+ BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

TÊN ĐỀ TÀI

NHẬN DANG KHUÔN MẶT

GVHD: Thầy Lê Đình Duy

Thầy Phạm Nguyễn Trường An

SVTH : Nguyễn Trọng Ngô Việt Du

MSSV : 12520565

Lớp :CS114.K21

…., tháng… năm

I/ Giới thiệu

Nhận dạng khuôn mặt đã là một vấn đề được thực hiện trên khắp thế giới đối với nhiều người; vấn đề này đã xuất hiện trong nhiều lĩnh vực và khoa học, đặc biệt là khoa học máy tính, các lĩnh vực khác rất quan tâm Trong công nghệ này là: Cơ điện tử, Rô bốt, tội phạm học, v.v. đến thư viện xử lý ảnh Intel OpenCV và C # .Net, các thư viện này cho phép tôi chụp và xử lý ảnh của thiết bị chụp trong thời gian thực. Mục tiêu chính của bài viết này là trình bày và giải thích cách dễ nhất để thực hiện trình nhận diện và nhận dạng khuôn mặt trong thời gian thực cho nhiều người bằng cách sử dụng Phân tích thành phần chính (PCA) với eigenface để triển khai nó trong nhiều lĩnh vực.

II/ Lịch sử

Nhận dạng khuôn mặt là một ứng dụng máy tính tổng hợp cho các thuật toán phức tạp sử dụng các kỹ thuật toán học và ma trận, chúng nhận được hình ảnh ở chế độ raster (định dạng kỹ thuật số) và sau đó xử lý và so sánh pixel theo pixel bằng các phương pháp khác nhau để có kết quả nhanh hơn và đáng tin cậy phụ thuộc vào việc sử dụng máy để xử lý điều này do sức mạnh tính toán rất lớn mà các thuật toán, chức năng và thói quen này yêu cầu, đây là những kỹ thuật phổ biến nhất được sử dụng để giải quyết vấn đề hiện đại này:

III/ Kỹ thuật:

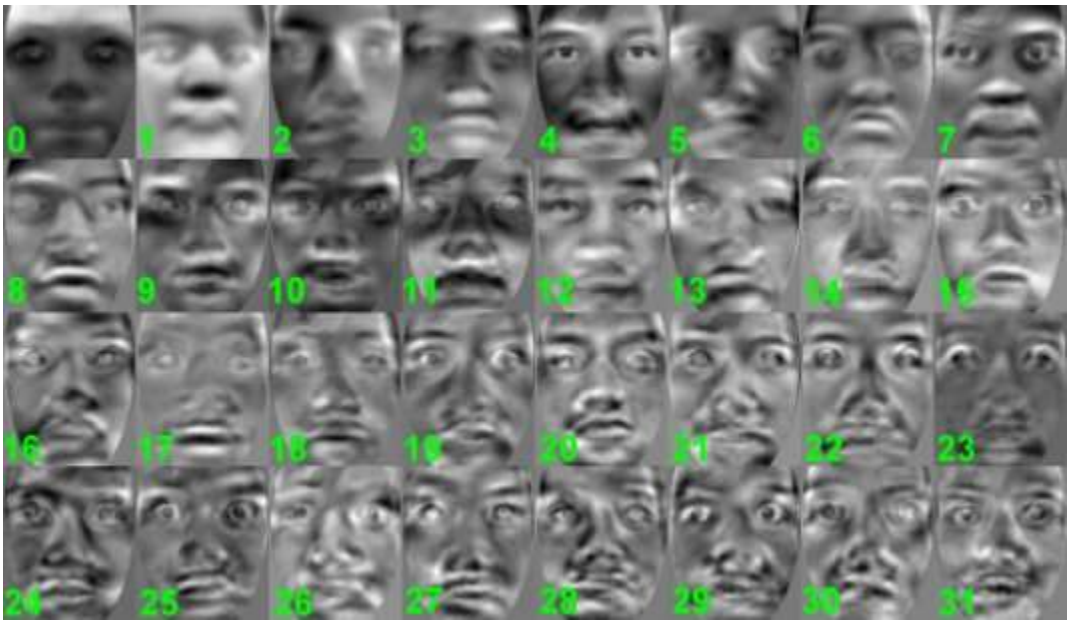
1/ Truyên thông

Một số thuật toán nhận dạng khuôn mặt xác định khuôn mặt bằng cách trích xuất các mốc hoặc tính năng từ hình ảnh khuôn mặt của đối

tượng. Ví dụ, một thuật toán có thể phân tích vị trí, kích thước và / hoặc hình dạng tương đối của mắt, mũi, xương gò má và hàm. Các tính năng này sau đó được sử dụng để tìm kiếm các hình ảnh khác có tính năng phù hợp. Các thuật toán khác bình thường hóa một bộ sưu tập hình ảnh khuôn mặt và sau đó nén dữ liệu khuôn mặt, chỉ lưu dữ liệu trong hình ảnh hữu ích để phát hiện khuôn mặt. Một hình ảnh thăm dò sau đó được so sánh với dữ liệu khuôn mặt. Một trong những hệ thống thành công sớm nhất dựa trên các kỹ thuật khớp mẫu được áp dụng cho một tập hợp các đặc điểm khuôn mặt nổi bật, cung cấp một kiểu biểu diễn khuôn mặt nén. Các thuật toán nhận dạng có thể được chia thành hai cách tiếp cận chính, hình học, xem xét các tính năng phân biệt hoặc trắc quang, đó là một cách tiếp cận thống kê để chắt lọc một hình ảnh thành các giá trị và so sánh các giá trị với các mẫu để loại bỏ phương sai. Các thuật toán nhận dạng phổ biến bao gồm Phân tích thành phần chính với eigenface, Phân tích phân biệt tuyến tính, Biểu đồ khớp khớp đàn hồi, mô hình Hidden Markov và kết hợp liên kết động được thúc đẩy bởi noron.

văn bản lấy từ **[1]**

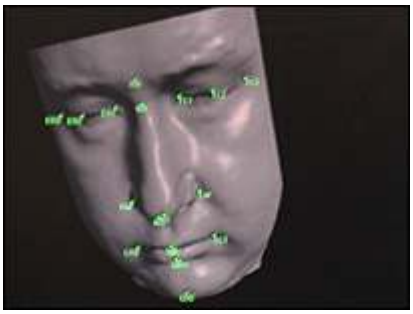
Một ví dụ về EigenFaces:



hình ảnh được lấy từ **[4]**

2/ 3-D

Một xu hướng mới nổi, được tuyên bố là đạt được độ chính xác chưa từng thấy trước đây, là nhận dạng khuôn mặt ba chiều. Kỹ thuật này sử dụng cảm biến 3 chiều để nắm bắt thông tin về hình dạng của khuôn mặt. Thông tin này sau đó được sử dụng để xác định các đặc điểm đặc biệt trên bề mặt của khuôn mặt, chẳng hạn như đường viền của hốc mắt, mũi và cằm. Một lợi thế của nhận dạng khuôn mặt 3 chiều là nó không bị ảnh hưởng bởi những thay đổi về ánh sáng như các kỹ thuật khác. Nó cũng có thể xác định một khuôn mặt từ một loạt các góc nhìn, bao gồm cả chế độ xem hồ sơ. Ngay cả một kỹ thuật kết hợp 3D hoàn hảo cũng có thể nhạy cảm với các biểu thức. Với mục tiêu đó, một nhóm tại Technion đã áp dụng các công cụ từ hình học số liệu để coi các biểu thức là hình học.



hình ảnh được lấy từ **[2]**

3/ Phân tích kết cấu da

Một xu hướng mới nổi khác sử dụng các chi tiết hình ảnh của da, như được chụp trong các hình ảnh kỹ thuật số hoặc quét tiêu chuẩn. Kỹ thuật này, được gọi là phân tích kết cấu da, biến các đường nét, hoa văn và đốm rõ ràng trên da của một người thành không gian toán học Các thử nghiệm đã chỉ ra rằng với việc bổ sung phân tích kết cấu da, hiệu suất trong việc nhận diện khuôn mặt có thể tăng 20 đến 25%. Nó thường được sử dụng trong các hệ thống bảo mật và có thể được so sánh với các sinh trắc học khác như hệ thống nhận dạng mống mắt hoặc mắt.

văn bản lấy từ **[1]**

IV/ EmguCV

Emgu CV là thư viện mã nguồn mở .Net đa nền tảng cho thư viện xử lý ảnh Intel OpenCV. Cho phép các hàm OpenCV được gọi từ các ngôn ngữ tương thích .NET như C #, VB, VC ++, IronPython, v.v ... Trình bao bọc có thể được biên dịch trong Mono và chạy trên Linux / Mac OS X.

văn bản lấy từ **[3]**

Nói theo cách riêng của tôi, EmguCV là một Wrapper tuyệt vời, điều này cho phép thực hiện những điều rất thú vị và nhiệm vụ của thị giác máy tính, bộ thư viện này cho phép thực hiện vô số dự án tuyệt vời trong lĩnh vực này, EmguCV có nhiều chức năng cho phép chúng tôi làm việc với CPU và GPU hiệu suất đáng kể với các đề cập mới nhất.

Dự án SW tuyệt vời này cho phép làm việc và làm:

- Nhận dạng ký tự quang học (OCR)
- Phát hiện khuôn mặt
- Phát hiện người đi bộ
- Dự án Kinect
- Tái tạo 3D
- Trình phát hiện tính năng SURF ... giữa nhiều nhiệm vụ thú vị khác.

1/ Khái niệm cơ bản về EmguCV

Nếu bạn chưa từng làm việc với trình bao bọc này, bạn muốn xem cách thêm tham chiếu vào dự án hoặc giải quyết rắc rối, hãy xem bài viết / hướng dẫn thần thánh này của **C_Johnson** :

Tạo dự án xử lý ảnh EMGU đầu tiên của bạn

Bài viết sau đây được thiết kế để hiển thị những người mới đến EMGUcv cách thiết lập dự án từng bước. Thông tin này được cung cấp miễn phí tại đây , tuy nhiên, bài viết này được thiết kế để làm cho quá

trình thân thiện hơn với người dùng. EMGU là trình bao bọc ac # cho OpenCV, nó khác với các trình bao bọc khác vì nó được viết hoàn toàn bằng c # và không sử dụng mã không an toàn. EMGU mở thư viện các chức năng lập trình OpenCV (Thư viện thị giác máy tính nguồn mở) chủ yếu nhắm vào tầm nhìn máy tính thời gian thực cho các nhà phát triển C #. OpenCV ban đầu được phát triển bởi Intel và hiện được Willow Garage hỗ trợ.

Các phiên bản hiện tại cho cả kiến trúc x86 và x64 đều có sẵn để tải xuống tại trang web Sourceforge của họ .

Vì EMGU là một trình bao bọc cho mã c ++, có hai loại *thư viện Liên kết động động* (DLL) được sử dụng. Có những cái EMGU với cú pháp EMGU luôn tham chiếu trong tên và những cái opencv khác nhau. Thiết lập một dự án đầu tiên là một trở ngại chung cho nhiều người mới và bạn không đơn độc.

Nếu bạn đã tải xuống mã nguồn, bạn sẽ cần phải đọc **chương trình cơ bản của Chương trình cơ bản** . Nếu bạn đã sao chép một ví dụ từ thư mục trích xuất EMGU, hãy xem phần **EMGU.CV.Invoke Ngoại lệ và Xử lý sự cố** .

1.1/ giả định

Giả định rằng người dùng có kinh nghiệm cơ bản về c # và có thể tạo dự án c # mới. Giả định rằng mỗi người dùng đã tải xuống bản cập nhật mới nhất cho nền tảng của họ và có Ví dụ HelloWorld chạy từ thư mục trích xuất EMGU \ EMGU.Examples \ Hello World.

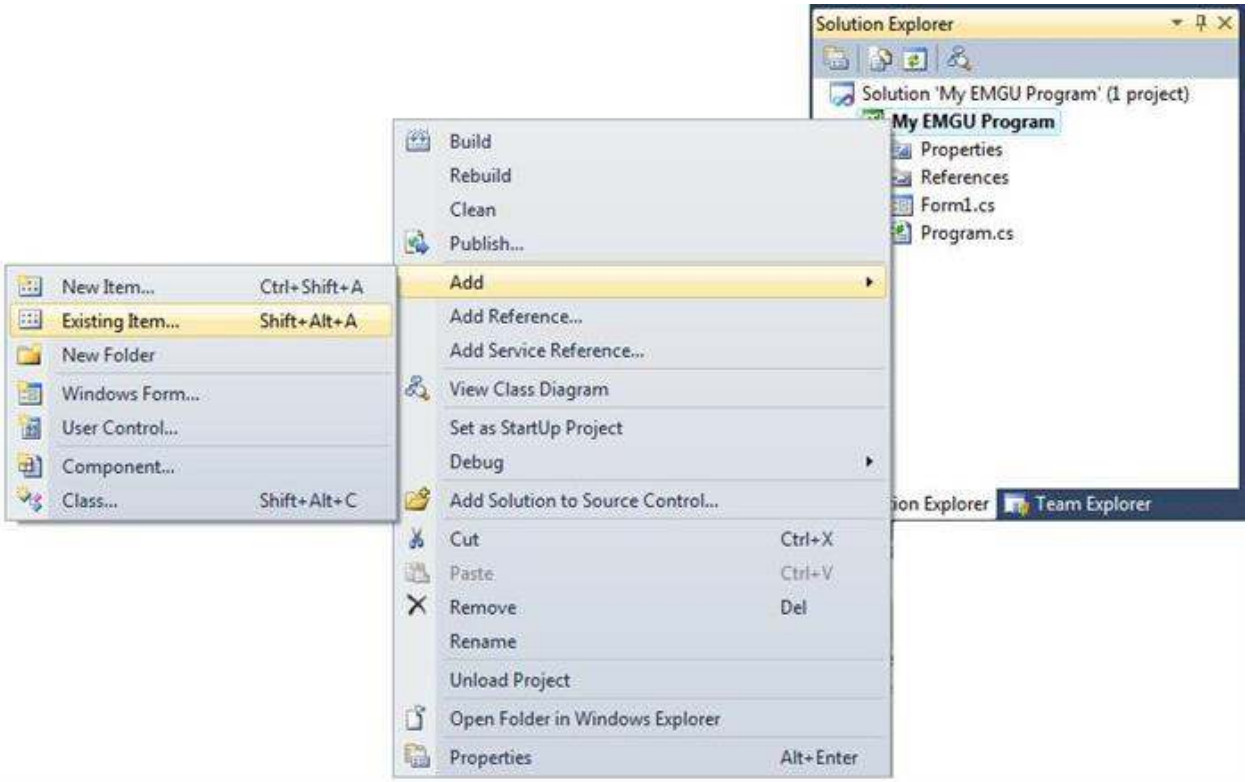
-Yêu cầu cơ bản

Như với bất kỳ thư viện c # nào, có một số DLL cần thiết cần tham khảo trong dự án của bạn. Bắt đầu một ứng dụng Windows Form c # mới và gọi nó là những gì bạn thích. Tất cả DLL được đề cập đều nằm trong

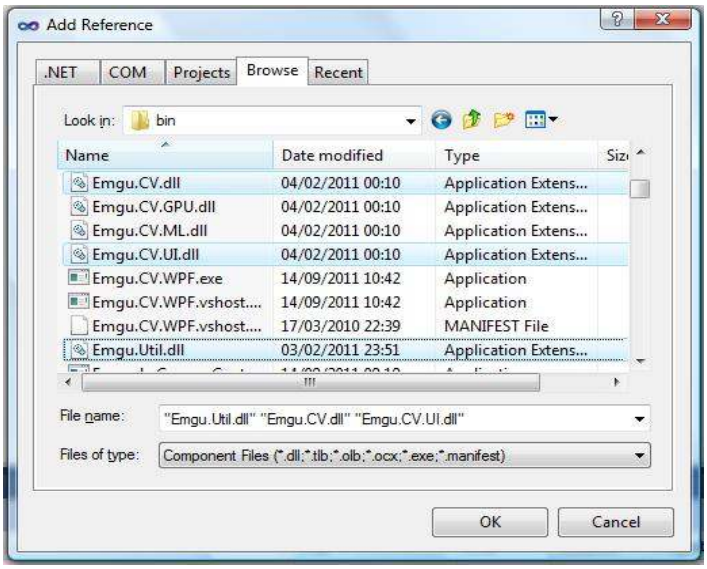
Thư mục trích xuất EMGU \ bin bạn sẽ cần phải nhớ điều này. Để bắt đầu, bạn cần tham khảo 3 EMGU DLL.

- Emgu.CV.dll
- Emgu.CV.UI.dll
- Emgu.Util.dll

Điều này được thực hiện bằng cách nhấp chuột phải vào tên dự án của bạn hoặc thư mục Tài liệu tham khảo trong trình khám phá giải pháp. Đi đến Thêm tài liệu tham khảo.



Hoặc thay thế bằng cách sử dụng mục menu Project> Thêm tài liệu tham khảo. Khi cửa sổ Thêm tham chiếu mở, chọn DLL được liệt kê ở trên và nhấp vào OK.



Bây giờ bạn sẽ thấy chúng được liệt kê trong thư mục Tài liệu tham khảo trong cửa sổ giải pháp thám hiểm. Ba DLL này là các thư viện c # cụ thể EMGU đã đề cập trước đây. Những điều này một mình sẽ không cho phép bạn sử dụng bất kỳ chức năng xử lý hình ảnh nào, vì vậy vui lòng đọc phần còn lại của bài viết.

Bây giờ bạn cần tham chiếu những thứ này trong bất kỳ lớp biểu mẫu nào mà bạn sẽ sử dụng mã. Các tài liệu tham khảo bạn sẽ sử dụng sẽ phụ thuộc vào những gì bạn đang làm trong điều khoản xử lý hình ảnh. Nhìn vào các ví dụ và chúng sẽ có những cái bạn yêu cầu. Để giúp bạn bắt đầu, hãy thêm phần sau vào đầu mã Form1.cs phía sau.

```
using Emgu.CV;
using Emgu.Util;
using Emgu.CV.Structure;
```

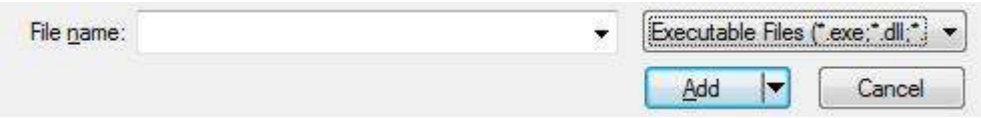
-Phương pháp ưa thích

Bây giờ đối với các thư viện c ++ phức tạp hơn, để tải, hiển thị, truy cập dữ liệu hình ảnh và thực hiện nhiều chức năng đơn giản hơn, bạn chỉ cần hai tệp. **Lưu ý** rằng "220" là số phiên bản sẽ thay đổi theo các bản cập nhật (opencv_core ***. Dll, opencv_imgproc ***. Dll).

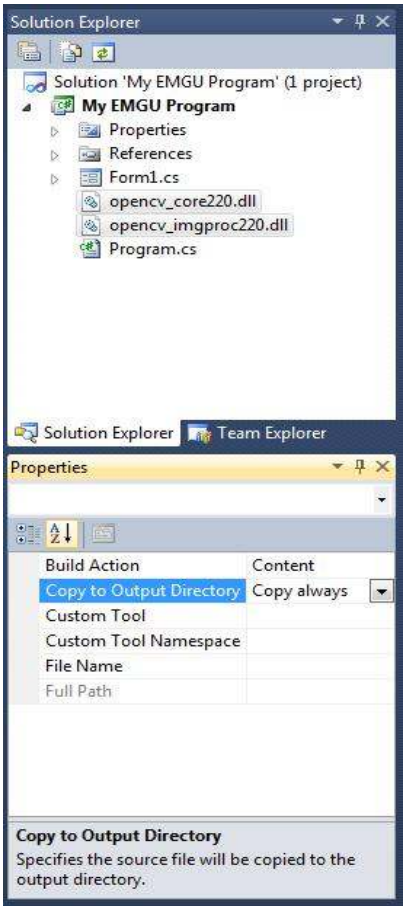
- opencv_core220.dll
- opencv_imgproc220.dll

Bây giờ vì đây là các tệp c ++ được gói, bạn không thể tham chiếu chúng như bạn đã làm với các tệp EMGU. Bạn cần thêm các tệp này vào dự án của bạn. Điều này có thể được thực hiện theo hai cách, Nhấp chuột phải vào tên dự án của bạn trong trình khám phá giải pháp (Đây không phải là cách bắt đầu với Giải pháp 'Tên dự án của bạn' ..., nhưng là một trong những điều này). Đi đến Thêm> Mục hiện có bằng hộp thoại mở tệp chuẩn. Chọn hai tệp ở trên, trong trường hợp bạn quên chúng nằm trong Thư mục trích xuất EMGU \ bin.

Gợi ý : nếu bạn không thể thấy. Hãy chắc chắn rằng bạn thay đổi bộ lọc tệp ở dưới cùng bên phải thành Tệp thực thi.



Bây giờ bạn sẽ có thể xem các tệp của mình trong cửa sổ giải pháp thám hiểm. Bạn sẽ cần phải thay đổi các thuộc tính để chọn cả hai bằng cách giữ phím Ctl và nhấp chuột trái vào chúng (thay vào đó bạn có thể thực hiện việc này riêng lẻ). Bây giờ hãy nhìn vào cửa sổ Properties, bạn sẽ thấy 6 trường hai trong số này sẽ chứa đầy nội dung. Bạn quan tâm đến Thư mục sao chép vào đầu ra. Thay đổi điều này từ bản sao Đừng sao chép bản sao sang bản sao luôn luôn bản sao.



Nếu bạn đang sử dụng các phần tổng hợp x64, hãy chuyển đến phần x64 và đảm bảo bạn thiết lập dự án của mình để biên dịch sang kiến trúc x64 ngoài việc bạn đã sẵn sàng bắt đầu xử lý hình ảnh. Lý do đây là phương pháp ưa thích là bây giờ, nếu bạn thay đổi từ Gỡ lỗi sang Phát hành, các tệp này sẽ luôn có sẵn cho chương trình của bạn và sẽ không có lỗi xảy ra. Chuyển đến phần đọc và hiển thị phần hình ảnh **Chương trình cơ bản** để bắt đầu.

-Phương pháp ít ưu tiên

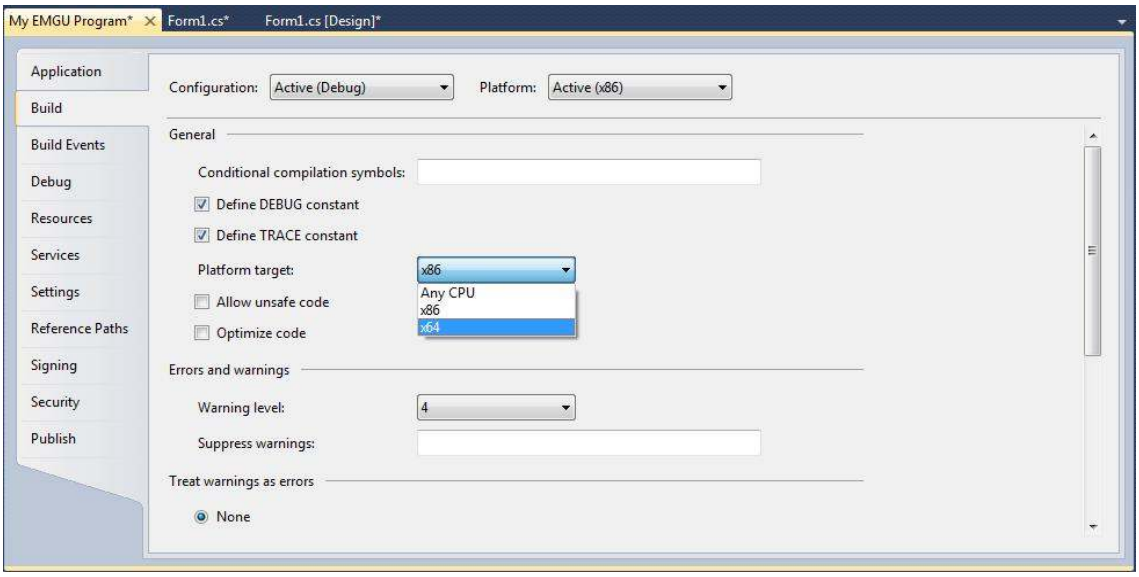
Mặc dù không được khuyến khích nhưng điều này thường đơn giản nhất và nếu bạn có một kiến trúc dự án phức tạp, điều này có thể ngăn các nhà thám hiểm giải pháp trông rất lộn xộn. Chỉ cần điều hướng trong windows explorer vào thư mục trích xuất EMGU \ bin sao chép các tệp

dll có liên quan **opencv_core220.dll** và **opencv_imgproc220.dll** vào thư mục bin \ Debug dự án của bạn hoặc vào thư mục bin \ Release. Điều này sẽ thay đổi với các phiên bản x64 vì nó sẽ là thư mục bin \ x64 \ Debug hoặc thư mục Phát hành thay thế.

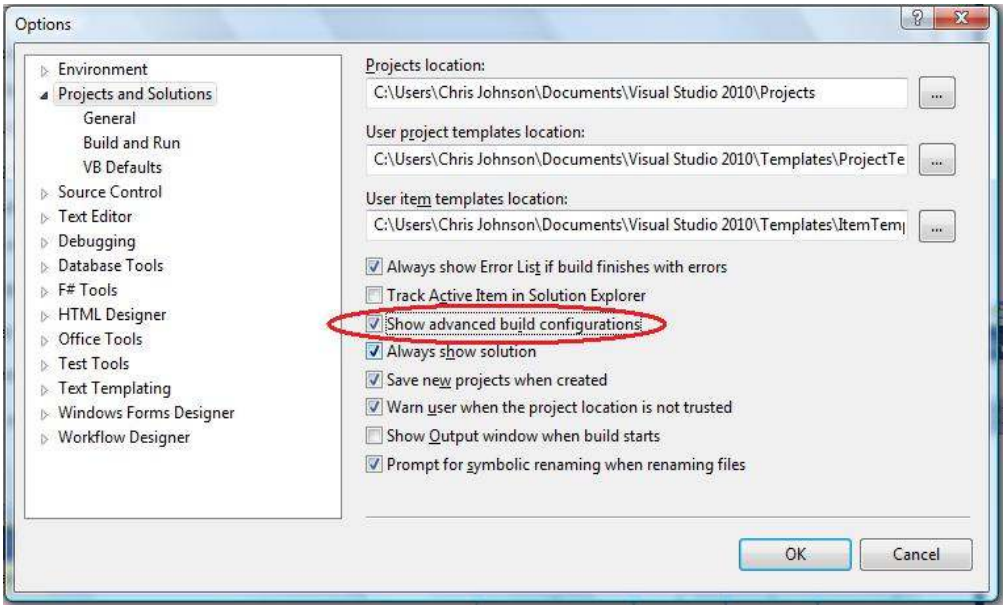
Mặc dù các lợi ích không rõ ràng ở đây, hãy tưởng tượng nếu bạn yêu cầu tất cả các tệp opencv DLL thì bạn sẽ có thêm 34 tệp trong trình khám phá giải pháp, tuy nhiên, rất hiếm khi xảy ra trường hợp này.

-Kiến trúc x64 và EMGU.CV.Invoke

Nếu bạn đang chạy một hệ thống x64 hoặc thiết kế cho chúng, bạn sẽ phải tải xuống các DLL riêng biệt. Các bước hình thành dự án là giống hệt nhau tuy nhiên bạn sẽ cần thay đổi một tham số xây dựng bổ sung. Nhấp chuột phải vào tệp dự án của bạn trong trình khám phá giải pháp và chọn Thuộc tính trực tuyến ở phía dưới. Chọn tab của Build Build 'từ thanh ruy băng ở bên phải cửa sổ này. Sẽ có một tùy chọn cho Nền tảng mục tiêu: với trình đơn thả xuống, thay đổi tùy chọn này từ x86 thành x64.



Gợi ý: Nếu bạn đang sử dụng phiên bản express của studio hình ảnh, bạn có thể không thấy tùy chọn x64 trong trường hợp như vậy, đi tới tùy chọn menu Công cụ> Tùy chọn. Trong cửa sổ này, sử dụng các mũi tên ở phía bên trái để mở rộng và thu gọn tùy chọn. Chọn các dự án và giải pháp khác và chọn hộp kiểm Hiển thị cấu hình bản dựng nâng cao.



Điều này bây giờ sẽ cho phép biên dịch chạy nếu điều này **không** được thực hiện chính xác. Ngay khi bạn truy cập vào bất kỳ mã EMGU nào, một ngoại lệ sẽ được ném ' EMGU.CV.Invoke' thông qua một ngoại lệ với ' InnerException' "Một nỗ lực đã được thực hiện để tải một chương trình có định dạng không chính xác ....

-CODE chương trình EMGU cơ bản

Để giúp bạn bắt đầu một chương trình đơn giản tải hình ảnh và hiển thị nó trong hộp hình ảnh đã được cung cấp và thêm một chút về chương trình nâng cao sẽ cho biết cách truy cập dữ liệu hình ảnh và chuyển đổi giữa các loại hình ảnh.

Chỉ có các phiên bản **x64** hiện có sẵn, x86 sẽ được cung cấp trong thời gian ngắn.

Nếu bạn đã tải xuống mã mẫu, bạn sẽ bắt đầu với 3 cảnh báo cho các tham chiếu không được tìm thấy. Mở rộng thư mục Tài liệu tham khảo trong trình khám phá giải pháp xóa 3 bằng các biểu tượng cảnh báo màu vàng và Thêm tham chiếu mới vào chúng, các bước có sẵn Phần Yêu cầu cơ bản.

Đã có một mục nút và một picturebox mục được thêm vào mẫu chính. Có tên mặc định không được thay đổi. Khi chúng tôi nhấp vào nút, chúng tôi muốn mở hộp thoại chọn tệp và hình ảnh và hiển thị nó trong picturebox.

```
private void button1_Click(object sender, EventArgs e)
{
   OpenFileDialog Openfile = new OpenFileDialog();
   if (Openfile.ShowDialog() == DialogResult.OK)
   {
      Image<Bgr, Byte> My_Image = new Image<Bgr,
byte>(Openfile.FileName);
      pictureBox1.Image = My_Image.ToBitmap();
   }
}
```

Mã này rất đơn giản, OpenFileDialog được gọi là 'Openfile' được sử dụng để chọn và tệp hình ảnh. Hình ảnh này sau đó được đọc vào

một Image đối tượng màu gọi là 'My_Image'. Hình ảnh được hiển thị bằng cách gán thuộc Image tính của picturebox. Điều này đòi hỏi một Bitmap và bằng cách gọi .ToBitmap()chức năng của 'My_Image', điều này đạt được.

Điều này cực kỳ đơn giản để đạt được một khi quy trình chính xác đã được thực hiện trong việc thiết lập dự án. Một thay thế cho mục tượng hình được cung cấp thông qua thư viện EMGU.CV.UI và được sử dụng trong các ví dụ.

-Phần xử lý ảnh

Dự án mã nguồn nâng cao hơn một chút sẽ có các cảnh báo tương tự như được mô tả trong Chương trình cơ bản đã được cung cấp. Các tài liệu tham khảo sẽ cần thay thế. Trong chương trình này có một cuộc biểu tình của chuyển đổi một Image từ colour để greyscale và truy cập vào Data các điểm ảnh riêng biệt. Mặc dù các phương pháp triệt tiêu dữ liệu phổ hình ảnh không hiệu quả nhất nhưng đây là một ví dụ điển hình về việc truy cập thuộc tính hình ảnh Data .

-Phần chuyển đổi hình ảnh

Chuyển đổi hình ảnh trong EMGU có thể phức tạp. Trong chương trình ví dụ, một Bgr hình ảnh màu được chuyển đổi Gray hoặc thang độ xám.

```
Image<gray,byte> gray_image = My_Image.Convert<gray,byte>();
```

Tuy nhiên, cuối cùng bạn sẽ muốn sử dụng một độ sâu (Tdepth) khác của hình ảnh chứ không phải Byte. Vấn đề với phương pháp này là bạn chỉ có thể chuyển đổi độ sâu hoặc màu một lần cho mỗi cuộc gọi. Hãy nói rằng chúng tôi muốn chuyển đổi từ Hình ảnh <bgr, byte> sang Hình ảnh <màu xám, gấp đôi> bạn sẽ phải sử dụng cú pháp sau.

```
Image<Gray,byte> gray_image = My_Image.Convert<Gray,byte>();
Image<Gray,double> gray_image =
My_Image.Convert<Gray,double>();
//or alternatively in one line
Image<Gray,> gray_image =
My_Image.Convert<Gray,byte>().Convert<Gray,double>();
//alternatively
Image<Gray,Byte> gray_image =
My_Image.Convert<Bgr,double>().Convert<Gray,double>();
```

-Dữ liệu hình ảnh

Có một vài cách để truy cập dữ liệu hình ảnh và gán giá trị cho nó. Có hai phương thức khả dụng, truy cập trực tiếp bằng thuộc tính Dữ liệu hình ảnh của truy cập từ xa hơn. Cả hai đều được chứng minh ở đây. Lưu ý điều quan trọng là phải tôn trọng độ sâu phổ ảnh khi truy cập thuộc tính Dữ liệu. Hình ảnh thang độ xám sẽ có độ sâu là một vì vậy sẽ được tham chiếu là [x, y, 0] tuy nhiên hình ảnh màu có độ sâu 3, [x, y, 0], [x, y, 1] & [x, y, 2] tương ứng với các phổ màu Xanh lam, Xanh lục & Đỏ (Bgr).

Hãy nói rằng chúng tôi muốn **gán** một giá trị cho một pixel ở vị trí [0,0] một giá trị. Sử dụng phương pháp từ xa dễ dàng hơn chúng ta có thể sử dụng:

```
//Colour Image
My_Image[0, 0] = new Bgr(Color.Red);

//Gray Image
gray_image[0, 0] = new Gray(200);
```

Hoặc chúng tôi sử dụng Data tài sản

```
//Colour Image
Color R = Color.Red;
My_Image.Data[0,0,2] = R.R; //Write to the Red Spectrum
My_Image.Data[0,0,1] = R.G; //Write to the Green Spectrum
My_Image.Data[0,0,0] = R.B; //Write to the Blue Spectrum

//Gray Image
gray_image[0, 0] = new Gray(200);
```

Vì vậy, viết lên pixel khá đơn giản nhưng việc **đọc** giá trị pixel thì sao.

```
//Colour Image
Bgr my_Bgr = My_Image[0, 0];

//Gray Image
Gray my_Gray = gray_image[0, 0];
```

Bây giờ, trong nhiều trường hợp, bạn sẽ không muốn làm việc với (Bgrhay Gray), vì vậy việc chuyển đổi chúng là rất quan trọng.

```
//BGR to Color
Color my_colour = Color.FromArgb((int)value.Red, (int)value.Blue,
(int)value.Green);

//Gray to int
int my_intensity = (int) my_Gray.Intensity;
```

Bạn sẽ nhận thấy rằng mỗi giá trị được chuyển thành một số nguyên để cho phép mất dữ liệu này là do cường độ được lưu trữ tự nhiên dưới dạng gấp đôi. Tuy nhiên, trong trường hợp này, phương thức dễ dàng hơn là truy cập thuộc tính Image Data . Nếu bạn muốn làm việc với dữ

liệu hình ảnh, không có yêu cầu phải liên tục chuyển đổi giữa Gray và số nguyên, v.v. Bạn có thể truy cập Data trực tiếp vào hình ảnh và sử dụng nó.

```
//Colour
Color my_colour = Color.FromArgb(My_Image.Data[0, 0, 0],
My_Image.Data[0, 0, 1], My_Image.Data[0, 0, 2]);

//Gray Image
int my_intensity = gray_image.Data[0, 0, 0];
```

Đơn giản hơn nhiều và dễ dàng hơn để làm việc khi xử lý hình ảnh Data trong một vòng lặp. Để kiểm tra cách thực hiện một vòng lặp, vui lòng tải xuống mã nguồn Xử lý hình ảnh nhỏ hơn.

V/ Sử dụng áp dụng vào bài

Đầu tiên khai báo tất cả các biến một đối tượng quan trọng để sử dụng:

-Khai báo tất cả các biến, vectơ và haarcascades

```
        Image<bgr,> currentFrame;
        Capture grabber;
        HaarCascade face;
        HaarCascade eye;
        MCvFont font = new
MCvFont(FONT.CV_FONT_HERSHEY_TRIPLEX, 0.5d, 0.5d);
        Image<gray,> result, TrainedFace = null;
        Image<gray,> gray = null;
        List<image<gray,>> trainingImages = new List<image<gray,>>();
        List<string> labels= new List<string>();
```

```
List<string> NamePersons = new List<string>();
int ContTrain, NumLabels, t;
string name, names = null;
```

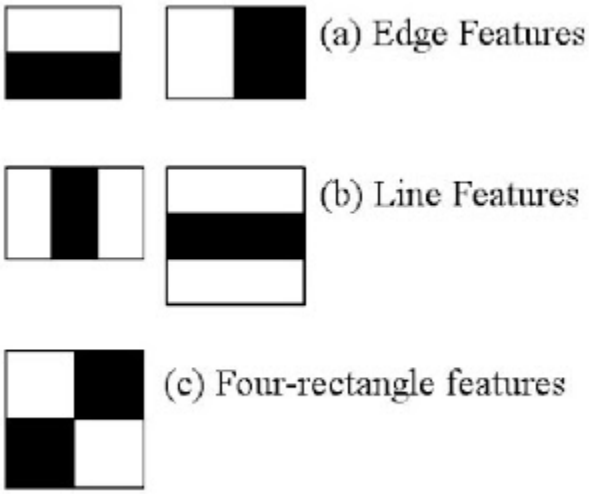
Sau đó tải các haarcascades để phát hiện khuôn mặt, sau đó tôi thực hiện một thủ tục nhỏ, để tải các khuôn mặt và nhãn được đào tạo trước đó cho mỗi hình ảnh được lưu trữ trước đó:

Haar Cascade là gì?

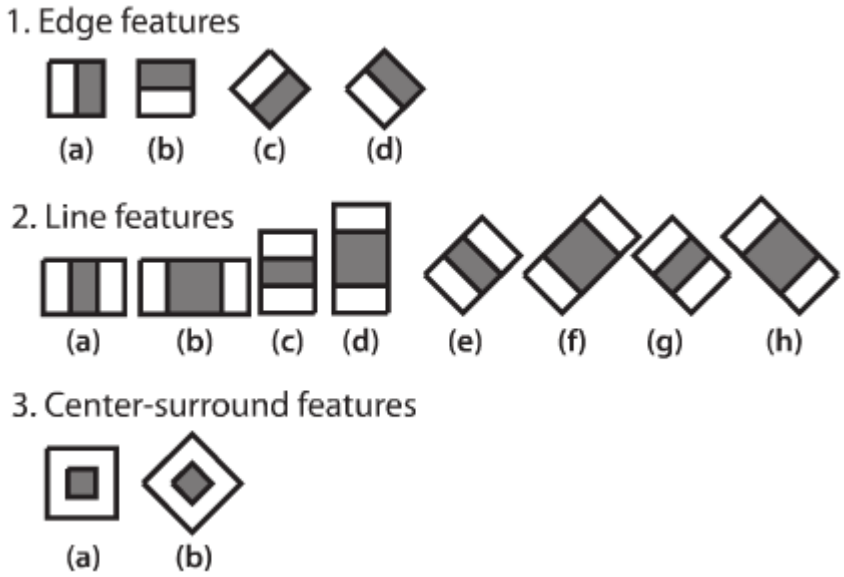
Về cơ bản là sử dụng các đặc trưng loại *Haar* và sau đó sử dụng thật nhiều đặc trưng đó qua nhiều lượt (*cascade*) để tạo thành một cỗ máy nhận diện hoàn chỉnh. Vẫn khó hiểu phải không? Vậy chúng ta nhảy vào từng khái niệm một nhé.

Đặc trưng Haar

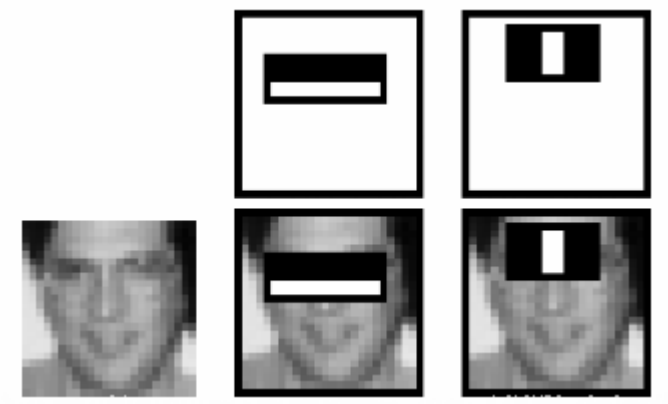
Nếu bạn đã làm việc với xử lý ảnh hoặc Convolutional Neural Networks rồi thì chắc bạn cũng không còn lạ gì với các bộ lọc trong xử lý ảnh nữa cả — nếu không, bạn có thể đọc phần này trong một bài của mình về CNN để hiểu thêm. Các ví dụ bộ lọc được liệt kê ở dưới, trong đó *a)* là các bộ lọc bắt các cạnh trong ảnh, và *b)* bắt các đường thẳng trong ảnh, tương tự như các bộ lọc đã được mình nhắc tới trong bài trên. Ngoài ra, còn có các bộ lọc Haar khác, như ví dụ *c)* về đặc trưng 4-hình vuông dưới đây,



hoặc đặc trưng nằm gọn trong trung tâm một vùng như ví dụ *3.* trong ảnh dưới đây:



Tuy nhiên, cách áp dụng các bộ lọc này khác một chút so với các cửa sổ bộ lọc bên CNN. Ở CNN, bộ lọc chiếm toàn bộ cửa sổ trượt, trong khi ở đặc trưng Haar, bộ lọc chỉ chiếm một phần trong cửa sổ trượt thôi. Điều đó được minh hoạ trên ảnh sau:

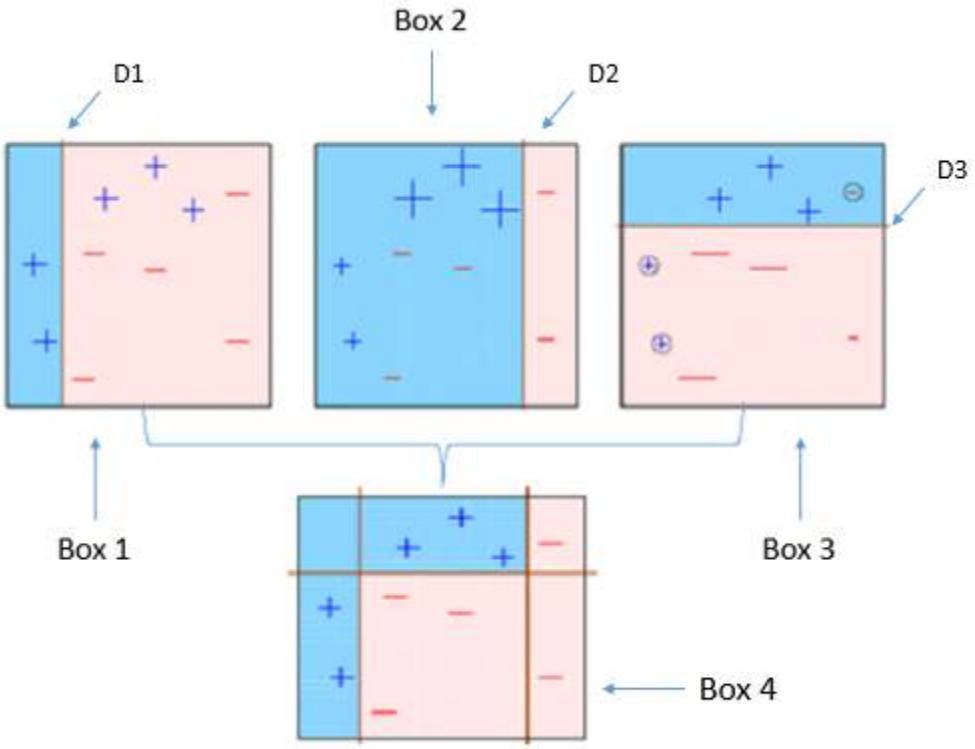


Trong hình trên, cửa sổ trượt được đặt ngay ngắn vừa gọn để nhìn được toàn bộ ảnh. Các bạn có thể nhận ra rằng bộ lọc đầu trong đó đang tìm một "cạnh" phân cách giữa mắt/lông mày với mũi, vì ở đoạn đọc có chênh lệch về màu đáng kể; và ở bộ lọc sau, mô hình đang tìm đường sống mũi, vì ở đó sẽ có màu sáng hơn so với 2 bên (vì nó cao hơn dễ bắt sáng). Và như đã nói trên, bộ lọc Haar chỉ nhìn cụ thể vào một vùng trong cửa sổ để tìm thôi: trong khuôn mặt thì mũi lúc nào cũng ở chính giữa chứ không ở các góc, nên không cần nhìn các góc để làm gì cả.

Để tăng tốc tính toán khi tính các bộ lọc trên, chúng ta thường sử dụng integral image. Mục này bạn có thể tham khảo thêm trong bài viết này của Hải Hà Chan để biết thêm chi tiết nhé.

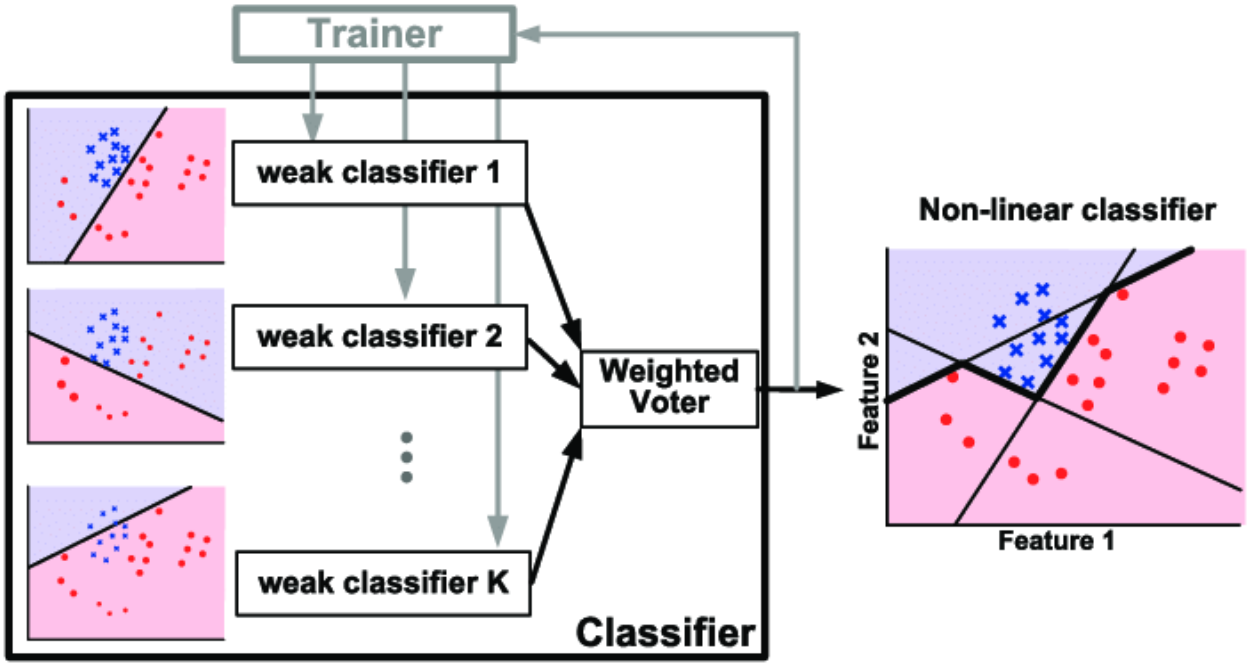
Làm sao mà biết được bộ lọc Haar nào tốt? Nhìn cái nào cũng như nhau...

Đúng vậy, chúng ta có tầm 160k+ các bộ lọc như vậy cơ! Tuy nhiên, chúng ta có thể sử dụng *Adaboost (adaptive boosting)* để kết hợp các bộ lọc trên như sau:



Ảnh trên là minh hoạ của quá trình boosting: khi bạn có các classifier yếu khác nhau, kết hợp chúng để tạo thành một classifier mạnh. Việc kết hợp như các bạn thấy khá đơn giản, chỉ là các khối logic AND/OR phụ thuộc vào kết quả được đưa ra. Câu trả lời là một lớp perceptron đơn giản! Một tổ hợp tuyến tính đơn giản đã có thể tính được kết hợp các đầu ra quyết định trên, bạn có thể tìm hiểu thêm ở mục ngoài lề này trong bài về linear classifier mình đã viết.

Vậy đó là boosting rồi, còn adaptive ở đây là gì? Ở boosting thường thì các classifier yếu trên có tiếng nói ngang nhau, bình đẳng và dân chủ; nhưng sau khi qua Adaboost, nhưng classifier khôn hơn thì tiếng nói sẽ có trọng lượng cao hơn. Sau đây là một hình ảnh minh hoạ khác để giúp bạn dễ tưởng tượng hơn chút:



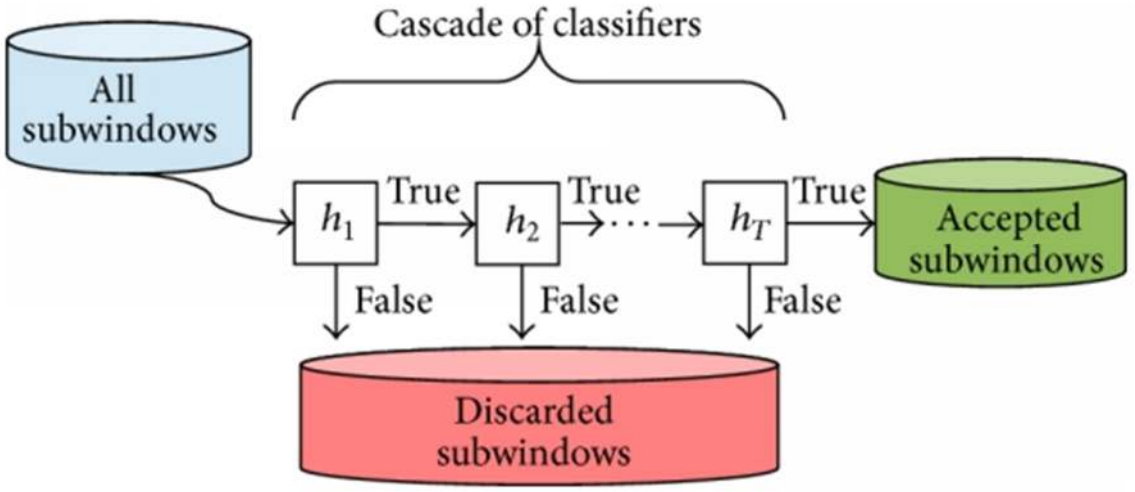
Kết quả là 2 bộ lọc ở mục trên là những bộ lọc tốt nhất theo như Adaboost.

Về thuật toán Adaboost, bạn có thể tham khảo thêm ở bài này của bạn Phạm Minh Phương.

Các đặc trưng trên bắt gì chả được đâu chỉ mỗi mặt?

Đúng như vậy, các bộ lọc Haar kể cả sau Adaboost như trên vẫn chỉ bắt được những đặc trưng rất cơ bản, và để nhận ra một khuôn mặt thì chúng ta cần tầm 6000 các đặc trưng như vậy! Vậy chúng ta cần có một cách để vote xem cửa sổ đó có chứa mặt không, mà vẫn phải xử lý đủ nhanh cho cả 6000 đặc trưng đó: câu trả lời là *Cascade of Classifiers*, được đề xuất bởi Paul Viola và Michael Jones vào năm 2001.

Việc *cascade* đó được thiết kế như sau: trong 6000+ đặc trưng đó, chia chúng ra thành rất nhiều bước. Trong đó, mỗi lần cửa sổ trượt qua một vùng bước ảnh, từng bước một sẽ được xử lý: nếu bước 1 nhận đó là mặt, chúng ta chuyển qua bước 2; và nếu không thì chúng ta bỏ qua vùng đó và trượt cửa sổ đi chỗ khác. Nếu một vùng pass toàn bộ các bước test mặt đó thì cửa sổ đó có chứa mặt người.



Các vùng không chứa mặt sẽ bị vứt vào hộp đỏ kia và không bao giờ được nhớ tới nữa, và các vùng có mặt sẽ được đưa vào hộp xanh để xử lý tiếp.

Chúng ta có thể chia kết quả ra làm 4 loại:

Tên	Đáp án đúng	Đáp án của chúng ta
True positive	True	True
True negative	False	False
False positive	False	True
False negative	True	False

Trong đó, nếu dự đoán đúng rồi thì không cần bàn cãi, nhưng nếu chúng ta dự đoán sai một bước sẽ có thể không qua các bước còn lại. Vì vậy, trong quá trình train, mô hình sẽ lựa chọn các classifier tốt nhất với độ tự tin hợp lý để ưu tiên việc *false negative* không được phép tồn tại — vì nếu chúng ta gặp *false positive*, các bước sau sẽ loại ví dụ đó cho chúng

ta sau. Trong mô hình nhận mặt người, bước đầu tiên đúng có 2 bộ lọc trên, với tỉ lệ false negative rất gần 0, và tỉ lệ false positive là 40%.

Sau đó các cửa sổ nằm trong ô xanh sẽ được trả lại làm các ô chứa mặt: nếu có các ô đè lên nhau cùng chứa một mặt, các toạ độ các góc sẽ được cộng vào lấy trung bình

```
Tải haarcascades để nhận diện khuôn mặt
face = new HaarCascade("haarcascade_frontalface_alt_tree.xml");
eye = new HaarCascade("haarcascade_eye.xml");
try
{
Tải các khuôn mặt và nhãn được huấn luyện trước cho mỗi hình ảnh
   string Labelsinfo = File.ReadAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt");
   string[] Labels = Labelsinfo.Split('%');
   NumLabels = Convert.ToInt16(Labels[0]);
   ContTrain = NumLabels;
   string LoadFaces;

   for (int tf = 1; tf < NumLabels+1; tf++)
   {
      LoadFaces = "face" + tf + ".bmp";
      trainingImages.Add(new Image<gray,>(Application.StartupPath +
"/TrainedFaces/" + LoadFaces));
      labels.Add(Labels[tf]);
   }
}
catch(Exception e)
{
   //MessageBox.Show(e.ToString());
```

```
    MessageBox.Show("Nothing in binary database, please add at least a face", "Triained faces load", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
}
```

Khởi tạo thiết bị chụp và sự kiện FrameGrabber thực hiện phát hiện và cung cấp hình ảnh cho mỗi khung hình được chụp:

```
grabber = new Capture();
grabber.QueryFrame();
Khởi tạo sự kiện FrameGraber
Application.Idle += new EventHandler(FrameGrabber);
button1.Enabled = false;
```

Chuyển đến sự kiện FrameGrabber (phần chính của nguyên mẫu), chúng tôi sử dụng các phương thức và đối tượng quan trọng nhất: DetectHaarCascade And EigenObjectRecognizer và thực hiện các thao tác Đối với mỗi khuôn mặt được phát hiện trong một khung hình:

```
MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
                face,
                1.2,
                10,

Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
                new Size(20, 20));
Hành động cho từng phần tử được phát hiện
                 foreach (MCvAvgComp f in facesDetected[0])
                 {
                    t = t + 1;
                    result = currentFrame.Copy(f.rect).Convert<gray,>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
```

```
vẽ mặt được phát hiện trong ô 0 (màu xám) có màu xanh lam

                    currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);


                    if (trainingImages.ToArray().Length != 0)
                    {
TermCriteria để nhận dạng khuôn mặt với số lượng hình ảnh được đào tạo như maxIteration
                    MCvTermCriteria termCrit = new
MCvTermCriteria(ContTrain, 0.001);

Nhận dạng khuôn mặt Eigen
                    EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(
                      trainingImages.ToArray(),
                      labels.ToArray(),
                      5000,
                      ref termCrit);

                    name = recognizer.Recognize(result);


Gắn nhãn cho mỗi khuôn mặt được phát hiện và nhận dạng
                    currentFrame.Draw(name, ref font, new Point(f.rect.X -
2, f.rect.Y - 2), new Bgr(Color.LightGreen));

                    }
}
```

Thông số:

haarObj: Phân loại tầng Haar trong thang đo đại diện bên trongFactor: Yếu tố mà cửa sổ tìm kiếm được chia tỷ lệ giữa các lần quét tiếp theo, ví dụ: 1.1 có nghĩa là tăng 10% cửa sổ

minNeighbor: Số lượng tối thiểu (trừ 1) hình chữ nhật lân cận tạo nên một đối tượng. Tất cả các nhóm có số lượng hình chữ nhật nhỏ hơn min_neighbor-1 đều bị từ chối. Nếu min_neighbor bằng 0, hàm không có bất kỳ nhóm nào và trả về tất cả các hình chữ nhật ứng cử viên được phát hiện, có thể hữu ích nếu người dùng muốn áp dụng quy trình nhóm tùy chỉnh

cờ: Phương thức hoạt động. Hiện tại cờ duy nhất có thể được chỉ định là CV_HAAR_DO_CANNY_PRUNING. Nếu được đặt, hàm sẽ sử dụng trình phát hiện cạnh Canny để từ chối một số vùng hình ảnh chứa quá ít hoặc quá nhiều cạnh và do đó không thể chứa đối tượng được tìm kiếm. Các giá trị ngưỡng cụ thể được điều chỉnh để phát hiện khuôn mặt và trong trường hợp này, việc cắt tỉa sẽ tăng tốc độ xử lý.

minSize: Kích thước cửa sổ tối thiểu. Theo mặc định, nó được đặt thành kích thước của các mẫu mà trình phân loại đã được đào tạo trên (~ 20x20 để nhận diện khuôn mặt)

-Làm thế nào để đào tạo nguyên mẫu?

Tôi thực hiện phần này một cách dễ dàng nhất có thể, nguyên mẫu phát hiện khuôn mặt liên tục (Mỗi khung hình) và bạn có thể thêm khuôn mặt được phát hiện này vào cơ sở dữ liệu hình ảnh với một nhãn tương ứng, hình ảnh được đào tạo sẽ hiển thị trong imageBoxFrameGrabber và quá trình sẽ kết thúc! !

Hãy ghi nhớ: Các thuật toán nhận dạng khuôn mặt dựa trên PCA (Phân tích thành phần chính) thực hiện nhiều so sánh và đối sánh giữa một khuôn mặt được phát hiện và hình ảnh được đào tạo được lưu trữ trong cơ sở dữ liệu nhị phân vì lý do này và để cải thiện độ chính xác của nhận dạng, bạn nên thêm một số hình ảnh của Cùng một người ở các góc độ,

vị trí và điều kiện độ chói khác nhau, khóa đào tạo này làm nguyên mẫu này vững chắc và rất chính xác.

Mã của nút đào tạo (Điều này thực hiện việc thêm các mặt và nhãn đào tạo cho mỗi):

```
try
        {
Trained face counter
            ContTrain = ContTrain + 1;

Lấy khung màu xám từ thiết bị chụp
            gray = grabber.QueryGrayFrame().Resize(320, 240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

Nhận diện khuôn mặt
            MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
            face,
            1.2,
            10,

Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUN
ING,
            new Size(20, 20));


Hành động cho từng phần tử được phát hiện
            foreach (MCvAvgComp f in facesDetected[0])
            {
               TrainedFace = currentFrame.Copy(f.rect).Convert<gray,>();
               break;
            }

Thay đổi kích thước khuôn mặt được phát hiện cho lực lượng để so sánh
cùng kích thước với
```

```
Kiểm tra hình ảnh với phương pháp kiểu nội suy bậc ba
TrainedFace = result.Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
            trainingImages.Add(TrainedFace);
            labels.Add(textBox1.Text);

Hiển thị khuôn mặt được thêm vào trong thang màu xám
            imageBox1.Image = TrainedFace;

Viết số lượng các mặt được xử lý trong một văn bản tệp để tải thêm
            File.WriteAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt",
trainingImages.ToArray().Length.ToString() + "%");

Viết nhãn của các mặt được xử lý trong văn bản tệp để tải thêm
for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
            {
                trainingImages.ToArray()[i -
1].Save(Application.StartupPath + "/TrainedFaces/face" + i + ".bmp");
                File.AppendAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");
            }

            MessageBox.Show(textBox1.Text + "´s face detected and
added :)", "Training OK", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
        catch
        {
            MessageBox.Show("Enable the face detection first", "Training
Fail", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);
        }
    }
```

VI/ cải thiện thuật toán

1/ Cải thiện độ chính xác

Các tham số mặc định (scale_factor = 1.1, min_neighbor = 3, flags = 0) được điều chỉnh để phát hiện đối tượng chính xác nhưng chậm.

Ngoài ra, bạn có thể sửa đổi kích thước cho một giá trị lớn, Sửa đổi điều này trong mã:

```
MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
face,
1.1,
3,
0,
new Size(20, 20));
```

Sửa đổi thông số thứ 3 thành **2500** hoặc **3000** thay vì 5000, sửa đổi này làm cho EigenObjectRecognizer nghiêm ngặt hơn / hiệu quả hơn.

```
Nhận dạng khuôn mặt Eigen
                    EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(
                      trainingImages.ToArray(),
                      labels.ToArray(),
                      5000,
                      ref termCrit);
```

2/ Cải thiện hiệu xuất CPU?

Tất cả các thuật toán xử lý hình ảnh đòi hỏi nhiều sức mạnh tính toán, trong trường hợp này, quy trình bên trong được thực hiện cho CPU với nguyên mẫu sw này rất khó để CPUS monocore chậm hơn, cách dễ dàng để cải thiện hiệu suất của Demo này là sửa đổi các tham số sử dụng Phương pháp DetectHaarCascade, những phương thức này cho phép

giảm số lần lặp, phần phê bình và So sánh hình ảnh thời gian thực được chụp cho webcam cải thiện đáng kể hiệu năng của ứng dụng.

Hãy ghi nhớ: giảm các giá trị của các tham số này sẽ ảnh hưởng đến hiệu quả của Thuật toán nhận dạng.

3/ Lựa chọn đầu tiên:

Để thao tác nhanh hơn trên hình ảnh video thực, các cài đặt là: scale_factor = 1.2, min_neighbor = 2, flags = CV_HAAR_DO_CANNY_PRUNING, min_size = <tối thiểu> (ví dụ: ~ 1/4 đến 1/16 của khu vực hình ảnh trong trường hợp hội nghị video).

Ngoài ra, bạn có thể sửa đổi tham số Minsize cho một giá trị lớn.

Ẩn mã sao chép

```
// DetectHaarCascade Config for optimal performance

MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
                face,
                1.2,
                2,

Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
                new Size(20, 20));
```

4/ Sự lựa chọn thứ hai:

Nhận một hình thu nhỏ của người Hồi giáo hoặc thay đổi kích thước bản chụp ảnh gốc để giảm thời gian xử lý Trong phương thức FrameGrabber sửa đổi các giá trị kích thước cho kích thước nhỏ (ban đầu là 320x240)

Thí dụ:

Ẩn mã sao chép

```
Lấy thiết bị chụp mẫu hiện tại
currentFrame = grabber.QueryFrame().Resize(260, 200,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
```

Nhớ làm tương tự trong nút Đào tạo:

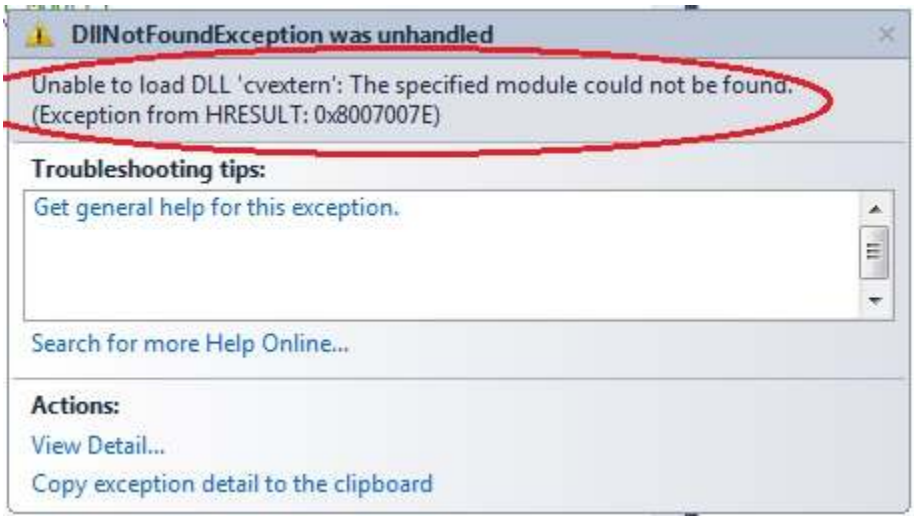
```
gray = grabber.QueryGrayFrame().Resize(260, 200,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
```

Nguyên mẫu không hiển thị những người lạ là không rõ, Tại sao?

Là một hạn chế của phương pháp Eigenface, không phải là lỗi, thay vào đó hãy sử dụng thuật toán LBPH (có những ưu và nhược điểm) có sẵn trên EmguCV 2.4.2 trở lên.

VII/ Một số lỗi mắc phải

Để tránh những lỗi như thế này:



1. Lần đầu tiên tải xuống và giải nén FaceRecPro_Demo.zip, nó có các thư viện OS (DLL) bên ngoài được sử dụng cho một số chức năng OpenCV, sau khi sao chép các DLL này trong C: / Windows / System32 hoặc trong thư mục bin của Dự án này

2. Sau đó tải xuống EmguCV (Wrapper của OpenCV cho C #) tại đây: http://sourceforge.net/projects/emgucv/files/ , cài đặt nó và sau đó đi đến C: \ Emgu \ emgucv-windows-x86 2.2.1.1150 \ bin thư mục và bản sao: opencv_calib3

hãy nhớ DLL này là Thư viện OpenCV và cần thiết để chạy bất kỳ dự án nào sử dụng EgmuCV

đây là cách dễ nhất để chạy bản demo hoặc dự án, được tìm thấy và Sao chép các tệp này vào thư mục bản demo hoặc thư mục bin (nếu bạn muốn "chạy" mã nguồn):

opencv_calib3d220.dll, opencv_of dll (DLL từ Nội dung OpenCV trong Tải xuống EmguCV) và: Emgu.CV.dll Emgu.CV.UI.dll Emgu.Util.dll cv110.dll cvaux110.dll cvextern.dll cxcore110.dll highgui110.dll (Nội dung vào Demo Tải xuống zip)

Hãy ghi nhớ: bạn có thể sao chép các tệp này trong thư mục Windows / System32 / và quên các vấn đề phụ thuộc của bạn cho dự án này và dự án khác sử dụng Emgu hoặc openCV (Emgu.CV.dll, Emgu.CV.UI.dll, Emgu.Util. dll NÊN luôn đi trong thư mục bin hoặc .exe) **hoặc Tải xuống tất cả dự án đã sẵn sàng (Phiên bản được tối ưu hóa) và các tệp tại đây:**

3/ Điểm quan tâm

Tôi gặp nhiều vấn đề với các vectơ sử dụng với EmguCV, vì lý do này tôi đã học cách sử dụng danh sách các thành phần như vectơ và nó thực sự hoạt động cho dự án của tôi.

Tôi học được nhiều cách xử lý hình ảnh, PCA và EigenFaces và để tối ưu hóa mã, do nhu cầu tài nguyên rất lớn cho một phần của thuật toán thị giác nhân tạo.

Ý tưởng dự án này xuất hiện sau khi thấy một người đàn ông Iron cảnh XD XD

VIII/ Tài liệu tham khảo

Cuốn sách tham khảo chính thức cho OpenCV là: "Học OpenCV: Tầm nhìn máy tính với Thư viện OpenCV", từ O'Reilly (2008)

Có hàng trăm cuốn sách hay khác về thị giác máy tính không dành riêng cho OpenCV:

- "Tầm nhìn máy tính: Cách tiếp cận hiện đại" của Forsyth và Ponce (2002).
- "Tầm nhìn máy tính: Thuật toán và ứng dụng" của Szeliski (2011).
- "Xử lý ảnh kỹ thuật số" của Gonzalez và Woods (2001).
- "Hướng dẫn cần thiết để xử lý ảnh" của Bovik (2009).
- "Tầm nhìn và ứng dụng máy tính: Hướng dẫn cho sinh viên và học viên", của Jähne và Haußecker (2000).
- Bảng tính xử lý ảnh HIPR2 (giải thích đơn giản về nhiều chủ đề thị giác máy tính)

[1] http://en.wikipedia.org/wiki/Facial_recognition_system

[2] "Các công ty hướng đến tương lai sinh trắc học" của Dominic Bailey, BBC News, tại Biometrics 2006

[3] http://www.emgu.com/wiki/index.php/Main_Page

[4] http://www.shervinemami.info/faceRecognition.html