Computer Vision - Homework 1

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). Upside Down: 藉由遞迴方式巡迴影像的所有 Pixel 值後進行上下轉換

```
/// <summary>
/// 上下翻轉
/// </summary>
/// <param name="srcImg">原始圖檔</param>
/// <returns>結果圖檔</returns>
private static Bitmap GetUpsideDownBitmap(Bitmap srcImg)
{
    if (srcImg == null)
        throw new Exception("source image is null");

    var dstBmp = new Bitmap(srcImg.Width, srcImg.Height);

    // x, y 分別為圖形像素的座標, 左上為 (0, 0)
    for (var x = 0; x < srcImg.Width; x++)
    {
        for (var y = 0; y < srcImg.Height; y++)
        {
            // 取得 (x, y) 的 pixel 值, 並更新至新的座標位置, 更新像素位置算法如下:
            // x 軸座標不變, y 軸座標為圖形高度 - 1 - y (ex. y = 0, img.height =
255, y' = img.height - 1 - 0 = 254)
            dstBmp.SetPixel(x, srcImg.Height - 1 - y, srcImg.GetPixel(x, y));
        }
    }

    return dstBmp;
}
```

(B). Right-Side Left: 藉由遞迴方式巡迴影像的所有 Pixel 值後進行左右轉換

```
/// <summary>
/// 左右翻轉
/// </summary>
/// <param name="srcImg">原始圖檔</param>
/// <returns>結果圖檔</returns>
private static Bitmap GetRightSideLeftBitmap(Bitmap srcImg)
```

```
{
    if (srcImg == null)
        throw new Exception("source image is null");

    var dstBmp = new Bitmap(srcImg.Width, srcImg.Height);

    // 方法同上，僅 x, y 軸進行交換，改固定 y 軸，x 軸座標為圖形寬度 - 1 - x
    for (var x = 0; x < srcImg.Width; x++)
    {
        for (var y = 0; y < srcImg.Height; y++)
        {
            dstBmp.SetPixel(srcImg.Width - 1 - x, y, srcImg.GetPixel(x, y));
        }
    }

    return dstBmp;
}
```

(C). Diagonally Mirrored: 將影像 (x, y) 的 pixel 指向 (y, x) 的對稱位置。ex. (2, 0) > (0, 2)

```
/// <summary>
/// 對角線翻轉
/// </summary>
/// <param name="srcImg">原始圖檔</param>
/// <returns>結果圖檔</returns>
private static Bitmap GetDiagonallyMirroredBitmap(Bitmap srcImg)
{
    if (srcImg == null)
        throw new Exception("source image is null");

    var dstBmp = new Bitmap(srcImg.Width, srcImg.Height);

    for (var x = 0; x < srcImg.Width; x++)
    {
        for (var y = 0; y < srcImg.Height; y++)
        {
            // 將 (y, x) 設定為 (x, y) 的值
            dstBmp.SetPixel(y, x, srcImg.GetPixel(x, y));
        }
    }

    return dstBmp;
}
```

(D). Rotate Image: 此部分引入了 OpenCvSharp4 及 OpenCvSharp4.Runtime.Windows 兩個函式庫，並透過取得中心點後使用 GetRotationMatrix2D 進行影像轉換

```
/// <summary>
/// 取得旋轉後的影像
/// </summary>
/// <param name="angle">角度(正值為逆時鐘旋轉, 負值為順時鐘旋轉)</param>
/// <param name="scale">影像比例</param>
/// <param name="srcImg">來源影像</param>
/// <returns>結果影像</returns>
private static Mat GetRotateImage(double angle, double scale, Mat srcImg)
{
```

```
    var dstImg = new Mat();

    // 取得影像中心後藉由 OpenCvSharp 的 GetRotationMatrix2D 函式進行角度的旋轉
    var imageCenter = new Point2f(srcImg.Cols / 2f, srcImg.Rows / 2f);
    var rotationMat = Cv2.GetRotationMatrix2D(imageCenter, angle, scale);

    Cv2.WarpAffine(srcImg, dstImg, rotationMat, srcImg.Size());

    return dstImg;
}
```

(E). Resize: 同 (D)，使用 OpenCvSharp4 的 Resize 函式進行影像尺寸的變更

```
/// <summary>
/// 取得 Resize 後的影像
/// </summary>
/// <param name="zoom">縮放比例(百分比)</param>
/// <param name="srcImg">來源影像</param>
/// <returns>結果影像</returns>
private static Mat GetResizeImage(double zoom, Mat srcImg)
{
    var dstImg = new Mat();
    // 取得要 zoomin 的 size 後藉由 OpenCvSharp 的 Resize 函式進行尺寸轉換
    var size = new OpenCvSharp.Size(srcImg.Width * zoom, srcImg.Height * zoom);

    Cv2.Resize(srcImg, dstImg, size);

    return dstImg;
}
```

(F). Binary Image: 同 (D)，使用 OpenCvSharp4 的 Threshold 設定 Thresh 值及最大值進行二元影像轉換

```
/// <summary>
/// 取得二元影像
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <returns>結果影像</returns>
private static Mat GetBinaryImage(Mat srcImg)
{
    var dstImg = new Mat();

    // 藉由 OpenCvSharp 的 Threshold 函式定義 Thresh 值以及最大值，並進行 Binary 轉換
    Cv2.Threshold(srcImg, dstImg, 128, 255, ThresholdTypes.Binary);

    return dstImg;
}
```

結果圖片

(A).

(B).

(C).

(D).

(E).

(F).