PatchMatch-Based Neighborhood Consensus for Semantic Correspondence

Jae Yong Lee[1] Joseph DeGol[2] Victor Fragoso[2] Sudipta N. Sinha[2]

[1] University of Illinois [2] Microsoft

Abstract

We address estimating dense correspondences between two images depicting different but semantically related scenes. End-to-end trainable deep neural networks incorporating neighborhood consensus cues are currently the best methods for this task. However, these architectures require exhaustive matching and 4D convolutions over matching costs for all pairs of feature map pixels. This makes them computationally expensive. We present a more efficient neighborhood consensus approach based on PatchMatch. For higher accuracy, we propose to use a learned local 4D scoring function for evaluating candidates during the PatchMatch iterations. We have devised an approach to jointly train the scoring function and the feature extraction modules by embedding them into a proxy model which is end-to-end differentiable. The modules are trained in a supervised setting using a cross-entropy loss to directly incorporate sparse keypoint supervision. Our evaluation on PF-PASCAL *and* SPAIR-71K *shows that our method significantly outperforms the state-of-the-art on both datasets while also being faster and using less memory.*

1. Introduction

Computing pixel correspondence in two or more images is a fundamental step in computer vision tasks ranging from 3D vision [1, 7, 13, 19, 43, 51] to image editing [21, 2, 3, 8, 12, 52] and scene understanding [11, 36, 33]. The problem variants where the images depict the same scene (e.g., stereo, optical flow, and wide baseline matching) are extensively studied and many methods already exist [39, 4, 14, 42, 44, 16]. We address the dense semantic correspondence task [36, 28, 54, 22] where the two input images depict common visual concepts. The goal is to find corresponding pixels for semantically related object parts or scene elements as shown in Figure 1. Large intra-class appearance and shape variations make semantic correspondence challenging, and it continues to receive much attention from the community [46, 58, 47, 49, 33, 34].

The top performing methods for computing semantic correspondences rely on neighborhood consensus, which refers to a variety of techniques for filtering sparse feature matches or dense pixel correspondence based on 2D spatial context. While hand-crafted neighbourhood consensus filters have long been in use [50, 6, 35], Rocco *et al*. proposed NC-Net [49], the first trainable neighborhood consensus neural network containing multiple 4D convolutional layers for robust match filtering. ANC-Net [34] proposed a similar model with adaptive non-isotropic 4D convolution kernels. However, both methods sacrifice computational efficiency in favor of accuracy. The multiple 4D convolution layers in these models cause high memory usage and high

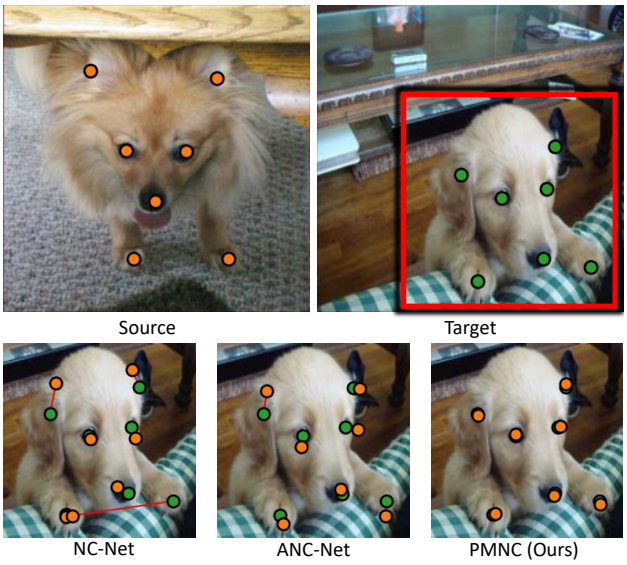

Figure 1: The top row shows an image pair from PF-PASCAL with ground truth keypoint annotations. The bottom row shows transferred keypoints on the target image computed using NC-Net [46], ANC-Net [34], and our method (PMNC). Errors are shown using red lines connecting the predicted orange keypoints to the expected green keypoints. PMNC outperforms NC-Net and ANC-Net on this example and produces state-of-the-art results on the PF-PASCAL and SPAIR-71K datasets.

Code available at http://github.com/leejaeyong7/PMNC

running times during the forward pass.

We take a different approach to neighborhood consensus inspired by PatchMatch [3]. We call our method PatchMatch Neighborhood Consensus (PMNC). Similar to NC-Net [49] and ANC-Net [34], PMNC uses a CNN feature backbone and computes 4D correlations to compare all feature map pixel pairs in the two images. However, unlike NC-Net and ANC-Net, PMNC does not filter the full 4D correlation map using multiple 4D convolutional layers. Instead, it uses PatchMatch-based inference on the 4D correlation map. Conventional PatchMatch cannot easily process the 4D correlation tensors for neighborhood consensus. Therefore, we propose a modified PatchMatch method, where we introduce a learned scoring function for comparing the similarity of patches in the two images. This scoring function performs 4D convolution locally on the 4D correlation map to produce a matching score. We invoke this function only at selective locations in the 4D map. The function is used to compare a small number of propagation candidates for each pixel during each PatchMatch iteration. In practice, PMNC computes 4D convolutions on a fraction of the full 4D search space which makes it more efficient.

Because PMNC is non-differentiable, it is difficult to train the CNN backbone and the scoring function using backpropagation. We overcome this by devising a differentiable proxy model into which we embed our local scoring function and feature backbone modules. While training this proxy model, we invoke the scoring function densely in the target image for a small number of 2D locations in the source image (locations where ground truth keypoints are available). The sparse ground truth keypoint positions in the target image are relaxed to 2D probability maps (similar to ANC-Net [34]). Then, the parameters of the feature backbone and the scoring function are jointly optimized to minimize the deviation between the predicted and ground truth probability maps. With this training scheme, we achieve state-of-the-art results on both the PF-PASCAL and SPAIR-71K datasets while also being faster than the state-of-the-art solutions and requiring less memory.

In summary, our **contributions** are (1) PMNC, a novel PatchMatch-inspired method that avoids exhaustive 4D convolutions but still allows the benefits of learned neighborhood consensus; (2) a simple approach to train PMNC that uses a proxy model and sparse keypoint supervision to jointly train the neighborhood consensus function and CNN feature modules; and (3) extensive experiments with standard datasets against current dense semantic correspondence benchmarks that show that our method achieves the best accuracy more quickly with less memory.

2. Related Work

PatchMatch. PatchMatch [3] is a randomized algorithm proposed to accelerate correspondence search in image editing tasks while exploiting 2D spatial coherence and smoothness inherent in image correspondence problems. It was later adopted for stereo matching [7], multi-view stereo [20, 51], and optical flow estimation [59, 40]. Since then, a few notable extensions of PatchMatch have been proposed. Hu *et al*. [25] incorporated a coarse-to-fine matching strategy for handling larger displacements. Galliani *et al*. [20] introduced parallelism in the propagation step inspired by belief propagation. Duggal *et al*. [15] proposed DeepPruner based on differentiable PatchMatch for stereo matching. We propose an alternative to differentiable PatchMatch and show its efficacy on semantic matching, a task for which we were unable to train a DeepPruner model. One key difference is that we use PatchMatch to improve neighborhood consensus (NC) methods to avoid expensive exhaustive 4D convolutions. We use PatchMatch only during inference along with a learned function.

Semantic correspondence via optimization. In early work on semantic correspondence estimation [36, 10, 9, 28, 26], handcrafted local features such as SIFT [39] and HOG [11] were used for dense matching along with various discrete optimization techniques. Cho *et al*. [10, 9] formulated sparse semantic matching as graph matching. Liu *et al*. [36] proposed SIFT Flow, generalizing optical flow to dense semantic flow. Such methods were extended to deal with large variations in scale [55] using deformable spatial pyramids [28, 26]. Taniai *et al*. [54] proposed jointly solving cosegmentation and semantic correspondence and showed that solving both tasks together led to higher accuracy on both tasks. Ham *et al*. [22] studied using sparse annotations for semantic flow estimation in Proposal Flow and released two datasets (PF-PASCAL, PF-WILLOW) for semantic correspondence. Kim *et al*. [30] used PatchMatch in a similar way as us but do not incorporate learning. Liu *et al*. [37] formulated semantic flow as an optimal transport problem and reported excellent results on SPAIRS-71K.

Semantic correspondence via learning. Long *et al*. [38] showed that CNN-based models trained for image classification learned features that could compute correspondence at finer image scales. Recently, learned methods for semantic correspondence based on CNNs have become popular [23, 29, 58, 41, 33]. Min *et al*. [41] also released SPAIR-71K, a challenging dataset with 71K images for the semantic correspondence task. Rocco et al. [46] proposed a trainable CNN-based model with geometric matching constraints. Jeon *et al*. [27] proposed Guided Semantic Flow (GSF) which extracts a sparse set of confident matches and uses them to guide correspondence search in ambiguous image regions. GSF and concurrent work ANC-Net [34] (see later) are the two top methods on PF-PASCAL.

Filtering Matches via Neighborhood Consensus. Neighborhood consensus matching was first explored for sparse feature matching [61, 50] to check whether the set of matches were locally spatially coherent in both images.

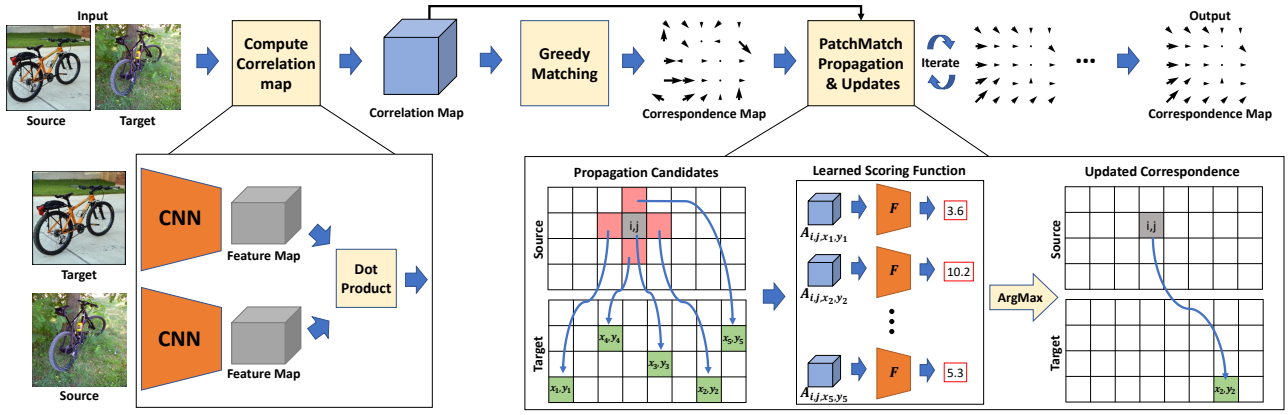

Figure 2: **Overview of inference phase**. PMNC first computes the feature maps of the source and target images using a CNN-based feature extractor. Then, it computes a 4D correlation map by computing the similarities (dot products) of all pairs of source and target features. Next, PMNC computes an initial correspondence map from the correlation map by associating a source pixel with the target pixel with the highest feature similarity. Then, PMNC iteratively refines the correspondence map by running a sequence of PatchMatch propagation-and-update steps (see the sub-figure in the bottom right). PMNC returns the refined correspondence map as the final output. PMNC avoids the use of exhaustive 4D convolutions over the correlation map by using PatchMatch propagation-and-update steps. At each step, for a source pixel (i, j) (shown in grey), PMNC (1) proposes candidates (x, y) in the target (shown in green); (2) evaluates each candidate using the scoring function F by analyzing a 4D local correlation patch A_{ijxy}; and (3) updates the correspondence with the best score.

Bilateral functions for modeling smooth image motion [35] and an approach using grid motion statistics [5] are other techniques for filtering matches using the same insight. Rocco *et al*. [49] proposed a 4D convolution based neighborhood consensus layer and end-to-end trainable model called NC-Net. The 4D filtering of correlation maps can better disambiguate local patches, but NC-Net is computationally expensive and limited to matching low resolution images. Rocco *et al*. [48] recently proposed Sparse NC-Net that uses fewer 4D convolutions by sparsely sampling the 4D correlation map. It uses K-nearest neighbors on dense correlation tensors to find good match candidates and limits 4D filtering to those candidates. However, Sparse NC-Net was specifically designed for wide-baseline matching and was not evaluated for dense semantic correspondence.

ANC-Net is a recent extension of NC-Net proposed by Li *et al*. [34] for semantic correspondence tasks. ANC-Net consistently outperforms NC-Net on both PF-PASCAL and SPAIRS-71K. The authors attribute this improvement to using self-similarity and adaptive window sizes in their neighborhood consensus modules and to training their model in a supervised setting. GSF and ANC-Net were published concurrently and are the top two methods on both PF-PASCAL and SPAIR-71K. We set the new state-of-the-art on both datasets and outperform ANC-Net and GSF.

While our work is similar to ANC-Net because we incorporate neighborhood consensus and directly use sparse keypoint supervision, there are a couple of notable differences. First, we use cross entropy loss to compare the predicted and ground truth 2D probability maps during training. In contrast, ANC-Net uses L2 loss to minimize a different measure of distributional difference. They add a second loss to encourage one-to-one matching but we do not use it. Second, our learned 4D scoring function is applied multiple times in an iterative fashion at selective locations in the 4D feature correlation map. However, ANC-Net applies the learned 4D filters on two 4D tensors, the first of which is the same correlation map as ours but the second one is computed by their self-similarity module.

3. PatchMatch Neighborhood Consensus

Figure 2 shows how PMNC computes the semantic correspondences at inference time. First, PMNC calculates a correlation map by exhaustively computing the similarities between source and target images using their image-patch feature descriptors. PMNC then greedily computes the initial correspondence map by composing it with the correspondences with the largest similarity score via the $\arg\max$ operator. Lastly, PMNC runs a few PatchMatch [7] iterations to refine and produce the final correspondence map.

3.1. Correlation Map and Correspondence Map

The first step is the computation of the correlation map encoding all the similarities between the source and target image patches. In Figures 2 and 3 (lower left box), we illustrate how PMNC computes the correlation map. We use a ResNet-101 [24] backbone as a feature extractor for both inference and training. We denote the source image as S and target image as T. Let $f^S \in \mathbb{R}^{H\times W\times d}$ and $f^T \in \mathbb{R}^{H\times W\times d}$ denote the feature maps of the source and

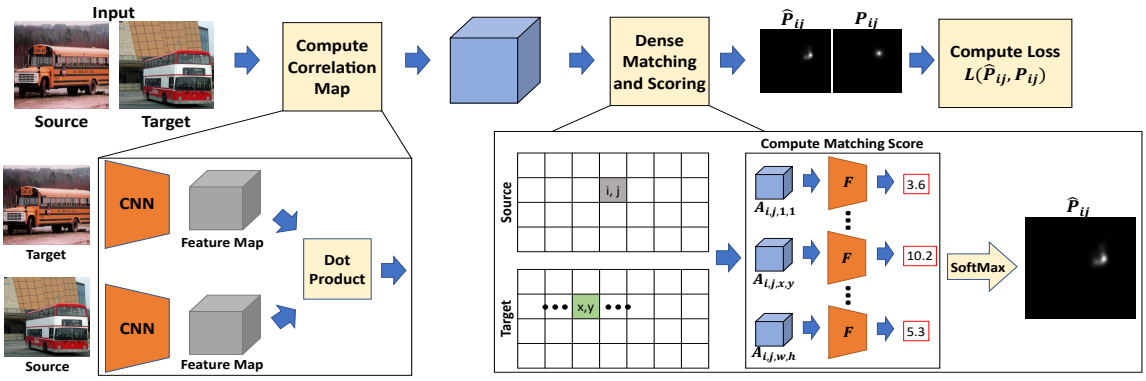

Figure 3: **Overview of training phase**. PMNC first computes the 4D correlation maps between the source and target images. Then for a pair of annotated keypoints in the source and target images, the learned function F is used to predict the probability of matching the source pixel to all the pixels in the target image. A 2D probability map is obtained by densely evaluating all target pixel locations by first computing a matching score using F and then applying a 2D softmax operator (see bottom right of figure). The parameters of F and the CNN are optimized by minimizing a cross entropy loss. The loss measures the deviation between the predicted and the ground-truth probability maps. We assume that the ground truth probabilities follow a 2D Gaussian distribution centered at the location of the annotated keypoint in the target image.

Layer	Number of Filters	Filter Size	Output Size
Convolution	16	$3 \times 3 \times 3 \times 3$	$5 \times 5 \times 5 \times 5$
ReLU	-	-	$5 \times 5 \times 5 \times 5$
Convolution	16	$3 \times 3 \times 3 \times 3$	$3 \times 3 \times 3 \times 3$
ReLU	-	-	$3 \times 3 \times 3 \times 3$
Convolution	1	$3 \times 3 \times 3 \times 3$	$1 \times 1 \times 1 \times 1$

Table 1: **Scoring function network architecture.** The function F takes a 4D patch $A_{ijkl} \in \mathbb{R}^{r \times r \times r \times r}$ and outputs a matching score. The table shows the details of the underlying network architecture which consists of two layers of 4D convolutions along with ReLU operations.

target images, respectively. We compute a 4D dense correlation map $C \in \mathbb{R}^{H \times W \times H \times W}$ that contains all the pixel-wise similarities $C_{ijkl} = \langle f^S_{ij}, f^T_{kl} \rangle$, where $\langle \cdot, \cdot \rangle$ is the inner product, and $f^S_{ij} \in \mathbb{R}^d$ and $f^T_{kl} \in \mathbb{R}^d$ are the feature vectors with unit norm for the pixels (i, j) and (k, l) from the source and target feature maps, respectively. Note that the descriptor of a pixel in our feature maps describes an image patch. We denote a local 4D patch of size $r \times r \times r \times r$ cropped from the correlation map C at location (i, j, k, l), as A_{ijkl}. The next step computes the correspondence map from the 4D correlation map in a greedy fashion. Specifically, we compute for every source feature in f^S and find the feature in f^T that has the best similarity score. The results of this step often contain incorrect matches but they serve as an initialization for the proposed iterative refinement method.

3.2. PatchMatch Inference

We perform PatchMatch based optimization to iteratively refine the correspondence map D. The PatchMatch algorithm [3] achieves this using three steps.

Initialization: With the correlation map that has a patch-wise correlation value between source image and target image, we construct the initial correspondence map D^0 that maps source pixels in f^S into target pixels in f^T based on the maximum correlation value:

$$D^0_{ij} = \arg\max_{kl} (C_{ijkl}) \in \mathbb{R}^2. \quad (1)$$

Note that the current dense correspondences are obtained from pure pixel-wise correlations without neighborhood consensus, and we let the correspondence map be refined through the propagation and update steps.

Propagation: We follow the Red-Black PatchMatch propagation method [20] for sampling candidate correspondences. For each source pixel (i, j), we obtain a set of candidate correspondences $\mathcal{S}^k_{ij}$, where

$$\mathcal{S}^k_{ij} = \{D^k_{ij}, D^k_{i,j+1}, D^k_{i,j-1}, \ldots\}, \quad (2)$$

from the correspondence of adjacent pixels in the correspondence map D^k at iteration k of PatchMatch. The adjacent pixels are chosen using the propagation kernel, which defines the shape of the local neighborhood candidate.

Update: Given the set of propagation candidates $\mathcal{S}^k_{ij}$, we evaluate each candidate correspondence with the scoring function F, and update the correspondence map by taking the correspondence value with the highest score. Mathematically, this operation can be described as follows

$$D^{k+1}_{ij} = \arg\max \left(\{F(A_{ij,(D^k_{xy})}) \mid D^k_{xy} \in \mathcal{S}^k_{ij}\} \right), \quad (3)$$

where $A_{ij,(D^k_{xy})}$ is the 4D patch cropped from C associating the pixels (i, j) and D^k_{xy}. The bottom-right of Figure

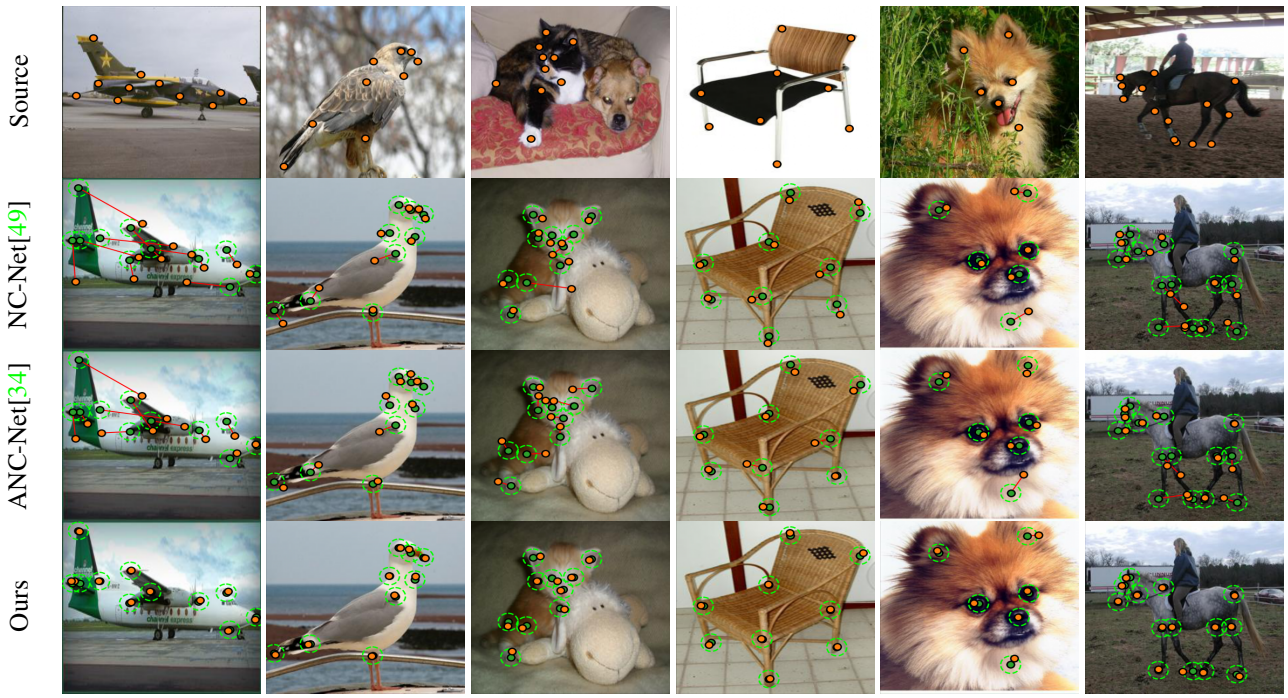

Figure 4: **Qualitative results on PF-PASCAL.** Top row shows source images and annotated keypoints. The rows below show NC-Net, ANC-Net and PMNC$_{best}$ results on the target image. We indicate the predicted locations and the GT locations using orange and green dots respectively, and we connect them by red line segments to visualize the errors. The circles correspond to the PCK threshold of $\alpha = 0.05$. On these six examples, PMNC$_{best}$ outperforms both NC-Net and ANC-Net.

2 illustrates this process. By using the trained patch scoring function F, we evaluate the matching scores for each propagation candidate in $\mathcal{S}_{ij}^k$ and update the current correspondence value D_{ij}^k with the correspondence that yields the highest score. We repeat the propagation and update process until the correspondence map converges.

3.3. Scoring Function

A crucial component of the refinement process is a function F that predicts the correctness of a correspondence. Unlike ANC-Net [34] and NC-Net [49] that use 4D convolutions over the whole correlation map C to refine and compute the correspondences, we instead propose a scoring function F that predicts the correctness of a correspondence $(i,j) \leftrightarrow (k,l)$ given a 4D patch A_{ijkl} extracted from the correlation map C. Different from the 4D convolutions over the whole correlation map C, which are computationally expensive and have large memory footprints, our proposed 4D patch-scoring function only applies 4D convolutions on a selected 4D patch extracted from the correlation map. By only using convolutions on a small 4D patch, PMNC reduces the computational complexity and memory footprint.

Our proposed scoring function $F : \mathbb{R}^{r\times r\times r\times r} \rightarrow \mathbb{R}$ maps a 4D patch of size r to a single value. This function F is based on a neural network composed of 4D convolutions and ReLU layers. Table 1 shows the architecture of F for a 4D patch of size $r = 7$. Section 3.4 describes how we learn the parameters of F and the CNN-based feature extractor.

3.4. Training Phase

Our inference pipeline is not differentiable. Therefore, we define a proxy problem to learn the parameters of F and the CNN-based feature extractor. Figure 3 illustrates the training procedure. Like ANC-Net [34], we use sparse labeled keypoint matches for supervision. During training, we first compute the correlation map C (Sec. 3.1). Then, for each source keypoint annotation (i,j), PMNC evaluates all the candidate target pixels (x,y) by predicting their matching scores from their 4D-patches A_{ijxy} using the scoring function F. PMNC then computes a 2D matching score map by arranging all the predicted scores following the pixel ordering of the target image. Next, by applying a softmax operation over the 2D matching score map, PMNC obtains a 2D probability map $\hat{P}_{ij}$ over all the target pixels.

PMNC computes a 2D probability map $\hat{P}_{ij}$ for each source keypoint annotation (i,j) following the architecture illustrated in Figure 3. This proxy architecture forces the feature extractor and patch scoring function F to produce meaningful results when we use PatchMatch as an alternative to the exhaustive matching algorithm.

To learn the parameters of the scoring function F and the CNN-based feature extractor, we compute the ground truth

Method	α				Time(s)	Memory(MB)
	0.1	0.05	0.03	0.01		
NCNet [49]	79.0	54.3	30.9	4.9	**0.29**	**406**
ANCNet [34]	85.9	58.1	31.9	5.1	0.33	1,310
GSF [27]	84.5	62.8	-	-	-	-
PMNC$_{fast}$	**86.8**	**74.5**	**58.0**	**14.7**	**0.09**	**273**
PMNC$_{best}$	**90.6**	**82.4**	**71.6**	**29.1**	0.96	2,610

Table 2: **Evaluation on PF-PASCAL [22].** PMNC$_{fast}$ and PMNC$_{best}$ are the two most accurate methods in terms of PCK metrics for different α values. PMNC$_{fast}$ is the fastest and most memory efficient, whereas PMNC$_{best}$ is the most accurate. The image resolution was 400×400 pixels. First and second places are indicated by bold blue and black respectively. The NC-Net [49] and ANC-Net [34] results were obtained using original code from the authors. The GSF [27] results are taken from their paper.

2D probability map P_{ij} of each source keypoint annotation (i, j) using a Gaussian distribution centered at the ground truth target correspondence (k, l) in f^T, *i.e.*,

$$P_{ij} = \mathcal{N}\left((k, l), \sigma\right) \in \mathbb{R}^{H \times W}, \tag{4}$$

where σ is the standard deviation denoting uncertainty. In our implementation we used $\sigma = 0.6$. Finally, because now we have ground truth P_{ij} and predicted probability maps $\hat{P}_{ij}$, we use the cross entropy loss $H(\cdot)$ to compare the deviation of two probability distributions, *i.e.*,

$$\mathcal{L} = \sum_{ij} H(P_{ij}, \hat{P}_{ij}). \tag{5}$$

We minimize $\mathcal{L}$ over the parameters of the scoring function F and the CNN feature extractor using backpropagation.

4. Experiments

We now report results on the PF-PASCAL [22] and SPAIR-71K [41] datasets. We measure timings on a PC with an Intel Core i9-7920X 2.9 Ghz CPU, 64 GB of DDR4 2666 Mhz RAM, and an NVidia RTX 2080TI GPU. We use the Percent Correct Keypoint (PCK) metric [60] to measure the precision of the sparse semantic correspondence with variable threshold α. The smaller the α, the stricter the measure. For PF-PASCAL, α is scaled by the larger image dimension, and for SPAIR-71K, α is scaled by the larger dimension of the object's bounding box.

PMNC$_{fast}$ and PMNC$_{best}$ denote two configurations of our method. PMNC$_{fast}$ uses spatial resolution equivalent to the fourth layer of ResNet-101, which resizes the original image by $\frac{1}{16}$ with $r = 5$. PMNC$_{best}$ uses the third layer of ResNet-101, which resizes the original image by $\frac{1}{8}$ with $r = 7$. For both configurations, we use 1546 feature map channels obtained by concatenating the third and fourth feature channels of ResNet-101. We use images at 400×400 pixel resolution, set $\sigma = 0.6$, and use two PatchMatch iterations for all experiments except in the ablation study.

# Iter.	σ	r	α			
			0.01	0.03	0.05	0.1
0	0.60	7×7	15.2	60.8	77.9	89.4
1	0.60	7×7	26.7	70.2	81.9	90.4
2	0.60	7×7	29.1	**71.0**	**82.4**	90.6
3	0.60	7×7	29.0	70.7	81.9	**91.2**
4	0.60	7×7	**29.4**	**71.0**	82.1	91.1
2	**0.15**	7×7	27.6	69.2	80.9	88.7
2	**0.30**	7×7	**31.4**	70.3	80.1	88.4
2	**0.45**	7×7	30.0	**71.1**	80.1	89.2
2	**0.60**	7×7	29.1	71.0	**82.4**	90.6
2	**0.75**	7×7	24.4	68.3	80.6	**91.0**
2	**0.90**	7×7	20.0	62.7	78.4	90.3
2	0.60	$\mathbf{3 \times 3}$	13.7	57.8	71.8	84.7
2	0.60	$\mathbf{5 \times 5}$	**29.7**	70.4	80.4	89.1
2	0.60	$\mathbf{7 \times 7}$	29.1	**71.0**	**82.4**	**90.6**

Table 3: **Effect of PMNC parameters**. We report PCK metrics for various α thresholds, PatchMatch iteration counts (# Iter.), σ values in the GT probability maps, and patch sizes r used in 4D correlations. The top section shows that most of the performance gain happens during the first two iterations. The middle section shows that smaller σ values result in improvement for $\alpha \leq 0.05$. The lower section shows that larger patches improve accuracy.

4.1. PF-Pascal

The PF-PASCAL [22] dataset contains 1,351 image pairs selected from the PASCAL VOC [17] dataset, where the ground truth keypoint matches were manually annotated. The dataset is split into around 700, 300, and 300 pairs for training, validation, and test, respectively. While NC-Net [49] uses both source-to-target and target-to-source pairs, we only use source-to-target pairs in our evaluation.

PMNC achieves the best PCK and timings. Table 2 shows the results on the PF-Pascal dataset. This Table presents the PCK metric for $\alpha \in [0.01, 0.1]$, the inference time in seconds, and memory requirements in megabytes. We observe that at any precision threshold α, PMNC$_{best}$ outperforms all the baselines but requires longer inference times and more memory. On the other hand, PMNC$_{fast}$ still achieves better PCK compared to NC-Net, ANC-Net and GSF, while requiring less time and memory. Since the code for GSF is not available, we are unable to report its timings and memory usage.

Cross entropy loss produces a significant benefit. To understand which part of the method provides the gain over the most similar baseline (ANC-Net), we analyzed the PCK values for our method based on zero PatchMatch iterations.

Architecture		aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	dog	horse	moto	person	plant	sheep	train	tv	all
Pretrained	CNNGeo [46]	21.3	15.1	34.6	12.8	31.2	26.3	24.0	30.6	11.6	24.3	20.4	12.2	19.7	15.6	14.3	9.6	28.5	28.8	18.1
	A2Net [53]	20.8	17.1	37.4	13.9	33.6	29.4	26.5	34.9	12.0	26.5	22.5	13.3	21.3	20.0	16.9	11.5	28.9	31.6	20.1
	WeakAlign [47]	23.4	17.0	41.6	14.6	37.6	28.1	26.6	32.6	12.6	27.9	23.0	13.6	21.3	22.2	17.9	10.9	31.5	34.8	21.1
	NC-Net [49]	24.0	16.0	45.0	13.7	35.7	25.9	19.0	50.4	14.3	32.6	27.4	19.2	21.7	20.3	20.4	13.6	33.6	40.4	26.4
	ANC-Net [34]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	28.7
	GSF [27]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	33.5
	PMNC$_{best}$	24.6	19.3	57.3	16.9	37.0	21.4	14.7	54.3	17.4	37.6	37.0	25.4	18.7	28.2	21.0	19.1	29.9	32.6	28.8
Trained / Fine-Tuned	CNNGeo [46]	23.4	16.7	40.2	14.3	36.4	27.7	26.0	32.7	12.7	27.4	22.8	13.7	20.9	21.0	17.5	10.2	30.8	34.1	20.6
	A2Net [53]	22.6	18.5	42.0	16.4	37.9	30.8	26.5	35.6	13.3	29.6	24.3	16.0	21.6	22.8	20.5	13.5	31.4	36.5	22.3
	WeakAlign [47]	22.2	17.6	41.9	15.1	38.1	27.4	27.2	31.8	12.8	26.8	22.6	14.2	20.0	22.2	17.9	10.4	32.2	35.1	20.9
	NC-Net [49]	17.9	12.2	32.1	11.7	29.0	19.9	16.1	39.2	9.9	23.9	18.8	15.7	17.4	15.9	14.8	9.6	24.2	31.1	20.1
	HPF [41]	25.2	18.9	52.1	15.7	38.0	22.8	19.1	52.9	17.9	33.0	32.8	20.6	24.4	27.9	21.1	15.9	31.5	35.6	28.2
	SFNet [33]	26.9	17.2	45.5	14.7	38.0	22.2	16.4	55.3	13.5	33.4	27.5	17.7	20.8	21.1	16.6	15.6	32.3	35.9	26.3
	SCOT [37]	34.9	20.7	63.8	21.1	**43.5**	27.3	21.3	63.1	**20.0**	42.9	42.5	31.1	29.8	35.0	27.7	24.4	48.4	40.8	35.6
	PMNC$_{fast}$	**49.5**	**31.6**	**67.1**	**29.8**	40.2	**52.0**	**39.0**	**65.2**	16.2	**67.9**	**50.4**	**40.3**	**41.4**	**42.9**	**30.2**	**31.1**	**60.4**	**53.8**	**44.7**
	PMNC$_{best}$	**54.1**	**35.9**	**74.9**	**36.5**	**42.1**	**48.8**	**40.0**	**72.6**	**21.1**	**67.6**	**58.1**	**50.5**	**40.1**	**54.1**	**43.3**	**35.7**	**74.5**	**59.9**	**50.4**

Table 4: **PCK evaluation ($\alpha = 0.1$) on SPAIR-71K per category using ResNet-101 [24] backbones**. The rows marked as "Pretrained" are for methods pretrained on PASCAL-VOC [17] and PF-PASCAL [22]; the "Trained / Fine Tuned" rows are for those trained on SPAIR-71K. The best/second-best methods are marked in bold blue/black respectively. Amongst "Pretrained" methods, PMNC$_{best}$ is the second best method. This shows that the PMNC methods generalize better than other baselines. For "Trained / Fine Tuned", PMNC$_{best}$ and PMNC$_{fast}$ have the highest and second highest accuracy. The results for CNNGeo [46], A2Net [53], WeakAlign [47] NC-Net [49] and HPF [41] are from the HPF [41] paper. The ANC-Net [34], GSF [27], SFNet [33] and SCOT [37] results are from the respective papers.

For α=0.01 and 0.1, the PCK values are 15.2 and 89.4 respectively (see first row of the Table 3). The corresponding values for ANC-Net are 5.1 and 85.9 (see Table 2). This is a fair comparison as both methods were trained on the same training data and annotations. We think the main reason for the superior performance of our method (10.3% and 10.4% gain in PCK) is due to our use of cross entropy loss whereas ANC-Net uses a distance based loss.

Figure 4 shows qualitative results obtained on the PF-Pascal dataset. The top row shows the source images, and the rows below show the estimated keypoint correspondences for NC-Net, ANC-Net, and PMNC. We visualize the pixel correspondence error between the estimated and true target locations with a red line and the tolerance $\alpha = 0.05$ region with a green circle. We see that our method tends to produce lower pixel correspondence errors since the red lines are not visible. Moreover, most PMNC predictions fall inside the green circles whereas the baselines tend to produce higher errors and more red lines are visible.

A few PatchMatch iterations suffice. The top five rows of Table 3 show the impact of running up to four PatchMatch iterations. For α = 0.01, we see 11.5%, 13.9%, and 14.2% improvement in PCK after 1, 2, and 4 iterations respectively. For α = 0.03 and α = 0.05, the best results, an improvement of 16.9% and 5.5% respectively, were obtained with two iterations. For α = 0.1, the best results were obtained with three iterations, but the improvement after two iterations was small. We attribute the convergence after only a few iterations to our use of greedy matching for PatchMatch initialization. It is known that better initialization can result in faster convergence in PatchMatch [40].

There is a positive correlation between σ and α. The six middle rows of Table 3 show that using larger σ values for generating the ground-truth probabilistic maps improves performance when using larger α values. This is because a larger σ relaxes the ground-truth probability map, which corresponds to a larger PCK tolerance.

Larger patches improve performance. The bottom three rows of Table 3 show that the larger the patch, the better the performance across many performance thresholds (*i.e.*, α values). This is because the scoring function is using more context. However, the larger the patch, the higher the computational cost because of the 4D convolutions.

4.2. SPair-71k

We also evaluate on SPAIR-71K [41] which has image pairs with occlusions (unlike PF-PASCAL [22]) and larger viewpoint and scale variations. It has 70,958 image pairs with 53,340 for *training*, 5,384 for *validation*, and 12,234 for *testing*. We use the provided subsets in our experiments.

PMNC$_{best}$ is the second best pre-trained model. Table 4 shows results on all 18 object categories. The top section reports performance when models are trained on PF-PASCAL and tested on SPAIR-71K. PMNC$_{best}$ performs best on the most reported categories and has the second highest PCK. This shows that our models trained on PF-PASCAL generalize reasonably well to SPAIR-71K.

PMNC$_{best}$ is the best fine-tuned model. The bot-

	Methods	View-Point			Scale			Truncation				Occlusion				All
		Easy	Med	Hard	Easy	Med	Hard	None	Src	Tgt	Both	None	Src	Tgt	Both	
Pretrained	CNNGeo [46]	25.2	10.7	5.9	22.3	16.1	8.5	21.1	12.7	15.6	13.9	20.0	14.9	14.3	12.4	18.1
	A2Net [53]	27.5	12.4	6.9	24.1	18.5	10.3	22.9	15.2	17.6	15.7	22.3	16.5	15.2	14.5	20.1
	WeakAlign [47]	29.4	12.2	6.9	25.4	19.4	10.3	24.1	16.0	18.5	15.7	23.4	16.7	16.7	14.8	21.1
	NC-Net [49]	34.0	18.6	12.8	31.7	23.8	14.2	29.1	22.9	23.4	21.0	29.0	21.1	21.8	19.6	26.4
	ANC-Net [34]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	28.7
	GSF [27]	40.6	22.3	17.8	39.5	30.1	18.7	37.0	28.7	27.1	27.7	36.4	27.8	27.5	23.7	33.5
	PMNC$_{best}$	34.6	22.7	18.2	32.6	27.0	19.5	31.0	28.0	24.2	23.5	32.0	24.1	21.8	17.4	28.8
Trained / Fine-Tuned	CNNGeo[46]	28.8	12.0	6.4	24.8	18.7	10.6	23.7	15.5	17.9	15.3	22.9	16.1	16.4	14.4	20.6
	A2Net [53]	30.9	13.3	7.4	26.1	21.1	12.4	25.0	17.4	20.5	17.6	24.6	18.6	17.2	16.4	22.3
	WeakAlign [47]	29.3	11.9	7.0	25.1	19.1	11.0	24.0	15.8	18.4	15.6	23.3	16.1	16.4	15.7	20.9
	NC-Net [49]	26.1	13.5	10.1	24.7	17.5	9.9	22.2	17.1	17.5	16.8	22.0	16.3	16.3	15.2	20.1
	HPF[41]	35.6	20.3	15.5	33.0	26.1	15.8	31.0	24.6	24.0	23.7	30.8	23.5	22.8	21.8	28.2
	SCOT [37]	42.7	28.0	23.9	41.1	33.7	21.4	39.0	32.4	30.0	30.0	39.0	30.3	28.1	26.0	35.6
	PMNC$_{fast}$	**46.5**	**42.9**	**41.1**	**48.6**	**43.7**	**34.2**	**48.6**	**42.1**	**38.4**	**35.4**	**48.2**	**39.1**	**37.4**	**34.2**	**44.7**
	PMNC$_{best}$	**53.3**	**47.4**	**45.9**	**53.7**	**49.6**	**41.5**	**54.3**	**46.8**	**45.0**	**41.9**	**54.2**	**43.9**	**43.0**	**38.4**	**50.4**

Table 5: **PCK evaluation ($\alpha = 0.1$) on SPAIR-71K per nuisance (view point, scale, truncation, and occlusion)**. The difficulty levels for view point and scale are labeled easy, medium, and hard. For truncation and occlusion, the difficultly labels are none, source, target, and both. Rows are labeled in the same way as in Table 4. The "Pretrained" block shows that PMNC$_{best}$ has the second highest accuracy. However, the bottom block shows that when all the methods were trained, PMNC$_{best}$ and PMNC$_{fast}$ have the highest and second highest PCK accuracy per nuisance and difficulty level. The PCK metrics for CNNGeo [46], A2Net [53], WeakAlign [47] NC-Net [49] and HPF [41] are from the HPF [41] paper. The ANC-Net [34], GSF [27] and SCOT [37] PCK metrics are from the respective papers.

Figure 5: **SPAIR-71K results.** Transferred keypoints shown on the target image obtained using PMNC$_{best}$. The color coding scheme is the same as Figure 4. The provided 2D bounding boxes showing the object locations is not used by our method. Our method computes accurate semantic correspondences despite having view-point, scale, and illumination changes.

tom section of Table 4 shows results for methods trained on SPAIR-71K. We can see that PMNC$_{best}$ has the highest PCK for 14/18 categories. PMNC$_{fast}$ ranks second in 16/18 categories. Overall, we outperform existing works by a significant margin for all categories except chair and bottle. We achieve the best overall PCK of **50.4** with PMNC$_{best}$. The next best method (excluding our own PMNC$_{fast}$ is SCOT [37] at **35.6**. Table 5 shows how difficulty level affects PCK for each method. PMNC$_{best}$ is the best and PMNC$_{fast}$ is the second best method for all the categories. Also our method's performance margin over baselines improves going from *Easy* to *Hard*.

5. Conclusion

We propose PMNC, a new approach to estimate dense semantic correspondence between two images. We jointly train the CNN-based feature extractor and neighborhood consensus-based 4D scoring function using an end-to-end differentiable model and sparse keypoint supervision. During inference, we refine the dense correspondence map using PatchMatch. PMNC clearly outperforms ANC-Net [34] and GSF [27] on PF-PASCAL and SPAIR-71K and sets the new state-of-the-art on both datasets.

References

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] Aayush Bansal, Yaser Sheikh, and Deva Ramanan. Shapes and context: In-the-wild image synthesis and manipulation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[3] Connelly Barnes, Eli Shechtman, Dan B Goldman, and Adam Finkelstein. The generalized patchmatch correspondence algorithm. In *European Conference on Computer Vision*, pages 29–43. Springer, 2010.

[4] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[5] J. Bian, W. Lin, Y. Matsushita, S. Yeung, T. Nguyen, and M. Cheng. Gms: Grid-based motion statistics for fast, ultra-robust feature correspondence. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4181–4190, 2017.

[7] Michael Bleyer, Christoph Rhemann, and Carsten Rother. Patchmatch stereo-stereo matching with slanted support windows. In *Bmvc*, volume 11, pages 1–11, 2011.

[8] Dongdong Chen, Lu Yuan, Jing Liao, Nenghai Yu, and Gang Hua. Stereoscopic neural style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6654–6663, 2018.

[9] Minsu Cho, Karteek Alahari, and Jean Ponce. Learning graphs to match. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 25–32, 2013.

[10] Minsu Cho, Jungmin Lee, and Kyoung Mu Lee. Reweighted random walks for graph matching. In *European conference on Computer vision*, pages 492–505. Springer, 2010.

[11] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

[12] Soheil Darabi, Eli Shechtman, Connelly Barnes, Dan B Goldman, and Pradeep Sen. Image melding: Combining inconsistent images using patch-based synthesis. *ACM Transactions on graphics (TOG)*, 31(4):1–10, 2012.

[13] Joseph DeGol, Timothy Bretl, and Derek Hoiem. Improved structure from motion using fiducial marker matching. In *Proc. of the European Conference on Computer Vision (ECCV)*, 2018.

[14] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superpoint: Self-supervised interest point detection and description. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[15] Shivam Duggal, Shenlong Wang, Wei-Chiu Ma, Rui Hu, and Raquel Urtasun. Deeppruner: Learning efficient stereo matching via differentiable patchmatch. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8092–8101, 2019.

[17] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, 2010.

[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.

[19] Victor Fragoso, Pradeep Sen, Sergio Rodriguez, and Matthew Turk. Evsac: accelerating hypotheses generation by modeling matching scores with extreme value theory. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2472–2479, 2013.

[20] Silvano Galliani, Katrin Lasinger, and Konrad Schindler. Massively parallel multiview stereopsis by surface normal diffusion. In *The IEEE International Conference on Computer Vision (ICCV)*, June 2015.

[21] Yoav HaCohen, Eli Shechtman, Dan B Goldman, and Dani Lischinski. Non-rigid dense correspondence with applications for image enhancement. *ACM transactions on graphics (TOG)*, 30(4):1–10, 2011.

[22] Bumsub Ham, Minsu Cho, Cordelia Schmid, and Jean Ponce. Proposal flow: Semantic correspondence from object proposals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3475–3484, 2016.

[23] Kai Han, Rafael S Rezende, Bumsub Ham, Kwan-Yee K Wong, Minsu Cho, Cordelia Schmid, and Jean Ponce. Scnet: Learning semantic correspondence. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1831–1840, 2017.

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[25] Yinlin Hu, Rui Song, and Yunsong Li. Efficient coarse-to-fine patchmatch for large displacement optical flow. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5704–5712, 2016.

[26] Junhwa Hur, Hwasup Lim, Changsoo Park, and Sang Chul Ahn. Generalized deformable spatial pyramid: Geometry-preserving dense correspondence estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1392–1400, 2015.

[27] Sangryul Jeon, Dongbo Min, Seungryong Kim, Jihwan Choe, and Kwanghoon Sohn. Guided semantic flow. In *European Conference on Computer Vision*, pages 631–648. Springer, 2020.

[28] Jaechul Kim, Ce Liu, Fei Sha, and Kristen Grauman. Deformable spatial pyramid matching for fast dense correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2307–2314, 2013.

[29] Seungryong Kim, Dongbo Min, Bumsub Ham, Sangryul Jeon, Stephen Lin, and Kwanghoon Sohn. Fcss: Fully convolutional self-similarity for dense semantic correspondence. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6560–6569, 2017.

[30] Seungryong Kim, Dongbo Min, Stephen Lin, and Kwanghoon Sohn. Dctm: Discrete-continuous transformation matching for semantic flow. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[31] Simon Korman and Shai Avidan. Coherency sensitive hashing. *IEEE transactions on pattern analysis and machine intelligence*, 38(6):1099–1112, 2015.

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017.

[33] Junghyup Lee, Dohyung Kim, Jean Ponce, and Bumsub Ham. Sfnet: Learning object-aware semantic correspondence. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2278–2287, 2019.

[34] Shuda Li, Kai Han, Theo W Costain, Henry Howard-Jenkins, and Victor Prisacariu. Correspondence networks with adaptive neighbourhood consensus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10196–10205, 2020.

[35] Wen-Yan Daniel Lin, Ming-Ming Cheng, Jiangbo Lu, Hongsheng Yang, Minh N Do, and Philip Torr. Bilateral functions for global motion modeling. In *European Conference on Computer Vision*, pages 341–356. Springer, 2014.

[36] Ce Liu, Jenny Yuen, and Antonio Torralba. Sift flow: Dense correspondence across scenes and its applications. *IEEE transactions on pattern analysis and machine intelligence*, 33(5):978–994, 2010.

[37] Yanbin Liu, Linchao Zhu, Makoto Yamada, and Yi Yang. Semantic correspondence as an optimal transport problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4463–4472, 2020.

[38] Jonathan Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence. In *In NIPS*, 2014.

[39] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 2, pages 1150–1157 vol.2, 1999.

[40] Jiangbo Lu, Hongsheng Yang, Dongbo Min, and Minh N Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1854–1861, 2013.

[41] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Hyperpixel flow: Semantic correspondence with multi-layer neural features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3395–3404, 2019.

[42] Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. Lf-net: learning local features from images. In *Advances in neural information processing systems*, pages 6234–6244, 2018.

[43] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim, Paul Merrell, et al. Detailed real-time urban 3d reconstruction from video. *International Journal of Computer Vision*, 78(2-3):143–167, 2008.

[44] Jerome Revaud, Philippe Weinzaepfel, César Roberto de Souza, and Martin Humenberger. R2D2: repeatable and reliable detector and descriptor. In *NeurIPS*, 2019.

[45] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. Deepmatching: Hierarchical deformable dense matching. *International Journal of Computer Vision*, 120(3):300–323, 2016.

[46] Ignacio Rocco, Relja Arandjelovic, and Josef Sivic. Convolutional neural network architecture for geometric matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6148–6157, 2017.

[47] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6917–6925, 2018.

[48] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. Efficient neighbourhood consensus networks via submanifold sparse convolutions. In *European conference on computer vision*, 2020.

[49] Ignacio Rocco, Mircea Cimpoi, Relja Arandjelović, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Neighbourhood consensus networks. In *Advances in Neural Information Processing Systems*, pages 1651–1662, 2018.

[50] Cordelia Schmid and Roger Mohr. Local grayvalue invariants for image retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 19(5):530–535, 1997.

[51] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2016.

[52] Pradeep Sen, Nima Khademi Kalantari, Maziar Yaesoubi, Soheil Darabi, Dan B Goldman, and Eli Shechtman. Robust patch-based hdr reconstruction of dynamic scenes. *ACM Trans. Graph.*, 31(6):203–1, 2012.

[53] Paul Hongsuck Seo, Jongmin Lee, Deunsol Jung, Bohyung Han, and Minsu Cho. Attentive semantic alignment with offset-aware correlation kernels. In *European Conference on Computer Vision*, pages 367–383. Springer, 2018.

[54] Tatsunori Taniai, Sudipta N Sinha, and Yoichi Sato. Joint recovery of dense correspondence and cosegmentation in two images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4246–4255, 2016.

[55] Moria Tau and Tal Hassner. Dense correspondences across scenes and scales. *IEEE transactions on pattern analysis and machine intelligence*, 38(5):875–888, 2015.

[56] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. *arXiv preprint arXiv:2003.12039*, 2020.

[57] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015.

[58] Nikolai Ufer and Bjorn Ommer. Deep semantic feature matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6914–6923, 2017.

[59] Li Xu, Jiaya Jia, and Yasuyuki Matsushita. Motion detail preserving optical flow estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1744–1757, 2011.

[60] Yi Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2878–2890, dec 2013.

[61] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artif. Intell.*, 1995.