1. test_webcam.py

```
import cv2

# Open the default camera (0),1 , 0 untuk webcam bawaan ,1 untuk webcam
tambahan
cap = cv2.VideoCapture(1)

if not cap.isOpened():
    print("Error: Could not open webcam.")
    exit()

print("Press 'q' to quit.")

while True:
    ret, frame = cap.read()

    if not ret:
        print("Failed to grab frame")
        break

    # Display the resulting frame
    cv2.imshow('Webcam Test', frame)

    # Press 'q' to exit
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# Release everything when done
cap.release()
cv2.destroyAllWindows()
```

2. Jalankan_revisi1.py

```
import subprocess
import os

#path
python_path = os.path.join("env","Scripts","python.exe")

# Jalankan app.py
p1 = subprocess.Popen([python_path, "this3.py"])

# Jalankan main.py
p2 = subprocess.Popen([python_path, "main1.py"])

# Tunggu keduanya selesai (opsional, biasanya biarkan berjalan)
p1.wait()
p2.wait(
```

3. this.py

```
mport cv2
import torch
import pymysql
from datetime import datetime
from ultralytics import YOLO

# Load YOLOv5n (nano, tiny)
model = YOLO("yolov5n.pt")

# ======================
# Database Connection
# ======================
conn = pymysql.connect(
    host='localhost',
    user='root',
    password='',
    database='db_traffic2'
)
cursor = conn.cursor(pymysql.cursors.DictCursor)

# ======================
# Load YOLOv5 Tiny
# ======================
# model = torch.hub.load('ultralytics/yolov5', 'yolov5n', pretrained=True)
vehicle_classes = [2, 3, 5, 7]  # car, motorcycle, bus, truck

# ======================
# Video Source
# ======================
cap = cv2.VideoCapture(1)  # webcam
# cap = cv2.VideoCapture('video/pagi.mp4')

ret, frame = cap.read()
if not ret:
    print("Failed to read from video source.")
    exit()

height2, width2, _ = frame.shape

# ======================
# Tracking & Speed Vars
# ======================
vehicle_id_counter = 0
```

```
tracker_dict = {}
speed_line_y1 = 250
speed_line_y2 = 360
meters_travel_pixel = 0.4
last_vehicle_ids_in_zone = set()

# Background subtractor for motion check
fgbg = cv2.createBackgroundSubtractorMOG2()

# =====================
# Main Loop
# =====================
while True:
    ret, frame = cap.read()
    if not ret:
        break

    # -----------------
    # Contour detection (motion mask)
    # -----------------
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    fgmask = fgbg.apply(gray)
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
    closing = cv2.morphologyEx(fgmask, cv2.MORPH_CLOSE, kernel)
    opening = cv2.morphologyEx(closing, cv2.MORPH_OPEN, kernel)
    dilation = cv2.dilate(opening, kernel)
    _, bins = cv2.threshold(dilation, 220, 255, cv2.THRESH_BINARY)

    contours, _ = cv2.findContours(bins, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)[-2:]

    contour_boxes = []
    for cnt in contours:
        area = cv2.contourArea(cnt)
        if 400 < area < 40000:
            x, y, w, h = cv2.boundingRect(cnt)
            contour_boxes.append((x, y, x + w, y + h))

    # -----------------
    # YOLO detection
    # -----------------
    results = model(frame, verbose=False)  # run inference
    detections = results[0].boxes.data.cpu().numpy()  # (x1,y1,x2,y2,conf,cls)

    vehicle_ids_in_zone = set()
```

```
for *xyxy, conf, cls in detections:
    if int(cls) not in vehicle_classes:
        continue

    x1, y1, x2, y2 = map(int, xyxy)
    cx = int((x1 + x2) / 2)
    cy = int((y1 + y2) / 2)

    # -----------------
    # Cross-check with contour (require overlap)
    # -----------------
    overlap = False
    for (cx1, cy1, cx2, cy2) in contour_boxes:
        if x1 < cx2 and x2 > cx1 and y1 < cy2 and y2 > cy1:
            overlap = True
            break
    if not overlap:
        continue  # ignore YOLO detections without motion

    # -----------------
    # Simple centroid tracking
    # -----------------
    matched_id = None
    for vid, info in tracker_dict.items():
        old_cx, old_cy = info['pos']
        if abs(cx - old_cx) < 50 and abs(cy - old_cy) < 50:
            matched_id = vid
            break

    if matched_id is None:
        matched_id = vehicle_id_counter
        vehicle_id_counter += 1
        tracker_dict[matched_id] = {
            'pos': (cx, cy),
            't1': None,
            't2': None,
            'logged': False,
            'last_cy': cy,
            'direction': None
        }
    else:
        tracker_dict[matched_id]['pos'] = (cx, cy)

    vehicle = tracker_dict[matched_id]
```

```
        last_cy = vehicle['last_cy']
        vehicle['last_cy'] = cy

        # Detection zone check
        if speed_line_y1 <= cy <= speed_line_y2:
            vehicle_ids_in_zone.add(matched_id)

        # Downward detection
        if not vehicle['t1'] and cy >= speed_line_y1:
            vehicle['t1'] = datetime.now()
        elif not vehicle['t2'] and cy >= speed_line_y2:
            vehicle['t2'] = datetime.now()
            vehicle['direction'] = 'bawah'

        # Upward detection
        if not vehicle['t1'] and cy <= speed_line_y2:
            vehicle['t1'] = datetime.now()
        elif not vehicle['t2'] and cy <= speed_line_y1:
            vehicle['t2'] = datetime.now()
            vehicle['direction'] = 'atas'

        # Speed calculation
        if vehicle['t1'] and vehicle['t2'] and not vehicle['logged']:
            delta_time = (vehicle['t2'] - vehicle['t1']).total_seconds()
            if delta_time > 0:
                speed_kmh = (meters_travel_pixel / delta_time) * 3.6
                direction = vehicle['direction'] or 'unknown'
                print(f"ID {matched_id} SPEED: {speed_kmh:.2f} km/h DIR:
{direction}")

                # if speed_kmh >= 10:
                cursor.execute(
                    "INSERT INTO vehicle_log (vehicle_id, direction, timestamp, speed)
VALUES (%s, %s, %s, %s)",
                    (matched_id, direction, datetime.now(), round(speed_kmh, 2))
                )
                conn.commit()
                vehicle['logged'] = True

        # Draw box + ID + Speed
        cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)
        cv2.putText(frame, f"ID:{matched_id}", (x1, y1 - 5),
                    cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 255), 1)

        # if vehicle['t1'] and vehicle['t2']:
```

```
        #     delta_time = (vehicle['t2'] - vehicle['t1']).total_seconds()
        #     if delta_time > 0:
                # speed_kmh = (meters_travel_pixel / delta_time) * 3.6
                # cv2.putText(frame, f"{speed_kmh:.1f} km/h", (x1, y2 + 15),
                #             cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 255, 0), 2)

    # Zone vehicle count
    if vehicle_ids_in_zone != last_vehicle_ids_in_zone:
        last_vehicle_ids_in_zone = vehicle_ids_in_zone.copy()
        count = len(vehicle_ids_in_zone)
        cursor.execute("INSERT INTO total_vehicle (count, timestamp) VALUES
(%s, %s)", (count, datetime.now()))
        conn.commit()

        cursor.execute("SELECT count FROM total_vehicle ORDER BY id DESC
LIMIT 1")
        latest_vehicle_count = cursor.fetchone()
        current_count = latest_vehicle_count['count'] if latest_vehicle_count else 0

        density_level = "low"
        if count >= 10 and count < 20:
            density_level = "medium"
        elif count >= 20:
            density_level = "high"

        # Hitung kepadatan
        L = 20  # panjang jalan (meter)
        N = current_count
        P = N / L  # kepadatan kendaraan per meter
        kendaraanPerKm = round(P * 1000)

        if P <= 0.1:
            kategori = "Sepi"
        elif P <= 0.20:
            kategori = "Sedang"
        else:
            kategori = "Padat"

        cursor.execute("""
            INSERT INTO traffic_density (timestamp, vehicle_count, road_length,
density_value, density_category)
            VALUES (%s, %s, %s, %s, %s)
        """, (datetime.now(), N, L, P, kategori))
        conn.commit()
```

```
            print(f"🚗 Vehicles in zone: {count}")

        # Draw lines
        cv2.line(frame, (0, speed_line_y1), (width2, speed_line_y1), (0, 255, 255), 2)
        cv2.line(frame, (0, speed_line_y2), (width2, speed_line_y2), (255, 0, 255), 2)

        cv2.imshow("YOLOv5 + Contour Vehicle Detection", frame)
        if cv2.waitKey(1) & 0xFF == 27:
            break



cap.release()
cursor.close()
conn.close()
cv2.destroyAllWindows()
```

4. Main.py

```
from flask import Flask, jsonify, render_template
import pymysql

app = Flask(__name__)

# Database config
db_config = {
    'host': 'localhost',
    'user': 'root',
    'password': '',
    'database': 'db_traffic2'
}

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/get_video')
def get_video_data():
    try:
        conn = pymysql.connect(**db_config)
        cursor = conn.cursor(pymysql.cursors.DictCursor)

        # Total vehicle log entries
        cursor.execute("SELECT COUNT(*) AS total FROM vehicle_log")
        total = cursor.fetchone()['total']

        # Last 10 vehicle log entries
        cursor.execute("SELECT * FROM vehicle_log ORDER BY id DESC
LIMIT 10")
        last_records = cursor.fetchall()

        # Get latest count from total_vehicle table
        cursor.execute("SELECT count FROM total_vehicle ORDER BY id DESC
LIMIT 1")
        latest_vehicle_count = cursor.fetchone()
        current_count = latest_vehicle_count['count'] if latest_vehicle_count else 0

        return jsonify({
            'total_entries': total,
            'last_10_entries': last_records,
            'vehicles_in_zone': current_count  # ⬅ NEW FIELD
        })
```

```
    except Exception as e:
        return jsonify({'error': str(e)}), 500

    finally:
        cursor.close()
        conn.close()


if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

5. index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <title>Traffic Monitor</title>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 30px;
         background-color: #f4f4f4;
      }

      h1 {
         color: #333;
      }

      #stats {
         margin-bottom: 20px;
      }

      .stat {
         font-size: 18px;
         margin-bottom: 10px;
      }

      table {
         border-collapse: collapse;
         width: 100%;
         background-color: white;
      }

      th,
      td {
         border: 1px solid #ccc;
         padding: 8px 12px;
         text-align: center;
      }

      th {
         background-color: #eee;
      }
```

```
        tr:nth-child(even) {
            background-color: #f9f9f9;
        }

        .loading {
            color: #888;
        }
    </style>
</head>

<body>
    <h1>Monitoring Trafik</h1>

    <div id="stats">
        <div class="stat">🔢 Total Entri Log Kendaraan: <span id="total-entries"
class="loading">Loading...</span></div>
        <div class="stat">🚗 Kendaraan yang Saat Ini Berada di Antara Jalur: <span
id="vehicle-zone"
                class="loading">Loading...</span></div>
        <div class="stat">🚗 kepadatan trafik saat ini: <span id="kepadatan"
class="loading">Loading...</span></div>
    </div>

    <h2>Kepadatan Trafik Realtime</h2>
    <table>
        <thead>
            <tr>
                <th>Waktu</th>
                <th>Jumlah Kendaraan</th>
                <th>Panjang Jalan (m)</th>
                <th>Nilai Kepadatan</th>
                <th>Kategori Kepadatan</th>
            </tr>
        </thead>
        <tbody id="density-table-body">
            <tr><td colspan="5" class="loading">Loading...</td></tr>
        </tbody>
    </table>

    <h2 id="h2nya">📋 10 Entri Kendaraan Terakhir</h2>
    <table>
        <thead>
            <tr>
                <th>ID</th>
```

```
                <th>ID Kenderaan</th>
                <th>Arah</th>
                <th>Waktu</th>
                <!-- <th>Kecepatan (km/h)</th> -->
            </tr>
        </thead>
        <tbody id="log-table-body">
            <tr>
                <td colspan="5" class="loading">Loading data...</td>
            </tr>
        </tbody>
    </table>

    <script>

        function formatTanggalIndonesia(str) {
            const hariMap = {
                "Sun": "Minggu",
                "Mon": "Senin",
                "Tue": "Selasa",
                "Wed": "Rabu",
                "Thu": "Kamis",
                "Fri": "Jumat",
                "Sat": "Sabtu"
            };

            const parts = str.replace(" GMT", "").split(", ");
            const hari = hariMap[parts[0]] || parts[0];
            return `${hari}, ${parts[1]}`;
        }


        async function loadData() {
            try {
                const response = await fetch('/get_video');
                const data = await response.json();

                // Update stats
                document.getElementById('total-entries').textContent =
data.total_entries;
                document.getElementById('vehicle-zone').textContent =
data.vehicles_in_zone;

                const N = data.vehicles_in_zone;  // Jumlah kendaraan terdeteksi dalam
L meter
```

```
            const L = 20;                  // Panjang zona deteksi (dalam meter)
            const P = N / L;               // Kepadatan kendaraan per meter

            const kendaraanPerKm = Math.round(P * 1000);  // Estimasi kendaraan
dalam 1 km

            if (data.last_density && data.last_density.length > 0) {
               const latestDensity = data.last_density[0];

               let keterangan = `${latestDensity.density_category} | Estimasi
${kendaraanPerKm || '-'} kendaraan per 1 km`;
               document.getElementById('kepadatan').textContent = keterangan;

               const tbodyDensity = document.getElementById('density-table-
body');
               tbodyDensity.innerHTML = '';

               data.last_density.forEach(item => {
                  const row = document.createElement('tr');
                  row.innerHTML = `
                     <td>${formatTanggalIndonesia(item.timestamp)}</td>
                     <td>${item.vehicle_count}</td>
                     <td>${item.road_length}</td>
                     <td>${item.density_value}</td>
                     <td>${item.density_category}</td>
                  `;
                  tbodyDensity.appendChild(row);
               });
            } else {
               document.getElementById('kepadatan').textContent = "Belum ada
data";
            }

            // Populate table
            const tbody = document.getElementById('log-table-body');
            tbody.innerHTML = '';

            data.last_10_entries.forEach(entry => {
               const row = document.createElement('tr');
               row.innerHTML = `
                  <td>${entry.id}</td>
                  <td>${entry.vehicle_id}</td>
                  <td>${entry.direction}</td>
                  <td>${formatTanggalIndonesia(entry.timestamp)}</td>
               `;
```

```
            //  <td>${entry.speed.toFixed(1)}</td>
            tbody.appendChild(row);
        });

        // const jumlahTampil = 10;  // 💡 Ubah sesuai kebutuhan (bisa pakai
input juga)
        // const entriesToShow = data.entries.slice(0, jumlahTampil);  //
Flexible slice

        // entriesToShow.forEach(entry => {
        //     const row = document.createElement('tr');
        //     row.innerHTML = `
        //         <td>${entry.id}</td>
        //         <td>${entry.vehicle_id}</td>
        //         <td>${entry.direction}</td>
        //         <td>${formatTanggalIndonesia(entry.timestamp)}</td>
        //         <td>${entry.speed.toFixed(1)}</td>
        //     `;
        //     tbody.appendChild(row);
        // });


      } catch (err) {
        console.error('Error loading data:', err);
        document.getElementById('log-table-body').innerHTML = `<tr><td
colspan="5">Error loading data</td></tr>`;
      }
    }

    // Initial load
    loadData();

    // Refresh every 10 seconds
    setInterval(loadData, 2500);
  </script>
</body>

</html>
```

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : Riska	**Pembimbing I : Muhammad Basri, S.T.,M.T.**
NIM : 218280185	**Pembimbing II : Ahmad Selao, S.TP.,M.Sc.**
Judul Skripsi : ANALISIS MONITORING KEPADATAN KENDARAAN SECARA REALTIME	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1		Konsultasi 1 Sebaiknya seperti judul aplikasinya	
Konsultasi 2		Konsultasi 2 Acc hasil	
Konsultasi 3		Konsultasi 3 Perbaiki saat seminar hasil telah di tuntaskan	
Konsultasi 4 belajar lagi Acc Hasil		Konsultasi 4 Acc tutup	
Konsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah...

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan ...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare, 12 Juli 2025

Mengetahui
Ketua Program Studi

Mughaffir Yunus, ST., MT.
NBM. 1252 809

Mahasiswa

Riska
NIM. 218280185

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

KARTU MONITORING BIMBINGAN

MAHASISWA PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH PAREPARE

SKRIPSI

Mahasiswa : Riska	**Pembimbing I : Muh. Basri, S.T.,M.T.**
NIM : 218280185	**Pembimbing II : Ahmad Selao,S.TP.,M.Sc.**
Judul Skripsi : SISTEM MONITORING KEPADATAN KENDARAAN SECARA REALTIME	

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 1		Konsultasi 1	
Konsultasi 2		Konsultasi 2	
Konsultasi 3		Konsultasi 3	
Konsultasi 4		Konsultasi 4	
Konsultasi 5		Konsultasi 5	

Lanjut ke halaman sebelah...

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik

Lanjutan...

ARAHAN PEMBIMBING I	HARI/TGL & PARAF PEMBIMBING	ARAHAN PEMBIMBING II	HARI/TGL & PARAF PEMBIMBING
Konsultasi 6		Konsultasi 6 Acc seminar proposal	
Konsultasi 7		Konsultasi 7	
Konsultasi 8		Konsultasi 8	
Konsultasi 9		Konsultasi 9	
Konsultasi 10		Konsultasi 10	

Parepare,

Mengetahui
Ketua Program Studi

Marlina, S.Kom.,M.Kom.
NBM. 1162 680

Mahasiswa

Riska
NIM. 218280185

Perhatian :

1. Mahasiswa wajib konsultasi minimal 5 kali
2. Kartu ini wajib dibawa oleh mahasiswa disetiap konsultasi dan diisi oleh Pembimbing
3. Kartu ini wajib dilampirkan pada laporan skripsi dan menjadi salah satu persyaratan untuk ikut seminar proposal/ujian skripsi
4. Kartu ini dicetak di atas kertas karton berwarna hijau muda dan dicetak timbal balik