

Interactive Computer Graphics

Final Project: Oil Painting Effect

R06944007 楊喬諳 B03901163 鄭煦 R06943006 李洛曦

Part 1: Non-learning based

References

(1) Oil Painting Algorithm

- <https://supercomputingblog.com/graphics/oil-painting-algorithm>

(2) CLD-shaped Brushstrokes in Non-Photorealistic Rendering

- <https://arxiv.org/ftp/arxiv/papers/1002/1002.4317.pdf>

Source Code (我們自己實作)

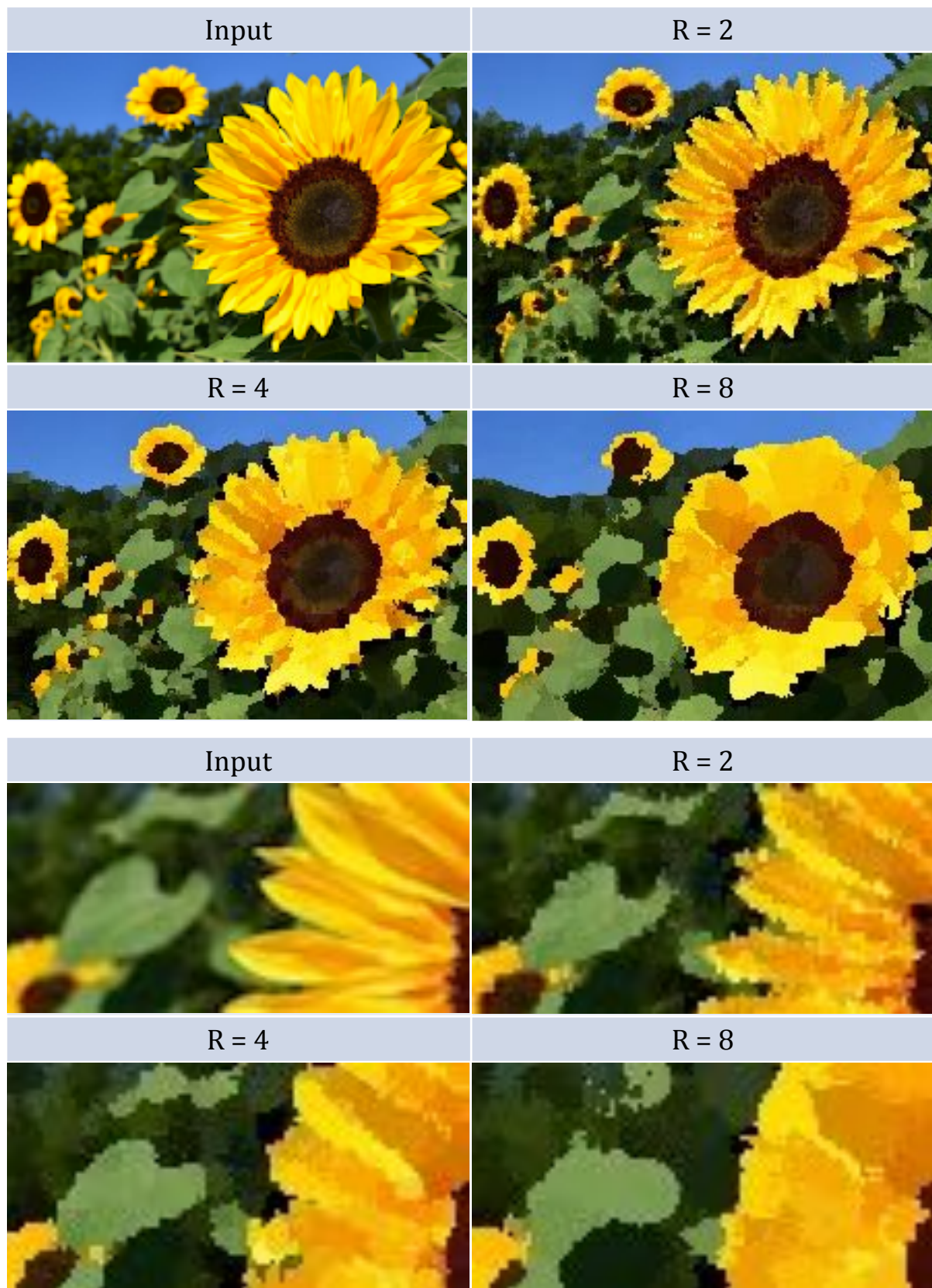
Github: <https://github.com/tartarskunk/OilPainting>

Approach 1-1



觀察油畫，可發現其筆觸較為粗糙，顏色分佈是散出來的。跟真實影像不同，同一個小區塊會是相同的顏色。

對每個pixel，統計其周圍R內的RGB intensity分佈，進行voting，選出最適合的代表。在將此pixel設為其值。



Approach 1-1: Result

Approach 1-2

前述方法的缺點在於，沒有考慮到local distribution的資訊，使得整張圖的筆刷大小R都是固定的，成果看起來不自然。

此方法改進了這個缺點，對於每個pixel都去統計其周圍各個orientation的分佈，數學式如下：

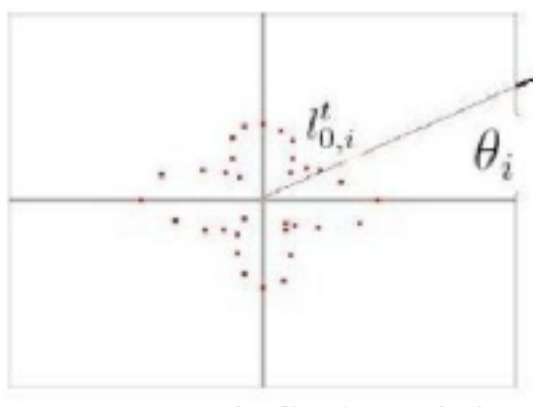
$$\theta_i = i \frac{2\pi}{32}; i = 1, \dots, 32.$$

$$M_0 = \frac{1}{N_x N_y} \sum_{i=1}^{N_x} \sum_{j=1}^{N_y} \tilde{b}(i, j),$$

$$M_{0,\ell}^i(x, y) = \frac{1}{\ell} \sum_{r=0}^{\ell} \tilde{b}(x + \lfloor r \cos \theta_i \rfloor, y + \lfloor r \sin \theta_i \rfloor),$$

$$l_{0,i}^\tau(x, y) = \min \left\{ \ell : \frac{|M_{0,\ell}^i(x, y) - M_0|}{M_0} \leq \tau \right\}.$$

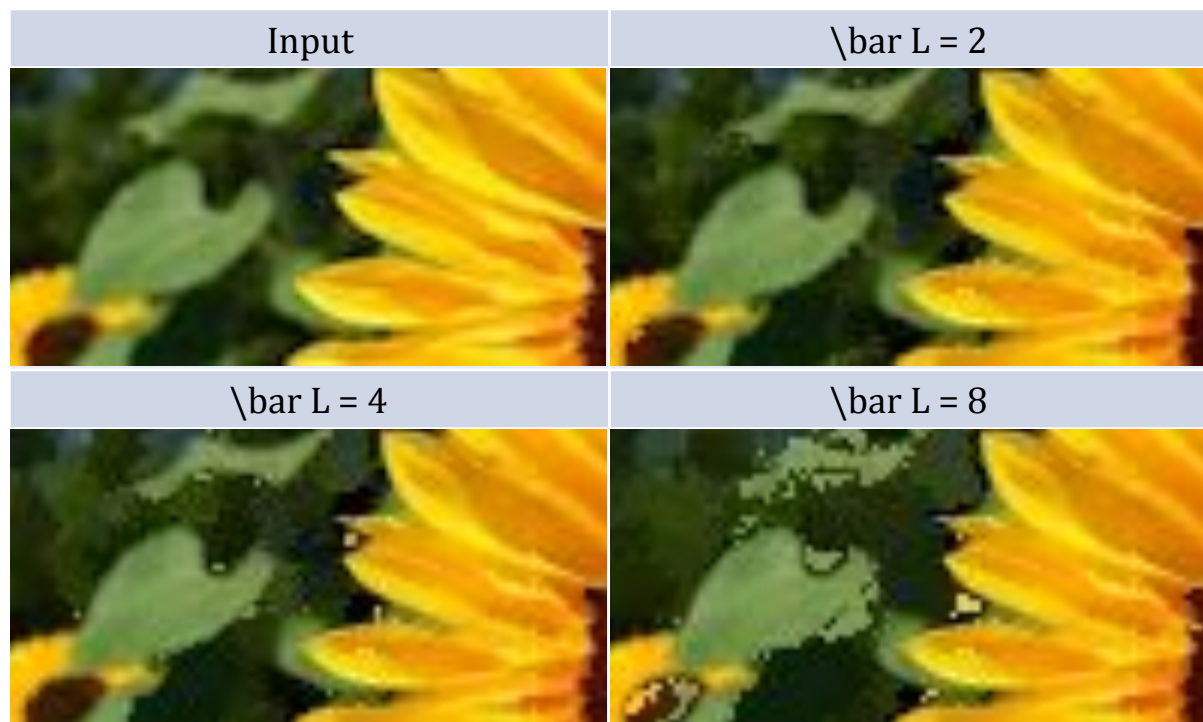
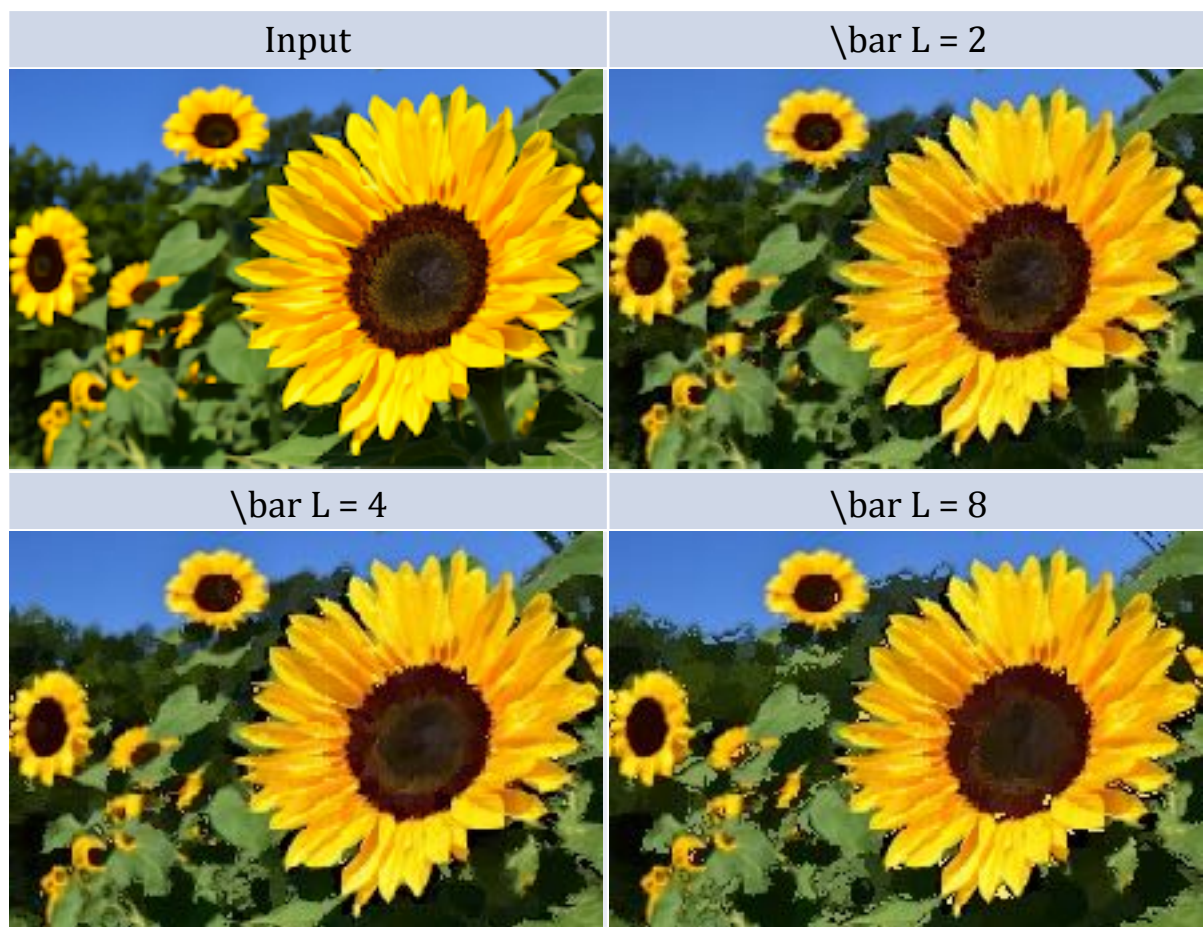
可以得到如下的分佈圖：



統計完 $l(x,y)$ 之後，再利用此分佈計算出對於每個 (x,y) 的最適宜筆刷大小 α

$$\bar{L} = \frac{\alpha}{32} \sum_{i=0}^{31} l_{0,i}^\tau(x, y)$$

上式中的 \bar{L} 為需要外部輸入的參數。



Approach 1-2: Result

Other Results (Higher Resolution)



Part 2: Learning based

References

- (1) A Neural Algorithm of Artistic Style
- (2) Perceptual Losses for Real-Time Style Transfer and Super-Resolution
- (3) Instance Normalization

Source Code (網路上的Open Source)

Gitbub: <https://github.com/lengstrom/fast-style-transfer>

Approach

在影像分類上，CNN通常被拿來過濾出feature map的filter。然而透過其中不同層的reconstruction，我們可以發現在高層次的feature map通常會保留影像的內容，因為其receptive field較小，反之較低層的feature map保留的是影像的風格。

我們使用一個19層的VGG-19 Network並將其分為5個block。

首先取得影像「內容」的feature。輸入一張圖片，欲使通過CNN的圖片與輸入原圖相同，我們訂定Square-Error Loss Function：

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F_{ij}^l - P_{ij}^l)^2$$

求其導數以作back propagation。

$$\frac{\partial \mathcal{L}_{content}}{\partial F_{ij}^l} = \begin{cases} (F_{ij}^l - P_{ij}^l) & \text{if } F_{ij}^l > 0 \\ 0 & \text{if } F_{ij}^l < 0 \end{cases}$$

再來求影像「風格」的feature：取兩層間的correlation形成一個Gram矩陣以消除位置的資訊：

$$G_{ij}^l = \sum_{k,w} x_{kwi}^l \cdot x_{kwj}^l$$
$$i = 1, 2 \dots N, j = 1, 2 \dots N$$

使目標與風格與輸入原圖相同，亦可以使用Square-Error Loss Function，將白噪輸入至網路，調整出最好的權重筆後加兩個error function相加：







$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

利用以上error function，可以得到能輸出最接近原圖風格的CNN。

Result

使用下圖的梵谷畫作作為訓練的target style。



Input	Output
	
	
	

Conclusion

在這次project中，我們實作兩種不同的Oil Painting演算法。

(1) Non-learning based主要依靠prior knowledge，可以直接產生效果，不過計算速度比learning的inference階段慢。

(2) Learning based需要有油畫模板做data，而且需要train的時間。只是做style transfer，並不能直接轉變成油畫風。