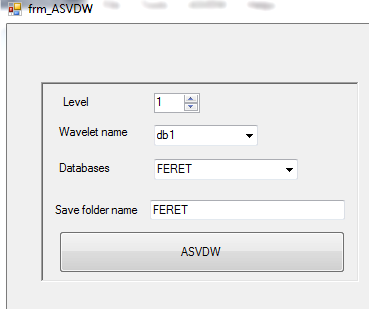
**Hướng dẫn sử dụng các toolbox trong C++**

1. **Sử dụng ComboBox để tạo đường dẫn thư mục ảnh cần xử lý và lưu kết quả sang foder khác.**

Trong mục Toolbox, tạo comboBox

Nhập danh sách tên forder trong phần Collection

Tạo TextBox để hiển thị tên của forder lựa chọn cần lưu trữ.



Click vào phím button và thêm dãy câu lệnh sau vào, phần này có chức năng là khi khởi động chương trình, sẽ yêu cầu lựa chọn forder ảnh cần xử lý và lưu ảnh sang 1 forder mới. Ban đầu sẽ bắt buộc yêu cầu lựa chọn forder  
  
Code:

System::String^ pathfolder;

//txt\_savefolder->Text= cB\_dataname->Text;

if (txt\_savefolder->Text== "")

{

MessageBox::Show("Please put the Folder to save ASVDW images","Color Face Image Enhancement Using ASVDW", MessageBoxButtons::OK,MessageBoxIcon::Asterisk);

return;

}

else

{

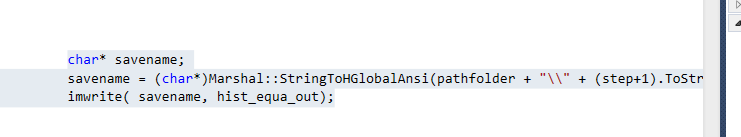
pathfolder ="..\\" + txt\_savefolder->Text;

marshal\_context context;

LPCTSTR cstr = context.marshal\_as<const TCHAR\*>(pathfolder);

CreateDirectory (cstr, NULL);

}

Tiếp theo, dung lệnh imwrite để in ra ảnh mới sau khi xử lý, để tránh nhầm lẫn ảnh sau xử lý và trước khi xử lý, phải in ảnh sau xử lý ra 1 foder mới với tên ảnh mới để phân biệt với ảnh cũ. Dùng đoạn code sau vào trong chương trình C++  
  


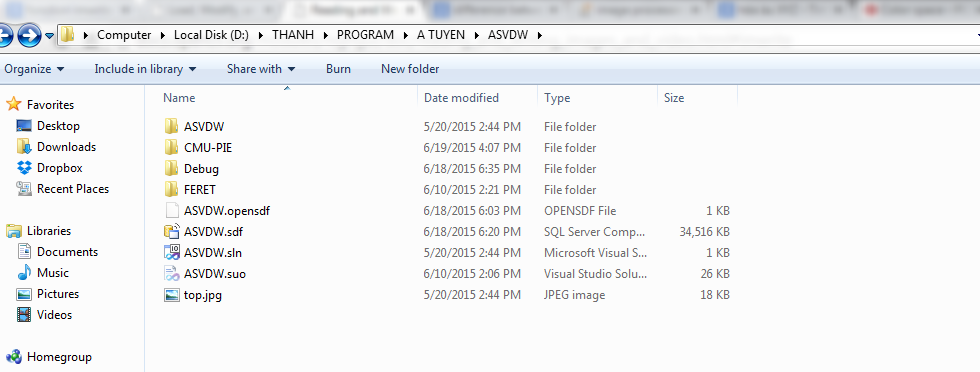
Code:

char\* savename;

savename = (char\*)Marshal::StringToHGlobalAnsi(pathfolder + "\\" + (step+1).ToString()+".bmp").ToPointer();

imwrite( savename, hist\_equa\_out);

Ảnh sau khi xử lý sẽ được save lại trong 1 forder mới, forder này nằm trong nơi lưu trữ chương trình Project.



Trên hình, ảnh sau khi xử lý sẽ được save vào forder CMU-PIE hoặc FERET

(đây là các thư viện về ảnh mặt người).

1. **Hàm chuyển đổi từ std::String sang System::String (hiển thị string của window form)**

// Lấy string từ *std*::string

*std*::string n = symbol->get\_data();

// Hàm chuyển sang System::string của Windows Form

*System*::String^ result = gcnew *System*::Stri(n.*c\_str*());

*Console*::*WriteLine*(result);

// Hiển thị kết quả lên textbox

textBox1->*Text* = result;

1. **Hàm chuyển đổi từ System::String sang std::String**

// convert from 'System::String ^' to 'std::string'

*msclr*::*interop*::*marshal\_context* context;

*std*::string output\_name1 = context.*marshal\_as*<*std*::string>(output\_name);

1. **Openfiledialoge**

// ----------------OPEN FILE IMAGE---------------

openFileDialog->*Filter* = "Image Files|\*.bmp; \*.jpg; \*.png|All Files (\*.\*)|\*.\*||"; openFileDialog->*Title* = "Select a image file";

openFileDialog->*InitialDirectory* = "";

openFileDialog-> *Multiselect* = "True";

int number\_of\_files;

array<*System*::String^>^ filename\_src; // array of filenames

if ( openFileDialog->*ShowDialog*() == *System*::*Windows*::*Forms*::*DialogResult*::*OK* )

{

filename\_src = openFileDialog->*FileNames*;

number\_of\_files = openFileDialog->*FileNames*->*Length*;

}

else return;

*System*::String^ OpenFileName;

*System*::String^ step\_str;

int step = 0;

while (step <number\_of\_files)

{

OpenFileName = filename\_src[step];

OpenFileName = OpenFileName->Replace("\\","\\\\");

char\* imgname = (char\*)Marshal::StringToHGlobalAnsi(OpenFileName).ToPointer(); // convert to kieu char lay ten filecho opencv

Mat img = cv::imread(imgname);

Mat gray = img.clone();

cvtColor(img, gray , CV\_BGR2GRAY);

// Ket thuc mo file

/////////////// chuyen sang file tiep theo ////////////////

step++;

}

1. **Truy xuất điểm ảnh**

for (int i = 0; i < abc.rows; i++)

for (int j = 0; j < abc.cols; j++)

{

abc.at<float>(cv::Point(i , j)) = (255 - abc.at<float>(cv::Point(i , j)));

}

1. **Tạo cửa sổ TRACK BAR**

int iLowH = 0;

int iHighH = 179;

int iLowS = 0;

int iHighS = 255;

int iLowV = 0;

int iHighV = 255;

namedWindow("Control", CV\_WINDOW\_AUTOSIZE); //create a window called "Control"

cvCreateTrackbar("LowH", "Control", &iLowH, 179); //Hue (0 - 179)

cvCreateTrackbar("HighH", "Control", &iHighH, 179);

cvCreateTrackbar("LowS", "Control", &iLowS, 255); //Saturation (0 - 255)

cvCreateTrackbar("HighS", "Control", &iHighS, 255);

cvCreateTrackbar("LowV", "Control", &iLowV, 255); //Value (0 - 255)

cvCreateTrackbar("HighV", "Control", &iHighV, 255);

1. **Find and Drawn Contours**

*vector*<*vector*<cv::Point>> contours;

findContours(imgROI, contours, CV\_RETR\_LIST , CV\_CHAIN\_APPROX\_SIMPLE);

Mat drawing = Mat::zeros( imgROI.size(), CV\_8UC1 );

for( int i = 0; i< contours.*size*(); i++ )

{

drawContours(img, contours, i, Scalar (112,255,255), 1 ,8);

}

1. **Threshold Binary**

threshold(imgROI,imgROI ,100,255, CV\_THRESH\_BINARY );

1. **Collect color range of image**

inRange(imgHLS, Scalar(0, 100, 100), Scalar(20, 255, 255), imgROI);

1. **Display image on picture box**

int rows\_img = (img.rows/4)\*4;

int cols\_img = (img.cols/4)\*4;

Mat display\_img = Mat(rows\_img, cols\_img , CV\_8UC3);

resize(img, display\_img, display\_img.size(), 0, 0, INTER\_LINEAR);

pB\_src->*Image* = gcnew

*System*::*Drawing*::*Bitmap*(display\_img.size().width, display\_img.size().height, display\_img.step,

*System*::*Drawing*::*Imaging*::*PixelFormat*::*Format24bppRgb*,(*System*::*IntPtr*) display\_img.data);

pB\_src->*Refresh*();

1. **Split database (classify image to class or individual folder)**

OpenFileName = filename\_src[number\_image\*i+j];

*System*::String^ path;

path = *System*::*IO*::*Path*::*GetDirectoryName*(OpenFileName);

OpenFileName = OpenFileName->*Replace*("\\","\\\\");

path = path->*Replace*("\\","\\\\");

char\* imgname = (char\*)*Marshal*::*StringToHGlobalAnsi*(OpenFileName).*ToPointer*();

// convert to kieu char lay ten filecho opencv

Mat src = cv::imread(imgname);

filename = (number\_image\*i+j+1).*ToString*()+".bmp";

SaveFileName = path +"\\test\\" + filename;

char\* savename = (char\*)*Marshal*::*StringToHGlobalAnsi*(SaveFileName).*ToPointer*();

imwrite(savename,src);

**0 .**

1. **Hàm lấy giá trị từ ô TEXT trên window form chuyển thành giá trị integer**  
   EnergyThreshold=*System*::*Convert*::*ToInt32*(txt\_energythreshold->*Text*);

PixelLimit=*System*::*Convert*::*ToInt32*(txt\_pixellimit->*Text*);

1. **Tính logarit cơ số bất kỳ**

**Ví dụ 1.** Cho α=log220. Hãy tính log205 theo α

\* Vì hai logarit không cùng cơ số nên ta sẽ đổi cơ số của logarit cần tính về cơ số của logarit đã cho

log205=log25 / log220

1. Chú ký khi dùng các hàm toán học như : exp(n); log(n); log10(n) ….Thì n phải là số dạng float hoặc double

Có thể đưa gia tri **“n”** kiểu float hoặc double vào vòng lặp (như while, do…while, for …) sau đó cho “n++” vẫn thỏa mãn n tăng teo kiểu số nguyên.

Ví dụ:

float n = 1;

int temp = 0;

float tempS, S;

while(tempS < 2000.)

{

float x, y;

x = *exp*(n);

y = *log10*(n);

S = float (x - 1999\*y);

tempS +=S;

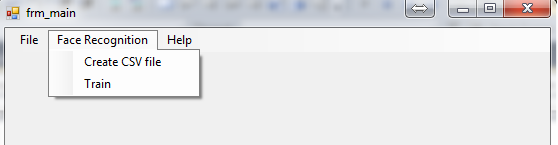
n++;

temp++;

}

1. **Cách tạo danh sách cửa sổ làm việc trên window form**

**Bước 1**: Từ cửa sổ Toolbox chọn menuTrip add vào Form

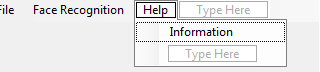


**Bước 2:** click vào phần main windowform, chọn

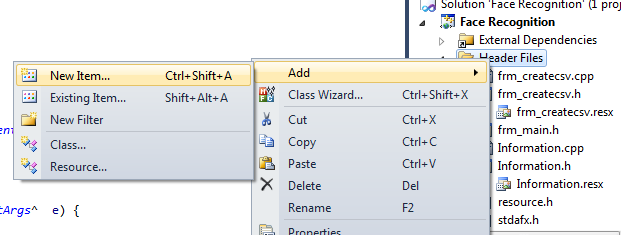
D:\THANH\MASTER\PROGRAM\New folder\Capture.PNG

Điền tên cửa sổ cần tạo ở đây (có thể lựa chọn, thay đổi các properties của cửa sổ đã tạo)

**Bước 3**: Có thể tạo tiếp cửa sổ ở danh sách này, ở đây có cửa sổ **Information**



**Bước 4**: Tạo windows form mới với tên **Information**



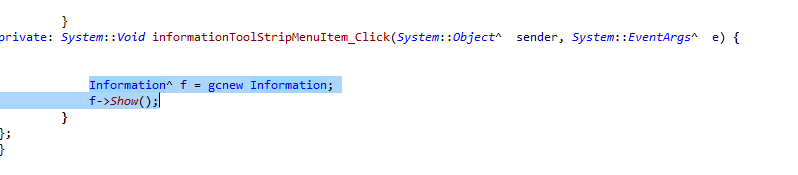
**Bước 5:** Quay lại cửa sổ **main,** click double vào cửa sổ information vừa tạo ở bước 2.

Thêm dòng code sau:

Information^ f = gcnew Information;

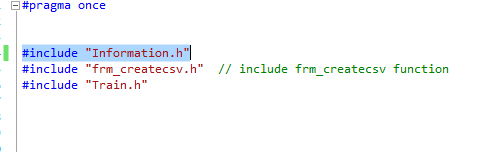
f->*Show*();

với information là tên cửa cửa sổ, có thể thay thế tên khác



Tiếp theo lên đầu của trang đang code, thêm lệnh

#include "Information.h"



**Bước 6**: Chạy chương trình và xem kết quả

1. **Opencv library**

**$(OPENCV\_DIR)\include**

**$(OPENCV\_DIR)\lib**

opencv\_core243d.lib

opencv\_imgproc243d.lib

opencv\_highgui243d.lib

opencv\_ml243d.lib

opencv\_video243d.lib

opencv\_features2d243d.lib

opencv\_calib3d243d.lib

opencv\_objdetect243d.lib

opencv\_contrib243d.lib

opencv\_legacy243d.lib

opencv\_flann243d.lib

wavelet2d.lib

1. **Project header**

#include <iostream>

#include <fstream>

#include <string>

#include <fstream> // lam viec voi file text

#include <sstream>

#include <stdlib.h>

#include <Windows.h>

#include <msclr/marshal\_cppstd.h>

#include <opencv2/calib3d/calib3d.hpp>

#include <opencv2/contrib/contrib.hpp>

#include <opencv2/core/core.hpp>

#include <opencv2/features2d/features2d.hpp>

#include <opencv2/highgui/highgui.hpp>

#include <opencv2/imgproc/imgproc.hpp>

using namespace *std*;

using namespace cv;

using namespace *System*::*Runtime*::*InteropServices*;

using namespace *msclr*::*interop*;

using namespace *System*::*Data*;

using namespace *System*::*Drawing*;

1. **Trích xuất ROI của ảnh dựa theo contour**

/////////// Access to label of image for processing ////////////////

*vector*<*vector*<cv::Point>> contours;

*vector*<Vec4i> hierarchy;

findContours(superpixel\_mask, contours , hierarchy, CV\_RETR\_TREE, CV\_CHAIN\_APPROX\_SIMPLE, cv::Point(0, 0) );

Mat region\_of\_label;

Mat drawing = Mat::zeros(img.size(), CV\_8UC3 );

for (int i = 0; i < contours.*size*(); i++)

{

drawContours(img , contours, i , cv::Scalar(0,255,255), 2, 8, hierarchy, 0, cv::Point());

Rect r = boundingRect(contours[i]);

region\_of\_label = (img(r)); /////// extract processing label

//////////////// Process region of label //////////////

Mat region\_label\_binary;

cv::threshold(region\_of\_label, region\_label\_binary ,100,255, CV\_THRESH\_BINARY );

region\_label\_binary.copyTo(region\_of\_label);\*/

}

1. **Trích xuất ROI của ảnh với hình dạng bất kỳ (labels superpixel)**

//////////////// Mask for each label

Mat superpixel\_mask; //// create mask of label region

superpixel\_mask = labelImg == 9; //// take label to mask image

Mat superpixel\_in\_img;

img.copyTo(superpixel\_in\_img, superpixel\_mask);

////////////////// Process ROI labels /////////////

cv::threshold(superpixel\_in\_img, superpixel\_in\_img ,100,255, CV\_THRESH\_BINARY );

///////////////// After processing, copy processed labels to original image

superpixel\_in\_img.copyTo(img, superpixel\_mask);

imshow("",img);

1. **Biến đổi SVD (Singular Value Decomposition)**

/////////// SVD transform /////////

int m = img.rows;

int n = img.cols;

////////// Create (float) Mat to store color channels ////////////

Mat B = Mat\_<float>(m,n);

Mat G = Mat\_<float>(m,n);

Mat R = Mat\_<float>(m,n);

Mat img\_src;

/////////// Have to convert Image from 8Uc1 to CV\_32F //////////////

img.convertTo(img\_src, CV\_32F);

////////////// Spit image to 3 color channels //////////////

for (int i = 0;i<img.cols;i++)

{

for (int j =0; j<img.rows;j++)

{

B.at<float>(cv::Point(i, j)) = img\_src.at<Vec3f>(cv::Point(i, j))[0];

G.at<float>(cv::Point(i, j)) = img\_src.at<Vec3f>(cv::Point(i, j))[1];

R.at<float>(cv::Point(i, j)) = img\_src.at<Vec3f>(cv::Point(i, j))[2];

}

}

//////// SVD transform /////////

cv::SVD svd(B); //// create structure SVD and implement SVD transform on B image

//////// SVD Inverse transform /////////

Mat B\_svd = (svd.u)\*Mat::diag(svd.w)\*svd.vt;

////////// Other way to implement SVD transform

//Mat w, u, vt;

//cv::SVD::compute(R,w,u,vt,cv::SVD::FULL\_UV);

// ////// Finds the global minimum and maximum in an array and store it

//double minw, maxw;

//minMaxLoc(w, &minw, &maxw);

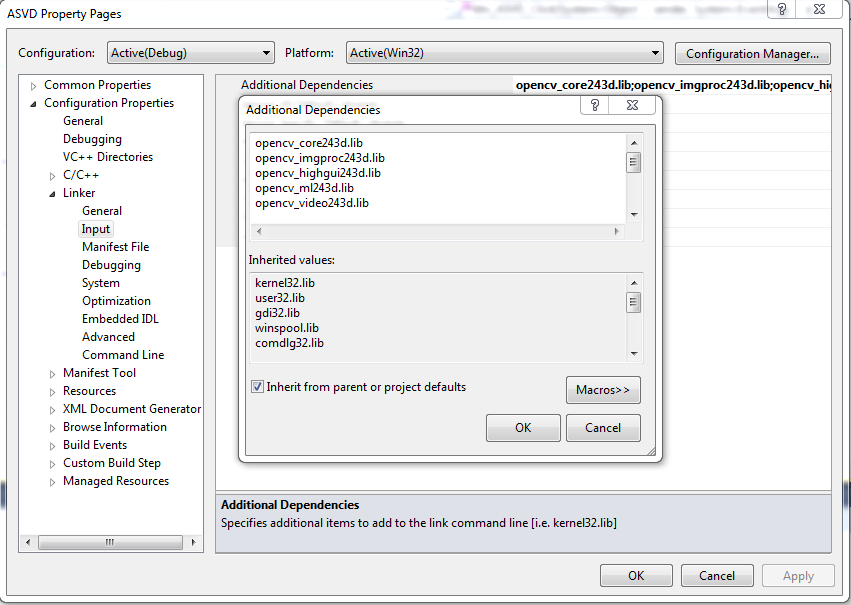
//////// To display image, have to convert image from CV\_32F to CV\_8UC1

svd.w.convertTo(svd.u, CV\_8UC1);

B\_svd.convertTo(B\_svd, CV\_8UC1);

imshow("",svd.w);

1. **Lỗi “ vector subscript out of range line 932”**

****

1. **Thực hiện phép biến đổi Wavelet**

//////////\*\*\*\*\*\*\*\*\*\*\* DWT Implement \*\*\*\*\*\*\*\*\*\*

int rows = img.rows;

int cols = img.cols;

Mat img\_src(img); ///// clone source image

/////// create 2D vector to store image /////////

*vector*<*vector*<double> > vec\_input(rows , *vector*<double>(cols));

/////// Put pixel value to 2D vector /////////

for (int i=0; i < rows; i++)

{

for (int j =0; j < cols; j++)

{

unsigned char temp;

temp = img\_src.at<unsigned char>(cv::Point(j, i));

vec\_input[i][j] = (double) temp;

}

}

////// create length vector, output vector... which store output result affter using DWT ///

*vector*<int> length;

*vector*<double> output, flag;

///// get parametter from WindowForm //////

*msclr*::*interop*::*marshal\_context* context;

*string* nm = context.*marshal\_as*<*std*::string>(cB\_wavname->*Text*);

int J = (int)nUD\_level->*Value*;

///// Doing DWT transform ////

dwt\_2d(vec\_input,J,nm,output,flag,length);

//// Compute wavelet subband image (for display)

*vector*<int> length\_2; // luu tru kich thuoc cua anh

dwt\_output\_dim\_sym(length,length\_2,J);

// length2 is gives the integer vector that contains the size of subimages that will

// combine to form the displayed output image. The last two entries of length2 gives the

// size of DWT ( rows\_n by cols\_n)

int siz = length\_2.*size*();

int rows\_n=length\_2[siz-2];

int cols\_n = length\_2[siz-1];

*vector*<*vector*< double> > dwtdisp(rows\_n, *vector*<double>(cols\_n));

dispDWT(output, dwtdisp, length ,length\_2, J);

*vector*<*vector*<double> > dwt\_output= dwtdisp;

double max;

///// max value is needed to take care of overflow

max\_array(dwt\_output,max);

Mat img\_dwt = Mat::zeros(rows\_n,cols\_n, img\_src.type());

// dwt\_hold is created to hold the dwt output as further operations need to be

// carried out on dwt\_output in order to display scaled images.

*vector*<*vector*<double> > dwt\_hold (rows\_n, *vector*<double>( cols\_n));

dwt\_hold = dwt\_output;

// Setting coefficients of created image to the scaled DWT output values

for (int i = 0; i < img\_dwt.rows; i++ )

{

for (int j = 0; j < img\_dwt.cols; j++ )

{

if ( dwt\_output[i][j] <= 0.0)

{

dwt\_output[i][j] = 0.0;

}

if ( i <= (length\_2[0]) && j <= (length\_2[1]) )

{

img\_dwt.at<unsigned char>(cv::Point(j, i)) = (char) ( (dwt\_output[i][j] / max) \* 255.0);

}

else

{

img\_dwt.at<unsigned char>(cv::Point(j, i))= (char) (dwt\_output[i][j]);

}

}

}

//////////// End of computing wavelet subband image (for display)

////////// \*\*\*\*\*\*\* Display each subband image \*\*\*\*\*\*\*\* /////////

*vector*<double> sub\_image;

sub\_image = output;

int rows\_1 = length[0];

int cols\_1 = length[1];

Mat LL = Mat(rows\_1, cols\_1, CV\_64F);

Mat HL = Mat(rows\_1, cols\_1, CV\_64F);

Mat LH = Mat(rows\_1, cols\_1, CV\_64F);

Mat HH = Mat(rows\_1, cols\_1, CV\_64F);

////// \*\*\*\*\*\*\*\*\* convert from 1D vector to 2D matrix \*\*\*\*\*\*\*\*\*\*

//////// Low-Low sub-band ///////

for (int k = 0; k < rows\_1\*cols\_1; k++)

{

int i = k/cols\_1; // thuong->dong

int j = k%cols\_1; // du->cot

LL.at<double>(cv::Point(j, i)) = sub\_image[k];

}

//////// High-Low sub-band ///////

for (int k = rows\_1\*cols\_1; k < 2\*rows\_1\*cols\_1; k++)

{

int k1 = k - rows\_1\*cols\_1;

int i = k1/cols\_1; // thuong->dong

int j = k1%cols\_1; // du->cot

HL.at<double>(cv::Point(j, i)) = sub\_image[k];

}

//////// Low-High sub-band ///////

for (int k = 2\*rows\_1\*cols\_1; k < 3\*rows\_1\*cols\_1; k++)

{

int k1 = k - 2\*rows\_1\*cols\_1;

int i = k1/cols\_1; // thuong->dong

int j = k1%cols\_1; // du->cot

LH.at<double>(cv::Point(j, i)) = sub\_image[k];

}

///////// High-high sub-band /////////

for (int k = 3\*rows\_1\*cols\_1; k < 4\*rows\_1\*cols\_1; k++)

{

int k1 = k - 3\*rows\_1\*cols\_1;

int i = k1/cols\_1; // thuong->dong

int j = k1%cols\_1; // du->cot

HH.at<double>(cv::Point(j, i)) = sub\_image[k];

}

////\*\*\*\*\*\*\*\*\*\*\*\*\*\* change parametter\*\*\*\*\*\*\*\*\*\*\*\*\*\* ////////////

for (int i = 0; i < LH.rows; i++ )

{

for (int j = 0; j < LH.cols; j++)

{

LH.at<double>(cv::Point(j,i)) = (LH.at<double>(cv::Point(j,i)))\*2;

}

}

///////\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Return value\*\*\*\*\*\*\*\*\*\*\*\*\* /////

for (int k = 2\*rows\_1\*cols\_1; k < 3\*rows\_1\*cols\_1; k++)

{

int k1 = k - 2\*rows\_1\*cols\_1;

int i = k1/cols\_1; // thuong->dong

int j = k1%cols\_1; // du->cot

sub\_image[k] = LH.at<double>(cv::Point(j, i));

}

//////////// Finding IDWT ///////////

*vector*<*vector*<double> > idwt\_output(rows, *vector*<double>(cols));

idwt\_2d\_sym(sub\_image,flag, nm, idwt\_output,length);

cv::Mat idwt\_img = Mat::zeros(idwt\_output.*size*(),idwt\_output[0].*size*(), img.type());

for (int i = 0; i < idwt\_img.rows; i++ )

{

for (int j = 0; j < idwt\_img.cols; j++ )

{

idwt\_img.at<unsigned char>(cv::Point(j, i)) = (char) (idwt\_output[i][j]);

}

}

LL.convertTo(LL, CV\_8UC1);

LH.convertTo(LH, CV\_8UC1);

HL.convertTo(HL, CV\_8UC1);

HH.convertTo(HH, CV\_8UC1);

1. **Tính gradientface của ảnh**

B1: ( Viết hàm này ở đầu cương trình)

B2: vào hàm main và tạo

Mat graface = gradientface(img);

//normalize(graface, graface, 0, 255, CV\_MINMAX);

//graface.convertTo(graface, CV\_8UC3);

int row\_s = (img.rows/4)\*4;

int col\_s = (img.cols/4)\*4;

Mat graface\_s = Mat(row\_s, col\_s, img.type());

resize(graface, graface\_s, graface\_s.size(), 0, 0, INTER\_LINEAR);

**Hàm tính gradient face**

static Mat gradientface(Mat OriImg)

{

int OriH,OriW;

OriH=OriImg.rows;

OriW=OriImg.cols;

Mat OriImg\_d = Mat(OriH, OriW, CV\_64F);

OriImg.convertTo(OriImg\_d, CV\_64F);

double\*\* OriR = (double\*\*)*malloc*( OriH\*sizeof(double\* )+OriW\*OriH\*sizeof(double));

double\* OriR\_j = (double\* )(OriR + OriH);

for(int y = 0; y!=OriH ; ++y, OriR\_j+=OriW )

OriR[y] = OriR\_j;

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

OriR[y][x]= (double)OriImg\_d.at<double>(cv::Point(x, y));

}

}

double min\_orir=999999;

double max\_orir=0;

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

if(OriR[y][x]<min\_orir) min\_orir=OriR[y][x];

if(OriR[y][x]>max\_orir) max\_orir=OriR[y][x];

}

}

//lB\_result->Items->Add("min values: " + min\_orir.ToString());

//lB\_result->Items->Add("max values: " + max\_orir.ToString());

double mean\_r=0;

double sigma\_r=0;

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

mean\_r=mean\_r+OriR[y][x];

}

}

mean\_r=mean\_r/(OriH\*OriW);

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

sigma\_r=sigma\_r+((OriR[y][x]-mean\_r)\*(OriR[y][x]-mean\_r));

}

}

sigma\_r=sqrt(sigma\_r/(OriH\*OriW));

double\*\* GauR = (double\*\*)*malloc*( OriH\*sizeof(double\* )+OriW\*OriH\*sizeof(double));

double\* GauR\_j = (double\* )(GauR + OriH);

for(int y = 0; y!=OriH ; ++y, GauR\_j+=OriW )

GauR[y] = GauR\_j;

double pi=3.14159;

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

GauR[y][x]=1/(2\*pi\*sigma\_r\*sigma\_r)\*pow(2.71828,((-1\*(OriR[y][x])-mean\_r)\*(OriR[y][x]-mean\_r))/(2\*sigma\_r\*sigma\_r));

}

}

// diplay gaussian image

double max\_Gaussf=(-999999);

double min\_Gaussf=999999;

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

if(GauR[y][x]>max\_Gaussf) max\_Gaussf=GauR[y][x];

if(GauR[y][x]<min\_Gaussf) min\_Gaussf=GauR[y][x];

}

}

Mat GaussfImg = Mat(OriH, OriW, CV\_8UC1);

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

GauR[y][x]=(GauR[y][x]-min\_Gaussf)/(max\_Gaussf-min\_Gaussf)\*255;

GaussfImg.at<unsigned char>(cv::Point(x, y)) = (unsigned char)GauR[y][x];

}

}

//imshow("Gauss", GaussfImg);

///

//------------Gradientfaces---------------

double\*\* graR = (double\*\*)*malloc*( OriH\*sizeof(double\* )+OriW\*OriH\*sizeof(double));

double\* graR\_j = (double\* )(graR + OriH);

for(int y = 0; y!=OriH ; ++y, graR\_j+=OriW )

graR[y] = graR\_j;

double gau\_r\_x;

double gau\_r\_y;

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

gau\_r\_x=0;

gau\_r\_y=0;

if(x!=0)

{

if(x!=OriW-1)

{

gau\_r\_x=((GauR[y][x-1]-GauR[y][x])+(GauR[y][x]-GauR[y][x+1]))/2;

}

else

{

gau\_r\_x=((GauR[y][x-1]-GauR[y][x])+(GauR[y][x]-0))/2;

}

}

else

{

gau\_r\_x=((0-GauR[y][x])+(GauR[y][x]-GauR[y][x+1]))/2;

}

if(y!=0)

{

if(y!=OriH-1)

{

gau\_r\_y=((GauR[y-1][x]-GauR[y][x])+(GauR[y][x]-GauR[y+1][x]))/2;

}

else

{

gau\_r\_y=((GauR[y-1][x]-GauR[y][x])+(GauR[y][x]-0))/2;

}

}

else

{

gau\_r\_y=((0-GauR[y][x])+(GauR[y][x]-GauR[y+1][x]))/2;

}

if(gau\_r\_x\*gau\_r\_y>0)

{

if(gau\_r\_x>0)

{

graR[y][x]=*atan*(gau\_r\_y/gau\_r\_x);

}

else

{

graR[y][x]=pi+*atan*(gau\_r\_y/gau\_r\_x);

}

}

if(gau\_r\_x\*gau\_r\_y<0)

{

if(gau\_r\_x>0)

{

graR[y][x]=(2\*pi)+*atan*(gau\_r\_y/gau\_r\_x);

}

else

{

graR[y][x]=pi+*atan*(gau\_r\_y/gau\_r\_x);

}

}

if(gau\_r\_x\*gau\_r\_y==0)

{

if(gau\_r\_x==0)

{

if(gau\_r\_y>=0)

{

graR[y][x]=pi/2;

}

else

{

graR[y][x]=(3\*pi)/2;

}

}

else

{

if(gau\_r\_x>0)

{

graR[y][x]=0;

}

else

{

graR[y][x]=pi;

}

}

}

graR[y][x]=(graR[y][x]\*180)/pi;

}

}

double max\_r=(-9999);

double min\_r=9999;

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

if(graR[y][x]>max\_r) max\_r=graR[y][x];

if(graR[y][x]<min\_r) min\_r=graR[y][x];

}

}

Mat graRImg = Mat(OriH, OriW, CV\_8UC1);

for(int y=0;y<OriH;y++)

{

for(int x=0;x<OriW;x++)

{

graR[y][x]=(graR[y][x]-min\_r)/(max\_r-min\_r)\*255;

graRImg.at<unsigned char>(cv::Point(x, y)) = (unsigned char)graR[y][x];

}

}

return graRImg;

}

////////////

1. Hiện giá trị mảng lên listbox

Tạo listbox từ Toollist

Add code sau vào chương trình

lB\_array->*Items*->*Add*("No."+ k.*ToString*()+": "+vec\_input[i][j]+" Pixels");

1. Hiện giá trị lên Textbox

txt\_xi\_LL\_B->*Text* = xi\_LL\_G.*ToString*();txt\_xi\_LL\_B->*Refresh*();

1. **Vẽ số thứ tự nhãn vào ROI contour**

Viết hàm sau lên đầu chương trình

void writeLabel\_text(cv::Mat& im, const *std*::string label, *std*::vector<cv::Point>& contour)

{

int fontface = cv::FONT\_HERSHEY\_SIMPLEX;

double scale = 0.4;

int thickness = 1;

int baseline = 0;

cv::Size text = cv::getTextSize(label, fontface, scale, thickness, &baseline);

cv::Rect r = cv::boundingRect(contour);

cv::Point pt(r.x + ((r.width - text.width) / 2), r.y + ((r.height + text.height) / 2));

cv::rectangle(im, pt + cv::Point(0, baseline), pt + cv::Point(text.width, -text.height), CV\_RGB(255,255,255), CV\_FILLED);

cv::putText(im, label, pt, fontface, scale, CV\_RGB(0,0,0), thickness, 8);

}

Viết hàm sau vào trong vòng lặp contour

for (int i = 0; i < contours.size(); i++)

{

///// find boundary of ROI

Rect label\_ROI = boundingRect(contours[i]);

/////// Draw boundary of label

*vector*<Vec4i> hierarchy;

drawContours( superpixel\_in\_img , contours, i , cv::Scalar(0,255,255), 1, 8, hierarchy, 0, cv::Point());

//// convert int to String (char)

char number[10];

*itoa* (label,number,10);

//// create text region

cv::Size text = cv::getTextSize(number, cv::FONT\_HERSHEY\_SIMPLEX, 0.4, 2, 0);

//// find center Point of ROI

cv::Point center\_of\_label(label\_ROI.x + ((label\_ROI.width - text.width) / 2), label\_ROI.y + ((label\_ROI.height + text.height) / 2));

//// Put text to ROI //////

cv::putText(superpixel\_in\_img, number, center\_of\_label, cv::FONT\_HERSHEY\_SIMPLEX, 0.4, CV\_RGB(255,0,0), 1, 8);

}

1. **Vẽ histogram projection (vertical, horizontal)**

**Cho đoạn code sau lên trên đầu chương trình**

static Mat mareHistogram(Mat& img, int t)

{

//col or row histogram?

int sz = (t)?img.rows:img.cols;

Mat mhist = Mat::zeros(1, sz, CV\_8U);

//count nonzero value and check max V

int max=-100;

for(int j=0; j < sz; ++j)

{

Mat data = (t)?img.row(j):img.col(j);

int v = countNonZero(data);

mhist.at< unsigned char >(j) = v;

if(v > max)

max=v;

}

Mat histo;

int width,height;

if(t)

{

width = max;

height = sz;

histo = Mat::zeros(cv::Size(width, height), CV\_8U);

for(int i=0; i < height; ++i)

{

for(int j=0; j < mhist.at< unsigned char >(i); ++j)

histo.at< unsigned char >(i,j) = 255;

}

}else{

width = sz;

height = max;

histo = Mat::zeros(cv::Size(width,height), CV\_8U);

for(int i=0; i< width; ++i)

{

for(int j=0; j< mhist.at< unsigned char >(i); ++j)

histo.at< unsigned char >(max-j-1,i) = 255;

}

}

return histo;

}

**Gọi hàm trongm hàm main**

Mat bin\_img1;

threshold(img1, bin\_img1, 180, 255, CV\_THRESH\_BINARY);

imshow("bin", bin\_img1);

Mat hHisto = mareHistogram(bin\_img1, 0);

imshow("", hHisto);

1. **Find circle on image (answer sheet)**

/////// Check answer form //////

threshold(gray, gray , 180, 255, CV\_THRESH\_BINARY);

Mat answer\_form = gray(cv::Range(120, 358) , cv::Range(50, 700));

*vector*<Vec3f> circles\_form;

HoughCircles(answer\_form.clone(), circles\_form, CV\_HOUGH\_GRADIENT, 2.5, 110, 70, 75, 30 ,50 );

int answer[6];

if (circles\_form.*size*() < 3)

{

for( *size\_t* i = 0; i < circles\_form.*size*(); i++ )

{

cv::Point center\_1(cvRound(circles\_form[i][0]), cvRound(circles\_form[i][1]));

int radius\_1 = cvRound(circles\_form[i][2]);

circle(answer\_form, center\_1, radius\_1+3, Scalar(0,0,255) , 2, 8,0);

cv::Point zero(answer\_form.rows/2 ,0);

Rect rect(center\_1.x-radius\_1, center\_1.y-radius\_1, radius\_1\*2,radius\_1\*2);

Mat submat = answer\_form(rect);

threshold(submat, submat, 100, 255, CV\_THRESH\_BINARY);

int non\_zero = countNonZero(submat);

int total = submat.rows\*submat.cols;

float p;

p = (float) non\_zero/ (float) total;

if (p ==1 || p < 0.4)

{

answer[i] = 1;

}

else if (p < 1 || p > 0.77)

{

answer[i] = 0;

}

}

}

1. **Separate or divide image to parts**

img1 = gray(cv::Range(1550, 1910) , cv::Range(0, 3800));

1. **Hiện cửa sổ Message Box**

if (txt\_savefolder->*Text*== "")

{

*MessageBox*::*Show*("Thông báo","Tên cửa sổ thong báo", *MessageBoxButtons*::*OK*,*MessageBoxIcon*::*Asterisk*);

return;

}

1. **Clear giá trị listbox, picturebox, textbox, button**

lB\_defects->*Items*->*Clear*();

pB\_orig->*Image* = nullptr; pB\_orig->*Refresh*();

btn\_clear->*Text* = "RESULT";

txt\_h->*Clear*(); txt\_w->*Clear*();txt\_hl->*Clear*();txt\_time->*Clear*();

1. **Vertical and Horizontal Projection**

Vertical

for (int i = 150; i < 500; i++){

Mat row\_gray = gray.row(i);

int k = gray.cols - cv::countNonZero(row\_gray);

}

Horizontal

cv::Mat1b img = cv::imread("img.png", CV\_LOAD\_IMAGE\_GRAYSCALE);

cv::Mat1i black\_pixels(img.cols, 1);

for(int x = 0; x < img.cols; x++){

cv::Mat col = img.col(x);

black\_pixels(x,0) = img.rows - cv::countNonZero(col);

}

1. **Check answer Form**

///////////// TESTING CHECK ANSWER //////////

*vector*<Vec3f> circles;

HoughCircles(question\_answer\_1.clone(), circles, CV\_HOUGH\_GRADIENT, 2.5, 300, 20, 70, 40 ,50 );

for( *size\_t* i = 0; i < circles.size(); i++ )

{

cv::Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));

int radius = cvRound(circles[i][2]) + 10;

circle(question\_answer\_1, center, radius, Scalar(0,0,255) , 2, 8,0);

//char number[10];

//itoa (i,number,10);

////////create text region

//cv::Size text = cv::getTextSize(number, cv::FONT\_HERSHEY\_SIMPLEX, 0.4, 2, 0);

/////////Put text to ROI //////

//cv::putText(question\_answer\_1, number, center, cv::FONT\_HERSHEY\_SIMPLEX, 1, CV\_RGB(255,0,0), 3, 8);

Rect rect(center.x-radius, center.y-radius, radius\*2,radius\*2);

Mat submat = question\_answer\_1(rect);

threshold(submat, submat, 180, 255, CV\_THRESH\_BINARY);

int non\_zero = countNonZero(submat);

int total = submat.rows\*submat.cols;

float p;

p = (float) non\_zero/ (float) total;

}

if (circles.size() != 6)

{

lB\_test->*Items*->*Add*("No." + step.*ToString*() + ": " + circles.size().*ToString*());

namedWindow("error", CV\_WINDOW\_NORMAL);

imshow("error", question\_answer\_1);

}

/////// TEST CHECK ANSWER ///////////

*vector*<Vec3f> circles;

HoughCircles(question\_answer\_1.clone(), circles, CV\_HOUGH\_GRADIENT, 2.5, 300, 20, 70, 40 ,50 );

int sort\_array[6];

for (*size\_t* i = 0; i < circles.size(); i++)

{

cv::Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));

cv::Point zero (0, 0);

////// distance of i circle

sort\_array[i] = euclideanDist(center, zero);

}

sort\_1D\_array(sort\_array,6);

for( *size\_t* i = 0; i < circles.size(); i++ )

{

for (int j = 0; j < 6; j++)

{

cv::Point center(cvRound(circles[i][0]), cvRound(circles[i][1]));

int radius = cvRound(circles[i][2]) + 10;

circle(question\_answer\_1, center, radius, Scalar(0,0,255) , 2, 8,0);

Rect rect(center.x-radius, center.y-radius, radius\*2,radius\*2);

Mat submat = question\_answer\_1(rect);

threshold(submat, submat, 200, 255, CV\_THRESH\_BINARY);

int non\_zero = countNonZero(submat);

int total = submat.rows\*submat.cols;

float p;

p = (float) non\_zero/ (float) total;

lB\_test->*Items*->*Add*("No." + i.*ToString*() + p.*ToString*());

cv::Point zero (0, 0);

if (p == 1 || p < 0.5)

{

int distance = euclideanDist(center, zero);

if (distance == sort\_array[j])

{

lB\_test->*Items*->*Add*("No.1 : " + j.*ToString*());

namedWindow("result" , CV\_WINDOW\_NORMAL);

imshow("result" , question\_answer\_1);

}

}

}

}

1. Tính toán thời gian chạy chương trình

Copy đoạn sau trên đầu chương trình, sau dòng “using name ….”

// Compute the Time

#define TIMER\_INIT *LARGE\_INTEGER* frequency;*LARGE\_INTEGER* t1,t2;double elapsedTime;*QueryPerformanceFrequency*(&frequency);

// Use to start the performance timer

#define TIMER\_START *QueryPerformanceCounter*(&t1);

// Use to stop the performance timer and output the result to the standard stream. Less verbose than \c TIMER\_STOP\_VERBOSE

#define TIMER\_STOP *QueryPerformanceCounter*(&t2);elapsedTime=(float)(t2.*QuadPart*-t1.*QuadPart*)/frequency.*QuadPart*;//std::wcout<<elapsedTime<<L" sec"<<endl;

//End of Compute the Time

Đặt đoạn code sau trước đoạn chương trình cần tính toán

/////// Begin time counting /////

TIMER\_INIT

TIMER\_START

Đặt đoạn code sau sau đoạn chương trình cần tính toán

TIMER\_STOP

//// finish time counting //////

Hiển thị thời gian tính toán lên textbox

///////// Display time counting ///////////

txt\_time->*Text* = ((float)elapsedTime).*ToString*()+" secs";

1. Chuyển đổi System::String to char\* hoặc char\*[]

char\* a4\_answer[4];

a4\_answer[0] = (char\*)*Marshal*::*StringToHGlobalAnsi*(txt\_a4\_ques\_1->*Text*).*ToPointer*();

for (int i = 0; i < 4; i++)

{

lB\_test->*Items*->*Add*(a4\_answer[i]->*ToString*());

}

1. **Tạo và ghi kết quả ra file csv**

**Tạo file CSV**

////////////////// Create CSV file ///////////////

*System*::String^ csvname;

csvname ="..\\csvfiles\\" + txt\_csvname->*Text*+".csv";

*marshal\_context* context2;

*LPCTSTR* cstr1 = context2.*marshal\_as*<const *TCHAR*\*>(csvname);

*fstream* dataFile;

dataFile.*open*(cstr1, *ios*::*out* | *ios*::*app*);

if(dataFile.*fail*())

{

dataFile.*open*(cstr1, *ios*::*out* | *ios*::*app*);

}

**Thứ tự ghi file csv**

dataFile << imgname << ";" << label << ";" << answer1 << ";" << answer2 << ";" << answer3 << ";" << answer4 << "\n";

**Openfiledialog để đọc tên file**

openFileDialog->*Filter* = "Image Files|\*.bmp; \*.jpg; \*.png|All Files (\*.\*)|\*.\*||"; openFileDialog->*Title* = "Select a image file";

openFileDialog->*InitialDirectory* = "";

openFileDialog-> *Multiselect* = "True";

int number\_of\_files;

array<*System*::String^>^ filename\_src; // array of filenames

if ( openFileDialog->*ShowDialog*() == *System*::*Windows*::*Forms*::*DialogResult*::*OK* )

{

filename\_src = openFileDialog->*FileNames*;

number\_of\_files = openFileDialog->*FileNames*->*Length*;

}

else return;

*System*::String^ OpenFileName;

*System*::String^ step\_str;

int step = 0;

while (step <number\_of\_files)

{

OpenFileName = filename\_src[step];

OpenFileName = OpenFileName->*Replace*("\\","\\\\");

char\* imgname = (char\*)*Marshal*::*StringToHGlobalAnsi*(OpenFileName).*ToPointer*(); // convert to kieu char lay ten filecho opencv

*System*::String^ answer1\_str = a4\_correct\_answer[0].*ToString*(); /////// Convert int value to System::String to work with Form

char\* answer1 = (char\*)*Marshal*::*StringToHGlobalAnsi*(answer1\_str).*ToPointer*();

step\_str = (step).*ToString*();

char\* label = (char\*)*Marshal*::*StringToHGlobalAnsi*(step\_str).*ToPointer*();

1. **Automatic image Processing on Cloud Driver or Dropbox**

**Bước 1:** Tạo bộ đếm thời gian “Timer” từ toolbox

**Bước 2:** Tạo phím khởi động (khi click, chương trình sẽ tự động chạy và xử lý đến khi bấm dừng)

Code tạo sự kiện cho Button

if (btn\_start->*Text* == "Start")

{

richtxt\_answerForm->*Clear*();

////// timer Start /////

timer->*Start*();

}

else

{

timer->*Stop*();

btn\_start->*Text* = "Start";

}

**Bước 3:** Click vào “timer”

Chuyển trạng thái của button sang “Stop”

btn\_start->*Text* = "Stop";

Tạo đường dẫn đến thư mục xử lý (Dropbox or Cloud Driver)

////////// Get Folder link ///////////

*System*::String^ path;

*System*::String^ pathsave;

if (cB\_folder->*Text* == L"My Folder")

{

path = "D:\\THANH\\MyImageFolder\\Original";

}

if (cB\_folder->*Text* == L"Dropbox")

{

path = "C:\\Users\\THANH\Dropbox\\MyImageFolder";

}

Tạo mảng chứa file trong thư mục, đếm số file trong thư mục

/////// Get name and file number of folder

//System::IO::Directory::Filter = "Image Files|\*.bmp; \*.jpg; \*.png; \*.pgm|All Files (\*.\*)|\*.\*||";

/////// Create array to store fileEntries

array<*System*::String^> ^fileEntries = *System*::*IO*::Directory::*GetFiles*(path);

int no\_files = fileEntries->*Length*;

**Bước 4:** Sắp xếp file trong Dropbox dựa trên thời gian copy file

File mới nhất có trong folder sẽ nằm ở phần tử đầu tiên của mảng FileEntries

for(int i=0; i<no\_files-1; i++)

{

for(int j=i+1; j<no\_files; j++)

{

if (

*System*::*IO*::Directory::*GetCreationTime*(fileEntries[i])>*System*::*IO*::Directory::*GetCreationTime*(fileEntries[j]))

{

*System*::String^ t = fileEntries[i];

fileEntries[i] = fileEntries[j];

fileEntries[j] = t;

}

}

}

**Bước 5:** Bắt đầu xử lý với file đầu tiên trong thư mục

if (no\_files == 0)

{

txt\_process->*Text* = "No Image";

txt\_process->*Refresh*();

}

else

{

txt\_process->*Text* = "Processing ...(" + no\_files.*ToString*() + ")";

txt\_process->*Refresh*();

/////// After sort files, the lastest file is fileEntries[0] ///////

*System*::String^ OpenFileName;

OpenFileName = fileEntries[0];

OpenFileName = OpenFileName->*Replace*("\\","\\\\");

char\* imgname;

imgname = (char\*)*Marshal*::*StringToHGlobalAnsi*(OpenFileName).*ToPointer*();

Mat img = cv::imread(imgname);

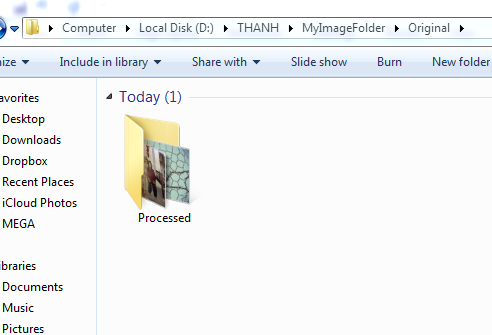
Mat gray = img.clone();

Bắt đầu các bước xử lý ảnh tiếp theo tại đây!

**Bước 6:** Sau khi xử lý ảnh, ảnh đã xử lý sẽ được chuyển sang 1 folder mới:

**“Processed” (Tạo folder này nằm trong đường dẫn folder chứa ảnh xử lý)**

*System*::String^ process\_folder = "\\Processed\\";



**Bước 7**: Sau khi chuyển sang folder mới, cần xử lý tiếp ảnh tiếp theo. Đầu tiên tạo 1 filename mới “filename2”

Chuyển đổi vị trí **“FileEntries[0]”** vừa xử lý thành vị trí **filename2**

**Code:** Đặt ở cuối **Timer\_click**

/////////// Transfer Processed file to other Folder (here is "Processed Folder" in main folder)

*System*::String^ fileName2 = fileEntries[0]->*Insert*(path->*Length*, process\_folder);

////////// Move position of processed File to next file //////////

if (*System*::*IO*::*File*::*Exists*(fileName2))

{

*System*::*IO*::*File*::*Delete*(fileName2);

}

*System*::*IO*::*File*::*Move*( fileEntries[0], fileName2 );

1. **Face Detection by on VideoCam or Video**

CascadeClassifier detection; // Khai bao class detection

detection.load("C:\\Users\\THANH\\Documents\\Visual Studio 2010\\Projects\\TestDevice\\haarcascade\_frontalface\_alt.xml");

if (!detection.load("C:\\Users\\THANH\\Documents\\Visual Studio 2010\\Projects\\TestDevice\\haarcascade\_frontalface\_alt.xml"))

{

*MessageBox*::*Show*("error","Warning", *MessageBoxButtons*::*OK*,*MessageBoxIcon*::*Asterisk*);

return;

}

VideoCapture capture = cv::VideoCapture(0);

if (!capture.isOpened())

{

*MessageBox*::*Show*("Can not load VideoCam","Warning", *MessageBoxButtons*::*OK*,*MessageBoxIcon*::*Asterisk*);

return;

}

Mat frame, gray;

*vector*<Rect> faces;

for(;;)

{

capture >> frame; // get a new frame from camera

cvtColor(frame, gray, CV\_BGR2GRAY);

equalizeHist(gray,gray);

detection.detectMultiScale(gray , faces , 1.4 , 3, 0|CV\_HAAR\_SCALE\_IMAGE , cv::Size(5, 5));

/\*detection.detectMultiScale(gray, faces, 1.1, 2, CV\_HAAR\_SCALE\_IMAGE | CV\_HAAR\_DO\_CANNY\_PRUNING, cv::Size(0,0), cv::Size(5,5));\*/

for(int i=0; i < faces.*size*();i++)

{

cv::Point pt1(faces[i].x+faces[i].width, faces[i].y+faces[i].height);

cv::Point pt2(faces[i].x,faces[i].y);

rectangle(frame, pt1, pt2, cv::Scalar(0,0,0), 2, 8, 0);

}

imshow("edges", frame);

waitKey(100);

}

1. **Build Tesseract libraries**

Choose Visual Studio project properties

------C/C++ -> General -> C:\......\tesseract\include

------Linker -> General -> C:\tesseract\lib

------Linker -> Input

liblept168.lib

libtesseract302.lib

copy liblept168.dll and libtesseract302.dll to Project folder

**Testing code**

Mat gray = img.clone();

cvtColor(gray, gray , CV\_BGR2GRAY);

// Ket thuc mo file

tesseract::TessBaseAPI \*myOCR = new tesseract::TessBaseAPI();

if (myOCR->Init(NULL, "eng")) {

*fprintf*(*stderr*, "Could not initialize tesseract.\n");

*exit*(1);

}

Rect text1ROI(80, 50, 800, 110);

Rect text2ROI(190, 200, 550, 50);

// recognize text

myOCR->TesseractRect( gray.data, 1, gray.step1(), text1ROI.x, text1ROI.y, text1ROI.width, text1ROI.height);

const char \*text1 = myOCR->GetUTF8Text();

myOCR->TesseractRect( gray.data, 1, gray.step1(), text2ROI.x, text2ROI.y, text2ROI.width, text2ROI.height);

const char \*text2 = myOCR->GetUTF8Text();

Rect textROI(0, 759, 0, 125);

myOCR->TesseractRect( gray.data, 1, gray.step1(), textROI.x, textROI.y, textROI.width, textROI.height);

const char \*text1 = myOCR->GetUTF8Text();

string t1(text1);

t1.*erase*(*std*::*remove*(t1.*begin*(), t1.*end*(), '\n'), t1.*end*());

string t2(text2);

t2.*erase*(*std*::*remove*(t2.*begin*(), t2.*end*(), '\n'), t2.*end*());

*System*::String^ result1 = gcnew *System*::String(t1.*c\_str*());

*Console*::*WriteLine*(result1);

/\*System::String^ result2 = gcnew System::String(t2.c\_str());

Console::WriteLine(result2);

\*/

lB\_result->*Items*->*Add*(result1);

1. **FolderBrowserDialog**

/////////// Open Folder diglog //////////////

*FolderBrowserDialog*^ folderBrowserDialog;

folderBrowserDialog = gcnew *System*::*Windows*::*Forms*::*FolderBrowserDialog*;

// Show the FolderBrowserDialog.

*System*::*Windows*::*Forms*::*DialogResult* result = folderBrowserDialog->*ShowDialog*();

*System*::String^ folderName;

if ( folderBrowserDialog->*ShowDialog*() == *System*::*Windows*::*Forms*::*DialogResult*::*OK* )

{

folderName = folderBrowserDialog->*SelectedPath*;

folderName = folderName->*Replace*("\\","\\\\");

}

else

return;

txt\_folderName->*Text* = folderName;

1. **OCR Nhận dạng chữ bằng xử lý ảnh**

**Cài đặt thư viện Tesseract**

Choose Visual Studio project properties

------C/C++ -> General -> C:\......\tesseract\include

------Linker -> General -> C:\tesseract\lib

------Linker -> Input

liblept168.lib

libtesseract302.lib

copy liblept168.dll and libtesseract302.dll to Project folder

**Code tham khảo:**

<https://github.com/xuwangyin/opencv-tesseract/blob/master/opencv-tesseract.cpp>

|  |
| --- |
| #include <baseapi.h> |
|  | // #include <allheaders.h> |
|  | #include <sys/time.h> |
|  | #include <opencv2/highgui/highgui.hpp> |
|  | #include <opencv2/core/core.hpp> |
|  | #include <opencv2/imgproc/imgproc.hpp> |
|  |  |
|  | using namespace cv; |
|  |  |
|  | int main(int argc, char\* argv[]) { |
|  |  |
|  | // initilize tesseract OCR engine |
|  | tesseract::TessBaseAPI \*myOCR = |
|  | new tesseract::TessBaseAPI(); |
|  |  |
|  | printf("Tesseract-ocr version: %s\n", |
|  | myOCR->Version()); |
|  | // printf("Leptonica version: %s\n", |
|  | // getLeptonicaVersion()); |
|  |  |
|  | myOCR->Init(NULL, "eng")); |
|  |  |
|  | tesseract::PageSegMode pagesegmode = static\_cast<tesseract::PageSegMode>(7); // treat the image as a single text line |
|  | myOCR->SetPageSegMode(pagesegmode); |
|  |  |
|  | // read iamge |
|  | namedWindow("tesseract-opencv", 0); |
|  | Mat image = imread("sample.png", 0); |
|  |  |
|  | // set region of interest (ROI), i.e. regions that contain text |
|  | Rect text1ROI(80, 50, 800, 110); |
|  | Rect text2ROI(190, 200, 550, 50); |
|  |  |
|  | // recognize text |
|  | myOCR->TesseractRect( image.data, 1, image.step1(), text1ROI.x, text1ROI.y, text1ROI.width, text1ROI.height); |
|  | const char \*text1 = myOCR->GetUTF8Text(); |
|  |  |
|  | myOCR->TesseractRect( image.data, 1, image.step1(), text2ROI.x, text2ROI.y, text2ROI.width, text2ROI.height); |
|  | const char \*text2 = myOCR->GetUTF8Text(); |
|  |  |
|  | // remove "newline" |
|  | string t1(text1); |
|  | t1.erase(std::remove(t1.begin(), t1.end(), '\n'), t1.end()); |
|  |  |
|  | string t2(text2); |
|  | t2.erase(std::remove(t2.begin(), t2.end(), '\n'), t2.end()); |
|  |  |
|  | // print found text |
|  | printf("found text1: \n"); |
|  | printf(t1.c\_str()); |
|  | printf("\n"); |
|  |  |
|  | printf("found text2: \n"); |
|  | printf(t2.c\_str()); |
|  | printf("\n"); |
|  |  |
|  | // draw text on original image |
|  | Mat scratch = imread("sample.png"); |
|  |  |
|  | int fontFace = FONT\_HERSHEY\_PLAIN; |
|  | double fontScale = 2; |
|  | int thickness = 2; |
|  | putText(scratch, t1, Point(text1ROI.x, text1ROI.y), fontFace, fontScale, Scalar(0, 255, 0), thickness, 8); |
|  | putText(scratch, t2, Point(text2ROI.x, text2ROI.y), fontFace, fontScale, Scalar(0, 255, 0), thickness, 8); |
|  |  |
|  | rectangle(scratch, text1ROI, Scalar(0, 0, 255), 2, 8, 0); |
|  | rectangle(scratch, text2ROI, Scalar(0, 0, 255), 2, 8, 0); |
|  |  |
|  | imshow("tesseract-opencv", scratch); |
|  | waitKey(0); |
|  |  |
|  | delete [] text1; |
|  | delete [] text2; |
|  |  |
|  | // destroy tesseract OCR engine |
|  | myOCR->Clear(); |
|  | myOCR->End(); |
|  |  |
|  | return 0; |
|  | } |

1. **Rotate image**

//////// Rotate angle /////////

double angle = 180.0;

//////// scale ///////////

double scale = 1.0;

/////// center of rotation //////

cv::Point2f center(gray.cols/2 , gray.rows/2);

/////// Rotate function ///////

Mat gray\_rot = getRotationMatrix2D(center, angle, scale);

Mat clone = gray.clone();

warpAffine(gray, clone, gray\_rot, gray.size());

imshow("", clone);

1. **Extract superpixel Label (SLIC) and apply change**
2. **Install**

Firstly, add **SLIC.h** to header, and **SLIC function.cpp** to Source file

Add **#include <slic.h>**  to the top of **SLIC function.cpp**

Add **#include <slic.h>**  to the top of **Form.h**

<http://ivrl.epfl.ch/supplementary_material/RK_SLICSuperpixels/index.html>

**SLIC.h and SLIC function.cpp**

D:\THANH\MASTER\LABORATORY\PROGRAM\CODE\COMPLETE\SUPERPIXELS

1. **Use Code**

**Apply SLIC superpixel to image:**

Mat gray = img.clone();

cvtColor(img, gray , CV\_BGR2GRAY);

// Ket thuc mo file

Mat img\_input = img.clone();

// Ket thuc mo file

///// Create number of superpixels (number of label)

UINT numSuperpixel = 20;

SLIC slic\_img;

slic\_img.GenerateSuperpixels(gray, numSuperpixel );

////// Get image with contour after superpixels

Mat slic\_output = slic\_img.GetImgWithContours(cv::Scalar(0, 0, 255));

// Get the labels

Mat1i labelImg(gray.rows, gray.cols, slic\_img.GetLabel());

// Get the actual number of labels

// may be less that n\_of\_superpixels

double max\_dlabel;

minMaxLoc(labelImg, *NULL*, &max\_dlabel);

int max\_label = int(max\_dlabel);

**Extract to super pixel label and apply changes**

for (int label = 0; label <= max\_label ; ++label)

{

Mat label\_image;

label\_image = labelImg == label; //////// mask of region label

//////// superpixel label on original image demension ///////

Mat superpixel\_in\_img;

img\_input.copyTo(superpixel\_in\_img, label\_image);

//////// crop each superpixel label with label dimension /////////

Rect r;

*std*::vector< *std*::vector<cv::Point> > contours;

*std*::vector<cv::Vec4i>hierarcy;

findContours(label\_image,contours,CV\_RETR\_CCOMP,CV\_CHAIN\_APPROX\_SIMPLE);

for( int i = 0; i< contours.size(); i++ )

{

drawContours(label\_image, contours, i, Scalar (255), CV\_FILLED);

r = boundingRect(contours[i]);

}

/////////// create ROI of label region ////////////

Mat region\_of\_label(superpixel\_in\_img(r));

////////// apply changes to label region //////////

Mat roiGray;

cvtColor(region\_of\_label, roiGray, COLOR\_BGR2GRAY);

GaussianBlur(roiGray,roiGray, cv::Size(3,3), 2);

Mat roiGray3b;

cvtColor(roiGray, roiGray3b, COLOR\_GRAY2BGR);

roiGray3b.copyTo(region\_of\_label);

////////////////// Mask for each label ////////////////

//Mat superpixel\_mask; //// create mask of label region

//superpixel\_mask = region\_of\_label == label; //// take label to mask image

///////////////// After processing, copy processed labels to original image

superpixel\_in\_img.copyTo(img\_input, label\_image);

}

**Reference:**

1. <http://stackoverflow.com/questions/32822856/how-to-access-the-segments-of-the-image>
2. <https://github.com/davidstutz/vlfeat-slic-example/blob/master/vlfeat_slic_cli/main.cpp>
3. **Write and Read “.txt” files**

Copy following code to button area, **before** Openfiledialog code:

if (txt\_train->*Text*== "")

{

*MessageBox*::*Show*("Please put the name of txt file","ERROR", *MessageBoxButtons*::*OK*,*MessageBoxIcon*::*Asterisk*);

return;

}

*System*::String ^ output\_name = "..\\txtfiles\\" + txt\_train->*Text* + ".txt";

*fstream* dataFile;

// convert from 'System::String ^' to 'std::string'

*msclr*::*interop*::*marshal\_context* context;

*std*::string output\_name1 = context.*marshal\_as*<*std*::string>(output\_name);

dataFile.*open*(output\_name1, *ios*::*out* | *ios*::*app*);

if(dataFile.*fail*())

{

dataFile.*open*(output\_name1, *ios*::*out* | *ios*::*app*);

}

txt\_train->*Clear*();

txt\_train->*Text* = output\_name;

Write to .txt files;

/////////////// write values to txt file /////////////////

dataFile<< mean\_wBGR\_1 <<";" << mean\_wBGR\_2 << "\n ";

Read .txt files;

//////////////////// read CSV file //////////////////

*msclr*::*interop*::*marshal\_context* context;

*string* fn\_text = context.*marshal\_as*<*std*::string>(txt\_train->*Text*);

*string* line , label, mean\_wBGR\_1, mean\_wBGR\_2;

char separator = ';';

*std*::*ifstream* file(fn\_text.*c\_str*(), *ifstream*::*in*);

while (*getline*(file, line))

{

*stringstream* liness(line);

while (*getline*(liness, label, separator))

{

*getline*(liness, mean\_wBGR\_1, separator);

*getline*(liness, mean\_wBGR\_2);

*System*::String^ mean\_w1 = gcnew *System*::String(mean\_wBGR\_1.*c\_str*());

*Console*::*WriteLine*(mean\_w1);

*System*::String^ mean\_w2 = gcnew *System*::String(mean\_wBGR\_2.*c\_str*());

*Console*::*WriteLine*(mean\_w2);

lB\_results->*Items*->*Add*(mean\_w1 + " " + mean\_w2);

}

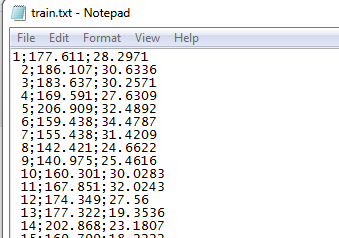
}

1. **Std::String to std::int or std:: float**

int i = *atoi*(label.*c\_str*());

1. **Read text files and write values to array**

For example, we have text file which contains value like this:



We will read this file and write values to array:

**Code**

float trainingData[100][2]; /////////// this array which we will write values to it //////////

//////////////////// read CSV file //////////////////

*msclr*::*interop*::*marshal\_context* context;

*string* fn\_text = context.*marshal\_as*<*std*::string>(txt\_train->*Text*);

*string* line , label, mean\_wBGR\_1, mean\_wBGR\_2;

char separator = ';';

*std*::*ifstream* file(fn\_text.*c\_str*(), *ifstream*::in);

while (*getline*(file, line))

{

*stringstream* liness(line);

while (*getline*(liness, label, separator))

{

int i = *atoi*(label.*c\_str*());

*getline*(liness, mean\_wBGR\_1, separator);

*getline*(liness, mean\_wBGR\_2);

*System*::String^ mean\_w1 = gcnew *System*::String(mean\_wBGR\_1.*c\_str*());

*Console*::*WriteLine*(mean\_w1);

*System*::String^ mean\_w2 = gcnew *System*::String(mean\_wBGR\_2.*c\_str*());

*Console*::*WriteLine*(mean\_w2);

*System*::String^ label\_str= gcnew *System*::String(label.*c\_str*());

*Console*::*WriteLine*(label\_str);

lB\_results->*Items*->*Add*(label\_str + ":"+ mean\_w1 + " " + mean\_w2);

trainingData[i-1][0] = *atof*(mean\_wBGR\_1.*c\_str*());

trainingData[i-1][1] = *atof*(mean\_wBGR\_2.*c\_str*());

}

}

for (int i = 0; i < 100; i++)

{

lB\_test->*Items*->*Add*(i.*ToString*() + ":" + trainingData[i][0].*ToString*() + " " + trainingData[i][1].*ToString*());

}

1. **Convert std::string to std:: float**

trainingData[i-1][0] = *atof*(mean\_wBGR\_1.*c\_str*());

1. **Get sub-directory name from openfiledialog**

path = *System*::*IO*::*Path*::*GetDirectoryName*(openFileDialog->*FileName*);

path = path->*Replace*("\\","\\\\");

1. **Check folder exit and move file to other sub-directory**

if (*System*::*IO*::*File*::*Exists*(fileName\_error))

{

*System*::*IO*::*File*::*Delete*(fileName\_error);

}

*System*::*IO*::*File*::*Move*( filename\_src[step], fileName\_error );

1. **Extract filename from path**

/////// After sort files, the lastest file is fileEntries[0] ///////

*System*::String^ OpenFileName;

OpenFileName = fileEntries[0];

OpenFileName = OpenFileName->*Replace*("\\","\\\\");

*msclr*::*interop*::*marshal\_context* context;

*std*::string str\_imgname = context.*marshal\_as*<*std*::string>(OpenFileName);

*std*::string res = str\_imgname.*substr*( str\_imgname.*find\_last\_of*("\\") + 1 );

**Example**

int main ()

{

std::string str="We think in generalities, but we live in details.";

// (quoting Alfred N. Whitehead)

std::string str2 = str.substr (3,5); // "think"

std::size\_t pos = str.find("live"); // position of "live" in str

std::string str3 = str.substr (pos); // get from "live" to the end

std::cout << str2 << ' ' << str3 << '\n';

return 0;

}