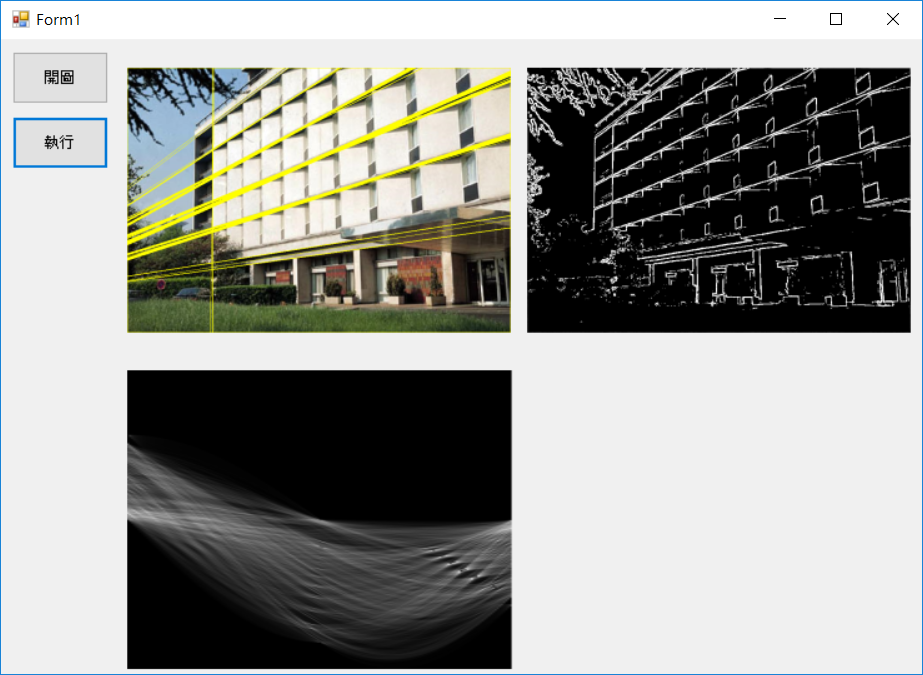
姓名：林佑恩 班級：四子三丙  
學號：1103105336老師：陳昭和老師

1. Please make a program for Hough transform using the following algorithm. In the output, the line segments found by Hough transform must be drawn on the original image using the yellow bold line. (Please use C++ based programming language, can’t use Matlab and LabVIEW )

結果：



程式碼：

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing.Imaging;

using System.Windows.Forms.DataVisualization.Charting;

using \_20170412\_work04\_HoughTransform.Properties;

namespace \_20170412\_work04\_HoughTransform

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Bitmap ori;

private const double Deg2Rad = Math.PI / 90.0;

private void button1\_Click(object sender, EventArgs e)

{

OpenFileDialog ofd = new OpenFileDialog();

ofd.Filter = "Image Files(\*.BMP;\*.JPG;\*.GIF;\*.PNG)|\*.BMP;\*.JPG;\*.GIF;\*.PNG";

if (ofd.ShowDialog() == DialogResult.OK)

{

pictureBox1.Image = Image.FromFile(ofd.FileName);

}

ori = pictureBox1.Image as Bitmap;

//gray = pictureBox1.Image as Bitmap;

pictureBox1.Image = ori;

}

private void button2\_Click(object sender, EventArgs e)

{

var imageRect = new Rectangle(0, 0, ori.Width, ori.Height); // Image rectangle.

var newBitmap = new Bitmap(imageRect.Width, imageRect.Height);// New bitmap for the image with sobel

var gray = new Bitmap(imageRect.Width, imageRect.Height);

var ori\_data = ori.LockBits(imageRect, ImageLockMode.ReadWrite, ori.PixelFormat);

var gray\_data = gray.LockBits(imageRect, ImageLockMode.ReadWrite, gray.PixelFormat);

var newBitmapData = newBitmap.LockBits(imageRect, ImageLockMode.ReadWrite, ori.PixelFormat);

var byteCount = ori\_data.Stride \* ori\_data.Height;// Stride is the amount of bytes in a row

int diagonal = (int)(Math.Sqrt(ori.Width \* ori.Width + ori.Height \* ori.Height)) + 1; //-√2\*D 到 √2\*D

var acc\_Rect = new Rectangle(0, 0, 180, 2 \* diagonal);

var accumulator = new Bitmap(180, 2 \* diagonal);

var dataAccumulator = accumulator.LockBits(acc\_Rect, ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);

unsafe

{

var ori\_bmp = (byte\*)ori\_data.Scan0;

var gray\_bmp = (byte\*)gray\_data.Scan0;

var newb\_bmp = (byte\*)newBitmapData.Scan0;

var hough\_bmp = (byte\*)dataAccumulator.Scan0;

GrayLevel(ori.Height, ori.Width, ori\_data.Stride, ori\_bmp, gray\_bmp);

SobelOperator(ori.Height, ori.Width, ori\_data.Stride, newb\_bmp, gray\_bmp);

int h = ori.Height;

int w = ori.Width;

var sin\_tab = new double[181];

var cos\_tab = new double[181];

for (int angle = 0; angle < 181; angle++) //-90 到 90

{

var theta = (angle - 90) \* Math.PI / 180.0;

cos\_tab[angle] = Math.Cos(theta);

sin\_tab[angle] = Math.Sin(theta);

}

var hough\_accumulation = new int[2 \* diagonal, 181]; //累加器

var max = 0;

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

{

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

{

int p\_Index = i \* ori\_data.Stride + j \* 3;

if (newb\_bmp[p\_Index] > 25)

{

for (int angle = 0; angle < 181; angle++)// -90 到 90

{

var r = i \* sin\_tab[angle] + j \* cos\_tab[angle];

var nr = (int)(r + diagonal); //shift r to positive value

hough\_accumulation[nr, angle]++; // accumulation

if (max < hough\_accumulation[nr, angle])

{

max = hough\_accumulation[nr, angle];

}

}

}

}

}

double alpha = 255.0 / max;

int size = (int)(max \* alpha);

for (int i = 0; i < accumulator.Height; i++)

{

for (int j = 0; j < accumulator.Width; j++)

{

var a = i \* dataAccumulator.Stride + j \* 3;

int tmp = (int)(hough\_accumulation[i, j]);// \* alpha);

//if (tmp != 0) tmp += 50;

hough\_bmp[a] = (byte)(tmp);

hough\_bmp[a + 1] = (byte)(tmp);

hough\_bmp[a + 2] = (byte)(tmp);

}

}

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

{

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

{

int p\_Index = i \* ori\_data.Stride + j \* 3;

for (int angle = 0; angle < 181; angle++)

{

var r = i \* sin\_tab[angle] + j \* cos\_tab[angle];

var nr = (int)(r + diagonal);

double tmp = hough\_accumulation[nr, angle]\* alpha;

if (tmp > 85.5)

{

ori\_bmp[p\_Index] = 0; //B

ori\_bmp[p\_Index + 1] = 255; //G

ori\_bmp[p\_Index + 2] = 255; //R

}

}

}

}

accumulator.UnlockBits(dataAccumulator);

}//unsafe

ori.UnlockBits(ori\_data);

gray.UnlockBits(gray\_data);

newBitmap.UnlockBits(newBitmapData);

pictureBox1.Image = ori ;

pictureBox2.Image = newBitmap ;

pictureBox3.Image = accumulator;

}

public unsafe void GrayLevel(int h, int w, int stride, byte\* ori\_bmp, byte\* gray\_bmp)

{

for (int i = 0; i < h; i++) //轉成gray level

{

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

{

int p\_Index = i \* stride + j \* 3;

byte pixel = Convert.ToByte((ori\_bmp[p\_Index + 0] + ori\_bmp[p\_Index + 1] + ori\_bmp[p\_Index + 2]) / 3);

gray\_bmp[p\_Index + 0] = pixel; //R

gray\_bmp[p\_Index + 1] = pixel; //G

gray\_bmp[p\_Index + 2] = pixel; //B

}

}

}

public unsafe void SobelOperator(int h, int w, int stride, byte\* newb\_bmp, byte\* ori\_bmp)

{

var Gx = new sbyte[9] { -1, -2, -1, 0, 0, 0, 1, 2, 1 };

var Gy = new sbyte[9] { -1, 0, 1, -2, 0, 2, -1, 0, 1 };

int tmp = 3 / 2;

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

{

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

{

int p\_index = i \* stride + j \* 3;

if (i < tmp || i >= (h - tmp) || j < tmp || j >= (w - tmp))

{

for (int k = 0; k < 3; k++)

{

int a = i \* stride + j \* 3 + k;

newb\_bmp[a] = ori\_bmp[a];

}

continue;

}

int x = 0;

var sum = new int[3] { 0, 0, 0 };

for (int ii = i - tmp; ii < (i - tmp) + 3; ii++)

{

for (int jj = j - tmp; jj < (j - tmp) + 3; jj++)

{

int p\_in = ii \* stride + jj \* 3;

sum[0] += Gx[x] \* ori\_bmp[p\_in];

sum[1] += Gy[x] \* ori\_bmp[p\_in];

x++;

}

}

int f = Math.Abs(sum[0]) + Math.Abs(sum[1]);

if (f > 255)

f = 255;

else f = 0;

newb\_bmp[p\_index] = Convert.ToByte(f);

newb\_bmp[p\_index + 1] = Convert.ToByte(f);

newb\_bmp[p\_index + 2] = Convert.ToByte(f);

}

}// for (int i = 0; i < ori.Height; i++)

}

}

}