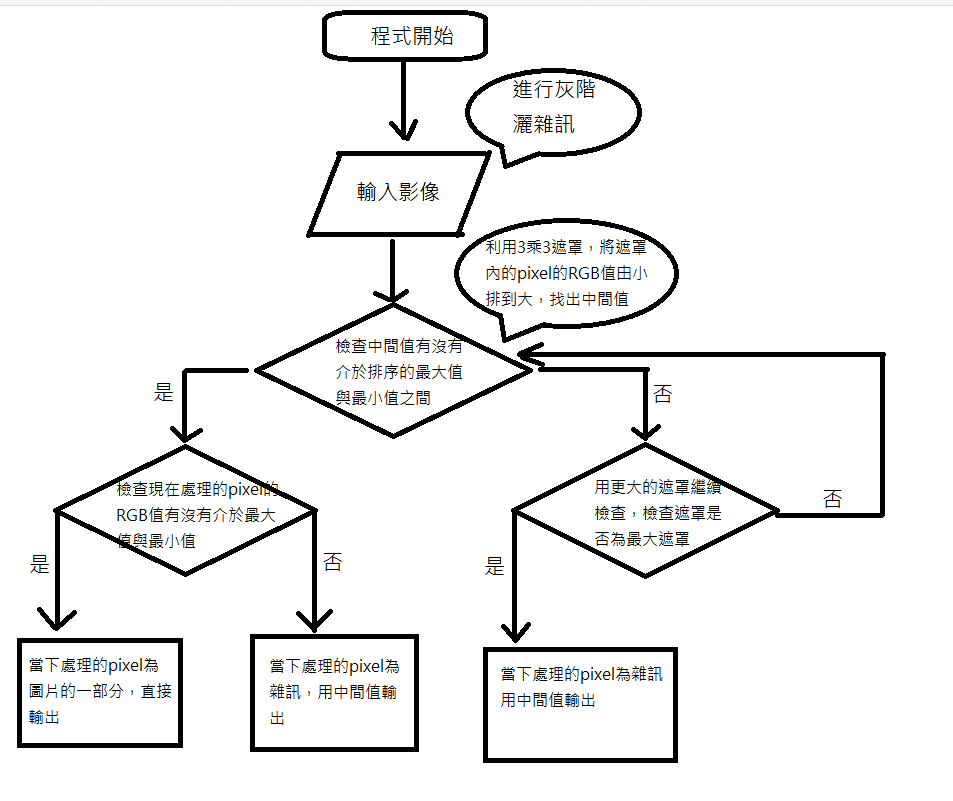
7108056022 陳冠霖

流程圖:



第一步:

讀入彩色影像後，將影像轉為灰階，灰階為每個pixel的RGB三通道為一樣的值，接著利用random的方式灑上胡椒鹽雜訊，雜訊為0或255，各占一半。

程式碼:

/\*按下button1將影像彩色影像，進行灰階和灑雜訊，最後放進來pictureBox1裡\*/

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

{

/\*載入圖片\*/

var dialog = new OpenFileDialog();

dialog.Filter = "影像(\*.jpg/\*.png/\*.gif/\*.bmp)|\*.jpg;\*.png;\*.gif;\*.bmp";

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

{

var filename = dialog.FileName;

IntPtr image = CvInvoke.cvLoadImage(filename, Emgu.CV.CvEnum.LOAD\_IMAGE\_TYPE.CV\_LOAD\_IMAGE\_ANYCOLOR);

size1 = CvInvoke.cvGetSize(image);

dest1 = new Image<Bgr, byte>(size1);

CvInvoke.cvCopy(image, dest1, IntPtr.Zero);

pictureBox1.Image = dest1.ToBitmap();

}

/\*轉灰階\*/

int w1 = pictureBox1.Image.Width;

int h1 = pictureBox1.Image.Height;

Bitmap bm1 = (Bitmap)pictureBox1.Image;

for (int i = 0; i < w1; i++)

{

for (int j = 0; j < h1; j++)

{

Color c1 = bm1.GetPixel(i, j);

int r1 = c1.R;

int g1 = c1.G;

int b1 = c1.B;

int avg1 = (r1 + g1 + b1) / 3;

bm1.SetPixel(i, j, Color.FromArgb(avg1, avg1, avg1));

}

}

/\*灑胡椒鹽雜訊\*/

Bitmap bm2 = new Bitmap(pictureBox1.Image);

/\*雜訊要站照片的多少%\*/

double noisep = 0.2;

/\*有多少雜訊點\*/

int noise;

noise = (int)(w1 \* h1 \* noisep);

/\*隨機變數\*/

Random x = new Random();

/\*用來放隨機變數\*/

int a = 0, b = 0;

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

{

/\*由於遮罩最多到7\*7，所以將邊界不處理，所以範圍為:3-(寬-3)、3-(長-3)\*/

a = x.Next(3, (w1-3));

b = x.Next(3, (h1-3));

/\*胡椒鹽雜訊的白點跟黑點各占一半\*/

if (i < (noise / 2))

{

bm2.SetPixel(a, b, Color.FromArgb(255, 255, 255));

}

else

{

bm2.SetPixel(a, b, Color.FromArgb(0, 0, 0));

}

}

pictureBox1.Image = bm2;

}

第二步:

使用中值濾波器來檢查整張影像中的pixel是雜訊還是影像中的一部分，首先利用三乘三遮罩，將當下處理的pixel與剩下的8個pixel，進行排序由小到大，如果中間值介於最大值與最小值之間且當下處理的pixel也介於最大值與最小值之間，代表當下處理的pixel為影像細節的一部分；當下處理的pixel沒有介於最大值與最小值之間，視為雜訊，雜訊的話就用中間值來替代原本的RGB值，如果一開始中間值沒有介於最大值與最小值之間，則繼續用更大的遮罩檢查，有可能為影像的一部分，如五乘五、七乘七，如果中間值介於最大值與最小值之間且當下處理的pixel也介於最大值與最小值之間，代表當下處理的pixel為影像細節的一部分；當下處理的pixel沒有介於最大值與最小值之間，視為雜訊，雜訊的話就用中間值來替代原本的RGB值，最後如果到了檢查用的最大遮罩，中間值還是沒有介於最大值與最小值之間，直接視為雜訊，用中間值代替原本的RGB值。

程式碼:

/\*中值濾波器\*/

/\*按下button2，用遮罩來檢查灰階影像上的pixel是影像的一部分或是雜訊\*/

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

{

/\*點陣圖(Bitmap)抓去Picturebox1的影像\*/

Bitmap bm1 = new Bitmap(pictureBox1.Image);

int width1 = pictureBox2.Size.Width;

int height1 = pictureBox2.Size.Height;

/\*Mask(3\*3遮罩)、Mask1(5\*5遮罩)、Mask2(7\*7遮罩)，用越來越大的遮罩檢查影像上的pixel\*/

int[] Mask = new int[9];

int[] Mask1 = new int[25];

int[] Mask2 = new int[49];

/\*Swap暫存空間\*/

int temp = 0;

/\*依照遮大大小count對3\*3、count1對5\*5、count2對7\*7的位置\*/

int count = 0,count1 = 0,count2 = 0;

/\*最小值(min)、最大值(max)、中間值(med)\*/

int min = 0, max = 0, med = 0;

/\*正在處理的pixel\*/

int now = 0;

for (int i = 0; i < width1; i++)

{

for (int j = 0; j < height1; j++)

{

/\*3\*3遮罩的邊界不處理\*/

if ((i != 0) & (i != width1 - 1) & (j != 0) & (j != height1 - 1))

{

count = 0;

/\*3\*3遮罩\*/

for (int x = -1; x <= 1; x++)

{

for (int y = -1; y <= 1; y++)

{

/\*抓取遮罩位置的RGB值\*/

Color c3 = bm1.GetPixel(i + x, j + y);

int rgb = c3.R;

/\*灰階只有一個通道，rgb丟到陣列\*/

Mask[count] = rgb;

if (count < 8)

{

count++;

}

else

{

break;

}

/\*存取現在處理的pixel\*/

if (x == 0 && y == 0)

{

now = rgb;

}

}

}

/\*將陣列裡的遮罩RGB值進行排列\*/

for (int k = 1; k < 9; k++)

{

for (int m = 0; m < 8; m++)

{

if (Mask[m] > Mask[m + 1])

{

temp = Mask[m];

Mask[m] = Mask[m + 1];

Mask[m + 1] = temp;

}

}

}

/\*最大值排序後的陣列最後一個\*/

max = Mask[8];

/\*最小值排序後的陣列第一個\*/

min = Mask[0];

/\*中間值排序後的陣列中間\*/

med = Mask[4];

/\*中間值有介於最大值與最小值之間\*/

if (med > min && med < max)

{

/\*檢查縣在處理的pixel有沒有介於最大值與最小值之間\*/

if (now > min && now < max)

{

/\*有的話，將當下的pixel存取下來，為圖片的一部分\*/

bm1.SetPixel(i, j, Color.FromArgb(now, now, now));

}

else

{

/\*沒有的話，為雜訊，用中間值代替\*/

bm1.SetPixel(i, j, Color.FromArgb(med, med, med));

}

}

/\*中間值沒有介於最大值與最小值之間\*/

else

{

/\*5\*5遮罩的邊界不處理\*/

if ((i >= 2) && (i < width1 - 2) && (j >= 2) && (j < height1 - 2))

{

count1 = 0;

/\*5\*5遮罩\*/

for (int x = -2; x <= 2; x++)

{

for (int y = -2; y <= 2; y++)

{

/\*抓取遮罩位置的RGB值\*/

Color c5 = bm1.GetPixel(i + x, j + y);

int rgb = c5.R;

/\*灰階只有一個通道，rgb丟到陣列\*/

Mask1[count1] = rgb;

if (count1 < 24)

{

count1++;

}

else

{

break;

}

/\*存取現在處理的pixel\*/

if (x == 0 && y == 0)

{

now = rgb;

}

}

}

/\*將陣列裡的遮罩RGB值進行排列\*/

for (int k = 1; k < 25; k++)

{

for (int m = 0; m < 24; m++)

{

if (Mask1[m] > Mask1[m + 1])

{

temp = Mask1[m];

Mask1[m] = Mask1[m + 1];

Mask1[m + 1] = temp;

}

}

}

/\*最大值排序後的陣列最後一個\*/

max = Mask1[24];

/\*最小值排序後的陣列第一個\*/

min = Mask1[0];

/\*中間值排序後的陣列中間\*/

med = Mask1[12];

/\*中間值有介於最大值與最小值之間\*/

if (med > min && med < max)

{

/\*檢查縣在處理的pixel有沒有介於最大值與最小值之間\*/

if (now > min && now < max)

{

/\*有的話，將當下的pixel存取下來，為圖片的一部分\*/

bm1.SetPixel(i, j, Color.FromArgb(now, now, now));

}

else

{

/\*沒有的話，為雜訊，用中間值代替\*/

bm1.SetPixel(i, j, Color.FromArgb(med, med, med));

}

}

/\*中間值沒有介於最大值與最小值之間\*/

else

{

/\*7\*7遮罩的邊界不處理\*/

if ((i >= 3) && (i < width1 - 3) && (j >= 3) && (j < height1 - 3))

{

count2 = 0;

/\*7\*7遮罩\*/

for (int x = -3; x <= 3; x++)

{

for (int y = -3; y <= 3; y++)

{

/\*抓取遮罩位置的RGB值\*/

Color c7 = bm1.GetPixel(i + x, j + y);

int rgb = c7.R;

/\*灰階只有一個通道，rgb丟到陣列\*/

Mask2[count2] = rgb;

if (count2 < 48)

{

count2++;

}

else

{

break;

}

/\*存取現在處理的pixel\*/

if (x == 0 && y == 0)

{

now = rgb;

}

}

}

/\*將陣列裡的遮罩RGB值進行排列\*/

for (int k = 1; k < 49; k++)

{

for (int m = 0; m < 48; m++)

{

if (Mask2[m] > Mask2[m + 1])

{

temp = Mask2[m];

Mask2[m] = Mask2[m + 1];

Mask2[m + 1] = temp;

}

}

}

/\*最大值排序後的陣列最後一個\*/

max = Mask2[48];

/\*最小值排序後的陣列第一個\*/

min = Mask2[0];

/\*中間值排序後的陣列中間\*/

med = Mask2[24];

/\*中間值有介於最大值與最小值之間\*/

if (med > min && med < max)

{

/\*檢查縣在處理的pixel有沒有介於最大值與最小值之間\*/

if (now > min && now < max)

{

/\*有的話，將當下的pixel存取下來，為圖片的一部分\*/

bm1.SetPixel(i, j, Color.FromArgb(now, now, now));

}

else

{

/\*沒有的話，為雜訊，用中間值代替\*/

bm1.SetPixel(i, j, Color.FromArgb(med, med, med));

}

}

/\*中間值沒有介於最大值與最小值之間\*/

else

{

/\*沒有的話，為雜訊，用中間值代替\*/

bm1.SetPixel(i, j, Color.FromArgb(med, med, med));

}

}

}

}

}

}

}

}

/\*用剛剛記錄下的每個位置pixel的RGB值放到pictureBox2\*/

pictureBox2.Image = bm1;

}

結果圖:

