Convolutional neural network architecture for geometric matching

Ignacio Rocco, Relja Arandjelovic, Josef Sivic

Convolutional neural network architecture for geometric matching

Ignacio Rocco, Relja Arandjelović, and Josef Sivic

Abstract—We address the problem of determining correspondences between two images in agreement with a geometric model such as an affine, homography or thin-plate spline transformation, and estimating its parameters. The contributions of this work are three-fold. First, we propose a convolutional neural network architecture for geometric matching. The architecture is based on three main components that mimic the standard steps of feature extraction, matching and simultaneous inlier detection and model parameter estimation, while being trainable end-to-end. Second, we demonstrate that the network parameters can be trained from synthetically generated imagery without the need for manual annotation and that our matching layer significantly increases generalization capabilities to never seen before images. Finally, we show that the same model can perform both instance-level and category-level matching giving state-of-the-art results on the challenging PF, TSS and Caltech-101 datasets.

Index Terms—Convolutional neural network, geometric matching, image alignment, category-level matching.

1 Introduction

Estimating correspondences between images is one of the fundamental problems in computer vision [1], [2] with applications ranging from large-scale 3-D reconstruction [3] to image manipulation [4] and semantic segmentation [5]. Traditionally, correspondences consistent with a geometric model such as epipolar geometry or planar affine transformation, are computed by detecting and matching local features (such as SIFT [6] or HOG [7], [8]), followed by pruning incorrect matches using local geometric constraints [9], [10] and robust estimation of a global geometric transformation using algorithms such as RANSAC [11] or Hough transform [6], [12], [13]. This approach works well in many cases but fails in situations that exhibit (i) large changes of depicted appearance due to e.g. intra-class variation [8], or (ii) large changes of scene layout or non-rigid deformations that require complex geometric models with many parameters which are hard to estimate in a manner robust to outliers.

In this work we build on the traditional approach and develop a convolutional neural network (CNN) architecture that mimics the standard matching process. First, we replace the standard local features with powerful trainable convolutional neural network features [14], [15], which allows us to handle large changes of appearance between the matched images. Second, we develop trainable matching and transformation estimation layers that can cope with noisy and incorrect matches in a robust way, mimicking the good practices in feature matching such as the second nearest neighbor test [6], neighborhood consensus [9], [10] and Hough transform-like estimation [6], [12], [13].

The outcome is a convolutional neural network architecture trainable for the end task of geometric matching, which can handle large appearance changes, and is therefore suitable for both instance-level and category-level matching problems.

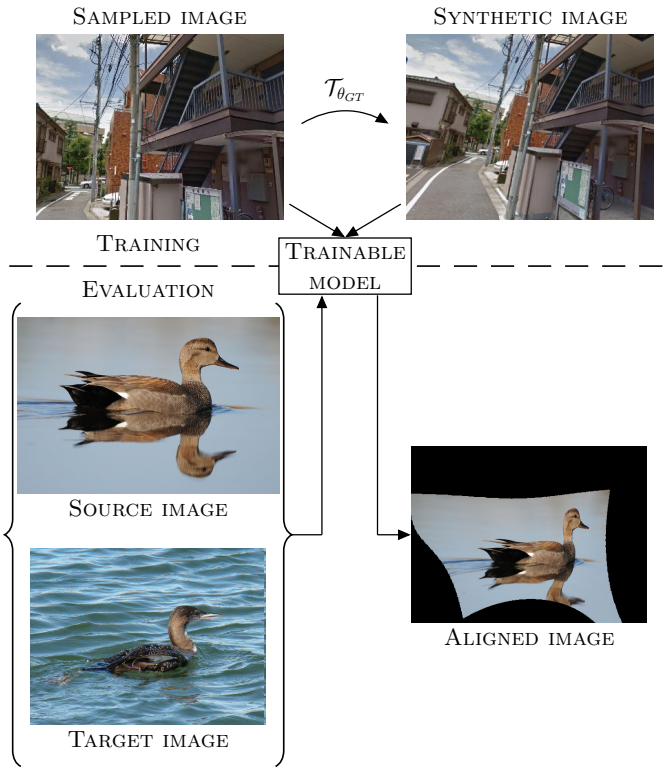


Fig. 1. **Top:** The proposed model can be trained from synthetic image pairs, avoiding the need for manual annotation. **Bottom:** At evaluation time, the trained geometry estimation network automatically aligns two images with substantial appearance differences. It is able to estimate large deformable transformations robustly in the presence of clutter.

The contributions of this work are three-fold. First, we propose a convolutional neural network architecture for geometric matching, which mimics the standard steps of feature extraction, matching and simultaneous inlier detection and model parameter

- *I. Rocco and J. Sivic are with Inria and the Département d'informatique de l'ENS, École normale supérieure, CNRS, PSL Research University, 75005 Paris, France.*
  E-mail: {ignacio.rocco, josef.sivic}@inria.fr
- *J. Sivic is also with the Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague.*
- *R. Arandjelović is with DeepMind.*
  E-mail: relja@google.com

estimation, while being trainable end-to-end. Second, we demonstrate that the network parameters can be trained from synthetically generated imagery without the need for manual annotation and that our matching layer significantly increases generalization capabilities to never seen before images. Finally, we show that the same model can give state-of-the-art results on several challenging datasets for category-level image alignment.

All training and evaluation code, as well as our trained networks, is available online [16].

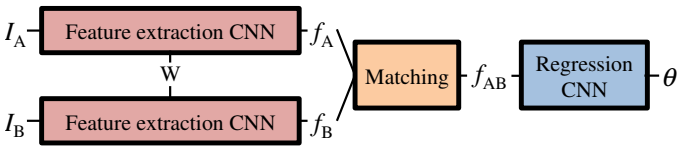


Fig. 2. **Diagram of the proposed architecture.** Images I_A and I_B are passed through feature extraction networks which have tied parameters W, followed by a matching network which matches the descriptors. The output of the matching network is passed through a regression network which outputs the parameters of the geometric transformation.

2 Related work

The classical approach for finding correspondences involves identifying interest points and computing local descriptors around these points [6], [9], [17], [18], [19], [20], [21]. While this approach performs relatively well for instance-level matching, the feature detectors and descriptors lack the generalization ability for category-level matching.

Recently, convolutional neural networks have been used to learn powerful feature descriptors which are more robust to appearance changes than the classical descriptors [22], [23], [24], [25], [26], [27]. However, these works are focused on learning descriptors [22], [23], [26], [27], or a similarity measure between descriptors [24], [25], [28], and do not target the problem of finding the transformation relating the two input images. In this work, we go a step further from CNN descriptors, and seek to also learn to estimate the geometric transformation.

Related are also network architectures for estimating inter-frame motion in video [29], [30], [31] or instance-level homography estimation [32], however their goal is very different from ours, targeting high-precision correspondence with very limited appearance variation and background clutter. Closer to us is the network architecture of [33] which, however, tackles a different problem of fine-grained category-level matching (different species of birds) with limited background clutter and small translations and scale changes, as their objects are largely centered in the image. In addition, their architecture is based on a different matching layer, which we show not to perform as well as the matching layer used in our work.

Some works, such as [8], [20], [34], [35], [36], [37], have addressed the hard problem of category-level matching, but rely on traditional non-trainable optimization for matching [20], [34], [35], [36], [37], or guide the matching using object proposals [8]. On the contrary, our approach is fully trainable in an end-to-end manner and does not require any optimization procedure at evaluation time, or guidance by object proposals.

Others [38], [39], [40] have addressed the problems of instance and category-level correspondence by performing joint image alignment. However, these methods differ from ours as they: (i) require class labels; (ii) don't use CNN features; (iii) jointly align a large set of images, while we align image pairs; and (iv) don't use a trainable CNN architecture for alignment as we do.

3 Architecture for geometric matching

In this section, we introduce a new convolutional neural network architecture for estimating parameters of a geometric transformation between two input images. The architecture is designed to mimic the classical computer vision pipeline (e.g. [41]), while using differentiable modules so that it is trainable end-to-end for the geometry estimation task. The classical approach consists of the following stages: (i) local descriptors (e.g. SIFT) are extracted from both input images, (ii) the descriptors are matched across images to form a set of tentative correspondences, which are then used to (iii) robustly estimate the parameters of the geometric model using RANSAC or Hough voting.

Our architecture, illustrated in Fig. 2, mimics this process by: (i) passing input images I_A and I_B through a siamese architecture consisting of convolutional layers, thus extracting feature maps f_A and f_B which are analogous to dense local descriptors, (ii) matching the feature maps ("descriptors") across images into a tentative correspondence map f_{AB}, followed by a (iii) regression network which directly outputs the parameters of the geometric model, θ, in a robust manner. The inputs to the network are the two images, and the outputs are the parameters of the chosen geometric model, e.g. a 6-D vector for an affine transformation.

In the following, we describe each of the three stages in detail.

3.1 Feature extraction

The first stage of the pipeline is feature extraction, for which we use a standard CNN architecture. A CNN without fully connected layers takes an input image and produces a feature map $f \in \mathbb{R}^{h \times w \times d}$, which can be interpreted as a $h \times w$ dense spatial grid of d-dimensional local descriptors. A similar interpretation has been used previously in instance retrieval [42], [43], [44], [45] demonstrating high discriminative power of CNN-based descriptors. Thus, for feature extraction we use the VGG-16 network [15], cropped at the `pool4` layer (before the ReLU unit), followed by per-feature L2-normalization. We use a pre-trained model, originally trained on ImageNet [46] for the task of image classification. As shown in Fig. 2, the feature extraction network is duplicated and arranged in a siamese configuration such that the two input images are passed through two identical networks which share parameters.

3.2 Matching network

The image features produced by the feature extraction networks should be combined into a single tensor as input to the regressor network to estimate the geometric transformation. We first describe the classical approach for generating tentative correspondences, and then present our matching layer which mimics this process.

3.2.1 Tentative matches in classical geometry estimation

Classical methods start by computing similarities between all pairs of descriptors across the two images. From this point on, the original descriptors are discarded as all the necessary information for geometry estimation is contained in the pairwise descriptor similarities and their spatial locations. Secondly, the pairs are

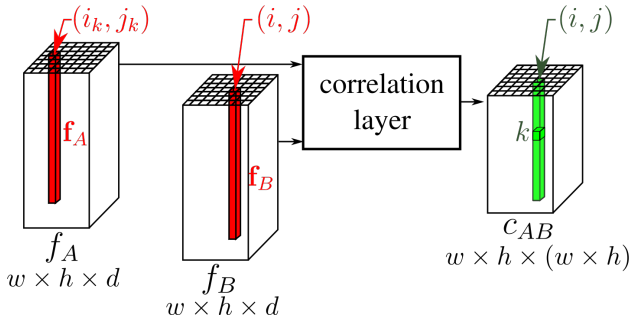


Fig. 3. **Correlation map computation with CNN features.** The correlation map c_{AB} contains all pairwise similarities between individual features $\mathbf{f}_A \in f_A$ and $\mathbf{f}_B \in f_B$. At a particular spatial location (i, j) the correlation map output c_{AB} contains all the similarities between $\mathbf{f}_B(i, j)$ and all $\mathbf{f}_A \in f_A$.

pruned by either thresholding the similarity values, or, more commonly, only keeping the matches which involve the nearest (most similar) neighbors. Furthermore, the second nearest neighbor test [6] prunes the matches further by requiring that the match strength is significantly stronger than the second best match involving the same descriptor, which is very effective at discarding ambiguous matches.

3.2.2 Matching layer

Our matching layer applies a similar procedure. Analogously to the classical approach, only descriptor similarities and their spatial locations should be considered for geometry estimation, and not the original descriptors themselves.

To achieve this, we propose to use a *correlation layer* followed by normalization. Firstly, all pairs of similarities between descriptors are computed in the correlation layer. Secondly, similarity scores are processed and normalized such that ambiguous matches are strongly down-weighted.

In more detail, given L2-normalized dense feature maps $f_A, f_B \in \mathbb{R}^{h \times w \times d}$, the correlation map $c_{AB} \in \mathbb{R}^{h \times w \times (h \times w)}$ outputted by the correlation layer contains at each position the scalar product of a pair of individual descriptors $\mathbf{f}_A \in f_A$ and $\mathbf{f}_B \in f_B$, as detailed in Eq. (1).

$$c_{AB}(i, j, k) = \mathbf{f}_B(i, j)^T \mathbf{f}_A(i_k, j_k) \tag{1}$$

where (i, j) and (i_k, j_k) indicate the individual feature positions in the $h \times w$ dense feature maps, and $k = h(j_k - 1) + i_k$ is an auxiliary indexing variable for (i_k, j_k).

A diagram of the correlation layer is presented in Fig. 3. Note that at a particular position (i, j), the correlation map c_{AB} contains the similarities between $\mathbf{f}_B$ at that position and *all* the features of f_A.

As is done in the classical methods for tentative correspondence estimation, it is important to postprocess the pairwise similarity scores to remove ambiguous matches. To this end, we apply a channel-wise normalization of the correlation map at each spatial location to produce the final tentative correspondence map f_{AB}. The normalization is performed by ReLU, to zero out negative correlations, followed by L2-normalization, which has two desirable effects. First, let us consider the case when descriptor $\mathbf{f}_B$ correlates well with only a single feature in f_A. In this case, the normalization will amplify the score of the match, akin to the nearest neighbor matching in classical geometry estimation. Second, in the case of the descriptor $\mathbf{f}_B$ matching multiple features in f_A due to the existence of clutter or repetitive patterns, matching scores will be down-weighted similarly to the second nearest neighbor test [6]. However, note that both the correlation and the normalization operations are differentiable with respect to the input descriptors, which facilitates backpropagation thus enabling end-to-end learning.

3.2.3 Discussion

The first step of our matching layer, namely the correlation layer, is somewhat similar to layers used in DeepMatching [29] and FlowNet [30]. However, DeepMatching [29] only uses deep RGB patches and no part of their architecture is trainable. FlowNet [30] uses a spatially constrained correlation layer such that similarities are are only computed in a restricted spatial neighborhood thus limiting the range of geometric transformations that can be captured. This is acceptable for their task of learning to estimate optical flow, but is inappropriate for larger transformations that we consider in this work. Furthermore, neither of these methods performs score normalization, which we find to be crucial in dealing with cluttered scenes.

Previous works have used other matching layers to combine descriptors across images, namely simple concatenation of descriptors along the channel dimension [32] or subtraction [33]. However, these approaches suffer from two problems. First, as following layers are typically convolutional, these methods also struggle to handle large transformations as they are unable to detect long-range matches. Second, when concatenating or subtracting descriptors, instead of computing pairwise descriptor similarities as is commonly done in classical geometry estimation and mimicked by the correlation layer, image content information is directly outputted. To further illustrate why this can be problematic, consider two pairs of images that are related with the same geometric transformation – the concatenation and subtraction strategies will produce different outputs for the two cases, making it hard for the regressor to deduce the geometric transformation. In contrast, the correlation layer output is likely to produce similar correlation maps for the two cases, regardless of the image content, thus simplifying the problem for the regressor. In line with this intuition, in Sec. 7.3 we show that the concatenation and subtraction methods indeed have difficulties generalizing beyond the training set, while our correlation layer achieves generalization yielding superior results.

3.3 Regression network

The normalized correlation map is passed through a regression network which directly estimates parameters of the geometric transformation relating the two input images. In classical geometry estimation, this step consists of robustly estimating the transformation from the list of tentative correspondences. Local geometric constraints are often used to further prune the list of tentative matches [9], [10] by only retaining matches which are consistent with other matches in their spatial neighborhood. Final geometry estimation is done by RANSAC [11] or Hough voting [6], [12], [13].

We again mimic the classical approach using a neural network, where we stack two blocks of convolutional layers, followed by batch normalization [47] and the ReLU non-linearity, and add a final fully connected layer which regresses to the parameters of the transformation, as shown in Fig. 4. The intuition behind this architecture is that the estimation is performed in a bottom-up manner

Fig. 4. **Architecture of the regression network.** It is composed of two convolutional layers without padding and stride equal to 1, followed by batch normalization and ReLU, and a final fully connected layer which regresses to the P transformation parameters.

somewhat like Hough voting, where early convolutional layers vote for candidate transformations, and these are then processed by the later layers to aggregate the votes. The first convolutional layers can also enforce local neighborhood consensus [9], [10] by learning filters which only fire if nearby descriptors in image A are matched to nearby descriptors in image B, and we show qualitative evidence in Sec. 7.5 that this indeed does happen.

3.3.1 Discussion

A potential alternative to a convolutional regression network is to use fully connected layers. However, as the input correlation map size is quadratic in the number of image features, such a network would be hard to train due to a large number of parameters that would need to be learned, and it would not be scalable due to occupying too much memory and being too slow to use. It should be noted that even though the layers in our architecture are convolutional, the regressor can learn to estimate large transformations. This is because one spatial location in the correlation map contains similarity scores between the corresponding feature in image B and *all* the features in image A (c.f. equation (1)), and not just the local neighborhood as in [30].

4 Geometric Transformations

Three different parametric geometric transformations were employed in this work: affine, homography and thin-plate spline. The details of their parametrizations are presented next. As images are warped using the reverse mapping, the transformations map coordinates from the target image B to the source image A.

4.1 Affine transformation

An affine transformation is a 6 degree-of-freedom linear transformation capable of modeling translation, rotation, non-isotropic scaling and shear. It can be parametrized by a 6 dimensional vector θ_{AFF}:

$$\theta_{\text{AFF}} = [a_{11}, a_{12}, a_{21}, a_{22}, t_x, t_y], \tag{2}$$

such that points $P_B = [x_B, y_B]^T$ are mapped to points $P_A = [x_A, y_A]^T$ according to:

$$P_A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} P_B + \begin{bmatrix} t_x \\ t_y \end{bmatrix}. \tag{3}$$

4.2 Homography transformation

A homography transformation deforms a given quadrilateral $\mathcal{Q}_B = \{Q_{B1}, \ldots, Q_{B4}\}$ into any other given quadrilateral $\mathcal{Q}_A = \{Q_{A1}, \ldots, Q_{A4}\}$, while keeping collinearity. It has 8 degrees-of-freedom and is more flexible than the affine transformation, as it can handle perspective since parallel lines need not remain parallel. Homography is the model relating 2-D images

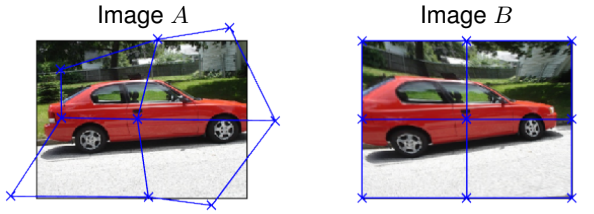


Fig. 5. **Thin-plate spline control points.** Illustration of the 3×3 TPS grid of control points used in the thin-plate spline transformation model.

(pinhole projections) of a 3-D plane. We adopt the 4-point homography parametrization from [32], which consists of defining the quadrilateral $\mathcal{Q}_B$ of the target image to be the outer edge of the image, and using the coordinates of the quadrilateral $\mathcal{Q}_A$ of the source image as the 8-dimensional vector θ_{HOM}:

$$\theta_{\text{HOM}} = [x_{Q_{A1}}, \ldots, x_{Q_{A4}}, y_{Q_{A1}}, \ldots, y_{Q_{A4}}]. \tag{4}$$

This parametrization can be converted to the 3×3 homography matrix H [2], which is used to perform the actual transformation:

$$\begin{aligned} x_A &= \frac{h_{11}\, x_B + h_{12}\, y_B + h_{13}}{h_{31}\, x_B + h_{32}\, y_B + h_{33}}, \\ y_A &= \frac{h_{21}\, x_B + h_{22}\, y_B + h_{23}}{h_{31}\, x_B + h_{32}\, y_B + h_{33}}, \end{aligned} \tag{5}$$

where, h_{ij} are elements of the homography matrix H.

4.3 Thin-plate spline transformation

The thin-plate spline (TPS) transformation [48] is a parametric model which performs smooth 2-D interpolation given a set of k corresponding control points $\mathcal{C}_A = \{C_{A1}, \ldots, C_{Ak}\}$ and $\mathcal{C}_B = \{C_{B1}, \ldots, C_{Bk}\}$ between two images. In this work we use $k = 9$ and arrange the control points $\mathcal{C}_B$ in a 3×3 uniform grid on the target image, as illustrated in Fig. 5. Because control points $\mathcal{C}_B$ are fixed for all image pairs, the TPS transformation is parametrized only by the control points $\mathcal{C}_A$ in the source image:

$$\theta_{\text{TPS}} = [x_{C_{A1}}, \ldots, x_{C_{A9}}, y_{C_{A1}}, \ldots, y_{C_{A9}}]. \tag{6}$$

Then, the thin-plate spline transformation maps points $P_B = [x_B, y_B]^T$ to points $P_A = [x_A, y_A]^T$ according to:

$$\begin{aligned} x_A &= a_x + b_x\, x_B + c_x\, y_B + \sum_{i=1}^{k} w_{xi} U(\|P_B - C_{Bi}\|), \\ y_A &= a_y + b_y\, x_B + c_y\, y_B + \sum_{i=1}^{k} w_{yi} U(\|P_B - C_{Bi}\|). \end{aligned} \tag{7}$$

Here, $U(z) = z^2 \log z^2$ and the parameters a, b, c and w are computed from θ_{TPS} by:

$$\begin{aligned} [w_{x1}, \ldots, w_{xk}, a_x, b_x, c_x]^T &= L^{-1}[x_{C_{A1}}, \ldots, x_{C_{Ak}}, 0, 0, 0]^T \\ [w_{y1}, \ldots, w_{yk}, a_y, b_y, c_y]^T &= L^{-1}[y_{C_{A1}}, \ldots, y_{C_{Ak}}, 0, 0, 0]^T, \end{aligned} \tag{8}$$

where L^{-1} is a constant matrix which needs to be computed only once, as it depends only on the fixed control points $\mathcal{C}_B$. Please refer to [48] for further details.

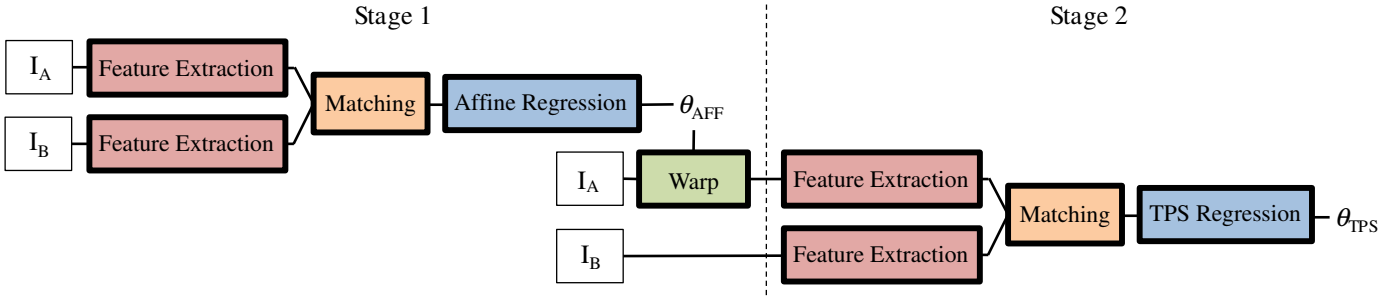


Fig. 6. **Estimating progressively more complex geometric transformations.** Images A and B are passed through a network which estimates an affine transformation with parameters θ_{AFF} (see Fig. 2). Image A is then warped using this transformation to roughly align it with B, and passed along with B through a second network which estimates a thin-plate spline (TPS) transformation that refines the alignment.

4.4 Hierarchy of transformations

A commonly used approach when estimating image to image transformations is to start by estimating a simple transformation and then progressively increase the model complexity, refining the estimates along the way [18], [20], [41]. The motivation behind this method is that estimating a very complex transformation could be hard and computationally inefficient in the presence of clutter, so a robust and fast rough estimate of a simpler transformation can be used as a starting point, also regularizing the subsequent estimation of the more complex transformation.

We follow the same good practice and start by estimating an affine transformation (or alternatively a homography) which performs a rough alignment. The estimated affine transformation is then used to align image A to image B using an image resampling layer [49]. The aligned images are then passed through a second geometry estimation network which estimates the parameters of a thin-plate spline transformation. The final estimate of the geometric transformation is then obtained by composing the two transformations. The process is illustrated in Fig. 6, and detailed in Algorithm 2.

4.5 Iterative refinement

When input images are related by a large transformation, it is difficult to obtain many good matches, so a single pass through the geometry estimation network might produce a poor alignment. In such cases, performing several iterations of the estimation can be beneficial, as illustrated in Fig. 7, since it allows the number of matches to progressively grow. This approach has proven to be particularly useful for instance-level alignment, as detailed in section 6.4.

5 Training

In order to train the parameters of our geometric matching CNN, it is necessary to design the appropriate loss function, and to use suitable training data. We address these two important points next, and also provide details about the implementation.

5.1 Loss function

We assume a fully supervised setting, where the training data consists of pairs of images and the desired outputs in the form of the parameters θ_{GT} of the ground-truth geometric transformation. The loss function $\mathcal{L}$ is designed to compare the estimated transformation θ with the ground-truth transformation θ_{GT} and, more importantly, compute the gradient of the loss function with respect

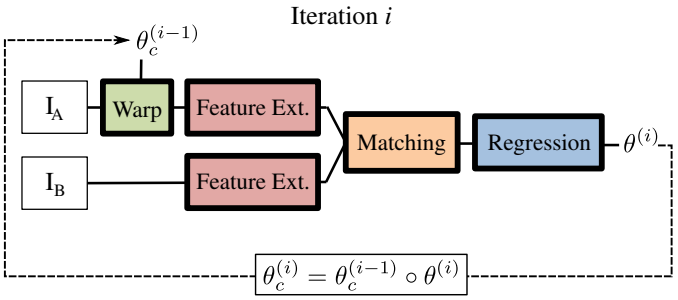


Fig. 7. **Iterative transformation refinement.** In iteration *i*, image A is warped using the cumulative transformation estimate $\theta_c^{(i-1)}$ obtained from the previous iteration ($\theta_c^{(0)}$ is initialized to identity). A fine alignment, $\theta^{(i)}$, between image B and the warped image A is estimated and chained onto $\theta_c^{(i-1)}$ to form the refined cumulative transformation estimate $\theta_c^{(i)}$.

to the estimates $\frac{\partial \mathcal{L}}{\partial \theta}$. This gradient is then used in a standard manner to learn the network parameters which minimize the loss function by using backpropagation and Stochastic Gradient Descent.

It is desired for the loss to be general and not specific to a particular type of geometric model, so that it can be used for estimating affine, homography, thin-plate spline or any other geometric transformation. Furthermore, the loss should be independent of the parametrization of the transformation and thus should not directly operate on the parameter values themselves. We address all these design constraints by measuring loss on an imaginary grid of points $\mathcal{G} = \{G_i\} = \{(x_i, y_i)\}_{i=1...N}$ which is being deformed by the transformation. Namely, we construct a grid of points in image space, transform it using the neural network estimated and ground-truth transformations $\mathcal{T}_\theta$ and $\mathcal{T}_{\theta_{GT}}$ with parameters θ and θ_{GT}, respectively, and measure the discrepancy between the two transformed grids by summing the squared distances between the corresponding grid points:

$$\mathcal{L}(\theta, \theta_{GT}) = \frac{1}{N} \sum_{i=1}^{N} \|G'_i - G''_i\|^2 \tag{9}$$

where $G'_i = \mathcal{T}_\theta(G_i)$ and $G''_i = \mathcal{T}_{\theta_{GT}}(G_i)$ are the transformed grid points according to the estimated and ground-truth transformations respectively. The grid points are uniformly distributed in the image using normalized coordinates, i.e. $x_i, y_i \in [-1, 1]$. Note that we construct the coordinate system such that the center of the image is at $(0, 0)$ and that the width and height of the

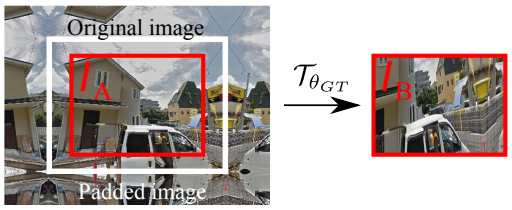


Fig. 8. **Synthetic image generation.** Symmetric padding is added to the original image to enlarge the sampling region, its central crop is used as image A, and image B is created by performing a randomly sampled transformation $\mathcal{T}_{\theta_{GT}}$.

image are equal to 2, i.e. the bottom left and top right corners have coordinates $(-1, -1)$ and $(1, 1)$, respectively.

The gradient of the loss function with respect to the transformation parameters, needed to perform backpropagation in order to learn network weights, can be computed easily if the location of the transformed grid points $G'_i = \mathcal{T}_\theta(G_i)$ is differentiable with respect to θ. This is commonly the case, for example, when $\mathcal{T}$ is an affine transformation, $\mathcal{T}_\theta(G_i)$ is linear in parameters θ and therefore the loss can be differentiated in a straightforward manner.

5.2 Training data

Our training procedure requires fully supervised training data consisting of image pairs and a known geometric relation. Training CNNs usually requires a lot of data, and no public datasets exist that contain many image pairs annotated with their geometric transformation. Therefore, we opt for training from synthetically generated data, which gives us the flexibility to gather as many training examples as needed, for any 2-D geometric transformation of interest. Given that this training data is obtained for free, the approach can be classified as unsupervised (or self-supervised), even though the loss function requires the ground-truth transformation parameters.

5.2.1 *Synthetic pair generation*

We generate each training pair (I_A, I_B), by sampling I_A from a public image dataset, and generating I_B by applying a random transformation $\mathcal{T}_{\theta_{GT}}$ to I_A. More precisely, I_A is created from the central crop of the original image, while I_B is created by transforming the original image with added symmetrical padding in order to avoid border artifacts; the procedure is shown in Fig. 8.

5.2.2 *Synthetic training datasets*

The images used for the synthetic pair generation are sampled from the Tokyo Time Machine dataset [45] which contains Google StreetView images of Tokyo. We select 20k images for training and 20k for validation.

During training, a batch is first selected from the training split of the dataset, and then a random transformation for each image in the batch is sampled independently from reasonable ranges. These ranges depend on the geometric model chosen for training the network.

In the case of the affine transformation, we chose a rotation angle $\theta \sim \mathcal{U}(-\pi/12, \pi/12)$, a shear angle $\phi \sim \mathcal{U}(-\pi/6, \pi/6)$, anisotropic scaling factors $\lambda_1, \lambda_2 \sim \mathcal{U}(0.75, 1.25)$, and translations $t_x, t_y \sim \mathcal{U}(-0.25, 0.25)$. These parameters are defined on the SVD decomposition of the affine transformation (see [2] sec. 2.4.3), and must then be composed to obtain the $[a_{ij}]$ matrix described in section 4.1:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = R(\theta)R(-\phi)\text{diag}(\lambda_1, \lambda_2)R(\phi). \quad (10)$$

In the case of the homography and thin-plate spline transformations, the target points $\mathcal{Q}_A$ and $\mathcal{C}_A$ are obtained by perturbing the fixed $\mathcal{Q}_B$ and $\mathcal{C}_B$ with random translations $\delta_x, \delta_y \sim \mathcal{U}(-0.4, 0.4)$:

$$\begin{aligned} Q_{Ai} &= Q_{Bi} + (\delta_x, \delta_y), \\ C_{Ai} &= C_{Bi} + (\delta_x, \delta_y). \end{aligned} \quad (11)$$

In all cases, the uniform distribution was used in order not to impose a strong prior on the transformation parameters. The ranges were chosen to roughly cover the observed transformations in the PF-WILLOW dataset.

Algorithm 1: Training procedure using synthetic pairs

input : Image database DB
CNN model M_W
output: Trained CNN model M_W

for *training epochs* **do**
  for I *in DB* **do**
    `// Generate synthetic training pair`
    Sample random transformation θ_{GT}
    $I_A =$ central crop of I;
    $I_B = \mathcal{T}_{\theta_{\text{GT}}}(I)$;
    `// Estimate transformation` θ
    $\theta = M_W(I_A, I_B)$;
    `// Compute loss and update model`
    $L = \mathcal{L}(\theta, \theta_{GT})$;
    $W = \text{update}(W, \frac{\partial \mathcal{L}}{\partial W})$;
  end
end

5.3 Implementation details

We use the PyTorch library [50] and train the networks using the Adam [51] optimizer with learning rate 10^{-3}, and a batch size of 16. There is no need for jittering as instead of data augmentation we can simply generate more synthetic training data. Input images are resized to 240×240 producing 15×15 feature maps that are passed into the matching layer. Single-stage models specific for each particular geometric transformation (affine, homography or thin-plate spline) are trained, with transformations sampled randomly at training time according to the previously described procedure. Each network is trained until convergence which typically occurs after 20 epochs (25000 iterations), and takes between 4 and 8 hours on a single GPU, depending on the complexity of the geometric model. The training algorithm is detailed in Algorithm 1.

At evaluation time, the single-stage models featuring different geometric transformations can be used in conjunction as illustrated in Fig. 6. Trained networks can also be executed iteratively as described in section 4.5 and illustrated in Fig. 7. The evaluation algorithm is detailed in Algorithm 2.

Algorithm 2: **Transformation estimation using two-stage network**

input : Source and target images (I_A, I_B)
Stage 1 CNN model M_1
Stage 2 CNN model M_2
output: Aligned image I''_A

```
// First stage
```
$\theta_1 = M_1(I_A, I_B)$;
$I'_A = \mathcal{T}_{\theta_1}(I_A)$;
```
// Second stage
```
$\theta_2 = M_2(I'_A, I_B)$;
$I''_A = \mathcal{T}_{\theta_2}(I'_A)$;

6 Experimental results

In this section we compare our method to baselines and the state-of-the-art for both category-level and instance-level alignment problems.

In the case of category-level alignment, both qualitative and quantitative evaluation is performed on three different datasets previously used for this task: the PF dataset [8], the TSS dataset [52] and the Caltech-101 dataset [53].

In the case of instance-level alignment, qualitative and quantitative evaluation is performed on the Graffiti benchmark [19]. In addition, qualitative alignment results are presented for the Tokyo Time Machine dataset [45].

A different single-stage model for each of the affine, homography and thin-plate spline transformations was trained independently. Both single-stage (Fig. 2) and two-stage (Fig. 6) alignment strategies were investigated. Furthermore, the iterative refinement procedure described in section 7 was used for the Graffiti benchmark.

6.1 PF dataset

This dataset contains image pairs depicting different instances of the same classes, such as ducks and cars, but with large intra-class variations, e.g. the cars are often of different make, or the ducks can be of different subspecies. Furthermore, the images contain significant background clutter, as can be seen in Fig. 9. It contains a total of 2251 image pairs from two subgroups: PF-WILLOW (900 pairs, introduced in [54]) and PF-PASCAL (1251 pairs, introduced in [8]). Images from each pair were manually selected to ensure that objects have similar poses.

6.1.1 Evaluation metric

The quality of the obtained alignment is assessed by exploiting the keypoint annotation provided with the PF dataset. The task is to predict the locations of predefined keypoints from image A in image B. We do so by estimating a geometric transformation $\mathcal{T}$ that warps image A into image B, and applying the same transformation to the keypoint locations $\mathbf{P}_A = \{P^i_A\}_{i=1,\dots,n}$ in I_A, to obtain the estimated keypoint locations $\{\mathcal{T}(P^i_A)\}_{i=1,\dots,n}$ in I_B. The alignment quality is then computed using the standard evaluation metric for this benchmark, the average probability of correct keypoint (PCK) [55], being the proportion of keypoints that are correctly matched. A keypoint is considered to be matched correctly if the distance between its predicted location $\mathcal{T}(P^i_A)$ and its ground-truth position P^i_B is below a predefined threshold L. Therefore, the PCK is computed as follows:

TABLE 1
Matching accuracy on the PF dataset (PF-PASCAL/PF-WILLOW) measured in terms of the PCK (α = 0.1). All the numbers apart from ours and RANSAC are taken from [8].

Methods	PF-PASCAL	PF-WILLOW
DeepFlow [56]	0.21	0.20
GMK [35]	0.27	0.27
SIFT Flow [34]	0.33	0.38
DSP [36]	0.30	0.37
Proposal Flow (SS+NAM) [8]	0.36	0.52
Proposal Flow (SS+PHM) [8]	0.42	0.55
Proposal Flow (SS+LOM) [8]	0.45	0.56
RANSAC with our features (affine)	0.44	0.46
Ours (affine)	0.46	0.48
Ours (homography)	0.48	0.49
Ours (TPS)	0.51	0.54
Ours (affine + TPS)	0.51	**0.60**
Ours (homography + TPS)	**0.53**	**0.60**
Ours (2×TPS)	0.52	0.57

$$\text{PCK} = \frac{|\{P^i_A \in \mathbf{P}_A, d(\mathcal{T}(P^i_A), P^i_B) < L\}|}{n}, \qquad (12)$$

where the distance threshold is $L = \alpha \cdot \max(h, w)$, $\alpha = 0.1$ and (h, w) are the height and width of the object bounding box, respectively.

6.1.2 Results

We compare our method against SIFT Flow [34], Graph-matching kernels (GMK) [35], Deformable spatial pyramid matching (DSP) [36], DeepFlow [56], and all three variants of Proposal Flow (NAM, PHM, LOM) [8]. As shown in Tab. 1, our method outperforms all others and sets the new state-of-the-art on this data. The best competing methods are based on Proposal Flow and make use of object proposals, which enables them to guide the matching towards regions of images that contain objects. Their performance varies significantly with the choice of the object proposal method, illustrating the importance of this guided matching. On the contrary, our method does not use any guiding, but it still manages to outperform even the best Proposal Flow and object proposal combination.

Furthermore, we also compare to affine transformations estimated with RANSAC using the same descriptors as our method (VGG-16 `pool4`). The parameters of this baseline have been tuned extensively to obtain the best result by adjusting the thresholds for the second nearest neighbor test and by pruning proposal transformations which are outside of the range of likely transformations. Our affine estimator outperforms the RANSAC baseline on this task by a 2% margin.

Fig. 9 illustrates the effectiveness of our method in category-level matching, where challenging pairs of images from the Proposal Flow dataset [8], containing large intra-class variations, are aligned correctly. The method is able to robustly, in the presence of clutter, estimate large translations, rotations, scale changes, as well as non-rigid transformations and some perspective changes.

6.2 TSS dataset

The TSS dataset introduced in [52] contains 400 image pairs of three subgroups: FG3DCar contains 195 image pairs of cars, JODS contains 81 image pairs of airplanes, horses and cars and PASCAL

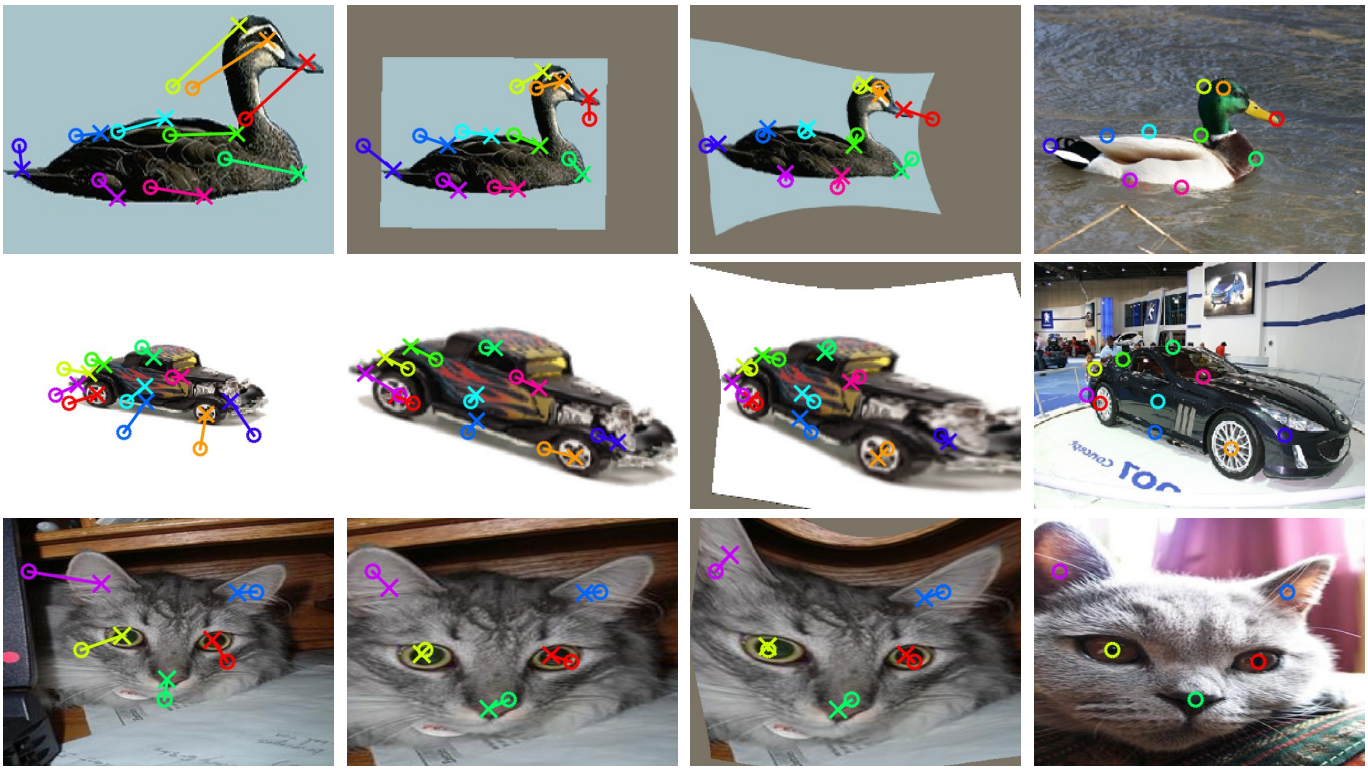


Fig. 9. **Qualitative results on the PF dataset.** Each row shows one test example from the Proposal Flow dataset. Ground truth matching keypoints, only used for alignment evaluation, are depicted as crosses and circles for images A and B, respectively. Keypoints of same color are supposed to match each other after image A is aligned to image B. To illustrate the matching error, we also overlay keypoints of B onto different alignments of A so that lines that connect matching keypoints indicate the keypoint position error vector. Our method manages to roughly align the images with an affine transformation (column 2), and then perform finer alignment using thin-plate spline (TPS, column 3). The top two examples are from the PF-WILLOW dataset while the bottom one is from PF-PASCAL.

contains 124 image pairs of bicycles, motorbikes, buses, cars and trains. For all 400 image pairs, approximate ground-truth optical flow data is provided.

6.2.1 Evaluation metric

The evaluation metric used for the TSS dataset is also the PCK, presented in Eq. (12), previously presented for the PF dataset. However, the ground-truth optical flow data available in the TSS dataset allows to compute the PCK using a dense set of keypoints $\mathbf{P}_A = \{P_A^i \in \mathbf{M}_A\}_{i=1,\dots,n}$, composed by the foreground pixels inside the segmentation mask $\mathbf{M}_A$ of the object in I_A. Therefore, this allows for the alignment to be densely evaluated over the object of interest, in contrast to the PCK computation for the PF dataset, where the alignment is only evaluated in a handful of manually annotated keypoints.

Regarding the distance threshold $L = \alpha \cdot \max(h, w)$ used for the PCK computation, the criterion used in [8], [52] is adopted, where the reported values are computed with $\alpha = 0.05$ and with (h, w) being the dimensions of the target image.

6.2.2 Results

The quantitative results for the TSS dataset are presented in Tab. 2, in terms of the mean PCK over the set of image pairs. For each of the 400 pairs, both the forward (from I_A to I_B) and backward (from I_B to I_A) alignments are computed an evaluated, resulting in a total of 800 evaluation pairs.

The middle columns of Tab. 2 present mean PCK over the three subsets of the TSS dataset: FG3DCar, JODS and PASCAL; and the right-most column presents the mean PCK over the whole TSS dataset. It can be observed that our single-stage models improve the overall average score by up to 3%, while the two-stage models achieve the best results on all the different subsets, and improve by up to 7% over the previously published results.

TABLE 2
Matching accuracy on the TSS dataset in terms of PCK ($\alpha = 0.05$). The three intermediate columns show the results for each subset of the TSS dataset: FG3DCar, JODS and PASCAL. The last column shows the PCK result over the whole dataset.

Methods	FG3DCar	JODS	PASCAL	All
DSP [36]	0.49	0.47	0.38	0.45
SIFT Flow [34]	0.63	0.51	0.36	0.52
Taniai *et al.* [52]	0.83	0.60	0.48	0.67
Proposal Flow (SS+LOM) [8]	0.79	0.65	0.53	0.68
Ours (affine)	0.81	0.65	0.51	0.68
Ours (homography)	0.83	0.66	0.52	0.70
Ours (TPS)	0.84	**0.72**	0.51	0.71
Ours (affine + TPS)	**0.89**	**0.72**	0.54	**0.75**
Ours (homography + TPS)	0.88	**0.72**	**0.55**	**0.75**
Ours (2×TPS)	0.86	0.70	0.52	0.72

In Fig. 10 we present qualitative results on the TSS dataset. In order to assess the visual quality of the obtained results, we also present the ground-truth aligned and segmented images provided with the dataset in the right-most column.

As it can be observed, the proposed method can produce good

alignments results, which are close to the ground-truth alignment.

6.3 Caltech-101 dataset

Following the same procedure as in [8], [36], the alignment quality is also evaluated on the Caltech-101 dataset [53]. For each of the 101 categories, 15 image pairs were chosen randomly, resulting in 1515 evaluation pairs. These pairs are the same as in [8].

6.3.1 Evaluation metrics

As no keypoint annotations are provided for the Caltech-101 dataset, PCK cannot be used to assess the matching accuracy. Since segmentations masks are provided, we follow [8], [36] and evaluate the quality of segmentation mask alignment using the following metrics: label transfer accuracy (LT-ACC), intersection-over-union (IoU), and localization error (LOC-ERR).

Let (I_A, I_B) be a pair of images with ground-truth segmentation masks $(\mathrm{M}_A, \mathrm{M}_B)$, and $\mathcal{T}$ be the estimated transformation from I_A to I_B. Then, the transferred annotated mask from the source image, $\mathrm{M}'_A = \mathcal{T}(\mathrm{M}_A)$, is compared with the ground-truth mask in the target image M_B to assess the alignment quality.

The label transfer accuracy metric (LT-ACC), measures the number of correctly transferred foreground and background pixels in the following way:

$$\text{LT-ACC} = \frac{|\{P \in I_B, \mathrm{M}'_A(P) = \mathrm{M}_B(P)\}|}{|P \in I_B|} \tag{13}$$

Therefore, the numerator of the LT-ACC adds up the number of background pixels which are correctly mapped to background and the foreground pixels which are correctly mapped to foreground.

The intersection-over-union (IoU), or Jaccard index, as the LT-ACC also compares the mask alignment quality. However, contrary to LT-ACC, it only considers the correctly aligned foreground pixels, ignoring the background. It is computed in the following way:

$$\text{IoU} = \frac{|\mathrm{M}'_A \cap \mathrm{M}_B|}{|\mathrm{M}'_A \cup \mathrm{M}_B|}. \tag{14}$$

Finally, the localization error (LOC-ERR) metric measures the spatial error of each transferred pixel [36], assuming that the images are related by a translation and anisotropic scaling transformation which aligns the bounding boxes of the source and target images.

To this end, two normalized coordinates systems are defined relative to the source and target image bounding boxes, such that their origins are set on the top-left corners of the object bounding boxes, and the coordinates are normalized by the widths and heights of the bounding boxes.

Let $O_A = (x_{O_A}, y_{O_A})$ and $O_B = (x_{O_B}, y_{O_B})$ be the top-left corners of the objects bounding boxes on I_A and I_B respectively, and (h_A, w_A), (h_B, w_B) the bounding box dimensions.

Then LOC-ERR metric measures the disagreement between the coordinates of the original point $P_A = (x_A, y_A)$ relative to O_A, and its transformed coordinates $P'_A = \mathcal{T}(P_A) = (x'_A, y'_A)$ relative to O_B, in the following way:

$$\text{LOC-ERR} = \frac{1}{|V|} \sum_{(P_A, P'_A) \in V} |\hat{x}_A - \hat{x}'_A| + |\hat{y}_A - \hat{y}'_A|, \tag{15}$$

where $(\hat{x}_A, \hat{y}_A)$ and $(\hat{x}'_A, \hat{y}'_A)$ are the normalized coordinates of P_A and $P'_A = \mathcal{T}(P_A)$:

$$\begin{aligned} (\hat{x}_A, \hat{y}_A) &= \left(\frac{x_A - x_{O_A}}{w_A}, \frac{y_A - y_{O_A}}{h_A} \right) \\ (\hat{x}'_A, \hat{y}'_A) &= \left(\frac{x'_A - x_{O_B}}{w_B}, \frac{y'_A - y_{O_B}}{h_B} \right) \end{aligned} \tag{16}$$

and V is the set of all pairs of points (P_A, P'_A) in which the transformed points $P'_A = \mathcal{T}(P_A)$ fall inside the bounds of image I_B.

TABLE 3
Evaluation on the Caltech-101 dataset. Matching quality is measured in terms of LT-ACC and IoU. The best two Proposal Flow methods (RP, LOM and SS, LOM) are included here. All numbers apart from ours are taken from [54].

Methods	LT-ACC	IoU	LOC-ERR
DeepFlow [56]	0.74	0.40	0.34
GMK [35]	0.77	0.42	0.34
SIFT Flow [34]	0.75	0.48	0.32
DSP [36]	0.77	0.47	0.35
Proposal Flow (RP, LOM) [54]	0.78	0.50	0.26
Proposal Flow (SS, LOM) [54]	0.78	0.50	0.25
Ours (affine)	0.78	0.51	**0.24**
Ours (homography)	0.80	0.52	**0.24**
Ours (TPS)	0.80	0.53	**0.24**
Ours (affine + TPS)	0.79	**0.55**	0.26
Ours (homography + TPS)	**0.81**	**0.55**	0.25
Ours (2×TPS)	0.80	0.54	0.26

6.3.2 Results

The quantitative results on the Caltech-101 dataset are presented in Tab. 3. As it can be observed, our approach outperforms the state-of-the-art by a significant margin, obtaining, for example, an IoU of 0.55 compared to the previous best result of 0.50.

In addition, it can be observed that the LOC-ERR metric values do not follow the trend of the other two metrics. This is because the LOC-ERR metric makes the invalid assumption that the images are related with a translation and anisotropic scaling transformation.

The benefit of the two-stage approaches is clear from the more realistic IoU metric, where adding a second stage achieves an IoU of 0.55 compared to 0.53 of the single-stage best performing model.

In Fig. 11, we present a qualitative comparison of the results obtained by our method and other previous methods on images from this dataset. For each example, the second row presents the transferred segmentation mask $\mathcal{T}(\mathrm{M}_A)$ of I_A overlaid with I_B, for each of the methods. As it can be visually assessed, the proposed approach achieves a superior alignment than most of the previous methods.

6.4 Graffiti benchmark

This section presents the results of the proposed method on the challenging Graffiti instance-level matching benchmark [19]. This benchmark contains 6 images of the same planar scene with increasingly varying viewpoint, with up to 70° azimuthal rotation from the reference image. Ground-truth homography transformations from the reference image 1 to images 2-6 are available with the dataset. We employed the same homography estimation CNN used for the category-level alignment datasets, trained from synthetic StreetView image pairs. To overcome the

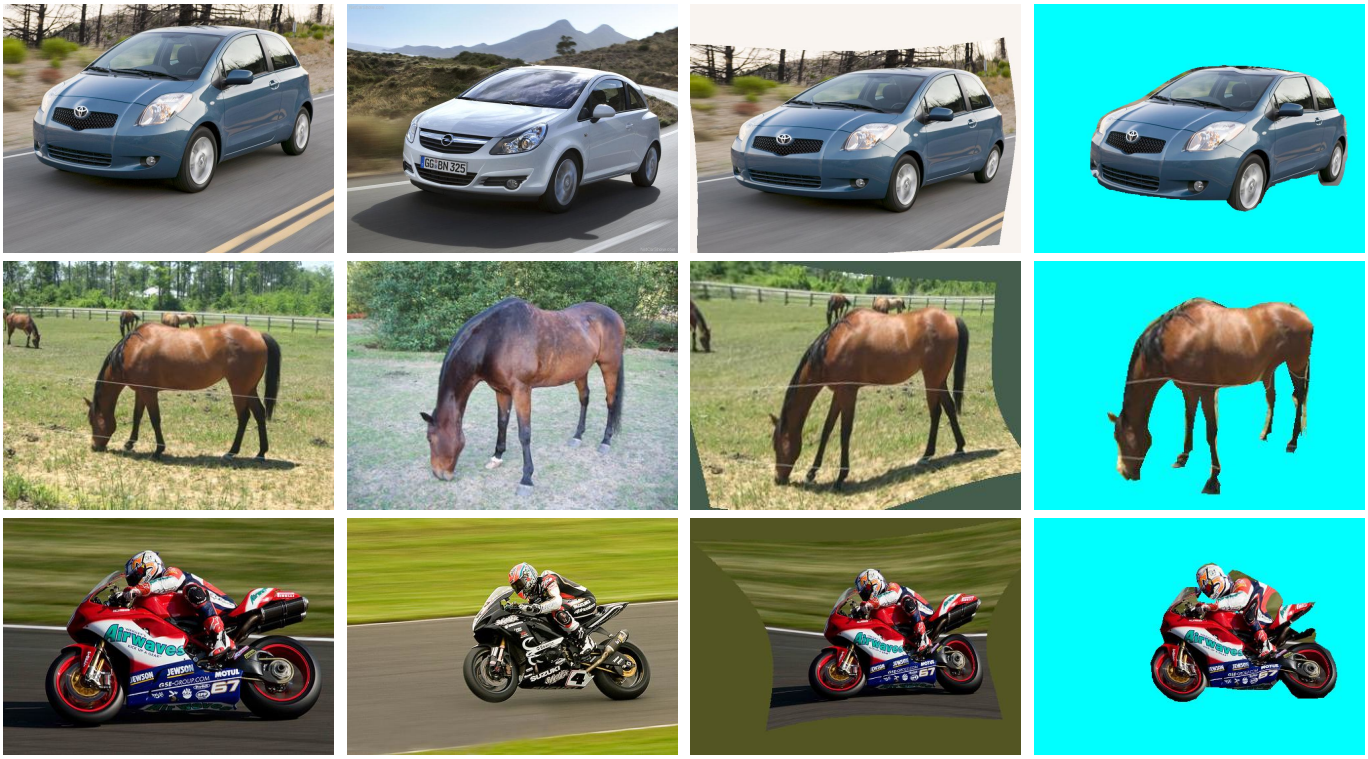


Image A | Image B | Aligned A (affine+TPS) | Aligned A (ground-truth)

Fig. 10. **Qualitative results on the TSS dataset.** Each row shows one test example from the TSS dataset. The last column shows the ground-truth alignment used for evaluation. Example 1 is from TSS-FG3DCar, examples 2-3 are from TSS-JODS, and 4-7 from TSS-PASCAL.

(a) Image pairs | (b) DeepFlow | (c) GMK | (d) SIFT Flow | (e) DSP | (f) Proposal Flow | (g) Ours (aff.+TPS)

Fig. 11. **Qualitative results on the Caltech-101 dataset.** Each block of two rows corresponds to one example, where column (a) shows the original images – image A in the first row and image B in the second row. The remaining columns of the first row show image A aligned to image B using various methods. The second row shows image B overlaid with the segmentation map transferred from image A.

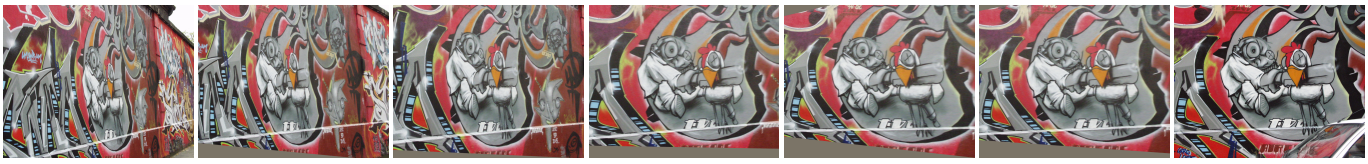


Fig. 12. **Qualitative results on the Graffiti benchmark, pair** $(1, 5)$**.** The first and last columns show the source and target images. The intermediate columns show the progress of the alignment at different iterations.

large viewpoint variation in this dataset, we perform iterative refinement, as described in section 4.5, with a total of 5 iterations. The same CNN model is used for all iterations.

6.4.1 Evaluation metric

In order to measure the quality of the estimated homography transformation, the average endpoint error is used:

$$\text{AEE}(H, H_{GT}) = \frac{1}{N} \sum_{i=1}^{N} \|G'_i - G''_i\| \tag{17}$$

where $G'_i = \mathcal{T}_H(G_i)$ and $G''_i = \mathcal{T}_{H_{GT}}(G_i)$ are the transformed sampling points $G = [1, 2, \dots, h] \times [1, 2, \dots, w]$ when applying the estimated and ground-truth homographies, respectively. Note that this is similar to the proposed loss (9), but without squaring the distances.

6.4.2 Results

Quantitative results are presented on Tab. 4. The proposed method is compared against state-of-the-art methods for this dataset, such as SIFT features (DoG+SIFT) [6], [18], SIFT features with affine-covariant detectors (DoG-affine+SIFT) [19] and ASIFT [57]. Features are matched and filtered using the second nearest-neighbor test [18] and the homography transformation is estimated with the locally optimized RANSAC algorithm [58] with the following parameter settings: distance threshold $\theta = 6$px, estimation confidence $\eta_0 = 0.999$, and the final refinement step using all inliers. The rest of the parameters are set to their default values. The RANSAC algorithm is executed 5 times and the mean error values are reported. In all cases, the standard deviations were below 0.3 pixels.

As it can be observed from Tab. 4, although our method does not produce the best results, it still achieves reasonable alignments for pairs $(1, 2)$ up to $(1, 5)$, and fails for pair $(1, 6)$. The lower performance of our method when compared to local interest points methods can be explained by the lower resolution of the input image, which is resized from 800×640px to 240×240px, and also by the low resolution of the extracted CNN features, which is of 15×15. In addition, due to the max-pooling operations, the exact positions of the image features cannot be recovered from the CNN features.

However, it is interesting to point out that while DoG-SIFT fails for pair $(1, 5)$, our method does not. This confirms the intuition that CNN feature descriptors have some degree of affine invariance, achieved due to multiple pooling operations. The invariance enables multiple good initial matches to be established, producing a good initial transformation estimate, which is then progressively refined using the iterative procedure.

Qualitative results for the pair $(1, 5)$ of the Graffiti benchmark are shown in Fig. 12. The figure also shows the progression of the alignment as more refinement iterations are performed. As it can be observed, the obtained alignment is qualitatively good.

TABLE 4
Evaluation on the Graffiti benchmark. Matching quality is measured in terms of the AEE (px), being the original image of 640×800 px. Our homography estimation model is run recursively five times.

Methods	Image pair				
	$(1, 2)$	$(1, 3)$	$(1, 4)$	$(1, 5)$	$(1, 6)$
DoG+SIFT [19]	0.63	1.61	2.89	fail	fail
DoG-affine+SIFT [19]	0.60	1.57	1.44	2.40	2.75
ASIFT [57]	**0.45**	**1.35**	**1.02**	**0.96**	**1.62**
Ours (5×homography)	4.17	4.81	3.17	2.55	fail

6.5 Tokyo Time Machine dataset

Qualitative results for the Tokyo Time Machine dataset [45] are shown in Fig. 14. The images have been captured at different points in time which are months or years apart. Note that, by automatically highlighting the differences (in the feature space) between the aligned images, it is possible to detect changes in the scene, such as occlusions, changes in vegetation, or structural differences e.g. new buildings being built.

7 Discussions and Ablation Studies

In this section we examine the importance of various components of our architecture, and discuss about the impact of the training set, the learned filters and the limitations of the method.

7.1 Correlation versus concatenation and subtraction.

Replacing our correlation-based matching layer with feature concatenation or subtraction, as proposed in [32] and [33], respectively, incurs a large performance drop, as shown in Tab. 5. The behavior is expected as we designed the matching layer to only keep information on pairwise descriptor similarities rather than the descriptors themselves, as is good practice in classical geometry estimation methods, while concatenation and subtraction do not follow this principle.

7.2 Normalization

Table 5 also shows the importance of the correlation map normalization step, where the normalization improves results from 41% to 48%. The step mimics the second nearest neighbor test used in classical feature matching [6], as discussed in Sec. 3.2. Note that [30] also uses a correlation layer, but they do not normalize the map in any way, which is clearly suboptimal.

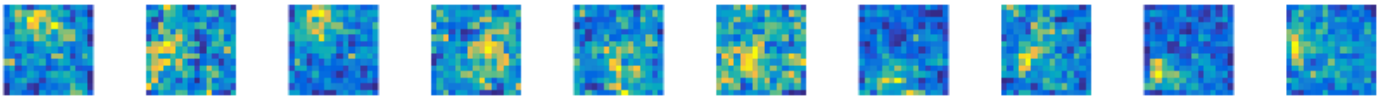


Fig. 13. **Filter visualization.** Some convolutional filters from the first layer of the regressor, acting on the tentative correspondence map, show preferences to spatially co-located features that transform consistently to the other image, thus learning to perform the local neighborhood consensus criterion often used in classical feature matching. Refer to the text for more details on the visualization.

(a) Image A (b) Image B (c) Aligned image A (d) Overlay of (b) and (c) (e) Difference map

Fig. 14. **Qualitative results on the Tokyo Time Machine dataset.** Each row shows a pair of images from the Tokyo Time Machine dataset, and our alignment along with a "difference map", highlighting absolute differences between aligned images in the descriptor space. Our method successfully aligns image A to image B despite of viewpoint and scene changes (highlighted in the difference map).

TABLE 5
Ablation studies on the matching layer and effect of the training dataset. Accuracy is measured in terms of the PCK on the PF-WILLOW dataset using the single-stage affine model.

Methods	PCK
Subtraction [33]	0.24
Concatenation [32]	0.34
Ours (without normalization)	0.41
Ours (trained on PASCAL)	0.47
Ours (trained on StreetView)	0.48

7.3 Generalization

In order to assess the influence on the performance of the trained method with respect to the training dataset used, we train a second model using PASCAL VOC 2011 [59] images, instead of StreetView images. As seen in Tab. 5, our method is relatively unaffected by the choice of training data as its performance is similar regardless whether it was trained with StreetView or PASCAL images. We also attribute this to the design choice of operating on pairwise descriptor similarities rather than the raw descriptors.

7.4 Geometric models

Different configurations of the proposed method have been analyzed, varying the geometric models and using both single-stage (as in Fig. 2) and two-stage approaches (as in Fig. 6).

Results from Tables 1, 2, and 3 show that the two-stage homography+TPS is the best performing approach, being slightly superior to affine+TPS. On the other hand, TPS alone is the best single-stage approach, but interestingly, two-iteration TPS performs worse than both affine+TPS and homography+TPS. This confirms the intuition discussed in section 4.4, where using a simpler geometric model to perform the rough alignment is expected to be more robust than using a more complex one.

7.5 What is being learned?

We examine filters of size $7 \times 7 \times 225$ from the first convolutional layer of the regressor, which operate directly on the output of the matching layer, i.e. the tentative correspondence map. We observe that two filter properties emerge from training: (i) filters specialize in detecting matches in specific positions in image A, and (ii) filters learn to mimic local neighborhood consensus for robust match estimation. In order to visualize this, each $1 \times 1 \times 225$ 1-D slice through the channels of one convolutional filter at a particular spatial location is reshaped as a 15×15 image. Recall that the 225 channels correspond to flattened similarities with image A (see Fig. 3 and Eq. (1)), therefore these images show the filter's preferences to matches in specific locations in image A. For visualization, we pick the peaks from all slices of filter weights and average them together to produce a single image. Several filters are shown in Fig. 13. It can be observed that matches form clusters, which means that spatially co-located features in image B (within the 7×7 support of the filter) respond strongly to spatially consistent locations in image A, therefore confirming that this layer has learned to mimic local neighborhood consensus. Furthermore, it can be observed that the size of the preferred spatial neighborhood varies across filters, thus showing that filters specialize for certain scale changes. Finally, the fact that the location of the highest filter weights (bright yellow) is different for different filters shows that the filters specialize for different locations in image A.

7.6 Limitations

Next, we analyze limitations of the proposed method and discuss possible ways of alleviating them.

TABLE 6
Evaluation of robustness to occlusions in terms of PCK on the PF-WILLOW dataset.

Methods	Occluded area		
	0%	10%	20%
Proposal Flow (SS, LOM) [54]	0.56	0.46	0.30
Ours (homography+TPS)	**0.60**	0.46	0.32
Ours trained with 10% occlusions (hom.+TPS)	0.57	**0.47**	0.36
Ours trained with 20% occlusions (hom.+TPS)	0.48	0.43	**0.38**

7.6.1 Robustness to occlusion

The robustness of the proposed method to occlusion was assessed by computing the PCK on the PF-WILLOW dataset when substituting a rectangular portion of each image in the dataset with a crop from a different unrelated image. Both positions and aspect ratios of these rectangles were independently and randomly sampled for each image. The results are shown in Tab. 6. It can be observed that all methods, including Proposal Flow [54], degrade significantly when occluding 10% or 20% of the area of the images. Although retraining the proposed method replicating the occlusion procedure helps to improve the performance on the occluded data, it also degrades the performance on the unoccluded case. Therefore, alignment with significant occlusion still presents a challenge for the current methods, including ours.

7.6.2 Multiple objects

Currently, the proposed method can only produce a global alignment of the image pair, and handling multiple objects is still a challenge. This is in line with current datasets on category-level image alignment which contain a single foreground object, and with all competing methods which also make this assumption. This limitation could be addressed by incorporating an attention mechanism.

7.6.3 Learning better features

Although the proposed architecture is fully differentiable, which makes it end-to-end trainable for the task of semantic alignment, we have observed that finetuning the feature extraction CNN does not improve alignment performance. This is because our synthetic dataset used for training does not contain rich appearance variations present in the real category-level alignment datasets used for evaluation. While supervision from synthetic data comes with no cost and is useful to train the regression CNN, it is not suited for learning better image features for alignment. As a solution to this problem, we have developed a combined approach using synthetic data for training the regression CNN and real data for finetuning the feature extraction CNN [60].

7.6.4 Confidence in the estimated transformation

The proposed method does not currently produce a measure of the confidence in the estimated transformation. However, the soft-inlier count presented in [60] could be employed for this purpose.

7.6.5 Asymmetry in the method

Given an pair of images I_A and I_B, the method is trained to produce the alignment in one direction only. In order to make the method more symmetric, the alignments in both directions could be estimated simultaneously and a cycle consistency loss [61], [62] could be incorporated.

7.7 Computational cost

The presented method currently takes about 1.6s per 240×240px image pair, which is $1.5\times$ faster than SIFT Flow and $6\times$ faster than Proposal Flow, when run on a modern CPU. Furthermore, the presented method can also be run on the GPU, which allows to obtain an additional $40\times$ speedup.

8 Conclusions

We have described a network architecture for geometric matching trainable from synthetic imagery without the need for manual annotations. The architecture is modular and flexible, and can be applied iteratively, in order to estimate large transformations, or in a cascade, enabling estimation of complex transformations. Thanks to our matching layer, the network generalizes well to never seen before imagery, reaching state-of-the-art results on several challenging datasets for category-level matching. The method has also proven useful for instance-level alignment, obtaining reasonable alignment for the challenging Graffiti benchmark. This work opens-up the possibility of applying our architecture to other difficult correspondence problems such as matching across large changes in illumination (day/night) [45] or depiction style [63].

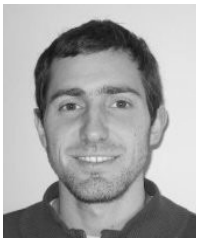
Acknowledgments

This work has been partly supported by ERC grant LEAP (no. 336845), ANR project Semapolis (ANR-13-CORD-0003), the Inria CityLab IPL, CIFAR Learning in Machines & Brains program and ESIF, OP Research, development and education Project IMPACT No. CZ.02.1.01/0.0/0.0/15_003/0000468.

References

[1] D. A. Forsyth and J. Ponce, *Computer vision: a modern approach*. Prentice Hall Professional Technical Reference, 2002.
[2] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.
[3] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski, "Building Rome in a day," in *Proc. ICCV*, 2009.
[4] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *Proc. ACM SIGGRAPH*, 2011.
[5] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," in *Proc. CVPR*, 2013.
[6] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, 2004.
[7] N. Dalal and B. Triggs, "Histogram of Oriented Gradients for Human Detection," in *Proc. CVPR*, 2005.
[8] B. Ham, M. Cho, C. Schmid, and J. Ponce, "Proposal flow: Semantic correspondences from object proposals," in *arXiv:1703.07144*, 2017.
[9] C. Schmid and R. Mohr, "Local grayvalue invariants for image retrieval," *IEEE PAMI*, 1997.
[10] J. Sivic and A. Zisserman, "Video Google: A text retrieval approach to object matching in videos," in *Proc. ICCV*, 2003.
[11] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Comm. ACM*, 1981.
[12] Y. Lamdan, J. T. Schwartz, and H. J. Wolfson, "Object recognition by affine invariant matching," in *Proc. CVPR*, 1988.
[13] B. Leibe, A. Leonardis, and B. Schiele, "Robust object detection with interleaved categorization and segmentation," *IJCV*, 2008.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, 2012.
[15] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. ICLR*, 2015.
[16] "Project webpage (code/networks)." http://www.di.ens.fr/willow/research/cnngeometric/.
[17] C. Harris and M. Stephens, "A combined corner and edge detector." in *Alvey vision conference*, 1988.

[18] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proc. ICCV*, 1999.
[19] K. Mikolajczyk and C. Schmid, "An affine invariant interest point detector," in *Proc. ECCV*, 2002.
[20] A. Berg, T. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondence," in *Proc. CVPR*, 2005.
[21] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proc. ECCV*, 2006.
[22] M. Jahrer, M. Grabner, and H. Bischof, "Learned local descriptors for recognition and matching," in *Computer Vision Winter Workshop*, 2008.
[23] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *Proc. ICCV*, 2015.
[24] X. Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, "MatchNet: Unifying feature and metric learning for patch-based matching," in *Proc. CVPR*, 2015.
[25] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proc. CVPR*, 2015.
[26] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk, "PN-Net: Conjoined triple deep network for learning local image descriptors," *arXiv preprint arXiv:1601.05030*, 2016.
[27] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han, "Large-scale image retrieval with attentive deep local features," in *Proc. CVPR*, 2017.
[28] H. Altwaijry, E. Trulls, J. Hays, P. Fua, and S. Belongie, "Learning to match aerial images with deep attentive architectures," in *Proc. CVPR*, 2016.
[29] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "DeepFlow: Large displacement optical flow with deep matching," in *Proc. ICCV*, 2013.
[30] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "FlowNet: Learning optical flow with convolutional networks," in *Proc. ICCV*, 2015.
[31] J. Thewlis, S. Zheng, P. Torr, and A. Vedaldi, "Fully-trainable deep matching," in *Proc. BMVC.*, 2016.
[32] D. DeTone, T. Malisiewicz, and A. Rabinovich, "Deep image homography estimation," *arXiv preprint arXiv:1606.03798*, 2016.
[33] A. Kanazawa, D. W. Jacobs, and M. Chandraker, "WarpNet: Weakly supervised matching for single-view reconstruction," in *Proc. CVPR*, 2016.
[34] C. Liu, J. Yuen, and A. Torralba, "SIFT Flow: Dense correspondence across scenes and its applications," *IEEE PAMI*, 2011.
[35] O. Duchenne, A. Joulin, and J. Ponce, "A graph-matching kernel for object categorization," in *Proc. ICCV*, 2011.
[36] J. Kim, C. Liu, F. Sha, and K. Grauman, "Deformable spatial pyramid matching for fast dense correspondences," in *Proc. CVPR*, 2013.
[37] J. L. Long, N. Zhang, and T. Darrell, "Do convnets learn correspondence?" in *NIPS*, 2014.
[38] E. G. Learned-Miller, "Data driven image models through continuous joint alignment," *IEEE PAMI*, 2006.
[39] F. Shokrollahi Yancheshmeh, K. Chen, and J.-K. Kamarainen, "Unsupervised visual alignment with similarity graphs," in *Proc. CVPR*, 2015.
[40] T. Zhou, Y. J. Lee, S. X. Yu, and A. A. Efros, ""FlowWeb: Joint image set alignment by weaving consistent, pixel-wise correspondences"," in *Proc. CVPR*, 2015.
[41] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proc. CVPR*, 2007.
[42] H. Azizpour, A. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "Factors of transferability from a generic ConvNet representation," *arXiv preprint arXiv:1406.5774*, 2014.
[43] A. Babenko and V. Lempitsky, "Aggregating local deep features for image retrieval," in *Proc. ICCV*, 2015.
[44] Y. Gong, L. Wang, R. Guo, and S. Lazebnik, "Multi-scale orderless pooling of deep convolutional activation features," in *Proc. ECCV*, 2014.
[45] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. CVPR*, 2016.
[46] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. CVPR*, 2009.
[47] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. ICML*, 2015.
[48] F. L. Bookstein, "Principal warps: Thin-plate splines and the decomposition of deformations," *IEEE PAMI*, 1989.
[49] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, ""Spatial Transformer Networks"," in *NIPS*, 2015.
[50] "PyTorch," http://pytorch.org/.
[51] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.
[52] T. Taniai, S. N. Sinha, and Y. Sato, "Joint recovery of dense correspondence and cosegmentation in two images," in *Proc. CVPR*, 2016.
[53] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE PAMI*, 2006.
[54] B. Ham, M. Cho, C. Schmid, and J. Ponce, "Proposal Flow," in *Proc. CVPR*, 2016.
[55] Y. Yang and D. Ramanan, "Articulated human detection with flexible mixtures of parts," *IEEE PAMI*, 2013.
[56] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "DeepMatching: Hierarchical deformable dense matching," *IJCV*, 2015.
[57] G. Yu and J.-M. Morel, "ASIFT: An Algorithm for Fully Affine Invariant Comparison," *Image Processing On Line*, 2011.
[58] K. Lebeda, J. Matas, and O. Chum, "Fixing the locally optimized RANSAC," in *Proc. BMVC.*, 2012.
[59] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results," http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.
[60] I. Rocco, R. Arandjelović, and J. Sivic, "End-to-end weakly-supervised semantic alignment," in *Proc. CVPR*, 2018.
[61] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros, "Learning dense correspondence via 3d-guided cycle consistency," in *Proc. CVPR*, 2016.
[62] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networkss," in *Proc. ICCV*, 2017.
[63] M. Aubry, B. Russell, and J. Sivic, "Painting-to-3D model alignment via discriminative visual elements," *ACM Transactions on Graphics*, 2013.



Ignacio Rocco received the MSc degree in Vision, Mathematics and Machine Learning (MVA) from École Normale Supérieure Paris-Saclay in 2016 and is currently pursuing a PhD degree in computer vision in the WILLOW research group at Inria / École Normale Supérieure, Paris. His current research is focused on the image alignment problem, investigating different trainable methods to efficiently solve this task.

Relja Arandjelović received the BA and MEng degrees from the University of Cambridge in 2009, and PhD from the University of Oxford in 2013. He then spent one year as a postdoctoral researcher at the University of Oxford, and two years as a postdoctoral researcher at Inria / École Normale Supérieure, Paris. Since 2016, he is a senior research scientist at DeepMind. His research is focused on learning from little or no labelled data, including work on multimodal learning, large scale image retrieval and place recognition, and image alignment.

Josef Sivic received MSc degree from the Czech Technical University in Prague, PhD from the University of Oxford and Habilitation degree from École Normale Supérieure in Paris. He currently holds a permanent position as a senior researcher (directeur de recherche) at Inria in Paris. He has published more than 60 scientific publications, has served as an area chair for major computer vision conferences and as a program chair for ICCV'15. He currently serves as an associate editor for the International Journal of Computer Vision and IEEE Transactions on Pattern Analysis and Machine Intelligence. Since 2014, he leads ERC project LEAP.