RANCANG BANGUN APLIKASI PENGHITUNGAN OKUPANSI MANUSIA DENGAN CCTV MENGGUNAKAN *OBJECT DETECTION MODEL* YOLOV8

MUHAMMAD AZHAR TAWAKKAL

H071201041

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

RANCANG BANGUN APLIKASI PENGHITUNGAN OKUPANSI MANUSIA DENGAN CCTV MENGGUNAKAN *OBJECT DETECTION MODEL* YOLOV8

MUHAMMAD AZHAR TAWAKKAL

H071201041

PROGRAM STUDI SISTEM INFORMASI DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS HASANUDDIN

MAKASSAR

2024

RANCANG BANGUN APLIKASI PENGHITUNGAN OKUPANSI MANUSIA DENGAN CCTV MENGGUNAKAN *OBJECT DETECTION MODEL* YOLOV8

MUHAMMAD AZHAR TAWAKKAL

H071201041

Skripsi

sebagai salah satu syarat untuk mencapai gelar sarjana

Program Studi Sistem Informasi

pada

PROGRAM STUDI SISTEM INFORMASI

DEPARTEMEN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERESITAS HASANUDDIN

MAKASSAR

2024

SKRIPSI

RANCANG BANGUN APLIKASI PENGHITUNGAN OKUPANSI MANUSIA DENGAN CCTV MENGGUNAKAN *OBJECT DETECTION MODEL* YOLOV8

MUHAMMAD AZHAR TAWAKKAL
H071201041

Skripsi,

telah dipertahankan di depan Panitia Ujian Sarjana Sistem Informasi pada
20 September 2024
dan dinyatakan telah memenuhi syarat kelulusan
pada

Program Studi Sistem Informasi
Departemen Matematika
Fakultas Matematika dan Ilmu Pengetahuan Alam
Universitas Hasanuddin
Makassar

Mengesahkan:	Mengetahui:
Pembimbing Tugas Akhir,	Ketua Program Studi,
A. Muh. Amil Siddik, S.Si., M.Si	Prof. Dr. Jeffry Kusuma, Ph. D
NIP. 199110032019031015	NIP. 196411121987031002

PERNYATAAN KEASLIAN SKRIPSI DAN PELIMPAHAN HAK CIPTA

Dengan ini saya menyatakan bahwa, skripsi berjudul "Rancang Bangun Aplikasi Penghitungan Okupansi Manusia Dengan CCTV Menggunakan *Object Detection* Model YOLOv8" adalah benar karya saya dengan arahan dari pembimbing A. Muh. Amil Siddik, S.Si., M.Si sebagai Pembimbing Utama. Karya ilmiah ini belum diajukan dan tidak sedang diajukan dalam bentuk apa pun kepada perguruan tinggi mana pun. Sumber informasi yang berasal atau dikutip dari karya yang diterbitkan maupun tidak diterbitkan dari penulis lain telah disebutkan dalam teks dan dicantumkan dalam Daftar Pustaka skripsi ini. Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini adalah karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut berdasarkan aturan yang berlaku.

Dengan ini saya melimpahkan hak cipta (hak ekonomis) dari karya tulis saya berupa skripsi ini kepada Universitas Hasanuddin.

Makassar, 4-September-2024

MUHAMMAD AZHAR TAWAKKAL
H071201041

ABSTRAK

MUHAMMAD AZHAR TAWAKKAL. **Rancang Bangun Aplikasi Penghitungan Okupansi Manusia Dengan CCTV Menggunakan *Object Detection Model* YOLOv8** (dibimbing oleh A. Muh. Amil Siddik).

Latar belakang. Metode tradisional penghitungan okupansi manusia dalam analisis okupansi dianggap kurang akurat, sehingga berkembanglah berbagai metode modern, seperti penggunaan sensor kamera berbasis *object detection*. Beberapa penelitian sebelumnya telah mengembangkan sistem deteksi okupansi manusia menggunakan model-model seperti Faster R-CNN, Mask R-CNN, dan YOLOv3, namun masih memiliki kekurangan dalam akurasi dan implementasi. **Tujuan**. Penelitian ini bertujuan mengembangkan aplikasi web penghitungan okupansi manusia menggunakan model *object detection* YOLOv8, mengetahui cara menerapkan aplikasi, dan mengetahui bagaimana performa aplikasi dalam melakukan penghitungan. **Metode.** Aplikasi dikembangkan menggunakan metode *waterfall* dengan model YOLOv8 sebagai modelnya dan DeepSORT untuk *tracking*. **Hasil**. Aplikasi penghitungan okupansi manusia dikembangkan menggunakan *framework* Flask dengan *database* menggunakan MySQL. Aplikasi melakukan deteksi secara *real-time* menggunakan model YOLOv8 dan objeknya di-*track* menggunakan DeepSORT. Akurasi mencapai 100% ketika *traffic* hanya 1-3 orang namun menurun hingga 75% ketika padat. **Kesimpulan**. Aplikasi berhasil menjalankan penghitungan okupansi secara *real-time* dengan bantuan model YOLOv8 dan aplikasi bisa digunakan untuk membantu *stakeholder* dalam mengatur okupansi manusia di suatu area.

Kata Kunci: Penghitungan Okupansi, YOLOv8, DeepSORT, *Real-time*, CCTV

ABSTRACT

MUHAMMAD AZHAR TAWAKKAL. **Design of Human Occupancy Calculation Application with CCTV Using Object Detection Model YOLOv8** (supervised by A. Muh. Amil Siddik).

Background. The traditional method of calculating human occupancy in occupancy analysis is considered less accurate, so various modern methods, such as the use of camera sensors based on object detection, are developed. Several previous studies have developed human occupancy detection systems using models such as Faster R-CNN, Mask R-CNN, and YOLOv3, but they still have shortcomings in accuracy and implementation. **Aim**. This research aims to develop a web application for human occupancy calculation using the YOLOv8 object detection model, find out how to implement the application and determine how the application performs the calculation. **Methods.** The application was developed using the waterfall method with the YOLOv8 model as the model and DeepSORT for tracking. **Results**. The human occupancy calculation application was developed using the Flask framework with MySQL Database. The application performs real-time detection using the YOLOv8 model and the objects are tracked using DeepSORT. Accuracy reaches 100% when traffic is only 1-3 people but decreases to 75% when congested. **Conclusion**. The application successfully runs real-time occupancy calculations with the help of the YOLOv8 model and the application can be used to assist stakeholders in managing human occupancy in an area.

Keywords: Occupancy Counting, People Occupancy, YOLOv8, DeepSORT, Real-time, CCTV

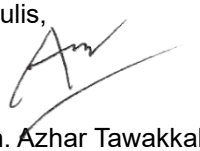
UCAPAN TERIMA KASIH

Penelitian yang saya lakukan dapat terlaksana dengan sukses dan skripsi ini dapat terampungkan atas bimbingan, diskusi dan arahan Bapak A. Muh. Amil Siddik, S.Si.,M.Si sebagai dosen pembimbing, Bapak Dr. Eng. Armin Lawi, S.Si., M.Eng. sebagai dosen penguji 1, dan Bapak Jeriko Gormantara, S.Si., M.Si sebagai dosen penguji 2. Saya mengucapkan berlimpah terima kasih kepada mereka. Penghargaan yang tinggi juga saya sampaikan kepada PT STECHOQ Robotika Indonesia dan Pihak MSIB yang telah memberikan penulis kesempatan untuk melakukan magang sehingga penelitian ini dapat dilakukan

Ucapan terima kasih juga saya ucapkan kepada pimpinan Universitas Hasanuddin, pimpinan Fakultas Matematika dan Ilmu Pengetahuan Alam, pimpinan Departemen Matematika, pimpinan Prodi Sistem Informasi serta Dosen dan Staff Universitas Hasanuddin yang telah memfasilitasi saya selama menempuh pendidikan saya.

Akhirnya, kepada kedua orang tua tercinta saya mengucapkan limpah terima kasih dan sembah sujud atas doa, pengorbanan dan motivasi mereka selama saya menempuh pendidikan. Penghargaan yang besar juga saya sampaikan kepada teman-teman saya di Galaxy Dev, PT STECHOQ Robotika Indonesia, Bangkit Academy, dan KKNT ITTG Bantaeng Desa Bonto Rannu yang telah memberi motivasi bagi penulis selama menempuh pendidikan.

Penulis,



Muh. Azhar Tawakkal

DAFTAR ISI

Halaman

DAFTAR TABEL

Nomor Urut Halaman

DAFTAR GAMBAR

Nomor Urut Halaman

DAFTAR RUMUS

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Analisis okupansi merupakan hal yang fundamental di berbagai aspek, mulai dari arsitektur, perencanaan kota, dan masih banyak lagi. Analisis okupansi ini sangat penting apabila suatu daerah ingin menjadi sebuah *smart city*. Analisis okupansi ini dapat membantu untuk menyelesaikan berbagai masalah, seperti penggunaan fasilitas umum yang lebih efisien, peningkatan kualitas pelayanan bagi masyarakat, bahkan hingga pengurangan biaya operasional (Kim, Choi, Moon, Moon, & Sung, 2023). Hal serupa juga dapat diterapkan dalam bangunan komersial dimana analisis okupansi manusia dapat dilakukan untuk meningkatkan kenyamanan karyawan dan efisiensi penggunaan energi dalam kantor.

Analisis okupansi manusia tentunya memerlukan data okupansi terlebih dahulu. Penghitungan okupansi manusia dapat dilakukan dengan berbagai cara. Cara tradisional untuk melakukan ini adalah dengan menghitung secara manual jumlah okupansi dalam suatu ruangan. Namun, karena cara ini rawan akan kesalahan, maka muncullah berbagai metode baru untuk membantu menghitung okupansi. Ada banyak metode baru yang dapat digunakan untuk melakukan okupansi, seperti menggunakan kadar CO_2 dalam ruangan sebagai parameter, menggunakan sensor frekuensi radio, dan menggunakan sensor kamera (Wang, Hong, Piette, & Pritoni, 2019).

Penghitungan okupansi manusia dengan sensor kamera mengandalkan *object detection* untuk mengenali objek manusia yang melalui sensor kamera. *Object detection* merupakan bagian dari *computer vision* yang merupakan proses menemukan suatu objek pada gambar atau video dan mencocokkannya dengan label yang telah ditentukan sebelumnya. *Object detection* ini didorong oleh banyaknya keberhasilan penerapan *Deep Neural Network* (DNN) dan *Convolutional Neural Network* (CNN) belakangan ini. Hingga saat ini, penerapan *object detection* di dunia sudah sangat banyak, contohnya seperti mobil otomatis, pengenalan wajah, hingga deteksi kanker (Amjoud & Amrouch, 2023). Melihat semua contoh tersebut, deteksi manusia melalui kamera tentunya bisa dilakukan. Hal inilah yang memungkinkan adanya penghitungan okupansi menggunakan sensor kamera.

Jenis kamera yang cocok melakukan penghitungan okupansi ini adalah kamera CCTV. Hal ini disebabkan karena kamera CCTV sudah sering ditemukan di tempat-tempat umum sehingga tidak memerlukan pemasangan perangkat baru untuk melakukan penghitungan okupansi. Selain itu, posisi kamera CCTV biasanya diletakkan di posisi yang tinggi dimana objek manusia serta pergerakannya dapat terlihat dengan baik. Hal ini membuat penghitungan okupansi menjadi lebih mudah.

Dalam hal deteksi okupansi manusia, ada beberapa penelitian yang telah dilakukan. Pada tahun 2020, Duta Sayoga dkk. Mengembangkan sebuah sistem deteksi okupansi menggunakan *computer vision* untuk *smart building* dan *automation*. Sistem yang dibuat menggunakan Faster R-CNN dengan akurasi rata-

rata mencapai 83.45%. Akan tetapi, dalam penelitian tersebut, peneliti hanya menguji sistem dengan gambar dan tidak menguji menggunakan video. Selain itu, penelitian tersebut juga tidak memberikan bagaimana sistem tersebut akan di-*deploy* untuk digunakan oleh banyak pengguna (Sayoga, Kusuma, & Hasibuan, 2020).

Ada juga penelitian yang dilakukan oleh Deni Tri Laksono dkk. dalam mengembangkan deteksi okupansi manusia. Sistem yang mereka kembangkan menggunakan model CNN dengan akurasi sistem dalam menghitung jumlah manusia sebesar 62% dengan berbagai sudut kamera dan intensitas cahaya. Sedangkan untuk akurasi menghitung orang masuk sebesar 73%, orang keluar sebesar 64%, dan orang 62% orang yang berada di dalam ruangan. Melihat dari akurasi yang dicapai, sistem dapat dikembangkan lebih jauh lagi agar mendapat akurasi yang lebih optimal (Lakson, et al., 2022).

Selain itu, ada juga penelitian yang dilakukan oleh Chee Jia Hong dkk. Mengenai penghitungan manusia menggunakan *object detection* dan *object tracking*. Sistem yang dikembangkan menggunakan model Mask R-CNN. Sistem yang dikembangkan mampu mendeteksi manusia dengan baik dan menghitung jumlah manusia yang ada di *frame*. Namun, sistem tidak mencatat berapa orang yang telah dideteksi oleh sistem selama interval waktu tertentu (Chee & Mazlan, 2023).

Untuk aplikasi yang menggunakan model YOLO, ada juga penelitian yang dilakukan oleh Hamam Mokayed dkk. Penelitian yang dilakukan menggunakan model YOLOv3 untuk model *object detection* sistem. Kemudian, selain mengembangkan sistem *detection* dan *tracking*, mereka juga mengembangkan fitur *threshold* yang dapat digunakan untuk menentukan berapa banyak yang telah masuk, keluar, atau yang berada di dalam suatu area. *Threshold* tersebut yang membantu sistem untuk melakukan penghitungan okupansi. Setelah sistem selesai, peneliti melakukan *deploy* dengan membuat web untuk sistem agar sistem bisa digunakan dengan mudah oleh pengguna (Mokayed, Quan, Alkhaled, & Sivakumar, 2022).

Melihat penelitian sebelumnya, penulis melihat ada beberapa hal yang bisa ditingkatkan. Model yang digunakan dalam sistem penghitungan okupansi dapat diubah dengan model terbaru, yang salah satunya adalah model YOLOv8. Beberapa penelitian tersebut juga tidak melakukan *deploy* dari sistem yang telah dibuat, sehingga penulis akan mengembangkan sebuah web yang akan memanfaatkan sistem penghitungan okupansi menggunakan *object detection* dan *object tracking* dengan tujuan agar sistem ini dapat digunakan meskipun penggunanya tidak memiliki latar belakang IT. Selain itu, penulis juga akan mengembangkan fitur *threshold* yang dapat diatur oleh pengguna sehingga sistem dapat menyesuaikan sesuai dengan keperluan pengguna tanpa harus melakukan perubahan di kodenya. Penulis juga akan mengembangkan fitur *dashboard* untuk melihat perkembangan okupansi dan mengunduh data hasil penghitungan okupansi.

Dengan adanya penelitian ini, diharapkan performa yang lebih baik pada sistem deteksi okupansi manusia dibandingkan penelitian sebelumnya. Deteksi yang lebih baik akan membantu dalam penghitungan okupansi agar menjadi lebih akurat.

Selain itu, penelitian ini juga diharapkan dapat memberikan gambaran bagaimana model *object detection* dapat diterapkan dalam aplikasi web, khususnya dalam menghitung okupansi. Penerapan model dalam bentuk web membuat siapa saja dengan latar belakang apapun dapat menggunakan model tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas, maka rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana cara merancang dan membangun aplikasi penghitungan okupansi yang menerapkan YOLOv8 dalam sistemnya?
2. Bagaimana cara menerapkan sistem pada suatu area yang perlu dihitung okupansinya?
3. Bagaimana performa aplikasi dalam menghitung okupansi di suatu area?

1.3 Batasan Masalah

Agar pembahasan tidak terlalu luas, maka diperlukan batasan-batasan masalah. Batasan masalahnya antara lain :

1. Model yang digunakan adalah model YOLOv8x yang dilatih menggunakan COCO dataset.
2. Pengujian akan dilakukan menggunakan kamera CCTV pada kantor PT Stechoq Robotika Indonesia
3. Pengukuran performa sistem dilakukan menggunakan penghitungan manual untuk mengetahui seberapa baik model mendeteksi serta menghitung manusia yang ada pada suatu area

1.4 Tujuan Penelitian

Adapun tujuan penelitian kali ini sebagai berikut :

1. Mengetahui cara merancang dan membangun aplikasi penghitungan okupansi yang menerapkan YOLOv8 dalam sistemnya.
2. Mengetahui cara menerapkan sistem pada suatu area yang perlu dihitung okupansinya.
3. Mengetahui bagaimana performa aplikasi dalam menghitung okupansi di suatu area.

1.5 Manfaat Penelitian

Penelitian ini diharapkan dapat memberikan suatu cara baru untuk *stakeholder* mengetahui data tingkat okupansi suatu area dan membantu mereka dalam mengambil keputusan.

1.6 Teori

1.6.1 Studi Kasus

Penelitian ini dilakukan di PT STECHOQ Robotika Indonesia yang terletak di Kabupaten Sleman, D.I. Yogyakarta. PT STECHOQ Robotika Indonesia menginginkan sebuah sistem yang mampu menghitung jumlah karyawan yang keluar masuk dalam ruangan kantor menggunakan kamera CCTV yang telah terpasang disana serta menghitung berapa orang yang ada di dalam ruangan. *Stakeholder* meminta untuk menggunakan *object detection* dalam penghitungannya serta menambahkan beberapa fitur seperti kustomisasi garis hitung, *dashboard*, hingga fitur untuk mengunduh data hasil penghitungan. Harapannya, sistem ini dapat menjadi dasar bagi sistem *object detection* disana dimana mereka menargetkan kedepannya sistem ini dapat menjadi sistem absensi otomatis dengan mendeteksi wajah karyawan yang datang dengan CCTV.

1.6.2 Okupansi Manusia

Okupansi merujuk kepada kondisi dimana sesuatu sedang ditempati atau digunakan. Okupansi juga dapat diartikan sebagai keadaan yang menjelaskan adanya penghuni pada suatu tempat (Sayed, Himeur, & Bensaali, 2022). Dengan kata lain, okupansi manusia merupakan keadaan yang menjelaskan adanya manusia pada suatu tempat. Data penghitungan okupansi manusia dapat dianalisis untuk menyelesaikan berbagai permasalahan. Beberapa contoh diantaranya seperti penggunaan fasilitas umum yang lebih efisien, peningkatan kualitas pelayanan bagi masyarakat, bahkan hingga pengurangan biaya operasional (Kim, Choi, Moon, Moon, & Sung, 2023). Dalam masalah bangunan komersial, data okupansi juga dapat berguna seperti untuk efisiensi listrik atau pemenuhan kebutuhan karyawan.

1.6.3 Sistem *Real-Time*

Sistem *real-time* merupakan sebuah sistem yang berjalan dengan memberikan suatu respon pasti dalam batasan waktu yang ditentukan (Dahiya, 2023). Dengan kata lain, sistem ini akan merespon suatu kejadian dalam waktu yang dapat diperkirakan. Dalam sistem *real-time*, apabila sistem tidak dapat merespon tepat waktu sesuai yang diperkirakan, maka akan ada konsekuensi serius yang terjadi. Sehingga, aspek waktu sangat penting dalam sistem ini.

Dalam penelitian ini, penghitungan okupansi berjalan secara *real-time*. Setiap *frame* yang ditangkap oleh kamera CCTV, akan dikirimkan ke aplikasi. Setelah itu, setiap *frame* tersebut akan diproses untuk menghitung jumla orang yang ada di *frame* tersebut. Setelah selesai, *frame* tersebut akan ditampilkan di halaman web agar bisa dipantau oleh pengguna. Apabila sistem memproses terlalu lama, maka

aplikasi web tidak akan mampu menampilkan CCTV secara *real-time* dan pengawasannya menjadi terganggu.

1.6.4 Object Detection

Object Detection merupakan salah satu dari tugas *computer vision* yang berkaitan dengan mengidentifikasi suatu objek pada gambar. Dalam tugas *object detection*, gambar akan diproses oleh komputer dan nantinya akan menghasilkan output berupa *bounding box* beserta label untuk *bounding box* tersebut (Brownlee, 2021). *Object detection* sendiri bekerja dengan menemukan lokasi objek pada suatu gambar lalu mencocokkan objek yang ditemukan dengan label *class* yang telah ditetapkan sebelumnya (Amjoud & Amrouch, 2023).

Untuk melakukan *object detection*, ada banyak model yang telah dibuat yang dapat digunakan. Menurut Ravpreet dan Sarbjeet (Kaur & Singh, 2023), Model untuk *object detection* terbagi menjadi dua, yaitu *two stage object detector* dan *one stage object detector*.

1. *Two stage object detector* adalah model yang memisahkan antara proses lokalisasi objek dan klasifikasi objek pada gambar. Model terlebih dahulu menentukan *region* untuk objek yang ditemukan lalu *region* tersebut diklasifikasikan berdasarkan *class* yang ada. Beberapa contoh model dari kategori ini adalah Faster RCNN dan Mask RCNN
2. *One stage object detector* adalah model yang melakukan proses lokalisasi dan klasifikasi objek secara bersamaan menggunakan *Diffusion-Convolutional Neural Network* (DCNN) tanpa perlu memisahnya menjadi tugas terpisah. Beberapa contoh model dari kategori ini adalah SSD dan keluarga model YOLO.

1.6.5 Perbandingan Berbagai Model *Object Detection*

Dalam memilih model *object detection*, ada beberapa hal yang perlu diperhatikan khususnya dalam melakukan deteksi secara *real-time*. Pertama, kecepatan model dalam melakukan deteksi objek. Dalam melakukan deteksi *real-time*, khususnya aplikasi yang akan dipantau terus-menerus, diperlukan model yang mampu memproses setiap frame dengan cepat sehingga tampilan kamera di aplikasi web dapat berjalan dengan mulus dengan *delay* seminimal mungkin. Kedua, akurasi model dalam melakukan deteksi objek. Aplikasi juga memerlukan model deteksi dengan akurasi yang tinggi, dengan harapan aplikasi dapat bekerja dengan berbagai resolusi kamera yang ada.

Beberapa penelitian telah dilakukan untuk membandingkan beberapa model *object detection* yang ada. Pada tahun 2021, Lu Tan dkk. membandingkan model YOLOv3, Faster R-CNN, dan SSD untuk melakukan deteksi pil obat secara *real-time*. Adapun hasil yang ditemukan adalah sebagai berikut :

Tabel 1 Perbandingan Model *Object Detection* oleh Lu Tan Dkk. (Tan, Huangfu, Wu, & Chen, 2021)

Algoritma	mAP%	FPS
YOLOv3	79.02	69
RetinaNet	79.61	22
SSD	79.03	41

Selain itu, penelitian serupa juga dilakukan Dalmar Dakari dan Cleo Daniel pada tahun 2023. Dalam penelitian mereka, mereka membandingkan beberapa model yaitu YOLOv4, SSD, dan Faster R-CNN. Dalam penelitiannya, mereka menggunakan dataset test dari COCO sebagai pengujiannya dan mendapat hasil sebagai berikut :

Tabel 2 Perbandingan Model *Object Detection* oleh Dalmar Dakari dan Cleo Daniel (Aboyomi & Daniel, 2023)

Algoritma	mAP%	FPS
YOLOv4	54.30	40
Faster R-CNN	61.20	8
SSD	56.80	22

Perbandingan lainnya juga dilakukan oleh Debani Prasad Mishra dkk. pada tahun 2023. Di dalam penelitiannya, mereka membandingkan performa dari lima buah model, yaitu YOLO, Fast R-CNN, ResNet, Faster R-CNN, dan SSD. Adapun hasil yang diperoleh dari penelitian tersebut adalah sebagai berikut :

Tabel 3 Perbandingan Model *Object Detection* oleh Debani Prasad dkk. (Mishra, Rout, Mishra, & Salkuti, 2023)

Algoritma	Akurasi	FPS
YOLO	63.4	45
Fast R-CNN	70.0	0.5
ResNet	76.4	5
Faster R-CNN	73.2	7
SSD	76.8	19

Melihat dari berbagai hasil penelitian tersebut, dapat disimpulkan bahwa model yang cocok untuk deteksi objek secara *real-time* adalah model YOLO. Hal ini disebabkan karena model YOLO memiliki FPS yang paling baik diantara semua model yang telah dibandingkan. Selain itu, akurasi yang dimiliki oleh YOLO terbilang cukup baik meskipun bukan yang terbaik diantara model yang lain.

1.6.6 Dataset COCO

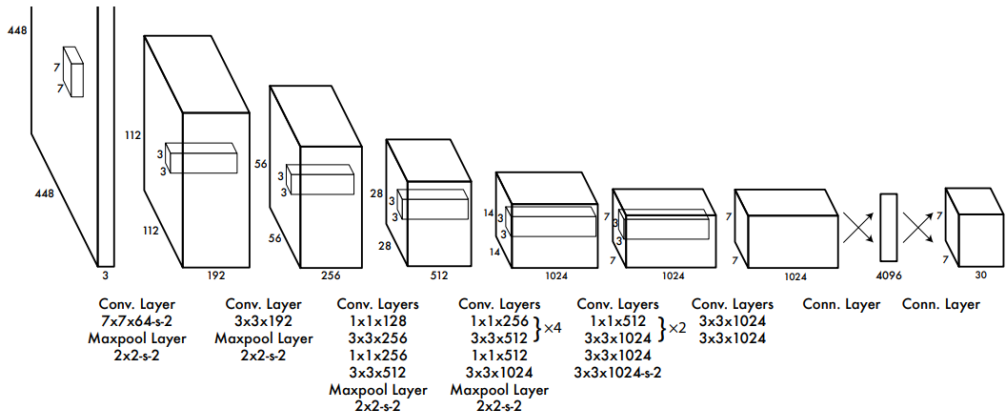
Dataset COCO merupakan dataset skala besar yang digunakan untuk melakukan *object detection*, *segmentation*, *key-point detection* dan *captioning*. Dataset ini disusun oleh Tsung-Yi Lin dengan tim yang beranggotakan 10 orang dan datasetnya telah diperbaharui terus menerus hingga 2020. Hingga saat ini, dataset COCO terbaru, COCO 2017 terdiri atas 118 ribu data *train*, 41 ribu data *test*, dan 5 ribu data validasi. Isi datasetnya dibagi menjadi 91 kategori dimana setiap kategorinya merupakan objek yang mudah dikenali oleh anak berumur 4 tahun. Setiap data gambar pada dataset ini diambil dari kejadian sehari-hari pada konteks yang alami (Lin, et al., 2014).

1.6.7 Model YOLO

Model YOLO (*You Only Look Once*) merupakan model *object detection* yang dirancang oleh Joseph Redmon dkk. pada tahun 2016. YOLO disebut *only look once,* karena YOLO mampu menggambarkan *object detection* hanya dalam satu masalah regresi saja. Model YOLO hanya melihat gambar sekali saja, kemudian menghasilkan deteksi beserta *bounding box*. Berbeda dengan model seperti CNN atau Faster R-CNN yang dimana *object detection* terdiri atas 2 tahap, yaitu penentuan *region* lalu klasifikasi atas *region* tersebut.

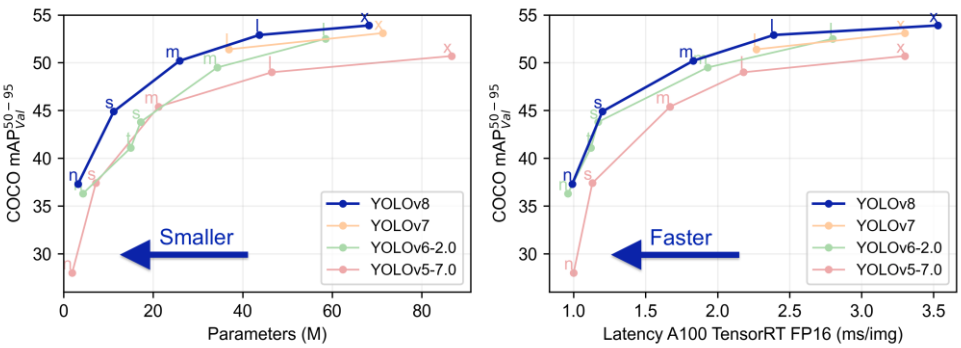
YOLO melakukan *object detection* dengan cara memperkirakan koordinat *bounding box* untuk suatu objek dan menentukan kategori dari objek tersebut. Dalam melakukan *object detection*, YOLO akan membagi gambar menjadi *grid* berukuran S $\times$ S dimana setiap *grid* terdiri atas 5 *tuple*, yaitu x, y, w, h, dan *confidence score*. Objek x dan y menunjukkan koordinat, w dan h merupakan panjang dan lebar *bounding box* yang diprediksi, dan *confidence score* merupakan probabilitas terbesar suatu *bounding box* (Kaur & Singh, 2023).

YOLO sendiri merupakan model CNN. C*onvolutional layer* awal pada jaringan akan mengekstrak fitur dari gambar sedangkan *fully connected layer* akan melakukan prediksi probabilitas output dan koordinat dari *bounding box*. YOLO sendiri terdiri atas 24 *convolutional layers* diikuti dengan 2 *fully connected layers*. YOLO menggunakan 1×1 *reduction layers* diikuti dengan 3×3 *convolutional layers*. Susunan jaringannya adalah seperti berikut (Redmon, Divvala, Girshick, & Farhadi, 2016).



Gambar 1 Struktur Jaringan Model YOLO (Wu, 2018)

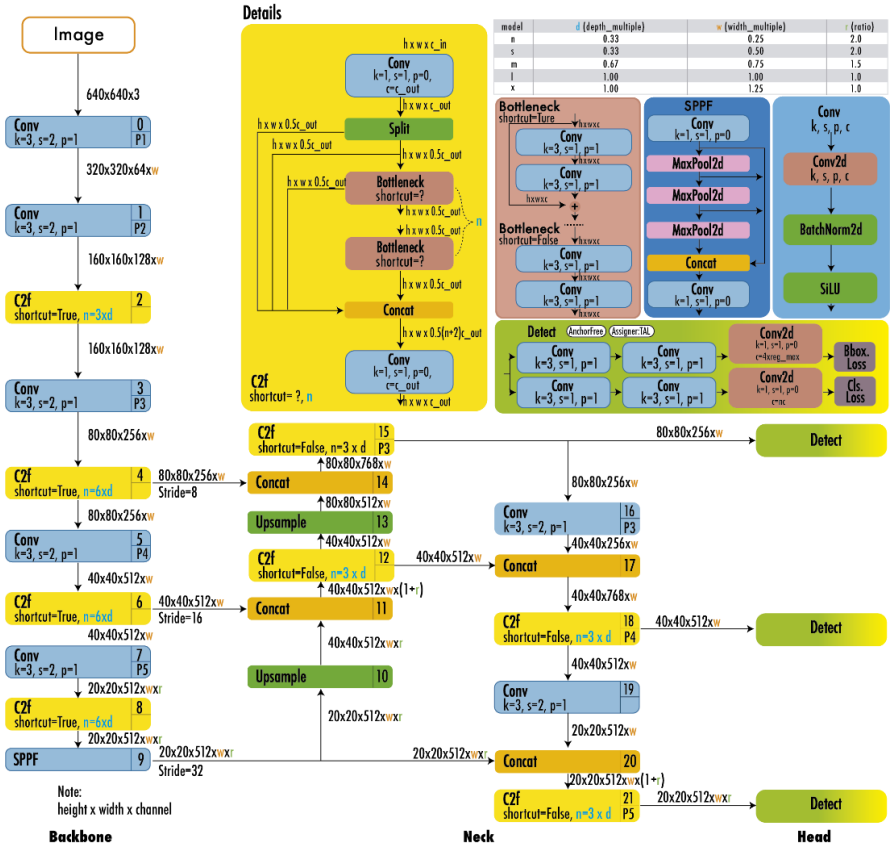
Dikutip dari situs Ultralytics (Jocher & Waxmann, YOLOv8, 2023), hingga saat ini YOLO telah mencapai hingga versi YOLOv8. YOLOv8 tidak dikembangkan oleh penemu YOLO, melainkan oleh tim yang bernama Ultralytics. YOLOv8 sendiri memiliki performa paling baik diantara seluruh versi YOLO sebelumnya. Berdasarkan data dari Ultralytics, jika dibandingkan model-model tahun sebelumnya, YOLOv8 merupakan model terbaik untuk model YOLO. Hal ini bisa dilihat pada grafik berikut :



Gambar 2 Grafik Perbandingan Beberapa Model YOLO Terbaru (Jocher & Waxmann, YOLOv8, 2023)

Pada grafik tersebut, terlihat bahwa model YOLOv8 memiliki nilai mAP yang paling baik saat dicoba ke *dataset* COCO jika dibandingkan beberapa model sebelumnya. Dengan parameter dan latensi yang sama, model YOLOv8 mampu menghasilkan mAP yang lebih baik daripada model-model sebelumnya. Sehingga, dengan beberapa pertimbangan tersebut, maka model YOLOv8 akan digunakan dalam penelitian kali ini.

Arsitektur YOLOv8 juga sudah mengalami banyak perkembangan. Adapun arsitektur dari YOLOv8 adalah sebagai berikut :



Gambar 3 Struktur Jaringan Model YOLOv8 (Terven & Cordova-Esparza, 2023)

Jika dibandingkan dengan model YOLO yang pertama, ada beberapa peningkatan yang dilakukan di YOLOv8. Pertama, YOLOv8 telah menggunakan *backbone* yang lebih kuat seperti C2f yang dapat meningkatkan *feature extraction* dari input yang diberikan. Selain itu, YOLOv8 juga menerapkan *Feature Pyramid Network* (FPN) yang memungkinkan YOLOv8 mendeteksi objek dengan berbagai ukuran, dari ukuran kecil hingga besar. YOLOv8 juga menerapkan konsep *anchor-free model* dimana dalam mendeteksi *bounding box* objek, model tidak memerlukan lagi *predefined anchor box* dalam prosesnya, sehingga model dapat lebih baik dalam mendeteksi objek yang kecil (Terven & Cordova-Esparza, 2023).

Model YOLOv8 sendiri terdiri atas berbagai ukuran, mulai yang terkecil (model n) hingga yang terbesar (model x). Perbedaan model-model tersebut dapat dilihat dari *performance metrics* masing-masing model berikut (Jocher & Qaddoumi, Object Detection Ultralytics, 2024) :

Tabel 4 Perbandingan Berbagai Ukuran Model YOLOv8

Model	**Ukuran gambar (*pixels*)**	**mAP^{val} 50-95**	**Kecepatan (Dengan CPU dan format ONNX (ms)**	**Kecepatan (Dengan GPU A100 TensorRT (ms)**	***Params* (M)**	***FLOPs* (B)**
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Tabel tersebut menjelaskan mengenai bagaimana performa dari masing-masing model terhadap gambar berukuran 640 pixel pada dataset COCO yang ditunjukkan oleh nilai mAP^{val}. Nilai tersebut menunjukkan bagaimana rata-rata *precision* dan *recall* model terhadap setiap *class* objek yang dilatih pada model dengan besar *threshold* dari 50% hingga 95%. Dengan gambar yang sama, terlihat bahwa nilai mAP^{val} untuk model x merupakan yang paling besar. Dengan resolusi kamera CCTV yang terkadang tidak begitu baik, maka model x sangat cocok untuk bisa mengekstrak informasi sebanyak mungkin dari keterbatasan tersebut.

Dalam menggunakan model YOLOv8, ada beberapa komponen atau konfigurasi diatur untuk menyesuaikan model dengan kebutuhan penelitian :

a. *Conf*, yaitu pengaturan *confidence* untuk mengetahui batas *confidence* yang diterima model. *Confidence* merupakan sebuah nilai diantara 0 hingga 1 yang menunjukkan seberapa yakin model terhadap deteksinya. Nilai yang digunakan adalah nilai *default* atau 0.25.
b. *Stream_buffer*, yaitu untuk mengatur apakah semua *frame* harus dilakukan *buffer* atau hanya perlu mengembalikan *frame* terbaru saja saat melakukan pemrosesan. Untuk ini, *stream_buffer* bernilai *true* karena aplikasi akan berjalan secara *real-time*.
c. *Classes*, yaitu untuk mengatur kelas objek apa saja yang akan dideteksi oleh model. Penelitian ini hanya memerlukan model mendeteksi objek manusia sehingg konfigurasi in bernilai 0 yang mewakili kelas objek *person*/manusia.
d. *Source*, yaitu untuk mengatur sumber data (gambar, video, atau *stream*) yang akan dilakukan *object detection*. Untuk penelitian ini, *source* akan berisi objek *streaming* kamera CCTV yang dibuat menggunakan *library* OpenCV

1.6.8 *Back-End* Aplikasi

Back-end aplikasi merupakan bagian dari aplikasi yang mengurus hal seperti penyimpanan data, logika aplikasi, hingga pemrosesan di sisi server. Secara umum, *back-end* terdiri atas tiga komponen (Codecademy Team, 2024) :

a. Server, yaitu komputer yang menerima dan mengolah *request* dari computer *client.*
b. Aplikasi, yaitu *software* yang dijalankan di server yang mengurus *request*, mengambil data dari *database* dan mengirim respon kembali ke *client.*
c. *Database*, yaitu sistem yang menyimpan dan mengatur data yang memungkinkan data untuk diambil dan diperbaharui jika diperlukan

Konsep ini telah digunakan hampir di seluruh aplikasi web yang ada sekarang. Contohnya dapat dilihat saat berbelanja *online* di salah satu web di internet. Ketika pelanggan sudah menemukan barang yang diinginkan, mereka akan memasukkan barang tersebut ke keranjang. Saat barang dimasukkan, datanya akan disimpan ke *database*. Kemudian ketika akan dibayar, maka di balik layar, aplikasi akan mengelola pembayaran itu dan memberikan respon ke pengguna bahwa pembayaran telah berhasil (Concepta, 2023).

1.6.9 Python

Python merupakan salah satu Bahasa pemrograman yang popular saat ini. Python merupakan bahasa pemrograman tingkat tinggi yang berorientasi obyek yang memiliki semantik yang dinamis. Python sendiri termasuk *interpreted programming language* karena kode dalam Python bisa dijalankan tanpa harus di-*compile* terlebih dahulu. Python sangat mengutamakan *readability* dalam *syntax* yang digunakan sehingga kodenya lebih mudah dipahami. Hal ini juga menyebabkan pengelolaan kode menjadi lebih mudah.

Python telah banyak digunakan untuk berbagai jenis tugas. Python sering digunakan sebagai bahasa pemrograman dalam *Artificial Intelligence* (AI) atau *machine learning*. Hal ini disebabkan karena banyaknya *library* di Python yang mendukung tugas ini, seperti Tensorflow, Keras, Sci-kit Learn, dan masih banyak lagi. Selain itu, Python juga bisa digunakan untuk membuat aplikasi web. Python memiliki beberapa *framework* yang dapat digunakan untuk mengembangkan web, seperti Flask dan Django. Python juga digunakan untuk berbagai tugas lainnya, seperti visualisasi data, komputasi sains, otomatisasi, dan masih banyak lagi.

1.6.10 Flask

Flask merupakan sebuah *framework* yang biasa digunakan sebagai *web service* dan dibangun dengan menggunakan bahasa pemrograman Python. Flask termasuk dalam kategori *microframework* karena Flask tidak memerlukan *library* khusus dalam penggunaannya (Louise & Saian, 2023).

Flask memiliki berbagai kelebihan. Flask sangat mudah dipelajari untuk para pemula. Hal ini didukung dengan dokumentasi bagus yang diberikan dari *developer*

Flask. Flask juga memberikan banyak kebebasan kepada penggunanya untuk membuat proyeknya. Flask juga fleksibel, sehingga apabila ada suatu bagian proyek yang diubah, maka seluruh strukturnya tidak akan kacau. Flask juga lebih ringan karena ia tidak bergantung ke berbagai ekstensi untuk bisa berfungsi. Dengan berbagai kelebihan tersebut, Flask cocok untuk digunakan membuat aplikasi skala kecil (Deery, 2023).

1.6.11 OpenCV

OpenCV merupakan salah satu *library* di Python untuk *computer vision* dan *machine learning*. OpenCV memiliki sekitar 2500 algoritma yang telah dioptimalkan yang dapat digunakan untuk berbagai hal, seperti deteksi dan pengenalan wajah, identifikasi objek, *tracking* pergerakan objek, klasifikasi tindakan manusia di video, dan masih banyak lagi. OpenCV telah diunduh lebih dari 18 juta kali dan telah digunakan oleh berbagai perusahaan ternama di dunia, seperti Google, Microsoft, dan Intel.

1.6.12 OpenPyXL

OpenPyXL merupakan *library* Python yang digunakan untuk membuat dan membaca file Excel 2010 di Python. *Library* ini memungkinkan pengguna untuk melakukan otomatisasi tugas di Excel yang membuatnya menjadi salah satu alat yang cocok digunakan untuk melakukan analisis data, laporan data, dan tugas-tugas administrasi lainnya. OpenPyXL mampu melakukan banyak hal, seperti membuat dan membaca file Excel, mengatur workbook dan worksheet, mengakses dan mengubal *cell* pada Excel, hingga melakukan *styling* dan *formatting* pada Excel.

1.6.13 MySQL

MySQL merupakan salah satu *database* relasional yang banyak digunakan untuk menyimpan data suatu aplikasi. MySQL dikatakan sebagai *database* relasional karena cara penyimpanan datanya yang disimpan dalam sebuah tabel yang terdiri atas beberapa kolom. MySQL menerapkan konsep hubungan antar tabel yang dapat mengatasi masalah seperti inkonsistensi, data duplikat, hingga data yang hilang pada suatu *database*.

MySQL juga menerapakan mekanisme yang bernama *Structured Query Languange* (SQL). SQL merupakan sebuah bahasa yang digunakan untuk mengakses data di *database*. SQL ini memungkinkan pengguna untuk melakukan berbagai operasi di *database*, seperti menulis dan membaca data, menggabungkan tabel, hingga melakukan operasi matematis seperti mencari rata-rata, nilai maksimal, dan menjumlahkan isi data.

1.6.14 Pengukuran Performa

Pengukuran performa dilakukan untuk menguji apakah model dan aplikasi telah menjalankan tugasnya dengan baik. Performa model akan diukur menggunakan penghitungan manual dengan membandingkan jumlah orang yang berhasil dihitung oleh sistem dengan jumlah orang yang sebenarnya. Sehingga,

untuk menilai akurasi model saat dijalankan pada aplikasi, dapat digunakan akurasi berikut:

$$akurasi = \frac{jumlah\ orang\ terdeteksi}{jumlah\ orang\ sebenarnya} \quad (1)$$

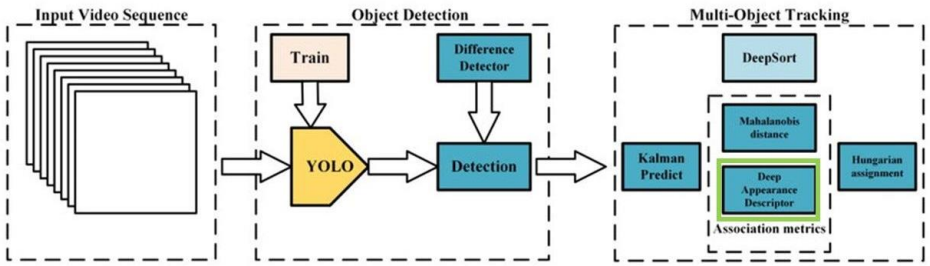
1.6.15 Object Tracking

Object Tracking merupakan suatu permasalahan untuk menentukan suatu lokasi, jalur, dan karakteristik suatu objek berdasarkan data yang diterima dari sensor. Sensor yang digunakan dalam *object tracking* sangat bervariasi, mulai dari radar, sonar, kamera, mikrofon, hingga sensor-sensor lain yang dapat digunakan untuk mengumpulkan informasi mengenai suatu objek dalam suatu lingkungan. *Object tracking* biasanya dilakukan untuk menentukan jumlah, keberadaan, hingga kondisi suatu obyek yang meliputi hal-hal seperti posisi, ketinggian, atau ciri-ciri lain dari objek (Challa, Morelande, Musicki, & Evans, 2011).

Salah satu contoh penerapan *object tracking* ini dapat dilihat pada kamera pengawasan. Kamera pengawas sudah banyak digunakan saat ini khususnya dalam hal keamanan dalam suatu bangunan seperti bandara, bank, mall, stasiun, kantor, dan bahkan rumah pribadi juga sudah banyak yang menggunakannya. Dengan menggunakan kamera seperti CCTV, manusia di suatu area dapat dideteksi perilakunya dan data yang diperoleh dapat digunakan untuk mengambil tindakan lebih lanjut (Challa, Morelande, Musicki, & Evans, 2011).

1.6.16 DeepSORT

DeepSORT merupakan salah satu metode *tracking* yang digunakan dalam *object tracking*. DeepSORT sendiri merupakan peningkatan dari metode *Simple Online and Realtime Tracking* (SORT). DeepSORT sendiri mencoba untuk meningkat efisiensi pergantian ID pada metode SORT sebelumnya. Untuk memperbaiki pergantian ID tersebut, DeepSORT menerapkan *Convolutional Neural Network* (CNN) pada pemberian ID dengan menggunakan informasi pergerakan dan penampilan suatu objek (Wojke, Bewley, & Paulus, 2017).



Gambar 4 Proses *Tracking* pada DeepSORT (Kanjee, 2020)

DeepSORT dimulai setelah dilakukan *object detection* menggunakan model YOLOv8. Setiap objek yang terdeteksi pada *frame* akan disimpan nilainya dalam

vektor d_i yang terdiri atas (u, v, γ, h) dimana (u, v) merupakan titik tengah *bounding box,* γ merupakan aspek rasio, dan h sebagai tinggi. Deteksi objek baru akan dibuatkan *tracking* x_i yang terdiri atas data *bounding box* (u, v, γ, h) serta vektor ciri dari objek tersebut kemudian disimpan ke *measurement space*. Apabila terdapat deteksi baru, maka deteksi tersebut akan dibandingkan dengan *tracking* yang sudah ada.

Setelah itu, untuk mengetahui asosiasi antara hasil deteksi dan *tracking* yang sudah ada, maka ada dua konsep *distance* yang digunakan, yaitu *mahalanobis distance* dan *cosine distance*. *Mahalanobis distance* digunakan untuk mengetahui bagaimana asosiasi antara posisi obyek yang dideteksi dan prediksi posisi *tracking* yang sudah ada sebelumnya yang diperoleh menggunakan Kalman Filter dengan kecepatan konstan dan model observasi linear. Prediksi Kalman Filter akan menghasilkan prediksi posisi *bounding box* yang disimpan dalam Hx_i yang juga terdiri atas (u, v, γ, h).

Adapun rumus yang digunakan adalah sebagai berikut:

$$d^{(1)}(i,j) = (d_j - Hx_i)^T S_i^{-1} (d_j - Hx_i) \quad (2)$$

Keterangan :

$d^{(1)}(i,j) = Mahalanobis\ distance\ untuk\ track\ ke-i\ dan\ deteksi\ ke-j$

$d_j = deteksi\ ke-j,$

$Hx_i = prediksi\ posisi\ track\ ke-i\ pada\ measurement\ space$

$S_i = kovarian\ pada\ track\ ke-i$

Kemudian, selain berdasarkan posisinya, asosiasi deteksi dan *tracking* juga dapat dilakukan berdasarkan tampilan objek. DeepSORT menerapkan CNN dalam prosesnya untuk membantu mengenali suatu objek. DeepSORT sendiri memiliki arsitektur CNN seperti berikut :

Tabel 5 Arsitektur CNN DeepSORT

Nama	Ukuran Patch/Stride	Ukuran Output
Conv 1	3×3 / 1	32 × 128 × 64
Conv 2	3×3 / 1	32 × 128 × 64
Max Pool 3	3×3 / 2	32 × 64 × 32
Residual 4	3×3 / 1	32 × 64 × 32
Residual 5	3×3 / 1	32 × 64 × 32
Residual 6	3×3 / 2	64 × 32 × 16
Residual 7	3×3 / 1	64 × 32 × 16
Residual 8	3×3 / 2	128 × 16 × 8
Residual 9	3×3 / 1	128 × 16 × 8
Dense 10		128
Batch dan normalisasi ℓ_2		128

Dalam DeepSORT, ketika suatu data objek masuk, maka akan dilakukan *feature extraction* dari data tersebut. *Feature extraction* tersebut akan menghasilkan

vektor ciri yang mewakili objek tersebut. Vektor ciri yang telah diperoleh akan digunakan pada rumus *cosine distance* berikut :

$$d^{((2)}(i,j) = \min\{1 - r_j{}^T r_k{}^{(i)} | r_k{}^{(i)} \epsilon R_i\} \quad (3)$$

Keterangan :

$d^{((2)}(i,j) = cosine\ distance\ untuk\ track\ ke-i\ dan\ deteksi\ ke-j$

$r_j = vektor\ ciri\ \ deteksi\ ke-j$

$r_k{}^{(i)} = vektor\ ciri\ ke-k\ pada\ track\ ke-i$

$R_i = kumpulan\ vektor\ ciri\ untuk\ track\ ke-i$

Setelah mendapatkan hasil dari *mahalanobis distance* dan *cosine distance*, maka selanjutnya keduanya akan digabung menjadi rumus *cost* berikut :

$$c_{i,j} = \lambda d^{(1)}(i,j) + (1-\lambda) d^{((2)}(i,j) \quad (4)$$

Keterangan :

$c_{i,j} = association\ cost\ antara\ track\ ke-i\ dan\ deteksi\ ke-j$

$d^{(1)}(i,j) = Mahalanobis\ distance\ untuk\ track\ ke-i\ dan\ deteksi\ ke-j$

$d^{((2)}(i,j) = cosine\ distance\ untuk\ track\ ke-i\ dan\ deteksi\ ke-j$

$\lambda = faktor\ bobot$

Setelah mendapat semua nilai *cost*, maka semua nilai *cost* akan dimasukkan ke dalam *distance matrix*. Dengan menggunakan *hungarian algorithm*, maka akan dicari nilai terkecil dari matrix tersebut. Misalnya, jika nilai $c_{1,1}$ yang terkecil, artinya deteksi d_1 merupakan milik *tracking* x_1. Setelah itu, nilai *tracking* akan diperbarui berdasarkan objek yang terdeteksi tersebut.

1.6.17 Bot Telegram

Telegram merupakan aplikasi media sosial multi-platform yang ditemukan oleh pengusaha asal Rusia bernama Pavel Durov. Telegram merupakan aplikasi yang berfokus pada enkripsi, privasi, dan juga *open-source* (Gordon & Fernandez, 2024). Telegram memiliki salah satu fitur yang banyak digunakan oleh pengembang aplikasi, yaitu fitur Bot

Bot Telegram merupakan aplikasi kecil yang berjalan di dalam telegram. Bot telegram ini dapat digunakan untuk berbagai hal seperti menerima pembayaran, mengirimkan pesan otomatis, dan masih banyak lagi. Salah satu alasan mengapa bot Telegram banyak digunakan adalah bot Telegram bersifat gratis, sehingga pengembang aplikasi bisa menggunakannya tanpa perlu mengeluarkan uang tambahan lagi.

1.6.18 *Black Box Testing*

Black box testing merupakan pengujian suatu sistem/aplikasi dimana penguji tidak memperhatikan struktur kode dari aplikasinya, melainkan fokus sepenuhnya ke respons aplikasi (Williams, 2010). Dalam melakukan *testing*, pembuat aplikasi dapat memposisikan diri mereka sebagai pengguna aplikasi. *Testing* dilakukan dengan

berinteraksi langsung dengan sistem yang telah dibuat. Penguji akan memberikan target dari setiap fitur yang diuji. Jika fitur telah berjalan sesuai dengan perkiraan, maka dapat diasumsikan bahwa sistem telah lolos *black box testing*. Namun, jika respon dari aplikasi tidak sesuai dengan perkiraan, maka diasumsikan ada yang salah dengan aplikasinya sehingga perlu diperbaiki.

Black box testing dilakukan untuk menemukan masalah yang mungkin ditemui saat aplikasi sedang digunakan. Ada beberapa masalah yang biasa ditemukan saat melakukan *black box testing*:

a. Fungsi program yang hilang atau tidak berfungsi semestinya
b. Kesalahan *interface*
c. Kesalahan struktur data yang digunakan di *interface*
d. Kesalahan performa atau perilaku aplikasi
e. Kesalahan inisialisasi dan terminasi

1.6.19 *User Acceptance Test* (UAT)

User acceptance test merupakan tahapan terakhir dari bagian pengembangan aplikasi. Tahapan ini bertujuan untuk mengetahui apakah aplikasi yang dikembangkan telah siap digunakan oleh pengguna. Berbeda dengan *black box testing*, UAT biasanya dilakukan oleh orang lain selain pengembang aplikasi (Leung & Wong, 1997)

UAT bisa dilakukan dengan berbagai cara. UAT bisa dilakukan dengan cara menyebarkan kuesioner kepada para pengguna. Ketika pengguna mencoba aplikasi yang diuji, mereka akan memberikan nilai serta tanggapan untuk setiap skenario yang ada di kuesioner. Mereka juga dapat memberikan saran apabila ada yang perlu dikembangkan dari setiap skenarion. Selain itu, UAT juga dapat dilakukan dengan melakukan wawancara langsung kepada para pengguna aplikasi. Dengan cara ini, pengembang dapat berinteraksi langsung dengan pengguna untuk mengetahui apakah aplikasi yang dikembangkan sudah layak pakai atau belum.

BAB II
METODOLOGI PENELITIAN

2.1 Sumber Data

Data yang digunakan pada penelitian ini adalah dataset COCO tahun 2017 yang terdiri atas data *train* yang terdiri atas 118 ribu gambar*, test* yang terdiri atas 41 ribu gambar*,* dan *validation* yang terdiri atas 5 ribu gambar. Dataset COCO terdiri atas 91 kategori, namun untuk penelitian kali ini hanya kelas *person* saja yang digunakan. Apabila diperlukan data terbaru, maka akan diambil rekaman CCTV dari PT Stechoq Robotika Indonesia untuk membuat model yang baru.

2.2 Waktu dan Tempat Penelitian

2.2.1 Waktu penelitian

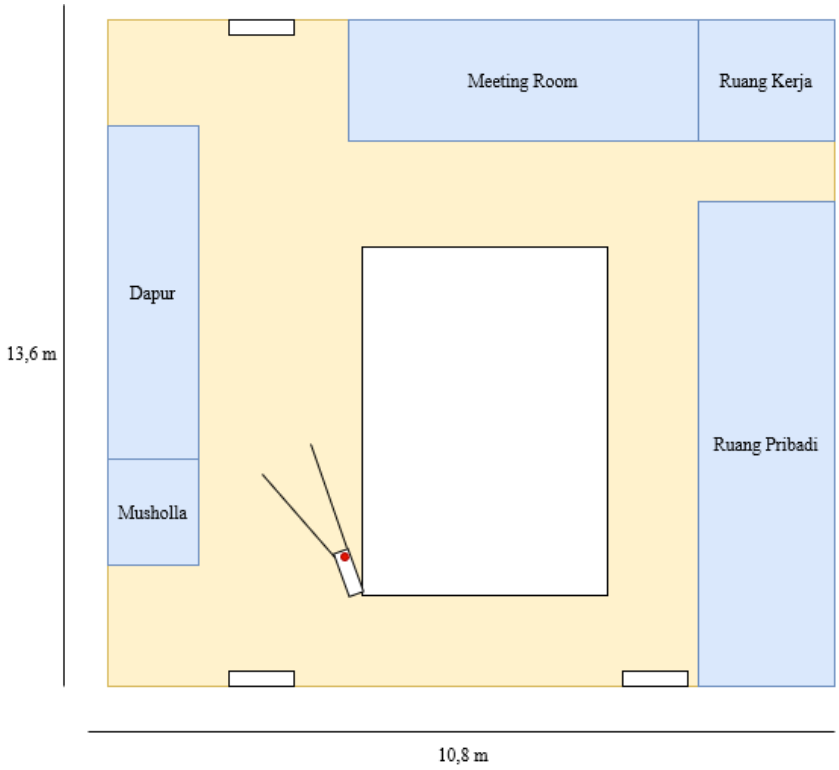
Berikut tabel waktu pelaksanaan mulai dari pengumpulan data hingga pengujian performa aplikasi.

Tabel 6 Waktu Penelitian

No	Tugas	September	Oktober	November	Desember
1.	Analisis Kebutuhan				
2.	Perancangan Sistem				
3.	Implementasi rancangan sistem				
4.	Pengujian sistem				
5.	*Maintenance* sistem				

2.2.2 Tempat penelitian

Penelitian dilakukan di Kantor PT Stechoq Robotika Indonesia yang bertempat di Jalan Sawitsari, Condongcatur, Kecamatan Depok, Kabupaten Sleman, Daerah Istimewa Yogyakarta. Adapun ruang yang digunakan memiliki denah seperti berikut :



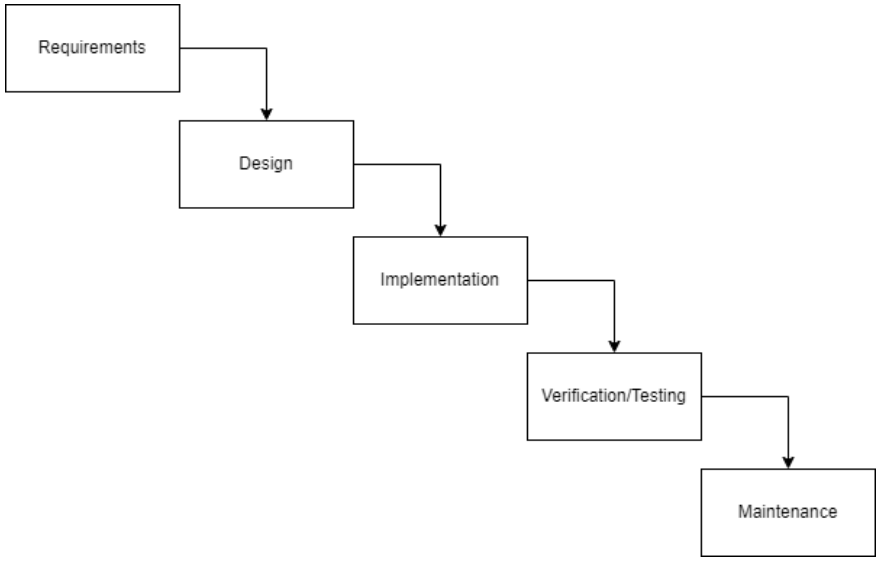
Gambar 5 Denah Tempat Penelitian (13,6 m x 10,8 m)

Adapun gambaran ruangannya dari sudut pandang CCTV adalah sebagai berikut:

Gambar 6 Ruangan dari Sudut Pandang CCTV

2.3 Tahapan Penelitian

Metode penelitian yang digunakan pada aplikasi ini adalah adaptasi dari metode *waterfall*. Metode *waterfall* merupakan metode manajemen proyek yang terdiri atas 5 langkah, yaitu *requirements* (analisis kebutuhan), *design* (perancangan), *implementation* (proses pengerjaan), *verification/testing* (percobaan), dan *maintenance* (pengelolaan) (Hoory & Bottorff, 2024).



Gambar 7 Metode *Waterfall*

2.3.1 *Requirements*

Dalam mengembangkan aplikasi tersebut, ada beberapa *requirement* yang perlu dipenuhi dalam aplikasi yang ingin dibuat. *Requirement* tersebut antara lain :

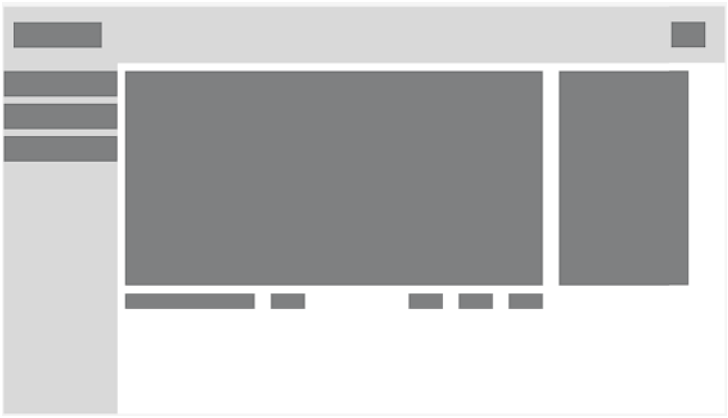
1. Sistem yang dibuat dapat mendeteksi orang yang masuk
2. Sistem yang dibuat dapat menghitung jumlah orang masuk per hari
3. Web dapat melihat informasi statistik jumlah orang masuk dalam jangka waktu tertentu
4. Pembuatan web *dashboard* integrasi dengan model AI

2.3.2 *Design*

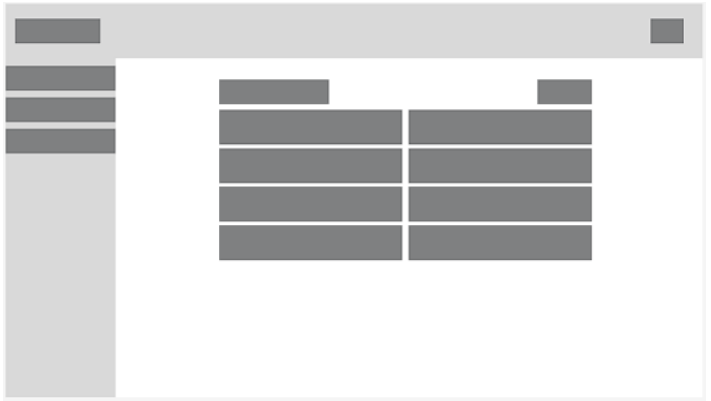
Tahap desain dimulai dengan membuat kerangka/*wireframe* dari tampilan aplikasinya nanti. *Wireframe* ini akan menjadi pedoman saat membuat tampilan dari aplikasi web. Ada beberapa *wireframe* halaman yang telah dibuat, yaitu *wireframe dashboard*, kamera, dan *settings*/pengaturan

Gambar 8 *Wireframe Dashboard*

Gambar 9 *Wireframe* penghitungan

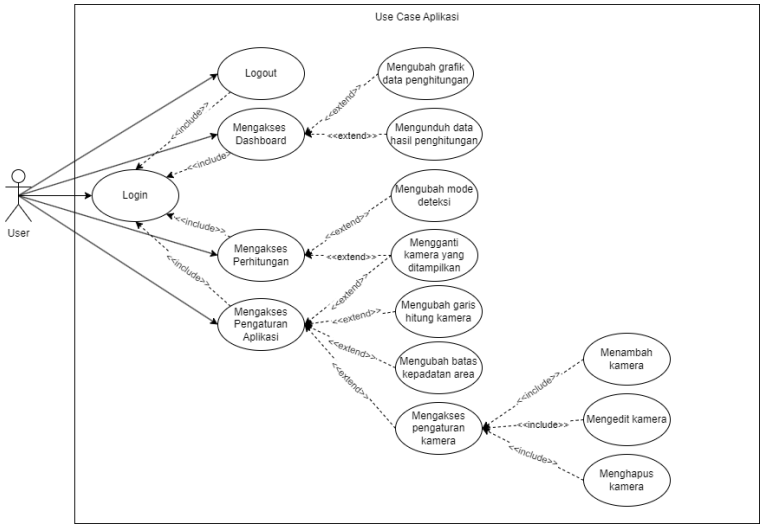


Gambar 10 *Wireframe settings*



Gambar 11 *Wireframe* kamera

Setelah selesai membuat *wireframe*, selanjutnya perlu juga dirancang bagaimana *use case diagram* dari aplikasi yang dibuat. Diagram ini akan menggambarkan bagaimana pengguna akan berinteraksi dengan sistem nantinya.

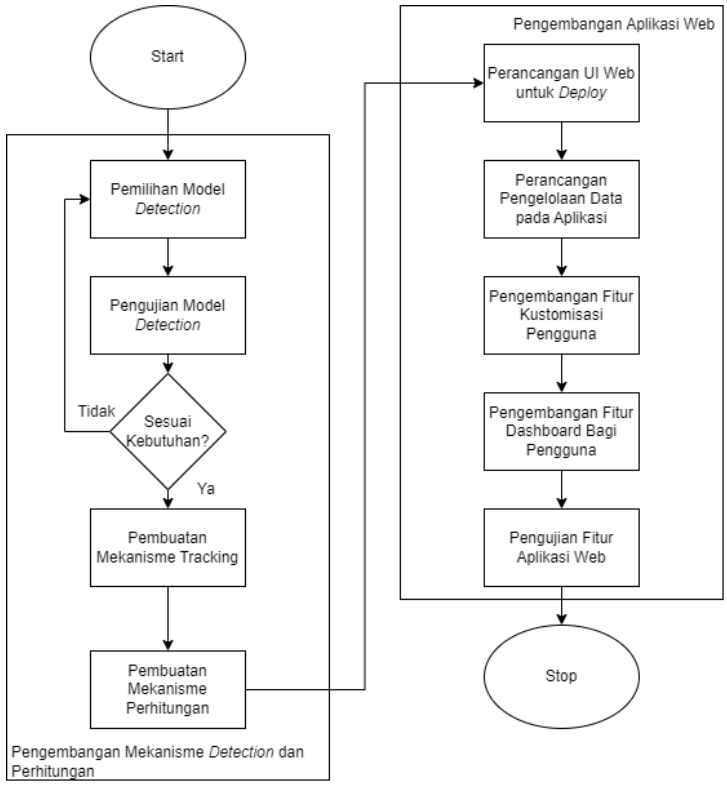


Gambar 12 *Use Case Diagram*

2.3.3 *Implementation*

Tahapan ini adalah tahapan dimana proses koding dimulai. Tahapan ini dimulai dengan mengembangkan mekanisme *detection* dan penghitungan. Pertama, dilakukan pemilihan model *object detection* yang tugasnya sesuai dengan keperluan aplikasi. Setelah diperoleh model yang sesuai, model kemudian diuji untuk mengetahui bagaimana performa dari model. Jika model dirasa sudah sesuai, maka selanjutnya dirancang mekanisme *tracking* yang berbasis model yang dipilih sebelumnya. Apabila perancangan *tracking* sudah selesai, selanjutnya dirancang lagi mekanisme penghitungan objek manusia yang akan dipasangkan bersama dengan *tracking*.

Bagian selanjutnya adalah mengembangkan aplikasi webnya. Pengembangan web dimulai dengan merancang tampilan atau UI untuk webnya. Selain UI, mekanisme pengelolaan data pada aplikasi juga perlu dirancang dengan baik agar fungsional aplikasi berjalan dengan baik. Setelah rancangan UI dan pengelolaan data selesai, selanjutnya akan dibuat fitur kustomisasi untuk pengguna, dimana fitur ini digunakan pengguna untuk mengubah garis hitung yang akan digunakan sebagai indikator dalam melakukan penghitungan objek. Fitur lainnya yang perlu dikembangkan adalah fitur *dashboard* dimana pengguna dapat melihat hasil penghitungan aplikasi melalui *dashboard* tersebut. Setelah semuanya selesai, tahapan terakhir adalah pengujian aplikasi untuk memastikan semuanya berjalan sesuai rencana.



Gambar 13 *Flowchart* Pengembangan

2.3.4 *Testing*

Sistem yang telah dibuat akan diukur performanya secara manual. Hal ini dilakukan dengan membandingkan jumlah orang yang terdeteksi dengan sistem dan jumlah orang sebenarnya yang lewat.

2.3.5 *Maintenance*

Sistem yang telah dibuat, dijalankan selama beberapa hari untuk melihat apakah ada *bug* atau masalah dari aplikasi yang telah dikembangkan. Apabila ditemukan masalah, maka masalah tersebut harus segera diselesaikan.

2.4 Instrumen Penelitian

Adapun instrumen penelitian yang digunakan berupa perangkat keras (*hardware*) dan perangkat lunak (*software*) sebagai berikut:

2.4.1 Perangkat Keras (Hardware)

Adapun *hardware* yang digunakan selama penelitian adalah laptop dan kamera CCTV. Adapun spesifikasi perangkat-perangkat tersebut sebagai berikut:

Tabel 7 Spesifikasi *Hardware*

No	Komponen	Spesifikasi
1.	*Operating System* (OS)	Windows 11 Home
2.	*Processor*	AMD Ryzen 9 5900HX with Radeon Graphics 3.30 GHz
3.	RAM	16 GB
4.	GPU	NVIDIA GeForce RTX 3060

Tabel 8 Spesifikasi CCTV

No	Komponen	Spesifikasi
1.	Merk	HIKVISION
2.	Tipe Model	DS-2CE76D0T-EXIPF
3.	Sensor Gambar	2 MP CMOS
4.	Resolusi	1920 × 1080
5.	*Frame Rate* Rekaman	TVI : 1080P, 25 FPS/30 FPS AHD : 1080P, 25 FPS/30 FPS CVI : 1080P, 25 FPS/30 FPS CVBS: PAL/NTSC
6.	Lensa	2,8 mm *fixed lens*
7.	Pengaturan *Angle*	Geser: 0° - 360° Kemiringan: 0° - 75° Rotasi: 0° - 360°

CCTV yang akan digunakan dipasang dengan posisi seperti pada gambar, yaitu tepat di depan ruangan pekerja. Pemasangan dengan sudut seperti itu bertujuan untuk memastikan bahwa setiap orang yang lewat dapat terlihat dengan jelas dan tidak tertutup oleh objek atau pekerja lain apabila mereka sedang berjalan

bersamaan. Posisi seperti ini akan meminimalisir kemungkinan kesalahan sistem dalam melakukan *tracking*.

Gambar 14 Posisi CCTV yang Terpasang

2.4.2 Perangkat Lunak (Software)

Adapun *software* yang digunakan selama penelitian adalah sebagai berikut:

Tabel 9 Daftar *Software* yang Digunakan

No	*Software*	Kegunaan
1.	Python	Bahasa pemrograman yang digunakan dalam pembuatan model dan pengembangan aplikasi web.
2.	MySQL	*Database* yang digunakan untuk menyimpan data hasil penghitungan okupansi.
3.	Ultralytics	Library yang digunakan untuk menjalankan model YOLO pada program Python.
4.	OpenCV	Library yang digunakan untuk mengolah data gambar yang diterima dari kamera CCTV.
5.	Flask	*Framework* yang digunakan untuk mengembangkan aplikasi web menggunakan Bahasa pemrograman Python.
6.	OpenPyXL	*Library* yang digunakan untuk mengunduh data hasil penghitungan dalam bentuk Excel
7.	Jupyter Notebook	Aplikasi yang digunakan untuk menjalankan *file* berformat .ipynb dalam pengembangan model
8.	Visual Studio Code	Aplikasi yang digunakan untuk menulis kode aplikasi web