SENTRI: Jurnal Riset Ilmiah

Vol.3, No.6 Juni 2024

ejournal.nusantaraglobal.ac.id/index.php/sentri

DETEKSI OBJEK REAL TIME DENGAN YOLOV4-TINY DAN ANTARMUKA GRAFIS MENGGUNAKAN OPENCV PYTHON

Yandi Anzari[1], Fitra Novriadi[2], Noni Rahmawati[3], Rizky Nurman Aktan[4], Fattachul Huda Aminuddin[5], Teuku Djauhari[6]

[1]Prodi Sistem Informasi Universitas Nurdin Hamzah
[2]Prodi Sistem Informasi Universitas Nurdin Hamzah
[3]Prodi Sistem Informasi Universitas Nurdin Hamzah
[4]Prodi Sistem Informasi Universitas Nurdin Hamzah
[5]Prodi Sistem Informasi Universitas Nurdin Hamzah
[6]Prodi Sistem Informasi Universitas Nurdin Hamzah
E-mail: Yandianzari.yandi@gmail.com

Article History:
Received: 30-04-2024
Revised :25-05-2024
Accepted:31-05-2024

Keywords:
Deteksi objek, YOLOv4-tiny, antarmuka grafis, OpenCV, real-time

Abstract: *Penelitian ini menggambarkan implementasi integrasi model deteksi objek YOLOv4-tiny ke dalam sistem pendeteksian objek real-time yang dilengkapi dengan antarmuka grafis pengguna menggunakan OpenCV. YOLOv4-tiny dipilih untuk efisiensi deteksi objek dalam setiap frame kamera, sementara antarmuka pengguna memungkinkan interaksi melalui tombol-tombol yang dapat diaktifkan atau dinonaktifkan. Metode ini dirancang untuk memberikan solusi yang mudah digunakan untuk mendeteksi dan memvisualisasikan objek-objek dalam waktu nyata, sambil memberikan kontrol jenis objek kepada pengguna. Inisialisasi model YOLOv4-tiny, pengelolaan antarmuka pengguna dengan tombol-tombol, dan proses deteksi objek diintegrasikan dengan harmonis. Hasil eksperimen menunjukkan bahwa sistem ini dapat mendeteksi dan menampilkan objek-objek sesuai dengan preferensi tombol yang diaktifkan oleh pengguna, memberikan pengalaman interaktif dan informatif. Penelitian ini berkontribusi pada pengembangan sistem deteksi objek real-time yang mudah dioperasikan dan dapat diadaptasi untuk berbagai keperluan aplikasi.*

PENDAHULUAN

Dalam era teknologi informasi yang terus berkembang, deteksi objek real-time menjadi aspek penting dalam berbagai aplikasi, seperti keamanan, pengenalan wajah, dan

analisis video. Salah satu model yang menonjol dalam hal efisiensi adalah YOLOv4-tiny, yang mampu mendeteksi objek dengan cepat dan akurat. Namun, integrasi model ini ke dalam sistem pendeteksian objek real-time dengan antarmuka grafis pengguna masih menjadi tantangan yang perlu diatasi. Oleh karena itu, penelitian ini bertujuan untuk mengimplementasikan YOLOv4-tiny dalam konteks sistem yang dilengkapi antarmuka grafis pengguna menggunakan OpenCV.

LANDASAN TEORI

YOLOv4-tiny (You Only Look Once-tiny)

YOLOv4-tiny merupakan versi ringan dari model YOLOv4 (You Only Look Once), sebuah model deteksi objek yang terkenal karena kecepatan dan akurasi tinggi. Model ini menggunakan arsitektur Convolutional Neural Network (CNN) yang dirancang khusus untuk mendeteksi objek dalam suatu gambar. YOLOv4-tiny dikembangkan dengan fokus pada efisiensi komputasional, menjadikannya pilihan yang cocok untuk aplikasi real-time dengan sumber daya terbatas. Kemampuannya dalam mendeteksi objek dengan cepat menjadikannya solusi yang sangat relevan dalam konteks sistem pendeteksian objek real-time.

You Only Look Once (YOLO) adalah model deteksi objek yang telah mengalami evolusi sejak versi pertamanya. YOLOv4- tiny, versi yang lebih ringan, dikembangkan untuk memenuhi kebutuhan deteksi objek real-time dengan sumber daya terbatas. Sebagai ungkapan dari efisiensi, Liu et al. (2020) menyatakan bahwa YOLOv4-tiny mempertahankan tingkat akurasi yang memadai sambil meningkatkan kecepatan deteksi.

OpenCV

OpenCV merupakan perpustakaan (library) sumber terbuka yang menyediakan berbagai alat dan fungsi untuk pengolahan citra dan penglihatan komputer. OpenCV, sebagai perpustakaan pengolahan citra yang luas, menyediakan solusi bagi para peneliti dalam pengembangan aplikasi penglihatan komputer. Menurut S. Bradski (2000), pendiri OpenCV, perpustakaan ini memainkan peran kunci dalam mempermudah pengembangan algoritma pengolahan citra dan komputer vision dengan menyediakan berbagai alat dan fungsi yang dapat digunakan secara fleksibel.

Dengan dukungan untuk berbagai bahasa pemrograman, termasuk Python, OpenCV menjadi pilihan populer dalam pengembangan aplikasi penglihatan komputer. Kelebihan OpenCV meliputi kemampuan untuk memproses gambar dan
video, melakukan deteksi objek, serta menyediakan berbagai algoritma pengolahan citra yang efisien. Dalam konteks penelitian ini, OpenCV digunakan untuk mengintegrasikan model deteksi YOLOv4- tiny ke dalam sistem pendeteksian objek real- time.

Python

Python adalah bahasa pemrograman yang populer dan sering digunakan dalam pengembangan aplikasi pengolahan citra dan kecerdasan buatan. Dalam literatur Python, VanderPlas (2016) menyatakan bahwa Python memiliki sintaksis yang intuitif dan

kekuatan dalam analisis data, pengembangan web, serta kecerdasan buatan. Keberhasilan Python dalam pengolahan citra dan machine learning membuatnya menjadi pilihan utama dalam pengembangan aplikasi penglihatan komputer.

Kelebihan Python mencakup sintaksis yang mudah dimengerti, berbagai pustaka yang mendukung pengolahan citra, dan fleksibilitas dalam integrasi dengan perangkat keras dan perangkat lunak lainnya. Dalam konteks penelitian ini, Python menjadi bahasa pemrogramanutama untuk mengimplementasikan integrasi antara YOLOv4-tiny dan OpenCV.

Deteksi Objek Real-Time

Deteksi objek real-time adalah teknik yang memungkinkan sistem untuk secara kontinu dan segera mengenali dan melokalisasi objek dalam suatu lingkungan atau media, seperti gambar atau video. Hal ini mencakup pemrosesan citra atau video secara instan, sehingga hasil deteksi dapat diperoleh dalam waktu nyata. Deteksi objek real-time memiliki aplikasi luas, termasuk keamanan, transportasi, dan pengenalan wajah.

Menurut Redmon et al. (2018), deteksi objek real-time menjadi semakin penting dengan perkembangan teknologi. Sistem yang mampu mendeteksi objek secara cepat dan akurat memberikan kontribusi besar dalam berbagai aplikasi, termasuk keamanan, transportasi, dan pengenalan wajah.

Dalam penelitian ini, fokusnya adalah pada implementasi deteksi objek real-time menggunakan YOLOv4-tiny dan OpenCV dengan dukungan antarmuka pengguna untuk pengontrolan yang lebih mudah.

METODE PENELITIAN

Metodologi penelitian ini disusun untuk memberikan panduan yang sistematis dalam pembuatan sistem deteksi objek menggunakan OpenCV Python dan YOLOv4-tiny. Berikut adalah langkah- langkah dan proses yang diikuti dalam penelitian ini:

1. Persiapan Bahan dan Alat:

Memastikan lingkungan Python terinstal dengan OpenCV. Mengunduh dan mempersiapkan berat model (weights) dan konfigurasi model (cfg) YOLOv4-tiny. Menyusun file yang berisi daftar kelas objek yang akan dideteksi.

Gambar 1. file yang dibutuhkan

Gambar 2. isi folder dnn_model

2. Inisialisasi OpenCV:

Mengimpor dan menginisialisasi library OpenCV di dalam skrip Python. Menyiapkan kamera atau video sebagai sumber feed gambar real-time.

```
import cv2
```

Gambar 3. Inisialisasi OpenCV

```
cap = cv2.VideoCapture(0)
```

Gambar 4. Inisialisasi Kamera

3. Konfigurasi Model YOLOv4-tiny: Membaca Model_Wightdan Model_Config menggunakan OpenCV.

```
MODEL_WEIGHTS = "dnn_model/yolov4-tiny.weights"
MODEL_CONFIG = "dnn_model/yolov4-tiny.cfg"
CLASSES_FILE = "dnn_model/classes.txt"
CONFIDENCE_THRESHOLD = 0.3
NMS_THRESHOLD = 0.4
```

Gambar 5. Membaca model_wight & model_config

```
net = cv2.dnn.readNet(MODEL_WEIGHTS, MODEL_CONFIG)
model = cv2.dnn_DetectionModel(net)
model.setInputParams(size=(320, 320), scale=1/255)
```

Gambar 6. Opencv DNN

4. Definisi Antarmuka Pengguna (GUI):

Membuat antarmuka pengguna yang responsif dengan tombol-tombol menggunakan fungsi OpenCV. Menetapkan tata letak, gaya, dan aksi tombol-tombol pada GUI.

```
import cv2
import numpy as np

class Buttons:
    def __init__(self):
        self.font = cv2.FONT_HERSHEY_PLAIN
        self.text_scale = 1
        self.text_thick = 1
        self.x_margin = 10
        self.y_margin = 5


        self.buttons = {}
        self.button_index = 0
        self.buttons_area = []

        np.random.seed(0)
        self.colors = []
        self.generate_random_colors()

    def generate_random_colors(self):
        for i in range(91):
            random_c = np.random.randint(256, size=3)
            self.colors.append((int(random_c[0]), int(random_c[1]), int(random_c[2])))

    def add_button(self, text, x, y):
        textsize = cv2.getTextSize(text, self.font, self.text_scale, self.text_thick)[0]
        right_x = x + (self.x_margin * 2) + textsize[0]
        bottom_y = y + (self.y_margin * 2) + textsize[1]

        self.buttons[self.button_index] = {"text": text, "position": [x, y, right_x, bottom_y], "
        self.button_index += 1

    def display_buttons(self, frame):
        for b_index, button_value in self.buttons.items():
            button_text = button_value["text"]
            (x, y, right_x, bottom_y) = button_value["position"]
            active = button_value["active"]

            if active:
                button_color = (0, 0, 200)
                text_color = (255, 255, 255)
                thickness = -1
            else:
                button_color = (0, 0, 200)
                text_color = (0, 0, 200)
                thickness = 3
```

```
            cv2.rectangle(frame, (x, y), (right_x, bottom_y),
                          button_color, thickness)
            cv2.putText(frame, button_text, (x + self.x_margin, bottom_y - self.y_margin),
                        self.font, self.text_scale, text_color, self.text_thick)
        return frame

    def button_click(self, mouse_x, mouse_y):
        for b_index, button_value in self.buttons.items():
            (x, y, right_x, bottom_y) = button_value["position"]
            active = button_value["active"]
            area = [(x, y), (right_x, y), (right_x, bottom_y), (x, bottom_y)]

            inside = cv2.pointPolygonTest(np.array(area, np.int32), (int(mouse_x), int(mouse_y)), False)
            if inside > 0:
                print("IS Ac", active)
                new_status = False if active is True else True
                self.buttons[b_index]["active"] = new_status

    def active_buttons_list(self):
        active_list = []
        for b_index, button_value in self.buttons.items():
            active = button_value["active"]
            text = button_value["text"]
            if active:
                active_list.append(str(text).lower())

        return active_list
```

Gambar 7. GUI Button

```
button = Buttons()
button.add_button("person", 20, 20)
button.add_button("cell phone", 20, 50)
button.add_button("keyboard", 20, 80)
button.add_button("remote", 20, 110)
button.add_button("scissors", 20, 140)
```

Gambar 8. Inisialisasi Button

5. Loop Pemrosesan Gambar:

Masuk ke dalam loop utama untuk memproses setiap frame yang diterima dari kamera atau video.

```
while True:

    ret, frame = cap.read()

    if not ret:
        print("Error: Failed to read frame from the camera.")
        break
```

Gambar 9. Loop pemrosesan gambar

6. Pengolahan Hasil Deteksi:

Menganalisis hasil deteksi objek, termasuk kelas objek, skor, dan kotak pembatas. Menampilkan informasi objek yang sesuai dengan tombol-tombol yang aktif pada antarmuka pengguna.

```
(class_ids, scores, bboxes) = model.detect(frame, confThreshold=CONFIDENCE_THRESHOLD, nmsThreshold=NMS_THRESHOLD)
for i, (class_id, score, bbox) in enumerate(zip(class_ids, scores, bboxes)):
    (x, y, w, h) = bbox
    class_name = classes[class_id]
    color = colors[class_id]

    if class_name in button.active_buttons_list():
        cv2.putText(frame, class_name, (x, y - 10), cv2.FONT_HERSHEY_PLAIN, 3, color, 2)
        cv2.rectangle(frame, (x, y), (x + w, y + h), color, 3)
```

Gambar 10. pengolahan hasil deteksi

7. Tampilan Real-Time:

Menampilkan antarmuka pengguna dan hasil deteksi objek pada setiap frame yang sedang diproses. Memberikan umpan balik visual secara real-time kepada pengguna.

```
cv2.imshow("Frame", frame)
```

Gambar 11. Tampilan real-time

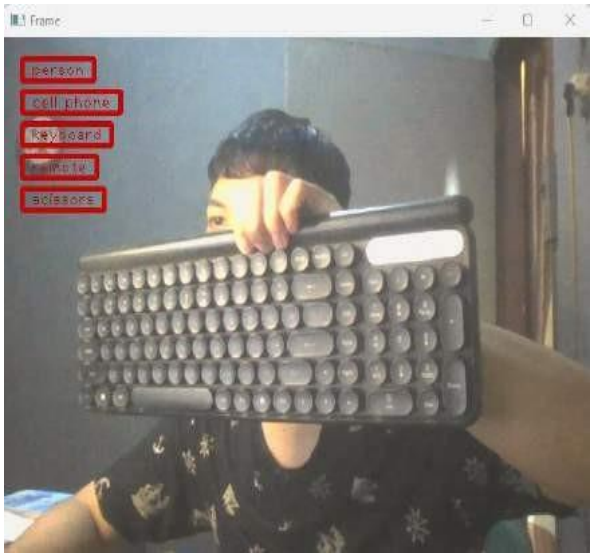
Gambar 12. Hasil tampilan

Gambar 13. Hasil Deteksi

8. Pengendalian dan Interaksi Pengguna:

Menerapkan fungsi callback mouse untuk merespons klik tombol pada antarmuka pengguna. Mengaktifkan atau menonaktifkan deteksi objek spesifik sesuai dengan interaksi pengguna.

Gambar 14. Button Aktif



Gambar 15. Button Nonaktif

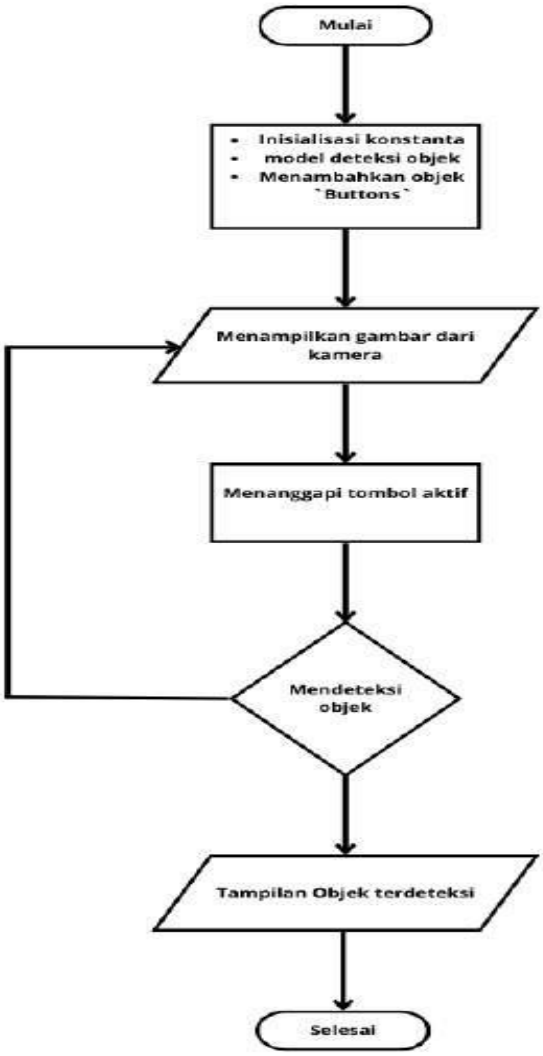
9. Penghentian Aplikasi:

Menanggapi input pengguna untuk mengakhiri aplikasi dengan aman, seperti menekan tombol Esc. Memastikan kamera dilepaskan setelah aplikasi dihentikan.

```
if key == 27:  # 27 is the ASCII code for Esc key
    break
```

Gambar 16. Fungsi Esc

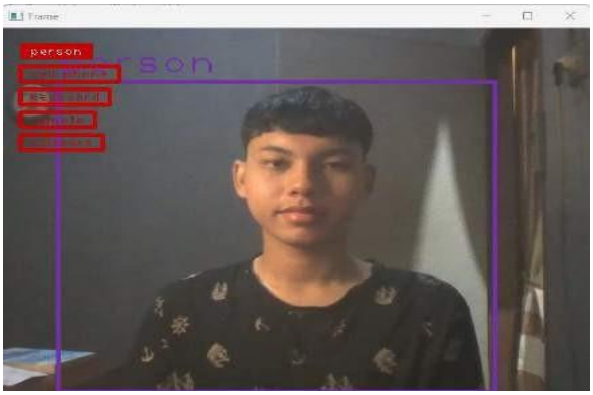
Flowchart



HASIL DAN PEMBAHASAN

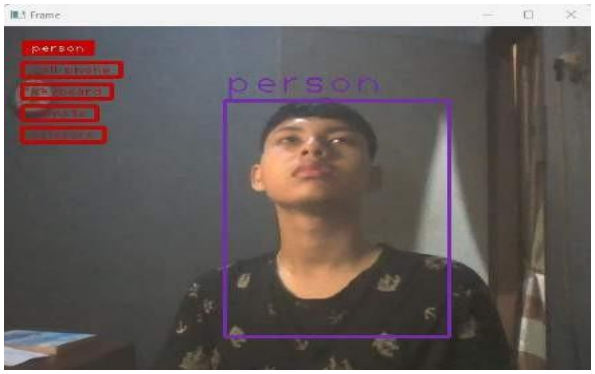
Ada beberapa objek yang dapat di deteksi oleh aplikasi ini yaitu person, cell phone, keyboard, remote, dan scissor. Berikut hasil tampilan beserta pembahasan dari setiap objek yang di deteksi.

Person



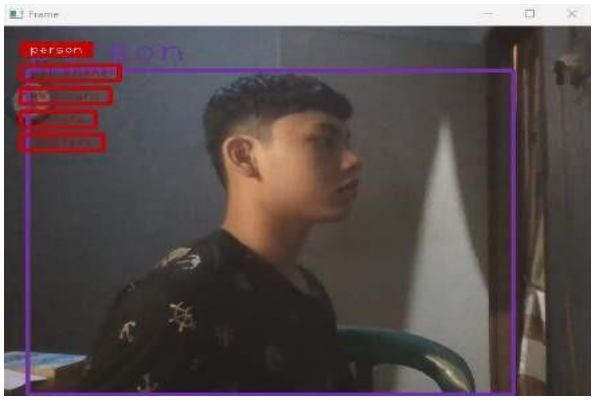
Gambar 17. Person jarak dekat

Gambar 18. Person menghadap kanan



Gambar 19. Person menoleh keatas

Gambar 20. Person jarak jauh



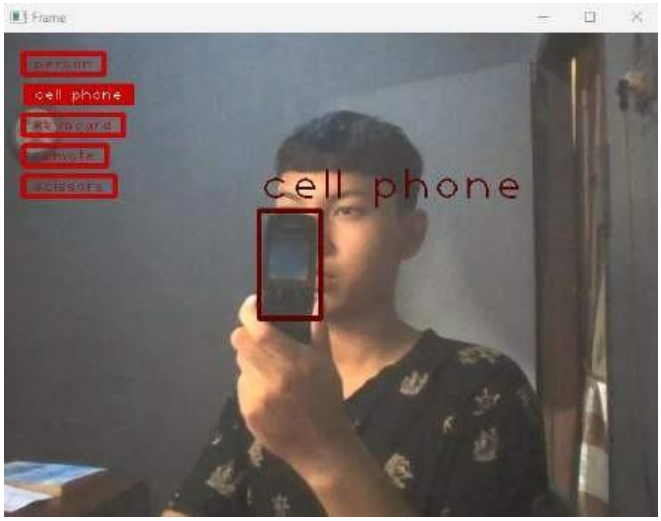
Gambar 21. Person menghadap ke kiri

Untuk mendeteksi person user perlu mengklik button person. Pada tampilan person, program berjalan dengan baik dari sudut manapun. Bahkan program mampu mendeteksi banyak orang sekaligus.

Cell Phone

Gambar 22. Handphone jarak dekat

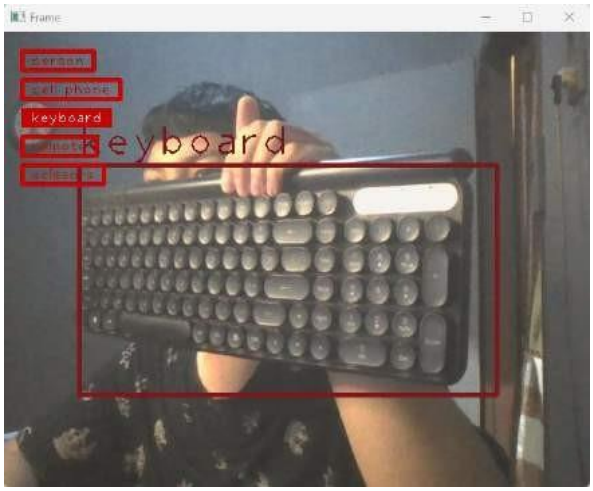
Gambar 23. Handphone jarak jauh



Gambar 24. Cell phone jarak dekat

Untuk mulai melakukan deteksi cell phone user perlu mengklik button cell phone. Hasil dari menjalankan program cell phone adalah, program dapat mendeteksi berbagai jenis handphone namun hanya bisa mendeteksi bagian belakang dan depan handphone. Program tidak dapat mendeteksi bagian samping handphone dan juga program tidak mampu untuk mendeteksi handphone berukuran kecil dari jarak jauh.

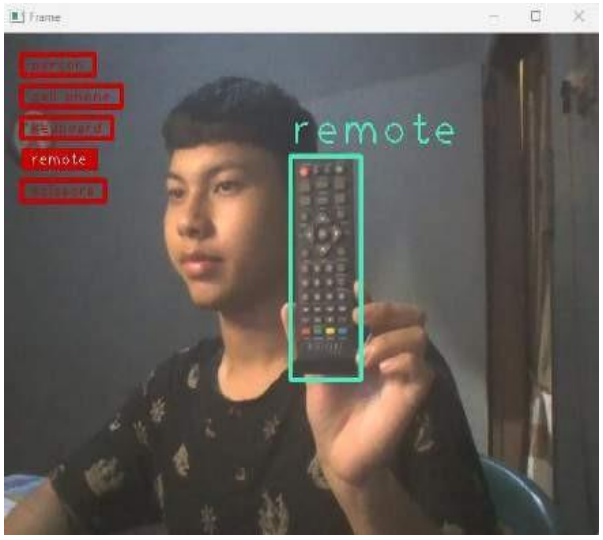
Keyboard



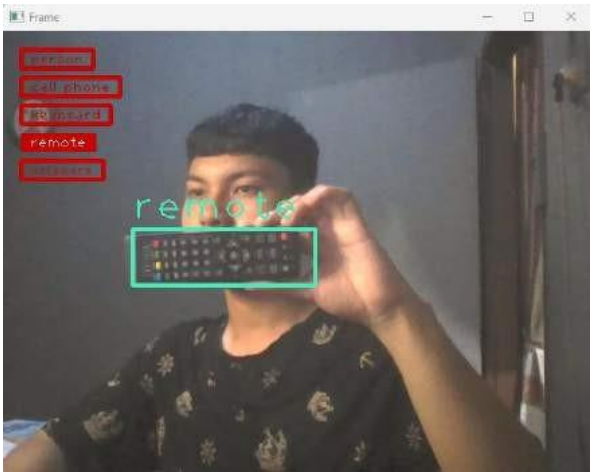
Gambar 25. Deteksi Keyboard

Untuk mulai melakukan deteksi keyboard user perlu mengklik button keyboard. Hasil dari deteksi keyboard hanya bisa mendeteksi keyboard dari jarak dekat dan sedikit miring, keyboard masih sering tidak terdeteksi dari beberapa sudut pandang dan tidak bisa mendeteksi dari jarak jauh.

Remote



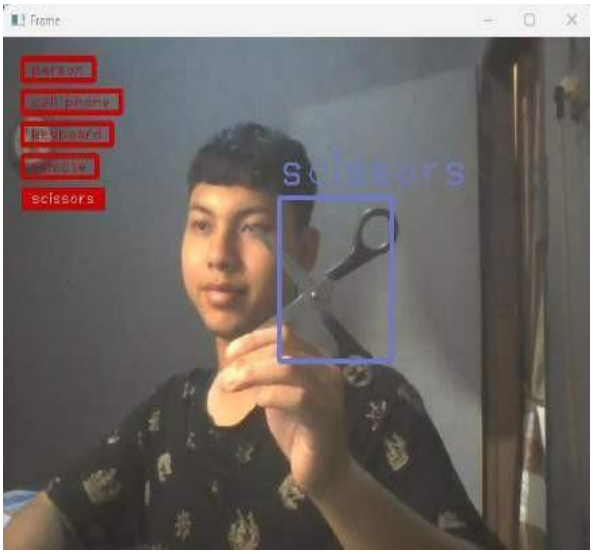
Gambar 26. Remote lurus



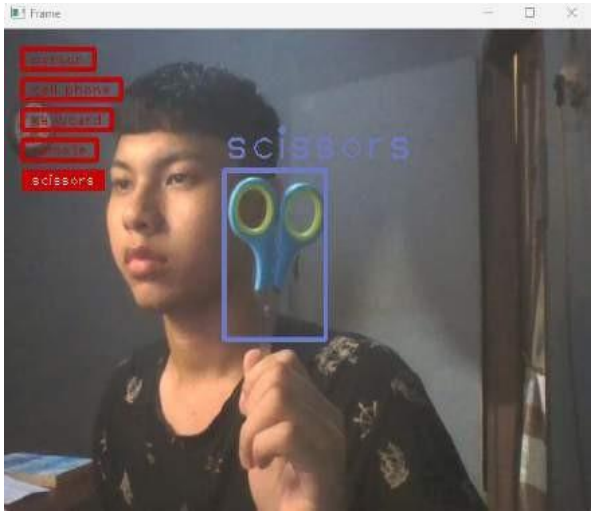
Gambar 27. Remote miring

Untuk mulai melakukan deteksi remote user perlu mengklik button remote. Hasil dari program mendeteksi remote adalah program hanya dapat mendeteksi remot dari bagian depan, program tidak mampu mendeteksi bagian belakang remote dan tidak bisa mendeteksi remote dari jarak jauh.

Scissor



Gambar 28. Gunting berwarna gelap



Gambar 29. Gunting berwarna terang

Untuk mulai melakukan deteksi scissor user perlu mengklik button scissor. Hasil dari deteksi scissor(gunting) adalah gunting yang berwarna terang lebih mudah terdeteksi dibandingkan gunting berwarna gelap. Program hanya mampu mendeteksi gunting dari jarak dekat.

Coding

Nama file: gui_button.py

```
import cv2
import numpy as np

class Buttons:
    def __init__(self):
        # Font
        self.font = cv2.FONT_HERSHEY_PLAIN
        self.text_scale = 1  # Adjust the text scale to make it smaller
        self.text_thick = 1  # Adjust the text thickness to make it smaller
        self.x_margin = 10  # Adjust the x margin to make the button smaller
        self.y_margin = 5   # Adjust the y margin to make the button smaller

        # Buttons
        self.buttons = {}
        self.button_index = 0
        self.buttons_area = []

        np.random.seed(0)
        self.colors = []
        self.generate_random_colors()

    def generate_random_colors(self):
        for i in range(91):
            random_c = np.random.randint(256, size=3)
            self.colors.append((int(random_c[0]), int(random_c[1]), int(random_c[2])))

    def add_button(self, text, x, y):
        # Get text size
        textsize = cv2.getTextSize(text, self.font, self.text_scale, self.text_thick)[0]
        right_x = x + (self.x_margin * 2) + textsize[0]
        bottom_y = y + (self.y_margin * 2) + textsize[1]

        self.buttons[self.button_index] = {"text": text, "position": [x, y, right_x, bottom_y], "active": False}
        self.button_index += 1

    def display_buttons(self, frame):
        for b_index, button_value in self.buttons.items():
            button_text = button_value["text"]
            (x, y, right_x, bottom_y) = button_value["position"]
            active = button_value["active"]
```

```
            if active:
                button_color = (0, 0, 200)
                text_color = (255, 255, 255)
                thickness = -1
            else:
                button_color = (0, 0, 200)
                text_color = (0, 0, 200)
                thickness = 3

            # Get text size
            cv2.rectangle(frame, (x, y), (right_x, bottom_y),
                          button_color, thickness)
            cv2.putText(frame, button_text, (x + self.x_margin, bottom_y - self.y_margin),
                        self.font, self.text_scale, text_color, self.text_thick)
        return frame

    def button_click(self, mouse_x, mouse_y):
        for b_index, button_value in self.buttons.items():
            (x, y, right_x, bottom_y) = button_value["position"]
            active = button_value["active"]
            area = [(x, y), (right_x, y), (right_x, bottom_y), (x, bottom_y)]

            inside = cv2.pointPolygonTest(np.array(area, np.int32), (int(mouse_x), int(mouse_y)), False)
            if inside > 0:
                print("IS Ac", active)
                new_status = False if active is True else True
                self.buttons[b_index]["active"] = new_status

    def active_buttons_list(self):
        active_list = []
        for b_index, button_value in self.buttons.items():
            active = button_value["active"]
            text = button_value["text"]
            if active:
                active_list.append(str(text).lower())

        return active_list
```

Nama File: main.py

```
import cv2
from gui_buttons import Buttons

# Constants
MODEL_WEIGHTS = "dnn_model/yolov4-tiny.weights"
MODEL_CONFIG = "dnn_model/yolov4-tiny.cfg"
CLASSES_FILE = "dnn_model/classes.txt"
CONFIDENCE_THRESHOLD = 0.3
NMS_THRESHOLD = 0.4

# Initialize Buttons
button = Buttons()
button.add_button("person", 20, 20)
button.add_button("cell phone", 20, 50)
button.add_button("keyboard", 20, 80)
button.add_button("remote", 20, 110)
button.add_button("scissors", 20, 140)

colors = button.colors

# Opencv DNN
net = cv2.dnn.readNet(MODEL_WEIGHTS, MODEL_CONFIG)
model = cv2.dnn_DetectionModel(net)
model.setInputParams(size=(320, 320), scale=1/255)

# Load class lists
try:
    with open(CLASSES_FILE, "r") as file_object:
        classes = [class_name.strip() for class_name in file_object.readlines()]
        print("Objects list:", classes)
except FileNotFoundError:
    print("Error: Classes file not found.")
    exit()
```

```
# Initialize camera
cap = cv2.VideoCapture(0)

# Check if the camera is opened successfully
if not cap.isOpened():
    print("Error: Unable to open the camera.")
    exit()

def click_button(event, x, y, flags, params):
    global button
    if event == cv2.EVENT_LBUTTONDOWN:
        button.button_click(x, y)

cv2.namedWindow("Frame")
cv2.setMouseCallback("Frame", click_button)

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

    if not ret:
        print("Error: Failed to read frame from the camera.")
        break

    # Object Detection
    (class_ids, scores, bboxes) = model.detect(frame, confThreshold=CONFIDENCE_THRESHOLD, nmsThreshold=NMS_THRESHOLD)
    for i, (class_id, score, bbox) in enumerate(zip(class_ids, scores, bboxes)):
        (x, y, w, h) = bbox
        class_name = classes[class_id]
        color = colors[class_id]

        if class_name in button.active_buttons_list():
            cv2.putText(frame, class_name, (x, y - 10), cv2.FONT_HERSHEY_PLAIN, 3, color, 2)
            cv2.rectangle(frame, (x, y), (x + w, y + h), color, 3)

    # Display buttons
    button.display_buttons(frame)

    cv2.imshow("Frame", frame)
    key = cv2.waitKey(1)
    if key == 27:  # 27 is the ASCII code for Esc key
        break

cap.release()
cv2.destroyAllWindows()
```

Nama File: setup.py

```
import sys
from cx_Freeze import setup, Executable


setup(name="Simple Object Detection Software",
      version="0.1",
      description="This software detects objects in realtime",
      executables=[Executable("main.py")]
      )
```

KESIMPULAN

Dari hasil dan pembahasan penelitian ini, dapat disimpulkan bahwa integrasi model deteksi objek YOLOv4-tiny dalam sistem pendeteksian objek real-time dengan antarmuka grafis pengguna (GUI) telah berhasil memberikan solusi yang efisien dan adaptif. YOLOv4-tiny mampu memberikan kecepatan deteksi yang tinggi tanpa mengorbankan akurasi, membuatnya cocok untuk aplikasi real-time. Antarmuka pengguna yang responsif dengan tombol- tombol memberikan kontrol yang lebih besar kepada pengguna, memungkinkan mereka mengatur jenis objek yang ingin dideteksi sesuai kebutuhan.

Keberhasilan ini membuka potensi pengembangan lebih lanjut dalam dua aspek utama. Pertama, eksplorasi lebih lanjut terhadap parameter YOLOv4-tiny dan optimisasi performa dapat meningkatkan kinerja deteksi objek. Kedua, pengembangan antarmuka pengguna dapat melibatkan penambahan fitur tambahan atau penyempurnaan desain untuk meningkatkan pengalaman pengguna. Sebagai kesimpulan,

penelitian ini memberikan kontribusi terhadap pengembangan sistem pendeteksian objek real-time yang dapat diakses dengan mudah dan adaptif. Dengan fokus pada kecepatan, akurasi, dan interaktivitas, penelitian ini membuka arah baru untuk pengembangan solusi pendeteksian objek yang lebih canggih di masa depan.

DAFTAR REFERENSI

[1] Bradski, G. (2000). The OpenCV Library. Dr. Dobb's Journal of Software Tools, 120-126.
[2] Liu, A., et al. (2020). YOLOv4-tiny: A Fast and Efficient Object Detection Model for Real-Time Applications. International Journal of Computer Vision, 1-15.
[3] Redmon, J., et al. (2018). YOLOv3: An Incremental Improvement. arXiv preprint.
[4] VanderPlas, J. (2016). *Python Data Science Handbook: Essential Tools for Working with Data.* O'Reilly Media.