COTR: Correspondence Transformer for Matching Across Images
Wei Jiang\textsuperscript{1}, Eduard Trulls\textsuperscript{2}, Jan Hosang\textsuperscript{2}, Andrea Tagliasacchi\textsuperscript{2,3}, Kwang Moo Yi\textsuperscript{1}
\textsuperscript{1}University of British Columbia, \textsuperscript{2}Google Research, \textsuperscript{3}University of Toronto

Abstract

We propose a novel framework for finding correspondences in images based on a deep neural network that, given two images and a query point in one of them, finds its correspondence in the other. By doing so, one has the option to query only the points of interest and retrieve sparse correspondences, or to query all points in an image and obtain dense mappings. Importantly, in order to capture both local and global priors, and to let our model relate between image regions using the most relevant among said priors, we realize our network using a transformer. At inference time, we apply our correspondence network by recursively zooming in around the estimates, yielding a multiscale pipeline able to provide highly-accurate correspondences. Our method significantly outperforms the state of the art on both sparse and dense correspondence problems on multiple datasets and tasks, ranging from wide-baseline stereo to optical flow, without any retraining for a specific dataset. We commit to releasing data, code, and all the tools necessary to train from scratch and ensure reproducibility.

1. Introduction

Finding correspondences across pairs of images is a fundamental task in computer vision, with applications ranging from camera calibration [22, 28] to optical flow [32, 15], Structure from Motion (SfM) [56, 28], visual localization [55, 53, 36], point tracking [35, 68], and human pose estimation [43, 20]. Traditionally, two fundamental research directions exist for this problem. One is to extract sets of sparse keypoints from both images and match them in order to minimize an alignment metric [33, 55, 28]. The other is to interpret correspondence as a dense process, where every pixel in the first image maps to a pixel in the second image [32, 60, 77, 72].

The divide between sparse and dense emerged naturally from the applications they were devised for. Sparse methods have largely been used to recover a single global camera motion, such as in wide-baseline stereo, using geometrical constraints. They rely on local features [34, 74, 44, 13] and further prune the putative correspondences formed with them in a separate stage with sampling-based robust matchers [18, 3, 12], or their learned counterparts [75, 7, 76, 64, 54]. Dense methods, by contrast, usually model small temporal changes, such as optical flow in video sequences, and rely on local smoothness [35, 24]. Exploiting context in this manner allows them to find correspondences at arbitrary locations, including seemingly texture-less areas.

In this work, we present a solution that bridges this divide, a novel network architecture that can express both forms of prior knowledge – global and local – and learn them implicitly from data. To achieve this, we leverage the inductive bias that densely connected networks possess in representing smooth functions [1, 4, 48] and use a transformer [73, 10, 14] and further prune the putative correspondences formed with them in a separate stage with sampling-based robust matchers [18, 3, 12], or their learned counterparts [75, 7, 76, 64, 54]. Dense methods, by contrast, usually model small temporal changes, such as optical flow in video sequences, and rely on local smoothness [35, 24]. Exploiting context in this manner allows them to find correspondences at arbitrary locations, including seemingly texture-less areas.

In this work, we present a solution that bridges this divide, a novel network architecture that can express both forms of prior knowledge – global and local – and learn them implicitly from data. To achieve this, we leverage the inductive bias that densely connected networks possess in representing smooth functions [1, 4, 48] and use a transformer [73, 10, 14] and further prune the putative correspondences formed with them in a separate stage with sampling-based robust matchers [18, 3, 12], or their learned counterparts [75, 7, 76, 64, 54]. Dense methods, by contrast, usually model small temporal changes, such as optical flow in video sequences, and rely on local smoothness [35, 24]. Exploiting context in this manner allows them to find correspondences at arbitrary locations, including seemingly texture-less areas.

In this work, we present a solution that bridges this divide, a novel network architecture that can express both forms of prior knowledge – global and local – and learn them implicitly from data. To achieve this, we leverage the inductive bias that densely connected networks possess in representing smooth functions [1, 4, 48] and use a transformer [73, 10, 14]
to automatically control the nature of priors and learn how to utilize them through its attention mechanism. For example, ground-truth optical flow typically does not change smoothly across object boundaries, and simple (attention-agnostic) densely connected networks would have challenges in modelling such a discontinuous correspondence map, whereas a transformer would not. Moreover, transformers allow encoding the relationship between different locations of the input data, making them a natural fit for correspondence problems.

Specifically, we express the problem of finding correspondences between images \( I \) and \( I' \) in functional form, as \( x' = F_\Phi(x \mid I, I') \), where \( F_\Phi \) is our neural network architecture, parameterized by \( \Phi \), \( x \) indexes a query location in \( I \), and \( x' \) indexes its corresponding location in \( I' \); see Figure 1. Differently from sparse methods, COTR can match arbitrary query points via this functional mapping, predicting only as many matches as desired. Differently from dense methods, COTR learns smoothness implicitly and can deal with large camera motion effectively.

Our work is the first to apply transformers to obtain accurate correspondences. Our main technical contributions are:

- we propose a functional correspondence architecture that combines the strengths of dense and sparse methods;
- we show how to apply our method recursively at multiple scales during inference in order to compute highly-accurate correspondences;
- we demonstrate that COTR achieves state-of-the-art performance in both dense and sparse correspondence problems on multiple datasets and tasks, without retraining;
- we substantiate our design choices and show that the transformer is key to our approach by replacing it with a simpler model, based on a Multi-Layer Perceptron (MLP).

2. Related works

We review the literature on both sparse and dense matching, as well as works that utilize transformers for vision.

**Sparse methods.** Sparse methods generally consist of three stages: keypoint detection, feature description, and feature matching. Seminal detectors include DoG [34] and FAST [51]. Popular patch descriptors range from hand-crafted [34, 9] to learned [42, 66, 17] ones. Learned feature extractors became popular with the introduction of LIFT [74], with many follow-ups [13, 44, 16, 49, 5, 71]. Local features are designed with sparsity in mind, but have also been applied densely in some cases [67, 32]. Learned local features are trained with intermediate metrics, such as descriptor distance or number of matches.

Feature matching is treated as a separate stage, where descriptors are matched, followed by heuristics such as the ratio test, and robust matchers, which are key to deal with high outlier ratios. The latter are the focus of much research, whether hand-crafted, following RANSAC [18, 12, 3], consensus- or motion-based heuristics [11, 31, 6, 37], or learned [75, 7, 76, 64]. The current state of the art builds on attentional graph neural networks [54]. Note that while some of these theoretically allow feature extraction and matching to be trained end to end, this avenue remains largely unexplored. We show that our method, which does not divide the pipeline into multiple stages and is learned end-to-end, can outperform these sparse methods.

**Dense methods.** Dense methods aim to solve optical flow. This typically implies small displacements, such as the motion between consecutive video frames. The classical Lucas-Kanade method [35] solves for correspondences over local neighbourhoods, while Horn-Schunck [24] imposes global smoothness. More modern algorithms still rely on these principles, with different algorithmic choices [59], or focus on larger displacements [8]. Estimating dense correspondences under large baselines and drastic appearance changes was not explored until methods such as DeMoN [72] and SfMLearner [77] appeared, which recovered both depth and camera motion – however, their performance fell somewhat short of sparse methods [75]. Neighbourhood Consensus Networks [50] explored 4D correlations – while powerful, this limits the image size they can tackle. More recently, DGC-Net [38] applied CNNs in a coarse-to-fine approach, trained on synthetic transformations, GLU-Net [69] combined global and local correlation layers in a feature pyramid, and GOCor [70] improved the feature correlation layers to disambiguate repeated patterns. We show that we outperform DGC-Net, GLU-Net and GOCor over multiple datasets, while retaining our ability to query individual points.

**Attention mechanisms.** The attention mechanism enables a neural network to focus on part of the input. Hard attention was pioneered by Spatial Transformers [26], which introduced a powerful differentiable sampler, and was later improved in [27]. Soft attention was pioneered by transformers [73], which has since become the de-facto standard in natural language processing – its application to vision tasks is still in its early stages. Recently, DETR [10] used Transformers for object detection, whereas ViT [14] applied them to image recognition. Our method is the first application of transformers to image correspondence problems. ¹

1 A concurrent relevant work for feature-less image matching was proposed shortly after our work became public [63].
3. Method

We first formalize our problem (Section 3.1), then detail our architecture (Section 3.2), its recursive use at inference time (Section 3.3), and our implementation (Section 3.4).

3.1. Problem formulation

Let $x \in [0, 1]^2$ be the normalized coordinates of the query point in image $I$, for which we wish to find the corresponding point, $x' \in [0, 1]^2$, in image $I'$. We frame the problem of learning to find correspondences as that of finding the best set of parameters $\Phi$ for a parametric function $F_\Phi(x | I, I')$ minimizing

$$\arg \min_{\Phi} \mathbb{E}_{(x, x', I, I') \sim D} \mathcal{L}_{\text{corr}} + \mathcal{L}_{\text{cycle}},$$

where $D$ is the training dataset of ground correspondences, $\mathcal{L}_{\text{corr}}$ measures the correspondence estimation errors, and $\mathcal{L}_{\text{cycle}}$ enforces correspondences to be cycle-consistent.

3.2. Network architecture

We implement $F_\Phi$ with a transformer. Our architecture, inspired by [10, 14], is illustrated in Figure 2. We first crop and resize the input into a $256 \times 256$ image, and convert it into a downsampled feature map size $16 \times 16 \times 256$ with a shared CNN backbone, $E$. We then concatenate the representations for two corresponding images side by side, forming a feature map size $16 \times 32 \times 256$, to which we add positional encoding $\Omega$ (i.e., $\text{MeshGrid}(0:1, 0:2)$ of size $16 \times 32 \times 2$) to produce a context feature map $c$ (of size $16 \times 32 \times 256$):

$$c = [E(I), E(I')] + \mathcal{P}(\Omega),$$

where $[;]$ denotes concatenation along the spatial dimension – a subtly important detail novel to our architecture that we discuss in greater depth later on. We then feed the context feature map $c$ to a transformer encoder $T_E$, and interpret its results with a transformer decoder $T_D$, along with the query point $x$, encoded by $P$ – the positional encoder used to generate $\Omega$. We finally process the output of the transformer decoder with a fully connected layer $D$ to obtain our estimate for the corresponding point, $x'$.

$$x' = F_\Phi(x | I, I') = D(T_D(P(x), T_E(c))).$$

For architectural details of each component please refer to supplementary material.

Importance of context concatenation. Concatenation of the feature maps along the spatial dimension is critical, as it allows the transformer encoder $T_E$ to relate between locations within the image (self-attention), and across images (cross-attention). Note that, to allow the encoder to distinguish between pixels in the two images, we employ a single positional encoding for the entire concatenated feature map; see Fig. 2. We concatenate along the spatial dimension rather than the channel dimension, as the latter would create artificial relationships between features coming from the same pixel locations in each image. Concatenation allows the features in each map to be treated in a way that is similar to words in a sentence [73]. The encoder then associates and relates them to discover which ones to attend to given their context – which is arguably a more natural way to find correspondences.

Linear positional encoding. We found it critical to use a linear increase in frequency for the positional encoding, as opposed to the commonly used log-linear strategy [73, 10], which made our optimization unstable; see supplementary material. Hence, for a given location $x = [x, y]$ we write

$$p_k(x) = \left[\sin(k \pi x \top), \cos(k \pi x \top)\right],$$

where $N = 256$ is the number of channels of the feature map. Note that $p_k$ generates four values, so that the output of the encoder $P$ is size $N$.

Querying multiple points. We have introduced our framework as a function operating on a single query point, $x$. However, as shown in Fig. 2, extending it to multiple query points is straightforward. We can simply input multiple queries at once, which the transformer decoder $T_D$ and the decoder $D$ will translate into multiple coordinates. Importantly, while doing so, we disallow self attention among the query points in order to ensure that they are solved independently.
Inference with recursive with zoom-in

mensurate with the current zoom level, and resize them to
original image around the estimated points, of a size com-
In subsequent zoom-ins, we crop square patches from the
256×256 them to
arbitrary size, in the initial step we simply resize (i.e. stretch)
256×256 aspects images of fixed
Dealing with images of arbitrary size

Our network ex-
planes (the unmasked pixels in Fig. 4).

We keep, for each image, the 20 image pairs with the largest
20 pixels on the
We then compute the common area between the remaining
256×256, and estimate the initial correspondences.

We obtain ac-
Figure 3. Recursive COTR at inference time – We obtain ac-
curate correspondences by applying our functional approach re-
cursively, zooming into the results of the previous iteration, and
and running the same network on the pair of zoomed-in crops. We gradu-
ually focus on the correct correspondence, with greater accuracy.

3.3. Inference

We next discuss how to apply our functional approach at
inference time in order to obtain accurate correspondences.
Inference with recursive with zoom-in. Applying the pow-
ful transformer attention mechanism to vision problems
comes at a cost – it requires heavily downsampled feature
maps, which in our case naturally translates to poorly local-
ized correspondences; see Section 4.6. We address this by
exploiting the functional nature of our approach, applying
out network \( F \) recursively. As shown in Fig. 3, we itera-
tively zoom into a previously estimated correspondence, on
both images, in order to obtain a refined estimate. There
is a trade-off between compute and the number of zoom-in
steps. We ablated this carefully on the validation data and
settled on a zoom-in factor of two at each step, with four
zoom-in steps. It is worth noting that multiscale refinement
is common in many computer vision algorithms [32, 15],
but thanks to our functional correspondence model, realizing
such a multiscale inference process is not only possible, but
also straightforward to implement.

Compensating for scale differences. While matching im-
ages recursively, one must account for a potential mismatch
in scale between images. We achieve this by making the
scale of the patch to crop proportional to the commonly vis-
ible regions in each image, which we compute on the first
step, using the whole images. To extract this region, we
compute the cycle consistency error at the coarsest level,
for every pixel, and threshold it at \( \tau_{\text{visible}}=5 \) pixels on the
256×256 image; see Fig. 4. In subsequent stages – the
zoom-ins – we simply adjust the crop sizes over \( I \) and \( I' \)
so that their relationship is proportional to the sum of valid
pixels (the unmasked pixels in Fig. 4).

Dealing with images of arbitrary size. Our network ex-
pects images of fixed 256×256 shape. To process images of
arbitrary size, in the initial step we simply resize (i.e. stretch)
them to 256×256, and estimate the initial correspondences.
In subsequent zoom-ins, we crop square patches from the
original image around the estimated points, of a size com-
mensurate with the current zoom level, and resize them to

256×256. While this may seem a limitation on images with
non-standard aspect ratios, our approach performs well on
KITTI, which are extremely wide (3.3:1). Moreover, we pre-
sent a strategy to tile detections in Section 4.4.

Discarding erroneous correspondences. What should we
do when we query a point is occluded or outside the viewport
in the other image? Similarly to our strategy to compensate
for scale, we resolve this problem by simply rejecting corres-
pondences that induce a cycle consistency error (3) greater
than \( \tau_{\text{cycle}}=5 \) pixels. Another heuristic we apply is to termi-
nate correspondences that do not converge while zooming
in. We compute the standard deviation of the zoom-in es-
mates, and reject correspondences that oscillate by more
than \( \tau_{\text{std}}=0.02 \) of the long-edge of the image.

Interpolating for dense correspondence. While we could
query every single pixel in order to obtain dense esti-
mates, it is also possible to densify matches by computing
sparse matches first, and then interpolating using barycentric
weights on a Delaunay triangulation of the queries. This
interpolation can be done efficiently using a GPU rasterizer.

3.4. Implementation details

Datasets. We train our method on the MegaDepth
dataset [30], which provides both images and corresponding
dense depth maps, generated by SfM [56]. These images
come from photo-tourism and show large variations in ap-
pearance and viewpoint, which is required to learn invariant
models. The accuracy of the depth maps is sufficient to learn
accurate local features, as demonstrated by [16, 54, 71]. To
find co-visible pairs of images we can train with, we first
filter out those with no common 3D points in the SfM model.
We then compute the common area between the remaining
pairs of images, by projecting pixels from one image to the
other. Finally, we compute the intersection over union of the
projected pixels, which accounts for different image sizes.
We keep, for each image, the 20 image pairs with the largest
overlap. This simple procedure results in a good combina-
tion of images with a mixture of high/low overlap. We use
115 scenes for training and 1 scene for validation.

Implementation. We implement our method in Py-
Torch [46]. For the backbone \( E \) we use a ResNet50 [23],
initialized with weights pre-trained on ImageNet [52]. We
use the feature map after its fourth downsampling step (after

![Figure 3. Recursive COTR at inference time – We obtain accurate correspondences by applying our functional approach recursively, zooming into the results of the previous iteration, and running the same network on the pair of zoomed-in crops. We gradually focus on the correct correspondence, with greater accuracy.](image_url)

![Figure 4. Estimating scale by finding co-visible regions – We show two images we wish to put in correspondence, and the estimated regions in common – image locations with a high cycle-consistency error are masked out.](image_url)
the third residual block), which is of size $16 \times 16 \times 1024$, which we convert into $16 \times 16 \times 256$ with $1 \times 1$ convolutions. For the transformer, we use $6$ layers for both encoder and decoder. Each encoder layer contains a self-attention layer with $8$ heads, and each decoder layer contains an encoder-decoder attention layer with $8$ heads, but with no self-attention layers, in order to prevent query points from communicating between each other. Finally, for the network that converts the Transformer output into coordinates, $D$, we use a $3$-layer MLP, with $256$ units each, followed by ReLU activations.

**On-the-fly training data generation.** We select training pairs randomly, pick a random query point in the first image, and find its corresponding point on the second image using the ground truth depth maps. We then select a random zoom level among one of ten levels, uniformly spaced, in log scale, between $1 \times$ and $10 \times$. We then crop a square patch at the desired zoom level, centered at the query point, from the first image, and a square patch that contains the corresponding point in the second image. Given this pair of crops, we sample $100$ random valid correspondences across the two crops – if we cannot gather at least $100$ valid points, we discard the pair and move to the next.

**Staged training.** Our model is trained in three stages. First, we freeze the pre-trained backbone $E$, and train the rest of the network, for $300k$ iterations, with the ADAM optimizer [29], a learning rate of $10^{-4}$, and a batch size of $24$. We then unfreeze the backbone and fine-tune everything end-to-end with a learning rate of $10^{-5}$ and a batch size of $16$, to accommodate the increased memory requirements, for $2M$ iterations, at which point the validation loss plateaus. Note that in the first two stages we use the whole images, resized to $256 \times 256$, as input, which allows us to load the entire dataset into memory. In the third stage we introduce zoom-ins, generated as explained above, and train everything end-to-end for a further $300k$ iterations.

### 4. Results

We evaluate our method with four different datasets, each aimed for a different type of correspondence task. We do not perform any kind of re-training or fine-tuning. They are:

- **HPatches** [2]: A dataset with planar surfaces viewed under different angles/illumination settings, and ground-truth homographies. We use this dataset to compare against dense methods that operate on the entire image.
- **KITTI** [19]: A dataset for autonomous driving, where the ground-truth 3D information is collected via LIDAR. With this dataset we compare against dense methods on complex scenes with camera and multi-object motion.
- **ETH3D** [57]: A dataset containing indoor and outdoor scenes captured using a hand-held camera, registered with SfM. As it contains video sequences, we use it to evaluate how methods perform as the baseline widens by increasing the interval between samples, following [69].

#### 4.1. HPatches

We follow the evaluation protocol of [69, 70], which computes the Average End Point Error (AEPE) for all valid pixels, and the Percentage of Correct Keypoints (PCK) at a given reprojection error threshold – we use $1, 3$, and $5$ pixels. Image pairs are generated taking the first (out of six) images for each scene as reference, which is matched against the other five. We provide two results for our method: ‘COTR’, which uses $1,000$ random query points for each image pair, and ‘COTR + Interp.’, which interpolates correspondences for the remaining pixels using the strategy presented in Section 3.3. We report our results in Table 1.

Our method provides the best results, with and without interpolation, with the exception of PCK-1px, where it remains close to the best baseline. We note that the results for this threshold should be taken with a grain of salt, as several scenes do not satisfy the planar assumption for all pixels. To provide some evidence for this, we reproduce the results for GLU-Net [69] using the code provided by the authors to measure PCK at $3$ pixels, which was not computed in the paper. COTR outperforms it by a significant margin.

#### 4.2. KITTI

To evaluate our method in an environment more complex than simple planar scenes, we use the KITTI dataset [39, 40]. Following [70, 65], we use the training split for this evaluation, as ground-truth for the test split remains private – all methods, including ours, were trained on a separate dataset. We report results both in terms of AEPE, and ‘Fl.’ – the

| Method | AEPE ↓ | PCK-1px ↑ | PCK-3px ↑ | PCK-5px ↑ |
|--------|--------|-----------|-----------|-----------|
| LiteFlowNet [25] | 118.85 | 13.91 | – | 31.64 |
| PWC-Net [61, 62] | 96.14 | 13.14 | – | 37.14 |
| DGC-Net [38] | 33.26 | 12.00 | – | 58.06 |
| GLU-Net [69] | 25.05 | 39.55 | 71.52 | 78.54 |
| GLU-Net+GOCor [70] | 20.16 | 41.55 | – | 81.43 |
| **COTR** | 7.75 | 40.91 | **82.37** | **91.10** |
| **COTR + Interp.** | 7.98 | 33.08 | 77.09 | 86.33 |

Table 1. Quantitative results on HPatches – We report Average End Point Error (AEPE) and Percent of Correct Keypoints (PCK) with different thresholds. For PCK-1px and PCK-5px, we use the numbers reported in literature. We **bold** the best method and underline the second best.
Figure 5. **Qualitative examples on KITTI** – We show the optical flow and its corresponding error map (“jet” color scheme) for three examples from KITTI-2015, with GLU-Net [69] as a baseline. COTR successfully recovers both the global motion in the scene, and the movement of individual objects, even when nearby cars move in opposite directions (top) or partially occlude each other (bottom).

| Method                  | AEPE \( \downarrow \) | FL \( \% \) \( \downarrow \) | AEPE \( \downarrow \) | FL \( \% \) \( \downarrow \) |
|-------------------------|---------------------|------------------|---------------------|------------------|
| LiteFlowNet [25]         | 4.00                | 17.47            | 10.39               | 28.50            |
| PWC-Net [61, 62]         | 4.14                | 20.28            | 10.35               | 33.67            |
| DGC-Net [58]            | 8.50                | 32.28            | 14.97               | 50.98            |
| GLU-Net [69]            | 3.34                | 18.93            | 9.79                | 37.52            |
| RAFT [65]               | 2.15                | 9.30             | 5.04                | 17.8             |
| GLU-Net+GOCor [70]      | 2.68                | 15.43            | 6.68                | 27.57            |
| **COTR**                | 1.26                | 7.36             | 2.62                | 9.92             |
| **COTR**+Interp.         | 2.26                | 10.50            | 6.12                | 16.90            |

Table 2. **Quantitative results on KITTI** – We report the Average End Point Error (AEPE) and the flow outlier ratio (“Fl.”) on the 2012 and 2015 versions of the KITTI dataset. Our method outperforms most baselines, with the interpolated version being on par with RAFT, and slightly edging out GLU-Net+GOCor.

percentage of optical flow outliers. As KITTI images are large, we randomly sample 40,000 points per image pair, from the regions covered by valid ground truth.

We report the results on both KITTI-2012 and KITTI-2015 in Table 2. Our method outperforms all the baselines by a large margin. Note that the interpolated version also performs similarly to the state of the art, slightly better in terms of flow accuracy, and slightly worse in terms of AEPE, compared to RAFT [65]. It is important to understand here that, while COTR provides a drastic improvement over compared methods, we are evaluating only on points where COTR returns confident results, which is about 81.8% of the queried locations – among the 18.2% of rejected queries, 67.8% fall out of the borders of the other image, which indicates that our filtering is reasonable. This shows that COTR provides highly accurate results in the points we query and retrieve estimates for, and is currently limited by the interpolation strategy. This suggests that improved interpolation strategies based on CNNs, such as those used in [41], would be a promising direction for future research.

In Fig. 5 we further highlight cases where our method shows clear advantages over the competitors – we see that the objects in motion, *i.e.*, cars, result in high errors with GLU-Net, which is biased towards a single, *global* motion. Our method, on the other hand, successfully recovers the flow fields for these cases as well, with minor errors at the boundaries, due to interpolation. These examples clearly demonstrate the role that attention plays when estimating correspondences on scenes with moving objects.

Finally, we stress that while our method is trained on MegaDepth, an urban dataset exhibiting only global, rigid motion, for which ground truth is only available on stationary objects (mostly building facades), our method proves capable of recovering the motion of objects moving in different directions; see Fig. 5, bottom. In other words, it learns to find *precise, local correspondences* within images, rather than global motion.

### 4.3. ETH3D

We also report results on the ETH3D dataset, following [69, 70]. This task is closer to the ‘sparse’ scenario, as performance is only evaluated on pixels corresponding to SfM locations with valid ground truth, which are far fewer than for HPatches or KITTI. We summarize the results in terms of AEPE in Table 3, sampling pairs of images with an increasing number of frames between them (the sampling “rate”), which correlates with baseline and, thus, difficulty. Our method produces the most accurate correspondences for every setting, tied with LiteFlowNet [25] at a 3-frame difference, and drastically outperforms every method as the

| Method                  | AEPE \( \downarrow \) | Rate=3 | Rate=5 | Rate=7 | Rate=9 | Rate=11 | Rate=13 | Rate=15 |
|-------------------------|---------------------|--------|--------|--------|--------|---------|---------|---------|
| LiteFlowNet [25]         | 1.66                | 2.58   | 6.05   | 12.95  | 29.67  | 52.41   | 74.96   |
| PWC-Net [61, 62]         | 1.75                | 2.10   | 3.21   | 5.59   | 14.35  | 27.49   | 43.41   |
| DGC-Net [33]            | 2.49                | 3.28   | 4.18   | 5.35   | 6.78   | 9.02    | 12.23   |
| GLU-Net [69]            | 1.98                | 2.54   | 3.49   | 4.24   | 5.61   | 7.55    | 10.78   |
| RAFT [65]               | 1.92                | 2.12   | 2.33   | 2.58   | 3.90   | 8.63    | 13.74   |
| **COTR**                | 1.66                | 1.82   | 1.97   | 2.13   | 2.27   | 2.41    | 2.61    |
| **COTR**+Interp.        | 1.21                | 1.52   | 1.46   | 2.47   | 3.82   | 5.23    | 3.76    |

Table 3. **Quantitative results for ETH3D** – We report the Average End Point Error (AEPE) at different sampling “rates” (frame intervals). Our method performs significantly better as the rate increases and the problem becomes more difficult.
baseline increases; see qualitative results in Fig. 6.

4.4. Image Matching Challenge

Accurate, 6-DOF pose estimation in unconstrained urban scenarios remains too challenging a problem for dense methods. We evaluate our method on a popular challenge for pose estimation with local features, which measures performance in terms of the quality of the estimated poses, in terms of mean average accuracy (mAA) at a 5° and 10° error threshold; see [28] for details.

We focus on the stereo task. As this dataset contains images with unconstrained aspect ratios, instead of stretching the image before the first zoom level, we simply resize the short-edge to 256 and tile our coarse, image-level estimates — e.g. an image with 2:1 aspect ratio would invoke two tiling instances. If this process generates overlapping tiles (e.g. with a 4:3 aspect ratio), we choose the estimate that gives best cycle consistency among them. We pair our method with DEGENSAC [12] to retrieve the final pose, as recommended by [28] and done by most participants.

We summarize the results in Table 4. We consider the top performers in the 2020 challenge (a total of 228 entries can be found in the leaderboards [link]). As the challenge places a limit on the number of keypoints, instead of matches, we consider both categories (up to 2k and up to 8k keypoints per image), for fairness — note that our method has no notion of keypoints, instead, we query at random locations.

With 2k matches and excluding the methods that feature semantic masking — a heuristic employed in the challenge by some participants to filter out keypoints on transient structures such as the sky or pedestrians — COTR ranks second overall. These results showcase the robustness and generality of our method, considering that it was not trained specifically to solve wide-baseline stereo problems. In contrast, the other top entries are engineered towards this specific application. We also provide results lowering the cap on the number of matches (N) under pure COTR and one entry with 2048 keypoints under COTR guided matching. Pure COTR outperforms all methods in the 2k-keypoints category (other than those specifically excluded) with as few as 512 matches per image. With ☉ we indicate clickable URLs to the leaderboard webpage.

Table 4. Stereo performance on IMC2020 — We report mean Average Accuracy (mAA) at 5° and 10°, and the number of inlier matches, for the top IMC2020 entries, on all test scenes. We highlight the best method in bold and underline the second-best. We exclude entries with components specifically tailored to the challenge, which are enclosed in parentheses, but report them for completeness. Finally, we report results with different number of matches (N) under pure COTR and one entry with 2048 keypoints under COTR guided matching. Pure COTR outperforms all methods in the 2k-keypoints category (other than those specifically excluded) with as few as 512 matches per image. With ☉ we indicate clickable URLs to the leaderboard webpage.

Figure 6. Qualitative examples on ETH3D — We show results for GLU-Net [69] and COTR for two examples, one indoors and one outdoors. Correspondences are drawn in green if their reprojection error is below 10 pixels, and red otherwise.

Figure 7. Qualitative examples for IMC2020 — We visualize the matches produced by COTR (with N = 512) for some stereo pairs in the Image Matching Challenge dataset. Matches are coloured red to green, according to their reprojection error (high to low).
We use the HPatches dataset, with more granularity than with a factor of two at each zoom provides a good balance we use for inference, and display the histogram of pixels whether a simpler approach would suffice. We explore the distribution shifts to the left and gets squeezed, yielding more accurate estimates. While zooming in more is nearly always beneficial, we found empirically that four zoom-ins up to ‘rate=9’, gradually increasing until 3.65% at ‘rate=15’. The error clearly decreases as more zoom is applied. We validate the effectiveness of filtering out bad correspondences (Section 3.3) on the ETH3D dataset, where it improves AEPE by roughly 5% relative. More importantly, it effectively removes correspondences with a potentially high error. This allows the dense interpolation step to produce better results. We find that on average 1.2% of the correspondences are filtered out on this dataset – below 1% up to ‘rate=9’, gradually increasing until 3.65% at ‘rate=15’.

**On the role of the transformer.** Transformers are powerful attention mechanisms, but also costly. It is fair to wonder whether a simpler approach would suffice. We explore the use of MLPs in place of transformers, forming a pipeline similar to [21], and train such a variant – see supplementary material for details. In Fig. 9, we see that the MLP yields globally-smooth estimates, as expected, which fail to model the discontinuities that occur due to 3D geometry. On the other hand, COTR with the transformer successfully aligns source and target even when such discontinuities exist.

**Zooming.** To evaluate how our zooming strategy affects the localization accuracy of the correspondences, we measure the errors in the estimation at each zoom level, in pixels. We use the HPatches dataset, with more granularity than we use for inference, and display the histogram of pixels errors at each zoom level in Fig. 10. As we zoom-in, the distribution shifts to the left and gets squeezed, yielding more accurate estimates. While zooming in more is nearly always beneficial, we found empirically that four zoom-ins with a factor of two at each zoom provides a good balance between compute and accuracy.

5. Conclusions and future work

We introduced a functional network for image correspondence that is capable to address both sparse and dense matching problems. Through a novel architecture and recursive inference scheme, it achieves performance on par or above the state of the art on HPatches, KITTI, ETH3D, and one scene from IMC2020. As future work, in addition to the improvements we have suggested throughout the paper, we intend to explore the application of COTR to semantic and multi-modal matching, and incorporate refinement techniques to further improve the quality of its dense estimates.

**Acknowledgements**

This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Discovery Grant, Google’s Visual Positioning System, Compute Canada, and Advanced Research Computing at the University of British Columbia.
References

[1] Matan Atzmon and Yaron Lipman. SAL: Sign Agnostic Learning of Shapes from Raw Data. In Conference on Computer Vision and Pattern Recognition, 2020. 1

[2] Vassileios Baltas, Karel Lenc, Andrea Vedaldi, and Krystian Mikolajczyk. Hpatches: A Benchmark and Evaluation of Handcrafted and Learned Local Descriptors. In Conference on Computer Vision and Pattern Recognition, 2017. 5

[3] Daniel Barath, Jiri Matas, and Jana Noskova. MAGSAC: Marginalizing Sample Consensus. In Conference on Computer Vision and Pattern Recognition, 2019. 1, 2, 7

[4] Ronen Basri, Meirav Galun, Amnon Gefman, David Jacobs, Yoni Kasten, and Shira Kritchman. Frequency Bias in Neural Networks for Input of Non-uniform Density. In International Conference on Machine Learning, 2020. 1

[5] Aritra Bhowmik, Stefan Gumhold, Carsten Rother, and Eric Brachmann. Reinforced Feature Points: Optimizing Feature Detection and Description for a High-level Task. In Conference on Computer Vision and Pattern Recognition, 2020. 2

[6] JiaWang Bian, Wen-Yan Lin, Yasuyuki Matsushita, Sai-Kit Yeung, Tan-Dat Nguyen, and Ming-Ming Cheng. GMS: Grid-based Motion Statistics for Fast, Ultra-robust Feature Correspondence. In Conference on Computer Vision and Pattern Recognition, 2017. 2

[7] Eric Brachmann and Carsten Rother. Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. In International Conference on Computer Vision, 2019. 1, 2

[8] Thomas Brox and Jitendra Malik. Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(3):500–513, 2010. 2

[9] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary Robust Independent Elementary Features. In European Conference on Computer Vision, 2010. 2

[10] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In European Conference on Computer Vision, pages 213–229, Springer, 2020. 1, 2, 3, 12

[11] Luca Cavalli, Viktor Larsson, Martin Ralf Oswald, Torsten Sattler, and Marc Pollefeys. Handcrafted outlier detection revisited. In European Conference on Computer Vision, 2020. 2, 7

[12] Ondrej Chum, Tomas Werner, and Jiri Matas. Two-view Geometry Estimation Unaffected by a Dominant Plane. In Conference on Computer Vision and Pattern Recognition, 2005. 1, 2, 7

[13] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. In Conference on Computer Vision and Pattern Recognition Workshops, 2018. 1, 2, 7

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In International Conference on Learning Representations, 2021. 1, 2, 3

[15] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks. In International Conference on Computer Vision, 2015. 1, 4

[16] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A Trainable CNN for Joint Description and Detection of Local Features. In Conference on Computer Vision and Pattern Recognition, 2019. 2, 4

[17] Patrick Ebel, Anastasiia Mishchuk, Kwang Moo Yi, Pascal Fua, and Eduard Trulls. Beyond Cartesian Representations for Local Descriptors. In International Conference on Computer Vision, 2019. 2, 7

[18] Martin A Fischler and Robert C Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Communications of the ACM, 24(6):381–395, 1981. 1, 2

[19] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. International Journal of Robotics Research, 2013. 5

[20] Riza Alp Güler, Natalia Neverova, and Jasonas Kokkinos. DensePose: Dense Human Pose Estimation In The Wild. In Conference on Computer Vision and Pattern Recognition, 2018. 1

[21] Oshri Halimi, Ido Imanuel, Or Litany, Giovanni Trappolini, Emanuele Rodolà, Leonidas Guibas, and Ron Kimmel. Towards Precise Completion of Deformable Shape. European Conference on Computer Vision, 2020. 2, 8, 12

[22] Richard Hartley and Andrew Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, 2 edition, 2003. 1

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In Conference on Computer Vision and Pattern Recognition, 2016. 4, 12

[24] Berthold KP Horn and Brian G Schunck. Determining Optical Flow. Artificial intelligence, 17(1-3):185–203, 1981. 1, 2

[25] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. Lite-FlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation. In Conference on Computer Vision and Pattern Recognition, 2018. 5, 6

[26] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In Advances in Neural Information Processing Systems, 2015. 2

[27] Wei Jiang, Weiwei Sun, Andrea Tagliasacchi, Eduard Trulls, and Kwang Moo Yi. Linearized Multi-sampling for Differentiable Image Transformation. In International Conference on Computer Vision, 2019. 2

[28] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. Image Matching Across Wide Baselines: From Paper to Practice. International Journal of Computer Vision, pages 1–31, 2020. 1, 5, 7

[29] Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. In International Conference on Learning Representations, 2014. 5
[59] Deqing Sun, Stefan Roth, and Michael J Black. Secrets of Optical Flow Estimation and Their Principles. In Conference on Computer Vision and Pattern Recognition, 2010. 2

[60] Deqing Sun, Stefan Roth, and Michael J Black. A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them. International Journal of Computer Vision, 106(2):115–137, 2014. 1

[61] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume. In Conference on Computer Vision and Pattern Recognition, 2018. 5, 6

[62] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Models Matter, so Does Training: An Empirical Study of CNNs for Optical Flow Estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(6):1408–1423, 2019. 5, 6

[63] Jianming Sun, Zehong Shen, Yuan Wang, Hujun Bao, and Xiaowei Zhou. LoFTR: Detector-Free Local Feature Matching with Transformers. CVPR, 2021. 2

[64] Weiwei Sun, Wei Jiang, Eduard Trulls, Andrea Tagliasacchi, and Kwang Moo Yi. ACNe: Attentive Context Normalization for Robust Permutation-Equivariant Learning. In Conference on Computer Vision and Pattern Recognition, 2020. 1, 2

[65] Zachary Teed and Jia Deng. RAFT: Recurrent All-Pairs Field Transforms for Optical Flow. In European Conference on Computer Vision, 2020. 5, 6, 12, 13

[66] Yurun Tian, Xin Yu, Bin Fan, Fuchao Wu, Huub Heijnen, and Vassileios Baltmas. SOSNet: Second Order Similarity Regularization for Local Descriptor Learning. In Conference on Computer Vision and Pattern Recognition, 2019. 2

[67] Engin Tola, Vincent Lepetit, and Pascal Fua. Daisy: An Efficient Dense Descriptor Applied to Wide-baseline Stereo. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(5):815–830, 2009. 2

[68] Carlo Tomasi and T Kanade Detection. Tracking of Point Features. International Journal of Computer Vision, 1991. 1

[69] Prune Truong, Martin Danelljan, and Radu Timofte. GLU-Net: Global-Local Universal Network for Dense Flow and Correspondences. In Conference on Computer Vision and Pattern Recognition, 2020. 2, 5, 6, 7, 8

[70] Prune Truong, Martin Danelljan, Luc Van Gool, and Radu Timofte. GOCor: Bringing Globally Optimized Correspondence Volumes into Your Neural Network. In Advances in Neural Information Processing Systems, 2020. 2, 5, 6, 7

[71] Michal J Tyszkiewicz, Pascal Fua, and Eduard Trulls. DISK: Learning Local Features with Policy Gradient. In Advances in Neural Information Processing Systems, 2020. 2, 4, 7

[72] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and Motion Network for Learning Monocular Stereo. In Conference on Computer Vision and Pattern Recognition, 2017. 1, 2

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in Neural Information Processing Systems, 2017. 1, 2, 3, 12

[74] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned Invariant Feature Transform. In European Conference on Computer Vision, pages 467–483, 2016. 1, 2

[75] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to Find Good Correspondences. In Conference on Computer Vision and Pattern Recognition, 2018. 1, 2

[76] Jiahui Zhang, Dawei Sun, Zixin Luo, Anbang Yao, Lei Zhou, Tianwei Shen, Yurong Chen, Long Quan, and Hongen Liao. Learning Two-View Correspondences and Geometry Using Order-Aware Network. International Conference on Computer Vision, 2019. 1, 2, 7

[77] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised Learning of Depth and Ego-Motion from Video. In Conference on Computer Vision and Pattern Recognition, 2017. 1, 2
COTR: Correspondence Transformer for Matching Across Images

Supplementary Material

A. Compute

The functional (and recursive) nature of our approach, coupled with the use of a transformer, means that our method has significant compute requirements. Our currently non-optimized prototype implementation queries one point at a time, and achieves 35 correspondences per second on a NVIDIA RTX 3090 GPU. This limitation could be addressed by careful engineering in terms of tiling and batching. Our preliminary experiments show no significant drop in performance when we query different points inside a given crop – we could thus potentially process any queries at the coarsest level in a single operation, and drastically reduce the number of operations in the zoom-ins (depending on how many queries overlap in a given crop). We expect this will speed up inference drastically. In addition to batching the queries at inference time, we plan to explore its use on non-random points (such as keypoints) and advanced interpolation techniques.

B. Log-linear vs Linear

Here, we empirically demonstrate that linear positional encoding is important. We train two COTR models with different positional encoding strategies; see Section 3.2. One model uses log-linear increase in the frequency of the sine/cosine function, and the other uses linear increase instead. Fig. A shows that COTR successfully converges using the linear increase strategy. However, as shown in Fig. B, COTR fails to converge with the commonly used log-linear strategy [73, 10]. We suspect that this is because the task of finding correspondences does not involve very high frequency components, but further investigation is necessary and is left as future work.

C. Architectural details for COTR

Backbone. We use the lower layers of ResNet50 [23] as our CNN backbone. We extract the feature map with 1024 channels after layer3, i.e., after the fourth downsampling step. We then project the feature maps with 1024 channels with a 1 x 1 convolution to 256 channels to reduce the amount of computation that happens within the transformers.

Transformers. We use 6 layers in both the transformer encoder and the decoder. Each encoder layer contains an 8-head self-attention module, and each decoder layer contains an 8-head encoder-decoder attention module. Note that we disallow the self-attention in the decoder, in order to maintain the independence between queries – queries should not affect each other.

MLP. Once the transformer decoder process the results, we obtain a 256 dimensional vector that represents where the correspondence should be. We use a 3-layer MLP to regress the corresponding point coordinates from the 256-dimensional latent vector. Each layer contains 256 neurons, followed by ReLU activations.

D. Architectural details for the MLP variant

Backbone. We use the same backbone in COTR. The difference here is that, once the feature map with 256 channels is obtained, we apply max pooling to extract the global latent vector for the image, as suggested in [21]. We also tried a variant where we do not apply global pooling and use a fully-connected layer to bring it down to a manageable size of 1024 neurons but it quickly provided degenerate results, where all correspondence estimates were at the centre.

MLP. With the latent vectors from each image, we use a 3 layer MLP to regress the correspondence coordinates. Specifically, the input to the coordinate regressor is a 768-dimensional vector, which is the concatenation of two global latent vectors for the input images and the positional encoded query point. Similarly to the MLP used in COTR, each linear layer contains 256 neurons, and followed by ReLU activations.

E. Comparing with RAFT [65]

RAFT [65] performs better in KITTI-type of scenarios, not necessarily so for other cases. To show this, we provide
results for RAFT [65] on all other datasets in Table A. On KITTI, sparse COTR still performs best, and with the interpolation strategy it is roughly on par with RAFT [65]. On other datasets, COTR outperforms RAFT [65] by a large margin\(^1\).

\(^1\)Note that RAFT [65] requires two input images of the same size. We resize them to 1024×1024 for HPatches and the Image Matching Challenge.

Table A. RAFT on ETH3D, KITTI, HPatches, and IMC2020.