

三維電腦視覺與深度學習應用
第十九組書面期末報告

R11943113 葉冠宏, R11922185 杜嘉煒, R10922172 彭曼翊

SuperGlue: Learning Feature Matching with Graph Neural Networks

1.引言

SuperGlue是一種能夠同時進行特徵匹配以及過濾outlier的網路，其中特徵的匹配是通過GNN的架構，以及最佳化求解來達成的。它作為目前特徵匹配網路中的SOTA，不言而喻，其研究價值非常高。本文主要嘗試了針對SuperGlue的預訓練模型的實際運用，主要分爲了三個小實驗。分別為SuperGlue的論文效果驗證，結合Physarum Dynamics以取代SuperGlue內的Sinkhorn Algorithm，以及將SuperGlue和多種不同方法在同一個3D任務上進行效果對比。

2. 相關文獻

在傳統matching的流程裡面，可以分成feature detection和description、feature matching、outlier filtering這些步驟。

2.1 Feature detection 和description

2.1.1 SIFT

在SIFT模型中，我們主要是去透過local gradient的方向，以及形成blob的概念來去用differences-of-Gaussians (DoG)函數和調整sigma的大小參數來去標記出特徵點的位置。其中DoG的函數定義如下：

$$DOG(x, y; s) = L(x, y; s + \Delta s) - L(x, y; s) \approx \frac{\Delta s}{2} \nabla^2 L(x, y; s)$$

透過不斷嘗試不同的sigma值，我們可以標記出不同層度的特徵點。

$$\sigma_{i+1} = k \sigma_i$$

透過這樣的方法，即使影像經過縮小、放大，我們仍然可以去標記出特徵點。

2.1.2 ORB

在ORB的演算法當中，我們會先用FAST的演算法去找出圖像中的關鍵點。方法是對於圖像中的任一點p，我們會先去看以其為圓心的周遭的點，如果中心點p比周圍的點都亮，或是比周圍的點都暗的話，則我們會先選為初步的關鍵點。接下來，我們會針對該關鍵點進行特徵的向量提取，用的方法是BRIEF演算法。BRIEF所用的算法步驟如下：

- (1)首先利用高斯函數對影像進行平滑處理，以防其對於高頻率的噪點過於敏感。
- (2)接著，我們對關鍵點p為中心的高斯分布中取出一個抽樣點。然後我們再以這個抽樣點為中心的高斯分布取出一個二號抽樣點。我們去比較這兩個抽樣點之間的灰階亮度。如果一號抽樣點比二號抽樣點亮的話，我們就記為1，否則就去記為0。
- (3)我們會重複(2)的步驟，直到產生一個約128~512 bit長度的字串。我們用此來當作那個關鍵點的descriptor。

2.1.3 Superpoint

Superpoint相較於上述傳統的方法，可以更精準的抓出特徵點的所在位置，以及其所含的特徵是什麼。他的架構可以分成三個部分，在我們拿到一個影像之後，我們會把他丟到一個encoder中做轉換，接著分別丟到interest point decoder和descriptor decoder的卷積神經網路中做更進一步的運算。

Encoder的作用主要是去減少影像的維度，主要是去參考VGG的架構。其在經過許多convolutional layers之後，接著透過pooling和非線性的激化函數(activation function)進行空間上的降採樣(spatial downsampling)。

而在Interest point decoder的部分，經過encoder的轉換之後，我們經過許多的卷積神經網路轉換，最後透過softmax的轉換函數，以及reshape的動作，最後標記出特徵點的所在位置。在訓練上面，我們是先用一些簡單的幾何圖形讓模型去學會標記一些重點的角點做一個預訓練(pre-train)的動作，形成一個基本的detector。然後我們把影像透過Homographic Adaptation，也就是把影像做各種旋轉、平移、縮放，讓模型可以自我學會標記interest point (是一種pseudo ground truth)。

在descriptor decoder的部分，我們則是把encoder所傳入的資訊，做一個卷積神積網路的轉換，接著我們使用bi-cubic的內插法(interpolation)，以及L2-norm的轉換，來提取到影像特徵點的描述特徵。

最後，我們把一張影像，以及其旋轉平移過後的影像(warped image)，分別送入superpoint的模型之中去標記出各自的interest point，以及description，我們去比對兩張影像的配對點，可以形成一個description loss。而各自的interest point標記，也可以和ground truth之間形成interest point loss。我們藉由這樣的方法來進一步的去訓練superpoint的模型。其中，值得注意的是，我們會有一個dustbin去儲存可能因為warping的原因無法配對上的interest point。藉由這樣的方法，我們可以去找出一張影像的特徵點還有其描述向量。

2.2 Feature matching

2.2.1 Nearest neighbor matching

針對一群的點，我們會先採用中位數，以及垂直、水平等輪流的二元分割方式不斷地針對每個點作出對應的分割區域，最後以二元樹的分割方式去儲存相關資訊。在搜尋一個點的時候，我們會以水平、垂直的分割方式從二元樹所儲存的分割區域中逐漸去收斂縮小所需搜尋的區域，最後尋找到最近的對應搜尋點。

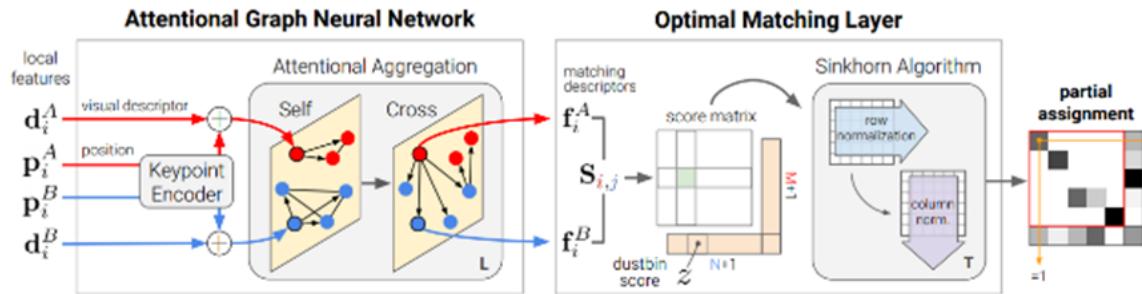
2.3 Outlier filtering

2.3.1 RANSAC

當我們偵測到兩張影像的許多matches之後，當中可能存在著很多的outlier。我們可以隨機取一些matches做出transformation的矩陣。接著我們把其中一個影像用這個transformation矩陣去轉換到第二張影像，我們去看說轉換後有幾個特徵點是在另外一個配對點的誤差範圍內的，就算為是inliers，否則就為outlier。我們希望可以重複這樣的步驟以找到最佳的transformation矩陣，使inlier的數目越多越好。

3. SuperGlue介紹

在3D電腦視覺中，在feature detection之後，最關鍵的技術就是特徵的匹配(feature matching)，這樣才可以為下游的其他研究提供更準確的預測結果。SuperGlue就是我們已經有了許多偵測出來的特徵點(feature detection points)後，一種結合神經網路以及最佳化演算法的方法，使得特徵匹配的準確率可以變得更好。



假設現在我們有兩張影像A和B，其各自有detectors所判定的feature point位置 P_A 及 P_B ，以及其description的特徵向量 d_A 及 d_B 。其中，影像A有M個local feature，影像B有N個local feature。SuperGlue架構由兩個步驟，attentional graph neural network(注意力圖神經網路)以及optimal matching layer(配對最佳化)來使配對率變得更好。前半段的部分，首先，我們會使用encoder(關鍵點編碼器)將關鍵點位置P和視覺描述子d映射到一個高維度的向量之中。接下來使用的GNN架構主要可以分成self和cross兩個部分。這個部分主要是去模仿人類在匹配點的時候會去左右觀察來找尋最佳的匹配。這邊我們會藉由一系列的數學運算以及轉換來求得一個score matrix。最後我們藉由sinkhorn的演算法來找尋點和點之間的最佳匹配機率。

3.1. 簡略實現步驟

3.1.1 KeyPoint Encoder

關鍵點編碼器中左側的X結合了視覺的外觀和位置的信息。其中，位置的信息有經過了MLP的神經網路轉換。數學式子如下：

$$^{(0)} \mathbf{x}_i = \mathbf{d}_i + \text{MLP}_{\text{enc}}(\mathbf{p}_i).$$

這個公式其實是對視覺描述和特徵點的位置做Coupling。這種編碼器使圖網絡能夠在以後的推理中同時考慮外觀和位置，特別是當與attentional aggregation(注意力機制)結合時，是語言處理中流行的"位置編碼器"的一個實例。

3.1.2 Attentional Aggregation

假設有兩張影像A和B, A為query的影像, B為search的影像。首先, 針對query影像的部分, 我們把前面encoder所轉換出的向量 x_Q , 相乘一個權重 w_1 之後, 加上bias b_1 , 可以形成 q_i 的向量。而針對search影像的部分, 我們把encoder所轉換出的向量 x_S , 乘上權重 w_2 和 w_3 之後, 加上bias b_2 和 b_3 , 形成 k_j 和 v_j 的值。我們可以詮釋為一個query 在keypoint的特徵 k_j 上有其對應值 v_j 。其相關的數學式子呈現如下:

$$\begin{aligned} q_i &= W_1^{(\ell)} x_i^Q + b_1 \\ \begin{bmatrix} k_j \\ v_j \end{bmatrix} &= \begin{bmatrix} W_2 \\ W_3 \end{bmatrix} (\ell) x_i^S + \begin{bmatrix} b_2 \\ b_3 \end{bmatrix} \end{aligned}$$

在求出向量 q_i , k_j 之後, 我們對其乘積經過一個softmax的函數, 求得 α 。我們可以視作為 q 向量和 k 向量的相似度。數學式子如下:

$$\alpha_{ij} = \text{Softmax}_j(q_i^\top k_j)$$

因為所要搜索的點有一個對應的值 v_j , 我們針對前面softmax函數所得的 α , 和那個 v_j 去做一個相乘。這邊可以視為如果query和keypoint的向量相近的話, 我們才去取他的value值做相加。相加的值 m 就是aggregation。

其所呈現的數學式子如下:

$$m_{\mathcal{E} \rightarrow i} = \sum_{j:(i,j) \in \mathcal{E}} \alpha_{ij} v_j$$

$$\{j : (i, j) \in \mathcal{E}\} \mathcal{E} \in \{\mathcal{E}_{\text{self}}, \mathcal{E}_{\text{cross}}\}$$

其中這些的計算可以來自自己影像中的向量比較, 形成intra-image edges, 也就是self的attentional aggregation。而來自於不同影像的向量比較則形成inter-image edges, 就是所謂的cross attentional aggregation。透過比較自己影像中的local point, 我們可以更加釐清影像中feature之間的相似度, 以及彼此之間的關係。而比較不同影像間的關係則可以得知轉換過空間過後, feature之間的轉移關係。根據這些關係, 進一步的調整彼此間weight的權重(message passing formulation)。

在計算出aggregation m 之後, 我們會和 $x_A(l)$ 合併矩陣, 經過一個MLP的神經網路。在經過多層的運算之後, 假設我們令影像A最後一層的向量為 $x_A(L)$ 。經過了最後一層的權重 w_L 相乘之後, 在相加到bias b 之後, 最終形成 f_A 的向量。數學式子如下:

$${}^{(\ell+1)}\mathbf{x}_i^A = {}^{(\ell)}\mathbf{x}_i^A + \text{MLP} \left(\left[{}^{(\ell)}\mathbf{x}_i^A \parallel \mathbf{m}_{\mathcal{E} \rightarrow i} \right] \right)$$

$$\mathbf{f}_i^A = \mathbf{W} \cdot {}^{(L)}\mathbf{x}_i^A + \mathbf{b}, \quad \forall i \in \mathcal{A}$$

針對每一對feature之間配對的關係，我們把所求得來自影像A和影像B的fA和fB做內積的相乘，得到Sij，也就是score matrix中每個元素的值，可以視為兩兩相似度的關係。數學式子如下：

$$S_{ij} = \langle \mathbf{f}_i^A, \mathbf{f}_j^B \rangle, \forall (i, j) \in \mathcal{A} \times \mathcal{B}$$

3.1.3 Partial assignment matrix

我們假設點和點之間有對應的機率，如果把這樣的機率儲存在一個矩陣，叫做Partial assignment matrix。我們把前面所求得的score matrix 和partial assignment matrix的值去做相乘，就可以得到總得分數total score。我們希望去最大化所得到的總得分。

$$\sum_{i,j} S_{ij} P_{ij}$$

然而，這個最佳化問題仍受到以下的限制：

一、每一個特徵點最多會有一個最佳的匹配點。此外，每一個點和其他特徵點的配對機率應該相加為1，也就是呈現如下的限制式：

$$\mathbf{P}\mathbf{1}_N \leq \mathbf{1}_M \quad \text{and} \quad \mathbf{P}^\top \mathbf{1}_M \leq \mathbf{1}_N. \quad \mathbf{P} \in [0, 1]^{M \times N}$$

二、部分特徵點因為一些特殊原因（遮擋，噪音等）找不到它的匹配點，我們會把之放在所謂的dustbin中。因此partial assignment matrix加上dustbin之後形成的矩陣如下：

$$\bar{\mathbf{P}}\mathbf{1}_{N+1} = \mathbf{a} \quad \text{and} \quad \bar{\mathbf{P}}^\top \mathbf{1}_{M+1} = \mathbf{b}.$$

$$\mathbf{a} = [\mathbf{1}_M^\top \quad N]^\top \quad \mathbf{b} = [\mathbf{1}_N^\top \quad M]^\top$$

3.1.4 Sinkhorn Algorithm

sinkhorn 演算法，是一個可微分版的Hungarian演算法，傳統上用於bipartite matching。由於我們必須讓每一個列的機率相加為1，每一欄的機率也相加為1。因此我們藉由經過許多輪的權重分配計算，達成這項任務。

3.1.5 Loss計算

我們定義應該要互相配對到的組合屬於M的集合，而因為旋轉轉換造成無法配對上的特徵點也希望其配對到dustbin之中，這些點集合屬於I或是J集合。我們希望這些機率越大越好，使得如下定義的loss函數越少越好。數學定義如下：

$$\begin{aligned} \text{Loss} = & - \sum_{(i,j) \in \mathcal{M}} \log \bar{\mathbf{P}}_{i,j} \\ & - \sum_{i \in \mathcal{I}} \log \bar{\mathbf{P}}_{i,N+1} - \sum_{j \in \mathcal{J}} \log \bar{\mathbf{P}}_{M+1,j} \end{aligned}$$

$$\mathcal{M} = \{(i,j)\} \subset \mathcal{A} \times \mathcal{B}$$

$$\mathcal{I} \subseteq \mathcal{A} \text{ and } \mathcal{J} \subseteq \mathcal{B}$$

4. 實作部分

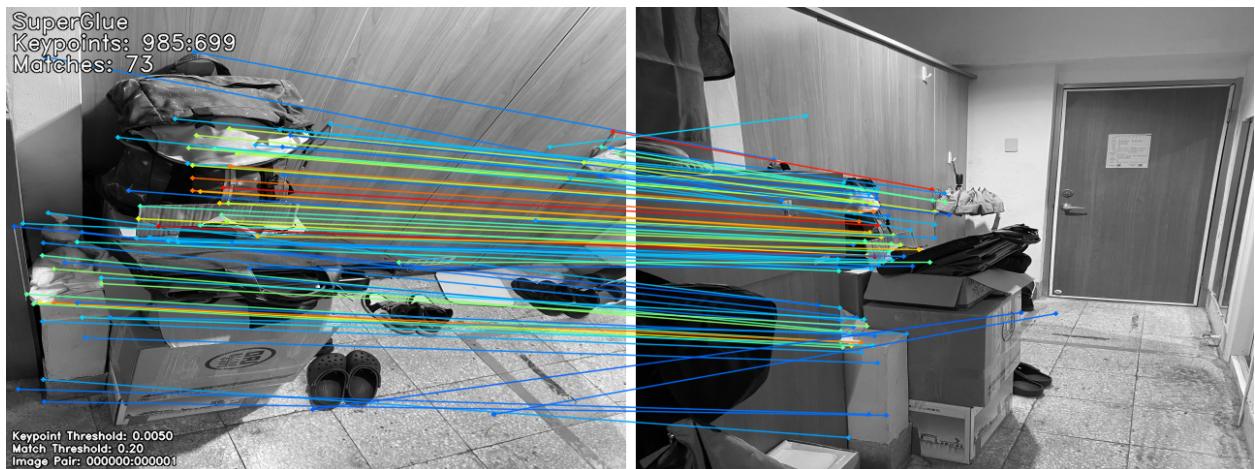
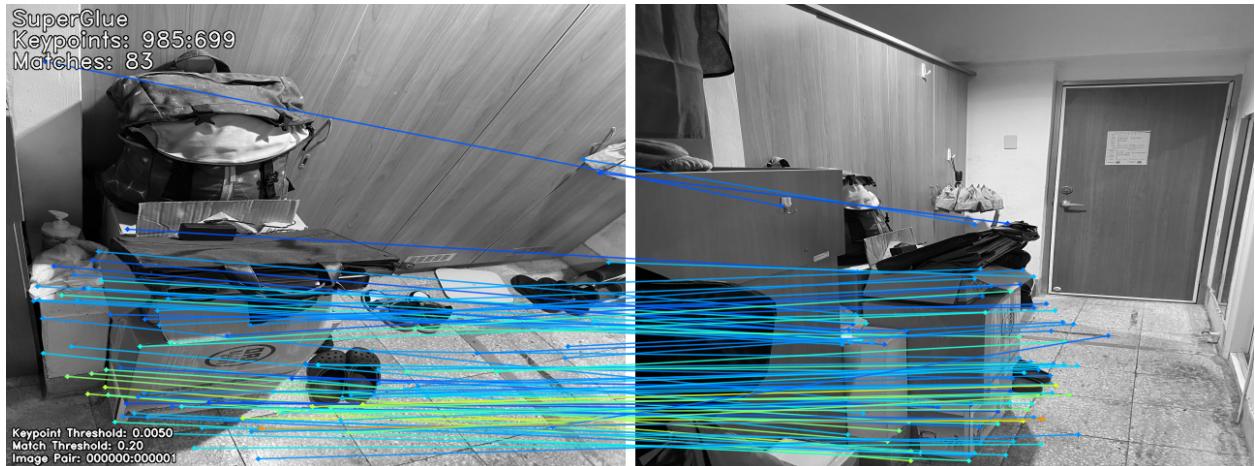
在實作部分，我們各嘗試了三種不同的小實驗，分別為SuperGlue在Visual Odometry和trajectory上的運用和對比、以Physarum Dynamics取代super glue演算法中的Sinkhorn算法，以及SuperGlue在室內與室外效果的對比驗證。

4.1 實驗一：SuperGlue在室內與室外效果的對比驗證

4.1.1 SuperGlue預訓練模型的效果驗證

由於SuperGlue預訓練模型擁有多種不同模式(indoor和outdoor)以及不同的參數效果(resize, max_superpoints等)，因此在本實驗中，本組嘗試了使用自己的數據集，針對不同的室內外圖片參數上的微調進行比對，驗證其效果是否會如文章內宣稱的一樣。

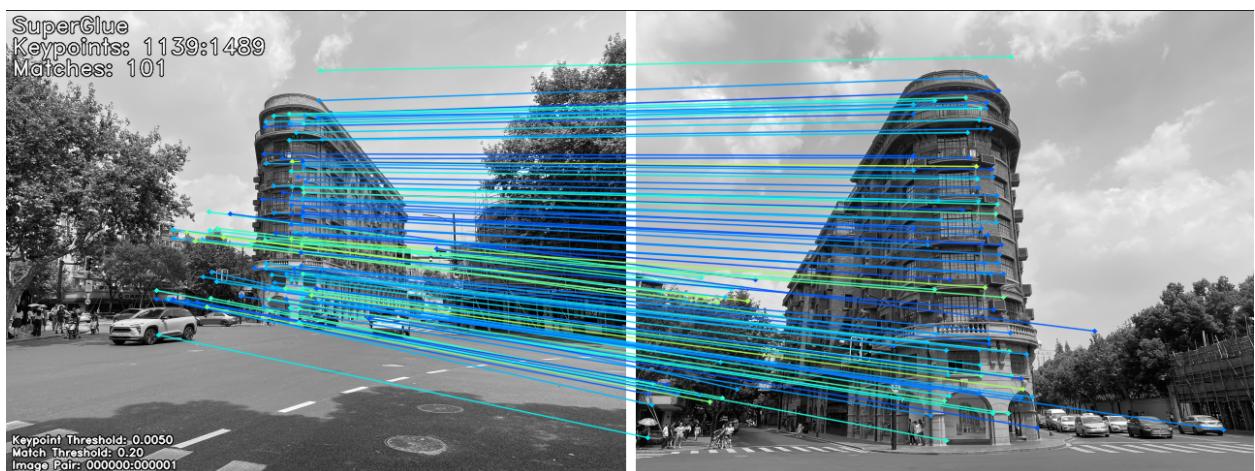
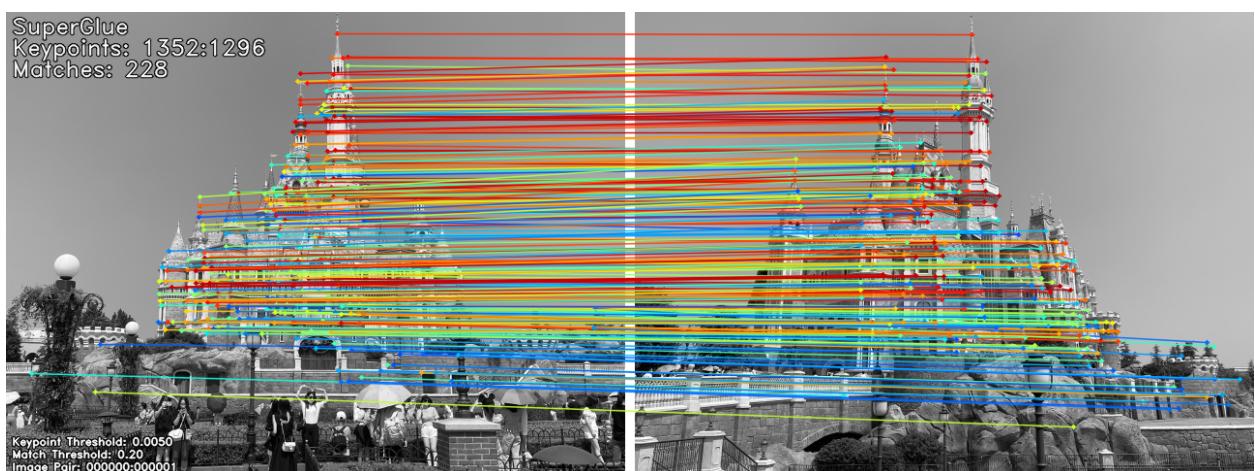
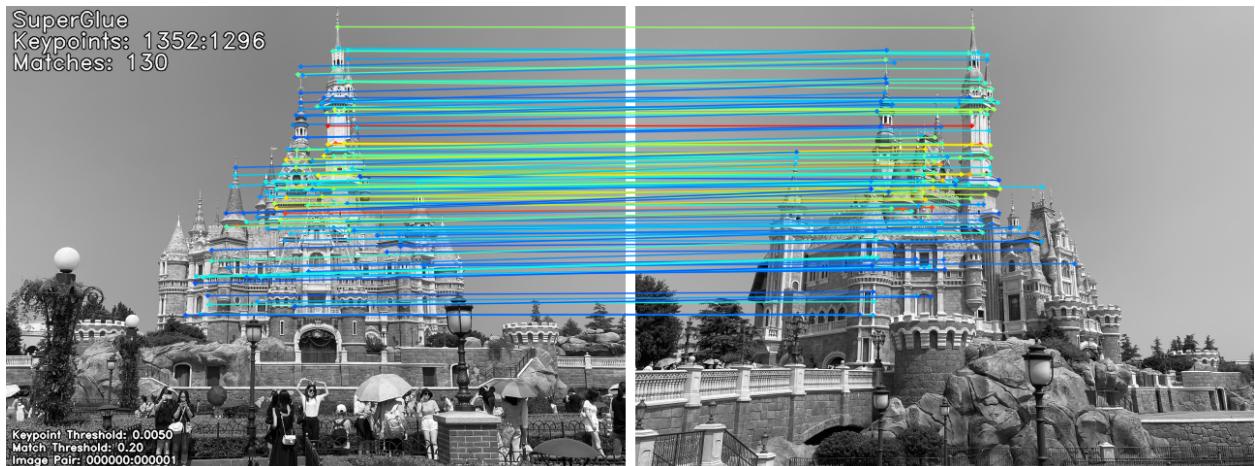
首先我們嘗試在相同的兩張圖片內使用完全相同的參數，只更改其Indoor模式以及Outdoor模式，結果如下：

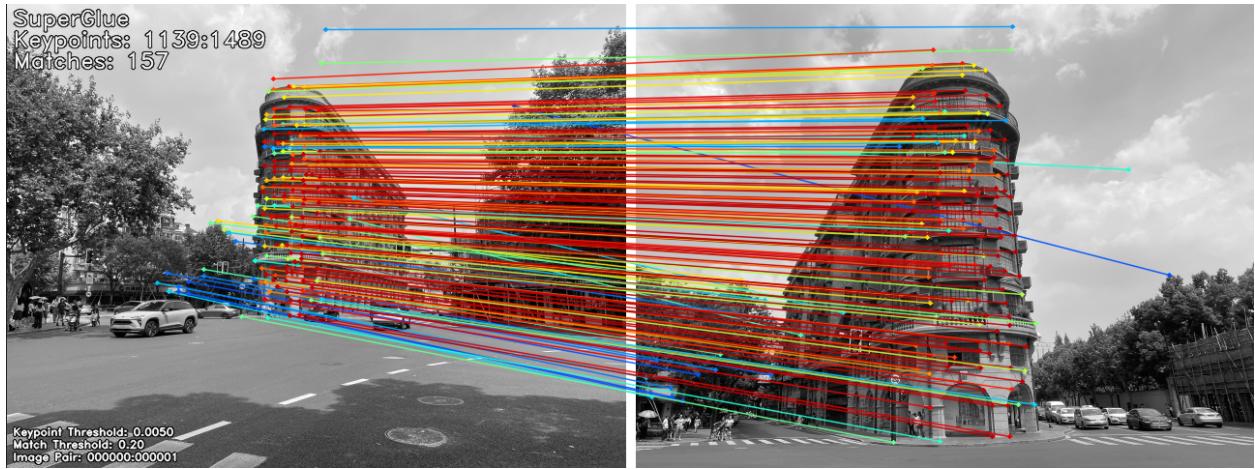


我們可以發現：在完全相同的兩張圖片內使用不同的室內外模式，其效果會有很大的改變。我們明顯發現很多匹配對的置信度在使用不對應場景的模式下相較於對的應場景的模式下變化不明顯（這邊我們設定匹配線越紅則代表信心越高，越綠色則代表信心越低）。

但是有趣的是，我們在室內拍攝的圖片，使用室外的模式，匹配對變低（從八十五變為七十五），準確度和置信度卻提高了（紅色變多），這點可以從書包附近的匹配點看出來。室內模式下書包附近幾乎沒有匹配點，但是在室外模式下書包附近多了接近十條匹配對，並且 SuperGlue認為信心很足。

接下來我們嘗試室外的圖片用於不同的模式下的效果，結果如下圖所示：





同樣的，我們可以發現，在室內情況內的室外圖片比對，儘管整體而言準確率很高，但是 SuperGlue對尋找到的匹配對的信心就遠沒有室外情況的信心大。

接下來我們針對相同的圖片做了不同程度的微調，細節如下：

在限制SuperPoints查找的最大值為100，並且為室外模式的情況下，效果如圖：



限制SuperPoints查找的最大值為200：



此時我們再將最大值設定為500：



由此可見，當我們的最大SuperPoint的數值達到一定程度時，SuperGlue不會將所有的SuperPoint的匹配對全部列出。我們可以從前兩張圖和第三張圖的對比看出，第一、第二張圖片的最左側三個superpoint並沒有顯示在第三張內。

而對於Resize和限制最大匹配量的微調則不太會影響到SuperGlue整體的效果，也不會可視化在輸出圖片中，因此在此不在多做贅述。

4.2. 實驗二:Physarum Dynamics 結合 SuperGlue

4.2.1 Physarum Dynamics

Physarum Dynamics是一種十分高效並且可微調的一種一般線性問題求解器，它可以在Deep Neural Network中以即插即用的方法作為一個層使用。同時它是一種基於數學生物學的方法，經常可以被用於穩健的網絡設計和解決最短路徑等問題。

4.2.2 Physarum Dynamics替代掉Sinkhorn 算法

之所以Physarum Dynamics可以替代Sinkhorn算法的理由在於兩者均為可以解決最優傳輸的問題，它們的目的相同。除此之外，Physarum Dynamics可以無需可行的初始點就可以快速收斂，從理論上來說它的效果會比Sinkhorn的算法的效果更加高效與快速。

Physarum Dynamics的算法如圖所示

Algorithm 1: γ -AuxPD Layer

```
1 Input: LP problem parameters  $A, b, c$ , initial point  
       $x_0$ , Max iteration number  $K$ , step size  $h$ , accuracy  
      level  $\epsilon$ , approximate diameter  $\gamma_P$   
2 Set  $x_s \leftarrow x_0$  if  $x_0$  is provided else  $\text{rand}([n], (0, 1))$   
3 Perturb cost  $c \leftarrow c + \gamma_P \mathbf{1}_0$  where  $\mathbf{1}_0$  is the binary  
      vector with unit entry on the indices  $i$  with  $c_i = 0$   
4 for  $i = 1$  to  $K$  do  
5   Set:  $W \leftarrow \text{diag}(x_s/c)$   
6   Compute:  $L \leftarrow AWA^T$   
7   Compute:  $p \leftarrow L^{-1}b$  using iterative solvers  
8   Set:  $q \leftarrow WA^Tp$   
9   Update:  $x_s \leftarrow (1 - h)x_s + hq$   
10  Project onto  $\mathbb{R}_{\geq \epsilon}$ :  $x_s \leftarrow \max(x_s, \epsilon)$   
11 end  
12 Return:  $x_s$ 
```

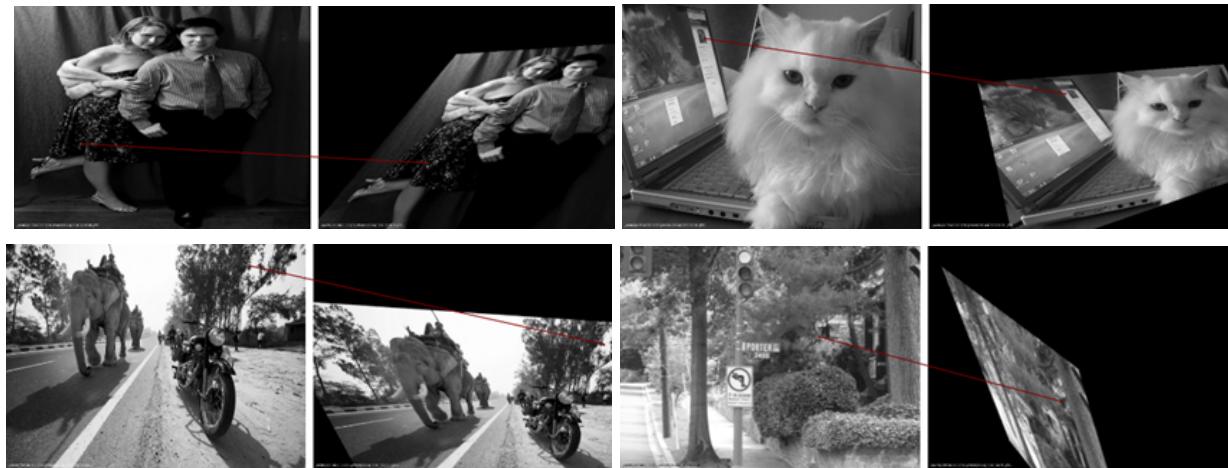
4.2.3 Physarum Dynamics with SuperGlue

由於SuperGlue是一個預訓練的模型並且作者不打算公開其訓練流程，因此，我們嘗試使用第三方開源來還原SuperGlue的訓練過程，並且結合Physarum Dynamics替換掉Sinkhorn 算法。

在本訓練集裡，我們使用了25000張COCO 2014的數據集，訓練epoch為1，GPU為一張V100，手動將最大SuperPoint數目設置為200。

在訓練過程中，我們會隨機對相同圖片的做不同的扭轉、平移、剪裁等，並且每10張圖片就會輸出一張以做可視化。由於運算資源不足並且極其耗時(每10張圖片需要超過十秒)，因此經過一個Epoch後的訓練結果如下圖所示

Good Pairs:



Bad Pairs:



其中，接近九成的圖片為沒有匹配到正確的關鍵點，約一成的的圖片為可以匹配到正確的關鍵點。

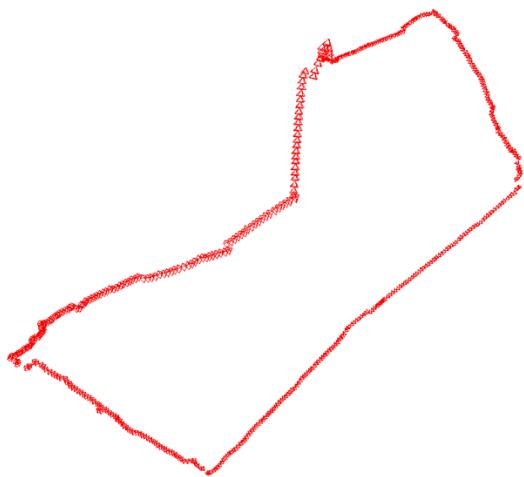
出現以上這種情況的原因，經過討論後我們認為有以下幾點。首先是因為我們人為限制了 SuperPoint的最大查找數目，這就導致了非常多的更加强大的潛在的匹配點不會出現在圖片中，程式也就自然無法對其進行學習，導致錯誤的匹配位置。其次因為本組顯示卡計算資源不足，在實驗過程batchsize只能設為1才能勉強長時間運行，這也就導致均值和方差沒辦法很好的進行計算，損失函數也會很震蕩，從而導致較大範圍內的影響各種參數使得性能下降。

4.3 實驗三：

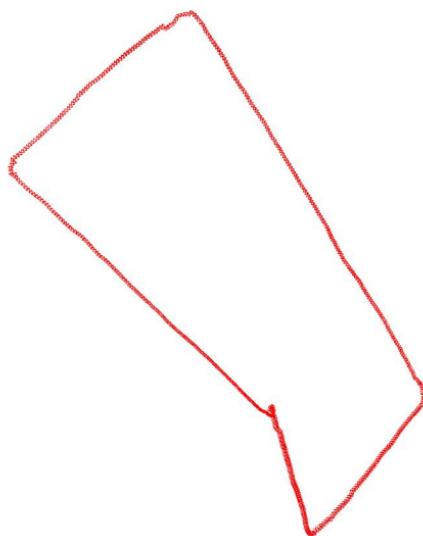
4.3.1 Visual Odometry的比較

由於SuperGlue是一項主要運用在2D空間中高性能快速尋找匹配點的方法，為了能讓SuperGlue方法與3D電腦視覺相結合，本組嘗試使用兩種不同的方法對作業三進行了改寫，使用了台大校園環境的連續拍攝圖片。本次實驗中，對照組為ORB+Brute-Force Matcher的結合，實驗組為SuperPoints + SuperGlue的結合。其中，ORB和Superpoint為特徵抽取的方法，然後brute-force matcher以及superGlue為matching的演算法。以下為實驗的結果：

對照組:ORB+Brute-Force Matcher



實驗組:SuperPoints + SuperGlue



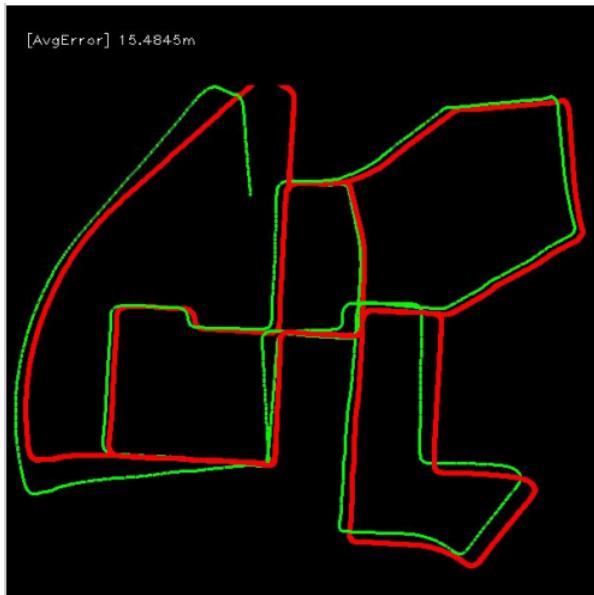
在上述的實驗中，我們是藉由兩張相鄰的影像，在考慮scaling, triangulation等相關因素之後，就可以計算出每張影像將較於第一張影像的translation和rotation的資訊。最後，再利用Open3D把金字塔用pose的資訊轉換作圖，從而找到隨著視角移動的Camera在整個3D空間中的移動軌跡以及拍攝平面。

從上圖的實驗結果可以看出，行人在移動和拍攝時，orb+brute-force matcher無法極其精確且完整的繪畫出行動軌跡，其距離具有極大的偏移，並且在每一個轉角的位置都會出現無法找到Pose或每一個Frame形成的金字塔間的距離過大的問題。而superpoint+superglue的組合，我們則可以看出實驗結果明顯比前者來的平滑許多，而且穩定性也比較高。

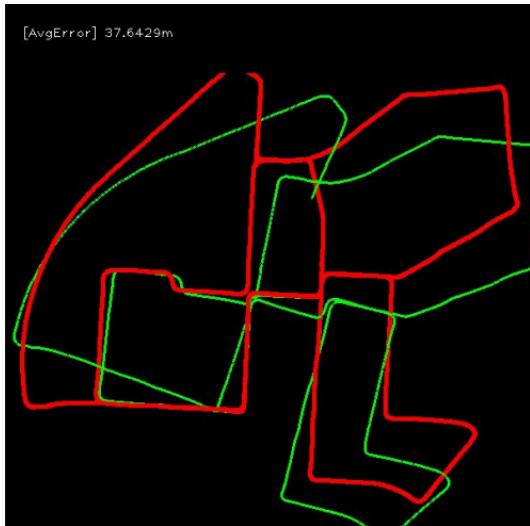
4.3.2 trajectory 的比較

在這個實驗之中，我們使用的是KITTI網站所提供的odometry 資料集，其中含的資料是圖片以及有關圖片pose的真實資訊。我們的實驗組是superpoint+superglue，對照組有兩個，其中一個是superpoint+flann matcher，第二個是orb+brute-force matcher。以下為各組的實驗結果，其中紅色的線表示ground truth，而綠色的線代表所估計的trajectory：

superpoint+superglue:



superpoint+flann matcher:



orb+brute-force matcher:



我們可以觀察到，在orb+brute-force matcher的這個組合中，其和ground truth的平均誤差是約344公尺，而superpoint+flann matcher的這個組合和ground truth的誤差是37.6公尺，superpoint+superglue的誤差則是15.4公尺。

所以，傳統的orb+brute-force matcher模型所造成的誤差是很大的。而在基於一樣的feature identifier, 也就是superpoint, 的情況下, superglue的matching能力明顯是比其他matcher, 例如:flann...等, 還要來的好。

5. 遇到的困難

首先，當我們選定題目後，我們遇到最大的困難在於SuperGlue是一個已經預訓練好的模型，而且作者並不打算公開SuperGlue的訓練過程，這就對我們的實驗拓展造成了很大的阻礙。因為是預訓練好的模型，本組無法從頭開始去對模型改造該網路並查看效果，因此經過討論後，本組才嘗試進行SuperGlue的實際應用上的研究。

其次，SuperGlue是一種主要運用於2D空間內的尋找匹配對的方法，如何讓它和3D視覺相聯繫，經過討論後，本組才嘗試使用SuperGlue替換掉作業三中的ORB，並且在Trajectory上針對不同的方法進行比對。

最後，我們嘗試了通過Physarum Dynamics替換掉Sinkhorn算法，但是在實際過程中，CUDA報錯需要4.1TB的顯示卡內存。由於計算資源完全不夠（一張V100）以及運行時間過長（2500張輸出需要九小時），只能縮減參數量，並且設定最大的SuperPoint檢測數目以加快實驗進度。因此才會實驗失敗，出現百分之九十圖片匹配錯誤。

6. 結論：

總體而言，SuperGlue的表現效果符合預期，它在室內和室外的效果上表現都非常出色，在V100上可以以非常快速的找到圖片中所有的可能的匹配點。不僅如此，通過章節4.1內的實驗，SuperGlue的室外模式的效果遠超室內模式，尤其體現在準確度和置信度上，甚至可以找到室內模式中無法找到的匹配對，但是相對的，其他不太有信心的匹配對的數目則會相對變少。除此之外，SuperGlue的SuperPoints查找的最大值如果被限制，則會影響到整個圖片的匹配對質量，並且如果SuperPoint數值達到一定程度時，SuperGlue不會將所有的SuperPoint的匹配對全部列出。最後，對於SuperGlue的Resize和限制最大匹配量的微調則不太會影響到SuperGlue整體的效果。

對於4.2中的實驗結果僅有百分之十是正確匹配點，原因之一在於計算資源不足，這也就導致了我們只能限制SuperPoint的搜尋數值，而這一點通過4.1的實驗已經證明會極大程度上影響到SuperGlue的效果。除此之外，在勉強運行的情況下，每一張圖片迭代的時間也極其耗時。

而在4.3的實驗當中，我們明顯可以觀察到superglue的匹配能力可以使得在3D視覺的應用研究上有了明顯的效果提升。然而，superglue的運算需要強大的硬體資源，否則運算速度會很慢，導致其無法很好地應用在某些需要即時運算的場景中。因此，對於不同的應用場景

, 有時候準確率和運算的即時性兩者孰輕孰重將交由使用者去評估衡量該使用哪一種方法可以有效地達成自己的研究目標。

7. 分工

1.R11943113 葉冠宏 PPT製作(演算法介紹、實驗三)、期中口頭報告、期末口頭報告、code撰寫(實驗三)、書面報告

2.R11922185 杜嘉煒 PPT製作(實驗二)、期末口頭報告、code撰寫(實驗二)、書面報告

3.R10922172 彭旻翊 PPT製作(實驗一)、期末口頭報告、code撰寫(實驗一)

Github連結:

https://github.com/isjwdt/3DCV_G19_Final_22Fall

8. 參考文獻:

1. Marco C. (2013). ‘Sinkhorn Distances: Lightspeed Computation of Optimal Transport’. NIPS 2013.
2. Paul-Edouard S., Daniel D., Tomasz M., Andrew R. (2020). ‘SuperGlue: Learning Feature Matching with Graph Neural Networks’. CVPR 2020.
3. Daniel D, Tomasz M, Andrew R. (2018). ‘SuperPoint: Self-Supervised Interest Point Detection and Description’. CVPR 2018.
4. Realcat. (2022). ‘SuperGlue一种基于图卷积神经网络的特征匹配算法’. Available at: <https://vincentqin.tech/posts/superglue/> (Accessed: 22 December 2022).
5. Magic Leap. (2020). ‘Learning Feature Matching with Graph Neural Networks’. Available at: <https://psarlin.com/superglue/> (Accessed: 23 December 2022).
6. Hung-yi L. (2020) ‘Graph Neural Network’. Available at: <https://www.youtube.com/watch?v=eybCCtNKwzA> (Accessed: 25 December 2022).
7. Tomasz M. (2020). ‘Deep Visual SLAM Frontends: SuperPoint, SuperGlue, and SuperMaps’. Available at: <https://www.youtube.com/watch?v=u7Yo5EtOATQ> (Accessed: 22 December 2022).

8. HeatherJiaZG. (2020). <https://github.com/HeatherJiaZG/SuperGlue-pytorch>
9. Yuan G., Hamidreza K., Andreas K., Kurt M., Mohammadamin S. (2022). ‘Physarum Inspired Dynamics to Solve Semi-Definite Programs’. Retrieved from <https://arxiv.org/abs/2111.02291>.
10. Junqi. (2020). Available at: <https://zhuanlan.zhihu.com/p/261966288> (Accessed: 22 December 2022).
11. ‘Features from accelerated segment test’ (2022). Wikipedia. Available at [https://en.wikipedia.org/wiki/Features_from_accelerated_segment_test#:~:text=Feature~s%20from%20accelerated%20segment%20test%20\(FAST\)%20is%20a%20corner%20detection,~and%20was%20published%20in%202006](https://en.wikipedia.org/wiki/Features_from_accelerated_segment_test#:~:text=Feature~s%20from%20accelerated%20segment%20test%20(FAST)%20is%20a%20corner%20detection,~and%20was%20published%20in%202006) (Accessed: 23 December 2022).
12. ‘Comparison of Nearest Neighbor Search Algorithms’. <https://rahvee.gitlab.io/comparison-nearest-neighbor-search/> (Accessed: 23 December 2022).