**TRƯỜNG ĐẠI HỌC BÁCH KHOA TP. HỒ CHÍ MINH**

**KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH**

**---------------**

****

**BÁO CÁO THỰC TẬP TỐT NGHIỆP**

**Xây dựng ứng dụng**

**cảnh báo tài xế buồn ngủ**

**GVHD : Nguyễn Quang Hùng**

**Sinh viên : Nguyễn Văn Mạnh 51302307**

**Lê Đình Sơn 51303406**

Thành phố Hồ Chí Minh, 06/2017

Mục lục

[Mục lục 1](#_Toc483824578)

[Lời cảm ơn 2](#_Toc483824579)

[I. Giới thiệu: 3](#_Toc483824580)

[1. Vấn đề hiện tại. 3](#_Toc483824581)

[2. Mục tiêu của giai đoạn thực tập. 3](#_Toc483824582)

[II. Thư viện OpenCV, OpenMP: 5](#_Toc483824583)

[1. OpenCV: 5](#_Toc483824584)

[2. OpenMP: 9](#_Toc483824585)

[III. Cách thức hoạt động: 11](#_Toc483824586)

[1. Quá trình hoạt động : 11](#_Toc483824587)

[2. Các bước thực hiện: 14](#_Toc483824588)

[IV. Kết quả: 18](#_Toc483824589)

[V. Kết luận: 20](#_Toc483824590)

[1. Các công việc đã thực hiện: 20](#_Toc483824591)

[2. Kinh nghiệm thu được 20](#_Toc483824592)

[3. Hạn chế và khó khăn 20](#_Toc483824593)

[4. Hướng phát triển trong tương lai 21](#_Toc483824594)

[VI. Tham khảo: 22](#_Toc483824595)

# Lời cảm ơn

Chúng tôi xin gửi lời cảm ơn đến thầy Nguyễn Quang Hùng – giảng viên hướng dẫn thực hiện đề tài tại trường đại học Bách Khoa thành phố Hồ Chí Minh đã giúp đỡ chúng tôi trong suốt thời gian thực hiện đề tài này.

Cùng với đó, chúng tôi cũng chân thành gửi lời cảm ơn đến [Intel Corporation](https://en.wikipedia.org/wiki/Intel_Corporation), [Willow Garage](https://en.wikipedia.org/wiki/Willow_Garage), Itseez, NVIDIA những tổ chức đã phát triển Library OpenCV, Cuda. Chúng tôi cũng cảm ơn các tác giả của các tài liệu đã được trích dẫn.

Mặc dù đã có cố gắng trong phạm vi và khả năng cho phép, nhưng không thể tránh khỏi những thiếu sót, rất mong được sự góp ý, chỉ bảo thêm của quý thầy cô và các bạn.

Cuối cùng xin chân thành cảm ơn thầy cô và các bạn đã giành thời gian đọc tài liệu này.

# Giới thiệu:

## Vấn đề hiện tại.

Hiện nay vấn đề an toàn giao thông ở Việt Nam là một trong những vấn đề cần quan tâm, lưu ý hàng đầu. Theo các báo cáo gần đây, Việt Nam là một trong những nước có số vụ tai nạn đứng đầu thế giới. Một trong những nguyên nhân gây ra vấn đề đó chính là từ tài xế lái xe điều khiển các loại phương tiện giao thông, việc phải lái xe thường xuyên gây mất ngủ, mệt mỏi của tài xế là một trong những hiện trạng phổ biến ở Việt Nam. Điều đó gây tác động, gây ra nhiều vụ tai nạn nghiệm trọng mỗi năm, đặc biệt trên các loại phương tiện: xe tải, xe khách,.. chủ yếu hoạt động vào buổi tối, đêm hoặc điều khiển phương tiện liên tiếp nhiều ngày không có thời gian nghỉ ngơi đầy đủ, gây ra thiệt hại lớn về người và của. Vì vậy việc đối phó với sự mệt mỏi của tài xế là rất quan trọng trong việc cải thiện an toàn đường bộ. Do đó bài toán được đặt ra ở đây là thiết kế một hệ thống để xác định sự mệt mỏi, buồn ngủ của tài xế một cách gần như chính xác để đưa ra cảnh báo phù hợp để tạo ra sự an toàn của mọi người tham gia giao thông.

Hệ thống được tổng hợp dữ liệu từ camera được gắn trực tiếp trên phương tiện lái xe để lấy được hình ảnh trực tiếp khuôn mặt của tài xế. Chúng tôi đề xuất một phương pháp để phân tích dữ liệu và xác định ngưỡng của sự mệt mỏi, buồn ngủ dựa trên nhiều yếu tố. Hệ thống camera sẽ gửi dữ liệu thông tin và hoạt động về một máy chủ, máy chủ sẽ thực hiện phân tích và phản hồi về điện thoại cho người lái xe về nguy cơ có thể ngủ thiếp đi.

## Mục tiêu của giai đoạn thực tập.

* Giai đoạn 1: chúng tôi tìm hiểu rõ về thư viện OpenCV, xây dựng các tập training để phục vụ cho việc xác định các đối tượng: xây dựng các tập training từ: hình ảnh khuôn mặt chính diện, miệng và mắt được chạy trên máy tính laptop.
* Giai đoạn 2: ứng dụng thư viện và các tập đã tranining được vào việc thực hiện xác định khuôn mặt, mũi, mắt trên ảnh và camera. Ứng dụng GPU vào quá trình traning các tập dữ liệu để tăng tốc khả năng tính toán.
* Giai đoạn 3: tiếp tục xác định các đối tượng và tính toán thời gian chớp mắt, ngáp miệng dựa trên hình ảnh trực tiếp từ camera để đưa ra cảnh báo chính xác.

# Thư viện OpenCV, OpenMP:

## OpenCV:

1. Khái niệm

* OpenCV là một mã nguồn mở được phát hành dưới giấy phép của thư viện BSD bao gồm hàng trăm thuật toán và do đó miễn phí cho việc học tập lẫn thương mại. Thư viện được hỗ trợ trên C++, C, Python, Java và hỗ trợ trên Windows, Linux, MacOS, iOS và Android. OpenCV được thiết kế cho hiệu quả tính toán và tập trung mạnh mẽ vào các ứng dụng thời gian thực. Được viết tối ưu hóa C/C++, thư viện có thể tận dụng lợi thế trong quá trình đa lõi. Thêm với OpenCL, nó có thể tận dụng sự tăng tốc của phần cứng của nền tảng tính toán không đồng nhất bên dưới.
* Được phổ biến trên toàn thế giới, OpenCV có hơn 47.000 người dùng và số lượt tải ước tính vượt quá 14.000.000. Ứng dụng trên nhiều mục đích khác nhau, ví dụ như nghệ thuật tương tác, kiểm tra mìn, bản đồ khâu trên web, hoặc thong qua các robot tiên tiến.
* OpenCV có mô hình cấu trúc, bao gồm một số thư viện được chia sẻ hoặc tĩnh. Bao gồm các mô hình sau:

+ Chức năng cốt lõi (Core functionality): mô đun nhỏ gọn xác định cấu trúc dữ liệu cơ bản, bao gồm mảng đa chiều và chức năng cơ bản được sử dụng bởi tất cả các mô hình khác.

+ Xử lý hình ảnh (Image processing ) : mô đun xử lý ảnh bao gồm lọc tuyến tính và phi tuyến tính, chuyển đổi hình ảnh hình học ( thay đổi kích thước, phác đồ, phối cảnh đối tượng, sửa đổi dựa trên bảng chung), chuyển đổi không gian màu và biểu đồ,..

+ Video: mô đun phân tích video bao gồm ước lượng chuyển động, phép trừ nền và thuật toán theo dõi đối tượng.

+ Calib3d: các thuật toán hình học đa khung cơ bản, hiệu chuẩn máy ảnh đơn và âm thanh nổi, ước lượng các đối tượng, thuật toán tương phản và các yếu tố tái tạo 3d.

+ Features2d: trình do đặc tính nổi bật, mô tả và trình kết hợp mô tả.

+ Objdetect: phát hiện các đối tượng và trường hợp của các lớp được xác định trước ( ví dụ: khuôn mặt, mắt, người, xe hơi … )

+ Hightgui: một giao diện dễ sử dụng với các tính năng UI đơn giản.

+ Videoio: một giao diện dễ sử dụng để thu thập video và nén video.

+ GPU: các thuật toán tăng tốc GPU từ các mô đun OpenCV khác nhau

+ Một số mô đun khác hổ trợ người dùn, chẳng hạn như FLANN và trình bao bọc thử nghiệm của Google, các liên kết Python và những thứ khác.

1. Ứng dụng của OpenCV:

* Bộ công cụ tính năng 2D và 3D.
* Định giá.
* Hệ thống nhận dạng khuôn mặt
* Nhận dạng cử chỉ.
* Tương tác giữa con người và máy tính. (HCI)
* Robot di động
* Sự hiểu biết về chuyển động
* Nhận dạng đối tượng
* Sự phân khúc và nhìn nhận
* Tầm nhìn âm thanh nổi Stereopsis: nhận thức sâu sắc từ hai camera.
* Cấu trúc từ chuyển động (SFM)
* Theo dõi chuyển động.
* Thực tế gia tăng.

Để hỗ trợ một số các lĩnh vực trên, OpenCV bao gồm một thư viện học máy thống kê bao gồm:

* Tăng áp lượng (Boosting)
* Cây quyết định
* Cải thiện độ dốc cây
* Thuật toán tối đa hóa kỳ vọng.
* Thuật toán k-nearest neighbor.
* Phân loại Naive Bayes.
* Mạng lưới thần kinh nhân tạo.
* Rừng ngẫu nhiên.
* Máy vector hỗ trợ (SVM).

1. Ứng dụng cùng với sự hỗ trợ Cuda:

* Giới thiệu:

+ Bộ gia tốc GPU hiện tại đã trở nên mạnh mẽ và có khả năng đủ để thực hiện tính toán mục đích chung (GPGPU). Đây là một xu hướng đang phát triển rất nhanh, tạo ra rất nhiều quan tâm từ các nhà khoa học, các nhà nghiên cứu và các kỹ sư phát triển ứng dụng đòi hỏi nhiều tính toán. Mặc dù gặp khó khăn trong việc triển khai các thuật toán về GPU, nhiều người đang làm việc này để kiểm tra tốc độ của chúng. Để hỗ trợ những nỗ lực này, có rất nhiều ngôn ngữ và công cụ tiên tiến đã có sẵn như CUDA, OpenCL, C++ AMP, trình gỡ lỗi, trình hồ sơ và nhiều công cụ khác.

+ Phần quan trọng của Computer Vision là xử lý hình ảnh, khu vực mà các bộ tăng tốc đồ họa ban đầu được thiết kế cho. Các bộ phận khác cũng giả sử tính toán song song lớn và thường bản đồ tự nhiên để kiến trúc GPU. Vì vậy nó là thách thức nhưng rất đáng để thực hiện tất cả các lợi thế và đẩy nhanh OpenCV về bộ xử lý đồ họa.

* Mục đích:

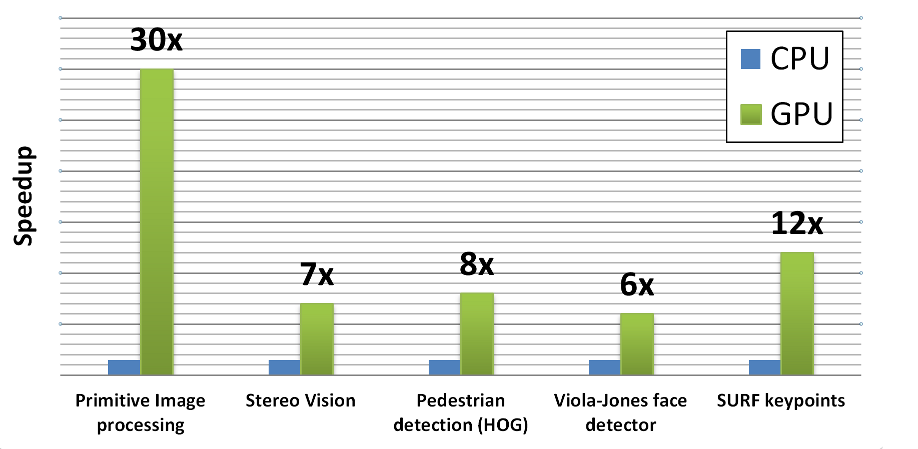
+ Cung cấp cho các nhà phát triển với một khung nhìn về máy tính thuận tiện trên GPU, duy trì tính nhất quán về khái niệm với chức năng CPU hiện tại.

+ Đạt được hiệu năng tốt nhất với GPU ( hạt nhân hiệu quả được điều chỉnh cho các kiến trúc hiện đại, tối ưu hóa luồng dữ liệu như thực hiện đồng thời, sao chép chồng chéo, không sao chép).

+ Tính tròn vẹn ( thực hiện càng nhiều càng tốt, ngay cả khi tốc độ không phải là điều tuyệt vời nhất, như là cho phép chạy một thuật toán hoàn toàn trên GPU).

* Hiệu năng:

Tesla C2050 versus Core i5-760 2.8Ghz, SSE, TBB



+ Mô đun GPU OpenCV được viết bằng CUDA, do đó nó có lợi từ hệ sinh thái CUDA. Có một cộng đồng lớn, các hội nghị, ban hành, những công cụ và thư viện được phát triển như:  NVIDIA NPP, CUFFT, Thrust.

+ Mô đun GPU được thiết kế như là máy chủ API mở rộng. Thiết kế này cung cấp cho người dùng một sự kiểm soát rõ ràng về cách dữ liệu được chuyển giữa GPU và CPU. Mặc dù người dùng phải viết một số mã bổ sung để bắt đầu sử dụng GPU, cách tiếp cận này rất linh hoạt và cho phép tính toán hiệu quả hơn.

Các mô đun GPU bao gồm các lớp cb::gpu::GpuMath, đây là nơi chứa chính cho các dữ liệu được giữ trong bộ nhớ GPU. Đó là giao diện rất giống với cv::Math, CPU tương ứng của nó. Tất cả các chức năng của GPU nhận GpuMat làm các tham số đầu vào và đầu ra. Điều này cho phép một số thuật toán GPU mà không cần tải dữ liệu. Giao diện GPU API cũng tương tự giao diện CPU. Vì vậy các nhà phát triển đã quen thuộc với OpenCV trên CPU có thể bắt đầu sử dụng GPU ngay lập tức.

## OpenMP:

1. Khái niệm

* OpenMP (Open Multi-processing) là một giao diện lập trình ứng dụng (API) hỗ trợ lập trình đa xử lý chia sẻ bộ nhớ đa nền tảng trên C, C++, Fortran trên hầu hết các nền tảng, kiến trúc tập lệnh và các hệ điều hành, bao gồm: Solaris, AIX, HP-UX, Linux, macOS, và Windows. Nó bao gồm một tập hợp các chỉ thị trình biên dịch, các vòng lặp của thư viện và các biến môi trường ảnh hưởng đến thời gian thực thi.
* OpenMP được quản lý bởi hiệp hội công nghệ phi lợi nhuận OpenMP Architecture Review Board (hay OpenMP ARB) cùng được xác định bởi các nhóm nhà cung cấp phần cứng và phần mềm máy tính lớn, bao gồm AMD, IBM, Intel, Cray, HP, Fujitsu, Nvidia, NEC, Red Hat, Texas Instruments, Oracle Corporation và nhiều hơn nữa.
* OpenMP sử dụng một mô hình di động, có thể mở rộng cho các lập trình viên một giao diện đơn giản và linh hoạt để phát triển các ứng dụng song song cho các nền tảng khác nhau, từ máy tính để bàn đến tiêu chuẩn siêu máy tính.
* Một ứng dụng được xây dựng với mô hình của lập trình song song có thể chạy trên một cụm máy tính sử dụng cả OpenMP và Message Passing Interface (MPI), như vậy OpenMP được sử dụng cho tính song song trong một nốt (đa lõi) trong khi MPI được sử dụng song song giữa các nốt. Có những nổ lực để chạy OpenMP trên phần mềm hệ thống phân phối của các hệ thống bộ nhớ chia sẻ, để chuyển OpenMP vào MPI và mở rộng OpenMP cho các hệ thống bộ nhớ không chia sẻ.

1. Thiết kế:

* OpenMP là một ứng dụng đa luồng, một phương pháp song song, theo đó một luồng chính ( một loạt các lệnh thực hiện liên tiếp) chia ra thành một số lượng các luồng phụ thuộc và hệ thống phân chia một nhiệm vụ giữa chúng. Các luồng sau đó chạy đồng thời với môi trường thời gian chạy phân bố luồng cho các bộ xử lý khác nhau.
* Theo mặc định, mỗi luồng thực hiện đoạn mã song song một cách độc lập. Các cấu trúc chia sẻ công việc có thể được sử dụng để phân chia một tác vụ giữa các luồng thực thi một phần được phân bố của mã. Cả song song công việc và song song dữ liệu có thể đạt được bằng cách sử dụng OpenMP theo cách này.

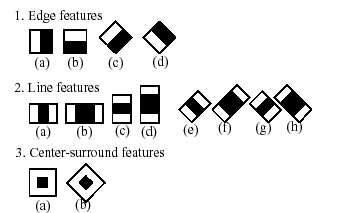
# Cách thức hoạt động:

## Quá trình hoạt động :

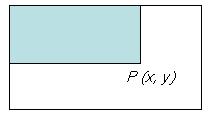
1. Phương pháp :

Hiện này có rất nhiều phương pháp để nhận dạng khuôn mặt người, dựa vào tính chất của các phương pháp, ta có thể chia làm 4 loại chính như sau: Hướng tiếp cận dựa trên tri thức: mã hóa hiểu biết của con người về các loại khuôn mặt và tạo ra các tập luật để xác định khuôn mặt; hướng tiếp cận dựa trên đặc tả không thay đổi: mục tiêu của các thuật toán là tìm ra các đặt trưng mô tả cấu trúc khuôn mặt người ( các đặc trưng không thay đổi với tư thế, vị trí đặt thiết bị thu hình, độ sáng tối thay đổi,…); hướng tiếp cận dựa trên so khớp mẫu: dùng các mẫu chuẩn hay các đặc trưng của khuông mặt người; hướng tiếp cận dựa trên diện mạo: phương pháp học từ một tập ảnh huấn luyện để xác định khuôn mặt người. Chương trình xác định vị trí khuôn mặt, mắt, miệng mà chúng tôi thiết kế là dựa trên hướng tiếp cận dựa trên diện mạo, sử dụng bộ phân loại mạnh Adaboost là sự kết hợp của các bộ phân loại yếu dựa trên các đặc trưng Haar-like để xác định khuôn mặt, mắt, miệng. Đặc trưng Haar-like do Viola và Jone công bố:

* Đầu tiên, một classifier (cụ thể là một cascade của các classifier boosted làm việc với các tính năng giống Haar-like) được huấn luyện với vài trăm quan điểm mẫu của một đối tượng cụ thể (trong hệ thống của chúng tôi là khuôn mặt của người, mắt, miệng) , được gọi là các ví dụ tích cực, được quy mô đến cùng kích thước (20x20) và các ví dụ tiêu cực – những hình ảnh tùy ý có cùng kích thước: con vật, xe hơi, hoa quả…
* Sau khi trình phân loại đã được huấn luyện , nó có thể áp dụng được cho khu vực quan tâm có cùng kích thước như được sử dụng trong quá trình huấn luyện) trong một hình ảnh đầu vào. Trình phân loại sẽ xuất ra một “1” nếu khu vực này có hiển thị đối tượng chúng tôi quan tâm (tức là khuôn mặt, mắt, miệng ) và là “0” nếu không xuất hiện các đối tượng đó. Để tìm kiếm đối tượng trong toàn bộ hình ảnh, bạn có thể di chuyển cửa sổ tìm kiếm qua hình ảnh và kiểm tra mọi vị trí sử dụng trình phân loại. Trình phân loại được thiết kế sao cho có thể dễ dàng thay đổi kích cỡ để có thể tìm thấy các đối tượng quan tâm ở các kích cỡ khác nhau, hiệu quả hơn khi thay đổi kích thước của hình ảnh. Vì vậy để tìm kiếm một đối tượng với kích thước không rõ ràng trong hình ảnh, quá trình quét nên được thực hiện nhiều lần ở các quy mô kích thước khác nhau.
* Từ “cascade” trong “phân loại” có nghĩa là phân loại kết quả bao gồm một số phân loại đơn giản (giai đoạn) sau đó được áp dụng đến một khu vực quan tâm cho đến khi giai đoạn nào đó ứng cử viên bị từ chối hoặc giai đoạn đó được thông qua. Từ “boosted” ( thúc đẩy) có nghĩa là trình phân loại ở mọi giai đoạn của tầng tự tạo phức tạp và chúng được xây dựng từ các phân loại cơ bản sử dụng một trong bốn kỹ thuật boosted khác nhau ( bỏ phiếu có trọng số).  Currently Discrete Adaboost, Real Adaboost, Gentle Adaboost and Logitboost đều được hỗ trợ. Các phân lọa cơ bản là các bộ phân loại cây quyết định có ít nhất hai lá. Các tính năng Haar-like là đầu vào cho các phân loại cơ bản, được mô tả như hình dưới. Thuật toán hiện tại được sử dụng các tính năng Haar-like như sau:



* Tính năng được sử dụng trong một bộ phận phân loại đặc biệt được xây dựng bởi hình dạng của nó (1a, 2b, …), vị trí trong khu vực quan tâm và quy mô ( tỷ lệ này không bằng thang đo được sử dụng ở giai đoạn phát hiện, mặc dù hai thang đo này là nhân). Ví dụ trong trường hợp tính năng dòng thứ 3 (2c), phản hồi được tính như là sự khác biệt giữa tổng các điểm ảnh dưới hình chữ nhật bao phủ toàn bộ tính năng (bao gồm cả sọc trắng và sọc đen ở giữa ) và tổng các điểm ảnh hình ảnh dưới dải đen nhân với ba để bù đắp cho sự khác biệt trong kích thước của khu vực. Các khoản giá trị của pixel trên một vùng hình chữ nhật trong tính toán nhanh chóng bằng cách sử dụng các hình ảnh tích phân.
* Viola và Jones đưa ra một khái niệm gọi là Integral Image, là một mảng 2 chiều với kích thước bằng với kích của ảnh cần tính các đặc trưng Haar-like. Bắt đầu từ vị trí trên, bên trái đến vị trí dưới, phải của ảnh, việc tính toán này đơn thuần chỉ đựa trên phép cộng số nguyên đơn giản, do đó tốc độ thực hiện rất nhanh.



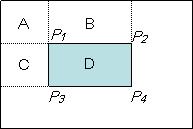
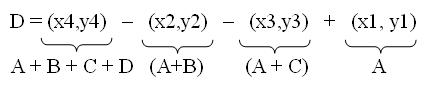
http://2.bp.blogspot.com/-_sc_-nfrj1s/TzI5hpdtXFI/AAAAAAAAAfQ/LAntPDhSXCI/s1600/IntegralImag-2-new.gif.jpg

* Sau khi đã tính được Integral Image, việc tính tổng các giá trị mức xám của một vùng bất kỳ nào đó trên ảnh thực hiện rất đơn giản theo cách sau:

+ Giả sử ta cần tính tổng các giá trị mức xám của vùng D như trong hình dưới , ta có thể tính như sau:

D = A + B + C + D – (A+B) – (A+C) + A

+ Với A + B + C + D chính là giá trị tại điểm P4 trên Integral Image, tương tự như vậy A+B là giá trị tại điểm P2, A+C là giá trị tại điểm P3, và A là giá trị tại điểm P1. Vậy ta có thể viết lại biểu thức tính D ở trên như sau:

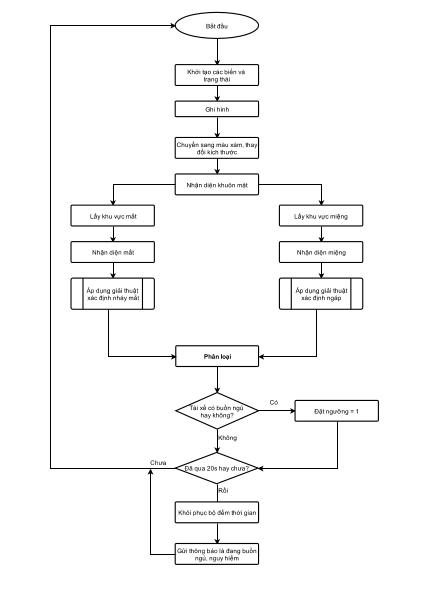


1. Cách xác định:

Hệ thống duyệt qua từng khung hình mà máy ảnh gửi đến, trong mỗi khung hình, hệ thống duyệt qua lần lượt từng khu vực nhỏ, với kích thước thích hợp, và áp dụng giải thuật xác định đối tượng - trình bày ở trên, đưa ra kết quả là khu vực đó có tồn tại đối tượng cần tìm kiếm hay không, và tương tự với những khung hình khác. Sau khi quá trình xác định hoàn tất, hệ thống trả về danh sách các khu vực mà hệ thống xác định là có đối tượng cần tìm kiếm.

## Các bước thực hiện:

Hệ thống hoạt động theo sơ đồ như sau:



1. Nhận dạng khuôn mặt

* Phần đầu tiên của giải thuật nhận biết buồn ngủ là nhận diện khuôn mặt. Bởi vì phương pháp của chúng tôi yêu cầu phải phân tích và phát hiện hình ảnh trong mỗi khung hình, đồng thời chạy qua lần lượt từng khung hình, vì vậy vấn đề về hiệu năng thì cực kì quan trọng.
* Sau quá trình đánh giá các giải pháp có thể giải quyết vấn đề này, chúng tôi nhận thấy rằng giải thuật của Viola và Jones tương thích với nhu cầu của bài toán này, với khoảng 200 đặc điểm khác nhau, tỷ lệ nhận dạng khoảng 95%, một tỷ lệ khá cao, và có thể chấp nhận được trong trường hợp này.
* Hệ thống sử dụng camera để lấy những khung hình, qua đó gửi những khung hình này đến hệ thống detect. Danh sách những khu vực có khuôn mặt do hệ thống phát hiện sẽ là đầu vào cho phần nhận diện miệng và mắt, chớp mắt và ngáp.

1. Nhận dạng Mắt

* Thay vì áp dụng cách tìm kiếm toàn bộ khu vực trong mỗi khung hình để nhận diện mắt ( cách làm này sẽ tốn thời gian tương đương phần nhận diện khuôn mặt, tuy nhiên kết quả tìm kiếm sẽ dễ bị những hình ảnh như cúc áo, ốc vít làm nhiễu kết quả) thì ta sẽ sử dụng kết quả của phần Nhận diện khuôn mặt để làm đầu vào cho phần này, giúp giảm thiểu khu vực cần tìm kiếm, giảm thời gian tìm kiếm đồng thời kết quả tìm kiếm trong khu vực này cũng sẽ mang nhiều ý nghĩa hơn bởi vì mắt chỉ nằm trên khuôn mặt mới có ý nghĩa
* Để tiết kiệm thời gian hơn nữa, thì thay vì tìm kiếm trên toàn bộ khung hình chứa khuôn mặt, ta sẽ chỉ tìm kiếm phần nửa trên của khuôn mặt.
* Bởi vì việc nháy mắt đa số xảy ra khi cả 2 mắt cùng nhắm, do đó ta chỉ xét đến 2 trường hợp 1) nhận diện được cả 2 mắt và 2) không nhận diện được mắt.

1. Nhận dạng Miệng

* Tương tự như phần Nhận diện Mắt, ta cũng chỉ tìm kiếm vùng miệng trên một nửa dưới của khuôn mặt. Nhờ việc này, diện tích tìm kiếm thay vì 100% thì chỉ còn 50%, đem đến tốc độ 2x cho việc tìm kiếm này.
* Để hiện thực việc này, ta sử dụng một hàm có sẵn trong OpenCV để hỗ trợ, hàm này hiện thực giải thuật đã nêu ở đầu tiên. Nó nhận một số tham số để có thể giúp việc tìm kiếm diễn ra nhanh hơn và hiệu suất cao hơn.
* Ví dụ:

+ minSize: kích thước nhỏ nhất của hình ảnh cần tìm kiếm, (20, 20).

+ minNeighbor: tham số này quy định số lượng khu vực láng giềng mà một khu vực cần có để được công nhận.

+ scaleFactor: tham số chỉ ra rằng hình ảnh có thể co giãn để có thể tìm kiếm.

1. Nhận dạng Nháy mắt

Để có thể xác định được nháy mắt mà thời gian của nháy mắt, ta sẽ sử dụng ít nhất 3 khung hình. 2 khung hình dùng để xác định 1 lần nháy mắt, và 1 khung hình còn lại để xác định độ dài của nháy mắt. Để hỗ trợ việc này, ta sẽ sử dụng thư việc thời gian của Python. Sau khi nháy mắt mà độ dài nháy mắt được xác định, các thông số này được lưu vào 2 dãy, 1 dãy lưu trữ thời gian nháy mắt diễn ra, dãy còn lại lưu trữ độ dài của mỗi lần nháy mắt, phục vụ việc tính thời gian nháy mắt trung bình.

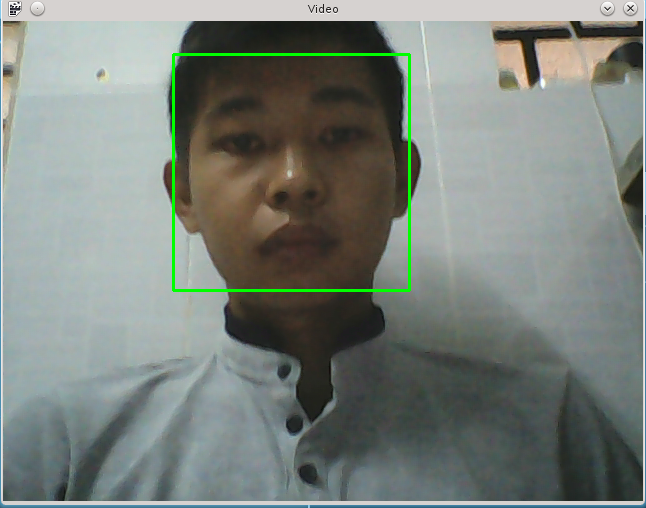
1. Nhận dạng Ngáp

* Tương tự như việc xác định nháy mắt, việc xác định ngáp cũng sẽ áp dụng cách làm tương tự. Tuy nhiên việc xác định ngáp sẽ khó xác định hơn so với xác định nháy mắt, do đó chúng tôi quyết định đặt ra một ngưỡng là 5 khung hình mà không tìm thấy miệng, sẽ xác định 1 lần ngáp.
* Có nhiều trường hợp có thể dẫn đến việc nhận diện sai, ví dụ như tài xế đưa mắt nhìn xung quanh xem đường, nói chuyện lớn, hoặc lấy tay che miệng…

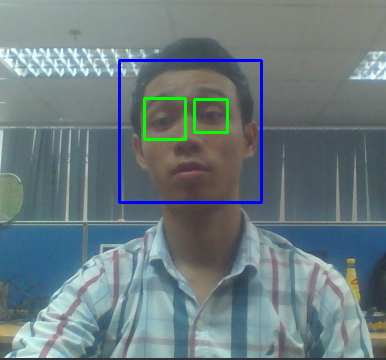
# Kết quả:

Sau khi hiện thực những thành phần trên của hệ thống, sau đây là kết quả mà nhóm đã thu được đến thời điểm này:

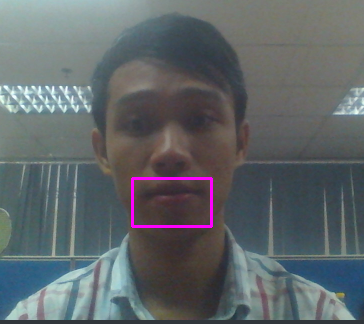
* Nhận diện khuôn mặt:



* Nhận diện khuôn mặt + mắt:

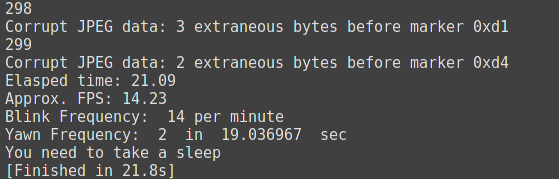


* Nhận diện miệng:



Từ những thông số sau khi ghi hình được, hệ thống tính toán được tần suất nháy mắt, ngáo, qua đó nhận định xem là tài xế có đang buồn ngủ hay không.

Kết quả sau cùng, hệ thống nhận định rằng người dùng đang buồn ngủ:



Gửi khuyến nghị dừng lại và nghỉ ngơi.

# Kết luận:

## Các công việc đã thực hiện:

Trong giới hạn thời gian của môn Thực Tập Tốt Nghiệp, chúng tôi đã thực hiện một số công việc chính như sau:

* Tìm hiểu OpenCV, Cuda, OpenMP.
* Hiện thực phần ứng dụng để xác định được khuôn mặt, miệng, mắt đề đưa ra cảnh báo.
* Tối ưu hóa việc huấn luyện tập dữ liệu dựa trên ứng dụng CUDA.

## Kinh nghiệm thu được

Sau khi hoàn thành đề tài này, chúng tôi đã thu được nhiều kinh nghiệm quý giá có ý nghĩa lớn trong học tập, nghiên cứu cũng như trong công việc thực tế sau này:

* Hiểu rõ cách hoạt động của một hệ thống xác định được khuôn mặt và đưa ra sự cảnh báo hợp lý và đúng thời điểm.
* Làm quen hơn với hệ điều hành Linux: thực hiện các thao tác, các lệnh, cũng như xây dựng các công cụ trên hệ điều hành Linux giúp chúng tôi làm quen hơn với nó
* Kỹ năng lên kế hoạch, phân chia và sắp xếp công việc: với quỹ thời gian có hạn, chúng tôi đã học được cách lên kế hoạch, cũng như phân chia và sắp xếp công việc một cách hợp lý sao cho đạt được các mục tiêu đã đề ra
* Kỹ năng tìm kiếm và xử lý thông tin: để có được kết quả tốt, chúng tôi phải tìm hiểu thông tin từ rất nhiều nguồn, qua đó phải xử lý để có được các thông tin cần thiết, phục vụ cho mục đích cuối cùng

## Hạn chế và khó khăn

Trong giai đoạn thực hiện đề tài này, chúng tôi đã gặp phải một số vấn đề khó khăn và dẫn đến một số hạn chế của đề tài:

* Vì hạn chế về thời gian, chúng tôi chỉ mới huấn luận được tập tin để xác định các đối tượng chưa chính xác lắm.
* Việc cài đặt OpenCV với hỗ trợ Cuda toolkit còn gặp nhiều khó khăn do gặp nhiều vấn đề khi cài đặt, chưa có nhiều nguồn để tham khảo và sửa chữa lỗi.
* Hiện tại chúng tôi mới hiện thực được mô phỏng sử dụng trực tiếp camera của máy tính cá nhân, chưa có công cụ sử dụng camera trong thực tế trên phương tiện giao thông và xây dựng server để tổng hợp dữ liệu được thu thập về.

Tuy nhiên, với nền tảng đã tìm hiểu được về OpenCV và CUDA, chúng tôi tự tin rằng có thể hoàn thiện chức năng tổng hợp dữ liệu và đưa ra cảnh báo chính xác kịp thời cho nhiều người dùng cùng sử dụng.

## Hướng phát triển trong tương lai

Trong tương lai, chúng tôi dự định sẽ thực hiện một số công việc sau đây:

* Hoàn thiện việc xây dựng file huấn luyện hoàn thiện hơn.
* Xây dựng ứng dung để có thể sử dụng trong thực tế.
* Xây dựng server để tổng hợp và phân tích thông tin.
* Tìm hiểu thêm về phần Cuda để tăng tốc việc xử lý dữ liệu.
* Phía tài xế: phát triển các ứng dụng chạy trên Smartphone hoặc Tablet, ứng dụng này sẽ xác định khuôn mặt qua camera thời gian thực.
* Phía server: phát triển giải thuật học máy qua Deep Learning để tạo ra các mạng nhận diện chính xác hơn cho ứng dụng. Phát triển giải thuật Deep-learning dùng các framework như Google Tensorflow, Intel DAAL, Nvidia Cafee, … để tạo ra các mạng học máy.

# Tham khảo:

<https://devtalk.nvidia.com/default/topic/951477/are-tools-like-opencv_traincascade-gpu-accelerated-in-opencv4tegra-/>

<http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html>

<http://docs.opencv.org/trunk/d7/d8b/tutorial_py_face_detection.html>

<https://en.wikipedia.org/wiki/Haar-like_features>

<https://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html>

<http://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html>