Robust Tightly-Coupled Visual-Inertial Odometry with Pre-built Maps in High Latency Situations Supplementary Document

Hujun Bao, Weijian Xie, Quanhao Qian, Danpeng Chen, Shangjin Zhai, Nan Wang, Guofeng Zhang

1 Introduction

This is the supplementary document that our main paper refers to. It contains some additional implement details and additional experiment results.

2 AR Demo

We use panoramic camera[1] collected panoramic image data five months ago. After that, we use the SfM algorithm [4] to generate point clouds and keyframes. We further processed the point clouds using MeshLab[2] to obtain a simplified dense mesh. When a mobile phone first requests a localization module, the simplified mesh is sent to the mobile phone through the network. The feature detector we used in SfM and localization module is SuperPoint [1]. Fig. 1 is the model of the outdoor environment in the supplementary video, which was reconstructed by the SfM algorithm. After SfM reconstruction, we get a map consisting of keyframes and point clouds. When the localization module receives a request, the first step is to extract the SuperPoint features with descriptors from the request image. Then, we can get keyframes similar to the request image and corresponding point clouds through image retrieval. If enough 2D-3D matches can get by feature descriptor matching, we will apply a Perspective-n-Point solver inside a RANSAC loop [2, 3] to estimate the localization pose.

For comparison purposes, we have developed a loosely coupled demo based on ARCore, which we call ARCore-LC. ARCore-LC accepts the real-time pose $\left[\boldsymbol{R}_i^G, \boldsymbol{p}_i^G\right]$ from ARCore and send the localization request to the server at a specific time interval. After receiving the localization pose $\left[\boldsymbol{R}_i^M, \boldsymbol{p}_i^G\right]$, $\left[\boldsymbol{R}_G^M, \boldsymbol{p}_G^M\right]$ is updated as follows:

$$\begin{aligned} \boldsymbol{R}_G^M &= \boldsymbol{R}_i^M \boldsymbol{R}_i^{G^T} \\ \boldsymbol{p}_G^M &= \boldsymbol{p}_i^G - \boldsymbol{R}_M^G \boldsymbol{p}_i^M \end{aligned} \tag{1}$$

The pose $\left[\boldsymbol{R}_i^O, \boldsymbol{p}_i^O\right]$ of ARCore-LC real-time output is calculated as follows:

$$\begin{aligned} \boldsymbol{R}_i^O &= \boldsymbol{R}_G^M \boldsymbol{R}_i^G \\ \boldsymbol{p}_i^O &= \boldsymbol{R}_G^M \boldsymbol{p}_i^G + \boldsymbol{p}_G^M \end{aligned} \tag{2}$$

ARCore-LC will request localization results every 15 meters or 10 seconds.

3 Additional Experiment on Synthetic Dataset

We made statistics on the computational cost of each module. As Table 1 shows, with the use of map point, although the computation cost of the KLT tracking and solver has increased, the overall settlement time remains at a relatively low level. We also deploy the ray casting algorithm on the remote server in the implementation. So the result of the localization request will directly include the localization pose and point clouds. The computation cost of extracting point clouds by localization pose is included in localization latency. Therefore, only when the system uses the real-time pose to obtain the point cloud will it generate the computation cost in the local device. In addition, we implement projection-based matching in a separate thread to make VIO more efficient.

Fig. 1: The reconstructed 3D point cloud of an outdoor scene by SfM.

Table 1: Time statistics of each module on synthetic datasets in computer. *LP* stands for using real-time poses to obtain local map points.

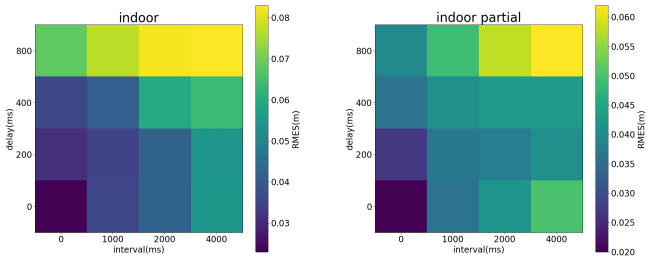
Module	BVIO	RTC-VIO	RTC-VIO w/o LP
Point Clouds extraction	/	20ms	/
Projection-based Matching	/	14ms	14ms
KLT tracking	12ms	13.90ms	13.37ms
Solver	2.2ms	7.9ms	4.9ms

We further evaluate the effects of localization time delay and localization frequency on the accuracy of the algorithm. Fig. 2 shows that with the increase of interval and time delay, the error will increase gradually. From Table 2, excluding *indoor partial*, we can find that local map point constraints have a noticeable effect on suppressing error accumulation, especially in the *outdoor* dataset. The *outdoor* dataset contains a large amount of simple plane structure such as ground, which is the reason why local map point constraints work well. In addition, our algorithm is particularly suitable for sequences like *indoor partial*, where the camera moves in a local region. In this way, we can quickly obtain enough global map points only after a few times of global localization. Since the field of view does not change significantly, the global map points can continue to take effect and suppress the accumulation of drift. In this case, our method can still work well at extremely low localization frequencies and high time delays.

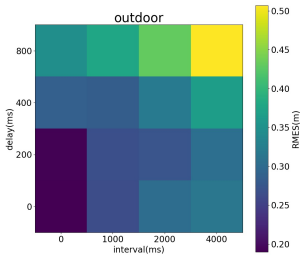
[1] https://www.insta360.com/cn/product/insta360-onex2
[2] https://www.meshlab.net/

Table 2: Evaluation on general localization performanceon synthetic dataset with APE (m). The parameter 'delay' means the time delay of localization(ms), the parameter 'interval' means the minimal interval between localizations (ms). We add the translation noise with a mean of 2cm and a variance of 0.25cm to ground-truth poses.

Dataset	indoor	indoor w/o normal	indoor partial	indoor partial w/o normal	outdoor	outdoor w/o normal
delay:0 interval:0	**0.023**	**0.023**	**0.020**	**0.020**	**0.190**	0.192
delay:0 interval:1000	0.036	**0.034**	0.036	**0.034**	**0.265**	0.304
delay:0 interval:2000	**0.042**	**0.042**	0.042	**0.038**	**0.303**	0.310
delay:0 interval:4000	0.055	**0.054**	**0.050**	**0.050**	**0.316**	0.343
delay:0 interval:8000	**0.063**	0.110	**0.055**	0.058	0.447	**0.409**
delay:0 interval:12000	**0.111**	0.136	**0.058**	0.063	0.585	**0.552**
delay:200 interval:0	**0.031**	**0.031**	0.027	**0.026**	**0.192**	0.296
delay:200 interval:1000	**0.035**	0.037	0.037	**0.036**	**0.267**	0.300
delay:200 interval:2000	**0.042**	0.046	**0.038**	0.040	**0.274**	0.321
delay:200 interval:4000	**0.054**	0.065	**0.041**	0.047	**0.306**	0.395
delay:200 interval:8000	**0.072**	0.084	**0.054**	0.060	**0.400**	0.491
delay:200 interval:12000	**0.119**	0.129	**0.053**	0.067	0.541	**0.539**
delay:400 interval:0	**0.036**	0.037	**0.036**	0.037	**0.287**	0.313
delay:400 interval:1000	**0.041**	0.049	0.041	**0.037**	**0.284**	0.327
delay:400 interval:2000	**0.060**	0.065	0.043	**0.042**	**0.319**	0.368
delay:400 interval:4000	**0.064**	0.068	**0.043**	0.052	**0.366**	0.403
delay:400 interval:8000	**0.076**	0.105	**0.052**	0.055	**0.528**	0.541
delay:400 interval:12000	**0.092**	0.129	0.061	**0.056**	**0.527**	0.588
delay:800 interval:0	**0.068**	0.085	**0.040**	0.042	0.347	**0.344**
delay:800 interval:1000	**0.077**	0.088	0.049	**0.046**	0.378	**0.330**
delay:800 interval:2000	**0.082**	0.131	**0.058**	0.063	**0.431**	0.432
delay:800 interval:4000	**0.083**	0.148	**0.062**	0.069	**0.507**	0.606
delay:800 interval:8000	**0.143**	0.193	**0.066**	0.078	**0.542**	0.685
delay:800 interval:12000	**0.149**	0.221	**0.067**	0.074	**0.619**	0.719



(a) (b)



(c)

Fig. 2: (a) The heat map of RMSE changes with internal and delay of sequence indoor. (b) The heat map of RMSE changes with internal and delay of sequence indoor partial. (c) The heat map of RMSE changes with internal and delay of sequence outdoor.

4 Comparison with Hololens

We make extensive comparison with Hololens 2[3]. In subsequent descriptions, we will refer to Hololens2 as Hololens for short. Since Hololens is a closed source commercial software, we can only analyze it through real-time experiments. As far as we know, unlike our method, which can utilize the pre-built map generated by the SfM algorithm, Hololens relies on the map generated by itself. So the accuracy of the map which Hololens can use depends on its tracking accuracy. To visualize the trajectory of the Hololens, we install an App called Graffiti 3D from the App Store. In the supplementary video of the Hololens test, we use the Graffiti 3D to draw the trajectory line as an AR effect. A person using HoloLens walked around the test scenario three times and recorded the AR result of Hololens 2. The purpose of the first round is to build the map. The following two rounds are designed to test the tracking accuracy of Hololens with the pre-built map created in the first round. We can see an apparent cumulative error in the first round from the video. Even if the loop closure occurs, the cumulative error is not completely eliminated. Repeating the same route still has almost the same accumulation drift and AR jumping problem.

Furthermore, we record the trajectory of Hololens, ARCore-LC, and our method according to the operation of the supplementary video. As shown in Fig. 3, Hololens has a significant accumulate error on the Z-axis, with a difference of 3 meters between the maximum and minimum heights. Compared with Hololens, our method can smoothly suppress the accumulated drift on each axis without abnormal jump changes like ARCore-LC.

References

[1] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pp. 224–236, 2018.

[3]https://www.microsoft.com/en-us/d/hololens-2/91pnzzznzwcp?activetab=pivot:overviewtab

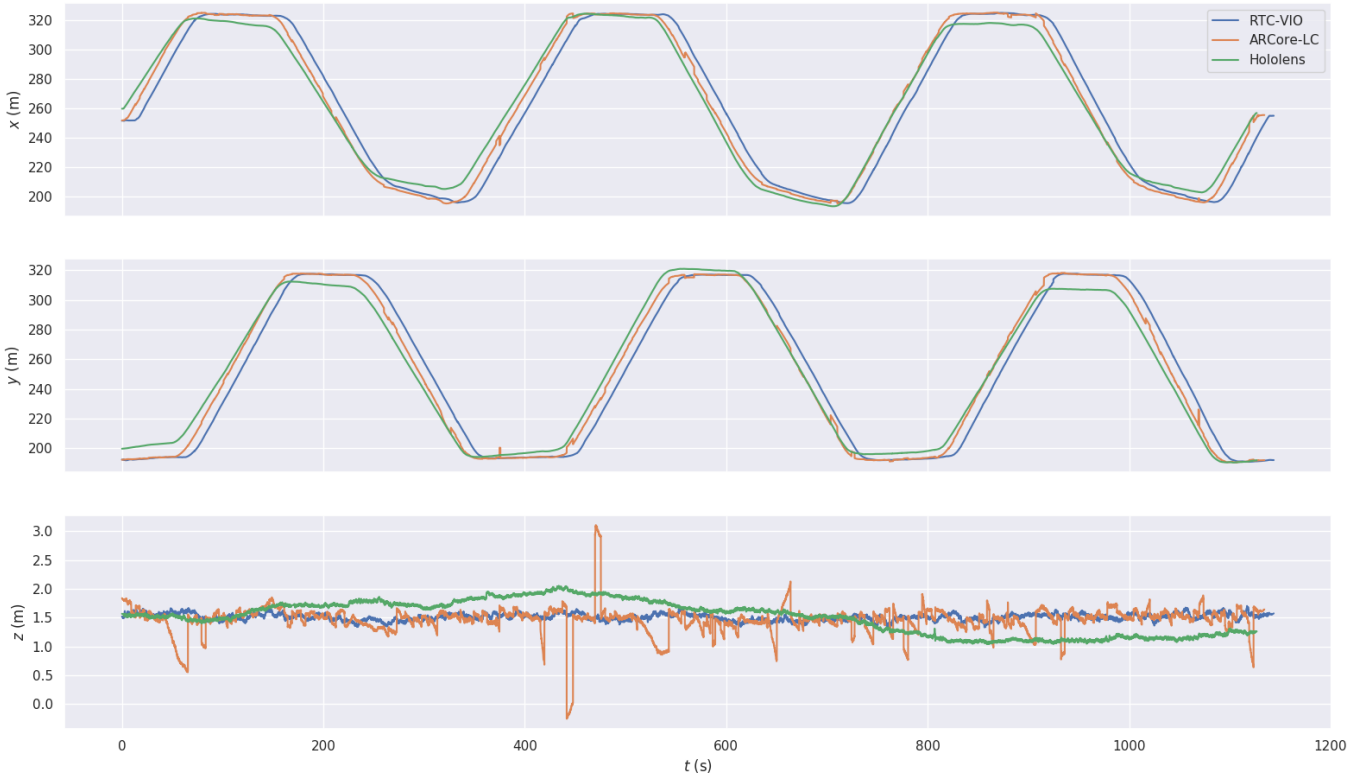


Fig. 3: The trajectory of Hololens, ARCore-LC, and our method. The z axis is the direction of gravity.

[2] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):930–943, 2003.

[3] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[4] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4104–4113, 2016.