Supplementary Document

Sang Jun Lee[1,2], Deokhwa Kim[1], Sung Soo Hwang[2], Donghwan Lee[1]
[1]NAVERLABS, Korea
[2]Handong Global University, Korea
{eowjd4}@naver.com, {deokhwa.kim, donghwan.lee}@naverlabs.com, {sshwang}@handong.edu

Abstract

This supplementary document provides derivation and formula about scale factor update and pose smoothing in more detail as mentioned in Section 3.4 and 3.6 respectively in the paper. Moreover, this document demonstrates more detailed method for the training of the proposed feature, SuperORB, with the evaluation.

1. Pose smoothing

In the proposed system, a local map with poses generated by the VO sub-system is aligned from local coordinates to global coordinates. At this moment, the local map is aligned based on a pose estimated by the VL sub-system, it means that the alignment performance depends on the quality of localized poses. However, sometimes when the VL pipeline estimates the wrong pose, the effect of the alignment of the VO pose (for convenience we call VO/VL pose as the pose estimated from the VO/VL sub-system) should be reduced in the result of estimating the wrong pose using pose smoothing technique. Given a pair of poses between the VL sub-system and the VO sub-system, we make VL pose smooth by regularizing the relative translation and rotation from VO pose with the weight w. Let **K** is the intrinsic matrix and **c, t, R** respectively indicates the center of the camera, the translation vector, and the rotation matrix. When $(\cdot)_{vo}$ and $(\cdot)_{vl}$ indicate the vector or matrix estimated from the VO sub-system and VL sub-system, the weight value w is obtained as:

$$w = \frac{1}{e}(\text{if } e < 1 \text{ then } w = 1). \tag{1}$$

where $e = \frac{1}{P}\sum_i^P (x_i - \mathbf{K}(\mathbf{R}_{vl}X_i + \mathbf{t}_{vl}))^2$ is the projection error between a projected 2D point x from a 3D point X, and P is the number of visible 3D points. If the 6 DoF pose estimated by VL sub-system is accurate, the projection error is decreased. Then, the weighted VL pose can be estimated as:

$$\mathbf{R}^*_{vl} = \mathbf{R}_{vo}\mathbf{R}^w_{rel} \tag{2}$$

$$\mathbf{t}^*_{vl} = -\mathbf{R}^*_{vl}\mathbf{c}^w_{vl}, \tag{3}$$

where $\mathbf{c}^w_{vl} = \mathbf{c}_{vo} + w \cdot \mathbf{c}_{rel}$, and $\mathbf{c}_{rel} = \mathbf{c}_{vl} - \mathbf{c}_{vo}$. Derivation of weighted translation is relatively easier than rotation case. For the case of rotation, we obtain rotation axis **a** and rotation angle θ as follows; rotation axis **a** can be obtained by:

$$\mathbf{a} = \frac{\mathbf{v}_{vo} \times \mathbf{v}_{vl}}{\|\mathbf{v}_{vo} \times \mathbf{v}_{vl}\|}, \tag{4}$$

where two principal axes $\mathbf{v}_{vo} = \det(\mathbf{M}_{vo})\mathbf{m}^3_{vo}, \mathbf{v}_{vl} = \det(\mathbf{M}_{vl})\mathbf{m}^3_{vl}$, $\det(\cdot)$ indicates determinant of a matrix, and **M=KR**, and $\mathbf{m}^3$ denotes the third-row vector of **M**. The rotation angle θ can be easily obtained by:

$$\theta = \arccos\frac{\mathbf{v}_{vo} \cdot \mathbf{v}_{vl}}{\|\mathbf{v}_{vo}\| \, \|\mathbf{v}_{vl}\|}. \tag{5}$$

Finally, the weighted relative rotation for weighted rotation matrix is calculated by using Rodrigues formula [5] given **a**, θ, w and as:

$$\mathbf{R}^w_{rel} = \mathbf{I}+\sin(w\theta)\mathbf{A} + (1 - \cos(w\theta))\mathbf{A}^2, \tag{6}$$

where **A** is the skew matrix of the rotation axis **a**.

2. Scale update

The scale factor between the offline and the online maps is needed to be converged to an appropriate value when it is updated. To converge the scale factor, recursive Bayesian estimation can be derived. In the form of recursive Bayesian estimation [3],

$$p(s_t|M_{1:t}) \propto p(M_t|s_t)p(s_t|M_{1:t-1}), \tag{7}$$

where s_t is the estimated scale factor in time t, and M_t denotes the matching pairs on corresponding super key frames between offline and online maps. Then the prediction term is:

$$p(s_t|M_{1:t-1}) = \int p(s_t|s_{t-1})p(s_{t-1}|M_{1:t-1})ds_{t-1} \quad (8)$$

where $p(s_t|s_{t-1}) \sim N(s_{t-1}, \sigma_t^p)$ is a motion model as a transition, $\sigma_t^p = \sigma_{min} + (\sigma_{max} * \theta)/\theta_{min}$ proposed in [7] and θ can be obtained as like in Equation (5).

The update term becomes $p(M_t|s_t) \sim N(s_t, \sigma_t^m)$ where s_t and σ_t^m are respectively mean and standard deviation of relative scale factor for all inlier matching pairs of 3D points in M_t, but we take s_t as a relative scale factor that has minimal Euclidean distance error for all inliers in order to avoid the explosion of scale factor in an update that is occurred in the inaccurately generated 3D point pairs.

If the motion model is linear transition and the noise error is normal distribution as $s_{t+1} = s_t + v$ where $v \sim N(0, \sigma_t^p)$, recursive Bayesian estimation is equal to Kalman Filter as discussed in [3], so that it can be derived as follows.

Let $\widehat{x}_{t,t-1}$ and$\widehat{P}_{t,t-1}$ are a state and a state uncertainty in prediction step predicted from trial $t-1$ to t, and $\widehat{x}_{t,t}$ and $\widehat{P}_{t,t}$ are the updated state and its uncertainty in update step. In general, an estimated scale factor affects to next trial because it applies to alignment process. So, in every trial t, before calculating s_t and σ_t^m for the relative scale factor, the 3D points in online map is aligned to original scale dividing by $\widehat{x}_{t-1,t-1}$. And final relative scale factors_t^* after applying Kalman Filter becomes $s_t^* = \widehat{x}_{t,t}/\widehat{x}_{t-1,t-1}$. This is for converging to an absolute relative scale factor.

Therefore, In initial time (t = 1), $\widehat{x}_{1,1} = s_1$, and to get s_1, $\widehat{x}_{0,0}$ is set to 1 to scale 3D points in online map described above. And for next trial (t >1), in prediction step:

$$\widehat{x}_{t,t-1} = \widehat{x}_{k-1,k-1} \quad (9)$$

$$\widehat{P}_{k,k-1} = \widehat{P}_{k-1,k-1} + \sigma_k^p. \quad (10)$$

In update step:

$$K_k = \frac{\widehat{P}_{k,k-1}}{\widehat{P}_{k,k-1} + \sigma_k^m} \quad (11)$$

$$\widehat{P}_{k,k} = (1 - K_k)\widehat{P}_{k,k-1} \quad (12)$$

$$\widehat{x}_{k,k} = \widehat{x}_{k,k-1} + K_k(s_k - \widehat{x}_{k,k-1}), \quad (13)$$

where K_k is the weight to leverage the measurement.

3. SuperORB

3.1. Architecture

SuperORB's network architecture is the same as SuperPoint's architecture [2], except that the dimension of output feature is 64 instead of 256. We used a smaller feature dimension is to speed up the pipeline of the visual localization sub-system.

3.2. Training details

SuperPoint and SuperORB are trained as follows:

Step 1. Train only the interesting points detector. The pseudo ground truth is obtained as follows:

- SuperPoint: using synthetic images
- SuperORB: applying ORB detector to real-world images.

Step 2. Extracting interesting points by inferencing the trained detector.

Step 3. Applying homographic transformations to the extracted interesting points and aggregate them.

Step 4. Jointly training the detector and the descriptor by using pseudo ground truth in Step 3.

The only difference between the SuperPoint and SuperORB is Step 1; The former uses interesting points from synthetic images while the latter utilizes the ORB detector for generating pseudo ground truth. We trained SuperPoint and SuperORB features by using the MS-COCO dataset [4] and training splits of 7 scenes dataset [6]. Note that only the MS-COCO dataset is used in [2]. The Hyperparameters are the same as [2].

3.3. Evaluation

In our experiments, we evaluate SuperPoint and SuperORB by using HPatches dataset [1]. First, we measure detector repeatability on the HPatches dataset. In this experiment, we resized the input images to 240 x 320 resolution and extract 300 key points for each image. We set the non-maximum suppression (NMS) size to 4 and use a correct distance of 3 pixels.

We also compare the homography estimation accuracy of SuperPoint and SuperORB on HPatches dataset. We extract a maximum of 1,000 key points from each input image that is resized to 480 x 640 resolution. We used a correct distance of 3 pixels. The experimental results are shown in Table 1. SuperPoint-64 that are trained in this work shows better performance when compared to original SuperPoint, i.e. SuperPoint-256, even though they have smaller feature sizes. When comparing SuperPoint and SuperORB, the former shows comparable detector repeatability but outperforms in homography estimation accuracy with a large margin.

	Dim.	Rep.	Homography
SuperPoint-256 [2]	256	0.576	0.684
SuperPoint-64	64	0.579	0.731
SuperORB-64	64	0.584	0.634

Table 1. HPatches detector repeatability and homography estimation results.

	chess	stairs	heads	office	pumpkin	redkitchen	fire
SuperPoint-64	0.76	0.63	0.66	0.70	0.70	0.68	0.43
SuperORB-64	**0.84**	**0.75**	**0.74**	**0.79**	**0.78**	**0.72**	**0.84**

Table 2. Recall for detecting local map points in visual odometry systems.

SuperORB is designed to find interesting points that are close to the ORB features for robust scale estimation. In Table 2, we present the recall for detecting local map points, i.e. ORB key points, when utilizing the SuperPoint and SuperORB detectors. Note that test sequences in the 7 scenes dataset are used for this experiment. The recall is defined as follows:

$$\text{recall} = \frac{\text{\# of detected ORB features}}{\text{\# of ORB features}}. \tag{14}$$

As shown in Table 2, SuperORB detects more map points construct by the visual odometry sub-system than SuperPoint. As a result, when applying SuperORB, we can more robustly estimate the relative scale factor between the local and the global maps.

References

[1] V. Balntas and et al. Hpatches: A benchmark and evaluation of handcrafted and learned local descriptors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[2] D. deTone and et al. Superpoint: Self-supervised interest point detection and description. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018.

[3] et al. J. Diard. A survey of probabilistic models using the bayesian programming methodology as a unifying framework. 2003.

[4] T. Y. Lin and et al. Microsoft coco: Common objects in context. *European conference on computer vision, Springer, Cham*, 2014.

[5] J. G. Ratcliffe. Foundations of hyperbolic manifolds. *New York. Springer-Verlag*, 3, 1994.

[6] J. Shotton and et al. Scene coordinate regression forests for camera relocalization in rgbd images. *In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013.

[7] E. Sucar and J. B. Hayet. Bayesian scale estimation for monocular slam based on generic object detection for correcting scale drift. *International Conference on Robotics and Automation (ICRA). IEEE*, 2018.