高等影像處理

Homework 4: Convolution: Image smoothing and edge detection

學號： 61247051S 姓名： 陳建壹

一、本作業所用之程式語言及編譯器

Language: C#

Compiler: Roslyn

二、程式功能

1. HW1, HW2, HW3 之功能
2. 使用者自訂kernel
3. 影像平滑化
4. 影像邊緣偵測

三、程式流程或演算法

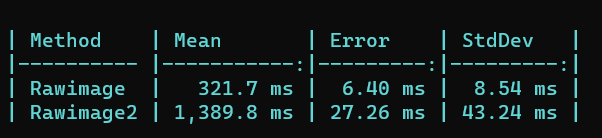
|  |
| --- |
| RawImage Convolution(RawImage input, Mask kernel) {  offset = kernel.Size / 2;  output = new(Width, Height);  foreach(x, y : Width, Height) {  value = 0.0f;  foreach(i, j : offset, offset){  pX = Clamp(x + i, 0, Width - 1);  pY = Clamp(y + j, 0, Height - 1);  kernelValue = kernel[i + offset][j + offset];  pIndex = pY \* Width + pX;  value += (input[pIndex] & 0xFF) \* kernelValue;  }  int index = y \* Width + x;  value = Clamp(value / kernel.Scalar, 0, 255);  output[index] = (value << B | value << G | value << R | 0xFF000000);  }  return output;  } |
| RawImage EdgeDetection(RawImage input) {    smooth = Smooth(input);  grayscale = GrayScale(smooth);  sobelXmask = LoadMask(SobelX);  sobelYmask = LoadMask(SobelY);  sobelXimage = Convolution(grayscale, sobelXmask);  sobelYimage = Convolution(grayscale, sobelYmask);  Function grad = (gx, gy) => Clamp(Math.Sqrt(gx \* gx + gy \* gy), 0, 255);  gradient = OverlayCalculate(sobelXimage, sobelYimage, grad);  return Reverse(gradient);  }  sobelX = { -1, 0, 1, -2, 0, 2, -1, 0, 1 }  sobelY = { -1, -2, -1, 0, 0, 0, 1, 2, 1 }  gaussian = { 2, 4, 5, 4, 2, 4, 9, 12, 9, 4, 5, 12, 15, 12, 5, 4, 9, 12, 9, 4, 2, 4, 5, 4, 2 }/159 |

四、測試結果

|  |
| --- |
| 範例一為[平滑化]之測試結果。左圖為input，右為output。由右圖output可觀察到input被轉為灰階且馬卡龍顆粒變得較為平滑 |
| 範例二為[平滑化]之測試結果。左圖為input，右為output。由右圖output可觀察到input被轉為灰階且變得較為平滑 |
| 範例三為[邊緣偵測]之測試結果。左圖為input，右為output。由右圖output可觀察到input之邊緣 |
| 範例四為[邊緣偵測]之測試結果。左圖為input，右為output。由右圖output可觀察到input之邊緣 |

五、程式撰寫心得

本次作業除了完成卷積運算，也改寫了儲存影像的資料結構,可以支援Bitwise操作RGBA值，加快了執行速度，整個RawImage class的結構也更為精簡可讀。另外也將目前的演算法改為支援Multithread, 對於大型圖檔處理有顯著的加速效果。(包含noise, convolution, gaussian, grayscale, rotation)



對影像進行Convolution運算，使用BenchmarkDotnet進行測試 , (RawImage為改寫後, RawImage2為改寫前.)