = jumlah kendaraan yang dideteksi oleh

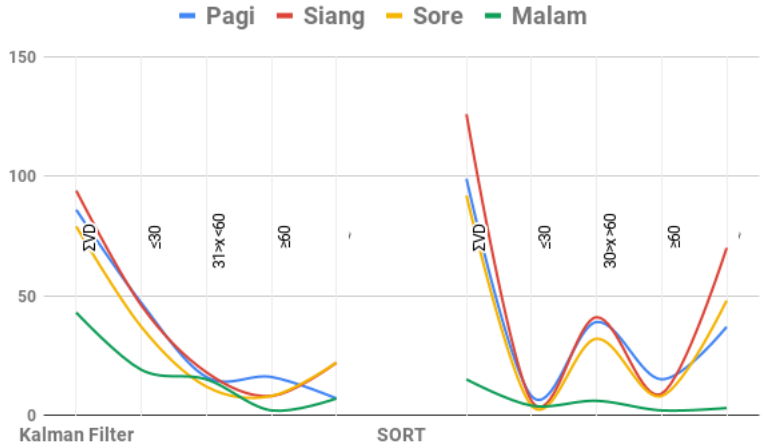
sistem, selanjutnya akan secara otomatis dideteksi estimasi kecepatan dari kendaraan tersebut. Ada 5 kategori estimasi pengukuran kecepatan yaitu:

1. Pada Pagi hari:
   - Kalman Filter
     Σ VD dideteksi sebanyak 86 unit, kecepatan ≤30 dideteksi sebanyak 47 unit, kecepatan diantara 31> x <60 sebanyak 16 unit, kecepatan ≥60 sebanyak 16 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 7 unit.
   - SORT
     Σ VD dideteksi sebanyak 99 unit, kecepatan ≤30 dideteksi sebanyak 8 unit, kecepatan diantara 31> x <60 sebanyak 39 unit, kecepatan ≥60 sebanyak 15 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 37 unit.
2. Pada Siang hari:
   - Kalman Filter
     Σ VD dideteksi sebanyak 94 unit, kecepatan ≤30 dideteksi sebanyak 46 unit, kecepatan diantara 31> x <60 sebanyak 18 unit, kecepatan ≥60 sebanyak 8 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 22 unit.
   - SORT
     Σ VD dideteksi sebanyak 126 unit, kecepatan ≤30 dideteksi sebanyak 6 unit, kecepatan diantara 31> x <60 sebanyak 41 unit, kecepatan ≥60 sebanyak 9 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 70 unit.
3. Pada Sore hari:
   - Kalman Filter
     Σ VD dideteksi sebanyak 79 unit, kecepatan ≤30 dideteksi sebanyak 37 unit, kecepatan diantara 31> x <60 sebanyak 12 unit, kecepatan ≥60 sebanyak 8 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 22 unit.
   - SORT
     Σ VD dideteksi sebanyak 92 unit, kecepatan ≤30 dideteksi sebanyak 4 unit, kecepatan diantara 31> x <60

sebanyak 32 unit, kecepatan ≥60 sebanyak 8 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 48 unit.

4. Pada Malam hari:

   - Kalman Filter
     Σ VD dideteksi sebanyak 43 unit, kecepatan ≤30 dideteksi sebanyak 19 unit, kecepatan diantara 31> x <60 sebanyak 15 unit, kecepatan ≥60 sebanyak 2 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 7 unit.
   - SORT
     Σ VD dideteksi sebanyak 15 unit, kecepatan ≤30 dideteksi sebanyak 4 unit, kecepatan diantara 31> x <60 sebanyak 6 unit, kecepatan ≥60 sebanyak 2 unit dan kecepatan (kecepatan tak terdeteksi) sebanyak 3 unit.



Gambar 4.13: Grafik pengujian pengukuran kecepatan kendaraan bedasarkan waktu

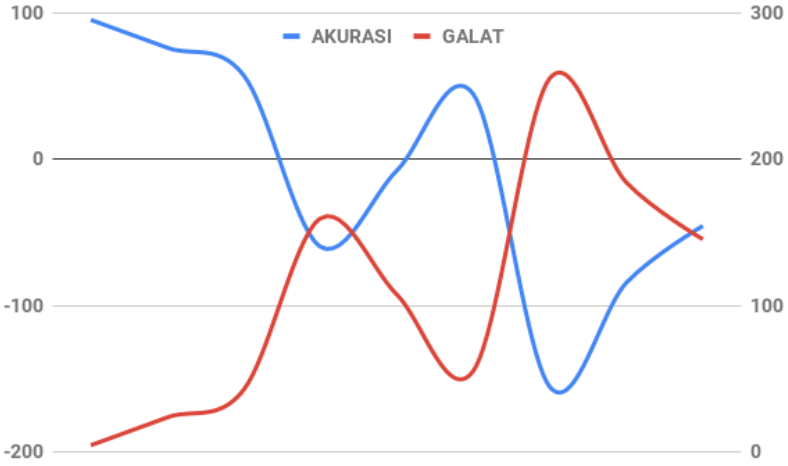
4.6.4 Pengujian Akurasi Kecepatan Kendaraan

Pengujian akurasi kecepatan kendaraan bertujuan untuk mengetahui seberapa besar akurasi program jika dibandingkan dengan kecepatan pada spidometer kendaraan. Pengujian dilakukan sebanyak 3 kali, sebelum melakukan pengujian dilakukan pengambilan data dengan resolusi yang digunakan dalam pengujian ini yaitu 1280x720 piksel dengan kecepatan kamera sebesar 30fps. Ada 4 parameter kecepatan yang diuji, yaitu 20 km/h, 30 km/h, 40km/h dan 60 km/h dengan panjang RoI yang berbeda masing-masing di jarak 15m, 20m, dan 25m.

Tabel 4.14: Tabel Pengujian Akurasi Kecepatan Kendaraan pada 20 km/h

SPIDO METER	UJI	JARAK RoI	KECEPATAN (km/h)	GALAT (%)	AKURASI (%)
20 km/h	1	15 m	19.01	4.95	95.05
		20 m	15.21	23.95	76.05
		25 m	11.41	42.95	57.05
	2	15 m	51.92	159.60	-59.60
		20 m	41.54	107.70	-7.70
		25 m	31.15	55.75	44.25
	3	15 m	71.05	255.25	-155.25
		20 m	56.84	184.20	-84.20
		25 m	49.09	145.45	-45.45

Pengujian akurasi kecepatan kendaraan pada 20 km/h sebanyak 3 kali menghasilkan akurasi terbaik sebesar 95.05% dengan galat terkecil sebesar 4.95% pada percobaan pertama di jarak RoI 15 meter.



Gambar 4.14: Grafik pengujian akurasi kecepatan kendaraan pada 20 km/h

Tabel 4.15: Tabel Pengujian Akurasi Kecepatan Kendaraan pada 30 km/h

SPIDO METER	UJI	JARAK RoI	KECEPATAN (km/h)	GALAT (%)	AKURASI (%)
30 km/h	1	15 m	72.97	143.23	-43.23
		20 m	58.38	94.60	5.40
		25 m	43.78	45.93	54.07
	2	15 m	61.36	104.53	-4.53
		20 m	49.10	63.67	36.33
		25 m	36.81	22.70	77.30
	3	15 m	64.29	114.30	-14.30
		20 m	51.43	71.43	28.57
		25 m	38.57	28.57	71.43

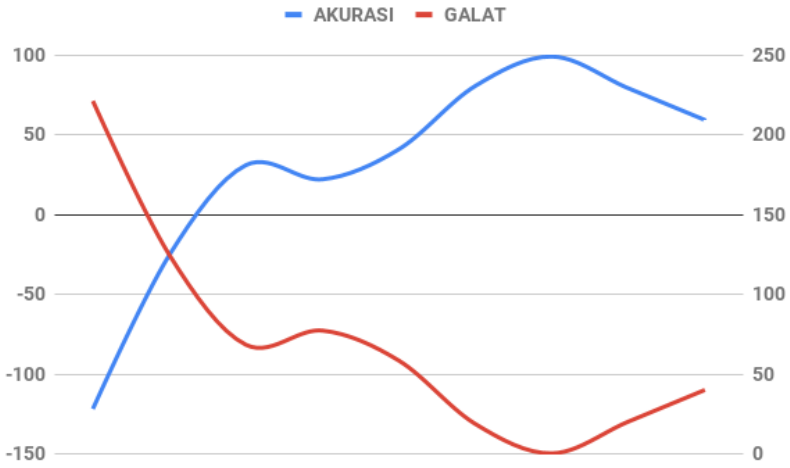
Pengujian akurasi kecepatan kendaraan pada 30 km/h sebanyak 3 kali menghasilkan akurasi terbaik sebesar 77.30% dengan galat terkecil sebesar 22.70% pada percobaan kedua di jarak RoI 25 meter.

Gambar 4.15: Grafik pengujian akurasi kecepatan kendaraan pada 30 km/h

Tabel 4.16: Tabel Pengujian Akurasi Kecepatan Kendaraan pada 40 km/h

SPIDO METER	UJI	JARAK RoI	KECEPATAN (km/h)	GALAT (%)	AKURASI (%)
40 km/h	1	15 m	128.57	221.43	-121.43
		20 m	90.00	125.00	-25.00
		25 m	67.50	68.75	31.25
	2	15 m	71.05	77.63	22.38
		20 m	63.53	58.83	41.18
		25 m	47.65	19.13	80.88
	3	15 m	39.71	0.73	99.28
		20 m	31.76	20.60	79.40
		25 m	23.82	40.45	59.55

Pengujian akurasi kecepatan kendaraan pada 40 km/h sebanyak 3 kali menghasilkan akurasi terbaik sebesar 99.28% dengan galat terkecil sebesar 0.73% pada percobaan ketiga di jarak RoI 15 meter.

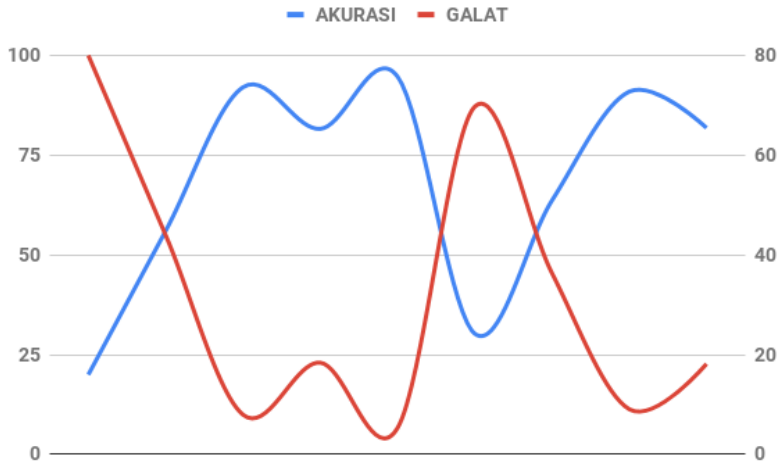


Gambar 4.16: Grafik pengujian akurasi kecepatan kendaraan pada 40 km/h

Tabel 4.17: Tabel Pengujian Akurasi Kecepatan Kendaraan pada 60 km/h

SPIDO METER	UJI	JARAK RoI	KECEPATAN (km/h)	GALAT (%)	AKURASI (%)
60 km/h	1	15 m	108.00	80.00	20.00
		20 m	86.40	44.00	56.00
		25 m	64.80	8.00	92.00
	2	15 m	71.05	18.42	81.58
		20 m	56.84	5.27	94.73
		25 m	18.18	69.70	30.30
	3	15 m	81.82	36.37	63.63
		20 m	65.45	9.08	90.92
		25 m	49.10	18.17	81.83

Pengujian akurasi kecepatan kendaraan pada 60 km/h sebanyak 3 kali menghasilkan akurasi terbaik sebesar 94.72% dengan galat terkecil sebesar 5.27% pada percobaan kedua di jarak RoI 20 meter.



Gambar 4.17: Grafik pengujian akurasi kecepatan kendaraan pada 60 km/h

Pengujian akurasi kecepatan kendaraan sebanyak 3 kali dengan membandingkan nilai kecepatan spidometer sepeda motor dan nilai kecepatan estimasi program menghasilkan nilai yang bervariasi dengan tingkat galat dan akurasi yang berbeda-beda. Pengambilan data video yang dilakukan dengan merekam spidometer sepeda motor juga tidak memiliki nilai presisi yang tepat sehingga memungkinkan adanya galat yang mana juga akan mempengaruhi akurasi program.

Galat yang tinggi juga dipengaruhi oleh deteksi objek yang terdeteksi setelah memasuki garis RoI bahkan mendekati garis keluar dari RoI. Akurasi yang menghasilkan nilai minus menandakan objek yang terdeteksi mendekati garis keluar RoI sehingga nilai *frame* yang didapat memiliki selisih yang besar yang menghasilkan nilai estimasi kecepatan kendaraan sangat besar. Dalam hal ini ketepatan deteksi objek dan *tracking* objek sangat mempengaruhi nilai pengukuran kecepatan.

Halaman ini sengaja dikosongkan

BAB 5
PENUTUP

5.1 Kesimpulan

Dalam penelitian ini, telah diimplementasikan serangkaian prosedur untuk menentukan deteksi kecepatan kendaraan dengan proses penjejakan berbasis *deep learning* dan setelah dilakukan beberapa pengujian data, didapatkan kesimpulan sebagai berikut:

1. Sistem tidak dapat dijalankan secara *real-time* karena membutuhkan waktu sekitar 0.3s untuk memproses setiap *frame*-nya, sehingga untuk satu menit video dengan jumlah *frame* sebanyak 1765 membutuhkan waktu selama 10 menit untuk memproses data hingga menghasilkan estimasi pengkuran kecepatan kendaraan.

2. Semakin tinggi spesifikasi *processor* dan GPU komputer yang digunakan, maka dapat mempercepat proses komputasi program hingga 0,1s setiap *frame*-nya dan membutuhkan waktu selama 3 menit untuk memproses data sebanyak 1765 *frame* dengan menggunakan i7 generasi ke-8 GTX 1050 Ti.

3. Pengujian sistem pada waktu malam hari mempengaruhi hasil deteksi objek sehingga banyak objek yang tidak terdeteksi dan tidak dapat dilakukan *tracking* untuk mendapatkan estimasi pengukuran kecepatan kendaraan.

4. Penggunaan *deep learning* dalam sistem ini menggunakan model YOLOv3 *(You Only Look Once)* memiliki akurasi deteksi objek yang sangat akurat tanpa terjadi kesalahan deteksi atau kesalahan dalam *label class object*.

5. Penambahan metode SORT pada sistem untuk *tracking* objek menghasilkan *output* estimasi kecepatan kendaraan dengan objek *class* yang sesuai dan id objek yang cenderung konstan tidak berubah jika dibandingkan dengan penggunaan metode Kalman Filter yang tidak dapat menghasilkan *output* estimasi kecepatan kendaraan dengan objek *class* yang sesuai serta id objek yang cenderung berubah-ubah.

6. Program memiliki akurasi tinggi pada percobaan pertama untuk kecepatan spidomoter 20km/h sebesar 95.05% dan pada percobaan ketiga untuk kecepatan spidomoter 40km/h sebesar 99.28% di jarak RoI 15 m. Untuk kecepatan 60km/h akurasi tinggi didapatkan pada percobaan kedua sebesar 94.73% di jarak RoI 20m. Sedangkan untuk kecepatan 30km/h hanya memiliki akurasi tinggi sebesar 77.3% pada percobaan kedua di jarak RoI 25m.

5.2 Saran

Demi pengembangan lebih lanjut mengenai tugas akhir ini, disarankan beberapa langkah lanjutan sebagai berikut:

1. Perhitungan kecepatan kendaraan pada sistem membuat deteksi *objek* menggunakan YOLOv3 tidak berjalan maksimal untuk hasil *output* estimasi kecepatan kendaraan sehingga ada beberapa *objek* yang tidak terkakulasi oleh program.
2. Sistem membutuhkan waktu yang lama untuk memproses data karena komputasi program untuk *deep learning* sangat berat sehingga tidak cocok untuk digunakan pada sistem yang mengaharuskan *real-time*.
3. Sistem membutuhkan komputer dengan spesifikasi GPU yang tinggi untuk mempercepat prosess komputasi.
4. *Tracking* pada objek mempengaruhi jumlah deteksi dan perhitungan kecepatan kendaraan sehingga dibutuhkan metode *mutiple object tracking* yang lebih handal.

DAFTAR PUSTAKA

[1] "New camera ip hong ngoai 1.3 megapixel vivotek ib8354-c." http://thuongtin.co/camera-ip-hong-ngoai-1-3-megapixel-vivotek-\ib8354-c.html/. terakhir diakses tanggal 28 Juni 2019 7:07 PM. (Dikutip pada halaman xi, xii, 6, 42).

[2] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, vol. 521. (Dikutip pada halaman xi, 8).

[3] "Convolutional neural network." https://www.mathworks.com/solutions/deep-learning/convolutional-neural-network.html/. terakhir diakses tanggal 28 Juni 2019 7:39 PM. (Dikutip pada halaman xi, 9).

[4] S. Chowdhury, "Large scale traffic surveillance : Vehicle detection and classification using cascade classifier and convolutional neural network," Institut of Engineering and Management, Kolkata. (Dikutip pada halaman xi, 9, 10, 11).

[5] "Review: Yolov3 - you only look once (object detection)." https://towardsdatascience.com/review-yolov3-you-only-look-once-object-detec\tion-eab75d7a1ba6/. terakhir diakses tanggal 28 Juni 2019 7:55 PM. (Dikutip pada halaman xi, xii, 13, 27, 45).

[6] "What's new in yolo v3?." https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b/. terakhir diakses tanggal 28 Juni 2019 8:21 PM. (Dikutip pada halaman xi, 15).

[7] THRUN, Sebastian, W. Burgard, and D. Fox, "Probabilistic robotics 1 st eds: Chapter 1-3," USA: Wiley and Sons, 2000. (Dikutip pada halaman xi, 17, 18).

[8] "Kalman filter." https://en.wikipedia.org/wiki/Kalman_filter/. terakhir diakses tanggal 28 Juni 2019 9:09 PM. (Dikutip pada halaman xi, 20).

[9] "Intersection over union (iou) for object detection." https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detec\tion/. terakhir diakses tanggal 28 Juni 2019 9:36 PM. (Dikutip pada halaman xi, 22).

[10] E. Bochinski, V. Eiselein, and T. Sikora, "High-speed tracking-by-detection without using image information," IEEE AVSS Lecce - ITALY, August 2017. (Dikutip pada halaman xi, 23).

[11] Y.-J. Cha, W. Choi, G. Suh, S. Mahmoudkhan, and O. Buyukozturk, "Autonomous structural visual inspection using region-based deep learning for detecting multiple damage types," Computer-Aided Civil and Infrastructure Engineering, 2017. (Dikutip pada halaman xi, 28).

[12] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, and R. Girshick, "Microsoft coco: Common objects in context," arXiv, 2015. (Dikutip pada halaman xii, 44).

[13] M. Maj, "Object detection and image classification with yolo," Appsilon Data Science, 2018. (Dikutip pada halaman xii, 46).

[14] B. P. S. K. Surabaya, Kota Surabaya Dalam Angka 2018 - Banyaknya Kendaraan Bermotor menurut Jenisnya. 2018. (Dikutip pada halaman 1).

[15] A. Ramadlan, Pemantauan Pelanggaran Kecepatan Kendaraan Menggunakan Dua Sensor Kamera. Institut Teknologi Sepuluh Nopember, 2018. (Dikutip pada halaman 2).

[16] N. H. Tsani, "The implementattion of vehicle speed detection using webcam with frame difference method," bachelor thesis, Telkom University, August, 2017. (Dikutip pada halaman 5).

[17] Z. C. Lawa, M. Najoan, A. M. Lumenta, and M. Tuegeh, "Perancangan teknologi ip camera di jaringan radio wirelless pt. pln wilayah suluttenggo," Jurusan Teknik Elektro-FT, UNSRAT, 2012. (Dikutip pada halaman 6).

[18] I. W. S. E. P, A. Y. Wijaya, and R. Soelaiman, "Klasifikasi citra menggunakan convolutional neural network (cnn) pada caltech 101," JURNAL TEKNIK ITS Vol. 5, No. 1, ISSN: 2337-3539, 2016. (Dikutip pada halaman 7, 9).

[19] J. Lu1, C. Ma, L. Li, X. Xing, Y. Zhang, Z. Wang, and J. Xu, "A vehicle detection method for aerial image based on yolo," Journal of Computer and Communications, 2018, 6, 98-107, 2018. (Dikutip pada halaman 11).

[20] J. Lu1, C. Ma, L. Li, X. Xing, Y. Zhang, Z. Wang, and J. Xu, "Detection of apple lesions in orchards based on deep learning methods of cyclegan and yolov3-dense," Journal of Sensors Article ID 7630926, 13 pages, 2019. (Dikutip pada halaman 12).

[21] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," CoRR vol. abs/1506.02640, 2015. (Dikutip pada halaman 12).

[22] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," arXiv, 2018. (Dikutip pada halaman 13).

[23] W. G. dan Bishop. G, "An introduction to the kalman filter," University of North Carolina, Chapel Hill, 2006. (Dikutip pada halaman 16).

[24] P. H. A. dan Thakore. D. G., "Moving object tracking using kalman filter," International Journal of Computer Science and Mobile Computing Vol.2 Issue. 4 pg. 326-332., 2013. (Dikutip pada halaman 19).

[25] H. V. A. K. dan Kusworo Adi, "Implementasi object tracking untuk mendeteksi dan menghitung jumlah kendaraan secara otomatis menggunakan metode kalman filter dan gaussian mixture model," Youngster Physics Journal Vol. 5 No. 1 ISSN : 2302 - 7371, 2016. (Dikutip pada halaman 20).

[26] N. WojkeâĂă, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," arXiv, 2017. (Dikutip pada halaman 20).

[27] Umam and M. Khothiybul, "Analisis matematika algoritma hungaria pada metode penugasan," Undergraduate thesis, Universitas Islam Negeri Maulana Malik Ibrahim, 2009. (Dikutip pada halaman 21).

[28] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," arXiv, 2017. (Dikutip pada halaman 21, 24).

[29] R. Kalman, "A new approach to linear filtering and prediction problems," Journal of Basic Engineering vol. 82 no. Series D pp. 35âĂŞ45, 1960. (Dikutip pada halaman 22).

LAMPIRAN

Gambar 1: Perekaman spidometer 20km/h pada sepedamotor untuk uji akurasi kecepatan

Gambar 2: Perekaman spidometer 30km/h pada sepedamotor untuk uji akurasi kecepatan

Gambar 3: Perekaman spidometer 40km/h pada sepedamotor untuk uji akurasi kecepatan

Gambar 4: Perekaman spidometer 60km/h pada sepedamotor untuk uji akurasi kecepatan

Berikut lampiran dari hasil *output* sistem untuk video fps-30 dan video malam hari, untuk hasil lainnya dapat dilihat pada link berikut: **https://drive.google.com/open?id=1ddosGr09lVyHDEuzPWXKKtLpOOpdGmxp**

Video size 1280 720 – FPS 30

1:motorbike - 1.0
Kecepatan: 135.0 km/h
2:motorbike - 18.0
Kecepatan: 90.0 km/h
3:car - 50.0
Kecepatan: 101.25 km/h
4:motorbike - 68.0
Kecepatan: 90.0 km/h
5:motorbike - 54.0
Kecepatan: 39.51219512195122 km/h
6:motorbike - 102.0
Kecepatan: 90.0 km/h
7:motorbike - 129.0
Kecepatan: 810.0 km/h
8:car - 71.0
Kecepatan: 33.75 km/h
9:motorbike - 163.0
Kecepatan: 324.0 km/h
10:car - 175.0
Kecepatan: 810.0 km/h
11:motorbike - 182.0
Kecepatan: 95.29411764705883 km/h
12:car - 153.0
Kecepatan: 42.63157894736842 km/h
13:car - 162.0
Kecepatan: 22.19178082191781 km/h
14:motorbike - 213.0
Kecepatan: 50.625 km/h
15:car - 204.0
Kecepatan: 37.674418604651166 km/h
16:car - 252.0
Kecepatan: 180.0 km/h
17:motorbike - 261.0

Kecepatan: 810.0 km/h
18:motorbike - 256.0
Kecepatan: 85.26315789473684 km/h
19:car - 158.0
Kecepatan: 39.51219512195122 km/h
20:motorbike - 307.0
Kecepatan: 70.43478260869566 km/h
21:car - 208.0
Kecepatan: 28.928571428571427 km/h
22:motorbike - 318.0
Kecepatan: 101.25 km/h
23:motorbike - 320.0
Kecepatan: 540.0 km/h
24:car - 282.0
Kecepatan: 52.25806451612903 km/h
25:car - 319.0
Kecepatan: 43.78378378378378 km/h
26:motorbike - 337.0
Kecepatan: 64.8 km/h
27:motorbike - 370.0
Kecepatan: 180.0 km/h
28:car - 311.0
Kecepatan: 34.46808510638298 km/h
29:car - 316.0
Kecepatan: 39.51219512195122 km/h
30:motorbike - 378.0
Kecepatan: 85.26315789473684 km/h
31:motorbike - 388.0
Kecepatan: 101.25 km/h
32:motorbike - 380.0
Kecepatan: 57.857142857142854 km/h
33:car - 248.0
Kecepatan: 31.764705882352942 km/h
34:truck - 398.0

Kecepatan: 77.14285714285714 km/h
35:motorbike - 402.0
Kecepatan: 124.61538461538461 km/h
36:car - 410.0
Kecepatan: 231.42857142857142 km/h
37:car - 383.0
Kecepatan: 35.21739130434783 km/h
38:car - 375.0
Kecepatan: 37.674418604651166 km/h
39:motorbike - 447.0
Kecepatan: 270.0 km/h
40:motorbike - 442.0
Kecepatan: 95.29411764705883 km/h
41:motorbike - 456.0
Kecepatan: 108.0 km/h
42:car - 393.0
Kecepatan: 43.78378378378378 km/h
43:car - 421.0
Kecepatan: 46.285714285714285 km/h
44:motorbike - 488.0
Kecepatan: 540.0 km/h
45:car - 464.0
Kecepatan: 27.93103448275862 km/h
46:motorbike - 511.0
Kecepatan: 70.43478260869566 km/h
47:motorbike - 521.0
Kecepatan: 28.928571428571427 km/h
48:car - 513.0
Kecepatan: 49.09090909090909 km/h
49:car - 479.0
Kecepatan: 25.3125 km/h
50:motorbike - 583.0
Kecepatan: 124.61538461538461 km/h
51:car - 526.0

Kecepatan: 34.46808510638298 km/h
52:car - 519.0
Kecepatan: 37.674418604651166 km/h
53:motorbike - 624.0
Kecepatan: 810.0 km/h
54:car - 545.0
Kecepatan: 49.09090909090909 km/h
55:motorbike - 635.0
Kecepatan: 202.5 km/h
56:motorbike - 642.0
Kecepatan: 77.14285714285714 km/h
57:motorbike - 638.0
Kecepatan: 57.857142857142854 km/h
58:motorbike - 660.0
Kecepatan: 124.61538461538461 km/h
59:car - 571.0
Kecepatan: 46.285714285714285 km/h
60:truck - 698.0
Kecepatan: 1620.0 km/h
61:motorbike - 703.0
Kecepatan: 95.29411764705883 km/h
62:car - 695.0
Kecepatan: 43.78378378378378 km/h
63:motorbike - 745.0
Kecepatan: 540.0 km/h
64:motorbike - 739.0
Kecepatan: 115.71428571428571 km/h
65:car - 651.0
Kecepatan: 45.0 km/h
66:car - 655.0
Kecepatan: 42.63157894736842 km/h
67:motorbike - 764.0
Kecepatan: 405.0 km/h
68:motorbike - 761.0

Kecepatan: 85.26315789473684 km/h
69:car - 682.0
Kecepatan: 45.0 km/h
70:car - 702.0
Kecepatan: 36.0 km/h
71:motorbike - 788.0
Kecepatan: 64.8 km/h
72:motorbike - 786.0
Kecepatan: 52.25806451612903 km/h
73:motorbike - 784.0
Kecepatan: 52.25806451612903 km/h
74:car - 822.0
Kecepatan: 1620.0 km/h
75:motorbike - 803.0
Kecepatan: 62.30769230769231 km/h
76:truck - 854.0
Kecepatan: 180.0 km/h
77:car - 772.0
Kecepatan: 43.78378378378378 km/h
78:truck - 868.0
Kecepatan: 135.0 km/h
79:motorbike - 888.0
Kecepatan: 270.0 km/h
80:truck - 884.0
Kecepatan: 81.0 km/h
81:car - 850.0
Kecepatan: 33.06122448979592 km/h
82:car - 876.0
Kecepatan: 21.31578947368421 km/h
83:car - 900.0
Kecepatan: 42.63157894736842 km/h
84:car - 898.0
Kecepatan: 27.0 km/h
85:motorbike - 950.0

Kecepatan: 405.0 km/h
86:car - 890.0
Kecepatan: 36.81818181818182 km/h
87:car - 966.0
Kecepatan: 231.42857142857142 km/h
88:car - 901.0
Kecepatan: 46.285714285714285 km/h
89:car - 929.0
Kecepatan: 31.153846153846153 km/h
90:car - 933.0
Kecepatan: 31.764705882352942 km/h
91:car - 1022.0
Kecepatan: 405.0 km/h
92:car - 1007.0
Kecepatan: 30.566037735849058 km/h
93:motorbike - 1038.0
Kecepatan: 324.0 km/h
94:motorbike - 1048.0
Kecepatan: 81.0 km/h
95:car - 991.0
Kecepatan: 38.57142857142857 km/h
96:car - 988.0
Kecepatan: 36.81818181818182 km/h
97:motorbike - 1062.0
Kecepatan: 231.42857142857142 km/h
98:car - 1016.0
Kecepatan: 28.42105263157895 km/h
99:car - 1078.0
Kecepatan: 108.0 km/h
1765 frames 0.3853368687562159 s/frame

NB :	
Merah ~	**37**
Biru >= 60	**15**
Hijau 31 > x > 60	**39**
Kuning <=30	**8**
Jumlah Data	**99**

Video size 1280 720 – MALAM

1:car - 16.0
Kecepatan: 17.088607594936708 km/h
2:car - 22.0
Kecepatan: 33.75 km/h
3:car - 23.0
Kecepatan: 30.337078651685392 km/h
4:car - 111.0
Kecepatan: 36.0 km/h
5:car - 147.0
Kecepatan: 35.526315789473685 km/h
6:car - 170.0
Kecepatan: 900.0 km/h
7:car - 175.0
Kecepatan: 180.0 km/h
8:car - 165.0
Kecepatan: 40.298507462686565 km/h
9:car - 201.0
Kecepatan: 385.7142857142857 km/h
10:car - 186.0
Kecepatan: 20.454545454545453 km/h
11:car - 222.0
Kecepatan: 38.028169014084504 km/h
12:car - 237.0
Kecepatan: 15.168539325842696 km/h
13:truck - 281.0
Kecepatan: 79.41176470588235 km/h
14:car - 271.0
Kecepatan: 46.55172413793103 km/h
15:car - 298.0
Kecepatan: 84.375 km/h
1897 frames 0.3404661773315655 s/frame

NB:	
Merah ~	**3**
Biru >= 60	**2**
Hijau 31 > x > 60	**6**
Kuning <=30	**4**
Jumlah Data	**15**

BIOGRAFI PENULIS

Selvy Andy Wijaya, lahir pada 10 April 1997 di Malang, Jawa Timur. Penulis lulus dari SMP Negeri 3 Malang pada tahun 2012 kemudian melanjutkan pendidikan ke SMA Negeri 10 Malang hingga akhirnya lulus pada tahun 2015. Penulis kemudian melanjutkan pendidikan S-1 ke Departemen Teknik Komputer, FTE-ITS Surabaya. Saat di kuliah penulis aktif menjadi staff HIMATEKTRO ITS 2017/2018. Penulis juga aktif menjadi Asisten laboratorium Telematika B201 hingga saat ini. Selama masa kuliah penulis aktif dalam organisasi dan juga aktif dalam *development group network* di Lab B201. Bagi pembaca yang memiliki kritik, saran atau pertanyaan mengenai tugas akhir ini dapat menghubungi penulis melalui email selvyandywijaya@gmail.com.

Halaman ini sengaja dikosongkan