Computer Vision - Homework 5

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). Gray Dilation: 傳入 Gray Image 跟 Octagon kernel，藉由迴圈取得所有 Pixel 值後，取得 kernel 區域內最大值後更新至 Pixel。

```
/// <summary>
/// 取得 Gray Dilation 影像
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <param name="kernel">Kernel</param>
/// <returns>Gray Dilation Bitmap</returns>
private static Bitmap GetGrayDilationBitmap(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++)
        {
            var max = 0;

            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)
                {
                    var tmp = srcImg.GetPixel(px, py).R + point.Value;

                    if (max < tmp)
                    {
                        max = tmp;
                    }
                }
            }

            result.SetPixel(x, y, Color.FromArgb(max, max, max));
        }
    }

    return result;
}
```

(B). Gray Erosion: 傳入 Gray Image 跟 Octagon kernel，藉由迴圈取得 Kernel 區域內的最小值，將 Pixel 更新為其值。

```
/// <summary>
/// 取得 Gray Erosion 影像
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <param name="kernel">Kernel</param>
/// <returns>Gray Erosion Bitmap</returns>
private static Bitmap GetGrayErosionBitmap(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++)
        {
            var min = 256;

            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)
                {
                    var tmp = srcImg.GetPixel(px, py).R - point.Value;
                    if (tmp < min)
                    {
                        min = tmp;
                    }
                }
            }

            if (min < 0)
            {
                min = 0;
            }

            result.SetPixel(x, y, Color.FromArgb(min, min, min));
        }
    }

    return result;
}
```

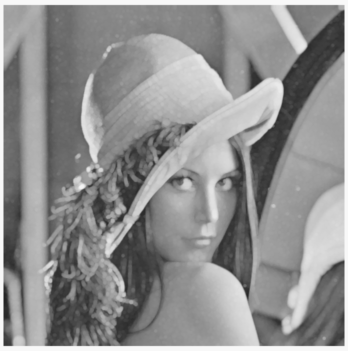
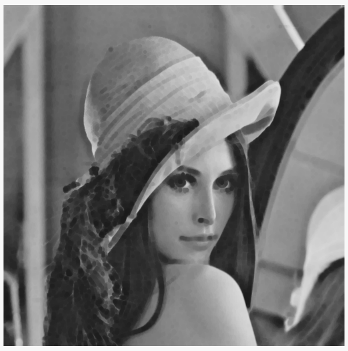
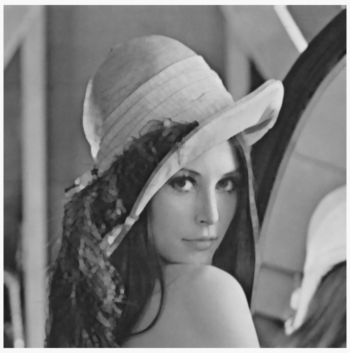
(C). Gray Opening: 先 Gray Erosion 再 Gray Dilation 後得到 Opening Image

```
/// <summary>
/// 取得 Gray Opening 影像
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <param name="kernel">Kernel</param>
/// <returns>Gray Opening Bitmap</returns>
private static Bitmap GetGrayOpeningBitmap(Bitmap srcImg, IList<Point> kernel)
{
    return GetGrayDilationBitmap(GetGrayErosionBitmap(srcImg, kernel), kernel);
}
```

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

```
/// <summary>
/// 取得 Gray Closing 影像
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <param name="kernel">Kernel</param>
/// <returns>Gray Closing Bitmap</returns>
private static Bitmap GetGrayClosingBitmap(Bitmap srcImg, IList<Point> kernel)
{
    return GetGrayErosionBitmap(GetGrayDilationBitmap(srcImg, kernel), kernel);
}
```

結果圖片

(A). 	(B). 	(C). 
(D).