SuperGlue: Learning Feature Matching with Graph Neural Networks

Group 19

R11943113 葉冠宏
R11922185 杜嘉煒
R10922172 彭旻翊

Motivation

An effective model for feature matching should aim at finding all correspondences between reprojections of the same 3D points and identifying keypoints that have no matches.

We formulate SuperGlue as solving an optimization problem, whose cost is predicted by a deep neural network. This alleviates the need for domain expertise and heuristics – we learn relevant priors directly from the data.

Problem definition

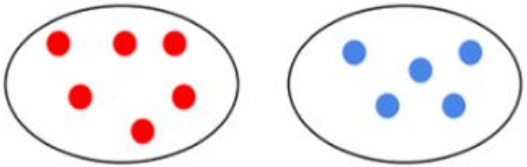
- In vision based localization, one of the most critical techniques to accurately match the location of the query image taken and restore the camera pose is local feature matching.
-
- The matching of graphs after modeling is generally an NP-hard quadratic assignment problem.
-
- SuperGlue learns a flexible cost using a deep neural network that transforms the problem into the equivalent of the optimal transportation problem in graph theory, whereupon it can be conveniently solved using some off-the-shelf algorithms.

Problem definition

Inputs → 

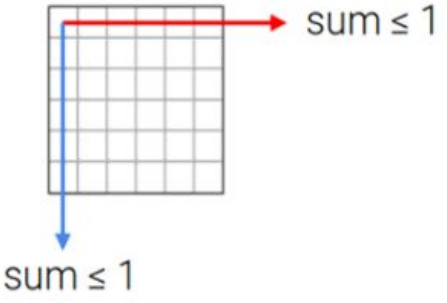
→ **Outputs**

- Images **A** and **B**
- **2 sets** of **M**, **N** **local features**
  - Keypoints: $\mathbf{p}_i := (x, y, c)_i$
    - Coordinates (x, y)
    - Confidence c
  - Visual descriptors: $\mathbf{d}_i$



Single a match per keypoint
+ occlusion and noise
→ a **soft partial assignment**:

$$\mathbf{P} \in [0, 1]^{M \times N}$$



Problem definition

A minimal matching pipeline

SuperGlue: context aggregation + matching + filtering

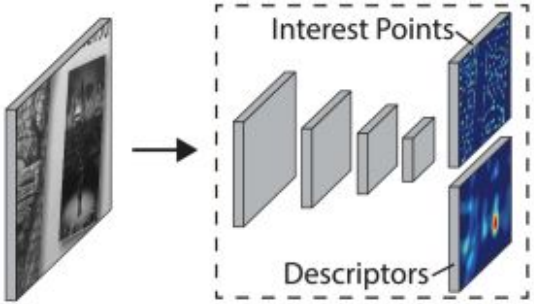
image pair

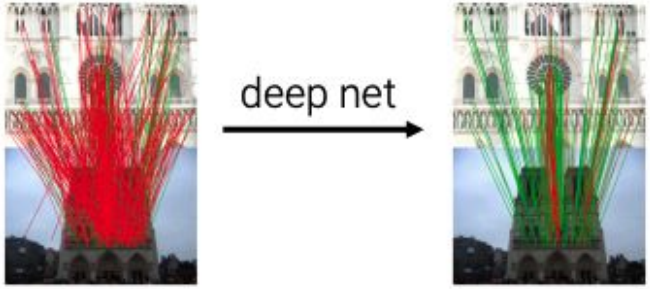
detection → description → feature matching → outlier filtering → pose estimation

> Classical: SIFT, ORB
> Learned: SuperPoint, D2-Net

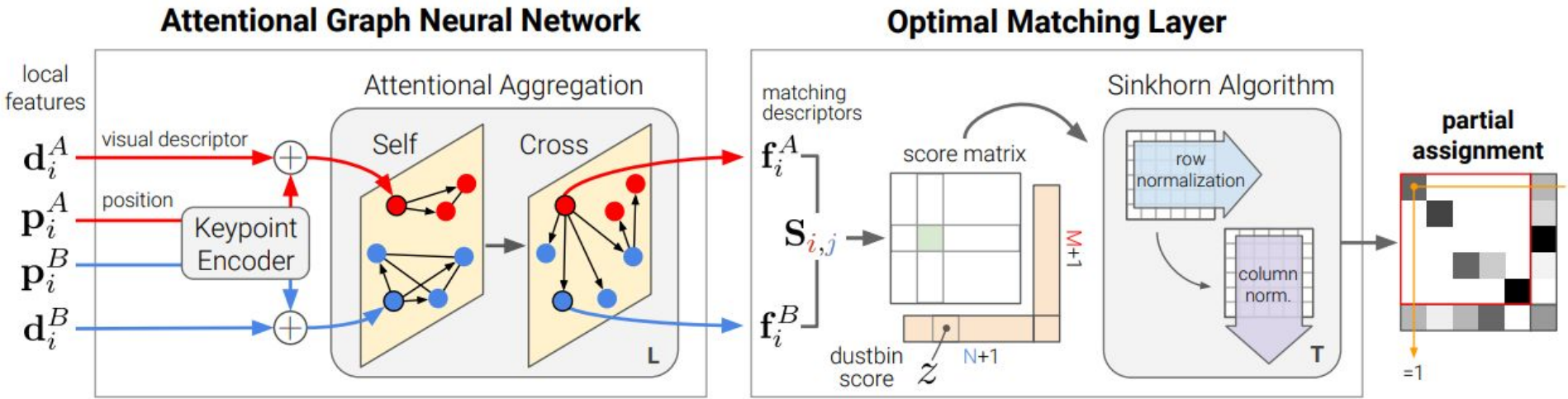
Nearest Neighbor Matching

> Heuristics: ratio test, mutual check
> Learned: classifier on set

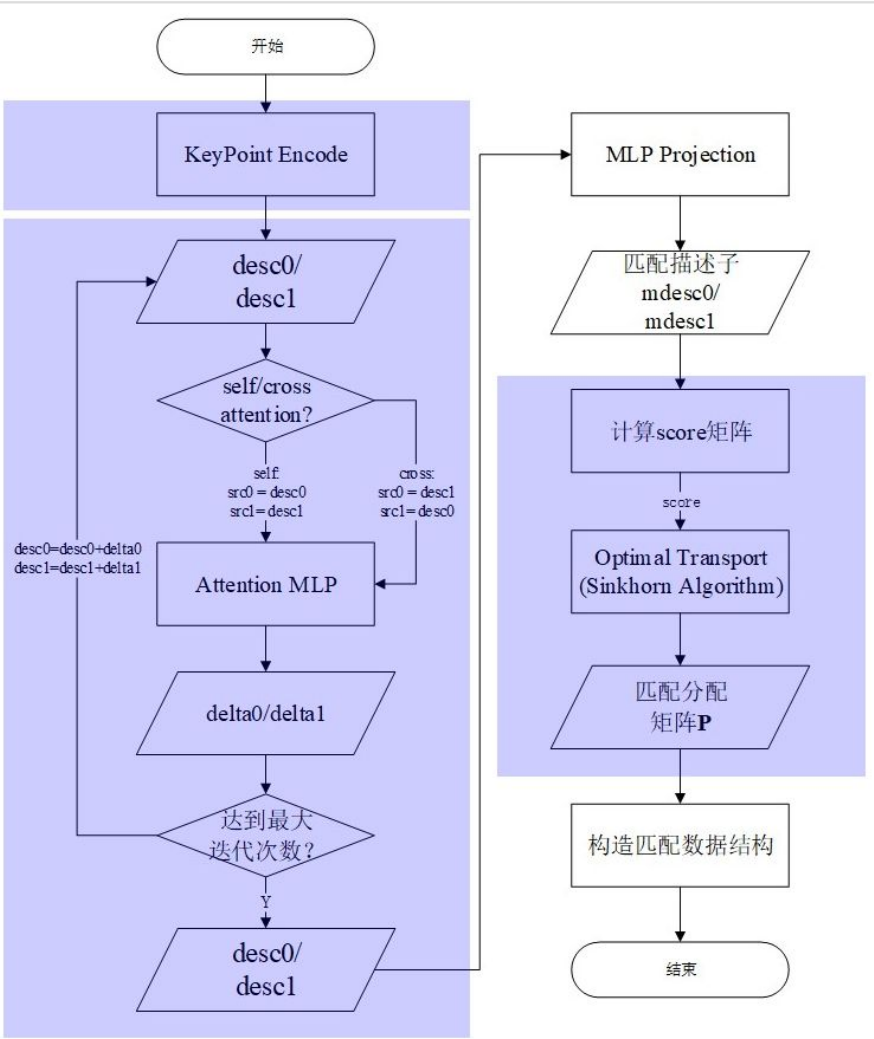




Methodology

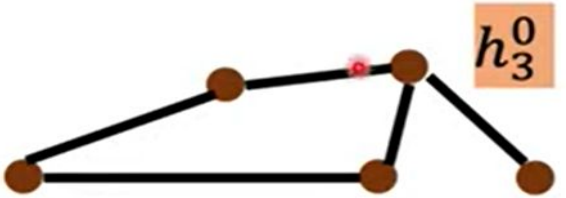


Methodology



Attentional graph neural network

GAT (Graph Attention Networks)

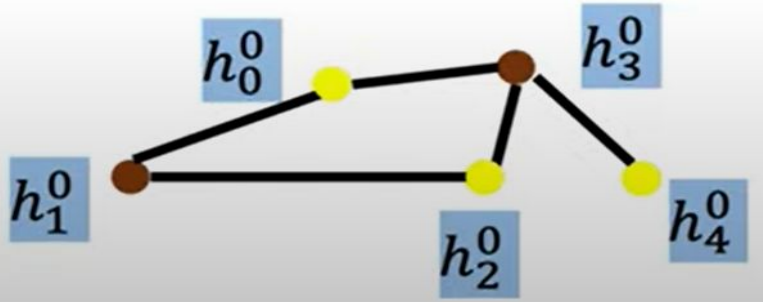


$$h_3^0 = e_{3,0} \cdot h_0^0 + e_{3,2} \cdot h_2^0 + e_{3,4} \cdot h_4^0$$

$$f(h_3^0, h_0^0) = e_{3,0}$$

$$f(h_3^0, h_2^0) = e_{3,2}$$

$$f(h_3^0, h_4^0) = e_{3,4}$$

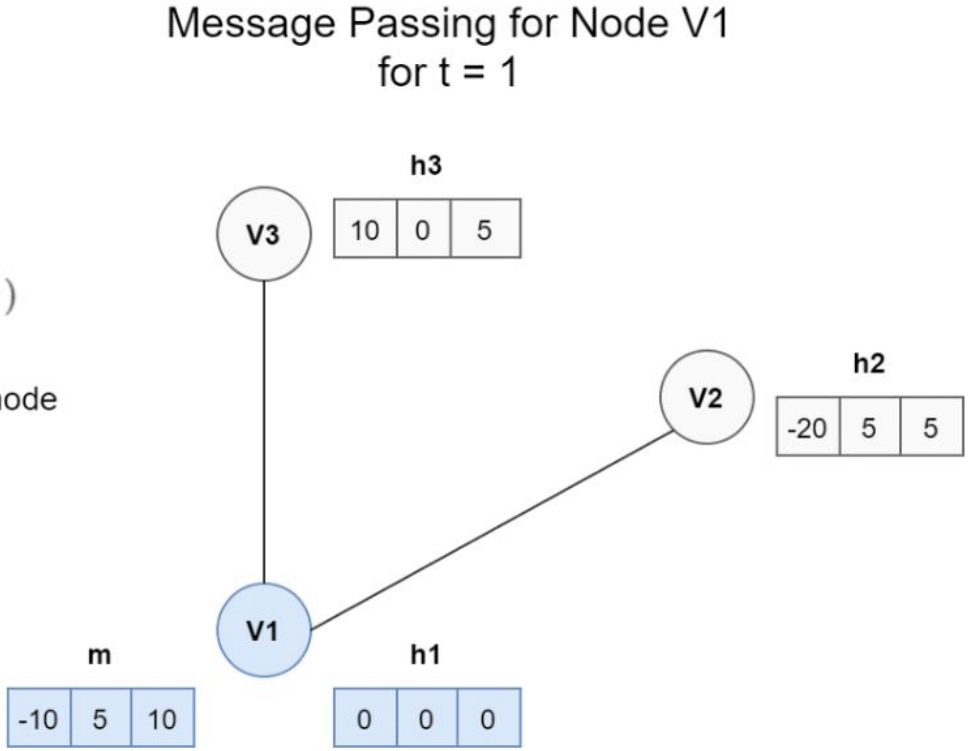


Message Passing Neural Networks

$$m_v^{t+1} = \sum_{w \in N(v)} h_w^t$$

$$h_v^{t+1} = average(h_v, m_v^{t+1})$$

ht - hidden state for each node



A very simple example of message passing architecture for node **V1**. In this case a message is a sum of neighbour's hidden states. The update function is an average between a message **m** and **h1**. Gif created by author

Partial soft assignment matrix

Constraints:

i) A keypoint can have at most a single correspondence in the other image.

ii) Some keypoints will be unmatched due to occlusion and failure of the detector.

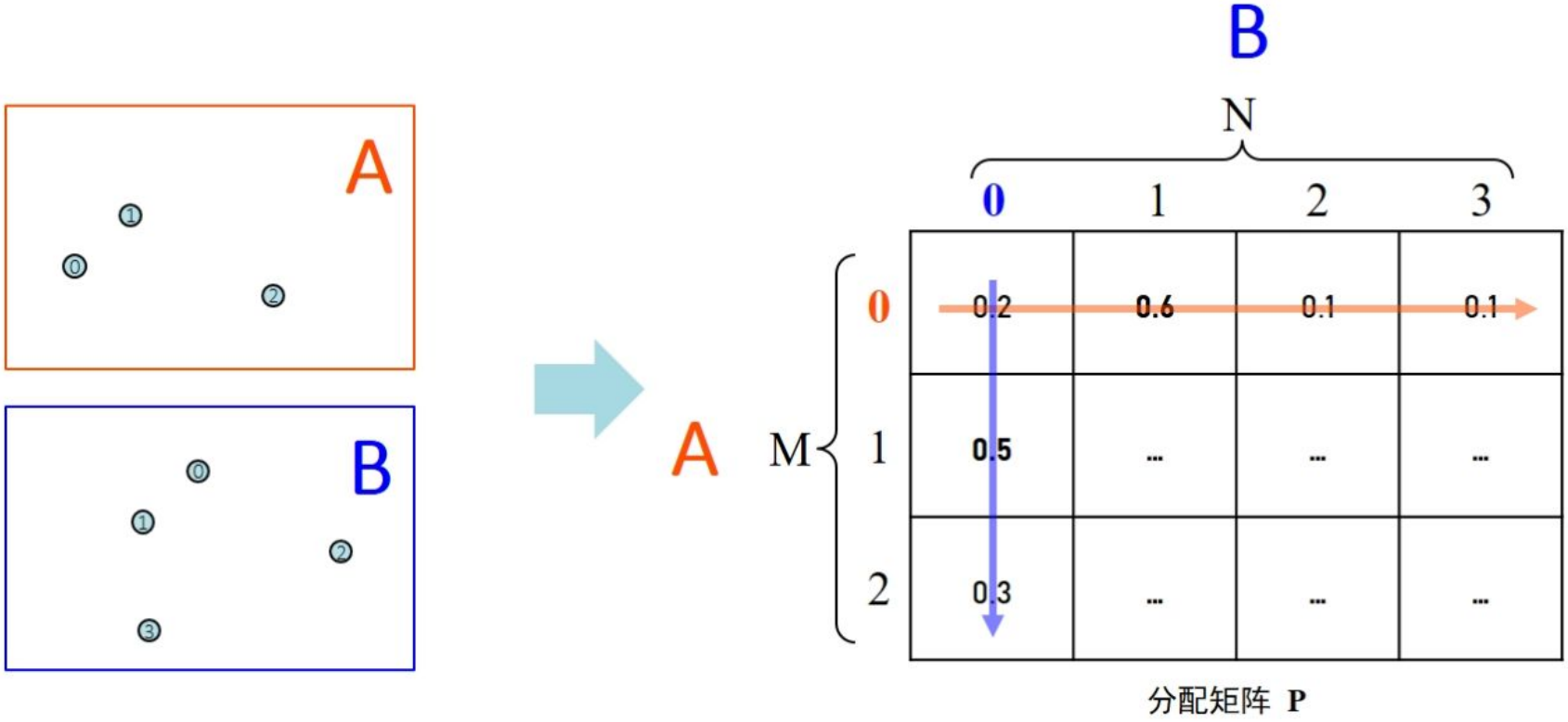
=>each possible correspondence should have a confidence value.

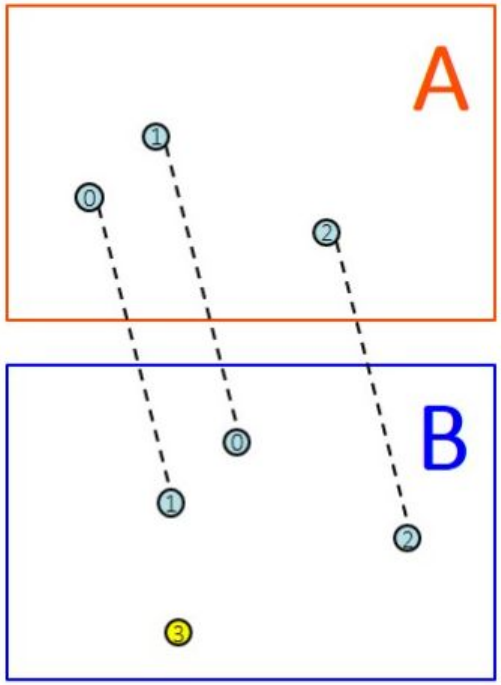
$$\mathbf{P}\mathbf{1}_N \leq \mathbf{1}_M \quad \text{and} \quad \mathbf{P}^\top \mathbf{1}_M \leq \mathbf{1}_N. \quad \mathbf{P} \in [0,1]^{M \times N}$$

=>Goal:design a neural network that predicts the assignment P from 2 sets of local features.

Partial assignment

$$\mathbf{P}\mathbf{1}_N \leq \mathbf{1}_M \quad \text{and} \quad \mathbf{P}^\top \mathbf{1}_M \leq \mathbf{1}_N \tag{1}$$





B

N

A \ M	0	1	2	3
0	0.2	**0.6**	0.1	0.1
1	**0.5**	0.2	0.1	0.0
2	0.3	0.2	0.4	0.0

分配矩阵 **P**

Step1:

Keypoint encoder: $${}^{(0)}\mathbf{x}_i = \mathbf{d}_i + \mathrm{MLP}_{\mathrm{enc}}\left(\mathbf{p}_i\right). \qquad (2)$$

Step2:

$$\mathbf{q}_i = \mathbf{W}_1\, {}^{(\ell)}\mathbf{x}_i^{Q} + \mathbf{b}_1 \qquad S, (Q, S) \in \{A, B\}^2$$

Layer l,Image A, Image B

$$\begin{bmatrix}\mathbf{k}_j \\ \mathbf{v}_j\end{bmatrix} = \begin{bmatrix}\mathbf{W}_2 \\ \mathbf{W}_3\end{bmatrix} {}^{(\ell)}\mathbf{x}_j^{S} + \begin{bmatrix}\mathbf{b}_2 \\ \mathbf{b}_3\end{bmatrix}$$

Q has query keypoints, S has source keypoints

Query:qi retrieves the Values:vj based on their attributes the Keys:kj

=> $$\alpha_{ij} = \mathrm{Softmax}_j\left(\mathbf{q}_i^{\top}\mathbf{k}_j\right)$$

=> $$\mathbf{m}_{\mathcal{E}\to i} = \sum_{j:(i,j)\in\mathcal{E}} \alpha_{ij}\mathbf{v}_j,$$

$\mathbf{m}_{\mathcal{E}\to i}$:the result of aggregation

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

Self:within the same image, Cross:between images

=> $${}^{(\ell+1)}\mathbf{x}_i^{A} = {}^{(\ell)}\mathbf{x}_i^{A} + \mathrm{MLP}\left(\left[{}^{(\ell)}\mathbf{x}_i^{A} \,||\, \mathbf{m}_{\mathcal{E}\to i}\right]\right),$$

Step3:

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

Note:the matching descriptors are not normalized, and their magnitude can change per feature and during training to reflect the prediction confidence.

$\mathbf{S}_{i,j} = < \mathbf{f}_i^A, \mathbf{f}_j^B >, \forall (i,j) \in \mathcal{A} \times \mathcal{B}, \quad \mathbf{S} \in \mathbb{R}^{M \times N}$:Score prediction

Total score: $\sum_{i,j} \mathbf{S}_{i,j} \mathbf{P}_{i,j}$ =>Goal: maximize total score

More constraints:

To let the network suppress some keypoints, we augment each set with a dustbin so that unmatched keypoints are explicitly assigned to it.

$$\bar{\mathbf{S}}_{i,N+1} = \bar{\mathbf{S}}_{M+1,j} = \bar{\mathbf{S}}_{M+1,N+1} = z \in \mathbb{R}.$$

$\mathbf{a} = \begin{bmatrix} \mathbf{1}_M^\top & N \end{bmatrix}^\top \mathbf{b} = \begin{bmatrix} \mathbf{1}_N^\top & M \end{bmatrix}^\top$:the number of expected matches for each keypoint and dustbin in A and B

While keypoints in A will be assigned to a single keypoint in B or the dustbin

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

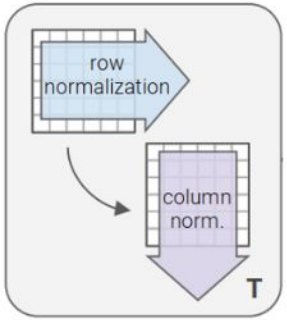
Sinkhorn Algorithm(solving differentiable optimal transport problem)

$$\max \sum_{i,j} \bar{\mathbf{S}}_{i,j} \bar{\mathbf{P}}_{i,j}$$

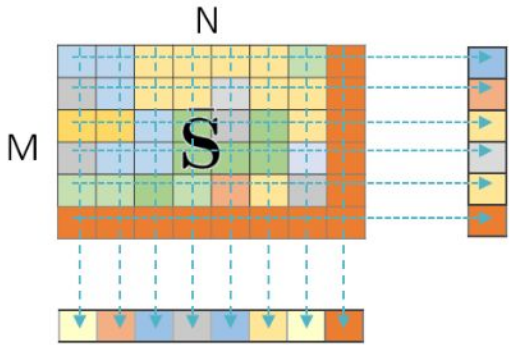
Sinkhorn Algorithm 进行迭代求解

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

$$\mathbf{a} = \begin{bmatrix} \mathbf{1}_M^{\top} & N \end{bmatrix}^{\top} \qquad \mathbf{b} = \begin{bmatrix} \mathbf{1}_N^{\top} & M \end{bmatrix}^{\top}$$



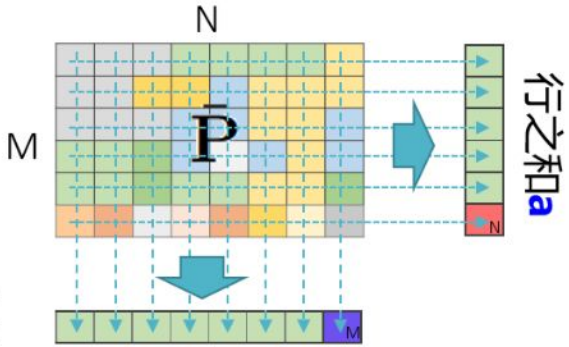
通过对S矩阵**不停的缩放更新**使得分配矩阵的**行/列之和为1**



代价矩阵

Sinkhorn Algorithm

通过对S矩阵不停的缩放更新，每一行求和，然后该行除以该数字，这样一来可以使得每一行的和为1，之后×a，这样可以使得每一行的和为a对应的元素，对每一行执行相同的操作。同样的对列执行相同的操作。



列之和b

分配矩阵

Sinkhorn Algorithm

1次迭代

步骤1：计算目前的行之和=
1+2+3+4+10 = 20
5+6+7+8+10 = 36
2+6+1+3+10 = 22
10+10+10+10+10=50

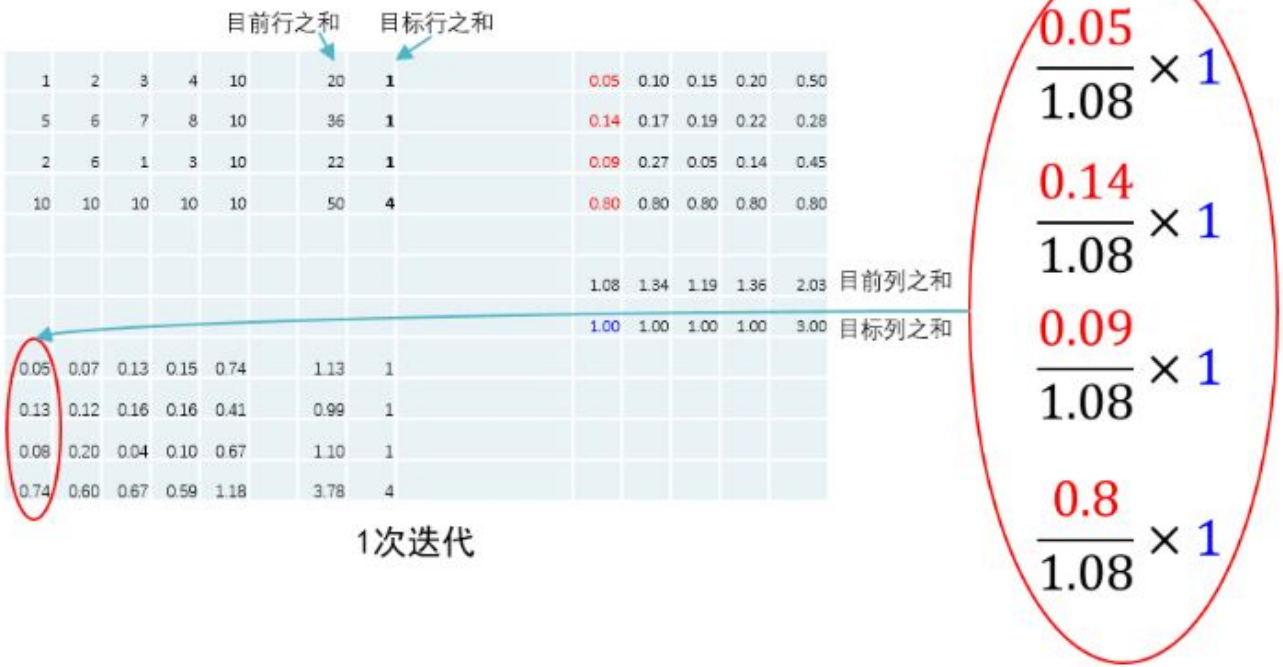
步骤2：对于P矩阵的每一行，分别除以上述行之和并乘目标行之和a

$$\frac{1}{20}\times 1 \quad \frac{2}{20}\times 1 \quad \frac{3}{20}\times 1 \quad \frac{4}{20}\times 1 \quad \frac{10}{20}\times 1$$

$$\frac{5}{36}\times 1 \quad \frac{6}{36}\times 1 \quad \frac{7}{36}\times 1 \quad \frac{8}{36}\times 1 \quad \frac{10}{36}\times 1$$

$$\frac{2}{22}\times 1 \quad \frac{6}{22}\times 1 \quad \frac{1}{22}\times 1 \quad \frac{3}{22}\times 1 \quad \frac{10}{22}\times 1$$

$$\frac{10}{50}\times 4 \quad \frac{10}{50}\times 4 \quad \frac{10}{50}\times 4 \quad \frac{10}{50}\times 4 \quad \frac{10}{50}\times 4$$



1次迭代

$$\frac{0.05}{1.08}\times 1$$

$$\frac{0.14}{1.08}\times 1$$

$$\frac{0.09}{1.08}\times 1$$

$$\frac{0.8}{1.08}\times 1$$

步骤3：计算目前的列之和=
0. 05+0. 14+0. 09+0. 80=1. 08
0. 10+0. 17+0. 27+0. 80=1. 34
0. 15+0. 19+0. 05+0. 80=1. 19
0. 20+0. 22+0. 14+0. 80=1. 36
0. 50+0. 28+0. 45+0. 80=2. 03

步骤4：对于P矩阵的每一列，分别除以上述列之和并乘目标列之和b

1	2	3	4	10	20	1		0.05	0.10	0.15	0.20	0.50
5	6	7	8	10	36	1		0.14	0.17	0.19	0.22	0.28
2	6	1	3	10	22	1		0.09	0.27	0.05	0.14	0.45
10	10	10	10	10	50	4		0.80	0.80	0.80	0.80	0.80
								1.08	1.34	1.19	1.36	2.03 目前列之和
								1.00	1.00	1.00	1.00	3.00 目标列之和
0.05	0.07	0.13	0.15	0.74	1.13	1						
0.13	0.12	0.16	0.16	0.41	0.99	1						
0.08	0.20	0.04	0.10	0.67	1.10	1						
0.74	0.60	0.67	0.59	1.18	3.78	4						

1次迭代

0.05	0.07	0.13	0.15	0.74	1.13	1		0.04	0.07	0.11	0.13	0.65
0.13	0.12	0.16	0.16	0.41	0.99	1		0.13	0.13	0.17	0.17	0.41
0.08	0.20	0.04	0.10	0.67	1.10	1		0.08	0.19	0.03	0.09	0.61
0.74	0.60	0.67	0.59	1.18	3.78	4		0.78	0.63	0.71	0.62	1.25
								1.03	1.01	1.02	1.01	2.93 目前列之和
								1.00	1.00	1.00	1.00	3.00 目标列之和
0.04	0.07	0.11	0.13	0.67	1.01	1						
0.13	0.12	0.16	0.16	0.42	1.00	1						
0.07	0.18	0.03	0.09	0.63	1.01	1						
0.76	0.63	0.70	0.62	1.28	3.98	4						

2次迭代

0.04	0.07	0.11	0.13	0.67	1.01	1		0.04	0.06	0.11	0.13	0.66
0.13	0.12	0.16	0.16	0.42	1.00	1		0.13	0.12	0.16	0.16	0.42
0.07	0.18	0.03	0.09	0.63	1.01	1		0.07	0.18	0.03	0.09	0.62
0.76	0.63	0.70	0.62	1.28	3.98	4		0.76	0.63	0.70	0.62	1.29
								1.00	1.00	1.00	1.00	2.99 目前列之和
								1.00	1.00	1.00	1.00	3.00 目标列之和
0.04	0.06	0.11	0.13	0.66	1.00	1						
0.13	0.12	0.16	0.16	0.43	1.00	1						
0.07	0.18	0.03	0.09	0.62	1.00	1						
0.76	0.63	0.70	0.62	1.29	4.00	4						

3次迭代

0.04	0.06	0.11	0.13	0.66	1.00	1		0.04	0.06	0.11	0.13	0.66
0.13	0.12	0.16	0.16	0.43	1.00	1		0.13	0.12	0.16	0.16	0.43
0.07	0.18	0.03	0.09	0.62	1.00	1		0.07	0.18	0.03	0.09	0.62
0.76	0.63	0.70	0.62	1.29	4.00	4		0.76	0.63	0.70	0.62	1.29
								1.00	1.00	1.00	1.00	3.00 目前列之和
								1.00	1.00	1.00	1.00	3.00 目标列之和
0.04	0.06	0.11	0.13	0.66	1.00	1						
0.13	0.12	0.16	0.16	0.43	1.00	1						
0.07	0.18	0.03	0.09	0.62	1.00	1						
0.76	0.63	0.70	0.62	1.29	4.00	4						

4次迭代

Loss

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

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

$\mathcal{I} \subseteq \mathcal{A}$ and $\mathcal{J} \subseteq \mathcal{B}$ unmatched if they do not have any reprojection in their vicinity

Experiment 1-indoor, outdoor matching comparison

1. Indoor matching is challenging due to the lack of texture, the abundance of self-similarities, the complex 3D geometry of scenes, and large viewpoint changes.

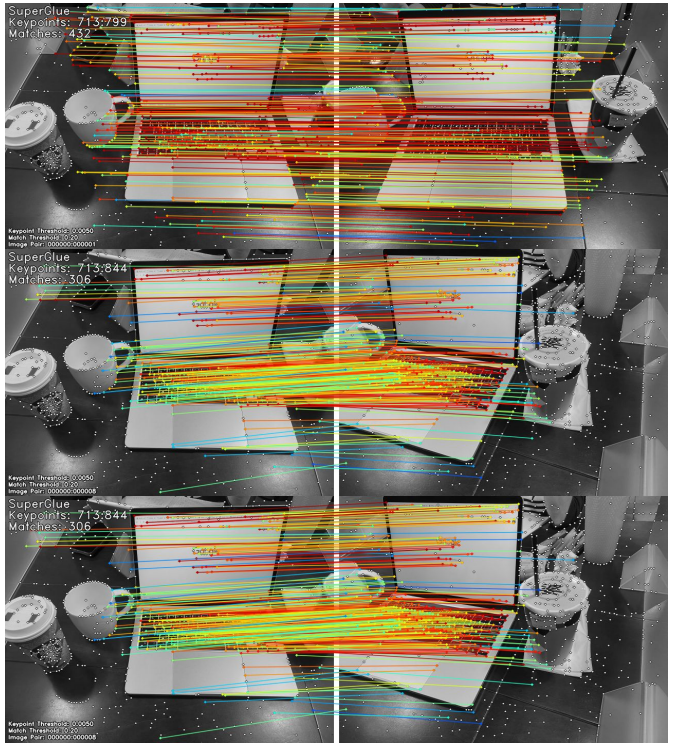
2.Superglue use ScanNet for indoor training.

230M for training, 1500 for testing

3.It operates on the full set of possible matches.

4.We tried lots of different setting.

(resize, max_keypoint, threshold)



Experiment 1-indoor, outdoor matching comparison

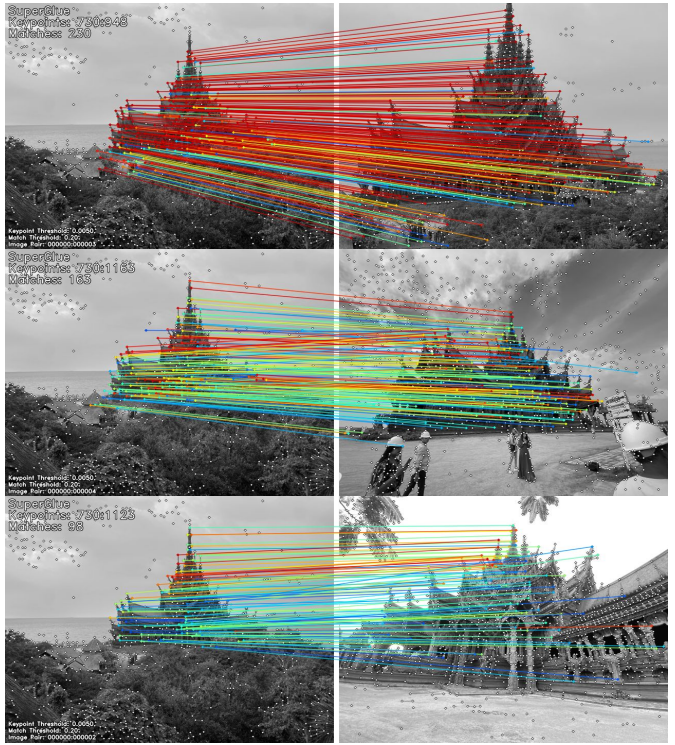
1.outdoor matching is challenging due to the lighting changes and occlusion.

2.Superglue use PhotoTourism for outdoor training.

3.We tried lots of different setting.

(resize, max_keypoint, threshold)

4.Both indoor and outdoor matching have better matching pairs and accuracy than others.



Experiment 2-Physarum Dynamics

- It is an efficient and differentiable solver for general linear programming problems which can be used in a plug and play manner within DNNs as a layer.

- Based on mathematical biology, often used in robust network design and solving problems such as shortest paths and LPs.

- We tried replacing the Sinkhorn Algorithm in SuperGlue with Physarum Dynamics

Why Physarum Dynamics?

- The Sinkhorn Algorithm in SuperGlue is used to solve the differentiable optimal transport problem, which is the same purpose as Physarum Dynamics.

- Physarum Dynamics can be used in a plug and play manner within deep neural networks as a layer, which converges quickly without the need for a feasible initial point.

Physarum Dynamics Algorithm

Algorithm 1: $\gamma-$AuxPD Layer

Input: LP problem parameters A, b, c, initial point x_0, Max iteration number K, step size h, accuracy level ϵ, approximate diameter γ_P

Set $x_s \leftarrow x_0$ if x_0 is provided else rand($[n], (0, 1)$)

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 $c_i = 0$

for $i = 1$ *to* K **do**

  Set: $W \leftarrow \text{diag}(x_s/c)$

  Compute: $L \leftarrow AWA^T$

  Compute: $p \leftarrow L^{-1}b$ using iterative solvers

  Set: $q \leftarrow WA^Tp$

  Update: $x_s \leftarrow (1-h)x_s + hq$

  Project onto $\mathbb{R}_{\geq\epsilon}$: $x_s \leftarrow \max(x_s, \epsilon)$

end

Return: x_s

Implement of Physarum Dynamics Algorithm

DataSet: COCO 2014 (25000 images)

GPU: V100 * 1

Epoch: 1

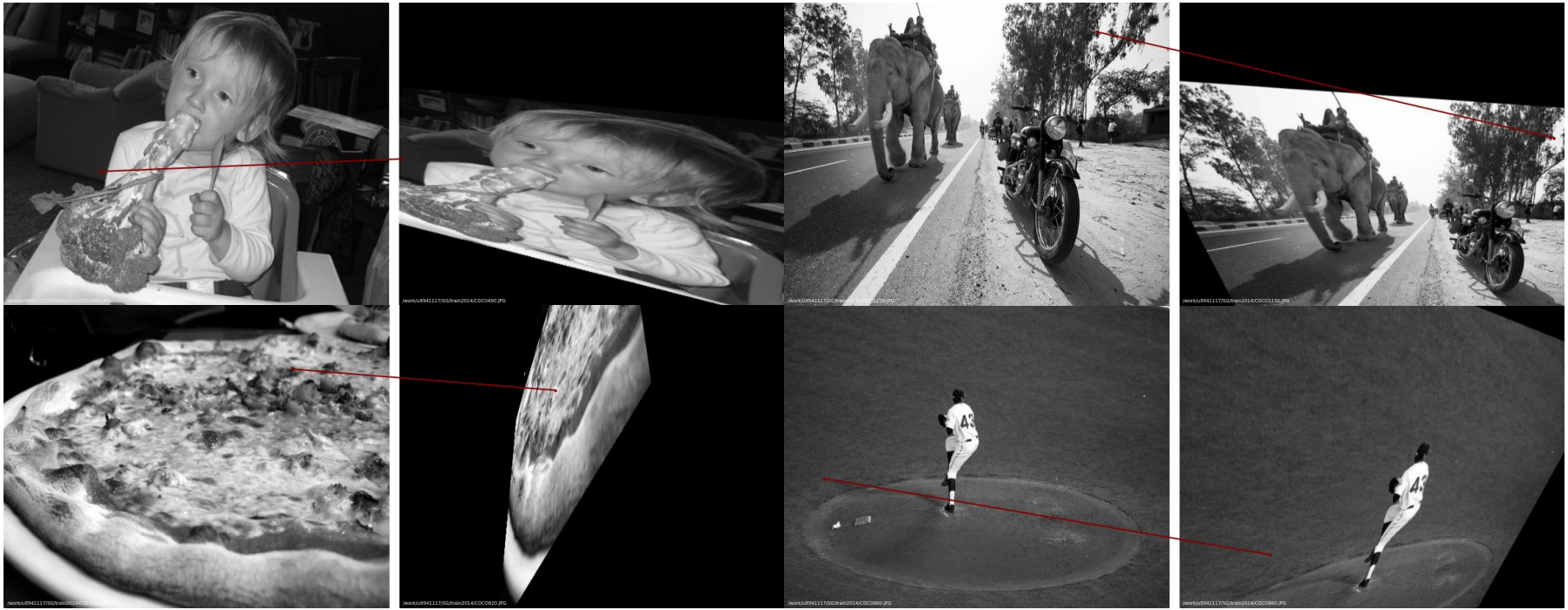
Max_KeyPoints_by_SuperPoints: 220

Inadequacy

- Limited by GPU

  If you need to find all Superpoint detector results, CUDA error reporting requires at least 4.1T of VRAM. I manually limited the maximum detection amount of superpoint to 220. This greatly affects the performance of the model. It also affects the speed of iteration.

Good Pairs

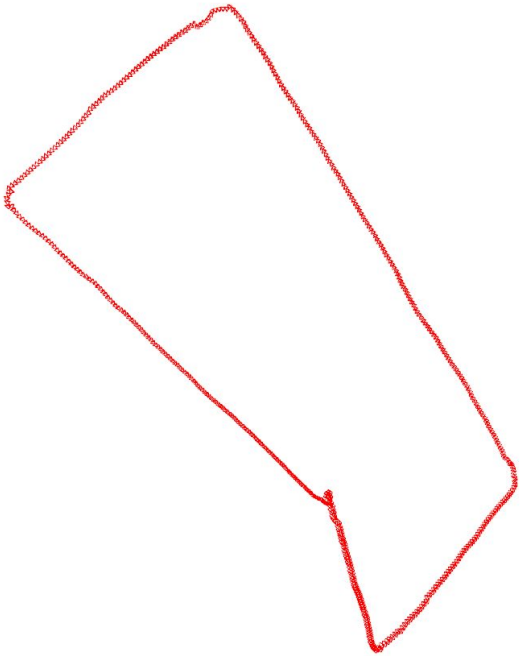


Bad Pairs

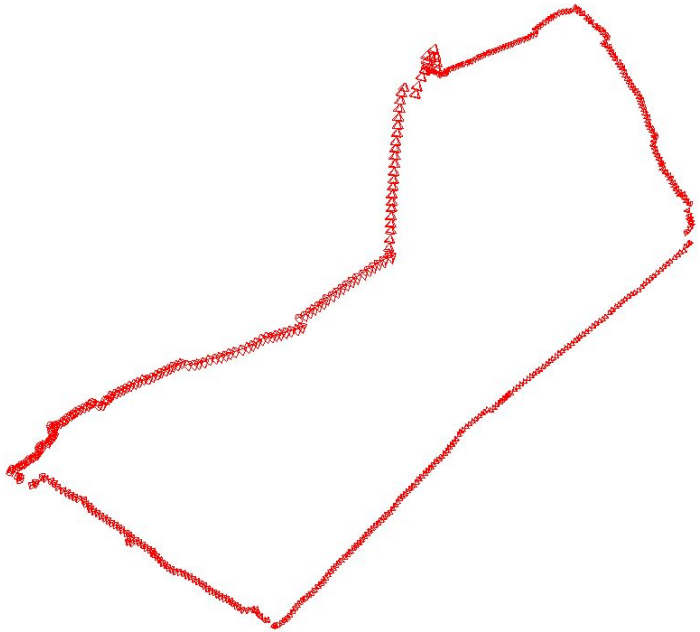
Experiment 3.1-visual odometry comparison

Dataset: HW3=>we can observe that the quality has improved and become less affected by the bumpy movement of the shooter.

Superpoint+superglue



ORB+Brute-Force Matcher



Experiment 3.2-trajectory comparison

1.dataset:KITTI odometry dataset

2.methods:We use different feature detectors and matchers to calculate the estimated pose and the corresponding trajectory.We compare them with the ground truth position.

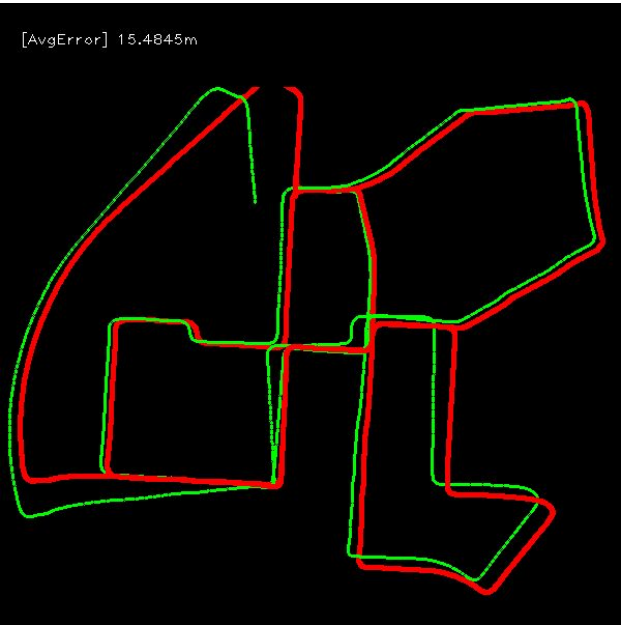
-superpoint+superglue

-superpoint+FLANN based Matcher

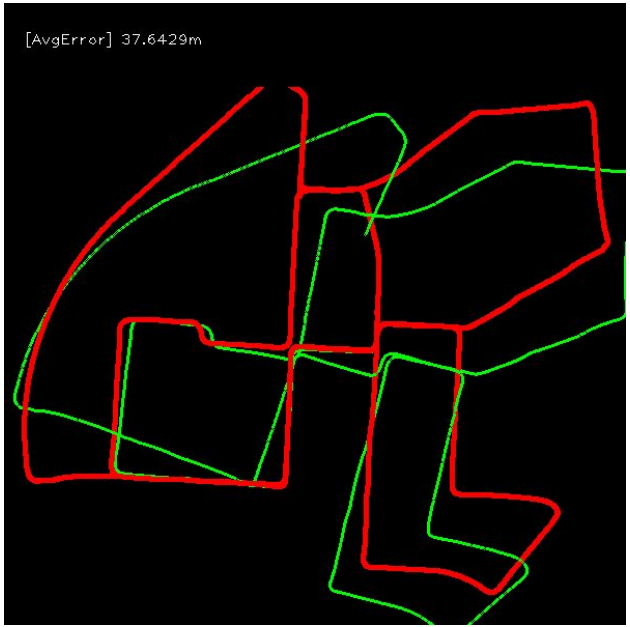
-ORB+Brute-Force Matcher

-

superpoint+superglue



superpoint+FLANN Matcher



ORB+Brute-Force Matcher

Result:(red line represents groundtruth, green line represents estimated trajectory)
on average,***superpoint+superglue*** method has around 15.4 m deviation from the groundtruth.
superpoint+FLANN matcher has around 37.6 m deviation from the groundtruth.
ORB+brute-force matcher has around 344 m deviation from the groundtruth.
=> superglue has improved the matching quality compared with other matchers.

Conclusion

1.Superglue is an effective method for feature matching which uses graph neural network and optimization algorithm. It increases the accuracy and stability of the matching pairs.
However, it requires large computation and slows down the speed of implementation compared to other methods. Hence, depending on which scenario you are, you could decide what algorithms to use.

2.Although Physarum Dynamics can replace the Sinkhorn Algorithm, it is only based on current experiments that it is impossible to effectively find an accurate matching point in a limited period of time, and resources and time are extremely needed

3.SuperPoint + Superglue pre-trained model the authors provided is better than others. It also provided different configurations for indoor scenes and outdoor scenes.

Division of work

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。github編輯

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、github編輯

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

Reference

- Paul-Edouard S., Daniel D., Tomasz M., Andrew R. (2020). 'SuperGlue: Learning Feature Matching with Graph Neural Networks'. CVPR 2020.
- Daniel D, Tomasz M, Andrew R. (2018). 'SuperPoint: Self-Supervised Interest Point Detection and Description'. CVPR 2018.
- 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: 26 November 2022).
- Magic Leap. (2020). 'Learning Feature Matching with Graph Neural Networks'. Available at: https://psarlin.com/superglue/ (Accessed: 27 November 2022).
- Hung-yi L. (2020) 'Graph Neural Network'. Available at: https://www.youtube.com/watch?v=eybCCtNKwzA (Accessed: 27 November 2022).
- Tomasz M. (2020). 'Deep Visual SLAM Frontends: SuperPoint, SuperGlue, and SuperMaps'. Available at: https://www.youtube.com/watch?v=u7Yo5EtOATQ (Accessed: 27 November 2022).