Homography-Based Loss Function for Camera Pose Regression

Clémentin Boittiaux, Ricard Marxer, Claire Dune, Aurélien Arnaubec, Vincent Hugel

Homography-Based Loss Function for Camera Pose Regression

Clémentin Boittiaux[1,2,3], Ricard Marxer[2], Claire Dune[3], Aurélien Arnaubec[1] and Vincent Hugel[3]

***Abstract*—Some recent visual-based relocalization algorithms rely on deep learning methods to perform camera pose regression from image data. This paper focuses on the loss functions that embed the error between two poses to perform deep learning based camera pose regression. Existing loss functions are either difficult-to-tune multi-objective functions or present unstable reprojection errors that rely on ground truth 3D scene points and require a two-step training. To deal with these issues, we introduce a novel loss function which is based on a multiplane homography integration. This new function does not require prior initialization and only depends on physically interpretable hyperparameters. Furthermore, the experiments carried out on well established relocalization datasets show that it minimizes best the mean square reprojection error during training when compared with existing loss functions.**

***Index Terms*—Localization, Deep Learning for Visual Perception.**

I. Introduction

THIS paper addresses the problem of relocating a robot in a place that it has previously visited. In many cases, GPS localization is either not available (*e.g.*, in an underwater environment), very noisy (*e.g.*, cities with high buildings), or insufficient for the target application (*e.g.*, relocalizing a robot inside a building). Under these constraints, it may be possible to estimate a more accurate position for the robot by exploiting its visual observations. The problem, termed visual-based localization [1], consists in retrieving the pose of a camera in a known 3D scene from an instantaneous image. This problem also appears in loop closure when performing Simultaneous Localization And Mapping (SLAM) [1].

Until the 2010s, this was solved with classic computer vision methods leveraging engineered descriptors like SIFT or ORB, and PnP/RANSAC schemes [2], [3]. With recent advances in machine learning, some steps of these methods were greatly outperformed by deep learning based approaches. For instance, learned features have proven to be much more robust and accurate than the aforementioned descriptors for tasks like image classification and image retrieval [4], [5]. By bootstrapping such features, end-to-end gradient-based pose regressors emerged [6], [7], [8], [9]. When quantifying the error from a reference pose, these techniques are all confronted with the need to embed the difference between two poses in $SE(3)$ into a scalar. Some more recent approaches focus on learning robust features thanks to deep learning based algorithms and solve the pose estimation with classical computer vision methods [10], [11].

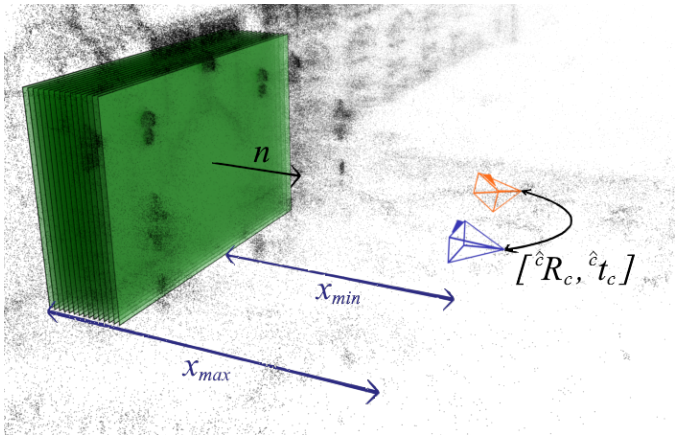


Fig. 1. Set of parallel virtual planes (green) between x_{min} and x_{max} depth used to compute our homography loss that quantifies the pose error $[{}^{\hat{c}}\boldsymbol{R}_c, {}^{\hat{c}}\boldsymbol{t}_c]$ between ground truth (blue) and estimated (orange) cameras. The planes' normal $\boldsymbol{n}$ and the ground truth camera's optical axis are co-linear. Planes are infinite, but for the sake of visualization they are represented as rectangles.

This paper focuses on the loss functions used in deep learning for camera pose regression. One of the main challenges in defining an error in $SE(3)$ is to properly weight the translation and rotation components in the final error. Some losses rely on a scene-agnostic linear combination of these components, that is, they weight translation and rotation errors regardless of what the camera observes [6], [8], [12]. Their physical meaning is often difficult to understand because rotation and translation do not lie on the same group. Moreover, setting the explicit weight for each component usually requires empirical fine-tuning and depends on the scene [6], [8]. Other losses implicitly weight the two components by minimizing the distance between the reprojection of 3D scene points onto the cameras' image planes. These approaches face some issues regarding their initialization and their differentiability in $SE(3)$ [7]. To deal with the problems described above, this paper introduces a new loss function that requires no prior initialization and depends on intuitive parameters. This new loss function approximates the reprojection error as poses get close. We define planes parallel to the ground truth reference image plane for a given range of distances covering the scene. We then express the error in terms of the homographies between the

Manuscript received: November 23, 2021; Revised March 9, 2022; Accepted April 3, 2022.

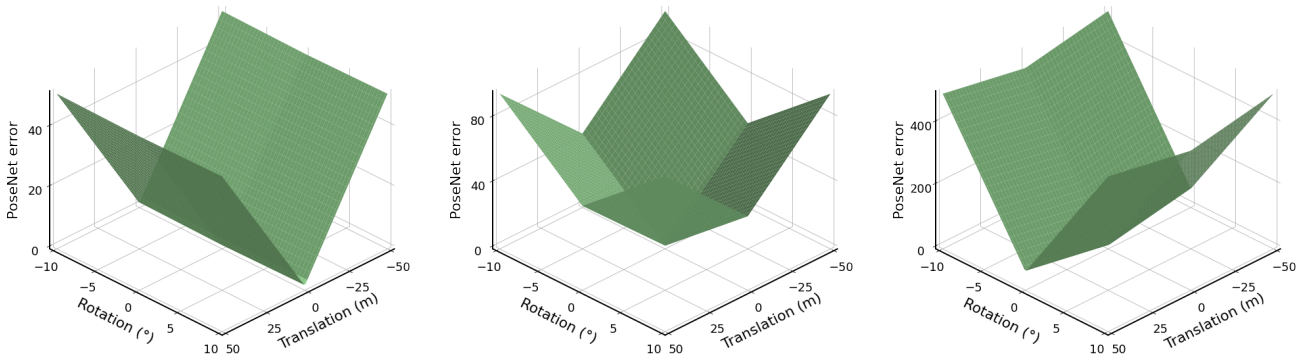
This paper was recommended for publication by Editor Sven Behnke upon evaluation of the Associate Editor and Reviewers' comments. This work was supported by Ifremer

[1]Ifremer, Zone Portuaire de Bregaillon, La Seyne-sur-Mer, France firstname.name@ifremer.fr

[2]Université de Toulon, Aix Marseille Univ, CNRS, LIS, Toulon, France

[3]Université de Toulon, COSMER, Toulon, France

Digital Object Identifier (DOI): see top of this page.



(a) $\beta = 5$ (b) $\beta = 500$ (c) $\beta = 5000$

Fig. 2. Effect of β on the PoseNet loss (1) for a particular scene: (a) low β values lead to high variation of error with respect to translation and little change with respect to rotation; (b) well-chosen β leads to a clear local minimum around the optimal parameters; (c) a large β induces a small variation of translation.

ground truth and estimated poses induced by these planes. The experiments carried out on popular relocalization datasets show that this new loss minimizes best the reprojection error during training when compared with previously mentioned losses.

Section II reviews existing loss functions [6], [7], [8], [12] and discusses their theoretical and practical characteristics. Section III describes the new loss function for camera pose regression. Section IV details the implementation of the new loss and the existing losses mentioned in Section II in a simple end-to-end camera relocalization pipeline using a single backbone architecture[1]. Section V is dedicated to the performance evaluation of all the losses on the Cambridge [6] and 7-Scenes [13] datasets, and to the discussion of the results.

II. Existing pose regression loss functions

Previous computer vision approaches to camera pose regression have focused on active search methods [3]. They are purely geometry-based and rely on image keypoints extraction. With the recent renaissance of neural networks and deep learning techniques, we have seen a paradigm shift arise: end-to-end pose regression with convolutional neural networks allow to directly infer a pose from training image data [6], [7], [12]. These methods are often far more robust to noise and easy to use, but less accurate [1]. Nowadays, state-of-the-art algorithms try to keep the best of both approaches by replacing some steps of geometry-based methods using deep learning techniques [8], [9], [10], [11].

Visual-based relocalization aims at finding the pose $(\boldsymbol{t}, \boldsymbol{q}) \in SE(3)$ of a camera expressed in a world reference frame, where $\boldsymbol{t} = {}^w\boldsymbol{t}_c \in \mathbb{R}^3$ and $\boldsymbol{q} = {}^w\boldsymbol{q}_c \in SO(3)$ are respectively the 3D position and quaternion vector representing the orientation of the camera in the world frame.

In all end-to-end deep learning based methods, the cornerstone is the loss function that embeds into a scalar the error between an estimated pose $(\hat{\boldsymbol{t}}, \hat{\boldsymbol{q}})$ and the ground truth $(\boldsymbol{t}, \boldsymbol{q})$ in $SE(3)$. We make a review of existing loss functions and present their characteristics.

[1]Our code will be made publicly available at github.com/clementinboittiaux/homography-loss-function

A. Loss functions

a) PoseNet [6]: this loss function weights the contribution of translation and rotation errors into a single quantity using a scale factor:

$$\mathcal{L}_P = \left\| \hat{\boldsymbol{t}} - \boldsymbol{t} \right\|_2 + \beta \left\| \hat{\boldsymbol{q}} - \frac{\boldsymbol{q}}{\|\boldsymbol{q}\|} \right\|_2 \tag{1}$$

where β is a positive scalar that weights rotation importance over translation.

b) Homoscedastic uncertainty [7], [12]: tries to reach an optimal balance between rotation and translation errors. It is achieved by optimizing global scalars $\hat{s}_t$ and $\hat{s}_q$ through backpropagation of the following loss function:

$$\mathcal{L}_{HU} = \left\| \hat{\boldsymbol{t}} - \boldsymbol{t} \right\|_1 e^{-\hat{s}_t} + \hat{s}_t + \left\| \boldsymbol{q} - \frac{\hat{\boldsymbol{q}}}{\|\hat{\boldsymbol{q}}\|} \right\|_1 e^{-\hat{s}_q} + \hat{s}_q \tag{2}$$

where $\hat{s}_t$ and $\hat{s}_q$ respectively represent the natural logarithm of the translational and rotational homoscedastic task noise variance.

c) Geometric reprojection loss [7]: a function derived from the classical reprojection error. Its training needs some 3D points of the scene for each view. Let $\pi(\boldsymbol{t}, \boldsymbol{q}, \boldsymbol{P})$ be the function that projects a 3D point $\boldsymbol{P}$ into the camera view with pose $(\boldsymbol{t}, \boldsymbol{q})$, the Geometric reprojection loss function is defined as:

$$\mathcal{L}_G = \frac{1}{|\mathcal{G}|} \sum_{\boldsymbol{P}_i \in \mathcal{G}} \left\| \pi(\boldsymbol{t}, \boldsymbol{q}, \boldsymbol{P}_i) - \pi(\hat{\boldsymbol{t}}, \hat{\boldsymbol{q}}, \boldsymbol{P}_i) \right\|_1 \tag{3}$$

where $\mathcal{G}$ is the subset of 3D points observed by the current view.

d) MaxError [8]: DSAC final pose regression relies on the following loss function that we will refer to as MaxError:

$$\mathcal{L}_{ME} = max \left(\measuredangle(\boldsymbol{q}, \hat{\boldsymbol{q}}), \left\| \hat{\boldsymbol{t}} - \boldsymbol{t} \right\| \right) \tag{4}$$

where $\measuredangle(\boldsymbol{q}, \hat{\boldsymbol{q}})$ is the measured angle in degrees between rotations in 3D space induced by $\boldsymbol{q}$ and $\hat{\boldsymbol{q}}$, $\boldsymbol{t}$ and $\hat{\boldsymbol{t}}$ are expressed in cm in order to reduce the magnitude gap between translation and rotation. Note that in the original paper, this loss is only applied as a step in their full relocalization pipeline. Here, we only evaluate the performance of the loss within a much simpler end-to-end pose regressor.

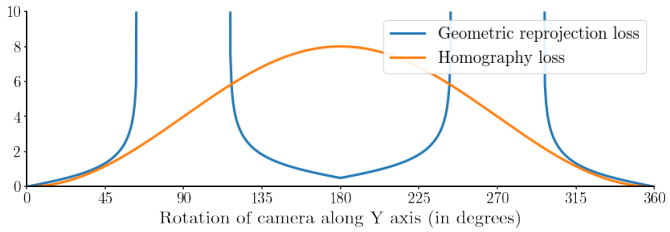


Fig. 3. Evolution of the Geometric reprojection loss function (blue) and our Homographic loss (orange) when the relative pose of the ground truth and estimated cameras varies with a rotational movement along the Y axis.

B. Loss functions characteristics

An issue with PoseNet loss is that β is not easily set. The rotation error is defined as the $L2$ norm between $\hat{\boldsymbol{q}}$ and $\boldsymbol{q}$ unit quaternions. This does not relate to any intuitive geometric phenomena. Moreover, all poses are optimized with the same relative weight between translational and rotational errors, no matter what the camera observes. Furthermore, translation and rotation errors are not comparable metrics. Fig. 2 shows the influence of β on PoseNet's loss function. A small value induces strong translational gradients but a flat profile in orientation, whereas high β values assign importance to rotation over flat translation evolution. Given a scene, a well chosen β allows optimizing the parameters in all dimensions. PoseNet is a muti-objective loss and presents the common problems encountered in such setting. The β selection is not obvious and needs to be determined through trial and error. Even with a properly set β, stochastic optimization may converge to different optima on the Pareto front.

MaxError solves the compromise between translational and rotational errors by fixing a heuristically chosen scale between them (*i.e.*, translation in cm and rotation in degrees). While the scale is physically interpretable, it shares the other issues with PoseNet related to the multi-objective optimization. In this case, the problem is tackled by minimizing the highest error at each step. Like PoseNet, it also shares the same global weighting between translation and rotation errors for all frames.

The Homoscedastic loss [7], [12] reveals characteristics similar to PoseNet and MaxError losses. However, its parameters are more robust to the initialization, since they are optimized during training.

Geometric reprojection loss implicitly solves the translation and rotation weighting problem by directly computing the reprojection error of the observed 3D points in each image. Furthermore, the contributions of rotations and translations can be found automatically and locally in each viewpoint. However, this loss is often unstable. As it will be detailed in the next section, it highly depends on the initialization of the estimated poses and on the scene points visible from the ground truth camera. Poor pose estimates and outlier points will easily lead to divergence in the case of stochastic optimization. Heuristic procedures such as pre-training initialization (using a different technique) or clipping of the error are required to stabilize training. Moreover, the loss presents a wide undesirable local minimum when points are projected on the backside of the image plane (see Fig. 3).

III. The Proposed Homography-Based Loss Function

This section introduces a novel loss function for pose regression based on homographies between ground truth and estimated poses. Through this formulation, we aim to approximate the reprojection error while avoiding some of its drawbacks.

A. Motivations for approximating the reprojection error

The reprojection error consists in measuring the 2D distance between the projection of a set of 3D points into two camera views. If the poses are identical, then the points are superimposed. It has been widely used to solve computer vision problems such as mosaicking or 3D reconstruction [14]. Its physical meaning is easy to understand because it can be represented graphically in the image plane. Furthermore, under the commonly used Gaussian assumption on the reprojection noise, the least square minimization of this error is equivalent to the maximum likelihood estimator which is asymptotically efficient [15].

Its use in deep learning models is more cumbersome for multiple reasons. It relies on the choice of the 3D points that are projected on the image plane. Depending on the method used to estimate the camera ground truth poses, 3D points of the scene may already be available. However, if the scene geometry is not available, one might have to triangulate 3D points from the camera poses and 2D-2D matches. Moreover, in the initial state of the neural network, pose predictions are initialized around an arbitrary value that is often far from the ground truth. Some points can be projected to infinity if they are in the camera (x, y) plane (see Fig. 4). Reprojection

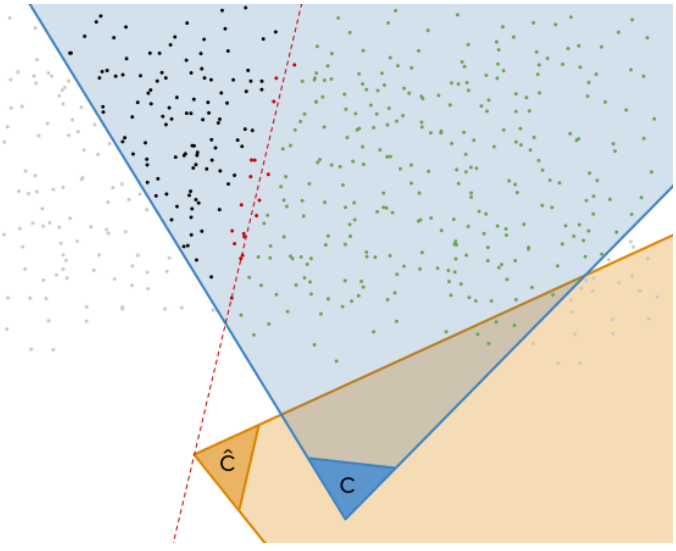


Fig. 4. Top view representation of the ground truth (blue) and estimated (orange) camera views and the scene point cloud. The grey points are outside the field of view of the ground truth. The green points are in front of the image planes of the two cameras. The dashed red line represents the x, y plane of the estimated camera frame. The red dots that are close to this plane are projected to infinity through the pinhole projection model. The black dots project backward from the image plane of the estimated camera.

error may also lead to a local minimum when 3D points are projected onto the backside of the image plane (see Fig. 3). To overcome these problems, the network is usually initialized by first training it with another loss function for a few epochs [7]. In addition, if a point is projected to infinity during the optimization, it results in an infinite loss, causing the model to diverge. In practice, this problem is often addressed by clipping reprojection error distances that exceed a threshold. However, in doing so, all clipped points lay on a flat maximum with a zero-valued gradient, therefore not contributing to the optimization.

Alternatively, the homography-based loss function we design tackles this issue by relying on the integration of homographies computed for a set of virtual parallel planes. It offers a competitive accuracy and a high numerical stability making a simple single step learning possible.

B. Homography-based loss function

The idea behind this work is that the 3D points used to express a pose error such as a reprojection error do not need to be real points, but can well be a set of designated virtual (or hypothetical) points. To eliminate the infinite error problem, we could, for instance, regularly sample virtual 3D points located in front of the image planes of both cameras. Nevertheless, as the poses become more distant from each other, shared 3D observations become scarce and virtual point sampling becomes increasingly difficult.

Following this rationale, but eliminating the problems related to the choice of points, we can cut the scene into planes containing an infinity of these points. We can then calculate our error directly in the homographies induced by these planes. The homography being defined as the transformation of the same 3D plane from one projective view to another. That is, given a plane in the 3D scene and its projected points in a camera view, the homography maps these points into another camera view.

When the poses are superimposed, their homography induced by any plane not containing the center of the cameras is equal to the identity matrix. In Appendix A, we extend this property by demonstrating that, given a specific ${}^c\boldsymbol{n}$, poses are superimposed if and only if their homography is equal to the identity matrix. Thus, for a given plane, we may quantify the error as the difference between the identity matrix and the homography induced by that plane between the ground truth and the estimated poses. We define the error for that plane as the squared Frobenius norm of the resulting difference matrix. We then integrate these errors for all possible planes between two given boundaries (see Fig. 1).

Let us define the homography [14]:

$$ {}^{\hat{c}}\boldsymbol{H}_c = {}^{\hat{c}}\boldsymbol{R}_c - {}^{\hat{c}}\boldsymbol{t}_c \, {}^c\boldsymbol{n}^T / x \tag{5} $$

where ${}^c\boldsymbol{n}$ is the normal to the considered plane expressed in the ground truth camera frame, x is the distance to that plane and ${}^{\hat{c}}\boldsymbol{R}_c$, ${}^{\hat{c}}\boldsymbol{t}_c$ are the rotation and translation of the ground truth camera expressed in the estimated camera frame. For the sake of clarity, we will further use the notation $\boldsymbol{H} = {}^{\hat{c}}\boldsymbol{H}_c$.

Next, we show how the homographic error, defined as the squared Frobenius norm of its difference with the identity matrix, is related to the reprojection error. Let $\boldsymbol{P}$ be a 3D point observed by two cameras. Let $\boldsymbol{p} = (p_x, p_y, 1)^T$ and $\boldsymbol{p'} = (p'_x, p'_y, 1)^T$ be the 2D homogeneous representations of the projection of $\boldsymbol{P}$ in two different camera views. The reprojection error of $\boldsymbol{p}$ is defined as:

$$ e(\boldsymbol{p}) = (p_x - p'_x)^2 + (p_y - p'_y)^2 \tag{6} $$

$$ = (\boldsymbol{p} - \boldsymbol{p'})^T (\boldsymbol{p} - \boldsymbol{p'}) \tag{7} $$

We now assume that $\boldsymbol{P}$ is in a plane that induces a homography $\boldsymbol{H}$ between the two camera views. We want to retrieve the reprojection error by expressing $\boldsymbol{p'}$ in terms of $\boldsymbol{Hp}$. Let us explicit $\boldsymbol{H}$ components:

$$ \boldsymbol{H} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{pmatrix} \tag{8} $$

We note $\boldsymbol{p''}$ the 2D homogeneous point resulting from $\boldsymbol{Hp}$:

$$ \boldsymbol{p''} = \boldsymbol{Hp} = (p''_x, p''_y, s)^T \tag{9} $$

where $s = h_{31} p_x + h_{32} p_y + 1$.
By the definition of the homography, $\boldsymbol{Hp} \sim \boldsymbol{p'}$ in homogeneous coordinates. Thus, in the euclidean space:

$$ \boldsymbol{p'} = \frac{\boldsymbol{p''}}{s} = \frac{\boldsymbol{Hp}}{s} \tag{10} $$

When replacing (10) into (7) we can express the reprojection error in terms of $\boldsymbol{H}$:

$$ e(\boldsymbol{p}) = \left(\boldsymbol{p} - \frac{\boldsymbol{Hp}}{s} \right)^T \left(\boldsymbol{p} - \frac{\boldsymbol{Hp}}{s} \right) \tag{11} $$

$$ = \boldsymbol{p}^T \left(\boldsymbol{I} - \frac{\boldsymbol{H} + \boldsymbol{H}^T}{s} + \frac{\boldsymbol{H}^T \boldsymbol{H}}{s^2} \right) \boldsymbol{p} \tag{12} $$

where $\boldsymbol{I}$ is the identity matrix.
As the estimated pose tends towards the ground truth pose, s tends towards 1. We will use the approximation $s \approx 1$ to simplify (12). This way, our homographic error will tend to the reprojection error when poses are close. Then, (12) becomes:

$$ e(\boldsymbol{p}) = \boldsymbol{p}^T (\boldsymbol{I} - \boldsymbol{H})^T (\boldsymbol{I} - \boldsymbol{H}) \, \boldsymbol{p} \tag{13} $$

While we have now expressed $e(\boldsymbol{p})$ in terms of $\boldsymbol{H}$, this error still relies on specific 2D points in the camera view. As we do not want our loss to rely on any specific 3D point, it should not rely either on their 2D projection. Working around this, we could consider that $\boldsymbol{p}$ could be any point on the sensor. To cover all possibilities, we integrate the error over the entire sensor. To facilitate this task, we will take the trace of our error to use the trace cyclic property. Since our error is a scalar, it is equal to its trace:

$$ e(\boldsymbol{p}) = \mathrm{Tr} \left(\boldsymbol{p}^T (\boldsymbol{I} - \boldsymbol{H})^T (\boldsymbol{I} - \boldsymbol{H}) \, \boldsymbol{p} \right) \tag{14} $$

Then, we can use the cyclic property of the trace to isolate $\boldsymbol{p}$:

$$ e(\boldsymbol{p}) = \mathrm{Tr} \left(\boldsymbol{p}\boldsymbol{p}^T (\boldsymbol{I} - \boldsymbol{H})^T (\boldsymbol{I} - \boldsymbol{H}) \right) \tag{15} $$

with

$$ \boldsymbol{p}\boldsymbol{p}^T = \begin{pmatrix} p_x^2 & p_x p_y & p_x \\ p_y p_x & p_y^2 & p_y \\ p_x & p_y & 1 \end{pmatrix} \tag{16} $$

Let w and h be the respective width and height of our sensor, we can integrate the error over all points of our sensor:

$$\int_{-w/2}^{w/2}\int_{-h/2}^{h/2} \mathrm{Tr}\left(\boldsymbol{p}\boldsymbol{p}^T(\boldsymbol{I}-\boldsymbol{H})^T(\boldsymbol{I}-\boldsymbol{H})\right)\mathrm{d}p_x\,\mathrm{d}p_y \tag{17}$$

$$= \mathrm{Tr}\left(\begin{pmatrix} \frac{hw^3}{12} & 0 & 0 \\ 0 & \frac{wh^3}{12} & 0 \\ 0 & 0 & wh \end{pmatrix}(\boldsymbol{I}-\boldsymbol{H})^T(\boldsymbol{I}-\boldsymbol{H})\right) \tag{18}$$

This results in a diagonal matrix simply weighting the dimensions of the reprojection according to the size of the sensor. As we want our loss to be generic to the size of the sensor, we will simply drop this matrix.

We finally have our homographic error which, by definition, because $(\boldsymbol{I}-\boldsymbol{H})$ is real, is equivalent to a Frobenius norm:

$$\mathrm{Tr}\left((\boldsymbol{I}-\boldsymbol{H})^T(\boldsymbol{I}-\boldsymbol{H})\right) = \|\boldsymbol{I}-\boldsymbol{H}\|_F^2 \tag{19}$$

We further extend the definition of the single plane homographic error (19) to a slab[2]. We integrate the expression over the planes within a given range of distances and along a particular direction. Let x_{min} and x_{max} be the minimum and maximum distances of the planes containing our observations. We introduce the analytic form of our homography-based loss function:

$$\mathcal{L}_H = \frac{1}{x_{max}-x_{min}}\int_{x_{min}}^{x_{max}} \|\boldsymbol{I}-\boldsymbol{H}\|_F^2\,\mathrm{d}x \tag{20}$$

Note that we normalize the loss by the region of the considered scene dimension ($x_{max}-x_{min}$). This is because every frame has its own distance range of observations. By normalizing, we ensure that each frame cost is on the same scale. We can then solve the integral by substitution of (5) in (20) resulting in our final loss function:

$$\mathcal{L}_H = \mathrm{Tr}\left(\boldsymbol{A}+\boldsymbol{B}\frac{\ln\left(\frac{x_{max}}{x_{min}}\right)}{x_{max}-x_{min}} + \frac{\boldsymbol{C}}{x_{min}\cdot x_{max}}\right) \tag{21}$$

where

$$\boldsymbol{A} = (\boldsymbol{I}-{}^{\hat{c}}\boldsymbol{R}_c)(\boldsymbol{I}-{}^{\hat{c}}\boldsymbol{R}_c)^T \tag{22}$$

$$\boldsymbol{B} = {}^{c}\boldsymbol{n}\,{}^{\hat{c}}\boldsymbol{t}_c^T(\boldsymbol{I}-{}^{\hat{c}}\boldsymbol{R}_c) + \left({}^{c}\boldsymbol{n}\,{}^{\hat{c}}\boldsymbol{t}_c^T(\boldsymbol{I}-{}^{\hat{c}}\boldsymbol{R}_c)\right)^T \tag{23}$$

$$\boldsymbol{C} = {}^{c}\boldsymbol{n}\,{}^{\hat{c}}\boldsymbol{t}_c^T\left({}^{c}\boldsymbol{n}\,{}^{\hat{c}}\boldsymbol{t}_c^T\right)^T \tag{24}$$

C. Parameterization

By looking at (21), we can isolate the 5 parameters our loss relies on. ${}^{\hat{c}}\boldsymbol{R}_c$ and ${}^{\hat{c}}\boldsymbol{t}_c$ are directly computed from ground truth and estimated poses. We set ${}^{c}\boldsymbol{n} = (0,0,-1)^T$, so that all homographies are induced by planes parallel to the ground truth sensor, as if they faced the camera. This leaves us with 2 parameters, x_{min} and x_{max}, representing the minimum and maximum distances of these planes to the ground truth sensor. We introduce two different ways to set these parameters, inspired by different uses of the loss, and leading to different implementations.

[2]defined as the set between two parallel planes as in [16]

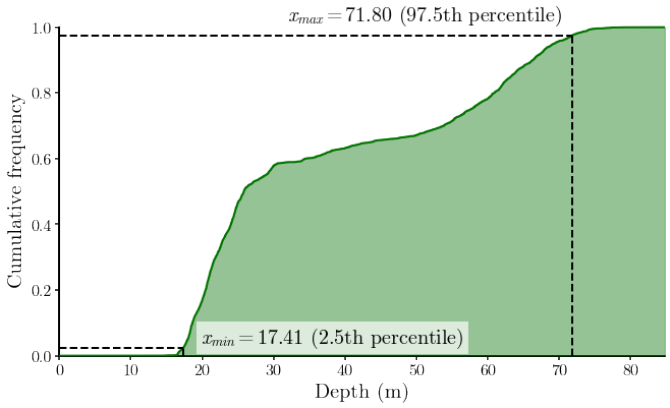


Fig. 5. Cumulative histogram of the depths of the points in the scene. x_{min} and x_{max} depth values can be selected as the $2.5th$ and $97.5th$ percentile.

The first method best approximates the reprojection error, but requires 3D data of the scene. Given this data, the two parameters can be computed for each frame. For every frame, we compute a depth histogram of its 3D observations. We then set its x_{min} and x_{max} parameters as a given percentile of the distribution of the depths (see Fig. 5). We refer to this method as *Local homography* loss function.

The other way of setting these parameters does not require any 3D information of the scene. If no 3D data is available, it is possible to manually set global x_{min} and x_{max} intuitively according to the images in the dataset, by estimating minimum and maximum scene limits. Unlike the aforementioned method, these parameters will be shared by every frame. We refer to this manner of configuring the loss as *Global homography* loss function. Note that if 3D data is available, it is also possible to set global x_{min} and x_{max} from a global depth histogram.

IV. Experiments

We implemented PoseNet [6], Homoscedastic [7], [12], Geometric [7] and MaxError [8] losses. We tested all losses on Cambridge [6] and 7-Scenes [13] datasets with a MobileNetV2 [5] architecture. We compared the results between them and with previously published ones.

A. Datasets

In [17], Brachmann *et al.* show that the performance of a relocalization method on a given dataset is greatly impacted by the method used to build the "ground truth" of this dataset. Very popular methods for estimating the camera poses, and the scene geometry are Structure-from-Motion (SfM) and depth-based SLAM. In our case, some losses might be advantaged by one method or the other. To try to alleviate this issue when benchmarking the different losses, we evaluated them on two datasets whose ground truth poses were estimated using different methods. Cambridge dataset was built using SfM and consists of 6 outdoor scenes of Cambridge city while 7-Scenes was built using depth-based SLAM and provides 7 different

TABLE I
POSES MEAN REPROJECTION DISTANCE ON TRAIN AND TEST SETS, PERCENTAGE OF POSES LOCALIZED WITHIN A GIVEN THRESHOLD

Scene	Existing				Ours	
	PoseNet ($\beta = 500$)	Homoscedastic	MaxError	Geometric reprojection loss	Global homography	Local homography
Great Court	10.5px, **117.7px**, **13%**, **36.4%**	**9.469px**, 148.1px, 7.6%, 26.6%	220.9px, 623.9px, 0.4%, 2.8%	67.46px, 182.9px, 1.1%, 8.4%	92.9px, 235px, 0.7%, 6.2%	143px, 261.3px, 0.4%, 1.1%
King's College	7.18px, 33.93px, 64.4%, 92.7%	5.18px, 24.67px, 60.1%, 92.1%	215.9px, 204.3px, 6.1%, 26.8%	4.53px, **16.16px**, **71.7%**, **94.2%**	4.8px, 23.2px, 61.2%, 92.7%	**4.52px**, 23.07px, 61.5%, 91.8%
Old Hospital	9.02px, 97.5px, 23.6%, 56%	8.19px, 80.63px, 18.7%, 56.6%	98.23px, 177.2px, 9.3%, 34.6%	6.56px, **63.84px**, **28.6%**, **71.4%**	6.76px, 100.2px, 15.9%, 51.6%	**6px**, 92.8px, 23.1%, 61%
Shop Façade	17.19px, 135.4px, 15.5%, **68%**	12.18px, 125px, 14.6%, 49.5%	194.8px, 218.9px, 3.9%, 31.1%	10.91px, **117.4px**, **18.4%**, 56.3%	10.82px, 149px, 11.7%, 47.6%	**10.27px**, 130.6px, 12.6%, 58.3%
St Mary's Church	16.34px, 161.5px, 13.4%, 50.8%	13.08px, 125px, 16.8%, 56.6%	109.1px, 260.1px, 2.6%, 20.6%	33.51px, **104.6px**, 13.6%, 52.6%	10.94px, 114.7px, 17.0%, 56.2%	**10.15px**, 108.5px, **18.1%**, **57.5%**
Street	36.29px, 790.2px, 0.4%, 2.1%	32.04px, 758.2px, 0.2%, 1.8%	224.4px, 768.5px, 0.1%, 0.4%	390.5px, **505.2px**, 0%, 1.1%	25.02px, 733.6px, 0.5%, 2.4%	**23.32px**, 683px, **0.6%**, **3.5%**
chess	4.36px, 40.49px, 78.4%, 91.5%	1.92px, 30.73px, 80.4%, 94%	2.43px, 34.76px, 80.8%, 96.4%	1.51px, **26.46px**, 80.9%, 95.1%	1.94px, 30.27px, 82%, **96.5%**	**1.5px**, 28.85px, **82.3%**, 96.2%
fire	4.59px, 80.03px, 34.8%, 61.3%	2.09px, 88.35px, 32.5%, 66%	2.23px, 86.92px, **35.4%**, **68.8%**	1.67px, 89.16px, 30.5%, 65.1%	2px, 83.89px, 32.3%, 66.3%	**1.59px**, **79.05px**, 31.8%, 64.3%
heads	7.56px, 96.01px, 32.1%, 57%	3.34px, 90.13px, 30.8%, 53%	3.53px, 71.75px, 31.6%, **62.4%**	2.46px, 75.13px, 31.7%, 55%	2.93px, 76.22px, 33.3%, 59.2%	**2.34px**, **69.85px**, **33.9%**, 58.3%
office	4.97px, 55.03px, 60%, 90.1%	2.65px, 59.25px, 54.7%, 86.6%	3.06px, 59.68px, **62.3%**, 87.6%	2.22px, 50.3px, 59.5%, 90.3%	2.31px, 55.2px, 57.3%, **90.7%**	**2.05px**, **46px**, **62.3%**, 86.1%
pumpkin	3.38px, 121.1px, **59%**, 74.4%	1.66px, 80.3px, 51.7%, 73.2%	2px, 89.6px, 51.9%, **77.2%**	1.35px, 87.5px, 50.9%, 71.9%	1.63px, 97.3px, 50.1%, 71.5%	**1.2px**, **69.1px**, 53.5%, 73.9%
redkitchen	4.59px, 70.71px, 45.1%, 74.6%	2.24px, 89.24px, 45.8%, 73.8%	3px, 79.38px, 54.9%, 79.8%	**1.9px**, 83.71px, 48.4%, 77%	2.85px, 78.69px, 50.5%, 75.4%	2.11px, **66.31px**, **57.1%**, **81.7%**
stairs	4.24px, 123.4px, 12.7%, 36.6%	1.87px, 127px, 13.5%, 58.6%	2.26px, 133.1px, **22.2%**, **59%**	1.65px, 153.9px, 4%, 28.1%	1.75px, 144.9px, 18.4%, 57.9%	**1.49px**, **120.9px**, 17%, 55.8%

TABLE II
COMPARING OUR IMPLEMENTATION WITH [7] ON KING'S COLLEGE

	PoseNet $\beta = 500$	Homoscedastic	Geometric
Kendall *et al.* [7]	1.66m, 4.86°	0.99m, 1.06°	0.88m, 1.04°
Ours	0.76m, 0.90°	0.87m, 1.15°	0.64m, 0.89°

indoor scenes. All scenes in both datasets are visited several times, and train and test sequences consist of different visits.

B. Network

Kendall *et al.* used GoogLeNet [4] as a regression backbone. They then replaced the classification head with 2 dense layers with respective feature sizes of 2048 and 7 (3 for translation and 4 for the quaternion). In this paper, we use the MobileNetV2 [5] backbone provided by PyTorch and proceed to the same replacement. We chose the MobileNetV2 backbone for its versatility. Similarly to Kendall *et al.*, we load weights from MobileNetV2 pre-trained on ImageNet which are available from the PyTorch Hub. Note that we normalize our input images, as suggested by PyTorch. However, we do not crop our resized image to stay consistent with [7]. In PoseNet [6], the network is reportedly trained on random crops of the resized images. We tested this approach and found that it greatly deteriorates the results. We suggest that by applying a random crop to the image, it artificially moves the optical center of the camera, which might not be ideal when trying to estimate its pose. We deliberately chose to test our loss function on a simple and unique end-to-end network rather than on a more complete pipeline like DSAC [8] or DSAC++ [9]. Our aim is to provide an alternative to the existing pose regression loss functions, not to provide a complete pose regression system. Therefore, this study compares the loss functions on a single regression model, to easily compare and reproduce results. The use of this loss in the full pipeline of existing state-of-the-art relocalization systems remains in the scope of future work.

C. Losses specifications

We train the model in a common mini-batch setting, where each optimization step is based on the average loss over all camera poses of the batch. For the PoseNet loss, we fix $\beta = 500$ to compare our results with those reported in [7]. For the Homoscedastic loss, we initialize its parameters as suggested by Kendall *et al.*, $\hat{s}_t = 0.0$ and $\hat{s}_q = -3.0$. As discussed earlier, when implementing the Geometric reprojection loss, we clip the reprojection error of each point at 100 to avoid divergence. As for MaxError loss, estimated quaternions always ended up converging towards the null vector on the Cambridge dataset. We fixed this by adding a regularization term, $MSE\left(\|q\|_2, 1\right)$, to the final loss. Nevertheless, we only reached $\sim 1.2k - 3.5k$ epochs before the loss diverges. For our homographic losses, we chose x_{min} and x_{max} as a percentile of the depth distribution of 3D points in the scene, as discussed in section (III-C). x_{min} and x_{max} were set to the $2.5th$ and $97.5th$ percentiles, respectively.

D. Training

We train all models with an Adam optimizer [18], [19] with a learning rate of 10^{-4}. We train for 5k epochs with a batch size of 64, dropping the last batch if smaller. During our experiments, we found that for the homography losses, an Adam's epsilon of 10^{-14} instead of the default 10^{-8} leads to better results. This is because at the end of the optimization, our losses reach very low values $\sim 10^{-4}$. For the Geometric reprojection loss, we initialize the network by training for 500 epochs on the Homoscedastic loss.

V. RESULTS AND DISCUSSION

In Table II we report median translation and rotation errors on different losses and compare them with previously published ones. We show that our implementation's results are consistent with PoseNet [6], [7]. Small differences may be due to the use of MobileNetV2 instead of GoogLeNet as a backbone. PoseNet differences can be explained by the cropping done in the original implementation [6] which is no longer performed in more recent studies [7].

Table I presents the results we obtained on Cambridge and 7-Scenes datasets. We report two different types of metrics. The first criterion is the mean reprojection distance. For a given camera, we select the 3D points that it observes as given by our ground truth. We then compute the L2 norm between the projection of these points on the ground truth and the estimated camera view. We clip the reprojection distance of each point at 1000px to reduce the impact of outliers on the metric. Finally, we compute the mean of all these distances. The second criterion is the percentage of poses localized within a given threshold in meters and in degrees. For each loss

function and for each scene, we report i) the mean reprojection distance on the train set, ii) the mean reprojection distance on the test set, iii) & iv) the second criterion with different threshold values on the test set. We use different threshold values for Cambridge and 7-Scenes datasets because the ratio between average translation and rotation errors is not the same. For Cambridge, thresholds are 2m/2° and 3m/5°. For 7-Scenes, thresholds are 0.25m/10° and 0.5m/15°.

We argue that the different metrics may be more favorable to different losses. When evaluating poses independently of what the camera observes, the percentage of poses localized within a given threshold might be a better indicator of performance than the mean reprojection distance. On the other hand, when evaluating poses based on their shared observations, the mean reprojection distance is a better indicator of whether images captured from estimated and ground truth cameras poses result in the same view. Both evaluation methods are more or less important depending on the application, *e.g.*, for ego-motion we might need to perform best on the percentage criterion, while in augmented reality we may seek to improve the mean reprojection distance. This is why, in addition to the percentage of poses localized within a given threshold classically reported in previous work [10], [11], [20], we also report mean square reprojection distance on the train and test sets. We choose to report this error on the train dataset because this way we directly monitor how the losses optimize it.

Overall, we notice that the Geometric reprojection loss performs better on the Cambridge dataset, while our homography loss shows the best results on the 7-Scenes dataset. As argued by Brachmann *et al.* [17], this could be explained by the relation between the method used to estimate the "ground truth" poses and the cost minimized by the loss function. As Cambridge ground truth poses were obtained using SfM, the Geometric reprojection loss minimizes the same quantities using the exact same data as the ones used to estimate the ground truth. Conversely, 7-Scenes poses were obtained using depth-based SLAM. While our loss does not minimize the same quantities as depth-based SLAM, it might greatly benefit from the access to dense depth maps since its parameters can be directly deduced from them. Interestingly, the proposed homography-based loss shows the best results with regard to the reprojection distance on the training set, which is by definition what the Geometric reprojection loss minimizes. This could be explained by the improved stability and convexity of our proposed losses in a mini-batch stochastic gradient descent training setting.

VI. Conclusion

We introduced a new loss function for camera pose regression, which is based on the integration of homographies' virtual planes between the minimum and maximum scene distances to the sensor. It relies on two physically interpretable parameters that can either be tuned manually or computed from 3D data. Moreover, it requires no prior initialization to converge. The obtained results show that it optimizes best the mean reprojection distance on the train set than any other loss. Depending on the application, it provides a competitive drop-in alternative to existing pose regression losses. Our loss might also be a good replacement to the Geometric reprojection loss if 3D data is not available or if the target application needs to regress a pose with regard to the camera observations without relying on specific 3D points, *e.g.*, learning features. Future work will concentrate in testing the proposed loss in more complete relocalization pipelines like DSAC [8], DSAC++ [9] or PixLoc [11].

Appendix A

For ease of reading, we will use the following notations in the appendix: $\boldsymbol{t} = {}^{\hat{c}}\boldsymbol{t}_c$, $\boldsymbol{R} = {}^{\hat{c}}\boldsymbol{R}_c$, $\boldsymbol{H} = {}^{\hat{c}}\boldsymbol{H}_c$ and $\boldsymbol{n} = {}^{c}\boldsymbol{n}$.

In this appendix, we demonstrate that our loss function only reaches its minimum when poses are superimposed. That is, when $\boldsymbol{R} = \boldsymbol{I}$ and $\boldsymbol{t} = \boldsymbol{0}_3$. Note that our loss considers planes parallel to our sensor image plane. Therefore, our proof is only valid for $n = (0, 0, -1)^T$.

Given any 3×3 matrix $\boldsymbol{M} \in \mathcal{R}^{3\times3}$, its squared Frobenius norm is

$$\|\boldsymbol{M}\|_F^2 = \sum_{i=1}^{3}\sum_{j=1}^{3} m_{ij}^2 \tag{25}$$

where m_{ij} is the ith row, jth column element of $\boldsymbol{M}$. Because the squared Frobenius norm of such a matrix is the sum of the square of all its elements, it is clear that $\|\boldsymbol{I} - \boldsymbol{H}\|_F^2$ can only be positive. It is also clear that its integration over a positive interval can only be positive and therefore, $\mathcal{L}_H \geq 0$ (20). We also know that the homography between two superimposed views is the identity matrix

$$[\boldsymbol{R}, \boldsymbol{t}] = [\boldsymbol{I}, \boldsymbol{0}_3] \Rightarrow \boldsymbol{H} = \boldsymbol{I} \Rightarrow \mathcal{L}_H = 0 \tag{26}$$

Because $\mathcal{L}_H \geq 0$, that means that its minimum is 0.

From (25), we can deduce that

$$\mathcal{L}_H = 0 \Leftrightarrow \boldsymbol{H} = \boldsymbol{I} \tag{27}$$

which means that our loss reaches its minimum only when the homography is the identity matrix. It remains to prove that $\boldsymbol{H} = \boldsymbol{I} \Rightarrow [\boldsymbol{R}, \boldsymbol{t}] = [\boldsymbol{I}, \boldsymbol{0}_3]$.

By injecting (5) in (27)

$$\boldsymbol{R} = \boldsymbol{I} + \frac{\boldsymbol{t}\boldsymbol{n}^T}{d} \tag{28}$$

And because $\boldsymbol{R} \in SO(3)$ it has the property $\boldsymbol{R}\boldsymbol{R}^T = \boldsymbol{I}$. We can use this property to constrain $\boldsymbol{t}$.

$$\left(\boldsymbol{I} + \frac{\boldsymbol{t}\boldsymbol{n}^T}{d}\right)\left(\boldsymbol{I} + \frac{\boldsymbol{t}\boldsymbol{n}^T}{d}\right)^T = \boldsymbol{I} \tag{29}$$

$$\Leftrightarrow \boldsymbol{t}\boldsymbol{n}^T + \boldsymbol{n}\boldsymbol{t}^T + \frac{\boldsymbol{t}\boldsymbol{n}^T\boldsymbol{n}\boldsymbol{t}^T}{d} = \boldsymbol{0}_{3\times3} \tag{30}$$

We fix $\boldsymbol{n} = (0, 0, -1)^T$ and note $\boldsymbol{t} = (t_x, t_y, t_z)^T$. We can decompose (30)

$$\boldsymbol{t}\boldsymbol{n}^T = \begin{pmatrix} 0 & 0 & -t_x \\ 0 & 0 & -t_y \\ 0 & 0 & -t_z \end{pmatrix} \tag{31}$$

$$\boldsymbol{n}\boldsymbol{t}^T = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ -t_x & -t_y & -t_z \end{pmatrix} \tag{32}$$

$$\boldsymbol{t}\boldsymbol{n}^T\boldsymbol{n}\boldsymbol{t}^T = \begin{pmatrix} t_x^2 & t_x t_y & t_x t_z \\ t_y t_x & t_y^2 & t_y t_z \\ t_z t_x & t_z t_y & t_z^2 \end{pmatrix} \tag{33}$$

From (30), (31), (32) and (33) we deduce

$$t_x = 0 \tag{34}$$

$$t_y = 0 \tag{35}$$

And two possibilities for t_z

$$\begin{cases} t_z = 0 \\ t_z = 2d \end{cases} \tag{36}$$

We can further constrain $\boldsymbol{t}$ by using another property of $SO(3)$, that is, $\det(\boldsymbol{R}) = 1$. From (28) and (31) with $t_x = 0$, $t_y = 0$ and $d \neq 0$

$$\det(\boldsymbol{R}) = \det\left(\begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 - \frac{t_z}{d} \end{pmatrix}\right) = 1 - \frac{t_z}{d} \tag{37}$$

$$\Leftrightarrow 1 - \frac{t_z}{d} = 1 \Leftrightarrow t_z = 0 \tag{38}$$

We have just shown that $\boldsymbol{H} = \boldsymbol{I} \Rightarrow \boldsymbol{t} = \boldsymbol{0}_3$. From (28) we find that

$$\boldsymbol{R} = \boldsymbol{I} + \frac{\boldsymbol{0}_3\boldsymbol{n}^T}{d} = \boldsymbol{I} \tag{39}$$

Which means that

$$\boldsymbol{H} = \boldsymbol{I} \Rightarrow [\boldsymbol{R}, \boldsymbol{t}] = [\boldsymbol{I}, \boldsymbol{0}_3] \tag{40}$$

By putting (26), (27) and (40) together we find

$$\mathcal{L}_H = 0 \Leftrightarrow [\boldsymbol{R}, \boldsymbol{t}] = [\boldsymbol{I}, \boldsymbol{0}_3] \tag{41}$$

Our loss minimum is only reached when poses are superimposed.

REFERENCES

[1] N. Piasco, D. Sidibé, C. Demonceaux, and V. Gouet-Brunet, "A survey on visual-based localization: On the benefit of heterogeneous data," *Pattern Recognition*, vol. 74, pp. 90–109, 2018.

[2] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[3] T. Sattler, B. Leibe, and L. Kobbelt, "Efficient & effective prioritized matching for large-scale image-based localization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 9, pp. 1744–1756, 2017.

[4] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[6] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[7] A. Kendall and R. Cipolla, "Geometric loss functions for camera pose regression with deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[8] E. Brachmann, A. Krull, S. Nowozin, J. Shotton, F. Michel, S. Gumhold, and C. Rother, "Dsac - differentiable ransac for camera localization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] E. Brachmann and C. Rother, "Learning less is more - 6d camera localization via 3d surface regression," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] P.-E. Sarlin, C. Cadena, R. Siegwart, and M. Dymczyk, "From coarse to fine: Robust hierarchical localization at large scale," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[11] P.-E. Sarlin, A. Unagar, M. Larsson, H. Germain, C. Toft, V. Larsson, M. Pollefeys, V. Lepetit, L. Hammarstrand, F. Kahl, and T. Sattler, "Back to the feature: Learning robust camera localization from pixels to pose," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2021, pp. 3247–3257.

[12] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[13] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene coordinate regression forests for camera relocalization in rgb-d images," in *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2013.

[14] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[15] P. Réfrégier, *Noise theory and application to physics: from fluctuations to information*. Springer Science & Business Media, 2004.

[16] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[17] E. Brachmann, M. Humenberger, C. Rother, and T. Sattler, "On the Limits of Pseudo Ground Truth in Visual Camera Re-Localization," in *International Conference on Computer Vision (ICCV)*, 2021.

[18] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[19] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[20] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.