Lampiran

Lampiran 1 Koding Notebook Google Colab Algortima YOLOv8

```
!nvidia-smi

import os
HOME = os.getcwd()
print(HOME)

# Pip install method (recommended)
!pip install ultralytics==8.2.103 -q
from IPython import display
display.clear_output()
# prevent ultralytics from tracking your activity
!yolo settings sync=False
import ultralytics
ultralytics.checks()

from ultralytics import YOLO
from IPython.display import display, Image

!mkdir -p {HOME}/datasets
%cd {HOME}/datasets
!pip install roboflow==1.1.48 --quiet
import roboflow
roboflow.login()
rf = roboflow.Roboflow()
project = rf.workspace("model-examples").project("football-players-
obj-detection")
dataset = project.version(2).download("yolov8")
%cd {HOME}

!yolo task=detect mode=train model=yolov8s.pt
data={dataset.location}/data.yaml epochs=100 imgsz=800 plots=True

%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/confusion_matrix.png',
width=600)

%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/results.png', width=600)

%cd {HOME}
Image(filename=f'{HOME}/runs/detect/train/val_batch0_pred.jpg',
width=600)
```

```
%cd {HOME}
!yolo task=detect mode=val
model={HOME}/runs/detect/train/weights/best.pt
data={dataset.location}/data.yaml

%cd {HOME}
!yolo task=detect mode=predict
model={HOME}/runs/detect/train/weights/best.pt conf=0.25
source={dataset.location}/test/images save=True

project.version(dataset.version).deploy(model_type="yolov8",
model_path=f"{HOME}/runs/detect/train/")
```

Lampiran 2 Koding Notebook Google Colab Algortima YOLOv9

```
!nvidia-smi

import os
HOME = os.getcwd()
print(HOME)

!git clone https://github.com/SkalskiP/yolov9.git
%cd yolov9
!pip install -r requirements.txt -q

!pip install -q roboflow

!wget -P {HOME}/weights -q
https://github.com/WongKinYiu/yolov9/releases/download/v0.1/yolov9-
c.pt
!wget -P {HOME}/weights -q
https://github.com/WongKinYiu/yolov9/releases/download/v0.1/yolov9-
e.pt
!wget -P {HOME}/weights -q
https://github.com/WongKinYiu/yolov9/releases/download/v0.1/gelan-
c.pt
!wget -P {HOME}/weights -q
https://github.com/WongKinYiu/yolov9/releases/download/v0.1/gelan-
e.pt

!ls -la {HOME}/weights

%cd {HOME}/yolov9

import roboflow
```

```
roboflow.login()
rf = roboflow.Roboflow()
project = rf.workspace("roboflow-jvuqo").project("football-players-
detection-3zvbc")
version = project.version(8)
dataset = version.download("yolov9")

%cd {HOME}/yolov9
!python train.py \
--batch 16 --epochs 100 --img 640 --device 0 --min-items 0 --close-
mosaic 15 \
--data {dataset.location}/data.yaml \
--weights {HOME}/weights/gelan-c.pt \
--cfg models/detect/gelan-c.yaml \
--hyp hyp.scratch-high.yaml

!ls {HOME}/yolov9/runs/train/exp/

from IPython.display import Image
Image(filename=f"{HOME}/yolov9/runs/train/exp/results.png",
width=1000)

from IPython.display import Image
Image(filename=f"{HOME}/yolov9/runs/train/exp/confusion_matrix.png",
width=1000)

from IPython.display import Image
Image(filename=f"{HOME}/yolov9/runs/train/exp/val_batch0_pred.jpg",
width=1000)

%cd {HOME}/yolov9
!python val.py \
--img 640 --batch 32 --conf 0.001 --iou 0.7 --device 0 \
--data {dataset.location}/data.yaml \
--weights {HOME}/yolov9/runs/train/exp/weights/best.pt

!python detect.py \
--img 1280 --conf 0.1 --device 0 \
--weights {HOME}/yolov9/runs/train/exp/weights/best.pt \
--source {dataset.location}/test/images

import glob
from IPython.display import Image, display
for image_path in
glob.glob(f'{HOME}/yolov9/runs/detect/exp3/*.jpg')[:2]:
      display(Image(filename=image_path, width=600))
```

```
version.deploy(model_type="yolov9",
model_path=f"{HOME}/yolov9/runs/train/exp")
```

Lampiran 3 Koding Notebook Google Colab Algortima YOLOv11

```
!nvidia-smi

import os
HOME = os.getcwd()
print(HOME)

%pip install "ultralytics<=8.3.40" supervision roboflow
# prevent ultralytics from tracking your activity
!yolo settings sync=False
import ultralytics
ultralytics.checks()

!mkdir {HOME}/datasets
%cd {HOME}/datasets
from google.colab import userdata
from roboflow import Roboflow
ROBOFLOW_API_KEY = userdata.get('ROBOFLOW_API_KEY')
rf = Roboflow(api_key=ROBOFLOW_API_KEY)
workspace = rf.workspace("liangdianzhong")
project = workspace.project("-qvdww")
version = project.version(3)
dataset = version.download("yolov11")

%cd {HOME}
!yolo task=detect mode=train model=yolo11s.pt
data={dataset.location}/data.yaml epochs=100 imgsz=640 plots=True

!ls {HOME}/runs/detect/train/

from IPython.display import Image as IPyImage
IPyImage(filename=f'{HOME}/runs/detect/train/confusion_matrix.png',
width=600)

from IPython.display import Image as IPyImage
IPyImage(filename=f'{HOME}/runs/detect/train/results.png',
width=600)

from IPython.display import Image as IPyImage
IPyImage(filename=f'{HOME}/runs/detect/train/val_batch0_pred.jpg',
width=600)
```

```
!yolo task=detect mode=val
model={HOME}/runs/detect/train/weights/best.pt
data={dataset.location}/data.yaml

!yolo task=detect mode=predict
model={HOME}/runs/detect/train/weights/best.pt conf=0.25
source={dataset.location}/test/images save=True

project.version(dataset.version).deploy(model_type="yolov11",
model_path=f"{HOME}/runs/detect/train/")
```

Lampiran 4 Koding Deteksi Objek

```
from ultralytics import YOLO
import cv2
import tkinter as tk
from tkinter import filedialog, messagebox, simpledialog
from PIL import Image, ImageTk
import threading
import os
import time

class ObjectDetectorApp:
    def __init__(self, root):
        self.root = root
        self.root.title("YOLO Object Detector")
        self.root.geometry("1200x700")

        # Dark Theme Colors
        self.bg_color = "#121212"
        self.frame_color = "#1E1E1E"
        self.text_color = "#FFFFFF"
        self.button_color = "#333333"
        self.active_button_color = "#555555"

        # Model Paths
        self.model_paths = {
            "YOLOv8": "file/yolo8/best.pt",
            "YOLOv9": "file/yolo9/best.pt",
            "YOLOv11": "file/yolo11/best.pt",
            "Damage Detection": "file/damage_detection/best.pt"  #
Tambahkan model deteksi kerusakan
        }
        self.current_model = None
        self.model = None
```

```
        # IP Camera variables
        self.ip_camera_url = "http://192.168.45.218:8080/video"
        self.cap = None
        self.camera_thread = None
        self.camera_running = False
        self.zoom_level = 1.0
        self.zoom_step = 0.1
        self.skip_frames = 1
        self.frame_counter = 0

        self.root.configure(bg=self.bg_color)
        self.root.grid_columnconfigure(0, weight=1)
        self.root.grid_rowconfigure(1, weight=1)

        # Create menu bar
        self.create_menu()

        # === Top Button Frame ===
        self.button_frame = tk.Frame(root, bg=self.bg_color)
        self.button_frame.grid(row=0, column=0, pady=10,
sticky="ew")

        self.model_var = tk.StringVar(root)
        self.model_var.set("Select Model")
        self.model_menu = tk.OptionMenu(
            self.button_frame, self.model_var,
*self.model_paths.keys(), command=self.select_model)
        self._style_widget(self.model_menu)
        self.model_menu.pack(side=tk.LEFT, padx=10)

        self.select_image_button = tk.Button(
            self.button_frame, text="Select Image",
command=self.select_image)
        self._style_widget(self.select_image_button)
        self.select_image_button.pack(side=tk.LEFT, padx=10)

        self.select_video_button = tk.Button(
            self.button_frame, text="Select Video",
command=self.select_video)
        self._style_widget(self.select_video_button)
        self.select_video_button.pack(side=tk.LEFT, padx=10)

        self.ip_camera_button = tk.Button(
            self.button_frame, text="IP Camera",
command=self.toggle_ip_camera)
        self._style_widget(self.ip_camera_button)
        self.ip_camera_button.pack(side=tk.LEFT, padx=10)
```

```
        self.detect_button = tk.Button(
            self.button_frame, text="Detect Objects",
command=self.detect_objects, state=tk.DISABLED)
        self._style_widget(self.detect_button)
        self.detect_button.pack(side=tk.LEFT, padx=10)

        # === Image Display Area ===
        self.image_frame = tk.Frame(root, bg=self.bg_color)
        self.image_frame.grid(row=1, column=0, padx=10, pady=10,
sticky="nsew")
        self.image_frame.grid_columnconfigure(0, weight=1)
        self.image_frame.grid_columnconfigure(1, weight=1)
        self.image_frame.grid_rowconfigure(0, weight=1)

        self.original_frame = self._create_label_frame("Original",
0)
        self.original_label = tk.Label(self.original_frame,
bg=self.frame_color)
        self.original_label.pack(expand=True, fill="both", padx=5,
pady=5)

        self.result_frame = self._create_label_frame("Detection
Result", 1)
        self.result_label = tk.Label(self.result_frame,
bg=self.frame_color)
        self.result_label.pack(expand=True, fill="both", padx=5,
pady=5)

        # === State Variables ===
        self.image_path = None
        self.video_path = None
        self.original_image = None
        self.result_image = None
        self.video_thread = None
        self.video_running = False

        # Bind keyboard shortcuts
        self.root.bind('<Key>', self.handle_keypress)

    def create_menu(self):
        menubar = tk.Menu(self.root, bg=self.button_color,
fg=self.text_color)

        # File menu
        file_menu = tk.Menu(menubar, tearoff=0,
bg=self.button_color, fg=self.text_color)
```

```
        file_menu.add_command(label="Open Image",
command=self.select_image)
        file_menu.add_command(label="Open Video",
command=self.select_video)
        file_menu.add_separator()
        file_menu.add_command(label="Exit", command=self.on_closing)
        menubar.add_cascade(label="File", menu=file_menu)

        # Model menu
        model_menu = tk.Menu(menubar, tearoff=0,
bg=self.button_color, fg=self.text_color)
        for model_name in self.model_paths.keys():
            model_menu.add_command(label=model_name,
                                   command=lambda m=model_name:
self.select_model(m))
        menubar.add_cascade(label="Models", menu=model_menu)

        # View menu
        view_menu = tk.Menu(menubar, tearoff=0,
bg=self.button_color, fg=self.text_color)
        view_menu.add_command(label="Zoom In (+)", command=lambda:
self.adjust_zoom(1))
        view_menu.add_command(label="Zoom Out (-)", command=lambda:
self.adjust_zoom(-1))
        view_menu.add_command(label="Reset Zoom (0)",
command=lambda: self.adjust_zoom(0))
        menubar.add_cascade(label="View", menu=view_menu)

        # Help menu
        help_menu = tk.Menu(menubar, tearoff=0,
bg=self.button_color, fg=self.text_color)
        help_menu.add_command(label="About",
command=self.show_about)
        menubar.add_cascade(label="Help", menu=help_menu)

        self.root.config(menu=menubar)

    def _style_widget(self, widget):
        widget.config(
            width=15, height=2,
            bg=self.button_color, fg=self.text_color,
            activebackground=self.active_button_color,
            activeforeground=self.text_color,
            relief=tk.FLAT,
            font=('Helvetica', 10, 'bold'),
            highlightthickness=0
        )
```

```
    def _create_label_frame(self, title, col):
        frame = tk.LabelFrame(
            self.image_frame, text=title,
            bg=self.frame_color, fg=self.text_color,
            font=('Helvetica', 10, 'bold'),
            relief=tk.FLAT
        )
        frame.grid(row=0, column=col, padx=5, pady=5, sticky="nsew")
        return frame

    def select_model(self, selected_model):
        try:
            self.current_model = selected_model
            model_path = self.model_paths[selected_model]
            self.model = YOLO(model_path)
            self.model.fuse()  # Fuse model for faster inference
            messagebox.showinfo("Model Loaded", f"Successfully
loaded {selected_model}")
            self.detect_button.config(state=tk.NORMAL)

            # Update tampilan berdasarkan model yang dipilih
            if "Damage" in selected_model:
                self.detect_button.config(text="Detect Damage")
            else:
                self.detect_button.config(text="Detect Objects")
        except Exception as e:
            messagebox.showerror("Error", f"Failed to load model:
{e}")
            self.model = None
            self.detect_button.config(state=tk.DISABLED)

    def select_image(self):
        self.image_path =
filedialog.askopenfilename(filetypes=[("Image Files",
"*.jpg;*.jpeg;*.png")])
        if self.image_path:
            try:
                self.video_path = None  # Reset video
                self.stop_camera()  # Stop camera if running
                self.original_image = Image.open(self.image_path)
                self.display_image(self.original_image,
self.original_label)
                self.detect_button.config(state=tk.NORMAL if
self.model else tk.DISABLED)
            except Exception as e:
```

```
                messagebox.showerror("Error", f"Failed to open
image: {e}")

    def select_video(self):
        self.video_path =
filedialog.askopenfilename(filetypes=[("Video Files",
"*.mp4;*.avi;*.mov")])
        if self.video_path:
            self.image_path = None  # Reset image
            self.stop_camera()  # Stop camera if running
            self.original_label.config(image='', text="Video
Loaded")
            self.detect_button.config(state=tk.NORMAL if self.model
else tk.DISABLED)

    def toggle_ip_camera(self):
        if self.camera_running:
            self.stop_camera()
        else:
            self.start_camera()

    def start_camera(self):
        if not self.model:
            messagebox.showwarning("Warning", "Please select a model
first")
            return

        self.image_path = None  # Reset image
        self.video_path = None  # Reset video
        self.camera_running = True
        self.ip_camera_button.config(text="Stop Camera")
        self.original_label.config(image='', text="Camera
Starting...")

        # Initialize camera capture
        self.cap = cv2.VideoCapture(self.ip_camera_url)
        self.cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)
        self.cap.set(cv2.CAP_PROP_FRAME_WIDTH, 640)
        self.cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 480)
        self.cap.set(cv2.CAP_PROP_FPS, 30)

        if not self.cap.isOpened():
            messagebox.showerror("Error", "Failed to connect to IP
camera")
            self.stop_camera()
            return
```

```
        self.camera_thread =
threading.Thread(target=self._run_camera_detection)
        self.camera_thread.start()

    def stop_camera(self):
        if self.camera_running:
            self.camera_running = False
            self.ip_camera_button.config(text="IP Camera")
            if self.camera_thread and self.camera_thread.is_alive():
                self.camera_thread.join()
            if self.cap:
                self.cap.release()
                self.cap = None
            self.original_label.config(image='', text="Camera
Stopped")
            self.result_label.config(image='', text="Detection
Result")

    def _run_camera_detection(self):
        prev_time = 0
        fps = 0

        while self.camera_running and self.cap.isOpened():
            ret, frame = self.cap.read()
            if not ret:
                messagebox.showerror("Error", "Failed to read from
camera")
                self.stop_camera()
                break

            # Calculate FPS
            current_time = time.time()
            fps = 1 / (current_time - prev_time)
            prev_time = current_time

            # Skip frames for better performance
            self.frame_counter += 1
            if self.frame_counter % (self.skip_frames + 1) != 0:
                continue

            # Resize frame for faster processing
            frame = cv2.resize(frame, (640, 480))

            # Convert to PIL Image for display in original frame
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            original_pil = Image.fromarray(frame_rgb)
            self.display_image(original_pil, self.original_label)
```

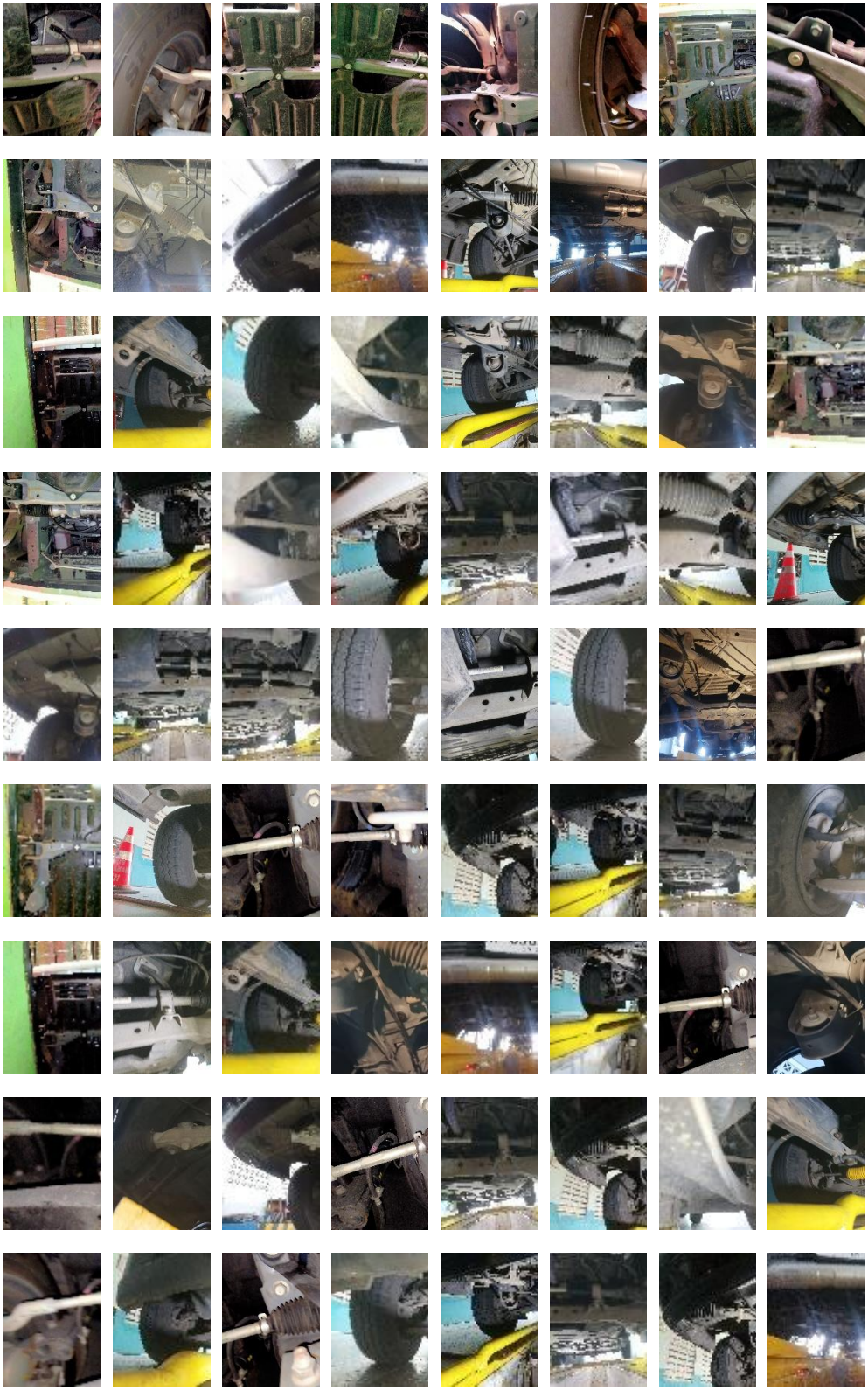
```
            # Perform detection
            results = self.model(frame, imgsz=320, verbose=False)
            annotated = results[0].plot(line_width=1)
            annotated_rgb = cv2.cvtColor(annotated,
cv2.COLOR_BGR2RGB)

            # Apply zoom
            if self.zoom_level != 1.0:
                h, w = annotated_rgb.shape[:2]
                center_x, center_y = w // 2, h // 2
                new_w, new_h = int(w / self.zoom_level), int(h /
self.zoom_level)
                x1 = max(0, center_x - new_w // 2)
                y1 = max(0, center_y - new_h // 2)
                x2 = min(w, center_x + new_w // 2)
                y2 = min(h, center_y + new_h // 2)
                zoomed_frame = annotated_rgb[y1:y2, x1:x2]
                annotated_rgb = cv2.resize(zoomed_frame, (w, h))

            # Add FPS and zoom info
            cv2.putText(annotated_rgb, f"FPS: {int(fps)}", (10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0),
2)
            cv2.putText(annotated_rgb, f"Zoom:
{self.zoom_level:.1f}x", (10, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0),
2)

            # Tambahkan informasi model yang digunakan
            cv2.putText(annotated_rgb, f"Model:
{self.current_model}", (10, 90),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 255, 0),
2)

            # Display result
            result_pil = Image.fromarray(annotated_rgb)
            self.display_image(result_pil, self.result_label)

    def detect_objects(self):
        if self.image_path:
            self._detect_image()
        elif self.video_path:
            self._detect_video()
        else:
            messagebox.showwarning("Warning", "No file selected.")
```

```
    def _detect_image(self):
        try:
            results = self.model.predict(self.image_path)
            annotated = results[0].plot()
            annotated_rgb = cv2.cvtColor(annotated,
cv2.COLOR_BGR2RGB)
            self.result_image = Image.fromarray(annotated_rgb)
            self.display_image(self.result_image, self.result_label)

            output_path = self.image_path.replace('.',
f'_{self.current_model.lower()}_detected.')
            cv2.imwrite(output_path, annotated)
            messagebox.showinfo("Success", f"Detection result saved:
{output_path}")

            # Jika model deteksi kerusakan, tampilkan hasil tambahan
            if "Damage" in self.current_model:
                self._show_damage_analysis(results)
        except Exception as e:
            messagebox.showerror("Error", f"Detection failed: {e}")

    def _show_damage_analysis(self, results):
        """Fungsi khusus untuk menampilkan analisis kerusakan"""
        damage_count = {}
        for box in results[0].boxes:
            cls = int(box.cls)
            name = results[0].names[cls]
            damage_count[name] = damage_count.get(name, 0) + 1

        analysis_text = "Damage Analysis:\n"
        for damage_type, count in damage_count.items():
            analysis_text += f"- {damage_type}: {count} detected\n"

        messagebox.showinfo("Damage Analysis", analysis_text)

    def _detect_video(self):
        if self.video_thread and self.video_thread.is_alive():
            self.video_running = False
            self.video_thread.join()

        self.video_running = True
        self.video_thread =
threading.Thread(target=self._run_video_detection)
        self.video_thread.start()

    def _run_video_detection(self):
        cap = cv2.VideoCapture(self.video_path)
```

```
        output_path = self.video_path.replace('.',
f'_{self.current_model.lower()}_detected.')
        fourcc = cv2.VideoWriter_fourcc(*'mp4v')
        out = None

        while cap.isOpened() and self.video_running:
            ret, frame = cap.read()
            if not ret:
                break

            # Display original frame
            frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
            original_pil = Image.fromarray(frame_rgb)
            self.display_image(original_pil, self.original_label)

            # Perform detection
            results = self.model.predict(source=frame,
verbose=False)
            annotated = results[0].plot()
            annotated_rgb = cv2.cvtColor(annotated,
cv2.COLOR_BGR2RGB)
            result_pil = Image.fromarray(annotated_rgb)
            self.display_image(result_pil, self.result_label)

            if out is None:
                h, w = annotated.shape[:2]
                out = cv2.VideoWriter(output_path, fourcc, 20.0, (w,
h))
            out.write(annotated)

            if not self.video_running:
                break

        cap.release()
        if out:
            out.release()
        messagebox.showinfo("Done", f"Video saved: {output_path}")

    def display_image(self, image, label):
        max_size = (550, 550)
        image.thumbnail(max_size, Image.LANCZOS)
        tk_image = ImageTk.PhotoImage(image)
        label.config(image=tk_image)
        label.image = tk_image

    def adjust_zoom(self, direction):
        if direction == 1:  # Zoom in
```

```
            self.zoom_level += self.zoom_step
            self.zoom_level = min(self.zoom_level, 3.0)
        elif direction == -1:  # Zoom out
            self.zoom_level -= self.zoom_step
            self.zoom_level = max(self.zoom_level, 1.0)
        else:  # Reset zoom
            self.zoom_level = 1.0

    def handle_keypress(self, event):
        if not self.camera_running:
            return

        # Zoom in with '+' or '='
        if event.char in ('+', '='):
            self.adjust_zoom(1)
        # Zoom out with '-'
        elif event.char == '-':
            self.adjust_zoom(-1)
        # Reset zoom with '0'
        elif event.char == '0':
            self.adjust_zoom(0)
        # Stop camera with 'q' or ESC
        elif event.char == 'q' or event.keysym == 'Escape':
            self.stop_camera()

    def show_about(self):
        about_text = """YOLO Object Detector
Versi 1.0
Dikembangkan dengan Python, OpenCV, dan Ultralytics YOLO

© 2023 Aplikasi Deteksi Objek"""
        messagebox.showinfo("Tentang", about_text)

    def on_closing(self):
        self.stop_camera()
        if self.video_thread and self.video_thread.is_alive():
            self.video_running = False
            self.video_thread.join()
        self.root.destroy()

if __name__ == "__main__":
    root = tk.Tk()
    app = ObjectDetectorApp(root)
    root.protocol("WM_DELETE_WINDOW", app.on_closing)
    root.mainloop()
```

Lampiran 5 Dataset Gambar sistem kemudi



Lampiran 6 Biodata Penulis

BIODATA PENULIS

Nama : Jihan Luis

Notar : 22031050

Tempat / Tanggal Lahir : Banyumas, 29 Mei 2001

Jenis Kelamin : Laki – Laki

Status : Belum Kawin

Alamat Asal : Cindaga RT 01 / RW 01, Kecamatan Kebasen, Kabupaten Banyumas, Jawa Tengah

Telpon : 089507588226

Email : luisakungem@gmail.com

Motto : "Be smart and rich so you won't be oppressed"