The Unreasonable Effectiveness of Pre-Trained Features for Camera Pose Refinement (Supplementary)

Gabriele Trivigno[1] Carlo Masone[1] Barbara Caputo[1] Torsten Sattler[3]
[1] Politecnico di Torino
[2] Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague
{gabriele.trivigno,carlo.masone,barbara.caputo}@polito.it torsten.sattler@cvut.cz

Supplementary

In this supplementary material we show:

- an ablation on different scoring functions to demonstrate the effectiveness of simple pixelwise comparison, as mentioned in L500 of the main paper;
- a convergence analysis to test the robustness of our algorithm to initialization, as discussed in L422 of the main paper;
- additional insights on hyperparameters;
- a discussion on inference time;
- a pseudo-code version of our algorithm.

1. Scoring functions

In our paper we showed the effectiveness of dense, pre-trained features for assessing pose similarity, against the previous methods that adopt sparse, specialized features. The main idea behind these experiments is to test whether dense features provide an actual advantage or if the same results would hold for sparse comparisons as well. Thus, we devised several alternative scoring functions that could be used to rank candidates using sparse comparisons.

As a reminder, this scoring function is needed to compute the loss from Eq. 1 of the main paper ($\mathcal{L}_{\mathcal{F}_\theta}(T|I_q, I_T)$), which at each step is used to compare the rendered candidates against the query, ranking them. In Tab. 1 we compare the following cost functions:

- (1): the scoring function adopted in our method; detailed in Eq. 2 of the main paper, *i.e.*, the pixelwise L2 distance between feature maps, normalized along the channels;
- (2): a straightforward alternative to dense comparison is to use exhaustive matching of detected keypoints. To this end, we use ALIKED [13] to obtain for each image a set of keypoints and associated descriptors $\{k_i, f_i\}, k_i \in \mathbb{R}^2, f_i \in \mathbb{R}^d$. Computing the mutual nearest neighbors between the descriptors of the query I_q and a candidate I_T, we obtain a set of matched keypoints $\{k_i, k_j\}, i \in K_{I_q}, j \in K_{I_T}$. Finally, the score is the reprojection error among matched keypoints, *i.e.*, their spatial distance (in pixel space). Thus:

$$\mathcal{L}_{\mathcal{F}_\theta}(T|I_q, I_T) = \sum_{i \in K_{I_q}, j \in K_{I_T}} ||k_i - k_j||^2 \ . \quad (1)$$

- (3): exhaustive matching increases significantly the cost of computing the loss. Moreover, keypoints that are in opposite locations in the considered image pairs do not provide a useful signal for refining the pose. Thus, a natural alternative to reduce the cost is to match keypoints locally. In this scenario, the nearest neighbors are computed only for keypoints that satisfy $||k_i - k_j||_2 \leq W$, where W is the patch size that defines the local window around each keypoint in which we compute matches.
- (4): implicit matching is an intriguing concept proposed in [5]. The main idea is that a standard CNN can be used to extract keypoints, in place of a dedicated keypoint detector. The assumption is that in such networks, each channel has learnt to detect a certain kind of features; thus by looking for local maxima in each channel of the feature maps, these spatial location can be compared among pairs of images without matching descriptors. To test this approach, given a feature volume $F_l \in \mathbb{R}^{C,H,W}$, we compute, for each channel: $k_c = \underset{h,w \in H,W}{argmax}\ F_l^{c,h,w}$. They are extracted both for the query k_c^q, and a candidate k_c^T.
To reduce noise we smooth these locations by applying a gaussian filter over a window of size W and then compare them:

$$\mathcal{L}_{\mathcal{F}_\theta}(T|I_q, I_T, l) = \sum_{c \in C_l} ||k_c^q - k_c^T||^2 \ . \quad (2)$$

Results. Results in Tab. 1 show that among scoring functions based on sparse comparisons (2, 3, 4), performances are proportional to the computational cost, *i.e.*, the more accurate is (2) which is also the more expensive. Overall, simple dense comparison (1) is the best performing one, while being also lightweight and hyperparameter-free. This effect can be understood in light of the discussion in Sec.

Scoring function	ShopFacade
(1) Dense Comparison	12 / 0.45
(2) Exhaustive Matching	20 / 0.93
(3) Patch-wise Matching	34 / 1.36
(4) Implicit Matching	85 / 1.92

Table 1. **Ablation on scoring function**. Shows the effectiveness of densely comparing feature maps against more sophisticated cost functions. Median errors reported in $cm/°$.

Coarse Features	Fine Features	ShopFacade	OldHospital
CNN features: ResNet-18			
CosPlace [2]	ImageNet	12 / 0.45	39 / 0.73
ImageNet	ImageNet	12 / 0.55	46 / 0.80
SimCLR [3]	SimCLR [3]	18 / 0.62	50 / 0.83
Transformer: ViT small			
DINOv2 [9]	DINOv2 [9]	34 / 0.81	59 / 1.15

Table 2. **Ablation on feature extractors**, to test whether the property of dense features being robust estimators of *visual similarity*, which is traditionally associated with feature maps from CNN architectures, holds for state-of-the-art vision transformers. Median errors in $cm/°$.

3.1 of the main paper: comparing dense features allows to fully exploit the properties of deep networks as *perceptual similarity* [11] estimators, and it provides a smoother signal (w.r.t. sparse features) thanks to the spatial structure of feature maps.
This property of dense feature maps was one of the core ideas behind our paper. While this effect was studied mainly with feature maps from CNN architectures [1, 8, 11], in Tab. 2 we experiment with the state-of-the-art vision transformer trained in DINOv2 [9]. In these architectures each image patch is encoded and processed as a token. In order to use this model for our algorithm, we compute the distance between corresponding tokens in a pair of images, using different layers of the encoder to preserve our *Coarse-to-Fine* approach.
While these tokens, paired with positional encoding, preserve spatial information, we find that using these features yields only adequate results, much lower than what can be achieved with a simple ResNet-18. These findings can be explained in light of the receptive field of each token being constrained to be equal or higher than the patch size (14 pixel specifically), and the fact that the self-attention scheme embeds some global context into each patch. This argument was recently sustained in RoMa [6], which proposes to refine DINOv2 features with a specialized CNN architecture. In Tab.5 of the main paper, we experiment with this architecture and find that it surpasses or match other specialized pose refiners, as shown in the table. Note that RoMa features rely on an architecture with roughly 80x more parameters than the ResNet-18 that we adopt.

2. Optimization hyperparameters

In this section we provide additional insights and ablations on some key hyperparameters of our algorithm. As discussed in Sec. 4.1 of the main paper, a key element for the success of our pose refinement is exploiting a *Coarse-to-Fine* approach, where we gradually move from deeper features to shallower ones. Given that we employ 3 different feature levels (coarse-medium-fine), this entails choosing 2 hyperparameters, namely N_1 and N_2. N_1 indicates after how many steps we switch from coarse to medium; N_2, reported as a negative value, represents that the last N_2 steps are carried out with the shallower features. Tab. 3 reports results on 2 Cambridge scenes, showing the effect of these 2 values. For these experiments, when varying N_1, we keep fixed the number of steps after N_1. When changing N_2, the number of steps before is fixed.

Another important part of our method is multi-hypothesis tracking [4], optimizing independently multiple *beams*. In principle, using more beams should always improve results, although this assumption does not hold if the total number of candidates sampled at each step is fixed, which is desirable in order to contain computational cost. Thus, we study this trade-off in Fig. 1, where we ablate the effect of not using beams at all (*i.e.*, $nbeams = 1$), or more. In our main experiments we use 3 beams. The number of candidates is 50 in the beginning, and it is slowly reduced to 20 in the last steps.

N_1	N_2	ShopFacade	OldHospital
15	-10	16 / 0.74	50 / 1.42
30	-10	<u>12</u> / <u>0.45</u>	<u>39</u> / <u>0.73</u>
50	-10	13 / 0.47	41 / 0.74
30	0	20 / 0.50	43 / 0.80
30	-20	12 / 0.43	37 / 0.72
30	-30	10 / 0.42	36 / 0.70

Table 3. **N. of steps before switching to coarser features**. We test different values of N_1 (switch from coarse to mid-level features), and N_2 (switch to finer features). Underlined values are the default used in the main paper. Median errors reported in $cm/°$.

Results. As we state in the main manuscript, our algorithm is robust to the choice of the values of (N_1, N_2), provided

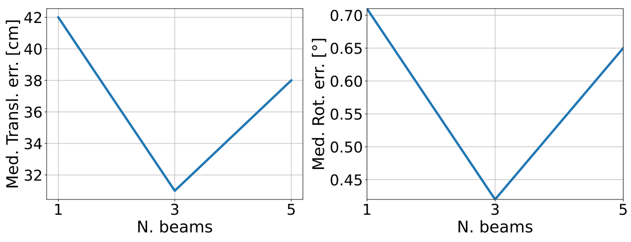

Figure 1. **Number of independent beams**. Results on KingsCollege.

that enough steps are performed with coarse features in the beginning. This is because, despite the fact that all feature levels exhibit a convex basin around each pose, the convergence basin is narrower for shallower features. Since initialization from retrieval can yield large baselines, it is important to rely on coarse features for enough steps to refine the pose just enough to fall into the convergence basin of the next finer levels.
This is apparent from Tab. 3, as it shows that doing too few steps with *conv3* features ($N_1 = 15$) has the biggest impact on performances. On the other hand, doing more steps does not harm performances, although it makes convergence slower.
Regarding the value of N_2, doing more steps improves results, however the improvement is small, and for this reason we kept it to -10 to exploit the best trade-off between cost and performance gain.

Fig. 1 proves the usefulness of relying on multiple optimization threads (*beams*) in parallel. However, using too many beams is also counterproductive; since the number of candidates is the same in these experiments, if the number of beams increases, each beam will sample less candidates, thus reducing their ability to explore the state space, and ultimately harming performances.

2.1. Convergence analysis

As with any refinement algorithm, the accuracy of the initial poses is a crucial factor that affects convergence speed, as well as performances. To study the sensitivity of our method to the initial error, we perform the following experiment, similarly to [12]: we randomly perturb the ground truth poses with different error magnitudes, and then run our algorithm for a fixed number of steps. We use magnitudes of $1, 5, 10, 15$ meters and $5, 10, 20, 30$ degrees, and for each magnitude we repeat the sampling 10 times to carry out a more robust analysis.
These experiments are performed on ShopFacade, and we run our optimization for 40 steps. Note that results in the main paper for Cambridge scenes are obtained with 80 steps; in this setup we used less iterations due to the high number of combinations and repetitions of each experiment (160 runs in total). The emerging trends and the conclusions hold nonetheless.

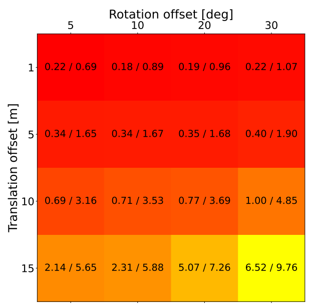

Figure 2. After 20 steps

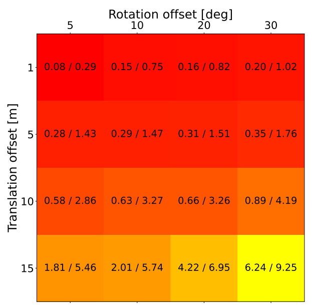

Figure 3. After 40 steps

Figure 4. **Convergence analysis.** For the scene of ShopFacade, we randomly perturb ground truth poses with fixed magnitudes of error, and then run our optimization to asses its robustness to initialization. Numbers are reported as median errors, averaged over 10 runs, as $m/°$.

Results. Fig. 4 displays in a matrix the results for each translation/rotation combination of errors, after 20 and 40 iterations. At a glance, it is evident from the color map that the obtained accuracy is more correlated with translation error. This effect is understandable as details of the scene might be less recognizable from a distance, thus falling outside of the convergence basin. On the other hand, at a close distance, our optimization can recover from a high rotation error even if there is very little overlap in the views.
Overall, our algorithm is robust to errors up to 5 meters, regardless of the rotation, while performances start to degrade at 10 meters.

3. Inference cost

We do not claim inference speed among the selling points for our method, since in the literature we did not find a reliable comparison on the same hardware among different methods and implementations. Nonetheless, we report here a breakdown of the time required to optimize a pose over 80 iterations, which is the number of steps that we used to obtain results for Cambridge scenes. Times are measured on a RTX4090. Rendering the Gaussian cloud from [7] takes $0.8ms$; and in total we render 2600 candidates for each query over the steps.
Extracting features with a truncated ResNet-18, with FP16 precision and batching takes $0.1ms$ per image at the lowest resolution (256×320). At the highest resolution that we use (320×480), it takes $0.2ms$. Considering that we use 3 beams, our approach takes on average $2.4s$ for Cambridge, and about $8.7s$ for Aachen (as we perform more iterations). Our optimization relies on independent beams, which can be implemented with multiprocessing, reducing runtime re-

spectively to $1.1s$ and $4.5s$ on the same hardware. When used to refine HLoc poses, we use only 5 iterations, which takes as little as 200ms.

3.1. Comparison with PixLoc

On our RTX4090, PixLoc takes $3.1s$ per query independently of the scene. Our method is more versatile as it does not require any training, and it can be coupled with any dense scene representations, whereas PixLoc requires E2E training and a point cloud. Tab.4 of the main paper shows how our method can be useful as an efficient preprocessing steps in the setting proposed in [10], with different kinds of meshes. Our method also works better than PixLoc as a post-processing step on HLoc poses, and on night queries. On indoor datasets and small outdoor scenes, PixLoc achieves superior results, although being slower.

4. Algorithm pseudocode

Below in Algorithm 1 we provide a high-level pseudo-code of our algorithm. It highlights: (i) the *render&compare* structure of our approach, (ii) the fact that the model that we use is a function of the step, (iii) that the particle filter, and thus the noise applied during sampling, are also a function of the step.
It does not contain, for simplicity, the multiple beams which are optimized in parallel, or other low-level details.
More in detail, the pseudo-code shows, starting from the initial estimate (est_center, est_qvec), a loop for each query where, in each step:

- The number of candidates (variable N_cand), and the noise magnitude ($noise\_t, noise\_R$) are obtained deterministically as a function of the step;
- the particle filter, based on the noise magnitude and current pose estimate, is used to sample N_cand new hypothesis;
- the model is obtained as a function of the step (the backbone will be truncated at a certain layer), and it is used to extract features from the query q_feats and the sampled candidates $rend\_feats$;
- given the features, the sampled candidates are given a score by the function $rank\_poses$; finally the scores are used to update the current estimate

We will release our implementation publicly upon acceptance, as we believe it can prove useful to the community.

Algorithm 1 MCLoc pose refinement

```
$N \leftarrow n\_steps$
$renderer \leftarrow load\_scene\_model()$
for $query \in query\_list$ do
    $est\_center, est\_qvec \leftarrow init\_pose()$
    for $step \in 1..N$ do
        $N\_cand \leftarrow get\_N\_cand(step)$
        $noise\_t, noise\_R \leftarrow get\_perturb\_pars(step)$
        $sampler \leftarrow part\_filter(N\_cand, noise\_t, noise\_R)$

        $poses \leftarrow sampler.sample(est\_center, est\_qvec)$
        $renders \leftarrow renderer(poses)$

        $model \leftarrow get\_model(step)$
        $q\_feats \leftarrow extract\_features(query, model)$
        $rend\_feats \leftarrow extract\_features(renders, model)$

        $center, qvec \leftarrow rank\_poses(q\_feats, rend\_feats)$

        $est\_center, est\_qvec \leftarrow update(center, qvec)$
    end for
end for
```

References

[1] Seyed Ali Amirshahi, Marius Pedersen, and Stella X. Yu. Image quality assessment by comparing cnn features between images. In *Image Quality and System Performance*, 2016. 2

[2] Gabriele Berton, Carlo Masone, and Barbara Caputo. Rethinking visual geo-localization for large-scale applications. In *CVPR*, 2022. 2

[3] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[4] Changhyun Choi and Henrik Christensen. Robust 3d visual tracking using particle filtering on the special euclidean group: A combined approach of keypoint and edge features. *Intl. Jour. of Robotics Research*, 33, 2012. 2

[5] Titus Cieslewski, Michael Bloesch, and Davide Scaramuzza. Matching features without descriptors: implicitly matched interest points. *arXiv preprint arXiv:1811.10681*, 2018. 1

[6] Johan Edstedt, Qiyu Sun, Georg Bökman, Mårten Wadenbäck, and Michael Felsberg. Roma: Revisiting robust losses for dense feature matching. *arXiv preprint arXiv:2305.15404*, 2023. 2

[7] Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, and George Drettakis. 3d gaussian splatting for real-time radiance field rendering. *ACM Transactions on Graphics*, 42 (4), 2023. 3

[8] Jongyoo Kim and Sanghoon Lee. Deep learning of human visual sensitivity in image quality assessment framework. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[9] Maxime Oquab, Timothée Darcet, Theo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Russell Howes, Po-Yao Huang, Hu Xu, Vasu Sharma, Shang-Wen Li, Wojciech Galuba, Mike Rabbat, Mido Assran, Nicolas Ballas, Gabriel Synnaeve, Ishan Misra, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. Dinov2: Learning robust visual features without supervision, 2023. 2

[10] Vojtech Panek, Zuzana Kukelova, and Torsten Sattler. MeshLoc: Mesh-Based Visual Localization. In *European Conference on Computer Vision (ECCV)*, 2022. 4

[11] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 586–595, 2018. 2

[12] Zichao Zhang, Torsten Sattler, and Davide Scaramuzza. Reference pose generation for long-term visual localization via learned features and view synthesis. *International Journal of Computer Vision*, 129:821–844, 2021. 3

[13] Xiaoming Zhao, Xingming Wu, Weihai Chen, Peter CY Chen, Qingsong Xu, and Zhengguo Li. Aliked: A lighter keypoint and descriptor extraction network via deformable transformation. *IEEE Transactions on Instrumentation and Measurement*, 2023. 1