Computer Vision - Homework 4

開發環境

- OS: Windows 10 Pro
- Program Language: C# (with .Net Core 3.1)
- IDE: Visual Studio 2019
- Project: Console Application

程式說明

程式碼主要寫在 Program.cs，各題目程式皆已實作個別方法，由 Main entry 進行呼叫，答案結果儲存於 answers 資料夾。

各題目程式碼片段、參數及相關演算法說明如下：

(A). Dilation: 傳入 Binary Image 跟 Octagon kernel，藉由迴圈取得所有 Pixel 值後，若 Pixel 值為白色，將 Kernel 區域的 Pixel 更新為白色。

```
/// <summary>
/// 產生 Dilation 影像 (膨脹影像)
/// </summary>
/// <param name="srcImg">來源圖片</param>
/// <param name="kernel">kernel</param>
/// <returns>Dilation 影像</returns>
private static Bitmap GetDilationBitmap(Bitmap srcImg, IList<Point> kernel)
{
    var result = new Bitmap(srcImg);

    for (var x = 0; x < srcImg.Width; x++)
    {
        for (var y = 0; y < srcImg.Height; y++)
        {
            // 判斷是否為白色
            if (srcImg.GetPixel(x, y).R != 255) continue;

            foreach (var point in kernel)
            {
                var px = x + point.X;
                var py = y + point.Y;

                if (px >= 0 && px < srcImg.Width && py >= 0 && py <
srcImg.Height)
                {
                    result.SetPixel(px, py, Color.White);
                }
            }
        }
    }

    return result;
}
```

(B). Erosion: 傳入 Binary Image 跟 Octagon kernel，藉由迴圈判斷 Kernel 區域內是否存在且皆為白色，若是則將 Pixel 值更新為白色。

```
/// <summary>
/// 產生 Erosion 影像 (侵蝕影像)
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <param name="kernel">kernel</param>
/// <returns>Erosion 影像</returns>
private static Bitmap GetErosionBitmap(Bitmap srcImg, IList<Point> kernel)
{
    var result = CreateBlankBitmap(srcImg.Width, srcImg.Height);

    for (var x = 0; x < srcImg.Width; x++)
    {
        for (var y = 0; y < srcImg.Height; y++)
        {
            var contained = true;

            foreach (var point in kernel)
            {
                var px = x + point.X;
                var py = y + point.Y;

                if (px < 0 ||
                    px >= srcImg.Width ||
                    py < 0 ||
                    py >= srcImg.Height ||
                    srcImg.GetPixel(px, py).R != 255)
                {
                    contained = false;
                    break;
                }
            }

            if(contained)
                result.SetPixel(x, y, Color.White);
        }
    }

    return result;
}
```

(C). Opening: 先 Erosion 再 Dilation 後得到 Opening Image

```
/// <summary>
/// 取得 Open Bitmap (先 Erosion 再 Dilation)
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <param name="kernel">結構</param>
/// <returns></returns>
private static Bitmap GetOpenBitmap(Bitmap srcImg, IList<Point> kernel)
{
    return GetDilationBitmap(GetErosionBitmap(srcImg, kernel), kernel);
}
```

(D). Closing: 先 Dilation 再 Erosion 後得到 Closing Image

```
/// <summary>
/// 取得 Closing Bitmap (先 Dilation 再 Erosion)
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <param name="kernel">結構</param>
/// <returns></returns>
private static Bitmap GetClosingBitmap(Bitmap srcImg, IList<Point> kernel)
{
    return GetErosionBitmap(GetDilationBitmap(srcImg, kernel), kernel);
}
```

(E). Hit-and-miss transform: 建立兩個 Kernel 分別為 L1 及 L2, 將 原影像與 L1 進行 erosion, 原影像的補集與 L2 進行 erosion. 最後再取兩影像的交集

```
private static readonly List<Point> L1 = new List<Point>
{
    new Point(0, -1),
    new Point(0, 0),
    new Point(1, 0)
};

private static readonly List<Point> L2 = new List<Point>
{
    new Point(1, -1),
    new Point(2, -1),
    new Point(1, -2)
};

/// <summary>
/// 取得 HitAndMiss Bitmap
/// </summary>
/// <param name="srcImg"></param>
/// <returns></returns>
private static Bitmap GetHitAndMissBitmap(Bitmap srcImg)
{
    var result = CreateBlankBitmap(srcImg.Width, srcImg.Height);
    var cmpImg = new Bitmap(srcImg);

    for (var x = 0; x < srcImg.Width; x++)
    {
        for (var y = 0; y < srcImg.Height; y++)
        {
            cmpImg.SetPixel(x, y, srcImg.GetPixel(x, y).R == 255 ? Color.Black :
Color.White);
        }
    }

    var e1 = GetErosionBitmap(srcImg, L1);
    var e2 = GetErosionBitmap(cmpImg, L2);

    for (var x = 0; x < srcImg.Width; x++)
    {
        for (var y = 0; y < srcImg.Height; y++)
```

```
            {
                if (e1.GetPixel(x, y).R == 255 && e2.GetPixel(x, y).R == 255)
                {
                    result.SetPixel(x, y, Color.White);
                }
            }
        }

        return result;
    }
```

結果圖片

(A).

(B).

(C).

(D).

(E).