Neural Refinement for Absolute Pose Regression with Feature Synthesis

Shuai Chen[1] Yash Bhalgat[2] Xinghui Li[1] Jia-Wang Bian[1]
Kejie Li[1] Zirui Wang[1] Victor Adrian Prisacariu[1]
[1]Active Vision Lab, University of Oxford
[2]Visual Geometry Group, University of Oxford

Abstract

Absolute Pose Regression (APR) methods use deep neural networks to directly regress camera poses from RGB images. However, the predominant APR architectures only rely on 2D operations during inference, resulting in limited accuracy of pose estimation due to the lack of 3D geometry constraints or priors. In this work, we propose a test-time refinement pipeline that leverages implicit geometric constraints using a robust feature field to enhance the ability of APR methods to use 3D information during inference. We also introduce a novel Neural Feature Synthesizer (NeFeS) model, which encodes 3D geometric features during training and directly renders dense novel view features at test time to refine APR methods. To enhance the robustness of our model, we introduce a feature fusion module and a progressive training strategy. Our proposed method achieves state-of-the-art single-image APR accuracy on indoor and outdoor datasets. Code will be released at https://github.com/ActiveVisionLab/NeFeS.

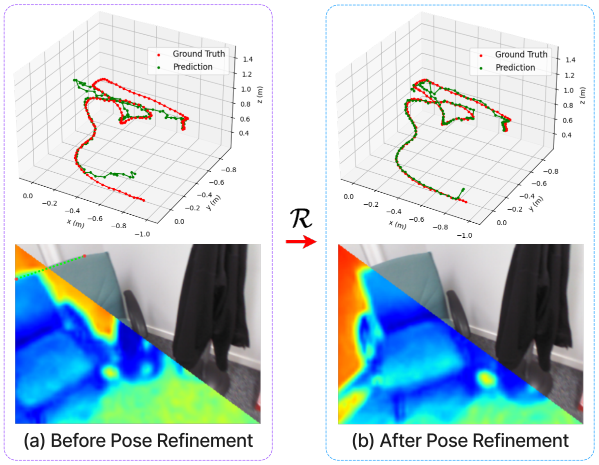


Figure 1. Our pose refinement ($\mathcal{R}$) improves (coarse) pose predictions from other methods using novel feature synthesis to achieve pixel-wise alignment. **Top left / right:** 3D plots of predicted (**green**) and ground-truth (**red**) camera positions. **Bottom left / right:** alignment between rendered features and query image.

1. Introduction

Camera relocalization is a crucial task that allows machines to understand their position and orientation in 3D space. It is an essential prerequisite for applications such as augmented reality, robotics, and autonomous driving, where the accuracy and efficiency of pose estimation are important. Recently, Absolute Pose Regression (APR) methods [21–23] have been shown to be effective in directly estimating camera pose from RGB images using convolutional neural networks. The simplicity of APR's architecture offers several potential advantages over classical geometry-based methods [5, 43, 45], involving end-to-end training, cheap computation cost, and low memory demand.

Latest advances in APR, particularly the use of novel view synthesis (NVS) [10, 11, 29, 32, 33, 49] to generate new images from random viewpoints as data augmentation during training, have significantly improved the pose regression performance. Despite this, state-of-the-art (SOTA) APRs still have the following limitations: (i) They predict the pose of a query image by passing it through a CNN, which typically disregards geometry at inference time. This causes APR networks to struggle to generalize to viewpoints that the training data fails to cover [46]; (ii) The unlabeled data, often sampled from the validation/testing set, used for finetuning the APR network [8, 10, 11] may not be universally available in real-life circumstances, and this semi-supervised finetuning is also time-consuming.

To address these limitations, we propose a novel test-time refinement pipeline for APR methods. Unlike prior works that explore extended Kalman filters [34], pose graph optimization [8], or pose auto-encoders [49], our method integrates an *implicit* representation based geometric refinement into an end-to-end learning framework, where gradients can be backpropagated to the APR network. We test our proposed method across different APR architectures to demonstrate its robustness and effectiveness. Furthermore, we propose a Neural Feature Synthesizer (NeFeS) network to encode the 3D geometry of a scene implicitly into an MLP. NeFeS renders dense features from novel viewpoints for refinement. To ensure the robustness of feature rendering, we introduce a Feature Fusion module into NeFeS that combines the rendered color and features and is trained in a progressive manner. Our method leverages prior literature on volume rendering to inherently constrain geometric consistency during test time using implicit 3D neural feature fields. As such, our approach occupies a middle ground between APR and methods informed by geometry.

We summarize our main contributions as follows: **First**, we propose a test-time refinement pipeline that greatly improves the pose-estimation accuracy of any APR model without using additional unlabeled data and exhibits a new *single-frame* APR SOTA performance on standard benchmarks. **Second**, we propose a Neural Feature Synthesizer (NeFeS) network that implicitly encodes 3D geometric features. NeFeS refines an initial pose by rendering a dense feature map and making the comparison with the query image feature. **Third**, we propose a progressive training strategy and a Feature Fusion module to improve the robustness of the rendering ability of the NeFeS model.

2. Related Work

Absolute Pose Regression (APR). APR methods have been widely studied due to their simple and lightweight formulation that allows the camera pose to be directly regressed using an end-to-end neural network. PoseNet [21–23] introduced the first APR solution using GoogLeNet-backbone, followed by various architectures like the hourglass network [31], attention layers [50, 51, 61], separated translation and rotation prediction [35, 63], or LSTM [60].

To further improve APR accuracy, some works utilize sequential information. These approches incorporate temporal constraints such as visual odometry [8, 39, 58], motion [34], temporal filtering [15], and multitasking [39]. Recent APR methods also benefit from novel view synthesis, where one line of approaches focuses on generating large amounts of extra photo-realistic synthetic data [11, 33, 38] via randomly sampled virtual camera poses. However, generating high-quality offline synthetic data may take several days [33] for each scene. Other approaches [10, 11] use NeRF [29, 32] as a direct matching module to perform unlabeled finetuning [8] using extra images without ground-truth pose annotation. However, finetuning takes significant time and assumes that extra unlabeled data can be easily obtained.

While the aforementioned works enhance APR training, we focus on improving generic APR methods during test time. Unlike prior works that only exam means for test-time refinement on a single specific APR architecture, such as extended Kalman filters [34], pose graph optimization [8], or pose auto-encoders [49], our method exhibits strong flexibility to be adapted to a wide range of APR architectures on both camera positions and orientations, achieving state-of-the-art results without extra unlabeled data.

Notably, classical geometry-based techniques [4–6, 28, 41, 43–45] that require explicit feature correspondence search [16, 17, 26, 42, 54, 55] also employ test-time refinement to improve localization accuracy. For example, [28, 41, 43] build upon image retrievals and pre-computed SfM model to perform standard geometric refinement via neural network-based feature matcher, PnP+RANSAC, or dense featuremetric-alignment. Our method, however, offers end-to-end neural feature refinement via implicit representation, enhancing existing APR models without external storage, pre-computed data, or manual tuning.

Neural Radiance and Feature Fields. Neural Radiance Fields (NeRF) [32] revolutionized novel view image synthesis and 3D surface reconstruction. NeRF's implicit 3D representation and differentiable volume rendering enable self-supervised optimization from RGB images, avoiding costly 3D annotations. iNeRF [64] showed that NeRF can be inverted for pose optimization. Recent approaches such as BARF [27] and its counterparts [3, 14, 62] simultaneously train NeRF by treating camera poses as learnable parameters in simple, non-360°scenes. Parallel works, NICE-SLAM [65] and iMAP [53], use NeRF for dense geometry and real-time camera tracking. Direct-PN [10] uses NeRF as a direct matching module to compute the photometric errors and propagate the error gradients back to the pose regression network. DFNet [11] extends this method to outdoor scenarios with robust feature extraction. LENS [33] uses NeRF to generate a synthetic training dataset based on manually tuned scene bounds and parameters.

Recently, NeRF models have been extended to directly predict and render *feature fields* alongside density and appearance fields. Typically, these feature fields are learned by supervision from a 2D feature extractor using volumetric rendering. [2, 24, 57] showed that these 3D feature fields outperform 2D baselines [9, 12, 25] on downstream tasks such as 2D object retrieval or 3D segmentation. CLIP-Fields [48] established feature fields as scene memory for robot navigation. This work explores distilled neural feature fields for camera relocalization, highlighting their role in test-time pose refinement.

3. Method

In this section, we present a detailed outline of our approach. Sec. 3.1 provides a high-level overview of our refinement framework. Sec. 3.2 describes the architecture and training details of our proposed NeFeS network along with its two components: *Exposure-adaptive Affine Color Transformation (ACT)* and *Feature Fusion module*.

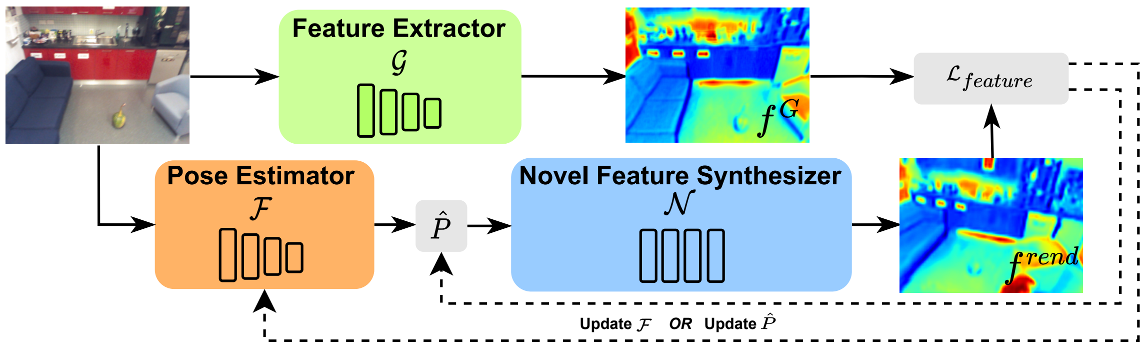


Figure 2. **Illustration of the pose refinement pipeline.** The query image is processed by a pose estimator $\mathcal{F}$, typically an absolute pose regressor, to obtain a coarse camera pose $\hat{P}$. Our novel feature synthesizer $\mathcal{N}$ renders a dense feature map f^{rend} based on $\hat{P}$. Simultaneously, the feature extractor $\mathcal{G}$ extracts the feature map f^G from the query image. We then compute the feature-metric error between f^{rend} and f^G, denoted as $\mathcal{L}_{feature}$. This error is backpropagated to update either the parameters of $\mathcal{F}$ or the pose $\hat{P}$ directly.

3.1. Refinement Framework for APR

Given a query image I, an absolute pose regression (APR) network $\mathcal{F}$ directly regresses the camera pose $\hat{P}$ of I: $\hat{P} = \mathcal{F}(I)$. The network is typically trained with ground truth image-pose pairs. While APR-based methods are much more efficient than geometry-based methods since they require only a single forward pass of the network, the quality of their predictions is often significantly worse than those of geometry-based methods due to the lack of any 3D geometry-based reasoning [46].

In contrast to prior APR research, which attempts to improve APR by adding constraints to the training loss or making architectural changes to the backbone network, we propose an alternative method to refine the results of APR methods by backpropagating a feature-metric error at inference time. Our method has three major components (see Fig. 2): (1) a pretrained APR network, denoted as $\mathcal{F}$, which provides an initial pose; (2) a differentiable novel feature synthesizer $\mathcal{N}$ that directly renders dense feature maps given a camera pose; (3) an off-the-shelf feature extractor $\mathcal{G}$ that extracts the dense feature map of the query image. In our implementation, the feature extraction module from [11] is employed as the feature extractor $\mathcal{G}$.

The refinement procedure is as follows: (i) The query image I is passed through the pretrained APR model $\mathcal{F}$ to predict a coarse camera pose $\hat{P}$. (ii) The feature synthesizer $\mathcal{N}$ renders a dense feature map $f^{rend} \in \mathbb{R}^{n \times c}$ given the coarse camera pose $\hat{P}$, where $n = h \times w$, and h and w are the spatial dimensions of the feature map[1]. (iii) At the same time, the feature extractor $\mathcal{G}$ extracts a feature map $f^G = \mathcal{G}(I)$ from the query image, where $f^G \in \mathbb{R}^{n \times c}$. (iv) The pose $\hat{P}$ is iteratively refined by minimizing the feature cosine similarity loss $\mathcal{L}_{feature}$ [11] between f^{rend} and f^G:

[1]Note: We treat the n dimension as the feature rather than c dimension.

$$\mathcal{L}_{feature} = \sum_{i=1}^{c} \left(1 - \frac{\langle f^{rend}_{:,i} \cdot f^G_{:,i} \rangle}{\|f^{rend}_{:,i}\|_2 \cdot \|f^G_{:,i}\|_2} \right) \quad (1)$$

where $f^{rend}_{:,i}, f^G_{:,i} \in \mathbb{R}^n$, $\langle \cdot, \cdot \rangle$ denotes the inner product between two vectors and $\| \cdot \|_2$ represents the L2 norm. Different from the common feature matching literature's [26, 54] convention, our features are normalized along the spatial direction instead of the channel direction to ensure the consistency of the neighboring pixels.

Our method can be regarded as post-processing to the initial pose $\hat{P}$. We do not save the updated weights of the APR method since we restart from the initial state when given a new query image.

3.2. Neural Feature Synthesizer

We propose a Neural Feature Synthesizer (NeFeS) model that directly renders dense feature maps of a given viewpoint to refine the predictions of an underlying APR network. Similar to NeRF-W [29], our NeFeS architecture uses a base MLP module with *static* and *transient* heads that predict the static and transient density ($\sigma^{(s)}$ and $\sigma^{(\tau)}$) and view-dependent color ($c^{(s)}$ and $c^{(\tau)}$) respectively, given an input 3D position ($\mathbf{x}$) and viewing direction ($\mathbf{d}$). We use the frequency encoding [32, 59] to encode all 3D positions and view directions. The transient head models the colors of the 3D points using an isotropic normal distribution and predicts a view-dependent variance value (β^2) for the transient color distribution. To render the color of a given pixel, the original volume rendering formulation in NeRF [32] is augmented to include the transient colors and densities, and the color of a given image-pixel ($\hat{\mathbf{C}}(\mathbf{r})$) is computed as a composite of the static and transient components. Here, $\mathbf{r}$ denotes the ray (corresponding to the pixel) on which points are sampled to compute the volume rendering quadrature approximation [30]. The variances of sampled points along

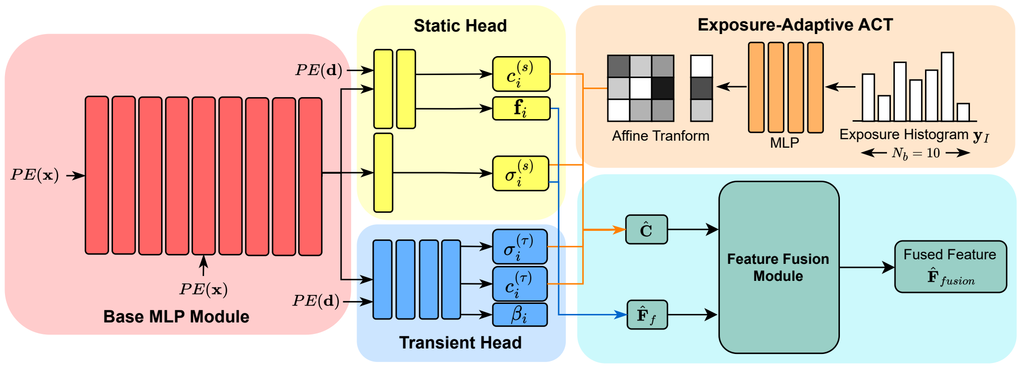


Figure 3. The architecture of our proposed NeFeS model. The query 3D position $\mathbf{x}$ is fed to the network after positional encoding $PE(\cdot)$. The network then splits into two heads: the static head and the transient head. Given a viewing direction $\mathbf{d}$, the rendered color map is generated by fusing static RGB value $c_i^{(s)}$, the transient RGB value c_i^{τ} and their corresponding density values $\sigma_i^{(s)}$ and σ_i^{τ}, while the rendered feature map is formed only by static features $\mathbf{f}_i$ and density $\sigma_i^{(s)}$. In addition, the color map adopts exposure-adaptive ACT to compensate for exposure differences between images. The final feature map $\hat{\mathbf{F}}_{fusion}$ is the concatenation of rendered RGB and feature map processed by the feature fusion module.

the corresponding ray are also rendered using only the transient densities (and *not* the static densities) to obtain a per-pixel color variance $\beta(\mathbf{r})^2$. We refer reader to [29] for more details on the static+transient volume rendering.

We expand the output of the static MLP to also predict features for an input 3D position. The output dimension is $N_c + N_f$, where N_f features are predicted along with RGB values. The per-pixel features are rendered using the same volume rendering quadrature approximation [30]:

$$\hat{\mathbf{F}}_f(\mathbf{r}) = \sum_{i=1}^{N} T_i \left(1 - \exp\left(-\sigma_i^{(s)}\delta_i\right)\right)\mathbf{f}_i, \quad T_i = \exp\left(-\sum_{j=1}^{i-1}\sigma_j^{(s)}\delta_j\right) \tag{2}$$

where $\mathbf{f}_i$ and $\sigma_i^{(s)}$ are the feature and density predicted by the static MLP for a sampled point on the ray, and δ_i is the distance between sampled quadrature points i and $i+1$.

Fig. 3 demonstrates the architecture of our proposed NeFeS model. We propose two crucial components in the rendering pipeline of our NeFeS architecture that ensure the robustness of our rendered features.

Exposure-adaptive ACT. In the context of camera relocalization, testing images may differ in exposure or lighting from training sequences. To address this, DFNet [11] proposed using the luminance histogram of the query image as a latent code input to the color prediction head of the NeRF MLP. However, since our NeFeS outputs both colors and features simultaneously, we find this approach perturbs the feature output values and causes instability. Ideally, the feature descriptors should be able to maintain local invariance even under varying exposure. Inspired by Urban Radiance Fields (URF) [40], we propose to use an *exposure-adaptive Affine Color Transformation (ACT)* which is a 3×3 matrix $\mathbf{K}$ and a 3-dimensional bias vector $\mathbf{b}$ predicted by a 4-layer MLP with the query image's luminance histogram $\mathbf{y}_I$. Unlike URF, which uses a pre-determined exposure code, we use the query image's histogram embedding for accurate appearance rendering of unseen testing images. The final per-pixel color $\hat{\mathbf{C}}(\mathbf{r})$ is computed using the affine transformation as $\hat{\mathbf{C}}(\mathbf{r}) = \mathbf{K}\hat{\mathbf{C}}_{rend}(\mathbf{r}) + \mathbf{b}$, where $\hat{\mathbf{C}}_{rend}(\mathbf{r})$ is the rendered per-pixel color obtained using the static and transient MLPs.

Feature Fusion Module We propose a Feature Fusion module to fuse the rendered colors and features to produce the final feature map. The rendered colors and features are concatenated and fed into the fusion module consisting of three 3x3 convolutions, followed by a 5x5 convolution and a batch normalization layer. During inference, we render colors and features for all $H \times W$ image pixels and the resulting $H{\times}W{\times}(N_c{+}N_f)$ tensor is processed by the module. Note, for efficiency during training, we sample $S{\times}S$ regions to render and apply the loss to those pixels each iteration.

We use $\mathcal{H}$ to represent the fusion module. The final output feature result is:

$$\hat{\mathbf{F}}_{fusion}(\mathcal{R}) = \mathcal{H}(\hat{\mathbf{C}}(\mathcal{R}), \hat{\mathbf{F}}_f(\mathcal{R})) \tag{3}$$

where $\mathcal{R}$ is the sampled region as described above.

We experimentally find that the fusion module produces more robust features than the input rendered features $\hat{\mathbf{F}}_f$. We refer readers to the supplementary for detailed ablations.

Training the Feature Synthesizer The high-level concept of training the NeFeS is motivated by feature field distillation proposed in [24], which essentially distills the 2D backbone features into a 3D NeRF model. However, 2D features in our NeFeS need to be closely related to the direct matching formulation [10, 20]. In this work, we use the trained 2D feature extractor from [11] to produce the feature labels due to its effectiveness in generating domain invariant features.

Loss Functions. The total loss used to train our NeFeS model consists of a photometric loss $\mathcal{L}_{rgb}$ and two l_1-based feature-metric losses:

$$\mathcal{L} = \mathcal{L}_{rgb} + \lambda_1 \mathcal{L}_f + \lambda_2 \mathcal{L}_{fusion}. \tag{4}$$

The photometric loss is defined as the negative log-likelihood of a normal distribution with variance $\beta(\mathbf{r})^2$:

$$\begin{aligned}\mathcal{L}_{rgb}(\mathbf{r}) =& \frac{1}{2\beta_i(\mathbf{r})^2} \left\|\mathbf{C}(\mathbf{r}) - \hat{\mathbf{C}}(\mathbf{r})\right\|_2^2 \\ &+ \frac{1}{2}\log\beta(\mathbf{r})^2 + \frac{\lambda_s}{K}\sum_{k=1}^{K}\sigma_k^{(\tau)}\end{aligned} \tag{5}$$

where $\mathbf{r}$ is the ray direction corresponding to an image pixel, $\mathbf{C}_i(\mathbf{r})$ and $\hat{\mathbf{C}}_i(\mathbf{r})$ are the ground-truth and rendered pixel colors. The third term in Eq. (5) is a sum of the transient densities of all the points on ray $\mathbf{r}$ and is used to ensure that transient densities are sparse.

The feature losses are simply l_1 losses:

$$\mathcal{L}_f = \sum_{\mathbf{r}\in\mathcal{R}} \|\hat{\mathbf{F}}_f(\mathbf{r}) - \mathbf{F}_{\mathbf{img}}(I,\mathbf{r})\|_1, \tag{6}$$

and

$$\mathcal{L}_{fusion} = \sum_{\mathbf{r}\in\mathcal{R}} \|\hat{\mathbf{F}}_{fusion}(\mathbf{r}) - \mathbf{F}_{\mathbf{img}}(I,\mathbf{r})\|_1. \tag{7}$$

where $\mathbf{F}_{\mathbf{img}}(I,\cdot)$ are the features extracted from the training images using the pre-trained 2D feature extractor [11]. Note that, $\mathcal{L}_f$ is applied to the rendered features $\hat{\mathbf{F}}_f$ and $\mathcal{L}_{fusion}$ is applied to the fused features $\hat{\mathbf{F}}_{fusion}$. We experimentally find that using l_1 gives more robust features than l_2 and cosine feature loss for the test time refinement.

Progressive Training. We propose using a progressive schedule to train the NeFeS model. We first train the color and density part of the network for T_1 epochs to bootstrap the correct 3D geometry for the network. For these epochs, only $\mathcal{L}_{rgb}$ is used. Then we add $\mathcal{L}_f$ with weight λ_1 for the next T_2 epochs to train the feature part of the static MLP. Since the ground-truth features may not be fully multi-view consistent, we apply *stop-gradients* to the predicted density for the feature rendering branch. And finally, we add the feature fusion loss $\mathcal{L}_{fusion}$ with weight λ_2 for the last T_3 epochs. Since the feature fusion module takes both RGB images and 2D features as input, we randomly sample N_{crop} patches of $S \times S$ regions of the image and features to increase training efficiency. According to our experiments, this progressive training schedule leads to better convergence and performance. In addition, we apply semantic filtering to improve the network training results. Specifically, we use an off-the-shelf panoptic segmentation method [13] to mask out temporal objects in the scene such as people and moving vehicles.

3.3. Direct Pose Refinement

While our method is primarily designed to optimize APR, it is also possible to directly optimize camera pose parameters. We explore this feature by showing a possible scenario wherein the source of the pose estimation is either a black box or cannot be optimized (e.g. the initial camera pose comes from image retrieval). In these settings, we can set up our proposed method to directly refine the camera poses. Specifically, given an estimated camera pose $\hat{P} = [\mathbf{R}|\mathbf{t}]$, where $\mathbf{R}$ is rotation and $\mathbf{t}$ is the translation component, our method optimizes the camera poses using tangent space backpropagation[2]. Additionally, we found that using two different learning rates for the translation and rotation parts helps achieve faster and more stable convergence for camera pose refinement. This is different from the standard convention used in [3, 14, 27, 62] . We refer our readers to supplementary material for more details.

4. Experiments

We implement our method in PyTorch [37]. Implementation details about the NeFeS architecture and progressive training scheduling can be found in the supplementary[3].

4.1. Evaluation on Cambridge Landmarks

We evaluate our proposed refinement method on Cambridge Landmarks [23] , which is a popular outdoor dataset used for benchmarking pose regression methods. The dataset contains handheld smartphone images of scenes with large exposure variations and covers an area of $875m^2$ to $5600m^2$. The training sequences contain 200-1500 samples, and test sets are captured from different sequences. For each test image, we refine the model using the approach in Sec. 3.1 for $m = 50$ iterations.

We first test our method on top of an open-sourced SOTA single-frame APR method. Tab. 1 summarizes the results of our method and existing APR methods. Our method achieves the best accuracy across all four scenes when coupled with the DFNet. In Sec. 4.3, we demonstrate the performance of our method with other APR approaches. Particularly, our method improves DFNet by as much as 70.6% compared to its scene average results. All the per-scene performances from the other compared methods are taken from their papers, except for MS-Transformer+PAE, which only reports the scene average median errors. We encourage our readers to check out supplementary for more thorough comparisons and ablations to the Cambridge Landmarks dataset.

[2]We use the LieTorch [56] library for this.
[3]Supplementary: Implementation Details

Methods	Kings	Hospital	Shop	Church	Average
PoseNet(PN)[23]	1.66/4.86	2.62/4.90	1.41/7.18	2.45/7.96	2.04/6.23
PN Learn σ^2[22]	0.99/1.06	2.17/2.94	1.05/3.97	1.49/3.43	1.43/2.85
geo. PN[22]	0.88/1.04	3.20/3.29	0.88/3.78	1.57/3.32	1.63/2.86
LSTM PN[60]	0.99/3.65	1.51/4.29	1.18/7.44	1.52/6.68	1.30/5.51
MapNet[8]	1.07/1.89	1.94/3.91	1.49/4.22	2.00/4.53	1.63/3.64
TransPoseNet[50]	0.60/2.43	1.45/3.08	0.55/3.49	1.09/4.94	0.91/3.50
MS-Transformer[51]	0.83/1.47	1.81/2.39	0.86/3.07	1.62/3.99	1.28/2.73
MS-Transformer+PAE[49]	-	-	-	-	0.96/2.73
DFNet [11]	0.73/2.37	2.00/2.98	0.67/2.21	1.37/4.03	1.19/2.90
DFNet + **NeFeS$_{50}$(ours)**	**0.37/0.54**	**0.52/0.88**	**0.15/0.53**	**0.37/1.14**	**0.35/0.77**

Table 1. **Comparisons on Cambridge Landmarks.** We compare our proposed test-time refinement method with single-frame APR methods. The subscript of **NeFeS$_{50}$** denotes the number of optimization iterations used for APR refinement. We report the median position and orientation errors in $m/°$. The best results are highlighted in **bold**.

Methods	Chess	Fire	Heads	Office	Pumpkin	Kitchen	Stairs	Average
PoseNet	0.10/4.02	0.27/10.0	0.18/13.0	0.17/5.97	0.19/4.67	0.22/5.91	0.35/10.5	0.21/7.74
MapNet	0.13/4.97	0.33/9.97	0.19/16.7	0.25/9.08	0.28/7.83	0.32/9.62	0.43/11.8	0.28/10.0
MS-Transformer	0.11/6.38	0.23/11.5	0.13/13.0	0.18/8.14	0.17/8.42	0.16/8.92	0.29/10.3	0.18/9.51
PAE	0.13/6.61	0.24/12.0	0.14/13.0	0.19/8.58	0.17/7.28	0.18/8.89	0.30/10.3	0.19/9.52
DFNet	0.03/1.12	0.06/2.30	0.04/2.29	0.06/1.54	0.07/1.92	0.07/1.74	0.12/2.63	0.06/1.93
DFNet + **NeFeS$_{50}$(ours)**	**0.02/0.57**	**0.02/0.74**	**0.02/1.28**	**0.02/0.56**	**0.02/0.55**	**0.02/0.57**	**0.05/1.28**	**0.02/0.79**

Table 2. **Comparisons on 7-Scenes dataset.** We compare the proposed refinement method with previous single-frame APR methods. We evaluate all methods with SfM ground truth poses provided in [7], measured in median translational error (m) and rotational error (°). Numbers in **bold** represent the best performance.

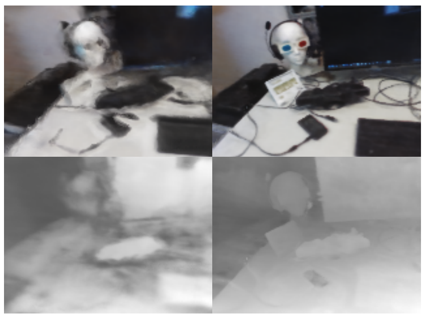


Figure 4. Qualitative comparison between the NeRFs trained by dSLAM GT pose (a) vs. SfM GT pose (b). As illustrated, SfM NeRF (PSNR 19.94 dB) can render superior geometric details (bottom row) than dSLAM NeRF (PSNR 16.11 dB).

4.2. Evaluation on 7-Scenes

We further evaluate our method on Microsoft 7-Scenes dataset [19, 52], which includes seven indoor scenes ranging in size from $1m^3$ to $18m^3$ and up to 7000 training images for each scene. We sub-sampled the large training sequences in the dataset to 1000 images for training our NeFeS model. The original 'ground truth' (GT) poses are obtained from RGB-D SLAM (dSLAM) [36]. However, we observe imperfection in the GT poses due to the asynchronous data between the RGB and depth sequences, and this results in low-quality NeRF renderings, as shown in Fig. 4. Thus, we use alternative 'ground truth' provided by [7] for our experiments. The authors [7] demonstrate that their camera poses reconstructed by COLMAP [47], a structure-from-Motion (SfM) library, are more accurate for image-based relocalization. We refer the reader to our supplementary [4] for more details about the GT poses.

For a fair comparison, we use the SfM poses to re-train baseline APR methods using their official code, except for PoseNet, in which we use the open-sourced code from [10]. We trained each APR method 3-4 times to select the best-performing model. We use DFNet + NeFeS$_{50}$ with the same settings as in Cambridge for our pose refinement experiment. We achieve state-of-the-art results (59%+ better in scene average) in all scenes by running 50 optimization steps (see Tab. 2). The results by using the dSLAM ground-truth poses are included in the supplementary[5], where we also achieve SOTA accuracy.

It is imperative to note that our method is not constrained by the number of optimization steps for the refinement process. In our experience, nearly 50% of the entire pose error improvement is accomplished within the initial 10 steps. It

[4]Supplementary: 7-Scenes Dataset Details
[5]Supplementary: APR Comparisons with dSLAM GT

Dataset	7-Scenes	Cambridge
PoseNet	0.21m/7.74	2.04m/6.23
+ Ours	**0.08m/2.83°**	**0.54m/1.05°**
MS-Trans.	0.18m/9.51°	1.28m/2.73°
+ Ours	**0.11m/3.46°**	**0.43m/1.04°**
DFNet	0.06m/1.93°	1.19m/2.90°
+ Ours	**0.02m/0.79°**	**0.35m/0.77°**

Table 3. **Pose refinement on different APR architectures.** Our refinement method can effectively improve pose estimation results for different APR architectures.

Dataset	NetVlad	Optimize Pose
7-Scenes	0.32m/13.72°	**0.14m/3.97°**
Cambridge	3.18m/7.74°	**1.15m/1.30°**

Table 4. **Pose refinement on NetVlad** Our method also works on poses initialized with non-APR-based methods, such as NetVlad image retrieval. Since the initial pose error is relatively large, we refine the poses with 100 iterations.

is up to the user to find the optimal trade-off that fits their computational budget. A detailed analysis of this facet is provided in Sec. 4.7.

4.3. Refinement for Different APRs

Tab. 3 shows the results of our method with different APR architectures. Our proposed method exhibits versatility, operating beneficially under various APR architectures, such as PoseNet (classic pose regression architecture), MS-Transformer (EfficentNet CNN backbones with transformer blocks), and DFNet (multi-task network that predicts domain invariant features and poses). A full table with per-scene results is provided in Supplementary.

4.4. Optimize APR vs. Optimize Pose

Besides boosting APR, our proposed approach can also refine the initial coarse camera pose, as outlined in Sec. 3.3. We first show a use case of this scenario by coupling our method with image retrieval, where the initial pose can only be optimized due to the non-differentiable nature of the retrieval process. Given a query image, we retrieve its nearest neighbor from the training data using NetVLAD [1] and use the associated pose as the initial pose. We set the learning rate to be lr_R and lr_t for rotation and translation components, respectively. Specifically, for indoor scenes, we set $lr_R = 0.0087$ (corresponds to $0.5°$ in radiance) and $lr_t = 0.01$. For outdoor scenes, we set $lr_R = 0.01$ and $lr_t = 0.1$. Tab. 4 summarises the experimental results, indicating substantial improvements in pose accuracy over the NetVLAD retrieved coarse pose, exceeding the performance of many prior APR approaches.

Dataset	DFNet	Optimize Pose	Optimize APR
7-Scenes	0.06m/1.93°	0.04m/1.16°	**0.02m/0.79°**
Cambridge	1.19m/2.90°	0.66m/1.39°	**0.35m/0.77°**

Table 5. **Pose refinement vs. APR refinement** We study on the optimization over APR vs. the pose. Both methods can effectively optimize poses. However, optimizing APR can obtain lower errors than optimizing poses given the same number of iterations.

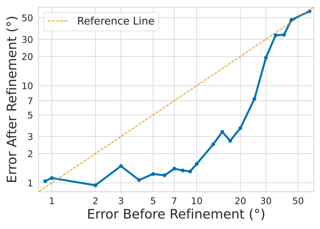
We further conducted controlled experiments to investigate if performance disparities exist between two types of optimization: optimizing the APR's parameters or directly optimizing the pose itself. We evaluated both modes on the DFNet with NeFeS$_{50}$ refinement, as illustrated in Tab. 5. The result suggests that while both refinement approaches can effectively improve the pose accuracy, optimizing the neural network's parameters obtains a better result than directly optimizing the pose itself, which is an interesting insight. Nevertheless, optimizing the pose remains is also valuable, particularly when the initial pose is derived from a non-differentiable or a black-box pose estimator.

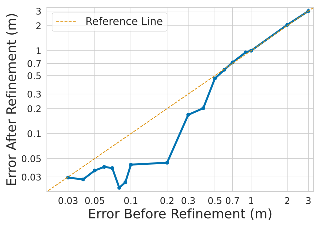
4.5. Pose Refinement Bounds

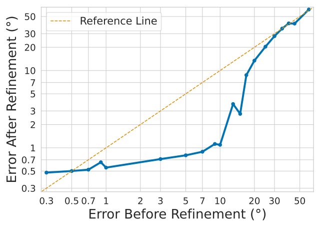
Our refinement method relies on matching rendered features with query image features during test time, so it may fail when there is not sufficient overlap between the two feature maps. To determine the bounds of our refinement method, we randomly perturb the ground-truth pose and determine the maximum perturbation at which our method stops converging. We jitter the orientation or position of the ground truth pose components separately while gradually increasing the magnitude of the perturbation. We use two scenes, an indoor scene (*7-Scenes: Heads*) and an outdoor scene (*Cambridge: Shop Facade*), to illustrate our results in Fig. 5. We observe that our method cannot refine pose errors larger than $35°$. In case of translational errors, our method can refine errors up to $0.6m$ on *Heads* (indoor scene) and up to $4m$ in *Shop Facade* (outdoor scene). This difference may come from the discrepancy in scale between indoor and outdoor settings. For example, in the small-scale scene of *Heads*, the camera is closer to the objects, hence even small movements lead to a large change in the rendering.

4.6. NeFeS Ablation

This section presents the ablation study of our NeFeS network. In Tab. 6a, we gradually remove the exposure-adaptive ACT and the Feature Fusion module and evaluate their impact on the performance of our approach on *Cambridge Shop Facade*. The results demonstrate that the removal of each component leads to a deterioration in pose accuracy, indicating the effectiveness of both components.


(a) Rotation test (Indoor)


(b) Translation test (Indoor)


(c) Rotation test (Outdoor)

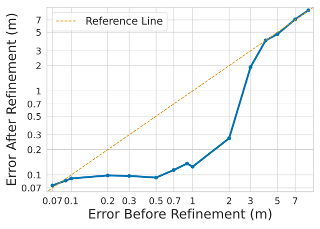

(d) Translation test (Outdoor)

Figure 5. Experiments on pose refinement bounds of our method in indoor and outdoor scenes. Each plot shows errors before (x-axis) and after (y-axis) refinement when ground-truth pose is perturbed by varying magnitudes. Dashed green line is '$y=x$'. Points below this line indicate a reduction in pose error using our refinement method.

(a) NeFeS Architecture Ablation

Method	Shop Facade
Initial Pose Error	0.67m/2.21°
Refine w/ NeFeS (ours)	**0.15m/0.53°**
- Exposure-adaptive ACT Ⓐ	0.15m/1.20°
- Ⓐ+Feature Fusion	0.37m/1.62°

(b) Training Scheduling Ablation

Method	Shop Facade
Combined	0.17m/0.80°
Progressive	**0.15m/0.53°**

Table 6. **(a)** Ablation on NeFeS architecture. **(b)** Ablation on the proposed training scheduling

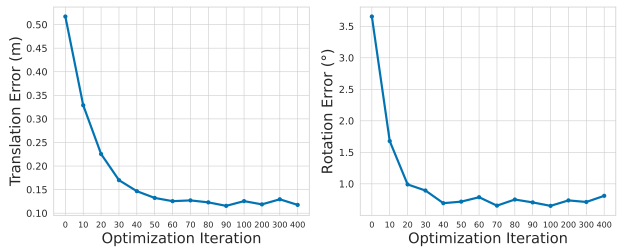


Figure 6. Plots of rotation and translation errors against the number of iteration on *Cambridge: Shop Facade* scene.

A noteworthy insight from our architecture design is that the superior pose estimation accuracy is attributed to integrating both Exposure-adaptive ACT and the Feature Fusion module. The feature fusion module can smooth potential noises (outliers) from directly-rendered 3D features.

In Tab. 6b, we compare our progressive training scheduling with the combined scheduling, where all three loss terms have been enabled simultaneously since the beginning of the training. The results reveal that the progressive training scheduling results in better accuracy, providing further support for our design decisions.

4.7. Number of Iterations vs. Accuracy Trade-off

In Fig. 6, we plot the relationship between the number of optimization iterations and pose error. Both translation and rotation errors reduce significantly within the first 10-20 iterations. Hence, given a computational budget, an operating point can be (optionally) chosen w.r.t. the performance-time trade-off. The errors start to plateau around 50 steps. Although we can achieve even lower errors with more iterations, we think using 50 steps strikes a balance between accuracy and efficiency, and explains how we set our previous experiments.

4.8. Spatial vs Channel-wise Normalization

As described in Sec. 3.1, we empirically find spatial-wise normalized features yield higher accuracy than channel-wise normalized features when computing the feature cosine similarity loss $\mathcal{L}_{feature}$, evidenced by Tab. 7. This is likely due to our spatially sensitive dense direct matching, akin to methods like Direct Sparse Odometry [18]. In contrast, channel-wise normalization can introduce inconsistencies among neighboring pixels.

Methods	Shop
Channel	0.20m/1.05°
Spatial	**0.15m/0.53°**

Table 7. Performance comparison between using Channel-wise vs. Spatial-wise normalization in loss $\mathcal{L}_{feature}$ during refinement.

5. Conclusion

We tackle the camera relocalization problem and improve absolute pose regression (APR) methods by proposing a test-time refinement method. In particular, we design a novel model named Neural Feature Synthesizer (NFS), which can encode 3D geometric features. Given an estimated pose, the NFS renders a dense feature map, compares it with the query image features, and back-propagates the error to refine estimated camera poses from APR methods. In addition, we propose a progressive learning strategy and a feature fusion module to improve the feature robustness of the NFS model. The experiments demonstrate that our method can greatly improve the accuracy of APR methods. Our method provides a promising direction for improving the accuracy of APR methods.

References

[1] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, 2016. 7

[2] Yash Bhalgat, Iro Laina, João F Henriques, Andrew Zisserman, and Andrea Vedaldi. Contrastive lift: 3d object instance segmentation by slow-fast contrastive fusion. In *NeurIPS*, 2023. 2

[3] Wenjing Bian, Zirui Wang, Kejie Li, Jiawang Bian, and Victor Adrian Prisacariu. NoPe-NeRF: Optimising neural radiance field with no pose prior. In *CVPR*, 2023. 2, 5

[4] Eric Brachmann and Carsten Rother. Learning Less is More - 6D Camera Localization via 3D Surface Regression. In *CVPR*, 2018. 2

[5] Eric Brachmann and Carsten Rother. Visual camera re-localization from RGB and RGB-D images using DSAC. *IEEE TPAMI*, 2021. 1

[6] Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC - Differentiable RANSAC for Camera Localization. In *CVPR*, 2017. 2

[7] Eric Brachmann, Martin Humenberger, Carsten Rother, and Torsten Sattler. On the limits of pseudo ground truth in visual camera re-localisation. In *ICCV*, 2021. 6

[8] S. Brahmbhatt, J. Gu, K. Kim, J. Hays, and J. Kautz. Geometry-Aware Learning of Maps for Camera Localization. In *CVPR*, 2018. 1, 2, 6

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 2

[10] Shuai Chen, Zirui Wang, and Victor Prisacariu. Direct-PoseNet: Absolute pose regression with photometric consistency. In *3DV*, 2021. 1, 2, 4, 6

[11] Shuai Chen, Xinghui Li, Zirui Wang, and Victor Prisacariu. DFNet: Enhance absolute pose regression with direct feature matching. In *ECCV*, 2022. 1, 2, 3, 4, 5, 6

[12] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 2

[13] Bowen Cheng, Ishan Misra, Alexander G Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, 2022. 5

[14] Shin-Fang Chng, Sameera Ramasinghe, Jamie Sherrah, and Simon Lucey. Garf: Gaussian activated radiance fields for high fidelity reconstruction and pose estimation. In *ECCV*, 2022. 2, 5

[15] Ronald Clark, Sen Wang, Andrew Markham, Niki Trigoni, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *CVPR*, 2017. 2

[16] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *CVPRW*, 2018. 2

[17] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-net: A trainable cnn for joint detection and description of local features. In *CVPR*, 2019. 2

[18] J. Engel, V. Koltun, and D. Cremers. DSO: Direct sparse odometry. In *IEEE TPAMI*, 2017. 8

[19] Ben Glocker, Shahram Izadi, Jamie Shotton, and Antonio Criminisi. Real-time rgb-d camera relocalization. In *ISMAR*, 2013. 6

[20] M. Irani and P. Anandan. All about direct methods. In *Workshop Vis. Algorithms: Theory Pract.*, 1999. 4

[21] A. Kendall and R. Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *ICRA*, 2016. 1, 2

[22] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *CVPR*, 2017. 6

[23] A. Kendall, M. Grimes, and R. Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, 2015. 1, 2, 5, 6

[24] Sosuke Kobayashi, Eiichi Matsumoto, and Vincent Sitzmann. Decomposing nerf for editing via feature field distillation. In *NeurIPS*, 2022. 2, 4

[25] Boyi Li, Kilian Q Weinberger, Serge Belongie, Vladlen Koltun, and Rene Ranftl. Language-driven semantic segmentation. In *ICLR*, 2022. 2

[26] Xinghui Li, Kai Han, Shuda Li, and Victor Prisacariu. Dual-resolution correspondence networks. In *NeurIPS*, 2020. 2, 3

[27] Chen-Hsuan Lin, Wei-Chiu Ma, Antonio Torralba, and Simon Lucey. Barf: Bundle-adjusting neural radiance fields. In *ICCV*, 2021. 2, 5

[28] Philipp Lindenberger, Paul-Edouard Sarlin, Viktor Larsson, and Marc Pollefeys. Pixel-Perfect Structure-from-Motion with Featuremetric Refinement. *ICCV*, 2021. 2

[29] Ricardo Martin-Brualla, Noha Radwan, Mehdi S. M. Sajjadi, Jonathan T. Barron, Alexey Dosovitskiy, and Daniel Duckworth. NeRF in the Wild: Neural Radiance Fields for Unconstrained Photo Collections. In *CVPR*, 2021. 1, 2, 3, 4

[30] N. Max. Optical models for direct volume rendering. In *IEEE Transactions on Visualization and Computer Graphics*, 1995. 3, 4

[31] I. Melekhov, J. Ylioinas, J. Kannala, and E. Rahtu. Image-based localization using hourglass networks. In *ICCVW*, 2017. 2

[32] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3

[33] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. LENS: Localization enhanced by nerf synthesis. In *CoRL*, 2021. 1, 2

[34] Arthur Moreau, Nathan Piasco, Dzmitry Tsishkou, Bogdan Stanciulescu, and Arnaud de La Fortelle. Coordinet: uncertainty-aware pose regressor for reliable vehicle localization. In *WACV*, 2022. 1, 2

[35] T. Naseer and W. Burgard. Deep regression for monocular camera-based 6-dof global localization in outdoor environments. In *IROS*, 2017. 2

[36] R. Newcombe, A. Davison, S. Izadi, P. Kohli, O. Hilliges, J. Shotton, D. Molyneaux, S. Hodges, D. Kim, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *ISMAR*, 2011. 6

[37] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 5

[38] Pulak Purkait, Cheng Zhao, and Christopher Zach. Synthetic view generation for absolute pose regression and image synthesis. In *BMVC*, 2018. 2

[39] N. Radwan, A. Valada, and W. Burgard. Vlocnet++: Deep multitask learning for semantic visual localization and odometry. In *IEEE Robotics and Automation Letters*, 2018. 2

[40] Konstantinos Rematas, Andrew Liu, Pratul P Srinivasan, Jonathan T Barron, Andrea Tagliasacchi, Thomas Funkhouser, and Vittorio Ferrari. Urban radiance fields. In *CVPR*, 2022. 4

[41] Paul-Edouard Sarlin, Cesar Cadena, Roland Siegwart, and Marcin Dymczyk. From coarse to fine: Robust hierarchical localization at large scale. In *CVPR*, 2019. 2

[42] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 2

[43] Paul-Edouard Sarlin, Ajaykumar Unagar, Måns Larsson, Hugo Germain, Carl Toft, Viktor Larsson, Marc Pollefeys, Vincent Lepetit, Lars Hammarstrand, Fredrik Kahl, and Torsten Sattler. Back to the Feature: Learning robust camera localization from pixels to pose. In *CVPR*, 2021. 1, 2

[44] T. Sattler, B. Leibe, and L. Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012.

[45] T. Sattler, B. Leibe, and L. Kobbelt. Efficient & Effective Prioritized Matching for Large-Scale Image-Based Localization. In *IEEE TPAMI*, 2017. 1, 2

[46] T. Sattler, Q. Zhou, M. Pollefeys, and L. Leal-Taixe. Understanding the limitations of cnn-based absolute camera pose regression. In *CVPR*, 2019. 1, 3

[47] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 6

[48] Nur Muhammad Mahi Shafiullah, Chris Paxton, Lerrel Pinto, Soumith Chintala, and Arthur Szlam. Clip-fields: Weakly supervised semantic fields for robotic memory. In *Workshop on Language and Robotics at CoRL*, 2022. 2

[49] Yoli Shavit and Yosi Keller. Camera pose auto-encoders for improving pose regression. In *ECCV*, 2022. 1, 2, 6

[50] Yoli Shavit, Ron Ferens, and Yosi Keller. Paying attention to activation maps in camera pose regression. In *arXiv preprint arXiv:2103.11477*, 2021. 2, 6

[51] Yoli Shavit, Ron Ferens, and Yosi Keller. Learning multi-scene absolute pose regression with transformers. In *ICCV*, 2021. 2, 6

[52] Jamie Shotton, Ben Glocker, Christopher Zach, Shahram Izadi, Antonio Criminisi, and Andrew Fitzgibbon. Scene coordinate regression forests for camera relocalization in rgb-d images. In *CVPR*, 2013. 6

[53] Edgar Sucar, Shikun Liu, Joseph Ortiz, and Andrew Davison. iMAP: Implicit mapping and positioning in real-time. In *ICCV*, 2021. 2

[54] Jiaming Sun, Zehong Shen, Yuang Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-free local feature matching with transformers. *CVPR*, 2021. 2, 3

[55] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, , and Akihiko Torii. InLoc: Indoor Visual Localization with Dense Matching and View Synthesis. *CVPR*, 2018. 2

[56] Zachary Teed and Jia Deng. Tangent space backpropagation for 3d transformation groups. In *CVPR*, 2021. 5

[57] Vadim Tschernezki, Iro Laina, Diane Larlus, and Andrea Vedaldi. Neural Feature Fusion Fields: 3D distillation of self-supervised 2D image representations. In *3DV*, 2022. 2

[58] A. Valada, N. Radwan, and W. Burgard. Deep auxiliary learning for visual localization and odometry. In *ICRA*, 2018. 2

[59] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 3

[60] F. Walch, C. Hazirbas, L. Leal-Taixe, T. Sattler, S. Hilsenbeck, and D. Cremers. Image-based localization using lstms for structured feature correlation. In *ICCV*, 2017. 2, 6

[61] Bing Wang, Changhao Chen, Chris Xiaoxuan Lu, Peijun Zhao, Niki Trigoni, and Andrew Markham. Atloc: Attention guided camera localization. In *AAAI*, 2020. 2

[62] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021. 2, 5

[63] J. Wu, L. Ma, and X. Hu. Delving Deeper into Convolutional Neural Networks for Camera Relocalization. In *ICRA*, 2017. 2

[64] Lin Yen-Chen, Pete Florence, Jonathan T. Barron, Alberto Rodriguez, Phillip Isola, and Tsung-Yi Lin. inerf: Inverting neural radiance fields for pose estimation. In *arxiv arXiv:2012.05877*, 2020. 2

[65] Zihan Zhu, Songyou Peng, Viktor Larsson, Weiwei Xu, Hujun Bao, Zhaopeng Cui, Martin R. Oswald, and Marc Pollefeys. Nice-slam: Neural implicit scalable encoding for slam. In *CVPR*, 2022. 2