**Report: Using lines & Matching performance**

**August 25th, 2022**

1. **Previous works**

In previous meetings, I focused on point-matching algorithms based on binary descriptors. My experiments have been carried out on the Office sequences of the TartanAIR dataset, which have many line segments and repetitive patterns.

I introduced two improvement methods for the matching: (1) using Sinkhorn-Knopp and (2) using 3D stereo with the estimated pose (by MSCKF). Following the bellow figure, the Sinkhorn-3D result can close to the F1 score of Superglue.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Distance | Precision | Recall | F1 |
| Hungarian | *05-frames:* | 0.7284456 | 0.769619 | 0.748467 |
| *10- frames:* | 0.5460115 | 0.6083574 | 0.575501 |
| *20- frames:* | 0.3763342 | 0.4275599 | 0.400315 |
| Hungarian-3D | *05- frames:* | 0.8385701 | **0.823137** | **0.83078** |
| *10- frames:* | 0.7012934 | **0.714638** | 0.7079 |
| *20- frames:* | 0.5302445 | **0.541864** | 0.53599 |
| Sinkhorn | *05- frames:* | **0.961448** | 0.3477075 | 0.510715 |
| *10- frames:* | **0.939178** | 0.2506506 | 0.395697 |
| *20- frames:* | **0.898606** | 0.1557977 | 0.265554 |
| Sinkhorn-3D | *05- frames:* | 0.8862197 | 0.7597507 | 0.81242 |
| *10- frames:* | **0.817389** | **0.655074** | **0.71614** |
| *20- frames:* | **0.706716** | 0.4813608 | **0.55247** |
| Superglue | *05- frames:* | **0.937973** | **0.792673** | **0.85922** |
| *10- frames:* | **0.770154** | **0.687071** | **0.72624** |
| *20- frames:* | 0.581929 | **0.552207** | **0.56668** |

Figure 1: Comparison methods in Office scenario: best = bold green, second best = bold black

In this report, I continue this approach by conducting two tasks:

* First: using lines to improve matching.
* Second: comparing execution time of Superglue & Sinkhorn-3D match.

1. **Current works**
2. **Lines in matching**

We have two sets of keypoints extracted from the source and target image. A sub-set of source keypoints can represent a detected line in the source image. As a result, a line matching method is equivalent to point matching methods. To estimate the pose of this line in the target image, I match this sub-set to all target keypoints.

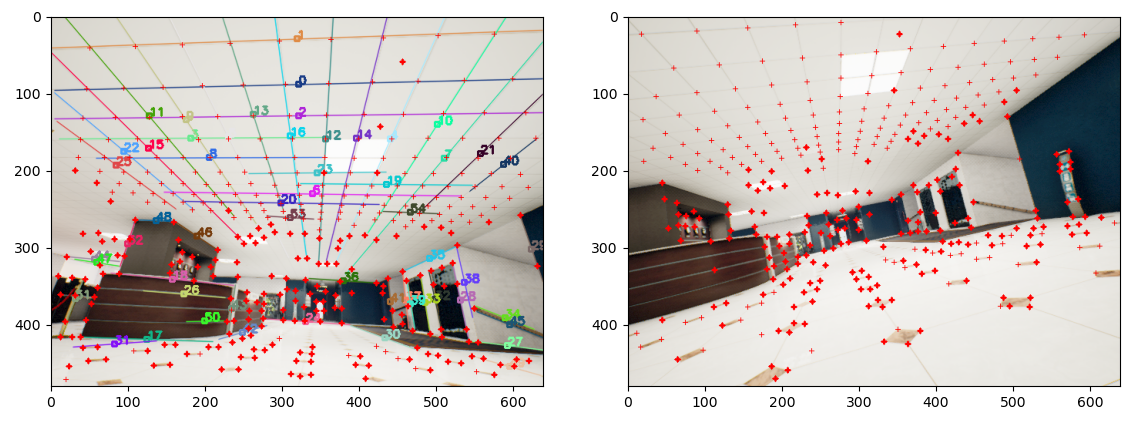


Figure 2: Detected lines & keypoints in source & target image (distance = 20 frame): Keypoints = red crosses, distinguished keypoints are bolder than repetitive keypoints.

I concentrate all lines matching results to a final matching table (Size = M x N, M = num source keypoints, N = num target keypoints) before adjusting new values in the cross matrix. In my experiments, this method is better than SuperGlue in some conditions but far worse overall (like naïve Hungarian with higher recall but F1 score is the same).

Because of this unstable matching result, I postponed this approach to rethinking my current line representation methods. I addressed two problems of that:

* Lack of order for points in a line.
* No geometry connection between points in the subset.

I must resolve these issues before proposing any matching algorithm.

1. **Point-matching performance in CPU & GPU**

Complete matching algorithms usually have two main steps: 1st – extracts points & descriptor in images, 2nd – perform matching. SuperGlue & Sinkhorn-3D is 2nd step, but SLAM applications need to consider both.

In this part, I compare experiments in CPU and GPUs; I also propose some speedup ways in SLAM practically.

1. ***With 8-Core CPU***

My matching method is about seven times faster than SuperGlue when executing in an 8-core CPU:

* Superglue match: **71 msec**
* Sinkhorn-3D match: **11 msec** (3.4 msec for Sinkhorn & 7.7 msec for Hungarian)

However, keypoints detection is time-consuming: 500 msec for each image. The total time is about 1100 msec for matching keypoints in two frames by Superglue.

1. ***SuperGlue (CNN) acceleration with GPU***

We all know that GPU can accelerate CNN inference; however, the limit FPS is 30 Hz because it needs 30-50 msec to call the execution from CPU to GPU.

Figure 3: Full matching time by SuperGlue in CPU, Mobile GPU, PC GPU

(\*) Embedded PCs (for drones) may have similar GPU to mobile laptop GPU, but their CPU is usually slower than the CPU used in this experiment.

1. ***Sinkhorn-3D acceleration in GPU***

My Sinkhorn-3D match has two parts: Sinkhorn-Knopp and Hungarians. In my current investigation, GPU can accelerate two times for Sinkhorn-Knopp and four times for Hungarian. Combined with optimized CNN keypoints extraction (the technique in TensorRT), full execution pipelines will spend about 100-150 msec (6-10 FPS), two times faster than Superglue in practical terms.

I can replace SuperPoint using SURF-GPU (synchronize time about 1 msec); the matching time can reduce to 40-50 msec (20-25 FPS). However, I do not have any accuracy test with SURF.

1. **Future works**

There are two main directions I need to focus on:

1. Line matching: solving the two issues in line representation. It will be more crucial when representing at the object level.
2. Sinkhorn-3D: complete the implementation for SLAM. Better point association / flow-motion can improve trajectory estimation in VIO.

(\*\*) SLAM baselines like ORB-SLAM-2, VINS-Fusion, and S-MSCKF have low success rates in TartanAIR because of the data association challenge (mainly aperture, occlusions).