DenseGAP: Graph-Structured Dense Correspondence Learning with Anchor Points

Zhengfei Kuang\textsuperscript{1,2}, Jiaman Li\textsuperscript{1,3}, Mingming He\textsuperscript{2}, Tong Wang\textsuperscript{1,2}, Yajie Zhao\textsuperscript{2}

\textsuperscript{1}University of Southern California
\textsuperscript{2}USC Institute for Creative Technologies
\textsuperscript{3}Stanford University

Abstract

Establishing dense correspondence between two images is a fundamental computer vision problem, which is typically tackled by matching local feature descriptors. However, without global awareness, such local features are often insufficient for disambiguating similar regions. And computing the pairwise feature correlation across images is both computation-expensive and memory-intensive. To make the local features aware of the global context and improve their matching accuracy, we introduce DenseGAP, a new solution for efficient Dense correspondence learning with a Graph-structured neural network conditioned on Anchor Points. Specifically, we first propose a graph structure that utilizes anchor points to provide sparse but reliable prior on inter- and intra-image context and propagates them to all image points via directed edges. We also design a graph-structured network to broadcast multi-level contexts via light-weighted message-passing layers and generate high-resolution feature maps at low memory cost. Finally, based on the predicted feature maps, we introduce a coarse-to-fine framework for accurate correspondence prediction using cycle consistency. Our feature descriptors capture both local and global information, thus enabling a continuous feature field for querying arbitrary points at high resolution. Through comprehensive ablative experiments and evaluations on large-scale indoor and outdoor datasets, we demonstrate that our method advances the state-of-the-art of correspondence learning on most benchmarks. All of our training and evaluation codes are available at https://formyfamily.github.io/DenseGAP/.

1 Introduction

Image correspondence is the foundation of many computer vision tasks, such as geometric matching \cite{28, 41, 40}, pose estimation \cite{39, 37}, visual localization \cite{53}, and optical flow \cite{28, 41, 40, 16}. Although being long explored, it remains an open question, especially for images under large appearance or view changes, or containing textureless or repetitive regions. The classic solution to this problem is based on keypoint detection and matching \cite{25, 24, 37}. This line of methods is highly efficient but limited by the missing-detection issue \cite{34}. Thus, the more recent works eliminate the dependency on keypoint detection by considering every point for matching and building dense correspondence.

Recent works on dense correspondence learning build 4D correlations between images using local features extracted for each point, followed by a neighbor consensus filtering strategy to select confident matches \cite{35, 34, 22, 21}. These methods are effective in finding denser matches but still suffer from two major limitations: (1) computing full points correlation is expensive and memory-intensive, especially on high-resolution images; (2) the extracted local features lack global context, making them indistinguishable in textureless or repetitive regions. The follow-up methods \cite{22, 45} adopt coarse-to-fine frameworks to reduce the computational cost but struggle with the small receptive fields. To overcome these problems, we propose the idea to utilize sparse correspondences as a bridge to connect every point in the global context. This is motivated by the human perception that distinguishes similar regions in the scene with global information obtained by a few salient points.

In this paper, we present a novel way of introducing sparse priors to dense correspondence learning using anchor points, a set of paired salient points corresponded across images. With these anchor points, we propose to learn a context-aware feature field for querying correspondence at arbitrary image positions. We adopt a graph representation that connects the anchor points to every image position so as to model different levels of context and propagate them to the whole graph. Based on this representation, to integrate the global information into the local features, we further design three simple but effective message-passing layers: the inter-points layer binds anchor points to introduce the inter-image correlation, the intra-points layer aggregates information among anchor points within an image and builds the intra-image context, and the point-to-image layer broadcasts the above global contexts to every point and fuses it with the local features. Utilizing the predicted features, we finally present a coarse-to-fine framework to learn accurate dense correspondence based on cycle consistency. Extensive ablative experiments and comparisons show that our learned feature descriptors effectively boost the performance of dense correspondence prediction. In particular, our approach can help in complex tasks such as surface normal prediction, depth estimation, and object detection, where global context plays a critical role in extracting point-level information.

Our main contributions are summarised as follows: Firstly, we propose to use anchor points as priors for dense
correspondence learning in a graph structure, which connects all local points in a global context. Secondly, we design a network based on the graph representation with three lightweight message-passing layers for propagating and aggregating multi-level context information. Finally, our novel dense correspondence prediction pipeline achieves state-of-the-art performance, which supports arbitrary correspondence query for high-resolution input images and effectively embeds the global context to the local feature descriptors.

2 Related Work

Image Correspondence. The well-adopted pipeline for establishing image correspondence usually consists of feature detection \([25, 17, 18, 30, 43]\), description \([4, 86, 19, 82, 29, 26, 27, 33, 12, 9]\), and matching \([37]\). The typical drawback of these detector-based methods is the missing-detection problem, which limits both the accuracy and the number of matches. To address the problem, detector-free approaches are explored. Some achieve feature matching by extracting features on a dense grid across the images \([35]\) and use coarse-to-fine frameworks to reduce memory footprint and improve fine-level matching \([34, 22, 53]\). However, these frameworks still require heavy computation of inter-image correlation and neglect the contextual cues. Another line of the detection-free methods \([28, 41, 40]\) aim to generate pixel-level correspondence and bridge correspondence learning and optical flow estimation. They work well for continuous frames but are inadequate to handle image pairs with large displacements. Very recently, the concurrent works \([39, 16]\) involve global context between matches by using transformers \([42]\) which achieve great success in many NLP and vision tasks \([11, 6, 51]\) based on the attention mechanism. Different from these works, we propose to adopt sparse correspondence as prior and design lightweight network layers to efficiently propagate the contextual information to all image points, allowing predicting dense correspondence for arbitrary points.

Graph-Structured Network. The graph-structured representation is applied in various domains, such as image \([20, 9]\), video \([44]\), skeleton \([50, 1]\) and mesh \([45, 49]\) due to the flexibility of this data structure. In recent years, more interests has put into relating graph representation with neural networks. The framework of graph neural network (GNN) is first proposed in \([38]\), which formulates as node, edges, message passing layers to assemble information from a graph structure. Inspired by GNN, other methods have applied graph networks to vision tasks such as image recognition \([47]\), object detection \([35]\), point cloud learning \([49]\) and so on. In this work, we introduce anchor points into our framework and bring the graph representation into correspondence learning. The graph representation is inspired by SuperGlue \([37]\) which introduces a graph neural network for matching sparse keypoints between images. But the key difference is that we propose a more sophisticated graph to model multi-level contexts using sparse correspondence as prior and develop a general architecture to infuse the contextual information into local features. In our graph-structured network, the message passing layers are implemented with the attention-based mechanism of Transformer \([42]\), which is also used by some recent works \([39, 16]\).

3 Method

3.1 Anchor Points

We propose to solve the problem of finding dense correspondence between a pair of images by first extracting a feature descriptor for arbitrary query points in one image and then using it to compute the correspondence in the other image. To efficiently encode the global information (e.g. inter- and intra-image context) in the feature descriptor, we introduce anchor points to bridge all the points across images. The anchor points are a set of corresponding points from a pair of images that usually specify spatial locations of the salient features (e.g. blobs, corners). They can be obtained by off-the-shelf sparse matching algorithms (e.g. \([37, 26]\)), serving as reliable priors and modeling global contexts. We aim to achieve the following goals using anchor points: (1) information exchanging across two images by anchor points; (2) global context gathering and broadcasting via anchor points within one image; and (3) reducing the cost by avoiding of building a full set correlation. Therefore, we build a graph with anchor points and image points as nodes, connecting them with directed edges. By applying the message-passing mechanism \([3]\), we achieve the information propagation in the graph and aggregation for each node.

Representation. Given a pair of images \(I_a\) and \(I_b\), and a normalized pixel coordinate \(x \in [0, 1]^2\) in \(I_a\) representing the query point, our target is to find its correspondence \(y \in [0, 1]^2\) in \(I_b\). To achieve it, we adopt the approach introduced in \([46]\) by extracting the feature descriptors \(F_a\) and \(F_b\) of both images and computing \(y\) as the expectation of the correlation-based distribution over \(I_b\):

\[
y = \sum_{y \in I_b} y \cdot \text{Softmax}_y(F_a(x)^T F_b(y)).
\]

Note that \(x \in [0, 1]^2\) means a continuous coordinate and \(x \in I_a\) indicates the pixel of \(I_a\). To learn \(F_a\) and \(F_b\), we use anchor points, \(\mathcal{A}_a \subset [0, 1]^2\), \(\mathcal{A}_b \subset [0, 1]^2\) and their one-to-one correspondence \(\mathcal{A} = \{(x, y)| x \in \mathcal{A}_a, y \in \mathcal{A}_b\}\) as prior. We connect them with the image points in a directed graph \(\mathcal{G}\), as shown in Fig.\(1\). In \(\mathcal{G}\), we first build nodes for \(\mathcal{A}\), i.e., \(\mathcal{V}_a = \{v_x|x \in \mathcal{A}_a\}\) and \(\mathcal{V}_b = \{v_y|y \in \mathcal{A}_b\}\), and nodes for all image points of \(I_a\) and \(I_b\), i.e., \(\mathcal{V}_a = \{v_x|x \in I_a\}\) and \(\mathcal{V}_b = \{v_y|y \in I_b\}\). Then we connect them by three types of directed edges which we denote as \((v_x, v_y)_c\):

\[
\begin{align*}
\mathcal{E}_c &= \{(v_x, v_y)|(s, r) \in \mathcal{A}\} \cup \{(v_x, v_y)|(s, r) \in \mathcal{A}\}, \\
\mathcal{E}_s &= \{(v_x, v_y)|v_x, v_r \in \mathcal{V}_x, v_r \in \mathcal{V}_y\}, \\
\mathcal{E}_i &= \{(v_x, v_y)|v_x \in \mathcal{V}_x, v_r \in \mathcal{V}_y\} \\
&\cup \{(v_x, v_y)|v_x \in \mathcal{V}_x, v_r \in \mathcal{V}_y\}.
\end{align*}
\]

\(\mathcal{E}_c\) indicate inter-points edges between anchor points from both images for inter-image communication; \(\mathcal{E}_s\) represent intra-points edges between anchor points within the same image for intra-image communication; \(\mathcal{E}_i\) are points-to-image edges from anchor points to image points,
used to broadcast the information from anchor points to everywhere. Thus, the graph is represented as \( G = (V = \{V_a, V_b, V_c, V_g\}, E = \{E_c, E_s, E_i\}) \).

We build a neural network based on this graph structure. Inspired by message-passing concept in graphical models, we design a message-passing layer for each type of edges \( E' \in E \) as follows:

\[
z^a_r = \sum_{(v_s, v_r) \in E'} \alpha_{E'}(z^{in}_s) \cdot \beta_{E'}(z^{in}_s, z^{in}_r)
\]

\[
z^{out}_r = \rho_{E'}(z^{in}_r, z^a_r).
\]  

In detail, the message-passing layer first reprojects the input node attributes \( z^{in}_s \) by the function \( \alpha_{E'} \), and then calculates the messages passed through the edges \((v_s, v_r)\) by the function \( \beta_{E'} \). Finally, it aggregates all information sent to the target node \( v_r \) (denoted as \( z^a_r \)) and outputs the updated attributes by a feed-forward function \( \rho_{E'} \). All the functions in Eq. [3] vary according to the edge types.

### 3.2 Message-Passing Layers

#### Inter-Points Message-Passing Layer

This layer updates the features of anchor points by aggregating messages along the edges \( E_s \). Each node is connected to all the others by \( E_s \) within the image, forming a complete subgraph.

We update the node attribute based on the multi-head attention (MHA) used in [42], which has been proved a highly effective neural architecture in various mainstream vision tasks [6, 11], including building sparse correspondence [37]. In our setting, each node is updated based on a weighted sum over its neighbours during the aggregation step (Fig. [2]). For each node \( v_r \) connected by a set of incoming edges \((v_s, v_r)\), the first step is to generate the query vector \( Q^h_r \) from \( z_r \), and the key \( K^h_s \) and value vector \( V^h_s \) from \( z_s \) for each head \( h \).

Then, in each head, we sum up the value of all \( z_s \) weighted by the attention \( A^h_{s,r} \), calculated using the query and key vectors. Finally, we concatenate the aggregated value of all heads and use a feed-forward network \( F_{out} \) (a two-layer MLP) to refine the output. Specifically,

\[
Q^h_r = W_q^h z_r, \quad [K^h_s, V^h_s] = [W_k^h, W_v^h] z_s,
\]

\[
A^h_{s,r} = \text{Softmax}_s \left( (K^h_s)^T Q^h_r / \sqrt{d_k} \right),
\]

\[
V^h_r = \sum_s A^h_{s,r} \cdot V^h_s,
\]

\[
z^r_{out} = F_{out}(z_r + W_{out}(V^0_r \odot V^1_r \odot \ldots \odot V^h_r)),
\]

where \( W_q, W_k, W_v, W_{out} \) are weight matrices, and \( d_k \) is the dimension of \( K^h_s \).

We adapt this attention model to our message passing layer, where the functions in Eq. [3] for this layer are defined as:

\[
\alpha^h_{E_s}(z_s) = V^h_s,
\]

\[
\beta^h_{E_s}(z_s, z_r) = A^h_{s,r},
\]

\[
\rho_{E_s}(z_r, z_a) = F_{out}(z_r + W_{out}(z^0_r \odot z^1_r \odot \ldots \odot z^h_r)).
\]

#### Points-to-Image Message-Passing Layer

This layer is designed to propagate the information learned by anchor points to all image points along the edges \( E_i \). Each pair of
anchor point and image point are connected by only one directed edge in \( \mathcal{E}_a \) and form a complete bipartite subgraph. The functions for \( \mathcal{E}_a \) are the same as ones for \( \mathcal{E}_s \) (Eq. 6), while only the image points have their features updated in this layer by aggregating the updates from anchor points.

### 3.3 Graph-Structured Network

Based on the graph structure, we design a network to learn feature descriptors conditioned on the input images \((I_a, I_b)\) and anchor points \((A_a, A_b)\). The network contains two modules and updates the features in a coarse-to-fine manner. The propagation module integrates all message-passing layers and updates the features at the coarse level with larger receptive fields, which effectively reduces the computation cost while efficiently capturing global priors. The refinement module combines the updated coarse features and the local features at fine level to preserve the local structure details.

As shown in Fig. 3, we first initialize local features at coarse and fine levels using a typical convolutional neural network (CNN), denoted as \( F^c \) and \( F^f \). Then we compute the features of anchor points by bilinearly interpolating \( F^c \) and obtain the features \( F_{a,b}^c \) and \( F_{a,b}^f \) for anchor points in \( I_a \) and \( I_b \) respectively. Together with \( F^c_a \) and \( F^c_b \), they form the input of the propagation module, indicating the initial attributes of the nodes in \( \mathcal{V}_x, \mathcal{V}_y, \mathcal{V}_a, \mathcal{V}_b \), which will be updated by the message-passing layers. The propagation module consists of \( N_f \) intra-points message-passing layers and \( N_f \) inter-points message-passing layers, followed by one point-to-image message-passing layer. We alternate the inter- and intra-points message-passing layers, starting with one inter-points message-passing layer. In addition, in each of the message-passing layers, we concatenate the node attributes with 2D position embeddings, which are calculated by the 2D sinusoidal position encoding method proposed in [6]. With the unique positional information, the learned features are position-dependent and more robust against matching ambiguity in indistinctive or textureless regions. This module finally outputs two updated coarse features \( F_{a,b}^{c'} \) and \( F_{a,b}^{f'} \), and feed them to the refinement module. In the refinement module, we bilinearly upsample \( F_{a,b}^{c'} \) and \( F_{a,b}^{f'} \) to fine level, concatenate them with the corresponding fine features and finally feed them to one convolutional layer to generate the result \( F_{a,b}^{f''} \) and \( F_{a,b}^{f''} \).

### 3.4 Coarse-to-Fine Training Strategy

Since our network predicts both coarse- and fine-level features, we use a coarse-to-fine matching strategy introduced by [46] to compute the correspondence at a lower resolution followed by a local refinement at a finer scale. Given a query point \( x \) in \( I_a \), we first find its coarse correspondence \( y_c \) using \( F_{a,b}^{c'} \) and then crop a local window centered at \( y_c \) in \( F_{a,b}^{f''} \), extracting the final correspondence \( y \) within the window.

#### Losses.

To train the model given an image pair, we randomly sample the query point \( x \) from the pixels that can find ground-truth correspondences on the other image. For the set of training pairs \( Q = \{(x, y_{gt})\} \), the loss function is defined as the error between the established matches and ground-truth correspondence:

\[
L = \sum_{(x,y_{gt}) \in Q} \frac{1}{\sigma_y} (\|y_{gt} - y_c\|_2 + \|y_{gt} - y\|_2),
\]

where \( \sigma_y \) is the uncertainty of the prediction proposed in [46]. Besides, the anchor points are also randomly sampled from the points with known ground-truth correspondence. Meanwhile, we design a grid filter to make them evenly distributed. For more details, please refer to the supplementary material.

#### Adaptive Position Embedding.

When training with fixed-size images, the learned model will degrade when testing with size-free images. To address this problem, we propose a simple and efficient method to augment the pixel coordinates with a random scale for each image. Specifically, in every training iteration, we assign a scale \( r_a = (r_{a1}, r_{a2}) \) for \( I_a \), and \( r_b = (r_{b1}, r_{b2}) \) for \( I_b \). Then, for every point \( x = (x_1, x_2) \) from \( I_a \), we scale it to \((x_1 \cdot r_{a1}, x_2 \cdot r_{a2})\) before feeding it to the position encoder, and apply to \( I_b \) in the same way. This method significantly improves the result in size-free evaluations.

### 3.5 Runtime Correspondence Prediction

At inference, we first extract anchor points for both input images, and then feed them to our graph-structured network to generate the feature maps. For any query point in \( I_a \), we use the same coarse-to-fine method as training to compute its correspondence in \( I_b \). Although we use the ground-truth correspondence anchor points for training, our model can adapt well to the anchor points generated by other sparse matching methods at inference. In our experiments, we use SuperGlue [37] which can efficiently produce strong matches. In addition, for any query points, we propose a metric based on cycle consistency to measure the confidence of its correspondence, which can be used to filter the matches. Given \( x \) in \( I_a \), we find its correspondence \( y \) in \( I_b \) and then go back and search \( y \)'s correspondence in \( I_a \) to get the result \( x' \). The
Figure 3: Overview of the framework. Given two images and anchor points, we first extract the coarse and fine feature maps of each image. Then we obtain the features of anchor points from the coarse feature maps as input to the Propagation Module. The output of the module is updated coarse feature maps, and is then fed with the fine feature maps to the Refinement Module. This module finally generates the updated fine feature maps.

cycle consistency is defined as the euclidean distance between $x$ and $x'$.

4 Experiments

This section starts with the training dataset and implementation details, followed by the evaluations of our approach on different applications. Finally, we conduct a comprehensive ablation study of the proposed network structure and demonstrate its effectiveness and robustness.

Training Datasets. Our model is trained with MegaDepth [23] and ScanNet [8] for outdoor and indoor scenes respectively. The former consists of over 600,000 preprocessed image pairs introduced by CAPS [46]. We follow the same training and validation split with 130 scenes for training and 37 for validation. ScanNet is a large-scale indoor dataset with 1513 training scenes and 100 testing scenes. We use the same training and testing image pairs as [37] in our experiments.

Implementation Details. We adopt a modified ResNet-18 [14] as backbone to extract feature maps. The size of coarse and fine feature maps are $\frac{1}{8}$ and $\frac{1}{2}$ of the input image size respectively. We set the number of layers $N_l$ to 4 and attention heads to 4 as well. While searching the correspondence in fine-level features, we set the window size as $\frac{1}{8}$ of the fine-level feature map’s size. The learning rate is initially set to 0.0001, and reduced by half after each 50K iterations. We train our model for 120K iterations and fine-tune it with anchor points predicted by SuperGlue [37] for additional 20K iterations. When testing, we limit the number of input anchor points to 500. For more details about the implementation, please refer to the supplementary material.

Performance. We use PyTorch [32] to train our model on a single NVIDIA Tesla V100 with batch size 5, which requires approximately 28 GB memory. The total training time is about 36 hours. As for testing, it takes about 173 milliseconds and 2.9 GB GPU memory with a NVIDIA GeForce GTX 1080 Ti to generate feature maps and solve 2000 query points on a $480 \times 640$ image.

4.1 Image Matching

Dataset and Evaluation Metrics. HPatches is a benchmark dataset with 108 image sequences for evaluating the image matching accuracy. Each sequence contains one query image and five reference images with either changing illumination or viewpoints (52 sequences for illumination and 56 sequences for viewpoint). We follow D2-Net [12] and use mean matching accuracy (MMA) as our evaluation metrics. The MMA is calculated as the percentage of corrected matches in sampled query points within a given threshold against ground truth matching.

Results. In this evaluation, we use keypoints extracted by SIFT [25] as our query points. We filter out all correspondences with cycle consistency larger than 5 pixels, and select top 2000 matches for each image pair. We compare our result with R2D2 [33], D2-Net [12], CAPS [46], SuperGlue [37], LofTR [39] and DualRC-Net [22]. Our model achieves the best overall performance with a large number of correspondences as shown in Fig. 4.

4.2 Geometric Matching

Dataset and Evaluation Metrics. We use the percentage of correct keypoints (PCK) metric to evaluate our model. We consider a correspondence as correct if it is close enough (e.g. within a given threshold) to the ground truth. Both HPatches [2] (viewpoint sequences only) and MegaDepth [23] are used for this evaluation. Following [46], we densely sample correspondences between test image pairs and evaluate the PCK score on them.

Results. We show the results of our model and the state-of-the-art methods (GOCor [40], GLU-Net [41] and CAPS [46]) in Fig. 5. For a fair comparison, we also train CAPS [46] with our loss function, and test it separately.
2
2
2
2
9
6
4
3
10
4
5
3
5
7
6
6
8
3
8
9
8
10
7
9
4
7
10
5
26x168
MMA
35x54
0.0
35x109
0.2
35x163
0.4
35x217
0.6
35x272
0.8
35x326
1.0
54x471
w/
54x471
(labeled as CAPS
different methods.
Figure 4: HPatches evaluation. Left: MMA comparison with previous work. Right: The mean number of correspondences for
ScanNet [8], we adopt the same sampling strategy as in
SIFT [25] and SuperPoint [10] predict limited keypoints on
we change the way of generating query points. Since both
Glue [37]. The setting of indoor evaluation is similar except
we compare our results with DualRC-Net [22] and Super-
els. After that, the top 8000 points are selected. In Tab. 1,
response points with cycle consistency larger than 5 pix-
generated from SIFT [25], and we remove the predicted cor-
Cor [26] and Ratio [25]. We adopt the same metrics as [37], where we report the area under cum-
DenseGAP* removes both intra-points and inter-
Sec 4.2 to produce dense points. We compare our model
with other methods in Tab. 2 and show qualitative results in
Fig. 6. In both cases, our model outperforms the baseline
methods by a large margin. We attribute it to two factors:
(1) our model is robust to the outliers in anchor points; (2)
our model can produce much denser correspondence which
greatly boosts the performance of RANSAC pose estimation
for a more unbiased result.

4.4 Ablation Study
We conduct two ablation studies on MegaDepth with the
PKC metric, as shown in Fig. 7. In the first study, we test the
performance of the proposed model under different settings.
It consists of four variants: (1) Model w/o graph removes
the graph-structured representations and only preserves the
local feature extractor; (2) Model with lower resolution (LR)
changes the resolution of feature maps to 1/4 and 1/16 of
the image size; (3) & (4) Models with 200/100 anchor
points(AP) reduce the number of anchor points to 200 and
100, which shows the effect of limited anchor points on the
performance. The first two variants decrease the score in dif-
ferent patterns, which indicates the feasibility and inevitabil-
ity of our design. Reducing the number of anchor points does
not affect the result much until a large quantity of anchor
points are removed, showing that our model is robust to the
number of initial anchor points.

The second study focuses on the effectiveness of our
message-passing layers. We first design two variants of our
model and test them using the original setting: (1) Model
w/o intra removes the intra-points message-passing layer;
and (2) Model w/o point removes both intra-points and inter-

Figure 4: HPatches evaluation. Left: MMA comparison with previous work. Right: The mean number of correspondences for
different methods.

Table 1: Outdoor Pose Estimation Evaluation.

| Method             | AUC(5) | AUC(10) | AUC(20) |
|--------------------|--------|---------|---------|
| DualRC-Net [22]    | 32.56  | 47.60   | 64.43   |
| SP [10]+SuperGlue [37] | 34.81  | 50.46   | 64.43   |
| DenseGAP           | 41.17  | 56.87   | 70.22   |

(labeled as CAPS w/ Dense). DenseGAP significantly out-
performs other methods on the MegaDepth dataset and
achieves a comparable result on the HPatches dataset with
GOCor [40] and GLU-Net [41]. Both methods we think
are naturally well-fitted to predict displacements that can be
interpolated bilinearly, such as the Homography space in
HPatches. However, in a more general scenario with real-
world non-planar objects (MegaDepth), our DenseGAP out-
performs them by learning the distinctive features of each
query point. Moreover, DenseGAP has remarkable improve-
ments comparing to CAPS [46], which uses a similar coarse-
to-fine strategy to generate correspondences, thus indicating
the effectiveness of our graph-structured network.

4.3 Relative Pose Estimation
Datasets and Evaluation Metrics. We conduct evalua-
tions on pose estimation using MegaDepth [23] for outdoor
scenes and ScanNet [8] for indoor scenes respectively. We
randomly select 2459 image pairs from 37 validation scenes
for evaluation on outdoor scenes. As for the indoor, we use
1500 image pairs of ScanNet [8] provided by [37]. We adopt
the same metrics as [37], where we report the area under cum-
ulative error curve (AUC) of pose error up to thresholds
5°, 10°, 20°. The relative pose between images is estimated by
applying RANSAC [13] on predicted correspondences.

Results. In the outdoor evaluation, We use the output of
SuperGlue [37] as our anchor points. Our query points are
generated from SIFT [25], and we remove the predicted cor-
respondence points with cycle consistency larger than 5 pix-
els. After that, the top 8000 points are selected. In Tab. 1
we compare our results with DualRC-Net [22] and Super-
Glue [37]. The setting of indoor evaluation is similar except
we change the way of generating query points. Since both
SIFT [25] and SuperPoint [10] predict limited keypoints on
ScanNet [8], we adopt the same sampling strategy as in

Table 2: Indoor Pose Estimation Evaluation. The asterisk (*)
indicates the model is trained on MegaDepth [23].

| Method               | AUC(5) | AUC(10) | AUC(20) |
|---------------------|--------|---------|---------|
| ORB [36]+GMS [5]    | 5.21   | 13.65   | 25.36   |
| D2-Net [12]+NN      | 5.25   | 14.53   | 27.96   |
| ContextDesc [26]+Ratio [25] | 6.64   | 15.01   | 25.75   |
| DualRC-Net [22]*    | 6.94   | 17.06   | 29.58   |
| SP [10]+SuperGlue [37] | 16.16  | 33.81   | 51.84   |
| DenseGAP*           | 16.93  | 34.85   | 53.16   |
| DenseGAP            | 17.01  | 36.07   | 55.66   |
Figure 5: Dense geometric matching evaluation. (a) Comparison of PCK scores. (b) Qualitative results on HPatches. Row 1: Left: Input Image, Right: Error map of dense correspondences, and anchor points (colored in green) generated by SuperGlue [37]. We calculate the reprojection errors of each query point and generate the error map using bilinear interpolation. Row 2: Correspondences between two images (indicated by different colors). The error bar on the right is only used for the error map. Note that we use the officially released pretrained model of GOCor [40] and GLU-Net [41] in this experiment.

Figure 6: Correspondences are colored by their epipolar error calculated from ground truth relative poses (Green means inliers, and red means outliers). We set the error threshold to $1 \times 10^{-3}$ for both indoor scenes and outdoor scenes. For DualRC-Net [22] and DenseGAP, we demonstrate top 500 correspondences in the indoor scene and 1000 correspondences for the outdoor scene.

Figure 7: Our model in different settings (left) and using different combinations of message-passing layers (right).

5 Conclusion

We propose a novel dense image correspondence learning approach that utilizes anchor points with a graph-structured network. With the contextual information by anchor points, the feature descriptors that fuse local information can be predicted for any query point, intra-image global context, and inter-image correlation. Our model is demonstrated to outperform the state-of-the-art on multiple tasks. The extensive ablation studies show the effectiveness of each network design and the robustness of anchor point selection. An end-to-end solution with joint anchor points detection and correspondence learning could be an interesting future direction. Finally, although this work focuses on dense correspondence, we believe it has the generalization capacity to various other tasks such as normal estimation and optical flow.
[28] Melekhov, I.; Tulpin, A.; Sattler, T.; Pollefeys, M.; Rahtu, E.; and Kannala, J. 2019. DGC-Net: Dense Geometric Correspondence Network. In IEEE Winter Conference on Applications of Computer Vision, WACV 2019, 1034–1042. IEEE.

[29] Mishchuk, A.; Mishkin, D.; Radenovic, F.; and Matas, J. 2017. Working hard to know your neighbor's margins: Local descriptor learning loss. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4826–4837.

[30] Mishkin, D.; Radenovic, F.; and Matas, J. 2018. Repeatability Is Not Enough: Learning Affine Regions via Discriminability. In Computer Vision - ECCV 2018, volume 11213 of Lecture Notes in Computer Science, 287–304. Springer.

[31] Noh, H.; Araujo, A.; Sim, J.; Weyand, T.; and Han, B. 2017. Large-Scale Image Retrieval with Attentive Deep Local Features. In IEEE International Conference on Computer Vision, ICCV 2017, 3476–3485. IEEE Computer Society.

[32] Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Köpf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8024–8035.

[33] Revaud, J.; de Souza, C. R.; Humenberger, M.; and Weinzaepfel, P. 2019. R2D2: Reliable and Repeatable Detector and Descriptor. In Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 12405–12415.

[34] Rocco, I.; Arandjelovic, R.; and Sivic, J. 2020. Efficient Neighbourhood Consensus Networks via Submanifold Sparse Convolutions. In Computer Vision - ECCV 2020, volume 12354 of Lecture Notes in Computer Science, 605–621. Springer.

[35] Rocco, I.; Cimpoi, M.; Arandjelovic, R.; Torii, A.; Pajdla, T.; and Sivic, J. 2018. Neighbourhood Consensus Networks. In Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 1658–1669.

[36] Rublee, E.; Rabaud, V.; Konolige, K.; and Bradski, G. R. 2011. ORB: An efficient alternative to SIFT or SURF. In IEEE International Conference on Computer Vision, ICCV 2011, 2564–2571. IEEE Computer Society.

[37] Sarlin, P.; DeTone, D.; Malisiewicz, T.; and Rabinovich, A. 2020. SuperGlue: Learning Feature Matching With Graph Neural Networks. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, 4937–4946. IEEE.

[38] Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2009. The Graph Neural Network Model. IEEE Trans. Neural Networks, 20(1): 61–80.

[39] Sun, J.; Shen, Z.; Wang, Y.; Bao, H.; and Zhou, X. 2021. LoFTR: Detector-Free Local Feature Matching with Transformers. CoRR, abs/2104.00680.

[40] Truong, P.; Danelljan, M.; Gool, L. V.; and Timofte, R. 2020. GOCor: Bringing Globally Optimized Correspondences Volumes into Your Neural Network. In Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020.

[41] Truong, P.; Danelljan, M.; and Timofte, R. 2020. GLU-Net: Global-Local Universal Network for Dense Flow and Correspondences. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, 6257–6267. IEEE.

[42] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is All you Need. In Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, 5998–6008.

[43] Verdie, Y.; Yi, K. M.; Fua, P.; and Lepetit, V. 2015. TILDE: A Temporally Invariant Learned Detector. In IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, 5279–5288. IEEE Computer Society.

[44] Vicol, P.; Tapaswi, M.; Castrejon, L.; and Fidler, S. 2018. MovieGraphs: Towards Understanding Human-Centric Situations From Videos. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, 8581–8590. IEEE Computer Society.

[45] Wang, N.; Zhang, Y.; Li, Z.; Fu, Y.; Liu, W.; and Jiang, Y. 2018. Pixel2Mesh: Generating 3D Mesh Models from Single RGB Images. In Computer Vision - ECCV 2018, volume 11215 of Lecture Notes in Computer Science, 55–71. Springer.

[46] Wang, Q.; Zhou, X.; Harriharan, B.; and Snavely, N. 2020. Learning Feature Descriptors Using Camera Pose Supervision. In Computer Vision - ECCV 2020, volume 12346 of Lecture Notes in Computer Science, 757–774. Springer.

[47] Wang, X.; Ye, Y.; and Gupta, A. 2018. Zero-Shot Recognition via Semantic Embeddings and Knowledge Graphs. In 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, 6857–6866. IEEE Computer Society.

[48] Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic Graph CNN for Learning on Point Clouds. ACM Trans. Graph., 38(5): 146:1–146:12.

[49] Wen, C.; Zhang, Y.; Li, Z.; and Fu, Y. 2019. Pixel2Mesh++: Multi-View 3D Mesh Generation via Deformation. In 2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, 1042–1051. IEEE.

[50] Yan, S.; Xiong, Y.; and Lin, D. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), 7444–7452. AAAI Press.

[51] Yang, F.; Yang, H.; Fu, J.; Lu, H.; and Guo, B. 2020. Learning Texture Transformer Network for Image Super-Resolution. In 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, 5790–5799. IEEE.

[52] Yi, K. M.; Trulls, E.; Lepetit, V.; and Fua, P. 2016. LIFT: Learned Invariant Feature Transform. In Computer Vision - ECCV 2016, volume 9910, 467–483. Springer.

[53] Zhong, Q.; Sattler, T.; and Leal-Taixé, L. 2020. Patch2Pix: Epipolar-Guided Pixel-Level Correspondences. CoRR, abs/2012.01909.