

GPGPU Lab3 Report

R05922087 張逸寧

Optimization

Laplacian solver

對於題目原先公式可以推得一系數矩陣，

0	1	0
1	-4	1
0	1	0

觀察後可發現 Poisson Editing 公式在此應用下的物理意義為水平與垂直的像素遞移，並且發現系數矩陣與 Laplacian Solver 相同，在數學意義上是利用微分並積回來增加函數連續性，而在圖學物理意義上是將斷點邊界模糊化。故將系數矩陣擴大至 5*5，且增設 45 度角之像素係數，增加像素傳遞速度，使用新系數矩陣為：

1	0	1	0	1
0	2	2	2	0
1	2	-24	2	1
0	2	2	2	0
1	0	1	0	1

此方法，可使疊代迅速收斂，但每一次疊代與原矩陣相較之下較花費時間，詳細效能在下一章節做比較。

Pixel subsample

將原像素取樣本，已 $1/4$ 為例，將原像素四格的 RGB 取平均後的值， Cr ，進行疊代，將疊代完成後的值， Cr' ，與 Cr 的差加入原像素四格的 RGB。需注意的是，在樣本比例轉換時，被 mask 遮罩住的像素也需要進行疊代，否則會產生邊界斷裂的情況，造成後面的疊代產生錯誤。

此方法，可加速疊代時的擴散，且因疊代每迴圈為平方量級的花費，故可降低總體的收斂時間。

Combination

因 Laplacian solver 可加速擴散，但每一迴圈增加執行時間，故再配合 Pixel subsample，可降低總體執行時間，在低像素時，兩者的配合能使擴散非常快速。詳細效能再下一章節做比較。

Experiment

Image 1:













Baseline	Laplacian	Optimal (Both)
 <p>2 0.448s</p>	 <p>2 0.432s</p>	 <p>1/4: 2, 1/1: 2 0.439s</p>
 <p>2000 0.477s</p>	 <p>2000 0.529</p>	 <p>1/4: 2000, 1/1: 2 0.483s</p>
 <p>6000 0.54s</p>	 <p>5000 0.647</p>	 <p>1/4: 4000, 1/1: 2 0.512s</p>
 <p>20000 0.754s</p>	 <p>8000 0.841s</p>	 <p>1/4: 4000, 1/1: 2000 0.564s</p>

Image 2:

Baseline	Laplacian	Optimal (Both)
 <p>2 0.45s</p>	 <p>2 0.45s</p>	 <p>1/4: 2, 1/1: 2 0.468s</p>
 <p>2000 0.529s</p>	 <p>2000 0.607s</p>	 <p>1/4: 2000, 1/1: 2 0.509s</p>
 <p>6000 0.65s</p>	 <p>5000 0.831s</p>	 <p>1/4: 4000, 1/1: 2 0.558s</p>
 <p>20000 1.076s</p>	 <p>8000 1.05s</p>	 <p>1/4: 4000, 1/1: 2000 0.698s</p>

Conclusion

上述實驗中，第一個圖表為基本測資，第二格圖表為額外測資。兩個圖表中，第一欄為基本做法，第二欄為 Laplacian solver 做法，第三欄為 Optimal 的做法 (Laplacian solver 加 Pixel subsampling)。其中每一格，上方為實驗結果圖，下方為疊代次數與總體執行時間。而在 Optimal 中，疊代次數分為 1/4 比例與 1/1 (未縮圖)比例的組合方式。

根據上述實驗結果可以發現，經由 Laplacian solver 可讓收斂疊代次數減少約一半，但整體執行時間不減反增，原因是每一次疊代的花費較多。故再套上 subsampling 的技術，在大像素下，像素傳遞更快，整體需要的疊代次數亦會降低，且每一次疊代的花費因樣本比例也大幅下降，故在整體執行上，能增進約 1.5 倍的效能。