**Texture Synthesis – project proposal**

1. **Title：**texture synthesis using wang tile and graph cut
2. **Motivation：**

在texture synthesis的領域中，由於wang tile[1]和graph cut[2]為同年發表，因此Wang tile在實作上並未參考Graph cut algorithm而參考的是dijkstra’s algorithm[3]，本次Project期望能夠在wang tile上使用graph cut algorithm期望能夠彌補[1]所造成的某些缺陷。

1. **Previous Work**

在圖學中，Texture synthesis一直以來都是一個很重要的問題，而解決這類型問題的方法分為Procedure-based、statistics-based、Example-based。這三種方法當中，以example-based為最快速且方便的生成方式，而在example-based中，我們又可以分成三塊分別為pixel-based[3]、patch-based[2]以及tile-based[1]，其中以tile-based再生成效率和結果上都非常迅速

Tile-based[1]主要是透過事先將input texture取出幾塊sample並且透過將sample排列組合的方式產生幾個tile，當我們要合成出大塊texture時，我們可以類似拼圖的方式去拼貼tile，讓相鄰的tile其邊是相同的來合成出一塊看起來像但又不完全相似的texture出來，但是因為使用dijkstra’s algorithm以及切割方式不同，而導致了幾個重要的缺點[4]：

* **Corner problem**：在找尋最小的minimum cost時並沒有確切的考慮到角落是否有確實的被切到，而導致其實角落部分並不屬於該邊，而造成合成出來的影像會有怪異的現象發生(圖一)。



圖一

* **Joint problem**：再將數塊sample貼起來時wang tile使用了dijkstra’s algorithm企圖在這些sample重複處找尋成本最小的，但此algorithm其實相當難找到這些sample中重疊部分成本最小的區域(圖二)



圖二

* **Sampling problem**：因為是從幾個samples中拼裝成tile，因此生成出來的圖形勢必只會在那幾個sample中變換，因此若取sample時只取input中的某些區塊，可能無法合成出相當不錯的圖形

1. **Proposed Method**

在[4]中他們針對這些問題提出了strict wang tile並且使用了graph cut algorithm作為尋找最小的cut的方法(圖三)，這樣的方法有一個缺點在於必須要全圖搜尋所以很浪費時間。

圖三 主要是將b當作一個image將另一張圖覆蓋在上面並且找尋minimum cut

本次project將會以實作[4]為主，但我們完全不需要將圖a做旋轉45度，而是直接透過我們所產生的圖三a未轉45度時的情況去進行合成。合成的方式如圖四，我們合成的順序是由左至右依次合成紫色圈圈的部分，這樣的方法我們可以確保Tile合成正確，而且我們還可以只要依據使用者指定的大小進行裁切，完全不需要考量到菱形裁切所會遇到的相關問題，我們可以直接切出藍綠色的部分當做合成結果。

|  |  |
| --- | --- |
| D:\Java Program\Wang Tile\說明.bmp |  |
| 圖四 合成方式以及合成結果 | |

我們接下來將討論進行Graph cut，如圖五所示，根據[2]中提到我們可以將要被Graph cut的影像當成Patch A，而Patch B的影像我們透過隨機在Sample圖中找到比較接近於原圖的子圖，這樣我們可以保證我們切出來的結果圖會跟原圖的基本結構不變，在Edge連結的方式我們將邊界的部分連到Node A，並且將中間因為來自不同Patch而產生的Artifact接到Patch B，如此我們可以確保這些點尋找min cut的結果一定會為Patch B，可以使Artifact因此而消失。而點與點之間Edge的決定方式是左上方的Pixel全部連結右邊和下面的Pixel；右上方的Pixel全部連結右邊和上面的Pixel；右上方的Pixel全部連結左邊和下面的Pixel；左下方的Pixel全部連結左邊和上面的Pixel，這樣我們可以保證我們一定會從Source流到Sink。並且透過[5]我們可以快速有效率的產生Graph cut將接縫進行有效的處理。

|  |  |
| --- | --- |
|  |  |
|  |
| 圖五 [2]當中型二的Graph cut圖 Pixel之間的連結方式以及結果圖 | |

1. **Result**

圖六顯示了一些成果圖，我們發現此方法在生成Near Regular Texture上面仍然有待加強，並不像paper那樣可以找到那麼優秀的synthesis圖片

|  |
| --- |
| D:\Java Program\Wang Tile\Images\brick.jpgD:\Java Program\Wang Tile\result3.jpg |
| D:\Java Program\Wang Tile\Images\cloth33.jpgD:\Java Program\Wang Tile\result2.jpg |
| 圖六 結果圖 sample大小為32 \* 32 生成512 \* 512 的圖 |

圖七圖八為我們和原始的Wang tile進行比較，至少我們看不出差異性的存在，在sample數較大的情況下，我們的方法不需要將Tile取成菱形就能比較簡單的合成出原始Wang tile達到的效果，而且合成的效果和sample取得大小有差，如果sample取得太小將會造成破碎狀況相當頻繁的結果。

|  |  |
| --- | --- |
|  | D:\Java Program\Wang Tile\fruit\result1-16.jpg |
| D:\Java Program\Wang Tile\fruit\result1-32.jpg | D:\Java Program\Wang Tile\fruit\result1-64.jpg |
| D:\Java Program\Wang Tile\Images\source_fruit.jpg | |
| 圖七 左上為來自Wang tile[1] 其他圖是我們的做法 右上是16\*16 的sample左下是32 \* 32 的sample右下 64 \* 64的sample合成512\*512的結果 最下面的圖是合成更大的圖後的效果 | |

|  |  |
| --- | --- |
|  | D:\Java Program\Wang Tile\result\result1-16.jpg |
| D:\Java Program\Wang Tile\result\result2-32.jpg | D:\Java Program\Wang Tile\result\result3-64.jpg |
| 圖八 左上為來自Wang tile[1] 其他圖是我們的做法 右上是16\*16 的sample左下是32 \* 32 的sample右下 64 \* 64的sample合成512\*512的結果 最下面的圖是合成更大的圖後的效果 | |

1. **Reference**

[1] M.F. Cohen, J.Shade, S.Hiller, O.Deussen **,”Wang Tiles for Image and Texture Generation”**, ACM Transaction on Graphics(Proceedings of ACM Siggraph) (2003)

[2] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, Aaron Bobick ,**” Graphcut Textures: Image and Video Texture Synthesis Using Graph Cuts”,** , ACM Transaction on Graphics(Proceedings of ACM Siggraph) (2003)

[3] Efros, A., AND Freeman, W. 2001. **” Image quilting for texture synthesis.”**

In Proceedings of SIGGRAPH 2001, 341–346.

[4] Zhang Xinyu, Kim Young J. ,**“Efficient Texture Synthesis Using Strict Wang Tiles”**, Graphical Models, 2008

[5] Yuri Boykov and Vladimir Kolmogorov,**“An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision”**,IEEE Transactions on PAMI, Vol. 26, No. 9, pp. 1124-1137, Sept. 2004