電機所-陳治言-R06921084

Description :

一開始，先處理圖片檔的標頭資訊，包括 size，offset，width，height，bits 等資訊，之後再將圖片的色彩值讀入，因為這次範例圖片只有 8 位元 色彩，所以每個畫素只需要 1byte 即可儲存。而作業分為 3 部份，分別為 binary、histogram 和 connected components。處理完所有的圖片之後，再將標頭與畫素資料一起寫出並存，並且分別產生 3 個檔案，binary_lena、histogram_lena 和 connected_components_lena 三張輸出圖片，即可完成這次的作業。

Algorithm :

一開始的 binary，使用非常簡單的迭代演算法即可，將數值小於 128 全部調整為 0，大於 128 的則是調整為 255。

再者的 histogram，透過上面的步驟中，一起統計出 0 到 255 的所有像素的出現頻率。再來，將結果直接輸出成圖片，因為圖片高度為 512，所以必須事先將頻率最高的像素設為 512，並將其他畫素依比例縮小，並將新的出現頻率繪製在 512x512 的圖片上。

最後的 connected_components，也是使用上課教的的迭代演算法來做計算，並使用 4 連通來做計算，將所有的 label 都完整標記上去之後，在開始過濾像素數小於 500 的區域，將其標示為背景而不考慮。最後，在掃描圖形，依序將各區域之中心和邊界給計算出來，最後繪製於圖上，並將原本圖片的黑色像素設為灰色，以利看出各區域的中心與邊界範圍。

Principal Code :

◎**Binary**

```
for(i=0; i<bmpInfo.biHeight; ++i)
    for(j=0; j<bmpInfo.biWidth; ++j)
    {
        //統計 0 到 255 的出現頻率供下個步驟使用
        ++histogram[BMPdata[i][j].color];
        //開始處理 binary
        if(BMPdata[i][j].color < 128)
```

```
                BMPdata[i][j].color = 0;
            else
                BMPdata[i][j].color = 255;
        }
```

◎Histogram

```
//先找出像素的最大值
for(i=0; i<256; ++i)
        max = max > histogram[i] ? max : histogram[i];
//將像素依比例縮小
for(i=0; i<256; ++i)
        histogram[i] = histogram[i] * 512 / max ;
//開始填入圖片畫格
for(i=0; i<512; ++i)
        for(j=0; j<256; ++j)
        {
            if(histogram[j] > 0)
            {
                BMPdata[i][j*2].color = 0;
                BMPdata[i][j*2+1].color = 0;
                --histogram[j];
            }
            else
            {
                BMPdata[i][j*2].color = 255;
                BMPdata[i][j*2+1].color = 255;
            }
        }
```

◎Connected_component

```
//先將圖上的所有像素標上號碼
for(i=0; i<bmpInfo.biHeight; ++i)
        for(j=0; j<bmpInfo.biWidth; ++j)
        {
            if(cnt[i][j])
            cnt[i][j] = count;
            ++count;
        }
```

```
//開始執行迭代演算法，使用 4 連通的方式來標記
do
{
	count = 0;
	for(i=0; i<bmpInfo.biHeight; ++i)
		for(j=0; j<bmpInfo.biWidth; ++j)
		{
			if(cnt[i][j])
			{
				//檢查上方像素
				if(i != 0 && cnt[i-1][j] && cnt[i-1][j] < cnt[i][j])
				{
					cnt[i][j] = cnt[i-1][j];
					count = 1;
				}
				//檢查左方像素
				f(j != 0 && cnt[i][j-1] && cnt[i][j-1] < cnt[i][j])
				{
					cnt[i][j] = cnt[i][j-1];
					count = 1;
				}
			}
		}

	for(i=bmpInfo.biHeight-1; i>-1; --i)
		for(j=bmpInfo.biWidth-1; j>-1; --j)
		{
			if(cnt[i][j])
			{
			//開始更新標籤編號，使用較小數字當作標籤
			if(i != bmpInfo.biHeight-1 && cnt[i+1][j] && cnt[i+1][j] < cnt[i][j])
			{
				cnt[i][j] = cnt[i+1][j];
				count = 1;
			}
			if(j != bmpInfo.biWidth-1    && cnt[i][j+1] && cnt[i][j+1] < cnt[i][j])
			{
				cnt[i][j] = cnt[i][j+1];
```

```
                        count = 1;
                    }
                }
            }
}while(count);//結束編號，所有的連通區域已標示完畢

//開始計算該區域的中心與邊界
for(i=0; i<bmpInfo.biHeight; ++i)
    for(j=0; j<bmpInfo.biWidth; ++j)
    {
        if(label[cnt[i][j]])
        {
            //使用重心公式，將所有座標相加除以總像素
            chosen[label[cnt[i][j]]-1].xc = chosen[label[cnt[i][j]]-1].xc + i;
            chosen[label[cnt[i][j]]-1].yc = chosen[label[cnt[i][j]]-1].yc + j;
            ++chosen[label[cnt[i][j]]-1].pc;
            //更新區域最大邊界
            if(i > chosen[label[cnt[i][j]]-1].buttom)
                chosen[label[cnt[i][j]]-1].buttom = i;
            if(i < chosen[label[cnt[i][j]]-1].top)
                chosen[label[cnt[i][j]]-1].top = i;
            if (j > chosen[label[cnt[i][j]]-1].right)
                chosen[label[cnt[i][j]]-1].right = j;
            if (j < chosen[label[cnt[i][j]]-1].left)
                chosen[label[cnt[i][j]]-1].left = j;
          }
       }

//在圖上標記寬度為 3pixel 的十字中心
for(j=-4; j<5; ++j)
{
    BMPdata[cntx+j][cnty].color = 0;
    BMPdata[cntx+j][cnty-1].color = 0;
    BMPdata[cntx+j][cnty+1].color = 0;
    BMPdata[cntx][cnty+j].color = 0;
    BMPdata[cntx-1][cnty+j].color = 0;
    BMPdata[cntx+1][cnty+j].color = 0;
}
```

```
//開始畫出各區域的邊界
for(k=0; k<bmpInfo.biHeight; ++k)
{
	for(l=0; l<bmpInfo.biWidth; ++l)
	{
		if(chosen[i].top <= k && k <= chosen[i].buttom)
		{
			//畫出寬度為 2pixel 的左邊界
			if(l == chosen[i].left)
			{
				BMPdata[k][l].color=0;
				BMPdata[k][l+1].color=0;
			}
			//畫出寬度為 2pixel 的右邊界
			if(l == chosen[i].right)
			{
				BMPdata[k][l].color=0;
				BMPdata[k][l-1].color=0;
			}
		}
		if(chosen[i].left <= l && l <= chosen[i].right)
		{
			//畫出寬度為 2pixel 的上邊界
			if(k == chosen[i].top)
			{
				BMPdata[k][l].color=0;
				BMPdata[k+1][l].color=0;
			}
			//畫出寬度為 2pixel 的下邊界
			if(k == chosen[i].buttom)
			{
				BMPdata[k][l].color=0;
				BMPdata[k-1][l].color=0;
			}
		}
	}
}
```

Parameters：

編譯程式碼 g++ -o lena lena.cpp

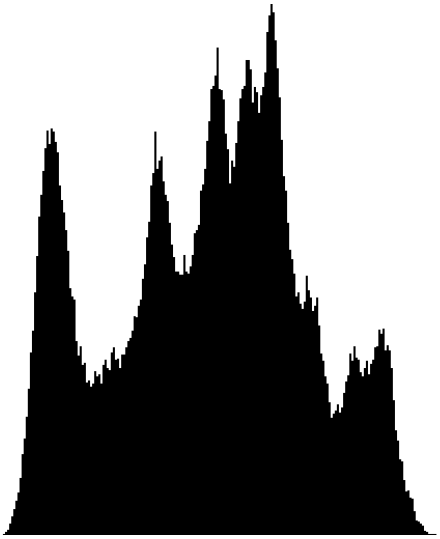
執行程式 ./lena lena.bmp

lena.bmp 是我們的 InputImage

Resulting Images：

binary_lena.bmp

histogram_lena.bmp



connected_components_lena.bmp