Faster-OilPainting: 快速照片油畫化

NTU Interactive Computer Graphic, 2018 Fall Final term project

Teammates

B04902028 洪浩翔
B04902090 施長元

Introduction

電腦視覺領域中，將圖片風格化是很常見的應用。本組先前專注於 survey 將圖片油畫化的相關研究，找到幾個相關的方案；但這些方案大多成果不太理想，或是效率不佳，有時單張圖片的運算時間就需要數秒甚至數分鐘的時間進行風格轉換，有些轉換的風格並不盡理想。

因此，我們參酌各方法重新設計了一個油畫演算法，針對了多個方向進行優化，使得油畫化照片的效率可以達到近乎 Real time，甚至我們將此算法推廣至 GIF 動畫進行轉換，使用我們的算法可以讓 144 個畫格(408x728)的動畫在 1 分鐘左右完成轉換風格。 (Based on i5-4210M with stock frequency)

Implementation

- 油畫化
  1. Gradient
     為了筆刷的方向性，我們採用了 Edge detector 尋找 X 方向與 Y 方向的分量。我們嘗試了四種不同的 Edge detector: Prewitt, Roberts, Scharr, Sobel。經過嘗試，我們認為 Sobel 與 Scharr 的效果最佳；以這個分量，我們用來決定稍後的筆劃長度，以及筆劃的方向。
     您可以在參數給定採用不同的 Edge detector。
  2. Smoothing
     接下來，為了讓本身的圖片看起來就像一幅畫，我們先對圖片套用了 MedianBlur；我們採用過 Gaussian Blur、Bilateral Filtering，後來覺得效果差不多，然而 MedianBlur 似乎速度最快，因此我們採用 MedianBlur。
  3. Draw order
     原先我們從上而下，從左至右的把筆劃畫上，然而出來的效果不盡人意，因此我們打亂上色的順序，隨機決定要畫哪一筆劃；並在這個步驟決定筆劃的密度(跳過某些筆劃)，避免重複上色並加速。
  4. Palette
     調色盤的參數若給定 0，表示不使用調色盤，則使用橢圓圓心的顏色作為筆劃顏色；若有給定調色盤參數，則依照參數對顏色進行 Quantize(ex. 設定為 30，則整張圖片每個 Channel 只能使用 30 個顏

色進行著色；每個筆劃挑選與這些顏色最接近的顏色進行上色)。然而，這樣的設定會使一塊色塊整片使同一個顏色(ex. 天空整片藍)；因此我們在決定最接近顏色時，再加入 Random 使色塊不要太單調。

5. Ellipse
   依照第三點決定的 order，用一個個橢圓代表筆劃；橢圓的長度與角度參考第一點的 Gradient(X, Y 分量的斜邊作為長度，arctan 作為角度)；橢圓的顏色則是由第四點的”是否要模擬調色盤決定”。

- GIF to JPG
  倘若使用 python imageio 切 GIF，色彩會與 opencv 所使用的 encoding 不同，造成轉換風格怪異；因此我們使用 Linux 的指令 convert 將 gif 切成一個一個的 frame。

- JPG to GIF
  使用 imageio 將轉換過的 frame 合成一個新 gif。

Difficulty & Solution

- 顏色相近性問題: e.g. 天空一整片，湖面一整片
  針對相近大面積顏色上色問題，我們試著在 Gradient 加入 Gaussian noise 來增加大面積同顏色處的雜訊，希望在這些地方讓他的顏色可以不同些；然而效果並沒有變好，反而有可能使畫面崩壞；最後，我們透過在 color 上加入 random integer 來改善這個問題。

- 速度問題
  原先的目標是希望程式可以 real time 跑完，然而最初的的演算法大張的圖片(3821x2867)需要 1 分鐘。我們透過捨去部分效益不大的功能與步驟，並對其他的功能做優化，才使速度有所提升。最後再透過參數的調整，成功使時間大幅縮短至可接受範圍內(1 分鐘以上->5 秒)。

- GIF and JPG
  原先我們使用 `python Pillow` 中的 `PIL.image` 來切割 GIF，但是編碼與色彩的不同造成有些 GIF 轉出來的圖片有一些問題 (這是一個 open issue)，因此我們最後放棄使用這個套件，使用 `subprocess` 直接利用命令列 call `convert` 這個 unix 的指令進行 GIF 切割成 JPG。

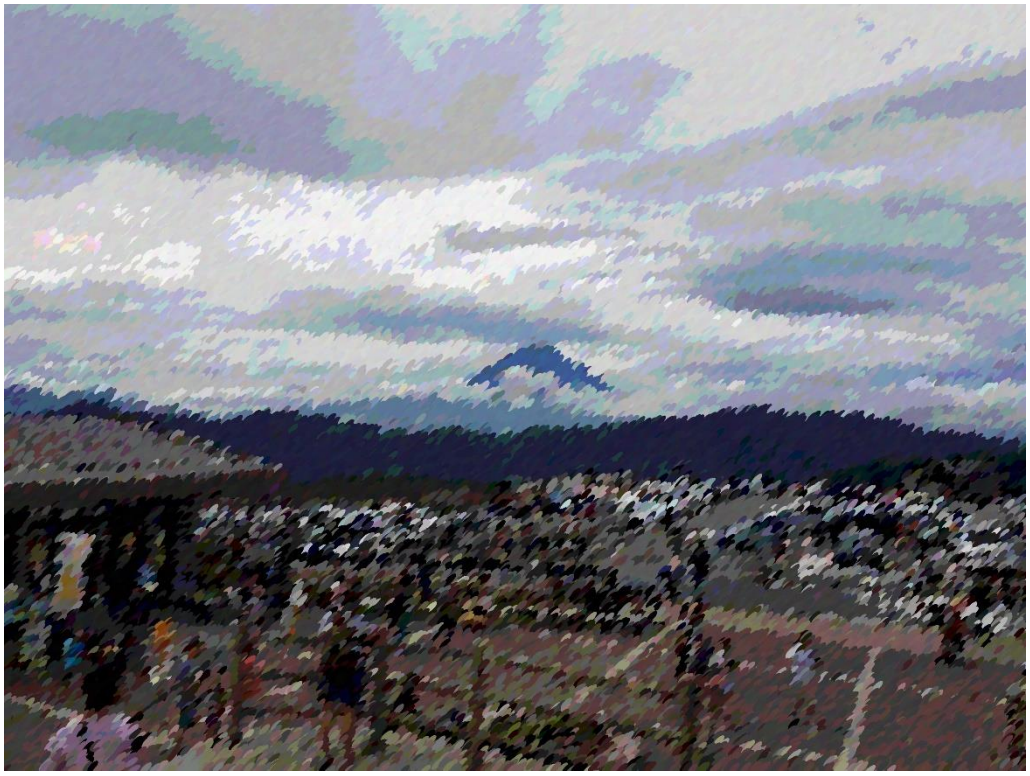
- Edge Detectors
  在後面數頁展示不同的 Edge Detectors 的效果；可以看到，Sobel, Scharr 的表現是比較理想的。

Brush_width: 8, palette: 30, 圖片大小 3821x2867

原圖:

Prewitt:

Roberts:



Scharr

Sobel

Performance

= Split GIF(if GIF) + Gradient + Blur + Draw eclipse + Merge GIF(if GIF)

Filename	Resolution	Brush_width	Time lapse(real)
20180901_134306.jpg	3821x2867	8	5.026 secs.
Obersee.jpg	1920x1080	6	1.566 secs
hallstadt_winter.jpg	1000x667	5	1.107 secs
g1.gif	600x400x19	3	7.494 secs
g2.gif	728x408x123	4	49.547 secs
g3.gif	728x408x144	4	62.462 secs

(Based on Ryzen 1600 stock frequency)

⇨ Not totally real time, but this is fast.

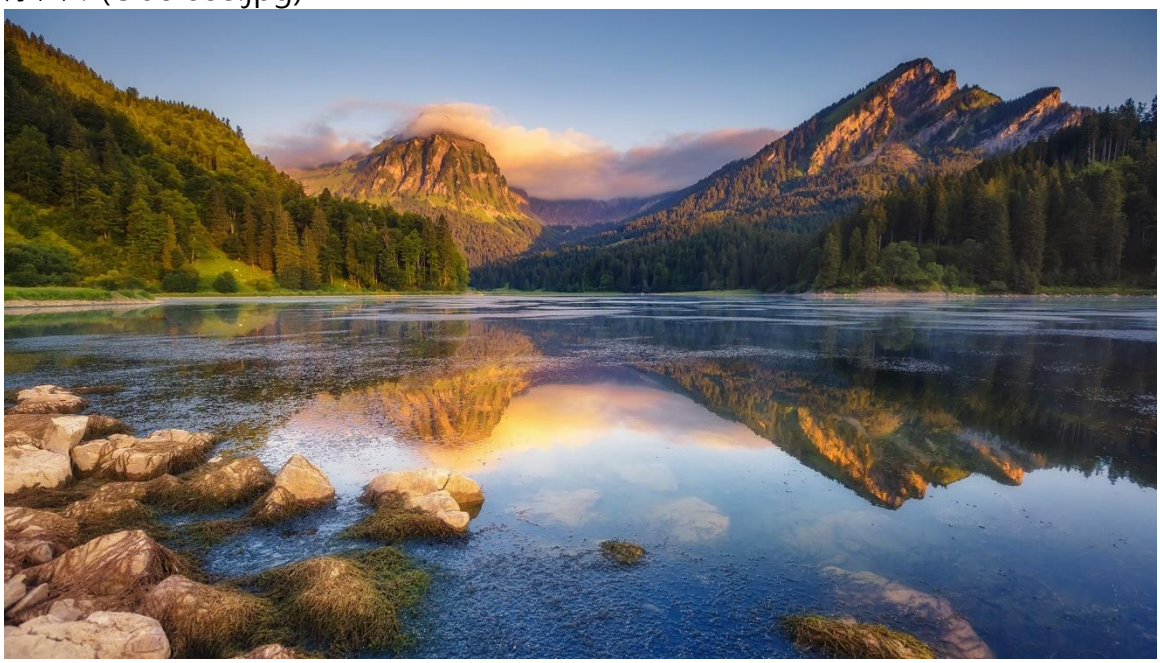
Source Code

https://github.com/shihehe73/Faster-OilPainting

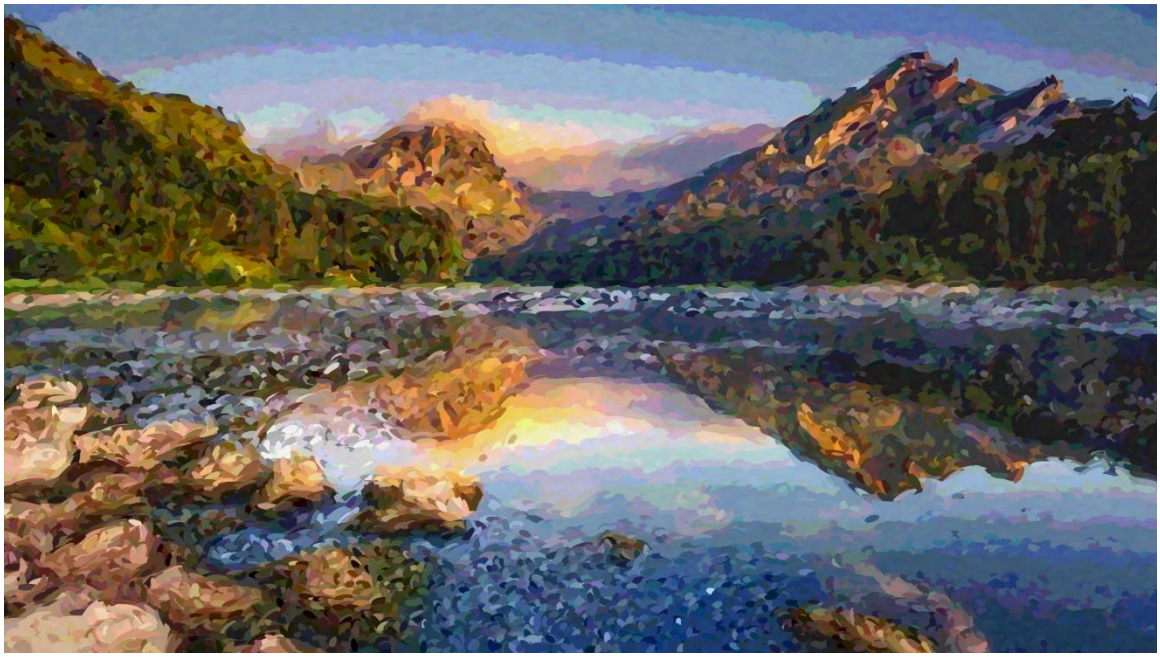
Result

靜態文件敘述的關係，並沒辦法放入 GIF 呈現我們的結果，煩請至我們的 Github repo (Faster-OilPainting/testdata) 觀看我們所轉換的 GIF 範例。

原圖：(Obersee.jpg)

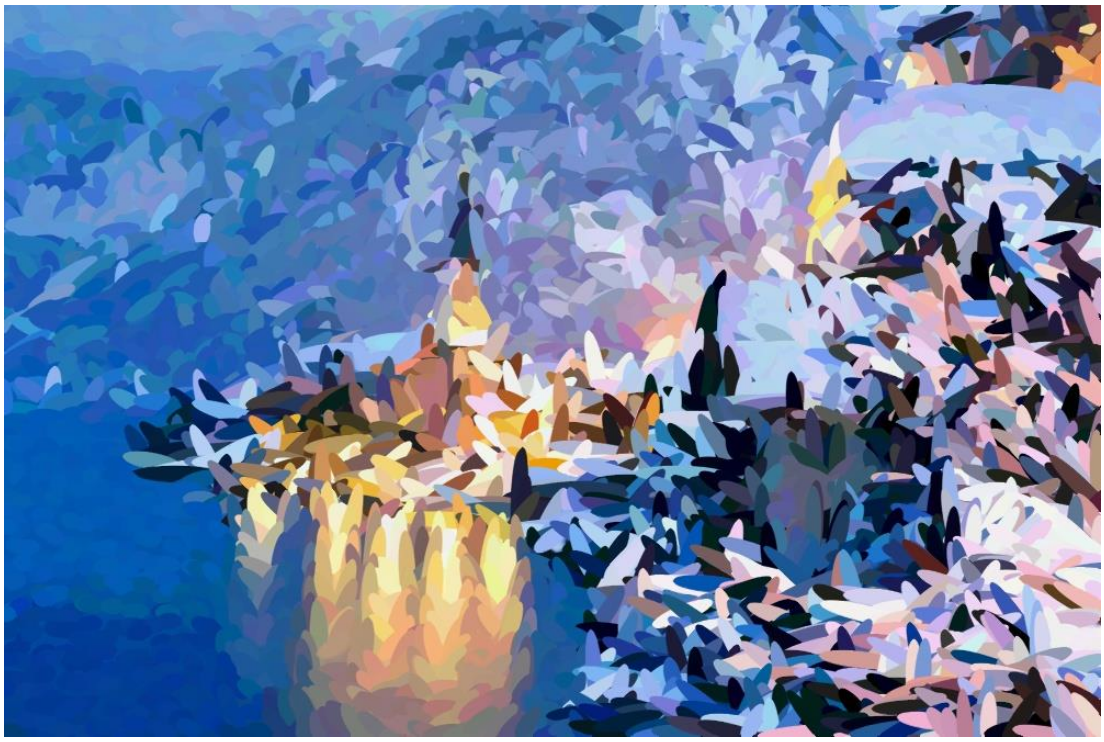


轉換後：



原圖：(hallstadt_winter.jpg)

轉換後：



原圖：(munich.jpg)

轉換後：