TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN 1**

**NGHIÊN CỨU XÂY DỰNG ỨNG DỤNG NHẬN BIẾT TÍN HIỆU ĐÈN GIAO THÔNG QUA CAMERA**

*Người hướng dẫn*: **TS.PHẠM VĂN HUY**

*Người thực hiện*: **NGUYỄN CÔNG MINH - 51303338**

**NGUYỄN QUANG HUY- 51303302**

Lớp **: 13050303**

Khoá  **: 17**

**THÀNH PHỐ HỒ CHÍ MINH NĂM 2017**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN 1**

**NGHIÊN CỨU XÂY DỰNG ỨNG DỤNG NHẬN BIẾT TÍN HIỆU ĐÈN GIAO THÔNG QUA CAMERA**

*Người hướng dẫn*: **TS.PHẠM VĂN HUY**

*Người thực hiện*: **NGUYỄN CÔNG MINH - 51303338**

**NGUYỄN QUANG HUY- 51303302**

Lớp **: 13050303**

Khoá  **: 17**

**THÀNH PHỐ HỒ CHÍ MINH NĂM 2017**

LỜI CẢM ƠN

Chúng em xin cảm ơn thầy Phạm Văn Huy và các bạn trường Đại học Tôn Đức Thắng khoa Công Nghệ Thông Tin đã giúp chúng em hoàn thành tốt đồ án này. Bái báo cáo có thể có sai sót mong thầy cô thông cảm.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng chúng tôi và được sự hướng dẫn của TS.Phạm Văn Huy. 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 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 chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Nguyễn Công Minh*

*Nguyễn Quang Huy*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

# MỤC LỤC

[DANH MỤC HÌNH VẼ, ĐỒ THỊ 0](#_Toc482917055)

[CHƯƠNG 1. GIỚI THIỆU 0](#_Toc482917056)

[I. Giới thiệu về thị giác máy tính 0](#_Toc482917057)

[II. Giới thiệu về openCV 0](#_Toc482917058)

[1. Cấu trúc openCV 1](#_Toc482917059)

[2. Không gian màu 2](#_Toc482917060)

[III. Haar – Cascade 4](#_Toc482917061)

[1. Haar – like feature 5](#_Toc482917062)

[2. Cascade Classifier 5](#_Toc482917063)

[3. Negative Sample 6](#_Toc482917064)

[4. Positive Sample 6](#_Toc482917065)

[IV. 7](#_Toc482917066)

[V. S 7](#_Toc482917067)

[CHƯƠNG 2. DEMO ỨNG DỤNG 8](#_Toc482917068)

[VI. Ý tưởng 8](#_Toc482917069)

[VII. Hiện thực ứng dụng 8](#_Toc482917070)

[1. Thiết lập 8](#_Toc482917071)

[2. Quá trình huấn luyện 8](#_Toc482917072)

[3. Phân lớp và khoanh vùng đối tượng 11](#_Toc482917073)

[4. Nhận biết màu 15](#_Toc482917074)

[VIII. Kết luận 17](#_Toc482917075)

[1. Ưu điểm 17](#_Toc482917076)

[2. Ngược điểm 17](#_Toc482917077)

[3. Hướng phát triển 17](#_Toc482917078)

[TÀI LIỆU THAM KHẢO 18](#_Toc482917079)

# DANH MỤC HÌNH VẼ, ĐỒ THỊ

[Hình II‑1 Logo OpenCV 0](#_Toc482917337)

[Hình II‑2 Cấu trúc openCV. 2](#_Toc482917338)

[Hình II‑3 Dải màu HSV. 4](#_Toc482917339)

[Hình III‑1 Quá trình huấn luyện trên console. 5](#_Toc482917340)

[Hình III‑2 Khoanh vùng đối tượng (không nhiễu). 6](#_Toc482917341)

[Hình III‑3 Khoanh vùng đối tượng (nhiễu nhiều). 6](#_Toc482917342)

[Hình VII‑1 Các tập tin trong thư mục Haar training. 9](#_Toc482917343)

[Hình VII‑2 Tạo danh sách Negative. 9](#_Toc482917344)

[Hình VII‑3 Tạo danh sách Positive. 10](#_Toc482917345)

[Hình VII‑4 Danh sách Positive sau khi tạo xong. 10](#_Toc482917346)

[Hình VII‑5 Chỉnh sửa thông số huấn luyện. 11](#_Toc482917347)

[Hình VII‑6 Thông số trong tập tin xml sau khi luyện. 11](#_Toc482917348)

[Hình VII‑7 Đèn giao thông dạng 1. 12](#_Toc482917349)

[Hình VII‑8 Đèn giao thông dạng 2. 12](#_Toc482917350)

[Hình VII‑9 Đèn giao thông dạng 3. 13](#_Toc482917351)

[Hình VII‑10 Khoanh vùng đèn giao thông dạng 2. 14](#_Toc482917352)

[Hình VII‑11 Khoanh vùng đèn giao thông dạng 3 (không nhiễu). 14](#_Toc482917353)

[Hình VII‑12 Khoanh vùng đèn giao thông dạng 3 (có nhiễu). 14](#_Toc482917354)

[Hình VII‑13 Nhận diện tín hiệu đèn đỏ. 16](#_Toc482917355)

[Hình VII‑14 Nhận diện tín hiệu đèn xanh. 16](#_Toc482917356)

# GIỚI THIỆU

## Giới thiệu về thị giác máy tính

Cặp mắt là một trog những phần quan trọng nhất của con người,nó giúp ta nhìn thấy biết được màu sắc,hình dáng… của mọi sự vật.Có thể nói,cặp mắt là món quà mà con người có được từ khi mới sinh ra.

Thế nhưng những vật vô tri như máy tính,camera,robot.. thì vốn không thể “nhìn” như con người,vì thế lĩnh vực Thị giác máy tính ra đời để tìm ra các giải pháp giúp cho những vật ấy có thể “nhìn” cũng như nhận biết các sự vật trong cược sống.

Một cách tổng quát, **Thị giác máy tính** (**computer vision**) là một lĩnh vực bao gồm các phương pháp thu nhận, [xử lý ảnh kỹ thuật số](https://vi.wikipedia.org/w/index.php?title=X%E1%BB%AD_l%C3%BD_%E1%BA%A3nh_k%E1%BB%B9_thu%E1%BA%ADt_s%E1%BB%91&action=edit&redlink=1), phân tích và nhận dạng các hình ảnh, nói chung là dữ liệu đa chiều từ thế giới thực để cho ra các thông tin số hoặc biểu tượng.Việc phát triển lĩnh vực này bắt nguồn từ việc sao chép các khả năng thị giác con người bởi sự nhận diện và hiểu biết một hình ảnh mang tính điện tử. Sự nhận diện hình ảnh có thể xem là việc giải quyết vấn đề của các biểu tượng thông tin từ dữ liệu hình ảnh qua cách dùng các mô hình được xây dựng với sự giúp đỡ của các ngành [lý thuyết học](https://vi.wikipedia.org/w/index.php?title=L%C3%BD_thuy%E1%BA%BFt_h%E1%BB%8Dc&action=edit&redlink=1), [thống kê](https://vi.wikipedia.org/wiki/Th%E1%BB%91ng_k%C3%AA), [vật lý](https://vi.wikipedia.org/wiki/V%E1%BA%ADt_l%C3%BD) và [hình học](https://vi.wikipedia.org/wiki/H%C3%ACnh_h%E1%BB%8Dc).

## Giới thiệu về openCV

OpenCV (Open Source Computer Vision) là một thư viện mã nguồn mở về thị giác máy với hơn 500 hàm và hơn 2500 các thuật toán đã được tối ưu về xử lý ảnh, và các vấn đề liên quan tới thị giác máy. OpenCV là thư viện được thiết kế để chạy trên nhiều nền tảng khác nhau (cross-patform) như Window, Linux, Mac, iOS …



Hình II‑1 Logo OpenCV

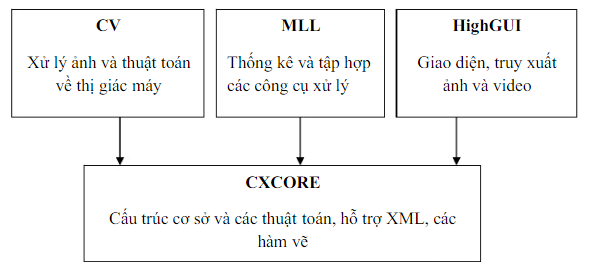
Dự án về OpenCV được khởi động từ những năm 1999, đến năm 2000 nó được giới thiệu trong một hội nghị của IEEE về các vấn đề trong thị giác máy và nhận dạng, tuy nhiên bản OpenCV 1.0 mãi tới tận năm 2006 mới chính thức được công bố và năm 2008 bản 1.1 pre-release) mới được ra đời. Tháng 10 năm 2009, bản OpenCV thế hệ thứ hai ra đời, phiên bản này có giao diện của C++ (khác với phiên bản trước có giao diện của C) và có khá nhiều điểm khác biệt so với phiện bản thứ nhất.

Thư viện OpenCV ban đầu được sự hỗ trợ từ Intel, sau đó được hỗ trợ bởi Willow Garage, một phòng thí nghiệm chuyên nghiên cứu về công nghệ robot. Cho đến nay, OpenCV vẫn là thư viện mở, được phát triển bởi nguồn quỹ không lợi nhuận (none -profit foundation) và được sự hưởng ứng rất lớn của cộng đồng.

### Cấu trúc openCV

Thư viện OpenCV có thể được chia thành 2 phần (module) chính. Phần căn bản (basic) là mã nguồn được nhóm phát triển xây dựng và kiểm định toàn diện, gồm các thuật toán đã được thế giới công nhận và đánh giá dựa trên cơ sở lý thuyết chắc chắn. Phần mở rộng (contribution) được nhiều tổ chức khoa học khác nhau trên thế giới đóng góp, gồm nhiều thuật toán cập nhật được xây dựng dựa trên các công trình nghiên cứu, bài báo mới đăng trong thời gian gần đây. Do vậy các thuật toán trong phần mở rộng có độ ổn định và tối ưu không cao. Từ phiên bản 3.0, phần mở rộng được tách riêng không còn được gộp chung với thư viện mặc định.

OpenCV là thư viện mã nguồn mở được đóng gói thành tập tin nén. Tùy vào phiên bản dành cho các hệ điều hành khác nhau mà tập tin nén này có định dạng tương ứng. Thư viện OpenCV cung cấp cho người dùng các cấu trúc dữ liệu, đối tượng và hàm bằng cách khai báo nguyên mẫu (prototype) của chúng trong các tập tin thư viện C/C++ (\*.h, \*.hpp,…) và định nghĩa chi tiết trong các tập tin mã nguồn (\*.c, \*.cpp). Với mức độ sử dụng OpenCV, chỉ cần giải nén các tập tin đã được biên dịch sẵn rồi thực hiện các thao tác cài đặt đường dẫn cho thích hợp để hệ điều hành tìm đến đúng vị trí của các tập tin thư viện. Ở mức độ cao hơn, nếu muốn hiệu chỉnh sửa đổi thuật toán hay sử dụng phần mở rộng của OpenCV ta cần phải biên dịch mã nguồn trực tiếp trên máy trước khi cài đặt.



Hình II‑2 Cấu trúc openCV.

**CXCORE:** Chứa các định nghĩa kiểu dữ liệu cơ sở. Ví dụ, các cấu trúc dữ liệu cho ảnh, điểm và hình chữ nhật được định nghĩa trong cxtypes.h. CXCORE cũng chứa đại số tuyến tính và phương pháp thống kê, chức năng duy trì và điều khiển chuỗi. Một số ít, các chức năng đồ họa để vẽ trên ảnh cũng được đặt ở đây.

**CV**: Chứa các thuật toán về xử lý ảnh và định kích cỡ camera. Các chức năng hình họa máy tính cũng được đặt ở đây.

**CVAUX**: Dùng cho nhận diện mặt và chúng được ứng dụng rộng rãi cho mục đích đó.

**HIGHGUI**: Chứa các giao diện vào ra cơ bản, nó cũng chứa các khả năng mở rộng cửa sổ và vào ra video.

**CVCAM**: Chứa các giao diện cho video truy cập qua DirectX trên nềnWindows 32 bits.

### Không gian màu

Không gian màu là một mô hình toán học dùng để mô tả các màu sắc trong thực tế được biểu diễn dưới dạng số học. Trên thực tế có rất nhiều không gian màu khác nhau được mô hình để sử dụng vào những mục đích khác nhau. Trong openCV có ba không gian màu cơ bản hay được nhắc tới và ứng dụng nhiều, đó là hệ không gian màu RGB, HSV và CMYK.

#### Không gian màu RGB

RGB là không gian màu rất phổ biến được dùng trong đồ họa máy tính và nhiều thiết bị kĩ thuật số khác. Ý tưởng chính của không gian màu này là sự kết hợp của 3 màu sắc cơ bản : màu đỏ (R, Red), xanh lục (G, Green) và xanh lơ (B, Blue) để mô tả tất cả các màu sắc khác. Nếu như một ảnh số được mã hóa bằng 24bit, nghĩa là 8bit cho kênh R, 8bit cho kênh G, 8bit cho kênh B, thì mỗ kênh này màu này sẽ nhận giá trị từ 0-255. Với mỗi giá trị khác nhau của các kênh màu kết hợp với nhau ta sẽ được một màu khác nhau, như vậy ta sẽ có tổng cộng 255x255x255 = 1.66 triệu màu sắc.

#### Không gian màu CMYK

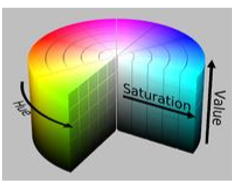
CMYK là không gian màu được sử dụng phổ biến trong ngành công nghiệp in ấn. Ý tưởng cơ bản của hệ không gian này là dùng 4 màu sắc cơ bản để phục vụ cho việc pha trộn mực in. Trên thực tế, 3 màu là C=Cyan: xanh lơ, M=Magenta: hồng xẫm, và Y=Yellow: vàng để biểu diễn các màu sắc khác nhau. Nếu lấy màu hồng xẫm cộng với vàng sẽ ra màu đỏ, màu xẫm kết hợp với xanh lơ sẽ cho xanh lam ... Sự kết hợp của 3 màu trên sẽ cho ra màu đen, tuy nhiên màu đen ở đây khôn phải là đen tuyệt đối và thường có độ tương phản lớn, nên trong ngành in, để tiết kiệm mực in người ta thêm vào màu đen để in những chi tiết có màu đen thay vì phải kết hợp 3 màu sắc trên.

Nguyên lý làm việc của hệ màu này như sau : Trên một nền giấy trắng, khi mỗi màu này được in lên sẽ loại bỏ dần đi thành phần màu trắng. 3 màu C, M, Y khác nhau in theo những tỉ lệ khác nhau sẽ loại bỏ đi thành phần đó một cách khác nhau và cuối cùng cho ta màu sắc cần in. Khi cần in màu đen, thay vì phải in cả 3 màu người ta dùng màu đen để in lên. Nguyên lý này khác với nguyên lý làm việc của hệ RGB ở chỗ hệ RGB là sự kết hợp của các thành phần màu, còn hệ CMYK là sự loại bỏ lẫn nhau của các thành phần màu.

#### Không gian màu HSV

HSV và cũng gần tương tự như HSL là không gian màu được dùng nhiều trong việc chỉnh sửa ảnh, phân tích ảnh và một phần của lĩnh vực thị giác máy tính. Hệ không gian này dựa vào 3 thông số sau để mô tả màu sắc H = Hue: màu sắc, S = Saturation: độ đậm đặc, sự bảo hòa, V = value: giá trị cường độ sáng.

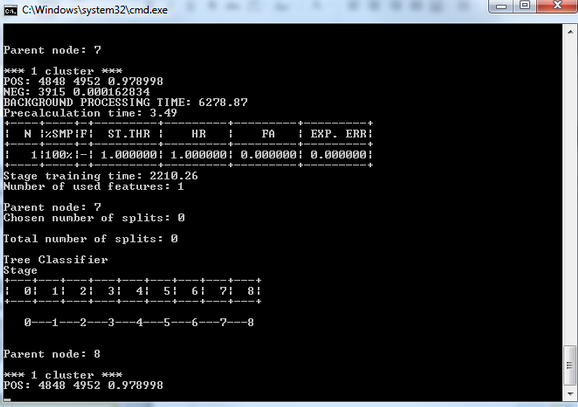
Không gian màu này thường được biểu diễn dưới dạng hình trụ hoặc hình nón. Theo đó, đi theo vòng tròn từ 0 -360 độ là trường biểu diễn màu sắc(Hue). Trường này bắt đầu từ màu đỏ đầu tiên (red primary) tới màu xanh lục đầu tiên (green primary) nằm trong khoảng 0-120 độ, từ 120 - 240 độ là màu xanh lục tới xanh lơ (green primary - blue primary). Từ 240 - 360 là từ màu đen tới lại màu đỏ.



Hình II‑3 Dải màu HSV.

## Haar – Cascade

Haar – Cascade là giải thuật kết hợp giữa phương thức Haar-like feature và phương pháp Cascade, được sử dụng trong xử lý ảnh về nhận dạng đối tượng (Object detection) bằng cách huấn luyện trên hai tập dữ liệu hình ảnh có sẵn là Positive và Negative thành file .xml hoặc .yml. Tổng số hình phải luyện là trên 1000 hình để chương trình có thể nhận diện đúng, càng nhiều hình thì kết quả nhận diện càng hiệu quả. Quá trình huấn luyện được gọi là Haar Training (Cascade Training), sau khi huấn luyện xong Cascade Classifier sẽ kiểm tra kết quả qua một mẫu thử (hình ảnh, video hoặc trực tiếp qua thiết bị ghi hình).



Hình III‑1 Quá trình huấn luyện trên console.

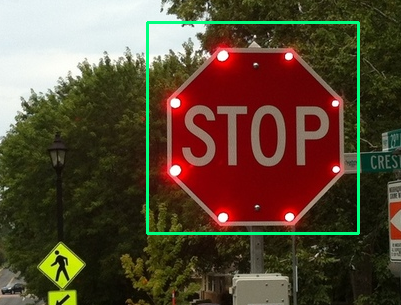
### Haar – like feature

Haar – like feature là một phương thức dùng để phát hiện đối tượng trong hình ảnh qua các cấu trúc cụ thể trong hình.

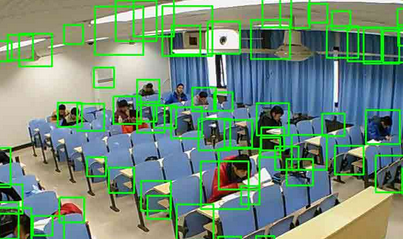
Hình được chia thành nhiều vùng cụ thể và giải thuật sẽ tra cứu trong từng điểm ảnh (pixel) tìm ra sự khác biệt. Những sự khác biệt đó sẽ được mã hóa thành các thông số và lưu trữ lại thông qua quá trình luyện.

### Cascade Classifier

Sau khi huấn luyện xong, chương trình sẽ gọi kết quả để kiểm tra (file .xml hoặc .yml). OpenCV có hỗ trợ lớp “CascadeClassifier” và hiện thực bằng cách gọi phương thức “detectMultiScale(param 1, param 2, …)”, phương thức trên sẽ phân loại qua các pixel và khoanh vùng các đối tượng bằng một vector<Shape>. Dựa vào vector<Shape> chương trình sẽ hiển thị trên hình ảnh cho người lập trình biết vùng nhận diện đối tượng bằng các hình chữ nhật hay hình tròn hoặc có thể là hình đa giác. Qua đó, người lập trình có thể biết được hiệu quả của tập huấn luyện hiện tại và có thể đưa ra cách chỉnh sửa tập huấn luyện đó để có kết quả chính xác hơn.



Hình III‑2 Khoanh vùng đối tượng (không nhiễu).



Hình III‑3 Khoanh vùng đối tượng (nhiễu nhiều).

### Negative Sample

Negative sample mang ý nghĩa là một mẫu tiêu cực, đây là các hình ảnh không chứa các đối tượng muốn nhận diện và có thể gọi là hình ảnh gây nhiễu. Các hình ảnh này thường là các phông nền, các đối tượng ở gần đối tượng muốn nhận diện,...Các hình ảnh negative này luyện càng nhiều thì mức độ khử nhiễu càng cao.

### Positive Sample

Ngược lại với Negative sample, Positive sample là mẫu tích cực gồm cac hình ảnh của đối tượng muốn nhận diện. Các hình ảnh này phải được luyện ở các góc độ khác nhau, các thời điểm khác nhau để lấy được nhiều thông tin về hình dạng của đối tượng nhận diện. Các hình ảnh positive này luyện càng nhiều thì độ chính xác càng cao.

## 

## S

# DEMO ỨNG DỤNG

## Ý tưởng

Ứng dụng sẽ sử dụng song song hai chức năng là khoanh vùng đèn giao thông và xác định màu của đèn. Đầu tiên, dựa vào tập huấn luyện xác định vị trí của đèn, hiển thị lên cho người dùng. Sau đó, duyệt trên các vùng đã xác định, kiểm tra xem có tồn tại một trong ba màu (xanh, đỏ, vàng) hay không. Nếu có thì sẽ hiển thị thông báo bằng cách vẽ một hình tròn có màu tương ứng dưới góc phải màn hình. Nếu ba màu trên không tồn tại trong vùng xác định thì sẽ không có thông báo nào cả để tránh hiện tượng nhận sai các vùng bị nhiễu.

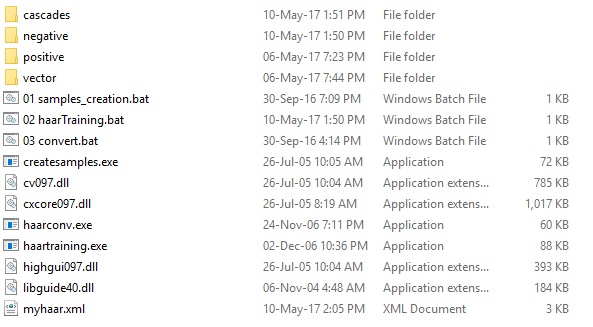
## Hiện thực ứng dụng

### Thiết lập

* OpenCV 3: Các phiên bản openCV 3.0, 3.1, 3.2, …
* Hệ điều hành: Window 10.
* Ngôn ngữ: Ứng dụng được viết bằng ngôn ngữ C++.
* Sử dụng phần mềm Visual Studio 2015 để lập trình.
* Hai tập dữ liệu hình ảnh Positive và Negative (mỗi tập khoảng hơn 500 hình):
  + Để có được tập dữ liệu hình ảnh ta phải tải từng ảnh trên internet như vậy sẽ rất mất thời gian nên chúng tôi đã sử dụng phần mềm **Total Video Converter. Tải tại:** [**http://download.cnet.com/Total-Video-Audio-Converter/3000-2194\_4-76208009.html**](http://download.cnet.com/Total-Video-Audio-Converter/3000-2194_4-76208009.html)
  + Add File (Video cần cắt ảnh) -> Option -> Chọn Frame rate 5000Hz (5 frame hình / giây) -> OK -> Nhấn Convert.

### Quá trình huấn luyện

Với hệ điều hành Window, người lập trình có thể tải xuống bộ công cụ luyện Haar-Training của Mahdi Rezaei về mảng khoa học máy tính tại Đại học Auckland, New Zealand (<https://www.cs.auckland.ac.nz/~m.rezaei/Tutorials/Haar-Training.zip>).

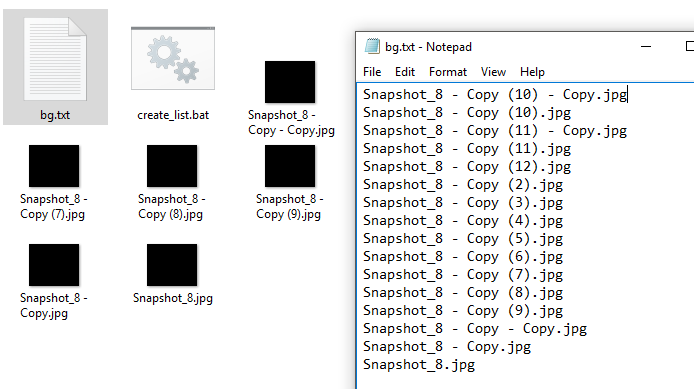


Hình VII‑1 Các tập tin trong thư mục Haar training.

* Bước 1: Tạo danh sách Negative.

- Chuyển tất cả dữ liệu hình Negative vào thư mục **rawdata** trong thư mục **negative**.

- Chạy file **create\_list.bat** để tạo file **bg.txt** chứa tên của các file ảnh

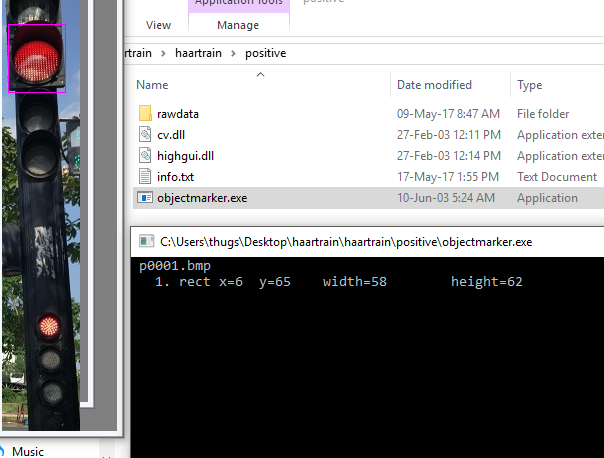


Hình VII‑2 Tạo danh sách Negative.

* Bước 2: Tạo danh sách Positive.

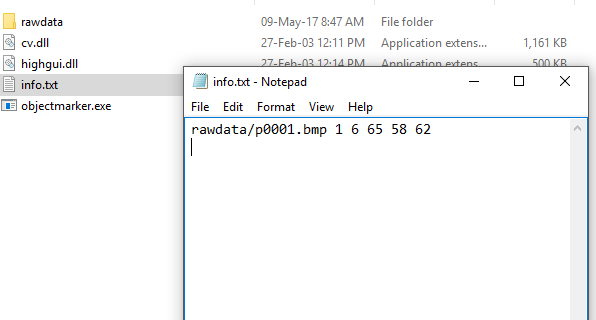
- Chuyển tất cả dữ liệu hình Positive vào thư mục **rawdata** trong thư mục **positive**.

- Chạy tập tin **objectmarker.exe** để lưu lại vị trí của đối tượng trong hình.



Hình VII‑3 Tạo danh sách Positive.

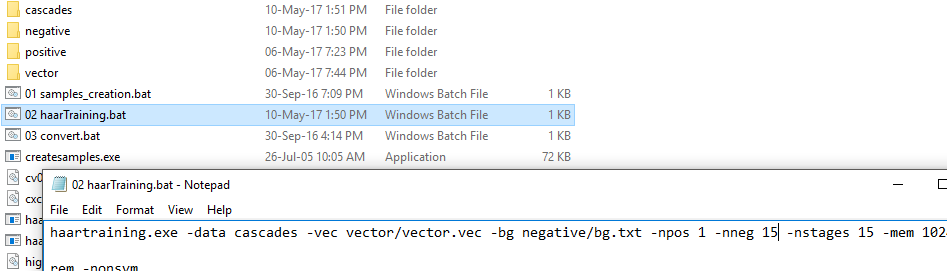
- Tập dữ liệu positive sẽ được lưu lại thành một danh sách trong tập tin **info.txt**.



Hình VII‑4 Danh sách Positive sau khi tạo xong.

* Bước 3: Chỉnh sửa thông số huấn luyện.

- Nhấp chuột phải vào tập tin **haarTraining.bat** chọn **edit** để chỉnh sửa thông số huấn luyện.



Hình VII‑5 Chỉnh sửa thông số huấn luyện.

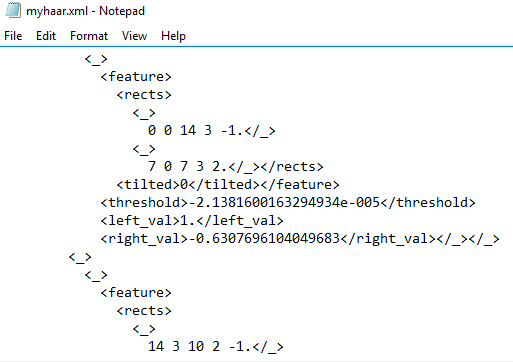
-npos <số lượng ảnh positive>.

-nneg <số lượng ảnh negative>.

* Bước 4: Huấn luyện.

- Chạy lần lượt 3 tập tin theo thứ tự **samples\_creation.bat**, **haarTraining.bat** và **convert.bat**.

- Kết quả cuối cùng sẽ tạo ra một tập tin .xml chứa thông tin dữ liệu hình ảnh đã được mã hóa.



Hình VII‑6 Thông số trong tập tin xml sau khi luyện.

### Phân lớp và khoanh vùng đối tượng

#### Đối tượng nhận dạng

Đối tượng nhận dạng sẽ là các cột đèn giao thông trên đường Nguyễn Hữu Thọ và đường Nguyễn Thị Thập, Quận 7, thành phố Hồ Chí Minh.

Sau khi khảo sát, nhóm đã xác định được hai đoạn đường trên có các loại đèn giao thông như sau:



Hình VII‑7 Đèn giao thông dạng 1.



Hình VII‑8 Đèn giao thông dạng 2.



Hình VII‑9 Đèn giao thông dạng 3.

#### Cài đặt

Để có một kết quả tốt thì tối thiểu phải luyện là 1000 dữ liệu hình ảnh (500 positive và 500 negative). Sau khi luyện xong, tập tin .xml sẽ dược gọi để sử dụng bằng lớp **CascadeClassifier**.

- Source code:

CascadeClassifier classifier;

classifier.load(CASCADE\_FILE\_NAME);

Sau khi tải tập tin .xml, cần chuyển hình ảnh (image) hoặc khung hình (frame) về mức xám và cân bằng Histogram (biểu đồ phân bố cường độ điểm ảnh có giá trị).

- Source code:

cvtColor(mFrame, mGray, COLOR\_BGR2GRAY);

equalizeHist(mGray, mGray);

Trong tập huấn luyện, các hình ảnh positive của đối tượng nhận diện có kích thước không tương đồng với nhau so với đối tượng trong mẫu thử (hình ảnh hoặc các khung hình trong video hay trực tiếp). Vì thế, việc nhận diện sẽ khó khăn nên openCV có hỗ trợ phương thức detectMultiScale (nhận diện với nhiều tỉ lệ). Ở trong ứng dụng nhóm quy định nhận diện đối tượng khi đối tượng có kích thước tối thiểu là 30x30 pixel.

- Source code:

classifier.detectMultiScale(mGray, vFound, 1.1, 2, 0 | CASCADE\_SCALE\_IMAGE, Size(30, 30));

#### Kiểm tra khoanh vùng

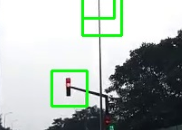
Khi chạy thực nghiệm, ứng dụng có thể nhận diện được các đèn giao thông nhưng vẫn còn nhiễu bởi các đối tượng có cấu trúc tương tự: cột đèn, cột điện,…



Hình VII‑10 Khoanh vùng đèn giao thông dạng 2.



Hình VII‑11 Khoanh vùng đèn giao thông dạng 3 (không nhiễu).



Hình VII‑12 Khoanh vùng đèn giao thông dạng 3 (có nhiễu).

### Nhận biết màu

#### Dải màu hsv của đèn giao thông

Sử dụng hàm **inRange** trong openCV để lấy các vùng màu cần thiết trong hình hoặc khung hình (lưu ý: phải chuyển sang dải màu HSV trước khi dùng hàm **inRange**). Các vùng màu cần lấy lần lượt là:

- Màu đỏ:

<1>: Scalar(0, 100, 100) -> Scalar(10, 255, 255)

<2>: Scalar(160, 100, 100) -> Scalar(179, 255, 255)

- Màu vàng:

<1>: Scalar(0, 100, 100) -> Scalar(10, 255, 255)

<2>: Scalar(5, 50, 50) -> Scalar(15, 255, 255)

- Màu xanh lá:

<1>: Scalar(30, 59, 50) -> Scalar(79, 158, 171)

<2>: Scalar(0, 255, 0) -> Scalar(229, 255, 204)

Mỗi màu đều có 2 vùng màu, sử dụng hàm **addWeighted** để hai vùng màu của mỗi màu thành một.

Sau khi thực hiện các bước trên, kết quả sẽ thu lại được 3 threshold của 3 màu đỏ, vàng và xanh lá.

#### Tìm vị trí màu trên ảnh

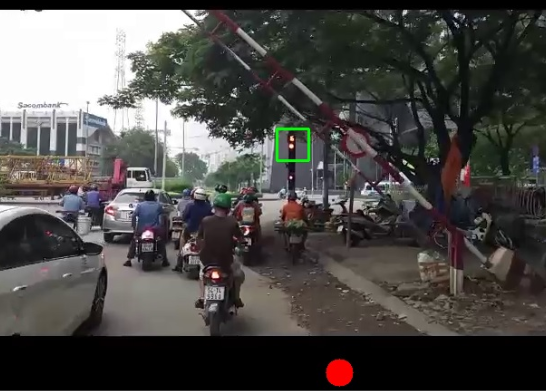
Dùng hàm **findContours** của openCV để nhận biết viền trong mỗi threshold của từng màu. Sau đó, sử dụng lớp đối tượng **Moments** của openCV để xác định vị trí và độ lớn của các cụm màu trong threshold.

Trước khi lưu lại các vị trí, cần phải khử nhiễu bằng cách kiểm tra kích thước của các cụm màu. Nếu cụm màu có kích thước nhỏ hơn 20x20 pixel thì đó là nhiễu cần loại bỏ. Nếu cụm màu lớn hơn hoặc bằng 2/3 màn hình hiển thị thì đó là do vật ở quá gần hoặc đó phông nền (background). Kết quả thu về dược một danh sách các vị trí màu đã được khử nhiễu.

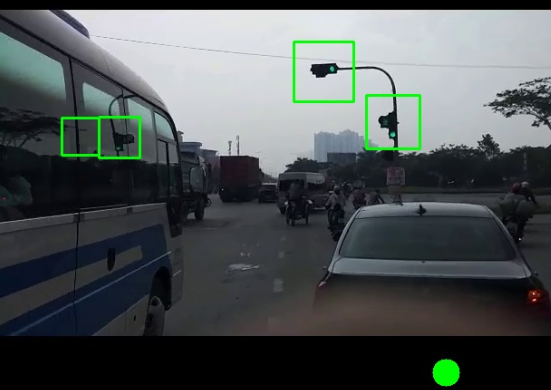
#### Kết hợp nhận diện màu với khoanh vùng

Sử dụng vòng lặp duyệt tuần tự, cứ mỗi khoanh vùng đối tượng thì kiểm tra bên trong có cụm màu nào không. Nếu có thì trả về màu nhận biết được (khi kiểm tra màu nên sử dụng lệnh **else if** để lấy duy nhất một màu đèn, tránh hiện tượng 1 lúc hiển thị 3 màu đèn).

Kết quả chạy thực nghiệm:



Hình VII‑13 Nhận diện tín hiệu đèn đỏ.



Hình VII‑14 Nhận diện tín hiệu đèn xanh.

## Kết luận

### Ưu điểm

* Nhận biết được hình dạng đèn giao thông và phân biệt được trạng thái của màu đèn.
* Khử nhiễu tốt.
* Có thể nhận dạng trên 3 phương diện: Hình ảnh, video, real time (thiết bị ghi hình).

### Ngược điểm

* Thuật toán khá nặng dẫn đến việc xử lý lâu khi áp dụng trên video và real time.
* Vẫn còn nhiễu tại một số điểm nhỏ dẫn đến sai sót đôi chút.
* Tập huấn luyện hình ảnh vẫn khá ít.

### Hướng phát triển

* Tối ưu và cải thiện thuật toán để quá trình xử lý tốt hơn.
* Tăng số lượng tập huấn luyện (khoảng trên 2000 hình positive, và trên 2000 hình negative) để giảm nhiễu và nhận diện chính xác hơn.
* Cập nhật thêm tính năng nhận biết thời gian chờ của đèn giao thông.
* Cập nhật thêm tính năng thông báo dành cho người đi bộ của các đèn giao thông hiện đại.

# TÀI LIỆU THAM KHẢO

1. M.Amutha Bharathi. “Color Detection in OpenCV”.
2. Jesse Levinson, Jake Askeland, Jennifer Dolson, Sebastian Thrun. “Trafﬁc Light Mapping, Localization, and State Detection for Autonomous Vehicles”.
3. Nguyễn Văn Long. “Ứng dụng xử lý ảnh trong thực thế với thư viện OpenCV C/C++”.
4. Sander Soo. “Object detection using Haar-cascade Classifier”.
5. N. Dalal and B. Triggs, “Histograms of oriented gradients for human dectection” in IEEE conference on computer vision and Pattern Recognition, 2005.
6. Moises Diaz-Cabrera and Pietro Cerri. “Traﬃc Light Recognition During the Night Based on Fuzzy Logic Clustering”.
7. Mahmoud Abdallah and Daniel Eiland. “Traffic Signal Detection and Tracking”, 2011.
8. Subhransu Maji et al. Classification using Intersection Kernel Support Vector Machines is Effcient. IEEE Computer Vision and Pattern Recognition 2008.