

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN – ĐIỆN TỬ**



NGUYỄN TẤN PHÁT

**MÔ HÌNH TỰ ĐỘNG GIÁM SÁT
TÌNH TRẠNG TÀI XẾ ỨNG DỤNG TRÍ TUỆ
NHÂN TẠO VÀ XỬ LÝ ẢNH**

**ĐỒ ÁN TỔNG HỢP
KỸ THUẬT ĐIỆN TỬ - TỰ ĐỘNG HOÁ**

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN – ĐIỆN TỬ**



NGUYỄN TẤN PHÁT

**MÔ HÌNH TỰ ĐỘNG GIÁM SÁT
TÌNH TRẠNG TÀI XẾ ỨNG DỤNG TRÍ TUỆ
NHÂN TẠO VÀ XỬ LÝ ẢNH**

**ĐỒ ÁN TỐT TỔNG HỢP
KỸ THUẬT ĐIỆN TỬ - TỰ ĐỘNG HOÁ**

Người hướng dẫn
TS. Nguyễn Hoàng Nam

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022

LỜI CẢM ƠN

Đầu tiên, em xin gửi lời cảm ơn trực tiếp đến thầy TS.Nguyễn Hoàng Nam, thầy là giảng viên hướng dẫn cũng như người đã truyền đạt trực tiếp kiến thức, giúp em hoàn thành đề tài “Mô hình tự động giám sát tình trạng tài xế ứng dụng trí tuệ nhân tạo và xử lý ảnh” một cách hoàn thiện nhất có thể.

Thứ hai, em xin gửi lời cảm ơn đến thầy cô khoa Điện-Điện tử, cùng bạn bè trong nhóm đã giúp em giải quyết những vướng mắc cũng như bổ sung lượng kiến thức còn thiếu để hoàn tất đề tài được giao.

Cuối cùng, là lời cảm ơn đến ba mẹ và gia đình trong suốt mùa dịch qua đã giúp em cũng như khích lệ, động viên tinh thần, qua đó giúp em hoàn thành môn học bằng hết khả năng của mình.

Một lần nữa, em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày 03 tháng 08 năm 2022

Tác giả

Công trình được hoàn thành tại Trường Đại học Tôn Đức Thắng

Cán bộ hướng dẫn khoa học: TS.Nguyễn Hoàng Nam

Đồ án tốt nghiệp/tổng hợp được bảo vệ tại **Hội đồng đánh giá Đồ án tốt nghiệp/tổng hợp của Trường Đại học Tôn Đức Thắng** vào ngày... /.../.....

Xác nhận của Chủ tịch Hội đồng đánh giá **Đồ án tốt nghiệp/tổng hợp và Trưởng khoa quản lý chuyên ngành sau khi nhận Đồ án tốt nghiệp/tổng hợp** đã được sửa chữa (nếu có).

CHỦ TỊCH HỘI ĐỒNG

TRƯỞNG KHOA

.....

.....

CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS.Nguyễn Hoàng Nam Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Đồ án tốt nghiệp/ tổng hợp còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Đồ án tốt nghiệp/ tổng hợp của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

TP. Hồ Chí Minh, ngày 03 tháng 08 năm 2022

Tác giả

Tp.HCM, ngày 28 tháng 03 năm 2022

NHIỆM VỤ ĐỒ ÁN TỔNG HỢP
(Bản nhiệm vụ này được đóng vào trang thứ nhất của đồ án)

Họ tên sinh viên: Nguyễn Tấn Phát

Ngành: Kỹ thuật điều khiển và tự động hóa Lớp: 15040302 MSSV: 41503006

1. Tên đề tài: Mô hình tự động giám sát tình trạng tài xế ứng dụng trí tuệ nhân tạo và xử lý ảnh
2. Nhiệm vụ (yêu cầu về nội dung và số liệu ban đầu):
 - Tìm hiểu mô hình hệ thống giám sát tình trạng tài xế, công nghệ trí tuệ nhân tạo và xử lý ảnh liên quan
 - Thiết kế và thi công mô hình hệ thống
 - Lập trình điều khiển và giám sát
 - Khảo sát chất lượng hệ thống
 - Viết báo cáo và trình bày poster theo mẫu

3. Ngày giao nhiệm vụ đồ án tốt nghiệp: 28/03/2022
4. Ngày bảo vệ 50% đồ án tốt nghiệp: 16/05/2022 - 20/05/2022
5. Ngày hoàn thành và nộp về khoa: 11/07/2022 - 15/07/2022
6. Giáo viên hướng dẫn: Phần hướng dẫn:
1. TS. Nguyễn Hoàng Nam 100%

Nội dung và yêu cầu ĐATN đã được thông qua Khoa và Bộ môn.

Ngày 04 tháng 04 năm 2022
CHỦ NHIỆM BỘ MÔN

GIẢNG VIÊN HƯỚNG DẪN

TS. Võ Hoàng Duy

TS. Nguyễn Hoàng Nam

TRƯỜNG KHOA

TS. Đồng Sĩ Thiên Châu

LỊCH TRÌNH LÀM ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Nguyễn Tấn Phát

Lớp: 1504030

MSSV: 41503006

Tên đề tài: Ứng dụng trí tuệ nhân tạo trong mô hình giám sát tình trạng của tài xế

Tuần/Ngày	Khối lượng		GVHD ký
	Đã thực hiện	Tiếp tục thực hiện	
1 (28/3-3/4/2022)	Chuẩn bị linh kiện cần thiết cho mô hình Tìm hiểu đề tài	Tìm hiểu code python, môi trường viết code, các hàm liên quan Viết python trên mini PC	
2 (4/4-10/4/2022)	Phương án 1 viết chương trình phát hiện tài xế ngủ gật trên python qua Facial Landmark Tìm hiểu môi trường ảo anaconda để viết chương trình	Chạy chương trình Đánh giá chương trình, phương án 1 Chạy chương trình trên mini PC	
3 (11/4-17/4/2022)	Đánh giá phương án 1 Chạy thử và fix bugs (nếu có) Tối ưu code	Chuẩn bị phương án 2, ứng dụng thêm phát hiện những đặc điểm ngoài để thêm tính chính xác cho code	

4 (18/4-24/4/2022)	Chạy thử phương án 2 Fix bugs (nếu có)	Tối ưu code cho phương án 2	
5 (25/4-1/5/2022)	Đánh giá phương án 2 So sánh phương án 1 và 2	Gộp tính năng phương án 1 và 2	
6 (2/5-8/5/2022)	Gộp tính năng phương án 1 và 2	Chuẩn bị phương án 3	
7 (9/5-15/5/2022)	Phương án 3: ứng dụng mô hình khác để phát hiện tài xế ngủ gật (yolov5)	Tìm kiếm thêm ảnh về hành động ngủ gật để máy học Sử dụng mô hình yolo cho phương án 3 Chuẩn bị báo cáo giữa kỳ.	
Kiểm tra giữa kỳ (16/5-22/5/2022)	Đánh giá khối lượng hoàn thành.....% được tiếp tục/không tiếp tục thực hiện ĐATN		
8 (23/5-30/5/2022)	Tiếp tục tối ưu code cho mô hình	Tiếp tục tối ưu code cho mô hình	
9 (31/5-6/6/2022)	Viết báo cáo Tối ưu code, thuật toán Kiểm tra lại mô hình	Tối ưu code, thuật toán Kiểm tra lại mô hình	
10 (7/6-13/6/2022)	Tối ưu code, thuật toán Kiểm tra lại mô hình	Tối ưu code, thuật toán Kiểm tra lại mô hình	
11 (14/6-21/6/2022)	Tối ưu code, thuật toán Kiểm tra lại mô hình	Tối ưu code, thuật toán Kiểm tra lại mô hình	

12 (22/6-29/6/2022)	Tối ưu code, thuật toán Kiểm tra lại mô hình	Tối ưu code, thuật toán. Kiểm tra lại mô hình	
13 (30/6-6/7/2022)	Tối ưu code, thuật toán Kiểm tra lại mô hình Nhận xét, đánh giá mô hình	Tối ưu code, thuật toán. Kiểm tra lại mô hình Thử nghiệm	
14 (7/7-14/7/2022)	Hoàn tất mô hình và báo cáo		
Nộp Đồ án tốt nghiệp	Đã hoàn thành.....% Đồ án tốt nghiệp được bảo vệ/không được bảo vệ ĐATN		

MÔ HÌNH TỰ ĐỘNG GIÁM SÁT TÌNH TRẠNG TÀI XẾ ỨNG DỤNG TRÍ TUỆ NHÂN TẠO VÀ XỬ LÝ ẢNH TÓM TẮT

- Với mong muốn học hỏi thêm về xử lý ảnh cũng như ứng dụng trí tuệ nhân tạo giải quyết các bài toán liên quan đến thực tế, em tham gia đề tài “Mô hình tự động giám sát tình trạng tài xế ứng dụng trí tuệ nhân tạo và xử lý ảnh”.
- Mong muốn rằng đề tài khi hoàn thành sẽ giúp ích cho tài xế- những người tham gia giao thông mỗi ngày . Với mô hình trên, tài xế sẽ được nhắc nhở mỗi khi ngủ gật hoặc có những triệu chứng buồn ngủ, qua đó, ý thức hơn trong lúc tham gia giao thông, bảo vệ an toàn cho bản thân và mọi người xung quanh.

MỤC LỤC

DANH MỤC HÌNH VẼ.....	XII
DANH MỤC BẢNG BIỂU.....	XV
DANH MỤC CÁC CHỮ VIẾT TẮT	XVI
CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI.....	1
1.1 LÝ DO CHỌN ĐỀ TÀI:	1
1.2 MỤC TIÊU VÀ NỘI DUNG NGHIÊN CỨU:	1
1.3 GIỚI HẠN CHO NỘI DUNG NGHIÊN CỨU:	2
1.4 BỐ CỤC:	2
CHƯƠNG 2. CƠ SỞ LÝ THUYẾT	4
2.1 TỔNG QUAN VỀ XỬ LÝ ẢNH:	4
2.2 MỘT SỐ VẤN ĐỀ CƠ BẢN CẦN QUAN TÂM:	6
2.2.1 Ảnh, điểm ảnh và một số vấn đề liên quan:	6
2.2.2 Hệ màu RGB ^[7] :.....	6
2.2.3 Biểu diễn một ảnh màu:	7
2.2.4 Khái niệm về Tensor :	9
2.2.5 Biểu diễn ảnh xám(gray image):.....	10
2.3 FACE DETECTION:.....	11
2.3.1 Thuật toán HaarCascade:.....	11
2.3.2 Cách hoạt động của HaarCascade:.....	11
2.4 THUẬT TOÁN FACIAL LANDMARK:	13
2.4.1 Lưu đồ cho thuật toán:.....	13
2.4.2 Các bước thực hiện thuật toán:.....	14
2.4.3 Khoảng cách Euclide :	17
2.5 NGÔN NGỮ VÀ LÝ DO CHỌN:	17
2.6 THƯ VIỆN MEDIAPIPE ^[4] :	17
2.7 THƯ VIỆN OPENCV VÀ NUMPY ^[1] :	18
2.7.1 Tổng quan về hai thư viện trên:	18
2.7.2 Một số hàm trong thư viện sử dụng trong đề tài ^{[2], [3]} :	19
CHƯƠNG 3. TÍNH TOÁN VÀ THIẾT KẾ.....	20
3.1 TỔNG QUAN VỀ THIẾT BỊ :	20
3.2 CHI TIẾT VỀ CÁC THIẾT BỊ:	21
3.2.1 CPU NUC:	21
3.2.2 Các thiết bị còn lại của mô hình:	22
3.2.3 Triển khai mô hình thực tế:	26
3.3 CÀI NHỮNG THƯ VIỆN CẦN THIẾT CHO THIẾT BỊ CŨNG NHƯ LƯU Ý KHI CÀI ĐẶT:	26
CHƯƠNG 4. THI CÔNG HỆ THỐNG.....	28
4.1 FLOWCHART HỆ THỐNG:	28
4.2 GIẢI THÍCH CÁC KHỐI CHÍNH:	29

CHƯƠNG 5. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ	32
5.1 MODULE 1: PHÁT HIỆN ĐỘ ĐÓNG MỞ CỦA MẮT VÀ MIỆNG:	32
5.1.1 <i>Phát hiện độ đóng mở của mắt:</i>	32
5.1.2 <i>Phát hiện độ đóng mở của miệng:</i>	38
5.1.3 <i>Kết hợp 2 mô hình trên:</i>	41
5.2 MODULE 2: PHÁT HIỆN HƯỚNG DI CHUYỂN CỦA MẶT (3D) ĐỂ PHÁT HIỆN ĐỘ TẬP TRUNG CỦA TÀI XẾ:.....	42
5.2.1 <i>Khai báo thư viện và tham số:</i>	42
5.2.2 <i>Khởi hàm chính:</i>	42
5.2.3 <i>Kết quả thu được:</i>	44
5.2.4 <i>Bảng thử nghiệm:</i>	46
5.3 MODULE 3: MODULE HOÀN CHỈNH:	47
5.4 MODULE 4: TRAINING 1 MODEL PHÁT HIỆN TRẠNG THÁI TÀI XẾ ĐANG TỈNH TÁO HAY BUỒN NGỦ ỨNG DỤNG YOLOV5:.....	47
CHƯƠNG 6. KẾT LUẬN	54
6.1 KẾT LUẬN:	54
6.1.1 <i>Ưu điểm:</i>	54
6.1.2 <i>Nhược điểm:</i>	54
6.2 HƯỚNG PHÁT TRIỂN:	55
TÀI LIỆU THAM KHẢO	56

DANH MỤC HÌNH VẼ

Hình 2.1: Mô hình đơn giản cho xử lý ảnh.....	4
Hình 2.2: Các bước xử lý cơ bản cho một hệ thống xử lý ảnh.....	5
Hình 2.3: Hệ màu RGB.....	6
Hình 2.4: 3 hệ số màu RGB.....	7
Hình 2.5 : Biểu diễn ma trận cho ảnh kích thước 800x600.....	7
Hình 2.6: Ảnh màu kích thước 3x3 biểu diễn dạng ma trận.....	8
Hình 2.7: Tách ma trận thành 3 ma trận cùng kích thước, mỗi ma trận ứng với từng màu R,G và B.....	8
Hình 2.8: Dạng tổng quát của ma trận kích thước 800x600.....	8
Hình 2.9: Biểu diễn dữ liệu 1 chiều(vector v) và 2 chiều(ma trận W).....	9
Hình 2.10: Hình hộp có kích thước (x,y,h)	9
Hình 2.11: Khối tensor có kích thước 28x28x3.....	10
Hình 2.12: Biểu diễn ảnh xám.....	10
Hình 2.13: (a) Bộ lọc cạnh– (b) bộ lọc đường– (c) bộ lọc bắt các đặc trưng hình vuông.....	11
Hình 2.14: Cơ bắt đặc trưng của Haar Cascade.....	12
Hình 2.15: Các bước nhận diện Haar Cascade.....	12
Hình 2.16: Khuôn mặt được chia thành 68 điểm.....	14
Hình 2.17: Mắt khi mở (tỉnh táo) và khi nhắm(ngủ).....	15
Hình 2.18 : Điểm landmarks cho mắt trái và mắt phải.....	16
Hình 2.19: Ảnh khuôn mặt cắt từ camera được chuyển thành 468 điểm cho bề mặt ảnh 3D.....	18
Hình 3.1 : Tổng quan về thiết bị.....	20
Hình 3.2: CPU NUC– bộ xử lý.....	21
Hình 3.3: Màn hình 8-inch dùng để hiển thị.....	22
Hình 3.4: Bàn phím và chuột dùng để điều khiển cũng như tương tác các thiết bị khác.....	22

Hình 3.5: Bộ nguồn 19V, 65W cấp nguồn cho CPU hoạt động.....	23
Hình 3.6 : Bộ nguồn 12V , 1A cấp nguồn cho màn hình.....	23
Hình 3.7 : Camera webcam 1080p.....	24
Hình 3.8 : Loa di động mini HOCO dùng để phát cảnh báo	24
Hình 3.9: Tẩu điện sang 220V DC , dùng cấp nguồn cho hệ thống trong môi trường thực tế.....	25
Hình 3.10 : Mô hình thực tế.....	25
Hình 3.11 : Các file cài đặt dlib cho window tùy phiên bản.....	26
Hình 4.1 : Mô hình khuôn mặt BlazeFace.....	29
Hình 5.1: 68 điểm landmarks từ dlib.....	34
Hình 5.2: Kết quả đo tỷ lệ mắt và xuất cảnh báo buồn ngủ khi độ đóng mở mắt đạt ngưỡng.....	37
Hình 5.3: Kết quả đo tỷ lệ miệng và xuất cảnh báo ngáp khi độ đóng mở miệng đạt ngưỡng.....	40
Hình 5.4 : Hệ thống update khi có thêm phát hiện ngáp , vẽ đường viền quanh mắt và miệng.....	41
Hình 5.5 : Khi ngưỡng mắt thấp trong khoảng thời gian đặt , cảnh báo buồn ngủ...41	
Hình 5.3 : Cảnh báo ngáp.....	42
Hình 5.4: Phát hiện hướng nhìn xuống và nhìn thẳng(ánh sáng đầy đủ).....	44
Hình 5.5 : Phát hiện được những tư thế còn lại(trường hợp ánh sáng đầy đủ).....	44
Hình 5.6 : Tư thế đầu thẳng (tập trung) trong điều kiện thiếu sáng và có đeo kính....45	
Hình 5.7 : Mắt tập trung khi nhìn sang trái lâu (có đeo kính).....	46
Hình 5.8 : Mắt tập trung khi nhìn sang phải lâu (có đeo kính).....	47
Hình 5.9 : Mô hình hoàn chỉnh kết hợp 3 mô hình trên.....	48
Hình 5.10 : Tập dataset đã thu thập.....	49
Hình 5.11: Quá trình đánh nhãn.....	50
Hình 5.12: Hành động buồn ngủ được đánh nhãn.....	50
Hình 5.13: Tập tin sau khi được đánh nhãn.....	51
Hình 5.14: Model dự đoán lúc tỉnh táo.....	52

Hình 5.15: Model dự đoán hành vi buồn ngủ (1).....	52
Hình 5.16: Model dự đoán hành vi buồn ngủ (2).....	53

DANH MỤC BẢNG BIỂU

DANH MỤC CÁC CHỮ VIẾT TẮT

EAR : EYE ASPECT RATIO

RGB: Red Green Blue

CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI

1.1 Lý do chọn đề tài:

- Thứ nhất, đề tài phát hiện tình trạng tài xế là một đề tài có tính ứng dụng cao trong đời sống hiện nay trong hiện trạng người dân sở hữu xe ô tô nhiều do đời sống phát triển. Trong khi việc gây tai nạn lúc lái xe diễn ra mỗi ngày. Trong đó, tài xế ngủ quên trên xe cũng như tình trạng sức khỏe mệt mỏi cũng là một trong những lý do chính dẫn đến tình trạng gây tai nạn không mong muốn.
- Thứ hai, qua đồ án này em có thể học hỏi thêm kiến thức về Computer Vision cũng như Machine Learning, các thuật toán và giải thuật về xử lý ảnh. Qua đó, có thể củng cố kiến thức bản thân, nghiên cứu và góp phần tạo ra những ứng dụng cũng như thuật toán hữu ích trong tương lai.
- Trên là hai lý do chính mà em quyết định chọn đề tài “Mô hình tự động giám sát tình trạng tài xế ứng dụng trí tuệ nhân tạo và xử lý ảnh” nhằm đưa ra ứng dụng thực tế, giúp tài xế có thể chủ động hơn đến tình trạng của bản thân khi đang tham gia giao thông, tránh những tình huống, tai nạn không mong muốn.

1.2 Mục tiêu và nội dung nghiên cứu:

Mục tiêu nghiên cứu đề tài trên , được chia thành những phần nghiên cứu sau:

- Tìm hiểu những kiến thức liên quan đến xử lý ảnh bằng ngôn ngữ python.
- Tìm hiểu những kiến thức , thuật toán giúp giải quyết bài toán trên.
- Tìm hiểu cách cài đặt , xử lý thư viện khi ứng dụng nhiều thư viện trong bài toán.
- Xây dựng chương trình , so sánh những ưu, nhược điểm của giải thuật.
- Tìm hiểu ứng dụng lắp đặt trên mô hình mini PC NUC.
- Viết báo cáo.

1.3 Giới hạn cho nội dung nghiên cứu:

Do đặc tính xử lý ảnh trong thực tế còn phụ thuộc rất nhiều vào yếu tố tự nhiên cũng như yếu tố thiết bị. Trong điều kiện cho phép cũng như theo khả năng cá nhân, đề tài em được thực hiện theo những giới hạn sau:

- Điều kiện tự nhiên như ngày nắng gắt, đêm tối, ... là khác nhau cho nên em thực hiện bài toán trên trong điều kiện ánh sáng ổn định để đạt kết quả tốt hơn.
- Đối với thiết bị, em chọn mini PC một phần vì tính nhỏ gọn, lưu động, bền bỉ, xử lý mạnh mẽ được các bài toán về nhận diện cũng như tích hợp rất nhiều cổng kết nối(USB , HDMI , audio , ...), tính năng tích hợp như wifi, bluetooth,... Ngoài ra, đối với nhu cầu mỗi người có thể dễ dàng nâng cấp thiết bị này sang bản cấu hình cao hơn tùy vào khả năng cho phép .

1.4 Bố cục:

Đề thuận tiện giải quyết nội dung và mục tiêu đã đề ra, báo cáo trên được chia thành các chương sau đây:

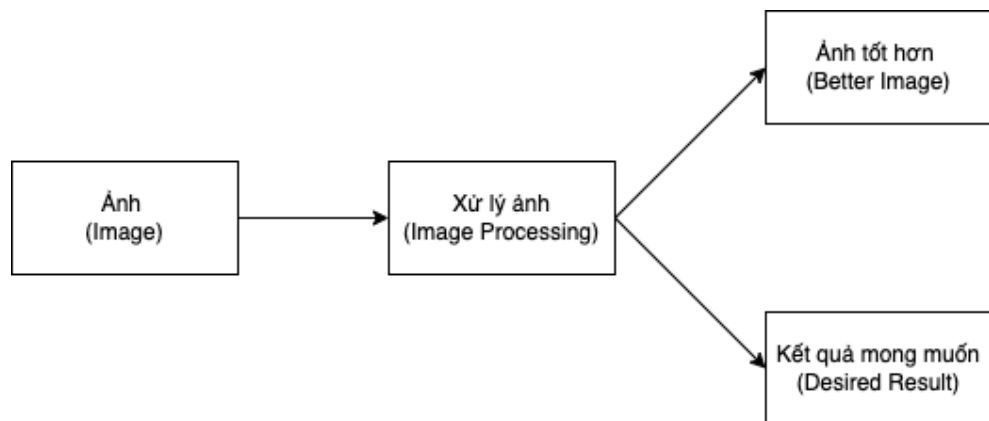
- Chương 1: Tổng quan đề tài.
 - Chương này trình bày các vấn đề liên quan đến lý do chọn đề tài, mục tiêu, nội dung nghiên cứu, các giới hạn mà đề tài gặp phải, bố cục đề tài.
- Chương 2: Cơ sở lý thuyết.
 - Chương này trình bày chính vào nguồn kiến thức và lý thuyết chính mà đề tài trên cần như xử lý ảnh, các thư viện liên quan, các giải thuật nhận diện như: Facial Landmarks, Mediapipe và module Face-mesh, khoảng cách Euclidean, ...
- Chương 3: Tính toán thiết kế hệ thống theo bài toán.
 - Chương này tổng quan về mini PC NUC, cách cài đặt các thư viện và set-up liên quan cho mô hình.

- Chương 4: Thi công hệ thống.
 - Chương này trình bày chương trình chính của hệ thống và xây dựng chương trình theo bài toán trên mini PC .
- Chương 5: Kết quả, nhận xét và đánh giá.
 - Chương này trình bày kết quả và nhận xét những gì mà mô hình và thuật toán có được. Qua đó, so sánh và đánh giá những ưu, nhược điểm, những điều đạt được hoặc chưa đạt được so với mục tiêu đề ra.
- Chương 6: Kết luận và hướng phát triển.
 - Chương này đưa ra kết luận tổng quan cuối cùng cho đồ án, đồng thời đưa ra hướng phát triển nếu mô hình được hoàn thiện để ứng dụng vào thực tiễn cuộc sống .

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

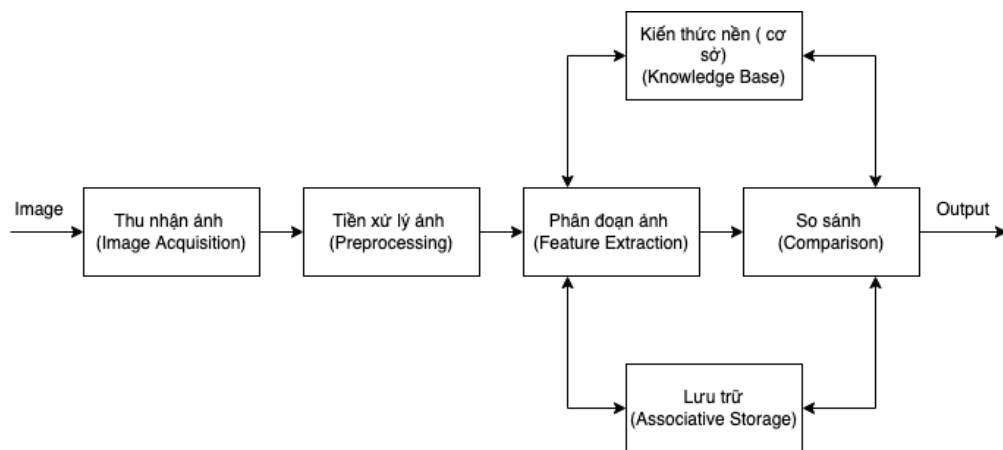
2.1 Tổng quan về xử lý ảnh:

- Xử lý ảnh là một nguồn kiến thức rộng lớn, một ngành khoa học mới mẻ nhưng phát triển nhanh chóng, đầy triển vọng, nó giúp ta giải quyết nhiều bài toán cho Machine learning(máy học), Deep learning(học sâu) và AI(Artificial Intelligence – Trí tuệ nhân tạo). Những bài toán này phát triển mạnh do nó giải quyết nhiều khía cạnh thực tiễn cho cuộc sống như: phân loại biển số xe trong các bãi đỗ, phân biệt con người có đeo khẩu trang nơi công cộng hay không, phát hiện ngu gậy ,
- Các phương pháp chính mà xử lý ảnh hướng đến như: phân tích ảnh , tăng chất lượng ảnh, tăng độ sáng và độ phân giải cho ảnh. Từ đó, giải quyết các bài toán liên quan. Sau đây , là mô hình đơn giản cho xử lý ảnh:



Hình 2.1: Mô hình đơn giản cho xử lý ảnh

- Trước đây, khi công nghệ còn chưa phát triển mạnh mẽ thì ảnh thu được từ camera là ảnh tương tự (analog). Hiện nay, với đà phát triển không ngừng của công nghệ, ảnh được lấy từ camera(màu hoặc trắng), sau đây, được chuyển trực tiếp sang ảnh số(digital). Mục đích cho việc chuyển sang ảnh số để giải quyết các bài toán ứng dụng vào đời sống như phát hiện người, vật thể....
- Mô tả sau đây là các bước cơ bản dành cho một hệ thống xử lý ảnh:



Hình 2.2: Các bước xử lý cơ bản cho một hệ thống xử lý ảnh

- Các module trên được mô tả như sau:

Bước Thu Nhận Ảnh: Đơn giản để hiểu thì khi camera đưa tới khung ảnh cần thì ảnh vật thể được thu nhận. Ảnh này trả về ảnh màu hoặc trắng đen.

Bước Tiền Xử Lý: Ảnh sau khi được thu nhận có thể gặp nhiều vấn đề như nhiễu hoặc độ tương phản thấp. Khi những vấn đề này xảy ra thì bộ tiền xử lý có nhiệm vụ nâng cao chất lượng ảnh, lọc nhiễu, tăng độ tương phản,... để hình ảnh tốt hơn, sắc nét và rõ ràng để thuận tiện cho những bước xử lý sau.

Bước Phân Đoạn Ảnh: Ảnh đưa vào là quá lớn, mà xử lý 1 vùng ảnh quá lớn thì tốn rất nhiều dung lượng và cần cấu hình cao. Đôi khi, lại không cho kết quả chính xác. Do vậy, ở bước này hình ảnh này được phân vùng thành những vùng ảnh nhỏ hơn để dễ dàng phân tích và nhận dạng ảnh.

Bước so sánh: Tại đây, ảnh được so sánh với các ngưỡng đặt ra tùy bài toán nhận dạng, ví dụ đặc trưng của đóng mở mắt ở ngưỡng 0.25-0.3, dưới mức này là mắt đang đóng, ngược lại là đang mở, ...

Kiến thức nền: Đơn giản hoá thì đây là bước dùng kiến thức cá nhân hoặc nghiên cứu để thuật toán chính xác hơn, nhẹ nhàng và đơn giản để chạy.

Lưu trữ: để dữ liệu được so sánh và chạy liên tục, thì bắt buộc ta phải có bộ lưu trữ, ta có thể lưu trữ trên local máy hoặc tiện lợi hơn thì lưu trữ trên đám mây(cloud). Việc hiện đại hoá như hiện tại thì lưu trữ đám mây cũng rất tiện lợi và không tốn quá nhiều chi phí. Hơn thế, lưu trữ đám mây ít gặp những

vấn đề về mất mát dữ liệu như lưu trữ cục bộ(local). Do vậy, nên ứng dụng nhiều hơn việc lưu trữ này cho các bài toán cần lượng data lớn.

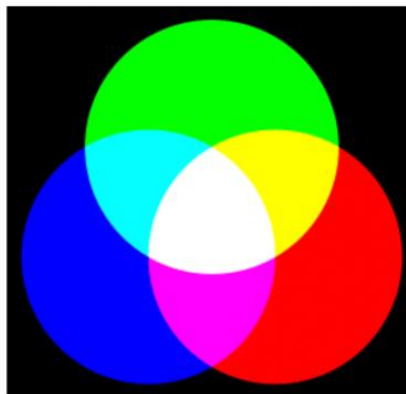
2.2 Một số vấn đề cơ bản cần quan tâm:

2.2.1 Ảnh, điểm ảnh và một số vấn đề liên quan:

- Đầu tiên khái niệm ảnh: là một tập hợp lớn gồm rất nhiều điểm ảnh(hay còn gọi là pixel)
- Pixel ảnh(hay điểm ảnh): là phần tử nhỏ nhất trong tập hợp ảnh có tọa độ (x,y). Pixel gồm pix (picture), el thay cho element, dịch chung là 1 phần tử của tấm ảnh. Điểm ảnh thường được ví như nguyên tử của một ảnh.
- Mỗi một pixel ảnh biểu diễn một màu sắc nhất định.
- Khoảng cách giữa các pixel ảnh càng gần hoặc mật độ điểm ảnh càng lớn (độ phân giải ảnh) thì ảnh càng nét , càng sống động. Độ phân giải này ví như một mặt phẳng 2D(x,y) chứa các điểm ảnh ở trong, độ phân giải càng lớn, ảnh chứa càng nhiều thông tin và càng rõ nét hơn.

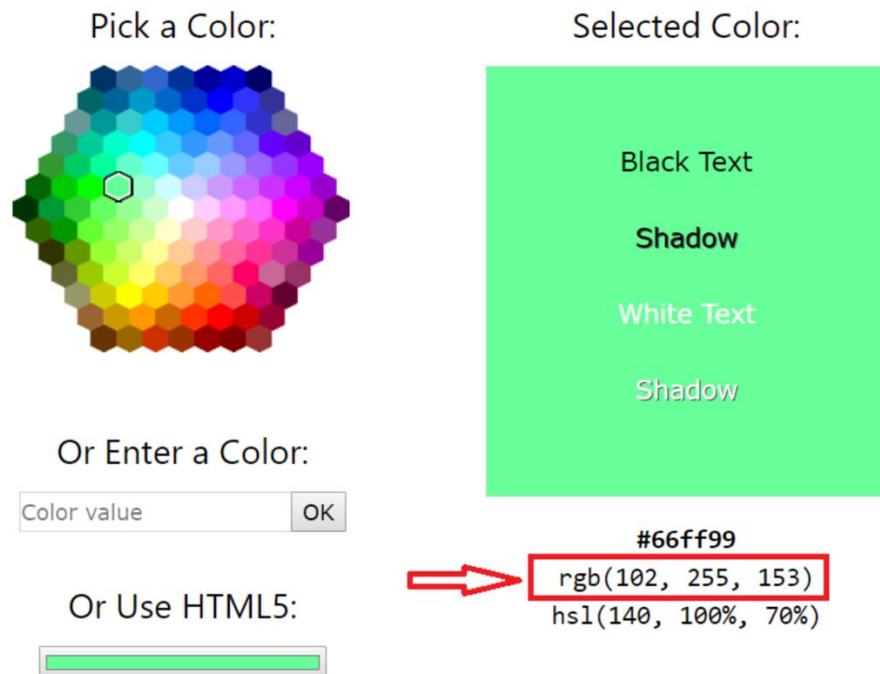
2.2.2 Hệ màu RGB^[7]:

- Đây là kênh màu gồm 3 kênh chính gồm Red (đỏ), Green (xanh lục), Blue (xanh lam), nhờ 3 kênh trên, trộn chúng với nhau theo 1 tỷ lệ ta tạo nên những màu khác.



Hình 2.3: Hệ màu RGB^[7]

- Khi ta chọn 1 màu sắc bất kỳ thì ta được 1 bộ 3 kênh màu như ảnh 2.5. Bộ 3 màu này chạy từ số nguyên $[0,255]$, với mỗi 3 tương ứng sẽ cho ra 1 màu khác nhau. Do mỗi màu có tổng 256 cách chọn, nên tùy cách phối màu mỗi người sẽ ra được 1 màu khác nhau, tổng số màu có thể tạo ra được là $256 \times 256 \times 256 = 167.777.216$ màu.



Hình 2.4: 3 hệ số màu RGB^[7]

2.2.3 Biểu diễn một ảnh màu:

- Ví dụ để biểu diễn một ảnh màu có kích thước 800×600 pixels tương ứng chiều dài ảnh là 800 pixels, chiều rộng là 600 pixels ta biểu diễn chúng bằng một ma trận có kích thước 600×800 (do định nghĩa ma trận gồm số hàng nhân số cột).

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \dots & w_{1,800} \\ w_{2,1} & w_{2,2} & \dots & w_{2,800} \\ \dots & \dots & \dots & \dots \\ w_{600,1} & w_{600,2} & \dots & w_{600,800} \end{bmatrix}$$

Hình 2.5 : Biểu diễn ma trận cho ảnh kích thước 800×600

- Trong ma trận trên, mỗi phần tử $\omega_{i,j}$ biểu diễn cho một pixel. Mỗi pixel tương ứng cho 1 màu, tấm ảnh có kích thước như trên sẽ chứa rất nhiều pixels. Cụ thể, ta vẽ 1 hình chữ nhật có kích thước dài 800, rộng 600 đơn vị, rồi chia hình chữ nhật này như ô cờ ca rô với mỗi hình vuông nhỏ có kích thước 1x1 đơn vị, thì mỗi ô vuông là một pixel, mỗi pixel biểu diễn một màu.
- Nhưng để biểu diễn 1 ô vuông pixel trên ta cần 3 thông số R, G và B tương ứng (r,g,b), do vậy, $\omega_{i,j} = (r_{i,j}, g_{i,j}, b_{i,j})$. Biểu diễn dưới dạng ma trận sẽ như sau:

$$\begin{bmatrix} (100, 100, 50) & (101, 112, 3) & (131, 20, 80) \\ (150, 210, 130) & (10, 120, 130) & (111, 120, 130) \\ (10, 260, 30) & (200, 20, 30) & (100, 20, 3) \end{bmatrix}$$

Hình 2.6: Ảnh màu kích thước 3x3 biểu diễn dạng ma trận

- Trong việc lưu trữ và xử lý, ta không thể nào lưu trữ dạng ma trận như trên mà phải tách giá trị màu trong mỗi pixel ảnh ra một ma trận riêng để dễ lưu trữ và xử lý bài toán. Khi đó, m trận trên trở thành:

$$\begin{bmatrix} 100 & 101 & 131 \\ 150 & 10 & 111 \\ 10 & 200 & 100 \end{bmatrix}, \begin{bmatrix} 100 & 112 & 20 \\ 210 & 120 & 130 \\ 260 & 20 & 20 \end{bmatrix}, \begin{bmatrix} 50 & 3 & 80 \\ 130 & 130 & 130 \\ 30 & 30 & 3 \end{bmatrix}$$

R

G

B

Hình 2.7: Tách ma trận thành 3 ma trận cùng kích thước, mỗi ma trận ứng với từng màu R,G và B^[7]

$$\begin{bmatrix} (r_{1,1}, g_{1,1}, b_{1,1}) & (r_{1,2}, g_{1,2}, b_{1,2}) & \dots & (r_{1,800}, g_{1,800}, b_{1,800}) \\ (r_{2,1}, g_{2,1}, b_{2,1}) & (r_{2,2}, g_{2,2}, b_{2,2}) & \dots & (r_{2,800}, g_{2,800}, b_{2,800}) \\ \dots & \dots & \dots & \dots \\ (r_{600,1}, g_{600,1}, b_{600,1}) & (r_{600,2}, g_{600,2}, b_{600,2}) & \dots & (r_{600,800}, g_{600,800}, b_{600,800}) \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} r_{1,1} & r_{1,2} & \dots & r_{1,800} \\ r_{2,1} & r_{2,2} & \dots & r_{2,800} \\ \dots & \dots & \dots & \dots \\ r_{600,1} & r_{600,2} & \dots & r_{600,800} \end{bmatrix}, \begin{bmatrix} g_{1,1} & g_{1,2} & \dots & g_{1,800} \\ g_{2,1} & g_{2,2} & \dots & g_{2,800} \\ \dots & \dots & \dots & \dots \\ g_{600,1} & g_{600,2} & \dots & g_{600,800} \end{bmatrix}, \begin{bmatrix} b_{1,1} & b_{1,2} & \dots & b_{1,800} \\ b_{2,1} & b_{2,2} & \dots & b_{2,800} \\ \dots & \dots & \dots & \dots \\ b_{600,1} & b_{600,2} & \dots & b_{600,800} \end{bmatrix}$$

Hình 2.8: Dạng tổng quát của ma trận kích thước 800x600

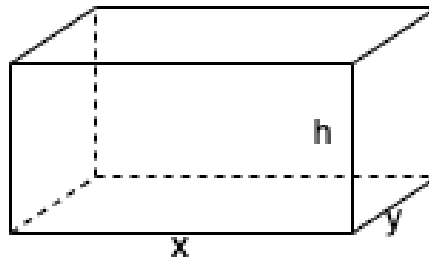
2.2.4 Khái niệm về Tensor :

- Để biểu diễn dữ liệu 1 chiều (1D) người ta dùng vector, dạng mặc định của vector là dạng cột.
- Khi dữ liệu dạng 2 chiều (2D) để biểu diễn, người ta dùng ma trận (có kích thước số hàng nhân số cột).

$$v = \begin{bmatrix} v_1 \\ v_2 \\ \dots \\ v_n \end{bmatrix}, W = \begin{bmatrix} w_{11} & w_{12} & \dots & w_{1n} \\ w_{21} & w_{22} & \dots & w_{2n} \\ \dots & \dots & \dots & \dots \\ w_{m1} & w_{m2} & \dots & w_{mn} \end{bmatrix}$$

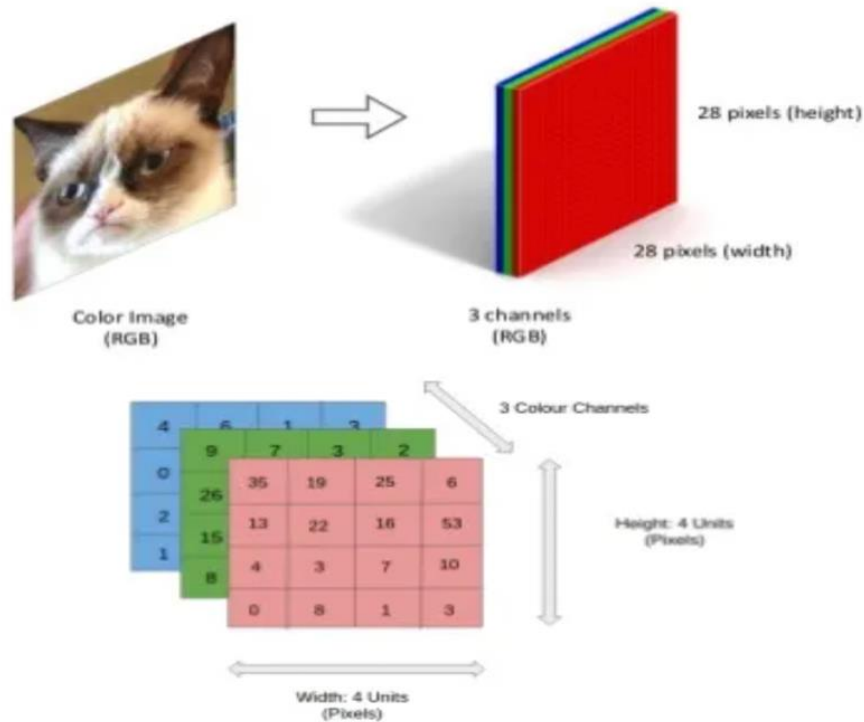
Hình 2.9: Biểu diễn dữ liệu 1 chiều(vector v) và 2 chiều(ma trận W)

- Tuy nhiên, dữ liệu không chỉ giải quyết ở 1 chiều hoặc 2 chiều mà còn ở dạng 3 chiều hay hơn. Để giải quyết tình trạng trên, ta có khái niệm tensor dành cho dữ liệu ở nhiều chiều, ví như dữ liệu 3 chiều. Đơn giản thì tensor là sự sắp xếp nhiều ma trận $m \times n$ kích thước giống nhau, chồng lên nhau.



Hình 2.10: Hình hộp có kích thước (x,y,h)

- Theo như định nghĩa trên, ta có thể hình dung mặt đáy(x,y) là ma trận $x \times y$, cả hình hộp ta xếp chồng h ma trận trên lên nhau.



Hình 2.11: Khối tensor có kích thước 28x28x3^[7]

2.2.5 Biểu diễn ảnh xám(gray image):

- Ảnh xám (monochromatic): tương tự như ví dụ trên ta cũng biểu diễn 1 ảnh có kích thước 800x600 nhưng thay vì có 3 kênh màu(channels) như RGB thì ảnh xám chỉ biểu diễn bởi giá trị chạy từ 0 đến 255(càng tiến đến 255 càng nhạt). Với giá trị biểu diễn màu đen là 0, màu trắng là 255.

$$\begin{bmatrix} 0 & 215 & \dots & 250 \\ 12 & 156 & \dots & 1 \\ \dots & \dots & \dots & \dots \\ 244 & 255 & \dots & 12 \end{bmatrix}$$

Hình 2.12: Biểu diễn ảnh xám

- Để chuyển từ hệ màu RGB sang ảnh xám ta chỉ dùng 1 biến x để biểu diễn (do 3 kênh màu quy đổi thành 1 kênh) với công thức được biểu diễn như sau:

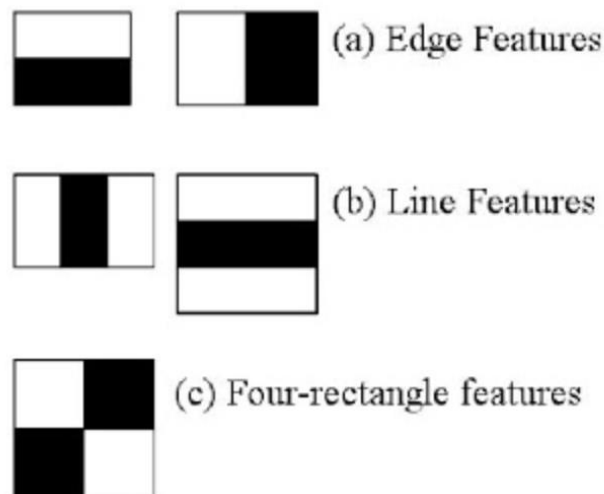
$$x = r*0.299 + g*0.587 + b*0.114 .$$

2.3 Face Detection:

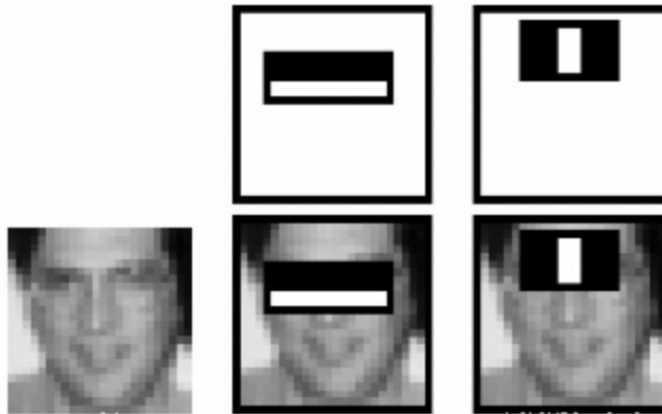
2.3.1 Thuật toán HaarCascade^[25]:

- Thuật toán để phát hiện khuôn mặt thường dùng là Haar Cascade (do Viola-Jones đồng sáng tạo) được phát triển vào năm 2001 trong những bài báo của riêng họ.
- Haar được xem như những model đã được training sẵn với các đặc trưng của các phần trên cơ thể cần phát hiện . Haar cascade được hiểu đơn giản là sử dụng haar qua thật nhiều lượt để trích xuất đặc trưng qua đó , cho kết quả tối ưu nhất mà ta mong muốn.
- Ta có thể down load những model đầy tại trang github của chính họ cho việc xử lý bài toán mà ta cần.

2.3.2 Cách hoạt động của HaarCascade:

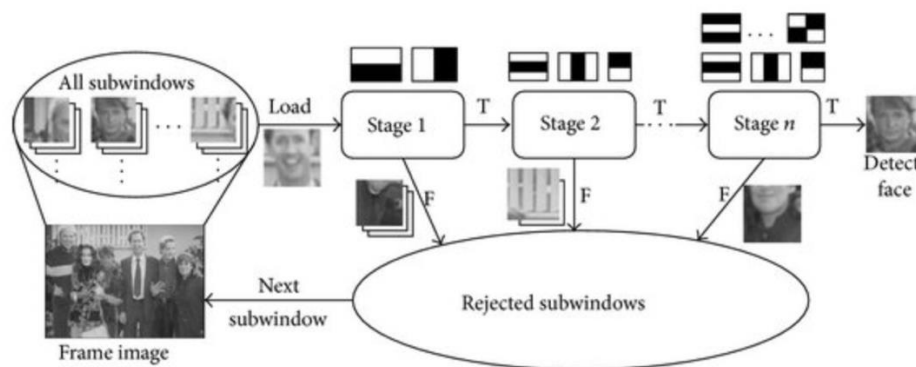


Hình 2.13: (a) Bộ lọc cạnh– (b) bộ lọc đường– (c) bộ lọc bắt các đặc trưng hình vuông



Hình 2.14: Cơ bản đặc trưng của Haar Cascade

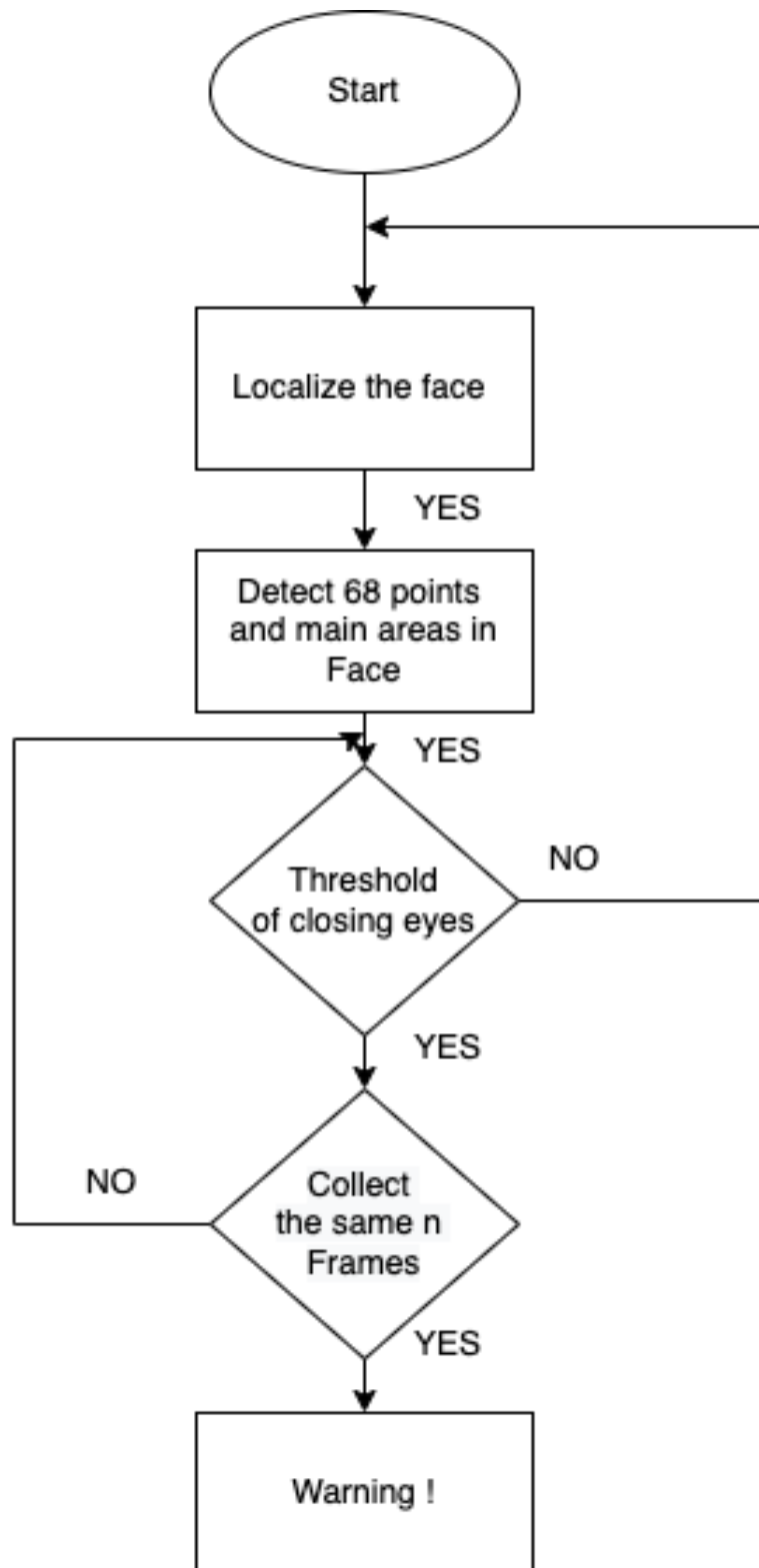
- Một khuôn mặt có những đặc điểm sau: ví dụ vùng mắt sẽ tối hơn vùng dưới mặt, mắt sẽ tối hơn mũi. Như ảnh minh họa trên, cơ chế chạy trên 1 ảnh của Haar gọi là running window, tức những cửa sổ chạy. Những cửa sổ này chạy xuống dần, tìm kiếm. Khi vùng nhận diện apply khớp từ vài chục đến vài chục nghìn window giống kết quả thì cho kết quả là mặt người hay không.



Hình 2.15: Các bước nhận diện Haar Cascade

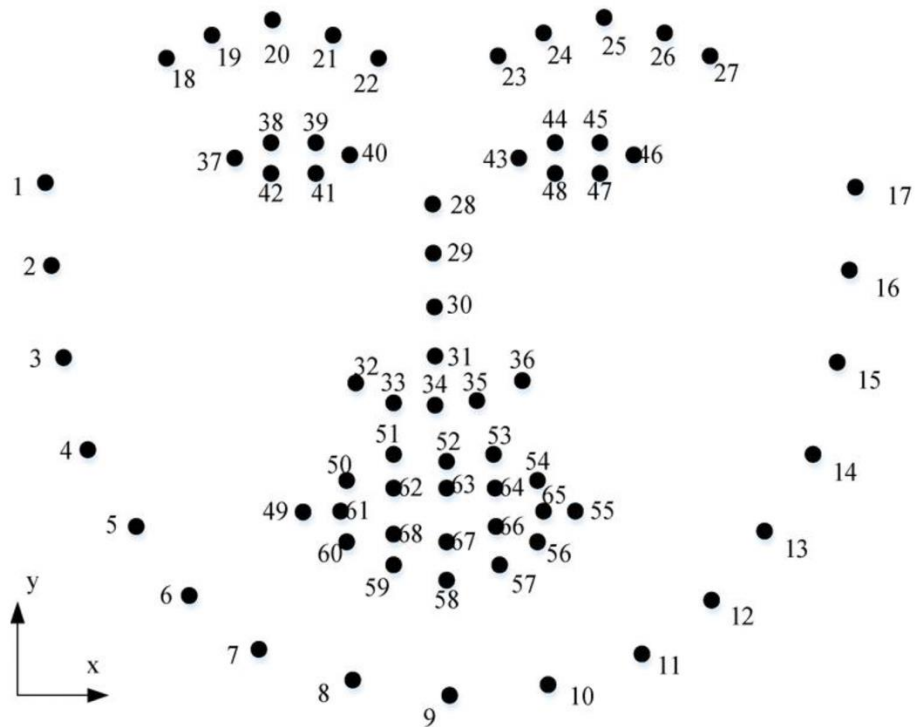
2.4 Thuật toán Facial Landmark:

2.4.1 Lưu đồ cho thuật toán:



2.4.2 Các bước thực hiện thuật toán:

- **Bước 1:** Detect khuôn mặt bằng Facial Landmark cùng thư viện Dlib (v18.10) và OpenCV(chia khuôn mặt thành 68 điểm) gồm 7 khu vực chính trên khuôn mặt:
 - Mắt (Eyes) gồm: mắt trái, mắt phải.
 - Chân mày (EyeBrows) gồm: chân mày trái, chân mày phải.
 - Mũi (Nose).
 - Miệng (Mouth).
 - Cằm (Jawline).



Hình 2.16: Khuôn mặt được chia thành 68 điểm

- Facial Landmark dùng để nhận ra những điểm quan trọng trên khuôn mặt , sử dụng phương thức dự đoán hình dạng (shape prediction).
- Có 2 bước tiến trình chính của Facial Landmark là:
 - **B1** : Khoanh vùng khuôn mặt trong ảnh (localize the face in the image).

- **B2** : Phát hiện ra những điểm chính yếu của khuôn mặt trên Face ROI (Automatic Region of Interest) – Khoanh vùng những khu vực chính.
- B1 ta ứng dụng thư viện OpenCV với phương pháp Haar cascades đã được xây dựng sẵn để khoanh vùng được khuôn mặt.
- Bộ chuyển đổi khuôn mặt thành 68 điểm nằm trong thư viện Dlib.
- B2 được sử dụng dựa trên:
 - Một tập khuôn mặt đã được training sẵn , đánh nhãn thủ công . Những ảnh này chia khuôn mặt theo toạ độ x,y , thành những điểm cụ thể xung quanh cấu trúc khuôn mặt (ảnh 2).
 - Cụ thể , phương pháp này tính xác suất về khoảng cách của các pixels ảnh đầu vào.
- Dựa trên B1 và B2, phương pháp Facial Landmark trên chia khuôn mặt thành 68 điểm. Ứng dụng kết quả đây, ta áp dụng để phát hiện khuôn mặt cũng như giải các bài toán liên quan trong thực tế.
- Để phát hiện liệu tài xế có ngủ gục hay không , ta xét cụm điểm mắt trái [37;42] và cụm điểm mắt phải [43;48]. Để xét xem tài xế có buồn ngủ hay không ta ứng dụng.
- EAR (Eye Aspect Ratio): tỷ lệ khung hình của mắt.
- Như đã trình bày bên trên, để phát hiện xem tài xế có ngủ gật không, ta sẽ xét xoay quanh 6 điểm mỗi mắt (37 đến 42 cho mắt trái và 43 đến 48 cho mắt phải).



Hình 2.17: Mắt khi mở (tỉnh táo) và khi nhắm(ngủ)

- Công thức tính tỷ lệ khung hình của mắt :

$$EAR = \frac{(p2 - p6) + (p3 - p5)}{2 \cdot (p1 - p4)}$$

- Tỷ lệ EAR càng lớn thì mắt mở càng to.
- Tỷ lệ EAR càng bé thì chứng tỏ mắt đang đóng.



Hình 2.18 : Điểm landmarks cho mắt trái và mắt phải.

- Như ảnh trên thì [37;42] và [43;48] là những điểm cho mắt trái và phải nhưng thư viện face_utils có thể giúp ta lấy những điểm này thông qua cách gọi tên “left_eye” và “right_eye”.
- **Bước 2:** Detect khuôn mặt qua camera(B1 trong tiến trình của Facial Landmark).
- **Bước 3:** Chọn xét 2 cụm Landmark [37;42] và [43;48]. Sau đấy, tính toán khoảng cách giữa 2 mí mắt và đặt ngưỡng. Việc đặt ngưỡng để xem khi nào ta mở mắt và khi nào nhắm mắt. Khi nhắm mắt khoảng cách giữa 2 mí mắt nhỏ, ngược lại khi mở mắt thì khoảng cách này to.
- **Bước 4:** Nếu trong n frame (các frame ảnh liên tiếp nhau được thu về - do mình cấu hình) mà phát hiện được ngưỡng thấp (mắt nhắm), thì ta tiến hành phát audio cảnh báo. Ta nối dây công 3.5 để phát âm thanh ra.
- Ta xác định tương tự để tìm ra ngưỡng mở của miệng và cảnh báo ngáp.

2.4.3 Khoảng cách Euclide :

- Để giải quyết vấn đề liên quan đến khoảng cách cách điểm ta cần tìm hiểu về khoảng cách euclide, từ đó giải quyết được các bài toán liên quan đến độ đóng mở mắt và miệng.
- Để hiểu đơn giản, thì khoảng cách giữa 2 điểm A và B sẽ là chiều dài của đoạn thẳng AB nối 2 điểm này lại với nhau. Ta xét trong hệ toạ độ Decartes, nếu ta có 2 điểm $A=(A_1,A_2,A_3,A_4,\dots,A_x)$ tương tự điểm $B=(B_1,B_2,B_3,\dots,B_x)$ nằm chứa trong không gian Euclide thì khoảng AB có chiều dài được xác định theo công thức:

$$\begin{aligned}d(a,b) &= \sqrt{(A_1-B_1)^2 + (A_2-B_2)^2 + (A_3-B_3)^2 + \dots + (A_x-B_x)^2} \\ &= \sqrt{\sum_{i=1}^x (A_i-B_i)^2}\end{aligned}$$

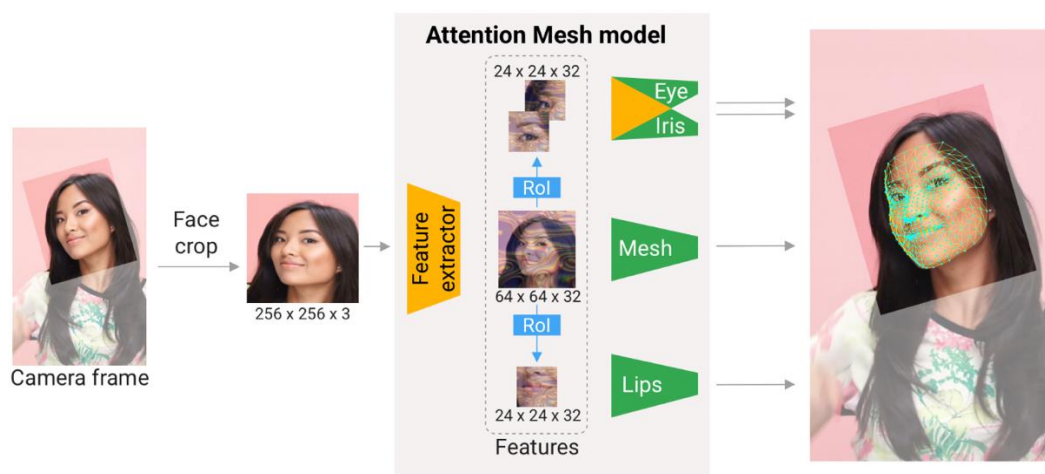
2.5 Ngôn ngữ và lý do chọn:

- Để giải quyết những bài toán đã đưa ra, em quyết định chọn python làm ngôn ngữ chính để nghiên cứu do những đặc tính sau:
 - Đây là ngôn ngữ mã nguồn mở, có cộng đồng lớn nên sẽ dễ mở rộng dự án bằng việc học hỏi được nhiều hơn cũng như tham khảo được nhiều cái hay cái mới để ứng dụng vào bài toán thực tế đang giải quyết.
 - Ngôn ngữ có syntax đơn giản, dễ đóng gói và mở rộng .
 - Giải quyết được nhiều vấn đề liên quan đến các bài toán xử lý ảnh hiện đại.

2.6 Thư viện Mediapipe^[4]:

- Thư viện Dlib bên trên giúp chúng ta giải quyết vấn đề độ đóng mở của mắt hoặc miệng qua 68 điểm để đưa ra cảnh báo nếu tài xế ngủ gật nhưng vấn đề gặp phải là nó chỉ chính xác nếu khuôn mặt tài xế nhận được trên frame camera là 2D.
- Do vậy, để tăng tính chính xác cho thuật toán, ta sử dụng thêm thư viện Mediapipe, cụ thể là ứng dụng module Face Mesh để chia ảnh khuôn mặt thành

một mạng lưới gồm 468 điểm kết nối nhau, tạo thành bề mặt 3D bao phủ khuôn mặt. Từ đó, xác định hướng khuôn mặt có lệch khỏi tầm camera không và có biểu hiện buồn ngủ hay không. Khi tài xế không nhìn đối diện vào khung camera (đặt khung đúng với vị trí lái xe) một khoảng thời gian, hay đang nhìn vào những hướng khác quá lâu thì sẽ cảnh báo tài xế đang mất tập trung, có thể gây nguy hiểm khi tham gia giao thông. Từ đó, phát tín hiệu âm thanh cảnh báo tài xế.



Hình 2.19: Ảnh khuôn mặt cắt từ camera được chuyển thành 468 điểm cho bề mặt ảnh 3D^[4]

2.7 Thư viện OpenCV và Numpy^[1]:

2.7.1 Tổng quan về hai thư viện trên:

- Thư viện Open CV(Open-computer-Vision), đúng như tên gọi, đây là một thư viện mở lớn dành cho việc xử lý các bài toán liên quan đến thị giác máy tính cũng như giải quyết các vấn đề liên quan xử lý ảnh hiện tại.
- Thư viện này được build bằng ngôn ngữ C/C++ , nên cho kết quả tính toán output cực kì nhanh .Do vậy ,ta có thể áp dụng thư viện vào các ứng dụng liên quan đến xử lý realtime (thời gian thực tế) như đề tài: “Mô hình tự động giám sát tình trạng tài xế ứng dụng trí tuệ nhân tạo và xử lý ảnh”

- Để cài đặt open-cv python trên mô hình cũng như môi trường lab, ta dùng công cụ pip, có lệnh như sau : “*pip install opencv-python*”.
- Về Numpy, đây là một thư viện dùng để xử lý các vấn đề liên quan đến mảng. Ở đây, ta sẽ chuyển các ảnh kỹ thuật số về dạng mảng hay ma trận, từ đó trích xuất các thông số để so sánh, giải quyết bài toán đưa ra.
- Để sử dụng 2 thư viện trên, ta khai báo như sau:

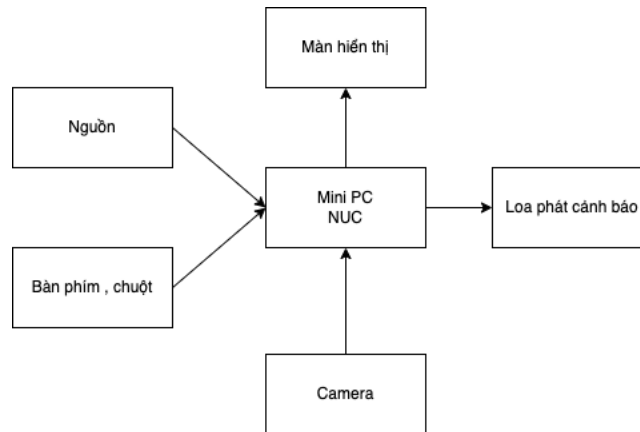
“ *import cv2* ” , “*import numpy as np*”

2.7.2 Một số hàm trong thư viện sử dụng trong đề tài^{[2], [3]}:

- *cv2.VideoCapture()*: ta tạo 1 biến, gán tham số vào hàm, với tham số 0-n, hàm trả về ID của camera kết nối với CPU, với tham số là file ảnh hoặc video, hàm trả về file ảnh hoặc video đó. Tùy vào ý đồ bài toán mà ta chọn hình ảnh, video hoặc realtime camera.
- *cv2.cvtColor()*: nếu ta cần hệ màu khác BGR, hàm giúp ta chuyển thành hệ màu thích hợp. Như thư viện mediapipe cần RGB thì truyền vào tham số *cv2.BGR2RGB*.
- *cv2.flip(frame, 1)*: lật ngược frame ảnh hiện tại theo trục y. Nếu muốn lật frame theo trục x ta đổi tham số 1 thành 0.
- *cv2.imshow()*: để khởi động frame. Truyền vào 2 tham số là tên frame và frame ảnh muốn khởi động.
- *cv2.putText()*^[5]: dùng để hiển thị text trên frame cảnh báo.
- *cv2.waitKey()*: truyền vào tham số 0 sẽ tạo nên vòng lặp vô cực cho frame ảnh. Như ta mở cam lên thì frame đầu sẽ lặp vô hạn khiến cho frame không hiển thị được liên tục các frame ảnh sau. Thường ta sẽ kết hợp với 1 số bất kỳ (1ms hoặc số ms mong muốn) và điều kiện để kết thúc và thoát vòng lặp.
- *np.concatenate()*: nối các mảng Numpy.

CHƯƠNG 3. TÍNH TOÁN VÀ THIẾT KẾ

3.1 Tổng quan về thiết bị :



Hình 3.1 : Tổng quan về thiết bị

- Với yêu cầu bài toán trên, thiết bị em chọn gồm có:
- CPU NUC: là khối chính giải quyết các thuật toán, được ví như thiết bị đầu não của cả mô hình hệ thống.
- Màn hiển thị: 8 inch có kèm cảm ứng để làm 1 màn phụ hiển thị các kết quả ra thực tế.
- Bàn phím, chuột: dùng để nhập xuất code cũng như tương tác giữa cpu và các thiết bị khác.
- Khối loa: dùng để phát cảnh báo nếu tài xế ngủ gật hoặc mất tập trung.
- Nguồn: khối nguồn cấp cho CPU hoạt động.
- Kết nối: ngoài camera cắm trực tiếp vào cổng usb và màn hình cần xuất từ cổng hdmi. Các thiết bị còn lại đều kết nối qua bluetooth, khiến mô hình trở nên linh hoạt dễ dàng mang theo, cũng như triển khai lắp đặt.

3.2 Chi tiết về các thiết bị:

3.2.1 CPU NUC:

- Lý do chọn cpu NUC, là do bộ xử lý intel mạnh mẽ, thích hợp giải quyết những bài toán về xử lý ảnh.
- Tính nhỏ gọn, dễ lắp đặt.
- Nhiều cổng kết nối.



Hình 3.2: CPU NUC– bộ xử lý

- Cấu hình CPU NUC:
 - Intel core i3 7100U
 - Intel HD620 Graphics
 - SSD 120Gb
 - Ram 8Gb DDR4
 - Intel Wireless-AC 8265 + Bluetooth 4.2
 - 4 x USB 3.0
 - 1 x GbE LAN 10/100/1000
 - 1 x HDMI
 - 1 x DisplayPort 1.2 (via USB type C)

- 1 x Line Out (SPDIF)
- 1 x micro SDXC card slot
- 1 x adaptor 19V, 65W
- 1 x VESA mount Bracket
- Chạy win 10 pro

3.2.2 Các thiết bị còn lại của mô hình:



Hình 3.3: Màn hình 8inch dùng để hiển thị



Hình 3.4: Bàn phím và chuột dùng để điều khiển cũng như tương tác các thiết bị khác



Hình 3.5: Bộ nguồn 19V, 65W cấp nguồn cho CPU hoạt động



Hình 3.6 : Bộ nguồn 12V , 1A cấp nguồn cho màn hình



Hình 3.7 : Camera webcam 1080p



Hình 3.8 : Loa di động mini HOCO dùng để phát cảnh báo



Hình 3.9: Tẩu điện sang 220V DC , dùng cấp nguồn cho hệ thống trong môi trường thực tế



Hình 3.10 : Mô hình thực tế

3.2.3 Triển khai mô hình thực tế:

- Khi ở trong thực tế ta sử dụng tần điện 220V DC để cấp nguồn cho cả thiết bị.
- Loa mini sẽ được thay thế bằng dàn loa của xe hơi, kết nối với NUC qua cổng AUX.
- Camera, NUC và màn hình phụ được lắp đặt theo vị trí ngồi của tài xế sao cho thoải mái nhất do tính cơ động của mỗi thiết bị.

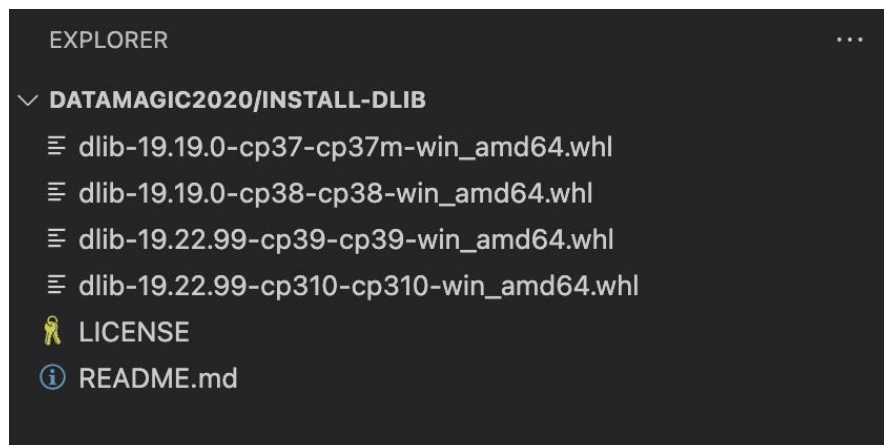
3.3 Cài những thư viện cần thiết cho thiết bị cũng như lưu ý khi cài đặt:

- Đầu tiên, ta cài opencv-python cho thiết bị qua lệnh:
“pip install opencv-python”
- Tiếp theo, ta cài cmake và dlib, lưu ý phải cài thư viện cmake để hỗ trợ cho thư viện dlib, không có cmake không thể cài dlib, cài những thư viện trên bằng câu lệnh:

“pip install cmake”

“pip install dlib”

- Lưu ý với hệ điều hành window sau khi cài cmake, ta download gói dlib về tùy vào phiên bản python, do dlib cài trực tiếp rất khó trên môi trường window.
- Ta download những file trên tại link:
“https://github.com/datamagic2020/Install-dlib/blob/main/dlib-19.22.99-cp310-cp310-win_amd64.whl”



Hình 3.11 : Các file cài đặt dlib cho window tùy phiên bản

- Nếu muốn cài đặt dlib cho python version 3.8, ta down file “dlib-19.19.0-cp38-cp38-win_amd64.whl ” về chung thư mục hoặc 1 thư mục nào đó để nhớ path tùy mỗi người.
- Sau đây cài đặt bằng cách:

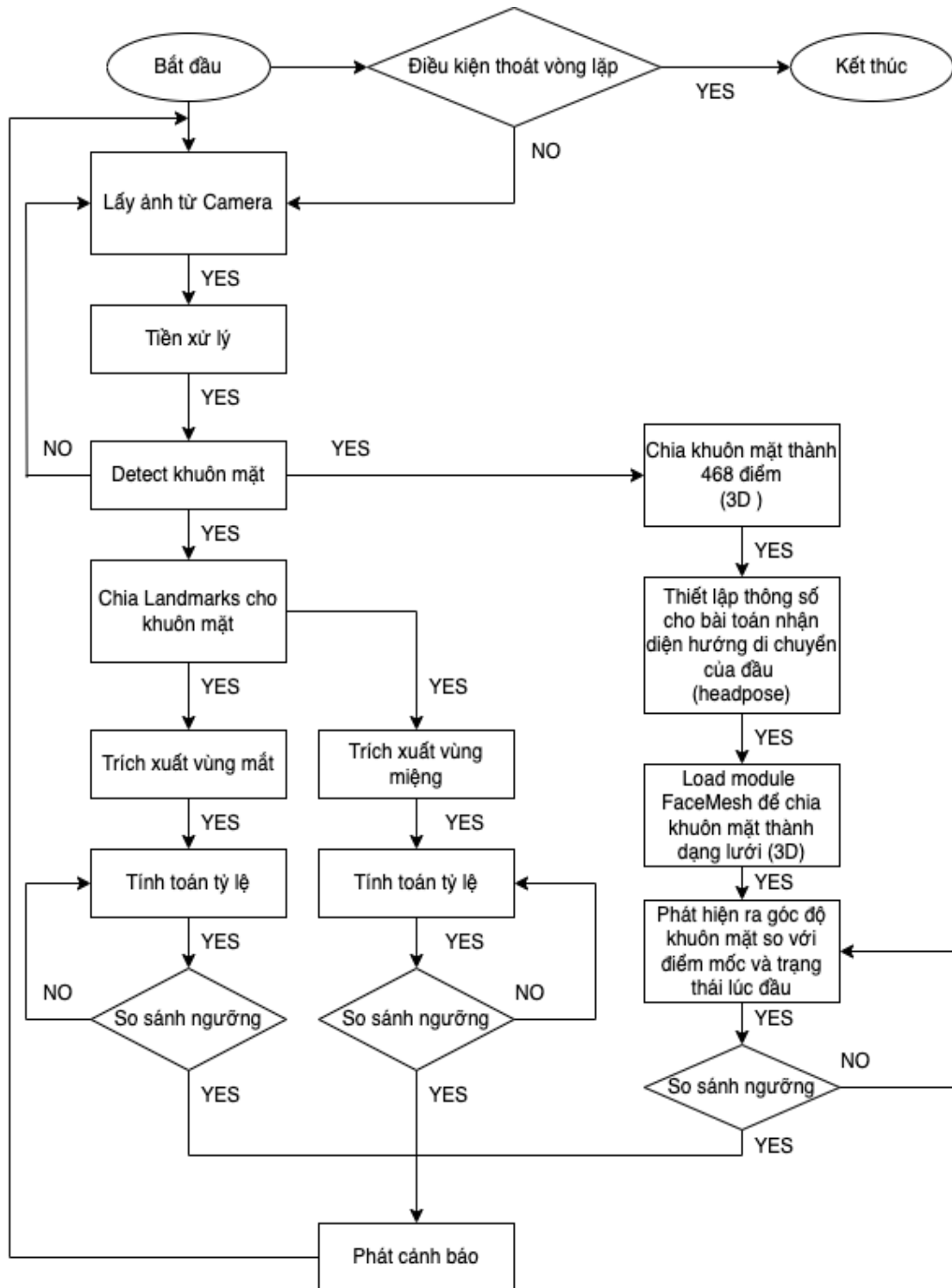
“pip install “đường dẫn đến file đã tải””
- Tiếp theo, là cài thư viện Mediapipe:

“pip install mediapipe”
- Đối với những thư viện kèm theo, ta cứ dùng công cụ pip mà cài đặt.
- Để kiểm tra những thư viện đã cài đặt, ta gõ lệnh:

“pip list”

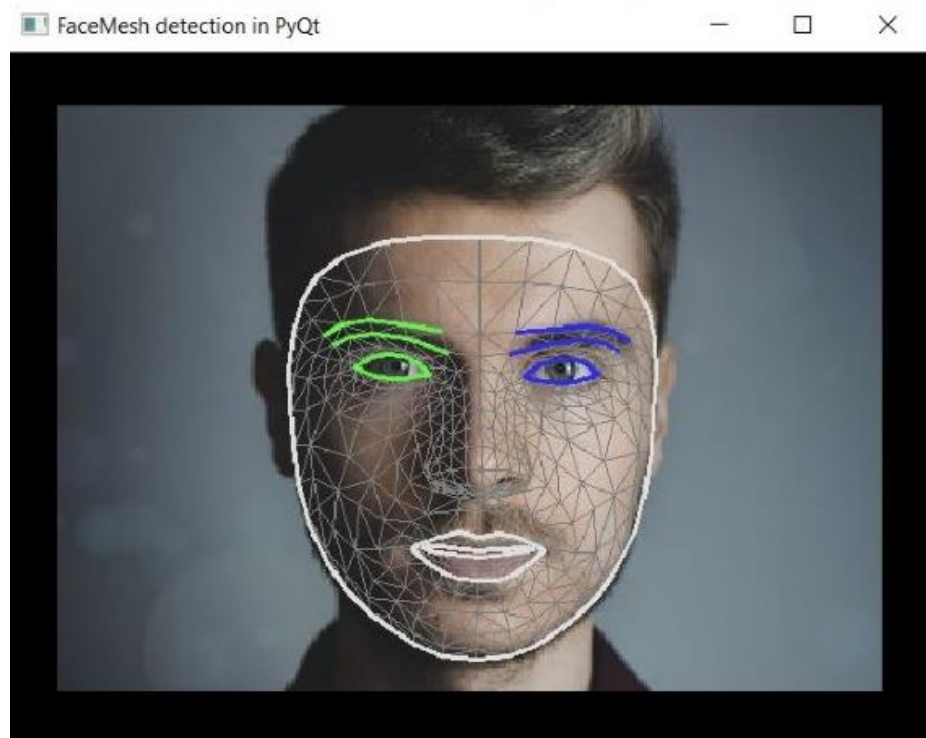
CHƯƠNG 4. THI CÔNG HỆ THỐNG

4.1 Flowchart hệ thống:



4.2 Giải thích các khối chính:

- Đầu tiên, để truy cập camera, ta ứng dụng thư viện imutils, chương trình sẽ thiết lập kết nối camera, lấy từng frame ảnh để tiến hành xử lý.
- Sau đến quá trình tiền xử lý, ta tiến hành liên tục (lặp vô hạn) các frame nhận được, sau đó tiến hành:
 - Tăng độ sáng, độ tương phản để bước nhận diện được trơn tru hơn
 - Giảm kích thước ảnh => Để tốc độ xử lý được tốt hơn (ngay cả cho những thiết bị xử lý yếu như raspberry pi).
 - Do màu chuẩn của thư viện opencv là BGR và bộ nhận diện ảnh của mediapipe là RGB, của dlib là ảnh xám nên ta cần qua bước chuyển đổi RGB sang 2 dạng còn lại.
- Bước chia landmarks cho mặt và tính toán tỷ lệ đã trình bày ở thư viện dlib.
- Tiếp theo, là bước nhận diện hướng di chuyển của đầu, ta ứng dụng mô hình khuôn mặt BlazeFace.



Hình 4.1 : Mô hình khuôn mặt BlazeFace^[6]

- Thư viện mediapipe chia khuôn mặt thành 468 điểm, sao đó nối thành 1 mạng (mesh). Từ mạng facemesh này, ta có 1 tập hợp có thể mô phỏng bề mặt trước của khuôn mặt.
- Để tính toán tư thế 3D của khuôn mặt trong ảnh ứng dụng thư viện trên ta cần thông tin sau:
 - Thứ nhất, ta cần thông tin tọa độ 2D (x,y) của các điểm đặc trưng. Đối với mô hình trên, ta chọn khoé mắt, đầu mũi, khoé miệng,.....Dlib cung cấp cho ta bộ dò đặc trưng để phát hiện những vị trí này.
 - Thứ hai, ta cần thông tin tọa độ 3D của các điểm trên gồm: đầu mũi, cằm, góc trái mắt trái, góc phải mắt phải, góc trái miệng, góc phải miệng.
- Trong trường hợp sử dụng FaceMesh để trích xuất khuôn mặt ta quan tâm 6 điểm chính^[4]:
 - Số 33: Đuôi mắt trái
 - Số 263: Đuôi mắt phải
 - Số 1: Chóp mũi
 - Số 61 và 291: 2 bên khoé miệng
 - Số 199: Chính giữa cằm

=>Tìm ra tọa độ x,y và z
- Ta quy đổi x,y và z tìm được bởi thư viện sang độ:
 - $x = \text{angles}[0] * 360$
 - $y = \text{angles}[1] * 360$
 - $z = \text{angles}[2] * 360$
- Ngoài ra, do tọa độ được quy đổi từ 2D(x,y) sang 3D(mô hình faceMesh), nên ta đặt điều kiện cho x,y để phát hiện các góc mong muốn.
- Nếu góc của $y < 10$, $y > 10$ hoặc $x < -10$, $x > 10$.
- Có nghĩa đầu tài xế đang tiến tiến sang các phía $x+x'$, $y+y'$.
- Do vậy, tài xế đang không nhìn thẳng mà đang mất tập trung.

- Khối cảnh báo: ta sẽ phát 2 loại cảnh báo thứ nhất là trên màn hình sẽ có dòng chữ nhắc nhở , thứ 2 là hàm tạo ra âm thanh sẽ đề cập ở chương sau.

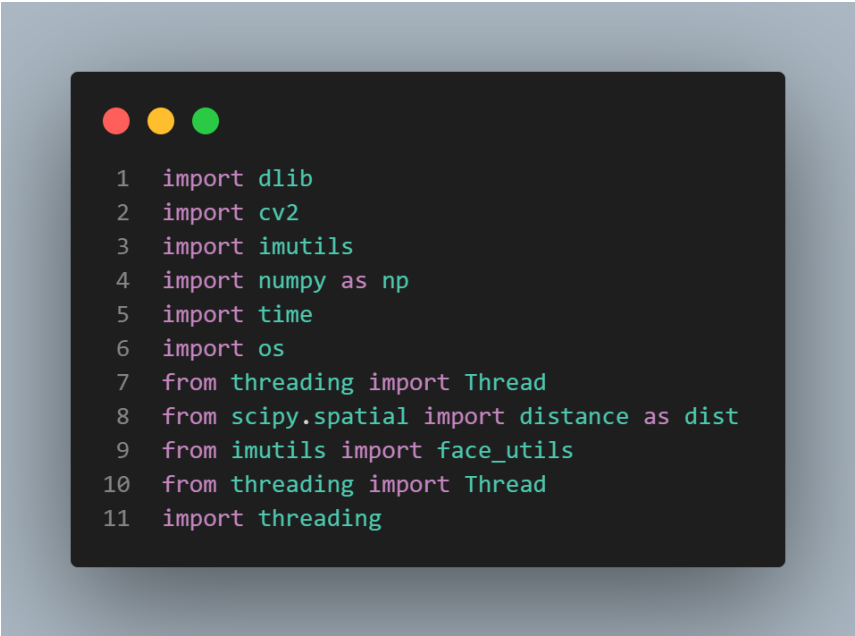
CHƯƠNG 5. KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

5.1 Module 1: Phát hiện độ đóng mở của mắt và miệng:

5.1.1 Phát hiện độ đóng mở của mắt^[26]:

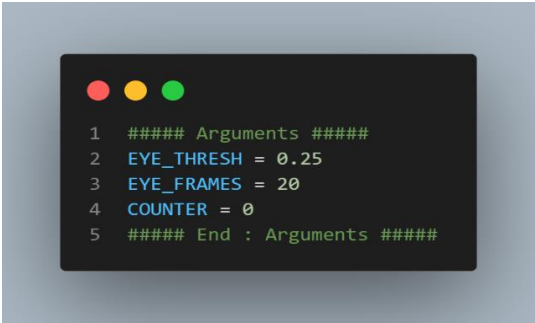
5.1.1.1 Khai báo thư viện và tham số

- Đầu tiên, ta khai báo các thư viện cần thiết.



```
1 import dlib
2 import cv2
3 import imutils
4 import numpy as np
5 import time
6 import os
7 from threading import Thread
8 from scipy.spatial import distance as dist
9 from imutils import face_utils
10 from threading import Thread
11 import threading
```

- Tiếp theo, ta khai báo các giá trị tham số cần có trong bài.
- Trong đó:
 - EYE_THRESH là ngưỡng mắt để so sánh.
 - EYE_FRAMES là khung hình tài xế ngủ gật.
 - COUNTER dùng để đếm khung hình tài xế nhắm mắt từ đó xét theo điều kiện.



```
1 ##### Arguments #####
2 EYE_THRESH = 0.25
3 EYE_FRAMES = 20
4 COUNTER = 0
5 ##### End : Arguments #####
```

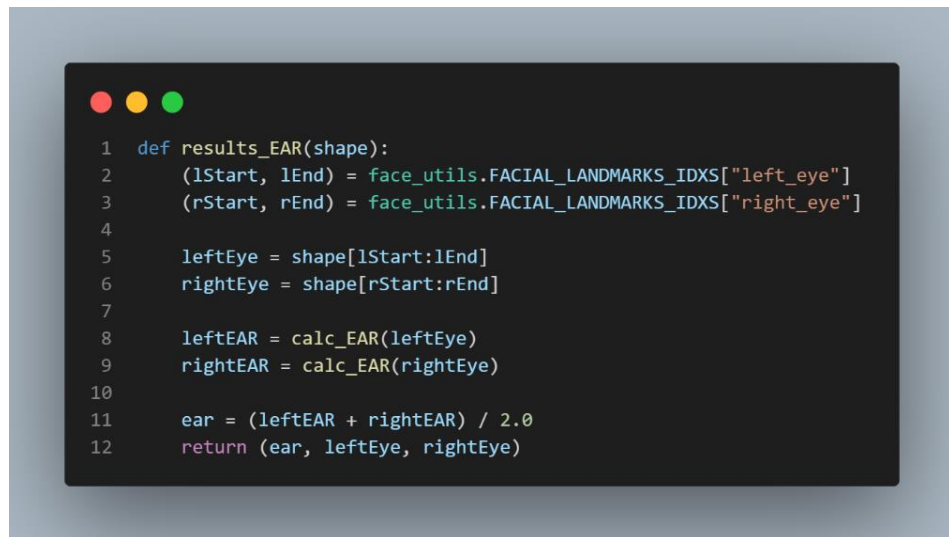
5.1.1.2 Khai báo hàm:

- Ta khai báo các hàm:
 - Hàm `calc_EAR` dùng để tính tỷ lệ của mắt (Eye Aspect Ratio).
 - Hàm này dùng Module **distance** của thư viện **Scipy** (“Sign Pi”)– Extension của Numpy, để tính khoảng cách của mắt (điểm trên và dưới)
 - Hàm này trả về tỷ lệ “ear” là kết quả sau khi tính toán.

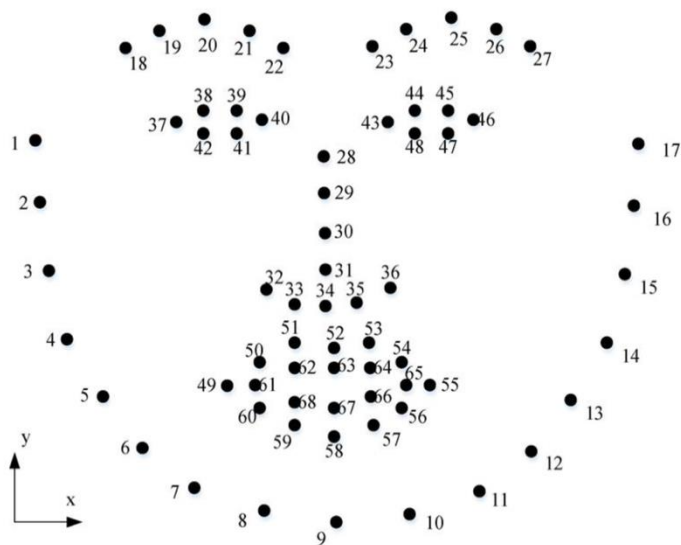


```
1 ##### Functions #####
2 def calc_EAR(eye):
3     A = dist.euclidean(eye[1], eye[5])
4     B = dist.euclidean(eye[2], eye[4])
5     C = dist.euclidean(eye[0], eye[3])
6     ear = (A + B) / (2.0 * C)
7     return ear
```

- Tiếp theo, ta tính toán tỷ lệ “ear” trung bình và của mắt trái, mắt phải .
- Ứng dụng Module **face_utils** của thư viện **imutils** và module **FACIAL_LANDMARKS_IDXS** (cũ là 68 điểm) để lấy ID điểm vùng mắt trái và phải .
- Ta gán những điểm đó thành dạng List bằng phương pháp slicing (lấy nhiều phần tử 1 lúc).
- Sau đó, ta gọi hàm `calc_EAR` để tính toán tỷ lệ cho mắt trái và mắt phải.
- Hàm này trả về “ear trung bình”, tỷ lệ mắt trái và phải.



- Tiếp đến là khối lệnh hàm main.
- Đầu tiên, hàm sẽ khởi tạo 2 bộ gồm bộ nhận diện (detector) và bộ dự đoán (predictor)
 - Bộ nhận diện: ta sử dụng thư viện xử lý ảnh open-cv để load model haarcascade từ đó nhận diện được khuôn mặt trong ảnh , video hoặc camera.
 - Bộ dự đoán: ta sử dụng thư viện Dlib để load file shape_predictor từ đó chia khuôn mặt đã phát hiện được bên trên thành 68 điểm.



Hình 5.1: 68 điểm landmarks từ dlib^[8]

- Để load file cần nhận diện, ta khởi tạo biến:
`cap = cv2.VideoCapture()`
- Với tham số bên trong VideoCapture là 1 đường dẫn file ảnh, file video hoặc thời gian thực qua webcam như bài nghiên cứu. Ta chọn giá trị 0 để khởi chạy camera, nếu có camera thứ 2 ta chọn tham số 1, tương tự thứ n ta chọn tham số n+1.
- Tiếp đến, để mở camera, ta cho chạy vòng lặp While, đến khi nào nhận được điều kiện thì thoát vòng lặp.
- Ret, frame là 2 biến được khởi tạo qua cap.read(), trong đó:
 - Ret là biến trả về giá trị bool (true hoặc false) chỉ kết nối đến camera thành công hay không, thành công sẽ trả về kết quả true.
 - Frame là khung hình được khởi tạo.
- Gray là biến xám, mục đích của biến này là chuyển khung hình màu thành khung hình xám, qua đó việc xử lý hình ảnh sẽ nhẹ nhàng hơn. Lưu ý, do opencv chỉ đọc hệ màu BGR nên mới dùng **BGR2GRAY**. Đa số các ứng dụng khác đều dùng hệ màu RGB. OpenCV dùng hệ màu này do họ nhận thấy đa phần các nhà sản xuất máy ảnh và cung cấp phần mềm đều dùng đến hệ màu này.
- Rects là biến dùng để phát hiện khuôn mặt và tạo hình chữ nhật quanh nó.
- Ta dùng một vòng lặp for để hiển thị khung phát hiện khuôn mặt này, trong đây gồm 4 tham số: x, y, w, h.
- Ta tạo một biến shape để lưu khung phát hiện với ảnh đã chuyển sang ảnh xám.
- Biến shape này được chuyển sang dạng mảng.
- Từ dạng mảng bên trên ta dùng hàm results_EAR để lấy ra thông số tỷ lệ cần.
- Sau đây, là điều kiện để phát hiện ngủ gật. Nếu khung hình nhắm mắt có tỷ lệ EAR dưới tỷ lệ ngưỡng đã đặt, ta cho biến COUNTER đếm, nếu COUNTER đếm được 1 ngưỡng khung hình ta đã đặt có nghĩa tài xế đang có xu thế nhắm mắt lâu => tài xế đang ngủ gật và phát cảnh báo.

```
1 def main():
2     detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
3     predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
4
5     cap = cv2.VideoCapture(0)
6     while cap.isOpened() :
7         ret, frame = cap.read()
8         frame = imutils.resize(frame, width=500)
9
10        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
11
12        rects = detector.detectMultiScale(gray, scaleFactor=1.1,
13        minNeighbors=5, minSize=(30, 30),
14        flags=cv2.CASCADE_SCALE_IMAGE)
15
16        for (x, y, w, h) in rects:
17            rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))
18            shape = predictor(gray, rect)
19            shape = face_utils.shape_to_np(shape)
20
21            eye = results_EAR(shape)
22            ear = eye[0]
23            leftEye = eye [1]
24            rightEye = eye[2]
25
26            leftEyeHull = cv2.convexHull(leftEye)
27            rightEyeHull = cv2.convexHull(rightEye)
28
29
30            if ear <= EYE_THRESH:
31                COUNTER += 1
32                print(COUNTER)
33                if COUNTER >= EYE_FRAMES:
34                    print('Warning Sir')
35                    t1=threading.Thread(target=alarm, args=('wake up sir',))
36                    # alarm('Wake up sir')
37                    t1.start()
38                    t1.join()
39                    cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
40                                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
41                    cv2.drawContours(frame, [leftEyeHull], -1, (0, 0, 255), 1)
42                    cv2.drawContours(frame, [rightEyeHull], -1, (0, 0, 255), 1)
43            else:
44                COUNTER = 0
45                alarm_status = False
46                cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
47                cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)
48                cv2.putText(frame, "EAR: {:.2f}".format(ear), (380, 30),
49                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
50
51
52            cv2.imshow("Detect Driver State : EAR", frame)
53            if cv2.waitKey(10) & 0xFF == ord('q') :
54                break
55        cap.release()
56        cv2.destroyAllWindows()
57
58 ##### End Functions #####
```

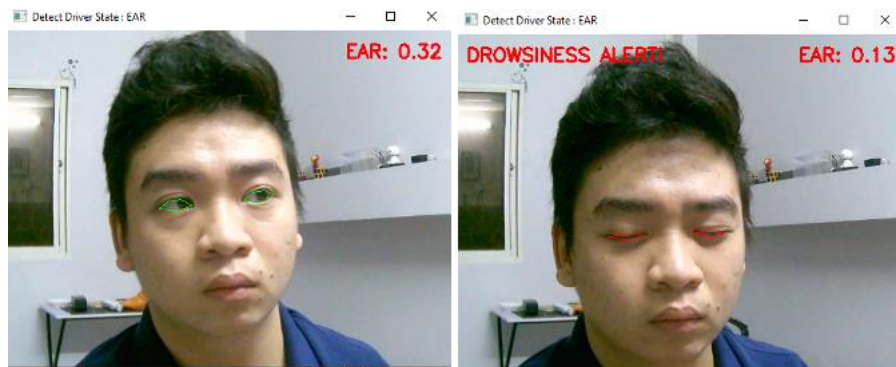
5.1.1.3 Chạy chương trình:

- Để chạy chương trình, ta chạy dòng lệnh sau:

If __name__ == "__main__":

main()

5.1.1.4 Kết quả:



Hình 5.2: Kết quả đo tỷ lệ mắt và xuất cảnh báo buồn ngủ khi độ đóng mở mắt đạt ngưỡng.

5.1.1.5 Bảng thử nghiệm:

Trạng thái(state)	Nhắm mắt
Số lần thử	50
Số lần nhận diện được	44/50
Tỉ lệ chính xác	88%

5.1.1.6 Nhận xét:

- Ưu điểm:
 - Trong điều kiện ánh sáng ổn định kết quả có độ chính xác cao.
 - Phát cảnh báo bằng âm thanh nên tài xế có thể bị đánh thức.

- **Nhược điểm:**

- Hoạt động không chính xác trong điều kiện tối , phụ thuộc vào ánh sáng tự nhiên quá nhiều.
- Không thể phát hiện nếu mắt bị che (đeo kính đen, trời quá tối ,....).
- Nếu khuôn mặt nằm ngoài tầm quét thì sẽ không phát hiện được mắt.

5.1.2 Phát hiện độ đóng mở của miệng:

5.1.2.1 Khai báo thư viện và tham số:

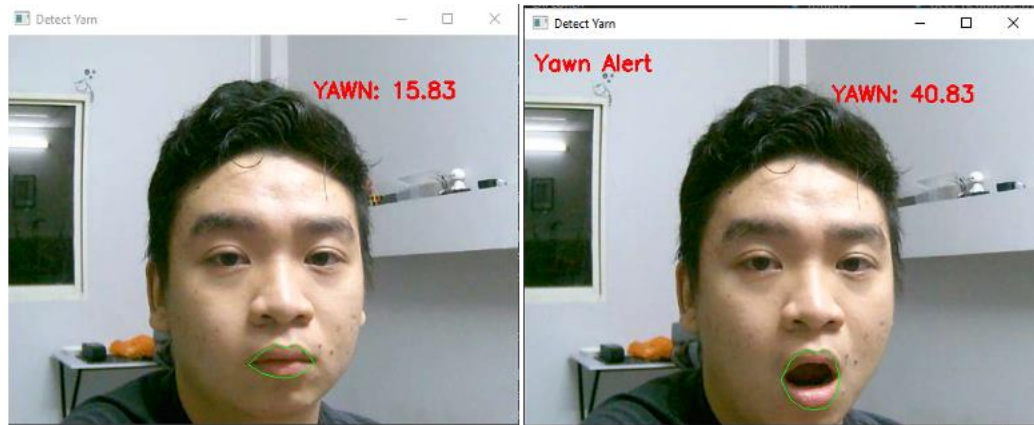
- Do quy tắc phát hiện độ đóng mở của miệng giống như của mắt đều dùng Dlib để phát hiện cụm điểm trong tổng thể 68 điểm nên việc khai báo thư viện là như nhau.
- Ta chỉ thêm ngưỡng phát hiện ngáp: YARN_THRESH = 25.

5.1.2.2 Khai báo hàm:

```
1 def lip_distance(shape):
2     top_lip = shape[50:53]
3     top_lip = np.concatenate((top_lip, shape[61:64]))
4
5     low_lip = shape[56:59]
6     low_lip = np.concatenate((low_lip, shape[65:68]))
7
8     top_mean = np.mean(top_lip, axis=0)
9     low_mean = np.mean(low_lip, axis=0)
10
11     distance = abs(top_mean[1] - low_mean[1])
12     return distance
```

```
1 def main():
2     detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
3     predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')
4     cap = cv2.VideoCapture(0)
5     while cap.isOpened():
6         ret, frame = cap.read()
7         frame = imutils.resize(frame, width=500)
8
9         gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
10
11        rects = detector.detectMultiScale(gray, scaleFactor=1.1,
12        minNeighbors=5, minSize=(30, 30),
13        flags=cv2.CASCADE_SCALE_IMAGE)
14        #for rect in rects:
15        for (x, y, w, h) in rects:
16            rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))
17
18            shape = predictor(gray, rect)
19            shape = face_utils.shape_to_np(shape)
20            distance = lip_distance(shape)
21            lip = shape[48:60]
22            cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)
23
24            if (distance > YAWN_THRESH):
25                cv2.putText(frame, "Yawn Alert", (10, 30),
26                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
27                if alarm_status2 == False :
28                    alarm_status2 = True
29                    t2 = Thread(target=alarm, args=('take some fresh air sir',))
30                    t2.start()
31                    t2.join
32            else:
33                alarm_status2 = False
34
35            cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
36            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
37
38            cv2.imshow("Detect Yawn", frame)
39            if cv2.waitKey(10) & 0xFF == ord('q') :
40                break
41        cap.release()
42    cv2.destroyAllWindows()
```


5.1.2.3 Kết quả:



Hình 5.3: Kết quả đo tỷ lệ miệng và xuất cảnh báo ngáp khi độ đóng mở miệng đạt ngưỡng.

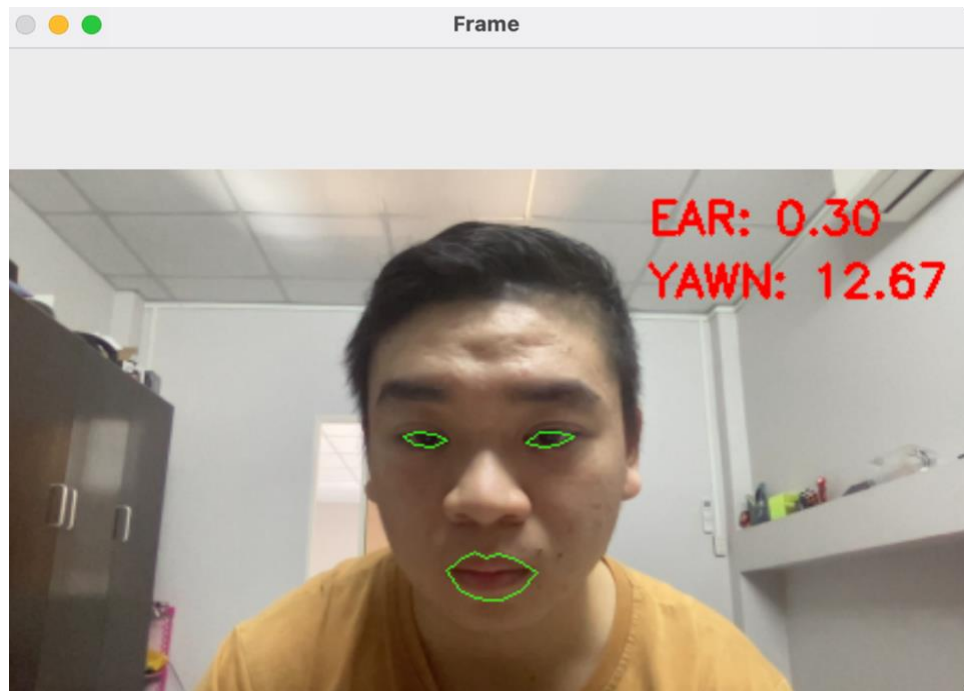
5.1.2.4 Bảng thử nghiệm:

Trạng thái(state)	Ngáp
Số lần thử	50
Số lần nhận diện được	48/50
Tỉ lệ chính xác	96%

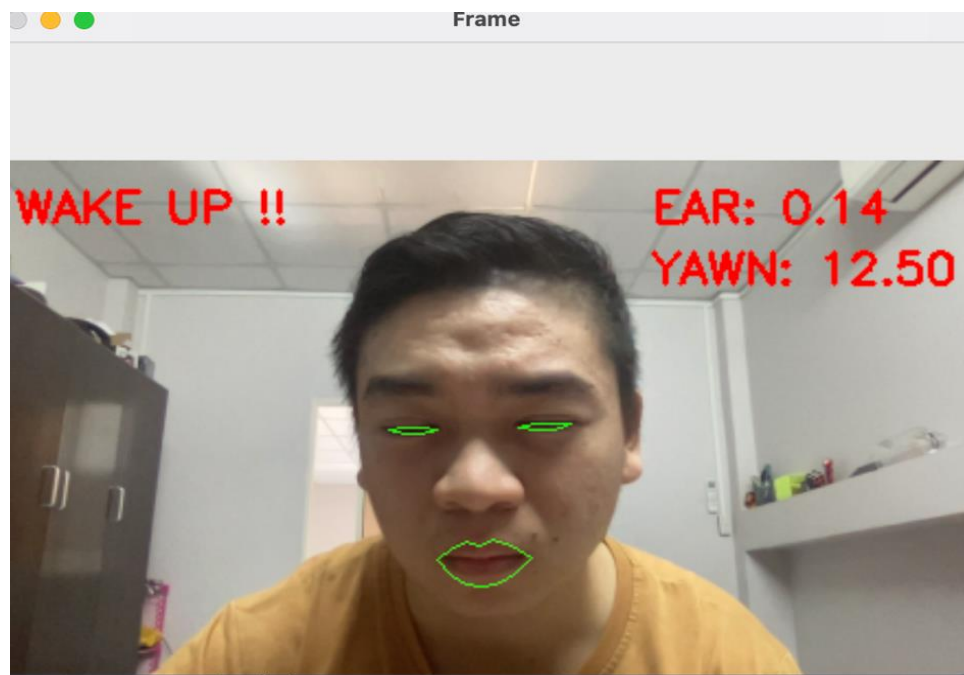
5.1.2.5 Nhận xét:

- Ưu điểm:
 - Phát hiện được hành vi ngáp của tài xế.
 - Tăng độ chính xác cho mô hình.
- Nhược điểm:
 - Nếu miệng bị che (đeo khẩu trang , ...) thì sẽ không phát hiện được.

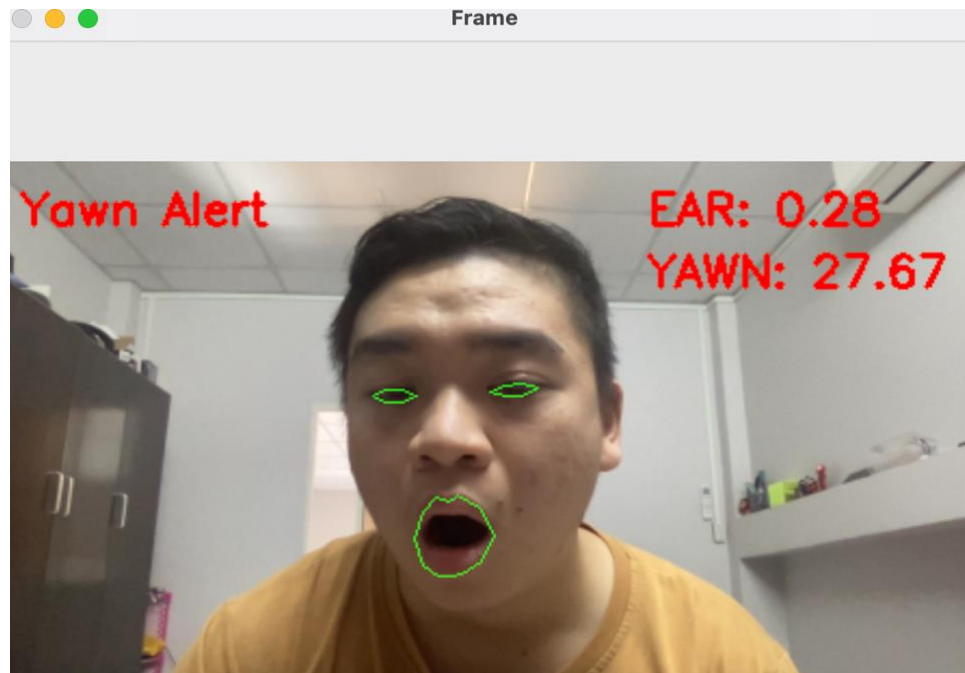
5.1.3 Kết hợp 2 mô hình trên:



Hình 5.4 : Hệ thống update khi có thêm phát hiện ngáp , vẽ đường viền quanh mắt và miệng



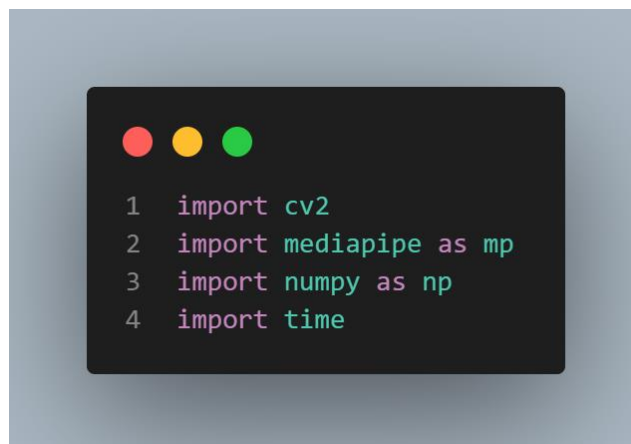
Hình 5.5 : Khi ngưỡng mắt thấp trong khoảng thời gian đặt , cảnh báo buồn ngủ



Hình 5.3 : Cảnh báo ngáp

5.2 Module 2: Phát hiện hướng di chuyển của mặt (3D) để phát hiện độ tập trung của tài xế:

5.2.1 Khai báo thư viện và tham số:



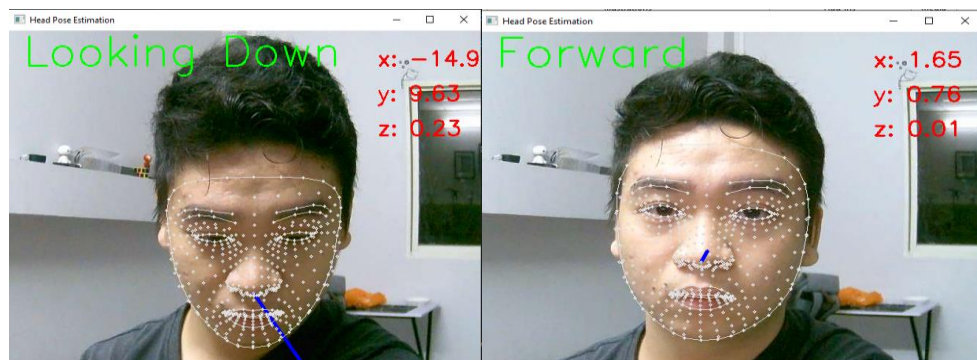
5.2.2 Khởi hàm chính:

- Ta có hàm main là hàm chính.
- Đầu tiên ta cần khởi chạy model từ module FaceMesh của thư viện Mediapipe.

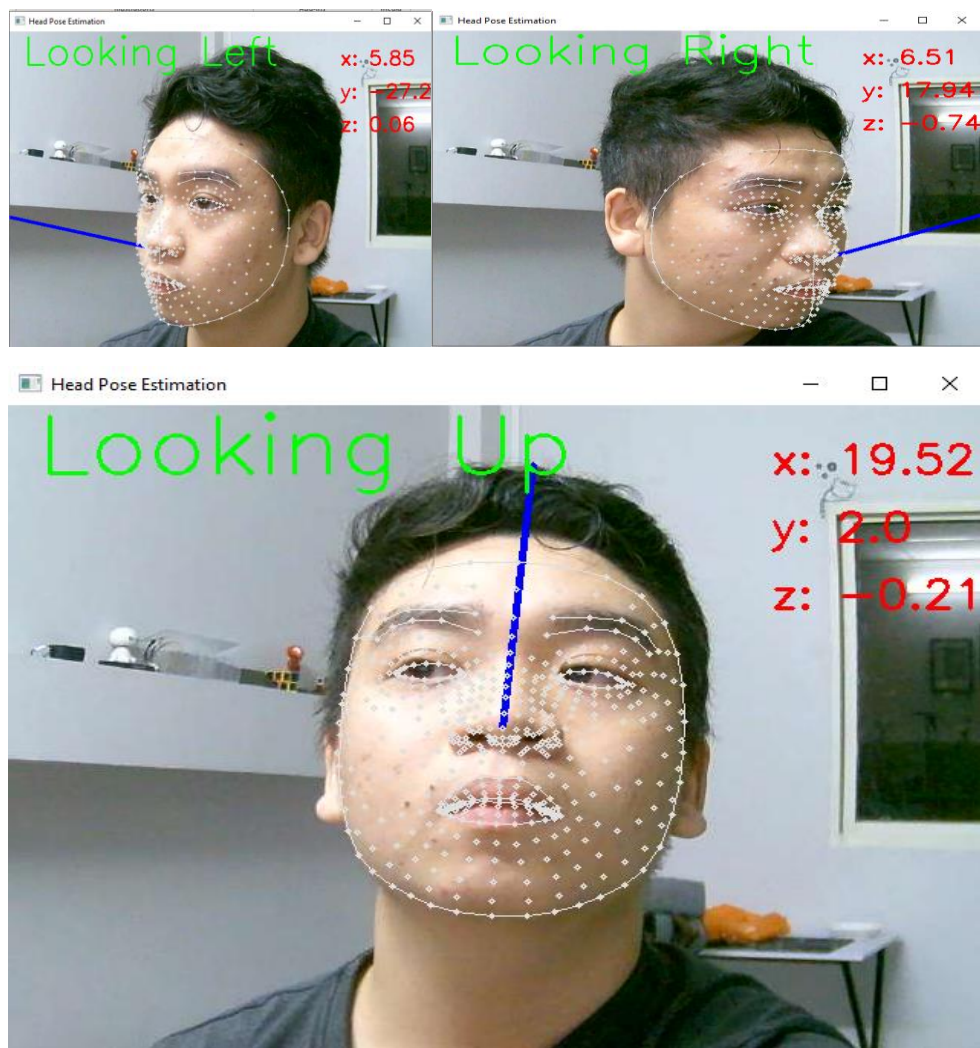
```
1 mp_face_mesh = mp.solutions.face_mesh
2 face_mesh = mp_face_mesh.FaceMesh(min_detection_confidence=0.5, min_tracking_confidence=0.5)
3 mp_drawing = mp.solutions.drawing_utils
4 drawing_spec = mp_drawing.DrawingSpec(thickness=1, circle_radius=1)
```

- Sau đây, ta cho chạy vòng lặp While để khởi chạy camera.
- Một vật thể 3D, có 2 loại chuyển động chính đối với máy ảnh:
 - Transition(dịch): di chuyển từ điểm x, y, z cũ sang tọa độ $x+x', y+y', z+z'$; mới. Được biểu diễn bằng vector $t(x-x', y-y', z-z')$
 - Rotation (quay): ta có thể xoay máy ảnh quanh trục x, y hoặc z . Do vậy, một chuyển động quay cũng có 3 bậc tự do.
- Để tính toán tư thế 3D của khuôn mặt, ta cần biết:
 - Tọa độ 2D(x, y) của các điểm đặc trưng.
 - Vị trí 3D của các điểm tương tự:
 - ❖ Góc trái của mắt trái(33)
 - ❖ Góc phải của mắt phải(263)
 - ❖ Cằm(199)
 - ❖ Đầu mũi(1)
 - ❖ Góc trái của miệng(61)
 - ❖ Góc phải của miệng(291)

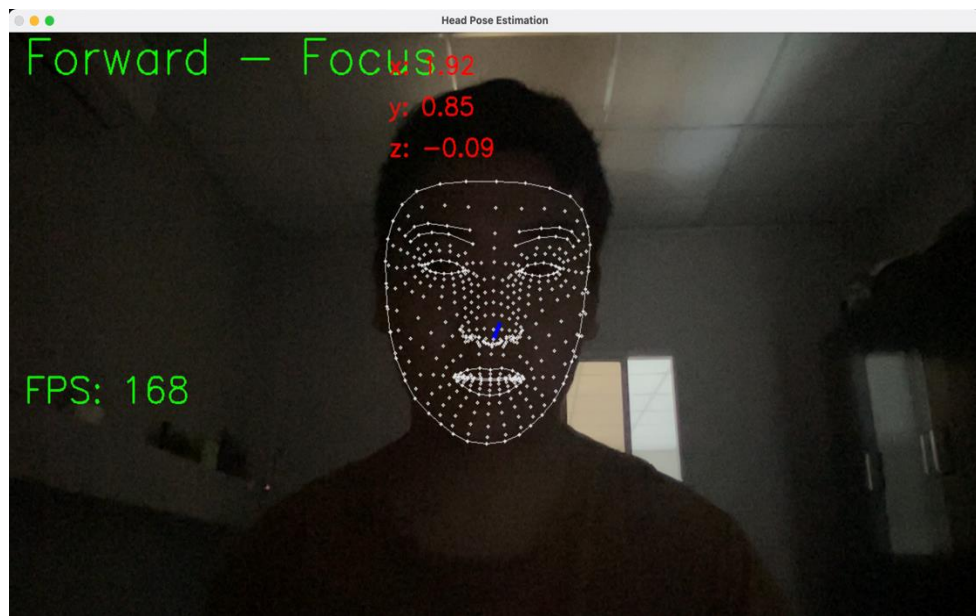
5.2.3 Kết quả thu được:



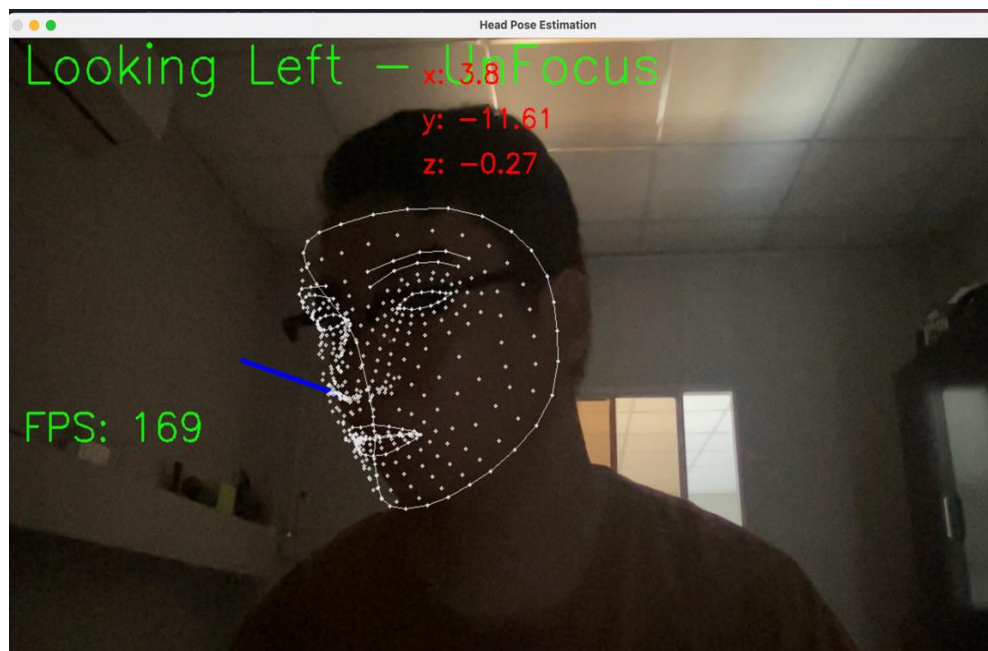
Hình 5.4: phát hiện hướng nhìn xuống và nhìn thẳng(ánh sáng đầy đủ)



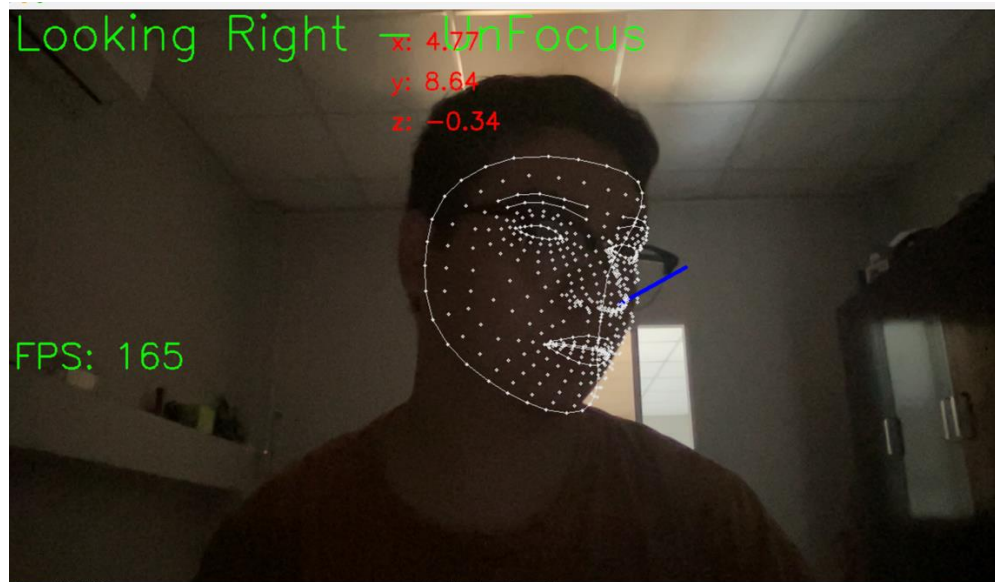
Hình 5.5 : Phát hiện được những tư thế còn lại(trường hợp ánh sáng đầy đủ)



Hình 5.6 : Tư thế đầu thẳng (tập trung) trong điều kiện thiếu sáng và có đeo kính



Hình 5.7 : Mắt tập trung khi nhìn sang trái lâu (có đeo kính)



Hình 5.8 : Mắt tập trung khi nhìn sang phải lâu (có đeo kính)

5.2.4 Bảng thử nghiệm:

5.2.4.1 Ánh sáng tốt:

Trạng thái (state)	Gục đầu
Số lần thử	50
Số lần nhận diện được	48/50
Tỉ lệ chính xác	96%

5.2.4.2 Ánh sáng kém, 1000 mẫu thử:

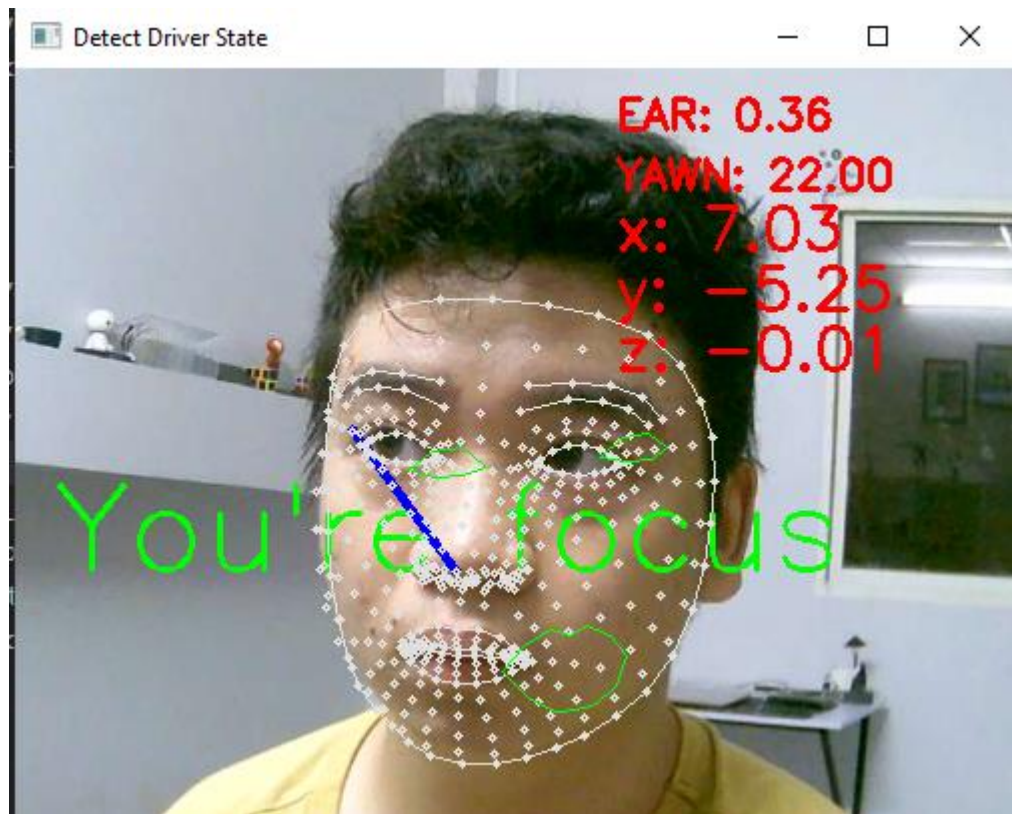
	Số khung hình phát hiện được tư thế đầu khi không đeo kính	Số khung hình phát hiện được tư thế đầu khi đeo kính
Ánh sáng đầy đủ	918/1000 = 91.8%	905/1000=90.5%
Thiếu sáng	888/1000 = 88.6%	778/1000=77.8%

5.2.4.3 Đánh giá:

- Phát hiện tư thế đầu hỗ trợ rất lớn cho mô hình trên.

- Qua đây, ta có thể đánh giá được độ tập trung của tài xế, cảnh báo họ khi thiếu sự tập trung.
- Hơn thế, mô hình hỗ trợ cho điểm yếu của phát hiện độ đóng mở mắt, miệng là khi thiếu ánh sáng hoạt động không tốt, riêng độ tập trung vẫn được kiểm tra trong trường hợp có đeo kính và ban đêm, cho kết quả >70%.

5.3 Module 3: Module hoàn chỉnh:



Hình 5.9 : Mô hình hoàn chỉnh kết hợp 3 mô hình trên

5.4 Module 4: Training 1 model phát hiện trạng thái tài xế đang tỉnh táo hay buồn ngủ ứng dụng YOLOv5^[19]:

- Đầu tiên, phải clone mô hình yolov5 đã train sẵn từ nguồn^[9]:
<https://github.com/ultralytics/yolov5>

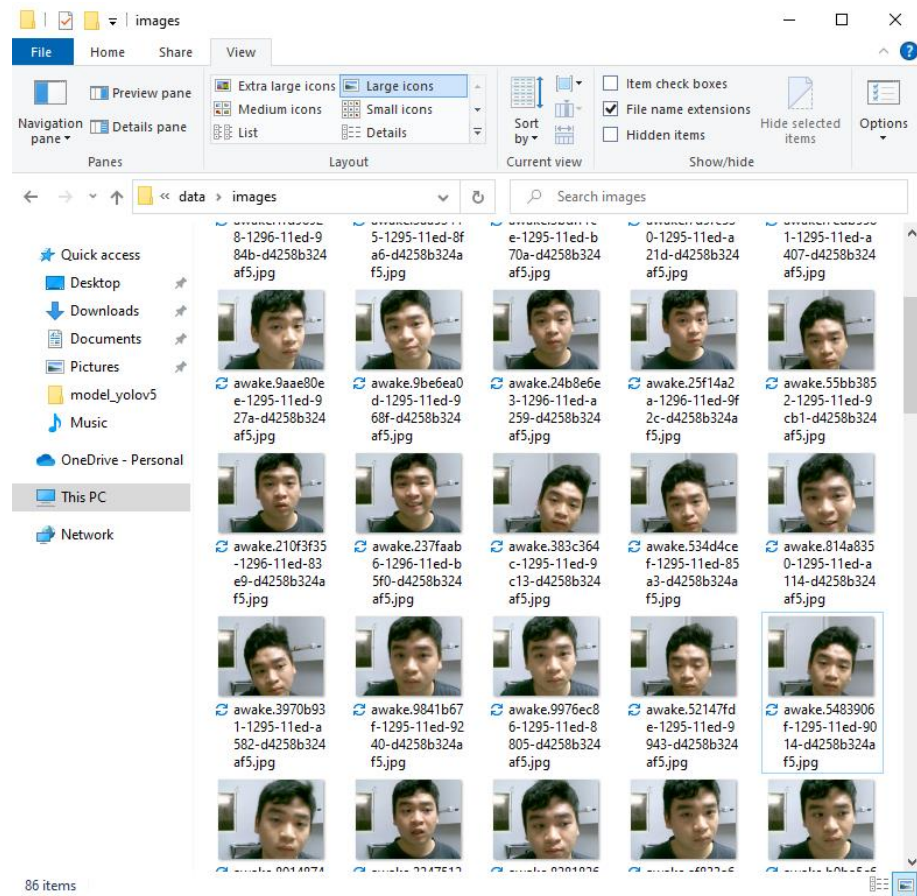

```
git clone https://github.com/ultralytics/yolov5 # clone
cd yolov5
pip install -r requirements.txt # install
```

- Những class mà model yolov5 đã được train sẵn để nhận biết^[10]:
<https://gist.github.com/AruniRC/7b3dadd004da04c80198557db5da4bda>
- Gồm 81 class, riêng con người nằm ở class thứ 2.
- Bước tiếp theo ta lấy mẫu để train bằng file collector.py.



```
1 import uuid
2 import cv2
3 import os
4 import time
5
6 IMAGES_PATH = os.path.join('data', 'images')
7 labels = ['awake', 'drowsy']
8 NUMBER_IMAGES = 10
9
10 cap = cv2.VideoCapture(0)
11 for label in labels :
12     print('Collecting Images {}'.format(label))
13     time.sleep(5)
14
15     for img_num in range(NUMBER_IMAGES) :
16         print('collecting images for {}, images number : {}'.format(label, img_num))
17
18         ret, frame = cap.read()
19
20         imgname = os.path.join(IMAGES_PATH, label+'.'+str(uuid.uuid1())+'.jpg')
21
22         #write out images to file
23         cv2.imwrite(imgname, frame)
24         cv2.imshow('Image Collector', frame)
25
26         # 2 second to delay
27         time.sleep(2)
28
29         if cv2.waitKey(10) & 0xFF == ord('q') :
30             break
31
32 cap.release()
33 cv2.destroyAllWindows()
```

- Hình ảnh được train vào thư mục data/images .
- Với bước lấy mẫu là 10 ảnh cho mỗi trạng thái
- Gồm 2 trạng thái là awake (tỉnh táo) , drowsy (buồn ngủ)



Hình 5.10 : Tập dataset đã thu thập

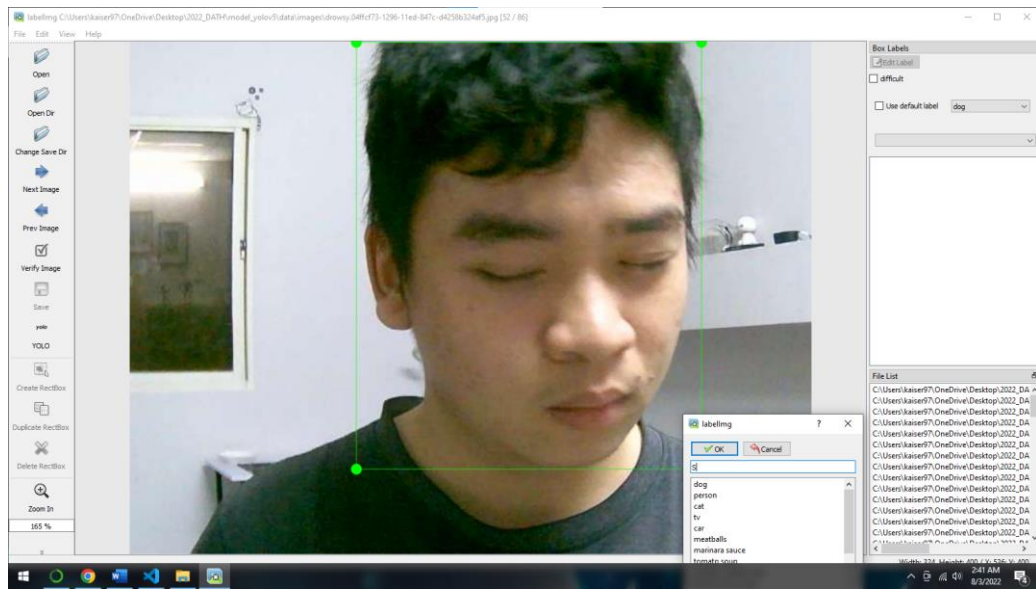
- Tiếp theo là bước đánh nhãn những mẫu ta vừa thu được :
- Ta lấy file đánh nhãn đã được viết từ link github sau đây :
<https://github.com/heartexlabs/labelImg.git>
- Để hiển thị file đánh nhãn ra giao diện đồ họa , ta cần cài thư viện pyqt5 theo cú pháp sau đây :

```
conda install pyqt=5  
conda install -c anaconda lxml  
pyrcc5 -o libs/resources.py resources.qrc
```

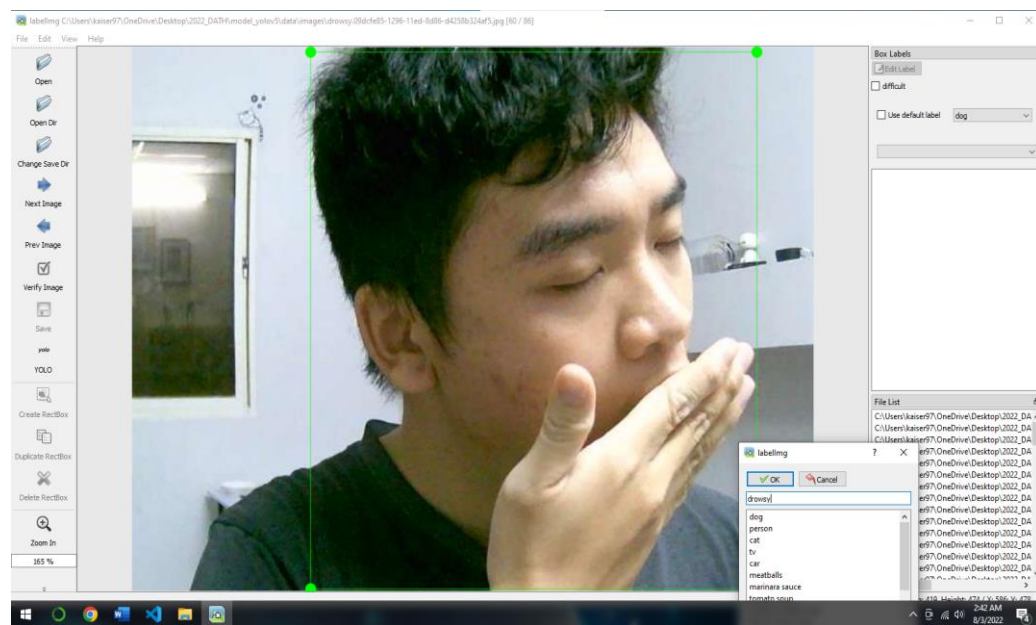
- Sau đó , ta tiến hành vào thư mục labelImg để khởi chạy file đánh nhãn :

```
pyrcc5 -o libs/resources.py resources.qrc  
python labelImg.py
```

- Trong thư mục data , ta tạo ra 1 thư mục labels dùng để đánh nhãn ảnh

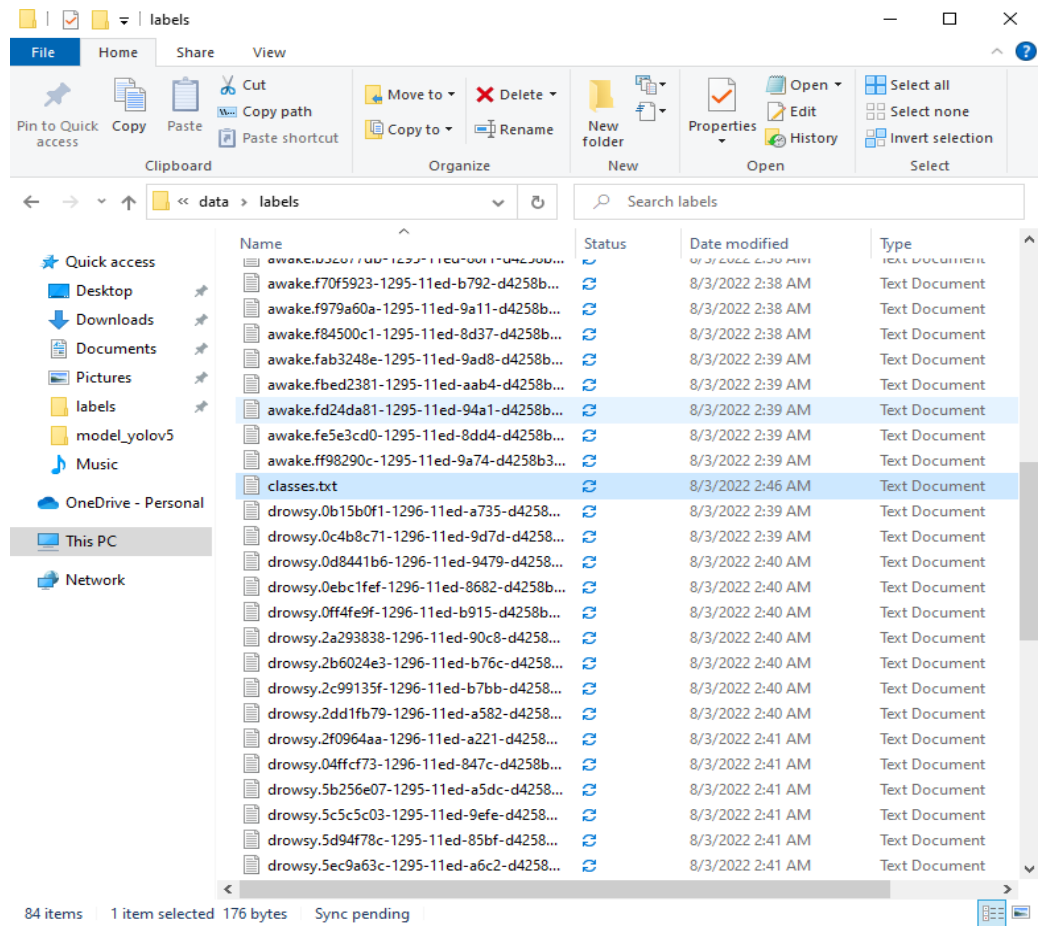


Hình 5.11: Quá trình đánh nhãn.



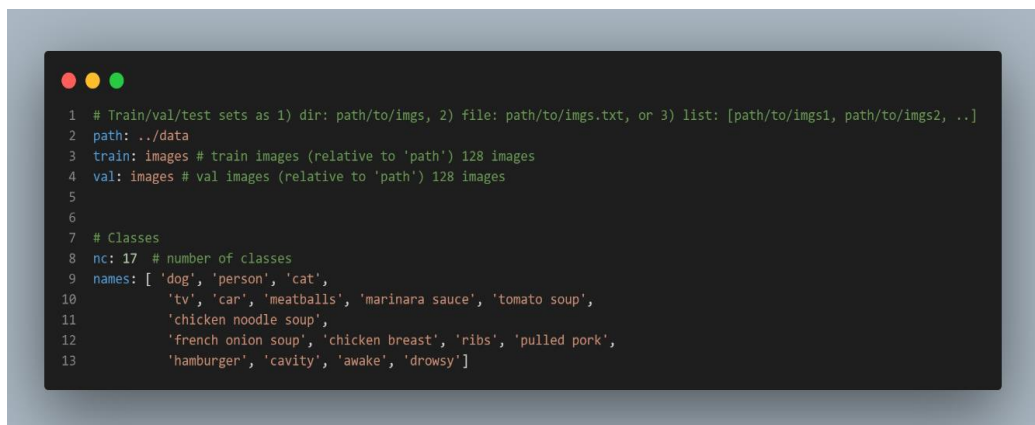
Hình 5.12: Hành động buồn ngủ được đánh nhãn.

- Ta được thành quả là file labels sau:



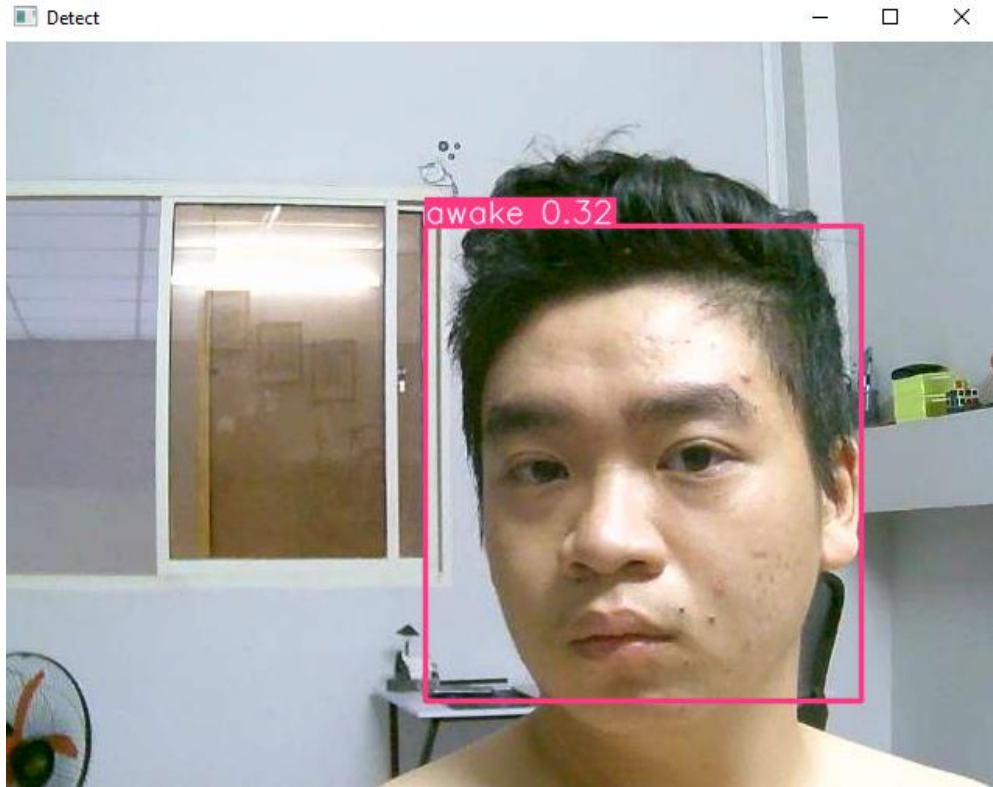
Hình 5.13: Tập tin sau khi được đánh nhãn.

- Tiếp đến, ta vào trang github của yolov5 trên để custom 1 model riêng.
- Tạo 1 file dataset.yml



- Chọn 1 model để train.
- Chạy lệnh:
“python train.py --img 320 --batch 16 --epochs 10 --data dataset.yaml --weights yolov5s.pt --workers 2”

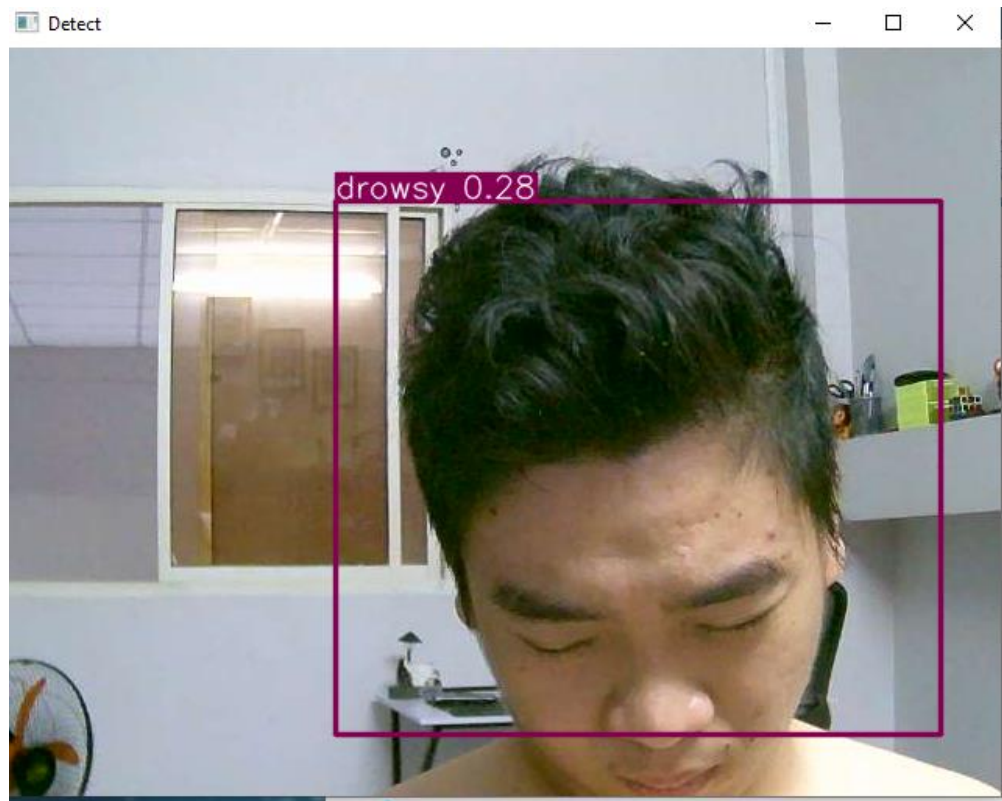
- Kết quả thu được sau khi chạy 50 epochs:



Hình 5.14: Model dự đoán lúc tỉnh táo



Hình 5.15: Model dự đoán hành vi buồn ngủ (1)



Hình 5.16: Model dự đoán hành vi buồn ngủ (2)

CHƯƠNG 6. KẾT LUẬN

6.1 Kết luận:

- Bài nghiên cứu đã giải quyết được phần lớn nhiệm vụ cơ bản được giao.
- Qua bài nghiên cứu, đã lĩnh hội được khá nhiều kiến thức bổ trợ về xử lý ảnh, từ đó có nguồn vốn kiến thức để tạo nên nhiều ứng dụng hoặc thuật toán hỗ trợ cho đời sống con người hơn.
- Khai thác mô hình mini PC, có bộ vi điều khiển mạnh có thể giúp ích cho việc giải quyết nhiều bài toán hơn sau này cũng như mở rộng nhiều dự án lớn cần vi xử lý mạnh mẽ.
- Chương trình được mô phỏng trên hệ điều hành MacOS chip M1, so sánh với phần mềm window thì có thể thấy được code chạy trơn tru ít gặp lỗi hoặc bị xung đột giữa các thư viện với nhau.
- Thực ra ta có thể cài hệ điều hành linux/ubuntu để mô hình được thuận lợi hơn nhưng chọn hệ điều hành window là vì tính đại trà mà mọi người đều dùng được và dễ tiếp cận.

6.1.1 Ưu điểm:

- Sử dụng tiết kiệm được chi phí hết mức có thể nhưng vẫn đảm bảo có một CPU mạnh mẽ để giải quyết các thuật toán.
- Mô hình có tính ứng dụng trong thực tế cao.
- Kết quả mô phỏng cho kết quả dự kiến cao.

6.1.2 Nhược điểm:

- Phụ thuộc quá nhiều vào điều kiện tự nhiên (độ sáng, tối, góc quay còn thiếu chính xác, ...ngoại cảnh).
- Phụ thuộc vào góc độ nếu quay khuôn mặt quá 45 độ cụ thể từ 50 đến 60 độ thì không bắt được FaceMesh dẫn đến không detect được khuôn mặt.

6.2 Hướng phát triển:1

- Cải thiện độ chính xác cho mô hình bằng những kiến thức sẽ bổ sung.
- Học hỏi thêm những thuật toán hay cách giải quyết mới vào mô hình hiện tại.
- Ứng dụng cho mô hình nhỏ gọn hơn.
- Viết chương trình chạy được trên mobile, gửi thông tin về gia đình nếu có sự kiện xảy ra.
- Xây dựng kết nối mạng cho mô hình để xử lý được tốt hơn.

TÀI LIỆU THAM KHẢO

- [1] Nguyễn Thanh Tuấn (2020), *Deep Learning Cơ Bản v2*.
- [2] Tutorial of OpenCV, https://docs.opencv.org/4.x/d9/df8/tutorial_root.html
- [3] Documentation of Numpy, <https://numpy.org/doc/stable/>
- [4] Documentation of Mediapipe, <https://google.github.io/mediapipe/>
- [5] Tutorial of OpenCV on GeeksForGeeks, <https://www.geeksforgeeks.org/opencv-python-tutorial/?ref=lbp>
- [6] Facial landmark make easy with Mediapipe,
<https://www.samproell.io/posts/yarppg/yarppg-face-detection-with-mediapipe/>
- [7] Giới thiệu về Xử lý ảnh, <https://nttuan8.com/bai-5-gioi-thieu-ve-xu-ly-anh/>
- [8] Trương Quốc Định và Nguyễn Đăng Quang (2015), *Hệ thống phát hiện tình trạng ngủ gật của lái xe*, *Tạp chí Khoa học Trường Đại Học Cần Thơ*.
- [8] https://www.researchgate.net/figure/The-face-shape-with-68-landmarks_fig3_330402357
- [9] Dự án yolov5, <https://github.com/ultralytics/yolov5>
- [10] Class trên yolov5,
<https://gist.github.com/AruniRC/7b3dadd004da04c80198557db5da4bda>
- [11] <https://github.com/heartexlabs/labelImg>
- [12] Nicholas Renotte channel, <https://www.youtube.com/c/NicholasRenotte>
- [13] Mì AI, <https://www.miai.vn/>
- [14] KIMe Lab channel, <https://www.youtube.com/c/KIMeLab>
- [15] Lato' channel, https://www.youtube.com/channel/UCDt2vE_57gsgC-xxxBGJDkQ
- [16] Nicolai Nielsen - Computer Vision & AI,
<https://www.youtube.com/c/NicolaiNielsenComputerVisionAI>
- [17] https://www.youtube.com/watch?v=NCae1MeHBBU&t=620s&ab_channel=ChandanShivashankara
- [18] https://www.youtube.com/watch?v=7WPdEajSL6c&ab_channel=ProgrammingHut
- [19] https://www.youtube.com/watch?v=tFNJGim3FXw&t=4023s&ab_channel=NicholasRenotte

- [20] <https://www.analyticsinsight.net/the-difference-between-artificial-intelligence-and-machine-learning/>
- [21] EPS Online Education channel, khoá học xử lý ảnh số, <https://www.youtube.com/c/EPSONlineEducation>
- [22] MS IT channel, xử lý ảnh số, https://www.youtube.com/channel/UCg_RXZ-3XfU4D5cy9y3raXQ
- [23] Son Nguyen Channel, khoá xử lý số, <https://www.youtube.com/c/SonNguyencover>
- [24] Một số bài báo trên Viblo, <https://viblo.asia/p/ai-vs-machine-learning-vs-deep-learning-1VgZvMDrKAw>
- [25] <https://onetech.vn/blog/haar-cascade-la-gi-13561>
- [26] <https://pyimagesearch.com/2017/04/24/eye-blink-detection-opencv-python-dlib/>
- [27] Nghiên cứu về tỷ lệ đóng mở mắt, <https://www.sciencedirect.com/science/article/pii/S2667241322000039#:~:text=Eye%20Aspect%20Ratio%20Eye%20Aspect,it%20into%20the%20following%20formula.>
- [28] <https://medium.com/analytics-vidhya/eye-aspect-ratio-ear-and-drowsiness-detector-using-dlib-a0b2c292d706>
- [29] <https://www.hrpub.org/download/20191230/UJEEEB9-14990984.pdf>
- [30] https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3356401
- [31] <https://github.com/topics/eye-aspect-rat>
