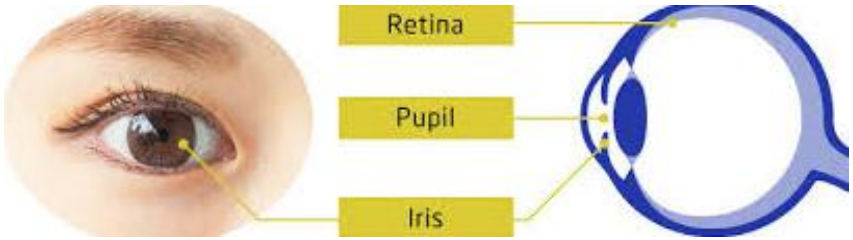
Iris Detection using Mediapipe and Python

Rajesh G.D.

September 26, 2024

What is Iris Detection?



- **Iris Detection** is a computer vision task aimed at detecting and tracking the iris of the human eye in real-time.
- The iris is the colored part of the eye surrounding the pupil. Its shape and position are crucial in biometric identification, gaze tracking, and medical diagnostics.
- Iris detection can be used in:
  - Eye-tracking systems
  - Augmented reality (AR) applications
  - User authentication (biometrics)
- Challenges include accurately tracking iris movements under various lighting and head pose conditions.

How does Mediapipe Work for Iris Detection?

- **Mediapipe** is a framework by Google for building multimodal, cross-platform machine learning pipelines.
- For iris detection, Mediapipe's **Face Mesh** uses 468 3D landmarks to map facial features, including eyes.
- Key Features of Mediapipe for Iris Detection:
  - Accurate real-time face and eye landmark detection.
  - Refinement of eye landmarks with an additional 5-point iris model.
  - Works on multiple platforms, including web, mobile, and desktop.
- Mediapipe is known for its efficiency and high performance with minimal computational resources.

Step 1: Import Necessary Libraries

```
import cv2
import mediapipe as mp
import numpy as np
import csv
import os
import time
```

- This step includes importing the necessary libraries:
  - `cv2`: For handling video input and image processing.
  - `mediapipe`: For face landmark detection.
  - `numpy`: For numerical operations and calculations.
  - `csv`: For handling CSV file operations.
  - `os`: For file and directory operations.
  - `time`: For handling timing operations.

Step 2: Remove Existing CSV File

```
csv_file = 'iris_pupil_data.csv'
if os.path.exists(csv_file):
os.remove(csv_file)
```

- Define the name of the CSV file to store data.
- Check if the CSV file already exists; if it does, remove it to start fresh.

Step 3: Initialize Mediapipe Face Mesh

```
mp_face_mesh = mp.solutions.face_mesh
```

- Initialize the Mediapipe Face Mesh solution which is used for detecting facial landmarks.

Step 4: Define Landmark Indices

```
LEFT_IRIS = [474, 475, 476, 477]
RIGHT_IRIS = [469, 470, 471, 472]
LEFT_PUPIL = [468, 469, 470, 471]
RIGHT_PUPIL = [473, 474, 475, 476]
LEFT_EYE_OUTER_CORNER = 33
RIGHT_EYE_OUTER_CORNER = 263
```

- Define the indices for the iris and pupil landmarks for both eyes.
- These indices correspond to the facial landmark positions defined by Mediapipe.

Step 5: Set Up the Webcam

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

- Initialize the webcam to capture video input.

Step 6: Define Distance Thresholds

```
distance_threshold_min = 59  # Minimum distance in cm
distance_threshold_max = 60  # Maximum distance in cm
pixel_distance_min = 100  # Minimum pixel distance between the eyes for 59 cm
pixel_distance_max = 110  # Maximum pixel distance between the eyes for 60 cm
```

- Define the distance thresholds for detecting how close the person is to the camera.
- Specify corresponding pixel distances based on camera calibration.

Step 7: Open CSV File

```
with open(csv_file, mode='w', newline='') as file:
writer = csv.writer(file)
writer.writerow(['Frame',
'Left Iris Center X', 'Left Iris Center Y', 'Left Iris Radius',
'Right Iris Center X', 'Right Iris Center Y', 'Right Iris Radius',
'Left Pupil Center X', 'Left Pupil Center Y', 'Left Pupil Radius',
'Right Pupil Center X', 'Right Pupil Center Y', 'Right Pupil Radius'])
```

- Open the CSV file in write mode and initialize the CSV writer.
- Write the headers to the CSV file to describe the data.

Step 8: Mediapipe FaceMesh Initialization

```
with mp_face_mesh.FaceMesh(
max_num_faces=1,
refine_landmarks=True,
min_detection_confidence=0.5,
min_tracking_confidence=0.5) as face_mesh:
```

- Initialize the Mediapipe Face Mesh with specified parameters.
- Limit detection to one face, enable landmark refinement, and set confidence thresholds.

Step 9: Frame Capture Loop

```
frame_number = 0
max_frames = 500  # Capture only 500 frames
process_start_frame = 50  # Start processing after first 50 frames
process_end_frame = max_frames - 50  # Stop processing before last 50 frames
```

- Initialize variables for frame capture.
- Limit the number of frames captured and set processing boundaries.

Step 10: Read Frame from Webcam

```
while cap.isOpened() and frame_number < max_frames:
success, image = cap.read()
if not success:
print("Ignoring empty camera frame.")
continue
```

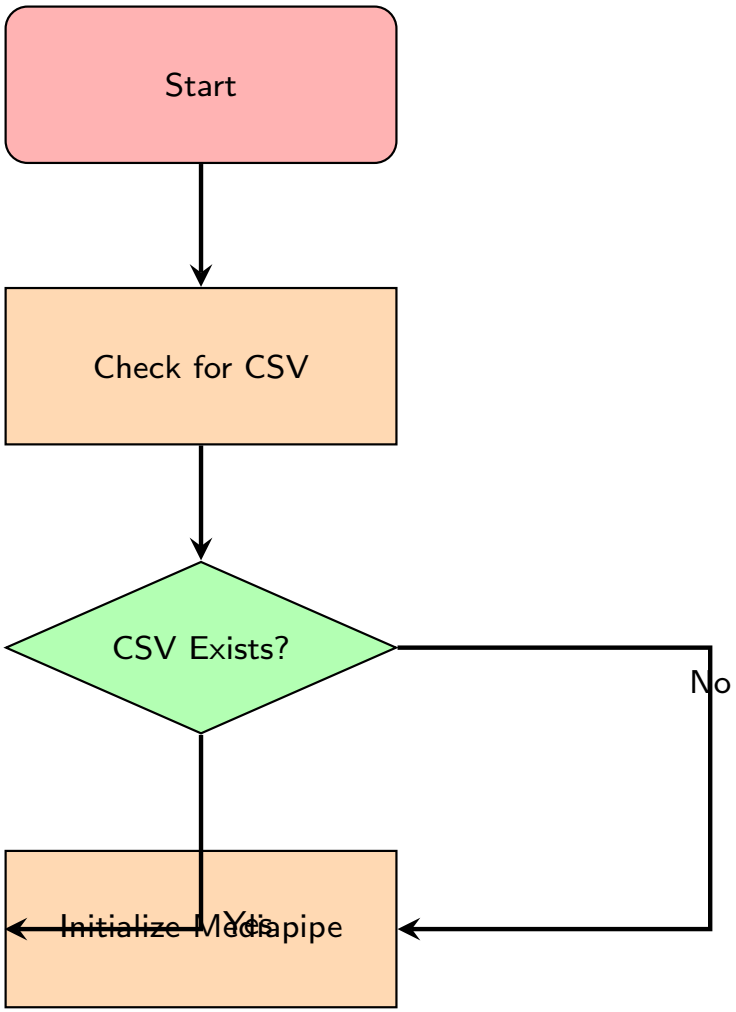
- Capture frames in a loop until the maximum frame limit is reached or webcam is closed.
- Check if the frame was successfully captured; if not, ignore the empty frame.

Step 11: Preprocess Image

```
image = cv2.cvtColor(cv2.flip(image, 1), cv2.COLOR_BGR2RGB)
image.flags.writeable = False
```

- Flip the image horizontally for a mirrored view.
- Convert the image color from BGR to RGB for processing with Mediapipe.
- Set the image as non-writable to optimize performance during processing.

Flowchart of the Iris Detection Program



Thank You!

Thank you for your attention!

Questions are welcome.