Excellence in Learning Innovation

Laporan Praktikum
Matematika Komputer 2
Program untuk menghitung berat badan seseorang

Disusun Oleh:

Kurniawan Rizki Trinanta Sembiring (230535607266)

Mochammad Haidar Ridho (230535605491)

Nadia Fida (230535600298)

PROGRAM STUDI S1 TEKNIK INFORMATIKA

DEPARTEMEN TEKNIK ELEKTRO DAN INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI MALANG

2024

DAFTAR ISI

BAB I
PEMBAHASAN

1.1 Penggunaan yolo v3.weight pada perhitungan dan pengukuran

Yolov3 adalah sebuah model deteksi objek yang terkenal karena kecepatan dan akurasinya yang tinggi. Keunggulan Yolov3 dalam mendeteksi objek membuatnya cocok untuk aplikasi seperti mengukur tinggi dan berat badan seseorang melalui gambar atau video dari kamera. Berikut adalah beberapa alasan mengapa Yolov3 cocok untuk tujuan ini:

1. **Kemampuan untuk Mendeteksi Objek dengan Akurasi Tinggi**

   Yolov3 mampu melakukan identifikasi objek yang luar biasa dengan tingkat presisi yang tinggi. Yolov3 menghasilkan pekerjaan yang cukup baik dalam mendeteksi manusia pada setting ini, yaitu dalam menentukan berat dan tinggi badan.

2. **Kemampuan Lokalisasi yang Baik**

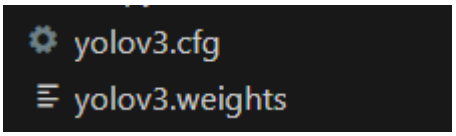
   Dengan memberikan kotak pembatas yang sesuai dengan objek yang diidentifikasi, Yolov3 mampu menentukan lokasi benda pada gambar secara akurat. Dalam hal ini, Yolov3 memberikan kemampuan untuk membuat anotasi lokasi tubuh manusia, yang penting untuk mengukur tinggi badan secara tepat guna menghitung berat badan.

3. **Kemudahan dalam penggunaan**

   Yolov3 mudah diatur saat menggunakan OpenCV dan alat Python lainnya untuk mengimplementasikan kode yang disebutkan di atas. Mempertimbangkan hal ini, mengintegrasikannya ke dalam aplikasi berbasis Python seperti perangkat lunak yang dirancang untuk mengukur berat dan tinggi badan melalui kamera adalah keputusan yang dapat ditepat.

1.2 Penjelasan Kode Program

Penggunaan program Yolo untuk membantu perhitungan



Kode Mengatur Pendeteksian pada kertas Putih:

```
def detect_white_paper(image):
    hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
    lower_white = np.array([0, 0, 200])
    upper_white = np.array([180, 30, 255])
    mask = cv2.inRange(hsv, lower_white, upper_white)
                    contours,    _    =    cv2.findContours(mask,    cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    max_contour = max(contours, key=cv2.contourArea)
    x, y, w, h = cv2.boundingRect(max_contour)
    scaling_factor = 800 / image.shape[1]
    adjusted_width = w * scaling_factor
    print(adjusted_width)
    cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    return image
```

Penjelasan:

- def detect_white_paper(image) : Pada baris pertama ini berfungsi mendeklarasikan detect_white_paper untuk mengambil sebuah gambar sebagai argumen.
- hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV) : Pada baris ini berfungsi mengubah gambar dari mode warna BGR (Blue-Green-Red) menjadi mode warna HVS (Hue-Saturation-Value). Warna putih diidentifikasi lebih baik dalam mode warna HSV.
- lower_white = np.array([0, 0, 200]): Mendefinisikan batas bawah untuk warna putih dalam format HSV.
- upper_white = np.array([180, 30, 255]): Pada baris ini berfungsi mendefinisikan batas atas untuk warna putih dalam format HSV.

- mask = cv2.inRange(hsv, lower_white, upper_white): Pada baris kelima berfungsi menciptakan masker yang menyoroti area di dalam gambar yang memiliki warna putih, sesuai dengan batas warna putih yang telah ditentukan sebelumnya.
- contours, _ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE): Pada baris ke enam berfungsi untuk menemukan kontur dalam masker yang telah dibuat. Kontur adalah tepi dari area putih yang ada dalam masker. Mode RETR_EXTERNAL berarti hanya kontur eksternal yang akan ditemukan (kontur terluar), sedangkan CHAIN_APPROX_SIMPLE akan menyederhanakan kontur dengan hanya menyimpan titik-titik sudut.
- max_contour = max(contours, key=cv2.contourArea): Pada baris ke 7 berfungsi untuk mengambil kontur terbesar berdasarkan luasnya. Kontur ini diasumsikan sebagai kertas putih.
- x, y, w, h = cv2.boundingRect(max_contour): Pada baris ke 8 berfungsi untuk mencari kotak pembatas (bounding box) yang melingkupi kontur terbesar yang telah ditemukan. x dan y adalah koordinat titik pojok kiri atas, sedangkan w dan h adalah lebar dan tinggi kotak pembatas.
- scaling_factor = 800 / image.shape[1]: Pada baris ke 9 berfungsi menentukan faktor skala untuk mengubah lebar kotak pembatas agar sesuai dengan lebar gambar. Lebar gambar akan diubah menjadi 800 piksel, dan faktor skala ini akan digunakan untuk menghitung lebar kotak pembatas yang disesuaikan.
- adjusted_width = w * scaling_factor: Pada baris ke 10 ini berfungsi menghitung ulang lebar kotak pembatas dengan mengalikan lebar aslinya dengan faktor skala yang telah ditentukan.
- cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2): Pada baris ke 11 berfungsi menggambar kotak pada gambar asli dengan menggunakan koordinat yang telah ditemukan sebelumnya. Kotak ini akan digambar di sekitar kertas putih yang terdeteksi.

- return image: Pada kode ini berfungsi mengembalikan gambar yang telah dimodifikasi dengan menambahkan kotak di sekitar kertas putih yang terdeteksi.

Kode Mengatur Pendeteksian Pada Objek Manusia:

```
# Load YOLO
net = cv2.dnn.readNet("yolov3.weights", "yolov3.cfg")
layer_names = net.getLayerNames()
output_layers = [layer_names[i - 1] for i in net.getUnconnectedOutLayers()]

# Load nama kelas (coco.names)
with open("coco.names", "r") as f:
    classes = f.read().strip().split("\n")

# Load model Mediapipe Pose
mp_pose = mp.solutions.pose
pose = mp_pose.Pose()

# Buka webcam
cap = cv2.VideoCapture(0)

# Jarak kita terhadap kamera (dalam meter)
jarak_kamera = 300

# Inisialisasi variabel untuk menyimpan posisi y tulisan
posisi_y_teks = {
    "Tinggi Diperkirakan": 20,
    "Jarak Bahu": 40,
    "Radius Bahu": 60,
    "Pengukuran Tinggi": 80,
    "Berat Diperkirakan": 100,
    "Jarak Manusia": 120,
}

while True:
    # Baca frame dari webcam
    ret, frame = cap.read()
    frame_kertas = detect_white_paper(frame)
    # Deteksi objek menggunakan YOLO
    tinggi, lebar, channel = frame.shape
    blob = cv2.dnn.blobFromImage(
        frame, 0.00392, (416, 416), (0, 0, 0), True, crop=False
    )
    net.setInput(blob)
    outs = net.forward(output_layers)

    # Informasi deteksi objek
    tinggi_orang = []
```

```
  for out in outs:
      for detection in out:
          scores = detection[5:]
          class_id = np.argmax(scores)
          confidence = scores[class_id]
          if confidence > 0.5 and class_id == 0:  # Kelas 0 adalah 'orang'
              w = detection[2] * lebar
              h = detection[3] * tinggi

              # Hitung tinggi berdasarkan perbandingan tinggi sebenarnya dengan tinggi di dalam layar
              tinggi_orang_ini = (h / tinggi) * jarak_kamera
              tinggi_orang.append((tinggi_orang_ini, detection))
```

Penjelasan:

Kode di atas berfungsi membuka webcam, menggunakan YOLO untuk mendeteksi objek (orang) dalam frame, serta menggunakan MediaPipe untuk estimasi pose tubuh. Selain itu, juga terdapat fungsi detect_white_paper() yang bertujuan untuk mendeteksi kertas putih. Kemudian, loop utama membaca frame dari webcam, mendeteksi objek, mengukur tinggi orang dan jaraknya dari kamera, serta menampilkan informasinya di layar.

Kode Mengatur Estimasi Berat Badan:

```
# Hitung volume tabung
def fungsi_integral(y):
  r_dikuadrat = (radius_bahu_cm) ** 2  # Konversi cm ke meter
  return pi * r_dikuadrat

  volume, _ = quad(fungsi_integral, 0, pengukuran_tinggi_cm)  # Konversi cm ke meter
# Estimasi berat badan (densitas manusia ~1 kg/L)
  berat_diperkirakan_kg = volume   # Konversi ke gram
  berat_diperkirakan_kg /= 1000  # Konversi gram ke kilogram
```

Penjelasan:

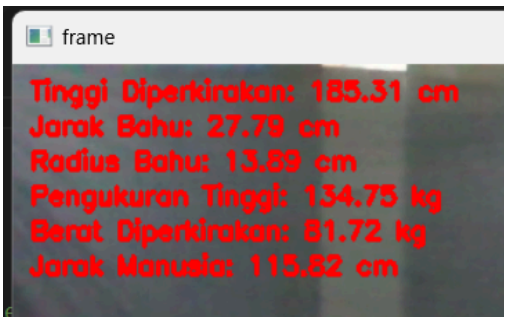
Kode `fungsi_integral` berfungsi menghitung volume tabung berdasarkan radius bahu yang diberikan. Kemudian, volume tabung tersebut digunakan untuk mengestimasi berat badan dengan asumsi densitas manusia sekitar 1 kg/L. Volume

diubah menjadi kilogram untuk mendapatkan estimasi berat badan dalam kilogram.

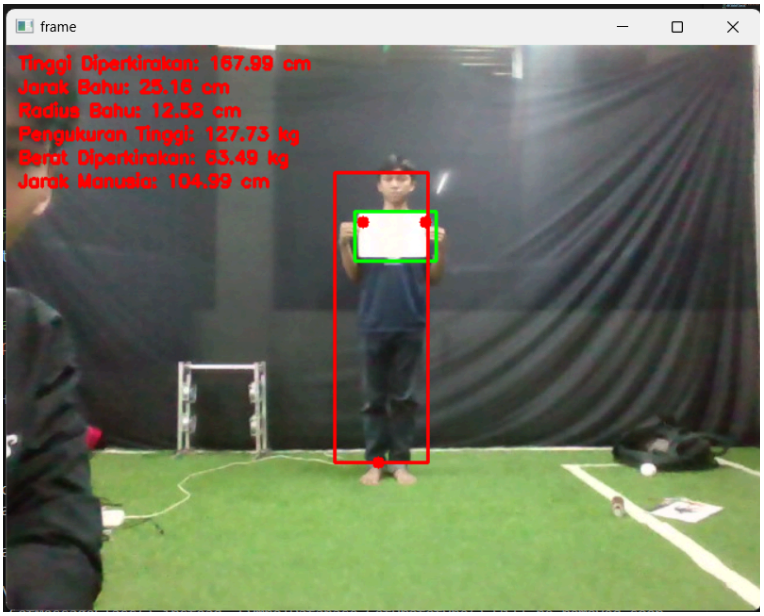
1.3 Keefektifan penggunaan kertas pada perhitungan

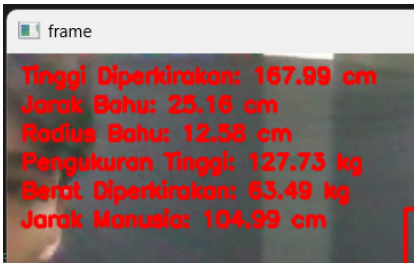
Karena penggunaan yolo sebagai pendeteksi objek yang lebih akurat oleh karena itu kami hanya menggunakan yolo sebagai kunci utama dari proses perhitungan berat badan yang kami lakukan dengan memanfaatkan opencv sebagai media untuk menangkap dan pengenalan terhadap tubuh manusia. Mengenai keefektifan dari Penggunaan kertas digunakan sebagai referensi yang stabil dan mudah diidentifikasi, sebenarnya kami tidak memanfaatkan algoritma untuk menghitung kertas dan lebih menggunakan yolo karena tingkat akurasi yang tinggi dan kertas sendiri sebagai elemen pendukung atau patokan visual, bukan sebagai elemen utama dalam proses perhitungan. Untuk memperjelas kembali mengenai kode program terdapat fungsi untuk mendeteksi kertas fungsi tersebut hanya ditujukan untuk mendeteksi kertas saja. Sementara untuk hasil bisa dilihat bahwa penggunaan fungsi detect_white_paper hanya untuk mendeteksi dan tidak membantu untuk perhitungan. Efektivitas penggunaan kertas sebagai tolok ukur komputasi bergantung pada seberapa baik kertas diidentifikasi dalam proses dan dapat digunakan sebagai referensi visual.

1.4 hasil tangkapan



Ini adalah percobaan tangkapan pertama dari program perhitungan. Hasil dari perhitungan adalah sebagai berikut didapat tinggi diperkirakan adalah 185.31 cm dan beratnya adalah 81.72 kg.





Ini adalah percobaan tangkapan kedua dengan orang yang berbeda dari program perhitungan. Hasil dari perhitungan adalah sebagai berikut didapat tinggi diperkirakan adalah 167.99 cm dan beratnya adalah 63.49 kg.