Computer Vision - Homework 3

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). original image and its histogram: 迴圈取得所有 Pixel 值後，根據灰階值的對應進行數量的加總後繪製直方圖，X 軸為 0 ~ 255 的灰階值，Y 軸為對應灰階值的數量加總。(本題加入 ScottPlot 的參考作為直方圖繪製的函式庫)

```
/// <summary>
/// 取得直方圖
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <returns>Scott.Plot</returns>
private static Plot GetHistogram(Bitmap srcImg)
{
    if (srcImg == null)
        throw new Exception("source image is null");

    // xs: x 軸(0 ~ 255 的灰階值)
    var xs = new double[256];
    for (var x = 0; x < 256; x++) xs[x] = x;

    // ys: y 軸; 影像內灰階值對應的數量加總
    var ys = new double[256];

    for (var x = 0; x < srcImg.Width; x++)
    for (var y = 0; y < srcImg.Height; y++)
    {
        var color = srcImg.GetPixel(x, y);
        var gray = (color.R + color.G + color.B) / 3;
        ys[gray] += 1;
    }

    var plt = new Plot();
    plt.Title("Histogram");
    plt.XLabel("Gray (0~255)");
    plt.YLabel("Count");
    plt.PlotBar(xs, ys, barWidth: 1.4D, outlineWidth: 0);

    return plt;
}
```

(B). image with intensity divided by 3 and its histogram

```
/// <summary>
/// 取得較暗的影像
/// </summary>
/// <param name="srcImg"></param>
/// <returns></returns>
private static Bitmap GetIntensityDivided3Image(Bitmap srcImg)
{
    if (srcImg == null)
        throw new Exception("source image is null");

    var destImg = new Bitmap(srcImg);

    for (var x = 0; x < srcImg.Width; x++)
    for (var y = 0; y < srcImg.Height; y++)
    {
        var color = srcImg.GetPixel(x, y);
        // 值 / 3
        destImg.SetPixel(x, y, Color.FromArgb(color.R / 3, color.G / 3, color.B
/ 3));
    }

    return destImg;
}
```

(C). image after applying histogram equalization to (b) and its histogram: 先取得 Histogram 的灰階值數量, 並計算機率. 完成後依照histogram equalization histogram linearization 公式計算後更新至對應的 x, y 值

```
/// <summary>
/// 取得 Histogram Equalization Image
/// </summary>
/// <param name="srcImg">來源影像</param>
/// <returns>Histogram Equalization 影像</returns>
private static Bitmap GetEqualizationImage(Bitmap srcImg)
{
    // 機率分布
    var density = new double[256];
    var destImg = new Bitmap(srcImg);

    // 取得所有灰階值
    for (var x = 0; x < srcImg.Width; x++)
    for (var y = 0; y < srcImg.Height; y++)
    {
        var color = srcImg.GetPixel(x, y);
        var gray = (color.R + color.G + color.B) / 3;
        density[gray] += 1;
    }

    // 藉由灰階值數量計算機率
    for (var index = 0; index < 256; index++)
        density[index] = density[index] / (srcImg.Width * srcImg.Height * 1.0);

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

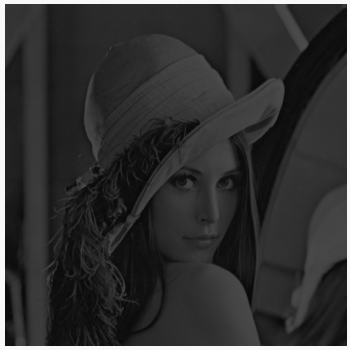
```
    {
        double sum = 0;
        var value = srcImg.GetPixel(x, y);
        for (var k = 0; k <= value.R; k++) sum += density[k];
        var rgb = (byte) Math.Round(255 * sum);
        destImg.SetPixel(x, y, Color.FromArgb(rgb, rgb, rgb));
    }

    return destImg;
}
```

結果圖片

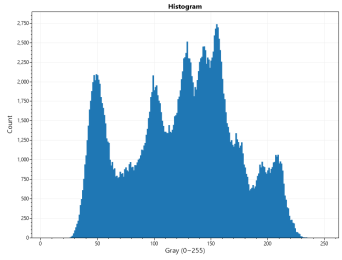
(A-1).

(B-1).

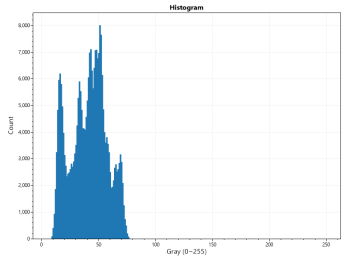


(C-1).

(A-2).



(B-2).



(C-2).