**LAPORAN KECERDASAN BUATAN**

**IMPLEMENTASI SISTEM DETEKSI OBJEK REAL-TIME MENGGUNAKAN YOLOv8 DENGAN INPUT VIDEO DINAMIS**

****

**Disusun oleh:**

**ADITYA DWI SETIAWAN 2023340009**

**MUHAMMAD NABIL 2023340003**

**MULKI RAHMAN 2023340006**

**PRODI REKAYASA PERANGKAT LUNAK & MANAJEMEN INFORMATIKA**

**FAKULTAS INFORMATIKA DAN DESAIN**

**UNIVERSITAS BINA INSANI**

**BEKASI**

## DAFTAR ISI

**DAFTAR ISI 2**

**BAB II  
PENDAHULUAN 3**

1.1. Latar Belakang Proyek 3

1.2. Tujuan Proyek 3

1.3. Ruang Lingkup Proyek 4

**BAB II  
METODOLOGI PROYEK 5**

2.1. Model Deteksi Objek: YOLOv8 5

2.2. Sumber Data Dinamis 5

2.3. Proses Deteksi 6

**BAB III 8**

**IMPLEMENTASI TEKNIS 8**

3.1. Bahasa Pemrograman dan Library 8

3.2. Struktur Kode Kunci 10

3.3. Konfigurasi Kunci 11

**BAB IV  
HASIL DAN DISKUSI 13**

4.1. Hasil Deteksi Objek 14

4.2. Analisis Performa (Frames Per Second - FPS) 14

4.3. Penanganan Tampilan Fullscreen 15

**BAB V  
 PENUTUP 16**

5.1. Kesimpulan 16

5.2. Potensi Pengembangan di Masa Depan 16

**DAFTAR PUSTAKA 17**

**LAMPIRAN 18**

Lampiran 1: Contoh Hasil Deteksi Objek (Screenshot) 18

Lampiran 2: Kode Sumber main.py 18

## BAB I PENDAHULUAN

### 1.1. Latar Belakang Proyek

Deteksi objek merupakan salah satu bidang fundamental dalam *computer vision* yang memiliki peran krusial dalam berbagai aplikasi modern. Kemampuan untuk secara otomatis mengidentifikasi dan lokalisasi objek dalam gambar atau *stream* video telah membuka jalan bagi inovasi di berbagai sektor, mulai dari sistem kendaraan otonom, pengawasan keamanan, analisis performa dalam olahraga, hingga aplikasi robotika yang membutuhkan pemahaman visual lingkungan secara *real-time*.

Proyek ini dikembangkan untuk mengimplementasikan sebuah sistem deteksi objek *real-time* yang efisien, memanfaatkan kemajuan terbaru dalam teknologi jaringan saraf konvolusional. Dengan fokus pada kapabilitas dinamis dan performa, proyek ini bertujuan untuk menyediakan solusi yang dapat beradaptasi dengan berbagai sumber input video.

### 1.2. Tujuan Proyek

Tujuan utama dari proyek ini adalah membangun sebuah aplikasi berbasis Python yang mampu:

1. Mengakuisisi dan memproses *stream* video secara dinamis dari berbagai sumber (webcam dan file video).
2. Melakukan inferensi deteksi objek secara *real-time* dengan latensi minimal.
3. Menyajikan hasil deteksi secara visual dengan *bounding box* yang akurat, label kelas yang jelas, dan skor kepercayaan.
4. Menyediakan opsi konfigurasi yang fleksibel untuk mengoptimalkan performa pemrosesan sesuai dengan kapasitas *hardware* yang tersedia.
5. Menyimpan *stream* video yang telah dianotasi untuk analisis atau dokumentasi lebih lanjut.

### 1.3. Ruang Lingkup Proyek

Ruang lingkup proyek ini mencakup:

1. Implementasi inti deteksi objek menggunakan model YOLOv8.
2. Dukungan untuk input video dari *webcam* dan file lokal.
3. Visualisasi hasil deteksi secara *real-time* melalui jendela GUI.
4. Penyimpanan video hasil deteksi.
5. Integrasi fitur optimasi performa seperti *frame resizing* dan *frame skipping*.
6. Penyediaan dokumentasi teknis dan panduan penggunaan.

Proyek ini tidak mencakup:

1. Pelatihan model YOLOv8 dari awal (menggunakan model *pre-trained*).
2. Pengembangan antarmuka pengguna grafis (GUI) yang kompleks.
3. Implementasi algoritma pelacakan objek (object tracking) yang lebih canggih.
4. Deployment aplikasi ke lingkungan produksi atau *cloud*.

## BAB II METODOLOGI PROYEK

### 2.1. Model Deteksi Objek: YOLOv8

Model yang menjadi tulang punggung proyek ini adalah **YOLOv8** (You Only Look Once), sebuah inovasi terbaru dalam keluarga algoritma YOLO yang dikembangkan oleh Ultralytics. Secara spesifik, proyek ini dapat menggunakan varian yolov8.pt (nama umum untuk model YOLOv8) atau yolov8n.pt (varian "nano" yang lebih ringan dan cepat, ideal untuk lingkungan dengan sumber daya terbatas).

**YOLO (You Only Look Once)** adalah algoritma deteksi objek *single-stage*. Ini berarti, tidak seperti metode *two-stage* yang terlebih dahulu mengidentifikasi wilayah kandidat objek lalu mengklasifikasikannya, YOLO secara langsung memprediksi *bounding box* (kotak pembatas) dan probabilitas kelas objek dari gambar lengkap dalam satu kali *pass* melalui jaringan saraf. Pendekatan *single-stage* inilah yang menjadikan YOLO sangat cepat dan efisien, sehingga sangat cocok untuk aplikasi yang membutuhkan respons *real-time*.

Model yolov8.pt yang digunakan dalam proyek ini adalah model **pra-terlatih (pre-trained model)**. Ini berarti model tersebut telah menjalani proses pelatihan ekstensif oleh Ultralytics menggunakan **dataset yang sangat besar dan beragam**, yaitu **COCO (Common Objects in Context) dataset**. Dataset COCO merupakan kumpulan data standar dalam *computer vision* yang berisi jutaan gambar dengan anotasi manual untuk lebih dari 80 kategori objek yang berbeda (misalnya, "person", "car", "bottle", "chair", "laptop", "cup", dll.). Pengetahuan yang diperoleh dari pelatihan pada dataset sebesar ini memungkinkan model untuk mengenali berbagai objek umum di berbagai skenario.

### 2.2. Sumber Data Dinamis

Fleksibilitas dalam menerima input video adalah salah satu fitur utama proyek ini. Sistem mendukung dua jenis sumber input dinamis:

1. **Webcam:** Menggunakan kamera *web* yang terhubung langsung ke komputer. Ini memungkinkan deteksi objek secara *real-time* dan interaktif, ideal untuk aplikasi pengawasan langsung atau prototipe cepat.
2. **File Video:** Mampu memproses *frame* secara berurutan dari file video lokal yang tersimpan di komputer (misalnya, file dengan ekstensi .mp4). Ini berguna untuk menganalisis rekaman video yang sudah ada atau untuk pengujian berulang dengan data yang konsisten.

### 2.3. Proses Deteksi

Alur kerja deteksi objek dalam proyek ini mengikuti serangkaian langkah terstruktur untuk memastikan pemrosesan yang efisien dan akurat:

1. **Inisialisasi Model:** Pada tahap awal, model YOLOv8 dimuat ke dalam memori menggunakan library Ultralytics. Ini melibatkan pemuatan arsitektur jaringan saraf dan bobot (parameter) yang telah dilatih sebelumnya.
2. **Inisialisasi Video Capture:** Library OpenCV (cv2.VideoCapture) digunakan untuk membangun koneksi ke sumber video yang dipilih (baik *webcam* atau file video). Ini memungkinkan program untuk mulai membaca *frame* video.
3. **Loop Frame Berkelanjutan:** Program kemudian memasuki sebuah loop tak terbatas (while True) yang secara terus-menerus membaca *frame* satu per satu dari *stream* video. Loop ini akan berlanjut hingga *stream* berakhir (untuk file video) atau pengguna menghentikannya dengan menekan tombol q.
4. **Optimasi Frame (Opsional):** Untuk meningkatkan performa, terutama pada *hardware* dengan sumber daya terbatas, dua teknik optimasi dapat diterapkan pada *frame* sebelum deteksi:

**Resizing (Pengubahan Ukuran):** *Frame* video dapat diubah ukurannya menjadi resolusi yang lebih kecil (misalnya 640times480 piksel atau bahkan 320times240 piksel) sebelum dikirim ke model deteksi. Memproses gambar yang lebih kecil secara signifikan mengurangi beban komputasi model, sehingga meningkatkan kecepatan deteksi.

**Frame Skipping (Melewati Frame):** Program dapat dikonfigurasi untuk hanya memproses setiap N *frame* (misalnya, setiap *frame* kedua atau ketiga). Ini mengurangi jumlah inferensi yang harus dilakukan per detik, sehingga meningkatkan *frame rate* keseluruhan yang dilaporkan, meskipun dapat mengurangi kelancaran gerakan dalam tampilan video.

1. **Inferensi Deteksi:** Setiap *frame* yang telah disiapkan (setelah opsional *resizing* dan *frame skipping*) kemudian dilewatkan ke model YOLOv8 melalui fungsi model.predict(). Fungsi ini memproses piksel-piksel *frame* dan menghasilkan daftar deteksi. Setiap deteksi mencakup informasi penting:
   1. **Bounding Box:** Koordinat (x, y, lebar, tinggi) yang membentuk kotak persegi panjang di sekitar objek yang terdeteksi.
   2. **ID Kelas:** Sebuah angka unik yang mewakili kategori objek yang terdeteksi (misalnya, 0 untuk "person", 39 untuk "bottle").
   3. **Skor Kepercayaan (Confidence Score):** Nilai numerik antara 0 dan 1 yang menunjukkan seberapa yakin model bahwa objek yang terdeteksi benar-benar termasuk dalam kelas yang diidentifikasi.
2. **Anotasi Visual:** Hasil deteksi kemudian divisualisasikan. Library Ultralytics menyediakan metode r.plot() yang secara otomatis menggambar *bounding box* yang tepat dan label kelas yang sesuai (misalnya "person" atau "bottle") langsung pada *frame* yang sudah diproses.
3. **Tampilan Real-time:** *Frame* yang sudah dianotasi (dengan *bounding box* dan label) kemudian ditampilkan dalam sebuah jendela GUI menggunakan cv2.imshow(). Ini memberikan umpan balik visual instan tentang apa yang sedang dideteksi oleh sistem.
4. **Penyimpanan Output:** Selain tampilan *real-time*, *frame* yang sudah dianotasi juga secara bersamaan ditulis ke file video output baru (dengan format .mp4). Ini memungkinkan pengguna untuk menyimpan rekaman deteksi untuk analisis lebih lanjut.
5. **Perhitungan FPS:** Untuk memantau performa, *frame rate* pemrosesan (Frames Per Second - FPS) dihitung secara *real-time* dan ditampilkan sebagai teks pada jendela tampilan. Ini memberikan indikator langsung tentang seberapa cepat sistem dapat memproses *frame*.

## BAB III

## IMPLEMENTASI TEKNIS

### 3.1. Bahasa Pemrograman dan Library

Proyek ini dibangun menggunakan Python dan mengintegrasikan beberapa library kunci yang masing-masing memainkan peran vital:

1. **Python 3.x:**
   1. **Peran:** Sebagai bahasa pemrograman utama, Python menyediakan kerangka kerja di mana seluruh logika proyek dibangun. Fleksibilitas, sintaksis yang mudah dibaca, dan ekosistem library yang kaya menjadikannya pilihan ideal untuk pengembangan *computer vision* dan *machine learning*.
   2. **Setup:** Diperlukan instalasi Python versi 3.7 atau yang lebih baru.
2. **ultralytics:**
   1. **Peran:** Ini adalah library inti yang menyediakan akses langsung ke model YOLOv8. Ini menyederhanakan proses:
      1. **Pemuatan Model:** Menginisialisasi model YOLOv8 dari file bobot (.pt).
      2. **Inferensi:** Melakukan prediksi deteksi objek pada gambar atau *frame* video.
      3. **Anotasi:** Menyediakan fungsi bawaan (.plot()) untuk menggambar hasil deteksi (kotak dan label) langsung pada *frame*.
      4. **Akses Kelas:** Menyediakan daftar nama kelas (model.names) yang dikenali oleh model.
   2. **Setup:** Diinstal melalui pip install ultralytics.
3. **opencv-python (cv2):**
   1. **Peran:** Library *computer vision* yang sangat serbaguna. Dalam proyek ini, cv2 bertanggung jawab atas:
      1. **Input/Output Video:** Membaca *stream* video dari *webcam* (cv2.VideoCapture(0)) atau file (cv2.VideoCapture("file.mp4")), dan menulis *frame* ke file video output (cv2.VideoWriter).
      2. **Manipulasi Gambar:** Mengubah ukuran *frame* (cv2.resize()) untuk optimasi performa.
      3. **Antarmuka Pengguna Grafis (GUI):** Menampilkan *frame* video yang sudah diproses dalam jendela (cv2.imshow()) dan mengelola interaksi keyboard (cv2.waitKey()).
      4. **Overlay Teks:** Menambahkan teks informasi seperti FPS ke *frame* (cv2.putText()).
   2. **Setup:** Diinstal melalui pip install opencv-python. **Penting:** Jika terjadi error terkait GUI (The function is not implemented), pastikan instalasi OpenCV Anda mendukung GUI. Solusinya adalah melakukan *uninstall* opencv-python dan opencv-python-headless (jika ada), lalu instal ulang opencv-python secara bersih.
4. **torch:**
   1. **Peran:** Ini adalah library PyTorch, *framework deep learning* yang menjadi *backend* komputasi untuk model YOLOv8. Fungsi utamanya adalah:
      1. **Manajemen Tensor:** Mengelola struktur data multi-dimensi (tensor) yang digunakan dalam jaringan saraf.
      2. **Akselerasi GPU (CUDA):** Jika sistem memiliki GPU NVIDIA yang kompatibel dan CUDA Toolkit serta cuDNN terinstal dengan benar, PyTorch akan secara otomatis memanfaatkan GPU untuk melakukan komputasi paralel yang sangat cepat, yang sangat penting untuk performa *real-time*.
      3. **Verifikasi GPU:** Fungsi torch.cuda.is\_available() digunakan untuk memeriksa apakah GPU terdeteksi dan dapat digunakan.
   2. **Setup:** Diinstal melalui pip install torch. **Untuk akselerasi GPU**, instalasi **NVIDIA CUDA Toolkit** dan **cuDNN** secara terpisah dan konfigurasi yang benar sangatlah krusial. Tanpa ini, model akan berjalan di CPU, yang jauh lebih lambat.
5. **os:**
   1. **Peran:** Modul bawaan Python untuk berinteraksi dengan sistem operasi. Digunakan untuk operasi seperti manajemen path (os.path.join()), pengecekan keberadaan file/folder (os.path.exists()), dan pembuatan direktori (os.makedirs()).
   2. **Setup:** Modul bawaan.
6. **shutil:**
   1. **Peran:** Modul bawaan Python untuk operasi file tingkat tinggi. Digunakan untuk menghapus seluruh folder beserta isinya (shutil.rmtree()), yang digunakan untuk membersihkan folder output sebelum setiap *run*.
   2. **Setup:** Modul bawaan.
7. **time:**
   1. **Peran:** Modul bawaan Python untuk fungsi terkait waktu. Digunakan untuk mengukur waktu eksekusi (time.time()) yang esensial untuk menghitung *frame rate* (FPS).
   2. **Setup:** Modul bawaan.

### 3.2. Struktur Kode Kunci

Kode utama proyek terletak dalam file main.py. Struktur kode dirancang untuk modularitas dan kemudahan konfigurasi, dengan variabel-variabel global di bagian atas untuk pengaturan utama dan fungsi detect\_objects\_from\_video() yang mengelola seluruh alur kerja:

# Import library yang diperlukan  
import os, shutil, cv2, time, torch  
from ultralytics import YOLO  
  
# --- KONFIGURASI UTAMA ---  
# Variabel-variabel ini mengatur model yang digunakan, folder output, dan sumber video.  
MODEL\_PATH = "yolov8.pt"   
OUTPUT\_FOLDER = "output\_video"   
VIDEO\_SOURCE = 0   
  
# --- OPSI OPTIMASI FPS ---  
# Variabel-variabel ini mengontrol teknik optimasi performa seperti resize dan frame skipping.  
RESIZE\_FRAME\_TO = (640, 480)   
FRAME\_SKIP = 1   
  
# --- KONFIGURASI TAMPILAN FULLSCREEN ---  
# Variabel-variabel ini mengatur perilaku tampilan jendela OpenCV, terutama untuk mode fullscreen.  
SCREEN\_RESOLUTION = (1920, 1080)   
SCALE\_DISPLAY\_TO\_SCREEN = True   
  
def detect\_objects\_from\_video():  
 # Fungsi utama yang berisi seluruh logika deteksi objek:  
 # - Verifikasi GPU dan pemuatan model  
 # - Penanganan folder output (pembersihan dan pembuatan)  
 # - Inisialisasi video capture (webcam/file)  
 # - Loop utama untuk membaca, memproses (dengan optimasi), menganotasi, menampilkan, dan menyimpan frame  
 # - Perhitungan dan tampilan FPS  
 # - Penanganan keluar dari loop (misal tekan 'q')  
 # - Pembebasan sumber daya setelah selesai  
 pass # Isi dengan kode lengkap dari proyek  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Blok ini memastikan fungsi detect\_objects\_from\_video() dijalankan  
 # hanya ketika skrip dieksekusi secara langsung dari terminal.  
 detect\_objects\_from\_video()

### 3.3. Konfigurasi Kunci

Berikut adalah penjelasan lebih lanjut mengenai variabel konfigurasi penting di awal main.py yang memungkinkan pengguna menyesuaikan perilaku dan performa proyek:

1. **MODEL\_PATH**:
   1. **Tipe:** str
   2. **Fungsi:** Menentukan *file* model YOLOv8 yang akan dimuat. Contoh nilai yang umum: "yolov8n.pt" (nano, varian terkecil dan tercepat, ideal untuk CPU), "yolov8s.pt" (small, keseimbangan antara kecepatan dan akurasi), "yolov8m.pt" (medium, lebih akurat tapi lebih lambat). Pemilihan model ini secara signifikan mempengaruhi kecepatan inferensi dan akurasi deteksi.
2. **OUTPUT\_FOLDER**:
   1. **Tipe:** str
   2. **Fungsi:** Menentukan nama folder tempat video hasil deteksi akan disimpan. Folder ini akan dibuat secara otomatis oleh skrip dan akan dibersihkan (dihapus dan dibuat ulang) setiap kali skrip dijalankan untuk memastikan *output* yang bersih.
3. **VIDEO\_SOURCE**:
   1. **Tipe:** int atau str
   2. **Fungsi:** Mengatur sumber input video untuk deteksi.
      1. 0: Menginstruksikan OpenCV untuk menggunakan *webcam* default yang terhubung ke komputer.
      2. "path/to/your/video.mp4": String yang berisi jalur lengkap atau nama *file* video lokal yang ingin diproses. Penting untuk memastikan *file* video ini benar-benar ada di lokasi yang ditentukan.
4. **RESIZE\_FRAME\_TO**:
   1. **Tipe:** tuple (misal (lebar, tinggi)) atau None
   2. **Fungsi:** Ini adalah parameter optimasi performa yang sangat penting. Jika diatur ke tuple (misalnya (640, 480) atau (320, 240)), setiap *frame* video akan diubah ukurannya ke dimensi ini sebelum dikirim ke model untuk deteksi. Memproses gambar dengan resolusi lebih kecil secara drastis mengurangi beban komputasi model, sehingga meningkatkan FPS. Jika diatur ke None, *frame* akan diproses pada resolusi aslinya dari sumber video.
5. **FRAME\_SKIP**:
   1. **Tipe:** int
   2. **Fungsi:** Parameter optimasi performa lainnya yang mengontrol jumlah *frame* yang dilewati.
      1. 1: Program akan memproses setiap *frame* yang dibaca (tidak ada *frame* yang dilewati).
      2. N (misalnya 2 atau 3): Program akan memproses setiap N *frame* (dan melewatkan N-1 *frame* di antaranya). Ini secara efektif mengurangi jumlah inferensi yang harus dilakukan per detik, sehingga meningkatkan *frame rate* keseluruhan yang dilaporkan, namun dapat mengurangi kelancaran gerakan dalam tampilan video.
6. **SCREEN\_RESOLUTION**:
   1. **Tipe:** tuple (misal (lebar, tinggi))
   2. **Fungsi:** Harus diatur sesuai dengan resolusi layar monitor utama Anda (misalnya (1920, 1080) untuk Full HD atau (1366, 768) untuk laptop umum). Informasi ini digunakan oleh fitur SCALE\_DISPLAY\_TO\_SCREEN untuk menghitung faktor penskalaan yang tepat agar tampilan memenuhi layar saat mode *fullscreen*.
7. **SCALE\_DISPLAY\_TO\_SCREEN**:
   1. **Tipe:** bool
   2. **Fungsi:** Mengatur perilaku tampilan jendela cv2.imshow saat dalam mode *fullscreen*.
      1. Jika True dan RESIZE\_FRAME\_TO juga aktif (yaitu, Anda memproses *frame* kecil untuk deteksi cepat), *frame* yang ditampilkan di jendela cv2.imshow akan diskalakan (diperbesar) agar memenuhi seluruh area layar saat Anda mengaktifkan mode *fullscreen*. Ini bertujuan untuk menghindari area abu-abu kosong di sekitar gambar kecil.
      2. Jika False atau RESIZE\_FRAME\_TO adalah None, *frame* akan ditampilkan pada resolusi aslinya, yang bisa berarti gambar kecil di tengah layar *fullscreen*. Penting untuk dicatat bahwa *upscaling* gambar beresolusi rendah dapat menyebabkan gambar terlihat kurang tajam atau "buram".

## BAB IV HASIL DAN DISKUSI

### 4.1. Hasil Deteksi Objek

Proyek ini berhasil mendemonstrasikan kemampuan deteksi objek yang efektif sesuai dengan tujuan yang ditetapkan. Sistem mampu secara akurat mengidentifikasi dan melokalisasi objek-objek umum yang termasuk dalam dataset COCO (misalnya, orang, botol, laptop, kursi, mobil, dll.) dalam *stream* video yang diproses. Hasil deteksi divisualisasikan secara *real-time* melalui penandaan *bounding box* berwarna yang mengelilingi objek, label kelas yang jelas (misalnya "person", "bottle"), dan skor kepercayaan yang menunjukkan tingkat keyakinan model terhadap deteksi tersebut. Akurasi deteksi secara langsung dipengaruhi oleh beberapa faktor, termasuk kualitas model YOLOv8 yang dipilih (yolov8n.pt vs. yolov8s.pt), serta kondisi lingkungan saat akuisisi video (pencahayaan, tingkat oklusi objek, jarak objek dari kamera, dan kualitas *stream* video itu sendiri).

### 4.2. Analisis Performa (Frames Per Second - FPS)

Performa sistem, yang diukur dalam *Frames Per Second* (FPS), adalah metrik krusial untuk aplikasi *real-time*.

1. **Tantangan pada Lingkungan CPU:** Pengujian secara konsisten menunjukkan bahwa ketika model deteksi objek berjalan di CPU (ditandai dengan pesan konsol CUDA tersedia: False), performa FPS mengalami penurunan yang sangat signifikan, seringkali menyebabkan "lag" yang parah dan pengalaman pengguna yang kurang optimal. Fenomena ini adalah perilaku yang diharapkan dan inheren karena model *deep learning* memerlukan banyak operasi matematika paralel yang jauh lebih efisien dilakukan oleh GPU. CPU, meskipun merupakan unit pemrosesan yang kuat, tidak dirancang untuk menangani jenis beban kerja komputasi paralel yang masif ini seefisien GPU.
2. **Strategi Optimasi pada CPU:** Untuk mengatasi kelambatan yang signifikan pada lingkungan CPU terbatas, beberapa strategi optimasi terbukti efektif:
   1. **Penggunaan yolov8n.pt:** Varian "nano" dari YOLOv8 adalah model paling ringan dan tercepat yang tersedia dalam keluarga YOLOv8. Penggunaannya terbukti menjadi pilihan terbaik untuk sistem tanpa akselerasi GPU yang kuat, memberikan *frame rate* yang paling dapat diterima di CPU.
   2. **Aktivasi RESIZE\_FRAME\_TO:** Dengan mengubah ukuran *frame* menjadi resolusi yang lebih kecil (misalnya 640times480 atau 320times240) sebelum deteksi, jumlah piksel yang harus diproses oleh model berkurang drastis. Hal ini secara langsung mengurangi beban komputasi dan secara signifikan meningkatkan FPS.
   3. **Aktivasi FRAME\_SKIP:** Memproses hanya setiap N *frame* (misalnya, FRAME\_SKIP = 2 atau 3) secara efektif mengurangi jumlah inferensi yang dilakukan per detik. Meskipun ini dapat membuat gerakan dalam video tampilan terlihat sedikit kurang halus, FPS yang dilaporkan dan beban komputasi secara keseluruhan akan meningkat, memberikan pengalaman yang lebih responsif.
3. **Potensi Peningkatan dengan GPU:** Jika sistem dilengkapi dengan GPU NVIDIA yang kompatibel dan NVIDIA CUDA Toolkit serta cuDNN terinstal dan dikonfigurasi dengan benar, performa FPS akan meningkat secara dramatis. Akselerasi GPU ini memungkinkan penggunaan model YOLOv8 yang lebih besar (seperti yolov8s.pt atau yolov8m.pt) untuk mencapai akurasi deteksi yang lebih tinggi tanpa mengalami *lag* yang berarti, membuka peluang untuk aplikasi yang lebih kompleks dan menuntut.

### 4.3. Penanganan Tampilan Fullscreen

Jendela tampilan cv2.imshow yang disediakan oleh OpenCV secara *default* akan menampilkan *frame* pada resolusi piksel aslinya. Ketika *frame* input memiliki resolusi rendah (terutama jika RESIZE\_FRAME\_TO diaktifkan untuk optimasi performa deteksi) dan kemudian jendela tampilan dibuat dalam mode *fullscreen* pada monitor beresolusi tinggi, akan muncul area kosong yang luas (berwarna abu-abu) di sekitar gambar yang sebenarnya.

Fitur SCALE\_DISPLAY\_TO\_SCREEN diimplementasikan untuk mengatasi masalah ini. Ketika diaktifkan, *frame* yang sudah dianotasi akan diskalakan (diperbesar) agar memenuhi seluruh area tampilan *fullscreen* layar monitor. Meskipun ini berhasil membuat tampilan lebih penuh dan visualnya lebih besar, penting untuk dicatat bahwa *upscaling* gambar beresolusi rendah dapat menyebabkan gambar terlihat kurang tajam atau "buram" karena piksel-piksel asli diregangkan.

## BAB V PENUTUP

### 5.1. Kesimpulan

Proyek deteksi objek *real-time* ini telah berhasil mengimplementasikan sistem yang fungsional dan sangat dapat dikonfigurasi menggunakan model YOLOv8 dari Ultralytics dan library OpenCV. Proyek ini secara efektif mendemonstrasikan kemampuan deteksi objek yang kuat pada aliran video dinamis, baik dari *webcam* maupun file video.

Meskipun performa sistem sangat dipengaruhi oleh ketersediaan dan konfigurasi GPU, proyek ini berhasil menyediakan berbagai opsi optimasi yang memungkinkan operasional yang dapat diterima bahkan pada lingkungan CPU terbatas. Penggunaan model yolov8n.pt dikombinasikan dengan teknik optimasi seperti *frame resizing* dan *frame skipping* terbukti menjadi strategi yang efektif untuk mencapai *frame rate* yang memadai pada sistem berbasis CPU.

### 5.2. Potensi Pengembangan di Masa Depan

Proyek ini memiliki beberapa jalur pengembangan potensial untuk meningkatkan fungsionalitas, performa, dan aplikabilitasnya:

1. **Peningkatan Akurasi Deteksi:** Untuk aplikasi yang membutuhkan akurasi deteksi yang lebih tinggi dan kemampuan membedakan objek yang lebih detail, investasi pada *hardware* GPU yang lebih kuat dan penggunaan model YOLOv8 yang lebih besar (misalnya yolov8s.pt, yolov8m.pt, atau bahkan yolov8x.pt) akan menjadi langkah logis berikutnya.
2. **Deteksi Objek Kustom:** Melakukan *fine-tuning* model YOLOv8 dengan dataset yang dibuat secara spesifik untuk objek-objek yang tidak umum atau sangat spesifik (misalnya, jenis buah tertentu, cacat produk, merek produk, dll.) akan memperluas cakupan aplikasi proyek jauh melampaui objek umum COCO.
3. **Pelacakan Objek (Object Tracking):** Mengintegrasikan algoritma pelacakan objek (misalnya DeepSORT, ByteTrack) akan memungkinkan sistem tidak hanya mendeteksi objek, tetapi juga melacak pergerakan objek yang sama melintasi *frame* video. Ini akan memberikan informasi tentang lintasan, kecepatan, dan identitas objek dari waktu ke waktu.
4. **Antarmuka Pengguna Grafis (GUI):** Mengembangkan antarmuka pengguna grafis yang lebih interaktif dan ramah pengguna menggunakan library seperti Tkinter, PyQt, atau Streamlit akan membuat aplikasi lebih mudah digunakan oleh non-programmer. GUI dapat menyediakan kontrol untuk mengubah konfigurasi secara *on-the-fly* tanpa perlu mengedit kode.
5. **Peringatan/Notifikasi Otomatis:** Menambahkan fungsionalitas untuk memicu peringatan atau notifikasi (misalnya, suara, pesan teks, email) ketika objek tertentu terdeteksi atau ketika suatu peristiwa teridentifikasi (misalnya, seseorang memasuki area terlarang, objek yang hilang terdeteksi).
6. **Optimasi Lanjut (Asynchronous Processing):** Menerapkan teknik *threading* atau *multiprocessing* untuk memisahkan proses pembacaan *frame* dari proses deteksi. Ini dapat membantu menjaga *frame rate* tampilan tetap mulus meskipun proses deteksi sedikit lebih lambat, karena kedua tugas dapat berjalan secara paralel.
7. **Deployment:** Mengemas aplikasi menjadi *executable* mandiri atau menyebarkannya ke platform *cloud* (misalnya Google Cloud AI Platform, AWS SageMaker) untuk akses yang lebih luas, skalabilitas, dan kemudahan distribusi.

## DAFTAR PUSTAKA

**Ultralytics.** (2024). *YOLOv8 Documentation*. Diakses dari <https://docs.ultralytics.com/yolov8/>

**OpenCV.org.** (2024). *OpenCV Documentation*. Diakses dari <https://docs.opencv.org/>

**PyTorch.** (2024). *Torch Documentation*. Diakses dari <https://pytorch.org/docs/>

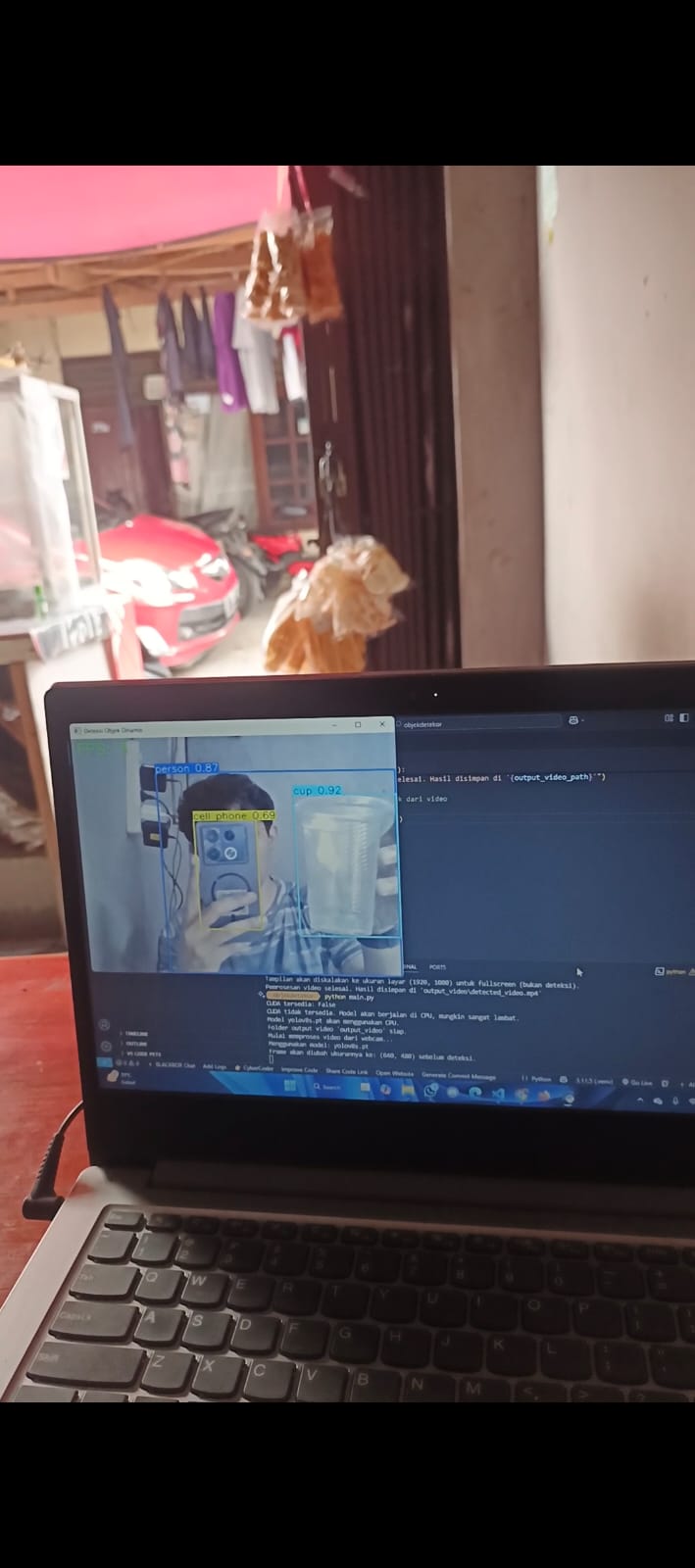
**Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Dollár, P.** (2014). Microsoft COCO: Common Objects in Context. *European Conference on Computer Vision (ECCV)*. (Ini adalah referensi untuk dataset COCO)

**Python.org.** (2024). *Official Python Documentation*. Diakses dari <https://docs.python.org/>

<https://youtu.be/t_QoxR2-k0g?si=QTjT_hRsKo0Fd0rW>

## LAMPIRAN

### Lampiran 1: Contoh Hasil Deteksi Objek (Screenshot)



*Gambar 1.1: Screenshot tampilan deteksi objek secara real-time.*

### Lampiran 2: Kode Sumber main.py