Learning Accurate Dense Correspondences and When to Trust Them Appendix

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

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,

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

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,

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

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 (2) simplifies, and when inserting into (1), we obtain,

$$p\left(y|\varphi\right) = \sum_{m=1}^{M} \alpha_m \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}. \tag{3}$$

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,

$$\begin{aligned}
P_R &= P(\|y-\mu\|_\infty < R) && \text{(4a)}\\
&= \int_{\{y\in\mathbb{R}^2 : \|y-\mu\|_\infty < R\}} p(y|\varphi) dy && \text{(4b)}\\
&= \sum_m \alpha_m \int_{\mu_u-R}^{\mu_u+R} \frac{1}{\sqrt{2}\sigma_m} e^{-\sqrt{2}\frac{|u-\mu_u|}{\sigma_m}} du \\
&\quad \int_{\mu_v-R}^{\mu_v+R} \frac{1}{\sqrt{2}\sigma_m} e^{-\sqrt{2}\frac{|v-\mu_v|}{\sigma_m}} dv && \text{(4c)}\\
&= \sum_m \alpha_m \left[1 - \exp(-\sqrt{2}\frac{R}{\sigma_m})\right]^2 && \text{(4d)}
\end{aligned}$$

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 1, 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

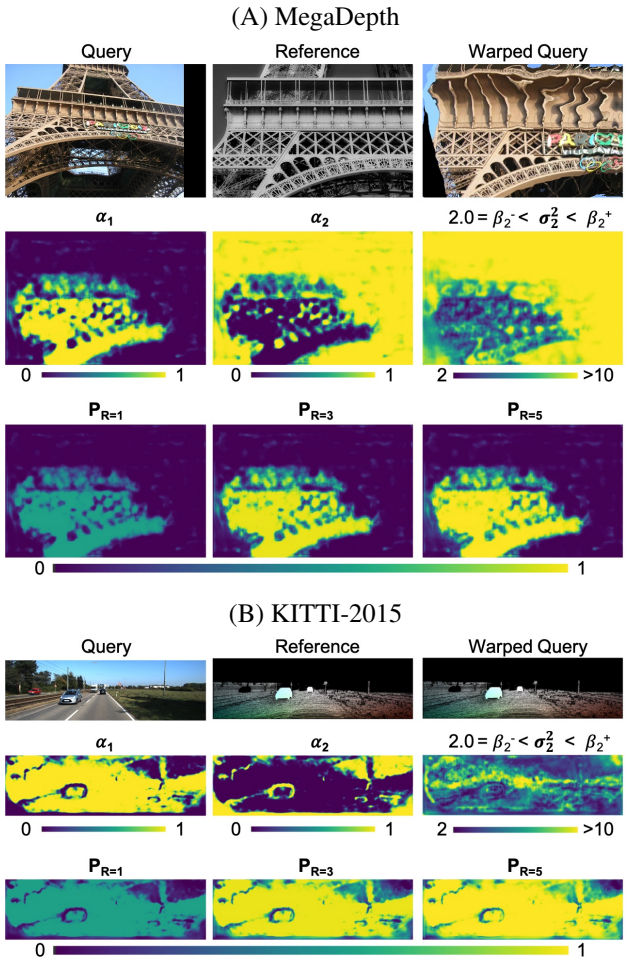

Figure 1. Visualization of the mixture parameters $(\alpha_m)_{m=1}^M$ and σ_2^2 predicted by our final network PDC-Net, on multiple image pairs. PDC-Net has $M = 2$ Laplace components and here, we do not represent the scale parameter σ_1^2, since it is fixed as $\sigma_1^2 = 1.0$. We also show the resulting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

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,

$$L(\theta; \mathcal{D}) = -\frac{1}{N}\sum_{n=1}^{N} \log p\left(Y^{(n)} | \Phi(X^{(n)}; \theta)\right) \tag{5a}$$

$$= -\frac{1}{N}\sum_{n=1}^{N}\sum_{ij} \log p\big(y_{ij}^{(n)} | \varphi_{ij}(X^{(n)}; \theta)\big) \tag{5b}$$

Inserting (3) into (5), we obtain for the last term the following expression,

$$L_{ij} = -\log p\big(y_{ij}^{(n)} | \varphi_{ij}(X^{(n)}; \theta)\big) \tag{6a}$$

$$= -\log\left(\sum_{m=1}^{M} \alpha_m \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}\right) \tag{6b}$$

$$= -\log\left(\sum_{m=1}^{M} \frac{e^{\tilde{\alpha}_m}}{\sum_{m=1}^{M} e^{\tilde{\alpha}_m}} \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}\right) \tag{6c}$$

$$= \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m}\right) - \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m} \frac{1}{2\sigma_m^2} e^{-\sqrt{\frac{2}{\sigma_m^2}}|y-\mu|_1}\right) \tag{6d}$$

$$= \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m}\right) - \log\left(\sum_{m=1}^{M} e^{\tilde{\alpha}_m - \log(2) - s_m - \sqrt{2}e^{-\frac{1}{2}s_m}.|y-\mu|_1}\right) \tag{6e}$$

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 (6) 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 [13].

Base synthetic dataset: For our base synthetic dataset, we use the same data as in [25]. Specifically, pairs of images are created by warping a collection of images from the

Figure 2. Visualization of our perturbations applied to a pair of reference and query images (Sec. 3.4 of the main paper).

DPED [9], CityScapes [3] and ADE-20K [29] datasets, according to synthetic affine, TPS and homography transformations. The transformation parameters are the ones originally used in DGC-Net [16].

These image pairs are further augmented with additional random independently moving objects. To do so, the objects are sampled from the COCO dataset [14], 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 $\epsilon = \sum_i \varepsilon_i$, 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 [23]. 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 [1]. 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. 2, 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 [4]. Specifically, we use the MegaDepth dataset, consisting of 196 different scenes reconstructed from 1.070.468 internet photos using COLMAP [20]. 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. 1- 2, 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:

$$\left((\tilde{\alpha}_m)_{m=1}^M, (h_m)_{m=1}^M\right)^l = Q_\theta\left(Y^l; u^l; \Phi^{l-1}\right)_{ij} \quad (7)$$

where $\tilde{\alpha}_m$ refers to the output of the uncertainty predictor, 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 [26, 25] 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

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 1. 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 2. 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.

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 [8] 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 [26, 25], 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 [22], 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 [25] and inspired by GLU-Net [26]. It estimates the dense flow field relating an input image pair. The network is composed of three pyramid levels and it uses VGG-16 [2] 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 [26]. However, here, to reduce the number of weights,

we use feed-forward layers instead of DenseNet connections [8] 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-Net-s, 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.

$$\mathcal{L}(\theta) = \sum_{l=1}^{K} \gamma_l L_l + \eta \left\| \theta \right\| , \tag{8}$$

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 (5) for the probabilistic models, including our approach PDC-Net. The second term of the loss (8) 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 [26]. Specifically, as a preprocessing step, the training images are mean-centered and normalized using mean and standard deviation of ImageNet dataset [11]. 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 (8) 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 (8) 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 (8). 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 (8) 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 [25], we train with 3 local and global optimization iterations.

Our system is implemented using Pytorch [18] and our networks are trained using Adam optimizer [10] 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.

F1: F1 designates the percentage of outliers averaged over all valid pixels of the dataset [6]. They are defined as follows, where Y indicates the ground-truth flow field and $\hat{Y}$ the estimated flow by the network.

$$F1 = \frac{\left\| Y - \hat{Y} \right\| > 3 \text{ and } \frac{\left\| Y - \hat{Y} \right\|}{\|Y\|} > 0.05}{\text{\#valid pixels}} \tag{9}$$

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 [28]. 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

$$R_{err} = cos^{-1} \frac{Tr(R^{-1}\hat{R}) - 1}{2} , \tag{10}$$

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 $\hat{T}$.

$$T_{err} = cos^{-1} \frac{T \cdot \hat{T}}{\|T\| \left\| \hat{T} \right\|} , \tag{11}$$

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 [28], *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 [22]. 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 [22], all the images are resized to have minimum dimension 480 pixels.

RobotCar: In RobotCar, we used the correspondences originally introduced by [15]. During evaluation, following [22], all the images are resized to have minimum dimension 480 pixels.

ETH3D: The Multi-view dataset ETH3D [21] 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 [20]) 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 per interval, to be consistent with RANSAC-Flow. This metric is different than the one originally used by [26, 25] for ETH3D, where the PCK was calculated per image instead.

KITTI: The KITTI dataset [6] 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 [7]. 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

	MegaDepth			RobotCar			KITTI-2012		KITTI-2015		YFCC100M		
	PCK-1	PCK-3	PCK-5	PCK-1	PCK-3	PCK-5	AEPE ↓	F1 (%) ↓	AEPE ↓	F1 (%) ↓	mAP @5°	mAP @10°	mAP @20°
DGC-Net [16]	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 [26]	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 [25]	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 3. 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.

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 [5] and 5-pt Nister algorithm [17], 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 [22], and the intrinsic camera parameters are modified accordingly.

3D reconstruction on Aachen dataset: We use the set-up of [19], 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 [20]. 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. 6.

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 [25] respectively. We reported results of RANSAC-Flow [22] 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 [16] and RANSAC-Flow [22]. Fig. 3 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. 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 [16], GLU-Net [26] and GLU-Net-GOCor [26, 25]. 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. 3. Our approach PDC-Net outperforms all other self-supervised methods, particularly in terms of accuracy (PCK and F1). 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 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 com-

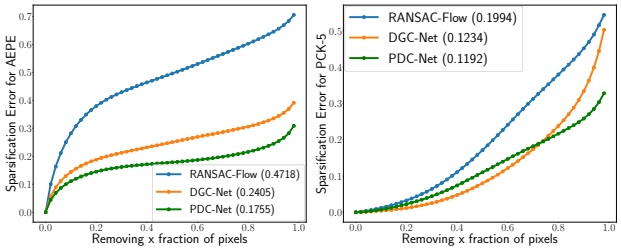

Figure 3. Sparsification Error plots for AEPE (left) and PCK-5 (right) on RobotCar. Smaller AUSE (in parenthesis) is better.

	KITTI-2015			MegaDepth			YFCC100M	
	EPE	F1 (%)	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 4. 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.

puted 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.3. Qualitative results

Here, we first present qualitative results of our approach PDC-Net on the KITTI-2015 dataset in Fig. 5. 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. 8 and Fig. 9, 10, 11, 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 [27]. Texture or style transfer (*e.g.* for AR) also largely benefit from it.

In Fig. 7, we visually compare the estimated confidence maps of RANSAC-Flow [22] 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. 7.

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 different uncertainty prediction architectures in Fig. 4. 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

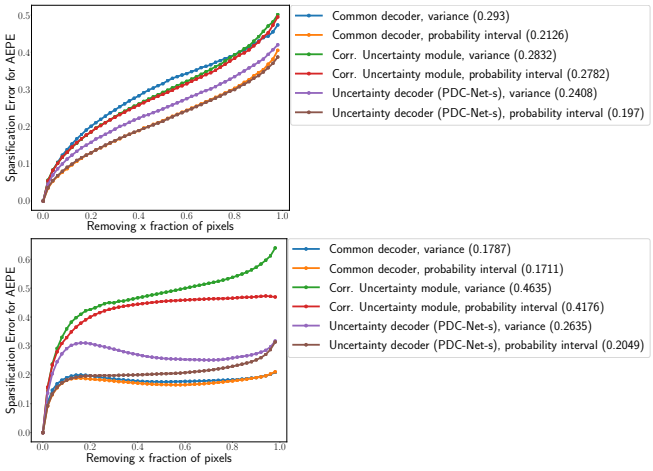

Figure 4. Sparsification Error plots for AEPE on MegaDepth (top) and KITTI-2015 (bottom), for different uncertainty decoder architecture, and using either the variance V or our probability interval $P_{R=1}$ as confidence measure. Smaller AUSE (in parenthesis) is better. All networks are modelled with a constrained mixture of Laplace, and trained only on the first stage (Sec. 4.1 of paper).

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. 4, 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. 4 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 F1 metric on KITTI-2015 is slightly worst.

Constrained mixture parametrization: In Tab. 4, 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 $\beta_1^- = 0.0 < \sigma_1^2 < \beta_1^+ = 1.0$. Both networks obtain similar flow performance results on KITTI-2015 and MegaDepth. Only on optical flow dataset KITTI-2015, the alternative of $\sigma_1^2 = \beta_1^- = \beta_1^+ = 1.0$ 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 $\sigma_1^2 < 1$) 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 $2.0 < \sigma_2^2 < \infty$ (PDC-Net-s) to constraining it, as $2.0 < \sigma_2^2 < \beta_2^+ = s^2$, 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. 4, 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 $2.0 < \sigma_2^2 < \beta_2^+ = s^2$ 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 F1 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.

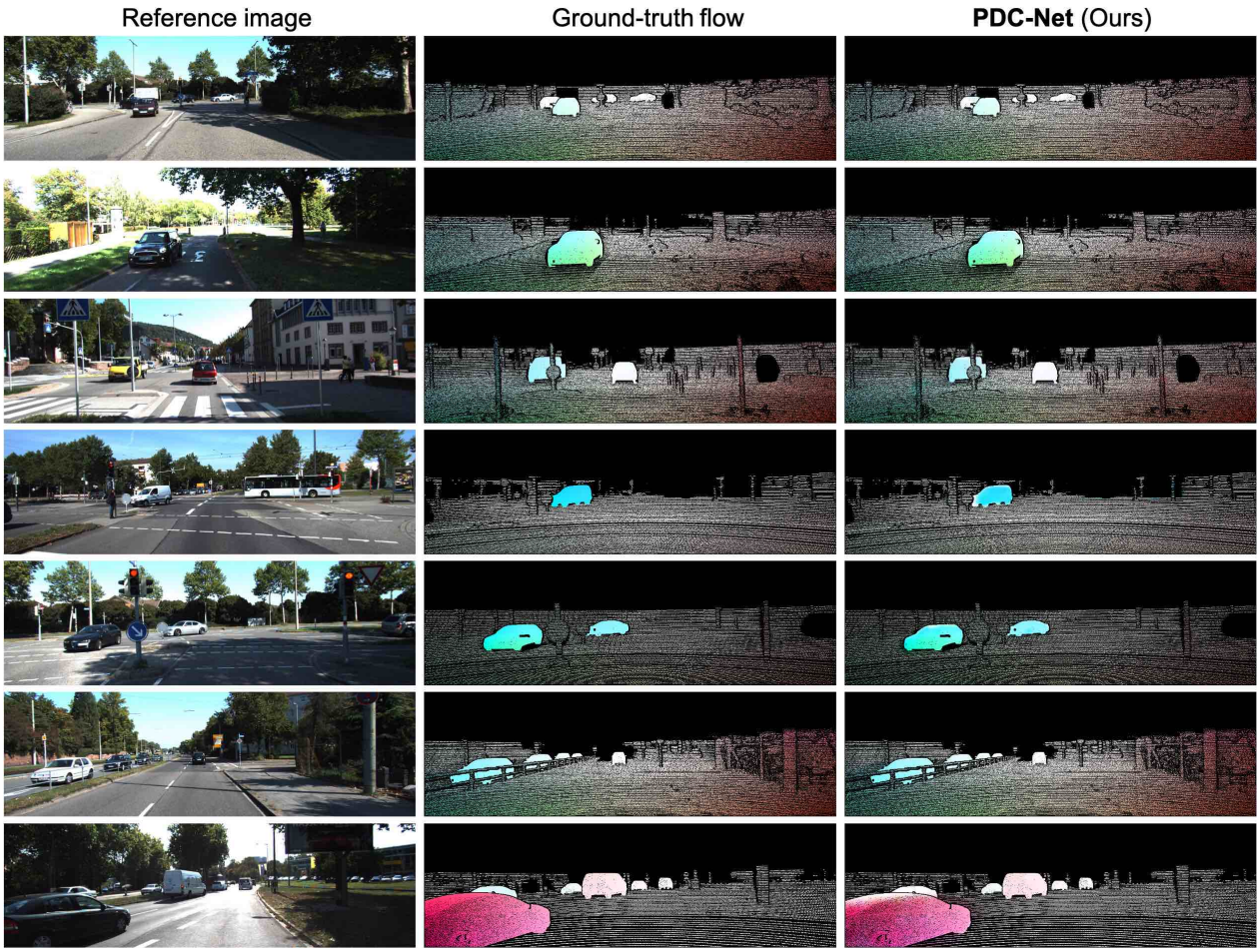

Figure 5. Qualitative examples of our approach PDC-Net applied to images of KITTI-2015. We plot directly the estimated flow field for each image pair.

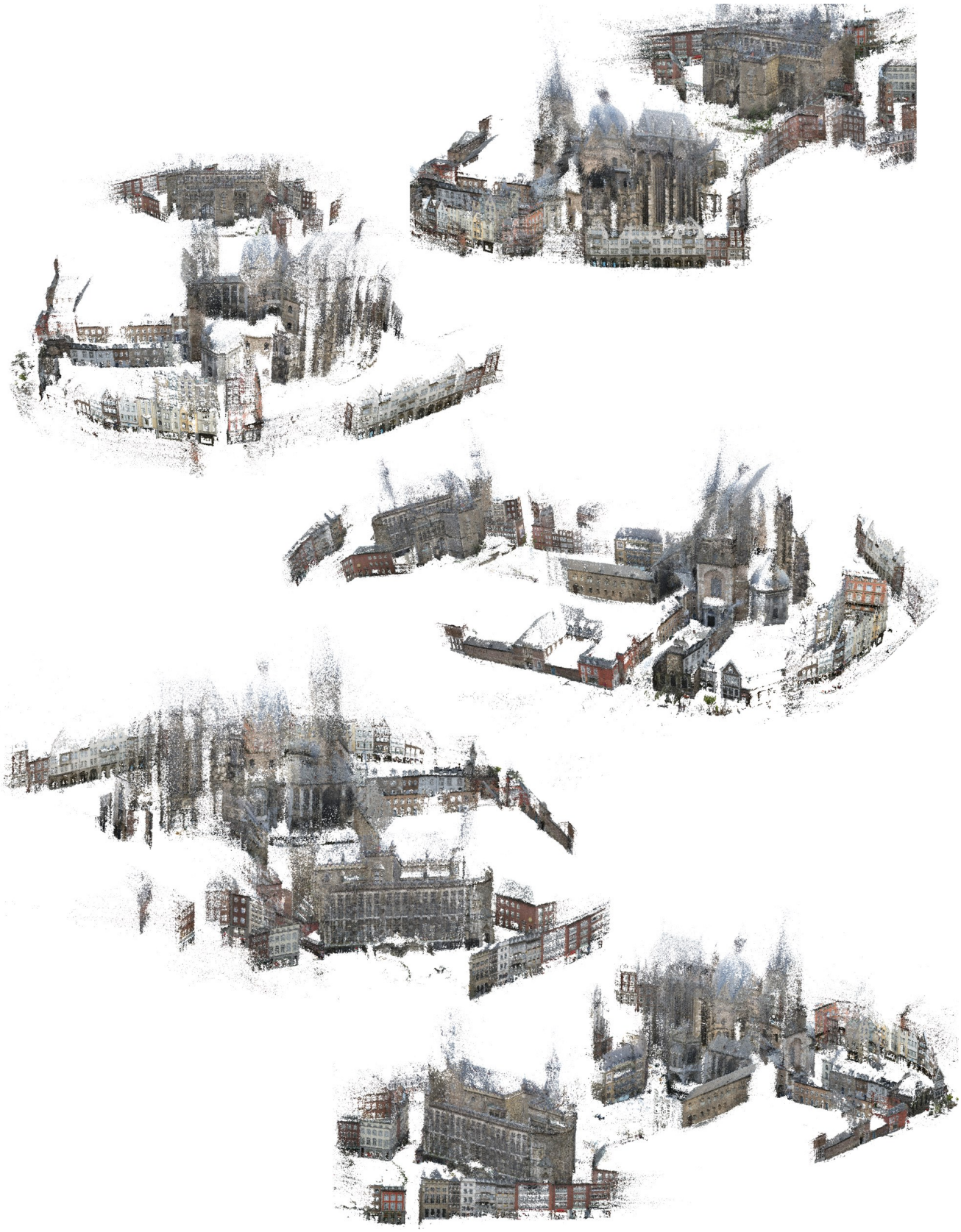

Figure 6. Visualization of the 3D reconstruction of Aachen city.

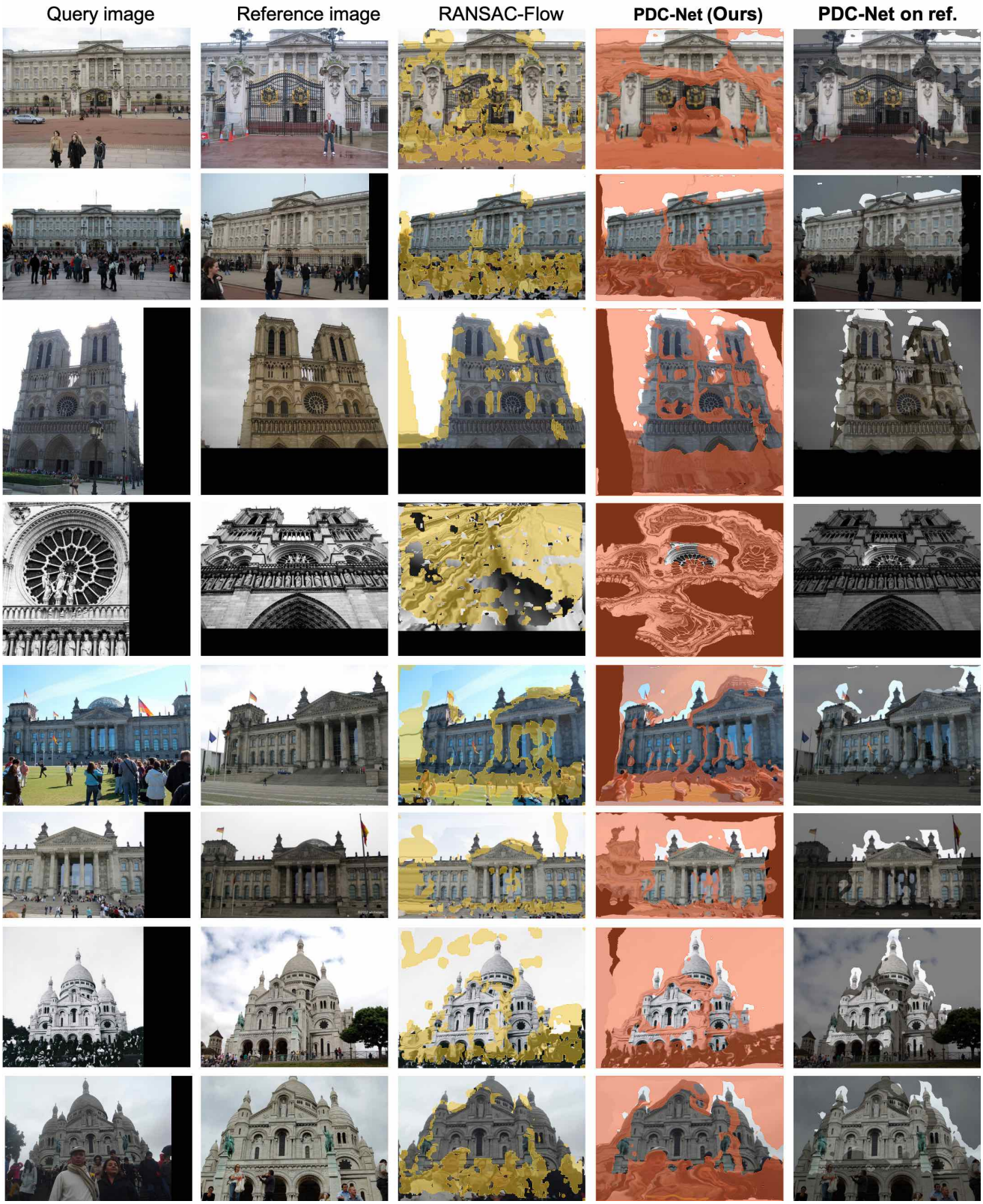

Figure 7. Visual comparison of RANSAC-Flow and our approach PDC-Net on image pairs of the YFCC100M dataset [24]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the RANSAC-Flow and PDC-Net respectively. Both networks also predict a confidence 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).

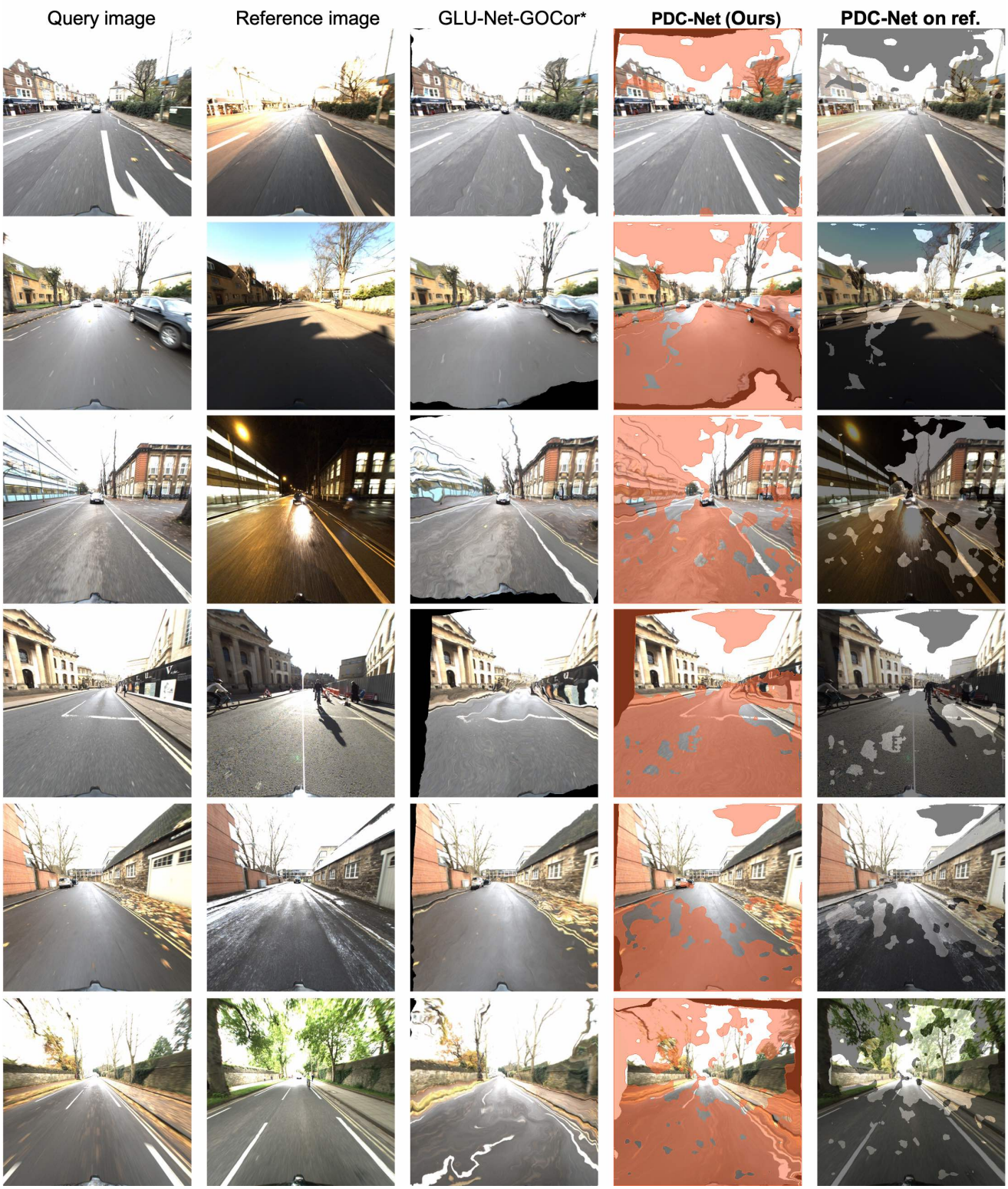

Figure 8. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseling GLU-Net-GOCor*, applied to images of the RobotCar dataset [12]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the GLU-Net-GOCor* and PDC-Net 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).

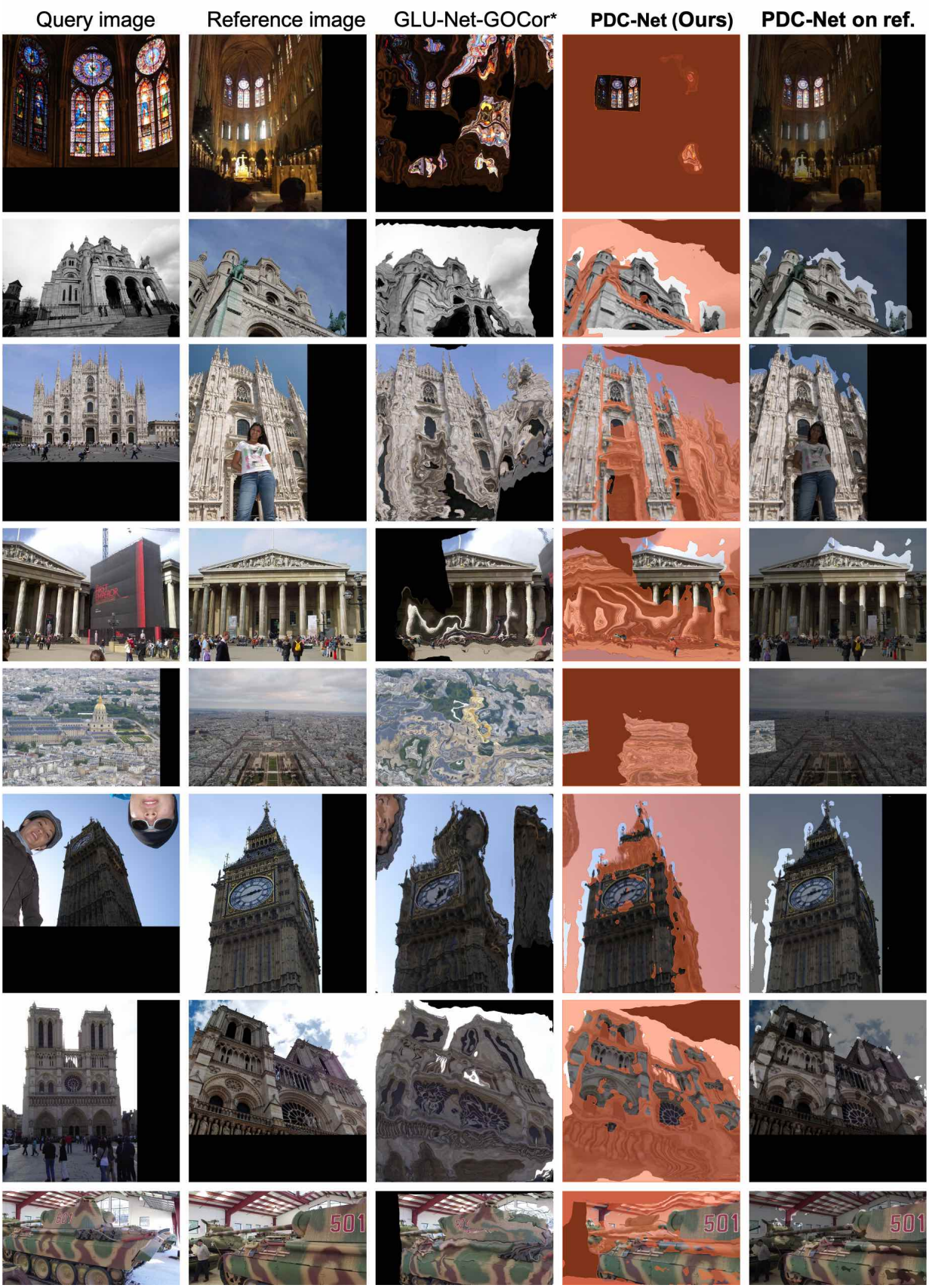

Figure 9. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseline GLU-Net-GOCor*, applied to images of the MegaDepth dataset [13]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the GLU-Net-GOCor* and 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).

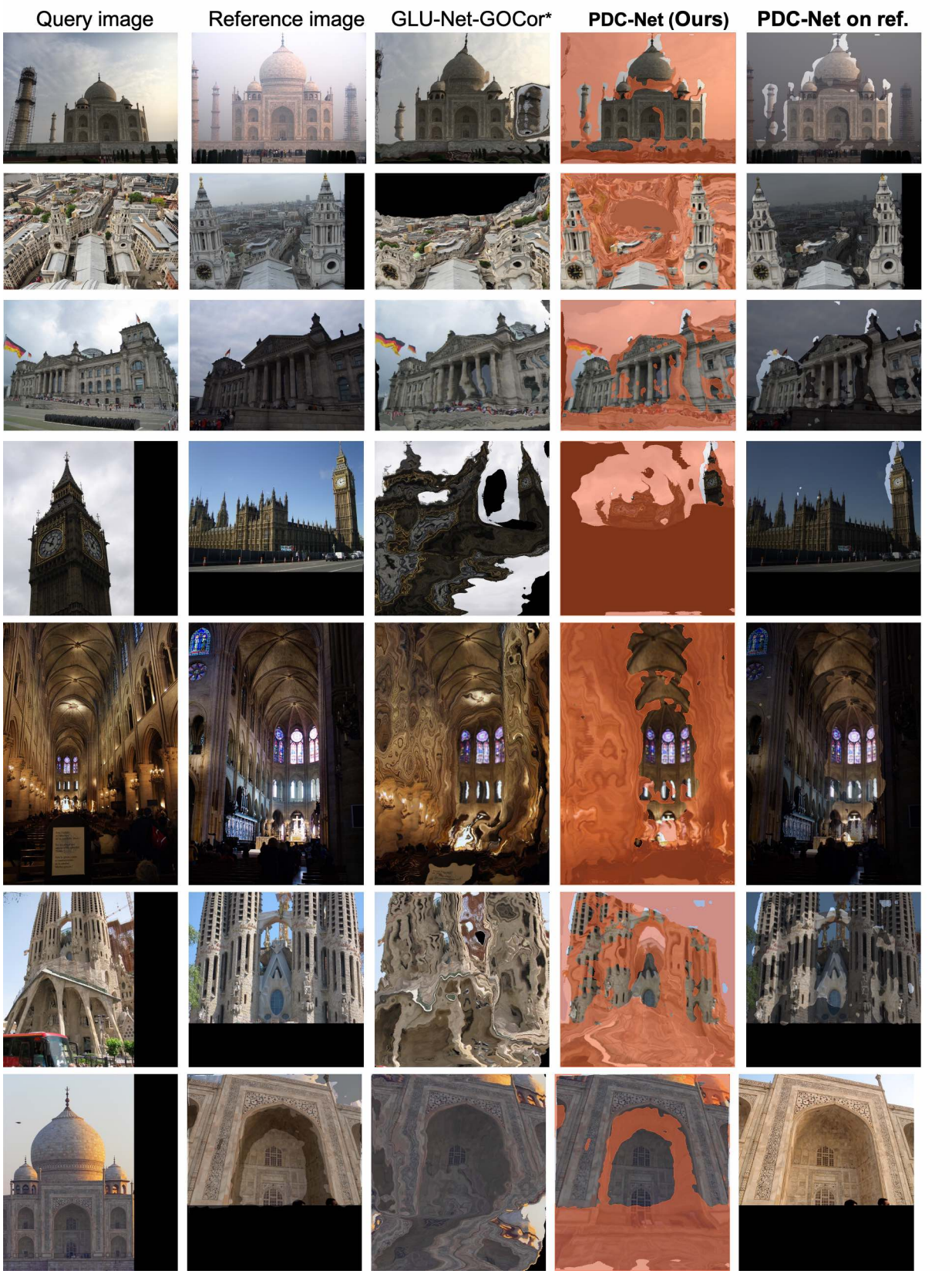

Figure 10. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseline GLU-Net-GOCor*, applied to images of the MegaDepth dataset [13]. In the 3^{rd} and 4^{th} columns, we visualize the query images warped according to the flow fields estimated by the GLU-Net-GOCor* and 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 11. Qualitative examples of our approach PDC-Net and corresponding non-probabilistic baseline GLU-Net-GOCor*, applied to images of the MegaDepth dataset [13]. 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).

References

[1] 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. 3

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

[3] 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. 3

[4] 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. 3

[5] 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. 8

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

[7] 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. 7

[8] 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. 5, 6

[9] 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. 3

[10] 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. 6

[11] 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. 6

[12] 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. 14

[13] 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, 15, 16, 17

[14] 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. 3

[15] 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. 7

[16] 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. 3, 8

[17] David Nistér. An efficient solution to the five-point relative pose problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(6):756–777, June 2004. 8

[18] 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. 6

[19] 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. 8

[20] 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. 3, 7, 8

[21] 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. 7

[22] 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. 5, 7, 8, 9

[23] 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. 3

[24] 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. 13

[25] 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. 2, 4, 5, 6, 7, 8

[26] 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. 4, 5, 6, 7, 8

[27] 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. 9

[28] 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

[29] 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. 3