國立交通大學資訊工程學系

資訊專題競賽報告

Image Processing and Simple Effects UI

專題題目說明、價值與貢獻自評（限100字內）：

主要是想實作圖片特效，包括利用「影像深度預測」完成的照片流體特效, 文字移動特效，以及「計算消失點與圖片視角轉換」完成的字體沿消失線移動特效。最後將以上特效整合成一個可以讓使用者操作的介面。

專題隊員：

學號	姓名	手機	E-mail	負責項目說明	專題內貢獻度(%)
0816021	吳佩怡	0966423770	20714betty@gmail.com	Text moving along vanishing line effect	50%
0816064	吳中赫	0908411239	a0908665830@gmail.com	UI, Fluid Effect, Text Move Effect	50%

本專題如有下列情況則請說明：

1.為累積之成果(含論文及專利)、2.有研究生參與提供成果、3.為大型研究之一部份。

無

相關研究生資料（無則免填）：

級別年級	姓名	提供之貢獻	專題內貢獻度(%)

【說明】上述二表格之專題內貢獻度累計需等於100%。

指導教授簡述及簡評：

佩怡與中赫同學，在確定方向為視訊特效合成後，即投入不少心力在文獻收集、新穎的模型訓練測試，可惜受限於設備，新模型的訓練時間過久。但他們也很快的轉換方向，盡速地朝向利用深度等特徵進行單張影像的特效合成，在有限時間內盡力完成可demo數項特效的系統。

指導教授簽名：林奕成

中　華　民　國　一　一　一　年　　月　　日

專題摘要

一、 關鍵詞

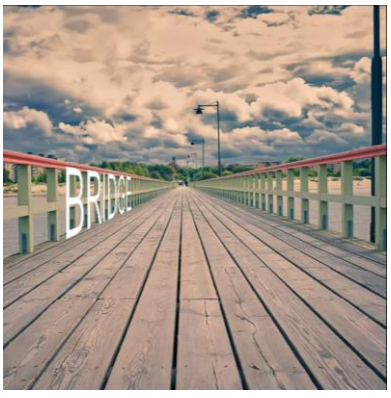
Depth Estimation
Vanishing Point
Perspective Transformation
UI
Text Move-On Image
Fluid Effect

二、 專題研究動機與目的

一開始是想實作可以讓文字在影片中跟著移動的特效，原先這種特效都必須由剪接特效軟體做長時間的調整，並人工製作。所以我們利用現有的 Video Consistent Depth Estimation 實作上述影片特效，但原先影片特效的部分耗費了大量時間以及機器運算資源，所以我們改將目標著重在照片特效上。利用現有 Image Depth Estimation 實作一些相關的延伸特效；另一項是消失點相關特效，我們希望讓圖片疊加使用者指定文字後做出有空間感的移動。最後將上述兩項以圖片連續播放的方式製成影片，從 2D 延伸出 3D 的視覺效果。

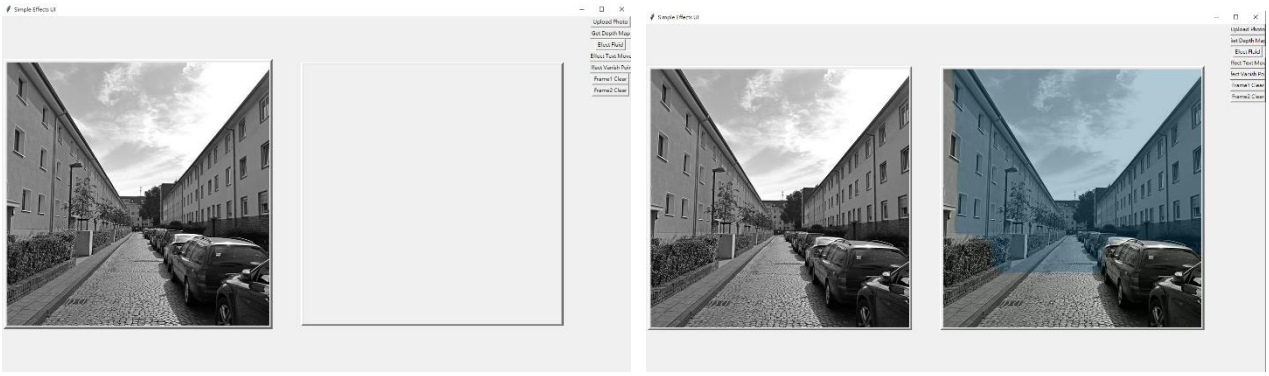
三、 現有相關研究概況及比較

現有的動畫剪輯軟體(如 Photoshop AE)可以達到我們實作出的類似效果。 但 AE 的影片特效需要使用者一幀一幀的調整，如文字部分需要手動定格位置、流體特效需要在各幀畫面上手動進行區域上色。而我們的 UI 讓使用者可以輸入簡單參數就直接生成短影片，將製作特效的過程簡單化，並且應用了深度預測模型、透過演算法計算消失點，做出的效果和人工調整畫面貼近現實的程度差不多。附圖為兩者在文字貼牆特效上的效果比較，左圖是使用 AE 手動調整，右圖是經過演算法計算消失線並經過矩陣轉換做出的效果。



四、 專題重要貢獻

相較於 Photoshop AE、PS 等特效軟體需要對每個畫面做手動微調，我們的實作成果在製作簡易特效方面可以讓使用者更快速便捷的產出所需的影片效果，僅需選擇圖片、輸入文字即可輸出所需的特效短影片。並且目前的軟體較少應用到深度模型，因此在這個方面我們實作出的效果會比人工調整畫面更加精準。附圖為我們簡易的操作介面，可以從裝置的所有位置上傳照片到 UI，若需要下載完成特效的影片也很方便。



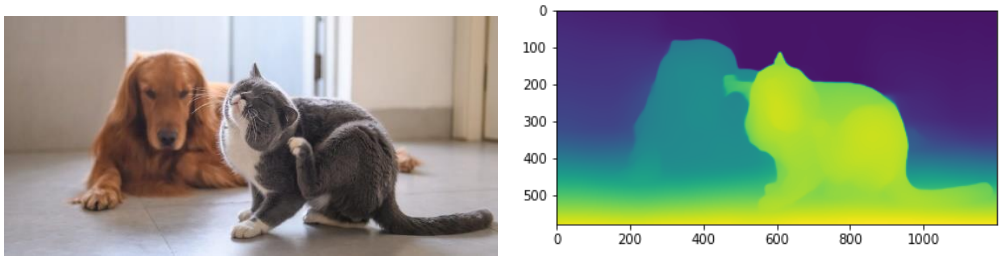
五、 團隊合作方式

一開始以閱讀電腦視覺相關主題的多篇論文為主，經過對論文中的方法做一些嘗試和討論後，最後決定著重在對單張照片做特效渲染。實作分工的部分由一個人著重在利用照片深度值模型延伸出一些特效，另一個人主要實作消失點計算與相關文字特效。期間會定期討論並改善各自 project 中的缺失，如照片播放幀率沒有設定好導致看起來卡頓、流體特效的上色區域透明度改善等問題。各自負責內容皆完成後便透過 python GUI 合併成一個方便使用者統一操控特效的介面，也能更清楚的呈現成果。

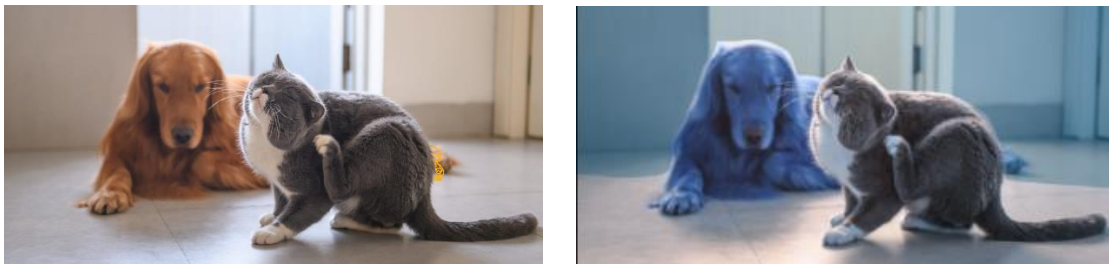
六、 設計原理、研究方法與步驟

初期以閱讀電腦視覺相關的各式主題的多篇論文為主，例如：consistent video depth estimation, camera pose estimation, 3D inpainting…等，並曾嘗試運行其中的一些 code。但因為在操作的過程中遇到一些難以克服的問題，如在影片連續深度值推算部分由於模型運行過於耗費時間與機器資源、相機位置推算的模型在環境安裝上有版本無法相容等問題，最終選定使用單張照片做出特效後再生成短影片。後續實作方面我們選擇以 python open-cv 為主要的套件，針對圖片處理分為兩個部分：照片深度值以及圖片空間消失點相關特效。

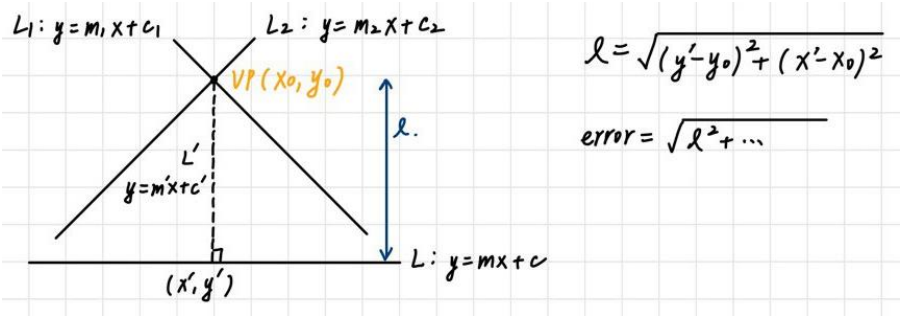
首先是照片深度值推算模型的部分，我們使用的是 Intel-Midas v3 model，對每張照片做深度預測後會得到一張 depth map。附圖為預測結果示例，可以從左邊的原圖與右邊深度預測圖對比中看出預測有一定的準確度。



文字按深度移動特效的部分以賦予文字 PNG 特定深度值以製造移動時被前方物體遮住的效果，而流體特效則是在指定深度值範圍內將符合條件的 pixel 上色。附圖為實作的結果，左圖是我們將 logo(黃色圖案)固定再 depth = 15 並做移動，可以發現他會被實際位置離鏡頭較近的貓咪蓋住；右圖則是將照片中深度值判斷在貓咪後方的 pixel 都疊上半透明的藍色。

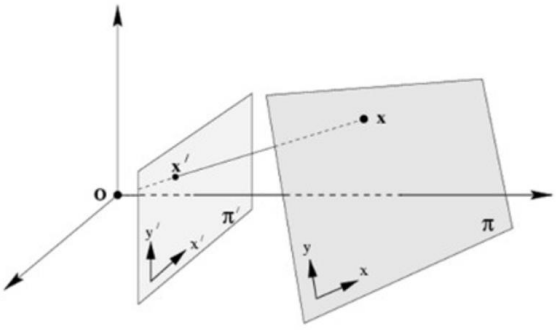


消失點的部分則是先對原圖做灰階模糊處理與邊緣偵測，再使用霍夫直線查找(Hough Lines)描繪出所有理想的線條，最後利用這些線條以類似 RANSAC (RANdom SAmple Consensus)的演算法計算空間中的消失點。首先過濾出角度最合適的所有線段成為集合(數量上限為15)，在此集合中按順序兩兩尋找交點，取這些點中 error 最低的點座標作為最終的預估消失點。各點的 error 數值為此點到各直線距離的平方和之平方根。以下為計算方法圖示。



附圖為執行後的結果。左圖為灰階處理後偵測出的邊緣，中間為霍夫直線查找的結果(以顏色分類，垂直為紅色、水平為藍色、其餘綠色，圖中數字為線段在 Lines[]陣列中的 index)，右圖中的紅點則是計算出的消失點。

至於文字圖片變形的貼牆效果則是運用了圖片的透視變換(Perspective Transformation)，實作方法為將欲變形的圖片的四個頂點重新 fit 到指定的四點座標上，如左下方圖片所示，原圖片中的 pixel 元素會被投影到一個新的視平面(viewing plane)上。右圖為搭配先前計算出的消失線實際應用的結果(圖中紅色線為消失線)。



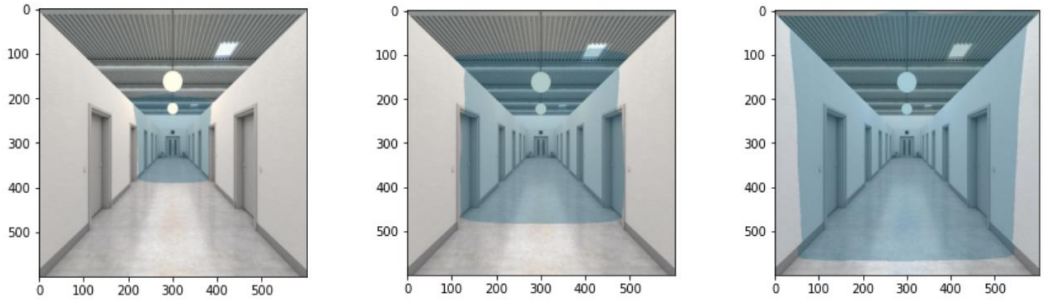
最後，UI 的部分是使用 tkinter 寫出基本框架，上傳照片並點選可以跑上述特效的按鈕，製作出的特效影片即會在 UI 右方的 frame 中播放。

七、 系統實現與實驗

Fluid Effect:

首先利用 Intel-Midas v3 model 對每張照片做深度預測，並得到照片的 Depth Map，接著根據照片特定 pixel 的深度值幫照片加上如液體般的濾鏡，並從 depth 較小的 pixel 跑到較大的

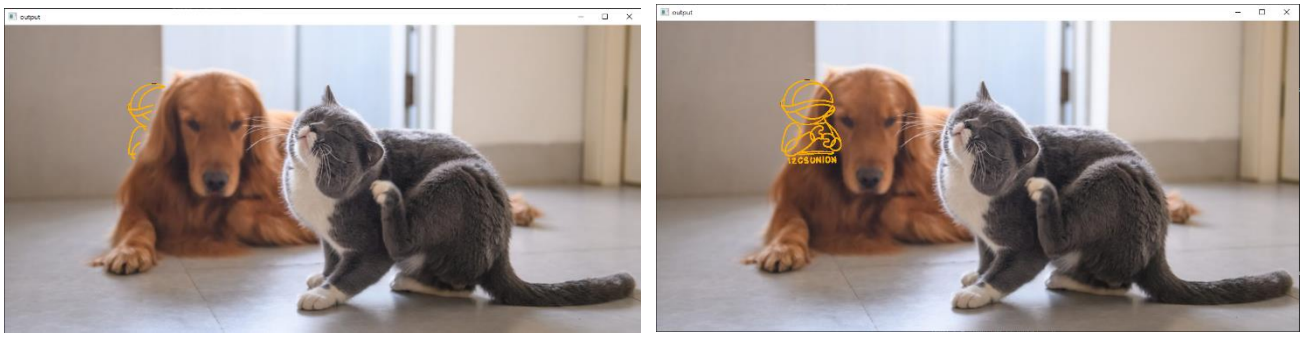
pixel，並將每一張做完特效的 frame 用 open-cv 輸出一部影片，這樣就會得出有如液體從照片深處流出的影片特效。



depth<25　　depth<45　　depth<85

Text / PNG Move on Image:

首先利用 Intel-Midas v3 model 對每張照片做深度預測，並得到照片的 Depth Map，接著將想要輸入的文字或是影片，先轉換成 PNG 檔，並將非文字部分的透明度調成 0，最後指定一個特定的深度值，讓文字的 PNG 檔在這個深度值上移動，就會有文字可能會被比他深度淺的物體蓋住的效果。可以看到設定不同 depth，logo 會有深淺不同的效果。



depth=10　　**depth=30**

Text Moving Along Vanishing Line:

依照 section6 中的步驟實作。在消失點計算的試驗中，雖然已經透過 Hough Lines 直線檢測篩選長度、演算法中包含篩選線條角度，但在一些構圖線條過於複雜的特殊案例中仍會造成無法檢測出消失點的情況，如附圖的棋盤格地板嚴重的干擾了計算。目前可以做到將此類結果標示為無法判讀。另一待研究改善的案例為附圖右，由於圖中仍然有直線線段，但由於此類空間不符合當初查找的演算法設定，因此雖然顯示推算成功(圖中紅點)，但結果是錯誤的。透過以上試驗，我們最後決定事先在圖片選擇上做一些限制，僅對空間感明顯、視覺上可粗略判斷出消失點的圖像做後續的特效處理。

另一個試驗的內容是 text PNG 在經過透視變換後會產生文字本身的邊緣畫質降低的問題。由於我們的特效需要透明背景的文字，因此目前方法是以將圖片從 RGB 轉為 RGBA，偵測圖片

中黑色的部分並透過把 alpha channel 設為 0 使這些黑色 pixel 轉為透明。但因為畫質降低、文字的邊緣變得不清晰，經過測試後最終將判斷為背景色的標準由 RGB(0,0,0)調整為 RGB(160,160,160)，成功的解決了文字邊緣散布黑點的問題。附圖為調整前(左)和調整後(右)的結果。

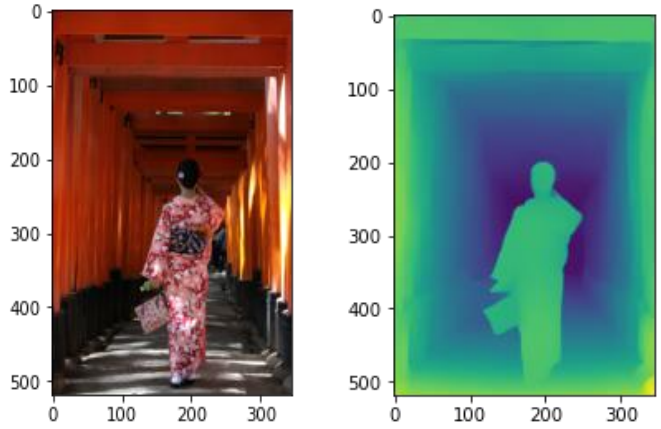
User Interface:

先利用 tkinter 寫出一個基本的 UI 框架，並加入上傳照片的按鈕上傳想要做特效的照片，再利用 tkinter 在 UI 介面加入按鈕，把上述特效的程式內嵌進這個 function，執行按鈕即可得出的影片特效 mp4。最後利用 tkvideo 這個 package 讓做完特效的影片可以在 UI 上播放。

八、 效能評估與成果

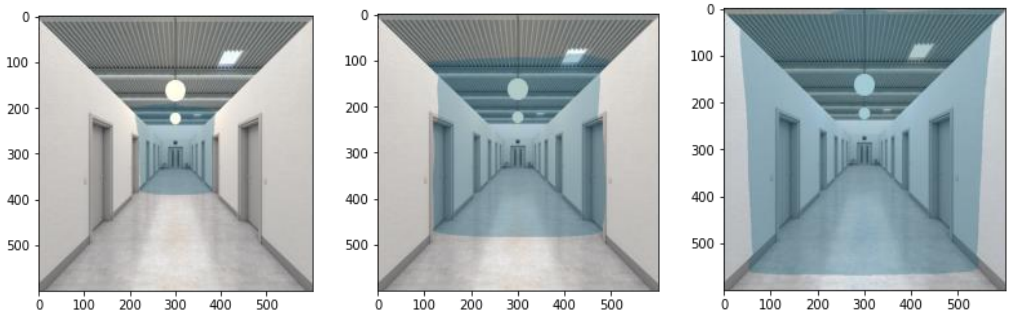
Image Depth Estimation:

左為原圖，透過 Intel-Midas V3 model 得到的深度預測圖如下右圖，可以發現預測效果很好，物體間邊界深度很明顯。而在我們測試下得到這個一張照片的 Depth Map 約 30 秒，相對來說就算使花比較久的時間，但比起使用 video 做深度預測需要花到三四個小時，這樣的效果算非常好。



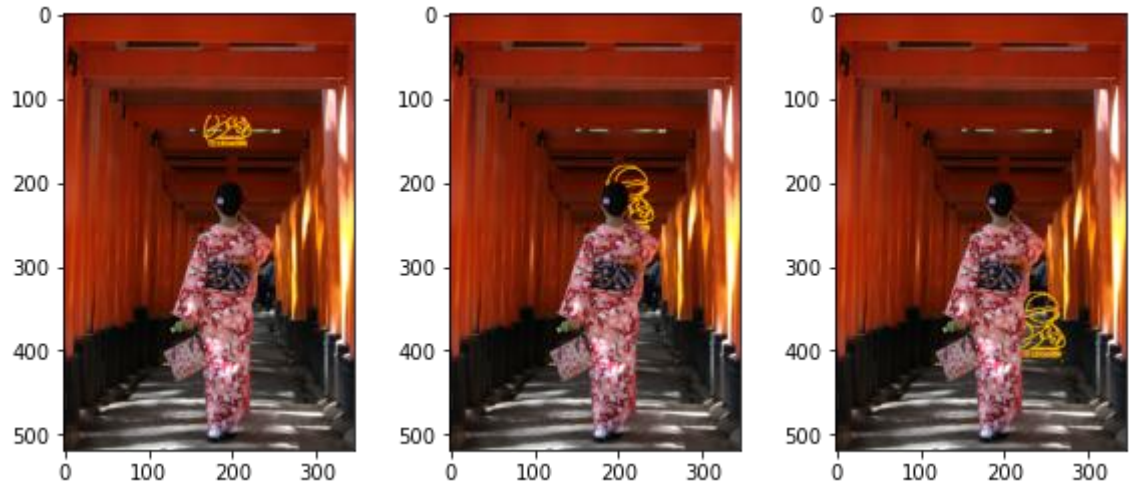
Fluid Effect:

因為加上得 Filter 有動起來的關係，真的很像水在流動。



Text / PNG Moving on Image:

可以發現將(Text/PNG)的深度設得比柱子淺，柱子會確實遮蔽物體。



Text Moving Along Vanishing Line:

以下為文字沿消失線移動的成果圖。

依照目前的方法與設定，生成一部短影片約耗時一分鐘，大多時間是用於生成此 200 張不同角度的透視變換效果圖。現階段還無法全自動化判斷出最適合做為標準線的線段，且如 section7 提到的在圖片上的選擇有一些限制，但大多數空間感明顯的照片用目前的方法都可以實現。

UI:

先點選 Upload Photo，UI 的左邊方框就會顯現出要做特效的原圖，按下右上方任一個 Effect 按鈕後，在右邊播放加上特效後的影片，也會將特效影片存在暫存資料夾，可以直接下載。主要的缺點在於 image 在做 depth estimation 時需要跑非常長一段時間。

九、 結論

針對照片深度特效的部分，我認為我們深度預測對於大部分照片都很精確，但其實還有更多更多的應用，例如可以利用深度加上臉部偵測去實作更多特效內容，而我們沒有做到的則是加強現有的演算法，讓 UI 可以更方便快速的，而針對深度影片則因為運算量過於龐大而沒辦法得出我們想要的即時成果，所以我們選擇棄用。

而關於文字沿消失線移動特效的部分，未來有機會的話可以加入一些機器學習的演算法，讓整個處理過程自動化，如在消失線的選擇上不用額外做人工判斷；自動過濾掉會干擾結果的圖片，如不相關線段過多、空間感不明顯...等。

十、 參考文獻

Wide-Baseline Relative Camera Pose Estimation with Directional Learning
arthurchen0518.github.io
Consistent Depth Estimation
Consistent Video Depth Estimation (roxanneluo.github.io)
Monocular Camera Depth Estimation
Monocular Camera Depth Estimation with Neural Networks in OpenCV C++ - Youtube
Consistent Depth of Moving Objects in Video
https://dynamic-video-depth.github.io/
Vanishing Point with RANSAC algorithm
https://sikasjc.github.io/2018/04/27/vanishing_point/
Open CV documentation: Hough Line Transform
https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html
Open CV documentation: Geometric Image Transformations
https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html
Intel-isl/MiDas Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer
GitHub - isl-org/MiDaS: Code for robust monocular depth estimation described in "Ranftl et. al., Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer, TPAMI 2020"
UI 設計
為應用程式設計圖形化介面，使用 Python Tkinter 模組 (rs-online.com)