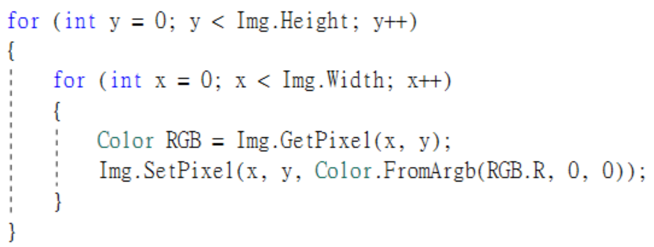
Report

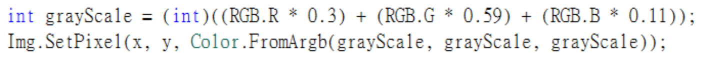
Problems and Methods

1. RGB Extraction & transformation



```
for (int y = 0; y < Img.Height; y++)
{
    for (int x = 0; x < Img.Width; x++)
    {
        Color RGB = Img.GetPixel(x, y);
        Img.SetPixel(x, y, Color.FromArgb(RGB.R, 0, 0));
    }
}
```

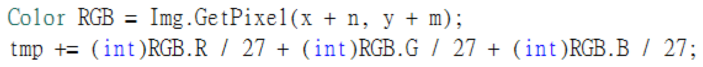
把每個 pixel 的 RGB 三個 channel 分離出來



```
int grayScale = (int)((RGB.R * 0.3) + (RGB.G * 0.59) + (RGB.B * 0.11));
Img.SetPixel(x, y, Color.FromArgb(grayScale, grayScale, grayScale));
```

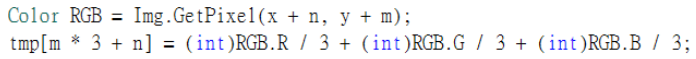
灰階的部分則是用 30% red 59% green 11% blue 的 channel 來實現

2. Smooth filter (mean and median)



```
Color RGB = Img.GetPixel(x + n, y + m);
tmp += (int)RGB.R / 27 + (int)RGB.G / 27 + (int)RGB.B / 27;
```

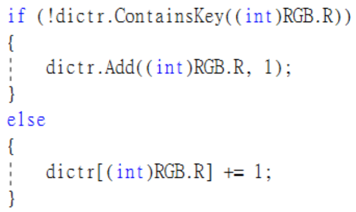
將九宮格周圍的 8 個 pixel 與 RGB channel 取平均



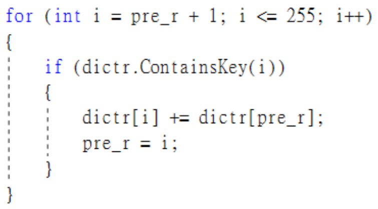
```
Color RGB = Img.GetPixel(x + n, y + m);
tmp[m * 3 + n] = (int)RGB.R / 3 + (int)RGB.G / 3 + (int)RGB.B / 3;
```

將九宮格周圍的 RGB channel 取平均後，再取 8 個 pixel 中中位數

3. Histogram Equalization

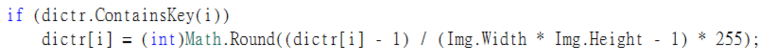


```
if (!dictr.ContainsKey((int)RGB.R))
{
    dictr.Add((int)RGB.R, 1);
}
else
{
    dictr[(int)RGB.R] += 1;
}
```



```
for (int i = pre_r + 1; i <= 255; i++)
{
    if (dictr.ContainsKey(i))
    {
        dictr[i] += dictr[pre_r];
        pre_r = i;
    }
}
```

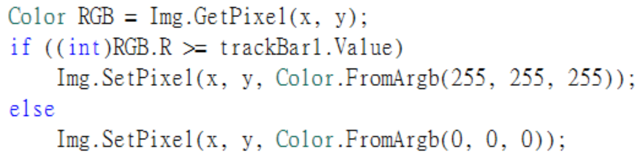
分別對 RGB channel 用 dictionary 做統計與累加



```
if (dictr.ContainsKey(i))
    dictr[i] = (int)Math.Round((dictr[i] - 1) / (Img.Width * Img.Height - 1) * 255);
```

計算比例後給予新的值

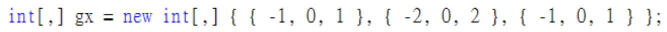
4. A user-defined thresholding



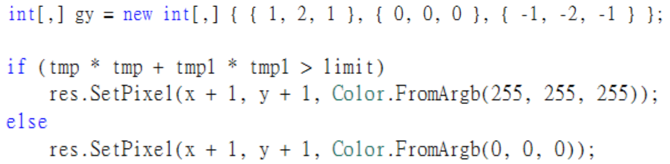
```
Color RGB = Img.GetPixel(x, y);
if ((int)RGB.R >= trackBar1.Value)
    Img.SetPixel(x, y, Color.FromArgb(255, 255, 255));
else
    Img.SetPixel(x, y, Color.FromArgb(0, 0, 0));
```

給 trackbar 值後，比該值大的設為白色，其餘設黑色

5. Sobel edge detection



```
int[,] gx = new int[,] { { -1, 0, 1 }, { -2, 0, 2 }, { -1, 0, 1 } };
```

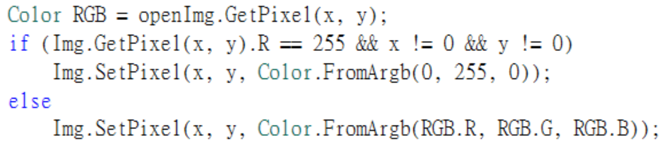


```
int[,] gy = new int[,] { { 1, 2, 1 }, { 0, 0, 0 }, { -1, -2, -1 } };

if (tmp * tmp + tmp1 * tmp1 > limit)
    res.SetPixel(x + 1, y + 1, Color.FromArgb(255, 255, 255));
else
    res.SetPixel(x + 1, y + 1, Color.FromArgb(0, 0, 0));
```

分別用 gx 跟 gy 遍歷圖片上每個點做垂直水平與組合的邊緣偵測，得到的數值分別與 limit 做比較，比較大為白色，其餘為黑色。

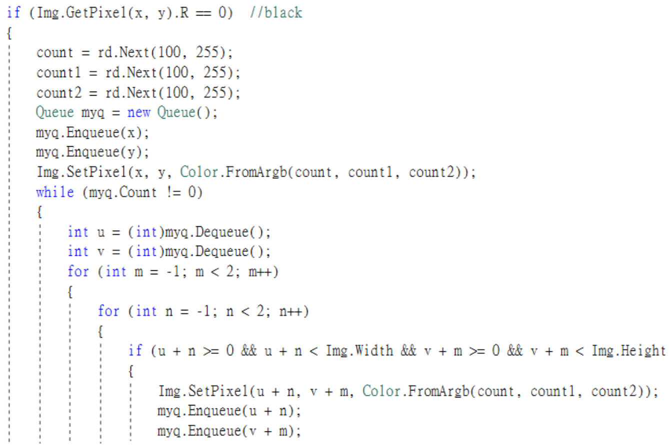
6. Edge overlapping



```
Color RGB = openImg.GetPixel(x, y);
if (Img.GetPixel(x, y).R == 255 && x != 0 && y != 0)
    Img.SetPixel(x, y, Color.FromArgb(0, 255, 0));
else
    Img.SetPixel(x, y, Color.FromArgb(RGB.R, RGB.G, RGB.B));
```

因為 Edge overlapping 是在 Sobel edge detection 後做，所以 edge 圖的 pixel 只有黑白兩種狀況，當是 edge 圖是白色時就輸出 edge 圖的 pixel，其餘則輸出原圖

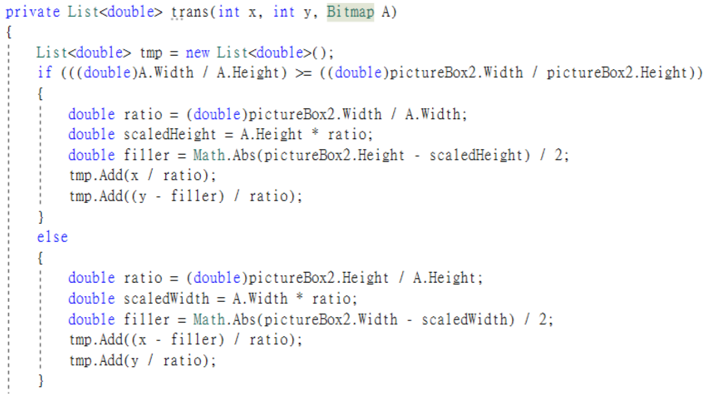
7. Connected Component



```
if (Img.GetPixel(x, y).R == 0)  //black
{
    count = rd.Next(100, 255);
    count1 = rd.Next(100, 255);
    count2 = rd.Next(100, 255);
    Queue myq = new Queue();
    myq.Enqueue(x);
    myq.Enqueue(y);
    Img.SetPixel(x, y, Color.FromArgb(count, count1, count2));
    while (myq.Count != 0)
    {
        int u = (int)myq.Dequeue();
        int v = (int)myq.Dequeue();
        for (int m = -1; m < 2; m++)
        {
            for (int n = -1; n < 2; n++)
            {
                if (u + n >= 0 && u + n < Img.Width && v + m >= 0 && v + m < Img.Height
                {
                    Img.SetPixel(u + n, v + m, Color.FromArgb(count, count1, count2));
                    myq.Enqueue(u + n);
                    myq.Enqueue(v + m);
```

只針對黑色 pixel 來做，遇到時就用 bfs 去找相連的黑色 pixel，並給個 random 的顏色

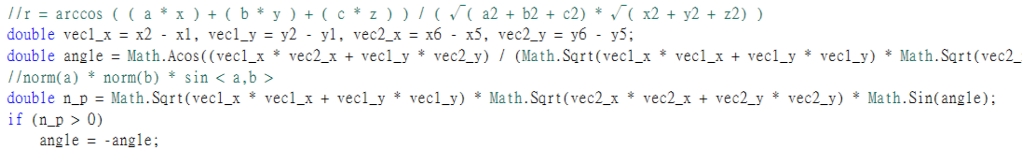
8. Image registration



```
private List<double> trans(int x, int y, Bitmap A)
{
    List<double> tmp = new List<double>();
    if (((double)A.Width / A.Height) >= ((double)pictureBox2.Width / pictureBox2.Height))
    {
        double ratio = (double)pictureBox2.Width / A.Width;
        double scaledHeight = A.Height * ratio;
        double filler = Math.Abs(pictureBox2.Height - scaledHeight) / 2;
        tmp.Add(x / ratio);
        tmp.Add((y - filler) / ratio);
    }
    else
    {
        double ratio = (double)pictureBox2.Height / A.Height;
        double scaledWidth = A.Width * ratio;
        double filler = Math.Abs(pictureBox2.Width - scaledWidth) / 2;
        tmp.Add((x - filler) / ratio);
        tmp.Add(y / ratio);
    }
```

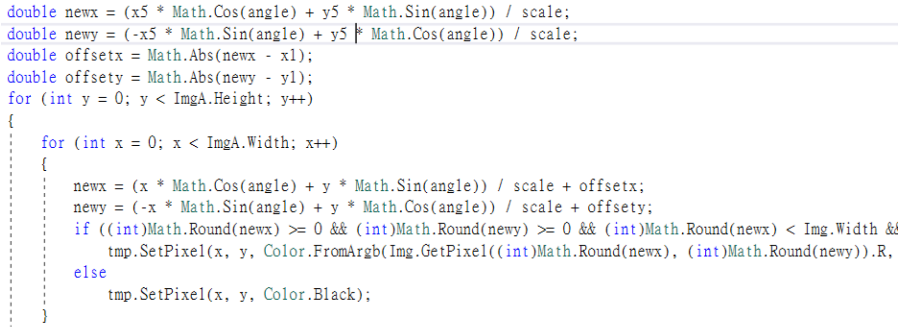
先取滑鼠點擊 picturebox 的位置，由於 picturebox 的 sizemode 是用 zoom，

代表 picturebox 會對 picture 做縮放再顯示，所以滑鼠所點的位置並不是 picture 的真實位置，因此用上面的 code 將滑鼠所點的位置轉換回，picture 上的真實位置。



```
//r = arccos ( ( a * x ) + ( b * y ) + ( c * z ) ) / ( √( a2 + b2 + c2) * √( x2 + y2 + z2) )
double vec1_x = x2 - x1, vec1_y = y2 - y1, vec2_x = x6 - x5, vec2_y = y6 - y5;
double angle = Math.Acos((vec1_x * vec2_x + vec1_y * vec2_y) / (Math.Sqrt(vec1_x * vec1_x + vec1_y * vec1_y) * Math.Sqrt(vec2_
//norm(a) * norm(b) * sin < a,b >
double n_p = Math.Sqrt(vec1_x * vec1_x + vec1_y * vec1_y) * Math.Sqrt(vec2_x * vec2_x + vec2_y * vec2_y) * Math.Sin(angle);
if (n_p > 0)
    angle = -angle;
```

接著用向量公式來算，角度與縮放。



```
double newx = (x5 * Math.Cos(angle) + y5 * Math.Sin(angle)) / scale;
double newy = (-x5 * Math.Sin(angle) + y5 * Math.Cos(angle)) / scale;
double offsetx = Math.Abs(newx - x1);
double offsety = Math.Abs(newy - y1);
for (int y = 0; y < ImgA.Height; y++)
{
    for (int x = 0; x < ImgA.Width; x++)
    {
        newx = (x * Math.Cos(angle) + y * Math.Sin(angle)) / scale + offsetx;
        newy = (-x * Math.Sin(angle) + y * Math.Cos(angle)) / scale + offsety;
        if ((int)Math.Round(newx) >= 0 && (int)Math.Round(newy) >= 0 && (int)Math.Round(newx) < Img.Width &&
            tmp.SetPixel(x, y, Color.FromArgb(Img.GetPixel((int)Math.Round(newx), (int)Math.Round(newy)).R,
        else
            tmp.SetPixel(x, y, Color.Black);
    }
```

用正解圖大小的空 bitmap 映射到旋轉過的圖，直接取旋轉過圖的 pixel 值填在空 bitmap，這樣就不需要處理旋轉時需要考慮填充的問題。另外旋轉與縮放都是以原點為基準點，因此要給個 offset。

Results

1. RGB Extraction & transformation

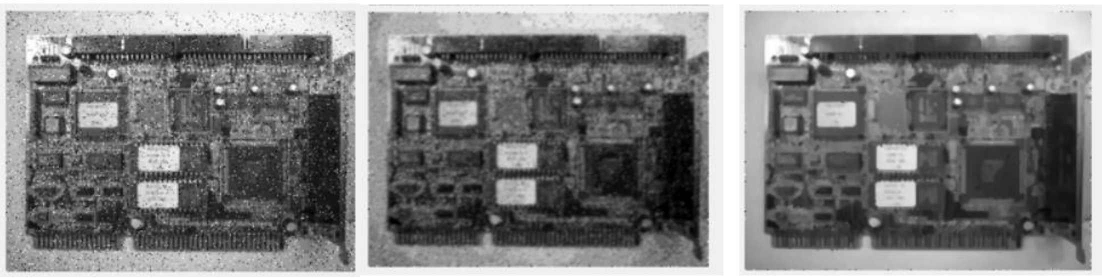
R

G

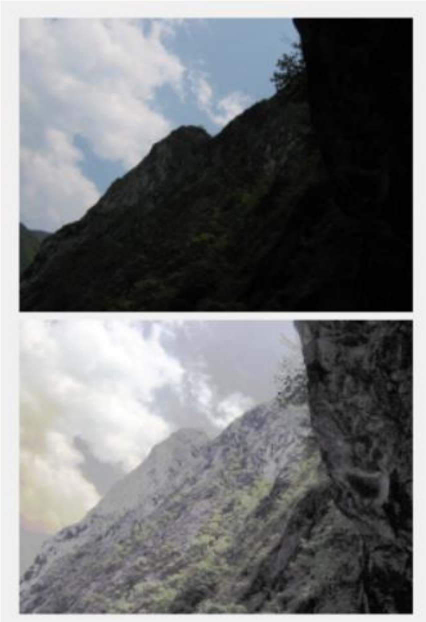
B

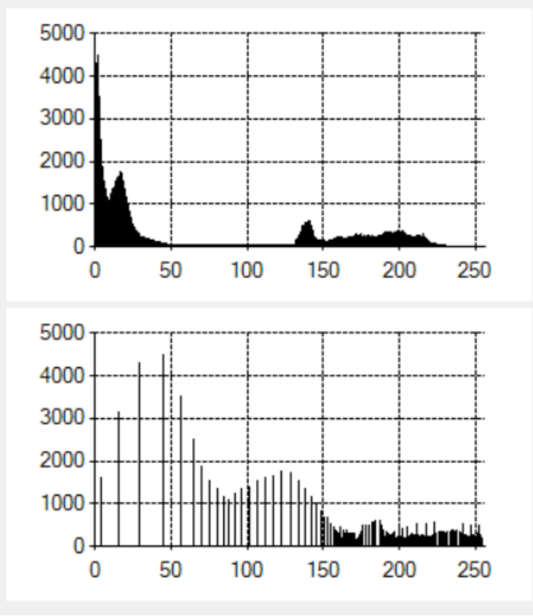
Greyscale

2. Smooth filter (mean and median)

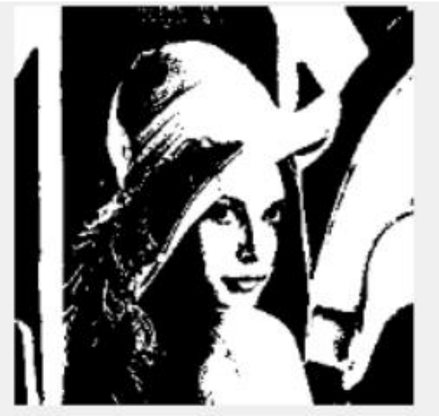


3. Histogram Equalization

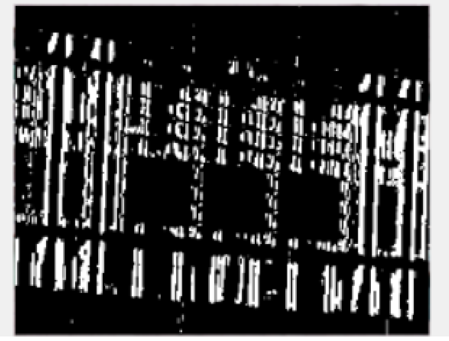


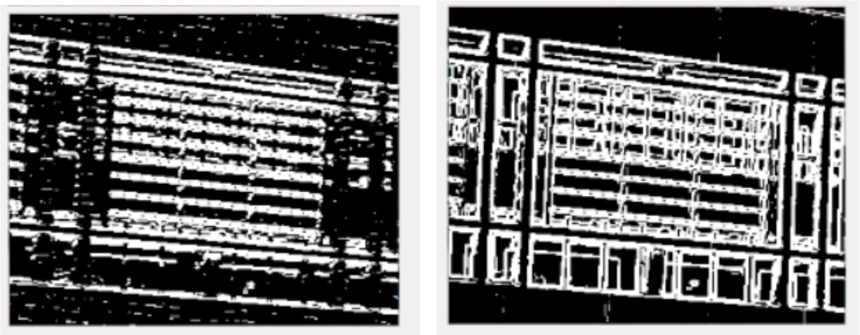


4. A user-defined thresholding

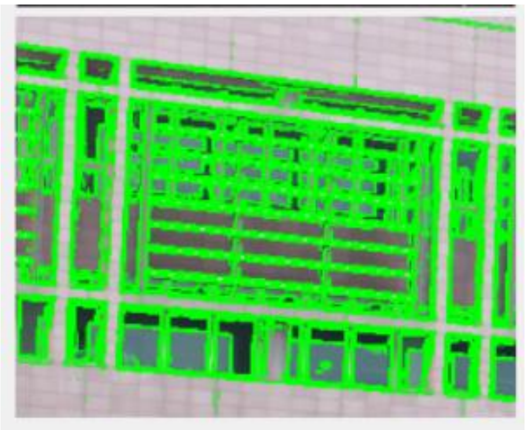


5. Sobel edge detection





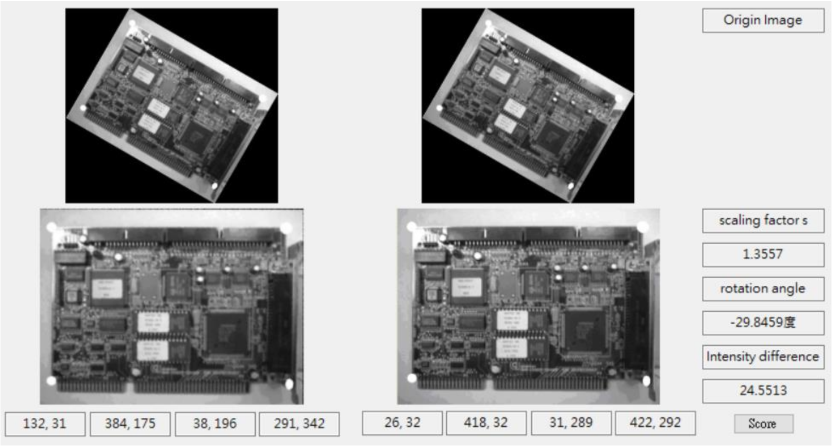
6. Edge overlapping



7. Connected Component

44

8. Image registration



scaling factor s
1.3557
rotation angle
-29.8459度
Intensity difference
24.5513

132, 31	384, 175	38, 196	291, 342	26, 32	418, 32	31, 289	422, 292