# LAPORAN TUGAS AKHIR

# IMPLEMENTASI PENGOLAHAN CITRA DIGITAL UNTUK MENGHITUNG JUMLAH KENDARAAN MASUK PADA JALAN RAYA

Diajukan guna melengkapi sebagian syarat dalam mencapai gelar Sarjana Strata Satu (S1)


Disusun Oleh:

Nama : Ivan Prasetyo Nugroho

N.I.M. : 41420120059

Pembimbing: Rachmat Muwardi, B.Sc., S.T., M.Sc.


**PROGRAM STUDI TEKNIK ELEKTRO**
**FAKULTAS TEKNIK**

**UNIVERSITAS MERCU BUANA**

**JAKARTA**

**2022**

# BAB I
# PENDAHULUAN

## 1.1 Latar Belakang

Semakin majunya perkembangan teknologi saat ini, menjadikan pengaplikasian teknologi memasuki berbagai ranah aspek kehidupan. Salah satunya ialah di bidang tata kelola kota, khususnya pada pengaturan lalu lintas di jalan raya perkotaan. Semakin banyaknya pemilik kendaraan bermotor, berefek kepada semakin padat pula kendaraan bermotor di jalan raya. Kepadatan kendaraan bermotor pada umumnya terletak pada persimpangan karena kendaraan bermotor harus bergantian melewati persimpangan secara bergantian dengan kendaraan bermotor dari arah yang lain, di mana pengaturannya dilakukan oleh lampu lalu lintas (*traffic light*). Permasalahan yang timbul dari gambaran kasus pada paragraf di atas ialah, pada persimpangan akan muncul antrean panjang pada daerah persimpangan sehingga menimbulkan kemacetan. Sebab kemacetan yang terjadi selain dari banyaknya jumlah kendaraan bermotor yang lewat, ialah dari kurang efektifnya sistem pengaturan lalu lintas pada daerah persimpangan. Efek dari kurang efektifnya suatu sistem pengaturan lalu lintas ialah, terjadi kepadatan di salah satu atau lebih dari masing-masing sisi persimpangan.

Penelitian terkait deteksi objek pada citra digital sudah banyak dilakukan salah satunya ialah penelitian pertama oleh Aldhiyatika Amwin (2021) untuk mendeteksi dan mengklasifikasikan objek mobil, bus, truk, becak, dan sepeda motor dengan masukan kamera CCTV, serta mengevaluasi hasil deteksi sistem yang dibuat dengan menggunakan model YOLOv3. Selanjutnya ialah penelitian yang dilakukan oleh Vuong Xuan Can et al (2021) yang bertujuan hampir sama dengan penelitian pertama namun dengan menggunakan model YOLOv4, pada penelitian ini model YOLOv4 dibandingkan hasil deteksinya dengan model algoritma lain seperti Haar Cascade dan MOG dengan kesimpulan model YOLOv4 memiliki kinerja paling baik. Penelitian berikutnya ialah penelitian oleh Jeong-ah Kim et al (2020) yang bertujuan membandingkan performa hasil deteksi dari model algoritma YOLOv4, SSD, dan Faster-RCNN dengan kesimpulan YOLOv4 memiliki performa terbaik dibadingkan SSD dan Faster-RCNN. Selanjutnya ialah penelitian oleh Yan Miao et al (2020) yang bertujuan untuk mengukur dan membandingkan performa dari model algoritma YOLOv3, SSD dan Faster-RCNN dalam keadaan video dengan pencahayaan yang kurang (gelap), dan didapatkan bahwa YOLOv3 mendapatkan performa terbaik dengan AP sebesar 90,66% pada *frame rate* 30,03fps. Penelitian terakhir ialah dilakukan oleh Yu Zhang (2022) yang bertujuan untuk mengukur performa dari model algoritma YOLOv5 yang ditambahkan metode *data enhancement* bernama *Flip-Mosaic*, dimana dapat disimpulkan bahwa performa YOLOv5 dengan tambahan *Flip-Mosaic* memiliki performa lebih baik dibandingkan dengan yang tidak.

Melihat permasalahan yang terjadi, dibutuhkan suatu sistem di mana suatu sistem tersebut dapat menghitung kendaraan bermotor yang akan menuju ke suatu persimpangan dengan memanfaatkan proses pengolahan citra digital dari kamera yang dipasang pada jalan raya. Diharapkan dengan adanya sistem tersebut, dapat digunakan untuk peningkatan efisiensi dan efektivitas sistem pengaturan lalu lintas yang sudah ada. Sistem penghitung kendaraan bermotor menghasilkan data berupa angka jumlah kendaraan bermotor yang akan lewat di suatu persimpangan. Data tersebut kemudian digunakan sebagai masukan suatu sistem lanjutan untuk memperhitungkan durasi menyala lampu merah, kuning, dan hijau pada suatu sisi persimpangan jalan. Data terukur juga dapat digunakan sebagai masukan ke dinas pemerintahan terkait dalam mengatur dan mengontrol jumlah kendaraan yang akan melewati suatu area.

## 1.2 Rumusan Masalah

Pada pembuatan Tugas Akhir ini terdapat beberapa permasalahan yang timbul, yaitu sebagai berikut:

1. Mengapa diperlukan suatu sistem yang dapat mendeteksi dan menghitung mobil secara otomatis?
2. Bagaimana proses yang dilakukan untuk mendeteksi dan menghitung mobil dari suatu masukan citra digital?
3. Apakah data hitung dari sistem yang telah dibuat memiliki ketepatan dengan data sebenarnya?
4. Bagaimana dan apakah proses yang dilakukan untuk menilai kinerja dari sistem yang telah dibuat?
5. Apa saja komponen evaluasi untuk mengetahui performa dari sistem yang dibuat dan berapakah nilainya?

## 1.3 Tujuan dan Manfaat

Tujuan dan manfaat dirumuskan untuk memberikan arah kegiatan pembuatan sistem sehingga dapat dijadikan ukuran tingkat keberhasilannya. Tujuan dan manfaat dirumuskan meliputi tujuan umum pembuatan Tugas Akhir. Tujuan dan manfaat yang dimaksudkan adalah sebagai berikut:

1. Diharapkan sistem dapat membantu dalam menghitung jumlah mobil pada jalan raya/tol, sebagai masukan dalam menghitung tingkat kepadatan lalu lintas.
2. Sistem yang terimplementasi diharapkan dapat mendeteksi dan menghitung mobil dari masukan citra digital.
3. Sistem diharapkan dapat menghasilkan data hitung yang tepat atau mendekati dengan data sebenarnya.
4. Sistem yang terimplementasi diharapkan dapat memiliki kinerja yang baik dalam mendeteksi dan menghitung mobil yang dibuktikan dengan proses evaluasi.
5. Performa dari sistem yang telah dibuat dapat diukur dengan beberapa komponen/variabel evaluasi.

## 1.4 Batasan Masalah

Pada pembuatan Tugas Akhir ini terdapat permasalahan yang lebih fokus, maka dilakukan suatu pembatasan masalah, diantaranya:

1. Sistem yang penulis implementasi merupakan untuk kepentingan edukasi sebagai pengaplikasian materi yang penulis telah peroleh saat kuliah.
2. Kendaraan yang dideteksi ialah mobil saja.
3. Sistem yang penulis implementasi menggunakan algoritma YOLO (*You Only Look Once*) untuk mendeteksi objek pada masukan citra digital.
4. Data citra digital pada pengujian sistem, menggunakan data *video* yang diambil oleh penulis memiliki resolusi 720p dan *frame rate* 30fps.
5. Pengujian kinerja sistem menggunakan metode *Confussion Matrix*.

## 1.5 Metodologi Penelitian

Dalam proses untuk menyelesaikan tugas akhir, diperlukan serangkaian proses agar pengerjaan tugas akhir ini dapat selesai dengan hasil yang sebaik mungkin. Berikut ini ialah proses-proses yang akan dilakukan untuk membangun sebuah sistem pendeteksi dan penghitung jumlah kendaraan pada jalan raya:

a. Mengambil *video* sebagai masukan sistem.
b. Memilih algoritma *object detection* yang akan dipakai pada sistem.
c. Membuat desain diagram alur/*flowchart* dan diagram blok sebagai acuan dalam pembuatan sistem.
d. Membuat implementasi kode untuk menghasilkan sistem pendeteksi dan penghitung mobil mengacu pada diagram alur yang telah dirancang.
e. Membuat implementasi kode untuk mengevaluasi atau menguji hasil dari sistem yang telah dibuat dengan menggunakan metode *confussion matrix* dimana dilakukan perhitungan nilai akurasi, *recall*, dan *precision*.
f. Melakukan proses pengujian apakah data terukur dari sistem sesuai dengan data aktual.
g. Membuat laporan perancangan dan pengujian yang dilampirkan pada laporan tugas akhir.

## 1.6 Sistematika Penulisan

Laporan tugas akhir ini terdiri dari 5 bab, antara lain :

a. Bab I Pendahuluan

Pada bab ini akan dijelaskan mengenai latar belakang masalah, tujuan, manfaat, batasan masalah, metodologi penelitian dan sistematika penulisan dari tugas akhir yang dikerjakan.

b. Bab II Landasan Teori

Bab ini terdiri dari dua sub-bab yang antara lain ialah tinjauan pustaka dan dasar teori. Pada tinjauan pustaka akan dipaparkan beberapa penelitian yang sudah pernah dilakukan dengan topik terkait tugas akhir yang penulis kerjakan. Dan pada sub-bab dasar teori, akan dipaparkan penjelasan tentang bahasa pemrograman, algoritma *object detection* yang digunakan, dan metode evaluasi yang digunakan untuk menguji sistem yang penulis kerjakan.

c. Bab III Metodologi Penelitian / Perancangan Alat dan Sistem

Pada bab ini akan dijelaskan dijabarkan blok diagram, implementasi kode, pengujian sistem dengan metode *confussion matrix* yang disertai dengan proses perhitungan akurasi, *recall*, dan *precision*.

d. Bab IV Hasil dan Pembahasan

Pada bab ini akan dipaparkan analisis data-data yang didapatkan dari hasil pengujian.

e. Bab V Penutup

Menjelaskan mengenai kesimpulan akhir penelitian dan saran-saran yang direkomendasikan untuk perbaikan proses pengujian dan pengerjaan tugas akhir selanjutnya.

# BAB II
# LANDASAN TEORI

## 2.1 Tinjauan Pustaka

Tinjauan pustaka berisi informasi tentang teori dan uraian teori yang menjadi dasar referensi TA (Tugas Akhir), berupa pembahasan hasil-hasil penelitian sebelumnya yang sesuai dengan tujuan yang telah ditetapkan. Tinjauan pustaka ini dapat membantu mahasiswa dalam menyusun kerangka berpikir ilmiah, merumuskan *hipotesis*, dan menetapkan metode penelitian, pengumpulan dan analisis data. Kajian pustaka dapat merujuk pada hasil penelitian seperti laporan tugas akhir dan jurnal.

Penelitian pertama yang dilakukan oleh Amwin, A., (2021). Dengan judul "DETEKSI DAN KLASIFIKASI KENDARAAN BERBASI ALGORITMA *YOU ONLY LOOK ONCE* (YOLO)". Tujuan dari penelitian ini ialah untuk mendeteksi dan mengklasifikasikan kendaraan dengan menggunakan masukan video dari *Closed Circuit Television* (CCTV). Pada penelitian terdapat 5 *class* dataset yang akan dideteksi dan diklasifikasikan, yaitu mobil, bus, truk, becak, dan sepeda motor menggunakan algoritma YOLOv3. Hasil dari penelitian tersebut menunjukkan bahwa hasil *Average Precision* (AP) pada mobil sebesar 99,88%, pada sepeda motor 97,79%, becak sebesar 100%, truk sebesar 100%, dan bus 99,09%. Sedangkan nilai *mean Average Precision* (mAP) sebesar 99,35% dengan waktu proses selama 4 detik. Pada penelitian tersebut dapat disimpulkan bahwa semakin tinggi nilai mAP, maka akan semakin akurat pula proses pendeteksian objek. Selain itu, hasil deteksi objek dapat dipengaruhi juga oleh kualitas video, semakin tinggi kualitas video maka hasil klasifikasi dan lokalisasi dari *bounding boxes* akan semakin tepat.

Penelitian kedua, yaitu dilakukan oleh Oktaviano, E.K., (2020). Dengan judul "*INDONESIAN LICENSE PLATE RECOGNITION USING YOLOV3 AND OPTICAL CHARACTER RECOGNITION*". Tujuan dari penelitian ini ialah mendeteksi dan mengambil data dari pelat nomor mobil dari masukan gambar dari kamera. Penelitian tersebut menggunakan algoritma YOLOv3 dan *Optical Character Recognition* (OCR) untuk mendeteksi pelat nomor serta karakter di dalamnya. Pada penelitian ini, dilakukan proses evaluasi dengan menghitung data akurasi dan mAP dari sistem. Hasilnya yaitu didapatkan mAP sebesar 88,87% dan akurasi sebesar 80,2% dengan waktu proses kurang dari 0,4 detik.

Selanjutnya pada penelitian ketiga yang dilakukan oleh Mahto, Pooja., et al, (2020), dengan judul "*REFINING YOLOV4 FOR VEHICLE DETECTION*". Tujuan dari penelitian ini ialah untuk meningkatkan performa dari algoritma YOLOv4 dan membandingkan dengan YOLOv3. Hasil penelitian ini ditemukan bahwa YOLOv3 memiliki performa deteksi lebih baik dari YOLOv4 dalam mendeteksi objek di pencahayaan yang kurang.

Selanjutnya ialah penelitian keempat, yaitu dilakukan oleh Can, Vuong Xuan., et al, (2021), yang berjudul "*VEHICLE DETECTION AND COUNTING UNDER MIXED TRAFFIC CONDITIONS IN VIETNAM USING YOLOV4*". Tujuan dari penelitian ini ialah untuk menghitung dan mengklasifikasikan jenis kendaraan yang lewat, yaitu terdapat *class* mobil, bus, truk, sepeda motor, dan sepeda. Pada penelitian ini algoritma YOLOv4 dibandingkan dengan algoritma lain, yaitu Haar Cascade dan MOG. Lalu dihitung *Quality* dari hasil deteksinya dengan rumus:

$$Quality = \frac{TP}{TP + FP + FN}$$

Lalu didapatkan hasil dan divisualisasikan menjadi tabel berikut:

Tabel 2.1 Hasil dari percobaan

(Can, Vuong Xuan., et al, 2021)

| Algoritma | Quality (%) | |
|---|---|---|
| | Deteksi Saat Jam Sibuk | Deteksi Saat Bukan Jam Sibuk |
| Haar Cascade | 82.9 | 71.6 |
| MOG | 80.1 | 72.3 |
| YOLOv4 | 99.2 | 99.4 |

Selanjutnya ialah penelitian kelima, yaitu dari B, Chethan Kumar., et al, (2020), yang berjudul "*YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications*". Tujuan dari penelitian ini ialah untuk membandingkan performa antara YOLOv3 yang diimplementasi menggunakan *dataset* gambar dengan YOLOv4 yang diimplementasi menggunakan *dataset* video yang di dalam gambar dan video tersebut mengandung banyak jenis objek (mobil, manusia, sepeda, dan lain-lain). Dan didapatkan hasil yaitu YOLOv3 memiliki tingkat akurasi sebesar 98% dalam mendeteksi *dataset* gambar dan YOLOv4 memiliki akurasi sebesar 99% dalam mendeteksi *dataset* video.

Selanjutnya penelitian keenam, yaitu dari S, Asha C., et al, (2018), yang berjudul "*Vehicle Counting for Traffic Management System using YOLO and Correlation Filter*". Penelitian tersebut bertujuan untuk membuat suatu sistem yang dapat mengukur kepadatan lalu lintas dengan menggunakan YOLO sebagai algoritma deteksi objek dan *correlation filter* untuk meningkatkan performa deteksi objek terutama dalam hal akurasi. Pada penelitian ini, diujikan 7 video dengan jumlah *frame* yang berbeda. Dan hasil penelitian tersebut menghasilkan data dengan tabel sebagai berikut:

Tabel 2.2 Hasil dari percobaan

(S, Asha C., et al, 2018)

| ***File* Video** | **Total *Frame*** | **Total kendaraan (*ground-truth*)** | **Total kendaraan (Hasil Deteksi)** | **Hilang/Deteksi Berulang/*Error*** | ***Precision* (%)** | ***Recall* (%)** | ***F-score*** | **Akurasi (%)** |
|---|---|---|---|---|---|---|---|---|
| 1.mp4 | 899 | 10 | 10 | 0/0/0 | 100,0 | 100,0 | 100,0 | 100% |
| 2.mp4 | 715 | 16 | 15 | 1/0/1 | 97,7 | 100,0 | 96,7 | 93.7% |
| 3.mp4 | 845 | 17 | 18 | 0/1/1 | 100,0 | 94,4 | 97,1 | 94.4% |
| 4.mp4 | 3598 | 58 | 55 | 3/0/3 | 94,8 | 100,0 | 97,3 | 94.8% |
| 5.mp4 | 2100 | 25 | 23 | 2/0/2 | 92,0 | 100,0 | 95,8 | 92.0% |
| 6.mp4 | 5528 | 67 | 67 | 1/1/2 | 98,5 | 98,5 | 98,5 | 97.0% |
| 7.mp4 | 2999 | 28 | 28 | 0/0/0 | 100 | 100 | 100 | 100% |

Selanjutnya ialah penelitian ketujuh yaitu yang dilakukan oleh Kim, J., et al, (2020), dengan judul "*Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition*". Penelitian ini bertujuan untuk membandingkan performa algoritma deteksi objek Faster-RCNN, YOLOv4, dan SSD untuk mendeteksi dan mengklasifikasikan objek berupa kendaraan berjenis mobil, *mini van, big van, mini truck, truck,* dan mobil *compact*. Dan dapat disimpulkan dari penelitian tersebut bahwa dari ketiga algoritma di atas, YOLOv4 ialah algoritma yang memiliki performa terbaik terutama apabila YOLOv4 dijalankan dengan GPU tambahan. Hasil performa dari ketiga algoritma di atas ialah sebagai berikut:

Tabel 2.3 Hasil dari percobaan

(Kim, J., et al, 2020)

| **Model** | **F1-score** | **Precision** | **Recall** | **mAP** |
|---|---|---|---|---|
| YOLOv4 | 0.96 | 0.93 | 0.98 | 98.19 |
| SSD | 0.88 | 0.90 | 0.87 | 90.56 |
| Faster-RCNN | 0.90 | 0.86 | 0.94 | 93.40 |

Penelitian kedelapan, yaitu dilakukan oleh Wu, T.-H., et al, (2021), dengan judul "*Real-Time Vehicle and Distance Detection Based on Improved YOLO v5 Network*". Penelitian ini bertujuan untuk meningkatkan performa dari model deteksi objek YOLOv5 dengan membuat struktur *neural network* baru yang bernama YOLOv5-Ghost. Penelitian ini mengimplementasikan YOLOv5-Ghost ke dalam suatu sistem yang terpasang pada *autonomous car* yang diperuntukkan mengukur jarak dari mobil ke objek di depannya, yaitu mobil, bus, sepeda motor, sepeda, truk, dan manusia.

Penelitian kesembilan, yang dilakukan oleh Miao, Y., et al, (2020), yang berjudul "*A Nighttime Vehicle Detection Method Based on YOLOv3*". Penelitian ini bertujuan untuk mendeteksi objek mobil di jalan raya dalam keadaan gelap pada malam hari. Selain itu, pada penelitian ini dibandingkan pula performa antara YOLOv3, Faster R-CNN, dan SSD dalam mendeteksi objek pada keadaan cahaya yang minimal. Dan hasil dari penelitian tersebut menyebutkan model YOLOv3 memiliki performa terbaik dibandingkan dengan model yang lain dengan nilai AP sebesar 90,66% pada *frame rate* 30,03fps, sedangkan Faster-RCNN mendapatkan nilai AP 80,52% dengan 9,77fps, dan SSD mendapatkan AP 90,45% dengan 25,94fps.

Selanjutnya ialah penelitian kesepuluh yang dilakukan oleh Zhang, Yu., et al, (2022), yang berjudul "*Real-Time Vehicle Detection Based on Improved YOLO v5*". Pada penelitian ini, sistem dibuat dengan mengimplementasi algoritma YOLOv5 dan dengan ditambahkan metode *data enhancement* bernama *Flip-Mosaic*. Dari penelitian ini didapatkan bahwa deteksi objek dari YOLOv5 tanpa *data enhancement* dan dengan *data enhancement* menghasilkan hasil yang berbeda. YOLOv5 dengan tambahan *Flip Mosaic* dapat meningkatkan akurasi pendeteksian objek.

Penelitian selanjutnya, yaitu penelitian kesebelas yang dilakukan oleh Tariq, A., et al, (2021), dengan judul "*Real Time Vehicle Detection and Colour Recognition using tuned Features of Faster-RCNN*". Pada penelitian ini bertujuan untuk mendeteksi warna dari mobil yang terekam kamera pada jalan raya. Penelitian ini mengimplementasikan model Faster-RCNN sebagai algoritma deteksi objek. Hasil dari penelitian ini menunjukkan hasil akurasi dari Faster R-CNN ialah 95,31% dengan keaadan cuaca berawan atau mendung.

Penelitian kedua belas, yaitu dilakukan oleh Suhao, L., et al, (2018), yang berjudul "*Vehicle type detection based on deep learning in traffic scene*". Tujuan dari penelitian ini ialah untuk mendeteksi tipe dari kendaraan yang terekam kamera dengan mengklasifikasikannya menjadi mobil, minibus, dan SUV. Pada penelitian ini, *dataset* diambil dari MIT dan *Caltech car* dengan mengimplementasikan model Faster R-CNN sebagai algoritma deteksi objeknya. Pada penelitian ini, diimplementasikan *depth learning network* model bernama ZF model (*Zeiler and Fergus*) dan VGG16 yang digunakan untuk melatih dan *tuning* parameter.

Penelitian selanjutnya, yaitu penelitian ketiga belas yang dilakukan oleh Manana, M., et al, (2018), dengan judul "*Preprocessed Faster RCNN for Vehicle Detection*". Penelitian ini bertujuan untuk meminimalkan waktu *training* pada model yang ditujukan untuk mendeteksi kendaraan bermotor. Pada penelitian ini model Faster R-CNN diimplementasikan dengan metode *preprocessing* sebagai metode untuk mempercepat waktu *training*. Penelitian ini menghasilkan waktu *training* 50% lebih cepat dibandingkan model yang tidak menggunakan metode *preprocessing*.

Penelitian keempat belas, yaitu dilakukan oleh Firmansyah, Riza A., et al, (2018), yang berjudul "PEMBUATAN HAAR-CASCADE DAN *LOCAL BINARY PATTERN* SEBAGAI SISTEM PENDETEKSI HALANGAN PADA *AUTOMATIC GUIDED VEHICLE*". Tujuan dari penelitian ini ialah untuk membuat *Automatic Guided Vehicle* (AGV) agar AGV tersebut dapat berjalan dengan menghindari objek di depannya. Pada implementasinya, deteksi objek pada penelitian ini menggunakan algoritma Haar-Cascade dengan menggunakan *dataset training* sebanyak 300 citra positif dan 2317 citra negatif. Dari hasil percobaan penelitian ini, tingkat keberhasilan atau akurasi dari deteksi objek menggunakan Haar-Cascade ialah sebesar 81,7%.

Penelitian terakhir, yaitu penelitian kelima belas yang dilakukan oleh Kasper-Eulaers, Margrit., et al, (2021), dengan judul "*Short Communication: Detecting Heavy Goods Vehicle in Rest Areas in Winter Conditions Using YOLOv5*". Pada penelitian ini bertujuan untuk mendeteksi parkir kosong pada area parkir kendaraan berat dengan kondisi musim bersalju. Sistem menggunakan kamera termal untuk mendeteksi kendaraan berat dengan memanfaatkan perbedaan suhu yang terekan oleh kamera. Pada penelitian ini, terimplementasi peningkatan performa dengan cara menambah *dataset training* berupa gambar tampak depan dan tampak belakang dari kendaraan berat. Dan dari penelitian ini, dihasilkan peningkatan performa akurasi dalam pendeteksian kendaraan berat setelah ditambahkan *dataset training*.

Tabel 2.4 Perbandingan Tinjauan Pustaka

| **No.** | **Judul** | **Peneliti** | **Spesifikasi Sistem** | |
|---|---|---|---|---|
| | | | ***Hardware*** | ***Interface*** |
| 1 | Deteksi Dan Klasifikasi Kendaraan Berbasis Algoritma *You Only Look Once* (YOLO) | 1. Aldhiyatika Amwin | • CCTV<br>• Laptop | *Prototype* |
| 2 | *Indonesian License Plate Recognition Using YOLOv3 And Optical Character Recognition* | 1. Edra Kevin Oktaviano | • Laptop<br>• Kamera | Simulasi |

| | | | | |
|---|---|---|---|---|
| 3 | *Refining YOLOv4 For Vehicle Detection* | 1. Pooja Mahto<br>2. Priyamm Garg<br>3. Pranav Seth<br>4. J Panda | • Laptop<br>• CCTV | *Prototype* |
| 4 | *Vehicle Detection And Counting Under Mixed Traffic Conditions In Vietnam Using YOLOv4* | 1. Vuong Xuan Can<br>2. Phan Xuan Vu<br>3. Mou Rui-fang<br>4. Vu Trong Thuat<br>5. Vu Van Duy<br>6. Nguyen Duy Noi | • Laptop<br>• Kamera | Simulasi |
| 5 | *YOLOv3 and YOLOv4: Multiple Object Detection for Surveillance Applications* | 1. Chethan Kumar B<br>2. Punitha R<br>3. Mohana | • Laptop<br>• Kamera | Simulasi |
| 6 | *Vehicle Counting for Traffic Management System using YOLO and Correlation Filter* | 1. Asha C S<br>2. A V Narasimhadhan | • Laptop<br>• CCTV | *Prototype* |
| 7 | *Comparison of Faster-RCNN, YOLO, and SSD for Real-Time Vehicle Type Recognition* | 1. Jeong-ah Kim<br>2. Ju-Yeong Sung<br>3. Se-ho Park | • Laptop<br>• CCTV | *Prototype* |
| 8 | *Real-Time Vehicle and Distance Detection Based on Improved YOLO v5 Network* | 1. Tian-Hao Wu<br>2. Tong-Wen Wang<br>3. Ya-Qi Liu | • Laptop<br>• CARLA Simulator | Simulasi |

| | | | | |
|---|---|---|---|---|
| 9 | *A Nighttime Vehicle Detection Method Based on YOLOv3* | 1. Yan Miao<br>2. Lu Liu<br>3. Fu Liu<br>4. Yun Liu<br>5. Tao Hou | • Laptop<br>• Kamera | Simulasi |
| 10 | *Real-Time Vehicle Detection Based on Improved YOLO v5* | 1. Yu Zhang<br>2. Zhongyin Guo<br>3. Jianqing Wu<br>4. Yuan Tian<br>5. Haotian Tang<br>6. Xinming Guo | • Laptop<br>• CCTV | *Prototype* |
| 11 | *Real Time Vehicle Detection and Colour Recognition using tuned Features of Faster-RCNN* | 1. Abdullah Tariq<br>2. Muhammad Zeeshan Khan<br>3. Muhammad Usman Ghani Khan | • Laptop<br>• Kamera | Simulasi |
| 12 | *Vehicle type detection based on deep learning in traffic scene* | 1. Li Suhao<br>2. Lin Jinzhao<br>3. Li Guoquan<br>4. Bai Tong<br>5. Wang Huiqian<br>6. Pang Yu | • Laptop<br>• Kamera | Simulasi |
| 13 | *Preprocessed Faster RCNN for Vehicle Detection* | 1. Mduduzi Manana<br>2. Chunling Tu<br>3. Pius Adewale Owolawi | • Laptop<br>• Kamera | Simulasi |

| | | | | |
|---|---|---|---|---|
| 14 | Pembuatan Haar-Cascade Dan *Local Binary Pattern* Sebagai Sistem Pendeteksi Halangan Pada *Automatic Guided Vehicle* | 1. Riza Agung Firmansyah<br>2. Enggar Alfianto | • Laptop<br>• Kamera<br>• AGV | *Prototype* |
| 15 | *Short Communication: Detecting Heavy Goods Vehicle in Rest Areas in Winter Condition* | 1. Margrit Kasper-Eulaers<br>2. Nico Hahn<br>3. Stian Berger<br>4. Tom Sebulonsen<br>5. Øystein Myrland<br>6. Per Egil Kummervold | • Laptop<br>• Kamera termal | *Prototype* |

## 2.2. Pengolahan Citra Digital (*Image Processing*)

Gambar 2.1 Ilustrasi *image processing*

Pengolahan citra digital merupakan suatu proses dalam mengolah dan memproses suatu sinyal masukan berupa citra digital yang ditransformasikan menjadi gambar berbeda sebagai keluaran. Pengolahan citra digital dilakukan dengan tujuan memperbaiki atau meningkatkan kualitas citra agar dapat lebih mudah diinterpretasi, dimanipulasi atau dianalisis oleh suatu sistem lanjutan. Berdasarkan tujuan transformasinya, operasi pengolahan citra digital dikelompokkan menjadi sebagai berikut:

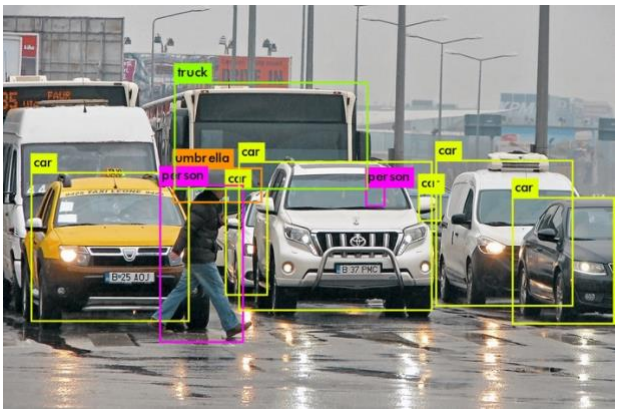
- Peningkatan kualitas citra (*image enhancement*)

  Operasi peningkatan citra merupakan suatu operasi yang bertujuan untuk meningkatkan beberapa fitur tertentu di dalam suatu citra.

- Pemulihan citra (*image restoration*)

  Operasi pemulihan citra merupakan suatu operasi untuk memperbaiki atau mengembalikan kondisi citra ke kondisi sebelumnya yang sudah diketahui yang diakibatkan oleh adanya gangguan yang menyebabkan beberapa fitur di dalam citra mengalami kerusakan atau perubahan yang tidak diinginkan.

## 2.3. Deteksi Objek

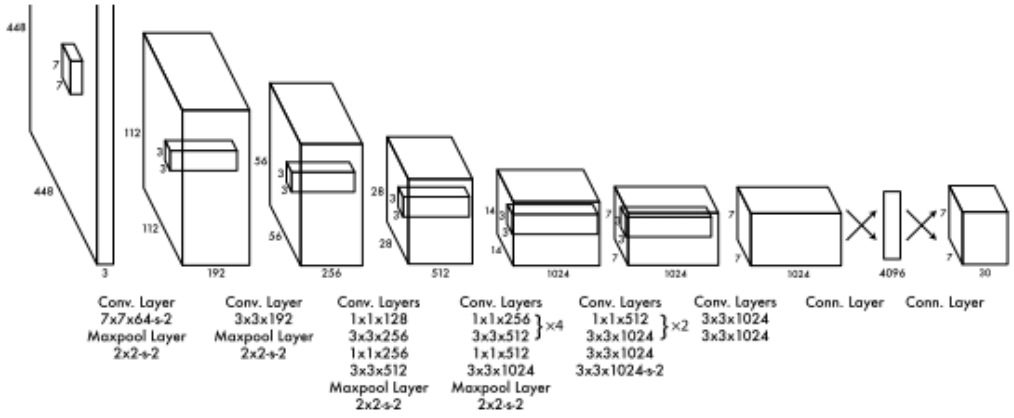



Gambar 2.2 Ilustrasi deteksi objek

Deteksi objek pada pengolahan citra digital merupakan penentuan keberadaan dan posisi suatu objek pada citra. Secara sederhana, deteksi objek ialah melakukan pemindaian pada keseluruhan citra dan menentukan kelas (keberadaan) suatu objek dan lokasi koordinat objek tersebut.

Alur pendeteksian objek ialah dimulai dari masukan, yaitu suatu citra. Citra masukan lalu dimasukkan ke sebuah model atau algoritma pada sistem yang telah dilatih untuk mendeteksi suatu objek. Setelah pemrosesan dari model atau algoritma telah selesai, akan menghasilkan koordinat yang merepresentasikan posisi suatu objek pada citra. Koordinat yang dihasilkan akan menghasilkan sebuah kotak pembatas (*bounding box*) di sekitar objek yang dideteksi. Terdapat beberapa model atau algoritma yang dapat digunakan untuk mendeteksi objek pada citra, yaitu di antaranya ialah *Faster R-CNN, YOLO, Mask R-CNN*, dan lain sebagainya.

## 2.4. *You Only Look Once* (YOLO)




Gambar 2.3 Arsitektur YOLO

*You Only Look Once* (YOLO) merupakan salah satu algoritma deteksi objek secara *real-time* yang berbasis *Convolutional Neural Network* (CNN). YOLO pertama kali dikembangkan oleh Redmon et al. pada tahun 2015. Dalam pendeteksian objek, YOLO hanya menggunakan satu lapisan jaringan syaraf atau biasa disebut dengan jaringan syaraf tunggal (*single neural network*). Jaringan tunggal pada YOLO tersebut akan membagi masukan citra menjadi beberapa *regions* dan memprediksi *bounding boxes* serta kemungkinan pada setiap *regions* secara bersamaan. YOLO memiliki kelebihan, yaitu YOLO dapat mendeteksi suatu objek dengan lebih cepat. Namun dengan kecepatan deteksinya itu pula muncul suatu kelemahan, yaitu YOLO dapat membuat lebih banyak melakukan kesalahan lokalisasi objek pada masukan citra, dan YOLO lemah dalam mendeteksi objek berukuran kecil pada citra. Kelemahan YOLO ini muncul dikarenakan pada tahap deteksi, YOLO tidak melakukan langkah proposal terlebih dahulu.

Algoritma YOLO membagi citra menjadi beberapa sel *grid* (petak kecil) yang memiliki ukuran s x s. Jika suatu bagian dari objek terdeteksi masuk ke dalam suatu sel *grid,* maka sel *grid* tersebut bertanggung jawab untuk melakukan deteksi pada objek tersebut. Tanggung jawab dari sel *grid* tersebut ialah memprediksi lokasi *bounding boxes* dan nilai *confidence score*, serta prediksi klasifikasi *class* dari objek terdeteksi. Nilai *confidence score* pada YOLO didefinisikan oleh rumus:

$$Confidence\ Score = Pr(Object)\ X\ IoU_{pred}^{truth}$$

Apabila pada sel *grid* tidak terdapat objek terdeteksi, maka nilai *confidence score*-nya akan nol. Gabungan banyak dari sel *grid* ini akan membentuk *bounding boxes*. Setiap *bounding boxes* terdiri dari variabel x, y, w, h, dan *confidence score*. Variabel x dan y merupakan nilai titik koordinat pusat *bounding boxes*. Variabel w dan h merupakan nilai lebar dan tinggi *bounding boxes* relatif terhadap ukuran keseluruhan citra. Sedangkan variabel *confidence score* merupakan nilai keyakinan bahwa di dalam suatu *bounding boxes* terdapat objek terdeteksi.

Nilai keyakinan suatu *class* terdeteksi di dalam *bounding boxes* disebut *class confidence score*. *Class confidence score* mengukur nilai keyakinan atau probabilitas terhadap hasil klasifikasi dan lokalisasi objek. *Class confidence score* untuk setiap *bounding boxes* dirumuskan sebagai berikut:

$$\Pr(Class_i|Object) \cdot \Pr(Object) \cdot IoU_{pred}^{truth} = \Pr(Class_i) \cdot IoU_{pred}^{truth}$$

Keterangan:

- $\Pr(Class_i|Object)$ : probabilitas kondisional *class* i.
- $\Pr(Object)$ : probabilitas *class* i.

## 2.5. OpenCV


Gambar 2.4 Logo OpenCV

*Computer vision* ialah suatu kecerdasan buatan yang memungkinkan untuk melakukan pengenalan objek pada suatu citra yang di dapat dari suatu sumber yang dapat berupa gambar atau video. OpenCV ialah salah satu *library* yang digunakan untuk digunakan untuk mengimplementasi pemrograman *computer vision* dengan didukung oleh fungsi-fungsi yang terdapat di *library* OpenCV.

OpenCV didukung di beberapa bahasa pemrograman, diantaranya bahasa C, C++, Python, dan Java. OpenCV dapat dijalankan pada sistem operasi MacOS, Linux maupun Windows.

**2.6. Python**


Gambar 2.5 Logo Python

Python merupakan salah satu bahasa pemrograman dengaan tujuan umum, tingkat tinggi, dan merupakan bahasa *Object Oriented Programming* (OOP) atau bahasa yang berorientasi objek. Python mulai dibuat dan dikembangkan pada akhit tahun 1980 oleh Guido van Rossum di Centrum Wiskunde & Informatica (CWI) di Belanda. Pyhton dikembangkan sebagai penerus bahasa ABC. Hingga saat ini, Python telah merilis beberapa versi dengan versi terbaru ialah 3.10.8. Python merupakan bahasa popular yang digunakan di bidang pembuatan *website, Data Science, Machine Learning, Internet of Things* (IoT), dan banyak bidang-bidang yang lain. Python memiliki banyak *library* yang dapat membatu melakukan pemrograman suatu sistem, yaitu salah satunya ialah *library* OpenCV. Berikut ini ialah beberapa kelebihan bahasa Python:

a. Python memiliki sintaks yang cukup sederhana dibanding bahasa pemrograman lain, seperti Java, C, atau PHP.
b. Python memiliki fleksibilitas, karena dapat dikombinasikan dengan bahasa pemrograman lain.
c. Python memiliki banyak *library* yang dapat integrasikan, sehingga memudahkan untuk membuat atau mengembangkan suatu sistem.

## 2.7. PyCharm

Gambar 2.6 Logo PyCharm

Dalam membuat dan mengembangkan suatu sistem yang mengharuskan untuk menulis kode bahasa pemrograman, dibutuhkan suatu *Integrated Development Environment* (IDE). IDE ialah suatu aplikasi yang memiliki fitur-fitur untuk menuliskan kode bahasa pemrograman dan mengembangkan suatu sistem perangkat lunak. PyCharm ialah salah satu IDE yang digunakan untuk menuliskan kode terkhusus untuk bahasa Python. PyCharm sendiri ialah aplikasi yang dikembangkan oleh perusahaan yang berbasis di Ceko, yaitu JetBrains. PyCharm memiliki fitur-fitur yang dapat mendukung pengembangan sistem perangkat lunak seperti *debugger*, *unit test*, dan dapat terintegrasi dengan sistem *version control* seperti Git. Pycharm bersifat lintas platform, sehingga dapat digunakan di berbagai sistem operasi seperti Windows, macOS, dan Linux.

## 2.8. Video

Video merupakan suatu media elektronik untuk menyampaikan suatu pesan melalui gambar yang bergerak. Video tersusun dari beberapa gambar yang ditampilkan secara bergantian. Kecepatan gambar yang ditampilkan pada video biasa disebut *frame rate* yang memiliki satuan *Frame Per Second* (FPS). FPS merupakan merupakan satuan yang menyatakan berapa banyak gambar yang tertampil pada video dalam 1 detik. Contohnya suatu video memiliki *frame rate* sebesar 30fps, maka video tersebut menampilkan gambar secara bergantian sebanyak 30 gambar dalam 1 detik. Semakin besar nilai FPS video, maka pergerakan objek pada video akan semakin halus. Di masa sekarang, nilai *frame rate* video memiliki beberapa nilai yang beragam, yaitu 24fps, 30fps, 60fps, 70fps, 120fps, dan masih banyak nilai yang lainnya.

# BAB III
# METODOLOGI PENELITIAN / PERANCANGAN ALAT DAN SISTEM

## 3.1 Metodologi Penelitian

Penelitian yang dibuat oleh penulis ini menggunakan metode penelitian eksperimen. Metode eksperimen yang digunakan pada penelitian yang penulis lakukan ialah penelitian dibuat menggunakan bahasa *phyton* sebagai bahasa pemrograman dan PyCharm sebagai *Integrated Development Environment* (IDE). Pada program yang dibuat, akan diimplementasi algoritma deteksi objek menggunakan *library* OpenCV dan YOLO. Lalu, hasil dari sistem yang dibuat akan dievaluasi menggunakan metode *confusion matrix.*

Pertama-tama video berupa video dengan resolusi 360p dengan *frame rate* 30fps akan digunakan sebagai masukan sistem. Sistem akan mendeteksi jumlah mobil dengan menggunakan algoritma YOLO dengan bantuan *library* OpenCV, dan akan menghitung jumlah mobil secara kontinyu selama video diputar. Apabila terdeteksi mobil pada *frame* video masukan, maka akan ditampilkan *bounding boxes* pada objek yang terdeteksi serta dilakukan penambahan jumlah mobil.

Setelah proses dari sistem selesai, maka perlu dilakukan proses evaluasi yang mengukur kualitas dari sistem yang telah dibuat dengan menggunakan metode *confusion matrix* dengan mengambil data dan menghitung variabel-variabel sebagai berikut:

a. Akurasi

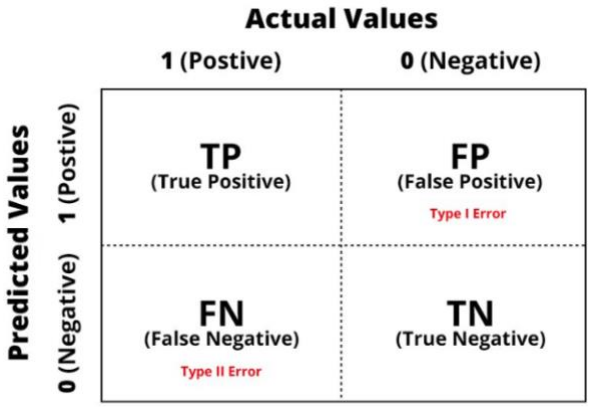
Pada pemodelan klasifikasi objek, akurasi merupakan *metrics* persentase model memprediksi objek dengan benar. Akurasi digunakan sebagai *metrics* untuk melihat performa model secara singkat. Berikut ini ialah rumus untuk menghitung nilai akurasi dari suatu model:

$$\text{Akurasi} = \frac{\text{Jumlah prediksi benar}}{\text{Jumlah prediksi tereksekusi}}$$

Tetapi *metrics* akurasi memiliki kekurangan, yaitu ketika pengklasifikasian model harus mendeteksi *imbalanced data* dan *class* yang memiliki bobot yang sangat tinggi jika salah terprediksi. Contoh kasusnya ialah apabila terdapat 2 *class* dengan nama *class* A dan *class* B. Suatu model berhasil mendeteksi 99 *class* A dengan benar, namun model salah mendeteksi 1 *class* B yang dimana *class* B ini merupakan kasus yang sangat fatal apabila salah dideteksi.

b. *Confusion Matrix*

*Confusion Matrix* atau bisa disebut juga *error matrix* pada dasarnya memberikan informasi perbandingan hasil prediksi yang tereksekusi oleh model dengan hasil prediksi sebenarnya. *Confusion matrix* dapat digambarkan menjadi suatu tabel matriks yang menggambarkan kinerja dari suatu model terhadap data uji yang nilai sebenarnya telah diketahui.




Gambar 3.1 Tabel matriks *confusion matrix*

Berikut ini ialah variabel-variabel terkait pada *confusion matrix*:

- *Positive* (P): sampel positif.
- *Negative* (N): sampel negatif.
- *True Positive* (TP): sampel bernilai positif diprediksi benar.
- *True Negative* (TN): sampel bernilai negatif diprediksi benar.
- *False Positive* (FP): sampel bernilai positif diprediksi salah.
- *False Negative* (FN): sampel bernilai negatif diprediksi salah.

c. *F1-Score*

*F1-Score* adalah *Harmonic Mean* antara nilai *precision* dan *recall*. Nilai rentang dari *F1-Score* sebesar [0, 1]. *F1-Score* bertujuan untuk mengetahui seberapa tepat pengklasifikasian pada suatu model. Berikut ini ialah rumus untuk menghitung nilai *F1-Score*:

$$F1\text{-}Score = 2 \text{ x } \frac{1}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}}$$

d. *Precision* dan *Recall*

Nilai *precision* ialah nilai perbandingan antara total prediksi benar pada sampel positif dengan total sampel positif yang diprediksi oleh model. Nilai *precision* menunjukkan seberapa baik model dalam memprediksi sampel positif dengan benar. Berikut ini ialah rumus dari *precision*:

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Nilai *recall* ialah nilai perbandingan antara total prediksi benar pada sampel positif dengan total smpel positif yang terdeteksi. Nilai *recall* menunjukkan seberapa baik model dalam memprediksi suatu *class* dengan benar. Berikut ini ialah rumus dari *recall*:

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

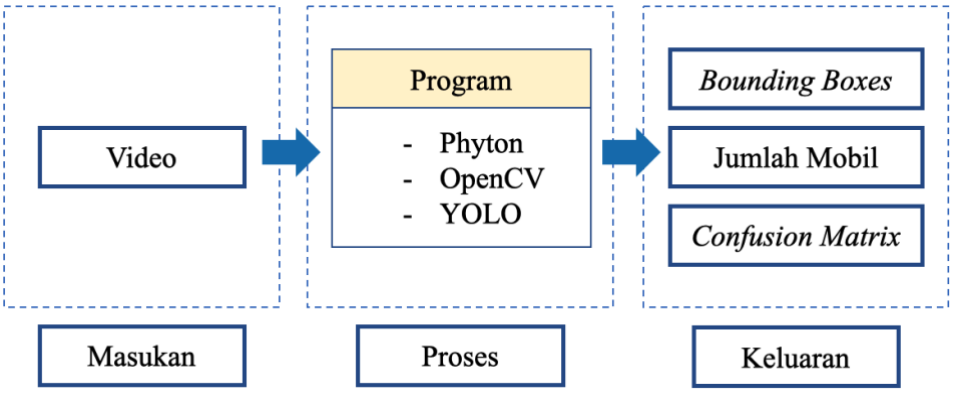
Nilai antara *precision* dan *recall* saling berhubungan, berikut ini ialah hubungan antara kedua nilai tersebut terhadap penilaian dari performa suatu model:

- *High Recall, Low Precision* ialah model memberi hasil prediksi yang kurang baik pada sampel positif, tetapi model dapat meminimalkan prediksi *false positive*.
- *Low Recall, High Precision* ialah model memberi hasil prediksi yang baik pada sampel positif, tetapi akan banyak prediksi *false positive*.

Setelah itu akan dilakukan perbandingan dengan cara menjalankan sistem pada *environment* yang berbeda, sebagai contoh dilakukan perbandingan apabila sistem dijalankan pada *environment* yang terpasang *Graphical Processing Unit* (GPU) dengan yang tidak terpasang GPU.

### 3.2 Blok Diagram Sistem

Dalam pembuatan sistem, terlebih dahulu dilakukan pembuatan blok diagram sistem sebagai gambara atau acuan dalam pembuatan sistem agar dapat tercapai target yang dikehendaki. Blok diagram yang telah dibuat ialah sebagai berikut:




Gambar 3.2 Blok Diagram

Pada blok diagram yang telah dibuat yang ditunjukkan oleh gambar 3.1 terdapat tiga blok pada sistem, yaitu masukan, proses, dan keluaran. Masukan dari sistem ialah *file* video yaitu berupa rekaman video jalan raya yang dilewati oleh mobil-mobil. Setelah itu, dari masukan akan masuk ke blok proses yang di dalamnya terdapat phyton sebagai bahasa pemrograman pada proses. OpenCV digunakan sebagai *library* pembantu dalam melakukan pengolahan citra. Dan YOLO sebagai model yang digunakan untuk mendeteksi objek pada citra. Selanjutnya, hasil dari blok proses ialah berupa keluaran dalam bentuk *bounding boxes*, jumlah mobil, dan *confusion matrix*. *Bounding boxes* merupakan objek kotak yang mengelilingi objek yang terdeteksi. Sedangkan, *confusion matrix* ialah sebuah metode yang digunakan untuk mengevaluasi performa dari sistem yang dibuat. Di dalam *confusion matrix* terdapat beberapa variabel yang akan dijadikan tolak ukur performa dari sistem, yaitu beberapa diantaranya ialah akurasi, *recall*, *precision*, dan *F1-score*. Penjelasan dari masing-masing bagian dari blok diagram ialah sebagai berikut:

a. Video

   Video masukan ialah berupa *file* video yang direkam oleh penulis dengan resolusi 720p dan *frame rate* 30fps. *File* video yang direkam ialah dalam format mp4.

b. Phyton

   Phyton ialah salah satu bahasa pemrograman yang multifungsional, phyton dapat digunakan untuk membuat program untuk *website*, aplikasi, *machine learning*, dan lain sebagainya. Pada sistem yang dibuat oleh penulis, phyton dipilih sebagai bahasa pemrograman.

c. OpenCV

OpenCV ialah sebuah *open source library* yang penggunaannya dapat membantu untuk melakukan pemrosesan citra atau *image processing*. Dengan OpenCV, fitur-fitur yang terkandung di dalam citra dapat didapatkan dan dapat dimodifikasi untuk selanjutnya dilakukan proses lebih lanjut.

d. YOLO

YOLO ialah suatu algoritma yang digunakan untuk mendeteksi objek yang terdapat pada suatu citra atau biasa disebut *object detection*. Pada sistem yang dibuat oleh penulis, akan digunakan YOLOv4-tiny dan beberapa versi YOLO lain sebagai pembanding secara performa.

e. *Bounding Boxes*

*Bounding Boxes* ialah merupakan kotak yang tertampil pada citra untuk menunjukkan posisi atau letak objek yang telah dideteksi. *Bounding boxes* yang tertampil pada sistem yang dibuat oleh penulis akan menampilkan informasi nama *class* atau objek yang dideteksi berikut pula dengan nilai *confidence*-nya.

f. *Confusion Matrix*

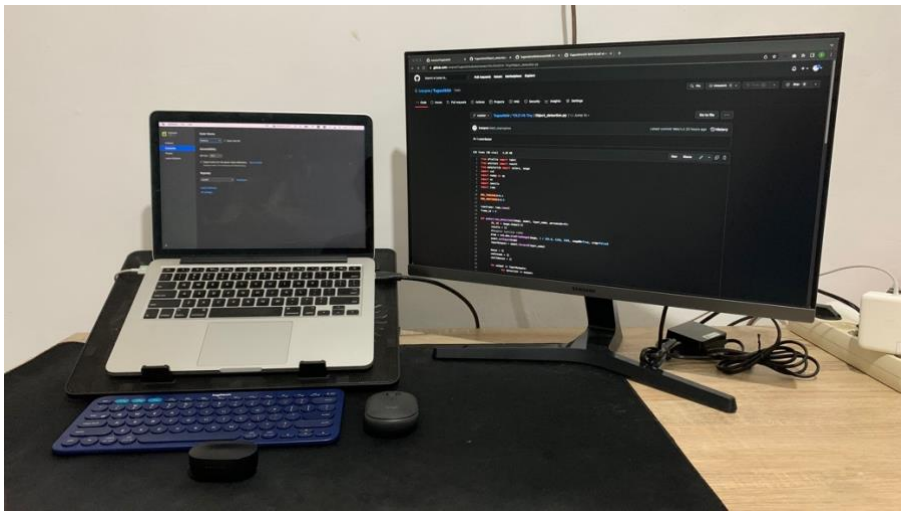
*Confusion matrix* merupakan suatu metode untuk mengevaluasi sistem yang dibuat oleh penulis. Beberapa variabel yang akan dihitung pada sistem yang dibuat ialah akurasi, *recall, precision*, dan *F1-score*.

### 3.3 Perancangan Perangkat Keras

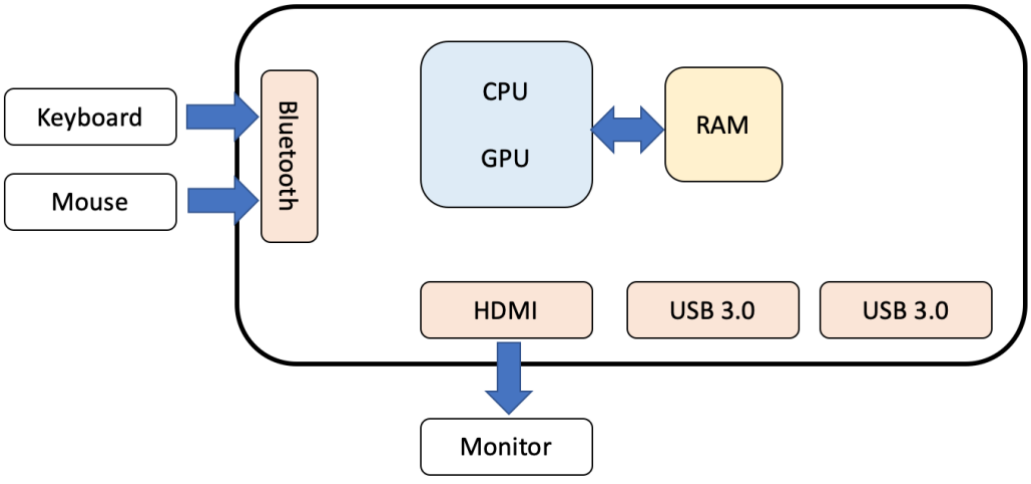
Perangkat keras yang digunakan dalam sistem yang penulis buat ialah monitor dan laptop dengan spesifikasi sebagai berikut:

Tabel 3.1 Spesifikasi Perangkat Keras

| | |
|---|---|
| Type Laptop | MacBook Pro (Retina, 13-inch, Early 2015) |
| *Processor* | 2,7 GHz Dual-Core Intel Core i5 |
| *Memory* | 16 GB 1867 MHz DDR3 |
| *Graphics* | Intel Iris Graphics 6100 1536 MB |
| Sistem Operasi | macOS Monterey 12.3.1 |



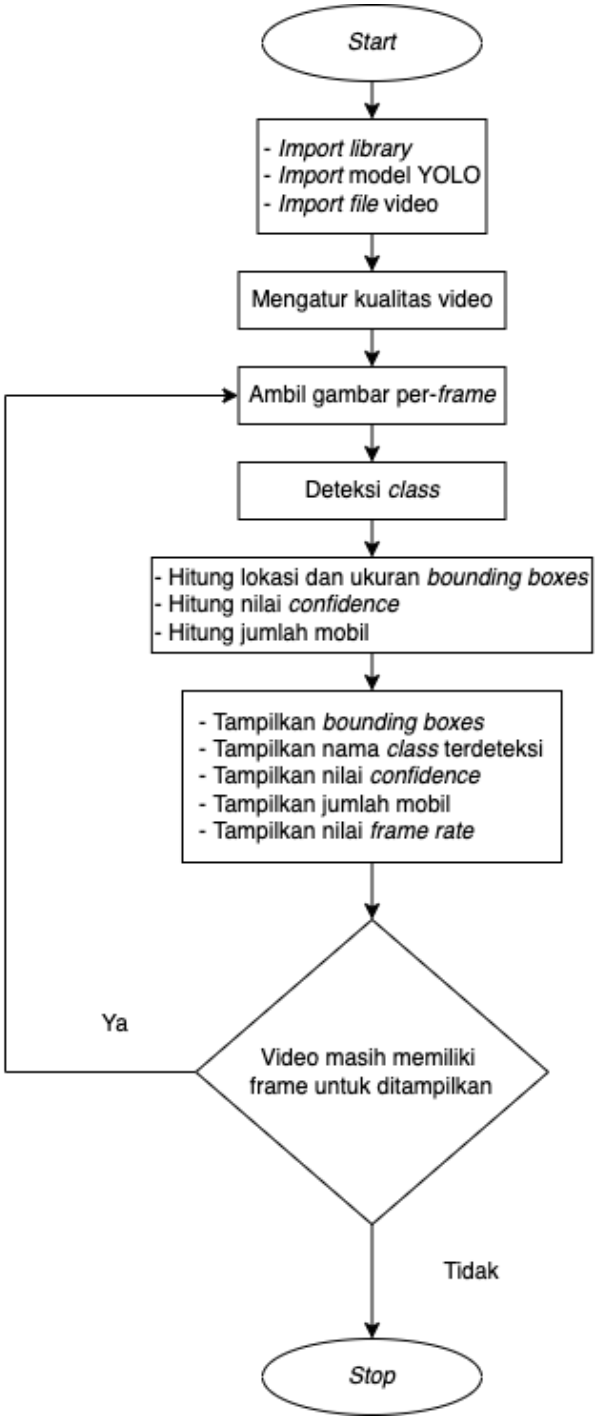
Gambar 3.3 Tampilan perangkat keras




Gambar 3.4 Arsitektur Perangkat Keras

### 3.4 Perancangan Program Perangkat Lunak

Dalam mengembangkan sebuah sistem perangkat lunak, diperlukan suatu aucan sebelum memulai membuat kode yang biasanya dapat berupa suatu algoritma atau diagram alur. Berikut ini ialah diagram alur dari sistem yang akan dibuat oleh penulis:




Gambar 3.5 Diagram alur program perangkat lunak

Dari diagram alur di atas, dapat dijabarkan alur program yang dibuat sebagai berikut:

a. *Start*,

   Saat program dimulai, lakukan *import library* yang digunakan, model YOLO, dan *file* video yang akan digunakan.

b. Mengatur kualitas video,

   Video diatur kualitasnya agar objek yang terdapat di video dapat dideteksi oleh YOLO.

c. Mengambil *frame* citra,

   Mengambil satu *frame* dari video untuk kemudian dideteksi citranya.

d. Deteksi objek,

   Dari *frame* yang diambil dilakukan pengecekan *class* atau objek pada citra.

e. Menghitung lokasi *bounding boxes*, *confidence*, jumlah mobil,

   Apabila objek atau *class* telah terdeteksi, dilakukan perhitungan lokasi dan ukuran dari *bounding boxes* yang akan ditampilkan untuk menandai objek. Selanjutnya, hitung nilai *confidence* dari hasil deteksi objek. Mobil yang dideteksi, akan dijadikan masukkan ke perhitungan jumlah mobil yang lewat atau terdeteksi.

f. Tampilkan keluaran,

   Langkah selanjutnya, tampilkan *bounding boxes*, nama *class* atau objek yang dideteksi, nilai *confidence*, jumlah mobil terdeteksi, dan nilai *frame rate* dalam satuan fps.

g. Cek *frame*,

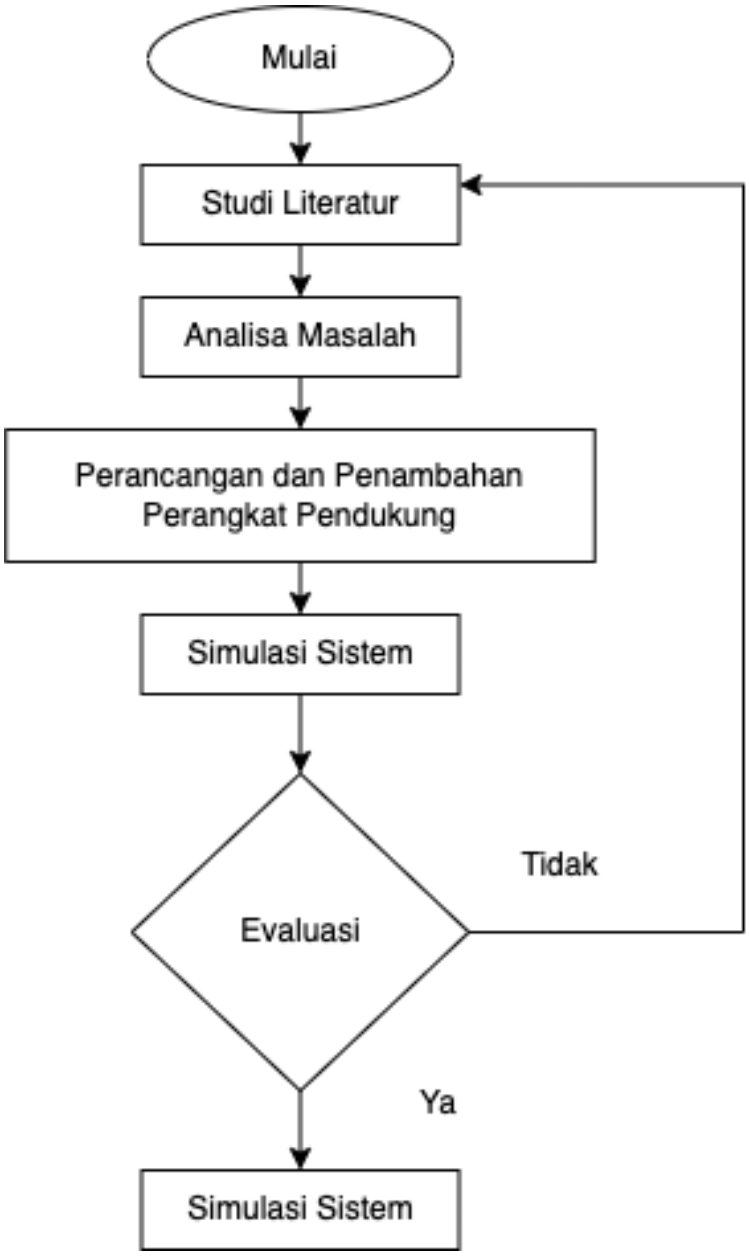
   Selanjutya dilakukan pengecekan bahwa video masih mempunyai *frame* yang dapat dimasukkan sebagai citra. Apabila masih, maka program akan berjalan kembali dari langkah c. Apabila *frame* dari video telah habis, maka program akan keluar atau berhenti.

h. Stop,

   Ketika tidak ada *frame* sisa dari video, maka sistem akan diberhentikan.

### 3.5 Skema Penelitian

Dalam proses penelitian, terdapat beberapa langkah yang ditempuh dari awal penentuan tema hingga tercapai sebuah kesimpulan. Berikut ini ialah diagram alur dari proses penelitian:




Gambar 3.6 Skema penelitian

Pada gambar 3.6 di atas, ialah digambarkan diagram alur dari penelitian yang penulis lakukan. Penjelasan dari masing-masing langkah ialah sebagai berikut:

a. Mulai

   Pada tahap ini penulis mengumpulkan informasi-informasi masalah yang muncul dari lingkungan riil dan dengan bantuan informasi yang didapatkan dari jurnal dan internet sehingga penulis mendapatkan judul "Implementasi Pengolahan Citra Digital Untuk Menghitung Jumlah Kendaraan Masuk Pada Jalan Raya".

b. Studi Literatur

   Pada tahap ini, dilakukan pencarian dan pengkajian informasi terkait sistem yang akan dibuat dengan sumber bermacam-macam seperti buku, internet, jurnal, dan lain sebagainya. Serta pengumpulan data-data dan spesifikasi sistem atau perangkat yang diperlukan untuk mengimplementasi sistem yang dibuat.

c. Analisa Masalah

   Pada tahapan ini, dilakukan analisa dari teori-teori dan informasi yang didapatkan dari berbagai sumber agar dapat menghasilkan dengan semaksimal mungkin.

d. Perancangan dan Penambahan Perangkat Pendukung

   Pada tahap ini, dilakukan perancangan dari sistem yang kemudian diimplementasi menjadi sebuah sistem.

e. Simulasi Sistem

   Melakukan simulasi sistem untuk melihat hasil dari sistem yang telah dibuat.

f. Evaluasi

Setelah sistem terimplementasi dan dapat dijalankan, diperlukan suatu tahapan untuk mengukur tingkat performa dari sistem yang telah dibuat. Apabila performa dirasa belum memuaskan, kemudian akan dilakukan perbaikan hingga mendapatkan hasil yang maksimal.

g. Kesimpulan

Penarikan kesimpulan harus relevan dengan dari hasil, evaluasi, dan tujuan dari penelitian