**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA ĐIỆN – ĐIỆN TỬ**


**PHẠM CHÍ THÀNH**

# XỬ LÍ ẢNH PHÁT HIỆN NGỦ GẬT CHO TÀI XẾ

## ĐỒ ÁN CHUYÊN NGÀNH

## KĨ THUẬT ĐIỆN TỬ - VIỄN THÔNG

**Người hướng dẫn**

**ThS. Trần Thành Nam**

**TP HỒ CHÍ MINH, NGÀY 03 THÁNG 12 NĂM 2022**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA ĐIỆN – ĐIỆN TỬ**


**PHẠM CHÍ THÀNH**

# XỬ LÍ ẢNH PHÁT HIỆN NGỦ GẬT CHO TÀI XẾ

## ĐỒ ÁN CHUYÊN NGÀNH

## KĨ THUẬT ĐIỆN TỬ - VIỄN THÔNG

**Người hướng dẫn**

**ThS. Trần Thành Nam**

**TP HỒ CHÍ MINH, NGÀY 03 THÁNG 12 NĂM 2022**

# LỜI CẢM ƠN

Em xin chân thành cảm ơn ThS. Trần Thành Nam đã giúp đỡ em trong quá trình thực hiện đồ án này. Em cũng xin cảm ơn Khoa Điện – Điện tử, Trường đại học Tôn Đức Thắng đã tạo điều kiện cho Em học tập và nghiên cứu để hoàn thành đồ án này.

TP. Hồ Chí Minh, ngày 03 tháng 12 năm 2022

Tác giả

Phạm Chí Thành

# 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 ThS. Trần Thành 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 nhúng 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 nhúng 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 12  năm 2022*

*Tác giả*

*Phạm Chí Thành*

TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG
KHOA ĐIỆN – ĐIỆN TỬ
BỘ MÔN ĐIỆN TỬ - VIỄN THÔNG

# PHIẾU THEO DÕI TIẾN ĐỘ THỰC HIỆN ĐỒ ÁN CHUYÊN NGÀNH

Họ tên sinh viên: Phạm Chí Thành MSSV: 41901040

Lớp: 19040201 Ngành: Kỹ thuật Điện tử - viễn thông

Họ tên GVHD: ThS. Trần Thành Nam

Tên đề tài: Xử lí ảnh phát hiện ngủ gật cho tài xế.

| **Tuần/Ngày** | **Khối lượng nội dung đã thực hiện (Đề nghị ghi chi tiết)** | **GVHD xác nhận** |
|---|---|---|
| **1 (26/9-2/10)** | - Nhận tờ nhiệm vụ Đồ án.<br>- Hiểu rõ yêu cầu nội dung đề tài, quy định thực hiện đồ án. | …./…./2022 |
| **2 (3/10-9/10)** | - Tìm hiểu Python, OpenCv môi trường lập trình<br>- Tìm hiểu về các hàm, thư viện liên quan.<br>- Tìm hiểu môi trường ảo anaconda để viết chương trình. | …./…./2022 |
| **3 (10/10-16/10)** | - Lập trình được chương trình phát hiện khuôn mặt.<br>- Lập trình chương trình lấy dữ liệu ảnh từ camera. | …./…./2022 |
| **4 (17/10-23/10)** | - Tìm hiểu về Facial Landmark.<br>- Lí thuyết về cách tính điểm đóng mở mắt.<br>- Lập trình chương trình phát hiện đóng mở mắt. | …./…./2022 |

| | | |
|---|---|---|
| **5**<br>**(24/10-30/10)** | - Tìm hiểu cách tính điểm đóng mở miệng.<br>- Lập trình chương trình đóng mở miệng. | …./…./2022 |
| | **GVHD đánh giá** khối lượng hoàn thành……..%<br>☐ Tiếp tục ☐ Không tiếp tục | …./…./2022 |
| **6**<br>**(31/11-07/11)** | **Báo cáo 50%** | …./…./2022 |
| **7**<br>**(08/11-14/11)** | - Kết hợp đóng và mở miệng thành một chương trình.<br>- Đánh giá<br>- Tối ưu chương trình | …./…./2022 |
| **8**<br>**(15/11-21/11)** | -Viết báo cáo | …./…./2022 |
| **9**<br>**(22/11-28/11)** | -Viết báo cáo | …./…./2022 |
| **10**<br>**(29/12-04/12)** | -Kiểm tra đạo văn<br>-Kiểm tra thuật toán, chương trình<br>-Tối ưu chương trình | …./…./2022 |
| | **GVHD** đã **duyệt** báo cáo đồ án, đề nghị:<br>☐ Được bảo vệ ☐ Không được bảo vệ | …./…./2022 |
| **11**<br>**05/12-07/12** | **Báo cáo 100%** | |

# XỬ LÍ ẢNH PHÁT HIỆN NGỦ GẬT CHO TÀI XẾ

## 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 vả 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

# DANH MỤC CÁC CHỮ VIẾT TẮT

EAR : EYE ASPECT RATIO

MAR: MOUNTH ASPECT RAITO

RGB: Red Green Blue

# CHƯƠNG 1 TỔNG QUAN VỀ ĐỀ TÀI

## 1.1 Lí do chọn đề tài

- Cùng với sự phát triển của đời sống, kinh tế - xã hội hiện đại nhu cầu vận tải hàng hóa đường dài ngày càng tăng. Cùng với đó, lịch trình di chuyển của tài xế cũng theo đó mà dày đặc hơn khiến cho thời gian nghỉ của tài xế ngắn lại. Điều đó vô tình dẫn đến một hệ lụy là số vụ tai nạn thương tâm do các tài xế ngủ gật trong quá trình vận chuyển cũng theo đó mà tăng lên.
- Nhằm đề ra một giải pháp cho vấn đề trên, phương pháp xây dựng một ứng dụng sử dụng xử lý ảnh để phát hiện tình trạng ngủ gật của tài xế được đề xuất. Đề xuất mang tính ứng dụng thực tiễn cao, hơn nữa trong quá trình nghiên cứu và phát triển sinh viên học hỏi và tiếp thu được kiến thức, kỹ năng và kinh nghiệm về xử lý ảnh số.
- Ngoài ra xử lí ảnh có nhiều ứng dụng rộng rãi trong đời sống và sản xuất

## 1.2 Mục tiêu và nội dung nghiên cứu

- Đây vẫn là một đề tài mới mẻ đối với chuyên ngành của sinh viên thực hiện nên trong quá trình thực hiện và triển khai sẽ gặp nhiều khó khăn và trở ngại phát sinh.

## 1.3 Mục tiêu

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.
- Viết báo cáo.

## 1.4 Giới hạn 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 Laptop cá nhân như một phần ví 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 , …). Cũng như là thiết bị có sẵn để có thể thực hiện và tiết kiệm chi phí đến mức tối đa.

### 1.5 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 Euclide, ...
- Chương 3: Thực hiện chương trình.
  • ở phần này tập trung vào việc xây dựng lưu đồ giải thuật và các cách cài đặt các thư viện và thực hiện những cài đặt liên quan cho mô hình để tiến hành lập trình.
- 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 GIỚI THIỆU VỀ XỬ LÍ ẢNH

## 2.1 Ảnh là gì ?

– Trước khi đi vào xử lí ảnh, thì đầu tiên chúng ta phải hiểu rõ thành phần nào cấu tạo thành ảnh. Hình ảnh được thể hiện qua hai kích thước cơ bản là chiều rộng và chiều cao dựa trên số lượng pixel (hay điểm ảnh). Ví dụ, nếu kích thước ảnh là 480 x 360 (rộng – cao), thì tổng số pixels của ảnh sẽ là 172,800.

## 2.2 Pixel là gì ?

– Vậy thì pixel (hay điểm ảnh) là một khối màu rất nhỏ và là đơn vị cơ bản nhất để tạo nên một bức ảnh.

– Vậy bức ảnh sẽ được biểu diễn theo dạng ma trận hàng nhân cột. Ví dụ để biểu diễn một ảnh có kích thước 480 x 360 thì ma trận biểu diễn sẽ là 360 x 480 (hàng x cột) như hình sau:

$$\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,480} \\ w_{1,2} & w_{2,2} & \cdots & w_{2,480} \\ \cdots & \cdots & \cdots & \cdots \\ w_{360,1} & w_{360,2} & \cdots & w_{360,480} \end{bmatrix}$$

– Tuy nhiên để biểu diễn 1 màu ta cần 3 thông số (r,g,b) nên gọi wi j = (ri j;gi j;bi j) để biểu diễn dưới dạng ma trận thì sẽ như sau:

$$\begin{bmatrix} r_{1,1} & r_{1,2} & \cdots & r_{1,480} \\ r_{1,2} & r_{2,2} & \cdots & r_{2,480} \\ \cdots & \cdots & \cdots & \cdots \\ r_{360,1} & r_{360,2} & \cdots & r_{360,480} \end{bmatrix}$$

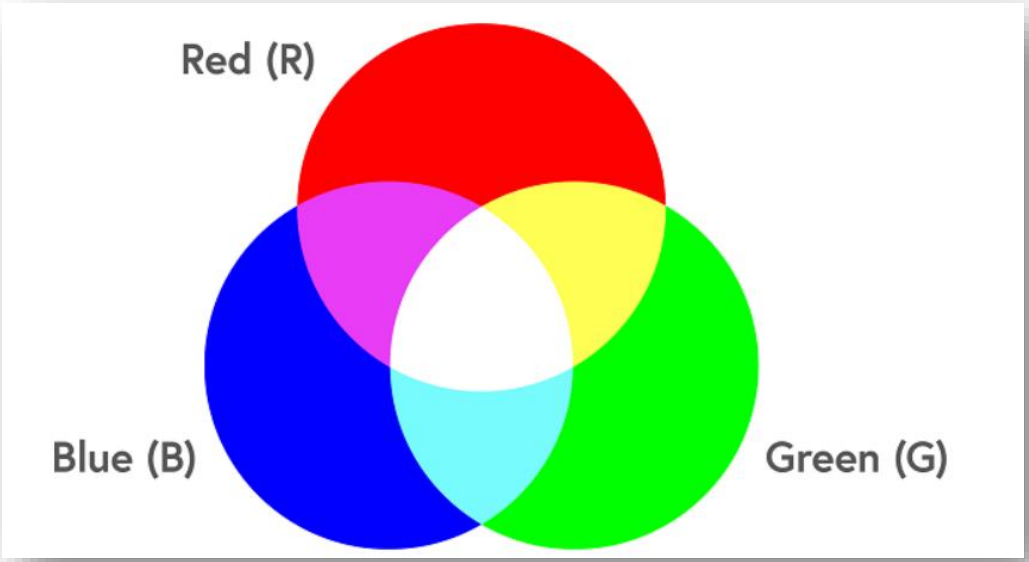
$$\begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,480} \\ b_{1,2} & b_{2,2} & \cdots & b_{2,480} \\ \cdots & \cdots & \cdots & \cdots \\ b_{360,1} & b_{360,2} & \cdots & b_{360,480} \end{bmatrix}$$

$$\begin{bmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,480} \\ g_{1,2} & g_{2,2} & \cdots & g_{2,480} \\ \cdots & \cdots & \cdots & \cdots \\ g_{360,1} & g_{360,2} & \cdots & g_{360,480} \end{bmatrix}$$

*Hình 1 Ở trên là 3 ma trận đã được tách ra và mỗi ma trận biểu thị cho một màu cụ thể (đỏ, xanh dương, xanh lá).*

### 2.3 Hệ màu RGB là gì ?

- Một pixel được tạo thành từ 3 số nguyên từ 0 đến 255 (các số nguyên thể hiện cường độ của màu đỏ, xanh lục và xanh lam).
- RBG là viết tắt của Red (đỏ), Blue (xanh dương), Green (xanh lục) đây là 3 màu chính của ánh sáng khi được tách ra từ lăng kính. Nếu muốn tạo thành một màu mới thì chỉ cần phối trộn 3 màu ở một tỉ lệ nhất định.




*Hình 2 Phối trộn màu bổ sung: thêm đỏ vào xanh lá cây tạo ra vàng; thêm vàng vào xanh lam tạo ra trắng. [1]*

- Ví dụ khi bạn chọn màu ở đây. Khi bạn chọn một màu thì sẽ ra một bộ ba số tương ứng (r,g,b)
- Với mỗi bộ 3 số r, g, b nguyên trong khoảng [0, 255] sẽ cho ra một màu khác nhau. Do có 256 cáchchọn r, 256 cách chọn màu g, 256 cách chọn b => tổng số màu có thể tạo ra bằng hệ màu RGB là: 256 * 256 * 256 = 16777216 màu
- Khi mà khoảng cách giữa các điểm ảnh càng gần hoặc mật độ điểm ảnh càng lớn (độ phân giải ảnh) thì ảnh càng nét , càng sống động. Ta ví độ phân giải ảnh 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

- Grayscale: giống như ảnh màu thì ảnh xám cũng được biểu diễn thành ma trận tuy nhiên vì là ảnh xám nên chỉ cần biểu diễn trên một ma trận là đủ.

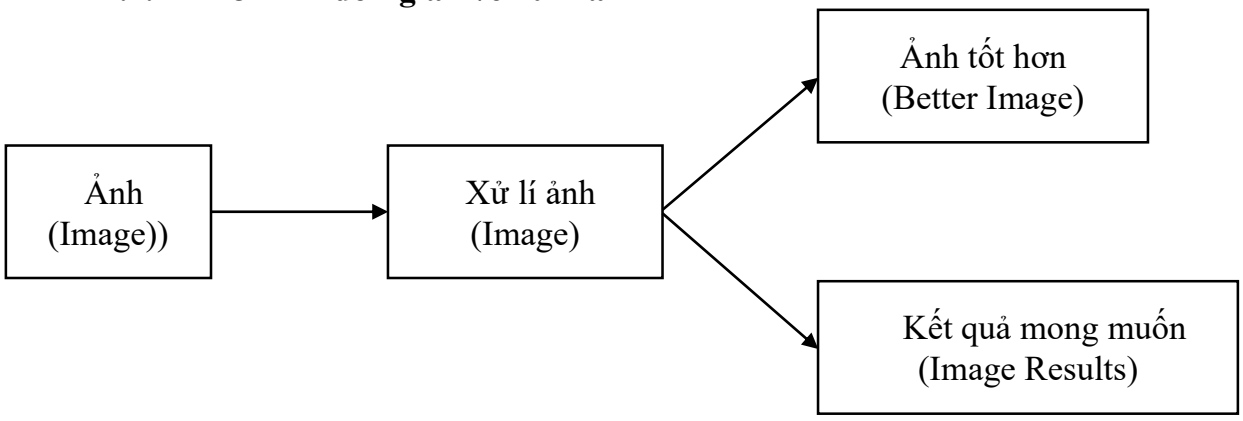
$$\begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,480} \\ w_{1,2} & w_{2,2} & \cdots & w_{2,480} \\ \cdots & \cdots & \cdots & \cdots \\ w_{360,1} & w_{360,2} & \cdots & w_{360,480} \end{bmatrix}$$

*Hình 3 Một pixel là một số nguyên có giá trị từ 0 đến 255 (giá trị 0 biểu diễn cho màu đen và giá trị 255 biểu diễn cho màu đen).*

### 2.4 Xử lí ảnh là gì ?

- Trước đây công nghệ - khoa học chưa phát triển khi ảnh thu được chỉ là tín hiệu tương tự (Analog) thì hiện nay sự phát triển đã phát triển vượt bật và bây giờ xử lý hình ảnh là quá trình chuyển đổi hình ảnh thành dạng kỹ thuật số và thực hiện một số thao tác nhất định để lấy kết quả mong muốn từ hình ảnh đó. Hệ thống xử lý hình ảnh thường xử lý tất cả các hình ảnh dưới dạng tín hiệu 2D khi áp dụng một số phương pháp xử lý tín hiệu định trước.
- 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.

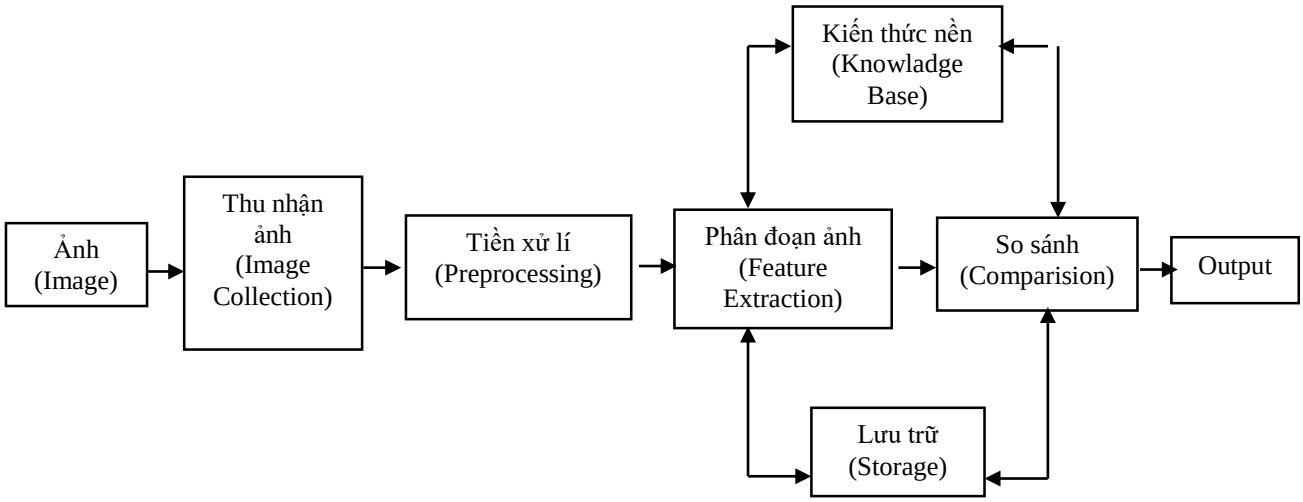
#### 2.4.1 Mô hình đơn giản về xử lí ảnh




- Dưới đây là 5 mô hình xử lí ảnh chính:

- Trực quan hóa (Visualization) - Tìm các đối tượng không nhìn thấy trong hình ảnh.
- Recognition (Recognition) - Phân biệt hoặc phát hiện các đối tượng trong hình ảnh.
- Làm sắc nét và phục hồi (Sharpening and restoration) - Tạo hình ảnh nâng cao từ hình ảnh gốc.
- Nhận dạng mẫu (Pattern recognition) - Đo các mẫu khác nhau xung quanh các đối tượng trong ảnh.
- Truy xuất (Retrieval) - Duyệt và tìm kiếm hình ảnh từ cơ sở dữ liệu lớn về hình ảnh kỹ thuật số tương tự như hình ảnh gốc.

### 2.4.2 Các bước xử lí ảnh cơ bản




*Hình 4 Các bước xử lí ảnh cơ bản*

– Mô tả chức năng các khối

- 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 này 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 tuỳ 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 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.

## 2.5 Một số ứng dụng xử lý ảnh phổ biến [2]

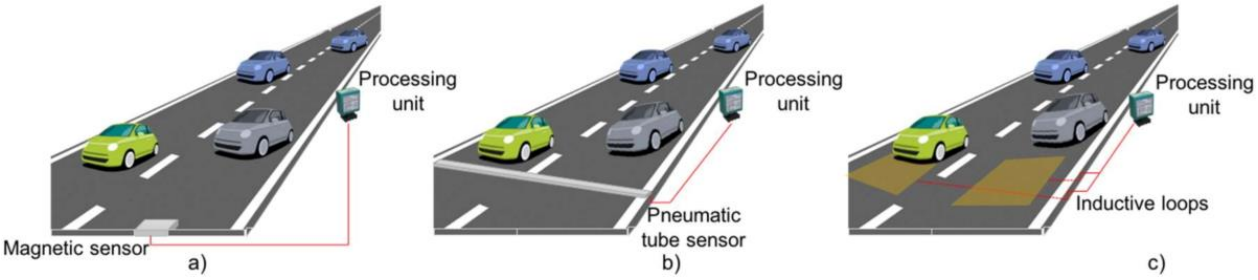
– Truy xuất hình ảnh y tế

Xử lý hình ảnh đã được sử dụng rộng rãi trong nghiên cứu y học và lập các phát đồ điều trị hiệu quả hơn. Ví dụ, được ứng dụng để phát hiện sớm ung thư vú bằng cách sử dụng thuật toán phát hiện nốt sần phức tạp trong quá trình quét vú. Do việc sử dụng trong y tế yêu cầu được đào tạo chuyên sâu nên các ứng dụng này yêu cầu triển khai và đánh giá trước khi chúng có thể được chấp nhận sử dụng.

– Công nghệ cảm biến giao thông

Trong trường hợp cảm biến giao thông, sử dụng hệ thống xử lý hình ảnh video. Khi quay video, một số vùng phát hiện sẽ phát tín hiệu “bật” bất cứ khi nào có phương tiện đi vào vùng đó và sau đó phát tín hiệu “tắt” bất cứ khi nào phương tiện ra khỏi

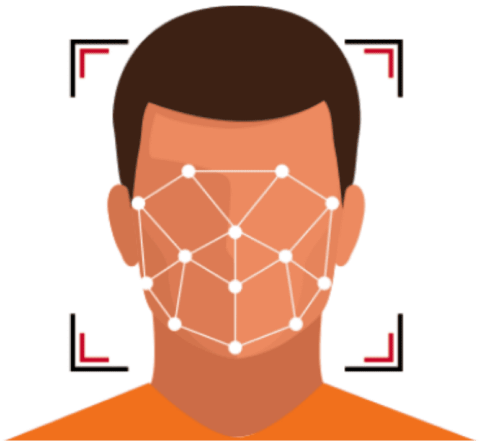
vùng phát hiện. Các vùng phát hiện này có thể được thiết lập cho nhiều làn đường và có thể được sử dụng để cảm nhận lưu lượng truy cập trong một trạm cụ thể.




*Hình 5 Mô hình phát hiện mật độ giao thông [3]*

– Phát hiện khuôn mặt

Một trong những ứng dụng phổ biến nhất của xử lý ảnh mà chúng ta sử dụng ngày nay là nhận diện khuôn mặt. Nó tuân theo các thuật toán học sâu trong đó máy được đào tạo đầu tiên với các đặc điểm cụ thể của khuôn mặt người, chẳng hạn như hình dạng khuôn mặt, khoảng cách giữa hai mắt, v.v. Sau khi dạy cho máy các đặc điểm khuôn mặt người này, nó sẽ bắt đầu chấp nhận tất cả các đối tượng trong một hình ảnh giống với khuôn mặt của con người. Nhận diện khuôn mặt là một công cụ quan trọng được sử dụng trong bảo mật, sinh trắc học và thậm chí cả các bộ lọc có sẵn trên hầu hết các ứng dụng truyền thông xã hội ngày nay.



*Hình 6 Ứng dụng phổ biến của xử lí ảnh [4]*

# CHƯƠNG 3 CƠ SỞ LÍ THUYẾT

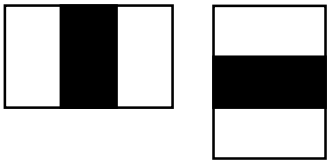
## 3.1 Face Detection

### 3.1.1 Haarcascade [5]

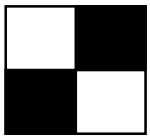
- Thuật toán để phát hiện khuôn mặt thường dùng là Haar Cascade (do ViolaJones đồng sáng tạo) được phát triển vào năm 2001 trong những bài báo của riêng họ.
- Haar được xem như những model đã được training sẵn với các đặc trưng củacác phần trên cơ thể cần phát hiện . Haar cascade được hiểu đơn giản là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

### 3.1.2 Cách hoạt độn của thuật toán

(a) Edge Features

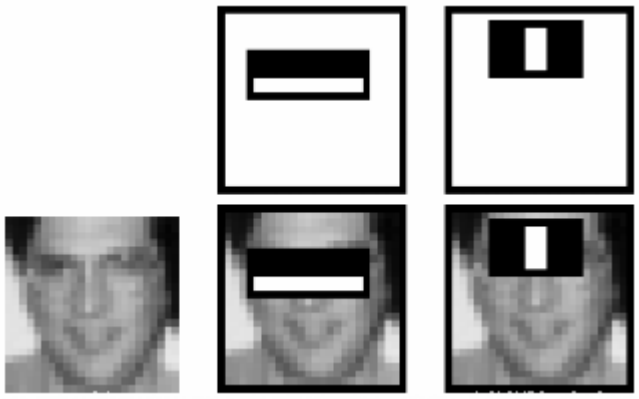


(b) Line Features



(c) Four - rectangle Features

*Hình 7 (a) Bộ lọc cạnh– (b) bộ lộc đường– (c) bộ lọc bắt các đặc trưng hình vuông*



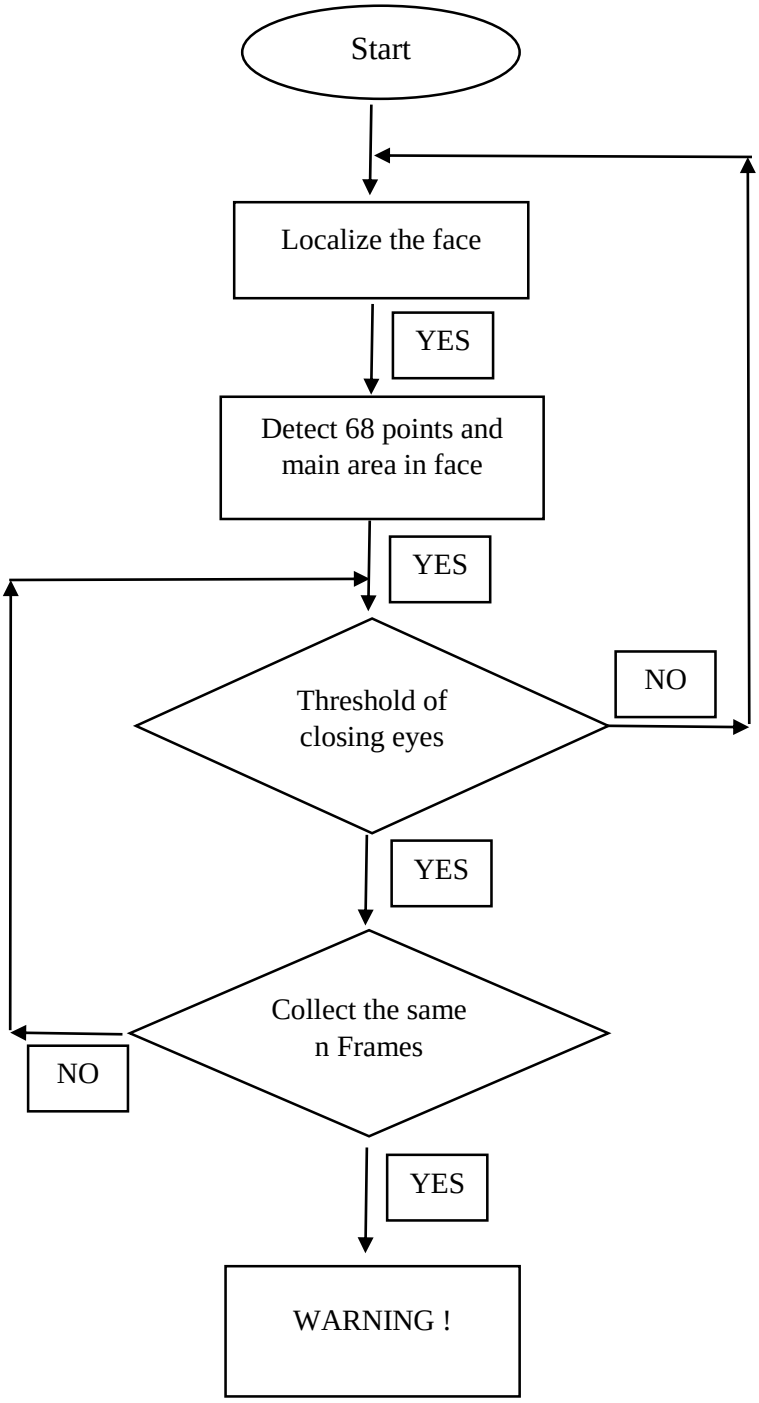
*Hình 8 Điểm bám bắt cơ bản đặc trưng của Haar Cascade*

- Một khuôn mặt có những đặt điểm sau: ví dụ vùng mắt sẽ tối hơn vùng dưới mặt và mắt sẽ tối hơn mũi. Như ảnh minh hoạ 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.

### 3.2 Thuật toán Facical Landmark

- Phát hiện mốc trên khuôn mặt là một nhiệm vụ thị giác máy tính, trong đó một mô hình cần dự đoán các điểm chính đại diện cho các vùng hoặc mốc trên khuôn mặt của con người – mắt, mũi, môi và các điểm khác. Phát hiện mốc trên khuôn mặt là một tác vụ cơ bản có thể được sử dụng để thực hiện các tác vụ thị giác máy tính khác, bao gồm ước tính tư thế đầu, xác định hướng nhìn, phát hiện cử chỉ trên khuôn mặt và hoán đổi khuôn mặt.
- Bộ dữ liệu khuôn mặt tiêu chuẩn cung cấp chú thích gồm 68 tọa độ x và y cho biết những điểm quan trọng nhất trên khuôn mặt của một người. Dlib là một thư viện nguồn mở thường được sử dụng có thể nhận dạng các mốc này trong một hình ảnh.

### 3.3 Lưu đồ thuật toán



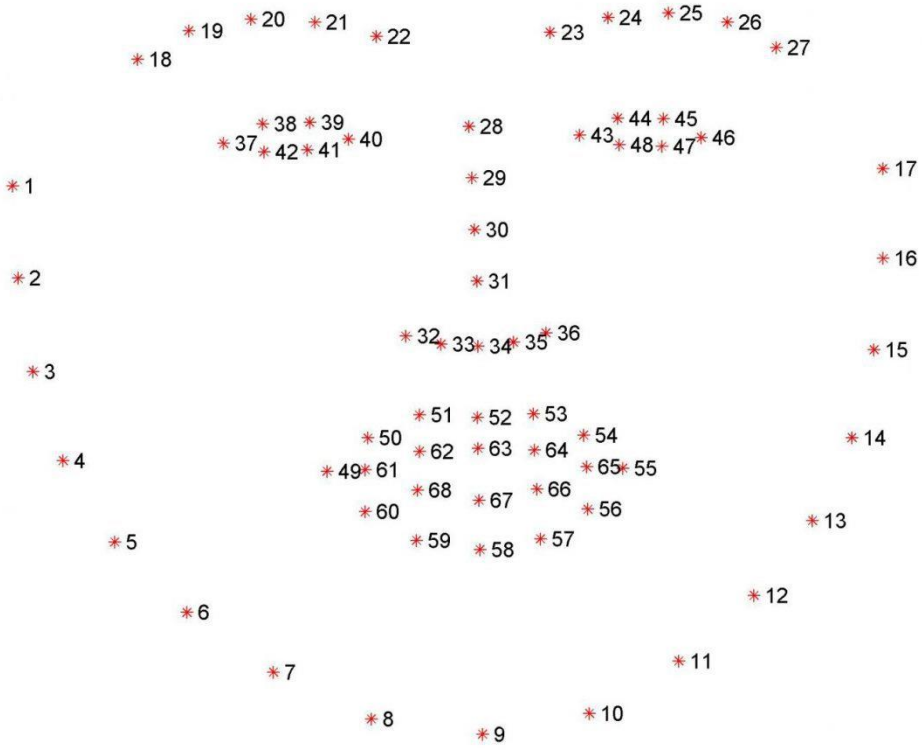

*Hình 9 Lưu đồ giải thuật phát hiện nhắm mắt - ngáp*

### 3.3.1 Các bước thực hiện thuật toán

– Việc xác định facial landmark gồm có hai bước:

- Bước 1: Xác định được vị trí khuôn mặt trong bức ảnh
- Bước 2: Xác định được các điểm tạo nên cấu trúc của khuôn mặt

– Việc xác định vị trí khuôn mặt có thể được thực hiện bằng nhiều cách từ đơn giản như thuật toán Haar cascades đến phức tạp như các thuật toán dựa trên deep-learning. Tuy nhiên dù sử dụng thuật toán nào, mục đích cuối cùng là ta sẽ thu được một vùng (thường là hình vuông) được xác định bởi tọa độ (x,y) bao quanh khuôn mặt trong bức ảnh.




*Hình 10 68 điểm landmarks từ dlib [5]*

– Sau khi xác định được khuôn mặt trong bức ảnh, chúng ta sẽ xác định cấu trúc của khuôn mặt. Có rất nhiều kiểu cấu trúc khuôn mặt khác nhau nhưng về cơ bản, chúng ta sẽ phải xác định được những phần sau:

- Miệng
- Lông mày phải
- Lông mày trái
- Mắt phải
- Mắt trái
- Mũi
- Hàm

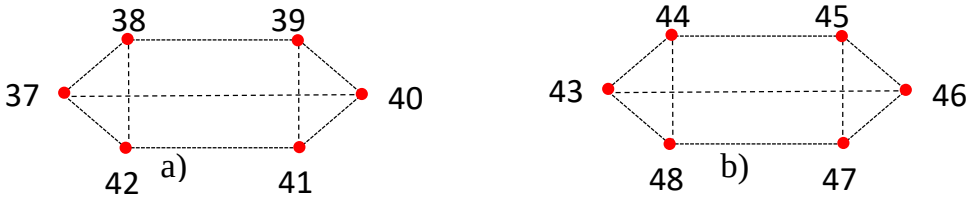
## 3.4 Tính toán tỉ lệ đóng mở của mắt – miệng [6] [7]

– Để phát hiện tài xế lái xe có trong tình trạng buồn ngủ hay ngủ gật không thì chúng ta cần phải tính toán tỉ lệ đóng mở mắt và miệng trong thời gian thực.

### 3.4.1 Tỉ lệ mắt (EAR)

– Phương pháp Facial Landmark trên chia khuôn mặt thành 68 điểm. Ứng dụng kết quả đấy, 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].




*Hình 11 Vị trí a) mắt trái và b) mắt phải trong Facical Landmark 68 điểm*

– Công thức tính tỉ lệ khung mắt:


a) b)

*Hình 12 a) Mắt mở khi tỉnh táo và b) đóng khi ngủ*

$$EAR = \frac{|p2 - p6| + |p3 - p5|}{2\ x\ |p1 - p4|}$$

– Tại các điểm từ p1 tới p6 hiển thị dạng 2D, khi mắt mở thì giá trị sẽ là một hằng số nhưng khi ngưỡng mắt tiến dần về không thì mắt sẽ đóng. Giá trị sẽ không phụ thuộc vào tư thế đầu và khoảng cách của đầu và có sự khác biệt nhỏ giữa các cá nhân khi mắt mở. Vì nháy mắt được thực hiện đồng bộ bởi cả hai mắt, EAR của cả hai mắt được tính trung bình. Do tính đơn giản của công thức tính toán, nó có hiệu suất thời gian thực cao.

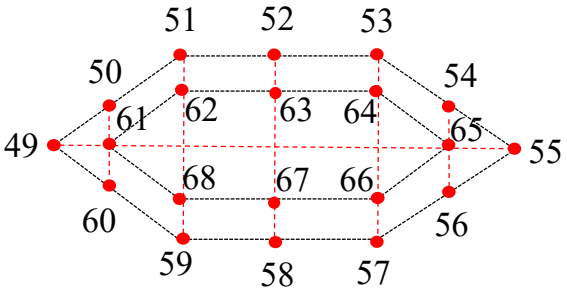
$$Eye_{Blink\_Detect} = (EAR_{Left\_Eye} + EAR_{Right\_Eye})/2$$

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

– Ngoài ra, mặc dù được xem là có độ chính xác phát hiện cao và đơn giản, nhưng nó sẽ làm giảm độ chính xác khi có chuyển động đột ngột của khuôn mặt hoặc khi khuôn mặt ở quá xa máy ảnh.

### 3.4.2 Tỉ lệ miệng

– Theo cơ chế sinh học khi con người ngáp thì miệng sẽ mở to, 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.




*Hình 13 Các điểm của miệng trên Facical Landmark*

– Công thức tính tỉ lệ khuôn miệng (MAR)

$$MAR = \frac{|50 - 60| + |51 - 59| + |52 - 58| + |53 - 57| + |54 - 56|}{2\ x\ |49 - 55|}$$

### 3.4.3 Khoảng cách Euclide [8]

– 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=(A1,A2,A3,A4,…..,Ax) tương tự điểm B =(B1,B2,B3,….,Bx) nằm chứa trong không gian Euclide thì khoảng AB có chiều dài được xác định theo công thức:

$$d\ (a,b) = \sqrt{\sum_{i=1}^{x} (A_i - B_i)^2}$$

*Hình 14 Công thức tính khoảng cách*

## 3.5 Ngôn ngữ và lí 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 câu lệnh 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.

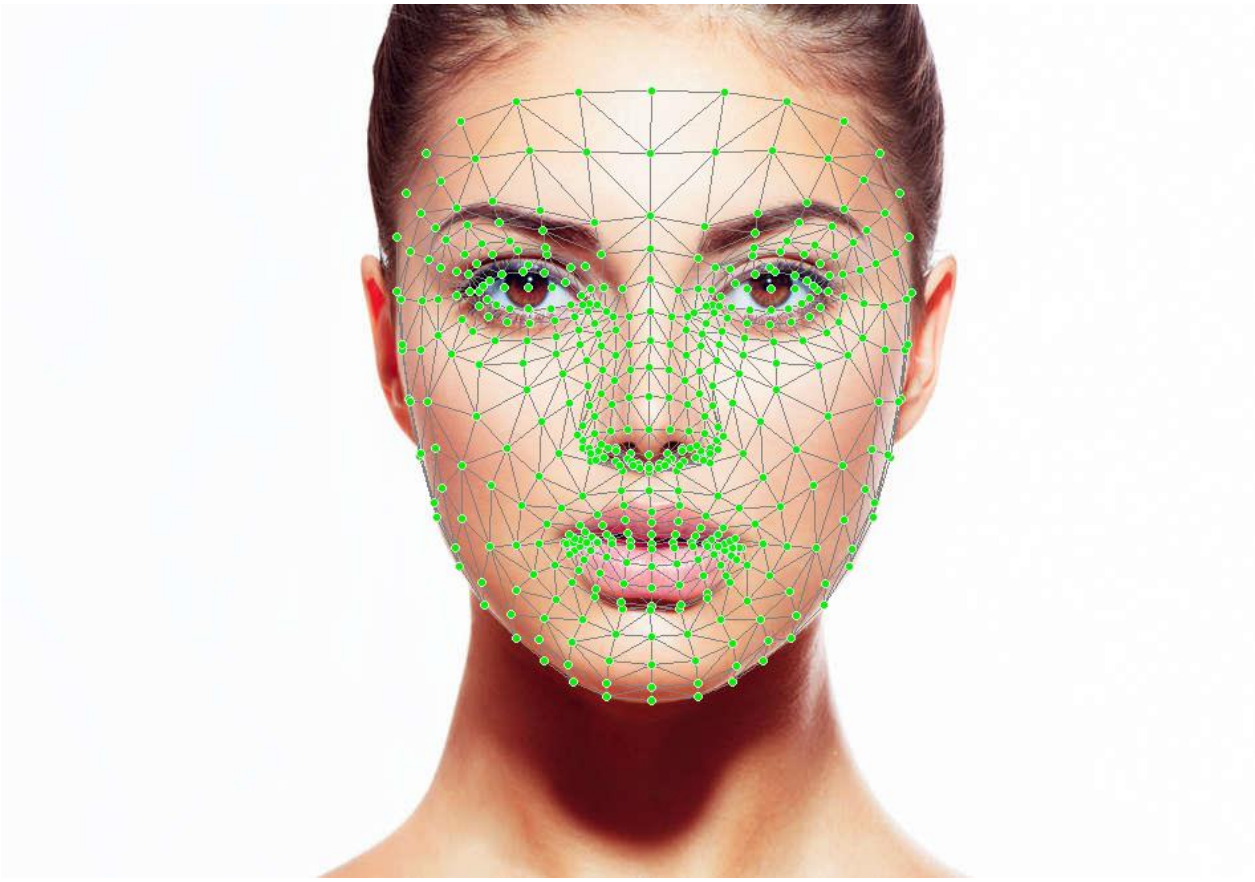
## 3.6 Các thư viện được sử dụng

### 3.6.1 Thư viện Mediapipe [9]

– Với sự phát triển nhanh chóng của khoa học – kĩ thuật thì công nghệ trí tuệ nhân tạo (AI) đang xuất hiện hầu như tất cả các lĩnh vực trong đời sống. Với tính chính xác và tính tự động hóa cao nó rất thích hợp để ứng dụng giải quyết các yêu cầu

đòi hỏi độ khó cao. Mắc dù vậy, các mô hình lại yêu cầu phần cứng khá cao. Chính vì thế, Google đã đưa ra một một bộ công cụ cung cấp cho các bài toán về trí tuệ nhân tạo (AI) hay máy học (Machine learning) đã được tối ưu để chạy trên nhiều nền tảng khác nhau, với tên gọi là MediaPipe.

– Chính vì thế thư viện Dlib bên trên giúp chúng ta giải quyết vấn đề độ đóng mở của mắt hoặc miệng qua 68 điểm để và 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.



*Hình 15 Tọa độ khuôn mặt được chia thành 468 điểm*

### 3.6.2 Thư viện OpenCV và thư viện Numpy [10]

– 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 vả xử lý ảnh"
- Để cài đặt open-cv python trên mô hình cũng như môi trường thí nghiệm, 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 kĩ 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"**

### 3.6.3 Các hàm được sử dụng [11] [12]

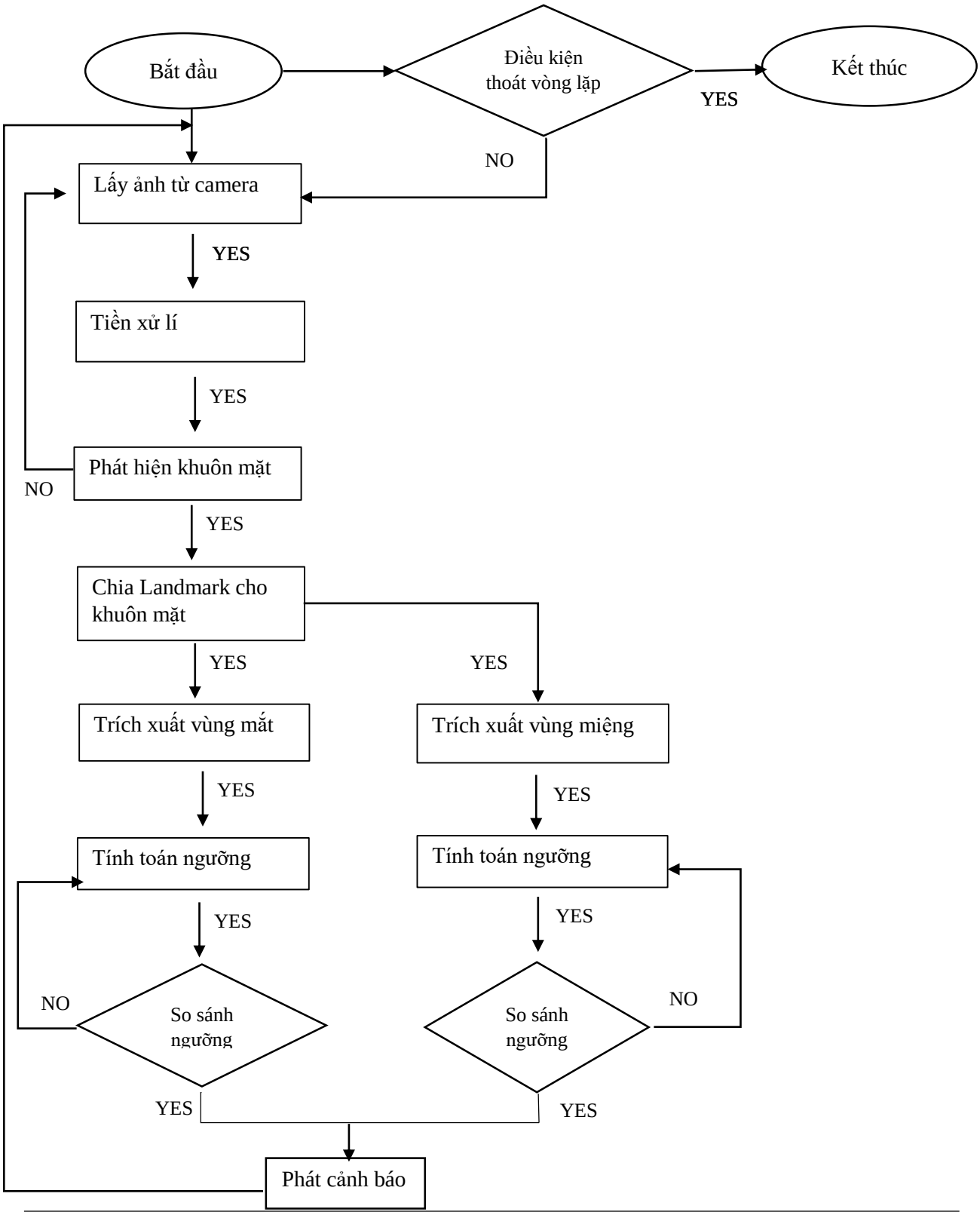
- **cv2.VideoCapture():** ta tạo 1 biến, gán tham số vào hàm, với tham số 0-n, hàm trả về ID của camera kết nối với CPU, với tham số là file ảnh hoặc video, hàm trả về file ảnh hoặc video đó. Tuỳ vào ý đồ bài toán mà ta chọn hình ảnh, video hoặc realtime camera.
- **cv2.cvtColor()**: nếu ta cần hệ màu 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()** [13]: 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 kì (1ms hoặc số ms mong muốn) và điều kiện để kết thúc và thoát vòng lặp.

- **np.concatenate()**: nối các mảng Numpy.

# CHƯƠNG 4 THỰC HIỆN CHƯƠNG TRÌNH

## 4.1 Lưu đồ giải thuật



## 4.2 Chức năng các khối

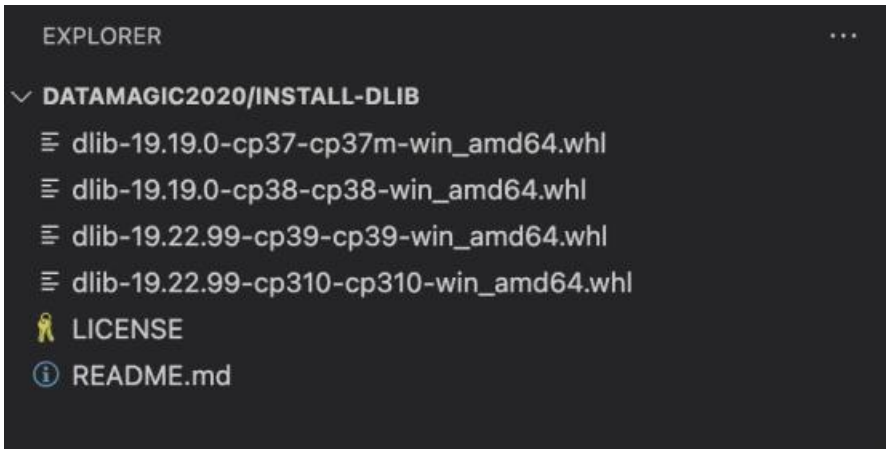
- Đầu tiên, đề truy cập camera, ta ứng dụng thư viện imutils, chương trình sẽ thiết lập kết nối camera, lấy từng frame ảnh đế tiến hành xử lý.
- Sau đến quá trình tiền xử lý, ta tiến hành liên tục (lặp vô hạn) các frame nhận đươc, sau đó tiến hành:
    - Tăng độ sáng, độ tương phản để bước nhận diện 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 rasberry 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 bài ở thư viện dlib. Ta cần phải chú ý một số điểm như sau:
    - Mắt trái [37;42]
    - Mắt phải [43;48]
    - Miệng: môi ngoài [49:60], môi trong [61:68]
- Bước tiếp theo sẽ là tìm ngưỡng mắt và miệng. 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.
- Tương tự vậy ta cũng thực hiện giống như với ngưỡng miệng.
- Ở bước so sánh ngưỡng nếu ngưỡng mắt tính được nhỏ hơn ngưỡng mắt đã đặt hay ngưỡng miệng tính được lớn hơn ngưỡng miệng đã đặt thì đưa ra cảnh báo buồn ngủ.

## 4.3 Tiến hành lập trình

### 4.3.1 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:
  "pip install cmake"
  "pip install dlib"
- Lưu ý với hệ điều hành window sau khi cài cmake, ta download gói dlib về tuỳ 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 16 Các file cài đặt dlib cho window tuỳ 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 dễ nhớ và đường dẫn tuỳ mỗi người.
- Sau đấy 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 5 KẾT QUẢ, NHẬN XÉT VÀ ĐÁNH GIÁ

## 5.1 Phát hiện độ đóng mở của mắt và miệng

### 5.1.1 Phát hiện độ đóng mở của mắt [14]

**Khai báo thư viện và tham số**

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

```
import dlib
import cv2
import imutils
import numpy as np
import time
import os
from threading import Thread
from scipy.spatial import distance as dist
from imutils import face_utils
from threading import Thread
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.

```
##### dat nguonng mat #####
EYE_THRESH = 0.25
EYE_FRAMES = 30
COUNTER = 0
```

**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.

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

```
def results_EAR(shape):

    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    leftEye =shape[lStart:lEnd]
    rightEye=shape[rStart:rEnd]

    leftEAR =calc_EAR(leftEye)
    rightEAR=calc_EAR(rightEye)

    ear= (leftEAR + rightEAR) / 2.0
    return(ear, leftEye, rightEye)
```

– 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 xử dụng thư viện xử lý ảnh open-cv để tải về model haarcascade từ đó nhận diện được khuôn mặt trong ảnh , video hoặc camera.

- Bộ dự đoán: ta xử dụng thư viện Dlib để tải về file shape_predictor từ đó chia khuôn mặt đã phát hiện được bên trên thành 68 điểm.
- 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 đấy, 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.

```
#### thuc hien vong lap pht hien mat ####
while TRUE:
    ret, frame = cap.read()
    frame=imutils.resize(frame, width=500)
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_cascade(gray)
    rects= detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
                                     minSize = (30, 30), flags = cv2. CASCADE_SCALE_IMAGE)
    ### phat hien mat ####
    for face in faces:

        x1=face.left()
        y1=face.top()
        x2=face.right()
        y2=face.bottom()
        cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

       #### phat hien mat ###

        for (x, y, w, h) in rects:
            rect=dlib.rectangle(int(x), int(y), int(x+w), int(y+h))
            shape = predictor(gray, rect)
            shape=face_utils.shape_to_np(shape)

            eye = results_EAR(shape)
            ear = eye[0]
            leftEye = eye[1]
            rightEye = eye[2]
            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull= cv2.convexHull(rightEye)
            cv2.drawContours(frame, [leftEyeHull], -1,  (0, 255, 0),1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0),1)

            if ear <= EYE_THRESH:
                COUNTER +=1
                print (COUNTER)
                print(ear)
                if COUNTER >= EYE_FRAMES:
                    if alarm_status == False:
                        alarm_status = True
                        t = Thread(target=sound_alarm, args=('wake up sir',))
                        t.daemon = True
                        t.start()

                    cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

            else:
                COUNTER = 0
                alarm_status = False
            cv2.putText(frame, "EAR: {:.2f}".format(ear), (380, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0 , 255), 2)
    cv2.imshow("Detect Driver State: EAR", frame)
    if(cv2.waitKey(1)==ord('q')):
        break

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

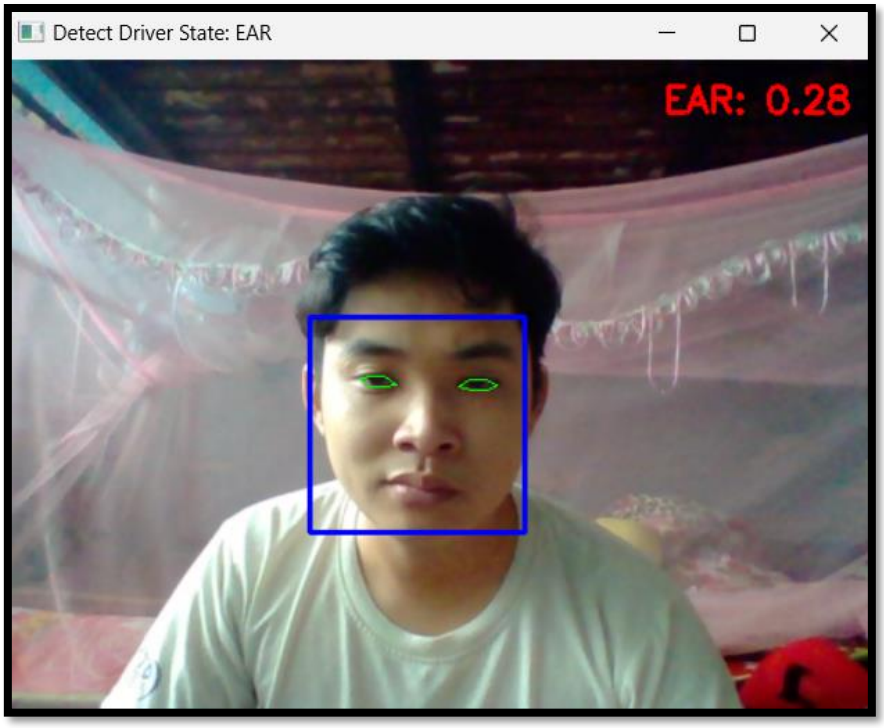
**Chạy chương trình:**

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

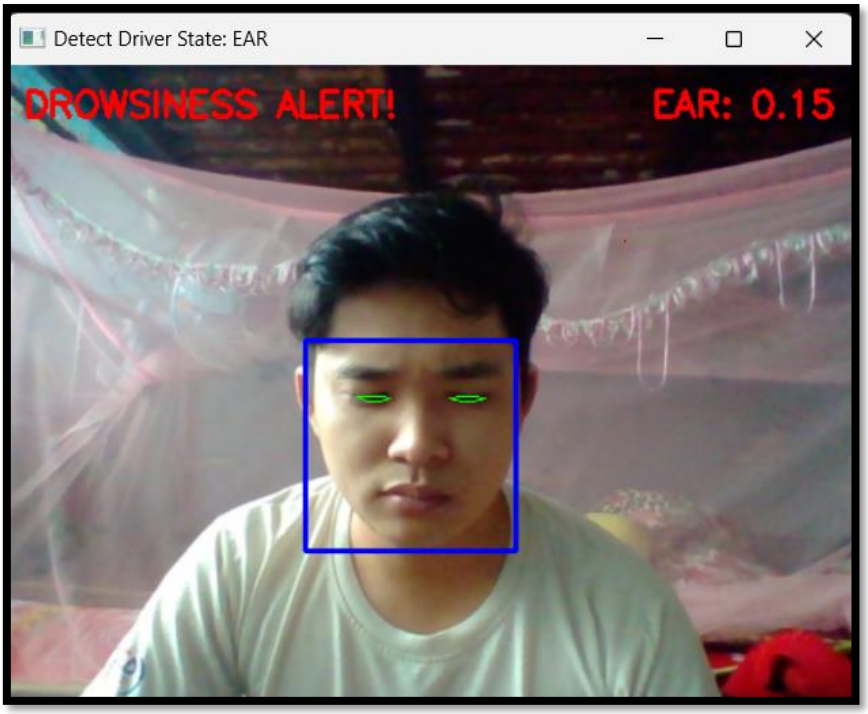
If __name__==”__main__”:

main()

**Kết quả:**




*Hình 17 Trạng thái bình thường*




*Hình 18 Trạng thái nhắm mắt*

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

**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

**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.

**Khai báo hàm:**

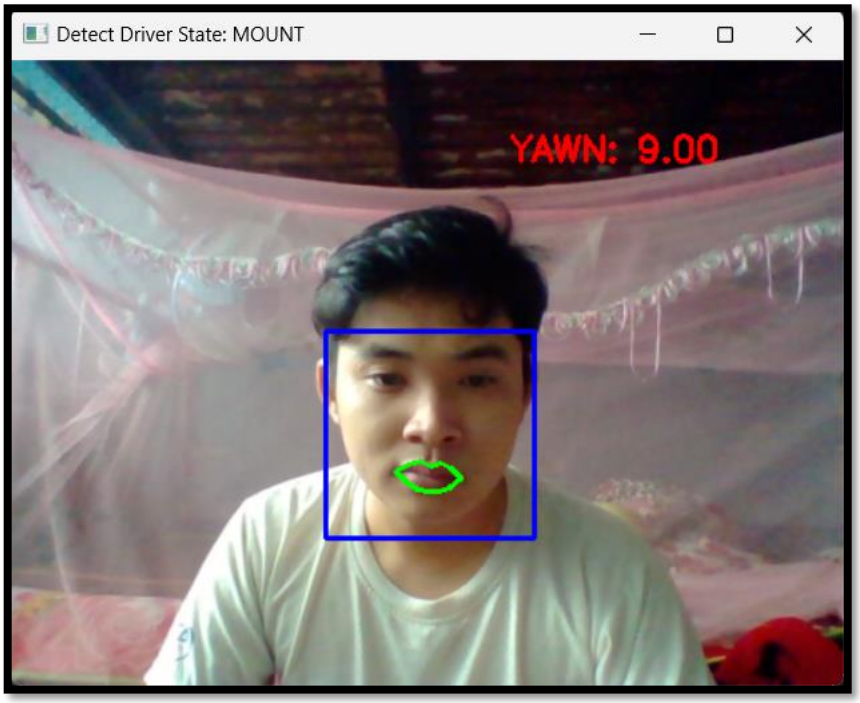
– Hàm tính độ mở của miệng:

```
23  def lip_distance(shape):
24      top_lip=shape[50:53]
25      top_lip=np.concatenate((top_lip, shape[61:64]))
26
27      low_lip=shape[56:59]
28      low_lip=np.concatenate((low_lip, shape[65:68]))
29
30      top=np.mean(top_lip, axis=0)
31      low=np.mean(low_lip, axis=0)
32
33      distance=abs(top[1] - low[1])
34      return distance
```

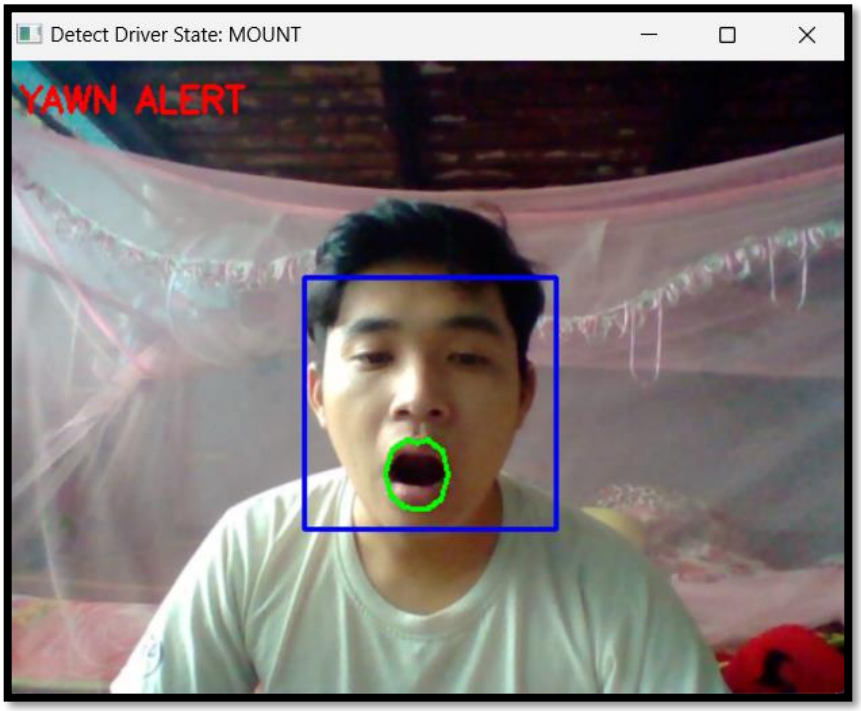
– Hàm chính:

```python
def main():
    cap = cv2.VideoCapture(0)
    detector = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
    face_cascade = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

    MOUNTH_THRESH = 25
    MOUNTH_FRAMES = 20
    COUNTER = 0

    #### thuc hien vong lap pht hien mat  va mieng ####
    while TRUE:
        ret, frame = cap.read()
        frame=imutils.resize(frame, width=500)
        gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        faces = face_cascade(gray)
        rects= detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
                                         minSize = (30, 30), flags = cv2. CASCADE_SCALE_IMAGE)

    ### phat hien khuon mat ####
        for face in faces:

            x1=face.left()
            y1=face.top()
            x2=face.right()
            y2=face.bottom()
            cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

     #### phat hien mieng ###

            for (x, y, w, h) in rects:
                rect=dlib.rectangle(int(x), int(y), int(x+w), int(y+h))
                shape = predictor(gray, rect)
                shape=face_utils.shape_to_np(shape)
                distance= lip_distance(shape)
                print(distance)
                lip=shape[48:60]
                cv2.drawContours(frame,[lip], -1, (0,255,0), 2)

                if (distance > MOUNTH_THRESH):
                    cv2.putText(frame,"YAWN ALERT",(10, 30),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                    if not alarm_status2 :
                        alarm_status2 =True
                        print('warring warring')
                        t2=threading.Thread(target=sound_alarm, args=('wake up',))
                        t2.daemon=True
                        t2.start()
                else:
                    alarm_status2=False

                    cv2.putText(frame,"YAWN: {:.2f}".format(distance),(300, 60),
                                cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)


        cv2.imshow("Detect Driver State: MOUNT", frame)
        if(cv2.waitKey(1)==ord('q')):
            break

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

**Kết quả:**




*Hình 19 Trạng thái bình thường*




*Hình 20 Tráng thái buồn ngủ*

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

**Nhận xét**

– Ưu điểm:

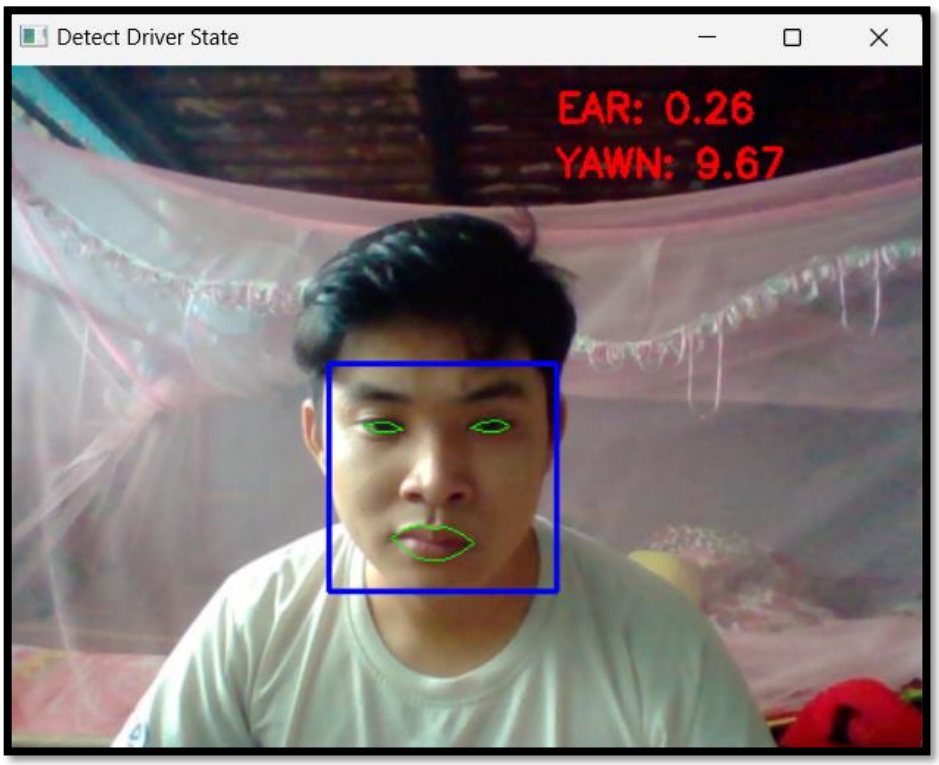
• Phát hiện được hành vi ngáp của tài xế.

• Tăng độ chính xác cho mô hình.

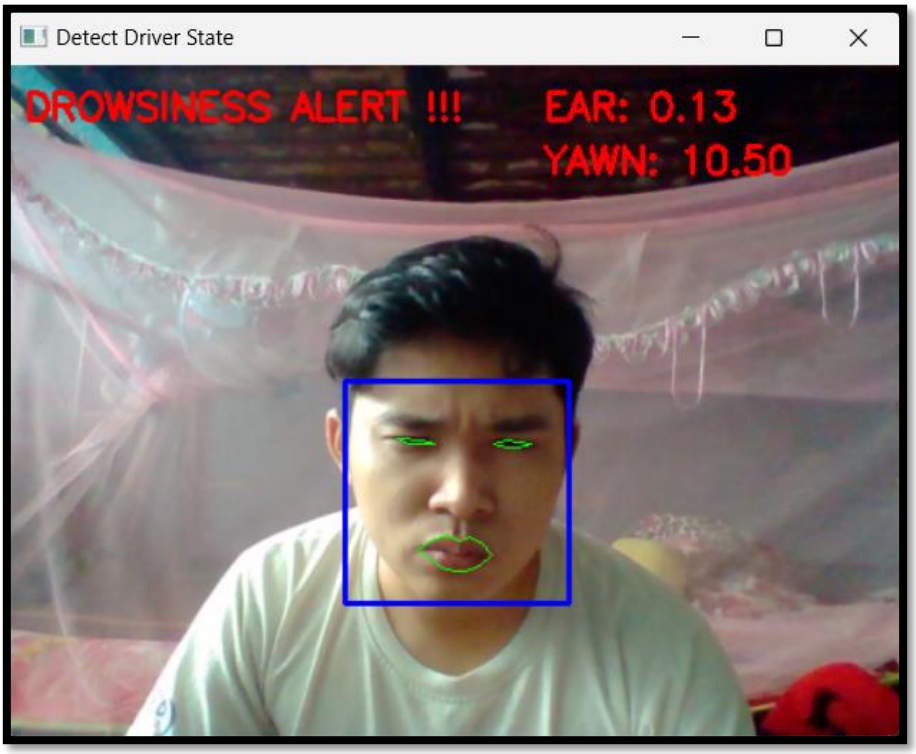
– Nhược điểm:

• Nếu miệng bị che (đeo khẩu trang , …) thì sẽ không phát hiện được.

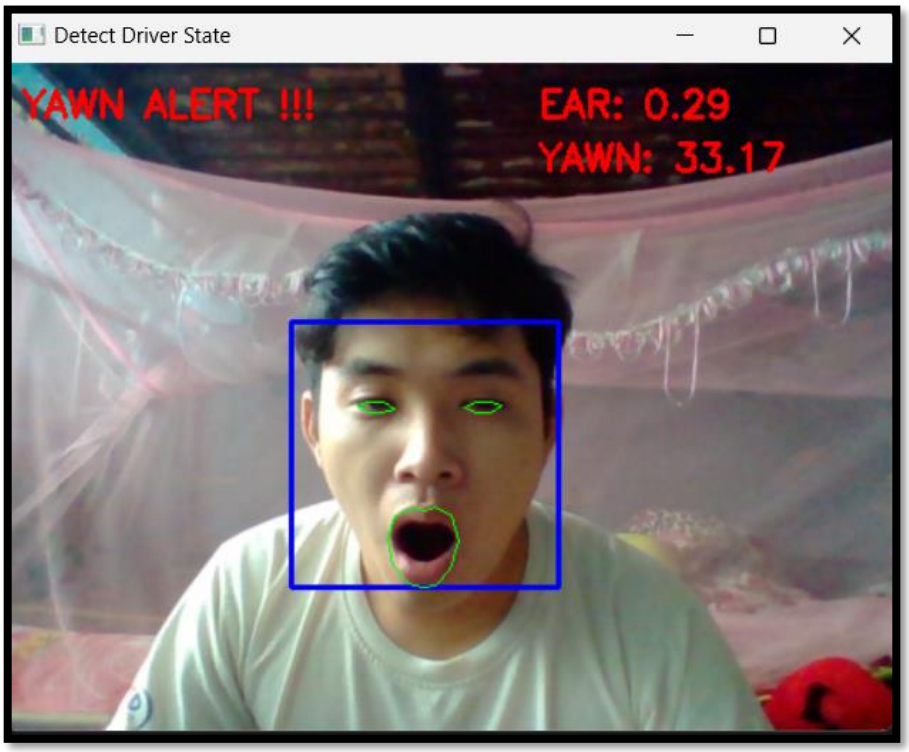
### 5.1.3 Kết hợp 2 mô hình trên




*Hình 21 Trạng thái bình thường*




*Hình 22 Trạng thái ngủ gật*




*Hình 23 Trạng thái ngáp ngủ*

# 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.
- Nếu đước xử lí qua 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.
- 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.

## 6.2 Ưu điểm

- Sử dụng tiết kiệm được chi phí hết mức có thể nhưng vẫn đảm bảo có một CPU mạnh mẽ đê 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.

## 6.3 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.

## 6.4 Hướng phát triển tương lai

- Nâng cấp mô hình xử lí với Facical Landmark 468 điểm để phát hiện hướng nhìn của tài xế.

- Áp dụng các ứng dụng xử lí ảnh mới hơn như huấn luyện cho mô hình bằng một tập mẫu ảnh lớn để tăng độ chính xác.
- Cải thiện độ chính xác và tốc độ xử lí cho mô hình bằng những phần cứng mạnh mẽ hơn cộng với 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 LIỆ THAM KHẢO

[1] H. m. RBG. https://printgo.vn/he-mau-rgb-la-gi-nhung-dieu-can-biet-ve-he-mau-rgb-v988.

[2] "Simplilearn," . https://www.simplilearn.com/image-processing-article?tag=what%20is%20image%20proccessing.

[3] "Semantic Scholar,".https://www.semanticscholar.org/paper/Sensor-Technologies-for-Intelligent-Transportation-Ib%C3%A1%C3%B1ez-Zeadally/64ddaae6b81e40b1ef2f842952a4a548798e528f.

[4] "Rank One Computing," . https://roc.ai/2018/11/02/how-automated-face-recognition-fr-works/.

[5] N. Đ. Q. Trương Quốc Định, "Hệ thống phát hiện tình trạng ngủ," *Tạp chí Khoa học Trường Đại Học Cần Thơ,* 2015.

[6] Y. B. Burcu Kır Savaş, "Real Time Driver Fatigue Detection Based on SVM".

[7] R. W. WANGHUA DENG, "Real-Time Driver-Drowsiness Detection," 2017.

[8] "Wilipedia," . https://en.wikipedia.org/wiki/Euclidean_distance.

[9] D. o. Mediapipe. . https://google.github.io/mediapipe/.

[10] N. T. Tuấn, "Deep Learning Cơ Bản v2," 2020.

[11] "Tutorial of OpenCV," . https://docs.opencv.org/4.x/d9/df8/tutorial_root.html.

[12] D. o. Numpy. . https://numpy.org/doc/stable/.

[13] T. o. O. o. GeeksForGeeks. https://www.geeksforgeeks.org/opencvpython-tutorial/?ref=lbp.

[14] "Pyimagesearch" . https://pyimagesearch.com/2017/04/10/detect-eyes-nose-lips-jaw-dlib-opencv-python/.

# PHỤ LỤC

## Chương trình phát hiện nhắm mắt

```
from pickle import TRUE
from sys import flags
import cv2
import numpy as np
import dlib
import imutils
import time
import os
import signal
from threading import Thread
from pyparsing import lineStart
from scipy.spatial import distance as dist
from imutils import face_utils
import threading
import playsound
from playsound import playsound

##### ham am thanh ####
def sound_alarm(path):
    # play an alarm sound
    playsound(r'C:\Users\chith\OneDrive\Desktop\2022_Python\alert.mp3')


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

def results_EAR(shape):

    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    leftEye =shape[lStart:lEnd]
    rightEye=shape[rStart:rEnd]

    leftEAR =calc_EAR(leftEye)
    rightEAR=calc_EAR(rightEye)
```

```
    ear= (leftEAR + rightEAR) / 2.0
    return(ear, leftEye, rightEye)

def main():
    cap = cv2.VideoCapture(0)
    detector = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
    face_cascade = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

    ##### dat nguonng mat #####
    EYE_THRESH = 0.25
    EYE_FRAMES = 30
    COUNTER = 0

    #### thuc hien vong lap pht hien mat ####
    while TRUE:
        ret, frame = cap.read()
        frame=imutils.resize(frame, width=500)
        gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
        faces = face_cascade(gray)
        rects= detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
                                minSize = (30, 30), flags = cv2. CASCADE_SCALE_IMAGE)
        ### phat hien mat ####
        for face in faces:

            x1=face.left()
            y1=face.top()
            x2=face.right()
            y2=face.bottom()
            cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

           #### phat hien mat ###

            for (x, y, w, h) in rects:
                rect=dlib.rectangle(int(x), int(y), int(x+w), int(y+h))
                shape = predictor(gray, rect)
                shape=face_utils.shape_to_np(shape)

                eye = results_EAR(shape)
                ear = eye[0]
                leftEye = eye[1]
                rightEye = eye[2]
                leftEyeHull = cv2.convexHull(leftEye)
                rightEyeHull= cv2.convexHull(rightEye)
                cv2.drawContours(frame, [leftEyeHull], -1,  (0, 255, 0),1)
```

```
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0),1)

            if ear <= EYE_THRESH:
               COUNTER +=1
               print (COUNTER)
               print(ear)
               if COUNTER >= EYE_FRAMES:
                  if alarm_status == False:
                     alarm_status = True
                     t = Thread(target=sound_alarm, args=('wake up sir',))
                     t.daemon = True
                     t.start()

                  cv2.putText(frame, "DROWSINESS ALERT!", (10, 30),
                           cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)

            else:
               COUNTER = 0
               alarm_status = False
            cv2.putText(frame, "EAR: {:.2f}".format(ear), (380, 30),
                     cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0 , 255), 2)
      cv2.imshow("Detect Driver State: EAR", frame)
      if(cv2.waitKey(1)==ord('q')):
         break

   cap.release()
   cv2.destroyAllWindows()

###### ket thuc ham #####
if __name__ == "__main__":
   main()
```

## Chương trình phát hiện đóng mở miệng

```
from pickle import TRUE
from sys import flags
import cv2
import numpy as np
import dlib
import imutils
import time
import os
import signal
from threading import Thread
from pyparsing import lineStart
from scipy.spatial import distance as dist
from imutils import face_utils
import threading
import playsound
from playsound import playsound

##### ham am thanh ####
def sound_alarm(path):
    # play an alarm sound
    playsound(r'C:\Users\chith\OneDrive\Desktop\2022_Python\alert.mp3')

def lip_distance(shape):
    top_lip=shape[50:53]
    top_lip=np.concatenate((top_lip, shape[61:64]))

    low_lip=shape[56:59]
    low_lip=np.concatenate((low_lip, shape[65:68]))

    top=np.mean(top_lip, axis=0)
    low=np.mean(low_lip, axis=0)

    distance=abs(top[1] - low[1])
    return distance

def main():
    cap = cv2.VideoCapture(0)
    detector = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
    face_cascade = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

    MOUNTH_THRESH = 25
    MOUNTH_FRAMES = 20
    COUNTER = 0
```

```
#### thuc hien vong lap pht hien mat  va mieng ####
while TRUE:
    ret, frame = cap.read()
    frame=imutils.resize(frame, width=500)
    gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_cascade(gray)
    rects= detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5,
                                minSize = (30, 30), flags = cv2. CASCADE_SCALE_IMAGE)

### phat hien khuon mat ####
    for face in faces:

        x1=face.left()
        y1=face.top()
        x2=face.right()
        y2=face.bottom()
        cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

 #### phat hien mieng ###

        for (x, y, w, h) in rects:
            rect=dlib.rectangle(int(x), int(y), int(x+w), int(y+h))
            shape = predictor(gray, rect)
            shape=face_utils.shape_to_np(shape)
            distance= lip_distance(shape)
            print(distance)
            lip=shape[48:60]
            cv2.drawContours(frame,[lip], -1, (0,255,0), 2)

            if (distance > MOUNTH_THRESH):
                cv2.putText(frame,"YAWN ALERT",(10, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
                if not alarm_status2 :
                        alarm_status2 =True
                        print('warring warring')
                        t2=threading.Thread(target=sound_alarm, args=('wake up',))
                        t2.daemon=True
                        t2.start()
            else:
                alarm_status2=False

                cv2.putText(frame,"YAWN: {:.2f}".format(distance),(300, 60),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
```

```
        cv2.imshow("Detect Driver State: MOUNT", frame)
        if(cv2.waitKey(1)==ord('q')):
            break

    cap.release()
    cv2.destroyAllWindows()

###### ket thuc ham #####
if __name__ == "__main__":
    main()
```

## Chương trình phát hiện buồn ngủ

```
from pickle import TRUE
from sys import flags
import cv2
import numpy as np
import dlib
import imutils
import time
import os
import signal
from threading import Thread
from pyparsing import lineStart
from scipy.spatial import distance as dist
from imutils import face_utils
import threading
import playsound
from playsound import playsound
from gtts import gTTS

    ###ham am thanh ###
def saying():
    tts = gTTS(text="wake up ! wake up !", lang='en')
    tts.save("pcvoice.mp3")
    # to start the file from python
    os.system("start pcvoice.mp3")

def sound_alarm(path):
    # play an alarm sound
    playsound(r'C:\Users\chith\OneDrive\Desktop\2022_Python\alert.mp3')
```

```
            #### KHAI BAO HAM ####
    ## tinh nguong mat ###
def calc_EAR(eye):
    A = dist.euclidean(eye[1], eye[5])
    B = dist.euclidean(eye[2], eye[4])
    C = dist.euclidean(eye[0], eye[3])
    ear= (A + B) / (2.0 * C)
    return ear

    ### tinh nguong mieng ###
def lip_distance(shape):
    top_lip=shape[50:53]
    top_lip=np.concatenate((top_lip, shape[61:64]))

    low_lip=shape[56:59]
    low_lip=np.concatenate((low_lip, shape[65:68]))

    top=np.mean(top_lip, axis=0)
    low=np.mean(low_lip, axis=0)

    distance=abs(top[1] - low[1])
    return distance

    ### tra ve ket qua tinh toan nguong mat ###
def results_EAR(shape):

    (lStart, lEnd) = face_utils.FACIAL_LANDMARKS_IDXS["left_eye"]
    (rStart, rEnd) = face_utils.FACIAL_LANDMARKS_IDXS["right_eye"]
    leftEye =shape[lStart:lEnd]
    rightEye=shape[rStart:rEnd]

    leftEAR =calc_EAR(leftEye)
    rightEAR=calc_EAR(rightEye)

    ear= (leftEAR + rightEAR) / 2.0
    return(ear, leftEye, rightEye)

    ### khoi ham chinh ###
def main():
    cap = cv2.VideoCapture(0)
    detector = cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface_default.xml')
    face_cascade = dlib.get_frontal_face_detector()
    predictor = dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')

    ### dat nguonng mat va mieng ###
```

```
   EYE_THRESH = 0.25
   EYE_FRAMES = 20
   COUNTER = 0
   MOUNTH_THRESH = 25

   while TRUE:
      ret, frame = cap.read()
      frame=imutils.resize(frame, width=500)
      gray=cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
      faces = face_cascade(gray)
      rects= detector.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize = (30, 30), flags =
cv2. CASCADE_SCALE_IMAGE)

      ### phat hien khuon mat ###
      for face in faces:
         x1=face.left()
         y1=face.top()
         x2=face.right()
         y2=face.bottom()
         cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 0), 2)

         ### phat hien mieng va mui ###
         for (x, y, w, h) in rects:
            rect = dlib.rectangle(int(x), int(y), int(x + w),int(y + h))
            shape = predictor(gray, rect)
            shape = face_utils.shape_to_np(shape)

            eye = results_EAR(shape)
            ear = eye[0]
            print("Ear = ",round(ear, 3))
            leftEye = eye [1]
            rightEye = eye[2]

            distance = lip_distance(shape)
            print("Distance = ",round(distance, 3))

            leftEyeHull = cv2.convexHull(leftEye)
            rightEyeHull = cv2.convexHull(rightEye)
            cv2.drawContours(frame, [leftEyeHull], -1, (0, 255, 0), 1)
            cv2.drawContours(frame, [rightEyeHull], -1, (0, 255, 0), 1)

            lip = shape[48:60]
            cv2.drawContours(frame, [lip], -1, (0, 255, 0), 1)

            ###phat hien nham mat ###
```

```
            if ear <= EYE_THRESH:
                COUNTER += 1
                print("Counter = ",round(COUNTER, 3))
                if COUNTER >= EYE_FRAMES:
                    if alarm_status == False:
                        alarm_status = True
                        t = Thread(target=sound_alarm,
args=('wake up wake  up',))
                        t.daemon = True
                        t.start()
                    cv2.putText(frame, "DROWSINESS ALERT !!!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),2)
            else:
                COUNTER = 0
                alarm_status = False

            ### phat hien ngap ###
            if (distance > MOUNTH_THRESH):
                cv2.putText(frame, "YAWN ALERT !!!", (10, 30),
                            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255),2)
                if alarm_status2 == False :
                    alarm_status2 = True
                    t = Thread(target=sound_alarm,
args=('wake up wake up',))
                    t.daemon = True
                    t.start()
            else:
                alarm_status2 = False

            ### hien thi thong so ra man hinh ###
            cv2.putText(frame, "EAR: {:.2f}".format(ear), (300, 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
            cv2.putText(frame, "YAWN: {:.2f}".format(distance), (300, 60),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 255), 2)
        cv2.imshow("Detect Driver State", frame)

        ### phim thoat khoi man hinh ###
        if(cv2.waitKey(1)==ord('q')):
            break
    cap.release()
    cv2.destroyAllWindows()

### ket thuc ham ###
if __name__ == "__main__":
    main()
```