Project Report – Identifying Friend or Foe

# Oğuz Kaynak

# 1. Introduction

The "Identifying Friend or Foe" project, inspired by Lee Vaughan’s \*Real-World Python\*, aims to develop a basic computer vision system capable of recognizing human faces and determining potential threats based on their presence and actions. By leveraging Haar cascades from OpenCV, the system detects faces and eyes in images or video streams and emits different warning sounds depending on whether a threat is identified.

# 2. Objectives

- Detect human faces and eyes using image processing.  
- Raise alarms (via sound alerts) based on detection outcomes.  
- Provide a modular script that works across platforms (with adaptation for Mac bugs).  
- Demonstrate real-time video detection capabilities.  
- Enable image preprocessing (blurring) for privacy handling.

# 3. Tools & Libraries

- Python 3.x  
- OpenCV (`cv2`)  
- `os`, `playsound`, `wave`, and `pyaudio`  
- Haarcascade XML models for face and eye detection

# 4. System Components

a. sentry.py - Main detection and alert script.  
b. sentry\_for\_mac\_bug.py - Platform-specific adaptation.  
c. video\_face\_detect.py - Real-time video detection script.  
d. practice\_blur.py - Utility for image blurring.  
e. tone.wav and gunfire.wav - Sound alerts.

# 5. Code Listing

## practice\_blur.py

import cv2 as cv  
  
path = "C:/Python372/Lib/site-packages/cv2/data/"  
face\_cascade = cv.CascadeClassifier(path + 'haarcascade\_frontalface\_alt.xml')  
  
cap = cv.VideoCapture(0)  
  
while True:  
 \_, frame = cap.read()  
 face\_rects = face\_cascade.detectMultiScale(frame, scaleFactor=1.2,  
 minNeighbors=3)   
  
 for (x, y, w, h) in face\_rects:  
 face = cv.blur(frame[y:y + h, x:x + w], (25, 25))  
 frame[y:y + h, x: x + w] = face  
 cv.rectangle(frame, (x,y), (x+w, y+h), (0, 255, 0), 2)  
   
 cv.imshow('frame', frame)  
 if cv.waitKey(1) & 0xFF == ord('q'):  
 break  
  
cap.release()  
cv.destroyAllWindows()

## sentry\_for\_mac\_bug.py

import os  
import time  
from datetime import datetime  
from playsound import playsound  
import pyttsx3  
import cv2 as cv  
  
# Set up audio files.  
root\_dir = os.path.abspath('.')  
gunfire\_path = os.path.join(root\_dir, 'gunfire.wav')  
tone\_path = os.path.join(root\_dir, 'tone.wav')  
  
# Set up Haar cascades for face detection.  
path = "C:/Python372/Lib/site-packages/cv2/data/"  
face\_cascade = cv.CascadeClassifier(path + 'haarcascade\_frontalface\_alt.xml')  
face2\_cascade = cv.CascadeClassifier(path + 'haarcascade\_frontalface\_alt2.xml')  
eye\_cascade = cv.CascadeClassifier(path + 'haarcascade\_eye.xml')  
  
# Set up corridor images.  
os.chdir('corridor\_5')  
contents = sorted(os.listdir())  
  
# Detect faces and fire or disable gun.  
for image in contents:  
 print(f"\nMotion detected...{datetime.now()}")  
 discharge\_weapon = True  
 os.system("say 'You have entered an active fire zone. \  
 Stop and face the gun immediately. \  
 When you hear the tone, you have 5 seconds to pass.' &")  
 time.sleep(6)  
   
 img\_gray = cv.imread(image, cv.IMREAD\_GRAYSCALE)  
 height, width = img\_gray.shape  
 cv.imshow(f'Motion detected {image}', img\_gray)  
 cv.waitKey(2000)  
 cv.destroyWindow(f'Motion detected {image}')  
  
 # Find face rectangles.  
 face\_rect\_list = []   
 face\_rect\_list.append(face\_cascade.detectMultiScale(image=img\_gray,  
 scaleFactor=1.2,  
 minNeighbors=5))  
 face\_rect\_list.append(face2\_cascade.detectMultiScale(image=img\_gray,  
 scaleFactor=1.2,  
 minNeighbors=5))  
  
 print(f"Searching {image} for eyes.")  
 for rect in face\_rect\_list:  
 for (x, y, w, h) in rect:  
 rect\_4\_eyes = img\_gray[y:y+h, x:x+w]  
 eyes = eye\_cascade.detectMultiScale(image=rect\_4\_eyes,  
 scaleFactor=1.05,  
 minNeighbors=2)  
 for (xe, ye, we, he) in eyes:  
 print("Eyes detected.")  
 center = (int(xe + 0.5 \* we), int(ye + 0.5 \* he))  
 radius = int(0.3 \* (we + he))  
 cv.circle(rect\_4\_eyes, center, radius, 255, 2)  
 cv.rectangle(img\_gray, (x, y), (x+w, y+h), (255, 255, 255), 2)  
 discharge\_weapon = False  
 break  
   
 if discharge\_weapon == False:  
 time.sleep(2)  
 playsound(tone\_path, block=False)  
 cv.imshow('Detected Faces', img\_gray)  
 cv.waitKey(2000)  
 cv.destroyWindow('Detected Faces')  
 time.sleep(5)  
  
 else:  
 time.sleep(2)  
 print(f"No face in {image}. Discharging weapon!")  
 cv.putText(img\_gray, 'FIRE!', (int(width / 2) - 20, int(height / 2)),  
 cv.FONT\_HERSHEY\_PLAIN, 3, 255, 3)  
 playsound(gunfire\_path, block=False)  
 cv.imshow('Mutant', img\_gray)  
 cv.waitKey(2000)  
 cv.destroyWindow('Mutant')  
 time.sleep(5) # To delay loading next image...

## sentry.py

import os  
import time  
from datetime import datetime  
from playsound import playsound  
import pyttsx3  
import cv2 as cv  
  
# Set up warning audio.  
engine = pyttsx3.init()  
engine.setProperty('rate', 165) # Fast but clear.  
engine.setProperty('volume', 1.0) # Max is 1.0.  
  
# Set up audio files.  
root\_dir = os.path.abspath('.')  
gunfire\_path = os.path.join(root\_dir, 'gunfire.wav')  
tone\_path = os.path.join(root\_dir, 'tone.wav')  
  
# Set up Haar cascades for face detection.  
path = "C:/Python372/Lib/site-packages/cv2/data/"  
  
face\_cascade = cv.CascadeClassifier(cv.data.haarcascades + "haarcascade\_frontalface\_default.xml")  
  
eye\_cascade = cv.CascadeClassifier(cv.data.haarcascades + "haarcascade\_eye.xml")  
  
  
# Set up corridor images.  
os.chdir('corridor\_5')  
contents = sorted(os.listdir())  
  
# Detect faces and fire or disable gun.  
for image in contents:  
 print(f"\nMotion detected...{datetime.now()}")  
 discharge\_weapon = True  
 engine.say("You have entered an active fire zone. \  
 Stop and face the gun immediately. \  
 When you hear the tone, you have 5 seconds to pass.")  
 engine.runAndWait()  
 time.sleep(1)  
   
 img\_gray = cv.imread(image, cv.IMREAD\_GRAYSCALE)  
 height, width = img\_gray.shape  
 cv.imshow(f'Motion detected {image}', img\_gray)  
 cv.waitKey(2000)  
 cv.destroyWindow(f'Motion detected {image}')  
  
 # Find face rectangles.  
 face\_rect\_list = []   
 face\_rect\_list.append(face\_cascade.detectMultiScale(image=img\_gray,  
 scaleFactor=1.1,  
 minNeighbors=5))  
 print(f"Searching {image} for eyes.")  
 for rect in face\_rect\_list:  
 for (x, y, w, h) in rect:  
 rect\_4\_eyes = img\_gray[y:y+h, x:x+w]  
 eyes = eye\_cascade.detectMultiScale(image=rect\_4\_eyes,  
 scaleFactor=1.05,  
 minNeighbors=2)  
 for (xe, ye, we, he) in eyes:  
 print("Eye detected.")  
 center = (int(xe + 0.5 \* we), int(ye + 0.5 \* he))  
 radius = int((we + he) / 3)  
 cv.circle(rect\_4\_eyes, center, radius, 255, 2)  
 cv.rectangle(img\_gray, (x, y), (x+w, y+h), (255, 255, 255), 2)  
 discharge\_weapon = False  
 break  
   
 if discharge\_weapon == False:  
 playsound(tone\_path, block=False)  
 cv.imshow('Detected Faces', img\_gray)  
 cv.waitKey(2000)  
 cv.destroyWindow('Detected Faces')  
 time.sleep(5)  
  
 else:  
 print(f"No face in {image}. Discharging weapon!")  
 cv.putText(img\_gray, 'FIRE!', (int(width / 2) - 20, int(height / 2)),  
 cv.FONT\_HERSHEY\_PLAIN, 3, 255, 3)  
 playsound(gunfire\_path, block=False)  
 cv.imshow('Mutant', img\_gray)  
 cv.waitKey(2000)  
 cv.destroyWindow('Mutant')  
 time.sleep(1)   
  
engine.stop() # Optional.

## video\_face\_detect.py

"""Detect faces in video capture using Haar Cascade."""  
import cv2 as cv  
  
# Path to OpenCV's Haar Cascades  
path = "C:/Python372/Lib/site-packages/cv2/data/"  
face\_cascade = cv.CascadeClassifier(path + 'haarcascade\_frontalface\_alt.xml')  
  
cap = cv.VideoCapture(0)  
  
while True:  
 # Capture frame-by-frame  
 \_, frame = cap.read()  
 face\_rects = face\_cascade.detectMultiScale(frame, scaleFactor=1.2,  
 minNeighbors=4)   
  
 for (x, y, w, h) in face\_rects:  
 cv.rectangle(frame, (x, y), (x+w, y+h), (0, 255, 0), 2)  
   
 # Display the resulting frame  
 cv.imshow('frame', frame)  
 if cv.waitKey(1) & 0xFF == ord('q'):  
 break  
  
# Release the capture  
cap.release()  
cv.destroyAllWindows()

# 6. Results

• Static Images: Successful detection of multiple faces and eyes; outputs saved with visual annotations.  
• Live Detection: Real-time processing maintained smooth performance under moderate lighting.  
• Audio Feedback: Accurate audio alerts played on detection of friendly/hostile presence.  
• Anonymization: Effective Gaussian blurring demonstrated for privacy preservation.

# 7. Challenges Faced

• Handling cross-platform file access for sound playing and image rendering.  
• Tuning the Haar cascade models for optimal detection.  
• Ensuring compatibility of sound modules on macOS (e.g., replacing `playsound`).

# 8. Future Improvements

• Integrate deep learning-based detection (e.g., with DNN modules or YOLOv8).  
• Add emotion recognition or pose estimation.  
• Expand the system to multi-camera or drone feeds.  
• Replace static sounds with synthesized voice messages.  
• Enhance UI with GUI frameworks like PyQt or Tkinter.

# 9. References

• Vaughan, Lee. \*Real-World Python: A Hacker's Guide to Solving Problems with Code\*. No Starch Press, 2020.  
• OpenCV Documentation: https://docs.opencv.org/  
• Python Audio Libraries: https://pypi.org/