

Final Project For Image Processing (view)

動態版報告連結，可以看到影片和 .gif：

<https://shimmering-watch-639.notion.site/Final-Project-For-Image-Processing-view-a3f998296f4a48e6965e538f3be9ad65>

Group members & Division of works

成員名單與分工

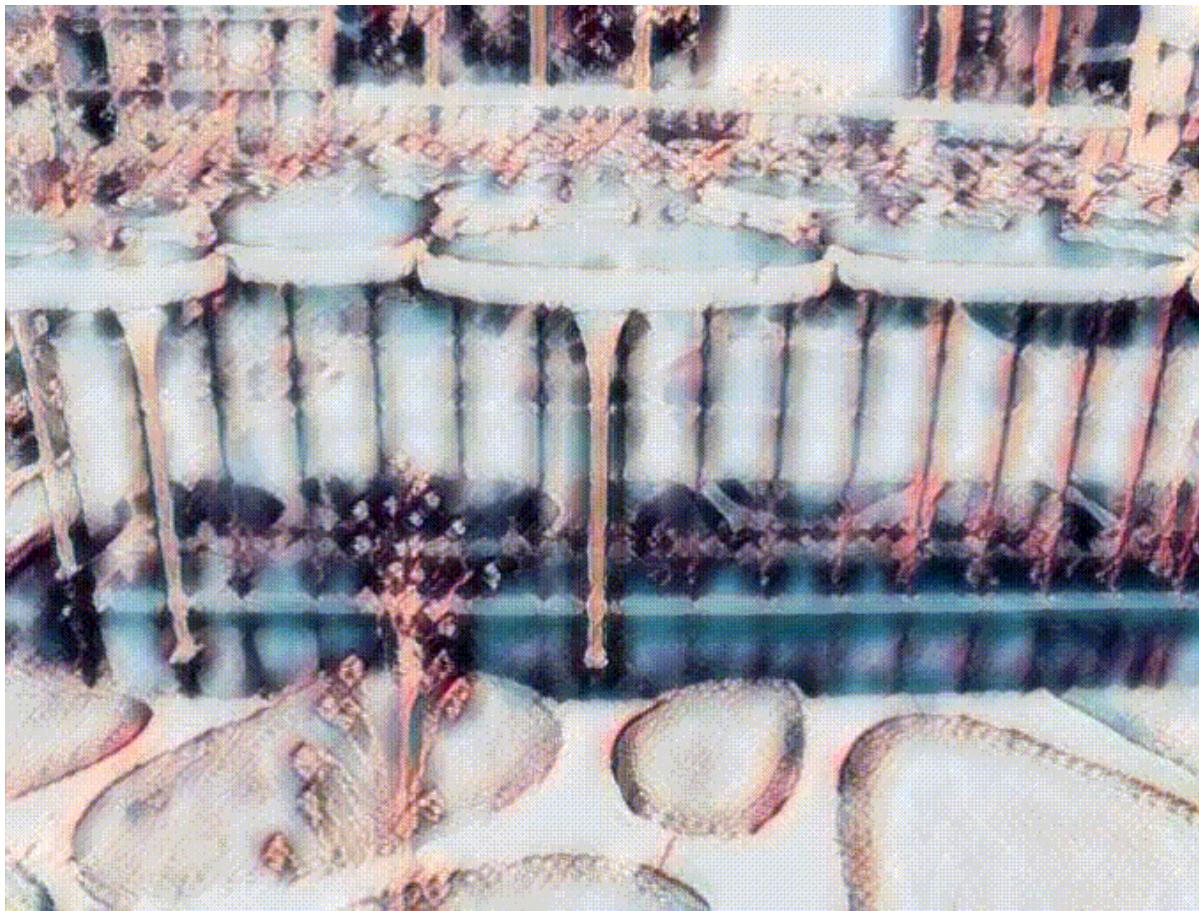
姓名	系級	學號	分工
林俊佑	資工所碩一	r11922114	圖片轉換
蔡歐寶	網媒所博二	d10944007	立體應用
洪世彬	資工所碩一	p11922005	影片轉換

▼ Methods you used

影片轉換

在影像轉換的部分，使用 Linear Transform 已經可以取得不錯的結果，所以我們主要著重在影片的 Style Transfer。

Linear Transform 對每一幀使用相同的轉換矩陣 T ，已經可以有效降低每一幀之間的差異性，但在一些 Style 上的表現仍有進步空間。像是在以下影片中可以看出仍有部分閃爍的情況出現。



目前解決影片閃爍問題有的兩個常見方向，第一種方式是使用光流分析去找到每一幀之間的物體移動方向及距離，再利用此資訊去輔助風格轉換。第二種是想辦法保留更完整的 Content 資訊，透過減少每一幀的形變來降低閃爍的問題。

第一種方式雖然能很好的降低閃爍現象，但是該方式因為需要計算多幀之間的差異性，需要較高的計算成本。並且光流分析的方式對於資料來源的品質有較高的要求，畫質較差的影片容易分析出錯誤的結果。因此我們找了 CCPL 這篇論文，它採用的是第二種方式。

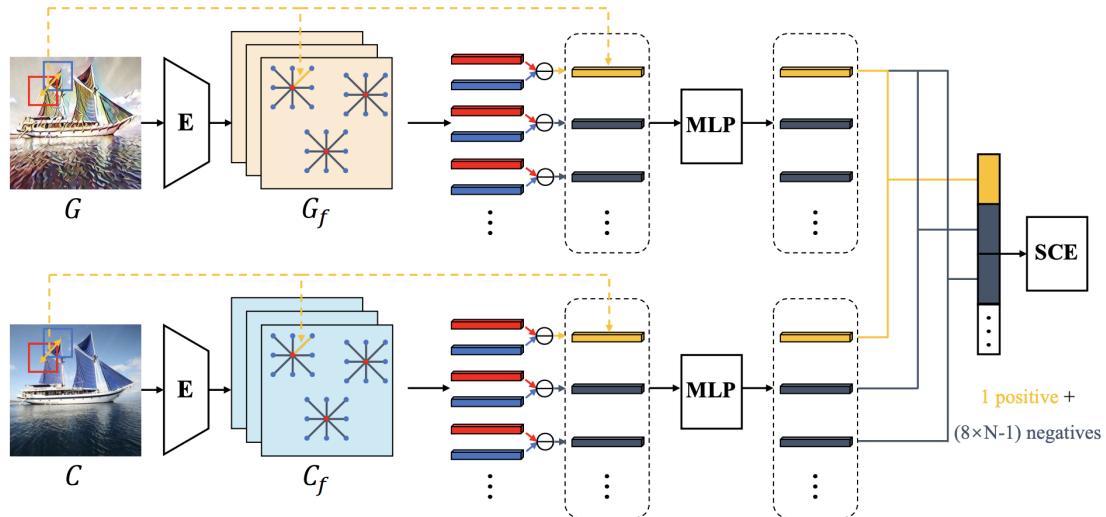
在此引入 CCPL: Contrastive Coherence Preserving Loss for Versatile Style Transfer
作為新的比較方法[2]

在論文中，CCPL 提出了一個它們設計的 Loss Function，目的在減少 Content 的形變。該 Loss Function 並非取代原本的 Content Loss 而是加入訓練流程中。公式如下：

$$L_{\text{total}} = \lambda_c \cdot L_c + \lambda_s \cdot L_s + \lambda_{ccp} \cdot L_{ccp}.$$

CCPL 的設計採用了 Contrastive Learning。Contrastive Learning 是一種 Self-supervised 的訓練方式，透過準備一筆相似資料和 N 筆的不相似資料，讓 Model 學習相似項的特徵。

透過對 Feature Map 做多點的 Sample，並挑選其中一組 Vector 當作相似資料(下圖紅色資料點延伸的黃色 Vector)，讓其他 Vector (其餘藍色 Vector)當作不相似資料來達成 Contrastive Learning 的條件。



而 Loss Function 最終的設計如下：

$$L_{ccp} = \sum_{m=1}^{8 \times N} -\log \left[\frac{\exp(d_g^m \cdot d_c^m / \tau)}{\exp(d_g^m \cdot d_c^m / \tau) + \sum_{n=1, n \neq m}^{8 \times N} \exp(d_g^m \cdot d_c^n / \tau)} \right],$$

轉換後的影片可以明顯看出 Content 的保留更完整，而影片的閃爍現象確實有減少。



圖片轉換

在圖片轉換中我們使用了來自 LinearStyleTransfer[1] 的模型進行轉換，並且做了以下兩種方向的嘗試

- 利用 style transfer 來製造相片的景深與散景
- 利用將圖片切割來高速生成高畫質圖片

在製造相片的景深與散景的部分，我們會準備多張圖片分別帶有各類景深/散景的特徵作為風格圖，查看是否可以用 style transfer 抓出裡面的特徵，也會使用該模型的不同特徵抽取層來比較效果的明顯與否。最後還會試著依照當前主流計算攝影的演算法，把景深與散景套用在風格轉換中。

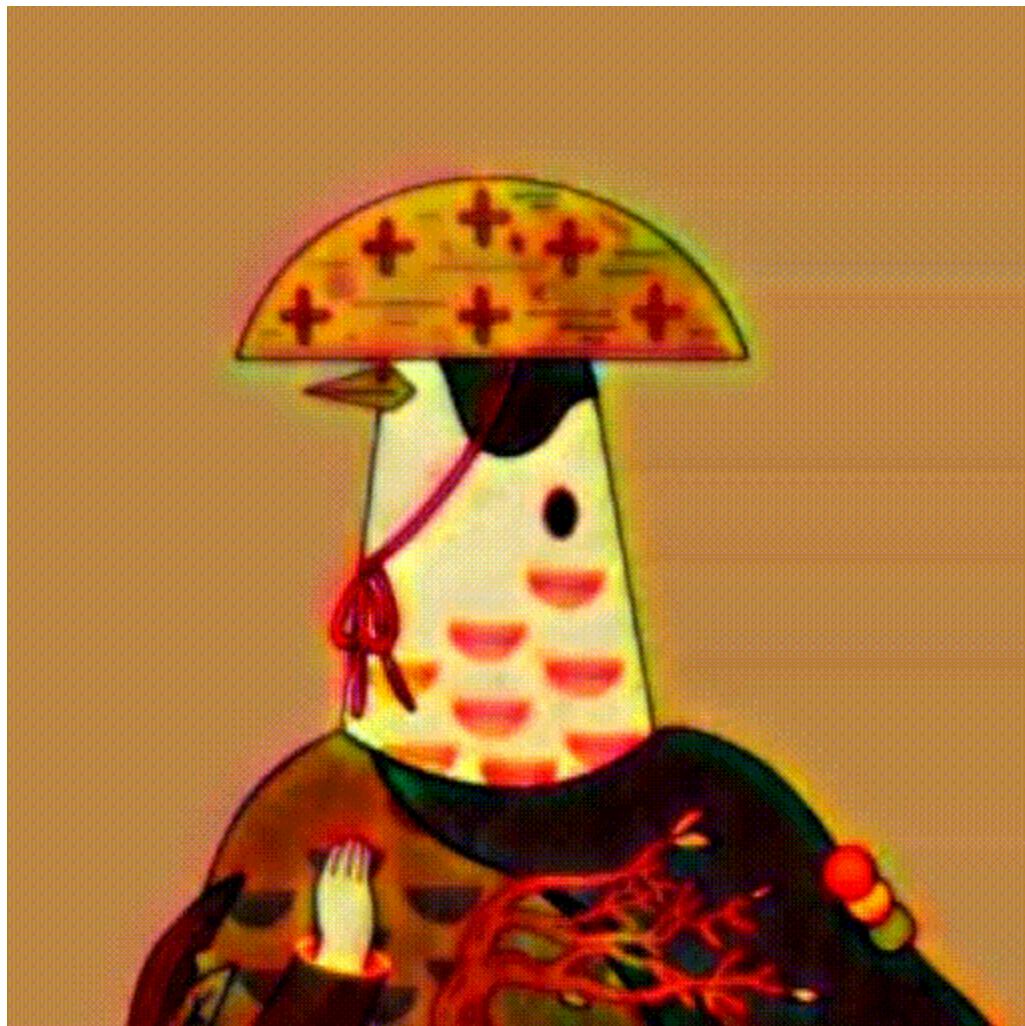
在利用圖片切割生成高畫質圖像時我們會先把原圖放入模型生成線性轉換層，再把圖片切割成 9 小塊放入線性轉換層平行運算，比較呈現出來的效果以及在各類硬體的計

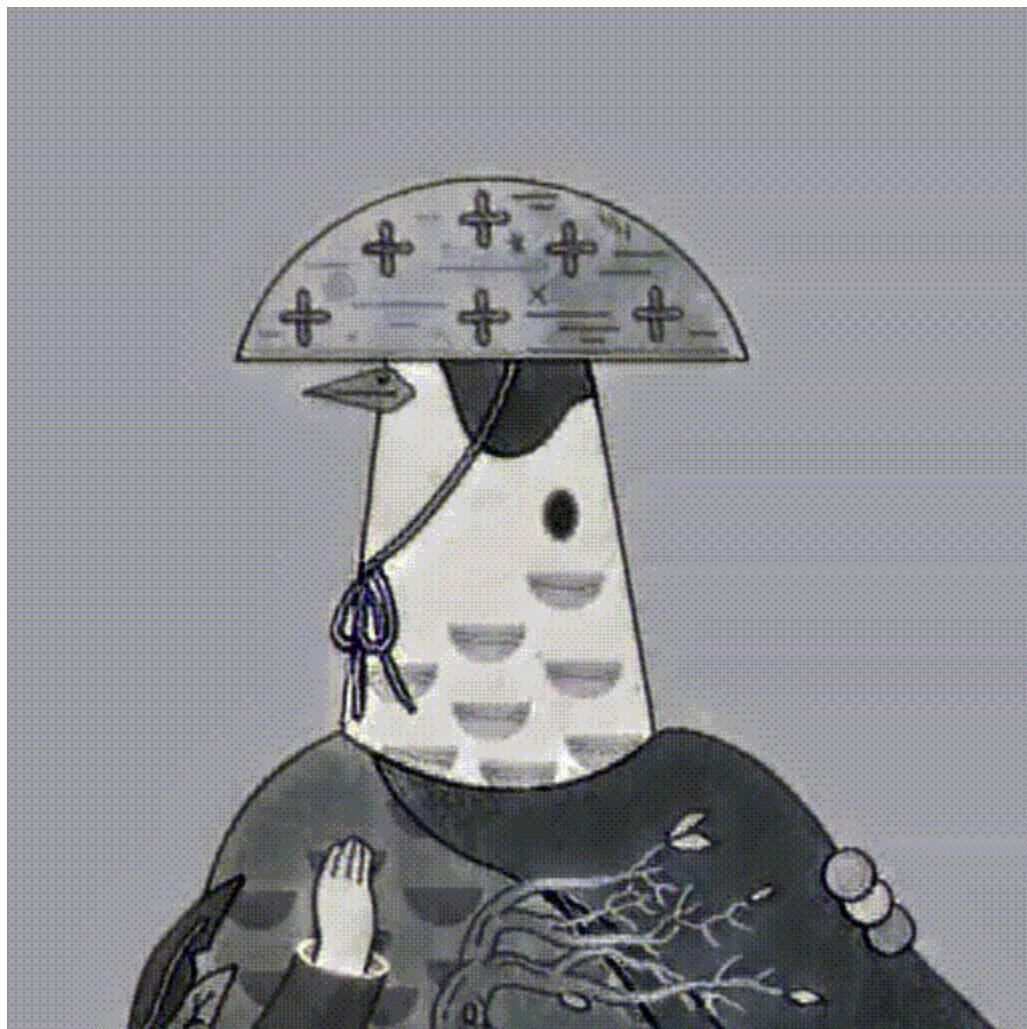
算時間

▼ Results

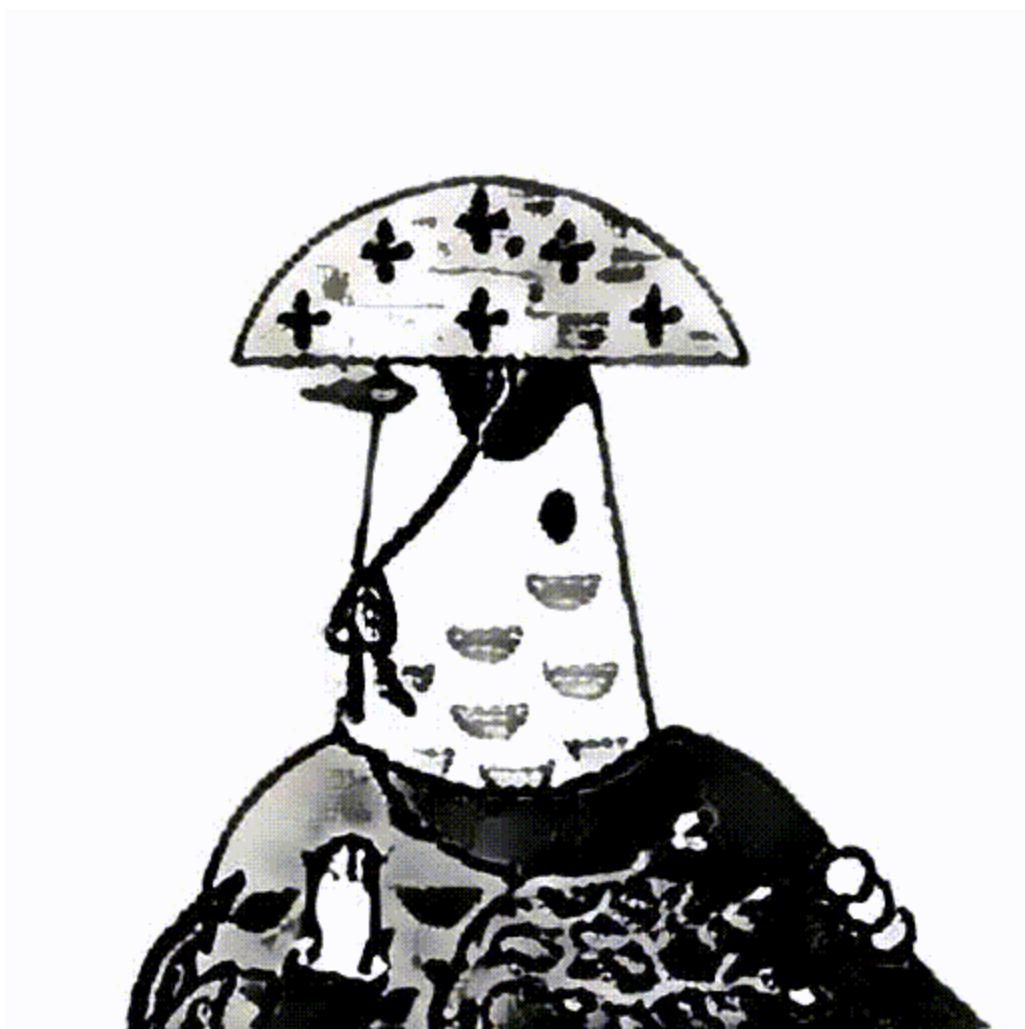
Final Project 相關影像轉換結果

武士雞影片



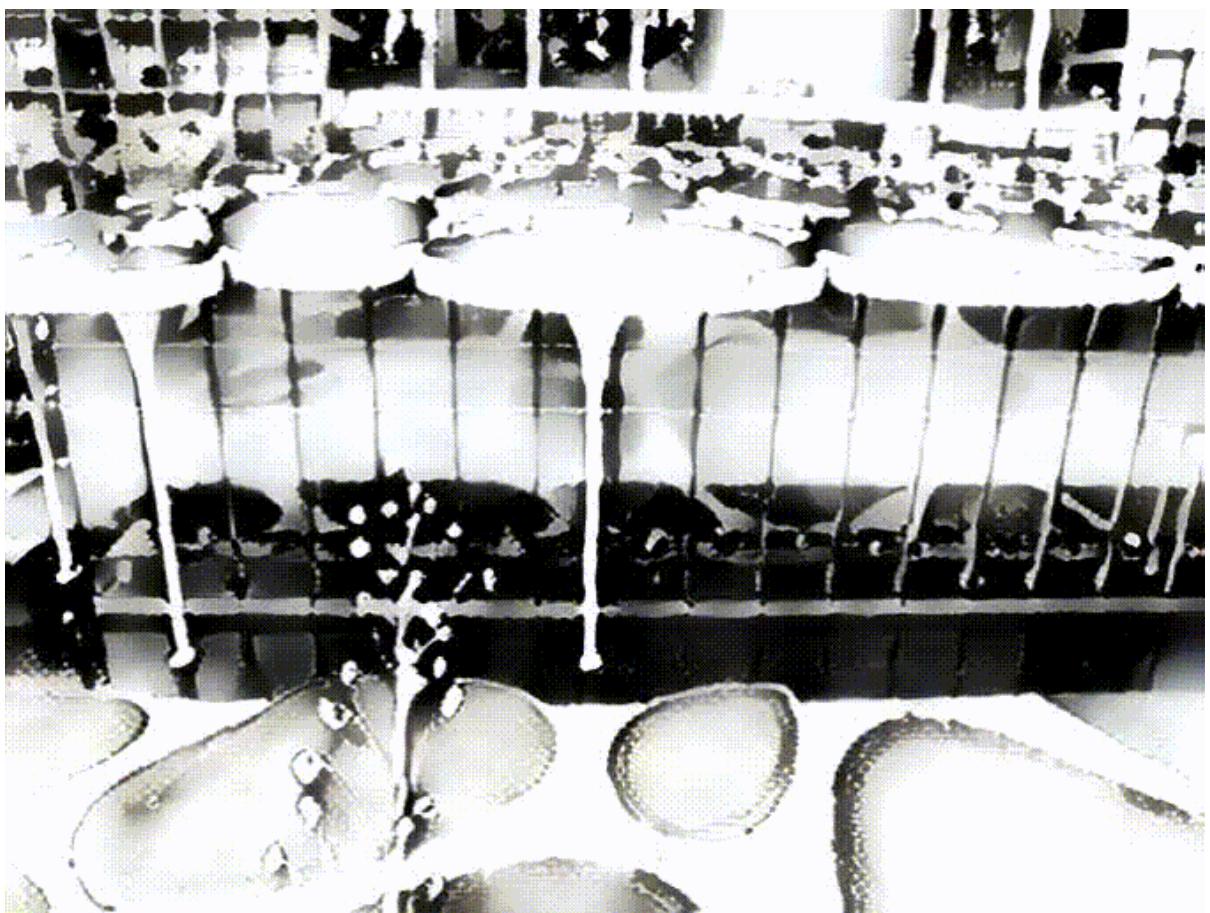






空拍影片

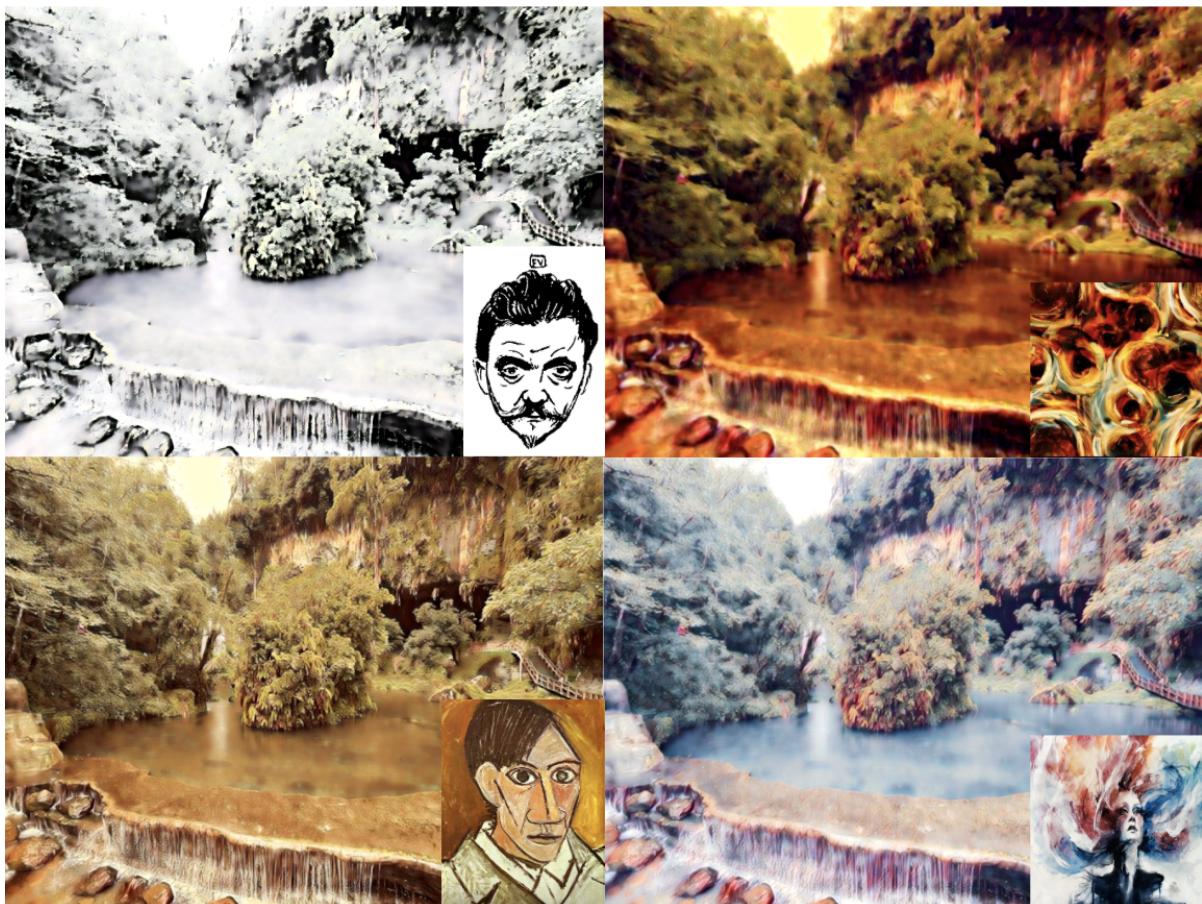






武士雞圖片



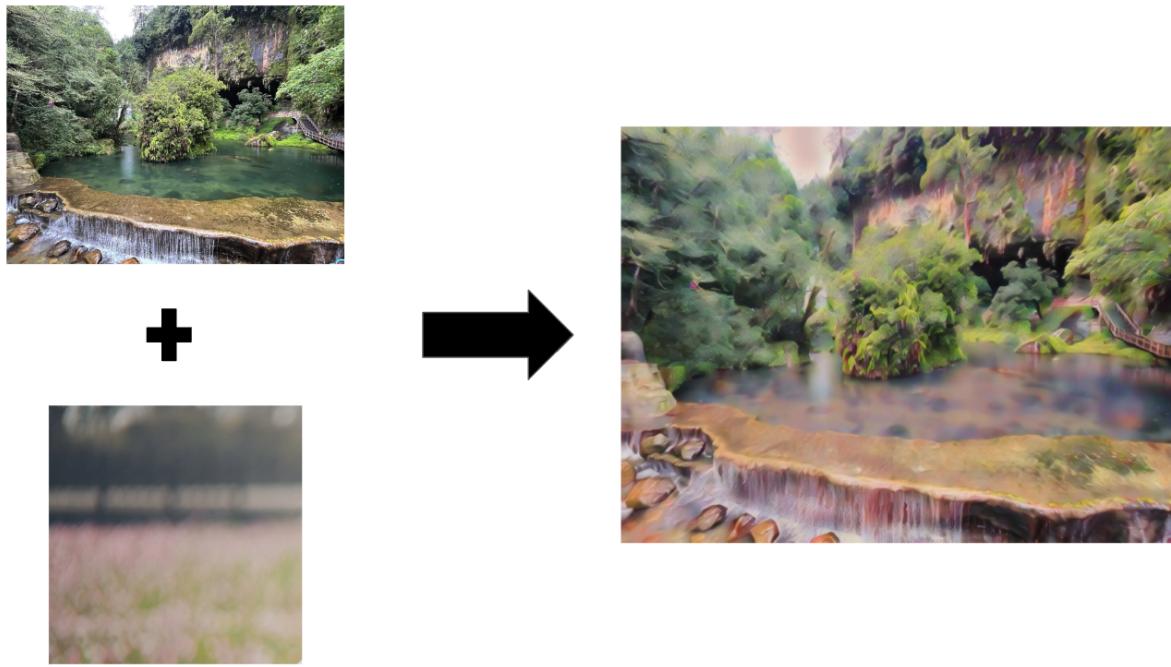


▼ Application

圖片轉換的應用

前後景虛化效果的風格轉換

首先將一個典型意義上具有光學缺陷的景深圖放入得到以下結果

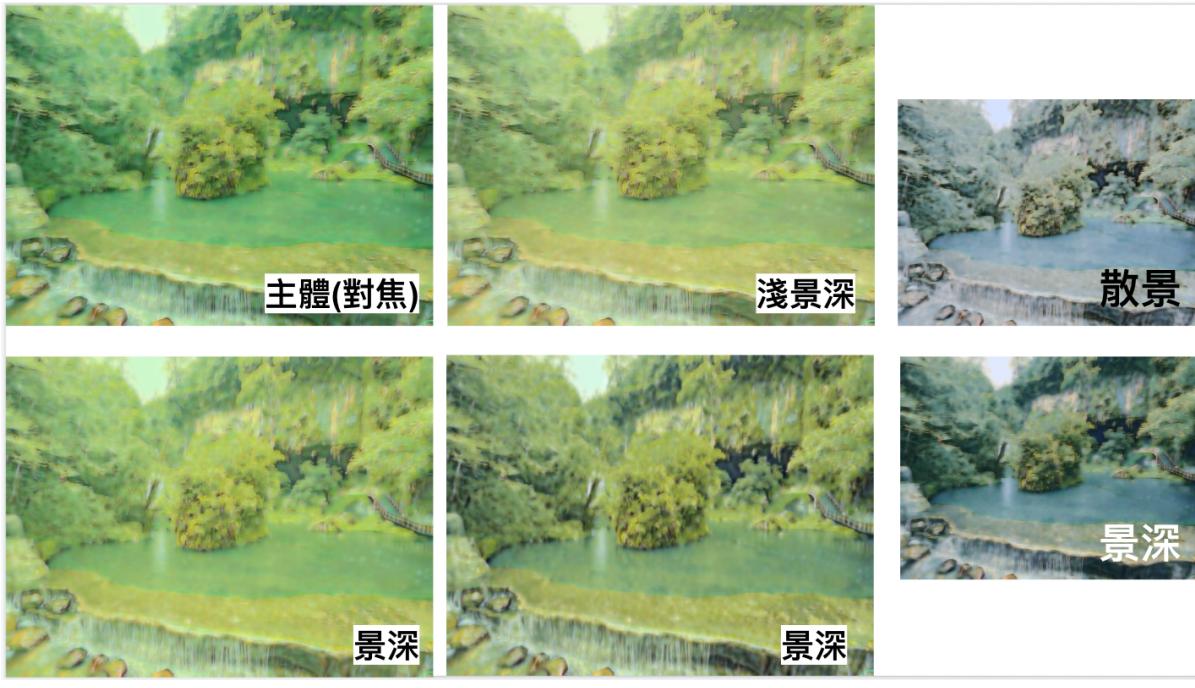


範例轉換結果

可以發現圖片上確實出現了風格圖中特有的質感, 於是我們找了顏色近似的各類風格圖, 想看看風格轉換模型是否可以成功的抓出不同種類的特色



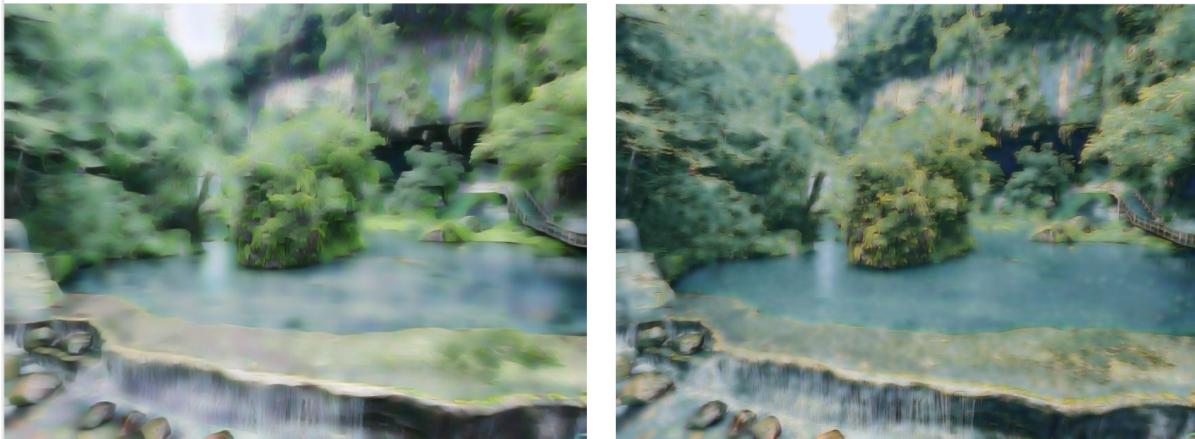
風格圖



轉換結果

可以看到, style transfer 基本上只可以抓到一點點的特徵, 導致各個風格, 細節上的差異都基本無法展現。

因此我們轉而去比較不同特徵抽取層的效果



比較不同特徵抽取層

可以發現抽不同層會把紋理的明顯程度放大和縮小, 在實際面反而很近似不同的景深所呈現的效果

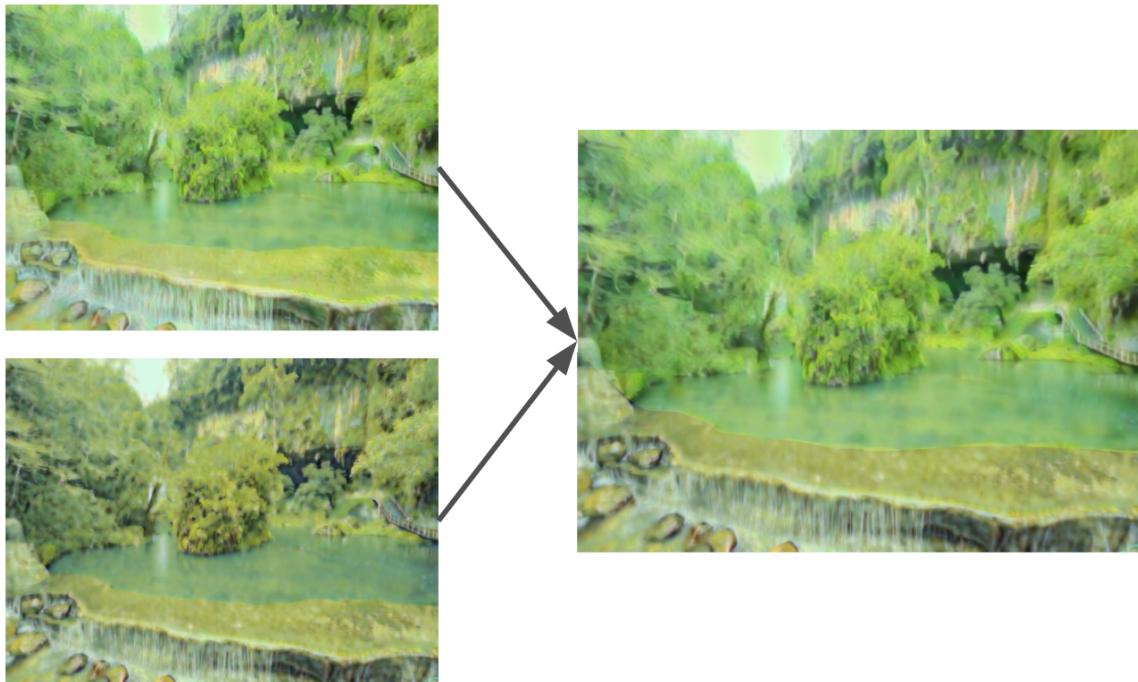
之後我們試著比較散景轉換的結果



比較散景轉換(左)和景深轉換(右)

可以發現這個 style transfer 基本上完全無法轉換散景的風格只不過是讓圖片多了一些粗粗的紋理

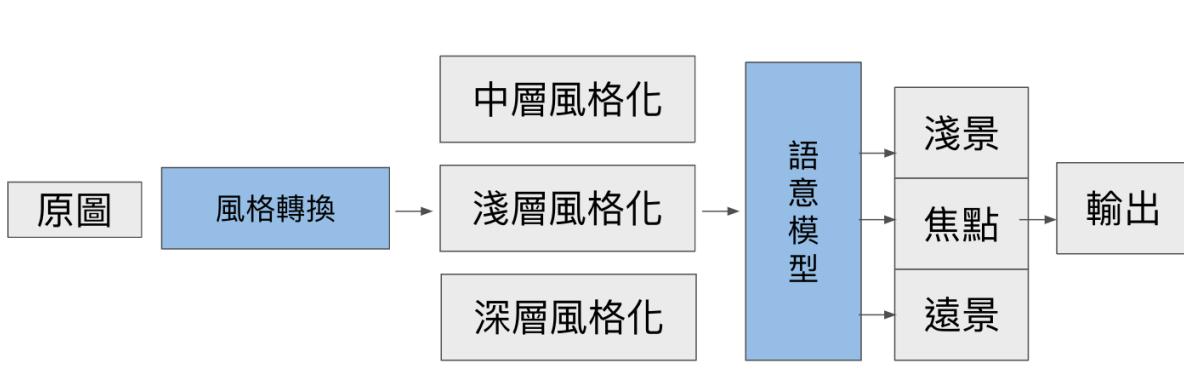
最後，我們試著把圖片依照實體切割用不同深度的 style transfer 結果組合，想實作具有前後景虛化效果的風格轉換。得到下方的圖片作為成果。



兩張圖片依湖水為界被合成

可以感受到焦點明顯落在前方的石頭上，遠處漸漸變得模糊

以下提供實作上的架構



快速生成高畫質圖像

實際操作下，我們先利用模型把完整的圖片做一次轉換，生成線性轉換矩陣

接著把完整的圖片切割成 9 塊，分別平行調用線性轉換矩陣進行轉換

最終把各自轉換結果合成回原圖，產生結果如下



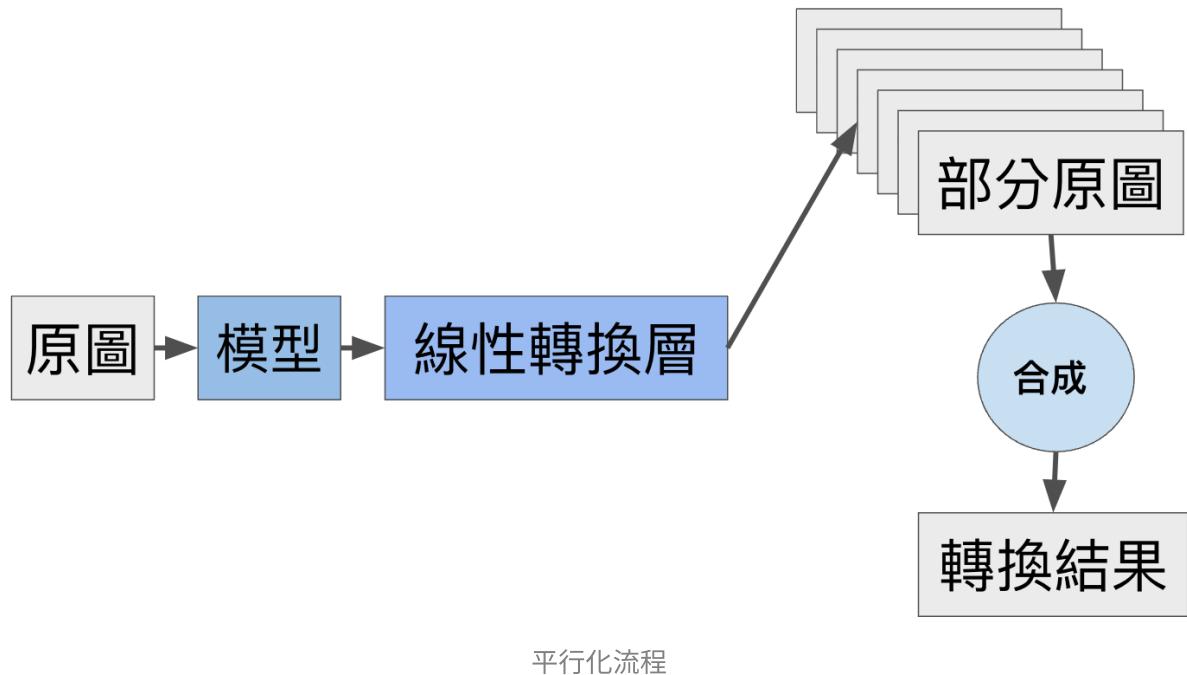
原圖(左), 利用高速拼接的結果(右)

分別利用 apple mac pro 晶片, 和 colab 上的 Tesla T4 顯卡生成以上成果, 分別執行時間如下

其中 M1 不支援硬體加速, T4 可以調用 CUDA API 達成硬體層加速

	單張邊長 1024 的圖片生成時間	單張邊長 3072 的圖片生成時 間	平行化後理論速度
M1 pro	4s	22s	8s
Tesla T4	0.1s	0.1s	0.2s

平行化後裡論速度是用以下平行化流程來過估算, 且假設套用線性層生成圖片的時間和生成第一張線性層的時間相同



360 application

360 views application

如果2D圖像可以--->那360的表現？

由於style transfer在具有高度扭曲的圖像上，可能有邊界模糊與透視辨識問題，

我們利用本篇論文的各種style，發現高彩度與高細節圖像與360 panoramic的content結合效果較理想。以下為實驗的列表：

立體視覺的表現



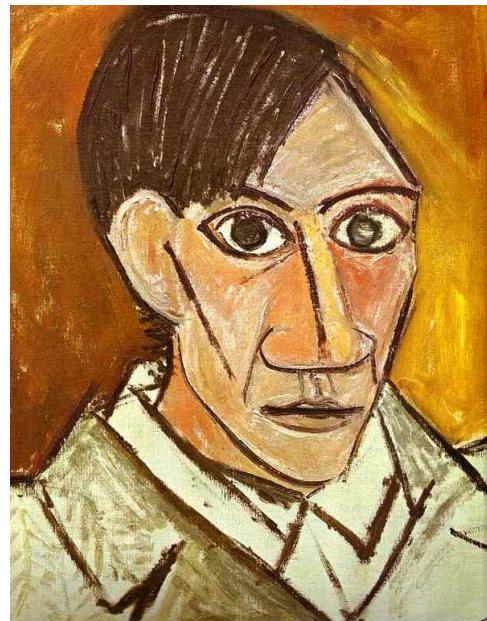
最佳成果—成像仍維持邊緣銳利，單一物件辨識度高，高 affinity！



素描表現：拉高對比強烈，失真地板質地無法表現地面景深質材變化（失去立體度與主角影子）



style image左上圖畢卡索，右上圖陳澄



而原本最令人擔心的上下shrink收縮天地，從上圖可看出，在風格轉化後仍然具有content的辨識度。因此未來目標達full 4K解析度的360在style image sequence的表現令人期待！

可連此用YoutubeAPP觀看360效果：

下方連結為style image 陳澄波淡水圖像；content image為我們試作的武士雞全景

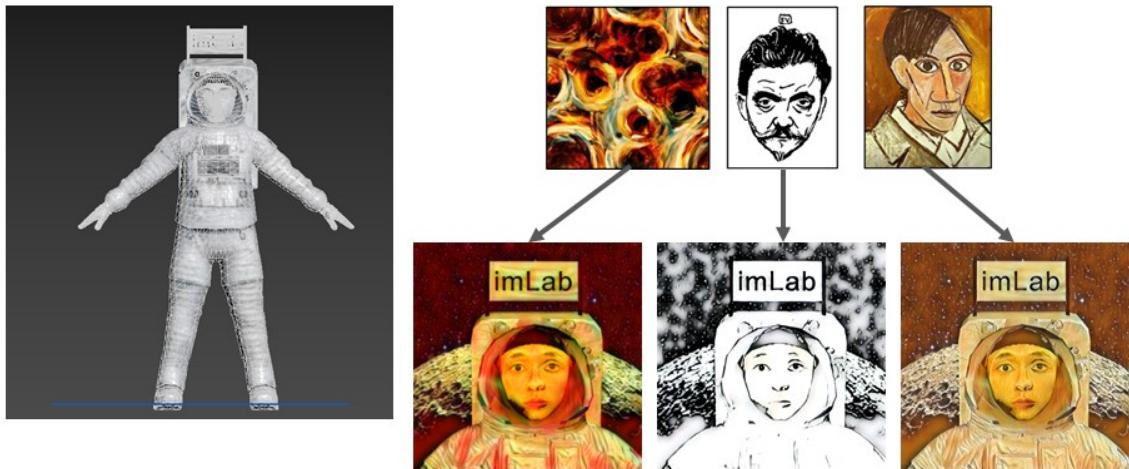


3D application

如果2D圖像可以--->那3D算圖後+style transfer的表現？

首先將3D模型算圖出單圖成為content image後，再加入style image後的試驗效果，其實不太理想，因為模型本圖的美術節理與質感不夠豐富：

如果2D圖像可以--->那3D算圖後+style transfer的表現？



接著我們試想回歸到材質本身結合style image作法的可能：

3D界的Style transfer = 卡通演算？

卡通渲染（英語：Toon Shading）是一種非真實感繪製（NPR），旨在使電腦生成的圖像呈現出手繪般的效果。

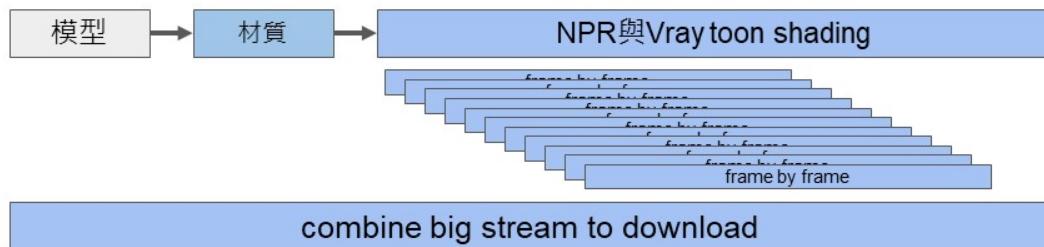
Non-photorealistic rendering

物理的渲染（Physically Based Rendering，PBR）的相對

(NPR) 高度風格化的算圖

- 梯度漫反射（卡通風格陰影）
- 局部高光
- 邊緣光
- 描邊

3D算圖image+Style transfer

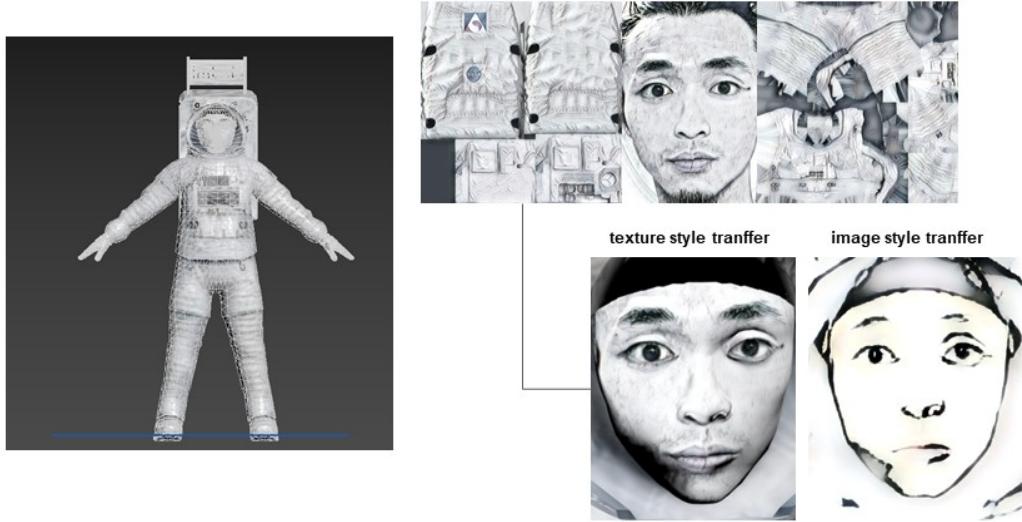


texture Style transfer再3D算圖



由於卡通算圖的引擎常不適用於網路webGL，亦很難用PBR通道設定，因此若要維持高度通用的PBR材質，又有style image風格化的效果，應該從PBR可成像的方向去思考。

如果2D圖像可以--->那3D材質先行style transfer的表現？



https://s3-us-west-2.amazonaws.com/secure.notion-static.com/b2f797b6-bed1-4522-8485-443e316c9b4e/PSM_V03_D776_Lunar_landscape.mp4

上方可觀看影片效果。

texture style tranffer



image style tranffer



再看一次放大細節，左圖為材質風格轉化後的realtime效果；右圖為圖像style image後的效果，確實失真了許多細節。因此未來風格轉化可應用於遊戲世界的材質應用，成為使用者自定義各個模型材質，而非用算圖引擎直接運算全世界的風格。

▼ Discussions

影片轉換

- What did you find?

在影片轉換的部分，雖然 CCPL 轉出的影片比較沒有閃爍的問題，但保留過多 Content 資訊也造成風格部分較缺失。就成果來看，使用 CCPL 的影片比較像是套用了一層濾鏡，並沒有非常好的將 Style 中的筆觸、抽象表達等轉換到最終影片中。

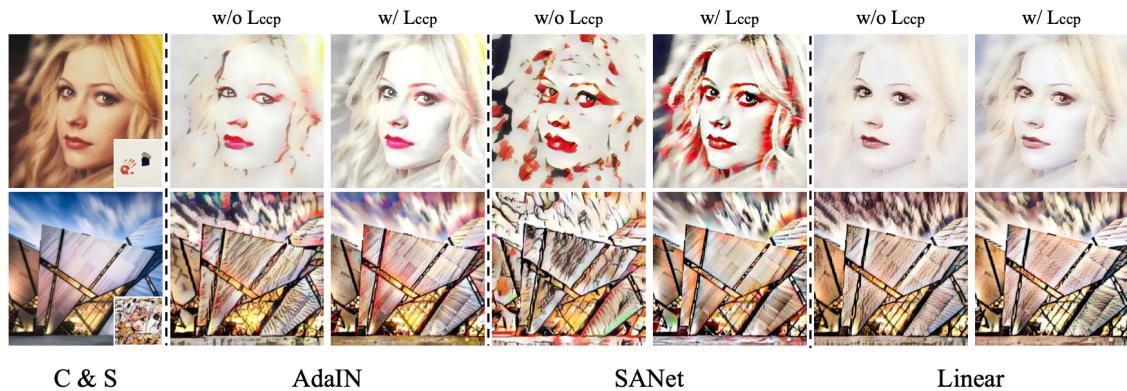
- What could be better?

我認為影片轉換仍有蠻大的進步空間，即使 CCPL 已經降低閃爍的問題，但實際上仍存在。且在前一個問題中提到的，因 CCPL 的設計關係，Style 的轉換並不完整。

另外，雖然本次提到的兩篇論文都是任意風格轉換，但實際在轉換上，仍舊有各自更擅長的 Style 類型。或許未來可以針對不同風格類型採取不一樣的轉換方式，讓 Model 在不同的 Style 皆有不錯的表現。

- What's the difference between the methods?

CCPL 與 Linear Transform 在轉換方法上其實並無衝突，前者著重於 Loss Function 的設計，而後者則是轉換的 Model 本身。CCPL 的作者甚至有將他們設計的 Loss Function 套用到 Linear Transform 的 Model 上使用，也取得了不錯的效果。(見下圖)



圖片轉換

- What did you find?

在各類景深的轉換中發現, style transfer 是可以抓取到紋理的特徵, 但沒有辦法呈現太細節的差異。至於散景因為牽涉到很明確的光學特性, 不是單純的放入紋理, style transfer 幾乎無法處理此類轉換。

另外在分割圖片生成高畫質圖像的部分, 可以利用平行運算大幅提高運算速度, 但速度提升僅限於未做硬體加速的裝置, 在支援硬體加速的裝置上運算反而會降低運算速度。此外, 雖然模型具有高親和力的特色, 但用在各個小分割圖上, 還是會因為內容差異導致色差。

- What could be better?

在景深轉換中, 可以利用抽取不同特徵層來強化紋理, 藉此做出不同效果, 再配合語意引擎, 可以實踐具有前後景虛化的 style transfer

在分割圖片生成高畫質圖像中, 我們發現利用此模型生成不同畫質的圖片, 具有依照畫質執行時間呈現線性成長的特徵(在不可考慮其餘運算, IO 讀寫下), 所以理論

上可以把圖片拆成非常多片來加速。

- What's the difference between the methods?

在各類景深的實驗中，我們把計算攝影的理論帶入了風格轉換，也證明了其可行性，傳統風格轉換只聚焦在單一圖片的轉換，我們提出的解決方案則說明了一種方法，可以利用組合多張不同特徵抽取層的圖片達到使風格轉換的結果有焦點，前景深，後景深的差異。

在分割圖片生成高畫質圖像中，我們嘗試了一種平行運算的方案，使沒有硬體優化的設備也可以擁有一定的效能提升。

References

1. <https://github.com/sunshineatnoon/LinearStyleTransfer>
2. <https://github.com/JarrentWu1031/CCPL>
3. <https://shimmering-watch-639.notion.site/Final-Project-For-Image-Processing-view-a3f998296f4a48e6965e538f3be9ad65> 動態版本報告