

TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT THÀNH PHỐ HỒ CHÍ MINH

KHOA ĐÀO TẠO CHẤT LƯỢNG CAO



ĐỒ ÁN 3

Tìm hiểu thư viện Accord.Net và viết ứng dụng minh họa

SVTH : NGUYỄN VÕ HOÀNG

MSSV : 17110143

GVHD: HUỲNH XUÂN PHỤNG

Tp. Hồ Chí Minh, ngày 19 tháng 12 năm 2020

Mục lục

CHƯƠNG I: CƠ SỞ LÝ THUYẾT.....	3
CHƯƠNG II: SƠ BỘ VỀ THUẬT TOÁN VIOLAS – JONES	6
CHƯƠNG III: Demo.....	11

CHƯƠNG I: CƠ SỞ LÝ THUYẾT

1. Tổng quan về thư viện Accord.NET

- Accord.NET là một framework được sử dụng cho việc tính toán khoa học (scientific computing) trong .NET. Framework Accord.NET gồm nhiều thư viện khác cùng với đó là một loạt các ứng dụng vào trong tính toán khoa học, chẳng hạn như xử lý dữ liệu thống kê (statistical data processing), machine learning, nhận dạng mẫu (pattern recognition), phép tính bao gồm nhưng không giới hạn (including but not limited to), thị giác máy tính (computer vision) và thuật toán xử lý âm thanh (computer audition). Ngoài ra Accord.NET còn cung cấp một lượng lớn thuật toán phân phối xác suất, kiểm định giả thuyết thống kê, hàm kernel và hỗ trợ cho hầu hết các kỹ thuật đo lường hiệu suất phổ biến.
- Accord.NET Framework còn là một framework học máy .NET kết hợp với các thư viện xử lý âm thanh và hình ảnh hoàn toàn được viết bằng C#. Nó là một khung hoàn chỉnh để xây dựng cấp độ sản xuất thị giác máy tính, thử nghiệm máy tính, xử lý tín hiệu và các ứng dụng thống kê ngay cả khi sử dụng cho mục đích thương mại. Một bộ ứng dụng mẫu toàn diện giúp khởi động nhanh và chạy nhanh, đồng thời tài liệu và wiki phong phú giúp điền thông tin chi tiết.
- Nhưng ứng dụng phổ biến trong Accord.NET gồm có:
 - Classification
 - Regression
 - Clustering
 - Distributions
 - Hypothesis Tests
 - Kernel Methods
 - Imaging

- Audio and Signal
- Vision

2. Các thành phần trong Accord.NET

- Các thư viện thường được sử dụng cho Scientific Computing:
 - **Accord.Math:** Chứa một thư viện mở rộng dành cho ma trận, cùng với một bộ các phương pháp phân rã ma trận số, các thuật toán tối ưu hóa số cho các bài toán bị ràng buộc và không bị ràng buộc, các chức năng đặc biệt và các công cụ khác cho các ứng dụng khoa học.
 - **Accord.Statistics:** Phân phối xác suất, các mô hình và phương pháp thống kê như hồi quy tuyến tính và hồi quy logistic, Mô hình Markov ẩn, Trường ngẫu nhiên có điều kiện (Ẩn), Phân tích thành phần chính, Bình phương ít nhất một phần, Phân tích phân biệt, Phương pháp và Kernel methods và nhiều kỹ thuật liên quan khác.
 - **Accord.MachineLearning:** Hỗ trợ Vector Machines, Cây quyết định(Decision tree), các mô hình Naive Bayesian, K-means, Gaussian Mixture và các thuật toán chung như RANSAC, Cross-validation và Grid-Search cho các ứng dụng học máy.
 - **Accord.Neuro:** Các thuật toán học tập thần kinh như Levenberg-Marquardt (LM), Sự lan truyền ngược có khả năng phục hồi song song, Deep learning, Máy tính bị giới hạn Boltzmann, các thủ tục khởi tạo như Nguyen-Widrow và các phương pháp liên quan đến mạng nơ-ron khác.
- Các thư viện thường được sử dụng cho Signal and Image Processing:
 - **Accord.Imaging :** Công cụ dò tìm điểm ưa thích (Harris, SURF và FAST), phương pháp tìm ảnh giống nhau và ghép ảnh, tạo hình ảnh tích hợp và các phép biến đổi hình ảnh khác, cộng với các bộ lọc hình ảnh bổ sung để xử lý hình ảnh trong các ứng dụng.
 - **Accord.Audio:** Xử lý, chuyển đổi, lọc và xử lý tín hiệu âm thanh cho các ứng dụng thống kê và học máy.

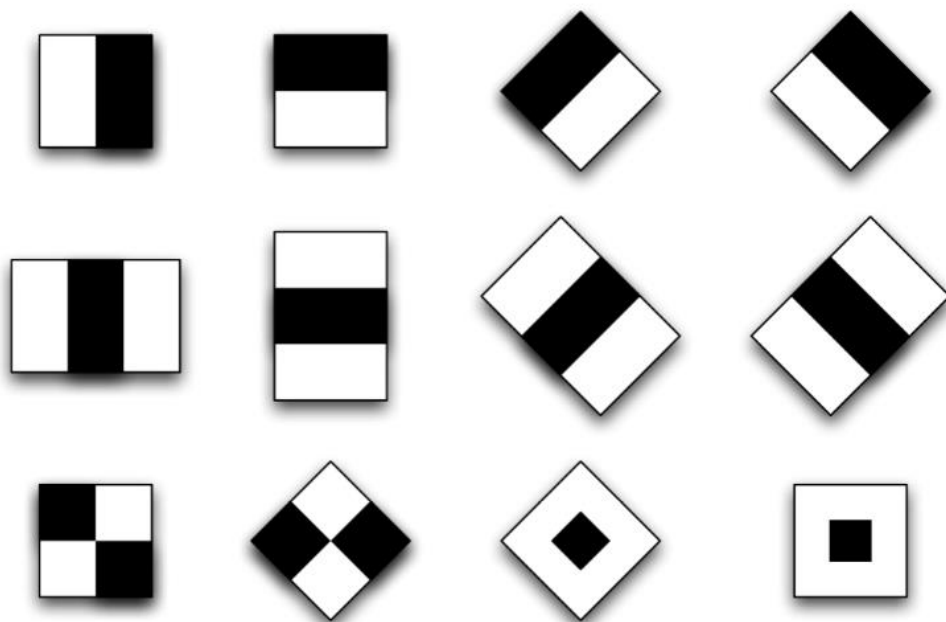
- **Accord.Vision:** Theo dõi và phát hiện khuôn mặt thời gian thực, cũng như các phương pháp chung để phát hiện, theo dõi và chuyển đổi các đối tượng trong luồng hình ảnh. Chứa các định nghĩa phân tầng Camshift và Trình theo dõi động theo mẫu.
- Một số thư viện hỗ trợ khác:
 - **Accord.Controls:** Gồm có biểu đồ tần suất(Histograms), biểu đồ điểm (scatter-plots), dữ liệu dạng bảng cho ứng dụng khoa học.
 - **Accord.Controls.Imaging:** Các điều khiển Windows Forms để hiển thị và xử lý hình ảnh. Chứa một trình điều khiển ImageBox thuận tiện bắt chước hành vi MessageBox truyền thống để nhanh chóng hiển thị hoặc kiểm tra hình ảnh.
 - **Accord.Controls.Audio:** Windows Forms điều khiển để hiển thị dạng sóng và thông tin liên quan đến âm thanh.
 - **Accord.Controls.Vision:** Các thành phần và điều khiển của Windows Forms để theo dõi các chuyển động của đầu, mặt và tay cũng như các tác vụ khác liên quan đến thị giác máy tính.

3. Phương thức cài đặt Accord.NET

- Framework Accord.NET được phân chia trong các thư viện, có sẵn thông qua trình cài đặt thực thi, các kho lưu trữ nén độc lập và các gói NuGet.

CHƯƠNG II: SƠ BỘ VỀ THUẬT TOÁN VIOLAS – JONES

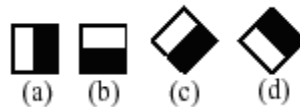
- Thuật toán viola jones sử dụng cửa sổ 24x24 để đánh giá các đặc trưng của ảnh. Nếu xem xét tất cả các tham số của các đặc trưng, ta tính được khoảng 160.000+ đặc trưng cho mỗi cửa sổ.
- Các đặc trưng Haar-Like là những hình chữ nhật được phân thành các vùng khác nhau như hình:



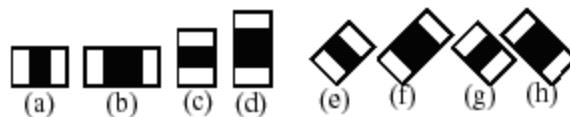
- Đặc trưng do Viola và Jones công bố gồm 4 đặc trưng cơ bản để xác định khuôn mặt người. Mỗi đặc trưng Haar-Like là sự kết hợp của hai hay ba hình chữ nhật trắng hay đen như trong hình sau:



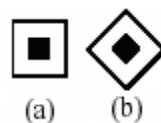
- Các hình trên còn được gọi là Haar features đại diện độ sáng tối của các vùng trên gương mặt là khác nhau. Ví dụ: vùng mắt tối hơn vùng má, vùng mũi sáng hơn vùng hai bên
- Kết quả của mỗi đặc trưng được tính bằng hiệu của tổng các pixel trong miền ô trắng trừ đi tổng các pixel trong miền ô đen.
- Để sử dụng các đặc trưng này vào việc xác định khuôn mặt người, 4 đặc trưng Haar-Like cơ bản được mở rộng ra và được chia làm 3 tập đặc trưng như sau:
- Đặc trưng cạnh(edge feature)



- Đặc trưng đường(line feature)



- Đặc trưng xung quanh tâm(center-surround features)



- Dùng các đặc trưng trên, ta có thể tính được các giá trị của đặc trưng Haar-Like là sự chênh lệch giữa tổng của các pixel của vùng đen và vùng trắng như trong công thức sau:

$$f(x) = \text{Tổng}_{\text{vùng đen}}(\text{các mức xám của pixel}) - \text{Tổng}_{\text{vùng trắng}}(\text{các mức xám của pixel})$$

- Viola và Joines đưa ra một khái niệm gọi là Integral Image, là một mảng 2 chiều với kích thước bằng với kích thước của ảnh cần tính đặc trưng Haar-Like, với mỗi phần tử của mảng này được tính bằng cách tính tổng của điểm ảnh phía trên (dòng-1) và bên trái (cột-1) của nó.
- Integral Image: giá trị ở pixel (x, y) là tổng của các pixel ở trên và bên trái (x,y). Cho phép tính tổng của các pixel trong bất kì hình chữ nhật chỉ với 4 giá trị ở 4 góc.



- Công thức tính Integral Image :

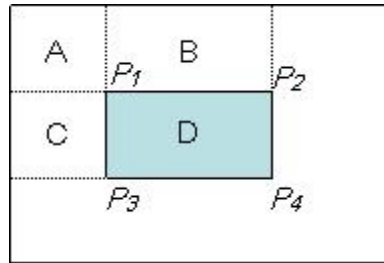
$$P(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

- Sau khi 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 giá trị mức xám của vùng D như hình dưới, ta có thể tính được như sau:

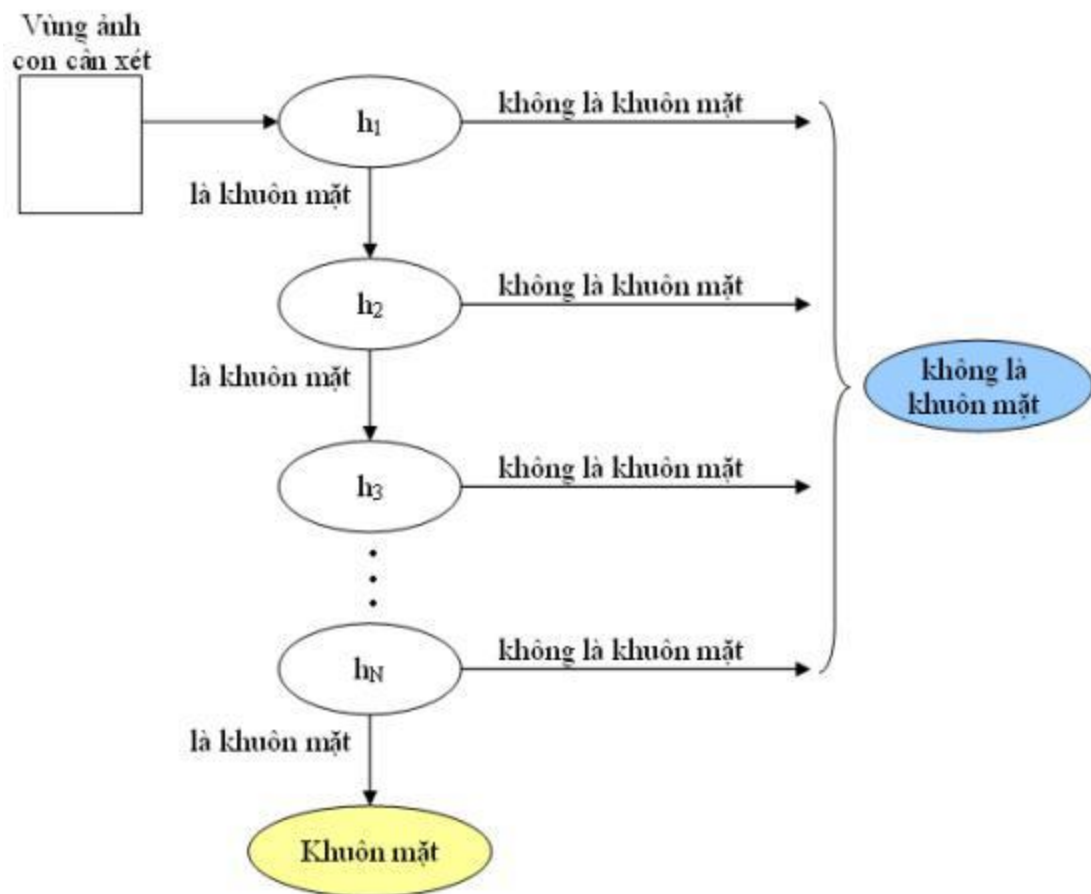
$$D = A + B + C + D - (A+B) - (A+C) + A$$

- 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:

$$D = \underbrace{(x_4, y_4)}_{A+B+C+D} - \underbrace{(x_2, y_2)}_{(A+B)} - \underbrace{(x_3, y_3)}_{(A+C)} + \underbrace{(x_1, y_1)}_A$$



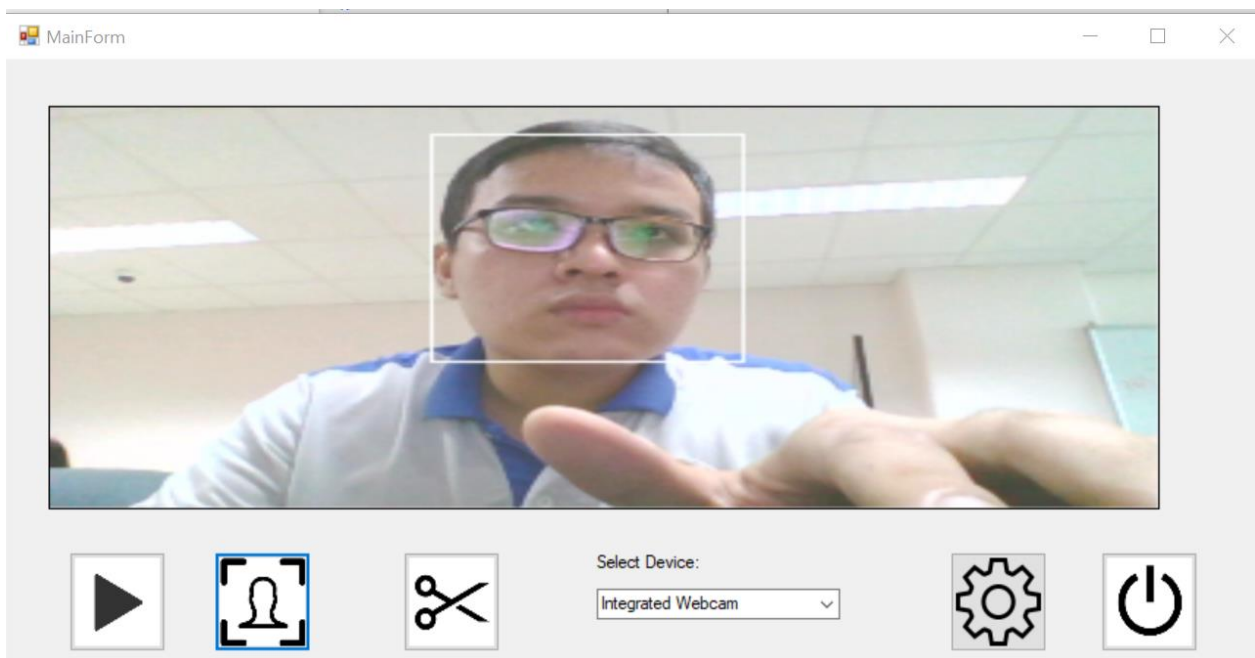
- Mặc dù một ảnh có thể chứa một hoặc nhiều khuôn mặt nhưng số lượng vật không phải khuôn mặt vẫn lớn hơn rất nhiều \Rightarrow thuật toán nên tập trung vào việc bỏ những vật không phải khuôn mặt một cách nhanh chóng.
- Một bộ phân lớp cascade (cascade classifier) được sử dụng tất cả các đặc trưng được nhóm vào vài stage. Mỗi stage gồm một số các đặc trưng.
- Mỗi stage được sử dụng để xác định một cửa sổ có phải là khuôn mặt hay không:



CHƯƠNG III: Demo

1. Nhận diện gương mặt thông qua webcam

Ảnh tổng quát demo



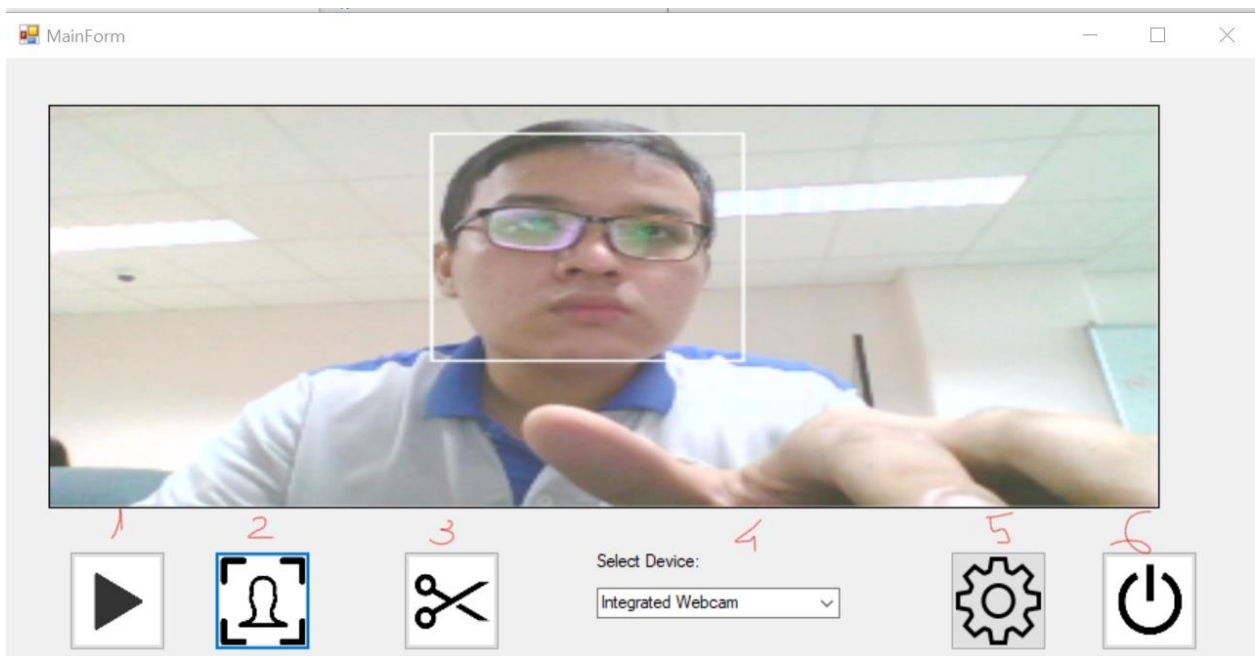
- Các package đã sử dụng:

	Accord by Accord.NET Accord.NET Framework core library.	v3.8.0
	Accord.Controls.Imaging by Accord.NET PictureBox with support for 16-bit images, color sliders, hue pickers, video player controls for Windows Forms.	v3.8.0
	Accord.Controls.Vision by Accord.NET Control an interface using head, face and hand tracking.	v3.8.0
	Accord.Imaging by Accord.NET Point detectors (i.e. SURF, Fast, Harris, SUSAN, HOG), large collection of image filters, texture generation, bag-of-visua...	v3.8.0
	Accord.MachineLearning by Accord.NET Supervised and unsupervised algorithms for classification, clustering, optimization and search, such as SVMs, Naive B...	v3.8.0
	Accord.Math by Accord.NET Matrix library, matrix decompositions (QR, SVD, LU, Cholesky, NMF), linear and non-linear optimization, constrained p...	v3.8.0
	Accord.Statistics by Accord.NET Conduct statistical analysis (i.e. PCA, KPCA, LDA, ANOVA), hypothesis tests, create Linear, Logistic and Generalized Line...	v3.8.0
	Accord.Video by Accord.NET Capture video from USB cameras, network (IP) cameras, folders of pictures and other devices.	v3.8.0
	Accord.Video.DirectShow by Accord.NET Capture, read and write videos using DirectShow, supporting USB web cameras, capture devices, video files and others.	v3.8.0
	Accord.Video.FFMPEG by Accord.NET Read and write videos using FFMPEG, including WebM, H.264 and Xvid/DivX files.	v3.8.0
	Accord.Video.VFW by Accord.NET Read and write AVI video files using Video For Windows (VFW).	v3.8.0
	Accord.Vision by Accord.NET Real-time face recognition, object tracking and computer vision library.	v3.8.0

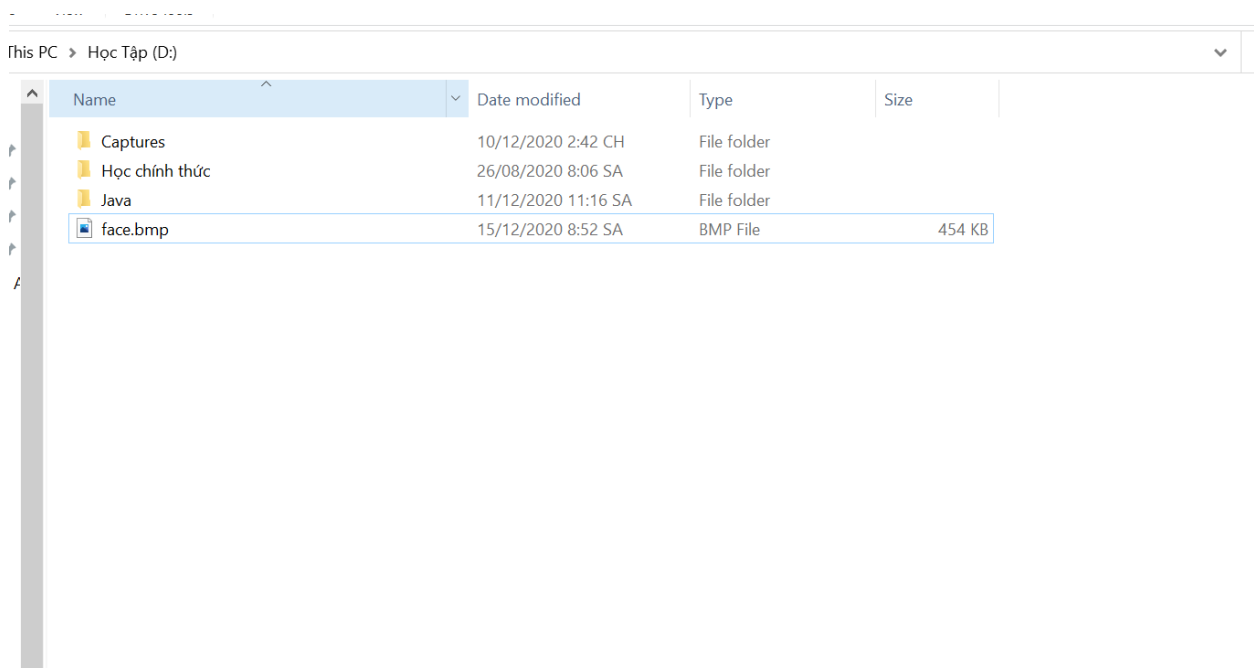
- **Các thư viện chính của Accord. NET được sử dụng bao gồm :**

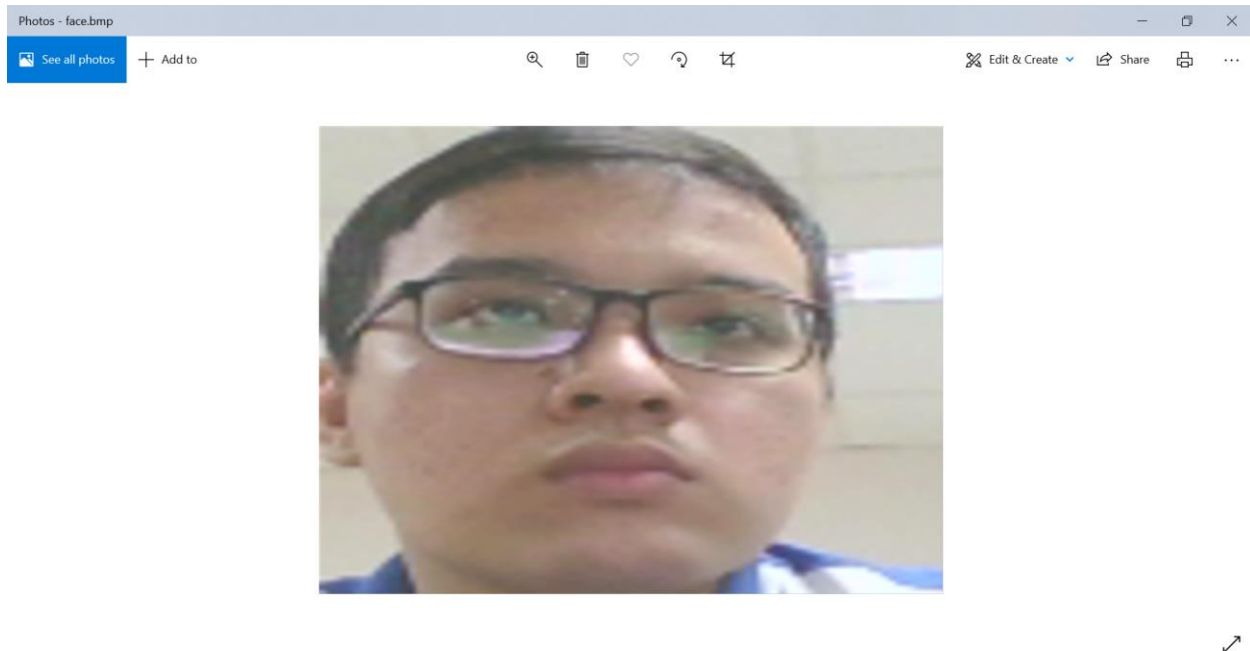
- Accord.Imaging.Filters : Thư viện về việc khai thác hình ảnh có chứa các function khai thác và lựa chọn các dữ liệu ảnh
- Accord.Vision.Detection : Chứa các thuật toán khai thác phân tích hình ảnh chung
- Accord.Vision.Detection.Cascades: Thư viện có chứa các thuật toán xử lý hình ảnh Cascades
- Accord.Video.DirectShow: Sử dụng thiết bị trực tiếp của thiết bị.
- Accord.Video: Sử dụng các hàm điều chỉnh hình ảnh quay màn hình hiển thị
- Accord.Imaging.Filters: Xử lý các khung hình được lấy từ từ việc quay màn hình và tiến dùng để train dữ liệu nhận diện hay edit hiển thị

Kết quả demo :

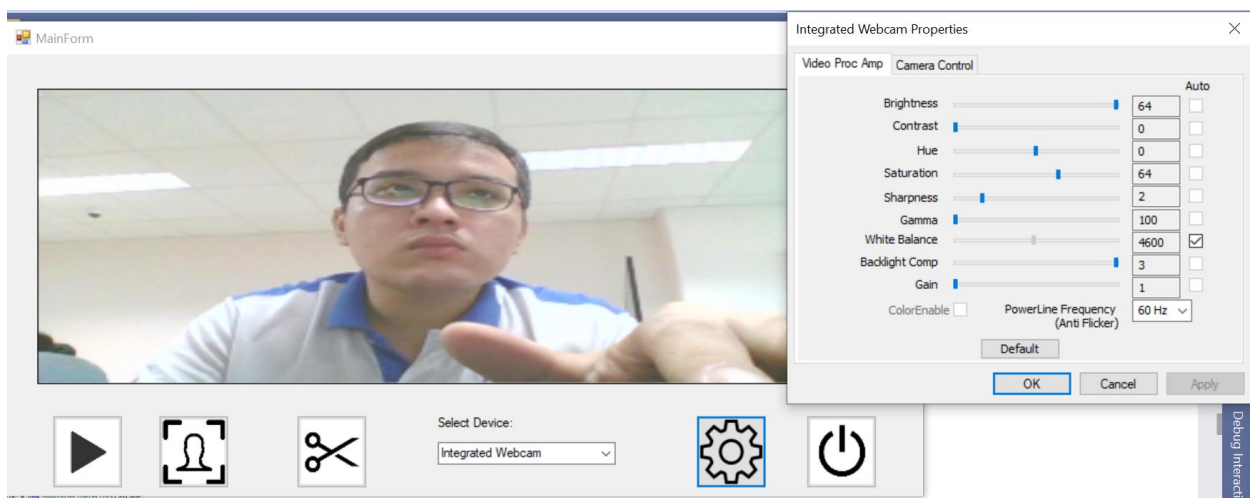


1. Khởi động quay màn hình
2. Tiến hành nhận diện khuôn mặt
3. Tiến hành chụp màn hình (ở đây là khuôn mặt)





4. Hiện thị các thiết bị quay màn hình có kết nối với máy
5. Cài đặt hình ảnh hiển thị



6. Thoát phần mềm

1.1 Giải thích chi tiết về code

- Các biến sẽ dùng để cài đặt dữ liệu và các hoạt động của Accord

```
private IVideoSource videoSource = null; //xử lý dữ liệu video

private HaarObjectDetector detector; //xử lý trạng thái nhận diện gương mặt

private Camshift tracker = null; //xử lý trạng thái quay của cam
```

```

private RectanglesMarker marker; //xử lý vẽ khung nhận diện gương mặt

private bool detecting = false; // biểu thị trạng thái nhận diện gương mặt
private bool tracking = false; // biểu thị trạng thái quay

private FilterInfoCollection videoDevices; // xác định thiết bị quay

private Rectangle rect1; //biến lấy khung vùng quay

private Bitmap grayImage; // lấy lý ảnh

```

```

public MainForm()
{
    InitializeComponent();
    HaarCascade cascade = new FaceHaarCascade();
    detector = new HaarObjectDetector(cascade,
        25, ObjectDetectorSearchMode.Single, 1.2f,
        ObjectDetectorScalingMode.GreaterToSmaller);

    try
    {
        videoDevices = new FilterInfoCollection(FilterCategory.VideoInputDevice);

        if (videoDevices.Count == 0)
            throw new ApplicationException();

        foreach (FilterInfo device in videoDevices)
        {
            devicesCombo.Items.Add(device.Name);
        }
    }
    catch (ApplicationException)
    {
        devicesCombo.Items.Add("Không tìm thấy thiết bị ghi hình");
        devicesCombo.Enabled = false;
        buttonPlay.Enabled = false;
    }

    devicesCombo.SelectedIndex = 0;
}

```

- Tùy chỉnh các hàm dành cho xác định khuôn mặt cho thuật toán khuôn mặt HaarCascade cũng như chọn chế độ xác định gương mặt của thuật toán đó.
- Ngoài ra còn có các lệnh tìm kiếm thiết bị thu hình được tích hợp với máy tính và truyền tên thiết bị ấy lên combobox trên giao diện để người dùng lựa chọn

```

private void OpenVideoSource(IVideoSource source)
{
    this.Cursor = Cursors.WaitCursor;
}

```



```

        CloseVideoSource();

        videoSourcePlayer1.VideoSource = source;
        videoSourcePlayer1.Start();

        videoSource = source;

        this.Cursor = Cursors.Default;
    }

    private void CloseVideoSource()
    {
        this.Cursor = Cursors.WaitCursor;

        videoSourcePlayer1.SignalToStop();

        for (int i = 0; (i < 50) && (videoSourcePlayer1.IsRunning); i++)
        {
            Thread.Sleep(100);
        }
        if (videoSourcePlayer1.IsRunning)
            videoSourcePlayer1.Stop();

        tracker = new Camshift();

        tracker.Conservative = true;
        tracker.AspectRatio = 1.5f;

        videoSourcePlayer1.BorderColor = Color.Black;
        this.Cursor = Cursors.Default;
    }

```

- Đây là hai phương thức việc mở dữ liệu là OpenVideoSource với mục đích bắt đầu sử dụng luồng dữ liệu được đưa vào tức những thứ quay từ cam ra màn hình hiển thị của giao diện. Và phương thức còn lại là xử lý việc đóng luồng dữ liệu quay đó và ngắt kết nối với thiết bị quay.

```

private static VideoCapabilities selectResolution(VideoCaptureDevice device)
{
    foreach (var cap in device.VideoCapabilities)
    {
        if (cap.FrameSize.Height == 240)
            return cap;
        if (cap.FrameSize.Width == 320)
            return cap;
    }

    return device.VideoCapabilities.Last();
}

```

- Phương thức để chỉnh sửa các kích frame của hình ảnh quay được từ cam để đưa lên màn hình hiển thị

```
private void buttonPlay_Click(object sender, EventArgs e)
{
    VideoCaptureDevice videoSource = new
VideoCaptureDevice(videoDevices[devicesCombo.SelectedIndex].MonikerString);

    videoSource.VideoResolution = selectResolution(videoSource);

    OpenVideoSource(videoSource);
}

private void buttonOption_Click(object sender, EventArgs e)
{
    if ((videoSource != null) && (videoSource is VideoCaptureDevice))
    {
        try
        {
            ((VideoCaptureDevice)videoSource).DisplayPropertyPage(this.Handle);
        }
        catch (NotSupportedException)
        {
            MessageBox.Show("Không có video ", "Lỗi",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
}
```

- Nút Play chọn thiết bị sử dụng từ combobox và tiến hành mở luồng dữ liệu từ cam đến màn hình hiển thị. Nút Option có mục đích sử dụng form cài đặt cấu hình được truyền lên đã được tích hợp sẵn trong Accord.

```
private void videoSourcePlayer1_NewFrame(object sender, ref Bitmap image)
{
    if (!detecting && !tracking)
        return;

    lock (this)
    {
        if (detecting)
        {
            detecting = false;
            tracking = false;

            UnmanagedImage im = UnmanagedImage.FromManagedImage(image);
        }
    }
}
```

```

float xscale = image.Width / 160f;
float yscale = image.Height / 120f;

ResizeNearestNeighbor resize = new ResizeNearestNeighbor(160, 120);
UnmanagedImage downsample = resize.Apply(im);
grayImage = (Bitmap)image.Clone();
Rectangle[] regions = detector.ProcessFrame(downsample);

if (regions.Length > 0)
{
    tracker.Reset();

    Rectangle face = regions[0];

    Rectangle window = new Rectangle(
        (int)((regions[0].X + regions[0].Width / 2f) * xscale),
        (int)((regions[0].Y + regions[0].Height / 2f) * yscale),
        1, 1);

    window.Inflate(
        (int)(0.4f * regions[0].Width * xscale),
        (int)(0.6f * regions[0].Height * yscale));

    tracker.SearchWindow = window;
    tracker.ProcessFrame(im);

    marker = new RectanglesMarker(window);
    rect1 = window;
    marker.ApplyInPlace(im);

    image = im.ToManagedImage();

    tracking = true;
    detecting = true;
}
else
{
    detecting = true;
}
}
else
{
    if (marker != null)
        image = marker.Apply(image);
}
}
}

```

- Đây là phần xử lý hình ảnh được lấy từ Cam và tiến hành tái cấu trúc thành dữ liệu để đưa vào thuật toán tiến hành huấn luyện với việc lấy từng khung ảnh từ màn hình **Bitmap image** đến việc chuyển đổi kích thước hình ảnh đồng thời

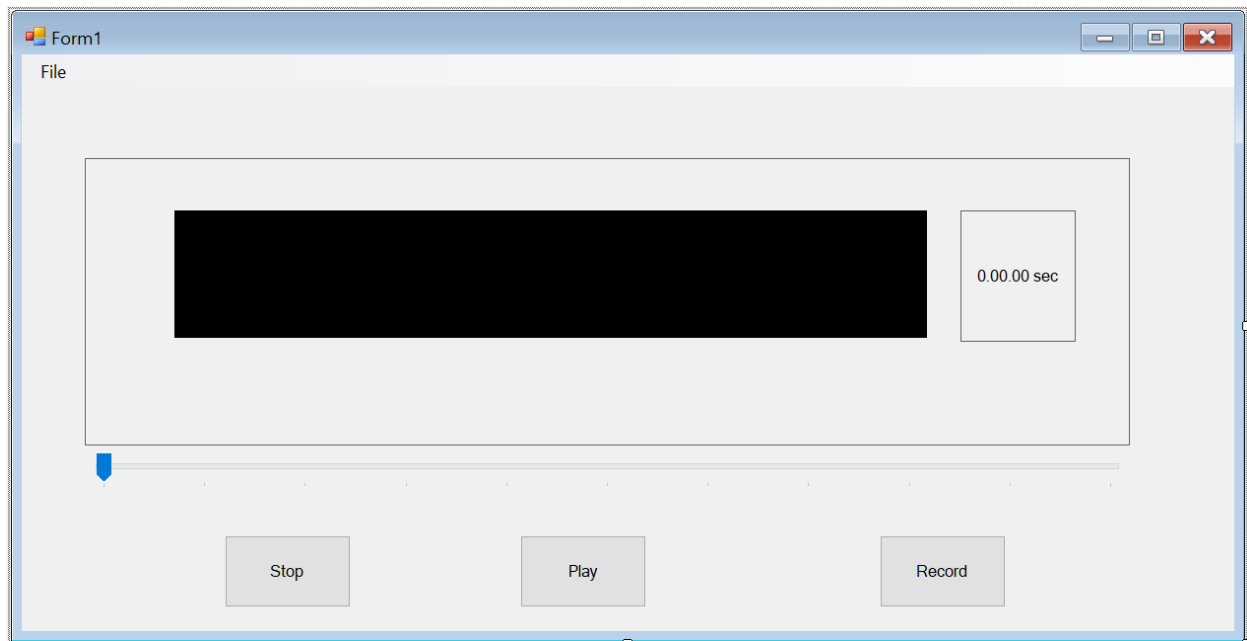
thiết định được cấu trúc khoan vùng khuôn mặt cũng như phạm vi khoanh qua biến **Rectangle window**.

```
private void buttonCapture_Click(object sender, EventArgs e)
{
    try
    {
        if (rect1 != null)
        {
            Crop crop = new Crop(rect1);
            Bitmap newImage = crop.Apply(grayImage);
            ResizeBicubic resixe = new ResizeBicubic(640, 480);
            newImage = resixe.Apply(newImage);
            newImage.Save(@"D:\face.bmp");
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.ToString());
    }
}
```







- Tiến hành chụp màn hình được tích hợp trong Accord chúng ta lấy được các thông tin để chụp màn hình thông qua 2 biến là **rec1** dùng để lấy kích thước chụp khuôn mặt và **grayimage** để lấy được hình ảnh được truyền vào lúc quay. Tiếp đó dùng hàm **Crop** để cắt khung hình theo **rec1** nơi mà khuôn mặt được khoanh vì **rec1** chứa tọa độ nói khoanh mặt. Cuối cùng là tiến hành lưu hình ảnh đó.







2. Phần mềm thu âm (Audio record)

Ảnh tổng quát



- Các Package đã sử dụng

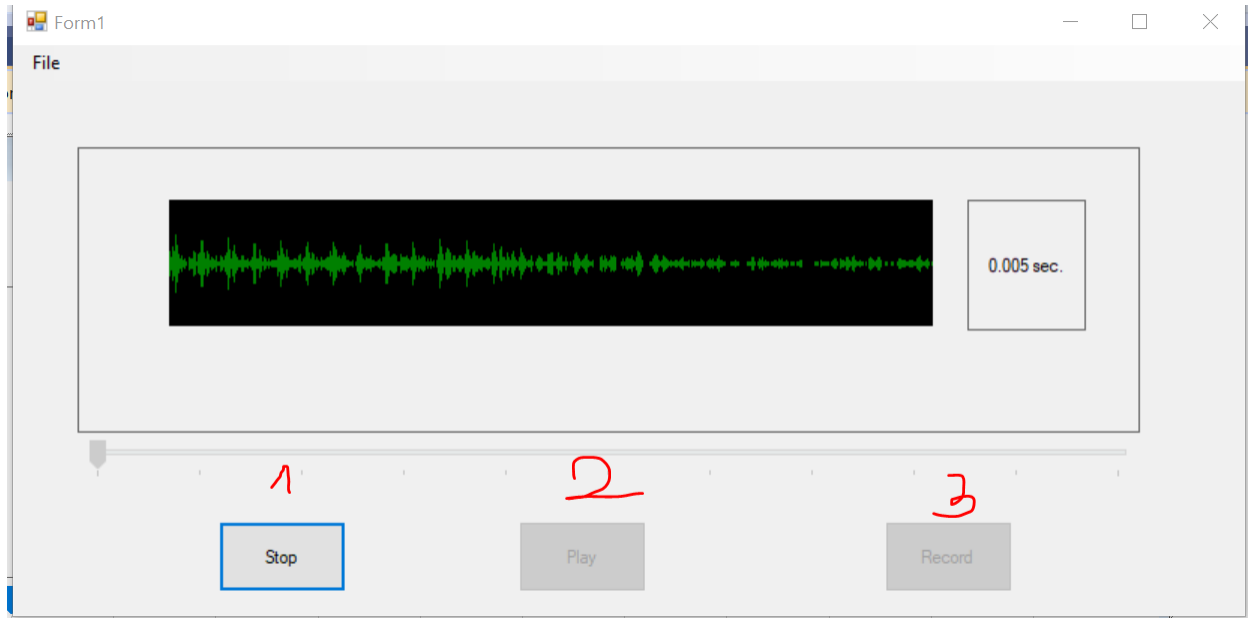
	Accord by Accord.NET Accord.NET Framework core library.	v3.8.0
	Accord.Audio by Accord.NET Audio filters (i.e. envelope, high/low-pass, FFT), temporal windows (i.e. Blackman, raised cosine), signal generators and...	v3.8.0
	Accord.Controls by Accord.NET Plot histograms, scatterplots, tabular data, matrices and other objects in Windows Forms applications.	v3.8.0
	Accord.Controls.Audio by Accord.NET Display waveforms and audio-related visualizations in Windows Forms.	v3.8.0
	Accord.DirectSound by Accord.NET Play and record audio using DirectSound.	v3.8.0
	Accord.MachineLearning by Accord.NET Supervised and unsupervised algorithms for classification, clustering, optimization and search, such as SVMs, Naive B...	v3.8.0

	Accord.MachineLearning by Accord.NET	v3.8.0
	Supervised and unsupervised algorithms for classification, clustering, optimization and search, such as SVMs, Naive B...	
	Accord.Math by Accord.NET	v3.8.0
	Matrix library, matrix decompositions (QR, SVD, LU, Cholesky, NMF), linear and non-linear optimization, constrained p...	
	Accord.Statistics by Accord.NET	v3.8.0
	Conduct statistical analysis (i.e. PCA, KPCA, LDA, ANOVA), hypothesis tests, create Linear, Logistic and Generalized Line...	
	SharpDX by Alexandre Mutel	v4.2.0
	Core assembly for all SharpDX assemblies.	
	SharpDX.DirectSound by Alexandre Mutel	v4.2.0
	Assembly providing DirectX - DirectSound managed API.	
	ZedGraph by ZedGraph Project	v5.1.7
	ZedGraph is a class library, user control, and web control for .net, written in C#, for drawing 2D Line, Bar, and Pie Chart...	

- **Các thư viện chính của Accord. NET được sử dụng bao gồm :**

- Accord.Audio : sử dụng cung cấp các hàm xử lý âm thanh tổng quát về phần Audio
- Accord.Audio.Formats: cung cấp các hàm mã hóa đầu thu và giải mã hóa
- Accord.DirectSound: cung cấp các hàm lấy thiết bị thu âm.

Kết quả demo :



1. Dừng chức năng quay hoặc dừng chức năng nghe lại đoạn thu âm
2. Chạy đoạn thu âm vừa thu được
3. Thu âm từ thiết bị
4. Cuối cùng là menu file với việc lựa chọn lưu đoạn thu âm và đóng chương trình.

2.1. Giải thích code

- Các biến được sử dụng

```
private MemoryStream stream; //xử lý các luồng lưu dữ liệu tạm thời

private IAudioSource source; //xử lý nguồn âm thanh vào (thu âm)
private IAudioOutput output; // xử lý nguồn âm thanh xuất (nghe)

private WaveEncoder encoder; //mã hóa âm thanh
private WaveDecoder decoder; // giải mã âm thanh

private float[] current; //xử lý lấy trạng thái chạy bản ghi âm

private int frames; //để lấy các frame
private int sample; // để lấy các sample từ loa truyền vào

private TimeSpan duration; // đếm thời gian kéo dài của bản thu âm
```

```

public Form1()
{
    InitializeComponent();
    wavechart1.SimpleMode = true;
    wavechart1.AddWaveform("wave", Color.Green, 1, false);
    updateButtons();
}

```

- Xử lý cơ bản chế độ hình ảnh hiển thị của sơ đồ âm thanh khi thi.

```

private void updateButtons()
{
    if (InvokeRequired)
    {
        BeginInvoke(new Action(updateButtons));
        return;
    }

    if (source != null && source.IsRunning)
    {
        buttonPlay.Enabled = false;
        buttonStop.Enabled = true;
        buttonRecord.Enabled = false;
        trackBar1.Enabled = false;
    }
    else if (output != null && output.IsRunning)
    {
        buttonPlay.Enabled = false;
        buttonStop.Enabled = true;
        buttonRecord.Enabled = false;
        trackBar1.Enabled = true;
    }
    else
    {
        buttonPlay.Enabled = stream != null;
        buttonStop.Enabled = false;
        buttonRecord.Enabled = true;
        trackBar1.Enabled = decoder != null;
        trackBar1.Value = 0;
    }
}

```

- Xử lý các trạng thái hiển thị của các nút theo từng trường hợp khác nhau như khi nhấn nút thu thì nút dừng bật lên và nút play thì vô hiệu hóa.


```

private void updateWaveForm(float[] sample, int length)
{
    if (InvokeRequired)
    {
        BeginInvoke(new Action(() => wavechart1.UpdateWaveform("wave",
sample, length)));
    }
    else
    {
        wavechart1.UpdateWaveform("wave", current, length);
    }
}

private void updateTrackBar(int value)
{
    if (InvokeRequired)
    {
        BeginInvoke(new Action(() => trackBar1.Value =
Math.Max(trackBar1.Minimum, Math.Min(trackBar1.Maximum, value))));
    }
    else
    {
        trackBar1.Value = Math.Max(trackBar1.Minimum,
Math.Min(trackBar1.Maximum, value));
    }
}

```

- Hai hàm trên với mục đích là cập nhật trạng thái của sơ đồ âm thanh khi thu và thanh tiến trình phát đoạn thu khi phát.

```

private void source_NewFrame(object sender, NewFrameEventArgs eventArgs)
{
    eventArgs.Signal.CopyTo(current);

    updateWaveForm(current, eventArgs.Signal.Length);
    encoder.Encode(eventArgs.Signal);
    duration += eventArgs.Signal.Duration;
    sample += eventArgs.Signal.Samples;
    frames += eventArgs.Signal.Length;
}

```

- Khi bắt đầu record thì khi đó một Frame dữ liệu mới bắt đầu và phương thức này đại diện cho điều đó đồng thời truyền vào các sự kiện có trong Frame đó vào các biến đã tạo ở trên là duration, sample, frame.

```

private void output__FramePlayingStarted(object sender, PlayFrameEventArgs e)
{
    updateTrackBar(e.FrameIndex);
    if (e.FrameIndex + e.Count < decoder.Frames)
    {

```

```

        int previous = decoder.Position;
        decoder.Seek(e.FrameIndex);

        Signal s = decoder.Decode(e.Count);
        decoder.Seek(previous);
        updateWaveForm(s.ToFloat(), s.Length);
    }
}

private void output_PlayingFinished(object sender, EventArgs e)
{
    updateButtons();
    Array.Clear(current, 0, current.Length);
    updateWaveForm(current, current.Length);
}

private void output_NewFrameRequested(object sender, NewFrameRequestedEventArgs e)
{
    e.FrameIndex = decoder.Position;

    Signal signal = decoder.Decode(e.Frames);

    if (signal == null)
    {
        e.Stop = true;
        return;
    }
    e.Frames = signal.Length;
    signal.CopyTo(e.Buffer);
}

```

- Ba phương thức trên với mục đích tuần tự là yêu cầu play bản record vừa được thu, dừng nghe bản vừa thu cùng reset sơ đồ sóng âm lại, phương thức cuối với mục đích nhận lấy và đặt các yêu cầu phát âm thanh.

```

private void buttonRecord_Click(object sender, EventArgs e)
{
    source = new AudioCaptureDevice()
    {
        DesiredFrameSize = 4096,
        SampleRate = 22050,
        Format = SampleFormat.Format16Bit
    };

    source.NewFrame += source_NewFrame;
    source.AudioSourceError += source_AudioSourceError;
    current = new float[source.DesiredFrameSize];

    stream = new MemoryStream();
    encoder = new WaveEncoder(stream);
}

```

```

        source.Start();
        updateButtons();
    }

    private void buttonPlay_Click(object sender, EventArgs e)
    {
        stream.Seek(0, SeekOrigin.Begin);

        decoder = new WaveDecoder(stream);

        if (trackBar1.Value < decoder.Frames)
            decoder.Seek(trackBar1.Value);
        trackBar1.Maximum = decoder.Samples;

        output = new AudioOutputDevice(this.Handle, decoder.SampleRate,
decoder.Channels);

        output.FramePlayingStarted += output__FramePlayingStarted;
        output.NewFrameRequested += output_NewFrameRequested;
        output.Stopped += output_PlayingFinished;

        output.Play();

        updateButtons();
    }

    private void buttonStop_Click(object sender, EventArgs e)
    {
        if (source != null)
        {
            source.SignalToStop();
            source.WaitForStop();
        }
        if (output != null)
        {
            output.SignalToStop();
            source.WaitForStop();
        }

        updateButtons();
        Array.Clear(current, 0, current.Length);

        updateWaveForm(current, current.Length);
    }
}

```

- Khi nút record được bấm thì nó sẽ khởi tạo một phương thức lấy âm thanh của thư viện **Accord** là **AudioCaptureDevice** đồng thời điều chỉnh lại âm thanh thu được. Cùng sử dụng **source_NewFrame** đã được tạo ở trên để lấy các thông số của đoạn thu cùng mã hóa nó qua

encoder. Đồng thời lưu đoạn thu đã được mã hóa đó vào **Memory Stream** là biến **stream** với vai trò là bộ nhớ tạm thời.

- Với nút Play thì chúng ta sẽ lấy phần dữ liệu tạm thời được lưu trong biến **stream** và tiến hành giải mã hóa và sử dụng các biến xử lý sự kiện đã được khai báo trước đó là **output__FramePlayingStarted**, **output_NewFrameRequested**, **output_PlayingFinished** để xử lý sự kiện của thanh track và sơ đồ sóng âm thanh.
- Nút Stop với mục đích là việc dừng việc play đoạn thu âm khi nút Play bật và dừng thu khi nút stop bật.

```
private void saveFileDialog1_FileOk(object sender, CancelEventArgs e)
{
    Stream fileStream = saveFileDialog1.OpenFile();
    stream.WriteTo(fileStream);
    fileStream.Close();
}
```

- Phương thức được sử dụng để lưu dữ liệu có trong Memory Stream hiện tại thành dạng file.

Tài liệu tham khảo

1. [Accord.NET Machine Learning Framework \(accord-framework.net\)](http://accord-framework.net)