Learning Accurate Dense Correspondences and When to Trust Them

Learning Accurate Dense Correspondences and When to Trust Them

Prune Truong Martin Danelljan Luc Van Gool Radu Timofte
Computer Vision Lab, ETH Zurich, Switzerland
{prune.truong, martin.danelljan, vangool, radu.timofte}@vision.ee.ethz.ch

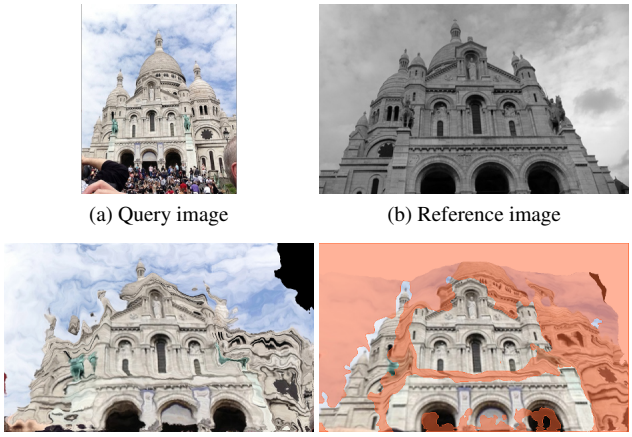
Abstract

Establishing dense correspondences between a pair of images is an important and general problem. However, dense flow estimation is often inaccurate in the case of large displacements or homogeneous regions. For most applications and down-stream tasks, such as pose estimation, image manipulation, or 3D reconstruction, it is crucial to know when and where *to trust the estimated matches.*

In this work, we aim to estimate a dense flow field relating two images, coupled with a robust pixel-wise confidence map indicating the reliability and accuracy of the prediction. We develop a flexible probabilistic approach that jointly learns the flow prediction and its uncertainty. In particular, we parametrize the predictive distribution as a constrained mixture model, ensuring better modelling of both accurate flow predictions and outliers. Moreover, we develop an architecture and training strategy tailored for robust and generalizable *uncertainty prediction in the context of self-supervised training. Our approach obtains state-of-the-art results on multiple challenging geometric matching and optical flow datasets. We further validate the usefulness of our probabilistic confidence estimation for the task of pose estimation. Code and models are available at* https://github.com/PruneTruong/PDCNet.

1. Introduction

Finding pixel-wise correspondences between pairs of images is a fundamental computer vision problem with numerous important applications, including dense 3D reconstruction [51], video analysis [44, 57], image registration [55, 62], image manipulation [15, 37], and texture or style transfer [27, 35]. Dense correspondence estimation has most commonly been addressed in the context of optical flow [2, 17, 22, 60], where the image pairs represent consecutive frames in a video. While these methods excel in the case of small appearance changes and limited displacements, they cannot cope with the challenges posed by the more general geometric matching task. In geometric matching, the images can stem from radically different views of the same scene, often captured by different cameras and at different occasions. This leads to large displacements and significant appearance transformations between the frames.



(a) Query image (b) Reference image (c) Baseline (d) **PDC-Net** (Ours)

Figure 1. Estimating dense correspondences between the query (a) and the reference (b) image. The query is warped according to the resulting flows (c)-(d). The baseline (c) does not estimate an uncertainty map and is therefore unable to filter the inaccurate flows at *e.g.* occluded and homogeneous regions. In contrast, our PDC-Net (d) not only estimates accurate correspondences, but also *when to trust them*. It predicts a robust uncertainty map that identifies accurate matches and excludes incorrect and unmatched pixels (red).

In contrast to optical flow, the more general dense correspondence problem has received much less attention [40, 48, 53, 64]. Dense flow estimation is prone to errors in the presence of large displacements, appearance changes, or homogeneous regions. It is also ill-defined in case of occlusions or in *e.g.* sky, where predictions are bound to be inaccurate (Fig. 1c). For geometric matching applications, it is thus crucial to know *when and where to trust* the estimated correspondences. For instance, pose estimation, 3D reconstruction, and image-based localization require a set of highly robust and accurate matches as input. The predicted dense flow field must therefore be paired with a *robust* confidence estimate (Fig. 1d). Uncertainty estimation is also indispensable for safety-critical tasks, such as autonomous driving and medical imaging. In this work, we set out to

expand the application domain of dense correspondence estimation by learning to predict reliable confidence values.

We propose the Probabilistic Dense Correspondence Network (PDC-Net), for joint learning of dense flow and uncertainty estimation, applicable even for extreme appearance and view-point changes. Our model predicts the conditional probability density of the flow, parametrized as a constrained mixture model. However, learning *reliable and generalizable* uncertainties without densely annotated real-world training data is a highly challenging problem. Standard self-supervised techniques [40, 46, 64] do not faithfully model real motion patterns, appearance changes, and occlusions. We tackle this challenge by introducing a carefully designed architecture and improved self-supervision to ensure robust and generalizable uncertainty predictions.

Contributions: Our main contributions are as follows. **(i)** We introduce a *constrained mixture model* of the predictive distribution, allowing the network to flexibly model both accurate predictions and outliers with large errors. **(ii)** We propose an architecture for predicting the parameters of our predictive distribution, that carefully exploits the information encoded in the correlation volume, to achieve generalizable uncertainties. **(iii)** We improve upon self-supervised data generation pipelines to ensure more robust uncertainty estimation. **(iv)** We utilize our uncertainty measure to address extreme view-point changes by iteratively refining the prediction. **(v)** We perform extensive experiments on a variety of datasets and tasks. In particular, our approach sets a new state-of-the-art on the Megadepth geometric matching dataset [34], on the KITTI-2015 training set [13], and outperforms previous dense methods for pose estimation on the YFCC100M dataset [61]. Moreover, without further post-processing, our confident dense matches can be directly input to 3D reconstruction pipelines [51], as shown in Fig. 2.

2. Related work

Confidence estimation in geometric matching: Only very few works have explored confidence estimation in the context of dense geometric or semantic matching. Novotny *et al*. [42] estimate the reliability of their trained descriptors by using a self-supervised probabilistic matching loss for the task of semantic matching. A few approaches [19, 47, 48] represent the final correspondences as a 4D correspondence volume, thus inherently encoding a confidence score for each tentative match. However, these approaches are usually restricted to low-resolution images, thus hindering accuracy. Moreover, generating one final confidence value for each match is highly non-trivial since multiple high-scoring alternatives often co-occur. Similarly, Wiles *et al*. [68] learn dense descriptors conditioned on an image pair, along with their distinctiveness score. However, the latter is trained with hand-crafted heuristics, while we instead do not make assumption on what the confidence score should be, and learn it directly from the data with a single unified loss. In DGC-Net, Melekhov *et al*. [40] predict both dense correspondence and matchability maps relating image pairs. However, their matchability map is only trained to identify out-of-view pixels rather than to reflect the actual reliability of the matches. Recently, Shen *et al*. [53] proposed RANSAC-Flow, a two-stage image alignment method, which also outputs a matchability map. It performs coarse alignment with multiple homographies using RANSAC on off-the-shelf deep features, followed by fine alignment. In contrast, we propose a unified network that estimates probabilistic uncertainties.

Figure 2. 3D reconstruction of Aachen [49] using the dense correspondences and uncertainties predicted by PDC-Net.

Uncertainty estimation in optical flow: While optical-flow has been a long-standing subject of active research, only a handful of methods provide uncertainty estimates. A few approaches [1, 3, 29, 30, 32] treat the uncertainty estimation as a post-processing step. Recently, some works propose probabilistic frameworks for joint optical flow and uncertainty prediction instead. They either estimate the model uncertainty [11, 24], termed epistemic uncertainty [26], or focus on the uncertainty from the observation noise, referred to as aleatoric uncertainty [26]. Following recent works [12, 71], we focus on aleatoric uncertainty and how to train a generalizable uncertainty estimate in the context of self-supervised training. Wannenwetsch *et al*. [67] propose ProbFlow, a probabilistic approach applicable to energy-based optical flow algorithms [3, 45, 58]. Gast *et al*. [12] propose probabilistic output layers that require only minimal changes to existing networks. Yin *et al*. [71] introduce HD^{3}F, a method to estimate uncertainty locally at multiple spatial scales and aggregate the results. While these approaches are carefully designed for optical flow data and restricted to small displacements, we consider the more general setting of estimating reliable confidence values for dense geometric matching, applicable to *e.g*. pose-estimation and 3D reconstruction. This brings additional challenges, including coping with significant appearance changes and large geometric transformations.

3. Our Approach: PDC-Net

We introduce PDC-Net, a method for estimating the dense flow field relating two images, coupled with a robust pixel-wise confidence map. The later indicates the reliability and accuracy of the flow prediction, which is necessary for pose estimation, image manipulation, and 3D-reconstruction tasks.

3.1. Probabilistic Flow Regression

We formulate dense correspondence estimation with a probabilistic model, which provides a framework for learning both the flow and its confidence in a unified formulation. For a given image pair $X = (I^q, I^r)$ of spatial size $H \times W$, the aim of dense matching is to estimate a flow field $Y \in \mathbb{R}^{H\times W\times 2}$ relating the reference I^r to the query I^q. Most learning-based methods address this problem by training a network F with parameters θ that directly predicts the flow as $Y = F(X;\theta)$. However, this does not provide any information about the confidence of the prediction.

Instead of generating a single flow prediction Y, our goal is to learn the conditional probability density $p(Y|X;\theta)$ of a flow Y given the input X. This is generally achieved by letting a network predict the parameters $\Phi(X;\theta)$ of a family of distributions $p(Y|X;\theta) = p(Y|\Phi(X;\theta)) = \prod_{ij} p(y_{ij}|\varphi_{ij}(X;\theta))$. To ensure a tractable estimation, conditional independence of the predictions at different spatial locations (i,j) is generally assumed. We use $y_{ij} \in \mathbb{R}^2$ and $\varphi_{ij} \in \mathbb{R}^n$ to denote the flow Y and predicted parameters Φ respectively, at the spatial location (i,j). In the following, we generally drop the sub-script ij to avoid clutter.

Compared to the direct approach $Y = F(X;\theta)$, the generated parameters $\Phi(X;\theta)$ of the predictive distribution can encode richer information about the flow prediction, including its uncertainty. In probabilistic regression techniques for optical flow [12, 24] and a variety of other tasks [26, 54, 66], this is most commonly performed by predicting the *variance* of the estimate y. In these cases, the predictive density $p(y|\varphi)$ is modeled using Gaussian or Laplace distributions. In the latter case, the density is given by,

$$\mathcal{L}(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\sigma_u^2}} e^{-\sqrt{\frac{2}{\sigma_u^2}}|u-\mu_u|} . \frac{1}{\sqrt{2\sigma_v^2}} e^{-\sqrt{\frac{2}{\sigma_v^2}}|v-\mu_v|} \tag{1}$$

where the components u and v of the flow vector $y = (u,v) \in \mathbb{R}^2$ are modelled with two conditionally independent Laplace distributions. The mean $\mu = [\mu_u, \mu_v]^T \in \mathbb{R}^2$ and variance $\sigma^2 = [\sigma_u^2, \sigma_v^2]^T \in \mathbb{R}^2$ of the distribution $p(y|\varphi) = \mathcal{L}(y|\mu,\sigma^2)$ are predicted by the network as $(\mu,\sigma^2) = \varphi(X;\theta)$ at every spatial location.

3.2. Constrained Mixture Model Prediction

Fundamentally, the goal of probabilistic deep learning is to achieve a predictive model $p(y|X;\theta)$ that coincides with empirical probabilities as well as possible. We can get important insights into this problem by studying the empirical error distribution of a state-of-the-art matching model, in this case GLU-Net [64], as shown in Fig. 3. Errors can be categorized into two populations: inliers (in red) and outliers (in blue). Current probabilistic methods [12, 24, 44] mostly rely on a Laplacian model (1) of $p(y|X;\theta)$. Such a model is effective for correspondences which are easily estimated to be either inliers or outliers with *high certainty*, by predicting a low or high variance respectively. However, often the network is not certain whether a match is an inlier or outlier. A single Laplace can only predict an intermediate variance, which does not faithfully represent the more complicated uncertainty pattern in this case.

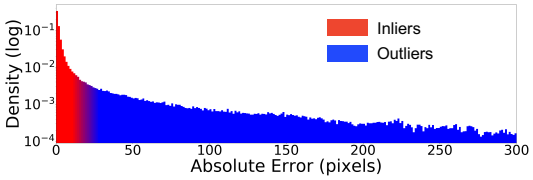


Figure 3. Distribution of errors $|\widehat{y}-y|$ on MegaDepth [34] between the flow $\widehat{y}$ estimated by GLU-Net [64] and the ground-truth y.

Mixture model: To achieve a flexible model capable of fitting more complex distributions, we parametrize $p(y|X;\theta)$ with a mixture model. In general, we consider a distribution consisting of M components,

$$p\left(y|\varphi\right) = \sum_{m=1}^{M} \alpha_m \mathcal{L}\left(y|\mu,\sigma_m^2\right) . \tag{2}$$

While we have here chosen Laplacian components (1), any simple density function can be used. The scalars $\alpha_m \geq 0$ control the weight of each component, satisfying $\sum_{m=1}^{M} \alpha_m = 1$. Note that all components have the same mean μ, which can thus be interpreted as the estimated flow vector, but different variances σ_m^2. The distribution (2) is therefore unimodal, but can capture more complex uncertainty patterns. In particular, it allows to predict the probability of inlier (red) and outlier (blue) matches (Fig. 3), each modeled by separate Laplace components.

Mixture constraints: In general, we now consider a network Φ that, for each pixel location, predicts the mean flow μ along with the variance σ_m^2 and weight α_m of each component, as $\left(\mu, (\alpha_m)_{m=1}^M, (\sigma_m^2)_{m=1}^M\right) = \varphi(X;\theta)$. However, a potential issue when predicting the parameters of a mixture model is its permutation invariance. That is, the predicted distribution (2) is unchanged even if we change the order of the individual components. This can cause confusion in the learning, since the network first needs to *decide* what each component should model before estimating the individual weights α_m and variances σ_m^2.

We propose a model that breaks the permutation invariance of the mixture (2), which simplifies the learning and

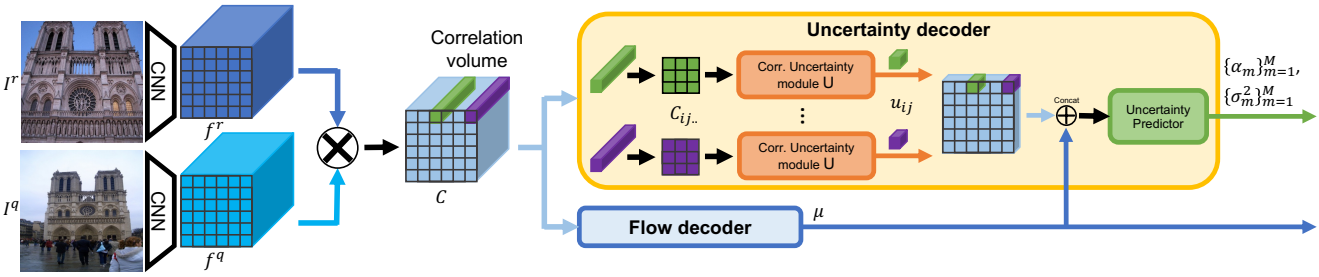


Figure 4. The proposed architecture for flow and uncertainty estimation. The correlation uncertainty module U_θ independently processes each 2D-slice $C_{ij\cdot\cdot}$ of the correlation volume. Its output is combined with the estimated mean flow μ to predict the weight $\{\alpha_m\}_1^M$ and variance $\{\sigma_m^2\}_1^M$ parameters of our constrained mixture model (2)-(4).

greatly improves the robustness of the estimated uncertainties. In essence, each component m is tasked with modeling a specified range of variances σ_m^2. We achieve this by constraining the mixture (2) as,

$$0 < \beta_1^- \leq \sigma_1^2 \leq \beta_1^+ \leq \beta_2^- \leq \sigma_2^2 \leq \ldots \leq \sigma_M^2 \leq \beta_M^+ \quad (3)$$

For simplicity, we here assume a single variance parameter σ_m^2 for both the u and v directions in (1). The constants β_m^-, β_m^+ specify the range of variances σ_m^2. Intuitively, each component is thus responsible for a different range of uncertainties, roughly corresponding to different regions in the error distribution in Fig. 3. In particular, component $m = 1$ accounts for the most accurate predictions, while component $m = M$ models the largest errors and outliers. To enforce the constraint (3), we first predict an unconstrained value $h_m \in \mathbb{R}$, which is then mapped to the given range as,

$$\sigma_m^2 = \beta_m^- + (\beta_m^+ - \beta_m^-)\,\text{Sigmoid}(h_m). \quad (4)$$

The constraint values β_m^+, β_m^- can either be treated as hyper-parameters or learned end-to-end alongside θ.

Lastly, we emphasize an interesting interpretation of our constrained mixture formulation (2)-(3). Note that the predicted weights α_m, in practice obtained through a final SoftMax layer, represent the probabilities of each component m. Our network therefore effectively *classifies* the flow prediction at each pixel into the separate uncertainty intervals (3). We visualize in Fig. 5 the predictive log-distribution with $M = 2$ for three cases. The red and blue matches are with certainty predicted as inlier and outlier respectively, thus requiring only a single active component. In ambiguous cases (green), our mixture model (2)-(3) predicts the probability of inlier vs. outlier, giving a better fit compared to a single-component alternative. As detailed next, our network learns this ability without any extra supervision.

Training objective: As customary in probabilistic regression [7, 12, 14, 24, 26, 54, 65, 66], we train our method using the negative log-likelihood as the only objective. For one input image pair $X = (I^q, I^r)$ and corresponding ground-truth flow Y, the objective is given by

$$-\log p\big(Y|\Phi(X;\theta)\big) = -\sum_{ij} \log p\big(y_{ij}|\varphi_{ij}(X;\theta)\big)\,. \quad (5)$$

In Appendix B.1, we provide efficient analytic expressions of the loss (5) for our constrained mixture (2)-(4), that also ensure numerical stability. As detailed in Sec. 4.1, we can train our final model using either a self-supervised strategy where X and Y are generated by artificial warping, using real sparse ground-truth Y, or a combination of both. Next, we present the architecture of our network Φ that predicts the parameters of our constrained mixture model (2).

3.3. Uncertainty Prediction Architecture

Our aim is to predict an uncertainty value that quantifies the *reliability* of a proposed correspondence or flow vector. Crucially, the uncertainty prediction needs to *generalize* well to real scenarios, not seen during training. However, this is particularly challenging in the context of self-supervised training, which relies on synthetically warped images or animated data. Specifically, when trained on simple synthetic motion patterns, such as homography transformations, the network learns to heavily rely on global smoothness assumptions, which do not generalize well to more complex settings. As a result, the network learns to *confidently* interpolate and extrapolate the flow field to regions where no robust match can be found. Due to the significant distribution shift between training and test data, the network thus also infers confident, yet highly erroneous predictions in homogeneous regions on real data. In this section, we address this problem by carefully designing an architecture that greatly limits the risk of the aforementioned issues. Our architecture is visualized in Figure 4.

Current state-of-the-art dense matching architectures rely on feature correlation layers. Features f are extracted at resolution $h \times w$ from a pair of input images, and densely correlated either globally or within a local neighborhood of

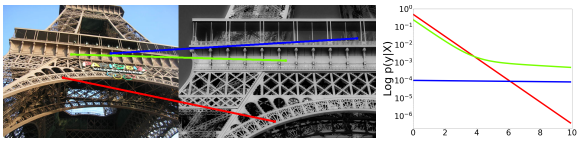
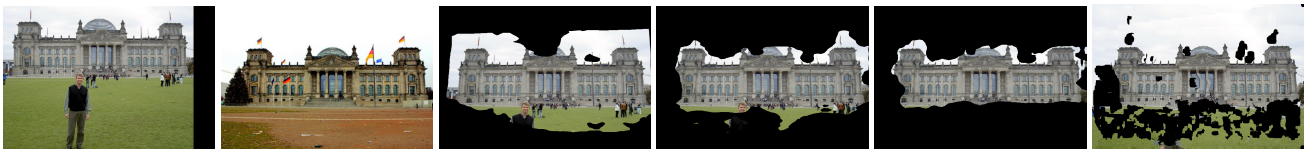


Figure 5. Predictive log-distr. $\log p(y|X)$ (2)-(3) for an inlier (red), outlier (blue), and ambiguous (green) match. Our mixture model faithfully represents the uncertainty also in the latter case.



(a) Query image (b) Reference image (c) Common decoder (d) Our decoder (e) Our decoder and data (f) RANSAC-Flow

Figure 6. Visualization of the estimated uncertainties by masking the warped query image to only show the confident flow predictions. The standard approach (c) uses a common decoder for both flow and uncertainty estimation. It generates overly confident predictions in the sky and grass. The uncertainty estimates are substantially improved in (d), when using the proposed architecture described in Sec. 3.3. Adding the flow perturbations for self-supervised training (Sec. 3.4) further improves the robustness and generalization of the uncertainties (e). For reference, we also visualize the flow and confidence mask (f) predicted by the recent state-of-the-art approach RANSAC-Flow [53].

size d. In the later case, the output correlation volume is best thought of as a 4D tensor $C \in \mathbb{R}^{h\times w\times d\times d}$. Computed as dense scalar products $C_{ijkl} = (f^r_{ij})^{\mathrm{T}} f^q_{i+k,j+l}$, it encodes the deep feature similarity between a location (i,j) in the reference frame I^r and a displaced location $(i+k, j+l)$ in the query I^q. Standard flow architectures process the correlation volume by first vectorizing the last two dimensions, before applying a sequence of convolutional layers over the *reference coordinates* (i,j) in order to predict the final flow.

Correlation uncertainty module: The straightforward strategy for implementing the parameter predictor $\Phi(X;\theta)$ is to simply increase the number of output channels to include all parameters of the predictive distribution. However, this allows the network to rely primarily on the local neighborhood when estimating the flow and confidence at location (i,j), and thus to ignore the actual reliability of the match and appearance information at the specific location. It results in over-smoothed and overly confident predictions, unable to identify ambiguous and unreliable matching regions, such as the sky. This is visualized in Fig. 6c.

We instead design an architecture that assesses the uncertainty at a specific location (i,j), without relying on neighborhood information. We note that the 2D slice $C_{ij\cdot\cdot} \in \mathbb{R}^{d\times d}$ encapsulates rich information about the matching ability of location (i,j), in the form of a confidence map. In particular, it encodes the distinctness, uniqueness, and existence of the correspondence. We therefore create a *correlation uncertainty decoder* U_θ that independently reasons about each correlation slice as $U_\theta(C_{ij\cdot\cdot})$. In contrast to standard decoders, the convolutions are therefore applied over *the displacement dimensions* (k,l). Efficient parallel implementation is ensured by moving the first two dimensions of C to the batch dimension using a simple tensor reshape. Our strided convolutional layers then gradually decrease the size $d\times d$ of the displacement dimensions (k,l) until a single vector $u_{ij} = U_\theta(C_{ij\cdot\cdot}) \in \mathbb{R}^n$ is achieved for each spatial coordinate (i,j) (see Fig. 4).

Uncertainty predictor: The cost volume does not capture uncertainty arising at motion boundaries, crucial for real data with independently moving objects. We thus additionally integrate predicted flow information in the estimation of its uncertainty. In practise, we concatenate the estimated mean flow μ with the output of the correlation uncertainty module U_θ, and process it with multiple convolution layers. It outputs all parameters of the mixture (2), except for the mean flow μ (see Fig. 4). As shown in Fig. 6d, our uncertainty decoder, comprised of the correlation uncertainty module and the uncertainty predictor, successfully masks out most of the inaccurate and unreliable matching regions.

3.4. Data for Self-supervised Uncertainty

While designing a suitable architecture greatly alleviates the uncertainty generalization issue, the network still tends to rely on global smoothness assumptions and interpolation, especially around object boundaries (see Fig. 6d). While this learned strategy indeed minimizes the Negative Log Likelihood loss (5) on self-supervised training samples, it does not generalize to real image pairs. In this section, we further tackle this problem from the data perspective in the context of self-supervised learning.

We aim at generating less predictable synthetic motion patterns than simple homography transformations, to prevent the network from primarily relying on interpolation. This forces the network to focus on the appearance of the image region in order to predict its motion and uncertainty. Given a base flow $\tilde{Y}$ relating $\tilde{I}^r$ to $\tilde{I}^q$ and representing a simple transformation such as a homography as in prior works [40, 46, 63, 64], we create a residual flow $\epsilon = \sum_i \varepsilon_i$, by adding small local perturbations ε_i. The query image $I^q = \tilde{I}^q$ is left unchanged while the reference I^r is generated by warping $\tilde{I}^r$ according to the residual flow ϵ. The final perturbed flow map Y between I^r and I^q is achieved by composing the base flow $\tilde{Y}$ with the residual flow ϵ.

An important benefit of introducing the perturbations ε is to teach the network to be uncertain in regions where it cannot identify them. Specifically, in homogeneous regions such as the sky, the perturbations do not change the appearance of the reference ($I^r \approx \tilde{I}^r$) and are therefore unnoticed by the network. However, since the perturbations break the global smoothness of the synthetic flow, the flow errors on those pixels will be higher. In order to decrease the loss (5), the network will thus need to estimate a larger uncertainty for these regions. We show the impact of introducing the flow perturbations for self-supervised learning in Fig. 6e.

3.5. Geometric Matching Inference

In real settings with extreme view-point changes, flow estimation is prone to failing. Our confidence estimate can be used to improve the robustness of matching networks to such cases. Particularly, our approach offers the opportunity to perform multi-stage flow estimation on challenging image pairs, without any additional network components.

Confidence value: From the predictive distribution $p(y|\varphi(X;\theta))$, we aim at extracting a single confidence value, encoding the reliability of the corresponding predicted flow vector μ. Previous probabilistic regression methods mostly rely on the variance as a confidence measure [12, 24, 26, 66]. However, we observe that the variance can be sensitive to outliers. Instead, we compute the probability P_R of the true flow being within a radius R of the estimated mean flow μ. This is expressed as,

$$P_R = P(|y-\mu| < R) = \int_{\{y\in\mathbb{R}^2:|y-\mu|<R\}} p(y|\varphi)dy. \quad (6)$$

Compared to the variance, the probability value P_R also provides a more interpretable measure of the uncertainty.

Multi-stage refinement strategy: For extreme view-point changes with large scale or perspective variations, it is particularly difficult to infer the correct motion field in a single network pass. While this is partially alleviated by multi-scale architectures, it remains a major challenge in geometric matching. Our approach allows to split the flow estimation process into two parts, the first estimating a simple transformation, which is then used as initialization to infer the final, more complex transformation.

One of the major benefits of our confidence estimation is the ability to *identify* the set of accurate matches from the densely estimated flow field. After a first forward network pass, these accurate correspondences can be used to estimate a coarse transformation relating the image pair, such as a homography transformation. A second forward pass can then be applied to the coarsely aligned image pair, and the final flow field is constructed as a composition of the fine flow and the homography transform. While previous works also use multi-stage refinement [46, 53], our approach is much simpler, applying the *same* network in both stages and benefiting from the internal confidence estimation.

4. Experimental results

We integrate our approach into a generic pyramidal correspondence network and perform comprehensive experiments on multiple geometric matching and optical flow datasets. We also show that our method can be used for various tasks, including pose estimation and dense 3D reconstruction. Further results, analysis, visualizations and implementation details are provided in the Appendix.

4.1. Implementation Details

We adopt the recent GLU-Net-GOCor [63, 64] as our base architecture. It consists in a four-level pyramidal network operating at two image resolutions and employing a VGG-16 network [5] pre-trained on ImageNet for feature extraction. At each level, we add our uncertainty decoder (Sec. 3.3) and propagate the uncertainty prediction to the next level. We model the probability distribution $p(y|\varphi)$ with a constrained mixture (Sec. 3.2) with $M = 2$ Laplace components, where the first is fixed to $\sigma_1^2 = \beta_1^- = \beta_1^+ = 1$ to represent the very accurate predictions, while the second models larger errors and outliers, as $2 = \beta_2^- \leq \sigma_2^2 \leq \beta_2^+$, where β_2^+ is set to the square of the training image size.

Our training consists of two stages. First, we follow the self-supervised training procedure of [63, 64]. Random homography transformations are applied to images compiled from different sources to ensure diversity. For better compatibility with real 3D scenes and moving objects, the data is further augmented with random independently moving objects from the COCO [36] dataset. We further apply our perturbation strategy described in Sec. 3.4. In the second stage, we extend the self-supervised data with real image pairs with sparse ground-truth correspondences from the MegaDepth dataset [34]. We additionally finetune the backbone feature extractor. For fair comparison, we also train a version of GLU-Net-GOCor, denoted GLU-Net-GOCor*, using the same settings and data.

For datasets with very extreme geometric transformations, we also report using a multi-scale strategy. In particular, we extend our two-stage refinement approach (Sec. 3.5) by resizing the reference image to different resolutions. The resulting image pairs are passed through the network and we fit a homography for each pair, using our predicted flow and uncertainty map. We select the homography with the highest percentage of inliers, and scale it to the images original resolutions. The original image pair is then coarsely aligned and from there we follow the same procedure, as explained in Sec. 3.5. We refer to this option as Multi Scale (MS).

4.2. Geometric Correspondences and Flow

We first evaluate our PDC-Net in terms of the quality of the predicted flow field.

Datasets and metrics: We evaluate on standard datasets with sparse ground-truth, namely the **RobotCar** [33, 39],

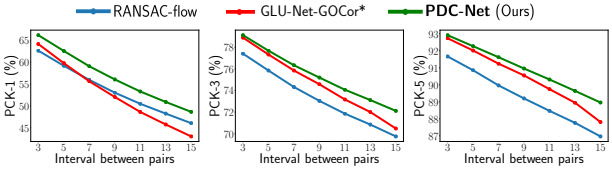


Figure 7. Results on ETH3D [52]. PCK-1 (left), PCK-3 (center) and PCK-5 (right) are plotted w.r.t. the inter-frame interval length.

	MegaDepth			RobotCar		
	PCK-1	PCK-3	PCK-5	PCK-1	PCK-3	PCK-5
SIFT-Flow [37]	8.70	12.19	13.30	1.12	8.13	16.45
NCNet [48]	1.98	14.47	32.80	0.81	7.13	16.93
DGC-Net [40]	3.55	20.33	32.28	1.19	9.35	20.17
GLU-Net [64]	21.58	52.18	61.78	2.30	17.15	33.87
GLU-Net-GOCor [63]	37.28	61.18	68.08	2.31	17.62	35.18
RANSAC-Flow (MS) [53]	53.47	83.45	86.81	2.10	16.07	31.66
GLU-Net-GOCor*	57.77	78.61	82.24	2.33	17.21	33.67
PDC-Net	70.75	86.51	88.00	2.54	**18.97**	**36.37**
PDC-Net (MS)	**71.81**	**89.36**	**91.18**	**2.58**	18.87	36.19

Table 1. PCK (%) results on sparse correspondences of the MegaDepth [34] and RobotCar [33, 39] datasets.

MegaDepth [34] and **ETH3D** [52] datasets. RobotCar depicts outdoor road scenes, taken under different weather and lighting conditions. Images are particularly challenging due to their numerous textureless regions. MegaDepth images show extreme view-point and appearance variations. Finally, ETH3D represents indoor and outdoor scenes captured from a moving hand-held camera. For RobotCar and MegaDepth, we evaluate on the correspondences provided by [53], which includes approximately 340M and 367K ground-truth matches respectively. For ETH3D, we follow the protocol of [64], sampling image pairs at different intervals to analyze varying magnitude of geometric transformations, resulting in 600K to 1100K matches per interval. In line with [53], we employ the Percentage of Correct Keypoints at a given pixel threshold T (PCK-T) as metric.

Results: In Tab. 1 we report results on MegaDepth and RobotCar. Our method PDC-Net outperforms all previous works by a large margin at all PCK thresholds. In particular, our approach is significantly more accurate and robust than the very recent RANSAC-Flow, which utilizes an extensive multi-scale (MS) search. Interestingly, our uncertainty-aware probabilistic approach also outperforms the baseline GLU-Net-GOCor* in pure flow accuracy. This clearly demonstrates the advantages of casting the flow estimation as a probabilistic regression problem, advantages which are not limited to uncertainty estimation. It also substantially benefits the accuracy of the flow itself through a more flexible loss formulation. In Fig. 7, we plot the PCKs on ETH3D. Our approach is consistently better than RANSAC-Flow and GLU-Net-GOCor* for all intervals.

Generalization to optical flow: We additionally show that our approach generalizes well to accurate estimation of optical flow, even though it is trained for the very different task of geometric matching. We use the established **KITTI** dataset [13], and evaluate according to the standard metrics, namely AEPE and Fl. Since we do not fine-tune on KITTI, we show results on the training splits in Tab. 2. Our approach outperforms all previous generic matching methods (upper part) by a large margin in terms of both Fl and AEPE. Surprisingly, PDC-Net also obtains better results than all optical flow methods (bottom part), even outperforming the recent RAFT [60] in Fl metric on KITTI-2015.

	KITTI-2012		KITTI-2015	
	AEPE ↓	Fl (%) ↓	AEPE ↓	Fl (%) ↓
DGC-Net [40]	8.50	32.28	14.97	50.98
GLU-Net [64]	3.14	19.76	7.49	33.83
GLU-Net-GOCor [63]	2.68	15.43	6.68	27.57
RANSAC-Flow [53]	-	-	12.48	-
GLU-Net-GOCor*	2.26	9.89	5.53	18.27
PDC-Net	**2.08**	**7.98**	5.22	**15.13**
PWC-Net [59]	4.14	21.38	10.35	33.7
LiteFlowNet [20]	4.00	-	10.39	28.5
HD^{3}F [71]	4.65	-	13.17	24.0
LiteFlowNet2 [21]	3.42	-	8.97	25.9
VCN [70]	-	-	8.36	25.1
RAFT [60]	-	-	**5.04**	17.4

Table 2. Optical flow results on the training splits of KITTI [13]. The upper part contains generic matching networks, while the bottom part lists specialized optical flow methods, not trained on kitti.

4.3. Uncertainty Estimation

Next, we evaluate our uncertainty estimation. To assess the quality of the uncertainty estimates, we rely on Sparsification Error plots, in line with [1, 25, 67]. The pixels having the highest uncertainty are progressively removed and the AEPE or PCK of the remaining pixels is calculated, which results in the Sparsification curve. The Error curve is constructed by subtracting the Sparsification to the Oracle, for which the AEPE and PCK are calculated when the pixels are ranked according to the ground-truth error. As evaluation metric, we use the Area Under the Sparsification Error curve (AUSE). In Fig. 8, we compare the Sparsification Error plots on MegaDepth, of our PDC-Net with other dense methods providing a confidence estimation, namely DGC-Net [40] and RANSAC-Flow [53]. Our probabilistic method PDC-Net produces uncertainty maps that much better correspond to the true errors.

4.4. Pose and 3D Estimation

Finally, to show the joint performance of our flow and uncertainty prediction, we evaluate our approach for pose estimation. This application has traditionally been dominated by sparse matching methods.

Pose estimation: Given a pair of images showing different view-points of the same scene, two-view geometry estimation aims at recovering their relative pose. We follow the standard set-up of [72] and evaluate on 4 scenes of the **YFCC100M** dataset [61], each comprising 1000 image pairs. As evaluation metrics, we use mAP for dif-

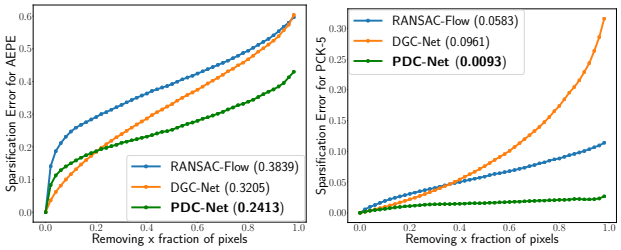


Figure 8. Sparsification Error plots for AEPE (left) and PCK-5 (right) on MegaDepth. Smaller AUSE (in parenthesis) is better.

ferent thresholds on the angular deviation between ground truth and predicted vectors for both rotation and translation. Results are presented in Tab. 3. Our approach PDC-Net outperforms the recent D2D [68] and obtains very similar results than RANSAC-Flow, while being 12.2 times faster. With our multi-scale (MS) strategy, PDC-Net outperforms RANSAC-Flow while being 3.6 times faster. Note that RANSAC-Flow employs its own MS strategy, using additional off-the-shelf features, which are exhaustively matched with nearest neighbor criteria [38]. In comparison, our proposed MS is a simpler, faster and more unified approach. We also note that RANSAC-Flow relies on a semantic segmentation network to better filter unreliable correspondences, in *e.g.* sky. Without this segmentation, the performance is drastically reduced. In contrast, our approach can directly estimate highly robust and generalizable confidence maps, without the need for additional network components. The confidence masks of RANSAC-Flow and our approach are visually compared in Fig. 6e-6f.

Extension to 3D reconstruction: We also qualitatively show the usability of our approach for dense 3D reconstruction. We compute dense correspondences between day-time images of the Aachen city from the Visual Localization benchmark [49, 50]. Accurate matches are then selected by thresholding our confidence map, and fed to COLMAP [51] to build a 3D point-cloud. It is visualized in Fig. 2.

4.5. Ablation study

Here, we perform a detailed analysis of our approach in Tab. 4. As baseline, we use a simplified version of GLU-Net, called BaseNet [64]. It is a three-level pyramidal network predicting the flow between an image pair. Our probabilistic approach integrated in this smaller architecture results in PDC-Net-s. All methods are trained using only the first-stage training, described in Sec. 4.1.

Probabilistic model (Tab. 4, top): We first compare BaseNet to our approach PDC-Net-s, modelling the flow distribution with a constrained mixture of Laplace (Sec. 3.2). On both KITTI-2015 and MegaDepth, our approach brings a significant improvement in terms of flow metrics. Note also that performing pose estimation by taking all correspondences (BaseNet) performs very poorly, which demonstrates the need for robust uncertainty estimation. While an unconstrained mixture of Laplace already drastically improves upon the single Laplace component,

	mAP @5°	mAP @10°	mAP @20°	Run-time (s)
Superpoint [8]	30.50	50.83	67.85	-
SIFT [38]	46.83	68.03	80.58	-
D2D [68]	55.58	66.79	-	-
RANSAC-Flow (MS+SegNet) [53]	64.88	73.31	81.56	9.06
RANSAC-Flow (MS) [53]	31.25	38.76	47.36	8.99
PDC-Net	63.90	73.00	81.22	**0.74**
PDC-Net (MS)	**65.18**	**74.21**	**82.42**	2.55

Table 3. Two-view geometry estimation on YFCC100M [61].

	KITTI-2015			MegaDepth			YFCC100M	
	AEPE	Fl (%)	AUSE	PCK-1 (%)	PCK-5 (%)	AUSE	mAP @5°	mAP @10°
BaseNet (L1-loss)	7.51	37.19	-	20.00	60.00	-	15.58	24.00
Single Laplace	6.86	34.27	0.220	27.45	62.24	0.210	26.95	37.10
Unconstrained Mixture	6.60	32.54	0.670	30.18	66.24	0.433	31.18	42.55
Constrained Mixture (PDC-Net-s)	**6.66**	**32.32**	**0.205**	**32.51**	**66.50**	**0.210**	**33.77**	**45.17**
Commun Dec.	6.41	32.03	**0.171**	31.93	**67.34**	0.213	31.13	42.21
Corr unc. module	**6.32**	**31.12**	0.418	31.97	66.80	0.278	**33.95**	**45.44**
Unc. Dec. (Fig 4) (PDC-Net-s)	6.66	32.32	0.205	**32.51**	66.50	**0.210**	33.77	45.17
BaseNet **w/o** Perturbations	7.21	37.35	-	20.74	59.35	-	15.15	23.88
BaseNet **w** Perturbations	7.51	37.19	-	20.00	60.00	-	15.58	24.00
PDC-Net-s **w/o** Perturbations	7.15	35.28	0.256	31.53	65.03	0.219	32.50	43.17
PDC-Net-s **w** Perturbations	**6.66**	**32.32**	**0.205**	**32.51**	**66.50**	**0.210**	**33.77**	**45.17**

Table 4. Ablation study. In the top part, different probabilistic models are compared (Sec. 3.1-3.2). In the middle part, a constrained Mixture is used, and different architectures for uncertainty estimation are compared (Sec. 3.3). In the bottom part, we analyze the impact of our training data with perturbations (Sec. 3.4).

the permutation invariance of the unconstrained mixture confuses the network, which results in poor uncertainty estimates (high AUSE). Constraining the mixture instead results in better metrics for both the flow and the uncertainty.

Uncertainty architecture (Tab. 4, middle): While the compared uncertainty decoder architectures achieve similar quality in flow prediction, they provide notable differences in uncertainty estimation. Only using the correlation uncertainty module leads to the best results on YFCC100M, since the module enables to efficiently discard unreliable matching regions, in particular compared to the common decoder approach. However, this module alone does not take into account motion boundaries. This leads to poor AUSE on KITTI-2015, which contains independently moving objects. Our final architecture (Fig. 4), additionally integrating the mean flow into the uncertainty estimation, offers the best compromise.

Perturbation data (Tab. 4, bottom): While introducing the perturbations does not help the flow prediction for BaseNet, it provides significant improvements in uncertainty *and* in flow performance for our PDC-Net-s. This emphasizes that the improvement of the uncertainty estimates originating from introducing the perturbations, also leads to improved and more generalizable flow predictions.

5. Conclusion

We propose a probabilistic deep network for estimating the dense image-to-image correspondences and associated confidence estimate. Specifically, we train the network to predict the parameters of the conditional probability density of the flow, which we model with a constrained mixture of Laplace distributions. Moreover, we introduce an architecture and improved self-supervised training strategy, designed for *robust and generalizable* uncertainty prediction. Our approach PDC-Net sets a new state-of-the-art on multiple geometric matching and optical flow datasets. It also outperforms dense matching methods on pose estimation.

Acknowledgements: This work was supported by the ETH Zürich Fund (OK), a Huawei Gift, Huawei Technologies Oy (Finland), Amazon AWS, and an Nvidia GPU grant.

References

[1] Oisin Mac Aodha, Ahmad Humayun, M. Pollefeys, and G. Brostow. Learning a confidence measure for optical flow. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1107–1120, 2013. 2, 7

[2] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *International Journal of Computer Vision*, 92(1):1–31, 2011. 1

[3] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *INTERNATIONAL JOURNAL OF COMPUTER VISION*, 12:43–77, 1994. 2

[4] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. Albumentations: Fast and flexible image augmentations. *Information*, 11(2), 2020. 15

[5] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In *BMVC*, 2014. 6, 17

[6] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 14

[7] Martin Danelljan, Luc Van Gool, and Radu Timofte. Probabilistic regression for visual tracking. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 7181–7190. IEEE, 2020. 4

[8] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *2018 IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 224–236, 2018. 8

[9] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019. 15

[10] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. 19

[11] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ICML'16, page 1050–1059. JMLR.org, 2016. 2

[12] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 3369–3378, 2018. 2, 3, 4, 6

[13] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *I. J. Robotic Res.*, 32(11):1231–1237, 2013. 2, 7, 18, 19

[14] Fredrik K. Gustafsson, Martin Danelljan, Goutam Bhat, and Thomas B. Schön. Energy-based models for deep probabilistic regression. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XX*, volume 12365 of *Lecture Notes in Computer Science*, pages 325–343. Springer, 2020. 4

[15] Yoav HaCohen, Eli Shechtman, Dan B. Goldman, and Dani Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. Graph.*, 30(4):70, 2011. 1

[16] Jared Heinly, Johannes Lutz Schönberger, Enrique Dunn, and Jan-Michael Frahm. Reconstructing the World* in Six Days *(As Captured by the Yahoo 100 Million Image Dataset). In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 19

[17] Berthold K. P. Horn and Brian G. Schunck. "determining optical flow": A retrospective. *Artif. Intell.*, 59(1-2):81–87, 1993. 1

[18] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society, 2017. 16, 17

[19] Shuaiyi Huang, Qiuyue Wang, Songyang Zhang, Shipeng Yan, and Xuming He. Dynamic context correspondence network for semantic alignment. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 2010–2019. IEEE, 2019. 2

[20] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8981–8989, 2018. 7

[21] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A Lightweight Optical Flow CNN - Revisiting Data Fidelity and Regularization. 2020. 7

[22] Junhwa Hur and S. Roth. Optical flow estimation in the deep learning age. *ArXiv*, abs/2004.02853, 2020. 1

[23] Andrey Ignatov, Nikolay Kobyshev, Radu Timofte, Kenneth Vanhoey, and Luc Van Gool. Dslr-quality photos on mobile devices with deep convolutional networks. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 3297–3305, 2017. 14

[24] Eddy Ilg, Özgün Çiçek, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part VII*, pages 677–693, 2018. 2, 3, 4, 6

[25] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. In *2017*

IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017, pages 1647–1655. IEEE Computer Society, 2017. 7

[26] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30, pages 5574–5584. Curran Associates, Inc., 2017. 2, 3, 4, 6

[27] Seungryong Kim, Dongbo Min, Somi Jeong, Sunok Kim, Sangryul Jeon, and Kwanghoon Sohn. Semantic attribute matching networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 12339–12348, 2019. 1

[28] Diederik P. Kingma and Jimmy 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*, 2015. 18

[29] Claudia Kondermann, Daniel Kondermann, Bernd Jähne, and Christoph S. Garbe. An adaptive confidence measure for optical flows based on linear subspace projections. In *Pattern Recognition, 29th DAGM Symposium, Heidelberg, Germany, September 12-14, 2007, Proceedings*, pages 132–141, 2007. 2

[30] Claudia Kondermann, Rudolf Mester, and Christoph S. Garbe. A statistical confidence measure for optical flows. In *Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part III*, pages 290–301, 2008. 2

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States.*, pages 1106–1114, 2012. 18

[32] Jan Kybic and Claudia Nieuwenhuis. Bootstrap optical flow confidence and uncertainty measure. *Comput. Vis. Image Underst.*, 115(10):1449–1462, 2011. 2

[33] Måns Larsson, Erik Stenborg, Lars Hammarstrand, Marc Pollefeys, Torsten Sattler, and Fredrik Kahl. A cross-season correspondence dataset for robust semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 9532–9542, 2019. 6, 7, 26

[34] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 2041–2050, 2018. 2, 3, 6, 7, 14, 27, 28, 29

[35] Jing Liao, Yuan Yao, Lu Yuan, Gang Hua, and Sing Bing Kang. Visual attribute transfer through deep image analogy. *ACM Trans. Graph.*, 36(4), July 2017. 1

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. 6, 14

[37] Ce Liu, Jenny Yuen, and Antonio Torralba. SIFT flow: Dense correspondence across scenes and its applications. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(5):978–994, 2011. 1, 7

[38] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004. 8

[39] Will Maddern, Geoff Pascoe, Chris Linegar, and Paul Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. 6, 7, 19

[40] Iaroslav Melekhov, Aleksei Tiulpin, Torsten Sattler, Marc Pollefeys, Esa Rahtu, and Juho Kannala. DGC-Net: Dense geometric correspondence network. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019. 1, 2, 5, 7, 14, 20

[41] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, June 2004. 19

[42] David Novotny, Samuel Albanie, Diane Larlus, and Andrea Vedaldi. Self-supervised learning of geometrically stable features through probabilistic introspection. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018. 2

[43] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 18

[44] Jianing Qian, Junyu Nan, Siddharth Ancha, Brian Okorn, and David Held. Robust instance tracking via uncertainty flow. *CoRR*, abs/2010.04367, 2020. 1, 3

[45] Jérôme Revaud, Philippe Weinzaepfel, Zaïd Harchaoui, and Cordelia Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *CVPR*, pages 1164–1172. IEEE Computer Society, 2015. 2

[46] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Convolutional neural network architecture for geometric matching. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 39–48, 2017. 2, 5, 6

[47] I. Rocco, R. Arandjelović, and J. Sivic. Efficient neighbourhood consensus networks via submanifold sparse convolutions. In *European Conference on Computer Vision*, 2020. 2

[48] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelovic, Akihiko Torii, Tomás Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 1658–1669, 2018. 1, 2, 7

[49] Torsten Sattler, Will Maddern, Carl Toft, Akihiko Torii, Lars Hammarstrand, Erik Stenborg, Daniel Safari, Masatoshi Okutomi, Marc Pollefeys, Josef Sivic, Fredrik Kahl, and

Tomás Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8601–8610, 2018. 2, 8, 19

[50] Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *British Machine Vision Conference, BMVC 2012, Surrey, UK, September 3-7, 2012*, pages 1–12, 2012. 8

[51] Johannes L. Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4104–4113, 2016. 1, 2, 8, 15, 19

[52] Thomas Schöps, Johannes L. Schönberger, Silvano Galliani, Torsten Sattler, Konrad Schindler, Marc Pollefeys, and Andreas Geiger. A multi-view stereo benchmark with high-resolution images and multi-camera videos. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2538–2547, 2017. 6, 7, 19

[53] Xi Shen, François Darmon, Alexei A Efros, and Mathieu Aubry. Ransac-flow: generic two-stage image alignment. In *16th European Conference on Computer Vision*, 2020. 1, 2, 5, 6, 7, 8, 17, 19, 20, 21

[54] Yichen Shen, Zhilu Zhang, Mert R. Sabuncu, and Lin Sun. Learning the distribution: A unified distillation paradigm for fast uncertainty estimation in computer vision. *CoRR*, abs/2007.15857, 2020. 3, 4

[55] Abhinav Shrivastava, Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Data-driven visual similarity for cross-domain image matching. *ACM Transaction of Graphics (TOG) (Proceedings of ACM SIGGRAPH ASIA)*, 30(6), 2011. 1

[56] Patrice Y. Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In *Proceedings of the Seventh International Conference on Document Analysis and Recognition - Volume 2*, ICDAR '03, page 958, USA, 2003. IEEE Computer Society. 15

[57] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27, pages 568–576. Curran Associates, Inc., 2014. 1

[58] Deqing Sun, Stefan Roth, and Michael J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *Int. J. Comput. Vision*, 106(2):115–137, Jan. 2014. 2

[59] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8934–8943, 2018. 7

[60] Zachary Teed and Jia Deng. RAFT: recurrent all-pairs field transforms for optical flow. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part II*, pages 402–419, 2020. 1, 7

[61] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: the new data in multimedia research. *Commun. ACM*, 59(2):64–73, 2016. 2, 7, 8, 25

[62] Prune Truong, Stefanos Apostolopoulos, Agata Mosinska, Samuel Stucky, Carlos Ciller, and Sandro De Zanet. Glampoints: Greedily learned accurate match points. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10731–10740, 2019. 1

[63] Prune Truong, Martin Danelljan, Luc Van Gool, and Radu Timofte. GOCor: Bringing globally optimized correspondence volumes into your neural network. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2020. 5, 6, 7, 14, 16, 17, 18, 19, 20

[64] Prune Truong, Martin Danelljan, and Radu Timofte. GLU-Net: Global-local universal network for dense flow and correspondences. In *2020 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2020*, 2020. 1, 2, 3, 5, 6, 7, 8, 16, 17, 18, 19, 20

[65] Ali Varamesh and Tinne Tuytelaars. Mixture dense regression for object detection and human pose estimation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 13083–13092, 2020. 4

[66] Stefanie Walz, Tobias Gruber, Werner Ritter, and Klaus Dietmayer. Uncertainty depth estimation with gated images for 3d reconstruction. In *23rd IEEE International Conference on Intelligent Transportation Systems, ITSC 2020, Rhodes, Greece, September 20-23, 2020*, pages 1–8. IEEE, 2020. 3, 4, 6

[67] Anne S. Wannenwetsch, Margret Keuper, and Stefan Roth. Probflow: Joint optical flow and uncertainty estimation. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 1182–1191, 2017. 2, 7

[68] Olivia Wiles, Sébastien Ehrhardt, and Andrew Zisserman. D2D: learning to find good correspondences for image matching and manipulation. *CoRR*, abs/2007.08480, 2020. 2, 8

[69] Bartlomiej Wronski, Ignacio Garcia-Dorado, Manfred Ernst, Damien Kelly, Michael Krainin, Chia-Kai Liang, Marc Levoy, and Peyman Milanfar. Handheld multi-frame super-resolution. *ACM Trans. Graph.*, 38(4):28:1–28:18, 2019. 21

[70] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32, pages 794–805. Curran Associates, Inc., 2019. 7

[71] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6044–6053, 2019. 2, 7

[72] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Hongen Liao, and Long

Quan. Learning two-view correspondences and geometry using order-aware network. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 5844–5853, 2019. 7, 19

[73] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *Int. J. Comput. Vis.*, 127(3):302–321, 2019. 14

Appendix

In this appendix, we first give a detailed derivation of our probabilistic model as a constrained mixture of Laplace distributions in Sec. A. In Sec. B, we then derive our probabilistic training loss and explain our training procedure in more depth. We subsequently follow by providing additional information about the architecture of our proposed networks as well as implementation details in Sec. C. In Sec. D, we extensively explain the evaluation datasets and set-up. Then, we present more detailed quantitative and qualitative results in Sec. E. Finally, we perform detailed ablative experiments in Sec. F.

A. Detailed derivation of probabilistic model

Here we provide the details of the derivation of our uncertainty estimate.

Probabilistic formulation: We model the flow estimation as a probabilistic regression with a constrained mixture density of Laplacian distributions (Sec. 3.2 of the main paper). Our mixture model, corresponding to equation (2) of the main paper, is expressed as,

$$p\left(y|\varphi\right) = \sum_{m=1}^{M} \alpha_m \mathcal{L}(y|\mu, \sigma_m^2) \tag{7}$$

where, for each component m, the bi-variate Laplace distribution $\mathcal{L}(y|\mu, \sigma_m^2)$ is computed as the product of two independent uni-variate Laplace distributions, such as,

$$\begin{aligned} \mathcal{L}(y|\mu, \sigma_m^2) &= \mathcal{L}(u, v|\mu_u, \mu_v, \sigma_u^2, \sigma_v^2) && \text{(8a)} \\ &= \mathcal{L}(u|\mu_u, \sigma_u^2).\mathcal{L}(v|\mu_v, \sigma_v^2) && \text{(8b)} \\ &= \frac{1}{\sqrt{2\sigma_u^2}} e^{-\sqrt{\frac{2}{\sigma_u^2}}|u-\mu_u|} \cdot \\ &\quad \frac{1}{\sqrt{2\sigma_v^2}} e^{-\sqrt{\frac{2}{\sigma_v^2}}|v-\mu_v|} && \text{(8c)} \end{aligned}$$

where $\mu = [\mu_u, \mu_v]^T \in \mathbb{R}^2$ and $\sigma_m^2 = [\sigma_u^2, \sigma_v^2]^T \in \mathbb{R}^2$ are respectively the mean and the variance parameters of the distribution $\mathcal{L}(y|\mu, \sigma_m^2)$. In this work, we additionally define equal variances in both flow directions, such that $\sigma_m^2 = \sigma_u^2 = \sigma_v^2 \in \mathbb{R}$. As a result, equation (8) simplifies, and when inserting into (7), we obtain,

$$p\left(y|\varphi\right) = \sum_{m=1}^{M} \alpha_m \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}. \tag{9}$$

Confidence estimation: Our network Φ thus outputs, for each pixel location, the parameters of the predictive distribution, *i.e.* the mean flow μ along with the variance σ_m^2 and weight α_m of each component, as $\left(\mu, (\alpha_m)_{m=1}^M, (\sigma_m^2)_{m=1}^M\right) = \varphi(X;\theta)$. However, we aim at obtaining a *single* confidence value to represent the relibiability of the estimated flow vector μ. As a final confidence measure, we thus compute the probability P_R of the true flow being within a radius R of the estimated mean flow vector μ. This is expressed as,

$$\begin{aligned} P_R &= P(\|y-\mu\|_\infty < R) && \text{(10a)} \\ &= \int_{\{y\in\mathbb{R}^2 : \|y-\mu\|_\infty < R\}} p(y|\varphi) dy && \text{(10b)} \\ &= \sum_m \alpha_m \int_{\mu_u-R}^{\mu_u+R} \frac{1}{\sqrt{2}\sigma_m} e^{-\sqrt{2}\frac{|u-\mu_u|}{\sigma_m}} du \\ &\quad \int_{\mu_v-R}^{\mu_v+R} \frac{1}{\sqrt{2}\sigma_m} e^{-\sqrt{2}\frac{|v-\mu_v|}{\sigma_m}} dv && \text{(10c)} \\ &= \sum_m \alpha_m \left[1 - \exp(-\sqrt{2}\frac{R}{\sigma_m})\right]^2 && \text{(10d)} \end{aligned}$$

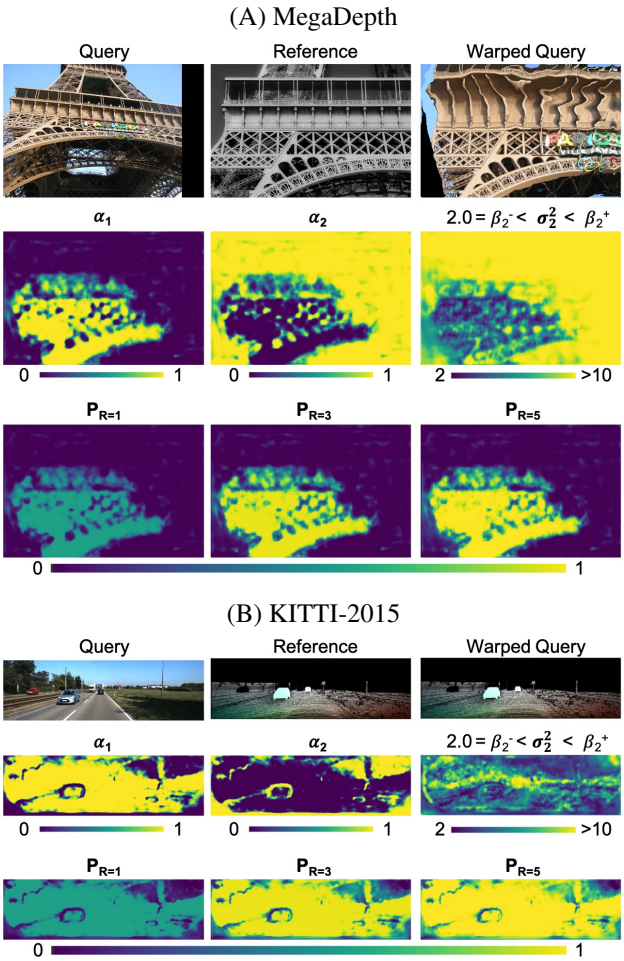


Figure 9. Visualization of the mixture parameters $(\alpha_m)_{m=1}^M$ and σ_2^2 predicted by our final network PDC-Net, on multiple image pairs. PDC-Net has $M = 2$ Laplace components and here, we do not represent the scale parameter σ_1^2, since it is fixed as $\sigma_1^2 = 1.0$. We also show the resulting confidence maps P_R for multiple R.

where we have here expressed P_R with the standard deviation parameters σ_m instead of the variance parameters σ_m^2 for ease of notation. This confidence measure is used to identify the accurate matches by thesholding P_R. In Fig 9, we visualize the estimated mixture parameters $(\alpha_m)_{m=1}^M, (\sigma_m^2)_{m=1}^M$, and the resulting confidence map P_R for multiple image pair examples.

B. Training details

In this section, we derive the numerically stable Negative Log-Likelihood loss, used for training our network PDC-Net. We also describe in details the employed training datasets.

B.1. Training loss

Similar to conventional approaches, probabilistic methods are generally trained using a set of *iid.* image pairs $\mathcal{D} = \left\{X^{(n)}, Y^{(n)}\right\}_{n=1}^N$. The negative log-likelihood provides a general framework for fitting a distribution to the training dataset as,

$$L(\theta; \mathcal{D}) = -\frac{1}{N}\sum_{n=1}^{N} \log p\left(Y^{(n)} | \Phi(X^{(n)}; \theta)\right) \tag{11a}$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\sum_{ij} \log p\big(y_{ij}^{(n)} | \varphi_{ij}(X^{(n)}; \theta)\big) \tag{11b}$$

Inserting (9) into (11), we obtain for the last term the following expression,

$$L_{ij} = -\log p\big(y_{ij}^{(n)} | \varphi_{ij}(X^{(n)}; \theta)\big) \tag{12a}$$

$$= -\log\left(\sum_{m=1}^{M} \alpha_m \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}\right) \tag{12b}$$

$$= -\log\left(\sum_{m=1}^{M} \frac{e^{\tilde{\alpha}_m}}{\sum_{m=1}^{M} e^{\tilde{\alpha}_m}} \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}\right) \tag{12c}$$

$$= \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m}\right) - \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m} \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}\right) \tag{12d}$$

$$= \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m}\right) - \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m - \log(2) - s_m - \sqrt{2} e^{-\frac{1}{2}s_m}.|y-\mu|_1}\right) \tag{12e}$$

where $s_m = log(\sigma_m^2)$. Indeed, in practise, to avoid division by zero, we use $s_m = \log(\sigma_m^2)$ for all components $m \in \{0, .., M\}$ of the mixture density model. For the implementation of the loss, we use a numerically stable log-sumexp function.

With a simple regression loss such as the L1 loss, the large errors represented by the heavy tail of the distribution in Fig. 3 of the main paper have a disproportionately large impact on the loss, preventing the network from focusing on the more accurate predictions. On the contrary, the loss (12) enables to down-weight the contribution of these examples by predicting a high variance parameter for them. Modelling the flow estimation as a conditional predictive distribution thus improves the accuracy of the estimated flow itself.

B.2. Training datasets

Due to the limited amount of available real correspondence data, most matching methods resort to self-supervised training, relying on synthetic image warps generated automatically. We here provide details on the synthetic dataset that we use for self-supervised training, as well as additional information on the implementation of the perturbation data (Sec. 3.4 of the main paper). Finally, we also describe the generation of the sparse ground-truth correspondence data from the MegaDepth dataset [34].

Base synthetic dataset: For our base synthetic dataset, we use the same data as in [63]. Specifically, pairs of images are created by warping a collection of images from the DPED [23], CityScapes [6] and ADE-20K [73] datasets, according to synthetic affine, TPS and homography transformations. The transformation parameters are the ones originally used in DGC-Net [40].

These image pairs are further augmented with additional random independently moving objects. To do so, the objects are sampled from the COCO dataset [36], and inserted on top of the images of the synthetic data using their segmentation masks. To generate motion, we randomly sample affine transformation parameters for the foreground objects, which are independent of the background transformations. This can be interpreted as both the camera and the objects moving independently of each other. The final synthetic flow is composed of the object motion flow field at the location of the moving object in the reference image, or the background flow field otherwise. It results in 40K image pairs, cropped at resolution 520×520.

Perturbation data for robust uncertainty estimation: Even with independently moving objects, the network still learns to primarily rely on interpolation when estimating the flow field and corresponding uncertainty map relating an image pair. We here describe in more details our data generation strategy for more robust uncertainty prediction. From a base flow field $Y \in \mathbb{R}^{H\times W\times 2}$ relating a reference image $\tilde{I}^r \in \mathbb{R}^{H\times W\times 3}$ to a query image $\tilde{I}^q \in \mathbb{R}^{H\times W\times 3}$, we introduce a residual flow $\epsilon = \sum_i \varepsilon_i$,

Figure 10. Visualization of our perturbations applied to a pair of reference and query images (Sec. 3.4 of the main paper).

by adding small local perturbations $\varepsilon_i \in \mathbb{R}^{H\times W\times 2}$. More specifically, we create the residual flow by first generating an elastic deformation motion field E on a dense grid of dimension $H \times W$, as described in [56]. Since we only want to include perturbations in multiple small regions, we generate binary masks $S_i \in \mathbb{R}^{H\times W\times 2}$, each delimiting the area on which to apply one local perturbation ε_i. The final residual flow (perturbations) thus take the form of $\epsilon = \sum_i \varepsilon_i$, where $\varepsilon_i = E \cdot S_i$. Finally, the query image $I^q = \tilde{I}^q$ is left unchanged while the reference I^r is generated by warping $\tilde{I}^r$ according to the residual flow ϵ, as $I^r(x) = \tilde{I}^r(x + \epsilon(x))$. The final perturbed flow map Y between I^r and I^q is achieved by composing the base flow $\tilde{Y}$ with the residual flow ϵ, as $Y(x) = \tilde{Y}(x + \epsilon(x)) + \epsilon(x)$.

In practise, for the elastic deformation field E, we use the implementation of [4]. The masks S_i should be between 0 and 1 and offer a smooth transition between the two, so that the perturbations appear smoothly. To create each mask S_i, we thus generate a 2D Gaussian centered at a random location and with a random standard deviation (up to a certain value) on a dense grid of size $H \times W$. It is then scaled to 2.0 and clipped to 1.0, to obtain a smooth regions equal to 1.0 where the perturbation will be applied, and transition regions on all sides from 1.0 to 0.0.

In Fig. 10, we show examples of generated residual flows and their corresponding perturbed reference I^r, for a particular base flow Y, and query $\tilde{I}^r$ and reference $\tilde{I}^q$ images. As one can see, for human eye, it is almost impossible to detect the presence of the perturbations on the perturbed reference I^r. This will enable to "fool" the network in homogeneous regions, such as the road in the figure example, thus forcing it to predict high uncertainty in regions where it cannot identify them.

MegaDepth training: To generate the training pairs with sparse ground-truth, we adapt the generation protocol of D2-Net [9]. Specifically, we use the MegaDepth dataset, consisting of 196 different scenes reconstructed from 1.070.468 internet photos using COLMAP [51]. The camera intrinsics and extrinsics as well as depth maps from Multi-View Stereo are provided by the authors for 102.681 images.

For training, we use 150 scenes and sample up to 500 random images with an overlap ratio of at least 30% in the sparse SfM point cloud. For each pair, all points of the second image with depth information are projected into the first image. A depth-check with respect to the depth map of the first image is also run to remove occluded pixels. It results in around 58.000 training pairs, which we resized so that their largest dimension is 520. Note that we use the same set of training pairs at each training iteration. For the validation dataset, we sample up to 100 image pairs from 25 different scenes, leading to approximately 1800 image pairs.

During the second stage of training, we found it crucial to train on *both* the synthetic dataset with perturbations and the sparse data from MegaDepth. Training solely on the sparse correspondences resulted in less reliable uncertainty estimates.

C. Architecture details

In this section, we first describe the architecture of our proposed uncertainty decoder (Sec. 3.3 of the main paper). We then give additional details about our proposed final architecture PDC-Net and its corresponding baseline GLU-Net-GOCor*. We also describe the architecture of BaseNet and its probabilistic derivatives, employed for the ablation study. Finally, we share all training details and hyper-parameters.

C.1. Architecture of the uncertainty decoder

Correlation uncertainty module: We first describe the architecture of our Correlation Uncertainty Module U_θ (Sec. 3.3 of the main paper). The correlation uncertainty module processes each 2D slice $C_{ij..}$ of the correspondence

volume C independently. More practically, from the correspondence volume tensor $C \in \mathbb{R}^{b\times h\times w\times (d\times d)}$, where b indicates the batch dimension, we move the spatial dimensions $h \times w$ into the batch dimension and we apply multiple convolutions in the displacement dimensions $d \times d$, *i.e.* on a tensor of shape $(b \times h \times w) \times d \times d \times 1$. By applying the strided convolutions, the spatial dimension is gradually decreased, resulting in an uncertainty representation $u \in \mathbb{R}^{(b\times h\times w)\times 1\times 1\times n}$, where n denotes the number of channels. u is subsequently rearranged, and after dropping the batch dimension, the outputted uncertainty tensor is $u \in \mathbb{R}^{h\times w\times n}$.

Note that while this is explained for a local correlation, the same applies for a global correlation except that the displacement dimensions correspond to $h \times w$. In Tab. 5- 6, we present the architecture of the convolution layers applied on the displacements dimensions, for a local correlation with search radius 4 and for a global correlation applied at dimension $h \times w = 16 \times 16$, respectively.

Uncertainty predictor: We then give additional details of the Uncertainty Predictor, that we denote Q_θ (Sec. 3.3 of the main paper). The uncertainty predictor takes the flow field $Y \in \mathbb{R}^{h\times w\times 2}$ outputted from the flow decoder, along with the output $u \in \mathbb{R}^{h\times w\times n}$ of the correlation uncertainty module U_θ. In a multi-scale architecture, it additionally takes as input the estimated flow field and predicted uncertainty components from the previous level. At level l, for each pixel location (i, j), this is expressed as:

$$\left((\tilde{\alpha}_m)_{m=1}^M, (h_m)_{m=1}^M\right)^l = Q_\theta\left(Y^l; u^l; \Phi^{l-1}\right)_{ij} \qquad (13)$$

where $\tilde{\alpha}_m$ refers to the output of the uncertainty predictor,

Inputs	Convolutions	Output size
C; $(b \times h \times w) \times 9 \times 9 \times 1$	$conv_0$, $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 7 \times 7 \times 32$
$conv_0$; $(b \times h \times w) \times 7 \times 7 \times 32$	$conv_1$, $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 5 \times 5 \times 32$
$conv_1$; $(b \times h \times w) \times 5 \times 5 \times 32$	$conv_2$, $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 3 \times 3 \times 16$
$conv_2$; $(b \times h \times w) \times 3 \times 3 \times 16$	$conv_3$, $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 1 \times 1 \times n$

Table 5. Architecture of the correlation uncertainty module for a local correlation, with a displacement radius of 4. Implicit are the BatchNorm and ReLU operations that follow each convolution, except for the last one. K refers to kernel size, s is the used stride and p the padding.

Inputs	Convolutions	Output size
C $(b \times h \times w) \times 16 \times 16 \times 1$	$conv_0$; $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 14 \times 14 \times 32$
$conv_0$; $(b \times h \times w) \times 14 \times 14 \times 32$	3 × 3 max pool, s=2 $conv_1$, $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 5 \times 5 \times 32$
$conv_1$; $(b \times h \times w) \times 5 \times 5 \times 32$	$conv_2$, $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 3 \times 3 \times 16$
$conv_2$; $(b \times h \times w) \times 3 \times 3 \times 16$	$conv_3$, $K = (3 \times 3)$, s=1, p=0	$(b \times h \times w) \times 1 \times 1 \times n$

Table 6. Architecture of the correlation uncertainty module for a global correlation, constructed at resolution 16×16. Implicit are the BatchNorm and ReLU operations that follow each convolution, except for the last one. K refers to kernel size, s is the used stride and p the padding.

which is then passed through a SoftMax layer to obtain the final weights α_m. σ_m^2 is obtained from h_m according to constraint equation (3) of the main paper.

In practise, we have found that instead of feeding the flow field $Y \in \mathbb{R}^{h\times w\times 2}$ outputted from the flow decoder to the uncertainty predictor, using the second last layer from the flow decoder leads to slightly better results. This is because the second last layer from the flow decoder has larger channel size, and therefore encodes more information about the estimated flow.

Architecture-wise, the uncertainty predictor Q_θ consists of 3 convolutional layers. The numbers of feature channels at each convolution layers are respectively 32, 16 and $2M$ and the spatial kernel of each convolution is 3×3 with stride of 1 and padding 1. The first two layers are followed by a batch-normalization layer with a leaky-Relu non linearity. The final output of the uncertainty predictor is the result of a linear 2D convolution, without any activation.

C.2. Architecture of PDC-Net

We use GLU-Net-GOCor [64, 63] as our base architecture, predicting the dense flow field relating a pair of images. It is a 4 level pyramidal network, using a VGG feature backbone. It is composed of two sub-networks, L-Net and H-Net which act at two different resolutions. The L-Net takes as input rescaled images to $H_L \times W_L = 256 \times 256$ and process them with a global GOCor module followed by a local GOCor module. The resulting flow is then upsampled to the lowest resolution of the H-Net to serve as initial flow, by warping the query features according to the estimated flow. The H-Net takes input images at unconstrained resolution $H \times W$, and refines the estimated flow with two local GOCor modules.

For the baseline GLU-Net-GOCor*, we adopt the GLU-Net-GOCor architecture and simply replace the DenseNet connections [18] of the flow decoders by standard residual blocks. The mapping decoder is also modified to include residual connections. This drastically reduces the number of weights while not having any impact on performance. As in [64, 63], the VGG-16 backbone is initialized to the pre-trained weights on ImageNet.

From the baseline GLU-Net-GOCor*, we create our probabilistic approach PDC-Net by inserting our uncertainty decoder at each pyramid level. As noted in C.1, in practise, we feed the second last layer from the flow decoder to the uncertainty predictor of each pyramid level instead of the predicted flow field. It leads to slightly better results. The uncertainty prediction is additionally *propagated from one level to the next*. More specifically, the flow decoder takes as input the uncertainty prediction (all parameters Φ of the predictive distribution except for the mean flow) of the previous level, in addition to its original inputs (which include the mean flow of the previous

level). The uncertainty predictor also takes the uncertainty and the flow estimated at the previous level. As explained in Sec. 4.1 of the main paper, we use a constrained mixture with $M = 2$ Laplace components. The first component is set so that $\sigma_1^2 = 1$, while the second is learned as $2 = \beta_2^- \leq \sigma_2^2 \leq \beta_2^+$. Therefore, the uncertainty predictor only estimates σ_2^2 and $(\alpha_m)_{m=1}^{M=2}$ at each pixel location. We found that fixing $\sigma_1^2 = \beta_1^- = \beta_1^+ = 1.0$ results in better performance than for example $\beta_1^- = 0.0 < \sigma_1^2 < \beta_1^+ = 1.0$. Indeed, in the later case, during training, the network focuses primarily on getting the very accurate, and confident, correspondences (corresponding to σ_1^2) since it can arbitrarily reduce the variance. Generating fewer, but accurate predictions then dominate during training to the expense of other regions. This is effectively alleviated by setting $\sigma_1^2 = 1.0$, which can be seen as introducing a strict prior on this parameter.

C.3. Inference multi-stage

Here, we provide implementation details for our multi-stage inference strategy (Sec. 3.5 of the main paper). After the first network forward pass, we select matches with a corresponding confidence probability $P_{R=1}$ superior to 0.1, for $R = 1$. Since the network estimates the flow field at a quarter of the original image resolution, we use the filtered correspondences at the quarter resolution and scale them to original resolution to be used for homography estimation. To estimate the homography, we use OpenCV's findHomography with RANSAC and an inlier threshold of 1 pixel.

C.4. Inference multi-scale

We then give additional details about our multi-scale strategy (MS). We extend our two-stage refinement approach (Sec. 3.5) by resizing the reference image to different resolutions. Specifically, following [53], we use seven scales: 0.5, 0.88, 1, 1.33, 1.66 and 2.0. As for the implementation, to avoid obtaining very large input images (for scaling ratio 2 for example), we use the following scheme: we resize the reference image for scaling ratios below 1, keeping the aspect ratio fixed and the query image unchanged. For ratios above 1, we instead resize the query image by one divided by the ratio, while keeping the reference image unchanged. This ensures that the resized images are never larger than the original image dimensions. The resulting image pairs are then passed through the network and we fit a homography for each pair, using our predicted flow and uncertainty map. In particular, as in our two-stage inference strategy, we select matches with a corresponding confidence probability $P_{R=1}$ superior to 0.1, for $R = 1$, at the estimated flow resolution, *i.e.* at a quarter of the input image resolution. To estimate the homography, we use OpenCV's findHomography with RANSAC and an inlier threshold of 1 pixel. From all image pairs with their corresponding scaling ratios, we then select the homography with the highest percentage of inliers, and scale it to the images original resolutions. The original image pair is then coarsely aligned using this homography and from there we follow the same procedure, as explained in Sec. 3.5.

C.5. Architecture of BaseNet

As baseline to use in our ablation study, we use BaseNet, introduced in [63] and inspired by GLU-Net [64]. It estimates the dense flow field relating an input image pair. The network is composed of three pyramid levels and it uses VGG-16 [5] as feature extractor backbone. The coarsest level is based on a global correlation layer, followed by a mapping decoder estimating the correspondence map at this resolution. The two next pyramid levels instead rely on local correlation layers. The dense flow field is then estimated with flow decoders, taking as input the correspondence volumes resulting from the local feature correlation layers. Moreover, BaseNet is restricted to a pre-determined input resolution $H_L \times W_L = 256 \times 256$ due to its global correlation at the coarsest pyramid level. It estimates a final flow-field at a quarter of the input resolution $H_L \times W_L$, which needs to be upsampled to original image resolution $H \times W$. The mapping and flow decoders have the same number of layers and parameters as those used for GLU-Net [64]. However, here, to reduce the number of weights, we use feed-forward layers instead of DenseNet connections [18] for the flow decoders.

We create the different probabilistic versions of BaseNet, presented in the ablation study Tab. 4 of the main paper, by modifying the architecture minimally. Moreover, for the probabilistic versions modeled with a constrained mixture, we use $M = 2$ Laplace components. The first component is set so that $\sigma_1^2 = 1$, while the second is learned as $2 = \beta_2^- \leq \sigma_2^2 \leq \beta_2^+ = \infty$. For the network referred to as PDC-Nets, which also employs our proposed uncertainty architecture (Sec. 3.3 of the main paper), we add our uncertainty decoder at each pyramid layer, in a similar fashion as for our final network PDC-Net. We train all networks on the synthetic data with the perturbations, which corresponds to our first training stage (Sec. 4.1).

C.6. Implementation details

Since we use pyramidal architectures with K levels, we employ a multi-scale training loss, where the loss at different pyramid levels account for different weights.

$$\mathcal{L}(\theta) = \sum_{l=1}^{K} \gamma_l L_l + \eta \left\| \theta \right\| , \tag{14}$$

where γ_l are the weights applied to each pyramid level and L_l is the corresponding loss computed at each level, which

refers to the L1 loss for the non-probabilistic baselines and the negative log-likelihood loss (11) for the probabilistic models, including our approach PDC-Net. The second term of the loss (14) regularizes the weights of the network. Moreover, during the self-supervised training, we do not apply any mask, which means that out-of-view regions (that do not have visible matches) are included in the training loss. Since the image pairs are related by synthetic transformations, these regions do have a correct ground-truth flow value. When finetuning on MegaDepth images however, the loss is applied only at the locations of the sparse ground-truth.

For training, we use similar training parameters as in [64]. Specifically, as a preprocessing step, the training images are mean-centered and normalized using mean and standard deviation of ImageNet dataset [31]. For all local correlation layers, we employ a search radius $r = 4$.

For the training of BaseNet and its probabilistic derivatives (including PDC-Net-s), which have a pre-determined fixed input image resolution of $(H_L \times W_L = 256 \times 256)$, we use a batch size of 32 and train for 106.250 iterations. We set the initial learning rate to 10^{-2} and gradually decrease it by 2 after 56.250, 75.000 and 93.750 iterations. The weights in the training loss (14) are set to be $\gamma_1 = 0.32, \gamma_2 = 0.08, \gamma_3 = 0.02$ and to compute the loss, we down-sample the ground-truth to estimated flow resolution at each pyramid level.

For GLU-Net-GOCor* and PDC-Net, we down-sample and scale the ground truth from original resolution $H \times W$ to $H_L \times W_L$ in order to obtain the ground truth flow fields for L-Net. During the first stage of training, *i.e.* on purely synthetic images, we down-sample the ground truth from the base resolution to the different pyramid resolutions without further scaling, so as to obtain the supervision signals at the different levels. During this stage, the weights in the training loss (14) are set to be $\gamma_1 = 0.32, \gamma_2 = 0.08, \gamma_3 = 0.02, \gamma_4 = 0.01$, which ensures that the loss computed at each pyramid level contributes equally to the final loss (14). During the second stage of training however, which includes MegaDepth, since the ground-truth is sparse, it is inconvenient to down-sample it to different resolutions. We thus instead up-sample the estimated flow field to the ground-truth resolution and compute the loss at this resolution. In practise, we found that both strategies lead to similar results during the self-supervised training. During the second training stage, the weights in the training loss (14) are instead set to $\gamma_1 = 0.08, \gamma_2 = 0.08, \gamma_3 = 0.02, \gamma_4 = 0.02$, which also ensures that the loss terms of all pyramid levels have the same magnitude.

During the first training stage on uniquely the self-supervised data, we train for 135.000 iterations, with batch size of 15. The learning rate is initially equal to 10^{-4}, and halved after 80.000 and 108.000 iterations. Note that during this first training stage, the feature back-bone is frozen, but further finetuned during the second training stage. While finetuning on the composition of MegaDepth and the synthetic dataset, the batch size is reduced to 10 and we further train for 195.000 iterations. The initial learning rate is fixed to 5.10^{-5} and halved after 120.000 and 180.000 iterations. The feature back-bone is also finetuned according to the same schedule, but with an initial learning rate of 10^{-5}. For the GOCor modules [63], we train with 3 local and global optimization iterations.

Our system is implemented using Pytorch [43] and our networks are trained using Adam optimizer [28] with weight decay of 0.0004.

D. Experimental setup and datasets

In this section, we first provide details about the evaluation datasets and metrics. We then explain the experimental set-up in more depth.

D.1. Evaluation metrics

AEPE: AEPE is defined as the Euclidean distance between estimated and ground truth flow fields, averaged over all valid pixels of the reference image.

PCK: The Percentage of Correct Keypoints (PCK) is computed as the percentage of correspondences $\tilde{\mathbf{x}}_j$ with an Euclidean distance error $\|\tilde{\mathbf{x}}_j - \mathbf{x}_j\| \leq T$, w.r.t. to the ground truth $\mathbf{x}_j$, that is smaller than a threshold T.

Fl: Fl designates the percentage of outliers averaged over all valid pixels of the dataset [13]. They are defined as follows, where Y indicates the ground-truth flow field and $\hat{Y}$ the estimated flow by the network.

$$Fl = \frac{\left\|Y - \hat{Y}\right\| > 3 \text{ and } \frac{\left\|Y - \hat{Y}\right\|}{\|Y\|} > 0.05}{\#\text{valid pixels}} \tag{15}$$

Sparsification Errors: Sparsification plots measure how well the estimated uncertainties fit the true errors. The pixels of a flow field are sorted according to their corresponding uncertainty, in descending order. An increasing percentage of the pixels is subsequently removed, and the AEPE or PCK of the remaining pixels is calculated. We refer to these curves as Sparsification. As reference, we also compute the Oracle, which represents the AEPE or PCK calculated when the pixels are ranked according to the true error, computed with the ground-truth flow field. Ideally, if the estimated uncertainty is a good representation of the underlying error distribution, the Sparsification should be close to the Oracle. To compare methods, since each approach results in a different oracle, we use the Area Under the Sparsification Error Curve (AUSE), where the Sparsification Error is defined as the difference between the sparsification and its oracle. We compute the sparsification error curve on each

image pair, and normalize it to $[0, 1]$. The final error curve is the average over all image pairs of the dataset.

mAP: For the task of pose estimation, we use mAP as the evaluation metric, following [72]. The absolute rotation error $|R_{err}|$ is computed as the absolute value of the rotation angle needed to align ground-truth rotation matrix R with estimated rotation matrix $\hat{R}$, such as

$$R_{err} = cos^{-1} \frac{Tr(R^{-1}\hat{R}) - 1}{2} , \tag{16}$$

where operator Tr denotes the trace of a matrix. The translation error T_{err} is computed similarly, as the angle to align the ground-truth translation vector T with the estimated translation vector $\hat{T}$.

$$T_{err} = cos^{-1} \frac{T \cdot \hat{T}}{\|T\| \left\|\hat{T}\right\|} , \tag{17}$$

where $\cdot$ denotes the dot-product. The accuracy Acc-κ for a threshold κ is computed as the percentage of image pairs for which the maximum of T_{err} and $|R_{err}|$ is below this threshold. mAP is defined according to original implementation [72], *i.e.* mAP @5° is equal to Acc-5, mAP @10° is the average of Acc-5 and Acc-10, while mAP @20° is the average over Acc-5, Acc-10, Acc-15 and Acc-20.

D.2. Evaluation datasets and set-up

MegaDepth: The MegaDepth dataset depicts real scenes with extreme viewpoint changes. No real ground-truth correspondences are available, so we use the result of SfM reconstructions to obtain sparse ground-truth correspondences. We follow the same procedure and test images than [53]. More precisely, we randomly sample 1600 pairs of images that shared more than 30 points. The test pais are from different scenes than the ones we used for training and validation. We use 3D points from SfM reconstructions and project them onto the pairs of matching images to obtain correspondences. It results in approximately 367K correspondences. During evaluation, following [53], all the images are resized to have minimum dimension 480 pixels.

RobotCar: In RobotCar, we used the correspondences originally introduced by [39]. During evaluation, following [53], all the images are resized to have minimum dimension 480 pixels.

ETH3D: The Multi-view dataset ETH3D [52] contains 10 image sequences at 480×752 or 514×955 resolution, depicting indoor and outdoor scenes. They result from the movement of a camera completely unconstrained, used for benchmarking 3D reconstruction. The authors additionally provide a set of sparse geometrically consistent image correspondences (generated by [51]) that have been optimized over the entire image sequence using the reprojection error. We sample image pairs from each sequence at different intervals to analyze varying magnitude of geometric transformations, and use the provided points as sparse ground truth correspondences. This results in about 500 image pairs in total for each selected interval, or 600K to 1000K correspondences. Note that, in this work, we computed the PCK over the whole dataset (image pairs of all the sequences) per interval, to be consistent with RANSAC-Flow. This metric is different than the one originally used by [64, 63] for ETH3D, where the PCK was calculated per image instead and averaged per sequence.

KITTI: The KITTI dataset [13] is composed of real road sequences captured by a car-mounted stereo camera rig. The KITTI benchmark is targeted for autonomous driving applications and its semi-dense ground truth is collected using LIDAR. The 2012 set only consists of static scenes while the 2015 set is extended to dynamic scenes via human annotations. The later contains large motion, severe illumination changes, and occlusions.

YFCC100M: The YFCC100M dataset represents touristic landmark images. The ground-truth poses were created by generating 3D reconstructions from a subset of the collections [16]. Since our network PDC-Net outputs flow fields at a quarter of the images original resolution, which are then usually up-sampled, for pose estimation, we directly select matches at the outputted resolution and further scale the correspondences to original resolution. From the estimated dense flow field, we identify the accurate correspondences by thresholding the predicted confidence map P_R (Sec 3.5 of the main paper, and Sec. A). Specifically, we select correspondences for which the corresponding confidence level at $R = 1$ is superior to 0.1, such as $P_{R=1} > 0.1$. We then use the selected matches to estimate an essential matrix with RANSAC [10] and 5-pt Nister algorithm [41], relying on OpenCV's 'findEssentialMat' with an inlier threshold of 1 pixel divided by the focal length. Rotation matrix $\hat{R}$ and translation vector $\hat{T}$ are finally computed from the estimated essential matrix, using OpenCV's 'recoverPose'. The original images are resized to have a minimum dimension of 480, similar to [53], and the intrinsic camera parameters are modified accordingly.

3D reconstruction on Aachen dataset: We use the set-up of [49], which provides a list of image pairs to match. We compute dense correspondences between each pair. We resize the images by keeping the same aspect ratio so that the minimum dimension is 600. We select matches for which the confidence probability $P_{R=1}$ is above 0.3, and feed them to COLMAP reconstruction pipeline [51]. Again, we select matches at a quarter of the image resolution and scale the matches to original resolution. Following Fig. 2 of the main paper, additional qualitative representations of the resulting 3D reconstruction are shown in Fig. 14.

	MegaDepth			RobotCar			KITTI-2012		KITTI-2015		YFCC100M		
	PCK-1	PCK-3	PCK-5	PCK-1	PCK-3	PCK-5	AEPE ↓	Fl (%) ↓	AEPE ↓	Fl (%) ↓	mAP @5°	mAP @10°	mAP @20°
DGC-Net [40]	3.55	20.33	32.28	1.19	9.35	20.17	8.50	32.28	14.97	50.98	6.73	12.55	22.42
GLU-Net [64]	21.58	52.18	61.78	2.30	17.15	33.87	3.14	19.76	7.49	33.83	21.35	30.73	42.91
GLU-Net-GOCor [63]	37.28	61.18	68.08	2.31	17.62	35.18	2.68	15.43	**6.68**	27.57	24.53	33.56	45.34
GLU-Net-GOCor*	41.36	62.37	67.23	2.07	15.57	30.86	2.95	14.05	7.14	25.02	24.55	32.55	43.31
PDC-Net	53.06	70.88	73.94	**2.54**	**18.85**	**36.24**	**2.44**	**11.09**	6.82	**21.79**	47.40	56.46	65.33
PDC-Net (MS)	**56.49**	**76.65**	**80.18**	2.53	18.68	36.03	-	-	-	-	**53.80**	**63.94**	**73.86**

Table 7. Results on multiple geometric and optical flow datasets as well as for pose estimation on the YFCC100M dataset. All methods are trained on purely self-supervised data.

On all datasets, we use our multi-stage strategy (Sec. 3.5 of the main paper), except for the KITTI datasets. Indeed, on optical flow data, which shows limited displacements and appearance variation, multi-stage strategy does not bring any improvements, and solely increases the runtime. For evaluation, we use 3 and 7 steepest descent iterations in the global and local GOCor modules [63] respectively. We reported results of RANSAC-Flow [53] using MOCO features, which gave the best results overall.

E. Detailed results

In this section, we first provide detailed results on uncertainty estimation. Subsequently, we present results of our approach after solely the first training stage, *i.e.* on uniquely self-supervised data. Finally, we present extensive qualitative results and comparisons.

E.1. Detailed results on uncertainty estimation

Here, we present sparsification errors curves, computed on the RobotCar dataset. As in the main paper, Sec. 4.3, we compare our probabilistic approach PDC-Net, to dense geometric methods providing a confidence estimation, DGC-Net [40] and RANSAC-Flow [53]. Fig. 11 depicts the sparsification error curves on RobotCar. As on MegaDepth, our approach PDC-Net estimates uncertainty map which better fit the underlying errors.

E.2. Detailed results on ETH3D

In Tab. 8, we show the detailed results on ETH3D, corresponding to Fig. 7 of the main paper.

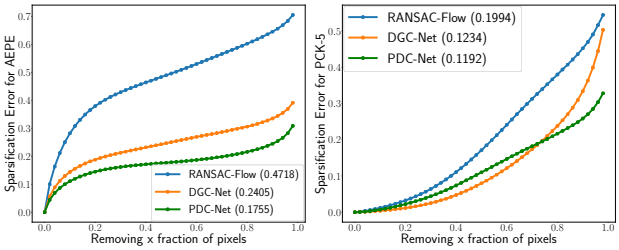


Figure 11. Sparsification Error plots for AEPE (left) and PCK-5 (right) on RobotCar. Smaller AUSE (in parenthesis) is better.

E.3. Results when trained on purely synthetic data

For completeness, here, we show results of our approach PDC-Net, after only the first stage of training (described in Sec. 4.1 of the main paper and Sec. B), *i.e.* after training on purely synthetically generated image warps, on which we overlaid moving objects and our flow perturbations (Sec. 3.4 of the main paper). For fair comparison, we compare to self-supervised approaches that also only rely on synthetic image warps, namely DGC-Net [40], GLU-Net [64] and GLU-Net-GOCor [64, 63]. We also include our baseline, the non probabilistic model GLU-Net-GOCor* trained on the same data. Results on multiple datasets are presented in Tab. 7. Our approach PDC-Net outperforms all other self-supervised methods, particularly in terms of accuracy (PCK and Fl). Notably, our probabilistic modeling of the problem improves the learning, leading to large gains in flow estimation accuracy, as evidenced by comparing PDC-Net to non probabilistic baseline GLU-Net-GOCor*. Also note

		RANSAC-Flow	GLU-Net-GOCor*	PDC-Net (Ours)
interval = 3	PCK-1px [%]	62.62	64.16	**66.05**
	PCK-3px [%]	77.41	78.89	**79.13**
	PCK-5px [%]	91.66	92.73	**92.95**
interval = 5	PCK-1px [%]	59.18	59.79	**62.48**
	PCK-3px [%]	75.85	77.33	**77.71**
	PCK-5px [%]	90.85	92.01	**92.32**
interval = 7	PCK-1px [%]	55.95	55.62	**58.96**
	PCK-3px [%]	74.32	75.85	**76.38**
	PCK-5px [%]	89.96	91.24	**91.67**
interval = 9	PCK-1px [%]	53.03	52.04	**56.05**
	PCK-3px [%]	73.04	74.58	**75.30**
	PCK-5px [%]	89.20	90.55	**91.11**
interval = 11	PCK-1px [%]	50.48	48.61	**53.29**
	PCK-3px [%]	71.86	73.17	**74.22**
	PCK-5px [%]	88.47	89.74	**90.55**
interval = 13	PCK-1px [%]	48.23	45.74	**50.95**
	PCK-3px [%]	70.85	72.01	**73.35**
	PCK-5px [%]	87.76	88.95	**89.98**
interval = 15	PCK-1px [%]	46.04	43.01	**48.70**
	PCK-3px [%]	69.75	70.48	**72.36**
	PCK-5px [%]	86.97	87.81	**89.30**

Table 8. Metrics evaluated over scenes of ETH3D with different intervals between consecutive pairs of images (taken by the same camera). Note that, in this work, we computed the PCK over the whole dataset (image pairs of all the sequences) per interval, to be consistent with evaluations on RobotCar and MegaDepth introduced in [53]. This metric is different than the one originally used by [64, 63] for ETH3D, where the PCKs were calculated per image instead and averaged per sequence. High PCK is better.

	KITTI-2015			MegaDepth			YFCC100M	
	EPE	Fl (%)	AUSE	PCK-1 (%)	PCK-5 (%)	AUSE	mAP @5°	mAP @10°
Uncertainty not propagated between levels	6.76	**31.84**	0.212	29.9	65.13	0.213	31.50	42.19
PDC-Net-s	**6.66**	32.32	**0.205**	**32.51**	**66.50**	**0.197**	**33.77**	**45.17**
$M = 2; 0.0 < \sigma_1^2 < 1.0, 2.0 < \sigma_2^2 < \infty$	6.69	32.58	**0.181**	32.47	65.45	0.205	30.50	40.75
$M = 2; \sigma_1^2 = 1.0, 2.0 < \sigma_2^2 < \infty$ (PDC-Net-s)	6.66	32.32	0.205	**32.51**	66.50	**0.197**	**33.77**	**45.17**
$M = 2; \sigma_1^2 = 1.0, 2.0 < \sigma_2^2 < \beta_2^+ = s^2$	**6.61**	**31.67**	0.208	31.83	**66.52**	0.204	33.05	44.48
$M = 2; \sigma_1^2 = 1.0, 2.0 < \sigma_2^2 < \beta_2^+ = s^2$	6.61	31.67	**0.208**	31.83	**66.52**	**0.204**	33.05	44.48
$M = 3; \sigma_1^2 = 1.0, 2.0 < \sigma_2^2 < \beta_2^+ = s^2, \sigma_3^2 = s^2$	**6.41**	**30.54**	0.212	**31.89**	66.10	0.214	**34.90**	**45.86**

Table 9. Ablation study. For all methods, we model the flow as a constrained mixture of Laplace distributions (Sec. 3.2 of the main paper), and we use our uncertainty prediction architecture (Sec. 3.3 of the main paper). In the top part, we show the impact of propagating the uncertainty estimates in a multi-scale architecture. In the second part, we compare different parametrization of the constrained mixture (Sec. 3.2 of the main paper, mostly equation (3)). Here, s refers to the image size used during training, *i.e.* $s = 256$. In the bottom part, we analyse the impact of the number of components M used in the constrained mixture model.

that for a *single* forward pass, PDC-Net only increases inference time by 14.3% over baseline GLU-Net-GOCor* for drastically better results.

Moreover, for the non probabilistic methods, we computed the relative poses on YFCC100M using *all* estimated dense correspondences and RANSAC. As previously stated, the poor results emphasize the necessity to infer a confidence prediction along with the dense flow prediction, in order to be able to use the estimated matches for down-stream tasks.

E.4. Qualitative results

Here, we first present qualitative results of our approach PDC-Net on the KITTI-2015 dataset in Fig. 13. PDC-Net clearly identifies the independently moving objects, and does very well in static scenes with only a single moving object, which are particularly challenging since not represented in the training data.

In Fig. 16 and Fig. 17, 18, 19, we qualitatively compare our approach PDC-Net to the baseline GLU-Net-GOCor* on images of the RobotCar and the Megadepth datasets respectively. We additionally show the performance of our uncertainty estimation on these examples. By overlaying the warped query image with the reference image at the locations of the identified accurate matches, we observe that our method produces *highly precise correspondences*. Our uncertainty estimates successfully identify accurate flow regions and also correctly exclude in most cases homogeneous and sky regions. These examples show the benefit of confidence estimation for high quality image alignment, useful *e.g.* in multi-frame super resolution [69]. Texture or style transfer (*e.g.* for AR) also largely benefit from it.

In Fig. 15, we visually compare the estimated confidence maps of RANSAC-Flow [53] and our approach PDC-Net on the YFCC100M dataset. Our confidence maps can accurately *segment* the object from the background (sky). On the other hand, RANSAC-Flow predicts confidence maps, which do not exclude unreliable matching regions, such as the sky. Using these regions for pose estimation for example, would result in a drastic drop in performance, as evidenced in Tab. 3 of the main paper. Note also the ability of our predicted confidence map to identify *small accurate flow regions*, even in a dominantly failing flow field. This is the case in the fourth example from the top in Fig. 15.

F. Detailed ablation study

Finally, we provide detailed ablative experiments. As in Sec. 4.5 of the main paper, we use BaseNet as base network to create the probabilistic models, as described in Sec. C. Similarly, all networks are trained on solely the first training stage.

Uncertainty architecture, sparsification plots: For completeness, we present the full sparsification error plots for

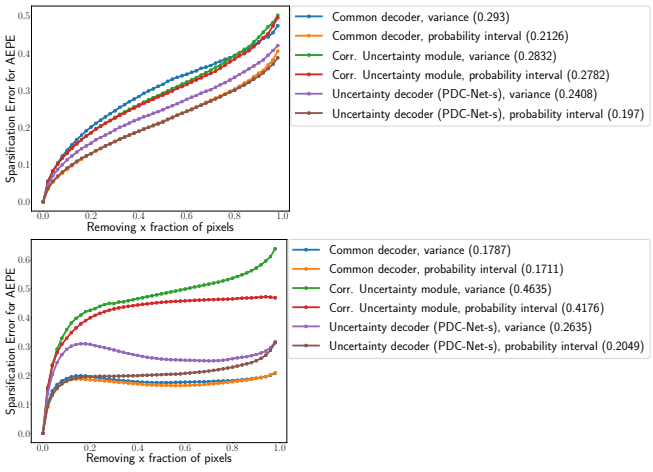


Figure 12. Sparsification Error plots for AEPE on MegaDepth (top) and KITTI-2015 (bottom), for different uncertainty decoder architecture, and using either the variance V or our probability interval $P_{R=1}$ as confidence measure. Smaller AUSE (in parenthesis) is better. All networks are modelled with a constrained mixture of Laplace, and trained only on the first stage (Sec. 4.1 of the main paper).

different uncertainty prediction architectures in Fig. 12. They correspond to Tab. 4, middle part of the main paper.

Confidence value, variance against probability of interval: We here compare using the variance of the constrained mixture probability density $V = \sum_{m=1}^{M} \alpha_m \sigma_m^2$, or the probability of the confidence interval P_R (Sec. 3.5 of the main paper), as a final pixel-wise confidence value, associated with the predicted flow field. In Fig. 12, for each of the compared uncertainty prediction architecture, we further compute the sparsification error plots, using either the inverse of the variance $1/V$, or the probability $P_{R=1}$, as confidence measure. On both MegaDepth and KITTI-2015, the probability of the confidence interval $P_{R=1}$ appears as a better performing measure of the uncertainty.

Propagation of uncertainty components: In a multi-scale network architecture, the predicted uncertainty parameters of the mixture at a particular network level can be further propagated to the next level. We use this strategy, by feeding the predicted uncertainty components of the previous level to the flow decoder and to the uncertainty predictor of the current level. In Tab. 9 top part, we show the impact of this uncertainty propagation. We compare our approach PDC-Net-s with multi-scale uncertainty propagation, to a network where uncertainty estimation at each level is done independently of the previous one. For all presented datasets and metrics, propagating the uncertainty predictions boosts the performance of the final network. Only the Fl metric on KITTI-2015 is slightly worst.

Constrained mixture parametrization: In Tab. 9, middle part, we then compare different parametrization for the constrained mixture of Laplace distributions. In our final network, we fixed the first component σ_1^2 of the mixture, as $\sigma_1^2 = \beta_1^- = \beta_1^+ = 1.0$.

Firstly, we compare with a version where the first component is constrained instead as $\beta_1^- = 0.0 \leq \sigma_1^2 \leq \beta_1^+ = 1.0$. Both networks obtain similar flow performance results on KITTI-2015 and MegaDepth. Only on optical flow dataset KITTI-2015, the second alternative of $\beta_1^- = 0.0 \leq \sigma_1^2 \leq \beta_1^+ = 1.0$ obtains a better AUSE, which is explained by the fact that KITTI-2015 shows ground-truth displacements with a much smaller magnitude than in the geometric matching datasets. When estimating the flow on KITTI, it thus results in a larger proportion of very small flow errors (lower EPE and higher PCK than on geometric matching data). As a result, on this dataset, being able to model very small error (with $\sigma_1^2 \leq 1$) is beneficial. However, fixing $\sigma_1^2 = 1.0$ instead produces better AUSE on MegaDepth and it gives significantly better results for pose estimation on YFCC100M. As previously explained in Sec. C.2, fixing $\sigma_1^2 = 1$ enables for the network to equally focus on getting accurate correspondences (bounded by the fixed σ_1^2) and improving the inaccurate flow regions during training. It can be seen as introducing an additional prior constraint on the distribution.

We then compare leaving the second component's higher bound unconstrained, as $2.0 \leq \sigma_2^2 \leq \infty$ (PDC-Net-s) to constraining it, as $2.0 \leq \sigma_2^2 \leq \beta_2^+ = s^2$, where s refers to the image size used during training. All results are very similar, the fully constrained network obtains slightly better flow results but slightly worst uncertainty performance (AUSE). However, we found that constraining β_2^+ leads to a more stable training in practise, which is why we adopted the constrains $\sigma_1^2 = \beta_1^- = \beta_1^+ = 1$, and $2.0 = \beta_2^- \leq \sigma_2^2 \leq \beta_2^+ = s^2$ for our final network.

Number of components of the constrained mixture: Finally, we compare $M = 2$ and $M = 3$ Laplace components used in the constrained mixture in Tab. 9, bottom part. In the case of $M = 3$, the first two components are set similarly to the case $M = 2$, *i.e.* as $\sigma_1^2 = 1.0$ and $2.0 \leq \sigma_2^2 \leq \beta_2^+ = s^2$ where β_2^+ is fixed to the image size used during training (256 here). The third component is set as $\sigma_3^2 = \beta_3^+ = \beta_3^- = \beta_2^+$. The aim of this third component is to identify outliers (such as out-of-view pixels) more clearly. The 3 components approach obtains a better Fl value on KITTI-2015 and slightly better pose estimation results on YFCC100M. However, its performance on MegaDepth and in terms of pure uncertainty estimation (AUSE) slightly degrade. As a result, for simplicity we adopted the version with $M = 2$ Laplace components. Note, however, that more components could easily be added.

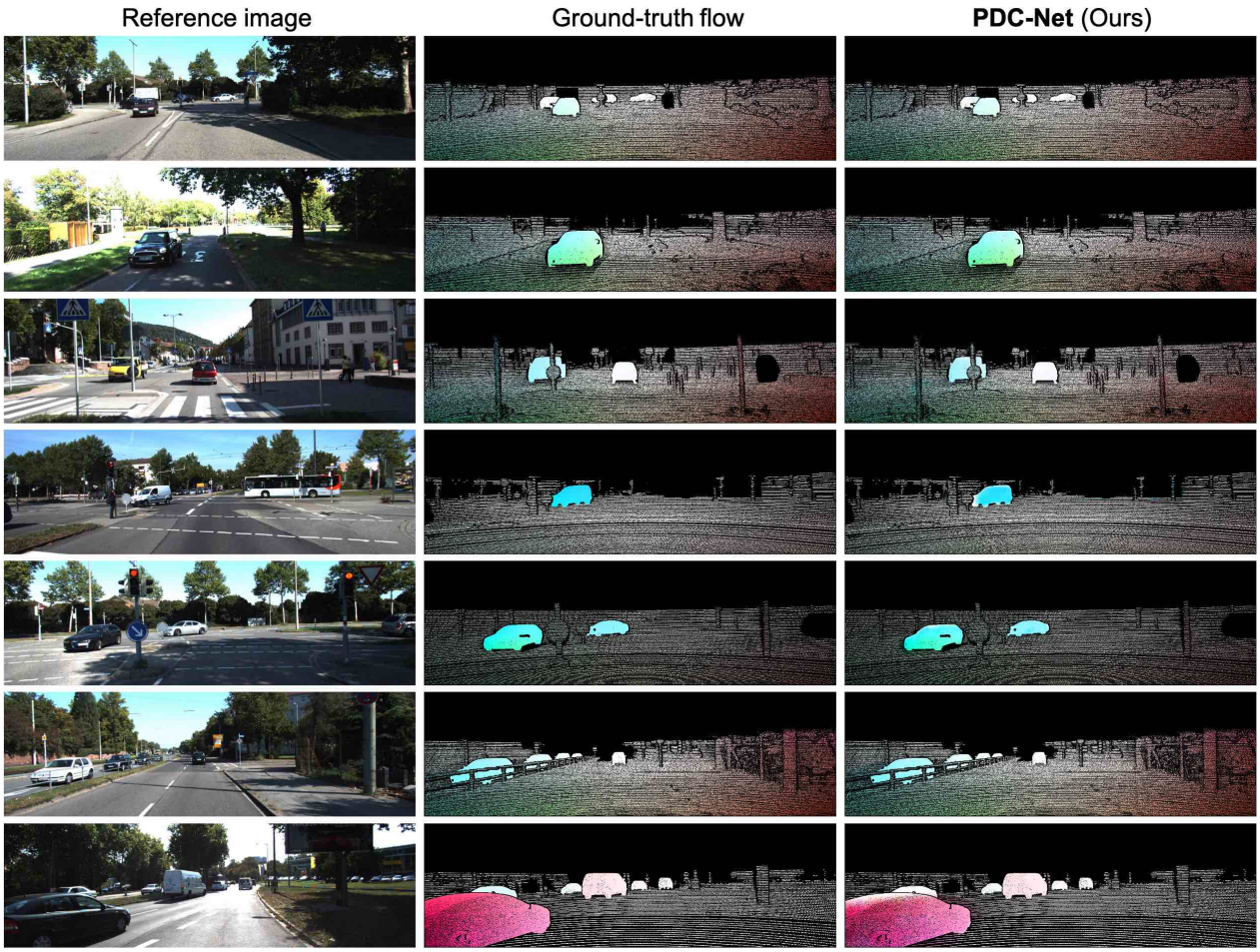


Figure 13. Qualitative examples of our approach PDC-Net applied to images of KITTI-2015. We plot directly the estimated flow field for each image pair.

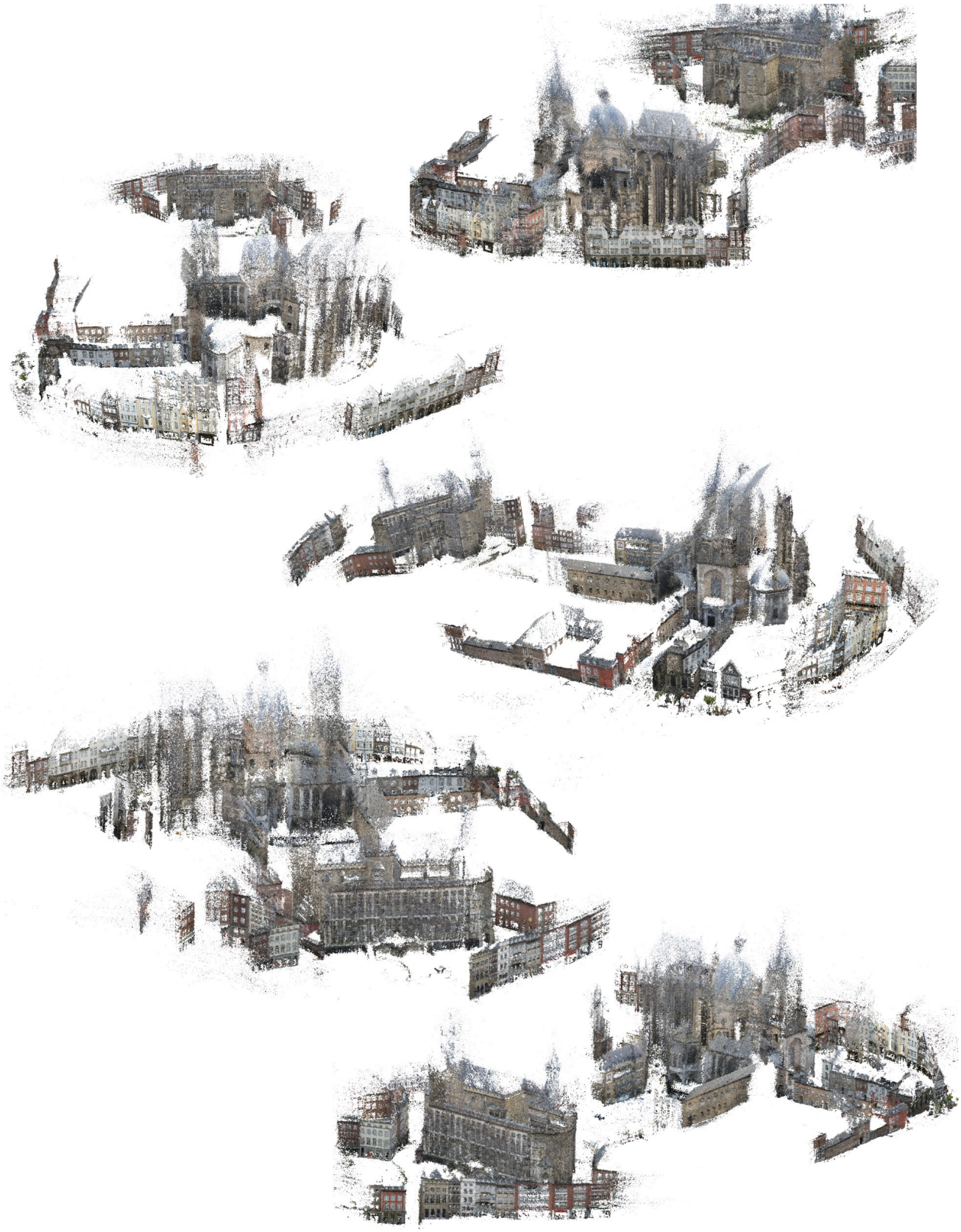


Figure 14. Visualization of the 3D reconstruction of Aachen city.

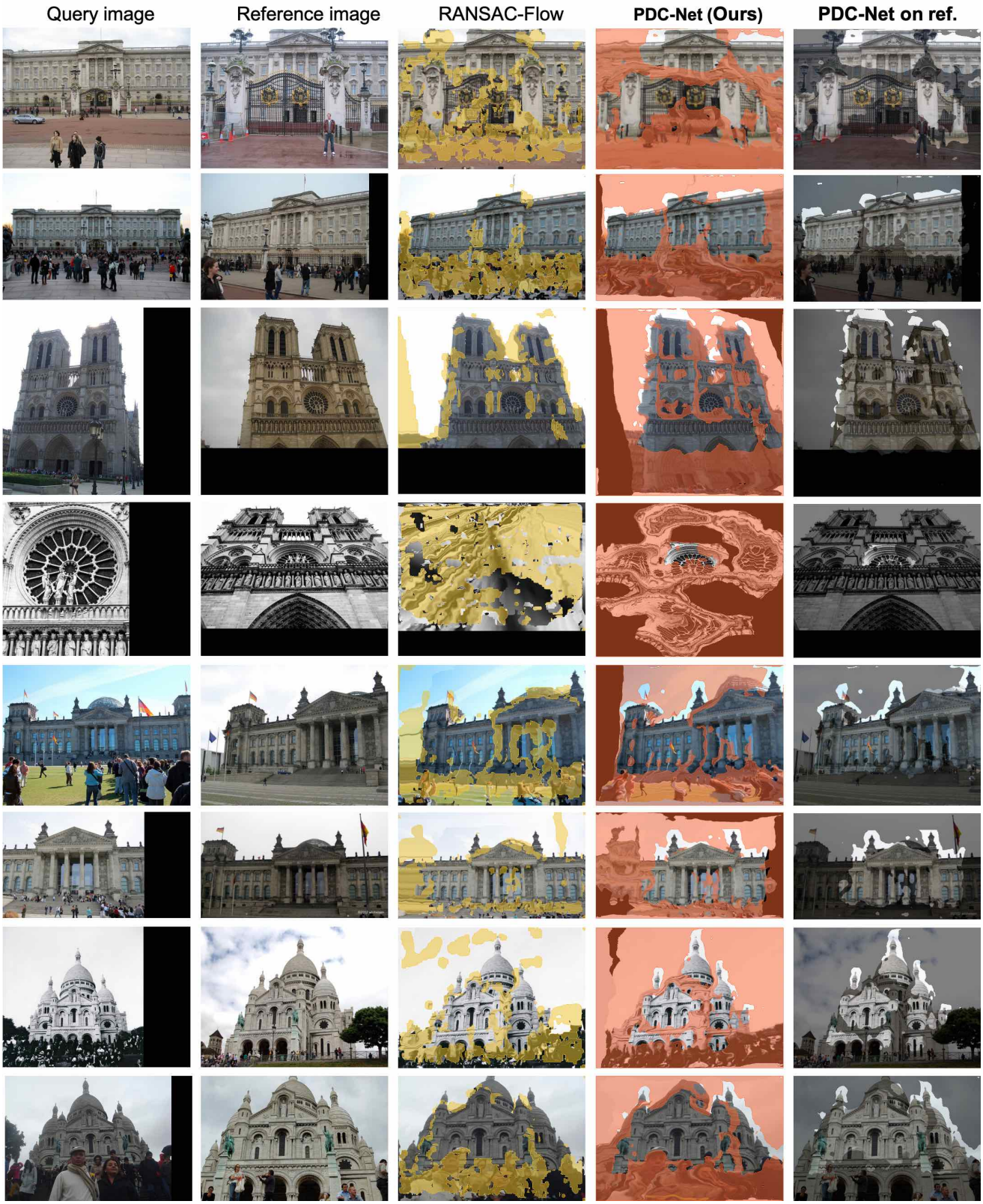


Figure 15. Visual comparison of RANSAC-Flow and our approach PDC-Net on image pairs of the YFCC100M dataset [61]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the RANSAC-Flow and PDC-Net respectively. Both networks also predict a confidence map, according to which the regions represented in respectively yellow and red, are unreliable or inaccurate matching regions. In the last column, we overlay the reference image with the warped query from PDC-Net, in the identified accurate matching regions (lighter color).

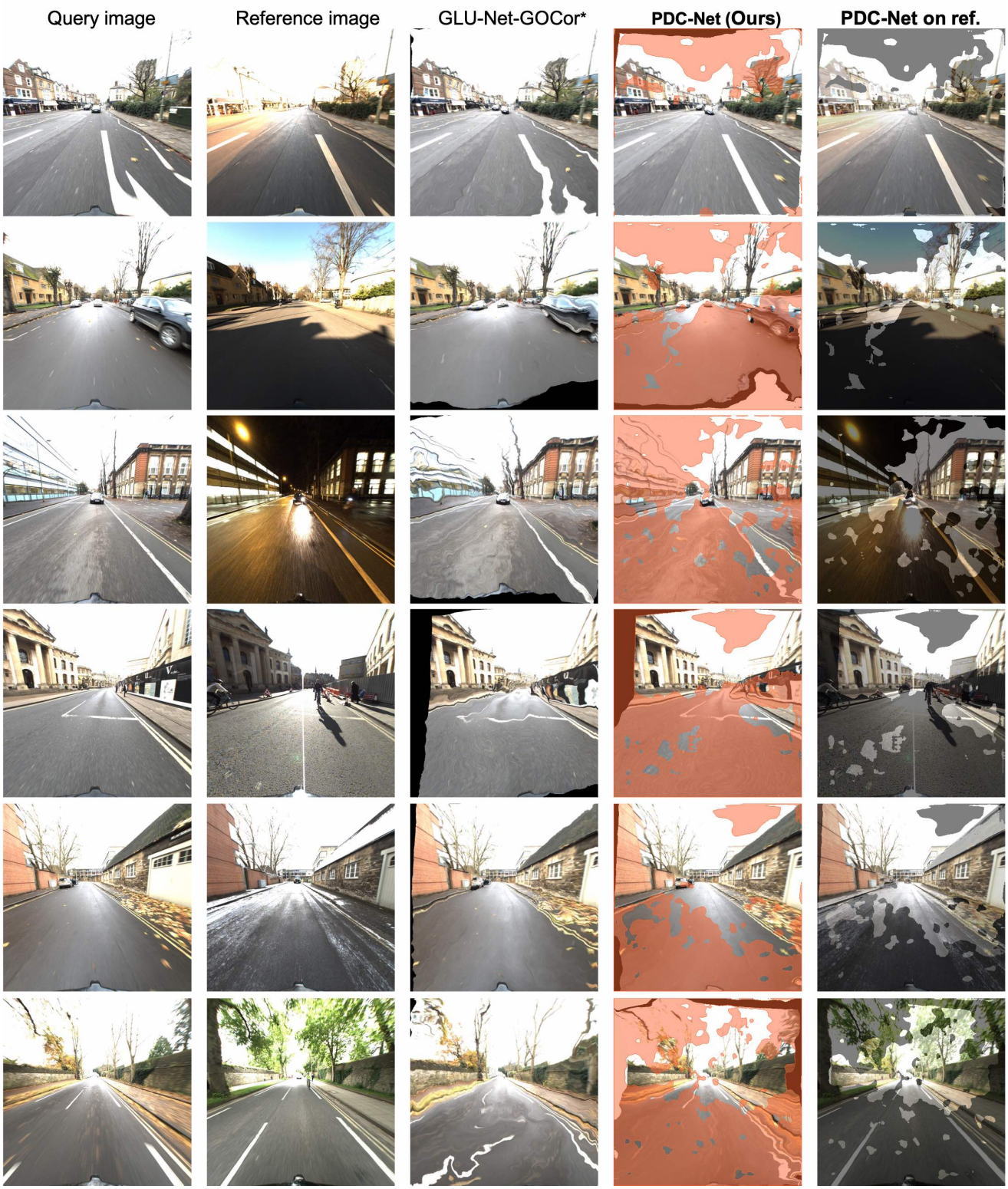


Figure 16. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseling GLU-Net-GOCor*, applied to images of the RobotCar dataset [33]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the GLU-Net-GOCor* and PDC-Net respectively. PDC-Net also predicts a confidence map, according to which the regions represented in red, are unreliable or inaccurate matching regions. In the last column, we overlay the reference image with the warped query from PDC-Net, in the identified accurate matching regions (lighter color).

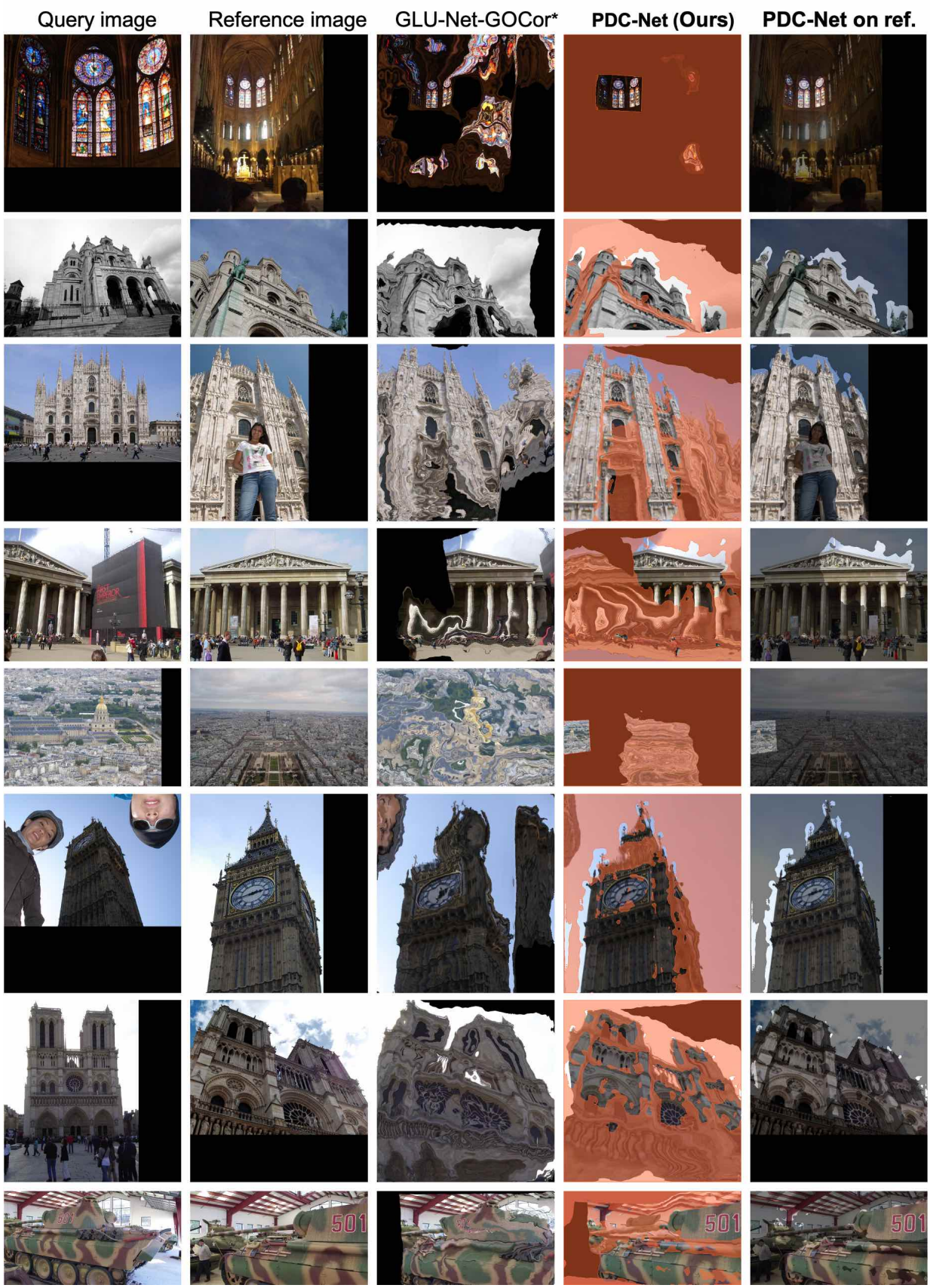


Figure 17. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseline GLU-Net-GOCor*, applied to images of the MegaDepth dataset [34]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the GLU-Net-GOCor* and PDC-Net respectively. PDC-Net also predicts a confidence map, according to which the regions represented in red, are unreliable or inaccurate matching regions. In the last column, we overlay the reference image with the warped query from PDC-Net, in the identified accurate matching regions (lighter color).

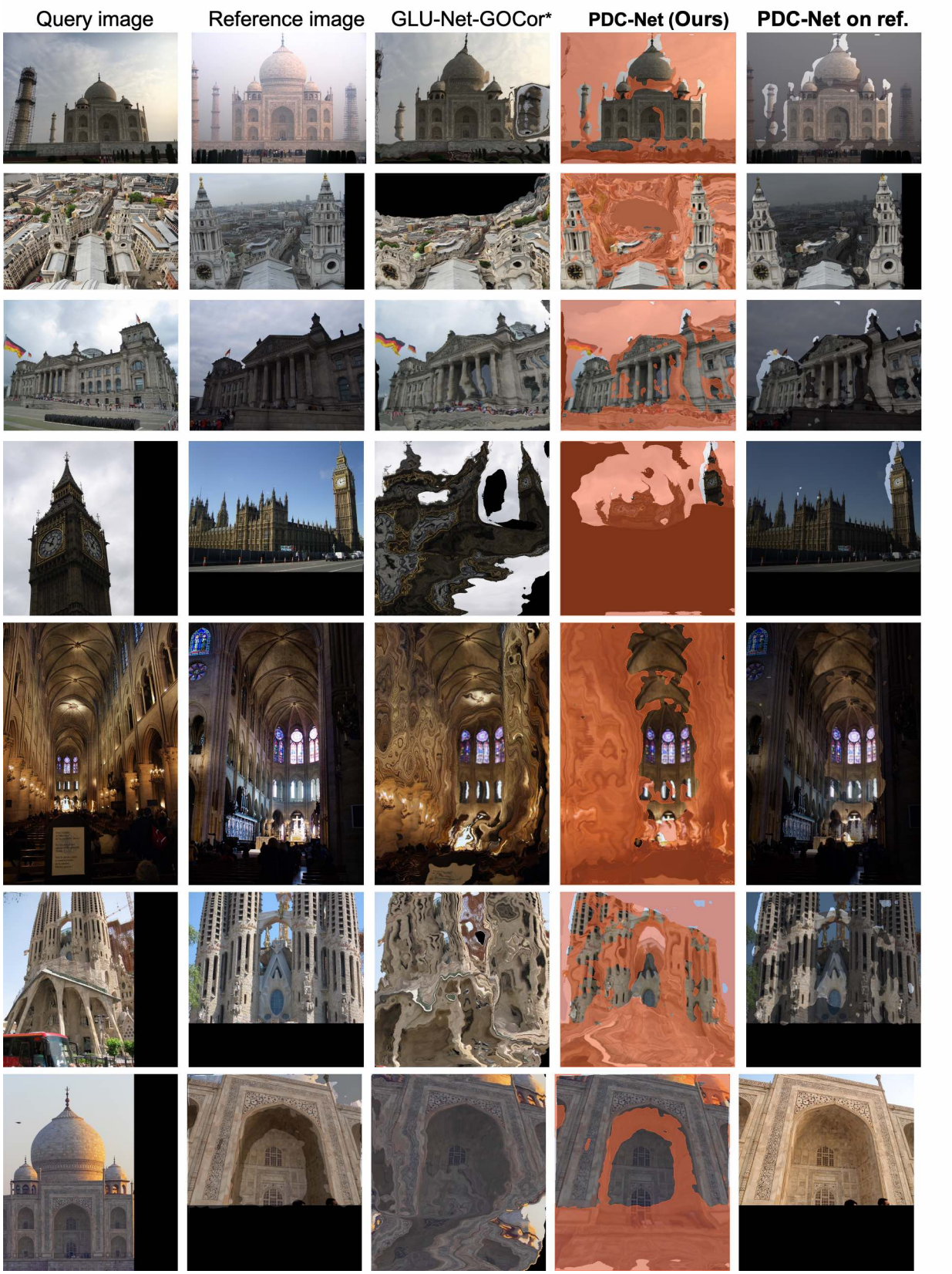


Figure 18. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseline GLU-Net-GOCor*, applied to images of the MegaDepth dataset [34]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the GLU-Net-GOCor* and PDC-Net respectively. PDC-Net also predicts a confidence map, according to which the regions represented in red, are unreliable or inaccurate matching regions. In the last column, we overlay the reference image with the warped query from PDC-Net, in the identified accurate matching regions (lighter color).

Figure 19. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseline GLU-Net-GOCor*, applied to images of the MegaDepth dataset [34]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the GLU-Net-GOCor* and PDC-Net respectively. PDC-Net also predicts a confidence map, according to which the regions represented in red, are unreliable or inaccurate matching regions. In the last column, we overlay the reference image with the warped query from PDC-Net, in the identified accurate matching regions (lighter color).