HAL
open science

Neural Reprojection Error: Merging Feature Learning and Camera Pose Estimation

Hugo Germain, Vincent Lepetit, Guillaume Bourmaud

To cite this version:

Hugo Germain, Vincent Lepetit, Guillaume Bourmaud. Neural Reprojection Error: Merging Feature Learning and Camera Pose Estimation. Conference on Computer Vision and Pattern Recognition, Jun 2021, Online, United States. hal-03482286

HAL Id: hal-03482286

https://hal.science/hal-03482286v1

Submitted on 15 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

arXiv:2103.07153v1 [cs.CV] 12 Mar 2021

Neural Reprojection Error: Merging Feature Learning and Camera Pose Estimation

Hugo Germain[1] Vincent Lepetit[1] Guillaume Bourmaud[2]
[1]LIGM, École des Ponts, Univ Gustave Eiffel, CNRS, Marne-la-vallée, France
[2]IMS, University of Bordeaux, Bordeaux INP, CNRS, France
{hugo.germain, vincent.lepetit}@enpc.fr guillaume.bourmaud@u-bordeaux.fr

Abstract

Absolute camera pose estimation is usually addressed by sequentially solving two distinct subproblems: First a feature matching problem that seeks to establish putative 2D-3D correspondences, and then a Perspective-n-Point problem that minimizes, w.r.t. the camera pose, the sum of so-called Reprojection Errors (RE). We argue that generating putative 2D-3D correspondences 1) leads to an important loss of information that needs to be compensated as far as possible, within RE, through the choice of a robust loss and the tuning of its hyperparameters and 2) may lead to an RE that conveys erroneous data to the pose estimator. In this paper, we introduce the Neural Reprojection Error (NRE) as a substitute for RE. NRE allows to rethink the camera pose estimation problem by merging it with the feature learning problem, hence leveraging richer information than 2D-3D correspondences and eliminating the need for choosing a robust loss and its hyperparameters. Thus NRE can be used as training loss to learn image descriptors tailored for pose estimation. We also propose a coarse-to-fine optimization method able to very efficiently minimize a sum of NRE terms w.r.t. the camera pose. We experimentally demonstrate that NRE is a good substitute for RE as it significantly improves both the robustness and the accuracy of the camera pose estimate while being computationally and memory highly efficient. From a broader point of view, we believe this new way of merging deep learning and 3D geometry may be useful in other computer vision applications. Source code and model weights will be made available at hugogermain.com/nre.

1. Introduction

Absolute camera pose estimation is a fundamental step to many computer vision applications, such as Structure-from-Motion (SfM) [20, 38, 39, 44] and visual localization [36, 42, 43]. Given a pre-acquired 3D model of the world, we aim at estimating the most accurate camera pose of an unseen query image. In practice, as illustrated on the left hand-side of Figure 2, this problem is often addressed by sequentially solving two distinct subproblems: First, a feature matching problem that seeks to establish putative 2D-3D correspondences between the 3D point cloud and the image to be localized, and then a Perspective-n-Point (P*n*P) problem that uses these correspondences as inputs to minimize a sum of so-called reprojection errors w.r.t. the camera pose. The Reprojection Error (RE) is a function of

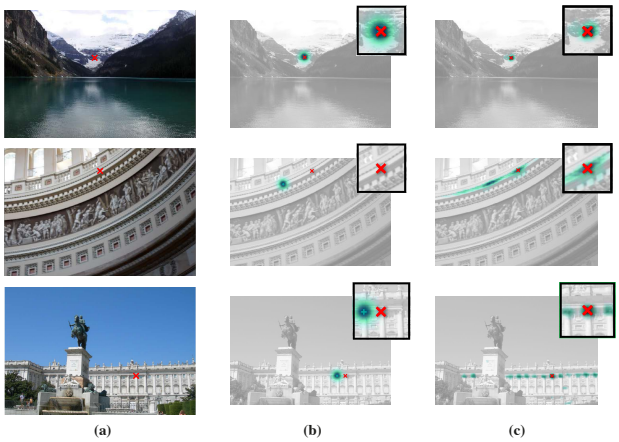


Figure 1. **Neural Reprojection Error (NRE) as a substitute for Reprojection Error (RE): (a)** Given a 3D point $\mathbf{u}$, a query image $\mathtt{I}$ and its ground truth camera pose, $\mathbf{u}$ can be reprojected into the image plane of $\mathtt{I}$ to obtain a 2D point $\times$. **(b)** RE takes as input a camera pose and a putative 2D-3D correspondence between $\mathbf{u}$ and a 2D location $+$ in $\mathtt{I}$, reprojects $\mathbf{u}$ to obtain a 2D point $\mathbf{q}$, computes the euclidean distance between $+$ and $\mathbf{q}$ and finally applies a robust loss function (shown in turquoise as a function of $\mathbf{q}$). In ambiguous (middle) or multimodal (bottom) cases, generating a 2D-3D correspondence may lead to a loss function that conveys erroneous data to the pose estimator. **(c)** NRE does not rely on 2D-3D correspondences, thus $+$ does not exist anymore. Instead, NRE employs a dense loss map (shown in turquoise as a function of $\mathbf{q}$) that contains much more information than RE, especially in ambiguous and multimodal cases. As a result, a pose estimator is significantly more accurate and robust using NRE than RE.

a 2D-3D correspondence and the camera pose. It consists in reprojecting the 3D point, using the camera pose, into the query image plane, computing the euclidean distance between this reprojection and its putative 2D correspondent, and applying a robust loss function, such as Geman-McClure or Tukey's biweight [4,55]. The robust loss allows to reduce the influence of outlier 2D-3D correspondences.

We argue that this strong decoupling of the matching stage from the P*n*P stage limits both the accuracy and the robustness of the camera pose estimate. Generating putative 2D-3D correspondences leads to an important loss of information since the 3D model and the query image are summarized into a set of 2D-3D coordinates. This loss of information needs to be compensated as far as possible within RE through the choice of a robust loss and the tuning of its hyperparameters, which usually depend on both the visual content and the amount of outliers generated by the matching stage. Moreover, outlier correspondences convey erroneous data to the pose estimator (see fig. 1).

Contributions:

(i) We propose the *Neural Reprojection Error* (NRE) as a substitute for RE. NRE does not require a 2D-3D correspondence as input but relies on a *dense loss map*. A *dense loss map* contains much more information than a simple 2D-3D correspondence and conveys data of higher quality to the pose estimator. As a result, the need for choosing a robust loss and its hyperparameters is also eliminated. Computing a *dense loss map* essentially involves cross-correlations between descriptors that are extracted using a neural network, hence the name *Neural Reprojection Error*.

(ii) Our derivation of NRE makes it differentiable not only w.r.t. to the camera pose but also w.r.t. the descriptors. Thus, providing ground-truth camera poses and minimizing NRE w.r.t. the descriptors yields a well-posed feature learning problem tailored for pose estimation. NRE merges the feature learning problem and the camera pose estimation problem in a new way and allows to rethink the recent end-to-end direct feature metric pose refinement methods that need to consider two different losses.

(iii) To estimate the camera pose efficiently, we propose to minimize a sum of NRE terms in a coarse-to-fine manner. As a result, we never compute or store any high-resolution dense loss map. We also describe how to perform the optimization using an M-estimator sample consensus approach followed by a graduated non-convexity procedure. We experimentally demonstrate that our novel NRE-based pose estimator is a good substitute for RE-based pose estimators as it significantly improves both the robustness and the accuracy of the camera pose estimate while being computationally and memory highly efficient.

In the remainder of the paper, after discussing the related work, we introduce some notations and describe our method. We provide a detailed discussion to highlight the differences between NRE and existing approaches. We finally present our evaluation results.

2. Related work

NRE has connections with several research areas, namely, feature learning, learning to match features, end-to-end camera pose estimation and robust optimization. A detailed literature review on these topics seems out of the scope of this paper. Instead, for each topic, we will explain how NRE is related to it and refer the reader to recent papers containing a detailed literature review on it.

Feature learning methods [6,7,16,17,19,27,29,32,33,40,49,51,53,54] learn to transform an image into robust dense descriptors. Minimizing NRE w.r.t. the descriptors allows to learn features tailored for pose estimation. Our training loss is similar to the one proposed in S2DNet [19]. Thus S2DNet features are in theory well suited to be used as inputs of our novel NRE-based pose estimator. However, as we show in our experiments, S2DNet computes by nature high-resolution dense correspondence maps which is both computationally and memory highly inefficient, hence making our NRE-based pose estimator impractical. By merging feature learning and pose estimation, our loss intrinsically integrates a bilinear interpolation operator. It allows us to learn *coarse* robust features and *fine* discriminative features which we combine in a coarse-to-fine strategy. As a result, using our "NRE features" as input, our NRE-based pose estimator is both fast and memory highly efficient but also significantly more robust and accurate compare to the case where we use S2DNet features as input.

Learning-based matching methods [11, 13, 31, 34, 41, 57] take descriptors and/or putative correspondences as input and output probabilities of correspondences. NRE takes as input dense loss maps which are essentially the negative logarithm of probabilities of correspondences. In our current formulation, we do not employ any sophisticated learning-based matching method to produce these inputs, but a single dot product between descriptors followed by a softmax. Using a state-of-the-art matching architecture would likely improve the results of NRE but we left this as future work.

End-to-end camera pose estimation methods [9, 10, 12, 22,23,28,47,52] learn jointly all the parameters of the camera pose estimator by backpropagating through it. Different architectures of camera pose estimators have been proposed in the literature. Among these architectures, end-to-end feature metric pose refinement methods [28, 47, 52], are the ones that are the closest to NRE as their architectures explicitly minimize a sum of reprojection errors by leveraging richer information than simple 2D-3D correspondences. In Sec. 7.2 we provide a detailed discussion to explain the fundamental differences between these methods and NRE.

Robust optimization methods [2–5, 15, 56] are tailored to

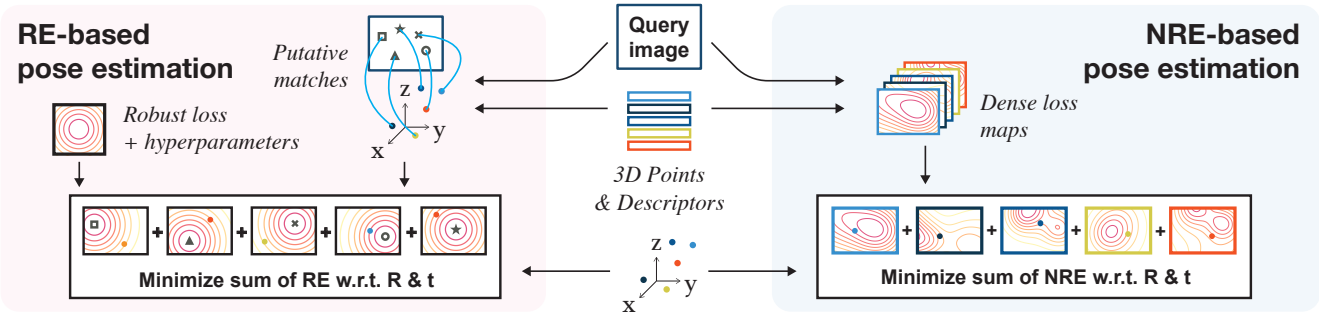


Figure 2. NRE-based pose estimator *vs*. RE-based pose estimator. **Left:** In an RE-based pose estimator, putative 2D-3D correspondences are initially established. Then, a sum of RE terms is minimized, w.r.t. the camera pose, using these correspondences, a robust loss and its hyperparameters as inputs. **Right:** In an NRE-based pose estimator, dense loss maps are computed instead of 2D-3D correspondences. Then the query pose is estimated by minimizing a sum of NRE terms, effectively leveraging richer information than simple 2D-3D correspondences and alleviating the need for choosing a robust loss and its hyperparameters.

minimize a sum of non-convex terms. This is essentially what the P*n*P stage seeks to achieve as it consists in minimizing a sum of RE terms. In Sec. 7.1 we provide a detailed discussion to highlight the fact that RE is a special case of NRE which allows to relate NRE to standard robust optimization problems. From another point of view, recent methods, such as [3, 5], allow to eliminate the need for setting the hyperparameter of the robust loss by marginalizing it. NRE is also able to eliminate this need but in a very different manner. Consequently, in the experiments we will compare the performances of these RE-based estimators against our novel NRE-based estimator.

3. Background and notations

In this paper, we assume a sparse 3D point cloud $\{\mathbf{u}_n^{\mathtt{G}}\}_{n=1\ldots N}$, whose coordinates are expressed in a global coordinate system $\mathtt{G}$, as well as a database $\mathcal{D}$ of geo-localized (w.r.t. $\mathtt{G}$) reference images are given, and we seek to estimate the pose (*i.e.* the rotation matrix $\mathtt{R}_{\mathtt{QG}}$ and the translation vector $\mathbf{t}_{\mathtt{QG}}$) of a query image $\mathtt{I}_{\mathtt{Q}}$ coming from a calibrated camera.

Dense descriptors $\mathtt{H}_{\mathtt{Q}}$ of $\mathtt{I}_{\mathtt{Q}}$ are extracted using a convolutional neural network $\mathcal{F}$ with parameters Θ: $\mathtt{H}_{\mathtt{Q}} := \mathcal{F}(\mathtt{I}_{\mathtt{Q}}; \Theta)$. Similarly, $\mathcal{F}$ is used to compute a set of descriptors $\{\mathbf{h}_n\}_{n=1\ldots N}$ for each 3D point $\{\mathbf{u}_n^{\mathtt{G}}\}_{n=1\ldots N}$ in the database $\mathcal{D}$.

The warping function $\omega(\mathbf{u}_n^{\mathtt{G}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}) := \mathtt{K}\pi\left(\mathtt{R}_{\mathtt{QG}}\mathbf{u}_n^{\mathtt{G}} + \mathbf{t}_{\mathtt{QG}}\right)$ allows to warp a 3D point $\mathbf{u}_n^{\mathtt{G}}$ to obtain a 2D point $\mathbf{p}_n^{\mathtt{Q}}$ onto the image plane of $\mathtt{I}_{\mathtt{Q}}$, *i.e.* $\mathbf{p}_n^{\mathtt{Q}} = \omega\left(\mathbf{u}_n^{\mathtt{G}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)$, where $\mathtt{K}$ is the camera calibration matrix and $\pi(\mathbf{u}) := [\mathbf{u}_x/\mathbf{u}_z, \mathbf{u}_y/\mathbf{u}_z]^{\mathtt{T}}$ is the projection function.

Let us now introduce the concept of *correspondence map*. In this paper, the correspondence map $\mathtt{C}_{\mathtt{Q},n}$ of $\mathbf{u}_n^{\mathtt{G}}$ in $\mathtt{I}_{\mathtt{Q}}$ is computed as follows: $\mathtt{C}_{\mathtt{Q},n} = g\left(\mathbf{h}_n * \mathtt{H}_{\mathtt{Q}}\right)$ where g is the softmax function and $*$ is the spatial convolution operator. The value $\mathtt{C}_{\mathtt{Q},n}\left(\mathbf{p}_n^{\mathtt{Q}}\right)$ describes how likely it is that pixel location $\mathbf{p}_n^{\mathtt{Q}}$ in $\mathtt{I}_{\mathtt{Q}}$ corresponds to $\mathbf{u}_n^{\mathtt{G}}$. $\mathtt{C}_{\mathtt{Q},n}$ also has an extra category $\mathbf{p}_n^{\mathtt{Q}} = \mathbf{out}$ that corresponds to the case where $\mathbf{u}_n^{\mathtt{G}}$ is not seen in $\mathtt{I}_{\mathtt{Q}}$. By definition, $\mathtt{C}_{\mathtt{Q},n}\left(\mathbf{p}_n^{\mathtt{Q}} = \mathbf{out}\right) := 0$. Thus, $\mathtt{C}_{\mathtt{Q},n}$ has $|\mathring{\Omega}_{\mathtt{Q}}| = 1 + H_{\mathtt{Q}} \times W_{\mathtt{Q}}$ categories, where $H_{\mathtt{Q}}$ and $W_{\mathtt{Q}}$ are the number of rows and columns of $\mathtt{H}_{\mathtt{Q}}$, $\Omega_{\mathtt{Q}}$ is the set of all the pixel locations in $\mathtt{H}_{\mathtt{Q}}$ and $\mathring{\Omega}_{\mathtt{Q}} := \{\Omega_{\mathtt{Q}}, \mathbf{out}\}$.

The following notations will also be useful: $[\![\cdot]\!]$ is the Iverson bracket ($[\![\text{True}]\!] = 1$ and $[\![\text{False}]\!] = 0$), $\lfloor\cdot\rfloor$ is the floor function and $\|\cdot\|$ is the L2 norm.

4. Neural reprojection error

In this section, we first introduce the standard RE and then we present our novel NRE.

4.1. Reprojection error

The RE, that is used by most of the camera pose estimation methods, corresponds to the following equation:

$$\mathrm{RE}\left(\mathbf{u}_n^{\mathtt{G}}, \mathbf{p}_n^{\mathtt{Q}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right) := \psi_\sigma\left(\left\|\mathbf{p}_n^{\mathtt{Q}} - \omega\left(\mathbf{u}_n^{\mathtt{G}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right\|\right) \quad (1)$$

where $\{\mathbf{p}_n^{\mathtt{Q}}, \mathbf{u}_n^{\mathtt{G}}\}$ is a 2D-3D correspondence and $\psi_\sigma(\cdot)$ is a parametric robust loss, such as Geman-McClure or Tukey's biweight [4, 55], that allows to reduce the influence of large residuals. Estimating the camera pose by minimizing a sum of RE terms enforces the 3D model and the query image to be summarized into a set of putative correspondences which results in a significant and irreversible loss of information. This loss of information needs to be compensated as far as possible through the choice of a robust loss and the tuning of its hyperparameters, that usually depend on both the visual content and the outliers distribution. Moreover, outlier correspondences convey erroneous data to the pose estimator. On the contrary, our novel loss, which we introduce in the next section, leverages richer information from the 3D model and the query image than RE and as a result eliminates the need for choosing a robust loss and its hyperparameters.

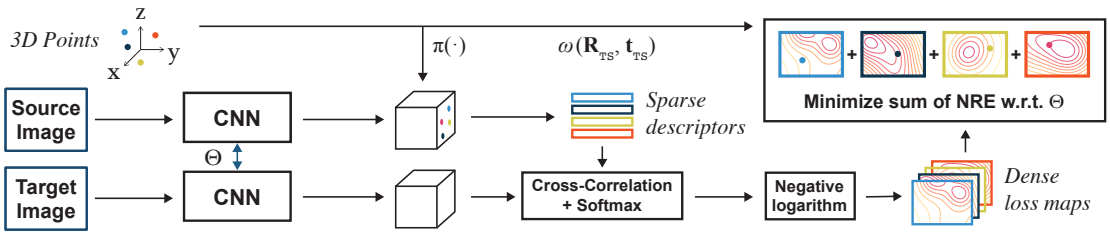


Figure 3. **Learning image descriptors tailored for camera pose estimation:** Given a pair of target/source images ($\mathtt{I_T}$, $\mathtt{I_S}$), 3D points $\{\mathbf{u}_n^{\mathtt{S}}\}_{n=1\ldots N}$ (seen in both $\mathtt{I_S}$ and $\mathtt{I_T}$) and ground truth camera poses ($\mathtt{R_{TS}}$ and $\mathbf{t}_{\mathtt{TS}}$), we first extract dense representations for both images. For each 3D, we compute dense loss maps and minimize the S2D-CE loss with respect to the backbone parameters Θ.

4.2. Our novel loss

Instead of computing the loss as a robust parametric function of the euclidean distance between the reprojected 3D point and its putative 2D correspondent in the query image, our novel loss function evaluates the discrepancy between two probability mass functions (pmf): the *matching* pmf and the *reprojection* pmf. In the rest of this section, we first define these two pmf and then introduce our novel loss.

Matching probability mass function: This pmf describes how likely it is that the descriptor at the 2D image location $\mathbf{p}_n^{\mathtt{Q}}$ in $\mathtt{H_Q}$ corresponds to the descriptor $\mathbf{h}_n$ of the 3D point $\mathbf{u}_n^{\mathtt{G}}$.

$$q_{\mathrm{m}}\left(\mathbf{p}_n^{\mathtt{Q}}|s_n,\mathtt{H_Q},\mathbf{h}_n\right) := s_n\,\mathtt{C}_{\mathtt{Q},n}\left(\mathbf{p}_n^{\mathtt{Q}}\right) + \frac{1-s_n}{|\mathring{\Omega}_{\mathtt{Q}}|}\,, \tag{2}$$

where the binary selector variable $s_n \in \{0,1\}$ allows to choose between two components: the predicted correspondence map and the outlier uniform pmf. The latter component introduces robustness against erroneous correspondence maps that may occur because of non-covisibility, occlusions, failure of the deep network, etc. We show in Fig. 4(b) an example of the negative logarithm of a correspondence map.

Reprojection probability mass function: This pmf describes how likely it is that a 2D location $\mathbf{p}_n^{\mathtt{Q}} \in \mathring{\Omega}_{\mathtt{Q}}$ corresponds to the reprojection of a 3D point $\mathbf{u}_n^{\mathtt{G}}$ using camera pose $\mathtt{R_{QG}}$ and $\mathbf{t}_{\mathtt{QG}}$.

$$\begin{aligned} q_{\mathrm{r}}\left(\mathbf{p}_n^{\mathtt{Q}}|\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right) := \\ \mathtt{w}_{00,n}\left[\!\left[\mathbf{p}_n^{\mathtt{Q}} = \lfloor\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\rfloor\right]\!\right] + \\ \mathtt{w}_{10,n}\left[\!\left[\mathbf{p}_n^{\mathtt{Q}} = \lfloor\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\rfloor + [1,0]^{\mathtt{T}}\right]\!\right] + \\ \mathtt{w}_{01,n}\left[\!\left[\mathbf{p}_n^{\mathtt{Q}} = \lfloor\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\rfloor + [0,1]^{\mathtt{T}}\right]\!\right] + \\ \mathtt{w}_{11,n}\left[\!\left[\mathbf{p}_n^{\mathtt{Q}} = \lfloor\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\rfloor + [1,1]^{\mathtt{T}}\right]\!\right]\,, \end{aligned} \tag{3}$$

where the weights $\mathtt{w}_{i,j}$ are bilinear interpolation coefficients, *i.e.*

$$\begin{aligned} \mathtt{w}_{00,n} &:= (1-x_n)(1-y_n)\,, & \mathtt{w}_{10,n} &:= x_n(1-y_n)\,, \\ \mathtt{w}_{01,n} &:= (1-x_n)\,y_n\,, & \mathtt{w}_{11,n} &:= x_n y_n\,, \end{aligned}$$

with

$$\begin{aligned} x_n &:= \left(\lfloor\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\rfloor - \omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\right)_x\,, \\ y_n &:= \left(\lfloor\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\rfloor - \omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\right)_y\,. \end{aligned}$$

Equation 3 sets a non-zero weight to the four image locations surrounding the reprojection of the 3D point $\mathbf{u}_n^{\mathtt{G}}$ under camera pose parameters $\mathtt{R_{QG}}$ and $\mathbf{t}_{\mathtt{QG}}$, and a zero weight to the rest of the image. In a slight abuse of notation, if a reprojection $\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)$ falls outside of the image boundaries or if the 3D point has negative depth, *i.e.* $\left(\mathtt{R_{QG}}\mathbf{u}_n^{\mathtt{G}} + \mathbf{t}_{\mathtt{QG}}\right)_z \leq 0$, we consider that

$$\begin{aligned} &\lfloor\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\rfloor + [\cdot,\cdot]^{\mathtt{T}} := \mathbf{out} \text{ and} \\ &\mathtt{w}_{00,n} := 1,\ \mathtt{w}_{10,n} = \mathtt{w}_{01,n} = \mathtt{w}_{11,n} := 0. \end{aligned}$$

We show in Fig. 4(d) an example of a reprojection pmf.

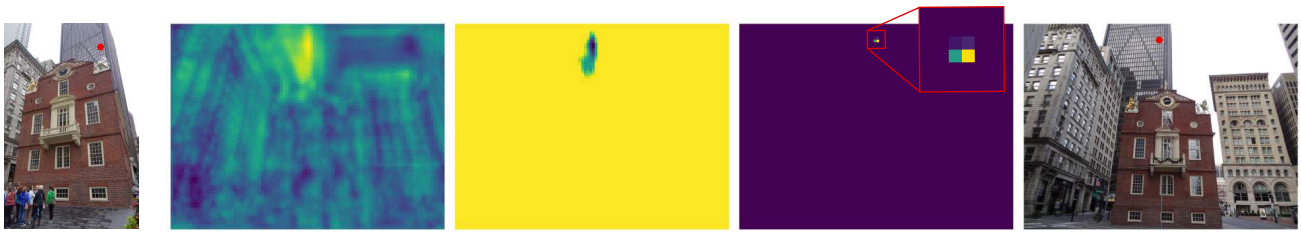
Assuming perfect descriptors and a perfect camera pose, the two pmf should be the same. This analysis is the fundamental idea of this paper: (a) given ground truth camera pose, we will make the *matching* pmf fit the *reprojection* pmf to learn descriptors tailored for pose estimation, (b) given descriptors, we will make the *reprojection* pmf fit the *matching* pmf to estimate the camera pose.

We propose to evaluate the discrepancy between the *matching* pmf (Eq. 2) and the *reprojection* pmf (Eq. 3) using the following Cross-Entropy (CE):

$$\begin{aligned} &\mathrm{CE}\left(q_{\mathrm{r}}\left(\mathbf{p}_n^{\mathtt{Q}}|\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)||q_{\mathrm{m}}\left(\mathbf{p}_n^{\mathtt{Q}}|s_n,\mathtt{H_Q},\mathbf{h}_n\right)\right) \\ &= -\sum_{\mathbf{p}_n^{\mathtt{Q}}\in\mathring{\Omega}_{\mathtt{Q}}} q_{\mathrm{r}}\left(\mathbf{p}_n^{\mathtt{Q}}|\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\ln\left(q_{\mathrm{m}}\left(\mathbf{p}_n^{\mathtt{Q}}|s_n,\mathtt{H_Q},\mathbf{h}_n\right)\right) \\ &= s_n\tilde{\mathtt{C}}_{\mathtt{Q},n}\left(\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\right) + (1-s_n)\ln|\mathring{\Omega}_{\mathtt{Q}}| \\ &:= \mathrm{NRE}\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{H_Q},\mathbf{h}_n,\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}},s_n\right) \end{aligned} \tag{4}$$

where $\tilde{\mathtt{C}}_{\mathtt{Q},n}(\mathbf{p}) := -\ln\left(\mathtt{C}_{\mathtt{Q},n}(\mathbf{p})\right)\ \forall\mathbf{p}\in\mathring{\Omega}_{\mathtt{Q}}$ is called a *dense loss map*. The notation $\tilde{\mathtt{C}}_{\mathtt{Q},n}\left(\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)\right)$ corresponds to performing a bilinear interpolation at location $\omega\left(\mathbf{u}_n^{\mathtt{G}},\mathtt{R_{QG}},\mathbf{t}_{\mathtt{QG}}\right)$ in $\tilde{\mathtt{C}}_{\mathtt{Q},n}$.

From the point of view of the 3D point $\mathbf{u}_n^{\mathtt{G}}$, Eq. 4 is a reprojection loss that depends on descriptors extracted by a convolutional neural network $\mathcal{F}$ (see Sec. 3). Thus, we will refer to Eq. 4 as the *Neural Reprojection Error*.



(a) (b) (c) (d) (e)

Figure 4. **Visualizations of maps involved in the derivation of NRE:** We show an example of **(a)** a source image and a reprojected 3D point $\mathbf{u}^{\mathtt{G}}_n$ using ground truth camera pose, **(b)** the non-robust dense loss map $\tilde{\mathtt{C}}_{\mathtt{Q},n}$ with respect to the target image **(e)**, **(c)** the robust dense loss map $\mathtt{L}_{\mathtt{Q},n}$, and **(d)** the target reprojection probability mass function used at training time.

From a practical point of view, given query dense descriptors $\mathtt{H}_{\mathtt{Q}}$ as well as 3D points and descriptors $\{\mathbf{u}^{\mathtt{G}}_n, \mathbf{h}_n\}_{n=1\ldots N}$, it is possible to estimate the camera pose by minimizing a sum of NRE terms w.r.t. $\mathtt{R}_{\mathtt{QG}}$, $\mathbf{t}_{\mathtt{QG}}$ and $\{s_n\}_{n=1\ldots N}$ (see Sec. 5). Here, NRE relies on the dense loss maps directly which significantly reduces the amount of lost information compared to RE. Consequently, the need for choosing a robust loss and its hyperparameters is eliminated and all the information is kept available to estimate the camera pose.

Our novel NRE is differentiable not only w.r.t. to the camera pose but also w.r.t. the descriptors $\mathtt{H}_{\mathtt{Q}}$ and $\mathbf{h}_n$. Thus, providing ground-truth camera poses and minimizing NRE w.r.t. the descriptors yields a well-posed feature learning problem tailored for the pose estimation (see Sec. 6). NRE merges the feature learning problem and the camera pose estimation problem in a new way and allows to rethink the recent end-to-end feature metric pose refinement (see Sec. 7.2).

5. Camera pose estimation

Our novel NRE can be used to estimate the camera pose. Given a query image, from which query dense descriptors $\mathtt{H}_{\mathtt{Q}}$ are extracted, as well as 3D points and descriptors $\{\mathbf{u}^{\mathtt{G}}_n, \mathbf{h}_n\}_{n=1\ldots N}$, we obtain a camera pose estimate by minimizing the following sum of NRE terms (Eq. 4) w.r.t. $\mathtt{R}_{\mathtt{QG}}$ and $\mathbf{t}_{\mathtt{QG}}$:

$$\begin{aligned}
&\mathcal{L}\left(\mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right) \\
&= \min_{s_1, s_2, \ldots, s_N} \sum_{n=1}^{N} \mathrm{NRE}\left(\mathbf{u}^{\mathtt{G}}_n, \mathtt{H}_{\mathtt{Q}}, \mathbf{h}_n, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}, s_n\right) \\
&= \sum_{n=1}^{N} \min\left(\ln|\mathring{\Omega}_{\mathtt{Q}}|, \tilde{\mathtt{C}}_{\mathtt{Q},n}\left(\omega\left(\mathbf{u}^{\mathtt{G}}_n, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right)\right) \qquad (5) \\
&\approx \sum_{n=1}^{N} \mathtt{L}_{\mathtt{Q},n}\left(\omega\left(\mathbf{u}^{\mathtt{G}}_n, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right) \qquad (6)
\end{aligned}$$

where the loss maps $\mathtt{L}_{\mathtt{Q},n}$ are defined as follows:

$$\mathtt{L}_{\mathtt{Q},n}\left(\mathbf{p}\right) := \min\left(\ln|\mathring{\Omega}_{\mathtt{Q}}|, \tilde{\mathtt{C}}_{\mathtt{Q},n}\left(\mathbf{p}\right)\right) \forall \mathbf{p} \in \mathring{\Omega}_{\mathtt{Q}}\,. \qquad (7)$$

Instead of performing a bilinear interpolation in $\tilde{\mathtt{C}}_{\mathtt{Q},n}$ followed by a truncation as in Eq. 5, we apply a truncation to each element of $\tilde{\mathtt{C}}_{\mathtt{Q},n}$ once (Eq. 7) and then perform a bilinear interpolation (Eq. 6). This approximation enables both a sparse storage of each loss map $\mathtt{L}_{\mathtt{Q},n}$ and an efficient smoothing procedure (see Sec. 5.2).

Our loss function is robust against outliers, since large values in $\tilde{\mathtt{C}}_{\mathtt{Q},n}$ are truncated at $\ln|\mathring{\Omega}_{\mathtt{Q}}|$. We show in Fig. 4(c) an example of a robust *dense loss map* ($\mathtt{L}_{\mathtt{Q},n}$).

Minimizing Eq. 6 is a non-convex optimization problem, thus we proceed in two steps: a sampling-based initialization step followed by gradient-based refinement step.

5.1. Initialization step

To obtain an initial pose estimate, we employ an *M-estimator SAmple Consensus* approach (MSAC) [50]. The method is very similar to a RANdom SAmple Consensus approach (RANSAC) [18]) but does not require any user defined inlier/outlier threshold. Each iteration consists of 1) randomly sampling 3 loss maps, 2) estimating a putative camera pose from these 3 loss maps and 3) evaluating Eq. 6 with that putative camera pose. Step 2 can be efficiently implemented using a standard P3P solver since:

$$\begin{aligned}
&\arg\min_{\mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}} \sum_{n=1}^{3} \mathtt{L}_{\mathtt{Q},n}\left(\omega\left(\mathbf{u}^{\mathtt{G}}_n, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right) \\
&= \mathrm{P3P}\left(\left\{\mathbf{u}^{\mathtt{G}}_n, \arg\min_{\mathbf{p}} \mathtt{L}_{\mathtt{Q},n}\left(\mathbf{p}\right)\right\}_{n=1\ldots 3}\right). \qquad (8)
\end{aligned}$$

5.2. Refinement step

Refining the initial camera pose remains a difficult optimization problem since each loss map in Eq. 6 may have plateaus and local minima (see Fig. 1 middle and bottom rows) and the initial pose estimate may not be accurate enough for a gradient-based method to avoid a poor local minimum.

Thus, we employ a Graduated Non-Convexity approach (GNC) [8] that builds a sequence of successively smoother (and therefore easier to optimize) approximations of the original loss function. The optimization scheme consists

of optimizing the sequence of loss functions, with the solution from the previous objective used as starting point for the next one. However, Eq. 6 is not a standard robust optimization problem [56]. Therefore, we propose to apply a Gaussian-homotopy-like method [30] and consider the following smoothed version of the original loss function (a derivation of that equation is given in the appendix):

$$\breve{\mathcal{L}}_{\sigma}\left(\mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right) := \sum_{n=1}^{N} \sum_{\mathbf{q} \in \Gamma_{\mathtt{Q},n}} \text{-}\left(\ln |\mathring{\Omega}_{\mathtt{Q}}| \text{-} \mathtt{L}_{\mathtt{Q},n}\left(\mathbf{q}\right)\right) k_{\sigma}\left(\left\|\mathbf{q} \text{-} \omega\left(\mathbf{u}_{n}^{\mathtt{G}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right\|\right) \quad (9)$$

where $k_{\sigma}\left(\|\mathbf{r}\|\right) := \frac{1}{2\pi\sigma^2} e^{-\frac{\|\mathbf{r}\|^2}{2\sigma^2}}$ is an isotropic Gaussian kernel with standard variation σ and $\Gamma_{\mathtt{Q},n}$ is the set of pixel locations whose corresponding values in $\tilde{\mathtt{C}}_{\mathtt{Q},n}$ have not been truncated in Eq. 7. In Eq. 9, a large value of σ leads to a highly smoothed version of the original loss function while a small value of σ corresponds to a loss function that is very similar to Eq. 6. Therefore, in practice, we will start the optimization with a value of σ that is large enough, to avoid getting stuck in a poor local minimum and progressively decrease its value. Since Eq. 9 is a standard robust optimization problem, we employ an Iterated Reweighted Least Squares (IRLS) approach to minimize each optimization problem within the GNC [8] and use the stopping criterion proposed in [56].

5.3. Coarse-to-fine strategy

From a practical point of view, the robustness and the accuracy of the camera pose estimate directly depends on the loss maps, especially their resolution. However, producing high resolution loss maps is an inefficient strategy: most of the computational time would be spent computing cross-correlations in regions distant from the true correspondent locations. Instead, we propose a coarse-to-fine strategy: we first estimate a coarse camera pose using low-resolution loss maps and then refine it using local high-resolution ones.

For a given query image of size $H \times W \times 3$, we proceed as follows: 1) Coarse dense descriptors of size $H/16 \times W/16 \times 1280$ are extracted using a *coarse* network ($\mathcal{F}_{\text{coarse}}$). 2) Low-resolution loss maps of size $H/16 \times W/16$ are computed. 3) We run an MSAC [50]+P3P to obtain an initial coarse pose estimate. 4) We apply a GNC [8] procedure (still using low-resolution correspondence maps) to refine that initial coarse estimate. 5) Fine dense descriptors of size $H/2 \times W/2 \times 288$ are extracted using a *fine* network ($\mathcal{F}_{\text{fine}}$). 6) Local high-resolution loss maps of size 64×64 are computed at the location of the reprojected 3D points using the coarse pose estimate. 7) We apply a GNC [8] procedure starting from the coarse pose estimate to obtain our final pose estimate. Implementation details are provided in the appendix.

This coarse-to-fine strategy allows to obtain a camera pose estimate very efficiently while significantly reducing the amount of required memory, since we never compute or store any high-resolution loss map (see Tab. 3).

6. Learning image descriptors

Our novel camera pose estimation method (see Sec. 5) essentially consists in minimizing a sum of NRE terms, w.r.t. the camera pose, assuming that the underlying descriptor extractor networks $\mathcal{F}_{\text{coarse}}$ and $\mathcal{F}_{\text{fine}}$ provide robust and discriminative descriptors. Therefore, we need to learn these networks. Let us recall that NRE (Eq. 4) is differentiable w.r.t. the descriptors $\mathtt{H}_{\mathtt{Q}}$ and $\mathbf{h}_n$. Thus we can learn to extract descriptors using NRE as training loss. We provide pairs of target/source images ($\mathtt{I}_{\mathtt{T}}$, $\mathtt{I}_{\mathtt{S}}$), 3D points $\{\mathbf{u}_n^{\mathtt{S}}\}_{n=1\ldots N}$ (seen in both $\mathtt{I}_{\mathtt{S}}$ and $\mathtt{I}_{\mathtt{T}}$) and ground truth camera poses ($\mathtt{R}_{\mathtt{TS}}$ and $\mathbf{t}_{\mathtt{TS}}$). For each pair of images, we perform gradient descent over the following loss function (see Fig. 3):

$$\mathcal{L}\left(\Theta\right) = \sum_{n=1}^{N} \text{NRE}\left(\mathbf{u}_n^{\mathtt{S}}, \mathtt{H}_{\mathtt{T}}, \mathbf{h}_n, \mathtt{R}_{\mathtt{TS}}, \mathbf{t}_{\mathtt{TS}}, s_n = 1\right), \quad (10)$$

with $\mathtt{H}_{\mathtt{T}} = \mathcal{F}\left(\mathtt{I}_{\mathtt{T}}; \Theta\right)$, $\mathtt{H}_{\mathtt{S}} = \mathcal{F}\left(\mathtt{I}_{\mathtt{S}}; \Theta\right)$ and $\mathbf{h}_n = \mathtt{H}_{\mathtt{S}}\left(\mathtt{K}\pi\left(\mathbf{u}_n^{\mathtt{S}}\right)\right)$. The selector variable s_n is set to one in order to ease the gradient propagation. As explained in Sec. 5.3, in practice, we employ two networks: a *coarse* network $\mathcal{F}_{\text{coarse}}$ and a *fine* network $\mathcal{F}_{\text{fine}}$. Thus we need to train two networks with different architectures, which are detailed in the appendix.

7. Discussion

7.1. RE is a special case of NRE

In RE-based pose estimation, we are given 2D-3D correspondences $\{\mathbf{u}_n^{\mathtt{G}}, \mathbf{p}_n^{\mathtt{Q}}\}_{n=1\ldots N}$. Let us consider a single 2D-3D correspondence. Assuming that $\mathbf{p}_n^{\mathtt{Q}}$ has integer pixel coordinates, we can build a one-hot-encoded correspondence map $\mathtt{C}_{\mathtt{Q},n}$ such that $\mathtt{C}_{\mathtt{Q},n}\left(\mathbf{p}_n^{\mathtt{Q}}\right) = 1$ and zeros everywhere else. In this case, Eq. 7 is a dense loss map $\mathtt{L}_{\mathtt{Q},n}$ that equals zero at the location $\mathbf{p}_n^{\mathtt{Q}}$ and $\ln |\mathring{\Omega}_{\mathtt{Q}}|$ everywhere else, and Eq. 9 becomes:

$$\mathcal{L}_{\sigma}(\mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}) = \sum_{n=1}^{N} \text{-} \ln |\mathring{\Omega}_{\mathtt{Q}}| k_{\sigma}\left(\left\|\mathbf{p}_n^{\mathtt{Q}} \text{-} \omega\left(\mathbf{u}_n^{\mathtt{G}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right\|\right). \quad (11)$$

In Eq. 11, each term within the sum corresponds to Eq. 1 with a negative gaussian function as robust loss, whose shape is similar to the truncated quadratic kernel [55]. Thus, RE is a special case of NRE. In the experiments, we will consider minimizing Eq. 11 to fairly compare RE *vs*. NRE.

Features	Pose estimator	Hyperparam.	Translation Error			Rotation Error		
			0.25m	1m	5m	2°	5°	10°
S2DNet [19]	RE *LO-RANSAC* [14]	$\tau = 4$	0.54 (+23%)	0.45 (+32%)	0.33 (+32%)	0.54 (+23%)	0.47 (+27%)	0.45 (+32%)
S2DNet [19]	RE *GC-RANSAC* [2]	$\tau = 4$	0.54 (+23%)	0.43 (+26%)	0.31 (+24%)	0.53 (+20%)	0.47 (+27%)	0.43 (+26%)
S2DNet [19]	RE *MAGSAC++* [5]	N/A	0.51 (+16%)	0.43 (+26%)	0.31 (+24%)	0.51 (+16%)	0.45 (+22%)	0.42 (+24%)
S2DNet [19]	RE *Minimize Eq. 11*	$\sigma = 5$	0.53 (+20%)	0.44 (+29%)	0.31 (+24%)	0.52 (+18%)	0.46 (+24%)	0.43 (+26%)
S2DNet [19]	FPR *Minimize Eq. 12*	Cf. Appendix	0.49 (+11%)	0.42 (+24%)	0.30 (+20%)	0.48 (+ 9%)	0.44 (+19%)	0.42 (+24%)
S2DNet [19]	NRE	N/A	**0.44** (+ 0%)	**0.34** (+ 0%)	**0.25** (+ 0%)	**0.44** (+ 0%)	**0.37** (+ 0%)	**0.34** (+ 0%)

Table 1. **NRE-based vs. RE-based pose estimators:** We evaluate the gain in performance of our novel NRE-based pose estimator against state-of-the-art RE-based pose estimators on the MegaDepth dataset [25]. For a fair comparison, each method employs S2DNet [19] features, even our NRE-based pose estimator. For the methods that have an hyperparameter, we optimized it and report the best results. We report the error at several thresholds for translation and rotation (lower is better). The scores between brackets show the relative deterioration w.r.t. to NRE. We find that our NRE-based pose estimator significantly outperforms all the RE-based estimators. We include the performance of the best feature-metric pose refinement estimator, which is covered in more details in the appendix.

7.2. NRE *vs*. End-to-end feature metric pose refinement

End-to-end Feature metric Pose Refinement (FPR) methods [28,47,52] seek to minimize a loss of the following form at "test-time":

$$\mathcal{L}_\sigma\left(\mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right) := \sum_{i=1}^{N} \psi_\sigma\left(\left\|\mathbf{h}_n\text{-}\mathtt{H}_{\mathtt{Q}}\left(\omega\left(\mathbf{u}_n^{\mathtt{G}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right)\right\|\right). \quad (12)$$

In Eq. 12, each term within the sum consists in reprojecting a 3D point into the query image plane but taking the distance in the space of descriptors. From this point of view, FPR is similar to NRE as it tries to leverage richer image information than simple 2D-3D correspondences. However FPR still requires choosing/learning a robust loss and tuning/learning its hyperparameters, so from this point of view it has the same limitations as RE.

But the major difference between FPR and NRE is that minimizing Eq. 12 w.r.t. the descriptors does not yield a well-posed feature learning problem. In order to learn descriptors tailored for pose estimation, FPR methods must consider at least two losses. In [52], a pixelwise contrastive loss is added (as well as a term involving the Hessian of the pose), while [28] and [47] unroll several steps of an optimizer to obtain a computational graph and use a distance between the ground truth pose and the predicted pose to supervise the training. On the contrary, minimizing NRE w.r.t. the descriptors yields a well-posed feature learning problem. Thus NRE is the first method to unify the feature learning problem and the camera pose estimation problem in a single loss and allows to rethink the end-to-end FPR strategy.

8. Experiments

In this section, we experimentally demonstrate that our novel NRE-based pose estimator significantly outperforms state-of-the art RE-based pose estimators. We also show that our coarse-to-fine strategy markedly reduces the amount of required memory and the overall computational time of our NRE-based pose estimator.

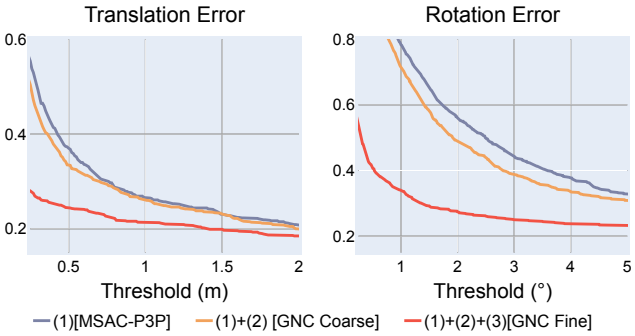


Figure 5. **Ablation study:** We report the cumulative error curves in pose estimation (lower is better), on the hardest category of our Megadepth study. We find that each step of our NRE-based coarse-to-fine estimator brings significant improvements.

8.1. Dataset and method

We assembled an evaluation dataset of 3000 Megadepth [25] image pairs, sampled from the validation set. Using the provided SfM model reconstructed using SIFT [26], we create image pairs which contain at least 50 covisible 3D points. We evenly split them based on their viewpoint distances to create three difficulty categories, which we name *Easy*, *Medium* and *Hard*. At test-time for every pair of source and target images, we aim at predicting the absolute camera pose of the target image, based on the 3D points visible in the source image. We report the pose estimation error for several precision thresholds.

8.2. RE-based *vs*. NRE-based pose estimator

In this first evaluation, we compare RE-based pose estimators against our novel NRE-based pose estimator. In or-

Category	Features	Pose estimator	Translation Error			Rotation Error		
			0.25m	1m	5m	2°	5°	10°
Easy	S2DNet	NRE	0.17 (+ 42%)	0.12 (+100%)	0.09 (+200%)	0.16 (+ 45%)	0.13 (+ 86%)	0.10 (+100%)
	NRE Features	NRE	**0.12** (+ 0%)	**0.06** (+ 0%)	**0.03** (+ 0%)	**0.11** (+ 0%)	**0.07** (+ 0%)	**0.05** (+ 0%)
Medium	S2DNet	NRE	0.29 (+ 53%)	0.20 (+ 67%)	0.15 (+150%)	0.27 (+ 60%)	0.22 (+ 69%)	0.19 (+ 90%)
	NRE Features	NRE	**0.19** (+ 0%)	**0.12** (+ 0%)	**0.06** (+ 0%)	**0.17** (+ 0%)	**0.13** (+ 0%)	**0.10** (+ 0%)
Hard	S2DNet	NRE	0.44 (+ 30%)	0.34 (+ 42%)	0.25 (+108%)	0.44 (+ 33%)	0.37 (+ 37%)	0.34 (+ 42%)
	NRE Features	NRE	**0.34** (+ 0%)	**0.24** (+ 0%)	**0.12** (+ 0%)	**0.33** (+ 0%)	**0.27** (+ 0%)	**0.24** (+ 0%)

Table 2. **NRE-based pose estimator using NRE features *vs*. NRE-based pose estimators using S2DNet features:** We evaluate the gain in performance of our NRE features against S2DNet [19] features using the same NRE-based pose estimator. We compare pose estimation on Megadepth [25] images evenly split in three difficulty categories. We report the error at several thresholds for translation and rotation (lower is better). The scores between brackets show the relative deterioration w.r.t. to NRE features. We show that using our NRE features, the resulting estimated pose is markedly more accurate than using S2DNet features.

Features Pose estimator	S2DNet RE	S2DNet NRE	NRE NRE
Feature extraction	28.2ms	28.2ms	N/A
Feature extraction *coarse*	N/A	N/A	15.5ms
Feature extraction *fine*	N/A	N/A	7.2ms
Compute correspondence maps	300ms	300ms	N/A
Compute *coarse* correspondence maps	N/A	N/A	8ms
Compute *local fine* correspondence maps	N/A	N/A	3ms
Pose initialization (single iteration)	0.9ms	1.1ms	1.1ms
Pose refinement	0.11s	0.61s	N/A
Pose refinement *coarse*	N/A	N/A	0.15s
Pose refinement *fine*	N/A	N/A	0.28s
Total features memory	2949MB	2949MB	591MB
Total correspondence maps memory	7680MB	7680MB	46MB

Table 3. **Computational time and memory requirement study:** We report the average inference time on Megadepth [25] images with 1000 3D points. We show that our coarse-to-fine approach enables a much faster pose estimation and allows for larger scene scaling.

der to have a fair comparison, we use S2DNet [19] features for all methods evaluated in this study.

Baselines: We compare our NRE-based pose estimator against multiple state-of-the-art RE-based pose estimators. This includes LO-RANSAC [14], GC-RANSAC [2] and MAGSAC++ [5], which all aim at finding inlier correspondences from putative matches. We also add the minimization of Eq. 11 and Eq. 12.

For all RE-based pose estimators, we follow S2DNet [19] and provide raw putative 2D-to-3D matches based on the correspondence map argmax location. For our NRE estimator, we use the same correspondence maps but preserve all the information. For all methods requiring hyperparameter tuning, we run several evaluations to find the optimal one on our dataset. More details are provided in the appendix.

Results: We report pose estimation errors for the aforementionned methods in Tab. 1. We find our NRE-based pose estimator consistently provides significant improvements over other RE-based estimators. In addition as shown in the appendix, we find hyperparameter tuning has a significant impact on performance for parametric RE estimators. Our NRE-estimator however, requires no tuning.

8.3. Coarse-to-fine experiment

We provide an ablation study in Fig. 5 of our coarse-to-fine strategy. We find that each step of our NRE-based estimator brings significant improvements. We now compare the performance coupling the NRE estimator with NRE features trained on the same training set as S2DNet [19], using our coarse-to-fine strategy. We report in Tab. 2 the pose estimation error on all categories from our Megadepth [25] benchmark. We find that using NRE features brings an additional leap in performance, by up to 200%. Thanks to our coarse-to-fine formulation, this is all achieved at a fraction of the cost of S2DNet [19]. As reported in Tab. 3, NRE features have a memory footprint which is over 16 times lighter, while also performing a lot faster. This is a key component for practical applications, or when scaling up to larger amount of keypoints or images. Additional qualitative and quantitative results are provided in the appendix.

9. Conclusion

In this paper, we introduced the *Neural Reprojection Error* (NRE) as a substitute for the widely used Reprojection Error (RE). NRE allows to perform absolute camera pose estimation by leveraging richer information than RE and eliminates the need for choosing a robust loss and its hyperparameters. We also proposed a coarse-to-fine optimization strategy that allows to very efficiently minimize a sum of NRE terms w.r.t. the camera pose. We experimentally demonstrated that replacing RE with NRE significantly improved the accuracy and the robustness of the camera pose estimate while being computationally and memory highly efficient. Our derivation of NRE merges the feature learning problem and the absolute camera pose estimation problem in a new way that allows to rethink the end-to-end feature-metric pose refinement strategy. From a broader point of view, we believe this new way of merging deep learning and 3D geometry may be useful in other computer vision applications.

Appendix

In the following pages, we present additional quantitative results, qualitative results and experimental details about the Neural Reprojection Error.

A. Additional Experiments

A.1. NRE-based pose estimator *vs*. Feature metric Pose Refinement

We compare our novel NRE-based pose estimator against Feature-Metric Pose Refinement (FPR) methods. As explained in Section 7.2, FPR methods seek to minimize Eq. 12. As such, FPR benefits from dense information contained in query feature maps, but requires to choose a robust loss function and tune its hyperparameters.

To complement our RE-based *vs*. NRE-based pose estimators study presented in Tab. 1, we propose to reuse S2DNet [19] features to perform FPR, initialized from our best RE pose estimator (MAGSAC++ [5]). To merge information from all three feature extraction levels from S2DNet [19], we try upsampling and concatenating descriptors, as well as a coarse-to-fine alternative in which we iteratively refine predictions from the previous (coarser) level.

We report pose estimation errors in Tab. 4 for FPR and NRE estimators. We show results using the Huber [21] robust loss as well as the Barron [4] loss. We find that NRE performs consistently better while eliminating the need for choosing a robust loss.

A.2. Experiments on Aachen Night [35]

So far, we evaluated the performances of our NRE-based pose estimator on MegaDepth [25]. Here, we run a similar study on the Aachen Night [35, 37] dataset. This challenging outdoor dataset consists of $4,328$ sparsely sampled daytime database images, and 98 nighttime query images. To have a fair comparison between NRE-based and RE-based pose estimators, we pair each query image with an oracle nearest-neighbor database image and use all of its visible 3D points to predict the query pose. Similar to the MegaDepth study, we report results for RE-based, FPR-based and NRE-based pose estimators, using S2DNet features in Tab. 5. For FPR-based pose estimators we pick the best configuration from 4.

As in the MegaDepth experiment, our NRE-based pose estimator consistently provides significant improvement over other pose estimators. We also compare the performance coupling the NRE-based pose estimator with NRE features trained on the same training set as S2DNet [19]. We report in Tab. 6 the pose estimation errors. We again find that using NRE features brings an additional leap in performance.

A.3. Experiments on InLoc [46]

To evaluate the generalization capabilities in an indoor scenario, we run the same experiment on the InLoc [46] dataset. This dataset consists of 329 query images, for $9,972$ database images. Unlike Aachen Night, we have access to dense aligned depth maps for all database images. To provide a fair comparison, we also pair each query image with an oracle nearest-neighbor database image and use SuperPoint [16] detections (lifted to 3D using the depth maps) in the database images as inputs. Results are reported in Tab. 5.

We find that our NRE-based pose estimator provides consistent improvements at the coarsest threshold, and overall competitive performance on the medium and fine ones. The fact the relative improvement brought by our NRE-based pose estimator is not as significant as for the other datasets can be attributed to the domain shift with respect to the training images. Nonetheless, despite being trained on outdoor images we find that our NRE features bring additional improvements compared to S2DNet [19] features, as shown in Tab. 6.

B. Qualitative results

In Fig. 6, we show several examples of query images from the MegaDepth [25] validation set with a reprojected 3D point and the corresponding coarse dense loss map computed using our coarse NRE features. It highlights that the dense loss maps keep much more information than RE. As a consequence, as we show in our experiments, our novel NRE-based pose estimator significantly outperforms RE-based pose estimators.

C. Derivation of Equation 9

In this section, we show how Eq. 8 (in the submited wersion of the paper) is obtained.

The robust dense loss map $\mathtt{L}_{\mathtt{Q},n,\sigma}$ can be smoothed using an isotropic Gaussian kernel as follows:

$$\breve{\mathtt{L}}_{\mathtt{Q},n,\sigma}(\mathbf{p}) := \sum_{\mathbf{r}} k_\sigma(\|\mathbf{r}\|)\, \mathtt{L}_{\mathtt{Q},n}(\mathbf{p}+\mathbf{r})$$

$$= \mathtt{L}_{\mathtt{Q},n}(\mathbf{out}) \sum_{\mathbf{r}} k_\sigma(\|\mathbf{r}\|) + \sum_{\mathbf{r}} k_\sigma(\|\mathbf{r}\|)\left(\mathtt{L}_{\mathtt{Q},n}(\mathbf{p}+\mathbf{r}) - \mathtt{L}_{\mathtt{Q},n}(\mathbf{out})\right) \tag{13}$$

$$= \sum_{\mathbf{r}} k_\sigma(\|\mathbf{r}\|)\left(\mathtt{L}_{\mathtt{Q},n}(\mathbf{p}+\mathbf{r}) - \mathtt{L}_{\mathtt{Q},n}(\mathbf{out})\right) + \mathrm{cst}_{\mathbf{p}} \tag{14}$$

$$= \sum_{\mathbf{q}\in\Omega_{\mathtt{Q}}} k_\sigma(\|\mathbf{q}-\mathbf{p}\|)\left(\mathtt{L}_{\mathtt{Q},n}(\mathbf{q}) - \mathtt{L}_{\mathtt{Q},n}(\mathbf{out})\right) + \mathrm{cst}_{\mathbf{p}} \tag{15}$$

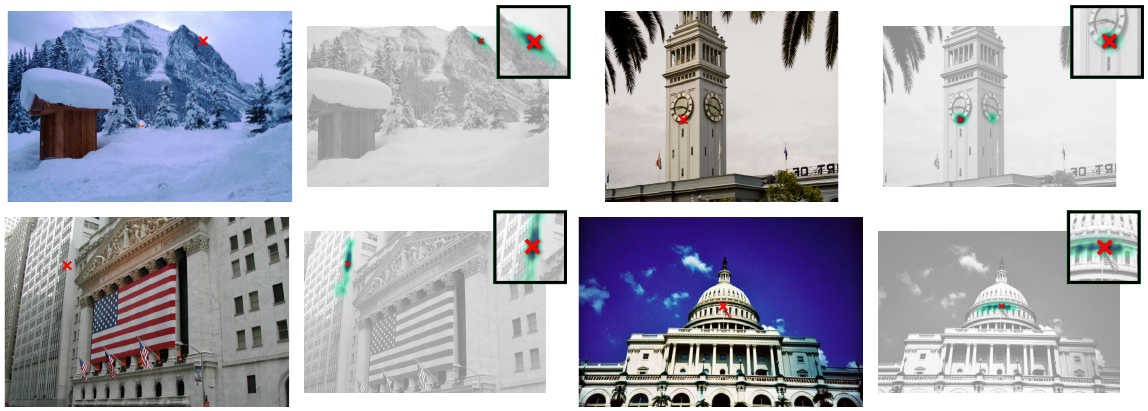


Figure 6. **Qualitative results:** These qualitatives results correspond to additional examples for columns (a) and (b) in Fig.1. It highlights that the dense loss maps keep much more information than RE. As a consequence our novel NRE-based pose estimator significantly outperforms RE-based pose estimators.

Features	Pose estimator	Fusion	ψ	Translation Error			Rotation Error		
				0.25m	1m	5m	2°	5°	10°
S2DNet [19]	RE *MAGSAC++* [5]	N/A	N/A	0.51 (+ 16%)	0.43 (+ 26%)	0.31 (+ 24%)	0.51 (+ 16%)	0.45 (+ 22%)	0.42 (+ 24%)
S2DNet [19]	FPR *Min. Eq. 12*	C2F	Huber [21]	0.70 (+ 59%)	0.65 (+ 91%)	0.52 (+108%)	0.69 (+ 57%)	0.63 (+ 70%)	0.58 (+ 71%)
S2DNet [19]	FPR *Min. Eq. 12*	C2F	Barron [4]	0.55 (+ 25%)	0.44 (+ 29%)	0.30 (+ 20%)	0.55 (+ 25%)	0.48 (+ 30%)	0.43 (+ 26%)
S2DNet [19]	FPR *Min. Eq. 12*	Concat.	Huber [21]	0.49 (+ 11%)	0.42 (+ 24%)	0.30 (+ 20%)	0.48 (+ 9%)	0.44 (+ 19%)	0.42 (+ 24%)
S2DNet [19]	FPR *Min. Eq. 12*	Concat.	Barron [4]	0.49 (+ 11%)	0.42 (+ 24%)	0.30 (+ 20%)	0.48 (+ 9%)	0.44 (+ 19%)	0.42 (+ 24%)
S2DNet [19]	NRE	N/A	N/A	**0.44** (+ 0%)	**0.34** (+ 0%)	**0.25** (+ 0%)	**0.44** (+ 0%)	**0.37** (+ 0%)	**0.34** (+ 0%)

Table 4. **NRE-based pose estimator *vs*. Feature-Metric Pose Refinement:** We evaluate the gain in performance of our novel NRE-based pose estimator against the Feature-Metric Pose Estimation (FPR) variant on the MegaDepth dataset. Here FPR consists in minimizing Eq. 12 using as initialization the camera pose estimate from RE *MAGSAC++* [5]. We find here that minimizing Eq. 12 allows to improve the camera pose estimate from *MAGSAC++*, however our novel NRE again shows superior performance, while requiring no robust kernel selection. The scores between brackets show the relative deterioration w.r.t. to NRE.

$$= \sum_{\mathbf{q}\in\Omega_{\mathtt{Q}}} k_\sigma\left(\|\mathbf{q}-\mathbf{p}\|\right)\left(\mathtt{L}_{\mathtt{Q},n}\left(\mathbf{q}\right) - \ln|\mathring{\Omega}_{\mathtt{Q}}|\right) + \mathrm{cst}_{\mathbf{p}} \quad (16)$$

$$= \sum_{\mathbf{q}\in\Gamma_{\mathtt{Q},n}} k_\sigma\left(\|\mathbf{q}-\mathbf{p}\|\right)\left(\mathtt{L}_{\mathtt{Q},n}\left(\mathbf{q}\right) - \ln|\mathring{\Omega}_{\mathtt{Q}}|\right) + \mathrm{cst}_{\mathbf{p}} \quad (17)$$

where $k_\sigma\left(\|\mathbf{r}\|\right) := \frac{1}{2\pi\sigma^2} e^{-\frac{\|\mathbf{r}\|^2}{2\sigma^2}}$ is an isotropic Gaussian kernel with standard variation σ and $\Gamma_{\mathtt{Q},n}$ is the set of pixel locations whose corresponding values in $\mathtt{L}_{\mathtt{Q},n}$ are lower than $\ln|\mathring{\Omega}_{\mathtt{Q}}|$. Equation 17 leads to the smoothed cost function:

$$\check{\mathcal{L}}_\sigma\left(\mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right) := \sum_{n=1}^{N} \sum_{\mathbf{q}\in\Gamma_{\mathtt{Q},n}} \text{-}\Big(\ln|\mathring{\Omega}_{\mathtt{Q}}| \text{-} \mathtt{L}_{\mathtt{Q},n}\left(\mathbf{q}\right)\Big) k_\sigma\left(\left\|\mathbf{q}\text{-}\omega\left(\mathbf{u}_n^{\mathtt{G}}, \mathtt{R}_{\mathtt{QG}}, \mathbf{t}_{\mathtt{QG}}\right)\right\|\right), \quad (18)$$

which is a robust non-linear least squares problem and therefore can be minimized using the IRLS algorithm.

D. Technical details

D.1. Coarse-to-Fine Strategy (Sec. 5.3)

Step 6 of our coarse-to-fine strategy consists in computing local high-resolution loss maps of size 64×64 at the location of the reprojected 3D points using the coarse pose estimate. The idea of that step is to transform the low-resolution loss maps into high-resolution loss maps to obtain a much more accurate pose estimate. The question is: How can we combine a low-resolution robust loss map with a local high-resolution discriminative loss map ? We proceed as follows:

1. A coarse correspondence map $\mathtt{C}_{\text{coarse}}$ is of size $H/16 \times W/16$. Let us recall that by definition $\sum_{\mathbf{p}\in\mathring{\Omega}_{\text{coarse}}} \mathtt{C}_{\text{coarse}}\left(\mathbf{p}\right) = 1$.

2. Compute the local high resolution correspondence map $\mathtt{C}_{\text{fine}}$ of size 64×64 at the location of the reprojected 3D points (using the coarse pose estimate) $\mathbf{q}$:

Features	Pose Estimator	Aachen Night			InLoc-DUC1			InLoc-DUC2		
		0.25m, 2°	0.5m, 5°	5m, 10°	0.25m, 2°	0.5m, 5°	5m, 10°	0.25m, 2°	0.5m, 5°	5m, 10°
S2DNet	MAGSAC++ [5]	0.46 (+ 55%)	0.28 (+ 80%)	0.10 (+229%)	0.62 (+ 3%)	0.41 (+ 2%)	0.31 (+ 11%)	0.70 (+ 11%)	0.44 (+ 5%)	0.30 (+ 2%)
S2DNet	RE *Min. Eq. 10*	0.32 (+ 7%)	0.20 (+ 27%)	0.08 (+165%)	**0.58** (- 4%)	0.40 (+ 1%)	0.31 (+ 13%)	0.66 (+ 6%)	0.47 (+ 13%)	0.39 (+ 31%)
S2DNet	FPR *Min. Eq. 11*	0.32 (+ 7%)	0.20 (+ 27%)	0.06 (+ 97%)	0.61 (+ 1%)	0.41 (+ 4%)	0.29 (+ 4%)	0.63 (+ 1%)	**0.41** (- 4%)	0.31 (+ 5%)
S2DNet	NRE	**0.30** (+ 0%)	**0.15** (+ 0%)	**0.03** (+ 0%)	0.60 (+ 0%)	**0.39** (+ 0%)	**0.28** (+ 0%)	**0.62** (+ 0%)	0.42 (+ 0%)	**0.29** (+ 0%)

Table 5. **NRE-based *vs*. RE-based *vs*. FPR-based pose estimators on Aachen Night [35] and InLoc [46]:** We evaluate the gain in performance of our novel NRE-based pose estimator against state-of-the-art RE-based and FPR-based pose estimators. For a fair comparison, *each method uses the same oracle nearest-neighbor database image for each query image*. Moreover, each method employs S2DNet [19] features, even our NRE-based pose estimator. For the methods that have an hyperparameter, we optimized it and report the best results. We report the error at several thresholds for translation and rotation (lower is better). The scores between brackets show the relative deterioration w.r.t. to NRE. On Aachen, there is no strong domain shift w.r.t. MegaDepth images that are used to train S2DNet, as a result the dense loss maps are accurate and our NRE-based pose estimator significantly outperforms its competitors. On InLoc, there is a strong domain shift (InLoc is an indoor dataset), as a result the dense loss maps are not very informative and our NRE-based pose estimator does not significantly outperform its competitors.

Features	Pose Estim.	Aachen Night			InLoc-DUC1			InLoc-DUC2		
		0.25m, 2°	0.5m, 5°	5m, 10°	0.25m, 2°	0.5m, 5°	5m, 10°	0.25m, 2°	0.5m, 5°	5m, 10°
S2DNet	NRE	0.30 (+ 12%)	0.15 (+ 37%)	0.03 (+ 55%)	0.60 (+ 1%)	0.40 (+ 3%)	0.28 (+ 10%)	0.63 (+ 1%)	0.42 (+ 10%)	0.30 (+ 3%)
NRE Features	NRE	**0.26** (+ 0%)	**0.11** (+ 0%)	**0.02** (+ 0%)	**0.59** (+ 0%)	**0.39** (+ 0%)	**0.25** (+ 0%)	**0.62** (+ 0%)	**0.38** (+ 0%)	**0.29** (+ 0%)

Table 6. **NRE features *vs*. S2DNet features for NRE-based pose estimators on Aachen Night [35] and InLoc [46]:** We evaluate the gain in performance of our NRE features against S2DNet [19] features using the same NRE-based pose estimator. We compare pose estimation on Aachen Night [35] and InLoc [46] images. For a fair comparison, *each method uses the same oracle nearest-neighbor database image for each query image*. We report the error at several precision thresholds for translation and rotation (lower is better). The scores between brackets show the relative deterioration w.r.t. to NRE features. On Aachen, there is no strong domain shift w.r.t. MegaDepth images that are used to train both S2DNet and our NRE feature, as a result the dense loss maps are accurate and we obtain improvements similar to the ones we obtained in our MegaDepth experiment. On InLoc, there is a strong domain shift (InLoc is an indoor dataset), as a result neither S2DNet dense loss maps nor the dense loss maps obtained using our NRE features are very informative. As a result, the pose estimated ugin NRE features is not markedly more accurate than the pose obtained using S2DNet features.

(a) Extract a 64×64 region in the dense fine descriptors around $\mathbf{q}$.

(b) Compute the dot product with the fine descriptor of the 3D point and apply a softmax to obtain $\mathtt{C}_{\text{fine}}$.

Thus by definition $\sum_{\mathbf{p} \in \mathcal{N}_{64 \times 64}(\mathbf{q})} \mathtt{C}_{\text{fine}}(\mathbf{p}) = 1$.

3. $\mathtt{C}_{\text{fine}}$ corresponds to a region of size 8x8 in $\mathtt{C}_{\text{coarse}}$. Compute the sum of these 64 pixels in $\mathtt{C}_{\text{coarse}}$. We call this scalar $norm_{coarse}$.

4. Multiply $\mathtt{C}_{\text{fine}}$ by $\frac{norm_{coarse}}{64}$ to obtain $\mathtt{C}_{\text{fine norm}}$. $\mathtt{C}_{\text{fine norm}}$ is a local high-resolution version of $\mathtt{C}_{\text{coarse}}$.

5. The final local high resolution loss map is obtained classically: $\mathtt{L}_{\text{fine}} = \min\left(\ln|\mathring{\Omega}_{\text{fine}}|, -\ln(\mathtt{C}_{\text{fine norm}})\right)$. By definition, outside of the 64×64 region, the value of the loss is $\ln|\mathring{\Omega}_{\text{fine}}|$.

D.2. Network Architectures (Sec. 6)

Coarse network architecture. The purpose of the coarse network $\mathcal{F}_{\text{coarse}}$ is to provide robust descriptors that are used to obtain a coarse pose estimate. To deal with ambiguous cases, it should leverage image context. This motivates a deep architecture with a wide receptive field and a large descriptor size. On the other hand, the network should output dense descriptors of sufficient resolution to reliably estimate a coarse camera pose. We experimentally found that an effective stride of 16 is sufficient. To satisfy these specifications, we opted for an Inception-v3 [45] backbone and modified it accordingly. We changed some kernel sizes and truncated the network at the layer Mixed-6e. In the end our final architecture has a receptive field of 927 pixels and produces dense descriptors of size $H/16 \times W/16 \times 1280$.

Fine network architecture. The purpose of the fine network $\mathcal{F}_{\text{fine}}$ is to provide discriminative high-resolution descriptors that are used to refine the coarse pose estimate. However, producing high-resolution descriptors takes a lot of memory. This motivates a deep architecture with a small receptive field and a small descriptor size. We

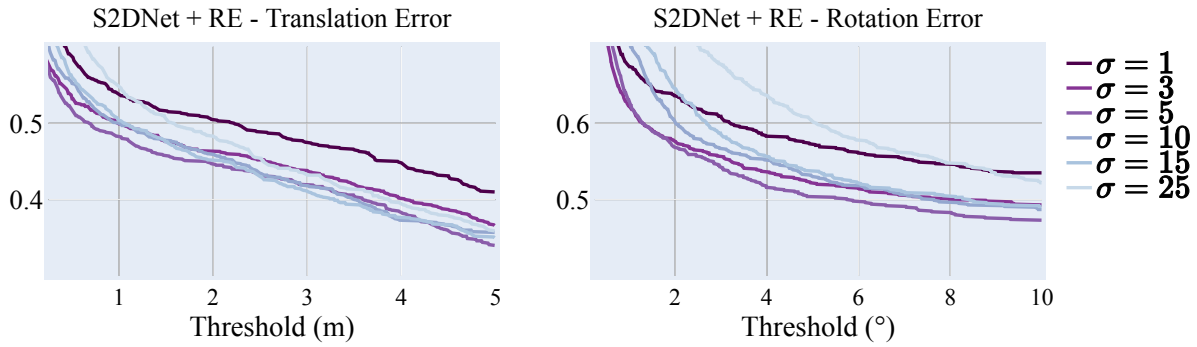


Figure 7. **Tuning the hyperparameter of an RE-based pose estimator:** We report the cumulative error curves in pose estimation (lower is better), on the hardest category of our Megadepth study, for the RE-based pose estimator that consists in minimize Eq. 10. We find that a careful hyperparameter tuning is very important. On the contrary, our novel formalism leads to a loss that does not possess any hyperparameter.

experimentally found that an effective stride of 2 is a good balance between accuracy and memory consumption. To satisfy these specifications, we opted again for a modified Inception-v3 [45] backbone. We only keep the stride of 2 at the first layer and remove any Max-Pooling layer, and we truncate the model at the Mixed-5d layer. Our final architecture has a receptive field of 43 pixels and produces dense descriptors of size $H/2 \times W/2 \times 288$.

Implementation details. The coarse network $\mathcal{F}_{\text{coarse}}$ and the fine network $\mathcal{F}_{\text{fine}}$ are trained independently. Both networks use the same training data which comes from the MegaDepth dataset [25]. As D2-Net [17], we remove scenes which overlap with the PhotoTourism [1,48] test set. We train our networks on image pairs ($\mathtt{I_S}$ and $\mathtt{I_T}$) with an arbitrary overlap.

To train $\mathcal{F}_{\text{fine}}$, we extract random crops of size 800×800 and randomly sample a maximum of 64 3D points visible in both $\mathtt{I_S}$ and $\mathtt{I_T}$. Using such large crops may seem an overkill since $\mathcal{F}_{\text{fine}}$ has a small receptive field. Let us highlight that using $C \times C$ crops allows to produce correspondence maps of size $C/2 \times C/2$ which essentially consists in comparing each source patch against C^2 target patches. Thus, even if $\mathcal{F}_{\text{fine}}$ has a small receptive field, the larger the crops during training the better the descriptors, and 800×800 is the maximum size that could fit in memory.

To train $\mathcal{F}_{\text{coarse}}$, we use entire images as inputs since the network has a very large receptive field and randomly sample a maximum of 64 3D points visible in both $\mathtt{I_S}$ and $\mathtt{I_T}$. Each network is trained using early stopping on the MegaDepth validation set. We use Adam [24] with an initial learning rate of 10^{-3} and apply a multiplicative decaying factor of $e^{-0.1}$ at every epoch.

D.3. Timing

We run all our training and experiments on a machine equipped with an Intel(R) Xeon(R) E5-2630 CPU at 2.20GHz, and an NVIDIA GeForce GTX 1080Ti GPU. The timing results reported in Tab. table:timings where obtained using a Python implementation of the previously described algorithms. Source code will be made available.

D.4. Implementation details about the RE-based *vs.* NRE-based pose estimators study

- In our RE-based *vs.* NRE-based pose estimators study, we used LO-RANSAC [14], GC-RANSAC [2] and MAGSAC++ [5] implementations provided in OpenCV 4.5.0 [1].
- We show in Fig. 7 the cumulative errors curves for several σ values when minimizing Eq. 1 on the hardest category of our Megadepth [25] study. These results stress how important a careful hyperparameter tuning is in standard RE pose estimators.
- Throughout our paper we run the coarse GNC with decreasing σ values ranging from 2.0 to 0.6. For the fine GNC, we use values between 8.0 and 0.6.

Acknowledgement

This project has received funding from the Bosch Research Foundation (*Bosch Forschungsstiftung*).

[1] https://docs.opencv.org/master/d9/d0c/group__calib3d.html

References

[1] Phototourism Challenge, CVPR 2019 Image Matching Workshop. 2019.

[2] D. Barath and J. Matas. Graph-Cut RANSAC. In *CVPR*, pages 6733–6741, 2018.

[3] D. Barath, J. Matas, and J. Noskova. MAGSAC: Marginalizing Sample Consensus. In *CVPR*, 2019.

[4] J. T. Barron. A General and Adaptive Robust Loss Function. In *CVPR*, pages 4331–4339, 2019.

[5] D. Baráth, J. Noskova, M. Ivashechkin, and J. Matas. MAGSAC++, A Fast, Reliable and Accurate Robust Estimator. In *CVPR*, pages 1301–1309, 2020.

[6] A. Benbihi, M. Geist, and C. Pradalier. ELF: EMbedded Localisation of Features in Pre-Trained CNN. In *ICCV*, pages 7940–7949, 2019.

[7] A. Bhowmik, S. Gumhold, C. Rother, and E. Brachmann. Reinforced Feature Points: Optimizing Feature Detection and Description for a High-Level Task. In *CVPR*, pages 4948–4957, 2020.

[8] A. Blake and A. Zisserman. *Visual Reconstruction*. MIT press, 1987.

[9] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother. DSAC – Differentiable RANSAC for Camera Localization. In *CVPR*, 2017.

[10] E. Brachmann and C. Rother. Learning Less Is More - 6D Camera Localization via 3D Surface Regression. *CoRR*, abs/1711.10228, 2017.

[11] E. Brachmann and C. Rother. Neural- Guided RANSAC: Learning Where to Sample Model Hypotheses. In *ICCV*, 2019.

[12] M. Bui, T. Birdal, H. Deng, S. Albarqouni, L. Guibas, S. Ilic, and N. Navab. 6D Camera Relocalization in Ambiguous Scenes via Continuous Multimodal Inference. In *ECCV*, 2020.

[13] C. Choy, J. Lee, R. Ranftl, J. Park, and V. Koltun. High-Dimensional Convolutional Networks for Geometric Pattern Recognition. In *CVPR*, pages 11227–11236, 2020.

[14] O. Chum, J. Matas, and J. Kittler. Locally Optimized RANSAC. In *DAGM-Symposium*, 2003.

[15] O. Chum, T. Werner, and J. Matas. Two-View Geometry Estimation Unaffected by a Dominant Plane. In *CVPR*, pages 772–779, 2005.

[16] D. Detone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-Supervised Interest Point Detection and Description. In *CVPR*, 2018.

[17] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-Net: A Trainable CNN for Joint Description and Detection of Local Features. In *CVPR*, 2019.

[18] M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, 24, 1981.

[19] H. Germain, G. Bourmaud, and V. Lepetit. S2DNet: Learning Image Features for Accurate Sparse-to-Dense Matching. In *ECCV*, 2020.

[20] J. Heinly, J. L. Schönberger, E. Dunn, and J.-M. Frahm. Reconstructing the World* in Six Days *(as Captured by the Yahoo 100 Million Image Dataset). In *CVPR*, 2015.

[21] P. Huber. Robust estimation of a location parameter. *Annals of Mathematical Statistics*, 35:492–518, 1964.

[22] A. Kendall and R. Cipolla. Geometric Loss Functions for Camera Pose Regression with Deep Learning. In *CVPR*, pages 5974–5983, 2017.

[23] A. Kendall, M. Grimes, and R. Cipolla. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In *ICCV*, pages 2938–2946, 2015.

[24] D. P. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. *CoRR*, abs/1412.6980, 2014.

[25] Z. Li and N. Snavely. Megadepth: Learning Single-View Depth Prediction from Internet Photos. In *CVPR*, 2018.

[26] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *IJCV*, 60(2), 2004.

[27] Z. Luo, L. Zhou, X. Bai, H. Chen, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan. Aslfeat: Learning Local Features of Accurate Shape and Localization. In *CVPR*, pages 6589–6598, 2020.

[28] Z. Lv, F. Dellaert, J. M. Rehg, and A. Geiger. Taking a Deeper Look at the Inverse Compositional Algorithm. In *CVPR*, pages 4581–4590, 2019.

[29] A. Mishchuk, D. Mishkin, F. Radenović, and J. Matas. Working Hard to Know Your Neighbor's Margins: Local Descriptor Learning Loss. In *NeurIPS*, 2017.

[30] H. Mobahi and J. W. Fisher. On the Link Between Gaussian Homotopy Continuation and Convex Envelopes. In *CVPR*, pages 43–56, 2015.

[31] K. Moo yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. Learning to Find Good Correspondences. In *CVPR*, pages 2666–2674, 2018.

[32] J. Revaud, C. De Souza, M. Humenberger, and P. Weinzaepfel. R2d2: Reliable and repeatable detector and descriptor. In *NeurIPS*, volume 32, pages 12405–12415. Curran Associates, Inc., 2019.

[33] I. Rocco, R. Arandjelović, and J. Sivic. Efficient Neighbourhood Consensus Networks via Submanifold Sparse Convolutions. *IEEE TPAMI*, 2020.

[34] P.-E. Sarlin, D. Detone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning Feature Matching with Graph Neural Networks. In *CVPR*, 2020.

[35] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. Kahl, and T. Pajdla. Benchmarking 6DOF Outdoor Visual Localization in Changing Conditions. In *CVPR*, 2018.

[36] T. Sattler, A. Torii, J. Sivic, M. Pollefeys, H. Taira, M. Okutomi, and T. Pajdla. Are Large-Scale 3D Models Really Necessary for Accurate Visual Localization? In *CVPR*, 2017.

[37] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. Image Retrieval for Image-Based Localization Revisited. In *BMVC*, 2012.

[38] J. L. Schönberger and J.-M. Frahm. Structure-From-Motion Revisited. In *CVPR*, 2016.

[39] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise View Selection for Unstructured Multi-View Stereo. In *ECCV*, 2016.

[40] X. Shen, C. Wang, X. Li, Z. Yu, J. Li, C. Wen, M. Cheng, and Z. He. RF-Net: An End-To-End Image Matching Network Based on Receptive Field. In *CVPR*, pages 8132–8140, 2019.

[41] W. Sun, W. Jiang, E. Trulls, A. Tagliasacchi, and K. M. Yi. ACNe: Attentive Context Normalization for Robust Permutation-Equivariant Learning. In *CVPR*, pages 11286–11295, 2020.

[42] L. Svärm, O. Enqvist, F. Kahl, and M. Oskarsson. City-Scale Localization for Cameras with Known Vertical Direction. *IEEE TPAMI*, 39(7), 2017.

[43] L. Svärm, O. Enqvist, M. Oskarsson, and F. Kahl. Accurate Localization and Pose Estimation for Large 3D Models. In *CVPR*, 2014.

[44] C. Sweeney, V. Fragoso, T. Höllerer, and M. Turk. Large Scale SfM with the Distributed Camera Model. In *International Conference on 3D Vision*, 2016.

[45] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the Inception Architecture for Computer Vision. In *CVPR*, pages 2818–2826, 2016.

[46] H. Taira, M. Okutomi, T. Sattler, M. Cimpoi, M. Pollefeys, J. Sivic, T. Pajdla, and A. Torii. Inloc: Indoor Visual Localization with Dense Matching and View Synthesis. *CoRR*, abs/1803.10368, 2018.

[47] C. Tang and P. Tan. BA-Net: Dense Bundle Adjustment Network. In *ICLR*, 2019.

[48] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L. Li. YFCC100M: The New Data in Multimedia Research. *Commun. ACM*, 59, 2016.

[49] Y. Tian, X. Yu, B. Fan, F. Wu, H. Heijnen, and V. Balntas. SOSNet: Second Order Similarity Regularization for Local Descriptor Learning. In *CVPR*, 2019.

[50] P. H. Torr and A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, 78(1):138–156, 2000.

[51] M. Tyszkiewicz, P. Fua, and E. Trulls. DISK: Learning Local Features with Policy Gradient. In *NeurIPS*, 2020.

[52] L. Von Stumberg, P. Wenzel, Q. Khan, and D. Cremers. GN-Net: The Gauss-Newton Loss for Multi-Weather Relocalization. *IEEE Robotics and Automation Letters*, 5(2):890–897, 2020.

[53] Q. Wang, X. Zhou, B. Hariharan, and N. Snavely. Learning Feature Descriptors Using Camera Pose Supervision. In *ECCV*, 2020.

[54] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned Invariant Feature Transform. In *ECCV*, 2016.

[55] C. Zach and G. Bourmaud. Iterated Lifting for Robust Cost Optimization. In *BMVC*, 2017.

[56] C. Zach and G. Bourmaud. Descending, Lifting or Smoothing: Secrets of Robust Cost Optimization. In *ECCV*, pages 547–562, 2018.

[57] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao. Learning Two-View Correspondences and Geometry Using Order-Aware Network. In *ICCV*, 2019.